Xcalar Insight Online Help

For Cluster Administrators

Publication date: 12/13/2016

www.xcalar.com

Table of Contents

Getting started

This section provides information that helps you get started with Xcalar Insight. Read this section if you have not used this tool.

Requirements for Xcalar Insight

Before starting Xcalar Insight, make sure that the following requirements are met:

- You use Chrome as a web browser on your computer. Xcalar Insight is not supported on other browsers, and does not run on mobile devices such as tablets or cell phones.
- The data sources from which you want to obtain the datasets are accessible to Xcalar Insight through the NFS or HDFS protocol.

The minimum recommended screen resolution for Xcalar Insight is 1024 x 768.

Basic Xcalar Insight concepts

This section explains basic concepts you must understand to use Xcalar Insight efficiently.

Understanding the purpose of Xcalar Insight

Xcalar Insight is an HTML5-based visual tool that enables you to interactively and intuitively design queries and algorithms, and manipulate your data to derive insights. The dataflows generated when you use Xcalar Insight interactively can be saved. You can run the saved dataflows on demand or at scheduled intervals at a later time.

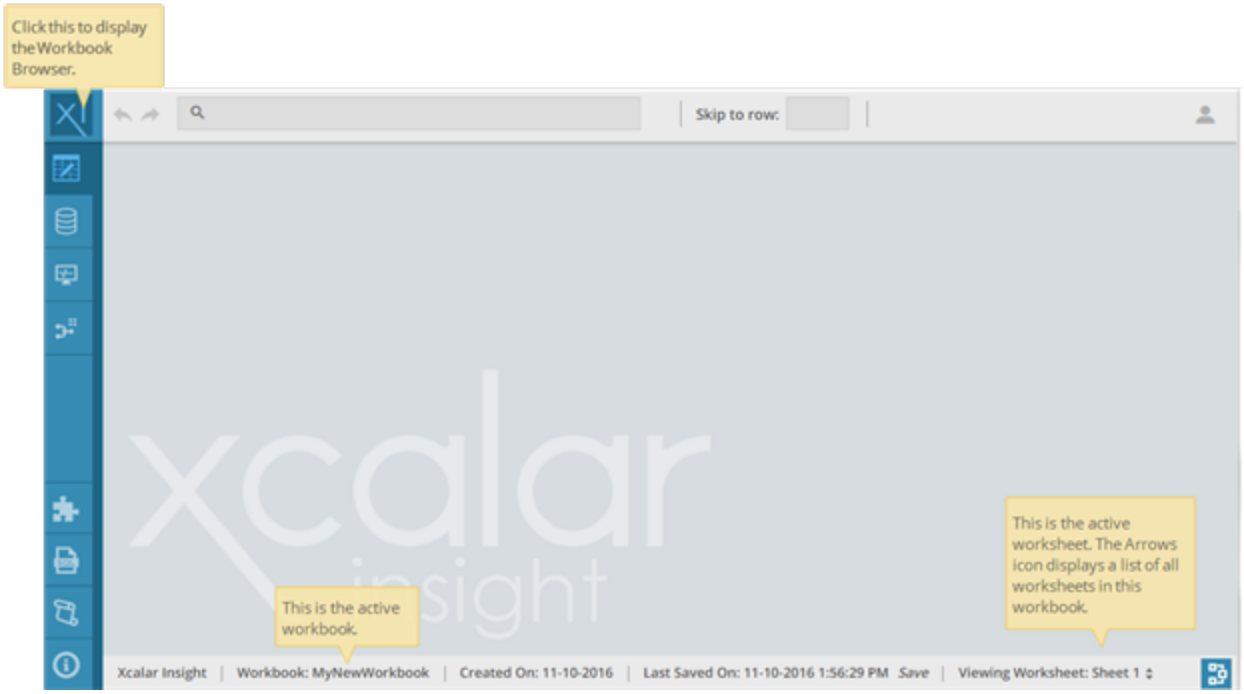
The data being queried can be arbitrary data in arbitrary formats. It can reside on data sources such as an NFS, HDFS, Amazon S3, or local ext4 file system.

Understanding workbooks and worksheets

You can create workbooks in Xcalar Insight, which can be regarded as files. Within a workbook, you can create as many worksheets as you want. Worksheets in the same workbook usually contain data related to the same project. The worksheet displayed on your screen is the active worksheet.

The relationship between a worksheet and a workbook is similar to the relationship between an Excel worksheet and workbook.

The following sample screen shows a **Worksheet** window for a workbook titled MyNewWorkbook.



Understanding tables

Tables are created from a dataset reference that points to a data source. You can manipulate and transform data within a table (for example, by sorting or filtering) or you can manipulate multiple tables to create a new table (for example, by joining). You use table operations to implement an algorithm that you design for deriving meaning from your data. With Xcalar's True Data-In-Place architecture, data is pointed to at the source and analyzed without any ETL (Extract, Transform, and Load).

Each table is created with a name unique in the worksheet. You can rename a table after the table is created.

When you create a table from the dataset reference, you can select the fields of interest to be included in the table. Fields from the data source can be added to an existing table any time.

Data formats supported

Xcalar Insight natively supports CSV, JSON, Excel, and raw text, and can support proprietary formats using UDFs.

> **NOTE:** The CSV format includes not only comma-separated values but also other delimiter-separated values. The default field delimiter is tab and the default record delimiter is newline (\n). You can select the delimiter for separating fields and for separating records.

How Xcalar Insight saves your work

Xcalar Insight saves the results of all data operations as they are being completed. For example, after you filter a column, the resultant table is saved automatically. You do not have to click **Save**.

If your work is for manipulating the appearance of a table or a table column, the work is saved automatically when you sign out. However, before this automatic save, if you refresh your browser, your browser displays a message warning about unsaved changes. You must click **Save** to avoid losing the changes. The following screenshot shows the location of **Save**.

> **EXAMPLE:** After you delete the first table column and resize the second column, you can sign out without clicking **Save**. These changes are automatically saved. However, if you refresh the browser after making these changes instead of signing

> out, the browser warns that you have unsaved changes. If you proceed with the browser refresh, the table in the refreshed browser does not reflect the deletion of the first column and the new width of the second column.

> **IMPORTANT:** If you use the same browser window to go to another website, you can use the browser's **Back** button to return to Xcalar Insight. This causes the browser to reload the Xcalar Insight page, and the result is similar to a browser refresh. This means that your changes are lost. Therefore, always click **Save** before you use the same browser window to navigate to another website.

Understanding Xcalar product versions

XCE and Xcalar Insight have separate version numbers. You can display both versions through Xcalar Insight.

Displaying Xcalar Compute Engine product versions and license expiration date

To display Xcalar product versions and the license expiration date, follow these steps:

1. Click in the upper right corner of the Xcalar Insight window.
2. In the drop-down menu, click **About** to display a modal window that lists product versions, license expiration date, and copyright information.

Easy steps (with default settings)

This section describes the steps up to the point where you can start performing operations in tables presented in a workbook. It assumes that:

- the data format of your data source is natively supported by XDP.
- field names in the data source do not contain illegal characters.
- you accept all the default settings suggested by Xcalar Insight.

> **IMPORTANT**: The easy steps are appropriate if you want to get started quickly so that you can familiarize yourself with Xcalar Insight. When you are ready to use Xcalar Insight in a production environment, follow the steps in Detailed steps (with custom settings), which provides detailed information that might be required for your particular data source.

Step 1: Starting Xcalar Insight

Follow these steps to start Xcalar Insight:

1. Point the browser using the HTTPS protocol to the location where Xcalar Insight is installed. For example, enter the following URL:

   ```
   https://xcalar.mycompany.com/index.html
   ```

2. Log in by using the login name and password given to you by the Xcalar Compute Engine administrator.

> **NOTE:** Each Xcalar Insight user can have one login at a given time. Closing the browser window does not automatically log you out of Xcalar Insight. If you are logged in to Xcalar Insight and you try to log in again, a message informs you that you cannot have concurrent logins. You can either go back to the login screen or continue the login, which causes Xcalar Insight to terminate your other connection.

> Remember that if you have unsaved changes through the other connection, the termination causes the changes to be lost.

Step 2: Creating a workbook as your workspace

Follow these steps to create a workbook:

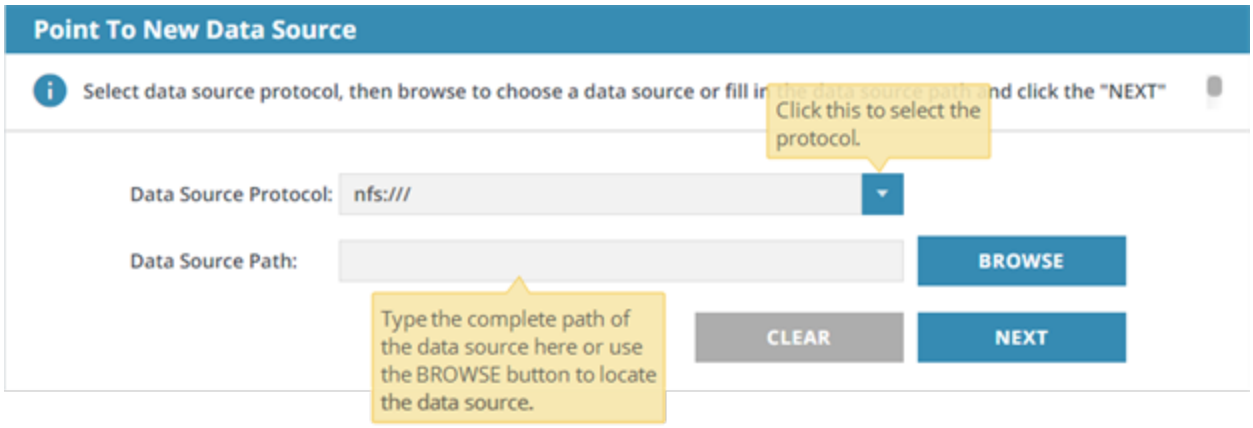
1. The **Workbook Browser** is automatically displayed after you log in to Xcalar Insight for the first time. Enter the workbook name and click **Create Workbook** on the **New Workbook** card.
2. Click  to activate the workbook. The icon is the first one on the menu when you hover over the workbook card.

Step 3: Pointing to your data source

Follow these steps to point to your data source:

1. Click  (**Data Stores** icon). Its location is shown in the following screenshot:

2. In the **Point To New Data Source** window, select the protocol for accessing the data source. It can be NFS or HDFS. The following screen illustrates the **Point To New Data Source** window.



3. Click **BROWSE** to display the **Browse Data Source** window, in which you can navigate to

the data source. Click to choose the file or directory to be used as your data source, then click **NEXT**.

4. The **Point to New Data Source** window shows a preview of the records in the data source. Create a dataset reference and a table at the same time by clicking **CREATE TABLE**. The table contains all columns in the data source.

You now have a dataset reference pointing to a data source through the selected protocol. The data reference is listed in the **Dataset Refs** section of the left panel.

Next step

If you know what task to perform on your table, see Common tasks to find the instructions for the task. Otherwise, go to the following topics to learn about various Xcalar Insight windows:

- Workbook Browser
- Worksheet window with tables
- Xcalar Insight window

Detailed steps (with custom settings)

This section provides detailed information about the steps up to the point where you can start performing operations in tables presented in a Xcalar Insight workbook. Read this section if you want to better understand the fields displayed when you start using Xcalar Insight and if you want to override default settings when pointing to your data source.

Starting Xcalar Insight

Follow these steps to start Xcalar Insight:

1. Point the browser using the HTTPS protocol to the location where Xcalar Insight is installed. For example, enter the following URL that represents the location of Xcalar Insight:

   ```
   https://xcalar.mycompany.com/index.html
   ```

2. Log in by using the login name and password given to you by the Xcalar Compute Engine administrator. If this is your first login, the **Workbook Browser** is displayed. Otherwise, Xcalar Insight resumes where you left off the last time you used it.

> **NOTE:** Each Xcalar Insight user can have one login at a given time. Closing the browser window does not automatically log you out of Xcalar Insight. If you are logged in to Xcalar Insight and you try to log in again, a message informs you that you cannot have concurrent logins. You can either go back to the login screen or continue the login, which causes Xcalar Insight to terminate your other connection. Remember that if you have unsaved changes through the other connection, the termination causes the changes to be lost.

Before creating a workbook as your workspace

Before creating a book, decide on the workbook name. Workbook names can be changed at any time.

The following list describes the naming conventions:

- Names are case-sensitive.
- Length can be from 1 to 255 characters.
- There are no restrictions on special characters.

Creating a workbook as your workspace

Follow these steps to create a workbook:

1. The **Workbook Browser** is automatically displayed after you log in to Xcalar Insight for the first time. At other times, you can display or dismiss the **Workbook Browser** by clicking  in the upper left corner of the window. In the **Workbook Browser**, enter the workbook name and click **Create Workbook** on the **New Workbook** card.

   After the workbook is created, it contains one worksheet named **Sheet 1**.

2. The newly created workbook is inactive. To activate it, click  , which is the first icon on the menu when you hover over the workbook card. (An active workbook is the one you can work on, and you can have only one active workbook at a time.)

For more information about the **Workbook Browser**, see Workbook Browser.

Prerequisites for pointing to your data source

Make sure that your data source meets the following prerequisites before pointing Xcalar Insight to it:

- Pointing to a data source means telling Xcalar Insight to derive metadata from one or multiple files containing raw data. These files must be accessible to XDP via the NFS or HDFS protocol.
- The data source can be a file or a directory. If you point to a directory, Xcalar Insight points to data in all files in the directory. You can also point to subdirectories recursively. Make sure that all files being pointed to are in the same format and have the same schema. Otherwise, the result might be unpredictable.
- Unicode is supported for both delimiter-separated values and JSON that is UTF-8 encoded. UTF-16 or other encoding systems are not supported without UDFs.

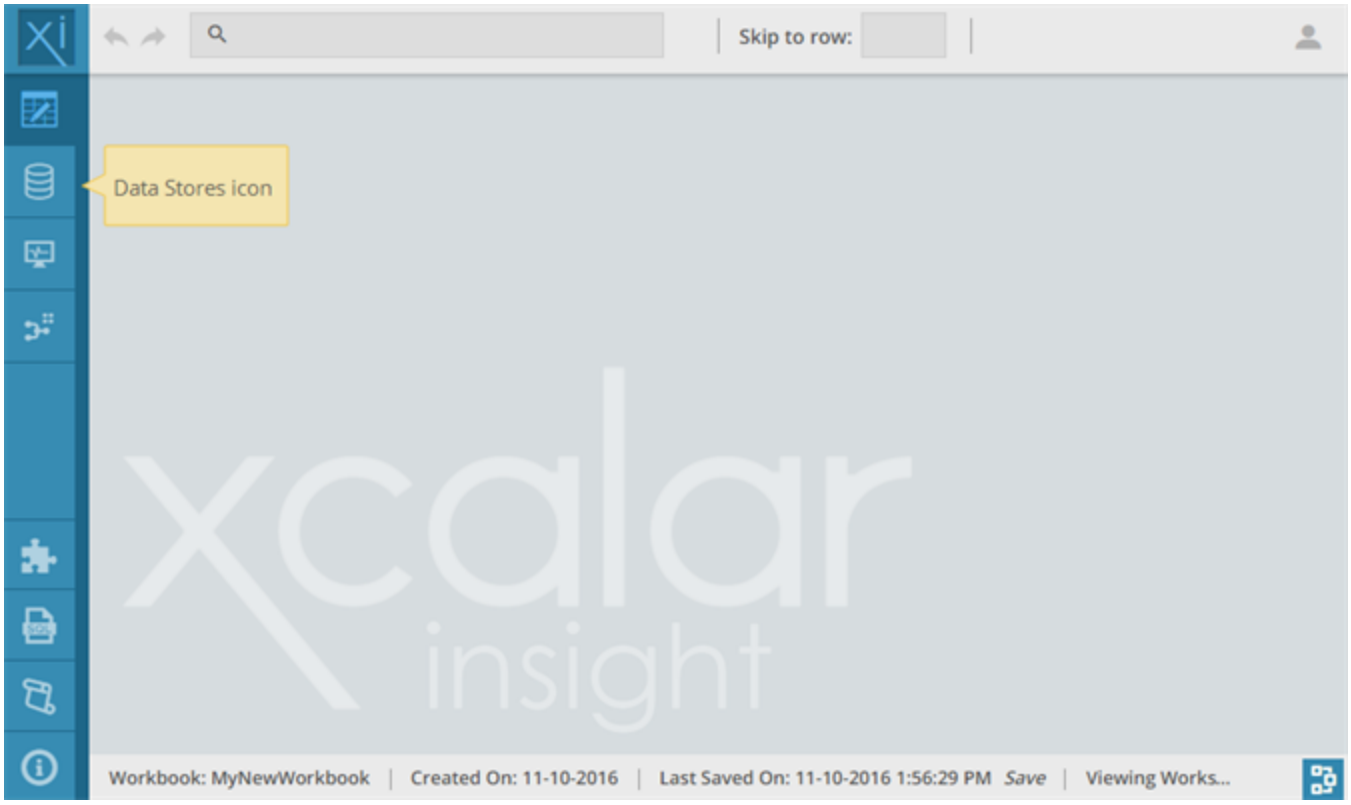
- If you plan to create a dataset by pointing to multiple files with the same schema, create a directory and place the files in the directory. Then you can point to the directory. Alternatively, if the source files are in a directory with other files, you can rename the source files so that Xcalar Insight can find them with the wild card character (*) when pointing to them. For example, suppose you want to create a dataset by pointing to airlines data gathered in the year 2015 and 2016, which are in two files. You can create a directory named AirlinesData and place both files in the directory. Then point to the AirlinesData directory. If you do not want to create a directory, you can give names to the files such as airlines2015.csv and airlines2016.csv. Then instruct Xcalar Insight to point to airlines201*.csv. Preparing the source files in this way enables you to create one dataset with records from both files.

> **CAUTION**: Before you proceed, determine whether the data source contains field names with illegal characters. If so, you can still point to the data source but you must use a user-defined function (UDF) provided by Xcalar to replace these characters. (Information about using UDFs is available later in this section.) If you do not use the UDF when pointing to your data, you might see unexpected query results. The following is a complete list of disallowed characters in field names:
>
> - single quote mark (')
> - double quote mark (")
> - parenthesis (( ))
> - square bracket ([ ])
> - left or right brace ({ })
> - backslash (\)
> - two consecutive colons (::)
> - two consecutive dashes (--)
> - comma (,)
> - period (.)
> - asterisk (*)

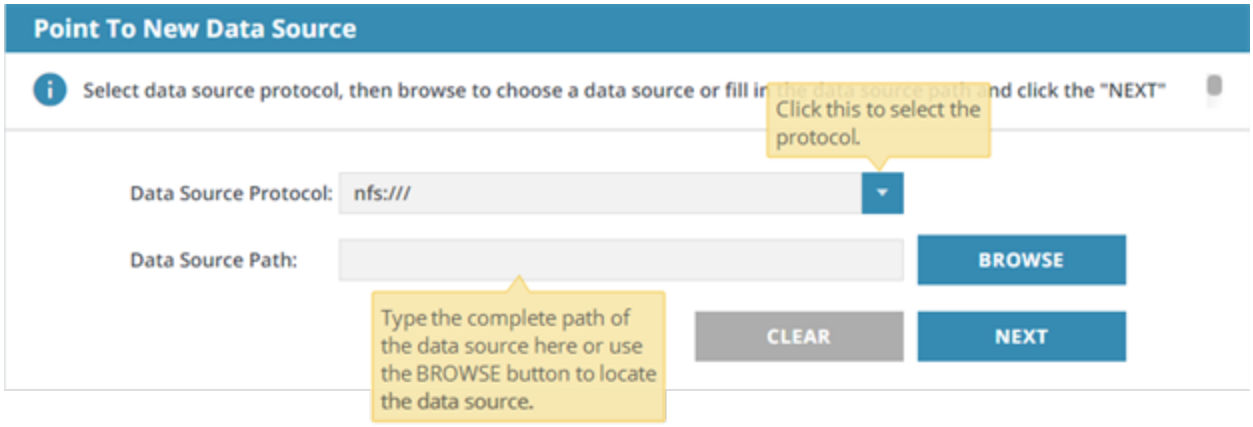
Pointing to your data source

Follow these steps to point to your data source:

1. Click  (**Data Stores** icon). Its location is shown in the following screenshot:



2. In the **Point To New Data Source** window, select the protocol for accessing the data source. It can be NFS or HDFS. The following screen illustrates the **Point To New Data Source** window.



3. Select the path to the data source in one of two ways:

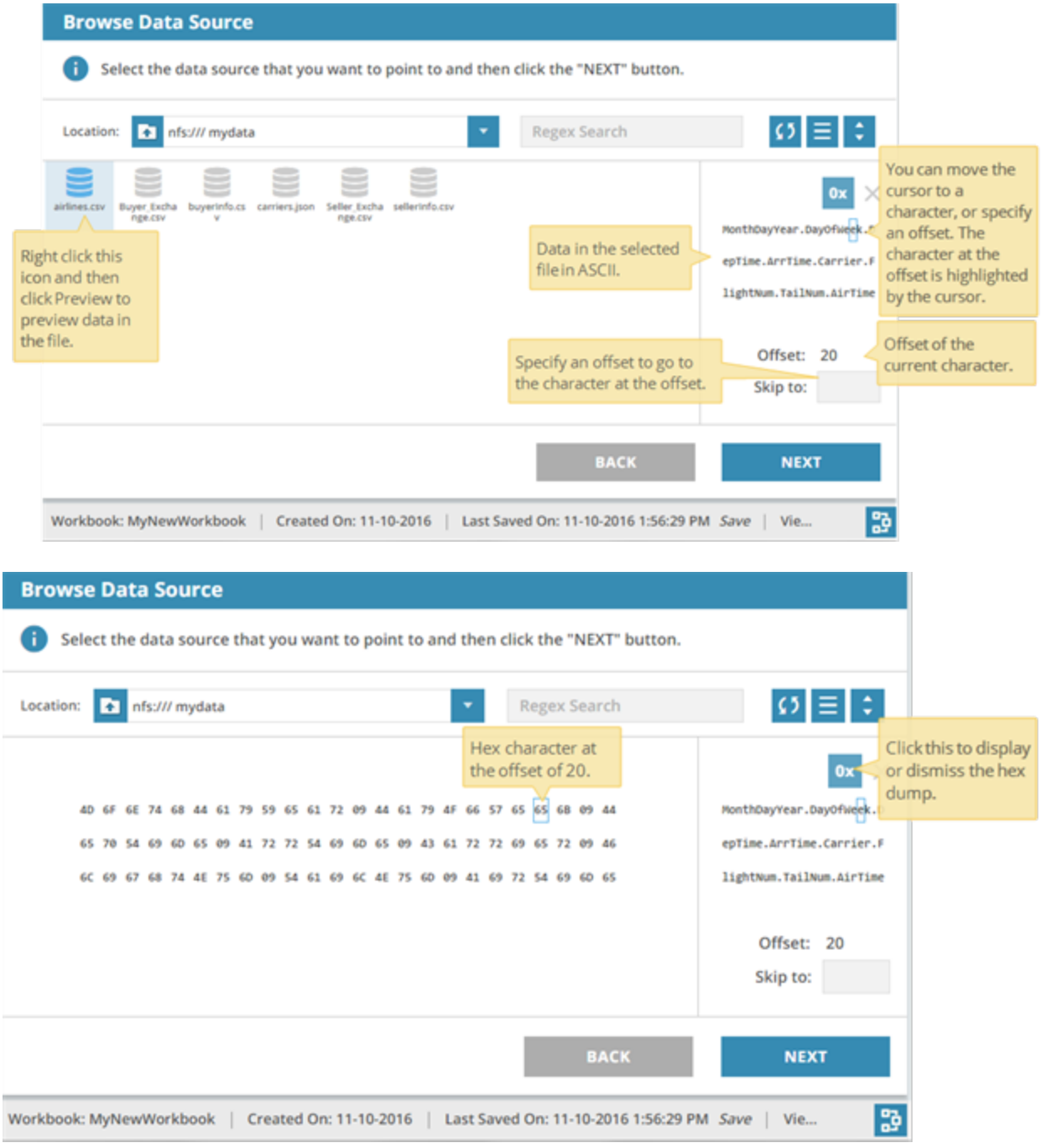
   - Enter the complete path to a file or directory as in the following example:

     `/mydata/airlines.csv`

   - Click **BROWSE** to display the **Browse Data Source** window, in which you can navigate to the data source. Click to choose the file or directory to be used as your data source.

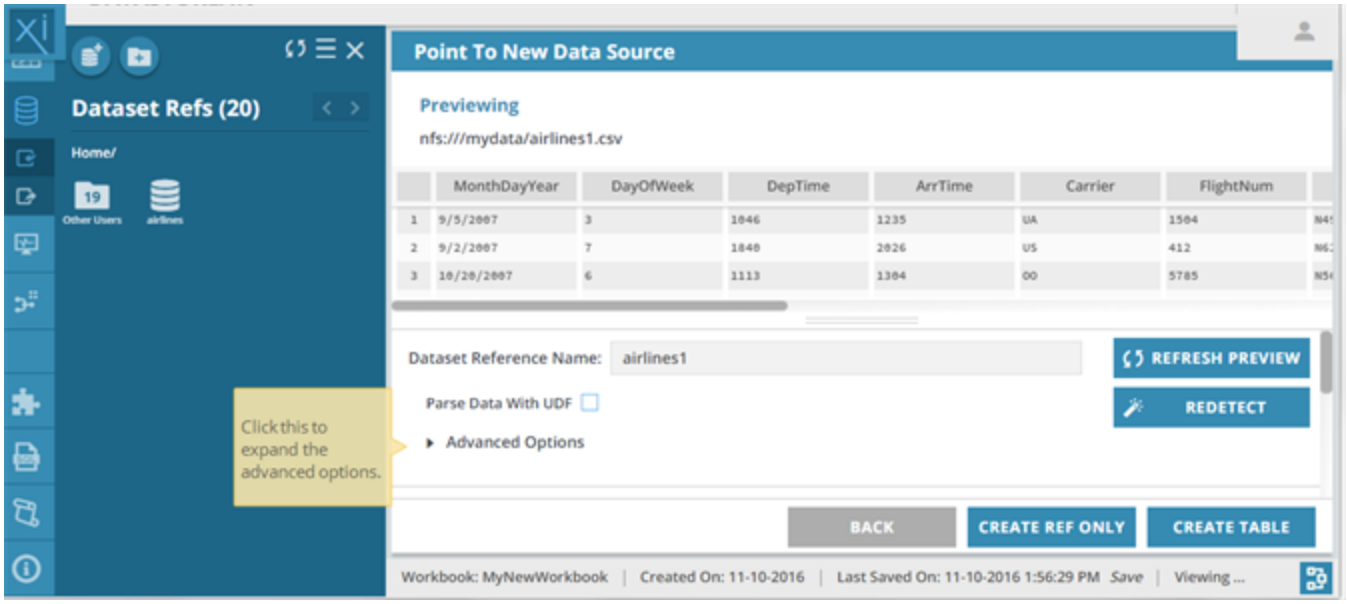
     > **TIP:** If you have a large number of files, use the **Regex Search** field to filter out irrelevant files. For example, typing the regular expression **us(t|a)** filters out all files but these two: **usability** and **justin**.

     Optionally, you can right click a file icon in this window and then click **Preview** to preview the raw data before selecting the file. You can preview the file in ASCII format or as a hex dump. No previewer is available for directories. The following screenshots explain how to preview a file named airlines.csv.



4. (Optional) After you select the data source, you can specify values for the advanced options as described in About advanced options. The following screenshot shows the location of the options. In most cases, you can skip the advanced options and go on to the

next step.



5. Click **NEXT**.
6. The **Point to New Data Source** window shows a preview of the records in the data source in a tabular format. This preview enables you to verify that you have selected the intended data source file. You can position your mouse on a column header and drag it to resize each column. You can also use your mouse to scroll the rows.
7. In the **Dataset Reference Name** field, accept the suggested name or type the desired name. The dataset reference gives you a way to preview the dataset and create tables from the dataset after you successfully point to your data source. The name can be changed later. For more information about dataset references, see Managing dataset references.

   Expand the following list to see the dataset reference naming rules:

   - Dataset reference names are case-sensitive.
   - Length can be from 1 to 255 characters.
   - Characters can be A-Z, a-z, 0-9, and underscore (_).
   - First character must be a letter.

8. If you want Xcalar Insight to invoke a user-defined function (UDF) when parsing the data in

the data source, select **Parse Data With UDF** and then specify the module name and function name. A UDF for parsing data before Xcalar Insight points to the data is called a streaming UDF.

For more information about UDFs, see Understanding user-defined functions (UDFs) and Using user-defined functions (UDFs). If no UDFs are needed for parsing the data, skip to the next step.

> **IMPORTANT**: Parsing a data source with a streaming UDF is required if your data source contains illegal characters. For a list of these characters, see Prerequisites for pointing to your data source.

9. The fields in the rest of the **Point to New Data Source** window tell Xcalar Insight about the data format and layout. Xcalar Insight automatically detects the format and layout of the data source, but you can override the default values. :

> **TIP:** After you enter values in these fields, if you want to see the values recommended by Xcalar Insight again, click **REDETECT**.

The following table describes the fields.

Field	Description
Format	It can be separated values, JSON, Excel, or Text.

Field	Description
Promote First Row as Header	Depending on the data format, Xcalar Insight automatically selects or deselects this option. For example, if the data source is a CSV file, Xcalar Insight recommends that you promote the first row as header. When pointing to multiple files with the same first row, Xcalar Insight promotes the first row of the first file and omits the first row of subsequent files. If you use the **Skip Rows** field to skip rows that do not require analysis, Xcalar Insight promotes the first row after the row-skipping. > **EXAMPLE:** The file **employees.csv** contains comments in the first row, and the strings **Name** and **Salary** in cells on the second row. You can specify **1** in the **Skip Rows** field to eliminate the first row and promote the row containing **Name** and **Salary** as header. > **IMPORTANT:** You can promote the first row only at this point. You will not be able to change the first row into the header row at a later time.
Record Delimiter	The character for separating records, which can be \n or a null character.
Field Delimiter	The character separating fields, which can be a tab or a comma.

Field	Description
Quoting Character	The character surrounding a string in the data source. The double quote mark (") is the default quoting character in Xcalar Insight. If the data source uses another character to enclose a string, enter the character in this field.
Skip Rows	The number of rows to omit at the beginning of each file when XCE points to the data source. Typically you omit rows if they do not contain data requiring analysis (for example, rows containing comments). If you point to a directory with multiple files, the same number of rows are skipped in each file. If you use a UDF when pointing to a data source, the UDF parses the data before the rows are skipped.

10. Follow one of these two steps:
    - To create a dataset reference only to reference the dataset that will be created by XDP, click **CREATE REF ONLY**. You can use this dataset reference to preview data and create tables later.
    - To create a dataset reference and a table at the same time, click **CREATE TABLE**. The table contains all columns in the dataset.

You now have a dataset on your cluster pointing to a data source through the selected protocol. The dataset reference associated with the dataset is listed in the **Dataset Refs** section of the left panel. If you create a table at the time the dataset reference is created, you also have a table in the active worksheet.

About advanced options

Advanced options are provided when you select a data source. The options enable you to accomplish the following tasks:

- point to files with names matching the specified pattern
- point to files recursively

- set the dataset size

Pattern matching

To point to only files with names that match a particular string pattern, type the string with a wild card character in the **Pattern** field. The wild card character is an asterisk (*). For example, to point to all files with names ending with csv, type ***csv** in this field.

Alternatively, you can enter a regular expression in the **Pattern** field and select the **regex** option. For example, you can enter the regular expression **sellers(1|2|3)** to point to **sellers1info.csv**, **sellers2data.csv**, and **sellers3.Info.csv**, but not **serllers4.csv**.

Recursive pointing

If you point to a directory, you can tell Xcalar Insight to traverse all subdirectories to point to all files in the subdirectories.

When pointing to files recursively, you can use pattern matching to restrict the files to those with names that match the specified pattern.

Dataset size

The **Dataset Size** field limits the size of the dataset created by Xcalar Insight when it points to a data source. If pointing to a data source results in a dataset that exceeds the setting of this field, the attempt to point to the data source fails with an error message.

> **NOTE:** Regardless of the setting, if the size of the smallest file in the data source exceeds this limit, the attempt to point to the data source fails. The dataset created by Xcalar Insight must point to at least one file in the data source. It cannot point to a partial file.

The following list describes the meanings of the options for this field:

- **Entire file/folder**: If this option is selected, the maximum dataset size is 1 TB by default. Due to the high limit, it is likely that Xcalar Insight can point to all data in the entire data source, whether it is a file or a folder. This option is particularly useful for interactive

analysis.

You, the cluster administrator, can change the maximum dataset size by setting the value of the **MaxInteractiveDataSize** parameter through the **Setup** icon of the Monitor. For information about setting parameter values, see Configuring parameters. The maximum dataset size can be changed by the cluster administrator. Consult your administrator to determine the current value.

> **EXAMPLE:** If your data source is a 900-GB file or a folder consisting of files that add up to 900 GB, Xcalar Insight can point to all data in the data source because the dataset size is below the 1-TB limit. If your data source is a 1.2 TB file, the attempt to point to it fails. If your data source is a folder consisting of files that add up to 1.2 TB, Xcalar Insight points to as many files in the folder as possible until the 1-TB limit is reached, provided that the smallest file in the folder is less than 1 TB.

- **Custom size**: Typically, you use this option when you point to a folder. You can set a limit on the dataset size such that Xcalar Insight will create a dataset from as many files as allowed by the limit. If you point to a data source that is a single file, the file must be under the specified limit, or the attempt to point to it fails. This option is particularly useful for previewing raw data and modeling.

  The factory default for this field is 10 GB. You can override this default through the **Settings** icon in the Monitor. Your new default value is applied each time you point to a new data source. (For more information about the Monitor, see Using the Monitor.) The **Custom size** field automatically reflects the default currently in effect. If pointing to a data source requires a size greater than the default, simply enter the desired value in the **Custom size** field.

  > **EXAMPLE:** Suppose the dataset size is left at the factory default (10 GB). If your data source is a 5-GB file or folder, you can point to all data in the data source. If the data source is a 15-GB file, the attempt to point to it fails. If the

> data source is a 15-GB folder, Xcalar Insight points to as many files as allowed by the 10-GB limit, provided that the smallest file is smaller than 10 GB.

> **EXAMPLE:** Suppose you use the Monitor to change the default dataset size to 20 GB. Each time you point to a new data source, the **Custom size** field automatically shows the default you set (20 GB). To override this default, simply enter the desired value. For example, to be able to point to a data source that is a 30-GB file, you must enter 30 GB or a greater value.

What to do next

Follow one of these steps:

- Create a table (if a table is not yet created) as described in Creating a table.
- If you know what task you need to perform, see Common tasks to find the instructions for the task.
- Learn more about various Xcalar Insight windows from these topics:
  - Workbook Browser
  - Worksheet window with tables
  - Xcalar Insight window

Common tasks

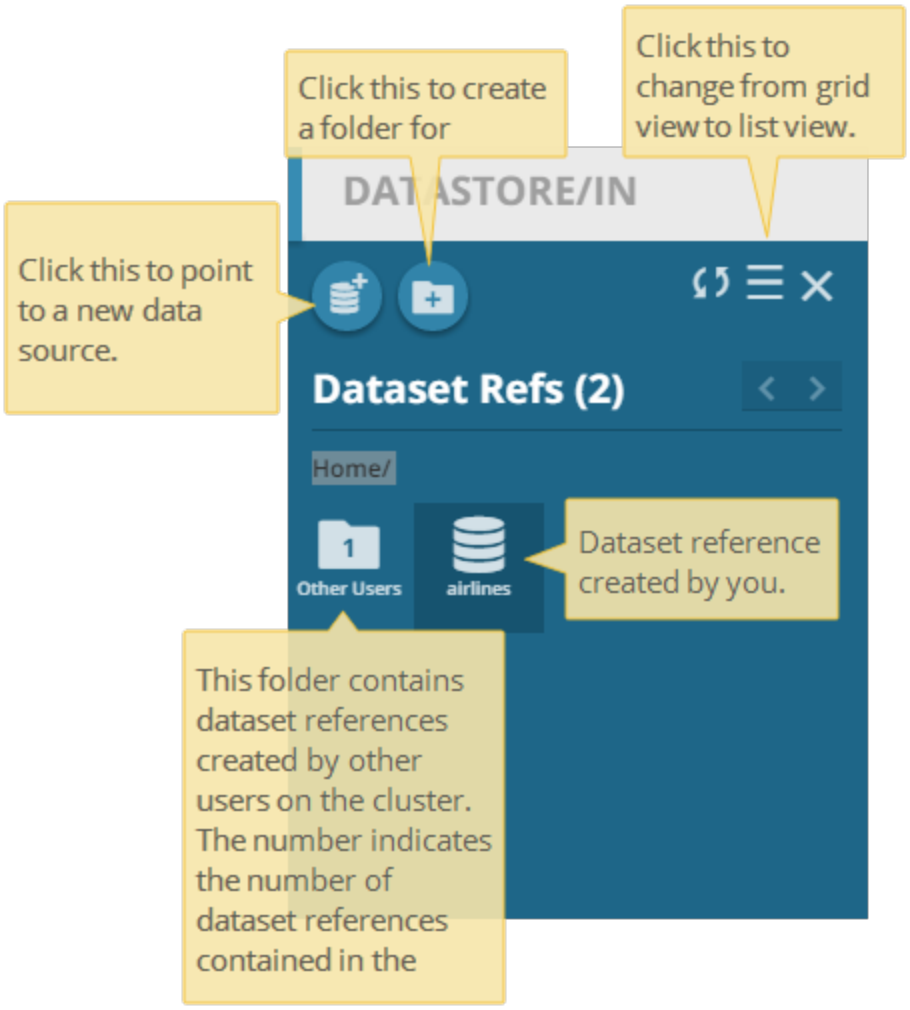
This section assumes that you have followed the instructions in Detailed steps (with custom settings). The tasks described are the ones that you most likely use when you try to gain meaning from your data with Xcalar Insight.

Managing dataset references

When you point to a data source, XDP creates a dataset (typically in memory) which is the representation of raw data and metadata. In order for you to interact with the dataset, Xcalar Insight creates a dataset reference through which you can preview the data in the dataset and create tables with all or selected fields in the dataset.

Xcalar Insight provides a default name for each dataset reference when you point to a new data source. The name is the same as the name of the source file or directory. You can enter a different name or accept the default. You cannot rename the dataset reference at a later time.

Dataset references are listed in the **DATASTORE/IN** panel and are shared by all users on the same cluster. The following partial screenshot shows an example of the dataset reference list.



Organizing dataset references in folders

In the **Dataset Refs** panel, you can organize dataset references in folders. For example, you can create a folder named **Finance** for dataset references pointing to data sources in the Finance department, another folder named **Engineering** for dataset references pointing to data sources in the Engineering department, and so on.

Deleting a dataset reference

You can delete a dataset reference whether or not you created it. However, you must drop all tables in your workbook created from the dataset reference before you can delete the dataset

reference. If you have an active, hidden, or temporary table that was created from the dataset reference, the dataset is still in use. As a result, you cannot delete the dataset reference.

If another user has tables that were created from your dataset reference, you can delete it from your list of data references. However, the dataset reference continues to exist for the other user.

> **EXAMPLE**: You (userA) have a dataset reference named airlines. If tables (whether active, hidden, or temporary) created from airlines exist in your workbook, your attempt to delete airlines results in an error message. After you drop the tables created from this dataset reference, you can successfully delete it. If another user, userB, has tables created from airlines, even though airlines no longer appears in your dataset reference list, userB continues to see it in userA's dataset reference folder. The dataset reference can be deleted by userB only after userB drops all tables created from it.

Effects of deleting a dataset reference

After you delete a dataset reference, it no longer appears in your list of dataset references. As a result, you no longer have a way to preview the dataset or create tables from it.

The dataset associated with a dataset reference being deleted is also removed from memory, provided that no other users are using the dataset. Therefore, deleting a dataset reference might cause the cluster to release memory.

Steps for deleting a dataset reference

Follow these steps to delete a dataset reference:

1. In the **Dataset Refs** list, navigate to the dataset reference that you want to delete. It can be located in your folder or another user's folder.
2. Right click the icon or name of the dataset reference.
3. In the drop-down menu, click **Delete**.
4. When a confirmation message is display, click **CONFIRM** if you are sure that you no longer need the dataset reference.

What you cannot do to other users' dataset references

You can see other users' dataset references. You can preview and create tables from these dataset references. However, you cannot perform the following tasks if you are not the owner of the dataset reference:

- Renaming the folder containing the dataset references
- Moving the dataset reference to another folder
- Creating a folder in another user's dataset reference folder

Effect of cluster restart on dataset references

After a cluster is restarted, some dataset references previously in your list of dataset references might no longer be displayed. Only the dataset references used in your active workbook are shown immediately after the restart.

> **NOTE**: The datasets and the dataset references still exist; they are available for you to use again when needed.

A dataset reference appears in the list again when:

- you activate a workbook that uses the dataset reference, or
- a batch dataflow is run against the dataset

> **EXAMPLE**: You create a dataset reference called airlines, which is used in workbook1, and a dataset reference called carrier, which is used in workbook2. Both dataset references are visible in the list of dataset references, regardless of which workbook is active. After a cluster restart, if workbook1 is active, only the dataset reference named airlines is displayed. To also display the dataset reference named carrier, activate workbook2.

Creating a table

This section assumes that you have created one or multiple dataset references. Also, this section assumes that you are viewing the worksheet where you want to create the table. See Understanding workbooks and worksheets for information about how to switch between worksheets.

You can create as many tables as you want from one dataset reference.

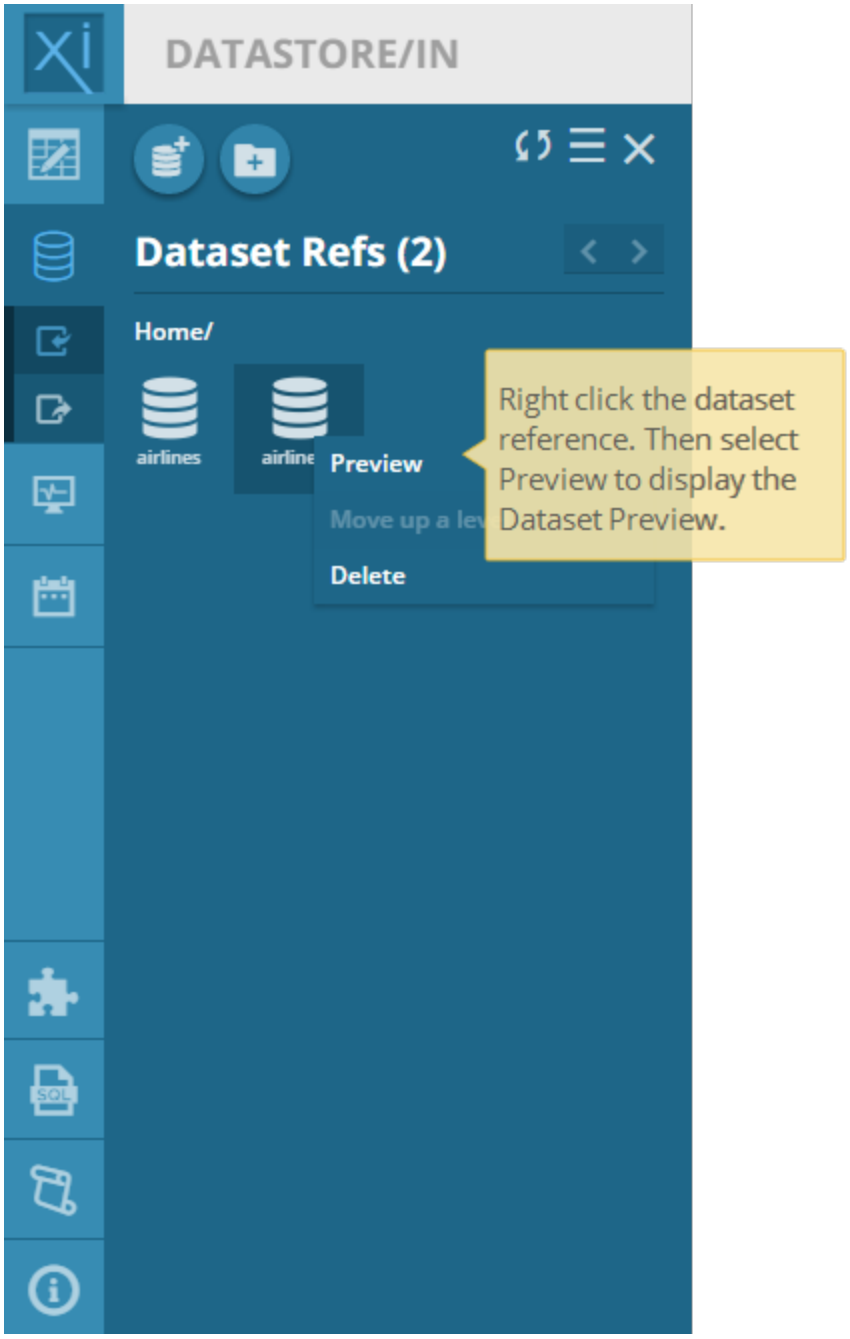
> **NOTE:** Tables can be created as a result of table operations. For example, joining two tables results in a new table. Running a batch dataflow, which is an advanced feature, can also create a table. This section is about creating a table by selecting fields in the data source.

Displaying the Dataset Preview window

The **Dataset Preview** window is displayed automatically after you point to a new data source. Alternatively, follow these steps to display the **Dataset Preview** window:

1. Click  (**Data Stores** icon).
2. In the **Dataset Refs** panel, right click a dataset reference and then select **Preview** in the

pop-up menu, as shown in the following screenshot:

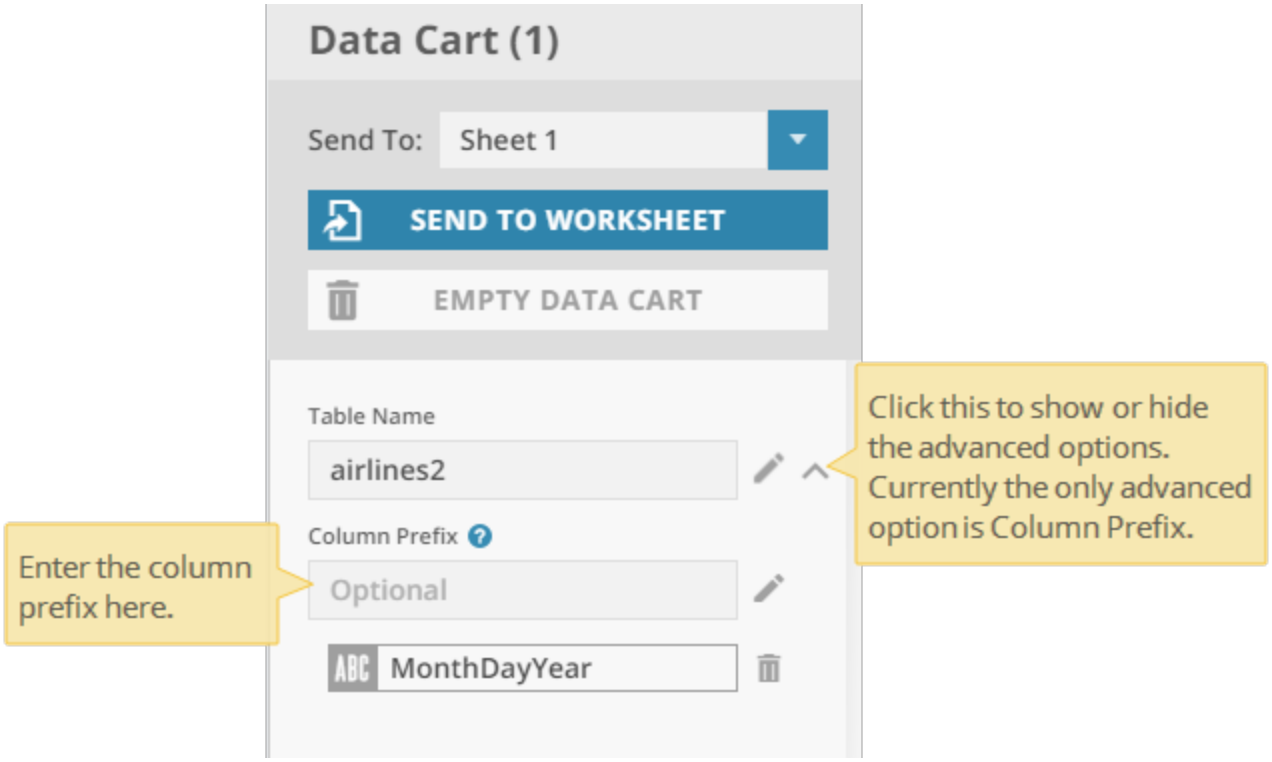


Create a table from the columns in the Dataset Preview window

In the **Dataset Preview** window, follow these steps to create a table:

1. Follow one of these steps:
   - Click one or multiple column headings to select the columns that form the table. Clicking a selected column heading deselects the column.
   - Click **Select All** if you want all columns to be in the table.
   - Create a table without any columns. You can select the columns after the table is created.
2. In the right panel (**Data Cart** panel), select the worksheet to which the selected columns will be sent.
3. (Optional) In the **Data Cart** panel, edit the table name. A default table name, based on the dataset reference name, is provided.

   The following list describes the naming conventions for a table:
   - Names are case-sensitive.
   - Length can be from 1 to 255 characters.
   - Characters can be A-Z, a-z, 0-9, hyphen (-), and underscore (_).
   - The name must not be a duplicate of an existing table name.
   - First character must be a letter.
4. (Optional) Display the advanced options as shown in the following screenshot to enter the column prefix. If you do not specify a prefix, Xcalar Insight uses the table name as the prefix.



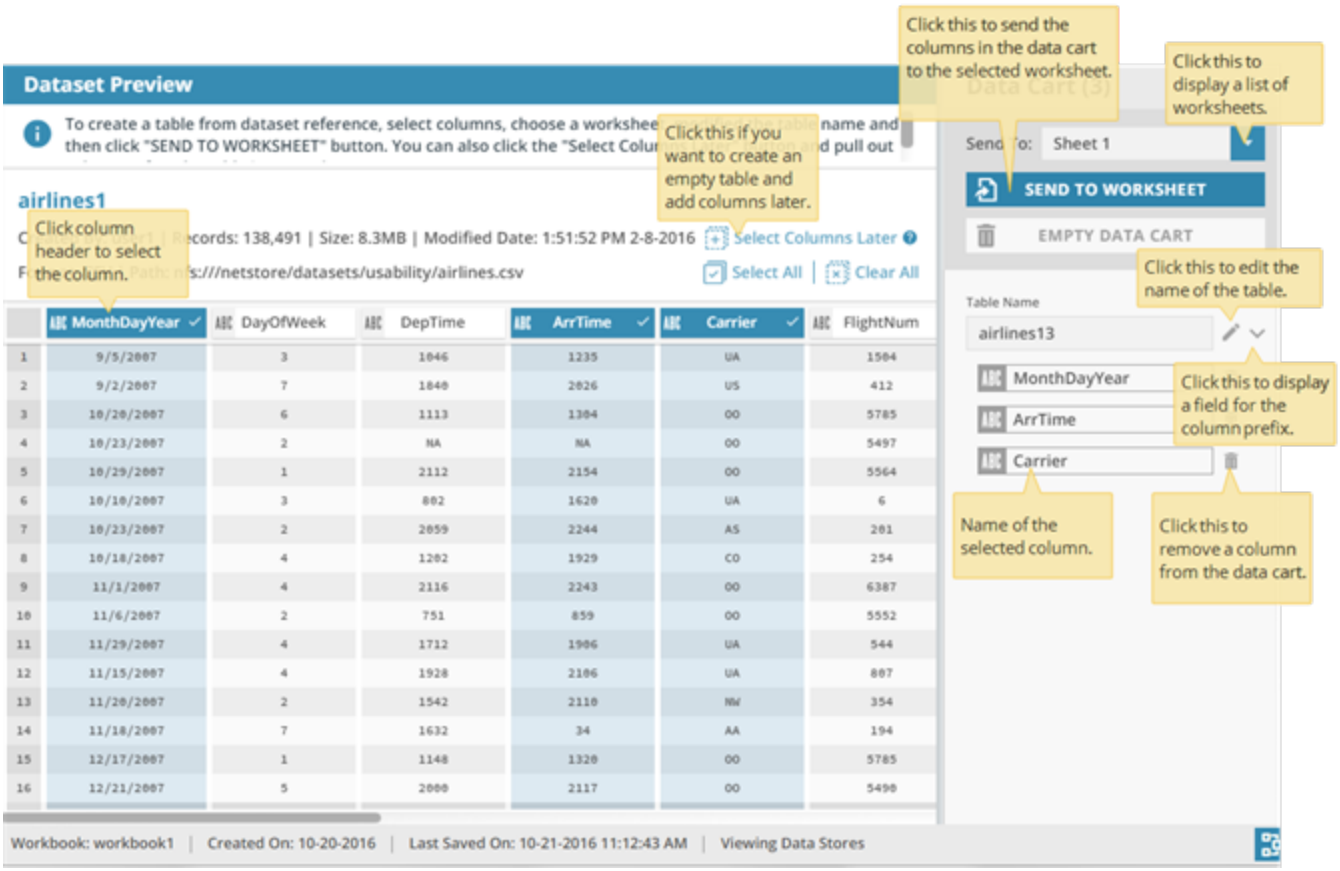
A column prefix is used to distinguish columns with the same name in the same table. Typically, column name conflicts occur due to a Join operation. If the left and right tables have the same name for a column, the prefixes (if different) help you identify the origin of each column in the resultant table. (If the prefixes of the tables being joined happen to be the same, you are prompted to change the prefix before starting the Join operation.) You can change a prefix at table creation time or when preparing for a Join operation. You cannot change the prefix at other times.

The following list describes the naming conventions:

- Names are case-sensitive.
- Length can be from 1 to 255 characters.
- Characters can be A-Z, a-z, 0-9, hyphen (-), and underscore (_).

5. If you want to remove columns already in the data cart, click the **Delete** icon to remove a column, or **EMPTY DATA CART** to remove all columns from the data cart.
6. Click **SEND TO WORKSHEET**. All columns shown in the data cart are sent to the same worksheet.

The following sample screen shows the **Dataset Preview** window and the **Data Cart** panel.

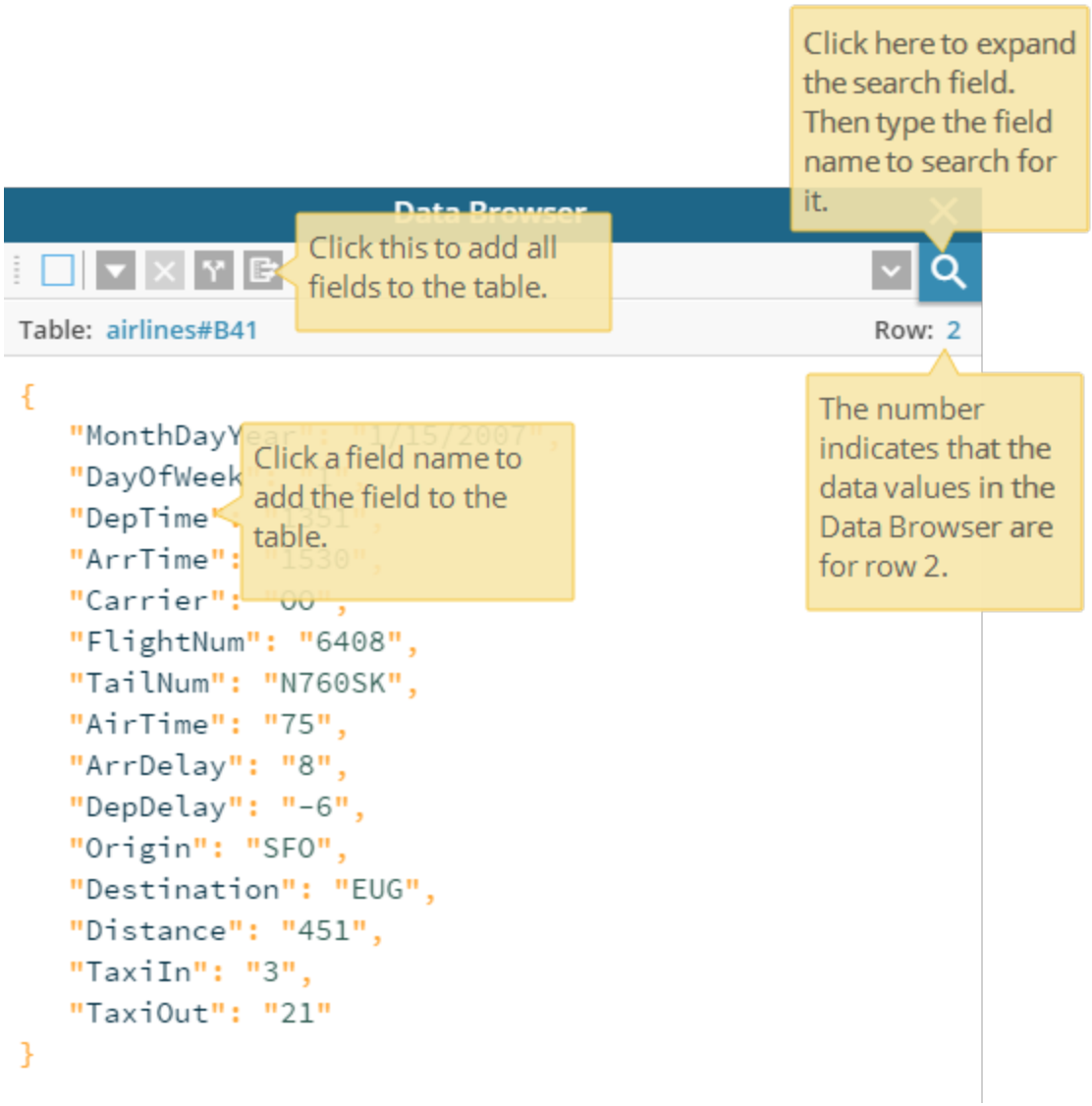


For more information about the worksheet window after you create a table, see Worksheet window with tables. You are now ready for transforming your data by using the various functions in Xcalar Insight.

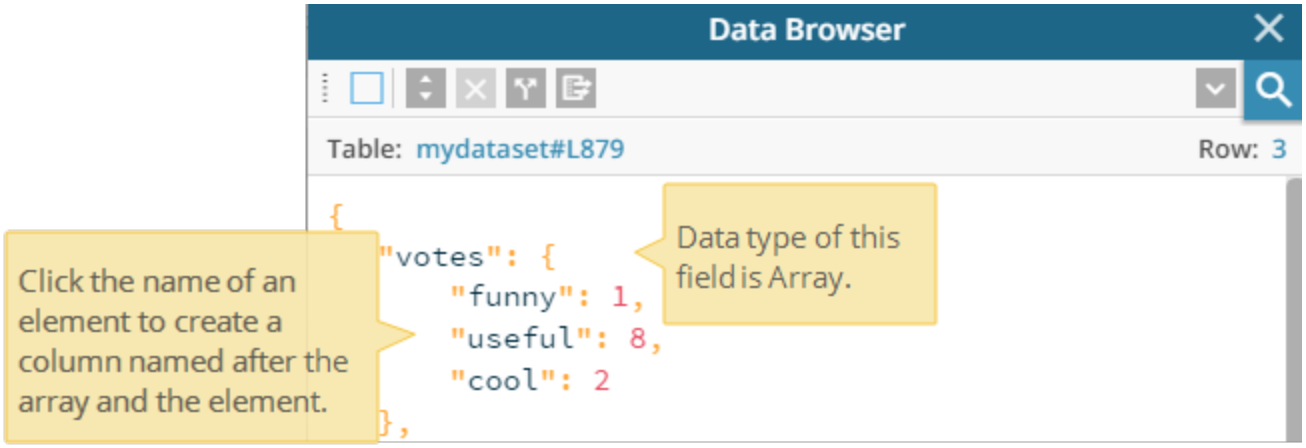
Adding columns from a dataset to a table

This section assumes that you have created a table. Follow these steps to add columns to a table:

1. Open the worksheet containing the table where you want to add a column.
2. In the table, double click any row in the **DATA** column to open the **Data Browser**. The browser displays all fields in the dataset and their values in the row where you open the browser.
3. Select a field to add to the table in one of the following ways:
   - Click a field name to select it.
   - Search for a field to locate it. Then click the field name to select it.
   - Add all fields to the table. The existing fields in the table are not duplicated. Only the ones that are currently not in the table are added.



4. If the data type of a field is Array, you can add an element of the array to the table as a column. The following partial screenshot shows the **Data Browser** containing an array. In this example, a column named votes.useful of the type Integer is created.



> **NOTE:** If you click a field that is already in a table, the **Data Browser** closes and the table is scrolled to the column, which is highlighted.

The **Data Browser** has other features such as column comparison among tables in the same or different worksheets. See Data Browser for more information about these features.

Changing the data type for a column

After you create a dataset reference to point to a data source, you can create tables from the fields in the dataset. Each column in the table created is assigned a data type, which can be one of the following:

- Array
- Boolean
- Float
- Integer
- Mixed
- Object
- String
- Unknown

A column's initial data type is Mixed if there are different types of data in the column. A column's initial data type is Unknown if Xcalar Insight cannot determine the data type or if all the values in the column are null or FNF (field not found).

> **NOTE:** A table cell contains FNF when the row does not have a column that other rows in the table have. For example, if a column named EmployeeID exists for all rows except row 1, the value on row 1 under the EmployeeID column header is FNF.

> **IMPORTANT:** The initial data type for a value generated by a UDF is always String. If your UDF produces a value that needs to be a type other than String, it is important to follow the instructions in this section to change its data type before performing operations on the value.

Restrictions on data type changes

Remember the following restrictions when you attempt to change a column's data type:

- Changing a column's data type to Array, Object, or Unknown is not allowed.
- You can change the data type for a column only if the column contains values. If you create a column with the **Add a column** option in the column drop-down menu, you cannot set the data type for the new, empty column.

Supported integer types and floating point values

The following integer types are supported:

- Signed and unsigned 32-bit integers
- Signed and unsigned 64-bit integers

Both 32-bit and 64-bit floating-point values are supported.

Different ways to change the data type

You can use one of the following methods to change the data type:

- Use the **Change data type** option in the column drop-down menu. This method changes the data type for one column, and does not create a column.
- Select the **Map...** option in the column drop-down menu. This method keeps the existing column intact and creates a column with the new data type.
- Use the **Smart Type Casting** feature. This method changes the data type for one or multiple columns. It does not create new columns.

Using the **Change data type** option

Follow these steps to change the data type for one column:

1. Right click the column header to display the column drop-down menu.
2. Select the **Change data type** option in the column drop-down menu.
3. Select the data type in the menu.

Using the Map function to change the data type

After you select the **Map...** option in the column drop-down menu, the **Map** panel is displayed.

The following list describes the general rules and tips for entering information in a Xcalar Insight panel.

- By default, the name of the column from which you start the function is displayed as an argument. If you want to use another column as an argument, you can change the argument to the name of another column.
- When typing a column name, always start with the dollar sign ($).
- When typing a column name, always include the prefix if one exists. (A derived column does not have a column name prefix.)
- If the panel calls for the resultant column name, type the name without preceding it with the dollar sign ($).
- To avoid mistakes when typing a column name as an operand, place the insertion point in the argument field, move your mouse to the active table and click anywhere in the desired column. The column's name is automatically entered in the operand field.
- If an operand accepts a column name, it also accepts an aggregate name that you created when you calculated the aggregate value for a column.
- If necessary, click in the panel to bring the specified table into focus. You can then scroll the table to make the desired column visible.
- You can undo any typing with the undo key or key combination that you normally use in your browser.

Follow these steps to create a column with the desired data type:

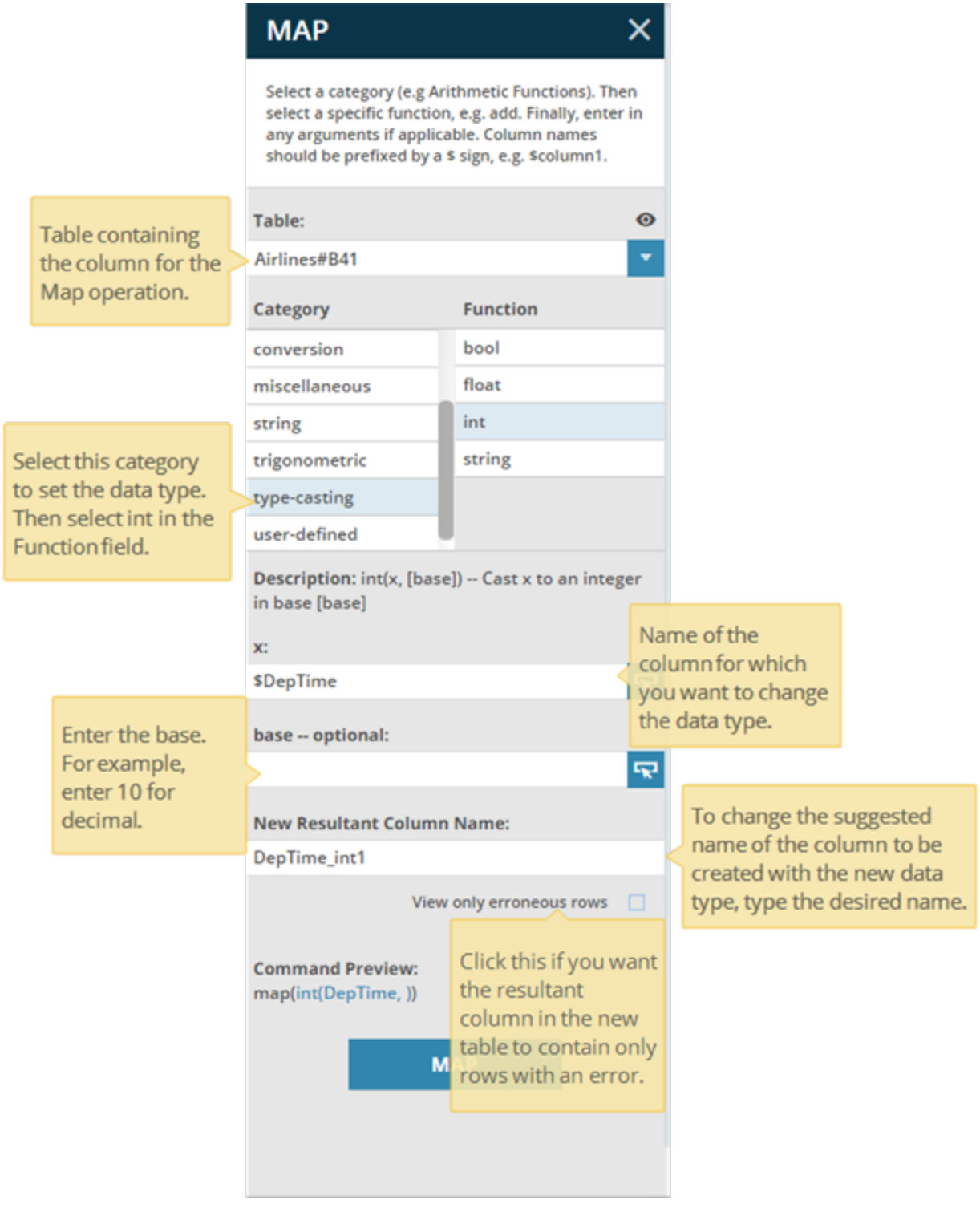
1. In the **Table** field, if the desired table name is not already displayed, click the down arrow to display a list of tables and then select the desired table.
2. In the **Category** field, select **type-casting**.
3. In the **Function** field, select the desired data type.
4. The first argument (x) is the active column name by default. You can accept the default or enter the name for another column whose data type will be changed.

   If the desired data type is integer, enter the base as the second argument. For example, for base 10, type **10**; for binary, type **2**.

5. Type a column name in the **New Resultant Column Name** field.
6. If you are interested only in the rows where the **Map** function fails, select **View only erroneous rows**. For example, if you want to know if there are values in a particular column that cannot be type cast into Integer, select this option. The resultant table will contain only rows with data in this column that cannot be changed by the Map function.
7. Click **Map**. A new column is created with the new data type.

> **NOTE:** If you did not enter the second argument, an error message is displayed, requiring you to click the **No Argument** check box. After you click the check box, click Map.

The following sample **Map** panel illustrates how you can change the data type to integer.



Using the Smart Type Casting feature

Xcalar Insight provides data type suggestions for columns in your table. Follow these steps to change the data type for one or multiple columns:

1. Right click the table title to display the drop-down menu.
2. Select **Smart Type Casting...**. The **SMART CAST** panel is displayed.
3. The **SMART CAST** panel shows the data types suggested for particular columns. To change a suggestion, click a data type and select a new data type from the pop-up menu.
4. (Optional)For any column that is not in the **SMART CAST** panel, you can click anywhere in the column to place it in the panel. Then click the data type for the column and select a new data type from the pop-up menu.

   If for any reason you want to remove all columns and their data types in the panel, click **CLEAR ALL**.
5. Click **CAST**.

> **TIP:** To display the suggestions again (for example, after you edited data types for several columns), click **DETECT**. Remember, however, that doing so removes columns that you added to the panel. The **SMART CAST** panel now contains only columns with data types suggested by Xcalar Insight.

Counting occurrences of unique values in a column

Follow these steps to count the occurrences (frequency) of each value in a column:

1. Right click the header of the column in which you want to count the occurrences of its values.
2. From the drop-down menu, select **Profile...**. The **Profile** window containing a bar graph is displayed. The bar graph shows the distribution of values in the column.

> **TIP:** It might take a while for the bar graph to complete. You can close the window and re-select **Profile...** in the column drop-down menu later to view the finished bar graph.

> **NOTE:** Using **Profile** to determine the frequency of each value does not create a table. If you need to create a table to show the frequency of each value, use the Count function, which is one of the Group By functions. For more information about Group By functions, see Grouping data in a table.

Sorting the frequency

You can sort the bars in ascending or descending order. Or you can keep the original order.

> **EXAMPLE:** Suppose the values in the selected column represent days of the week, the bar graph is originally sorted by day1, day2, day3, and so on. You can sort the bars based on the frequency of each value. If you sort in ascending order, the value with the lowest frequency is displayed first.

Displaying frequency for a range or a single value

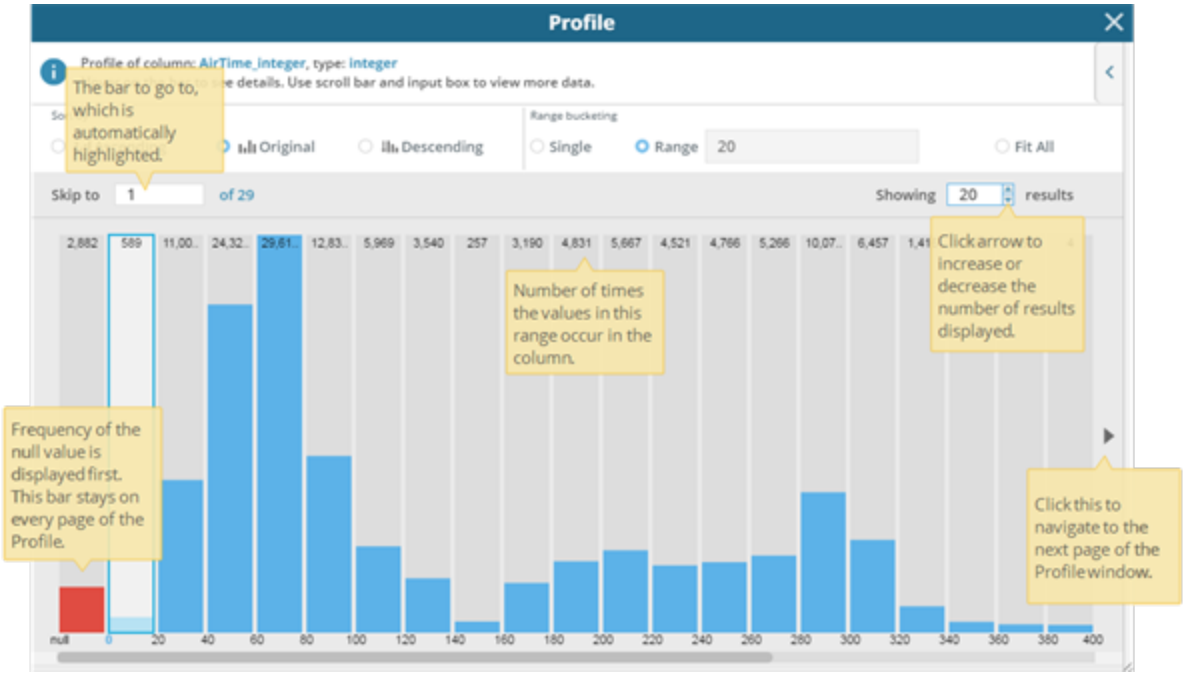
If the data type of a column is Integer or Float, you can use the **Range Bucketing** section to determine how the bars are displayed as described in the following list:

- Select **Single** to count occurrences of single values, in which case each bar represents the frequency of one value. If the data type is not Integer or Float, you can display bars for single values only.
- To count the occurrences of values that fall in a range, select **Range** and enter the range size. Each range is represented by a bar in the graph.
- If you want Xcalar Insight to choose the range size such that all bars can fit in the window, select **Fit All**.

> **NOTE:** A bar is displayed only if there is at least one occurrence of the value represented by the bar. For example, if the range size is 50, and there are no occurrences of any value in the range of 0 to 50, the graph does not display a bar for this range.

Controlling the number of results displayed

If you select **Fit All**, you cannot control the number of results displayed. Otherwise, you can increase or decrease the number of results displayed. The number is incremented by 10. The following screenshot shows where you can control the number of results.



> **NOTE:** The bar showing the frequency of the null value is always displayed first. It is not counted in the number of results shown. The number of results reflects only the bars for the non-null values.f

Displaying counts or percentages

By default, the frequency is displayed as a count at the top of each bar. Click the bar to change the count to a percentage.

If the number or percentage is not completely visible because it is wider than the bar, hover over the bar to display a tooltip, which contains the field name, the value (either as a single value or as a range), and the frequency.

> **EXAMPLE:** Suppose the value 1 in the selected column appears 3,000 times, the count (3,000) is displayed on the bar. Click anywhere in the bar and the count changes to 20%, which means that 20% of the values in the column are 1.

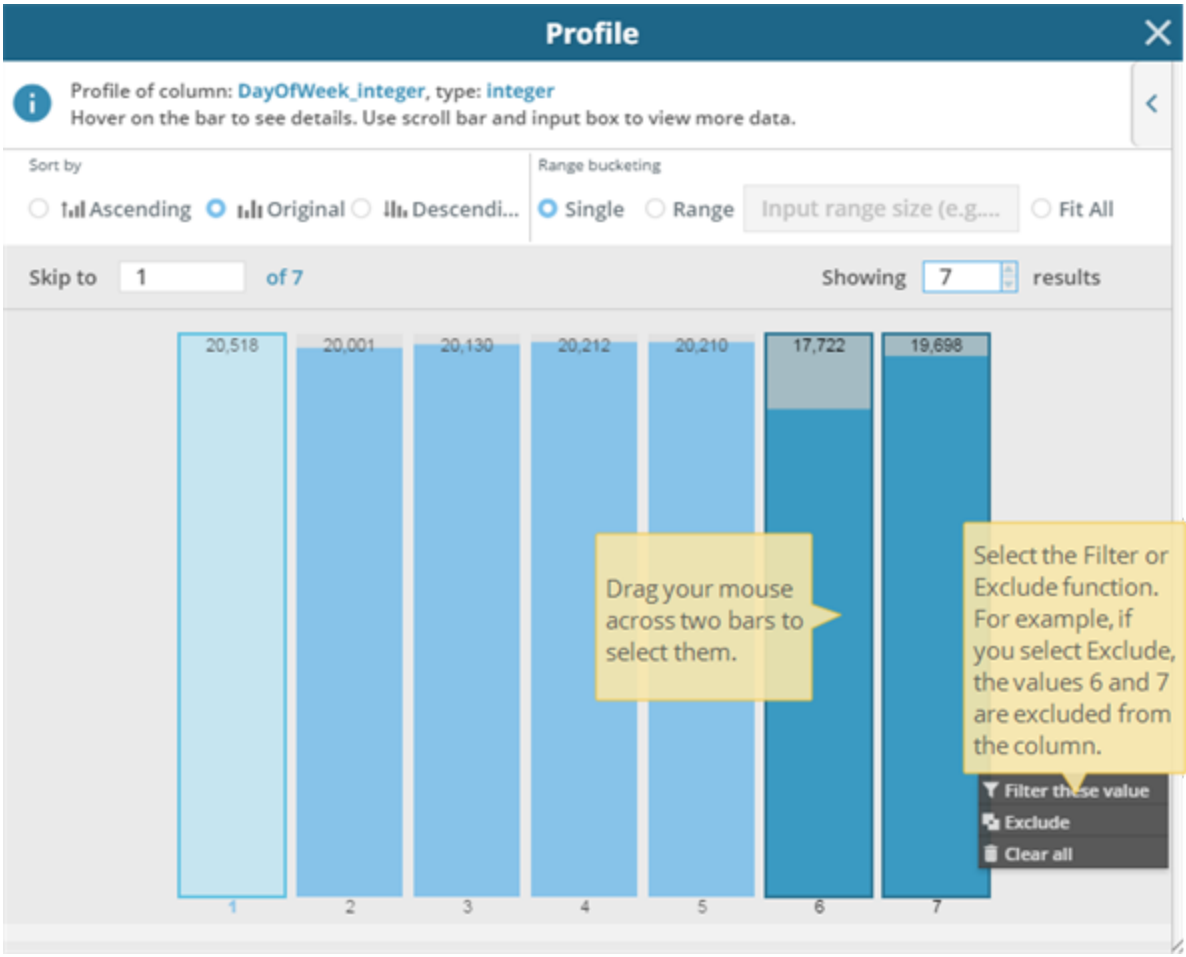
Filtering by using the Profile window

To filter or exclude a value in a column, follow the instructions in Filtering in a table column. In addition, you can perform filtering after viewing the frequencies of various values in a column. The **Profile** window provides a convenient way to filter or exclude a value in the table. You can also filter or exclude a range of values.

Follow these steps to filter or exclude a value through the **Profile** window:

1. Place the cursor in the bar for the value to be filtered or excluded. Drag in any direction inside the bar. When you release the mouse button, a pop-up is displayed. To select multiple bars in the graph to filter or exclude more values, drag the mouse across the bars. You can select only contiguous bars.
2. Select **Filter** in the pop-up to include only this value (or a range of values) in the column. Select **Exclude** to exclude this value (or a range of values) in the column. The **Profile** window is closed after you make the selection, and the column in the table is updated to reflect the filter operation.

The following screenshot provides an example showing how to filter or exclude two values.



Undoing and redoing an operation

You can undo and redo an operation in one of two ways.

Using the Undo and Redo buttons

The  (**Undo**) and  (**Redo**) buttons are located in the upper left corner of the **Worksheet** window. When you hover over the button, a text string is displayed to indicate what operation will be undone or redone.

The **Undo** and **Redo** buttons work for operations that are internal to a table or operations that result in a new version of a table.

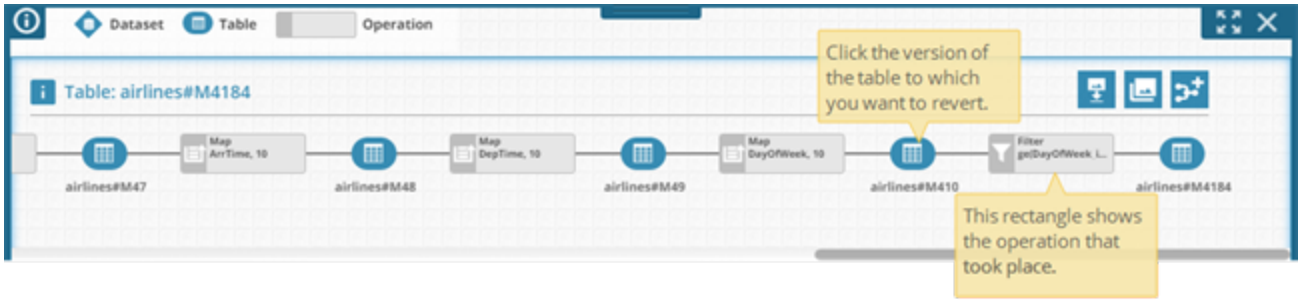
> **EXAMPLE:** After you delete a column, you can undo the deletion by clicking the **Undo** button.

> **EXAMPLE:** After you perform a filter on a particular value in a column, a new version of the table is created with only the rows containing that particular value in the column. To revert to the version of the table before the filter operation, click the **Undo** button.

Using the dataflow graph

If an operation results in a new version of the table, you can use the dataflow graph to revert to a previous version. The following example illustrates how to undo the filter operation:

1. Click  (**dataflow graph** icon) in the lower-right corner of the Xcalar Insight window.
2. In the dataflow graph, click the table before the filter operation. A list of options is displayed in a pop-up menu. Click **Revert To This Table**.



3. To hide the dataflow graph, click again or click the **Close** button in the upper right corner of the dataflow graph panel.

Filtering in a table column

> **TIP:** You might want to count the occurrences of a value before filtering it. See Counting occurrences of unique values in a column for information about counting the number of times a value appears in a column. You can then use the **Profile** window to perform the filtering.

If you want to filter a value in a table column, follow these steps:

1. Click a value in a column that you want to filter. For example, in a flight information table, if you are interested in flights with a departure time at 740, in the **DepTime** column, click **740**. The value you click can be in any row.
2. In the drop-down menu, select **Filter this value**. A new version of the table is created with rows containing 740 as the departure time. All other rows are deleted.

> **NOTE:** If you want to exclude the value, click **Exclude this value**. For example, excluding 740 means that the new version of the table contains all rows with a departure time other than 740.

Filtering based on conditions

If you want to filter based on a condition (for example, whether a string value contains a substring or whether an integer value falls in a particular range), follow these steps:

1. Right click the heading of the column containing the value to filter. A drop-down menu appears.
2. Select **Filter...** from the menu. The **FILTER** panel appears.

   The following list describes the general rules and tips for entering information in a Xcalar Insight panel.

- By default, the name of the column from which you start the function is displayed as an argument. If you want to use another column as an argument, you can change the argument to the name of another column.
- When typing a column name, always start with the dollar sign ($).
- When typing a column name, always include the prefix if one exists. (A derived column does not have a column name prefix.)
- If the panel calls for the resultant column name, type the name without preceding it with the dollar sign ($).
- To avoid mistakes when typing a column name as an operand, place the insertion point in the argument field, move your mouse to the active table and click anywhere in the desired column. The column's name is automatically entered in the operand field.
- If an operand accepts a column name, it also accepts an aggregate name that you created when you calculated the aggregate value for a column.
- If necessary, click in the panel to bring the specified table into focus. You can then scroll the table to make the desired column visible.
- You can undo any typing with the undo key or key combination that you normally use in your browser.

3. Select the filter function. For example, for a column of data type Integer or Float, you can select the **between** function to filter values that exist between two values.

   The following functions are supported:

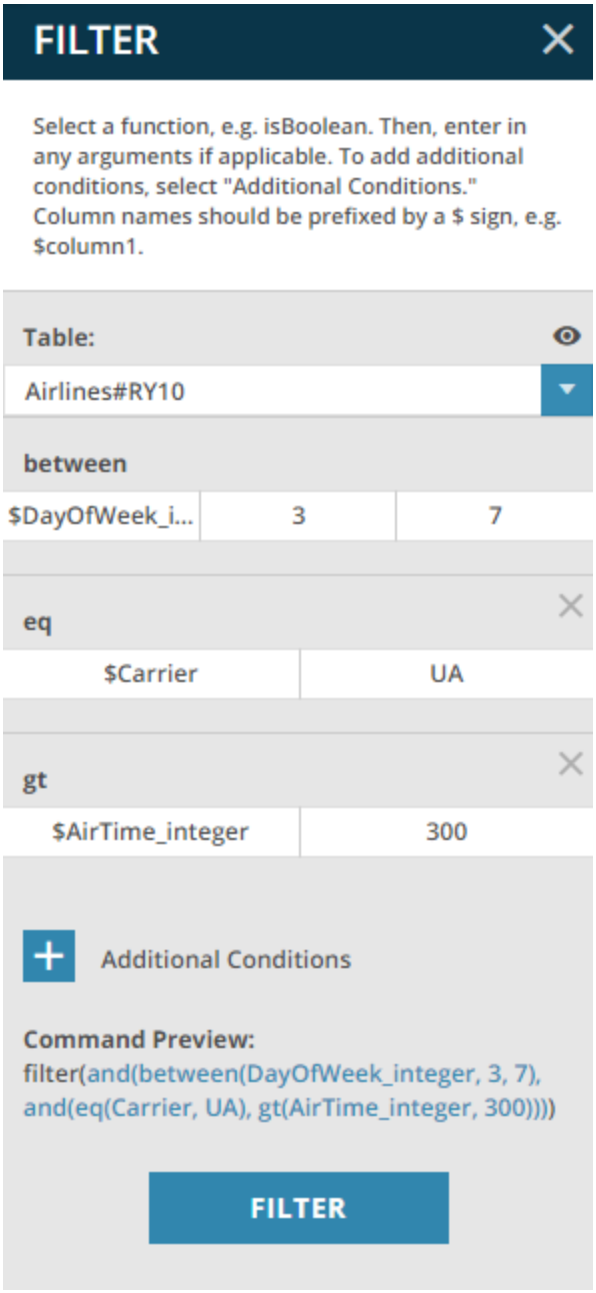
   - and
   - between
   - contains
   - eq
   - exists
   - ge
   - gt

- if
- ifInt
- ifStr
- isBoolean
- isFloat
- isInteger
- le
- like
- lt
- not
- or
- regex

> **NOTE:** If you select a filter function that is not applicable to the data type of the column, an error message is displayed. For example, if the data type of the Distance column is String, and you try to run the gt (greater than) function on the column, an error message is displayed. You can change the data type in the **Cast** field to either Integer or Float, which enables the function to run properly. However, this data type change is temporary. The column in the table (the Distance column) continues to be of data type String.

4. Based on the description of the function, fill out the fields.
5. Click **FILTER**. Only rows that meet all specified filter conditions are kept in the table.

> **EXAMPLE:** In the table for airlines information, you can filter days between 3 and 7 in the **DayOfWeek** column, the carrier UA in the **Carrier** column, and air time that is greater than 300 minutes in the **AirTime** column as illustrated in the following screenshot. The resultant table contains only rows that meet all these conditions.



Creating an integrity constraint violations table

Tables created in Xcalar might contain integrity constraint violations. In Xcalar Insight, a table containing only rows with integrity constraint violations is called an ICV table.

The following list describes the typical scenarios when integrity constraint violations occur:

- Integrity constraint violations occur when the values are inserted or updated in the data source before Xcalar Insight points to it. The dataset might include erroneous values in a particular field. For example, in a dataset for flight information, the Airtime field, which should contain numeric values, might contain a string value such as 5O (instead of 50) due to a data entry error.
- A Xcalar Insight function cannot operate on a value in a table cell. The cell in the resultant ICV table displays a message about the cause of the violation.

  For example, in an integer column, there are rows containing FNF (field not found). If you use the Map function to increment values in the column, the ICV table displays the following message in each row with FNF:

  ```
  Some variables are undefined during evaluation
  ```

  Similarly, if you use the div function and the divisor is 0, the row containing the resultant value is also considered an ICV row. The ICV table displays the following message in each row where the division cannot be performed:

  ```
  Divide by zero error
  ```

Creating an ICV table

Rows containing integrity constraint violations might be of interest because you can gain insights from erroneous data (for example, by finding unexpected values in an analysis of data for fraud detection) or because you want to filter out data that exist due to errors.

Functions for creating an ICV table

The Map and Group By functions enable you to create an ICV table. This means that when you use the Map or Group By function to create a table, you can specify that the table includes only

rows with integrity constraint violations.

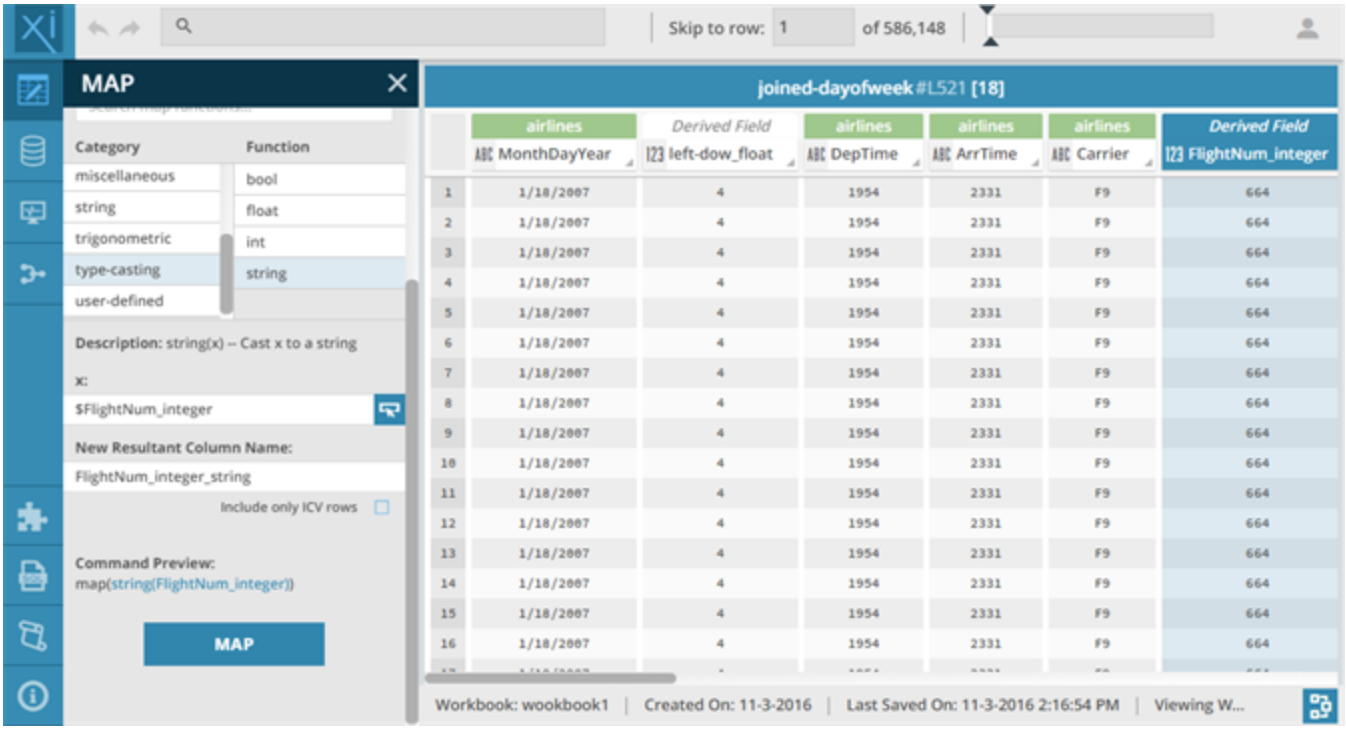
This section assumes that you understand how to use the Map and Group By functions in Xcalar Insight. If not, familiarize yourself with these functions by reading the following topics:

- Using the Map function to create new values
- Grouping data in a table.

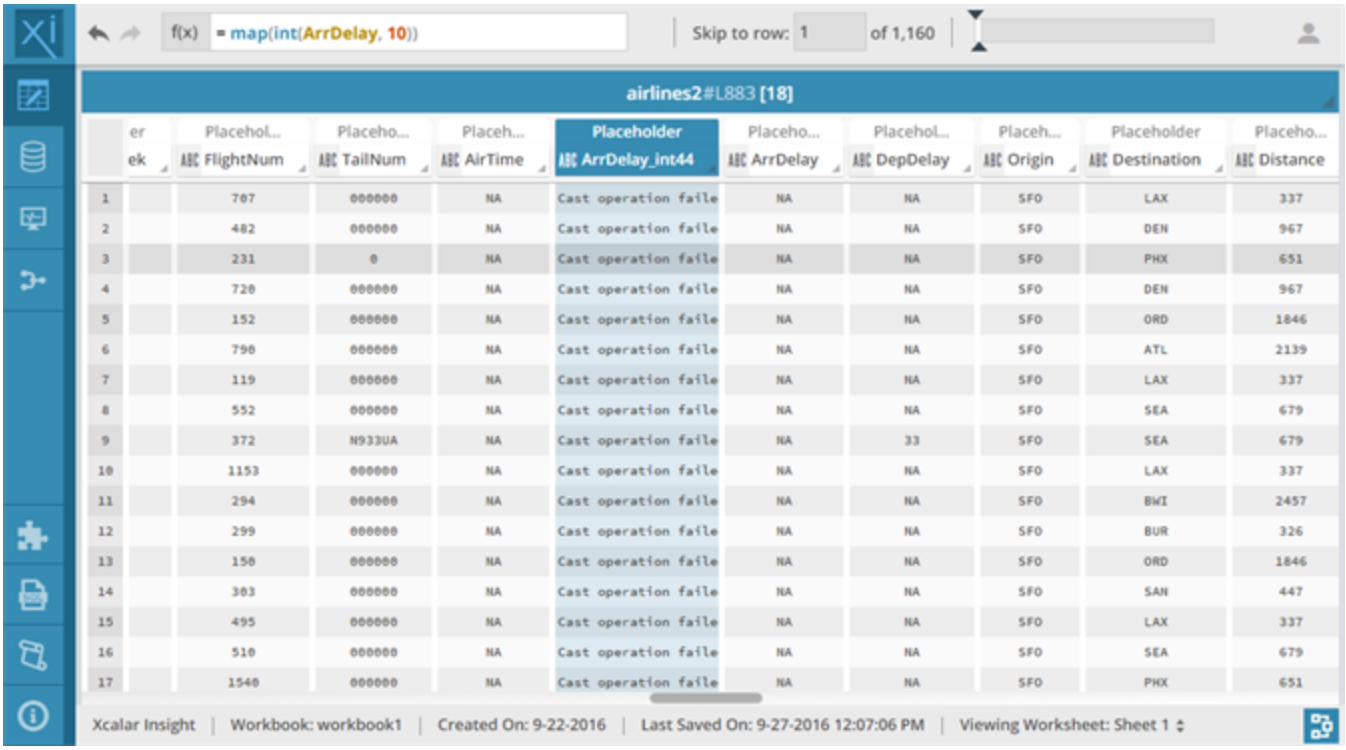
Example of a Map function for creating an ICV table

In the table created from a dataset with flight information, the column named ArrDelay contains the number of minutes for each flight's arrival delay. Suppose the initial data type for this column is String, you can use the **type-casting** function in the **MAP** panel to change its type to Integer. If you select the **View only erroneous rows** in the **MAP** panel, the new table contains only the rows with integrity constraint violations in the resultant column.

The following screenshot illustrates the **MAP** panel for changing the data type of a column.



The following screenshot illustrates the resultant table containing only rows with integrity constraint violations in the ArrDelay column.

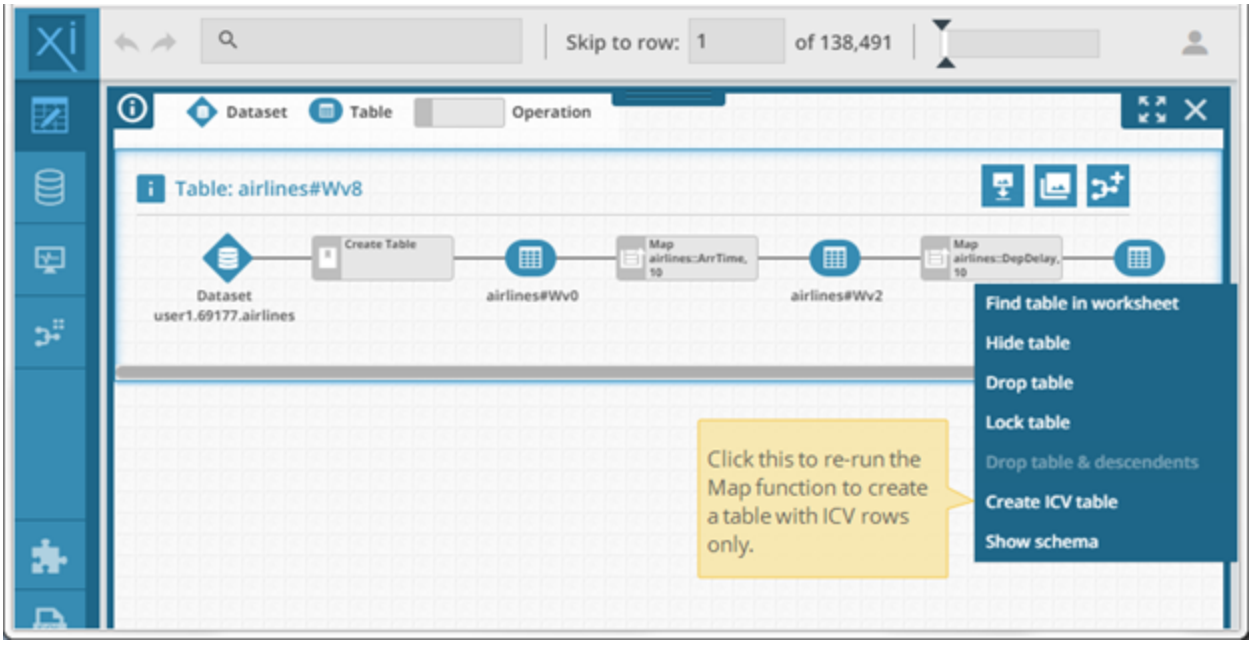


	er ek	Placehol... FlightNum	Placeho... TailNum	Placeh... AirTime	Placeholder ArrDelay_int44	Placeho... ArrDelay	Placehol... DepDelay	Placeh... Origin	Placeholder Destination	Placeho... Distance
1		707	000000	NA	Cast operation faile	NA	NA	SFO	LAX	337
2		482	000000	NA	Cast operation faile	NA	NA	SFO	DEN	967
3		231	0	NA	Cast operation faile	NA	NA	SFO	PHX	651
4		720	000000	NA	Cast operation faile	NA	NA	SFO	DEN	967
5		152	000000	NA	Cast operation faile	NA	NA	SFO	ORD	1846
6		790	000000	NA	Cast operation faile	NA	NA	SFO	ATL	2139
7		119	000000	NA	Cast operation faile	NA	NA	SFO	LAX	337
8		552	000000	NA	Cast operation faile	NA	NA	SFO	SEA	679
9		372	N933UA	NA	Cast operation faile	NA	33	SFO	SEA	679
10		1153	000000	NA	Cast operation faile	NA	NA	SFO	LAX	337
11		294	000000	NA	Cast operation faile	NA	NA	SFO	BWI	2457
12		299	000000	NA	Cast operation faile	NA	NA	SFO	BUR	326
13		150	000000	NA	Cast operation faile	NA	NA	SFO	ORD	1846
14		303	000000	NA	Cast operation faile	NA	NA	SFO	SAN	447
15		495	000000	NA	Cast operation faile	NA	NA	SFO	LAX	337
16		510	000000	NA	Cast operation faile	NA	NA	SFO	SEA	679
17		1540	000000	NA	Cast operation faile	NA	NA	SFO	PHX	651

Re-running a function in a dataflow to create an ICV table

This section assumes that you are familiar with the dataflow graph in a worksheet. For information about dataflows, see Dataflow graph.

In a dataflow graph, you can create an ICV table through a Map or Group by function previously executed. For example, in the following dataflow graph, you can click the icon for the airlines3 table to display a pop-up menu. In the menu, you can select **Integrity Constraints Table** to create a table with the same Map function shown in the dataflow. This method generates a table in the same way as selecting **Include only ICV rows** in the **MAP** panel.

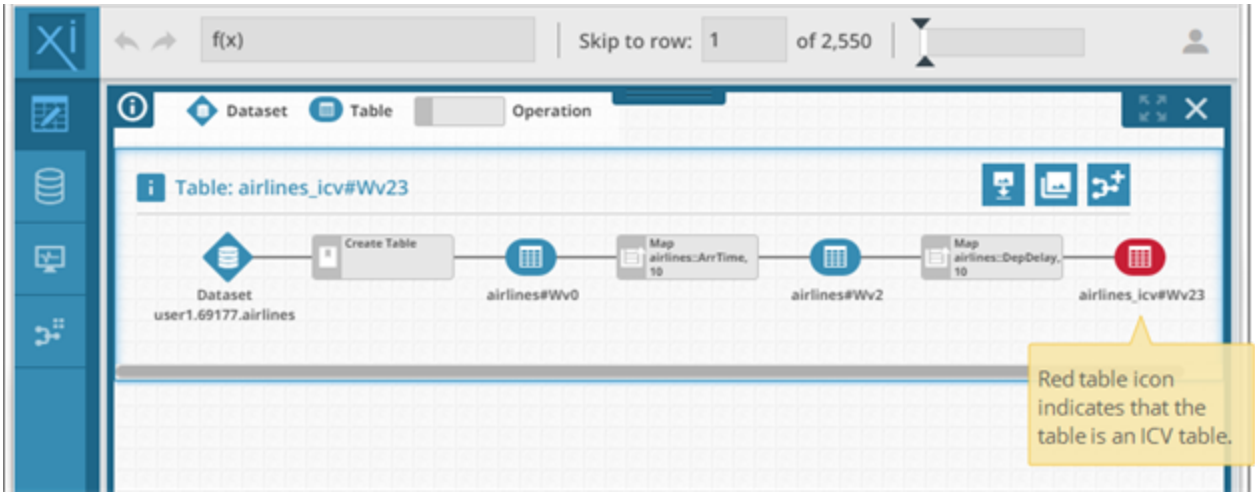


The following list describes the results of creating an ICV table from the dataflow graph:

- A new table is created by the Map or Group By function. In this example, the new table contains only rows with integrity constraint violations in the Distance_integer column.
- The new table is an active table with its own dataflow graph. The table icon in the graph is red to indicate that the table is an ICV table.
- The dataflow graph from which the new table is created stays in the worksheet and remains unchanged.

> **NOTE:** From the dataflow graph, you can create an ICV table from any table produced by a Map or Group By function. It can be a temporary or active table.

The following screenshot shows the dataflow graph after the ICV table is created.



Joining tables

You can join tables in the same worksheet. The Join function works in the same way as the SQL join clause.

> **NOTE:** You can join two tables or join a table to itself (that is, perform a self join). You cannot use one operation to join more than two tables.

By default, the Join function uses one key from each table for joining. That is, you can select one column from each table by which the tables are joined.

You can, however, perform a multi-clause join, which is a Join function that uses pairs of join keys.

Follow these steps to join tables:

1. Right click the header of a column in one of the tables that you want to join. A drop-down menu is displayed.
2. Select **Join...** in the menu. The **JOIN** panel appears.
3. Select the type of Join function to perform. By default, **Inner join** is selected as the type of Join operation. The following types of Join operations are supported:
    - Inner join
    - Left outer join
    - Right outer join
    - Full outer join
4. In the **Right Table** field, click the **Down** arrow to display a list of tables. Then select the table to join to the left table.
5. To select the columns to join, you can either click **SMART SUGGEST** to display column names suggested by Xcalar Insight or fill out the **Left Columns** and **Right Columns** fields.

(Optional) For a multi-clause join, click **ADD ANOTHER CLAUSE** to add another pair of columns to join.

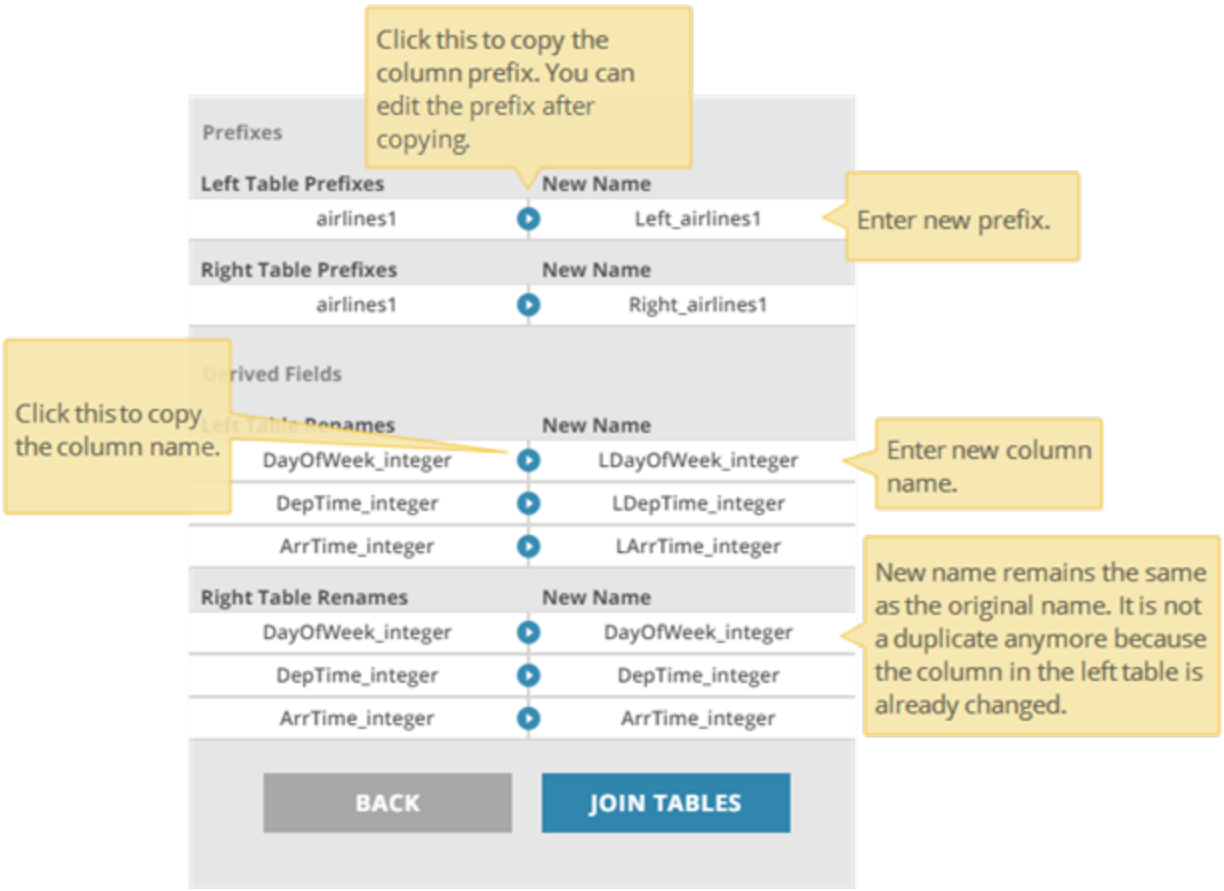
Verify the table names and column names in the **Command Preview** section.

6. Click **NEXT**.
7. Enter the new table name.
8. In the **Column Selector** section, select the columns from the left and right tables that are included in the resultant table.
9. (Optional) Select **Keep original tables** if you want both tables to remain on the worksheet. If you want to keep only the resultant table, leave this option unchecked.
10. (Optional) Check the number of rows in the **Estimated join size** section. Xcalar Insight provides the minimum, maximum, and median number of rows in the resultant table. The time required to complete the Join operation is proportional to the number of rows created. Based on the estimates, you can decide whether you should perform the operation at this time.
11. Click **JOIN TABLES**. A new table is created as a result of joining the tables with the keys you selected.

> **NOTE:** If there are duplicate column names in the resultant table, you are prompted to rename the columns before the Join operation takes place.

Renaming columns to avoid column name duplication

The following partial screenshot of the **JOIN** panel illustrates how you rename columns to avoid column name duplication in the resultant table created by a self join operation.



Fields that exist originally in the data source can be distinguished through column name prefixes. In this example, you can change the prefix for the left table from airlines1 to Left_ airlines1 and the prefix for the right table from airlines1 to Right_airlines1. The resultant table will have columns named Left_airlines1::Carrier and Right_airlines::Carrier, which no longer conflict with each other.

The following list describes the naming conventions:

- Names are case-sensitive.
- Length can be from 1 to 255 characters.
- Characters can be A-Z, a-z, 0-9, hyphen (-), and underscore (_).

Because derived fields do not have prefixes, you can rename the column to avoid the name duplication. In this example, the column DayOfWeek_integer in the left table can be renamed to LDayOfWeek_integer. The column DayOfWeek_integer in the right table can be either renamed or remain unchanged.

The following list describes the naming conventions for a table column:

- The name can consist of any alphanumeric characters, space, and special characters except the following characters:
  - single quote mark (')
  - double quote mark (")
  - parenthesis (( ))
  - square bracket ([ ])
  - left or right brace ({ })
  - backslash (\)
  - two consecutive colons (::)
  - two consecutive dashes (--)
  - comma (,)
  - period (.)
  - asterisk (*)
- First character must be a letter.
- The name must not end with a dash or a space.
- The name must not be **DATA**. Because column names are case sensitive, strings such as **data** and **Data** are acceptable.
- The name must consist of 1 to 255 characters.

> **NOTE:** Remember that even if you do not rename a column, you must copy the original name to the **New Name** field. Do not leave the **New Name** field blank.

After renaming the columns, click **JOIN TABLES**. A new table is created as a result of joining the tables with the keys you selected.

Grouping data in a table

You can use the Group By function in a similar way as you would with the Group By clause in SQL. You can group by one column or multiple columns (up to 96 columns).

Aggregate functions supported

The following aggregate functions are supported:

- Average
- Count
- List aggregate
- Maximum
- Maximum integer
- Minimum
- Minimum integer
- Sum
- Sum integer

Grouping data by columns

Follow these steps to group data by one or multiple columns:

1. Right click the header of the column that you want to group by to display the drop-down menu. Then select **Group By...**.

   The following list describes the general rules and tips for entering information in a Xcalar Insight panel.

   - By default, the name of the column from which you start the function is displayed as an argument. If you want to use another column as an argument, you can change the argument to the name of another column.
   - When typing a column name, always start with the dollar sign ($).

- When typing a column name, always include the prefix if one exists. (A derived column does not have a column name prefix.)
- If the panel calls for the resultant column name, type the name without preceding it with the dollar sign ($).
- To avoid mistakes when typing a column name as an operand, place the insertion point in the argument field, move your mouse to the active table and click anywhere in the desired column. The column's name is automatically entered in the operand field.
- If an operand accepts a column name, it also accepts an aggregate name that you created when you calculated the aggregate value for a column.
- If necessary, click  in the panel to bring the specified table into focus. You can then scroll the table to make the desired column visible.
- You can undo any typing with the undo key or key combination that you normally use in your browser.

2. (Optional)In the **GROUP BY** panel, click **ADD ANOTHER COLUMN** to group on an additional column. For example, in the table about airlines, you can start the Group By function from the **Carrier** column and add the **DayOfWeek** column as a second column to group on. In the resultant table, data is grouped by both the **Carrier** and **DayOfWeek** columns.
3. In the **GROUP BY** panel, follow these steps to select the function to perform for the Group By operation:
   a. Select the aggregate function to group.

      (The following functions are supported:

      - avg
      - count
      - listAgg
      - max
      - maxInteger

- min
- mininteger
- sum
- sumInteger

b. Enter the field name to which the function applies. For example, in the table for airlines information, you can specify the avg function be applied to the **Distance** field. This function calculates the average distance for the column being grouped on.

4. Accept the default name for the new column or enter a new name.
5. Select **Join table back to original** if you want to join the resultant table to the original table. This performs a left outer join, with the original table being the left table. Joining back the table provides a convenient way to compare the values in the column resulting from the Group By operation to values in the original column.
6. Select **Keep all columns** if you want the resultant table to include all columns in the original table. The column created by the Group By function is a derived column. All other columns retain the column prefix.

   If this option is not selected, the resultant table contains only the new column created from the Group By function and the columns you group on. All columns in the resultant table, including the ones you group on, become derived fields.

   > **NOTE:** This option is not applicable if you select **Join table back to original** because the table created by the Join operation automatically includes all the columns from the original table.

7. (Optional) Select **Include only ICV rows** if you want to filter the resultant table such that only erroneous rows are included. For more information about integrity constraint violations, see Creating an integrity constraint violations table.
8. Click **GROUP BY**.

Using the Map function to create new values

The Map function is divided into categories of functions that you use to create new values from one or multiple selected columns. These new values enable you to derive meanings from your dataset. For example, you can perform a trigonometric calculation from values in two selected columns to create a third column that contains the results of the calculation, or you can generate a column of unique values.

This section describes the steps required to perform a Map function. For descriptions of all Map functions, see Map functions.

> **IMPORTANT**: Many Map functions work in a similar way as Excel functions. However, there might be differences such as differences in the number of operands accepted. Do not assume that the Map functions are identical to the Excel functions even though they share the same name.

Creating a column by the Map function

Follow these steps to create a column by the Map function:

1. Right click the header of the column that you want to create new values from. In the drop-down menu, select **Map...**. The **MAP** panel is displayed.

   The following list describes the general rules and tips for entering information in a Xcalar Insight panel.

   - By default, the name of the column from which you start the function is displayed as an argument. If you want to use another column as an argument, you can change the argument to the name of another column.
   - When typing a column name, always start with the dollar sign ($).
   - When typing a column name, always include the prefix if one exists. (A derived column does not have a column name prefix.)

- If the panel calls for the resultant column name, type the name without preceding it with the dollar sign ($).
- To avoid mistakes when typing a column name as an operand, place the insertion point in the argument field, move your mouse to the active table and click anywhere in the desired column. The column's name is automatically entered in the operand field.
- If an operand accepts a column name, it also accepts an aggregate name that you created when you calculated the aggregate value for a column.
- If necessary, click  in the panel to bring the specified table into focus. You can then scroll the table to make the desired column visible.
- You can undo any typing with the undo key or key combination that you normally use in your browser.

2. In the **MAP** panel, follow these steps to select the function:
   a. Select the category. The functions in the selected category are displayed.
   b. Select the function
   c. Accept the default name for the new column or enter a new name.

> **TIP:** You can quickly locate a function by typing its name or part of a its name in the **Search map functions...** field. Functions with a matching name are displayed in the **Category** and **Function** columns.

3. Based on the description of the function, fill out the operand fields.
4. Accept the default name for the new column or enter a new name.
5. (Optional) Select **Include only ICV rows** if you want to filter the resultant table such that only erroneous rows are included. For more information about integrity constraint violations, see Creating an integrity constraint violations table.
6. Click **MAP**.

Determining correlation between two values

When you observe the data values in a table, you might want to determine whether there is significant correlation between numerical values in two columns. For example, when you view information about flights originating from a particular airport, you might be interested to know the correlation between departure delays and arrival delays.

Finding the correlation coefficient for each pair of values in the table

Follow these steps to display the correlation coefficients for each pair of numerical values:

1. Right click anywhere in the table title bar or click the triangle in the lower-right corner of the table title bar to display a drop-down menu. The menu presents table options for manipulating the entire table.
2. Select **Correlations...**. The **Correlation and Quick Aggregates** modal window is displayed, which shows the correlation coefficient for each pair of values in the table. Xcalar Insight uses background colors to indicate the strength of the correlation. Dark blue represents a strong positive correlation; dark gray represents a strong negative correlation.

Determining aggregate values

Determining an aggregate value means using values from rows in a column as input to generate a new value. For example, based on values in the Salary column, you can run an aggregate function to determine the average salary.

You can determine the aggregate values for numeric columns in one of two ways:

- Calculate aggregate values for all columns in a table.
- Calculate aggregate values for a selected column.

Calculating aggregate values for all columns

Follow these steps to calculate aggregate values for all numeric columns in a table:

1. Right click anywhere in the table title bar or click the triangle in the lower-right corner of the table title bar to display a drop-down menu. The menu presents table options for manipulating the entire table.
2. Select **Correlations...**. The **Correlation and Quick Aggregates** modal window is displayed.
3. Click the **Quick Aggregate** vertical tab. The modal window displays the values resulting from common aggregate functions for all numeric columns.

Calculating aggregate values for a selected column

If you are only interested in viewing aggregate values for a single column, follow these steps:

1. Right click the header for the column. In the drop-down menu, select **Profile...**.
2. In the **Profile** modal window, the **Aggregate Summary** section displays the values resulting from common aggregate functions for the column.

If you are interested in calculating one particular aggregate value in a selected column or if you plan to use the aggregate value as input for other functions, follow these steps:

1. Right click the header for the column. In the drop-down menu, select **Aggregate**. The **AGGREGATE** panel is displayed.

   The following list describes the general rules and tips for entering information in a Xcalar Insight panel.

   - By default, the name of the column from which you start the function is displayed as an argument. If you want to use another column as an argument, you can change the argument to the name of another column.
   - When typing a column name, always start with the dollar sign ($).
   - When typing a column name, always include the prefix if one exists. (A derived column does not have a column name prefix.)
   - If the panel calls for the resultant column name, type the name without preceding it with the dollar sign ($).
   - To avoid mistakes when typing a column name as an operand, place the insertion point in the argument field, move your mouse to the active table and click anywhere in the desired column. The column's name is automatically entered in the operand field.
   - If an operand accepts a column name, it also accepts an aggregate name that you created when you calculated the aggregate value for a column.
   - If necessary, click  in the panel to bring the specified table into focus. You can then scroll the table to make the desired column visible.
   - You can undo any typing with the undo key or key combination that you normally use in your browser.

2. Select the aggregate function. For example, to calculate the average for the column, select the avg function.

3. (Optional) in the **New Resultant Aggregate Name** field, type the name of the aggregate value. The caret (^) is automatically entered in the field because the name must start with the caret. The naming conventions are the same as those for columns.

> **IMPORTANT:** If you plan to use an aggregate value in the future as input to a function such as a Map function, Xcalar strongly recommends that you assign a name to the value. In this way, you can use the name as an argument to the function. Do not use the actual aggregate value as the argument because the value might not be accurate when you run the function against the entire data source or when you use the function against another dataset.

The following list describes the naming conventions for a table column:

- The name can consist of any alphanumeric characters, space, and special characters except the following characters:
  - single quote mark (')
  - double quote mark (")
  - parenthesis (( ))
  - square bracket ([ ])
  - left or right brace ({ })
  - backslash (\)
  - two consecutive colons (::)
  - two consecutive dashes (--)
  - comma (,)
  - period (.)
  - asterisk (*)
- First character must be a letter.
- The name must not end with a dash or a space.
- The name must not be **DATA**. Because column names are case sensitive, strings such as **data** and **Data** are acceptable.
- The name must consist of 1 to 255 characters.

> **EXAMPLE:** You can name the resultant aggregate value average_salary (which shows as **^average_salary** in the **AGGREGATE** panel). Naming a resultant value is useful if you plan to use the value as input for another function. In this example, you can type **average_salary** as an operand in a **MAP** or **FILTER** panel, or you can type **^average_salary** as input for a function in the function bar.

Displaying saved aggregate values

If you use the **New Resultant Aggregate Name** field in the **AGGREGATE** panel, the value is saved under the aggregate name. If you want to use one of the saved aggregate values as input to a function, you can verify the name of the aggregate value by displaying a list of values. To display the saved aggregate values, click  to display the **TABLE** panel. Then click **Aggregates**.

Understanding and changing table statuses

A table can be active, hidden, or temporary. You can change the status of a table in multiple ways.

Tables consume memory regardless of their status. To determine how much memory each table uses, see Understanding memory usage.

Viewing the tables list

To display the tables list, follow these steps:

1. Click  (Worksheet icon).
2. In the **Worksheet** panel, click  to display the **Tables** panel. names of all tables on the worksheet. The tables are organized by their status.

Active table

An active table is a table being displayed on a worksheet. You can perform operations on a column in an active table. You can also perform operations on multiple active tables, for example, by joining them.

> **NOTE**: The table you are currently interacting with has a blue background for the table title. Other tables in the worksheet have a gray background for the table title, but these tables are also active tables.

Hidden table

A hidden table is a table that you set aside to reduce clutter in your worksheet. The hidden table can be added back to the worksheet in the future. When you hide a table, both the table and the dataflow graph associated with the table disappear from the worksheet.

Temporary table

Xcalar Insight creates temporary tables in several ways.

A temporary table is a table that is no longer active after you perform an operation or undo an operation.

> **Example**: Suppose you create from your dataset a table named **airlines#pe8**. It is the active table displayed in the worksheet, in which you can perform various operations. If you change the data type of a column in it, a new active table named **airlines#pe9** is created. The table named **airlines#pe8** is now categorized as a temporary table.

A temporary table can be created when you undo an operation either through the **Undo** button or the **Revert To This Table** option in a dataflow graph. Xcalar Insight does not drop the table resulting from the operation; instead, it categorizes the table as a temporary table.

> **Example**: Suppose you change a data type for a column in **airlines#pe8** and the resultant table is named **airlines#pe9**. If you undo the data type change, airlines#pe8 becomes the active table again, but **airlines#pe9** is not dropped. Instead, **airlines#pe9** is kept as a temporary table.

A temporary table can be added back to the worksheet to become an active table again.

> **NOTE**: Remember to click  in the **Tables** panel to refresh the temporary tables list after you run an operation or undo an operation. It ensures that the most up-to-date list is displayed.

Hiding a table

You can only hide an active table. You cannot hide a temporary table. Follow one of these steps to hide a table:

- Right click the title bar for an active table to display a menu. Then select **Hide Table**.
- Right click a table icon in the dataflow for the table. In the menu, select **Hide Table**.
- In the **Tables** panel, display the active tables. Select a table from the list and then click  . The following screenshot illustrates how to hide tables in the **Tables** panel

Changing a table to an active table

You can change a hidden or temporary table to an active table in one of two ways.

From the Tables panel

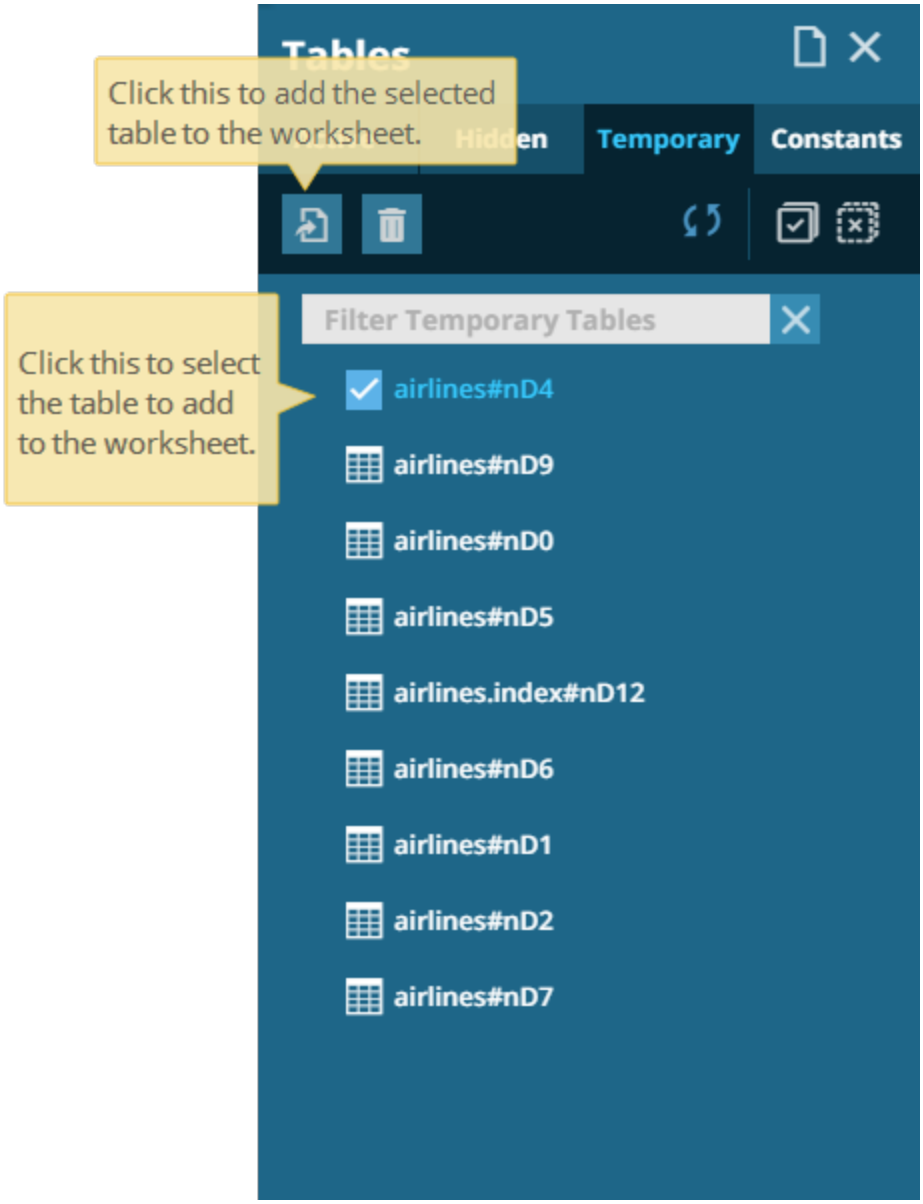
Follow these steps to change the status of a table to active:

1. In the **Tables** panel, display either a list of hidden tables or temporary tables.
2. Select the table that you want to change to an active table. You can select multiple tables from one list.

> **TIP:** The temporary table list might be long. To quickly locate the temporary table, type the complete or partial table name in the **Filter Temporary Tables** field to limit display only tables with the matching name.

3. Click  .

The following screenshot illustrates how to put a temporary table on the worksheet.



From the dataflow panel

Follow these steps to change the status of a table to active:

1. Display the dataflow panel for the worksheet.
2. Locate the table you want to add back to the worksheet.

> **NOTE:** The last table icon in a dataflow always represents an active table. Do not select this table icon from the dataflow.

3. Click the table icon to display a menu. Click **Add Table To Worksheet**.

Managing worksheets

Each workbook contains at least one worksheet. To manage the worksheets in a workbook, click  to display the **Worksheets** panel. You ca perform the following tasks related to worksheets in this panel:

- Renaming
- Hiding or unhiding
- Deleting
- Moving up or down the list

Understanding active and hidden worksheets

A worksheet that can be viewed in your workbook is an active worksheet. You can hide a worksheet so that it cannot be accidentally modified. Tables are listed in the same way whether they belong in an active or hidden worksheet.

The only task that you can perform on a hidden worksheet is unhiding it. Unhiding a worksheet means changing a worksheet back into an active worksheet.

Renaming a worksheet

When you create a worksheet, you can either provide a worksheet name or accept the one suggested by Xcalar Insight. The worksheet can be renamed at any time.

The following list describes the naming conventions:

- Names are case-sensitive.
- Length can be from 1 to 255 characters.
- There are no restrictions on special characters.

Follow these steps to rename a worksheet:

1. In the **Worksheet** panel, right click the name of a worksheet.
2. In the drop-down menu, click **Rename**.

3. Type the new name in the name field, then press Enter.

Hiding or unhiding a worksheet

You can hide a worksheet if it is not the only active worksheet. Follow these steps to hide a worksheet:

1. In the **Worksheet** panel, right click the name of a worksheet in the **Active Worksheet List**.
2. In the drop-down menu, click **Hide worksheet**.

Follow these steps to unhide a worksheet:

1. In the **Worksheet** panel, right click  for a worksheet in the **Hidden Worksheet List**.
2. In the drop-down menu, click **Unhide worksheet**. The worksheet becomes active and is automatically displayed in the Xcalar Insight window.

Deleting a worksheet

You can delete a worksheet if it is not the only active worksheet. Follow these steps to delete a worksheet:

1. In the **Worksheet** panel, right click the name of a worksheet in the **Active Worksheet List**.
2. In the drop-down menu, click **Delete worksheet**. If the worksheet does not contain any tables, it is deleted immediately. Otherwise, a modal window asks what you want to do with the tables in the worksheet. You can either drop all tables or hide the tables. For information about table statuses, see Understanding and changing table statuses.

> **IMPORTANT**: You cannot undo the action of dropping tables.

Moving a worksheet up or down

Follow these steps to move a worksheet up or down the list:

1. In the **Worksheet** panel, right click the name of a worksheet in the **Active Worksheet List**.
2. In the drop-down menu, click **Move up** or **Move down**. The worksheet name is moved up or down the list.

Working with a table containing FNF

You might see some table cells containing the string **FNF** (field not found). This section explains the meaning of FNF and how various operations work with records containing FNF.

When FNF occurs in a table cell

The following list describes the scenarios where you see FNF in a table cell:

- A row does not have a value for the column. For example, a column named EmployeeID exists for other rows, but in row 1, there is no value in the EmployeeID column. The value for row 1 in the EmployeeID column is FNF.

  > **NOTE**: Having an empty string for a column is not the same as having no value. For example, in a JSON file, row 1 is **{a: 1, b: ""}** and row 2 is **{a: 2}**. In row 1, the field b exists and contains an empty string. In row 2, no value exists for the field b. As a result, row 2 shows FNF for the b column.

- You change the data type for a column but the data in a cell cannot be changed. Suppose you change the data type of the EmployeeID column from string to integer. If the EmployeeID in row 2 is NA, it is changed to FNF because the string NA cannot be cast into an integer.
- A table is empty. For example, if no integrity constraint violations are found when you create an ICV table, or if you filter a nonexistent value, the resultant table is an empty table showing FNF in its cells.
- The result of a Map function is undefined. For example, when the div or mod function uses a zero as the divisor, the result of the function is FNF.

Filtering and excluding FNF

You can use a cell option to filter or exclude FNF just as you would other values.

Sorting a table by a column containing FNF

If a table contains a column with FNF, sorting the table by this column results in a table that excludes the rows with FNF. For example, if row 1 contains FNF in the EmployeeID column, after you sort the table by EmployeeID, row 1 of the table before the sort is not displayed in the resultant table.

If a column contains FNF in all rows, you cannot sort the table by the column. Trying to do so results in an error message.

Standardizing data in a column with FNF

One of the purposes of cleansing and standardizing data is to ensure that all data in a column is of the same type. If a column contains FNF, you can map FNF to a value that is useful for future queries. Mapping FNF to another value also enables a sort to include all rows in the resultant table. The following example illustrates how you can standardize data in an integer column have FNF cells.

To change all occurrences of FNF in this column to an integer, follow these steps:

1. Start the Map function in the EmployeeID column to display the **MAP** panel.
2. Select the **exists** function in the **conditional** category.
3. Accept the resultant column name (EmployeeID_exists).
4. Click **MAP**. The EmployeeID_exists column shows **true** if EmployeeID is an integer and **false** if EmployeeID is FNF.
5. Start the Map function in the EmployeeID_exists column to display the **MAP** panel. The **condition to test** field is shown as $EmployeeID_exists.
6. Select the **if** function in the **miscellaneous** category.
7. For the **value returned if condition is true** field, click the EmployeeID column header. The field now displays $EmployeeID.
8. For the **value returned if condition is false** field, type a value to which you want to map FNF. For example, you can enter a negative integer such as -1.
9. Accept the resultant column name (EmployeeID_exists_if).
10. Click **MAP**. The EmployeeID_exists_if column shows the original integer in EmployeeID if

one exists. For each row containing FNF in the EmployeeID column, the cell displays -1.

Now the EmployeeID_exists_if column is identical to the EmployeeID column except that FNF is replaced with an integer (-1). If you sort the table by this column, all rows are displayed after the sort.

> **NOTE**: The string or numeric value you map FNF to depends on your specific query. The exact Map function used for changing FNF to another value might be different from the one described in this example.

Counting occurrences of FNF in a column

Use the **Profile** option in the column drop-down menu to count the occurrences of FNF. In the **Profile** modal window, FNF is shown as **null**. The bar graph for null is displayed in red. For more information about the **Profile** modal window, see Counting occurrences of unique values in a column.

Advanced tasks

This section describes the procedures useful to you after you have understood the concepts described in Basic Xcalar Insight concepts and mastered the techniques described in Common tasks.

Using extensions

Xcalar Insight provides a set of extensions that enhance its features. This section describes how to invoke extensions in general and how each default extension works.

Invoking an extension

You can invoke an extension from the column drop-down menu or from the **My Extensions** panel.

Follow these steps to invoke an extension from the column drop-down menu:

1. Right click the header of a column on which you want to execute the extension.
2. In the column drop-down menu, select **Extensions...** (last option on the menu) to display the **My Extensions** panel. The panel show the extensions installed on the XCE cluster.
3. Click the extension name.

Follow these steps to invoke an extension from the **My Extensions** panel:

1. Click  to display the **My Extensions** panel.
2. Click the extension name.

The following list describes the general rules and tips for entering information in a Xcalar Insight panel.

- By default, the name of the column from which you start the function is displayed as an argument. If you want to use another column as an argument, you can change the argument to the name of another column.
- When typing a column name, always start with the dollar sign ($).
- When typing a column name, always include the prefix if one exists. (A derived column does not have a column name prefix.)
- If the panel calls for the resultant column name, type the name without preceding it with the dollar sign ($).

- To avoid mistakes when typing a column name as an operand, place the insertion point in the argument field, move your mouse to the active table and click anywhere in the desired column. The column's name is automatically entered in the operand field.
- If an operand accepts a column name, it also accepts an aggregate name that you created when you calculated the aggregate value for a column.
- If necessary, click in the panel to bring the specified table into focus. You can then scroll the table to make the desired column visible.
- You can undo any typing with the undo key or key combination that you normally use in your browser.

Horizontal Partition

The **Horizontal Partition** extension enables you to split your active table into multiple tables, keeping the schema of the table intact. After horizontal partitioning, you can choose to start operations on only the resultant table of interest, which consists of fewer records than the original table and hence the improved performance.

Understanding partitions

A partition is a table containing a subset of records that exist in the active table. The column from which you start the extension is the partition column.

A partition contains the rows with the same value in the partition column. You cannot specify the number of rows in a partition. Also, the partition value is determined by the exact value in a column. You cannot specify the condition for the rows in a partition. For example, you cannot specify that the values greater than 100 in the partition column be in one partition and the values less than 100 in the partition column be in another partition.

> **Example:** Your active table has a column named **Salary**, and the values in this column are 100, 200, 300, and 400. Suppose you start the **Horizontal Partition** extension from this column, Xcalar Insight creates 4 partitions: a partition containing all the rows with 100 as the Salary value, a partition containing all the rows with 200 as the Salary value , and so on.

Understanding the number of horizontal partitions

The number of partitions is the number of tables created by the extension. You can create 1 to 10 partitions.

Xcalar Insight chooses the values with the highest frequencies when creating the partitions.

> **Example:** Suppose four values exist in the partition column, which are 100, 200, 300, and 400, and you want to create two partitions, Xcalar Insight will create partitions for the values with the two highest frequencies. If the values 200 and 300 occur most frequently in the column, one partition is created for the rows with 200 and another is created for the rows with 300. No partitions are created for rows with 100 or 400.

If multiple values have the same frequencies in the partition column, the result might be nondeterministic. That is, partitions are created for some, but not all, values of the same frequency when the limit of the number of partitions is reached.

If the specified number of partitions is greater than the number of different values in the partition column, Xcalar Insight will create only the necessary number of partitions. For example, if only four values exist in the partition column and you specify 10 partitions, only four partitions will be created.

Invoking the **Horizontal Partition** extension

Follow these steps to invoke the Horizontal Partition extension:

1. In the **My Extensions** panel, select Horizontal Partition.
2. In the **Table** field, select the table containing the column you want to perform the partition on.
3. In the **No. of Partitions** field, type a number. This is the number of new tables created by the extension.
4. Click **APPLY**.

Windowing

The **Windowing** extension is useful for determining how data changes from one row to another. It uses the SQL **Lead** and **Lag** functions.

> **Example:** You have a table about read access to a data storage system. The table contains a Timestamp column and a Data_read column. You are interested in knowing how the amount of data read changes over time. The **Windowing** extension can sort the rows by Timestamp and creates an additional column showing the amount of data read in another row for comparison. You can then perform other operations on this newly created column.

Follow these steps to use the **Windowing** extension:

1. In the **My Extensions** panel, select the **Windowing** extension.
2. The following table describes each field in the modal window:

Field	Description
Window On	The column you run the **Windowing** extension on. This is the column containing values that change from row to row, and you are interested in analyzing the changes. Data type of the column must be integer, float, or string.
Sort On	Column on which you sort the table. Sorting is necessary before you can compare data from different rows. Data type of the column must be integer, float, or string.
Sort order	Ascending or descending.

Field	Description
Lag	If you want to compare data to a previous row after the sort, fill out this field. For example, type 1 to compare the current row to the one immediately before it, 2 to compare the current row to the one that is 2 rows before, and so on. If you do not use this function, type **0** in this field.
Lead	If you want to compare data to the next row after the sort, fill out this field. For example, type 1 if you want to compare the current row to the one immediately after it, 2 to compare the current row to the one that is 2 rows after, and so on. If you do not use this function, type **0** in this field.

3. Click **APPLY**. A new table containing the columns specified in the panel is created. It also contains an additional column with the data in another row, depending on how you specify the **Lag** or **Lead** value.

Example of the **Windowing** extension

In the table named Disk, the columns Timestamp and Data_read show the time of each read access to a storage system and the amount of data read, respectively. The following is the Disk table:

Timestamp	Data_read
1451628003	17747
1451638804	17840
1451635204	17810
1451631604	17777

Suppose you want to know by how much the data read changes from one timestamp to the next, fill out the fields in the Windowing modal window as follows:

Field	Value
Window On	$Data_read
Sort On	$Timestamp
Sort Order	ascending
Lag	0
Lead	1

The **Windowing** extension creates the following table:

orig_Timestamp	orig_Data_read	lead_1_Data_read
1451628003	17747	17777
1451631604	17777	17810
1451635204	17810	17840

In this new table, the Timestamp and Data_read columns are renamed with orig_ prepended. The table is sorted by timestamp. The new column, lead_1_Data_read, contains the orig_Data_read value in the following row. This new column enables you to analyze the changes of data read over time. For example, to find out how much the data amount changes between two timestamps, invoke the subtraction function in the Map operation by entering the following in the function bar:

```
map(sub(lead_1_Data_read, orig_Data_read))
```

Estimate Join Size

This extension estimates the size of a table created by a Join operation. The estimate gives you the minimum, maximum, and expected number of rows in the new table.

Follow these steps to estimate the size of the table created by a Join operation:

1. In the **My Extensions** panel, select **Estimate Join Size**.
2. Select the Join type.
3. In the **Table** field, select the table, which is the left table in the Join operation.
4. In the **Left Column** field, select the left table's column to join on.
5. In the **Right Table** field, select the right table.
6. In the Right Column field, select the right table's column to join on.
7. Fill out the **Limit on left table** and **Limit on right table** fields, which limit the number of rows Xcalar Insight uses for the estimate. A higher limit gives you a more accurate estimate but takes longer. Limiting the number of rows to about 1,000 rows provides a reasonable estimate without adversely affecting the computer that runs your browser.
8. Click **APPLY**. A pop-up window shows the estimated number of rows after the Join operation.

Generate Row Num

This extension creates a new column containing row numbers starting with 1.

Follow these steps to generate row numbers for a table:

1. In the **My Extensions** panel, select **GenRowNum**.
2. In the **Table** field, select the table for which you want to create a column with row numbers.
3. In the **New Column Name** field, type the name of the column to be created by this extension. The column with row numbers is created as the rightmost column in the table.

Understanding user-defined functions (UDFs)

Some built-in UDFs are included in default Python modules for you to enhance your data queries. You can create additional UDFs to further extend the functions of XCE.

When to run UDFs

You can run UDFs in these circumstances:

- When you execute a Map operation, you can run specify a UDF as a Map function.
- When you point to a data source, you run a UDF to parse the data first. A UDF run before Xcalar Insight points to a data source is called a streaming UDF.

> **EXAMPLE:** If you want to point your data source with a format not natively supported by Xcalar Data Platform, run a streaming UDF to convert that particular format into a natively supported format.

> **EXAMPLE:** If your data source uses a comma in field names, which is an illegal field name character in Xcalar Insight, run a streaming UDF to replace the comma before pointing to the data source.

Understanding UDF names

Functions are defined in modules. Therefore, to uniquely identify a UDF, both the module name and function name are needed. Xcalar Insight shows UDF names in the following format:

module_name:function_name

For example, if a module named **calendar** contains a function named **getWeekday**, the name for the UDF is **calendar:getWeekday**.

UDF naming conventions

Xcalar Insight does not have restrictions on function names.

The following list describes the requirements for module names:

- Names are case-insensitive.
- Only letters and numerals are allowed.
- First character must be a letter.

Limitations and restrictions

The following list describes the UDF limitations and restrictions:

- UDFs must be written in Python.
- The Python version supported by XCE is 2.7. Do not use code in the UDF that requires a Python 3 interpreter.
- The data type for values returned by a UDF is always String. (UDFs, however, can accept arguments of different data types.)
- Each UDF can have up to 16 parameters.
- Python code errors occurring during queries are not reported in Xcalar Insight.
- UDFs do not have read or write access to the file system including the dataset that Xcalar Compute Engine points to.
- UDFs cannot make any non-local network requests. Point directly to network resources rather than connecting to them in the Python code.
- UDFs cannot be used for aggregate operations.

Invoking modules or functions in a module

You can invoke third-party Python modules in UDFs, but they must be installed at a location accessible to the Python interpreter. That is, they must be on the Python search path.

The following list describes the restrictions on invoking UDFs:

- In a UDF, you cannot invoke another uploaded module. For example, if you created and uploaded a module named **calendar**, you cannot invoke **calendar** in another module named **calendar2**.

- You can nest functions in a Python module. However, the nested function cannot be called as a UDF in Xcalar Insight. For example, if the **calendar** module defines a function named **function1**, which in turn defines a function named **function2**, the UDF named **calendar:function1** is available for you to use in Xcalar Insight, but **function2** is not.

Using user-defined functions (UDFs)

This section describes the UDF-related tasks.

Creating a UDF in Xcalar Insight

UDFs are global. UDFs created by one Xcalar Insight user can be used and edited by all other users.

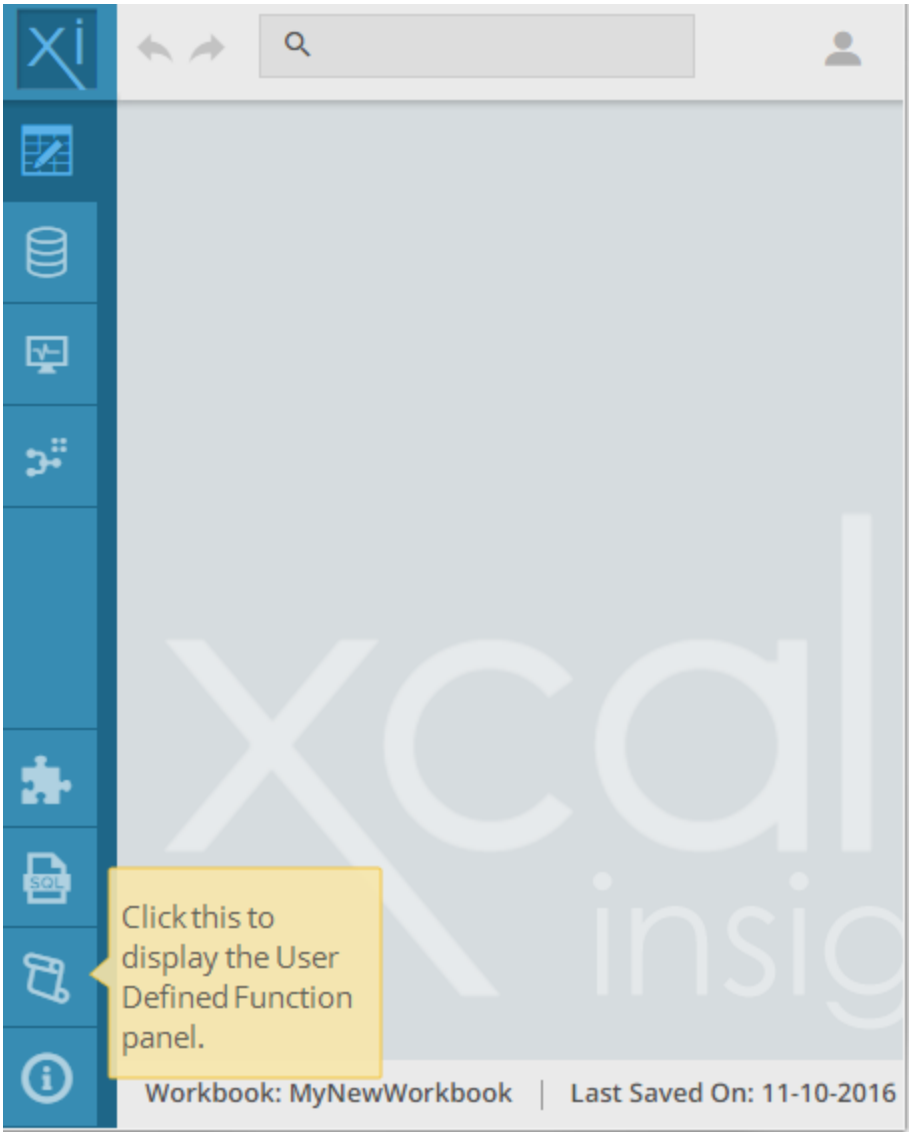
You define one or multiple UDFs in a Python module. You can create a module in these ways:

- Type the code directly in Xcalar Insight.
- Upload from your computer.

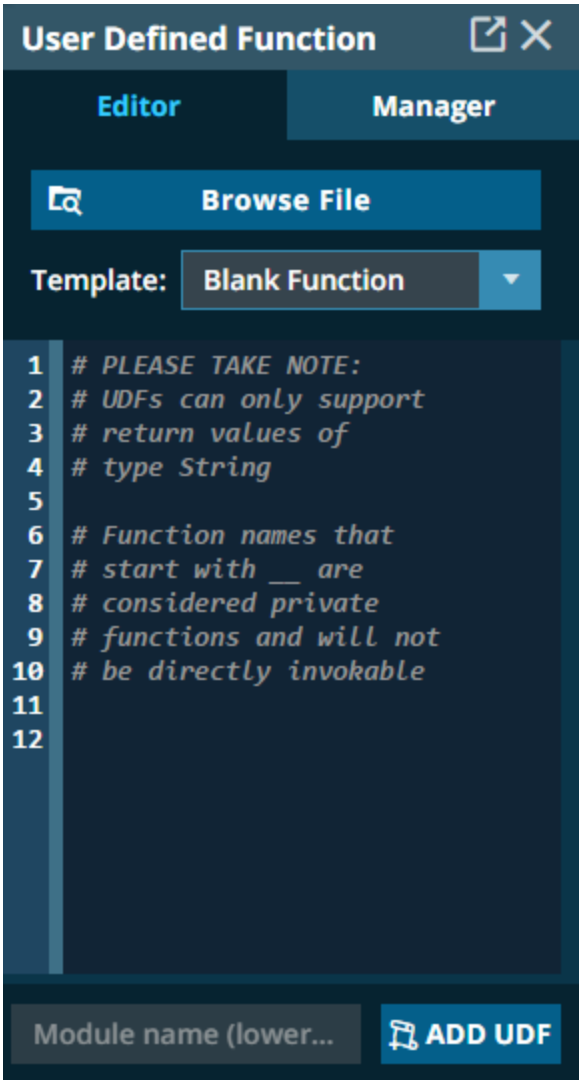
Both methods require that you display the **User Defined Function** panel.

Displaying the UDF panel

Click  to display the **User Defined Function** panel. The following screenshot shows the location of the UDF icon.



The following screenshot shows the **User Defined Function** panel.



Typing the code directly

Follow these steps to write the UDF in the panel and then save it to XCE:

1. Select the **Editor** tab if it is not selected already.
2. Click the drop-down arrow to display a list of existing modules. Select one to use it as a template. Alternatively, select **Blank function** if you want to type in an empty file.
3. Type the code in the panel.
4. In the **Module name** field, type the module name.

> **NOTE:** Module names are case insensitive. If you type letters in uppercase, Xcalar Insight displays them in lowercase when listing the modules. See UDF naming conventions for information about name requirements.

5. Click **ADD UDF**. If a module with the same name already exists, a dialog box asks whether you want to overwrite the existing module. If so, click **CONFIRM**. Otherwise, click **CANCEL** and then rename the module.

Uploading files

Follow these steps to upload a file as a UDF module from your computer to XCE:

1. Select the **Editor** tab if it is not selected already.
2. Click **Browse File**, which enables you to navigate to the folder on your computer that contains the file. Then select the file. The contents of the selected file are displayed in the **User Defined Function** panel.
3. In the **Module name** field, the name of the file is automatically displayed. Change the name if necessary.

> **NOTE:** If the original filename does not follow the naming conventions, an error message notifies you that the name is invalid. See UDF naming conventions for name requirements.

4. Click **ADD UDF**. If a module with the same name already exists, a dialog box asks whether you want to overwrite the existing module. If so, click **CONFIRM**. Otherwise, click **CANCEL** and then rename the module.

About Python code parsing

When you try to add a UDF, it is parsed by the Python interpreter. If there are parse errors, they are reported to you and the UDF is not added.

Editing a UDF

Follow these steps to edit an existing UDF:

1. Select the **Manager** tab in the **User Defined Function** panel. A list of modules is displayed.
2. Edit the module in one of two ways:
   - Click  to edit the desired module. Then edit the code in the **User Defined Function** panel. Add the edited module to XCE without changing its name.
   - Click  to download the desired module to your computer. Then modify the module by using an editor on your computer. Upload the file after you finish editing it.

> **IMPORTANT:** Because UDFs are global and accessible to all Xcalar Insight users, remember to notify other users of the changes so that they are aware of the revision.

Deleting a module

Follow these steps to download a module from XCE to your computer:

1. Select the **Manager** tab in the **User Defined Function** panel. A list of modules is displayed.
2. Click  to delete the module.

Example of a UDF for a database operation

Suppose you have the following table in Xcalar Insight showing some events and their dates, and you want to add a column that shows the day of week for each event. You can write a UDF to create the Weekday column.

Event	Date
Party	2014-06-07
Concert	2015-09-12
Conference	2015-09-14

Coding the UDF

Write the following code in a module named **calendar**:

```
import time
def getWeekday(column, format):
      parsedTime = time.strptime(column, format)
      weekday = time.strftime("%A", parsedTime)
      return weekday
```

The **getWeekday** function takes a column and a date-time format string as input. The Python built-in **time** module, which is imported into **calendar**, parses the input. The output from **calendar** is a new table with the Weekday column showing the day of week for each event.

After you upload the **calendar** module, it is available for all Xcalar Insight users to use.

Executing the UDF

In Xcalar Insight, start a Map operation from the **Date** column. Select **user-defined** as the category and then select **calendar:getWeekday** as the function. Specify the **Date** column ($Date) as the first argument and **%Y-%m-%d** as the second argument.

The process for executing a UDF is the same as those for executing other database functions. For more information about the Map operation, see Using the Map function to create new values.

Result of executing the UDF

The following table is created with an additional column showing the day of week for each date:

Event	Date	Weekday
Party	2014-06-07	Saturday
Concert	2015-09-12	Saturday
Conference	2015-09-14	Monday

Example of using a streaming UDF when pointing to a data source

If you want to point to a data source containing data not natively supported by XCE, you can execute a streaming UDF that converts the data format immediately before Xcalar Insight points to the data. In the following example, the data source contains files in XML, which you can convert into JSON with a streaming UDF.

Coding the streaming UDF

Write the following code in a module named **convertxmltojson**:

```
import xmltodict
import json
def xmlToJson(xmlString, fullPath):
     return json.dumps(xmltodict.parse(xmlString))
```

This module imports a third-party module named **xmltodict** and defines a function named **xmlToJson**. This function has two parameters:

- The **xmlString** parameter is a string containing the full contents of the data source.
- The **fullPath** parameter is a string, which is the full pathname to the data source.

The return value of the function will be used in the place of the actual file contents.

Upload this module so that it is available for execution.

Executing the streaming UDF

In the **Point To New Data Source** window, after you specify the pathname to the data source, follow these steps

1. Select the **Parse Data With UDF** check box.
2. Select the **convertxmltojson** module.
3. Select the **xmlToJson** function.

For more information about pointing to a data source, see Pointing to your data source.

Result of executing the streaming UDF

Suppose the data source is a file named **myFile.xml**. Its path, which you specified in the **Data Source Path** field, is passed to the UDF as the **fullPath** parameter. The contents of **myFile.xml** are passed to the UDF in **xmlString**. The function returns a string containing the corresponding JSON data to which the dataset reference will point.

Using the function or search bar

When Xcalar Insight displays a worksheet, it displays a function bar, which also serves as a search bar. The function bar is denoted with the function symbol, **f(x)**, whereas the search bar is denoted with the search icon ( ).

The function bar is in effect when a column is selected. Initially, the function bar displays the operation that created the column. In the bar, you can replace the operation with another one by typing a function name and its associated parameters.

If no columns are selected, the search bar is in effect. You can search for a string in table names or table column headers. You cannot use the search bar to search for a value in a table.

Searching for a string

You can search column headers and table names in the active worksheet. The search is case insensitive.

To move from one search result to the next, press the **Return** or **Enter** key.

About typing and executing an operation in the function bar

When entering an operation name in the function bar, always precede the name with the equal sign (=). After you finish typing, press the **Return** or **Enter** key to execute the operation.

> **NOTE:** You can execute an operation only on a column that has a column name. If the column header is empty, you cannot type in the function bar.

Operations that you can execute from the function bar are described in this section. You cannot execute other operations.

About column names in the function bar

The argument to an operation in the function bar can be a column name or a field name in the **Data Browser** for any row. Be sure to type a column name correctly when you copy it from

your table. Part of the column header in your table might be obscured. For example, the column name **DayOfWeek_integer** might be visible as **DayOfWeek** in a narrow column that cannot accommodate the entire column name. If you mistype a column name as an argument, you might get unpredictable results. For example, the resultant column might be filled with the FNF (field not found) value.

> **IMPORTANT:** In addition to column names as arguments for a function, you can use an aggregate value for a column that you saved earlier. You must precede the aggregate value name with a caret (^). For example, if the average value for the salary column is average_salary, enter **^average_salary** if you want to use it as an argument to the Map operation in the function bar.

Pull operation

Use the Pull operation to populate the active column with values from the column named in the operation. The source column can be a column already existing in the table or a column listed under the **DATA** header. After the Pull operation, the data type of the active column is changed to match the type of the named column.

> **NOTE:** If the source column has been formatted (for example, as percentages), the format is not copied to the active column. That is, the values in the active column after the pull are not in percentages.

> **EXAMPLE:** To populate the active column with the values in a column named Days, type **=pull(Days)** in the function bar. The rows of the active column are now filled with the same values as the rows in the Days column.

Map operation

You can execute all Map operations and pass the function names and argument values in the function bar.

> **EXAMPLE:** You can add the values 3 and 4 with the add function by typing **=map (add(3,4))**.

You can also invoke a UDF in a Map operation in the function bar as shown in the following example, which invokes a UDF for converting column1 to the UNIX timestamp:

```
map(default:convertToUnixTS(column1, ))
```

> **Recommendation**: The function bar provides a convenient way for advanced users to enter a Map function. You must be familiar with the Map function syntax to be able to successfully execute it in the function bar. Xcalar recommends that you use the MAP panel described in Using the Map function to create new values if you are not familiar with a Map function. The panel provides a description of the function and shows the number of operands appropriate for the function.

> **IMPORTANT**: Function names are case-sensitive. Be sure to type the name in the correct case when using the function bar. A function name in the incorrect case results in the **Error: Could not find function** message.

Example of a Map operation with nested functions

You can nest functions in an operation by using brackets. The following steps illustrate how to perform a Map operation with nested functions and store the resultant values in a new column:

1. Click **Add a column** in the column drop-down menu to create a blank column.
2. Click the header of the new column. Type the column name and press **Enter**. This column becomes the active column and the function bar displays **f(x) =**, which means that you can start typing in it.
3. For each column to be used as a function argument, verify that its data type is either Integer or Float. Change the data type if necessary.
4. Enter the following Map operation:

```
map(mult(mult(mult(div(sub(AskPrice_float, BidPrice_float),
AskPrice_float),100), div(add(BidSize_float,AskSize_
float),2)),100000))
```

This Map operation is equivalent to the following formula:

$$\frac{AskPrice\_float - BidPrice\_float}{AskPrice\_float} \times 100 \times \frac{BidSize\_float + AskSize\_float}{2} \times 100000$$

This operation has nested arithmetic functions such as Add, Sub (subtract), mult (multiply), and div (divide). When you type an opening bracket, Xcalar Insight automatically adds the closing bracket. If there are mismatched brackets, an error message is displayed when Xcalar Insight tries to execute the operation.

The resultant values of the Map operation are stored in the column created in Step 1.

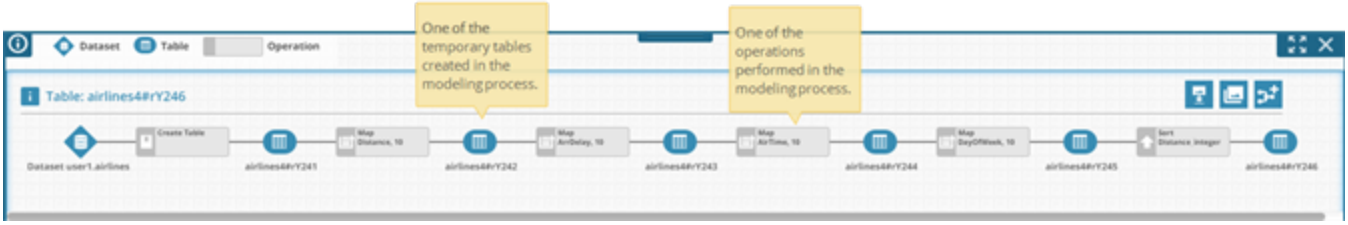
Understanding and creating batch dataflows

Xcalar Insight provides you with built-in functions for you to cleanse data and transform data to a useful format. In addition, you can use UDFs to extend the Xcalar Insight features to suit your data analysis methodology. Executing these functions interactively enables you to see results almost instantaneously. However, it does not show the results for the entire data source. To gain insight from the entire data source instead of a sample of your data, you must run a batch dataflow.

Understanding dataflows

When you use Xcalar Insight to design an algorithm for data analysis, a dataflow is created automatically. It consists of a series of elementary functions executed and tables created when you use Xcalar Insight interactively for modeling. You can view the dataflows leading up to each active table so that you can visually trace the origin and movement of data over time.

> **EXAMPLE:** You pull five columns from your data source to create a table and use the Smart Type Casting feature to change the data type. The dataflow shows that the table undergoes four Map operations to change the data type for four columns, and each operation results in a temporary table. If you sort the Distance_integer column after the data type change, the Sort operation is displayed after the last Map operation. The active table, named airlines4#rY246, is shown as the last table in the dataflow. The following screenshot illustrates the dataflow graph displayed in Xcalar Insight for airlines4#rY246.



> **NOTE:** The table icons in a dataflow graph are usually blue. A red table icon represents an ICV table, which is a table containing only integrity constraint violations. For more information about ICV tables, see Creating an integrity constraint violations table.

Understanding batch dataflows

While a dataflow graph enables you to visually trace data lineage, you cannot re-run the sequence of operations as shown in the graph. To do so, you must save the dataflow under a name. A saved dataflow is called a batch dataflow. It is for you to run as a background program in a similar manner as you would execute a batch file. Batch dataflows are shared among all users on the cluster.

Batch dataflows offer these advantages:

- You can run the set of operations in the dataflow on demand or on a specified schedule. As the data in your data source changes, you can develop insights from your data source over time.
- The set of operations are run against the entire data source.
- Because the operations in a dataflow are automated, you can avoid human errors that might happen if you had to manually execute the operations.
- The batch dataflow is permanent. A table in a batch dataflow can be dropped (for example, if you drop it to release memory), and Xcalar Insight can still run the batch dataflow and recreate that table.
- If you drop an active table, the dataflow associated with the table is also removed from Xcalar Insight. Table dropping cannot be undone, but if you have created a batch dataflow for this table, you can run the dataflow at any time to re-create the table.
- You can export selected columns from the table or the entire table created by a batch dataflow to another application. Alternatively, you can export the result of the batch dataflow to a table, which automatically appears in your worksheet.

Displaying dataflows for all tables

To display the dataflows for all tables in a worksheet, click  in the lower right corner of the Xcalar Insight window. For more information about each element in the dataflow graph, see Dataflow graph.

Creating a batch dataflow

Follow these steps to create a batch dataflow:

1. Click  to view the dataflows for all tables in the worksheet.
2. Locate the dataflow for the active table. This is the dataflow from which you create the batch dataflow.
3. Click  in the upper right corner of the dataflow graph to display the **DATAFLOW** panel.
4. In the panel, type the name of the dataflow.

   The following list describes the naming conventions:

   - Names are case-sensitive.
   - Length can be from 1 to 255 characters.
   - Characters can be A-Z, a-z, 0-9, hyphen (-), and underscore (_).
5. Select the columns to be included in the batch dataflow result. If the batch dataflow does not include a Join operation, skip the next step. Otherwise, go to the next step.
6. If the dataflow result contains columns that have duplicate names, you are prompted to rename the columns.

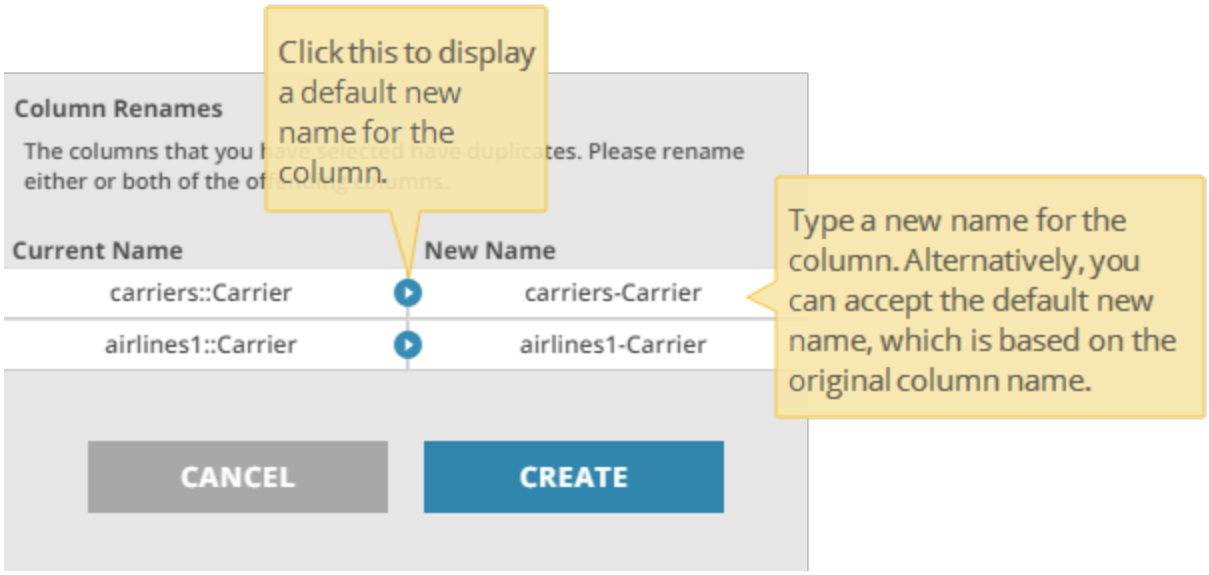
> **NOTE:** All columns created by a batch dataflow are derived columns. If a column before the Join operation has a prefix in the name, the prefix is removed when the batch dataflow creates the table. Therefore, if the left table in the Join operation contains a column named airlines1::Carrier and the right

> able contains a column named carrier::Carrier, the column names are considered duplicates.

The following partial screenshot of the **DATAFLOW** panel illustrates how to rename a duplicate column name when creating a batch dataflow. You can either type a new column name or accept the suggested new column name. The column name does not include a prefix.

The following list describes the naming conventions for a table column:

- The name can consist of any alphanumeric characters, space, and special characters except the following characters:
  - single quote mark (')
  - double quote mark (")
  - parenthesis (( ))
  - square bracket ([ ])
  - left or right brace ({ })
  - backslash (\)
  - two consecutive colons (::)
  - two consecutive dashes (--)
  - comma (,)
  - period (.)
  - asterisk (*)
- First character must be a letter.
- The name must not end with a dash or a space.
- The name must not be **DATA**. Because column names are case sensitive, strings such as **data** and **Data** are acceptable.
- The name must consist of 1 to 255 characters.



7. Click **CREATE**. The dataflow result, whether it is a file or table, contains only the selected columns.

Using batch dataflows

This section describes tasks related to batch dataflows.

Displaying batch dataflows

Follow these steps to display batch dataflows:

1. Click  to display a list of batch dataflows. All batch dataflows, including the ones created by other users, are displayed.
2. Click the name of the batch dataflow in the list to display the dataflow graph.

Result of running a batch dataflow

The icon shown at the end of a batch dataflow graph represents the final result created when you run the batch dataflow. The exact result depends on your selection under **Advanced Export Options** in the graph before you run the batch dataflow:

- **Export to file system**: By default, the table is exported to a csv file, which is saved according to the specifications of the export target. For more information about the meaning of exporting a table, see Exporting a table.

  The name of the export file must not be a duplicate. For example, if running the batch dataflow for the first time creates a file named export-airlines1#L811.csv, make sure that you change the name of the export file when you run the batch dataflow subsequently. Otherwise, your attempt to run the batch dataflow results in the following error message:

  ```
  Export file already exists.
  ```

- **Export to a Xcalar table**: This option exports the table created by the dataflow to another table that will be placed in your active worksheet. You must provide the exported table with a name that does not conflict with any existing table name.

> **NOTE**: You can export the result of a batch dataflow as a table only if you run it from the dataflow graph. If you run it according to a schedule, the batch dataflow always exports the result as a file.

Understanding tables exported from batch dataflows

The table exported to the worksheet from a batch dataflow works in the same way as any other table. It has its own dataflow graph. However, some differences exist between the table created through a batch dataflow and other tables as described below:

- You cannot create another batch dataflow from the dataflow graph associated with this table.
- All columns are derived columns. Column names do not have a prefix.
- You cannot trace the lineage of the data in this table. For example, you cannot determine the source of the data in or whether the data type of a column has been changed.

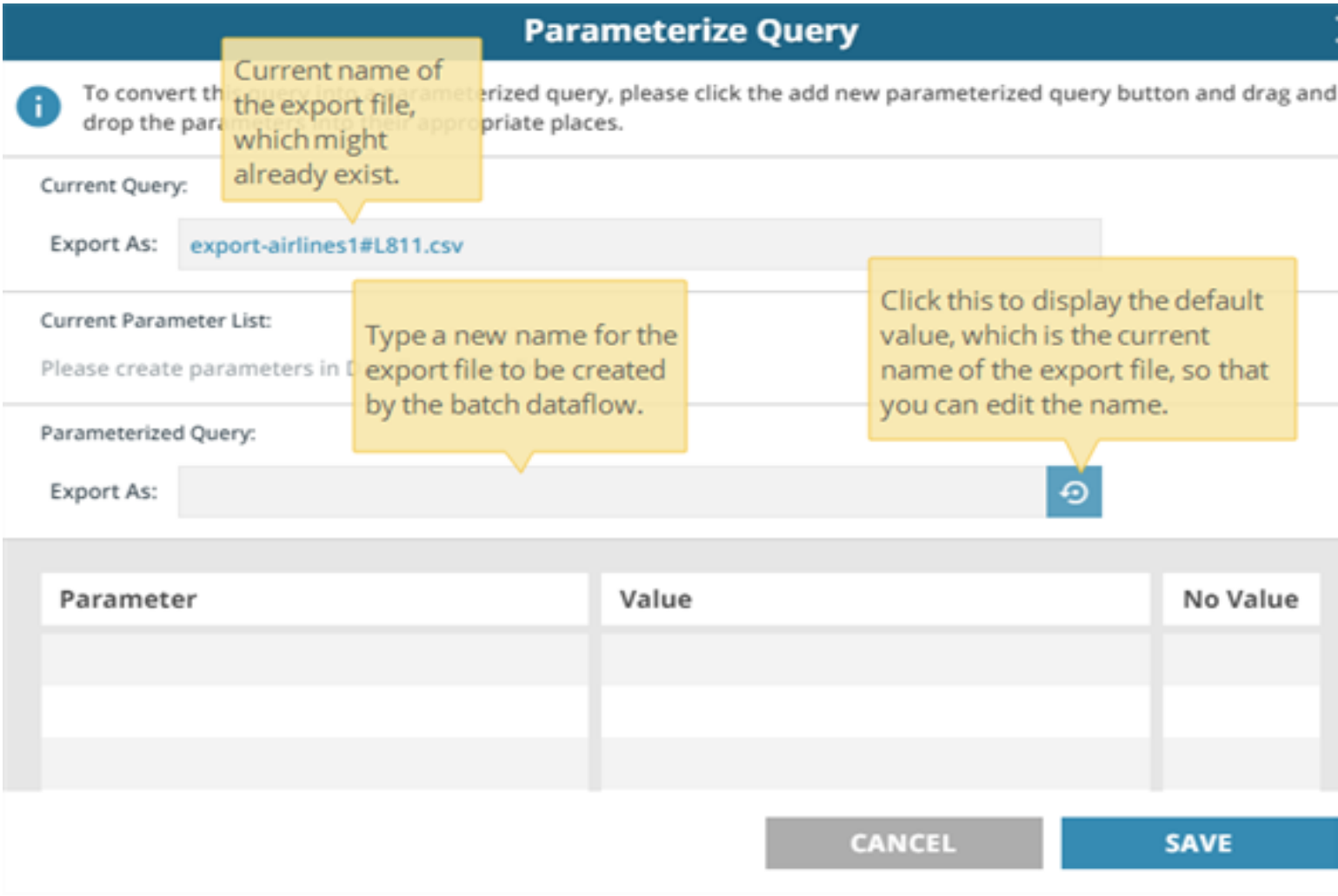
Specifying the name of the export file created by a batch dataflow

If you want to export the result of a batch dataflow to a file, follow these steps to specify the name of the export file:

1. Click the last icon in the batch dataflow graph. Then click the **Create Parameterized Query** pop-up.
2. In the **Parameterize Query** modal window, enter the export file name as shown in the following screenshot, then click **SAVE**.

   The icon for the batch dataflow result in the dataflow graph changes to green to indicate that the default name has been replaced by a user-specified name.

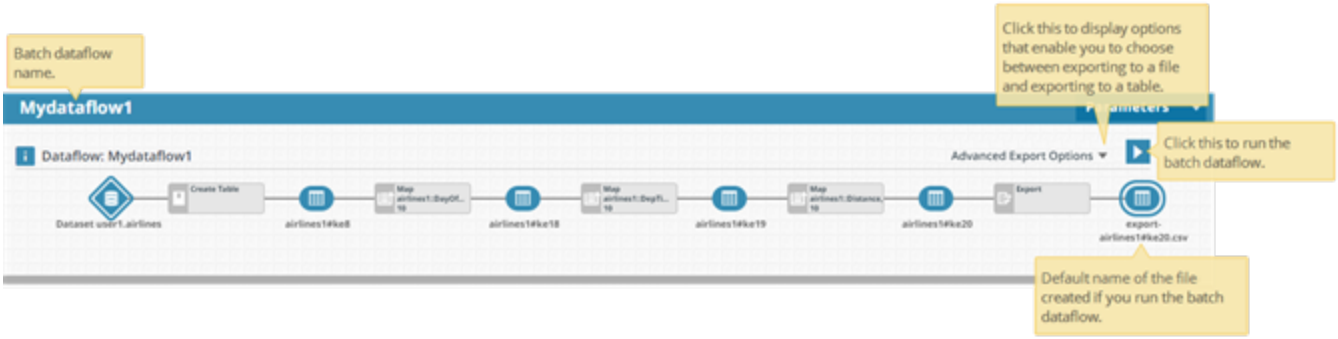
> **NOTE:** You can type the name of the export file as shown in the screenshot. Alternatively, you can parameterize the file name. For more information about parameterization, see Parameterizing queries.



Running a batch dataflow

In the batch dataflow graph, click to run the batch dataflow. When the batch dataflow is in progress, the icon turns into a spinner. You can click the spinner to terminate the execution of the batch dataflow.

The following screenshot shows an example of a batch dataflow.



You can also run a batch dataflow according to a pre-defined a schedule. For information about scheduling a batch dataflow, see Creating a schedule for a batch dataflow.

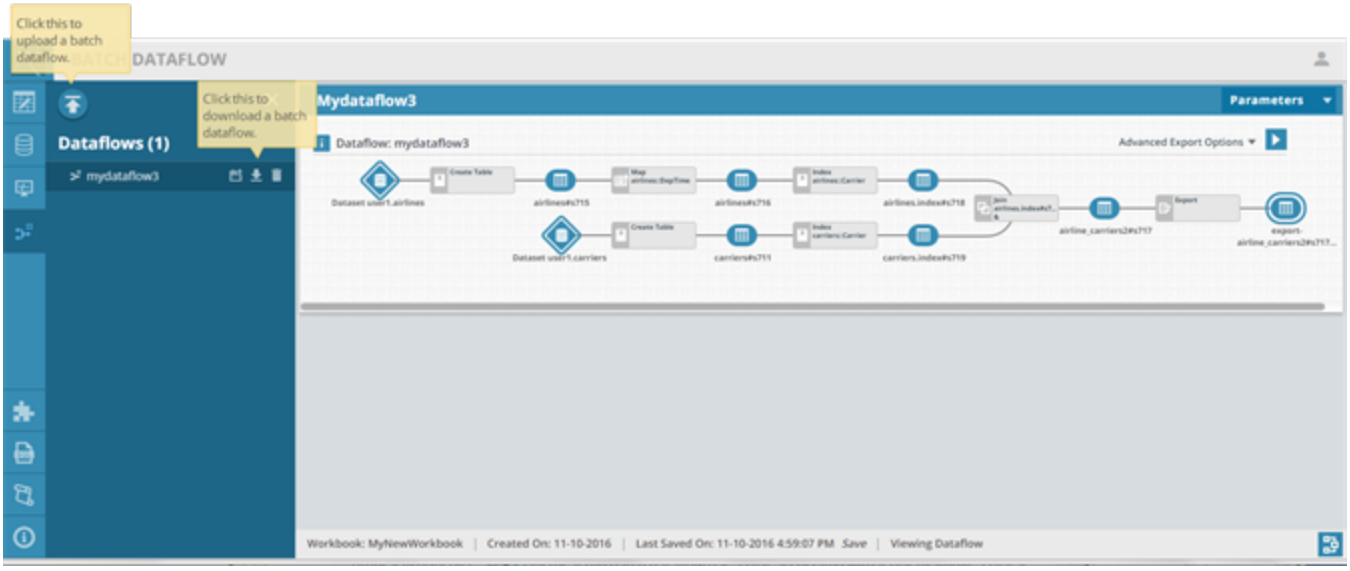
Downloading and uploading a batch dataflow

You can download a batch dataflow to your computer as a .tar.gz file. The downloaded batch dataflow can be uploaded to any XCE cluster.

Downloading

To download a batch dataflow, click  for the selected batch dataflow in the batch dataflow list.

The following screenshot shows the icons for downloading and uploading batch dataflows.



Uploading

You can upload a batch dataflow that was downloaded from the same XCE cluster or any other XCE cluster. If the batch dataflow has a UDF associated with it (for example, a UDF is used in one of the operations in the dataflow), the UDFs of the cluster are affected in one of the following ways:

- If the module name of the UDF does not conflict with any module name on the cluster, the module containing the UDF is uploaded to the cluster as well. For example, if the batch dataflow uses the UDF named calendar:getWeekday, the calendar module is added to your cluster when you upload the batch dataflow.
- If the module name of the UDF conflicts with a module name on the cluster, what happens depends on whether you choose to overwrite the module on the cluster with the module in the batch dataflow:
  - If you choose to overwrite the module on the cluster, all contents of the module on the cluster are replaced by the contents of the module in the batch dataflow. For example, if the batch dataflow uses the UDF named calendar:getWeekday, the module named calendar on the cluster is overwritten by the module from the batch dataflow.
  - If you do not choose to overwrite the module on the cluster, the same function on the cluster is used when you run the batch dataflow. If the function by the same name does not exist in the module, the attempt to run the batch dataflow fails.

    For example, if the batch dataflow uses the UDF named calendar:getWeekday and the cluster also has a UDF by the same name, the calendar:getWeekday UDF on the cluster is invoked when you run the batch dataflow. However, if the calendar module on the cluster does not contain a getWeekday function, you cannot run the batch dataflow. You must copy the function to the module on the cluster before trying to run the batch dataflow again.

    > **IMPORTANT:** You cannot revert a module after it is overwritten by a batch dataflow upload. Enable the overwriting only after you verify that you no

> onger need the contents of the existing module that has the same name as hat in the batch dataflow.

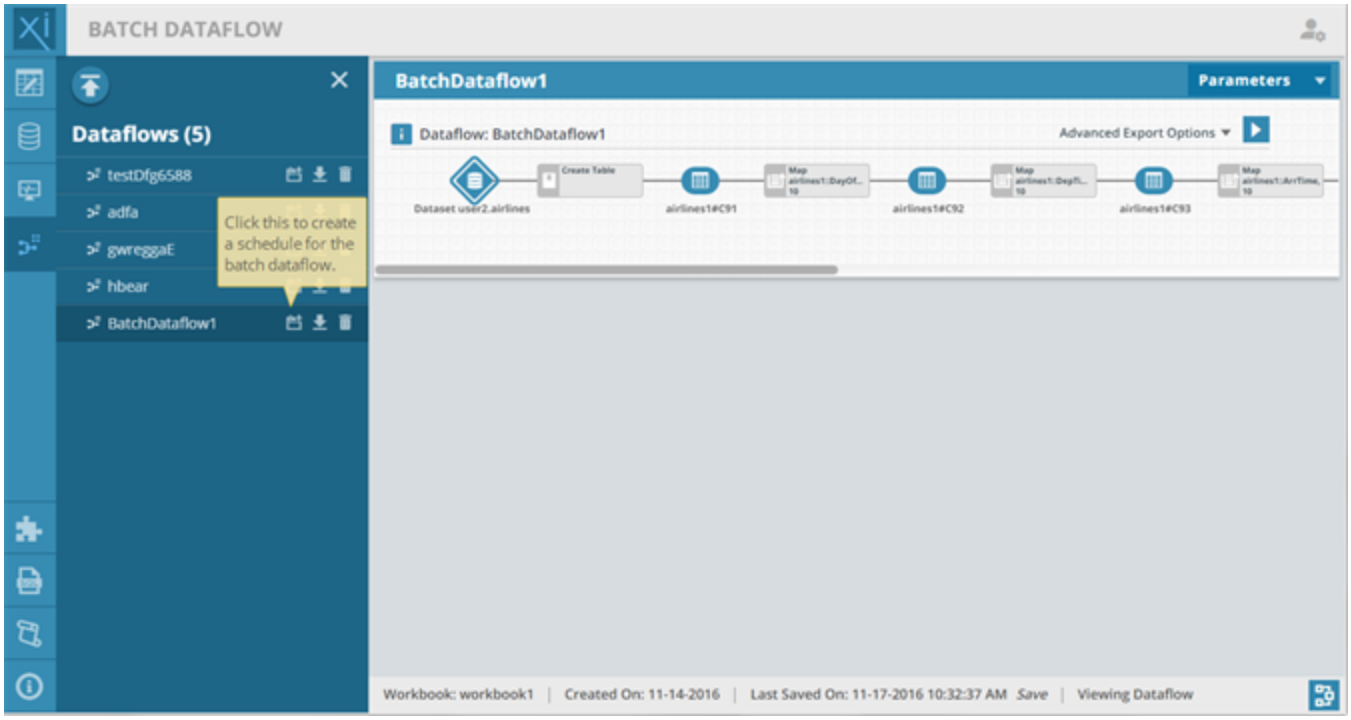
To upload a batch dataflow, follow these steps:

1. Click the icon for uploading a dataflow.
2. In the **Upload Dataflow** window, specify the following information:
   - The location of the previously downloaded batch dataflow, which must be in the .tar.gz format. You can click **BROWSE** to navigate to the location and select the file.
   - Name of the batch dataflow if you want to change the name.
   - Whether the UDFs in the batch dataflow will overwrite the ones that exist on the cluster if these UDFs have the same module name.
3. Click **UPLOAD**.

Creating a schedule for a batch dataflow

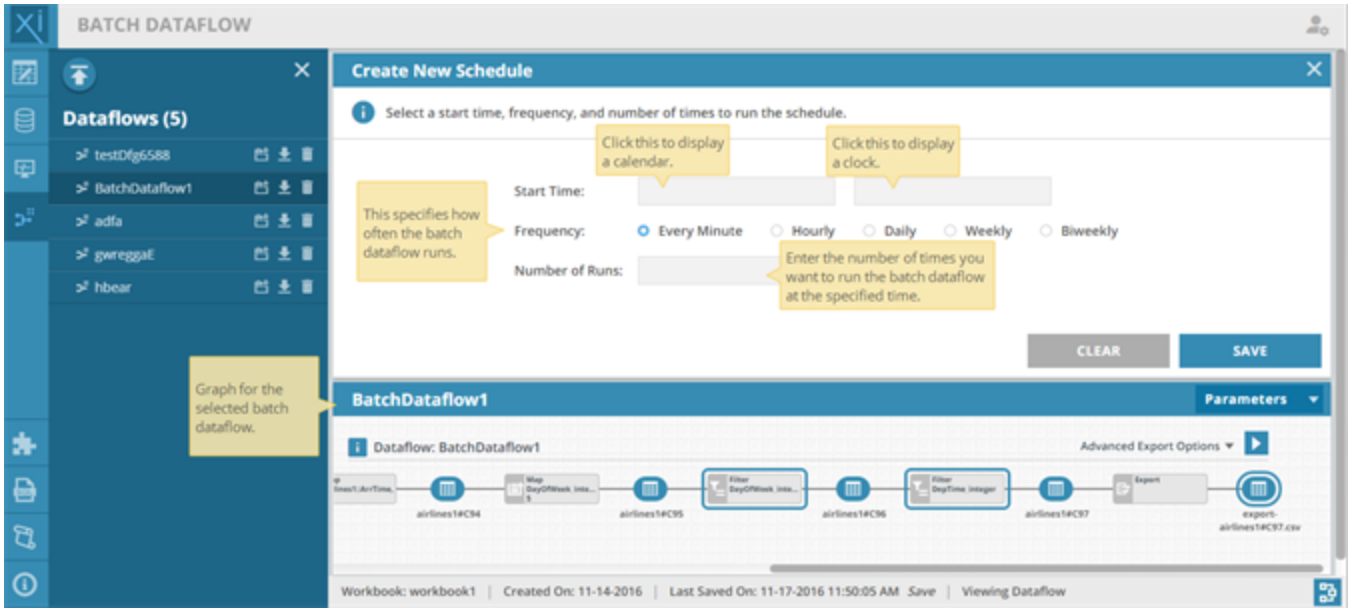
After you create a batch dataflow, you can run the dataflow at regular intervals. Follow these steps to create a schedule for a batch dataflow:

1. In the dataflow list, click  for the selected batch dataflow as illustrated in the following screenshot.



2. Specify the following information in the **Create New Schedule** panel. See the screenshot following the list for more information about how to specify the information.

   - The start time through a calendar and a 12-hour clock, which determines the first time the batch dataflow will be run. The time specified is based on the cluster time, and it must be in the future.
   - How often the batch dataflow is run.
   - How many times the batch dataflow is run. There is no maximum number of times the dataflow can be run at the specified time.

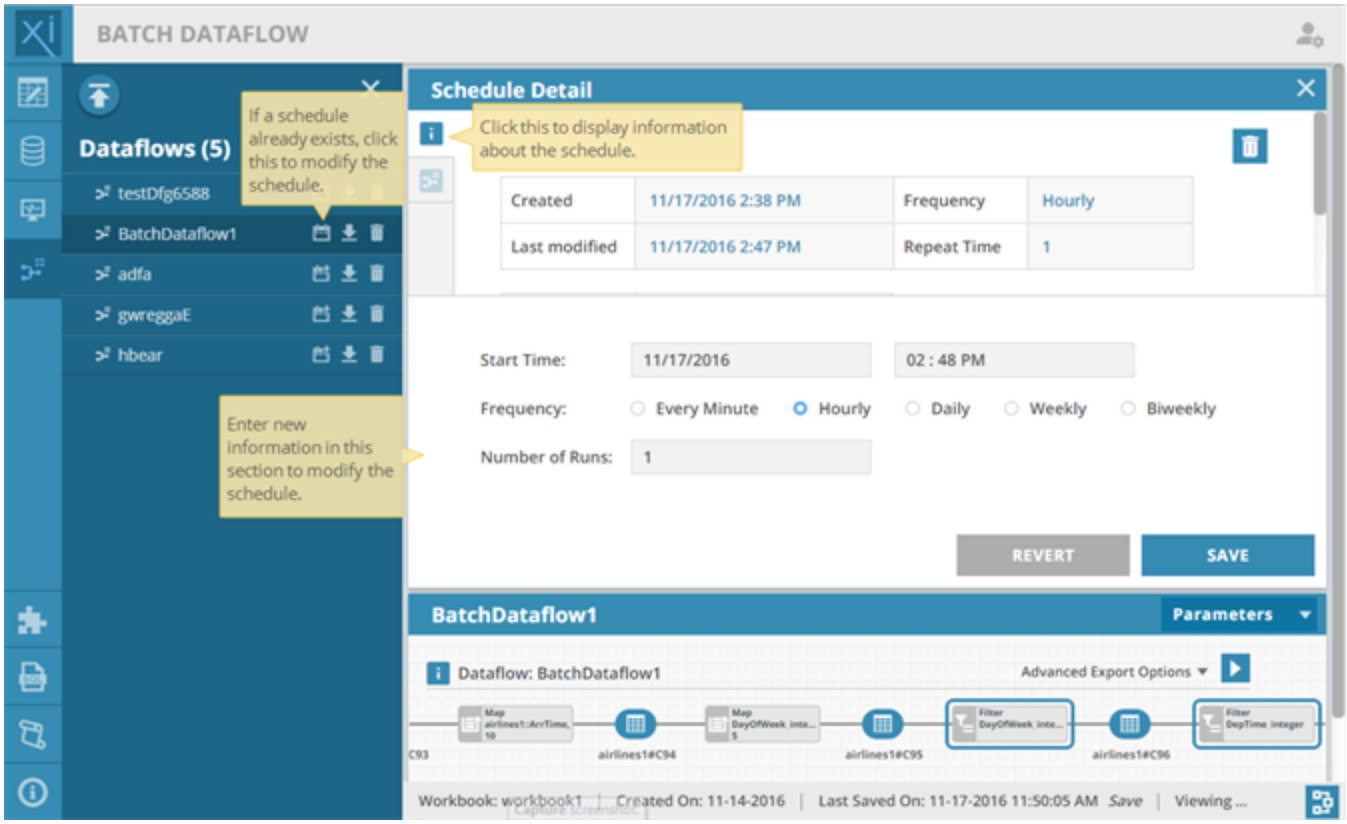


3. Click **SAVE**.

Modifying a batch dataflow schedule

Follow these steps to modify a batch dataflow schedule:

1. In the dataflow list, click  for the selected batch dataflow as illustrated in the following screenshot.

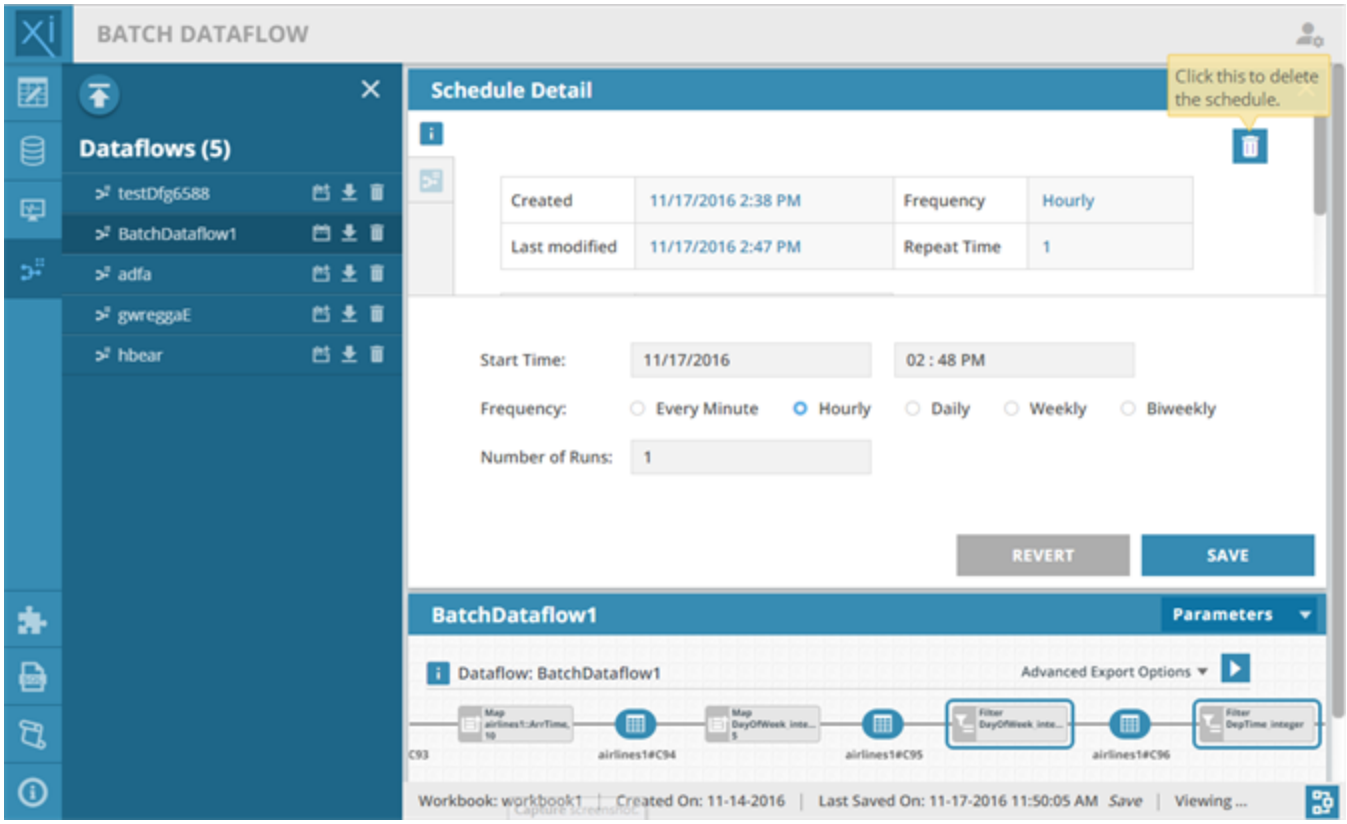


2. Enter new scheduling information. If you want to abandon the changes, click **REVERT**.
3. Click **SAVE**

Deleting a batch dataflow schedule

Follow these steps to delete a batch dataflow schedule:

1. In the dataflow list, click  for the selected batch dataflow.
2. Click the Delete icon as shown in the following screenshot.



3. When the confirmation message is displayed, click CONFIRM.

Exporting a table

Exporting a table means saving a table from XDP as a file to a location where other applications (or XDP) can access and use as a data source. You can save an entire table with all the columns or only selected columns.

> **NOTE:** Columns of the Array or Object type cannot be exported.

You can export a table in one of two ways:

- Manually export a table from the table drop-down menu
- Automatically export a table resulting from running a batch dataflow

Where tables are exported

By default, tables are exported to export files in a subdirectory in the /var/opt/xcalar/export directory, which is called the default export target. You can, however, create export targets for different export files. An export target specifies both the location of the export directory and whether a UDF is used to process the tables before the tables are saved to the export files.

When you export a table automatically by running a batch dataflow, the table can be exported to the default export target only. When you manually export a table, you can choose to use the default target or a user-defined target.

How export files are named

If you manually export a table, a default name of the export file is provided in the **EXPORT TABLES** panel. If you run a batch dataflow to export the resultant table, the default export file name is shown below the exported table icon of the batch dataflow graph. (The exported table icon is the last table icon in the graph.)

By default, Xcalar Insight displays an error if it detects a file name conflict when exporting a table. To avoid the name conflict, you must create a unique export file name. The procedure for creating or editing an export file name is described later in this section.

Exporting a table to an existing file does not generate an error if you override the default behavior. For example, when manually exporting a table, you can specify in the **EXPORT TABLE** panel that the newly exported table overwrites the existing file or appends to the existing file.

The following list describes the naming conventions:

- Names are case-sensitive.
- Length can be from 1 to 255 characters.
- Characters can be A-Z, a-z, 0-9, hyphen (-), and underscore (_).

Format of export files

By default, tables are exported to csv files. You can, however, process a table with a UDF to save it to a file in another format. For example, if you want to export a table to a file to be used by Excel, you can use a UDF to parse the table so that it can be saved in the Excel file format.

You must create an export target in advance if you want to use a UDF to process your table before saving it to an export file. Also, the UDF must be accessible to Xcalar Insight. For information about UDFs, see Understanding user-defined functions (UDFs).

Creating an export target

Creating an export target is necessary if one or both of these conditions are true:

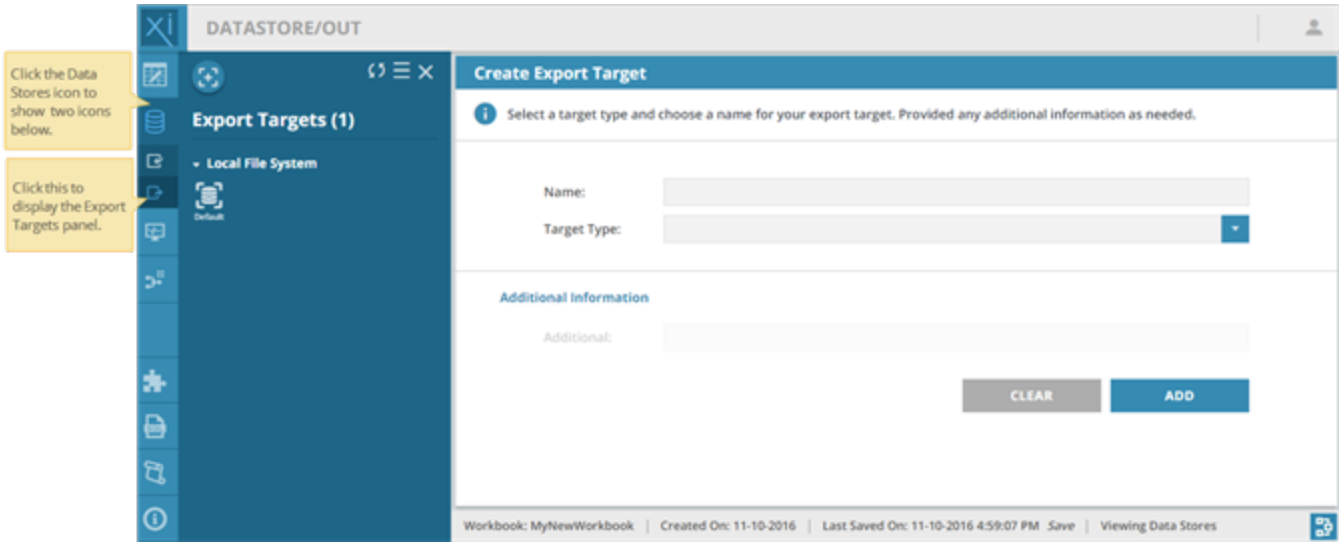
- You want export files to be saved at a location other than the default location (/var/opt/xcalar/export).
- You want the exported table to be processed by a UDF before being saved.

Follow these steps to create an export target:

1. Create a file system that will be used as the export location. For example, create the directory /var/my_exportdir.

> **IMPORTANT:** This directory must point to shared storage that can be accessible to all nodes of your cluster.

2. Click  to expand the icons for Data Stores. Then click  to display the **Export Target** panel and the **Create Export Target** window, The following screenshot shows the location of the Data Stores icons.



3. In the **Create Export Target** window, the information about the default export target is displayed. To create a target, click  in the **Export Targets** panel. The **Create Export Target** window now displays fields for you to define a new export target.

4. In the **Create Export Target** window, specify the name of the target. You need to refer to the target name when you export a table in the future.

   The following list describes the naming conventions:

   - Names are case-sensitive.
   - Length can be from 1 to 255 characters.
   - Characters can be A-Z, a-z, 0-9, hyphen (-), and underscore (_).

5. Enter the pathname of the export location. Export files using this target will be saved to this location.
6. (Optional) If you want the export file to be processed by a UDF, follow these steps:
    a. Select UDF.
    b. Enter the UDF module name and function name.
7. Click **ADD**.

Exporting a table from the table drop-down menu

Follow these steps to export a table:

1. In your worksheet, locate the table that you want to export. Click the table title bar to display the table drop-down menu.
2. Select **Export table** in the drop-down menu. The **EXPORT TABLE** panel is displayed.
3. In the **EXPORT TABLE** panel, fill out the following information:
    - Table to Export: If the table displayed is not the one you want to export, use the Down arrow to display a list of tables in the active worksheet and select the table from the list.
    - Export filename: Specify a unique name for the export file.
    - Export Location: It is the export target to be used for exporting the table. Use the Down arrow to display a list of targets that have been created.
    - Columns to export: By default, all columns are exported. You can select only a few columns.
4. (Optional) Click **Export Rows in Sorted Order** if you want the rows in the export file to be sorted in the same way as the table in your worksheet. Sorting the rows causes the export to take a longer time. This option is available if both of these conditions are true:
    - The table is exported without being processed by a UDF.
    - The table rows are already sorted.

5. (Optional) If you want to further customize the way in which the table is saved to the export file. see Advanced options for exporting a table.
6. Click **EXPORT**.

Advanced options for exporting a table

The advanced options in the **EXPORT TABLE** panel enables you to specify different ways to export a table:

Type

By default, the table is saved to a CSV file. You can specify that the export file be in SQL format.

For each CSV file, you can specify the field delimiter and record delimiter in a similar way as you specify these settings when pointing to a data source.

File

By default, the table is exported to a single file. You can specify that the table be exported to multiple files. You cannot, however, specify the exact number of files to be created. Xcalar Insight automatically decides the number of export files.

> **NOTE:** Specifying **Multiple Files** does not always result in multiple export files. It is possible that the table is exported to a single file even if this option is selected.

Header

By default, the column headers are saved to each export file. You can specify that the headers be saved to a separate file, in which case the data rows are saved to a file or files without the headers.

Overwrite

You can select one of the three ways of overwriting existing export files:

- **Do Not Overwrite**: By default, the table is exported to the file named in the export target, and if a table has been saved to the file, Xcalar Insight does not overwrite the file. It

displays an error message that tells you to create a new export file name.

- **Overwrite Existing**: You can choose to overwrite the existing file. If the export file named in the export target already contains a table previously exported to the file, the contents of the export file is overwritten.

  > **NOTE:** This option overwrites the contents of the entire directory to which a table was previously exported, regardless of how many export files are created as a result of exporting the current table. For example, a directory named /var/opt/xcalar/export/airlines contains multiple export files, which are named airlines-001.csv and airlines-002.csv. If you use airlines as the export file name to export the current table as a single file, the /var/opt/xcalar/export/airlines directory will contain the export file named airlines.csv. The files named airlines-001.csv and airlines-002.csv no longer exist.

- **Append To Existing**: You can add the newly created export file to an export location that already contains export files. How the appending is done depends on whether the table is exported as a single file or multiple files as illustrated by the following example.

  > **EXAMPLE:** If the export target already contains airlines-001.csv and airlines-002.csv, exporting a table as multiple files creates airlines-003.csv and airlines-004.csv in this directory. If the export target already contains airlines.csv, exporting a table as a single file appends the exported table's rows to airlines.csv.

  > **NOTE:** You cannot append a table being exported as a single file to multiple existing export files. Similarly, you cannot append a table being exported as multiple files to a single existing export file.

Exporting a table when running a batch dataflow

As the last step of running a batch dataflow, Xcalar Insight exports a table (or selected columns of a table) to a file located in the default export target. This step takes place automatically without your action.

For more information about batch dataflows, see the following topics:

- Understanding and creating batch dataflows
- Using batch dataflows

Parameterizing queries

Xcalar Insight supports parameterization of queries. This feature enables you to re-use an existing dataflow with customized values, eliminating the need for building a dataflow from scratch for each query. Specifically, you can take advantage of an existing batch dataflow to accomplish the following goals:

- apply the entire series of operations defined by a batch dataflow to another dataset
- substitute a filter function for an existing one in a batch dataflow
- name the export file created by a batch dataflow

This section assumes that you have already created a batch dataflow and you are familiar with the concept of exporting a table. The following topics provide information about dataflows and exporting tables:

- Understanding and creating batch dataflows
- Using batch dataflows
- Exporting a table

General steps for parameterizing a query

The following list describes the general steps for parameterizing a query:

1. Create a parameter.
2. Pass the parameter to the query.
3. Assign a value to the parameter.

The exact steps might vary depending on the type of query being parameterized.

Using the Parameterize Query modal window without a parameter

Xcalar Insight allows you to customize a batch dataflow without creating a parameter. To accomplish this, follow the instructions in Passing a parameter to display the **Parameterize**

Query modal window. Instead of dragging and dropping a parameter in the modal window, simply type the desired value where a parameter should be placed. Xcalar recommends that you define a parameter instead of entering the value directly because having a parameter provides a convenient way to update a particular value. You can also re-use the same parameter for different things in the dataflow. For example, you can use the same parameter in the data source path name and in the export file name to maintain consistency.

Parameter naming guidelines

When creating a parameter, remember to choose a name that describes the meaning of the parameter.

The following list describes the naming conventions for a parameter:

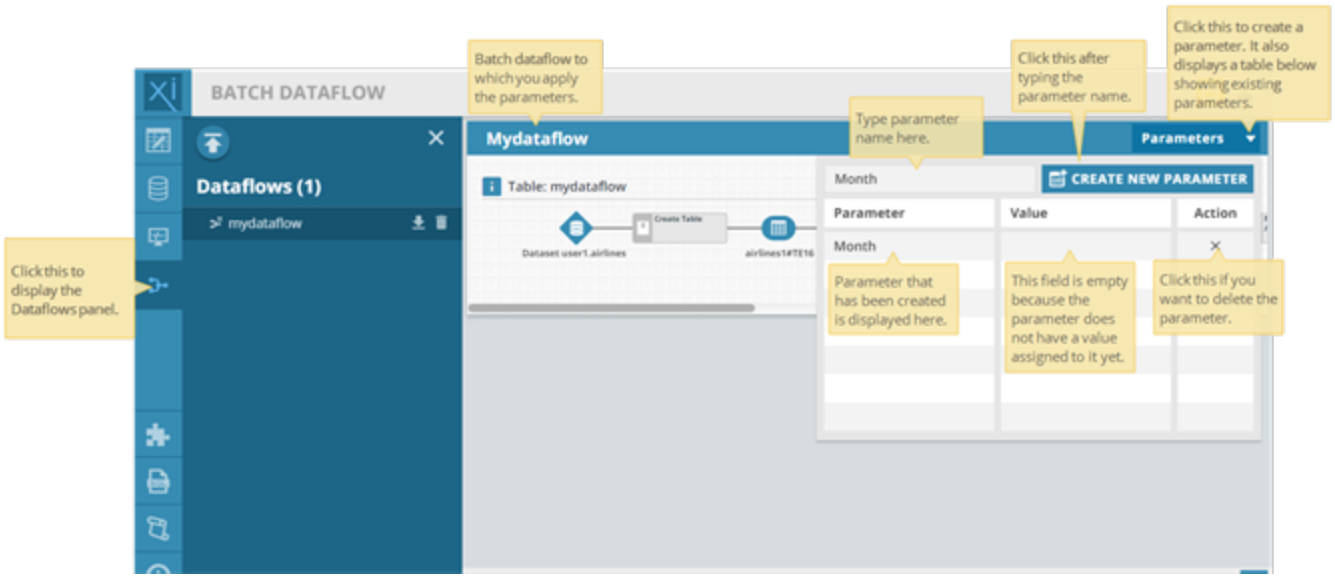
- Parameter names are case-sensitive.
- Length can be from 1 to 255 characters.
- Characters can be A-Z, a-z, and 0-9.
- No special characters (for example, underscore, parenthesis, hyphen, or space) are permitted.

Creating a parameter

Parameters are created for each batch dataflow. They are not shared among batch dataflows. You can create for each batch dataflow as many parameters as needed.

Follow these steps to create a parameter:

1. Click  to display the **Dataflows** panel. Then select the batch dataflow that you want to parameterize. The dataflow graph for the selected batch dataflow is displayed.
2. In the upper right corner of the window, click the Down arrow next to **Parameters**. Then type the parameter name as shown in the following screenshot.



3. Click **CREATE NEW PARAMETER** or press Enter.

Passing a parameter

Passing a parameter means substituting the parameter value for the value originally used in the batch dataflow. You can pass a parameter at different stages of a batch dataflow: when pointing to a data source, performing a filter operation, or exporting a table to a file. The exact steps for passing a parameter depend on what you want to parameterize.

Follow these steps to pass a parameter:

1. In the batch dataflow graph, click one of the following icon and then the **Create Parameterized Query** pop-up:

   - dataset icon
   - filter operation icon
   - exported table icon

   The **Parameterize Query** modal window is displayed. The exact contents of the modal window depend on the icon from which you launch the modal window.

2. In the **Parameter List**, verify that the parameter you want to use has been created. If not, create the parameter as described in Creating a parameter.

3. In the **Parameterize Query** section, drag and drop the parameter from the **Parameter List**. Where you drop the parameter depends on where you want the parameter value substitution to occur.
4. Assign a value to each parameter that you drag and drop as described in Assigning a value to a parameter.

> **TIP:** You can drag and drop more than one parameter. Suppose you have a parameter named Month and a parameter named Year. To parameterize the export operation, you can drag and drop both parameters to the **Export As** field so that the month and year are concatenated to form a part of the export file name (for example, to form a file name prefix).

Assigning a value to a parameter

The value assigned to a parameter is the value that will be used when the batch dataflow is run, instead of the original value used when the batch dataflow is defined.

> **EXAMPLE:** Suppose you have created a parameter named Month, you can assign a different value to it depending on the month in which the data in the dataset is collected. If the data is from the month of October, you can assign the string OCTOBER to the Month parameter, and then apply the Month parameter as the export file name. In this way the export file can contain the string OCTOBER (for example, export-airlines-OCTOBER.csv), which makes it obvious that the export file is created by a dataflow based on data collected in October.

Follow these steps to assign a value to a parameter:

1. In the batch dataflow graph, click one of the following icons:
   - dataset icon
   - filter operation icon

- exported table icon

Then click the **Create Parameterized Query** pop-up. The **Parameterize Query** modal window is displayed. The exact contents of the modal window depend on the icon from which you launch the modal window.

2. For each parameter that has been applied to an operation, in the **Value** column, enter the parameter value. For example, for the Month parameter, you can enter OCTOBER.

> **IMPORTANT:** When defining a parameter for a filter function, you can specify a string as the parameter value. For a string to be interpreted correctly, it must be enclosed in double quote marks. For example, if a column's data type is string, and you want to filter the string 10, you must enter "10" as the parameter value. Without the quote marks, the value is interpreted as an integer and the filter operation does not produce the expected results.

If you want the parameter to have no value, leave the **Value** column blank and click the check box in the **No Value** column.

> **EXAMPLE:** To parameterize the export file name, you can define two parameters (for example, firstDigit and secondDigit) for the file suffix such that the export file name is in this format:
>
> airlines<firstDigit><secondDigit>.csv
>
> When the suffix requires only one digit, you can assign no value to firstDigit and a numeral to secondDigit. Doing so creates files named airlines1.csv, airlines2.csv, and so on. When the suffix requires two digits, assign a numeral to both firstDigit and secondDigit. Doing so creates files named airlines11.csv, airlines12.csv, and so on

3. Click **SAVE**.

Example of parameterization

The following example illustrates how a batch dataflow is parameterized to create different export files based on the same batch dataflow.

The batch dataflow named FlightInfo in this example consists of these operations:

- Xcalar Insight points to a data source containing flight information from various air carriers and creates a table with four columns such as day of the week, air time, and so on.
- Smart type casting is used to cast the correct data type for each column.
- A filter operation is performed to include only rows from Sunday (1 in the DayOfWeek_ integer column).
- An export file is created with all the columns.

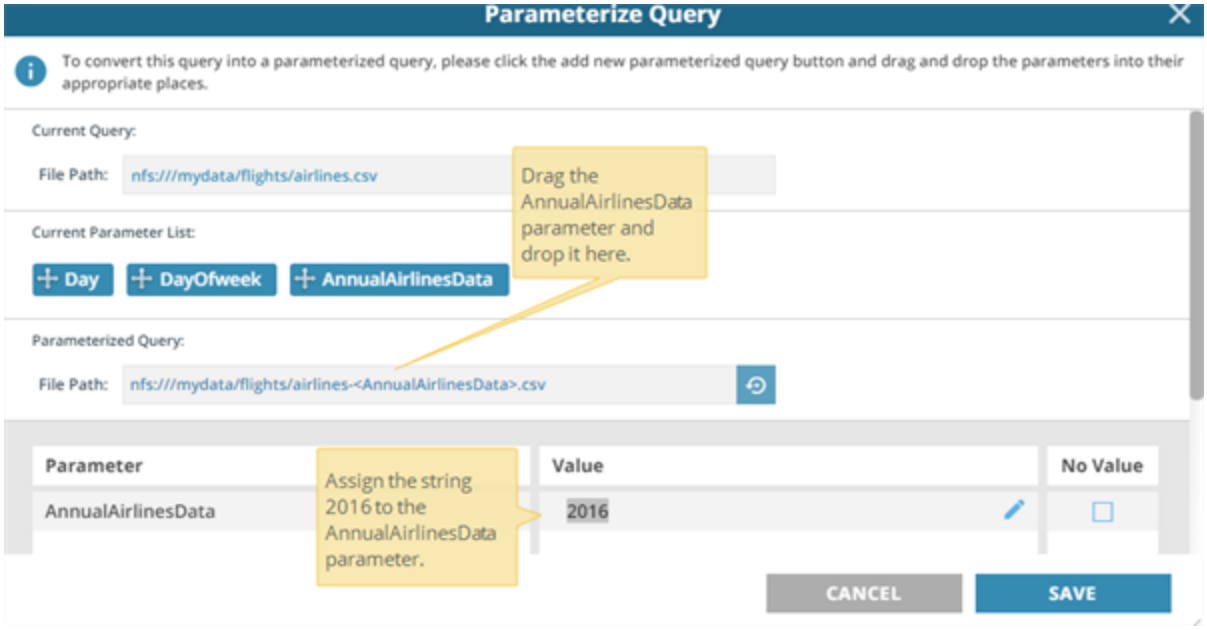
The following screenshot illustrates the batch dataflow graph:

Parameterizing the data source

You can parameterize the data source in a batch dataflow, which enables you to substitute a new pathname to the data source for the one currently used.

Follow these steps to parameterize the data source:

1. In the FlightInfo dataflow graph, create a parameter named AnnualAirlinesData.
2. In the FlightInfo dataflow graph, click the first icon and then click the **Create Parameterized Query** pop-up. The following modal window is displayed:

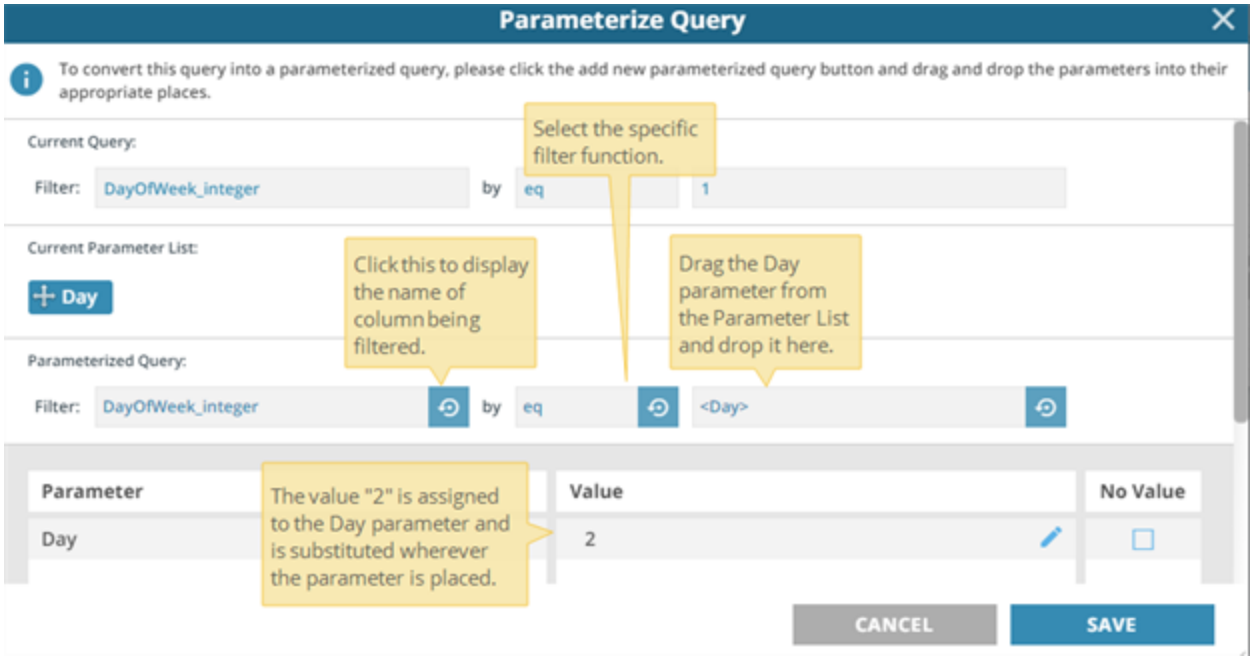


3. Drag and drop the AnnualAirlinesData parameter to the **File Path** field. This is the data source to which Xcalar Insight points when running the batch dataflow. In this example, Xcalar Insight points to nfs:///mydata/flights/airlines-2016.
4. Click **SAVE**. The first icon in the batch dataflow graph is displayed in green to indicate that it is parameterized.

Parameterizing the filter operation

To run the same batch dataflow but with other filtering criteria, follow these steps:

1. In the FlightInfo dataflow graph, create a parameter named Day.
2. In the FlightInfo dataflow graph, click the filter operation icon and then click the **Create Parameterized Query** pop-up. The following modal window is displayed:



3. In the **Parameterized Query** section, define the filter operation you want to use for this batch dataflow as follows:

    a. Select the column for the filtering. In this example, use the default value, which is the same column used by the current batch dataflow.

    b. Click the field following the word **by** to display a list of filter functions. In this example, select **eq**.

    c. Drag the Day parameter and drop it in the last field of this section.

    This operation filters the DayOfWeek_integer column to keep only the rows with the value equal to the value of the Day parameter.

4. For the Day parameter, type a number representing the day of week in the **Value** column. In this example, type 2 for Monday.
5. Click **SAVE**. The filter operation icon in the batch dataflow graph is displayed in green to indicate that it is parameterized.

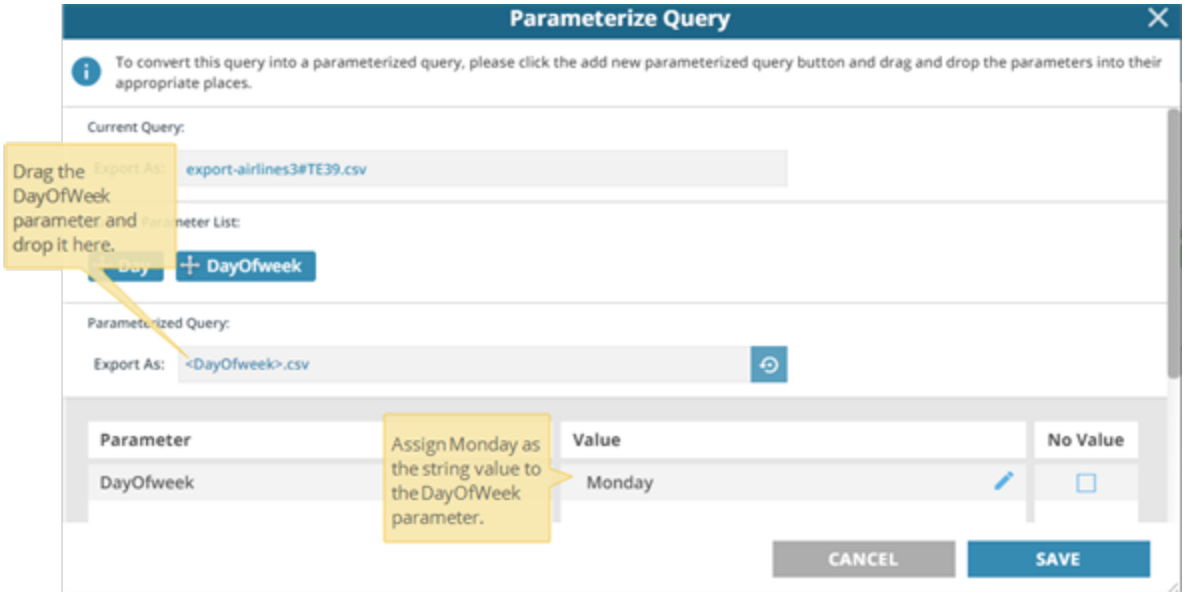
Parameterizing the export operation

You might want to paremeterize the export operation so that the export file is named in a way meaningful to you. You can name the export file based on the type of filter operation done in

the batch dataflow. For example, if the exported table contains only rows for flight data from Monday, you can parameterize the export operation to include the string Monday in the export file name.

Follow these steps to parameterize the export operation:

1. In the FlightInfo dataflow graph, create a parameter named DayOfWeek.
2. In the FlightInfo dataflow graph, click the exported table icon and then click the **Create Parameterized Query** pop-up. The following modal window is displayed:



3. Drag and drop the DayOfWeek parameter to the **Export As** field, which determines the export file name. You can include other characters in this field in addition to the parameter. For example, you can specify the following in the **Export As** field:

   airlines-<DayOfWeek>.csv

4. For the DayOfWeek parameter, type the string that will be used as a substitution in the export file name. In this example, type Monday.
5. Click **SAVE**. The exported table icon in the batch dataflow graph is displayed in green to indicate that it is parameterized. When you run this parameterized batch dataflow, the resultant table contains only flight information for Monday, and the table is exported to the file named Monday.csv.

The following screenshot illustrates the batch dataflow graph for FlightInfo after you finish parameterizing.

Before you run the batch dataflow, click the **Create Parameterized Query** pop-up for each green icon to verify that the parameters are defined properly. In this example, you can change the Day, DayOfWeek, and AnnualAirlinesData parameters to point to different data sources, filter based on different criteria, and name export files to reflect the purpose of running the dataflow.

Understanding memory usage

The XCE architecture makes optimal use of system resources (such as memory, CPU, and network resources). XCE can optimize memory usage even on a heterogeneous cluster where server nodes have different amounts of memory installed.

About XCE's use of the memory hierarchy

XCE's distributed shared memory (DSM) architecture automatically uses and manages the tiers of a memory hierarchy, which are shown in the following list:

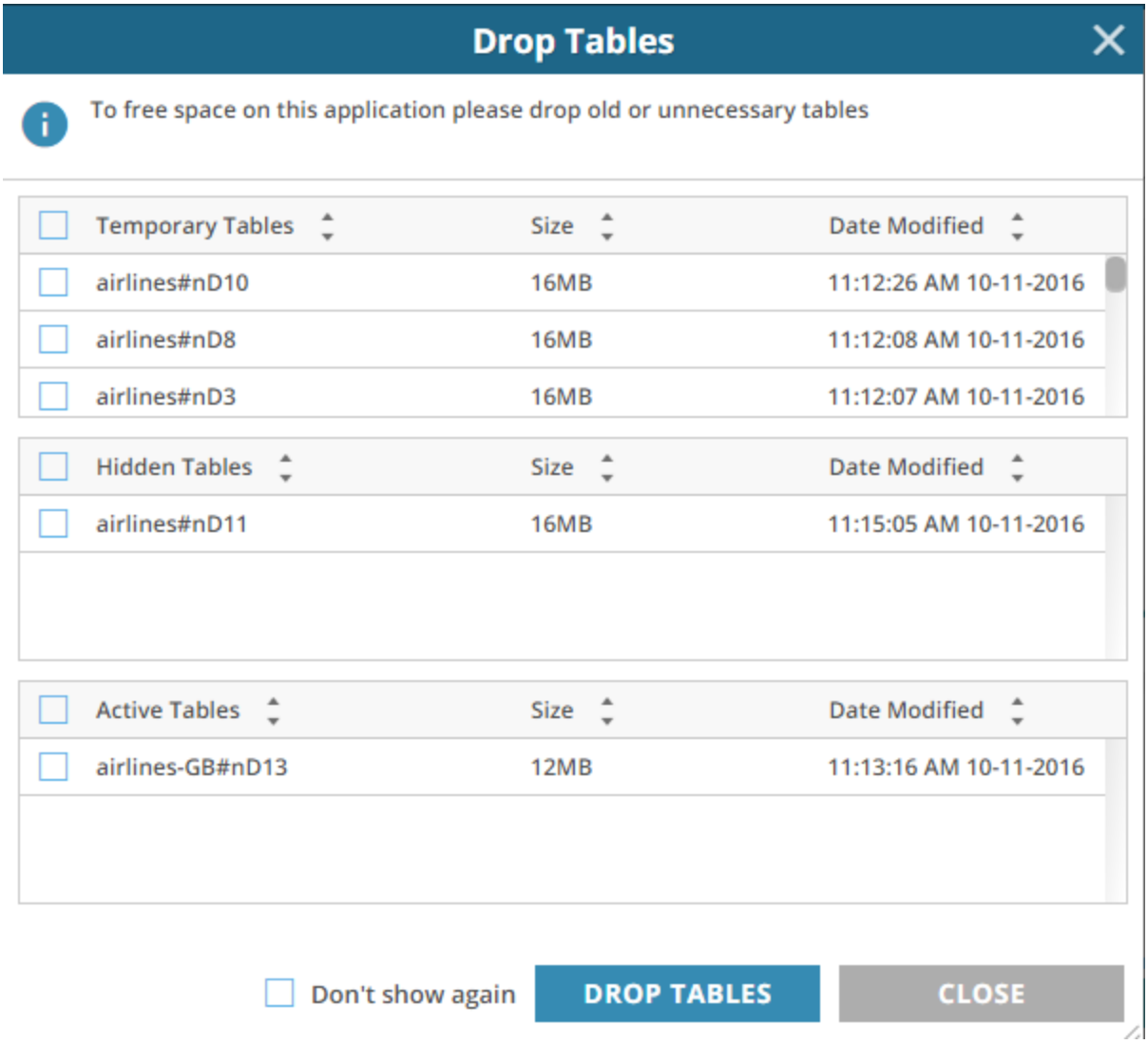
- DRAM
- Storage Class Memory (SCM)
- Solid state drives (SSDs)
- Hard disk drive (HDD)

XCE optimally distributes data and metadata across the cluster using sophisticated policies. When you interactively models dataflows, if the XCE cluster is low on memory, XCE ages and serializes data to lower tiers of the memory hierarchy. It also deserializes data when necessary. While all of this is accomplished without user intervention, Xcalar recommends that users conserve memory (with methods described in Optimizing memory usage) so that XCE can deliver the highest possible performance.

XDP gives you fine-grained control over the distribution of data and metadata across the cluster. Such control is particularly useful when you have to deal with non-uniform data distribution or coloration that occurs for real-world applications. Production systems often need to run optimized dataflows in a similar way as database administrators optimize SQL queries. Contact Xcalar if you need Xcalar's Field Technology Engineers to help operationalizing dataflows for you or provide training on operationalizing dataflows.

Understanding the low-memory notification

If XCE is low on memory, Xcalar Insight automatically displays a modal window similar to the following for every user who is logged in:



The tables shown in the modal window are the same as the ones listed in the **Tables** list. They are tables from all worksheets in your active workbook. As illustrated in this example, the tables listed do not consume a large amount of memory, and yet the user receives the notification because the available memory on the cluster has reached a low level. See Optimizing memory usage for information about what you can do to release or conserve memory.

Determining which table to drop

The following list describes the considerations when you determine which table to drop:

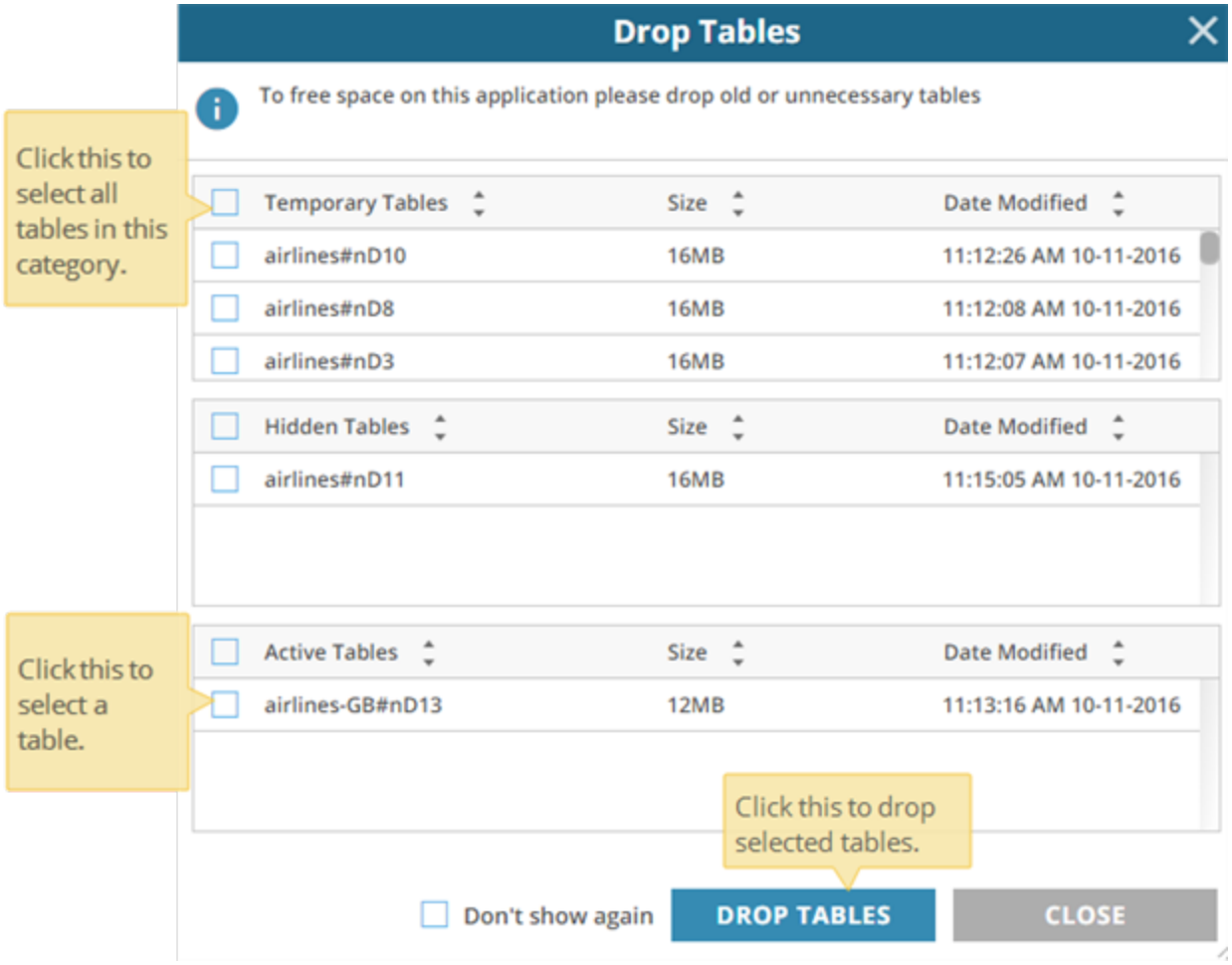
- From the **Drop Tables** modal window, you can see the amount of memory consumed by each table. The size information helps you decide the amount of memory that will be released due to the deletion.
- Temporary tables are created when you perform operations in a worksheet to cleanse data, transform data, and so on. They can accumulate fast over time, and it is likely that you do not need most of the temporary tables. For example, it is likely that you no longer need the temporary table resulting from a data type change. For more information about temporary tables, see Understanding and changing table statuses.
- Hidden tables are the ones you removed from the worksheet to make the worksheet easier to manage. Re-evaluate whether the table will ever be used again. Delete the hidden tables that are no longer useful for modeling your queries. For more information about hidden tables, see Understanding and changing table statuses.
- You might have multiple active tables across worksheets in your workbook. It is least likely that you want to drop active tables because they are the ones that you currently work on or worked on recently. However, it is acceptable to drop an active table if it is not needed anymore, for example, because another active table contains similar data.

> **IMPORTANT**: Dropping any kind of table is not reversible. You cannot undo the action of dropping a table.

Dropping a table from the Drop Tables modal window

Follow these steps to drop tables:

1. Click the **System** icon on the tool bar to display the **System** menu.
2. Click **Release Memory** to display the **Drop Tables** modal table.
3. In the **Drop Tables** modal, select the tables to drop. The following screenshot illustrates how to select the tables to drop. Before selecting a table, you can sort the tables by name, size, or date modified.



4. Click **DROP TABLES**.

Optimizing memory usage

Xcalar Insight displays a low-memory notification to all logged-in users when XCE reaches a pre-defined level of memory consumption. The Xcalar administrator might also notify users to free up memory before the automatic notification is sent. It is important for you to optimize memory usage to ensure that XCE can run at the highest performance level possible. See Understanding memory usage for more information about the low-memory notification and what steps to take in response to the notification.

Best practices for conserving memory

Xcalar Insight users can follow these best practices to conserve memory:

- Reduce the number of tables.
- Reduce the number of fields in the **Data Browser**.

> **NOTE**: Fields that come from your dataset cannot be removed individually. For example, if your dataset has two fields: **Carrier** and **Description**, you cannot remove only the **Carrier** field and keep the **Description** field to try to reduce the number of fields.

More detailed information about these best practices are provided in the sections that follow.

Effects of tables on memory consumption

Each table in your workbooks consumes memory regardless of the table's status, which can be active, hidden, or temporary. Keeping only the tables required for modeling your queries helps conserve memory.

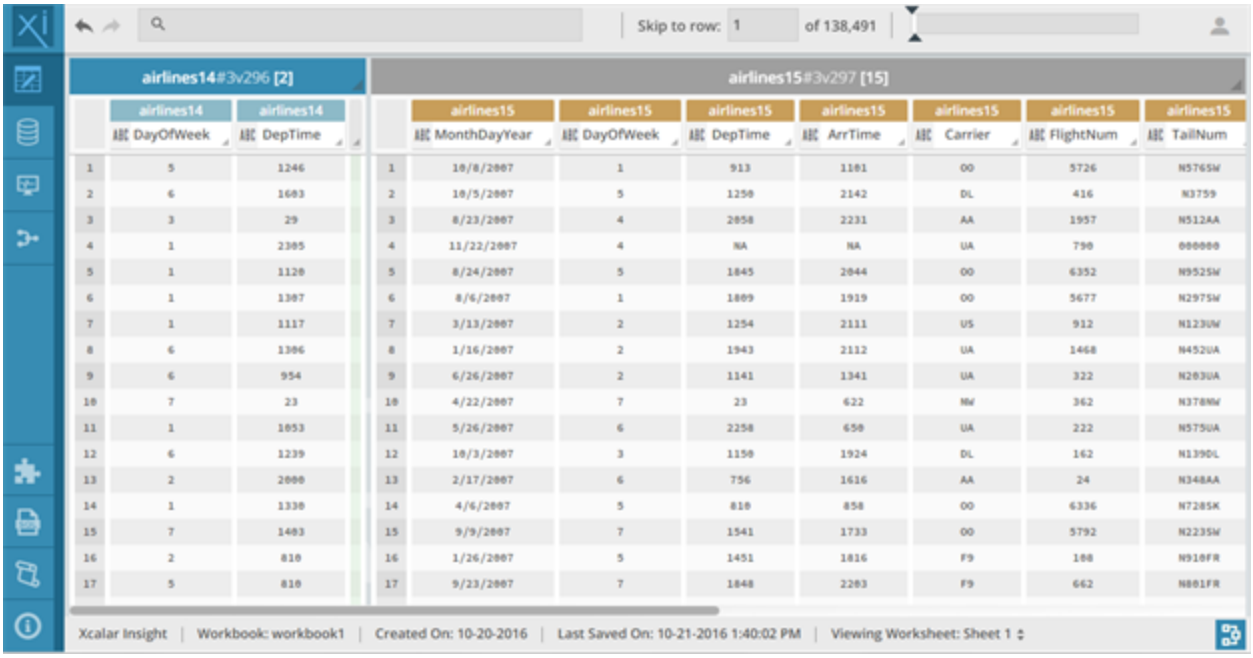
Methods for dropping tables

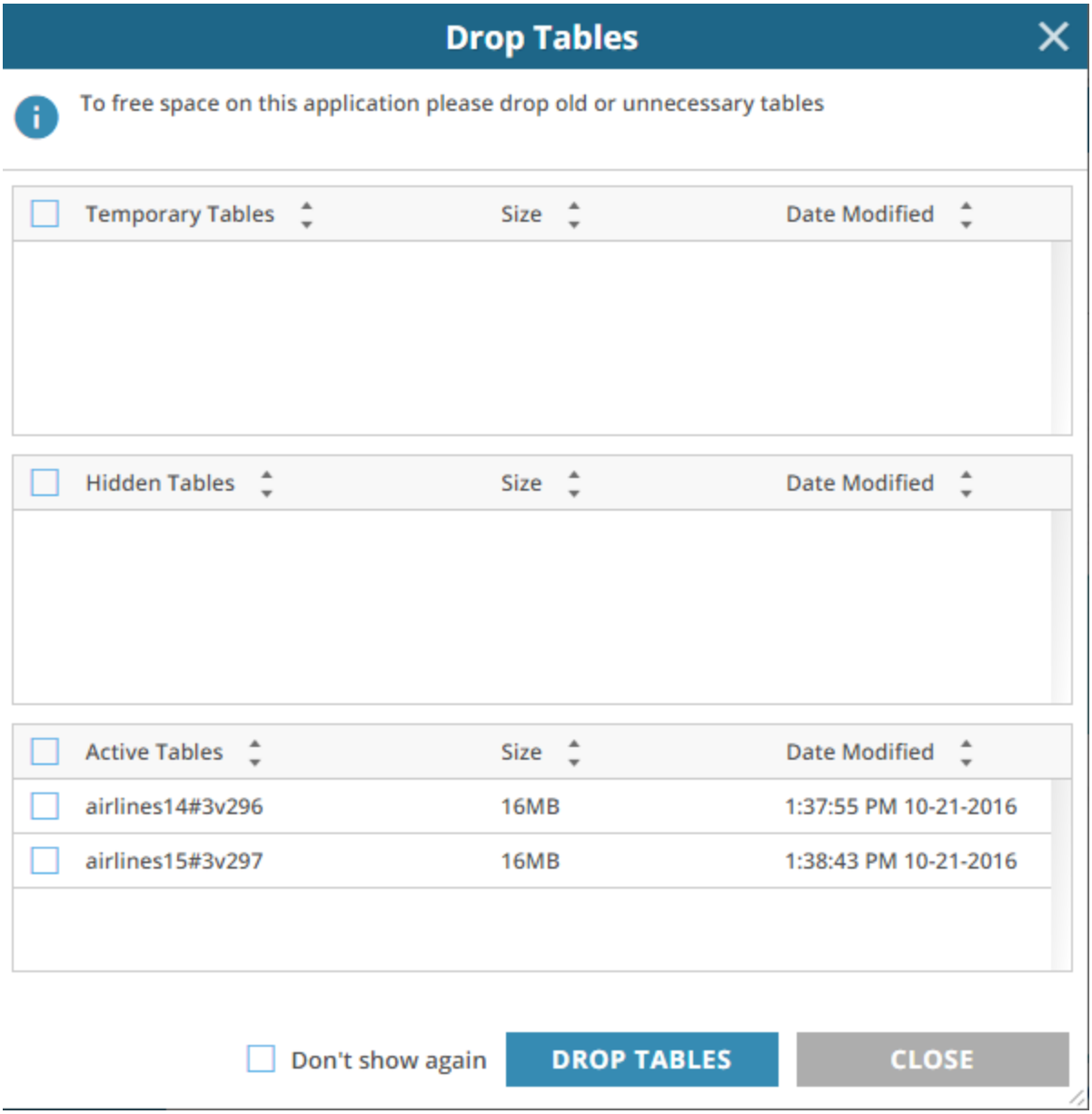
You can drop tables with these methods:

- Display the **Tables** panel. Then select the tables to drop. See Understanding and changing table statuses for more information about the **Tables** panel.
- In the query graph, right-click a table icon to display a pop-up menu. Select **Drop Table** to remove the table from the worksheet. This method enables you to see how the table was created or modified when you performed the modeling.
- In the active worksheet, locate the table to drop, and right-click the table title bar to display a drop-down menu. Then select **Drop Table**.
- In the **Monitor** tab, click the **Drop Tables** button to display the **Drop Tables** modal window, which lists all tables in your workbook and the amount of memory used by each . See Understanding memory usage for more information about the **Drop Tables** modal window.

Effects of table columns on memory used

The number of columns in a table has no significant effect on memory consumption. For example, from the same dataset reference, you can create two tables: one with 2 columns and one with 15 columns. Both tables use the same amount of memory, which is 16 MB. The following screenshots show the columns in the tables and the amount of memory used by each table.





Because the number of columns is not correlated with memory consumption, you cannot free up memory by deleting table columns.

Effects of fields in the Data Browser on memory used

Each table has a **Data Browser** associated with it. The **Data Browser** lists all the fields that are already in the table or can potentially be added to the table. Initially, each table you create from a data source consumes the same amount of memory as each table's **Data Browser** contains only the fields from the data source. These fields are listed in the prefixed fields section of the

Data Browser. This group of prefixed fields consumes a fixed amount of memory, regardless of how many fields are in the group.

After you perform operations on a table, derived columns are created in the resultant table and these columns are also added in the resultant table's **Data Browse**. They as derived fields. Each derived field consumes additional memory.

> **NOTE**: A table resulting from a batch dataflow does not have prefixed fields. The **Data Browser** for this table consists of only derived fields.

> **Example**: A two-column table is created from a dataset with two fields: **Destination** and **Distance**. If no operations have created columns in this table, the table uses 2.2 MB of memory. Suppose you change the **Distance** column's data type to integer. A new table is created with a derived column named **Distance_integer**. In the **Data Browser** for this table, the field **Distance** remains in the prefixed fields section, and a new field named **Distance_integer** is added in the derived fields section. If you use the **Drop Tables** modal window to check memory usage, you can see that the new table consumes 4.2 MB instead of 2.2 MB of memory because of the derived field added. The more columns your operations create, the more derived fields are added to the **Data Browser**. The derived fields increase the amount of memory consumed.

> **NOTE**: As in the case of other columns, deleting a derived column does not free up memory. For example, if you delete the **Distance_integer** column in the example above, the table continues to use 4.2 MB of memory. Memory used is not reduced because the derived field, **Distance_integer**, remains in the **Data Browser**.

Removing fields from the Data Browser

Because the number of fields in the **Data Browser** has an impact on memory used, the logical step to conserve memory is to reduce the number of fields in the **Data Browser**. However, remember that in the context of memory consumption, the collection of fields from the source

dataset (that is, the prefixed fields) is considered one unit. You cannot remove some fields and keep some fields in that collection. You can decide either to keep or to eliminate the collection in its entirety.

The contents of the **Data Browser** are not directly editable. To change the contents, use the Projection Mode feature, which enables you to keep only columns of interest in a table and to remove unnecessary ones from the **Data Browser**.

> **NOTE**: The term "projection" originates from relational algebra. Its meaning in Xcalar Insight should be clearly distinguished from its meaning in daily usage.

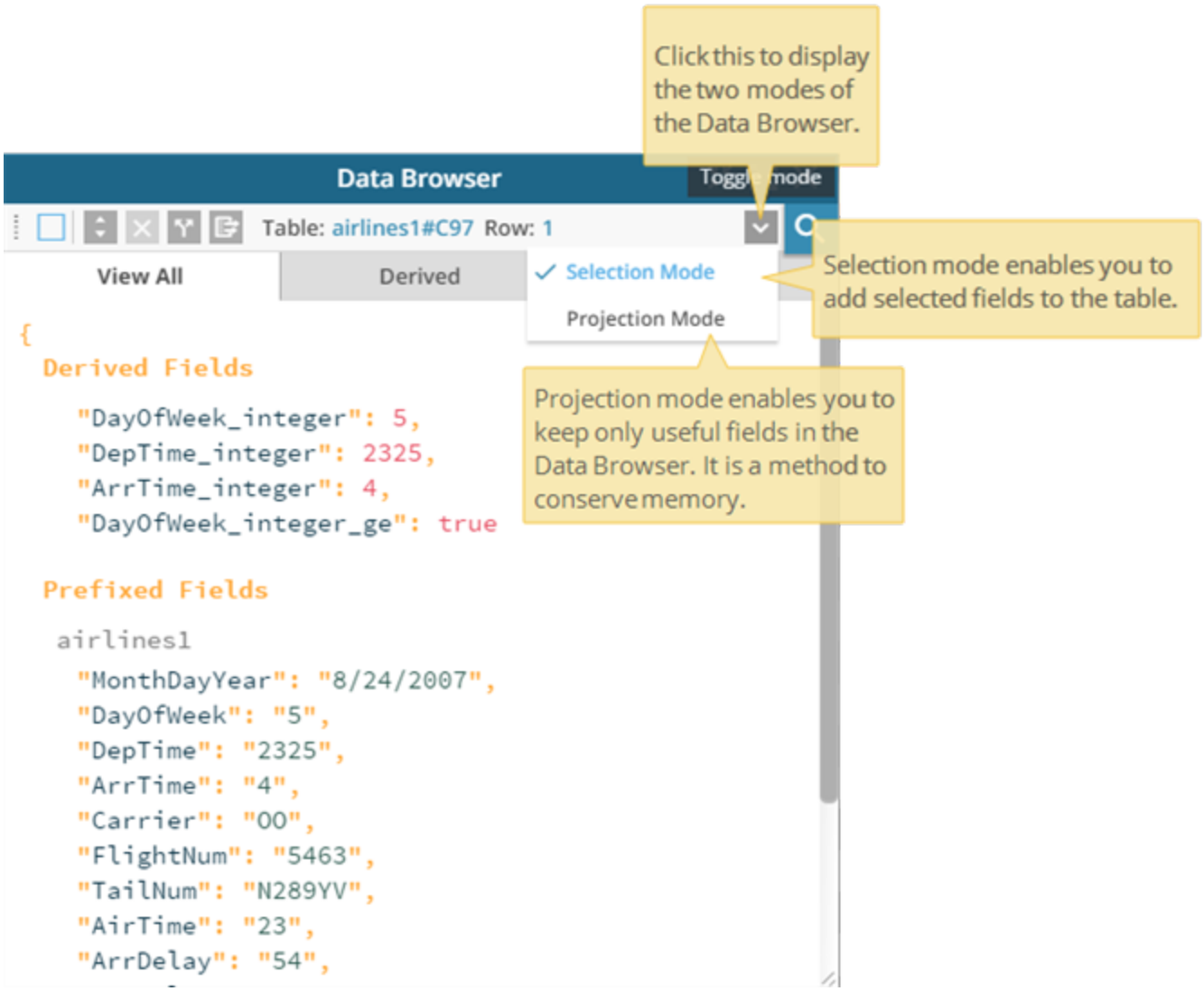
Understanding Projection Mode

Projection Mode in the **Data Browser** enables you to eliminate unnecessary fields that Xcalar Insight keeps in memory for a table. You can choose to keep only the fields that interest you in the table and in the **Data Browser**. In doing so, you can reduce the amount of memory used by the table.

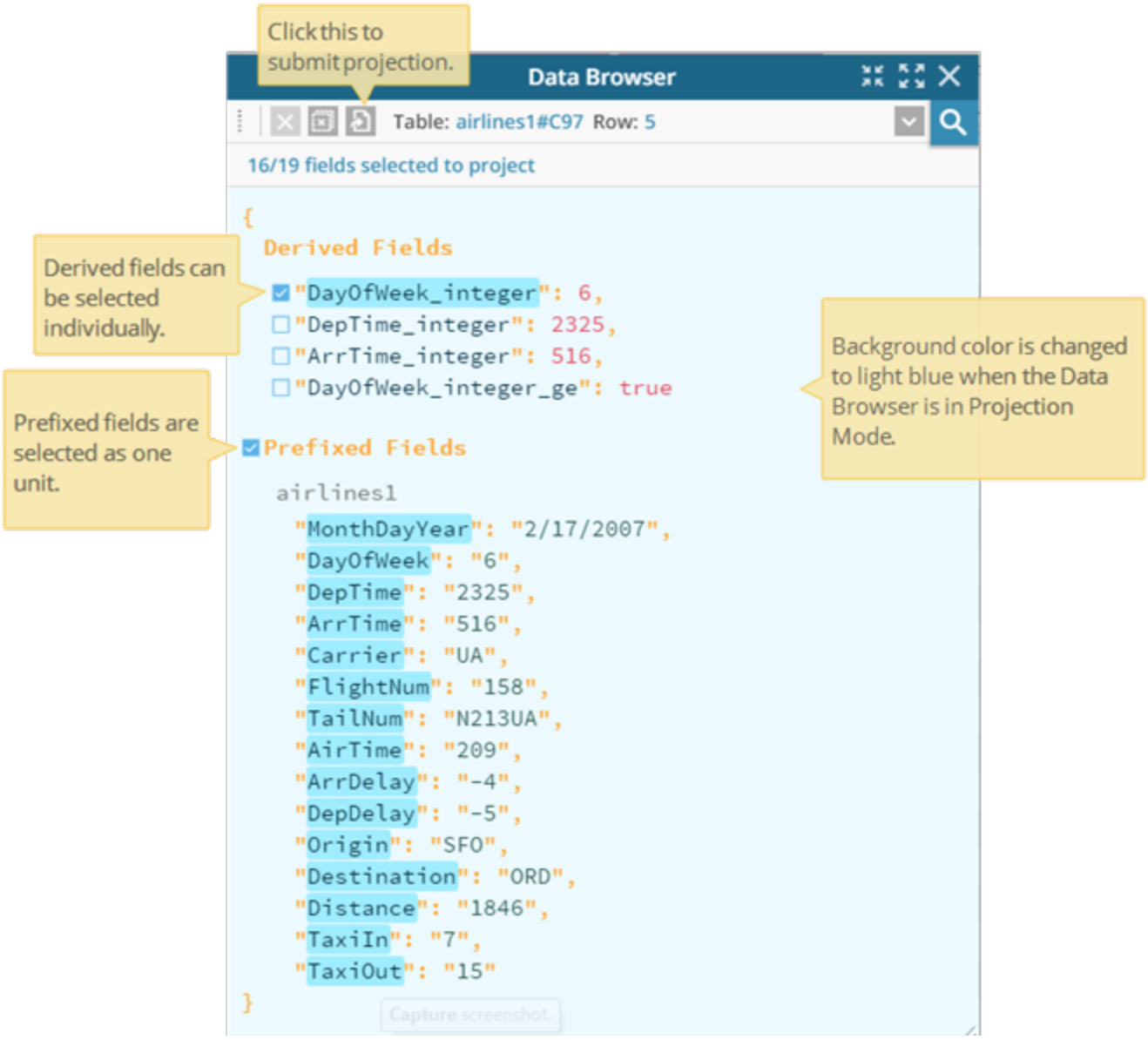
Using Projection Mode

Follow these steps to use Projection Mode to keep memory consumption low for a table:

1. Double click anywhere in the **DATA** column of the table to display the **Data Browser**.
2. Click the down arrow in the **Data Browser** to select **Projection Mode**. The following screenshot shows the location of the arrow.



3. Click the check box preceding the field name if you want to keep the field in the resultant table. For example, click **DayOfWeek_integer** if you want to keep it in the resultant table. You might have to use the scrollbar to view all field names. The following screenshot



4. Click the **submit projection** button. A new table is created with only the columns you chose in the **Data Browser**. As in the case of other operations, you can use the **Undo** button to revert the table to the previous version before the projection.

You can perform projections on any columns to reduce the number of columns in the active table so that the table is less cluttered. However, the main purpose of projection is to optimize memory usage. Therefore, pay attention to the number of fields being eliminated from the **Data Browser** when you decide which column to choose for projection.

Example of projection on derived fields

The **Data Browser** shows 19 fields, and 4 fields are derived fields such as **DayOfWeek_integer**, **DepTime_integer**, **DepTime_integer**, and so on. These derived fields are created due to operations such as Map. The rest (15 fields) are from the collection of fields in the source dataset. They are **MonthDayYear**, **Distance**, **DepTime**, and so on. The names of these fields have a prefix, such as **airlines.1**.

Suppose you perform a projection on **DayOfWeek_integer** and **DepTime_integer**, the resultant table contains only these columns. All fields except for **DayOfWeek_integer** and **DepTime_integer** are eliminated from the **Data Browser**. The resultant table now consumes much less memory than the table before the projection. For the resultant table, the XCE cluster no longer needs memory to track all 4 derived fields and the collection of fields from the data source.

Example of projection on prefixed fields

You cannot choose individual entries from the list of prefixed fields. The set of prefixed fields is eliminated or retained as one unit through projection. The prefixed fields consume a fixed amount of memory, which is about the same as that consumed by a derived field.

Suppose you are only interested in **airlines1::Distance** and **airlines1::MonthDayYear**, you cannot project these fields and discard other prefixed fields such as **airlines1::Carrier** and **airlines1::FlightNum**. The **Data Browser** must keep the entire collection of prefixed fields. After the projection, the unselected derived fields no longer exist in the table or the **Data Browser**. This frees up memory that otherwise would be needed to track 4 derived fields.

Using the Monitor

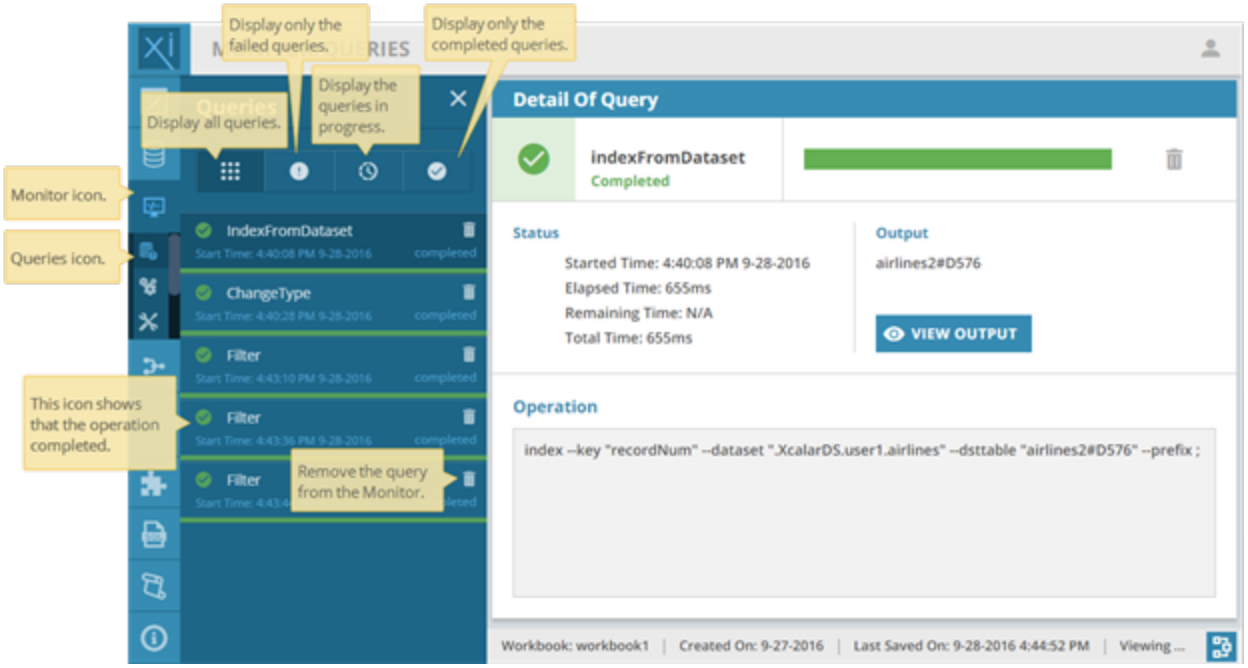
The Monitor icon ( ) in the Xcalar Insight menu enables you to monitor the status of XCE and specify the environment in which you use Xcalar Insight. This section describes the tasks you can perform with the Monitor.

> **NOTE:** If you log in as administrator, a **Setup** icon ( ) is available in the Monitor. See Using Setup (XDP administrators only) for information about how to set up the cluster through this icon.

Monitoring query status

Xcalar Insight keeps a record of all operations since you logged in and displays information about them. The operations include the ones you have undone. Follow these steps to display query status:

1. Click to start the Monitor. By default, the **Queries** panel is displayed. The following screenshot illustrates the icons in the **Queries** panel that control how the list of queries are displayed.



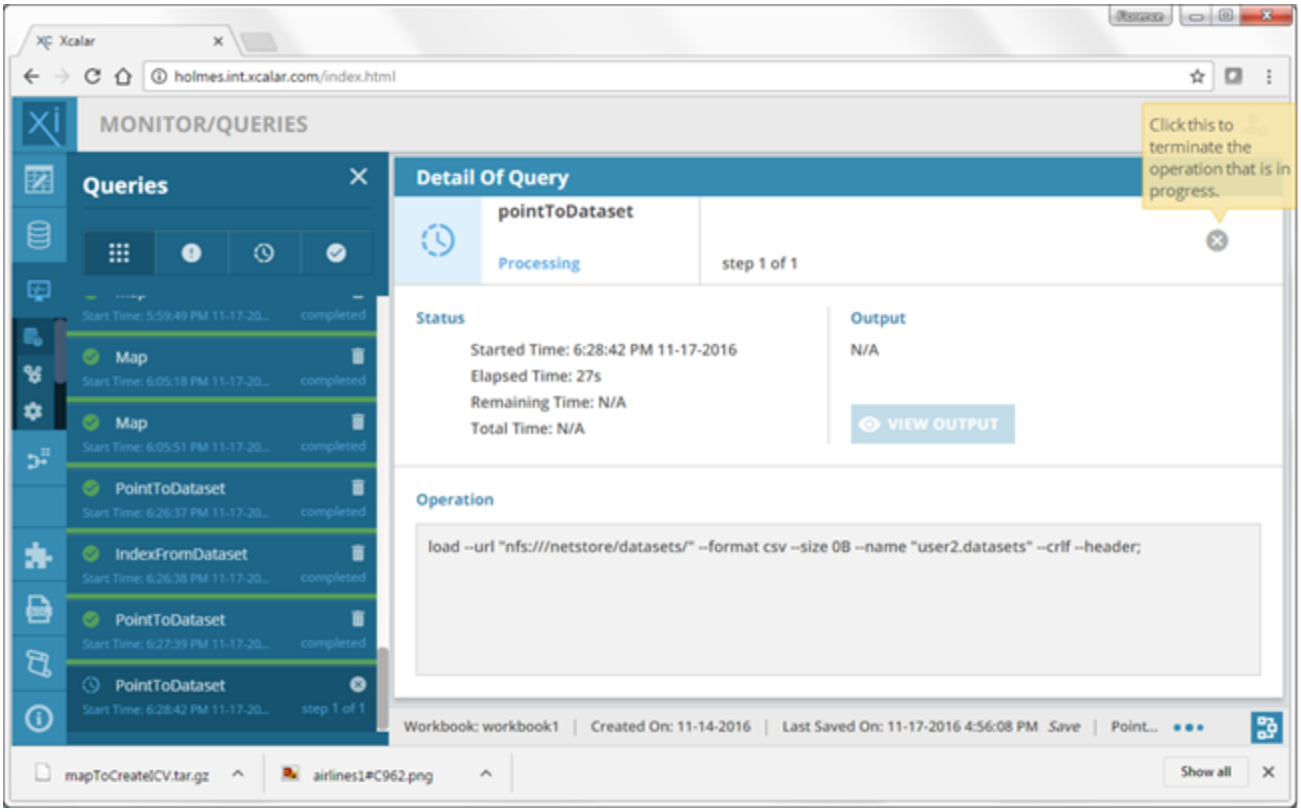
> **NOTE:** The icon for removing a query only affects the queries list in the **Queries** panel. It does not affect the output of the query. For example, a table created by a Filter function remains in the worksheet even after you remove the Filter function from the **Queries** panel.

2. Select a query in the panel to see detailed information about it. For example, you can see how long the operation took to complete.
3. You can click **VIEW OUTPUT** to go to the table in the worksheet. If it is a hidden table, it is added back to the worksheet as an active table. You cannot click **VIEW OUTPUT** if the output of a query is a table that has been dropped.

> **NOTE:** Because a query that has been undone is preserved in the list of queries, you can select an undone query to find its output, and then put the temporary table created by the query back to the worksheet.

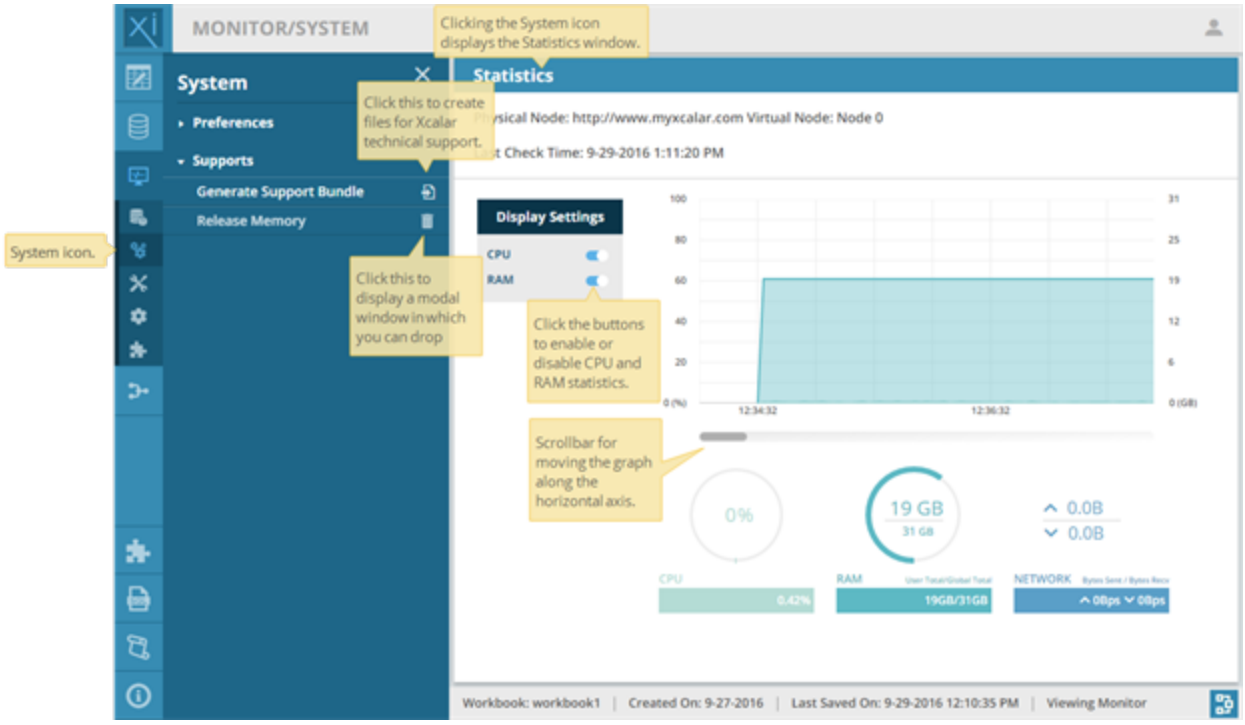
Canceling a query

If for any reason you want to terminate a query that is in progress, display the running queries in the **Queries** panel. Select the one you want to terminate and then click the Cancel icon either in the **Queries** panel or the **Detail of Query** window. The following screenshot shows an operation (pointing to a data source) in progress and the location of the icon for terminating the operation.



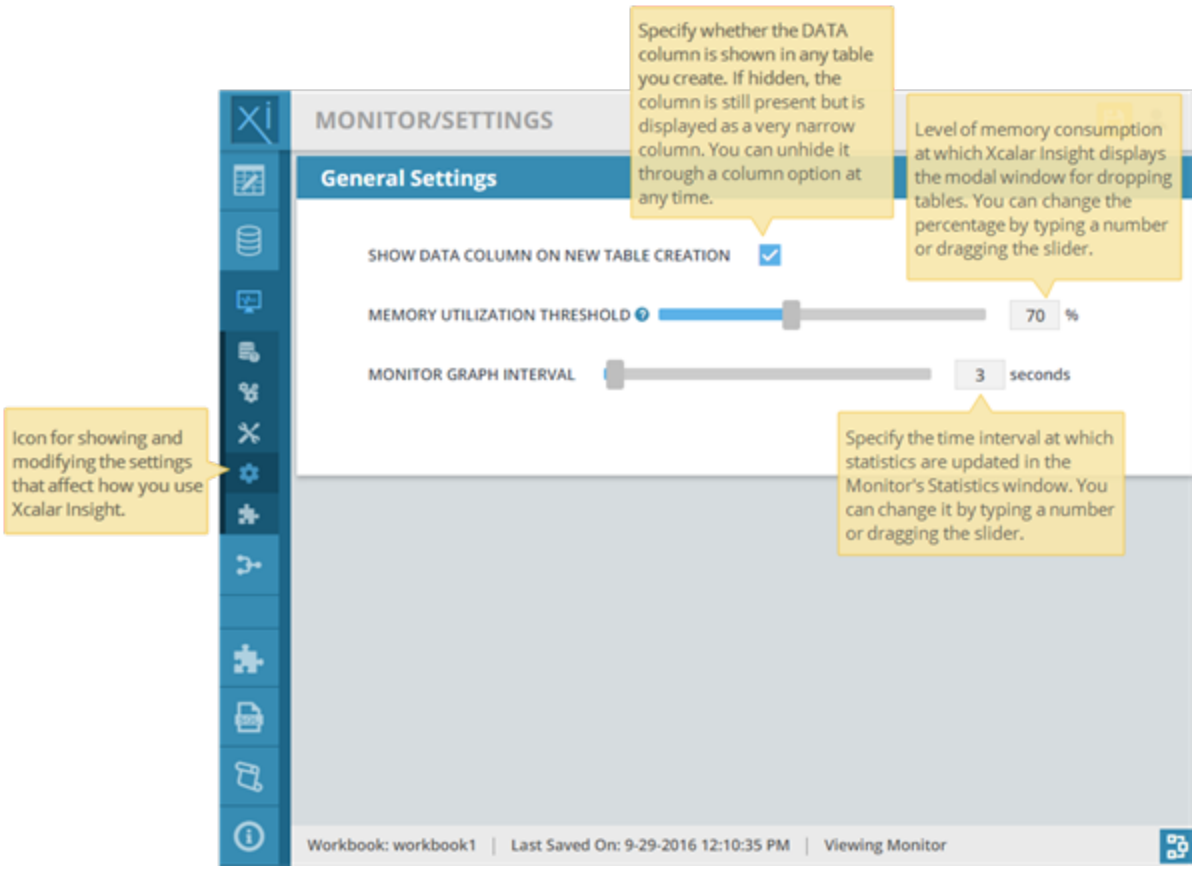
Monitoring the system

The Monitor provides an icon for you to display XCE statistics, generate files for Xcalar technical support, and release memory. The following screenshot shows the location of the **System** icon in the Monitor. The screenshot also illustrates how you can use the icons in the **System** panel to perform various tasks.



Setting the environment for Xcalar Insight

The Monitor provides a **Settings** icon for you to customize the environment in which you run Xcalar Insight. The following screenshot illustrates how you use the icons in the General Settings window to make changes.



Using Setup (XDP administrators only)

As an XDP administrator, if you click the Monitor icon () in the Xcalar Insight menu, you can perform all functions described in Using the Monitor. In addition, you can click the Setup icon () to configure XCE parameters and to manage the cluster.

Configuring parameters

The XCE parameters define how your cluster works, and you can set the values for some of the parameters. In addition, Xcalar technical support might instruct you to fine-tune some parameters to improve performance to suit your particular environment. Parameter values set in the Monitor take effect immediately and are persistent across cluster restarts.

To configure XCE parameters, click **Configuration** in the **Setup** panel. To change the value of a parameter, simply enter a new value in the **New Value** field and then click **CONFIRM & SAVE**. If you want to restore the parameter to its default value, click  .

> **NOTE:** If a **Config Parameter Name** field does not contain a name, you can enter a parameter name in the field under the guidance of Xcalar technical support. Do not use this field unless you are told to do so by Xcalar.

The following list explains the parameters that can be configured:

- **BufferCachePercentOfTotalMem**: XCE uses blocks of memory that can be allocated and freed quickly when running various processes. These memory blocks collectively form a buffer cache. This parameter specifies the percentage of total memory used as the buffer cache, which affects how much memory is left for dataset processing. The higher the buffer cache percentage, the less memory is available for XCE to use when trying to point to data sources. Xcalar recommends that you leave the parameter at its default, which is 60%. Contact Xcalar to determine whether adjusting the parameter value results in better performance.
- **MaxInteractiveDataSize**: It sets the maximum dataset size when the user selects **Entire**

file/folder under **Advanced Options** in the **Point to New Data Source** window. The default is 1 TB; there is not an upper limit for this value. Both the current value and the new value are displayed in bytes.

For more information about **Advanced Options** in the **Point to New Data Source** window, see Dataset size

- **XcalarRootCompletePath**: It displays the path to the root directory, which is /var/opt/xcalar. It cannot be changed.

Using Xcalar Insight as another user

You can switch to another user's account so that you can work on the user's behalf. You can perform all the operations that the user can perform, If the user is already logged in, switching to that user's account logs the user out of Xcalar Insight.

After you finish working in Xcalar Insight as a user, you can return to the administrator's account.

Follow these steps to use Xcalar Insight as another user:

1. In the **Setup** panel, click **Use Xcalar As** if a list of users is not already displayed. You can also click the **Refresh** icon to update the list.
2. Click a user name to switch to that user.

> **TIP:** You can type a partial user name in the **Search users** field to display only the user names that contain the string entered. For example, typing the string USER displays user1 and User2, but not john, in the list.

If the selected user is logged in, Xcalar Insight displays a confirmation message asking if you want to log out the user.

After you switch to the other user (for example, a user named test), a message similar to the following is displayed in the lower right corner of the Xcalar Insight window:

```
This is what Xcalar Insight looks like to: test
```

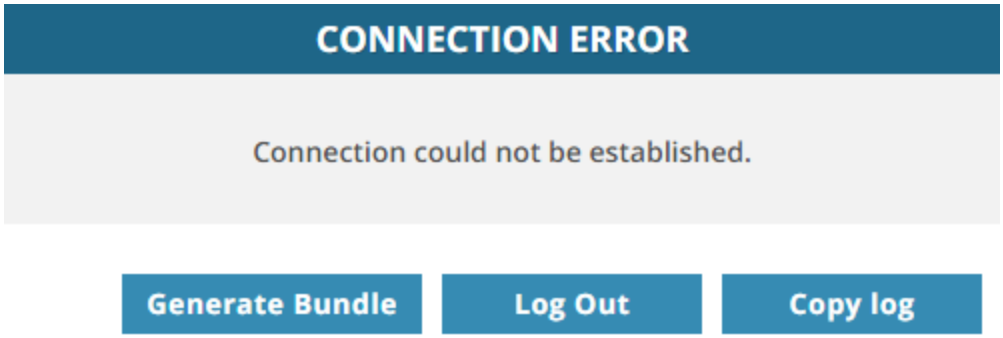
To return to the administrator's account, click the **Cancel** icon as shown in the following screenshot:

Stopping and starting the cluster

After you install XDP, the cluster is started automatically. You need to start the cluster manually only after one or more nodes crash or after you stop the cluster for maintenance or troubleshooting.

To stop the cluster, click **Stop Cluster** under **Configuration** in the **Setup** panel. All nodes are stopped at the same time. The following list describes the effects of stopping the cluster:

- Xcalar Insight displays the following message for all the logged in users:



- Operations in progress are canceled.

To restart the cluster, click **Start cluster** under **Configuration** in the **Setup** panel. The cluster can be started this way only if all of them are currently stopped. If one or more nodes are not stopped, the following message appears:

```
One of more of your XCE instances is currently running. Please restart
the cluster instead.
```

To work around this error, click **Restart Cluster** under **Configuration** in the **Setup** panel.

Viewing log messages

Updating license

Follow these steps to update your license:

1. In the **Setup** panel, expand the list of commands under **Configuration** if it is not expanded already.
2. Click **Update License**.
3. In the **Update License** modal window, enter the license key.
4. Click **CONFIRM**.

Reference

This section provides additional information that helps you navigate the Xcalar Insight user interface.

Xcalar Insight window

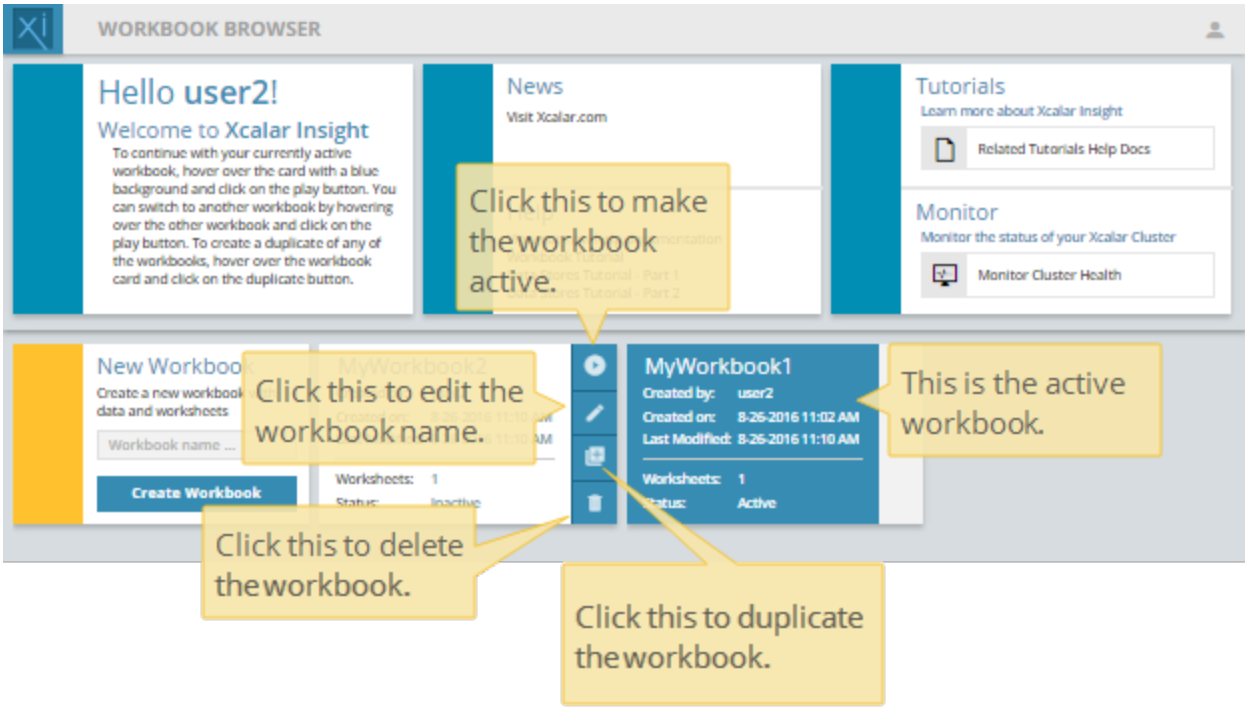
This section provides an overview of the Xcalar Insight window. The exact user interface elements depend on whether the worksheet contains a table and whether a column in the table is selected. The ones described in this section are visible regardless of the selected tab.

The following sample screen shows the location of each user interface element.

Workbook Browser

Clicking the **Workbook** icon in the upper left corner of the Xcalar Insight window opens the **Workbook Browser**. You can create a workbook as described in Creating a workbook as your workspace. The following sample screen shows the buttons in the **Workbook Browser** for these tasks:

- Activating a workbook
- Editing the workbook name
- Duplicating a workbook
- Deleting a workbook



About activating a workbook

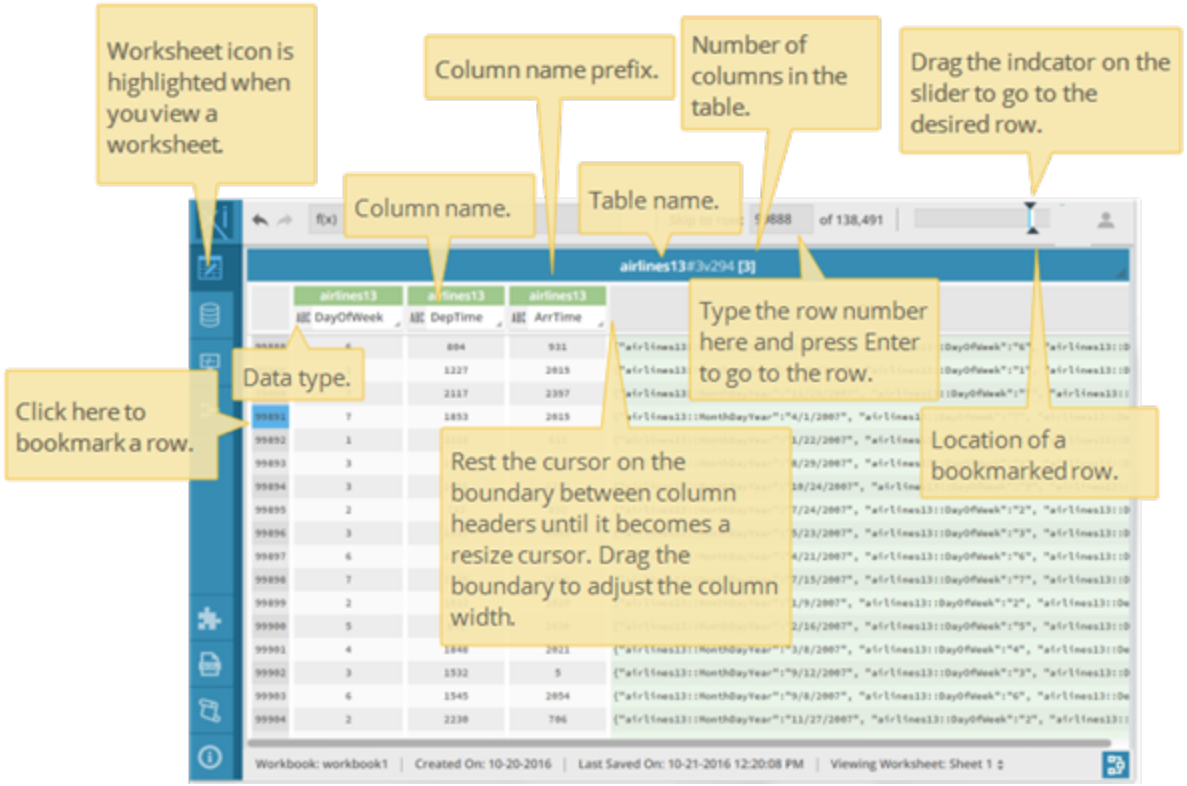
Upon creation, a workbook is inactive. You must activate the workbook before you can use it as your workspace.

After you log out and log in again to Xcalar Insight, the workbook that was active before you logged out remains active. However, if the Xcalar cluster is powered off and then powered on again, none of the workbooks are active and you must activate one of the workbooks in the **Workbook Browser**.

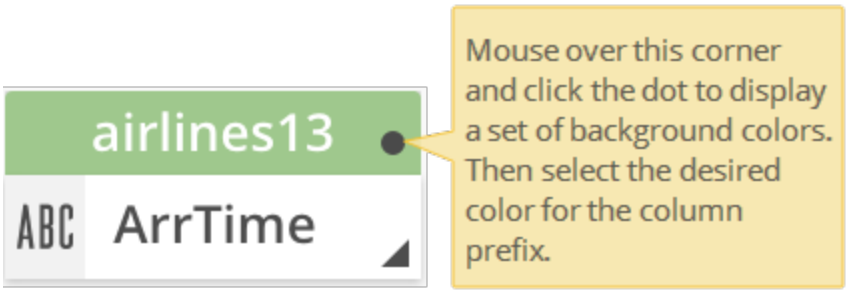
Worksheet window with tables

This section provides an overview of the **Worksheet** window that contains tables. The exact contents of the window depend on your tables and might be different from the one described in this section.

The following screenshot explains the purpose of each user interface element in a worksheet containing one table.



The column prefix in this example has a green background color. You can display a list of colors and select the desired color as shown in the following partial screenshot.



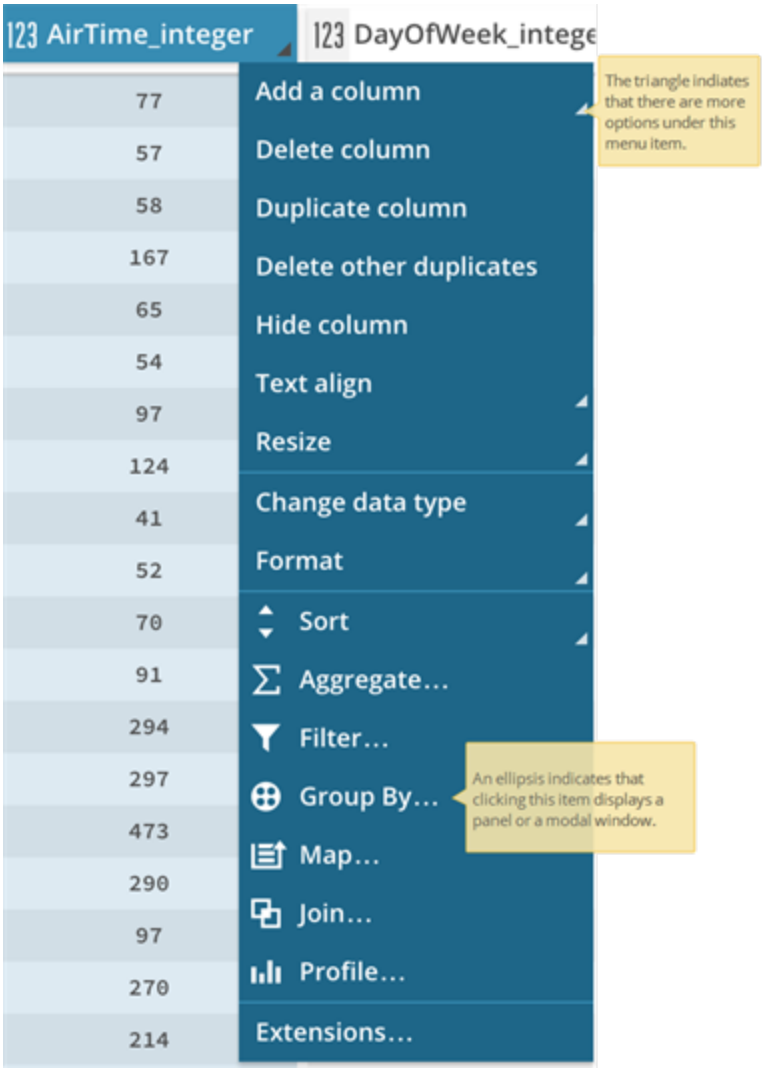
> **TIP:** The background color is applied to all column headers with the same prefix. Using different colors for different prefixes makes it easy to identify the origins of the columns. For example, if after a Join operation, some column names are prefixed with airlines13 and some are prefixed with airlines14, apply the green color to the airlines13 prefix and purple to the airlines14 prefix.

Menus displayable in the worksheet

Several menus can be displayed when you view a worksheet containing one or multiple tables.

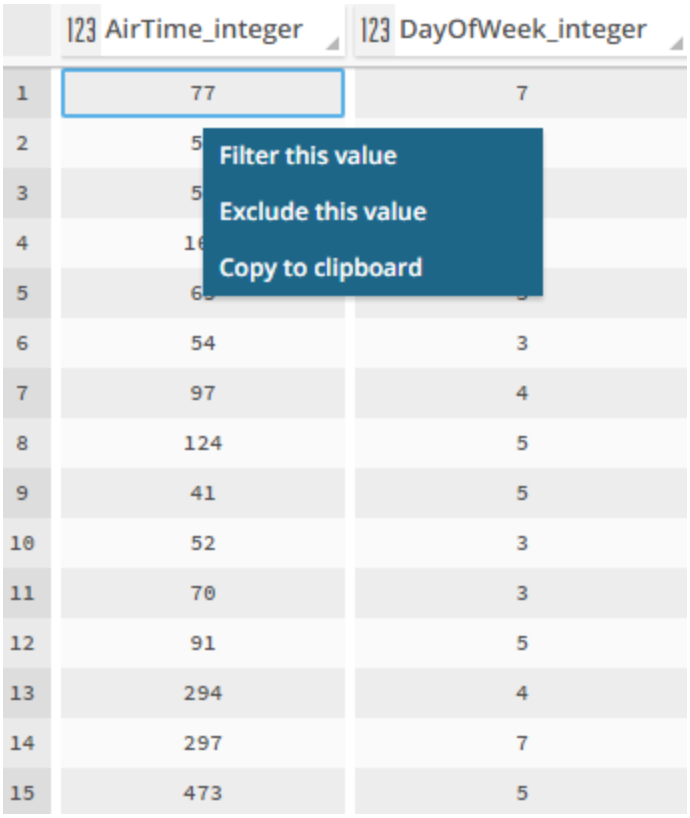
Column drop-down menu

Most of the data operations can be started from a column drop-down menu. The exact options in this menu depend on the column's data type. To display the menu, you can right-click the column header or click the triangle in the lower right corner of the column header. For simplicity, Xcalar Insight documentation instructs you to right click the column header to display the menu. The following partial screenshot shows an example of the column drop-down menu.



Cell pop-up menu

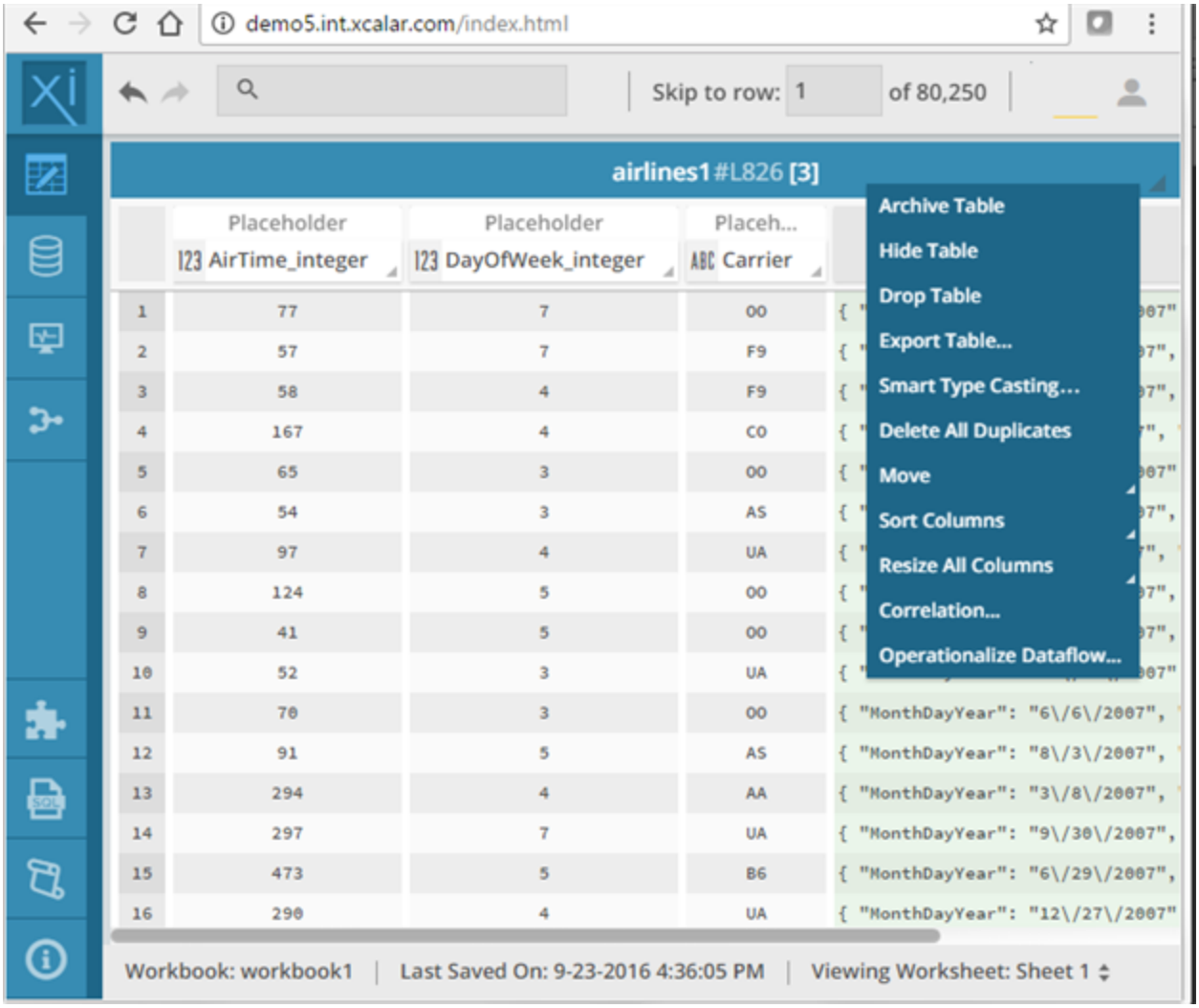
You can right click on a table cell to display a pop-up menu. The exact options in this menu depend on the column's data type. The following partial screenshot shows an example of a cell pop-up menu. The options in this menu are explained in Cell options.



	AirTime_integer	DayOfWeek_integer
1	77	7
2	5	
3	5	
4	1	
5	6	
6	54	3
7	97	4
8	124	5
9	41	5
10	52	3
11	70	3
12	91	5
13	294	4
14	297	7
15	473	5

Table drop-down menu

You can start operations that affect the entire table from the table drop-down menu. To display this menu, click anywhere in the title bar of the table (but not on the table title) or click the triangle in the lower right corner of the table title bar. For simplicity, Xcalar Insight documentation instructs you to click the table title bar. The following partial screen shows an example of the table drop-down menu. The options in this menu are explained in Table options.



How row bookmarking works

You can click a row number to bookmark a row. Clicking a bookmarked row number again removes the bookmark.

Bookmarking a row means that a marker representing the row is placed on the slider. To quickly locate a bookmarked row, click the marker on the slider.

Data Browser

Each table contains a default column named DATA. You can move your cursor to any row in this column and double click to display the **Data Browser**.

> **NOTE:** The DATA column can be hidden due to a setting in the **General Settings** window of the Monitor or the **Hide column** option in the column drop-down menu. For information about the Monitor, see Setting the environment for Xcalar Insight. To view the DATA column, simply click the DATA column header to display the column drop-down menu and select **Unhide column**.

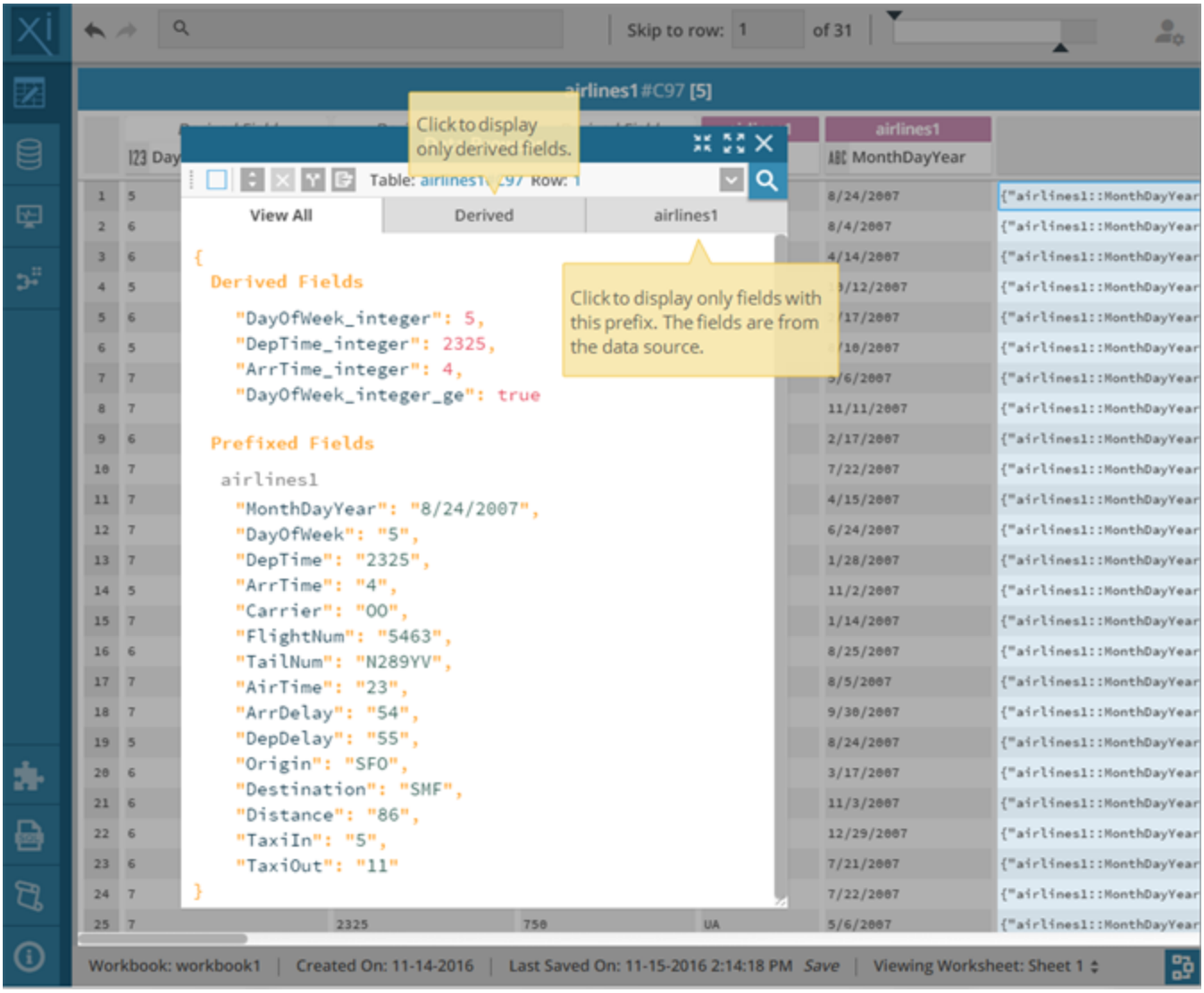
Contents of the Data Browser

The browser displays two kinds of fields:

- Prefixed fields, which are fields from the data source. If the table is a result of one or multiple Join operations, multiple sets of prefixed fields from multiple data sources are displayed. Each set has its own prefix.
- Derived fields, which are created through various operations. They do not have a prefix.

> **NOTE:** A table resulting from a batch dataflow contains only derived fields. Its **Data Browser** does not contain any fields from a data source. For more information about tables created from batch dataflows, see Using batch dataflows.

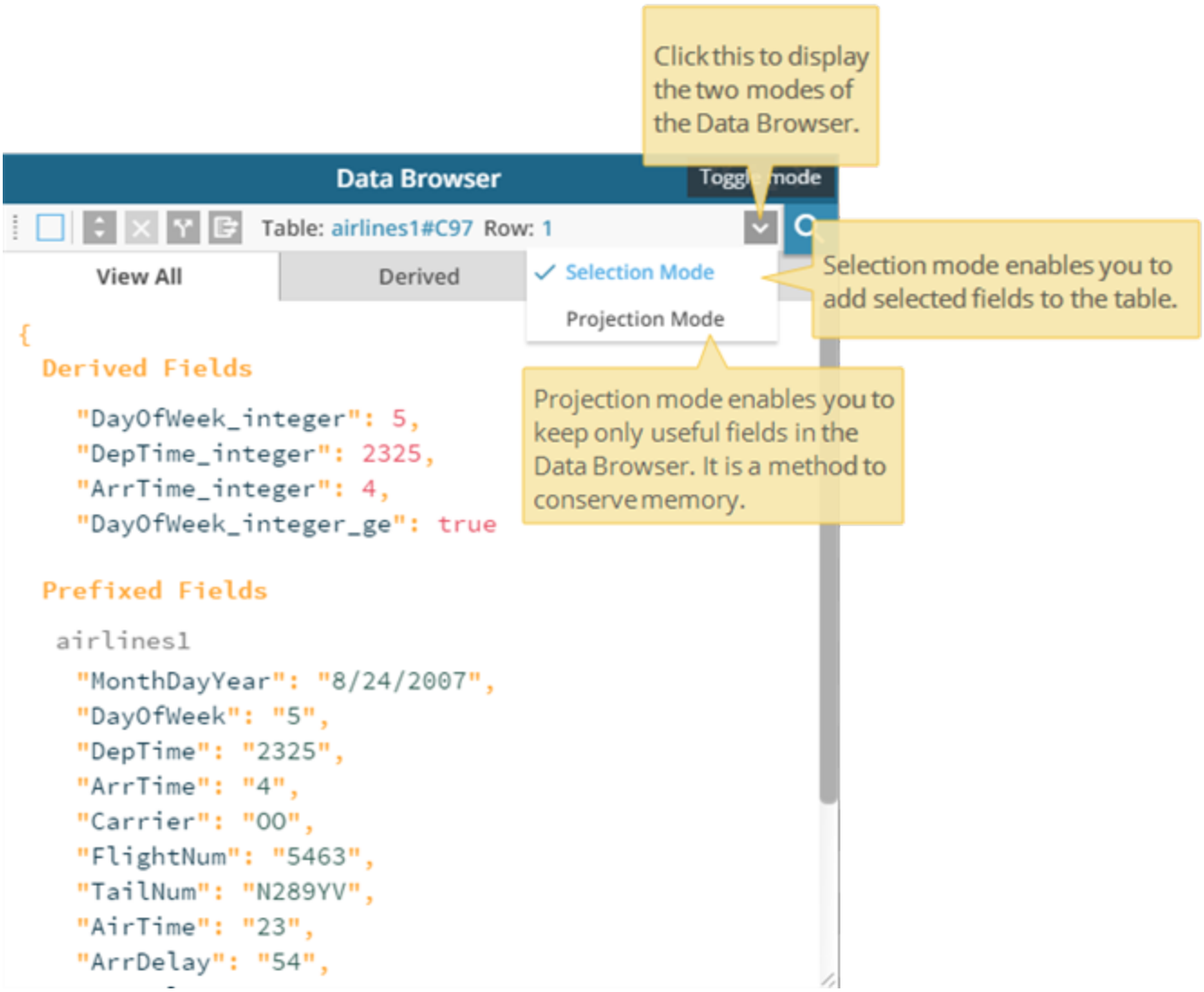
For each field, the value is also displayed. For example, in the following screen, the **Data Browser** for row 1 is displayed.



Tasks you can perform in selection mode

You can use the **Data Browser** in two modes: selection mode and projection mode. This section describes only the tasks performed in selection mode. For information about projection mode, see Removing fields from the Data Browser.

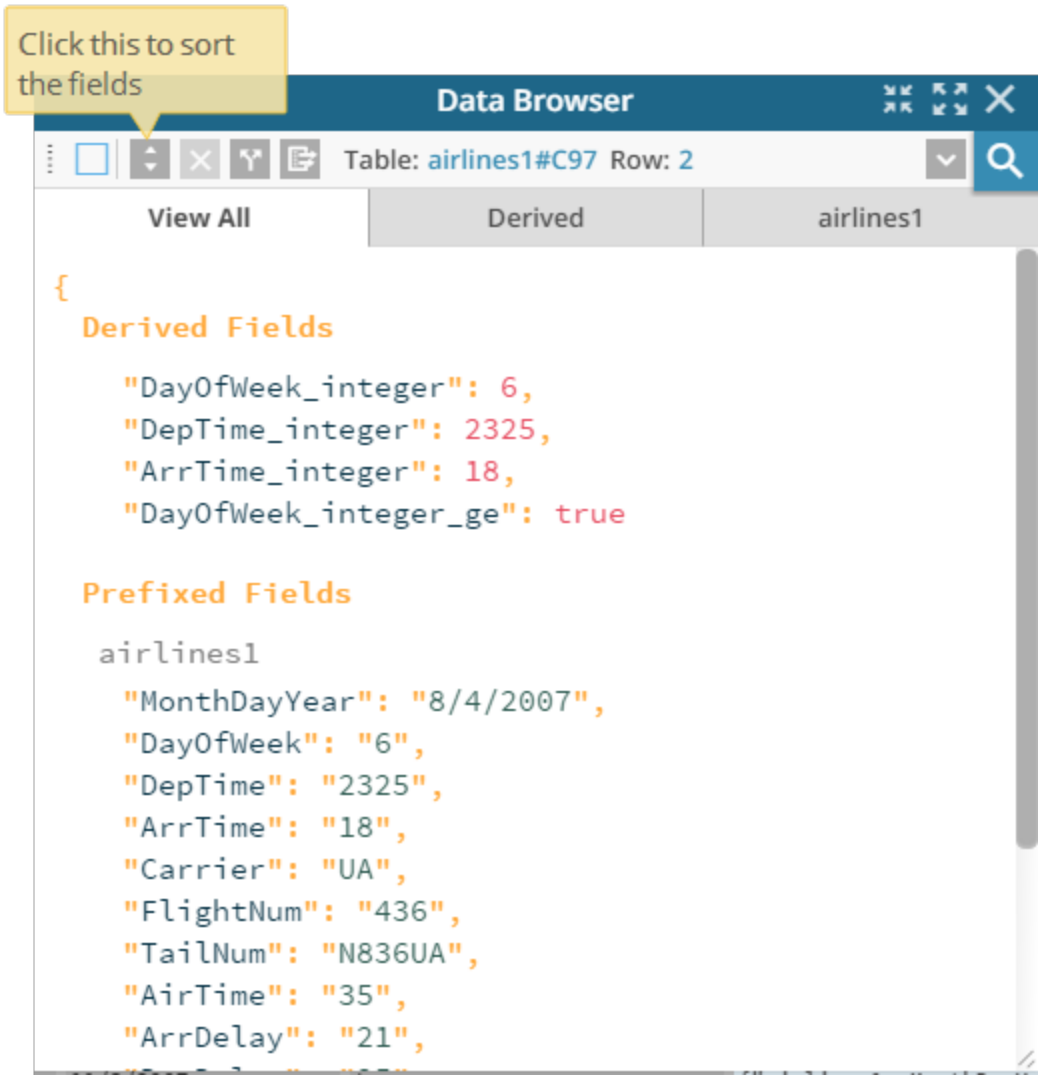
The following screenshot shows how to determine the **Data Browser**'s mode.



In selection mode, you can add columns to the active table as described in Adding columns from a dataset to a table. In addition, you can click the icons in the upper left corner of the browser to accomplish the tasks described in this section.

Sorting fields

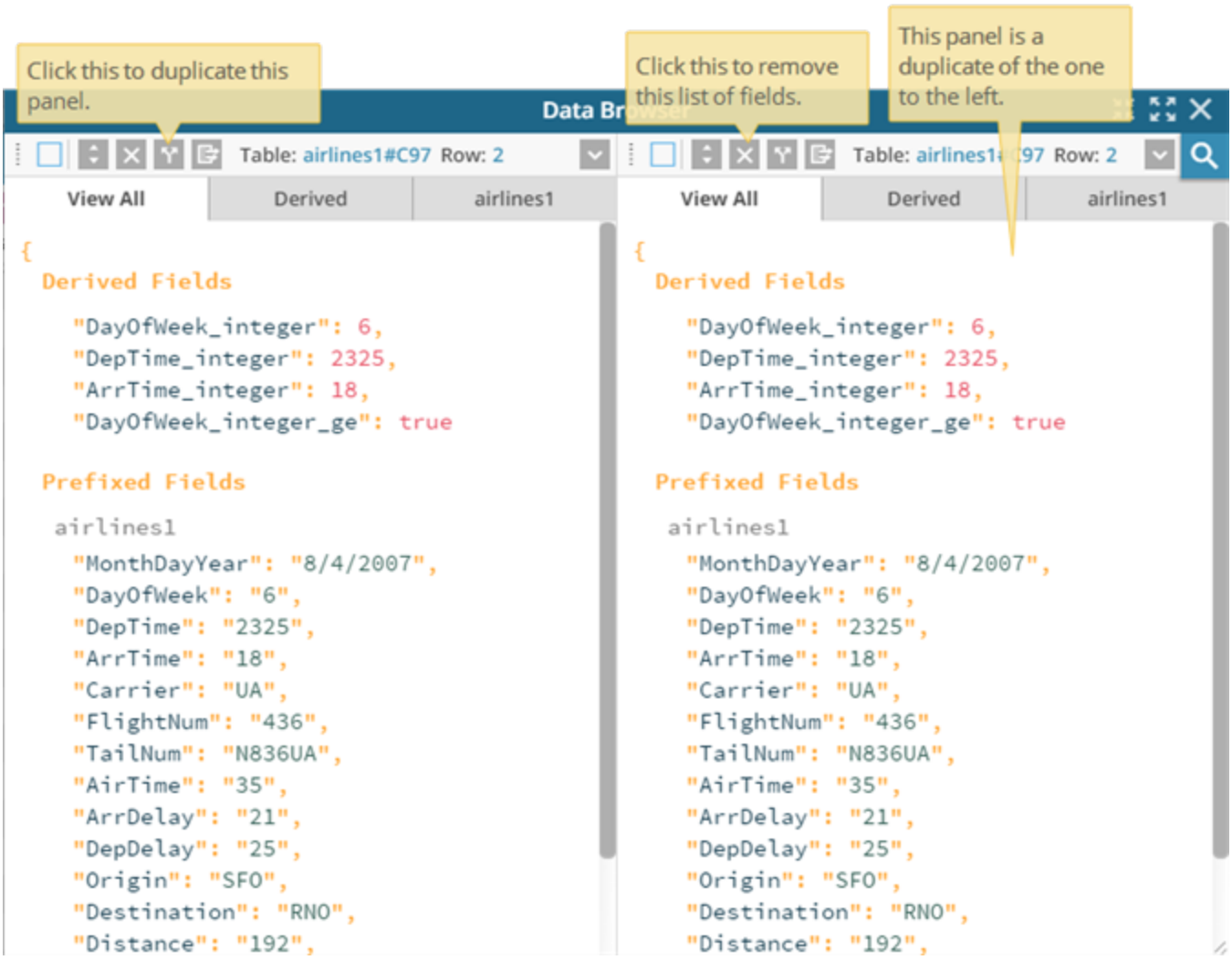
By default, the fields in the **Data Browser** are arranged in the same order as they are in the dataset. You can sort the fields in ascending or descending order. The following screenshot shows the location of the icon for sorting.



Duplicating the list of fields in the browser

Duplicating the list of fields is useful when you have a long list. Because each copy of the list has its own scroll bar, you can scroll each copy independently such that the fields that interest you appear side-by-side. The following screen shows the location of the icon for duplicating the field list.

In each panel, you can choose to view all fields, only the derived fields, or only fields with a particular prefix.



Comparing rows

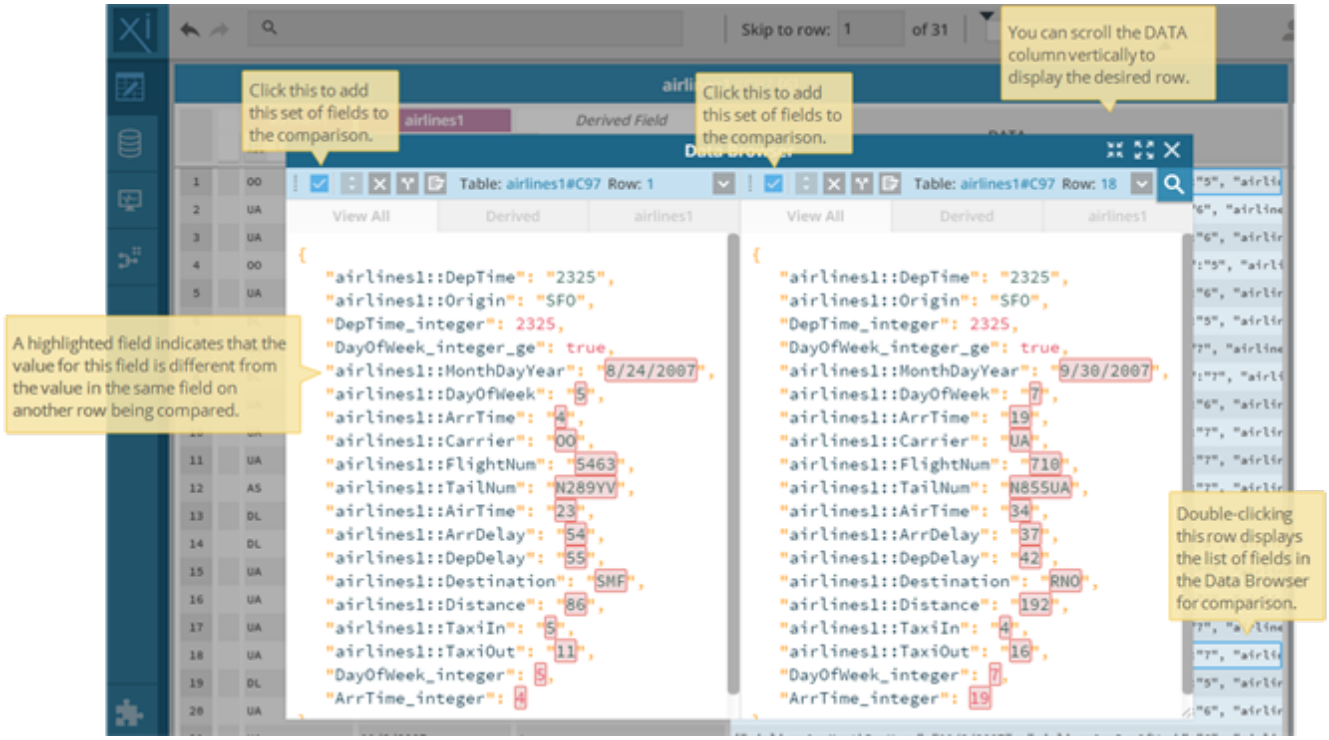
The **Data Browser** enables you to compare values in different rows from the same table or from different tables. Suppose the **Data Browser** for row 5 is displayed, you can double click another row in the **DATA** column (for example, row 10) to bring its data into the **Data Browser**. Now you can view the values from two rows side-by-side.

> **NOTE:** You might need to move the **Data Browser** to make the **DATA** column visible. Also, use the scroll bar to locate the table that interests you. When the **Data**

> **Browser** is displayed, you can only select rows visible in your window. You cannot scroll vertically to show rows that are currently out of the Xcalar Insight window.

To see how values differ, click the **Compare** icon for each list of fields that you want to compare. The values that are different between the rows are highlighted. The following screen shows the comparison between rows 1 and 18.

> **NOTE:** You can compare data from more than two rows.

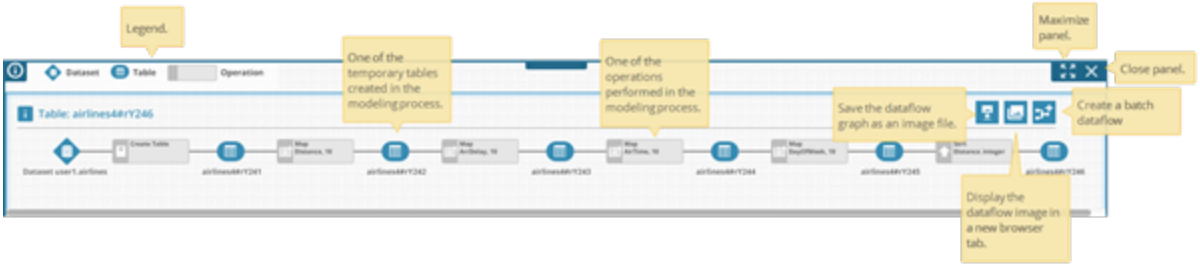


Dataflow graph

The dataflow graph provides an audit trail for your data. It helps you trace data lineage through all stages of your analytics pipeline. Over time, the data in your table might become increasingly complex due to the operations performed on the table. You can use the dataflow graph to determine how the data has been transformed since the table was created. For example, you can determine when a value in a column was filtered out or what operation was used to create a particular column.

Overview of the dataflow graph

The dataflow graph panel presents the dataflow graphs for all active tables in the same worksheet. The following screen shows an example of a dataflow graph.

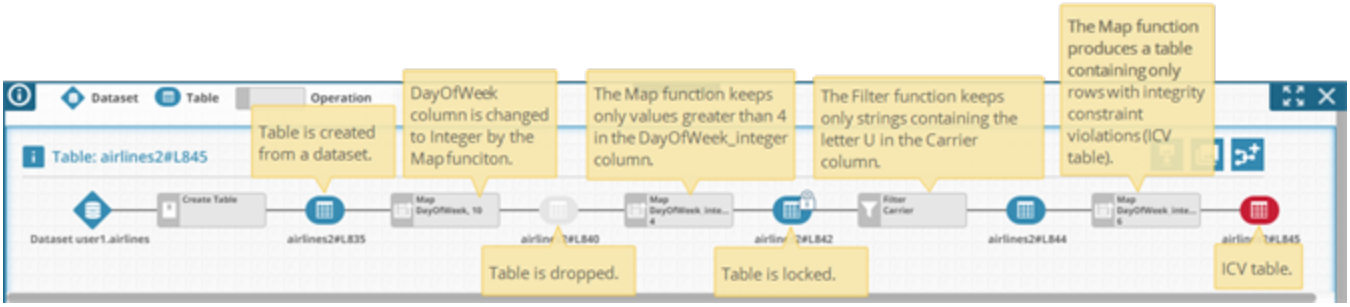


Example of a dataflow

In the following example, a series of operations take place on a table with a column named **DayOfWeek**, as explained in this list:

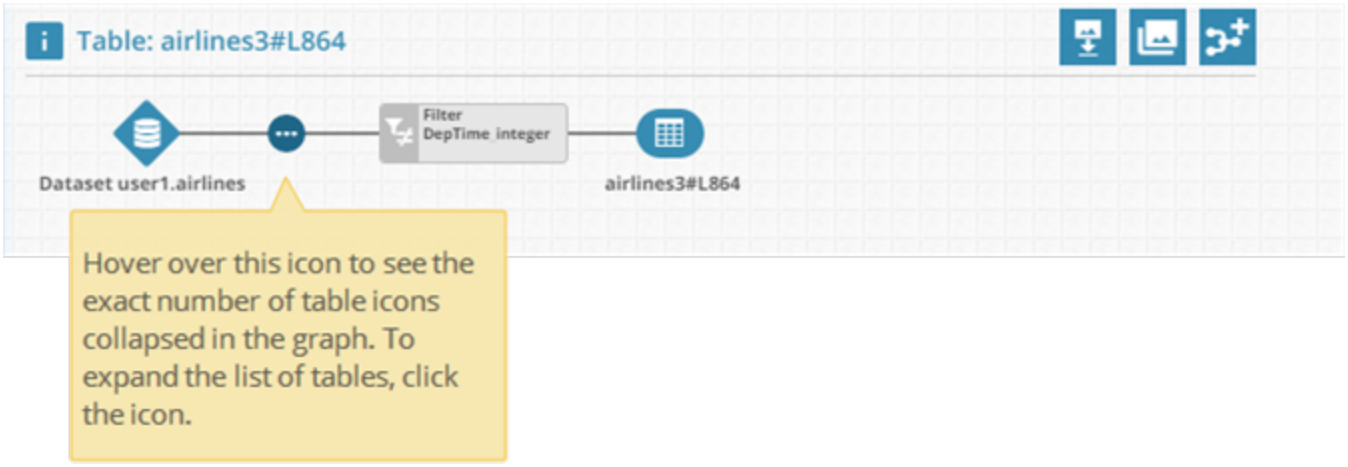
1. A table named **airlines2#L835** is created.
2. A Map function is executed to change the data type of the **DayOfWeek** column to Integer.
3. A Map function is executed to create a column that keeps values greater than 4 in the **DayOfWeek_integer** column.
4. The table is filtered to include only rows with the letter U in the Carrier column.
5. A Map function is executed on the DayOfWeek_integer column to produce a table with rows containing integrity constraint violations. For more information about integrity constraint violations, see Creating an integrity constraint violations table.

In this example, a table is dropped. This table is removed from Xcalar Insight permanently. You cannot add it back to your worksheet or perform operations on it.



Collapsing and expanding table icons in a dataflow graph

Xcalar Insight automatically collapses multiple table icons in a dataflow graph for better readability. The following screenshot shows the icon that indicates some table icons are collapsed in the dataflow graph. You can hover over the icon to see the exact number of table icons collapsed. Click the icon to expand the table icons.



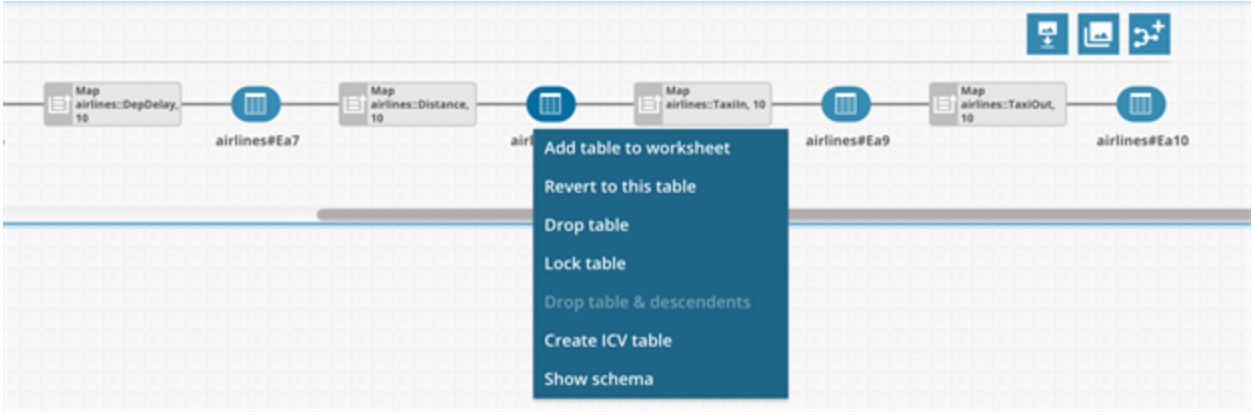
The following partial screenshot shows the icon that you can click to collapse a list of tables.

Tasks you can perform by clicking the table icon

You can click a table icon in the dataflow graph to display a pop-up that lists the actions you can take. The possible actions depend on whether the selected table is in the worksheet or is a temporary table or active table.

The last table in the dataflow is the active table. All other tables in the dataflow are temporary tables. For more information about different table statuses, see Understanding and changing table statuses.

The following partial screenshot shows the options available when you click a temporary table icon in a dataflow graph.



Temporary tables

You can perform these tasks only on temporary tables:

- Adding a table to the worksheet: You can click a temporary table icon in the dataflow graph and add the table to the worksheet. After you add the table to the worksheet, the table becomes an active table and has its own dataflow graph.
- Reverting to the selected table: If you want to restore the table to a particular state, click a table icon in the dataflow graph and revert to it. This causes the operation icons and the table icons to the right of the selected table icon to disappear.

> **NOTE:** Tables removed from the dataflow graph due to a revert are not dropped. They are categorized as **Temporary** in the **Tables** list. Temporary tables can be added back to the worksheet.

Active tables

You an perform these tasks on active tables only:

- Finding the table in the worksheet: You can quickly locate a particular table in a worksheet by clicking its icon in a dataflow graph. This feature is useful if you have many tables in a worksheet. Remember not to maximize the dataflow graph panel so that the table you try to locate is visible.
- Hiding the table: You can hide a table that you do not use often. The hidden table does not show up in the worksheet, but can be added back at any time. Also, the dataflow graph of a hidden table is removed from the dataflow graph panel.

Temporary and active tables

You can perform these tasks on temporary and active tables:

- Dropping a table: You can drop a table if the table is no longer needed. The icon for the dropped table remains in the dataflow graph but is displayed in gray. If the table is a part of other dataflow graphs, its icons in those dataflow graphs become gray as well. You cannot perform any operations on a dropped table.

  > **IMPORTANT:** You cannot undo the action of dropping a table.

  > **NOTE:** You cannot drop a table if an operation is taking place on that table.

- Locking a table: To prevent a table from being dropped accidentally, you can lock it. You cannot perform any tasks on a locked table until you unlock it.
- Show schema: A pop-up shows the number of rows in the table, in addition to the fields

and their data type in the table. If you click the field name, one or more table icons are highlighted. The highlighted icons represent all tables that also contain that field. This enables you to trace the lineage of a field. To remove the highlight, click a highlighted table icon.

> **EXAMPLE:** After a Join operation, it might not be obvious where a particular column comes from because multiple tables are involved in creating the resultant table. Use **Show Schema** to quickly determine the origin of the field.

> **EXAMPLE:** If a column in the active table was created by a function such as a Map function, you can use **Show Schema** to locate the first occurrence of the field and the operation that created the field.

- Create an ICV table: You can use the same function to create a version of this table, except that the resultant table is an ICV table. This option is available if the selected table is created by a Map or Group By function. For more information about ICV tables, see Creating an integrity constraint violations table.

Saving or displaying a dataflow graph as an image file

If you want to save a dataflow graph as an image (for example, to include the image in a presentation or to share the image with others), click  . The image is saved in the .png format.

Alternatively, you can click  to display the dataflow graph as an .png image in a new tab of your browser.

Table options

You can right click anywhere in the table title bar or click the triangle in the lower-right corner of the table title bar to display a drop-down menu. The menu presents table options for manipulating the entire table. This section explains the purpose of each table option.

Hide table

You can hide a table that you do not use often. The hidden table does not show up in the worksheet, but can be added back at any time. For more information about adding a hidden table to the worksheet, see Changing a table to an active table.

Minimize table

You can minimize a table so that the columns are not viewable. A small portion of the table is still visible in the worksheet, enabling you to click the table title bar to display the table options. One of the options in the menu from a hidden table is **Maximize table**, which returns the table to view again.

Drop table

You can drop a table you no longer need.

> **NOTE:** You cannot undo the action of dropping a table.

Export table

You can export a table or selected columns of a table to a location that is either the default location or a location defined as an export target.

Smart type casting

You can change the data type for one or multiple columns. For more information about smart type casting, see Changing the data type for a column.

Delete all duplicates

If you have created columns by duplication, you can delete all the duplicate columns.

Move

If you have multiple tables in the worksheet, you can move the table to the left or to the right.

If you have multiple worksheets in the workbook, you can move the table to a different worksheet.

Sort columns

You can arrange the table columns by sorting the column headers in ASCII order.

Resize all columns

You can resize all columns at once. For more information about resizing a column, see Resize.

Correlation

You can display the Correlation And Quick Aggregates modal window from this option. For more information about correlations and aggregates, see Determining correlation between two values.

Create batch dataflow

For each active table in the worksheet, you can create a batch dataflow. This option enables you to perform the same task as described in Creating a batch dataflow

Column options

This section describes the column options that enable you to manipulate table columns without performing a database operation. The options are available in the column drop-down menu that you display by right clicking the column header.

> **NOTE:** If you have an operation panel opened (for example, the **MAP** panel), you must exit that panel before you can display the column drop-down menu. To close the panel, right click the column header to display a pop-up and click in the command in the pop-up to exit the panel. Alternatively, you can click  in the panel to close it.

The column drop-down menu contains more options that the ones described here. The other options are described in the following topics:

- Determining aggregate values
- Filtering in a table column
- Grouping data in a table
- Using the Map function to create new values
- Joining tables
- Counting occurrences of unique values in a column
- Using extensions

Add a column

You can add a blank column and populate it with values from a function that you enter in the function bar. Follow these steps to add a blank column:

1. In the column drop-down menu, select **Add a column**.
2. Select **On the left** or **On the right**, depending on the desired location of the new column.

A blank , unnamed column is created.

> **NOTE:** The data type for the new column is unknown because there is no data. A data type is assigned when you execute a function on this column. After data is entered in the column, you can change the data type as described in Changing the data type for a column.

3. (Optional) Type the column name. You must press Enter to record the name.

   The following list describes the naming conventions for a table column:

   - The name can consist of any alphanumeric characters, space, and special characters except the following characters:
     - single quote mark (')
     - double quote mark (")
     - parenthesis (( ))
     - square bracket ([ ])
     - left or right brace ({ })
     - backslash (\)
     - two consecutive colons (::)
     - two consecutive dashes (--)
     - comma (,)
     - period (.)
     - asterisk (*)
   - First character must be a letter.
   - The name must not end with a dash or a space.
   - The name must not be **DATA**. Because column names are case sensitive, strings such as **data** and **Data** are acceptable.
   - The name must consist of 1 to 255 characters.

After you name the column, an equal sign (=) is displayed in the function bar, which enables you to immediately type a function that populates the new column with values.

> **NOTE:** When a column contains no values, the column drop-down menu has fewer options than the drop-down menu invoked from a column with values. For example, the Join and Filter functions are not available for a blank column. Also, if you start the Map function from a blank column, the column name is automatically entered in the **New Resultant Column Name** field in the **Map** panel.

Delete a column

In the column drop-down menu, select **Delete a column**. The column from which you invoke the drop-down menu is deleted.

> **NOTE:** Deleting a column does not remove the field from the dataset. You can always add a deleted column back to a table. See Adding columns from a dataset to a tablefor information about how to add a column.

Duplicate a column

In the column drop-down menu, select **Duplicate a column**. A new column is duplicated to the right of the one from which you invoke the drop-down menu.

The new column is named after the original column with an underscore and a number appended to it. For example, if the original column is named **ID**, the new column is named **ID_1**.

> **NOTE:** If you perform a sort or filter function on a column, the same function automatically takes effect on its duplicates. For example, when you sort a column named **ID**, its duplicate, **ID_1**, is sorted; when you filter out a particular value from **ID**, the same filtering takes place in **ID_1**. However, if you change the data type for a

> column, the change takes place only in the selected column. It does not take place in the duplicate column.
>
> Suppose you sort numerically a column of the String type, the column's data type is changed to Integer after the sort. While the column's duplicate is sorted as well, the data type of the duplicate column remains as String.

Delete other duplicates

In the column drop-down menu, select **Delete other duplicates**. All duplicates of the selected column are deleted.

Hide and unhide a column

In the column drop-down menu, select **Hide column**. The column is minimized. To view the column again, display the column drop-down menu and select **Unhide column**.

Text align

Follow these steps to align text in a column:

1. In the column drop-down menu, select **Text align**.
2. Select **Left Align**, **Right Align**, **Center Align**, or **Wrap Text**. The **Wrap Text** option automatically aligns text on the left.

Resize

You can resize a column by resting the cursor on the boundary of the column header until the cursor changes to a resize cursor. Then drag left or right to resize the column.

Alternatively, follow these steps to resize a column to a particular width:

1. In the column drop-down menu, select **Resize**.
2. Select one of the following:

- **Size to header**: The column width is changed to accommodate the header.
- **Size to contents**: The column width is changed to accommodate the widest value in the column
- **Size to fit all**: The column width is changed to accommodate both the header and the widest value in the column.

Change data type

You can change the data type as described in Changing the data type for a column. This option is not available if the column is an array.

Format

You can format values in a numeric column as percentages.

Round

You can round the values in a column by specifying the number of decimal places. This option is available only if the data type of the column is Float.

Split column

You can split a column if it contains strings. You cannot split a numeric column. New columns are created as a result of the split, and the original column remains in the table.

Follow these steps to split a column:

1. In the column drop-down menu, select **Split column**. A pop-up is displayed for you to specify how to split the column.
2. Type the delimiter in the **Split Column By** field. The delimiter is the character that separates the data to be split. Optionally, enter the number of splits.

> **EXAMPLE:** If the **Date** column contains dates in the mm/dd/yy format, you can enter the slash (/) as the delimiter and 1 as the number of splits. Two columns are created, one with the mm field and the other with the dd/yy field.

3. Click the check mark in the pop-up to submit your information.

Sort

You can sort values in a column so that values are displayed in ascending or descending order. Rows containing FNF in the sorted column are not included in the resultant table. Therefore, it is possible the resultant table contains fewer rows than before the sort. For more information about FNF, see Working with a table containing FNF.

> **IMPORTANT:** Sorting only affects how the rows are displayed. Subsequent operations do not preserve the result of the sort. For example, you can sort on a column named **Age** to display rows ordered by age. Later if you perform an arithmetic function to sum two other columns, the resultant table displayed after the function is in the original, unsorted order. The rows are no longer sorted by age.

Follow these steps to sort a column:

1. In the column drop-down menu, select **Sort**. A pop-up is displayed for you to specify whether to sort in ascending or descending order.
2. If the data type of the column is String and Xcalar Insight decides that most of the values in the column are numbers, it displays the **Sort Suggestion** dialog box. Depending on the desired method of sorting, click **Alphabetically** or **Numerically**.

> **NOTE:** If you select **Numerically**, the data type of the column is changed to integer.

Cell options

This section describes the cell options that enable you to manipulate a table based on a value in a particular cell. The exact cell options available depend on the data type of the table column.

To display the cell options, click a table cell.

Filter this value

This option creates a table based on the value in the selected cell. The resultant table contains only rows with the selected value in the column. This option is not available for an array column.

Exclude this value

This option creates a table based on the value in the selected cell. The resultant table contains all rows except the ones with the selected value in the column. This option is not available for an array column.

Copy to clipboard

You can copy the value in the selected cell to the clipboard so that you can paste the text in another location either in Xcalar Insight or another application.

Examine

This option is available only for an array column. It displays a portion of the **Data Browser** pertaining to the array in the selected cell. You can view the value for each element in the array, or add an element as a column. For more information about using the **Data Browser**, see Adding columns from a dataset to a table.

Pull all

This option is available only for an array column. It pulls all elements of the array to create a column for each element.

Map functions

This section describes the function categories supported under the Map function. If the input type is shown as Any, the data type can be boolean, float, integer, or string.

Arithmetic functions

Function	Description	Input type	Output type
abs	Returns the absolute value.	Float	Float
absInt	Returns the absolute value.	Integer	Integer
add	Adds operand 1 and operand 2.	Float or integer	Float
ceil	Returns the smallest integer that is greater than or equal to the input.	Float or integer	Integer
div	Divides operand 1 by operand 2. If operand 2 is 0, the result is FNF.	Float or integer	Float
exp	Raises e (mathematical constant) to the power that is equal to the input.	Float or integer	Float
floor	Returns the largest integer less than or equal to the input.	Float or integer	Integer
log	Returns the logarithm of the input to the base of e.	Float or integer	Float
log10	Returns the logarithm of the input to the base of 10.	Float or integer	Float

Function	Description	Input type	Output type
log2	Returns the logarithm of the input to the base of 2.	Float or integer	Float
mod	Returns the remainder when operand 1 is divided by operand 2. If operand 2 is 0, the result is FNF.	Float or integer	Float
mult	Multiplies operand 1 by operand 2.	Float or integer	Float
pow	Raises operand 1 to the power equal to operand 2.	Float or integer	Float
round	Rounds the input to the nearest integer.	Float or integer	Integer
sqrt	Returns the square root of the input.	Float or integer	Float
sub	Subtracts operand 2 from operand 1.	Float or integer	Float

Bitwise functions

Function	Description	Input type	Output type
bitand	Returns the result of the AND operation on operand 1 and operand 2.	Integer or boolean	Integer

Function	Description	Input type	Output type
bitlshift	Returns the result of a logical left shift on the bits of operand 1 by the number of bits specified by operand 2.	Integer or boolean	Integer
bitor	Returns the result of the OR operation on operand 1 and operand 2.	Integer or boolean	Integer
bitrshift	Returns the result of a logical right shift on the bits of operand 1 by the number of bits specified by operand 2.	Integer or boolean	Integer
bitxor	Returns the result of the XOR operation on operand 1 and operand 2.	Integer or boolean	Integer

Conditional functions

Function	Description	Input type	Output type
And	Returns the result of the AND operation on operand 1 and operand 2.	Boolean	Boolean
Between	Tests if operand 1 is equal to or greater than operand 2, and less than or equal to operand 3.	Float or integer	Boolean
Contains	Tests if operand 1 contains operand 2.	String	Boolean

Function	Description	Input type	Output type
Eq	Tests if operand 1 and operand 2 are equal.	Any (but operands must be of the same type)	Boolean
Exists	Tests if a value exists in the input column.	Any	Boolean
Ge	Tests if operand 1 is greater than or equal to operand 2.	Integer, float, or boolean	Boolean
Gt	Tests if operand 1 is greater than operand 2.	Integer, float, or boolean	Boolean
isBoolean	Tests if the input data type is boolean.	Any	Boolean
isFloat	Tests if the input data type is float.	Any	Boolean
isInf	Tests if the input is infinity.	Float	Boolean
isInt	Tests if the input data type is integer.	Any	Boolean
isStr	Tests if the input data type is string.	Any	Boolean
le	Tests if operand 1 is less than or equal to operand 2.	Integer, float, or boolean	Boolean
like	Tests if operand 1 is similar to operand 2.	String	Boolean

Function	Description	Input type	Output type
lt	Tests if operand 1 is less than operand 2.	Integer, float, or boolean	Boolean
neq	Tests if operand 1 is not equal to operand 2.	Any (but operands must be of the same type)	Boolean
not	Returns the NOT operator on the input.	Boolean	Boolean
or	Returns the result of the OR operation on operand 1 and operand 2.	Boolean	Boolean
regex	Tests if operand 1 matches operand 2, which can be a regular expression.	String	Boolean

Conversion functions

Function	Description	Input type	Output type
ipAddrToInt	Converts an IP address (with the specified number of octets) to an integer.	String	Integer
macAddrToInt	Converts a MAC address (with the specified number of octets) to an integer.	String	Integer

Miscellaneous functions

Function	Description	Input type	Output type
dhtHash	Generates a hash key used by the distributed hash table.	Any	Integer
genRandom	Generates a random number between the first operand and second operand.	Integer or boolean	Integer
genUnique	Generates a unique integer per row in a new column.	N/A	integer
if	Tests if a condition is true or false. If true, returns the first value; if false, returns the second value.	Integer, float, or boolean	Integer or float
ifInt	Tests if the input is an integer. If true, returns the first value; if false, returns the second value.	Any	Integer or float

Function	Description	Input type	Output type
ifStr	Tests if the input is a string. If true, returns the first value; if false, returns the second value.	Any	Integer or float

String functions

Function	Description	Input type	Output type
concat	Concatenates two strings.	String	String
countChar	Counts occurrences of specified character or string.	String	Integer
cut	Returns the string located at the numbered field. Field numbers start at 0. Fields are separated by a delimiter. For example, if the string in the Date column is 12/6/2016 and the delimiter is the / character and the field number is 3, the function returns 2016 because it is the third field in the string. The function does not return a value if the numbered field does not exist, the delimiter does not exist, or the delimiter is a multi-character string.	Input to parse: String Field number: Integer or boolean Delimiter: String	String

Function	Description	Input type	Output type
find	Returns the index of the search string (that is, the position of the string's first character). Starting index and ending index limit where the string can be found. Indexes start at 0 and are inclusive. To return an index as long as it is equal to or greater than the starting index, specify 0 for the ending index. If the string is not found within the specified indexes, the function returns -1. For example: For the Date column, you can search for 2007 with the starting index of 3 and ending index of 5. Suppose the source string is 1/1/2007, the function returns 4 because 2007 is found at index 4. Suppose the source string is 12/12/2007, the function returns -1 because 2007 is not found at an index between 3 and 5. However, if the ending index is set to 0, the function returns 6.	Column to search: String String to find: String Starting index: Integer or boolean Ending index: Integer or boolean	Integer
len	Returns the length of the input.	String	Integer
replace	Searches for a string and replaces it with another string in a specified column.	String	String

Function	Description	Input type	Output type
rfind	Same as the find function except that it returns the index of the last character of the search string.	Column to search: String String to find: String Starting index: Integer or boolean Ending index: Integer or boolean	Integer
strip	Eliminates the leading and trailing white space from the input string.	String	String
substring	Creates a string from the source that starts at the start index (inclusive) and ends at the end index (exclusive). This is the equivalent of the python command `str[startIdx:endIdx]`. To have the same effect as `str[startIdx:]`, specify 0 for the end index. To have the same effect as `str[:endIdx]`, specify 0 for the start index. To index back from the end of the string, use a negative index value.	Column to search: String Start index: Integer or boolean End index: Integer or boolean	String
wordCount	Returns the number of words in the input.	String	Integer

Trigonometric functions

Function	Description	Input type	Output type
acos	Returns the arccosine of input value, in radians.	Integer, float, or boolean	Float
acosh	Returns the hyperbolic arccosine of input value, in radians.	Integer, float, or boolean	Float
asin	Returns the arcsine of input value, in radians.	Integer, float, or boolean	Float
asinh	Returns the hyperbolic arcsine of input value, in radians.	Integer, float, or boolean	Float
atan	Returns the arctangent of input value, in radians.	Integer, float, or boolean	Float
atan2	Returns the principal value of arctangent(y/x), in radians.	Integer, float, or boolean	Float
atanh	Returns the hyperbolic arctangent of input value, in radians.	Integer, float, or boolean	Float
cos	Returns the cosine of input value expressed in radians.	Integer, float, or boolean	Float
cosh	Returns the hyperbolic cosine of input value expressed in radians.	Integer, float, or boolean	Float
degrees	Converts input value, expressed in radians, to degrees.	Integer, float, or boolean	Float

Function	Description	Input type	Output type
pi	Creates a column with the value of pi in each row.	N/A	Float
radians	Converts input value, expressed in degrees, to radians.	Integer, float, or boolean	Float
sin	Returns the sine of input value expressed in radians.	Integer, float, or boolean	Float
sinh	Returns the hyperbolic sine of input value expressed in radians.	Integer, float, or boolean	Float
tan	Returns the tangent of input value expressed in radians.	Integer, float, or boolean	Float
tanh	Returns the hyperbolic tangent of input value expressed in radians.	Integer, float, or boolean	Float

Type-casting functions

Function	Description	Input type	Output type
bool	Casts value to a boolean value as follows: • A string is converted to the boolean value, false. The only exception is that if the string is **true**, it is converted to the boolean value, true. • The number 0 is converted to the boolean value, false. • A non-zero float or integer value is converted to the boolean value, true.	Any	Boolean

Function	Description	Input type	Output type
float	Casts value to float.	Any	Float
int	Casts value to integer.	Any	Integer
string	Casts value to string.	Any	String

User-defined functions

User-defined functions are the ones that you define.