# Human Capital Analytics Reporting
# (HCA Reporting 12.0.2)
# User Guide

unicorn
human capital management

**Last Updated: 4/11/2024**

**Revision 1.2**

## Contents

# HCA Reporting

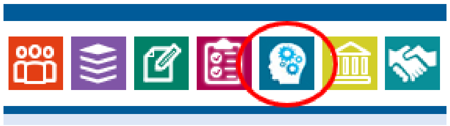
This document is intended for use with Unicorn HCM's Human Capital Analytics (HCA) Reporting solution. Unicorn HCM's HCA Reporting offering provides users with powerful solutions for analyzing & monitoring benefit, human resources and payroll information that is stored and managed in the HCM application.

HCA Reporting is the portal to HCM reporting and is based on IBM Cognos Analytics software. A user can use the tool to run reports and dashboards, author ad hoc reports, distribute and schedule reports.

# Accessing HCA Reporting

You must have Manager Services function security to HCA Reporting in order the run reports and dashboards. Users that need the ability to author or edit reports must have function security to HCA Author. See HCA Author.

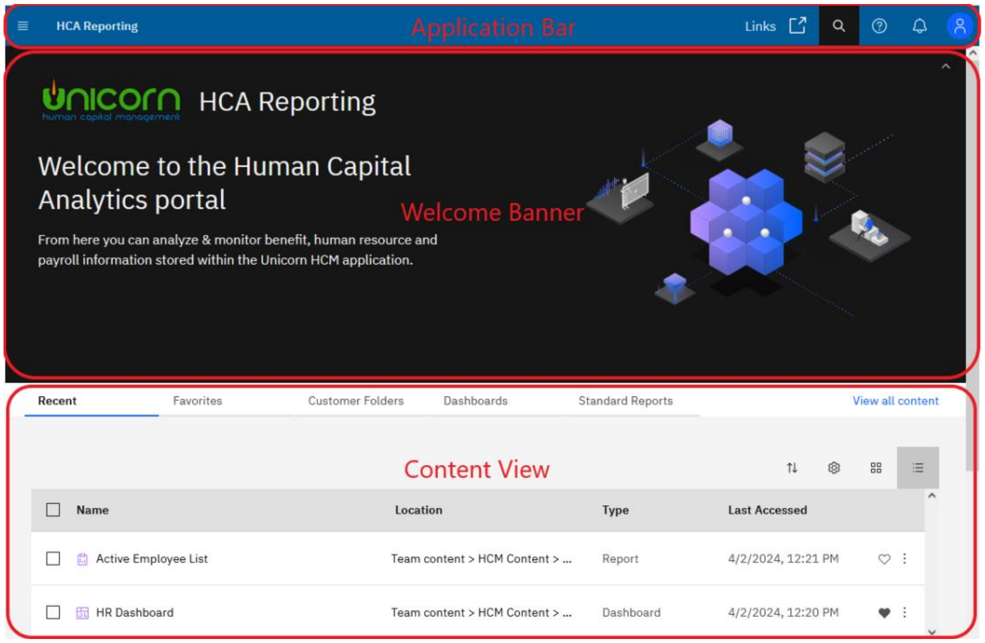
In Manager Services, start HCA Reporting by clicking the HCA Reporting tile or navigate to the Tools > HCA Reporting menu item.



# Layout

The **Application Bar** runs along the top. It provides access to Help, personal menus including Schedules and Sign Out as well as a view switcher.

The **Content View** has tabs to help you access reporting content. There is a tab with your recently run reports, a tab with your selected favorites, a tab going to a folder for your customer and tabs for accessing the standard dashboards and reports. The "View all content" link will open the full content view.

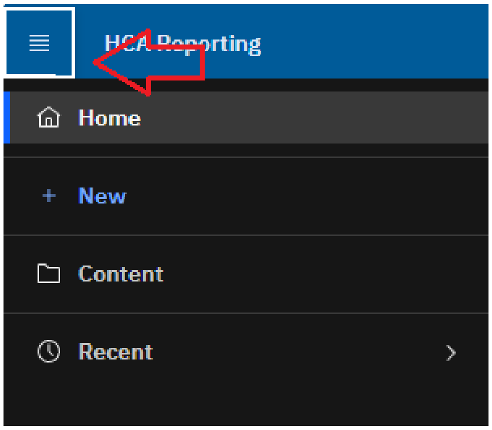


Welcome to the Human Capital Analytics portal

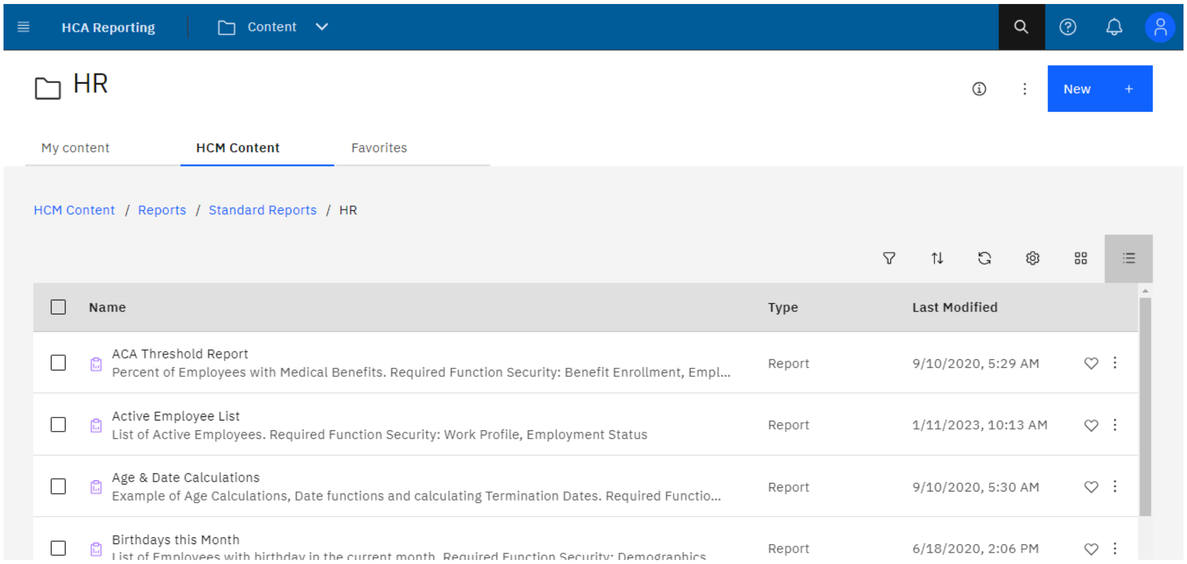
From here you can analyze & monitor benefit, human resource and payroll information stored within the Unicorn HCM application.

| Name | Location | Type | Last Accessed |
|---|---|---|---|
| Active Employee List | Team content > HCM Content > ... | Report | 4/2/2024, 12:21 PM |
| HR Dashboard | Team content > HCM Content > ... | Dashboard | 4/2/2024, 12:20 PM |

The **Open Menu** provides access to exiting content, creating new reports or to see recently run reports. You can navigate with in your personal My Content folder or navigate thru Shared content from here.



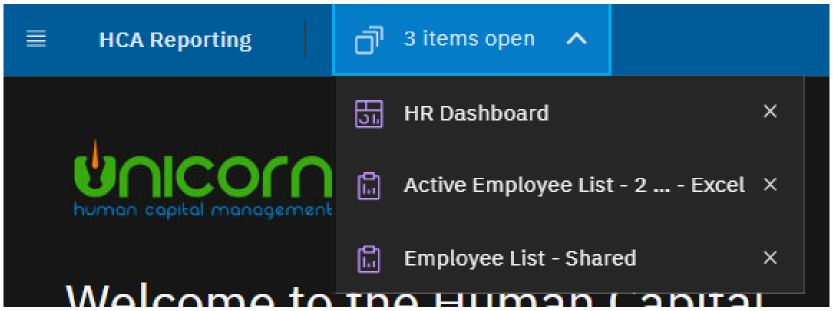
The **Content** menu item opens the **Content View** where you can navigate between your existing personal content in the “My Content” folder and shared and standard content under “HCM Content”. The Context Menu may display Folders or Reports, depending on what is selected. The top line of the Context Menu provides a trail for navigating around the folder hierarchy.



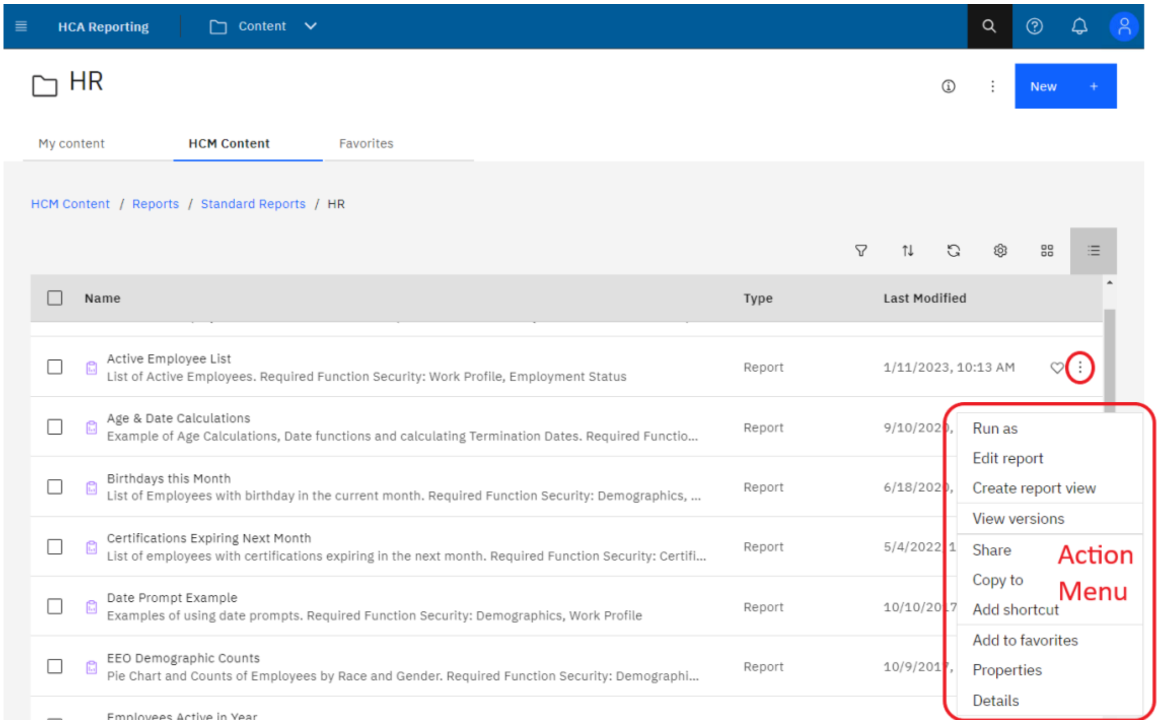
HCM Content / Reports / Standard Reports / HR

| Name | Type | Last Modified |
|---|---|---|
| ACA Threshold Report<br>Percent of Employees with Medical Benefits. Required Function Security: Benefit Enrollment, Empl... | Report | 9/10/2020, 5:29 AM |
| Active Employee List<br>List of Active Employees. Required Function Security: Work Profile, Employment Status | Report | 1/11/2023, 10:13 AM |
| Age & Date Calculations<br>Example of Age Calculations, Date functions and calculating Termination Dates. Required Functio... | Report | 9/10/2020, 5:30 AM |
| Birthdays this Month<br>List of Employees with birthday in the current month. Required Function Security: Demographics | Report | 6/18/2020, 2:06 PM |

**Tip**: If you have several reports open you can use the View Switcher drop down in the Application Bar to navigate between open reports and dashboards.



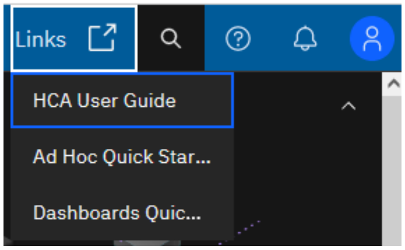
The **Action Menu** is a flyout menu that provides access to features that are relevant to the selected item. The available items will be different depending on the user's security rights or the object selected.



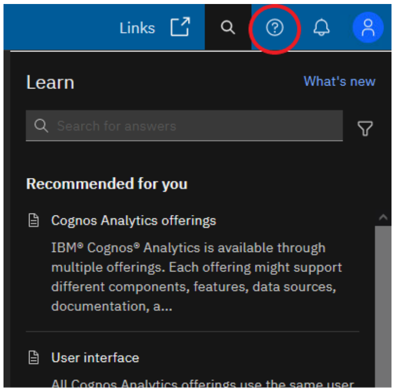
| Name | Type | Last Modified |
|---|---|---|
| Active Employee List<br>List of Active Employees. Required Function Security: Work Profile, Employment Status | Report | 1/11/2023, 10:13 AM |
| Age & Date Calculations<br>Example of Age Calculations, Date functions and calculating Termination Dates. Required Functio... | Report | 9/10/202 , |
| Birthdays this Month<br>List of Employees with birthday in the current month. Required Function Security: Demographics, ... | Report | 6/18/202 , |
| Certifications Expiring Next Month<br>List of employees with certifications expiring in the next month. Required Function Security: Certifi... | Report | 5/4/2022 1 |
| Date Prompt Example<br>Examples of using date prompts. Required Function Security: Demographics, Work Profile | Report | 10/10/20 7 |
| EEO Demographic Counts<br>Pie Chart and Counts of Employees by Race and Gender. Required Function Security: Demographi... | Report | 10/9/201 , |
| Employees Active in Year | | |

# Help

Links to HCA Reporting specific user guides are available under the Links icon in the Application bar.



IBM Help is available in the Learn section in the upper right corner of the Application bar.



Cognos Analytics offerings

IBM® Cognos® Analytics is available through multiple offerings. Each offering might support different components, features, data sources, documentation, a...

User interface

All Cognos Analytics offerings use the same user

## Finding reports

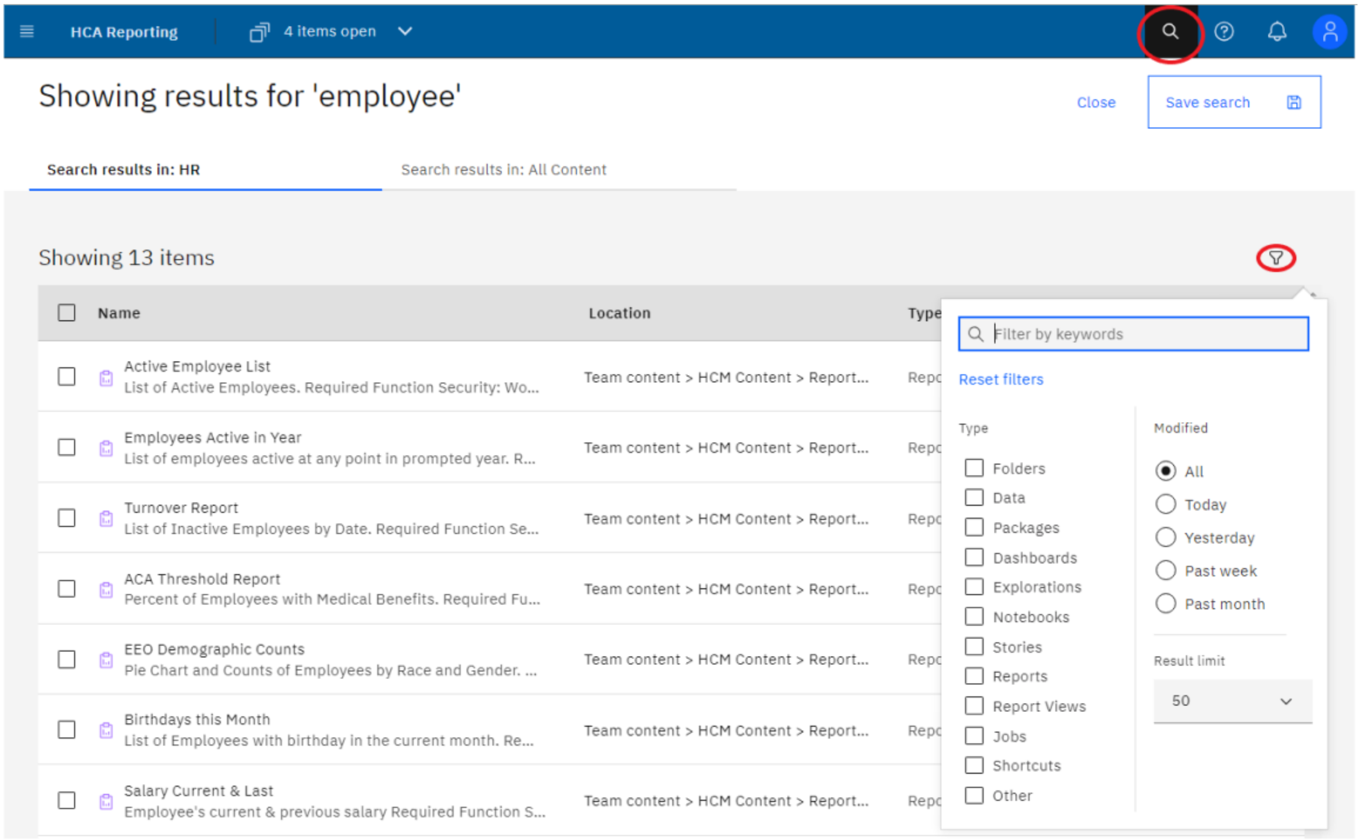
Reports will be available in 3 possible locations.

My Content – this folder is visible only to the user. Reports saved here will only be available to the current user.

Customer Shared Folder – this folder is visible to all user for the customer. This folder will be in this location: Team Content  > HCM Content > Reports > Customer Folder > {*customer number*} > Shared.

Cognos Group folder – this folder is visible to only users defined in a Security Group. Groups and memberships are defined in the HCA Security function. These folder locations will be: Team Content  > HCM Content > Reports > Customer Folder > {*customer number*} > Groups > {*group name*}.

You can use **Search** to find reports by Name. The "Search content" box is on the right side of the Application Bar. The search will return a list of objects and their location. You can use the filter icon to further refine your results.



Showing results for 'employee'

Showing 13 items

| □ | Name | Location | Type |
|---|---|---|---|
| □ | Active Employee List<br>List of Active Employees. Required Function Security: Wo... | Team content > HCM Content > Report... | Repo |
| □ | Employees Active in Year<br>List of employees active at any point in prompted year. R... | Team content > HCM Content > Report... | Repo |
| □ | Turnover Report<br>List of Inactive Employees by Date. Required Function Se... | Team content > HCM Content > Report... | Repo |
| □ | ACA Threshold Report<br>Percent of Employees with Medical Benefits. Required Fu... | Team content > HCM Content > Report... | Repo |
| □ | EEO Demographic Counts<br>Pie Chart and Counts of Employees by Race and Gender. ... | Team content > HCM Content > Report... | Repo |
| □ | Birthdays this Month<br>List of Employees with birthday in the current month. Re... | Team content > HCM Content > Report... | Repo |
| □ | Salary Current & Last<br>Employee's current & previous salary Required Function S... | Team content > HCM Content > Report... | Repo |

Type
- □ Folders
- □ Data
- □ Packages
- □ Dashboards
- □ Explorations
- □ Notebooks
- □ Stories
- □ Reports
- □ Report Views
- □ Jobs
- □ Shortcuts
- □ Other

Modified
- ● All
- ○ Today
- ○ Yesterday
- ○ Past week
- ○ Past month

Result limit
50

Recently run reports will be listed on the Portal homepage. Recently run reports are also available under the **Recent** items entry on the navigation bar.

## Viewing and interacting with reports

Navigate to a report or search and find a report in the **Recent**, **Shared, Groups** or **My content** folder, you can then click on the report name to run the report.

The report opens in a viewer. When a report runs in the interactive viewer, you also have options to see different data in the report by filtering, drilling up and down, and more.

Or from the entry context menu ..., click **Run as**. You can then choose the output format.

## Copying or moving reports

When you create a copy of an entry, you create a replica of that entry in another location in the portal. When you move an entry, you remove it from the current folder and place it in another folder. When you copy and move entries, the IDs and links are either maintained or overwritten. You must have read permissions for the entry that you are attempting to copy or move. You must also have permissions for the current folder, and write permissions for the target folder.

1. Navigate to the report. From the entry context menu ..., click **Copy or move**.
2. Locate the target folder, and click **Copy to**.

## Setting the package for a report

If you copy a report to a new environment you may need to set the package associated with the report to the customer number currently logged into. An example is copying a report from Live to Test.

1. Navigate to the report. From the entry context menu ..., click **Properties**.
2. Under General > Advanced, Use the “Set” link under “Source Package or data module” to change the Source Package. Select the package for the customer currently logged into. See Selecting a Package.

## Renaming a report

1. Navigate to the report. From the entry context menu ..., click **Properties**.
2. Click the report name field at the top of the Properties pane. This will place the cursor in the name field, edit the name as desired.

## Share your content by email

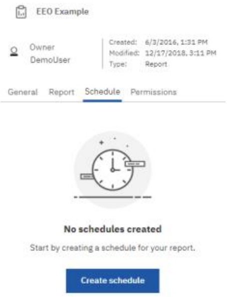
You can share Cognos® Analytics content in an email. When you are viewing your content, click the Share button .

## Scheduling a report

You schedule an entry, such as a report, to run it later or at a recurring date and time.

Only one schedule can be associated with each entry. If you require multiple schedules for a report, you can create report views, and then create a schedule for each view or create copies of the report.

1. Click the entry context menu ···, and then click **Properties**.
2. In the **Properties** pane, click the **Schedule** tab, and then click **Create Schedule**.



EEO Example

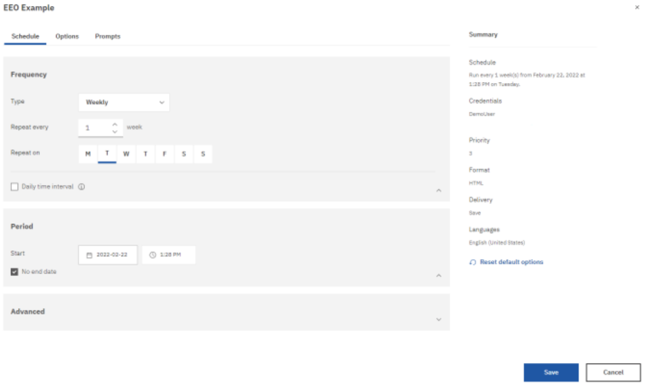
Owner
DemoUser

Created: 6/3/2016, 1:31 PM
Modified: 12/17/2018, 3:11 PM
Type: Report

**No schedules created**

Start by creating a schedule for your report.

3. In the **Create schedule** pane, specify the frequency, options & saved prompt values (if required). Set up any email recipients under the Options tab > “Send report by email” > “edit details”.



EEO Example

**Frequency**

Type: Weekly

Repeat every: 1 week

Repeat on: M **T** W T F S S

□ Daily time interval ⓘ

**Period**

Start: 2022-02-22 1:28 PM

☑ No end date

**Advanced**

Summary

Schedule
Run every 1 week(s) from February 22, 2022 at 1:28 PM on Tuesday.

Credentials
DemoUser

Priority
3

Format
HTML

Delivery
Save

Languages
English (United States)

If you no longer need a schedule, you can delete it. You can also disable schedules without losing any of the scheduling details. The schedules can be enabled later.

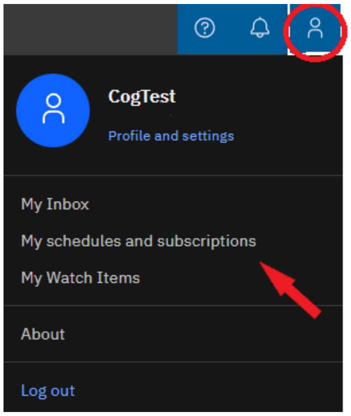
# My schedules and subscriptions

You can view all your scheduled activities and subscriptions on the **My schedules and subscriptions** panel.

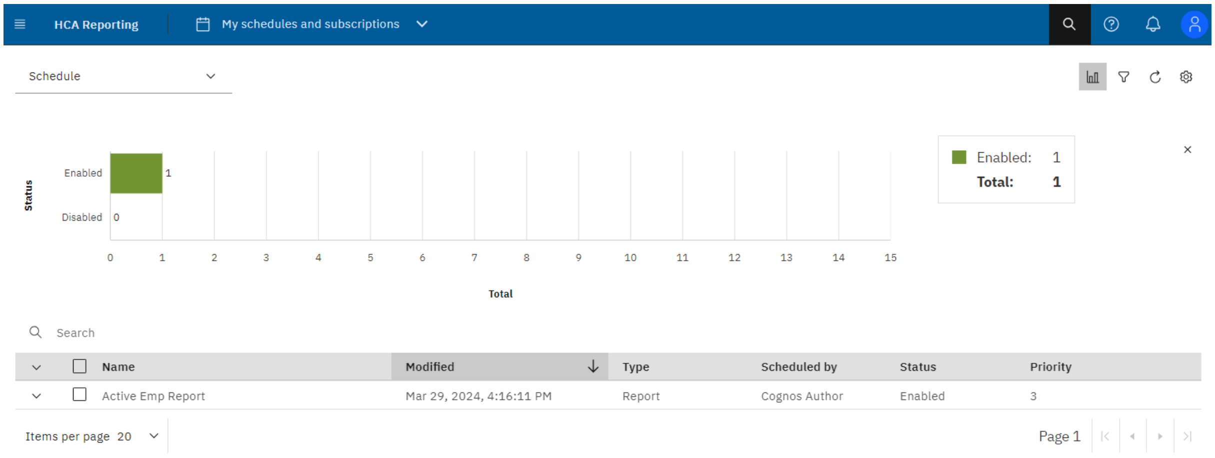
You can view a list of your scheduled activities that are current, past, or upcoming on a specific day. You can filter the list so that only the entries that you want appear. A bar chart shows you an overview of daily activities, by hour. If you switch views, you must refresh to see current data. For example, if you switch from **Past Activities** to **Upcoming Activities**, you must refresh to see current data in the panes.

You can see the **My schedules and subscriptions** option in the Personal menu 👤.

1. In the application bar, click your user name 👤 icon , and then click **My schedules and subscriptions**.

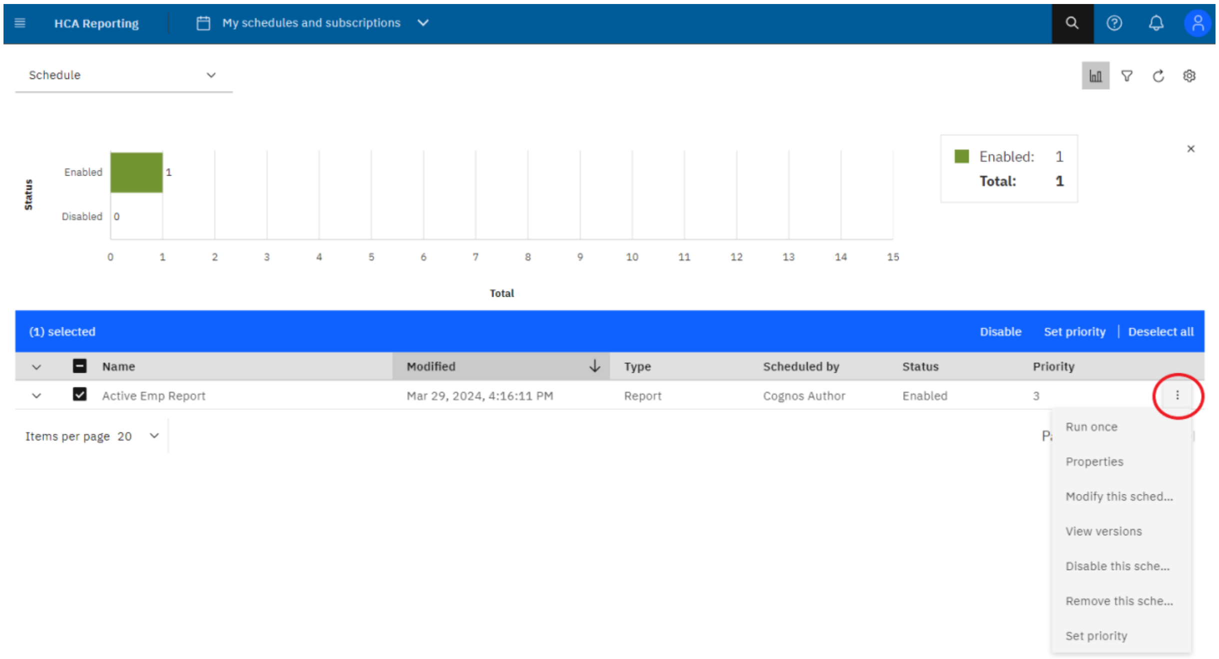


2. To see both a listing and a graph of your schedules and subscriptions for a specific time frame, click the Type button ⌄ and then click **Upcoming**, **Past**, or **Current**.



| | | Name | Modified ↓ | Type | Scheduled by | Status | Priority |
|---|---|---|---|---|---|---|---|
| ⌄ | □ | Active Emp Report | Mar 29, 2024, 4:16:11 PM | Report | Cognos Author | Enabled | 3 |

3. To perform actions on a schedule or subscription in the list, click the More button ⋯ next to the list item, and then click an action.



| Name | Modified | Type | Scheduled by | Status | Priority |
|---|---|---|---|---|---|
| Active Emp Report | Mar 29, 2024, 4:16:11 PM | Report | Cognos Author | Enabled | 3 |

## Renewing credentials used for scheduled reports

If you change your password in Manager Services and need your Cognos schedules to run immediately, you need to renew the credentials Cognos stores to run your schedules. If a schedule tries to run with saved credentials that no longer match the user's current HCM credentials, the schedule will fail.

Cognos automatically renews credentials once per day, but that is not quick enough in all situations. Using the Renew link under **My preferences** will allow a user to update the credentials stored by Cognos for use with schedules on demand.

You can see the **My preferences** option in the Personal menu .

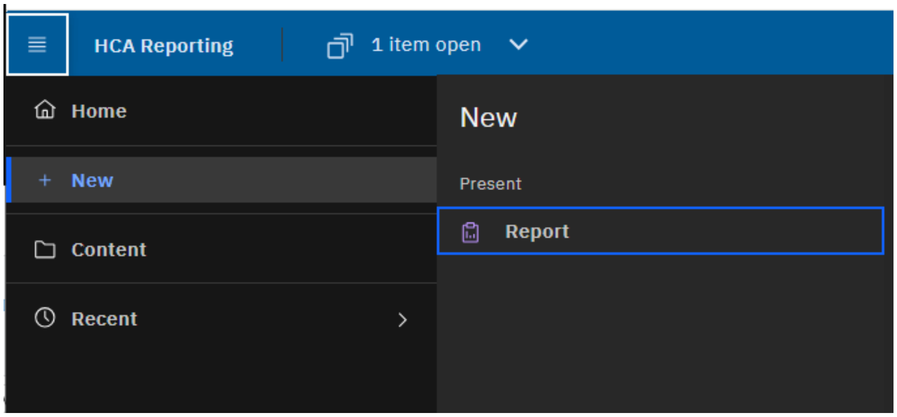
1. In the application bar, click your user name icon , and then click **My preferences**.
2. The "Renew" link is under Personal Menu > My Preferences > Personal Tab > Advanced.

## HCA Author

You must have function security to HCA Author in order to create or edit reports.

## Creating a report

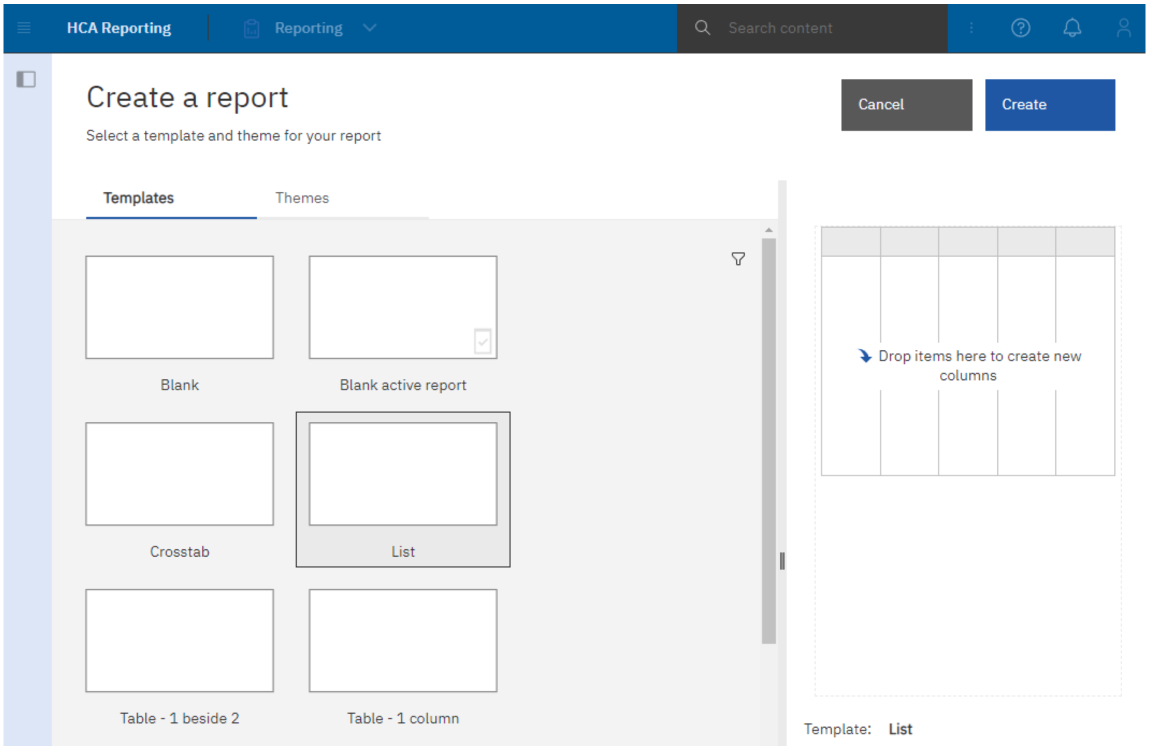
Create a report by clicking the **Open Menu** in the HCA Reporting portal, select **+ New** and then clicking **Report**.



## Templates and themes

IBM Cognos Analytics includes several basic report templates and color themes that you can choose from when you create a new report.

**Hint:** Selecting the “List” template will give you a template similar to prior versions of Cognos.

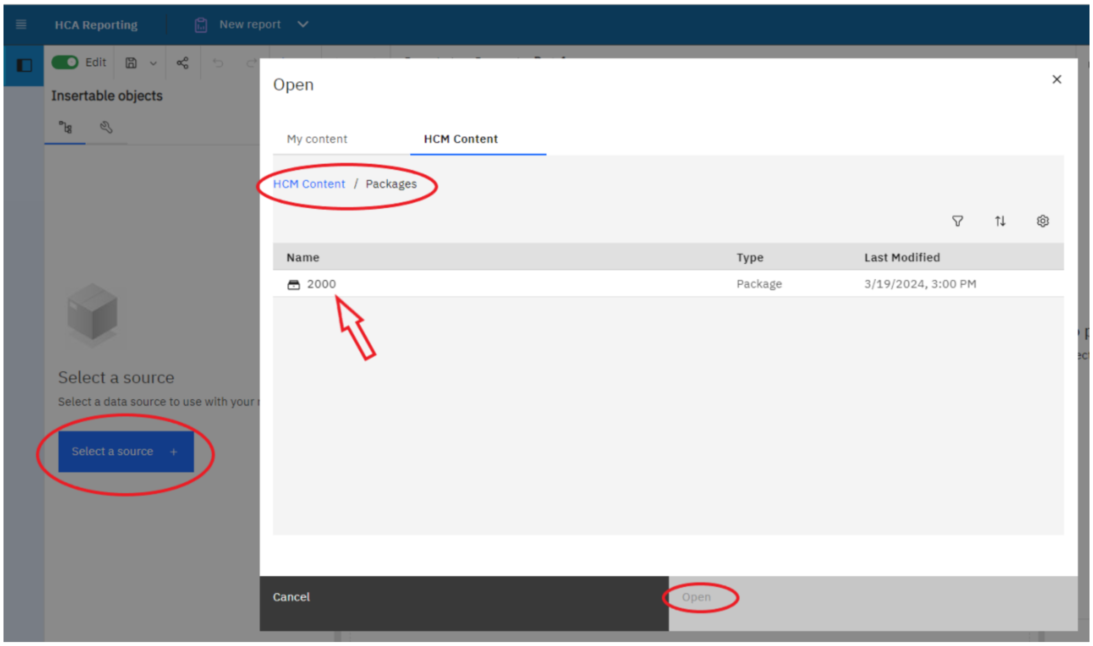


## Selecting a package

Add data to a report by loading packages in IBM Cognos Analytics - Reporting.

1. Click the **Sources** icon .
2. Click the **Add report data** icon ⊕ or the **Select a source** button.
3. Select the package that you want.

Look for a package matching your customer number in the HCM Content > Packages folder. Browse to HCM Content > Packages, select the package, then click **Open**.



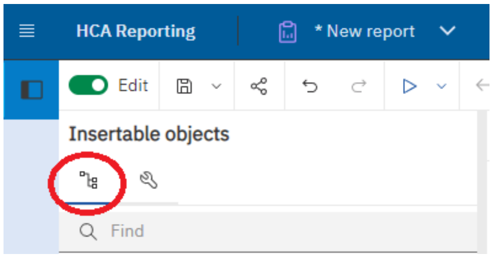
| Name | Type | Last Modified |
|---|---|---|
| 2000 | Package | 3/19/2024, 3:00 PM |

## Insertable objects

### Sources tab

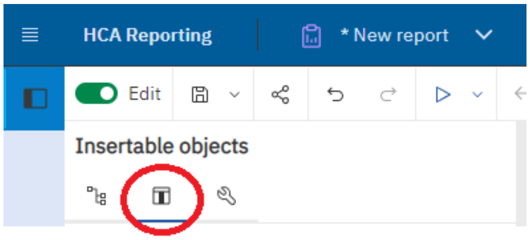
The **Sources** tab contains items from the package selected for the report, such as data items and calculations.

Search through the sources by typing a value in the **Find** field. As you type, the items that match your search string appear in the tab.



### Data Items tab

The **Data Items** tab shows the queries in the report.



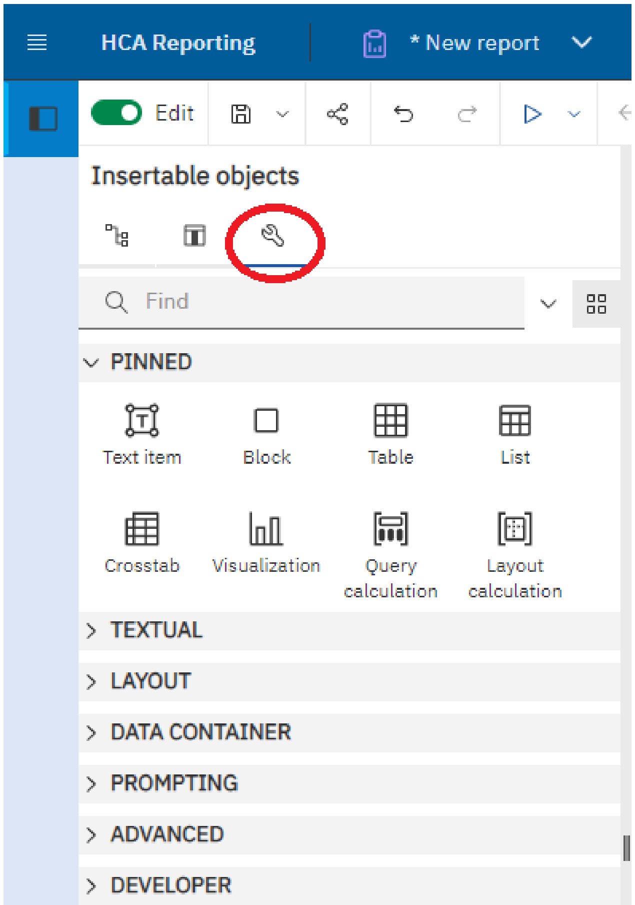
### Toolbox tab

The **Toolbox** tab 🔧 contains various objects that you can add to a report, such as visualizations, text, layout objects, and prompts. You can also add advanced objects such as custom controls and hyperlinks. Depending on the type of object, you can drag it from the Toolbox to the canvas or double-click it to open a window in which you define values for the object. When the object is placed on the canvas, its properties are displayed in the **Properties** pane.

Search through the objects by typing a value in the **Find** field. As you type, the items that match your search string appear in the tab.

Toolbox items are organized into groups. Click › to show all of the items available in each group or ⌄ to collapse the groups. Click ⊞ to toggle between a list view or a tree view of the items in the groups.

Add items that you use often to the **PINNED** group. Items in the **PINNED** group appear in the Add circular menu icon ⊕ on the canvas when you create a new report. To add a toolbox item to the **PINNED** group, right-click the item and click **Add to pinned toolbox items**. To remove an item from the **PINNED** group, right-click the item and click **Remove from pinned toolbox items**.



Tip: You can use a Query Calculation from the toolbox to create a new column in a List report.

## Add relational data to a report

Select the data items to appear in the report.

1. From the **Source** tab ., drag each query item to the report.

   **Note:** A flashing black bar indicates where you can drop an item. Items inserted in the report appear on the **Data Items** tab Data items.

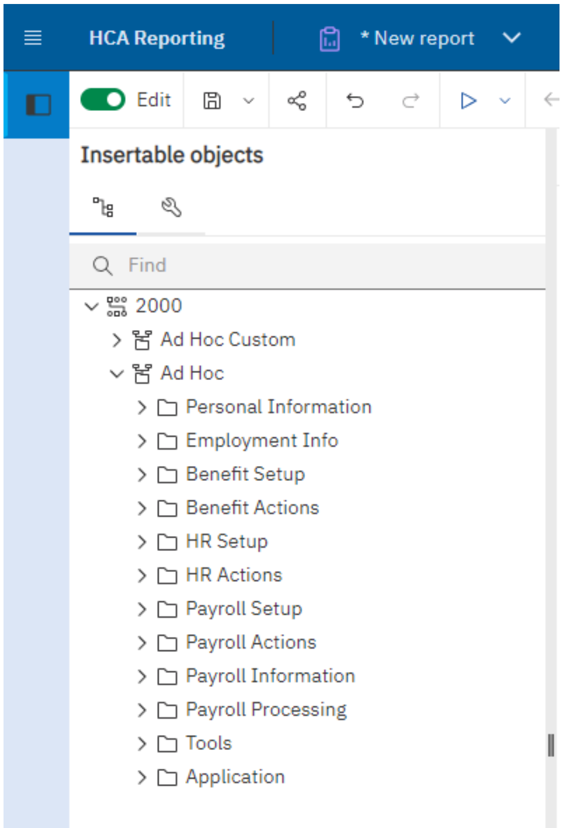
   Other ways to select data items are to double-click each item or to right-click each item and click **Insert**.

2. If you want to remove a data item from the report, select it, and in the report object toolbar, click the **More** icon ••• and then click the **Delete** option 🗑. To remove the data item from the report but keep it on the **Data Items** tab Data items, click **Cut** instead.

3. For more information about a query item, right-click the item in the **Source** tab and click **Properties**. You can also obtain more information by clicking **Lineage**.

## Data organization

Available data is organized to match the functions available in HCM Manager Services. A user will only be able access data for which they have at least read permission to in Manager Services.



Tips:

Key employee data is stored in 2 main query subjects. Fields like the employee's name, organization, job title, birth date, etc. are stored in query subjects that match equivalent functions in Manager Services.

- Ad Hoc>Personal Information>Demographics
- Ad Hoc>Employment Info>Work Profile

Query Subjects with (Current), (Last), (Current Year), etc. appended to the name are pre-filtered versions of the query subjects designed to make it easier for you to find commonly needed information without having to apply your own filters. For example, "Ad Hoc>HR Actions>Employment Status (Current)" is pre-filtered to only the employee's current status where as "Ad Hoc>HR Actions>Employment Status" contains every status an employee has ever had.

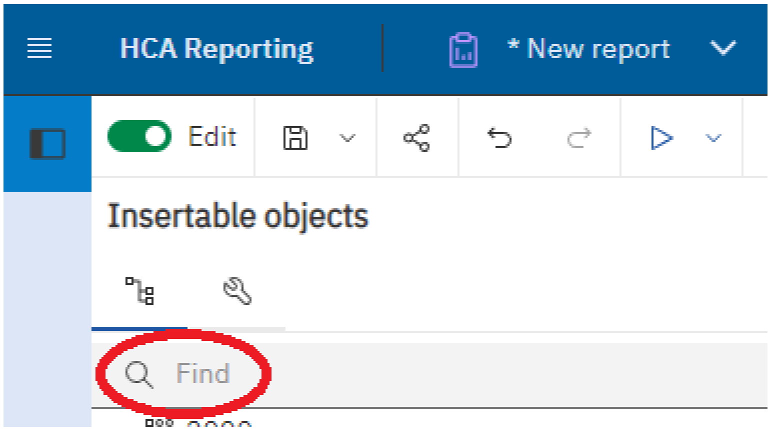
There are 4 different folders for Payment History. These all include the same fields and come from the same source. Each one has a different set of required filters. Choose the one the best fits your needs, whether the report is based on Pay Ending Date or Check Date. For best performance use the query subjects that include a Legal Entity prompt.

Always start Payment History reports with fields from the Check History Master. These query subjects contain the unique "header" record for each check.

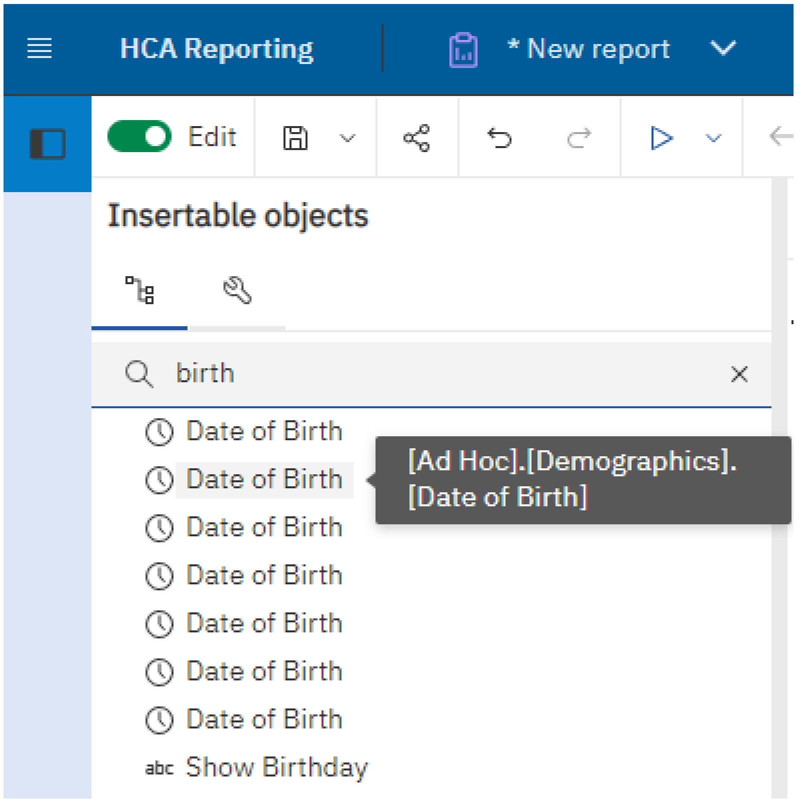
HCM stores accumulated payroll data. If you need payroll numbers across fixed calendar time frames like YTD or Quarter, this data can be retrieved from accumulator query subjects instead of Payment History. Accumulator query subjects store data by Legal Entity and Tax Year. The "Misc Payroll" query subjects are the header records for the accumulators. As a result, an example would be a filter applied for the Tax Year in Misc Payroll would also filter the tax year of any accumulator query subject included on the report. For earnings accumulators, use fields from the Pay Distribution folder and don't include the G/L Account field. The amounts per pay type will automatically be aggregated together for you.

## Searching for fields

You can use the Find area above the package to find fields in the package.



Hover the cursor over the query items returned by the search to view which query subject the field belongs to. It is important to choose the item from the query subject most appropriate for the report.



birth

[Ad Hoc].[Demographics].[Date of Birth]

Tip: If you know where a field is displayed on a screen in Manager Services. Look for a folder in the reporting tool that has the same name as the function in Manager Services. This is where you should find the field in the reporting tool.

## Data Security

Record-level security is based on the security method (Org Level Security, Reports To Security) configured in HCM. This prevents end-users from seeing employee data that they do not have permission to view, including compensation data.

Query subject availability follows Function Security in HCM Manager Services (function security controls what types of data an end-user can add, view, edit and delete for the employees they have access to in HCM). If a given HCM user doesn't have read access to a function in MS, they won't see that type of data in insertable objects for report authoring and won't be able to view that type of data in a report.

If an end-user has "No Salary" permission set in Organization Security, they will not be able to see compensation related data. The Salary field in the following query subjects will be zero in this situation: Salary History, Salary Changes.

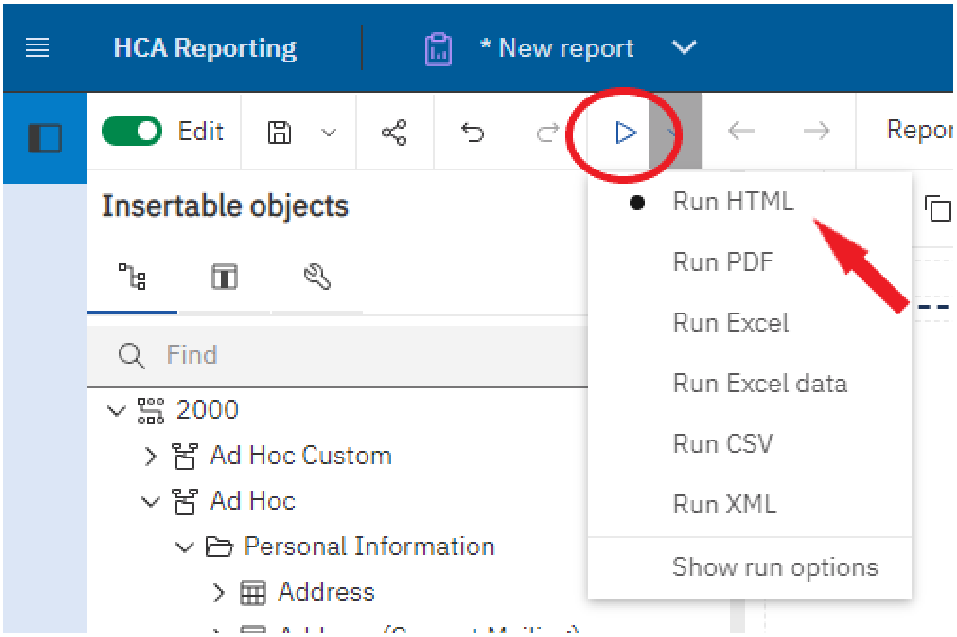
The Social Security Number field will be empty unless "Display Social Security Numbers in Manager Services" is selected in HR System defaults and the user has Function Security to the Demographics function OR "Display Social Security Numbers in Manager Services" is not selected but the user has Function Security Update permission to the Demographics or Misc Payroll functions.

The "Report Security Update" function should be run after any organization security change in Manager Services. This will update the record level permissions used in Ad Hoc reporting. Otherwise, the changes will be applied to report security in a nightly process.

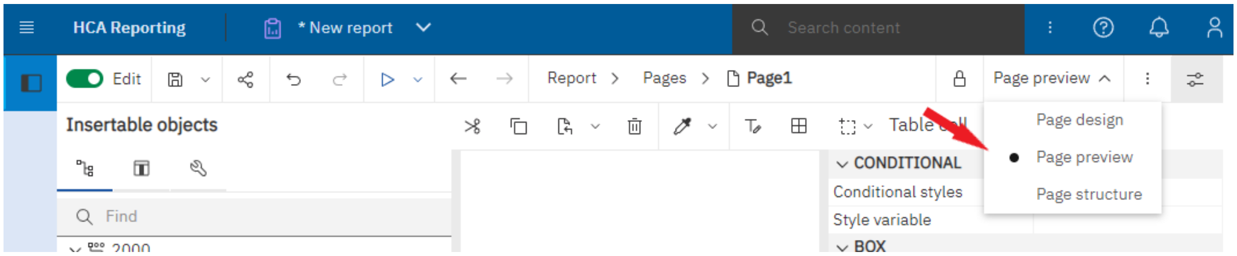
## Viewing results

After adding fields to the report, you can run the report or view a Preview of the results.

In the application bar, click Run options ▷, chose the format to run the report in. The report viewer will run in a separate tab. If the report has prompts, you will be prompted for values.



To see a preview of the report results, change the view from **Page Design** to **Page Preview**.

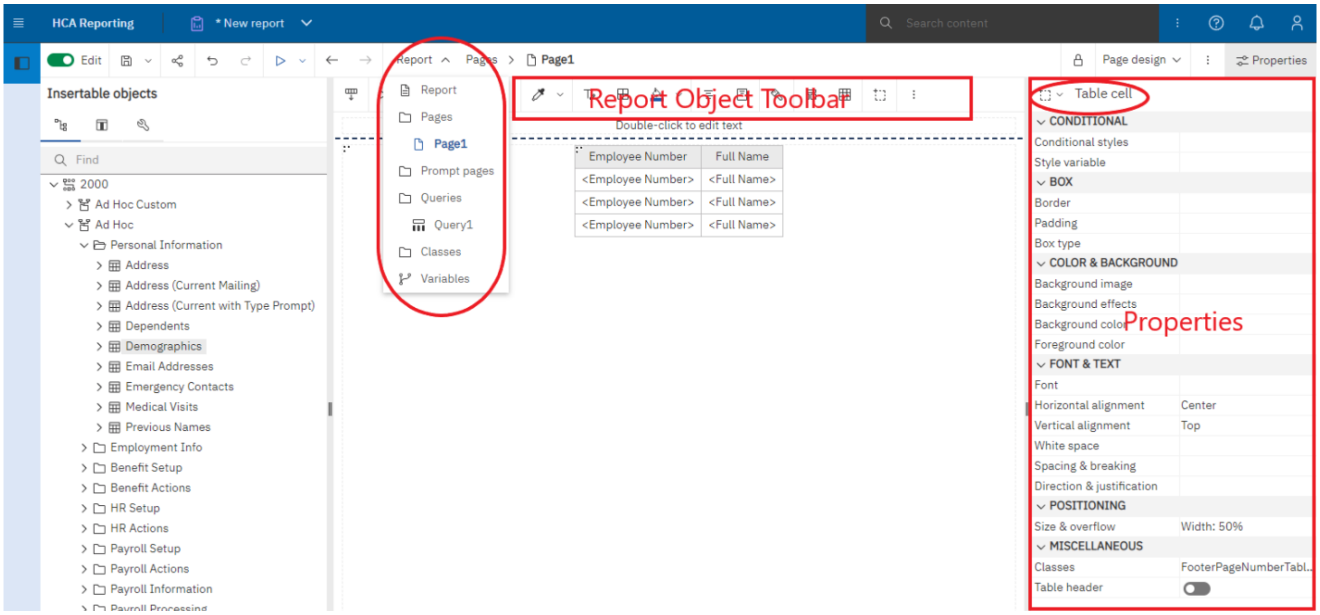


## Report authoring layout

The **Report Object Toolbar** gives access to features such as cut, paste, filtering, sorting, grouping and formatting.

**Properties** for the selected object are shown in the Properties pane. If properties are not visible click the show properties icon . You can also navigate to other object properties by using the object drop down list at the top of the Properties pane.

To navigate between report elements, pull down the **Report** drop down in the upper left. Under this drop down you can navigate to other elements, like the query or queries used in the report.



| Employee Number | Full Name |
|---|---|
| <Employee Number> | <Full Name> |
| <Employee Number> | <Full Name> |
| <Employee Number> | <Full Name> |

| | |
|---|---|
| Horizontal alignment | Center |
| Vertical alignment | Top |
| Size & overflow | Width: 50% |
| Classes | FooterPageNumberTabl... |

## Saving a report

In the application bar, when you view a report, tap either , or . The location of the save option depends on the type of report you view.

You also have the **Save as** option of saving the report version under a different name or in a different location.

Save Locations:

My Content – this folder is visible only to the user. Reports saved here will only be available to the current user.

Customer Shared Folder – this folder is visible to all user for the customer. Save a report here if you wish to share it with other users. This folder will be in this location: Team Content  > HCM Content > Reports > Customer Folder > {*customer number*} > Shared.

Cognos Group folder – this folder is visible to only users defined in a Security Group. Save a report here if you only want a select set of users to access the report. Groups and memberships are defined in the HCA Security function. These folder locations will be: Team Content  > HCM Content > Reports > Customer Folder > {*customer number*} > Groups > {*group name*}.

## Editing an existing report

Navigate to the report in the portal, from the entry context menu ⋯, click **Edit Report.**

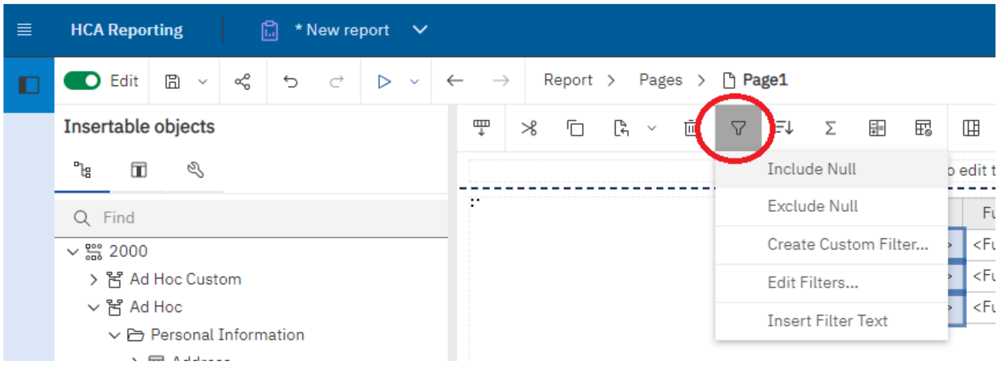
Alternatively, run the report by clicking the report name. Once the report is open choose **Edit**

Edit in the application bar.

## Creating a filter

Use the **Create Custom Filter** option to create a simple filter based on one data item.

On the canvas, select the item in the report that you want to filter on, and in the on-demand toolbar, click the **Filters** icon, and then click **Create Custom Filter**.



For all types of filters, there are some common settings you can define.

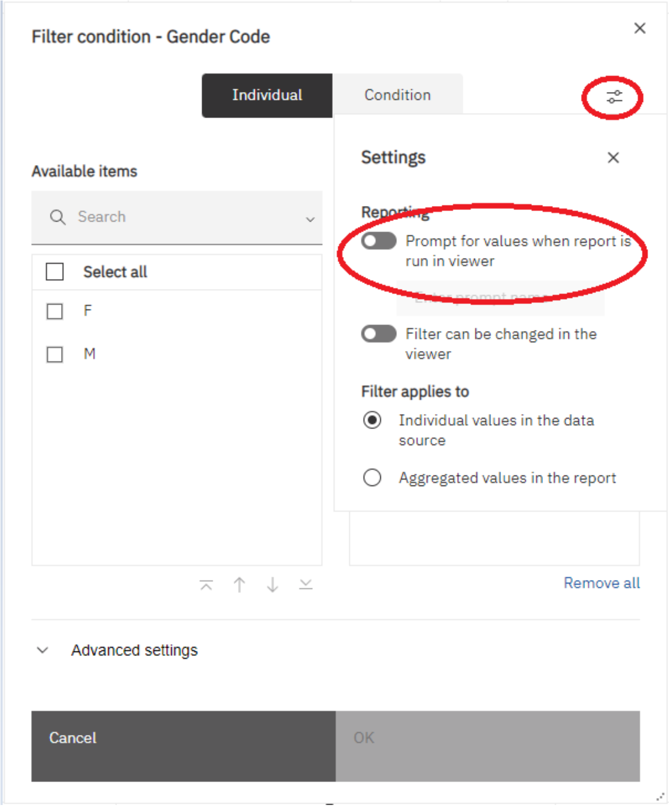
Click **Settings** and then do the following steps:

a. To prompt for filter values when the report is run, select **Prompt for values when report is run in viewer**, and type a name for the prompt.

b. To allow the filter criteria to be changed in the viewer, select **Filter can be changed in the viewer**.

Click **OK** when you're finished defining the filter conditions.

Create a filter with a prompt:



Filter condition - Gender Code

Individual | Condition

Available items

Search

□ Select all

□ F

□ M

Settings

Reporting

Prompt for values when report is run in viewer

Filter can be changed in the viewer

Filter applies to

● Individual values in the data source

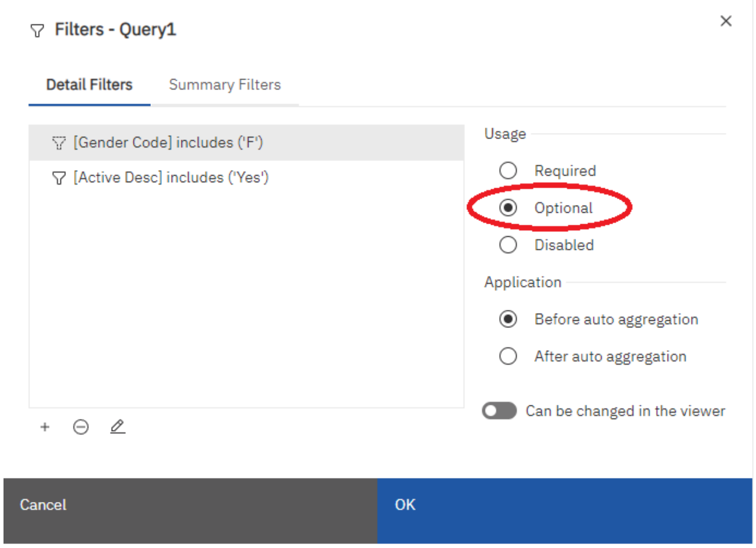
○ Aggregated values in the report

Remove all

Advanced settings

Cancel | OK

To make the use of a prompted filter optional, so that a value is not required to be selected for the report to run, set Usage to Optional on the Edit filters screen.



Filters - Query1

Detail Filters | Summary Filters

[Gender Code] includes ('F')

[Active Desc] includes ('Yes')

Usage

○ Required

● Optional

○ Disabled

Application

● Before auto aggregation

○ After auto aggregation

Can be changed in the viewer

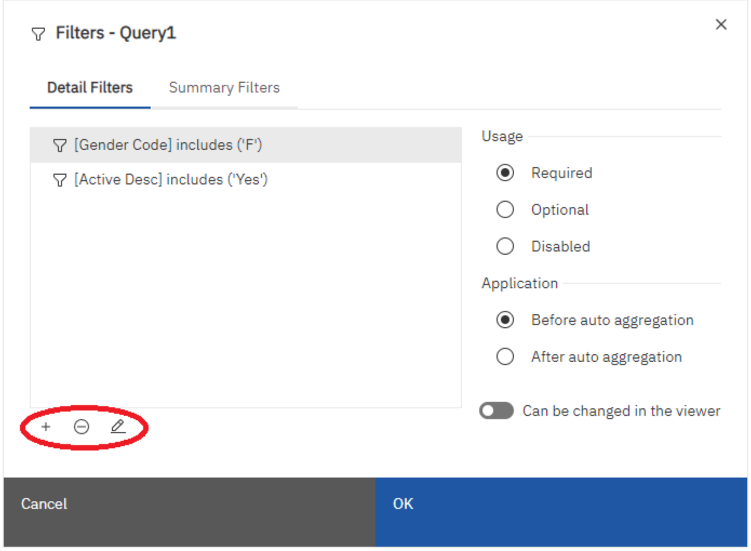
Cancel | OK

## Edit or remove a filter

After you created filters, you can edit or remove them.

1. Click a data container object that contains a filter.
2. To remove all filters from the object, in the report object toolbar, click the **Filters** icon and click **Remove All Filters**.
3. To edit a filter or remove a single filter, in the report object toolbar, click the **Filters** icon and click **Edit Filters**.

   Edit Filters window:



Filters - Query1

Detail Filters | Summary Filters

[Gender Code] includes ('F')

[Active Desc] includes ('Yes')

Usage

● Required

○ Optional

○ Disabled

Application

● Before auto aggregation

○ After auto aggregation

Can be changed in the viewer

⊕ to add a new filter

⊖ to remove the selected filter

✎ to edit the selected filter

Tip: Change Usage to Optional if a selection is not required for a prompted filter.

## Sorting relational data

You can sort items to view them in your preferred order. You can sort items in a list in ascending or descending order based on a value or a label, such as revenue or employee name. You can also perform advanced sorting to sort columns within groups or to sort a row or column using another data item.

1. Click the column or row on which to sort.
2. In the report object toolbar, click the **Sort** icon and click **Ascending** or **Descending**.

   An arrow appears beside the data item to indicate that a sort order was set.

When you specify a sort order for more than one column, the columns are sorted in the order in which they were inserted in the report. For example, you add columns A, B, and C to a report and specify a sort order for each. When you run the report, column A is sorted first, then column B, and then column C. You can change the order in which the columns are sorted in the **Edit Layout Sorting** options.

**Tip:** To remove a sort order, click **Don't Sort**.

## Group relational data

Group data items in a list report to remove duplicate values. For example, you have a report that shows all the products purchased and their product type. You group the Product type column so that each product type cell spans the products purchased cells.

After a column is grouped, you can move it elsewhere in the report.

1. Click the column(s) on which to group.

   You can click either the column title or one of the column cells.

   **Tip:** To perform multiple groupings at once, use Ctrl+click or Shift+click.

2. In the report object toolbar, click **Group/Ungroup** ⊞.

## Add a simple summary

You can add simple summaries to the groups in a report by using the summarize icon in the report object toolbar. This icon provides a subset of the summary functions available in IBM® Cognos® Analytics - Reporting. For list reports, a **Custom** option is also available so that you can add your own summary function in the expression of the data item.

In lists, the summary appears as a footer. If the column to which you added a summary is grouped, group and overall summaries appear. In crosstabs and charts, the summary appears as a node.

To change a summary, select it, click the **Show properties** icon, and in the **Properties** pane, under **Data Item**, click the **Summary** property and choose a different function.

1. Click the column to which to add a summary.
2. In the report object toolbar, click the **Summarize** icon Σ and click a summary type.
3. To change the summary label, do the following:
   - Click the label.
   - Click the **Show properties** icon, and In the **Properties** pane, under **Text Source**, set the **Source type** property to the source type to define the label.

- Set the property under **Source type** to specify the label. This property depends on the source type you chose. For example, if you chose **Data item value** as the source type, set the **Data item value** property to the data item to use to define the label.

## Format data

Format data in a report to improve readability. If you do not set **Data Format** properties here, data is formatted according to the properties set in the model.

Data formats are not applied in delimited text (CSV) and XML report outputs.

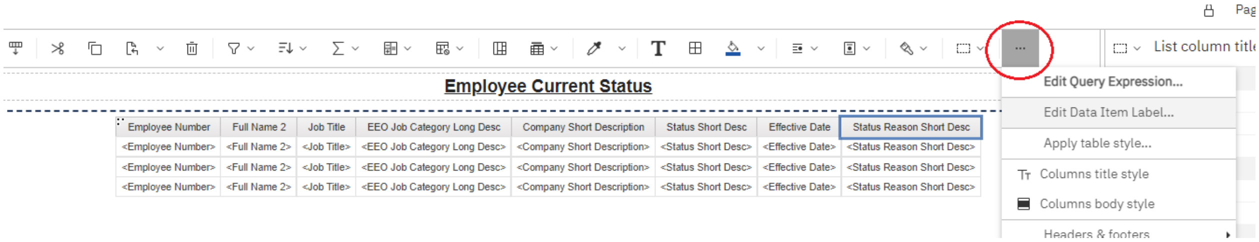
1. Click the object.
2. In the report object toolbar, click the **Data format** icon .

   **Tip:** You can also click the **Show properties** icon , and then double-click the **Data format** property.
3. Under **Format type**, click the format type to apply to the object.
4. To override any of the properties of the format type that were defined for the current layout, in the **Properties** box, click the property and specify its value.

## Edit column header

Select the header to modify then in the report design tool bar, select More (…) then "Edit Data Item Label…". Set the Data Item Label to the desired text.

Tip: The browser window may need to be maximized to see the More button.

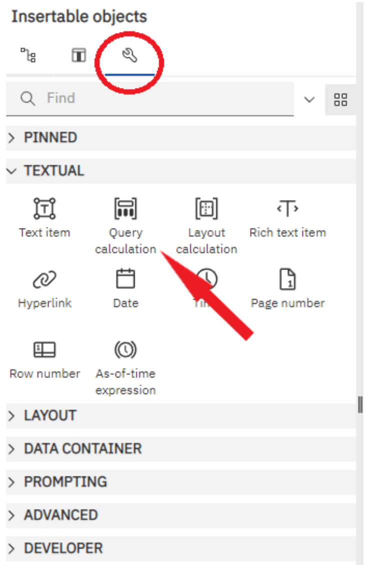


Employee Current Status

| Employee Number | Full Name 2 | Job Title | EEO Job Category Long Desc | Company Short Description | Status Short Desc | Effective Date | Status Reason Short Desc |
|---|---|---|---|---|---|---|---|
| <Employee Number> | <Full Name 2> | <Job Title> | <EEO Job Category Long Desc> | <Company Short Description> | <Status Short Desc> | <Effective Date> | <Status Reason Short Desc> |
| <Employee Number> | <Full Name 2> | <Job Title> | <EEO Job Category Long Desc> | <Company Short Description> | <Status Short Desc> | <Effective Date> | <Status Reason Short Desc> |
| <Employee Number> | <Full Name 2> | <Job Title> | <EEO Job Category Long Desc> | <Company Short Description> | <Status Short Desc> | <Effective Date> | <Status Reason Short Desc> |

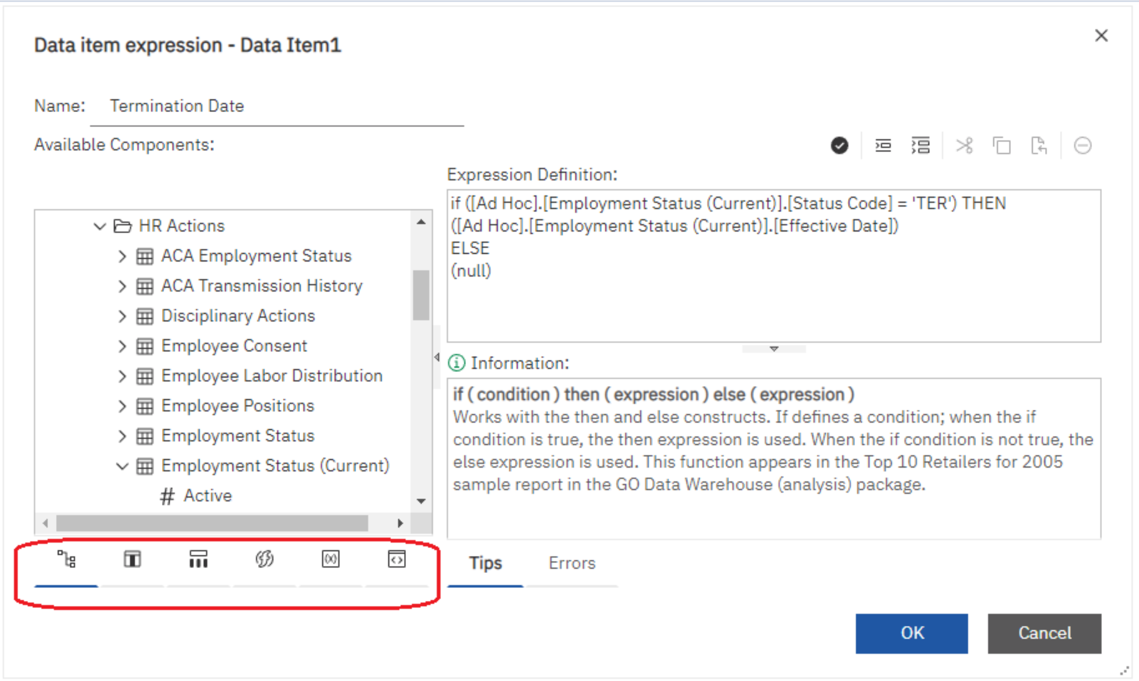
# Calculations

To add a new column to a list report, go to the Toolbox and drag a Query Calculation into the List.



In the Expression Definition box, build the expression using fields from the source package, existing fields from the query or type in your own values. Available options are contained in the tabs below "Available Components". Use single quotes (') to encapsulate string values.

For a list of available functions and constants see the Functions tab. These are also listed in Appendix A .



Data item expression - Data Item1

Name: Termination Date

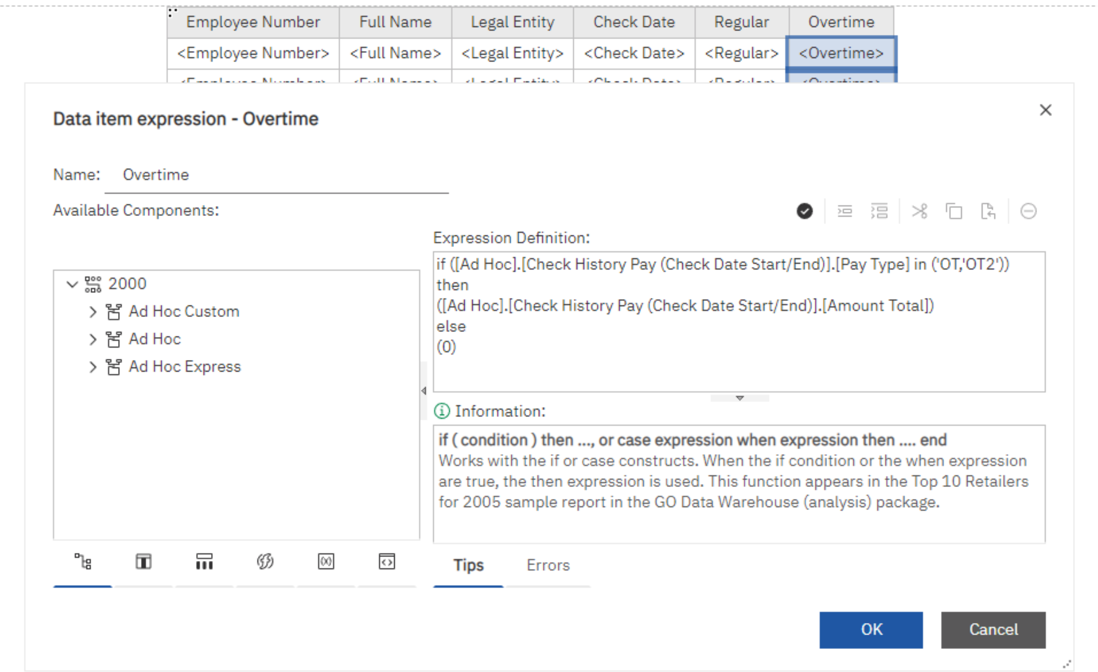
Expression Definition:

```
if ([Ad Hoc].[Employment Status (Current)].[Status Code] = 'TER') THEN
([Ad Hoc].[Employment Status (Current)].[Effective Date])
ELSE
(null)
```

Information:

**if ( condition ) then ( expression ) else ( expression )**
Works with the then and else constructs. If defines a condition; when the if condition is true, the then expression is used. When the if condition is not true, the else expression is used. This function appears in the Top 10 Retailers for 2005 sample report in the GO Data Warehouse (analysis) package.

Tip: You can use if..then statements to display data in rows instead of columns. This is an example of creating an Overtime column.



| Employee Number | Full Name | Legal Entity | Check Date | Regular | Overtime |
|---|---|---|---|---|---|
| <Employee Number> | <Full Name> | <Legal Entity> | <Check Date> | <Regular> | <Overtime> |

Data item expression - Overtime

Name: Overtime

Available Components:

Expression Definition:

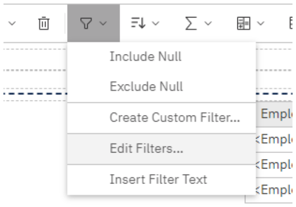
```
if ([Ad Hoc].[Check History Pay (Check Date Start/End)].[Pay Type] in ('OT','OT2'))
then
([Ad Hoc].[Check History Pay (Check Date Start/End)].[Amount Total])
else
(0)
```

Information:

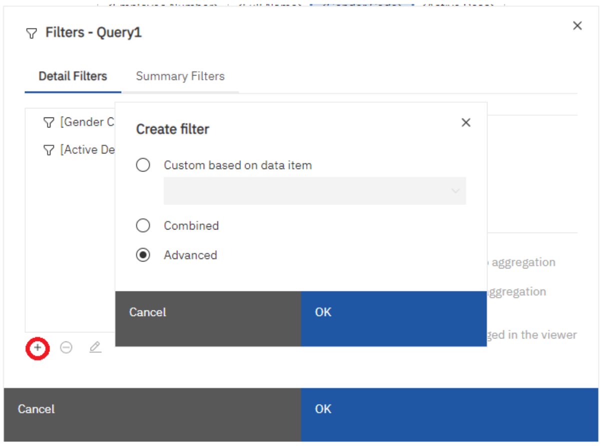
**if ( condition ) then ..., or case expression when expression then .... end**
Works with the if or case constructs. When the if condition or the when expression are true, the then expression is used. This function appears in the Top 10 Retailers for 2005 sample report in the GO Data Warehouse (analysis) package.

## Advanced filtering

There are multiple ways to add advanced filters to a report. One option is to use the Filters menu and selecting “Edit Filters…”.



On the filters screen, add a filter by clicking the “+” button. You can then choose Combined or Advanced.



Filters - Query1

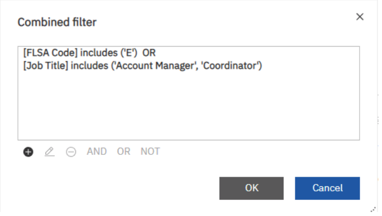
Create filter

○ Custom based on data item

○ Combined

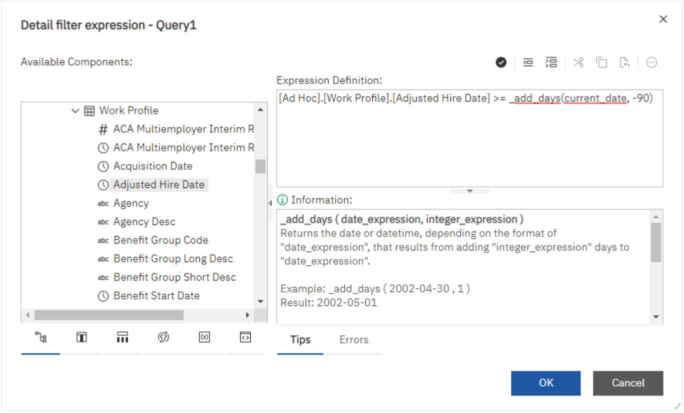
● Advanced

With Combined, you can use a wizard like interface to combine multiple filters with AND/OR statements.


Combined filter

[FLSA Code] includes ('E') OR
[Job Title] includes ('Account Manager', 'Coordinator')

With Advanced, you will get an empty Expression Definition where you can build your own filter statement. For a list of available functions and constants see the Functions tab. These are also listed in [Appendix A](#) . You can AND/OR to combine multiple statements in one filter.

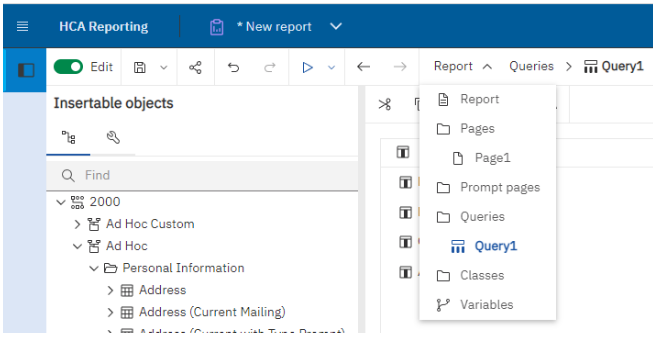

Detail filter expression - Query1

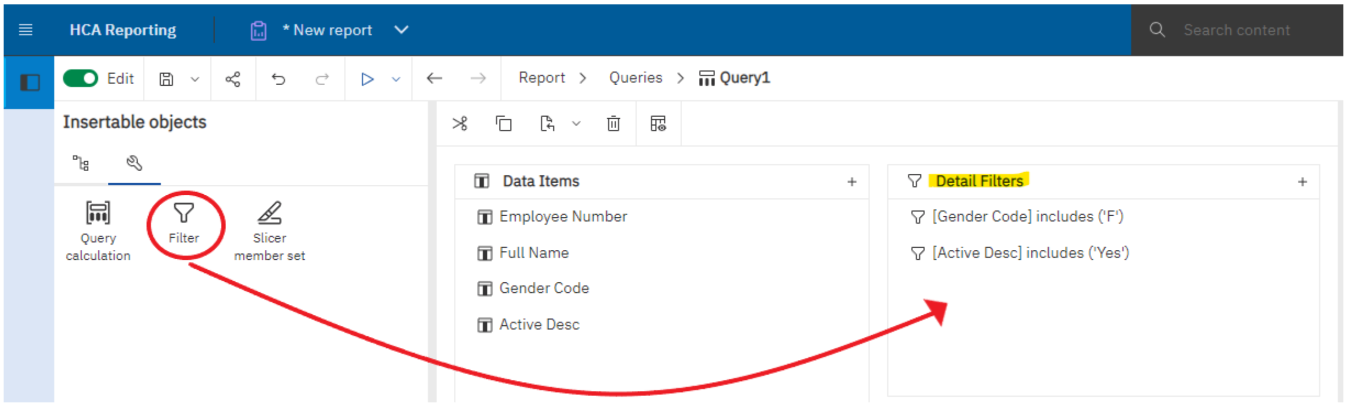
Expression Definition:
[Ad Hoc].[Work Profile].[Adjusted Hire Date] >= _add_days(current_date, -90)

Information:
_add_days ( date_expression, integer_expression )
Returns the date or datetime, depending on the format of "date_expression", that results from adding "integer_expression" days to "date_expression".

Example: _add_days ( 2002-04-30 , 1 )
Result: 2002-05-01

You can also create an advanced filter by navigating to the Query and dragging a Filter object from the Toolbox the Detail Filters area. You can then create your own filter in the Expression Definition window.

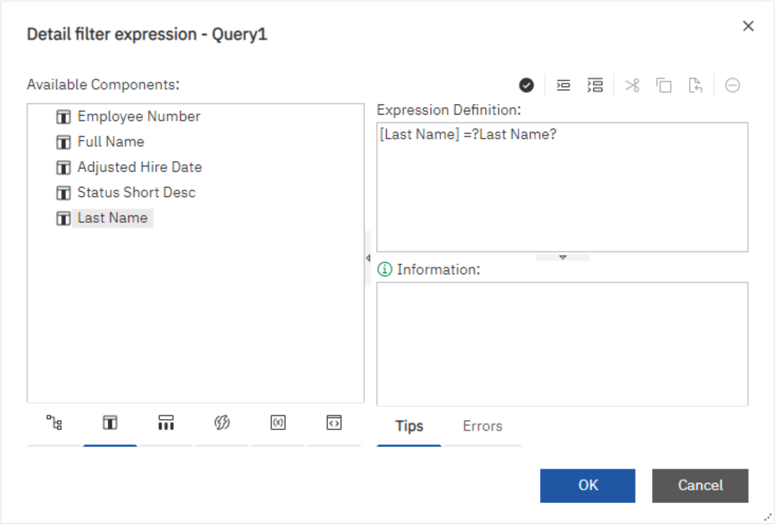



Detail Filters
[Gender Code] includes ('F')
[Active Desc] includes ('Yes')

## Advanced prompting

In addition to the standard prompting available on filters, there are other ways to generate prompts for an advanced filter.

For basic text prompts, values within question marks (?) will be prompted. For example:


Detail filter expression - Query1

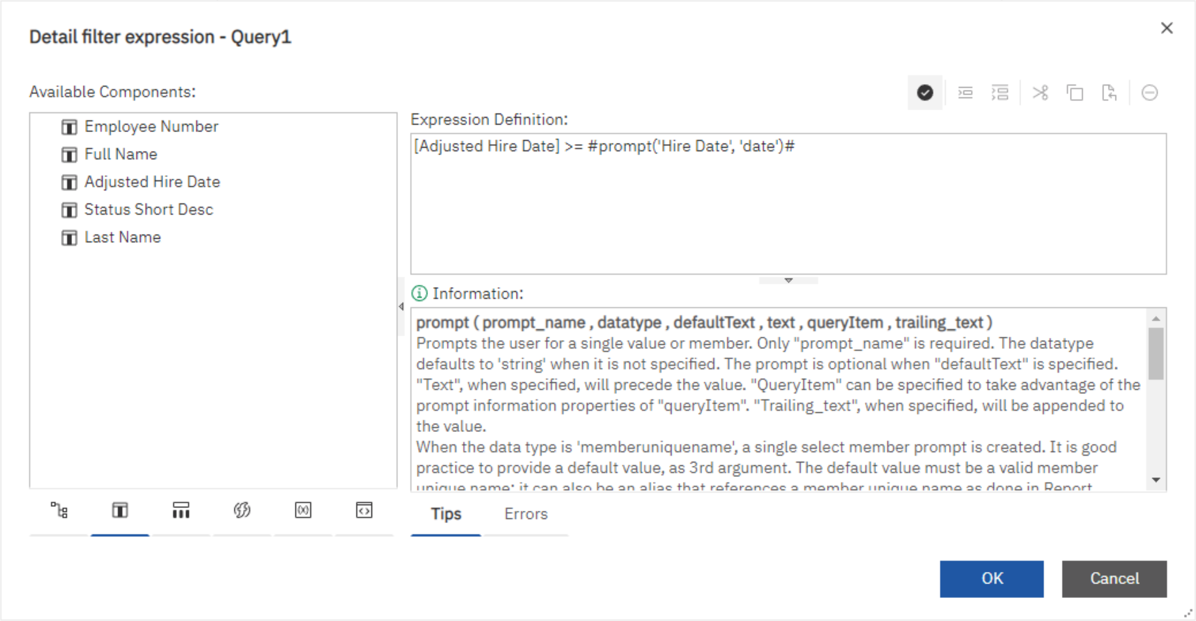
Expression Definition:

```
[Last Name] =?Last Name?
```

For more advanced custom prompts, the prompt function can be used. This allows for the definition of the prompt text and other elements of a prompt, like specifying it is a date prompt. The simple syntax is #prompt('{text}','{data type}')#.

Examples:

- o #prompt('Enter a Year', 'integer')#
- o #prompt('Age on Date', 'date')#


Detail filter expression - Query1

Expression Definition:

```
[Adjusted Hire Date] >= #prompt('Hire Date', 'date')#
```

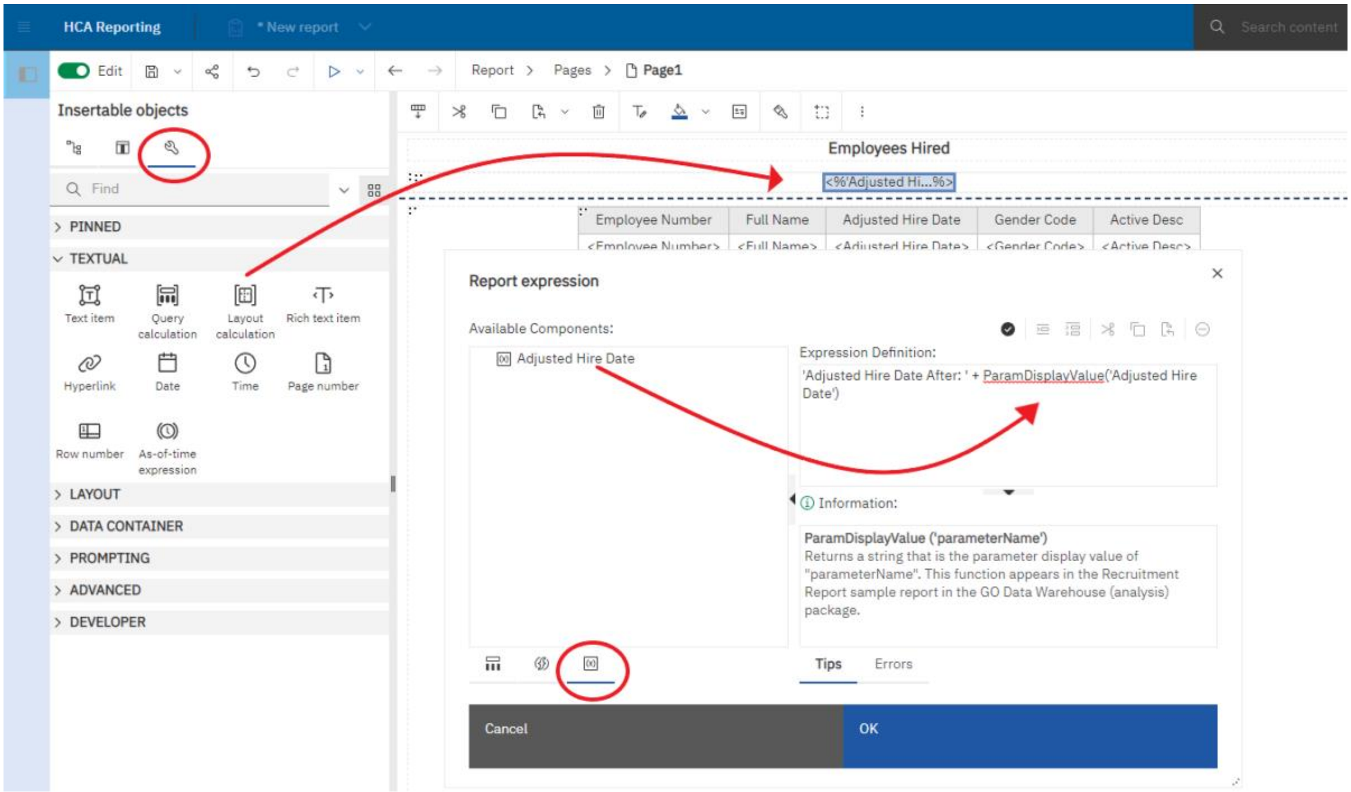
Information:

**prompt ( prompt_name , datatype , defaultText , text , queryItem , trailing_text )**
Prompts the user for a single value or member. Only "prompt_name" is required. The datatype defaults to 'string' when it is not specified. The prompt is optional when "defaultText" is specified. "Text", when specified, will precede the value. "QueryItem" can be specified to take advantage of the prompt information properties of "queryItem". "Trailing_text", when specified, will be appended to the value.
When the data type is 'memberuniquename', a single select member prompt is created. It is good practice to provide a default value, as 3rd argument. The default value must be a valid member unique name; it can also be an alias that references a member unique name as done in Report

## Prompt values in report header

If you have prompt values you wish to display in the report output, one easy way to do this is to add a Layout Calculation to the header. From Toolbox drag a Layout Calculation into the report header. Go to the parameters tab of the Layout calculation window. Double click on the parameter you want to add to the header. It will be added to the expression with the ParamDisplayValue function already included. If you want text before the parameter value you can add it before the ParamDisplayValue function. Use single quotes and + to concatenate the text to before the prompt value.

Tip: You can also use a Table object from the Toolbox if you want to organize displaying multiple prompt values.

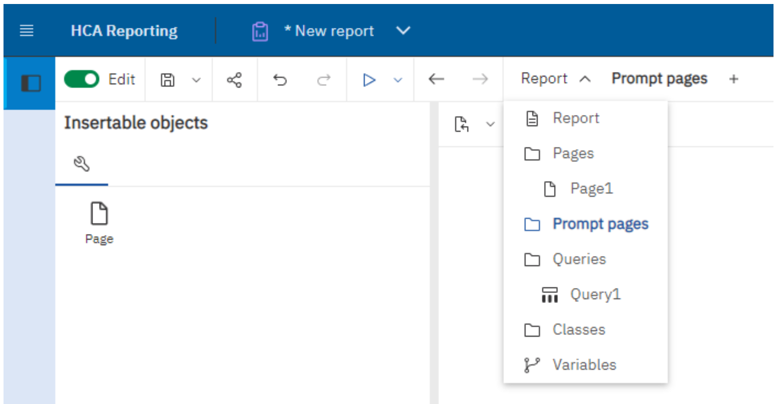


'Adjusted Hire Date After: ' + ParamDisplayValue('Adjusted Hire Date')

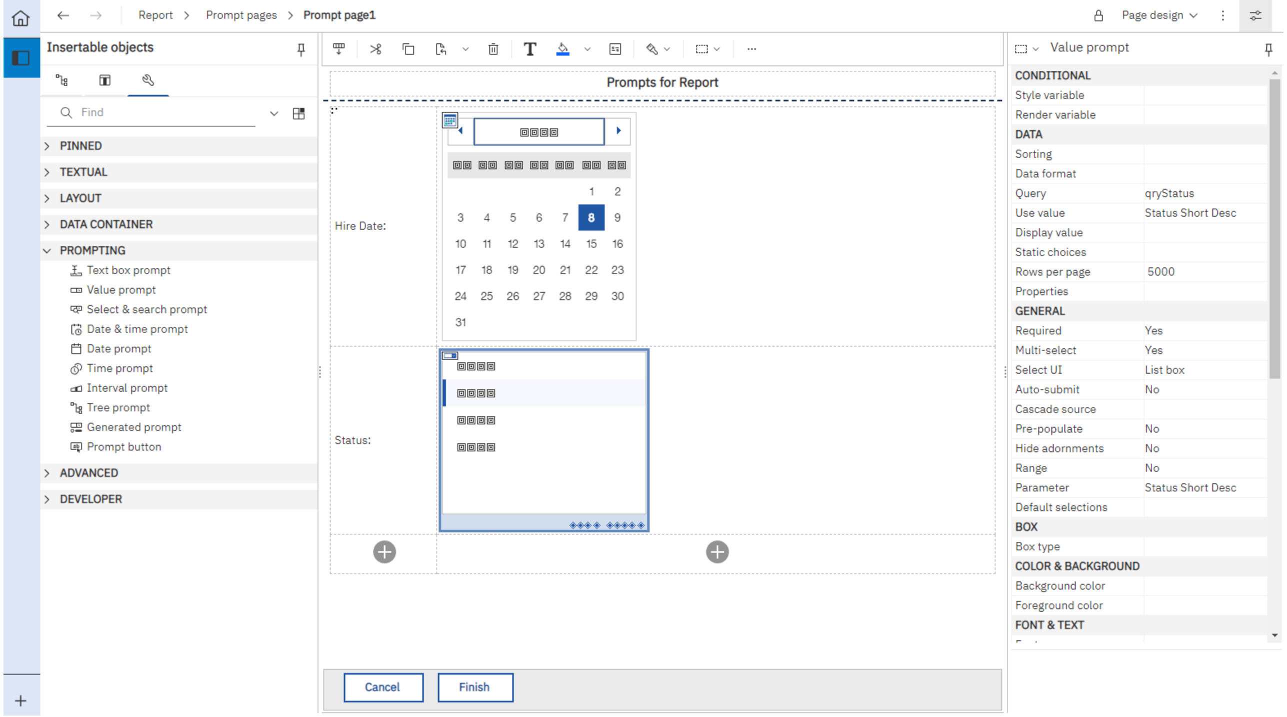
ParamDisplayValue ('parameterName')
Returns a string that is the parameter display value of "parameterName". This function appears in the Recruitment Report sample report in the GO Data Warehouse (analysis) package.

## Prompt Pages

For full control over the look of prompts, the prompts can be presented via a prompt page. To add a prompt page, navigate to “Prompt pages” under the Report pulldown and drag in a Page from the Insertable objects or click the + icon next to Prompt pages.



Use Table, Text items & Prompt objects to build out the page.

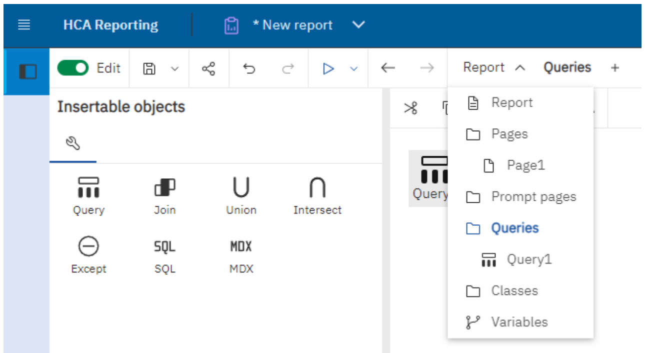


| | |
|---|---|
| CONDITIONAL | |
| Style variable | |
| Render variable | |
| DATA | |
| Sorting | |
| Data format | |
| Query | qryStatus |
| Use value | Status Short Desc |
| Display value | |
| Static choices | |
| Rows per page | 5000 |
| Properties | |
| GENERAL | |
| Required | Yes |
| Multi-select | Yes |
| Select UI | List box |
| Auto-submit | No |
| Cascade source | |
| Pre-populate | No |
| Hide adornments | No |
| Range | No |
| Parameter | Status Short Desc |
| Default selections | |
| BOX | |
| Box type | |
| COLOR & BACKGROUND | |
| Background color | |
| Foreground color | |
| FONT & TEXT | |

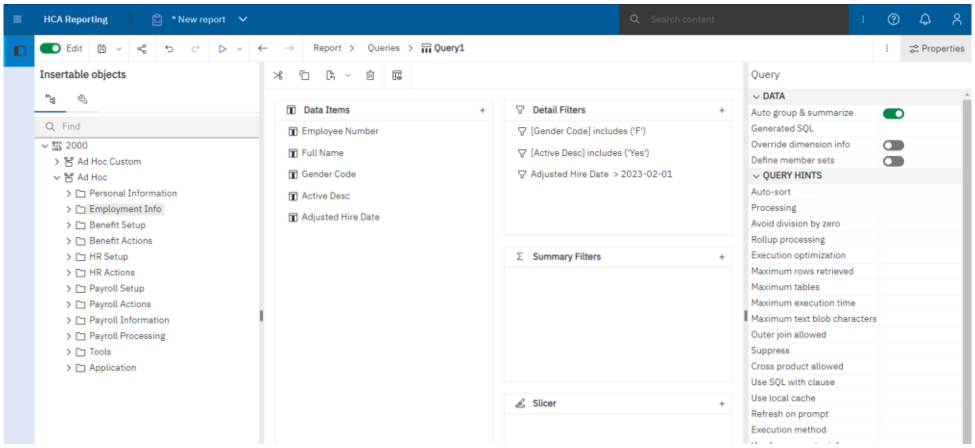
The prompt page will appear before the report runs.

## Editing queries & multiple queries

To access the query or queries for a report, navigate to Queries under the Report pulldown.

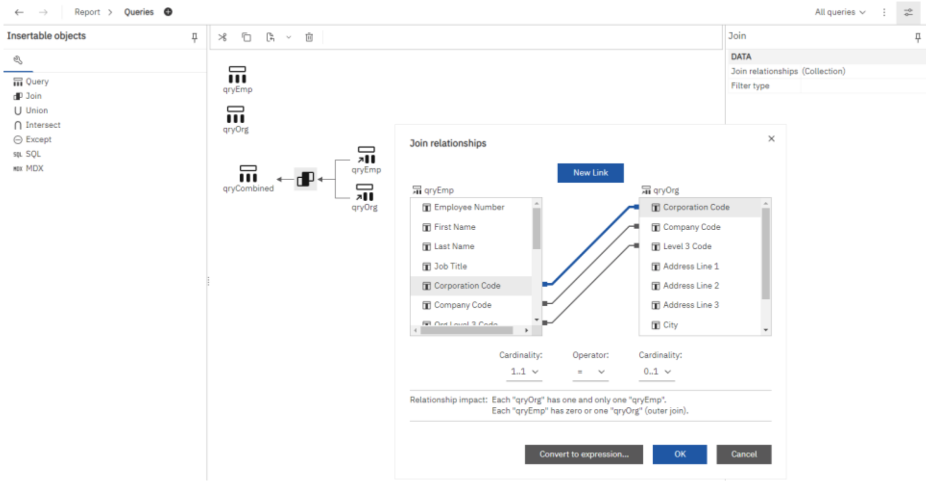


Fields can be added or removed from the query and advanced filters added or removed.



[Gender Code] includes ('F')
[Active Desc] includes ('Yes')
Adjusted Hire Date > 2023-02-01

In some cases, a report may require data to be related in ways not defined or supported in the underlying meta data model. Cognos allows a user to define sets of data via queries and then define the relationship of these data sets with custom joins. A user must understand relational database concepts in order to create and define these join relationships. Navigate to Queries under the Report pulldown to add & edit queries and joins.

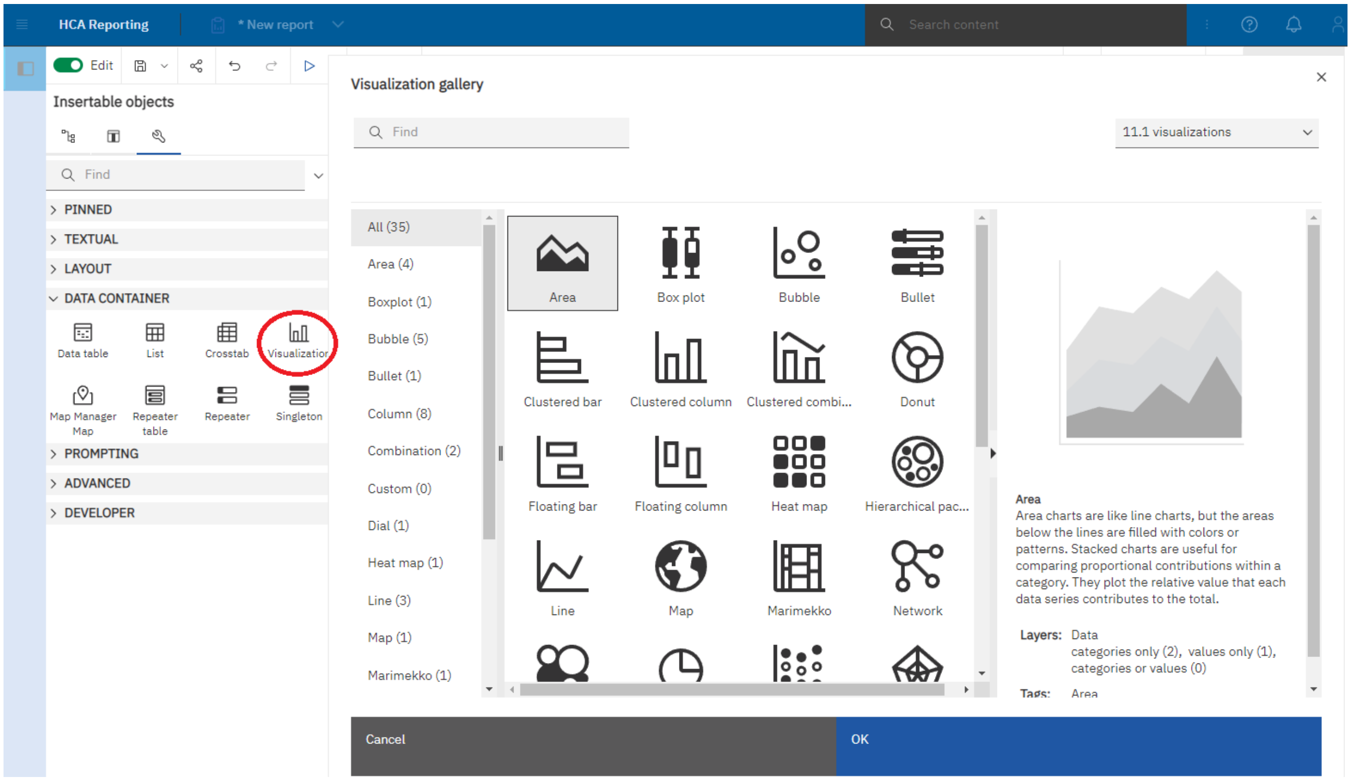


Cardinality: 1..1
Operator: =
Cardinality: 0..1

Relationship impact: Each "qryOrg" has one and only one "qryEmp".
Each "qryEmp" has zero or one "qryOrg" (outer join).

## Charts

Charts and graphs can also be included in Ad Hoc reports. Add a Visualization from the Toolbox in order to add a chart to a report.



Area

Area charts are like line charts, but the areas below the lines are filled with colors or patterns. Stacked charts are useful for comparing proportional contributions within a category. They plot the relative value that each data series contributes to the total.

Layers: Data categories only (2), values only (1), categories or values (0)

Tags: Area

## Set a report as your Manager Services home page

To set a report as your home page in Manager Services, do the following steps:

- Create a folder named Homepage in your Shared and/or My Content folders in Cognos.
- Move or copy the desired report into this Homepage folder.
- Go to the My Account screen in Manager Services and under the Home Page drop down, select the report. Reports in the Homepage folders will be able to be selected in the Home Page drop down list on My Account.

## HCA Security

Use the HCA Security function in Manager Services to create secured group folders to share reports with just selected users. When an HCA Security group is created, a folder by that name is created in HCM Content > Reports > Customer Folders > {customer number} > Groups > {group name}. Only users selected under “users” in the HCA Security function for that group will have access to this folder. Copy or move and any report you want to share with the group to this folder.

Any user with function security to HCA Security is considered and HCA Security Admin and can see the contents of any of the customer’s group folders.

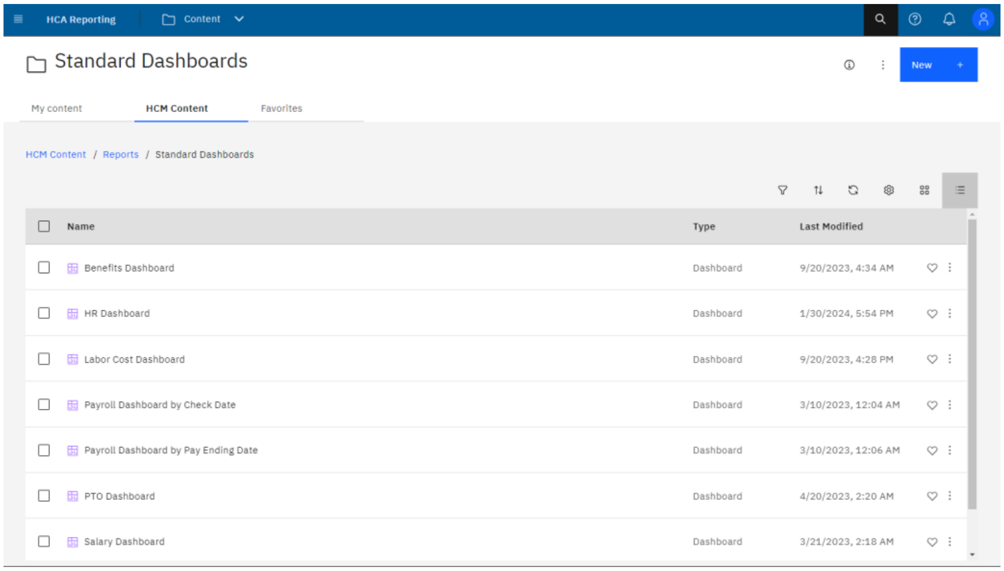
# HCA Dashboards

Human Capital Analytics (HCA) Dashboards provide an interactive means of monitoring, measuring, and analyzing key Benefit, HR and Payroll information that is stored and managed in the HCM application. The information is displayed in an interactive, intuitive, and visual way within the HCM Manager Services application. HCA Dashboards will help provide insights to tackle critical human capital challenges.

A library of Standard Dashboards is provided within the HCA Reporting tool. This provides a quick and easy way for users to see visualizations of their HCM data. The user will only need to click on the dashboard name to run it. Additionally, a dashboard can be set as the user’s Manager Services home page so that relevant data can be easily viewed when entering the Manager Services application.

## Available dashboards

You can find dashboards in HCA Reporting under HCM Content > Reports > Standard Dashboards.



Standard Dashboards

HCM Content / Reports / Standard Dashboards

| Name | Type | Last Modified |
|---|---|---|
| Benefits Dashboard | Dashboard | 9/20/2023, 4:34 AM |
| HR Dashboard | Dashboard | 1/30/2024, 5:54 PM |
| Labor Cost Dashboard | Dashboard | 9/20/2023, 4:28 PM |
| Payroll Dashboard by Check Date | Dashboard | 3/10/2023, 12:04 AM |
| Payroll Dashboard by Pay Ending Date | Dashboard | 3/10/2023, 12:06 AM |
| PTO Dashboard | Dashboard | 4/20/2023, 2:20 AM |
| Salary Dashboard | Dashboard | 3/21/2023, 2:18 AM |

## Dashboard security

Similar to Ad Hoc reports written using HCA Reporting, HCM organization security is applied to all HCA Dashboards. A user will only be able to see counts of employees on dashboards that correlate with the employees that they can view in Manager Services. For example, if a user can only view 10 employees in Manager Services then the visualizations in the dashboards will only represent the data for those 10 employees.

Salary Security is enforced. If a user has ‘No Salary’ access to an employee, then that employee’s salary fields will be 0 in the detail reports and their data will not be included on Salary related visualizations on dashboards.

Function security is enforced on the dashboard level. A user must have function security to Work Profile to see the HR Dashboard. A user must have function security to Benefit Enrollment to see the Benefit Dashboard. A user must have function security to Payment History to see the Payroll Dashboards. A user must have function security to Salary History or Salary Change to see the Salary Dashboards.

## Dashboard data

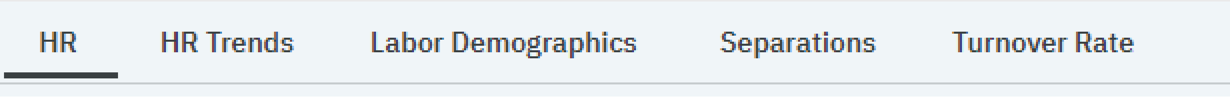
Dashboard data is retrieved from a data warehouse that is refreshed nightly. Same day transactions will not be displayed and should not make significant changes to most visualizations.

Dashboard data will also not be available for Test environments.

## How to interact with a dashboard

To run a dashboard, simply navigate to a dashboard in HCA Reporting and click on the dashboard name.

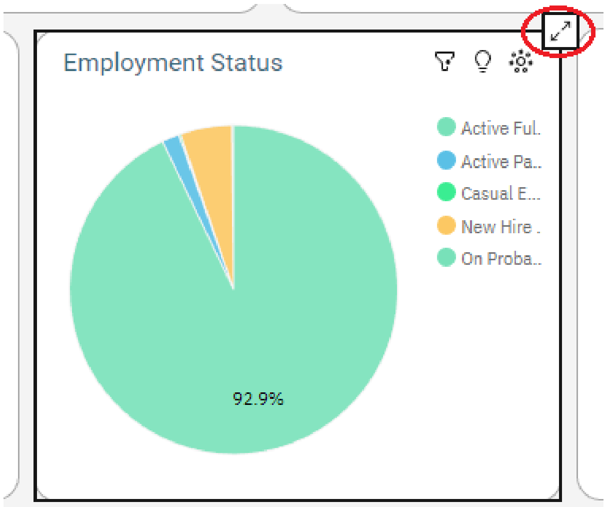
Each dashboard is a collection of visualizations. Related visualizations may be organized by tabs. A dashboard may also be organized into multiple tabs. For the HR Dashboard, you will see 5 tabs. Click on each tab to see different sets of visualizations.



## Expand visualizations

To see a larger version of a single visualization, click on the visualization to select it. Then click the expand icon.

Click the Collapse icon to return to the original size.

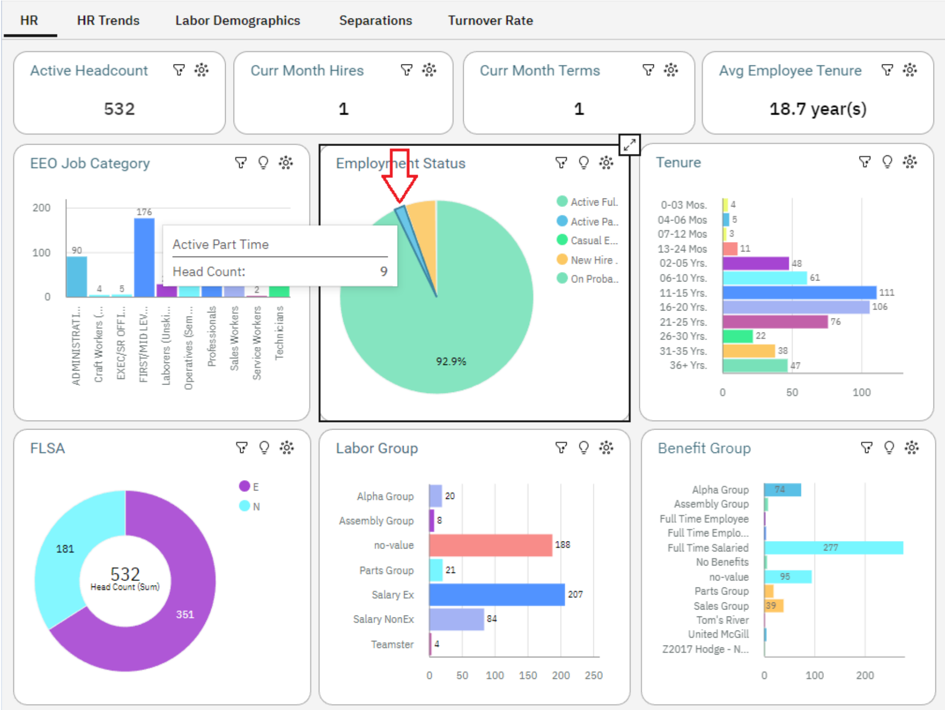


## Selecting data

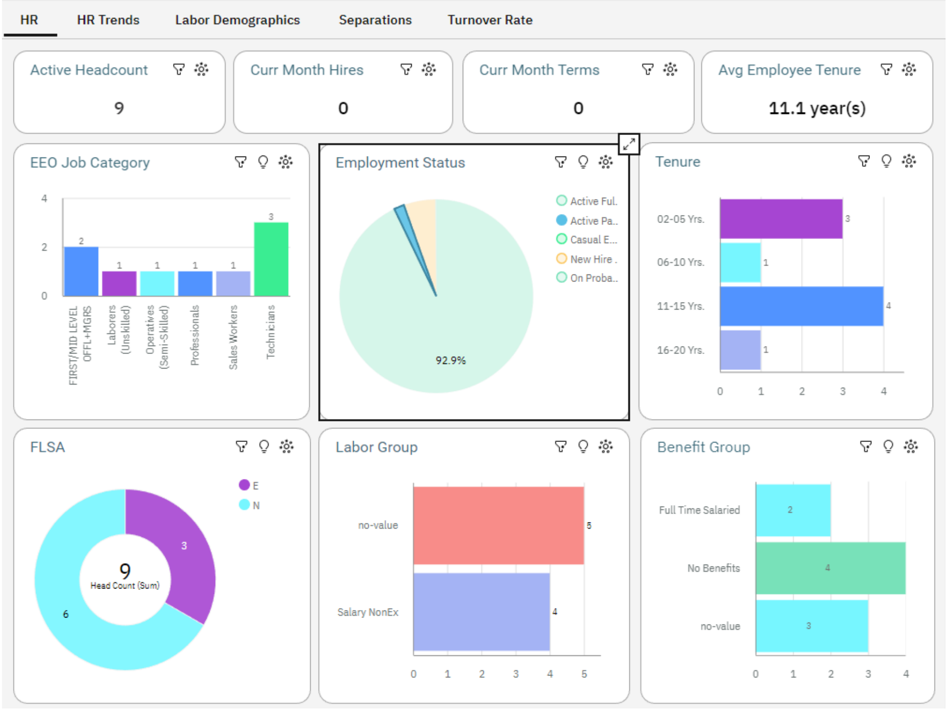
Selecting data in a visualization will update all other visualizations to be filtered by the element clicked.

For example, clicking the “Active Part Time” slice of the Employment Status Pie will cause the other visualizations to update to represent just the Active Part Time employees.

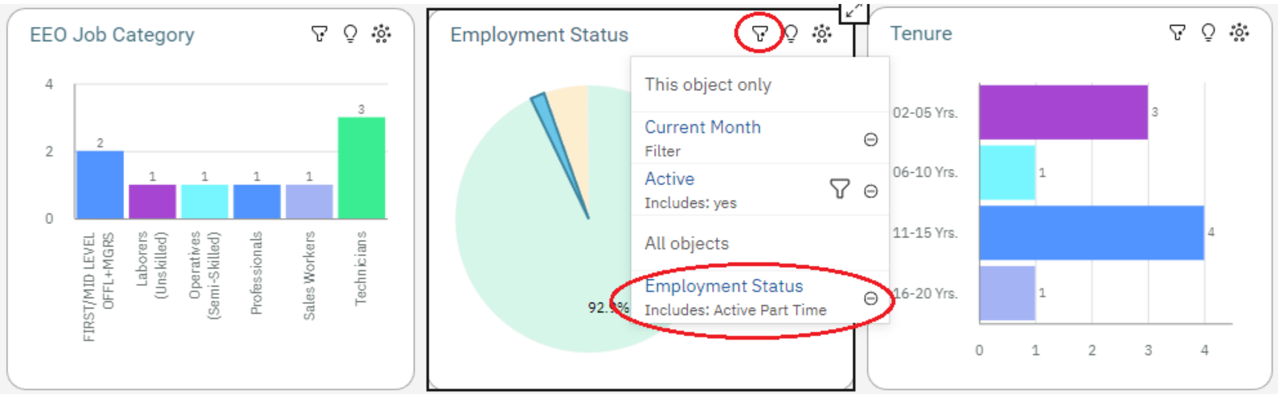
Before:



After clicking on the Active Part Time slice. Notice the other visualizations have updated to just show data for the 9 APT employees in this example.



By clicking the Filter icon on the visualization, you can see “Active Part Time” has been applied to all objects.



This object only

Current Month
Filter

Active
Includes: yes

All objects

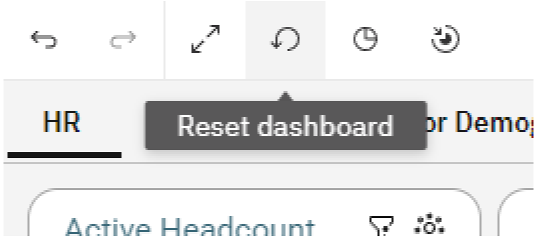
Employment Status
Includes: Active Part Time

## Resetting the dashboard

To get back to the full data in visualizations:

- click on the white space of a visualization
- click x for any filters under All objects (as seen above)
- click the “Reset dashboard” on the top left of the dashboard.

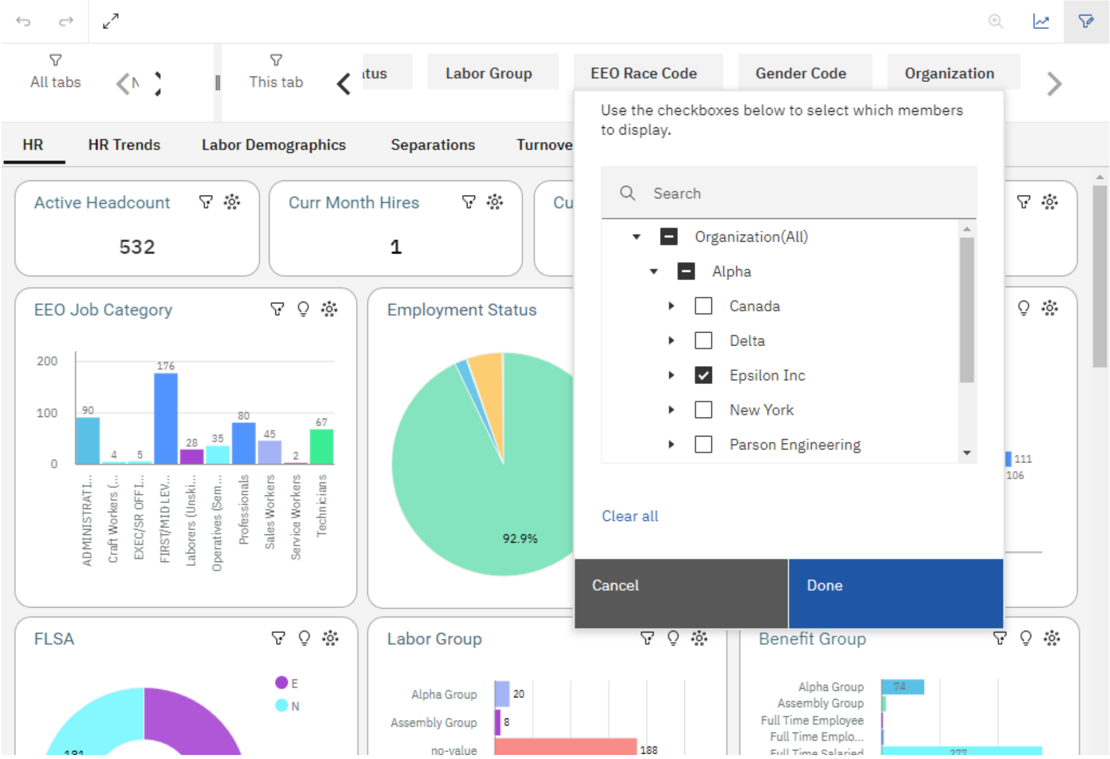
Reset dashboard:



## Applying other filters

Additional filters maybe applied if available in filter header.

For example, the Organizations filter maybe used to limit data to just certain organization levels.



Use the checkboxes below to select which members to display.

Search

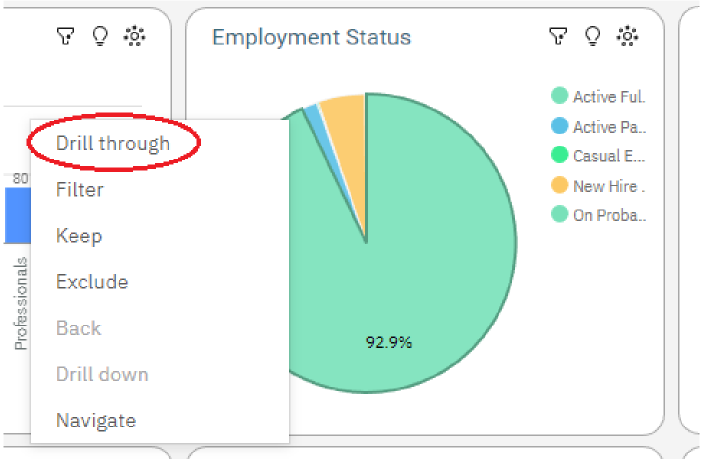
- ▾ ■ Organization(All)
  - ▾ ■ Alpha
    - ▸ □ Canada
    - ▸ □ Delta
    - ▸ ☑ Epsilon Inc
    - ▸ □ New York
    - ▸ □ Parson Engineering

Clear all

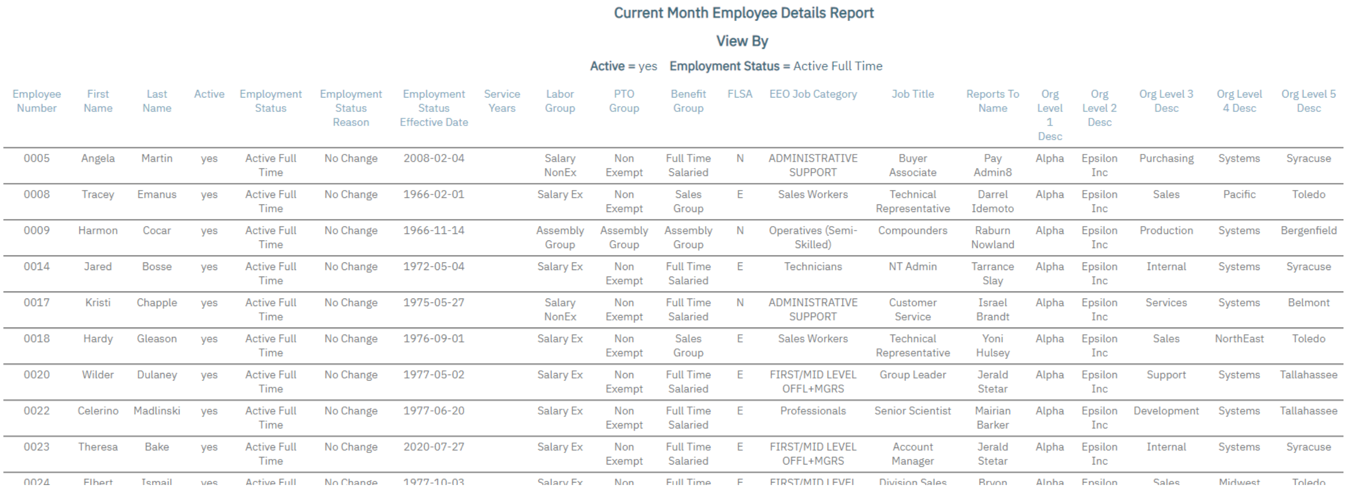
Cancel | Done

## Drill through to employee details

To see detail data on the employees that make up the selected data element you can drill through to a detail report. To do this, right click on a data element in a visualization and select the “Drill through” link.



In this example, clicking the Drill Through report icon will take you to a list of the 494 Active Full Time employees. Hit the back button on the browser to return to the dashboard.



Current Month Employee Details Report

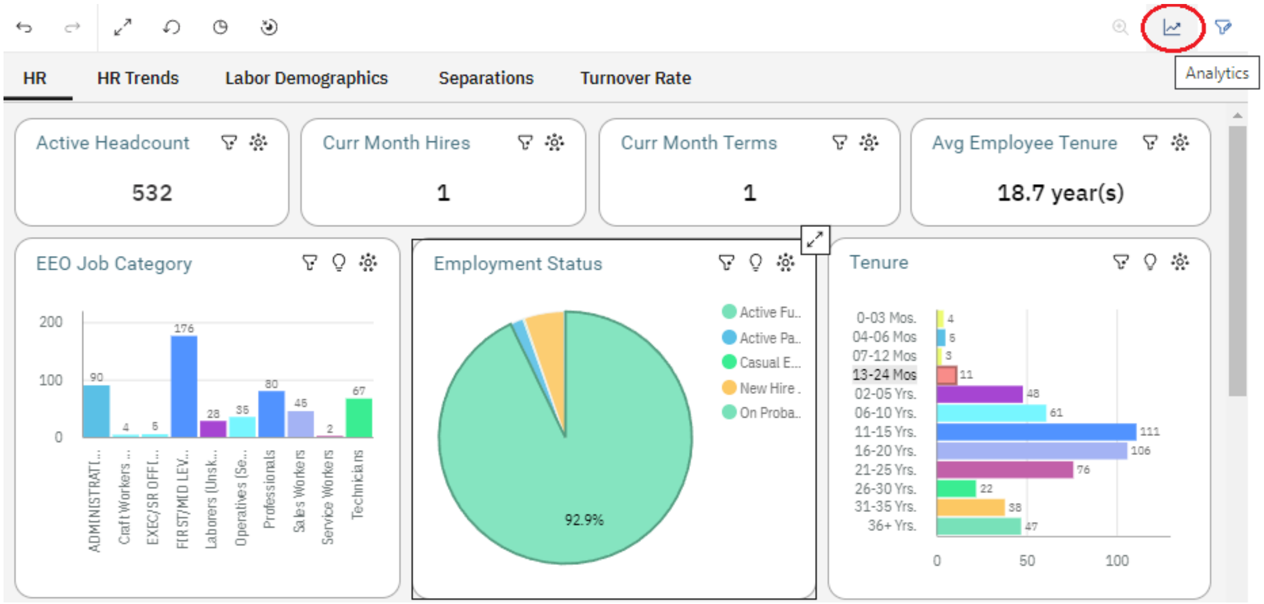
View By

**Active** = yes **Employment Status** = Active Full Time

| Employee Number | First Name | Last Name | Active | Employment Status | Employment Status Reason | Employment Status Effective Date | Service Years | Labor Group | PTO Group | Benefit Group | FLSA | EEO Job Category | Job Title | Reports To Name | Org Level 1 Desc | Org Level 2 Desc | Org Level 3 Desc | Org Level 4 Desc | Org Level 5 Desc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0005 | Angela | Martin | yes | Active Full Time | No Change | 2008-02-04 | | Salary NonEx | Non Exempt | Full Time Salaried | N | ADMINISTRATIVE SUPPORT | Buyer Associate | Pay Admin8 | Alpha | Epsilon Inc | Purchasing | Systems | Syracuse |
| 0008 | Tracey | Emanus | yes | Active Full Time | No Change | 1966-02-01 | | Salary Ex | Non Exempt | Sales Group | E | Sales Workers | Technical Representative | Darrel Idemoto | Alpha | Epsilon Inc | Sales | Pacific | Toledo |
| 0009 | Harmon | Cocar | yes | Active Full Time | No Change | 1966-11-14 | | Assembly Group | Assembly Group | Assembly Group | N | Operatives (Semi-Skilled) | Compounders | Raburn Nowland | Alpha | Epsilon Inc | Production | Systems | Bergenfield |
| 0014 | Jared | Bosse | yes | Active Full Time | No Change | 1972-05-04 | | Salary Ex | Non Exempt | Full Time Salaried | E | Technicians | NT Admin | Tarrance Slay | Alpha | Epsilon Inc | Internal | Systems | Syracuse |
| 0017 | Kristi | Chapple | yes | Active Full Time | No Change | 1975-05-27 | | Salary NonEx | Non Exempt | Full Time Salaried | N | ADMINISTRATIVE SUPPORT | Customer Service | Israel Brandt | Alpha | Epsilon Inc | Services | Systems | Belmont |
| 0018 | Hardy | Gleason | yes | Active Full Time | No Change | 1976-09-01 | | Salary Ex | Non Exempt | Sales Group | E | Sales Workers | Technical Representative | Yoni Hulsey | Alpha | Epsilon Inc | Sales | NorthEast | Toledo |
| 0020 | Wilder | Dulaney | yes | Active Full Time | No Change | 1977-05-02 | | Salary Ex | Non Exempt | Full Time Salaried | E | FIRST/MID LEVEL OFFL+MGRS | Group Leader | Jerald Stetar | Alpha | Epsilon Inc | Support | Systems | Tallahassee |
| 0022 | Celerino | Madlinski | yes | Active Full Time | No Change | 1977-06-20 | | Salary Ex | Non Exempt | Full Time Salaried | E | Professionals | Senior Scientist | Mairian Barker | Alpha | Epsilon Inc | Development | Systems | Tallahassee |
| 0023 | Theresa | Bake | yes | Active Full Time | No Change | 2020-07-27 | | Salary Ex | Non Exempt | Full Time Salaried | E | FIRST/MID LEVEL OFFL+MGRS | Account Manager | Jerald Stetar | Alpha | Epsilon Inc | Internal | Systems | Syracuse |
| 0024 | Elbert | Ismail | yes | Active Full | No Change | 1977-10-03 | | Salary Ex | Non | Full Time | E | FIRST/MID LEVEL | Division Sales | Bryon | Alpha | Epsilon | Sales | Midwest | Toledo |

## Analytics

To see further insights into the data, select a visualization and then select the Analytics icon.



Active Headcount 532

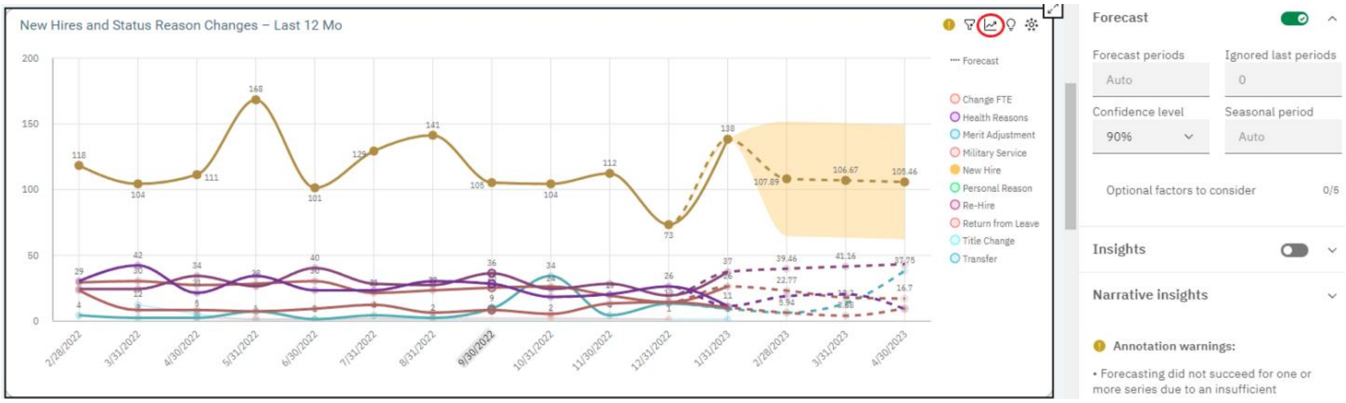
Curr Month Hires 1

Curr Month Terms 1

Avg Employee Tenure 18.7 year(s)

## Forecast

For Trend data you can use Forecast to see the values forecasted for the next 3 months. Select the Forecast icon on the visualization. Options can be selected to adjust the forecast. For example, if we are just beginning the current month you might want to set the “Ignored last periods” to 1 in order to not include the current partial month in the forecast. Select a single line to see a forecasted range of values.



Forecast

Forecast periods: Auto

Ignored last periods: 0

Confidence level: 90%

Seasonal period: Auto

Optional factors to consider 0/5

Insights

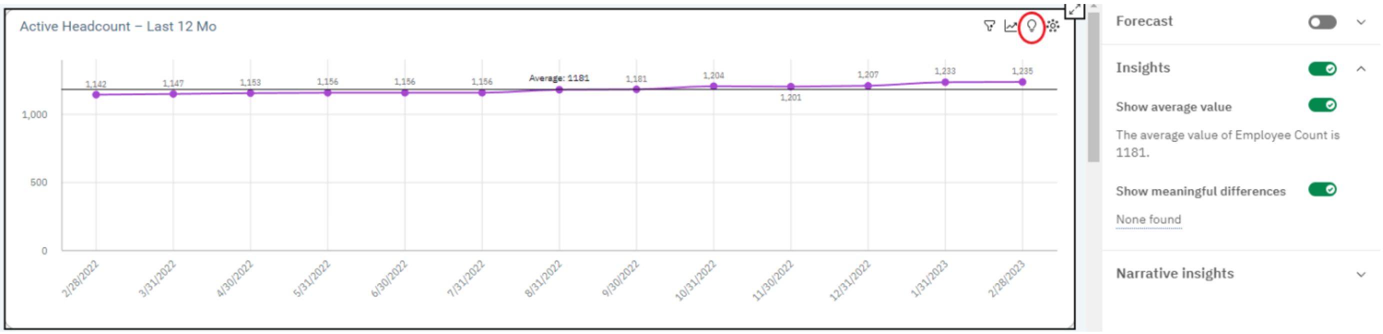
Narrative insights

Annotation warnings:

- Forecasting did not succeed for one or more series due to an insufficient

## Insights

To see further insights into the data, such as average values, select the Insights icon .



Forecast

Insights

Show average value

The average value of Employee Count is 1181.

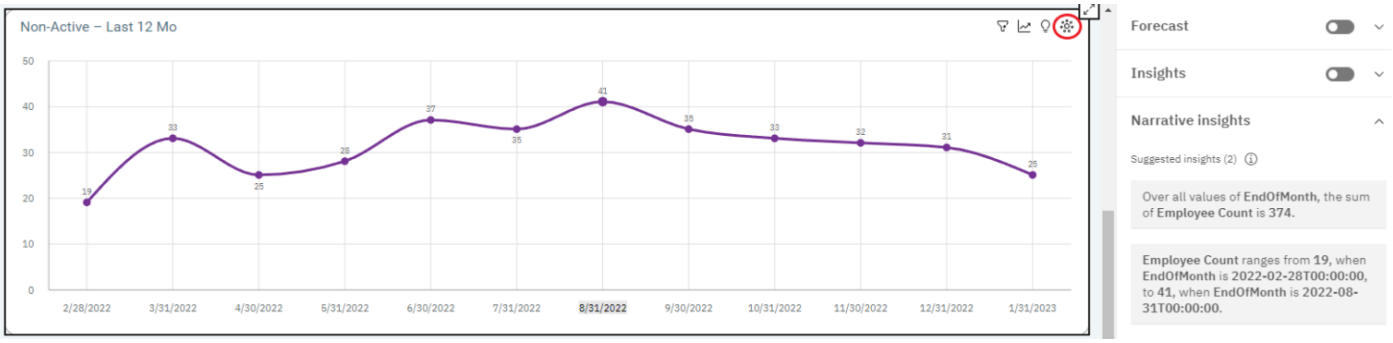
Show meaningful differences

None found

Narrative insights

## Narrative Insights

To view textual descriptions of highlights in the data, select a visualization and then select the Narrative insights icon .



Forecast

Insights

Narrative insights

Suggested insights (2)

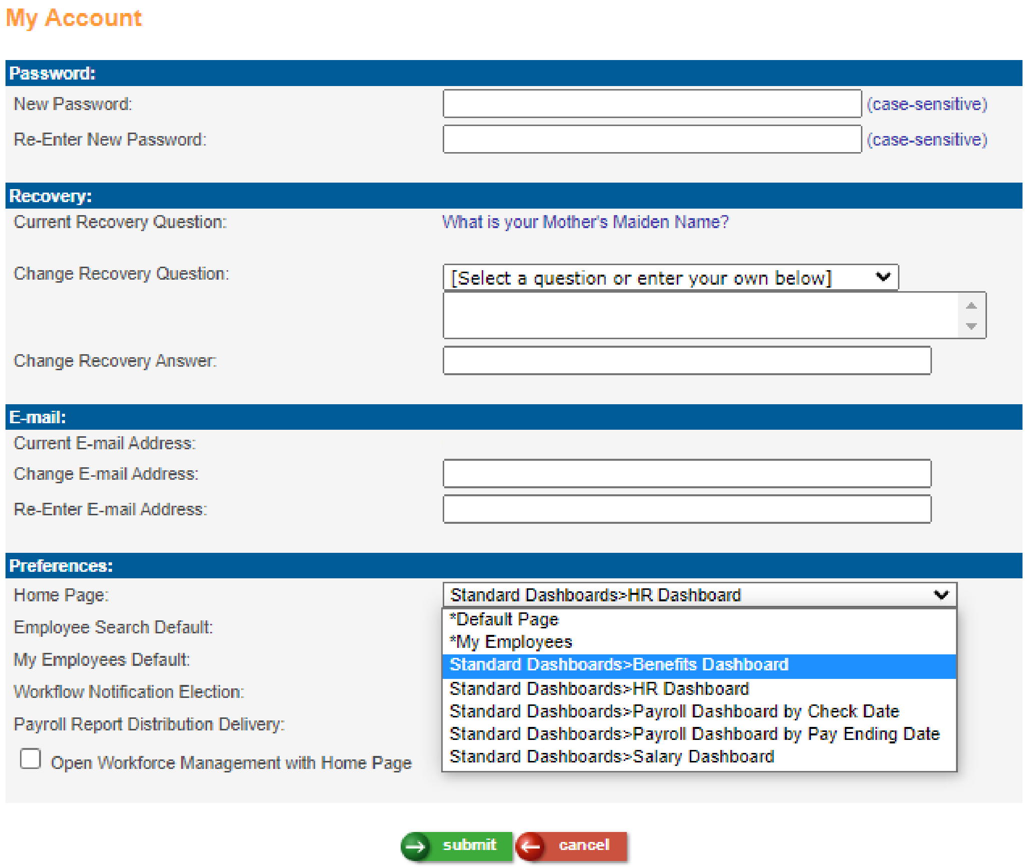
Over all values of EndOfMonth, the sum of Employee Count is 374.

Employee Count ranges from 19, when EndOfMonth is 2022-02-28T00:00:00, to 41, when EndOfMonth is 2022-08-31T00:00:00.

## Set a dashboard as your Manager Services home page

You have the ability to set a dashboard as your Manager Services home page.

Go to Manager Services “My Account” page. Under Preferences > Home Page, select the name of the Dashboard you want as your home page.



My Account

**Password:**

| | | |
|---|---|---|
| New Password: | | (case-sensitive) |
| Re-Enter New Password: | | (case-sensitive) |

**Recovery:**

| | |
|---|---|
| Current Recovery Question: | What is your Mother's Maiden Name? |
| Change Recovery Question: | [Select a question or enter your own below] |
| Change Recovery Answer: | |

**E-mail:**

| | |
|---|---|
| Current E-mail Address: | |
| Change E-mail Address: | |
| Re-Enter E-mail Address: | |

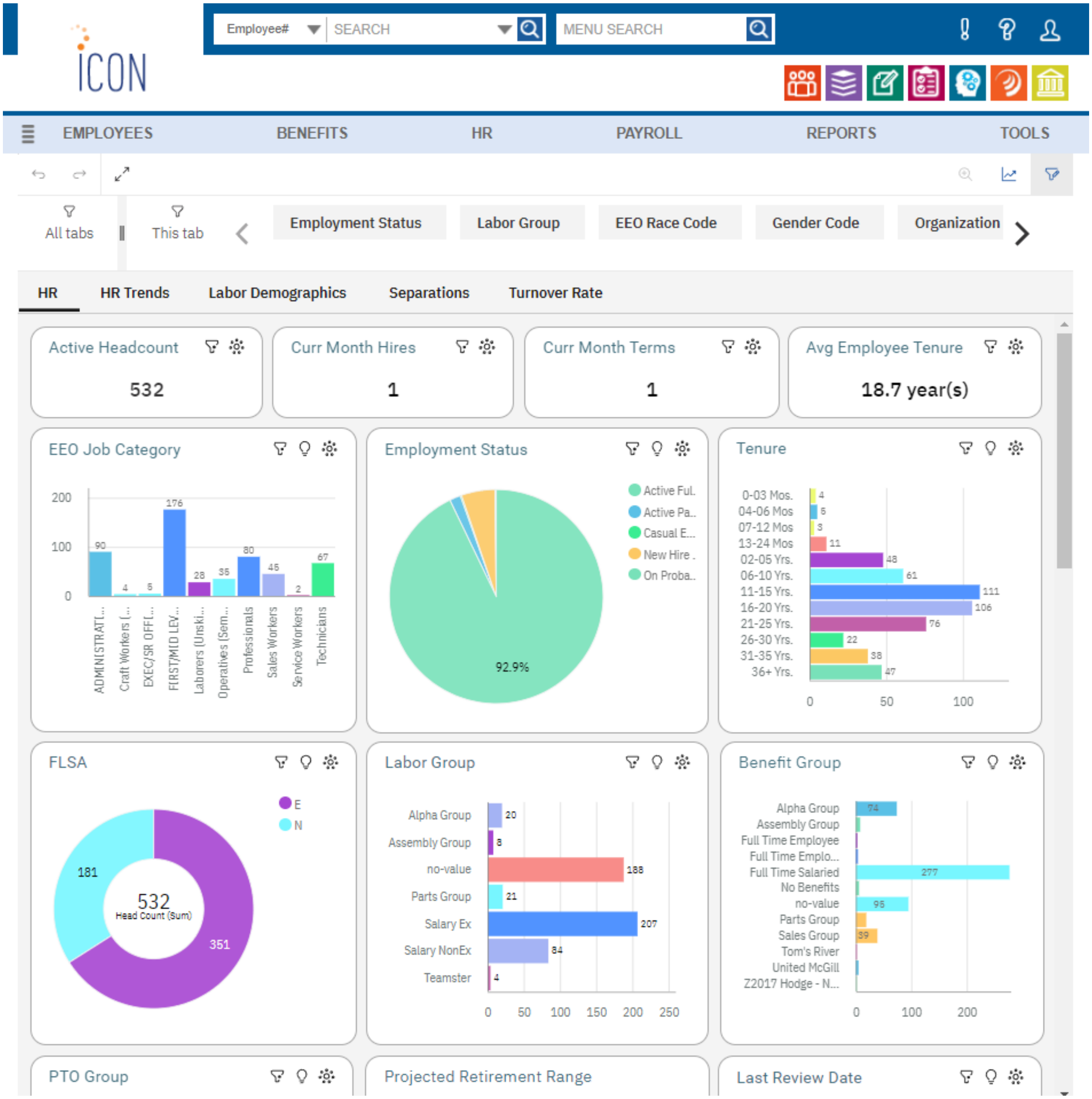
**Preferences:**

| | |
|---|---|
| Home Page: | Standard Dashboards>HR Dashboard |
| Employee Search Default: | |
| My Employees Default: | |
| Workflow Notification Election: | |
| Payroll Report Distribution Delivery: | |
| □ Open Workforce Management with Home Page | |

*Default Page
*My Employees
Standard Dashboards>Benefits Dashboard
Standard Dashboards>HR Dashboard
Standard Dashboards>Payroll Dashboard by Check Date
Standard Dashboards>Payroll Dashboard by Pay Ending Date
Standard Dashboards>Salary Dashboard

**Note: You don’t have to wait for the dashboard to finish loading before using other parts of Manager Services.

HR Dashboard as the Manager Services home page:



## Sharing dashboards

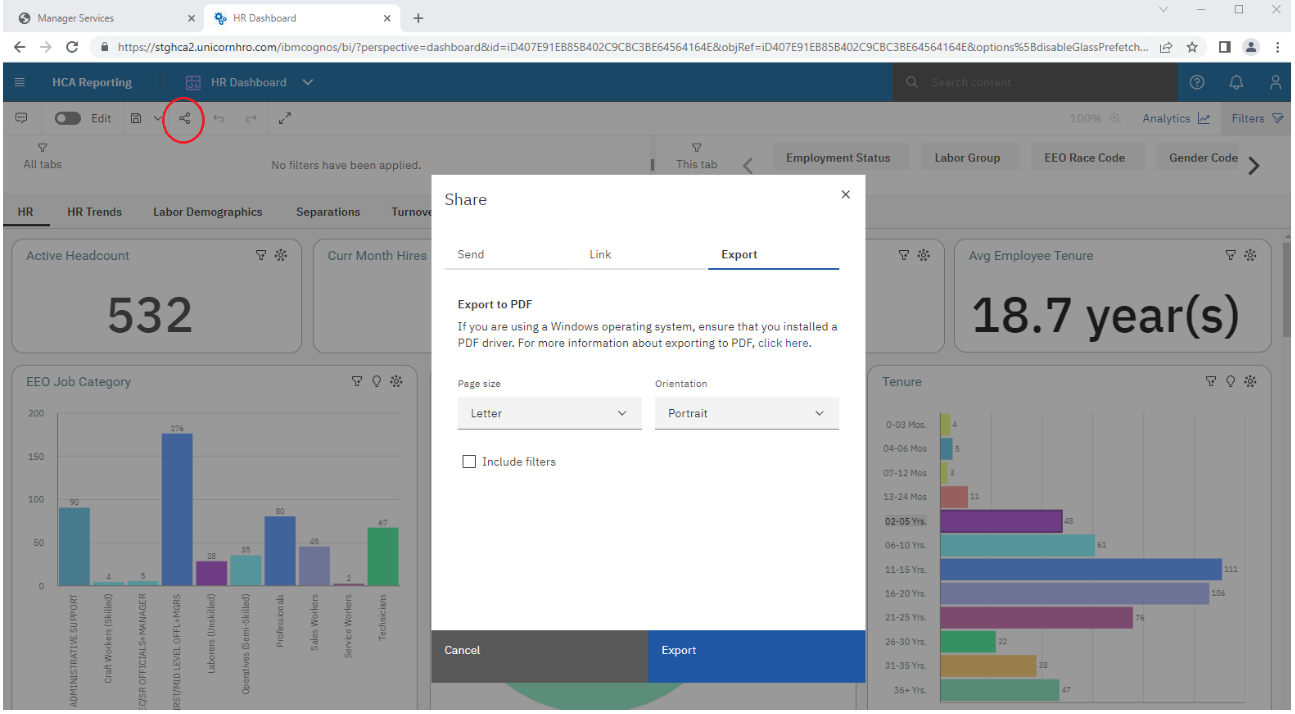
In order to share a dashboard via email or export it to a PDF, the dashboard must be run in HCA Reporting. It can't be done from the Manager Services home page.

Open the dashboard in HCA Reporting, expand the Share menu.

To export to PDF choose the Export tab

To include a picture of the dashboard in an email, choose the Send tab.

PDF:



Share

Send | Link | **Export**

**Export to PDF**

If you are using a Windows operating system, ensure that you installed a PDF driver. For more information about exporting to PDF, click here.

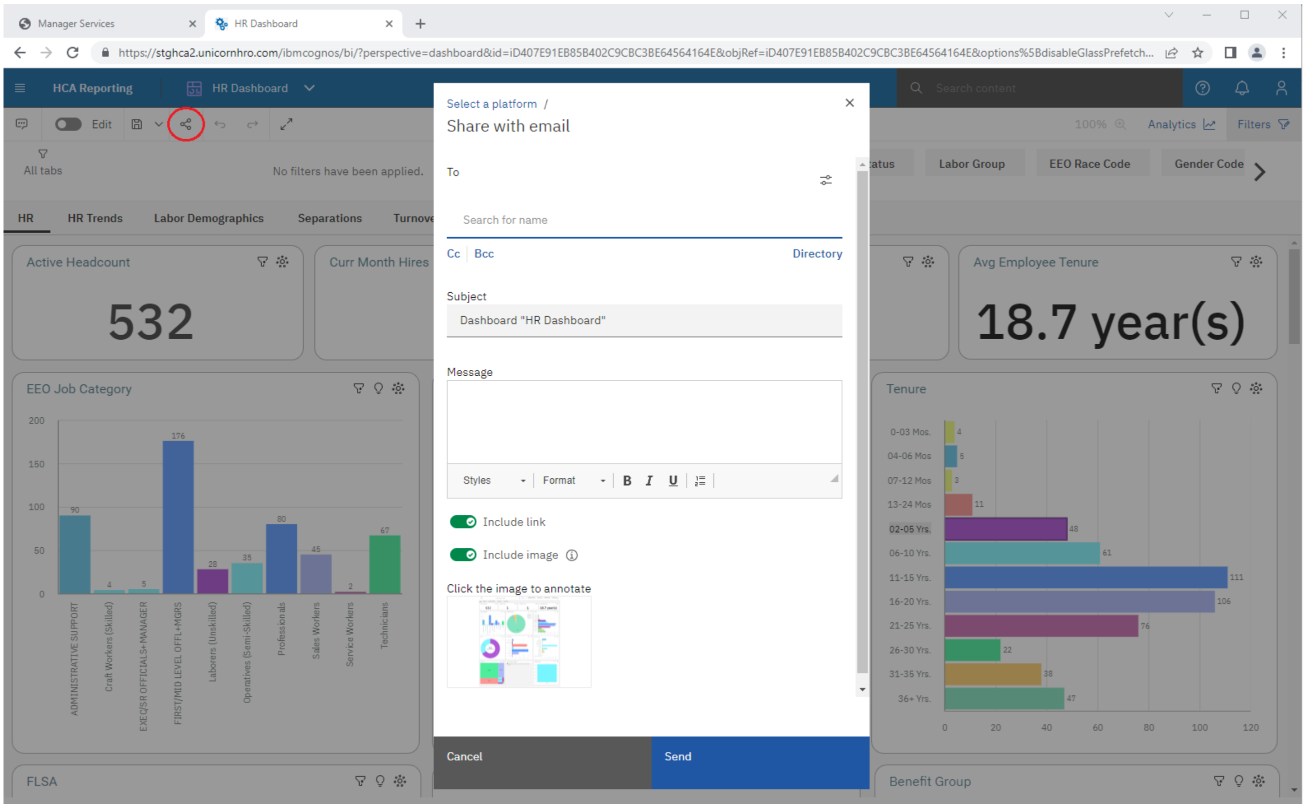
Page size: Letter

Orientation: Portrait

☐ Include filters

Cancel | Export

Email:



Select a platform /

Share with email

To

Search for name

Cc | Bcc | Directory

Subject: Dashboard "HR Dashboard"

Message

Styles | Format | B | I | U

Include link

Include image

Click the image to annotate

Cancel | Send

## Contact us

If you require additional technical support for this or any other HCM product, please contact HCM Support.

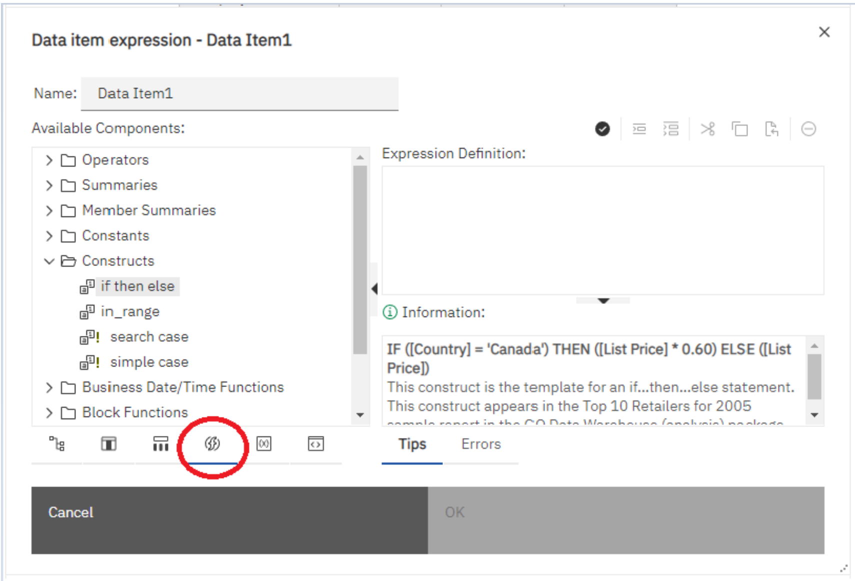
All client questions and assistance requests should be initiated in our Customer Support Portal, ConnectWise.

Please log in to ConnectWise and open a support ticket for your request (https://unicornhcm.myportallogin.com).

# Appendix A – Using the expression editor

An expression is any combination of operators, constants, functions, and other components that evaluates to a single value. You build expressions to create calculation and filter definitions. A calculation is an expression that you use to create a new value from existing values that are contained within a data item. A filter is an expression that you use to retrieve a specific subset of records. Build expressions that use the following components.

Tip: When building expressions, the functions listed in Appendix A can also be found under the Functions tab.



Data item expression - Data Item1

Name: Data Item1

IF ([Country] = 'Canada') THEN ([List Price] * 0.60) ELSE ([List Price])

This construct is the template for an if...then...else statement. This construct appears in the Top 10 Retailers for 2005

## Operators

Operators specify what happens to the values on either side of the operator. Operators are similar to functions; in that they manipulate data items and return a result.

**(**
Identifies the beginning of an expression.
Syntax:
( expression )

**)**
Identifies the end of an expression.
Syntax:
( expression )

*****
Multiplies two numeric values.
Syntax:
value1 * value2

**,**
Separates expression components.
Syntax:
expression ( parameter1, parameter2 )

**/**
Divides two numeric values.
Syntax:
value1 / value2

**||**
Concatenates, or joins, strings.
Syntax:
string1 || string2

**+**
Adds two numeric values.
Syntax:
value1 + value2

**-**
Subtracts two numeric values or negates a numeric value.
Syntax:
value1 - value2
or
- value

**<**
Compares the values that are represented by "value1" against "value2" and retrieves the values that are less than "value2".
Syntax:
value1 < value2

**<=**
Compares the values that are represented by "value1" against "value2" and retrieves the values that are less than or equal to "value2".
Syntax:
value1 <= value2

**<>**
Compares the values that are represented by "value1" against "value2" and retrieves the values that are not equal to "value2".
Syntax:
value1 <> value2

**=**
Compares the values that are represented by "value1" against "value2" and retrieves the values that are equal to "value2".
Syntax:
value1 = value2

**>**
Compares the values that are represented by "value1" against "value2" and retrieves the values that are greater than "value2".
Syntax:
value1 > value2

**->**
Separates the components in a literal member expression.
Syntax:
[namespace].[dimension].[hierarchy].[level]->[L1]

**>=**
Compares the values that are represented by "value1" against "value2" and retrieves the values that are greater than or equal to "value2".
Syntax:
value1 >= value2

**and**
Returns "true" if the conditions on both sides of the expression are true.
Syntax:
argument1 and argument2

**auto**
Works with summary expressions to define the scope to be adjusted based on the grouping columns in the query. The scope is context-dependent.
Syntax:
aggregate_function ( expression AUTO )

**between**
Determines if a value falls in a given range.
Syntax:
expression between value1 and value2
Example:
[Revenue] between 200 and 300
Result
Returns the number of results with revenues between 200 and 300.

**case**
Works with when, then, else, and end. Case identifies the beginning of a specific situation, in which when, then, and else actions are defined.
Syntax:
case expression { when expression then expression } [ else
expression ] end

**contains**
Determines if "string1" contains "string2".
Syntax:
string1 contains string2

**currentMeasure**
Keyword that can be used as the first argument of member summary functions.
Syntax:
aggregate_function ( currentMeasure within set expression )

**default**
Works with the lookup construct.
Syntax:
lookup (....) in (....) default (....)

**distinct**
A keyword used in an aggregate expression to include only distinct occurrences of values. See also the function unique.
Syntax:
distinct dataItem
Example:
count ( distinct [OrderDetailQuantity] )

**else**
Works with the if or case constructs. If the if condition or the case expression are not true, then the else expression is used.
Syntax:
if ( condition ) then .... else ( expression ) , or case .... else ( expression ) end

**end**
Indicates the end of a case or when construct.
Syntax:
case .... end

**ends with**
Determines if "string1" ends with "string2".
Syntax:
string1 ends with string2

**escape**
Determines if "string1" matches the pattern of "string2", with the character "char" optionally used to escape characters in the pattern string.
Syntax:
string1 LIKE string2 [ ESCAPE char ]
Example 1:
[PRODUCT_LINE] like ’G%’
Result:
All product lines that start with 'G'.
Example 2:
[PRODUCT_LINE] like ’%Ga%’ escape ’a’
Result:
All the product lines that end with 'G%'.

**for**
Works with summary expressions to define the scope of the aggregation in the query.
Syntax:
aggregate_function ( expression for expression { , expression } )

**for all**
Works with summary expressions to define the scope to be all the specified grouping columns in the query. See also the for clause.
Syntax:
aggregate_function ( expression for ALL expression { , expression } )

**for any**
Works with summary expressions to define the scope to be adjusted based on a

subset of the grouping columns in the query. Equivalent to the for clause.
Syntax:
aggregate_function ( expression for ANY expression { , expression } )

**for report**
Works with summary expressions to set the scope to be the whole query. See also the for clause.
Syntax:
aggregate_function ( expression for report )

**if**
Works with the then and else constructs. If defines a condition; when the if condition is true, the then expression is used. When the if condition is not true, the else expression is used.
Syntax:
if ( condition ) then ( expression ) else ( expression )

**in**
Determines if "expression1" exists in a given list of expressions.
Syntax:
expression1 in ( expression_list )

**in_range**
Determines if "expression1" exists in a given list of constant values or ranges.
Syntax:
expression1 in_range { constant : constant [ , constant :constant ] }
Example 1:
[code] in_range { 5 }
Result:
This is equivalent to [code] = 5.
Example 2:
[code] in_range { 5: }
Result:
This is equivalent to [code] >= 5.
Example 3:
[code] in_range { :5 }
Result:
This is equivalent to [code] <= 5.
Example 4:
[code] in_range { 5:10 }
Result:
This is equivalent to ( [code] >= 5 and [code] <= 10 ).
Example 5:
[code] in_range { :5,10,20: }
Result:
This is equivalent to ( [code] <= 5 or [code] = 10 or [code] >= 20 ).

**is missing**
Determines if "value" is undefined in the data.
Syntax:
value is missing

**is null**

Determines if "value" is undefined in the data.
Syntax:
value is null

**is not missing**
Determines if "value" is defined in the data.
Syntax:
value is not missing

**is not null**
Determines if "value" is defined in the data.
Syntax:
value is not null

**like**
Determines if "string1" matches the pattern of "string2", with the character "char" optionally used to escape characters in the pattern string.
Syntax:
string1 LIKE string2 [ ESCAPE char ]
Example 1:
[PRODUCT_LINE] like ’G%’
Result
All product lines that start with 'G'.
Example 2:
[PRODUCT_LINE] like ’%Ga%’ escape ’a’
Result
All the product lines that end with 'G%'.

**lookup**
Finds and replaces data with a value you specify. It is preferable to use the case construct.
Syntax:
lookup ( name ) in ( value1 --> value2 ) default ( expression )
Example:
lookup ( [Country]) in ( ’Canada’--> ( [List Price] * 0.60),
’Australia’--> ( [List Price] * 0.80 ) ) default ( [List Price] )

**not**
Returns TRUE if "argument" is false or returns FALSE if "argument" is true.
Syntax:
NOT argument

**or**
Returns TRUE if either of "argument1" or "argument2" are true.
Syntax:
argument1 or argument2

**prefilter**
Performs a summary calculation before applying the summary filter.
Syntax:
summary_function ([expression] prefilter)
Example:
total ( [Quantity] for report prefilter )

summaryFilter: total(
[Quantity] for [ProductNo] ) > 50000
Result
Sums the quantities in a report before the summary filter is applied.

**rows**
Counts the number of rows output by the query. Use with Count ().
Syntax:
count ( ROWS )

**starts with**
Determines if "string1" starts with "string2".
Syntax:
string1 starts with string2

**then**
Works with the if or case constructs. When the if condition or the when expression are true, the then expression is used.
Syntax:
if ( condition ) then ..., or case expression when expression then .... end

**when**
Works with the case construct. You can define conditions to occur when the WHEN expression is true.
Syntax:
case [expression] when ... end

## Constants

A constant is a fixed value that you can use in an expression.

**date**
Inserts the current system date.

**date-time**
Inserts the current system date and time.

**time with time zone**
Inserts a zero time with time zone.

**timestamp with time zone**
Inserts an example of a timestamp with time zone.

**interval**
Inserts a zero interval: 000 00:00:00.000.

**interval year**
inserts a zero year interval: 0 year.

**interval month**
Inserts a zero month interval: 0 month.

**interval year to month**
Inserts a zero year to month interval: 0000-00 year to month.

**interval day**
Inserts a zero day interval: 0 day.

**interval hour**
Inserts a zero hour interval: 0 hour.

**interval minute**
Inserts a zero minute interval: 0 minute.

**interval second**
Inserts a zero second interval: 0 second.

**interval day to hour**
Inserts a zero day to hour interval: 0 00 day to hour.

**interval day to minute**
Inserts a zero day to minute interval: 0 00:00 day to minute.

**interval day to second**
Inserts a zero day to second interval: 0 00:00:00.000000000 day to second.

**interval hour to minute**
Inserts a zero hour to minute interval: 00:00 hour to minute.

**interval hour to second**

Inserts a zero hour to second interval: 00:00:00.000000000 hour to second.

**interval minute to second**
Inserts a zero minute to second interval: 00:00.000000000 minute to second.

**null**
Inserts "null" if the expression conditions are not met.

**number**
Inserts the number 0, which can be replaced with a new numeric value.

**string**
Inserts an empty string as two single quotation marks between which you can type a string.

**time**
Inserts the current system time.

## Constructs

This list contains constructs and templates that can be used to create an expression. Templates combine multiple functions into a group. For example, the search case template includes the case, when, else, and end functions.

**if then else**
This construct is the template for an if...then...else statement.
Syntax:
IF ([Country] = ’Canada’) THEN ([List Price] * 0.60) ELSE ([List Price])

**in_range**
This is the template for an in_range expression.
Syntax:
[code] IN_RANGE { :30 , 40, 50, 999: }
Example 1:
[code] IN_RANGE { 5 }
Result:
This is equivalent to [code] = 5.
Example 2:
[code] IN_RANGE { 5: }
Result:
This is equivalent to [code] >= 5.
Example 3:
[code] IN_RANGE { :5 }
Result:
This is equivalent to [code] <= 5.
Example 4:
[code] IN_RANGE { 5:10 }
Result:
This is equivalent to ( [code] >= 5 and [code] <= 10 ).
Example 5:
[code] IN_RANGE { :5,10,20: }
Result:
This is equivalent to ( [code] <= 5 or [code] = 10 or [code] >= 20 ).

**search case**
This construct is the template for a search case, including the CASE, WHEN, ELSE and END functions.
Syntax:
CASE WHEN [Country] = ’Canada’ THEN ([List Price] * 0.60) WHEN
[CountryCode] > 100 THEN [List Price] * 0.80 ELSE [List Price] END

**simple case**
This construct is the template for a simple case, including the CASE, WHEN, ELSE and END functions.
Syntax:
CASE [Country] WHEN ’Canada’ THEN ([List Price] * 0.60) WHEN
’Australia’ THEN [List Price] * 0.80 ELSE [List Price] END

## Business date/time functions

This list contains business functions for performing date and time calculations.

**_add_seconds**
Returns the time or datetime, depending on the format of "time_expression", that results from adding "integer_expression" seconds to "time_expression".
Syntax:
_add_seconds ( time_expression, integer_expression )
Example 1:
_add_seconds ( 13:04:59 , 1 )
Result:
13:05:00
Example 2:
_add_seconds ( 2002-04-30 12:10:10.000, 1 )
Result:
2002-04-30 12:10:11.000

**_add_minutes**
Returns the time or datetime, depending on the format of "time_expression", that results from adding "integer_expression" minutes to "time_expression".
Syntax:
_add_minutes ( time_expression, integer_expression )
Example:
_add_minutes ( 13:59:00 , 1 )
Result:
14:00:00

**_add_hours**
Returns the time or datetime, depending on the format of "time_expression", that results from adding "integer_expression" hours to "time_expression".
Syntax:
_add_hours ( time_expression, integer_expression )
Example 1:
_add_hours ( 13:59:00 , 1 )
Result:
14:59:00
Example 2:
_add_hours ( 2002-04-30 12:10:10.000, 1 )
Result:
2002-04-30 13:10:10.000,

**_add_days**
Returns the date or datetime, depending on the format of "date_expression", that results from adding "integer_expression" days to "date_expression".
Syntax:
_add_days ( date_expression, integer_expression )
Example 1:
_add_days ( 2002-04-30 , 1 )
Result:
2002-05-01
Example 2:
_add_days ( 2002-04-30 12:10:10.000, 1 )

Result:
2002-05-01 12:10:10.000

**_add_months**
Adds "integer_expression" months to "date_expression". If the resulting month has fewer days than the day of month component, then the last day of the resulting month is returned. In all other cases the returned value has the same day of month component as "date_expression".
Syntax:
_add_months ( date_expression, integer_expression )
Example 1:
_add_months ( 2012-04-15 , 3 )
Result:
2012-07-15
Example 2:
_last_of_month ( _add_months ( 2012-02-29 , 1 ) )
Result:
2012-03-31

**_add_years**
Adds "integer_expression" years to "date_expression". If the "date_expression" is February 29 and resulting year is non leap year, then the resulting day is set to February 28. In all other cases the returned value has the same day and month as "date_expression".
Syntax:
_add_years ( date_expression, integer_expression )
Example:
_add_years ( 2012-04-15 , 1 )
Result:
2013-04-15

**_age**
Returns a number that is obtained from subtracting "date_expression" from today's date. The returned value has the form YYYYMMDD, where YYYY represents the number of years, MM represents the number of months, and DD represents the number of days.
Syntax:
_age ( date_expression )
Example:
_age ( 1990-04-30 ) (if today’s date is 2003-02-05)
Result:
120906, meaning 12 years, 9 months, and 6 days.

**_day_of_week**
Returns the day of week (1 to 7), where 1 is the first day of the week as indicated by the second parameter (1 to 7, 1 being Monday and 7 being Sunday). Note that in ISO 8601 standard, a week begins with Monday being day 1.
Syntax:
_day_of_week ( date_expression, integer )
Example:
_day_of_week ( 2003-01-01 , 1 )
Result:
3

**_day_of_year**
Returns the day of year (1 to 366) in "date_ expression". Also known as Julian day.
Syntax:
_day_of_year ( date_expression )
Example:
_day_of_year ( 2003-03-01 )
Result:
61

**_days_between**
Returns a positive or negative number representing the number of days between "date_expression1" and "date_expression2". If "date_expression1" < "date_expression2", then the result will be a negative number.
Syntax:
_days_between ( date_expression1 , date_expression2 )
Example:
_days_between ( 2002-04-30 , 2002-06-21 )
Result:
-52

**_days_to_end_of_month**
Returns a number representing the number of days remaining in the month represented by "date_expression".
Syntax:
_days_to_end_of_month ( date_expression )
Example:
_days_to_end_of_month ( 2002-04-20 14:30:22.123 )
Result:
10

**_first_of_month**
Returns a date or datetime, depending on the argument, by converting "date_expression" to a date with the same year and month but with the day set to 1.
Syntax:
_first_of_month ( date_expression )
Example 1:
_first_of_month ( 2002-04-20 )
Result:
2002-04-01
Example 2:
_first_of_month ( 2002-04-20 12:10:10.000 )
Result:
2002-04-01 12:10:10.000

**_last_of_month**
Returns a date or datetime, depending on the argument, that is the last day of the month represented by "date_expression".
Syntax:
_last_of_month ( date_expression )
Example 1:
_last_of_month ( 2002-01-14 )
Result:

2002-01-31
Example 2:
_last_of_month ( 2002-01-14 12:10:10.000 )
Result:
2002-01-31 12:10:10.000

**_make_timestamp**
Returns a timestamp constructed from "integer_expression1" (the year), "integer_expression2" (the month), and "integer_expression3" (the day). The time portion defaults to 00:00:00.000 .
Syntax:
_make_timestamp ( integer_expression1, integer_expression2, integer_expression3 )
Example:
_make_timestamp ( 2002 , 01 , 14 )
Result:
2002-01-14 00:00:00.000

**_months_between**
Returns a positive or negative integer number representing the number of months between "date_expression1" and "date_expression2". If "date_expression1" is earlier than "date_expression2", then a negative number is returned.
Syntax:
_months_between ( date_expression1, date_expression2 )
Example:
_months_between ( 2002-04-03 , 2002-01-30 )
Result:
2

**_week_of_year**
Returns the number of the week of the year of "date_expression" according to the ISO 8601 standard. Week 1 of the year is the first week of the year to contain a Thursday, which is equivalent to the first week containing January 4th. A week starts on Monday (day 1) and ends on Sunday (day 7).
Syntax:
_week_of_year ( date_expression )
Example:
_week_of_year ( 2003-01-01 )
Result:
1

**_years_between**
Returns a positive or negative integer number representing the number of years between "date_expression1" and "date_expression2". If "date_expression1" < "date_expression2" then a negative value is returned.
Syntax:
_years_between ( date_expression1, date_expression2 )
Example:
_years_between ( 2003-01-30 , 2001-04-03 )
Result:
1

**_ymdint_between**
Returns a number representing the difference between "date_expression1" and "date_expression2". The returned value has the form YYYYMMDD, where YYYY

represents the number of years, MM represents the number of months, and DD represents the number of days.

Syntax:

_ymdint_between ( date_expression1 , date_expression2 )

Example:

_ymdint_between ( 1990-04-30 , 2003-02-05 )

Result:

120906, meaning 12 years, 9 months and 6 days.

## Summaries

This list contains predefined functions that return either a single summary value for a group of related values or a different summary value for each instance of a group of related values.

**aggregate**
Returns a calculated value using the appropriate aggregation function, based on the aggregation type of the expression.
Syntax:
aggregate ( expression [ auto ] )
aggregate ( expression for [ all|any ] expression { , expression } )
aggregate ( expression for report )

**average**
Returns the average value of selected data items. Distinct is an alternative expression that is compatible with earlier versions of the product.
Syntax:
average ( [ distinct ] expression [ auto ] )
average ( [ distinct ] expression for [ all|any ] expression { , expression } )
average ( [ distinct ] expression for report )
Example:
average ( Sales )
Result:
Returns the average of all Sales values.

**count**
Returns the number of selected data items excluding null values. Distinct is an alternative expression that is compatible with earlier versions of the product.
Syntax:
count ( [ all | distinct ] expression [ auto ] )
count ( [ all | distinct ] expression for [ all|any ] expression { ,expression } )
count ( [ all | distinct ] expression for report )
Example:
count ( Sales )
Result:
Returns the total number of entries under Sales.

**maximum**
Returns the maximum value of selected data items. Distinct is an alternative expression that is compatible with earlier versions of the product.
Syntax:
maximum ( [ distinct ] expression [ auto ] )
maximum ( [ distinct ] expression for [ all|any ] expression { ,expression } )
maximum ( [ distinct ] expression for report )
Example:
maximum ( Sales )
Result:
Returns the maximum value out of all Sales values.

**median**
Returns the median value of selected data items.
Syntax:
median ( expression [ auto ] )

median ( expression for [ all|any ] expression { , expression } )
median ( expression for report )

**minimum**
Returns the minimum value of selected data items. Distinct is an alternative
expression that is compatible with earlier versions of the product.
Syntax:
minimum ( [ distinct ] expression [ auto ] )
minimum ( [ distinct ] expression for [ all|any ] expression { ,expression } )
minimum ( [ distinct ] expression for report )
Example:
minimum ( Sales )
Result:
Returns the minimum value out of all Sales values.

**moving-average**
Returns a moving average by row for a specified set of values of over a specified number of rows. The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources.
Syntax:
moving-average ( numeric_expression , numeric_expression [ at expression { , expression } ] [ <for-option> ] [ prefilter ] )
moving-average ( numeric_expression , numeric_expression [ <for-option> ] [ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto
Example:
moving-average ( Qty , 3 )
Result:
For each row, returns the quantity and a moving average of the current row and
the preceding two rows.

**moving-total**
Returns a moving total by row for a specified set of values over a specified number of rows. The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources.
Syntax:
moving-total ( numeric_expression , numeric_expression [ at expression { , expression } ] [ <for-option> ] [ prefilter ] )
moving-total ( numeric_expression , numeric_expression [ <for-option> ] [ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto
Example:
moving-total ( Qty , 3 )
Result:
For each row, returns the quantity and a moving total of the current row and the preceding two rows.

**percentage**
Returns the percent of the total value for selected data items. The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources. This function appears in the Percentage Calculation (by year) interactive sample report.
Syntax:
percentage ( numeric_expression [ at expression { , expression } ]
[ <for-option> ] [ prefilter ] )
percentage ( numeric_expression [ <for-option> ] [ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto

Example:
percentage ( Sales 98 )
Result:
Returns the percentage of the total sales for 1998 that is attributed to each sales
representative.

**percentile**
Returns a value, on a scale of one hundred, that indicates the percent of a distribution that is equal to or below the selected data items. The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources.
Syntax:
percentile ( numeric_expression [ at expression { , expression } ][ <for-option> ] [ prefilter ] )
percentile ( numeric_expression [ <for-option> ] [ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto
Example:
percentile ( Sales 98 )
Result:
For each row, returns the percentage of rows that are equal to or less than the
quantity value of that row.

**quantile**
Returns the rank of a value within a range that you specify. It returns integers to represent any range of ranks, such as 1 (highest) to 100 (lowest). The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources.
Syntax:
quantile ( numeric_expression , numeric_expression [ at expression { , expression } ] [ <for-option> ] [ prefilter ] )
quantile ( numeric_expression , numeric_expression [ <for-option> ][ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto
Example:
quantile ( Qty , 4 )
Result:
Returns the quantity, the rank of the quantity value, and the quantity values
broken down into 4 quantile groups (quartiles).

**rank**
Returns the rank value of selected data items. The sort order is optional; descending order (DESC) is assumed by default. If two or more rows tie, then there is a gap in the sequence of ranked values (also known as Olympic ranking). The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources. Distinct is an alternative expression that is compatible with earlier versions of the product. Null values are ranked last.
Syntax:
rank ( expression [ ASC|DESC ] { , expression [ ASC|DESC ] } [ at expression { , expression } ] [ <for-option>] [ prefilter ] )
rank ( [ distinct ] expression [ ASC|DESC ] { , expression[ ASC|DESC ] } [ <for-option>] [ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto
Example:
rank ( Sales 98 )
Result:
For each row, returns the rank value of sales for 1998 that is attributed to each
sales representative. Some numbers are skipped when a tie between rows occurs.

**running-average**

Returns the running average by row (including the current row) for a set of values. The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources.
Syntax:
running-average ( numeric_expression [ at expression { ,expression } ] [ <for-option> ] [ prefilter ] )
running-average ( numeric_expression [ <for-option> ] [ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto
Example:
running-average ( Qty )
Result:
For each row, returns the quantity and a running average of the current and the
previous rows.

**running-count**
Returns the running count by row (including the current row) for a set of values. The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources.
Syntax:
running-count ( numeric_expression [ at expression { , expression } ][ <for-option> ] [ prefilter ] )
running-count ( numeric_expression [ <for-option> ] [ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto
Example:
running-count ( Qty )
Result:
For each row, returns the quantity and a running count of the position of the current row.

**running-difference**
Returns a running difference by row, calculated as the difference between the value for the current row and the preceding row, (including the current row) for a set of values. The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources.
Syntax:
running-difference ( numeric_expression [ at expression { ,expression } ] [ <for-option> ] [ prefilter ] )
running-difference ( numeric_expression [ <for-option> ][ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto
Example:
running-difference ( Qty )
Result:
For each row, returns the quantity and a running difference between the value for
the current row and the preceding row.

**running-maximum**
Returns the running maximum by row (including the current row) for a set of values. The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources.
Syntax:
running-maximum ( numeric_expression [ at expression { ,expression } ] [ <for-option> ] [ prefilter ] )
running-maximum ( numeric_expression [ <for-option> ] [ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto
Example:
running-maximum ( Qty )
Result:
For each row, returns the quantity and a running maximum of the current and previous rows.

**running-minimum**
Returns the running minimum by row (including the current row) for a set of values. The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources.
Syntax:
running-minimum ( numeric_expression [ at expression { ,expression } ] [ <for-option> ] [ prefilter ] )
running-minimum ( numeric_expression [ <for-option> ] [ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto
Example:
running-minimum ( Qty )
Result:
For each row, returns the quantity and a running minimum of the current and previous rows.

**running-total**
Returns a running total by row (including the current row) for a set of values. The "<for-option>" defines the scope of the function. The "at" option defines the level of aggregation and can be used only in the context of relational datasources.
Syntax:
running-total ( numeric_expression [ at expression { ,expression } ] [ <for-option> ] [ prefilter ] )
running-total ( numeric_expression [ <for-option> ] [ prefilter ] )
<for-option> ::= for expression { , expression }|for report|auto
Example:
running-total ( Qty )
Result:
For each row, returns the quantity and a running total of the current and previous rows.

**total**
Returns the total value of selected data items. Distinct is an alternative expression that is compatible with earlier versions of the product.
Syntax:
total ( [ distinct ] expression [ auto ] )
total ( [ distinct ] expression for [ all|any ] expression { ,expression } )
total ( [ distinct ] expression for report )
Example:
total ( Sales )
Result:
Returns the total value of all Sales values.

## Common functions

Functions are pre-written formulas that simplify the process of creating calculations. Using functions, you can quickly create formulas that may be difficult to build yourself.

**abs**
Returns the absolute value of "numeric_expression". Negative values are returned as positive values.
Syntax:
abs ( numeric_expression )
Example 1:
abs ( 15 )
Result:
15
Example 2:
abs ( -15 )
Result:
15

**cast**
Converts "expression" to a specified data type. Some data types allow for a length and precision to be specified. Make sure that the target is of the appropriate type and size. The following can be used for "datatype_specification": character, varchar, char, numeric, decimal, integer, bigint, smallint, real, float, date, time, timestamp, time with time zone, timestamp with time zone, and interval. When type casting to an interval type, one of the following interval qualifiers must be specified: year, month, or year to month for the year-to-month interval datatype; day, hour,minute, second, day to hour, day to minute, day to second, hour to minute, hour to second, or minute to second for the day-to-second interval datatype. Notes:
When you convert a value of type timestamp to type date, the time portion of the timestamp value is ignored.
When you convert a value of type timestamp to type time, the date portion of the timestamp is ignored.
When you convert a value of type date to type timestamp, the time components of the timestamp are set to zero.
When you convert a value of type time to type timestamp, the date component is set to the current system date. It is invalid to convert one interval datatype to the other (for instance because the number of days in a month is variable). Note that you can specify the number of digits for the leading qualifier only, i.e. YEAR(4) TO MONTH, DAY(5). Errors will be reported if the target type and size are not compatible with the source type and size.
Syntax:
cast ( expression , datatype_specification )
Example 1:
cast ( ’123’ , integer )
Result:
123
Example 2:
cast ( 12345 , varchar ( 10 ) )
Result:
a string containing 12345

**ceil**
Returns the smallest integer that is greater than or equal to "numeric_expression".
Syntax:
ceil ( numeric_expression )

**ceiling**
Returns the smallest integer that is greater than or equal to "numeric_expression".

Syntax:
ceiling ( numeric_expression )
Example 1:
ceiling ( 4.22 )
Result:
5
Example 2:
ceiling ( -1.23 )
Result:
-1

**char_length**
Returns the number of logical characters in "string_expression”.
Syntax:
char_length ( string_expression )
Example:
char_length ( ’Canada’ )
Result:
6

**character_length**
Returns the number of characters in "string_expression".
Syntax:
character_length ( string_expression )
Example:
character_length ( ’Canada’ )
Result:
6

**coalesce**
Returns the first non-null argument (or null if all arguments are null). Requires two or more arguments in "expression_list".
Syntax:
coalesce ( expression_list )
Example:
coalesce ( [Unit price], [Unit sale price] )
Result:
Returns the unit price, or the unit sale price if the unit price is null.

**current_date**
Returns a date value representing the current date of the computer that the database software runs on.
Syntax:
current_date
Example:
current_date
Result:
2003-03-04

**current_time**
Returns a time with time zone value, representing the current time of the computer that runs the database software if the database supports this function. Otherwise, it represents the current time of the computer that runs IBM® Cognos® BI software.
Syntax:

current_time
Example:
current_time
Result:
16:33:11.354+05:00

**current_timestamp**
Returns a datetime with time zone value, representing the current time of the computer that runs the database software if the database supports this function. Otherwise, it represents the current time of the computer that runs IBM® Cognos® BI software.
Syntax:
current_timestamp
Example:
current_timestamp
Result:
2003-03-03 16:40:15.535+05:00

**exp**
Returns 'e' raised to the power of "numeric_expression". The constant 'e' is the base of the natural logarithm.
Syntax:
exp ( numeric_expression )
Example:
exp ( 2 )
Result
7.389056

**extract**
Returns an integer representing the value of datepart (year, month, day, hour,minute, second) in "datetime_expression".
Syntax:
extract ( datepart , datetime_expression )
Example 1:
extract ( year , 2003-03-03 16:40:15.535 )
Result:
2003
Example 2:
extract ( hour , 2003-03-03 16:40:15.535 )
Result:
16

**floor**
Returns the largest integer that is less than or equal to "numeric_expression".
Syntax:
floor ( numeric_expression )
Example 1**:**
floor ( 3.22 )
Result:
3
Example 2:
floor ( -1.23 )
Result:
-2

**ln**
Returns the natural logarithm of "numeric_expression".
Syntax:
ln ( numeric_expression )
Example:
ln ( 4 )
Result:
1.38629

**localtime**
Returns a time value, representing the current time of the computer that runs the database software.
Syntax:
localtime
Example:
localtime
Result:
16:33:11

**localtimestamp**
Returns a datetime value, representing the current timestamp of the computer that runs the database software.
Syntax:
localtimestamp
Example:
localtimestamp
Result:
2003-03-03 16:40:15

**lower**
Returns "string_expression" with all uppercase characters shifted to lowercase.
Syntax:
lower ( string_expression )
Example:
lower ( ’ABCDEF’ )
Result:
abcdef

**mod**
Returns the remainder (modulus) of "integer_expression1" divided by "integer_expression2". "Integer_expression2" must not be zero or an exception condition is raised.
Syntax:
mod ( integer_expression1, integer_expression2 )
Example:
mod ( 20 , 3 )
Result:
2

**nullif**
Returns null if "expression1" equals "expression2", otherwise returns "expression1".
Syntax:
nullif ( expression1, expression2 )

**octet_length**

Returns the number of bytes in "string_expression".
Syntax:
octet_length ( string_expression )
Example 1:
octet_length ( ’ABCDEF’ )
Result:
6
Example 2:
octet_length ( ’’ )
Result:
0

**position**
Returns the integer value representing the starting position of "string_expression1" in "string_expression2" or 0 when the "string_expression1" is not found.
Syntax:
position ( string_expression1 , string_expression2 )
Example 1:
position ( ’C’ , ’ABCDEF’ )
Result:
3
Example 2:
position ( ’H’ , ’ABCDEF’ )
Result:
0

**power**
Returns "numeric_expression1" raised to the power "numeric_expression2". If "numeric_expression1" is negative, then "numeric_expression2" must result in an integer value.
Syntax:
power ( numeric_expression1 , numeric_expression2 )
Example:
power ( 3 , 2 )
Result:
9

**_round**
Returns "numeric_expression" rounded to "integer_expression" places to the rightof the decimal point.
Notes: "integer_expression" must be a non-negative integer. Rounding takes place before data formatting is applied.
Syntax:
_round ( numeric_expression , integer_expression )
Example:
_round ( 1220.42369, 2 )
Result:
1220.42

**sqrt**
Returns the square root of "numeric_expression". "Numeric_expression" must be non-negative.
Syntax:
sqrt ( numeric_expression )
Example:
sqrt ( 9 )

Result:
3

**substring**

Returns the substring of "string_expression" that starts at position "integer_expression1" for "integer_expression2" characters or to the end of "string_expression" if "integer_expression2" is omitted. The first character in "string_expression" is at position 1.
Syntax:
substring ( string_expression , integer_expression1 [ ,integer_expression2 ] )
Example:
substring ( ’abcdefg’ , 3 , 2 )
Result:
Cd

**trim**

Returns "string_expression" trimmed of leading and trailing blanks or trimmed of a certain character specified in "match_character_expression". "Both" is implicit when the first argument is not stated and blank is implicit when the second argument is not stated.
Syntax:
trim ( [ [ trailing|leading|both ] [ match_character_expression ] , ] string_expression )
Example 1:
trim ( trailing ’A’ , ’ABCDEFA’ )
Result:
ABCDEF
Example 2:
trim ( both , ’ ABCDEF ’ )
Result:
ABCDEF

**upper**

Returns "string_expression" with all lowercase characters converted to uppercase.
Syntax:
upper ( string_expression )
Example:
upper ( ’abcdef’ )
Result:
ABCDEF

# Report functions

**Today**
Returns the current system date.
Syntax:
Today ()

**Now**
Returns the current system time.
Syntax:
Now ()

**AsOfDate**
Returns the date value of the AsOfDate expression, if it is defined. Otherwise, AsOfDate returns the report execution date.
Syntax:
AsOfDate ()

**AsOfTime**
Returns the time value of the AsOfTime expression, if it is defined. Otherwise, AsOfTime returns the report execution time.
Syntax:
AsOfTime ()

**ReportDate**
Returns the report execution date and time.
Syntax:
ReportDate ()

**ReportName**
Returns the report name. If a saved report view is run, returns the report view name.
Syntax:
ReportName ()

**ReportPath**
Returns the report path.
Syntax:
ReportPath ()

**ReportDescription**
Returns the report description.
Syntax:
ReportDescription ()

**ReportLocale**
Returns the run locale.
Syntax:
ReportLocale ()

**GetLocale**
Returns the run locale (deprecated).
Syntax:
GetLocale ()

**Locale**
Returns the run locale.
Syntax:
Locale ()

**ReportProductLocale**
Returns the product locale.
Syntax:
ReportProductLocale ()

**ReportAuthorLocale**
Returns the author locale.
Syntax:
ReportAuthorLocale ()

**ReportSaveDate**
Returns the date when the report was last saved.
Syntax:
ReportSaveDate ()

**ReportCreateDate**
Returns the date when the report was created.
Syntax:
ReportCreateDate ()

**ReportID**
Returns the report ID.
Syntax:
ReportID ()

**ReportOutput**
Returns the name of the output format, such as CSV, HTML, HTMLFragment,layoutDataXML, MHT, PDF, rawXML, spreadsheetML (Excel 2007 format),XHTML, xlsxData (Excel 2007 Data format), XLWA (Excel 2002 format), XML,singleXLS (deprecated), XLS (deprecated).
Syntax:
ReportOutput ()

**ReportOption**
Returns the value of the run option variable identified by "optionName", such as attachmentEncoding, burst, cssURL, email, emailAsAttachment, emailAsURL,
emailBody, emailSubject, emailTo, emailToAddress, metadataModel, outputEncapsulation, outputFormat, outputLocale, outputPageDefinition,
outputPageOrientation, primaryWaitThreshold, print, printer, printerAddress, prompt, promptFormat, saveAs, saveOutput, secondaryWaitThreshold,
verticalElements, or xslURL.
Syntax:
ReportOption (’optionName’)

**ServerName**
Returns the name of the web server where the run request originated from. The value may be empty if the request is executed from the scheduler.
Syntax:

ServerName ()

**ServerLocale**
Returns the locale of the server that runs the report.
Syntax:
ServerLocale ()

**ModelPath**
Returns the model path.
Syntax:
ModelPath ()

**ParamNames**
Returns all parameter names.
Syntax:
ParamNames ()

**ParamName**
Returns the parameter name of "parameterName".
Syntax:
ParamName (’parameterName’)

**ParamDisplayValue**
Returns a string that is the parameter display value of "parameterName".
Syntax:
ParamDisplayValue (’parameterName’)

**ParamValue**
Returns the parameter value of "parameterName".
Syntax:
ParamValue (’parameterName’)

**ParamCount**
Returns the parameter count of "parameterName".
Syntax:
ParamCount (’parameterName’)

**RowNumber**
Returns the current row.
Syntax:
RowNumber ()

**PageNumber**
Returns the current page number.
Syntax:
PageNumber ()

**PageCount**
Returns the current page count. This function works only when the report output is Adobe® PDF or Microsoft® Excel. If you save the report output, this function works for all formats.
Syntax:
PageCount ()

**IsPageCountAvailable**
Returns Boolean 1 (true) if the page count is available for the current execution of the report; otherwise, returns Boolean 0 (false).
Syntax:
IsPageCountAvailable ()

**HorizontalPageNumber**
Returns the current horizontal page number.
Syntax:
HorizontalPageNumber ()

**HorizontalPageCount**
Returns the current horizontal page count.
Syntax:
HorizontalPageCount ()

**PageName**
Returns the current page name.
Syntax:
PageName ()

**URLEncode**
Returns the URL encoded value of the input text.
Syntax:
URLEncode ('text')

**TOCHeadingCount**
Returns the table of contents heading count for a specified heading level.
Syntax:
TOCHeadingCount ( headingLevel )

**IsAccessible**
Returns Boolean 1 (true) if the report is run with the accessibility features enabled.Use this function as a variable expression with a conditional block to make yourreports accessible. For example, you can add a list or crosstab equivalent to a chartin reports that are run with accessibility features enabled.
Syntax:
IsAccessible()

**IsBooklet**
Returns Boolean 1 (true) if the report is a child report of a booklet; otherwise,returns Boolean 0 (false).
Syntax:
IsBooklet()

**ColumnNumber**
Returns the current column number.
Syntax:
ColumnNumber ()

**IsCrosstabRowNodeMember**
Returns Boolean 1 (true) if the current node is a crosstab row node member.
Syntax:
IsCrosstabRowNodeMember ()

**IsCrosstabColumnNodeMember**
Returns Boolean 1 (true) if the current node is a crosstab column node member.
Syntax:
IsCrosstabColumnNodeMember ()

**IsInnerMostCrosstabRowNodeMember**
Returns Boolean 1 (true) if the current node is an innermost crosstab row node member.
Syntax:
IsInnerMostCrosstabRowNodeMember ()

**IsInnerMostCrosstabColumnNodeMember**
Returns Boolean 1 (true) if the current node is an innermost crosstab column node member.
Syntax:
IsInnerMostCrosstabColumnNodeMember ()

**IsOuterMostCrosstabRowNodeMember**
Returns Boolean 1 (true) if the current node is an outermost crosstab row node member.
Syntax:
IsOuterMostCrosstabRowNodeMember ()

**IsOuterMostCrosstabColumnNodeMember**
Returns Boolean 1 (true) if the current node is an outermost crosstab column node member.
Syntax:
IsOuterMostCrosstabColumnNodeMember ()

**IsFirstColumn**
Returns Boolean 1 (true) if the current column is the first column.
Syntax:
IsFirstColumn ()

**IsLastColumn**
Returns Boolean 1 (true) if the current column is the last column.
Syntax:
IsLastColumn ()

## Data type casting functions

**_add_days**
Returns the datetime resulting from adding "integer_expression" days to "timestamp_expression".
Syntax:
_add_days ( timestamp_expression , integer_expression )
Example:
_add_days ( 2007-01-14 00:00:00.000 , 3 )
Result:
2007-01-17 00:00:00.000

**_add_months**
Returns the datetime resulting from adding "integer_expression" months to "timestamp_expression".
Syntax:
_add_months ( timestamp_expression , integer_expression )

**_add_years**
Returns the datetime resulting from adding "integer_expression" years to "timestamp_expression".
Syntax:
_add_years ( timestamp_expression , integer_expression )

**_age**
Returns a number by subtracting "timestamp_expression" from today's date.
Syntax:
_age ( timestamp_expression )
Example:
_age ([Query1].[Date]), where [Query1].[Date] is March 2, 2004, and today is July 8, 2009
Result:
50,406, where 5 is the number of years, 04 is the number of months, and 06 is the number of days.

**_day_of_week**
Returns the day of the week (between 1 and 7) for "timestamp_expression" where "integer_expression" indicates which day of that week is day 1. To determine "integer_expression", choose the day of the week and count from Monday; for example, if you choose Wednesday, "integer_expression" would be 3 because Wednesday is the third day from Monday.
Syntax:
_day_of_week ( timestamp_expression , integer_expression )
Example:
_day_of_week ( 2009-01-01 , 7 ), where 7 means that Sunday is the first day of the week.
Result:
5

**_day_of_year**
Returns the ordinal for the day of the year in "timestamp_ expression" (1 to 366). Also known as Julian day.
Syntax:
_day_of_year ( timestamp_expression )

**_days_between**
Returns a positive or negative number representing the number of days between "timestamp_expression1" and "timestamp_expression2". If "timestamp_expression1"< "timestamp_expression2", the result will be a negative number.
Syntax:

_days_between ( timestamp_expression1 , timestamp_expression2)

**_days_to_end_of_month**
Returns a number representing the number of days remaining in the month represented by "timestamp_expression".
Syntax:
_days_to_end_of_month ( timestamp_expression )

**_first_of_month**
Returns a datetime that is the first day of the month represented by "timestamp_expression".
Syntax:
_first_of_month ( timestamp_expression )
Example 1:
_first_of_month ( 2009-05-04 00:00:00.000 )
Result:
Returns 2009-05-01 00:00:00.000
Example 2:
_first_of_month (current_date)
Result:
Returns Jul 1, 2009 if the current date is July 30, 2009.

**_last_of_month**
Returns a datetime that is the last day of the month represented by "timestamp_expression".
Syntax:
_last_of_month ( timestamp_expression )

**_make_timestamp**
Returns a timestamp constructed from "integer_expression1" (the year), "integer_expression2" (the month), and "integer_expression3" (the day). The time portion defaults to 00:00:00.000 .
Syntax:
_make_timestamp ( integer_expression1 , integer_expression2 , integer_expression3 )

**_months_between**
Returns a positive or negative number representing the number of months between "timestamp_expression1" and "timestamp_expression2". If "timestamp_expression1"
< "timestamp_expression2", the result will be a negative number.
Syntax:
_months_between ( timestamp_expression1 , timestamp_expression2 )

**_week_of_year**
Returns the week number (1-53) of the year for "timestamp_expression". According to the ISO 8601, week 1 of the year is the first week to contain a Thursday, which is equivalent to the first week containing January 4th. A week starts on a Monday (day 1) and ends on a Sunday (day 7).
Syntax:
_week_of_year ( timestamp_expression )

**_years_between**
Returns a positive or negative integer representing the number of years between"timestamp_expression1" and "timestamp_expression2". If "timestamp_expression1"
< "timestamp_expression2", a negative value is returned.
Syntax:
_years_between ( timestamp_expression1 , timestamp_expression2 )

**_ymdint_between**
Returns a number representing the difference between "timestamp_expression1" and "timestamp_expression2". This value has the form YYMMDD, where YY represents the number of years, MM represents the number of months, and DD represents the number of days.
Syntax:
_ymdint_between ( timestamp_expression1 , timestamp_expression2 )
Example:
_ymdint_between ( [Query1].[Date (close date)] , [Query1].[Date (ship date)] ), where [Query1].[Date (close date)] is February 20, 2004, and [Query1].[Date (ship date)] is January 19, 2004.
Result:
101, where 1 is the number of months and 01 is the number of days.

**abs**
Returns the absolute value of "numeric_expression". If "numeric_expression" is negative, a positive value is returned.
Syntax:
abs ( numeric_expression )

**ceiling**
Returns the smallest integer that is greater than or equal to "numeric_expression".
Syntax:
ceiling ( numeric_expression )

**character_length**
Returns the number of characters in "string_expression".
Syntax:
character_length ( string_expression )

**date2string**
Returns a date as a string in YYYY-MM-DD format.
Syntax:
date2string ( date_expression )

**date2timestamp**
Converts "date_expression" to a timestamp. The time part of the timestamp will equal zero.
Syntax:
date2timestamp ( date_expression )

**date2timestampTZ**
Converts "date_expression" to a timestamp with a time zone. The time and time zone parts of the timestamp will equal zero.
Syntax:
date2timestampTZ ( date_expression )

**DTinterval2string**
Returns a date time interval as a string in DDDD HH:MM:SS.FFFFFFF or -DDDD HH:MM:SS.FFF format.
Syntax:
DTinterval2string ( date_time_interval_expression )

**DTinterval2stringAsTime**
Returns a date time interval as a string in HHHH:MM:SS.FFFFFFF or

HH:MM:SS.FFF format. Days are converted to hours.
Syntax:
DTinterval2stringAsTime ( date_time_interval_expression )

**exp**
Returns the constant 'e' raised to the power of "numeric_expression". The constant 'e' is the base of the natural logarithm.
Syntax:
exp ( numeric_expression )
Example:
exp ( 2 )
Result:
7.389056

**extract**
Returns an integer representing the value of "date_part_expression" in "datetime_expression". "Date_part_expression" could be the year, month, day, hour, minute, or second.
Syntax:
extract ( date_part_expression , datetime_expression )
Example 1:
extract ( year , 2003-03-03 16:40:15.535 )
Result:
2003
Example 2:
extract ( hour , 2003-03-03 16:40:15.535 )
Result:
16

**floor**
Returns the largest integer that is less than or equal to "numeric_expression".
Syntax:
floor ( numeric_expression )

**int2DTinterval**
Converts an integer to a date time interval. "String_expression" specifies what "integer_expression" represents: "ns" = nanoseconds, "s" = seconds (default), "m" =
minutes, "h" = hours, "d" = days.
Syntax:
int2DTinterval ( integer_expression , string_expression )
Example 1:
int2DTinterval (1020,"h")
Result:
42 days 12 hours
Example 2:
int2DTinterval (1020,"s")
Result:
17 minutes

**int2YMinterval**
Converts "integer_expression" to a year month interval. "String_expression" specifies what "integer_expression" represents: "y" = years, "m" = months (default).
Syntax:
int2YMinterval ( integer_expression , string_expression )

**ln**
Returns the natural logarithm of "numeric_expression".
Syntax:
ln ( numeric_expression )

**lower**
Returns "string_expression" with all uppercase characters converted to lowercase.
Syntax:
lower ( string_expression )

**mapNumberToLetter**
Adds "integer_expression" to "string_expression".
Syntax:
mapNumberToLetter ( string_expression , integer_expression )
Example:
mapNumberToLetter ( 'a' , 1 )
Result:
B

**mod**
Returns an integer value representing the remainder (modulo) of "integer_expression1" / "integer_expression2".
Syntax:
mod ( integer_expression1 , integer_expression2 )

**nullif**
Returns null if "string_expression1" equals "string_expression2" (case-insensitive), otherwise returns "string_expression1".
Syntax:
nullif ( string_expression1 , string_expression2 )

**number2string**
Converts "numeric_expression" to a string, using the %g format specifier (C/C++ syntax).
Syntax:
number2string ( numeric_expression )

**octet_length**
Returns the number of bytes in "string_expression".
Syntax:
octet_length ( string_expression )

**position**
Returns the integer value representing the starting position of "string_expression1" in "string_expression2".
Returns 0 if "string_expression1" is not found.
Syntax:
position ( string_expression1 , string_expression2 )

**power**
Returns "numeric_expression1" raised to the power of "numeric_expression2".
Syntax:
power ( numeric_expression1 , numeric_expression2 )
Example:

power ( 3 , 2 )
Result:
9

**round**
Returns "numeric_expression" rounded to the nearest value with "integer_expression" significant digits to the right of the decimal point. If "integer_expression" is negative, "numeric_expression" is rounded to the nearest absolute value with "integer_expression" significant digits to the left of the decimal point. Rounding takes place before data formatting is applied.
Syntax:
round ( numeric_expression , integer_expression )
Example:
round (125, -1)
Result:
130

**sqrt**
Returns the square root of "numeric_expression". "Numeric_expression" must not be a negative value.
Syntax:
sqrt ( numeric_expression )

**status**
Returns the status of "expression". Possible values are: 0 - OK, 1 - null, 2 – not available, 4 - divide by zero, 8 - overflow, 16 - security, 32 - error, 64 - new, 128 - sample, 256 - pending.
Syntax:
status ( expression )

**string2date**
Returns "string_expression" as a date in YYYY-MM-DD format.
Syntax:
string2date ( string_expression )

**string2double**
Returns a floating point number. "String_expression" has the following form:
"[whitespace] [sign] [digits] [digits] [ {d|D|e|E }[sign]digits]"
Syntax:
string2double ( string_expression )

**string2DTinterval**
Returns "string_expression" as a date time interval in [-]DD HH:MM[:SS[.FFF]] format.
Syntax:
string2DTinterval ( string_expression )

**string2int32**
Returns an integer. "String_expression" has the following form: "[whitespace]
[{+|-}] [digits]"
Syntax:
string2int32 ( string_expression )

**string2int64**
Returns a long integer. "String_expression" has the following form: "[whitespace] [{+|-}] [digits]"
Syntax:
string2int64 ( string_expression )

**string2time**
Returns "string_expression" as a time in HH:MM:SS.FFFFFFF format.
Syntax:
string2time ( string_expression )

**string2timestamp**
Returns "string_expression" as a timestamp in YYYY-MM-DD [T|t|[white space]+] HH:MM:SS.FFFFFFF format.
Syntax:
string2timestamp ( string_expression )

**string2timestampTZ**
Returns "string_expression" in YYYY-MM-DD HH:MM:SS.FFFFFFF +HHMM or YYYY-MM-DD [T|t] HH:MM:SS.FFF -HHMM format.
Syntax:
string2timestampTZ ( string_expression )

**string2YMinterval**
Returns "string_expression" as a Year Month Interval in [-]YY MM format.
Syntax:
string2YMinterval ( string_expression )

**substring**
Returns the substring of "string_expression" that starts at position "integer_expression1" for "integer_expression2" characters or to the end of "string_expression" if "integer_expression2" is -1. The first character in "string_expression" is at position 1.
Syntax:
substring ( string_expression , integer_expression1 , integer_expression2 )
Example:
substring ( [Sales (analysis)].[Sales staff].[Sales staff].[Sales staff].[Position code], 3 , 5 )
Result:
Returns characters 3 to 7 of the position codes.

**time2string**
Returns a time as a string in HH:MM:SS.FFF format.
Syntax:
time2string ( time_expression )

**timestamp2date**
Converts "timestamp_expression" to a date. The time part of the timestamp will be ignored.
Syntax:
timestamp2date ( timestamp_expression )

**timestamp2string**
Returns a timestamp as a string in YYYY-MM-DD HH:MM:SS.FFFFFFF format.
Syntax:
timestamp2string ( timestamp_expression )

**timestamp2timestampTZ**
Converts "timestamp_expression" to a timestamp with a time zone. The displacement part of the timestamp with the time zone will be zero.
Syntax:

timestamp2timestampTZ ( timestamp_expression )

**timestampTZ2date**
Converts "timestamp_time_zone_expression" to a date. The time and time zone parts of the timestamp will be ignored.
Syntax:
timestampTZ2date ( timestamp_time_zone_expression )

**timestampTZ2string**
Returns a timestamp with the time zone as a string in YYYY-MM-DD HH:MM:SS.FFFFFFF +HHMM or YYYY-MM-DD HH:MM:SS.FFF -HHMM format.
Syntax:
timestampTZ2string ( timestamp_time_zone_expression )

**timestampTZ2timestamp**
Converts "timestamp_time_zone_expression" to a timestamp. The displacement part of the timestamp with the time zone will be ignored.
Syntax:
timestampTZ2timestamp ( timestamp_time_zone_expression )

**timeTZ2string**
Returns a time with the time zone as a string in HH:MM:SS.FFF +HHMM or HH:MM:SS.FFFFFFF -HHMM format. For example, -05:30 means a timezone of GMT minus 5 hours and 30 minutes
Syntax:
timeTZ2string ( timeTZ_expression )

**trim**
Returns "string_expression" trimmed of any leading and trailing blanks or trimmed of the character specified by "match_character_expression". "Trim_what_expression" may be "leading", "trailing", or "both" (default). "Match_character_expression" can be an empty string to trim blanks or can specify a character to be trimmed.
Syntax:
trim ( trim_what_expression , match_character_expression , string_expression )

**upper**
Returns "string_expression" with all lowercase characters converted to uppercase.
Syntax:
upper ( string_expression )

**YMinterval2string**
Returns "year_month_interval_expression" as a string in (YY MM) or -(YY MM) format.
Syntax:
YMinterval2string ( year_month_interval_expression )

# Appendix B – Common errors

If an error is displayed, more information may be available by clicking the “Details” link on the error page.

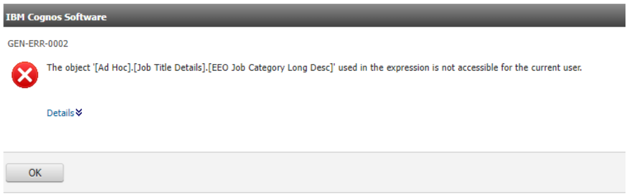
### Expression is not Accessible for the Current User

Indications:
Error message will include the statements like: [AdHoc].{QuerySubject_Name}.{field_name} used in the expression is not accessible for the current user.

Causes:
This is expected behavior if the user does not have permission to view the type of data included in the report. The user is trying to run a report which includes data from a function they do not have at least view permission to in Manager Services. A user cannot run a report with data they do not have permission to see in Manager Services.

Fixes:
The user must be granted at least view access to the function in Manager Services that corresponds to the data in the error. After function security is changed, the user must re-login to Manager Services to see the change take effect in HCA Reporting.
To see which query subjects are secured to which functions in Manager Services, you can run a report on Ad Hoc > Application > Ad Hoc Function Security List in HCA Ad Hoc.
If the user already has permission to the function, try removing function security for that user, saving, then re-granting permission to reset the security.

Example Screenshot:
In the sample screen shot below the user needs permission to the Job Titles function before the report will run.



IBM Cognos Software

GEN-ERR-0002

The object '[Ad Hoc].[Job Title Details].[EEO Job Category Long Desc]' used in the expression is not accessible for the current user.

### Firewall Security Rejection

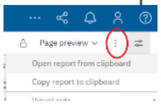
Indications:
Error message stating Firewall Security Rejection will appear when trying open HCA Reporting or trying to save or edit a report.

Causes:
This is an error from the Cognos application. If Cognos does not receive a request from a valid, known Cognos session, it will block it. This error means a Cognos session does not exist, has been lost, timed out or replaced. Cognos does not want an unknown session accessing it.
This can happen for a few reasons:
-If HCA Reporting is open in a browser window and in another window Manager Services has timed out or has been logged into again, the old Cognos session will be invalidated.
-A browser window with HCA Reporting has been left open for an extended period of time and the Cognos session has been timed out.
-This can also occur when trying to launch HCA Reporting in Manager Services. In that scenario, it means the seamless login to Cognos that happens at Manager Services login has failed. HCA items will also be

missing from the Manager Services Tools menu in this situation. The most likely reason this is because the user’s Active Directory account is locked out or the password needs to be reset.

Fixes:
Close all browser windows and login to Manager Services again.

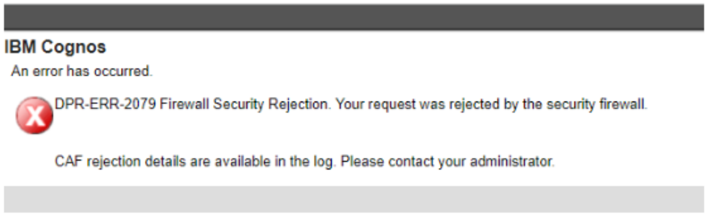
If this happens and the user needs to save the current report, one only option is to do this:
In HCA Reporting go to the “More” menu and select “Copy report to clipboard”.



Close that browser and log in to Manager Services again and create a blank report. Then got to the “More” and select “Open report from clipboard” and you should now have the same report back open and then it can be saved or saved as.

If the message appears when trying to launch HCA Reporting, it is possible the user’s account is locked out or passwords are out of sync. The user can try resetting their password in Manager Services. The password should not contain parts of their username or an ampersand, and they should not re-use a password they used previously in the system. The user should login to Manager Servies again after the password change. If launching HCA Reporting still fails to launch, contact iCON Support.

Example Screenshot:



IBM Cognos

An error has occurred.

DPR-ERR-2079 Firewall Security Rejection. Your request was rejected by the security firewall.

CAF rejection details are available in the log. Please contact your administrator.

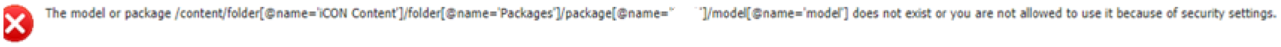
**Model or Package does not exist or you are not allowed to use it because of security settings**
**or**
**No access to property “metadataModelPackage”**

Indications:
Error message stating:
“Model or Package /content/folder[@name=’iCON Content’]/folder[@name=’Packages’]/package[@name={customer number}] does not exist or you are not allowed to use it because of security settings.”
Or
“You do not have access to the property “metadataModelPackage” when opening a report.

Causes:
This is expected behavior if the user tries to open a report designed from the package of a customer number different than the one currently logged into in Manager Services. This also applies to Live versus Test as these are each different customer numbers. You can only run reports designed for the customer you are logged into.

Fixes:
To run a report designed under a different customer number, the package of the report must be changed to the customer number you are currently logged into. See Setting the package for a report.

Example Screenshot:



```
The model or package /content/folder[@name='iCON Content']/folder[@name='Packages']/package[@name='      ']/model[@name='model'] does not exist or you are not allowed to use it because of security settings.
```

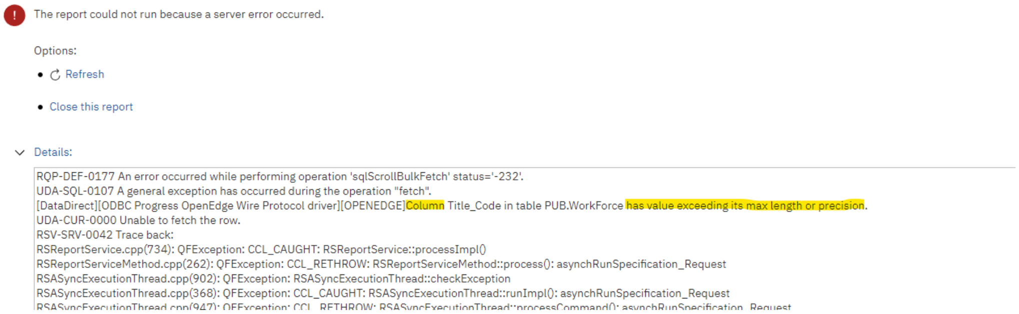
### Value Exceeding its Max Length or Precision

Indications:
Error message will include the statement: [DataDirect][ODBC Progress Driver][Progress]Column {column-name} in table {table-name} has a value exceeding its max length or precision.

Causes:
This is a database issue that must be addressed by UHRO.

Fixes:
Contact iCON Support.

Example Screenshot:



The report could not run because a server error occurred.

Options:

- Refresh
- Close this report

Details:

```
RQP-DEF-0177 An error occurred while performing operation 'sqlScrollBulkFetch' status='-232'.
UDA-SQL-0107 A general exception has occurred during the operation "fetch".
[DataDirect][ODBC Progress OpenEdge Wire Protocol driver][OPENEDGE]Column Title_Code in table PUB.WorkForce has value exceeding its max length or precision.
UDA-CUR-0000 Unable to fetch the row.
RSV-SRV-0042 Trace back:
RSReportService.cpp(734): QFException: CCL_CAUGHT: RSReportService::processImpl()
RSReportServiceMethod.cpp(262): QFException: CCL_RETHROW: RSReportServiceMethod::process(): asynchRunSpecification_Request
RSASyncExecutionThread.cpp(902): QFException: RSASyncExecutionThread::checkException
RSASyncExecutionThread.cpp(368): QFException: CCL_CAUGHT: RSASyncExecutionThread::runImpl(): asynchRunSpecification_Request
RSASyncExecutionThread.cpp(947): QFException: CCL_RETHROW: RSASyncExecutionThread::processCommand(): asynchRunSpecification_Request
```

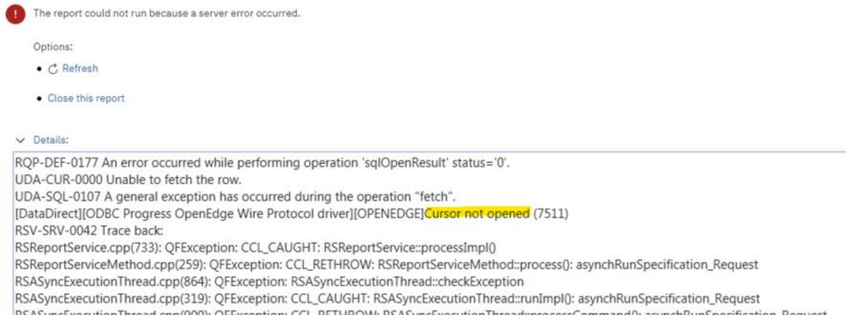
### Cursor Not Opened

Indications:
Error message will include the statement: "[DataDirect][ODBC Progress OpenEdge Wire Protocol driver][OPENEDGE]Cursor not opened".

Causes:
The user attempted to run or edit a report and the error appeared. This is an error from the Progress database and not directly related to Cognos or the report. The Progress database is busy or short on resources so the request to run a query from Cognos was not completed. This typically happens with more complex or data intensive reports. Progress has refused this attempt at retrieving the data.

Fixes:
Click the Refresh link on the error page to resubmit the query or switch to design view and try running the report again. It should work on future attempts. If not, try simplifying the report or remove fields until you identify the field causing the error to appear.

Example Screen Shot:



The report could not run because a server error occurred.

Options:

- Refresh
- Close this report

Details:

```
RQP-DEF-0177 An error occurred while performing operation 'sqlOpenResult' status='0'.
UDA-CUR-0000 Unable to fetch the row.
UDA-SQL-0107 A general exception has occurred during the operation "fetch".
[DataDirect][ODBC Progress OpenEdge Wire Protocol driver][OPENEDGE]Cursor not opened (7511)
RSV-SRV-0042 Trace back:
RSReportService.cpp(733): QFException: CCL_CAUGHT: RSReportService::processImpl()
RSReportServiceMethod.cpp(259): QFException: CCL_RETHROW: RSReportServiceMethod::process(): asynchRunSpecification_Request
RSASyncExecutionThread.cpp(864): QFException: RSASyncExecutionThread::checkException
RSASyncExecutionThread.cpp(319): QFException: CCL_CAUGHT: RSASyncExecutionThread::runImpl(): asynchRunSpecification_Request
RSASyncExecutionThread.cpp(909): QFException: CCL_RETHROW: RSASyncExecutionThread::processCommand(): asynchRunSpecification_Request
```

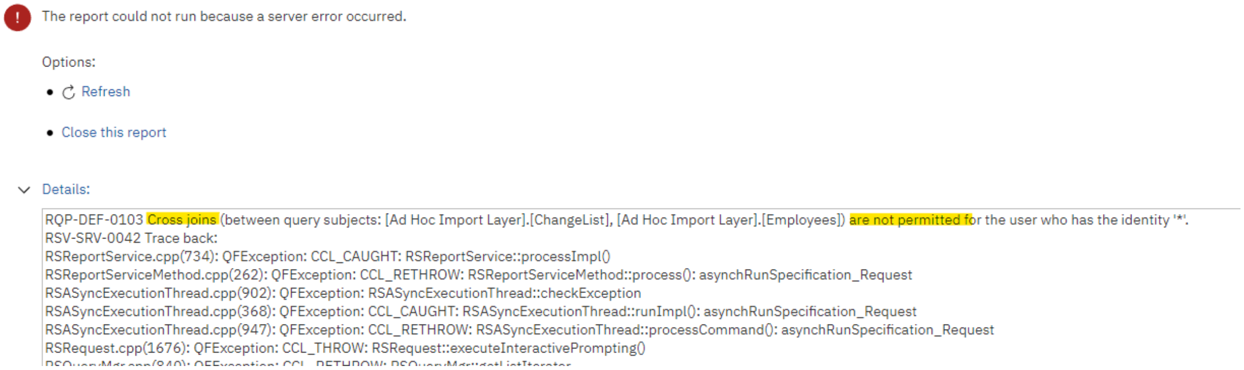
## Cross Join Error

Indications:
Error message will include the statement: “Cross joins between query subjects are not permitted”.

Causes:
The user attempted to use data from query subjects that are not related. There is no join between the data in the report model. This may lead to a massive, non-sensible report, so it is not allowed by Cognos.

Fixes:
Select data from different query subjects which do have a join or if it is believed a join is missing and should exist, contact iCON Support. Alternatively, users proficient in Cognos could create multiple queries in the report and the report author could define their own join logic for these queries.

Example Screenshot:



The report could not run because a server error occurred.

Options:

- Refresh
- Close this report

Details:

```
RQP-DEF-0103 Cross joins (between query subjects: [Ad Hoc Import Layer].[ChangeList], [Ad Hoc Import Layer].[Employees]) are not permitted for the user who has the identity '*'.
RSV-SRV-0042 Trace back:
RSReportService.cpp(734): QFException: CCL_CAUGHT: RSReportService::processImpl()
RSReportServiceMethod.cpp(262): QFException: CCL_RETHROW: RSReportServiceMethod::process(): asynchRunSpecification_Request
RSASyncExecutionThread.cpp(902): QFException: RSASyncExecutionThread::checkException
RSASyncExecutionThread.cpp(368): QFException: CCL_CAUGHT: RSASyncExecutionThread::runImpl(): asynchRunSpecification_Request
RSASyncExecutionThread.cpp(947): QFException: CCL_RETHROW: RSASyncExecutionThread::processCommand(): asynchRunSpecification_Request
RSRequest.cpp(1676): QFException: CCL_THROW: RSRequest::executeInteractivePrompting()
RSQueryMgr.cpp(840): QFException: CCL_RETHROW: RSQueryMgr::getListIterator
```