IBM Cognos Analytics
Version 11.1

Data Modeling Guide

IBM

©

Product Information

This document applies to IBM Cognos Analytics version 11.1.0 and may also apply to subsequent releases.

Contents

Chapter 1. Data modeling in Cognos Analytics

IBM® Cognos® Analytics provides web-based, self-service data modeling capabilities.

You can use data modeling in Cognos Analytics to fuse together many sources of data, including relational databases, Hadoop-based technologies, Microsoft Excel spreadsheets, text files, and so on. Using these sources, a data module is created that can then be used in reports, dashboards, and explorations.

Star schemas are the ideal database structure for data modules, but transactional schemas are equally supported.

You can enhance a data module by creating calculations, defining filters and navigation paths, and more.

After you save a data module, other users can access it. Save the data module in a folder that users, groups, and roles have appropriate permissions to access. This process is the same as saving a report or dashboard.

Tip: Data modeling in Cognos Analytics does not replace IBM Cognos Framework Manager, IBM Cognos Cube Designer, or IBM Cognos Transformer, which remain available for more complex modeling.

Intent-driven modeling

You can use intent-driven modeling to add tables to your data module. Intent-driven modeling proposes tables to include in the module, based on matches between the terms that you supply and metadata in the underlying sources.

While you are typing in keywords for intent-driven modeling, text from column and table names in the underlying data sources is retrieved. The intent field has a type-ahead list that suggests terms that are found in the source metadata.

Intent-driven modeling recognizes the difference between fact tables and dimension tables by the number of rows, data types, and distribution of values within columns. When possible, the intent-driven modeling proposal is a star or snowflake of tables. If an appropriate star or snowflake cannot be determined, then intent-driven modeling proposes a single table or a collection of tables.

Modeling user interface

Use the web modeling user interface to create, enhance, and edit data modules.

Access to this interface is controlled by the **Web-based modeling** capability that is managed by administrators. For more information, see the *Managing IBM Cognos Analytics* guide.

You can enter the web modeling user interface from the IBM Cognos Analytics portal in one of the following ways:

- In **Team content**, **My content**, or **Recent**, locate an existing data module, and click it to open.
- Click **New** +, and select **Data module**. Then, create a new data module.
- Use the **Quick launch** facility in the Cognos Analytics welcome page to upload a file. Drop the file in the **Data module** box, and start creating your data module.

When working with data modules, you can use the undo ↶ and redo ↷ actions in the application bar to revert or restore changes to the data module in the current editing session. You can undo or redo up to 20 times.

Sources panel

The **Sources** panel shows the sources of data that the data module contains. The sources can be data servers, uploaded files, data sets, packages, and other data modules.

Except for packages, you can expand the specific source to view its tables and columns. Drag tables onto the data module panel or onto the diagram to add them to the data module.

From the source context menu , you can initiate actions such as relinking sources or enabling data caching.

Data module panel

The data module tree shows the tables and columns of data that are included in the data module. This is the main space for editing the data module.

Click the context menu icon for the module, table, or column to view its modeling and editing context menu options. Here you can start joining tables, creating filters and calculations, or renaming and deleting items.

Click the **Add sources and tables** icon in the panel toolbar to add sources and tables to your data module. Clicking the **Identify navigation path members** icon underlines columns that are members of navigation paths. If none of the columns are underlined, the data module does not contain navigation paths.

Relationships diagram

The diagram is a graphical representation of table relationships in a data module. You can use the diagram view to examine the relationships, edit the data module, and view the cardinality information for the relationships.

Right-click a table in the diagram to view the table context menu that can be your starting point for creating joins or filters, renaming the table, viewing the table properties, or removing it from the module.

Click any table join to see the join summary information that includes the matching keys. When you right-click the join line, the context menu appears with options for editing or deleting the join.

Right-click one or more tables in the diagram, and click **Auto-arrange**. The diagram is redrawn around the first selected table allowing you to focus on the selected tables and their relationships.

In the **Diagram settings** box, select the **Cardinality** check box to show the cardinality of relationships between different tables in your data module. Move the **Degrees of separation** slider. Depending on the slider position, the diagram shows different degrees of relationships between tables. Select one or more tables in the diagram, and use the **Focus mode** to work with the selected tables.

Grid view

You can use the grid view to examine the actual data in table columns and rows.

Select a table or column in the data module tree or in the diagram, and click the grid icon to open the data view.

Validation panel

To validate the data module, click the validation icon in the application bar or in the **Data module** panel, or click **Validate** from the data module context menu.

If errors are discovered, the number of errors is displayed on the validation icon in the application bar , and the failed validation icon is displayed for tables, columns, expressions, or joins. Click the error icons to view the validation messages. Click the copy icon in the error messages to copy the messages to the clipboard for easier analysis or printing.

Expression editor

The expression editor is an SQL editing tool that you can use to create or edit SQL-based tables, calculations, filters, or data groups.

You can create expressions by typing the code or dragging items from the data module tree. The validation and data preview capabilities help to quickly verify and troubleshoot the expressions. The code editing capabilities include: inserting comments, function auto-complete, pretty-print, high-contrast mode, and different font sizes. The information panel shows details and provides examples of supported functions that are used in the expressions.

Chapter 2. Data modules and their sources

A data module is the result of modeling activities in IBM Cognos Analytics.

Data module sources

Data modules can be based on other data modules, data servers, uploaded files, data sets, and packages. You can combine multiple, different types of sources into one data module.

When you create a new data module in IBM Cognos Analytics, or update an existing module, you can choose five possible input source types from the **Select sources** dialog box.

Data modules

Data modules that are saved in **Team content** or **My content** can be used as sources for other data modules. The tables remain linked to the source data module, which is indicated by the linked table icon , and they are read-only. As long as the tables remain linked, any changes in the source module are reflected in the new data module. If you break the link, you can edit the tables. However, the source module changes are no longer reflected in the new module.

Data servers

Data servers represent databases for which connections exist in Cognos Analytics. The connections must already be created in **Manage** > **Data server connections** or **Manage** > **Administration console**. For more information, see *Managing IBM Cognos Analytics.*

Uploaded files

Uploaded files are Microsoft Excel (.xlsx and .xls) spreadsheets and text (.csv) files that contain comma-separated, tab-separated, semi colon-separated, or pipe-separated values. Files that are already uploaded to Cognos Analytics are stored in **Team content** or **My content**. You can also upload files after you start creating your data module by using the upload file facility in the **Select sources** dialog box.

For more information about uploaded files, see the *IBM Cognos Analytics Getting started* guide.

Data sets

Data sets contain data that is extracted from a package or a data module. Data sets are stored in **Team content** or **My content**. If the data in the data set changes, the change is reflected in the data module.

For more information about data sets, see the *IBM Cognos Analytics Getting started* guide.

Packages

You can use relational, dynamic query mode packages as sources for data modules. Packages are created in IBM Cognos Framework Manager and contain dimensions, query subjects, query items, and other data. Packages are located in **Team content** or **My content**.

When you use a package as your source, you can't select individual query subjects. You must drag the entire package into your data module. Only when you create relationships that involve query subjects in the package, you can view the structure of the package.

Tip: Query subjects and query items in packages are equivalents of tables and columns in data modules.

For more information about packages, see the *IBM Cognos Analytics Getting started* guide.

Creating a data module

A user can quickly create a data module that includes data from one or more sources of different types. The data module can be shared with other users, and used as a source to create reports, dashboards, and explorations.

Before you begin

Prepare the sources that you plan to use to create a data module.

- Save the sources to **Team content** or **My content**.

  The only exception are your data files that can be uploaded while the data module is created.
- Create data server connections in **Manage** > **Data server connections**.

  If your data server is Planning Analytics, you create the TM1 cube-based data modules in the administration interface, as soon as the data server connection is created. For more information, see *Creating data modules from Planning Analytics cubes* in the *Managing IBM Cognos Analytics* guide.
- Ensure that the **Allow web-based modeling** check box is selected on legacy JDBC data source connections.

  The data source connections are created in the **Administration console**. If the **Allow web-based modeling** check box is not selected for this type of connections, the connections are not available in **Manage** > **Data server connections**, and cannot be used as data module sources.

  Go to **Manage** > **Administration console** . On the **Configuration** tab, select **Data source connections**, and find the connection. From the connection properties, click the **Connection** tab where the **Allow web-based modeling** check box is located.

For more information, see "Data module sources" on page 5.

About this task

To access the data modeling user interface, users need execute and traverse permissions for the **Web-based modeling** capability. For more information about capabilities, see the *Managing IBM Cognos Analytics* guide.

Procedure

1. In the Cognos Analytics welcome page, click **New** > **Data module**.

   Tip: An alternative way to start creating a data module is to upload data files first by using the **Quick launch** facility. When you drop the files onto the Cognos Analytics welcome page, in the **Data module** box, you can immediately start creating your data module. Other sources can be added to the data module later.
2. In the **Select sources** dialog box, select one or more sources of any type.

   - To select a saved data module, data set, uploaded file, or package, click the **Team content** , **My content** , or **Recently viewed content** folder, and locate the source that you want to add. If needed, use search and filtering options to find the sources.

   - To select a data server, click the **Data servers and schemas** folder. Select the data server connection that you want. The available schemas in the data server are listed. Choose the schema that you want to use. Only schemas for which metadata is preloaded are displayed.

   - To upload a data file from your hard drive or LAN location, click the **Upload** icon , and browse for the file. By default, the file is saved to **My content**.
3. If all selected sources contain one table each, the basic data module is created, and you can proceed to step 5.

4. If any of the selected sources, such as a multi-tab spreadsheet or a data server, contain multiple tables, you have two options to add the tables to your data module:
   - **Select tables**

     You select the tables manually, and click **OK** to create the data module.
   - **Discover related tables**

     You provide some keywords that describe your modeling objectives. The keywords are auto-completed if a matching word is found in the source. When you select the keyword and click **Go**, the tables that include the keyword are suggested to you. You can accept the suggested tables, or try another keyword. After you accept the suggested selection of tables, the data module is created for you.

   The data module is created based on the chosen tables.
5. Examine the data module.
   - In the **Data module** panel, view the sources that are included in your data module.

     Except for packages, you can expand the sources to view their tables and columns.

     For data modules, the link icon on tables indicates that the tables are linked to the source data module.

     For data server connections and uploaded files, the table and column labels are cleaned up in English and some other languages in the following way:
     - For single files, the file extension, such as .xls or .csv, is removed from the table label. For example, `Customer_analysis.csv` is changed to `Customer Analysis`.
     - Characters such as underscore (_), dash (-), or slash (\) are replaced with the space character. For example, `Vehicle_class` is changed to `Vehicle Class`.
     - In column labels, all words are capitalized. For example, `Vehicle class` is changed to `Vehicle Class`.
     - Camel case strings are split into individual words. For example, `OrderDate` or `orderDate` is changed to `Order Date`.
     - Extra spaces are removed.
   - To view data, select a table or a column in a table, and click the data grid view.
   - To view relationships between tables, click the diagram view. Typically, the relationships are detected by the system and joins between tables are created automatically. If the tables are not joined, you need to join them manually.
   - To check if there are no broken expressions, click the validation view.
6. To create a test report from your data module, click **Try It** in the application toolbar.
   A new tab opens in your browser with IBM Cognos Analytics - Reporting open within it. Your data module is shown in **Sources**.
7. To save the data module, click **Save** or **Save as**.

What to do next

You can enhance the data module by adding calculations, filters, groups, and more. For more information, see Chapter 3, "Modeling metadata ," on page 13.

Adding new sources or tables

After a data module is created, you can add new sources or different tables from the sources that are already in the data module.

About this task

You can add a combination of source types in a data module.

Procedure

1. Open an existing data module.
2. In the **Data module** panel, click the **Add sources and tables** ⊕ icon.
3. Select one of the following options:
   - **Add new sources**

     Select and add new sources to the data module.
   - **Add more tables**

     Add tables from sources that are already in the data module. Only tables that aren't already included in the data module can be selected.
   - **Discover related tables**

     Add tables from sources that are already in the data module. This option is available only for sources that contain multiple tables, such as data servers and multi-sheet uploaded files. Based on keywords that you type, related tables are suggested to be added to the data module.
4. Save the data module.

Relinking sources

You can relink a data module source to a different source. After a successful relink, global calculations and relationships in the data module remain valid.

Here are some scenarios in which relinking a source can be useful:

- You build and test a data module against a test source. When the data module is ready, you relink the source to the intended production source.
- The current source in your data module is invalid, and you must use a new, valid source.
- You want to relink your data module from one data server to another data server, or from one schema to another schema.

  Relink between different types of data servers is supported, as well as between schemas and catalogs within data servers.

  Tip: Data server sources can be organized into schemas, catalogs, both, or none.

About this task

The relinked (target) source must be of the same type as the original source. A data server can be relinked only to a data server, an uploaded file to an uploaded file, and so on.

In addition to the matching source types, the following conditions must be met:

- All columns from the original source must exist in the target source, and the columns **Identifier** properties (case-sensitive) and data types must match.

  For example, file A with columns `ColA` and `ColB` can be relinked to file B with columns `ColA` and `ColB`. Relinking to file B with columns `colA` and `colB` would not work.

The data types of the matching columns must be compatible for the data module calculations and relationships to remain valid. For example, if the column data type in the original source is `date`, the column data type in the target source must also be `date`, and not `string` or `timestamp`.

- For data servers, packages, and data modules, all tables from the original source must exist in the target source, and the tables **Identifier** properties (case-insensitive) must match. If a matching table can't be found based on these criteria, the system also considers the table labels and matching columns identifiers (case-sensitive) when trying to find the right match.

  If a duplicate match is found in the target source, the last table in the list is used for the match.
- Extra columns and tables can exist in the target source.

  When relinking to a source that contains a table with extra columns, you can add the extra columns to the table in the data module by dragging the table from the **Sources** pane to the **Data module** pane.
- The source names, such as file and package names or data server connection names, do not need to match.

Tip: Columns and tables matching is done by comparing their **Identifier** property. The column or table **Identifier** value can be, but not always is, the same as the column or table name (**Label**). You can view the **Identifier** value in the column or table **Properties** pane, **Advanced** section.

Procedure

1. From **Team content** or **My content**, open your data module.
2. In the **Sources** pane, find the source that you want to relink.
3. From the source context menu, select **Relink**.
4. Select the source type that matches the original source type. If the original source is data server, select data server, if it's an uploaded file, select a file, and so on.
5. Click **Done**.

   If the relink was successful, a confirmation message is displayed.

   If the relink finished with errors, a message is displayed that suggests opening the validation view where the relink issues are listed. Resolve the issues, and save the data module. You can also save the data module with unresolved issues.

   Important: The validation process does not detect incompatible data types on columns. If there are columns with incompatible data types in your sources, and all other relink conditions are met, the successful relink message is displayed. This type of data issues must be reconciled in the sources.

Results

After you successfully relink a source in a data module, reports and dashboards that are based on this data module can start using the new source without any involvement from report authors.

Enabling data caching

You can enable data caching in a data module, and specify the cache expiry options.

Data caching can be enabled at a source level, or at a table level. Tables don't automatically inherit the cache options from their sources.

The following cache options can be specified for sources and tables:

- **No cache** - Data caching is disabled.
- **Automatic** - The data cache option that was specified for a source. This option is available for tables only.
- **Custom** - Enables data caching, and allows to specify the length of time to keep the cached data.
- **Macro** - Enables data caching that is based on a macro.

Procedure

1. From **Team content** or **My content**, open a data module.
2. To specify data caching for a source, perform the following steps:
   a) Click the **Sources** pane to expand it, and locate the source.
   b) From the source context menu, select **Data cache**.
   c) Specify one of the cache options, and click **OK**.
3. To specify data caching for a table, perform the following steps:
   a) In the **Data module** pane, select one or more tables, and from the context menu, click **Properties**.
   b) Under **Advanced** properties, click **Data cache**.
   c) Specify one of the cache options, and click **OK**.

      Tip: The **Automatic** cache option is the option that was specified for the table source. For example, if the source data cache was set to **Custom**, with the time limit of 1 day, the automatic cache option for the table is shown as **Automatic (1 day)**. If multiple tables are selected, the cache option of the table that was selected first is shown as the automatic option for the selected group of tables.
4. Save the data module.

Securing data

You can secure data at the value level by creating security filters.

A security filter defines which users, groups, or roles have access to specific data values in a table. When the users work with dashboards, reports, or explorations that use the table, only the data that is included in the security filter is visible to them.

There are business reasons for restricting access to data at such low level of granularity. For example, you have confidential data that only specific users are allowed to see. Or, a table contains many records, and your users need only a subset of those records.

Before you begin

The schema metadata for the associated data server connections must be loaded, and you must have write permissions for the connections and their signons.

Tables that are based on typed-in SQL bypass security filters. To avoid potential security risks, specify the `ibmcognos.typeinsqldisabled` property on the data server connection that your data module is based on. If an attempt is made to create an SQL-based table after this property is specified, the table is not created. If this property is specified after an SQL-based table was created, the query execution is stopped. For more information about Cognos-specific connection parameters, see the *IBM Cognos Analytics Managing Guide*.

About this task

This type of data security can be implemented only for data server sources.

Procedure

1. From **Team content** or **My content**, open a data module.

   The data module source must be a data server, or another source that includes data server tables.
2. Click the **Sources** pane to expand it.
3. Expand the data server schema to view its tables.
4. From a table context menu, select **Set data security**, and click **Add security definition**.

5. In the **Set data security** dialog box, create the filters by associating specific users, groups, or roles with columns in the table.

   a) In the **Users, groups and roles** pane, click the add icon ⊕. In your authentication namespace, locate the users, groups, or roles for which you want to define access to the table data, and select their associated check boxes. The selected names appear in the **Selected users, groups and roles** pane.

   b) In the **Filters** pane, from the **Select a column** drop-down list, select one column, and click **Create a filter**. Specify the required filter conditions, and click **OK**. You can add filters for other columns in the same way. To add filters for multiple columns at once, from the **Select a column** drop-down menu, select the **via expression editor** option. Your security definition can include one or multiple filters.

   c) Specify a name for the security definition, and click **OK**.

   The security definition is added to the **Security filters** tab in the table properties. In the **Sources** panel, the padlock icon appears beside the table name.

Chapter 3. Modeling metadata

The initial data module that you create manually or using intent modeling can be modified, edited, and enhanced.

You can enhance your data module by adding new tables or sources, applying filters, creating calculations and navigation paths, changing column formatting, and more.

Relationships

A relationship joins logically related objects that the users want to combine in a single query. Relationships exist between two tables.

You can modify or delete relationships, or create new ones so that the data module properly represents the logical structure of your business. Verify that the relationships that you require exist in the data module, the cardinality is set correctly, and referential integrity is enforced.

The diagram provides a graphical view of table relationships in a data module. You can use the diagram to create, examine, and edit the relationships.

Creating a relationship from scratch

You need to create relationships whenever the required relationships are not detected by the IBM Cognos software.

About this task

Relationships can be created between tables from the same source and from different sources.

The diagram is the most convenient place to view all data module relationships, and quickly discover the disconnected tables.

Important: The list of possible keys in the relationship editor excludes measures. This means that if a row in a column was misidentified as a measure, but you want to use it as an identifier, you will not see the row in the key drop-down list. You need to examine the data module to confirm that the usage property is correct on each column in the table.

Procedure

1. In the data module tree or in the diagram, click the table for which you want to create a relationship, and from the table context menu ⋯, click **Relationship**.

   Tip: You can also start creating a relationship using the following methods:

   - In the data module tree or in the diagram, control-click the two tables that you want to join in a relationship, and click **Relationship**.
   - On the **Relationships** tab in the table properties, click **Add a relationship**.

   If the data module does not include the table that you need, you can drag this table from **Selected sources** directly onto the diagram.
2. In the relationship editor, specify the second table to include in the relationship, and then select the matching columns in both tables.

   Depending on the method that you used to start the relationship, the second table might already be added, and you only need to match the columns. You can include more than one set of matching rows in both tables.
3. Find the matching columns in both tables, and select **Match selected columns**.
4. Specify the **Realtionship Type**, **Cardinality**, and **Optimization** options for the relationship.

5. Click **OK**.

Results

The new relationship appears on the **Relationships** tab in the properties page of the tables that you joined, and in the diagram view.

To view or edit all relationships defined for a table, go to the **Relationships** tab in the table properties. Click the relationship link, and make the modifications. To view a relationship from the diagram, click the join line to open a small graphical view of the relationship. To edit a relationship from the diagram, right-click the join line, and click **Edit relationship**.

To delete a relationship for a table, go to the **Relationships** tab in the table properties, and click the remove icon for the required relationship. To delete the relationship from the diagram, right-click the line joining the two tables, and click **Remove**.

Creating tables by combining existing tables

You can create new tables in your data module by combining multiple tables into one query. Tables from any supported sources can be combined.

Procedure

1. Open an existing data module, or start creating a new one.
2. From the data module context menu, or from a table, multiple tables, or package context menu, select **Create new table**.

   - If you activated this command from a table or multiple tables context menus, the table names appear in the **Create new table** dialog box, in the **Selected tables** pane. The table that was selected first is at the top of the list.
   - If you activated this command from the data module context menu, no table names appear in the **Selected tables** pane initially. Click **Select tables** to add tables that are already in the data module.
   - If you activated this command from a package context menu, the **Create a joined view** dialog box is displayed. This is because tables sourced from packages can be used to create views only. Only relational objects in the package are visible. OLAP objects aren't visible.

     Finish creating the view. The remaining steps are not applicable when using packages.

   You can click the add icon to add new tables, or the remove icon to remove tables from the list. These options are not available for a package.

   Tip: It's important to decide which table should be first in the list. If the new table is based on multiple tables, the first table properties are applied to the new table. For example, the **Usage** or **Data cache** properties on the new table are inherited from the first table. The columns in the new table also inherit their properties, such as **Label** or **Aggregate**, from the columns in the first table.
3. Depending on your selection of tables in the previous step, one or more of the following options are available to create the new table. Select one of the available options.

   Create a copy of a table
   This option is available when one table is selected. You can copy all columns from the table, or only the columns that you select. The new table is disassociated from the base table so changes in one table aren't reflected in the other table.

   Create a view of a table
   This option is available when one or more tables are selected. The columns in the base tables are referenced in the view. The column properties in the view are independent from the base table. You can select or deselect columns to include in your new table.

Create a joined view
This option is available when two tables are selected. You can select or deselect columns to include in your new table. Proceed to create a join between the two tables. For more information, see "Creating a relationship from scratch" on page 13.

Create a union of tables
This option is available when two or more tables are selected. All selected tables have the same number of columns. The columns are in the same order, and their data types are compatible. The new table includes all rows from all selected tables.

Tip: The help icon in the product provides information about incompatible data types when columns with such data types are discovered.

Create an intersect of tables
This option is available when two tables are selected. Both tables have the same number of columns. The columns are in the same order, and their data types match. The new table includes only rows that are shared between the two tables.

Create an except of tables
This option is available when two tables are selected. Both tables have the same number of columns. The columns are in the same order, and their data types match. The new table includes only rows that exist in the first table.

4. Click **Finish**.

   The new table appears at the top of the data module tree.
5. In tables created by using the `union`, `intersect`, and `except` operations, duplicate columns are removed by default. To include the duplicates, from the table context menu, click **Properties**, and under **Advanced**, set the **Duplicates** property to **Preserve**.
6. Save the data module.

What to do next

You can use the new tables in the same way as other data module tables. For example, you can create relationships between this type of tables and other tables. You can also create calculations and navigation paths that include columns from these tables.

Creating tables using custom SQL

You can create new tables in a data module that are based on custom SQL syntax. The SQL is executed against a source that is already in the data module.

If the SQL validation is successful, the table is populated with a set of projected column names and rows of data.

The supported types of SQL include Cognos SQL, native SQL, and pass-through SQL. For more information, see "Supported SQL types " on page 16.

Procedure

1. From the data module context menu, select **Create table using SQL**.
2. In the table editor, type the table name.
3. From the **SQL type** drop-down menu, select the type of SQL to use.
4. From the **Source** drop-down menu, select the source to associate the table with. For data server connections, select the connection name. For other types of sources, select the source location, which is either **Team content** or **My content**.
5. In the **Expression** box, type or paste the SQL syntax for your table. The syntax is executed only against the source that you selected in the previous step.

   The expression editor provides the following syntax validation and editing options:

- - Validate the syntax. You can validate the whole statement, or only selected segments of code.
- - Preview columns and rows in your projected table. If the syntax is not correct, the columns are not displayed.
- - View descriptions of functions, and examples of their usage.
- - Insert the cursor anywhere in a line of code and select this button to comment out the entire line. To comment out multiple lines of code, select the lines and select this button. The comment string (--) is added at the beginning of each selected line.

  Tip: To comment out sections of code, manually enclose the text between the following strings: `/*` and `*/`
- < > - Apply formatting to the code.
- - Use high-contrast mode.
- Change the font size.

6. Click **OK** to save the table.

   You can save the table even if it contains syntax errors, and edit the syntax later. However, you cannot modify any aspect of the SQL table, or view its data in the grid, until the table is successfully validated.

Results

The table name appears at the top of the data module tree. To edit the table SQL, from the table context menu, click **Edit SQL table**. You can also edit a column expression in an SQL-based table. However, subsequent updates to the original SQL statement might overwrite the updated expression.

What to do next

You can use and model SQL-based tables in the same way as other data module tables. For example, you can create relationships between this type of tables and other tables. You can also create calculations and navigation paths that include columns from these tables.

Supported SQL types

IBM Cognos Analytics supports three types of SQL: Cognos SQL, native SQL, and pass-through SQL.

Cognos SQL

Cognos SQL is an implementation of the standard SQL. It works with all relational and tabular data sources. This is the optimal type of SQL for use with Cognos Analytics applications. Using Cognos SQL is preferable because you can have fewer database restrictions. Cognos SQL improves table performance by, for example, removing unused elements at query time.

Cognos SQL does not support non-standard SQL.

Native SQL

Native SQL uses vendor-specific SQL constructs. Use native SQL to pass the SQL statement that you enter to the database. Cognos Analytics might add statements to what you enter. A native SQL statement can reference only one data source.

This type of SQL must be completely self-contained. It can't reference anything outside that SQL, such as database prompts, variables, or native formatting that would normally be supplied by the calling application.

If you are comfortable working with a native SQL version, you can use keywords that are not available in Cognos SQL, and copy and paste SQL statements from other applications to Cognos Analytics.

This type of SQL might not work with a different data source.

Pass-Through SQL

Use pass-through SQL when the SQL statement that you enter is not valid inside a derived table, such as in a `With` or `OrderBy` clause. Generally, you should use pass-through SQL only if you must create a table that contains source-specific constructs.

Pass-through SQL lets you use native SQL without any of the restrictions that the data source imposes on subqueries (pass-through SQL tables are not processed as subqueries). Instead, the SQL for each table is sent directly to the data source where the query results are generated.

Performance is slower because each table is sent to the source as a separate statement rather than being optimized by Cognos Analytics. Therefore, when choosing between native SQL and pass-through SQL, you must decide which is more important: performance or using SQL that is not permitted in a subquery.

Pass-through SQL must be completely self-contained. It must not reference anything outside of that SQL, such as database prompts, variables, or native formatting that would normally be supplied by the calling application.

A pass-through SQL statement might not work with a different data source.

Specifying column dependencies

Specify column dependencies to clarify data granularity in data module tables to avoid double-counting of repeated values when data is aggregated.

There are two common scenarios when column dependency can be specified for a table to avoid double-counting of values.

- A table contains replicated data (denormalized table).

  For example, a table that contains `City Population` and `Country Population` has the values for the `Country Population` repeated for all the cities that belong to a certain country.
- A table is joined to a fact table, and the join uses columns from the first table that have repeating values (blending data).

  For example, a table that contains data for each calendar date is joined to a table that contains data at a year level. The values for each year will be accessed for every date value, ending up with values that are inflated by a factor of 365.

Using the column dependencies feature, you create groups of columns that depend on a specific attribute. The groups are also related to each other in an order from coarse to fine granularity.

Tip: An attribute is a column that has the **Usage** property set to **Attribute** or **Identifier**, and a fact is a column that has the **Usage** property set to **Measure**.

About this task

You do not need to specify column dependencies for all tables. Do it only when double-counting would take place. Your decision to specify column dependencies affects other Cognos Analytics components, such as reports or dashboards.

Procedure

1. Open en existing data module or start creating a new module that is based on a source containing tables that include repeated data at different levels of granularity.
2. Identify the attribute and fact columns that could cause double-counting. For example, the table can contain data at the month, quarter, and year level.

   Verify if the column properties and data formats are properly specified. For example, change the **Usage** property to **Attribute**, or assign the data format of **Currency** to fact (measure) columns. Save the data module.

3. To see how the data is aggregated before column dependency is specified, you can create a report using your saved data module as a source. Later, you can use this report to verify the effect of adding column dependency groups on data aggregation.
4. From the table context menu, select **Specify column dependencies**.

   The column dependency view is opened.
5. Drag the attribute columns that you identified in step 2, such as `Current Year`, `Current Quarter`, and `Current Month`, from the data module panel to the column dependency view.
6. Drag the measures, such as `Year_Quantity`, `Quarter_Quantity`, and `Month_Quantity`, inside the related attribute area.

   Form a group around each attribute. Add other, related columns to the groups. Each column in the table should be in one group.
7. Order the columns in the group from coarse to fine granularity.
8. Save the data module.
9. To see how the data is aggregated after column dependency is specified, you can create a report using your saved data module as a source. Compare the aggregated results with the report from step 3.

Results

The following example shows how columns can be grouped.

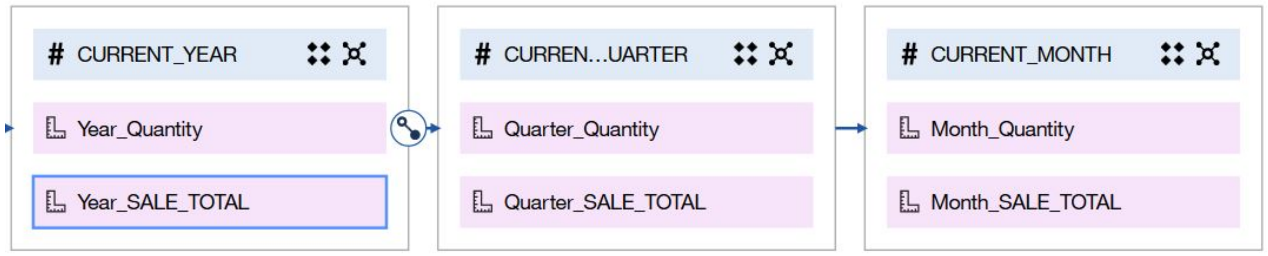


Figure 1. Example of column dependencies in a table

Hiding tables and columns

11.0.4

You can hide a table or column in a data module. The hidden tables or columns remain visible in the modeling interface, but they are not visible in the reporting and dashboarding interfaces. The hidden items are fully functional in the product.

About this task

Use this feature to provide an uncluttered view of metadata for the report and dashboard users. For example, when you hide columns that are referenced in a calculation, the metadata tree in the reporting and dashboarding interfaces shows only the calculation column, but not the referenced columns. When you hide the identifier columns used as keys for joins, the keys are not exposed in the dashboarding and reporting interfaces, but the joins are functional in all interfaces.

Procedure

1. In the data module tree, click the context menu icon for a table or column, and click **Hide**.

   You can also select multiple tables or columns to hide them at once.

   Tip: To un-hide the items, click the context menu icon for the hidden table or column, and click **Show**.
2. Save the data module.

Results

The labels on the hidden tables and columns are grayed out in the data module tree and in the diagram. Also, on the **General** tab of the table or column properties, the check box **This item is hidden from users** is selected.

The hidden tables and columns are not visible in the reporting and dashboarding interfaces.

Calculations

Calculations allow you to answer questions that cannot be answered by the source columns.

You can create basic arithmetic calculations and field concatenations, and more complex, custom calculations.

All calculations are added to the data module as selectable columns that can be used in reports, dashboards, explorations, and other Cognos Analytics content.

Creating basic calculations

You can create basic arithmetic calculations for columns with numeric data types, and concatenate text values for columns with the text data type.

About this task

The expression for these calculations is predefined and you only need to select it. For example, you can create a column `Revenue` by multiplying values for `Quantity` and `Unit price`. You can create a column Name by combining the `First name` and `Last name` columns.

Procedure

To create a simple arithmetic calculation for columns with numeric data types, use the following steps:

a) In the data module tree, right-click the column for which you want to create a calculation. For calculations that are based on two columns, use control-click to select the columns.

b) Click **Create calculation**.

c) In the **Create calculation** dialog box, type the calculation name and specify the calculation **Expression**.

d) Leave clear or select the **Calculate after aggregation** check box.

When you leave this check box clear, the calculation is performed on the column values before they are aggregated. If you select this check box, the calculation is performed after the values are aggregated. The calculation results might be different in each case.

e) To view the calculation expression, click **Use calculation editor**.

f) Click **OK**.

Results

The calculation appears as a stand-alone column at the top of the list of columns in the table. You can move it to a folder inside the same table.

What to do next

The calculation can be used as other columns in the table. From its context menu, you can access the different actions to edit the calculation, format its data, or remove it.

The calculation can be used in reports, dashboards, explorations, and other Cognos Analytics content.

Creating custom calculations

To create a custom calculation, you must define your own expression using the expression editor.

About this task

Custom calculations can be created at the data module level, at the table level, and at any folder level. The module-level and root folder-level calculations can reference columns from multiple tables.

Procedure

1. In the data module tree, right-click the data module name, a table name, or a folder name, and click **Calculation**.
2. In the expression editor, type the calculation name.
3. In the **Expression** box, define the expression for your calculation using the expression editor capabilities.
   - To enter a function for your expression, type the first character of the function name, and select the function from the drop-down list of suggested functions.
   - To add table columns to your expression, drag-and-drop one or more columns from the data module tree to the **Expression** box. The column name is added where you place the cursor in the expression editor.

     Tip: You can also double-click the column in the data module tree, and the column name appears in the expression editor.
4. Click **Validate** to check if the expression is valid. The calculation will be created even if the expression is not valid, but it won't be usable.
5. Click **OK**.

Results

Depending where you created it, the calculation appears as a stand-alone column at the top of the list of columns in a table, at the root of the data module, or in a folder.

What to do next

The calculation can be used as other columns in the table. From its context menu, you can access the different actions to edit the calculation, format its data, or remove it.

The calculation can be used in reports, dashboards, explorations, and other Cognos Analytics content.

Filters

A filter specifies the conditions that rows must meet to be retrieved from tables when the data module is used with reports, dashboards, explorations, and other Cognos Analytics content.

Data modules support the following types of filters:

- Embedded filters.

  These filters are always applied to a table when the data module is used in dashboards, reports, explorations, and so on.
- Selectable filters.

  These filters are created as selectable items in the data module tree, inside a table or at the root of the data module. The users can decide whether to apply these filters to dashboards, reports, explorations, and so on.

Security filters are a different type of filters that are specified at the source level. For more information, see “Securing data” on page 10.

Creating embedded filters

Create embedded filters to customize the view of data in the data module for specific use cases.

For example, you can filter out data that is irrelevant for certain geographies, time periods, product lines, and so on.

About this task

Embedded filters can be crated for a single column, or for multiple columns in a table.

Procedure

1. In the data module tree, locate the table that you want to add filters to, and choose one of the following options:
   - To create a filter for a single column in the table, from the column context menu, click **Filter**.
   - To create filters for multiple columns in a table, from the table context menu, click **Manage filters**. On the **Filters** tab, select the column for which you want to create the filter, and click **Add a filter**.

     The option **via expression editor** allows you to create a filter by using the expression editor. Specify the filter name, type its expression, and click **OK**. Next, you can either continue adding filters by using this option, or proceed to step 2.
2. Specify the filter values. The options to select values depend on the column data type.
   - For columns with integer and decimal data types, you have two options to specify the values: **Range** and **Individual items**. When you choose **Range**, you can use the slider to specify the range values, or type the range start and end values. When you choose **Individual items**, select the check boxes that are associated with the values.

     Tip: You can enter decimal values only for columns with the decimal data type. When you try to enter a decimal value for a column with the integer data type, the decimal separator is cleared.
   - For columns with date and time (timestamp) data types, specify a range of values before, after, or between the selected date and time, or select individual values.
   - For columns with text data types, select the check boxes that are associated with the values.

   To select values that are outside the range that you specified, click **Invert**.
3. If you used the **Manage filters** option, you can continue creating filters for other columns in the table. The filters that you created are listed on the **Filters** tab.

Results

After an embedded filter is created, the filter icon appears next to the column and table names in the data module panel, diagram, and expression editor.

The filter icon on a table indicates that the table contains at least one embedded filter. All embedded filters that a table contains are listed in the table properties, on the **Filters** tab.

What to do next

To edit a filter on a single column, from the column context menu click **Filter**, and modify the filter conditions. To remove the filter, click **Clear all**.

To edit or remove embedded filters for a table, from the table context menu click **Manage filters**. On the **Filters** tab, view or edit the filters, or remove them.

Creating selectable filters

Create selectable filters when you want to give users the choice of applying the filters in dashboards, reports, explorations, and so on.

The filters suggest possible options to filter data in the data module, but the users can also view unfiltered data.

About this task

Selectable filters can be created inside a table or outside a table, at the root of the data module.

Procedure

1. Open a data module and decide what type of a selectable filter you want to create.
   - To create a filter inside a table, from the table context menu, click **Filter**.

     The **Filter** option is also available for a folder inside the table, and for the table in the diagram.
   - To create a filter outside a table, from the data module context menu, click **Filter**.

     The **Filter** option is also available for a folder at the root of the data module.

   The expression editor is displayed.
2. Type the filter name.
3. In the **Expression** panel, provide the filter expression by using the expression editor capabilities.
4. Click **OK**

Results

The filter is added as a stand-alone entry in the data module tree with the filter icon before the filter name. Filters created at the folder-level are added to the applicable folder.

What to do next

To edit the filter, from the filter context menu, click **Edit filter**. To remove the filter, from the filter context menu, click **Remove**.

Creating data groups

You can organize the column data into custom groups so that the data is easier to read and analyze.

About this task

You can create two types of custom groups, depending on the column data type: one group type for columns with numeric data and the second group type for columns with text data. For example, in the `Employee code` column you can group employees into ranges, such as 0-100, 101-200, 200+. In the `Manager` column, you can group managers according to their rank, such as `First line manager`, `Senior manager`, and so on.

Procedure

1. In the data module tree, right-click the column where you want to group data, and click **Create data group**.
2. If you selected a numeric column, set the groups in the following way:
   a) In the **Grouped column name** field, specify a name for the grouped column.
   b) From the **Groups** menu, select the number of groups that you want to create. Each group is automatically assigned an equal number of values. When you change the number of groups, the values are dynamically redistributed and the range values are set.
   c) In the **Group names** column, replace the automatically-generated labels with meaningful names. However, if you change the number of groups, the custom labels are cleared, and the automatically-generated labels are restored for each group.
   d) If needed, manually lock the **Range border values** for each group. The **Higher** and **Lower** range border values can be changed to numeric inputs. To return to the equal distribution of values, click the **Reset distribution** icon.

e) To create a group for NULL values, select the **Group NULL values as** check box, and type a name for the group.

f) Click **Create**.

The grouped column appears at the top of the list of columns in the table. However, if you selected the **Replace the existing column** check box, the grouped column replaces the original column in the table.

3. If you selected a text column, set the groups in the following way:

   a) In the **Grouped column name** field, specify a name for the grouped column.

   b) In the **Groups** box, click **New group**, and type a group name.

   c) In the **Remaining items in column** box, control-select the values to include in the group, and click the **Add to group** → icon. The values are moved to the **Group items** box.

   d) Repeat steps b to c to create more groups.

   e) Optional: To create a group that contains all of the values that aren't already included in any group, select the **Group remaining and future values in** check box, and type a name for the group.

   f) Optional: To replace the original column in the table with the grouped column, select the **Replace the existing column** check box.

   g) Click **OK**.

   The grouped column appears at the top of the list of columns in the table. However, if you selected the **Replace the existing column** check box, the grouped column replaces the original column in the table.

Results

To edit the grouped column, right-click the column, and select **Edit data group**.

Cleaning data

Data is often messy and inconsistent. You might want to impose some formatting order on your data so that it's clearer and easier to read.

About this task

The **Clean** options that are available for a column depend on the column data type. Some options can be specified for multiple columns with the same data type, and some for singular columns only.

The following options are available to clean your data:

Whitespace

Trim leading and trailing whitespace

Select this check box to remove leading and trailing whitespace from strings.

Convert case to

UPPERCASE, **lowercase**, **Do not change**

Use this option to change the case of all characters in the string to either uppercase or lowercase, or to ensure that the case of each individual character is unchanged.

Return a substring of characters

Return a string that includes only part of the original string in each value. For example, an employee code can be stored as CA096670, but you need only the number 096670 so you use this option to remove the CA part. You can specify this option for singular columns only.

For the **Start** value, type a number that represents the position of a character in the string that will start the substring. Number 1 represents the first character in the string. For the **Length** value, specify the number of characters that will be included in the substring.

NULL values

11.0.4

Specify NULL-handling options for columns with text, numeric, date, and time data types that allow NULL values. When Cognos Analytics detects that a column does not allow NULL values, these options are not available for that column.

The default value for each option depends on the column data type. For text data, the default is an empty string. For numbers, the default is 0. For dates, the default is 2000-01-01. For time, the default is 12:00:00. For date and time (timestamp), the default is 2000-01-01T12:00:00.

The entry field for each option also depends on the column data type. For text, the entry field accepts alphanumeric characters, for numbers, the entry field accepts only numeric input. For dates, a date picker is provided to select the date, and for time, a time picker is provided to select the time.

The following NULL-handling options are available:

Replace this value with NULL

Replaces the text, numbers, date, and time values, as you specify in the entry field, with NULL.

For example, if you want to use an empty string instead of NULL in a given column, but your uploaded file sometimes uses the string `n/a` to indicate that the value is unknown, you can replace `n/a` with NULL, and then choose to replace NULL with the empty string.

Replace NULL values with

Replaces NULL values with text, numbers, date, and time values, as you specify in the entry field.

For example, for the Middle Name column, you can specify the following values to be used for cells where middle name does not exist: `n/a`, `none`, or the default empty string. For the Discount Amount column, you can specify `0.00` for cells where the amount is unknown.

Procedure

1. In the data module tree, click the context menu icon for a column, and click **Clean**.

   Tip: To clean data in multiple columns at once, control-select the columns that you want to clean. The **Clean** option is available only if the data type of each selected column is the same.
2. Specify the options that are applicable for the selected column or columns.
3. Click **Clean**.

Results

After you complete the **Clean** operation, the expression editor automatically creates an expression for the modified column or columns. To view the expression, open the column properties panel, and click the expression that is shown for the **Expression** property.

Creating navigation paths

A navigation path is a collection of non-measure columns that business users might associate for data exploration.

When a data module contains navigation paths, the dashboard users can drill down and back to change the focus of their analysis by moving between levels of information. The users can drill down from column to column in the navigation path by either following the order of columns in the navigation path, or by choosing the column to which they want to proceed.

About this task

You can create a navigation path with columns that are logically related, such as year, month, quarter, week. You can also create a navigation path with columns that are not logically related, such as product, customer, state, city.

Columns from different tables can be added to a navigation path. The same column can be added to multiple navigation paths.

A data module can have multiple navigation paths.

Procedure

1. In the data module panel, start creating a navigation path by using one of the following methods:

   - From the data module context menu ⋯, click **Properties**, and then click the **Navigation paths** tab. Click **Add a navigation path**. In the **Create navigation path** dialog box, drag columns from the data module panel to the navigation path panel. Change the order of columns as needed. Click **OK**.
   - In the data module tree, select one or more columns, and from the context menu ⋯ of any of the selected columns, click **Create navigation path**. The selected columns are listed in the **Create navigation path** dialog box. Click **OK**.

   Tip: The default name of the navigation path includes names of the first and last column in the path. You can change this name.
2. Save the data module to preserve the navigation path.
3. To modify a navigation path, from the data module context menu ⋯, click **Properties**, and then click the **Navigation paths** tab. Click the **Edit** link for the path that you want to modify. In the **Edit navigation path** dialog box, you can make the following modifications:

   - To add different columns, drag the columns from the data module to the navigation path. You can multi-select columns and drag them all at once.
   - To remove columns, click the remove ⊖ icon for the column.
   - To change the order of columns, drag them up or down.
   - To change the navigation path name, overwrite the existing name.

     The default name reacts to the changed order of columns. If you overwrite the default name, it does no longer change when you modify the group definition. The name cannot be blank.

Results

The navigation path is added to the data module and is available to users in dashboards and stories. If you select the option **Identify navigation path members** in the data module toolbar, the columns that are members of the navigation path are underlined.

What to do next

The modeler can modify the navigation path at any time, and re-save the data module.

To view the navigation path that a column belongs to, from the column context menu ⋯, click **Properties** > **Navigation paths**. Click the navigation path name to view or modify its definition.

To view all navigation paths in a data module, from the data module context menu ⋯, click **Properties** > **Navigation paths**. Click the navigation path name to view or modify its definition. To delete a navigation path, click the remove ⊖ icon for the path.

Formatting data

The column data format specifies how column values are displayed in dashboards, stories, explorations, or reports. You can choose a format type such as text, percent, or currency.

Uploaded Microsoft Excel spreadsheets retain the column data formats that were defined in Excel. These formats are set as default data formats in the base data modules that are created from these spreadsheets.

Each format type contains properties that further specify how the data is displayed. Initially, the properties are set based on the format options that are returned from the source. If no option is returned from the source, the property is set to the data module default.

About this task

Some characters are language-sensitive and display properly only when your locale supports the applicable font. For example, for Japanese currency symbols to display correctly, your locale must be set to Japanese.

You can define the format type for several columns at the same time if the columns contain the same type of data.

Procedure

1. In the data module tree, from a column context menu , click **Format data**.
2. In the **Data format** dialog box, select the appropriate **Format type**.

   For example, you can select **Date**, **Time**, **Date/Time**, or other type.

   If the data was not formatted in the source, the **Unformatted** type is assigned in the data module. If the source supplies a format, the format is reflected in the module.

   The **Custom** format is an advanced option where you can supply your own format string and use it to format the data.
3. Specify properties for the selected format type.

   For example, for the **Date** type, you can specify the **Date separator**, the **Date style**, and **Date ordering** properties. For the **Currency** type, you can specify the **Number of decimal places** property.

   The **Default** setting is specific to the property that it refers to. You can view the default for each property by clicking the information icon next to the property. For example, for the thousands separator the default value is inherited from the user's content language. For the decimal places default, if the property is not set, the number of decimal places vary depending on the number rendered.

   Click **Advanced options** to view and specify more data format properties. Click **Reset properties** if you want to reset the properties to default values.
4. Click **OK** to apply the formatting.

Validating data modules

Use the validation feature to check for invalid object references within a data module.

About this task

Validation identifies the following errors:

- A table or column that a data module is based on no longer exists in the source.
- A calculation expression is invalid.
- A filter references a column that no longer exists in the data module.
- A table or column that is referenced in a join no longer exists in the data module.

Procedure

1. In the application bar or in the **Data module** panel, click the validation icon . You can also click **Validate** from the data module context menu.

   The following indicators specify that the data module contains errors:

- The number of errors is displayed on top of the validation icon in the application bar
- The failed validation icon is displayed in the data module tree and in the diagram, next to the column where the error is discovered.

2. Click the error icons to view the number of errors and the details about the errors.

   The errors are listed in the **Validation issues** panel.
3. View the error messages by clicking the **Show details** link for each issue. To view the log of all errors, click the validate icon in the application bar , and then click **View details**.

   To copy the error messages to the clipboard, click the copy icon.
4. Using the validation messages, try to resolve the errors.

   You can save the data module with validation errors, and resolve the errors later.

Table and column properties

You can view and modify the table and column properties in a data module.

The properties can be accessed from the table or column context menu , on the **Properties** panel, **General** tab.

Label

Specifies the table or column name. You can change the name as needed.

This item is hidden from users

Use this property to hide a table or column in a data module. The hidden tables or columns remain visible in the modeling interface, but are not visible in the reporting and dashboarding interfaces. For more information, see "Hiding tables and columns" on page 18.

Expression

Shows the underlying expression for a column. Clicking the expression opens the expression editor where you can modify the expression.

Comments

Use this property to specify optional information about the table or column. The comment is not available outside of the modeling environment.

Screen tip

Use this property to specify an optional, short description of the table or column. The screen tip appears when you pause your pointer over the table or column name in the modeling, reporting, or dashboarding environment.

Usage

This property identifies the intended use for the data in the column.

The initial property value is based on the type of data that the column represents in the source. You need to verify that the property is set correctly. For example, if you import a numeric column that participates in a relationship, the **Usage** property is set to **Identifier**. You can change this property.

The following **Usage** types are supported:

- Identifier

Represents a column that is used to group or summarize data in a **Measure** column with which it has a relationship. It can also represent an index, date, or time column type. For example, `Invoice number`, or `Invoice date`.

- Measure

  Represents a column that contains numeric data that can be grouped or summarized, such as `Product Cost`.

- Attribute

  Represents a column that is not an **Identifier** or a **Measure**, such as `Description`.

Aggregate

The **Aggregate** property defines the type of aggregation that is applied to a column that summarizes data in a report or dashboard. For example, if the **Aggregate** property value of the `Quantity` column is **Total**, and it is grouped by `Product Name` in a report, the `Quantity` column in the report shows the total quantity of each product. Aggregated data improves query performance and helps to retrieve data faster.

The default type of aggregation is inherited from the source. When modifying this property, you can select values that the source does not provide, such as average or maximum. To know which aggregate value is required, you must understand what your data represents. For example, if you aggregate `Part number`, the aggregate values that apply are count, count distinct, maximum, and minimum.

The following aggregation types are supported:

- None (no aggregation is set up for a column)
- Average
- Count
- Count distinct
- Maximum
- Minimum
- Total

Data type

The column data type is inherited from the source and can't be modified in the data module.

Represents

Use this property to specify whether a column includes the date/time or geographic location type of data. This information is used in the reporting and dashboarding environments to suggest the most appropriate default visualizations, among other possibilities.

- Geographic location

  The values include **Continent**, **Sub Continent**, **Country**, **Region**, **State Province**, **County**, **City**, **Postal code**, **Street Address**, **Position**, **Latitude**, and **Longitude**.

- Time

  The values include **Date**, **Year**, **Quarter**, **Season**, **Month**, **Week**, **Day**, **Hour**, **Minute**, and **Second**.

Sorting

Use this property to enable or disable sorting for a column, and to specify the row to sort by, the sorting order, and the placement of NULL values in the column.

Identifier

For tables and columns, the property value is inherited from the source and cannot be changed in the data module. The column or table **Identifier** value can be, but not always is, the same as the column or table name (**Label**).

You can view the **Identifier** property in the **Advanced** section of the **Properties** panel.

Source

Shows the source name and path for a table or column. You can view the **Source** property in the **Advanced** section of the **Properties** panel

Chapter 4. Relative date analysis

With the relative dates feature, you can analyze measures filtered by date periods that are relative to a particular date. Examples of relative date filters include current quarter, last quarter, quarter-to-date, or month-to-date.

When using relative dates, you can create reports and dashboards that show date-filtered results in different visualizations, crosstabs, and so on. By default, the date-filtered measures in the data use today's date as the reference date in the analysis.

The implementation of this feature uses a subset of Cognos Analytics 11.1.0 base samples that include the sample calendar data modules, **Gregorian Calendar** and **Fiscal Calendar**. Ensure that the samples are available in your Cognos Analytics installation before you start other, related tasks. For more information, see "Sample calendars " on page 31.

Tip: The base samples are installed with the Cognos Analytics server, and an administrator imports the sample deployment **Samples for Install_11_1_0** to the content store. For more information, see "Importing the base samples" in the *IBM Cognos Analytics Samples Guide*.

To enable relative date analysis in Cognos Analytics, you must create a data module that maps your data to a calendar. This data module can then be used as a source for relative date analysis in reports and dashboards. For more information, see "Creating a data module for relative date analysis" on page 33.

If you want to customize the reference date for relative date analysis based on user roles, use the **_as_of_date** global parameter. For more information, see "Customizing the reference date " on page 42.

After the Cognos Analytics environment is set up for relative date analysis, users can use the relative date filters and measures to perform analysis of data in reports and dashboards. For more information, see "Relative date analysis" in the *IBM Cognos Analytics Reporting Guide*.

Video

Here's a video that captures the process of creating a data module for relative date analysis: video (https://youtu.be/4lDrF3jx98g).

Sample calendars

The IBM Cognos Analytics base samples include a set of sample calendars that you need to set up relative date analysis.

The sample calendar data modules and their sources can be found in the **Team content** > **Calendars** folder.

The following samples are available in this folder:

- **Gregorian calendar** data module.

  Contains dates from January 1, 1950 to December 31, 2050.
- **Fiscal calendar** data module.

  Contains dates from March 1, 1950 to February 28, 2050.
- **Fiscal calendars** folder.

  This folder contains 12 sample calendar data modules. Each of these calendars covers 100 years (1950 to 2050), but the dates in each calendar start in a different month. The month in the calendar name indicates the start of a fiscal year. For example, the **02. February 1** data module is the sample calendar for the fiscal year starting on February 1.

The **03. March 1** calendar is the same as the **Fiscal calendar**, and the **01. January 1** calendar is the same as the **Gregorian calendar**.

- **Source files** folder.

  This folder contains the source .csv files for the **Gregorian calendar** and the fiscal calendars data modules. The files contain dates for the associated calendars.

Columns and dates in the sample calendar data modules

The dates are listed in the following columns:

TheDate
The main reference date for each row.

PD_TheDate
Previous day from **TheDate**. The dates in this column are one day prior to **TheDate**.

ND_TheDate
Next day from **TheDate**. The dates in this column are one day after **TheDate**.

dYear
The date that is the beginning of the year to which **TheDate** belongs.

Tip: The fiscal calendars are denoted by the year in which the calendar ends.

PY_TheDate
Previous year for **TheDate**. The dates in this column are one year prior to **TheDate**.

NY_TheDate
Next year for **TheDate**. The dates in this column are one year after **TheDate**.

dQuarter
The date that is the beginning of the quarter to which **TheDate** belongs.

PQ_TheDate
Previous quarter for **TheDate**. The dates in this column are one quarter prior to **TheDate**.

NQ_TheDate
Next quarter for **TheDate**. The dates in this column are one quarter after **TheDate**.

dMonth
The date that is the beginning of the month to which **TheDate** belongs.

PM_TheDate
Previous month for **TheDate**. The dates in this column are one month prior to **TheDate**.

NM_TheDate
Next month for **TheDate**. The dates in this column are one month after **TheDate**.

Important: Do not modify the column names in the sample data modules and .csv files because that would break the relative date filters in the sample calendars.

Relative date filters in the sample calendar data modules

All sample calendar data modules contain a set of predefined, date-related filters. These filters are used to perform relative date analysis in different types of visualizations.

The following filters are predefined in the sample calendars:

- Prior month
- Prior quarter
- Prior year
- Current month
- Current quarter
- Current year

- MTD (month to date)
- QTD (quarter to date)
- YTD (year to date)
- Prior MTD
- Prior QTD
- Prior YTD
- Same month last quarter
- Same month last year
- Same quarter last year
- Same MTD last quarter
- Same MTD last year
- Same QTD last year

You can view the expression associated with each filter by clicking **Edit filter** from its context menu.

Creating a data module for relative date analysis

To enable relative date analysis, you need to create a data module where business data is associated to a calendar.

In this data module, at least one date column must be associated to a calendar, and at least one measure column must be associated to the date column. This association is done by using the column property **Lookup reference**.

Before you begin

The sample calendars must be available.

About this task

You can create a new data module from scratch, or add relative date capabilities to an existing data module.

Tip: The **Team content** > **Samples** > **Relative dates** folder contains the **Boston 311 report** and **Boston 311 dashboard** samples that illustrate the implementation of this feature in a report and dashboard. The **Data** folder that is included with these samples contains the associated data modules and their source .csv files. You can use these samples as a reference when creating your data module.

Procedure

1. Start creating a new data module, or open an existing data module.
2. Verify that your business data sources contain at least one date column and one measure column.
   a) From the date column context menu, select **Properties** > **General**. Ensure that the **Data type** property of the column is set to `Date`.

      If the **Data type** property is set to `Timestamp`, you can change the type to `Date` by using the `cast` function in the expression editor.

      If the data source is an Excel file or a CSV file, dates in the date column must be formatted using the ISO 8601 notation *`yyyy-mm-dd`*.

   b) From the measure column context menu, select **Properties** > **General**. Ensure that the **Usage** property of the column is set to `Measure`.

      If the **Usage** property is set to `Identifier`, you can change the property to `Measure`.

Tip: If your data module source is linked to its source, which is indicated by the link icon , you need to break the link. Otherwise, the data module is read-only, and you can't modify its properties. To break the link, select the **Break link** option from the data module context menu. However, do not break links in any of the sample calendar data modules.

3. In the **Data module** panel, click the **Add sources and tables** icon to add a calendar source, which can be one of the following sources:

   - The sample **Gregorian calendar** data module in the **Team content** > **Calendars** folder.
   - The sample **Fiscal calendar** data module in the **Team content** > **Calendars** folder.
   - One of the sample data modules in the **Team content** > **Calendars** > **Fiscal calendars** folder.

4. In your business data source that you specified in steps 1 and 2, associate at least one date column to the calendar, and at least one measure column to the date column.

   a) For the date column that you want to associate to the calendar, open **Properties**, and locate the **Lookup reference** property. From the **Lookup reference** drop-down menu, select the name of the calendar source that you added to the data module. If needed, repeat this step for other date columns.

      The relative date filters, such as **Prior year**, **Prior month**, **MTD**, and so on, appear under the date column. To view the full list of filters, see "Sample calendars " on page 31.

   b) For the measure column that you want to associate to the date, open **Properties**, and locate the **Lookup reference** property. From the **Lookup reference** drop-down menu, select the date column to reference. If you defined **Lookup reference** for multiple date columns, choose the date column that is appropriate for this measure. If needed, repeat this step for other measure columns.

      Tip: To specify the same **Lookup reference** property for multiple measure columns, multi-select the columns, and set the property.

      A set of date-filtered measures, with the measure name in square brackets, appears under the measure column. For example, `Prior year [Revenue]`, `Prior month [Revenue]`, or `MTD [Revenue]`.

   To use one or more of the date-filtered measures in calculations, you can create the calculations only against the data-module, and not against the tables that contain these measures. The calculations appear at the top of the data module tree.

5. Save the data module to a folder in **Team content**.

   Tip: If you add or remove a filter from a calendar data module, the data modules that reference this calendar through the **Lookup reference** property don't reflect the change until you close and re-open them.

Results

The data module can now be used to create dashboards and reports.

Creating relative date filters

A relative date filter specifies a range of dates that are relative to the **_as_of_date** global parameter.

The sample Gregorian and Fiscal calendars already contain a number of predefined relative date filters. If you need custom filters, you can add them to these calendars.

Before you begin

1. Know the columns in the sample calendar.

   New filters are added to the sample Gregorian or Fiscal calendar data modules. To understand how these calendars are structured, look at the columns, and view dates in different columns in the same row. For example, in the Gregorian calendar data module, for the column **TheDate** with the value of September 30, 2018, the related values for columns **dYear**, **PY_TheDate**, and **dMonth** are shown in the following table:

TheDate	dYear	PY_TheDate	dMonth
2018-09-30	2018-01-01	2017-09-30	2018-09-01

For more information, see "Sample calendars " on page 31.

2. Conceptualize the filter lower bound and upper bound in relation to the **_as_of_date** global parameter.

   A relative date filter defines a range of dates between the filter lower bound (range start) and upper bound (range end) dates. The lower and upper bounds are set against a reference date that is the **_as_of_date** parameter value.

   For example, for a year-to-date (YTD) filter, the lower bound date is the first day of the first month in the year that contains the **_as_of_date** date. The upper bound date is the date that is the **_as_of_date** parameter value. If the **_as_of_date** parameter is December 19, 2018 (**TheDate**), the lower bound date is January 1, 2018, and the upper bound date is December 19, 2018.

   By default, the **_as_of_date** parameter has a value of today. However, it can be set to a different date. For more information, see "Customizing the reference date " on page 42

3. Build the filter expression.

   The critical element of creating a relative date filter is the filter expression. Familiarize yourself with the expression syntax and variables before you start entering the code in the expression editor. For more information, see "Creating filter expressions" on page 36.

Procedure

1. From the **Team content** > **Calendars** folder, open the sample calendar data module where you plan to add the new filter.

   The data module contains one table with a number of existing filters. Add your new filter to this table.
2. From the table context menu, click **Filter**.
3. In the filter editor that is displayed, type the new filter name.
4. In the **Expression** pane, type or paste the filter expression.

   For example, to create the Last 12 months filter, enter the following expression:

```
// validate: 1 = 1
#$_this.parent.idForExpression# >=
        #queryValue($_this.parent.split.ref + '.dMonth',
                    $_this.parent.split.ref + '.TheDate = ' +
                      queryValue($_this.parent.split.ref + '.PY_theDate' ,
                          $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
AND
#$_this.parent.idForExpression# <
       #queryValue($_this.parent.split.ref + '.dMonth',
           $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
```

   Another filter example that you can use is Next 4 months.

   For more information, see "Creating filter expressions" on page 36.
5. Validate the expression.

   Validation of date filter expressions must be done manually because the validate button in the expression editor doesn't validate macro expressions. So you can only visually confirm that the following elements are correct:

   - The expression is preceded with the `// validate: 1 = 1` comment.
   - The outer block of the `queryValue` macro function is enclosed within hash marks (#).

- Each `queryValue` has matching round brackets `()` for its two arguments.

  Tip: You can debug filter expressions in Reporting. To do so, open a report that contains the relative date filters, and set the validation option in the report to **Information**.

6. Click **OK**.
   The new filter is added to the calendar table at the top of the list of filters. You can drag the filter to change its position in the list. The filter is created even if the expression contains errors. To modify the filter, from its context menu, click **Edit filter.**

Results

The new filter is now available to the data modules that reference this calendar through the **Lookup reference** property, and can be used for relative date analysis.

Tip: The new filter, as other filters in the calendar, should remain hidden.

Creating filter expressions

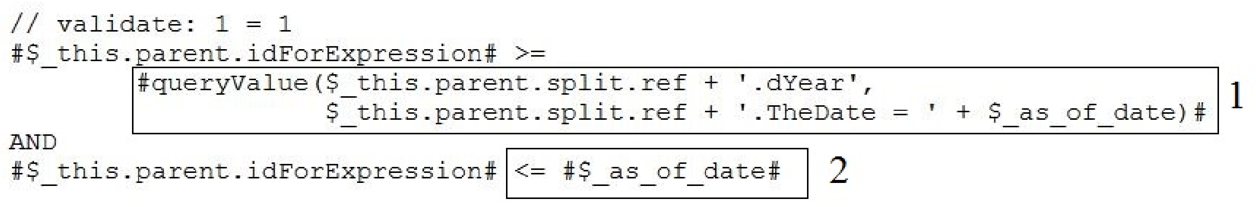
A relative date filter is based on an expression. The expression defines the filter lower and upper bounds, and the timeline between the bounds. The timeline is mapped to the `queryValue` macro.

When you create a new filter, you enter the expression in the expression editor.

Use the following syntax to create the filter expression:

```
// validate: 1 = 1
#$_this.parent.idForExpression# >= lower_bound_date expression#
AND
#$_this.parent.idForExpression# <= upper_bound_date expression#
```

For example, the year-to-date (YTD) filter that is available in the sample calendars data modules uses the following expression:



```
// validate: 1 = 1
#$_this.parent.idForExpression# >=
        #queryValue($_this.parent.split.ref + '.dYear',
                    $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#  1
AND
#$_this.parent.idForExpression# <= #$_as_of_date#  2
```

In this filter, expression 1 is the filter *lower_bound_date expression*. Expression 2 is the filter *upper_bound_date expression*.

The lower bound and upper bound code blocks are combined by using the AND operator.

Tip: The comment `// validate: 1 = 1` must always be included at the beginning of the expression.

The *lower_bound_date expression* and *upper_bound_date expression* are the elements that you must define for your filter. The remaining part of the expression remains unchanged for all filters.

For a description of variables that are used in relative date filter expressions, see "Expression variables " on page 38.

To define the lower bound and upper bound expressions, you need to complete the following tasks:

- Identify the move intervals for the filter timeline
- Map each move interval to one queryValue macro

Identify the move intervals for the filter timeline

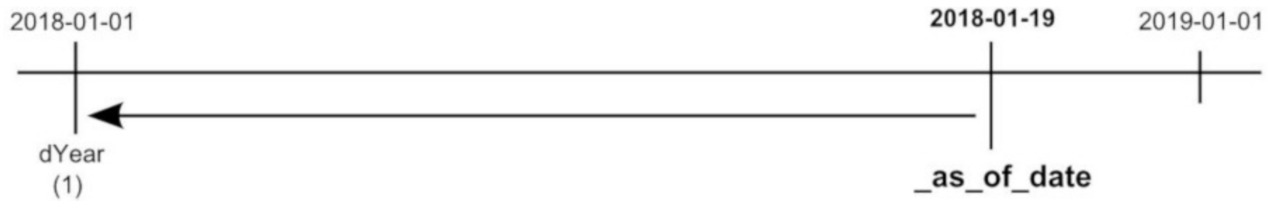
A timeline starts from the **_as_of_date** date, and then uses one or more move intervals (units of time) to reach the lower bound or upper bound date.

The sample calendars support the following move intervals: day, month, quarter, and year.

A move interval is expressed by using the following calendar columns:

- **PD_TheDate** - move to the previous day
- **ND_TheDate** - move to the next day
- **dYear** - move back to the first day of the year
- **PY_TheDate**- move back to the same or equivalent date in the previous year
- **NY_TheDate** - move forward to the same or equivalent date in the next year
- **dQuarter** - move back to the first day of the quarter
- **PQ_TheDate** - move back to the same or equivalent date in the previous quarter
- **NQ_TheDate** - move forward to the same or equivalent date in the next quarter
- **dMonth** - move back to the first day of the month
- **PM_TheDate** - move back to the same or equivalent date in the previous month
- **NM_TheDate** - move forward to the same or equivalent date in the next month

The type of filter implies which columns are used to express the timeline. For example, in the year-to-date (YTD) filter, the **dYear** column is the lower bound move interval, as shown in the following graphic. There is no upper bound move interval for this filter.



Map the move intervals to queryValue macros

After you identify the move intervals for the filter lower and upper bounds, you need to map each move interval to one `queryValue` macro.

The `queryValue` uses the following syntax.

```
#queryValue($_this.parent.split.ref + move_interval ,
$_this.parent.split.ref + ‘.TheDate =‘ + date)#
```

For example, here is how the move interval **dYear** is mapped to the `queryValue` macro (parts of the code in bold font) in the year-to-date (YTD) filter expression:

```
// validate: 1 = 1
#$_this.parent.idForExpression# >=
        #queryValue($_this.parent.split.ref + '.dYear',
                    $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
AND
#$_this.parent.idForExpression# <= #$_as_of_date#
```

Depending on the type of date filter, your expression might include multiple, nested `queryValue` macros, such as in the following **Prior YTD** filter:

```
// validate: 1 = 1
#$_this.parent.idForExpression# >=
  #queryValue($_this.parent.split.ref + '.dYear',
              $_this.parent.split.ref + '.TheDate = ' +
              queryValue($_this.parent.split.ref + '.PY_TheDate',
                         $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)
  )#

AND
#$_this.parent.idForExpression# <=
    #queryValue($_this.parent.split.ref + '.PY_TheDate',
                $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
```

You can find another example of nested `queryValue` macros in the Next 4 months filter.

Expression variables

The relative date filter expression uses a set of variables to define the filter conditions. The variables evaluate to specific values when the filter is applied in visualizations.

Use the information in this topic when creating relative date filter expressions.

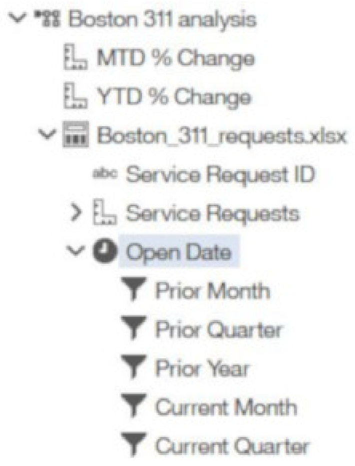
The following **Prior Year** filter expression from the sample Gregorian calendar can be used as an illustration when reading descriptions of the variables:

```
// validate: 1 = 1
#$_this.parent.idForExpression# >=
   #queryValue($_this.parent.split.ref + '.dYear',
               $_this.parent.split.ref + '.TheDate = ' +
                      queryValue($_this.parent.split.ref + '.PY_TheDate',
                                 $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)
)#
AND
#$_this.parent.idForExpression# <
    #queryValue($_this.parent.split.ref + '.dYear',
                $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
```

`$_this.parent`

This variable refers to a table column of type `Date` whose **Lookup reference** property is set to the calendar that you are referencing. The table with the `Date` column is in the data module that is used for relative date analysis. For more information, see "Creating a data module for relative date analysis" on page 33.

For example, in the following data module, `$_this.parent` refers to the `Open Date` column.



The following two variables function within the context of the `Date` column:

- `$_this.parent.idForExpression`

  This variable evaluates to the `idForExpression` for the `Date` column. The `idForExpression` is the full identifier that uniquely identifies the `Date` column within the data module. This identifier is not viewable from the user interface.

- `$_this.parent.split.ref`

  This variable evaluates to the calendar that is referenced by the **Lookup reference** property of the `Date` column.

All date filters in the calendar, including the new ones that you add, are accessible as child filters of the `Date` column in the data module that references this calendar through the **Lookup reference** property. The filters are used for relative date analysis in reports and dashboards.

`queryValue`

`queryValue` is one of the macro functions that Cognos Analytics provides.

Tip: To view the description of the `queryValue` macro, click the functions tab in the expression editor and search for the macro. The description is shown in the **Information** pane.

Within the context of relative date filters, `queryValue` returns the date value from the specified `Date` column, at the specified date. The specified `Date` column is the first parameter to the `queryValue` function. The second parameter is the specified date.

In the following example from the year-to-date (YTD) filter, the `queryValue` returns, from the calendar column `dYear`, the date where the calendar `TheDate` date equals to the `_as_of_date` date.

```
#queryValue($_this.parent.split.ref + '.dYear',
                    $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
```

Example filter: Last 12 months

This topic provides an expression for a date filter that includes the last 12 months relative to the **_as_of_date** parameter.

Paste this expression in the filter editor to create the `Last 12 months` date filter.

```
// validate: 1 = 1
#$_this.parent.idForExpression# >=
     #queryValue($_this.parent.split.ref + '.dMonth',
                 $_this.parent.split.ref + '.TheDate = ' +
                      queryValue($_this.parent.split.ref + '.PY_theDate' ,
                                 $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
AND
#$_this.parent.idForExpression# <
     #queryValue($_this.parent.split.ref + '.dMonth',
                 $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
```

Here are the steps that were used to build this expression:

1. Identify the calendar columns to use.

   The filter uses the calendar columns **TheDate**, **dMonth**, and **PY_TheDate** from the sample **Gregorian calendar** data module.

   For more information, see "Sample calendars " on page 31.

2. Define the filter lower and upper bounds.

   The filter lower bound is the first day of the month that is 12 months prior to the month containing the date that is represented by the **_as_of_date** parameter. The filter upper bound is the last day of the last complete month, relative to the date that is represented by the **_as_of_date** parameter.

   The following table shows the move intervals for the lower bound date when the **_as_of_date** date (**TheDate**) is January 19, 2019.

TheDate	**PY_TheDate**	**dMonth**
2019-01-19	2018-01-19	
2018-01-19		2018-01-01

   The filter lower bound date is 2018-01-01. The filter upper bound date is 2018-12-31.

3. Define the move intervals for the lower bound and upper bound timelines.

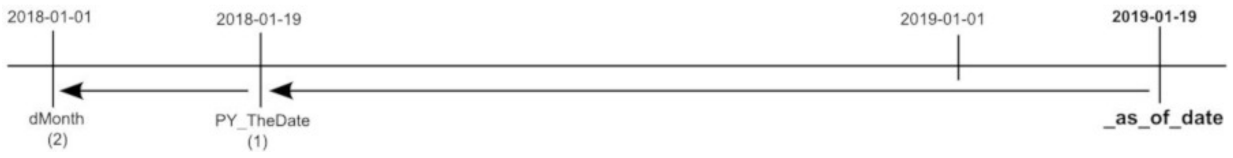
   The timeline consists of move intervals that are based on the columns **PY_TheDate** and **dMonth** .

The following move intervals exist for the lower bound timeline:

- Move interval 1: **PY_TheDate**
- Move interval 2: **dMonth**

The move interval for the upper bound timeline is **dMonth**.

Here is a graphical representation of the move intervals for the lower bound timeline when January 19, 2019 is the **_as_of_date** date.



4. Map each move interval to one `queryValue` macro.

   The expression of the lower bound consists of two `queryValue` macros. Each `queryValue` maps to one move interval within the lower bound expression. The initial move interval (**PY_TheDate**) is nested within the second move interval (**dMonth**), as shown below:

```
#$_this.parent.idForExpression# >=
        #queryValue($_this.parent.split.ref + '.dMonth',
                    $_this.parent.split.ref + '.TheDate = ' +
                      queryValue($_this.parent.split.ref + '.PY_theDate' ,
                                 $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
```

   The expression of the upper bound consists of one `queryValue` macro, as shown below:

```
#$_this.parent.idForExpression# <
      #queryValue($_this.parent.split.ref + '.dMonth',
                  $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
```

   This expression uses the less than (<) sign because the filter includes only dates prior to the upper bound, and not equal to the upper bound itself.

Example filter: Next 4 months

This topic provides an expression for a date filter that includes the next 4 months relative to the **_as_of_date** parameter.

You can paste this expression in the filter editor to create the `Next 4 months` date filter.

```
// validate: 1 = 1
#$_this.parent.idForExpression# >=
        #queryValue($_this.parent.split.ref + '.dMonth',
                    $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
AND
#$_this.parent.idForExpression# <
      #queryValue($_this.parent.split.ref + '.dMonth',
                  $_this.parent.split.ref + '.TheDate = ' +
          #queryValue($_this.parent.split.ref + 'NM_The_Date',
                      $_this.parent.split.ref + '.TheDate = ' +
              #queryValue($_this.parent.split.ref + 'NM_The_Date',
                          $_this.parent.split.ref + '.TheDate = ' +
                  #queryValue($_this.parent.split.ref + 'NM_The_Date',
                              $_this.parent.split.ref + '.TheDate = ' +
                       #queryValue($_this.parent.split.ref + 'NM_The_Date',
                                   $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)
                        )
                   )
              )
          )
      )#
```

Here are the steps that were used to build this expression:

1. Identify the calendar columns to use.

   The expression uses the **TheDate**, **dMonth**, and **NM_TheDate** columns from the sample **Gregorian calendar**.

   For more information, see "Sample calendars " on page 31.
2. Define the filter lower and upper bounds.

   The filter lower bound is the first day of the month containing the date that is represented by the **_as_of_date** parameter. The upper bound is the last day of the month that is 3 months after the month containing the **_as_of_date** date.

   The following table shows the move intervals for the upper bound dates when the **_as_of_date** date (**TheDate**) is December 19, 2018.

TheDate	**NM_TheDate**	**dMonth**
2018-12-19	2019-01-19	2018-12-01
2019-01-19	2019-02-19	
2019-02-19	2019-03-19	
2019-03-19	2019-04-19	
2019-04-19		2019-04-01

3. Define the move intervals for the lower bound and upper bound.

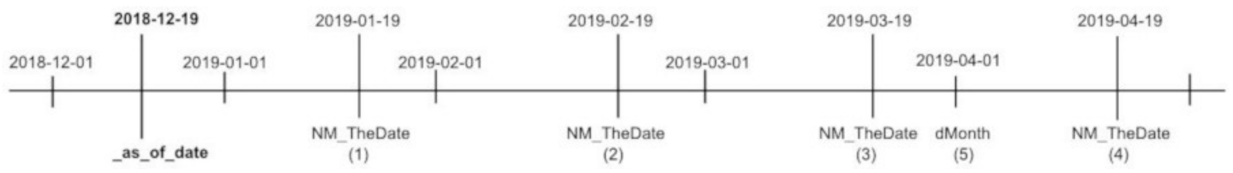
   The filter timeline consists of move intervals that are based on the columns **NM_TheDate** and **dMonth** .

   The move interval for the lower bound timeline is **dMonth**.

   The upper bound timeline includes the following move intervals:

   - Move interval 1: **NM_TheDate**
   - Move interval 2: **NM_TheDate**
   - Move interval 3: **NM_TheDate**
   - Move interval 4: **NM_TheDate**
   - Move interval 5: **dMonth**

   Here is a graphical representation of the timeline when the **_as_of_date** date is December 19, 2018.



4. Map each move interval to one `queryValue` macro.

   The lower bound expression consists of one `queryValue` macro, as shown below:

```
#$_this.parent.idForExpression# >=
        #queryValue($_this.parent.split.ref + '.dMonth',
                    $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)#
```

   The upper bound expression consists of 5 `queryValue` macros, nested within each other. Each `queryValue` maps to one move interval. The earlier move intervals are nested within the later move intervals, as shown below:

```
#$_this.parent.idForExpression# <
```

```
       #queryValue($_this.parent.split.ref + '.dMonth',
                   $_this.parent.split.ref + '.TheDate = ' +
           #queryValue($_this.parent.split.ref + 'NM_The_Date',
                       $_this.parent.split.ref + '.TheDate = ' +
               #queryValue($_this.parent.split.ref + 'NM_The_Date',
                           $_this.parent.split.ref + '.TheDate = ' +
                   #queryValue($_this.parent.split.ref + 'NM_The_Date',
                               $_this.parent.split.ref + '.TheDate = ' +
                        #queryValue($_this.parent.split.ref + 'NM_The_Date',
                                    $_this.parent.split.ref + '.TheDate = ' + $_as_of_date)
                         )
                    )
               )
           )
       )#
```

This expression uses the less than (<) sign because the filter includes only dates prior to the upper bound, and not equal to the upper bound itself.

Customizing the reference date

If the global parameter **_as_of_date** was set up by your administrator, you can change the date that your relative date periods are based on.

By default, the relative date periods are based on the current date. For example, when the current date is July 15, 2018, the YTD (year-to-date) filter includes data from January 1 to July 15, 2018, and the prior month filter includes data from June 1 to June 30, 2018. When you set a specific date as a value for the **_as_of_date** parameter, your analysis is done as of that date.

Before you begin

After the **_as_of_date** parameter is set up by the administrator, log out, and log back in to Cognos Analytics for this parameter to be displayed for you. For more information, see "Setting the _as_of_date global parameter" on page 42.

Procedure

1. In the Cognos Analytics welcome page, select the icon in the application bar.
2. For the **As Of Date** parameter, select a new date by using the calendar picker, and click **Apply**.
3. Re-run the reports and dashboards that use relative dates.

Results

The data in the reports and dashboards is updated based on the new reference date.

Setting the _as_of_date global parameter

As an administrator, you can set up the global parameter **_as_of_date**, and specify which roles it applies to. Using this parameter, the users can change the date that their relative date filters are based on.

Before you begin

Ensure that your samples **Team content** > **Calendars** > **Tools** folder contains the **Global parameter date picker** report.

Procedure

1. Go to **Manage** > **Customization**, and select the **Parameters** tab.
2. Click **Import**, and navigate to the sample report **Team content** > **Calendars** > **Tools** > **Global parameter date picker**.

as_of_date is added to the list of parameters. From the parameter context menu, click **Properties**. Here, you can add a description of the parameter, disable it, or specify a language-specific label for the parameter.

3. Assign roles for the **as_of_date** parameter, and specify its value.
   a) Go to **Manage** > **People**, and select the **Accounts** tab.
   b) Locate the role to which you want to assign this parameter, and in the role **Properties** panel, select the **Customization** tab.
   c) Next to **Parameters**, click **Settings**.
   d) Select the **as_of_date** parameter check box.

      Now, you can click **OK** to finish setting this parameter without changing the default date, which is the current date.

      If you want to set a specific date, select the **Set values** link, and using the calendar picker, select the date, and click **Apply**.
   e) If needed, repeat steps b to d for other roles. The date that you select can be different for different roles.
4. Ensure that the users log out, and log back in, to see the **as_of_date** global parameter in their **My parameters** ✓.

Results

The **as_of_date** parameter is now available to users who are members of the selected roles every time they run reports or dashboards that include relative date filters and measures that are filtered by relative dates.

Index