



Workday Prism Analytics

Product Summary

April 2024

Contents

Concept: Prism Analytics Data Management Workflow.....	7
Concept: Creating Reports to Import into Tables and Datasets.....	10
Concept: Prism Data Sources.....	11
Reference: External Data Limits.....	12
Table and Dataset Concepts.....	13
Concept: Tables.....	13
Concept: Table Error File.....	15
Concept: Datasets.....	16
Concept: Dataset Workspace.....	16
Concept: Data Catalog.....	19
Concept: Dataset Schema Changes.....	20
Concept: Dataset Pipelines.....	22
Concept: Dataset Stages.....	22
Concept: Unpivot Stages.....	23
Concept: Dataset Field Origin.....	23
Concept: Field Lineage.....	24
Concept: Table and Dataset Field Types.....	24
Concept: NULL Values in Tables and Datasets.....	28
Concept: Prism Calculated Fields.....	30
Concept: Hiding Dataset Fields.....	31
Concept: Dataset Integration Schedules.....	31
Reference: Dataset Stages.....	33
Reference: Table Error File Error Codes.....	35
Creating Tables and Datasets.....	35
Steps: Create a Table by File Upload.....	35
Steps: Create Table from a Workday Report.....	37
Steps: Create Table from an Existing Table or Dataset.....	38
Steps: Create a Table Manually.....	40
Steps: Create a Dataset with External Data (SFTP Server).....	41
Steps: Create a Dataset with External Data (Upload a File).....	44
Steps: Create a Dataset Using Workday Data.....	45
Steps: Create a Derived Dataset.....	48
Import a Table or Dataset into a Derived Dataset.....	49
Manage Dataset Integration Schedules.....	50
Upload a New File to a Dataset.....	51
View Prism Data Usage.....	52
View Table and Dataset Lineage.....	53
View Field Lineage.....	54
View Dataset Dependencies.....	55

Reference: Supported Date Formats for External Data in Tables and Datasets.....	55
Reference: Naming Guidelines.....	56
Reference: Supported File Formats for External Data in Tables and Datasets.....	58
Reference: Currency Format Requirements for External Data.....	59
Reference: Date Format Symbols.....	60
Reference: WPA_ Fields.....	61
Editing Tables.....	62
Parse External Data in a Table.....	62
Edit a Table.....	65
Reference: Table Field Attributes.....	66
Editing Datasets.....	67
Parse External Data in a Dataset.....	67
Add a Prism Calculated Field to a Dataset.....	70
Add a Stage to a Dataset.....	72
Manage Dataset Fields.....	73
Change Dataset Field Types.....	74
Convert Dataset Text Fields to Date Fields.....	75
Change the Dataset Example Rows.....	76
Concept: Dataset Example Data.....	78
Reference: Filter Stages.....	79
Reference: Group By Stages.....	80
Reference: Join Stages.....	81
Reference: Union Stages.....	83
Reference: Unpivot Stages.....	84
Reference: Boolean Expressions.....	84
Example: Unpivot Stock Vesting Data in a Dataset.....	87
Example: Bring in International-Formatted Numeric Fields.....	89
Changing Data in Tables.....	91
Create a Data Change Task.....	91
Create a Data Change Task Schedule.....	94
Concept: Data Change Tasks.....	96
Concept: Mapping Fields in Data Change Tasks.....	97
Managing Connections.....	97
Create an SFTP Connection.....	97
Securing Data in Tables and Datasets.....	100
Set Up Table Sharing.....	100
Set Up Dataset Sharing.....	102
Share a Table with Others.....	104
Share a Dataset with Others.....	105
Edit Prism Data Source Security.....	106
Concept: Security in Prism Analytics.....	108
Concept: Sharing Tables and Datasets.....	111
Concept: Relax Sharing Options.....	112
Concept: Sharing Datasets Using Relax Sharing Rules.....	114

Preparing Data for Analysis.....	122
Enable Contextual Publishing for Datasets.....	122
Publish a Dataset as a Prism Data Source Manually.....	123
Create Dataset Publish Schedules.....	123
Concept: Making Prism Data Available for Analysis.....	125
Concept: Dataset Publish Schedules.....	126
Concept: Coordinating Publish Schedules with Dataset Integration.....	128
Example: Create Dependent Publish Schedules for Datasets.....	128
FAQ: Dataset Publish Schedules.....	129
 Deleting Prism Analytics Data.....	 131
Truncate Data in a Table.....	131
Delete Rows of Data in a Table.....	132
Truncate Data in a Dataset.....	133
Unpublish a Dataset.....	134
Delete Rows from a Prism Data Source.....	134
Concept: Deleting Prism Data.....	135
 Managing Analytic Data Sources.....	 136
Install and Schedule an Analytic Data Source.....	136
Uninstall an Analytic Data Source.....	137
Concept: Workday-Delivered Analytic Data Sources.....	138
Concept: Managing Analytic Data Source Schedules.....	138
Concept: Peakon Employee Voice Analytic Data Source.....	138
 Reference: Prism Expression Language.....	 140
Concept: Prism Expression Language.....	140
Comparison Operators.....	142
Logical Operators.....	143
Arithmetic Operators.....	144
Conversion Functions.....	144
BUILD_CURRENCY.....	144
CAST.....	145
EPOCH_MS_TO_DATE.....	146
EXTRACT_AMOUNT.....	147
EXTRACT_CODE.....	147
EXTRACT_CODE_TEXT.....	148
TO_BOOLEAN.....	148
TO_CURRENCY.....	149
TO_DATE.....	150
TO_DECIMAL.....	151
TO_DOUBLE.....	151
TO_INT.....	152
TO_LONG.....	152
TO_STRING.....	153
Date Functions.....	154
DAYS_BETWEEN.....	154
DATE_ADD.....	154
EXTRACT.....	155
HOURS_BETWEEN.....	156
MILLISECONDS_BETWEEN.....	156

MINUTES_BETWEEN.....	157
SECONDS_BETWEEN.....	157
TODAY.....	158
TRUNC.....	158
YEAR_DIFF.....	160
Informational Functions.....	160
IS_VALID.....	160
Instance Functions.....	161
CREATE_MULTI_INSTANCE.....	161
INSTANCE_CONTAINS_ANY.....	161
INSTANCE_COUNT.....	162
INSTANCE_EQUALS.....	162
INSTANCE_IS_SUPERSET_OF.....	163
Logical Functions.....	164
CASE.....	164
COALESCE.....	165
Math Functions.....	165
DIV.....	165
EXP.....	166
FLOOR.....	166
HASH.....	167
LN.....	167
MOD.....	168
POW.....	168
ROUND.....	169
Text Functions.....	169
CIDR_MATCH.....	169
CONCAT.....	170
EXTRACT_COOKIE.....	171
EXTRACT_VALUE.....	171
FILE_NAME.....	172
HEX_TO_IP.....	173
INSTR.....	173
JAVA_STRING.....	174
JOIN_STRINGS.....	175
JSON_DECIMAL.....	175
JSON_DOUBLE.....	176
JSON_INTEGER.....	177
JSON_LONG.....	178
JSON_STRING.....	179
LENGTH.....	180
PACK_VALUES.....	180
REGEX.....	181
REGEX_REPLACE.....	182
REVERSE.....	184
SUBSTRING.....	184
TO_LOWER.....	185
TO_PROPER.....	185
TO_UPPER.....	185
TRIM.....	186
XPATH_STRING.....	186
URL Functions.....	188
URL_AUTHORITY.....	188
URL_FRAGMENT.....	188
URL_HOST.....	189
URL_PATH.....	190

URL_PORT.....	190
URL_PROTOCOL.....	191
URL_QUERY.....	192
URLDECODE.....	192
Window Functions.....	194
AVG.....	194
COUNT.....	196
FIRST.....	198
LAG.....	200
LAST.....	202
LEAD.....	203
MAX.....	205
MIN.....	208
RANK.....	211
ROW_NUMBER.....	213
SUM.....	216
Regular Expression Reference.....	218
Concept: Regular Expressions in Prism.....	218
Regex Literal and Special Characters.....	218
Regex Character Classes.....	219
Regex Line and Word Boundaries.....	221
Regex Quantifiers.....	221
Regex Capturing Groups.....	222

Concept: Prism Analytics Data Management Workflow

With Workday Prism Analytics, you can analyze your Workday and non-Workday data together without having to export it into a separate data warehouse and BI (business intelligence) application. This makes analysis faster, easier, more secure, and performed in a location where users can take action on it.

What are the steps involved in going from raw data (both internal and external to Workday) to visualizations and reports in Workday? What skills do you need to perform each step? This section explains each phase of the data workflow from transforming data to analyzing it.

Phase 1: Create Tables and Data Change Tasks to Bring in Data

The first phase in the data management workflow is to bring data into the Prism Analytics Data Catalog. You can bring in external data or Workday transactional data. The Data Catalog is where data analysts can see what data is available to them.

You bring in data by creating a table. A table is a Prism Analytics object that stores (materializes) data and represents it in a tabular format. A table has a schema and contains data that's valid against the schema.

After you create a table, you create a data change task to load or change data in the table.

You can also create a base dataset to bring in data. However, Workday recommends creating tables only.

Example: You create a table to contain external operational data, and then create a data change task that loads that data from a delimited file on an external server into the table. You create a schedule for the data change task to run on a recurring basis.

Example: You create a table and data change task using data in a Workday custom report.

If you're familiar with ETL workflows (extract, transform, and load), tables and data change tasks together encompass the extract logic.

Who does this phase?	Data Administrators or Data Analysts.
Where can I read more details?	Concept: Tables on page 13 Steps: Create a Table by File Upload on page 35 Steps: Create Table from a Workday Report on page 37 Steps: Create a Table Manually on page 40 Edit a Table on page 65 Concept: Data Change Tasks on page 96 Create a Data Change Task on page 91 Create a Data Change Task Schedule on page 94 Concept: Data Catalog on page 19 Concept: Creating Reports to Import into Tables and Datasets on page 10

Phase 2: Create Derived Datasets to Transform the Data

After you've brought in data into the Data Catalog, you create derived datasets based on tables and other datasets to transform, enrich, and join the data. Create a derived dataset to describe the processing logic that you need to prepare the data for analysis.

Derived datasets contain information on how to process, blend, and transform the data you import into them. Transforming data is an iterative process and typically involves creating multiple derived datasets based on other tables and derived datasets. You can:

- Create new fields by creating a Prism calculated field.
- Add 1 or more stages in a derived dataset that perform different actions, such as aggregating, joining, or filtering. Some stage types can change either the number of records or fields in the dataset.

Example: You can create a derived dataset that aggregates data into groups using a Group By stage, and then create another derived dataset based on this dataset that joins the aggregated data with another dataset using a Join stage.

Example: You can create a Prism calculated field to calculate the point of sales revenue per store.

From the perspective of an ETL workflow, the derived dataset encompasses the transform logic.

Who does this phase?	Data Administrators or Data Analysts.
Where can I read more details?	Concept: Datasets on page 16 Concept: Dataset Workspace on page 16 Steps: Create a Derived Dataset on page 48 Add a Prism Calculated Field to a Dataset on page 70 Concept: Dataset Stages on page 22 Manage Dataset Fields on page 73 Add a Stage to a Dataset on page 72 Reference: Filter Stages on page 79 Reference: Group By Stages on page 80 Reference: Join Stages on page 81 Reference: Union Stages on page 83 Reference: Unpivot Stages on page 84 View Table and Dataset Lineage on page 53

Phase 3: Apply Security to the Data

Workday applies no security to the objects inside the Data Catalog. External data doesn't have any security applied to it, and when you bring Workday data into the Data Catalog, Workday removes the existing security from the fields in the tables and datasets. As a result, any user who has access to a table or dataset in the Data Catalog can see all data in the objects they have access to.

Before you make any data in the Data Catalog available for analysis, you need to apply security to the data using the existing Workday security domains. Applying security to the data you plan to make available for analysis enables you to take advantage of Workday's strong, configurable security model for external data as well as Workday data.

You configure the data security by editing the data source security on the dataset or table, but Workday applies the security to the data in the form of a Prism data source (see next phase).

Who does this phase?	Security Administrators or Data Administrators.
Where can I read more details?	Concept: Security in Prism Analytics on page 108 Edit Prism Data Source Security on page 106

Phase 4: Make the Data Available for Analysis

You can decide which data in the Data Catalog to make available for analysis by creating a Prism data source from either a table or dataset. When Workday creates a Prism data source, it loads the data source with the data from the table or dataset, and applies the appropriate security restrictions to the data source, fields, records, and field values.

The way you create a Prism data source depends on the Data Catalog object:

- Table. Edit the table schema and select the **Enable for Analysis** option.
- Dataset. Publish the dataset. On the View Dataset Details report, select **Quick Actions > Publish**.

Workday applies the security domains configured in the data source security for the table or dataset, or it applies the *Prism: Default to Dataset Access* security domain if no data source security was configured. The *Prism: Default to Dataset Access* domain provides contextual access to a Prism data source based on your access to the underlying table or dataset.

You can use Prism data sources in visualizations and reports like any Workday delivered data source.

From the perspective of an ETL workflow, enabling a table for analysis and publishing a dataset are the load part of the process.

Who does this phase?	Data Administrators or Data Analysts.
Where can I read more details?	Concept: Making Prism Data Available for Analysis on page 125 Edit a Table on page 65 Concept: Dataset Publish Schedules on page 126 Create Dataset Publish Schedules on page 123 Publish a Dataset as a Prism Data Source Manually on page 123

Phase 5: Analyze and Visualize the Data

After a Prism data source is available, Report Writers can use it to create discovery boards and reports.

Related Information

Concepts

[Concept: Security in Prism Analytics](#)

Tasks

[Steps: Create a Table Manually](#)

[Steps: Create a Dataset with External Data \(SFTP Server\)](#)

[Steps: Create a Dataset with External Data \(Upload a File\)](#)

[Steps: Create a Dataset Using Workday Data](#)

[Steps: Create a Derived Dataset](#)

[Edit Prism Data Source Security](#)

[Publish a Dataset as a Prism Data Source Manually](#)

[Create Dataset Publish Schedules](#)

Reference

[The Next Level: Prism Analytics Community Guide](#)

[The Next Level: Move up the Workday Maturity Curve: Considerations in Defining an Analytics Strategy](#)

[The Next Level: Create a Vaccine Management Solution Using Prism and Discovery Boards](#)

[The Next Level: Estimating Prism Projects - Creating an Estimator Framework](#)

[The Next Level: Scoping a Prism Project](#)

Concept: Creating Reports to Import into Tables and Datasets

Tables and Data Change Tasks

To create a table from a Workday report, or to use a Workday report as a source in a data change task, the report must be configured:

- As an advanced report. Tables and data change tasks don't support importing other report types, such as matrix or composite.
- As a web service.
- For Prism Analytics. Select the **Enable for Prism** check box in the **Advanced** tab of the custom report.

Workday doesn't support selecting reports when the report includes:

- Fields from a related business object that have a many to 1 relationship with the primary business object.
- Prompts, except for optional prompts and required prompts with a default value set.

Note: If your custom report includes a Currency field, you must select the **Show Currency Column** check box in the **Field Options** column for the Currency field, located in the **Columns** tab of the report.

To optimize performance when running a data change task, consider these options:

- Use indexed data sources whenever possible.
- Use data sources that provide the smallest possible set of data that meets your needs. Example: If you're interested in compensation-related transactions, use employee compensation events instead of all business process transactions.

Datasets

To create a dataset from a Workday report, the report must be configured:

- As an advanced report. Tables and data change tasks don't support importing other report types, such as matrix or composite.
- As a web service.
- For Prism Analytics. Select the **Enable for Prism** check box in the **Advanced** tab of the custom report.

Workday doesn't support selecting reports when the report includes:

- Fields from a related business object that have a many to 1 relationship with the primary business object.

Note: If your custom report includes a Currency field, you must select the **Show Currency Column** check box in the **Field Options** column for the Currency field, located in the **Columns** tab of the report.

To optimize performance, consider these options:

- Use indexed data sources whenever possible.

- Use data sources that provide the smallest possible set of data that meets your needs. Example: If you're interested in compensation-related transactions, use employee compensation events instead of all business process transactions.

Related Information

Tasks

[Steps: Create Table from a Workday Report](#)

[Steps: Create a Dataset Using Workday Data](#)

Concept: Prism Data Sources

A Prism data source is a type of data source that gets its data from a Prism Analytics table or dataset. Typically, Prism data sources blend together Workday and non-Workday data.

Prism data sources are different than Workday-delivered data sources in several ways. Prism data sources:

- Are created by a colleague at your organization.
- Have their own primary business object that isn't linked to any other business object in Workday.
- Can become inactive. When a Prism data source is inactive, it exists in the tenant, but is empty and unavailable for querying in reports and discovery board visualizations.
- Can be deleted from your tenant. You can only remove a Prism data source when no reports or vizzes currently use the associated Prism data source.
- Only support some report types.
- Only support some field types.
- Only support some calculated fields. Some of the supported calculated fields provide limited functionality.

Prism data sources also have their own security. Security group access controls who can see a Prism data source.

With the appropriate permissions, you can:

- Create a viz in a discovery board using the Prism data source.
- Create a custom report using the Prism data source.
- View a Prism data source by accessing the **View Prism Data Source** report.
- Make the Prism data source inactive.
- Remove the Prism data source from the tenant.

Supported Custom Reports on Prism Data Sources

You can create these types of custom reports using a Prism data source:

- **Advanced.** Some features aren't supported, such as Subfilters. Workday only displays features that it supports. Also, only add a field from a related business object when necessary. Fields from related business objects might impact report performance.
- **Composite.** The Prism data source must include at least 1 Instance field.
- **Matrix.** Most features are supported. Lookup Prior Value isn't supported.
- **Simple.** All features are supported.
- **Transposed.** All features are supported.

Supported Calculated Fields on Prism Data Sources

You can create these calculated fields using Prism data sources:

- Arithmetic Calculation

- Build Date
- Concatenate Text
- Convert Text To Number
- Date Constant
- Date Difference
- Evaluate Expression
- Format Date
- Format Number
- Format Text
- Increment or Decrement Date
- Lookup Date Rollup
- Numeric Constant
- Substring Text
- Text Constant
- Text Length
- True/False Condition

Workday only makes available calculated fields that are supported for Prism data sources.

There are some limitations when creating some calculated fields:

- Calculated fields only work on supported field types. Example: date-related functions work only on fields of type Date.
- You can't use Instance or Multi-Instance fields in some calculated fields, such as Format Text and Substring Text. This is because Instance and Multi-Instance fields in Prism data sources only include the unique identifier information (also known as a WID), not the display name.

Related Information

Concepts

[Concept: Prism Analytics Data Management Workflow](#)

[Concept: Deleting Prism Data](#)

Tasks

[Unpublish a Dataset](#)

[Delete Rows from a Prism Data Source](#)

Reference

[The Next Level: Prism Analytics Community Guide](#)

Reference: External Data Limits

Workday enforces limits when you create a table or dataset using the REST API or the UI. The limits apply when you:

- Create a table or dataset in the UI by uploading a file. Workday creates and uses a bucket.
- Create a data change task in the UI by uploading a file. Workday creates and uses a file container.
- Run a data change task in the UI by uploading a file. Workday creates and uses a file container.
- Upload a file to a file container using the REST API.
- Create a bucket using the REST API.
- Upload a file to a bucket using the REST API.

Workday enforces these limits:

Limit	Value
Maximum size of a single file	256 MB compressed
Maximum number of buckets that can be created in a 24-hour rolling period	1,000
Maximum number of data change activities and bucket completions that can be run in a 24-hour rolling period	1,000
Maximum number of concurrent uploads using any method	10
Maximum number of files in all file containers and buckets in a 24-hour rolling period	50,000
Maximum size of all files in all file containers and buckets in a 24-hour rolling period	125 GB compressed

Related Information

Tasks

- [Steps: Create a Table by File Upload](#)
- [Create a Data Change Task](#)
- [Steps: Create a Dataset with External Data \(SFTP Server\)](#)
- [Steps: Create a Dataset with External Data \(Upload a File\)](#)
- [Upload a New File to a Dataset](#)

Table and Dataset Concepts

Concept: Tables

A table is a Workday Prism Analytics object that stores (materializes) data and represents it in a tabular format. A table has a user-defined schema and only contains data that's valid against the schema. The data in tables is backed by a distributed columnar data store.

You create tables to bring in data from multiple sources and store it in a central location, the Data Catalog (similar to a data warehouse). You can then join, transform, blend, and enrich table data using derived datasets based on the table. Use derived datasets to prepare data for analysis.

Tables Compared to Base Datasets

A table is similar to a base dataset with important differences.

Concept	Tables	Base Datasets
Schema	You define the table schema before you load data into the table. If you're familiar with databases, this is commonly referred to as schema-on-write. You can still change the table schema, but there are requirements and limitations. Example: You can add/remove fields at any time, but if you want to change the field type, the table must be empty (zero rows). Schema changes	Base datasets control underlying data stored in files. Dataset schemas describe how to read (and later transform) the data stored in those files. If you're familiar with databases, this is commonly referred to as schema-on-read. Although you can change the dataset schema at any time, you must make sure that the schema matches the data in the underlying files so that it recognizes

Concept	Tables	Base Datasets
	are destructive. Example: if you remove a field, you lose all data contained in the field.	the data correctly. If the schema doesn't match the file, then Prism sets value as null.
Source type	Tables can accept data from any type of source at any time, such as file upload, a dataset, or REST API. Create data change tasks to load data into and change data in a table. You can use a different source type for each data change task that works on the same target table. Example: You create 2 data change tasks that use the same target table and these source types: • A delimited file that inserts data. • A derived dataset that updates data.	A base dataset only accepts data from the same source type that you used when you created the dataset. That is, if you create a dataset using SFTP, it will only accept data from an SFTP server.
Data validation	When you load data into a table, Workday validates the data against the defined schema. If the value for a field doesn't match the field type or other field parameters (such as date format), then Workday marks the entire row as invalid and doesn't include the row in the table. Instead, the row is sent to an error file that you can download.	When you publish a dataset, Workday reads the data stored on disk and validates it against the current schema of the base dataset. If the value for a field doesn't match the field type or other field parameters (such as date format), then Workday marks that field value as NULL and includes the row in the published Prism data source.
NULL handling	Every field allows NULL values unless you configured it as required. Workday distinguishes between NULL values and empty string values in Text fields when reading a delimited source file to load data into a table. If a delimited file contains: • 2 consecutive field delimiters only, then Workday treats the field value as NULL. Example: "Smith", , "Tom" • 2 consecutive field delimiters with the 2 consecutive quote characters in between, then Workday treats the field value as an empty string. Example: "Smith", "", "Tom"	Every field allows NULL values. Workday doesn't distinguish between NULL values and empty string values in delimited files.
Name restrictions	Tables have 2 names: • Name. This is a display name that displays in the Data Catalog. You can change the display name.	Base (and derived) datasets have 2 names: • Name. This is the name that displays in the Data Catalog. You can change the display name. Publishing

Concept	Tables	Base Datasets
	API Name. This is a unique name used to reference the table in the API. You can't change the table API name after you create the table. Table fields also have a display name and an API Name. You can change the display name at any time. However, you can't change the API name after you create the field and save the table. Display, API, and field API names must be unique and conform to the name validation rules. See Reference: Naming Guidelines on page 56 for more details.	renamed datasets won't affect reports or discovery boards that use the original names. API Name. This is a unique name used to reference the dataset in the API. You can't change the dataset API name after you create the dataset. Dataset fields only allow 1 name and you can change it at any time. But you must ensure to fix any downstream errors that might result from changing the field name. Display and API names must be unique and conform to the name validation rules.
Deleting data	You can remove all rows (truncate) or some rows (delete) from a table. You can selectively delete rows based on a key field. You might want to do this when you append data to the table and you need to remove the rows added from a particular load.	You can remove all rows (truncate) from a base dataset. You can't delete a subset of rows.
Make Prism data source	Use the Enable for Analysis option when you create or edit the table schema to create a Prism data source.	Publish the dataset to create a Prism data source.
Row count	Workday knows exactly how many rows of data exist in a table, and it displays the number of rows in the Data Catalog and on the View Table Details report.	Workday doesn't know how many rows of data exist in a base dataset.

Related Information

Concepts

[Concept: Datasets](#) on page 16

Reference

[2020R1 What's New Post: Prism Analytics Tables](#)

[Reference: Naming Guidelines](#) on page 56

Concept: Table Error File

When you load data into a table, Workday validates the data against the defined schema.

If the value for a field doesn't match the field type or other field attributes (such as date format), then Workday marks the entire row as invalid and doesn't include the row in the table. Instead, Workday sends the row to an error file that you can download.

Use the error file to get a list of all rows that failed to load into the table. You can fix the errors in the data, remove the extra fields that Workday adds, and load the fixed data into the table.

The error file:

- Is a CSV file.
- Includes all fields defined in the table schema plus fields for troubleshooting:
 - Error Code
 - Error Message
- Includes all failed rows up to a maximum of 10,000 rows. If there are more than 10,000 error rows, then Workday rejects the load with a status of Failed.

To download the error file, access either:

- The **Activities** tab on the **View Table Details** report.
- The **Prism Activities Dashboard** from the **Data Catalog**.

Click the download icon for an activity that included some errors.

Related Information

Reference

[Reference: Table Error File Error Codes](#) on page 35

Concept: Datasets

A dataset is a Prism Analytics object that controls some underlying data and describes some processing logic to manipulate the data. A dataset is a description of the data, otherwise known as metadata. It contains all of the data about the data, plus a subset of example rows to help you understand the data values. You create datasets to prepare data for analysis.

A dataset can describe either Workday or non-Workday (external) data. You might want to create a dataset using external data to blend, transform, and enrich it with Workday data. This enables you to analyze your Workday and non-Workday data together without having to export it into a separate electronic data warehouse and business intelligence (BI) application.

A dataset can also describe data that's output from another table or dataset by deriving the dataset from an existing table or dataset.

You can create a:

- **Base dataset.** A base dataset is a dataset that is based on 1 or more source files. The source files can come from external sources (non-Workday data) or from the output of a Workday custom report. You create a base dataset when you create a new dataset using these options:
 - **from File.** This option creates a dataset with external data that you upload in the browser.
 - **from SFTP.** This option creates a dataset with external data that Workday retrieves from an SFTP server using an integration.
 - **from Custom Report.** This option creates a dataset with Workday data.
- **Derived dataset.** A derived dataset is a dataset that is based on 1 or more existing tables or datasets. The source data of a derived dataset comes from the output of existing tables and datasets. You use derived datasets to blend data together from different sources, such as Workday data and non-Workday data. Some stage types, such as Join and Union, are only available to derived datasets. You create a derived dataset when you create a new dataset using the **Derived Dataset** option.

Concept: Dataset Workspace

The **Edit Dataset Transformations** task is where you make changes to the dataset definition to manipulate data. This task acts like a single page application, meaning you can interact with elements on the task dynamically as if it were its own application. You can also make all the changes you want, such as adding Prism calculated fields and stages. When you're done, click **Save** to save all changes to the dataset.

If you have permission to edit a dataset, you can access the **Edit Dataset Transformations** task using these methods:

- Right-click the dataset name on the **Data Catalog** report and select **Edit Transformations**.
- Select **Edit Transformations** from the **Quick Actions** on the **View Dataset Details** report.
- Access the **Edit Dataset** task and select the dataset name that you want to edit.
- When creating a dataset for the first time, the workflow leads you to the **Edit Dataset Transformations** task.

When you view the **Edit Dataset Transformations** task, you see these components:

The screenshot displays the 'Edit Dataset Transformations' task for the dataset 'DDS lemonade sales'. The interface is divided into several sections:

- Top Bar:** Includes a search bar and user profile.
- Left Sidebar (1):** Contains a 'Change Pipeline' button and a list of pipelines: 'Import [Stage ID 1] TBL lemonade sales Import' and 'Manage Fields [Stage ID 2] 1 Calculated Field Manage'.
- Central Stage Editor (2):** Shows 'Stage ID 2 - Manage Fields' with a table of 39 rows. The table columns are: Hot_D..., Date, Total_Sa..., Cost_Center, Lemons_U..., and Sales in U... The table contains 39 rows of data, including dates from 06/26/2... to 07/24/2... and sales values in USD.
- Right Panel (4):** Displays 'Field Info' for the 'Sales in USD' field. It shows the field type as 'Currency', the expression function as 'BUILD_CURRENCY([Total_Sales], 'USD')', and the field origin as 'There are no previous fields'.

1. Pipeline list. (Derived datasets only.) Click **Change Pipeline** to view a collapsible panel that lists all tables and datasets that you've imported into the derived dataset. Importing a table or dataset creates a new pipeline. You can also use this panel to add a new pipeline by importing an additional table or dataset. When you select an item in the pipeline list, the pipeline details panel displays the details for that pipeline.
2. Pipeline details panel. This panel displays every stage in the dataset pipeline, starting with the first stage that created the pipeline. For base datasets that can be an Import or Parse stage depending on where the source data comes from. For derived datasets, it's an Import stage. You can:
 - Add, edit, and delete stages.
 - Manage dataset fields.
 - View stage descriptions.
 - View the number and names of Prism calculated fields in a stage.
 - Collapse this panel to increase the available space in the example data table.
3. Example data table. The example data table takes up most of the space on the **Edit Dataset Transformations** task. It displays the current view of the data (records and fields) for the output of the currently selected pipeline stage.

4. Inspector panel. This panel displays when you select a field in the example data table. You can hide this panel to increase the available space in the example data table. This panel has these tabs:
 - Field Info. This tab displays detailed information about the selected field, including statistics on the values in the field. All statistics are based on the data currently shown in the example data table. Therefore, to get more precise numbers, increase the number of example rows.
 - Functions Library. This tab displays the functions that you can use in a Prism calculated field expression, including description, syntax, and an example. You can search for a specific function. You can click the + icon next to the function name to insert the function at the current location in the Prism calculated field expression.
5. Prism calculated field expression bar. Click **Add field** to add a new Prism calculated field, and then enter the field expression. You can:
 - Use the expression bar later to edit the field expression.
 - Expand the expression builder to create and view multiline expressions.
 - Edit the field name in the inspector panel.
6. Stage statistics and search bar. For a selected stage, you can see the number of:
 - Fields
 - Prism calculated fields
 - Field-related errors

You can see the ID for stages. The stage ID is a unique ID that Workday assigns to the stage based on the order you added the stage to the dataset. For each stage you add, the stage ID increases.

If you use Microsoft Excel, there are potential limits on the number of characters your cells can contain when you download all values.
7. Download example data.
8. Search for a field and navigate to it directly.
9. Example data controls. Use this menu to filter the example data displayed in the dataset.
10. Table and list view. You can view the example data table as a table or a list. The field list view navigator enables you to see distinct, null, median, and top values at a glance. When you edit a new dataset, the default view is table view. Each time you change the view, the last view you select becomes the default view.
11. Edit dataset details. Use the configuration icon to open a pop-up where you can:
 - Change the dataset display name.
 - Create and edit tags.
 - Edit the dataset description.
 - View the dataset API name.
12. Dataset actions menu. Use this menu to quickly access the **View Dataset Details** and **View Dataset Lineage** reports.

Related Information Concepts

[Concept: Dataset Stages](#) on page 22

[Concept: Dataset Pipelines](#) on page 22

[Concept: Prism Expression Language](#) on page 140

Tasks

[Change the Dataset Example Rows](#) on page 76

Reference

[2021R1 What's New Post: Dataset Viewer](#)

Concept: Data Catalog

The **Data Catalog** report is your starting point for using Workday Prism Analytics. The report displays the data available to you. If you have the appropriate permissions, you can:

- Bring in and store data from multiple sources using either tables or base datasets.
- Create derived datasets to transform the data.
- Create data change tasks to change data in a table using data from a source.

You can also:

- Customize your view of datasets and tables by filtering them.
- See details for a selected object, such as a description, tags, or dependencies.
- Access details about reportable rows and Prism usage.
- Access documentation that supports your use of Prism Analytics using the **Help** button.

When you view the **Data Catalog** report, you see these components:

The screenshot shows the Workday Prism Analytics Data Catalog interface. It features a left sidebar (1) with navigation options: All Data, Tables, Datasets, Connections, Data Change Tasks, Data Change Activities, and Dataset Activities. The main area (2) displays a table of data objects. The table has columns for Name, Type, Recent Activity, and Last Modified. Two objects are listed: 'DDS Lemonade Sales' (Derived Dataset) and 'TBL Lemonade Sales' (Table). The 'DDS Lemonade Sales' object is selected, and its details are shown in a right-hand panel (3). The details panel includes sections for Details, API Name, Description, Last Modified By, Data Source Security, Last Successful Publish, Dependencies, Imports, and Fields. The 'Fields' section shows a search bar and a list of fields: 'Cost_Center' and 'Date'. The top header (4) contains a search bar and a 'Filter by Tags' button (6). A 'Remove all filters' button (7) is also present.

1. Data Catalog details panel. From this panel, you can:

- Create datasets and tables.
- Create data change tasks.
- Create connections.
- View dataset publishing activities in the **Dataset Activities** tab.
- View all Prism-related activities in the **Prism Activities Dashboard**.
- View data change activities.
- View Prism usage details.
- Open the Prism Analytics Data Management User Guide in a new browser tab. The entire documentation is also available in the Workday Community if you have access to Community.

2. Main Data Catalog view. This view displays all the data that you have permission to view. Right-clicking a dataset or table displays a list of actions that you can perform.

3. Inspector panel. This panel displays when you select an object in the Data Catalog. You can hide this panel to increase the available space in the main Data Catalog view. This panel displays detailed information about the selected object, such as if a dataset is **Enabled for Analysis**. If enabled (published), you can create reports and visualizations with the dataset. If enabled and outdated, the dataset is published but has changed since it was last published.

You can also view tags assigned to a dataset or table from the inspector panel. Clicking a tag in the panel displays the datasets or tables filtered by the tag.

4. Search for objects to filter the list of objects in the Data Catalog.
5. The number of objects displayed.
6. Filter by tags.

You can add or edit tags to organize your data if you have editor permission when you:

- Create a dataset by file upload.
- Create a derived dataset.
- Create a table.
- View a dataset or table.
- Edit a table.

Note: Tags you create are visible to all.

If you remove a tag from a dataset or table, the tag still displays in the tag menu if another dataset or table uses the tag.

If you change browsers or laptops, the tag filters you select in the **Data Catalog** report won't persist.

Workday doesn't store tag names but rather a randomly generated identifier for each tag. This is consistent with how Workday stores other filters in Data Catalog.

7. Filter the **Data Catalog** report based on when objects were last modified.

Workday automatically applies a **Last 7 days** filter that you can edit or delete.

Note: If you delete the filter, Workday will reapply it when you change sessions.

If you edit the filter without deleting it, Workday persists those changes to other sessions.

Workday provides the **Prism Datasets and Tables** data source to help you create reports and discovery boards about Prism Analytics tables and datasets. Although this is an indexed data source, none of the fields in it are indexed. As a result, it behaves like a standard data source.

Concept: Dataset Schema Changes

Schemas are the structures behind how your data is organized. A dataset might consist of multiple schemas that you define when creating the dataset. Each of these dataset components has its own schema:

- The files containing external data that you upload into a dataset.
- The custom report you create to bring Workday data into a dataset.
- The input to a pipeline stage. For the first stage in a base dataset, this schema is determined by the dataset source.
- The output from a pipeline stage. For the last stage in the Primary Pipeline, this schema determines the output schema of the entire dataset.

Schema changes can happen anywhere in a dataset. Schemas change when you add new fields or remove existing fields, such as when you:

- Add a Prism calculated field.
- Hide or expose a field.
- Add a stage that adds or deletes fields in that stage.

Schemas can also change when you:

- Upload an external file.
- Import a file from an SFTP server.
- Import data from a custom report.

Where changes occur in a dataset can affect other dataset components. Changes to dataset schemas can break components that were based on the original schema. Where changes occur can also affect how you manage changes.

Note: When you change the schema and Workday imports the new source file into the dataset, you must open the dataset and save the changes. If you don't save the dataset, Workday continues to use the old schema definition. The next time you publish the dataset, you might get inconsistent data in the Prism data source.

When you import data that changes the schema of your dataset from an SFTP server or custom report, you must manually edit the dataset to incorporate the changes. If new fields are added to the dataset, Workday hides them. Expose the new fields and save the dataset to include the fields in the dataset schema.

Workday recommends that you ensure that the dataset is up to date and includes the fields you want. In most cases, you use the Manage Fields stage to manage dataset schema changes.

Manage Fields

The Manage Fields stage uses the output of the previous stage as a baseline from which to monitor changes. When this baseline changes, Workday warns you in the pipeline details panel. If the baseline changes, in the Manage Fields stage, Workday:

- Displays all new and removed fields.
- Doesn't yet include all changes in the dataset schema.
- Displays no data in the example table.

Workday recommends that you:

- Add a Manage Fields stage at the beginning of the Primary Pipeline of a derived dataset when you want to monitor the schema of the table or dataset from which it's derived.
- Add a Manage Fields stage at the end of the Primary Pipeline of a dataset that you intend to publish. This enables you to detect any schema changes that might break reports that use the Prism data source of this published dataset.
- Add a Manage Fields stage at the end of a pipeline when you need to hide fields or expose new fields.
- Add a Manage Fields stage in a base dataset to ensure that no future integration can unintentionally remove an existing field.
- Include no more than 2 Manage Fields stages in a single pipeline.

In some cases, Workday handles schema changes without using a Manage Fields stage. Example: When uploading a new version of a source file into a dataset, Workday handles schema changes based on the source file header row:

Source File Contains Header Row	Result
Yes	Workday uses the field names in the header row of the dataset to determine which fields are in the new source file. You can add or delete fields anywhere in the source file. If the field names in the source file don't change, Workday updates the schema of the dataset into which you import the new source file.

Source File Contains Header Row	Result
No	Workday handles changes that occur only at the end of the source file. You can only add new fields at the end of the source file. If you do so, Workday updates the schema of the dataset into which you're importing the new source file.

Related Information**Tasks**

[Manage Dataset Fields](#) on page 73

[Steps: Create a Dataset with External Data \(SFTP Server\)](#) on page 41

[Steps: Create a Dataset Using Workday Data](#) on page 45

Reference

[The Next Level: Prism Analytics Best Practices](#)

Concept: Dataset Pipelines

Datasets contain 1 or more pipelines. A pipeline is a container of stages that models the flow of how data should be transformed. It consists of an ordered list of stages, each of which define how to modify the data at that point in the pipeline. Pipelines can contain 1 or more stages.

Base datasets contain 1 pipeline, and derived datasets can contain 1 or more pipelines. Every dataset has a Primary Pipeline.

The first stage in a pipeline brings in data from the dataset source. Stages listed after the first stage in a pipeline take the output of the previous pipeline as the input to the current stage. If you're familiar with ETL workflows (extract, transform, and load), each stage is 1 step in a development pipeline.

The last stage of any pipeline is the output for that pipeline. The output of the Primary Pipeline is the output of the entire dataset. Therefore, when you publish a dataset, the output of the Primary Pipeline will be materialized as the data in the Prism data source.

Related Information**Tasks**

[Add a Stage to a Dataset](#) on page 72

Concept: Dataset Stages

A stage is a pipeline object that takes in data, transforms it in some way, and outputs modified data. A stage performs a single computational function, such as joining data with a Join stage or parsing data from the dataset source with a Parse stage.

Prism Analytics supports these stage types:

- Import
- Parse
- Manage Fields
- Filter
- Group by
- Join
- Union
- Unpivot

Adding a stage can change the number of records and fields in the dataset.

Deleting a Union or Join stage disconnects but keeps the stage's pipelines. If you don't want to keep the disconnected pipelines, delete or use them in another Join or Union stage in the Primary Pipeline.

You can't delete or add an Import stage or a Parse stage.

Related Information

Tasks

[Add a Stage to a Dataset](#) on page 72

Reference

[Reference: Dataset Stages](#) on page 33

Concept: Unpivot Stages

The Unpivot stage enables you to arrange the data in derived datasets in a way that is more meaningful to you. You can convert fields (columns) to rows. The unpivot process consolidates data from 2 or more similar fields into a pair of new fields:

- A field created from the original field names (referred to as Input Fields)
- A field created from the original field values (referred to as Output Values)

When you unpivot:

- Workday populates the new fields with the values you specified for the unpivot.
- Workday repeats row values in the input fields you didn't include in the unpivot.

You can:

- Apply the unpivot to all field types.
- Rename your output values.
- Specify names for your pairs of new fields.
- Unpivot up to 150 input fields within each Unpivot stage.
- Create multiple pairs of new fields within each Unpivot stage.
- Add or delete an Unpivot stage anywhere in pipelines of a derived dataset.

Workday recommends that you create multiple pairs of new fields in a single Unpivot stage rather than 1 pair of new fields in many Unpivot stages.

Related Information

Reference

[Reference: Unpivot Stages](#) on page 84

[2020R2 What's New Post: Unpivot Stage](#)

Examples

[Example: Unpivot Stock Vesting Data in a Dataset](#) on page 87

Concept: Dataset Field Origin

You can see where a dataset field originates in pipelines for derived datasets. When tracing a field's origin, Workday searches for the location where the field is first introduced. Workday displays the most upstream field that you have permission on. To see the first occurrence of a field, you must have permission to view or edit transformations on all the datasets in the field's lineage including its first occurrence.

Examples of original fields include:

- Fields in base datasets created from the Dataset REST API.
- Fields in the Parse stage.
- Fields in tables.

- Fields in the Unpivot and Group by stages if the field is created within the stage.
- Output fields in the Union stage.
- Prism calculated fields.

Related Information

Tasks

[View Field Lineage](#) on page 54

Concept: Field Lineage

You can see the lineage of a dataset field and all transformations involving the field across different datasets. You can also trace the lineage of a calculated field and all dependent fields used in it. When tracing the lineage of a field, Workday displays the stages involving the field across all datasets or tables that you have permission on. Specific stage information for each stage is displayed in the inspector panel.

Related Information

Tasks

[View Field Lineage](#) on page 54

Reference

[2022R2 What's New Post: Field Lineage for Prism Analytics](#)

Concept: Table and Dataset Field Types

Each table and dataset field has a field type attribute. The field type is often referred to as a data type in other data applications.

The field type determines:

- What kind of values the field can hold.
- Which functions can use the field as an argument. Example: The CONCAT function only accepts Text fields as arguments.

When you create a table, you define the field type of each field. When you load data into a table from a delimited file, Workday validates the data against the defined schema. If the value for a field doesn't match the field type or other field parameters (such as date format), then Workday marks the entire row as invalid and doesn't include the row in the table. Instead, Workday sends the row to an error file that you can download.

When you create a dataset using external data, Workday attempts to guess the field type of each field by examining some of the data. However, you can change the field type if Workday assigned the wrong field type or if you want to apply a different field type.

Note: If you change the input field type in the last stage before a Manage Fields stage, the new field type must be compatible with the changed field type.

When you create a dataset from a custom report, Workday maps the Workday field types to dataset field types.

For Prism calculated fields, the expression result determines the field type for that field. If a Prism calculated field expression is `TO_INT(zipcode)`, then the field type for that field is Integer.

Note: Workday recommends using the Numeric field type in datasets where possible. When you use a different numeric field type, such as Double, you risk losing precision and getting erroneous results depending on the data and calculations in the dataset. In base datasets, set the field type as Numeric(x,y). You can also change the field type in a derived dataset using the CAST function. Example: `CAST([salary] AS decimal(12,4))`

Table and Dataset Field Types

Tables and datasets use these field types:

Table or Dataset Field Type	Description	Range of Values
Boolean	A Workday specific field type that contains boolean values.	True, False
Currency	A variable length decimal value that supports a maximum of 20 digits before the decimal point and a maximum of 6 digits after the decimal point, combined with a Workday-recognized, 3-digit currency code.	Currency values can represent any valid positive or negative value given the specified number of digits before and after the decimal point.
Date	Date combined with a time of day with fractional seconds based on a 24-hour clock.	Date range: January 1, 1753, through December 31, 9999 Time range: 00:00:00 through 23:59:59.997 Internally, Workday stores all Date type data in UTC format (coordinated universal time). If you bring in external data with time zone information, Workday converts the data to UTC. If you bring in external data with no time zone information, Workday doesn't convert the data and stores it as UTC.
Double	Double-precision 64-bit floating point number.	4.94065645841246544e-324d to 1.79769313486231570e+308d (positive or negative)
Instance	A Workday-specific field type that contains Workday Instance values. Each field also retains information about the business object the Instance field is based on. Instance fields represent a 1-to-1 relationship between 2 objects.	A hexadecimal value that references a WID.
Integer	32-bit integer (whole number).	-2,147,483,648 to 2,147,483,647
Long	64-bit long integer (whole number).	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Multi-Instance	A Workday-specific field type that contains a set of (zero or more) Workday Instance values. Each field also retains information about the business object the Instance values are based on. Multi-Instance fields represent a	Zero or more hexadecimal values that reference a WID.

Table or Dataset Field Type	Description	Range of Values
	1-to-many relationship between 2 objects.	
Numeric	A variable length numeric value that supports a total of 38 digits before and after the decimal point, but a maximum of 18 digits after.	Numeric values can represent any valid positive or negative value given the specified number of digits before and after the decimal point. All Numeric fields have an associated number of digits before and after the decimal point (similar, but not identical, to precision and scale). When you create a table or dataset from a Workday report, Workday assigns the digit values to the Numeric field. When you create a table or dataset from external data, you can specify the digit values for Numeric fields. Example: <code>Numeric(19, 2)</code> can store any value from 0.00 to 9,223,372,036,854,775,807.99 and <code>Numeric(19, 0)</code> can only store whole numbers from 0 to 9,223,372,036,854,775,807.
Text	Variable length non-Unicode text (also known as string) data.	Maximum text length of 2,147,483,647 characters.

Mapping Report Field Types to Table and Dataset Field Types

When you create a table or dataset from a custom report, Workday:

- Creates a field in the table or dataset for every report field that uses a supported field type.
- Assigns a field type to the fields in the table or dataset.

Workday only retains report fields that have field types that tables and datasets support. Workday doesn't include any field that uses an unsupported table or dataset field type.

After creating a table or dataset from a custom report, verify that the assigned field types are correct for your table or dataset needs. Example: You might need to change a field type from Numeric to Integer to match the field type when joining the dataset with another dataset.

This table explains how a report field type maps to a table or dataset field type when you create a table or dataset.

Report Field Type	Table or Dataset Field Type	Notes
Numeric	Numeric, Integer, or Long	The field type in the table or dataset depends on the number of digits before and after the decimal point in the report:

Report Field Type	Table or Dataset Field Type	Notes
		- Integer. Zero digits after, and less than or equal to 9 digits before the decimal point. - Long. Zero digits after, and greater than 9 digits and less than or equal to 18 digits before the decimal point. - Numeric. All other Numeric values. For Numeric fields, Workday assigns the number of digits before and after the decimal point based on the values in the custom report.
Text	Text	
Rich Text	Not supported	
Date	Date	
DateTimeZone	Not supported	
Time	Not supported	
Currency	Currency	
Boolean	Boolean	
Multi-Instance	Multi-Instance	Tables and datasets only include Multi-Instance report fields that are located in: - The primary business object. - A related business object that has a 1-to-1 relationship with the primary business object.
Single Instance	Instance	
Self-Referencing Instance	Instance	

Mapping Table and Dataset Field Types to Prism Data Source Field Types

When you enable a table for analysis or publish a dataset, Workday creates a Prism data source.

This table lists the field types Workday uses in a Prism data source:

Table or Dataset Field Type	Prism Data Source Field Type	Notes
Text	Text	
Boolean	Boolean	
Date	Date	Workday doesn't include any time information in the Date field in the Prism data source. Workday deletes any time information in

Table or Dataset Field Type	Prism Data Source Field Type	Notes
		the Date field when it creates the Prism data source.
Currency	Currency	
Integer	Numeric	
Long	Numeric	
Double	Numeric	
Numeric	Numeric	Workday writes Numeric fields with a maximum of 20 digits before and 6 digits after the decimal point. If a Numeric field contains a value that exceeds these maximums, then Workday rounds down the value when it creates the Prism data source.
Instance	Single Instance	Workday assigns the business object name associated with the Instance field in the dataset.
Multi-Instance	Multi-Instance	Workday assigns the business object name associated with the Multi-Instance field in the dataset.

Related Information

Tasks

[Change Dataset Field Types](#) on page 74

Reference

[Reference: Currency Format Requirements for External Data](#) on page 59

[The Next Level: Prism Analytics Best Practices](#)

Concept: NULL Values in Tables and Datasets

If a field value in a table or dataset is empty, it's considered a NULL value. When you enable a table for analysis or publish a dataset, Workday replaces all NULL values with a default value in the Prism data source. Prism data sources, discovery boards, and reports have no concept of NULL values.

How Tables Process NULL Values

A value can be NULL because the raw data from the source is missing values for a particular field.

How Datasets Process NULL Values

A value can be NULL for these reasons:

- The raw data from the source is missing values for a particular field.
- The raw data from an external data source can't be parsed for the specified field type.
- A Prism calculated field expression returns an empty or invalid result.

- The dataset contains a Join stage and:
 - You configure an outer join (you include all rows from at least one of the dataset pipelines).
 - There's no match from 1 pipeline to the other.

If you're familiar with SQL, this is similar to an unjoined foreign key.

Default Values by Field Type

Workday replaces NULLs with these values in a Prism data source:

Field Type	Value in Prism Data Source
Boolean	False
Currency	0 (with an empty currency code)
Date	(Blank) Note: If the date value in a table that is enabled for analysis is 0000-12-31T23:59:59.999Z, then Workday displays inconsistent result.
Numeric, Double, Integer, Long	0
Instance	(Blank)
Multi-Instance	(Blank)
Text	(Blank)

NULL Values in Prism Calculated Fields

The way datasets treat NULL values in Prism calculated fields is consistent with SQL standard. That means:

- Arithmetic calculations on numeric fields that involve a NULL return NULL. Example: 5 + NULL returns NULL.
- Comparison operations that result in a Boolean field that involve a NULL return NULL. Example: 5 > NULL returns NULL.

Currency fields have 2 components to the field value (the currency code and currency value), and as a result, they handle NULLs a little differently than numeric fields in arithmetic calculations.

- Addition and subtraction calculations on Currency fields that involve a NULL return NULL. Example: TO_CURRENCY("5.00 USD") + NULL returns NULL.
- Addition and subtraction calculations on Currency values that use different currency codes return NULL. Example: TO_CURRENCY("5.00 USD") + TO_CURRENCY("5.00 EUR") returns NULL.

You can't use Currency fields in comparison operations. However, you can test for NULL values by using this syntax:

```
value IS NULL
value IS NOT NULL
```

NULL Values in Group By Stages

When you create a Group By stage, you select a dataset field for grouping and another field for summarizing. How Workday treats NULL values depends on how you use the field:

- For grouping fields, any NULL values result in their own group.
- For summarization fields, NULL handling depends on the summarization type.

Summarization Type	Description
Average	Returns the average of all valid numeric values in the group. It sums all values in the provided expression and divides by the number of valid (not NULL) rows. If a Currency field contains multiple currency codes, the result is NULL.
Count	Returns the number of all rows in a group (counts all values, both NULL and non-NULL). If a Currency field contains multiple currency codes, the result is NULL.
Maximum	Returns the greatest of all non-NULL values, and NULL if all values are NULL. If a Currency field contains multiple currency codes, the result is NULL.
Minimum	Returns the lowest of all non-NULL values, and NULL if all values are NULL. If a Currency field contains multiple currency codes, the result is NULL.
Sum	Returns the total of all non-NULL values, and NULL if all values are NULL. If a Currency field contains multiple currency codes, the result is NULL.

Related Information**Concepts**

[Concept: Table and Dataset Field Types](#) on page 24

Reference

[Reference: Currency Format Requirements for External Data](#) on page 59

Concept: Prism Calculated Fields

A Prism calculated field is a user-created field that generates its values based on a calculation or condition, and returns a value for each input row. Values are computed based on expressions that can contain values from other fields, constants, mathematical operators, comparison operators, or built-in row functions.

Use Prism calculated fields to:

- Derive meaningful values from base fields, such as calculating someone's age based on their birthday.
- Do data cleansing, such as substituting 1 value for another.
- Compute new data values based on a number of input variables, such as calculating a profit margin value based on revenue and costs.

Sometimes you need several steps to achieve the result that you want. You can use the result of a Prism calculated field in the expressions of other Prism calculated fields, enabling you to define a chain of processing steps.

When you create a Prism calculated field, the inspector panel displays for the new field. To see a list of available functions, click the **Functions Library** tab in the inspector panel.

You can extract and export the expressions used in dataset pipelines as CSV files from the **View Dataset Lineage** report.

Related Information Concepts

[Concept: Prism Expression Language](#) on page 140

Tasks

[Add a Prism Calculated Field to a Dataset](#) on page 70

Reference

[The Next Level: Prism Analytics Best Practices](#)

Concept: Hiding Dataset Fields

A data administrator can control what fields of a dataset are visible. Hidden fields aren't visible further along in the development process. Example: if you hide a field in a stage in a pipeline, that field isn't visible in a later stage. Hidden fields aren't included in the Prism data source of a published dataset, or in an imported dataset of a derived dataset.

You might decide to hide a field to:

- **Protect sensitive data.** In some cases, you might want to hide fields to protect sensitive information. You can hide detail fields, but still allow access to summary information. Suppose that you have a dataset containing employee salary information. You might want to hide salaries per person, but still enable analysts to view average salary by department.
- **Hide unpopulated or sparse data fields.** You might have fields in your raw data that didn't have any data collected. The data collected might be too sparse to be valid for analysis. Suppose that a web application has a placeholder field for comments, but it was never implemented on the website so the comments field is empty. Hiding the field prevents analysts from using a field with mostly null values when they analyze the data.
- **Use a calculated field instead of the fields that it's based on.** You might add a Prism calculated field to transform the values of the raw data. You want your users to analyze the transformed values, not the raw values. Suppose that you have a "return reason code" field where the reason codes are numbers (1, 2, and 3). You could transform the numbers to the actual reason information (such as Didn't Fit, Changed Mind, and Poor Quality), so the data is more usable during analysis.
- **Hide Prism calculated fields that do interim processing.** As you work on your dataset to cleanse and transform the data, you might need to add interim Prism calculated fields to achieve a final result. These fields are necessary to do a processing step, but aren't intended for final consumption. You can hide these working fields so they don't clutter later stages or the dataset details.

Hide a field in the Manage Fields stage by unselecting the check box for a field. Although you can also hide a field in the inspector panel, it's a best practice to hide fields in the Manage Fields stage.

Related Information Concepts

[Concept: Prism Analytics Data Management Workflow](#) on page 7

Concept: Dataset Integration Schedules

Schedules for dataset integrations enable you to specify when, how often, and under what criteria to import data into a dataset. Workday enables you to import data into a dataset immediately on an ad hoc basis or according to a preconfigured schedule.

You can schedule the integration to run:

- Once in the future.
- On a recurring basis (Example: daily, weekly, or monthly).

- Only after another Prism scheduled process completes at a status you specify.

Integration Schedule Types

Schedule Type	Description
Recurring	An integration schedule that runs at specified intervals, such as daily, weekly, or monthly.
Dependent	An integration schedule that depends on the completion of another Prism scheduled process. For your dependency criteria, you can specify: • The process type, such as another Prism data integration. • The status that triggers integration, such as the process type successfully completing. Example: When the <i>Prism Data Acquisition Future Process</i> completes with no warnings or errors, begin integration. Note: A Prism Analytics integration schedule can depend only on another Prism-related process, such as bringing data into a dataset or publishing another dataset.

As you create an integration schedule, consider these actions that you can perform on it:

Action	Description
Activate	Activate a suspended integration schedule.
Change Schedule (recurring schedules only)	Edit the run frequency (daily, monthly, weekly), start time, and date range for the integration schedule. You can also change to another scheduled recurring process.
Delete	Permanently delete the integration schedule.
Edit Environment Restrictions	Select the environment in which you want the scheduled integration to run.
Edit	(Recurring Schedules) Edit the schedule name, recurrence criteria, and range of recurrence dates. To change the run frequency, use Change Schedule. (Dependent Schedules) Edit the schedule name, dependency, trigger status, and timed delay configurations.
Edit Scheduled Occurrence (recurring schedules only)	Update the schedule date and time for one particular occurrence of the scheduled request. You can also delete a particular occurrence of the scheduled integration.
Run Now	Run the integration schedule immediately on an ad hoc basis.

Action	Description
Suspend	Suspend use of the integration schedule. You can activate a suspended schedule.
Transfer Ownership	Transfer ownership of an integration schedule. Every process must have an assigned owner for the process to run. Example: Transfer ownership if the assigned owner becomes inactive. The person you transfer ownership to must have the appropriate security access.
View All Occurrences (recurring schedules only)	View all future occurrences of an integration schedule within a specified range of dates and times.
View Details	(Recurring schedules) View schedule details, such as recurrence criteria, error messages, the schedule owner and creator, and the next 10 scheduled integrations if applicable. (Dependent schedules) View schedule details, such as the dependency configuration, the schedule creator and owner, and the number of times run.

Related Information

Tasks

[Manage Dataset Integration Schedules](#) on page 50

[Steps: Create a Dataset with External Data \(SFTP Server\)](#) on page 41

[Steps: Create a Dataset Using Workday Data](#) on page 45

Reference: Dataset Stages

Prism Analytics supports these stage types:

Stage Type	Description
Import	The first stage in every pipeline in a derived dataset is an Import stage. An Import stage imports data from a source, and for derived datasets that source is the output of a table or an existing dataset. Workday automatically creates an Import stage when necessary. You can't delete or add an Import stage.
Parse	A Parse stage is a type of stage that enables you to describe the source data in a tabular format. Workday automatically displays the Parse stage when you create a dataset using external data. You can't delete or add a Parse stage.
Manage Fields	A Manage Fields stage is a type of stage that enables you to view field changes, select fields, and edit fields.

Stage Type	Description
Filter	A Filter stage is a type of stage that constrains rows in a dataset based on the filter criteria you define. Add Filter stages to limit the data in the dataset for analysis, such as on a particular region or year. You can add or delete a Filter stage in the middle or at the end of a pipeline.
Group By	A Group By stage is a type of stage that enables you to summarize (aggregate) multiple values in a dataset by specified groups. You can summarize the values using a summarization type, such as MIN, MAX, or SUM. Add Group By stages to get data to the appropriate level that you need to join the data in 1 dataset with another dataset. You can add or delete a Group By stage in the middle or at the end of a pipeline.
Join	A Join stage is a type of stage that combines fields from 2 dataset pipelines based on common values that exist in each pipeline. Add Join stages to view and use related data from different datasets. You can add Join stages to derived datasets only. Derived datasets are the only datasets that can use multiple datasets as their source data. You can add or delete a Join stage anywhere in the Primary Pipeline of the derived dataset.
Union	A Union stage is a type of stage that combines data from similar fields in different datasets into a single field. Add Union stages to: - Combine datasets that have similar, but not identical schema. - Combine datasets with the same schema but with source data from different locations, such as different SFTP servers. You can add Union stages to derived datasets only. Derived datasets are the only datasets that can use multiple datasets as their source data. You can add or delete a Union stage anywhere in the Primary Pipeline of the derived dataset.
Unpivot	An Unpivot stage is a type of stage that converts fields (columns) to rows. Add an Unpivot stage to consolidate data from 2 or more similar fields into a pair of new fields. You can add Unpivot stages to derived datasets only. You can add or delete an Unpivot stage anywhere in pipelines of the derived dataset.

Related Information
Concepts

[Concept: Dataset Stages](#) on page 22

Reference: Table Error File Error Codes

When you view an error file for a table load, you might see these errors:

Error Code	Notes
3000	The number of fields in the source doesn't match the number of fields in the wBucket schema. You use wBuckets when you load data into a table using the REST API.
3001	Invalid data - Text field
3002	Invalid data - Integer field. When the precision and scale of a Numeric field indicates that the field should contain integer values only, you might see this error. Example: Numeric(9,0)
3003	Invalid data - Numeric field
3004	Invalid data - Date field
3005	Invalid data - Boolean field
3006	Invalid data - Instance field
4000	The number of characters for the Text field exceeds the maximum allowed (32,000 characters).
4003	The numeric value is too large for the defined precision and scale for the Numeric field.
5000	This error code can correspond to either of these errors: - The error file has reached its maximum of 10 MB and has stopped recording new error messages. - The number of characters for the entire row of data exceeds the maximum allowed (500,000 characters).

Related Information
Concepts

[Concept: Table Error File](#) on page 15

Creating Tables and Datasets

Steps: Create a Table by File Upload

Prerequisites

Security: *Prism: Tables Create* domain in the Prism Analytics functional area.

Context

You can create a table by uploading 1 or more delimited files.

Workday:

- Uses the field information in the first file to define the table fields.
- Loads the data in the files into the table.

When you create a table by file upload, you can:

- Upload up to 100 files. All files must use the same schema.
- Modify the field type that Workday guesses for each field. Example: You can change the field type from Numeric to Text, or from Text to Multi-Instance.
- Add other fields to the table schema.
- Create a data change task to load data from files you upload into the new table.

Steps

1. Access the **Data Catalog** report.

2. Select **Create > Table**.

3. Enter the **Table Name**. You can change the table name when you edit the table.

4. (Optional) Change the **Table API Name**.

Workday automatically selects an API name based on the table name that you enter, modifying it to make it meet the name requirements. Click **Change** to change the API name. You can't change this name after you finish creating the table.

5. (Optional) Select **Enable for Analysis** to create a Prism data source using the data in this table.

Workday recommends enabling a table for analysis after:

- You apply the appropriate data source security to the table.
- The table contains the data that you want and receives new and updated data from a data change task.

6. (Optional) Create or edit 1 or more **Tags** to organize the table in the Data Catalog.

7. (Optional) Add a **Description** to help others understand what data this table contains.

8. On the **Select Schema Source** step, select **File Upload**.

9. Select 1 or more delimited files to upload.

When you upload multiple files, each file must use the same schema. Workday supports delimited files that are RFC 4180-compliant. For more information, see [RFC 4180](#).

10. On the **Edit Parsing Options** step, define how to parse the data in the file.

See [Parse External Data in a Table](#) on page 62.

11. On the **Edit Schema** step, review the fields Workday created based on the parsed file, and modify the fields if necessary.

Select a field in the list and view the field details in the inspector panel on the right side. You might want to:

- **Change** the field API Name. You can't change the API name after you save the table.
- Change the **Field Type** if Workday assigned the wrong field type. Example: Workday assigned the Numeric field type to a field with zip code data because the example rows it evaluated only contained numerals. But you know that some zip code values might contain letters or a hyphen, so you change the field type to Text.
- Change other field attributes, such as Date Format, based on the field type.
- Define some field constraints that ensure the accuracy and reliability of the data in the table, such as Required or Use as External ID.

See [Reference: Table Field Attributes](#) on page 66.

12. (Optional) Click **Add Field** to add 1 or more fields. In the inspector panel for the field, configure the field attributes.

See [Reference: Table Field Attributes](#) on page 66.

Result

Workday creates the table and starts loading the data in the files into the table. To view the data load progress you can go to:

- The **Activities** tab on the **View Table Details** report.
- The **Prism Activities Dashboard** on the **Data Catalog** report.

Refresh the page to get the most recent status.

Next Steps

To load data from a delimited file to the table again, create a data change task for the table.

If there were errors loading data into the table, download the error file from the data load from these locations:

- The **Activities** tab of the **View Table Details** report.
- The **Prism Activities Dashboard** report on the **Data Catalog** report.

Related Information**Concepts**

[Concept: Prism Analytics Data Management Workflow](#) on page 7

[Concept: Tables](#) on page 13

[Concept: Table and Dataset Field Types](#) on page 24

[Concept: Table Error File](#) on page 15

Tasks

[Parse External Data in a Table](#) on page 62

Reference

[Reference: WPA_ Fields](#) on page 61

[Reference: Supported File Formats for External Data in Tables and Datasets](#) on page 58

[Reference: Table Field Attributes](#) on page 66

[Reference: Naming Guidelines](#) on page 56

[Reference: External Data Limits](#) on page 12

[The Next Level: Prism Analytics Data Acquisition Best Practices](#)

Steps: Create Table from a Workday Report

Prerequisites

Security: *Prism: Tables Create* domain in the Prism Analytics functional area.

Context

You can create a table based on an existing Workday custom report by selecting a report as the schema source. Workday retains only the fields with field types that tables currently support.

When you create a table from a report, you can:

- Add other fields to the table schema.
- Create a data change task to load data from the report into the new table.

Steps

1. Access the **Data Catalog** report.
2. Select **Create > Table**.

3. Enter the **Table Name**. You can change the table name when you edit the table.
4. (Optional) Change the **Table API Name**.
Workday automatically selects an API name based on the table name that you enter, modifying it to make it meet the name requirements. Click **Change** to change the API name. You can't change this name after you finish creating the table.
5. (Optional) Select **Enable for Analysis** to create a Prism data source using the data in this table.
Workday recommends enabling a table for analysis after:
 - You apply the appropriate data source security to the table.
 - The table contains the data that you want and receives new and updated data from a data change task.
6. (Optional) Create or edit 1 or more **Tags** to organize the table in the Data Catalog.
7. (Optional) Add a **Description** to help others understand what data this table contains.
8. On the **Select Schema Source** step, select **Workday Report**.
9. Select the Workday custom report.
10. On the **Edit Schema** step, review the fields Workday created based on the report, and modify the fields if necessary.
Select a field in the list and view the field details in the inspector panel on the right side.
11. (Optional) Click **Add Field** to add 1 or more fields. In the inspector panel for the field, configure the field attributes.

Result

Workday creates an empty table using the schema defined in the report.

Next Steps

Create a data change task using the same report as the source to load data into the table from the report.

Related Information

Concepts

[Concept: Creating Reports to Import into Tables and Datasets](#) on page 10

[Concept: Prism Analytics Data Management Workflow](#) on page 7

[Concept: Tables](#) on page 13

[Concept: Table and Dataset Field Types](#) on page 24

Reference

[Reference: WPA_ Fields](#) on page 61

[Reference: Table Field Attributes](#) on page 66

[Reference: Naming Guidelines](#) on page 56

[The Next Level: Prism Analytics Best Practices](#)

Steps: Create Table from an Existing Table or Dataset

Prerequisites

Security:

- *Prism: Tables Create* domain in the Prism Analytics functional area.

- Any of these requirements:
 - *Prism: Tables Manage* domain in the Prism Analytics functional area.
 - *Table Viewer* or *Dataset Viewer* permission on the existing table or dataset.
 - *Table Editor* or *Dataset Editor* permission on the existing table or dataset.
 - *Table Owner* or *Dataset Owner* permission on the existing table or dataset.

Context

You can create a table based on an existing dataset or other table by selecting an existing table or dataset as the schema source. Workday defines the new table schema based on the output schema of the existing table or dataset.

If the existing dataset includes a Double field, consider converting it to a Numeric field in the dataset before creating the table.

When you create a table from an existing table or dataset, you can:

- Add other fields to the table schema.
- Create a data change task to load data from the existing table or dataset into the new table.

Steps

1. Access the **Data Catalog** report.
2. Select **Create > Table**.
3. Enter the **Table Name**. You can change the table name when you edit the table.
4. (Optional) Change the **Table API Name**.

Workday automatically selects an API name based on the table name that you enter, modifying it to make it meet the name requirements. Click **Change** to change the API name. You can't change this name after you finish creating the table.

5. (Optional) Select **Enable for Analysis** to create a Prism data source using the data in this table.

Workday recommends enabling a table for analysis after:

- You apply the appropriate data source security to the table.
- The table contains the data that you want and receives new and updated data from a data change task.

6. (Optional) Create or edit 1 or more **Tags** to organize the table in the Data Catalog.
7. (Optional) Add a **Description** to help others understand what data this table contains.
8. On the **Select Schema Source** step, select **Existing Table or Dataset**.
9. Select the dataset or table.

10. On the **Edit Schema** step, review the fields Workday created based on the existing dataset or table, and modify the fields if necessary.

Select a field in the list and view the field details in the inspector panel on the right side. You might want to:

- Define some field constraints that ensure the accuracy and reliability of the data in the table, such as Required or Use as External ID.
- Change other field attributes, such as Digits Before and Digits After, based on the field type.

See [Reference: Table Field Attributes](#) on page 66.

11. (Optional) Click **Add Field** to add 1 or more fields. In the inspector panel for the field, configure the field attributes.

Result

Workday creates an empty table using the schema that you defined.

Next Steps

Create a data change task to load data into the table.

Related Information

Concepts

[Concept: Prism Analytics Data Management Workflow](#) on page 7

[Concept: Tables](#) on page 13

[Concept: Table and Dataset Field Types](#) on page 24

Reference

[Reference: Table Field Attributes](#) on page 66

[Reference: Naming Guidelines](#) on page 56

[Reference: WPA_ Fields](#) on page 61

Steps: Create a Table Manually

Prerequisites

Security: *Prism: Tables Create* domain in the Prism Analytics functional area.

Context

You can create a table by manually defining each field in the table schema. When you create a table manually, the table is empty. You can create a data change task to load data into the table.

Steps

1. Access the **Data Catalog** report.
2. Select **Create > Table**.
3. Enter the **Table Name**. You can change the table name when you edit the table.
4. (Optional) Change the **Table API Name**.
Workday automatically selects an API name based on the table name that you enter, modifying it to make it meet the name requirements. Click **Change** to change the API name. You can't change this name after you finish creating the table.
5. (Optional) Select **Enable for Analysis** to create a Prism data source using the data in this table.
Workday recommends enabling a table for analysis after:
 - You apply the appropriate data source security to the table.
 - The table contains the data that you want and receives new and updated data from a data change task.
6. (Optional) Create or edit 1 or more **Tags** to organize the table in the Data Catalog.
7. (Optional) Add a **Description** to help others understand what data this table contains.
8. On the **Select Schema Source** step, select **Manual Input**.
The **Edit Schema** step displays where you can define each field in the table schema.
9. Click **Add Field** to add 1 or more fields.
10. In the inspector panel for the field, configure the field attributes.
See [Reference: Table Field Attributes](#) on page 66.

Next Steps

Create a data change task to load data into the table.

Related Information

Concepts

[Concept: Prism Analytics Data Management Workflow](#) on page 7

[Concept: Tables](#) on page 13

[Concept: Table and Dataset Field Types](#) on page 24

Reference

[Reference: WPA_ Fields](#) on page 61

[Reference: Supported File Formats for External Data in Tables and Datasets](#) on page 58

[Reference: Table Field Attributes](#) on page 66

[Reference: Naming Guidelines](#) on page 56

[The Next Level: Prism Analytics Best Practices](#)

Steps: Create a Dataset with External Data (SFTP Server)

Prerequisites

Security: *Prism Datasets: Create domain* in the Prism Analytics functional area.

Context

You can create a dataset using external data by transferring data from an SFTP server. You might want to create a dataset that gets its data from an external server when the server regularly collects or adds new data. You configure how often the dataset gets new data from the server.

For integration runs that transfer data from the SFTP server to succeed:

- The number of files must be less than 5,000.
- The time to transfer the data must be less than 5 hours.

Each file from the server should be less than 1 GB compressed (less than 10 GB uncompressed approximately).

Creating a dataset using external data creates a base dataset.

Steps

1. Access the **Data Catalog** report.
2. Select **Create > from SFTP**.

On the **Create Dataset Retrieval - Configure File Retrieval** task, you configure how to import the data from the SFTP server.

3. In the **Files** section, enter the filename or a filename pattern that represents 1 or more files.

The filename is case-sensitive. You can use the asterisk (*) and question mark (?) characters as wild cards to specify a filename pattern. Use the asterisk (*) to specify zero or more characters, and use the question mark (?) to specify exactly 1 character.

4. In the **Transport** section, specify how to connect to the SFTP server:

Option	Description
SFTP Address	Use this format: <code>sftp://domain_name</code> or <code>sftp://IP_address</code> To specify a port number, add it to the end of the domain name or IP address. If you don't specify a port number, Workday uses port 22.

Option	Description
Directory	(Optional) The directory on the server that contains the files. Directory names are case-sensitive. Include a leading slash (/) only for a full path, not a relative path.
Use Temp File	Writes the imported data to a temporary file in Workday with a randomly generated name. After the data import is complete, Workday automatically renames the file to the correct name. You might want to enable this option if the data import takes a very long time and might not finish before the next scheduled time to import data from the same server.
Authentication Method and Details	Select the type of security authentication that the SFTP server uses: • User Name / Password. • SSH Authentication. This option uses secure shell key authentication using X.509 certificates.

5. (Optional) In the **File Utilities** section, Consider these options:

Option	Description
Delete After Retrieval	Deletes the files on the SFTP server after the data is imported into the dataset. If Workday is unable to delete the files from the SFTP server, the data retrieval fails.
Decompress	Don't enable this option for datasets. You can transfer files that are compressed or not. For compressed files, Workday only supports gzip compression.
Decrypt Using	If you want to decrypt the imported files using Pretty Good Privacy (PGP), select a PGP Private Key Pair.

6. (Optional) In the **Environment Restrictions** section, at the **Restricted To** prompt, select the environment in which you want to use the settings defined in the **Transport** section.

If you leave this option empty, Workday applies the transport settings to each environment in which the dataset integration runs. When a dataset integration runs in a particular environment, such as Implementation or Production, the transport settings only work if the **Restricted To** option matches the current environment. When the current environment and the configured environment don't match, the dataset integration fails and retrieves no files from the SFTP server. You might want to restrict the transport settings to a particular environment to avoid inadvertently transferring test data to a non-test endpoint.

Example: You create the dataset in an Implementation environment and select Implementation in **Restricted To**. Later, you migrate this dataset to a Production environment and the next time the dataset integration runs, the integration fails. To ensure that the dataset integration runs successfully in the Production environment, edit the dataset integration details and either clear the **Restricted To** option or change it to Production.

7. On the **Create Dataset Retrieval - Schedule Request Type** task, in **Run Frequency**, specify how often to import data from the SFTP server.

If you're importing the data once in the future or on a schedule, specify the criteria for either on the **Create Dataset Retrieval - Schedule Integration** task.

After the dataset is created, you can run the integration to bring in data to the dataset on an ad hoc basis. From the related actions menu of the **View Dataset Details** report, select **Dataset > Run Integration Now**.

Note: You can't bring data into the same dataset at times that overlap with each other.

8. Specify a unique name for the dataset.

The dataset name is what displays in the Data Catalog. You can change the name when you create, edit, or copy the dataset.

9. (Optional) Change the dataset API name. Workday automatically selects an API name based on the dataset name you enter, modifying it to make it meet the name requirements. You can't change this name after you finish creating the dataset.

10. (Optional) Create or edit 1 or more tags to organize the dataset in the Data Catalog.

11. (Optional) Add a description to help others understand the data in this dataset. You can change the description when you edit the dataset.

12. Select how you want to update the data in the dataset when it receives new data from an integration run.

Option	Description
Replace	Workday deletes the existing data in the dataset and replaces it with the data it imports from the SFTP server.
Append	Workday keeps the existing data in the dataset and adds to it the new data it imports from the SFTP server. Workday imports all data in all files during every integration run. Append mode is different than incrementally updating data in a dataset. Whether the data in the dataset gets updated incrementally depends on if the SFTP server contains only incremental updates since the last integration run. The file schema must meet these requirements: • All files in every integration run must use the same parsing options (including the header row configuration) that were used during the first integration run. • The fields must be in the same order in all files in every integration run. • If the schema in a subsequent integration run contains new fields, the new fields must be located at the end of all previous fields. • If the file schema in a subsequent integration run deletes one or more fields, the deleted fields must be at the end. • Ensure that if the schema deletes fields, no future schema adds new fields, otherwise the integration run will fail. To ensure that all future integrations run successfully, always keep existing fields in the schema and only add new fields. If necessary, you can include empty (NULL) values in existing fields. • All files in a single integration must use the same schema.

Note: An integration fails when the schema of the new data doesn't contain a field that currently exists in the dataset, and the removed field is used in a stage in the dataset. Example: If the dataset includes

a Manage Fields stage and the integration brings in data that is missing a field in the dataset, the integration fails. That's because the Manage Fields stage works on every field in the dataset.

13. Click **Save**.

Workday creates the dataset, but it has no data until Workday imports the data and fields from the SFTP server during the first integration run. Depending on when you scheduled the data to import, the dataset might be empty for some time. Workday also adds 2 fields that provide information about each integration run. See [Reference: WPA_ Fields](#) on page 61.

14. (Optional) Change the name of your integration schedule. See [Manage Dataset Integration Schedules](#) on page 50.

15. Access the **Data Catalog** report, right-click the dataset you just created, and select **Edit**.

16. Configure how to parse the data in the files from the SFTP server.

See [Parse External Data in a Dataset](#) on page 67.

17. (Optional) [Add a Stage to a Dataset](#) on page 72.

You can add only some stage types to base datasets.

18. (Optional) [Add a Prism Calculated Field to a Dataset](#) on page 70.

You can add a Prism calculated field to any stage.

Related Information

Concepts

[Concept: Dataset Workspace](#) on page 16

[Concept: Datasets](#) on page 16

[Concept: Dataset Stages](#) on page 22

[Concept: Dataset Pipelines](#) on page 22

Reference

[Reference: Supported File Formats for External Data in Tables and Datasets](#) on page 58

[Reference: WPA_ Fields](#) on page 61

[Reference: Naming Guidelines](#) on page 56

[Reference: External Data Limits](#) on page 12

[The Next Level: Prism Analytics Best Practices](#)

Steps: Create a Dataset with External Data (Upload a File)

Prerequisites

Security: *Prism Datasets: Create* domain in the Prism Analytics functional area.

Context

You can create a dataset using external data by uploading a file. You might want to create a dataset by uploading a file when the data in the file is less likely to change over time.

When you create a dataset by uploading a file, the source data in the dataset remains the same over time. However, you can change the data in the dataset later by uploading a new file to the dataset. See [Upload a New File to a Dataset](#) on page 51.

Creating a dataset using external data creates a base dataset.

Steps

1. Access the **Data Catalog** report.
2. Select **Create > from File**.

3. Navigate to a file on your local machine and open it.

The maximum size file you can upload is 500 MB.

4. Define how to parse the data in the file.

See [Parse External Data in a Dataset](#) on page 67.

5. Specify a unique name for the dataset.

The dataset name is what displays in the Data Catalog. You can change the name when you create, edit, or copy the dataset.

6. (Optional) Change the dataset API name. Workday automatically selects an API name based on the dataset name you enter, modifying it to make it meet the name requirements. You can't change this name after you finish creating the dataset.

7. (Optional) Create or edit 1 or more tags to organize the dataset in the Data Catalog.

8. (Optional) Add a description to help others understand the data in this dataset. You can change the description when you edit the dataset.

9. (Optional) [Add a Stage to a Dataset](#) on page 72.

You can add only some stage types to base datasets.

10. (Optional) [Add a Prism Calculated Field to a Dataset](#) on page 70.

You can add a Prism calculated field to any stage.

Related Information

Concepts

[Concept: Dataset Workspace](#) on page 16

[Concept: Datasets](#) on page 16

[Concept: Dataset Stages](#) on page 22

[Concept: Dataset Pipelines](#) on page 22

Tasks

[Upload a New File to a Dataset](#) on page 51

Reference

[Reference: Supported File Formats for External Data in Tables and Datasets](#) on page 58

[Reference: Naming Guidelines](#) on page 56

[Reference: External Data Limits](#) on page 12

[The Next Level: Prism Analytics Best Practices](#)

Steps: Create a Dataset Using Workday Data

Prerequisites

Security: *Prism Datasets: Create* domain in the Prism Analytics functional area.

Context

You can create a dataset using Workday data. You do this by creating a dataset using an existing Workday custom report as the source for the dataset.

You configure how often the dataset gets new data from the report.

Workday retains only the fields with field types that datasets currently support.

Creating a dataset from a custom report creates a base dataset.

Steps

1. Access the **Data Catalog** report.

2. Select **Create > from Custom Report**.

Workday displays reports that meet the eligibility requirements for importing into Prism. See [Concept: Creating Reports to Import into Tables and Datasets](#) on page 10.

On the **Create Dataset Retrieval - Configure Report Retrieval** task, you configure how to import the data from the custom report.

3. Select a **Custom Report** that has the data you want to import into this dataset.

4. In the **Report Criteria** table, select values for the report prompts, if applicable.

Workday filters the report data with the specified values as the report runs and before importing the data into the dataset. As you complete this step, consider:

Option	Description
Value Type	This option affects how Workday determines the value for this field prompt: • Specify Value. Workday uses the same value that you specify here each time it runs the report to import data into the dataset. • Determine Value at Runtime. Workday uses the current value in a field you specify each time it runs the report to import data into the dataset.
Value	Workday uses the value or field you select here to filter the data in the report.

5. (Optional) In the **Environment Restrictions** section, at the **Restricted To** prompt, select the environment in which you want to use the settings defined in the **Transport** section.

If you leave this option empty, Workday applies the transport settings to each environment in which the dataset integration runs. When a dataset integration runs in a particular environment, such as Implementation or Production, the transport settings only work if the **Restricted To** option matches the current environment. When the current environment and the configured environment don't match, the dataset integration fails and retrieves no data from the specified custom report. You might want to restrict the transport settings to a particular environment to avoid inadvertently transferring test data to a non-test endpoint.

Example: You create the dataset in an Implementation environment and select Implementation in **Restricted To**. Later, you migrate this dataset to a Production environment and the next time the dataset integration runs, the integration fails. To ensure that the dataset integration runs successfully in the Production environment, edit the dataset integration details and either clear the **Restricted To** option or change it to Production.

6. On the **Create Dataset Retrieval - Schedule Request Type** task, in **Run Frequency**, specify how often to import data from the custom report.

If you're importing the data once in the future or on a schedule, specify the criteria for either on the **Create Dataset Retrieval - Schedule Integration** task.

After the dataset is created, you can run the integration to bring in data to the dataset on an ad hoc basis. From the related actions menu of the **View Dataset Details** report, select **Dataset > Run Integration Now**. You can't bring data into the same dataset at times that overlap with each other.

Note: An integration fails to bring in new data from the custom report when the report schema doesn't contain a field that currently exists in the dataset, and the removed field is used in a stage in the dataset. Example: If the dataset includes a Manage Fields stage and the integration brings in data that is missing a field in the dataset, the integration fails. That's because the Manage Fields stage works on every field in the dataset.

7. Specify a unique name for the dataset.

The dataset name is what displays in the Data Catalog. You can change the name when you create, edit, or copy the dataset.

8. (Optional) Change the dataset API name. Workday automatically selects an API name based on the dataset name you enter, modifying it to make it meet the name requirements. You can't change this name after you finish creating the dataset.
9. (Optional) Create or edit 1 or more tags to organize the dataset in the Data Catalog.
10. (Optional) Add a description to help others understand the data in this dataset. You can change the description when you edit the dataset.
11. Select how you want to update the data in the dataset when it receives new data from an integration run.

Option	Description
Replace	Workday deletes the existing data in the dataset and replaces it with the data it imports from the custom report.
Append	Workday keeps the existing data in the dataset and adds to it the new data it imports from the custom report. Workday imports all data in the report during every integration run, resulting in duplicate data in the dataset. Select append mode for a custom report dataset when you want a snapshot of the custom report data to maintain history in the dataset for trending use cases. Note: Ensure that you don't change the Column Heading Override XML Alias values in the custom report definition. Workday uses these values to map fields from the custom report into the dataset.

12. Click **Save**.

Workday creates the dataset, but it has no data until Workday runs the report and then imports the data and fields from the report during the first integration run. Depending on when you scheduled the data to import, the dataset might be empty for some time. Workday also adds 2 fields that provide information about each integration run. See [Reference: WPA_ Fields](#) on page 61.

13. (Optional) Change the name of your integration schedule. See [Manage Dataset Integration Schedules](#) on page 50
14. Access the **Data Catalog** report, right-click the dataset you just created, and select **Edit**.
15. (Optional) [Add a Stage to a Dataset](#) on page 72.
You can add only some stage types to base datasets.
16. (Optional) [Add a Prism Calculated Field to a Dataset](#) on page 70.
You can add a Prism calculated field to any stage.

Related Information

Concepts

[Concept: Dataset Workspace](#) on page 16

[Concept: Datasets](#) on page 16

[Concept: Dataset Stages](#) on page 22

[Concept: Dataset Pipelines](#) on page 22

[Concept: Creating Reports to Import into Tables and Datasets](#) on page 10

Reference

[Reference: WPA_ Fields](#) on page 61

[Reference: Naming Guidelines](#) on page 56

[The Next Level: Prism Analytics Best Practices](#)

Steps: Create a Derived Dataset

Prerequisites

Security:

- *Prism Datasets: Create* domain in the Prism Analytics functional area.
- Any of these requirements:
 - *Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - *Dataset Viewer* permission on the dataset to import into the derived dataset.
 - *Dataset Editor* permission on the dataset to import into the derived dataset.
 - *Dataset Owner* permission on the dataset to import into the derived dataset.

Context

When you first create a derived dataset, Workday creates the Primary Pipeline. Import other datasets into the derived dataset so you can blend data together.

You can add a stage to any pipeline in the dataset. However, some stages, such as the Join stage, can only be added to the Primary Pipeline.

When you add a Join or Union stage to the Primary Pipeline, you must select another pipeline in the derived dataset to blend with the Primary Pipeline. Workday uses the last stage of that pipeline as the input to the Join or Union stage.

Steps

1. Access the **Data Catalog** report.
2. Select **Create > Derived Dataset**.
3. Select a table or dataset from the list.
4. Specify a unique name for the dataset.

The dataset name is what displays in the Data Catalog. You can change the name when you create, edit, or copy the dataset.
5. (Optional) Change the dataset API name. Workday automatically selects an API name based on the dataset name you enter, modifying it to make it meet the name requirements. You can't change this name after you finish creating the dataset.
6. (Optional) Create or edit 1 or more tags to organize the dataset in the Data Catalog.
7. (Optional) Add a description to help others understand the data in this dataset. You can change the description when you edit the dataset.
8. Click **Edit Transformations** to access the **Edit Dataset Transformations** task.
9. The **Edit Dataset Transformations** task displays 1 pipeline (the Primary Pipeline) that contains an **Import** stage.
10. Create 1 or more additional pipelines by importing an existing dataset.

See [Import a Table or Dataset into a Derived Dataset](#) on page 49.
11. (Optional) [Add a Stage to a Dataset](#) on page 72.

You can add any stage type to derived datasets, but you can add some stage types to the Primary Pipeline only. To blend data from 2 pipelines, consider adding a Join stage to the Primary Pipeline.
12. (Optional) [Add a Prism Calculated Field to a Dataset](#) on page 70.

You can add a Prism calculated field to any stage in any pipeline.

Related Information

Concepts

[Concept: Dataset Workspace](#) on page 16

[Concept: Datasets](#) on page 16

[Concept: Dataset Stages](#) on page 22

[Concept: Dataset Pipelines](#) on page 22

Reference

[Reference: Naming Guidelines](#) on page 56

[The Next Level: Prism Analytics Best Practices](#)

Import a Table or Dataset into a Derived Dataset

Prerequisites

- Any of these security requirements:
 - *Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - *Dataset Editor* permission on the derived dataset.
 - *Dataset Owner* permission on the derived dataset.
- Any of these security requirements:
 - *Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - *Dataset Viewer* permission on the dataset to import into the derived dataset.
 - *Dataset Editor* permission on the dataset to import into the derived dataset.
 - *Dataset Owner* permission on the dataset to import into the derived dataset.

Context

A derived dataset is based on 1 or more existing tables or datasets. Use derived datasets to blend and combine together data from multiple sources. In order to blend and combine data from multiple sources, you need to import multiple tables or datasets into the derived dataset. When you first create a derived dataset, you base it on an existing table or dataset. Afterward, you must import other tables or datasets into the derived dataset.

When you import a table or dataset into a derived dataset, Workday creates a new pipeline. The pipeline name is the same as the table or dataset name you import. You can add stages to the new pipeline.

Once a derived dataset has multiple tables or datasets imported into it, you can add a stage, such as a Join stage, to the Primary Pipeline to blend data with any other pipeline.

Steps

1. Access the **Edit Dataset Transformations** task for a derived dataset.
2. In the **Pipelines** panel, click **Add Pipeline**.
3. Select a table or dataset from the list.

Result

The **Pipelines** panel displays the new pipeline with the first stage being an **Import** stage. The pipeline name is the same as the table or dataset you imported.

Next Steps

- (Optional) Add a stage, such as a Join stage, that blends together data from the Primary Pipeline and the pipeline you added.

Related Information

Tasks

[Steps: Create a Derived Dataset](#) on page 48

Manage Dataset Integration Schedules

Prerequisites

Security: *Prism Datasets: Manage* domain in the Prism Analytics functional area.

Context

You can manage how you set up integration schedules for base datasets created from:

- SFTP
- Custom reports

You can schedule the integration to run:

- Once in the future.
- On a recurring basis (Example: daily, weekly, or monthly).
- Only if another Prism scheduled process completes at a status you specify.

Steps

1. Access the **View Dataset Details** report for the dataset.
2. From the related actions menu, select **Dataset > Edit Integration Details**.
3. (Recurring schedules) As you set up the schedule, consider:

Option	Description
Catch Up Behavior	Select how many times the scheduled integration runs after maintenance issues cause errors. Example: If you schedule an integration to run multiple times in a week when your environment is down for maintenance, you can limit the process to run once instead of catching up all missed occurrences.

4. (Dependent schedules) As you set up the schedule, consider:

Option	Description
Dependency	Select a Prism-related schedule on which the dataset integration schedule depends.
Trigger on Status	Select the status of the scheduled future process that causes the dataset integration to run. Workday recommends using one of the completed statuses. Example: You select Prism Integration and an integration schedule called <i>Dataset Integration Schedule: Monthly Acquisition Expenses</i> . In the Trigger on Status field, you select Completed . Workday brings data into the dataset only after the <i>Dataset Integration Schedule: Monthly</i>

Option	Description
	<i>Acquisition Expenses</i> integration successfully completes.
Time Delayed Configuration	(Optional) Specify the number of days, hours, or minutes to delay running the dataset integration after the trigger. You might want to delay integration to review the latest source files.

5. (Optional) Change the name of the schedule in the **Request Name** field. Workday assigns a name to the schedule based on the name of the dataset and prepends *Dataset Integration Schedule:* to the name.
6. (Optional) Perform actions such as transferring ownership of the schedule or editing 1 scheduled occurrence.
 - a) Access the **View Integration Details** report for the dataset.
 - b) Find the integration schedule in the **Request Name** column on the **Schedules** tab.
 - c) From the related actions menu of the integration schedule, select **Schedule Future Process** and then the desired action.

Result

Workday imports data into the dataset based on the criteria you specified.

You can view the status of all scheduled integration processes in the **Process Monitor** and **Scheduled Future Processes** reports. The status includes the date and time of the last successful integration. The last successful integration date informs you about the freshness of the data brought into the dataset. Example: If the last successful integration date is 1 week ago, but your integration schedule is set to run daily, this discrepancy could indicate a failure in the integration process.

Related Information

Concepts

[Concept: Dataset Integration Schedules](#) on page 31

Upload a New File to a Dataset

Prerequisites

- Any of these security requirements:
 - *Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - *Dataset Editor* permission on the dataset.
 - *Dataset Owner* permission on the dataset.

Context

When you create a dataset by uploading a file, the data in that dataset stays the same over time. If you have a new version of the source file, you can upload it to the same dataset. You might want to upload a new file to update an existing dataset instead of creating a new dataset. When you update an existing dataset, you maintain any relationships with datasets that depend on the existing dataset.

When you upload a new file, all existing data is replaced with the data in the new file.

Sometimes, the fields in the source file might change, also known as a schema change. Fields might be added, deleted, or moved. When the schema changes and the new file is imported into the dataset, you must edit the dataset and save it to incorporate the changes in the dataset. If you don't save the dataset, it'll continue to use the old schema definition. The next time the dataset is published, you might get inconsistent data in the Prism data source.

Note: Uploading a file fails when the schema of the new file doesn't contain a field that currently exists in the dataset, and the removed field is used in a stage in the dataset. Example: If the dataset includes a Manage Fields stage and you try to upload a file that is missing a field in the dataset, the upload fails. That's because the Manage Fields stage works on every field in the dataset.

Steps

1. Access the **View Dataset Details** report for the dataset you want to update with new data.
2. Click **Upload File**.
3. In the confirmation dialog, click **Upload**.
4. Navigate to and select the local file.
5. If Workday successfully uploads the file, navigate to the **Edit Dataset Transformations** task.
If the schema changed, the **Save** button is active.
6. If the Save button is active, click **Save** to apply schema changes.

Result

Workday replaces the data in the dataset with the data in the file you uploaded. It updates the fields in the dataset if the schema in the uploaded file is different.

Next Steps

Verify that no schema changes broke any Prism calculated fields, stages, derived datasets, or Workday reports that depend on the dataset whose schema changed.

Related Information

Concepts

[Concept: Dataset Schema Changes](#) on page 20

Tasks

[Steps: Create a Dataset with External Data \(Upload a File\)](#) on page 44

Reference

[Reference: External Data Limits](#) on page 12

View Prism Data Usage

Prerequisites

- Any of these security requirements:
 - *Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - *Prism Datasets: Owner Manage* domain in the Prism Analytics functional area.
 - *Prism: Tables Manage* domain in the Prism Analytics functional area.
 - *Prism: Tables Owner Manage* domain in the Prism Analytics functional area.
 - *Prism Datasets: Publish* domain in the Prism Analytics functional area.
 - *Dataset Viewer* permission on 1 or more datasets.
 - *Dataset Editor* permission on 1 or more datasets.
 - *Dataset Owner* permission on 1 or more datasets.
 - *Table Viewer* permission on 1 or more tables.
 - *Table Editor* permission on 1 or more tables.
 - *Table Owner* permission on 1 or more tables.

Context

You can view how much Prism data your organization has used on your tenant. You might want to view your Prism data usage to ensure you're in compliance with your purchase agreement with Workday.

Workday displays this data usage information:

- **Reportable Rows.** This value summarizes all rows in published datasets and tables enabled for analysis in your tenant, including the datasets and tables that you don't have permission on.
- **Table and dataset usage.** This grid lists all tables and datasets that you have permission on, and includes the disk space used and number of reportable rows per table and dataset.

Steps

1. Access the **Data Catalog** report.
2. Click **Prism Usage**.

View Table and Dataset Lineage

Prerequisites

- Any of these security requirements for dataset lineage:
 - *Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - *Prism Datasets: Owner Manage* domain in the Prism Analytics functional area.
 - *Prism Datasets: Publish* domain in the Prism Analytics functional area.
 - *Dataset Viewer* permission on the dataset.
 - *Dataset Editor* permission on the dataset.
 - *Dataset Owner* permission on the dataset.
- Any of these security requirements for table lineage:
 - *Prism: Tables Manage* domain in the Prism Analytics functional area.
 - *Prism: Tables Owner Manage* domain in the Prism Analytics functional area.
 - *Prism Datasets: Publish* domain in the Prism Analytics functional area.
 - *Table Viewer* permission on the table.
 - *Table Editor* permission on the table.
 - *Table Owner* permission on the table.

Context

When you bring in data and transform it in Prism Analytics, you can create complex workflows containing multiple tables and datasets. You create a derived dataset by importing a table or dataset on which the derived dataset is based. The derived dataset depends on the table or dataset you import into it. You can visually see these dependencies by viewing the lineage for a table or dataset.

Viewing the lineage enables you to see dependencies, and to trace the origin of a derived dataset back to its tables and base datasets. The lineage gives you insight into the potential consequences of changes you make to your data (impact analysis).

Steps

1. Access the **Data Catalog** report.

2. Right-click a table or dataset whose lineage you want to view, and select **View Lineage**.

On the **View Table Lineage** or **View Dataset Lineage** report, the graph displays dependencies in both directions from the selected object where applicable:

- Upstream and downstream dependencies. When you view the lineage of a derived dataset, the graph displays the datasets imported into the derived dataset, and any other derived datasets that import this derived dataset.
- Downstream dependencies only. When you view the lineage of a table or base dataset, the graph displays any derived datasets that import this table or base dataset.

Related Information

Tasks

[View Dataset Dependencies](#) on page 55

Reference

[The Next Level: Prism Performance and Troubleshooting Tips](#)

View Field Lineage

Prerequisites

Any of these security requirements:

- *Prism Datasets: Manage* domain in the Prism Analytics functional area.
- *Prism Datasets: Owner Manage* domain in the Prism Analytics functional area.
- *Prism Datasets: Publish* domain in the Prism Analytics functional area.
- *Dataset Editor* permission on the dataset.
- *Dataset Owner* permission on the dataset.
- *Dataset Viewer* permission on the dataset and the tenant is configured to have the **Enable Prism Dataset Transformations** check box enabled.

Context

When you bring in data and transform it in Prism Analytics, you can create complex workflows containing multiple transformation stages across various datasets. You can visually see the different stages and datasets a field passed through up to that point by viewing the lineage for the field.

Steps

1. Access the **View Dataset Lineage** report for a derived dataset.
2. Select the field whose lineage that you want to view, and select **View Field Lineage**.

On the View Field Lineage report, the graph displays all transformations involving data in the field up to that point across all datasets you have access to.

When you access the View Field Lineage report, you start at the field that you selected, the root node, and you can:

- Select a node to view details in the inspector panel on the stage represented in that node.
- Control your view of the lineage by expanding and collapsing chevrons in the graph.
- View the changes across different datasets and tables, differentiated by a colored outline encompassing all nodes in a dataset.
- Navigate through decision nodes, where there's a divergence in the lineage.
- View all nodes in datasets and tables that you have at least View Table or View Transformations permissions for.
- Trace the lineage of all dependent fields for calculated fields.

Related Information

Concepts

[Concept: Dataset Field Origin](#) on page 23

[Concept: Field Lineage](#) on page 24

Reference

[2022R2 What's New Post: Field Lineage for Prism Analytics](#)

View Dataset Dependencies

Prerequisites

- Any of these security requirements:
 - Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - Prism Datasets: Owner Manage* domain in the Prism Analytics functional area.
 - Prism Datasets: Publish* domain in the Prism Analytics functional area.
 - Prism Datasets: Create* domain in the Prism Analytics functional area.
 - Dataset Viewer* permission on the dataset.
 - Dataset Editor* permission on the dataset.
 - Dataset Owner* permission on the dataset.

Context

You create a derived dataset by importing a table or dataset on which the derived dataset is based. The derived dataset depends on the table or dataset you import into it. You can view these dataset dependencies.

Steps

- Access the **Data Catalog** page.
- Select a dataset to open its inspector panel.
- Scroll down to the **Imported By** section to view the derived datasets that imported the selected dataset.

Reference: Supported Date Formats for External Data in Tables and Datasets

External data that you bring into the Data Catalog might contain fields with date or time values. Workday only supports some date formats. How Workday uses the date formats depends on the object you create:

- Table. When you define a Date field in the schema of a table, you can specify any of the supported date formats. The date values in the external data must match the specified date format in order for the row to be valid and loaded into the table.
- Base dataset. If Workday recognizes the format of a date field in the external file, it automatically assigns the Date field type when parsing the file.

Workday supports these date formats as well as any shortened versions of them:

Format Type	Format
Date and time	yyyy-MM-dd'T'HH:mm:ss.SSSZZ
	yyyy-MM-dd'T'HH:mm:ssZZ
	yyyy-MM-dd'T'HH:mm:ss

Format Type	Format
	EEE, dd MMM yyyy HH:mm:ss Z MM/dd/yy h:mm:ss a ZZ MM/dd/yy h:mm:ss a MM/dd/yy H:mm:ss ZZ MM/dd/yy H:mm:ss yy-MM-dd h:mm:ss a ZZ yy-MM-dd h:mm:ss a yy-MM-dd H:mm:ss ZZ yy-MM-dd H:mm:ss yyyy-MM-dd HH:mm:ss.SSS
Date only	yyyy-MM-ddZZ yy-MM-dd yyyy-MM-dd MM/dd/yy
Time only	'T'HH:mm:ssZZ 'T'HH:mm:ss HH:mm:ssZZ HH:mm:ss

Related Information
Concepts

[Concept: Table and Dataset Field Types](#) on page 24

Tasks

[Change Dataset Field Types](#) on page 74

Reference

[TO_DATE](#) on page 150

Reference: Naming Guidelines

Table and Dataset Names

Workday suggests that you follow these naming restrictions and recommendations. Workday might not enforce all naming restrictions depending on how you create the dataset.

Name Type	Table	Dataset (base and derived)
Name	This is the display name. • Must be unique in the Data Catalog. Table names are case insensitive. • Can contain a maximum of 255 characters.	This is the display name. • Must be unique in the Data Catalog. Dataset names are case insensitive. • Can contain a maximum of 255 characters.

Name Type	Table	Dataset (base and derived)
	- Can include any character, including multi-byte characters, as long as all characters are UTF-8 encoded. - Can't start with a space or WPA_. - Can't end with a space.	- Can include any character, including multi-byte characters, as long as all characters are UTF-8 encoded. - Can't start with a space or WPA_. - Can't end with a space. For datasets created from SFTP or custom reports, Workday enforces only these display name restrictions: - Must be unique in the Data Catalog. Dataset names are case insensitive. - Can contain a maximum of 255 characters.
API Name	- Must be unique in the Data Catalog. Table API names are case insensitive. - Can contain a maximum of 255 characters. - Can only include alphanumeric and underscore characters. - Must start with a letter. - Must end with an alphanumeric character. - Can't start with WPA_.	- Must be unique in the Data Catalog. Dataset API names are case insensitive. - Can contain a maximum of 255 characters. - Can only include alphanumeric and underscore characters. - Must start with a letter. - Must end with an alphanumeric character. - Can't start with WPA_.
Field Name	This is the display name. Can't start with a space or with WPA_. Table field names are case insensitive.	This is the display name. - Can't start with (or with WPA_. - Must be unique within the dataset. Dataset field names are case sensitive.
Field API Name	- Must be unique in the table. Table API field names are case insensitive. - Can contain a maximum of 255 characters. - Can only include alphanumeric and underscore characters. - Must start with a letter. - Can't end with an underscore character.	Workday uses dataset API field names as the dataset field display names.

Name Type	Table	Dataset (base and derived)
	Can't start with WPA_.	

Data Change Task Names

Data change task names:

- Must be unique in the Data Catalog. Data change task names are case insensitive.
- Can contain a maximum of 255 characters.
- Can include any character, including multi-byte characters, as long as all characters are UTF-8 encoded.
- Can't start with a space or WPA_.
- Can't end with a space.

Data change task API names:

- Must be unique in the Data Catalog. Table API names are case insensitive.
- Can contain a maximum of 255 characters.
- Can only include alphanumeric and underscore characters.
- Must start with a letter.
- Must end with an alphanumeric character.
- Can't start with WPA_.

Connection Names

SFTP connection names:

- Must be unique.
- Are case insensitive.
- Can contain a maximum of 255 characters.
- Can include any character, including multi-byte characters, as long as all characters are UTF-8 encoded.

Prism Data Source Names

The Prism data source name comes from the dataset API name of the published dataset.

wBucket Names

- Must be unique.
- Can contain a maximum of 255 characters.
- Can only include alphanumeric and underscore characters.
- Must start with a letter.
- Can't end with an underscore character.
- Can't start with WPA_.

Reference: Supported File Formats for External Data in Tables and Datasets

To bring in non-Workday data as a table or dataset, Workday parses the data into records (rows) and fields. All characters must be UTF-8 encoded.

Tables and datasets support these source file formats:

Format	Description
Delimited Text	A delimited file is a plain text file format for describing tabular data. Comma-separated value (CSV) files are the most common delimited files. It refers to any file that: • Is plain text (typically ASCII or Unicode characters) • Has 1 record per line. • Has records divided into fields. • Has the same sequence of fields for every record. Records are separated by line breaks, and fields within a line are separated by a delimiter (usually a comma character). If the delimiter also exists in the field values, it must be escaped. Workday supports single character escapes (such as a backslash), as well as enclosing field values in double quotes (as is common with CSV files).

Reference: Currency Format Requirements for External Data

External data that you bring into a table or dataset might contain fields with currency values. If Workday recognizes the format of a Currency field, it automatically assigns the Currency field type.

For Workday to recognize a single field value as valid currency data, it must meet these requirements:

- It must contain a numeric amount.
- The numeric amount can only use a period as the decimal separator.
- The numeric amount can't use any character to separate thousands.
- It must contain a valid 3-digit currency code that Workday recognizes.
- The 3-digit currency code can occur either before or after the numeric amount.
- It can contain valid currency symbols that Workday recognizes, but it must also contain the 3-digit currency code.
- It can't contain extraneous characters, but can contain extra spaces before or after the numeric amount or currency code.

If a Currency field contains any value that doesn't meet these requirements, Workday treats the value as NULL.

Example: Workday recognizes these single data values as valid currency data:

- 3000.00 USD
- \$3000.00 USD
- USD 3000.00
- USD \$3000.00
- -\$3,000.00 USD
- (\$3,000.00) USD

Related Information

Concepts

[Concept: Table and Dataset Field Types](#) on page 24

Tasks

[Change Dataset Field Types](#) on page 74

[Add a Prism Calculated Field to a Dataset](#) on page 70

Reference: Date Format Symbols

Workday recognizes specific characters as symbols to represent part of a date format when you create and edit tables and datasets. This section describes the symbols to use and the patterns use them in when you define your date format. The count and order of the symbols determine the date format.

Workday treats any characters in the pattern that aren't in the ranges of a-z or A-Z as quoted delimiter text. Example: Workday treats the slash (/) and colon (:) characters delimiter text even if they aren't escaped with single quotes.

Symbol	Meaning	Presentation	Examples	Notes
G	era	text	AD	
C	century of era (0 or greater)	number	20	
Y	year of era (0 or greater)	year	1996	Numeric presentation for year and week year fields are handled specially. Example: If the count of 'y' is 2, the year will be displayed as the zero-based year of the century, which is two digits.
x	week year	year	1996	Numeric presentation for year and week year fields are handled specially. Example: If the count of 'y' is 2, the year will be displayed as the zero-based year of the century, which is two digits.
w	week number of week year	number	27	
e	day of week (number)	number	2	
E	day of week (name)	text	Tuesday; Tue	If the number of pattern letters is 4 or more, the full form is used; otherwise a short or abbreviated form is used.
y	year	year	1996	
D	day of year	number	189	
M	month of year	month	July; Jul; 07	3 or more uses text, otherwise uses a number

Symbol	Meaning	Presentation	Examples	Notes
d	day of month	number	10	If the number of pattern letters is 3 or more, the text form is used; otherwise the number is used.
a	half day of day	text	PM	
K	hour of half day (0-11)	number	0	
h	clock hour of half day (1-12)	number	12	
H	hour of day (0-23)	number	0	
k	clock hour of day (1-24)	number	24	
m	minute of hour	number	30	
s	second of minute	number	55	
S	fraction of second	number	978	
z	time zone	text	Pacific Standard Time; PST	If the number of pattern letters is 4 or more, the full form is used; otherwise a short or abbreviated form is used.
Z	time zone offset/id	zone	-0800; -08:00; America/Los_Angeles	'Z' outputs offset without a colon, 'ZZ' outputs the offset with a colon, 'ZZZ' or more outputs the zone ID.
'	escape character for text-based delimiters	delimiter		
"	literal representation of a single quote	literal	'	

Reference: WPA_ Fields

When you create any table or a base dataset that uses an integration, Workday automatically creates extra fields in the table or dataset. These fields help you to uniquely identify rows in the table or dataset from different integration runs.

Field Name	Description
WPA_LoadID	This field returns a value of type Text containing a unique identifier of the integration run or data load activity that imported the current row of data into the dataset or table. Workday adds this field to both tables and datasets.

Field Name	Description
WPA_LoadTimestamp	This field returns a value of type Date (to the millisecond) containing the date and time of the integration run that imported the current row of data into the dataset or table. Workday adds this field to both tables and datasets.
WPA_RowID	This field returns a value of type Text containing a unique row identifier for each row in a data load activity or integration run. Workday adds this field to tables.
WPA_UpdateID	This field returns a value of type Text containing a unique identifier of the data load activity that updated the current row of data in the table. Workday adds this field to tables.
WPA_UpdateTimestamp	This field returns a value of type Date (to the millisecond) containing the date and time of the data load activity that updated the current row of data in the table. Workday adds this field to tables.

You can't modify or delete these fields, but you can hide them. Use these fields with the other fields to uniquely identify rows of data in the table or dataset from multiple integrations.

You can also use these fields to group data together from a single data load or integration. Example: you can create a Group By stage and group on the WPA_LoadID field and Count the number of rows from each integration run.

Related Information

Tasks

[Steps: Create a Dataset with External Data \(SFTP Server\)](#) on page 41

[Steps: Create a Dataset Using Workday Data](#) on page 45

Editing Tables

Parse External Data in a Table

Prerequisites

Security:

- *Prism: Tables Create* domain in the Prism Analytics functional area when creating a table.
- Any of these security requirements when editing a table:
 - *Prism Datasets: Owner Manage* domain in the Prism Analytics functional area.
 - *Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - *Table Editor* permission on the table.
 - *Table Owner* permission on the table.
 - *Can Insert Table Data* permission on the table.

Context

When you load a delimited file into a table, you must define how Workday parses the data. You define the parsing options on the **Edit Parsing Options** step when you load data into a table, such as creating a table by uploading a file, or when adding more rows to an existing table.

Workday supports delimited files that are RFC 4180-compliant. For more information, see [RFC 4180](#).

Steps

1. Access the **Edit Parsing Options** step for loading data into a table.
2. Configure the parsing options.

As you complete this task, consider:

Option	Description
Row Delimiter	Specifies the single character that separates rows (or records) in your source data files. In most delimited files, rows are separated by a new line, such as the line feed character, carriage return character, or carriage return plus line feed. Line feed is the standard new line representation on UNIX-like operating systems. Other operating systems (such as Windows) might use carriage return individually, or carriage return plus line feed. Selecting Any New Line causes Workday to recognize any of these representations of a new line as the row delimiter.
Field Delimiter	Specifies the single character that separates the fields (or columns) of a row in your source data files. Comma is the most common field delimiter.
Field Names	Specifies the default name of each field. You can change the field names after you finish defining the parsing options. Field names must conform to the name validation rules. Workday automatically treats the first line in each source file as a header row instead of as a row of data. If you don't want to use the first line as names for your fields, clear Use values from first row.
Escape Character	Specifies the single character used to escape the Quote Character or another instance of the Escape Character when a Quote Character is specified. Workday reads an escape character as data only if it's escaped with another escape character. If your data values contain quote characters as data, those characters must be escaped and the entire field value must be enclosed with the Quote Character. If not, then Workday assumes that the quote character denotes a new field.
Quote Character	The character that encloses a single field value, if any. Some delimited files use the quote character to enclose individual data values. The quote character is typically the double quote character ("). If a field value contains a field delimiter as data, then the field value must be enclosed in the Quote Character, otherwise Workday assumes that the field delimiter denotes a new field.

Option	Description
	If a field value contains the quote character as data, then the field value must be enclosed in the Quote Character and it must be escaped, either by the Escape Character or another quote character. If a field value contains a row delimiter (such as a new line character) as data, then the field value must be enclosed in the Quote Character. Suppose that you have a row with these 3 data values: <pre>weekly special wine, beer, and soda "2 for 1" or 9.99 each</pre> If the field delimiter is a comma, the quote character is a double quote, and the escape character is a double quote, then a correctly formatted row in the source data looks like: <pre>"weekly special", "wine, beer, and soda", ""2 for 1"" or 9.99 each"</pre>
Comment Character	Specifies the character that represents a comment at the beginning of a line of text. Workday ignores every line in the external file that starts with the comment character. Example: Select # as the Comment Character to ignore lines that start with #. When the Comment Character is empty, Workday reads all lines as rows of data.
Rows to ignore	Specifies the number of lines at the beginning of the file to ignore when reading the source file. To use this with the Use values from first row option, ensure that the line containing the field names is visible and is the first remaining line.
Jagged Rows	Select these options when the schema of the source file isn't an exact match of the table schema, and you want Workday to ignore any missing or extra fields at the end of the file schema.
Field Options	These options control how to handle whitespace characters in Text fields. - Trim leading spaces outside quotes. This option removes whitespace characters outside the quotes of Text fields before the quote character. - Trim trailing spaces outside quotes. This option removes whitespace characters outside the quotes of Text fields after the quote character. - Trim leading spaces in quotes. This option removes whitespace characters inside the quotes of Text fields at the beginning of the field value. - Trim trailing spaces in quotes. This option removes whitespace characters inside the quotes of Text fields at the end of the field value.

Related Information**Tasks**

[Steps: Create a Table by File Upload](#) on page 35

Reference

[Reference: Naming Guidelines](#) on page 56

Edit a Table

Prerequisites

Any of these security requirements:

- *Prism Datasets: Owner Manage* domain in the Prism Analytics functional area.
- *Prism Datasets: Manage* domain in the Prism Analytics functional area.
- *Table Editor* permission on the table.
- *Table Owner* permission on the table.
- *Table Schema Editor* permission on the table.

Context

You can edit a table by changing the table display name or changing the schema.

You can change the table schema by adding fields, deleting fields, or changing field attributes, such as the field type. However, you can only change field attributes when the table contains no data.

Steps

1. Access the **View Table Details** report for a table.
2. Select **Edit Schema** from the **Quick Actions** button, or from the related actions menu, select **Table > Edit**.
3. (Optional) Click **Add Field** to add 1 or more fields. In the inspector panel for the field, configure the field attributes.
See [Reference: Table Field Attributes](#) on page 66.
4. (Optional) Delete a field by clicking the trash can in the right-most column of a field in the list.
5. (Optional) Change the field attributes of an existing field.
 - a) Select a field in the list, and view the field details in the inspector panel on the right side.
 - b) In the inspector panel for the field, change the field attributes. You can't change the API name.
See [Reference: Table Field Attributes](#) on page 66.
6. Click **Next**.
7. (Optional) Change the **Table Display Name**.
8. (Optional) Select **Enable for Analysis** to create a Prism data source using the data in this table.
Workday recommends enabling a table for analysis after:
 - You apply the appropriate data source security to the table.
 - The table contains the data that you want and receives new and updated data from a data change task.
9. (Optional) Create or edit 1 or more **Tags** to organize the table in the Data Catalog.

Related Information**Concepts**

[Concept: Tables](#) on page 13

Reference

[Reference: Naming Guidelines](#) on page 56

Reference: Table Field Attributes

When you add or edit a field in a table in the Data Catalog, you define these attributes:

Field Attribute	Notes
Display Name	You can change this name at any time. The name must conform to name validation rules.
API Name	The API name must be unique in the table. Workday automatically selects an API name based on the field name you enter, modifying it to make it meet the name requirements. Click Change to change the API name. You can't change the API name after you save the table.
Field Type	Select the field type that the values in this field must match to be recognized as valid data. You need to configure additional field attributes for some field types you select.
Date Format	(Required for Date fields) Select the date format that the values in this field must match to be recognized as valid date data.
Digits Before and Digits After	(Required for Numeric fields) Enter the maximum number of digits before and after the decimal point that the values in this field can have to be recognized as valid numeric data. The sum of these 2 options must be less than or equal to 38.
Business Object	(Required for Instance and Multi-Instance fields) Select the business object to associate with the values in this Instance field.
Description	(Optional) Add a helpful field description that explains the meaning and data value characteristics of the field.
Required	Specifies that the field must contain data. Make a field required to ensure it doesn't contain a NULL value when you insert or update data in the table. When you insert or update data in a table and this field is NULL, Workday rejects the row and instead includes it in the error file.
Default Value	Use the Default Value to define a value for a field if the uploaded source file schema doesn't include that field. When the source file schema doesn't include a field, Workday uses the default value for all rows in the source file. Note: The Default Value is only used when the source file schema is missing a field, not when a particular field value is NULL.
Use as External ID	Use this attribute to mark a single field in a table as a key. Specify a field as the external ID when the values in the field uniquely identify each row from its source. Define a field as the external ID if you want to update or delete data in the table based on data in an external file. This attribute is similar to a primary key in a relational database. When using this attribute, consider:

Field Attribute	Notes
	- You can only define 1 field in a table as the external ID field. - Ensure that each field value in the external ID field is unique. If the field values aren't unique, you'll get unexpected results. Workday doesn't enforce the uniqueness. - You can't define a default value for fields used as an external ID. The field value must come from the external source and can't be NULL. - Workday automatically marks the external ID field as required.

Related Information**Concepts**

[Concept: Tables](#) on page 13

Reference

[Reference: Naming Guidelines](#) on page 56

Editing Datasets

Parse External Data in a Dataset

Prerequisites

- Base dataset using external data (from uploading a file or connecting to a server) exists in the Data Catalog.
- Security:
 - Prism Datasets: Create* domain in the Prism Analytics functional area when creating a dataset.
 - Any of these security requirements when editing an existing dataset:
 - Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - Dataset Editor* permission on the dataset.
 - Dataset Owner* permission on the dataset.

Context

When you bring external data into a dataset, you must describe the source data in a tabular format. You do this by describing how to parse the data.

Your data must:

- Be in plain text file format.
- Have 1 record per line.
- Have the same sequence of fields for every record separated by a common delimiter (such as a comma or tab).

Delimited records are separated by line breaks, and fields within a line are separated by a special character called the delimiter (usually a comma or tab character). If the delimiter also exists in the field values, it must be escaped. Datasets support single character escapes (such as a backslash), as well as enclosing field values in double quotes.

Steps

1. Access the **Edit Dataset Transformations** task for a base dataset using external data.
2. Edit the Parse stage.

As you complete this task, consider:

Option	Description
Row Delimiter	Specifies the single character that separates rows (or records) in your source data files. In most delimited files, rows are separated by a new line, such as the line feed character, carriage return character, or carriage return plus line feed. Line feed is the standard new line representation on UNIX-like operating systems. Other operating systems (such as Windows) might use carriage return individually, or carriage return plus line feed. Selecting Any New Line causes Workday to recognize any of these representations of a new line as the row delimiter.
Field Delimiter	Specifies the single character that separates the fields (or columns) of a row in your source data files. Comma and tab are the most common field delimiters.
Field Names	Specifies the default name of each field. You can change the field names in the Parse stage after you finish defining the parsing options. Field names must follow naming validation rules. Workday automatically treats the first line in each source file as a header row instead of as a row of data. If you don't want to use the first line as names for your fields, deselect the Field Names check box.
Escape Character	Specifies the single character used to escape the Quote Character or another instance of the Escape Character when a Quote Character is specified. Workday reads an escape character as data only if it's escaped with another escape character. If your data values contain quote characters as data, those characters must be escaped and the entire field value must be enclosed with the Quote Character. If not, then Workday assumes that the quote character denotes a new field. For comma-separated values (CSV) files, it's common practice to escape field delimiters by enclosing the entire field value within double quotes. If your source data uses this convention, then you should specify a Quote Character.
Quote Character	Some delimited files use the quote character to enclose individual data values. The quote character is typically the double quote character ("). If a field value contains a field delimiter as data, then the field value must be enclosed in the Quote Character, otherwise Workday assumes that the field delimiter denotes a new field. If a field value contains the quote character as data, then the field value must be enclosed in the Quote Character and it must

Option	Description
	be escaped, either by the Escape Character or another quote character. If a field value contains a row delimiter (such as a new line character) as data, then the field value must be enclosed in the Quote Character and Field values contain new lines must be selected. Suppose that you have a row with these 3 data values: <pre>weekly special wine, beer, and soda "2 for 1" or 9.99 each</pre> If the field delimiter is a comma, the quote character is a double quote, and the escape character is a backslash, then a correctly formatted row in the source data looks like: <pre>"weekly special","wine, beer, and soda","\"2 for 1\" or 9.99 each"</pre>
Rows to ignore	Specifies the number of lines at the beginning of the file to ignore when reading the source file while creating and publishing the dataset. To use this with the From First Table Row option, ensure that the line containing the field names is visible and is the first remaining line.
Field values contain new lines	Check this option if your source data might contain new line characters as part of a field value. When enabled, Workday reads the new line characters inside quote characters as part of the field value instead of as a row delimiter. Workday interprets any row delimiter character outside of quote characters as a new record. Enabling this option might impact the time to publish a dataset if Workday reads very large source files. Note that you might get unexpected results if you enable this option and the source file has malformed data (such as when a field value has either an opening or closing quote character, but not both). Try to ensure that your source data is well formed when using this option.
Trim trailing and leading whitespace characters in Text fields	Select this check box if you want to remove whitespace characters at the beginning and end of Text fields.

Related Information

Tasks

[Add a Stage to a Dataset](#) on page 72

[Steps: Create a Dataset with External Data \(SFTP Server\)](#) on page 41

[Steps: Create a Dataset with External Data \(Upload a File\)](#) on page 44

Reference

[Reference: Naming Guidelines](#) on page 56

Examples

[Example: Bring in International-Formatted Numeric Fields](#) on page 89

Add a Prism Calculated Field to a Dataset

Prerequisites

- Any of these security requirements:
 - Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - Dataset Editor* permission on the dataset
 - Dataset Owner* permission on the dataset.

Context

You can transform data in a dataset by adding Prism calculated fields to the dataset. Prism calculated fields change the number of fields in a dataset, they don't change the number of records.

A Prism calculated field contains these components:

Component	Description
Name	The name you specify is the field display name.
Expression	The expression describes a processing step that you want to perform on the data in other fields in the dataset. Expressions can include: - References to other fields as input. - Constant values as literal text, numeric, or date values. - Functions from the Prism expression language. - Arithmetic, comparison, and logical operators. For more information, see Concept: Prism Expression Language on page 140.
Field type	The expression determines the return value field type. If the expression includes a function, then the return value of the function determines the field type. If the expression doesn't include any function, then the operator determines the field type: - Arithmetic operators result in one of the numeric field types, depending on the input field types used in the expression. Example: Long * Integer results in a Long field, and Long * Numeric results in a Numeric field. Workday doesn't guarantee that each calculated field value will fit in the new field. Any calculated value that doesn't fit in the new field type becomes NULL. Workday automatically determines the digits before and after the decimal for Numeric field types. Numeric fields support a total of 38 digits before and after the decimal point, and a maximum of 18 digits after. - Comparison operators result in a Boolean field. - Logical operators result in a Boolean field.

You might need to create several Prism calculated fields to achieve the result you want. You can use the result of a Prism calculated field in the expressions of other Prism calculated fields to define a chain of processing steps.

You might want to use a Prism calculated field to:

Field Purpose	Example
Convert a field type to another field type.	Change an Integer field type to a Long field type, so that you can use the EPOCH_MS_TO_DATE function on it. Example: <pre>EPOCH_MS_TO_DATE(TO_LONG([Date in MS]))</pre>
Perform an arithmetic calculation.	Calculate the net profit based on the revenue and expenses. Example: <pre>[Revenue] - [Expenses]</pre> Calculate the percent of total revenue for a particular sale. Example: <pre>([Sale]/[Total Revenue])*100</pre>
Extract values from a different field.	Extract the currency codes from a currency field using the EXTRACT_CODE function. Example: <pre>EXTRACT_CODE([Revenue])</pre>
Combine the values from 2 Text fields into 1 Text field.	Combine separate fields consisting of Last Name and First Name into 1 field using the CONCAT function. Example: <pre>CONCAT([First Name], " ", [Last Name])</pre>
Test for a particular condition.	Test whether the year is between 2019 and 2020, inclusive. Example: <pre>[year] BETWEEN 2019 AND 2020</pre>
Pad the beginning of a Text field with leading zeros.	The [EEID] is a Text field containing numeric data of varying lengths, and you want to create an [Employee ID] field that is always 7 characters long, including leading zeros where needed. Create a calculated field called [PaddedID] that adds enough zeros to the beginning of [EEID] to create a full string, even if [EEID] is an empty string. Example: <pre>CONCAT("0000000", [EEID])</pre> Then create a calculated field called [Employee ID] that returns the last 7 characters of the [PaddedID] field. Example: <pre>SUBSTRING([PaddedID], (LENGTH([PaddedID])-7), 30)</pre>
Return a particular date given a Date field.	The [EFF_Date] field is a Date field, and you want to calculate and return the last day of the current month. <pre>DATE_ADD(DATE_ADD(TRUNC([EFF_Date], "month"), 1, "month"), -1, "day")</pre>

To delete a Prism calculated field, access the **Edit Dataset Transformations** task, and select the menu for the Prism calculated field you want to remove and select **Delete Field**. Deleting a field might cause errors if other Prism calculated fields refer to the deleted field.

Steps

1. Access the **Edit Dataset Transformations** task for a dataset.
2. Select a dataset pipeline (required for derived datasets) and stage into which to add the Prism calculated field.
3. Select **Add Field**.
4. Enter an expression in the expression editor.
If you use Currency fields that contain different codes, Workday treats the result of those calculations as NULL.
5. In the inspector panel, enter a name. Field names must follow naming validation rules.
6. Save the Prism calculated field by clicking **Enter** or **Return** on your keyboard.
Clicking another field on the page also saves the changes to the Prism calculated field.
7. (Optional) You can insert single and multiline comments into any location within a Prism expression.
Workday treats all text between these characters as comments: `/* */`
Note: Workday won't consider the data values as comments if you enclose these characters and the comment within double quotation marks.

Related Information

Concepts

[Concept: Prism Calculated Fields](#) on page 30

[Concept: Hiding Dataset Fields](#) on page 31

[Concept: Prism Expression Language](#) on page 140

Reference

[Reference: Naming Guidelines](#) on page 56

[The Next Level: Prism Analytics Best Practices](#)

Examples

[Example: Bring in International-Formatted Numeric Fields](#) on page 89

Add a Stage to a Dataset

Prerequisites

Any of these security requirements:

- *Prism Datasets: Manage* domain in the Prism Analytics functional area.
- *Dataset Editor* permission on the dataset.
- *Dataset Owner* permission on the dataset.

Context

Add stages to change your data from 1 format to another format.

One way you transform data in a dataset is by adding a stage. There are different types of stages. You can add a new stage to the end of any pipeline in your dataset. Some stage types can only be added to derived datasets.

Steps

1. Access the **Edit Dataset Transformations** task for a dataset.

2. (Required for derived datasets) Select the dataset pipeline where you want to add the stage.
Select the primary pipeline of a derived dataset for some stage types, such as Join or Union.
Base datasets have only 1 pipeline, which is the primary pipeline.
3. Click **Add Stage**, and select the type of stage to add.
4. Configure the stage parameters.

The parameters you define depend on the type of stage that you're adding.
5. (Optional) Add a description while editing any stage except the Import stage of a derived dataset.

Related Information

Reference

[Reference: Filter Stages](#) on page 79

[Reference: Group By Stages](#) on page 80

[Reference: Join Stages](#) on page 81

[Reference: Union Stages](#) on page 83

Manage Dataset Fields

Prerequisites

- *Prism Datasets: Create* domain in the Prism Analytics functional area when creating a dataset.
- Any of these security requirements:
 - *Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - *Dataset Editor* permission on the dataset.
 - *Dataset Owner* permission on the dataset.

Context

You can use the Manage Fields stage to view field changes, select fields, and edit fields.

Note: Decide on field names before you define Prism calculated fields and stages. Changing a field name later on will break Prism calculated field expressions and stages that rely on it.

Steps

1. Access the **Edit Dataset Transformations** task.
2. Add a Manage Fields stage to edit fields in the dataset.

Workday recommends that you add the Manage Fields stage at the:

- Beginning of the primary pipeline of a derived dataset.
- End of a primary pipeline that you intend to publish.

3. As you complete the Manage Fields stage, consider:

Option	Description
Input Name	Clear the check box to hide the field from future stages. You can hide fields to protect sensitive data or to use a calculated field instead of the fields that it's based on. Hide unpopulated or sparse fields or Prism calculated fields that do interim processing.
Output Name	Ensure that the new name: • Doesn't start with (or with WPA_.

Option	Description
	- Is unique within the dataset. - Is unique to any potential future field names that come from the source files. When you change a field name in the dataset and add a new field with the same name, you get unexpected results in the data when the schema updates.
Output Type	The field type determines which functions can use the field as an argument. Create a Prism calculated field to change a field type to a Date or Currency field type or to change a Currency field type to a numeric field type. For numeric types, you can specify the number of digits that go before and after the decimal point. For Instance and Multi-Instance types, you can select the business object name associated with the values in the field. Numeric field types include Integer, Long, Double, or Numeric.

Related Information

Concepts

[Concept: Hiding Dataset Fields](#) on page 31

[Concept: Prism Calculated Fields](#) on page 30

[Concept: Table and Dataset Field Types](#) on page 24

Tasks

[Add a Prism Calculated Field to a Dataset](#) on page 70

[Convert Dataset Text Fields to Date Fields](#) on page 75

Change Dataset Field Types

Prerequisites

- Any of these security requirements:
 - Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - Dataset Editor* permission on the dataset.
 - Dataset Owner* permission on the dataset.

Context

After importing external data into a dataset, if Workday assigned a different field type than the one you want, then you can change the field type. You might want to change the field type to:

- Accommodate some calculations you want to do.

Example: You could change an Integer field type to a Long field type to use the EPOCH_MS_TO_DATE function on it.

- Assign a field type to a value that Workday doesn't automatically recognize.

Example: If Workday doesn't automatically recognize an instance value, change it to the Instance field type.

Example: If Workday doesn't automatically recognize a date value, change it to the Date field type.

Use the Manage Fields stage to change most field types. However, create a Prism calculated field to make these field type changes:

From Field Type	To Field Type	Function
Text	Date	TO_DATE
Currency	Numeric, Double, Integer, or Long	EXTRACT_AMOUNT
Numeric, Double, Integer, or Long	Currency	BUILD_CURRENCY
Text	Currency	TO_CURRENCY
Instance, Multi-Instance	Multi-Instance	CREATE_MULTI_INSTANCE

Steps

1. Access the **Edit Dataset Transformations** task.
2. Select a dataset pipeline (required for derived datasets).
3. Add a Manage Fields stage.
4. Edit the Manage Fields stage by changing the field type in the Output Type drop-down menu.

Note: Every Instance field type must have a business object name. If you change the field type to Instance, click the settings icon and enter the business object name.

Related Information

Concepts

[Concept: Table and Dataset Field Types](#) on page 24

[Concept: Prism Calculated Fields](#) on page 30

Tasks

[Add a Prism Calculated Field to a Dataset](#) on page 70

Reference

[Reference: Currency Format Requirements for External Data](#) on page 59

Convert Dataset Text Fields to Date Fields

Prerequisites

- Dataset must have a field containing date data.
- Any of these security requirements:
 - *Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - *Dataset Editor* permission on the dataset.
 - *Dataset Owner* permission on the dataset.

Context

When you create a dataset using external data, the data might contain date or time information. When you create a dataset, Workday reads the data to determine which field type to assign to each field. If Workday recognizes the format of a date field, it automatically assigns the Date field type. However, Workday only

recognizes some date formats. If Workday doesn't recognize a date format, then it assigns the Text field type to the field.

If Workday doesn't recognize a date field format, you can create a Prism calculated field to convert the Text field type to a Date field type.

Steps

1. Access the **Edit Dataset Transformations** task for a dataset.
2. Select a dataset pipeline (required for derived datasets) and stage into which to add the Prism calculated field.
3. Add a new field.
4. In the expression editor for this new field, enter an expression that uses the `TO_DATE` function.
When you enter the expression using the `TO_DATE` function, make sure that you use the Text field you want to convert and the date format that best matches the values in that Text field. To quickly enter a field name, type a left square bracket (`[`).
Suppose that you have a field called `start_date` that contains data that looks like `25-May-2017`. Use this expression:
`TO_DATE([start_date], "dd-MMM-yyyy")`
5. In the inspector panel, enter a name. Field names must follow name validation rules.
6. Save the Prism calculated field by clicking **Enter** on your keyboard.
Clicking another field on the page also saves the changes to the Prism calculated field.

Result

Prism creates a new field and populates it by converting the data to the Date field type.

Related Information

Concepts

[Concept: Table and Dataset Field Types](#) on page 24

[Concept: Prism Calculated Fields](#) on page 30

Tasks

[Change Dataset Field Types](#) on page 74

[Add a Prism Calculated Field to a Dataset](#) on page 70

Reference

[Reference: Supported Date Formats for External Data in Tables and Datasets](#) on page 55

[Reference: Naming Guidelines](#) on page 56

Change the Dataset Example Rows

Prerequisites

- Any of these security requirements:
 - Prism Datasets: Manage* domain in the Prism Analytics functional area.
 - Prism Datasets: Owner Manage* domain in the Prism Analytics functional area.
 - Dataset Editor* permission on the dataset.
 - Dataset Owner* permission on the dataset.
 - Dataset Viewer* permission on the dataset and the tenant is configured to have the **Enable Prism Dataset Transformations** check box enabled.

- Any of these requirements:
 - *Dataset Viewer* permission on the source base dataset.
 - *Dataset Editor* permission on the source base dataset.
 - *Dataset Owner* permission on the source base dataset.
 - *Table Viewer* permission on the source table.
 - *Table Editor* permission on the source table.
 - *Table Owner* permission on the source table.

Context

Workday displays a subset of dataset rows to give you insight into your source data when you edit a dataset. Workday automatically displays the first 20,000 records starting with the first file in the dataset. You can change the example data to select between:

- **No Example**
- **Default**
- **Custom Example**

You can define rules to curate what data you see and how each stage impacts it. When you apply rules to the custom example data displayed, Workday reads the data from the files again and updates the statistics and field information for each field in the inspector panel.

You might want to create custom example rows to get more precise statistics on each field in the inspector panel.

You can disable all example data temporarily to improve responsiveness if your dataset has a lot of fields. After you make some changes to the dataset, such as adding a new stage or calculated field, you can then enable the example data by selecting the number of rows to display.

Note: If you have permissions to view the **View Dataset Transformations** report, then you can change example data but won't be able to save.

Steps

1. Access the **Edit Transformations** task for a derived dataset.
2. Select **Default Example**.
This field changes to reflect the current example option selected.
3. Select **Custom Example**.
4. Select **Add First Rule**.
You can add 1 rule per base dataset or table.
5. Select a source.
Workday displays the base datasets and tables that you have permission to view.
6. (Optional) Add a **Description** to help others understand how the rule impacts the data.
7. **Specify Conditions** by creating a conditional rule for the data.
You can add 1 or more conditions.
8. (Optional) Select **Convert to Advanced**.
Use the Prism expression language to write a boolean expression. For more information and examples, see [Reference: Boolean Expressions](#) on page 84.
You can convert rule conditions created in Basic mode to rule expressions in Advanced mode. If you switch back to Basic mode from Advanced mode, you need to define rule conditions again. Workday doesn't convert Advanced mode rule expressions to Basic mode rule conditions.
9. **Apply** the rule.
You can only save the rule if you have *Dataset Editor* permission for the current derived dataset.

10.(Optional) Download Example Data.

You can download the example data displayed to view a snapshot of how your stages impact the data before publishing.

Result

Workday applies the rule to the example data and displays rows that meet the defined conditions.

Next Steps

You can copy and paste the rules for a base dataset to other derived datasets using the same base dataset by selecting the related actions menu by the custom example rule.

Related Information

Reference

[Reference: Boolean Expressions](#) on page 84

[2022R2 What's New Post: Custom Examples for Prism Dataset Transformations](#)

Concept: Dataset Example Data

When you're editing dataset transformations, you can view how your changes affect an example of the data. This view enables you to see how different stages impact your data.

You can change the data displayed by selecting any of these example options:

Example Option	Description	Security Requirements
No Example	No data displayed. This option has the fastest load time.	<i>Dataset Editor</i> permission or better on the dataset.
Default Example	Workday displays a subset of the data, enabling you to assess how different stages affect a subset of your data.	<i>Dataset Editor</i> permission or better on the dataset.
Custom Example	This option is only available for derived datasets. You can create rules with different filter criteria for different subsets of your data. These rules enable you to curate what data you see and how each stage impacts it. You can add 1 rule per base dataset or table used in the derived dataset. The rules you define are specific to this derived dataset and don't affect other datasets with the same base dataset or table. When adding a filter stage, you can add more complex filter criteria similar to the Filter stage.	<i>Dataset Editor</i> permission or better on the derived dataset. - Any of these security requirements: <i>Dataset Viewer</i> permission or better on the base dataset. <i>Table Viewer</i> permission or better on the table.

Note: If you have permissions to view the **View Dataset Transformations** report, then you can change example data but won't be able to save.

Workday displays up to 20,000 rows when you select either Default or Custom Example.

Comparing Example Data to Published Data

Workday displays example data on a limited number of records, which typically is less than the number of records published in a dataset. If the dataset uses a Join stage and 1 or more Prism calculated fields after the Join stage, the example data displayed might be different than the published data. The published data is correct because it works on all data from the sources.

Example: Your dataset includes a Join stage using a right outer join and a Prism calculated field with a CASE function that evaluates the value in a field from the secondary (right) pipeline. The example data might not find a match and return NULL for the field value, whereas the published data finds a match and returns a non-null value. As a result, the CASE function returns a different value in the example data than in the published data.

Related Information

Reference

[2022R2 What's New Post: Custom Examples for Prism Dataset Transformations](#)

Reference: Filter Stages

You can convert filter conditions created in Basic mode to filter expressions in Advanced mode. If you switch back to Basic mode from Advanced mode, you need to define filter conditions again. Workday doesn't convert Advanced mode filter expressions to Basic mode filter conditions.

Edit Mode	Description
Basic	This mode displays prompts to help you define filter criteria, creating filter conditions. This mode is useful if you don't want to learn the details of the Prism Analytics expression language to write a filter expression manually.
Advanced	This mode enables you to define the filter criteria by writing an expression (the filter expression). Filter expressions must evaluate to true or false. This mode is useful if you want to write a filter expression that can't be expressed in Basic mode.

Basic Mode

Define these options when you configure Filter stages in Basic mode:

Filter Stage Option	Description
If All/If Any	If All—All filter conditions must be met for a row to remain in the output of the Filter stage. This option works like using an AND operator between each filter condition. If Any—Any filter condition can be met for the row to remain in the output of the Filter stage. This option works like using an OR operator between each filter condition.
Filter Condition	Click Add Filter to add a new filter condition. Select a field and operator from the prompts, and then enter a value in the empty text field. Workday reads the value in the text field exactly as is. You

Filter Stage Option	Description
	don't need to add any quotation marks or escape characters for Text field types. If the field you select is a currency field, you can enter an amount and select a currency code or select only a currency code. When the filter condition is configured as desired, click the check mark button to save the filter condition to the Filter stage. You can change the filter condition at any time by clicking its edit button.

Advanced Mode

In Advanced mode, use the Prism expression language to write a boolean expression. Example:
`TO_STRING([zip code]) LIKE("94*")`

For more information and examples, see [Reference: Boolean Expressions](#) on page 84.

You can insert single and multiline comments into any location within a Prism expression. Workday treats all text between these characters as comments: `/* */`

Note: Workday won't consider the data values as comments if you enclose these characters and the comment within double quotation marks.

Related Information

Concepts

[Concept: Dataset Stages](#) on page 22

Tasks

[Add a Stage to a Dataset](#) on page 72

Reference

[Reference: Boolean Expressions](#) on page 84

[Reference: Join Stages](#) on page 81

[Reference: Union Stages](#) on page 83

[Reference: Currency Format Requirements for External Data](#) on page 59

Reference: Group By Stages

As you complete this stage, consider:

Group By Stage Option	Description
Choose Grouping Fields	Select 1 or more fields by which to group values together. If you summarize (aggregate) values in a group that contains different currency codes, Workday returns NULL values.
Add Summarization Fields	Define 1 or more summarization types (aggregate functions) that apply to each grouping field.

You can select from these summarization types:

Summarization Type	Description
Average	Average returns the average of all valid numeric values for the specified grouping field. It sums

Summarization Type	Description
	all values in the specified field and divides by the number of valid (NOT NULL) rows. You can calculate the average on any numeric field.
Count	Count returns the number of rows for the specified grouping field.
Max	Max returns the largest (maximum) value from the specified grouping field. You can calculate the maximum on any numeric or date field.
Min	Min returns the smallest (minimum) value from the specified grouping field. You can calculate the minimum on any numeric or date field.
Sum	Sum returns the total of all values from the specified grouping field. You can calculate the sum on any numeric field.

Related Information

Concepts

[Concept: Dataset Stages](#) on page 22

Tasks

[Add a Stage to a Dataset](#) on page 72

Reference

[Reference: Filter Stages](#) on page 79

[Reference: Union Stages](#) on page 83

[The Next Level: Prism Analytics Best Practices](#)

Reference: Join Stages

You can include 2 datasets (dataset pipelines) in a Join stage. You can add additional Join stages in the pipeline if you need to join multiple datasets.

Join Stage Option	Description
Join Pipeline	Select a dataset pipeline to join with the primary pipeline. If there aren't any pipelines available, select Add Another Pipeline to create a pipeline by importing a dataset.
Match Rows	Select 1 or more fields from each dataset pipeline whose values should match each other. Select the fields that uniquely identify rows in each dataset pipeline. Defining the matching rows is similar to defining a primary key/foreign key relationship in relational database terms. You can also select an optional suggestions link that displays Workday recommended join keys. The suggestions are based on the number of matching rows between fields in the example data.

Join Stage Option	Description
Join Type	Select the join type. The join type specifies which rows from each dataset pipeline to include in the join result. • Inner Join. Workday includes rows that have matching values that exist in both pipelines. If a row from 1 pipeline doesn't match a row in the other pipeline, the row is omitted from the join result. • Left Outer Join. Workday includes all rows in the Primary pipeline and searches for a matching row in the other pipeline. If there's no matching row in the other pipeline, Workday populates each field from the other pipeline with NULL values. When the imported pipeline includes multiple matching rows, then Workday includes both rows in the join result. • Right Outer Join. Workday includes all rows in the imported dataset pipeline and searches for a matching row in the Primary pipeline. If there's no matching row in the Primary pipeline, Workday populates each field from the Primary pipeline with NULL values. When the Primary pipeline includes multiple matching rows, then Workday includes both rows in the join result. • Full Outer Join. Workday includes all rows from both pipelines. If a row from 1 pipeline doesn't match a row in the other pipeline, Workday populates each field from the nonmatching pipeline with NULL values. When there are multiple matching rows, Workday includes all rows in the join result. Note that Workday replaces all NULL values with default values when you publish the dataset.
Select Fields	Select which fields from each pipeline to include in the join result. Any field you don't include is dropped from that stage in the pipeline and all later stages.

Related Information

Concepts

[Concept: Dataset Stages](#) on page 22

Tasks

[Add a Stage to a Dataset](#) on page 72

Reference

[Reference: Filter Stages](#) on page 79

[Reference: Group By Stages](#) on page 80

[The Next Level: Prism Performance and Troubleshooting Tips](#)

[2023R1 What's New Post: Join Stage Suggestions](#)

Reference: Union Stages

You can include 2 datasets (dataset pipelines) in a Union stage, and add additional Union stages in the pipeline if you need to combine multiple datasets. If you're familiar with SQL, the Union stage is the equivalent of a UNION ALL operator.

Define at least 1 set of matched fields (field mapping) in a Union stage before saving it. If you don't specify a field in an input dataset for a field mapping, Workday will use a NULL value in that field from that input dataset.

Union Stage Option	Description
Union Pipeline	Select a dataset pipeline to combine with the primary pipeline. If there aren't any pipelines available, select Add Another Pipeline to create a pipeline by importing a dataset.
Match Fields—Union Output	The name of the field that will be output from the Union stage for each field mapping.
Match Fields—Primary Pipeline	The field from the primary pipeline to match with a field from the union pipeline.
Match Fields— <i>Union pipeline name</i>	The field from the union pipeline to match with a field from the primary pipeline.
Rematch	Click this button if you want to discard all field mappings and return to the default field mappings that Workday detects and configures.
Clear All	Click this button to discard all field mappings. Then define at least 1 field mapping in the Union stage before saving it.
Include All	Click this button to create a field mapping for each field in the input dataset pipeline. Then you can select which fields to match from the other input dataset pipeline.

Preparing Fields from Input Datasets

The datasets you want to combine in a Union stage might not have the same schema. Example: One dataset might have first name and last name information in a single field, and the other has that information in 2 fields. When this is the case, you can create Prism calculated fields for each input dataset so that you can combine them in a Union stage.

You can create Prism calculated fields for input datasets in these locations:

- In the original dataset. Any Prism calculated field you create in a dataset is available to both that dataset and any derived dataset that imports it as an input dataset.
- In the dataset pipeline of the derived dataset. Any Prism calculated field you create in a pipeline stage of a derived dataset is available only to that derived dataset. It doesn't get pushed back to the original input dataset. You might want to create a Prism calculated field in the pipeline of an imported dataset if you need a field to use a different field type, but you don't want to change the field type of the original dataset. Example: You could create a Prism calculated field to change a zip code field from Integer to Text to match it with a Text zip code field in another input dataset.

Related Information

Concepts

[Concept: Dataset Stages](#) on page 22

Tasks

[Add a Stage to a Dataset](#) on page 72

Reference

[Reference: Filter Stages](#) on page 79

[Reference: Group By Stages](#) on page 80

[Reference: Join Stages](#) on page 81

Reference: Unpivot Stages

For Workday to convert fields (columns) to rows in derived datasets, follow these requirements:

Requirement	Description
Number of Input Fields	Select at least 2 input fields to unpivot. Enter the same number of input fields for every group of input fields that you unpivot in a single Unpivot stage. Example: If you select 5 input fields to unpivot, each subsequent group of input fields that you unpivot in the stage must contain 5 input fields. If you unpivot multiple groups of input fields, the total number of input fields you can use for all groups together in a single Unpivot stage is 150.
Input Field Types	Select input fields that are the same field type, such as Text or Numeric. You can unpivot all field types. If you select input fields consisting of Instance or Multi-Instance fields, these input fields must use the same business object. If you select input fields consisting of Numeric fields, these input fields must have the same number of digits specified before and after the decimal point.
Output Field Names	Enter unique names for all pairs of new fields created from unpivoting.

Related Information**Concepts**

[Concept: Unpivot Stages](#) on page 23

Reference

[2020R2 What's New Post: Unpivot Stage](#)

Examples

[Example: Unpivot Stock Vesting Data in a Dataset](#) on page 87

Reference: Boolean Expressions

A boolean expression is an expression that evaluates to true or false. You can use the Prism expression language to write boolean expressions in:

- Filter expressions. Use Advanced mode when configuring a dataset Filter stage.
- Custom example rule expressions. Use Advanced mode when configuring a rule for the dataset custom example.
- Prism calculated fields. Example: Use a boolean expression in the CASE function.

The typical format of a boolean expression is:

```
field_name comparison_operator comparison_value
```

The comparison value must be of the same field type as the field in the expression.

You can also use logical operators (such as AND and OR) or arithmetic operators (such as + or /) to define more complex expressions.

When the field name includes a space or a special character, enclose the field name in square brackets: [ticker symbol].

When the comparison value is for a Text field type, enclose the value in double quotes (""). Example:

```
[Zip Code] = "94111-5224"
```

Comments in Expressions

You can insert single and multiline comments into any location within a Prism expression. Workday treats all text between these characters as comments: /* */

Note: Workday won't consider the data values as comments if you enclose these characters and the comment within double quotation marks.

Text Field Examples

```
[movie title] LIKE ("* and the*")
[contingent workers] IN ("Larry", "Curly", "Moe")
schedule NOT IN ("Saturday", "Sunday")
status IS NOT NULL
```

Numeric and Currency Field Examples

```
TO_STRING([zip code]) LIKE("94*")
[sale price] < 50.00
age >= 21
((EXTRACT_AMOUNT([Total Sales]) > 1000) AND (EXTRACT_CODE_TEXT([Total
Sales]) == "USD"))
```

Date Field Examples

```
[purchase date] BETWEEN 2019-06-01T00:00:00.000Z AND
2019-07-31T00:00:00.000Z
[graduation] >= 1990-01-01
```

Instance and Multi-Instance Field Examples

```

INSTANCE_IS_SUPERSET_OF([Cost Center - Manager], [Cost Center])

[Cost Center] IS EMPTY

[Journal Lines] IS NOT NULL

[Cost Center 1] != [Cost Center 2]

NOT INSTANCE_EQUALS([Cost Center 1], [Cost Center 2])

INSTANCE_CONTAINS_ANY([Regions], "070b0d082eee44e1928c808cc739b35f",
"f4c49debb3dc483baa8707dfe683503c")

```

Boolean Expressions on Date Type Fields

Boolean expressions on Date type fields must be in either of these formats:

```

yyyy-MM-ddTHH:mm:ss:SSSZ
yyyy-MM-dd

```

Don't enclose comparison values for Date fields in quotation marks or use any other punctuation. If the date value is in Text format rather than Date format, the value must be enclosed in quotes like all text values.

When specifying a range of dates, always write the earlier date first.

If the boolean expression is a shortened version of the full format, then any values not included are assigned a value of zero (0). Example: the expression BETWEEN 2019-06-01 AND 2019-07-31 is equivalent to this expression:

```
BETWEEN 2019-06-01T00:00:00.000Z AND 2019-07-31T00:00:00.000Z
```

The expression above doesn't include any values from July 31, 2019. To include values from July 31, 2019, use BETWEEN 2019-06-01 AND 2019-08-01.

Boolean Expressions on Currency Type Fields

Currency type fields in boolean expressions must be in a format recognized by Workday. If a Currency field contains any value that doesn't meet these requirements, Workday treats the value as NULL.

You can use multiple currency codes. Example:

```

((EXTRACT_CODE_TEXT([Annual Salary]) == "EUR") AND (EXTRACT_AMOUNT([Annual
Salary]) >= 90000))
OR
((EXTRACT_CODE_TEXT([Annual Salary]) == "USD") AND (EXTRACT_AMOUNT([Annual
Salary]) >= 100000))

```

Related Information

Concepts

[Concept: Prism Expression Language](#) on page 140

Tasks

[Change the Dataset Example Rows](#) on page 76

Reference

[Reference: Filter Stages](#) on page 79

[Comparison Operators](#) on page 142

Example: Unpivot Stock Vesting Data in a Dataset

This example illustrates how to create an Unpivot stage in a dataset to transpose fields (columns) of data into rows of data.

Context

You have a CSV file of stock vesting data for your workers. The workers' stock vests in 3 installments, with a different number of shares on each date. The file contains 1 row of data for each worker, and a separate field for each vesting date and the number of shares that vested on each date.

The CSV file contains these rows and fields:

Name	Vest Date 01	QTY Vest 01	Vest Date 02	QTY Vest 02	Vest Date 03	QTY Vest 03
Dominique	04/01/2020	80	07/01/2020	85	10/01/2020	92
Desmond	02/01/2021	70	05/03/2021	65	08/02/2021	79
Stacey	02/01/2021	90	05/03/2021	80	08/02/2021	90

You need to transpose the data so that you have:

- A field that contains all vesting dates for each worker.
- A field that contains the number of shares that vested on each date for each worker.

Prerequisites

Create a table by uploading a CSV file using the data in this example. See [Steps: Create a Table by File Upload](#) on page 35.

Create a derived dataset from the table. See [Steps: Create a Derived Dataset](#) on page 48.

Security: *Prism Datasets: Manage* domain in the Prism Analytics functional area.

Steps

1. From the **View Dataset Details** report of the derived dataset, click **Edit**.
2. Click **Add Stage**, and select **Unpivot**.

First, we'll unpivot the 3 date input fields, Vest_Date_01, Vest_Date_02, and Vest_Date_03.

3. Click the plus sign next to **Output Values** so you have a total of 3 pairs of prompts.
4. Select these values in the **Input Fields** and **Output Values** prompts:

Input Fields	Output Values
Vest_Date_01	Vesting Date 01
Vest_Date_02	Vesting Date 02
Vest_Date_03	Vesting Date 03

5. In **Field Names from Input**, enter Vesting Schedule.
6. In **Values from Output**, enter Vesting Dates.

7. Click **Preview** to see the results of the settings you've made so far.

On the **Output Fields** tab, Workday displays the results of the changes you defined in the Unpivot stage. Workday:

- Removes the input fields you selected.
- Displays the 2 new fields using the field names you defined.
- Displays the new rows created using data from the removed input fields. In this example, Workday displays 9 rows total.

8. Click the plus sign in the upper right corner of the Unpivot stage panel. Clicking the plus sign creates another group of input fields that you can unpivot into rows.

Next, we'll unpivot the 3 quantity input fields, QTY_Vest_01, QTY_Vest_02, and QTY_Vest_03.

9. Click the plus sign next to **Output Values** so you have a total of 3 pairs of prompts.

10. In the **Input Fields** and **Output Values** prompts, enter these values:

Input Fields	Output Values
QTY_Vest_01	Quantity Vesting 01
QTY_Vest_02	Quantity Vesting 02
QTY_Vest_03	Quantity Vesting 03

11. In **Field Names from Input**, enter Quantity Schedule.

12. In **Values from Output**, enter Number of Shares.

13. Click **Preview** to see the results of the settings you've made so far.

On the **Output Fields** tab, Workday displays the results of the changes you defined in the Unpivot stage.

14. Click **Done**, and click **Save**.

Result

The output of the Unpivot stage contains these rows and fields:

Name	Vesting Schedule	Vesting Dates	Quantity Schedule	Number of Shares
Dominique	Vesting Date 01	04/01/2020	QTY Vesting 01	80
Dominique	Vesting Date 02	07/01/2020	QTY Vesting 02	85
Dominique	Vesting Date 03	10/01/2020	QTY Vesting 03	92
Desmond	Vesting Date 01	02/01/2021	QTY Vesting 01	70
Desmond	Vesting Date 02	05/03/2021	QTY Vesting 02	65
Desmond	Vesting Date 03	08/02/2021	QTY Vesting 03	79
Stacey	Vesting Date 01	02/01/2021	QTY Vesting 01	90
Stacey	Vesting Date 02	05/03/2021	QTY Vesting 02	80
Stacey	Vesting Date 03	08/02/2021	QTY Vesting 03	90

Next Steps

(Optional) Add a Manage Fields stage after the Unpivot stage to hide the Quantity Schedule field. In this example, the Quantity Schedule field contains data that is redundant with the Vesting Schedule field.

Related Information**Concepts**

[Concept: Unpivot Stages](#) on page 23

Reference

[Reference: Unpivot Stages](#) on page 84

[2020R2 What's New Post: Unpivot Stage](#)

Example: Bring in International-Formatted Numeric Fields

This example describes how to use Prism Analytics to bring in external data that includes a numeric field containing International-formatted numbers.

Context

Your company has a delimited file with numeric data formatted using periods to separate thousands and commas to separate decimals. You want to bring this data into Prism Analytics as a Numeric field so that you can perform calculations on the data. However, Workday only recognizes data in an external file as valid numeric data when it:

- Uses a period as the decimal mark.
- Doesn't use any character to separate thousands.

You have a comma delimited (CSV) file with this data:

```
Key, Amount
1, "9.000,1222-"
2, "11.111,2333"
3, "9.999.999,34-"
4, "7.777.777,45"
5, "8.888,56"
```

In your file, no value has more than 4 digits after the decimal mark.

Prerequisites

Create a table by file upload, using the CSV file to define the source schema. The table should have these fields:

- **Key** of type Numeric
- **Amount** of type Text

Create a derived dataset based on the table, and use DDS Intl as the dataset name.

Security:

- *Prism: Tables Create* domain in the Prism Analytics functional area.
- *Prism: Manage File Containers* domain in the Prism Analytics functional area.
- *Prism Datasets: Create* domain in the Prism Analytics functional area.
- *Prism Datasets: Manage* domain in the Prism Analytics functional area.

Steps

1. Access the **Edit Dataset Transformations** task for the DDS Intl dataset.

2. Create a Prism calculated field that removes the thousands separators by replacing any period with an empty string.

- a) Select **Add Field**.

- b) Enter this expression in the expression editor, and press Enter or Return on your keyboard:

```
REGEX_REPLACE([Amount], "\.", "")
```

- c) In the inspector panel, enter this as the field name: **Amount no thousands**

3. Create a Prism calculated field that replaces the decimal comma with a decimal period.

- a) Select **Add Field**.

- b) Enter this expression in the expression editor, and press Enter or Return on your keyboard:

```
REGEX_REPLACE([Amount no thousands], ",", ".")
```

- c) In the inspector panel, enter this as the field name: **Amount decimal period**

4. Create a Prism calculated field that adds any negative sign that exists at the end of the value to the front of the value.

- a) Select **Add Field**.

- b) Enter this expression in the expression editor, and press Enter or Return on your keyboard:

```
CASE WHEN SUBSTRING([Amount decimal period], LENGTH([Amount decimal period])-1, LENGTH([Amount decimal period])) = "-"
THEN CONCAT("-", REGEX_REPLACE([Amount decimal period], ",", ""))
ELSE [Amount decimal period]
END
```

- c) In the inspector panel, enter this as the field name: **Amount minus sign**

5. Create a Prism calculated field that removes any minus sign at the end of the value.

- a) Select **Add Field**.

- b) Enter this expression in the expression editor, and press Enter or Return on your keyboard:

```
CASE WHEN SUBSTRING([Amount minus sign], LENGTH([Amount minus sign])-1, LENGTH([Amount minus sign])) = "-"
THEN SUBSTRING([Amount minus sign], 0, LENGTH([Amount minus sign])-1)
ELSE [Amount minus sign]
END
```

- c) In the inspector panel, enter this as the field name: **Amount final**

6. Create a Prism calculated field that converts the [Amount final] Text field to a Numeric field.

- a) Select **Add Field**.

- b) Enter this expression in the expression editor, and press Enter or Return on your keyboard:

```
CAST([Amount final] AS Decimal(20,4))
```

- c) In the inspector panel, enter this as the field name: **Amount Numeric**

7. Select **Add Stage** and then select **Manage Fields**.

8. Hide these fields by clicking the eye icon:

- Amount
- Amount no thousands
- Amount decimal period
- Amount minus sign
- Amount final

9. Select **Done**.

10. **Save** your dataset.

Result

The dataset contains these fields:

- Key as type Numeric
- Amount Numeric as type Numeric, with 20 digits before the decimal and 4 digits after.

Changing Data in Tables

Create a Data Change Task

Prerequisites

Security:

- *Prism: Manage File Containers* domain in the Prism Analytics functional area when uploading a file.
- Any of these security requirements:
 - *Prism: Tables Owner Manage* domain in the Prism Analytics functional area.
 - *Prism: Tables Manage* domain in the Prism Analytics functional area.
 - *Table Owner* permission on the table.
 - *Table Editor* permission on the table.
 - *Can Delete Table Data* permission on the table.
 - *Can Insert Table Data* permission on the table.
 - *Can Truncate Table Data* permission on the table.
 - *Can Update Table Data* permission on the table.

Context

You can change the rows of data in a table by creating and running a data change task. You can:

- Insert new rows.
- Update specific rows based on a key field.
- Delete specific rows based on a key field.
- Insert new and update existing rows in the same activity, known as an upsert operation.

How you change the rows in the table depends on the operation you select, such as upsert or delete, and the source data. The source you specify must contain the rows you want to change in the table, such as new rows to insert, or existing rows to update or delete.

To create, edit, or view a data change task on a target table, you must have permission on the target table. Example: To create a data change task using the upsert operation, you must have both insert and update permission on the target table.

Steps

1. Access the **Data Catalog** report.
2. Select **Create > Data Change Task**.

You can change the data change task name at the top of the left side panel.

3. On the **Source** step, select the source that contains the data you want to change in the target table. Workday uses the file upload source type by default. Click **Change Connection** to select a different source type. As you select a source type, consider:

Option	Description
Data Catalog	Select an existing dataset or table.
File Upload	Select 1 or more delimited files. When you upload multiple files, each file must use the same schema. Workday supports RFC 4180-compliant delimited files. For more information, see RFC 4180. When you select file upload as the source type, configure the Source Options to define how to parse the file.
SFTP	Select an SFTP connection that connects to an SFTP server containing 1 or more delimited files. When the SFTP connection contains multiple files, each file must use the same schema. Workday supports RFC 4180-compliant delimited files. For more information, see RFC 4180. For SFTP sources, you must also upload 1 local delimited file that uses the same schema as the files on the SFTP server. Workday uses the uploaded file to determine how to parse the SFTP files. Workday doesn't load the data from the uploaded file to the table. When you select SFTP as the source type, configure the Source Options to define how to parse the uploaded file.
Workday Report	Select an existing Workday custom report. Workday displays reports that meet the eligibility requirements for importing into tables. See Concept: Creating Reports to Import into Tables and Datasets on page 10.

4. On the **Source Options** step, define how to parse the data in the files you uploaded for file upload or SFTP sources.
- [Parse External Data in a Table](#) on page 62.
 - Review the fields Workday created based on the parsed file, and modify the fields if necessary. Select a field in the list and view the field details in the inspector panel on the right side. You might need to:
 - Change the **Field Type** when Workday assigns the wrong field type. Workday assigns the field type based on the first few rows only. Example: Workday assigns the Numeric field type to a field with ZIP code data because the example rows that it evaluates only contain numerals. However, based on your knowledge of the source data, you know that some ZIP code values contain letters or a hyphen, so you change the field type to Text.
 - Change other field attributes based on the field type to ensure that Workday correctly parses the data, such as Digits Before, Digits After, or Date Format.

See [Reference: Table Field Attributes](#) on page 66.

5. On the **Target** step, select the target table that contains the data you want to change.
6. Select the target operation to perform on the target table using the source data.

Option	Description
Insert	Workday keeps the existing data in the table and adds the new data in the source. This operation is also known as Append.
Truncate and Insert	Workday deletes the existing data in the table and replaces it with the data in the source. This operation is also known as Replace.
Delete	Workday deletes a row from the table when it matches a row in the source.
Update	Workday updates a row in the table when it matches a row in the source. To select this operation, you must use a table where you configured 1 field as the external ID.
Upsert	Workday updates a row in the table when a matching row already exists and inserts the row when it doesn't exist. To select this operation, you must use a table where you configured 1 field as the external ID.

7. On the **Mapping** step, select a field in the target table to use as the operation key for delete, update, or upsert operations.

You can specify 1 of these target table fields:

- The field configured as the external ID in the table: You can use this field for delete, update, or upsert operations.
- WPA_LoadID: You can use this field for delete or update operations only.
- WPA_RowID: You can use this field for delete or update operations only.

8. Select a source field for each target field that you want to modify. Workday requires that you map any field used as the operation key.

Workday lists the source fields that are compatible for a specific target field. If Workday doesn't list a source field you want, verify the source field attributes, such as the digits before, digits after, or business object. You can navigate to the **Source Options** step to change the field attributes for file upload and SFTP sources.

You can click **Reset Matches** to revert all mappings you changed to the simple match algorithm that Workday uses by default. The simple match algorithm:

- Is case insensitive.
- Ignores spaces.
- Ignores underscore characters.
- Matches on the field API name.
- Won't match fields with different field types.

9. On the **Review** step, verify the information. You can go back to any previous step and make any correction if necessary.

10. Click **Finish** and select **Run Without Saving**, **Save and Run Now**, or **Save**.

Result

When you save the data change task, Workday creates the data change task object and displays it on the **Data Change Tasks** tab of the **Data Catalog** report.

When you run the data change task, Workday starts a data change activity to change the data in the table based on the data in the source. You can view the data change activity progress and history on:

- The **Activities** tab of the **View Table Details** report. To fix errors on a table following a data change, download the error file from the data change activity. Workday only creates an error file for data change activities that use a file upload as the source.
- The **Data Change Activities** tab of the **Data Catalog** report.
- The **Prism Activities Monitor** report. You can view a detailed list of all types of Prism-related activities for a given date range.
- The **Prism Activities Dashboard** report. You can view an active dashboard of all Prism-related activities for the last 180 days. You can also see live updates to the status of activities and their run times.

Next Steps

To fix errors on a table following a data change, download the error file from the data change activity on the **Activities** tab of the **View Table Details** report. Workday only creates an error file for data change activities that use file upload or SFTP sources.

Related Information

Concepts

[Concept: Data Change Tasks](#) on page 96

[Concept: Mapping Fields in Data Change Tasks](#) on page 97

[Concept: Creating Reports to Import into Tables and Datasets](#) on page 10

Tasks

[Create an SFTP Connection](#) on page 97

Reference

[Reference: Table Field Attributes](#) on page 66

[Reference: External Data Limits](#) on page 12

[2022R1 What's New Post: Report Sources for Data Change Tasks](#)

[2022R1 What's New Post: SFTP Sources for Data Change Tasks](#)

[2021R2 What's New Post: Prism Analytics Data Management](#)

Create a Data Change Task Schedule

Prerequisites

Security:

- *Prism Datasets: Owner Manage* domain in the Prism Analytics functional area.
- *Prism Datasets: Manage* domain in the Prism Analytics functional area.
- *Table Editor* permission on the table.
- *Table Owner* permission on the table.
- *Can Delete Table Data* permission on the table.
- *Can Insert Table Data* permission on the table.
- *Can Truncate Table Data* permission on the table.
- *Can Update Table Data* permission on the table.

Context

You can create a schedule for running a data change task. You can:

- Create a schedule for data change tasks that use any source type other than file upload.
- Create 1 active schedule per data change task.
- Define schedules to run on a recurring basis (Example: daily, weekly, or monthly).
- Define schedules to run only if another Prism scheduled process completes at a status you specify.

Steps

1. Access the **View Data Change Task** report for the data change task you want to create a schedule for.
2. Select **Data Change Task > Create Schedule** from the related actions menu.
3. In **Run Frequency**, specify how often to run the data change task.

The choices include creating a dependent schedule.

4. Select the criteria for the schedule.
5. (Recurring schedules) As you configure the schedule, consider:

Option	Description
Priority	Unavailable for data change task schedules.
Catch Up Behavior	Select how many times the scheduled data change runs after maintenance issues cause errors.

6. (Dependent schedules) As you configure the schedule, consider:

Option	Description
Dependency	Select a Prism-related schedule on which the data change task schedule depends.
Trigger on Status	Select the status of the scheduled future process that causes the data change task to run. Workday recommends using one of the completed statuses.
Time Delayed Configurations	(Optional) Specify the number of days, hours, or minutes to delay running the data change task after the trigger.

Result

Workday runs the data change task based on the criteria that you specify, creating a data change activity. View the status of all scheduled processes on the:

- **Data Change Activities** tab of the **Data Catalog** report.
- **Prism Activities Monitor** report.
- **Prism Activities Dashboard** report.

Workday assigns a name to the schedule based on the name of the data change task and prepends *Execute Data Change Schedule:* to the name. You can change the name of the schedule in the **Request Name** field when you edit the schedule. Workday displays this name in the **Prism Activities Monitor**, **Prism Activities Dashboard**, **Process Monitor**, and **Scheduled Future Processes** reports to help you identify a specific process request.

Next Steps

From the related actions menu of the data change task, you can:

- View the schedule details.
- Edit the schedule.
- Expire the schedule.

Related Information

Reference

[Reference: External Data Limits](#) on page 12
[2021R2 What's New Post: Data Change Tasks](#)

Concept: Data Change Tasks

A data change task is a Prism Analytics object that defines how to change the data in a Prism Analytics table using data from a specified source. You can save data change tasks and run them at any time.

Create a data change task to output the transformed data from a derived dataset into a table. Example: You have a base dataset with data from a custom report, and you have a derived dataset that transforms the data in the base dataset. You can create a data change task that uses data from the derived dataset and loads it into a table.

You specify a data operation type, such as insert or delete, that instructs how to use the data in the source to change the table rows. You can create 1 or more data change tasks per table.

You create, edit, and manage data change tasks from the Data Catalog. Access the **Data Change Tasks** tab on the **Data Catalog** report to view all data change tasks you have permission on.

To create, view, or edit a data change task for a specific table, you must have permission to modify data in that table. Example: To create a data change task on the Claims table using the upsert operation, you must have permission to insert and update data in the Claims table.

You can:

- Run a data change task on a schedule, or manually on an ad hoc basis.
- Edit the data change task, such as change the operation type or field mapping, before you run it.
- Delete the data change task.
- View the data change activities to see the number of rows affected by the data change task and to download any failed rows.

You can use data change tasks to define these components:

Component	Description
Source	The object that contains the data that you want to load into the target table. Example: A delimited file that you upload, or an SFTP server, or a Data Catalog object, such as a dataset. Depending on the source type, you might need to define additional source options, such as parsing options.
Target table	The table with the data you want to change using the source data. You can only create a data change task using an existing table in the Data Catalog.
Target operation	The data operation to perform on the target table using the source data. Workday lists the operations supported for the selected source type. You can specify: • Insert • Delete • Truncate and Insert • Update • Upsert
Mapping	The specifications for matching source fields to the target table fields. You can match a source field to 1 or more target table fields.

Data Change Activities

When you run a data change task, Workday generates a data change activity. Each data change activity modifies the data in the table, based on the operation specified in the data change task and the data in the source.

You can view previously run data change activities on:

- The **Data Change Activities** tab on the **Data Catalog** report. The Data Change Activities tab displays all activities for all data change tasks and tables.
- The **Activities** tab on the **View Table Details** report. The Activities tab displays all activities for that table, both from data change activities and other data load activities. Example: You might have a different data load activity if you created the table from a file upload.
- The **Prism Activities Dashboard** report in the **Data Catalog**. The report displays all activity for the last 180 days. You can also download error reports from the report.

Related Information

Reference

[2021R2 What's New Post: Prism Analytics Data Management](#)

Concept: Mapping Fields in Data Change Tasks

When you create or edit a data change task, you define how to map fields from the source to target table fields. When you define the field mappings:

- You can map some or all target fields.
- Workday marks the target field specified as the delete key, update key, or upsert key as required. When the target operation is delete, the only field you need to map is the delete key field.
- Workday displays source fields that are compatible with the target field.

Numeric fields are compatible when the digits before and after the decimal in the source field are less than or equal to the digits in the target field. Example: You can map an Integer or Numeric (8,0) source field to a Numeric(10,0) or Numeric(12,2) target field, but you can't map a Numeric(20,6) to a Numeric(20,5).

Instance and Multi-Instance fields are compatible when the business object for both fields are the same.

Related Information

Reference

[2021R2 What's New Post: Prism Analytics Data Management](#)

Managing Connections

Create an SFTP Connection

Prerequisites

Security: *Prism: Manage Connection* domain in the Prism Analytics functional area.

Context

You can create a connection to an SFTP server so that you can use it as a source in a data change task to load external data into a table. SFTP connections:

- Can be used in 1 or more data change tasks as the source.
- Define how to connect to and authenticate against the server.

- Define the files to get from the server.

When you run a data change task that uses the SFTP connection as the source, Workday connects to the SFTP server using the configured authentication credentials and fetches the specified files. For the data change activity to succeed:

- The number of files must be less than 5,000.
- The time to transfer the data must be less than 5 hours.

Each file from the server should be less than 1 GB compressed (less than 10 GB uncompressed approximately).

Use the **Test Prism SFTP Connection** task to verify that Workday can connect to the SFTP server and retrieve files.

Steps

1. Access the **Create Prism SFTP Connection** task.

You can also select **Create > Connection** from the Data Catalog.

2. As you complete this task, consider:

Option	Description
Name	Workday displays the name in the Data Catalog. You can change the name at any time.
Description	Enter a description that describes the SFTP server and configured file pattern.
File Name/Pattern	Enter the filename or a filename pattern that represents 1 or more files. The filename is case-sensitive. You can use the asterisk (*) and question mark (?) characters as wild cards to specify a filename pattern. Use the asterisk (*) to specify zero or more characters, and use the question mark (?) to specify exactly 1 character.
SFTP Address	Use this format: <code>sftp://domain_name</code> or <code>sftp://IP_address</code> To specify a port number, add it to the end of the domain name or IP address. If you don't specify a port number, Workday uses port 22.
Directory	The directory on the server that contains the files. Directory names are case-sensitive. Include a leading slash (/) only for a full path, not a relative path.
Use Temp File	Writes the imported data to a temporary file in Workday with a randomly generated name. After the data import is complete, Workday automatically renames the file to the correct name. You might want to enable this option if the data import takes a very long time and might not finish before the next scheduled time to import data from the same server.
Host Key Fingerprint	The encryption key that the SFTP server will use for SSH communications.
Authentication Method and Details	Select the type of security authentication that the SFTP server uses: • User Name / Password.

Option	Description
	• SSH Authentication. This option uses secure shell key authentication using X.509 certificates.
Delete After Retrieval	Deletes the files on the SFTP server after the data is loaded into the target table. If Workday is unable to delete the files from the SFTP server, the data retrieval fails.
Decompress	Don't enable this option for data change tasks and tables. You can transfer files that are compressed or not. For compressed files, Workday only supports gzip compression.
Decrypt Using	If you want to decrypt the imported files using Pretty Good Privacy (PGP), select a PGP Private Key Pair.
Restricted To	Select the environment in which you want to use the settings defined in the Transport section. If you leave this option empty, Workday applies the transport settings to each environment in which the data change activity runs. When a data change activity runs in a particular environment, such as Implementation or Production, the transport settings only work if the Restricted To option matches the current environment. When the current environment and the configured environment don't match, the data change activity fails and retrieves no files from the SFTP server. You might want to restrict the transport settings to a particular environment to avoid inadvertently transferring test data to a non-test endpoint. Example: You create the SFTP connection and a data change task in an Implementation environment and select Implementation in Restricted To. Later, you migrate this SFTP connection and data change task to a Production environment. The next time the data change activity runs, it fails. To ensure that the data change activity runs successfully in the Production environment, edit the SFTP connection and either clear the Restricted To option or change it to Production.

Result

Workday creates the SFTP connection and displays it on the **Connections** tab in the Data Catalog.

Next Steps

Create a data change task using the SFTP connection as the source to load data into the table.

Related Information

Tasks

[Create a Data Change Task](#) on page 91

Reference

[Reference: Naming Guidelines](#) on page 56

[2022R1 What's New Post: SFTP Sources for Data Change Tasks](#)

Securing Data in Tables and Datasets

Set Up Table Sharing

Prerequisites

Security: *Set Up: Assignable Roles* domain in the Organization and Roles functional area.

Context

To enable sharing a table with another user or security group, you create tenant-specific roles that correspond to the table-related Workday provided roles. Sharing tables is a way to control access to individual tables.

Steps

1. Access the **Maintain Assignable Roles** task.
2. Set up roles for the table-related Workday-provided roles.

When you set up these tenant-specific roles, consider:

- The name you enter in **Role Name** becomes the prompt value in the **Permission** column on the **Edit Table Sharing** task.
- The security groups you select in **Role Assignees Restricted To** determine which users and groups you can share a table with.
- The security groups you select in **Assigned by Security Groups** determine which users can share a table.

Add the roles in the table below, but substitute security groups that your organization needs instead of Prism Data Writer and Prism Data Administrator:

Role Name	Workday Role	Enabled for	Self-Assign	Role Assignees Restricted to	Assigned by Security Groups
Table Owner - Prism	01. Table Owner - Prism	Prism Tables	Yes	Prism Data Writer	Prism Data Administrator Prism Table Owner (Workday Owned) Security Administrator
Table Editor - Prism	02. Table Editor - Prism	Prism Tables		Prism Data Writer	Prism Data Administrator Prism Table Owner (Workday Owned)

Role Name	Workday Role	Enabled for	Self-Assign	Role Assignees Restricted to	Assigned by Security Groups
					Security Administrator
Table Schema Editor - Prism	03. Table Schema Editor - Prism	Prism Tables		Prism Data Writer	Prism Data Administrator Prism Table Owner (Workday Owned) Security Administrator
Table Viewer - Prism	04. Table Viewer - Prism	Prism Tables		Prism Data Writer	Prism Data Administrator Prism Table Owner (Workday Owned) Security Administrator
Table Schema Viewer - Prism	05. Table Schema Viewer - Prism	Prism Tables		Prism Data Writer	Prism Data Administrator Prism Table Owner (Workday Owned) Security Administrator
Can Truncate Table Data - Prism	06. Can Truncate Table Data - Prism	Prism Tables		Prism Data Writer	Prism Data Administrator Prism Table Owner (Workday Owned) Security Administrator
Can Delete Table Data - Prism	07. Can Delete Table Data - Prism	Prism Tables		Prism Data Writer	Prism Data Administrator Prism Table Owner (Workday Owned)

Role Name	Workday Role	Enabled for	Self-Assign	Role Assignees Restricted to	Assigned by Security Groups
					Security Administrator
Can Update Table Data - Prism	08. Can Update Table Data - Prism	Prism Tables		Prism Data Writer	Prism Data Administrator Prism Table Owner (Workday Owned) Security Administrator
Can Insert Table Data - Prism	09. Can Insert Table Data - Prism	Prism Tables		Prism Data Writer	Prism Data Administrator Prism Table Owner (Workday Owned) Security Administrator
Can Select Table Data - Prism	10. Can Select Table Data - Prism	Prism Tables		Prism Data Writer	Prism Data Administrator Prism Table Owner (Workday Owned) Security Administrator

Note: You can substitute any user-based security groups your organization has created for Prism-related tasks instead of selecting Prism Data Writer or Prism Data Administrator.

Related Information

Concepts

[Concept: Sharing Tables and Datasets](#) on page 111

Tasks

[Share a Table with Others](#) on page 104

Set Up Dataset Sharing

Prerequisites

Security: *Set Up: Assignable Roles* domain in the Organization and Roles functional area.

Context

To enable sharing a dataset with another user or security group, you create tenant-specific roles that correspond to the dataset-related Workday provided roles. Sharing datasets is a way to control access to individual datasets.

Steps

1. Access the **Maintain Assignable Roles** task.
2. Set up roles for these Workday-provided roles: *Prism Dataset Owner*, *Prism Dataset Editor*, and *Prism Dataset Viewer*.

When you set up these tenant-specific roles, consider:

- The name you enter in **Role Name** becomes the prompt value in the **Permission** column on the **Edit Dataset Sharing** task.
- The security groups you select in **Role Assignees Restricted To** determine which users and groups you can share a dataset with.
- The security groups you select in **Assigned by Security Groups** determine which users can share a dataset.

Add the roles in the table below, but substitute security groups that your organization needs instead of Prism Data Writer and Prism Data Administrator:

Role Name	Workday Role	Enabled for	Self-Assign	Role Assignees Restricted to	Assigned by Security Groups
Dataset Owner	Prism Dataset Owner	Prism Dataset	Yes	Prism Data Writer	Prism Data Administrator Prism Dataset Owner (Workday Owned) Security Administrator
Dataset Editor	Prism Dataset Editor	Prism Dataset		Prism Data Writer	Prism Data Administrator Prism Dataset Owner (Workday Owned) Security Administrator
Dataset Viewer	Prism Dataset Viewer	Prism Dataset		Prism Data Writer	Prism Data Administrator Prism Dataset Owner (Workday Owned)

Role Name	Workday Role	Enabled for	Self-Assign	Role Assignees Restricted to	Assigned by Security Groups
					Security Administrator

Note: You can substitute any user-based security groups your organization has created for Prism-related tasks instead of selecting Prism Data Writer or Prism Data Administrator.

Related Information

Concepts

[Concept: Sharing Tables and Datasets](#) on page 111

Tasks

[Share a Dataset with Others](#) on page 105

Share a Table with Others

Prerequisites

Any of these security requirements:

- *Prism: Tables Owner Manage* domain in the Prism Analytics functional area.
- *Table Owner* permission on the table.

Context

When you create a table, you're the table owner. Being the owner of a table means you have **Table Owner** permission on the table. As a table owner, you can share it with other users and security groups. The table sharing feature is a way to control access to individual tables.

You might want to share a table with someone else so they can edit it, or import it into a derived dataset.

Table permissions control either:

- Metadata. Example: Table Schema Editor, Table Schema Viewer.
- Data. Example: Can Delete Table Data, Can Truncate Table Data, Can Select Table Data, Can Insert Table Data.
- Metadata and Data. Example: Table Owner, Table Editor, Table Viewer.

Steps

1. Access the **View Table Details** report.
2. Select **Actions > Security > Edit Table Sharing**.
3. Assign a user or security group to the desired **Permission**.

Note: When you grant someone one of the data permissions, that user must have access to view the table schema. Example: If you want to grant Can Insert Table Data permission, you must also grant either Table Schema Viewer or Table Viewer permission.

4. (Optional) Share the table with more users, or remove access from users listed in the table.
5. When you're done sharing the table with others, click **OK**.

Related Information

Concepts

[Concept: Sharing Tables and Datasets](#) on page 111

[Concept: Relax Sharing Options](#) on page 112

Concept: [Sharing Datasets Using Relax Sharing Rules](#) on page 114

Tasks

[Set Up Table Sharing](#) on page 100

Share a Dataset with Others

Prerequisites

Any of these security requirements:

- *Prism Datasets: Owner Manage* domain in the Prism Analytics functional area.
- *Dataset Owner* permission on the dataset.

Context

When you create a dataset, you're the dataset owner. Being the owner of a dataset means you have **Dataset Owner** permission on the dataset. As a dataset owner, you can share it with other users and security groups. The dataset sharing feature is a way to control access to individual datasets.

To share a derived dataset, you need owner permission on the current dataset and either:

- Relax Sharing Rules enabled and functional on the current dataset, or
- At least viewer permission on all upstream objects up until you reach the base datasets and tables, or until you reach a dataset with Relax Sharing Rules enabled and functional.

You might want to share a dataset with someone else so they can edit it, or import it into another (derived) dataset.

Note: If you want to share a derived dataset with someone, they must also have at least viewer permission on all upstream objects up until you reach the base datasets and tables, or until you reach a dataset with Relax Sharing Rules enabled and functional.

Steps

1. Access the **View Dataset Details** report.
2. Select **Actions > Security > Edit Dataset Sharing**.
3. Add a new row to the table, and select the **Permission** to assign to a user or security group.

You can configure these permissions:

Option	Description
Dataset Viewer	Users with this permission can: • View this dataset. • Import this dataset into a derived dataset.
Dataset Editor	User with this permission can do all tasks of a dataset viewer plus: • Make changes to (edit) this dataset.
Dataset Owner	User with this permission can do all tasks of a dataset editor plus: • Delete this dataset. • Share this dataset.

4. (Optional) Share the dataset with more users, or remove access from users listed in the table.
5. When you're done sharing the dataset with others, click **OK**.

Related Information**Concepts**

[Concept: Security in Prism Analytics](#) on page 108

[Concept: Relax Sharing Options](#) on page 112

[Concept: Sharing Datasets Using Relax Sharing Rules](#) on page 114

Edit Prism Data Source Security

Prerequisites

- Security: *Prism: Manage Data Source* domain in the Prism Analytics functional area.

Context

Before you make Prism data in a table or dataset available for analysis, configure the security (security domains and securing entities) that Workday applies to the data in the Prism data source. You configure the data source security by editing the table or dataset, but Workday applies the security to the data in the Prism data source.

The configured securing entities work with the configured security domains and their security groups to determine which users have access to which rows, fields, and field values in a Prism data source.

A securing entity is an Instance or Multi-Instance field that you use to constrain access to particular instance values for reporting and analytics. A securing entity:

- Is typically a role-enabled Instance field, such as Cost Center or Supervisory Organization.
- Is the Person Instance field (secured to the *Person Data: Person Reports* domain) for self-service security groups.
- Determines which instance values Workday displays to a user based on the role assigned to the user.

Use securing entities to control row-level and field value-level access in a Prism data source for users in constrained security groups.

For a user to have access to a particular row or field value in a Prism data source, they must be a member of 1 of these security groups:

- An unconstrained security group that has permissions on a domain configured in the data source security.
- A constrained security group that has permissions on a domain configured in the data source security, and the corresponding securing entity is configured.

Workday restricts user access to data in a Prism data source for these security groups:

- All unconstrained
- Role-based constrained
- Aggregation when role-based
- Intersection when role-based

Workday has tested and supports using securing entity fields that use these business objects:

- Company
- Company Hierarchy
- Cost Center
- Cost Center Hierarchy
- Person
- Location Hierarchy
- Region
- Region Hierarchy

- Supervisory Organization

Steps

1. Access the **Edit Data Source Security** task for the table or dataset you want to apply security to.
2. In the **Domains** prompt, select 1 or more security domains to use to determine who can see the Prism data source.

If you specify a security domain that has a constrained security group, then you must specify an appropriate securing entity.

3. (Optional) In the **Securing Entities** prompt, select 1 or more fields in the dataset. Workday lists the Instance or Multi-Instance fields in the table or dataset that act as securing entities.

The securing entities work with the:

- **Data Source Security** domains to determine row-level access for a user.
- **Field Level Security** domains to determine field value-level access for a user.

Note: Workday uses any in common logic when evaluating the contextual security using a Multi-Instance field.

Note: When you specify more than 1 securing entity that relates to the same security group, Workday uses the OR condition between them. Depending on how your security groups are set up, a user might see some additional rows or field values. Make sure you test the report results to ensure that the report produces expected results for each user.

4. In the **Default Domain(s) for Dataset Fields** prompt, select 1 or more security domains that Workday applies to every field in the Prism data source unless you override the domain for a particular field in the next section.

Note: When you add new fields to the table or dataset, Workday applies this default domain to the new fields. You might want to consider specifying a domain with more restrictive access. Then you can override the default domain on a per field basis to allow more access as necessary.

5. (Optional) You can select different domains to apply to specific fields to override the default domains.
6. Review any **Security Configuration Audit** messages to learn more about any issues with the configured securing entities and domains.
7. (Optional) Click **Back** to make any changes to the configured security options based on the audit messages.
8. Select the **Apply Security** check box to apply your changes.

If you want to restrict access to rows using any of these security group types, Workday can't honor those restrictions:

- Segment-based security groups
- Job-based security groups
- Manager's Manager security group

Result

Workday saves the security information. You can view the current security status by selecting **Actions > Security > View Data Source Security**.

Example

Suppose that you select these domains containing these security groups. To enforce contextual security at the row-level and field value-level, then use these fields as securing entities:

Security Domain	Contains This Security Group	Use This Securing Entity
Custom Domain 28	HR Partner (By Location)	Location
Custom Domain 29	Manager	Supervisory Organization
Custom Domain 30	HR Administrator	None required. HR Administrator is an unconstrained security group, so it doesn't require a securing entity.
Public Reporting Items	None.	None required. This domain provides access to all publicly available fields and Workday-delivered data sources.

Next Steps

Create the Prism data source by enabling the table for analysis or publishing the dataset. Workday applies the security restrictions to the data in the Prism data source.

Concept: Security in Prism Analytics

Prism Analytics uses Workday's strong, flexible, and configurable security model to control access to data, objects, and tasks. Which users have access to what data depends on where in the Prism data workflow they are. For more information on the workflow, see [Concept: Prism Analytics Data Management Workflow](#) on page 7.

Phase 1 and Phase 2: Create and Edit Tables, Datasets, and Data Change Tasks

In the first and second phases in the data management workflow, you bring data into the Prism Analytics Data Catalog and then transform it. You create and edit these objects in the Data Catalog:

- Tables. Tables include metadata and all data rows in the table.
- Derived datasets. Datasets include metadata and a subset of data as a small collection of example rows.
- Data change tasks. Data change tasks include a small collection of example rows from the target table, and from the source if it's another dataset or table.

When it comes to security with tables, data change tasks, and datasets, you control access to the metadata and data together.

Table and Dataset Security

Workday controls who can do what with tables and datasets in these ways:

Method	Notes
Security administrator grants access	Your Workday security administrator can configure Workday security domains to grant groups of users to be able to create, edit, or manage tables and datasets. Depending on how the administrator configures the security domains, some users might be able to create, view, edit, or act as an owner of all tables and datasets. Your Workday security administrator can configure these Workday security domains to grant access to groups of users: • <i>Prism Datasets: Create</i> • <i>Prism Datasets: Manage</i>

Method	Notes
	• <i>Prism Datasets: Owner Manage</i> • <i>Prism: Manage Data Source</i> • <i>Prism: Tables Create</i> • <i>Prism: Tables Manage</i> • <i>Prism: Tables Owner Manage</i>
Table sharing and dataset sharing	The user who created a table or dataset can share it with particular users and grant different levels of access to each user. When you create a table or dataset, you're the table owner or dataset owner. Being the owner means that you have Table Owner or Dataset Owner permission on the table or dataset. As an owner, you can grant different levels of access by assigning permissions to another user. Example: You can assign Table Viewer or Can Truncate Table Data permission to a table, or Dataset Editor permission to a dataset.

Access to a table or dataset is unconstrained. This means that any user who can create or view a table or dataset can view all fields and data (example data for datasets and data change tasks), regardless of the origin of the data. When you create a table or base dataset from a Workday report, Workday removes all security domains configured for the business objects in the table or dataset.

This unconstrained access only applies when you use these tasks and reports:

- View Table Details
- Edit Table
- View Dataset Details
- Edit Dataset Transformations
- View Dataset Transformations
- Create Data Change Task
- Edit Data Change Task

It doesn't apply to the data in a Prism data source.

Instead, you define the data source security to apply to the data in the Prism data source before making the data available for analysis.

Data Change Task Security

You can't configure access on a data change task directly. Instead, Workday controls your access to a data change task based on:

- Your target table permissions. Workday uses these table permission types:
 - Table Viewer
 - Table Editor
 - Table Owner
 - Can Delete Table Data
 - Can Insert Table Data
 - Can Update Table Data
- The specified operation type. You need permission to change data in the table that is compatible with the specified operation type. Example: To create or edit a data change task on the Claims table using the upsert operation, you must have permission to insert and update data in the Claims table.
- Your source access. You need permission to view the source, such as view permission on a custom report, source dataset, or SFTP connection. You don't need view permission on the source when the type is file upload.

Workday doesn't enable users to have view permission only on a data change task. If you have permission to edit a data change task, then you have permission to view it.

If you meet the source access requirement, then you can perform the actions below with the specified target table permissions:

	Table Viewer Only	Table Viewer and Can Delete/Insert/Update Table Data	Table Editor or Table Owner
View a data change task.	No	Yes	Yes
Create a data change task.	No	Yes, but you can only select an operation type that is compatible with your table permissions.	Yes
Edit the data change task operation type.	No	Yes, but you can only select an operation type that is compatible with your table permissions.	Yes
Edit the data change task source.	No	Yes, but you need permission on the new source.	Yes, but you need permission on the new source.
Edit other data change task properties (not the source or operation).	No	Yes	Yes

Phase 3: Apply Security to the Data

You define the security that Workday will apply to the Prism data source before you make the data in the Data Catalog available for analysis. By applying security to the Prism data source, you can ensure that both Workday and non-Workday data have the proper restrictions applied when viewed in a discovery board or report.

You configure the data source security by editing the table or dataset, but Workday applies the security to the data in the Prism data source.

Define the data source security after the data is ready to be exposed to other users by giving them access to the Prism data source.

To configure the data source security, you must access to the *Prism: Manage Data Source* security domain. For details, see [Edit Prism Data Source Security](#) on page 106.

You can restrict access to the data in a Prism data source at these levels:

- **Data source-level.** Specify 1 or more security domains that apply to the Prism data source that Workday creates. These domains determine which users can see the Prism data source. This is sometimes known as table-level security. If you don't configure any domain, Workday uses the *Prism: Default to Dataset Access* domain.
- **Row-level.** Optionally, you can enforce row-level security by specifying in the **Securing Entities** prompt 1 or more Instance or Multi-Instance fields in the table or dataset, such as Supervisory Organization. The securing entities work with the configured data source-level security domains to determine which users have access to which rows (and field values) in a Prism data source.
- **Field-level.** Specify 1 or more security domains to apply to the fields in the Prism data source. These domains determine which users can see each field in the Prism data source.
- **Field value-level.** Workday uses any configured **Securing Entities** with the configured field-level security domains to determine which users have access to which field values in a Prism data source.

Phase 4: Make the Data Available for Analysis

When you make Prism data available for analysis, Workday creates the Prism data source, loads it with the transformed data, and applies the appropriate security restrictions to the data.

The way you create a Prism data source and the security required depend on the Data Catalog object. For more information, see [Concept: Making Prism Data Available for Analysis](#) on page 125.

Optionally, you can create a Prism data source without any data source security configured. When no data source security is configured, Workday applies the *Prism: Default to Dataset Access* security domain to the Prism data source. The *Prism: Default to Dataset Access* domain provides contextual access to a Prism data source based on your access to the underlying table or dataset.

Phase 5: Analyze and Visualize the Data

Workday controls access to data in a Prism data source according to the configured data source security. The configured data source security determines who can see the Prism data source, and which rows, fields, and field values each user can see when they query the Prism data source in a discovery board or report.

Related Information

Tasks

[Share a Dataset with Others](#) on page 105

[Edit Prism Data Source Security](#) on page 106

Concept: Sharing Tables and Datasets

Workday enables you to have fine-grained control over what you can do with tables and datasets. Sharing tables and datasets is a way to control access to individual tables and datasets.

When your tenant is set up for table and dataset sharing, table owners and dataset owners can share a table or dataset with another user or security group. Example: You can control who can view a dataset, edit the schema of a table, insert data into a table, or delete table data.

How Inherited Permissions Work to Enable Table and Dataset Sharing

Workday provides sharing permission control using Workday roles, Workday-owned security groups, and inherited permissions.

Most Workday roles are tied to an organization. However, table-related and dataset-related Workday roles are tied to an object type, the table or dataset. By tying a role to an object type, Workday enables you to control which permissions a user inherits for a particular table or dataset.

Workday maps each table-related and dataset-related role to a Workday owned security group, and that security group automatically inherits permissions from 1 or more security domains.

Example: Workday maps the Workday role "Table Schema Editor" to the "Prism Table Schema Editor (Workday Owned)" security group, and that security group inherits View and Modify permissions on the *Prism: Tables Manage Schema* domain.

Example: Workday maps the Workday role "Prism Dataset Editor" to the "Prism Dataset Editor (Workday Owned)" security group, and that security group inherits View and Modify permissions on the *Prism: Datasets Manage* domain.

As a result of these connections, you can enable table and dataset sharing by creating a tenant-specific role and mapping it to a table-related or dataset-related Workday role. The tenant-specific role becomes the table or dataset permission that you can share with others.

Related Information

Tasks

[Share a Table with Others](#) on page 104

[Share a Dataset with Others](#) on page 105

Concept: Relax Sharing Options

By default, to access the current dataset, you must have access to all upstream datasets and tables.

Example: To share a derived dataset, you must have owner permission on the derived dataset and at least viewer permission on all upstream datasets and tables.

However, you can relax some permission requirements so that users require access on fewer upstream datasets and tables to have access to the current dataset.

To relax the sharing restrictions with tables and datasets, Workday provides these options when you edit dataset sharing and table sharing:

Option Name	Description
Relax Sharing Rules	When you enable Relax Sharing Rules on a table or dataset: • Owners of datasets derived from this table or dataset only need a viewer role on this table or dataset to share their derived dataset. • (Derived datasets only) Owners of this dataset can share it with other users without requiring those users to have a role on any upstream datasets or tables. You can protect sensitive data in upstream datasets and tables by eliminating the need for access by using Relax Sharing Rules in downstream datasets.
Prevent Relax Sharing on Derived Datasets	When you enable Prevent Relax Sharing on Derived Datasets, Workday revokes the Relax Sharing Rules functionality on all downstream derived datasets. You might want to enable Prevent Relax Sharing on Derived Datasets to prevent others from sharing derived datasets they own without requiring your permission on this table or dataset. Note: The Relax Sharing Rules option is only functional when no dataset or table upstream from it has Prevent Relax Sharing Rules on Downstream Datasets enabled.

To configure these options, you must have access to the *Prism: Manage Relax Sharing* domain in the Prism Analytics functional area.

Example: You create a table called Payrolls that contains sensitive data, and you create a derived dataset off of it called Filtered Payrolls that filters out the sensitive data. You want Norman Chan to view the Filtered Payrolls dataset, but not the Payrolls table. To accomplish this sharing scenario, you enable Relax Sharing Rules on Filtered Payrolls, and then assign Norman Chan viewer permission on it. Now, Norman can view Filtered Payrolls and create a derived dataset from it even though he doesn't have viewer permission on the Payrolls table. Also, he can share the derived dataset he created without having owner permission on Filtered Payrolls or Payrolls.

To do this...	You must have...
- View the current dataset details or table. - Import the current dataset or table into a derived dataset.	At least viewer permission on the current dataset or table, and either: - Relax Sharing Rules enabled and functional on the current dataset or table, or - At least viewer permission on all upstream objects up until you reach the base datasets and tables, or until you reach a dataset with Relax Sharing Rules enabled and functional.
- View the current dataset transformations. - Copy the current dataset (derived datasets only).	At least viewer permission on the current dataset, and either: - Relax Sharing Rules enabled and functional on the current dataset or table and at least viewer permission on each dataset or table that is imported into the current dataset, or - At least viewer permission on all upstream objects up until you reach the base datasets and tables, or until you reach a dataset with Relax Sharing Rules enabled and functional.
Edit the current dataset or table.	At least editor permission on the current dataset or table, and either: - Relax Sharing Rules enabled and functional on the current dataset or table and at least viewer permission on each dataset or table that is imported into the current dataset, or - At least viewer permission on all upstream objects up until you reach the base datasets and tables, or until you reach a dataset with Relax Sharing Rules enabled and functional.
Share the current dataset or table (the action that is specific to owners).	Owner permission on the current dataset or table, and either: - Relax Sharing Rules enabled and functional on the current dataset or table, or - At least viewer permission on all upstream objects up until you reach the base datasets and tables, or until you reach a dataset with Relax Sharing Rules enabled and functional.
View the lineage of the current table or dataset.	At least viewer permission on the current dataset or table. Note that when you view lineage, you only see the upstream or downstream tables and dataset on which you have viewer permission (or better).

Related Information

Tasks

[Share a Table with Others](#) on page 104

[Share a Dataset with Others](#) on page 105

Concept: Sharing Datasets Using Relax Sharing Rules

The relax sharing options work with table and dataset sharing permissions to determine which users can access a table or dataset. The kind of access users have determines their ability to:

- View the table or dataset.
- Make a copy of the table or dataset.
- Import the table or dataset into a derived dataset.
- Edit the table or dataset.
- Share the table or dataset with other users.

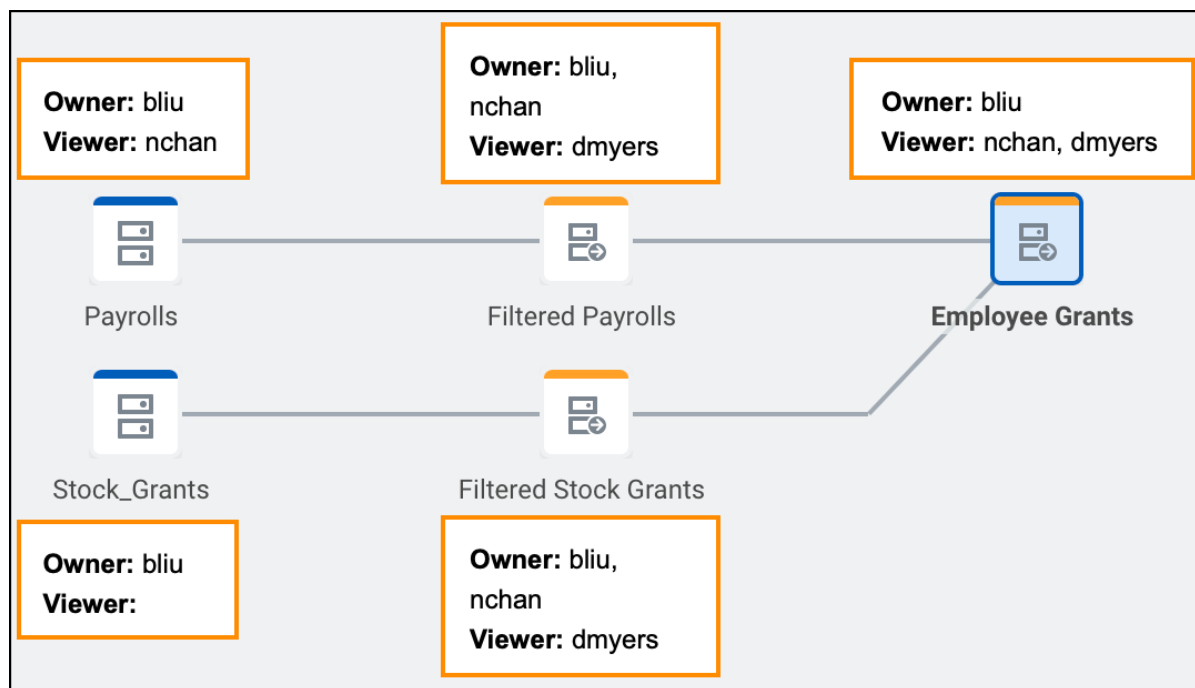
Scenario 1

In the first scenario, Beth Liu made multiple tables and datasets, and shared some of them with Norman Chan and Dawn Myers.

The Data Catalog includes these tables and datasets:

Table or Dataset	Owners	Viewers	Relax Sharing Rules Enabled	Prevent Relax Sharing on Derived Datasets Enabled
Payrolls (table)	bliu	nchan	No	No
Filtered Payrolls	bliu, nchan	dmyers	No	No
Stock Grants (table)	bliu	None	No	No
Filtered Stock Grants	bliu, nchan	dmyers	No	No
Employee Grants	bliu	nchan, dmyers	No	No

The dataset lineage for the Employee Grants dataset looks like this:



Beth Liu, Norman Chan, and Dawn Myers have this table and dataset access:

Table or Dataset	Beth Liu	Norman Chan	Dawn Myers
Payrolls	Can share, edit, and view	Can view	No access
Filtered Payrolls	Can share, edit, and view	Can edit, can view	No access
Stock Grants	Can share, edit, and view	No access	No access
Filtered Stock Grants	Can share, edit, and view	No access	No access
Employee Grants	Can share, edit, and view	No access	No access

Although Norman Chan is an owner of Filtered Payrolls, he can't share it with others because he only has View permission on the upstream dataset Payrolls.

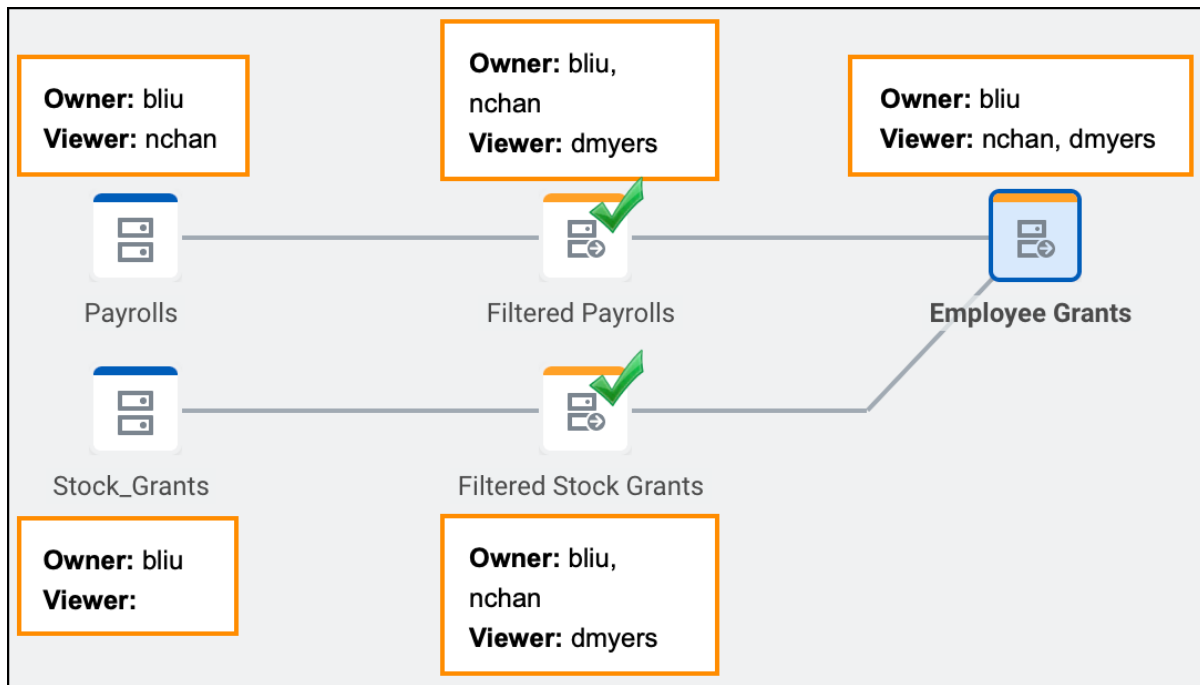
Scenario 2

This scenario is based on the previous scenario. Beth Liu enabled the Relax Sharing Rules check box on the Filtered Payrolls and Filtered Stock Grants datasets.

The Data Catalog includes these tables and datasets:

Table or Dataset	Owners	Viewers	Relax Sharing Rules Enabled	Prevent Relax Sharing on Derived Datasets Enabled
Payrolls	bliu	nchan	No	No
Filtered Payrolls	bliu, nchan	dmyers	Yes	No
Stock Grants	bliu	None	No	No
Filtered Stock Grants	bliu, nchan	dmyers	Yes	No
Employee Grants	bliu	nchan, dmyers	No	No

The dataset lineage for the Employee Grants dataset looks like this. The green check marks indicate that Relax Sharing Rules is enabled.



Beth Liu, Norman Chan, and Dawn Myers have this table and dataset access:

Table or Dataset	Beth Liu	Norman Chan	Dawn Myers
Payrolls	Can share, edit, and view	Can view	No access
Filtered Payrolls	Can share, edit, and view	Can share, edit, and view	Can view
Stock Grants	Can share, edit, and view	No access	No access
Filtered Stock Grants	Can share, edit, and view	Can share and view	Can view
Employee Grants	Can share, edit, and view	Can view	Can view

With Relax Sharing Rules enabled on Filtered Payrolls and Filtered Stock Grants:

- Norman can share Filtered Payrolls because he no longer needs owner permission on the upstream object Payrolls.
- Norman can share Filtered Stock Grants because he no longer needs owner permission on the upstream object Stock Grants, but he can't edit Filtered Stock Grants because he doesn't have viewer permission on the upstream object Stock Grants.
- Norman can view Employee Grants because he no longer needs viewer permission on all upstream objects.
- Dawn can view Filtered Payrolls, Filtered Stock Grants, and Employee Grants because she no longer needs viewer permission on all upstream objects.

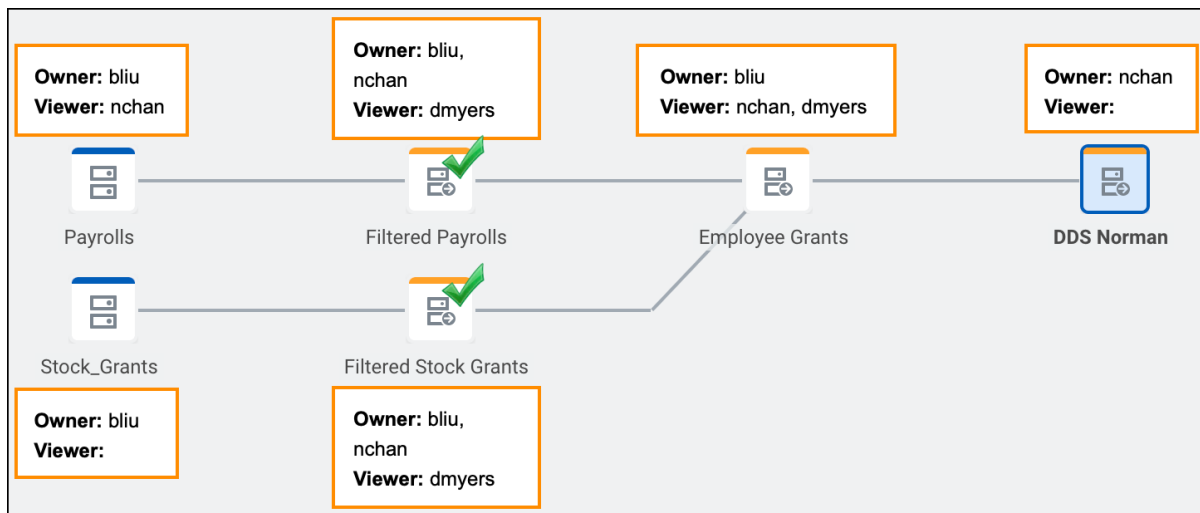
Scenario 3

This scenario is based on the previous scenario. Norman Chan created a derived dataset titled DDS Norman by importing the Employee Grants dataset.

The Data Catalog includes these tables and datasets:

Table or Dataset	Owners	Viewers	Relax Sharing Rules Enabled	Prevent Relax Sharing on Derived Datasets Enabled
Payrolls	bliu	nchan	No	No
Filtered Payrolls	bliu, nchan	dmyers	Yes	No
Stock Grants	bliu	None	No	No
Filtered Stock Grants	bliu, nchan	dmyers	Yes	No
Employee Grants	bliu	nchan, dmyers	No	No
DDS Norman	nchan	None	No	No

The dataset lineage for the DDS Norman dataset (as viewed by a Prism Data Administrator who can view all objects) looks like this:



Beth Liu, Norman Chan, and Dawn Myers have this table and dataset access:

Table or Dataset	Beth Liu	Norman Chan	Dawn Myers
Payrolls	Can share, edit, and view	Can view	No access
Filtered Payrolls	Can share, edit, and view	Can share, edit, and view	Can view
Stock Grants	Can share, edit, and view	No access	No access
Filtered Stock Grants	Can share, edit, and view	Can share and view	Can view
Employee Grants	Can share, edit, and view	Can view	Can view
DDS Norman	No access	Can edit, view	No access

Norman Chan didn't share DDS Norman with Beth Liu, so Beth doesn't see it in her Data Catalog and she doesn't know that it exists.

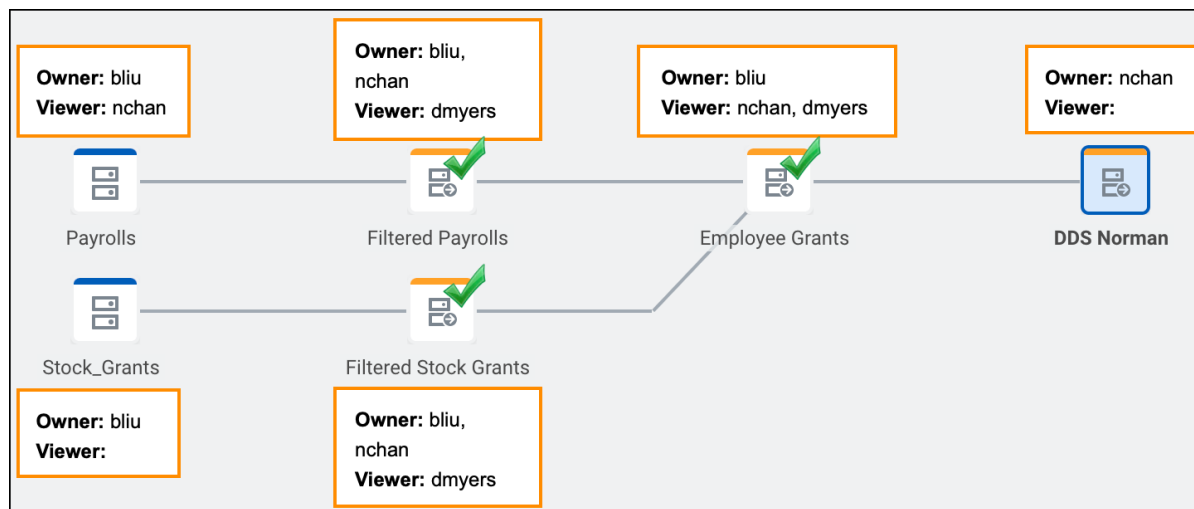
Although Norman created DDS Norman (he's the owner), he isn't able to share it with other users because he only has Viewer permission on Employee Grants, not Owner permission. To share a dataset, you must have Owner permission on all upstream objects up until you reach a dataset or table with Relax Sharing Rules enabled and functional. On the dataset or table with Relax Sharing Rules enabled, you only need Viewer permission.

Scenario 4

This scenario is based on the previous scenario. Beth Liu enabled the Relax Sharing Rules check box on the Employee Grants dataset.

Table or Dataset	Owners	Viewers	Relax Sharing Rules Enabled	Prevent Relax Sharing on Derived Datasets Enabled
Payrolls	bliu	nchan	No	No
Filtered Payrolls	bliu, nchan	dmyers	Yes	No
Stock Grants	bliu	None	No	No
Filtered Stock Grants	bliu, nchan	dmyers	Yes	No
Employee Grants	bliu	nchan, dmyers	Yes	No
DDS Norman	nchan	None	No	No

The dataset lineage for the DDS Norman dataset (as viewed by a Prism Data Administrator who can view all objects) looks like this:



Beth Liu, Norman Chan, and Dawn Myers have this table and dataset access:

Table or Dataset	Beth Liu	Norman Chan	Dawn Myers
Payrolls	Can share, edit, and view	Can view	No access
Filtered Payrolls	Can share, edit, and view	Can share, edit, and view	Can view

Table or Dataset	Beth Liu	Norman Chan	Dawn Myers
Stock Grants	Can share, edit, and view	No access	No access
Filtered Stock Grants	Can share, edit, and view	Can share and view	Can view
Employee Grants	Can share, edit, and view	Can view	Can view
DDS Norman	No access	Can share , edit, view	No access

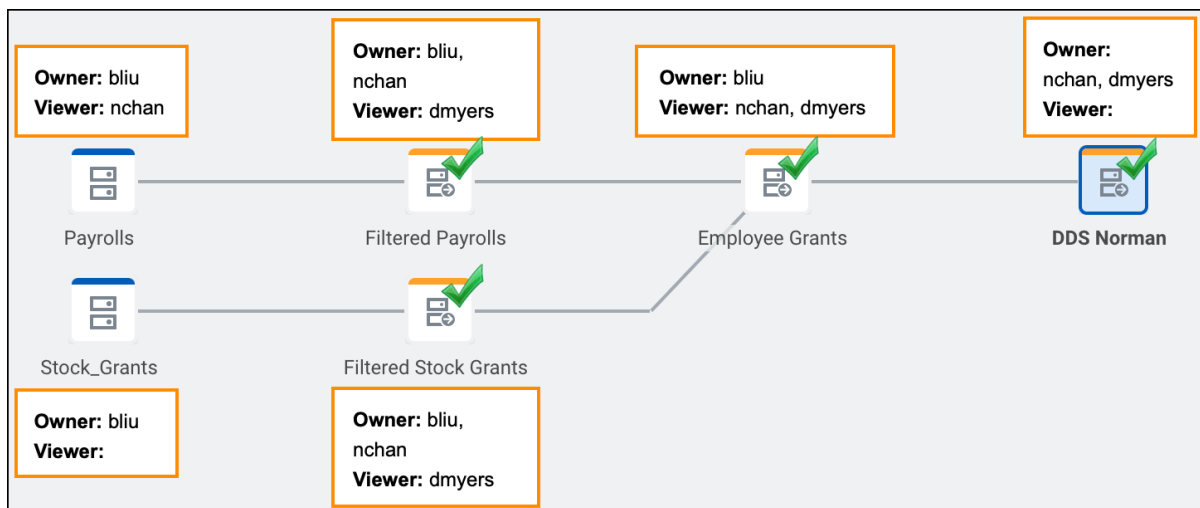
With Relax Sharing Rules enabled on Employee Grants (Norman has Viewer permission on Employee Grants), Norman can share DDS Norman with others.

Scenario 5

This scenario is based on the previous scenario. Norman Chan enabled the Relax Sharing Rules check box on DDS Norman, and shared DDS Norman with Dawn Myers, giving her Owner permission.

Table or Dataset	Owners	Viewers	Relax Sharing Rules Enabled	Prevent Relax Sharing on Derived Datasets Enabled
Payrolls	bliu	nchan	No	No
Filtered Payrolls	bliu, nchan	dmyers	Yes	No
Stock Grants	bliu	None	No	No
Filtered Stock Grants	bliu, nchan	dmyers	Yes	No
Employee Grants	bliu	nchan, dmyers	Yes	No
DDS Norman	nchan, dmyers	None	Yes	No

The dataset lineage for the DDS Norman dataset (as viewed by a Prism Data Administrator who can view all objects) looks like this:



Beth Liu, Norman Chan, and Dawn Myers have this table and dataset access:

Table or Dataset	Beth Liu	Norman Chan	Dawn Myers
Payrolls	Can share, edit, and view	Can view	No access
Filtered Payrolls	Can share, edit, and view	Can share, edit, and view	Can view
Stock Grants	Can share, edit, and view	No access	No access
Filtered Stock Grants	Can share, edit, and view	Can share and view	Can view
Employee Grants	Can share, edit, and view	Can view	Can view
DDS Norman	No access	Can share, edit, view	Can share, edit, view

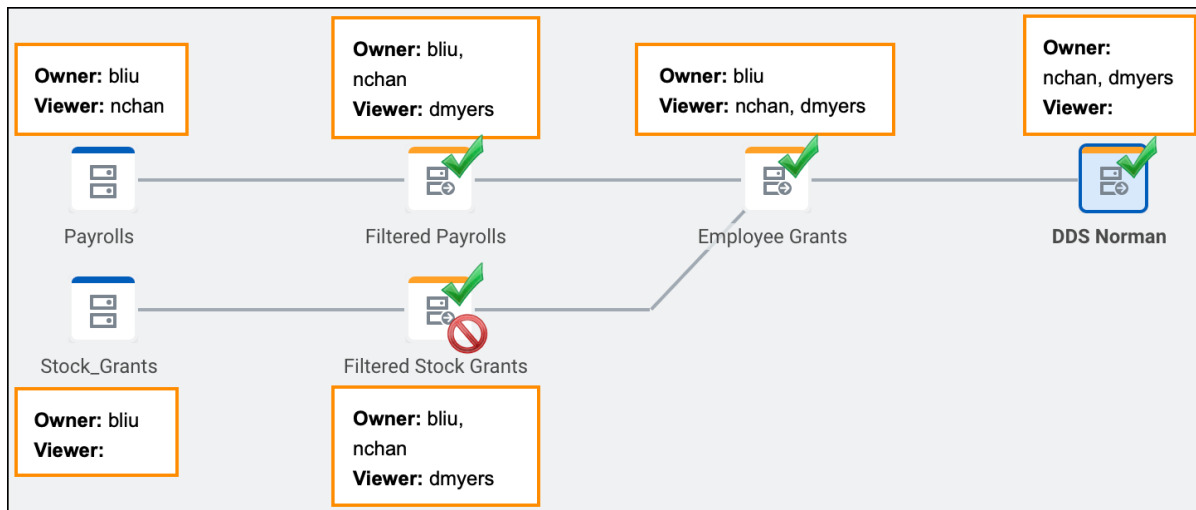
With Relax Sharing Rules enabled on DDS Norman while giving Dawn Myers Owner permission, Dawn can share, edit, and view DDS Norman. Beth Liu is unaware that DDS Norman exists and that Dawn can share it with others.

Scenario 6

This scenario is based on the previous scenario. Beth Liu enabled the Prevent Relax Sharing on Derived Datasets check box on Filtered Stock Grants.

Table or Dataset	Owners	Viewers	Relax Sharing Rules Enabled	Prevent Relax Sharing on Derived Datasets Enabled
Payrolls	bliu	nchan	No	No
Filtered Payrolls	bliu, nchan	dmyers	Yes	No
Stock Grants	bliu	None	No	No
Filtered Stock Grants	bliu, nchan	dmyers	Yes	Yes
Employee Grants	bliu	nchan, dmyers	Yes	No
DDS Norman	nchan, dmyers	None	Yes	No

The dataset lineage for the DDS Norman dataset (as viewed by a Prism Data Administrator who can view all objects) looks like this. The red circle with the line through it indicates the dataset that has Prevent Relax Sharing on Derived Datasets enabled:



Beth Liu, Norman Chan, and Dawn Myers have this table and dataset access:

Table or Dataset	Beth Liu	Norman Chan	Dawn Myers
Payrolls	Can share, edit, and view	Can view	No access
Filtered Payrolls	Can share, edit, and view	Can share, edit, and view	Can view
Stock Grants	Can share, edit, and view	No access	No access
Filtered Stock Grants	Can share, edit, and view	Can share and view	Can view
Employee Grants	Can share, edit, and view	Can view	Can view
DDS Norman	No access	Can edit, view	Can edit, view

Now that Prevent Relax Sharing on Derived Datasets is enabled on Filtered Stock Grants:

- Norman Chan can't share DDS Norman.
- Dawn Myers can't share DDS Norman.

Because Prevent Relax Sharing on Derived Datasets is enabled on Filtered Stock Grants, it cancels out the effect of Relax Sharing Rules being enabled on both Employee Grants and DDS Norman.

Related Information

Tasks

[Share a Table with Others](#) on page 104

[Share a Dataset with Others](#) on page 105

Preparing Data for Analysis

Enable Contextual Publishing for Datasets

Prerequisites

Security: These domains in the System functional area:

- *Security Activation*
- *Security Configuration*

Context

When you create and edit datasets, you typically need to build reports and visualizations to test the data you're transforming in the dataset. To test your work, you must publish your dataset, which requires access to the *Prism Datasets: Publish* domain.

You can set up your tenant to restrict users to publish datasets based on their contextual access to the dataset. Contextual publishing enables dataset users to publish a dataset based on their dataset permission, such as Dataset Owner or Dataset Editor.

You might want to enable contextual publishing to enable dataset users to test their work without giving them unconstrained access to publish all datasets.

To enable contextual publishing, you must have already enabled dataset sharing by creating assignable roles that correspond to dataset-related Workday roles.

Steps

1. Create a role-based (constrained) security group.

Select 1 of these roles for the **Assignable Role**:

- Dataset Owner
- Dataset Editor

Select **Role has access to the positions they support** for the **Access Rights to Multiple Job Workers** option.

You can create a role-based (constrained) security group to enable users to publish datasets they have Dataset Owner permission on. Example:

Security Group Property	Example
Name	Dataset Owner (role-based)
Assignable Role	Dataset Owner
Access Rights to Organizations	Applies To Current Organization And Unassigned Subordinates
Access Rights to Multiple Job Workers	Role has access to the positions they support

2. Edit the domain security policy for the *Prism Datasets: Publish* domain in the Prism Analytics functional area.
Add the role-based (constrained) security group you created, and assign both View and Modify task permissions.
3. Activate pending security policy changes.

Publish a Dataset as a Prism Data Source Manually

Prerequisites

- Security: *Prism Datasets: Publish* domain in the Prism Analytics functional area.

Context

This section describes how to publish a dataset immediately on an ad hoc basis.

Steps

1. Access the **View Dataset Details** report for the dataset you want to publish.
2. Click **Publish**, or from the related actions menu, select **Publishing > Publish Dataset**.

Workday:

- Reads the source data in the dataset.
 - Transforms the source data using the transformation logic defined in the dataset.
 - Creates a Prism data source and loads it with the transformed data. The Prism data source has the same name as the dataset API name.
 - Applies the appropriate security restrictions to the data.
3. View the publishing process status from the **Publishing Activities** tab on the **View Dataset Details** report.
Refresh the browser to see the most current status. The status includes the date and time that Workday last published the dataset successfully. The last successful published date informs you about the freshness of the data in the Prism data source. Example: If the last successful publish date is 1 week ago, but your publishing schedule is set to publish daily, this discrepancy could indicate a failure in the publishing process.
You can also view the status of manually run publish requests in the **Prism Activities Dashboard** report or the **Process Monitor** report.
 4. (Optional) You can cancel the publishing process by clicking **Cancel Publishing** on the **View Dataset Details** report.

Result

You can view the Prism data source by accessing the **View Prism Data Source** report, and selecting the name of the dataset you published.

Related Information

Tasks

[Unpublish a Dataset](#) on page 134

Reference

[The Next Level: Prism Analytics Data Acquisition Best Practices](#)

[The Next Level: Prism Performance and Troubleshooting Tips](#)

Create Dataset Publish Schedules

Prerequisites

Security: *Prism Datasets: Publish* domain in the Prism Analytics functional area.

Context

You can create a publish schedule for a dataset. You can schedule the publish to run:

- On a recurring basis (Example: daily, weekly, or monthly).
- Only if another Prism scheduled process completes at a status you specify.

You can't publish the same dataset at times that overlap with each other.

Steps

1. Access the **View Dataset Details** report for the dataset you want to publish.
2. From the related actions menu, select **Publishing > Create Schedule**.
3. In **Run Frequency**, specify how often to publish the dataset. The choices include creating a dependent publish schedule.
4. Select the criteria for the schedule.
5. (Recurring schedules) As you configure the schedule, consider:

Option	Description
Priority	Unavailable for publish schedules.
Catch Up Behavior	Select how many times the scheduled publish runs after maintenance issues cause errors. Example: If you schedule a publish to run multiple times in a week when your environment is down for maintenance, you can limit the process to run once instead of catching up all missed occurrences.

6. (Dependent schedules) As you configure the schedule, consider:

Option	Description
Dependency	Select a Prism-related schedule on which the publish schedule depends.
Trigger on Status	Select the status of the scheduled future process that triggers publishing the dataset. Workday recommends using one of the completed statuses. Example: You select Publish Dataset as your process type and a future publish schedule called All Currencies . In the Trigger on Status field, you select Completed . Workday publishes the dataset only after the scheduled All Currencies publish successfully completes.
Time Delayed Configuration	(Optional) Specify the number of days, hours, or minutes to delay publishing the dataset after the trigger. You might want to delay publishing to review the latest source files.

7. (Optional) Change the name of your publish schedule.

Workday assigns a name to the schedule based on the name of the dataset and prepends *Publish Schedule:* to the name. You can change the name of the schedule in the **Request Name** field when

you edit the schedule. Workday displays this name in the **Process Monitor** and **Scheduled Future Processes** reports to help you identify a specific process request.

Result

Workday publishes a dataset based on the criteria that you specify. When publishing, Workday:

- Reads the source data in the dataset.
- Transforms the source data using the transformation logic defined in the dataset.
- Creates a Prism data source if it doesn't already exist and loads it with the transformed data. The Prism data source has the same name as the dataset API name.
- Applies the appropriate security restrictions to the data.

View the status of all scheduled publishing processes on:

- The **Dataset Activities** tab of the **Data Catalog** report.
- The **Prism Activities Dashboard** report from the **Data Catalog** report.
- The **Prism Activities Monitor** report.

Refresh the browser to see the most current status. The status includes the date and time (UTC) that Workday last published the dataset successfully. The last successful publish date informs you about the freshness of the data in the Prism data source. Example: If the last successful publish date is 1 week ago, but you set your publish schedule to publish daily, this discrepancy could indicate a failure in the publishing process.

Related Information

Concepts

[Concept: Security in Prism Analytics](#) on page 108

Reference

[FAQ: Dataset Publish Schedules](#) on page 129

[The Next Level: Prism Analytics Best Practices](#)

Examples

[Example: Create Dependent Publish Schedules for Datasets](#) on page 128

Concept: Making Prism Data Available for Analysis

You can make Prism data in the Data Catalog available for analysis by creating a Prism data source from either a table or dataset. When Workday creates a Prism data source, it:

- Loads the data source with the data from the table or dataset.
- Applies the appropriate security restrictions to the data source, fields, records, and field values.

The way you create a Prism data source depends on the Data Catalog object:

Prism Object	Method	Security Requirements
Table	Use the Enable for Analysis option when you create or edit the table schema. See Edit a Table on page 65.	Any of these security requirements: • <i>Prism Datasets: Owner Manage</i> domain • <i>Prism Datasets: Manage</i> domain • <i>Table Editor</i> permission on the table • <i>Table Owner</i> permission on the table • <i>Table Schema Editor</i> permission on the table
Dataset	Publish the dataset. You can create a publish schedule or publish a dataset	<i>Prism Datasets: Publish</i> domain

Prism Object	Method	Security Requirements
	manually on an ad hoc basis. See Create Dataset Publish Schedules on page 123 and Publish a Dataset as a Prism Data Source Manually on page 123.	

Workday applies the security domains configured in the **Edit Data Source Security** task for the table or dataset.

When you first create a table or dataset, no security domain is applied to the data source. However, you can still create a Prism data source if you haven't specify a security domain for it. Workday applies the *Prism: Default to Dataset Access* security domain if no domain has been configured. The *Prism: Default to Dataset Access* domain provides contextual access to a Prism data source based on your access to the underlying table or dataset.

Workday recommends making Prism data available analysis after you edit the data source security for the table or dataset.

- Related Information**
- Concepts**
- [Concept: Prism Data Sources](#) on page 11
- Tasks**
- [Unpublish a Dataset](#) on page 134
- Reference**
- [The Next Level: Prism Analytics Best Practices](#)
 - [The Next Level: Prism Performance and Troubleshooting Tips](#)

Concept: Dataset Publish Schedules

Schedules for publishing datasets enable you to specify when, how often, and under what criteria to publish a dataset. Publish schedules differ from publishing immediately on an ad hoc basis using **Run Now**.

Publish Schedule Types

You can create these types of schedules:

Schedule Type	Description
Recurring	A publish schedule that runs at specified intervals, such as daily, weekly, or monthly.
Dependent	A publish schedule that depends on the completion of another Prism scheduled process. For your dependency criteria, you can specify: - The process type, such as bringing data into a dataset. - The status that triggers publishing, such as the process type successfully completing. Example: When the <i>Prism Data Acquisition Future Process</i> completes with no warnings or errors, begin publishing.

Schedule Type	Description
	Note: A Prism Analytics publish schedule can depend only on another Prism-related process, such as bringing data into a dataset or publishing another dataset.

After you create a publish schedule, consider these actions that you can perform on it:

Action	Description
Activate	Activate a suspended publish schedule.
Change Schedule (recurring schedules only)	Edit the run frequency (daily, monthly, weekly), start time, and date range for the publish schedule. You can also change to another scheduled recurring process.
Delete	Permanently delete the publish schedule.
Edit Environment Restrictions	Select the environment in which you want the scheduled publish to run.
Edit Schedule	(Recurring Schedules) Edit the schedule name, recurrence criteria, and range of recurrence dates. Note: To change the run frequency, use Change Schedule . (Dependent Schedules) Edit the schedule name, dependency, trigger status, and timed delay configurations.
Edit Scheduled Occurrence (recurring schedules only)	Update the schedule date and time for one particular occurrence of the scheduled request. You can also delete a particular occurrence of the scheduled publish.
Expire Schedule	Permanently stop the publish schedule. Expiring a schedule doesn't delete it. Note: You can't activate an expired schedule.
Run Now	Publish the dataset immediately on an ad hoc basis.
Suspend	Suspend use of the publish schedule. You can activate a suspended schedule.
Transfer Ownership	Transfer ownership of a publish schedule. Every process must have an assigned owner for the process to run. You might want to transfer ownership if the assigned owner becomes inactive. The person you transfer ownership to must have the appropriate security access.
View All Occurrences (recurring schedules only)	View all future occurrences of a publish schedule within a specified range of dates and times.

Action	Description
View Schedule	(Recurring schedules) View schedule details, such as recurrence criteria, error messages, the schedule owner and creator, and the next 10 scheduled launches if applicable. (Dependent schedules) View schedule details, such as the dependency configuration, the schedule creator and owner, and the number of times run.

Related Information**Tasks**

[Create Dataset Publish Schedules](#) on page 123

Reference

[FAQ: Dataset Publish Schedules](#) on page 129

[The Next Level: Prism Analytics Best Practices](#)

[The Next Level: Prism Performance and Troubleshooting Tips](#)

Concept: Coordinating Publish Schedules with Dataset Integration

The last integration that completes before publishing begins determines the freshness of data in a Prism data source. To ensure that data is as fresh as possible, schedule enough time for an integration to complete before a dataset publishing job begins.

As you create publish schedules for datasets that have scheduled integrations, consider how these scenarios affect the freshness of data in the Prism data source:

Scenario	Impact
Integration completes before a scheduled publish begins.	The published dataset will use the latest available data from the dataset source file.
Integration completes after a scheduled publish begins.	The published dataset will use the data from the dataset source file that existed before integration began.

To help ensure that integration completes before publishing begins, you can create a publish schedule that depends on the successful integration of the data.

Related Information**Tasks**

[Create Dataset Publish Schedules](#) on page 123

Example: Create Dependent Publish Schedules for Datasets

This is an example of a publish schedule with a dependency.

Context

As an HR Analyst, you're responsible for creating a weekly report that includes employee hiring and turnover data from your New York office. You'd like the data in the report to be as fresh as possible. You need to:

- Bring in the latest hiring and turnover data from your New York office every Monday morning.

- Make the data available for reporting only after you've brought in the latest data.

Prerequisites

- Create a dataset from SFTP integration. Schedule the integration to occur every Monday at 8 a.m. EST. Name the dataset *New York Hiring and Turnover Weekly*. Run the integration once to enable the publish option.
- Security: *Prism Datasets: Publish* domain in the Prism Analytics functional area.

Steps

1. Access the **View Dataset Details** report for the *New York Hiring and Turnover Weekly* dataset.
2. From the related actions menu of the **View Dataset Details** report, select **Publishing > Create Schedule**.
3. In **Run Frequency**, select **Dependent**.
4. In **Dependency**, select **Prism > Dataset Integration Schedule: New York Hiring and Turnover Weekly**.
5. In **Trigger on Status**, select **Completed**.

Result

Workday publishes the *New York Hiring and Turnover Weekly* dataset only after the successful integration of data from the SFTP server.

The HR Analyst can:

- View the publish schedule request in the **Process Monitor** and **Scheduled Future Processes** reports.
- View the status of all scheduled publish processes on these reports:
 - **Dataset Activities**
 - **Prism Activities Monitor**
 - **Prism Activities Dashboard**

Related Information

Concepts

[Concept: Making Prism Data Available for Analysis](#) on page 125

[Concept: Dataset Publish Schedules](#) on page 126

Tasks

[Create Dataset Publish Schedules](#) on page 123

[Steps: Create a Dataset with External Data \(SFTP Server\)](#) on page 41

FAQ: Dataset Publish Schedules

- [How do I edit a publish schedule?](#) on page 130
- [How do I edit or delete only 1 occurrence of a recurring publish schedule?](#) on page 130
- [How do I change a recurring publish schedule?](#) on page 130
- [How do I edit environment restrictions for a publish schedule?](#) on page 130
- [How do I delete a publish schedule?](#) on page 130
- [How do I expire a publish schedule?](#) on page 130
- [How do I suspend a publish schedule?](#) on page 130
- [How do I activate a suspended publish schedule?](#) on page 130
- [How do I view the details of a publish schedule?](#) on page 131
- [How do I view all occurrences of a recurring publish schedule?](#) on page 131
- [How do I immediately run a publish schedule on an ad hoc basis?](#) on page 131

- [How do I transfer ownership of a publish schedule?](#) on page 131

How do I edit a publish schedule?

From the related actions menu of the **View Dataset Details** report, select **Publishing > Edit Schedule**.

How do I edit or delete only 1 occurrence of a recurring publish schedule?

1. From the related actions menu of the **View Dataset Details** report, select **Publishing > View Schedule**.
2. From the related actions menu of the **View Scheduled Future Process** report, select **Schedule Future Process > Edit Scheduled Occurrence**.

How do I change a recurring publish schedule?

1. From the related actions menu of the **View Dataset Details** report, select **Publishing > View Schedule**.
2. From the related actions menu of the **View Scheduled Future Process** report, select **Schedule Future Process > Change Schedule**.

How do I edit environment restrictions for a publish schedule?

1. From the related actions menu of the **View Dataset Details** report, select **Publishing > View Schedule**.
2. From the related actions menu of the **View Scheduled Future Process** report, select **Schedule Future Process > Edit Environment Restrictions**.

How do I delete a publish schedule?

1. From the related actions menu of the **View Dataset Details** report, select **Publishing > View Schedule**.
2. From the related actions menu of the **View Scheduled Future Process** report, select **Schedule Future Process > Delete**.

How do I expire a publish schedule?

From the related actions menu of the **View Dataset Details** report, select **Publishing > Expire Schedule**.

How do I suspend a publish schedule?

1. From the related actions menu of the **View Dataset Details** report, select **Publishing > View Schedule**.
2. From the related actions menu of the **View Scheduled Future Process** report, select **Schedule Future Process > Suspend**.

How do I activate a suspended publish schedule?

1. From the related actions menu of the **View Dataset Details** report, select **Publishing > View Schedule**.
2. From the related actions menu of the **View Scheduled Future Process** report, select **Schedule Future Process > Activate**.

Note: You can only activate expired schedules.

How do I view the details of a publish schedule?	From the related actions menu of the View Dataset Details report, select Publishing > View Schedule .
How do I view all occurrences of a recurring publish schedule?	1. From the related actions menu of the View Dataset Details report, select Publishing > View Schedule. 2. From the related actions menu of the View Scheduled Future Process report, select Schedule Future Process > View All Occurrences.
How do I immediately run a publish schedule on an ad hoc basis?	1. From the related actions menu of the View Dataset Details report, select Publishing > View Schedule. 2. From the related actions menu of the View Scheduled Future Process report, select Schedule Future Process > Run Now.
How do I transfer ownership of a publish schedule?	1. From the related actions menu of the View Dataset Details report, select Publishing > View Schedule. 2. From the related actions menu of the View Scheduled Future Process report, select Schedule Future Process > Transfer Ownership.

Related Information Concepts

[Concept: Making Prism Data Available for Analysis](#) on page 125

[Concept: Dataset Publish Schedules](#) on page 126

Tasks

[Create Dataset Publish Schedules](#) on page 123

Deleting Prism Analytics Data

Truncate Data in a Table

Prerequisites

Any of these security requirements:

- *Prism: Tables Manage* domain in the Prism Analytics functional area.
- *Prism: Tables Owner Manage* domain in the Prism Analytics functional area.
- *Table Editor* permission on the table.
- *Table Owner* permission on the table.
- *Can Truncate Table Data* permission on the table.

Context

You can remove all data in a table by truncating the table. Truncating a table removes the data, but retains the schema. You might want to truncate a table if the table contains some bad data.

If you've previously published a derived dataset based on this table, the associated Prism data source still contains data. If you also want to remove the data from the associated Prism data source, then you must publish the derived dataset again. Publishing a derived dataset from a truncated table makes the associated Prism data source empty, but active.

If you selected the **Enable for Analysis** option for this table, then truncating the table also removes all data from the associated Prism data source. The Prism data source is empty, but active.

Steps

1. Access the **View Table Details** report for the table you want to truncate.
2. Select **Truncate Data** from the **Quick Actions** menu.

Delete Rows of Data in a Table

Prerequisites

Security:

- *Prism: Manage File Containers* domain in the Prism Analytics functional area when uploading a file.
- Any of these security requirements:
 - *Prism: Tables Owner Manage* domain in the Prism Analytics functional area.
 - *Prism: Tables Manage* domain in the Prism Analytics functional area.
 - *Table Owner* permission on the table.
 - *Table Editor* permission on the table.
 - *Can Delete Table Data* permission on the table.
 - *Can Truncate Table Data* permission on the table.

Context

You can delete a subset of rows in a table using these methods:

Row Deletion Method	Notes
By load ID	You can quickly delete all rows that came from a particular data load activity. Select Table > Delete Rows from the related actions menu of the table. Then select Previous Loads .
By data change task	You can delete specific rows that you specify using a data change task with the delete operation. You delete rows by specifying one these target fields as the Delete Key: • The field configured as the external ID • WPA_LoadID • WPA_RowID To delete rows using a data change task, you need a delimited file that contains a single field that contains the values in the delete key field. Example: You can delete the rows for particular values of the field configured as the external ID. Create a CSV file that contains

Row Deletion Method	Notes
	the values of the external ID field you want to delete. To delete some rows from the ClaimID field, your CSV file might look like: <pre>ClaimID 345999 345600 345601 345602</pre>

To delete specific rows using a data change task:

Steps

1. Access the **View Table Details** report for the table to delete from.
2. Select **Quick Actions > Data Change Task**.
3. (Optional) Change the data change task name that Workday created automatically at the top of the left side panel.
4. On the **Source** step, select the file that contains the values of the rows you want to delete.
5. On the **Source Options** step, define how to parse the data in the files.
6. On the **Target** step, select **Delete** as the **Target Operation**.
7. On the **Mapping** step, select a field in the target table to use as the **Delete Key**.
8. Select a source field for the target field that you specified as the delete key.

Related Information

Concepts

[Concept: Data Change Tasks](#) on page 96

Tasks

[Create a Data Change Task](#) on page 91

Truncate Data in a Dataset

Prerequisites

Any of these security requirements:

- *Prism Datasets: Manage* domain in the Prism Analytics functional area.
- *Dataset Editor* permission on the dataset.
- *Dataset Owner* permission on the dataset.

Context

You can remove the data in a base dataset by truncating the dataset. Truncating a dataset removes the data from the dataset, but retains its metadata, such as schema and transformations.

You might want to truncate a dataset if:

- The dataset integration uses Append mode and after several integrations the dataset contains some bad data.
- The dataset contains data that your organization later deems to be sensitive.

If you've previously published this dataset or a derived dataset based on this dataset, the associated Prism data source still contains data. If you also want to remove the data from the associated Prism data source,

then you must publish the dataset again. Publishing a truncated dataset makes the associated Prism data source empty, but active.

Steps

1. Access the **View Dataset Details** report for the dataset you want to truncate.
2. Select **Actions > Dataset > Truncate**.

Related Information

Concepts

[Concept: Deleting Prism Data](#) on page 135

Unpublish a Dataset

Prerequisites

- Prism data source exists in Workday, but no reports use it.
- Security: *Prism Datasets: Publish* domain in the Prism Analytics functional area.

Context

After you publish a dataset, you can unpublish it if necessary. When you unpublish a dataset, Workday removes the Prism data source that is based on the dataset, including all data in it. You might want to unpublish a dataset if you need to delete the dataset. You can only unpublish a dataset if no reports use the associated Prism data source.

Steps

1. Access the **View Dataset Details** report for the dataset you want to unpublish.
2. Select **Actions > Publishing > Unpublish Dataset**.

Result

Workday removes the Prism data source and its rows.

Related Information

Concepts

[Concept: Deleting Prism Data](#) on page 135

Delete Rows from a Prism Data Source

Prerequisites

Security: *Prism Datasets: Publish* domain in the Prism Analytics functional area.

Context

When you publish a dataset, Workday creates a Prism data source and populates it with rows containing the transformed data from the dataset. You can delete the rows in a Prism data source created from a published dataset. When you delete rows in a Prism data source, the Prism data source is empty and becomes inactive. If you need to delete rows from a Prism data source created from a table enabled for analysis, then truncate the table.

You might want to delete the rows in a Prism data source if the rows contain incorrect data and you don't want analysts creating reports using the bad data. You can then edit the dataset to correct the

transformation logic and publish the dataset again. When you republish the dataset, Workday populates the empty, inactive Prism data source with the new data.

Any reports that use an inactive Prism data source will be broken until you publish the dataset again.

Steps

1. Access the **View Prism Data Source** page, and select the Prism data source whose rows you want to delete.
2. Select **Actions > Prism Data Source > Delete Published Rows**.

Result

Workday removes all data from the Prism data source, and changes the Prism data source status to inactive.

Concept: Deleting Prism Data

You create tables, datasets, and Prism data sources filled with data. However, there might be times when you need to remove data from your tenant.

You can remove data and Prism objects:

Action	Notes
Make the Prism data source inactive.	When a Prism data source is inactive, it exists in the tenant, but is empty and unavailable for querying in reports and discovery board visualizations. You might want to make a Prism data source inactive if the data source contains incorrect data and you don't want analysts creating reports using the bad data. You can make a Prism data source inactive whether or not any reports or visualizations use it. How you make a Prism data source inactive depends on the object type it's based on: • Table. Edit the table schema and clear the Enable for Analysis option on the Edit Table task. • Dataset. Delete the rows in the Prism data source. Select Prism Data Source > Delete Published Rows from the related actions on the View Prism Data Source report.
Delete the Prism data source including all data in it.	You can only remove a Prism data source when no reports or vizzes currently use the associated Prism data source. How you remove a Prism data source depends on the object type it's based on: • Table. Delete the table from the Data Catalog. Right-click the table from the Data Catalog and select Delete. • Dataset. Unpublish the dataset. Select Publishing > Unpublish Dataset from the related actions of the dataset. You might want to unpublish a dataset to delete it from the Data Catalog.
Remove all data from a dataset (truncate).	You can remove the data in a base dataset by truncating the dataset. Truncating a dataset removes all data from the dataset. However, the dataset retains its metadata, such as schema and transformations.

Action	Notes
	You might want to truncate a dataset if the dataset integration uses Append mode and after several integrations the dataset contains some bad data. If you've previously published this dataset or a derived dataset based on this dataset, then the associated Prism data source still contains data. If you also want to remove the data from the associated Prism data source, then you must publish the dataset again. Publishing a truncated dataset makes the associated Prism data source empty, but active.
Delete a dataset.	When you delete a dataset, Workday removes the dataset definition from the Data Catalog and removes the source data stored on disk in your tenant. You might want to delete a dataset to make more space available to store data in your Data Catalog. You can only delete a dataset if it's not currently published and isn't imported into any derived dataset. Right-click a dataset from the Data Catalog and select Delete.
Delete rows from a table.	You can delete 1, multiple, or all rows from a table. When you delete rows from a table, the table remains in the Data Catalog.
Truncate a table.	When you truncate a table, Workday removes all rows from the table, and keeps the empty table in the Data Catalog. If you selected the Enable for Analysis option for this table, then truncating the table also removes all data from the associated Prism data source. The Prism data source is empty, but active.
Delete a table.	When you delete a table, Workday removes the table definition from the Data Catalog, including all data contained in it. You can only delete a table when: • No reports or vizzes currently use the associated Prism data source. • No data change task is configured for the table. Right-click the table from the Data Catalog and select Delete.

Related Information

Tasks

[Truncate Data in a Dataset](#) on page 133

[Unpublish a Dataset](#) on page 134

[Delete Rows from a Prism Data Source](#) on page 134

Managing Analytic Data Sources

Install and Schedule an Analytic Data Source

Prerequisites

Security: *Manage: Analytic Data Sources* domain in the Analytical Framework functional area.

Context

Install an analytic data source and define a schedule for updating the data in it.

Steps

1. Access the **Tenant Setup for Analytic Data Source** task.
2. Select the **Analytic Data Source** to install.
3. In **Run Frequency**, specify how often to update the data in the analytic data source.
4. Enter a name for the analytic data source schedule.
5. Select the **Schedule** tab, and select the criteria for the schedule.

When the analytic data source is based on data in Workday business objects that contain a very large number of instances, we recommend that you define the schedule to run:

- During a quiet time.
- Weekly or monthly.

Result

Workday creates the analytic data source and its schedule. Workday loads data from Workday business objects into the data source at the next scheduled time.

Next Steps

You can:

- Use the analytic data source in a discovery board or custom report.
- View the data source details using the **View Analytic Data Source (Workday Owned)** report.
- Edit, cancel, or view the analytic data source schedule.

Related Information

Concepts

[Concept: Workday-Delivered Analytic Data Sources](#) on page 138

[Concept: Managing Analytic Data Source Schedules](#) on page 138

Uninstall an Analytic Data Source

Prerequisites

Security: *Manage: Analytic Data Sources* domain in the Analytical Framework functional area.

Context

You can uninstall an analytic data source to remove all data from that data source. You might want to do uninstall the data source if you encounter a critical error while using it.

When you uninstall an analytic data source:

- Any reports that use the data source will display no results and display an error the next time you run the report.
- You can re-install the data source before running any existing reports that use the data source.

Steps

1. Access the **Uninstall/Cleanup Analytic Data Source** task.
2. Select the data source to uninstall, and click **Confirm**.

Concept: Workday-Delivered Analytic Data Sources

An analytic data source is a Workday-blended and delivered data source updated on a regular schedule that can contain Workday data or external data.

Analytic data sources use Prism Analytics and are designed for high performance and analytical reporting. Although analytic data sources can contain Workday data, the data isn't updated in real time like Workday-delivered data sources directly based on business objects. When you install the analytic data source, you define how often the data updates, such as daily, weekly, or monthly.

When Workday updates the data in an analytic data source, Workday:

- Runs a background job that extracts data from the Workday or external sources.
- Blends and transforms the data using Prism Analytics.
- Loads the data into the data source.

This processing takes time, depending on how much data is in your tenant and how often you schedule the data updates. As a result, the freshness of the data in your analytic data source depends on both the schedule frequency and the processing time required to update the data.

Related Information

Tasks

[Install and Schedule an Analytic Data Source](#) on page 136

Concept: Managing Analytic Data Source Schedules

After you install an analytic data source, you can manage the schedule you created for the data source. You need access to the *Manage Analytic Data Sources* domain in the Analytical Framework functional area to manage the schedule.

You can use these tasks and reports to manage the analytic data source schedule:

Task or Report Name	Notes
Cancel Analytic Data Source Publish Schedule	You can only cancel a schedule that currently exists.
Create Analytic Data Source Publish Schedule	You can only create a schedule if you previously canceled the schedule.
Edit Analytic Data Source Publish Schedule	Change the schedule settings on the Schedule tab.
View Analytic Data Source Publish Schedule	

Concept: Peakon Employee Voice Analytic Data Source

The Peakon Employee Voice analytic data source enables you to create custom employee engagement reporting. It extracts non-aggregated answers from the Peakon system, enabling you to analyze your Peakon data in more detail.

This analytic data source includes scores and comments from all question sets that you measure on your Peakon Employee Voice survey, including these:

- Engagement
- Diversity and Inclusion
- Health and Wellbeing
- COVID-19
- All company-specific questions

Consider the sensitive nature of the Peakon scores data as you interact with this analytic data source and its underlying dataset. All scores and comments are attached to an employee ID, making it possible to identify the individual scores and comments left by an employee on their engagement survey. This analytic data source is designed for companies with the appropriate safeguards.

Implementing a fully transparent survey requires a lot of trust from your employees. Ensure this trust by securing their data through your analysis in Workday. You secure this data by:

- Configuring the security policy for the *Peakon Responses Analytic Data Source* domain.
- Providing users access to the Peakon Employee Voice Raw Data dataset in the Data Catalog. All users with unconstrained access to the Data Catalog can access the data in the Peakon dataset and can share it with Prism users who have constrained access.

Workday recommends that you only provide access to this data to highly privileged users before sharing anonymized data with a wider audience.

Note: To install this analytic data source:

- Enable the Peakon Data Export API feature.
- Map the Workday tenant name in Peakon.

Peakon Employee Voice Raw Data Dataset

When you install this analytic data source, Workday also creates and adds a Prism Analytics dataset in the Data Catalog containing Peakon answers data. The data in the Peakon Employee Voice Raw Data dataset is the same as the data in the analytic data source. You can use Prism Analytics to join this dataset with other data, such as worker or external data, further enhancing your reporting and analytics capabilities in Discovery Boards and Report Writer.

Note: You can migrate between tenants any derived datasets, discovery boards, custom reports, or Prism data sources that you create based on the Peakon Employee Voice Raw Data dataset. However, before you migrate these assets to a different tenant, you must install the Peakon Employee Voice analytic data source in the target tenant.

Security for Peakon Employee Voice Data

Workday controls access to the analytic data source and the raw data dataset using the *Peakon Responses Analytic Data Source* domain.

To provide access to the data source and dataset:

1. Create or edit a user-based security group that contains the users who can use the data source and raw data dataset.
2. Edit the domain security policy for the *Peakon Responses Analytic Data Source* domain, and provide the security group both View and Modify permissions.
3. [Set Up Dataset Sharing](#) on page 102. Add the user-based security group in the **Role Assignees Restricted To** column for the **Prism Dataset Viewer Workday Role**.
4. Activate pending security policy changes.

Note: Consider the sensitive nature of the highly confidential Peakon scores data as you configure this domain security policy. Workday recommends that you only provide access to this data to highly privileged users before sharing anonymized data with a wider audience.

Related Information

Concepts

[Concept: Workday-Delivered Analytic Data Sources](#) on page 138

Tasks

[Install and Schedule an Analytic Data Source](#) on page 136

Reference: Prism Expression Language

Concept: Prism Expression Language

Prism Analytics includes its own expression language that's composed of functions and operators. You use the Prism expression language to create an *expression*, which you can use to create new values or filter existing values.

An expression computes or produces a value by combining fields, constant values, operators, and functions. An expression outputs a value of a particular field type, such as Numeric or Text values. Simple expressions can be a single constant value, the values of a given field, or just a function. You can use operators to join two or more simple expressions into a complex expression.

You can use expressions in datasets:

- **Calculated fields.** Use expressions to define calculated fields that operate on the source data. A calculated field expression generates its values based on a calculation or condition, and returns a value for each input row. Calculated field expressions can contain values from other fields, constants, mathematical operators, comparison operators, or built-in row functions.
- **Filter stages.** Use an expression in a Filter stage to limit the scope of the source data of a dataset.

The expression builder helps you create calculated field expressions in a dataset. It displays the available fields in the dataset, plus the list of the Prism functions. It validates your expressions for correct syntax, input field types, and so on.

Function Inputs and Outputs

Functions take one or more input values and return an output value. Input values can be a literal value or the name of a field that contains a value. In both cases, the function expects the input value to be a particular field type such as Text or Integer. Example: The CONCAT() function combines Text inputs and outputs a new Text.

This example demonstrates how to use the CONCAT() function to concatenate the values in the *month*, *day*, and *year* fields separated by the literal forward slash character:

```
CONCAT([month], "/", [day], "/", [year])
```

A function return value might be the same as its input type or it might be an entirely new field type. Example: The TO_DATE() function takes a Text as input, but outputs a Date value. If a function expects a Text, but is passed another field type as input, the function returns an error.

Typically, functions are classified by what field type they take or what purpose they serve. Example: CONCAT() is a text function and TO_DATE() is a field type conversion function.

Nesting Functions

Functions can take other functions as arguments. Example: You can use the CONCAT function as an argument to the TO_DATE() function. The final result is a Date value in the format 10/31/2014.

```
TO_DATE(CONCAT([month], "/", [day], "/", [year]), "MM/dd/yyyy")
```

The nested function must return the correct field type. So, because TO_DATE() expects text input and CONCAT() returns a text, the nesting succeeds.

Referring to Fields in the Current Dataset

Workday recommends enclosing all field names with square brackets ([]). Example:

```
TO_INT([sales])
```

```
TO_INT([Sale Amount])
```

```
TO_INT([2013_data])
```

```
TO_INT([count])
```

If a field name contains a] (closing square bracket), you must escape the closing square bracket by doubling it]]. Suppose you have this field name:

```
[Total Sales]
```

You enclose the entire field name in square brackets and escape the closing bracket that is part of the actual field name:

```
[[Total Sales]]]
```

Literal Text Values

To specify a literal or actual Text value, enclose the value in double quotes ("). Example: This expression converts the values of a *gender* field to the literal values of *male*, *female*, or *unknown*:

```
CASE WHEN [gender]="M" THEN "male"
      WHEN [gender]="F" THEN "female"
      ELSE "unknown" END
```

To escape a literal quote within a literal value itself, double the literal quote character. Example:

```
CASE WHEN [height]="60'" THEN "5 feet" WHEN [height]="72'"
      THEN "6 feet" ELSE "other" END
```

The REGEX() function is a special case. In the REGEX() function, Text expressions are also enclosed in quotes. When a Text expression contains literal quotes, double the literal quote character. Example:

```
REGEX([height], "\d\\'(\d)+'"')
```

Literal Date Values

To refer to a Date value in a Filter stage expression, you must use this format (or any shortened version of it) without any enclosing quotation marks or other punctuation:

```
yyyy-MM-ddTHH:mm:ss:SSSZ
```

Example:

```
[Order Date] BETWEEN 2016-06-01T00:00:00.000Z AND 2016-07-31T00:00:00.000Z
```

If the Filter expression is a shortened version of the full format, then Prism assigns any values that aren't included a value of zero (0). Example: This expression:

```
[Order Date] >= 2017-01-01
```

Is equivalent to:

```
[Order Date] >= 2017-01-01T00:00:00.000Z
```

If the date value is in Text format rather than a Date format, you must enclose the value in quotes.

To refer to a literal date value in a calculated field expression, you must specify the format of the date and time components using TO_DATE, which takes a Text literal argument and a format string. Example:

```
CASE WHEN [Order Date]=TO_DATE("2013-01-01 00:00:59 PST", "yyyy-MM-dd  
HH:mm:ss Z")  
THEN "free shipping" ELSE "standard shipping" DONE
```

Literal Numeric Values

For literal numeric values, you can just specify the number itself without any special escaping or formatting. Example:

```
CASE WHEN [is married]=1 THEN "married" WHEN [is married]=0  
THEN "not_married" ELSE NULL END
```

Related Information

Tasks

[Add a Prism Calculated Field to a Dataset](#) on page 70

Comparison Operators

Comparison operators are used to compare the equivalency or inequivalency of 2 expressions of the same field type. The result of a comparison expression is a Boolean value (returns true, false, or NULL for invalid, such as comparing a text value to a numeric value). Boolean expressions are most often used to specify data processing conditions or filter criteria.

Example: You can use comparison operators in a CASE expression:

```
CASE WHEN [age] <= 25 THEN "0-25"  
WHEN [age] <= 50 THEN "26-50"  
ELSE "over 50" END
```

This expression compares the value in the age field to a literal number value. If true, it returns the appropriate boolean value.

Operator	Meaning	Example
= or ==	Equal to	[order_date] = "12/22/2016"
>	Greater than	[age] > 18
!>	Not greater than (equivalent to <)	[age] !> 8
<	Less than	[age] < 30
!<	Not less than (equivalent to >=)	[age] !< 12
>=	Greater than or equal to	[age] >= 20
<=	Less than or equal to	[age] <= 29
<> or != or ^=	Not equal to	[age] <> 30

Operator	Meaning	Example
BETWEEN <i>min_value</i> AND <i>max_value</i>	Test whether a date or numeric value is within the <i>min</i> and <i>max</i> values (inclusive).	[year] BETWEEN 2014 AND 2016
IN(<i>list</i>)	Test whether a value is within a set.	[product_type] IN("tablet", "phone", "laptop")
LIKE(" <i>pattern</i> ")	Simple inclusive case-insensitive character pattern matching. The * character matches any number of characters. The ? character matches exactly 1 (a single) character.	[last_name] LIKE("?utch*") Matches <i>Kutcher</i> , <i>hutch</i> but not <i>Krutcher</i> or <i>crutch</i> [company_name] LIKE("workday") Matches <i>Workday</i> or <i>workday</i>
<i>value</i> IS NULL	Check whether a field value or expression is null (empty).	[ship_date] IS NULL Evaluates to <i>true</i> when the <i>ship_date</i> field is empty

Logical Operators

Use logical operators in expressions to test for a condition. Logical operators define Boolean expressions.

You might want to use logical operators in Filter transformations or CASE expressions. Filters test if a field or value meets some condition, such as testing if the value in a Date field falls between 2 other dates:

```
BETWEEN 2016-06-01 AND 2016-07-31
```

Operator	Meaning	Example
AND	Test whether 2 conditions are true.	
OR	Test if either of 2 conditions are true.	
NOT	Reverses the value of other operators.	- year NOT BETWEEN 2013 AND 2016 - first_name NOT LIKE("Jo?n*") Excludes <i>John</i>, <i>jonny</i> but not <i>Jon</i> or <i>Joann</i> - Weekday NOT IN("Saturday", "Sunday") - purchase_date IS NOT NULL Evaluates to <i>true</i> when the <i>purchase_date</i> field is not empty

Arithmetic Operators

Arithmetic operators perform basic math operations on 2 expressions of the same field type resulting in a numeric value.

Example: You can calculate the gross profit margin percentage using the values of a [total_revenue] and [total_cost] field:

```
(([total_revenue] - [total_cost]) / [total_cost]) * 100
```

Operator	Meaning	Example
+	Addition	[amount] + 10 Add 10 to the value of the [amount] field.
-	Subtraction	[amount] - 10 Subtract 10 from the value of the [amount] field.
*	Multiplication	[amount] * 100 Multiply the value of the [amount] field by 100.
/	Division	[bytes] / 1024 Divide the value of the [bytes] field by 1024 and return the quotient.

Conversion Functions

BUILD_CURRENCY

Description

BUILD_CURRENCY is a row function that constructs a Currency field from a numeric value and a Text or Instance value that contains a valid currency code. When the currency code isn't valid, this function returns NULL.

Syntax

`BUILD_CURRENCY(number_expression,currency_code_expression)`

Return Value

Returns a value of type Currency.

Input Parameters

number_expression

Required. A field or expression of type Double, Numeric, Integer, or Long.

currency_code_expression

Required. A field or expression of type Text or Instance that contains valid currency code data.

Examples

Convert the values of the Sale Price field (Numeric type) to a Currency field type using the currency codes from the Currency Code field:

```
BUILD_CURRENCY([Sale Price], [Currency Code])
```

CAST

Description

CAST is a row function that converts data values from one field type (data type) to another.

You can use CAST to convert these field types:

From Field Type	To Field Types
Boolean	Boolean, Numeric, Double, Integer, Long, Text
Date	Date, Text
Numeric	Boolean, Numeric, Double, Integer, Long, Text
Double	Boolean, Numeric, Double, Integer, Long, Text
Integer	Boolean, Numeric, Double, Integer, Long, Text
Long	Boolean, Numeric, Double, Integer, Long, Text
Instance	Instance, Text
Multi-Instance	Multi-Instance (using a different business object)
Text	Boolean, Numeric, Double, Instance, Integer, Long, Text

Syntax

```
CAST(field_name AS field_type)
```

Return Value

Returns one value per row of the specified field type.

Input Parameters

field_name

Required. A field or expression of a supported field type.

field_type

Required. The field type to convert the data values into.

To convert a field to the Text field type, specify String for this parameter.

When specifying Boolean as the field type, CAST converts the value of zero (0) to False, and all other values to True.

When specifying Instance or Multi-Instance as the field type, you must specify the business object using its unique identifier (WID). Use this syntax:

```
Instance(business_object_WID)
```

```
Multi_Instance(business_object_WID)
```

When specifying Numeric as the field type, specify the number of digits to the left of the decimal point (integers) and the number of digits to the right of decimal point (decimals). Use this syntax:

```
Decimal(integers, decimals)
```

Ensure that the number of integer digits specified is large enough to capture all possible data values. If the value for a row has more integer digits than the number of integer digits specified in the function, then CAST returns NULL. Example: CAST(99.9 AS decimal(1,1)) returns NULL.

Examples

Convert the WID values of the *Region* field from Text to Instance:

```
CAST([Region] AS Instance(eecb565181284b6a8ae8b45dc3ed1451))
```

```
CAST("3b122818d7934d1c8c663ddbe1937819" AS  
Instance(eecb565181284b6a8ae8b45dc3ed1451))
```

Convert the *amount* field to Text:

```
CAST([amount] AS string)
```

Convert the values of the *average_rating* field to a Numeric field type:

```
CAST([average_rating] AS decimal(1,2))
```

```
CAST(99.99 AS decimal(10,3)) returns 99.990.
```

```
CAST(99.99 AS decimal(20,2)) returns 99.99.
```

```
CAST(99.999 AS decimal(10,2)) returns 100.00.
```

```
CAST(99.99 AS decimal(1,2)) returns NULL.
```

EPOCH_MS_TO_DATE

Description

EPOCH_MS_TO_DATE is a row function that converts Long values to Date values, where the input number represents the number of milliseconds since the epoch.

Syntax

```
EPOCH_MS_TO_DATE(long_expression)
```

Return Value

Returns one value per row of type Date in UTC format *yyyy-MM-dd HH:mm:ss:SSS Z*.

Input Parameters

long_expression

Required. A field or expression of type Long representing the number of milliseconds since the epoch date (January 1, 1970 00:00:00:000 GMT).

Examples

Convert a number representing the number of milliseconds from the epoch to a human-readable date and time:

`EPOCH_MS_TO_DATE(1360260240000)` returns `2013-02-07T18:04:00:000Z` or February 7, 2013 18:04:00:000 GMT

Or if your data is in seconds instead of milliseconds:

`EPOCH_MS_TO_DATE(1360260240 * 1000)` returns `2013-02-07T18:04:00:000Z` or February 7, 2013 18:04:00:000 GMT

EXTRACT_AMOUNT

Description

EXTRACT_AMOUNT is a row function that takes a Currency value and extracts the numeric amount as a Numeric value.

Syntax

`EXTRACT_AMOUNT(currency_expression)`

Return Value

Returns a value of type Numeric.

Input Parameters

currency_expression

Required. A field or expression of type Currency.

Examples

Get the numeric values from the Salary field (Currency field type):

`EXTRACT_AMOUNT([Salary])`

EXTRACT_CODE

Description

EXTRACT_CODE is a row function that takes a Currency value and extracts the currency code as an Instance value.

Syntax

`EXTRACT_CODE(currency_expression)`

Return Value

Returns a value of type Instance.

Input Parameters

currency_expression

Required. A field or expression of type Currency.

Examples

Get the currency code information from the Salary field (Currency field type) as an Instance field:

```
EXTRACT_CODE([Salary])
```

EXTRACT_CODE_TEXT

Description

EXTRACT_CODE_TEXT is a row function that takes a Currency value and extracts the currency code as a Text value.

Syntax

```
EXTRACT_CODE_TEXT(currency_expression)
```

Return Value

Returns a value of type Text.

Input Parameters

currency_expression

Required. A field or expression of type Currency.

Examples

Get the currency code information from the Salary field (Currency field type) as a Text field:

```
EXTRACT_CODE_TEXT([Salary])
```

TO_BOOLEAN

Description

TO_BOOLEAN is a row function that converts Text, Boolean, Integer, Long, or Numeric values to Boolean.

Syntax

```
TO_BOOLEAN(expression)
```

Return Value

Returns one value per row of type Boolean.

Input Parameters

expression

Required. A field or expression of type Text, Boolean, Integer, Long, or Numeric.

The function converts these values to true:

1, 1.0, "true", "t", "yes", "y", "1"

The function converts these values to false:

0, 0.0, "false", "f", "no", "n", "0"

The function converts all other values to NULL.

Examples

Convert the values of the *is_contingent* field to a Boolean:

```
TO_BOOLEAN([is_contingent])
```

These expressions return true:

```
TO_BOOLEAN("TRUE")
```

```
TO_BOOLEAN("1")
```

```
TO_BOOLEAN(1.0)
```

These expressions return false:

```
TO_BOOLEAN("False")
```

```
TO_BOOLEAN("0")
```

```
TO_BOOLEAN(0.0)
```

These expressions return NULL:

```
TO_BOOLEAN("correct")
```

```
TO_BOOLEAN("1.0")
```

```
TO_BOOLEAN(1.1)
```

TO_CURRENCY

Description

TO_CURRENCY is a row function that converts Text values that contain valid currency-formatted data to Currency values.

Syntax

```
TO_CURRENCY(expression)
```

Return Value

Returns a value of type Currency.

Input Parameters

expression

Required. A field or expression of type Text that represents a valid currency-formatted value.

A valid currency-formatted value meets these requirements:

- Includes both the numeric value and currency code.
- Uses a period to separate digits before and after the decimal.
- Doesn't include any character to separate the thousands place.

Example: 10000 EUR and 12345.678 USD.

Examples

Convert this text value to a Currency field type:

```
TO_CURRENCY("1234.56 USD")
```

Convert the values of the Grant Price field to a Currency field type using the currency codes in the Grant Code field:

```
TO_CURRENCY(CONCAT(TO_STRING([Grant Price]), [Grant Code]))
```

Convert the Sale Price field Text field to a Currency field, but first transform the occurrence of any N/A values to NULL values using a CASE expression:

```
TO_CURRENCY(CASE WHEN [Sale Price]="N/A" then NULL ELSE [Sale Price] END)
```

TO_DATE

Description

TO_DATE is a row function that converts Text values to Date values, and specifies the format of the date and time elements in the string.

Syntax

```
TO_DATE(string_expression, "date_format")
```

Return Value

Returns one value per row of type Date (which by definition is in UTC).

Input Parameters

string_expression

Required. A field or expression of type Text.

date_format

Required. A pattern that describes how the date is formatted.

Examples

Define a new Date Prism calculated field based on the *order_date* base field, which contains timestamps in the format of: *2014.07.10 at 15:08:56 PDT*:

```
TO_DATE([order_date], "yyyy.MM.dd 'at' HH:mm:ss z")
```

Define a new Date Prism calculated field by first combining individual *month*, *day*, *year*, and *depart_time* fields (using CONCAT), and performing a transformation on *depart_time* to make sure three-digit times are converted to four-digit times (using REGEX_REPLACE):

```
TO_DATE(CONCAT([month], "/", [day], "/", [year], ":",  
REGEX_REPLACE([depart_time], "\b(\d{3})\b", "0$1")), "MM/dd/yyyy:HHmm")
```

Define a new Date Prism calculated field based on the *created_at* base field, which contains timestamps in the format of: *Sat Jan 25 16:35:23 +0800 2014* (this is the timestamp format returned by Twitter's API):

```
TO_DATE([created_at], "EEE MMM dd HH:mm:ss Z yyyy")
```

Related Information

Reference

[Reference: Date Format Symbols](#) on page 60

TO_DECIMAL

Description

TO_DECIMAL is a row function that converts Text, Boolean, Integer, Long, Double, or Numeric values to Numeric values with the default number of digits before and after the decimal point.

Syntax

TO_DECIMAL(*expression*)

Return Value

Returns one value per row of type Numeric with the default number of digits before and after the decimal point.

Input Parameters

expression

Required. A field or expression of type Text (must be numeric characters), Boolean, Integer, Long, Double, or Numeric.

Examples

Convert the values of the *average_rating* field to a Numeric field type:

```
TO_DECIMAL([average_rating])
```

Convert the *average_rating* field to a Numeric field type, but first transform the occurrence of any NA values to NULL values using a CASE expression:

```
TO_DECIMAL(CASE WHEN [average_rating]="N/A" then NULL ELSE [average_rating]
END)
```

TO_DOUBLE

Description

TO_DOUBLE is a row function that converts Text, Boolean, Integer, Long, or Double values to Double (a type of numeric) values.

Syntax

TO_DOUBLE(*expression*)

Return Value

Returns one value per row of type Double.

Input Parameters

expression

Required. A field or expression of type Text (must be numeric characters), Boolean, Integer, Long, or Double.

Examples

Convert the values of the *average_rating* field to a Double field type:

```
TO_DOUBLE([average_rating])
```

Convert the *average_rating* field to a Double field type, but first transform the occurrence of any *NA* values to *NULL* values using a CASE expression:

```
TO_DOUBLE(CASE WHEN [average_rating]="N/A" then NULL ELSE [average_rating]
END)
```

TO_INT

Description

TO_INT is a row function that converts Text, Boolean, Integer, Long, or Double values to Integer (whole number) values. When converting Double values, everything after the decimal will be truncated (not rounded up or down).

Syntax

```
TO_INT(expression)
```

Return Value

Returns one value per row of type Integer.

Input Parameters

expression

Required. A field or expression of type Text, Boolean, Integer, Long, or Double. If a Text field contains non-numeric characters, the function returns NULL.

Examples

Convert the values of the *average_rating* field to an Integer field type:

```
TO_INT([average_rating])
```

Convert the *flight_duration* field to an Integer field type, but first transform the occurrence of any *NA* values to NULL values using a CASE expression:

```
TO_INT(CASE WHEN [flight_duration]="N/A" then NULL ELSE [flight_duration] END)
```

TO_LONG

Description

TO_LONG is a row function that converts Text, Boolean, Integer, Long, Decimal, Date, or Double values to Long (whole number) values. When converting Decimal or Double values, everything after the decimal will be truncated (not rounded up or down).

Syntax

```
TO_LONG(expression)
```

Return Value

Returns one value per row of type Long.

Input Parameters

expression

Required. A field or expression of type Text (must be numeric characters only, no period or comma), Boolean, Integer, Long, Decimal, Date, or Double. When a Text field value includes a decimal, the function returns a NULL value.

Examples

Convert the values of the *average_rating* field to a Long field type:

```
TO_LONG([average_rating])
```

Convert the *average_rating* field to a Long field type, but first transform the occurrence of any *NA* values to *NULL* values using a CASE expression:

```
TO_LONG(CASE WHEN [average_rating]="N/A" then NULL ELSE [average_rating] END)
```

TO_STRING

Description

TO_STRING is a row function that converts values of other data types to Text (character) values.

Syntax

```
TO_STRING(expression)
```

```
TO_STRING(date_expression, date_format)
```

Return Value

Returns one value per row of type Text.

Input Parameters

expression

A field or expression of type Text, Boolean, Integer, Long, Numeric, Double, Instance, or Multi-Instance. When you convert an Instance or Multi-Instance field to a string, this function returns the unique identifier (WID), not the display name, of the field value. When a Multi-Instance field contains more than 1 value, this function concatenates each value into a single string with no spaces.

date_expression

A field or expression of type Date.

date_format

If converting a Date to Date, a pattern that describes how the date is formatted. See [TO_DATE](#) on page 150 for the date format patterns.

Examples

Convert the values of the *sku_number* field to a Text field type:

```
TO_STRING([sku_number])
```

Convert values in the *age* column into range-based groupings (binning), and cast output values to a Text:

```
TO_STRING(CASE WHEN [age] <= 25 THEN "0-25" WHEN [age] <= 50 THEN "26-50" ELSE "over 50" END)
```

Convert the values of a *timestamp* Date field to Text, where the timestamp values are in the format of: *2002.07.10 at 15:08:56 PDT*:

```
TO_STRING([timestamp], "yyyy.MM.dd 'at' HH:mm:ss z")
```

Date Functions

DAYS_BETWEEN

Description

DAYS_BETWEEN is a row function that calculates the whole number of days (ignoring time) between two date values (value1 - value2).

Syntax

DAYS_BETWEEN(*date_1*, *date_2*)

Return Value

Returns one value per row of type Integer.

Input Parameters

date_1

Required. A field or expression of type Date.

date_2

Required. A field or expression of type Date.

Examples

Calculate the number of days to ship a product by subtracting the value of the *order_date* field from the *ship_date* field:

```
DAYS_BETWEEN([ship_date], [order_date])
```

Calculate the number of days since a product's release by subtracting the value of the *release_date* field from the current date (the result of the TODAY expression):

```
DAYS_BETWEEN(TODAY(), [release_date])
```

DATE_ADD

Description

DATE_ADD is a row function that adds the specified time interval to a date value.

Syntax

DATE_ADD(*date*, *quantity*, "*interval*")

Return Value

Returns a value of type Date.

Input Parameters

date

Required. A field name or expression that returns a date value.

quantity

Required. An integer value. To add time intervals, use a positive integer. To subtract time intervals, use a negative integer.

interval

Required. One of the following time intervals:

- **millisecond** - Adds the specified number of milliseconds to a date value.
- **second** - Adds the specified number of seconds to a date value.
- **minute** - Adds the specified number of minutes to a date value.
- **hour** - Adds the specified number of hours to a date value.
- **day** - Adds the specified number of days to a date value.
- **week** - Adds the specified number of weeks to a date value.
- **month** - Adds the specified number of months to a date value.
- **quarter** - Adds the specified number of quarters to a date value.
- **year** - Adds the specified number of years to a date value.
- **weekyear** - Adds the specified number of weekyears to a date value.

Examples

Add 45 days to the value of the *invoice_date* field to calculate the date a payment is due:

```
DATE_ADD([invoice_date], 45, "day")
```

EXTRACT

Description

EXTRACT is a row function that returns the specified portion of a date value.

Syntax

```
EXTRACT("extract_value", date)
```

Return Value

Returns the specified extracted value as type Integer. EXTRACT removes leading zeros. For example, the month of April returns a value of 4, not 04.

Input Parameters

extract_value

Required. One of the following extract values:

- **millisecond** - Returns the millisecond portion of a date value. For example, an input date value of 2012-08-15 20:38:40.213 would return an integer value of 213.
- **second** - Returns the second portion of a date value. For example, an input date value of 2012-08-15 20:38:40.213 would return an integer value of 40.
- **minute** - Returns the minute portion of a date value. For example, an input date value of 2012-08-15 20:38:40.213 would return an integer value of 38.
- **hour** - Returns the hour portion of a date value. For example, an input date value of 2012-08-15 20:38:40.213 would return an integer value of 20.
- **day** - Returns the day portion of a date value. For example, an input date value of 2012-08-15 would return an integer value of **15**.
- **week** - Returns the ISO week number for the input date value. For example, an input date value of 2012-01-02 would return an integer value of 1 (the first ISO week of 2012 starts on Monday January 2). An input date value of 2012-01-01 would return an integer value of 52 (January 1, 2012 is part of the last ISO week of 2011).
- **month** - Returns the month portion of a date value. For example, an input date value of 2012-08-15 would return an integer value of 8.

- **quarter** - Returns the quarter number for the input date value, where quarters start on January 1, April 1, July 1, or October 1. For example, an input date value of 2012-08-15 would return an integer value of 3.
- **year** - Returns the year portion of a date value. For example, an input date value of 2012-01-01 would return an integer value of 2012.
- **weekyear** - Returns the year value that corresponds to the ISO week number of the input date value. For example, an input date value of 2012-01-02 would return an integer value of 2012 (the first ISO week of 2012 starts on Monday January 2). An input date value of 2012-01-01 would return an integer value of 2011 (January 1, 2012 is part of the last ISO week of 2011).

date

Required. A field name or expression that returns a date value.

Examples

Extract the hour portion from the *order_date* Date field:

```
EXTRACT("hour", [order_date])
```

Cast the value of the *order_date* Text field to a date value using TO_DATE, and extract the ISO week year:

```
EXTRACT("weekyear", TO_DATE([order_date], "MM/dd/yyyy HH:mm:ss"))
```

HOURS_BETWEEN**Description**

HOURS_BETWEEN is a row function that calculates the whole number of hours (ignoring minutes, seconds, and milliseconds) between two date values (value1 - value2).

Syntax

```
HOURS_BETWEEN(date_1, date_2)
```

Return Value

Returns one value per row of type Integer.

Input Parameters**date_1**

Required. A field or expression of type Date.

date_2

Required. A field or expression of type Date.

Examples

Calculate the number of hours to ship a product by subtracting the value of the *ship_date* field from the *order_date* field:

```
HOURS_BETWEEN([ship_date], [order_date])
```

MILLISECONDS_BETWEEN**Description**

MILLISECONDS_BETWEEN is a row function that calculates the whole number of milliseconds between two date values (value1 - value2).

Syntax

MILLISECONDS_BETWEEN(*date_1*, *date_2*)

Return Value

Returns one value per row of type Integer.

Input Parameters

date_1

Required. A field or expression of type Date.

date_2

Required. A field or expression of type Date.

Examples

Calculate the number of milliseconds it took to serve a web page by subtracting the value of the *request_timestamp* field from the *response_timestamp* field:

MILLISECONDS_BETWEEN([request_timestamp], [response_timestamp])

MINUTES_BETWEEN**Description**

MINUTES_BETWEEN is a row function that calculates the whole number of minutes (ignoring seconds and milliseconds) between two date values (value1 - value2).

Syntax

MINUTES_BETWEEN(*date_1*, *date_2*)

Return Value

Returns one value per row of type Integer.

Input Parameters

date_1

Required. A field or expression of type Date.

date_2

Required. A field or expression of type Date.

Examples

Calculate the number of minutes it took for a user to click on an advertisement by subtracting the value of the *impression_timestamp* field from the *conversion_timestamp* field:

MINUTES_BETWEEN([impression_timestamp], [conversion_timestamp])

SECONDS_BETWEEN**Description**

SECONDS_BETWEEN is a row function that calculates the whole number of seconds (ignoring milliseconds) between two date values (value1 - value2).

Syntax

```
SECONDS_BETWEEN(date_1, date_2)
```

Return Value

Returns one value per row of type Integer.

Input Parameters**date_1**

Required. A field or expression of type Date.

date_2

Required. A field or expression of type Date.

Examples

Calculate the number of seconds it took for a user to click on an advertisement by subtracting the value of the *impression_timestamp* field from the *conversion_timestamp* field:

```
SECONDS_BETWEEN([impression_timestamp], [conversion_timestamp])
```

TODAY**Description**

TODAY is a scalar function that returns the current system date and time as a date value (no time information). It can be used in other expressions involving Date type fields, such as YEAR_DIFF. Note that the value of TODAY is only evaluated at the time a dataset is published (it is not re-evaluated with each query).

Syntax

```
TODAY()
```

Return Value

Returns the current system date (no time) as a date value.

Examples

Calculate a user's age using YEAR_DIFF to subtract the value of the *birthdate* field from the current date:

```
YEAR_DIFF(TODAY(), [birthdate])
```

Calculate the number of days since a product's release using DAYS_BETWEEN to subtract the value of the *release_date* field from the current date:

```
DAYS_BETWEEN(TODAY(), [release_date])
```

TRUNC**Description**

TRUNC is a row function that truncates a date value to the specified format.

Syntax

```
TRUNC(date, "format")
```

Return Value

Returns a value of type Date truncated to the specified format.

Input Parameters

date

Required. A field or expression that returns a date value.

format

Required. One of the following format values:

- **millisecond** - Returns a date value truncated to millisecond granularity. Has no effect since millisecond is already the most granular format for date values. For example, an input date value of 2012-08-15 20:38:40.213 would return a date value of 2012-08-15 20:38:40.213.
- **second** - Returns a date value truncated to second granularity. For example, an input date value of 2012-08-15 20:38:40.213 would return a date value of 2012-08-15 20:38:40.000.
- **minute** - Returns a date value truncated to minute granularity. For example, an input date value of 2012-08-15 20:38:40.213 would return a date value of 2012-08-15 20:38:00.000.
- **hour** - Returns a date value truncated to hour granularity. For example, an input date value of 2012-08-15 20:38:40.213 would return a date value of 2012-08-15 20:00:00.000.
- **day** - Returns a date value truncated to day granularity. For example, an input date value of 2012-08-15 20:38:40.213 would return a date value of 2012-08-15 00:00:00.000.
- **week** - Returns a date value truncated to the first day of the week (starting on a Monday). For example, an input date value of 2012-08-15 (a Wednesday) would return a date value of 2012-08-13 (the Monday prior).
- **month** - Returns a date value truncated to the first day of the month. For example, an input date value of 2012-08-15 would return a date value of 2012-08-01.
- **quarter** - Returns a date value truncated to the first day of the quarter (January 1, April 1, July 1, or October 1). For example, an input date value of 2012-08-15 20:38:40.213 would return a date value of 2012-07-01.
- **year** - Returns a date value truncated to the first day of the year (January 1). For example, an input date value of 2012-08-15 would return a date value of 2012-01-01.
- **weekyear** - Returns a date value truncated to the first day of the ISO weekyear (the ISO week starting with the Monday which is nearest in time to January 1). For example, an input date value of 2008-08-15 would return a date value of 2007-12-31. The first day of the ISO weekyear for 2008 is December 31, 2007 (the prior Monday closest to January 1).

Examples

Truncate the *order_date* date field to day granularity:

```
TRUNC([order_date], "day")
```

Cast the value of the *order_date* Text field to a date value using TO_DATE, and truncate it to day granularity:

```
TRUNC(TO_DATE([order_date], "MM/dd/yyyy HH:mm:ss"), "day")
```

YEAR_DIFF

Description

YEAR_DIFF is a row function that calculates the fractional number of years between two date values (value1 - value2).

Syntax

YEAR_DIFF(*date_1*, *date_2*)

Return Value

Returns one value per row of type Double.

Input Parameters

date_1

Required. A field or expression of type Date.

date_2

Required. A field or expression of type Date.

Examples

Calculate the number of years a user has been a customer by subtracting the value of the *registration_date* field from the current date (the result of the TODAY expression):

```
YEAR_DIFF(TODAY(), [registration_date])
```

Calculate a user's age by subtracting the value of the *birthdate* field from the current date (the result of the TODAY expression):

```
YEAR_DIFF(TODAY(), [birthdate])
```

Informational Functions

IS_VALID

Description

IS_VALID is a row function that returns 0 if the returned value is NULL, and 1 if the returned value is NOT NULL. This is useful for computing other calculations where you want to exclude NULL values (such as when computing averages).

Syntax

IS_VALID(*expression*)

Return Value

Returns 0 if the returned value is NULL, and 1 if the returned value is NOT NULL.

Input Parameters

expression

Required. A field name or expression.

Examples

Define a Prism calculated field using `IS_VALID`. This returns a row count only for the rows where this field value is NOT NULL. If a value is NULL, it returns 0 for that row. In this example, we create a Prism calculated field (*sale_amount_not_null*) using the *sale_amount* field as the basis.

```
IS_VALID([sale_amount])
```

Then you can use the *sale_amount_not_null* Prism calculated field to calculate an accurate average for *sale_amount* that excludes NULL values:

```
SUM([sale_amount])/SUM([sale_amount_not_null])
```

Instance Functions

CREATE_MULTI_INSTANCE

Description

`CREATE_MULTI_INSTANCE` is a row function that constructs a Multi-Instance field from one or more provided Multi-Instance or Instance fields.

Syntax

```
CREATE_MULTI_INSTANCE(field_name [, field_name])
```

Return Value

Returns one value per row of type Multi-Instance.

Input Parameters

field_name

Required. A field of type Multi-Instance or Instance. All instance values must use the same business object.

Examples

Create a Multi-Instance field out of multiple Instance fields:

```
CREATE_MULTI_INSTANCE([Journal1], [Journal2], [Journal3])
```

Create a Multi-Instance field out of instance values:

```
CREATE_MULTI_INSTANCE(
  CAST("070b0d082eee44e1928c808cc739b35f" AS
    Instance(eecb565181284b6a8ae8b45dc3ed1451)),
  CAST("f4c49debb3dc483baa8707dfe683503c" AS
    Instance(eecb565181284b6a8ae8b45dc3ed1451))
)
```

INSTANCE_CONTAINS_ANY

Description

`INSTANCE_CONTAINS_ANY` is a row function that compares a Multi-Instance or Instance field to either a Multi-Instance field, an Instance field, or to a list of instance values, and returns True if at least one instance value exists in the first argument, and False if none of them exist.

Syntax

```
INSTANCE_CONTAINS_ANY(input_field, comparison_value , [comparison_value])
```

Return Value

Returns one value per row of type Boolean. This function returns NULL when it receives a Text value that isn't formatted as a valid instance value (WID format).

Input Parameters**input_field**

Required. A field of type Multi-Instance or Instance.

comparison_value

Required. A field of type Multi-Instance or Instance, or a Text value of a valid instance value.

Examples

Compare the Worktags Multi-Instance field to the Instance fields *Cost Center 1* and *Cost Center 2*:

```
INSTANCE_CONTAINS_ANY([Worktags], [Cost Center 1],[Cost Center 2])
```

INSTANCE_COUNT**Description**

INSTANCE_COUNT is a row function that returns the total number of instance values in a Multi-Instance or Instance field. This function returns 0 when the field is empty.

Syntax

```
INSTANCE_COUNT(field_name)
```

Return Value

Returns one value per row of type Integer.

Input Parameters**field_name**

Required. A field of type Multi-Instance or Instance.

Examples

Count the number of instance values in the *Journal Lines* Multi-Instance field:

```
INSTANCE_COUNT([Journal Lines])
```

INSTANCE_EQUALS**Description**

INSTANCE_EQUALS is a row function that compares a Multi-Instance or Instance field to either a Multi-Instance field, an Instance field, or to a list of instance values, and checks if the first argument exactly matches the instance values provided in the other arguments.

Syntax

```
INSTANCE_EQUALS(input_field, comparison_value [, comparison_value])
```

Return Value

Returns one value per row of type Boolean. This function returns NULL when it receives a Text value that isn't formatted as a valid instance value (WID format).

Input Parameters**input_field**

Required. A field of type Multi-Instance or Instance.

comparison_value

Required. A field of type Multi-Instance or Instance, or a Text value of a valid instance value.

Examples

Compare the *Worktags* Multi-Instance field to the Instance fields *Cost Center 1* and *Cost Center 2*:

```
INSTANCE_EQUALS([Worktags], [Cost Center 1],[Cost Center 2])
```

INSTANCE_IS_SUPERSET_OF**Description**

INSTANCE_IS_SUPERSET_OF is a row function that compares a Multi-Instance field to either a Multi-Instance field, an Instance field, or to a list of instance values, and returns True if every instance value exists in the first argument, and False if at least one doesn't exist.

Syntax

```
INSTANCE_IS_SUPERSET_OF(input_field, comparison_value [, comparison_value])
```

Return Value

Returns one value per row of type Boolean. This function returns NULL when it receives a Text value that isn't formatted as a valid instance value (WID format).

Input Parameters**input_field**

Required. A field of type Multi-Instance.

comparison_value

Required. A field of type Multi-Instance or Instance, or a Text value of a valid instance value.

Examples

Compare the *Worktags* Multi-Instance field to the Instance fields *Cost Center 1* and *Cost Center 2*:

```
INSTANCE_IS_SUPERSET_OF([Worktags], [Cost Center 1],[Cost Center 2])
```

Logical Functions

CASE

Description

CASE is a row function that evaluates each row in the dataset according to one or more input conditions, and outputs the specified result when the input conditions are met.

Syntax

```
CASE WHEN input_condition [AND|OR input_condition] THEN output_expression [...]
[ELSE other_output_expression] END
```

Return Value

Returns one value per row of the same type as the output expression. All output expressions must return the same field type.

If there are multiple output expressions that return different field types, then you will need to enclose your entire CASE expression in one of the field type conversion functions, such as `T0_INT`, to explicitly cast all output values to a particular field type.

Input Parameters

WHEN *input_condition*

Required. The WHEN keyword is used to specify one or more Boolean expressions (see the supported conditional operators). If an input value meets the condition, then the output expression is applied. Input conditions can include other row functions in their expression, but cannot contain summarization functions or measure expressions. You can use the AND or OR keywords to combine multiple input conditions.

THEN *output_expression*

Required. The THEN keyword is used to specify an output expression when the specified conditions are met. Output expressions can include other row functions in their expression, but cannot contain summarization functions or measure expressions.

ELSE *other_output_expression*

Optional. The ELSE keyword can be used to specify an alternate output expression to use when the specified conditions are not met. If an ELSE expression is not supplied, ELSE NULL is the default.

END

Required. Denotes the end of CASE function processing.

Examples

Convert values in the *age* column into range-based groupings (binning):

```
CASE WHEN [age] <= 25 THEN "0-25" WHEN [age] <= 50 THEN "26-50" ELSE "over 50"
END
```

Transform values in the *gender* column from one string to another:

```
CASE WHEN [gender] = "M" THEN "Male" WHEN [gender] = "F" THEN "Female" ELSE
"Unknown" END
```

The *vehicle* column contains the following values: truck, bus, car, scooter, wagon, bike, tricycle, and motorcycle. The following example converts multiple values in the *vehicle* column into a single value:

```
CASE WHEN [vehicle] in ("bike","scooter","motorcycle") THEN "two-wheelers"
ELSE "other" END
```

Related Information

Reference

[Comparison Operators](#) on page 142

[Logical Operators](#) on page 143

[Arithmetic Operators](#) on page 144

COALESCE

Description

COALESCE is a row function that returns the first valid value (NOT NULL value) from a comma-separated list of expressions.

Syntax

```
COALESCE(expression[,expression][, ...])
```

Return Value

Returns one value per row of the same type as the first valid input expression.

Input Parameters

expression

At least one required. A field name or expression.

Examples

The following example shows an expression to calculate employee yearly income for exempt employees that have a *salary* and non-exempt employees that have an *hourly_wage*. This expression checks the values of both fields for each row, and returns the value of the first expression that is valid (NOT NULL).

```
COALESCE([hourly_wage] * 40 * 52, [salary])
```

Math Functions

DIV

Description

DIV is a row function that divides two Long values and returns a quotient value of type Long (the result is truncated to 0 decimal places).

Syntax

```
DIV(dividend,divisor)
```

Return Value

Returns one value per row of type Long.

Input Parameters

dividend

Required. A field or expression of type Long.

divisor

Required. A field or expression of type Long.

Examples

Cast the value of the *file_size* field to Long and divide by 1024:

```
DIV(TO_LONG([file_size]), 1024)
```

EXP**Description**

EXP is a row function that raises the mathematical constant e to the power (exponent) of a numeric value and returns a value of type Double.

Syntax

```
EXP(power)
```

Return Value

Returns one value per row of type Double.

Input Parameters**power**

Required. A field or expression of a numeric type.

Examples

Raise e to the power in the *Value* field.

```
EXP([Value])
```

When the *Value* field value is 2.0, the result is equal to 7.3890 when truncated to four decimal places.

FLOOR**Description**

FLOOR is a row function that returns the largest integer that is less than or equal to the input argument.

Syntax

```
FLOOR(double)
```

Return Value

Returns one value per row of type Double.

Input Parameters**double**

Required. A field or expression of type Double.

Examples

Return the floor value of 32.6789:

```
FLOOR(32.6789) returns 32.0
```

HASH

Description

HASH is a row function that evenly partitions data values into the specified number of buckets. It creates a hash of the input value and assigns that value a bucket number. Equal values will always hash to the same bucket number.

Syntax

`HASH(field_name, integer)`

Return Value

Returns one value per row of type Integer corresponding to the bucket number that the input value hashes to.

Input Parameters

field_name

Required. The name of the field whose values you want to partition. When this value is NULL and the integer parameter is a value other than zero or NULL, the function returns zero, otherwise it returns NULL.

integer

Required. The desired number of buckets. This parameter can be a numeric value of any field type, but when it is a non-integer value, the value is truncated to an integer. When the value is zero or NULL, the function returns NULL. When the value is negative, the function uses absolute value.

Examples

Partition the values of the *username* field into 20 buckets:

`HASH([username], 20)`

LN

Description

LN is a row function that returns the natural logarithm of a number. The natural logarithm is the logarithm to the base e, where e (Euler's number) is a mathematical constant approximately equal to 2.718281828. The natural logarithm of a number x is the power to which the constant e must be raised in order to equal x.

Syntax

`LN(positive_number)`

Return Value

Returns the exponent to which base e must be raised to obtain the input value, where e denotes the constant number 2.718281828. The return value is the same field type as the input value.

For example, `LN(7.389)` is 2, because e to the power of 2 is approximately 7.389.

Input Parameters

positive_number

Required. A field or expression that returns a number greater than 0. Inputs can be of type Integer, Long, Double.

Examples

Return the natural logarithm of base number e, which is approximately 2.718281828:

`LN(2.718281828)` returns 1

`LN(3.0000)` returns 1.098612

`LN(300.0000)` returns 5.703782

MOD**Description**

MOD is a row function that divides two Long values and returns the remainder value of type Long (the result is truncated to 0 decimal places).

Syntax

`MOD(dividend, divisor)`

Return Value

Returns one value per row of type Long.

Input Parameters**dividend**

Required. A field or expression of type Long.

divisor

Required. A field or expression of type Long.

Examples

Cast the value of the *file_size* field to Long and divide by 1024:

`MOD(TO_LONG([file_size]), 1024)`

POW**Description**

POW is a row function that raises the a numeric value to the power (exponent) of another numeric value and returns a value of type Double.

Syntax

`POW(index, power)`

Return Value

Returns one value per row of type Double.

Input Parameters**index**

Required. A field or expression of a numeric type.

power

Required. A field or expression of a numeric type.

Examples

Calculate the compound annual growth rate (CAGR) percentage for a given investment over a five year span.

```
100 * POW([end_value]/[start_value], 0.2) - 1
```

Calculate the square of the *Value* field.

```
POW([Value], 2)
```

Calculate the square root of the *Value* field.

```
POW([Value], 0.5)
```

The following expression returns 1.

```
POW(0, 0)
```

ROUND**Description**

ROUND is a row function that rounds a numeric value to the specified number of decimal places and returns a value of type Double.

Syntax

```
ROUND(numeric_expression, number_decimal_places)
```

Return Value

Returns one value per row of type Double.

Input Parameters**numeric_expression**

Required. A field or expression of any numeric type.

number_decimal_places

Required. An integer that specifies the number of decimal places to round to.

Examples

Round the number 32.4678954 to two decimal places:

```
ROUND(32.4678954, 2) returns 32.47
```

Text Functions

CIDR_MATCH**Description**

CIDR_MATCH is a row function that compares two Text arguments representing a CIDR mask and an IP address, and returns 1 if the IP address falls within the specified subnet mask or 0 if it does not.

Syntax

```
CIDR_MATCH(CIDR_string, IP_string)
```

Return Value

Returns an Integer value of 1 if the IP address falls within the subnet indicated by the CIDR mask and 0 if it does not.

Input Parameters**CIDR_string**

Required. A field or expression that returns a Text value containing either an IPv4 or IPv6 CIDR mask (Classless InterDomain Routing subnet notation). An IPv4 CIDR mask can only successfully match IPv4 addresses, and an IPv6 CIDR mask can only successfully match IPv6 addresses.

IP_string

Required. A field or expression that returns a Text value containing either an IPv4 or IPv6 internet protocol (IP) address.

Examples

Compare an IPv4 CIDR subnet mask to an IPv4 IP address:

```
CIDR_MATCH("60.145.56.0/24", "60.145.56.246") returns 1
```

```
CIDR_MATCH("60.145.56.0/30", "60.145.56.246") returns 0
```

Compare an IPv6 CIDR subnet mask to an IPv6 IP address:

```
CIDR_MATCH("fe80::/70", "FE80::0202:B3FF:FE1E:8329") returns 1
```

```
CIDR_MATCH("fe80::/72", "FE80::0202:B3FF:FE1E:8329") returns 0
```

CONCAT**Description**

CONCAT is a row function that returns a Text by concatenating (combining together) the results of multiple Text expressions.

Syntax

```
CONCAT(value_expression[, value_expression][, ...])
```

Return Value

Returns one value per row of type Text.

Input Parameters**value_expression**

At least one required. A field name of any type, a literal string or number, or an expression that returns any value.

Examples

Combine the values of the *month*, *day*, and *year* fields into a single date field formatted as *MM/DD/YYYY*.

```
CONCAT([month], "/", [day], "/", [year])
```

EXTRACT_COOKIE

Description

EXTRACT_COOKIE is a row function that extracts the value of the given cookie identifier from a semi-colon delimited list of cookie key/value pairs. This function can be used to extract a particular cookie value from a combined web access log Cookie column.

Syntax

```
EXTRACT_COOKIE("cookie_list_string", cookie_key_string)
```

Return Value

Returns the value of the specified cookie key as type Text.

Input Parameters

cookie_list_string

Required. A field of type Text or literal string that has a semi-colon delimited list of cookie *key=value* pairs.

cookie_key_string

Required. The cookie key name for which to extract the cookie value.

Examples

Extract the value of the *vID* cookie from a literal cookie string:

```
EXTRACT_COOKIE("SSID=ABC; vID=44", "vID") returns 44
```

Extract the value of the *vID* cookie from a field named *Cookie*:

```
EXTRACT_COOKIE([Cookie], "vID")
```

EXTRACT_VALUE

Description

EXTRACT_VALUE is a row function that extracts the value for the given key from a string containing delimited key/value pairs.

Syntax

```
EXTRACT_VALUE(string, key_name [, delimiter], [pair_delimiter])
```

Return Value

Returns the value of the specified key as type Text.

Input Parameters

string

Required. A field of type Text or literal string that contains a delimited list of key/value pairs.

key_name

Required. The key name for which to extract the value.

delimiter

Optional. The delimiter used between the key and the value. If not specified, the value `u0003` is used. This is the Unicode escape sequence for the start of text character.

pair_delimiter

Optional. The delimiter used between key/value pairs when the input string contains more than one key/value pair. If not specified, the value `u0002` is used. This is the Unicode escape sequence for the end of text character.

Examples

Extract the value of the *lastname* key from a literal string of key/value pairs:

```
EXTRACT_VALUE("firstname;daria|lastname;hutch", "lastname", ";", "|") returns  
hutch
```

Extract the value of the *email* key from a Text field named *contact_info* that contains strings in the format of *key:value,key:value*:

```
EXTRACT_VALUE([contact_info], "email", ":", ",")
```

Related Information

Reference

[PACK_VALUES](#) on page 180

FILE_NAME

Description

FILE_NAME is a row function that returns the original file name from the source file system. This is useful when the source data that comprises a dataset comes from multiple files, and there is useful information in the file names themselves (such as dates or server names). You can use **FILE_NAME** in combination with other text processing functions to extract useful information from the file name.

Syntax

```
FILE_NAME( )
```

Return Value

Returns one value per row of type Text.

Examples

Your dataset is based on daily log files that use an 8 character date as part of the file name. For example, *20120704.log* is the file name used for the log file created on July 4, 2012. The following expression uses **FILE_NAME** in combination with **SUBSTRING** and **TO_DATE** to create a date field from the first 8 characters of the file name.

```
TO_DATE(SUBSTRING(FILE_NAME( ), 0, 8), "yyyymmdd")
```

Your dataset is based on log files that use the server IP address as part of the file name. For example, *172.12.131.118.log* is the log file name for server 172.12.131.118. The following expression uses **FILE_NAME** in combination with **REGEX** to extract the IP address from the file name.

```
REGEX(FILE_NAME( ), "(\\d{1,3}\\.(\\d{1,3}\\.(\\d{1,3}\\.(\\d{1,3})\\.log"))
```

HEX_TO_IP

Description

HEX_TO_IP is a row function that converts a hexadecimal-encoded Text value to a text representation of an IP address.

Syntax

HEX_TO_IP(*string*)

Return Value

Returns a value of type Text representing either an IPv4 or IPv6 address. The type of IP address returned depends on the input string. An 8 character hexadecimal string returns an IPv4 address. A 32 character long hexadecimal string returns an IPv6 address.

IPv6 addresses are represented in full length, without removing any leading zeros and without using the compressed :: notation. For example, 2001:0db8:0000:0000:ff00:0042:8329 rather than 2001:db8::ff00:42:8329.

Input strings that don't contain either 8 or 32 valid hexadecimal characters return NULL.

Input Parameters

string

Required. A field or expression that returns a hexadecimal-encoded Text value. The hexadecimal string must be either 8 characters long (in which case it's converted to an IPv4 address) or 32 characters long (in which case it's converted to an IPv6 address).

Examples

Return a plain text IP address for each hexadecimal-encoded string value in the *byte_encoded_ips* column:

```
HEX_TO_IP([byte_encoded_ips])
```

Convert an 8 character hexadecimal-encoded string to a plain text IPv4 address:

```
HEX_TO_IP(AB20FE01) returns 171.32.254.1
```

Convert a 32 character hexadecimal-encoded string to a plain text IPv6 address:

```
HEX_TO_IP(FE800000000000000000000000000000000202B3FFFE1E8329) returns  
fe80:0000:0000:0000:0202:b3ff:fe1e:8329
```

INSTR

Description

INSTR is a row function that returns an integer indicating the position of a character within a string that is the first character of the occurrence of a substring. The INSTR function is similar to the FIND function in Excel, except that the first letter is position 0 and the order of the arguments is reversed.

Syntax

INSTR(*search_string*,*substring*,*position*,*occurrence*)

Return Value

Returns one value per row of type Integer. The first position is indicated with the value of zero (0).

Input Parameters**search_string**

Required. The name of a field or expression of type Text (or a literal string).

substring

Required. A literal string or name of a field that specifies the substring to search for in *search_string*. Note that to search for the double quotation mark (") as a literal string, you must escape it with another double quotation mark: ""

position

Optional. An integer that specifies at which character in *search_string* to start searching for *substring*. A value of 0 (zero) starts the search at the beginning of *search_string*. Use a positive integer to start searching from the beginning of *search_string*, and use a negative integer to start searching from the end of *search_string*. When no position is specified, INSTR searches at the beginning of the string (0).

occurrence

Optional. A positive integer that specifies which occurrence of *substring* to search for. When no occurrence is specified, INSTR searches for the first occurrence of the substring (1).

Examples

Return the position of the first occurrence of the substring "http://" starting at the end of the *url* field:

```
INSTR([url], "http://", -1, 1)
```

The following expression searches for the second occurrence of the substring "st" starting at the beginning of the string "bestteststring". INSTR finds that the substring starts at the seventh character in the string, so it returns 6:

```
INSTR("bestteststring", "st", 0, 2)
```

The following expression searches backward for the second occurrence of the substring "st" starting at 7 characters before the end of the string "bestteststring". INSTR finds that the substring starts at the third character in the string, so it returns 2:

```
INSTR("bestteststring", "st", -7, 2)
```

JAVA_STRING**Description**

JAVA_STRING is a row function that returns the unescaped version of a Java unicode character escape sequence as a Text value. This is useful when you want to specify unicode characters in an expression. For example, you can use JAVA_STRING to specify the unicode value representing a control character.

Syntax

```
JAVA_STRING(unicode_escape_sequence)
```

Return Value

Returns the unescaped version of the specified unicode character, one value per row of type Text.

Input Parameters**unicode_escape_sequence**

Required. A Text value containing a unicode character expressed as a Java unicode escape sequence. Unicode escape sequences consist of a backslash '\' (ASCII character 92, hex 0x5c), a 'u' (ASCII 117, hex 0x75), optionally one or more additional 'u' characters,

and four hexadecimal digits (the characters '0' through '9' or 'a' through 'f' or 'A' through 'F'). Such sequences represent the UTF-16 encoding of a Unicode character. For example, the letter 'a' is equivalent to '\u0061'.

Examples

Evaluates whether the *currency* field is equal to the yen symbol.

```
CASE WHEN [currency] == JAVA_STRING("\u00a5") THEN "yes" ELSE "no" END
```

JOIN_STRINGS

Description

JOIN_STRINGS is a row function that returns a Text by concatenating (combining together) the results of multiple Text values with the separator in between each non-null value.

Syntax

```
JOIN_STRINGS(separator, value_expression, [value_expression][, ...])
```

Return Value

Returns one value per row of type Text.

Input Parameters

separator

Required. A field name of type Text, a literal string, or an expression that returns a Text.

value_expression

At least one required. A field name of any type, a literal string or number, or an expression that returns any value.

Examples

Combine the values of the *month*, *day*, and *year* fields into a single date field formatted as *MM/DD/YYYY*.

```
JOIN_STRINGS("/", [month], [day], [year])
```

The following expression returns NULL:

```
JOIN_STRINGS("+", NULL, NULL, NULL)
```

The following expression returns a+b:

```
JOIN_STRINGS("+", "a", "b", NULL)
```

JSON_DECIMAL

Description

JSON_DECIMAL is a row function that extracts a Numeric value from a field in a JSON object.

Syntax

```
JSON_DECIMAL(json_string, "json_field")
```

Return Value

Returns one value per row of type Numeric.

Input Parameters**json_string**

Required. The name of a field or expression of type Text (or a literal string) that contains a valid JSON object.

json_field

Required. The key or name of the field value you want to extract.

For top-level fields, specify the name identifier (key) of the field.

To access fields within a nested object, specify a dot-separated path of field names (for example *top_level_field_name.nested_field_name*).

To extract a value from an array, specify the dot-separated path of field names and the array position starting at 0 for the first value in an array, 1 for the second value, and so on (for example, *field_name.0*).

If the name identifier contains dot or period characters within the name itself, escape the name by enclosing it in brackets (for example, *[field.name.with.dot]*. *[another.dot.field.name]*

If the field name is null (empty), use brackets with nothing in between as the identifier, for example *[]*.

Examples

If you had a *top_scores* field that contained a JSON object formatted like this (with the values contained in an array):

```
{"practice_scores":["538.67","674.99","1021.52"], "test_scores":["753.21","957.88","1032.87"]}
```

You could extract the third value of the *test_scores* array using the expression, which returns "1032.87":
JSON_DECIMAL([top_scores],"test_scores.2")

JSON_DOUBLE**Description**

JSON_DOUBLE is a row function that extracts a Double value from a field in a JSON object.

Syntax

```
JSON_DOUBLE(json_string, "json_field")
```

Return Value

Returns one value per row of type Double.

Input Parameters**json_string**

Required. The name of a field or expression of type Text (or a literal string) that contains a valid JSON object.

json_field

Required. The key or name of the field value you want to extract.

For top-level fields, specify the name identifier (key) of the field.

To access fields within a nested object, specify a dot-separated path of field names (for example *top_level_field_name.nested_field_name*).

To extract a value from an array, specify the dot-separated path of field names and the array position starting at 0 for the first value in an array, 1 for the second value, and so on (for example, *field_name.0*).

If the name identifier contains dot or period characters within the name itself, escape the name by enclosing it in brackets (for example, *[field.name.with.dot]*. *[another.dot.field.name]*)

If the field name is null (empty), use brackets with nothing in between as the identifier, for example *[]*.

Examples

If you had a *top_scores* field that contained a JSON object formatted like this (with the values contained in an array):

```
{"practice_scores":["538.67","674.99","1021.52"], "test_scores":
["753.21","957.88","1032.87"]}
```

You could extract the third value of the *test_scores* array using the expression, which returns "1032.87":

```
JSON_DOUBLE([top_scores],"test_scores.2")
```

JSON_INTEGER

Description

JSON_INTEGER is a row function that extracts an Integer value from a field in a JSON object.

Syntax

```
JSON_INTEGER(json_string, "json_field")
```

Return Value

Returns one value per row of type Integer.

Input Parameters

json_string

Required. The name of a field or expression of type Text (or a literal string) that contains a valid JSON object.

json_field

Required. The key or name of the field value you want to extract.

For top-level fields, specify the name identifier (key) of the field.

To access fields within a nested object, specify a dot-separated path of field names (for example *top_level_field_name.nested_field_name*).

To extract a value from an array, specify the dot-separated path of field names and the array position starting at 0 for the first value in an array, 1 for the second value, and so on (for example, *field_name.0*).

If the name identifier contains dot or period characters within the name itself, escape the name by enclosing it in brackets (for example, *[field.name.with.dot]*. *[another.dot.field.name]*)

If the field name is null (empty), use brackets with nothing in between as the identifier, for example *[]*.

Examples

If you had an *address* field that contained a JSON object formatted like this:

```
{"street_address":"123 B Street", "city":"San Mateo", "state":"CA", "zip_code":"94403"}
```

You could extract the *zip_code* value using the expression, which returns "94403":

```
JSON_INTEGER([address], "zip_code")
```

If you had a *top_scores* field that contained a JSON object formatted like this (with the values contained in an array):

```
{"practice_scores":["538", "674", "1021"], "test_scores":["753", "957", "1032"]}
```

You could extract the third value of the *test_scores* array using the expression, which returns "1032":

```
JSON_INTEGER([top_scores], "test_scores.2")
```

JSON_LONG

Description

JSON_LONG is a row function that extracts a Long value from a field in a JSON object.

Syntax

```
JSON_LONG(json_string, "json_field")
```

Return Value

Returns one value per row of type Long.

Input Parameters

json_string

Required. The name of a field or expression of type Text (or a literal string) that contains a valid JSON object.

json_field

Required. The key or name of the field value you want to extract.

For top-level fields, specify the name identifier (key) of the field.

To access fields within a nested object, specify a dot-separated path of field names (for example *top_level_field_name.nested_field_name*).

To extract a value from an array, specify the dot-separated path of field names and the array position starting at 0 for the first value in an array, 1 for the second value, and so on (for example, *field_name.0*).

If the name identifier contains dot or period characters within the name itself, escape the name by enclosing it in brackets (for example, *[field.name.with.dot]*, *[another.dot.field.name]*).

If the field name is null (empty), use brackets with nothing in between as the identifier, for example *[]*.

Examples

If you had a *top_scores* field that contained a JSON object formatted like this (with the values contained in an array):

```
{"practice_scores":["538", "674", "1021"], "test_scores":["753", "957", "1032"]}
```

You could extract the third value of the *test_scores* array using the expression, which returns "1032":

```
JSON_LONG([top_scores], "test_scores.2")
```

JSON_STRING

Description

JSON_STRING is a row function that extracts a Text value from a field in a JSON object.

Syntax

```
JSON_STRING(json_string, "json_field")
```

Return Value

Returns one value per row of type Text.

Input Parameters

json_string

Required. The name of a field or expression of type Text (or a literal string) that contains a valid JSON object.

json_field

Required. The key or name of the field value you want to extract.

For top-level fields, specify the name identifier (key) of the field.

To access fields within a nested object, specify a dot-separated path of field names (for example *top_level_field_name.nested_field_name*).

To extract a value from an array, specify the dot-separated path of field names and the array position starting at 0 for the first value in an array, 1 for the second value, and so on (for example, *field_name.0*).

If the name identifier contains dot or period characters within the name itself, escape the name by enclosing it in brackets (for example, *[field.name.with.dot]*, *[another.dot.field.name]*).

If the field name is null (empty), use brackets with nothing in between as the identifier, for example *[]*.

Examples

If you had an *address* field that contained a JSON object formatted like this:

```
{"street_address": "123 B Street", "city": "San Mateo", "state": "CA", "zip": "94403"}
```

You could extract the *state* value using the expression:

```
JSON_STRING([address], "state")
```

If you had a *misc* field that contained a JSON object formatted like this (with the values contained in an array):

```
{"hobbies": ["sailing", "hiking", "cooking"], "interests": ["art", "music", "travel"]}
```

You could extract the first value of the *hobbies* array using the expression, which returns "sailing":

```
JSON_STRING([misc], "hobbies.0")
```

LENGTH

Description

LENGTH is a row function that returns the count of characters in a Text value.

Syntax

LENGTH(*string_expression*)

Return Value

Returns one value per row of type Integer.

Input Parameters

string_expression

Required. The name of a field or expression of type Text (or a literal string).

Examples

Return count of characters from values in the *name* field. For example, the value *Bob* would return a length of 3, *Julie* would return a length of 5, and so on:

```
LENGTH([name])
```

PACK_VALUES

Description

PACK_VALUES is a row function that returns multiple output values packed into a single string of key/value pairs separated by the default key and pair separators. The string returned is in a format that can be read by the EXTRACT_VALUE function. PACK_VALUES uses the same key and pair separator values that EXTRACT_VALUE uses (the Unicode escape sequences u0003 and u0002, respectively).

Syntax

PACK_VALUES(*key, value*[, *key, value*][, ...])

Return Value

Returns one value per row of type Text. If the value for either key or value of a pair is null or contains either of the separator values, the full key/value pair is omitted from the return value.

The key separator is u0003, which is the Unicode escape sequence for the start of text character. The pair separator is u0002, which is the Unicode escape sequence for the end of text character.

Input Parameters

key

At least one required. A field name of any type, a literal string or number, or an expression that returns any value.

value

At least one required. A field name of any type, a literal string or number, or an expression that returns any value. The expression must include one value instance for each key instance.

Examples

Combine the values of the [custid] and [age] fields into a single text field.

```
PACK_VALUES("ID", [custid], "Age", [age])
```

This expression returns ID\000035555\00002Age\0000329 when the value of the [custid] field is 5555 and the value of the [age] field is 29:

```
PACK_VALUES("ID", [custid], "Age", [age])
```

This expression returns Age\0000329 when the value of the [age] field is 29:

```
PACK_VALUES("ID", NULL, "Age", [age])
```

This expression returns 29 as a Text value when the [age] field is an Integer and its value is 29:

```
EXTRACT_VALUE(PACK_VALUES("ID", [custid], "Age", [age]), "Age")
```

Related Information

Reference

[EXTRACT_VALUE](#) on page 171

REGEX

Description

REGEX is a row function that performs a whole string match against a Text value with a regular expression and returns the portion of the string matching the first capturing group of the regular expression.

Syntax

```
REGEX(string_expression, "regex_matching_pattern")
```

Return Value

Returns the matched Text value of the first capturing group of the regular expression. If there is no match, returns NULL.

Input Parameters

string_expression

Required. The name of a field or expression of type Text (or a literal string).

regex_matching_pattern

Required. A regular expression pattern based on the regular expression pattern matching syntax of the Java programming language. To return a non-NULL value, the regular expression pattern must match the entire Text value.

Regular Expression Constructs

See the Regular Expression Reference for information on the constructs used for defining a regular expression matching pattern.

Capturing and Non-Capturing Groups

Groups are specified by a pair of parenthesis around a subpattern in the regular expression. A pattern can have more than one group and the groups can be nested. The groups are numbered 1-*n* from left to right, starting with the first opening parenthesis. There is always an implicit group 0, which contains the entire match. For example, the pattern:

```
(a(b*))+(c)
```

contains three groups:

```
group 1: (a(b*))
group 2: (b*)
group 3: (c)
```

Capturing Groups

By default, a group *captures* the text that produces a match, and only the most recent match is captured. The REGEX function returns the string that matches the first capturing group in the regular expression. For example, if the input string to the expression above was `abc`, the entire REGEX function would match to `abc`, but only return the result of group 1, which is `ab`.

Non-Capturing Groups

In some cases, you may want to use parenthesis to group subpatterns, but not capture text. A non-capturing group starts with `(?:` (a question mark and colon following the opening parenthesis). For example, `h(?:a|i|o)t` matches `hat` or `hit` or `hot`, but does not capture the `a`, `i`, or `o` from the subexpression.

Examples

Match all possible email addresses with a pattern of `username@provider.domain`, but only return the *provider* portion of the email address from the *email* field:

```
REGEX([email], "^([a-zA-Z0-9._%+-]+)@([a-zA-Z0-9._-]+)\\.([a-zA-Z]{2,4})$")
```

Match the request line of a web log, where the value is in the format of:

```
GET /some_page.html HTTP/1.1
```

and return just the requested HTML page names:

```
REGEX(weblog.request_line, "GET\\s/([a-zA-Z0-9._%-]+\\.([html]))sHTTP/[0-9.]+")
```

Extract the inches portion from a *height* field where example values are `6'2"`, `5'11"` (notice the escaping of the literal quote with a double double-quote):

```
REGEX([height], "\\d\\'(\d)+\"")
```

Extract all of the contents of the *device* field when the value is either `iPod`, `iPad`, or `iPhone`:

```
REGEX([device], "(iP[ao]d|iPhone)")
```

Related Information

Concepts

[Concept: Regular Expressions in Prism](#) on page 218

Reference

[Regex Literal and Special Characters](#) on page 218

[Regex Character Classes](#) on page 219

[Regex Line and Word Boundaries](#) on page 221

[Regex Quantifiers](#) on page 221

[Regex Capturing Groups](#) on page 222

REGEX_REPLACE

Description

REGEX_REPLACE is a row function that evaluates a *Text* value against a regular expression to determine if there is a match, and replaces matched strings with the specified replacement value.

Syntax

```
REGEX_REPLACE(string_expression, "regex_match_pattern", "regex_replace_pattern")
```

Return Value

Returns the *regex_replace_pattern* as a Text value when *regex_match_pattern* produces a match. If there is no match, returns the value of *string_expression* as a Text.

Input Parameters***string_expression***

Required. The name of a field or expression of type Text (or a literal string).

regex_match_pattern

Required. A string literal or regular expression pattern based on the regular expression pattern matching syntax of the Java programming language. You can use capturing groups to create backreferences that can be used in the *regex_replace_pattern*. You might want to use a string literal to make a case-sensitive match. For example, when you enter *jane* as the match value, the function matches *jane* but not *Jane*. The function matches all occurrences of a string literal in the string expression.

regex_replace_pattern

Required. A string literal or regular expression pattern based on the regular expression pattern matching syntax of the Java programming language. You can refer to backreferences from the *regex_match_pattern* using the syntax *\$n* (where *n* is the group number).

Regular Expression Constructs

See the Regular Expression Reference for information on the constructs used for defining a regular expression matching pattern.

Examples

Match the values in a *phone_number* field where phone number values are formatted as xxx.xxx.xxxx and replace them with phone number values formatted as (xxx) xxx-xxxx:

```
REGEX_REPLACE([phone_number], "([0-9]{3})\.([0-9]{3})\.([0-9]{4})", "\($1\)$2-$3")
```

Match the values in a *name* field where name values are formatted as *firstname lastname* and replace them with name values formatted as *lastname, firstname*:

```
REGEX_REPLACE([name], "(.*) (.*)", "$2, $1")
```

Match the string literal *mrs* in a *title* field and replace it with the string literal *Mrs*.

```
REGEX_REPLACE([title], "mrs", "Mrs")
```

Related Information**Concepts**

[Concept: Regular Expressions in Prism](#) on page 218

Reference

[Regex Literal and Special Characters](#) on page 218

[Regex Character Classes](#) on page 219

[Regex Line and Word Boundaries](#) on page 221

[Regex Quantifiers](#) on page 221

[Regex Capturing Groups](#) on page 222

REVERSE

Description

REVERSE is a row function that returns the characters of a string value in the opposite order.

Syntax

REVERSE(*string_expression*)

Return Value

Returns one value per row of type Text.

Input Parameters

string_expression

Required. The name of a field or expression of type Text (or a literal string).

Examples

Return the string *123 Main Street* in reverse order:

```
REVERSE("123 Main Street")
```

Returns teertS niaM 321.

SUBSTRING

Description

SUBSTRING is a row function that returns the specified characters of a Text value based on the given start and optional end position.

Syntax

SUBSTRING(*search_string*, *start*, *end*)

Return Value

Returns one value per row of type Text.

Input Parameters

search_string

Required. The name of a field or expression of type Text (or a literal string).

start

Required. An integer that specifies where the returned characters start (inclusive), with 0 being the first character of the string. If *start* is greater than the number of characters, then an empty string is returned. If *start* is greater than *end*, then an empty string is returned.

end

Optional. A positive integer that specifies where the returned characters end (exclusive), with the *end* character not being part of the return value. If *end* is greater than the number of characters, or is not specified, then the whole string value (from *start*) is returned.

Examples

Return the first letter of the *name* field:

SUBSTRING([name], 0, 1)

TO_LOWER

Description

TO_LOWER is a row function that converts all alphabetic characters in a Text value to lower case.

Syntax

TO_LOWER(*string_expression*)

Return Value

Returns one value per row of type Text.

Input Parameters

string_expression

Required. The name of a field or expression of type Text (or a literal string).

Examples

Return the literal input string *123 Main Street* in all lower case letters:

TO_LOWER("123 Main Street") returns *123 main street*

TO_PROPER

Description

TO_PROPER is a row function that returns a Text value with the first letter of each word capitalized.

Syntax

TO_PROPER(*string_expression*)

Return Value

Returns one value per row of type Text.

Input Parameters

string_expression

Required. The name of a field or expression of type Text (or a literal string).

Examples

TO_PROPER("123 Alameda de las Pulgas, San Mateo CA")

Returns 123 Alameda De Las Pulgas, San Mateo Ca

TO_UPPER

Description

TO_UPPER is a row function that converts all alphabetic characters in a Text value to upper case.

Syntax

TO_UPPER(*string_expression*)

Return Value

Returns one value per row of type Text.

Input Parameters**string_expression**

Required. The name of a field or expression of type Text (or a literal string).

Examples

```
TO_UPPER("123 Main Street")
```

Returns 123 MAIN STREET

TRIM**Description**

TRIM is a row function that removes leading and trailing spaces from a Text value.

Syntax

```
TRIM(string_expression)
```

Return Value

Returns one value per row of type Text.

Input Parameters**string_expression**

Required. The name of a field or expression of type Text (or a literal string).

Examples

Return the value of the *area_code* field without any leading or trailing spaces. Example:

```
TRIM([area_code])
```

TRIM(" 650 ") returns 650.

TRIM(" 650 123-4567 ") returns 650 123-4567. Note that the extra spaces in the middle of the string aren't removed, only the spaces at the beginning and end of the string.

XPATH_STRING**Description**

XPATH_STRING is a row function that takes XML and returns the first string matching the given XPath expression.

Syntax

```
XPATH_STRING(xml_expression, "xpath_expression")
```

Return Value

Returns one value per row of type Text.

If the XPath expression matches more than one string in the given XML node, this function will return the *first* match only. To return all matches, use `XPATH_STRINGS` instead.

Input Parameters

`xml_expression`

Required. The name of a field of type `Text` or a literal string that contains a valid XML node (a snippet of XML consisting of a parent element and one or more child nodes).

`xpath_expression`

Required. An XPath expression that refers to a node, element, or attribute within the XML string passed to this expression. Any XPath expression that complies to the [XML Path Language \(XPath\) Version 1.0](#) specification is valid.

Examples

These example `XPATH_STRING` expressions assume you have a field in your dataset named *address* that contains XML-formatted strings such as this:

```
<list>
  <address type="work">
    <street1>1300 So. El Camino Real</street1>
    <street2>Suite 600</street2>
    <city>San Mateo</city>
    <state>CA</state>
    <zipcode>94403</zipcode>
  </address>
  <address type="home">
    <street1>123 Oakdale Street</street1>
    <street2/>
    <city>San Francisco</city>
    <state>CA</state>
    <zipcode>94123</zipcode>
  </address>
</list>
```

Get the *zipcode* value from any *address* element where the *type* attribute equals *home*:

```
XPATH_STRING([address], "//*[address[@type='home']/zipcode]")
```

returns: 94123

Get the *city* value from the second *address* element:

```
XPATH_STRING([address], "/list/address[2]/city")
```

returns: San Francisco

Get the values from all child elements of the first address element (as one string):

```
XPATH_STRING([address], "/list/address")
```

returns: 1300 So. El Camino RealSuite 600 San MateoCA94403

URL Functions

URL_AUTHORITY

Description

URL_AUTHORITY is a row function that returns the authority portion of a URL string. The authority portion of a URL is the part that has the information on how to locate and connect to the server.

Syntax

URL_AUTHORITY(*URL_string*)

Return Value

Returns the authority portion of a URL as a Text value, or NULL if the input string is not a valid URL.

For example, in the string `http://www.workday.com/company/contact.html`, the authority portion is `www.workday.com`.

In the string `http://user:password@mycompany.com:8012/mypage.html`, the authority portion is `user:password@mycompany.com:8012`.

In the string `mailto:username@mycompany.com?subject=Topic`, the authority portion is NULL.

Input Parameters

URL_string

Required. A field or expression that returns a Text value in URI (uniform resource identifier) format of: *protocol:authority[/path][?query][#fragment]*.

The authority portion of the URL contains the host information, which can be specified as a domain name (`www.workday.com`), a host name (`localhost`), or an IP address (`127.0.0.1`). The host information can be preceded by optional user information terminated with `@` (for example, `username:password@workday.com`), and followed by an optional port number preceded by a colon (for example, `localhost:8001`).

Examples

Return the authority portion of URL string values in the *referrer* field:

URL_AUTHORITY([referrer])

Return the authority portion of a literal URL string:

URL_AUTHORITY("http://user:password@mycompany.com:8012/mypage.html") returns `user:password@mycompany.com:8012`.

URL_FRAGMENT

Description

URL_FRAGMENT is a row function that returns the fragment portion of a URL string.

Syntax

URL_FRAGMENT(*URL_string*)

Return Value

Returns the fragment portion of a URL as a Text value, NULL if the URL or does not contain a fragment, or NULL if the input string is not a valid URL.

For example, in the string `http://www.workday.com/contact.html#phone`, the fragment portion is `phone`.

In the string `http://www.workday.com/contact.html`, the fragment portion is NULL.

In the string `http://workday.com/news.php?topic=press#Workday%20News`, the fragment portion is `Workday%20News`.

Input Parameters**URL_string**

Required. A field or expression that returns a Text value in URI (uniform resource identifier) format of: `protocol:authority[/path][?query][#fragment]`.

The optional fragment portion of the URL is separated by a hash mark (#) and provides direction to a secondary resource, such as a heading or anchor identifier.

Examples

Return the fragment portion of URL string values in the *request* field:

```
URL_FRAGMENT([request])
```

Return the fragment portion of a literal URL string:

```
URL_FRAGMENT("http://workday.com/news.php?topic=press#Workday%20News") returns Workday%20News.
```

Return and decode the fragment portion of a literal URL string:

```
URLDECODE(URL_FRAGMENT("http://workday.com/news.php?topic=press#Workday%20News")) returns Workday News.
```

URL_HOST**Description**

URL_HOST is a row function that returns the host, domain, or IP address portion of a URL string.

Syntax

```
URL_HOST(URL_string)
```

Return Value

Returns the host portion of a URL as a Text value, or NULL if the input string is not a valid URL.

For example, in the string `http://www.workday.com/company/contact.html`, the host portion is `www.workday.com`.

In the string `http://admin:admin@127.0.0.1:8001/index.html`, the host portion is `127.0.0.1`.

In the string `mailto:username@mycompany.com?subject=Topic`, the host portion is NULL.

Input Parameters**URL_string**

Required. A field or expression that returns a Text value in URI (uniform resource identifier) format of: `protocol:authority[/path][?query][#fragment]`.

The authority portion of the URL contains the host information, which can be specified as a domain name (`www.workday.com`), a host name (`localhost`), or an IP address (`127.0.0.1`).

Examples

Return the host portion of URL string values in the *referrer* field:

```
URL_HOST([referrer])
```

Return the host portion of a literal URL string:

```
URL_HOST("http://user:password@mycompany.com:8012/mypage.html") returns mycompany.com.
```

URL_PATH

Description

URL_PATH is a row function that returns the path portion of a URL string.

Syntax

```
URL_PATH(URL_string)
```

Return Value

Returns the path portion of a URL as a Text value, NULL if the URL or does not contain a path, or NULL if the input string is not a valid URL.

For example, in the string `http://www.workday.com/company/contact.html`, the path portion is `/company/contact.html`.

In the string `http://admin:admin@127.0.0.1:8001/index.html`, the path portion is `/index.html`.

In the string `mailto:username@mycompany.com?subject=Topic`, the path portion is `username@mycompany.com`.

Input Parameters

URL_string

Required. A field or expression that returns a Text value in URI (uniform resource identifier) format of: *protocol:authority[/path][?query][#fragment]*.

The optional path portion of the URL is a sequence of resource location segments separated by a forward slash (/), conceptually similar to a directory path.

Examples

Return the path portion of URL string values in the *request* field:

```
URL_PATH([request])
```

Return the path portion of a literal URL string:

```
URL_PATH("http://workday.com/company/contact.html") returns /company/contact.html.
```

URL_PORT

Description

URL_PORT is a row function that returns the port portion of a URL string.

Syntax

`URL_PORT(URL_string)`

Return Value

Returns the port portion of a URL as an Integer value. If the URL does not specify a port, then returns -1. If the input string is not a valid URL, returns NULL.

For example, in the string `http://localhost:8001`, the port portion is 8001.

Input Parameters**URL_string**

Required. A field or expression that returns a Text value in URI (uniform resource identifier) format of: *protocol:authority[/path][?query][#fragment]*.

The authority portion of the URL contains the host information, which can be specified as a domain name (`www.workday.com`), a host name (`localhost`), or an IP address (`127.0.0.1`). The host information can be followed by an optional port number preceded by a colon (for example, `localhost:8001`).

Examples

Return the port portion of URL string values in the *referrer* field:

`URL_PORT([referrer])`

Return the port portion of a literal URL string:

`URL_PORT("http://user:password@mycompany.com:8012/mypage.html")` returns 8012.

URL_PROTOCOL**Description**

`URL_PROTOCOL` is a row function that returns the protocol (or URI scheme name) portion of a URL string.

Syntax

`URL_PROTOCOL(URL_string)`

Return Value

Returns the protocol portion of a URL as a Text value, or NULL if the input string is not a valid URL.

For example, in the string `http://www.workday.com`, the protocol portion is `http`.

In the string `ftp://ftp.workday.com/articles/workday.pdf`, the protocol portion is `ftp`.

Input Parameters**URL_string**

Required. A field or expression that returns a Text value in URI (uniform resource identifier) format of: *protocol:authority[/path][?query][#fragment]*

The protocol portion of a URL consists of a sequence of characters beginning with a letter and followed by any combination of letter, number, plus (+), period (.), or hyphen (-) characters, followed by a colon (:). For example: `http:`, `ftp:`, `mailto:`

Examples

Return the protocol portion of URL string values in the *referrer* field:

```
URL_PROTOCOL([referrer])
```

Return the protocol portion of the literal URL string:

```
URL_PROTOCOL("http://www.workday.com") returns http.
```

URL_QUERY

Description

URL_QUERY is a row function that returns the query portion of a URL string.

Syntax

```
URL_QUERY(URL_string)
```

Return Value

Returns the query portion of a URL as a Text value, NULL if the URL or does not contain a query, or NULL if the input string is not a valid URL.

For example, in the string `http://www.workday.com/contact.html`, the query portion is NULL.

In the string `http://workday.com/news.php?topic=press&timeframe=today#Workday%20News`, the query portion is `topic=press&timeframe=today`.

In the string `mailto:username@mycompany.com?subject=Topic`, the query portion is `subject=Topic`.

Input Parameters

URL_string

Required. A field or expression that returns a Text value in URI (uniform resource identifier) format of: *protocol:authority[/path][?query][#fragment]*.

The optional query portion of the URL is separated by a question mark (?) and typically contains an unordered list of key=value pairs separated by an ampersand (&) or semicolon (;).

Examples

Return the query portion of URL string values in the *request* field:

```
URL_QUERY([request])
```

Return the query portion of a literal URL string:

```
URL_QUERY("http://workday.com/news.php?topic=press&timeframe=today") returns  
topic=press&timeframe=today.
```

URLDECODE

Description

URLDECODE is a row function that decodes a Text value that has been encoded with the `application/x-www-form-urlencoded` media type. URL encoding, also known as percent-encoding, is a mechanism for encoding information in a Uniform Resource Identifier (URI). When sent in an HTTP GET request, `application/x-www-form-urlencoded` data is included in the query component of the request URI. When sent in an HTTP POST request, the data is placed in the body of the message, and the name of the media type is included in the message Content-Type header.

Syntax

`URLDECODE(URL_string)`

Return Value

Returns a value of type `Text` with characters decoded as follows:

- Alphanumeric characters (a-z, A-Z, 0-9) remain unchanged.
- The special characters hyphen (-), comma (,), underscore (_), period (.), and asterisk (*) remain unchanged.
- The plus sign (+) character is converted to a space character.
- The percent character (%) is interpreted as the start of a special escaped sequence, where in the sequence `%HH` , `HH` represents the hexadecimal value of the byte. Some common escape sequences:

Percent Encoding Sequence	Value
%20	space
%0A or %0D or %0D%0A	newline
%22	double quote (")
%25	percent (%)
%2D	hyphen (-)
%2E	period (.)
%3C	less than (<)
%3D	greater than (>)
%5C	backslash (\)
%7C	pipe ()

Input Parameters

URL_string

Required. A field or expression that returns a `Text` value. It is assumed that all characters in the input string are one of the following: lower-case letters (a-z), upper-case letters (A-Z), numeric digits (0-9), or the hyphen (-), comma (,), underscore (_), period (.) or asterisk (*) character. The percent character (%) is allowed, but is interpreted as the start of a special escaped sequence. The plus character (+) is allowed, but is interpreted as a space character.

Examples

Decode the values of the `url_query` field:

`URLDECODE([url_query])`

Convert a literal URL encoded string (`N%2FA%20or%20%22not%20applicable%22`) to a human-readable value:

`URLDECODE("N%2FA%20or%20%22not%20applicable%22")` returns `N/A` or `"not applicable"`.

Window Functions

AVG

Description

AVG is a window aggregate function that partitions rows into groups, orders rows by a field, and returns the average of all valid numeric values in the group. It sums all values in the group and divides by the number of valid (NOT NULL) rows. You can use AVG to calculate moving averages.

The **PARTITION BY** clause determines which fields to use to partition a set of input rows into groups. The **ORDER BY** clause determines how to order the rows in the partition.

Workday separates the input rows into groups according to the partitioning fields, orders the rows according to the ordering fields, and then computes the aggregate expression (average for this function) in each group.

Syntax

```
AVG(input_field) OVER(
  PARTITION BY partitioning_field [, partitioning_field]
  ORDER BY ordering_field [ASC | DESC] [, ordering_field [ASC | DESC]]
  RANGE BETWEEN value PRECEDING AND CURRENT ROW |
  ROWS win_boundary | BETWEEN win_boundary AND win_boundary
)
```

where *win_boundary* can be:

```
UNBOUNDED PRECEDING
value PRECEDING
UNBOUNDED FOLLOWING
value FOLLOWING
CURRENT ROW
```

Return Value

Returns a value of type `Numeric` or `Double` depending on the type of *input_field*.

Input Parameters

input_field

Required. The field on which to perform the aggregate function. You can use any numeric field or a Currency field.

OVER()

Required. OVER must be used within an AVG expression.

PARTITION BY *partitioning_field*

Required. Use the **PARTITION BY** clause to specify 1 or more fields to use to partition a group of input rows. You can specify any field type except Currency. Example: You specify the Month field as the partitioning field, so Workday groups into a single partition all records that have the same value for Month.

ORDER BY *ordering_field*

Required. Use the **ORDER BY** clause to specify how to order the input rows in the partition using the values in the specified field within each partition. You can specify any field type except Currency. However, you must use a numeric field type, such as Integer or Numeric when you use the RANGE clause.

You can use the DESC or ASC keywords to sort in descending order (high to low values, NULLs are last) or ascending order (low to high values, NULLs are first) for each ordering field. If you don't specify a sort order for an ordering field, Workday automatically sorts rows in ascending order.

ROWS | RANGE

Required. The ROWS and RANGE clauses define the specific number of rows (relative to the current row) within the partition by specifying a window frame. You define the window frame by specifying start and end points within the partition, known as window boundaries. The window frame is the set of input rows in each partition over which to calculate the aggregate expression (average for this function). The window frame can include one, several, or all rows of the partition.

Both ROWS and RANGE specify the range of rows relative to the current row, but RANGE operates logically on values (logical association) and ROWS operates physically on rows in the dataset (physical association).

RANGE limits the window frame to contain rows that have their values within the specified range, relative to the current value. ROWS limits the window frame to contain rows that are physically next to the current row.

Use RANGE to define absolute window boundaries, such as the past 3 months or year to date. When you use RANGE, the ORDER BY clause must use a numeric field type, such as Integer or Numeric.

Example: Suppose you have an Integer field called MonthNum that represents the number of the month in the year (values 1 to 12). To specify all values from the past 3 months, you would order by MonthNum and use RANGE BETWEEN 2 PRECEDING AND CURRENT ROW. This RANGE clause includes the current month and the previous 2 months, resulting in 3 months total.

Note: When you publish a dataset that contains a window function using RANGE, the number of rows in the window must be 1000 or less. If a particular window exceeds 1000 rows, the publish job fails.

win_boundary

Required. The window boundaries define the start and end points of the window frame. Window boundaries are relative to the current row.

A PRECEDING clause defines a window boundary that is lower than the current row (the number of rows to include before the current row). The FOLLOWING clause defines a window boundary that is greater than the current row (the number of rows to include after the current row).

If you specify only 1 window boundary, then Workday uses the current row as the other boundary in the window frame (either the upper or lower boundary depending on the expression syntax). The UNBOUNDED keyword includes all rows in the direction specified. When you need to specify both a start and end of a window frame, use the BETWEEN and AND keywords.

When specifying a specific number of rows, the *value* must be 100 or less.

Example: ROWS 2 PRECEDING means that the window is 3 rows in size, starting with 2 rows preceding until and including the current row.

Example: ROWS UNBOUNDED FOLLOWING means that the window starts with the current row and includes the current row and all rows that come after the current row.

Examples

You can calculate the moving average (rolling average or running average) sales for each employee:

```
AVG([Sales]) OVER(PARTITION BY [Employee] ORDER BY [SalesDate] DESC
ROWS UNBOUNDED PRECEDING)
```

You can calculate the overall average sales for every row in the partition, regardless of the fields in the ORDER BY clause:

```
AVG([Sales]) OVER(PARTITION BY [Employee] ORDER BY [SalesDate] DESC
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
```

You can calculate the rolling 12 month average:

```
AVG([fieldA]) OVER(PARTITION BY [fieldB] ORDER BY [Month] RANGE 11
PRECEDING)
```

The Month field must be a numeric field type, such as Integer or Numeric.

You can calculate the previous year to date average:

```
AVG([fieldA]) OVER(PARTITION BY [fieldB] ORDER BY [Year] RANGE 1 PRECEDING)
```

The Year field must be a numeric field type, such as Integer or Numeric.

COUNT

Description

COUNT is a window aggregate function that partitions rows into groups, orders rows by a field, and returns the total number of valid rows (NOT NULL) in the group. You can use COUNT together with other functions to calculate cumulative aggregates.

The PARTITION BY clause determines which fields to use to partition a set of input rows into groups. The ORDER BY clause determines how to order the rows in the partition.

Workday separates the input rows into groups according to the partitioning fields, orders the rows according to the ordering fields, and then computes the aggregate expression (count for this function) in each group.

Syntax

```
COUNT(input_field) OVER(
  PARTITION BY partitioning_field [, partitioning_field]
  ORDER BY ordering_field [ASC | DESC] [, ordering_field [ASC | DESC]]
  RANGE BETWEEN value PRECEDING AND CURRENT ROW |
  ROWS win_boundary | BETWEEN win_boundary AND win_boundary
)
```

where *win_boundary* can be:

```
UNBOUNDED PRECEDING
value PRECEDING
UNBOUNDED FOLLOWING
value FOLLOWING
CURRENT ROW
```

Return Value

Returns a value of type Long.

Input Parameters***input_field***

Required. The field on which to perform the aggregate function. You can use any numeric field or a Currency field.

OVER()

Required. OVER must be used within an COUNT expression.

PARTITION BY *partitioning_field*

Required. Use the PARTITION BY clause to specify 1 or more fields to use to partition a group of input rows. You can specify any field type except Currency. Example: You specify the Month field as the partitioning field, so Workday groups into a single partition all records that have the same value for Month.

ORDER BY *ordering_field*

Required. Use the ORDER BY clause to specify how to order the input rows in the partition using the values in the specified field within each partition. You can specify any field type except Currency. However, you must use a numeric field type, such as Integer or Numeric when you use the RANGE clause.

You can use the DESC or ASC keywords to sort in descending order (high to low values, NULLs are last) or ascending order (low to high values, NULLs are first) for each ordering field. If you don't specify a sort order for an ordering field, Workday automatically sorts rows in ascending order.

ROWS | RANGE

Required. The ROWS and RANGE clauses define the specific number of rows (relative to the current row) within the partition by specifying a window frame. You define the window frame by specifying start and end points within the partition, known as window boundaries. The window frame is the set of input rows in each partition over which to calculate the aggregate expression (count for this function). The window frame can include one, several, or all rows of the partition.

Both ROWS and RANGE specify the range of rows relative to the current row, but RANGE operates logically on values (logical association) and ROWS operates physically on rows in the dataset (physical association).

RANGE limits the window frame to contain rows that have their values within the specified range, relative to the current value. ROWS limits the window frame to contain rows that are physically next to the current row.

Use RANGE to define absolute window boundaries, such as the past 3 months or year to date. When you use RANGE, the ORDER BY clause must use a numeric field type, such as Integer or Numeric.

Example: Suppose you have an Integer field called MonthNum that represents the number of the month in the year (values 1 to 12). To specify all values from the past 3 months, you would order by MonthNum and use RANGE BETWEEN 2 PRECEDING AND CURRENT ROW. This RANGE clause includes the current month and the previous 2 months, resulting in 3 months total.

Note: When you publish a dataset that contains a window function using RANGE, the number of rows in the window must be 1000 or less. If a particular window exceeds 1000 rows, the publish job fails.

win_boundary

Required. The window boundaries define the start and end points of the window frame. Window boundaries are relative to the current row.

A **PRECEDING** clause defines a window boundary that is lower than the current row (the number of rows to include before the current row). The **FOLLOWING** clause defines a window boundary that is greater than the current row (the number of rows to include after the current row).

If you specify only 1 window boundary, then Workday uses the current row as the other boundary in the window frame (either the upper or lower boundary depending on the expression syntax). The **UNBOUNDED** keyword includes all rows in the direction specified. When you need to specify both a start and end of a window frame, use the **BETWEEN** and **AND** keywords.

When specifying a specific number of rows, the *value* must be 100 or less.

Example: **ROWS 2 PRECEDING** means that the window is 3 rows in size, starting with 2 rows preceding until and including the current row.

Example: **ROWS UNBOUNDED FOLLOWING** means that the window starts with the current row and includes the current row and all rows that come after the current row.

Examples

You can calculate the moving count (running count or rolling count) of sales for each employee:

```
COUNT([Sales]) OVER(PARTITION BY [Employee] ORDER BY [SalesDate] DESC
ROWS UNBOUNDED PRECEDING)
```

You can calculate the overall count of sales for every row in the partition, regardless of the fields in the **ORDER BY** clause:

```
COUNT([Sales]) OVER(PARTITION BY [Employee] ORDER BY [SalesDate] DESC
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
```

You can calculate the rolling 12 month count:

```
COUNT([fieldA]) OVER(PARTITION BY [fieldB] ORDER BY [Month] RANGE 11
PRECEDING)
```

The **Month** field must be a numeric field type, such as Integer or Numeric.

You can calculate the previous year to date count:

```
COUNT([fieldA]) OVER(PARTITION BY [fieldB] ORDER BY [Year] RANGE 1
PRECEDING)
```

The **Year** field must be a numeric field type, such as Integer or Numeric.

FIRST

Description

FIRST is a window aggregate function that partitions rows into groups, orders rows by a field, and returns the value from the first row in the group.

The **PARTITION BY** clause determines which fields to use to partition a set of input rows into groups. The **ORDER BY** clause determines how to order the rows in the partition.

Workday separates the input rows into groups according to the partitioning fields, orders the rows according to the ordering fields, and then computes the aggregate expression (first for this function) in each group.

Syntax

```
FIRST(input_field) OVER(
  PARTITION BY partitioning_field [, partitioning_field]
  ORDER BY ordering_field [ASC | DESC] [, ordering_field [ASC | DESC]]
  ROWS start_window_boundary
)
```

where *start_window_boundary* can be:

UNBOUNDED PRECEDING

Return Value

Returns a value of the same type as *input_field*.

Input Parameters

input_field

Required. The field on which to perform the aggregate function. You can use any numeric field or a Currency field.

OVER()

Required. OVER must be used within a FIRST expression.

PARTITION BY *partitioning_field*

Required. Use the PARTITION BY clause to specify 1 or more fields to use to partition a group of input rows. You can specify any field type except Currency. Example: You specify the Month field as the partitioning field, so Workday groups into a single partition all records that have the same value for Month.

ORDER BY *ordering_field*

Required. Use the ORDER BY clause to specify how to order the input rows in the partition using the values in the specified field within each partition. You can specify any field type except Currency.

You can use the DESC or ASC keywords to sort in descending order (high to low values, NULLs are last) or ascending order (low to high values, NULLs are first) for each ordering field. If you don't specify a sort order for an ordering field, Workday automatically sorts rows in ascending order.

ROWS

Required. The ROWS clause defines the specific number of rows (relative to the current row) within the partition by specifying a window frame. You define the window frame by specifying start and end points within the partition, known as window boundaries. The window frame is the set of input rows in each partition, relative to the current row, over which to calculate the aggregate expression (first for this function). The window frame can include one, several, or all rows of the partition.

window_boundary

Required. The window boundaries define the start and end points of the window frame. Window boundaries are relative to the current row.

A PRECEDING clause defines a window boundary that is lower than the current row (the number of rows to include before the current row). If you specify only 1 window

boundary, then Workday uses the current row as the last row in the window frame (the upper boundary). The UNBOUNDED keyword includes all rows in the direction specified.

LAG

Description

LAG is a window aggregate function that partitions rows into groups, orders rows by a field, and returns the value of a field in the row at the specified offset before (above) the current row in the group.

The PARTITION BY clause determines which fields to use to partition a set of input rows into groups. The ORDER BY clause determines how to order the rows in the partition.

Workday separates the input rows into groups according to the partitioning fields, orders the rows according to the ordering fields, and then computes the aggregate expression (lag for this function) in each group.

Syntax

```
LAG(input_field, offset, default_value) OVER(
  PARTITION BY partitioning_field [, partitioning_field]
  ORDER BY ordering_field [ASC | DESC] [, ordering_field [ASC | DESC]]
)
```

Return Value

Returns one value per row of the same type as the *input_field*.

Input Parameters

input_field

Required. The field on which to perform the aggregate function. You can specify any field type.

offset

Optional. The number of rows before the current row whose value to return. Must be a literal number greater than or equal to zero (0) and less than or equal to 100. If you don't specify the offset, Workday uses the value of 1.

default_value

Optional. The value this function returns when the offset row is outside the currently defined window or when the value in the offset row is NULL. *default_value* must be the same type as *input_field*. If you don't specify a default value, Workday uses the value of NULL.

OVER()

Required. OVER must be used within a LAG expression.

PARTITION BY *partitioning_field*

Required. Use the PARTITION BY clause to specify 1 or more fields to use to partition a group of input rows. You can specify any field type except Currency. Example: You specify the Month field as the partitioning field, so Workday groups into a single partition all records that have the same value for Month.

ORDER BY *ordering_field*

Required. Use the ORDER BY clause to specify how to order the input rows in the partition using the values in the specified field within each partition. You can specify any field type except Currency.

You can use the DESC or ASC keywords to sort in descending order (high to low values, NULLs are last) or ascending order (low to high values, NULLs are first) for each ordering field. If you don't specify a sort order for an ordering field, Workday automatically sorts rows in ascending order.

Examples

Example: You have a dataset with these rows and fields.

row_ID	Employee_Name	Eff_Date	Salary
1	Goh	1/1/18	49000
2	Goh	1/1/19	56000
3	Goh	1/1/17	44000
4	Freeman	1/1/18	65000
5	Freeman	1/1/17	57000
6	Freeman	1/1/19	69000
7	Smith	1/1/18	51000
8	Smith	1/1/19	56000
9	Smith	1/1/16	44000

You can order the rows for each employee in ascending (ASC) order by the effective date (*Eff_Date*) field, so the most recent salary comes first in each partition.

Use this expression in the *Salary_Increase* field to calculate the change in salary between each change in effective date:

```
[Salary] - (LAG([Salary], 1, [Salary]) OVER(
  PARTITION BY [Employee_Name]
  ORDER BY [Eff_Date] ASC)
)
```

You get these results:

row_ID	Employee_Name	Eff_Date	Salary	Salary_Increase
2	Goh	1/1/19	56000	7000
1	Goh	1/1/18	49000	5000
3	Goh	1/1/17	44000	0
6	Freeman	1/1/19	69000	4000
4	Freeman	1/1/18	65000	8000
5	Freeman	1/1/17	57000	0
8	Smith	1/1/19	56000	5000
7	Smith	1/1/18	51000	7000
9	Smith	1/1/16	44000	0

Related Information**Reference**

[Workday 32 What's New Post: Prism Analytics Data Preparation](#)

LAST**Description**

LAST is a window aggregate function that partitions rows into groups, orders rows by a field, and returns the value from the last row in the group.

The **PARTITION BY** clause determines which fields to use to partition a set of input rows into groups. The **ORDER BY** clause determines how to order the rows in the partition.

Workday separates the input rows into groups according to the partitioning fields, orders the rows according to the ordering fields, and then computes the aggregate expression (last for this function) in each group.

Syntax

```
LAST(input_field) OVER(
  PARTITION BY partitioning_field [, partitioning_field]
  ORDER BY ordering_field [ASC | DESC] [, ordering_field [ASC | DESC]]
  ROWS BETWEEN start_window_boundary AND end_window_boundary
)
```

where *start_window_boundary* can be:

CURRENT ROW

and where *end_window_boundary* can be:

UNBOUNDED FOLLOWING

Return Value

Returns a value of the same type as *input_field*.

Input Parameters***input_field***

Required. The field on which to perform the aggregate function. You can use any numeric field or a Currency field.

OVER()

Required. OVER must be used within a LAST expression.

PARTITION BY *partitioning_field*

Required. Use the **PARTITION BY** clause to specify 1 or more fields to use to partition a group of input rows. You can specify any field type except Currency. Example: You specify the Month field as the partitioning field, so Workday groups into a single partition all records that have the same value for Month.

ORDER BY *ordering_field*

Required. Use the **ORDER BY** clause to specify how to order the input rows in the partition using the values in the specified field within each partition. You can specify any field type except Currency.

You can use the DESC or ASC keywords to sort in descending order (high to low values, NULLs are last) or ascending order (low to high values, NULLs are first) for each ordering

field. If you don't specify a sort order for an ordering field, Workday automatically sorts rows in ascending order.

ROWS

Required. The ROWS clause defines the specific number of rows (relative to the current row) within the partition by specifying a window frame. You define the window frame by specifying start and end points within the partition, known as window boundaries. The window frame is the set of input rows in each partition, relative to the current row, over which to calculate the aggregate expression (last for this function). The window frame can include one, several, or all rows of the partition.

window_boundary

Required. The window boundaries define the start and end points of the window frame. Window boundaries are relative to the current row.

A FOLLOWING clause defines a window boundary that is after the current row (the number of rows to include after the current row). Workday uses the current row as the first row in the window frame (the lower boundary). The UNBOUNDED keyword includes all rows in the direction specified.

LEAD

Description

LEAD is a window aggregate function that partitions rows into groups, orders rows by a field, and returns the value of a field in the row at the specified offset after (below) the current row in the group.

The PARTITION BY clause determines which fields to use to partition a set of input rows into groups. The ORDER BY clause determines how to order the rows in the partition.

Workday separates the input rows into groups according to the partitioning fields, orders the rows according to the ordering fields, and then computes the aggregate expression (lead for this function) in each group.

Syntax

```
LEAD(input_field, offset, default_value) OVER(
  PARTITION BY partitioning_field [, partitioning_field]
  ORDER BY ordering_field [ASC | DESC] [, ordering_field [ASC | DESC]]
)
```

Return Value

Returns one value per row of the same type as the *input_field*.

Input Parameters

input_field

Required. The field on which to perform the aggregate function. You can specify any field type.

offset

Optional. The number of rows after the current row whose value to return. Must be a literal number greater than or equal to zero (0) and less than or equal to 100. If you don't specify the offset, Workday uses the value of 1.

default_value

Optional. The value this function returns when the offset row is outside the currently defined window or when the value in the offset row is NULL. *default_value* must be the same type as *input_field*. If you don't specify a default value, Workday uses the value of NULL.

OVER()

Required. OVER must be used within a LEAD expression.

PARTITION BY *partitioning_field*

Required. Use the PARTITION BY clause to specify 1 or more fields to use to partition a group of input rows. You can specify any field type except Currency. Example: You specify the Month field as the partitioning field, so Workday groups into a single partition all records that have the same value for Month.

ORDER BY *ordering_field*

Required. Use the ORDER BY clause to specify how to order the input rows in the partition using the values in the specified field within each partition. You can specify any field type except Currency.

You can use the DESC or ASC keywords to sort in descending order (high to low values, NULLs are last) or ascending order (low to high values, NULLs are first) for each ordering field. If you don't specify a sort order for an ordering field, Workday automatically sorts rows in ascending order.

Examples

Example: You have a dataset with these rows and fields.

row_ID	Employee_Name	Eff_Date	Salary
1	Goh	1/1/18	49000
2	Goh	1/1/19	56000
3	Goh	1/1/17	44000
4	Freeman	1/1/18	65000
5	Freeman	1/1/17	57000
6	Freeman	1/1/19	69000
7	Smith	1/1/18	51000
8	Smith	1/1/19	56000
9	Smith	1/1/16	44000

You can order the rows for each employee in descending (*DESC*) order by the effective date (*Eff_Date*) field, so the most recent salary comes first in each partition.

Use this expression in the *Salary_Increase* field to calculate the change in salary between each change in effective date:

```
[Salary] - (LEAD([Salary], 1, [Salary]) OVER(
  PARTITION BY [Employee_Name]
  ORDER BY [Eff_Date] DESC)
)
```

You get these results:

row_ID	Employee_Name	Eff_Date	Salary	Salary_Increase
2	Goh	1/1/19	56000	7000
1	Goh	1/1/18	49000	5000
3	Goh	1/1/17	44000	0
6	Freeman	1/1/19	69000	4000
4	Freeman	1/1/18	65000	8000
5	Freeman	1/1/17	57000	0
8	Smith	1/1/19	56000	5000
7	Smith	1/1/18	51000	7000
9	Smith	1/1/16	44000	0

Related Information

Reference

[Workday 32 What's New Post: Prism Analytics Data Preparation](#)

MAX

Description

MAX is a window aggregate function that partitions rows into groups, orders rows by a field, and returns the maximum (highest) value in the group.

The PARTITION BY clause determines which fields to use to partition a set of input rows into groups. The ORDER BY clause determines how to order the rows in the partition.

Workday separates the input rows into groups according to the partitioning fields, orders the rows according to the ordering fields, and then computes the aggregate expression (maximum for this function) in each group.

Syntax

```
MAX(input_field) OVER(
  PARTITION BY partitioning_field [, partitioning_field]
  ORDER BY ordering_field [ASC | DESC] [, ordering_field [ASC | DESC]]
  RANGE BETWEEN value PRECEDING AND CURRENT ROW |
  ROWS win_boundary | BETWEEN win_boundary AND win_boundary
)
```

where *win_boundary* can be:

```
UNBOUNDED PRECEDING
value PRECEDING
UNBOUNDED FOLLOWING
value FOLLOWING
CURRENT ROW
```

Return Value

Returns a value of type `Numeric` or `Double` depending on the type of *input_field*.

Input Parameters

input_field

Required. The field on which to perform the aggregate function. You can use any numeric field or a Currency field.

OVER()

Required. OVER must be used within a MAX expression.

PARTITION BY *partitioning_field*

Required. Use the PARTITION BY clause to specify 1 or more fields to use to partition a group of input rows. You can specify any field type except Currency. Example: You specify the Month field as the partitioning field, so Workday groups into a single partition all records that have the same value for Month.

ORDER BY *ordering_field*

Required. Use the ORDER BY clause to specify how to order the input rows in the partition using the values in the specified field within each partition. You can specify any field type except Currency. However, you must use a numeric field type, such as Integer or Numeric when you use the RANGE clause.

You can use the DESC or ASC keywords to sort in descending order (high to low values, NULLs are last) or ascending order (low to high values, NULLs are first) for each ordering field. If you don't specify a sort order for an ordering field, Workday automatically sorts rows in ascending order.

ROWS | RANGE

Required. The ROWS and RANGE clauses define the specific number of rows (relative to the current row) within the partition by specifying a window frame. You define the window frame by specifying start and end points within the partition, known as window boundaries. The window frame is the set of input rows in each partition over which to calculate the aggregate expression (maximum for this function). The window frame can include one, several, or all rows of the partition.

Both ROWS and RANGE specify the range of rows relative to the current row, but RANGE operates logically on values (logical association) and ROWS operates physically on rows in the dataset (physical association).

RANGE limits the window frame to contain rows that have their values within the specified range, relative to the current value. ROWS limits the window frame to contain rows that are physically next to the current row.

Use RANGE to define absolute window boundaries, such as the past 3 months or year to date. When you use RANGE, the ORDER BY clause must use a numeric field type, such as Integer or Numeric.

Example: Suppose you have an Integer field called MonthNum that represents the number of the month in the year (values 1 to 12). To specify all values from the past 3 months, you would order by MonthNum and use RANGE BETWEEN 2 PRECEDING AND CURRENT ROW. This RANGE clause includes the current month and the previous 2 months, resulting in 3 months total.

Note: When you publish a dataset that contains a window function using RANGE, the number of rows in the window must be 1000 or less. If a particular window exceeds 1000 rows, the publish job fails.

win_boundary

Required. The window boundaries define the start and end points of the window frame. Window boundaries are relative to the current row.

A PRECEDING clause defines a window boundary that is lower than the current row (the number of rows to include before the current row). The FOLLOWING clause defines a window boundary that is greater than the current row (the number of rows to include after the current row).

If you specify only 1 window boundary, then Workday uses the current row as the other boundary in the window frame (either the upper or lower boundary depending on the expression syntax). The **UNBOUNDED** keyword includes all rows in the direction specified. When you need to specify both a start and end of a window frame, use the **BETWEEN** and **AND** keywords.

When specifying a specific number of rows, the *value* must be 100 or less.

Example: **ROWS 2 PRECEDING** means that the window is 3 rows in size, starting with 2 rows preceding until and including the current row.

Example: **ROWS UNBOUNDED FOLLOWING** means that the window starts with the current row and includes the current row and all rows that come after the current row.

Examples

Example: You have a dataset with these rows and fields.

Supervisory Org	Quarter	Name	Comp Change
Marketing	2019-Q1	Goh	2000.00
Marketing	2019-Q1	Freeman	1000.00
Marketing	2019-Q1	Smith	2500.00
Consulting	2019-Q1	Gomez	5000.00
Consulting	2019-Q1	Kimura	3000.00
Consulting	2019-Q1	Fitz	3500.00
Consulting	2019-Q2	Gomez	0
Consulting	2019-Q2	Kimura	2000.00
Consulting	2019-Q2	Fitz	1500.00

You can calculate the highest change in compensation (*Comp Change* field) for each supervisory org in each quarter.

To ensure that Workday returns the same value for every row in a partition, order the rows in descending (*DESC*) order by the same field as the input field so the highest compensation change comes first in each partition.

Use this expression in the *Max Comp Change* field:

```
MAX([Comp Change]) OVER(
  PARTITION BY [Supervisory Org], [Quarter]
  ORDER BY [Comp Change] DESC
  ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
)
```

You get these results:

Supervisory Org	Quarter	Name	Comp Change	Max Comp Change
Consulting	2019-Q1	Gomez	5000.00	5000.00
Consulting	2019-Q1	Fitz	3500.00	5000.00
Consulting	2019-Q1	Kimura	3000.00	5000.00

Supervisory Org	Quarter	Name	Comp Change	Max Comp Change
Consulting	2019-Q2	Kimura	2000.00	2000.00
Consulting	2019-Q2	Fitz	1500.00	2000.00
Consulting	2019-Q2	Gomez	0	2000.00
Marketing	2019-Q1	Smith	2500.00	2500.00
Marketing	2019-Q1	Goh	2000.00	2500.00
Marketing	2019-Q1	Freeman	1000.00	2500.00

Related Information

Reference

[Workday 32 What's New Post: Prism Analytics Dataset Window Functions](#)

MIN

Description

MIN is a window aggregate function that partitions rows into groups, orders rows by a field, and returns the minimum (lowest) value in the group.

The **PARTITION BY** clause determines which fields to use to partition a set of input rows into groups. The **ORDER BY** clause determines how to order the rows in the partition.

Workday separates the input rows into groups according to the partitioning fields, orders the rows according to the ordering fields, and then computes the aggregate expression (minimum for this function) in each group.

Syntax

```
MIN(input_field) OVER(
  PARTITION BY partitioning_field [, partitioning_field]
  ORDER BY ordering_field [ASC | DESC] [, ordering_field [ASC | DESC]]
  RANGE BETWEEN value PRECEDING AND CURRENT ROW |
  ROWS win_boundary | BETWEEN win_boundary AND win_boundary
)
```

where *win_boundary* can be:

```
UNBOUNDED PRECEDING
value PRECEDING
UNBOUNDED FOLLOWING
value FOLLOWING
CURRENT ROW
```

Return Value

Returns a value of type `Numeric` or `Double` depending on the type of *input_field*.

Input Parameters

input_field

Required. The field on which to perform the aggregate function. You can use any numeric field or a Currency field.

OVER()

Required. OVER must be used within a MIN expression.

PARTITION BY *partitioning_field*

Required. Use the PARTITION BY clause to specify 1 or more fields to use to partition a group of input rows. You can specify any field type except Currency. Example: You specify the Month field as the partitioning field, so Workday groups into a single partition all records that have the same value for Month.

ORDER BY *ordering_field*

Required. Use the ORDER BY clause to specify how to order the input rows in the partition using the values in the specified field within each partition. You can specify any field type except Currency. However, you must use a numeric field type, such as Integer or Numeric when you use the RANGE clause.

You can use the DESC or ASC keywords to sort in descending order (high to low values, NULLs are last) or ascending order (low to high values, NULLs are first) for each ordering field. If you don't specify a sort order for an ordering field, Workday automatically sorts rows in ascending order.

ROWS | RANGE

Required. The ROWS and RANGE clauses define the specific number of rows (relative to the current row) within the partition by specifying a window frame. You define the window frame by specifying start and end points within the partition, known as window boundaries. The window frame is the set of input rows in each partition over which to calculate the aggregate expression (minimum for this function). The window frame can include one, several, or all rows of the partition.

Both ROWS and RANGE specify the range of rows relative to the current row, but RANGE operates logically on values (logical association) and ROWS operates physically on rows in the dataset (physical association).

RANGE limits the window frame to contain rows that have their values within the specified range, relative to the current value. ROWS limits the window frame to contain rows that are physically next to the current row.

Use RANGE to define absolute window boundaries, such as the past 3 months or year to date. When you use RANGE, the ORDER BY clause must use a numeric field type, such as Integer or Numeric.

Example: Suppose you have an Integer field called MonthNum that represents the number of the month in the year (values 1 to 12). To specify all values from the past 3 months, you would order by MonthNum and use RANGE BETWEEN 2 PRECEDING AND CURRENT ROW. This RANGE clause includes the current month and the previous 2 months, resulting in 3 months total.

Note: When you publish a dataset that contains a window function using RANGE, the number of rows in the window must be 1000 or less. If a particular window exceeds 1000 rows, the publish job fails.

win_boundary

Required. The window boundaries define the start and end points of the window frame. Window boundaries are relative to the current row.

A PRECEDING clause defines a window boundary that is lower than the current row (the number of rows to include before the current row). The FOLLOWING clause defines a window boundary that is greater than the current row (the number of rows to include after the current row).

If you specify only 1 window boundary, then Workday uses the current row as the other boundary in the window frame (either the upper or lower boundary depending on the expression syntax). The UNBOUNDED keyword includes all rows in the direction specified.

When you need to specify both a start and end of a window frame, use the **BETWEEN** and **AND** keywords.

When specifying a specific number of rows, the *value* must be 100 or less.

Example: **ROWS 2 PRECEDING** means that the window is 3 rows in size, starting with 2 rows preceding until and including the current row.

Example: **ROWS UNBOUNDED FOLLOWING** means that the window starts with the current row and includes the current row and all rows that come after the current row.

Examples

Example: You have a dataset with these rows and fields.

Supervisory Org	Quarter	Name	Comp Change
Marketing	2019-Q1	Goh	2000.00
Marketing	2019-Q1	Freeman	1000.00
Marketing	2019-Q1	Smith	2500.00
Consulting	2019-Q1	Gomez	5000.00
Consulting	2019-Q1	Kimura	3000.00
Consulting	2019-Q1	Fitz	3500.00
Consulting	2019-Q2	Gomez	0
Consulting	2019-Q2	Kimura	2000.00
Consulting	2019-Q2	Fitz	1500.00

You can calculate the lowest change in compensation (*Comp Change* field) for each supervisory org in each quarter.

To ensure that Workday returns the same value for every row in a partition, order the rows in ascending (ASC) order by the same field as the input field so the lowest compensation change comes first in each partition.

Use this expression in the *Min Comp Change* field:

```
MIN([Comp Change]) OVER(
  PARTITION BY [Supervisory Org], [Quarter]
  ORDER BY [Comp Change] ASC
  ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
)
```

You get these results:

Supervisory Org	Quarter	Name	Comp Change	Min Comp Change
Consulting	2019-Q1	Kimura	3000.00	3000.00
Consulting	2019-Q1	Fitz	3500.00	3000.00
Consulting	2019-Q1	Gomez	5000.00	3000.00
Consulting	2019-Q2	Gomez	0	0
Consulting	2019-Q2	Fitz	1500.00	0

Supervisory Org	Quarter	Name	Comp Change	Min Comp Change
Consulting	2019-Q2	Kimura	2000.00	0
Marketing	2019-Q1	Freeman	1000.00	1000.00
Marketing	2019-Q1	Goh	2000.00	1000.00
Marketing	2019-Q1	Smith	2500.00	1000.00

Related Information

Reference

[Workday 32 What's New Post: Prism Analytics Dataset Window Functions](#)

RANK

Description

RANK is a window aggregate function used to assign a ranking number to each row in a group. If multiple rows have the same ranking value (there's a tie), then Workday assigns the same rank value to the tied rows and skips the subsequent rank position.

The PARTITION BY clause determines which fields to use to partition a set of input rows into groups.

The ORDER BY clause determines how to order the rows in the partition before they're ranked.

Workday separates the input rows into groups according to the partitioning fields, orders the rows according to the ordering fields, and then computes the aggregate expression (rank for this function) in each group. The ranked rows in each group start at 1.

Syntax

```
RANK() OVER(
  PARTITION BY partitioning_field [, partitioning_field]
  ORDER BY ordering_field [ASC | DESC] [, ordering_field [ASC | DESC]]
)
```

Return Value

Returns a value of type Integer.

Input Parameters

OVER()

Required. OVER must be used within a RANK expression.

PARTITION BY *partitioning_field*

Required. Use the PARTITION BY clause to specify 1 or more fields to use to partition a group of input rows. You can specify any field type except Currency. Example: You specify the Month field as the partitioning field, so Workday groups together into a single partition all records that have the same value for Month.

ORDER BY *ordering_field*

Required. Use the ORDER BY clause to specify how to order the input rows in the partition using the values in the specified field within each partition. You can specify any field type except Currency.

You can use the DESC or ASC keywords to sort in descending order (high to low values, NULLs are last) or ascending order (low to high values, NULLs are first) for each ordering

field. If you don't specify a sort order for an ordering field, Workday automatically sorts rows in ascending order.

Examples

Example: You have a dataset with these rows and fields.

Employee	Sales Date	Sales
Goh	12/31/2018	140
Goh	11/30/2018	60
Goh	10/31/2018	140
Freeman	12/31/2018	160
Freeman	11/30/2018	60
Freeman	10/31/2018	110
Smith	12/31/2018	140
Smith	11/30/2018	60
Smith	10/31/2018	120

You can rank the sales for each employee in descending order, so the highest sales is given the ranking of 1. Use this expression in the *Rank Sales by Employee* field:

```
RANK() OVER(PARTITION BY [Employee] ORDER BY [Sales] DESC)
```

You get these results:

Employee	Sales Date	Sales	Rank Sales by Employee
Goh	12/31/2018	140	1
Goh	10/31/2018	140	1
Goh	11/30/2018	60	3
Freeman	12/31/2018	160	1
Freeman	10/31/2018	110	2
Freeman	11/30/2018	60	3
Smith	12/31/2018	140	1
Smith	10/31/2018	120	2
Smith	11/30/2018	60	3

You can also rank the sales for each date in descending order, so the highest sales is given the ranking of 1. Use this expression in the *Rank Sales by Date* field:

```
RANK() OVER(PARTITION BY [Sales Date] ORDER BY [Sales] DESC)
```

You get these results:

Employee	Sales Date	Sales	Rank Sales by Date
Freeman	12/31/2018	160	1
Goh	12/31/2018	140	2
Smith	12/31/2018	140	2
Goh	11/30/2018	60	1
Freeman	11/30/2018	60	1
Smith	11/30/2018	60	1
Goh	10/31/2018	140	1
Smith	10/31/2018	120	2
Freeman	10/31/2018	110	3

Notice that tied values are given the same rank number and the following rank position is skipped.

ROW_NUMBER

Description

ROW_NUMBER is a window aggregate function that partitions rows into groups, orders rows by a field, and assigns a unique, sequential number to each row in a group, starting at 1 for the first row in each group. ROW_NUMBER always assigns a unique value to each row in a group. You might want to use ROW_NUMBER to create a unique ID for each row in your dataset.

The PARTITION BY clause determines which fields to use to partition a set of input rows into groups.

The ORDER BY clause determines how to order the rows in the partition before they're assigned a sequential number.

Workday separates the input rows into groups according to the partitioning fields, orders the rows according to the ordering fields, and then computes the aggregate expression (row numbering for this function) in each group. The numbered rows in each group start at 1.

Syntax

```
ROW_NUMBER() OVER(
  PARTITION BY partitioning_field [, partitioning_field]
  ORDER BY ordering_field [ASC | DESC] [, ordering_field [ASC | DESC]]
)
```

Return Value

Returns a value of type Integer.

Input Parameters

OVER()

Required. OVER must be used within a ROW_NUMBER expression.

PARTITION BY *partitioning_field*

Required. Use the PARTITION BY clause to specify 1 or more fields to use to partition a group of input rows. You can specify any field type except Currency. Example: You specify the Month field as the partitioning field, so Workday groups together into a single partition all records that have the same value for Month.

ORDER BY *ordering_field*

Required. Use the ORDER BY clause to specify how to order the input rows in the partition using the values in the specified field within each partition. You can specify any field type except Currency.

You can use the DESC or ASC keywords to sort in descending order (high to low values, NULLs are last) or ascending order (low to high values, NULLs are first) for each ordering field. If you don't specify a sort order for an ordering field, Workday automatically sorts rows in ascending order.

Examples

Example: You have a dataset with these rows and fields.

Employee	Sales Date	Sales
Goh	12/31/2018	140
Goh	11/30/2018	60
Goh	10/31/2018	140
Freeman	12/31/2018	160
Freeman	11/30/2018	60
Freeman	10/31/2018	110
Smith	12/31/2018	140
Smith	11/30/2018	60
Smith	10/31/2018	120

You can assign a unique ID to the sales of each employee in descending order, so the highest sales is given the ranking of 1. Use this expression in the *Sales Num by Employee* field:

```
ROW_NUMBER() OVER(PARTITION BY [Employee] ORDER BY [Sales] DESC)
```

You get these results:

Employee	Sales Date	Sales	Sales Num by Employee
Goh	12/31/2018	140	1
Goh	10/31/2018	140	2
Goh	11/30/2018	60	3
Freeman	12/31/2018	160	1
Freeman	10/31/2018	110	2
Freeman	11/30/2018	60	3
Smith	12/31/2018	140	1
Smith	10/31/2018	120	2
Smith	11/30/2018	60	3

You can also assign a unique ID to the sales for each date in descending order, so the highest sales is given the ranking of 1. Use this expression in the *Sales Num by Date* field:

```
ROW_NUMBER() OVER(PARTITION BY [Sales Date] ORDER BY [Sales] DESC)
```

You get these results:

Employee	Sales Date	Sales	Sales Num by Date
Freeman	12/31/2018	160	1
Goh	12/31/2018	140	2
Smith	12/31/2018	140	3
Goh	11/30/2018	60	1
Freeman	11/30/2018	60	2
Smith	11/30/2018	60	3
Goh	10/31/2018	140	1
Smith	10/31/2018	120	2
Freeman	10/31/2018	110	3

You can also use ROW_NUMBER to determine the latest version of every row in a dataset that contains multiple rows per ID. In this scenario, the dataset requires a date field that represents when the information in that row became current. If you're familiar with data warehousing concepts, this is a type 2 slowly changing dimension table. You have a dataset with these rows and fields.

ID	Name	Region	Effective Date
G1	Gorman	West	01/01/2015
H1	Harris	West	01/01/2015
G1	Gorman	East	09/01/2016
H1	Harris	East	01/03/2017
H1	Harris	National	01/20/2021

To assign the value of 1 to the latest version of each ID, use this expression in the *Latest Version* field:

```
ROW_NUMBER() OVER(PARTITION BY [ID] ORDER BY [Effective Date] DESC)
```

You get these results:

ID	Name	Region	Effective Date	Latest Version
G1	Gorman	West	01/01/2015	2
H1	Harris	West	01/01/2015	3
G1	Gorman	East	09/01/2016	1
H1	Harris	East	01/03/2017	2
H1	Harris	National	01/20/2021	1

You can filter on *Latest Version* using a Filter Stage in order to return only the latest row for each ID.

SUM

Description

SUM is a window aggregate function that partitions rows into groups, orders rows by a field, and returns the total of all values in the group. You can use SUM to calculate running totals.

The PARTITION BY clause determines which fields to use to partition a set of input rows into groups. The ORDER BY clause determines how to order the rows in the partition.

Workday separates the input rows into groups according to the partitioning fields, orders the rows according to the ordering fields, and then computes the aggregate expression (sum for this function) in each group.

Syntax

```
SUM(input_field) OVER(
  PARTITION BY partitioning_field [, partitioning_field]
  ORDER BY ordering_field [ASC | DESC] [, ordering_field [ASC | DESC]]
  RANGE BETWEEN value PRECEDING AND CURRENT ROW |
  ROWS win_boundary | BETWEEN win_boundary AND win_boundary
)
```

where *win_boundary* can be:

```
UNBOUNDED PRECEDING
value PRECEDING
UNBOUNDED FOLLOWING
value FOLLOWING
CURRENT ROW
```

Return Value

Returns a value of type Numeric, Long, or Double depending on the type of *input_field*.

Input Parameters

input_field

Required. The field on which to perform the aggregate function. You can use any numeric field or a Currency field.

OVER()

Required. OVER must be used within an SUM expression.

PARTITION BY *partitioning_field*

Required. Use the PARTITION BY clause to specify 1 or more fields to use to partition a group of input rows. You can specify any field type except Currency. Example: You specify the Month field as the partitioning field, so Workday groups into a single partition all records that have the same value for Month.

ORDER BY *ordering_field*

Required. Use the ORDER BY clause to specify how to order the input rows in the partition using the values in the specified field within each partition. You can specify any field type except Currency. However, you must use a numeric field type, such as Integer or Numeric when you use the RANGE clause.

You can use the DESC or ASC keywords to sort in descending order (high to low values, NULLs are last) or ascending order (low to high values, NULLs are first) for each ordering field. If you don't specify a sort order for an ordering field, Workday automatically sorts rows in ascending order.

ROWS | RANGE

Required. The ROWS and RANGE clauses define the specific number of rows (relative to the current row) within the partition by specifying a window frame. You define the window frame by specifying start and end points within the partition, known as window boundaries. The window frame is the set of input rows in each partition over which to calculate the aggregate expression (sum for this function). The window frame can include one, several, or all rows of the partition.

Both ROWS and RANGE specify the range of rows relative to the current row, but RANGE operates logically on values (logical association) and ROWS operates physically on rows in the dataset (physical association).

RANGE limits the window frame to contain rows that have their values within the specified range, relative to the current value. ROWS limits the window frame to contain rows that are physically next to the current row.

Use RANGE to define absolute window boundaries, such as the past 3 months or year to date. When you use RANGE, the ORDER BY clause must use a numeric field type, such as Integer or Numeric.

Example: Suppose you have an Integer field called MonthNum that represents the number of the month in the year (values 1 to 12). To specify all values from the past 3 months, you would order by MonthNum and use RANGE BETWEEN 2 PRECEDING AND CURRENT ROW. This RANGE clause includes the current month and the previous 2 months, resulting in 3 months total.

Note: When you publish a dataset that contains a window function using RANGE, the number of rows in the window must be 1000 or less. If a particular window exceeds 1000 rows, the publish job fails.

win_boundary

Required. The window boundaries define the start and end points of the window frame. Window boundaries are relative to the current row.

A PRECEDING clause defines a window boundary that is lower than the current row (the number of rows to include before the current row). The FOLLOWING clause defines a window boundary that is greater than the current row (the number of rows to include after the current row).

If you specify only 1 window boundary, then Workday uses the current row as the other boundary in the window frame (either the upper or lower boundary depending on the expression syntax). The UNBOUNDED keyword includes all rows in the direction specified. When you need to specify both a start and end of a window frame, use the BETWEEN and AND keywords.

When specifying a specific number of rows, the *value* must be 100 or less.

Example: ROWS 2 PRECEDING means that the window is 3 rows in size, starting with 2 rows preceding until and including the current row.

Example: ROWS UNBOUNDED FOLLOWING means that the window starts with the current row and includes the current row and all rows that come after the current row.

Examples

You can calculate the running total (rolling sum or moving sum) of sales for each employee:

```
SUM([Sales]) OVER(PARTITION BY [Employee] ORDER BY [Sales] DESC
ROWS UNBOUNDED PRECEDING)
```

You can calculate the overall sum (total sales) for every row in the partition, regardless of the fields in the ORDER BY clause:

```
SUM([Sales]) OVER(PARTITION BY [Employee] ORDER BY [Sales] DESC
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
```

You can calculate the rolling 12 month sum:

```
SUM([fieldA]) OVER(PARTITION BY [fieldB] ORDER BY [Month] RANGE 11
PRECEDING)
```

The *Month* field must be a numeric field type, such as Integer or Numeric.

You can calculate the previous year to date sum:

```
SUM([fieldA]) OVER(PARTITION BY [fieldB] ORDER BY [Year] RANGE 1 PRECEDING)
```

The *Year* field must be a numeric field type, such as Integer or Numeric.

Regular Expression Reference

Concept: Regular Expressions in Prism

Regular expressions vary in complexity using a combination of basic constructs to describe a string matching pattern. This reference describes the most common regular expression matching patterns, but is not a comprehensive list.

Regular expressions, also referred to as regex or regexp, are a standardized collection of special characters and constructs used for matching strings of text. They provide a flexible and precise language for matching particular characters, words, or patterns of characters.

Prism Analytics regular expressions are based on the pattern matching syntax of the Java programming language. For more in depth information on writing valid regular expressions, refer to the [Java regular expression pattern documentation](#).

You can use regular expressions in Prism calculated field expressions that use either the REGEX or REGEX_REPLACE functions.

Regex Literal and Special Characters

This section describes the regular expression syntax for referring to literal characters, special characters, nonprintable characters (such as a tab or a newline), and special character escaping.

Literal Characters

The most basic form of pattern matching is the match of literal characters. If the regular expression is foo and the input string is foo, the match will succeed because the strings are identical.

Special Characters

Certain characters are reserved for special use in regular expressions. These special characters are called metacharacters. If you want to use special characters as literal characters, you must escape them.

Character Name	Character	Reserved For
opening bracket	[	Start of a character class
closing bracket	]	End of a character class

Character Name	Character	Reserved For
hyphen	-	Character ranges within a character class
backslash	\	General escape character
caret	^	Beginning of string, negating of a character class
dollar sign	\$	End of string
period	.	Matching any single character
pipe		Alternation (OR) operator
question mark	?	Optional quantifier, quantifier minimizer
asterisk	*	Zero or more quantifier
plus sign	+	Once or more quantifier
opening parenthesis	(	Start of a subexpression group
closing parenthesis	)	End of a subexpression group
opening brace	{	Start of min/max quantifier
closing brace	}	End of min/max quantifier

Escaping Special Characters

You can use these methods to treat a special character as a literal (ordinary) character:

- Precede the special character with a \ (backslash character). Example: to specify an asterisk as a literal character instead of a quantifier, use *.
- Enclose the special characters within \Q (starting quote) and \E (ending quote). Everything between \Q and \E is then treated as literal characters.
- To escape literal double-quotes in a REGEX() expression, double the double-quotes (""). Example: to extract the inches portion from a height field where example values are 6'2", 5'11":

```
REGEX([height], "\'(\d)+\"")
```

NonPrinting Characters

You can use special character sequence constructs to specify nonprintable characters in a regular expression. Some of the most commonly used constructs are:

Construct	Matches
\n	newline character
\r	carriage return character
\t	tab character
\f	form feed character

Regex Character Classes

Character Class Constructs

A character class allows you to specify a set of characters, enclosed in square brackets, that can produce a single character match. A character class matches to a single character only. For example, gr[ae]y

will match to gray or grey, but not to graay or graey. The order of the characters inside the brackets doesn't matter.

You can use a hyphen inside a character class to specify a range of characters. Example: `[a-z]` matches a single lower-case letter between a and z. You can also use more than 1 range, or a combination of ranges and single characters. Example: `[0-9X]` matches a numeric digit or the letter X. The order of the characters and the ranges doesn't matter.

A caret following an opening bracket specifies characters to exclude from a match. For example, `[^abc]` matches any character except a, b, or c.

Construct	Type	Description
<code>[abc]</code>	Simple	Matches a or b or c
<code>[^abc]</code>	Negation	Matches any character except a or b or c
<code>[a-zA-Z]</code>	Range	Matches a through z, or A through Z (inclusive)
<code>[a-d[m-p]]</code>	Union	Matches a through d, or m through p
<code>[a-z&&[def]]</code>	Intersection	Matches d, e, or f
<code>[a-z&&[^xq]]</code>	Subtraction	Matches a through z, except for x and q

Predefined Character Classes

Predefined character classes are convenient shorthands for commonly used regular expressions.

Construct	Description	Example
<code>.</code>	Matches any single character (except newline)	<code>.at</code> matches "cat", "hat", and also "bat" in the phrase "batch files"
<code>\d</code>	Matches any digit character (equivalent to <code>[0-9]</code>)	<code>\d</code> matches "3" in "C3PO" and "2" in "file_2.txt"
<code>\D</code>	Matches any non-digit character (equivalent to <code>[^0-9]</code>)	<code>\D</code> matches "S" in "900S" and "Q" in "Q45"
<code>\s</code>	Matches any single white-space character (equivalent to <code>[\t\n\x0B\f\r]</code>)	<code>\sbook</code> matches "book" in "blue book" but nothing in "notebook"
<code>\S</code>	Matches any single non-white-space character	<code>\Sbook</code> matches "book" in "notebook" but nothing in "blue book"
<code>\w</code>	Matches any alphanumeric character, including underscore (equivalent to <code>[A-Za-z0-9_]</code>)	<code>r\w*</code> matches "rm" and "root"
<code>\W</code>	Matches any non-alphanumeric character (equivalent to <code>[^A-Za-z0-9_]</code>)	<code>\W</code> matches "&" in "stmd &", "%" in "100%", and "\$" in "\$HOME"

POSIX Character Classes (US-ASCII)

POSIX has a set of character classes that denote certain common ranges. They're similar to bracket and predefined character classes, except they take into account the locale (the local language/coding system).

Construct	Description
\p{Lower}	A lower-case alphabetic character, [a - z]
\p{Upper}	An upper-case alphabetic character, [A - Z]
\p{ASCII}	An ASCII character, [\x00 - \x7F]
\p{Alpha}	An alphabetic character, [a - zA - z]
\p{Digit}	A numeric digit, [0 - 9]
\p{Alnum}	An alphanumeric character, [a - zA - z0 - 9]
\p{Punct}	A punctuation character, one of !"#\$%&'()*+,-./:;<=>?@[\]^_`{ }~
\p{Graph}	A visible character, [\p{Alnum}\p{Punct}]
\p{Print}	A printable character, [\p{Graph}\x20]
\p{Blank}	A space or tab, [t]
\p{Cntrl}	A control character, [\x00 - \x1F\x7F]
\p{XDigit}	A hexadecimal digit, [0 - 9a - fA - F]
\p{Space}	A whitespace character, [t\n\x0B\f\r]

Regex Line and Word Boundaries

You can use boundary matching constructs to specify where in a string to apply a matching pattern. For example, you can search for a particular pattern within a word boundary, or search for a pattern at the beginning or end of a line.

Construct	Description	Example
^	Matches from the beginning of a line (multi-line matches are currently not supported)	^172 matches the "172" in IP address "172.18.1.11" but not in "192.172.2.33"
\$	Matches from the end of a line (multi-line matches are currently not supported)	d\$ matches the "d" in "maid" but not in "made"
\b	Matches within a word boundary	\bis\b matches the word "is" in "this is my island", but not the "is" part of "this" or "island". \bis matches both "is" and the "is" in "island", but not in "this".
\B	Matches within a non-word boundary	\Bb matches "b" in "sbin" but not in "bash"

Regex Quantifiers

Quantifier Constructs

Quantifiers specify how often the preceding regular expression construct should match. The classes of quantifiers are:

- Greedy
- Reluctant
- Possessive

The difference between greedy, reluctant, and possessive quantifiers involves what part of the string to try for the initial match, and how to retry if the initial attempt doesn't produce a match.

By default, quantifiers are *greedy*. A greedy quantifier first tries for a match with the entire input string. If that produces a match, then it considers the match a success and the engine can move on to the next construct in the regular expression. If the first try doesn't produce a match, the engine backs-off 1 character at a time until it finds a match. So a greedy quantifier checks for possible matches in order from the longest possible input string to the shortest possible input string, recursively trying from right to left.

Adding a ? (question mark) to a greedy quantifier makes it *reluctant*. A reluctant quantifier first tries for a match from the beginning of the input string, starting with the shortest possible piece of the string that matches the regex construct. If that produces a match, then it considers the match a success and the engine can move on to the next construct in the regular expression. If the first try doesn't produce a match, the engine adds 1 character at a time until it finds a match. So a reluctant quantifier checks for possible matches in order from the shortest possible input string to the longest possible input string, recursively trying from left to right.

Adding a + (plus sign) to a greedy quantifier makes it *possessive*. A possessive quantifier is like a greedy quantifier on the first attempt (it tries for a match with the entire input string). The difference is that unlike a greedy quantifier, a possessive quantifier doesn't retry a shorter string if it doesn't find a match. If the initial match fails, the possessive quantifier reports a failed match. It doesn't make any more attempts.

Greedy Construct	Reluctant Construct	Possessive Construct	Description	Example
?	??	?+	Matches the previous character or construct once or not at all.	st?on matches "son" in "johnson" and "ston" in "johnston" but nothing in "clinton" or "version"
*	*?	*+	Matches the previous character or construct zero or more times.	if* matches "if", "iff" in "diff", or "i" in "print"
+	+?	++	Matches the previous character or construct 1 or more times.	if+ matches "if", "iff" in "diff", but nothing in "print"
{n}	{n}?	{n}+	Matches the previous character or construct exactly <i>n</i> times.	o{2} matches "oo" in "lookup" and the first 2 o's in "fooooo" but nothing in "mount"
{n, }	{n, }?	{n, }+	Matches the previous character or construct at least <i>n</i> times.	o{2, } matches "oo" in "lookup" all 5 o's in "fooooo" but nothing in "mount"
{n, m}	{n, m}?	{n, m}+	Matches the previous character or construct at least <i>n</i> times, but no more than <i>m</i> times.	F{2, 4} matches "FF" in "#FF0000" and the last 4 F's in "#FFFFFF"

Regex Capturing Groups

You can use a pair of parentheses around a subpattern in a regular expression to define a *group*. You can use regex groups to:

- **Apply regex operators and quantifiers to an entire group at once.**

- **Create a capturing group.** You can use capturing groups to specify matching values to save or return from your regular expression. By default, a group *captures* the text that produces a match. The portion of the string that matches the grouped subexpression is captured in memory for later retrieval or use.
- **Create a non-capturing group.**

Group Numbering

A regular expression can have more than 1 group, and the groups can be nested. The groups are numbered 1-*n* from left to right, starting with the first opening parenthesis. There is always an implicit group zero (0), which contains the entire match. Example:

```
(a(b*))+(c)
```

This pattern contains 3 groups:

```
group 1: (a(b*))
group 2: (b*)
group 3: (c)
```

Capturing Groups and the REGEX Function

You can use the REGEX function to extract a portion of a string. The REGEX function returns the value of the *first* capturing group only.

Suppose you have a field name called *email* that contains email addresses with this pattern: *username@provider.domain*. You can use the REGEX function to return just the *provider* portion of the email address from the *email* field:

```
REGEX([email], "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9._-]+\.[a-zA-Z]{2,4}$")
```

Capturing Groups and the REGEX_REPLACE Function

You can use the REGEX_REPLACE function to match a string, and replace matched strings with another value. The REGEX_REPLACE function takes 3 arguments:

- Input string
- Matching regex
- Replacement regex

You can use capturing groups to capture backreferences, but the entire match is always returned.

Capturing Groups and Backreferences

You can use a backreference to refer back to the matched content of a particular capturing group number. Typically, you use a backreference in the same regular expression that contains the capturing group. You specify a backreference by referring to the group number preceded by a backslash. Use \1 to refer to capturing group 1, \2 to refer to capturing group 2, and so on.

Suppose you want to match a pair of HTML tags and their enclosed text. You could capture the opening tag into a capturing group, and then use a backreference to match the corresponding closing tag:

```
(<([A-Z][A-Z0-9]*)\b[^\>]*.*?</\2>)
```

This regular expression contains 2 capturing groups:

- Group 1 contains the outermost parentheses and captures the entire string.
- Group 2 captures the string matched by `[A-Z][A-Z0-9]*`.

You can then refer to group 2 using the \2 backreference to match the corresponding closing HTML tag.

When you use the `REGEX_REPLACE` function, you can use a backreference to refer to a capturing group in the *previous* regular expression. The syntax is slightly different when you use a backreference to refer to a group in the previous regex. Use a dollar sign (\$) before the group number, such as \$1 to specify a backreference to group 1 of the previous expression.

Suppose you have a `phone_number` field where the values are formatted as `xxx.xxx.xxxx`. The following example matches the values in `phone_number` and replaces them with values formatted as `(xxx) xxx-xxxx`:

```
REGEX_REPLACE([phone_number], "([0-9]{3})\\.([0-9]{3})\\.([0-9]{4})", "\\($1\\) $2-$3")
```

Notice the backreferences in the replacement expression. They refer to the capturing groups of the previous matching expression.

Non-Capturing Groups

In some cases, you might want to use parenthesis to group subpatterns, but *not* capture text. A non-capturing group starts with `(?:` (a question mark and colon following the opening parenthesis). For example, `h(?:a|i|o)t` matches `hat` or `hit` or `hot`, but does not capture the `a`, `i`, or `o` from the subexpression.