SACAD1

SAP Analytics Cloud: Analytics Designer

EXERCISES AND SOLUTIONS

Course Version: 30
Course Duration: 7 Hours 55 Minutes
Material Number: 50158872

SAP Copyrights, Trademarks and Disclaimers

Typographic Conventions

American English is the standard used in this handbook.

The following typographic conventions are also used.

This information is displayed in the instructor's presentation	
Demonstration	
Procedure	
Warning or Caution	
Hint	
Related or Additional Information	
Facilitated Discussion	
User interface control	*Example text*
Window title	*Example text*

Contents

Unit 1
Exercise 1

Use Data Analyzer

Business Example

Some users ask you to quickly analyze your order figures for different products over the past several years. You have access to queries in your SAP BW system that can provide this information, but you have not built a dashboard or an application. Your SAP Analytics Cloud admin introduces you to the Data Analyzer application. You can use it to easily visualize data from SAP BW queries, SAP HANA Live views, and SAP Analytics Cloud models.

In this exercise, you first learn how to access and prepare the training landscape you will use throughout the course. In later steps, you use the Data Analyzer to quickly analyze the data from an SAP BW query without the need to build a model or write complex scripts. You will save the navigation state and filter settings by creating an insight.

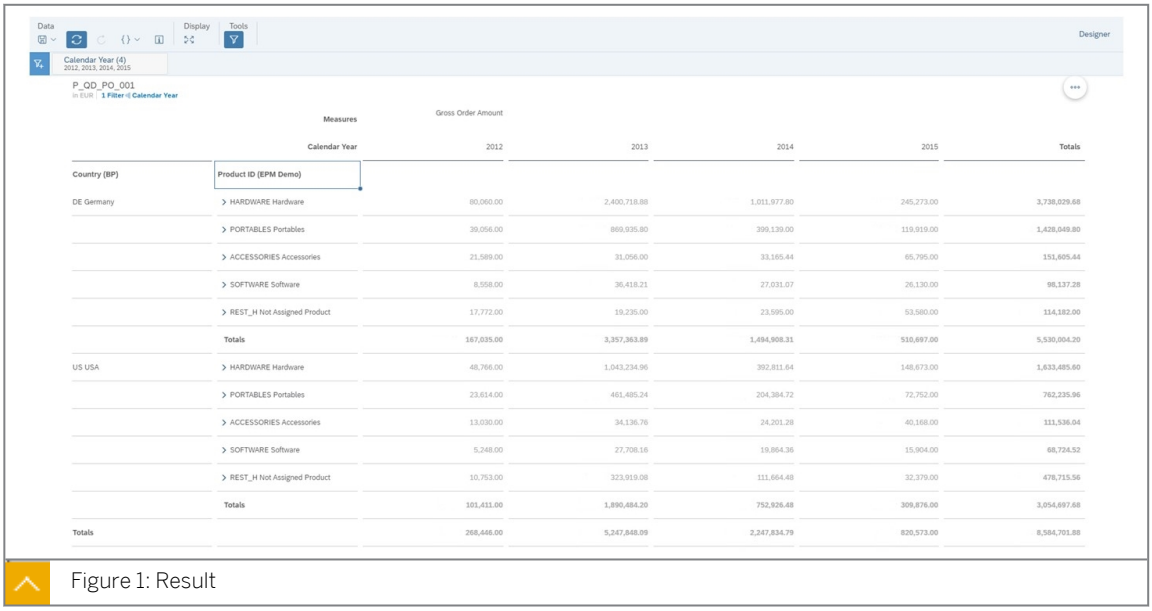


Figure 1: Result

Task 1: Run the Course Initialization Script

1. Run the SAC initialization script so that the course folder is copied to your local environment, in *N:\SAC\SACAD1*.

   The initialization script is available on the Windows Start page.

2. Add the course files folder *N:\SAC\SACAD1* to your Favorites so that it is easy to reach in future.

Task 2: Access SAP Analytics Cloud

1. Open Chrome in the Remote Desktop environment and access the SAP Analytics Cloud Tenant URL provided by your instructor. Log into the system using **`A##`** or **`B##`** for the *User* and **`Welcome1`** for the *Password*.

   Substitute **`##`** with your assigned user number.

Task 3: Launch the Data Analyzer and select a Data Source

1. Launch the Data Analyzer application from the Navigation Bar. Use **`SACAD1BW4`** as the *Connection* and BW Query **`P_QD_PO_001`** as the *Data Source*. Use the default query variable values when prompted.

Task 4: Explore the Data Analyzer Features

1. Access the *Available items panel*.
2. Configure the table to show the measure *Gross Order Amount* and the dimension *Calendar Year* in columns. Show the dimension *Country (BP)* in rows.

   Notice how the table is automatically updated to reflect the selections made.
3. Configure Data Analyzer for manual refresh. Add the *Product ID (EPM Demo)* dimension to the rows and refresh.
4. Set the *Product ID (EPM Demo)* to display in a hierarchy. Display the *Product Hierarchy 01* hierarchy.
5. Add a filter to only show values from the years 2012 - 2015.

Task 5: Create and Use an Insight

1. Save the current navigational state and filter settings as an insight. Make sure that the prompt dialog always opens when you open your saved insight.

   Use the details as shown in the following table:

Field	Value
Name	**`GR###_Insight_1`**
Description	**`Insight_1`**

2. Open the insight from folder *My Files* and only select *USA* in the prompt dialog.

Unit 1
Solution 1

Use Data Analyzer

Business Example

Some users ask you to quickly analyze your order figures for different products over the past several years. You have access to queries in your SAP BW system that can provide this information, but you have not built a dashboard or an application. Your SAP Analytics Cloud admin introduces you to the Data Analyzer application. You can use it to easily visualize data from SAP BW queries, SAP HANA Live views, and SAP Analytics Cloud models.

In this exercise, you first learn how to access and prepare the training landscape you will use throughout the course. In later steps, you use the Data Analyzer to quickly analyze the data from an SAP BW query without the need to build a model or write complex scripts. You will save the navigation state and filter settings by creating an insight.

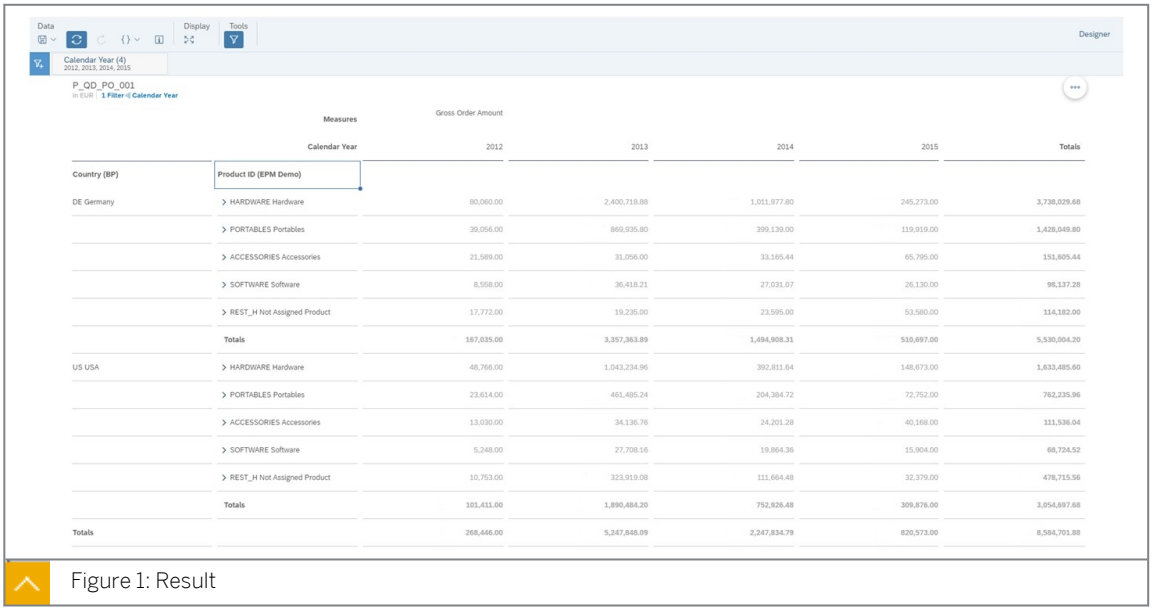


Figure 1: Result

Task 1: Run the Course Initialization Script

1. Run the SAC initialization script so that the course folder is copied to your local environment, in *N:\SAC\SACAD1*.

   The initialization script is available on the Windows Start page.

   a) In the Remote Desktop environment, choose the *Windows Start* button.

   b) Select the *Initialize Course* tile.

   c) Double-click the *SAC* folder.

   d) To execute the script, double-click the *Initialize_SAC* shortcut.

   e) At the next prompt, choose *Yes*.

   f) Double-click the *SACAD1* folder to see the content required for the course.

2. Add the course files folder *N:\SAC\SACAD1* to your Favorites so that it is easy to reach in future.
   a) Within the content of the *N:\SAC\SACAD1* folder displayed in Windows Explorer, in the left pane, in the context menu of the *Favorites* icon, choose *Add current location to Favorites*.
   Alternatively, you can drag the *SACAD1* folder from the local *N:\SAC* drive to *Favorites*.

Task 2: Access SAP Analytics Cloud

1. Open Chrome in the Remote Desktop environment and access the SAP Analytics Cloud Tenant URL provided by your instructor. Log into the system using **A##** or **B##** for the *User* and **Welcome1** for the *Password*.
   Substitute **##** with your assigned user number.
   a) In the Remote Desktop environment, choose the *Google Chrome* icon in the taskbar.
   b) Access SAP Analytics Cloud using the URL provided by your instructor.

> Note:
> The course's SAP Analytics Cloud system is configured to use a custom SAML Identity Provider, which is only accessible in the training landscape. You cannot access the URL outside of the training environment.

   c) Specify the information for authentication.
   Use the details as shown in the following table:

Field	Value
User	**A##** or **B##**
Password	**Welcome1**

Task 3: Launch the Data Analyzer and select a Data Source

1. Launch the Data Analyzer application from the Navigation Bar. Use **SACAD1BW4** as the *Connection* and BW Query **P_QD_PO_001** as the *Data Source*. Use the default query variable values when prompted.
   a) From the Navigation Bar choose  *Data Analyzer* .
   b) Choose *From a Data Source*.
   c) In the *Select Data Source* window, choose the *Connection* dropdown and choose *SACAD1BW4*.
   d) In the *Select Data Source* window, choose  to open the list of BW queries.
   e) In the *Search* bar, type **demo** and choose the *Search* icon.
   f) Select *P_QD_PO_001* from the search results.
   g) Choose *OK* on the *Select Data Source* window.

h) In the *Set Variables for P_QD_PO_001* window, leave the default variable values of *Germany* and *USA* for the *Country_BP* variable prompt.

i) Choose *Set*.

Task 4: Explore the Data Analyzer Features

1. Access the *Available items panel*.
   a) Open the *Designer* panel.
2. Configure the table to show the measure *Gross Order Amount* and the dimension *Calendar Year* in columns. Show the dimension *Country (BP)* in rows.
   a) In the *Available Items* panel, deselect the *Net Order Amount* measure.
   b) In the *Available Items* panel, choose the *Add dimension to columns* icon for the *Calendar Year* dimension.
   c) Confirm the *Country (BP)* dimension is visible in the table rows.

   Notice how the table is automatically updated to reflect the selections made.
3. Configure Data Analyzer for manual refresh. Add the *Product ID (EPM Demo)* dimension to the rows and refresh.
   a) In the *Data*-section of the toolbar, choose the *Auto Refresh* icon to disable it.
   b) In the *Available Items* panel, choose the *Add dimension to rows* icon for *Product ID (EPM Demo)* dimension. Notice that table is not automatically updated.
   c) In the *Data*-section of the toolbar, choose the *Refresh* icon to see the *Product ID (EPM Demo)* dimension in the table rows.
   d) Enable the *Auto Refresh* option.
4. Set the *Product ID (EPM Demo)* to display in a hierarchy. Display the *Product Hierarchy 01* hierarchy.
   a) Open the context menu for *Product ID (EPM Demo)* by performing a right mouse click on the cell containing *Product ID (EPM Demo)*.
   b) In the context menu, choose the *Select Hierarchy* option.
   c) In the *Select Hierarchy* window, choose *Product Hierarchy 01* from the dropdown.
   d) Choose *Set*.
5. Add a filter to only show values from the years 2012 - 2015.
   a) Choose the *Add Story Filter/Prompt* icon.
   b) Choose the *Calendar Year* dimension.
   c) In the *Set Filters for Calendar Year* window, choose *2012*, *2013*, *2014*, and *2015*.
   d) Choose *OK*.
   The table columns only show the selected years.

Task 5: Create and Use an Insight

1. Save the current navigational state and filter settings as an insight. Make sure that the prompt dialog always opens when you open your saved insight.

   Use the details as shown in the following table:

Field	Value
Name	**GR###_Insight_1**
Description	**Insight_1**

   a) In the upper left corner of the Data Analyzer, choose  *Save* and then *Save*.

   b) In the *Save Insight* dialog, in the folder *My Files* enter the name and description according to the table.

   c) In the *Advanced Options* section of the *Save Insight* dialog select *Automatically open prompt when insight opens.*and choose *OK*.

> Note:
> After saving an insight, the insight name is displayed in the header of the Data Analyzer. Also notice the generated URL that includes the insight ID.

2. Open the insight from folder *My Files* and only select *USA* in the prompt dialog.

   a) Choose  *Expand Navigation Bar* and choose  *Files* to open your insight *GR###_Insight_1*.

   b) In the *Set Variables for P_QD_PO_001* dialog, delete *Germany* and choose *Set* .

> Note:
> Only results of USA are shown in the output.

Unit 2
Exercise 2

Create Your First Application

Business Example

In your new role as application designer, you are asked to develop your first application. The application has three areas: Headline, KPI-Tiles, and Data Visualization. The data comes from a live SAP BW-system connection and from an acquired data source. You use various data-bound and non data-bound widgets.

> Note:
> As this is the first application that we build, the design itself is not important. In upcoming exercises, we will take a closer more detailed look at the design.

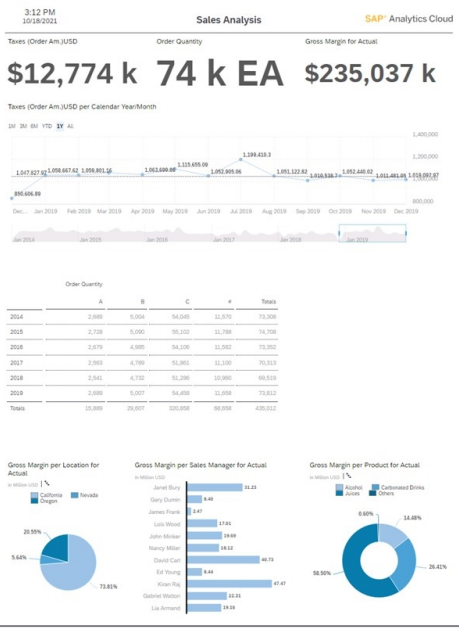


Figure 2: Result

Task 1: Create the Application

1. Create a new *Analytic Application* and save it with the name **`GR### My First Application`** directly into the *My Files* content area.

Task 2: Create the Headline

1. Add a *Clock* widget to the top left corner of your application. Do not display a logo in the clock. Change the name of the widget to **`CLOCK_Head`**.
2. Add the centered headline text **`Sales Analysis`** using a *Text* widget and define a border at the bottom. Change the name of the widget to **`TX_Headline`**.

3. Add an *Image* widget to the top right corner of your application. Change the name of the widget to **`IMG_Logo`**. Use the available *SAPC-Header_image.png* image in *N:\SAC \SACAD1*.

Task 3: Create the KPI-Tiles

1. Create the first KPI-Tile.

   Use the details as shown in the following table:

Field	Value
KPI-tile	*Numeric Point Chart*
Measure	*Taxes (Order Am.)USD*
Data source	*SACAD1_Model001*
Folder	*Public → SACAD1_30 → SACAD1 Content*
Filter year	*2019*
Widget name	**`CH_Tile_Tax`**

   Define the styling and layout as shown in the following figure.

Figure 3: Example KPI Tile design

> Note:
> The *SACAD1_Model001* uses a live connection to a BW Query in an SAP BW/4HANA system. The authorization to the BW system is maintained through SSO so you do not get a user/password prompt in our training system.

2. Create a second KPI-Tile.

   Use the details as shown in the following table:

Field	Value
KPI-tile	*Numeric Point Chart*
Measure	*Order Quantity*
Data source	*SACAD1_Model001*
Filter year	*2019*
Widget name	**`CH_Tile_Quantity`**

   The layout and design are similar to the previous KPI-Tile. Copy and paste the previous one and change the settings to show the data needed.

3. Create a third KPI-Tile as a copy of *CH_Tile_Quantity*.

   Use the details as shown in the following table:

Field	Value
KPI-tile	*Numeric Point Chart*
Measure	*Gross Margin*
Data source	*SACAD1_Model002*
Widget name	**`CH_Tile_Margin`**

Note:
The *SACAD1_Model002* is using an acquired (offline) data set.

Task 4: Create the Time Series Chart

1. Below the KPI-Tile area you want to present a Time Series Chart with the name **`CH_TaxesPerMonth`**. This presents the measure *Taxes (Order Am.)USD* per month from the previously-used data model *SACAD1_Model001*. To add more meaning to the data presented in the chart, you use a dynamic average as reference line. Make sure your chart setting results in a chart like the following:

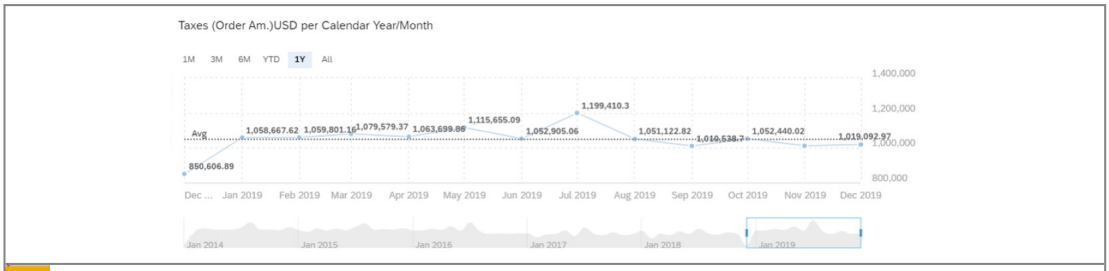


Figure 4: Final Design for Time Series Chart

Task 5: Create a Table

1. Below the Time Series Chart, add a Table.
   Use the details as shown in the following table:

Field	Value
Measure	*Order Quantity*
Rows	*Years*
Columns	*ABC-Ranking*
Table design	*Report-Styling*
Data source	*SACAD1_Model001*
Widget name	**`TBL_Quantity`**

   Ensure that you use **`TBL_Quantity`** as a name for the table.

Task 6: Create a Pie Chart, Bar Chart and Donut Chart

1. Below the Table add a Pie Chart.
   Use the details as shown in the following table:

Field	Value
Chart	*Pie*
Data source	*SACAD1_Model002*
Folder	*Public → SACAD1_30 → SACAD1 Content*
Measure	*Gross Margin*
Color Dimension	*Location*
Widget name	**`CH_GrossMarginPerLocation`**

2. Next to the Pie Chart add a Bar Chart.
   Use the details as shown in the following table:

Field	Value
Chart	*Bar*
Data source	*SACAD1_Model002*
Folder	*Public → SACAD1_30 → SACAD1 Content*
Chart Orientation	*Horizontal*
Measure	*Gross Margin*
Dimension	*Sales Manager*
Widget name	**`CH_GrossMarginPerSalesManager`**

3. Next to the Bar Chart, add a Donut Chart.
   Use the details as shown in the following table:

Field	Value
Chart	*Donut*
Data source	*SACAD1_Model002*
Folder	*Public → SACAD1_30 → SACAD1 Content*
Measure	*Gross Margin*
Dimension	*Product*
Widget name	**`CH_GrossMarginPerProduct`**

Task 7: Run and Test Your Application

1. Save the application you created and run it. In the Time Series Chart, explore the end-user capabilities to display a longer or shorter time period.
2. Explore how your application appears with different screen sizes. To test this, leave the full screen view of your browser window and simulate different screen sizes. What happens if your shrink the browser window? Does that behavior match to what we have designed? Does that behavior match to what you would expect as an end user?

3. Return to the Application Designer by switching to the other browser tab. Use the *Preview Device Toolbar* to simulate different sizes, for example, an iPad screen size. Rotate the orientation between portrait and landscape view.

Unit 2
Solution 2

Create Your First Application

Business Example

In your new role as application designer, you are asked to develop your first application. The application has three areas: Headline, KPI-Tiles, and Data Visualization. The data comes from a live SAP BW-system connection and from an acquired data source. You use various data-bound and non data-bound widgets.

> Note:
> As this is the first application that we build, the design itself is not important. In upcoming exercises, we will take a closer more detailed look at the design.

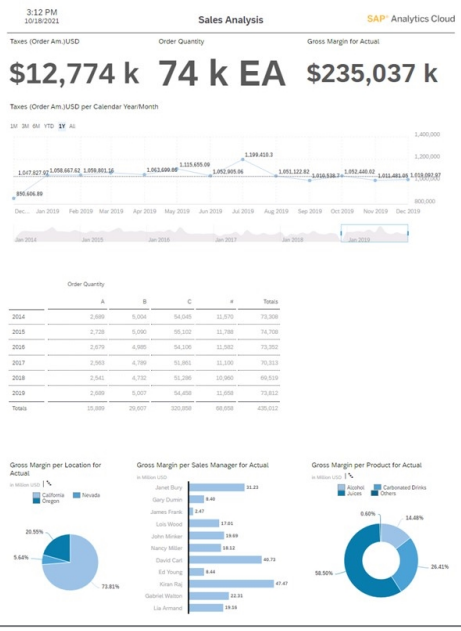


Figure 2: Result

Task 1: Create the Application

1. Create a new *Analytic Application* and save it with the name **`GR### My First Application`** directly into the *My Files* content area.

   a) From the Navigation Bar choose  *Analytic Applications* and then click *Application* in the *Create New* area.

   b) In the toolbar, choose the *Save* icon and choose *Save*.

   c) Type in **`GR### My First Application`** as the name.

d) Choose *OK*.

Task 2: Create the Headline

1. Add a *Clock* widget to the top left corner of your application. Do not display a logo in the clock. Change the name of the widget to **`CLOCK_Head`**.

   a) In the *Insert*-section of the toolbar, choose the + *Add* icon and *More Widgets* and choose *Clock*.

   b) Open the *Designer* panel.

   c) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`CLOCK_Head`**
Size and Position	
Width	**`160`**
Height	**`60`**
Left	**`30`**
Top	**`30`**
Clock Properties	
Show Logo	**`deactivate`** the switch
Time Format	choose **`1:00 PM`**
Date Format	choose **`2/20/2023`**

2. Add the centered headline text **`Sales Analysis`** using a *Text* widget and define a border at the bottom. Change the name of the widget to **`TX_Headline`**.

   a) In the *Insert*-section of the toolbar, choose the + *Add* icon and choose *Text*.

   b) Choose the text field and type in the title **`Sales Analysis`**.

   c) In the *Styling* section of the *Designer*, choose the settings for the*Text* widget as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`TX_Headline`**
Size and Position	
Width	**`1024`**
Height	**`40`**
Left	**`30`**

Field	Value
Top	**50**
Widget	
Border	**Bottom Border**

d) On the Canvas, select the text **Sales Analysis** and in the *Styling* section of the *Designer*, choose the settings as shown in the following table:

Field	Value
Font	
Text type	**Header 1**
Size	**22**
Alignment	**Center**

3. Add an *Image* widget to the top right corner of your application. Change the name of the widget to **IMG_Logo**. Use the available *SAPC-Header_image.png* image in *N:\SAC\SACAD1*.

a) In the *Insert*-section of the toolbar, choose the + *Add* icon and choose *Image*.

b) Choose *(Upload Image)*, select the provided image *SAC-Header_image.png* in *N:\SAC\SACAD1*, choose *Open* and then *Insert*.

c) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**IMG_Logo**
Size and Position	
Width	**200 px**
Height	**32 px**
Left	**854 px**
Top	**48 px**

Task 3: Create the KPI-Tiles

1. Create the first KPI-Tile.
Use the details as shown in the following table:

Field	Value
KPI-tile	*Numeric Point Chart*
Measure	*Taxes (Order Am.)USD*

Field	Value
Data source	*SACAD1_Model001*
Folder	*Public → SACAD1_30 → SACAD1 Content*
Filter year	*2019*
Widget name	**`CH_Tile_Tax`**

Define the styling and layout as shown in the following figure.

Figure 3: Example KPI Tile design

> Note:
> The *SACAD1_Model001* uses a live connection to a BW Query in an SAP BW/4HANA system. The authorization to the BW system is maintained through SSO so you do not get a user/password prompt in our training system.

a) In the *Insert*-section of the toolbar, choose the *Chart* icon.

b) Choose the model *SACAD1_Model001* that is located in the folder *Public → SACAD1_30 → SACAD1 Content*.

c) In the *Builder* panel, choose the settings as shown in the following table:

Field	Value
Chart Structure	
Chart Type	**`Indicator → Numeric Point`**
Primary Values	Choose *Add Measure* and select **`Taxes (Order Am.)USD`**.
Filters	Choose **`Delivery Date → Calendar Year`**. Select **`2019`** and choose *OK*.

d) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`CH_Tile_Tax`**
Quick Menus	
Visible in the Runtime	**`deactivate`** the switch
Size and Position	

Field	Value
Width	**340**
Height	**160**
Left	**30**
Top	**100**
Number Format	
Scale	**Thousand**
Scale Format	**k, m, bn**
Decimal Places	**0**

e) Choose the newly created chart element in the *Canvas* area and choose the *More Actions* icon.

f) Deactivate the subtitle by choosing *Show/Hide* → *Subtitle*.

g) Deactivate the *Primary Value Labels* by choosing *Show/Hide* → *Primary Value Labels*.

h) Deactivate the visualization of the filter by choosing *Show/Hide* → *Chart Details* → *Filter*.

2. Create a second KPI-Tile.

Use the details as shown in the following table:

Field	Value
KPI-tile	*Numeric Point Chart*
Measure	*Order Quantity*
Data source	*SACAD1_Model001*
Filter year	*2019*
Widget name	**CH_Tile_Quantity**

The layout and design are similar to the previous KPI-Tile. Copy and paste the previous one and change the settings to show the data needed.

a) Choose the element *CH_Tile_Tax* in the *Canvas* area and choose *Copy&Paste* → *Duplicate* from the toolbar.

b) In the *Builder* panel, change the settings as shown in the following table:

Field	Value
Chart Structure	

Field	Value
Primary Values	Choose the icon to remove the entry *Taxes (Order Am.)USD*. Choose *Add Measure* and select **`Order Quantity`**.

c) In the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`CH_Tile_Quantity`**
Size and Position	
Left	370
Top	100

3. Create a third KPI-Tile as a copy of *CH_Tile_Quantity*.
Use the details as shown in the following table:

Field	Value
KPI-tile	*Numeric Point Chart*
Measure	*Gross Margin*
Data source	*SACAD1_Model002*
Widget name	**`CH_Tile_Margin`**

> Note:
> The *SACAD1_Model002* is using an acquired (offline) data set.

a) Choose the element *CH_Tile_Quantity* in the *Canvas* area and choose *Copy&Paste* → *Duplicate* from the toolbar.

b) In the *Builder* panel, change the settings as shown in the following table:

Field	Value
Data Source	
Data Source	Choose *Change primary model* and choose *OK*. Choose *Select other model...* Search for model **`SACAD1_Model002`** in the folder *Public* → *SACAD1_30* → *SACAD1 Content* and select it.

Field	Value
Chart Structure	
Primary Values	Choose *Add Measure* and select **`Gross Margin`**.

c) In the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`CH_Tile_Margin`**
Size and Position	
Left	**`710 px`**
Top	**`100 px`**

Task 4: Create the Time Series Chart

1. Below the KPI-Tile area you want to present a Time Series Chart with the name **`CH_TaxesPerMonth`**. This presents the measure *Taxes (Order Am.)USD* per month from the previously-used data model *SACAD1_Model001*. To add more meaning to the data presented in the chart, you use a dynamic average as reference line. Make sure your chart setting results in a chart like the following:

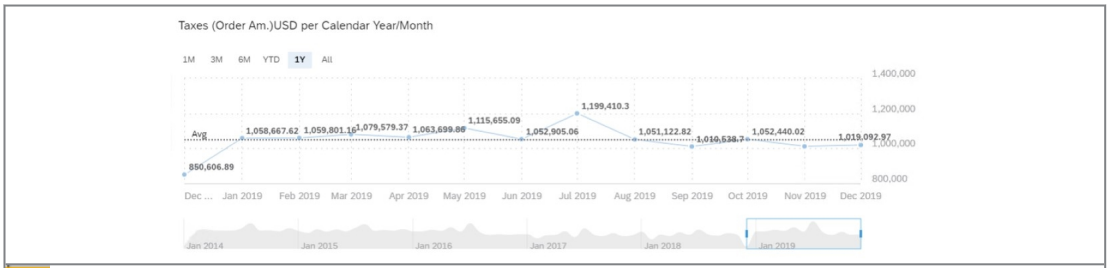


Figure 4: Final Design for Time Series Chart

a) Insert a new chart widget to the canvas.

b) Choose the icon to change the data source for the new chart and select **`SACAD1_Model001`** from the list of models.

c) Choose *OK*.

d) In the *Builder* panel, choose the settings as shown in the following table:

Field	Value
Chart Structure	
Chart Type	**`Trend → Time Series`**
Measure	**`Taxes (Order Am.)USD`**
Time	**`Delivery Date → Calendar Year/Month`**

e) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`CH_TaxesPerMonth`**
Size and Position	
Width	**`1024 px`**
Height	**`350 px`**
Left	**`30 px`**
Top	**`250 px`**

f) In the canvas area, deactivate the subtitle by choosing *More Actions* → *Show/Hide* → *Subtitle*.

g) In the canvas area, deactivate the visualization of the drill state by choosing *More Actions* → *Show/Hide* → *Chart Details* → *Drill*.

h) In the canvas area, activate the reference line by choosing *More Actions* → *Add* → *Reference Line*.

i) In the *Create Reference Line* dialog, choose the type *Dynamic*. Select *Taxes (Order Am.)USD* as *Measure* and *Average* as *Aggregation*.

j) Choose *OK*.

Task 5: Create a Table

1. Below the Time Series Chart, add a Table.

   Use the details as shown in the following table:

Field	Value
Measure	*Order Quantity*
Rows	*Years*
Columns	*ABC-Ranking*
Table design	*Report-Styling*
Data source	*SACAD1_Model001*
Widget name	**`TBL_Quantity`**

   Ensure that you use **`TBL_Quantity`** as a name for the table.

   a) In the *Insert*-section of the toolbar, choose the *Table* icon.

   b) In the *Builder* panel, ensure that *SACAD1_Model001* is selected as Data Source. If it is not selected, change it to that model and choose the settings as shown in the following table:

Field	Value
Table Structure	

Field	Value
Adaptive Column Width	**`Selected`**
Rows	*Calendar Year* should be selected by default
Columns	In the *Measures* block, choose the filter icon, select **`Order Quantity`** and choose *OK*. Add the dimension **`NAV: Product ABC-Rat`** to the columns (expand **`Product`** to select it).

c) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`TBL_Quantity`**
Size and Position	
Width	**`640 px`**
Height	**`370 px`**
Left	**`30 px`**
Top	**`660 px`**
Table Properties	
Template	**`Report-Styling`**

d) Choose *OK* in the *Warning* dialog about switching the template.

e) In the canvas area, deactivate the visualization of *Table Title* by choosing *More Actions* → *Show/Hide* → *Table Title*.

f) In the canvas area, deactivate the visualization of *Subtitle* by choosing *More Actions* → *Show/Hide* → *Subtitle*.

g) In the canvas area, deactivate the visualization of *Table Details* by choosing *More Actions* → *Show/Hide* → *Table Details*.

Task 6: Create a Pie Chart, Bar Chart and Donut Chart

1. Below the Table add a Pie Chart.
   Use the details as shown in the following table:

Field	Value
Chart	*Pie*
Data source	*SACAD1_Model002*
Folder	*Public* → *SACAD1_30* → *SACAD1 Content*

Field	Value
Measure	*Gross Margin*
Color Dimension	*Location*
Widget name	**CH_GrossMarginPerLocation**

a) In the *Insert*-section of the toolbar, choose the *Chart* icon.

b) Choose the icon to change the data source for the new chart and select **SACAD1_Model002** from the list of models.

c) In the *Builder* panel, choose the settings as shown in the following table:

Field	Value
Chart Structure	
Chart Type	**More → Pie**
Measures	Choose *Add Measure* and select **Gross Margin**.
Color	Choose *Add Dimension* and select **Location**

d) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**CH_GrossMarginPerLocation**
Size and Position	
Width	**270 px**
Height	**380 px**
Left	**30 px**
Top	**1070 px**

2. Next to the Pie Chart add a Bar Chart.

Use the details as shown in the following table:

Field	Value
Chart	*Bar*
Data source	*SACAD1_Model002*
Folder	*Public → SACAD1_30 → SACAD1 Content*
Chart Orientation	*Horizontal*
Measure	*Gross Margin*
Dimension	*Sales Manager*

Field	Value
Widget name	**CH_GrossMarginPerSalesManager**

a) In the *Insert*-section of the toolbar, choose the *Chart* icon.

b) Make sure that model *SACAD1_Model002* is selected.

c) In the *Builder* panel, choose the settings as shown in the following table:

Field	Value
Chart Structure	
Chart Type	**Comparison → Bar/Column**
Chart Orientation	*Horizontal*
Measures	Choose *Add Measure* and select **Gross Margin**.
Dimensions	Choose *Add Dimension* and select **Sales Manager**

d) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**CH_GrossMarginPerSalesManager**
Size and Position	
Width	**380 px**
Height	**380 px**
Left	**320 px**
Top	**1070 px**

3. Next to the Bar Chart, add a Donut Chart.
Use the details as shown in the following table:

Field	Value
Chart	*Donut*
Data source	*SACAD1_Model002*
Folder	*Public → SACAD1_30 → SACAD1 Content*
Measure	*Gross Margin*
Dimension	*Product*
Widget name	**CH_GrossMarginPerProduct**

a) In the *Insert*-section of the toolbar, choose the *Chart* icon.

b) Make sure that model *SACAD1_Model002* is selected.

c) In the *Builder* panel, choose the settings as shown in the following table:

Field	Value
Chart Structure	
Chart Type	**`More → Donut`**
Measures	Choose *Add Measure* and select **`Gross Margin`**.
Color	Choose *Add Dimension* and select **`Product`**

d) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`CH_GrossMarginPerProduct`**
Size and Position	
Width	**`334 px`**
Height	**`380 px`**
Left	**`720 px`**
Top	**`1070 px`**

Task 7: Run and Test Your Application

1. Save the application you created and run it. In the Time Series Chart, explore the end-user capabilities to display a longer or shorter time period.

   a) In the *File*-section of the toolbar, choose the *Save* icon and choose *Save*.

   b) In the top-right corner, choose the *Run Analytic Application* button.
   A new tab in your browser session opens and shows your application.

   c) In the time series chart, use the time-range selector to display a longer or shorter time period.

2. Explore how your application appears with different screen sizes. To test this, leave the full screen view of your browser window and simulate different screen sizes. What happens if your shrink the browser window? Does that behavior match to what we have designed? Does that behavior match to what you would expect as an end user?

   a) Choose the  *Restore Down* icon of your browser window.

   b) Reduce the width of the window so that it is smaller than its content.

   c) Choose the *Maximize* icon of your browser window.

3. Return to the Application Designer by switching to the other browser tab. Use the *Preview Device Toolbar* to simulate different sizes, for example, an iPad screen size. Rotate the orientation between portrait and landscape view.

a) Switch to the other browser tab.

b) Use the *Device* setting at the bottom of the screen and select **`iPad`**. Check the behavior of the canvas area.

c) Choose the  *Rotate* icon . Check the behavior of the canvas area.

d) Use the *Device* setting at the bottom of the screen to switch it back to **`Auto`**.

Unit 2
Exercise 3

Use Navigation Options in your Application

Business Example

You want to add navigation possibilities to your first application that you created. A filter line and two input controls will be offered to end-users of the application to make selections on the table and linked analysis will be set up between the charts to automatically filter related charts when filtering a specific chart.

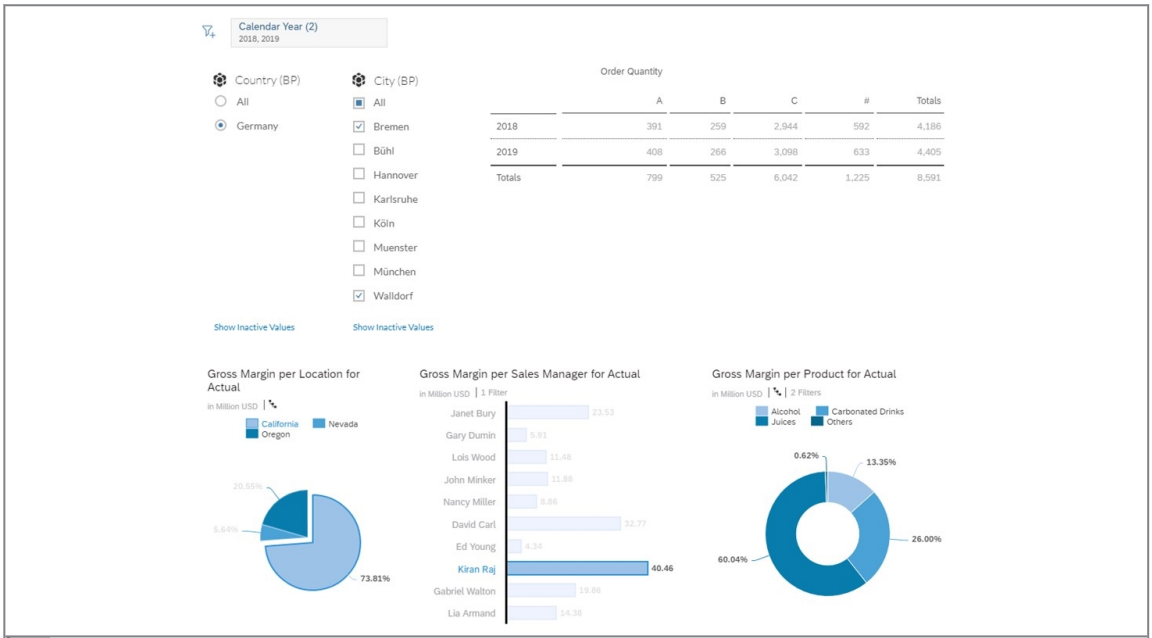


Figure 5: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### My First Application* and save it with the new name **`GR### Use navigation options in your application`**.

2. Alternatively you can create a copy of the application *GR000 My First Application* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Solutions* and save it with the name **`GR### Use navigation options in your application`** into the folder structure *My Files*.

Task 2: Create a Filter Line and Input Controls for the Table

1. Above the table add a *Filter Line* widget entitled **`FL_Quantity`** that enables navigation for the dimensions **`Calendar Year`**, **`Company Code`** and **`Organizational Unit`**. The widget should only filter data of the table **`TBL_Quantity`**.

2. First move table **`TBL_Quantity`** to the right to create space for the input controls to be created. Then create an *Input Control* widget entitled **`IP_Country`** on the left of it that

presents all entries of the dimension **`Country (BP)`**. The widget should enable single selection only. Ensure that all interactions with this widget only apply to the table **`TBL_Quantity`**. Also check the default enablement of the Cascading Effect.

3. On the left side of the table **`TBL_Quantity`** (and to the right of *Input Control* **`IP_Country`**) create an *Input Control* widget entitled **`IP_City`** that presents all entries of the dimension **`City (BP)`**. The widget should enable multiple selection. Ensure that all interactions with this widget only apply to the table **`TBL_Quantity`**.

Task 3: Create Linked Analysis between the Pie Chart, Bar Chart and Donut Chart

1. Selecting a part of the pie chart should automatically filter the bar chart and the donut chart based on the selection.
2. Selecting a bar on the bar chart should automatically filter the donut chart based on the selection. Use the Linked Widgets Diagram functionality to define it.
3. Use the Linked Widgets Diagram to display an overview of the filter line, the input controls and the linked analysis that you have set up between the 3 charts. Also show the widgets which are not linked.

Task 4: Run and Test Your Application

1. Save the application you created and run it. Explore the end-user capabilities of the filter line and the input controls related to the table presenting measure Order Quantity. Notice the cascading effect between the input control for dimension Country and the input control for dimension City.
2. Explore the measure Gross Margin using the linked analysis between the 3 charts, by first selecting a part of the pie chart, and then select a specific bar of the bar chart to see the donut chart updated accordingly. You have to analyze which Location has the highest contribution in Gross Margin, which Sales Manager has the highest Gross Margin in that Location and which Product has the highest Gross Margin for the Sales Manager in that Location.

Use Navigation Options in your Application

Business Example

You want to add navigation possibilities to your first application that you created. A filter line and two input controls will be offered to end-users of the application to make selections on the table and linked analysis will be set up between the charts to automatically filter related charts when filtering a specific chart.

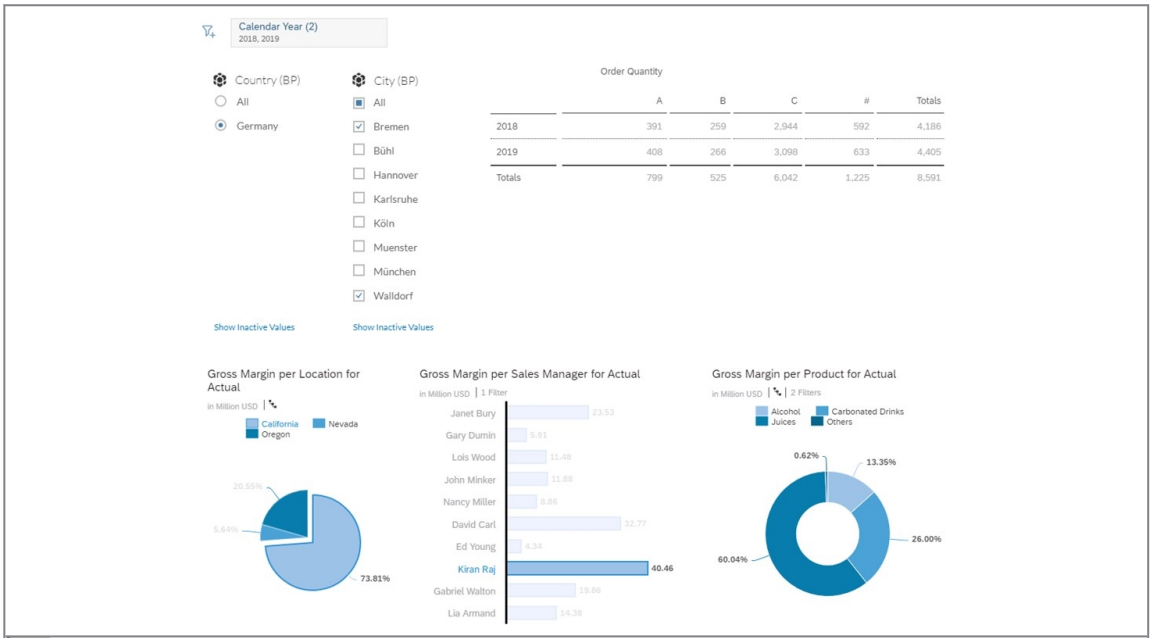


Figure 5: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### My First Application* and save it with the new name **`GR### Use navigation options in your application`**.

   a) In the application designer, switch to your already opened application *GR### My First Application*.

   b) In the toolbar, choose the *Save* icon and choose *Save as...*.

   c) Type in **`GR### Use navigation options in your application`** as the name and choose *OK*.

2. Alternatively you can create a copy of the application *GR000 My First Application* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Solutions* and save it with the name **`GR### Use navigation options in your application`** into the folder structure *My Files*.

a) From the Navigation Bar choose  *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Solutions*.

b) Select the checkbox for the application *GR000 My First Application* and choose  *Copy*.

c) Choose *My Files* and type in **`GR### Use navigation options in your application`**.

d) Choose *OK*.

e) Navigate to the *My Files* area.

f) Choose *GR### Use navigation options in your application* to open the file in edit mode.

Task 2: Create a Filter Line and Input Controls for the Table

1. Above the table add a *Filter Line* widget entitled **`FL_Quantity`** that enables navigation for the dimensions **`Calendar Year`**, **`Company Code`** and **`Organizational Unit`**. The widget should only filter data of the table **`TBL_Quantity`**.

   a) In the *Insert*-section of the toolbar, choose *Add* → *Filter Line*.

   b) In the *Builder* panel, choose the settings as shown in the following table:

Field	Value
Filter Line Structure	
Mode	**`Individual Widget Filter`**
Source Widget	**`TBL_Quantity`**
Dimension Selection	Add **`Calendar Year`**, **`Company Code`** and **`Organizational Unit`**

   c) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`FL_Quantity`**
Size and Position	
Width	**`1024`**
Height	**`60`**
Left	**`30`**
Top	**`600`**

2. First move table **`TBL_Quantity`** to the right to create space for the input controls to be created. Then create an *Input Control* widget entitled **`IP_Country`** on the left of it that presents all entries of the dimension **`Country (BP)`**. The widget should enable single

selection only. Ensure that all interactions with this widget only apply to the table **`TBL_Quantity`**. Also check the default enablement of the Cascading Effect.

a) In the *Styling* panel of **`TBL_Quantity`**, change size *Left* to **`420`**.

b) In the *Insert*-section of the toolbar, choose *Add* → *Input Control*.

c) In the upcoming dialog choose *Business Partner* → *Country (BP)*.

d) In the *Set Filters for Country (BP)* dialog select the *All Members* check box

e) In the *Settings for Users* area select **`Single Selection`**.

f) Click on *Ok* to close the dialog.

g) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`IP_Country`**
Size and Position	
Width	**`190`**
Height	**`370`**
Left	**`30`**
Top	**`660`**
Widget	
Border	**`No Border`**

h) In the *More Actions* menu of the widget select the *Linked Analysis* → *Settings* option.

i) Within the *Linked Analysis* pane on the right of the screen select the **`Only Selected Widgets`** option.

j) Click on the *Select Widgets* button

k) In the *Select Widgets* dialog select the check box next to **`TBL_Quantity`** and click the *OK* button to close the dialog again.

l) Click on the *Apply* button to activate your settings.

m) In the *More Actions* menu of the widget notice that the *Cascading Effect* option has been checked by default.

Note:
In the next step you will create *Input Control* **`IP_City`**. By using the cascading effect, only cities that belong to the selected country appear.

3. On the left side of the table **`TBL_Quantity`** (and to the right of *Input Control* **`IP_Country`**) create an *Input Control* widget entitled **`IP_City`** that presents all entries of

the dimension **`City (BP)`**. The widget should enable multiple selection. Ensure that all interactions with this widget only apply to the table **`TBL_Quantity`**.

a) In the *Insert*-section of the toolbar, choose *Add* → *Input Control*.

b) In the upcoming dialog choose *Business Partner* → *City (BP)*.

c) In the *Set Filters for City (BP)* dialog select the *All Members* check box.

d) In the *Settings for Users* area select **`Multiple Selection`**.

e) Click on *Ok* to close the dialog.

f) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`IP_City`**
Size and Position	
Width	**`190`**
Height	**`370`**
Left	**`220`**
Top	**`660`**
Widget	
Border	**`No Border`**

g) In the *More Actions* menu of the widget select the *Linked Analysis* → *Settings* option.

h) Within the *Linked Analysis* pane on the right of the screen select the **`Only Selected Widgets`** option.

i) Click on the *Select Widgets* button

j) In the *Select Widgets* dialog select the check box next to **`TBL_Quantity`** and click the *OK* button to close the dialog again.

k) Click on the *Apply* button to activate your settings.

Task 3: Create Linked Analysis between the Pie Chart, Bar Chart and Donut Chart

1. Selecting a part of the pie chart should automatically filter the bar chart and the donut chart based on the selection.

a) On the canvas, in the *More Actions* menu of the pie chart select *Linked Analysis* → *Settings*.

b) Within the *Linked Analysis* pane on the right of the screen select the **`Only Selected Widgets`** option.

c) Click on the *Select Widgets* button

d) In the *Select Widgets* dialog select the check boxes next to **`CH_GrossMarginPerProduct`** and **`CH_GrossMarginPerSalesManager`** and click the *OK* button to close the dialog again.

e) Within the *Linked Analysis* pane select the **`Filter on Data Point Selection`** option.

f) Click on the *Apply* button to activate your settings.

2. Selecting a bar on the bar chart should automatically filter the donut chart based on the selection. Use the Linked Widgets Diagram functionality to define it.

a) In the *Tools*-section of the toolbar, select the *Linked Widgets Diagram* icon.

b) On the *Linked Widgets Diagram* tab, in the *Show Diagram For* box, select *CH_GrossMarginPerSalesManager (Gross Margin)*

c) On the canvas right click the **`CH_GrossMarginPerSalesManager`** block and select *Manage Linked Widgets* .

d) Within the *Linked Widgets Diagram* pane on the right of your screen select *Add Target Widgets* and select **`CH_GrossMarginPerProduct`** .

e) Select the **`Filter on Data Point Selection`** option and press *Done*.

3. Use the Linked Widgets Diagram to display an overview of the filter line, the input controls and the linked analysis that you have set up between the 3 charts. Also show the widgets which are not linked.

a) On the *Linked Widgets Diagram* tab, in the *Show Diagram For* box, select *Overview* to see all linked analysis relationships in your application. Hover over a link between two widgets to see a tooltip indicating the relationship, either Linked Analysis, Filtering from Filter Line or Cascading Effect.

b) On the *Linked Widgets Diagram* tab, select *Include Unlinked Widgets* to see the other widgets, which are not linked.

Task 4: Run and Test Your Application

1. Save the application you created and run it. Explore the end-user capabilities of the filter line and the input controls related to the table presenting measure Order Quantity. Notice the cascading effect between the input control for dimension Country and the input control for dimension City.

a) In the *File*-section of the toolbar, choose the *Save* icon and choose *Save*.

b) In the top-right corner, choose the *Run Analytic Application* button.
A new tab in your browser session opens and shows your application.

c) Use the filter bar to restrict the displayed values of the table to the *Calendar Year* values **`2018`** and **`2019`**.

d) Use the *Input Controls* to filter on the country **`Germany`** and the two cities **`Bremen`** and **`Walldorf`**.
When you select country Germany, only German cities are presented due to the cascading effect.

2. Explore the measure Gross Margin using the linked analysis between the 3 charts, by first selecting a part of the pie chart, and then select a specific bar of the bar chart to see the donut chart updated accordingly. You have to analyze which Location has the highest

contribution in Gross Margin, which Sales Manager has the highest Gross Margin in that Location and which Product has the highest Gross Margin for the Sales Manager in that Location.

a) In the pie chart, click on the pie that presents **`California`**.
The bar chart and the donut chart are filtered. Sales Manager **`Kiran Raj`** has the highest Gross Margin in California (40.46 Million USD).

b) In the bar chart, click on the bar that presents **`Kiran Raj`**.
The donut chart is filtered. The Product **`Juices`** has the highest Gross Margin for **`Kiran Raj`** in **`California`** (60.04 %).

Unit 2
Exercise 4

Define the Layout of your Application

Business Example

Your end users request that the application should reflect the available width of the screen or device. They do not want to run into the situation of scrolling horizontally or vertically to be able to see all parts of the screen.

To achieve a cleaner design of the application, the main content should be presented in a tab strip layout.

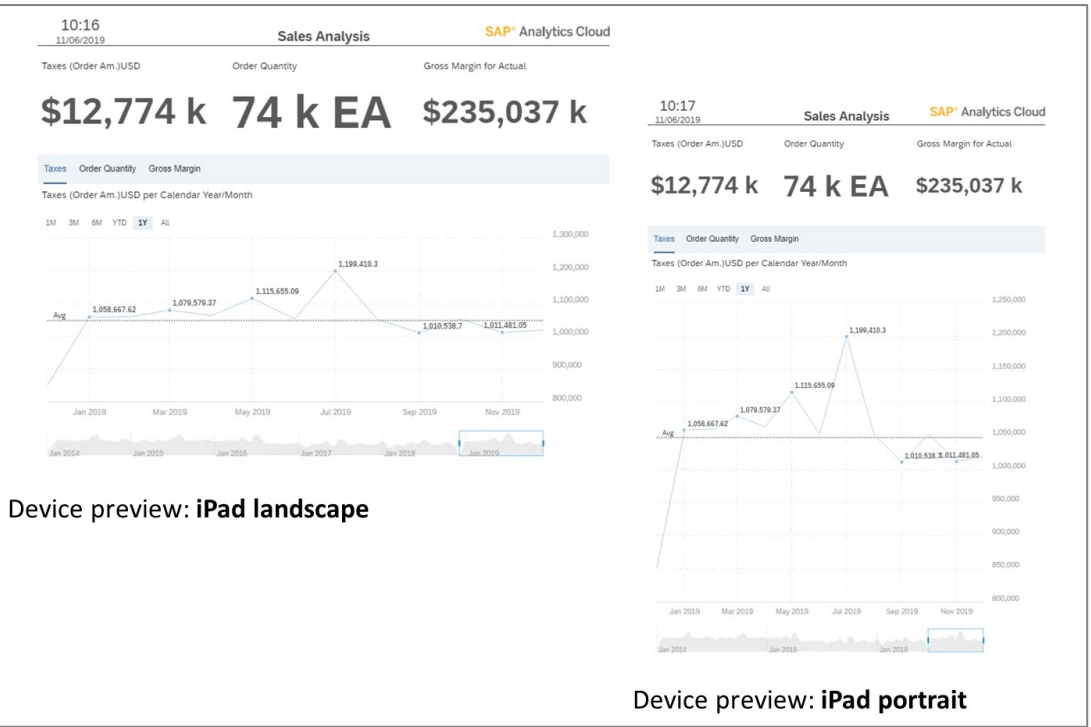


Figure 6: Result

After discussions with UX designers, the following principles regarding the arrangement of the elements should be applied. You will find more detailed information for each section of the application in each step description.

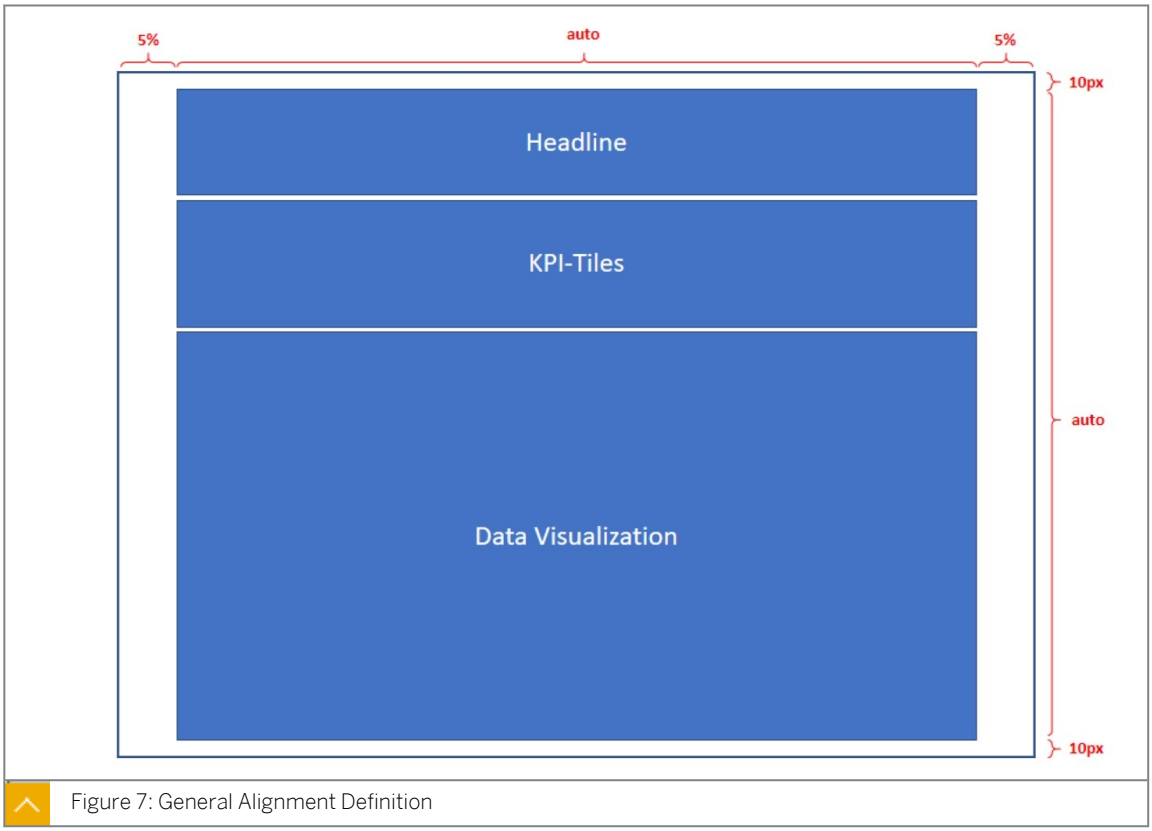


Figure 7: General Alignment Definition

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Use navigation options in your application* and save it with the new name **`GR### Define the layout of your application`**.
2. Alternatively you can create a copy of the application *GR000 Use navigation options in your application* which is available in the folder *Public → SACAD1_30 → SACAD1 Solutions* and save it with the name **`GR### Define the layout of your application`** into the folder structure *My Files*.

Task 2: Create the New Headline Section

1. Add a *Panel* widget to your application and name it **`PANEL_Header`**. The Headline should have a fixed height of **`80px`**. All other required sizing and position information can be taken from the general alignment definition mentioned in the exercise introduction.
2. Add the widgets *CLOCK_Head*, *TX_Headline*, *IMG_Logo* into the newly created panel widget. Assign the size and positioning settings so that the following requirements are fulfilled:
   - The clock covers **`15% of the width`** of the panel and has **`no left border`**. It has a **`fixed height of 80 px`** and **`no top border`**.
   - The text covers **`100% of the width`** of the panel. **`Height and top border are fixed with 32 px`**.
   - The logo is **`fixed to 200 px width`** and has **`no right border`**. The **`height is fixed with 32 px`** with a **`24 px border to the top`**.

Task 3: Create the New KPI-Tiles Section

1. Add a *Panel* widget to your application and name it **`PANEL_Tiles`**. The KPI-Tiles should have a fixed height of **`160px`** and should start **`10px`** underneath the *PANEL_Header*. All other required sizing and position information can be taken from the general alignment definition mentioned in the exercise introduction.
2. Add the widgets *CH_Tile_Tax*, *CH_Tile_Quantity*, *CH_Tile_Margin* into the newly created panel widget. Assign the size and positioning settings so that the following requirements are fulfilled:
   - Each KPI tile should cover **`100% of the available height`** of the panel.
   - Each KPI tile should cover **`33.3% of the available width`** so that they are positioned next to each other.

Task 4: Create the New Data Visualization Section

1. Add a *Tab Strip* widget to your application and name it **`TS_Content`**. The tab strip should use the available space starting 10 px below the KPI tiles and keeping a 10 px border to the bottom. All other required sizing and position information can be taken from the general alignment definition mentioned in the exercise introduction. The tab strip should contain three tabs labeled **`Taxes`**, **`Order Quantity`** and **`Gross Margin`**.
2. Place the chart *CH_TaxesPerMonth* into the *Taxes* tab. The chart should entirely cover the given space within the tab.
3. Place the filter line *FL_Quantity* into the *Order Quantity* tab. It should cover 100 % of the available width of the tab and first 60 px from the top.
4. Place the two input controls *IP_Country* and *IP_City* plus the table *TBL_Quantity* into the *Order Quantity* tab. All three widgets should cover the available height starting underneath the filter line to the bottom of the tab. The input control for countries is aligned to the left border of the tab and covers 190 px width. The input control for the cities is aligned 190px from the left border and covers 190px width. The table is aligned 380 px from the left border and covers the remaining width to the right border of the tab.
5. Add the widgets *CH_GrossMarginPerLocation*, *CH_GrossMarginPerSalesManager*, *CH_GrossMarginPerProduct* into the third tab. Assign the size and positioning settings so that the following requirements are fulfilled:
   - Each chart should cover **`100% of the available height`** of the panel.
   - Each chart should cover **`33.3% of the available width`** so that they are positioned next to each other.

Task 5: Save and Test your Application Design

1. Use the *Device* setting to simulate, for example, an iPad or an Laptop screen size. Rotate the orientation between portrait and landscape view.
2. Save your application and run it.

Define the Layout of your Application

Business Example

Your end users request that the application should reflect the available width of the screen or device. They do not want to run into the situation of scrolling horizontally or vertically to be able to see all parts of the screen.

To achieve a cleaner design of the application, the main content should be presented in a tab strip layout.

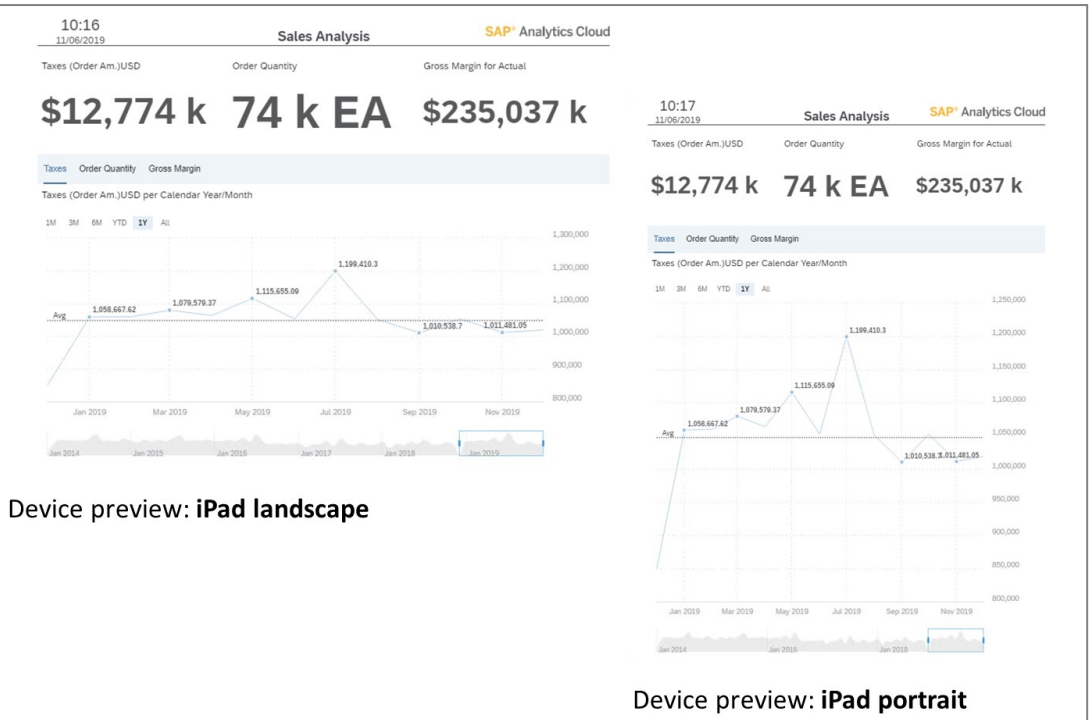


Figure 6: Result

After discussions with UX designers, the following principles regarding the arrangement of the elements should be applied. You will find more detailed information for each section of the application in each step description.

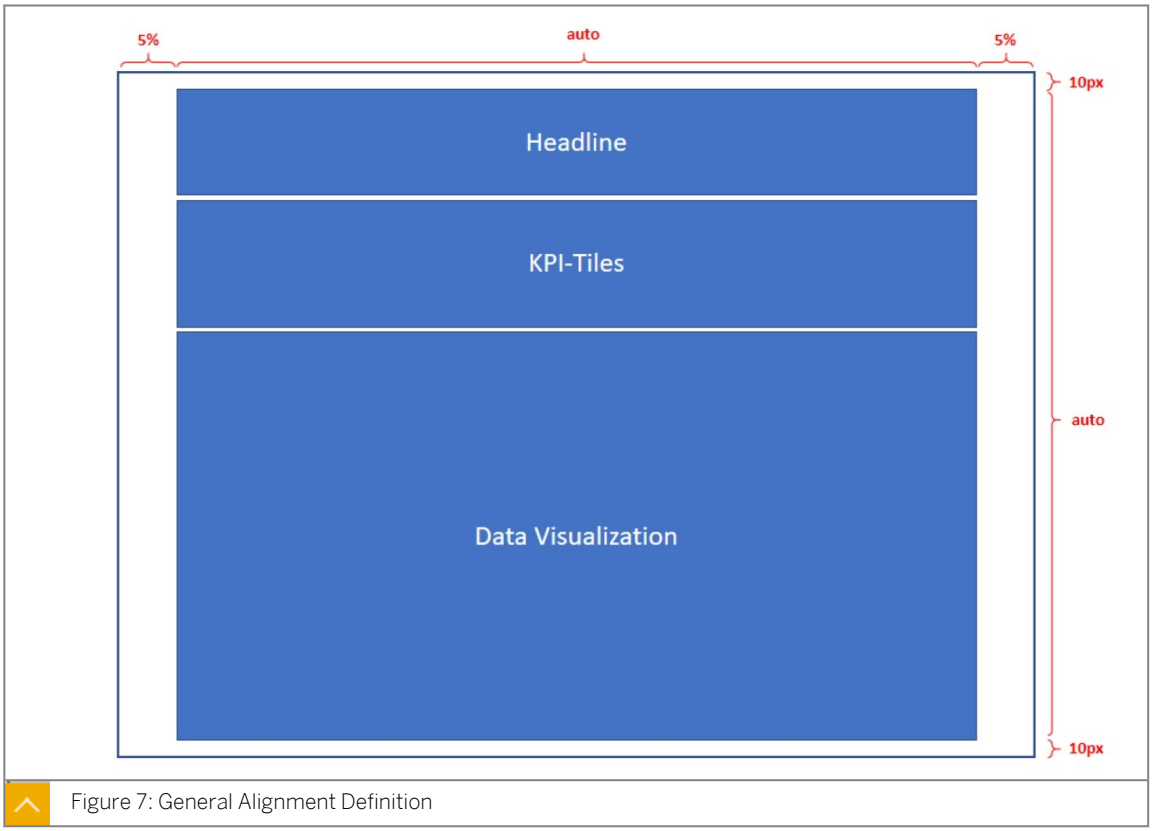


Figure 7: General Alignment Definition

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Use navigation options in your application* and save it with the new name **`GR### Define the layout of your application`**.
   a) In the application designer, switch to your already opened application *GR### Use navigation options in your application*.
   b) In the toolbar, choose the *Save* icon and choose *Save as...*.
   c) Type in **`GR### Define the layout of your application`** as the name and choose *OK*.
2. Alternatively you can create a copy of the application *GR000 Use navigation options in your application* which is available in the folder *Public → SACAD1_30 → SACAD1 Solutions* and save it with the name **`GR### Define the layout of your application`** into the folder structure *My Files*.
   a) From the Navigation Bar choose *Files → My Files → Public → SACAD1_30 → SACAD1 Solutions*.
   b) Select the checkbox for the application *GR000 GR### Use navigation options in your application* and choose *Copy*.
   c) Choose *My Files* and type in **`GR### Define the layout of your application`**.
   d) Choose *OK*.

e) Navigate to the *My Files* area.

f) Choose *GR### Define the layout of your application* to open the file in edit mode.

Task 2: Create the New Headline Section

1. Add a *Panel* widget to your application and name it **`PANEL_Header`**. The Headline should have a fixed height of **`80px`**. All other required sizing and position information can be taken from the general alignment definition mentioned in the exercise introduction.

   a) Within the *Insert*-section of the toolbar, choose *Add* → *Panel*.

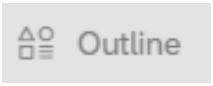
   b) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`PANEL_Header`**
Size and Position	
Width	**`90 %`**
Height	**`80 px`**
Left	**`5 %`**
Top	**`10 px`**
Right	**`auto`**
Bottom	**`auto`**
Widget	
Background Color	**`no color`**

> Note:
> You can use the *Device* setting at the bottom of the screen to select **`Auto`** to have a better overview when designing your application.

2. Add the widgets *CLOCK_Head*, *TX_Headline*, *IMG_Logo* into the newly created panel widget. Assign the size and positioning settings so that the following requirements are fulfilled:

   - The clock covers **`15% of the width`** of the panel and has **`no left border`**. It has a **`fixed height of 80 px`** and **`no top border`**.
   - The text covers **`100% of the width`** of the panel. **`Height and top border are fixed with 32 px`**.
   - The logo is **`fixed to 200 px width`** and has **`no right border`**. The **`height is fixed with 32 px`** with a **`24 px border to the top`**.

a) The easiest way to place the needed widgets into the newly created panel is to use the structural view in the 

Outline area . Select the clock *CLOCK_Head* widget and drag and drop it onto the panel *PANEL_Header* element.

b) Within the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Size and Position	
Width	**15 %**
Height	**80 px**
Left	**0 px**
Top	**0 px**
Right	**auto**
Bottom	**auto**

c) Within the *Outline* area, drag and drop the text widget *TX_Headline* onto the panel *PANEL_Header*.

d) Within the *Styling* pane, choose the settings as shown in the following table:

Field	Value
Size and Position	
Width	**100 %**
Height	**32 px**
Left	**0 px**
Top	**32 px**
Right	**auto**
Bottom	**auto**

e) Within the *Outline* area, drag and drop the image widget *IMG_Logo* onto the panel *PANEL_Header*.

f) Within the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Size and Position	
Width	**200 px**
Height	**32 px**
Left	**auto**
Top	**24 px**

Field	Value
Right	**0 px**
Bottom	**auto**

Task 3: Create the New KPI-Tiles Section

1. Add a *Panel* widget to your application and name it **PANEL_Tiles**. The KPI-Tiles should have a fixed height of **160px** and should start **10px** underneath the *PANEL_Header*. All other required sizing and position information can be taken from the general alignment definition mentioned in the exercise introduction.

   a) Within the *Outline*, select the *Canvas*.

   b) Within the *Insert*-section of the toolbar, choose *Add* → *Panel*.

   c) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**PANEL_Tiles**
Size and Position	
Width	**90 %**
Height	**160 px**
Left	**5 %**
Top	**100 px**
Right	**auto**
Bottom	**auto**
Widget	
Background Color	**no color**

2. Add the widgets *CH_Tile_Tax*, *CH_Tile_Quantity*, *CH_Tile_Margin* into the newly created panel widget. Assign the size and positioning settings so that the following requirements are fulfilled:

   - Each KPI tile should cover **100% of the available height** of the panel.
   - Each KPI tile should cover **33.3% of the available width** so that they are positioned next to each other.

   a) Within the *Outline* area, drag and drop the chart widget *CH_Tile_Tax* onto the panel *PANEL_Tiles*.

   b) Within the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Size and Position	

Field	Value
Width	`33.3 %`
Height	`auto`
Left	`0 px`
Top	`0 px`
Right	`auto`
Bottom	`0 px`

c) Within the *Outline* area, drag and drop the chart widget *CH_Tile_Quantity* onto the panel *PANEL_Tiles*.

d) Within the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Size and Position	
Width	`33.3 %`
Height	`auto`
Left	`33.3 %`
Top	`0 px`
Right	`auto`
Bottom	`0 px`

e) Within the *Outline* area, drag and drop the chart widget *CH_Tile_Margin* onto the panel *PANEL_Tiles*.

f) Within the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Size and Position	
Width	`33.3 %`
Height	`auto`
Left	`auto`
Top	`0 px`
Right	`0 px`
Bottom	`0 px`

Task 4: Create the New Data Visualization Section

1. Add a *Tab Strip* widget to your application and name it `TS_Content`. The tab strip should use the available space starting 10 px below the KPI tiles and keeping a 10 px border to the bottom. All other required sizing and position information can be taken from the general

alignment definition mentioned in the exercise introduction. The tab strip should contain three tabs labeled **`Taxes`**, **`Order Quantity`** and **`Gross Margin`**.

a) Within the *Outline*, select the *Canvas*.

b) Within the *Insert*-section of the toolbar, choose *Add* → *Tab Strip*.

c) Within the *Builder* panel, choose the + *Add* icon to create a third tab.

d) Change the text of the various tabs by clicking on the text area of the single tab. Use the settings as shown in the following table:

Name	Text (Optional)	Default
`Tab_1`	**`Taxes`**	*selected*
`Tab_2`	**`Order Quantity`**	
`Tab_3`	**`Gross Margin`**	

e) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`TS_Content`**
Size and Position	
Width	**`90 %`**
Height	**`auto`**
Left	**`5 %`**
Top	**`270 px`**
Right	**`auto`**
Bottom	**`10 px`**

2. Place the chart *CH_TaxesPerMonth* into the *Taxes* tab. The chart should entirely cover the given space within the tab.

a) Within the *Outline* area, drag and drop the chart widget *CH_TaxesPerMonth* onto the first tab.

b) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Size and Position	
Width	**`auto`**
Height	**`auto`**
Left	**`0 px`**
Top	**`0 px`**

Field	Value
Right	**0 px**
Bottom	**0 px**

3. Place the filter line *FL_Quantity* into the *Order Quantity* tab. It should cover 100 % of the available width of the tab and first 60 px from the top.
   a) Within the *Outline* area, drag and drop the filter line *FL_Quantity* onto the second tab.
   b) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Size and Position	
Width	**auto**
Height	**60 px**
Left	**0 px**
Top	**0 px**
Right	**0 px**
Bottom	**auto**

4. Place the two input controls *IP_Country* and *IP_City* plus the table *TBL_Quantity* into the *Order Quantity* tab. All three widgets should cover the available height starting underneath the filter line to the bottom of the tab. The input control for countries is aligned to the left border of the tab and covers 190 px width. The input control for the cities is aligned 190px from the left border and covers 190px width. The table is aligned 380 px from the left border and covers the remaining width to the right border of the tab.
   a) Within the *Outline* area, drag and drop the filter line *IP_Country* onto the second tab.
   b) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Size and Position	
Width	**190 px**
Height	**auto**
Left	**0 px**
Top	**60 px**
Right	**auto**
Bottom	**0 px**

   c) Within the *Outline* area, drag and drop the filter line *IP_City* onto the second tab.
   d) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Size and Position	
Width	**`190 px`**
Height	**`auto`**
Left	**`190 px`**
Top	**`60 px`**
Right	**`auto`**
Bottom	**`0 px`**

a) Within the *Outline* area, drag and drop the table *TBL_Quantity* onto the second tab.

b) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Size and Position	
Width	**`auto`**
Height	**`auto`**
Left	**`380 px`**
Top	**`60 px`**
Right	**`0 px`**
Bottom	**`0 px`**

5. Add the widgets *CH_GrossMarginPerLocation*, *CH_GrossMarginPerSalesManager*, *CH_GrossMarginPerProduct* into the third tab. Assign the size and positioning settings so that the following requirements are fulfilled:

   - Each chart should cover **`100% of the available height`** of the panel.
   - Each chart should cover **`33.3% of the available width`** so that they are positioned next to each other.

   a) Within the *Outline* area, drag and drop the chart widget *CH_GrossMarginPerLocation* onto the third tab.

   b) Within the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Size and Position	
Width	**`33.3 %`**
Height	**`auto`**
Left	**`0 px`**
Top	**`0 px`**

Field	Value
Right	**auto**
Bottom	**0 px**

c) Within the *Outline* area, drag and drop the chart widget *CH_GrossMarginPerSalesManager* onto the third tab.

d) Within the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Size and Position	
Width	**33.3 %**
Height	**auto**
Left	**33.3 %**
Top	**0 px**
Right	**auto**
Bottom	**0 px**

e) Within the *Outline* area, drag and drop the chart widget *CH_GrossMarginPerProduct* onto the third tab.

f) Within the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Size and Position	
Width	**33.3 %**
Height	**auto**
Left	**auto**
Top	**0 px**
Right	**0 px**
Bottom	**0 px**

Task 5: Save and Test your Application Design

1. Use the *Device* setting to simulate, for example, an iPad or an Laptop screen size. Rotate the orientation between portrait and landscape view.

   a) Use the *Device* setting at the bottom of the screen to select **iPad** or **Laptop**. Check the behavior of the *Canvas* area.

   b) Choose the  *Rotate* icon. Check the behavior of the *Canvas* area.

   c) Use the *Device* setting at the bottom of the screen to switch it back to **Auto**.

2. Save your application and run it.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.
   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.

Unit 3
Exercise 5

Use Basic Scripting Capabilities 1

Business Example

Your end users request two enhancements to your application. You will incorporate these in this exercise using basic scripting principles.

- To enable easy data selection of the years 2018 and 2019, you are asked to add buttons overlaying the tax chart. All other visualizations should not get influenced.
- Some end users find the design of the application too overloaded and wish that when the application is started, only the header and KPI-tile area are visible. When they click on one of the tiles, the tab strip should become visible and the tab that corresponds to the clicked tile should be activated.

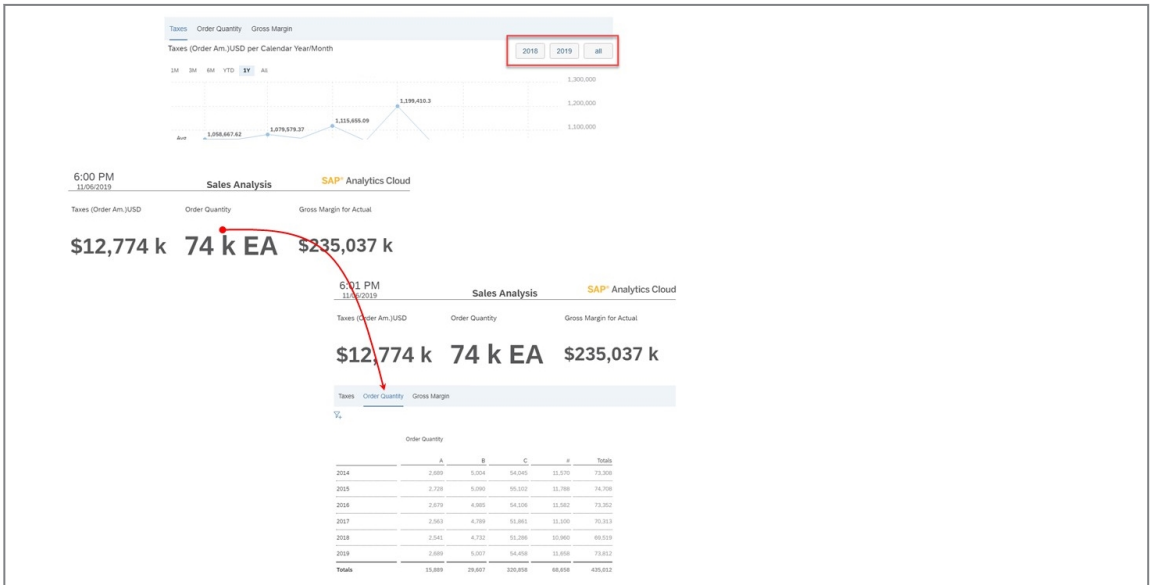


Figure 8: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Define the layout of your application* and save it with the new name **`GR### Use basic scripting capabilities 1`**.
2. Alternatively you can create a copy of the application *GR000 Define the layout of your application* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Solutions* and save it with the name **`GR### Use basic scripting capabilities 1`** into the folder structure *My Files*.

Task 2: Implement the Filtering Functionality

1. Add a *Button* widget onto the *Tab_1* tab and name it **`BUTTON_2018`**. We will focus on positioning in a later step of this exercise.

2. Define the code for *BUTTON_2018* that is needed to perform the following action:
   - The data source used in chart *CH_TaxesPerMonth* should be filtered on dimension **`0CALYEAR`** to the value **`2018`**.
3. Add another *Button* widget onto the *Tab_1* tab and name it **`BUTTON_2019`**. We will focus on positioning in a later step of this exercise.
4. Define the code for *BUTTON_2019* that is needed to perform the following action:
   - The data source used in chart *CH_TaxesPerMonth* should be filtered on dimension **`0CALYEAR`** to the value **`2019`**.
5. Add a third *Button* widget onto the *Tab_1* tab and name it **`BUTTON_ALL`**. We will focus on positioning in a later step of this exercise.
6. Define the code for *BUTTON_ALL* that is needed to perform the following action:
   - Any existing filter for dimension **`0CALYEAR`** should be removed on the data source used for chart *CH_TaxesPerMonth*.
7. Manage the positioning of the three buttons you just created. They all should be **`60 px wide`** and should be aligned with a **`10 px gap in between`**. They should be moving with the right border if the tab strip size changes. Check the desired layout at the beginning of the exercise description.

Task 3: Implement the New Start up and Navigation Concept

1. Make sure the *TS_Content* tab strip is initially not visible when the application is executed.
2. Define the code for *CH_Tile_Tax* that is needed to perform the following actions when the widget is clicked:
   - If the *TS_Content* tab strip is not visible, it should be set to **`visible = true`**.
   - The *Tab_1* tab is set as the selected tab.
3. Define the code for *CH_Tile_Quantity* that is needed to perform the following actions when the widget is clicked:
   - If the *TS_Content* tab strip is not visible, it should be set to **`visible = true`**.
   - The *Tab_2* tab is set as the selected tab.
4. Define the code for *CH_Tile_Margin* that is needed to perform the following actions when the widget is clicked:
   - If the *TS_Content* tab strip is not visible, it should be set to **`visible = true`**.
   - The *Tab_3* tab is set as the selected tab.

Task 4: Save and Test your Application Design

1. Save your application and run it.
2. Check if the requested start up and navigation concept works as requested.
3. Check if the new filtering functionality works for the *Taxes* tab.

Unit 3
Solution 5

Use Basic Scripting Capabilities 1

Business Example

Your end users request two enhancements to your application. You will incorporate these in this exercise using basic scripting principles.

- To enable easy data selection of the years 2018 and 2019, you are asked to add buttons overlaying the tax chart. All other visualizations should not get influenced.
- Some end users find the design of the application too overloaded and wish that when the application is started, only the header and KPI-tile area are visible. When they click on one of the tiles, the tab strip should become visible and the tab that corresponds to the clicked tile should be activated.

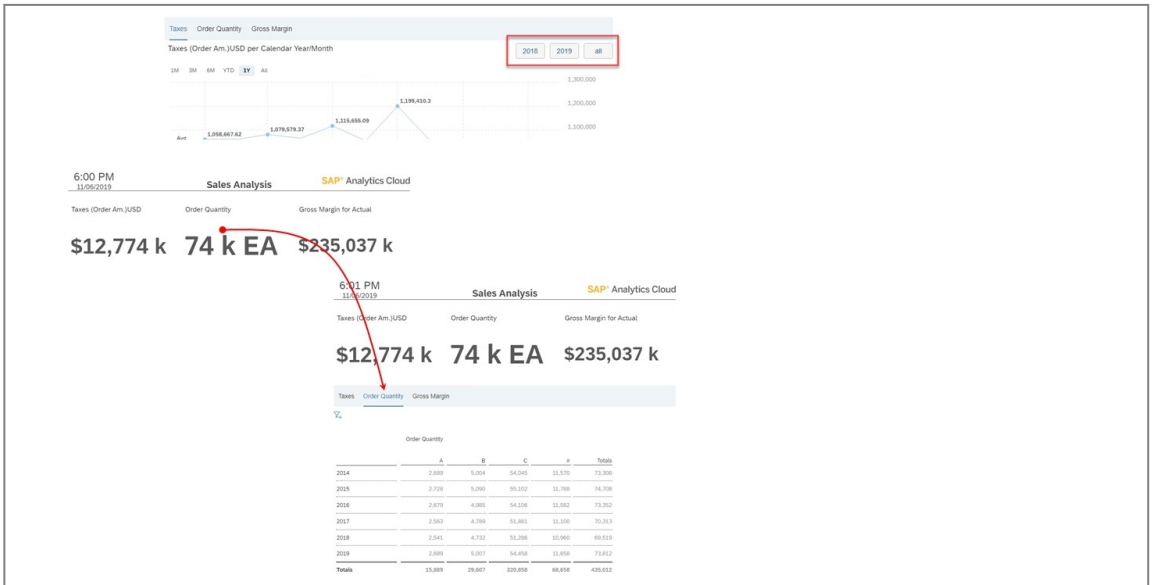


Figure 8: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Define the layout of your application* and save it with the new name **`GR### Use basic scripting capabilities 1`**.

   a) In the application designer, switch to your already opened application *GR### Define the layout of your application*.

   b) In the toolbar, choose the *Save* icon and choose *Save as...*.

   c) Type in **`GR### Use basic scripting capabilities 1`** as the name and choose *OK*.

2. Alternatively you can create a copy of the application *GR000 Define the layout of your application* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Solutions* and save it with the name **`GR### Use basic scripting capabilities 1`** into the folder structure *My Files*.

   a) From the Navigation Bar choose  *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Solutions*.

   b) Select the checkbox for the application *GR000 Define the layout of your application* and choose  *Copy*.

   c) Choose *My Files* and type in **`GR### Use basic scripting capabilities 1`**.

   d) Choose *OK*.

   e) Navigate to the *My Files* area.

   f) Choose *GR### Use basic scripting capabilities 1* to open the file in edit mode.

Task 2: Implement the Filtering Functionality

1. Add a *Button* widget onto the *Tab_1* tab and name it **`BUTTON_2018`**. We will focus on positioning in a later step of this exercise.

   a) Within the *Outline*, navigate to the *TS_Content* tabstrip and select *Tab_1*.

   b) Within the *Insert*-section of the toolbar, choose *Add* → *Button*.

   c) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`BUTTON_2018`**
Text	**`2018`**

2. Define the code for *BUTTON_2018* that is needed to perform the following action:

   - The data source used in chart *CH_TaxesPerMonth* should be filtered on dimension **`0CALYEAR`** to the value **`2018`**.

   a) Choose the button *BUTTON_2018* within the canvas area. In the upcoming menu, choose the  *Edit Scripts* button and select the *onClick* option to start the script editor.

   b) Enter the following line of code in the script editor:

```
CH_TaxesPerMonth.getDataSource().setDimensionFilter("0CALYEAR",
"2018");
```

3. Add another *Button* widget onto the *Tab_1* tab and name it **`BUTTON_2019`**. We will focus on positioning in a later step of this exercise.

   a) Within the *Insert*-section of the toolbar, choose *Add* → *Button*.

   b) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**BUTTON_2019**
Text	**2019**

4. Define the code for *BUTTON_2019* that is needed to perform the following action:

   - The data source used in chart *CH_TaxesPerMonth* should be filtered on dimension **0CALYEAR** to the value **2019**.

   a) Choose the button *BUTTON_2019* within the canvas area. In the upcoming menu, choose the *fx* *Edit Scripts* button and select the *onClick* option to start the script editor.

   b) Enter the following line of code in the script editor:

```
CH_TaxesPerMonth.getDataSource().setDimensionFilter("0CALYEAR",
"2019");
```

5. Add a third *Button* widget onto the *Tab_1* tab and name it **BUTTON_ALL**. We will focus on positioning in a later step of this exercise.

   a) Within the *Insert*-section of the toolbar, choose *Add* → *Button*.

   b) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**BUTTON_ALL**
Text	**all**

6. Define the code for *BUTTON_ALL* that is needed to perform the following action:

   - Any existing filter for dimension **0CALYEAR** should be removed on the data source used for chart *CH_TaxesPerMonth*.

   a) Choose the button *BUTTON_ALL* within the canvas area. In the upcoming menu, choose the *fx* *Edit Scripts* button and select the *onClick* option to start the script editor.

   b) Enter the following line of code in the script editor:

```
CH_TaxesPerMonth.getDataSource().removeDimensionFilter("0CALYEAR");
```

7. Manage the positioning of the three buttons you just created. They all should be **60 px wide** and should be aligned with a **10 px gap in between**. They should be moving with the right border if the tab strip size changes. Check the desired layout at the beginning of the exercise description.

   a) For the widget *BUTTON_ALL*, maintain the following settings in the *Styling* section of the *Designer*:

Field	Value
Size and Position	
Width	`60 px`
Height	`32 px`
Left	`auto`
Top	`5 px`
Right	`0 px`
Bottom	`auto`

b) For the widget *BUTTON_2019*, maintain the following settings in the *Styling* section of the *Designer*:

Field	Value
Size and Position	
Width	`60 px`
Height	`32 px`
Left	`auto`
Top	`5 px`
Right	`70 px`
Bottom	`auto`

c) For the widget *BUTTON_2018*, maintain the following settings in the *Styling* section of the *Designer*:

Field	Value
Size and Position	
Width	`60 px`
Height	`32 px`
Left	`auto`
Top	`5 px`
Right	`140 px`
Bottom	`auto`

Task 3: Implement the New Start up and Navigation Concept

1. Make sure the *TS_Content* tab strip is initially not visible when the application is executed.
   a) For the widget *TS_Content*, maintain the following setting in the *Styling* panel:

Field	Value
Actions	
Show this item at view time	☐

2. Define the code for *CH_Tile_Tax* that is needed to perform the following actions when the widget is clicked:
   - If the *TS_Content* tab strip is not visible, it should be set to **`visible = true`**.
   - The *Tab_1* tab is set as the selected tab.

   a) Choose the chart *CH_Tile_Tax* within the canvas area. In the upcoming menu, choose *Edit Scripts* → *onSelect* to open the script editor.

   b) Enter the following lines of code in the script editor:

```
if (TS_Content.isVisible() === false) {
    TS_Content.setVisible(true);
}
TS_Content.setSelectedKey("Tab_1");
```

3. Define the code for *CH_Tile_Quantity* that is needed to perform the following actions when the widget is clicked:
   - If the *TS_Content* tab strip is not visible, it should be set to **`visible = true`**.
   - The *Tab_2* tab is set as the selected tab.

   a) Choose the chart *CH_Tile_Quantity* within the canvas area. In the upcoming menu, choose *Edit Scripts* → *onSelect* to open the script editor.

   b) Enter the following lines of code in the script editor:

```
if (TS_Content.isVisible() === false) {
    TS_Content.setVisible(true);
}
TS_Content.setSelectedKey("Tab_2");
```

4. Define the code for *CH_Tile_Margin* that is needed to perform the following actions when the widget is clicked:
   - If the *TS_Content* tab strip is not visible, it should be set to **`visible = true`**.
   - The *Tab_3* tab is set as the selected tab.

   a) Choose the chart *CH_Tile_Margin* within the canvas area. In the upcoming menu, choose *Edit Scripts* → *onSelect* to open the script editor.

   b) Enter the following lines of code in the script editor:

```
if (TS_Content.isVisible() === false) {
    TS_Content.setVisible(true);
}
TS_Content.setSelectedKey("Tab_3");
```

Task 4: Save and Test your Application Design

1. Save your application and run it.

   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.

b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.

2. Check if the requested start up and navigation concept works as requested.
   a) When the application starts the whole tab strip should not be visible.
   b) When you click on one of the tiles the corresponding tab should become visible.
   c) Check with all three tiles.
3. Check if the new filtering functionality works for the *Taxes* tab.
   a) Open the *Taxes* tab and choose one of the buttons to make sure the desired selection works.
   b) Check with all three buttons.

Unit 3
Exercise 6

Use Basic Scripting Capabilities 2

Business Example

Your end users like the time series chart for the taxes but want to have greater flexibility on the time dimension shown in the chart. They request a dropdown box below the time series chart to choose if months, quarters, or years are visualized.

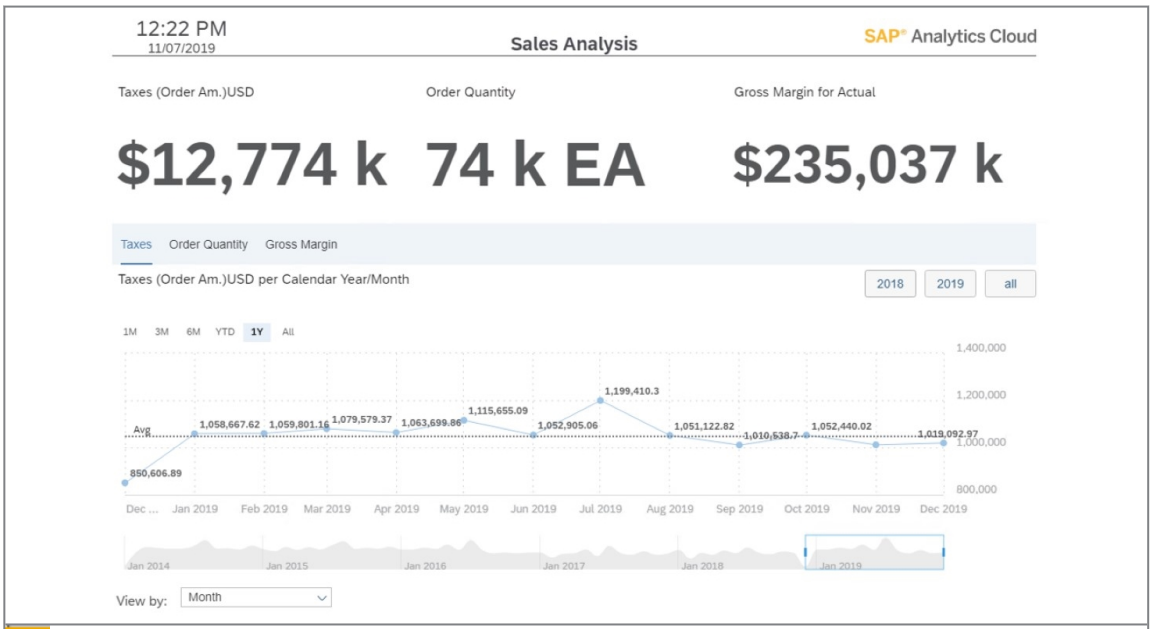


Figure 9: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Use basic scripting capabilities 1* and save it with the new name **`GR### Use basic scripting capabilities 2`**.

2. Alternatively you can create a copy of the application *GR000 Use basic scripting capabilities 1* which is available in the folder *Public* → *SACAD1_30*→ *SACAD1 Solutions* and save it with the name **`GR### Use basic scripting capabilities 2`** into the folder structure *My Files*.

Task 2: Implement the Filtering Functionality

1. Generate a **`40 px`** height empty space below the chart *CH_TaxesPerMonth* within the *Taxes* tab.

2. Add a new textbox to the *Taxes* tab, name it **`TX_View`**, and maintain the text **`View by:`**. Align it to the bottom left corner of the tab as shown in the exercise description.

3. Add a new *Dropdown* box to the *Taxes* tab and name it **`DD_View`**. Align it next to *TX_View* to the bottom left corner of the tab as shown in the exercise description.

4. For the dropdown box *DD_View*, create one entry each for **`Month`**, **`Quarter`**, and **`Year`** using the technical dimension name (in the underlying source system) as *value*. Define the entry *Month* as the default selected one.
5. Define the code for *DD_View* that is needed to perform the following action whenever a new value is selected:
   - All three time dimensions should be removed from the feed of the time axis.
   - The one selected in *DD_View* should be added to the feed of the time axis.

Task 3: Save and Test your Application Design

1. Save your application and run it.
2. Check if the new functionally that you implemented on the *Taxes* tab works as expected.

Use Basic Scripting Capabilities 2

Business Example

Your end users like the time series chart for the taxes but want to have greater flexibility on the time dimension shown in the chart. They request a dropdown box below the time series chart to choose if months, quarters, or years are visualized.

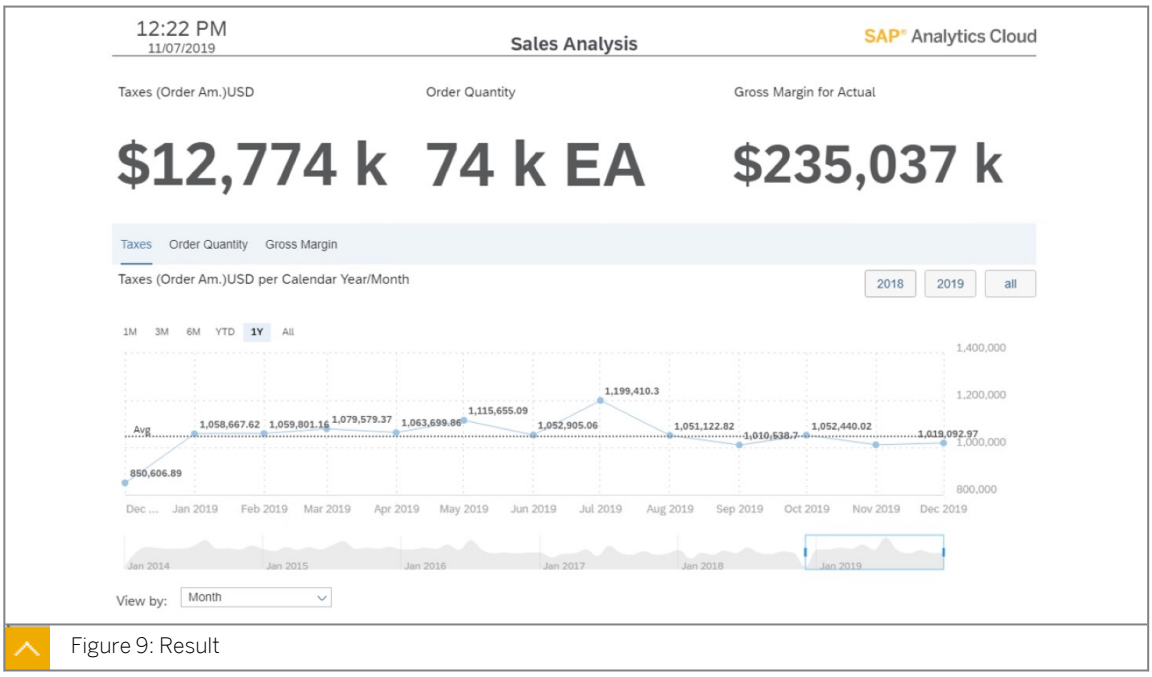


Figure 9: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Use basic scripting capabilities 1* and save it with the new name **`GR### Use basic scripting capabilities 2`**.

   a) In the application designer, switch to your already opened application *GR### Use basic scripting capabilities 1*.

   b) In the toolbar, choose the *Save* icon and choose *Save as...*

   c) Type in **`GR### Use basic scripting capabilities 2`** as the name and choose *OK*.

2. Alternatively you can create a copy of the application *GR000 Use basic scripting capabilities 1* which is available in the folder *Public* → *SACAD1_30*→ *SACAD1 Solutions* and save it with the name **`GR### Use basic scripting capabilities 2`** into the folder structure *My Files*.

a) From the Navigation Bar choose  *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Solutions*.

b) Select the checkbox for the application *GR000 Use basic scripting capabilities 1* and choose  *Copy*.

c) Choose *My Files* and type in **`GR### Use basic scripting capabilities 2`**.

d) Choose *OK*.

e) Navigate to the *My Files* area.

f) Choose *GR### Use basic scripting capabilities 2* to open the file in edit mode.

Task 2: Implement the Filtering Functionality

1. Generate a **`40 px`** height empty space below the chart *CH_TaxesPerMonth* within the *Taxes* tab.

   a) Select the chart *CH_TaxesPerMonth* in the *Outline* section and open the *Styling* panel.

   b) Change the *Bottom* border to **`40`** px.

2. Add a new textbox to the *Taxes* tab, name it **`TX_View`**, and maintain the text **`View by:`**. Align it to the bottom left corner of the tab as shown in the exercise description.

   a) Choose the *Tab_1* tab in the *Outline* section.

   b) Within the *Insert*-section of the toolbar, choose *Add* → *Text*.

   c) Select the textbox in the *Outline* section, click into it in the *Canvas* and type in **`View by:`** as text.

   d) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`TX_View`**
Size and Position	
Width	**`100 px`**
Height	**`32 px`**
Left	**`0 px`**
Top	**`auto`**
Right	**`auto`**
Bottom	**`8 px`**

3. Add a new *Dropdown* box to the *Taxes* tab and name it **`DD_View`**. Align it next to *TX_View* to the bottom left corner of the tab as shown in the exercise description.

   a) Choose the *Tab_1* tab in the *Outline* section.

b) Within the *Insert*-section of the toolbar, choose *Add* → *Dropdown*.

c) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**DD_View**
Size and Position	
Width	**100 px**
Height	**32 px**
Left	**80 px**
Top	**auto**
Right	**auto**
Bottom	**8 px**

4. For the dropdown box *DD_View*, create one entry each for **Month**, **Quarter**, and **Year** using the technical dimension name (in the underlying source system) as *value*. Define the entry *Month* as the default selected one.

a) Within the *Builder* panel, create three new entries by choosing the + icon three times and maintain the settings as shown in the following table:

ID	Text (Optional)	Default
0CALMONTH	**Month**	*selected*
0CALQUARTER	**Quarter**	
0CALYEAR	**Year**	

5. Define the code for *DD_View* that is needed to perform the following action whenever a new value is selected:

- All three time dimensions should be removed from the feed of the time axis.
- The one selected in *DD_View* should be added to the feed of the time axis.

a) Choose *DD_View* within the *Outline* area and choose the *fx* *Edit Scripts* button to start the script editor.

b) Enter the following line of code in the script editor:

```
CH_TaxesPerMonth.removeDimension("0CALMONTH", Feed.TimeAxis);
CH_TaxesPerMonth.removeDimension("0CALQUARTER", Feed.TimeAxis);
CH_TaxesPerMonth.removeDimension("0CALYEAR", Feed.TimeAxis);

CH_TaxesPerMonth.addDimension(DD_View.getSelectedKey(), Feed.TimeAxis);
```

Task 3: Save and Test your Application Design

1. Save your application and run it.

a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.

b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.

2. Check if the new functionally that you implemented on the *Taxes* tab works as expected.

a) Select one of the entries of the dropdown box on the *Taxes* tab and check if the chart changes accordingly.

Unit 3
Exercise 7

Use Advanced Scripting Capabilities 1

Business Example

Your UX designers criticized the three buttons on the *Taxes* tab. They encourage you to use a dropdown box at the bottom of the tab similar to the selection of the time dimension to achieve a cleaner layout of the application.

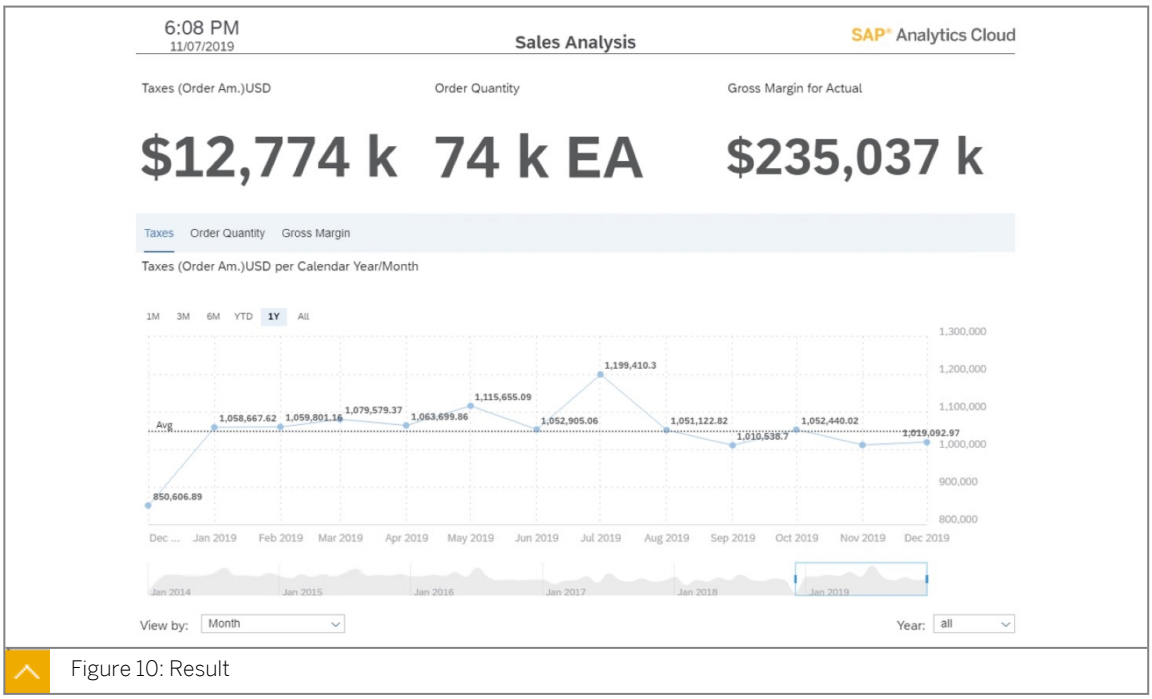


Figure 10: Result

In addition to the UI changes asked by the designers, you realize that your end users are split up in to two groups.

- Some like the approach to only see the KPI tiles at start up.
- Others would like to see the complete application directly when they open it.

As you do not want to maintain two different versions of the application in the future, you decide to introduce a URL parameter to be able to enable a *condensed* or *full* start up behavior.

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Use basic scripting capabilities 2* and save it with the new name **`GR### Use advanced scripting capabilities 1`**.

2. Alternatively you can create a copy of the application *GR000 Use basic scripting capabilities 2* which is available in the folder *Public → SACAD1_30 → SACAD1 Solutions*

and save it with the name **`GR### Use advanced scripting capabilities 1`** into the folder structure *My Files*.

Task 2: Implement the Filtering Functionality

1. Delete the three buttons *BUTTON_2018*, *BUTTON_2019*, and *BUTTON_all* from the *Taxes* tab.
2. Add a new *Text* to the *Taxes* tab, name it **`TX_Year`**, and maintain the text **`Year:`**. Align it to the bottom right corner of the tab as shown in the exercise description.
3. Add a new *Dropdown* box to the *Taxes* tab and name it **`DD_Year`**. Align it next to *TX_Year* to the bottom right corner of the tab as shown in the exercise description.
4. For the dropdown box *DD_Year*, create three entries **`all`**, **`2019`**, and **`2018`** and define the entry *all* as the default selected one.
5. Define the code for *DD_Year* that is needed to perform the following action whenever a new value is selected:
   - Check if one of the two listed years or the entry all is selected.
   - Depending on that, either filter the data in chart *CH_TaxesPerMonth* to the selected year or remove the filter for the dimension year.

Task 3: Implement a URL Parameter to Control the Start up

1. Create the a global script variable **`viewMode`** and expose it as URL parameter. It should contain the value **`full`** as the default value.
2. Define the code that is needed to perform the following action when the application is started:
   - When the URL parameter is set to **`condensed`**, the tab strip widget *TS_Content* should be invisible.
   - When the URL parameter holds any other value, the tab strip widget *TS_Content* should be visible.

Task 4: Save and Test your Application Design

1. Save your application and run it.
2. Check if the URL parameter works as expected by using **`condensed`** and **`full`** as the value for it.

Use Advanced Scripting Capabilities 1

Business Example

Your UX designers criticized the three buttons on the *Taxes* tab. They encourage you to use a dropdown box at the bottom of the tab similar to the selection of the time dimension to achieve a cleaner layout of the application.

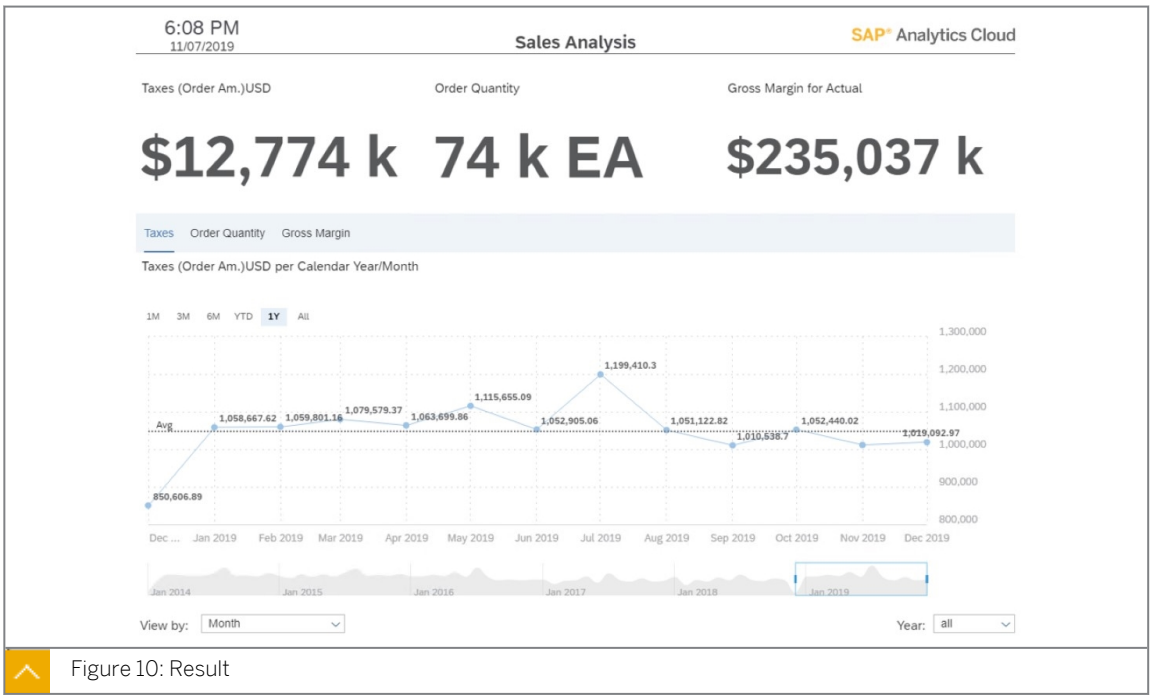


Figure 10: Result

In addition to the UI changes asked by the designers, you realize that your end users are split up in to two groups.

- Some like the approach to only see the KPI tiles at start up.
- Others would like to see the complete application directly when they open it.

As you do not want to maintain two different versions of the application in the future, you decide to introduce a URL parameter to be able to enable a *condensed* or *full* start up behavior.

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Use basic scripting capabilities 2* and save it with the new name **`GR### Use advanced scripting capabilities 1`**.

   a) In the application designer, switch to your already opened application *GR### Use basic scripting capabilities 2*.

b) In the toolbar, choose the *Save* icon and choose *Save as...*.

c) Type in **`GR### Use advanced scripting capabilities 1`** as the name and choose *OK*.

2. Alternatively you can create a copy of the application *GR000 Use basic scripting capabilities 2* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Solutions* and save it with the name **`GR### Use advanced scripting capabilities 1`** into the folder structure *My Files*.

a) From the Navigation Bar choose  *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Solutions*.

b) Select the checkbox for the application *GR000 Use basic scripting capabilities 2* and choose  *Copy*.

c) Choose *My Files* and type in **`GR### Use advanced scripting capabilities 1`**.

d) Choose *OK*.

e) Navigate to the *My Files* area.

f) Choose *GR### Use advanced scripting capabilities 1* to open the file in edit mode.

Task 2: Implement the Filtering Functionality

1. Delete the three buttons *BUTTON_2018*, *BUTTON_2019*, and *BUTTON_all* from the *Taxes* tab.

a) Within the *Outline* area, select the button you want to delete and choose *More* → *Delete*.

b) Repeat the step above for the other two buttons.

2. Add a new *Text* to the *Taxes* tab, name it **`TX_Year`**, and maintain the text **`Year:`**. Align it to the bottom right corner of the tab as shown in the exercise description.

a) Choose the *Tab_1* tab in the *Outline* section.

b) Within the *Insert*-section of the toolbar, choose *Add* → *Text*.

c) Within the text field, type in **`Year:`** as text.

d) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`TX_Year`**
Size and Position	
Width	**`50 px`**
Height	**`32 px`**
Left	**`auto`**

Field	Value
Top	**auto**
Right	**100 px**
Bottom	**8 px**

3. Add a new *Dropdown* box to the *Taxes* tab and name it **DD_Year**. Align it next to *TX_Year* to the bottom right corner of the tab as shown in the exercise description.
   a) Within the *Insert*-section of the toolbar, choose *Add* → *Dropdown*.
   b) Within the *Styling* panel, change the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**DD_Year**
Size and Position	
Width	**80px**
Height	**32 px**
Left	**auto**
Top	**auto**
Right	**0**
Bottom	**8 px**

4. For the dropdown box *DD_Year*, create three entries **all**, **2019**, and **2018** and define the entry *all* as the default selected one.
   a) Within the *Builder* panel, create three new entries by choosing the + icon three times and maintain the settings as shown in the following table:

ID	Text (Optional)	Default
all		*selected*
2019		
2018		

5. Define the code for *DD_Year* that is needed to perform the following action whenever a new value is selected:
   - Check if one of the two listed years or the entry all is selected.
   - Depending on that, either filter the data in chart *CH_TaxesPerMonth* to the selected year or remove the filter for the dimension year.

a) Choose *DD_Year* within the *Outline* area. Choose the *Edit Scripts* button to start the script editor.

b) Enter the following lines of code in the script editor:

```
var selection = DD_Year.getSelectedKey();
if (selection === "all") {
    CH_TaxesPerMonth.getDataSource().removeDimensionFilter("0CALYEAR");
} else {
    CH_TaxesPerMonth.getDataSource().setDimensionFilter("0CALYEAR",
selection);
}
```

Task 3: Implement a URL Parameter to Control the Start up

1. Create the a global script variable **`viewMode`** and expose it as URL parameter. It should contain the value **`full`** as the default value.

   a) Choose the *Add Script Variable* button in the *Outline* area.

   b) Maintain the following settings in the *Script Variable* panel and choose *Done*.

Field	Value
Structure	
Name	**`viewMode`**
Type	**`string`**
Default value	**`full`**
Expose variable via URL parameter	☑

2. Define the code that is needed to perform the following action when the application is started:

   - When the URL parameter is set to **`condensed`**, the tab strip widget *TS_Content* should be invisible.
   - When the URL parameter holds any other value, the tab strip widget *TS_Content* should be visible.

   a) Choose *Canvas* within the *Outline* area and choose the *Edit Scripts* button. Select *onInitialization* event to start the script editor.

   b) Enter the following lines of code in the script editor:

```
if (viewMode === "condensed") {
    TS_Content.setVisible(false);
} else {
    TS_Content.setVisible(true);
}
```

Task 4: Save and Test your Application Design

1. Save your application and run it.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.

b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.

2. Check if the URL parameter works as expected by using **`condensed`** and **`full`** as the value for it.

   a) Attach **`&p_viewMode=condensed`** to the URL you see in the browser with the executed application, press *Enter* and refresh the browser. The application should not show the tab strip with the detailed content.

   b) Change the URL to **`&p_viewMode=full`** at the end, press *Enter* and refresh the browser. The application should show the tab strip with the detailed content.

Unit 3
Exercise 8

Use Advanced Scripting Capabilities 2

Business Example

You receive some enhancement requests regarding the *Order Quantity* tab. Your users want to be able to do the following:

- See materials instead of years in the table.
- Calculate a percentage change on top of the current values. The absolute change and the forecast should be displayed in two additional columns. The calculation of these two values should be done in the application itself, not in the underlying data model.
- Select one of the in the source system existing hierarchies for dimension material.
- Select one of the in the data source existing countries as filter.
- Display one of the attributes related to the product "weight" in the table.
- Define the percentage change with a slider in the range of -50 to 50.

As you do not want to maintain all existing hierarchies and countries manually within the application itself, they should be read live from the source system each time the application is used.

Together with the UX team, you decide that the UI elements needed for the requested features will be placed in a pop-up screen to keep the main application layout as clean as possible. We have pre-prepared the pop-up screen already with the widgets you will need. You only need to take care of the scripts to perform the desired actions.

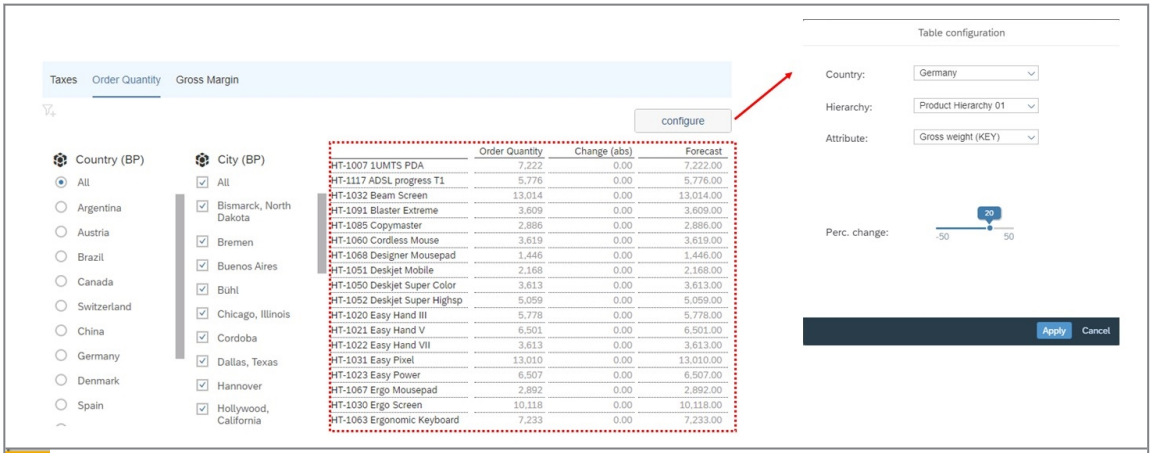


Figure 11: Result

Task 1: Create the Application

1. Create a copy of the application *GR000 Start Use advanced scripting capabilities 2* which is available in the folder *Public* → *SACAD1_30*→ *SACAD1 Content* and save it with the name **`GR### Use advanced scripting capabilities 2`**.

Task 2: Familiarize Yourself with the Given Application Structure

1. Compared to the previous exercise, we added the popup *POPUP_Config* for you. The popup contains all widgets (labels and navigation elements) you need for the following exercise. Just the header and footer is missing, plus the complete script logic. In addition, we already prepared a *BUTTON_Config* button within the *Tab_2* tab. The button should be used to open the popup but it's not yet configured to do so.

Task 3: Enable the Changes within the Table

1. Change the displayed data within *TBL_Quantity* to display *Product ID* in the *Rows* and the measure *Order Quantity* in the *Columns*. Enable the responsive visualization of the table with *Adaptive Column Width*.
2. Create a global script variable **`SV_PercChange`** that will store the percentage change value.
3. Add a calculation **`Change (abs)`** in the columns of *TBL_Quantity* that calculates the absolute change out of the given order quantity and the percentage change value.
4. Add a calculation **`Forecast`** in the columns of the table that calculates new order quantity including the change.

Task 4: Modify the Popup Screen

1. Enable the header and footer area for the predefined popup *POPUP_Config*. Choose **`Table configuration`** as the *Title* and rename the buttons to **`BUTTON_Ok`** and **`BUTTON_Cancel`**. Change the text for *BUTTON_Ok* to **`Apply`**.

Task 5: Create the Scripting Part

1. Define the code that is needed to open the popup *POPUP_Config* when the button *BUTTON_Config* is pressed.
2. Define the code that is needed to read all dimension values of the dimension *Country* and generate an entry in *DD_Country* for each. This should be done each time the application starts.
3. Define the code that is needed to read all hierarchies of dimension *Product ID (EPM Demo)* and generate an entry in *DD_Hierachy* for each. This should be done each time the application starts.
4. Define the code that is needed to read all the weight properties of dimension *Product ID (EPM Demo)* and generate an entry in *DD_Properties* for each. This should be done each time the application starts.
5. Define the code that is needed to close *POPUP_Config* without any further action when the *Cancel* button is clicked within the popup.
6. Define the code that is needed to perform the following actions when the button *Apply* is clicked:
   - The selected country in *DD_Country* should be used as filter for the data in the table *TBL_Quantity*.

- The selected hierarchy in *DD_Hierarchy* should be assigned to the product dimension in table *TBL_Quantity*.
- The selected property in *DD_Properties* should be displayed in the table *TBL_Quantity*.
- The selected value of *SL_PercChange* should be send to the global script variable *SV_PercChange*.
- *POPUP_Config* should be closed.

Task 6: Save and Test your Application Design

1. Save your application and run it.
2. Switch to the *Order Quantity* tab and use the *configure* button to open the pop-up. Select a country of your choice, one of the existing hierarchies, and change the percentage slider. Check if your table changes accordingly.

Unit 3
Solution 8

Use Advanced Scripting Capabilities 2

Business Example

You receive some enhancement requests regarding the *Order Quantity* tab. Your users want to be able to do the following:

- See materials instead of years in the table.
- Calculate a percentage change on top of the current values. The absolute change and the forecast should be displayed in two additional columns. The calculation of these two values should be done in the application itself, not in the underlying data model.
- Select one of the in the source system existing hierarchies for dimension material.
- Select one of the in the data source existing countries as filter.
- Display one of the attributes related to the product "weight" in the table.
- Define the percentage change with a slider in the range of -50 to 50.

As you do not want to maintain all existing hierarchies and countries manually within the application itself, they should be read live from the source system each time the application is used.

Together with the UX team, you decide that the UI elements needed for the requested features will be placed in a pop-up screen to keep the main application layout as clean as possible. We have pre-prepared the pop-up screen already with the widgets you will need. You only need to take care of the scripts to perform the desired actions.

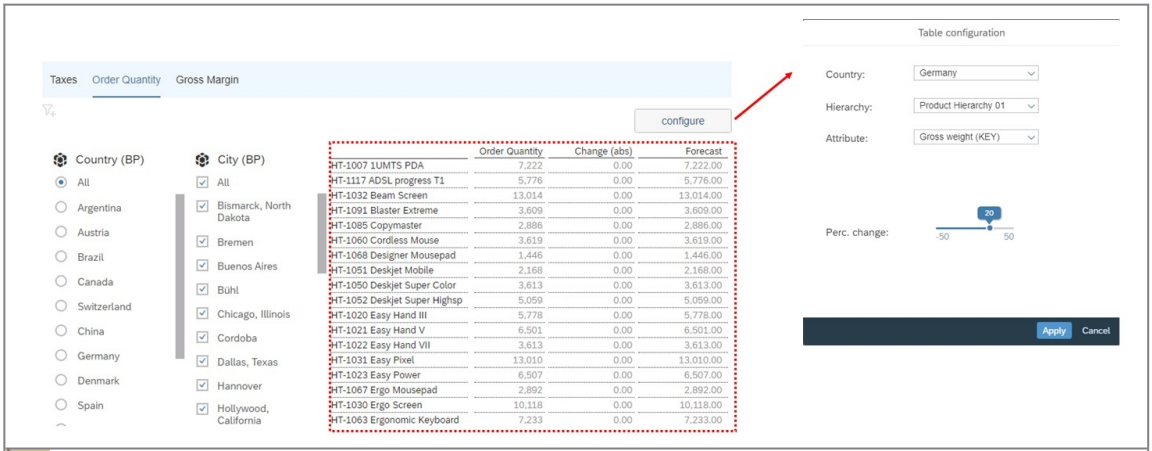


Figure 11: Result

Task 1: Create the Application

1. Create a copy of the application *GR000 Start Use advanced scripting capabilities 2* which is available in the folder *Public* → *SACAD1_30*→ *SACAD1 Content* and save it with the name **`GR### Use advanced scripting capabilities 2`**.

a) From the Navigation Bar choose  *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Content*.

b) Select the checkbox for the application *GR000 Start Use advanced scripting capabilities 2* and choose  *Copy* .

c) Choose *My Files* to choose your private folder.

d) Type in the name **`GR### Use advanced scripting capabilities 2`** and choose *OK*.

e) Navigate to the *My Files* area.

f) Choose *GR### Use advanced scripting capabilities 2* to open the file in edit mode.

Task 2: Familiarize Yourself with the Given Application Structure

1. Compared to the previous exercise, we added the popup *POPUP_Config* for you. The popup contains all widgets (labels and navigation elements) you need for the following exercise. Just the header and footer is missing, plus the complete script logic. In addition, we already prepared a *BUTTON_Config* button within the *Tab_2* tab. The button should be used to open the popup but it's not yet configured to do so.

   a) Check the *Outline* area to find the button and the popup. Click on the popup to see the definition of it.

Task 3: Enable the Changes within the Table

1. Change the displayed data within *TBL_Quantity* to display *Product ID* in the *Rows* and the measure *Order Quantity* in the *Columns*. Enable the responsive visualization of the table with *Adaptive Column Width*.

   a) Within the *Builder* panel, maintain the settings as shown in the following table:

Field	Value
Table Structure	
Adaptive Column Width	☑
Rows	remove *Calendar Year* and add *Product ID (EPM Demo)*
Columns	remove *NAV: Product ABC-Rat*

2. Create a global script variable **`SV_PercChange`** that will store the percentage change value.

   a) In the *Outline* area, choose the + *Add Script Variable* button.

   b) Maintain the following settings in the *Script Variable* panel and choose *Done*.

Field	Value
Structure	
Name	**`SV_PercChange`**

Field	Value
Type	**number**
Default Value	**0**

3. Add a calculation **Change (abs)** in the columns of *TBL_Quantity* that calculates the absolute change out of the given order quantity and the percentage change value.
   a) Within the *Builder* panel of *TBL_Quantity*, choose the ... *More* icon next to the Measures in the Columns and select *Add Calculation...*.
   b) In the *Calculation Editor*, type in **Change (abs)** as the *Name*.
   c) Use the formula **[Order Quantity * SV_PercChange] / 100** to calculate the result. Use the embedded value help to select the formula element. The value help shows up when you press **CTRL + Space** or when you start typing. When you do so, the formula looks as follows in the end:

```
["SACAD1_Model001":00O2SOEHXWHF3AU4LBEW8W475] * [@SV_PercChange] / 100
```

4. Add a calculation **Forecast** in the columns of the table that calculates new order quantity including the change.
   a) Within the *Builder* panel of *TBL_Quantity*, choose the ... *More* icon next to the Measures in the Columns and select *Add Calculation...*.
   b) In the *Calculation Editor*, type in **Forecast** as the *Name*.
   c) Use the formula **Order Quantity + Change (abs)** to calculate the result. Use the embedded value help to select the formula element. When you do so, the formula looks as follows in the end:

```
["SACAD1_Model001":00O2SOEHXWHF3AU4LBEW8W475] + [#Change (abs)]
```

Task 4: Modify the Popup Screen

1. Enable the header and footer area for the predefined popup *POPUP_Config*. Choose **Table configuration** as the *Title* and rename the buttons to **BUTTON_Ok** and **BUTTON_Cancel**. Change the text for *BUTTON_Ok* to **Apply**.
   a) Within the *Outline* area, find the pre-defined popup *POPUP_Config* and click on it.
   b) Within the *Builder* panel, **activate** the *Enable header & footer* option.
   c) Change the *Title* to **Table configuration**.
   d) Rename the IDs of the buttons to **BUTTON_Ok** and **BUTTON_Cancel**.
   e) Change the *Text* property of button *BUTTON_Ok* to **Apply**.
   f) Choose *Apply*.

Task 5: Create the Scripting Part

1. Define the code that is needed to open the popup *POPUP_Config* when the button *BUTTON_Config* is pressed.

a) Choose *BUTTON_Config* within the *Outline* area and choose the *Edit Scripts* button and choose *onClick* to start the script editor.

b) Enter the following line of code in the script editor:

```
POPUP_Config.open();
```

2. Define the code that is needed to read all dimension values of the dimension *Country* and generate an entry in *DD_Country* for each. This should be done each time the application starts.

a) Choose *Canvas* within the *Outline* area and choose the *Edit Scripts* button.

b) Choose *onInitialization* to start the script editor.

c) Add the following lines of code after the existing ones:

```
var CountryList =
TBL_Quantity.getDataSource().getMembers("0D_NW_BP__0D_NW_CNTRY");
for (var i=0; i<CountryList.length; i++){
    if (CountryList[i].id !== ""){
        DD_Country.addItem(CountryList[i].displayId,
CountryList[i].description);
    }
}
```

3. Define the code that is needed to read all hierarchies of dimension *Product ID (EPM Demo)* and generate an entry in *DD_Hierachy* for each. This should be done each time the application starts.

a) Choose *Canvas* within the *Outline* area and choose the *Edit Scripts* button to start the script editor.

b) Choose *onInitialization* to start the script editor.

c) Add the following lines of code after the existing ones:

```
var Hierarchies =
TBL_Quantity.getDataSource().getHierarchies("0D_NW_PRID");
console.log(Hierarchies);
for (var j=0; j<Hierarchies.length; j++){
        DD_Hierarchy.addItem(Hierarchies[j].id,
Hierarchies[j].description);
}
```

4. Define the code that is needed to read all the weight properties of dimension *Product ID (EPM Demo)* and generate an entry in *DD_Properties* for each. This should be done each time the application starts.

a) Choose *Canvas* within the *Outline* area and choose the *Edit Scripts* button to start the script editor.

b) Choose *onInitialization* to start the script editor.

c) Add the following lines of code after the existing ones:

```
var properties =
TBL_Quantity.getDataSource().getDimensionProperties("0D_NW_PRID");
```

```
for (var k=0; k<properties.length; k++){
    if (properties[k].description.includes("Weight") ||
properties[k].description.includes("weight")){
        DD_Properties.addItem(properties[k].id,
properties[k].description);
    }
}
```

5. Define the code that is needed to close *POPUP_Config* without any further action when the *Cancel* button is clicked within the popup.

   a) Choose *POPUP_Config* within the *Outline* area and choose the *Edit Scripts* button to start the script editor.

   b) Enter the following lines of code in the script editor:

```
if (buttonId === "BUTTON_Cancel") {
    POPUP_Config.close();
}
```

6. Define the code that is needed to perform the following actions when the button *Apply* is clicked:

   - The selected country in *DD_Country* should be used as filter for the data in the table *TBL_Quantity*.
   - The selected hierarchy in *DD_Hierarchy* should be assigned to the product dimension in table *TBL_Quantity*.
   - The selected property in *DD_Properties* should be displayed in the table *TBL_Quantity*.
   - The selected value of *SL_PercChange* should be send to the global script variable *SV_PercChange*.
   - *POPUP_Config* should be closed.

   a) Choose *POPUP_Config* within the *Outline* area and choose the *Edit Script* button to start the script editor.

   b) Add the following lines of code to the existing ones in the script editor:

```
if (buttonId === "BUTTON_Ok") {

TBL_Quantity.getDataSource().setDimensionFilter("0D_NW_BP__0D_NW_CNTRY"
, DD_Country.getSelectedKey());
    TBL_Quantity.getDataSource().setHierarchy("0D_NW_PRID",
DD_Hierarchy.getSelectedKey());
    var properties = ArrayUtils.create(Type.string);
    properties.push(DD_Properties.getSelectedKey());
    TBL_Quantity.setActiveDimensionProperties("0D_NW_PRID",properties);
    SV_PercChange = SL_PercChange.getValue();
    POPUP_Config.close();
}
```

Task 6: Save and Test your Application Design

1. Save your application and run it.

   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.

   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.

2. Switch to the *Order Quantity* tab and use the *configure* button to open the pop-up. Select a country of your choice, one of the existing hierarchies, and change the percentage slider. Check if your table changes accordingly.
   a) Choose the *Order Quantity* tab.
   b) Choose the *configure* button.
   c) Select **`USA`** as country.
   d) Select **`Product Hierarchy 01`** as hierarchy.
   e) Select **`Gross weight (KEY)`** as attribute.
   f) Select **`20`** as percentage change.
   g) Choose *Apply*.
   h) The pop-up should disappear and the data in the table should change.

Unit 3
Exercise 9

Looping the Result Set

Business Example

Looping over an existing result set in script offers a lot of new design options for your applications. In this exercise, you will get your first hands-on experience with a 2 step setup.

1. The idea of the first step is to present the top 3 countries in a leader board style that could be part of an dashboard. You will use an image showing the leader board and text fields on top of it to visualize the data.
2. As a second step, you could enable the visualization of all other countries in a tabular style. As the shown design does not fit to the one of a standard table, you create this visualization with text boxes on your own.

Of course all visualized data should update automatically as soon as the data changes, for example, while an end user selects a year.

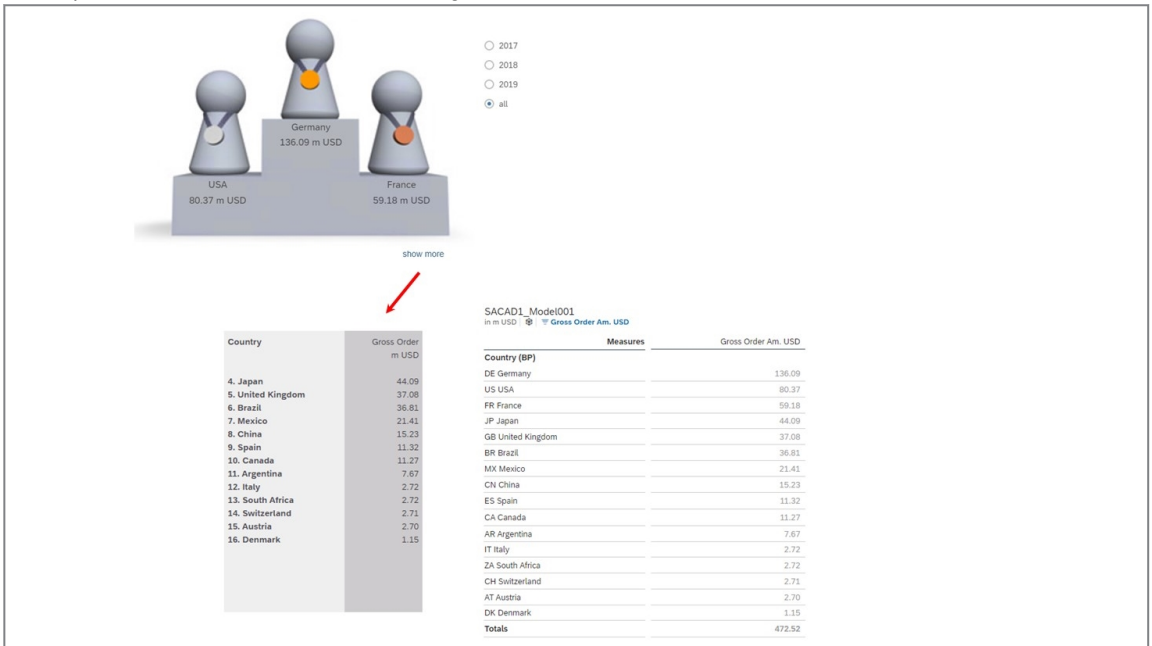


Figure 12: Result

In this exercise, we will work with a predefined start application containing all visualization widgets and some scripting already. Within this exercise, you only need to write a script function to loop over the result set and update all text fields accordingly. Of course you should not forget to include this function wherever needed to keep the values up to date.

Note:
As there is no self-sufficient data source that contains the result set, we use a hidden table widget with its contained data source to read the result set.

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Loop result set* from *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Loop result set`** into the folder structure *My Files*.
2. Open *GR### Loop result set* for editing.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze and understand the given application structure. Concentrate on the widgets starting with *TX_Top**, the table *TBL_Country*, and the already implemented navigation option with *RB_Years* and *BUTTON_More*.

Task 3: Implement the Existing Script Function where Needed

1. Think about at which situations the text widgets should get updated.

Hint:
You should have two different events that need to call the updating script function *SF_Generate*.

Task 4: Develop the Script to Visualize the Leader Board

1. Open the script editor for the existing script function *SF_Generate*.
2. Start with a code that reads the result set used by table *TBL_Country* and log it to the *Console* within the Chrome developer tools. Analyze the structure of the array and where to find the data you need.
3. Enhance the code so that it reads the first to third element of the result set array. For each element you write the description of the country dimension **`0D_NW_BP__0D_NW_CNTRY`** to the *TX_Top*_Name* text field.
4. For each element you write the formatted value of the measure dimension to the *TX_Top*_Value* text field. Use the scripting alias to reference the measure dimension.

Task 5: Save and Test your Application Design

1. Save your application and run it.
2. Check if the correct values are displayed for the Top 3 countries.

Task 6: Enhance your Script Function to Visualize the Tabular View

1. You will need to enhance the script function with a loop over all remaining entries of the result set and write each name and value into the corresponding text widget *TX_Names* or *TX_Values*. The result line should not be listed so exclude it in your script. If you check the given result at the beginning of the exercise again, you will notice that each country should be prefixed with its position within the result set and that the totals should not be displayed. Do not forget to generate a headline for each column.

Task 7: Save and Test your Application Design

1. Save your application and run it.
2. Check if the correct values are displayed in the tabular view after you choose the *show more* button.

Unit 3
Solution 9

Looping the Result Set

Business Example

Looping over an existing result set in script offers a lot of new design options for your applications. In this exercise, you will get your first hands-on experience with a 2 step setup.

1. The idea of the first step is to present the top 3 countries in a leader board style that could be part of an dashboard. You will use an image showing the leader board and text fields on top of it to visualize the data.
2. As a second step, you could enable the visualization of all other countries in a tabular style. As the shown design does not fit to the one of a standard table, you create this visualization with text boxes on your own.

Of course all visualized data should update automatically as soon as the data changes, for example, while an end user selects a year.

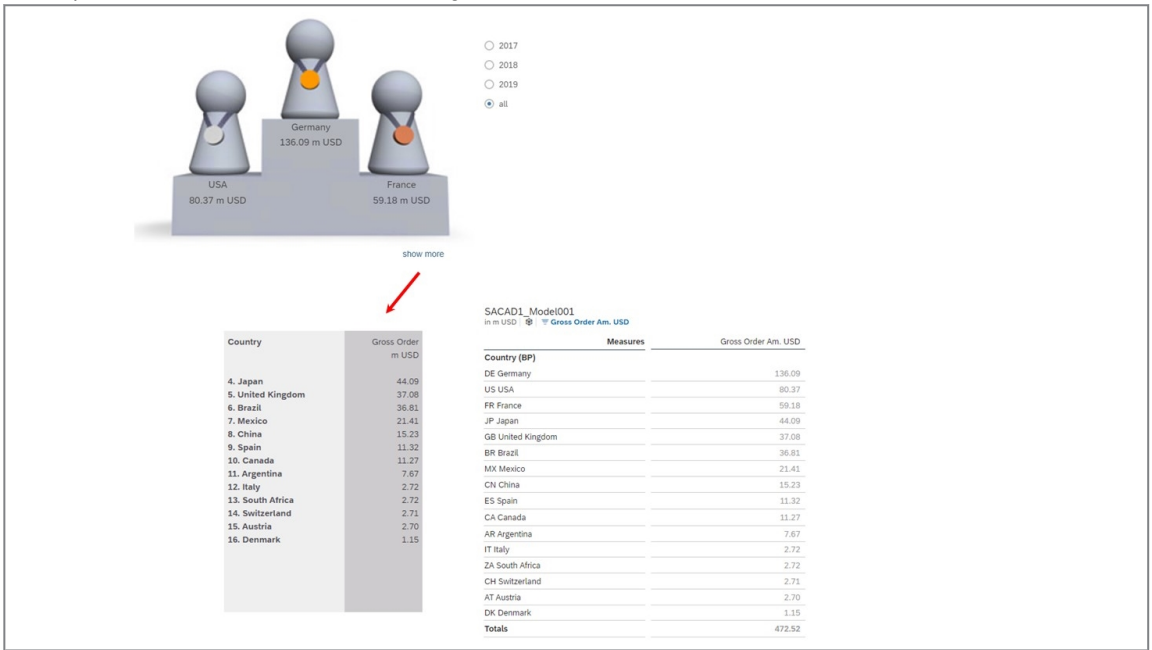


Figure 12: Result

In this exercise, we will work with a predefined start application containing all visualization widgets and some scripting already. Within this exercise, you only need to write a script function to loop over the result set and update all text fields accordingly. Of course you should not forget to include this function wherever needed to keep the values up to date.

Note:
As there is no self-sufficient data source that contains the result set, we use a hidden table widget with its contained data source to read the result set.

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Loop result set* from *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Loop result set`** into the folder structure *My Files*.
   a) From the Navigation Bar choose  *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Content*.
   b) Select the checkbox for the application *GR000 Start Loop result set* and choose  *Copy* .
   c) Choose *My Files* and type in **`GR### Loop result set`** as the name.
   d) Choose *OK*.
2. Open *GR### Loop result set* for editing.
   a) Navigate to the *My Files* area.
   b) Choose *GR### Loop result set* to open the file in edit mode.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze and understand the given application structure. Concentrate on the widgets starting with *TX_Top**, the table *TBL_Country*, and the already implemented navigation option with *RB_Years* and *BUTTON_More*.
   a) Check the *Styling* panel for the widgets and existing scripts.
   b) Make sure *TBL_Country* is set to show at view time.

Task 3: Implement the Existing Script Function where Needed

1. Think about at which situations the text widgets should get updated.

> 
> Hint:
> You should have two different events that need to call the updating script function *SF_Generate*.

   a) Choose *Canvas* within the *Outline* area and choose the *Edit Scripts* button . Choose *onInitialization* to start the script editor.
   b) Add the following line of code:

```
SO_Visualization.SF_Generate();
```

   c) Choose *TBL_Country* within the *Outline* area and choose the *Edit Scripts* button . Choose *onResultChanged* to start the script editor.
   d) Add the following line of code:

```
SO_Visualization.SF_Generate();
```

Task 4: Develop the Script to Visualize the Leader Board

1. Open the script editor for the existing script function *SF_Generate*.

a) Choose the script function *SF_Generate* within the *Outline* area and choose the *Edit Scripts* button to start the script editor.

2. Start with a code that reads the result set used by table *TBL_Country* and log it to the *Console* within the Chrome developer tools. Analyze the structure of the array and where to find the data you need.
   a) Add the following lines of code to the script function:

```
var resultSet = TBL_Country.getDataSource().getResultSet();
console.log(resultSet);
```

   b) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.
   c) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.
   d) Activate the *Developer tools* within Chrome by pressing **`Ctrl + Shift + I`** on your keyboard.
   e) Make sure the *Console* tab is activated.
   f) Within the console, look for an entry *Array* which should be near the bottom of the list.
   g) Unfold the *Array* entry and do the same for the first few entries below.

3. Enhance the code so that it reads the first to third element of the result set array. For each element you write the description of the country dimension **`0D_NW_BP__0D_NW_CNTRY`** to the *TX_Top*_Name* text field.
   a) Append the following lines of code to the script function *SF_Generate*:

```
var name = "";

name =  resultSet[0]["0D_NW_BP__0D_NW_CNTRY"].description;
TX_Top1_Name.applyText(name);

name =  resultSet[1]["0D_NW_BP__0D_NW_CNTRY"].description;
TX_Top2_Name.applyText(name);

name =  resultSet[2]["0D_NW_BP__0D_NW_CNTRY"].description;
TX_Top3_Name.applyText(name);
```

4. For each element you write the formatted value of the measure dimension to the *TX_Top*_Value* text field. Use the scripting alias to reference the measure dimension.
   a) Append the following lines of code to the script function *SF_Generate*:

```
var value = "";

value = resultSet[0][Alias.MeasureDimension].formattedValue + " m USD";
TX_Top1_Value.applyText(value);

value = resultSet[1][Alias.MeasureDimension].formattedValue + " m USD";
TX_Top2_Value.applyText(value);

value = resultSet[2][Alias.MeasureDimension].formattedValue + " m USD";
TX_Top3_Value.applyText(value);
```

Task 5: Save and Test your Application Design

1. Save your application and run it.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.

b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.

2. Check if the correct values are displayed for the Top 3 countries.
   a) Compare the displayed values either with entries within the console or the table.

Task 6: Enhance your Script Function to Visualize the Tabular View

1. You will need to enhance the script function with a loop over all remaining entries of the result set and write each name and value into the corresponding text widget *TX_Names* or *TX_Values*. The result line should not be listed so exclude it in your script. If you check the given result at the beginning of the exercise again, you will notice that each country should be prefixed with its position within the result set and that the totals should not be displayed. Do not forget to generate a headline for each column.
   a) Append the following lines of code to the script function *SF_Generate*:

```
var countries = "Country \n\n\n";
var values = "Gross Order \n m USD \n\n";

for (var x = 3; x < resultSet.length; x++) {
    if(resultSet[x]["0D_NW_BP__0D_NW_CNTRY"].id !==
Alias.TotalsMember) {
        countries = countries + ConvertUtils.numberToString(x + 1) +
". ";
        countries = countries + resultSet[x]
["0D_NW_BP__0D_NW_CNTRY"].description + "\n";
        values = values + resultSet[x]
["@MeasureDimension"].formattedValue + "\n";
    }
}

TX_Names.applyText(countries);
TX_Values.applyText(values);
```

Task 7: Save and Test your Application Design

1. Save your application and run it.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.

   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.

2. Check if the correct values are displayed in the tabular view after you choose the *show more* button.
   a) Compare the displayed values either with entries within the console or the table.

Unit 3
Exercise 10

Create an Application with Timer Function

Business Example

A timer can be used whenever something within your application should happen after a given period of time. If the timer restarts itself you create a permanent loop.

In this exercise, we will use a numeric point chart to present one measure value at a time. When the application starts, we want to change the presented measure every 5 seconds by looping through all available measures of the data source used for the chart. The user should be able to stop and restart the timer by clicking on the numeric point.

As a click on a numeric point must be precisely on the value, you overlay the chart with an invisible button to improve the usability for your users. The button should handle the code to stop or restart the loop through the measures.

Task 1: Create a New Empty Application

1. Create a new *Analytic Application* and save it with the name **`GR### Timer`** directly into the *My Files* content area.

Task 2: Create the Application Structure

1. Create a numeric point chart using *SACAD1_Model001* to display the measure *Gross Order Am. USD* and name it **`CH_Measure`**.

   Use the details as shown in the following table:

Field	Value
`Builder panel`	
Data source	*SACAD1_Model001*
Folder	*Public → SACAD1_30 → SACAD1 Content*
Chart Structure	*Indicator → Numeric Point*
Measure	*Gross Order Am. USD*
`Styling panel`	
Widget name	**`CH_Measure`**
Width	**`400 px`**
Height	**`200 px`**
Left (X)	**`0 px`**
Top (Y)	**`0 px`**
Scale	**`Auto-formatted`**

2. Create an invisible button with the name **`BUTTON_StartStop`** overlaying your numeric point chart using the following specifications:

Field	Value
Analytics Designer Properties	
Name	**`BUTTON_StartStop`**
Text	remove the standard text and leave the field empty
Size and Position	
Width	**`400`**
Height	**`200`**
Left	**`0`**
Top	**`0`**
Button Style	
Type	**`Lite Button`**

Task 3: Implement the Timer with the Script Logic

1. Add a new *Timer* technical widget with the default name **`Timer_1`**.
2. Write the code for timer *Timer_1* that is required to perform the following tasks:
   - Read all measures of *CH_Measure* and transform it into a one dimensional array containing the IDs only.
   - Read the current measure displayed in *CH_Measure*.
   - Identify the position of the current displayed measure within the array of measures and identify the new measure to be displayed.
   - Remove the current measure and display the new one.
   - Restart the timer again with a delay of **`5`** seconds to start the endless loop.

Task 4: Implement the Script Logic to Stop and Restart the Timer

1. Write the code for the timer *BUTTON_StartStop* that stops *Timer_1* if it is already running and starts it with a delay of 5 seconds if it is not running.

Task 5: Start the Timer when the Application Starts

1. Write the code for the *onInitialization* event that starts the timer with a delay of 5 seconds.

Task 6: Save and Test your Application Design

1. Save your application and run it.
2. Check if the displayed measure changes automatically every 5 seconds.
3. Check if the measures start from the beginning if the last one was displayed.
4. Check if you can stop and restart the changing of measures.

Unit 3
Solution 10

Create an Application with Timer Function

Business Example

A timer can be used whenever something within your application should happen after a given period of time. If the timer restarts itself you create a permanent loop.

In this exercise, we will use a numeric point chart to present one measure value at a time. When the application starts, we want to change the presented measure every 5 seconds by looping through all available measures of the data source used for the chart. The user should be able to stop and restart the timer by clicking on the numeric point.

As a click on a numeric point must be precisely on the value, you overlay the chart with an invisible button to improve the usability for your users. The button should handle the code to stop or restart the loop through the measures.

Task 1: Create a New Empty Application

1. Create a new *Analytic Application* and save it with the name **`GR### Timer`** directly into the *My Files* content area.

   a) From the Navigation Bar choose  *Analytic Applications* and then click *Application* in the *Create New* area.

   b) In the toolbar, choose the *Save* icon and choose *Save*.

   c) Type in **`GR### Timer`** as the name.

   d) Choose *OK*.

Task 2: Create the Application Structure

1. Create a numeric point chart using *SACAD1_Model001* to display the measure *Gross Order Am. USD* and name it **`CH_Measure`**.

   Use the details as shown in the following table:

Field	Value
`Builder panel`	
Data source	*SACAD1_Model001*
Folder	*Public → SACAD1_30 → SACAD1 Content*
Chart Structure	*Indicator → Numeric Point*
Measure	*Gross Order Am. USD*
`Styling panel`	
Widget name	**`CH_Measure`**
Width	**`400 px`**

Field	Value
Height	**200 px**
Left (X)	**0 px**
Top (Y)	**0 px**
Scale	**Auto-formatted**

a) Within the *Insert*-section of the toolbar, choose *Chart*.

b) Choose the model *SACAD1_Model001* that is located in the folder *Public* → *SACAD1_30* → *SACAD1 Content*.

c) Within the *Builder* panel, change the settings as shown in the following table:

Field	Value
Chart Structure	
Chart Type	**Indicator → Numeric Point**
Primary Values	Choose *Add Measure* and select **Gross Order Am. USD**.

d) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**CH_Measure**
Quick Menus	
Visible in the Runtime	**deactivate** the switch
Size and Position	
Width	**400**
Height	**200**
Left	**0**
Top	**0**
Number Format	
Scale	**Auto-formatted**

2. Create an invisible button with the name **BUTTON_StartStop** overlaying your numeric point chart using the following specifications:

Field	Value
Analytics Designer Properties	
Name	**BUTTON_StartStop**

Field	Value
Text	remove the standard text and leave the field empty
Size and Position	
Width	`400`
Height	`200`
Left	`0`
Top	`0`
Button Style	
Type	`Lite Button`

a) Within the *Insert*-section of the toolbar, choose *Add* → *Button*.

b) In the *Styling* panel, choose the settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	`BUTTON_StartStop`
Text	remove the standard text and leave the field empty
Size and Position	
Width	`400`
Height	`200`
Left	`0`
Top	`0`
Button Style	
Type	`Lite Button`

Task 3: Implement the Timer with the Script Logic

1. Add a new *Timer* technical widget with the default name `Timer_1`.

   a) Within the *Outline* panel, under *Scripting*, choose + next to *Timer*.

   b) In the *Timer Properties* panel, choose *Done*.

2. Write the code for timer *Timer_1* that is required to perform the following tasks:

   - Read all measures of *CH_Measure* and transform it into a one dimensional array containing the IDs only.
   - Read the current measure displayed in *CH_Measure*.

- Identify the position of the current displayed measure within the array of measures and identify the new measure to be displayed.
- Remove the current measure and display the new one.
- Restart the timer again with a delay of **5** seconds to start the endless loop.

a) Choose the *Edit Scripts* button next to *Timer_1* within the *Scripting* section of the *Outline* to open the script editor.

b) Add the following code:

```
var Measures = CH_Measure.getDataSource().getMeasures();

var MeasuresIDs = ArrayUtils.create(Type.string);
for (var x=0; x<Measures.length; x++){
    MeasuresIDs.push(Measures[x].id);
}

var currentMeasure = CH_Measure.getMeasures(Feed.ValueAxis);
var currentIndex = MeasuresIDs.indexOf(currentMeasure[0]);

var nextIndex = 0;
if (currentIndex === MeasuresIDs.length -1) {
    nextIndex = 0;
} else {
    nextIndex = currentIndex + 1;
}

CH_Measure.removeMeasure(MeasuresIDs[currentIndex],Feed.ValueAxis);
CH_Measure.addMeasure(MeasuresIDs[nextIndex], Feed.ValueAxis);

Timer_1.start(5);
```

Task 4: Implement the Script Logic to Stop and Restart the Timer

1. Write the code for the timer *BUTTON_StartStop* that stops *Timer_1* if it is already running and starts it with a delay of 5 seconds if it is not running.

   a) Choose the *Edit Scripts* button next to *BUTTON_StartStop* within the *Canvas* section of the *Outline* and choose the *onClick* event to open the script editor.

   b) Add the following code:

```
if (Timer_1.isRunning()) {
    Timer_1.stop();
} else {
    Timer_1.start(5);
}
```

Task 5: Start the Timer when the Application Starts

1. Write the code for the *onInitialization* event that starts the timer with a delay of 5 seconds.

   a) Choose the *Edit Scripts* button next to *Canvas* within the *Outline* and choose the *onInitialization* event to open the script editor.

   b) Add the following code:

```
Timer_1.start(5);
```

Task 6: Save and Test your Application Design

1. Save your application and run it.

a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.

b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.

2. Check if the displayed measure changes automatically every 5 seconds.
   a) 5 seconds after the application is started, the displayed measure changes.
3. Check if the measures start from the beginning if the last one was displayed.
   a) After *Number of Records* the loop should start again with *Gross Order Am. USD*.
4. Check if you can stop and restart the changing of measures.
   a) Click on the currently displayed measure. It should not change anymore and should stay as it is.
   b) Click on the currently displayed measure again. After 5 seconds, it should change to the next measure again.

Unit 4
Exercise 11

OPTIONAL: Define and use CSS in an Application

Business Example

The features and functions of an application are not the only important aspects. To make the user feel "at home", you want to adopt the corporate color theme to your application.

In this exercise, you will create a custom CSS class, assign it as global default class, and store it in an reusable theme in the end.

Task 1: Create the Application

1. Create a copy of the application *GR000 Start Use CSS* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Content* and save it with the name **`GR### Use CSS`**.

Task 2: Define the Application CSS and Assign it as Global Default Class

1. Create a Custom Class Name **`GR###`** and define Properties for the three CSS Selector Elements `.sap-custom-tab-strip-tab-bar`, `.sap-custom-tab-strip-tab` and `.sap-custom-tab-strip-selected-tab` that are related to a Tab Strip widget. In your corporate design guidelines, you find the following information regarding Tab Strip visualization: "Tab Strips use the HEX code `DCDCDC` as the background color, the labels are displayed in color `DarkSlateGrey`, and the currently used tab is displayed in `Bold` and `Italic` font."

   Note:
   Replace ### with your user name where needed.

2. Define your newly created CSS class **`GR###`** as *Global Default Class Name* in the *Canvas* styling options. Check if your style definitions are visible and the design of the Tab Strip changes accordingly.

Task 3: Load the CSS to a New Theme and Remove the Application CSS

1. Create a new theme with the name **`Theme GR###`** and load the application CSS into the theme.
2. To really "use" the CSS provided in the theme, you need to remove the CSS definition you created before from the Application CSS. Otherwise, this would still exist and "override" the CSS coming out of the theme.

Task 4: Save and Test your Application Design

1. Save and start your application.

OPTIONAL: Define and use CSS in an Application

Business Example

The features and functions of an application are not the only important aspects. To make the user feel "at home", you want to adopt the corporate color theme to your application.

In this exercise, you will create a custom CSS class, assign it as global default class, and store it in an reusable theme in the end.

Task 1: Create the Application

1. Create a copy of the application *GR000 Start Use CSS* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Content* and save it with the name **`GR### Use CSS`**.

   a) From the Navigation Bar choose  *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Content*.

   b) Select the checkbox for the application *GR000 Start Use CSS* and choose  *Copy* .

   c) Choose *My Files* to choose your private folder.

   d) Type in the name **`GR### Use CSS`** and choose *OK*.

   e) Navigate to the *My Files* area.

   f) Choose *GR### Use CSS* to open the file in edit mode.

Task 2: Define the Application CSS and Assign it as Global Default Class

1. Create a Custom Class Name **`GR###`** and define Properties for the three CSS Selector Elements `.sap-custom-tab-strip-tab-bar`, `.sap-custom-tab-strip-tab` and `.sap-custom-tab-strip-selected-tab` that are related to a Tab Strip widget. In your corporate design guidelines, you find the following information regarding Tab Strip visualization: "Tab Strips use the HEX code `DCDCDC` as the background color, the labels are displayed in color `DarkSlateGrey`, and the currently used tab is displayed in `Bold` and `Italic` font."

   Note:
   Replace ### with your user name where needed.

   a) In the *Outline* area, hover over the *Canvas* element and choose the  *Edit CSS* button.

b) Choose **`Tab Strip`** in the dropdown at the top of the editor screen to see the documentation of the available CSS selector elements and the supported properties.

c) Enter the following CSS to define the background color of the tab strip:

```
.GR### .sap-custom-tab-strip-tab-bar{
    background-color: #DCDCDC;
}
```

d) Add the following CSS to define the text color of the tabs:

```
.GR### .sap-custom-tab-strip-tab{
    color: DarkSlateGrey;
}
```

e) Add the following CSS to enhance the style definition of the selected tab:

```
.GR### .sap-custom-tab-strip-selected-tab{
    font-weight: bold;
    font-style: italic;
}
```

2. Define your newly created CSS class **`GR###`** as *Global Default Class Name* in the *Canvas* styling options. Check if your style definitions are visible and the design of the Tab Strip changes accordingly.

   a) In the *Outline* area, select the *Canvas* and open the *Styling* panel if it is not open already.

   b) Type **`GR###`** into the *Global Default Class Name* field.

   c) The Tab Strip should now reflect the definition of your CSS.

Task 3: Load the CSS to a New Theme and Remove the Application CSS

1. Create a new theme with the name **`Theme GR###`** and load the application CSS into the theme.

   a) Choose the *Theme* icon in the toolbar and select *Create Theme...*

   b) Enable the *Theme CSS*.

   c) Choose *Load Application CSS*.

   d) Choose *Save As*.

   e) Type in **`Theme GR###`** as the *Name*.

   f) Choose *OK*.

2. To really "use" the CSS provided in the theme, you need to remove the CSS definition you created before from the Application CSS. Otherwise, this would still exist and "override" the CSS coming out of the theme.

   a) In the *Outline* area, choose the *Edit CSS* button.

   b) Delete all of the existing CSS.

Task 4: Save and Test your Application Design

1. Save and start your application.

   a) Within the *File* section of the toolbar, choose the *Save* icon and select *Save*.

b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.

c) Check if your desired styling is applied to the application.

Unit 4
Exercise 12

Create a Dynamic Application Layout Using a Flow Layout Panel

Business Example

In this exercise, we will make use of a *Flow Layout Panel* to design a dynamic application layout. You will work with a pre-defined application containing colored boxes that stand for any content you can think of. We will concentrate on three different device sizes: "small", "medium", and "large" as shown in the following figure:

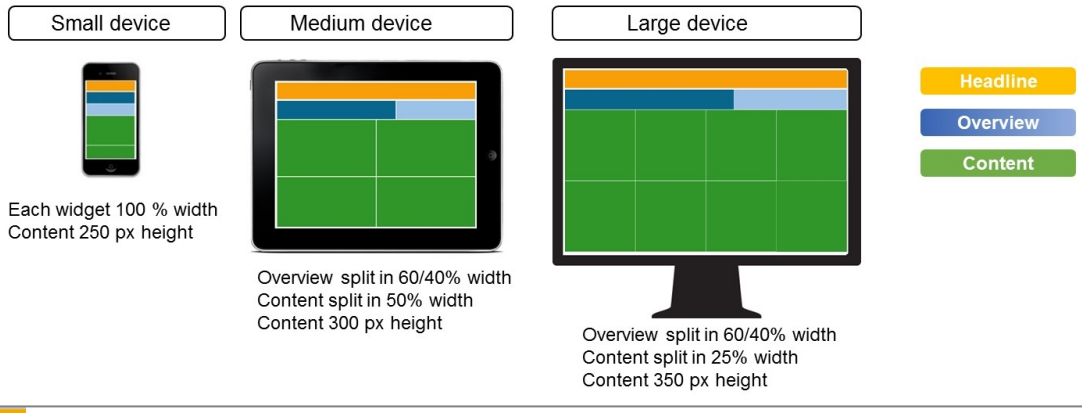


Figure 13: Results

Task 1: Create the Application

1. Create a copy of the application *GR000 Start Dynamic Layout 1* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Content* and save it with the name **`GR### Dynamic Layout 1`**.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze and understand the given application structure. In the application, you will find 11 colored panels that simulate our application content. The orange one is our application headline, the two blue ones simulate overview tiles, and the other 8 green ones simulate our application content like charts and tables, and so on. As the headline does not change within the different views we create it in a way that it stays outside the existing *Flow Layout Panel*.
2. Familiarize yourself with *Size and Position* settings of the *FlowLayoutPanel_1* widget.
3. Familiarize yourself with *Size and Position* settings of the *Overview1* and *Overview2* widgets.
4. Familiarize yourself with *Size and Position* settings of the *Content1* widget. All other content widgets are defined the same.

Task 3: Define the Dynamic Behaviour

1. Define the Breakpoint for the small devices (less or equal to 414 px screen width). All widgets should cover 100% of the available width. The overview widgets should have a height of 100px and all content widgets should have a height of 250 px.
2. Define the Breakpoint for the medium devices (less or equal to 1100 px screen width). All content widgets should cover 50% of the available width, *Overview1* should cover 60% and *Overview2* should cover 40%. The overview widgets should have a height of 100px and all content widgets should have a height of 300 px.

Task 4: Save and Test your Application Design

1. Save your application and run it.
2. Activate the *Developer tools* within *Chrome*.
3. Activate the *Device toolbar* and select different devices and orientations using the toolbar.

Create a Dynamic Application Layout Using a Flow Layout Panel

Business Example

In this exercise, we will make use of a *Flow Layout Panel* to design a dynamic application layout. You will work with a pre-defined application containing colored boxes that stand for any content you can think of. We will concentrate on three different device sizes: "small", "medium", and "large" as shown in the following figure:

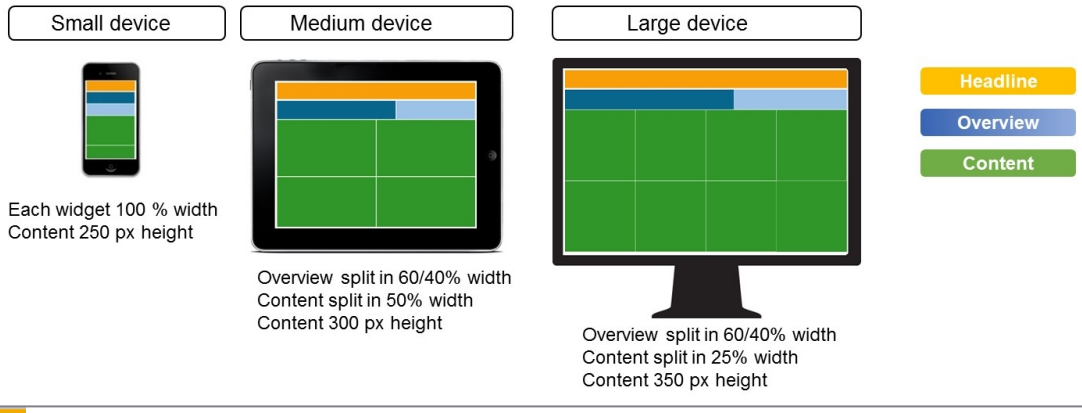


Figure 13: Results

Task 1: Create the Application

1. Create a copy of the application *GR000 Start Dynamic Layout 1* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Content* and save it with the name **`GR### Dynamic Layout 1`**.

   a) From the Navigation Bar choose *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Content*.

   b) Select the checkbox for the application *GR000 Start Dynamic Layout 1* and choose *Copy* .

   c) Choose *My Files* to choose your private folder.

   d) Type in the name **`GR### Dynamic Layout 1`** and choose *OK*.

   e) Navigate to the *My Files* area.

   f) Choose *GR### Dynamic Layout 1* to open the file in edit mode.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze and understand the given application structure. In the application, you will find 11 colored panels that simulate our application content. The orange one is our application headline, the two blue ones simulate overview tiles, and the other 8 green ones simulate our application content like charts and tables, and so on. As the headline does not change within the different views we create it in a way that it stays outside the existing *Flow Layout Panel*.
   a) Check the *Outline* area.
   b) The order of the elements within the *FlowLayoutPanel_1* is important. The first widget shown in the application is the one at the bottom of the list of nested widgets in the outline.
2. Familiarize yourself with *Size and Position* settings of the *FlowLayoutPanel_1* widget.
   a) Click on the *FlowLayoutPanel_1* widget and check the *Styling* panel in the *Designer*.
   b) It covers the remaining application area underneath the *Headline* panel.
3. Familiarize yourself with *Size and Position* settings of the *Overview1* and *Overview2* widgets.
   a) Click on the *Overview1* widget and check the *Styling* panel in the *Designer*.
   b) It takes 60% of the available width of the *FlowLayoutPanel_1* panel.
   c) Click on the *Overview2* widget and check the *Styling* panel in the *Designer*.
   d) It takes 40% of the available width of the *FlowLayoutPanel_1* panel.
4. Familiarize yourself with *Size and Position* settings of the *Content1* widget. All other content widgets are defined the same.
   a) Click on the *Content1* widget and check the *Styling* panel in the *Designer*.
   b) It takes 25% of the available width of the *FlowLayoutPanel_1* panel.
   c) The *Height* property is set to 350 px.

Task 3: Define the Dynamic Behaviour

1. Define the Breakpoint for the small devices (less or equal to 414 px screen width). All widgets should cover 100% of the available width. The overview widgets should have a height of 100px and all content widgets should have a height of 250 px.
   a) Within the *Builder* panel of *FlowLayoutPanel_1*, click on + *Add Breakpoint*.
   b) In the *When the screen width is* <= field, define **`414`** as the value.
   c) In the *Set the widget width* section, define the following value:

Each Widget	100 %

   d) In the *Set the widget height* section, define the following values. Use the + *Add Widget* button to create multiple rows.

Each Widget	250 px
Overview1	100 px
Overview2	100 px

2. Define the Breakpoint for the medium devices (less or equal to 1100 px screen width). All content widgets should cover 50% of the available width, *Overview1* should cover 60% and *Overview2* should cover 40%. The overview widgets should have a height of 100px and all content widgets should have a height of 300 px.
   a) Within the *Builder* panel of *FlowLayoutPanel_1*, click on + *Add Breakpoint*.
   b) In the *When the screen width is* <= field, define **`1100`** as the value.
   c) In the *Set the widget width* section, define the following values. Use the + *Add Widget* button to create multiple rows.

Each Widget	50 %
Overview1	60 %
Overview2	40 %

   d) In the *Set the widget height* section, define the following values. Use the + *Add Widget* button to create multiple rows.

Each Widget	300 px
Overview1	100 px
Overview2	100 px

Task 4: Save and Test your Application Design

1. Save your application and run it.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.
   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.
2. Activate the *Developer tools* within *Chrome*.
   a) Press **`Ctrl + Shift + I`** on your keyboard.
3. Activate the *Device toolbar* and select different devices and orientations using the toolbar.
   a) Press **`Ctrl + Shift + M`** on your keyboard.
   b) Select *iPhone SE* (as small device) to check if the application behaves as expected.
   c) Select *iPad Mini* (as medium device) to check if the application behaves as expected.

d) Select *iPad Air* (as large device) and switch to landscape mode using the  icon. Check if the application behaves as expected.

e) If you want, you can activate the *Show device frame* options using the  *More options* button on the right side of the device toolbar to get a more realistic preview. However, device frames are only available for a selected list of devices.

Unit 4
Exercise 13

OPTIONAL: Create a Highly Dynamic Application Layout Using Scripting

Business Example

In this exercise, we will make use of a combination of different techniques to create a highly dynamic application design. We will combine dynamic visibility, moving widgets, and a flow layout panel.

For small devices, we do not show the overview section and the most important content parts (4 out of 8) should be displayed in a pagebook.

For medium devices, we do not show the overview section and all content parts (8 out of 8) should be displayed with 50% width each.

For large devices, we want to show the overview section and all content parts (8 out of 8) should be displayed with 25% width each.

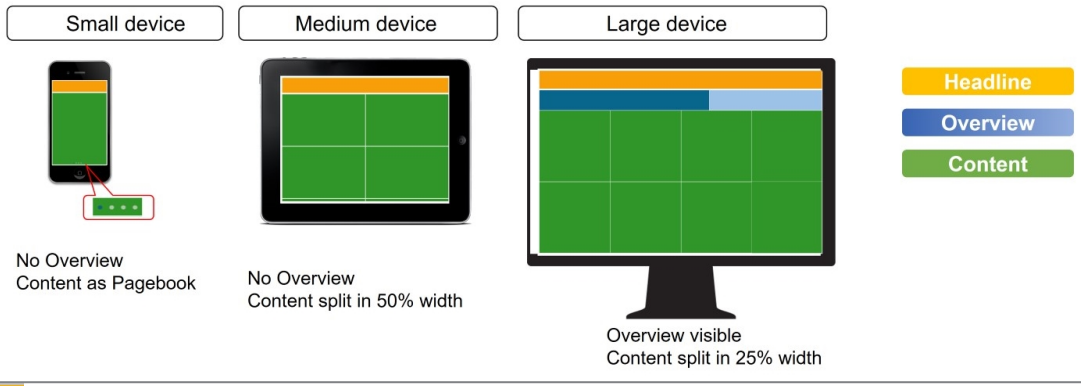


Figure 14: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Dynamic Layout 1* and save it with the new name **`GR### Dynamic Layout 2`**.

2. Alternatively you can create a copy of the application *GR000 Dynamic Layout 1* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Solutions* and save it with the name **`GR### Dynamic Layout 2`** into the folder structure *My Files*.

Task 2: Prepare the Overall Application Design

1. Add a *Page Book* widget with 4 pages to the application.

2. Adjust the styling properties of *PageBook_1* using the following values:

Field	Value
Size and Position	

Field	Value
Width	`100 %`
Height	`auto`
Left	`0 px`
Top	`100 px`
Right	`auto`
Bottom	`5 px`
Actions	
Show this item at view time	☐ `deselect`

3. As the two overview widgets (*Overview1* and *Overview2*) should only be visible in the special case of large screens, make sure these are invisible by default using the *Styling* panel.

Task 3: Create the Script to Generate the Dynamic Behaviour

1. As mentioned in the exercise description, we will need to adjust the screen design for small screens with up to 2400 pixel perimeter and large screens with over 3900 pixel perimeter. Start your *Application.onInitialization* script with calculating the perimeter and two *if statements* that will handle small and large screen adjustments.
2. We will now concentrate on the **`small`** screen design. Within the *onInitialization* script, enhance the existing script. Create a logic that processes the following actions.
   - Hide *FlowLayoutPanel_1*
   - Move *Content1* onto *Page_1*
   - Move *Content3* onto *Page_2*
   - Move *Content5* onto *Page_3*
   - Move *Content7* onto *Page_4*
   - Ensure that all content widgets use the maximum available size within the page
   - Show *PageBook_1*

   Like always, try to avoid repeating similar steps in your script over and over again. You might be able to define a list (or multiple) of elements you want to process and loop over it.
3. We now concentrate on the **`large`** screen design. Within the *onInitialization* script, enhance the existing script. Change the visibility to *true* for the widgets *Overview1* and *Overview2*.

Task 4: Save and Test your Application Design

1. To be able to simulate the correct mobile experience within the browser, you need to enable the *Mobile Support* feature in the *Analytic Application Details*.

2. Save your application and run it.
3. Activate the *Developer tools* within *Chrome* in case they are not already visible.
4. Activate the *Device toolbar* in case it is not already visible and select different devices and orientations using the toolbar.

OPTIONAL: Create a Highly Dynamic Application Layout Using Scripting

Business Example

In this exercise, we will make use of a combination of different techniques to create a highly dynamic application design. We will combine dynamic visibility, moving widgets, and a flow layout panel.

For small devices, we do not show the overview section and the most important content parts (4 out of 8) should be displayed in a pagebook.

For medium devices, we do not show the overview section and all content parts (8 out of 8) should be displayed with 50% width each.

For large devices, we want to show the overview section and all content parts (8 out of 8) should be displayed with 25% width each.

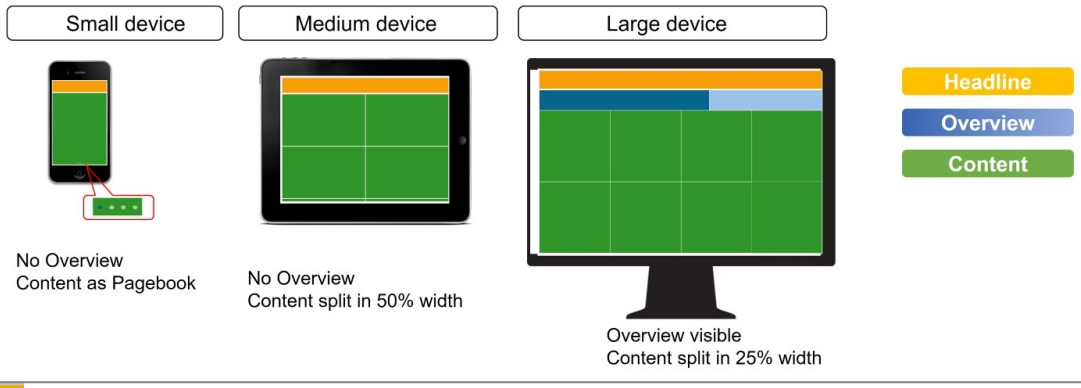


Figure 14: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Dynamic Layout 1* and save it with the new name **`GR### Dynamic Layout 2`**.
   a) In the application designer, switch to your already opened application *GR### Dynamic Layout 1*.
   b) In the toolbar, choose the *Save* icon and choose *Save as...*.
   c) Type in **`GR### Dynamic Layout 2`** as the name and choose *OK*.
2. Alternatively you can create a copy of the application *GR000 Dynamic Layout 1* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Solutions* and save it with the name **`GR### Dynamic Layout 2`** into the folder structure *My Files*.

a) From the Navigation Bar choose  *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Solutions*.

b) Select the checkbox for the application *GR000 Dynamic Layout 1* and choose  *Copy* .

c) Choose *My Files* and type in **`GR### Dynamic Layout 2`**.

d) Choose *OK*.

e) Navigate to the *My Files* area.

f) Choose *GR### Dynamic Layout 2* to open the file in edit mode.

Task 2: Prepare the Overall Application Design

1. Add a *Page Book* widget with 4 pages to the application.
   a) Within the *Outline*, select the *Canvas.*
   b) Within the *Insert*-section of the toolbar, choose *Add* → *Page Book*.
   c) Within the *Builder* panel of the *PageBook_1* widget, choose the  *Add* button twice to have four pages in the end.
2. Adjust the styling properties of *PageBook_1* using the following values:

Field	Value
Size and Position	
Width	**`100 %`**
Height	**`auto`**
Left	**`0 px`**
Top	**`100 px`**
Right	**`auto`**
Bottom	**`5 px`**
Actions	
Show this item at view time	 **`deselect`**

   a) Within the *Styling* panel of the *PageBook_1* widget, change the settings as shown in the step description.
3. As the two overview widgets (*Overview1* and *Overview2*) should only be visible in the special case of large screens, make sure these are invisible by default using the *Styling* panel.
   a) Within the *Styling* panel of the *Overview1* widget, deselect the *Show this item at view time* property.
   b) Within the *Styling* panel of the *Overview2* widget, deselect the *Show this item at view time* property.

Task 3: Create the Script to Generate the Dynamic Behaviour

1. As mentioned in the exercise description, we will need to adjust the screen design for small screens with up to 2400 pixel perimeter and large screens with over 3900 pixel perimeter. Start your *Application.onInitialization* script with calculating the perimeter and two *if statements* that will handle small and large screen adjustments.

   a) Within the *Outline* section, choose the *fx* *Edit Scripts* icon next to the *Canvas* and choose *onInitialization*.

   b) Add the following lines in the *onInitialization* script:

```
var perimeter = (Application.getInnerHeight().numberValue +
Application.getInnerWidth().numberValue) * 2;

if (perimeter < 2400) {

}

if (perimeter > 3900){

}
```

2. We will now concentrate on the **`small`** screen design. Within the *onInitialization* script, enhance the existing script. Create a logic that processes the following actions.

   - Hide *FlowLayoutPanel_1*
   - Move *Content1* onto *Page_1*
   - Move *Content3* onto *Page_2*
   - Move *Content5* onto *Page_3*
   - Move *Content7* onto *Page_4*
   - Ensure that all content widgets use the maximum available size within the page
   - Show *PageBook_1*

   Like always, try to avoid repeating similar steps in your script over and over again. You might be able to define a list (or multiple) of elements you want to process and loop over it.

   a) Add the following lines into the existing `if (perimeter < 2400) { }` statement:

```
FlowLayoutPanel_1.setVisible(false);
var elements = ArrayUtils.create(Type.Panel);
var pages = ArrayUtils.create(Type.string);
elements = [Content1, Content3, Content5, Content7];
pages = ["Page_1", "Page_2", "Page_3", "Page_4"];
for (var x=0; x<elements.length; x++){
    PageBook_1.moveWidget(pages[x], elements[x]);
    var element = elements[x];
    var layout = element.getLayout();
    layout.setHeight(LayoutValue.Auto);
    layout.setWidth(LayoutValue.Auto);
```

```
    layout.setBottom(LayoutValue.create(0, LayoutUnit.Pixel));
    layout.setTop(LayoutValue.create(0, LayoutUnit.Pixel));
    layout.setLeft(LayoutValue.create(0, LayoutUnit.Pixel));
    layout.setRight(LayoutValue.create(0, LayoutUnit.Pixel));
}
PageBook_1.setVisible(true);
```

3. We now concentrate on the **`large`** screen design. Within the *onInitialization* script, enhance the existing script. Change the visibility to *true* for the widgets *Overview1* and *Overview2*.
   a) Add the following lines into the existing `if (perimeter > 3900) { }` statement:

```
    Overview1.setVisible(true);
    Overview2.setVisible(true);
```

Task 4: Save and Test your Application Design

1. To be able to simulate the correct mobile experience within the browser, you need to enable the *Mobile Support* feature in the *Analytic Application Details*.
   a) Within the *File*-section of the toolbar, choose the *Edit Analytic Application* icon and select *Analytic Application Details*.
   b) Activate the *Enable Mobile Support* feature.
   c) Choose *Save*.
2. Save your application and run it.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.
   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.
3. Activate the *Developer tools* within *Chrome* in case they are not already visible.
   a) Press **`Ctrl + Shift + I`** on your keyboard.
4. Activate the *Device toolbar* in case it is not already visible and select different devices and orientations using the toolbar.
   a) Press **`Ctrl + Shift + M`** on your keyboard.
   b) Select *iPhone SE* (as small device) and choose the context menu of the reload icon in the Chrome browser to select *Empty Cache and Hard Reload* to check if the application behaves as expected.

   Note:
   You may have to use *Empty Cache and Hard Reload* more than once to see the desired result.

   c) Select *iPad Mini* (as medium device) and choose the context menu of the reload icon in the Chrome browser to select *Empty Cache and Hard Reload* to check if the application behaves as expected.
   d) Select *iPad Air* (as large device) and switch to landscape mode using the  icon.
   Choose the context menu of the reload icon in the Chrome browser to select *Empty Cache and Hard Reload* to check if the application behaves as expected.

Unit 6
Exercise 14

OPTIONAL: Export an Application to PDF

Business Example

Your end users want to export their analytic application to PDF for offline viewing. They also want the flexibility to specify custom headers, footers for the generated PDF document, and select the page orientation. Of course they do not want to see the settings section in the exported PDF in the end.

In this exercise, you will use the PDF exporting APIs to export an application to PDF. You will explore various PDF exporting options, and provide user control to set some of these options at runtime prior to generating the PDF document.

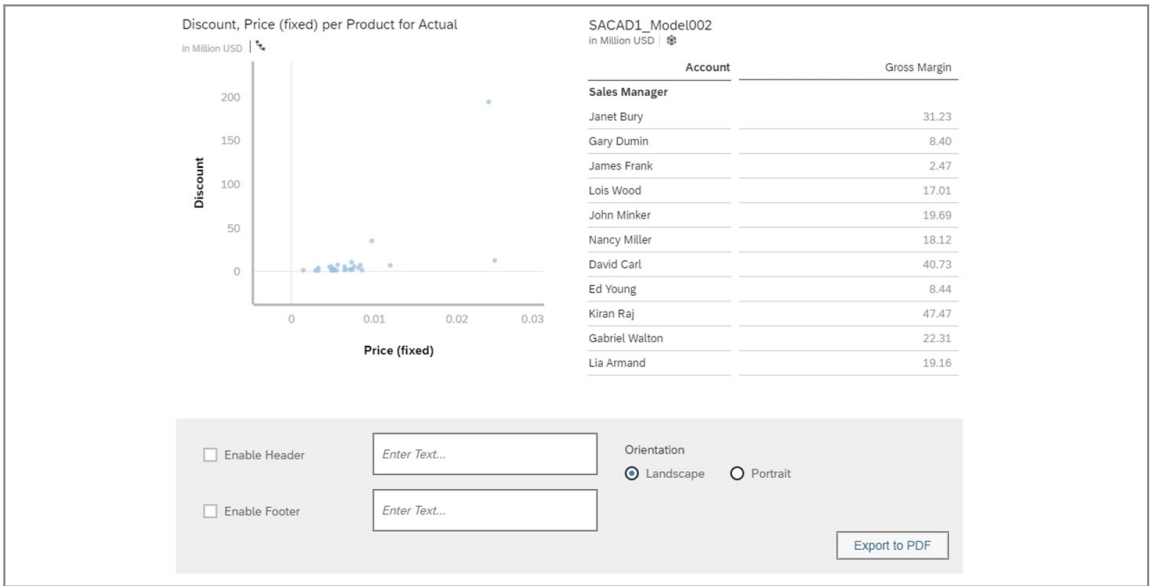


Figure 15: Result

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Export an application to PDF* from the *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Export an application to PDF`** into the folder structure *My Files*.
2. Open *GR### Export an application to PDF* for editing.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze the widgets within the *PNL_ExportToPDF* panel. We will add script to these widgets to demonstrate the PDF export feature.

Task 3: Add the PDF Export Technical Widget

1. Add the *Export to PDF* technical widget with default name and set the *Paper Size* to **`A4`**, *Exported File Name* to **`SACAD1_PDF_Export`** and *Page Number Location* to **`Footer`**.

Task 4: Set PDF Export Options at Runtime

1. Write the code that is required to set a header in the exported PDF file if the *Enable Header* checkbox is checked. Use the *INF_Header* input field to define the header text.
2. Write the code that is required to set a footer in the exported PDF file if the *Enable Footer* checkbox is checked. Use the *INF_Footer* input field to define the footer text.
3. Set the orientation of export PDF document to *Landscape or Portrait* based on user selection in *RB_Orientation* radio button group.
4. Write the code for *BTN_ExportPDF* to export the application to PDF.

Task 5: Save and Test your Application Design

1. Save your application.
2. Generate a PDF export of the application leveraging the PDF export options.

OPTIONAL: Export an Application to PDF

Business Example

Your end users want to export their analytic application to PDF for offline viewing. They also want the flexibility to specify custom headers, footers for the generated PDF document, and select the page orientation. Of course they do not want to see the settings section in the exported PDF in the end.

In this exercise, you will use the PDF exporting APIs to export an application to PDF. You will explore various PDF exporting options, and provide user control to set some of these options at runtime prior to generating the PDF document.

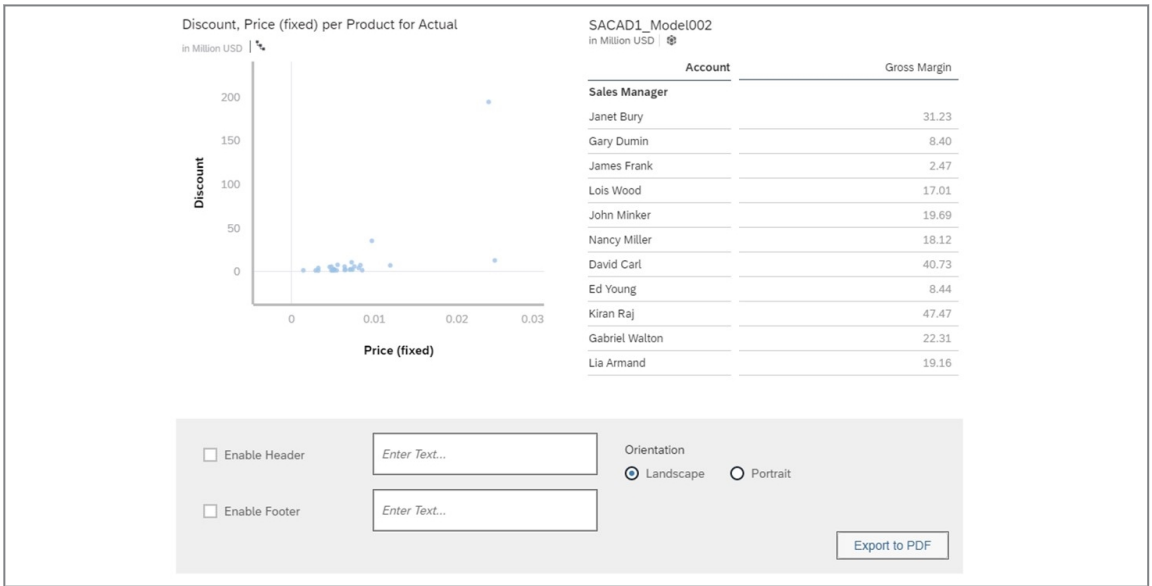


Figure 15: Result

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Export an application to PDF* from the *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Export an application to PDF`** into the folder structure *My Files*.

   a) From the Navigation Bar choose *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Content*.

   b) Select *GR000 Start Export an application to PDF* and choose the *Copy* icon.

   c) Choose *My Files* and type in **`GR### Export an application to PDF`** as the name.

   d) Choose *OK*.

2. Open *GR### Export an application to PDF* for editing.

a) Navigate to the *My Files* area.

b) Choose *GR### Export an application to PDF* to open the file in edit mode.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze the widgets within the *PNL_ExportToPDF* panel. We will add script to these widgets to demonstrate the PDF export feature.
   a) Check the different checkboxes and the radio button group which will be used to control PDF exporting options.

Task 3: Add the PDF Export Technical Widget

1. Add the *Export to PDF* technical widget with default name and set the *Paper Size* to **`A4`**, *Exported File Name* to **`SACAD1_PDF_Export`** and *Page Number Location* to **`Footer`**.
   a) Within the *Outline* panel, under *Scripting*, choose + next to *Export to PDF*.

   b) Set the properties for the *ExportToPDF_1* technical widget as shown in the following table:

Field	Value
Properties	
click on *Included Widgets*	**`deselect`** the whole node *PNL_Export-ToPDF* with all widgets below
General Settings	
Paper Size	**`A4`**
Page Number Location	**`Footer`**
Exported File Name	**`SACAD1_PDF_Export`**

   c) Choose *Done*.

Task 4: Set PDF Export Options at Runtime

1. Write the code that is required to set a header in the exported PDF file if the *Enable Header* checkbox is checked. Use the *INF_Header* input field to define the header text.
   a) Click on the *INF_Header* input field within the canvas area. In the upcoming menu, choose *Edit Scripts* to open the script editor.

   b) Add the following script in the script editor:

```
ExportToPDF_1.setHeaderText(INF_Header.getValue());
```

   c) Click on the *CB_Header* checkbox within the canvas area. In the upcoming menu, choose *Edit Scripts* to open the script editor.

   d) Add the following script in the script editor:

```
var length = CB_Header.getSelectedKeys().length;

if(length > 0){
    ExportToPDF_1.setHeaderVisible(true);
}else{
    ExportToPDF_1.setHeaderVisible(false);
}
```

2. Write the code that is required to set a footer in the exported PDF file if the *Enable Footer* checkbox is checked. Use the *INF_Footer* input field to define the footer text.
   a) Click on the *INF_Footer* input field within the canvas area. In the upcoming menu, choose *Edit Scripts* to open the script editor.
   b) Add the script below in the script editor:

```
ExportToPDF_1.setFooterText(INF_Footer.getValue());
```

   c) Click on the *CB_Footer* checkbox within the canvas area. In the upcoming menu, choose *Edit Scripts* to open the script editor.
   d) Add the following script in the script editor:

```
var length = CB_Footer.getSelectedKeys().length;

if(length > 0){
    ExportToPDF_1.setFooterVisible(true);
}else{
    ExportToPDF_1.setFooterVisible(false);
}
```

3. Set the orientation of export PDF document to *Landscape or Portrait* based on user selection in *RB_Orientation* radio button group.
   a) Click on the *RB_Orientation* radio button group within the canvas area. In the upcoming menu, choose *Edit Scripts* to open the script editor.
   b) Add the following script in the script editor:

```
var layout = RB_Orientation.getSelectedKey();
if(layout === 'Landscape'){
    ExportToPDF_1.setPageOrientation(PageOrientation.Landscape);
}else if(layout === 'Portrait'){
    ExportToPDF_1.setPageOrientation(PageOrientation.Portrait);
}
```

4. Write the code for *BTN_ExportPDF* to export the application to PDF.
   a) Click on the *BTN_ExportPDF* button within the canvas area. In the upcoming menu, choose *Edit Scripts* and select *onClick* to open the script editor.
   b) Add the following script in the script editor:

```
ExportToPDF_1.exportView();
```

Task 5: Save and Test your Application Design

1. Save your application.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.
   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.
2. Generate a PDF export of the application leveraging the PDF export options.
   a) Type in **`My PDF Header`** in the header input field and **`My PDF Footer`** in the footer input field.
   b) Check **`Enable Header`** and **`Enable Footer`** checkboxes.
   c) Set the orientation to **`Landscape`**.
   d) Choose the *Export to PDF* button to generate the PDF file.

e) Review the generated PDF document and notices the header, footers are added to the page.

Unit 6
Exercise 15

OPTIONAL: Configure Scheduling and Notification Options for an Application

Business Example

As an application designer, you can schedule an application to run at specified time interval and automatically generate a PDF export of the application. This PDF export can be sent to SAP Analytics Cloud or external users. SAP Analytics Cloud, analytics designer provides a scheduling event which can be used to execute a set of scripts only when the application is scheduled. This event provides a great deal of flexibility to the application designer, for example, they can choose to display or hide specific application widgets from the application in the generated PDF document. During the scheduling phase, the designer can also trigger notifications to be sent to SAP Analytics Cloud or the user's email address. Notifications allow you to send out an alert if a particular KPI is below the expected threshold.

In this exercise, you will use the scheduling and notifications APIs to schedule an application to run at specified time interval.

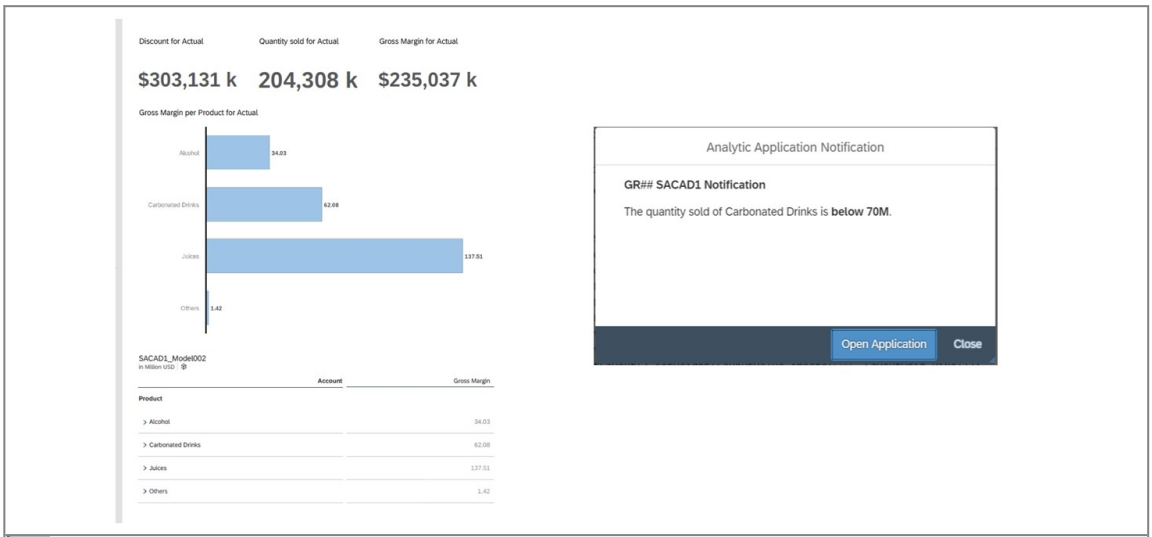


Figure 16: Result

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Scheduling your application* from the *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Scheduling your application`** into the folder structure *My Files*.

2. Open *GR### Scheduling your application* for editing.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze the *TS_Content* tab strip and the widgets contained within it. Pay attention to the size and position of the tab strip. We will dynamically move and position these widgets while scheduling the application.

2. Analyze the script variable *displayHeader*.

Task 3: Configure Scheduling and Notification Options for an Application

1. Write the required code to perform the following tasks on application *onInitialization* event:
   - Hide the *PANEL_Header* panel if the script variable *displayHeader* is set to **`false`**.
   - Move *CH_GrossMargin* chart and *TBL_GrossMargin* out of the *TS_Content* tabstrip. Position the chart **`10px`** below the panel *PANEL_Tiles* and the table **`10px`** directly below the chart. Hide the tab strip. These steps should be executed only when the application is scheduled.
2. Append the code to the application *onInitialization* event to send a custom notification to the user. The code will set the notification settings as shown in the following table:

Field	Value
"title"	**`GR## SACAD1 Notification`** where ## is your assigned user id
"isSendEmail"	**`false`**
"mode"	**`ApplicationMode.Present`**
"parameters"	Set *displayHeader* script variable to **`false`**
"receivers"	**`A## or B##`** where ## is your assigned user id
"content"	**`"The quantity sold of Carbonated Drinks is <b>below 70M</b>. "`**

Task 4: Save and Test your Application Design

1. Save your application.
2. Schedule the application to run with the properties as shown in the following table and view the output:

Field	Value
Start	**`10 minutes from current time`**
Email Subject	**`GR## SACAD1 Scheduling`**
Email Message	**`GR## SACAD1 Scheduling`**
Include Analytic Application Link	☑
Customize Script Variable	**`ON`**
Script Variable	**`displayHeader`** with *Default Value* set to **`false`**
SAP Analytics Cloud Recipients	**`A## or B##`** where ## is your assigned user id (select from list)

Field	Value
Non-SAP Analytics Cloud Recipients	**`Your own email address`**

Note:
Each SAP Analytics Cloud tenant has a limited number of scheduling slots. You might get an error message when you try to create your scheduling job. In this case, all slots for the selected time-frame are used already by other users. Try to choose another time in one or two hours from now on.

3. Check the PDF output sent to your email.
4. View the notification sent to the user in SAP Analytics Cloud.

OPTIONAL: Configure Scheduling and Notification Options for an Application

Business Example

As an application designer, you can schedule an application to run at specified time interval and automatically generate a PDF export of the application. This PDF export can be sent to SAP Analytics Cloud or external users. SAP Analytics Cloud, analytics designer provides a scheduling event which can be used to execute a set of scripts only when the application is scheduled. This event provides a great deal of flexibility to the application designer, for example, they can choose to display or hide specific application widgets from the application in the generated PDF document. During the scheduling phase, the designer can also trigger notifications to be sent to SAP Analytics Cloud or the user's email address. Notifications allow you to send out an alert if a particular KPI is below the expected threshold.

In this exercise, you will use the scheduling and notifications APIs to schedule an application to run at specified time interval.

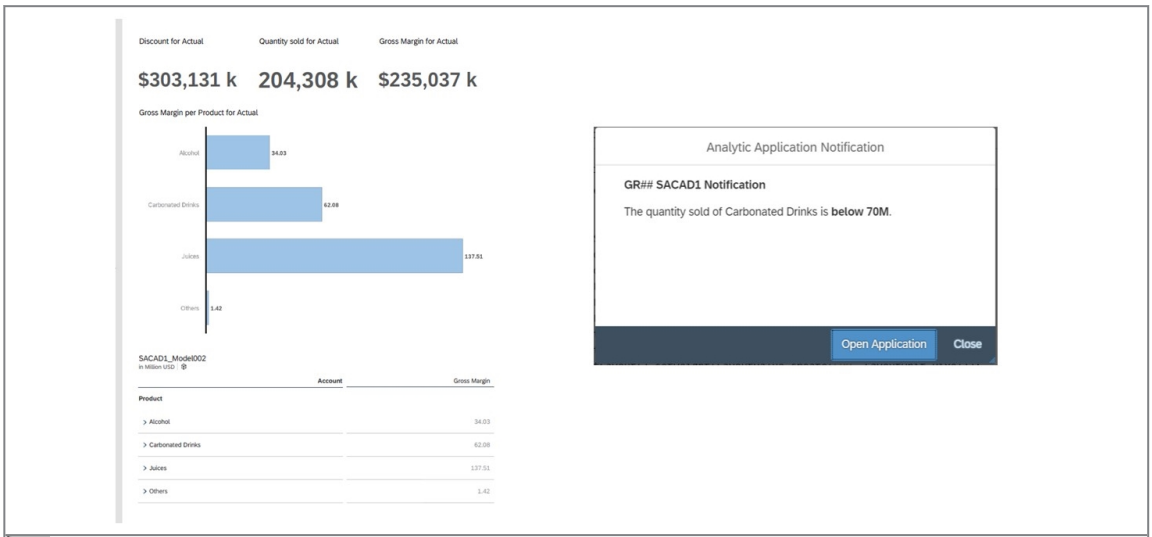


Figure 16: Result

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Scheduling your application* from the *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Scheduling your application`** into the folder structure *My Files*.

   a) From the Navigation Bar choose *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Content*.

   b) Select *GR000 Start Scheduling your application* and choose the *Copy* icon.

c) Choose *My Files* and type in **`GR### Scheduling your application`** as the name.

d) Choose *OK*.

2. Open *GR### Scheduling your application* for editing.

a) Navigate to the *My Files* area.

b) Choose *GR### Scheduling your application* to open the file in edit mode.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze the *TS_Content* tab strip and the widgets contained within it. Pay attention to the size and position of the tab strip. We will dynamically move and position these widgets while scheduling the application.

a) Choose the *TS_Content* tab strip and within the *Styling* panel, review its **`Height`** and **`Top`** property.

2. Analyze the script variable *displayHeader*.

a) Under *Scripting* , click on the *displayHeader* script variable. Notice the **`Type`** , **`Default Value`**, and **`Expose variable via URL parameter`** settings. This script variable will be used in the application to dynamically control the display of *PANEL_Header*.

Task 3: Configure Scheduling and Notification Options for an Application

1. Write the required code to perform the following tasks on application *onInitialization* event:

- Hide the *PANEL_Header* panel if the script variable *displayHeader* is set to **`false`**.
- Move *CH_GrossMargin* chart and *TBL_GrossMargin* out of the *TS_Content* tabstrip. Position the chart **`10px`** below the panel *PANEL_Tiles* and the table **`10px`** directly below the chart. Hide the tab strip. These steps should be executed only when the application is scheduled.

a) Click on *Canvas* within the *Outline* area and choose the *Edit Scripts → onInitialization* button to start the script editor.

b) Enter the following lines of code in the script editor:

```
if (displayHeader === "false" ){
        PANEL_Header.setVisible(false);
    }
if (Scheduling.isRunBySchedulePublication()){
    Application.moveWidget(CH_GrossMargin);
    Application.moveWidget(TBL_GrossMargin);
    CH_GrossMargin.getLayout().setBottom(LayoutValue.Auto);
    CH_GrossMargin.getLayout().setTop(LayoutValue.create(250,
LayoutUnit.Pixel));
    CH_GrossMargin.getLayout().setHeight(LayoutValue.create(500,
LayoutUnit.Pixel));
    TBL_GrossMargin.getLayout().setBottom(LayoutValue.Auto);
    TBL_GrossMargin.getLayout().setTop(LayoutValue.create(760,
LayoutUnit.Pixel));
    TBL_GrossMargin.getLayout().setHeight(LayoutValue.create(500,
LayoutUnit.Pixel));
    TS_Content.setVisible(false);

}
```

2. Append the code to the application *onInitialization* event to send a custom notification to the user. The code will set the notification settings as shown in the following table:

Field	Value
"title"	**`GR## SACAD1 Notification`** where ## is your assigned user id
"isSendEmail"	**`false`**
"mode"	**`ApplicationMode.Present`**
"parameters"	Set *displayHeader* script variable to **`false`**
"receivers"	**`A## or B##`** where ## is your assigned user id
"content"	**`"The quantity sold of Carbonated Drinks is <b>below 70M</b>. "`**

   a) Click on *Canvas* within the *Outline* area and choose the *Edit Scripts → onInitialization* button to start the script editor.

   b) Append the following lines of code in the script editor just below the line **`TS_Content.setVisible(false);`**. Make sure to substitute **`##`** in the *title* and *receivers* properties with your assigned user ID.

```
Application.sendNotification({
    "title": "GR## SACAD1 Notification",
    "content": "The quantity sold of Carbonated Drinks is <b>below
70M</b>.<br>",
    "isSendEmail": false,
    "receivers": ["A##"],
    "mode": ApplicationMode.Present,
    "parameters": [UrlParameter.create("p_displayHeader", "false")]
    });
```

Note:
The notification is also sent out to the email address of the user specified in the receivers property provided the *isSendEmail* is set to **`true`**. In this training environment, SAP Analytics Cloud users do not have valid email addresses, hence it is set to **`false`**. It is also possible to use Navigation APIs to embed a URL in the content body of the notification.

Task 4: Save and Test your Application Design

1. Save your application.
   a) Within the *File* section of the toolbar, choose the *Save* icon and select *Save*.
   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.
2. Schedule the application to run with the properties as shown in the following table and view the output:

Field	Value
Start	**`10 minutes from current time`**
Email Subject	**`GR## SACAD1 Scheduling`**
Email Message	**`GR## SACAD1 Scheduling`**
Include Analytic Application Link	☑
Customize Script Variable	**`ON`**
Script Variable	**`displayHeader`** with *Default Value* set to **`false`**
SAP Analytics Cloud Recipients	**`A## or B##`** where ## is your assigned user id (select from list)
Non-SAP Analytics Cloud Recipients	**`Your own email address`**

Note:
Each SAP Analytics Cloud tenant has a limited number of scheduling slots. You might get an error message when you try to create your scheduling job. In this case, all slots for the selected time-frame are used already by other users. Try to choose another time in one or two hours from now on.

a) Hover over the top of the application until the menu bar is visible.

b) Choose the *Schedule Publication* icon.

c) In the *Schedule Publication* dialog, configure the scheduling properties as outlined in the exercise step description.

d) Choose *Create*.

3. Check the PDF output sent to your email.

a) Open the PDF document sent to your email.

b) Validate that the header panel is not visible in the generated PDF output.

c) Notice that the table is visible below the chart and the tab strip is not shown in the PDF output.

4. View the notification sent to the user in SAP Analytics Cloud.

a) Hover over the top of the application until the menu bar is visible.

b) Use *Edit Analytic Application* to switch from runtime mode to edit mode.

c) Choose the *Notifications* icon on the top right of the page.

d) Choose *GR## SACAD1 Notification* to view the notification details.

Unit 6
Exercise 16

OPTIONAL: Use Bookmarks in an Application

Business Example

As an application designer, you can allow users to save either a personal or a global bookmark during runtime. Bookmarks allow end users to easily save the navigation state of an application and access the saved state for ease of analysis. For example, you have an application that has filters applied, or certain widgets hidden during the execution by the end user. You do not want your end users to spend time getting to the same navigation state each time they run the application. You would like them to open the saved state of the application, see one scenario, and then perhaps quickly switch to another scenario.

In this exercise, you will use the bookmarking APIs to save, load, and delete bookmarks.

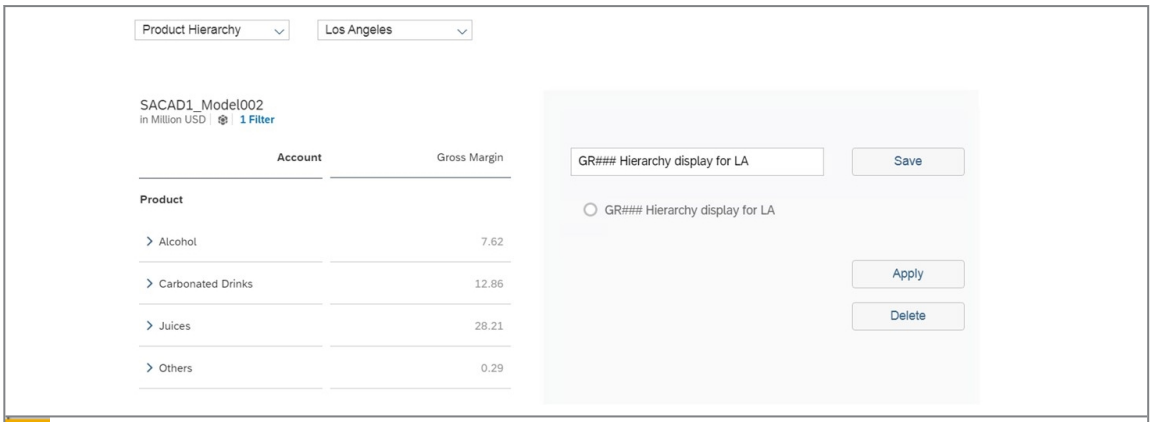


Figure 17: Result

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Use bookmarking in an application* from the *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Use bookmarking in an application`** into the folder structure *My Files*.
2. Open *GR### Use bookmarking in an application* for editing.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze the widgets within the *PANEL_Bookmarking* panel. We will add script to these widgets to demonstrate the Bookmarking feature.
2. Analyze the *DD_Cities* dropdown used to set the filter on the *Location* dimension.
3. Analyze the *DD_Hierarchy* dropdown used to show either a flat list or hierarchical view of the *Product* dimension in the table.
4. Analyze the scripts executed on launch of the application.

Task 3: Add the Bookmarking Technical Component

1. Add the *Bookmark Set* technical widget with the default name. Exclude all components in the bookmark set except **`RB_Bookmarks`**, **`DD_Cities`**, **`DD_Hiearchy`**, and **`TBL_Bookmarking`**.

Task 4: Create Bookmarks

1. Write the code for the *BUTTON_Save* button that is required to perform the following tasks:
   - Create a new personal bookmark using the *INF_Bookmark* input field to define the bookmark name. Overwrite the existing bookmarks that may have already been created with the same name.
   - Populate *RB_Bookmarks* with the bookmark ID and name.

Task 5: Read and Delete Bookmarks

1. Write the code required to read all application bookmarks on application startup and populate *RB_Bookmarks*.
2. Write the code required for *BUTTON_Delete* to delete the application bookmark selected from the RadioButtonGroup *RB_Bookmarks*. Update the RadioButtonGroup after bookmark deletion.

Task 6: Open Bookmarks

1. Write the code required for *BUTTON_Apply* to open the bookmark selected from the radio button *RB_Bookmarks*.

Task 7: Save and Test your Application Design

1. Save your application.
2. Change the table display to show **`Product Hierarchy`** and set the filter to **`Los Angeles`**. Save the view as a bookmark called **`GR### Hierarchy display for LA`**. Select **`San Francisco`** from the dropdown and save another view called **`GR### Hierarchy display for San Francisco`**.
3. Apply the bookmarks created earlier.
4. Delete the bookmark created earlier.

OPTIONAL: Use Bookmarks in an Application

Business Example

As an application designer, you can allow users to save either a personal or a global bookmark during runtime. Bookmarks allow end users to easily save the navigation state of an application and access the saved state for ease of analysis. For example, you have an application that has filters applied, or certain widgets hidden during the execution by the end user. You do not want your end users to spend time getting to the same navigation state each time they run the application. You would like them to open the saved state of the application, see one scenario, and then perhaps quickly switch to another scenario.

In this exercise, you will use the bookmarking APIs to save, load, and delete bookmarks.

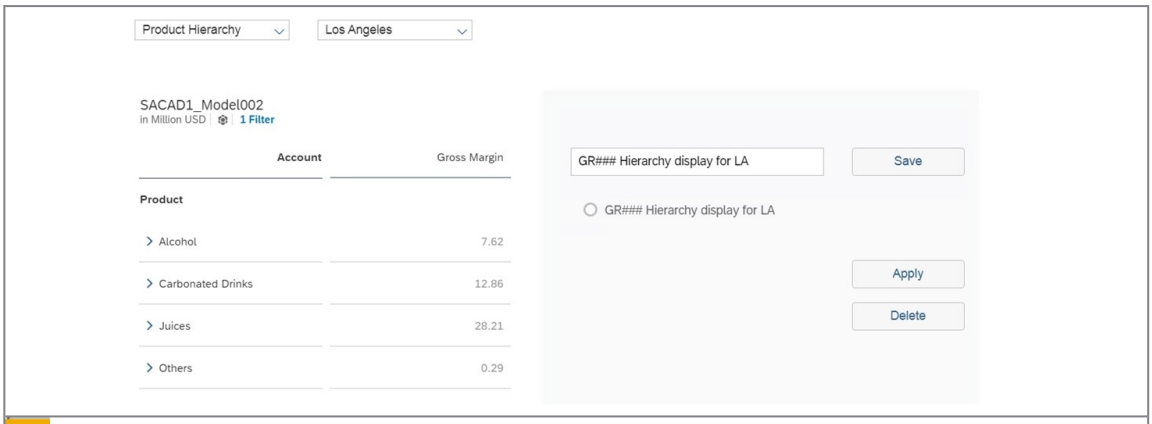


Figure 17: Result

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Use bookmarking in an application* from the *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Use bookmarking in an application`** into the folder structure *My Files*.

   a) From the Navigation Bar choose  *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Content*.

   b) Select *GR000 Start Use bookmarking in an application* and choose the  *Copy* icon.

   c) Choose *My Files* and type in **`GR### Use bookmarking in an application`** as the name.

   d) Choose *OK*.

2. Open *GR### Use bookmarking in an application* for editing.

   a) Navigate to the *My Files* area.

   b) Choose *GR### Use bookmarking in an application* to open the file in edit mode.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze the widgets within the *PANEL_Bookmarking* panel. We will add script to these widgets to demonstrate the Bookmarking feature.
   a) Check the different buttons which will be used to Save, Delete, and Apply bookmarks.
   b) Notice the *RB_Bookmarks* radio button which will be used to display application bookmarks.
   c) Notice the *INF_Bookmark* input field which will be used to specify a bookmark name.
2. Analyze the *DD_Cities* dropdown used to set the filter on the *Location* dimension.
   a) Choose the *DD_Cities* dropdown within the canvas area. In the upcoming menu, choose *Edit Scripts* to open the script editor.
   b) Review the existing script to understand how the dimension filter is applied to the table.
3. Analyze the *DD_Hierarchy* dropdown used to show either a flat list or hierarchical view of the *Product* dimension in the table.
   a) Choose the *DD_Hierarchy* dropdown within the canvas area. In the upcoming menu, choose *Edit Scripts* to open the script editor.
   b) Review the existing script to understand how the table display is changed to show the *Product* dimension as a flat list or in hierarchical format.
4. Analyze the scripts executed on launch of the application.
   a) Click on the *Canvas* from the outline area. In the upcoming menu, choose *Edit Scripts* → *onInitialization* to open the script editor.
   b) Review the existing script to populate the *DD_Cities* dropdown with *Location* dimension members.

Task 3: Add the Bookmarking Technical Component

1. Add the *Bookmark Set* technical widget with the default name. Exclude all components in the bookmark set except **`RB_Bookmarks`**, **`DD_Cities`**, **`DD_Hiearchy`**, and **`TBL_Bookmarking`**.
   a) Within the *Outline* panel, under *Scripting*, choose + next to *Bookmark Set*.
   b) Within the bookmark set properties, choose the *Included Components* button. Select the components as shown in the following table:

Component	Value
RB_Bookmarks	☑
DD_Cities	☑
DD_Hierarchy	☑
TBL_Bookmarking	☑

> Note:
> Application components that are excluded from the bookmark set will not maintain their state in the bookmark. They keep their state from the original application.

c) Choose *OK*.

d) Choose *Done*.

Task 4: Create Bookmarks

1. Write the code for the *BUTTON_Save* button that is required to perform the following tasks:

   - Create a new personal bookmark using the *INF_Bookmark* input field to define the bookmark name. Overwrite the existing bookmarks that may have already been created with the same name.
   - Populate *RB_Bookmarks* with the bookmark ID and name.

   a) Choose the *BUTTON _Save* button within the canvas area and choose *Edit Scripts → onClick* to open the script editor.

   b) Enter the following script in the script editor:

```
if (INF_Bookmark.getValue().length > 2) {
    var newBookmark = BookmarkSet_1.save(INF_Bookmark.getValue(),
false, true);
    RB_Bookmarks.addItem(newBookmark.id, newBookmark.name);
}
```

Task 5: Read and Delete Bookmarks

1. Write the code required to read all application bookmarks on application startup and populate *RB_Bookmarks*.

   a) Click on *Canvas* from the outline area. In the upcoming menu, choose *Edit Scripts → onInitialization* to open the script editor.

   b) Append the following code lines to the existing ones in the script editor:

```
RB_Bookmarks.removeAllItems();
var allBookmarks = BookmarkSet_1.getAll();
for (var j=0; j < allBookmarks.length; j++){
    RB_Bookmarks.addItem(allBookmarks[j].id, allBookmarks[j].name);
}
```

2. Write the code required for *BUTTON_Delete* to delete the application bookmark selected from the RadioButtonGroup *RB_Bookmarks*. Update the RadioButtonGroup after bookmark deletion.

   a) Choose the *BUTTON_Delete* button within the canvas area. In the upcoming menu, choose *Edit Scripts → onClick* to open the script editor.

   b) Add the following script in the script editor:

```
var bookmarkid = RB_Bookmarks.getSelectedKey();
if (bookmarkid !== undefined){
    BookmarkSet_1.deleteBookmark(bookmarkid);
```

```
    RB_Bookmarks.removeItem(bookmarkid);
}
```

Task 6: Open Bookmarks

1. Write the code required for *BUTTON_Apply* to open the bookmark selected from the radio button *RB_Bookmarks*.
   a) Choose the *BUTTON_Apply* button within the canvas area. In the upcoming menu, choose *Edit Scripts* → *onClick* to open the script editor.
   b) Add the following script in the script editor:

```
var bookmarkid = RB_Bookmarks.getSelectedKey();
if (bookmarkid !== undefined){
    BookmarkSet_1.apply(bookmarkid);
}
```

Task 7: Save and Test your Application Design

1. Save your application.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.
   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.
2. Change the table display to show **`Product Hierarchy`** and set the filter to **`Los Angeles`**. Save the view as a bookmark called **`GR### Hierarchy display for LA`**. Select **`San Francisco`** from the dropdown and save another view called **`GR### Hierarchy display for San Francisco`**.
   a) Select **`Product Hierarchy`** from the *DD_Hierarchy* dropdown.
   b) Select **`Los Angeles`** from the *DD_Cities* dropdown.
   c) Type **`GR### Hierarchy display for LA`** in the *Enter Bookmark Name (min. 3 characters)* input field and choose *Save*.
   d) Use the dropdown to switch the city to **`San Francisco`**.
   e) Type **`GR### Hierarchy display for San Francisco`** in the *Enter Bookmark Name (min. 3 characters)* input field and choose *Save*.
3. Apply the bookmarks created earlier.
   a) Select **`GR### Hierarchy display for LA`** from the radio button and choose the *Apply* button. Notice the table display is hierarchical and filtered for Los Angeles.
   b) Select **`GR### Hierarchy display for San Francisco`** from the radio button and choose the *Apply* button to apply the second bookmark. Notice the table is now filtered for San Francisco.
4. Delete the bookmark created earlier.
   a) Select **`GR### Hierarchy display for San Francisco`** from the radio button and choose *Delete*. The application bookmark is deleted and removed from the bookmarks list.

Unit 7
Exercise 17

OPTIONAL: Create a Dynamic R Visualization

Business Example

Some users approach you and ask for very special visualizations and you realize that you cannot achieve it with the standard chart widget. So you ask for support from a data science colleague. He explains to you that the needed visualization is easily created using the embedded R-server within SAP Analytics Cloud. He provides you with the few simple code lines that are needed to get the visualizations done.

In this exercise, you will implement the provided R-script code into a predefined application. You will enhance the application with a filtering functionality and use an input parameter to switch between two different visualizations within the R-script.

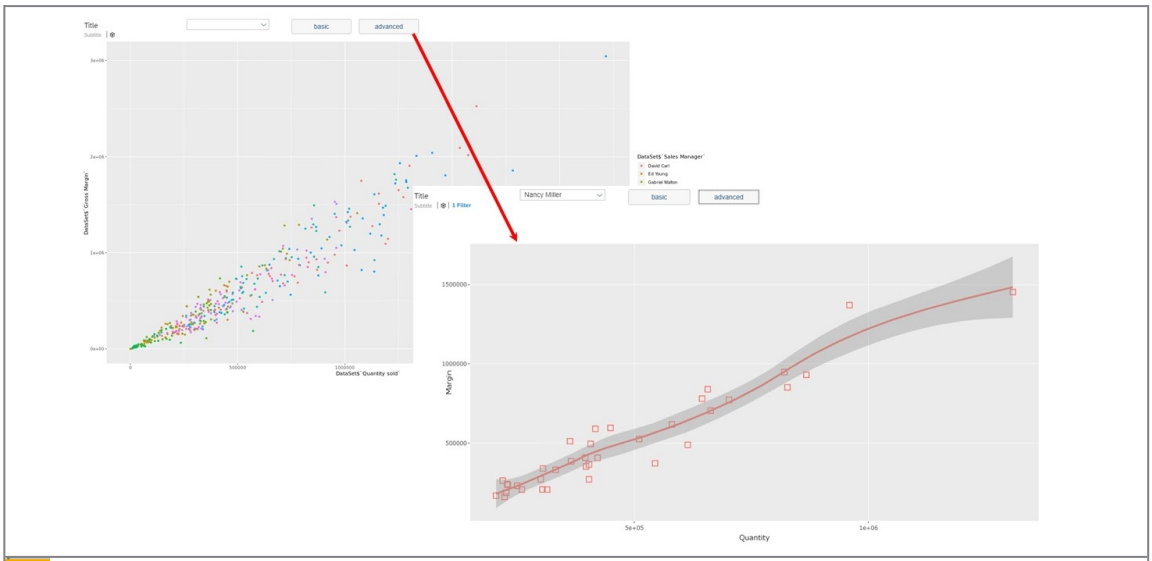


Figure 18: Result

Task 1: Create the Application

1. Create a copy of the application *GR000 Start R visualization* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Content* and save it with the name **`GR### R visualization`**.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze and understand the given application structure. You will find an existing empty R visualization, a dropdown box that will be filled whenever the result set of the R visualization changes, and two buttons.

Task 3: Create the Basic R Visualization

1. Define **`SACAD1_Model002`** as input data for the R visualization and use **`Date`** and **`Sales Manager`** in the *Rows*.

2. Rename the input data object you created in the previous step to **`DataSet`**.

3. Define the R script to create a simple scatter chart displaying the measures **`Quantity sold`** on the x-axis and **`Gross Margin`** on the y-axis. The data points for each **`Sales Manager`** should be visualized in a different color. Use the **`ggplot2`** library to generate the chart.

Note:
Your data science colleague who is familiar with R script provided the following lines of code to generate a basic scatter chart:

```
library(ggplot2)
library(plotly, warn.conflicts = FALSE)
ggplot(DataSet,  aes(x = `Quantity sold`, y = `Gross
Margin`, color = `Sales Manager`)) +
      geom_point()
```

Task 4: Implement the Navigation Option for Sales Managers

1. As the script to fill the dropdown box with all sales managers is provided already in the *onResultChanged* script of the R widget, you only need to care about what should happen when a sales manager is selected. Define the script that reads the selected sales manager and sends it as filter value to the data source used in the data frame **`DataSource`** of the R visualization widget.

Task 5: Save and Test your Application Design

1. Save your application and run it.

2. Check if the R visualization looks like the one in the exercise description and test if the selection of a sales manager refreshes and filters the chart.

Task 6: Implement the Option to Choose Different R Visualization Styles

1. In this step of the exercise, you will enable your end user to switch between the basic visualization to a more advanced one using the two already existing buttons, *basic* and *advanced*. To enable this, define the following script logic to the *BUTTON_Basic* and *BUTTON_Advanced* buttons.

   - The *BUTTON_Basic* should send the input parameter **`input`** with the value **`basic`** to the R visualization.
   - The *BUTTON_Advanced* should send the input parameter **`input`** with the value **`advanced`** to the R visualization.

2. Enhance the R script you use so that it checks if input parameter **`input`** exists and either store its value or a default one to a variable **`myStyle`** within R script.

3. Define a *if/else* logic within the R script to either visualize any advanced chart or the basic one we used already.

Note:
Your data science colleague who is familiar with R script provided the following lines of code to generate an advanced scatter chart:

```
    ggplot(DataSet, aes(x = `Quantity sold`, y =
`Gross Margin`, color = `Sales Manager`)) +
   geom_point(size = 3, shape = 0) +
   geom_smooth() +
   labs(title = "") +
   labs(x = "Quantity") +
   labs(y = "Margin") +
   theme(legend.position="none")
```

In addition to that new chart type, he recommended to import and apply the library *plotly* to enhance the chart with a rich set of new functions.

Task 7: Save and Test your Application Design

1. Save your application and run it.
2. Select a sales manager and choose the *advanced* button. Explore the features of the additional toolbar as soon as you hover over the chart.

OPTIONAL: Create a Dynamic R Visualization

Business Example

Some users approach you and ask for very special visualizations and you realize that you cannot achieve it with the standard chart widget. So you ask for support from a data science colleague. He explains to you that the needed visualization is easily created using the embedded R-server within SAP Analytics Cloud. He provides you with the few simple code lines that are needed to get the visualizations done.

In this exercise, you will implement the provided R-script code into a predefined application. You will enhance the application with a filtering functionality and use an input parameter to switch between two different visualizations within the R-script.

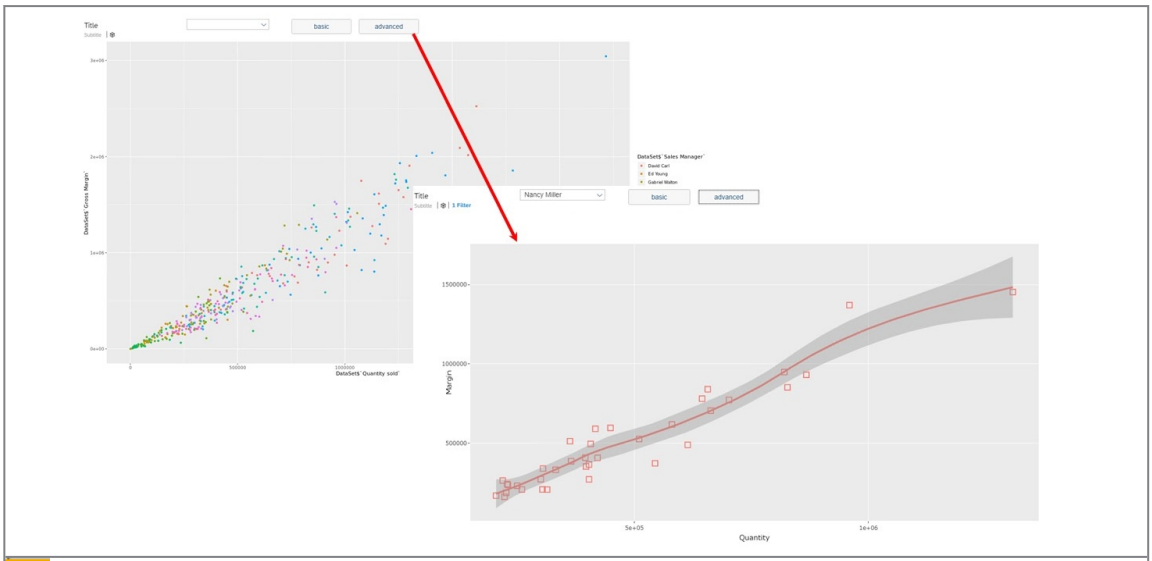


Figure 18: Result

Task 1: Create the Application

1. Create a copy of the application *GR000 Start R visualization* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Content* and save it with the name **`GR### R visualization`**.

   a) From the Navigation Bar choose *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Content*.

   b) Select the checkbox for the application *GR000 Start R visualization* and choose *Copy* .

   c) Choose *My Files* to choose your private folder.

   d) Type in the name **`GR### R visualization`** and choose *OK*.

e) Navigate to the *My Files* area.

f) Choose *GR### R visualization* to open the file in edit mode.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze and understand the given application structure. You will find an existing empty R visualization, a dropdown box that will be filled whenever the result set of the R visualization changes, and two buttons.
   a) Check the *Outline* area and the *onResultChanged* script of the *RVIZ_Scatter* widget.

Task 3: Create the Basic R Visualization

1. Define **`SACAD1_Model002`** as input data for the R visualization and use **`Date`** and **`Sales Manager`** in the *Rows*.
   a) Within the *Builder* panel of the *RVIZ_Scatter* widget, choose the *Add Input Data* button.
   b) Select the model **`SACAD1_Model002`** within the folder *Public* → *SACAD1_30* → *SACAD1 Content*.
   c) Choose the *Add Dimensions* button to define the data table structure.
   d) Select **`Date`** and **`Sales Manager`** and choose *OK*.
2. Rename the input data object you created in the previous step to **`DataSet`**.
   a) Within the *Builder* panel, click on the entry *SACAD1_Model002* and rename it to **`DataSet`**.
3. Define the R script to create a simple scatter chart displaying the measures **`Quantity sold`** on the x-axis and **`Gross Margin`** on the y-axis. The data points for each **`Sales Manager`** should be visualized in a different color. Use the **`ggplot2`** library to generate the chart.

Note:
Your data science colleague who is familiar with R script provided the following lines of code to generate a basic scatter chart:

```
library(ggplot2)
library(plotly, warn.conflicts = FALSE)
ggplot(DataSet,  aes(x = `Quantity sold`, y = `Gross
Margin`, color = `Sales Manager`)) +
      geom_point()
```

   a) Within the *Builder* panel, choose the *Add Script* button.
   b) Use the  *Expand* button to see the full screen.
   c) Type in the following code into the *Editor* section:

Note:
Related to **`Quantity Sold`**, **`Gross Margin`** and **`Sales Manager`** you need to type ` (backquote) instead of ‘

```
library(ggplot2)
library(plotly, warn.conflicts = FALSE)
ggplot(DataSet,  aes(x = `Quantity sold`, y = `Gross Margin`, color =
`Sales Manager`)) +
      geom_point()
```

In case you wonder why you define `warn.conflicts = FALSE` in the code, refer to this link to understand this:

https://stackoverflow.com/questions/39137110/

d) Choose the *Execute* button in the top right corner of the *Editor* section.

e) If a chart is visualized in the *Preview* section, choose *Apply* in the bottom right corner.

Task 4: Implement the Navigation Option for Sales Managers

1. As the script to fill the dropdown box with all sales managers is provided already in the *onResultChanged* script of the R widget, you only need to care about what should happen when a sales manager is selected. Define the script that reads the selected sales manager and sends it as filter value to the data source used in the data frame **`DataSource`** of the R visualization widget.

   a) Select *DD_SalesManager* within the *Outline* area and choose the *fx* *Edit Scripts* button .

   b) Add the following line of code:

```
RVIZ_Scatter.getDataFrame("DataSet").getDataSource().setDimensionFilte
r("Sales_Manager__5w3m5d06b5", DD_SalesManager.getSelectedKey());
```

Task 5: Save and Test your Application Design

1. Save your application and run it.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.
   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.
2. Check if the R visualization looks like the one in the exercise description and test if the selection of a sales manager refreshes and filters the chart.
   a) Compare the displayed R visualization with the one in the exercise description.
   b) Select one of the sales managers in the dropdown box and check if the chart changes.

Task 6: Implement the Option to Choose Different R Visualization Styles

1. In this step of the exercise, you will enable your end user to switch between the basic visualization to a more advanced one using the two already existing buttons, *basic* and *advanced*. To enable this, define the following script logic to the *BUTTON_Basic* and *BUTTON_Advanced* buttons.

- The *BUTTON_Basic* should send the input parameter **`input`** with the value **`basic`** to the R visualization.
- The *BUTTON_Advanced* should send the input parameter **`input`** with the value **`advanced`** to the R visualization.

a) Select *BUTTON_Basic* within the *Outline* area and choose the *Edit Scripts* button and select the *onClick* option to start the script editor.

b) Add the following line of code:

```
RVIZ_Scatter.getInputParameters().setString("input", "basic");
```

c) Select *BUTTON_Advanced* within the *Outline* area and choose the *Edit Scripts* button and select the *onClick* option to start the script editor.

d) Add the following line of code:

```
RVIZ_Scatter.getInputParameters().setString("input", "advanced");
```

2. Enhance the R script you use so that it checks if input parameter **`input`** exists and either store its value or a default one to a variable **`myStyle`** within R script.

a) Select the *RVIZ_Scatter* widget in the outline.

b) Within the *Builder* panel, choose the *Edit Script* button.

c) Use the *Expand* button to see the full screen.

d) Append the following lines of code to the existing ones.

```
if(exists("input")){
  myStyle <- input
} else {
  myStyle <- "basic"
}
```

3. Define a *if/else* logic within the R script to either visualize any advanced chart or the basic one we used already.

Note:
Your data science colleague who is familiar with R script provided the following lines of code to generate an advanced scatter chart:

```
    ggplot(DataSet, aes(x = `Quantity sold`, y =
`Gross Margin`, color = `Sales Manager`)) +
   geom_point(size = 3, shape = 0) +
   geom_smooth() +
   labs(title = "") +
   labs(x = "Quantity") +
   labs(y = "Margin") +
   theme(legend.position="none")
```

In addition to that new chart type, he recommended to import and apply the library *plotly* to enhance the chart with a rich set of new functions.

a) Select the *RVIZ_Scatter* widget in the outline.

b) Within the *Builder* panel, choose the *Edit Script* button.

c) Use the  *Expand* button to see the full screen.

d) Rearrange and enrich your existing code to match the following code:

Note:
Related to **`Quantity Sold`**, **`Gross Margin`** and **`Sales Manager`** you need to type ` (backquote) instead of ‘

```
library(ggplot2)
library(plotly, warn.conflicts = FALSE)

if(exists("input")){
  myStyle <- input
} else {
  myStyle <- "basic"
 }

if(myStyle == "advanced"){
     ggplot(DataSet, aes(x = `Quantity sold`, y = `Gross Margin`,
color = `Sales Manager`)) +
        geom_point(size = 3, shape = 0) +
        geom_smooth() +
        labs(title = "") +
        labs(x = "Quantity") +
        labs(y = "Margin") +
        theme(legend.position="none")
    ggplotly()
}else {
    ggplot(DataSet,  aes(x = `Quantity sold`, y = `Gross Margin`,
color = `Sales Manager`)) +
          geom_point()
}
```

e) Choose *Execute* and *Apply*.

Task 7: Save and Test your Application Design

1. Save your application and run it.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and choose *Save*.
   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.
2. Select a sales manager and choose the *advanced* button. Explore the features of the additional toolbar as soon as you hover over the chart.
   a) Select one of the sales managers in the dropdown box and check if the chart changes.
   b) Choose the *advanced* button and check if the chart changes.

Unit 7
Exercise 18

OPTIONAL: Use Smart Discovery in an Application

Business Example

As an analytic application developer, you can enable Smart Discovery in your application and discover additional information between columns within a data set.

In this exercise, you will learn how to launch Smart Discovery from an application and use it to analyze a measure while also setting the dimensions and measures to include in the analysis. On the results of Smart Discovery you will also see some possibilities of Smart Insight.

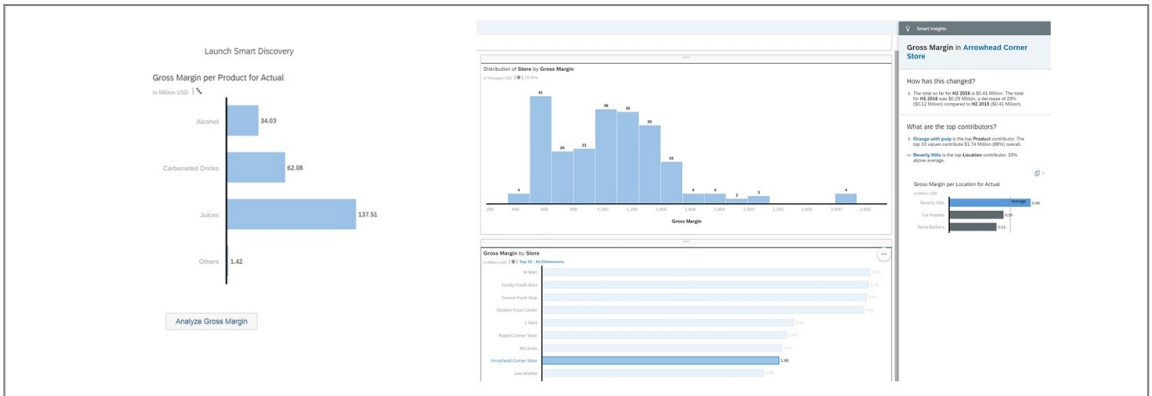


Figure 19: Result

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Use Smart Discovery in an application* from the *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Use Smart Discovery in an application`** into the folder structure *My Files*.

2. Open *GR### Use Smart Discovery in an application* for editing.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze the data source for the *CH_Product* chart and make note of the available dimensions and measures.

2. Notice the existing button in the application. You will add script to this button to demonstrate the Smart Discovery feature.

Task 3: Launch Smart Discovery

1. Define code for the *BTN_Analyze_Measure* button to launch Smart Discovery. Analyze **`Gross Margin`** measure on the level of **`Store`** and include **`Location`**, **`Store`**, **`Sales Manager`**, and **`Product`** dimensions to run your analysis.

Task 4: Save and Test your Application Design

1. Save your application.
2. Launch Smart Discovery and analyze the Gross Margin measure. Use Smart Insights to analyze the top contributor Product and Location for the Gross Margin in Arrowhead Corner Store. Also analyze which other locations contribute to the Gross Margin in Arrowhead Corner Store.

OPTIONAL: Use Smart Discovery in an Application

Business Example

As an analytic application developer, you can enable Smart Discovery in your application and discover additional information between columns within a data set.

In this exercise, you will learn how to launch Smart Discovery from an application and use it to analyze a measure while also setting the dimensions and measures to include in the analysis. On the results of Smart Discovery you will also see some possibilities of Smart Insight.

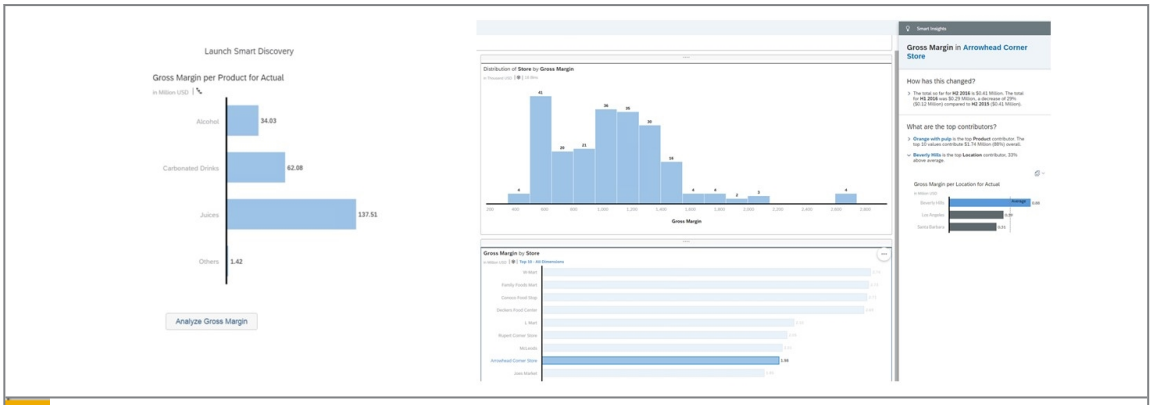


Figure 19: Result

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Use Smart Discovery in an application* from the *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Use Smart Discovery in an application`** into the folder structure *My Files*.

   a) From the Navigation Bar choose *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Content*.

   b) Select *GR000 Start Use Smart Discovery in an application* and choose *Copy* .

   c) Choose *My Files* and type in **`GR### Use Smart Discovery in an application`** as the name.

   d) Choose *OK*.

2. Open *GR### Use Smart Discovery in an application* for editing.

   a) Navigate to the *My Files* area.

   b) Choose *GR### Use Smart Discovery in an application* to open the file in edit mode.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze the data source for the *CH_Product* chart and make note of the available dimensions and measures.
   a) Select the *CH_Product* chart.
   b) Within the *Builder*, choose + *Add Measure* and review the full list of available measures.
   c) Within the *Builder*, choose + *Add Dimension* and review the full list of available dimensions.
2. Notice the existing button in the application. You will add script to this button to demonstrate the Smart Discovery feature.
   a) Check the *BTN_Analyze_Measure* button.

Task 3: Launch Smart Discovery

1. Define code for the *BTN_Analyze_Measure* button to launch Smart Discovery. Analyze **`Gross Margin`** measure on the level of **`Store`** and include **`Location`**, **`Store`**, **`Sales Manager`**, and **`Product`** dimensions to run your analysis.
   a) Select *BTN_Analyze_Measure* within the *Outline* area and choose *Edit Scripts → onClick* to start the script editor.
   b) Enter the following lines of code in the script editor:

```
var ds = CH_Product.getDataSource();

var SDsetting = SmartDiscoveryMeasureSettings.create(ds,
"[Account_BestRunJ_sold].[parentId].&[Gross_Margin]");

SDsetting.setIncludedDimensions(["Location_4nm2e04531","Store_3z2g5g06m
4", "Product_3e315003an", "Sales_Manager__5w3m5d06b5"]);
SDsetting.setEntityDimensions(["Store_3z2g5g06m4"]);


SmartDiscovery.buildStory(SDsetting);
```

Task 4: Save and Test your Application Design

1. Save your application.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.
   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.
2. Launch Smart Discovery and analyze the Gross Margin measure. Use Smart Insights to analyze the top contributor Product and Location for the Gross Margin in Arrowhead Corner Store. Also analyze which other locations contribute to the Gross Margin in Arrowhead Corner Store.
   a) Choose the *Analyze Gross Margin* button.
   b) Choose the *Launch Smart Discovery for "Gross Margin"* link in the popup window.
   c) In the *Smart Discovery* panel, confirm that **`Gross Margin`** is selected under *Target*.
   d) Under *Entity*, confirm that **`Store`** is selected.

e) Under *Advanced Settings*, confirm that **`Location`**, **`Store`**, **`Sales Manager`**, and **`Product`** dimensions are listed under *Included Columns*.

f) Choose *Run* to run Smart Discovery.

g) On the *Overview* tab, in the horizontal column chart *Gross Margin by Store*, right-click the bar for *Arrowhead Corner Store* and choose *Smart Insights...*.
Orange with Pulp is the top Product contributor and Beverly Hills is the top Location contributor.

h) Click on *Beverly Hills* to go to a column chart about Gross Margin per Location for Actual.
Los Angeles and Santa Barbara are the other two Location contributors.

Unit 8
Exercise 19

OPTIONAL: Use OData Calls in an Application

Business Example

Imagine you are analyzing your data and finding valuable insights from your SAP Analytics Cloud, analytics designer application. What's next? You want to act on it – whether that is to put in an order, restock a product, or follow up on an unexpected value. With SAP Analytics Cloud, analytics designer, you can take an action without leaving your application, by triggering an OData call into the transactional system and see the outcome of your action reflected in the same dashboard.

In this application, you will use an OData v4 service created in SAP S/4HANA system to read and update data in the SAP S/4HANA system. The application contains a table to show the latest discounted price of certain materials and a slider to determine the discount percentage to apply to the base product prices.

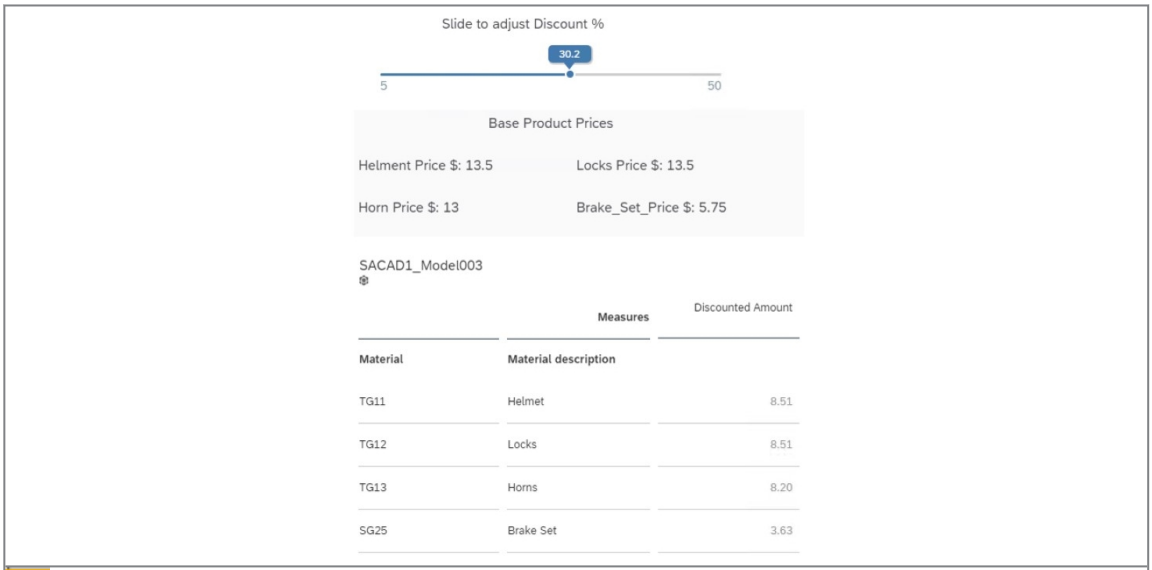


Figure 20: Result

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Use OData calls in an application* from the *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Use OData calls in an application`** into the folder structure *My Files*.
2. Open *GR### Use OData calls in an application* for editing.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze the textboxes within the *PANEL_Base_Prices* panel. The textboxes in the panel will display the base product prices for materials as shown in exercise description.
2. Analyze the existing script variables in the application.

Task 3: Add a Table to Display SAP S/4HANA Data

1. Add a table using *SACAD1_Model003* as the data source. Present the measure *Discounted Amount* in the columns. Add *Material* and *Material Description* to the rows. Position the table below the existing panel with the textboxes and resize so that the table has the same width as the panel above.

   Use **`TBL_Discounted_Amount`** as the name for the table.

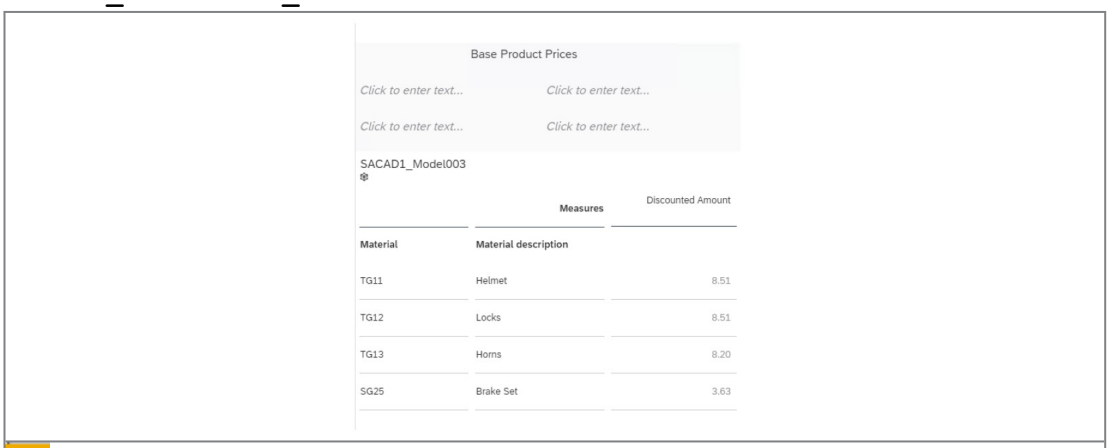


Figure 21: Example Table

2. In the *Styling* panel, set the *Canvas Content Layout* setting for the canvas to **`Center Aligned`**.

   The *Center Aligned* setting for the Canvas ensures that all widgets will be aligned to the middle of the page when the application is run.

Task 4: Read Data from an OData Service

1. Add an OData service called **`OData_Update_Price`** to your application. Select **`SACAD1S4H`** as the *Connection* and specify **`/sap/opu/odata4/iwbep/v4_sample/default/sap/zanalytics_material/0001/`** for the *End-Point URL*.

2. Add a new *Script Object* called **`SO_Price_Calc`** with a script function called **`getPrice`** and *Return Type* **`void`**.

3. Define the code for *getPrice* function to perform the following actions:

   - The material number and prices should be read from SAP S/4HANA using the OData service and the result stored in local array.
   - Loop through the local array and set values for existing script variables in the application to the **`StandardPrice`** of select material numbers. Use the information as shown in the following table to set the script variables values for the matching material numbers:

Script Variable Name	Material Number
SV_Helmet_Price	TG11
SV_Lock_Price	TG12
SV_Horn_Price	TG13

Script Variable Name	Material Number
SV_Brake_Set_Price	SG25

4. Define the code to perform the following actions on application launch:
   - Call the *getPrice()* function.
   - Use the existing textboxes in the application to display standard material prices.

Task 5: Update SAP S/4HANA Data Using the OData Service

1. Add a *Slider* widget with the name **`SL_Discount_Percent`** and position it above the panel with the textboxes. Set its range between **`5`** and **`50`** and the *Current Value* to **`5`** and choose the options to display both the range and current value labels.

Field	Value
Analytics Designer Properties	
Name	**`SL_Discount_Percent`**
Value of Slider	
Min Value	**`5`**
Max Value	**`50`**
Current Value	**`5`**
Slider Properties	
Display Min & Max Value Labels	☑
Display Current Value Label	☑

2. Create five new *Script Variables* with the names **`SV_Discount_Percentage`**, **`SV_Discounted_Lock_Price`**, **`SV_Discounted_Helmet_Price`**, **`SV_Discounted_Horn_Price`**, **`SV_Discounted_Brake_Set_Price`** all of *Type* **`number`**.
3. Add a new function called **`simulationPrice`** to the existing *SO_Price_Calc* script object and set the *Return Type* to **`void`**.
4. Define the code for the *simulationPrice* function to read the discount percentage from the slider and apply the discount to the base material prices.
5. Add another script function called **`submitPrice`** to the existing *SO_Price_Calc* script object and set the *Return Type* to **`void`**.
6. Define the code for *submitPrice* function to update the discounted material prices in the SAP S/4HANA system via the OData service.
7. Add code to the *SL_Discount_Percent* slider to perform the following actions:
   - Set *SV_Discount_Percentage* to the slider's current value.

- Call the *simulationPrice()* function.
- Call the *submitPrice()* function.
- Refresh the application.

8. Update the *Canvas* to perform the following actions on startup:
   - Set *SV_Discount_Percentage* to the slider's current value.
   - Call the *simulationPrice()* function.
   - Call the *submitPrice()* function.
   - Refresh the application.

Task 6: Save and Test your Application Design

1. Save your application.
2. View base material prices and update the discount applied to the base material prices.

Unit 8
Solution 19

OPTIONAL: Use OData Calls in an Application

Business Example

Imagine you are analyzing your data and finding valuable insights from your SAP Analytics Cloud, analytics designer application. What's next? You want to act on it – whether that is to put in an order, restock a product, or follow up on an unexpected value. With SAP Analytics Cloud, analytics designer, you can take an action without leaving your application, by triggering an OData call into the transactional system and see the outcome of your action reflected in the same dashboard.

In this application, you will use an OData v4 service created in SAP S/4HANA system to read and update data in the SAP S/4HANA system. The application contains a table to show the latest discounted price of certain materials and a slider to determine the discount percentage to apply to the base product prices.

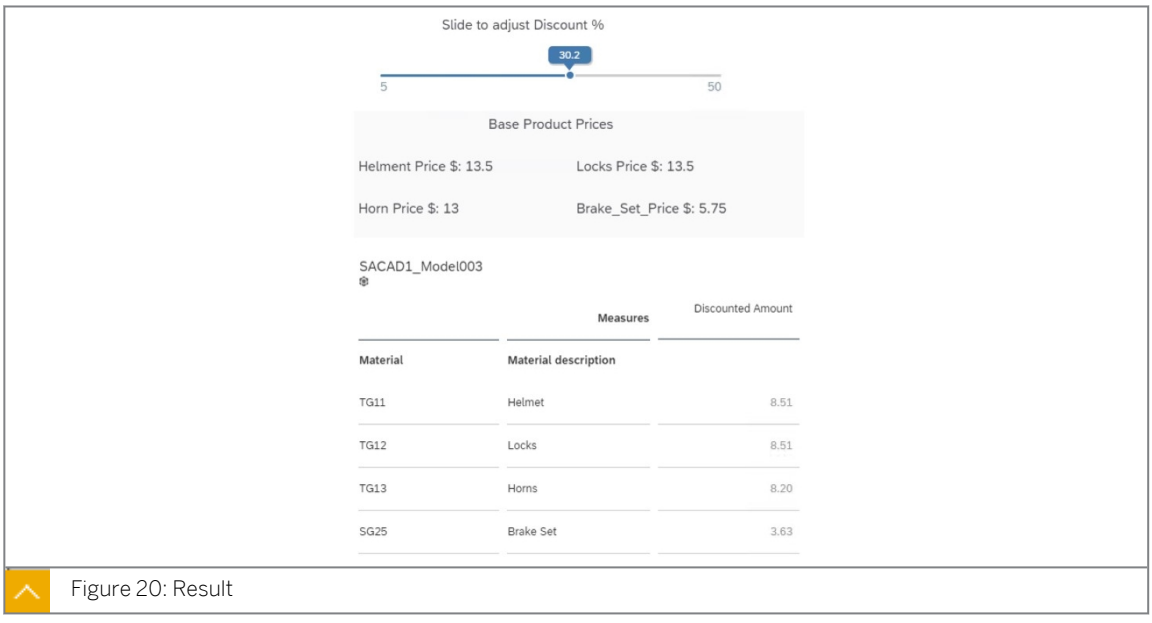


Figure 20: Result

Task 1: Copy the Starting Application to your Private Files Area and Open it for Editing

1. Copy *GR000 Start Use OData calls in an application* from the *Public* → *SACAD1_30* → *SACAD1 Content* folder and save it with the name **`GR### Use OData calls in an application`** into the folder structure *My Files*.

   a) From the Navigation Bar choose *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Content*.

   b) Select the checkbox for the application *GR000 Start Use OData calls in an application* and choose *Copy* .

c) Choose *My Files* and type in **`GR### Use OData calls in an application`** as the name.

d) Choose *OK*.

2. Open *GR### Use OData calls in an application* for editing.
   a) Navigate to the *My Files* area.
   b) Choose *GR### Use OData calls in an application* to open the file in edit mode.

Task 2: Familiarize Yourself with the Given Application Structure

1. Analyze the textboxes within the *PANEL_Base_Prices* panel. The textboxes in the panel will display the base product prices for materials as shown in exercise description.
   a) Check the different textboxes inside the *PANEL_Base_Prices*. We will write script to read material base prices and populate these textboxes.
2. Analyze the existing script variables in the application.
   a) Within the *Outline* panel, under *Scripting*, notice the four existing script variables.
   b) Choose each script variable in succession and notice they are of type number. You will use these variable to store base product prices of materials.

Task 3: Add a Table to Display SAP S/4HANA Data

1. Add a table using *SACAD1_Model003* as the data source. Present the measure *Discounted Amount* in the columns. Add *Material* and *Material Description* to the rows. Position the table below the existing panel with the textboxes and resize so that the table has the same width as the panel above.

   Use **`TBL_Discounted_Amount`** as the name for the table.

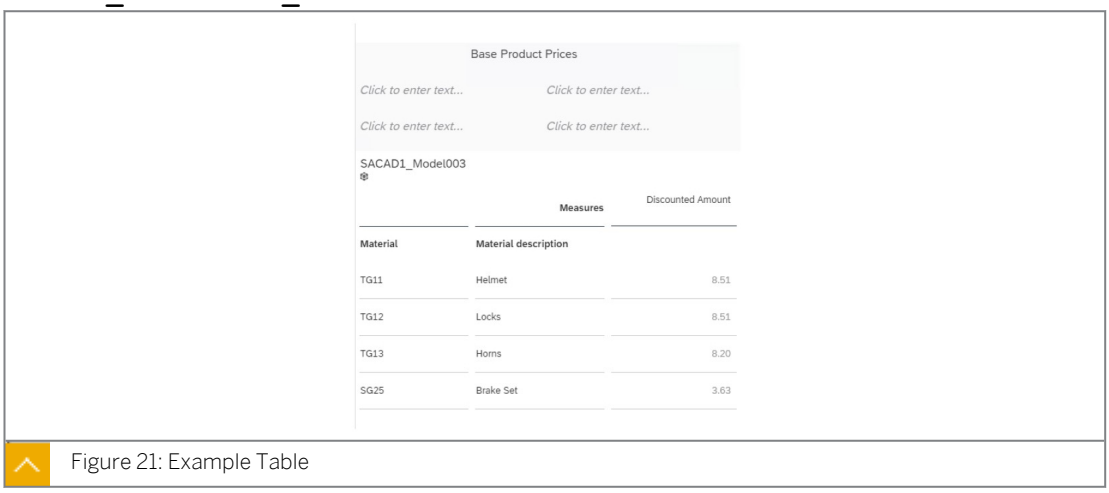


Figure 21: Example Table

   a) Within the *Insert*-section of the toolbar, choose *Table*.
   b) Choose the model *SACAD1_Model003* located under the folder *Public* → *SACAD1_30* → *SACAD1 Content*.
   c) Open the *Designer* panel.
   d) Within the *Builder* panel, choose the settings as shown in the following table:

Field	Value
Table Structure	

Field	Value
Rows	Click *+Add Measures/Dimensions* and select **`Material`** and **`Material Description`**,
Columns	In the *Measures*, make sure **`Discounted Amount`** is selected

e) Position the table below the existing panel that contains the four textboxes and resize the table to match the width of the panel.

f) Rename the table to **`TBL_Discounted_Amount`**.

2. In the *Styling* panel, set the *Canvas Content Layout* setting for the canvas to **`Center Aligned`**.

a) Open the *Designer* panel.

b) Within the *Styling* panel of the *Canvas* , set *Canvas Content Layout* setting to **`Center Aligned`**.

The *Center Aligned* setting for the Canvas ensures that all widgets will be aligned to the middle of the page when the application is run.

Task 4: Read Data from an OData Service

1. Add an OData service called **`OData_Update_Price`** to your application. Select **`SACAD1S4H`** as the *Connection* and specify **`/sap/opu/odata4/iwbep/v4_sample/default/sap/zanalytics_material/0001/`** for the *End-Point URL*.

a) Within the *Outline* panel, under *Scripting*, choose + next to *OData Services*.

b) Specify **`OData_Update_Price`** as the *Name*.

c) Select **`SACAD1S4H`** as the *Connection*.

d) Specify **`/sap/opu/odata4/iwbep/v4_sample/default/sap/zanalytics_material/0001/`** for the *End-Point URL*.

e) Choose *Refresh* to see the metadata of the OData service.

f) Choose *Done*.

2. Add a new *Script Object* called **`SO_Price_Calc`** with a script function called **`getPrice`** and *Return Type* **`void`**.

a) Within the *Outline* panel, under *Scripting*, choose + next to *Script Objects*. A new script object and a new script function is added.

b) Rename the script object **`SO_Price_Calc`**.

c) Rename the script function **`getPrice`** and confirm the *Return Type* is set to **`void`**.

3. Define the code for *getPrice* function to perform the following actions:

- The material number and prices should be read from SAP S/4HANA using the OData service and the result stored in local array.
- Loop through the local array and set values for existing script variables in the application to the **`StandardPrice`** of select material numbers. Use the information as shown in the following table to set the script variables values for the matching material numbers:

Script Variable Name	Material Number
SV_Helmet_Price	TG11
SV_Lock_Price	TG12
SV_Horn_Price	TG13
SV_Brake_Set_Price	SG25

a) Select *Edit Scripts* for the *getPrice* function to open the script editor.

b) Enter the following lines of code in the script editor:

```
var product_arr =
OData_Update_Price.getEntitiesFromEntitySet("MaterialPriceDataSet").val
ue;

for (var i=0;i<product_arr.length;i++)
{
    if (product_arr[i].Materialno === "TG11")
    {
        SV_Helmet_Price = product_arr[i].Standardprice;
        console.log(product_arr[i].Materialno);
        console.log(product_arr[i]);
    }
    else if (product_arr[i].Materialno === "TG12")
    {
        SV_Lock_Price = product_arr[i].Standardprice;
        console.log(product_arr[i].Materialno);
        console.log(product_arr[i]);
    }
    else if (product_arr[i].Materialno === "TG13")
    {
        SV_Horn_Price = product_arr[i].Standardprice;
        console.log(product_arr[i].Materialno);
        console.log(product_arr[i]);
    }
    else if (product_arr[i].Materialno === "SG25")
    {
        SV_Brake_Set_Price = product_arr[i].Standardprice;
        console.log(product_arr[i].Materialno);
        console.log(product_arr[i]);
    }
}
```

4. Define the code to perform the following actions on application launch:
   - Call the *getPrice()* function.
   - Use the existing textboxes in the application to display standard material prices.

   a) Select *Canvas* within the *Outline* area and choose the *Edit Scripts* → *onInitialization* button to start the script editor.

b) Enter the following lines of code in the script editor:

```
SO_Price_Calc.getPrice();

TXT_Horn_Price.applyText("Horn Price $: " + SV_Horn_Price.toString());
TXT_Lock_Price.applyText("Locks Price $: "  +
SV_Lock_Price.toString());
TXT_Brake_Set_Price.applyText("Brake_Set_Price $: " +
SV_Brake_Set_Price.toString());
TXT_Helmet_Price.applyText("Helment Price $: " +
SV_Helmet_Price.toString());
```

Task 5: Update SAP S/4HANA Data Using the OData Service

1. Add a *Slider* widget with the name **`SL_Discount_Percent`** and position it above the panel with the textboxes. Set its range between **`5`** and **`50`** and the *Current Value* to **`5`** and choose the options to display both the range and current value labels.

   a) Within the *Insert*-section of the toolbar, add a *Slider* to the canvas.

   b) Position the *Slider* above the text boxes.

   c) Open the *Designer* panel.

   d) Within the *Styling* panel, set the name as shown in the following table and use the *Builder* panel to choose the other settings as shown in the following table:

Field	Value
Analytics Designer Properties	
Name	**`SL_Discount_Percent`**
Value of Slider	
Min Value	**`5`**
Max Value	**`50`**
Current Value	**`5`**
Slider Properties	
Display Min & Max Value Labels	☑
Display Current Value Label	☑

2. Create five new *Script Variables* with the names **`SV_Discount_Percentage`**, **`SV_Discounted_Lock_Price`**, **`SV_Discounted_Helmet_Price`**, **`SV_Discounted_Horn_Price`**, **`SV_Discounted_Brake_Set_Price`** all of *Type* **`number`**.

   a) Within the *Outline* panel, under *Scripting*, choose + next to *Script Variables*.

   b) Name the variable **`SV_Discount_Percentage`** and set the *Type* to **`number`**.

   c) Create four additional *Script Variables* called **`SV_Discounted_Lock_Price`**, **`SV_Discounted_Helmet_Price`**, **`SV_Discounted_Horn_Price`**, **`SV_Discounted_Brake_Set_Price`** and set the *Type* to **`number`**.

3. Add a new function called **`simulationPrice`** to the existing *SO_Price_Calc* script object and set the *Return Type* to **`void`**.

   a) Within the *Outline* panel, under *Scripting*, choose the ••• *More* icon next to *SO_Price_Calc* and select *Add Script Function*.

   b) Rename the script function **`simulationPrice`** and confirm the *Return Type* is set to **`void`**.

4. Define the code for the *simulationPrice* function to read the discount percentage from the slider and apply the discount to the base material prices.

   a) Select *Edit Scripts* for the *simulationPrice* function to open the script editor.

   b) Enter the following lines of code in the script editor:

```
SV_Discounted_Lock_Price = SV_Lock_Price*(1-SV_Discount_Percentage/
100);
SV_Discounted_Brake_Set_Price = SV_Brake_Set_Price*(1-
SV_Discount_Percentage/100);
SV_Discounted_Horn_Price = SV_Horn_Price*(1-SV_Discount_Percentage/
100);
SV_Discounted_Helmet_Price = SV_Helmet_Price*(1-SV_Discount_Percentage/
100);
```

5. Add another script function called **`submitPrice`** to the existing *SO_Price_Calc* script object and set the *Return Type* to **`void`**.

   a) Within the *Outline* panel, under *Scripting*, choose the ••• *More* icon next to *SO_Price_Calc* and select *Add Script Function*.

   b) Rename the script function **`submitPrice`** and confirm the *Return Type* is set to **`void`**.

6. Define the code for *submitPrice* function to update the discounted material prices in the SAP S/4HANA system via the OData service.

   a) Select *Edit Scripts* for the *submitPrice* function to open the script editor.

   b) Enter the following lines of code in the script editor:

```
OData_Update_Price.executeAction("MaterialPriceUpdate",
{price:SV_Discounted_Helmet_Price.toString(), Material:'TG11'});
OData_Update_Price.executeAction("MaterialPriceUpdate",
{price:SV_Discounted_Lock_Price.toString(), Material:'TG12'});
OData_Update_Price.executeAction("MaterialPriceUpdate",
{price:SV_Discounted_Horn_Price.toString(), Material:'TG13'});
OData_Update_Price.executeAction("MaterialPriceUpdate",
{price:SV_Discounted_Brake_Set_Price.toString(), Material:'SG25'});
```

7. Add code to the *SL_Discount_Percent* slider to perform the following actions:

   - Set *SV_Discount_Percentage* to the slider's current value.
   - Call the *simulationPrice()* function.
   - Call the *submitPrice()* function.
   - Refresh the application.

   a) Choose the *SL_Discount_Percent* slider within the canvas area. In the upcoming menu, choose *Edit Scripts* to open the script editor.

b) Enter the following lines of code in the script editor:

```
SV_Discount_Percentage = SL_Discount_Percent.getValue();

SO_Price_Calc.simulationPrice();
SO_Price_Calc.submitPrice();

Application.refreshData();
```

8. Update the *Canvas* to perform the following actions on startup:

   - Set *SV_Discount_Percentage* to the slider's current value.
   - Call the *simulationPrice()* function.
   - Call the *submitPrice()* function.
   - Refresh the application.

   a) Select *Canvas* within the *Outline* area and choose the *Edit Scripts → onInitialization* button to start the script editor.

   b) Append the following lines of code to the existing script in the script editor:

```
SV_Discount_Percentage = SL_Discount_Percent.getValue();

SO_Price_Calc.simulationPrice();
SO_Price_Calc.submitPrice();
Application.refreshData();
```

Task 6: Save and Test your Application Design

1. Save your application.

   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.

   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.

2. View base material prices and update the discount applied to the base material prices.

   a) Check the textboxes in *Base Product Prices* to see the material base prices are read from SAP S/4HANA via OData service.

   b) Drag the slider to set different discount percentage values and notice how the discount is applied to the base material prices as reflected in the table.

Unit 8
Exercise 20

OPTIONAL: Embed a Web Page into an Analytic Application

Business Example

Using the Web Page widget, you can embed a Web page into your analytic application to enrich your analytics with additional live hosted content. It is also possible to pass messages between the application and the Web page widget using post message API, but we will cover that concept in the next exercise.

In this application, you will enhance the application created in the previous exercise and add a Web search engine to perform Web searches based on selected materials. The goal is to allow the user to compare material prices with prices of similar material available in the market. We will use the `setAddress` API of the Web Page widget to dynamically set the address that the Web page will navigate to.

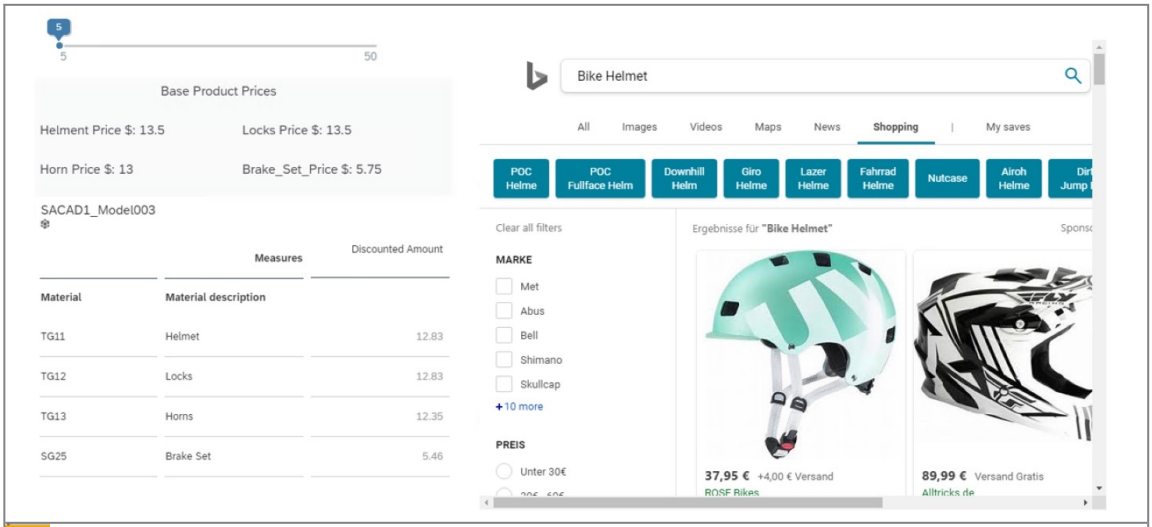


Figure 22: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Use OData calls in an application* and save it with the new name **`GR### Embed a web page into an analytical application`**.

2. Alternatively you can create a copy of the application *GR000 Use OData calls in an application* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Solutions* and save it with the name **`GR### Embed a web page into an analytical application`** into the folder structure *My Files*.

Task 2: Embed a Web Page Widget and Perform Web Searches

1. Add a *Web Page* widget with the name **`WP_Search`** and position it to the right of the existing widgets. Set the Web page address of the widget to **`https://www.bing.com`**.
2. Create a new *Script Variable* with the name **`SV_Base_URL`** of *Type* **`string`**. Set the default value of *SV_Base_URL* to **`https://www.bing.com`**.
3. Define the code that is needed so that the Web page shows the search results based on the material selected by the user in the *TBL_Discounted_Amount* table. Add **`Bike`** as a prefix to your search query and focus the web search on **`Shopping`**. In case multiple cells are selected, we only take the last one to search for.

Task 3: Save and Test your Application Design

1. Save your application and run it.
2. Click within any of the cells in the table to see the corresponding Web search results in Web Page widget.

OPTIONAL: Embed a Web Page into an Analytic Application

Business Example

Using the Web Page widget, you can embed a Web page into your analytic application to enrich your analytics with additional live hosted content. It is also possible to pass messages between the application and the Web page widget using post message API, but we will cover that concept in the next exercise.

In this application, you will enhance the application created in the previous exercise and add a Web search engine to perform Web searches based on selected materials. The goal is to allow the user to compare material prices with prices of similar material available in the market. We will use the `setAddress` API of the Web Page widget to dynamically set the address that the Web page will navigate to.

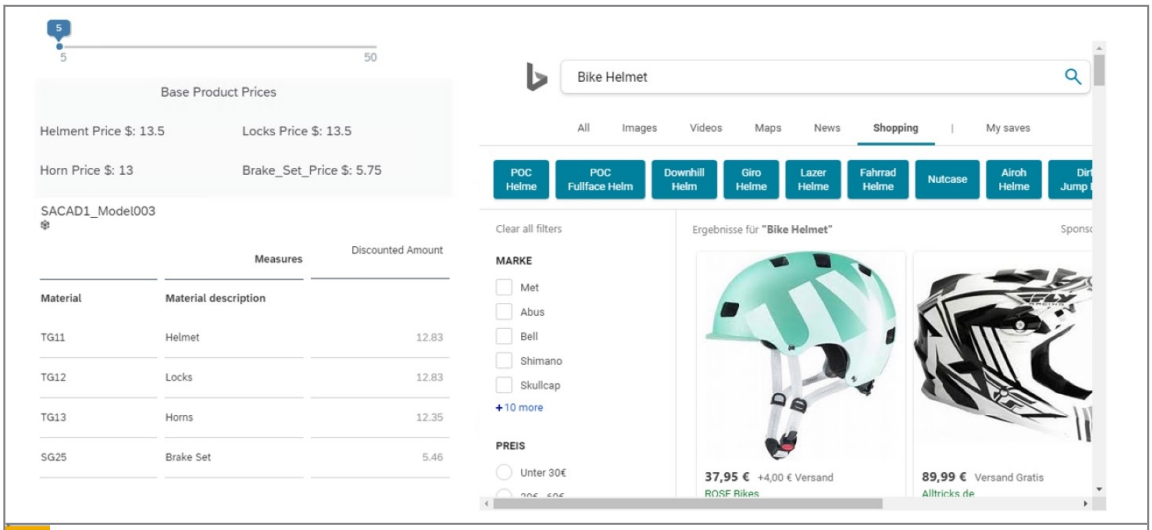


Figure 22: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Use OData calls in an application* and save it with the new name **`GR### Embed a web page into an analytical application`**.

   a) In the application designer, switch to your already opened application *GR### Use OData calls in an application*.

   b) In the toolbar, choose the *Save* icon and choose *Save as...*

   c) Type in **`GR### Embed a web page into an analytical application`** as the name and choose *OK*.

2. Alternatively you can create a copy of the application *GR000 Use OData calls in an application* which is available in the folder *Public* → *SACAD1_30* → *SACAD1 Solutions* and save it with the name **`GR### Embed a web page into an analytical application`** into the folder structure *My Files*.

   a) From the Navigation Bar choose  *Files* → *My Files* → *Public* → *SACAD1_30* → *SACAD1 Solutions*.

   b) Select the checkbox for the application *GR000 Use OData calls in an application* and choose  *Copy* .

   c) Choose *My Files* and type in **`GR### Embed a web page into an analytical application`**.

   d) Choose *OK*.

   e) Navigate to the *My Files* area.

   f) Choose *GR### Embed a web page into an analytical application* to open the file in edit mode.

Task 2: Embed a Web Page Widget and Perform Web Searches

1. Add a *Web Page* widget with the name **`WP_Search`** and position it to the right of the existing widgets. Set the Web page address of the widget to **`https://www.bing.com`**.

   a) Within the *Outline*, select the *Canvas*.

   b) Within the *Insert* section of the toolbar, choose *Add* → *More Widgets* → *Web Page*.

   c) Within the *Builder* section of the *Designer*, set the *Web Page Address* to **`https://www.bing.com`**.

   d) Within the *Styling* section of the *Designer*, set the *Name* to **`WP_Search`**.

   e) Position the *Web Page* widget to the right of the existing widgets.

2. Create a new *Script Variable* with the name **`SV_Base_URL`** of *Type* **`string`**. Set the default value of *SV_Base_URL* to **`https://www.bing.com`**.

   a) Within the *Outline* panel, under *Scripting*, choose  next to *Script Variables*.

   b) Rename the variable **`SV_Base_URL`** and set the *Type* to **`string`**.

   c) Set the *Default Value* to **`https://www.bing.com`** and press *Done* .

3. Define the code that is needed so that the Web page shows the search results based on the material selected by the user in the *TBL_Discounted_Amount* table. Add **`Bike`** as a prefix to your search query and focus the web search on **`Shopping`**. In case multiple cells are selected, we only take the last one to search for.

   a) Select *TBL_Discounted_Amount* within the *Outline* area, and choose the *Edit Scripts* → *onSelect* button to start the script editor.

b) Enter the following lines of code in the script editor:

```
var material_sel = TBL_Discounted_Amount.getSelections();
console.log( ['Table Selection: ', material_sel]);
if (material_sel.length > 0) {
    var material_members = ArrayUtils.create(Type.string);
    for (var i=0; i < material_sel.length; i++ ) {
        for(var dimension in material_sel[i]){
            material_members[i] =  material_sel[i][dimension];
            var URL = SV_Base_URL +"/shop?q=" + "Bike " +
material_members[i];
            WP_Search.setAddress(URL);
        }
    }
}
```

Task 3: Save and Test your Application Design

1. Save your application and run it.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.
   b) In the top right corner, choose the *Run Analytic Application* button. A new tab within your browser opens and shows you your application.
2. Click within any of the cells in the table to see the corresponding Web search results in Web Page widget.
   a) Click within any of the table columns in the **`Helmet`** row.
   b) Notice how the *Web Page* widget shows search results for bike helmets.
   c) Click on another table row and see the corresponding Web search results in the *Web Page* widget.

Unit 8
Exercise 21

OPTIONAL: Embed an Analytic Application into a Host Page

Business Example

You have your own corporate Web page and would like to embed your analytic application directly in this Web page and also share information back and forth between the Web page and the analytic application.

In this application, you will enhance the application created in the previous exercise and embed that application into a pre-created HTML-based application hosted on a Tomcat application server. You will use post message APIs to pass messages between the host application and your analytic application. You will launch the HTML-based application hosted on Tomcat, and use product images in the host application to filter the results in the analytic application. You will also pass messages back from analytic application to the host application and display material prices on top of the images.

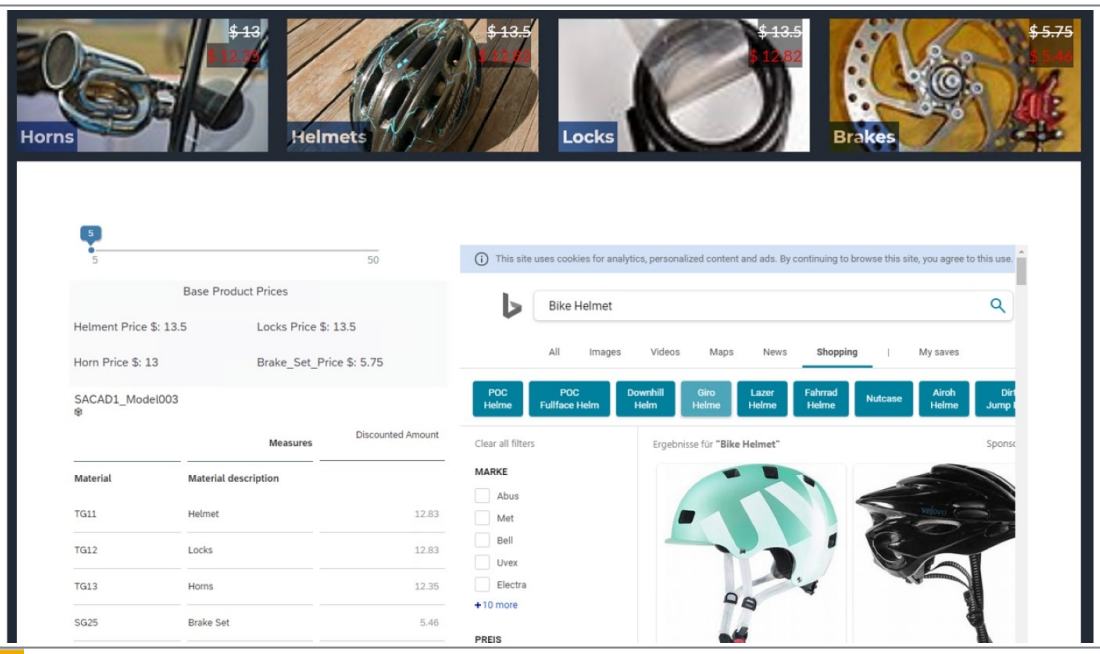


Figure 23: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Embed a web page into an analytical application* and save it with the new name **`GR### Embed an analytical application into a host page`**.

2. Alternatively you can create a copy of the application *GR000 Embed a web page into an analytical application* which is available in the folder *Public* → *SACAD1_30* → *SACAD1*

Solutions and save it with the name **`GR### Embed an analytical application into a host page`** into the folder structure *My Files*.

Task 2: Update Host HTML Application

1. Copy the URL of the *GR### Embed an analytical application into a host page* application.
2. Update the URL in the **`<iframe>`** tag in the *indexSACAD01.html* file to the URL copied from earlier step. Append **`?mode=embed`** to the end of the URL.
3. Within the *indexSACAD01.html* file, update the SAP Analytics Cloud tenant URL in the **`addClickHandler()`** function to match the tenant URL used for this training.
4. Save changes to the *indexSACAD01.html* file.

Task 3: Pass Messages to the Host Application

1. Update the *SL_Discount_Percent* slider and pass the discount percent value as a string to the host application.
2. Update the *Canvas* script to pass **`discount percent`** and base prices for **`horns, locks, helmets and brake sets`** as a string to the host application.

Task 4: Receive Messages from the Host Application

1. Update the *Canvas* script so that the *TBL_Discounted_Amount* table is filtered based on the image selected by the user in the host application.

Task 5: Save and Test your Application Design

1. Save your application.
2. Navigate to the **`N:\SAC\SACAD1`** folder in Windows Explorer and open the *User_Content* shortcut. Within the browser, navigate to *train-##* > → *SAC* → *SACAD1* and select the **`indexSACAD01.html`** file.
3. Test the application.

OPTIONAL: Embed an Analytic Application into a Host Page

Business Example

You have your own corporate Web page and would like to embed your analytic application directly in this Web page and also share information back and forth between the Web page and the analytic application.

In this application, you will enhance the application created in the previous exercise and embed that application into a pre-created HTML-based application hosted on a Tomcat application server. You will use post message APIs to pass messages between the host application and your analytic application. You will launch the HTML-based application hosted on Tomcat, and use product images in the host application to filter the results in the analytic application. You will also pass messages back from analytic application to the host application and display material prices on top of the images.

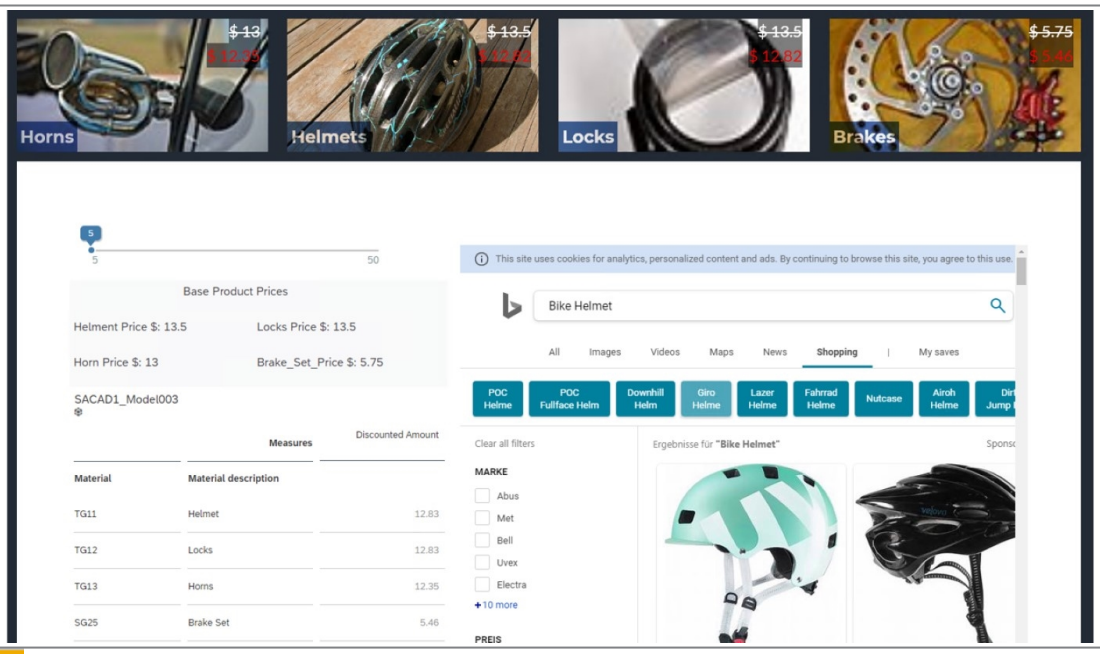


Figure 23: Result

Task 1: Create the Application

1. Open the application you created in the previous exercise with the name *GR### Embed a web page into an analytical application* and save it with the new name **`GR### Embed an analytical application into a host page`**.

   a) In the application designer, switch to your already opened application *GR### Embed a web page into an analytical application*.

b) In the toolbar, choose the *Save* icon and choose *Save as...*

c) Type in **GR### Embed an analytical application into a host page** as the name and choose *OK*.

2. Alternatively you can create a copy of the application *GR000 Embed a web page into an analytical application* which is available in the folder *Public → SACAD1_30 → SACAD1 Solutions* and save it with the name **GR### Embed an analytical application into a host page** into the folder structure *My Files*.

a) From the Navigation Bar choose *Files → My Files → Public → SACAD1_30 → SACAD1 Solutions*.

b) Select the checkbox for the application *GR000 Embed a web page into an analytical application* and choose *Copy* .

c) Choose *My Files* and type in **GR### Embed an analytical application into a host page**.

d) Choose *OK*.

e) Navigate to the *My Files* area.

f) Choose *GR### Embed an analytical application into a host page* to open the file in edit mode.

Task 2: Update Host HTML Application

1. Copy the URL of the *GR### Embed an analytical application into a host page* application.

a) Within the *File*-section of the toolbar, choose the *Share* icon and select *Share*.

b) In the *Share Application* panel, use the context menu to copy the application link and close the dialog.

2. Update the URL in the **<iframe>** tag in the *indexSACAD01.html* file to the URL copied from earlier step. Append **?mode=embed** to the end of the URL.

a) Using Windows Explorer, navigate to the **N:\SAC\SACAD1** directory.

b) Right-click on the **indexSACAD01.html** file and open it with *Notepad*.

c) Update the existing application URL in **<iframe>** tag with the URL copied earlier. Maintain the double quotes around the URL and add **?mode=embed** to the end of the URL. See the following figure for reference.

```
                </figcaption>
                <span></span>
                <span></span>
            </figure>
        </div>
        <iframe style="width: 1440px; height: 1120px;"
            src="https://sapeducf-dev.eu10.sapanalytics.cloud/sap/fpa/ui/tenants/77426/bo/application/88D62AE8E679B6F877B46A2580D011D5?mode=embed"></iframe>
    /div>

    script>
        document.addEventListener("readystatechange", (ev) => {
            if (document.readyState === "interactive") {
                addClickHandler();
            }
        });
```

Figure 24: Update URL

3. Within the *indexSACAD01.html* file, update the SAP Analytics Cloud tenant URL in the **addClickHandler()** function to match the tenant URL used for this training.

a) Within *indexSACAD01.html*, replace the existing SAP Analytics Cloud URL in the **`addClickHandler`** in four places with the SAP Analytics Cloud URL used for this training. See the following figure for reference:

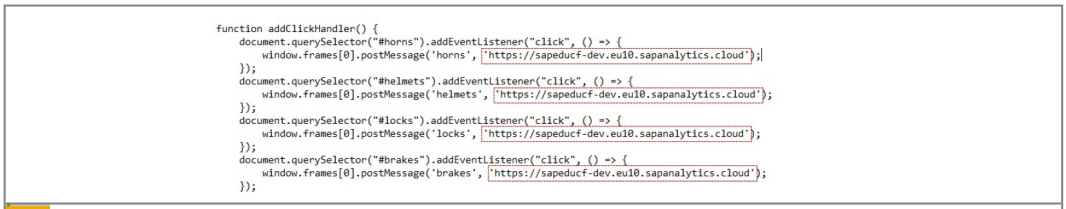


Figure 25: Replace SAP Analytics Cloud URL

4. Save changes to the *indexSACAD01.html* file.

a) From the Notepad menu bar, choose *File* → *Save* to save changes to the *indexSACAD01.html* file.

b) Close.

Task 3: Pass Messages to the Host Application

1. Update the *SL_Discount_Percent* slider and pass the discount percent value as a string to the host application.

a) Select the *SL_Discount_Percent* slider within the *Canvas* area. Choose *Edit Scripts* to open the script editor.

b) Append the following lines of code to the existing script in the script editor:

```
Application.postMessage(PostMessageReceiver.Parent, "{\"discount\":
"+ConvertUtils.numberToString(SV_Discount_Percentage) + "}", "*");
```

2. Update the *Canvas* script to pass **`discount percent`** and base prices for **`horns, locks, helmets and brake sets`** as a string to the host application.

a) Select *Canvas* within the *Outline* area and choose the *Edit Scripts* → *onInitialization* button to start the script editor.

b) Append the following lines of code to the existing script in the script editor:

```
Application.postMessage(PostMessageReceiver.Parent,
    "{\"discount\":
"+ConvertUtils.numberToString(SV_Discount_Percentage) +
        ", \"horns\": " + SV_Horn_Price.toString() +
    ", \"helmets\": " + SV_Helmet_Price.toString() +
    ", \"locks\": " + SV_Lock_Price.toString() +
    ", \"brakes\": " + SV_Brake_Set_Price.toString() +
    "}" , "*");
```

Task 4: Receive Messages from the Host Application

1. Update the *Canvas* script so that the *TBL_Discounted_Amount* table is filtered based on the image selected by the user in the host application.

a) Select *Canvas* within the *Outline* area and choose the *Edit Scripts* → *onPostMessageReceived* button to start the script editor.

b) Insert the following lines of code in the script editor:

```
if (message === "horns") {
TBL_Discounted_Amount.getDataSource().setDimensionFilter("2CZSACAD01C-
MATERIALCODE", "TG13");
}
```

```
if (message === "locks") {
TBL_Discounted_Amount.getDataSource().setDimensionFilter("2CZSACAD01C-
MATERIALCODE","TG12");
}

if (message === "helmets") {
TBL_Discounted_Amount.getDataSource().setDimensionFilter("2CZSACAD01C-
MATERIALCODE","TG11");
}

if (message === "brakes") {
TBL_Discounted_Amount.getDataSource().setDimensionFilter("2CZSACAD01C-
MATERIALCODE","SG25");
}
```

Task 5: Save and Test your Application Design

1. Save your application.
   a) Within the *File*-section of the toolbar, choose the *Save* icon and select *Save*.
2. Navigate to the **`N:\SAC\SACAD1`** folder in Windows Explorer and open the *User_Content* shortcut. Within the browser, navigate to *train-##* > → *SAC* → *SACAD1* and select the **`indexSACAD01.html`** file.
   a) Using Windows Explorer, navigate to the **`N:\SAC\SACAD1`** directory.
   b) Double-click the **`User_Content`** shortcut.

   Note:
   As a result, you will be directed to an address on our training landscape without a valid certificate. You can safely proceed here.

   c) In case you are prompted, choose *Advanced* and select the URL link to proceed.
   d) Navigate to *train-##* → *SAC* → *SACAD1*.
   e) Select the file **`indexSACAD01.html`** to open it in the browser.
3. Test the application.
   a) Drag the slider to left or right to adjust the discount percentage and notice how the discounted prices in the hosting application are updated.
   b) Click on the image of the **`Helmet`** and notice how the table is filtered to display just the row for helmets.
   c) Click on another image in the hosting application notice how the table is filtered accordingly.