HC300

SAP HANA Cloud Modeling

SYSTEM SETUP GUIDE
HANDS-ON PRACTICE

Collection: 2410

SAP Copyrights, Trademarks and Disclaimers

Contents

System Setup Guide

1. Introduction

Our practice systems are tailored to meet the specific needs of each course. As a result, each practice system is unique and contains specific technical details. Before you begin working on the exercises for your course, take the time to familiarize yourself with the information provided in this Setup Guide. Follow the instructions outlined in the exercise guide carefully and complete the exercises. The Setup Guide also provides useful information throughout the use of your practice system.

2. Latest Update of This Document (“Change History”)

Latest Update: 21.03.2025

3. System Landscape And Logon Information

System Landscape

The practice system for the learning journey *Developing Data Models with SAP HANA Cloud* is a shared practice environment.

This environment includes the following components:

- An SAP Business Technology Platform (BTP) subaccount, including –among others– an SAP HANA Cloud database instance, a Cloud Foundry runtime, and the SAP Business Application Studio integrated development environment
- A user account mapped to your SAP identity

SAP BTP Subaccount Entitlements

Your subaccount has the following entitlements with which you can perform the exercises:

- SAP HANA Cloud - hana

  An SAP HANA Cloud Database instance is already running

- SAP Business Application Studio - build-code

  This application is used to create graphical models and deploy them to the database instance and preview the data, among others. It also comes with an embedded SAP HANA Database Explorer to navigate the catalog of your database instance and execute SQL statements

  Note:
  In this landscape, SAP Business Application Studio is part of SAP Build Code. So, SAP Build Lobby is the central start page. In an exercise you will launch SAP Business Application Studio from SAP Build Lobby.

- SAP HANA Cloud - tools

  This application includes additional tools serving your SAP HANA Database instance, including SAP HANA Cockpit and Database Explorer

- Cloud Foundry runtime - standard

  The Cloud Foundry runtime is where the deployment of HDI Containers for SAP HANA modeling projects is done. Each HDI container materializing an HDB module is a service running in a Cloud Foundry space.

Work with Your Group Number in the Practice System

It's important to know your group number when using the practice system. You can find your group number in two ways:

- When you click the *Access* button to launch the practice system, a pop up appears, showing the group number.
- Go to *My Learning* → *Practice Systems*. The group number is displayed beneath the tile of the practice system.

Please make a note of your group number. It is an important piece of information, because in exercises values of some objects need to be replaced by # values.

Caution:
Looking at your group number, you may have to add leading zeroes.

Let's look at this example: suppose the pop up or *My Learning* displays your group number as 1.

- If the exercise asks you to replace ##, it will be 01.
- If the exercise asks you to replace ####, it will be 0001.

Note:
Please make sure you are always using your assigned group number, so you don't interfere with datasets used by another learner.

Note:
We also recommend to read the SAP Learning System Access: General User Guide to learn how to use the SAP Learning System Access portal.

Here you can find the technical information about your system and a step-by-step description to access the system(s):

Practice System Environment Recommendations

When using the practice system, please make sure that:

- You follow the instructions in the exercises, staying within the resource allocation limits of the course
- You do not share your user account and passwords with third parties except, if necessary, with your SAP Support channel.

Caution:
Some practice systems are shared by more than one learning journey. For example, *Developing Data Models in SAP HANA Cloud (SAP HANA Cloud Modeling)* and *Provisioning Data to SAP HANA Cloud*. In that case, if you have enrolled a practice system that serves one of them, it can also be used for the others.

If you choose *Enroll* for an additional practice system, you might get an error message saying: *The enrollment request is rejected as you are already enrolled in the system.* To perform the exercises, you need to access the practice system from the enrollment that is already active.

Note:
Do not close or navigate away from the Practice System page while you are working in a learning system. Some content may rely on this window to save your progress.

Note:
To end the session and stop subscription consumption, always close the tab.

Users in the Landscape

Some of the activities will be performed in an SAP Business Application Studio development space. Other activities mostly rely on the SAP HANA Database Explorer, which you can launch from SAP Business Application Studio.

In the training landscape, you will be using a number of users. The following table summarizes these different users:

System/Application	User Name/ID	Password
SAP Business Application Studio	[Single Sign-On from the SAP Learning Hub (Practice) portal]	[Single Sign-On from the SAP Learning Hub (Practice) portal]
SAP HANA Database Explorer	[Single Sign-On from the SAP Learning Hub (Practice) portal]	[Single Sign-On from the SAP Learning Hub (Practice) portal]
SAP HANA Cloud Database	**A####** [Single Sign-On from the SAP Learning Hub (Practice) portal]	[Single Sign-On from the SAP Learning Hub (Practice) portal]

Your assigned database user **A####** is derived from your group number. When using your assigned database user, make sure you always use the same convention: letter **A**, followed by your **group number on 4 digits**.

SAP HANA Cloud Database Instance

To connect SAP HANA Database Explorer to the right database, you will need the **host** and **port** of the SAP HANA Cloud database instance. In your landscape, the host and port of the database instance are as follows:

- Host:

  cb11e4ed-03fc-46e1-84dd-7ebb02f5af79.hna2.prod-eu10.hanacloud.ondemand.com

- Port:

  443

> Note:
> These two pieces of information are often grouped together (separated by a colon), to form the so-called *SQL Endpoint* of the database instance, which is **<Host>:<Port>**.

Specify an Identity Provider

When an authentication of your user is triggered, for example when logging on to SAP HANA Database Explorer, you might be asked to specify an identity provider, as shown in the screenshots below. It is used to map your logged in user to the relevant identities enabling you to access the system and perform the exercises.

Always choose identity provider **`cis4hcpractice-platform`**.

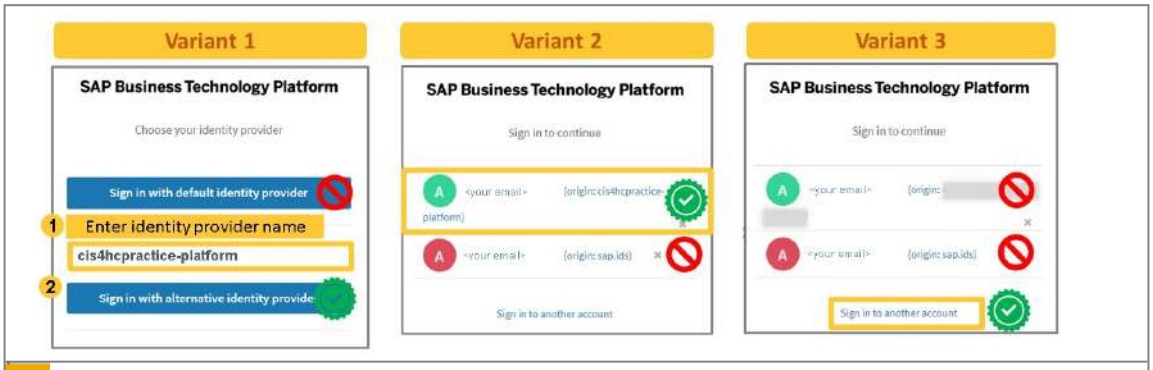


Figure 1: Choosing or Entering the Identity Provider Name

Depending on the dialog box that shows up, proceed as follows:

- Variant 1: No alternative identity provider listed

  In the text box with the text prompt *Enter the origin key of your identity provider*, enter **`cis4hcpractice-platform`**. Then choose *Sign in with alternative identity provider*.

- Variant 2: Your identity based on identity provider *cis4hcpractice-platform* is listed.

  Choose this identity.

- Variant 3: Identities are listed but none is based on identity provider*cis4hcpractice-platform*

  Choose *Sign in to another account* to display the *Variant 1* dialog box and specify identity provider **`cis4hcpractice-platform`**.

Cloud Foundry and Space Organization

Some of the activities you will perform involve the Cloud Foundry runtime. For example, deploying objects to the database.

At an early stage when configuring SAP Business Application Studio, you will log on to Cloud Foundry (with single sign-on), specifying the Cloud Foundry Organization and Cloud Foundry Space. Select the following values:

- Cloud Foundry Organization:

  SAP Practice Systems - HANA Cloud_hanacloud-sharedlearning-a000001-cf-eu10-sd

- Cloud Foundry Space:

  DEV

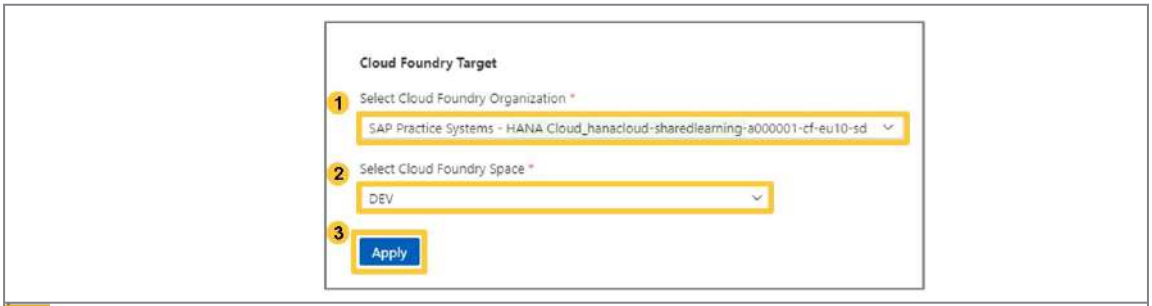


Figure 2: Choosing the Cloud Foundry Target

During your activity in SAP Business Application Studio, especially when resuming after a long break (typically at night), you might be asked to log back on to Cloud Foundry. In that case, please refer to the detailed instructions mentioned in Exercise 2 *Import an Existing Project*, task 4, steps 1 and 2.

4. Additional System Setup Tasks

None

The preparation of the practice system for this course is described in details in the first three exercises of your exercise handbook. It mostly consists in:

- Creating a development space in SAP Business Application Studio
- Downloading the necessary course files from a dedicated GitHub repository:https://github.com/SAP-samples/hana-cloud-modeling-learning-journey
- Importing a prepared project into SAP Business Application Studio, and building it.

  The project contains tables with data, calculation views, and so on.

- Copying additional course files needed for the exercises

  You will store these files on your computer in a folder called `HC300\` and, in the exercises, you will be instructed to open them or to upload them to SAP Business Application Studio.

- Customizing SAP Business Application Studio
- Connecting Database Explorer

5. Known System Issues

None

Creating Calculation Views - Dialog box shows the extension in the *Name* field

When you create a calculation view directly from the Explorer view (not with the wizard), you define the design-time file with its extension *.hdbcalculationview*. Then, in the *New Calculation View* dialog box, the (runtime) *Name* field includes the extension, where it should not.

Workaround:

The *Name* field cannot be edited at this stage, but anyway in the runtime name eventually generated, the extension will NOT be included, which is the expected behavior. This can be confirmed in the *View Properties* tab of the *Semantics*.

Avoid Copy/Pasting PDF Statements from the PDF Exercises Handbook – Prefer the Course Files

In general, SQL scripts used during the exercises are provided in the Exercise Handbook. This makes it easier to associate a script with the context in which you use it. However, due to PDF formatting, these scripts cannot always be copied/pasted from the handbook and executed as is. In particular, additional line breaks generated by the PDF generation can make the statement invalid, especially if they are located in the middle of a keyword, object name, etc.

Workaround:

SQL scripts are always provided as `.sql` or `.txt` files in your course files folder. It is recommended to use them as the source for upload or copy/paste, instead of the PDF handbook.

6. Additional Hints And Tips For Exercises of This Course

Like in many web-based applications, data caching sometimes affects access to a particular feature or execution of a workflow. If you suspect an anomaly could be due to data caching, close all instances of your web browser, then start your web browser again and use its native features to clear the cache. These features can be found in Google Chrome, Microsoft Edge and Mozilla Firefox under the *History* menu.

IMPORTANT: Please always complete the exercise you are working on before you suspend the system to avoid potential system inconsistencies and failure.

Caution:
Only one learning system can be accessed at a time.

7. Global Learning System Support

1. Go to SAP User Support Centerhttps://support.learning.sap.com/#.
2. Click *Need more help?* at the bottom of the page.
3. Click on *Create Case*.
4. Briefly describe your issue in the *Subject* field.
5. Click on the *Service Category* drop-down and select *Practice Systems*.
6. In the *Description*, copy and paste the text for Self-paced Learners in the box below.
7. Fill in the text with the necessary information about the issue.
8. Attach the screenshot of the issue if available (max 20MB). Click the option *choose a file or drag and drop*. Hit *Upload* when the file has been chosen.

9. Click *Submit*.

HC300

SAP HANA Cloud Modeling – Practical Exercises

EXERCISES AND SOLUTIONS

Course Version: 2410
Exercise Duration: 12 Hours 15 Minutes

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

Create a development space

Exercise Objectives

After completing this exercise, you will be able to:

- Launch SAP Business Application Studio from SAP Build Lobby
- Create a development space in SAP Business Application Studio

Business Example

You are at a customer site and want to set up data modeling scenarios using SAP Business Application Studio. You have to create the development environment for all future tasks related to SAP Business Application Studio.

Prerequisites

You need to have access to a valid Learning System Access landscape for course HC300. For the preparatory steps, please consult the System Setup Guide as well as the SAP Learning System Access: General User Guide.

1. Launch SAP Business Application Studio from SAP Build Lobby.

   Note:
   In this landscape, SAP Business Application Studio is part of SAP Build Code. So, SAP Build Lobby is the central start page.

2. In SAP Business Application Studio, create a new development space named **`HCMOD_A####`** of type *SAP HANA Native Application*. Add the *SAP HANA Performance Tools* extension.

   `A####` represents your SAP HANA user name, based on your the group number assigned to you when you provisioned the training system from the Learning System Access portal. Please refer to your System Setup Guide for more information.

   Result
   The new *HCMOD_A####* development space is created and started. This can take a couple of minutes.

3. Open the newly created development space.

   Caution:
   After some idle time, a development space is automatically shut down to preserve resources. In that case, you need to start it before you can open it.

   To do that, on the list of development spaces (SAP Business Application Studio landing page), click the *Start* icon of your development space.

Result

SAP Business Application Studio opens your development space *HCMOD_A####*, showing the *Welcome* tab.

Unit 1
Solution 1

Create a development space

Exercise Objectives

After completing this exercise, you will be able to:

- Launch SAP Business Application Studio from SAP Build Lobby
- Create a development space in SAP Business Application Studio

Business Example

You are at a customer site and want to set up data modeling scenarios using SAP Business Application Studio. You have to create the development environment for all future tasks related to SAP Business Application Studio.

Prerequisites

You need to have access to a valid Learning System Access landscape for course HC300. For the preparatory steps, please consult the System Setup Guide as well as the SAP Learning System Access: General User Guide.

1. Launch SAP Business Application Studio from SAP Build Lobby.

   Note:
   In this landscape, SAP Business Application Studio is part of SAP Build Code. So, SAP Build Lobby is the central start page.

   a) On SAP Build Lobby, on top right of your screen, choose *Product Switch*.

   b) Choose *Dev Space Manager*.

   Result
   The SAP Business Application Studio is launched.

2. In SAP Business Application Studio, create a new development space named **HCMOD_A####** of type *SAP HANA Native Application*. Add the *SAP HANA Performance Tools* extension.

   A#### represents your SAP HANA user name, based on your the group number assigned to you when you provisioned the training system from the Learning System Access portal. Please refer to your System Setup Guide for more information.

   a) In SAP Business Application Studio, choose *Create Dev Space*.

   b) In the *Dev Space name* field, enter **HCMOD_A####** .

   A#### represents your SAP HANA user name, based on your the group number assigned to you when you provisioned the training system from the Learning System Access portal. Please refer to your System Setup Guide for more information.

c) For the application type, select *SAP HANA Native Application*.

d) Select the additional extension *SAP HANA Performance Tools*.

e) Choose *Create Dev Space*.

Result

The new *HCMOD_A####* development space is created and started. This can take a couple of minutes.

3. Open the newly created development space.

> Caution:
> After some idle time, a development space is automatically shut down to preserve resources. In that case, you need to start it before you can open it.
>
> To do that, on the list of development spaces (SAP Business Application Studio landing page), click the *Start* icon of your development space.

a) Once the new development space shows a **RUNNING** status, click on the space name to open it.

Result

SAP Business Application Studio opens your development space *HCMOD_A####*, showing the *Welcome* tab.

Unit 1
Exercise 2

Import an Existing Project

Exercise Objectives

After completing this exercise, you will be able to:

- Connect SAP HANA Database Explorer to an SAP HANA Cloud database instance
- Download the course files supporting this learning journey from GitHub
- Import an existing project into your SAP Business Application Studio development space.
- Connect to Cloud Foundry
- Bind your HDB Module to a new SAP HANA DI Container
- Deploy your project

Business Example

You are at a customer site and want to start modeling in SAP Business Application Studio. A colleague already created a project for you to start with. This is the standard project used in the enterprise. You need to import it into your development space.

Overview of Exercise Tasks

- Task 1: Connect SAP HANA Database Explorer to your SAP HANA Cloud database instance
- Task 2: Download the resources needed to perform exercises
- Task 3: Import an existing Modeling project
- Task 4: Deploy the project to a new SAP HANA DI Container

Prerequisites

Your SAP Business Application Studio development environment should have been set up.

Note:
After some idle time, a development space is automatically shut down to preserve resources. In that case, you need to start it before you can open it.

Task 1: Connect SAP HANA Database Explorer to your SAP HANA Cloud database instance

1. From SAP Business Application Studio, launch SAP HANA Database Explorer.

   Result

   SAP HANA Database Explorer opens in a new browser tab.

Note:
Your SAP HANA Database Explorer might show no connections at all, except if you have already connected other SAP HANA instances in Database Explorer with the same identity and in the same SAP BTP region (eu10).

2. Create a connection to the SAP HANA Cloud database instance. The database instance details are provided in your SAP Learning System Access setup guide. You will use single sign-on (SSO) authentication.

   Result

   The connection to the SAP HANA Cloud database instance is created.

3. Confirm your assigned SAP HANA Cloud database user A####.

   This can be done by executing the SQL statement:`SELECT SESSION_USER FROM DUMMY;`

Task 2: Download the resources needed to perform exercises

1. Create a new folder `HC300` to store all the course files on your computer. Choose a location that you can easily access and/or create a shortcut to this folder.

   Later on, any course file you use can be found in this folder, always referred to as `HC300` or *the HC300 folder*.

2. Download the course files from GitHub.

   The repository URL is: https://github.com/SAP-samples/hana-cloud-modeling-learning-journey. It contains both an existing modeling project (in a .zip archive) and several files organized into different folders that you will need for some activities. The target folder for the download is the new `HC300` folder you created in the previous step.

3. On your computer, in the `HC300` folder, unzip the `HC300_2410_Course_Files.zip` archive.

Note:
There is NO need for an extra subfolder such as *HC300_2410_Course_Files*. So extracting the archive directly into your `HC300` folder is the simplest approach.

Note:
Depending on your operating system and tools, in particular your zip/unzip tool, the steps to extract the archive can vary. In the end, the objective is to get a folder structure such as `C:\HANA\HC300\` with all the content extracted from the `.zip` archive.

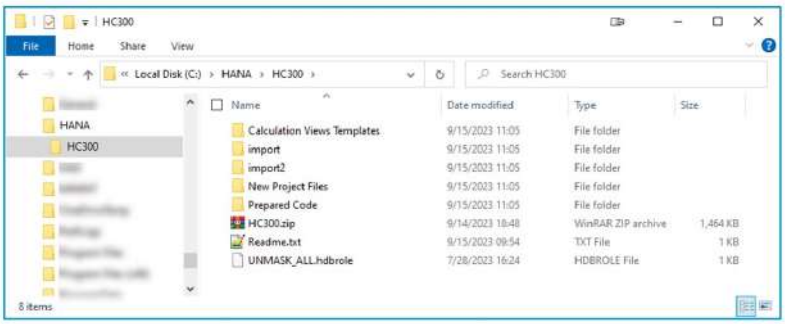


Figure 2: Extracted Course Files

Task 3: Import an Existing Modeling Project

1. Import the project you will use for most of your activities into SAP Business Application Studio.

   The project is provided as an `HC300.zip` file. It will be imported into the `/home/user/projects` folder.

2. Rename the project folder, adding _ and your student number **`A####`**.

> Hint:
> To rename a folder, it must be displayed in the *Explorer* view, but NOT as a primary folder in a workspace. So you need to open the parent folder `projects` first.

3. Open the renamed folder in the *Explorer* view.

   This is important to make sure you won't touch any file located in the other child folders of `projects`.

   Alternatively, you could also open the project in a workspace. This can be done from the *Get Started* window, by clicking directly the project folder in the *Projects* list. To keep it simple, a workspace is a list of folders that can be shown together in the SAP Business Application Studio*Explorer* pane, and share a number of settings.

4. Check that the course resources are present.

   The project is made up of one SAP HANA Database (HDB) module, stored in the `db` folder, where you will store all modeling objects you create. It already contains the design-time source files of a number of tables, calculation views and other objects that you will use during the activities.

   Within the `db/src/` folder, the design-time objects will be organized in different sub-folders, as follows:

   - **data**

     Contains the definition of tables (`.hdbtable` files), the data (`.csv` files) and the corresponding mapping definition (`.hdbtabledata` files).

   - **external_data**

     Contains references to external objects and data that are not stored in your project itself.

- models

  Contains ready-made calculation views that you will use as sources (`resources` folder), and will be used to store all the modeling artifacts that you will create during the training (`exercises` folder) and .

  This folder also contains a `solutions` folder, which includes the final (and also, when relevant, intermediate) stage of modeling exercises. You can consult these solution objects, for example, to compare them with the ones you are creating in the `exercises` folder.

> Note:
> To distinguish solution objects from the ones you will develop, all the solution objects have **_00** in their name. For example, *CVC_SO_00* is the solution object that corresponds to the *CVC_SO* calculation view that you will build during the course.

Task 4: Deploy the Project to a New SAP HANA DI Container

In order to deploy the project into SAP HANA, you need to first connect to Cloud Foundry, and then bind the HDB module of your project to an HDI container. You will create a new HDI container.

1. Log on to Cloud Foundry.

   You will authenticate using Single-Sign-On (SSO).

2. Choose the relevant Cloud Foundry target to deploy your modeling content in.

   This target is defined by the two following elements, in the same tab, *Login to Cloud Foundry*:

Cloud Foundry Organization	*SAP Practice Systems - HANA Cloud_hanacloud-sharedlearning-a000001-cf-eu10-sd*
Cloud Foundry Space	*DEV* (default space in your Cloud Foundry Organization)

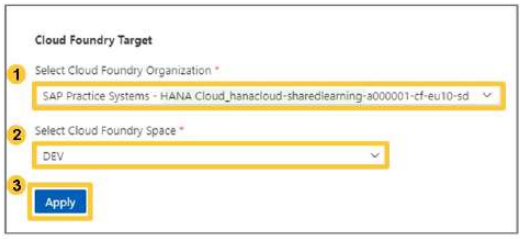


Figure 3: Choosing the Cloud Foundry Target

> Caution:
> At some point during your modeling activities, especially overnight, your connection to Cloud Foundry might time out. In that case, when resuming your exercises, log back in to Cloud Foundry and choose the target Organization and Space, as specified in the two last steps.

3. Bind the HANA database module of your project to a new HDI Container with the following name : **`HC300_A####-hdidb-ws-xxxxx`** (where A#### is your assigned student number).

   Use the default service instance name proposed by the SAP Business Application Studio wizard.

4. Bind the external service *cross_container_service_2* with the *EXT_SCHEMA_SERVICE* service.

5. Deploy the project.

6. Close the Terminal window.

Import an Existing Project

Exercise Objectives

After completing this exercise, you will be able to:

- Connect SAP HANA Database Explorer to an SAP HANA Cloud database instance
- Download the course files supporting this learning journey from GitHub
- Import an existing project into your SAP Business Application Studio development space.
- Connect to Cloud Foundry
- Bind your HDB Module to a new SAP HANA DI Container
- Deploy your project

Business Example

You are at a customer site and want to start modeling in SAP Business Application Studio. A colleague already created a project for you to start with. This is the standard project used in the enterprise. You need to import it into your development space.

Overview of Exercise Tasks

- Task 1: Connect SAP HANA Database Explorer to your SAP HANA Cloud database instance
- Task 2: Download the resources needed to perform exercises
- Task 3: Import an existing Modeling project
- Task 4: Deploy the project to a new SAP HANA DI Container

Prerequisites

Your SAP Business Application Studio development environment should have been set up.

Note:
After some idle time, a development space is automatically shut down to preserve resources. In that case, you need to start it before you can open it.

Task 1: Connect SAP HANA Database Explorer to your SAP HANA Cloud database instance

1. From SAP Business Application Studio, launch SAP HANA Database Explorer.

   a) Click the  *(Business Application Studio Menu)* icon top left and choose *View* → *Command Palette*.

   Alternatively, you can press **`Ctrl+Shift+P`** or insert the character **>** inside the *Search Bar*.

b) In the *Command Palette*, enter **`Database Expl`**.

c) Choose *SAP HANA: Open Database Explorer*.

d) If a browser tab shows up to choose an identity provider, choose *origin: cis4hcpractice-platform*.

You must either specify the name of the identity provider (variant 1), that is, **`cis4hcpractice-platform`**, or choose in the list of identities the one that is associated with it (variant 2). If several identity providers are listed but not is *cis4hcpractice-platform* (variant 3), choose *Sign in to another account* and proceed with the variant 1 workflow.

Figure 1: Choosing an Identity Provider

Result

SAP HANA Database Explorer opens in a new browser tab.

> Note:
> Your SAP HANA Database Explorer might show no connections at all, except if you have already connected other SAP HANA instances in Database Explorer with the same identity and in the same SAP BTP region (eu10).

2. Create a connection to the SAP HANA Cloud database instance. The database instance details are provided in your SAP Learning System Access setup guide. You will use single sign-on (SSO) authentication.

a) In SAP HANA Database Explorer, choose **+** *(Add an instance to the Database Explorer)*.

b) In the *Instance Type* dropdown list, choose *SAP HANA Database*.

c) Enter the Host and Port in the respective fields.

You can find this information in your SAP Learning System Access setup guide (section "Logon Information to Access SAP Training Systems" >> "SAP HANA Cloud Database Instance").

> Caution:
> Make sure you select the radio button in front of the *Port Number* field.

d) For the *Authentication Mode*, choose *Single sign-On*.

e) Select the *Connect to the database securely using TLS/SSL* check-box.

f) If it is selected, DEselect the *Verify the server's certificate using the trusted certificate below* check-box.

g) In the *Display name*, enter **`HC300_DB (A####) (SSO enabled)`** where **`A####`** is your assigned database user name.

h) Choose *OK*.

Result

The connection to the SAP HANA Cloud database instance is created.

3. Confirm your assigned SAP HANA Cloud database user A####.

   This can be done by executing the SQL statement:`SELECT SESSION_USER FROM DUMMY;`

   a) Select the new connection *HC300_DB (A####) (SSO enabled).*

   b) Top-left of the instances list, choose *Open SQL Console*.

   c) Enter the above SQL statement.

   d) Choose  *(Run)* or press **`F8`**.

   e) Check that your assigned user number **`A####`** is consistent with the one you got from the tile of your SAP Learning System Access instance (on the SAP Learning System Access portal).

Task 2: Download the resources needed to perform exercises

1. Create a new folder `HC300` to store all the course files on your computer. Choose a location that you can easily access and/or create a shortcut to this folder.

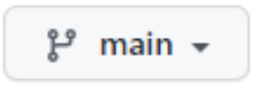
   Later on, any course file you use can be found in this folder, always referred to as `HC300` or *the HC300 folder*.

2. Download the course files from GitHub.

   The repository URL is: https://github.com/SAP-samples/hana-cloud-modeling-learning-journey. It contains both an existing modeling project (in a .zip archive) and several files organized into different folders that you will need for some activities. The target folder for the download is the new `HC300` folder you created in the previous step.

   a) In your web browser, go to the GitHub repository https://github.com/SAP-samples/hana-cloud-modeling-learning-journey

   b) In the branch selector, make sure that the branch *main* is currently selected

   .

   c) Select the `HC300_2410_Course_Files.zip` item.

   d) Click the top-right **...** ellipsis button and choose *Download*.

   e) Save the downloaded `.zip` file in your `HC300` folder.

   If you are not prompted for a target folder, it will generally default to a folder such as `Downloads`, easily accessible from your file explorer or from the browser itself. Then, you just need to move the downloaded archive to your `HC300` folder.

> Caution:
> This archive is NOT a ready-to-import project for SAP Business Application Studio, but it contains one, as well as a number of files such as SQL code, templates, and so on. So you must first unzip/extract the archive to see its content and work with it.

3. On your computer, in the `HC300` folder, unzip the `HC300_2410_Course_Files.zip` archive.

> Note:
> There is NO need for an extra subfolder such as *HC300_2410_Course_Files*. So extracting the archive directly into your `HC300` folder is the simplest approach.

a) Right-click the `.zip` file and choose *Extract here*.

b) If the extracted files are not located as expected, move them together to the right location.

c) After the `HC300_2410_Course_Files.zip` archive as been extracted, you can delete this archive.

> Note:
> Depending on your operating system and tools, in particular your zip/unzip tool, the steps to extract the archive can vary. In the end, the objective is to get a folder structure such as `C:\HANA\HC300\` with all the content extracted from the `.zip` archive.

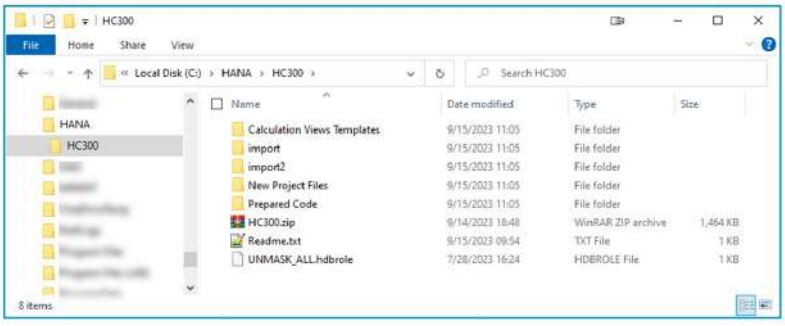


Figure 2: Extracted Course Files

Task 3: Import an Existing Modeling Project

1. Import the project you will use for most of your activities into SAP Business Application Studio.

   The project is provided as an `HC300.zip` file. It will be imported into the `/home/user/projects` folder.

   a) Go back to Business Application Studio.

   b) Select the *Explorer* icon in the Activity bar (left side bar).

c) Choose *Open Folder*.

> Note:
> If you have already worked with the *Explorer* view in this development space, click the *(Business Application Studio Menu)* icon top left and choose *File → Open Folder* instead.

d) Select the `/home/user/projects` folder.

e) Choose *OK*.
If a message asking if you want to leave the page appears, choose *Leave*.

f) In the opened *Get Started* tab, choose *Import*.

> Hint:
> If you do not see this *Get Started* tab, you can open it by choosing the top left menu icon, then choose *Help → Get Started*.

g) In your course files, find the previously extracted `HC300.zip` file and choose *Open*.

h) When the project import is completed, you should see a message on the bottom right corner telling you that the project has been imported.

> Note:
> The project is imported automatically in a new workspace.

2. Rename the project folder, adding _ and your student number **`A####`**.

> Hint:
> To rename a folder, it must be displayed in the *Explorer* view, but NOT as a primary folder in a workspace. So you need to open the parent folder `projects` first.

a) Choose *File → Open Folder*.

b) Select the folder *projects* and choose *OK*.

c) In the *Explorer* view, right-click the *HC300* folder and choose *Rename*.
Alternatively, you can press **`F2`**.

d) Adjust the folder name to **`HC300_A####`** and press **`Enter`**.

3. Open the renamed folder in the *Explorer* view.
This is important to make sure you won't touch any file located in the other child folders of `projects`.

a) Click the *(Menu)* icon and choose *File → Open Folder*.

b) Select the `projects` folder.

c) Select the `HC300_A####` folder and choose *OK*.

Result

The `HC300_A####` folder is now exposed as the root folder in the file explorer. It contains a number of files and sub-folders corresponding to an SAP HANA Cloud modeling project.

d) If needed, close the *Get Started* and *Welcome* tabs.

Alternatively, you could also open the project in a workspace. This can be done from the *Get Started* window, by clicking directly the project folder in the *Projects* list. To keep it simple, a workspace is a list of folders that can be shown together in the SAP Business Application Studio*Explorer* pane, and share a number of settings.

4. Check that the course resources are present.

The project is made up of one SAP HANA Database (HDB) module, stored in the `db` folder, where you will store all modeling objects you create. It already contains the design-time source files of a number of tables, calculation views and other objects that you will use during the activities.

Within the `db/src/` folder, the design-time objects will be organized in different sub-folders, as follows:

- **data**

  Contains the definition of tables (`.hdbtable` files), the data (`.csv` files) and the corresponding mapping definition (`.hdbtabledata` files).

- **external_data**

  Contains references to external objects and data that are not stored in your project itself.

- **models**

  Contains ready-made calculation views that you will use as sources (`resources` folder), and will be used to store all the modeling artifacts that you will create during the training (`exercises` folder) and .

  This folder also contains a `solutions` folder, which includes the final (and also, when relevant, intermediate) stage of modeling exercises. You can consult these solution objects, for example, to compare them with the ones you are creating in the `exercises` folder.

Note:
To distinguish solution objects from the ones you will develop, all the solution objects have **_00** in their name. For example, *CVC_SO_00* is the solution object that corresponds to the *CVC_SO* calculation view that you will build during the course.

Task 4: Deploy the Project to a New SAP HANA DI Container

In order to deploy the project into SAP HANA, you need to first connect to Cloud Foundry, and then bind the HDB module of your project to an HDI container. You will create a new HDI container.

1. Log on to Cloud Foundry.

   You will authenticate using Single-Sign-On (SSO).

   a) Click the *Menu* icon and choose *View* → *Command Palette*

      Alternatively, you can use the top search bar and type **>** before your entry.

   b) Enter **`CF`** in the command line. A list of Cloud Foundry commands should display.

   c) Choose **Login to Cloud Foundry**.

   d) Select the authentication method *SSO Passcode* and click the link *Open a new browser page...* just below.

   e) If you get a warning message related to opening an external website, choose *Open*.

   f) If needed, confirm your SAP BTP Sign-in identity. It must be consistent with the identity you used to access the SAP Learning System Access portal, and must use the identity provider *cis4hcpractice-platform*.

      Consult your SAP Learning System Access - System Setup Guide for more details, if needed.

   g) On the new browser tab, to copy the temporary password, choose the *copy* icon. Then close the tab.

   h) Back to SAP Business Application Studio, paste the temporary password and choose *Sign in*.

      Result

      You have been signed in to Cloud Foundry is displayed in a notification box bottom-right.

2. Choose the relevant Cloud Foundry target to deploy your modeling content in.

   This target is defined by the two following elements, in the same tab, *Login to Cloud Foundry*:

Cloud Foundry Organization	*SAP Practice Systems - HANA Cloud_hanacloud-sharedlearning-a000001-cf-eu10-sd*
Cloud Foundry Space	*DEV* (default space in your Cloud Foundry Organization)

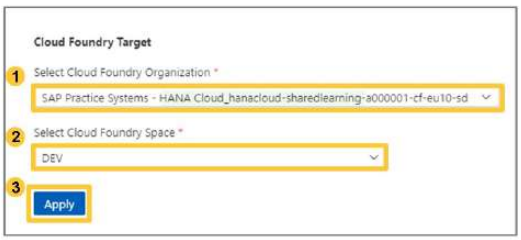


Figure 3: Choosing the Cloud Foundry Target

   a) Select the Cloud Foundry organization as specified

   b) Select the *DEV* Cloud Foundry space

   c) Choose *Apply*.

Caution:
At some point during your modeling activities, especially overnight, your connection to Cloud Foundry might time out. In that case, when resuming your exercises, log back in to Cloud Foundry and choose the target Organization and Space, as specified in the two last steps.

3. Bind the HANA database module of your project to a new HDI Container with the following name : **`HC300_A####-hdidb-ws-xxxxx`** (where A#### is your assigned student number).

   Use the default service instance name proposed by the SAP Business Application Studio wizard.

   a) In the *Explorer* view, expand the *SAP HANA Projects* sub-pane. It is located by default at the very bottom of the *Explorer* view.

   b) Under *HC300_A####/db → Database Connections*, select *hdi_db*.

   If needed, to refresh the list of HDB modules, click the *Synchronize* icon next to the *HC300_A####/db* entry

   c) Click on the green *Bind* icon on the right of *hdi_db*.

   d) In the *Select a binding option* drop down, choose **Bind to an HDI Container**.

   e) In the *Select an SAP HANA Service* drop down, Choose **Bind to the default service instance: HC300_A####-hdidb-ws-xxxxx**.

   Result

   A message displays at the bottom right of the screen saying that instance creation has started.

   f) Wait until the creation and binding of of the HDI container instance are complete.

   Result

   A message saying the *HC300_A####-hdidb-ws-xxxxx* service has been bound should be displayed in the bottom right of the screen. In the *Explorer* view, in the *SAP HANA Projects* sub-pane, the *HC300_A####/db → Database Connections → hdi_db (...)* icon is showing a green plug and the name of the bound service in parentheses.

   g) In the automatic undeployment related idalog box, answer *Enable and do not ask again*.

4. Bind the external service *cross_container_service_2* with the *EXT_SCHEMA_SERVICE* service.

   a) In the *Explorer* view, expand the *SAP HANA Projects* sub-pane

   b) Expand the *Database Connections* folder.

   Result

   You should see two entries. One for the *hdi_db* connection and one for *cross_container_service_2*.

   c) Click the green *Bind* icon on the right of the entry *cross-container-service-2*.

   d) In the drop-drown list, select *Bind to a user provided service*.

   e) Select the *EXT_SCHEMA_SERVICE* service.

Notice that, because the service name mentioned in the *mta.yaml* file is in sync with the actual name of an existing user-provided service in the *DEV* space, another option offers to execute the binding to the EXT_SCHEMA_SERVICE in a single click. You can choose either option.

Result

The cross container service is now bound. *(EXT_SCHEMA_SERVICE)* should now appear after the (logical) name of the service *cross-container-service-2*.

5. Deploy the project.
   a) In the *Explorer* view, expand the *SAP HANA Projects* sub-pane
   b) On the right of the *HC300_A####/db* item, which is the root of your HDB module, click on the  *(Deploy)* icon.

      Result

      The *Terminal* displays the deployment trace.
   c) Check that the deployment was successful.

      Result

      The HDB module of your project is now deployed inside the HDI container, meaning that all the database artifacts have been created. The mention *Pending Deployment* that was showing up next to each database object is gone.
6. Close the Terminal window.
   a) Click the X *(Close Panel)* icon at the right of the terminal header.

Unit 1
Exercise 3

Prepare the Training Environment

Exercise Objectives

After completing this exercise, you will be able to:

- Set up your SAP Business Application Studio development environment.
- Navigate SAP Business Application Studio
- Connect Database Explorer to an HDI Container

Business Example

You are at a customer site and want to start modeling in SAP Business Application Studio. You want to setup your environment for better productivity and familiarize yourself with the user interface.

Overview of Exercise Tasks

- Set up menus, tool bars and options in SAP Business Application Studio
- Connect the Database Explorer to your HDI Container
- Navigate your project in SAP Business Application Studio
- Grant access to the HDI container schema to your new user

Prerequisites

You should have created a development space into SAP Business Application Studio, imported and deployed the existing *HC300* project.

Task 1: Set up Menus, Tool Bars and Options in SAP Business Application Studio

To simplify the use of SAP Business Application Studio and improve your productivity, you will arrange menus, hide unused options and icons, and set up some basic options.

Caution:
In all the upcoming exercises, the instructions are based on the settings of SAP Business Application Studio (or its extensions) defined below. You can of course test other values for these settings, or modify other settings, but this might require minor adjustments to the instructions.

1. Make the menu bar always visible, on top.
2. Hide a number of activity bar icons that you will not need for this course. Only keep the following icons : *Explorer*, *Search* , *Source Control*, *HANA SQL Analyzer*, *Guide Center* and *SAP HANA Database Explorer*.
3. In the *Explorer* view, hide the *OUTLINE* and *TIMELINE* sub-panes.

4. Enable word wrapping in the text editor and check automatic save is enabled with a delay of 1000 milliseconds.

> Caution:
> With the *Auto Save* setting, you never lose work you have done on a file. However, make sure NOT to do any action such as testing the behavior of an editor on a "precious" object. Use dummy objects/copies instead.

Task 2: Connect the Database Explorer to your HDI Container

You will now connect your SAP HANA Database Explorer to the HDI Container created as part of the binding and deployment of your *HC300_A####* project. This must be done first in the external Database Explorer so that the connection can be viewed from within SAP Business Application Studio. You will also give a friendly name to this connection.

> Note:
> The external Database Explorer is the reference tool to visualize database objects and execute SQL. It is used by different roles, including admin users, and also by modelers as an alternative to the embedded Database Explorer of SAP Business Application Studio.

1. Add the HDI container of your deployed modeling project *HC300_A####* to the SAP HANA Database Explorer.

   The HDI container can be added automatically by launching the SAP HANA Database Explorer from SAP Business Application Studio (*SAP HANA Projects* sub-pane).

2. Modify the display name of your HDI Container connection to make it simpler.

   Use the following pattern: **`HC300_A####_HDI_CONTAINER`**, where A#### is your assigned user number

3. Check that your SAP HANA Cloud instance database connection and the renamed HDI container connection are now visible in SAP Business Application Studio.

> Note:
> In the Database Explorer, all databases and HDI container instances you have access to might be listed if they have already been added to Database Explorer (entries are not filtered by SAP BTP account or subaccount). You can filter the entries if needed.

Task 3: Navigate Your Project in SAP Business Application Studio

You will now familiarize yourself with the user interface, first in the *Explorer* view, then in the *Search* and *SAP HANA Database Explorer* views.

1. Navigate through the HC300_A#### project to look at data and models folders. Open the two files `Employees.hdbtable` and `CVD_ADRESSES.hdbcalculationview`.

2. Search for all files containing the string **`Employees`**. Then narrow down the result set to show only files containing **`Employees`** with matching case.

3. In the *SAP HANA Database Explorer* view, open the connection to your HDI container and display the data of the *HC::EMPLOYEES* table and the *HC::CVD_ADDRESSES* Column View.

Task 4: Grant Access to the HDI Container Schema to Your A#### User

Most of the modeling exercises in this learning journey will rely on the connection to the HDI Container, for example when previewing the data. However, a few exercises explore how a standard database user can query the data of an HDI Container.

To make this possible, your user *A####* must have the privilege to query data from the container. A generic role for enabling data access to your container's content has been created when you deployed the *HC300_A####* project for the first time.

The SQL statements to grant this role can be found in the file `SQL for External HDI Container Access.txt`. These statements must be executed in your SAP HANA Cloud database, Catalog View (NOT the HDI container), connected with user *A####*.

1. In the file explorer of your computer, open the script from your course files and copy its content to an SQL console connected to the *HC300_DB (A####) (SSO enabled)* connection.

```
SELECT * FROM ROLES WHERE ROLE_NAME LIKE '%HC300_A####%access_role';

CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','','HC300_A####_HDI_DB_1::acces
s_role', ?);
```

2. Adapt the script so that it uses your assigned user number **A####**.

3. Execute the SQL statements to grant your course participant role some privileges on your container.

Note:
You have two options to execute a single SQL statement in the SQL console:

- Select the statement an choose *Run (F8)*.
- Place the cursor anywhere within the statement and choose *Run Statement (F9)*, which is one of the possible options in the drop-down list next to the *Run* button.

4. Check that the statement was executed successfully. The result should show a message saying *Operation successful!* together with the executed statement.

Unit 1
Solution 3

Prepare the Training Environment

Exercise Objectives

After completing this exercise, you will be able to:

- Set up your SAP Business Application Studio development environment.
- Navigate SAP Business Application Studio
- Connect Database Explorer to an HDI Container

Business Example

You are at a customer site and want to start modeling in SAP Business Application Studio. You want to setup your environment for better productivity and familiarize yourself with the user interface.

Overview of Exercise Tasks

- Set up menus, tool bars and options in SAP Business Application Studio
- Connect the Database Explorer to your HDI Container
- Navigate your project in SAP Business Application Studio
- Grant access to the HDI container schema to your new user

Prerequisites

You should have created a development space into SAP Business Application Studio, imported and deployed the existing *HC300* project.

Task 1: Set up Menus, Tool Bars and Options in SAP Business Application Studio

To simplify the use of SAP Business Application Studio and improve your productivity, you will arrange menus, hide unused options and icons, and set up some basic options.

Caution:
In all the upcoming exercises, the instructions are based on the settings of SAP Business Application Studio (or its extensions) defined below. You can of course test other values for these settings, or modify other settings, but this might require minor adjustments to the instructions.

1. Make the menu bar always visible, on top.
   a) Click the  *(Business Application Studio Menu)* top left.
   b) Select *View* → *Appearance* → *Menu Bar*
   c) The menu bar should now be displayed on top.

> Note:
> If the menu items do not show up immediately, try to refresh the current browser tab (SAP Business Application Studio).

2. Hide a number of activity bar icons that you will not need for this course. Only keep the following icons : *Explorer*, *Search* , *Source Control*, *HANA SQL Analyzer*, *Guide Center* and *SAP HANA Database Explorer*.
   a) To remove the *Run and Debug* icon, right-click on it on the Activity bar on the left, and choose the first entry in the context menu: *Hide 'Run and Debug'*.

   > Hint:
   > Alternatively, you can right-click anywhere on the Activity bar on the left and deselect *Run and Debug* from the views list.

   b) Repeat this sub-step for all other views to hide.
3. In the *Explorer* view, hide the *OUTLINE* and *TIMELINE* sub-panes.
   a) Make sure the *Explorer* view is opened. If not, select the *Explorer* icon in the Activity bar.
   b) Right-click on the *OUTLINE* sub-pane and choose *Hide 'Outline'*.
   c) Right-click on the *TIMELINE* sub-pane and choose *Hide 'Timeline'*.
4. Enable word wrapping in the text editor and check automatic save is enabled with a delay of 1000 milliseconds.

   > Caution:
   > With the *Auto Save* setting, you never lose work you have done on a file. However, make sure NOT to do any action such as testing the behavior of an editor on a "precious" object. Use dummy objects/copies instead.

   a) Select the *Manage* icon (cogwheel) on the bottom left and choose *Settings*.
   b) In the *Search settings* field, enter **`word wrap`**.
   c) Set the *Editor : Word Wrap* option to **on**.
   d) In the *Search settings* field, enter **`auto save`**.
   e) Check that *Files : Auto Save* is set on **afterDelay** and that *Files: Auto Save Delay* is set to **1000** milliseconds.
   f) Close the *Settings* tab.

Task 2: Connect the Database Explorer to your HDI Container

You will now connect your SAP HANA Database Explorer to the HDI Container created as part of the binding and deployment of your *HC300_A####* project. This must be done first in the external Database Explorer so that the connection can be viewed from within SAP Business Application Studio. You will also give a friendly name to this connection.

> Note:
> The external Database Explorer is the reference tool to visualize database objects and execute SQL. It is used by different roles, including admin users, and also by modelers as an alternative to the embedded Database Explorer of SAP Business Application Studio.

1. Add the HDI container of your deployed modeling project *HC300_A####* to the SAP HANA Database Explorer.

   The HDI container can be added automatically by launching the SAP HANA Database Explorer from SAP Business Application Studio (*SAP HANA Projects* sub-pane).

   a) In the *Explorer* view of SAP Business Application Studio, expand the *SAP HANA Projects* sub-pane.

   b) On the right of the *HC300_A####/db* element, choose the *Open HDI Container* icon.

   c) If a dialog box about opening an external website shows up, choose *Open*.

   > Note:
   > You will not do this very often as part of this learning journey, so just choosing *Open* is fine. Otherwise, you can add `hana-cockpit` (wherein SAP HANA Database Explorer is exposed) as a trusted domain.

2. Modify the display name of your HDI Container connection to make it simpler.

   Use the following pattern: **`HC300_A####_HDI_CONTAINER`**, where A#### is your assigned user number

   a) In the database list, locate your HDI Container connection, which has the pattern *SharedDevKey@HC300_A####-hdidb-ws-xxxxx*. It should be the active/open one.

   Note that is starts with your project name, and includes the development space id (ws-xxxxx).

   b) Right-click the connection name and choose *Properties*.

   c) Replace the *Display Name* with **`HC300_A####_HDI_CONTAINER`** (A#### is your assigned user number) and choose *OK*.

3. Check that your SAP HANA Cloud instance database connection and the renamed HDI container connection are now visible in SAP Business Application Studio.

   a) Back to SAP Business Application Studio, in the Activity bar, choose *Database Explorer*.

   b) If needed, in the *Database List* sub-pane, choose *Refresh Database List*.

   c) Check that your two *HC300...* connections in the database list are present with the expected names:

   - HC300_DB (A####) (SSO enabled): this is the connection to the entire database
   - HC300_A####_HDI_CONTAINER: this is the restricted connection to a single HDI container

Note:
In the Database Explorer, all databases and HDI container instances you have access to might be listed if they have already been added to Database Explorer (entries are not filtered by SAP BTP account or subaccount). You can filter the entries if needed.

Task 3: Navigate Your Project in SAP Business Application Studio

You will now familiarize yourself with the user interface, first in the *Explorer* view, then in the *Search* and *SAP HANA Database Explorer* views.

1. Navigate through the HC300_A#### project to look at data and models folders. Open the two files `Employees.hdbtable` and `CVD_ADRESSES.hdbcalculationview`.
   a) Expand your HC300_A#### project in the *Explorer* view down to `HC300_A####/db/src/data/TRAINING`.
   b) Choose the file `Employees.hdbtable` to open it in the text editor.
   c) Expand your HC300_A#### project in the *Explorer* view down to `HC300_A####/db/src/models/resources`.
   d) Choose the file `CVD_ADRESSES.hdbcalculationview` to open it in the corresponding visual editor.
   e) Close the calculation view tab.
2. Search for all files containing the string **`Employees`**. Then narrow down the result set to show only files containing **`Employees`** with matching case.
   a) In the Activity bar, click the *Search* icon.
   b) In the search field, enter **`Employees`**. You should see all files containing *employees*, regardless of the case.
   c) Click the *Match Case* icon at the right of the search field.
   Result
   All the previous matches containing **`EMPLOYEES`** (in upper case) are filtered out.
3. In the *SAP HANA Database Explorer* view, open the connection to your HDI container and display the data of the *HC::EMPLOYEES* table and the *HC::CVD_ADDRESSES* Column View.
   a) In the *SAP HANA Database Explorer* view, expand the connection to your HDI container *HC300_A####_HDI_CONTAINER*.

Hint:
In general, if there are several or a lot of items in the database list, for example because you have some from other projects or trainings, you can filter the list. Use the *Select Databases* (funnel) icon, enter **`HC300`** and check the databases and HDI connections you want to display.

   b) Select *Tables*.

Result
A list of all the tables should appear in the *Catalog Browser* sub-pane.

c) Right-click the *HC::Employees* table and choose *Open Data*.

d) The query should be run automatically and show data in a result tab. If not, in the open SQL tab, choose the top-left *Run* icon.

e) Close the SQL tab. In the dialog box, choose *Don't Save*.

f) In the *Database List* sub-pane, select *Column Views*.

g) Right-click the *HC::CVD_ADRESSES* column view and choose *Open Data*.

Hint:
Similarly to the Database List, entries in the *Catalog Browser* sub-pane can be filtered, using the *Apply Filter* (funnel) icon.

h) Close the SQL tab. In the dialog box, choose *Don't Save*.

Task 4: Grant Access to the HDI Container Schema to Your A#### User

Most of the modeling exercises in this learning journey will rely on the connection to the HDI Container, for example when previewing the data. However, a few exercises explore how a standard database user can query the data of an HDI Container.

To make this possible, your user *A####* must have the privilege to query data from the container. A generic role for enabling data access to your container's content has been created when you deployed the *HC300_A####* project for the first time.

The SQL statements to grant this role can be found in the file `SQL for External HDI Container Access.txt`. These statements must be executed in your SAP HANA Cloud database, Catalog View (NOT the HDI container), connected with user *A####*.

1. In the file explorer of your computer, open the script from your course files and copy its content to an SQL console connected to the *HC300_DB (A####) (SSO enabled)* connection.

```
SELECT * FROM ROLES WHERE ROLE_NAME LIKE '%HC300_A####%access_role';

CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','','HC300_A####_HDI_DB_1::acces
s_role', ?);
```

   a) In your course files, open the folder `SQL for External HDI Container Access.txt`.

   b) Select the entire content of the file (press **Ctrl+A**) and to copy the script, press **Ctrl+C**.

   c) Hover your mouse over the *HC300_DB (A####) (SSO enabled)* connection and, on the right, choose *Open SAP HANA SQL Console*.

   d) To paste the script, choose **Ctrl+V**.

2. Adapt the script so that it uses your assigned user number **A####**.

   a) To replace the string, press **Ctrl+H**.

b) In the fields *Search for* and *Replace with*, enter respectively **A####** and your assigned user number.

c) Next to the button *Replace*, choose *All*.

3. Execute the SQL statements to grant your course participant role some privileges on your container.

Note:
You have two options to execute a single SQL statement in the SQL console:

- Select the statement an choose *Run (F8)*.
- Place the cursor anywhere within the statement and choose *Run Statement (F9)*, which is one of the possible options in the drop-down list next to the *Run* button.

a) Check the actual name of your HC300_A#### container access role (SQL statement in section #4.3.a).

```
SELECT * FROM ROLES WHERE ROLE_NAME LIKE '%HC300_A####%access_role';
```

b) Check that the role name in the following SQL statement is consistent with the result of the previous query.

c) Execute the SQL statements in section #4.3.b.

```
CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','','HC300_A####_HDI_DB_1::acces
s_role', ?);
```

4. Check that the statement was executed successfully. The result should show a message saying *Operation successful!* together with the executed statement.

Unit 2
Exercise 4

Create a simple DIMENSION Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Create a simple DIMENSION calculation view.
- Deploy and Preview a calculation view.

Business Example

You are working as a modeler at a customer site where the Enterprise Procurement Model (EPM) is used.

You often need to report on products contained in the SNWD_PD table. You decide to create a CVD_PD calculation view that you will be able to use whenever products are needed.

Overview of Exercise Tasks

- Create a DIMENSION calculation view.
- Deploy the calculation view and preview the data.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a DIMENSION Calculation View

1. Create a new DIMENSION calculation view named **CVD_PD**.
2. Add the SNWD_PD table in the projection node.
3. Select the following columns as output :
   - CLIENT
   - NODE_KEY
   - PRODUCT_ID
   - CATEGORY
   - SUPPLIER_GUID

Task 2: Deploy the Calculation View and Preview Data

1. Deploy the new calculation view.
2. Open *SAP HANA Database Explorer* and preview the data.

Unit 2
Solution 4

Create a simple DIMENSION Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Create a simple DIMENSION calculation view.
- Deploy and Preview a calculation view.

Business Example

You are working as a modeler at a customer site where the Enterprise Procurement Model (EPM) is used.

You often need to report on products contained in the SNWD_PD table. You decide to create a CVD_PD calculation view that you will be able to use whenever products are needed.

Overview of Exercise Tasks

- Create a DIMENSION calculation view.
- Deploy the calculation view and preview the data.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a DIMENSION Calculation View

1. Create a new DIMENSION calculation view named **`CVD_PD`**.
   a) Select *View* → *Command Palette*.
   b) Enter **`arti`** in the command line field.
   c) Choose the **Create SAP HANA Database Artifact** command.
   d) Choose the *Browse for folder* icon and select the `/home/user/projects/HC300_A####/db/src/models/exercises/` folder. Choose *OK*.
   e) Complete the creation form with the information provided in the following table :

      Table 1: Create SAP HANA Database Artifact

Property	Value
Artifact type	**Calculation View**
Artifact name	**`CVD_PD`**
Data Category	**DIMENSION**

   f) Choose *Create*.

2. Add the SNWD_PD table in the projection node.
   a) In the open calculation view editor, select the *Projection* node and choose the *Add Data Source* icon.
   b) Enter **`snwd_pd`** in the *Search* field.
   c) Select the SNWD_PD table with synonym HC::SNWD_PD.
   d) Choose *Finish*.
3. Select the following columns as output :
   - CLIENT
   - NODE_KEY
   - PRODUCT_ID
   - CATEGORY
   - SUPPLIER_GUID

   a) Click on the *Expand Details Panel* icon on the top right.
   b) If not already displayed, select the *Mapping* tab.
   c) Drag and drop each column listed above to the *Output columns* pane.
   d) Close the calculation view tab.

Task 2: Deploy the Calculation View and Preview Data

1. Deploy the new calculation view.
   a) In the *SAP HANA PROJECTS* sub-pane, select the CVD_PD calculation view.
   b) Click on the *Deploy* icon.
2. Open *SAP HANA Database Explorer* and preview the data.
   a) Go to *SAP HANA Database Explorer*.
   b) Select *Columns Views*.
   c) Right click on the HC::CVD_PD view and choose *Open Data*.
   d) Close the SQL tab without saving.

Unit 2
Exercise 5

Create a simple CUBE Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Create a calculation view of type CUBE

Business Example

You are working as a modeler at a customer site where the Enterprise Procurement Model (EPM) is used.

You have been asked to model a calculation view to retrieve the gross and net amounts of sales orders by currency and sales order ID, and ordered by gross amounts (descending).

You will then create a calculation view of type CUBE that answers the design requirement.

Overview of Exercise Tasks

- Create a Simple CUBE Calculation View

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *Exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVC_SALES_SIMPLE`**
Label	**`Sales`**
Data Category	*CUBE*
With star join	[Deselected]

3. In the *Aggregation* node, add the SNWD_SO table from EPM_MODEL as data source:
4. Expand the *Details* panel.

Note:
From SAP HANA 2.0 SPS03 onwards, the *Details* panel is hidden when you create a new calculation view or open an existing one.

5. If necessary, maximize the size of the calculation view editor.
6. On the *Mapping* tab of the *Aggregation* node, add the following columns to the output.

- *CURRENCY_CODE*
- *SO_ID* (sales order ID)
- *GROSS_AMOUNT*
- *NET_AMOUNT*

7. What do you notice about the order of columns in the *Output Columns* area?

_____

_____

_____

8. Modify the order of the columns so that *CURRENCY_CODE* comes before *SO_ID*.
9. Order the result set as follows:
   - *CURRENCY_CODE* (ascending)
   - *GROSS_AMOUNT* (descending)

> Hint:
> This is done in the *Semantics* node. In case the ↑↓ *Sort Result Set* icon is not visible, use the ◦◦◦ *Toolbar Overflow* button.

10. Check that the relevant type (*Attribute* or *Measure*) is assigned to each column. Gross and Net amounts should be flagged as measures, and other columns as attributes.
11. Deploy the *CVC_SALES_SIMPLE* calculation view.
    Check the deployment status.
12. Preview the data of the *CVC_SALES_SIMPLE* calculation view.

Unit 2
Solution 5

Create a simple CUBE Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Create a calculation view of type CUBE

Business Example

You are working as a modeler at a customer site where the Enterprise Procurement Model (EPM) is used.

You have been asked to model a calculation view to retrieve the gross and net amounts of sales orders by currency and sales order ID, and ordered by gross amounts (descending).

You will then create a calculation view of type CUBE that answers the design requirement.

Overview of Exercise Tasks

- Create a Simple CUBE Calculation View

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.

2. In your *Exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVC_SALES_SIMPLE`**
Label	**`Sales`**
Data Category	*CUBE*
With star join	[Deselected]

   a) In the *EXPLORER* tree, right-click the folder *HC300_A#### → db → src → exercises* folder and choose *New File*.

   b) Enter **`CVC_SALES_SIMPLE.hdbcalculationview`** as file name and press *Enter*.

   c) Enter the calculation view label and other properties as specified in the table.

   d) Choose *Create*.

   Result

   The calculation view graphical editor opens on the right of your screen.

3. In the *Aggregation* node, add the SNWD_SO table from EPM_MODEL as data source:

a) Select the *Aggregation* node.

b) Choose **+**, *Add Data Source*.

c) In the *Find Data Sources* window, click the search field and enter **so**.

d) In the search results, select the *SNWD_SO* table with synonym *HC::SNWD_SO*.

e) Choose *Finish*.

4. Expand the *Details* panel.

> Note:
> From SAP HANA 2.0 SPS03 onwards, the *Details* panel is hidden when you create a new calculation view or open an existing one.

a) Choose the *Expand Details Panel* icon.

b) Alternatively, you can double-click the corresponding node in the *Scenario* pane.

5. If necessary, maximize the size of the calculation view editor.
   a) Choose the *Maximize Detail Panel* icon.
6. On the *Mapping* tab of the *Aggregation* node, add the following columns to the output.
   - *CURRENCY_CODE*
   - *SO_ID* (sales order ID)
   - *GROSS_AMOUNT*
   - *NET_AMOUNT*

   a) On the *Mapping* tab, select the columns as per the table above.

   b) Choose the *Add To Output* button.
7. What do you notice about the order of columns in the *Output Columns* area?

   The columns are not ordered as specified in the above list even if they were selected in this order.
8. Modify the order of the columns so that *CURRENCY_CODE* comes before *SO_ID*.
   a) On the *Columns* tab of the *Aggregation* node, select the *CURRENCY_CODE* column and choose *Move Up* (the up arrow).

   Alternatively, you can use the cut/paste buttons. This is especially useful when you need to move several columns at a time and/or move columns further up or down, to a non-adjacent location.
9. Order the result set as follows:
   - *CURRENCY_CODE* (ascending)
   - *GROSS_AMOUNT* (descending)

Hint:
This is done in the *Semantics* node. In case the ↑↓ *Sort Result Set* icon is not visible, use the ••• *Toolbar Overflow* button.

a) On the *Columns* tab of the *Semantics* node, choose ↑↓ *Sort Result Set*.

b) Choose **+** Add, and define the sort criteria for the *CURRENCY_CODE* column.

c) Repeat the previous step for the *GROSS_AMOUNT* column. Make sure you choose the sort direction *Descending*.

d) Choose *OK*.

10. Check that the relevant type (*Attribute* or *Measure*) is assigned to each column. Gross and Net amounts should be flagged as measures, and other columns as attributes.
    a) On the *Columns* tab of the *Semantics* node, check the *Type* property.
    b) If needed, to change the *Type* property for a column, select it and use the *Mark as Attribute* and *Mark as Measure* buttons.
11. Deploy the *CVC_SALES_SIMPLE* calculation view.
    Check the deployment status.
    a) Choose the Deploy icon for your *CVC_SALES_SIMPLE* calculation view.
    b) In the *Console* pane, check that the deployment completes successfully.
12. Preview the data of the *CVC_SALES_SIMPLE* calculation view.
    a) Open *SAP HANA Database Explorer*.
    b) Select *Columns Views*, or refresh the content of the catalog browser by choosing the *Refresh* icon on the right.
    c) right-click the *HC::CVC_SALES_SIMPLE* view and choose *Open Data*.
    d) Close the SQL tab.

Unit 2
Exercise 6

Define a Time Dimension Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Manage reference date/time data inside the HDB module of your project
- Create a DIMENSION calculation view of type TIME

Business Example

You need to analyze sales data for your company, with the possibility to easily expand/collapse data aggregates based on different level of date granularity. For this purpose, you want to explore the capabilities of TIME dimension calculation views.

Overview of Exercise Tasks

- Task 1: Generate Time Data in your HDI Container
- Task 2: Create a Dimension Calculation View of type TIME

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Generate Time Data in your HDI Container

1. Generate data for an existing time table with the following settings:

Setting	Value
Calendar Type	Gregorian
Period Unit	Time - Day
Generate Data After Creation	Checked
From Year	2018
To Year	2019
First Day of Week	Sunday

   Don't forget you first need to select the check-box *Time* before you can choose the radio button *Day*. Also, you must select the check-box *Generate Data After Creation*.

2. Examine the contents of the generated source file that represents the time table.
3. Check the table contents using the Database Explorer.

Task 2: Create a Dimension Calculation View of type TIME

1. In your *exercises* folder, create a calculation view of the type time dimension using the values provided below.

Setting	Value
Name	CVD_DATE_ATTRIBUTES
Label	Date attributes
Data Category	DIMENSION
Type	TIME
Calendar	Gregorian
Granularity	Date
Table	HC::M_TIME_DIMENSION
Auto Create	*selected*

2. Review the generated definition of the calculation view focusing on mapping, filter expression and the hierarchy definition.
3. Generate a root node for the hierarchy.
4. Deploy the calculation view.
5. Open *SAP HANA Database Explorer* and preview the data.
6. Close all remaining tabs.

Unit 2
Solution 6

Define a Time Dimension Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Manage reference date/time data inside the HDB module of your project
- Create a DIMENSION calculation view of type TIME

Business Example

You need to analyze sales data for your company, with the possibility to easily expand/collapse data aggregates based on different level of date granularity. For this purpose, you want to explore the capabilities of TIME dimension calculation views.

Overview of Exercise Tasks

- Task 1: Generate Time Data in your HDI Container
- Task 2: Create a Dimension Calculation View of type TIME

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Generate Time Data in your HDI Container

1. Generate data for an existing time table with the following settings:

Setting	Value
Calendar Type	Gregorian
Period Unit	Time - Day
Generate Data After Creation	Checked
From Year	2018
To Year	2019
First Day of Week	Sunday

   a) In your *HC300_A####* project, right-click on the folder *db* and choose *Maintain Time Tables*.

   b) In the *Generate Time Data* window, use the settings provided above and choose *Next* then *Finish*.

   Don't forget you first need to select the check-box *Time* before you can choose the radio button *Day*. Also, you must select the check-box *Generate Data After Creation*.

2. Examine the contents of the generated source file that represents the time table.

a) In your *HC300_A####* project, expand the generated folder *time_tables* and double-click on the file *M_TIME_DIMENSION.hdbtable*.

3. Check the table contents using the Database Explorer.

a) Open the Database Explorer and expand your container *HC300_A####_HDI_CONTAINER* and select *Tables*.

b) Right-click on the table *M_TIME_DIMENSION* and choose *Open Data* and you should see records covering the years 2018 and 2019. In particular, notice the granularity of the record is at the level of date and also notice the related attributes that are generated for each date, such as month, quarter, year.

Task 2: Create a Dimension Calculation View of type TIME

1. In your *exercises* folder, create a calculation view of the type time dimension using the values provided below.

Setting	Value
Name	CVD_DATE_ATTRIBUTES
Label	Date attributes
Data Category	DIMENSION
Type	TIME
Calendar	Gregorian
Granularity	Date
Table	HC::M_TIME_DIMENSION
Auto Create	*selected*

a) Right-click your *exercises* folder and choose *New File*.

b) Name the created file **`CVD_DATE_ATTRIBUTES.hdbcalculationview`** and press *Enter*.

c) Use the table above to complete the fields and press *Create*.

2. Review the generated definition of the calculation view focusing on mapping, filter expression and the hierarchy definition.

a) Select the *Projection_1* node.

b) In the *Projection_1* node, select the *Filter Expression* tab and review the filters.

c) Select the *Semantics* node.

d) In the *Hierarchies* tab, double-click the *Gregorian_Hierarchy* item.

3. Generate a root node for the hierarchy.

a) Expand the *Properties* pane.

b) In the *Root node Visibility* dropdown list, choose *Add Root node*.

4. Deploy the calculation view.

a) In the *SAP HANA PROJECTS* sub-pane, select the **CVD_DATE_ATTRIBUTES** calculation view.

b) Click on the *Deploy* icon.

5. Open *SAP HANA Database Explorer* and preview the data.

a) Go to *SAP HANA Database Explorer*.

b) Select *Columns Views*.

c) Right click on the **HC::CVD_DATE_ATTRIBUTES** view and choose *Open Data*.

d) Close the SQL tab without saving.

6. Close all remaining tabs.

Unit 3
Exercise 7

Control the behavior of aggregation node

Exercise Objectives

After completing this exercise, you will be able to:

- Control the behavior of the aggregation node with the keep flag option.

Business Example

You have a sales order table storing the sales for different products in different stores. You need to calculate the minimum quantity sold by product.

You will create a first calculation view to calculate the quantity of each product in each store, then create a second calculation view using the first one, to find the minimum quantity sold for each product.

Overview of Exercise Tasks

- Create a CUBE calculation view on the ORDERS table.
- Create a second CUBE calculation view to calculate the minimum quantity by product.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a CUBE calculation view on the ORDERS table

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVC_ORDERS`**
Label	**`Orders`**
Data Category	*CUBE*
With star join	[Deselected]

3. In the *Aggregation* node, add the ORDERS table from the TRAINING schema as data source:
4. On the *Mapping* tab of the *Aggregation* node, add the following columns to the output.
   - *PRODUCT* (sales order ID)
   - *STORE*

- *QUANTITY*

5. Check that the relevant type (*Attribute* or *Measure*) is assigned to each column. Quantity should be flagged as measure with SUM aggregation, and other columns as attributes.
6. Deploy the *CVC_ORDERS* calculation view.
   Check the deployment status.
7. Preview the data of the *CVC_ORDERS* calculation view and look for the **mouse** sales. Note the quantities.
8. Change the SELECT statement to query the Quantity by product. What is the quantity for the **mouse** product ?

Task 2: Create a second CUBE calculation view to calculate the minimum quantity by product

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *Exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVC_PROD_STATS`**
Label	**`Product statistics`**
Data Category	*CUBE*
With star join	[Deselected]

3. In the *Aggregation* node, add the CVC_ORDERS data source:
4. On the *Mapping* tab of the *Aggregation* node, add the following columns to the output.
   - *PRODUCT* (sales order ID)
   - *QUANTITY*
5. Check that the relevant type (*Attribute* or *Measure*) is assigned to each column. Quantity should be flagged as measure with MIN aggregation.
6. Deploy the *CVC_PROD_STATS* calculation view.
   Check the deployment status.
7. Preview the data of the *CVC_PROD_STATS* calculation view and look for the **mouse** sales. Note the quantity.

   Why is the quantity not the expected minimum quantity (2) ?

   _____

   _____

   _____
8. Go back to the *CVC_ORDERS* calculation view and select the *keep flag* option for the *store* column. Deploy the calculation view to apply modifications.

9. Preview again the data of the *CVC_PROD_STATS* calculation view and look for the **mouse** sales. Note the quantity.

   Why is the quantity now correct ?

   _____

   _____

   _____

Unit 3
Solution 7

Control the behavior of aggregation node

Exercise Objectives

After completing this exercise, you will be able to:

- Control the behavior of the aggregation node with the keep flag option.

Business Example

You have a sales order table storing the sales for different products in different stores. You need to calculate the minimum quantity sold by product.

You will create a first calculation view to calculate the quantity of each product in each store, then create a second calculation view using the first one, to find the minimum quantity sold for each product.

Overview of Exercise Tasks

- Create a CUBE calculation view on the ORDERS table.
- Create a second CUBE calculation view to calculate the minimum quantity by product.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a CUBE calculation view on the ORDERS table

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVC_ORDERS`**
Label	**`Orders`**
Data Category	*CUBE*
With star join	[Deselected]

   a) In the *EXPLORER* tree, right-click the folder *HC300_A#### → db → src → exercises* folder and choose *New File*.

   b) Enter **`CVC_ORDERS.hdbcalculationview`** as file name and press *Enter*.

   c) In the *New Calculation View* dialog box, enter the calculation view label and other properties as specified in the table.

   d) Choose *Create*.

Result

The calculation view graphical editor opens on the right of your screen.

3. In the *Aggregation* node, add the ORDERS table from the TRAINING schema as data source:
   a) Select the *Aggregation* node.
   b) Choose **+**, *Add Data Source*.
   c) In the *Find Data Sources* window, click the search field and enter **`ord`**.
   d) In the search results, select the table *ORDERS* with synonym *HC::ORDERS*.
   e) Choose *Finish*.
4. On the *Mapping* tab of the *Aggregation* node, add the following columns to the output.
   - *PRODUCT* (sales order ID)
   - *STORE*
   - *QUANTITY*

   a) Expand the *Details* panel for the *Aggregation* node.
   b) On the *Mapping* tab, select the columns as per the table above.
   c) Choose the *Add To Output* button.
5. Check that the relevant type (*Attribute* or *Measure*) is assigned to each column. Quantity should be flagged as measure with SUM aggregation, and other columns as attributes.
   a) On the *Columns* tab of the *Semantics* node, check the *Type* property.
   b) If needed, to change the *Type* property for a column, select it and use the *Mark as Attribute* and *Mark as Measure* buttons.
   c) Check that the aggregation function applied to the *QUANTITY* column is SUM.
6. Deploy the *CVC_ORDERS* calculation view.
   Check the deployment status.
   a) Choose the Deploy icon for your CVC_ORDERS calculation view..
   b) In the *Console* pane, check that the deployment completes successfully.
7. Preview the data of the *CVC_ORDERS* calculation view and look for the **mouse** sales. Note the quantities.
   a) Open *SAP HANA Database Explorer*.
   b) Select *Columns Views* , right-click the *HC::CVC_ORDERS* view and choose *Open Data*.
   c) Note the quantities for the mouse sales : 2 and 6.
8. Change the SELECT statement to query the Quantity by product. What is the quantity for the **mouse** product ?

a) Change the SELECT statement to the following :

```
SELECT "PRODUCT","QUANTITY"
FROM "HC300_A####_HDI_DB_1"."HC::CVC_ORDERS"
```

b) Choose *Run*.

c) Note the quantity for the **mouse** product : 8

d) Close all tabs.

Task 2: Create a second CUBE calculation view to calculate the minimum quantity by product

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *Exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVC_PROD_STATS`**
Label	**`Product statistics`**
Data Category	*CUBE*
With star join	[Deselected]

a) In the *EXPLORER* tree, right-click the folder *HC300_A#### → db → src → exercises* folder and choose *New File*.

b) Enter **`CVC_PROD_STATS.hdbcalculationview`** as file name and press *Enter*.

c) Enter the calculation view label and other properties as specified in the table.

d) Choose *Create*.

Result
The calculation view graphical editor opens on the right of your screen.

3. In the *Aggregation* node, add the CVC_ORDERS data source:
   a) Select the *Aggregation* node.
   b) Choose **+**, *Add Data Source*.
   c) In the *Find Data Sources* window, click the search field and enter **`cvc_o`**.
   d) In the search results, select the calculation view *HC::CVC_ORDERS* from your HDI Container.
   e) Choose *Finish*.
4. On the *Mapping* tab of the *Aggregation* node, add the following columns to the output.
   - *PRODUCT* (sales order ID)
   - *QUANTITY*

   a) Expand the *Details* panel for the *Aggregation* node.

b) On the *Mapping* tab, select the columns as per the table above.

c) Choose the *Add To Output* button.

5. Check that the relevant type (*Attribute* or *Measure*) is assigned to each column. Quantity should be flagged as measure with MIN aggregation.
   a) On the *Columns* tab of the *Semantics* node, check the *Type* property.
   b) If needed, to change the *Type* property for a column, select it and use the *Mark as Attribute* and *Mark as Measure* buttons.
   c) Change the aggregation function applied to the *QUANTITY* column to **MIN**.
6. Deploy the *CVC_PROD_STATS* calculation view.
   Check the deployment status.
   a) Choose the Deploy icon for your CVC_PROD_STATS calculation view..
   b) In the *Console* pane, check that the deployment completes successfully.
7. Preview the data of the *CVC_PROD_STATS* calculation view and look for the **mouse** sales. Note the quantity.

   Why is the quantity not the expected minimum quantity (2) ?

   The calculation view CVC_PROD_STATS actually asked the sum of quantity by product and then did a minimum on a single row.

   a) Open *SAP HANA Database Explorer*.
   b) Select *Columns Views* , right-click the HC::CVC_PROD_STATS view and choose *Open Data*.
   c) Note the quantity for the mouse sales : 8.
8. Go back to the *CVC_ORDERS* calculation view and select the *keep flag* option for the *store* column. Deploy the calculation view to apply modifications.
   a) Go back to the *EXPLORER* tree and select the CVC_ORDERS calculation view.
   b) Open the *details* panel for the *semantics* node.
   c) Choose the *Customize Column Display* button.
   d) Select the *Keep Flag* option to add it to the column displayed properties.
   e) Choose *OK*.
   f) Scroll right to display the *Keep Flag* property in the table and select it for the *STORE* column.
   g) Choose the Deploy icon for your CVC_ORDERS calculation view..
9. Preview again the data of the *CVC_PROD_STATS* calculation view and look for the **mouse** sales. Note the quantity.

Why is the quantity now correct ?

The calculation view CVC_PROD_STATS asked the quantity by product but the CVC_ORDERS calculation view actually calculated the quantity by product and store. Then the CVC_PROD_STATS did a minimum on the resulting rows by product.

a) Open *SAP HANA Database Explorer*.

b) Select *Columns Views* , right-click the HC::CVC_PROD_STATS view and choose *Open Data*.

c) Note the quantity for the mouse sales : 2.

Unit 4
Exercise 8

Create a DIMENSION Calculation View with joins

Exercise Objectives

After completing this exercise, you will be able to:

- Define a join into a DIMENSION calculation view.
- Join several tables to get all needed columns.

Business Example

You are at a customer site where the Enterprise Procurement Model (EPM) is used. Information about business partners, products, sales orders, and purchase orders is located in several tables, and you need to structure this data in a report.

You already created a DIMENSION calculation view for the products and now need to create a DIMENSION calculation views for the *Business Partner* dimension. This view will be used later on to create the star join calculation views for purchase orders.

The view will join several master data tables related to the business partners, including the contact persons and their e-mails.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVD_BP_JOIN`**
Label	**`Business Partners`**
Data Category	*DIMENSION*
Type	*STANDARD*

3. Add a *Join* node *Join_1* to the *Scenario* pane, and add the following tables from EPM_MODEL schema as data sources to the new node:
   - Table *SNWD_BP*: Business Partners master data
   - Table *SNWD_CONTACT*: Contact person details
4. If needed, expand the *Details* panel.

5. In the *Join Definition* tab for *Join_1*, join the data sources as follows:

   The table *SNWD_BP* is considered as the left table, and the table *SNWD_CONTACT* as the right table. If needed, move the two tables to make sure you have SNWD_BP on the left and do not define the join incorrectly.

   - Join the field *CLIENT* of the left table to the field *CLIENT* of the right table.
   - Join the field *NODE_KEY* of the left table to the field *PARTNER_GUID* of the right table.

   Use a *Referential* join type and a *1..1* cardinality:

> Hint:
> To check the cardinality, in the *Properties* pane, you can choose *Propose Cardinality*.

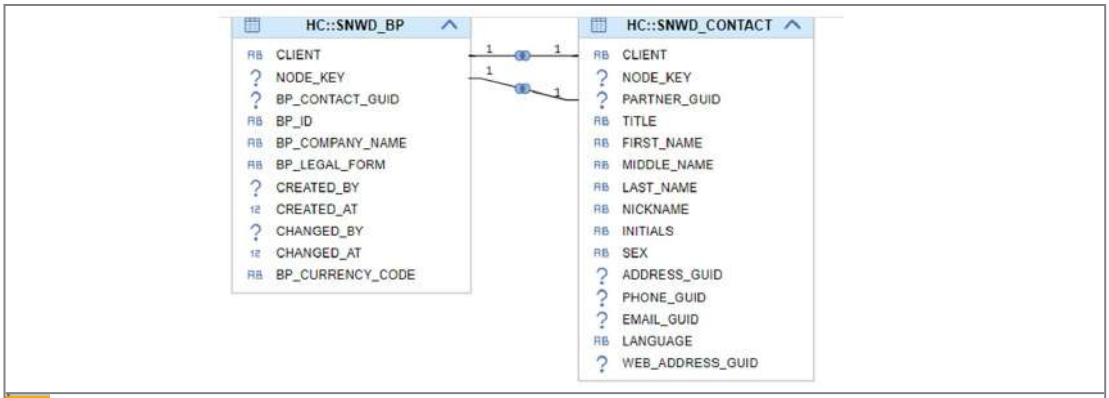


Figure 4: Creating Joins

6. In the *Mapping* tab for the node *Join_1*, add the following columns to the output:

Source table	Columns
SNWD_BP	• CLIENT • NODE_KEY • BP_ID • BP_COMPANY_NAME
SNWD_CONTACT	• FIRST_NAME • LAST_NAME • LANGUAGE

Hint:
You can add the columns from the two data sources to the output in a single operation, by selecting them all before clicking the *Add to Output* button.
Ensure that you do not collapse a data source in the tree because it removes the selection. Moreover, do not select an entire data source from the tree, because in this case all the columns will be added to the output.

7. Add another *Join* node *Join_2* to the *Scenario* pane, between the nodes *Join_1* and *Projection*, and add the two following data sources to this new node:
   - Node *Join_1*
   - Table *SNWD_BP_EM* (e-mail address) from EPM_MODEL

Hint:
To link a node to another one, select the source node and drag the arrow icon onto the target node. When the target node is reached, it gets a bold border, you can drop the arrow icon.

8. In the *Join Definition* tab, join the data sources of node *Join_2* as follows:
   The source *Join_1* is considered as the left table, and the table *SNWD_BP_EM* as the right table.
   - Join the field *CLIENT* of the left table to the field *CLIENT* of the right table.
   - Join the field *NODE_KEY* of the left table to the field *PARTNER_GUID* of the right table.

   Use a *Referential* join type and a *1..1* cardinality:

Note:
The join is defined on the same fields and with the same properties as the previous one.

9. On the *Mapping* tab for node *Join_2*, add the following columns to the output:

Source Table	Columns
Join_1	[All the columns]
SNWD_BP_EM	EMAIL_ADDRESS

Hint:
To add all the columns from *Join_1* in a single step, select the *Join_1* item from the *Data Sources* tree.

10. Define the *Join_2* node as the data source for the *Projection* node, and add all the columns of the *Projection* node to the output.
11. In the *Semantics* node, define the *BP_ID* column as a key column.

Hint:

If the *KEY* column is not displayed by default, click the *(Customize Column Display)* button.

12. Deploy the *CVD_BP_JOIN* calculation view.
    Check the deployment status.
13. Preview the data of the *CVD_BP_JOIN* calculation view.
14. Close all the open tabs in Business Application Studio.

Unit 4
Solution 8

Create a DIMENSION Calculation View with joins

Exercise Objectives

After completing this exercise, you will be able to:

- Define a join into a DIMENSION calculation view.
- Join several tables to get all needed columns.

Business Example

You are at a customer site where the Enterprise Procurement Model (EPM) is used. Information about business partners, products, sales orders, and purchase orders is located in several tables, and you need to structure this data in a report.

You already created a DIMENSION calculation view for the products and now need to create a DIMENSION calculation views for the *Business Partner* dimension. This view will be used later on to create the star join calculation views for purchase orders.

The view will join several master data tables related to the business partners, including the contact persons and their e-mails.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVD_BP_JOIN`**
Label	**`Business Partners`**
Data Category	*DIMENSION*
Type	*STANDARD*

   a) In the *EXPLORER* tree, right-click the folder *HC300_A####* → *db* → *src* → *exercises* folder and choose *New File*.

   b) Enter **`CVD_BP_JOIN.hdbcalculationview`** as file name and press *Enter*.

   c) Enter the calculation view label and other properties as specified in the table.

   d) Choose *Create*.

Result

The calculation view graphical editor opens on the right of your screen.

3. Add a *Join* node *Join_1* to the *Scenario* pane, and add the following tables from EPM_MODEL schema as data sources to the new node:

   - Table *SNWD_BP*: Business Partners master data
   - Table *SNWD_CONTACT*: Contact person details

   a) In the *Scenario* pane, add a Join node below the top nodes by selecting the *Create Join* node from the palette ( ) and clicking in the scenario pane below the projection node.

   b) Select the new *Join_1* node and click the + sign on the right of the node.

   c) In the *Find Data Sources* window, click the search field and enter **SNWD_**.

   d) In the search results, select the two tables *SNWD_BP* and *SNWD_CONTACT*.

   e) Choose *Finish*.

4. If needed, expand the *Details* panel.

   a) Choose the *Expand Details Panel* icon.

5. In the *Join Definition* tab for *Join_1*, join the data sources as follows:

   The table *SNWD_BP* is considered as the left table, and the table *SNWD_CONTACT* as the right table. If needed, move the two tables to make sure you have SNWD_BP on the left and do not define the join incorrectly.

   - Join the field *CLIENT* of the left table to the field *CLIENT* of the right table.
   - Join the field *NODE_KEY* of the left table to the field *PARTNER_GUID* of the right table.

   Use a *Referential* join type and a *1..1* cardinality:

> Hint:
> To check the cardinality, in the *Properties* pane, you can choose *Propose Cardinality*.

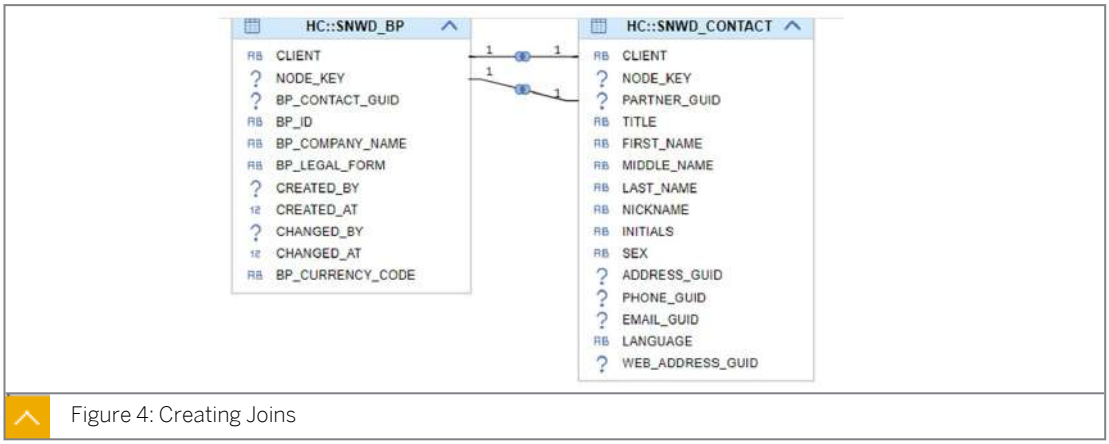


Figure 4: Creating Joins

a) In the *Join Definition* tab, if needed, drag and drop the table *SNWD_BP* so that it appears at the left of the *SNWD_CONTACTS* table.

b) Drag the *SNWD_BP.CLIENT* field to the *SNWD_CONTACT.CLIENT* field, and drag the *SNWD_BP.NODE_KEY* field to the *SNWD_CONTACT.PARTNER_GUID* field.

> Caution:
> For the second connector, make sure that you actually join the *PARTNER_GUID* of the right table (NOT the *NODE_KEY*).

c) To edit the join properties, select the new connector.

d) In the *PROPERTIES* pane, In the *Join Type* dropdown list, choose *Referential*.

e) In the *Cardinality* dropdown list, choose *1..1*.

f) Optionally, click the *Propose Cardinality* button.

> Note:
> The *Join Type* and *Cardinality* properties are defined for each join, in this case, the pair of connections between the tables.

6. In the *Mapping* tab for the node *Join_1*, add the following columns to the output:

Source table	Columns
SNWD_BP	• CLIENT • NODE_KEY • BP_ID • BP_COMPANY_NAME
SNWD_CONTACT	• FIRST_NAME • LAST_NAME • LANGUAGE

> Hint:
> You can add the columns from the two data sources to the output in a single operation, by selecting them all before clicking the *Add to Output* button. Ensure that you do not collapse a data source in the tree because it removes the selection. Moreover, do not select an entire data source from the tree, because in this case all the columns will be added to the output.

a) In the *Mapping* tab, select the columns as per the table above.

b) Choose the *Add To Output* button.

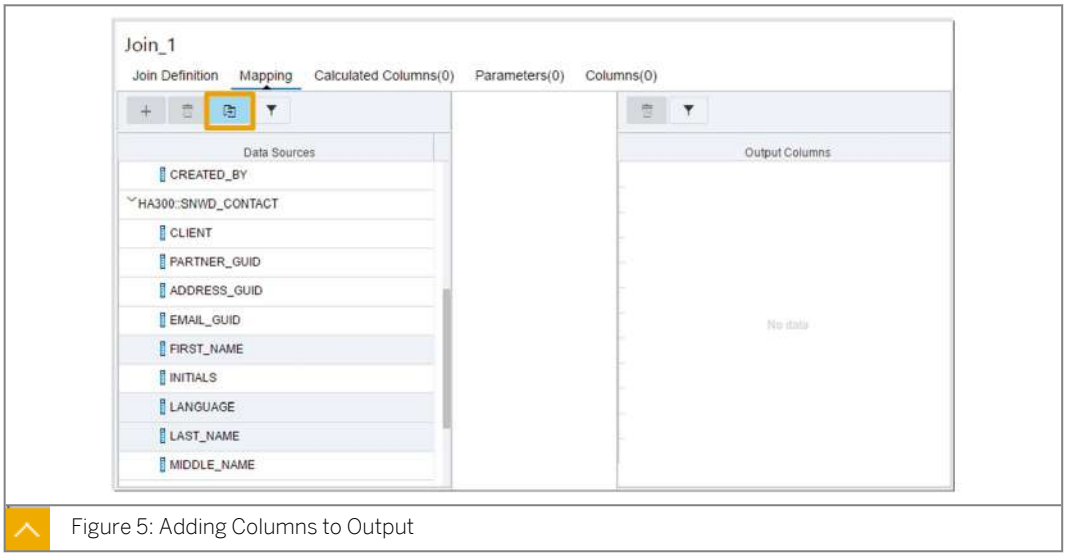


Figure 5: Adding Columns to Output

7. Add another *Join* node *Join_2* to the *Scenario* pane, between the nodes *Join_1* and *Projection*, and add the two following data sources to this new node:

   - Node *Join_1*
   - Table *SNWD_BP_EM* (e-mail address) from EPM_MODEL

> Hint:
> To link a node to another one, select the source node and drag the arrow icon onto the target node. When the target node is reached, it gets a bold border, you can drop the arrow icon.

   a) In the *Scenario* pane, add a new *Join* node *Join_2* below the top nodes by selecting the *Create Join* icon from the palette and clicking in the scenario pane between the *Join_1* and *Projection* nodes..

   b) Select the *Join_1* node.

   c) Drag the arrow icon to the *Join_2* node.

   d) Select the *Join_2* node.

   e) Choose the + icon on the right of the *Join_2* node.

   f) In the search field, enter **`SNWD_`**.

   g) Select the *SNWD_BP_EM* table.

   h) Choose *Finish*.

8. In the *Join Definition* tab, join the data sources of node *Join_2* as follows:

   The source *Join_1* is considered as the left table, and the table *SNWD_BP_EM* as the right table.

   - Join the field *CLIENT* of the left table to the field *CLIENT* of the right table.

- Join the field *NODE_KEY* of the left table to the field *PARTNER_GUID* of the right table.

Use a *Referential* join type and a *1..1* cardinality:

> Note:
> The join is defined on the same fields and with the same properties as the previous one.

a) On the *Join Definition* tab, drag the *Join_1.CLIENT* field to the *SNWD_BP_EM.CLIENT* field, and drag the *Join1.NODE_KEY* field to the *SNWD_BP_EM.PARTNER_GUID* field.

> Caution:
> For the second join, ensure that you actually join the *PARTNER_GUID* of the right table (NOT the *NODE_KEY*).

b) To edit the join properties, select the new connector.

c) In the *PROPERTIES* pane, in the *Join Type* dropdown list, choose *Referential*.

d) In the *Cardinality* dropdown list, choose *1..1*.

e) Choose *OK*.

9. On the *Mapping* tab for node *Join_2*, add the following columns to the output:

Source Table	Columns
Join_1	[All the columns]
SNWD_BP_EM	EMAIL_ADDRESS

> Hint:
> To add all the columns from *Join_1* in a single step, select the *Join_1* item from the *Data Sources* tree.

a) In the *Mapping* tab for *Join_2*, select the *Join_1* item and choose *Add to Output*.

You do not need to select any specific column from this data source, you need all columns.

b) Select the *SNWD_BP_EM → EMAIL_ADDRESS* field and choose *Add To Output*.

10. Define the *Join_2* node as the data source for the *Projection* node, and add all the columns of the *Projection* node to the output.

a) In the *Scenario* pane, select the *Join_2* node.

b) Drag the arrow icon to the *Projection* node.

c) In the *Mapping* tab of the *Projection* node, select the *Join_2* data source and choose *Add to Output*.

11. In the *Semantics* node, define the *BP_ID* column as a key column.

> Hint:
> If the *KEY* column is not displayed by default, click the *(Customize Column Display)* button.

a) In the *Scenario* pane, choose the *Semantics* node.

b) In the *Columns* tab, choose *(Customize Column Display)*, select the *Key* field and choose *OK*.

c) Select the *Key* checkbox of the *BP_ID* column.

12. Deploy the *CVD_BP_JOIN* calculation view.
Check the deployment status.

a) Choose the *Deploy* icon for your CVD_BP_JOIN calculation view..

b) In the *Console* pane, check that the deployment completes successfully.

13. Preview the data of the *CVD_BP_JOIN* calculation view.

a) Open *SAP HANA Database Explorer*.

b) Select *Column Views* , right-click the HC::CVD_BP_JOIN view and choose *Open Data*.

14. Close all the open tabs in Business Application Studio.

a) Right-click any open tab and choose *Close All*.

Unit 4
Exercise 9

Join Three Tables in a Single Join Node

Exercise Objectives

After completing this exercise, you will be able to:

- Combine more than two data sources in a single *Join* node
- Adjust *Join* node properties so that the joins are executed in the correct order to suit your scenario

Business Example

You are working as a modeler and would like to explore the possibility to define joins with more than two tables. However, you want to ensure that you can properly control the order in which several joins defined in the same node are executed. For this purpose, you will work on a sample data set and test different options.

Note:
When more than one two tables need to be joined at a given stage of the calculation scenario, it is of course still possible to use join nodes with only two joined tables and to stack these nodes. It might be relevant in some cases, especially when you want full control over the join order and/or a better legibility of the calculation scenario.

Overview of Exercise Tasks

- Task 1: Review the Data Sources Used in your Scenario
- Task 2: Import a Calculation View and Observe its Behavior
- Task 3: Modify the Multi-Join Order and Check the Impact on the Calculation View Output

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Review the Data Sources used in your Scenario

For the sake of simplicity, the sample data set is made up of three tables for Sales, Customers, and Countries. The three tables have names starting with *HC::MJ_* (for multi-join) and are already defined and populated with data (their design-time files are part of the HDB module that you built during exercise 1).

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the Database Explorer, open the three following tables to review their structure and content:

- HC::MJ_SALES
- HC::MJ_CUSTOMERS
- HC::MJ_COUNTRIES

3. Which columns from which tables would you suggest to use in a join definition to provide a consistent result?

Task 2: Import a Calculation View and Observe its Behavior

You will now import a calculation view including a single *Join* node, combining the three tables (Multi-Join).

1. Switch back to the *Explorer*.
2. In the workspace, import the *CVC_MJ_SALES.hdbcalculationview* file from your course files folder *HC300* → *Calculation Views Templates* into the folder *HC300_A####* → *db* → *src* → *exercises*.
3. Expand the *Details* panel and review the design of the *Join_1* node.

Hint:
To improve the screen layout, you can hide the workspace by clicking on the Explorer icon, and also use the *Maximize Details Panel* icon.

How many joins are defined? Which are the join types?

Which data source is defined as the *Central Table*?

How is the Multi-Join Order defined? Which join will be executed first?

4. Deploy the *CVC_MJ_SALES* calculation view.
   Check the deployment status.
5. Preview the calculation view data in the *Raw Data* tab.

How many rows are returned? Is it consistent with the specified Multi-Join Order?

Task 3: Modify the Multi-Join Order and Check the Impact on the Calculation View Output

Now, you want to modify the Multi-Join Order and check whether it affects the calculation view results.

1. Set the *Multi Join Order* property of the *Join_1* node to *Inside Out*.
2. Deploy the *CVC_MJ_SALES* calculation view again.
   Check the deployment status.

   Based on this change, how many rows do you expect in the calculation view output? Why?
3. Preview the calculation view data on the *Raw Data* tab.

   Does the output correspond to the expected behavior?
4. Close all open tabs in Business Application Studio.

Unit 4
Solution 9

Join Three Tables in a Single Join Node

Exercise Objectives

After completing this exercise, you will be able to:

- Combine more than two data sources in a single *Join* node
- Adjust *Join* node properties so that the joins are executed in the correct order to suit your scenario

Business Example

You are working as a modeler and would like to explore the possibility to define joins with more than two tables. However, you want to ensure that you can properly control the order in which several joins defined in the same node are executed. For this purpose, you will work on a sample data set and test different options.

Note:
When more than one two tables need to be joined at a given stage of the calculation scenario, it is of course still possible to use join nodes with only two joined tables and to stack these nodes. It might be relevant in some cases, especially when you want full control over the join order and/or a better legibility of the calculation scenario.

Overview of Exercise Tasks

- Task 1: Review the Data Sources Used in your Scenario
- Task 2: Import a Calculation View and Observe its Behavior
- Task 3: Modify the Multi-Join Order and Check the Impact on the Calculation View Output

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Review the Data Sources used in your Scenario

For the sake of simplicity, the sample data set is made up of three tables for Sales, Customers, and Countries. The three tables have names starting with *HC::MJ_* (for multi-join) and are already defined and populated with data (their design-time files are part of the HDB module that you built during exercise 1).

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the Database Explorer, open the three following tables to review their structure and content:

- HC::MJ_SALES
- HC::MJ_CUSTOMERS
- HC::MJ_COUNTRIES

a) Open *SAP HANA Database Explorer*.

b) Select *Tables*.

c) Select the *Apply Filter* icon and enter **MJ** in the search field then press *Enter*.

d) In the search results, right-click each table and choose *Open Data*.

3. Which columns from which tables would you suggest to use in a join definition to provide a consistent result?

   Based on the data structure, a relevant join between the three tables would be *MJ_SALES.CUSTOMER* to *MJ_CUSTOMERS.CUSTOMER_ID* and *MJ_CUSTOMERS.COUNTRY* to *MJ_COUNTRIES.COUNTRY_ID*.

Task 2: Import a Calculation View and Observe its Behavior

You will now import a calculation view including a single *Join* node, combining the three tables (Multi-Join).

1. Switch back to the *Explorer*.
2. In the workspace, import the *CVC_MJ_SALES.hdbcalculationview* file from your course files folder *HC300* → *Calculation Views Templates* into the folder *HC300_A####* → *db* → *src* → *exercises*.

   a) In the workspace, right-click your *exercises* folder and choose *upload*.

   b) Browse for the file *HC300* → *Calculation Views Templates* → *CVC_MJ_SALES.hdbcalculationview* and choose *Open*.

   c) Make sure that the calculation view ends up in the *.../exercises* folder.

   Result

   A new calculation view *CVC_MJ_SALES* is created in your project and opened in a new tab.

3. Expand the *Details* panel and review the design of the *Join_1* node.

Hint:
To improve the screen layout, you can hide the workspace by clicking on the Explorer icon, and also use the *Maximize Details Panel* icon.

a) Choose the *Expand Details Panel* icon.

Alternatively, you can double-click the *Join_1* node.

b) In the *Join_1* node, review the *Join Definition* and *Mapping* tabs.

c) To check a join definition, select the join connector in the *Join Definition* tab.

How many joins are defined? Which are the join types?

Two joins are defined in the *Join_1* node. The *MJ_SALES* to *MJ_CUSTOMERS* join is a *Left Outer* Join, and the *MJ_CUSTOMERS* to *MJ_COUNTRIES* join is an *Inner* Join.

Which data source is defined as the *Central Table*?

The *HC::MJ_SALES* table.

How is the Multi-Join Order defined? Which join will be executed first?

The Multi-Join Order is set to *Outside in*. This means that the *MJ_CUSTOMERS* to *MJ_COUNTRIES* join will be executed first.

4. Deploy the *CVC_MJ_SALES* calculation view.
   Check the deployment status.
   a) If needed, open the Explorer pane.
   b) Choose the Deploy icon for your CVC_MJ_SALES calculation view.
   c) In the *Console* pane, check that the deployment completes successfully.
5. Preview the calculation view data in the *Raw Data* tab.
   a) In the workspace, right-click the calculation view and choose *Data Preview*.
   b) Display the *Raw Data* tab.

   How many rows are returned? Is it consistent with the specified Multi-Join Order?

   Three rows are returned. This is consistent because the *MJ_SALES* to *MJ_CUSTOMERS* join was executed last, and it is a Left Outer Join. So all the records from the MJ_SALES table are returned, regardless of whether there are matching rows in the result of the join executed first (*MJ_CUSTOMERS* to *MJ_COUNTRIES*) or not.

Task 3: Modify the Multi-Join Order and Check the Impact on the Calculation View Output
Now, you want to modify the Multi-Join Order and check whether it affects the calculation view results.

1. Set the *Multi Join Order* property of the *Join_1* node to *Inside Out*.
   a) Display the *Join Definition* tab of the *Join_1* node and click anywhere in the white space.
   b) In the *PROPERTIES* pane, in the *Multi Join Order* drop-down list, select *Inside Out*.
2. Deploy the *CVC_MJ_SALES* calculation view again.
   Check the deployment status.
   a) Choose the Deploy icon for your CVC_MJ_SALES calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.

Based on this change, how many rows do you expect in the calculation view output? Why?

Only one row will be returned. Indeed, the *MJ_CUSTOMERS* to *MJ_COUNTRIES* join will be executed last. And because it is an Inner Join, only the sales orders with matching countries in the *MJ_COUNTRIES* table will be included. The only order that has a matching country is order #1 (customer C1, located in the US).

3. Preview the calculation view data on the *Raw Data* tab.
   a) In the workspace, right-click the calculation view and choose *Data Preview*.
   b) Display the *Raw Data* tab.

   Does the output correspond to the expected behavior?

   It does. Only one rows is returned, for order #1.

4. Close all open tabs in Business Application Studio.
   a) Right-click any open tab and choose *Close All*.

Unit 4
Exercise 10

Create a CUBE with Star Join Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Create a CUBE with Star Join calculation view.
- Reuse existing dimension calculation views.

Business Example

You need to report on purchase order data. To meet the business requirements, you have decided to create a CUBE with Star Join calculation view.

Note:
The fact table will be based on two sources from the SAP Enterprise Performance Management (EPM) model: the purchase order headers, and the purchase order Items.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Exercises to create the CVD_PD calculation view (Create a simple DIMENSION calculation view) and CVD_BP_JOIN calculation view (Create a DIMENSION calculation view using joins) should have been completed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVCS_PO`**
Label	**`Purchase Orders`**
Data Category	*CUBE*
With Star Join	[Selected]

3. Add a *Join* node *Join_1* to the *Scenario* pane, and add the following tables from EMP_MODEL as data sources to the new node:
   - Table *SNWD_PO*: Purchase order document headers
   - Table *SNWD_PO_I*: Purchase order line items

4. In the *Join Definition* tab, join the data sources of node *Join_1* as follows:
   The table *SNWD_PO* is considered as the left table, and the table *SNWD_PO_I* as the right table.
   - Join the field *CLIENT* of the left table to the field *CLIENT* of the right table.
   - Join the field *NODE_KEY* of the left table to the field *PARENT_KEY* of the right table.

   Use a Referential Join type and a *1..n* cardinality:

Hint:
To check the cardinality, in the *Properties* pane, you can choose *Propose Cardinality*.

5. In the *Mapping* tab for node *Join_1*, add the following columns to the output:

Source table	Columns
SNWD_PO	• CLIENT • PO_ID • PARTNER_GUID
SNWD_PO_I	• PO_ITEM_POS • PRODUCT_GUID • CURRENCY_CODE • GROSS_AMOUNT • TAX_AMOUNT • NET_AMOUNT

6. Define the *Join_1* node as the data source for the *Star Join* node.
7. Add the following dimension calculation views to the *Star Join* node:

Source Folder	Dimension Calculation View
exercises	CVD_BP_JOIN (Business partners)
exercises	CVD_PD (Products)

Note:
There are several columns with the same names (*CLIENT* or *NODE_KEY*) so alias names are automatically proposed to avoid duplicates. These aliases could be modified in the *Semantics* node if needed.

8. In the *Star Join* node, join the *Join_1* table with the two dimension calculation views as shown in the following table:

   Join_1 is considered as the left table, and the dimension calculation views as the right tables.

Left Table Column	Right Table Column
Join_1.CLIENT	CVD_BP_JOIN.CLIENT
Join_1.PARTNER_GUID	CVD_BP_JOIN.NODE_KEY
Join_1.CLIENT	CVD_PD.CLIENT
Join_1.PRODUCT_GUID	CVD_PD.NODE_KEY

   Use a Referential Join type and a *n..1* cardinality.

9. Select the following columns from the fact table to promote them to the semantics node :
   - PO_ID
   - PO_ITEM_POS
   - CURRENCY_CODE
   - GROSS_AMOUNT
   - TAX_AMOUNT
   - NET_AMOUNT
10. In the *Semantics* node, check that the relevant *Type* property is assigned to all the PRIVATE columns. The three columns with amounts are measures, all the others are attributes.
11. Hide the *CLIENT* and *NODE_KEY* columns (shared attributes) from the dimension calculation views *CVD_BP_JOIN* and *CVD_PD*. These columns are not needed for reporting.
12. Hiding columns does not prevent name conflict during deployment, that's why alias names are still needed. To be more specific in the naming, change the alias names to the *CLIENT* and *NODE_KEY* columns (shared columns), as follows:

Data Source	Column Name	Alias Name
CVD_BP_JOIN	CLIENT	**`CLIENT_BP`**
CVD_BP_JOIN	NODE_KEY	**`NODE_KEY_BP`**
CVD_PD	CLIENT	**`CLIENT_PD`**

Data Source	Column Name	Alias Name
CVD_PD	NODE_KEY	**NODE_KEY_PD**

13. Deploy the *CVCS_PO* calculation view.
    Check the deployment status.
14. Preview the data of the *CVCS_PO* calculation view. Display the gross and net amounts by company name and product category.
15. Close all the open tabs in Business Application Studio.

Unit 4
Solution 10

Create a CUBE with Star Join Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Create a CUBE with Star Join calculation view.
- Reuse existing dimension calculation views.

Business Example

You need to report on purchase order data. To meet the business requirements, you have decided to create a CUBE with Star Join calculation view.

Note:
The fact table will be based on two sources from the SAP Enterprise Performance Management (EPM) model: the purchase order headers, and the purchase order Items.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Exercises to create the CVD_PD calculation view (Create a simple DIMENSION calculation view) and CVD_BP_JOIN calculation view (Create a DIMENSION calculation view using joins) should have been completed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVCS_PO`**
Label	**`Purchase Orders`**
Data Category	*CUBE*
With Star Join	[Selected]

   a) In the *Workspace* tree, right-click the *HC300_A####* → *db* → *src* → *exercises* folder and choose *New File*.

   b) Enter **`CVCS_PO.hdbcalculationview`** as file name and press *Enter*.

   c) Enter the calculation view name and other properties as specified in the table.

d) Choose *Create*.

Result

The calculation view graphical editor opens on the right of your screen.

3. Add a *Join* node *Join_1* to the *Scenario* pane, and add the following tables from EMP_MODEL as data sources to the new node:

   - Table *SNWD_PO*: Purchase order document headers
   - Table *SNWD_PO_I*: Purchase order line items

   a) In the *Scenario* pane, add a *Join* node below the top node by selecting the *Create Join* icon from the palette and adding the node below the star join node.

   b) Select the new *Join_1* node and click the + sign on the right of the node.

   c) In the *Find Data Sources* window, click the search field and enter **`SNWD_PO`**.

   d) In the search results, select the two tables.

   e) Choose *Finish*.

4. In the *Join Definition* tab, join the data sources of node *Join_1* as follows:

   The table *SNWD_PO* is considered as the left table, and the table *SNWD_PO_I* as the right table.

   - Join the field *CLIENT* of the left table to the field *CLIENT* of the right table.
   - Join the field *NODE_KEY* of the left table to the field *PARENT_KEY* of the right table.

   Use a Referential Join type and a *1..n* cardinality:

   Hint:
   To check the cardinality, in the *Properties* pane, you can choose *Propose Cardinality*.

   a) In the *Details* pane, drag the *SNWD_PO.CLIENT* field to the *SNWD_PO_I.CLIENT* field, and then drag the *SNWD_PO.NODE_KEY* field to the *SNWD_PO_I.PARENT_KEY* field.

   Caution:
   For the second connector, make sure that you actually join the PARENT_KEY of the right table (NOT the NODE_KEY).

   b) To edit the join properties, select one of the new connectors.

   c) In the *Join Type* dropdown list, choose *Referential*.

   d) In the *Cardinality* dropdown list, choose *1..n*.

   e) Optionally, click the *Propose Cardinality* button.

5. In the *Mapping* tab for node *Join_1*, add the following columns to the output:

Source table	Columns
SNWD_PO	• CLIENT • PO_ID • PARTNER_GUID
SNWD_PO_I	• PO_ITEM_POS • PRODUCT_GUID • CURRENCY_CODE • GROSS_AMOUNT • TAX_AMOUNT • NET_AMOUNT

a) In the *Mapping* tab, select the columns as per the table above.

b) Choose the *Add To Output* button.

6. Define the *Join_1* node as the data source for the *Star Join* node.

a) In the *Scenario* pane, select the *Join_1* node.

b) Drag the arrow icon to the *Star Join* node.

7. Add the following dimension calculation views to the *Star Join* node:

Source Folder	Dimension Calculation View
exercises	CVD_BP_JOIN (Business partners)
exercises	CVD_PD (Products)

Note:
There are several columns with the same names (*CLIENT* or *NODE_KEY*) so alias names are automatically proposed to avoid duplicates. These aliases could be modified in the *Semantics* node if needed.

a) In the *Scenario* pane, choose the **+** sign on the right of the *Star Join* node.

b) In the search field, enter **CVD**.

c) Select the two dimension views mentioned in the table above.

d) Choose *Finish*.

e) In the *Alias Proposal* dialog box, confirm by choosing *OK*.

8. In the *Star Join* node, join the *Join_1* table with the two dimension calculation views as shown in the following table:

   Join_1 is considered as the left table, and the dimension calculation views as the right tables.

Left Table Column	Right Table Column
Join_1.CLIENT	CVD_BP_JOIN.CLIENT
Join_1.PARTNER_GUID	CVD_BP_JOIN.NODE_KEY
Join_1.CLIENT	CVD_PD.CLIENT
Join_1.PRODUCT_GUID	CVD_PD.NODE_KEY

   Use a Referential Join type and a *n..1* cardinality.

   a) In the *Join Definition* tab of the *Star Join* node, drag the field *CLIENT* from the *Join_1* data source to the field *CLIENT* of the calculation view *CVD_BP_JOIN*.

   b) Drag the field *PARTNER_GUID* from the *Join_1* data source to the field *NODE_KEY* of the calculation view *CVD_BP_JOIN*.

   c) In the *Join Type* dropdown list, choose *Referential*.

   d) In the *Cardinality* dropdown list, choose *n..1*.

   e) Drag the field *CLIENT* from the *Join_1* data source to the field *CLIENT* of the calculation view *CVD_PD*.

   f) Drag the field *PRODUCT_GUID* from the *Join_1* data source to the field *NODE_KEY* of the calculation view *CVD_PD*.

   g) In the *Join Type* dropdown list, choose *Referential*.

   h) In the *Cardinality* dropdown list, choose *n..1*.

9. Select the following columns from the fact table to promote them to the semantics node :

   - PO_ID
   - PO_ITEM_POS
   - CURRENCY_CODE
   - GROSS_AMOUNT
   - TAX_AMOUNT
   - NET_AMOUNT

   a) In the *Mapping* tab, select the columns as per the list above.

   b) Choose the *Add To Output* button.

10. In the *Semantics* node, check that the relevant *Type* property is assigned to all the PRIVATE columns. The three columns with amounts are measures, all the others are attributes.

a) In the *Semantics* node, display the *Columns* tab.

b) Check the *Type* property of each private column.

11. Hide the *CLIENT* and *NODE_KEY* columns (shared attributes) from the dimension calculation views *CVD_BP_JOIN* and *CVD_PD*. These columns are not needed for reporting.

a) Select the *Columns* tab of the *Semantics* node.

b) In the *Shared* columns list, select the *Hidden* checkbox of the two *CLIENT* columns and the two *NODE_KEY* columns.

12. Hiding columns does not prevent name conflict during deployment, that's why alias names are still needed. To be more specific in the naming, change the alias names to the *CLIENT* and *NODE_KEY* columns (shared columns), as follows:

Data Source	Column Name	Alias Name
CVD_BP_JOIN	CLIENT	**CLIENT_BP**
CVD_BP_JOIN	NODE_KEY	**NODE_KEY_BP**
CVD_PD	CLIENT	**CLIENT_PD**
CVD_PD	NODE_KEY	**NODE_KEY_PD**

a) In the *Shared* columns list, update the *Alias Name* property as per the table above.

13. Deploy the *CVCS_PO* calculation view.

Check the deployment status.

a) Choose the Deploy icon for your CVCS_PO calculation view..

b) In the *Console* pane, check that the deployment completes successfully.

14. Preview the data of the *CVCS_PO* calculation view. Display the gross and net amounts by company name and product category.

a) In the workspace, right-click the *CVCS_PO* calculation view and choose *Data Preview*.

b) In the *Analysis* tab, drag the *GROSS_AMOUNT* and *NET_AMOUNT* columns to the *Value Axis* area.

c) Drag the *BP_COMPANY_NAME* and *CATEGORY* columns to the *Label Axis* area.

d) Change to Table Display to have a clearer view of the result.

15. Close all the open tabs in Business Application Studio.

a) Right-click any open tab and choose *Close All*.

Unit 5
Exercise 11

Combine Two Data Sources with a Union Node

Exercise Objectives

After completing this exercise, you will be able to:

- Use *Union* nodes in calculation views
- Use constant value mapping in Union nodes.

Business Example

A company wants to analyze its products data, which is available in SAP HANA but in two different tables because it is extracted from two different ERP systems: one from SAP and the other from a third party vendor.

You decided to use a calculation view for this new report, where the *Union* node allows you to combine the data from the two data tables together.

To enable the union, you must harmonize the data types between the tables.

Overview of Exercise Tasks

- Task 1: Review the Data Sources used in your Scenario
- Task 2: Create a Calculation View to select the required data
- Task 3: Add a Union node to combine data sources
- Task 4: Setup the Union to define a type code value for the third party data

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Review the Data Sources used in your scenario

The products data come from two different sources. The first data set is stored in the SNWD_PD table. The second data set is stored in two tables : PRODUCT and PRODUCT_GROUP. You need to examine the data so you know how to combine them.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the Database Explorer, open the three following synonyms to review their structure and content:
   - HC::SNWD_PD
   - HC::PRODUCT
   - HC::PRODUCT_GROUP
3. Determine the correct columns to map the two data sources.

Which column can be mapped with the PRODUCT_ID column of the SNWD_PD table ?

Which column can be mapped with the CATEGORY column of the SNWD_PD table ?

Which column can be mapped with the CURRENCY_CODE column of the SNWD_PD table ?

Which column can be mapped with the PRICE column of the SNWD_PD table ?

Is there a column that can be mapped with the TYPE_CODE column of the SNWD_PD table ?

Task 2: Create a Calculation View to select the required data

1. Create a new calculation view using the following details:

Property	Value
Name	*CVC_PROD_UNION*
Label	**`Consolidation of products from two Source Systems`**
Data Category	*CUBE*
With Star Join	[Deselected]

2. Add a *Projection* node to the *Scenario* pane, below the *aggregation* node and add the *SNWD_PD* table to the new node. Rename the node **`SAP_products`** and select the following columns :
   - PRODUCT_ID
   - TYPE_CODE

- CATEGORY
- CURRENCY_CODE
- PRICE

3. Add a *Join* node to the *Scenario* pane, and add the *PRODUCT* and *PRODUCT_GROUP* tables to the new node. Rename it **`Third_party_products`**.
4. In the *Join Definition* tab for *Third_party_products*, join the data sources by joining the two *PRODUCT_GROUP* fields.

   Use a *Inner* join type and a *n..1* cardinality:

> Hint:
> To check the cardinality, in the *Properties* pane, you can choose *Propose Cardinality*.

5. In the *Mapping* tab for the node *Third_party_products*, add the following columns to the output:

Source table	Columns
PRODUCT	• PRODUCT_ID • BASIS_PRICE • CURRENCY
PRODUCT_GROUP	PRODUCT_GROUP_TEXT

> Hint:
> You can add the columns from the two data sources to the output in a single operation, by selecting them all before clicking the *Add to Output* button.
> Ensure that you do not collapse a data source in the tree because it removes the selection. Moreover, do not select an entire data source from the tree, because in this case all the columns will be added to the output.

Task 3: Add a Union node to combine data sources

1. Add a *Union* node to the *Scenario* pane, below the *aggregation* node, connect the *SAP_products* and the *Third_party_products* nodes to the *Union_1* node.
2. Add all the columns from *SAP_products* to the output of the union node :
3. Map the columns from *Third_party_products* to the existing output columns.
4. Connect the *Union_1* node to the *Aggregation* node and select all columns for output.
5. Deploy the *CVC_PROD_UNION* calculation view.

   Check the deployment status.

6. Preview the calculation view data and display the *CATEGORY* distinct values. Then add the *TYPE_CODE* column.

   What do you observe ? Why ?

   _____

   _____

   _____

Task 4: Setup the Union to define a type code value for the third party data

Depending on the requirements, you might want to assign a specific *TYPE_CODE* value for third party products, to know they come from the third party data source, or on the contrary to assign them the same type code as the SAP Products. Let's assume we are in the second scenario. You want to assign a TYPE_CODE of PR to all products.

1. Select the *Union_1* node and assign the value **`PR`** to the *TYPE_CODE* column for rows coming from the *Third_party_products* data source.
2. Deploy the *CVC_PROD_UNION* calculation view again.

   Check the deployment status.
3. Preview the calculation view data and display the *CATEGORY* distinct values. Then add the *TYPE_CODE* column.

   What do you observe ?

   _____

   _____

   _____
4. Display the average price by category and currency code.
5. Close all the open tabs in Business Application Studio.

Unit 5
Solution 11

Combine Two Data Sources with a Union Node

Exercise Objectives

After completing this exercise, you will be able to:

- Use *Union* nodes in calculation views
- Use constant value mapping in Union nodes.

Business Example

A company wants to analyze its products data, which is available in SAP HANA but in two different tables because it is extracted from two different ERP systems: one from SAP and the other from a third party vendor.

You decided to use a calculation view for this new report, where the *Union* node allows you to combine the data from the two data tables together.

To enable the union, you must harmonize the data types between the tables.

Overview of Exercise Tasks

- Task 1: Review the Data Sources used in your Scenario
- Task 2: Create a Calculation View to select the required data
- Task 3: Add a Union node to combine data sources
- Task 4: Setup the Union to define a type code value for the third party data

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Review the Data Sources used in your scenario

The products data come from two different sources. The first data set is stored in the SNWD_PD table. The second data set is stored in two tables : PRODUCT and PRODUCT_GROUP. You need to examine the data so you know how to combine them.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the Database Explorer, open the three following synonyms to review their structure and content:
   - HC::SNWD_PD
   - HC::PRODUCT
   - HC::PRODUCT_GROUP

   a) Open *SAP HANA Database Explorer*.

b) Select *Synonyms*.

c) Select the *Apply Filter* icon and enter **`SNWD_PD`** in the search field then press *Enter*.

d) In the search results, right-click the table and choose *Open Data*.

e) Select again the *Apply Filter* icon and enter **`PRODUCT`** in the search field then press *Enter*.

f) In the search results, right-click each table and choose *Open Data*.

3. Determine the correct columns to map the two data sources.

Which column can be mapped with the PRODUCT_ID column of the SNWD_PD table ?

The PRODUCT_ID column from the PRODUCT table.

Which column can be mapped with the CATEGORY column of the SNWD_PD table ?

The PRODUCT_GROUP_TEXT column of the PRODUCT_GROUP table.

Which column can be mapped with the CURRENCY_CODE column of the SNWD_PD table ?

The CURRENCY column of the PRODUCT table.

Which column can be mapped with the PRICE column of the SNWD_PD table ?

The BASIS_PRICE column of the PRODUCT table.

Is there a column that can be mapped with the TYPE_CODE column of the SNWD_PD table ?

No.

Task 2: Create a Calculation View to select the required data

1. Create a new calculation view using the following details:

Property	Value
Name	*CVC_PROD_UNION*
Label	**`Consolidation of products from two Source Systems`**
Data Category	*CUBE*
With Star Join	[Deselected]

a) In the *EXPLORER* tree, right-click the folder *HC300_A#### → db → src → exercises* folder and choose *New File*.

b) Enter **`CVC_PROD_UNION.hdbcalculationview`** as file name and press *Enter*.

c) Enter the calculation view label and other properties as specified in the table.

d) Choose *Create*.

Result

The calculation view graphical editor opens on the right of your screen.

2. Add a *Projection* node to the *Scenario* pane, below the *aggregation* node and add the *SNWD_PD* table to the new node. Rename the node **`SAP_products`** and select the following columns :

- PRODUCT_ID
- TYPE_CODE
- CATEGORY
- CURRENCY_CODE
- PRICE

a) In the *Scenario* pane, add a Projection node by selecting the *Create Projection* node from the palette ( ) and clicking in the scenario pane below the *aggregation* node.

b) Select the new *Projection_1* node with the right mouse button and choose *rename*.

c) Enter **`SAP_products`** and press *Enter*.

d) Select the now *SAP_products* node and click the + sign on the right of the node.

e) In the *Find Data Sources* window, click the search field and enter **`SNWD_PD`**.

f) In the search results, select the table and choose *Finish*.

g) In the *Mapping* tab, select the columns as per the list above.

h) Choose the *Add To Output* button.

3. Add a *Join* node to the *Scenario* pane, and add the *PRODUCT* and *PRODUCT_GROUP* tables to the new node. Rename it **`Third_party_products`**.

a) In the *Scenario* pane, add a Join node below the top nodes by selecting the *Create Join* node from the palette ( ) and clicking in the scenario pane below the aggregation node.

b) Select the new *Join_1* node with the right mouse button and choose *rename*.

c) Enter **`Third_party_products`** and press *Enter*.

d) Select the now *Third_party_products* node and click the + sign on the right of the node.

e) In the *Find Data Sources* window, click the search field and enter **`PRODUCT`**.

f) In the search results, select the two tables and choose *Finish*.

4. In the *Join Definition* tab for *Third_party_products*, join the data sources by joining the two *PRODUCT_GROUP* fields.

   Use a *Inner* join type and a *n..1* cardinality:

   > Hint:
   > To check the cardinality, in the *Properties* pane, you can choose *Propose Cardinality*.

   a) Drag the *PRODUCT.PRODUCT_GROUP* field to the *PRODUCT_GROUP.PRODUCT_GROUP* field.

   b) To edit the join properties, select the new connector.

   c) In the *PROPERTIES* pane, In the *Join Type* dropdown list, choose *Inner*.

   d) In the *Cardinality* dropdown list, choose *n..1*.

   e) Optionally, click the *Propose Cardinality* button.

5. In the *Mapping* tab for the node *Third_party_products*, add the following columns to the output:

Source table	Columns
PRODUCT	• PRODUCT_ID • BASIS_PRICE • CURRENCY
PRODUCT_GROUP	PRODUCT_GROUP_TEXT

   > Hint:
   > You can add the columns from the two data sources to the output in a single operation, by selecting them all before clicking the *Add to Output* button.
   > Ensure that you do not collapse a data source in the tree because it removes the selection. Moreover, do not select an entire data source from the tree, because in this case all the columns will be added to the output.

   a) In the *Mapping* tab, select the columns as per the table above.

   b) Choose the *Add To Output* button.

Task 3: Add a Union node to combine data sources

1. Add a *Union* node to the *Scenario* pane, below the *aggregation* node, connect the *SAP_products* and the *Third_party_products* nodes to the *Union_1* node.

   a) In the *Scenario* pane, add a Union node by selecting the *Create Union* node from the palette ( ) and clicking in the scenario pane below the *aggregation* node.

b) Connect the *SAP_products* node to the *Union_1* node to add it as the first data source.

c) Connect the *Third_party_products* node to the *Union_1* node to add it as the second data source.

2. Add all the columns from *SAP_products* to the output of the union node :

a) In the *Mapping* tab, select the *SAP_products* node.

b) Choose the *Add To Output* button.

3. Map the columns from *Third_party_products* to the existing output columns.

a) Select each of the following columns from the *Third_party_products* node and drag and drop them to the corresponding output columns :

Table 2: Union mapping

Columns from Third_party_products	Corresponding columns in output
PRODUCT_ID	PRODUCT_ID
BASIS_PRICE	PRICE
CURRENCY	CURRENCY_CODE
PRODUCT_GROUP_TEXT	CATEGORY

4. Connect the *Union_1* node to the *Aggregation* node and select all columns for output.

a) Select the *Union_1* node and drag the arrow icon to the *Aggregation* node.

b) In the *Mapping* tab, select the *Union_1* node.

c) Choose the *Add To Output* button.

5. Deploy the *CVC_PROD_UNION* calculation view.
Check the deployment status.

a) Choose the *Deploy* icon for your *CVC_PROD_UNION* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

6. Preview the calculation view data and display the *CATEGORY* distinct values. Then add the *TYPE_CODE* column.

What do you observe ? Why ?

For some categories there are two results, one with a type code of PR and one with no type code. This is due to the fact that products from the third party data source do not have any type code value.

a) In the workspace, right-click the calculation view and choose *Data Preview*.

b) In the *Analysis* pane, drag the *CATEGORY* column to the *Label Axis* area.

c) Then drag the *TYPE_CODE* column to the *Label Axis* area.

Task 4: Setup the Union to define a type code value for the third party data

Depending on the requirements, you might want to assign a specific *TYPE_CODE* value for third party products, to know they come from the third party data source, or on the contrary to assign them the same type code as the SAP Products. Let's assume we are in the second scenario. You want to assign a TYPE_CODE of PR to all products.

1. Select the *Union_1* node and assign the value **PR** to the *TYPE_CODE* column for rows coming from the *Third_party_products* data source.
   a) Select the *Union_1* node and open the *Details* pane.
   b) In the *Mapping* tab, select the *TYPE_CODE* column in the *Output columns* pane with the right mouse button and choose *Manage Mappings*.
   c) In the *Manage Mapping* dialog, in the *Constant Value* field for the *Third_party_products* data source, enter **PR** then choose *OK*.
2. Deploy the *CVC_PROD_UNION* calculation view again.
   Check the deployment status.
   a) Choose the *Deploy* icon for your *CVC_PROD_UNION* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
3. Preview the calculation view data and display the *CATEGORY* distinct values. Then add the *TYPE_CODE* column.

   What do you observe ?

   Now all products have a type code.

   a) In the workspace, right-click the calculation view and choose *Data Preview*.
   b) In the *Analysis* pane, drag the *CATEGORY* column to the *Label Axis* area.
   c) Then drag the *TYPE_CODE* column to the *Label Axis* area.
4. Display the average price by category and currency code.
   a) Add the CURRENCY_CODE column to the Label Axis area.
   b) Add the PRICE column to the Value Axis area.
   c) In the drop down list on the left of the *PRICE(SUM)* field, select **Avg**.
5. Close all the open tabs in Business Application Studio.
   a) Right-click any open tab and choose *Close All*.

Unit 6
Exercise 12

Use a Minus Node in a Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Use a *Minus* node to return the difference between two data sets.

Business Example

Your company wants to improve its sales of software products, and wants to target existing customers who have already bought Notebooks, but no software yet, with a dedicated campaign.

You need to create a calculation view that will return the corresponding data, based on two CUBE calculation views (*HC::CVC_NB_SO* for Notebooks sales and *HC::CVC_SW_SO* for Software sales). Those views list sales orders items in the corresponding category, with details about the customers (Business Partners), such as the country.

Note:
The analysis should be possible both at the Country level and at the Business Partner level.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation **`CVSQL_MINUS`**. The calculation view type should be *DIMENSION* and the description **`Customers – NB minus SW`**.
3. Add a *Minus* node below the top *Projection* node and connect it to this *Projection* node.
4. Add the two data sources to the *Minus_1* node.
5. Map the columns in the minus node.
   Make sure you start with the *NB* node. If not, you can use *Switch Order* in the context menu of the *Minus* node.

   Does the *CATEGORY* column need to be mapped?

6. Add the two required columns (*BP_COMPANY_NAME* and *COUNTRY*) to the top *Projection* node.
7. Deploy the *CVSQL_MINUS* calculation view

   Check the deployment status.
8. Preview the data of the calculation view.

   How many customers are retrieved? Where are they located?

   _____

   _____

   _____
9. Edit the default SQL query to request data at the *COUNTRY* level only.

   The modified statement should look as follows:

```
SELECT TOP 1000
    "COUNTRY"
FROM "HC300_A####_HDI_DB_1"."HC::CVSQL_MINUS";
```

   Which countries are returned? What does it mean?

   _____

   _____

   _____

   What about the other countries?

   _____

   _____

   _____
10. Close all open tabs in Business Application Studio.

Unit 6
Solution 12

Use a Minus Node in a Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Use a *Minus* node to return the difference between two data sets.

Business Example

Your company wants to improve its sales of software products, and wants to target existing customers who have already bought Notebooks, but no software yet, with a dedicated campaign.

You need to create a calculation view that will return the corresponding data, based on two CUBE calculation views (*HC::CVC_NB_SO* for Notebooks sales and *HC::CVC_SW_SO* for Software sales). Those views list sales orders items in the corresponding category, with details about the customers (Business Partners), such as the country.

Note:
The analysis should be possible both at the Country level and at the Business Partner level.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation **`CVSQL_MINUS`**. The calculation view type should be *DIMENSION* and the description **`Customers – NB minus SW`**.
   a) Right-click the *exercises* folder and choose *New File*.
   b) Enter **`CVSQL_MINUS.hdbcalculationview`** as file name and press *Enter*.
   c) Enter the calculation view label and change the *Data Category* to DIMENSION.
   d) Choose *Create*.
   Result
   The calculation view graphical editor opens on the right of your screen.
3. Add a *Minus* node below the top *Projection* node and connect it to this *Projection* node.
   a) In the palette, choose (*Create Minus*) and drop the node below the top *Projection* node.

Result

The node is called *Minus_1*.

b) Select the *Minus_1* node and drag the arrow icon at the right to the *Projection* node.

4. Add the two data sources to the *Minus_1* node.

a) Select the *Minus_1* node and click the + sign on the right of the node.

b) In the *Find Data Sources* window, click the search field and enter **NB_SO**.

c) In the search results, select the view and choose *Finish*.

d) Repeat the steps with **SW_SO** to add the second data source.

5. Map the columns in the minus node.

Make sure you start with the *NB* node. If not, you can use *Switch Order* in the context menu of the *Minus* node.

Does the *CATEGORY* column need to be mapped?

No. In this context, mapping (and querying) the *CATEGORY* column would return the entire set of customers who have bought Notebooks.

a) Select the *Minus_1* node and open the *Details* pane.

b) In the *Mapping* tab, choose  *Auto Map by Name*.

You might need to click the » *Toolbar Overflow* button first.

c) In the *Output Columns* area, right-click the *CATEGORY* column and choose *Remove Output Column*.

6. Add the two required columns (*BP_COMPANY_NAME* and *COUNTRY*) to the top *Projection* node.

a) In the *Mapping* tab of the *Projection* node, select the *Minus_1* data source.

b) Select the *BP_COMPANY_NAME* and the *COUNTRY* columns.

c) Choose *Add To Output*.

7. Deploy the *CVSQL_MINUS* calculation view

Check the deployment status.

a) Choose the *Deploy* icon for your *CVSQL_MINUS* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

8. Preview the data of the calculation view.

a) In the explorer, right-click the *CVSQL_MINUS* calculation view and choose *Data Preview.*.

b) Select the *Raw Data* tab.

How many customers are retrieved? Where are they located?

Four customers have bought notebooks but no software. They are located in DE, GB, FR, and BR (this information will be used later on).

9. Edit the default SQL query to request data at the *COUNTRY* level only.

   The modified statement should look as follows:

```
SELECT TOP 1000
    "COUNTRY"
FROM "HC300_A####_HDI_DB_1"."HC::CVSQL_MINUS";
```

   a) Select the SQL *(Edit SQL Statement in SQL Console)* icon.

   b) Remove `"BP_COMPANY_NAME",` from the SELECT clause of the SQL statement.

   c) Choose *Run*.

   Which countries are returned? What does it mean?

   GB and BR are returned. So in each of these countries, we have sold notebooks but no software at all.

   What about the other countries?

   DE and FR are not returned. Combined with the outcome of the initial data preview, this means that we have sold notebooks AND software in each of these countries.

10. Close all open tabs in Business Application Studio.

    a) Right-click any open tab and choose *Close All*.

Unit 7
Exercise 13

Use Rank Nodes to Partition, Order, and Extract Values from a Data Set

Exercise Objectives

After completing this exercise, you will be able to:

- Extract the top n values from a data set by using a Rank node
- Understand the impact of the client tool query on the way data is retrieved
- Define the rank node so that it adapts dynamically to the selected columns

Business Example

Your company wants to analyze its sales orders by extracting the best-selling products in each country.

You will first create a new calculation view with a rank node that extracts the top 5 products for each country, based on a provided calculation view that retrieves the sales order details from the source tables.

You will then explore additional capabilities of the Rank node.

Overview of Exercise Tasks

- Task 1: Create a New Calculation View Using a Rank Node
- Task 2: Modify the Calculation View to Better Control Rank Behavior
- Task 3: Use the Result Set Type Percentage
- Task 4: Use the Sum Aggregation Function

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a New Calculation View Using a Rank Node

Create a calculation view that will process a top 5 extraction on a provided sales order item view.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Preview the data of the calculation view *CVCS_SO* that will be used as a data source for the rank node. It is located in your *resources* folder.

   This calculation view returns information on sales order details (each sales order has several line items; one for each product ID that is part of the order). The dimension Calculation views are used to retrieve, among other things, the Customer Name (BP_COMPANY_NAME) and its country (COUNTRY), as well as the Product ID (PRODUCT_ID) and product category (CATEGORY).

Do you see several rows for a single SO_ID (Sales Order ID)? Can you explain why?

3. Create a new calculation view of the type *Cube*, with the following properties:

Field	Value
Name	**`CVC_SO_RANK`**
Description	**`Sales Orders Rank`**
Data Category	*CUBE*
With Star Join	[Deselected]

4. Add a new Rank node *Rank_1* to the calculation view scenario, and define it as the data source for the Aggregation node.

5. Define the calculation view *resources* → *CVCS_SO* as the data source for the *Rank_1* node, and add the following columns to the output:

   - COUNTRY
   - PRODUCT_ID
   - GROSS_AMOUNT

6. Define the settings of the Rank node so that it extracts, for each country, the five products that total the biggest sales amount (column: *GROSS_AMOUNT*).

   A rank column, named *RANK*, should be generated.

Property	Value
Aggregation	*Row*
Result Set Direction	*Top*
Result Set Type	*Absolute*
Target Value	*Fixed* Value: **`5`**
Offset	*Fixed* Value: [Leave Blank]
Generate Rank Column	[Selected]
Generated rank column name	**`RANK`** (do not keep the default column name)
Partition column	*COUNTRY*
Sort Column	*GROSS_AMOUNT* (direction: *Descending)*

7. In the Aggregation node, add all columns to the output.

8. Deploy the *CVC_SO_RANK* calculation view.
   Check the deployment status after activation.
9. Preview the data of the *CVC_SO_RANK* calculation view.

> Caution:
> Because the *Rank_1* node is using a CUBE calculation view as a data source, that CUBE calculation view feeds the rank node with a data set on which it applies the defined aggregation. This aggregation is based on the list of attribute columns requested by the query executed on top of the rank node. In the current scenario, the *CSCS_SO* view has executed an aggregation by *COUNTRY* and *PRODUCT_ID*.

How is the result data set ordered?

_____

_____

_____

10. Filter the data preview to *US* as the *COUNTRY* and check that you receive the expected result, which is shown in the following table:

COUNTRY	PRODUCT_ID	RANK	GROSS_AMOUNT
US	HT-1037	1	51 352 888.96
US	HT-1021	2	8 026 133.60
US	HT-1116	3	4 848 000.00
US	HT-1106	4	2 441 955.76
US	HT-1502	5	1 243 077.12

,

Result

> Note:
> The following intermediate solution object is available at this stage:
>
> *solutions → CVC_SO_RANK_00_STAGE1*

11. Close the preview tab.

Task 2: Modify the Calculation View to Better Control Rank Behavior

You will now modify the properties of the Rank node and output columns of the calculation view *CVC_SO_RANK*.

1. Add the column *BP_COMPANY_NAME* to the definition of the *Rank_1* node and up to the *Semantics* node.
2. Deploy the *CVC_SO_RANK* calculation view.
   Check the deployment status after activation.

3. Preview the data of the *CVC_SO_RANK* calculation view.

   What do you observe?

4. Modify the default SQL query of the data preview to remove the *BP_COMPANY_NAME* column. You must also remove this column from the `GROUP BY` clause. Then, execute the modified query and filter the result set on COUNTRY = 'US'.

   The modified SQL query should look as follows:

```
SELECT TOP 1000
    "COUNTRY",
    "PRODUCT_ID",
    "RANK",
    SUM("GROSS_AMOUNT") AS "GROSS_AMOUNT"
FROM ...
GROUP BY "COUNTRY", "PRODUCT_ID", "RANK";
```

   What do you observe?

   Result

   Note:
   The following intermediate solution object is available at this stage:

   solutions → CVC_SO_RANK_00_STAGE2.

5. Close the preview tabs.

6. Modify the *CVCS_SO_RANK* calculation view, adding the *BP_COMPANY_NAME* column to the *Logical Partition* of the Rank node. Then, deploy the calculation view.

   Result

   Note:
   The following intermediate solution object is available at this stage:

   solutions → CVC_SO_RANK_00_STAGE3.

7. Preview the data of the *CVC_SO_RANK* calculation view.

   How does the result set look?

8. Modify the default SQL query of the data preview to remove the *BP_COMPANY_NAME* column. You must also remove this column from the `GROUP BY` clause. Then execute the modified query and filter the result set on COUNTRY = 'US'.

   The modified SQL query should look as follows:

```
SELECT TOP 1000
    "COUNTRY",
    "PRODUCT_ID",
    "RANK",
    SUM("GROSS_AMOUNT") AS "GROSS_AMOUNT"
FROM ...
GROUP BY "COUNTRY", "PRODUCT_ID", "RANK";
```

   What do you observe?

9. Close the preview tabs.

10. Modify the view *CVC_SO_RANK* to activate the *Dynamic Partition Elements* in the properties of the *Rank_1* node. Deploy the calculation view.

    Result

> Note:
> The following intermediate solution object is available at this stage:
>
> *solutions → CVC_SO_RANK_00_STAGE4*.

11. Preview the data of the *CVC_SO_RANK* calculation view. You get the same result, partitioned by *COUNTRY* and *BP_COMPANY_NAME*.

12. Modify the default SQL query of the data preview to remove the *BP_COMPANY_NAME* column. You must also remove this column from the `GROUP BY` clause. Then execute the modified query and filter the result set on COUNTRY = 'US'.

    The modified SQL query should look as follows:

```
SELECT TOP 1000
    "COUNTRY",
    "PRODUCT_ID",
    "RANK",
    SUM("GROSS_AMOUNT") AS "GROSS_AMOUNT"
FROM ...
GROUP BY "COUNTRY", "PRODUCT_ID", "RANK";
```

    What do you observe?

13. Close all the open tabs in Business Application Studio.

Task 3: Use the Result Set Type Percentage

Now, you want to retrieve the top 10% best-selling products by country. You will adjust the calculation view accordingly.

1. Open the *CVCS_SO_RANK* calculation view.
2. In the Rank node definition, remove the column *BP_COMPANY_NAME* from the output columns. Confirm that it has automatically been removed from the list of columns used to partition the data set.
3. Adjust the Definition of the Rank node as follows (the table only mentions the properties you need to modify):

Property	Value
Result Set Type	*Percentage*
Target Value	*Fixed* Value: `0.1`

4. Deploy the *CVC_SO_RANK* calculation view.
   Check the deployment status after activation.

   Result

   Note:
   The following intermediate solution object is available at this stage:
   solutions → *CVC_SO_RANK_00_STAGE5*.

5. Preview the intermediate result set of the *Rank_1* node.

   What do you observe regarding the RANK column?

   _____

   _____

   _____

6. Filter the result set on COUNTRY = 'CN'.

   How many rows are returned?

   _____

   _____

   _____

   What is the max percentage in the RANK column?

   _____

   _____

   _____

Can you guess how many different products were sold to China?

7. Refresh the preview of the *CVCS_SO_RANK* calculation view.

   What do you notice regarding the result set order?

   What would you suggest to keep the result set ordered logically?

8. Close the preview tabs.
9. In the definition of the *CVCS_SO_RANK* calculation view, sort the result set of the calculation view by COUNTRY (asc) and RANK (asc).
10. Deploy the *CVC_SO_RANK* calculation view.
    Check the build status after activation.

    Result

> Note:
> The following intermediate solution object is available at this stage:
> *solutions → CVC_SO_RANK_00_STAGE6*.

11. Preview the *CVCS_SO_RANK* calculation view and confirm that the result set is now logically ordered.
12. Close all open tabs except the *Calculation View editor* tab for the *CSC_SO_RANK* calculation view.

Task 4: Use the Sum Aggregation Function

Now, you would like to find the products that sell worst in each country, but this time you want the number of products in the result set for each country to be determined as a percentage of the total sales in this country.

More specifically, you want to extract and rank, in each country, the products that generate the lowest revenue and represent 1% of the total sales (GROSS_AMOUNT) in the country.

1. In the *CVC_SO_RANK* calculation view, adjust the Definition of the Rank node as follows (the table only mentions the properties you need to modify):

Property	Value
Aggregation Function	*Sum*

Property	Value
Target Value	*Fixed* Value: `0.01`
Sort Column	*GROSS_AMOUNT* (direction: *Ascending)*

Note:
Keep the *Result Set Type* setting *Percentage*.

2. Deploy the *CVC_SO_RANK* calculation view.
   Check the deployment status after activation.

   Result

Note:
The following intermediate solution object is available at this stage:

solutions → CVC_SO_RANK_00_STAGE7

3. Preview the Calculation View. Check that the result subset for COUNTRY = 'CA' is consistent.

   How many rows do you get for Canada?

4. Close the data preview tab.
5. (Optional) Consider another way to achieve a similar result.

   Can you think of another approach that should work without modifying the sort order for the *GROSS_AMOUNT* column?

6. (Optional) Adjust the Definition of the Rank node as follows (the table only mentions the properties you need to modify).

Property	Value
Aggregation Function	*Sum*
Result Set Direction	*Down*
Sort Column	*GROSS_AMOUNT* (direction: *Descending)*

7. (Optional) Deploy the *CVC_SO_RANK* calculation view.

Check the deployment status after activation.

8. (Optional) Preview the Calculation View. Compare the results for COUNTRY = 'CA' with what you got before.

   Do you get the same 14 rows for Canada?

   _____

   _____

   _____

   Can you explain why?

   _____

   _____

   _____

Unit 7
Solution 13

Use Rank Nodes to Partition, Order, and Extract Values from a Data Set

Exercise Objectives

After completing this exercise, you will be able to:

- Extract the top n values from a data set by using a Rank node
- Understand the impact of the client tool query on the way data is retrieved
- Define the rank node so that it adapts dynamically to the selected columns

Business Example

Your company wants to analyze its sales orders by extracting the best-selling products in each country.

You will first create a new calculation view with a rank node that extracts the top 5 products for each country, based on a provided calculation view that retrieves the sales order details from the source tables.

You will then explore additional capabilities of the Rank node.

Overview of Exercise Tasks

- Task 1: Create a New Calculation View Using a Rank Node
- Task 2: Modify the Calculation View to Better Control Rank Behavior
- Task 3: Use the Result Set Type Percentage
- Task 4: Use the Sum Aggregation Function

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a New Calculation View Using a Rank Node

Create a calculation view that will process a top 5 extraction on a provided sales order item view.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Preview the data of the calculation view *CVCS_SO* that will be used as a data source for the rank node. It is located in your *resources* folder.
   a) In the explorer, expand your *HC300_A####* project to display the folder *db* → *src* → *resources*.
   b) Select then right-click the calculation view *CVCS_SO* and choose *Data Preview.*.

c) Select the arrow button on the right of the SO_ID column. Then choose *Sort ascending*.

This calculation view returns information on sales order details (each sales order has several line items; one for each product ID that is part of the order). The dimension Calculation views are used to retrieve, among other things, the Customer Name (BP_COMPANY_NAME) and its country (COUNTRY), as well as the Product ID (PRODUCT_ID) and product category (CATEGORY).

Do you see several rows for a single SO_ID (Sales Order ID)? Can you explain why?

Yes. This is because the source Calculation View provides sales order details by product.

3. Create a new calculation view of the type *Cube*, with the following properties:

Field	Value
Name	**CVC_SO_RANK**
Description	**Sales Orders Rank**
Data Category	*CUBE*
With Star Join	[Deselected]

a) In the *Explorer* tree, choose *HC300_A#### → db → src* and right-click the *exercises* folder. Choose *New File*.

b) Enter **CVC_SO_RANK.hdbcalculationview** as file name and press *Enter*.

c) Enter the calculation view properties as per the table above.

d) Choose *Create*.

Result

The calculation view graphical editor opens on the right of your screen.

4. Add a new Rank node *Rank_1* to the calculation view scenario, and define it as the data source for the Aggregation node.

a) In the *Scenario* pane, select the Rank node from the palette and drop it to the area below the Aggregation node.

Result

The new node is automatically named *Rank_1*.

b) Select the *Rank_1* node and drag the arrow icon to the Aggregation node.

5. Define the calculation view *resources → CVCS_SO* as the data source for the *Rank_1* node, and add the following columns to the output:

- COUNTRY
- PRODUCT_ID
- GROSS_AMOUNT

a) Select the new *Rank_1* node and choose + on the right of the node.

b) In the *Find Data Sources* window, select the *Search* field and enter **CVCS_SO**.

c) In the search results, select the view *HC::CVCS_SO* and choose *Finish*.

d) In the *Mapping* tab, select the columns as per the table.

e) Choose *Add To Output*.

6. Define the settings of the Rank node so that it extracts, for each country, the five products that total the biggest sales amount (column: *GROSS_AMOUNT*).

   A rank column, named *RANK*, should be generated.

Property	Value
Aggregation	*Row*
Result Set Direction	*Top*
Result Set Type	*Absolute*
Target Value	*Fixed* Value: **5**
Offset	*Fixed* Value: [Leave Blank]
Generate Rank Column	[Selected]
Generated rank column name	**RANK** (do not keep the default column name)
Partition column	*COUNTRY*
Sort Column	*GROSS_AMOUNT* (direction: *Descending*)

   a) Select the *Rank_1* node.

   b) In the *Definition* tab, define the rank properties using the values from the table.

      To add a column to the *Logical Partition*, choose +. Do the same to define the *Sort Column*.

7. In the Aggregation node, add all columns to the output.

   a) Select the Aggregation node.

   b) In the *Mapping* tab, select the *Rank_1* item and choose *Add To Output*.

      Alternatively, you can drag the *Rank_1* item from the *Data Sources* area to the *Output Columns* area.

8. Deploy the *CVC_SO_RANK* calculation view.

   Check the deployment status after activation.

   a) Choose the *Deploy* icon for your *CVC_SO_RANK* calculation view.

   b) In the *Console* pane, check that the deployment completes successfully.

9. Preview the data of the *CVC_SO_RANK* calculation view.

   a) Right-click the *CVC_SO_RANK* calculation view and choose *Data Preview*.

   b) Check the data displayed in the *Raw Data* tab.

> Caution:
> Because the *Rank_1* node is using a CUBE calculation view as a data source, that CUBE calculation view feeds the rank node with a data set on which it applies the defined aggregation. This aggregation is based on the list of attribute columns requested by the query executed on top of the rank node. In the current scenario, the *CSCS_SO* view has executed an aggregation by *COUNTRY* and *PRODUCT_ID*.

How is the result data set ordered?

The result set is ordered first by *COUNTRY* (ascending) — the partition column – and then by *GROSS_AMOUNT* (descending), which corresponds to *RANK* (ascending).

10. Filter the data preview to *US* as the *COUNTRY* and check that you receive the expected result, which is shown in the following table:

COUNTRY	PRODUCT_ID	RANK	GROSS_AMOUNT
US	HT-1037	1	51 352 888.96
US	HT-1021	2	8 026 133.60
US	HT-1116	3	4 848 000.00
US	HT-1106	4	2 441 955.76
US	HT-1502	5	1 243 077.12

,

a) Choose the arrow at the right of the *COUNTRY* column header

b) In the *Filter* field, enter **US** and press Enter.

> Note:
> This way of filtering does not re-execute the query against the calculation view, but just filters the current result set. It is valid in the current scenario because the complete result set has 80 rows, which is less than the maximum number of rows (1,000) defined by default in the settings of SAP Business Application Studio (*SAP HANA Database Explorer* section).
>
> In other scenarios, you might need to apply a filter to the data preview query itself, by choosing *Add Filter*. The query will then be re-executed against the calculation view.

Result

> Note:
> The following intermediate solution object is available at this stage:
>
> *solutions → CVC_SO_RANK_00_STAGE1*

11. Close the preview tab.

Task 2: Modify the Calculation View to Better Control Rank Behavior

You will now modify the properties of the Rank node and output columns of the calculation view *CVC_SO_RANK*.

1. Add the column *BP_COMPANY_NAME* to the definition of the *Rank_1* node and up to the *Semantics* node.
   a) On the *Mapping* tab of the *Rank_1* node, select the *BP_COMPANY_NAME* column and choose *Add To Output*.

      Alternatively, you can drag the column from the *Data Sources* pane to the *Output Columns* pane.

   b) In the *Output Columns* pane, right-click the column *BP_COMPANY_NAME* and choose *Propagate to Semantics*.

   c) In the confirmation dialog box, choose *OK*.

2. Deploy the *CVC_SO_RANK* calculation view.

   Check the deployment status after activation.

   a) Choose the *Deploy* icon for your *CVC_SO_RANK* calculation view.

   b) In the *Console* pane, check that the deployment completes successfully.

3. Preview the data of the *CVC_SO_RANK* calculation view.
   a) Right-click the *CVC_SO_RANK* calculation view and choose *Data Preview*.

   b) Display the *Raw Data* tab.

   What do you observe?

   There are still five top values generated for each country, but this time based on the total sales amount by customer/product pair. Compared with the scenario tested in Task 1, the CVCS_SO calculation view has aggregated the data set by COUNTRY, PRODUCT_ID and COMPANY_NAME. Then the Rank node has partitioned by COUNTRY, and ordered the data set by GROSS_AMOUNT (desc).

4. Modify the default SQL query of the data preview to remove the *BP_COMPANY_NAME* column. You must also remove this column from the `GROUP BY` clause. Then, execute the modified query and filter the result set on COUNTRY = 'US'.

   The modified SQL query should look as follows:

```
SELECT TOP 1000
    "COUNTRY",
    "PRODUCT_ID",
    "RANK",
    SUM("GROSS_AMOUNT") AS "GROSS_AMOUNT"
FROM ...
GROUP BY "COUNTRY", "PRODUCT_ID", "RANK";
```

   a) In the *Data Preview* tab, choose SQL *Edit Sql Statement In Sql Console*.

   b) Remove the column `"COMPANY_NAME",` both from the SELECT section and the GROUP BY section of the statement.

Do not forget to also remove the commas following the column name.

c) Choose  *Run*.

d) Choose the arrow at the right of the *COUNTRY* column header.

e) In the *Filter* field, enter **US** and press Enter.

What do you observe?

By removing the *BP_COMPANY_NAME* column from the query executed on top of the *CVCS_SO_RANK* Calculation View, the underlying **CVCS_SO** Calculation View has returned a data set without this column, so aggregated by **COUNTRY** and **PRODUCT_ID** only.

Result

Note:
The following intermediate solution object is available at this stage:

solutions → CVC_SO_RANK_00_STAGE2.

5. Close the preview tabs.

6. Modify the *CVCS_SO_RANK* calculation view, adding the *BP_COMPANY_NAME* column to the *Logical Partition* of the Rank node. Then, deploy the calculation view.

   a) In the *Scenario* pane, select the *Rank_1* node.

   b) On the *Definition* tab, under the *Partition Column* area, choose + and select the *BP_COMPANY_NAME* column with the drop-down list.

   c) Choose the *Deploy* icon for your *CVC_SO_RANK* calculation view.

   d) In the *Console* pane, check that the deployment completes successfully.

   Result

Note:
The following intermediate solution object is available at this stage:

solutions → CVC_SO_RANK_00_STAGE3.

7. Preview the data of the *CVC_SO_RANK* calculation view.

   a) Right-click the *CVC_SO_RANK* calculation view and choose *Data Preview*.

   b) Display the *Raw Data* tab.

   How does the result set look?

   The rank node has selected the Top 5 gross amounts for each country and each company.

8. Modify the default SQL query of the data preview to remove the *BP_COMPANY_NAME* column. You must also remove this column from the `GROUP BY` clause. Then execute the modified query and filter the result set on COUNTRY = 'US'.

   The modified SQL query should look as follows:

```
SELECT TOP 1000
    "COUNTRY",
    "PRODUCT_ID",
    "RANK",
    SUM("GROSS_AMOUNT") AS "GROSS_AMOUNT"
FROM ...
GROUP BY "COUNTRY", "PRODUCT_ID", "RANK";
```

   a) In the *Data Preview* tab, choose SQL *Edit Sql Statement In Sql Console*.

   b) Remove the column `"COMPANY_NAME",` both from the SELECT section and the GROUP BY section of the statement.

   Do not forget to also remove the commas following the column name.

   c) Choose *Run*.

   d) Choose the arrow at the right of the *COUNTRY* column header.

   e) In the *Filter* field, enter **US** and press Enter.

   What do you observe?

   The results are not consistent, because the column *BP_COMPANY_NAME* is included in the *Logical Partition* list but is not part of the top query applied to the view.

9. Close the preview tabs.

10. Modify the view *CVC_SO_RANK* to activate the *Dynamic Partition Elements* in the properties of the *Rank_1* node. Deploy the calculation view.

    a) In the *Scenario* pane, select the *Rank_1* node.

    b) In the *Definition* tab, select the *Dynamic Partition Elements* checkbox.

    c) Choose the *Deploy* icon for your *CVC_SO_RANK* calculation view.

    d) In the *Console* pane, check that the deployment completes successfully.

    Result

    > Note:
    > The following intermediate solution object is available at this stage:
    >
    > *solutions → CVC_SO_RANK_00_STAGE4*.

11. Preview the data of the *CVC_SO_RANK* calculation view. You get the same result, partitioned by *COUNTRY* and *BP_COMPANY_NAME*.

    a) Right-click the *CVC_SO_RANK* calculation view and choose *Data Preview*.

    b) Display the *Raw Data* tab.

12. Modify the default SQL query of the data preview to remove the *BP_COMPANY_NAME* column. You must also remove this column from the `GROUP BY` clause. Then execute the modified query and filter the result set on COUNTRY = 'US'.

    The modified SQL query should look as follows:

```
SELECT TOP 1000
    "COUNTRY",
    "PRODUCT_ID",
    "RANK",
    SUM("GROSS_AMOUNT") AS "GROSS_AMOUNT"
FROM ...
GROUP BY "COUNTRY", "PRODUCT_ID", "RANK";
```

    a) In the *Data Preview* tab, choose SQL *Edit Sql Statement In Sql Console*.

    b) Remove the column `"COMPANY_NAME",`, both from the SELECT section and the GROUP BY section of the statement.

    Do not forget to also remove the commas following the column name.

    c) Choose *Run*.

    d) Choose the arrow at the right of the *COUNTRY* column header.

    e) In the *Filter* field, enter **US** and press Enter.

    What do you observe?

    With the *Dynamic Partition Elements* option, a column selected for the *Logical Partition* will be automatically ignored when it is not included in the top SQL query.

13. Close all the open tabs in Business Application Studio.

    a) Right-click any open tab and choose *Close All*.

Task 3: Use the Result Set Type Percentage

Now, you want to retrieve the top 10% best-selling products by country. You will adjust the calculation view accordingly.

1. Open the *CVCS_SO_RANK* calculation view.

    a) In the *Explorer* view, select the *src* → *exercises* → *CVCS_SO_RANK* calculation view.

2. In the Rank node definition, remove the column *BP_COMPANY_NAME* from the output columns. Confirm that it has automatically been removed from the list of columns used to partition the data set.

    a) Double-click the *Rank_1* node and display the *Mapping* tab.

    b) In the *Output Columns* area, select the *BP_COMPANY_NAME* column and choose *Remove Output Column*.

    c) In the confirmation dialog box, read the warning and choose *Yes*.

    d) In the *Definition* tab, confirm that the *Logical Partition* only contains the *COUNTRY* column.

3. Adjust the Definition of the Rank node as follows (the table only mentions the properties you need to modify):

Property	Value
Result Set Type	*Percentage*
Target Value	*Fixed* Value: `0.1`

a) Select the *Rank_1* node.

b) In the *Definition* tab, adjust the properties as per the table.

4. Deploy the *CVC_SO_RANK* calculation view.
Check the deployment status after activation.

a) Choose the *Deploy* icon for your *CVC_SO_RANK* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

Result

Note:
The following intermediate solution object is available at this stage:
solutions → CVC_SO_RANK_00_STAGE5.

5. Preview the intermediate result set of the *Rank_1* node.

a) In the *Scenario* pane, right-click the *Rank_1* node and choose *Data Preview*.

b) Display the *Raw Data* tab.

What do you observe regarding the RANK column?

For each country (partition column), the RANK column expresses the cumulative percentage of products out of the total number of products for each country.

6. Filter the result set on COUNTRY = 'CN'.

a) Choose the arrow at the right of the *COUNTRY* column header.

b) In the *Filter* field, enter `CN` and press Enter.

How many rows are returned?

3 rows

What is the max percentage in the RANK column?

0.1; that is, 10%

Can you guess how many different products were sold to China?

The 3 rows represent exactly 10% of the list of products sold to China, so in total 30 different products were sold to China.

7. Refresh the preview of the *CVCS_SO_RANK* calculation view.
   a) Choose the *Refresh* icon in the *Raw Data* pane.

   What do you notice regarding the result set order?

   The result set does not seem to be ordered.

   What would you suggest to keep the result set ordered logically?

   It is possible to add sort criteria in the Semantics.

8. Close the preview tabs.
9. In the definition of the *CVCS_SO_RANK* calculation view, sort the result set of the calculation view by COUNTRY (asc) and RANK (asc).
   a) In the *Semantics* node, display the *Columns* tab.
   b) Choose *Sort Result Set*.

   > Note:
   > Depending on your screen size and display mode, the button might be hidden. In this case, choose the » *Toolbar Overflow* button.

   c) Choose + *Add*.
   d) Select the *COUNTRY* column.
   e) Keep the default sort order *Ascending*.
   f) Repeat these steps for the *RANK* column.
10. Deploy the *CVC_SO_RANK* calculation view.
    Check the build status after activation.
    a) Choose the *Deploy* icon for your *CVC_SO_RANK* calculation view.
    b) In the *Console* pane, check that the deployment completes successfully.

Result

> Note:
> The following intermediate solution object is available at this stage:
> *solutions → CVC_SO_RANK_00_STAGE6*.

11. Preview the *CVCS_SO_RANK* calculation view and confirm that the result set is now logically ordered.
    a) Right-click the *CVC_SO_RANK* calculation view and choose *Data Preview*.
    b) Display the *Raw Data* tab.
    c) Check that the result set is now ordered in a relevant way.
12. Close all open tabs except the *Calculation View editor* tab for the *CSC_SO_RANK* calculation view.
    a) Select then right-click the tab *CVC_SO_RANK.hdbcalculationview* and choose *Close Others*.

> Note:
> You can use the tooltip to check the complete tab name. As an aside, a design-time file always include the file extension — here, *.hdbcalculationview*– which is not the case for data preview tabs.

Task 4: Use the Sum Aggregation Function

Now, you would like to find the products that sell worst in each country, but this time you want the number of products in the result set for each country to be determined as a percentage of the total sales in this country.

More specifically, you want to extract and rank, in each country, the products that generate the lowest revenue and represent 1% of the total sales (GROSS_AMOUNT) in the country.

1. In the *CVC_SO_RANK* calculation view, adjust the Definition of the Rank node as follows (the table only mentions the properties you need to modify):

Property	Value
Aggregation Function	*Sum*
Target Value	*Fixed* Value: `0.01`
Sort Column	*GROSS_AMOUNT* (direction: *Ascending)*

> Note:
> Keep the *Result Set Type* setting *Percentage*.

    a) Select the *Rank_1* node.
    b) In the *Definition* tab, adjust the properties as per the table.
2. Deploy the *CVC_SO_RANK* calculation view.
   Check the deployment status after activation.
    a) Choose the *Deploy* icon for your *CVC_SO_RANK* calculation view.
    b) In the *Console* pane, check that the deployment completes successfully.

Result

> Note:
> The following intermediate solution object is available at this stage:
>
> *solutions → CVC_SO_RANK_00_STAGE7*

3. Preview the Calculation View. Check that the result subset for COUNTRY = 'CA' is consistent.
   a) Right-click the *CVC_SO_RANK* calculation view and choose *Data Preview*.
   b) Display the *Raw Data* tab.
   c) Choose the arrow at the right of the *COUNTRY* column header.
   d) In the *Filter* field, enter **CA** and press Enter.
   e) Confirm that the cumulative rank does not exceed 0.01 for any of the returned rows (in particular the last one).

   How many rows do you get for Canada?

   14 rows.

4. Close the data preview tab.
5. (Optional) Consider another way to achieve a similar result.

   Can you think of another approach that should work without modifying the sort order for the *GROSS_AMOUNT* column?

   An alternative would be to just change the *Result Set Direction* to *Down* instead of *Top*, and sort the GROSS_AMOUNT column in descending order. This should normally retrieve, for each country, the lowest gross amounts representing an overall 10% of the total sales in the country.

6. (Optional) Adjust the Definition of the Rank node as follows (the table only mentions the properties you need to modify).

Property	Value
Aggregation Function	*Sum*
Result Set Direction	*Down*
Sort Column	*GROSS_AMOUNT* (direction: *Descending)*

   a) Select the *Rank_1* node.
   b) In the *Definition* tab, adjust the properties as per the table.

7. (Optional) Deploy the *CVC_SO_RANK* calculation view.
   Check the deployment status after activation.

a) Choose the *Deploy* icon for your *CVC_SO_RANK* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

8. (Optional) Preview the Calculation View. Compare the results for COUNTRY = 'CA' with what you got before.

a) Right-click the *CVC_SO_RANK* calculation view and choose *Data Preview*.

b) Display the *Raw Data* tab.

c) Choose the arrow at the right of the *COUNTRY* column header.

d) In the *Filter* field, enter **CA** and press Enter.

Do you get the same 14 rows for Canada?

No. The result set for Canada includes 15 rows.

Can you explain why?

Actually, by selecting the bottom of the sorted result set, the Rank node has excluded the TOP amounts accounting for 90% of the total sales for each country, applying the rule “Cumulative GROSS_AMOUNT lower or equal to 90%”. So the remaining amounts can represent slightly more than 10% (which was not the case before), meaning that an additional row (the one taking the cumulative amount from just below 10% to just above 10%) is included.

Unit 8
Exercise 14

Create Restricted Columns

Exercise Objectives

After completing this exercise, you will be able to:

- Define Restricted Columns

Business Example

You are a consultant at a customer that sells electronic products. The computer equipment sales are above target, but due to a competitive market, the input device sales are below the expectations of the management. You have been asked to build individual measures to display total sales for their input device and computer sales.

This exercise takes you through the process of creating restricted measures to fulfill the reporting requirements.

Overview of Exercise Tasks

- Task 1: Create a New Calculation View
- Task 2: Create Restricted Columns by Product Type

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a New Calculation View

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation view using the data in the following table:

Field	Value
Name	CVC_SALES_ANALYSIS_RESTR
Label	Product sales analysis
Data Category	CUBE
With Star Join	[Deselected]

3. In the calculation view scenario, under the *Aggregation* node, add a Join node and define it as the source for the *Aggregation* node.
4. Add the following tables from the *TRAINING* schema to the *Join_1* node:
   - *SALES_DATA*
   - *PRODUCT*

5. In your *Join_1* node, create a join between the tables using the following data:

Field	Value
Left Table	HC::SALES_DATA
Right Table	HC:PRODUCT
Join Columns	PRODUCT_ID = PRODUCT_ID
Join Type	*Left Outer*
Cardinality	*n..1*

6. In the *Join_1* node, add the following columns to output:
   - HC::SALES_DATA.CURRENCY
   - HC::PRODUCT.PRODUCT_TEXT
   - HC::SALES_DATA.AMOUNT
7. In the *Join_1* node, change the name of the *AMOUNT* column to **`TOTAL_SALES`**.
8. Propagate the three columns in the output of the *Join_1* node to the Semantics.

Note:
This is sometimes faster than mapping the columns to the output node by node, especially in cases where there are several stacked nodes above the current node.

9. Deploy the calculation view.
   Check the deployment status after activation.

Task 2: Create Restricted Columns by Product Type

You will now create two restricted columns in the *Aggregation* node.

1. In the *Aggregation* node, add a restricted column using the data in the following table:

Field	Value
Name	INPUT_DEVICE_SALES
Label	Total Sales for Input Devices
Base Measure	TOTAL_SALES

Create the following column-based restrictions:

Table 3: Restrictions

Column	Operator	Value
PRODUCT_TEXT	EQUAL	(Fixed) Keyboard
PRODUCT_TEXT	EQUAL	(Fixed) Mouse

> Note:
> To assign value text for each of the restrictions, choose *Open Search Help* (or press **F4**) in the *Value* field.

2. Add another restricted column named **COMPUTER_SALES** using the data in the following table:

Field	Value
Name	COMPUTER_SALES
Label	Total Sales for Workstations and Laptops
Base Measure	TOTAL_SALES

3. Define an expression-based restriction using an SQL expression.

   The restriction should be based on the *PRODUCT_TEXT* column and include the products *Laptop* and *Workstation*.

> Note:
> To enter an SQL expression, select *Expression*.

   The SQL expression can be written as:

```
"PRODUCT_TEXT" = 'Laptop' OR "PRODUCT_TEXT" LIKE '%station'
```

> Hint:
> Don't pay attention to the red underline for 'Laptop' and LIKE in your expression. It should work all the same.

4. Deploy the calculation view.

   Check the deployment status after activation.

5. Preview the data of the calculation view and check the behavior of the two restricted columns you have defined.

6. Close the *Data Preview* tab.

Unit 8
Solution 14

Create Restricted Columns

Exercise Objectives

After completing this exercise, you will be able to:

- Define Restricted Columns

Business Example

You are a consultant at a customer that sells electronic products. The computer equipment sales are above target, but due to a competitive market, the input device sales are below the expectations of the management. You have been asked to build individual measures to display total sales for their input device and computer sales.

This exercise takes you through the process of creating restricted measures to fulfill the reporting requirements.

Overview of Exercise Tasks

- Task 1: Create a New Calculation View
- Task 2: Create Restricted Columns by Product Type

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a New Calculation View

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation view using the data in the following table:

Field	Value
Name	CVC_SALES_ANALYSIS_RESTR
Label	Product sales analysis
Data Category	CUBE
With Star Join	[Deselected]

   a) In the *Explorer* view, right-click the folder *HC300_A####* → *db* → *src* → *exercises* and choose *New File*.

   b) Enter **`CVC_SALES_ANALYSIS_RESTR.hdbcalculationview`** as file name and press *Enter*.

   c) Enter the calculation view properties as per the table above.

d) Choose *Create*.

Result

The calculation view graphical editor opens on the right of your screen.

3. In the calculation view scenario, under the *Aggregation* node, add a Join node and define it as the source for the *Aggregation* node.

a) In the *Scenario* pane, drop a new Join node below the Aggregation node.

Result

The new join node is automatically named *Join_1*.

b) Select the *Join_1* node and drag the arrow icon to the *Aggregation* node.

4. Add the following tables from the *TRAINING* schema to the *Join_1* node:

- *SALES_DATA*
- *PRODUCT*

a) Select the new *Join_1* node and choose + on the right of the node.

b) In the *Find Data Sources* window, select the *Search* field and enter **`SALES_DATA`**.

c) Select the *SALES_DATA* table with synonym *HC::SALES_DATA* and choose *Finish*.

d) Repeat the previous steps for the other table, entering **`PRODUCT`** in the *Search* field.

5. In your *Join_1* node, create a join between the tables using the following data:

Field	Value
Left Table	HC::SALES_DATA
Right Table	HC:PRODUCT
Join Columns	PRODUCT_ID = PRODUCT_ID
Join Type	*Left Outer*
Cardinality	*n..1*

a) In the *Join Definition* tab of the *Join_1* node, drag a connecting line between the columns *HC::SALES_DATA.PRODUCT_ID* and *HC::PRODUCT.PRODUCT_ID*.

b) Select the join connector and define the properties as shown in the table.

6. In the *Join_1* node, add the following columns to output:

- HC::SALES_DATA.CURRENCY
- HC::PRODUCT.PRODUCT_TEXT
- HC::SALES_DATA.AMOUNT

a) In the *Mapping* tab of the *Join_1* node, select the relevant columns and choose the *add to output* radio button.

b) Alternatively, you can right-click each column and choose *Add to Output*.

7. In the *Join_1* node, change the name of the *AMOUNT* column to **`TOTAL_SALES`**.

a) In the *Columns* tab of the *Join_1* node, select the *AMOUNT* column.

b) Enter **`TOTAL_SALES`** as the *Name* property.

8. Propagate the three columns in the output of the *Join_1* node to the Semantics.

> Note:
> This is sometimes faster than mapping the columns to the output node by node, especially in cases where there are several stacked nodes above the current node.

a) On the *Mapping* tab of the *Join_1* node, select the three columns in the *Output Columns*.

b) Right-click the selection and choose *Propagate to Semantics*.

c) Check the list of columns to be propagated and choose *OK*.

9. Deploy the calculation view.

Check the deployment status after activation.

a) Choose the *Deploy* icon for your *CVC_SALES_ANALYSIS_RESTR* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

Task 2: Create Restricted Columns by Product Type

You will now create two restricted columns in the *Aggregation* node.

1. In the *Aggregation* node, add a restricted column using the data in the following table:

Field	Value
Name	INPUT_DEVICE_SALES
Label	Total Sales for Input Devices
Base Measure	TOTAL_SALES

Create the following column-based restrictions:

Table 3: Restrictions

Column	Operator	Value
PRODUCT_TEXT	EQUAL	(Fixed) Keyboard
PRODUCT_TEXT	EQUAL	(Fixed) Mouse

> Note:
> To assign value text for each of the restrictions, choose *Open Search Help* (or press **`F4`**) in the *Value* field.

a) Select the *Aggregation* node and display the *Restricted Columns* tab.

b) Choose +.

c) Select the newly created column to display its properties.

d) Enter the data as shown in the table, both for the *GENERAL* and *RESTRICTIONS* areas.

Hint:
Use the + sign to add the two restrictions. You can use the value help to choose the correct value.

2. Add another restricted column named **COMPUTER_SALES** using the data in the following table:

Field	Value
Name	COMPUTER_SALES
Label	Total Sales for Workstations and Laptops
Base Measure	TOTAL_SALES

a) Choose *Back* at the top left of the restricted column pane.

b) Choose +.

c) Select the newly created column to display its properties.

d) In the *GENERAL* area, enter the data as shown in the table.

3. Define an expression-based restriction using an SQL expression.
The restriction should be based on the *PRODUCT_TEXT* column and include the products *Laptop* and *Workstation*.

Note:
To enter an SQL expression, select *Expression*.

The SQL expression can be written as:

```
"PRODUCT_TEXT" = 'Laptop' OR "PRODUCT_TEXT" LIKE '%station'
```

Hint:
Don't pay attention to the red underline for 'Laptop' and LIKE in your expression. It should work all the same.

a) In *Restrictions*, select *Expression*.

b) Enter the expression.

c) Choose *Back*.

4. Deploy the calculation view.
Check the deployment status after activation.

a) Choose the *Deploy* icon for your *CVC_SALES_ANALYSIS_RESTR* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

5. Preview the data of the calculation view and check the behavior of the two restricted columns you have defined.
   a) Select and right-click the *CVC_SALES_ANALYSIS_RESTR* calculation view and choose *Data Preview*.
   b) In the *Raw Data* tab, check that the values of the two restricted columns are consistent for each product.
   c) In the *Analysis* tab, drag the three measures to the *Value Axis* pane to see the overall sum of all measures.
6. Close the *Data Preview* tab.

Unit 8
Exercise 15

Create Calculated Columns

Exercise Objectives

After completing this exercise, you will be able to:

- Create Calculated Columns
- Use a Restricted Measure in a Calculated Measure to compute a “share of total sales” percentage for some products

Business Example

You are a consultant at a customer that sells electronic products. The computer equipment sales are above target, but due to a competitive market, the input device sales are below the expectations of the management. You have been asked to build individual measures to display total sales for their input device and computer sales.

Management wants to separately analyze the percentage sales share of either input device or computer sales in respect of the total company sales.

This exercise takes you through the process of creating calculated measures to fulfill the reporting requirements.

Prerequisites

You need to have completed the *Create Restricted Columns* exercise.

Note:
In case you have not completed the previous exercise, you can use the *CVC_SALES_ANALYSIS_RESTR_00_STAGE1* view provided in the `solutions` folder.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Open your *CVC_SALES_ANALYSIS_RESTR* calculation view. (Or the *CVC_SALES_ANALYSIS_RESTR_00_STAGE1* one provided in the `solutions` folder.)
3. In the *Aggregation* node, add a calculated column for the input device sales share using the data in the following table:

Field	Value
Name	INPUT_DEVICE_SALES_SHARE
Label	Percentage Share of Input Devices out of the Total Sales

Field	Value
Data Type	• DECIMAL • Length: 4 • Scale: 2
Column Type	Measure

4. Add the following expression to the calculated column and validate the syntax:

```
"INPUT_DEVICE_SALES" / "TOTAL_SALES"
```

Hint:
Instead of writing the entire expressions, you can select the Expression Editor and double-click the column names from the *Elements* pane.

You can also use the auto-complete feature to get a restricted list of elements (columns, input parameters, and so on) or functions based on your entry.

5. Add a calculated column for the computer sales share using the data in the following table:

Field	Value
Name	COMPUTER_SALES_SHARE
Label	Percentage Share of Computers out of the Total Sales
Data Type	• DECIMAL • Length: 4 • Scale: 2
Column Type	Measure

6. Add the following expression to the calculated column and validate the syntax:

```
"COMPUTER_SALES" / "TOTAL_SALES"
```

7. Deploy the calculation view.
   Check the deployment status after activation.
8. Preview the result data of the calculation view in the *Analysis* tab, showing the data in *Table Display* mode. To do so, use the following settings:
   - Labels axis: CURRENCY
   - Values axis: (all measures)

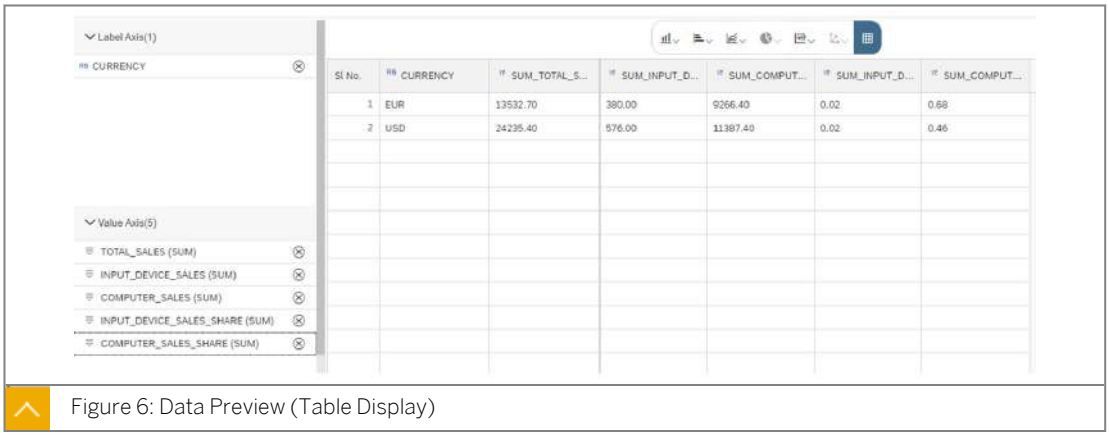


Figure 6: Data Preview (Table Display)

9. Close all open tabs in SAP Business Application Studio.

Unit 8
Solution 15

Create Calculated Columns

Exercise Objectives

After completing this exercise, you will be able to:

- Create Calculated Columns
- Use a Restricted Measure in a Calculated Measure to compute a “share of total sales” percentage for some products

Business Example

You are a consultant at a customer that sells electronic products. The computer equipment sales are above target, but due to a competitive market, the input device sales are below the expectations of the management. You have been asked to build individual measures to display total sales for their input device and computer sales.

Management wants to separately analyze the percentage sales share of either input device or computer sales in respect of the total company sales.

This exercise takes you through the process of creating calculated measures to fulfill the reporting requirements.

Prerequisites

You need to have completed the *Create Restricted Columns* exercise.

Note:
In case you have not completed the previous exercise, you can use the *CVC_SALES_ANALYSIS_RESTR_00_STAGE1* view provided in the `solutions` folder.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Open your *CVC_SALES_ANALYSIS_RESTR* calculation view. (Or the *CVC_SALES_ANALYSIS_RESTR_00_STAGE1* one provided in the `solutions` folder.)
3. In the *Aggregation* node, add a calculated column for the input device sales share using the data in the following table:

Field	Value
Name	INPUT_DEVICE_SALES_SHARE
Label	Percentage Share of Input Devices out of the Total Sales

Field	Value
Data Type	• DECIMAL • Length: 4 • Scale: 2
Column Type	Measure

a) In the *Aggregation* node, select the *Calculated Columns* tab.

b) Choose + and choose *Calculated Column*.

c) Select the newly created *CC_1* column to display the details.

d) Enter the data as shown in the table. You will find the *Column Type* property under the *Semantics* pane.

4. Add the following expression to the calculated column and validate the syntax:

```
"INPUT_DEVICE_SALES" / "TOTAL_SALES"
```

Hint:
Instead of writing the entire expressions, you can select the Expression Editor and double-click the column names from the *Elements* pane.

You can also use the auto-complete feature to get a restricted list of elements (columns, input parameters, and so on) or functions based on your entry.

a) Expand the *Expression* pane, and enter the listed expression.

Alternatively, choose *Expression Editor* and double-click the column names to select them from the *Elements* list.

b) Choose *Validate Syntax*.

5. Add a calculated column for the computer sales share using the data in the following table:

Field	Value
Name	COMPUTER_SALES_SHARE
Label	Percentage Share of Computers out of the Total Sales
Data Type	• DECIMAL • Length: 4 • Scale: 2
Column Type	Measure

a) If needed, to return to the list of calculated columns, choose *Back*.

b) In the Aggregation node, select the *Calculated Columns* tab.

c) Choose + and choose *Calculated Column*.

d) Select the new calculated column and enter the data as shown in the table. You will find the *Column Type* property under the *Semantics* pane.

6. Add the following expression to the calculated column and validate the syntax:

```
"COMPUTER_SALES" / "TOTAL_SALES"
```

a) Expand the *Expression* pane and enter the listed expression.

Alternatively, choose *Expression Editor* and double-click the column names to select them from the *Elements* list.

b) Choose *Validate Syntax*.

c) Choose *Back*.

7. Deploy the calculation view.

Check the deployment status after activation.

a) Choose the *Deploy* icon for your *CVC_SALES_ANALYSIS_RESTR* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

8. Preview the result data of the calculation view in the *Analysis* tab, showing the data in *Table Display* mode. To do so, use the following settings:

- Labels axis: CURRENCY
- Values axis: (all measures)

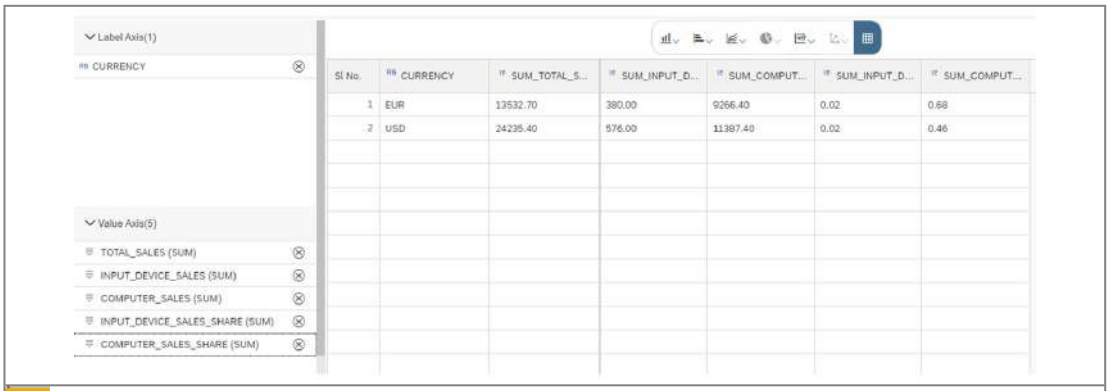


Figure 6: Data Preview (Table Display)

a) Select and right-click the *CVC_SALES_ANALYSIS_RESTR* calculation view and choose *Data Preview*.

b) Right-click the *CURRENCY* attribute and choose *Add to Labels Axis*.

c) To select all the measures, select the first one, and then press **`Shift`** while you select the last one.

d) Right-click the selection and choose *Add to Values Axis*.

9. Close all open tabs in SAP Business Application Studio.

a) Right-click any open tab and choose *Close All*.

Unit 8
Exercise 16

Define and Use Filters

Exercise Objectives

After completing this exercise, you will be able to:

- Filter view nodes on attributes or measures with a filter expression
- Understand how filtering and grouping interact with each other
- Understand how to filter data from the top query

Business Example

You want to apply a filter to an attribute to reduce the set of data retrieved by a view. You also want to explore the possibility to filter data based on the measures.

Overview of Exercise Tasks

- Task 1: Import a Calculation View and Filter its Data
- Task 2: Use Filter Conditions on Measures

Note:
In this exercise, when values include ##, replace the characters with the number your instructor assigned to you.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import a Calculation View and Filter its Data

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Import a new calculation view *CVC_SALES_FILTER* into the *src* → *exercises* folder of your *HC300* project.

   You can find the import file in your course files: *HC300* → *Calculation View Templates* → *CVC_SALES_FILTER.hdbcalculationview*.
3. Review quickly the design of the Calculation View, including the data source which you can open directly from the Calculation View definition (the *Projection_1* node at the bottom).

   The data source is a table that contains sales amount and quantities analyzed by customer, product, date, among others. In this exercise, you will focus on some of the available columns.

What columns are mapped to the top-most node of the calculation view?

4. Deploy the calculation view.
   Check the deployment status after activation.
5. Preview the calculation view data in the *Raw Data* tab.
6. You want to filter the data of the calculation view to return only the data for customer *3000*. Adjust the calculation view definition.
7. Deploy the calculation view.
   Check the deployment status after activation.
8. Preview the calculation view data in the *Raw Data* tab and check that the filter has the expected effect on the result set.
9. Modify the calculation view so that it only returns data for customers with an ID starting with 3 (for example 3001, 3147, and so on). Deploy the calculation view and preview the results.

Hint:
Use the `LIKE` predicate in the filter expression.

Note:
An intermediate solution object is available at this stage:

solutions → CVC_SALES_FILTER_00_STAGE1

Task 2: Use Filter Conditions on Measures

Now, you would like to analyze the sales quantities, and identify which customers have bought 10 or more pieces.

1. Modify the calculation view to add a filter expression at the *Aggregation* Node level. Deploy the calculation view and preview the results.

   Does the data preview return rows for customer 3001?

   How can you explain that?

2. Modify the SQL query of the data preview in order to remove the *PRODUCT_ID* column. Then execute the query and observe the results.

Hint:
Remove this column from both the *SELECT* and *GROUP BY* sections.

How many rows are returned? For which customers?

How can you explain that no data for customer *3001* is returned?

What would you suggest to enable filtering AFTER aggregation?

3. Close the preview tabs.
4. Modify the calculation view nodes so that data can be filtered AFTER aggregation. Deploy the calculation view and preview the results.

Hint:
The simplest way is to drag the new aggregation node just onto the connector between the *Projection_1* and *Aggregation* nodes.

What do you notice regarding the data that is returned in the result set?

5. Again, modify the query to remove the *PRODUCT_ID* column.

What is the outcome? How can you explain it?

Note:
An intermediate solution object is available at this stage:

solutions → CVC_SALES_FILTER_00_STAGE2

6. How could you force the aggregation of the QUANTITY measure by *CUSTOMER_ID* only?

7. Implement this approach in your *CVC_SALES_FILTER* calculation view. Save and build the calculation view and preview the results.

   Do you get the same results?

8. Close all open tabs in Business Application Studio.

Unit 8
Solution 16

Define and Use Filters

Exercise Objectives

After completing this exercise, you will be able to:

- Filter view nodes on attributes or measures with a filter expression
- Understand how filtering and grouping interact with each other
- Understand how to filter data from the top query

Business Example

You want to apply a filter to an attribute to reduce the set of data retrieved by a view. You also want to explore the possibility to filter data based on the measures.

Overview of Exercise Tasks

- Task 1: Import a Calculation View and Filter its Data
- Task 2: Use Filter Conditions on Measures

Note:
In this exercise, when values include ##, replace the characters with the number your instructor assigned to you.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import a Calculation View and Filter its Data

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Import a new calculation view *CVC_SALES_FILTER* into the *src* → *exercises* folder of your *HC300* project.

   You can find the import file in your course files: *HC300* → *Calculation View Templates* → *CVC_SALES_FILTER.hdbcalculationview*.

   a) In the *Explorer* view, right-click your *exercises* folder and choose *Upload*.

   b) Select the *HC300* → *Calculation Views Templates* → *CVC_SALES_FILTER.hdbcalculationview* file and choose *Open*.

   Result

   The new Calculation View is imported and opens in the editor.
3. Review quickly the design of the Calculation View, including the data source which you can open directly from the Calculation View definition (the *Projection_1* node at the bottom).

a) Observe the details of the different nodes

b) In the *Projection_1* node, right-click the data source *HC::SALES_DATA* and choose *Data Preview*.

The data source is a table that contains sales amount and quantities analyzed by customer, product, date, among others. In this exercise, you will focus on some of the available columns.

What columns are mapped to the top-most node of the calculation view?

CUSTOMER_ID, PRODUCT_ID and QUANTITY

4. Deploy the calculation view.
   Check the deployment status after activation.
   a) Choose the *Deploy* icon for the *CVC_SALES_FILTER* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
5. Preview the calculation view data in the *Raw Data* tab.
   a) Right-click the *CVC_SALES_FILTER* Calculation View and choose *Data Preview*.
   b) Select the *Raw Data* tab and review the contents.
   c) Close the preview tab.
6. You want to filter the data of the calculation view to return only the data for customer *3000*. Adjust the calculation view definition.
   a) In the Calculation View editor, select the node *Projection_1*.
   b) Display the tab *Filter Expression*.
      Depending on the layout, you might have to use the *More* button.
   c) In the *Expression* area, enter `"CUSTOMER_ID" = '3000'`.
      You can use the auto-completion propose the column name. Note that the double quotes are automatically added.
   d) Choose *Validate Syntax*.
   e) In the *Validation Status* dialog box, choose *OK*.
7. Deploy the calculation view.
   Check the deployment status after activation.
   a) Choose the *Deploy* icon for the *CVC_SALES_FILTER* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
8. Preview the calculation view data in the *Raw Data* tab and check that the filter has the expected effect on the result set.
   a) Right-click the *CVC_SALES_FILTER* Calculation View and choose *Data Preview*.
   b) Select the *Raw Data* tab and review the contents.
   c) Notice that only the rows for customer *3000* are displayed.
   d) Close the preview tab.

9. Modify the calculation view so that it only returns data for customers with an ID starting with 3 (for example 3001, 3147, and so on). Deploy the calculation view and preview the results.

Hint:
Use the `LIKE` predicate in the filter expression.

a) In the *Projection_1* node, modify the filter expression as follows:
`"CUSTOMER_ID" LIKE '3%'`

b) Choose *Validate Syntax* and if it is correct, choose *OK*.

c) Choose the *Deploy* icon for the *CVC_SALES_FILTER* calculation view.

d) In the *Console* pane, check that the deployment completes successfully.

e) Right-click the *CVC_SALES_FILTER* Calculation View and choose *Data Preview*.

f) Select the *Raw Data* tab and review the contents.

g) Close the preview tab.

Note:
An intermediate solution object is available at this stage:
solutions → CVC_SALES_FILTER_00_STAGE1

Task 2: Use Filter Conditions on Measures

Now, you would like to analyze the sales quantities, and identify which customers have bought 10 or more pieces.

1. Modify the calculation view to add a filter expression at the *Aggregation* Node level. Deploy the calculation view and preview the results.

a) Select the *Aggregation* node, and in the *Filter Expression* area, enter **`"QUANTITY" >= 10`**

b) Choose *Validate Syntax* and if it is correct, choose *OK*.

c) Choose the *Deploy* icon for the *CVC_SALES_FILTER* calculation view.

d) In the *Console* pane, check that the deployment completes successfully.

e) Right-click the *CVC_SALES_FILTER* Calculation View and choose *Data Preview*.

f) Select the *Raw Data* tab and review the contents.

Does the data preview return rows for customer 3001?

No.

How can you explain that?

The total quantity for customer 3001 is 15 (according to the data source HC::SALES_DATA. But the data is filtered based on the two columns CUSTOMER_ID and PRODUCT_ID. So the quantities that are returned are first aggregated by customer and product, and then filtered.

2. Modify the SQL query of the data preview in order to remove the *PRODUCT_ID* column. Then execute the query and observe the results.

Hint:
Remove this column from both the *SELECT* and *GROUP BY* sections.

a) From the last data preview, choose SQL *Edit SQL Statement in SQL Console*.

b) In the new tab, adjust the SQL query as follows:

```
SELECT TOP 1000
    "CUSTOMER_ID",
    SUM("QUANTITY") AS "QUANTITY"
FROM "HC300_A####_HDI_DB_1"."HC::CVC_SALES_FILTER"
GROUP BY "CUSTOMER_ID";
```

c) Choose *Run*.

How many rows are returned? For which customers?

Only one row, for customer *3000*.

How can you explain that no data for customer *3001* is returned?

Although the SQL query does not request the *PRODUCT_ID* column (and so data is aggregated by product only, as shown for customer 3000 compared with the previous data preview), the filter we defined is still applied to the data ENTERING the *Aggregation* node. The aggregation logic is dynamic, depending on which columns are requested, but the filter still operates at the same level, that is, before aggregation.

What would you suggest to enable filtering AFTER aggregation?

An option could be to add an aggregation node below the top *Aggregation* node where the filter is defined.

3. Close the preview tabs.

4. Modify the calculation view nodes so that data can be filtered AFTER aggregation. Deploy the calculation view and preview the results.

Hint:
The simplest way is to drag the new aggregation node just onto the connector between the *Projection_1* and *Aggregation* nodes.

a) Display the Calculation View editor again.

b) In the palette, choose *Create Aggregation* and hover the mouse on the connector between the *Projection_1* and *Aggregation* nodes. When the connector is highlighted in bold, click to create the new node.

c) In the top-left of the *Scenario* pane, choose *Auto layout*.

d) Make sure the columns are still mapped from the *Projection_1* node to the Semantics.

e) Choose the *Deploy* icon for the *CVC_SALES_FILTER* calculation view.

f) In the *Console* pane, check that the deployment completes successfully.

g) Right-click the *CVC_SALES_FILTER* Calculation View and choose *Data Preview*.

h) Select the *Raw Data* tab and review the contents.

What do you notice regarding the data that is returned in the result set?

It is still filtered on the *QUANTITY* column before aggregation is applied.

5. Again, modify the query to remove the *PRODUCT_ID* column.

a) From the last data preview, choose SQL *Edit SQL Statement in SQL Console*.

b) In the new tab, adjust the SQL query as follows:

```
SELECT TOP 1000
    "CUSTOMER_ID",
    SUM("QUANTITY") AS "QUANTITY"
FROM "HC300_A####_HDI_DB_1"."HC::CVC_SALES_FILTER"
GROUP BY "CUSTOMER_ID";
```

c) Choose *Run*.

What is the outcome? How can you explain it?

This time, we get the expected result. Because the column *PRODUCT_ID* was not requested, the data has been grouped by *CUSTOMER_ID* and aggregated by the *Aggregation_1* node, and then the filter on the *QUANTITY* column has been applied.

Note:
An intermediate solution object is available at this stage:

solutions → CVC_SALES_FILTER_00_STAGE2

6. How could you force the aggregation of the QUANTITY measure by *CUSTOMER_ID* only?

   An approach would be to remove the *PRODUCT_ID* from the columns mapping at the *Aggregation_1* node.

7. Implement this approach in your *CVC_SALES_FILTER* calculation view. Save and build the calculation view and preview the results.
   a) Close the preview tabs.
   b) In the Calculation View editor, select the *Aggregation_1* node.
   c) In the *Mapping* tab, right-click the *PRODUCT_ID* column in the *Output Columns* area and choose *Remove Output Column*.
   d) To confirm the column removal, choose *Yes*.
   e) In the *Aggregation* node, check whether the *PRODUCT_ID* has been removed. If not, repeat the same step.
   f) Choose the *Deploy* icon for the *CVC_SALES_FILTER* calculation view.
   g) In the *Console* pane, check that the deployment completes successfully.
   h) Right-click the *CVC_SALES_FILTER* Calculation View and choose *Data Preview*.
   i) Select the *Raw Data* tab and review the contents.

   Do you get the same results?

   Yes, the aggregation and filtering are executed in the right order, so the calculation view can be used with its default query (in particular, no need to adapt the query any longer).

8. Close all open tabs in Business Application Studio.
   a) Right-click any open tab and choose *Close All*.

Unit 8
Exercise 17

Define Filtering on SAP Client

Exercise Objectives

After completing this exercise, you will be able to:

- Use the native SAP Client filtering capabilities of graphical calculation views

Business Example

You want to use the SAP native CLIENT dimension (also known as MANDT) to retrieve consistent data.

Note:
For participants who might not be familiar with the classical database structure of SAP solutions such as SAP S/4HANA and SAP BW/4HANA, please note that the concept of CLIENT does not relate to any customer, but is generally intended to distinguish different sets of data (for example, *Development*, *Quality Assurance*, *Training*) within the same non-productive SAP system (same schema in the same database).

On production systems, it is always recommended to have only one CLIENT.

Overview of Exercise Tasks

- Task 1: Import a Calculation View and Preview its Data
- Task 2: Modify the Calculation View to Make It Client-Dependent

Note:
In this exercise, when values include A####, replace the characters with the user code assigned to you (one letter and five digits).

Prerequisites

- Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.
- The initial script should have been executed. It creates a user (A####) with the correct access rights.
- A connection to the HANA Cloud database (`HC300_DB (A####) (SSO enabled)`) should have been setup.

Task 1: Import a Calculation View and Preview its Data

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.

2. Import a new calculation view *CVD_CITIES_CLIENT* into the *src* → *exercises* folder of your *HC300_A####* project.

   You can find the import file in your course files: *HC300* → *Calculation View Templates* → *CVD_CITIES_CLIENT.hdbcalculationview*.

3. Review quickly the design of the Calculation View, in particular the *Client Column* setting for the data source *HC::CITIES*. You can also preview the content of this data source directly from the Calculation View definition (the *Projection_1* node at the bottom).

4. Deploy the calculation view.

   Check the deployment status after activation.

5. Preview the calculation view data in the *Raw Data* tab.

   The column *MANDT* contains the SAP Client number.

   In the Data Preview, how many distinct values of the field *CITY* do you see?

   _____

   _____

   _____

   At this stage, do you think that the view property *Default Client* has an effect on the retrieved data?

   _____

   _____

   _____

6. Close the preview tabs.

Task 2: Modify the Calculation View to Make it Client-Dependent

1. Define the *MANDT* from the data source *HC::CITIES* as the Client Column.

2. Deploy the calculation view.

   Check the deployment status after activation.

3. Preview the calculation view data in the *Raw Data* tab.

   What do you observe? Can you explain why?

   _____

   _____

   _____

4. Close the preview tab.

5. Modify your calculation view to display only data for SAP Client number **200**.

6. Deploy the calculation view.

   Check the deployment status after activation.

7. Preview the calculation view data in the *Raw Data* tab.

> Note:
> An intermediate solution object is available at this stage. This can be found at *solutions → CVD_CITIES_CLIENT_00_STAGE1*.

In the Data Preview, how many distinct values of the field *CITY* do you see?

8. Close the preview tab.
9. Modify your calculation view by changing the *Default Client* property to *Session Client*.
10. Deploy the calculation view.
    Check the deployment status after activation.
11. Preview the calculation view data in the *Raw Data* tab.

    What do you observe?

    Can you explain why?

    To be able to actually display the data filtered by the default client of your *A####* user, you will display the data of the calculation view from an SQL console connected to the database. In this context, the actual user executing the query will be your *A####* user.

> Note:
> In the users table of the *HC300_DB (A####) (SSO enabled)* SAP HANA database, your user *A####* has been assigned a default client value of 800, which means that whenever you execute a view that has the *Default Client* property set to *Session Client*, this parameter is passed to the view to filter the data and extract only the data for which the CLIENT column equals 800.

    Alternatively, you could also query the data from an external tool such as SAP BusinessObjects Analysis for MS Excel.

12. Preview the data of the Calculation View *CVD_CITIES_CLIENT* from the database connection.

Caution:
Make sure that you do NOT open the Console for your container.

Hint:
You can use the search capabilities of the Database Explorer (specifying a schema and searching the column views) to find the runtime object of your calculation view easily.

What do you observe?

13. Close all open tabs in Business Application Studio and come back to the *Explorer*.

Unit 8
Solution 17

Define Filtering on SAP Client

Exercise Objectives

After completing this exercise, you will be able to:

- Use the native SAP Client filtering capabilities of graphical calculation views

Business Example

You want to use the SAP native CLIENT dimension (also known as MANDT) to retrieve consistent data.

Note:
For participants who might not be familiar with the classical database structure of SAP solutions such as SAP S/4HANA and SAP BW/4HANA, please note that the concept of CLIENT does not relate to any customer, but is generally intended to distinguish different sets of data (for example, *Development*, *Quality Assurance*, *Training*) within the same non-productive SAP system (same schema in the same database).

On production systems, it is always recommended to have only one CLIENT.

Overview of Exercise Tasks

- Task 1: Import a Calculation View and Preview its Data
- Task 2: Modify the Calculation View to Make It Client-Dependent

Note:
In this exercise, when values include A####, replace the characters with the user code assigned to you (one letter and five digits).

Prerequisites

- Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.
- The initial script should have been executed. It creates a user (A####) with the correct access rights.
- A connection to the HANA Cloud database (`HC300_DB (A####) (SSO enabled)`) should have been setup.

Task 1: Import a Calculation View and Preview its Data

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.

2. Import a new calculation view *CVD_CITIES_CLIENT* into the *src* → *exercises* folder of your *HC300_A####* project.

   You can find the import file in your course files: *HC300* → *Calculation View Templates* → *CVD_CITIES_CLIENT.hdbcalculationview*.

   a) In the explorer, right-click your *exercises* folder and choose *Upload*.

   b) Select the *HC300* → *Calculation Views Templates* → *CVD_CITIES_CLIENT.hdbcalculationview* file and choose *Open*.

   Result

   The new Calculation View *CVD_CITIES_CLIENT* is imported and opens in the editor.

3. Review quickly the design of the Calculation View, in particular the *Client Column* setting for the data source *HC::CITIES*. You can also preview the content of this data source directly from the Calculation View definition (the *Projection_1* node at the bottom).

   a) Observe the details of the different nodes.

   b) In the *Projection_1* node, select the *Mapping* tab and select the *HC::CITIES* data source. Then expand the *PROPERTIES* pane and check the *Client Column* setting.

   c) In the *Projection_1* node, right-click the data source *HC::CITIES* and choose *Data Preview*.

4. Deploy the calculation view.

   Check the deployment status after activation.

   a) Choose the *Deploy* icon for the *CVD_CITIES_CLIENT* calculation view.

   b) In the *Console* pane, check that the deployment completes successfully.

5. Preview the calculation view data in the *Raw Data* tab.

   The column *MANDT* contains the SAP Client number.

   a) Right-click the *CVD_CITIES_CLIENT* calculation view and choose *Data Preview*.

   b) Select the *Raw Data* tab and review the contents.

   In the Data Preview, how many distinct values of the field *CITY* do you see?

   20

   At this stage, do you think that the view property *Default Client* has an effect on the retrieved data?

   No. The *Default Client* setting cannot have any effect because no CLIENT column has been defined for the source table *HC::CITIES*.

6. Close the preview tabs.

Task 2: Modify the Calculation View to Make it Client-Dependent

1. Define the *MANDT* from the data source *HC::CITIES* as the Client Column.

   a) In the *Projection_1* node, display the *Mapping* tab.

   b) Select the Data Source *HC::CITIES*.

c) In the *Properties* pane, set the *Client Column* property to *MANDT*.

2. Deploy the calculation view.
   Check the deployment status after activation.
   a) Choose the *Deploy* icon for the *CVD_CITIES_CLIENT* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
3. Preview the calculation view data in the *Raw Data* tab.
   a) Right-click the *CVD_CITIES_CLIENT* calculation view and choose *Data Preview*.
   b) Select the *Raw Data* tab.

   What do you observe? Can you explain why?

   The view retrieves all rows, regardless of the value of the *MANDT* column. This is because when the *Default Client* setting is left blank, the rows of a data source are not filtered by Client, even if the *Client Column* has been specified for this data source.
4. Close the preview tab.
5. Modify your calculation view to display only data for SAP Client number **200**.
   a) In the *Scenario* pane, select the *Semantics* node.
   b) In the *Default Client* drop-down list of the *View Properties* tab, choose *Fixed Client* and enter **200**.
6. Deploy the calculation view.
   Check the deployment status after activation.
   a) Choose the *Deploy* icon for the *CVD_CITIES_CLIENT* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
7. Preview the calculation view data in the *Raw Data* tab.

   Note:
   An intermediate solution object is available at this stage. This can be found at *solutions → CVD_CITIES_CLIENT_00_STAGE1*.

   a) Right-click the *CVD_CITIES_CLIENT* calculation view and choose *Data Preview*.
   b) Select the *Raw Data* tab and review the contents.

   In the Data Preview, how many distinct values of the field *CITY* do you see?

   8
8. Close the preview tab.
9. Modify your calculation view by changing the *Default Client* property to *Session Client*.
   a) In the Semantics node, select the *View Properties* tab.
   b) In the *Default Client* dropdown list, select *Session Client*.

10. Deploy the calculation view.

    Check the deployment status after activation.

    a) Choose the *Deploy* icon for the *CVD_CITIES_CLIENT* calculation view.

    b) In the *Console* pane, check that the deployment completes successfully.

11. Preview the calculation view data in the *Raw Data* tab.

    a) Right-click the *CVD_CITIES_CLIENT* calculation view and choose *Data Preview*.

    b) Select the *Raw Data* tab.

    What do you observe?

    The view does not retrieve any rows.

    Can you explain why?

    This is because, in Business Application Studio, the actual user executing the data preview is a technical user (not your *A####* user) who has no default client number assigned.

    To be able to actually display the data filtered by the default client of your *A####* user, you will display the data of the calculation view from an SQL console connected to the database. In this context, the actual user executing the query will be your *A####* user.

    Note:
    In the users table of the *HC300_DB (A####) (SSO enabled)* SAP HANA database, your user *A####* has been assigned a default client value of 800, which means that whenever you execute a view that has the *Default Client* property set to *Session Client*, this parameter is passed to the view to filter the data and extract only the data for which the CLIENT column equals 800.

    Alternatively, you could also query the data from an external tool such as SAP BusinessObjects Analysis for MS Excel.

12. Preview the data of the Calculation View *CVD_CITIES_CLIENT* from the database connection.

    Caution:
    Make sure that you do NOT open the Console for your container.

    Hint:
    You can use the search capabilities of the Database Explorer (specifying a schema and searching the column views) to find the runtime object of your calculation view easily.

    a) Go to *SAP HANA Database Explorer*.

b) In the *Database List*, select the *HC300_DB (A####) (SSO enabled)* entry.

c) Navigate the tree to display *HC300_DB (A####) (SSO enabled)* → *Catalog* → *Column Views*.

d) In the *Catalog Browser* pane, choose the *Apply filter* icon and select **Filter objects by schema**. A list of schemas should appear. You can deselect the *A####* schema and select the *HC300_A####_HDI_DB_1* one. Choose *OK*.

e) In the *Catalog Browser* pane, choose again the *Apply filter* icon and select **Filter objects by name**. enter **`CITIES`** and press **`Enter`**.

f) In the search results, right-click the column view *HC::CVD_CITIES_CLIENT* and choose *Open Data*.

g) Check the query result.

What do you observe?

By querying the data with the actual *A####* user, the data is now filtered by the default client number assigned to this user; that is, *800*.

13. Close all open tabs in Business Application Studio and come back to the *Explorer*.

a) Right-click any open tab and choose *Close All*.

b) Choose the *Explorer* icon.

Unit 8
Exercise 18

Implement Currency Conversion in a Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Apply currency conversion in a calculation view
- Leverage fixed currencies
- Reuse conversion settings between columns

Business Example

You are at a customer site where EPM data is available. USD is the general corporate reporting currency.

You have been asked to build calculation views for SAP HANA for the purpose of displaying sales data converted in the corporate currency.

Overview of Exercise Tasks

- Task 1: Import an Existing Calculation View and Analyze Its Definition
- Task 2: Implement Currency Conversion in the New Calculation View
- Task 3: Reuse the Conversion Settings in Another Column

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import an Existing Calculation View and Analyze Its Definition

The calculation view will be used later on to implement currency conversion.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Import the template calculation view *CVC_SO_CCY* into your *exercises* folder.

   A design-time file, *CVC_SO_CCY.hdbcalculationview*, is located in your exercise folder, *HC300 → Calculation Views Templates*, and is ready for import.
3. Deploy the imported calculation view.
4. Preview the data of the new calculation view, and confirm data availability. In particular, pay attention to the *CURRENCY_CODE* column and examine the types of available currencies.

Task 2: Implement Currency Conversion in the New Calculation View

You will use a column-based currency source and convert the data into a fixed currency (USD).

Note:
Use a conversion rate type called EZB (exchange rate provided by the European Central Bank). The training data includes conversion rates for EUR to USD and rate type EZB for two specific dates:

- 31/12/2012: 1 EUR = 1.3203 USD
- 31/12/2013: 1 EUR = 1.3744 USD

If you use a different date in your currency conversion settings, the system uses the first available conversion rate **before** the date you specified. For example, if you specify the conversion date 31/12/2014, you will get the same results as for conversion date 31/12/2013.

1. Modify the output columns of the calculation view to have two amounts, one in the source currency, another in the conversion USD currency. To do so, add a second instance of *GROSS_AMOUNT* to the output, and rename the two amount columns as follows:

Previous Name	New Name and New Label
GROSS_AMOUNT	*GROSS_AMOUNT_SRC*
GROSS_AMOUNT_1	*GROSS_AMOUNT_USD*

   Check that the new column has the type *Measure*.

2. At this stage, deploy the view, and preview the data.
   Note that for the moment the data is the same in both amount columns.

3. Assign the following semantics to the column *GROSS_AMOUNT_SRC*:

Field	Value
Semantic	Amount with Currency Code
Display Currency	• Type: *Column* • Assigned Column: *CURRENCY_CODE*

Note:
Do not enable this column for conversion, as we want to display it in the source currency.

4. Assign the following semantics to the column *GROSS_AMOUNT_USD*:
   You must define the semantics and enable conversion to the fixed currency *USD*.

Field	Value
Semantic Type	*Amount with Currency Code*
General Information	

Field	Value
Conversion	Yes
Decimal shifts	Yes
Rounding	No
Shift back	No
Reverse Look Up	No
Conversion Tables	
Rates	*HC::TCURR*
Configuration	*HC::TCURV*
Prefactors	*HC::TCURF*
Precision	*HC::TCURX*
Notations	*HC::TCURN*
Definition	
Client for currency conversion	*Fixed* Client: **800**
Exchange Type	*Fixed* *EZB*
Source Currency	Type: *Column* Assigned column: *CURRENCY_CODE*
Target Currency	Type: *Fixed* Assigned currency: *USD*
Conversion Date	*Fixed* Date: **20121231**
Exchange Rate	[blank]
Data Type	*Decimal (15,2)*
Upon Failure	*Fail*
Generate result currency column	No
Accuracy	Intermediate Rounding

5. Deploy the calculation view and preview the data.

   You will now see the two measures; the *GROSS_AMOUNT_SRC* in source currency, and the *GROSS_AMOUNT_USD* converted into the corporate reporting currency (USD).

Note:
As the two measures are assigned a semantic *Amount with Currency Code*, in the data preview, the currency is displayed in the same column as the amount.

Note:
The following intermediate solution object is available at this stage:

solutions → CVC_SO_CCY_00_STAGE1

Task 3: Reuse the Conversion Settings in Another Column

You want to add the column *NET_AMOUNT* to your Calculation View, and apply the same conversion settings as column *GROSS_AMOUNT_USD* to this column.

1. In the *Aggregation* node, add 2 instances of the column *NET_AMOUNT* to the output and rename the two new columns of the output as follows:

Previous Name	New Name and Label
NET_AMOUNT	*NET_AMOUNT_SRC*
NET_AMOUNT_1	*NET_AMOUNT_USD*

2. Assign the following semantics to the column *NET_AMOUNT_SRC*:

Field	Value
Semantic	Amount with Currency Code
Display Currency	• Type: *Column* • Assigned Column: *CURRENCY_CODE*

Note:
Do not enable this column for conversion, as we want to display it in the source currency.

3. Apply to the *NET_AMOUNT_USD* the same conversion settings as for column *GROSS_AMOUNT_USD*. Instead of defining these settings from scratch, you will reference the ones defined for the column *GROSS_AMOUNT_USD*.
4. Deploy the calculation view.
5. Preview the data of the calculation view.
6. Close all open tabs in Business Application Studio.

Unit 8
Solution 18

Implement Currency Conversion in a Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Apply currency conversion in a calculation view
- Leverage fixed currencies
- Reuse conversion settings between columns

Business Example

You are at a customer site where EPM data is available. USD is the general corporate reporting currency.

You have been asked to build calculation views for SAP HANA for the purpose of displaying sales data converted in the corporate currency.

Overview of Exercise Tasks

- Task 1: Import an Existing Calculation View and Analyze Its Definition
- Task 2: Implement Currency Conversion in the New Calculation View
- Task 3: Reuse the Conversion Settings in Another Column

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import an Existing Calculation View and Analyze Its Definition

The calculation view will be used later on to implement currency conversion.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Import the template calculation view *CVC_SO_CCY* into your *exercises* folder.

   A design-time file, *CVC_SO_CCY.hdbcalculationview*, is located in your exercise folder, *HC300* → *Calculation Views Templates*, and is ready for import.

   a) Right-click the *exercises* folder and choose *Upload*.

   b) Select the file *HC300* → *Calculation Views Templates* → *CVC_SO_CCY.hdbcalculationview* and choose *Open*.
3. Deploy the imported calculation view.

   a) Choose the *Deploy* icon for the *CVC_SO_CCY* calculation view.

   b) In the *Console* pane, check that the deployment completes successfully.

4. Preview the data of the new calculation view, and confirm data availability. In particular, pay attention to the *CURRENCY_CODE* column and examine the types of available currencies.
   a) Right-click the calculation view and choose *Data Preview*.
   b) Select the *Raw Data* tab and review the contents.
   c) Notice that all the data is expressed in EUR.
   d) Close the Data Preview.

Task 2: Implement Currency Conversion in the New Calculation View

You will use a column-based currency source and convert the data into a fixed currency (USD).

Note:
Use a conversion rate type called EZB (exchange rate provided by the European Central Bank). The training data includes conversion rates for EUR to USD and rate type EZB for two specific dates:

- 31/12/2012: 1 EUR = 1.3203 USD
- 31/12/2013: 1 EUR = 1.3744 USD

If you use a different date in your currency conversion settings, the system uses the first available conversion rate **before** the date you specified. For example, if you specify the conversion date 31/12/2014, you will get the same results as for conversion date 31/12/2013.

1. Modify the output columns of the calculation view to have two amounts, one in the source currency, another in the conversion USD currency. To do so, add a second instance of *GROSS_AMOUNT* to the output, and rename the two amount columns as follows:

Previous Name	New Name and New Label
GROSS_AMOUNT	*GROSS_AMOUNT_SRC*
GROSS_AMOUNT_1	*GROSS_AMOUNT_USD*

   Check that the new column has the type *Measure*.
   a) In the *Mapping* tab of the *Aggregation* node, right-click the *GROSS_AMOUNT* column and choose *Add to Output*.
   b) Choose *OK* in the renaming dialog.
   c) Select the *Columns* tab of the Semantics node.
   d) Enter the new names and labels for the two amount columns as shown in the table.
   e) Select the *Semantics* node and check that the new column has the type *Measure*.
2. At this stage, deploy the view, and preview the data.
   a) Choose the *Deploy* icon for the *CVC_SO_CCY* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.

c) Right-click the calculation view and choose *Data Preview*.

d) Select the *Raw Data* tab and review the contents.

e) Close the *Data Preview* tab.

Note that for the moment the data is the same in both amount columns.

3. Assign the following semantics to the column *GROSS_AMOUNT_SRC*:

Field	Value
Semantic	Amount with Currency Code
Display Currency	• Type: *Column* • Assigned Column: *CURRENCY_CODE*

Note:
Do not enable this column for conversion, as we want to display it in the source currency.

a) In the *Semantics* node, select the *Columns* tab.

b) Open the value help of the *Semantics* property for the *GROSS_AMOUNT_SRC* column.

c) In the *Semantic Type* drop down list, select *Amount with Currency Code*.

d) In the *Assign Semantics For GROSS_AMOUNT_SRC* dialog, enter the values as per the above table.

e) Choose *OK*.

4. Assign the following semantics to the column *GROSS_AMOUNT_USD*:
You must define the semantics and enable conversion to the fixed currency *USD*.

Field	Value
Semantic Type	*Amount with Currency Code*
General Information	
Conversion	Yes
Decimal shifts	Yes
Rounding	No
Shift back	No
Reverse Look Up	No
Conversion Tables	
Rates	*HC::TCURR*
Configuration	*HC::TCURV*
Prefactors	*HC::TCURF*

Field	Value
Precision	*HC::TCURX*
Notations	*HC::TCURN*
Definition	
Client for currency conversion	*Fixed* Client: **800**
Exchange Type	*Fixed* *EZB*
Source Currency	Type: *Column* Assigned column: *CURRENCY_CODE*
Target Currency	Type: *Fixed* Assigned currency: *USD*
Conversion Date	*Fixed* Date: **20121231**
Exchange Rate	[blank]
Data Type	*Decimal (15,2)*
Upon Failure	*Fail*
Generate result currency column	No
Accuracy	Intermediate Rounding

a) In the *Semantics* node, select the *Columns* tab.

b) Select the *GROSS_AMOUNT_USD* column and, in the *Semantics* field, open the value help.

c) Assign or check all the settings as specified in the table.

d) Choose *OK*.

5. Deploy the calculation view and preview the data.

a) Choose the *Deploy* icon for the *CVC_SO_CCY* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

c) Right-click the calculation view and choose *Data Preview*.

d) Select the *Raw Data* tab and review the contents.

e) Close the Data Preview tab.

You will now see the two measures; the *GROSS_AMOUNT_SRC* in source currency, and the *GROSS_AMOUNT_USD* converted into the corporate reporting currency (USD).

Note:
As the two measures are assigned a semantic *Amount with Currency Code*, in the data preview, the currency is displayed in the same column as the amount.

Note:
The following intermediate solution object is available at this stage:

solutions → CVC_SO_CCY_00_STAGE1

Task 3: Reuse the Conversion Settings in Another Column

You want to add the column *NET_AMOUNT* to your Calculation View, and apply the same conversion settings as column *GROSS_AMOUNT_USD* to this column.

1. In the *Aggregation* node, add 2 instances of the column *NET_AMOUNT* to the output and rename the two new columns of the output as follows:

Previous Name	New Name and Label
NET_AMOUNT	*NET_AMOUNT_SRC*
NET_AMOUNT_1	*NET_AMOUNT_USD*

   a) In the *Mapping* tab of the *Aggregation* node, double-click the *NET_AMOUNT* column in the *Data sources* pane.

   b) Repeat the previous step to add a second instance of the same column to the output.

   c) In the *Columns* tab of the *Semantics* node, enter the new names and labels for the two amount columns as shown in the table.

   Alternatively, in the *Mapping* tab of the *Aggregation node*, you can select each column in the *Output Columns* pane and modify its name and label in the *PROPERTIES* pane.

2. Assign the following semantics to the column *NET_AMOUNT_SRC*:

Field	Value
Semantic	Amount with Currency Code
Display Currency	• Type: *Column* • Assigned Column: *CURRENCY_CODE*

Note:
Do not enable this column for conversion, as we want to display it in the source currency.

   a) In the *Semantics* node, select the *Columns* tab.

   b) Open the value help of the *Semantics* property for the *NET_AMOUNT_SRC* column.

c) In the *Semantic Type* drop down list, select *Amount with Currency Code*.

d) In the *Assign Semantics For GROSS_AMOUNT_SRC* dialog, enter the values as per the above table.

e) Choose *OK*.

3. Apply to the *NET_AMOUNT_USD* the same conversion settings as for column *GROSS_AMOUNT_USD*. Instead of defining these settings from scratch, you will reference the ones defined for the column *GROSS_AMOUNT_USD*.

a) In the *Semantics* node, select the *GROSS_AMOUNT_USD* column.

b) Select the *Assign Semantics* icon and choose *Apply Conversion Reference*.

c) Select *Amount with Currency Code* as semantic type.

d) In the *Reuse Semantics* drop-down list, select the column *GROSS_AMOUNT_USD*.

e) Note that the conversion settings of the column *GROSS_AMOUNT_USD* are displayed, and choose *Next*.

f) In the *Target Measures* list, select the column *NET_AMOUNT_USD* and choose *Finish*.

Note:
Here, you have applied the "referenced" conversion settings to one column only. You could also have done that by selecting the drop down list *Semantics* for the *NET_AMOUNT_USD* column. However, the method you have used allows you to apply conversion settings from a reference column to several target columns.

4. Deploy the calculation view.

a) Choose the *Deploy* icon for the *CVC_SO_CCY* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

5. Preview the data of the calculation view.

a) Right-click the calculation view and choose *Data Preview*.

b) Select the *Raw Data* tab and review the contents.

c) Check that the conversion of the column *NET_AMOUNT_USD* is consistent with the one applied to the column *GROSS_AMOUNT_USD*.

6. Close all open tabs in Business Application Studio.

a) Right-click any open tab and choose *Close All*.

Unit 9
Exercise 19

Implement Variables

Exercise Objectives

After completing this exercise, you will be able to:

- Create variables to filter data based on a user's selection

Business Example

You are a consultant at a customer that sells electronics. You have been asked to build a calculation view to analyze sales data for certain customers. You would like to provide a pop-up prompt where the user must choose one or more customers each time the report is run.

Overview of Exercise Tasks

- Task 1: Import a New Calculation View for the Marketing E-Mail Campaign Analysis
- Task 2: Create a Variable to Enable Customer Selection and Filtering

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import a New Calculation View for the Marketing E-Mail Campaign Analysis

You will first import a Calculation View that displays the sales amount for each customer.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the explorer, import the *CVC_CAMPAIGN_ANALYSIS.hdbcalculationview* file from your course files folder *HC300 → Calculation Views Templates* into the folder *HC300_A#### → db → src → exercises*.
3. Review the design of the *CVC_CAMPAIGN_ANALYSIS* calculation view.
4. Deploy the *CVC_CAMPAIGN_ANALYSIS* calculation view.
   Check the deployment status.
5. Preview the calculation view data in the *Raw Data* tab.

Task 2: Create a Variable to Enable Customer Selection and Filtering

1. In the Semantics node, create a variable using the data from the following table and apply the filter to the *CUSTOMER* column.

Field	Value
Name	VAR_CUSTOMER
Label	Customer Selector
Selection Type	Single Value

Field	Value
Is Mandatory	[Selected]
Multiple Entries	[Selected]
View/Table Value Help	[Keep the Current view (selected by default)]
Reference Column	CUSTOMER
Apply Filter	CUSTOMER

2. Deploy the view.

   Check the deployment status after activation.

3. Preview the calculation view data in the *Raw Data* tab. Select the customers *Becker Berlin* and *High Tech Park*.

Hint:
To set the value of a variable or input parameter, choose the corresponding *From* cell, and then the search help button.

If needed, choose the + sign to add another entry.

4. Close all open tabs in Business Application Studio.

Unit 9
Solution 19

Implement Variables

Exercise Objectives

After completing this exercise, you will be able to:

- Create variables to filter data based on a user's selection

Business Example

You are a consultant at a customer that sells electronics. You have been asked to build a calculation view to analyze sales data for certain customers. You would like to provide a pop-up prompt where the user must choose one or more customers each time the report is run.

Overview of Exercise Tasks

- Task 1: Import a New Calculation View for the Marketing E-Mail Campaign Analysis
- Task 2: Create a Variable to Enable Customer Selection and Filtering

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import a New Calculation View for the Marketing E-Mail Campaign Analysis

You will first import a Calculation View that displays the sales amount for each customer.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the explorer, import the *CVC_CAMPAIGN_ANALYSIS.hdbcalculationview* file from your course files folder *HC300* → *Calculation Views Templates* into the folder *HC300_A####* → *db* → *src* → *exercises*.
   a) In the *Explorer* view, right-click your *exercises* folder and choose *Upload...*.
   b) Select the file *HC300* → *Calculation Views Templates* → *CVC_CAMPAIGN_ANALYSIS.hdbcalculationview* and choose *Open*.
   Result
   A new calculation view *CVC_CAMPAIGN_ANALYSIS* is created in your project and opened in a new tab.
3. Review the design of the *CVC_CAMPAIGN_ANALYSIS* calculation view.
   a) Review the *Join_1* and *Aggregation* nodes.
4. Deploy the *CVC_CAMPAIGN_ANALYSIS* calculation view.
   Check the deployment status.
   a) Choose the *Deploy* icon for the *CVC_CAMPAIGN_ANALYSIS* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
5. Preview the calculation view data in the *Raw Data* tab.

a) Right-click the calculation view and choose *Data Preview*.

b) Select the *Raw Data* tab and review the contents.

Task 2: Create a Variable to Enable Customer Selection and Filtering

1. In the Semantics node, create a variable using the data from the following table and apply the filter to the *CUSTOMER* column.

Field	Value
Name	VAR_CUSTOMER
Label	Customer Selector
Selection Type	Single Value
Is Mandatory	[Selected]
Multiple Entries	[Selected]
View/Table Value Help	[Keep the Current view (selected by default)]
Reference Column	CUSTOMER
Apply Filter	CUSTOMER

a) In the *Semantics* node, select the *Parameters* tab.

b) Choose + and choose *Variable*.

c) Select the new variable to open its properties.

d) Enter the details as shown in the table.

e) In the *Apply Filter* section, use the value help button and select the *CUSTOMER* column.

f) Choose *Back* at the top left of the variable properties screen.

2. Deploy the view.

   Check the deployment status after activation.

   a) Choose the *Deploy* icon for the *CVC_CAMPAIGN_ANALYSIS* calculation view.

   b) In the *Console* pane, check that the deployment completes successfully.

3. Preview the calculation view data in the *Raw Data* tab. Select the customers *Becker Berlin* and *High Tech Park*.

Hint:
To set the value of a variable or input parameter, choose the corresponding *From* cell, and then the search help button.

If needed, choose the + sign to add another entry.

a) Right-click the *CVC_CAMPAIGN_ANALYSIS* calculation view.

b) Choose *Data Preview*.

c) In the variables and input parameters screen, for the *VAR_CUSTOMER* variable, choose *From* and then choose the search help radio button.

d) From the list, select *Becker Berlin*.

e) Choose *OK*.

f) Choose the + icon to add another value.

g) In the new row for the *VAR_CUSTOMER* variable, choose *From* and then choose the search help radio button.

h) From the list, select *High Tech Park*.

i) Choose *OK*.

j) Choose *Open Content* and select the *Raw Data* tab.

k) Check that the output includes the specified customer only.

4. Close all open tabs in Business Application Studio.

a) Right-click any open tab and choose *Close All*.

Unit 9
Exercise 20

Implement Input Parameters

Exercise Objectives

After completing this exercise, you will be able to:

- Create input parameters with direct input and static value list
- Use input parameters in calculated columns

Business Example

You are a consultant at a customer that sells electronics. You have been asked to build a calculation view to analyze sales data for certain customers. Two marketing e-mails have been sent out to each customer. Your users want to find out how many days have passed between the two mailing campaigns and the order dates.

Overview of Exercise Tasks

- Task 1: Modify the Calculation View from the Previous Exercise
- Task 2: Create an Input Parameter to Select the E-mail Date and a Calculated Column to Determine the Number of Days Between the E-mail and the Sale Dates
- Task 3: Modify the Input Parameter to Enable User Entry with Default Value

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

You should also have completed the Implement Variables exercise. If not, you'll find an intermediate version of the view in your *solutions* folder : *CVC_CAMPAIGN_ANALYSIS_00_STAGE1*.

Task 1: Modify the Calculation View from the Previous Exercise

In your *CVC_CAMPAIGN_ANALYSIS* Calculation View, you will delete the *VAR_CUSTOMER* variable and add a new column from the source table to the Output.

The new column will be used later on to calculate the number of days elapsed since the last marketing campaigns.

1. Open your *CVC_CAMPAIGN_ANALYSIS* view.

> Note:
> If needed, you can use the *CVC_CAMPAIGN_ANALYSIS_00_STAGE1* view in the *solutions* folder.

2. Delete the variable *VAR_CUSTOMER*, which will no longer be used because you want to visualize data for all the customers.
3. In the *Join_1* node, add the column *SALES_DATA.SQL_DATE* to the output.

4. Add the new column to the *Aggregation* node.
5. In the *Semantics* node, rename the column *SQL_DATE* as **`SALE_DATE`**.
6. In the *Semantics* node, check that the new column has the type *Attribute*.

Task 2: Create an Input Parameter to Select the E-mail Date and a Calculated Column to Determine the Number of Days Between the E-mail and the Sale Dates
Calculate the number of days elapsed between the e-mail marketing campaign and the sales orders.

Note:
There were two marketing campaigns in 2019; one on January 25 and another on February 23.

1. In the semantics, create a new input parameter using the following details:

Hint:
To create an input parameter, select the *Parameters* tab of the *Semantics* node.

Table 4: General properties

Field	Value
Name	INP_EMAIL_DATE
Label	The date when the e-mail was sent
Is Mandatory	[Selected]
Multiple Entries	[NOT selected]
Parameter Type	Static List

Table 5: Parameter Type - Static List

Data Type	Date
List of values	• *Value:* 2019-01-25 *Description:* First e-mail • *Value:* 2019-02-23 *Description:* Second e-mail

Hint:
Use *Add* to define the list of values.

2. In the *Aggregation* node, add a calculated column to determine the number of days elapsed between the e-mail and the sale dates, using the following details:

Field	Value
General information	
Name	**DAYS_ELAPSED**
Description	**Days between Sale Date and e-mail**
Data Type	*INTEGER*
Semantics	
Column Type	*Attribute*
Expression	
Expression	`DAYS_BETWEEN("SALE_DATE",'$$INP_EMAIL_DATE$$')`

3. Deploy the calculation view.
   Check the deployment status after activation.

> Note:
> The following intermediate solution object is available at this stage:
> *solutions → CVC_CAMPAIGN_ANALYSIS_00_STAGE2*

4. Preview the data, and select a date for the parameter *INP_EMAIL_DATE*; for example 2019-01-25.

> Hint:
> If no input parameter shows up in the data preview tab the first time, close this tab and re-execute the data preview.

Can you explain why the calculation view retrieves NULL data for some customers?

5. Close the Data Preview.

Result

In the displayed data, check that the number of elapsed days is correctly calculated based on the input parameter that you selected.

Task 3: Modify the Input Parameter to Enable User Entry with Default Value and Calendar Value Help

When you query the view, the input parameter should be set by default to 2019-01-25, and enable the user to enter a different date.

1. Edit the input parameter *INP_EMAIL_DATE* to change its parameter according to the following table:

Field	Value
Parameter Type	Direct
Semantic Type	Date
Default Value(s)	Constant: **`2019-01-25`**

   Keep the other parameters as is.

2. Deploy the calculation view.

3. Preview the calculation view data on the *Raw Data* tab. In the *Variables and Input Parameters* screen, keep the default value or enter a different date. In the displayed data, check that the number of elapsed days is correctly calculated based on the input parameter you defined.

   Caution:
   If you choose the *Value Help*, a new dialog opens and you can select a date from the displayed calendar.

4. Close all open tabs in Business Application Studio.

Unit 9
Solution 20

Implement Input Parameters

Exercise Objectives

After completing this exercise, you will be able to:

- Create input parameters with direct input and static value list
- Use input parameters in calculated columns

Business Example

You are a consultant at a customer that sells electronics. You have been asked to build a calculation view to analyze sales data for certain customers. Two marketing e-mails have been sent out to each customer. Your users want to find out how many days have passed between the two mailing campaigns and the order dates.

Overview of Exercise Tasks

- Task 1: Modify the Calculation View from the Previous Exercise
- Task 2: Create an Input Parameter to Select the E-mail Date and a Calculated Column to Determine the Number of Days Between the E-mail and the Sale Dates
- Task 3: Modify the Input Parameter to Enable User Entry with Default Value

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

You should also have completed the Implement Variables exercise. If not, you'll find an intermediate version of the view in your *solutions* folder : *CVC_CAMPAIGN_ANALYSIS_00_STAGE1*.

Task 1: Modify the Calculation View from the Previous Exercise

In your *CVC_CAMPAIGN_ANALYSIS* Calculation View, you will delete the *VAR_CUSTOMER* variable and add a new column from the source table to the Output.

The new column will be used later on to calculate the number of days elapsed since the last marketing campaigns.

1. Open your *CVC_CAMPAIGN_ANALYSIS* view.

> Note:
> If needed, you can use the *CVC_CAMPAIGN_ANALYSIS_00_STAGE1* view in the *solutions* folder.

2. Delete the variable *VAR_CUSTOMER*, which will no longer be used because you want to visualize data for all the customers.

a) In the *Parameters* tab of the Semantics node, select the variable and choose *Remove*.

b) Confirm by choosing *Yes*.

3. In the *Join_1* node, add the column *SALES_DATA.SQL_DATE* to the output.
   a) In the *Scenario* tab, select the *Join_1* node.
   b) On the *Mapping* tab, select then right-click the column *SALES_DATA.SQL_DATE* and choose *Add to Output*.
4. Add the new column to the *Aggregation* node.
   a) In the *Scenario*, select the *Aggregation* node.
   b) On the *Mapping* tab, select then right-click the column *SQL_DATE* and select *Add to Output* from the context menu.
5. In the *Semantics* node, rename the column *SQL_DATE* as **`SALE_DATE`**.
   a) In the *Scenario*, select the *Semantics* node.
   b) In the *Columns* tab, change the name and label of the column *SQL_DATE* to **`SALE_DATE`**.
6. In the *Semantics* node, check that the new column has the type *Attribute*.
   a) On the *Columns* tab, check the *Type* cell for the *SALE_DATE* field; it must be *Attribute*.

Task 2: Create an Input Parameter to Select the E-mail Date and a Calculated Column to Determine the Number of Days Between the E-mail and the Sale Dates

Calculate the number of days elapsed between the e-mail marketing campaign and the sales orders.

Note:
There were two marketing campaigns in 2019; one on January 25 and another on February 23.

1. In the semantics, create a new input parameter using the following details:

Hint:
To create an input parameter, select the *Parameters* tab of the *Semantics* node.

Table 4: General properties

Field	Value
Name	INP_EMAIL_DATE
Label	The date when the e-mail was sent
Is Mandatory	[Selected]
Multiple Entries	[NOT selected]
Parameter Type	Static List

Table 5: Parameter Type - Static List

Data Type	Date
List of values	• *Value:* 2019-01-25 *Description:* First e-mail • *Value:* 2019-02-23 *Description:* Second e-mail

Hint:
Use *Add* to define the list of values.

a) On the *Parameters* tab of the *Semantics* node, choose + and then choose *Input Parameter*.

b) Select the new input parameter to open its properties.

c) Enter the details as shown in the General information table.

d) Expand the *Parameter Type - Static List* pane and enter the values specified in the second table. Use the + (Add) button to add each value.

e) Choose *Back* at the top left of the input parameter properties screen.

2. In the *Aggregation* node, add a calculated column to determine the number of days elapsed between the e-mail and the sale dates, using the following details:

Field	Value
General information	
Name	**DAYS_ELAPSED**
Description	**Days between Sale Date and e-mail**
Data Type	*INTEGER*
Semantics	
Column Type	*Attribute*
Expression	
Expression	`DAYS_BETWEEN("SALE_DATE",'$$INP_EMAIL_DATE$$')`

a) On the *Calculated Column* tab of the *Aggregation* node, choose + and then choose *Create Calculated Column*.

b) Select the newly created calculated column to open its properties.

c) Enter the details as shown in the table.

d) Choose *Back*.

3. Deploy the calculation view.
   Check the deployment status after activation.
   a) Choose the *Deploy* icon for the *CVC_CAMPAIGN_ANALYSIS* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.

> Note:
> The following intermediate solution object is available at this stage:
> *solutions → CVC_CAMPAIGN_ANALYSIS_00_STAGE2*

4. Preview the data, and select a date for the parameter *INP_EMAIL_DATE*; for example 2019-01-25.

> Hint:
> If no input parameter shows up in the data preview tab the first time, close this tab and re-execute the data preview.

   a) Right-click the calculation view and choose *Data Preview*.
   b) For your input parameter *INP_EMAIL_DATE*, choose the *From* cell and then the value help button.
   c) Select a date; for example, *2019-01-25*.
   d) Choose *OK*.
   e) Choose *Open Content* and select the *Raw Data* tab .

   Can you explain why the calculation view retrieves NULL data for some customers?

   This is because the join type assigned to the definition of the *Join_1* node is *Left Outer*, the left table being the Customers table. So, the calculation view retrieves all customers, even the ones that do not have sales data.

5. Close the Data Preview.

Result

In the displayed data, check that the number of elapsed days is correctly calculated based on the input parameter that you selected.

Task 3: Modify the Input Parameter to Enable User Entry with Default Value and Calendar Value Help

When you query the view, the input parameter should be set by default to 2019-01-25, and enable the user to enter a different date.

1. Edit the input parameter *INP_EMAIL_DATE* to change its parameter according to the following table:

Field	Value
Parameter Type	Direct
Semantic Type	Date
Default Value(s)	Constant: **2019-01-25**

Keep the other parameters as is.

a) In the *Semantics* node, select the *Parameter* tab and select the input parameter *INP_EMAIL_DATE*.

Note:
The *Parameter* tab is also accessible from the other nodes.

b) Set the values using the data in the table. The semantic type can be found in the *Parameter Type - Direct* section.

c) To define the default value, select the type *Constant* and enter the value **2019-01-25**.

2. Deploy the calculation view.

a) Choose the *Deploy* icon for the *CVC_CAMPAIGN_ANALYSIS* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

3. Preview the calculation view data on the *Raw Data* tab. In the *Variables and Input Parameters* screen, keep the default value or enter a different date. In the displayed data, check that the number of elapsed days is correctly calculated based on the input parameter you defined.

Caution:
If you choose the *Value Help*, a new dialog opens and you can select a date from the displayed calendar.

a) Right-click the calculation view and choose *Data Preview*.

b) For your input parameter *INP_EMAIL_DATE*, keep the default value, enter a different date, or select one from the calendar; for example, **2020-01-01**.

c) Choose *Open Content*.

d) Check that the data displayed corresponds to the date you entered.

4. Close all open tabs in Business Application Studio.

a) Right-click any open tab and choose *Close All*.

Unit 10
Exercise 21

Create a Level Hierarchy

Exercise Objectives

After completing this exercise, you will be able to:

- Define a level hierarchy in a calculation view
- Reuse the hierarchy definition in another calculation view
- Preview the hierarchy data within Business Application Studio

Business Example

You have been asked to build a hierarchy for products that will be used to analyze sales data. You will work on two imported calculation views that you will modify to create the hierarchy.

Overview of Exercise Tasks

- Task 1: Import an Existing Calculation View and Analyze Its Definition
- Task 2: Define a Level Hierarchy for the Products
- Task 3: Preview the Data of the Calculation View in Hierarchy Mode

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import the Calculation Views and Review Their Definition

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Import the following calculation views from your course files folder into your project *exercises* folder:
   - CVD_PRODUCTS_HIER.hdbcalculationview
   - CVC_SALES_HIER.hdbcalculationview

   They are located in the folder *HC300 → Calculation Views Templates*.
3. Open the two calculation views and review their design.

   Note that the CUBE calculation view (Sales) is consuming the DIMENSION calculation view (Products), and no hierarchy is defined in either of them.
4. Deploy the two imported calculation views.

Note:
You can deploy the two calculation views at the same time by deploying the whole database module. If not, you must deploy them in the relevant dependency order; that is, the *CVD_PRODUCTS_HIER* calculation view first, and then, the *CVC_SALES_HIER*.

5. Preview the data of both calculation views.

Task 2: Define a Level Hierarchy for the Products

You will first define a hierarchy Product Group > Product Name in the DIMENSION Calculation View *CVD_PRODUCTS_HIER*, and then reuse the definition in the CUBE Calculation View.

1. Close all open tabs in Business Application Studio.
2. Open the *CVD_PRODUCTS_HIER* Calculation View and create a new level hierarchy with the following properties:

Field	Value
Name	**`PROD_LEV_HIER`**
Label	**`Product Level Hierarchy`**
Level 1 Element	*PROD_GROUP*
Level 2 Element	*PROD_NAME*

   Keep the default properties for the hierarchy levels, in particular the level type *REGULAR* and the sort direction *Ascending*.

3. Deploy the *CVD_PRODUCTS_HIER* calculation view.
   Check the deployment status after activation.
4. Open the *CVC_SALES_HIER* Calculation View, and extract the Products hierarchy definition from the source calculation view *CVD_PRODUCTS_HIER*.

Hint:
In the *Semantics* node, choose *Extract Semantics*.

5. Deploy the *CVC_SALES_HIER* calculation view.
   Check the deployment status after activation.

Task 3: Preview the Data of the Calculation View in Hierarchy Mode

You will use the *Hierarchy* tab, which is natively integrated in Business Application Studio Data Preview.

1. Preview the data of the *CVC_SALES_HIER* calculation view.
2. Switch to the *Hierarchy* tab and explore the Quantity and Amount data hierarchically, based on the *PROD_LEV_HIER* hierarchy.
3. Close all open tabs in Business Application Studio.

Unit 10
Solution 21

Create a Level Hierarchy

Exercise Objectives

After completing this exercise, you will be able to:

- Define a level hierarchy in a calculation view
- Reuse the hierarchy definition in another calculation view
- Preview the hierarchy data within Business Application Studio

Business Example

You have been asked to build a hierarchy for products that will be used to analyze sales data. You will work on two imported calculation views that you will modify to create the hierarchy.

Overview of Exercise Tasks

- Task 1: Import an Existing Calculation View and Analyze Its Definition
- Task 2: Define a Level Hierarchy for the Products
- Task 3: Preview the Data of the Calculation View in Hierarchy Mode

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import the Calculation Views and Review Their Definition

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Import the following calculation views from your course files folder into your project *exercises* folder:
   - CVD_PRODUCTS_HIER.hdbcalculationview
   - CVC_SALES_HIER.hdbcalculationview

   They are located in the folder *HC300 → Calculation Views Templates*.

   a) Right-click the *exercises* folder and choose *Upload*.

   b) Select the file *HC300 → Calculation Views Templates → CVD_PRODUCTS_HIER.hdbcalculationview*, and choose *OK*.

   c) Repeat the previous steps for the other view, *CVC_SALES_HIER.hdbcalculationview*.
3. Open the two calculation views and review their design.

   Note that the CUBE calculation view (Sales) is consuming the DIMENSION calculation view (Products), and no hierarchy is defined in either of them.

a) Select each of the views and review their design.

b) In the *Semantics* node of each view, note that the number displayed together with the *Hierarchies* tab is 0.

4. Deploy the two imported calculation views.

> Note:
> You can deploy the two calculation views at the same time by deploying the whole database module. If not, you must deploy them in the relevant dependency order; that is, the *CVD_PRODUCTS_HIER* calculation view first, and then, the *CVC_SALES_HIER*.

a) Choose the *Deploy* icon for the entire *HC300_A####/db* module.

b) In the *Console* pane, check that the deployment completes successfully.

c) Alternatively, to deploy one view after another, choose the *Deploy* icon for your *CVD_PRODUCTS_HIER* calculation view. After checking that this first deployment was completed successfully, do the same for the *CVC_SALES_HIER* view.

5. Preview the data of both calculation views.

a) Right-click the *CVD_PRODUCTS_HIER* view and choose *Data Preview*.

b) Repeat the previous step with the *CVC_SALES_HIER* view.

Task 2: Define a Level Hierarchy for the Products

You will first define a hierarchy Product Group > Product Name in the DIMENSION Calculation View *CVD_PRODUCTS_HIER*, and then reuse the definition in the CUBE Calculation View.

1. Close all open tabs in Business Application Studio.

a) Right-click any open tab and choose *Close All*.

2. Open the *CVD_PRODUCTS_HIER* Calculation View and create a new level hierarchy with the following properties:

Field	Value
Name	**`PROD_LEV_HIER`**
Label	**`Product Level Hierarchy`**
Level 1 Element	*PROD_GROUP*
Level 2 Element	*PROD_NAME*

Keep the default properties for the hierarchy levels, in particular the level type *REGULAR* and the sort direction *Ascending*.

a) Select the *CVD_PRODUCTS_HIER* calculation view.

b) In the *Semantics Node*, select the *Hierarchies* tab.

c) Choose + to create a new hierarchy and choose *Level Hierarchy*.

d) Select the newly created hierarchy to display its properties.

e) Enter the *Name* and *Label* as provided in the table.

f) In the *Nodes* pane, choose + to create a new level and select the *PROD_GROUP* column as Level 1.

g) Repeat the previous step to add the *PROD_NAME* column as Level 2.

3. Deploy the *CVD_PRODUCTS_HIER* calculation view.

   Check the deployment status after activation.

   a) Choose the Deploy icon for your *CVD_PRODUCTS_HIER* calculation view..

   b) In the *Console* pane, check that the deployment completes successfully.

4. Open the *CVC_SALES_HIER* Calculation View, and extract the Products hierarchy definition from the source calculation view *CVD_PRODUCTS_HIER*.

   Hint:
   In the *Semantics* node, choose *Extract Semantics*.

   a) In the *exercises* folder, select the view *CVC_SALES_HIER*.

   b) In the *Semantics* node, display the *Hierarchies* tab and choose *Extract Semantics*.

   c) In the dialog box, select the hierarchy *PROD_LEV_HIER* (select the corresponding checkbox) and choose *OK*.

   d) Check that the hierarchy has been correctly created based on what is defined in the DIMENSION calculation view.

5. Deploy the *CVC_SALES_HIER* calculation view.

   Check the deployment status after activation.

   a) Choose the Deploy icon for your *CVC_SALES_HIER* calculation view..

   b) In the *Console* pane, check that the deployment completes successfully.

Task 3: Preview the Data of the Calculation View in Hierarchy Mode

You will use the *Hierarchy* tab, which is natively integrated in Business Application Studio Data Preview.

1. Preview the data of the *CVC_SALES_HIER* calculation view.

   a) Right-click the *CVC_SALES_HIER* view and choose *Data Preview*.

   b) Review the *Raw Data* tab.

2. Switch to the *Hierarchy* tab and explore the Quantity and Amount data hierarchically, based on the *PROD_LEV_HIER* hierarchy.

   a) Select the *Hierarchies* tab.

   b) Drag the *PROD_LEV_HIER* item to the *Selected Hierarchies* area.

   c) Drag the *QUANTITY* and *AMOUNT* measures to the *Selected Measures* area.

   d) On the right pane, expand some of the hierarchy nodes to display detailed data down to the PRODUCT level.

3. Close all open tabs in Business Application Studio.

   a) Right-click any open tab and choose *Close All*.

Unit 10
Exercise 22

Create a Parent-Child Hierarchy

Exercise Objectives

After completing this exercise, you will be able to:

- Define a parent-child hierarchy in a calculation view
- Manage root and orphan nodes

Business Example

You have been asked to build an employee hierarchy in order to analyze the number of remaining vacation days at each level of the hierarchy. The source data defining the relationships between employees and their managers is structured as a parent-child hierarchy.

Overview of Exercise Tasks

- Task 1: Implement a Parent-Child Hierarchy
- Task 2: Explore Additional Capabilities Relating to Root Nodes and Orphan Nodes

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Implement a Parent-Child Hierarchy

You will first work with a hierarchy which is "consistent", in the sense that each member has a single parent except the top-level member. So the hierarchy tree can be built without issue.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Create a new calculation view using the data in the following table:

Field	Value
Name	**`CVC_EMPLOYEE_HIER`**
Label	**`Employee Hierarchy`**
Data Category	*CUBE*
With Star Join	[Deselected]

3. Add a Join node *Join_1* to the *Scenario* pane, and add the following tables from the TRAINING schema as data sources to the new node:
   - *HC::EMPLOYEE*
   - *HC::EMPLOYEE_HIERARCHY*

4. Preview the data source *EMPLOYEE_HIEARCHY* directly from the *Join_1* node to check its structure.

   To whom is employee number D100003 reporting?

   _____

   _____

   _____

5. In your *Join_1* node, create a join between the tables using the following properties:

Left Table	HC::EMPLOYEE
Right Table	HC::EMPLOYEE_HIERARCHY
Join Columns	EMPLOYEE.DNUMBER = EMPLOYEE_HIERAR-CHY.CHILD_DNUMBER
Join Type	Left Outer
Cardinality	1..n

6. In the *Join_1* node, add the following columns to output:
   - EMPLOYEE.DNUMBER
   - EMPLOYEE.NAME
   - EMPLOYEE.REMAINDERDAYS
   - EMPLOYEE_HIERARCHY.PARENT_DNUMBER
7. Connect the *Join_1* node to the *Aggregation* node.
8. In the *Aggregation* node, add all columns to the output. Then, check that the *REMAINDERDAYS* columns are treated as a measure, and all other columns as attributes.
9. In the *Hierarchies* tab of the *Semantics* node, create a new parent-child hierarchy with the following properties:

Field	Value
Name	**`EMPLOYEE_HIERARCHY`**
Label	**`Employee Parent-Child Hierarchy`**
Child	*DNUMBER*
Parent	*PARENT_DNUMBER*
Cache	[Deselected]

Note:
While testing the calculation view with different hierarchy options, it is recommended to avoid caching the hierarchy so that the hierarchy is evaluated at each data preview.

Keep the default values for other properties.

Hint:
If the button to create a new hierarchy is grayed out, save and close the calculation view, then reopen it and proceed with the instructions.

10. On the *Columns* tab of the *Semantics* node, define the *NAME* attribute as the *Label Column* for *DNUMBER*.
11. Deploy the *CVC_EMPLOYEE_HIER* calculation view.
    Check the deployment status after activation.
12. Preview the data of the calculation view in a hierarchy.

Note:
It is also possible to display the data in Microsoft Excel, as you did in the previous exercise.

13. Close the data preview tab.

Task 2: Explore Additional Capabilities Relating to Root Nodes and Orphan Nodes

You are now going to work with a data set containing an orphan.

1. Replace the data source *HC::EMPLOYEE_HIERARCHY* with the table *EMPLOYEE_HIERARCHY2*.
   It is a table with exactly the same structure, but a few changes in the data set.
2. Preview the new data source *EMPLOYEE_HIEARCHY2* directly from the *Join_1* node to check the data.

   What is the main difference compared with the initial data set?

   _____

   _____

   _____
3. Deploy the *CVC_EMPLOYEE_HIER* calculation view.
   Check the deployment status after activation.
4. Preview the data of the calculation view in hierarchy mode.

   Where does the employee *D100012* appear ?

   _____

   _____

   _____
5. Change the hierarchy settings so that the orphan node *D100012* becomes the root node.
6. Deploy the *CVC_EMPLOYEE_HIER* calculation view.
   Check the deployment status after activation.
7. Preview the data of the calculation view in hierarchy mode.

Where does the employee *D100012* appear now ?

8. Close all the open tabs in Business Application Studio.

Unit 10
Solution 22

Create a Parent-Child Hierarchy

Exercise Objectives

After completing this exercise, you will be able to:

- Define a parent-child hierarchy in a calculation view
- Manage root and orphan nodes

Business Example

You have been asked to build an employee hierarchy in order to analyze the number of remaining vacation days at each level of the hierarchy. The source data defining the relationships between employees and their managers is structured as a parent-child hierarchy.

Overview of Exercise Tasks

- Task 1: Implement a Parent-Child Hierarchy
- Task 2: Explore Additional Capabilities Relating to Root Nodes and Orphan Nodes

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Implement a Parent-Child Hierarchy

You will first work with a hierarchy which is "consistent", in the sense that each member has a single parent except the top-level member. So the hierarchy tree can be built without issue.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Create a new calculation view using the data in the following table:

Field	Value
Name	**`CVC_EMPLOYEE_HIER`**
Label	**`Employee Hierarchy`**
Data Category	*CUBE*
With Star Join	[Deselected]

   a) In the *Explorer* view, choose *HC300_A####* → *db* → *src* and right-click the *exercises* folder. Choose *New File*.

   b) Enter **`CVC_EMPLOYEE_HIER.hdbcalculationview`** as file name and press *Enter*.

   c) Enter the calculation view label.

d) Choose *Create*.

Result

The calculation view graphical editor opens on the right of your screen.

3. Add a Join node *Join_1* to the *Scenario* pane, and add the following tables from the TRAINING schema as data sources to the new node:

- *HC::EMPLOYEE*
- *HC::EMPLOYEE_HIERARCHY*

a) In the *Scenario* pane, drop a join node from the palette below the top nodes.

b) Select the new *Join_1* node and choose + on the right of the node.

c) In the *Find Data Sources* window, select the *Search* field and enter **`EMPLOYEE`**.

d) In the search results, select the two needed tables by choosing the checkbox on the left of each table.

e) Choose *Finish*.

4. Preview the data source *EMPLOYEE_HIEARCHY* directly from the *Join_1* node to check its structure.

a) In the *Join_1* node, right-click the *EMPLOYEE_HIERARCHY* table and choose *Data Preview*.

b) Observe the name and content of the columns defining the dependencies between employees.

To whom is employee number D100003 reporting?

D100002

5. In your *Join_1* node, create a join between the tables using the following properties:

Left Table	HC::EMPLOYEE
Right Table	HC::EMPLOYEE_HIERARCHY
Join Columns	EMPLOYEE.DNUMBER = EMPLOYEE_HIERAR-CHY.CHILD_DNUMBER
Join Type	Left Outer
Cardinality	1..n

a) In your calculation view scenario pane, select the *Join_1* node and display its details.

b) Drag a connecting line between the *DNUMBER* of the left table, and the column *CHILD_DNUMBER* of the right table.

c) Select the join connector and, in the *PROPERTIES* pane, define the join type and cardinality as shown in the table.

In particular, check that the left table is actually *HC::EMPLOYEE*. If it is not, you can swap the tables with the *Swap Table* button.

6. In the *Join_1* node, add the following columns to output:

- EMPLOYEE.DNUMBER
- EMPLOYEE.NAME
- EMPLOYEE.REMAINDERDAYS
- EMPLOYEE_HIERARCHY.PARENT_DNUMBER

a) In the *Mapping* tab, right-click the above columns from the *Data Sources* area and choose *Add To Output*.

Alternatively, you can drag and drop the columns from the *Data Sources* area to the *Output Columns* area.

7. Connect the *Join_1* node to the *Aggregation* node.
   a) In the *Scenario* pane, select the *Join_1* node.
   b) Drag the arrow icon to the *Aggregation* node.
8. In the *Aggregation* node, add all columns to the output. Then, check that the *REMAINDERDAYS* columns are treated as a measure, and all other columns as attributes.
   a) In the *Scenario* pane, select the *Aggregation* node.
   b) On the *Mapping* tab, right-click the *Join_1* data source and choose *Add to Output*.
   c) Alternatively, you could do that from the *Join_1* node: on the *Mapping* tab, select all the columns in the *Output Columns* area, right-click the selection, and choose *Propagate to Semantics*.
   d) Check that the *REMAINDERDAYS* columns are treated as a measure, and all other columns as attributes.
9. In the *Hierarchies* tab of the *Semantics* node, create a new parent-child hierarchy with the following properties:

Field	Value
Name	**`EMPLOYEE_HIERARCHY`**
Label	**`Employee Parent-Child Hierarchy`**
Child	*DNUMBER*
Parent	*PARENT_DNUMBER*
Cache	[Deselected]

Note:
While testing the calculation view with different hierarchy options, it is recommended to avoid caching the hierarchy so that the hierarchy is evaluated at each data preview.

Keep the default values for other properties.

Hint:
If the button to create a new hierarchy is grayed out, save and close the calculation view, then reopen it and proceed with the instructions.

a) Select the *Hierarchies* tab of the *Semantics* node.

b) Choose + and select *Parent Child Hierarchy*.

c) Select the new hierarchy to display its properties.

d) Enter the *Name* and *Label* as shown in the table.

e) In the *Nodes* area, choose + and assign the *Child* and *Parent* columns as per the table.

f) In the *Properties* area, deselect the *Cache* checkbox.

g) Choose *Back* at the top left of the hierarchy properties screen.

10. On the *Columns* tab of the *Semantics* node, define the *NAME* attribute as the *Label Column* for *DNUMBER*.

a) Select the *Columns* tab.

b) For the *DNUMBER* row, select the *NAME* attribute from the drop-down menu of *Label Column*.

11. Deploy the *CVC_EMPLOYEE_HIER* calculation view.

Check the deployment status after activation.

a) Choose the *Deploy* icon for the *CVC_EMPLOYEE_HIER* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

12. Preview the data of the calculation view in a hierarchy.

Note:
It is also possible to display the data in Microsoft Excel, as you did in the previous exercise.

a) In the *Explorer* view, right-click the *CVC_EMPLOYEE_HIER* and choose *Data Preview*.

b) Display the *Hierarchy* tab and drag the *EMPLOYEE_HIERARCHY* to the *Selected Hierarchy* pane and the *REMAINDERDAYS* measure to the *Selected Measures* pane.

c) Explore the hierarchy. Observe that the total number of remainder days for a given parent node is composed of the parent's own value (not displayed specifically in this view) plus the total of all its children.

Note:
You can preview the data of the *HC::EMPLOYEES* if you want to check the number of remainder days for employees who are also managers (hierarchy nodes).

13. Close the data preview tab.

Task 2: Explore Additional Capabilities Relating to Root Nodes and Orphan Nodes

You are now going to work with a data set containing an orphan.

1. Replace the data source *HC::EMPLOYEE_HIERARCHY* with the table *EMPLOYEE_HIERARCHY2*.

   It is a table with exactly the same structure, but a few changes in the data set.

   a) In the *Join_1* node of calculation view *CVC_EMPLOYEE_HIER*, right-click the *HC::EMPLOYEE_HIERARCHY* data source and choose *Replace with Data Source*.

   b) In the *Find Data sources* window, enter **`hiera`** in the search field.

   c) Select the *EMPLOYEE_HIERARCHY2* table and choose *Next*.

   d) Choose *Finish*.

   Note:
   Because the table structure (and column names) are exactly the same, there is no modification to make at the *Manage Mappings* and *Manage Joins* steps.

2. Preview the new data source *EMPLOYEE_HIEARCHY2* directly from the *Join_1* node to check the data.

   a) In the *Join_1* node, right-click the HC::*EMPLOYEE_HIERARCHY2* data source and choose *Data Preview*.

   b) Observe the differences with the previous data set (if needed, it is already opened in a tab of SAP Business Application Studio).

   What is the main difference compared with the initial data set?

   The member *D100012* has no parent.

3. Deploy the *CVC_EMPLOYEE_HIER* calculation view.

   Check the deployment status after activation.

   a) Choose the *Deploy* icon for the *CVC_EMPLOYEE_HIER* calculation view.

   b) In the *Console* pane, check that the deployment completes successfully.

4. Preview the data of the calculation view in hierarchy mode.

   a) In the *Explorer* view, right-click the *CVC_EMPLOYEE_HIER* and choose *Data Preview*.

   b) Display the *Hierarchy* tab and drag the *EMPLOYEE_HIERARCHY* to the *Selected Hierarchy* pane and the *REMAINDERDAYS* measure to the *Selected Measures* pane.

   c) Explore the hierarchy.

   Where does the employee *D100012* appear ?

   Outside of the hierarchy.

5. Change the hierarchy settings so that the orphan node *D100012* becomes the root node.

a) Close the preview tab.

b) In the hierarchy definition, in the *Nodes* section, enter **`D100012`** as root node.

c) In the Properties section, select **Add Root node** for the *Root Node Visibility*.

6. Deploy the *CVC_EMPLOYEE_HIER* calculation view.
   Check the deployment status after activation.

   a) Choose the *Deploy* icon for the *CVC_EMPLOYEE_HIER* calculation view.

   b) In the *Console* pane, check that the deployment completes successfully.

7. Preview the data of the calculation view in hierarchy mode.

   a) In the *Explorer* view, right-click the *CVC_EMPLOYEE_HIER* and choose *Data Preview*.

   b) Display the *Hierarchy* tab and drag the *EMPLOYEE_HIERARCHY* to the *Selected Hierarchy* pane and the *REMAINDERDAYS* measure to the *Selected Measures* pane.

   c) Explore the hierarchy.

   Where does the employee *D100012* appear now ?

   As the root node.

8. Close all the open tabs in Business Application Studio.

   a) Right-click any open tab and choose *Close All*.

Unit 11
Exercise 23

Work with the SQL Console

Exercise Objectives

After completing this exercise, you will be able to:

- Use the SQL Console to query data from a database connection and a standard catalog schema.
- Use the SQL Console to query data from an HDI container connection.

Business Example

You would like to become familiar with the SQL Console by writing a few simple SQL statements, first on a classic database connection and then against a container.

Prerequisites

- Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.
- A connection to the HANA Cloud database (`HC300_DB (A####) (SSO enabled)`) should have been setup.

1. Open an SQL Console for your database connection *HC300_DB (A####) (SSO enabled)*.
2. Execute a query to read all data from table *ORDERS* in the schema *TRAINING*.
3. Execute a query to list only the columns *STORE*, *PRODUCT*, *QUANTITY* filtered by product *Mouse*.
4. Automate the generation of an SQL *SELECT* statement by using the provided context menu option against the table *ORDERS* which is located in the *TRAINING* schema.
5. Open an SQL Console for your HDI container.
6. Execute a query to read all data from table *Employees*.
7. Close all open tabs in Business Application Studio and come back to the *Explorer*.

Unit 11
Solution 23

Work with the SQL Console

Exercise Objectives

After completing this exercise, you will be able to:

- Use the SQL Console to query data from a database connection and a standard catalog schema.
- Use the SQL Console to query data from an HDI container connection.

Business Example

You would like to become familiar with the SQL Console by writing a few simple SQL statements, first on a classic database connection and then against a container.

Prerequisites

- Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.
- A connection to the HANA Cloud database (`HC300_DB (A####) (SSO enabled)`) should have been setup.

1. Open an SQL Console for your database connection *HC300_DB (A####) (SSO enabled)*.
   a) Switch to the *SAP HANA Database Explorer* view.
   b) Highlight your SAP HANA classic database connection *HC300_DB (A####) (SSO enabled)*, and then choose *Open SAP HANA SQL Console* on the right of the name.
2. Execute a query to read all data from table *ORDERS* in the schema *TRAINING*.
   a) In the empty tab, carefully enter the code:

```
SELECT * FROM "TRAINING"."ORDERS";
```

   b) Press *Run (F8)*.
3. Execute a query to list only the columns *STORE, PRODUCT, QUANTITY* filtered by product *Mouse*.
   a) Adjust the code as follows:

```
SELECT STORE, PRODUCT, QUANTITY FROM "TRAINING"."ORDERS" where PRODUCT
= 'Mouse';
```

   b) Press *Run (F8)*.
4. Automate the generation of an SQL *SELECT* statement by using the provided context menu option against the table *ORDERS* which is located in the *TRAINING* schema.
   a) In the top left pane, navigate to *HC300_DB (A####) (SSO enabled)* → *Catalog* → *Tables*.
   b) In the *Catalog Browser* pane, choose the *Apply filter* icon and select **Filter objects by schema**. A list of schemas should appear. Select the *TRAINING* schema. Choose *OK*.

Result

You should now see the tables listed in the lower pane.

c) In the lower left pane, right-click on the table *ORDERS* and choose the context menu option *Generate SELECT statement*.

d) Press **F8** or choose *Run* on the toolbar to execute and display the data.

Notice the other options that can be used to generate common SQL statements for the chosen table, such as CREATE.

5. Open an SQL Console for your HDI container.

a) In the *Database List* sub-pane, highlight your HDI container and then choose *Open SAP HANA SQL Console* .

Caution:
There are two different icons to open an SQL console connected to an HDI container. Please make sure to use the right one (*Open SAP HANA SQL Console*).

The left one, *Open SAP HANA SQL Console (Admin)*, is used to open an SQL console as user *container_schema_name..._DT*, connected to the schema *<container_schema_name>#DI*, which is the HDI container's API schema. So this is for HDI content administration and would not provide the necessary privileges for this exercise.

b) In the new console tab, check that the *Current Schema* corresponds to your HDI container schema name, that is, *HC300_A####_HDI_DB_1*.

6. Execute a query to read all data from table *Employees*.

a) In the empty tab, enter the code:

```
SELECT * FROM "HC::Employees"
```

Notice the use of the namespace *HC* that prefixes the table name (this topic is covered later). If you do not include the namespace, the table is not found. In addition, when working with containers, notice that the schema does not need to be specified because the schema is already known to the container (the schema *"belongs"* to the container and is unique).

b) Press *Run (F8)*.

7. Close all open tabs in Business Application Studio and come back to the *Explorer*.

a) Right-click any open tab and choose *Close All*.

b) Choose the *Explorer* icon.

Unit 11
Exercise 24

Read a Modeled Hierarchy using SQL

Exercise Objectives

After completing this exercise, you will be able to:

- Expose a hierarchy to SQL in a CUBE calculation view with star join.
- Query hierarchies defined in calculation views, using SQL statements

Business Example

You would like to understand how to access a hierarchy that has been defined in a calculation view, using SQL statements.

Overview of Exercise Tasks

- Task 1: Review the Dimension Calculation View that Defines the Shared Hierarchy
- Task 2: Review the Cube Calculation View that Exposes the Shared Hierarchy to SQL
- Task 3 : Create SQL to Query the Shared Hierarchy

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Review the Dimension Calculation View that Defines the Shared Hierarchy

Review the calculation view of type dimension, which exposes master data attributes of employees and uses a parent-child hierarchy to organize the employee data. You will access this hierarchy using SQL.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *resources* folder, open the Calculation View **`CVD_EMPLOYEE_HIER`** and examine the definition. Pay attention to the hierarchy that is defined in the calculation view.

Task 2: Review the Cube Calculation View that Exposes the Shared Hierarchy to SQL

Review the calculation view *CVCS_EMPLOYEE_DATA*. This calculation view joins employee data with the hierarchy dimension, but most importantly, also enables SQL access to the hierarchy.

1. In your *resources* folder, open the Calculation View *CVCS_EMPLOYEE_DATA* and examine the definition. Pay particular attention to the settings that enable SQL access to the shared hierarchy.
2. Preview the *CVCS_EMPLOYEE_DATA* calculation view with a hierarchical display.

Task 3: Create SQL to Query the Shared Hierarchy

You will now discover how to write the SQL that is used to query the calculation view, and in particular filter the results using the hierarchy.

1. Open a *SQL Console* for your HDI container.
2. On your device, from your *HC300* exercise files, open the file *Prepared Code → SQL for reading hierarchy structure.txt*, and copy and paste its entire contents to the SQL console.

```
-- First Part : Select data for all employees, independently of
hierarchy

SELECT
"DNUMBER",
"REMAINDERDAYS"
FROM "HC::CVCS_EMPLOYEE_DATA";

-- Second Part : Filter data on a parent node

SELECT
"DNUMBER",
"REMAINDERDAYS"
FROM "HC::CVCS_EMPLOYEE_DATA"
WHERE "PARENT_DNUMBER" = 'D100001';

-- Third Part : Filter data on a hierarchy node

SELECT
"DNUMBER",
"REMAINDERDAYS"
FROM "HC::CVCS_EMPLOYEE_DATA"
WHERE "H_EMPnode" = 'D100001';

--Fourth Part : Select the hierarchy information

SELECT
"H_EMPnode",
SUM("REMAINDERDAYS")
FROM "HC::CVCS_EMPLOYEE_DATA"
GROUP BY "H_EMPnode";
```

3. Review the SQL code to study how the hierarchy is read.
4. Execute the SQL statements and review the results.

   What is the second SELECT statement doing ?

   _____
   _____
   _____

   What is the third SELECT statement doing compared to the second statement ?

   _____
   _____
   _____

   What is the fourth SELECT statement doing ?

   _____
   _____
   _____

5. Close all open tabs in Business Application Studio and come back to the *Explorer* view.

Unit 11
Solution 24

Read a Modeled Hierarchy using SQL

Exercise Objectives

After completing this exercise, you will be able to:

- Expose a hierarchy to SQL in a CUBE calculation view with star join.
- Query hierarchies defined in calculation views, using SQL statements

Business Example

You would like to understand how to access a hierarchy that has been defined in a calculation view, using SQL statements.

Overview of Exercise Tasks

- Task 1: Review the Dimension Calculation View that Defines the Shared Hierarchy
- Task 2: Review the Cube Calculation View that Exposes the Shared Hierarchy to SQL
- Task 3 : Create SQL to Query the Shared Hierarchy

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Review the Dimension Calculation View that Defines the Shared Hierarchy
Review the calculation view of type dimension, which exposes master data attributes of employees and uses a parent-child hierarchy to organize the employee data. You will access this hierarchy using SQL.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *resources* folder, open the Calculation View **`CVD_EMPLOYEE_HIER`** and examine the definition. Pay attention to the hierarchy that is defined in the calculation view.
   a) If not already done, switch to the *Explorer* view of your *HC300* project.
   b) In your *resources* sub-folder, select the calculation view **`CVD_EMPLOYEE_HIER`**.
   c) Review the data source *HC::EMPLOYEE_HIERARCHY* by double-clicking the *Projection* node, so that you see on which columns the hierarchy is built.
   d) Review the hierarchy by selecting the *semantics* node and then selecting the *Hierarchies* tab and then select the hierarchy *H_EMP* and expand the nodes section. In particular, note how the hierarchy is a parent-child type (observe the hierarchy icon) and is based on the *CHILD_DNUMBER* and *PARENT_DNUMBER* attributes.

Task 2: Review the Cube Calculation View that Exposes the Shared Hierarchy to SQL
Review the calculation view *CVCS_EMPLOYEE_DATA*. This calculation view joins employee data with the hierarchy dimension, but most importantly, also enables SQL access to the hierarchy.

1. In your *resources* folder, open the Calculation View *CVCS_EMPLOYEE_DATA* and examine the definition. Pay particular attention to the settings that enable SQL access to the shared hierarchy.
   a) In your *resources* folder, select the Calculation View **CVCS_EMPLOYEE_DATA**.
   b) Review the *star join* node. Notice how the employee data and dimension are combined by selecting the *Join Definition* tab.
   c) Double-click the *Semantics* node and choose the tab *View Properties*. In the *General* sub-tab, you will see that *Enable Hierarchies for SQL Access* is already selected.
   d) Review the settings for the shared hierarchy by selecting the *Hierarchies* tab. Then, under the **Shared Hierarchies** sub–tab, select the hierarchy *H_EMP* and expand the section *SQL ACCESS*. Notice the name *H_EMPNode* that is generated from the hierarchy name + the fixed word 'node'. This is the name you need to refer to when you want to reference the hierarchy nodes in an SQL expression using *SELECT*, *WHERE*, *GROUP BY*, and so on. You can change the name here if you prefer to something more meaningful.
2. Preview the *CVCS_EMPLOYEE_DATA* calculation view with a hierarchical display.
   a) In the *Explorer* view, right-click the calculation view *CVCS_EMPLOYEE_DATA* and choose *Data Preview*.
   b) Display the *Hierarchy* tab and drag the *H_EMP* to the *Selected Hierarchy* pane and the *REMAINDERDAYS* measure to the *Selected Measures* pane.
   c) Explore the hierarchy. Observe that the total number of remainder days for a given parent node is composed of the parent's own value (not displayed specifically in this view) plus the total of all its children.

Task 3: Create SQL to Query the Shared Hierarchy

You will now discover how to write the SQL that is used to query the calculation view, and in particular filter the results using the hierarchy.

1. Open a *SQL Console* for your HDI container.
   a) Switch to the *SAP HANA Database Explorer* view.
   b) Highlight your database container and then choose *Open SAP HANA SQL Console* in the toolbar (last icon on the right).
2. On your device, from your *HC300* exercise files, open the file *Prepared Code → SQL for reading hierarchy structure.txt*, and copy and paste its entire contents to the SQL console.

```
-- First Part : Select data for all employees, independently of
hierarchy

SELECT
"DNUMBER",
"REMAINDERDAYS"
FROM "HC::CVCS_EMPLOYEE_DATA";

-- Second Part : Filter data on a parent node

SELECT
"DNUMBER",
"REMAINDERDAYS"
FROM "HC::CVCS_EMPLOYEE_DATA"
```

```
WHERE "PARENT_DNUMBER" = 'D100001';

-- Third Part : Filter data on a hierarchy node

SELECT
"DNUMBER",
"REMAINDERDAYS"
FROM "HC::CVCS_EMPLOYEE_DATA"
WHERE "H_EMPnode" = 'D100001';

--Fourth Part : Select the hierarchy information

SELECT
"H_EMPnode",
SUM("REMAINDERDAYS")
FROM "HC::CVCS_EMPLOYEE_DATA"
GROUP BY "H_EMPnode";
```

a) Double-click the file *HC300 → Prepared Code → SQL for reading hierarchy structure.txt*.

b) Select all the content of the file (press **Ctrl+A**) and copy it to the clipboard (press **Ctrl+C**).

c) To paste the content in the SQL Console, press **Ctrl+V** or right-click anywhere inside the SQL Console and choose *Paste*.

3. Review the SQL code to study how the hierarchy is read.

a) The key element in the SQL code is the use of the column name *H_EMPnode*, which is the generated column that represents the nodes at all levels of the hierarchy.

4. Execute the SQL statements and review the results.

What is the second SELECT statement doing ?

It is filtering the employees having D100001 as parent. There are five of them.

What is the third SELECT statement doing compared to the second statement ?

It is filtering the employees on the D100001 node, meaning that it takes all employees that have D100001 as a parent or grand parent etc. It also returns the D100001 employee.

What is the fourth SELECT statement doing ?

It is returning the sum of remainder days for each node of the hierarchy. Thus summing the remainder days of parent and children.

a) To execute the script, press **F8** or choose *Run* on the toolbar.

Hint:
To execute a single statement, you can either select it and press **F8**, or put your cursor inside and press **F9** *(Run Statement)*.

5. Close all open tabs in Business Application Studio and come back to the *Explorer* view.
   a) Right-click any open tab and choose *Close All*.
   b) Choose the *Explorer* icon.

Unit 11
Exercise 25

Work with SQLScript

Exercise Objectives

After completing this exercise, you will be able to:

- Use the SQL Console to execute an anonymous block.
- Use SQL Script statements such as variable declaration, loops and intermediate variables.

Business Example

SQLScript is usually written in the design-time artifacts: function and procedures. Unlike plain SQL, you would not normally write and execute SQLScript directly in the SQL Console of SAP Business Application Studio. However, for testing purposes it is possible to write and execute SQLScript directly in the SQL Console of SAP Business Application Studio without the use of a procedure or function wrapper. You use what is known as an **anonymous block**. This essentially is a DO - BEGIN - END structure. Your SQLScript must appear inside the BEGIN and END.

Note:
In this exercise, when values include `A####`, replace the characters with the user code assigned to you (one letter and four digits).

Prerequisites

- Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.
- Your `A####` user should have been created.
- A connection to the HANA Cloud database (`HC300_DB (A####) (SSO enabled)`) with your user *A####* should have been set up.

1. If not already there, switch to the *SAP HANA Database Explorer* and open an *SQL Console* in the database connection *HC300_DB (A####) (SSO enabled)*.
2. From your *HC300* resources folder, open the file *Prepared Code → SQL Script for Anonymous Block.txt*, and copy and paste its entire contents to the SQL console.
3. Review the SQLScript to study how we are using syntax that is not standard SQL but is part of the extensions provided with SQLScript.
4. Execute the SQLScript and review the results.
5. Close all open tabs in Business Application Studio and come back to the *Explorer*.

Unit 11
Solution 25

Work with SQLScript

Exercise Objectives

After completing this exercise, you will be able to:

- Use the SQL Console to execute an anonymous block.
- Use SQL Script statements such as variable declaration, loops and intermediate variables.

Business Example

SQLScript is usually written in the design-time artifacts: function and procedures. Unlike plain SQL, you would not normally write and execute SQLScript directly in the SQL Console of SAP Business Application Studio. However, for testing purposes it is possible to write and execute SQLScript directly in the SQL Console of SAP Business Application Studio without the use of a procedure or function wrapper. You use what is known as an **anonymous block**. This essentially is a DO - BEGIN - END structure. Your SQLScript must appear inside the BEGIN and END.

Note:
In this exercise, when values include `A####`, replace the characters with the user code assigned to you (one letter and four digits).

Prerequisites

- Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.
- Your `A####` user should have been created.
- A connection to the HANA Cloud database (`HC300_DB (A####) (SSO enabled)`) with your user *A####* should have been set up.

1. If not already there, switch to the *SAP HANA Database Explorer* and open an *SQL Console* in the database connection *HC300_DB (A####) (SSO enabled)*.
   a) Highlight your database connection *HC300_DB (A####) (SSO enabled)* and then choose *Open SAP HANA SQL Console* on its right.
2. From your *HC300* resources folder, open the file *Prepared Code → SQL Script for Anonymous Block.txt*, and copy and paste its entire contents to the SQL console.
   a) Double-click the file *HC300 → Prepared Code → SQL Script for Anonymous Block.txt*.
   b) Select all the content of the file (press **`Ctrl+A`**) and copy it to the clipboard (press **`Ctrl+C`**).

c) To paste the content in the SQL Console, press **Ctrl+V** or right-click anywhere inside the SQL Console and choose *Paste*.

```
-- use this next statement only if you want to re-run the script
-- DROP TABLE TAB;

-- notice how we are declaring a variable and also using a loop
DO
BEGIN
    DECLARE I INTEGER;
    CREATE ROW TABLE TAB (I INTEGER);
    FOR I IN 1..10 DO
        INSERT INTO TAB VALUES (:I);
    END FOR;
END;

-- let's see if the loop worked
SELECT * FROM TAB;

-- notice how we are filling variables to store interim results
DO
BEGIN
    T1 = SELECT I, 10 AS J FROM TAB;
    T2 = SELECT I, 20 AS K FROM TAB;
    T3 = SELECT J, K FROM :T1, :T2 WHERE :T1.I = :T2.I;
    SELECT * FROM :T3;
END;
```

3. Review the SQLScript to study how we are using syntax that is not standard SQL but is part of the extensions provided with SQLScript.
   a) Observe three key elements of SQLScript: a declared variable, a loop and we are filling and querying undeclared table variables with interim results.
4. Execute the SQLScript and review the results.
   a) Press **F8** or choose *Run* on the toolbar and review each result tab.
5. Close all open tabs in Business Application Studio and come back to the *Explorer*.
   a) Right-click any open tab and choose *Close All*.
   b) Choose the *Explorer* icon.

Unit 11
Exercise 26

Define a Data Source using a Table Function

Exercise Objectives

After completing this exercise, you will be able to:

- Create a table function
- Consume the table function as a data source in a DIMENSION calculation view
- Pass the user entry to an input parameter of a top level calculation view to the input parameter of the table function

Business Example

Your calculation view needs to find the name of an employee, even when the end user searching for that employee types the surname of the employee incorrectly. You decide to implement a table function to use the fault-tolerant search functionality.

You then use an input parameter to prompt the user for the employee name and you expect them to only enter partial names. You map the users' input to the table function's input to perform the search. The table function searches the person using the built-in fuzzy text search of SAP HANA.

Overview of Exercise Tasks

- Task 1: Create the Table Function
- Task 2: Use the Table Function in a Calculation View Type Dimension Using a Constant Mapping
- Task 3: Use the Table Function in a Calculation View Type Dimension Using an Input Parameter Mapping

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create the Table Function

You want to find the name of an employee, even when you type the surname of the employee incorrectly. You decide to achieve this by defining a table function.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the *Explorer* view of SAP Business Application Studio, create a new function **`TF_FUZZY_EMPLOYEES`** under the node *exercises*.
3. Copy and paste the function code from the prepared text file that you can find it in your *HC300* folder. The file you need is *Prepared Code → SQL for Table Function.txt*.

```
FUNCTION "HC::TF_FUZZY_EMPLOYEES" (LastNameFilter NVARCHAR(40))
    RETURNS table
```

```
        (
       "FIRST_NAME" NVARCHAR(40),
       "LAST_NAME" NVARCHAR(40),
       "SEX" NVARCHAR(1),
       "LANGUAGE" NVARCHAR(1),
       "EMAIL_ADDRESS" NVARCHAR(255)
        )
    LANGUAGE SQLSCRIPT
    SQL SECURITY INVOKER AS
BEGIN

RETURN

select
"FIRST_NAME",
"LAST_NAME",
"SEX",
"LANGUAGE",
"EMAIL_ADDRESS"
 from "HC::SNWD_EMPLOYEES"
 where contains("LAST_NAME", :LastNameFilter, FUZZY(0.5));

END
```

Note:
This code has *CONTAINS* and *FUZZY* statements. This feature enables you to find the names of employees even if they are misspelled. You can learn more about these features in the course SAP HANA Advanced Modeling.

4. Deploy your table function.

Task 2: Use the Table Function in a Calculation View Type Dimension Using a Constant Mapping

1. Create a calculation view of type DIMENSION in your *exercises* folder with the name and label **`CVD_FIND_EMPLOYEES`**.
2. Add a new *Table Function* node and assign the table function *TF_FUZZY_EMPLOYEES* to it.
3. Map the input parameter *LASTNAMEFILTER* of the table function to a constant value **`SMIHT`**, which represents a misspelled employee name.
4. Connect the *Table Function* node to the *Projection* node.
5. Map all incoming columns to the output columns of the *Projection* node.
6. Deploy your calculation view.
7. Test the table function by using the *Data Preview* function of the calculation view.

Note:
An intermediate solution object is available at this stage:

solutions → CVD_FIND_EMPLOYEES_00_STAGE1.hdbcalculationview.

Task 3: Use the Table Function in a Calculation View Type Dimension Using an Input Parameter Mapping

1. From the calculation view *CVD_FIND_EMPLOYEES* remove the mapping between the constant and the function input parameter.
2. Create a new input parameter with the label and name *Enter_Name*, a data type *NVARCHAR*, and length *40*.
3. Map the new input parameter *Enter_Name* to the input parameter *LASTNAMEFILTER* of the table function.
4. Deploy your calculation view.
5. Test the table function by using the *Data Preview* function of the calculation view. At the prompt, enter the deliberately misspelled name **`SMIHT`**.
6. Close all the open tabs in Business Application Studio.

Unit 11
Solution 26

Define a Data Source using a Table Function

Exercise Objectives

After completing this exercise, you will be able to:

- Create a table function
- Consume the table function as a data source in a DIMENSION calculation view
- Pass the user entry to an input parameter of a top level calculation view to the input parameter of the table function

Business Example

Your calculation view needs to find the name of an employee, even when the end user searching for that employee types the surname of the employee incorrectly. You decide to implement a table function to use the fault-tolerant search functionality.

You then use an input parameter to prompt the user for the employee name and you expect them to only enter partial names. You map the users' input to the table function's input to perform the search. The table function searches the person using the built-in fuzzy text search of SAP HANA.

Overview of Exercise Tasks

- Task 1: Create the Table Function
- Task 2: Use the Table Function in a Calculation View Type Dimension Using a Constant Mapping
- Task 3: Use the Table Function in a Calculation View Type Dimension Using an Input Parameter Mapping

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create the Table Function

You want to find the name of an employee, even when you type the surname of the employee incorrectly. You decide to achieve this by defining a table function.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the *Explorer* view of SAP Business Application Studio, create a new function **`TF_FUZZY_EMPLOYEES`** under the node *exercises*.
   a) Choose *View* → *Command Palette*
   b) Enter **`Artifact`** and choose the SAP HANA: Create SAP HANA Database Artifact option.

c) Choose the *Browse* button and choose the */home/user/projects/HC300_A####/db/src/models/exercises/* folder and choose *OK*.

d) For the artifact type, choose **Function (hdbfunction)**.

e) Enter **`TF_FUZZY_EMPLOYEES`** as function name and choose *Create*.

3. Copy and paste the function code from the prepared text file that you can find it in your *HC300* folder. The file you need is *Prepared Code → SQL for Table Function.txt*.

```
FUNCTION "HC::TF_FUZZY_EMPLOYEES" (LastNameFilter NVARCHAR(40))
    RETURNS table
        (
       "FIRST_NAME" NVARCHAR(40),
       "LAST_NAME" NVARCHAR(40),
       "SEX" NVARCHAR(1),
       "LANGUAGE" NVARCHAR(1),
       "EMAIL_ADDRESS" NVARCHAR(255)
        )
    LANGUAGE SQLSCRIPT
    SQL SECURITY INVOKER AS
BEGIN

RETURN

select
"FIRST_NAME",
"LAST_NAME",
"SEX",
"LANGUAGE",
"EMAIL_ADDRESS"
 from "HC::SNWD_EMPLOYEES"
 where contains("LAST_NAME", :LastNameFilter, FUZZY(0.5));

END
```

Note:
This code has *CONTAINS* and *FUZZY* statements. This feature enables you to find the names of employees even if they are misspelled. You can learn more about these features in the course SAP HANA Advanced Modeling.

a) First, completely remove the code that was generated in the function, so the file is completely empty.

b) Open the prepared text file which contains the SQL and copy and paste the entire contents to the new function.

4. Deploy your table function.

a) Choose the *Deploy* icon for the *TF_FUZZY_EMPLOYEES* function.

b) In the *Console* pane, check that the deployment completes successfully.

Task 2: Use the Table Function in a Calculation View Type Dimension Using a Constant Mapping

1. Create a calculation view of type DIMENSION in your *exercises* folder with the name and label **`CVD_FIND_EMPLOYEES`**.

a) Right-click your *exercises* folder and choose *New File*.

b) Enter **`CVD_FIND_EMPLOYEES.hdbcalculationview`** as file name and press *Enter*.

c) Enter the calculation view label and change the *Data Category* to DIMENSION.

d) Choose *Create*.

2. Add a new *Table Function* node and assign the table function *TF_FUZZY_EMPLOYEES* to it.
   a) From the toolbar, select the icon *Create Table Function* and then click in the empty space at the bottom of the flow, directly underneath the *Projection* node.

   b) In the *Table Function* node, choose *Add Table Function* (looks like a grid with a +). Be careful not to choose the standard 'add' icon that also uses a + symbol)

   c) In the *Search* field, enter **`Fuzzy`** and you will see only one entry appears.

   d) Select the table function *HC::TF_FUZZY_EMPLOYEES* and choose *Finish*.
   Be careful not to choose the *_00* version of the function, because this is the solution version and not the one you created.

3. Map the input parameter *LASTNAMEFILTER* of the table function to a constant value **`SMIHT`**, which represents a misspelled employee name.
   a) Double-click the *Table Function* node and choose the *Input Mapping* tab.

   b) In the toolbar, choose *Create Constant* (the last icon on the right)

   c) In the *Value* field, enter **`SMIHT`** and choose *OK*.

   d) Drag a line from the constant value on the left side to the *LASTNAMEFILTER* input parameter of the table function which is found on the right side so they are now mapped.

4. Connect the *Table Function* node to the *Projection* node.
   a) Drag a line from the arrow connector icon in the *Table Function* node to the bottom of the *Projection* node.

5. Map all incoming columns to the output columns of the *Projection* node.
   a) Select the *Projection* node and make sure that the *Mapping* tab is selected.

   b) Under *Data Sources*, right-click the header *TableFunction_1* and choose *Add To Output*, so that all columns appear under *Output Columns*.

6. Deploy your calculation view.
   a) Choose the *Deploy* icon for the *CVD_FIND_EMPLOYEES* calculation view.

   b) In the *Console* pane, check that the deployment completes successfully.

7. Test the table function by using the *Data Preview* function of the calculation view.
   a) Right-click your calculation view in the folder structure under *exercises* and select the *Data Preview* option. You should see results which include records where the last name closely matches the constant value.

Note:
An intermediate solution object is available at this stage:

solutions → CVD_FIND_EMPLOYEES_00_STAGE1.hdbcalculationview.

Task 3: Use the Table Function in a Calculation View Type Dimension Using an Input Parameter Mapping

1. From the calculation view *CVD_FIND_EMPLOYEES* remove the mapping between the constant and the function input parameter.
   a) Select the *Table Function* node and select the *Input Mapping* tab.
   b) Select the line that maps the constant to the function parameter and right-click to select *Remove* Mapping
   c) Highlight the constant value and choose *Remove* in the toolbar.
      You might find that the constant does not disappear. If this is the case, simply save, then close and re-open the calculation view to refresh the screen. The mapping should then disappear.
2. Create a new input parameter with the label and name *Enter_Name*, a data type *NVARCHAR*, and length *40*.
   a) With the *Table Function* node still selected click the *Parameters* tab.
   b) Choose +, and from the drop-down list select *Input parameter.*.
   c) Select the new input parameter and enter the name and label *Enter_Name*.
   d) Expand the section *Parameter Type — Direct* and from the drop-down list *Data Type*, choose *NVARCHAR.*.
   e) In the *Length* field, enter **`40`**.
   f) Use the < *Back* button to return to the main screen.
3. Map the new input parameter *Enter_Name* to the input parameter *LASTNAMEFILTER* of the table function.
   a) Still in the *Table Function* node, select the *Input Mapping* tab.
   b) If the parameter is still named IP_1, close and open the calculation view again.
   c) Drag a line from the input parameter *Enter_Name* on the left side to the *LASTNAMEFILTER* on the right side in order to map them.
4. Deploy your calculation view.
   a) Choose the *Deploy* icon for the *CVD_FIND_EMPLOYEES* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
5. Test the table function by using the *Data Preview* function of the calculation view. At the prompt, enter the deliberately misspelled name **`SMIHT`**.
   a) Right-click your calculation view and select the *Data Preview* option. After a few moments you will see a prompt for the input parameter *Enter_Name*, where you will enter the misspelled last name.
   b) In the *From* column, type **`SMIHT`**.

Note:
Do not try to display the *Value Help* dialog box. You would get a warning, because the input parameter is not designed to generate a list of possible values.

c) Choose *Open Content* to execute the data preview using your input value.

Result

You can see that SAP HANA still managed to find two employees with the surname *Smith*, despite you typing the surname incorrectly.

6. Close all the open tabs in Business Application Studio.

a) Right-click any open tab and choose *Close All*.

Unit 11
Exercise 27

Derive an Input Parameter Value using a Scalar Function

Exercise Objectives

After completing this exercise, you will be able to:

- Create calculation views that include input parameters where the value is derived from a scalar function.

Business Example

You would like to learn how to use SQLScript to generate results for an input parameter.

Overview of Exercise Tasks

- Task 1: Create a new scalar function to identify the top customer based on sales quantity
- Task 2: Create a calculation view that includes an input parameter that consumes your scalar function

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a new scalar function to identify the top customer based on sales quantity

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the *EXPLORER* view, create a new scalar function **`SF_GET_BEST_CUSTOMER`** in the folder *exercises*.
3. Create the SQLScript of your scalar function using the following code:

```
FUNCTION "HC::SF_GET_BEST_CUSTOMER"( )
       RETURNS best_cust nvarchar(10)
       LANGUAGE SQLSCRIPT
       SQL SECURITY INVOKER AS
BEGIN

SELECT customer_id into best_cust from "HC::SALES_DATA"
where quantity = (select max(quantity) from "HC::SALES_DATA");

END;
```

Note:
The SQLScript code can be copied from a text file in your *HC300 → Prepared Code* folder: *SQL for Scalar Function.txt*.

4. Deploy the scalar function and check the log to ensure that it was successful.

Task 2: Create a calculation view that includes an input parameter that consumes your scalar function

You will now use the function to derive the value of an input parameter and use the parameter value to filter your data.

1. Create a calculation view type CUBE in your *exercises* folder with the name and label **`CVC_SALES_IP_USING_FUNC`**.
2. Assign the table *SALES_DATA* as a data source to the aggregation node.
3. Add the columns *CUSTOMER_ID — QUANTITY — QTY_UNIT — SQL_DATE — PRODUCT_ID — AMOUNT — CURRENCY* to the output of the aggregation node.
4. Create an input parameter *IP_1* with label **`Highest Quantity Customer`** that filters the *CUSTOMER_ID* to the value returned by the scalar function.
5. Define a filter expression that uses the scalar function returned value to select only the best customer.
6. Define the sorting sequence of the column *QUANTITY* as descending so that the highest sales quantity is shown at the top of the display.
7. Deploy your calculation view.
8. Test the calculation view correctly returns the sales orders of the customer who made the biggest quantity purchase.
9. Close all the open tabs in Business Application Studio.

Unit 11
Solution 27

Derive an Input Parameter Value using a Scalar Function

Exercise Objectives

After completing this exercise, you will be able to:

- Create calculation views that include input parameters where the value is derived from a scalar function.

Business Example

You would like to learn how to use SQLScript to generate results for an input parameter.

Overview of Exercise Tasks

- Task 1: Create a new scalar function to identify the top customer based on sales quantity
- Task 2: Create a calculation view that includes an input parameter that consumes your scalar function

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a new scalar function to identify the top customer based on sales quantity

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the *EXPLORER* view, create a new scalar function **`SF_GET_BEST_CUSTOMER`** in the folder *exercises*.
   a) Navigate to *HC300_A#### → db → src → exercises*..
   b) Choose *View → Command Palette*
   c) Enter **`Artifact`** and choose the SAP HANA: Create SAP HANA Database Artifact option.
   d) Choose the *Browse* button and choose the */home/user/projects/HC300_A####/db/src/models/exercises/* folder and choose *OK*.
   e) For the artifact type, choose Function (hdbfunction).
   f) Enter **`SF_GET_BEST_CUSTOMER`** as function name and choose *Create*.
3. Create the SQLScript of your scalar function using the following code:

```
FUNCTION "HC::SF_GET_BEST_CUSTOMER"( )
       RETURNS best_cust nvarchar(10)
       LANGUAGE SQLSCRIPT
       SQL SECURITY INVOKER AS
```

```
BEGIN

SELECT customer_id into best_cust from "HC::SALES_DATA"
where quantity = (select max(quantity) from "HC::SALES_DATA");

END;
```

Note:
The SQLScript code can be copied from a text file in your *HC300 → Prepared Code* folder: *SQL for Scalar Function.txt*.

a) From Windows Explorer, in your *HC300 → Prepared Code* folder, double-click the file *SQL for Scalar Function.txt*.

b) Select all the code (**Ctrl+A**) and copy it to the clipboard (**Ctrl+C**).

c) In the *Function* source file, remove all existing content and then paste the entire code that you just copied (**Ctrl+V**).

4. Deploy the scalar function and check the log to ensure that it was successful.
   a) Choose the *Deploy* icon for the *SF_GET_BEST_CUSTOMER* function.
   b) In the *Console* pane, check that the deployment completes successfully.

Task 2: Create a calculation view that includes an input parameter that consumes your scalar function

You will now use the function to derive the value of an input parameter and use the parameter value to filter your data.

1. Create a calculation view type CUBE in your *exercises* folder with the name and label **CVC_SALES_IP_USING_FUNC**.
   a) Right-click the *exercises* folder and choose *New File*.
   b) Enter **CVC_SALES_IP_USING_FUNC.hdbcalculationview** as file name and press *Enter*.
   c) Update the label.
   d) Choose *Create*.

   Result

   The calculation view graphical editor opens on the right of your screen.
2. Assign the table *SALES_DATA* as a data source to the aggregation node.
   a) To the right of the *Aggregation* node, choose *Add Data Source*.
   b) In the *Search* field, enter **sales_data**.
   c) Select the table *SALES_DATA* and choose *Finish*.
3. Add the columns *CUSTOMER_ID — QUANTITY — QTY_UNIT — SQL_DATE — PRODUCT_ID — AMOUNT — CURRENCY* to the output of the aggregation node.
   a) Double-click on the aggregation node to open the settings area and ensure that the *Mapping* tab is selected.
   b) Drag the columns *CUSTOMER_ID — QUANTITY — QTY_UNIT — SQL_DATE — PRODUCT_ID — AMOUNT — CURRENCY* from the data source to the output columns.

4. Create an input parameter *IP_1* with label **`Highest Quantity Customer`** that filters the *CUSTOMER_ID* to the value returned by the scalar function.
   a) Select the *Parameters* tab.
   b) Choose + and select *Input parameter*.
   c) Select the new parameter *IP_1* so that the properties appear.
   d) In the *General* section, locate the *Label* field and enter **`Highest Quantity Customer`**.
   e) Select, as value for *Parameter Type*, *Derived From Procedure/Scalar Function*.
   f) Expand the *Parameter Type — Derived From Procedure/Scalar Function* section and choose *HC::SF_GET_BEST_CUSTOMER* for the *Procedure/Scalar Function*. Do not select *Input Enabled.*.
   g) Choose the *Back* button to return to the top level screen of the *Parameters* tab.
5. Define a filter expression that uses the scalar function returned value to select only the best customer.
   a) Select the *Filter Expression* tab of the *Aggregation* node.
   b) Expand *Components* → *Elements* → *Columns* and select *CUSTOMER_ID* so it is added to the expression.
   c) Add **=** after the column *CUSTOMER_ID* so the expression looks like this : **`"CUSTOMER_ID" =`**.
   d) Expand *Components* → *Elements* → *Input Parameters* and select *IP_1*.
      The complete expression now looks like this: **`"CUSTOMER_ID" = '$$IP_1$$'`**
   e) Choose *Validate Syntax* to ensure that you have no errors in the expression.
6. Define the sorting sequence of the column *QUANTITY* as descending so that the highest sales quantity is shown at the top of the display.
   a) Select the *semantics* node and then select the *Columns* tab.
   b) Select the toolbar icon*Sort Result Set* (look for the opposing arrows icon)
   c) Select the + button to add a column.
   d) From the drop-down list, select the column *QUANTITY*.
   e) Choose *Descending* for the sort direction.
   f) Press *OK*.
7. Deploy your calculation view.
   a) Choose the *Deploy* icon for the *CVC_SALES_IP_USING_FUNC* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
8. Test the calculation view correctly returns the sales orders of the customer who made the biggest quantity purchase.
   a) Right-click your calculation view and select the *Data Preview* option.
   b) Switch to the *Raw Data* tab and you should see only customer 3000 who made the highest purchase of 16 items.

9. Close all the open tabs in Business Application Studio.
   a) Right-click any open tab and choose *Close All*.

Unit 12
Exercise 28

Lesson 12.1 Implement External View for Value Help

Exercise Objectives

After completing this exercise, you will be able to:

- Implement a Calculation View for use as Value Help in another Calculation View.

Business Example

You would like to provide a list of countries for a variable used to filter customers, but need to filter the list on the EMEA region only. For this purpose you will build an external calculation view to use as value help.

Note:
The CUSTOMER table also contains a REGION column. However, this column does not contain regions such as AP, EMEA, NA, and so on, but instead the State (for US countries), or nothing (customers from Germany). So this column did not fulfill our requirement.

Overview of Exercise Tasks

- Task 1: Create a Value Help Calculation View
- Task 2: Consume the External Value Help View

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a Value Help Calculation View

First, create the value help view that will provide the allowed countries. We will use a fixed restriction.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Create a calculation view using the data in the following table:

Field	Value
Name	CVD_COUNTRIES_EMEA_VALUE_HELP
Label	Countries in EMEA
Data Category	DIMENSION

3. Assign the table *COUNTRIES* of the TRAINING schema as the data source to the default projection node.
4. In the projection node, map all columns to the output.
5. Define a filter expression to restrict countries to those in region *EMEA*.
   The resulting expression should look exactly like this : *"REGION"='EMEA'*
6. Deploy the calculation view.
7. Preview the calculation view raw data.

Task 2: Consume the External Value Help View

We will now create a calculation view that prompts the user to choose a country. However, the choices of countries will be restricted to those that are returned from the external value help view. In our case, those from EMEA region.

1. Create a calculation view using the data in the following table:

Field	Value
Name	CVD_CUSTOMERS_EMEA_VALUE_HELP
Label	Local Customers
Data Category	DIMENSION

2. Assign the table *CUSTOMER* from the TRAINING schema as the data source to the default projection node.
3. In the projection node, map all columns to the output.
4. Define a variable to allow the user to select a country, but the countries that are presented for selection must be provided from the restricted results of the external value help view *CVD_COUNTRIES_EMEA_VALUE_HELP*.

Field	Value
Name	VAR_COUNTRY
Label	Choose Country
View/Table Value Help	HC::CVD_COUNTRIES_EMEA_VAL-UE_HELP
Reference Column	HC::CVD_COUNTRIES_EMEA_VAL-UE_HELP.COUNTRY

5. Deploy the calculation view.
6. Preview the results to ensure that your value help for country only returns the countries in *EMEA*.
7. Close all the open tabs in Business Application Studio.

Unit 12
Solution 28

Lesson 12.1 Implement External View for Value Help

Exercise Objectives

After completing this exercise, you will be able to:

- Implement a Calculation View for use as Value Help in another Calculation View.

Business Example

You would like to provide a list of countries for a variable used to filter customers, but need to filter the list on the EMEA region only. For this purpose you will build an external calculation view to use as value help.

Note:
The CUSTOMER table also contains a REGION column. However, this column does not contain regions such as AP, EMEA, NA, and so on, but instead the State (for US countries), or nothing (customers from Germany). So this column did not fulfill our requirement.

Overview of Exercise Tasks

- Task 1: Create a Value Help Calculation View
- Task 2: Consume the External Value Help View

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a Value Help Calculation View

First, create the value help view that will provide the allowed countries. We will use a fixed restriction.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Create a calculation view using the data in the following table:

Field	Value
Name	CVD_COUNTRIES_EMEA_VALUE_HELP
Label	Countries in EMEA
Data Category	DIMENSION

a) In the *EXPLORER* tree, choose *HC300_A#### → db → src* and right-click the *exercises* folder. Choose *New File*.

b) Enter **`CVD_COUNTRIES_EMEA_VALUE_HELP.hdbcalculationview`** as file name and press *Enter*.

c) Enter the calculation view label and data category as specified in the table.

d) Choose *Create*.

3. Assign the table *COUNTRIES* of the TRAINING schema as the data source to the default projection node.
   a) Select the *Projection* node and choose + on the right of the node.
   b) In the *Find Data Sources* window, select the *Search* field and enter **`COUNTRIES`**.
   c) In the search result, select the table *COUNTRIES* (with synonym HC::COUNTRIES) and choose *Finish*.
4. In the projection node, map all columns to the output.
   a) Double-click the *Projection* node to open the detailed setting pane.
   b) Select the *Mapping* tab.
   c) Under the *Data Sources* pane, right-click on the data source header *HC::COUNTRIES* and choose *Add To Output*.
5. Define a filter expression to restrict countries to those in region *EMEA*.
   a) Select the *Filter Expression* tab, and under the section *Components* make sure the tab *Elements* is selected. Expand *Columns* and choose *REGION*.
   b) Enter **=**.
   c) Add a single quote (notice that a closing quote is automatically added).
   d) Between the single quotes, enter **`EMEA`**.

   The resulting expression should look exactly like this : *"REGION"='EMEA'*
6. Deploy the calculation view.
   a) Choose the *Deploy* icon for the *CVD_COUNTRIES_EMEA_VALUE_HELP* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
7. Preview the calculation view raw data.
   a) Right-click your calculation view and select the *Data Preview* option.
   b) Switch to the *Raw Data* tab and you should see only countries of the EMEA region.

Task 2: Consume the External Value Help View

We will now create a calculation view that prompts the user to choose a country. However, the choices of countries will be restricted to those that are returned from the external value help view. In our case, those from EMEA region.

1. Create a calculation view using the data in the following table:

Field	Value
Name	CVD_CUSTOMERS_EMEA_VALUE_HELP
Label	Local Customers
Data Category	DIMENSION

a) In the *Explorer* view, choose *HC300_A#### → db → src* and right-click the *exercises* folder. Choose *New File*.

b) Enter **`CVD_CUSTOMERS_EMEA_VALUE_HELP.hdbcalculationview`** as file name and press *Enter*.

c) Enter the calculation view label and data category as specified in the table.

d) Choose *Create*.

2. Assign the table *CUSTOMER* from the TRAINING schema as the data source to the default projection node.

a) Select the *Projection* node and choose + on the right of the node.

b) In the *Find Data Sources* window, select the *Search* field and enter **`CUSTOMER`**.

c) In the search result, select the table *CUSTOMER* (with synonym HC::CUSTOMER) and choose *Finish*.

3. In the projection node, map all columns to the output.

a) Double-click the *Projection* node to open the detailed setting pane.

b) Select the *Mapping* tab.

c) Under the *Data Sources* pane, right-click on the data source header *HC::CUSTOMER* and choose *Add To Output*.

4. Define a variable to allow the user to select a country, but the countries that are presented for selection must be provided from the restricted results of the external value help view *CVD_COUNTRIES_EMEA_VALUE_HELP*.

Field	Value
Name	VAR_COUNTRY
Label	Choose Country
View/Table Value Help	HC::CVD_COUNTRIES_EMEA_VAL-UE_HELP
Reference Column	HC::CVD_COUNTRIES_EMEA_VAL-UE_HELP.COUNTRY

a) Select the *Parameters* tab.

b) To view the drop-down list, choose +.

c) Select *Variable*. Notice that a placeholder variable is created, called *VAR_1*.

d) Select the variable *VAR_1* to open the detailed settings and enter the settings provided in the table.

e) Expand the section *Apply Filter*.

f) Select the *COUNTRY* column.

g) Choose *Back*.

5. Deploy the calculation view.

a) Choose the *Deploy* icon for the *CVD_CUSTOMERS_EMEA_VALUE_HELP* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

6. Preview the results to ensure that your value help for country only returns the countries in *EMEA*.

a) In the *Explorer* view, right-click the *CVD_CUSTOMERS_EMEA_VALUE_HELP* calculation view and choose *Data Preview.*.

b) The value help should only show countries of EMEA.

c) Choose a country (DE is a good choice as it has some data, whereas some countries do not).

d) Choose *Open Content*.

e) Select the *Raw Data* tab. You should see only customers in the country you have chosen.

7. Close all the open tabs in Business Application Studio.

a) Right-click any open tab and choose *Close All*.

Unit 13
Exercise 29

Access a Calculation View in Performance Analysis Mode

Exercise Objectives

After completing this exercise, you will be able to:

- Use the information provided by a Calculation View's Performance Analysis Mode.

Business Example

To ensure that your modeling decisions follow best practices, you decide to switch on the *Performance Analysis Mode* whilst building a calculation view so that you are notified of design choices that might affect performance.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the *Settings* of SAP Business Application Studio, set the *Performance Analysis Mode* threshold value for the number of table records to *10000*.

Note:
This value is voluntarily small to make sure you'll have a warning. In real life, your SAP HANA database administrator will probably want to set it at a much higher value.

3. View the definition of the calculation view *resources → CVC_SALES_PERF* in *Performance Analysis Mode*.

   Why do you now see a warning triangle inside the join node?

   _____

   _____

   _____

   How many rows does the *HC::SNWD_SO_I* data source contain?

   _____

   _____

   _____

How might you optimize this table?

4. There are two more warnings relating to potential performance issues. Locate them and describe the issues.

   Where can you find the issues ?

   What is the first warning about ?

   What is the second warning about ?

5. Correct the second issue, deploy and display again the Calculation View in Analysis Mode.

   Is there still a message about the order of the tables ?

6. View the definition of the calculation view *resources → CVC_PRICE_COMP* using the *Performance Analysis Mode*.

   There is a performance validation warning that informs you that you have not followed an SAP modeling recommendation. What is this warning?

7. In the *Settings* of SAP Business Application Studio, set the *Performance Analysis Mode* threshold value for the number of table records back to *0*.
8. Close all open tabs in the SAP Business Application Studio.

Unit 13
Solution 29

Access a Calculation View in Performance Analysis Mode

Exercise Objectives

After completing this exercise, you will be able to:

- Use the information provided by a Calculation View's Performance Analysis Mode.

Business Example

To ensure that your modeling decisions follow best practices, you decide to switch on the *Performance Analysis Mode* whilst building a calculation view so that you are notified of design choices that might affect performance.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In the *Settings* of SAP Business Application Studio, set the *Performance Analysis Mode* threshold value for the number of table records to *10000*.

Note:
This value is voluntarily small to make sure you'll have a warning. In real life, your SAP HANA database administrator will probably want to set it at a much higher value.

   a) Choose the cogwheel icon on the bottom left of SAP Business Application Studio, then choose *Settings*.

   b) Enter **`Performance`** in the *Search* field.

   c) Find the *Watt Feature > Calculation view Editor: Performance Analysis Threshold Value* parameter.

   d) In the threshold value field, enter **`10000`**.

   e) Close the *Settings* tab.

3. View the definition of the calculation view *resources → CVC_SALES_PERF* in *Performance Analysis Mode*.
   a) Double-click the calculation view *resources → CVC_SALES_PERF* and then select the *Performance Analysis Mode* by choosing the button above the node palette that looks like a speedometer. You should then see a message appear at the bottom of the screen *'The editor is in performance mode'*..

Why do you now see a warning triangle inside the join node?

This is a warning to let you know that the table *HC::SNWD_SO_I* exceeds the number of records defined in the modeler preferences.

How many rows does the *HC::SNWD_SO_I* data source contain?

Double-click the *Join_1* node and in the *Data Source Details* pane notice the number of rows for the *HC::SNWD_SO_I* table is 11760.

How might you optimize this table?

Consider partitioning the table according to the data selections that are made by users, for example, by client / currency code.

4. There are two more warnings relating to potential performance issues. Locate them and describe the issues.
   a) At the bottom of the editor, notice a warning triangle. When you hover over it, you see a pop-up box that reminds you that you have defined a filter expression on an aggregated column and that the right table contains more rows than the left one.

Where can you find the issues ?

After the message at the bottom of the screen.

What is the first warning about ?

We have defined a filter expression on an aggregated column.

What is the second warning about ?

The right table contains more records than the left one.

5. Correct the second issue, deploy and display again the Calculation View in Analysis Mode.
   a) Select the *Performance Analysis Mode* icon to get back to design mode.
   b) Double_click the *Join_1* node and open the *Join Definition*.
   c) In the *PROPERTIES* pane, choose the *Swap Table* icon.
   d) Choose the *Deploy* icon for the *CVC_SALES_PERF* calculation view.
   e) In the *Console* pane, check that the deployment completes successfully.
   f) Select the *Performance Analysis Mode* icon again to get into the Performance Analysis Mode and look for the warning messages.

Is there still a message about the order of the tables ?

No. Only the message about the filter on aggregated column.

6. View the definition of the calculation view *resources → CVC_PRICE_COMP* using the *Performance Analysis Mode*.
   a) Double-click the calculation view *resources → CVC_PRICE_COMP* and select the *Performance Analysis Mode* by choosing the button above the node palette that looks like a speedometer. You should then see a message appear at the bottom of the screen *'The editor is in performance mode'*..

   There is a performance validation warning that informs you that you have not followed an SAP modeling recommendation. What is this warning?

   Hover over the triangle at the bottom of the screen to read the text that informs you that you have joined data sources on a calculated column. This is not recommended if you want to obtain the very best performance.

7. In the *Settings* of SAP Business Application Studio, set the *Performance Analysis Mode* threshold value for the number of table records back to *0*.
   a) Choose the cogwheel icon on the bottom left of SAP Business Application Studio, then choose *Settings*.
   b) Enter **`Performance`** in the *Search* field.
   c) Find the *Watt Feature > Calculation view Editor: Performance Analysis Threshold Value* parameter.
   d) In the threshold value field, enter `0`.
   e) Close the *Settings* tab.
8. Close all open tabs in the SAP Business Application Studio.
   a) Right-click any open tab and choose *Close All*.

Unit 13
Exercise 30

Use Debug Query Mode

Exercise Objectives

After completing this exercise, you will be able to:

- Debug a Calculation View with the Debug Query Mode.

Business Example

You want to see how your calculation view behaves when a query calls it, so you view the calculation view in Debug Query mode to test it.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. From the folder *resources*, open the calculation view *src* → *models* → *resources* → *CVCS_SO* in Debug Query mode.
3. Launch the default debug query to generate SQL at each node.
4. Switch to the *Join_1* node and execute the generated SQL to view the intermediate results.
5. There is a specific measure value (5.36) that is causing problems and you want to isolate this amount at the join node. Modify and execute the SQL that was generated at the join node so that you display only records where *GROSS_AMOUNT* is equal to 5.36.
6. Close the Calculation View *CVCS_SO* before the next step (you will reopen it and use a different query).
7. Once more, launch the calculation view *CVCS_SO* but this time modify the default debug query to select only the columns *GROSS_AMOUNT* and *COUNTRY*, apply a filter to select only country FR, and then execute the modified query.
8. Review the columns that are grayed out due to pruning at the Semantics node and also at the Star Join node.
9. Close all open tabs in Business Application Studio.

Unit 13
Solution 30

Use Debug Query Mode

Exercise Objectives

After completing this exercise, you will be able to:

- Debug a Calculation View with the Debug Query Mode.

Business Example

You want to see how your calculation view behaves when a query calls it, so you view the calculation view in Debug Query mode to test it.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. From the folder *resources*, open the calculation view *src* → *models* → *resources* → *CVCS_SO* in Debug Query mode.
   a) In the *Explorer* view, expand the folder *resources*.
   b) Select the calculation view *CVCS_SO* to open it in the graphical editor.
   c) Choose *Debug Mode* (the last button on the right of the toolbar).
3. Launch the default debug query to generate SQL at each node.
   a) Double-click the node *Semantics*.
   b) Just above the debug query SQL code and to the right, choose *Execute*.
      You won't see any data yet as this step simply generates the SQL at each node of the calculation view.
4. Switch to the *Join_1* node and execute the generated SQL to view the intermediate results.
   a) Highlight the *Join_1* node.
   b) Select the *Debug Query* tab. You might have to scroll to find the tab which is the very last one.
   c) Just above the SQL code and to the right, choose *Execute*.
   d) View the results below the SQL code.
5. There is a specific measure value (5.36) that is causing problems and you want to isolate this amount at the join node. Modify and execute the SQL that was generated at the join node so that you display only records where *GROSS_AMOUNT* is equal to 5.36.

a) At the end of the SQL code add the following code :

```
where GROSS_AMOUNT = 5.36
```

This is an example of how you can customize the SQL code at any node to test different conditions.

b) Just above the SQL code and to the right, choose *Execute*.

c) View the new results below the SQL code.

6. Close the Calculation View *CVCS_SO* before the next step (you will reopen it and use a different query).

a) Click the X *Close* button on the corresponding tab.

7. Once more, launch the calculation view *CVCS_SO* but this time modify the default debug query to select only the columns *GROSS_AMOUNT* and *COUNTRY*, apply a filter to select only country FR, and then execute the modified query.

a) Select the calculation view *CVCS_SO* to open it.

b) Double-click the *Semantics* node.

c) To enable the debug mode, choose *Debug Mode*.

d) Carefully remove all columns except *GROSS_AMOUNT* and *COUNTRY* from the SQL statement (including the `GROUP BY` clause), and add the *WHERE* clause, so that the final code looks like this:

```
SELECT TOP 1000  SUM("GROSS_AMOUNT") AS "GROSS_AMOUNT" , "COUNTRY"
FROM  "HC300_A####_HDI_DB_1"."HC::CVCS_SO" WHERE "COUNTRY" = 'FR'
GROUP BY "COUNTRY"
```

e) Just above the SQL code and to the right, choose *Execute* to generate the SQL for each node in the calculation view

8. Review the columns that are grayed out due to pruning at the Semantics node and also at the Star Join node.

a) Select the *Semantics* node and then the *Columns* tab, and review the columns under each of the sub tabs *Private* and *Shared*. Notice how the unwanted columns are pruned (grayed out).

b) Select the *Star Join* node and then the *Columns* tab. Notice how the unwanted columns are pruned (grayed out).

> Note:
> Actually, the used (not pruned) columns are also slightly grayed out (not black, but a different/darker shade of gray). These used columns should normally show in black.

c) Back to the *Debug Query* tab, just above the SQL code and to the right, choose *Execute* to run the SQL.

d) View the results below the SQL code. You should see only the columns *COUNTRY* and *GROSS_AMOUNT*.

9. Close all open tabs in Business Application Studio.
   a) Right-click any open tab and choose *Close All*.

Unit 13
Exercise 31

Use the SAP HANA SQL Analyzer

Exercise Objectives

After completing this exercise, you will be able to:

- Check that SQL Analyzer is ready to work with
- Analyze a Calculation View from within SAP Business Application Studio

Business Example

In order to check the execution of calculation views, and that optimizations you made to their design is actually applied, you want to get further details on how queries executed against these views are executed.

Overview of Exercise Tasks

- Task 1: Check that the Application User has the necessary authorizations to use SQL Analyzer
- Task 2: Analyze the execution of a calculation view

Prerequisites

Your SAP Business Application Studio development environment should have been set up with the optional component *SAP HANA Performance Tools*, and the HC300 project should have been imported and deployed.

Task 1: Check that the Application User has the necessary authorizations to use SQL Analyzer

Most of the actions you trigger inside an HDI Container are executed by a technical user which is specific to this container. For example executing a query on top of a calculation view. In particular, when you want to analyze query plans with the SAP HANA SQL Analyzer, this technical user needs a couple of privileges. If not, you would get an error when invoking the *Generate Plan* command.

Checking these privileges can be done via an SQL script that you find in the file `SQL to Check SQL Analyzer Configuration.txt`.

1. In SAP Business Application Studio, open an SQL Console connected to your *HC300_A####_HDI_CONTAINER*.
2. In the file explorer of your computer, open the script from your course files and copy its content to the opened SQL console.

   The script is `HC300\SQL to Check SQL Analyzer Configuration.txt`.
3. Execute the first statement in section 1.3.a and copy its results

```
SELECT CURRENT_USER FROM DUMMY;
```

4. In the SQL console, replace *XXXX* with the actual technical user name.

5. Execute the second statement in section 1.5.a

```
SELECT GRANTEE, GRANTOR, OBJECT_TYPE, PRIVILEGE
FROM EFFECTIVE_PRIVILEGES
WHERE USER_NAME = 'XXXX'
AND GRANTEE = 'XXXX'
AND PRIVILEGE IN ('TRACE ADMIN','INIFILE ADMIN');
```

6. Check that the query result shows that the two roles *TRACE ADMIN* and *INIFILE ADMIN* are granted to your technical user *HC300_A####_HDI_DB_....._RT*.

   Result

   You are now ready to use SQL Analyzer from within your HDI Container.

Task 2: Analyze the Execution of a Calculation View

You want to generate and analyze the query plan of a calculation view deployed into your HDI Container. You will perform all the steps in SAP Business Application Studio, which is possible since SAP HANA Cloud QRC 2/2023.

1. in the *SAP HANA Database Explorer* view, generate the default query for your calculation view *CVCS_SO*.

Hint:
You can also preview the data of the calculation view from the *Explorer* view, and from the *Raw Data* tab, choose *Edit SQL Query in SQL Console*. The path to the design-time file of the calculation view is *src* → *models* → *resources* → *CVCS_SO*. However, from this entry point, the SQL Analyzer plan file needs to be downloaded, and then uploaded to the SAP HANA SQL Analyzer view.

2. Generate the SQL Query Plan for the default query. Give the plan file the prefix **`CVCS_SO_Default`**.

Note:
In other contexts, especially when your generate a lot of plans, you can also use your initials or any other string that helps you identify each plan file.

3. Observe the plan overview and answer the following questions.

   How many tables were accessed?

   _____

   _____

   _____

   What is the dominant operator of this plan?

   _____

   _____

   _____

4. Generate the plan file of a different query on top of the same calculation view, excluding all columns related to the business partners. Use the prefix **`CVCS_SO_NoBPColumns`**.

   The new query should look as follows:

```
SELECT TOP 1000
"CLIENT_1",
"SO_ID",
"CURRENCY_CODE",
"CATEGORY",
"PRODUCT_ID",
SUM("GROSS_AMOUNT") AS "GROSS_AMOUNT"
FROM "HC300_A####_HDI_DB_1"."HC::CVCS_SO"
GROUP BY "CLIENT_1", "SO_ID", "CURRENCY_CODE","CATEGORY", "PRODUCT_ID"
```

   The columns to exclude are:

   - BP_COMPANY_NAME
   - COUNTRY
   - COUNTRY_NAME
   - REGION

5. Observe the new plan overview and answer the following questions.

   How many tables were accessed?

   _____
   _____
   _____

   How can you explain the change compared with the previous query?

   _____
   _____
   _____

6. In the *CVCS_SO* calculation view, confirm the answer to the previous question by checking the type of join set in the *Star Join* node for the *CVD_BP2* dimension calculation view.

   What are the join type and cardinality?

   _____
   _____
   _____

   How does it impact query execution when no column from *HC::CVD_BP2* is requested?

   _____
   _____
   _____

7. Close all the open tabs in SAP Business Application Studio and go back to the *Explorer* view.

Unit 13
Solution 31

Use the SAP HANA SQL Analyzer

Exercise Objectives

After completing this exercise, you will be able to:

- Check that SQL Analyzer is ready to work with
- Analyze a Calculation View from within SAP Business Application Studio

Business Example

In order to check the execution of calculation views, and that optimizations you made to their design is actually applied, you want to get further details on how queries executed against these views are executed.

Overview of Exercise Tasks

- Task 1: Check that the Application User has the necessary authorizations to use SQL Analyzer
- Task 2: Analyze the execution of a calculation view

Prerequisites

Your SAP Business Application Studio development environment should have been set up with the optional component *SAP HANA Performance Tools*, and the HC300 project should have been imported and deployed.

Task 1: Check that the Application User has the necessary authorizations to use SQL Analyzer

Most of the actions you trigger inside an HDI Container are executed by a technical user which is specific to this container. For example executing a query on top of a calculation view. In particular, when you want to analyze query plans with the SAP HANA SQL Analyzer, this technical user needs a couple of privileges. If not, you would get an error when invoking the *Generate Plan* command.

Checking these privileges can be done via an SQL script that you find in the file `SQL to Check SQL Analyzer Configuration.txt`.

1. In SAP Business Application Studio, open an SQL Console connected to your *HC300_A####_HDI_CONTAINER*.
   a) If needed, launch SAP Business Application Studio
   b) In the *Activity* bar, choose *SAP HANA Database Explorer*.
   c) In the *Database List*, locate and hover your mouse over the *HC300_A####_HDI_CONTAINER* connection.
   d) Choose the icon *Open SAP HANA SQL Console* icon.
2. In the file explorer of your computer, open the script from your course files and copy its content to the opened SQL console.

The script is `HC300\SQL to Check SQL Analyzer Configuration.txt`.

a) In your course files, open the file `HC300\SQL to Check SQL Analyzer Configuration.txt`.

b) Select the entire content of the file (press **Ctrl+A**) and to copy the script, press **Ctrl +C**.

c) In the SQL Console, to paste the script, choose **Ctrl+V**.

3. Execute the first statement in section 1.3.a and copy its results

```
SELECT CURRENT_USER FROM DUMMY;
```

a) Select the first statement in section 1.3.a and press **F8**.

Alternatively, you can put your cursor anywhere in the first statement and press **F9** *(Run Statement)*.

Result

The identifier of the technical user is displayed in the *Result 1* tab. It should look like *HC300_A####_HDI_DB_1_..._RT*.

b) In the *Result 1* tab, double-click the technical user name.

c) Choose *Copy*.

d) Choose *Close*.

4. In the SQL console, replace *XXXX* with the actual technical user name.

a) Use the search and replace feature (**Ctrl + H**), pasting the content of your clipboard in the *Replace with* field.

b) Alternatively, you can just select the *XXXX* strings and paste the content of your clipboard. There are 2 replacements to make.

5. Execute the second statement in section 1.5.a

```
SELECT GRANTEE, GRANTOR, OBJECT_TYPE, PRIVILEGE
FROM EFFECTIVE_PRIVILEGES
WHERE USER_NAME = 'XXXX'
AND GRANTEE = 'XXXX'
AND PRIVILEGE IN ('TRACE ADMIN','INIFILE ADMIN');
```

a) Select the last statement in section 1.5.a and press **F9**.

b) Check that the SQL statement was executed successfully.

6. Check that the query result shows that the two roles *TRACE ADMIN* and *INIFILE ADMIN* are granted to your technical user *HC300_A####_HDI_DB_....._RT*.

Result

You are now ready to use SQL Analyzer from within your HDI Container.

Task 2: Analyze the Execution of a Calculation View

You want to generate and analyze the query plan of a calculation view deployed into your HDI Container. You will perform all the steps in SAP Business Application Studio, which is possible since SAP HANA Cloud QRC 2/2023.

1. in the *SAP HANA Database Explorer* view, generate the default query for your calculation view *CVCS_SO*.

> Hint:
> You can also preview the data of the calculation view from the *Explorer* view, and from the *Raw Data* tab, choose *Edit SQL Query in SQL Console*. The path to the design-time file of the calculation view is *src* → *models* → *resources* → *CVCS_SO*. However, from this entry point, the SQL Analyzer plan file needs to be downloaded, and then uploaded to the SAP HANA SQL Analyzer view.

a) Open the *SAP HANA Database Explorer* view.

b) In the *Database List* sub-pane, expand your container *HC300_A####_HDI_CONTAINER* and select *Column Views*.

c) In the *Catalog Browser* sub-pane, choose the *Apply Filter* icon (a funnel) and enter **`CVCS`**.

d) In the filtered list, right-click the *HC::CVCS_SO* calculation view and choose *Generate SELECT Statement*.

Result
The default SELECT statement shows up in a new SQL Console tab.

2. Generate the SQL Query Plan for the default query. Give the plan file the prefix **`CVCS_SO_Default`**.

> Note:
> In other contexts, especially when your generate a lot of plans, you can also use your initials or any other string that helps you identify each plan file.

a) In the SQL Console, choose  *(more)* → *[Analyze] Executed Plan*.

b) In the *Save Plan* dialog box, enter the *File Name Prefix* **`CVCS_SO_Default`**.

c) Choose *Save*.

Result
After a while, the calculation view results are displayed at the bottom of the SQL Console, and the *SAP HANA SQL Analyzer* view opens, showing the graphical and other tools to analyze the generated plan file.

3. Observe the plan overview and answer the following questions.

How many tables were accessed?

7

What is the dominant operator of this plan?

AGGREGATION

4. Generate the plan file of a different query on top of the same calculation view, excluding all columns related to the business partners. Use the prefix **CVCS_SO_NoBPColumns**.

   The new query should look as follows:

```
SELECT TOP 1000
"CLIENT_1",
"SO_ID",
"CURRENCY_CODE",
"CATEGORY",
"PRODUCT_ID",
SUM("GROSS_AMOUNT") AS "GROSS_AMOUNT"
FROM "HC300_A####_HDI_DB_1"."HC::CVCS_SO"
GROUP BY "CLIENT_1", "SO_ID", "CURRENCY_CODE","CATEGORY", "PRODUCT_ID"
```

   The columns to exclude are:

   - BP_COMPANY_NAME
   - COUNTRY
   - COUNTRY_NAME
   - REGION

   a) Go back to the tab where the default query on Calculation View *CVCS_SO* was analyzed.

   b) In the *SELECT* and *GROUP BY* sections of the SQL statement, remove the 4 columns mentioned above.

   c) Choose  *(more)* → *[Analyze] Executed Plan*.

   d) In the *Save Plan* dialog box, enter the *File Name Prefix* **CVCS_SO_NoBPColumns**.

   e) Choose *Save*.

5. Observe the new plan overview and answer the following questions.

   How many tables were accessed?

   3

   How can you explain the change compared with the previous query?

   An optimization such as join pruning might have been triggered when creating the query plan.

6. In the *CVCS_SO* calculation view, confirm the answer to the previous question by checking the type of join set in the *Star Join* node for the *CVD_BP2* dimension calculation view.

   a) If needed, to open the calculation view, from the *Explorer* view, double-click the calculation view *db* → *src* → *models* → *resources* → *CVCS_SO.hdbcalculationview*.

   b) Double-click the *Star Join* node and display the *Join Definition* tab.

c) Select any join arrow between the *Join_1* and *HC::CVD_BP2* sources.

d) In the *Properties* sub-pane, check the *Join Type* and the *Cardinality*.

What are the join type and cardinality?

The join type is *Referential* and the cardinality *n..1* (1 on the *CVD_BP2* side).

How does it impact query execution when no column from *HC::CVD_BP2* is requested?

Upon query execution, the *HC::CVD_BP2* source of the join is ignored, because the referential join property means/tells the optimizer that any record from *Join_1* will always have 1 (and only 1) matching record in *HC::CVD_BP2*.

7. Close all the open tabs in SAP Business Application Studio and go back to the *Explorer* view.

a) Right-click any open tab and choose *Close All*. Don't save anything.

b) In the *Activity Bar*, select the *Explorer* view.

Unit 14
Exercise 32

Control Parallelization in a Data Flow

Exercise Objectives

After completing this exercise, you will be able to:

- Control parallelization in the data flow of a calculation view.
- Check parallelization in the SAP HANA SQL Analyzer.

Business Example

Your calculation view processes large volumes of data and there appears to be a bottleneck caused by the generation of a complex calculated column within the data flow. You would like to force the data to be processed in parallel where the bottleneck occurs in the data flow to try and improve the performance of the calculation view.

Overview of Exercise Tasks

- Task 1: Create a Calculation View and implement parallelization
- Task 2: Analyze the execution to check parallelization

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a Calculation View and Implement Parallelization

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVD_CONTROL_PARALLELIZATION`**
Label	**`CVD_CONTROL_PARALLELIZATION`**
Data Category	*DIMENSION*

3. Add a new *Projection* node and rename the node label as **`Start_Parallel`**.
4. In the new *Projection* node, add the following data source:
   Table *HC::Empl_Salary_Current*
5. Expand the *Details* panel.
6. On the *Mapping* tab of the new *Start_Parallel* node, add all columns to the output.
7. Enable the Setting *Partition Local Execution* to start the parallelization and choose *Year* for the partition column.

8. Add a new *Projection* node above the one created earlier and rename the node label as **`Processing_In_Parallel`**.
9. Connect the projection node *Start_Parallel* to the projection node *Processing_In_Parallel* and map all columns to the output.
10. In the projection node *Processing_In_Parallel*, create a new calculated column using the information in the table below:

Setting	Value
Name	MonthlySalary
Data Type	DECIMAL
Length	10
Scale	2
Expression	"TotalSalary" / 12

Hint:
Make sure to use double quotes in the SQL expression so that the case of the *TotalSalary* column name is passed properly to the expression as *TotalSalary*, not *TOTALSALARY*.

11. Add a new *Union* node above the projection node *Processing_In_Parallel* and rename the union node label as **`Stop_Parallel`**.
12. Connect the projection node *Processing_In_Parallel* to the union node *Stop_Parallel* and map all columns to the output.
13. Terminate the parallel processing in the union node.
14. Connect the union node *Stop_Parallel* to the default projection node and map all columns to the output.
15. Use the auto layout feature to properly display the calculation view scenario.
16. Deploy the *CVD_CONTROL_PARALLELIZATION* calculation view.

Task 2: Analyze the Execution to Check Parallelization

1. Open *SAP HANA Database Explorer* tool and generate the SELECT statement.
2. Generate the SQL Analyzer Plan file. Use **`SQLA`** as prefix.

   Result
   The *HANA SQL Analyzer* view opens, showing the query execution plan.
3. Check the parallel execution of the query.

   Is there any parallel execution ?

4. Close all the open tabs in Business Application Studio and go back to the *Explorer* view.

Unit 14
Solution 32

Control Parallelization in a Data Flow

Exercise Objectives

After completing this exercise, you will be able to:

- Control parallelization in the data flow of a calculation view.
- Check parallelization in the SAP HANA SQL Analyzer.

Business Example

Your calculation view processes large volumes of data and there appears to be a bottleneck caused by the generation of a complex calculated column within the data flow. You would like to force the data to be processed in parallel where the bottleneck occurs in the data flow to try and improve the performance of the calculation view.

Overview of Exercise Tasks

- Task 1: Create a Calculation View and implement parallelization
- Task 2: Analyze the execution to check parallelization

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a Calculation View and Implement Parallelization

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVD_CONTROL_PARALLELIZATION`**
Label	**`CVD_CONTROL_PARALLELIZATION`**
Data Category	*DIMENSION*

   a) In the *Explorer* view, right-click the folder *HC300_A####* → *db* → *src* → *exercises* folder and choose *New File*.

   b) Enter **`CVD_CONTROL_PARALLELIZATION.hdbcalculationview`** as file name and press *Enter*.

   c) Enter the calculation view label and Data Category as specified in the table.

   d) Choose *Create*.

3. Add a new *Projection* node and rename the node label as **`Start_Parallel`**.

a) Select and put a *Projection* node from the palette into the area at the very bottom of the data flow canvas.

b) Right-click the node and choose *Rename* and enter the name **`Start_Parallel`** and press Enter.

4. In the new *Projection* node, add the following data source:

   Table *HC::Empl_Salary_Current*

   a) Select the *Start_Parallel* node that you just added.

   b) Choose **+** *Add Data Source*.

   c) In the *Find Data Sources* window, click the search field and enter **`empl`**.

   d) In the search results, select the table *HC::Empl_Salary_Current*.

   e) Choose *Finish*.

5. Expand the *Details* panel.

   a) Choose the  *Expand Details Panel* icon in the top right corner of the screen.

6. On the *Mapping* tab of the new *Start_Parallel* node, add all columns to the output.

   a) Select the new *Start_Parallel* node and on the *Mapping* tab, right-click the data source *HC::Empl_Salary_Current* and choose *Add to Output* so that all three columns are added to the output pane.

7. Enable the Setting *Partition Local Execution* to start the parallelization and choose *Year* for the partition column.

   a) Expand *PROPERTIES* section of the mapping tab (the lower part of the mapping tab)

   b) If you don't see the setting *Partition Local Execution*, re-select the data source *HC::Empl_Salary_Current* so that the setting *Partition Local Execution* appears under *PROPERTIES*.

   c) Select the setting *Partition Local Execution* so a tick appears in the box.

   d) Using the *Partition Column* drop-down selector, choose the column *Year*.

8. Add a new *Projection* node above the one created earlier and rename the node label as **`Processing_In_Parallel`**.

   a) Select and put a *Projection* node from the palette into the canvas area between the two existing projection nodes.

   b) Right-click the new node and choose *Rename* and enter the name **`Processing_In_Parallel`** so it has a clear meaning.

9. Connect the projection node *Start_Parallel* to the projection node *Processing_In_Parallel* and map all columns to the output.

   a) Drag a line from *Start_Parallel* projection node to the *Processing_In_Parallel* projection node to connect them.

   b) On the *Mapping* tab of the *Processing_In_Parallel* projection node right-click the data source *Start_Parallel* and choose *Add to Output* so that all three columns are added to the output pane.

10. In the projection node *Processing_In_Parallel*, create a new calculated column using the information in the table below:

Setting	Value
Name	MonthlySalary
Data Type	DECIMAL
Length	10
Scale	2
Expression	"TotalSalary" / 12

Hint:
Make sure to use double quotes in the SQL expression so that the case of the *TotalSalary* column name is passed properly to the expression as *TotalSalary*, not *TOTALSALARY*.

a) Select the projection node *Processing_In_Parallel* then select the tab *Calculated Columns* and choose the *Add* button.

b) Select the template calculated column *CC_1* and enter the details from the table.

11. Add a new *Union* node above the projection node *Processing_In_Parallel* and rename the union node label as **`Stop_Parallel`**.

    a) Select and drop a *Union* node from the palette to the area directly above the *Processing_In_Parallel* projection node.

    b) Right-click the union node and choose *Rename* and enter the name **`Stop_Parallel`** so it has a clear meaning.

12. Connect the projection node *Processing_In_Parallel* to the union node *Stop_Parallel* and map all columns to the output.

    a) Drag a line from *Processing_In_Parallel* projection node to the *Stop_Parallel* union node to connect them.

    b) On the *Mapping* tab of the *Stop_Parallel* union node right-click the data source *Processing_In_Parallel* and choose *Add to Output* so that all four columns are added to the output pane.

13. Terminate the parallel processing in the union node.

    a) In the union node, re-select the data source *Processing_In_Parallel* so that the setting *Partition Local Execution* appears under *PROPERTIES* (clicking the header of the data source acts like a toggle between settings under properties).

    b) Select the setting *Partition Local Execution* so a tick appears in the box.

14. Connect the union node *Stop_Parallel* to the default projection node and map all columns to the output.

    a) Drag a line from *Stop_Parallel* union node to the default projection node to connect them.

    b) Select the default projection node and on the *Mapping* tab right-click the data source *Stop_Parallel* and choose *Add to Output* so that all columns are added to the output pane.

15. Use the auto layout feature to properly display the calculation view scenario.
    a) Choose the *Auto Layout* icon to rearrange the scenario symbols.
16. Deploy the *CVD_CONTROL_PARALLELIZATION* calculation view.
    a) Choose the *Deploy* icon for your *CVD_CONTROL_PARALLELIZATION* calculation view..
    b) In the *Console* pane, check that the deployment completes successfully.

Task 2: Analyze the Execution to Check Parallelization

1. Open *SAP HANA Database Explorer* tool and generate the SELECT statement.
    a) Open the *SAP HANA Database Explorer* view.
    b) Select your HDI container, expand its content and select *Columns Views*.
    c) In the *Catalog Browser* sub-pane, set the filter to **`control`**.
    d) Right-click on the HC::CVD_CONTROL_PARALLELIZATION view and choose *Generate SELECT statement*.
2. Generate the SQL Analyzer Plan file. Use **`SQLA`** as prefix.
    a) Top-right of the SQL console tab, choose ... → *Executed Plan*.
    b) Enter **`SQLA`** as *File Name Prefix*.
    c) Choose *Save*.

    Result
    The *HANA SQL Analyzer* view opens, showing the query execution plan.
3. Check the parallel execution of the query.
    a) Select the *PLAN GRAPH* tab.
    b) Scroll down the execution plan to find the first operations.
    c) You should see two parallel table scans. One with a filter on year 2022 and another one on the year 2023.

    Is there any parallel execution ?

    Yes. The first two operations are table scans with filters on year 2022 or 2023 and are executed in parallel.
4. Close all the open tabs in Business Application Studio and go back to the *Explorer* view.
    a) Right-click any open tab and choose *Close All*. Don't save anything.
    b) Select the *Explorer* view.

Unit 14
Exercise 33

Implement Union Pruning in a Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Create a table pruning configuration table and implement pruning rules
- Observe the impact of pruning configuration on the calculation view execution

Business Example

You are working as a modeler on an HR Data Analysis project. The salary data is distributed on two different systems, so that only the most recent data is stored in the in-memory database (“hot” tier), and the historical data is stored in a “cold” tier that consumes less resources, as it is normally accessed less often. A single calculation view, with a union, has been created to combine all salary data in order to enable access to the entire data set with the same queries.

You have been asked to review this calculation view and to check whether it can be optimized.

Note:
In the context of this exercise, the multiple tier configuration is simulated by two different tables, defined in your HDB module and stored in your container schema. In a production system, the table containing the historical data might be accessed remotely using SDA. However, the design of the calculation view would remain the same as we have here.

Overview of Exercise Tasks

- Task 1: Analyze the Existing Calculation View.
- Task 2: Execute a Query on Top of the Calculation View.
- Task 3: Implement a Union Pruning to Optimize your Calculation View.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Analyze the Existing Calculation View

First, you want to see how the *Employee Salary* calculation view is designed and identify its data sources.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Open the Calculation View *db* → *src* → *resources* → *CVC_EMPL_SALARY*.
3. Observe the design of the *Union_1* node.

What are the data sources of the *Union_1* node?

4. Preview the data of both data sources.

> Hint:
> You can preview the data sources directly from the calculation view design, in the *Scenario* pane or in the *Mapping* tab.

To which years does the data from each data source correspond?

Task 2: Execute a Query on Top of the Calculation View

You want to query the total wages paid in 2023 to each employee.

1. Execute the default generated SELECT statement.
2. Adapt the default generated SELECT statement to return only the data for 2023, and execute it again.

> Hint:
> Because the SQL session is connected to your container schema, the schema name is not mandatory in the SQL statement and can be removed. You can also simplify the query by removing the *TOP 1000* clause because the source tables only contain a few records.

After your modifications, the code should look like the following example:

```
SELECT "Year", "EmployeeID", SUM("TotalSalary")
FROM "HC::CVC_EMPL_SALARY"
WHERE "Year" = 2023
GROUP BY "Year", "EmployeeID";
```

3. Analyze the SQL query. Enter **SQLA** as prefix.

How many tables have been used during the execution of the query?

What does it mean?

Task 3: Implement a Union Pruning to Optimize your Calculation View

You want to optimize this calculation view so that, whenever possible, the non-relevant data sources of the union node are pruned. For example, when querying data for 2022 or 2023, the historical data source should not be scanned at all because by design, it does not contain any valid data for these years.

> Note:
> To be able to compare the original view and the optimized one if needed, you will define the pruning in a **copy** of the original calculation view.

1. To make your screen cleaner, close all open tabs in Business Application Studio.
2. Copy the original Calculation View *CVC_EMPL_SALARY* from the *resources* folder to your *exercises* folder. Rename the copy **`CVC_EMPL_SALARY_PRUNING`**.

   > Hint:
   > Check that the runtime name of the calculation view is also updated by opening the *Semantics* → *View Properties* → *General* window and checking the *Name* property.

3. In the new (copied) calculation view, add a pruning configuration table with the name *Empl_Salary_Pruning*.

   > Caution:
   > Do NOT define the namespace (the namespace, *HC*, will be added automatically to your table identifier upon deployment).

   Define the pruning conditions as follows:

   Table 6: Pruning Configuration Entries

Input	Column	Operators	Low Value
HC::Empl_Salary_Current	Year	>=	2022
HC::Empl_Salary_Historical	Year	<	2022

   > Caution:
   > The pruning configuration entries define which value (or range of values) of a specified attribute CAN be found in each of the data sources.

4. Deploy the Calculation View.

   Check the build status.

5. From the default SELECT statement generated for the new calculation view, adapt the SQL query to return only the data for 2023, as you did in Task 2, and analyze the query.

After your modifications, the code should look as follows (note that it now refers to the new calculation view):

```
SELECT "Year", "EmployeeID", SUM("TotalSalary")
FROM "HC::CVC_EMPL_SALARY_PRUNING"
WHERE "Year" = 2023
GROUP BY "Year", "EmployeeID";
```

How many tables have been used during the execution of the query?

6. Close all the open tabs in Business Application Studio and go back to the *Explorer* view.

Unit 14
Solution 33

Implement Union Pruning in a Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Create a table pruning configuration table and implement pruning rules
- Observe the impact of pruning configuration on the calculation view execution

Business Example

You are working as a modeler on an HR Data Analysis project. The salary data is distributed on two different systems, so that only the most recent data is stored in the in-memory database (“hot” tier), and the historical data is stored in a “cold” tier that consumes less resources, as it is normally accessed less often. A single calculation view, with a union, has been created to combine all salary data in order to enable access to the entire data set with the same queries.

You have been asked to review this calculation view and to check whether it can be optimized.

Note:
In the context of this exercise, the multiple tier configuration is simulated by two different tables, defined in your HDB module and stored in your container schema. In a production system, the table containing the historical data might be accessed remotely using SDA. However, the design of the calculation view would remain the same as we have here.

Overview of Exercise Tasks

- Task 1: Analyze the Existing Calculation View.
- Task 2: Execute a Query on Top of the Calculation View.
- Task 3: Implement a Union Pruning to Optimize your Calculation View.

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Analyze the Existing Calculation View

First, you want to see how the *Employee Salary* calculation view is designed and identify its data sources.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Open the Calculation View *db* → *src* → *resources* → *CVC_EMPL_SALARY*.
   a) In the *Explorer* view, navigate to the folder *db* → *src* → *resources*.

b) Select the calculation view *CVC_EMPL_SALARY*.

c) Choose *Expand Details Panel*.

3. Observe the design of the *Union_1* node.
   a) In the *Scenario* pane, select the *Union_1* node.
   b) Display the *Mapping* tab.

   What are the data sources of the *Union_1* node?

   The *Union_1* node has two data sources: *HC::Empl_Salary_Current* and *HC::Empl_Salary_Historical*.

4. Preview the data of both data sources.

> Hint:
> You can preview the data sources directly from the calculation view design, in the *Scenario* pane or in the *Mapping* tab.

   a) In the *Mapping* tab of the *Union_1* node, select the data source *HC::Empl_Salary_Current* then right-click and choose *Data Preview*.
   b) Repeat the previous step for the other data source, *HC::Empl_Salary_Historical*.

> Note:
> Make sure to first select the other data source, and then right-click. If you do not do that, you might query the same data source again.

   To which years does the data from each data source correspond?

   Current data corresponds to the years 2022 and 2023, while historical data corresponds to the years 2020 and 2021.

Task 2: Execute a Query on Top of the Calculation View

You want to query the total wages paid in 2023 to each employee.

1. Execute the default generated SELECT statement.
   a) In the *SAP HANA Database Explorer* view, expand the database list.
   b) Expand your HDI container HC300_A####_HDI_CONTAINER and select *Columns Views*.
   c) In the *Catalog Browser* sub-pane, add a filter on **`salary`**.
   d) , right-click on the *HC::CVC_EMPL_SALARY* calculation view and choose *Generate SELECT statement*.
   e) Choose *Run*.
   f) Note that the data from both sources is combined together in the result set (salaries for the years 2020 to 2023).

2. Adapt the default generated SELECT statement to return only the data for 2023, and execute it again.

Hint:
Because the SQL session is connected to your container schema, the schema name is not mandatory in the SQL statement and can be removed. You can also simplify the query by removing the *TOP 1000* clause because the source tables only contain a few records.

After your modifications, the code should look like the following example:

```
SELECT "Year", "EmployeeID", SUM("TotalSalary")
FROM "HC::CVC_EMPL_SALARY"
WHERE "Year" = 2023
GROUP BY "Year", "EmployeeID";
```

a) Remove the `TOP 1000` clause.

b) Add the following *WHERE* clause to the SQL query, between the *FROM* and the *GROUP BY* lines.

```
WHERE "Year" = 2023
```

c) Optionally, remove the container schema name `"HC300_A####_HDI_DB_1".`, including the **.** (dot) separator.

d) To execute the query, choose *Run*.

e) Check that the query result is now filtered, showing only the data for 2023.

3. Analyze the SQL query. Enter **SQLA** as prefix.

a) Select the *Analyze* drop down list and choose *Generate SQL Analyzer Plan File*.

b) Enter **SQLA** as *File Name Prefix* and choose *Save*.

Result
The SQL plan file opens in the *HANA SQL Analyzer* view.

c) Look into the *DATA USAGE* block for the *Accessed Tables* number.

How many tables have been used during the execution of the query?

2 tables

What does it mean?

The two tables are scanned to check whether they contain data for 2023 but scanning the historical table is wasteful as there are no valid records found.

Task 3: Implement a Union Pruning to Optimize your Calculation View
You want to optimize this calculation view so that, whenever possible, the non-relevant data sources of the union node are pruned. For example, when querying data for 2022 or 2023, the historical data source should not be scanned at all because by design, it does not contain any valid data for these years.

> Note:
> To be able to compare the original view and the optimized one if needed, you will define the pruning in a **copy** of the original calculation view.

1. To make your screen cleaner, close all open tabs in Business Application Studio.
   a) Right-click any open tab and choose *Close All*.
2. Copy the original Calculation View *CVC_EMPL_SALARY* from the *resources* folder to your *exercises* folder. Rename the copy **`CVC_EMPL_SALARY_PRUNING`**.

> Hint:
> Check that the runtime name of the calculation view is also updated by opening the *Semantics* → *View Properties* → *General* window and checking the *Name* property.

   a) Come back to the *Explorer* view, select the *CVC_EMPL_SALARY.hdbcalculationview* from your *resources* folder and press **`Ctrl+C`** *(Copy)*.
   b) Select the *exercises* folder and press **`Ctrl+V`** *(Paste)*.
   c) Right-click on the copied calculation view and choose *Rename*.
   d) Enter **`_PRUNING`** after `CVC_EMPL_SALARY` and press *Enter*.
3. In the new (copied) calculation view, add a pruning configuration table with the name *Empl_Salary_Pruning*.

> Caution:
> Do NOT define the namespace (the namespace, *HC*, will be added automatically to your table identifier upon deployment).

Define the pruning conditions as follows:

Table 6: Pruning Configuration Entries

Input	Column	Operators	Low Value
HC::Empl_Salary_Current	Year	>=	2022
HC::Empl_Salary_Historical	Year	<	2022

> Caution:
> The pruning configuration entries define which value (or range of values) of a specified attribute CAN be found in each of the data sources.

   a) If needed, select the calculation view *CVC_EMPL_SALARY_PRUNING*.
   b) Choose *Expand Details Panel*.

c) In the *Semantics* node, select the *View Properties* tab.

d) In the *Advanced* tab, next to the *Pruning Configuration Table* field, choose **+** *Create Table*.

e) In the *Table Name* field, enter **`Empl_Salary_Pruning`**.

f) In the *Pruning Configuration Entries* table, choose **+** *Add* and define the condition for the *Current* data source, as per the table provided.

g) Repeat the previous step for the *Historical* data source.

h) Choose *Create and Build*.

Result

Three design-time files are created in your *exercises* folder with the name *Empl_Salary_Pruning* and different extensions.

4. Deploy the Calculation View.

Check the build status.

a) Choose the *Deploy* icon for your *CVC_EMPL_SALARY_PRUNING* calculation view..

b) In the *Console* pane, check that the deployment completes successfully.

5. From the default SELECT statement generated for the new calculation view, adapt the SQL query to return only the data for 2023, as you did in Task 2, and analyze the query.

After your modifications, the code should look as follows (note that it now refers to the new calculation view):

```
SELECT "Year", "EmployeeID", SUM("TotalSalary")
FROM "HC::CVC_EMPL_SALARY_PRUNING"
WHERE "Year" = 2023
GROUP BY "Year", "EmployeeID";
```

a) Go to *SAP HANA DATABASE EXPLORER*.

b) Select your HDI container and select *Columns Views*.

c) Right click on the HC::CVC_EMPL_SALARY_PRUNING view and choose *Generate SELECT statement*.

d) Remove the `TOP 1000` clause.

e) Add the following *WHERE* clause to the SQL query, between the *FROM* and the *GROUP BY* lines:

```
WHERE "Year" = 2023
```

f) Optionally, remove the container schema name `"HC300_A####_HDI_DB_1".`, including the **.** (dot) separator.

g) Select *Run* to execute the Query. Only data for 2023 should display.

h) Select the *Analyze* drop down list and choose *Generate SQL Analyzer Plan File*.

i) Enter **`SQLA`** as *File Name Prefix* and choose *Save*.

j) Look into the *DATA USAGE* block for the *Accessed Tables* number.

How many tables have been used during the execution of the query?

This time, only one table has been used. This shows that implementing pruning in the calculation view has optimized the behavior of the union node. Data sources are now completely excluded from the calculation scenario as soon as a filtering condition (here, the *WHERE* clause) does not match the pruning configuration entry for this data source.

6. Close all the open tabs in Business Application Studio and go back to the *Explorer* view.
   a) Right-click any open tab and choose *Close All*. Don't save anything.
   b) Select the *Explorer* view.

Unit 15
Exercise 34

Implement Static Cache

Exercise Objectives

After completing this exercise, you will be able to:

- Implement Static Cache in a Calculation View.

Business Example

The most popular calculation views are consumed by many queries which frequently request the same data. But performance is poor and you would like to improve this. You would like to implement static cache to find out if this improves performance.

Overview of Exercise Tasks

- Task 1: Create a Calculation View which Caches All Data
- Task 2: Define Specific Columns for Cache

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a Calculation View which Caches All Data

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Create a calculation view using the following settings:

Setting	Value
Name	CVC_STATIC_CACHE
Label	Static Cache
Data Category	CUBE
With Star Join	[Not selected]

3. Assign the table *CACHED_TABLE* to the default aggregation node.
4. Map all the columns of the table.
5. Enable the static cache.
6. Deploy the calculation view.
7. Execute a data preview of the calculation view in order to fill the cache.
8. Use the *Explain Plan* tool to confirm an identical query would use the cache.

Task 2: Define Specific Columns for Cache

1. Adjust the calculation view *CVC_STATIC_CACHE* by specifying that only the columns *COLUMN1* and *MEASURE* are to be cached.
2. Deploy the calculation view.
3. Execute a data preview of the calculation view in order to fill the cache.
4. Use the *Explain Plan* tool to confirm the default query that requests all columns, would not use the cache and would access the source table.
5. Adjust the default SQL query so that it requests only the columns that are cached.
6. Close all tabs in the SAP Business Application Studio.

Unit 15
Solution 34

Implement Static Cache

Exercise Objectives

After completing this exercise, you will be able to:

- Implement Static Cache in a Calculation View.

Business Example

The most popular calculation views are consumed by many queries which frequently request the same data. But performance is poor and you would like to improve this. You would like to implement static cache to find out if this improves performance.

Overview of Exercise Tasks

- Task 1: Create a Calculation View which Caches All Data
- Task 2: Define Specific Columns for Cache

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create a Calculation View which Caches All Data

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Create a calculation view using the following settings:

Setting	Value
Name	CVC_STATIC_CACHE
Label	Static Cache
Data Category	CUBE
With Star Join	[Not selected]

   a) Right-click your *exercises* folder and choose *New File*.

   b) Enter **`CVC_STATIC_CACHE.hdbcalculationview`** as file name and press *Enter*.

   c) Use the table above to complete the fields and press *Create*.

3. Assign the table *CACHED_TABLE* to the default aggregation node.
   a) Click on the '+' icon alongside the aggregation node and enter **`cached`** so that the *CACHED_TABLE* table appears in the results.

   b) Select *HC::CACHED_TABLE* and choose *Finish*.

4. Map all the columns of the table.

a) In the *Aggregation* node, select the mapping tab and drag all three columns from the *Data Sources* pane on the left side to the *Output Columns* pane on the right side.

5. Enable the static cache.
   a) Double-click on the *Semantics* node.
   b) Click *View Properties*.
   c) Click *Static Cache*.
   d) Click on the check-box *Enable Cache*.
6. Deploy the calculation view.
   a) Choose the *Deploy* icon for your *CVC_STATIC_CACHE* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
7. Execute a data preview of the calculation view in order to fill the cache.
   a) Right-click the calculation view *CVC_STATIC_CACHE* in the project structure and choose *Data Preview* . The cache is now filled with the complete data set from the calculation view.
8. Use the *Explain Plan* tool to confirm an identical query would use the cache.
   a) Switch to the *Raw Data* tab.
   b) Select *Edit SQL Statement in SQL Console* (the SQL button with the pencil)
   c) Choose *Analyze* → *Explain Plan*.
   d) In the *Plan* tab notice *RESULT CACHE* appears at the bottom of the operator list. This confirms the cache would be used and the database table would not be accessed.
   e) Close the SQL and data preview tabs.

Task 2: Define Specific Columns for Cache

1. Adjust the calculation view *CVC_STATIC_CACHE* by specifying that only the columns *COLUMN1* and *MEASURE* are to be cached.
   a) In the definition of the *CVC_STATIC_CACHE* calculation view, double-click on the *Semantics* node.
   b) Click *View Properties*.
   c) Click *Static Cache*.
   d) in the *Columns* section, click the **+** button to add a new item.
   e) Click on the drop-down selector for the new item.
   f) Select the check-box for the columns *COLUMN1* and *MEASURE* and choose *Select* to return to the main screen.
2. Deploy the calculation view.
   a) Choose the *Deploy* icon for your *CVC_STATIC_CACHE* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
3. Execute a data preview of the calculation view in order to fill the cache.

a) Right-click the calculation view *CVC_STATIC_CACHE* in the project structure and choose *Data Preview*. The cache is now filled with a reduced data set.

4. Use the *Explain Plan* tool to confirm the default query that requests all columns, would not use the cache and would access the source table.
   a) Switch to the *Raw Data* tab.
   b) Select *Edit SQL Statement* (the SQL button with the pencil)
   c) Choose *Analyze* → *Explain Plan*.
   d) In the *Plan* tab notice *TABLE SCAN* appears at the bottom of the operator list. This confirms the cache would not be used and instead, the source table would be accessed because the column *COLUMN2* was requested in the default query, but is not included in the cache.
5. Adjust the default SQL query so that it requests only the columns that are cached.
   a) Edit the query by using double-hyphens to ignore the SELECT and GROUP BY clauses for the column *COLUMN2*. The final code should look like this:

```
SELECT TOP 1000
"COLUMN1",
-- "COLUMN2",
SUM("MEASURE") AS "MEASURE"
FROM "HC300_A####_HDI_DB_1"."HC::CVC_STATIC_CACHE"
GROUP BY "COLUMN1" --, "COLUMN2"
```

Note:
You can also remove the references to *COLUMN2* instead of commenting them.

   b) Choose *Analyze* → *Explain Plan*.
   c) Note that in the *Plan* tab notice *RESULT CACHE* appears at the bottom of the operator list. This confirms the cache would be used because only the columns that are part of the cache definition were requested by the query.
6. Close all tabs in the SAP Business Application Studio.
   a) Right-click any open tab and choose *Close All*.

Unit 15
Exercise 35

Create a Snapshot

Exercise Objectives

After completing this exercise, you will be able to:

- Define a snapshot in a calculation view
- Generate and customize an interface view

Business Example

You have to create a calculation view to analyze sales data starting from 2019. You know that you are often using data from 2019 and 2020 to show flat screen sales and so decide to improve performance for those analysis by creating a snapshot on the main calculation view.

Overview of Exercise Tasks

- Task 1: Create the base Calculation View
- Task 2: Create a Snapshot in your Calculation View
- Task 3: Generate and customize the Interface View

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create the base Calculation View

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *Exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVC_SALES_SNAPSHOTS`**
Label	**`Sales Snapshots`**
Data Category	*CUBE*
With star join	[Deselected]

3. In the *Aggregation* node, add the *CVC_SALES* data source:
4. All all columns to the output
5. Deploy the *CVC_SALES_SNAPSHOTS* calculation view.
   Check the deployment status.
6. Preview the calculation view data in the *Analysis* tab and show net amounts by year.

Result

There are four different years from 2019 to 2022.

7. Change the Label axis from *YEAR* to *CATEGORY*.

   Result

   There are nineteen different categories. The largest amounts come from the **Flat screens** and **Handhelds** categories.

8. Close the Data Preview tab.

Task 2: Create a Snapshot in your Calculation View

You are often retrieving data for the Years 2019 and 2020, for the category Flat screens and so would like to improve performance by storing this historical data in a snapshot.

1. Create a new Snapshot Query in your *CVC_SALES_SNAPSHOTS* calculation view, with the following properties :

Field	Value
Query Name	**`Q_2019_2020_Flatscreens`**
Create Snapshot After Deployment Automatically	Selected
Keep Snapshot During Re-deployment	Selected
Mode of Generated Procedure	**Definer**
Interface View Name	**`I_CVC_SALES_SNAPHOTS`**
Query	`SELECT "PRODUCT_ID" , "YEAR" , "CATEGORY" , "CURRENCY_CODE" , SUM("GROSS_AMOUNT") AS "GROSS_AMOUNT" , SUM("NET_AMOUNT") AS "NET_AMOUNT" FROM "HC::CVC_SALES_SNAPSHOTS" WHERE "YEAR" in ('2019','2020') AND "CATEGORY" = 'Flat screens' GROUP BY "PRODUCT_ID" , "YEAR" , "CATEGORY" , "CURRENCY_CODE"`

2. Deploy the *CVC_SALES_SNAPSHOTS* calculation view.

   Check the deployment status.

3. Display the new snapshot table content.

   Result

   The table is filled with the 2019 and 2020 Flat screens sales.

4. Close the Data Preview tab.

Task 3: Generate and customize the Interface View

You want to be able to use the same calculation view to query data either from the whole CVC_SALES data source or from the snapshot table. You then decide to generate the proposed interface view.

1. Generate the Interface View for your *Q_2019_2020_Flatscreens* query.
2. Open the calculation view and change the mapping values for the *SOURCE* column as proposed in the following table :

Old value	New value
BASE	ALL
SNAPSHOT	FLATSCREEN_HISTORY

3. Change the value list of the input parameter to match the SOURCE values. Don't forget to also change the default value.
4. Deploy the *I_CVC_SALES_SNAPSHOTS* calculation view.

   Check the deployment status.
5. Display the *I_CVC_SALES_SNAPSHOTS* data preview in *Raw Data* tab, first with historical data then with all data.
6. Close all the open tabs in Business Application Studio.

Unit 15
Solution 35

Create a Snapshot

Exercise Objectives

After completing this exercise, you will be able to:

- Define a snapshot in a calculation view
- Generate and customize an interface view

Business Example

You have to create a calculation view to analyze sales data starting from 2019. You know that you are often using data from 2019 and 2020 to show flat screen sales and so decide to improve performance for those analysis by creating a snapshot on the main calculation view.

Overview of Exercise Tasks

- Task 1: Create the base Calculation View
- Task 2: Create a Snapshot in your Calculation View
- Task 3: Generate and customize the Interface View

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Create the base Calculation View

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. In your *Exercises* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVC_SALES_SNAPSHOTS`**
Label	**`Sales Snapshots`**
Data Category	*CUBE*
With star join	[Deselected]

   a) In the *EXPLORER* tree, right-click the folder *HC300_A####* → *db* → *src* → *exercises* and choose *New File*.

   b) Enter **`CVC_SALES_SNAPSHOTS.hdbcalculationview`** as file name and press *Enter*.

   c) Enter the calculation view label and other properties as specified in the table.

   d) Choose *Create*.

3. In the *Aggregation* node, add the *CVC_SALES* data source:
   a) Select the *Aggregation* node.
   b) Choose **+**, *Add Data Source*.
   c) In the *Find Data Sources* window, click the search field and enter **`CVC_SALES`**.
   d) In the search results, select the calculation view *HC::CVC_SALES*.
   e) Choose *Finish*.
4. All all columns to the output
   a) On the *Mapping* tab, select the *HC::CVC_SALES* entry and choose the *Add To Output* button..
5. Deploy the *CVC_SALES_SNAPSHOTS* calculation view.
   Check the deployment status.
   a) Choose the Deploy icon for your *CVC_SALES_SNAPSHOTS* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
6. Preview the calculation view data in the *Analysis* tab and show net amounts by year.
   a) Right-click the *CVC_SALES_SNAPSHOTS* Calculation View and choose *Data Preview*.
   b) In the *Analysis* tab drag the *NET_AMOUNT* column to the *Value Axis* and *YEAR* to the *Label Axis*.

   Result
   There are four different years from 2019 to 2022.
7. Change the Label axis from *YEAR* to *CATEGORY*.
   a) In the Label Axis pane, delete the *YEAR* column by choosing the cross icon on the right.
   b) Drag the *CATEGORY* to the *Label Axis*.

   Result
   There are nineteen different categories. The largest amounts come from the **Flat screens** and **Handhelds** categories.
8. Close the Data Preview tab.

Task 2: Create a Snapshot in your Calculation View
You are often retrieving data for the Years 2019 and 2020, for the category Flat screens and so would like to improve performance by storing this historical data in a snapshot.

1. Create a new Snapshot Query in your *CVC_SALES_SNAPSHOTS* calculation view, with the following properties :

Field	Value
Query Name	**`Q_2019_2020_Flatscreens`**
Create Snapshot After Deployment Auto-matically	Selected
Keep Snapshot During Re-deployment	Selected
Mode of Generated Procedure	**Definer**

Field	Value
Interface View Name	**I_CVC_SALES_SNAPHOTS**
Query	`SELECT "PRODUCT_ID" , "YEAR" , "CATEGORY" , "CURRENCY_CODE" , SUM("GROSS_AMOUNT") AS "GROSS_AMOUNT" , SUM("NET_AMOUNT") AS "NET_AMOUNT" FROM "HC::CVC_SALES_SNAPSHOTS" WHERE "YEAR" in ('2019','2020') AND "CATEGORY" = 'Flat screens' GROUP BY "PRODUCT_ID" , "YEAR" , "CATEGORY" , "CURRENCY_CODE"`

a) Select the View Properties tab in the semantics node.

b) Select the Snapshots tab.

c) Choose the Add icon to create a new Query.

d) Select Query_1 to show the details.

e) Enter the properties values as shown in the above table.

2. Deploy the *CVC_SALES_SNAPSHOTS* calculation view.
   Check the deployment status.
   a) Choose the *Deploy* icon for your *CVC_SALES_SNAPSHOTS* calculation view..
   b) In the *Console* pane, check that the deployment completes successfully.
3. Display the new snapshot table content.
   a) Select the *SAP HANA Database Explorer* tool.
   b) Select *Tables* in your HDI container.
   c) look for the *HC::CVC_SALES_SNAPSHOTS/Q_2019_2020_Flatscreens/SNAP/SNAPSHOT* table.
   d) Right-click and choose *Open Data*.

   Result
   The table is filled with the 2019 and 2020 Flat screens sales.
4. Close the Data Preview tab.

Task 3: Generate and customize the Interface View
You want to be able to use the same calculation view to query data either from the whole CVC_SALES data source or from the snapshot table. You then decide to generate the proposed interface view.

1. Generate the Interface View for your *Q_2019_2020_Flatscreens* query.

a) Go back to the definition of the *Q_2019_2020_Flatscreens* query.

b) Choose the *Generate interface calculation view* button on the right of the interface view name.

c) Explore the new *I_CVC_SALES_SNAPSHOTS* calculation view definition.

2. Open the calculation view and change the mapping values for the *SOURCE* column as proposed in the following table :

Old value	New value
BASE	ALL
SNAPSHOT	FLATSCREEN_HISTORY

a) If not yet done, open the *I_CVC_SALES_SNAPSHOTS* calculation view.

b) Select the *Union_1* node.

c) Select the *Mapping* tab.

d) In the *Output Columns*, right-click on the *SOURCE* column and choose *Manage Mappings*.

e) In the *Manage Mapping* table, change the values as shown in the above table.

f) Choose *OK*.

3. Change the value list of the input parameter to match the SOURCE values. Don't forget to also change the default value.

a) Select the *Parameters* tab.

b) Select the *I_SOURCE* parameter to show the details.

c) Expand the *Parameter Type - Static List* pane.

d) In the *Values* table, change the *Value* column as per the previous table.

e) Reduce the *Parameter Type - Static List* pane and expand the *Default Values* pane.

f) Change the constant to **`FLATSCREEN_HISTORY`**.

g) Choose *Back*.

4. Deploy the *I_CVC_SALES_SNAPSHOTS* calculation view.

Check the deployment status.

a) Choose the *Deploy* icon for your *I_CVC_SALES_SNAPSHOTS* calculation view..

b) In the *Console* pane, check that the deployment completes successfully.

5. Display the *I_CVC_SALES_SNAPSHOTS* data preview in *Raw Data* tab, first with historical data then with all data.

a) Right-click the *I_CVC_SALES_SNAPSHOTS* Calculation View and choose *Data Preview*.

b) Keep the parameter default value and choose *Open Content*.

c) Select the *Raw Data* tab. You should see only the 2019 and 2020 flatscreen sales.

d) Close the data tab.

e) change the value of the input parameter to **ALL**.

f) Choose *Open Content*.

g) Select the *Raw Data* tab. You then see all sales.

6. Close all the open tabs in Business Application Studio.

a) Right-click any open tab and choose *Close All*.

Unit 16
Exercise 36

Audit Calculation Views and their Dependencies

Exercise Objectives

After completing this exercise, you will be able to:

- Identify the origin of private columns in a calculation view scenario.
- List the dependent models that might be impacted by a change to a calculation view
- List all the tables used by a calculation view

Business Example

You are working on a calculation view with dimensions and need to check the origin of some of its output columns.

You have also been asked to modify a dimension calculation view, but would like to double-check which models, if any, reference this dimension view and might be impacted.

Overview of Exercise Tasks

- Task 1: Analyze the Lineage of a Column Inside a Calculation View
- Task 2: Analyze the Impact of a View Modification on Other Views
- Task 3 : Analyze the Data Lineage of a Calculation View

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Analyze the Lineage of a Column Inside a Calculation View

1. Open the calculation view *resources* → *CVCS_SO* and show the lineage for the column *SO_ID* (Sales Order ID).
2. Observe the data flow scenario in the left pane.

   From which node and data source does the column *SO_ID* originate?

   _____

   _____

   _____
3. Confirm the origin data source by displaying the *Columns* tab of the *Join_1* node.
4. Now, display the column lineage for the column GROSS_AMOUNT.
   Does the *GROSS_AMOUNT* come from the *HC::SNWD_SO* table or the *HC::SNWD_SO_I* table?

5. Close the *CVC_SO* calculation view.

Task 2: Analyze the Impact of a View Modification on Other Views

You have been asked to modify the Business Partner view `resources/CVD_BP2`. Before proceeding, you want to check which dependent models will be impacted and make sure your changes will not change the behavior of these dependent models.

1. Display the impact analysis for the dimension view *CVD_BP2*.
2. What other models reference the *CVD_BP2* view?

Task 3: Analyze the Data Lineage of a Calculation View

You have been asked to create a copy of a view into another system and so need to know if this other system does have all the required tables.

1. Display the data lineage for the view *CVCS_SO*.

   Which tables are used by CVCS_SO ?

   _____

   _____

   _____

2. Close all the open tabs in Business Application Studio.

Unit 16
Solution 36

Audit Calculation Views and their Dependencies

Exercise Objectives

After completing this exercise, you will be able to:

- Identify the origin of private columns in a calculation view scenario.
- List the dependent models that might be impacted by a change to a calculation view
- List all the tables used by a calculation view

Business Example

You are working on a calculation view with dimensions and need to check the origin of some of its output columns.

You have also been asked to modify a dimension calculation view, but would like to double-check which models, if any, reference this dimension view and might be impacted.

Overview of Exercise Tasks

- Task 1: Analyze the Lineage of a Column Inside a Calculation View
- Task 2: Analyze the Impact of a View Modification on Other Views
- Task 3 : Analyze the Data Lineage of a Calculation View

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Analyze the Lineage of a Column Inside a Calculation View

1. Open the calculation view *resources* → *CVCS_SO* and show the lineage for the column *SO_ID* (Sales Order ID).
   a) In your *HC300_A####* project, navigate to the resources folder.
   b) Select the *CVCS_SO* calculation view.
   c) Double-click the *Semantics* node to display the *Columns* tab.
   d) Select the *SO_ID* column and click *Show Lineage* icon.
      Notice at the bottom of the screen, you now see *Showing Column Lineage* with an option to *Exit* this mode.
2. Observe the data flow scenario in the left pane.

From which node and data source does the column *SO_ID* originate?

The *SO_ID* column "enters" the calculation scenario in the *Join_1* node. Because the lineage does not reach the *Projection_1* node below, we can conclude that the *SO_ID* column comes from the *HC::SNWD_SO* table or is a calculated column defined in *Join_1*.

3. Confirm the origin data source by displaying the *Columns* tab of the *Join_1* node.
   a) Select the *Join_1* node.
   b) Display the *Columns* tab.
   c) Note that the *HC::SNWD_SO.SO_ID* column is highlighted in orange.
4. Now, display the column lineage for the column GROSS_AMOUNT.
   Does the *GROSS_AMOUNT* come from the *HC::SNWD_SO* table or the *HC::SNWD_SO_I* table?
   a) Click the *Exit* option at the bottom of the screen to leave the column lineage mode.
   b) In the *Semantics* node, deselect, if needed, the *SO_ID* column.
   c) Select the *GROSS_AMOUNT* column and click the *Show Lineage* icon.
   d) Note that the lineage goes down to the *Projection_1* node, which means that the *GROSS_AMOUNT* column comes from the *HC::SNWD_SO_I* table; or it might be a calculated column defined in this node.
5. Close the *CVC_SO* calculation view.
   a) Right-click the open tab for *CVC_SO* and choose *Close*.

Task 2: Analyze the Impact of a View Modification on Other Views

You have been asked to modify the Business Partner view `resources/CVD_BP2`. Before proceeding, you want to check which dependent models will be impacted and make sure your changes will not change the behavior of these dependent models.

1. Display the impact analysis for the dimension view *CVD_BP2*.
   a) In your *HC300_A####* project, right-click the `resources/CVD_BP2` view and choose *Open Impact Analysis*.
2. What other models reference the *CVD_BP2* view?
   a) Choose the *Expand* button if the referencing calculation views are not yet displayed.
   b) The *CVD_BP2* is consumed by the *CVCS_PO3_AP*, the *CVCS_SO* , the *CVC_NB_SO*, the *CVCS_PO2_00*, and the *CVC_SW_SO* views.

Task 3: Analyze the Data Lineage of a Calculation View

You have been asked to create a copy of a view into another system and so need to know if this other system does have all the required tables.

1. Display the data lineage for the view *CVCS_SO*.
   a) In your *HC300_A####* project, right-click the `resources/CVCS_SO` view and choose *Show Data Lineage*.
   b) Choose the *Expand* button for all sources.

Which tables are used by CVCS_SO ?

CVCS_SO is using SNWD_SO and SNWD_SO_I. Through its dimensions it is also using the following tables : SNWD_PD, COUNTRIES, SNWD_CONTACT, SNWD_AD, SNWD_BP and SNWD_BP_EM.

2. Close all the open tabs in Business Application Studio.
   a) Right-click any open tab and choose *Close All*.

Unit 17
Exercise 37

Import, Rename and Copy Models

Exercise Objectives

After completing this exercise, you will be able to:

- Import models
- Manage models deployment
- Rename a model
- Identify common deployment errors when importing, moving, and copying models

Business Example

You are working as a modeler in SAP Business Application Studio, and need to make a few adjustments to the names or locations of your models, and also want to make a copy of models for testing purposes. Before that, you need to understand how to handle duplicate model names, multiple namespaces, and more generally avoid the most common deployment issues that can occur when deploying moved or copied objects.

Overview of Exercise Tasks

- Task 1: Import and Build Models
- Task 2: Rename a View
- Task 3: Import a New Folder with a Namespace Specification
- Task 4: Move a Model

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import and Build Models

You will import into your project a folder that contains two calculation views. One of these views,*CVCS_SO*, has the same name as an existing model in your `resources` folder, so the observations you will make on this imported view will be the same as if you had copied the view.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Clear the SAP Business Application Studio console.
3. In the *Explorer* view, drag and drop the *import* folder from your course files folder *HC300* into your *HC300_A####* → *db* → *src* → *exercises* project folder.
4. Deploy the *import* folder.

What do you observe?

5. Analyze the root cause by checking the log from the start of the deployment until you see an error report.

```
Error: "calc.scenario://HC::CVCS_SO": the object cannot be provided
more than once [8212002]
"src/models/exercises/import/CVCS_SO.hdbcalculationview": the file
would provide it
"src/models/resources/CVCS_SO.hdbcalculationview": the deployed file
already provides it
```

Why did the deployment fail?

6. In the *SAP HANA Database Explorer* view, check whether the view *CVD_PD3* has been created in your *HC300_A####_HDI_CONTAINER*

Has the *CVD_PD3* view been created? Why?

7. Try to deploy the calculation view *CVD_PD3* separately.

Has the deployment been successful? Why?

Task 2: Rename a View

To solve the issue for the view *CVCS_SO*, you will rename it.

1. Rename the imported *CVCS_SO* Calculation View as **CVCS_SO2**. You will only rename the design-time file and adjust the runtime object name accordingly.
2. Deploy the calculation view.
3. In the *SAP HANA Database Explorer* view, confirm that the view *CVCS_SO2* is now created in your *HC300_A####_HDI_CONTAINER*.
4. Clear the filter and close any open tabs in the SAP Business Application Studio.

Task 3: Import a New Folder with a Namespace Specification

You want to learn more about the way different namespaces impact the naming of runtime objects and the best practices to keep your content correctly organized.

1. Switch back to the *Explorer* view.

2. Clear the SAP Business Application Studio console.
3. In the *Explorer* view, drag and drop the *import2* folder from your course files folder *HC300* into your *HC300_A####* → *db* → *src* → *exercises* project folder.
4. Deploy the *import2* folder.

   What do you observe?

   _____

   _____

   _____

5. Observe carefully the content of the *import2* folder.

   The imported folder contains a *CVCS_SO.hdbcalculationview* file, but this time the deployment was successful. Can you explain why?

   _____

   _____

   _____

Task 4: Move a Model

You want to move the *CVCS_SO2* view from the import to the *import2* folder.

Note:
The calculation view *CVCS_SO2* is not referenced by any other object.

1. Clear the SAP Business Application Studio console.
2. Move the calculation view *CVCS_SO2* from the *import* to the *import2* folder.
   Confirm the view renaming in the confirmation dialog box (choose *Yes*) .

Hint:
You can use the Cut/Paste shortcuts or the context menu.

3. Deploy the calculation view.
4. In the *SAP HANA Database Explorer* view, confirm that the view *HC.import2::CVCS_SO2* is now created in your *HC300_A####_HDI_CONTAINER*.

   Was the *HC::CVCS_SO2* removed from the list of column views? Can you explain why?

   _____

   _____

   _____

5. To finalize the move of the view, that is, delete the original runtime object, deploy the *import* folder.
   Check the deployment status.

6. In the *SAP HANA Database Explorer* view, confirm that the view *HC::CVCS_SO2* has been deleted.
7. Clear the filter and close any open tabs in the SAP Business Application Studio.

Unit 17
Solution 37

Import, Rename and Copy Models

Exercise Objectives

After completing this exercise, you will be able to:

- Import models
- Manage models deployment
- Rename a model
- Identify common deployment errors when importing, moving, and copying models

Business Example

You are working as a modeler in SAP Business Application Studio, and need to make a few adjustments to the names or locations of your models, and also want to make a copy of models for testing purposes. Before that, you need to understand how to handle duplicate model names, multiple namespaces, and more generally avoid the most common deployment issues that can occur when deploying moved or copied objects.

Overview of Exercise Tasks

- Task 1: Import and Build Models
- Task 2: Rename a View
- Task 3: Import a New Folder with a Namespace Specification
- Task 4: Move a Model

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import and Build Models

You will import into your project a folder that contains two calculation views. One of these views,*CVCS_SO*, has the same name as an existing model in your `resources` folder, so the observations you will make on this imported view will be the same as if you had copied the view.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Clear the SAP Business Application Studio console.
   a) In the *Terminal* pane, choose the *Views and More Actions* icon (three dots) then choose *Clear Terminal*.
3. In the *Explorer* view, drag and drop the *import* folder from your course files folder *HC300* into your *HC300_A#### → db → src → exercises* project folder.

a) On your device, open the *HC300* folder and select the *import* folder.

b) Drag and drop it onto the *exercises* folder in Business Application Studio.

Result

An *import* folder is created in your project. It will be used specifically for this exercise.

4. Deploy the *import* folder.

a) In the *SAP HANA Projects* sub-pane of the *Explorer* view, choose the *Deploy* icon for the *import* folder.

b) In the *Console* pane, check that the deployment completes successfully.

What do you observe?

The deployment fails.

5. Analyze the root cause by checking the log from the start of the deployment until you see an error report.

```
Error: "calc.scenario://HC::CVCS_SO": the object cannot be provided
more than once [8212002]
"src/models/exercises/import/CVCS_SO.hdbcalculationview": the file
would provide it
"src/models/resources/CVCS_SO.hdbcalculationview": the deployed file
already provides it
```

Why did the deployment fail?

A *CVCS_SO* object already exists with the *HC* namespace.

6. In the *SAP HANA Database Explorer* view, check whether the view *CVD_PD3* has been created in your *HC300_A####_HDI_CONTAINER*

a) Switch to the *SAP HANA Database Explorer* view.

b) Select your container *HC300_A####_HDI_CONTAINER* and choose *Column Views*.

c) select the *Apply Filter* icon, enter **CVD_PD3** and press *Enter*.

Has the *CVD_PD3* view been created? Why?

The *CVD_PD3* view has not been deployed. This is because when you request the deployment of an entire folder in your project, either all the runtime files can be deployed successfully, or no file at all is deployed.

7. Try to deploy the calculation view *CVD_PD3* separately.

a) Go back to the *Explorer* view.

b) Choose the *Deploy* icon for the *CVD_PD3* calculation view.

c) In the *Console* pane, check that the deployment completes successfully.

Has the deployment been successful? Why?

Yes. The *CVD_PD3* view has been created because it was the only design-time file included in the scope of the deployment, and this one does not generate any deployment error (conflict in runtime object name or other deployment issue).

Task 2: Rename a View

To solve the issue for the view *CVCS_SO*, you will rename it.

1. Rename the imported *CVCS_SO* Calculation View as **`CVCS_SO2`**. You will only rename the design-time file and adjust the runtime object name accordingly.
   a) Right-click the *CVCS_SO* Calculation View and choose *Rename*.
   b) Modify the new name to **`CVCS_SO2.hdbcalculationview`** and press *Enter*.
   c) Choose *Yes* in the *Confirmation* dialog.

Note:
If you don't get a confirmation, the runtime name of the calculation view should normally be adjusted. You can check that in the general properties of the calculation view, in the Semantics.

2. Deploy the calculation view.
   a) Choose the *Deploy* icon for the *CVCS_SO2* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
3. In the *SAP HANA Database Explorer* view, confirm that the view *CVCS_SO2* is now created in your *HC300_A####_HDI_CONTAINER*.
   a) Switch to the *SAP HANA Database Explorer* view.
   b) Select your container *HC300_A####_HDI_CONTAINER* and choose *Column Views*.
   c) select the *Apply Filter* icon, enter **`CVCS_SO2`** and press *Enter*.
4. Clear the filter and close any open tabs in the SAP Business Application Studio.
   a) select the *Apply Filter* icon, remove any text and press *Enter*.
   b) Right-click any open tab and choose *Close All*.

Task 3: Import a New Folder with a Namespace Specification

You want to learn more about the way different namespaces impact the naming of runtime objects and the best practices to keep your content correctly organized.

1. Switch back to the *Explorer* view.
2. Clear the SAP Business Application Studio console.
   a) In the *Terminal* pane, choose the *Views and More Actions* icon (three dots) then choose *Clear Terminal*.
3. In the *Explorer* view, drag and drop the *import2* folder from your course files folder *HC300* into your *HC300_A#### → db → src → exercises* project folder.
   a) On your device, open the *HC300* folder and select the *import2* folder.

b) Drag and drop it onto the *exercises* folder in Business Application Studio.

Result

An *import2* folder is created in your project. It will be used specifically for this exercise.

4. Deploy the *import2* folder.

a) Choose the *Deploy* icon for the *import2* folder.

b) In the *Console* pane, check that the deployment completes successfully.

What do you observe?

The deployment is successful.

5. Observe carefully the content of the *import2* folder.

The imported folder contains a *CVCS_SO.hdbcalculationview* file, but this time the deployment was successful. Can you explain why?

The *import2* folder contains a *.hdinamespace* file with a namespace prefix *HC.import2*, which is different from the namespace *HC* assigned to the original *CVCS_SO* runtime object. Besides, the new CVCS_SO view has the corresponding namespace in its design. So the runtime object identifier is unique, even if the file has the same name.

a) Select the *import2* → *.hdinamespace* file and observe its content.

b) Select the *import2* → *CVCS_SO* calculation view and, in the *View Properties* tab of the *Semantics* node, check the assigned namespace in the *Name* field. This should be *HC.import2*.

c) Alternatively, you can right-click the *import2* → *CVCS_SO.hdbcalculationview* and choose *Open With* → *Text Editor*. Then check the identifier of the runtime view in the second row.

Note:
The *.hdinamespace* file must be deployed before, or at least at the same time as, the content of the corresponding folder. For example, if you had deployed the *import2* → *CVCS_SO* calculation view alone just after the import, it would have failed.

Task 4: Move a Model

You want to move the *CVCS_SO2* view from the import to the *import2* folder.

Note:
The calculation view *CVCS_SO2* is not referenced by any other object.

1. Clear the SAP Business Application Studio console.

a) In the *Terminal* pane, choose the *Views and More Actions* icon (three dots) then choose *Clear Terminal*.

2. Move the calculation view *CVCS_SO2* from the *import* to the *import2* folder.
   Confirm the view renaming in the confirmation dialog box (choose *Yes*) .

   Hint:
   You can use the Cut/Paste shortcuts or the context menu.

   a) In the *Explorer* view, select the *CVCS_SO2* calculation view from the *import* folder.

   b) Press **`Ctrl+X`** (Cut).

   c) Select the *import2* folder and press **`Ctrl+V`** (Paste).

   d) In the *Confirmation* dialog box, choose *Yes*.

3. Deploy the calculation view.
   a) Choose the *Deploy* icon for the *CVCS_SO2* calculation view.

   b) In the *Console* pane, check that the deployment completes successfully.

4. In the *SAP HANA Database Explorer* view, confirm that the view *HC.import2::CVCS_SO2* is now created in your *HC300_A####_HDI_CONTAINER*.
   a) Switch to the *SAP HANA Database Explorer* view.

   b) Select your container *HC300_A####_HDI_CONTAINER* and choose *Column Views*.

   c) select the *Apply Filter* icon, enter **`SO2`** and press *Enter*.

   d) Make sure there is a view *HC.import2::CVCS_SO2*.

   Was the *HC::CVCS_SO2* removed from the list of column views? Can you explain why?

   No, it was not. This is because deleting or moving a design-time file removes the corresponding runtime object only after its origin folder (or any parent folder) has been deployed successfully.

5. To finalize the move of the view, that is, delete the original runtime object, deploy the *import* folder.
   Check the deployment status.
   a) Come back to the *Explorer* view.

   b) Choose the *Deploy* icon for the *import2* folder.

   c) In the *Console* pane, check that the deployment completes successfully.

6. In the *SAP HANA Database Explorer* view, confirm that the view *HC::CVCS_SO2* has been deleted.
   a) Switch to the *SAP HANA Database Explorer* tool.

   b) Choose the *Refresh* icon of the *CATALOG BROWSER* pane.

Note:
Alternatively, If needed, select your container *HC300_A####_HDI_CONTAINER* and choose *Column Views* then select the *Apply Filter* icon, enter **SO2** and press *Enter*.

c) Note that the view *HC::CVCS_SO2* is no longer listed in the search results.

7. Clear the filter and close any open tabs in the SAP Business Application Studio.

a) select the *Apply Filter* icon, remove any text and press *Enter*.

b) Right-click any open tab and choose *Close All*.

Unit 18
Exercise 38

Create a New Project and an HDB Module

Exercise Objectives

After completing this exercise, you will be able to:

- Create a new project (MTA Application)
- Create a HDB module
- Deploy the project

Business Example

You have been working on an SAP HANA project as a modeler in SAP Business Application Studio for some time, and will soon work on a new project. Before that, you want to put into practice key concepts of MTA application, SAP HANA Database module, and namespace configuration, by creating a simple project and reviewing its key properties.

Note:
The scope of this course is restricted to the HDB Module, which means we cover only a part of what MTA projects encompass. To use the best project template for this need, we will use the *SAP HANA Database Project* project template.

Overview of Exercise Tasks

- Task 1: Create the Project and Database Module
- Task 2: Finalize the Project and Deploy the HDB Module

Note:
In this exercise, when values include `A####`, replace the characters with the user number assigned to you (one letter and four digits).

Prerequisites

Your SAP Business Application Studio development environment should have been set up.

Task 1: Create the Project and Database Module

1. If needed, launch SAP Business Application Studio and start your *HCMOD_A####* development space.
2. In the *Explorer* pane, create a new project based on the *SAP HANA Database Project* template, with the following properties:

Field	Value
Project Name	**`MyProject_A####`**

Field	Value
Module Name	**db**
Namespace	**myproj.db**
Schema Name	**MYSCHEMA_A####**
SAP HANA Database Version	*SAP HANA Cloud*
Bind the database module to a run-time environment service instance?	*Yes*
Create a new HDI service instance?	*Yes*
Service instance name	[Keep the default]
Use the default database instance of the selected Cloud Foundry space?	*Yes*

3. Open the project in a new workspace.

   After a little while, a dialog box *Leave Site*shows up to allow SAP Business Application Studio to leave the current *HC300_A####* project/workspace, and open the new project in a new workspace. invites you to open .

Task 2: Finalize the Project and Deploy the HDB Module

1. Review the content of the *mta.yaml* located at the root of your project.
2. In the project, create the following sub-folders within the *src* folder to organize the design-time content of your HDB module:
   - *data*
   - *models*
   - *synonyms*
   - *test*
3. Deploy the HDB Module *db*.
4. Close all the open tabs in SAP Business Application Studio.

Unit 18
Solution 38

Create a New Project and an HDB Module

Exercise Objectives

After completing this exercise, you will be able to:

- Create a new project (MTA Application)
- Create a HDB module
- Deploy the project

Business Example

You have been working on an SAP HANA project as a modeler in SAP Business Application Studio for some time, and will soon work on a new project. Before that, you want to put into practice key concepts of MTA application, SAP HANA Database module, and namespace configuration, by creating a simple project and reviewing its key properties.

Note:
The scope of this course is restricted to the HDB Module, which means we cover only a part of what MTA projects encompass. To use the best project template for this need, we will use the *SAP HANA Database Project* project template.

Overview of Exercise Tasks

- Task 1: Create the Project and Database Module
- Task 2: Finalize the Project and Deploy the HDB Module

Note:
In this exercise, when values include `A####`, replace the characters with the user number assigned to you (one letter and four digits).

Prerequisites

Your SAP Business Application Studio development environment should have been set up.

Task 1: Create the Project and Database Module

1. If needed, launch SAP Business Application Studio and start your *HCMOD_A####* development space.
2. In the *Explorer* pane, create a new project based on the *SAP HANA Database Project* template, with the following properties:

Field	Value
Project Name	**`MyProject_A####`**

Field	Value
Module Name	**`db`**
Namespace	**`myproj.db`**
Schema Name	**`MYSCHEMA_A####`**
SAP HANA Database Version	*SAP HANA Cloud*
Bind the database module to a run-time environment service instance?	*Yes*
Create a new HDI service instance?	*Yes*
Service instance name	[Keep the default]
Use the default database instance of the selected Cloud Foundry space?	*Yes*

a) Choose *View* → *Command Palette*

b) Enter **`project`** in the search field.

c) Choose *SAP Business Application Studio : New Project from Template*.

As an alternative to the sub-steps above, you can also choose *File* → *New Project from Template*.

d) Select the *SAP HANA Database Project* template and choose *Start*.

e) Enter the project properties according to the table and choose *Next* to navigate to each subsequent screens.

f) Confirm the project creation by choosing *Finish*.

3. Open the project in a new workspace.

After a little while, a dialog box *Leave Site*shows up to allow SAP Business Application Studio to leave the current *HC300_A####* project/workspace, and open the new project in a new workspace. invites you to open .

a) Choose *Leave Site*.

Result

The new project opens in a new workspace.

Task 2: Finalize the Project and Deploy the HDB Module

1. Review the content of the *mta.yaml* located at the root of your project.

a) Click the file *mta.yaml*.

Note:
You can also righ-click the *mta.yaml* file and choose *Open with* → *MTA Editor*, which opens the file with a dedicated user interface.

b) Note that the file contains the project ID and version, the list of modules (only one HDB module), and a resource used by this module, called *hdi_db*, of type **com.sap.xs.hdi-container**.

2. In the project, create the following sub-folders within the *src* folder to organize the design-time content of your HDB module:

   - *data*
   - *models*
   - *synonyms*
   - *test*

   a) Right-click the *db* → *src* folder and choose *New Folder*.

   b) Enter the folder name **`data`** and press *Enter*.

   c) Repeat the previous steps for the other folders.

3. Deploy the HDB Module *db*.

   a) In the *SAP HANA Projects* sub-pane, choose the *Deploy* icon for your *MyProject_A####/db* module.

   b) Check the deployment status in the Console.

4. Close all the open tabs in SAP Business Application Studio.

   a) Right-click any open tab and choose *Close All*.

Unit 18
Exercise 39

Set Up a Project to Access an External Schema

Exercise Objectives

After completing this exercise, you will be able to:

- Reference a service to access data outside of the application container
- Define synonyms for external objects and set up the relevant security

Business Example

In your project, you need to access data that is not managed within your application container. You will set up the application so that it can consume data from external schemas.

Overview of Exercise Tasks

- Task 1: Bind the User Provided Service to your Project
- Task 2: Define Synonyms and Authorization to the External Schema
- Task 3: Check That Your Synonyms are Correctly Defined

Prerequisites

- Your SAP Business Application Studio development environment should have been set up.
- The previous exercise on creating a new project should have been completed.

Task 1: Bind the User Provided Service to your Project

To access data outside of your HDI container, you need a user provided service that will connect to the external database. This user provided service has already been created and is called *EXT_SCHEMA_SERVICE*. It uses a technical user that has the relevant privileges to access your data, and you will need to transfer those rights to the Object Owner and Application User in your project.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. If needed, open your project *MyProject_A####*
3. Bind the user-provided service to your *MyProject_A####* project with the following information:

Field	Value
Connection type	*Existing user-provided service instance*
Service instance name	**`EXT_SCHEMA_SERVICE`**

4. Check the `mta.yaml` file.

Result

Your file should look like the following :

```
ID: MyProject_A####
version: 0.0.1
modules:
- name: db
  type: hdb
  path: db
  requires:
  - name: hdi_db
    properties:
      TARGET_CONTAINER: ~{hdi-container-name}
  - name: cross-container-service-1
    group: SERVICE_REPLACEMENTS
    properties:
      key: ServiceName_1
      service: ~{the-service-name}
resources:
- name: hdi_db
  type: com.sap.xs.hdi-container
  parameters:
    config:
      schema: MYSCHEMA_A####
  properties:
    hdi-container-name: ${service-name}
- name: cross-container-service-1
  type: org.cloudfoundry.existing-service
  parameters:
    service-name: EXT_SCHEMA_SERVICE
  properties:
    the-service-name: ${service-name}
```

Task 2: Define Synonyms and Authorization to the External Schema

You must configure synonyms and authorization in order to access the data from any schema outside of your container.

1. Check your project environment.

   A design decision for your project was that the synonyms should NOT have a namespace prefix. Does the current configuration of your db module follow this rule?

   _____

   _____

   _____

   How can you modify the namespace configuration only for the synonyms folder?

   _____

   _____

   _____

2. Create a new .hdinamespace file in your *synonyms* folder, with the following content:

```
{
    "name":      "",
    "subfolder": "ignore"
}
```

Hint:
To save time, you can copy/paste the *.hdinamespace* file from the *src* folder to the *synonyms* folder, and then open it and adjust its content.

3. Define the authorizations you want to grant to both the object owner and to the application user on objects.

   A design-time file, *ExternalAccessService.hdbgrants*, is located in your exercise folder, *HC300 → New Project Files*, and ready for import into the *synonyms* folder.

4. Deploy the entire *synonyms* folder and check the deployment status.

5. In your *db → src → synonym* folder, create a design-time **`TRAINING.hdbsynonym`** file that will store synonyms for tables located in the *TRAINING* schema.

6. Add a synonym for the table *ORDERS* in the *TRAINING* schema with the following properties:

   Table 7: Synonym Definition

Column	Value
Synonym Name	**`ORDERS`**
Object Name	**`ORDERS`**
Schema Name	**`TRAINING`**

Hint:
When using the *Find Data Sources* utility, you need to add the external schema access service you created earlier in the *Services* list.

7. Deploy the *TRAINING.hdbsynonym* file and check the deployment status.

Task 3: Check That Your Synonyms are Correctly Defined

To check that your settings for the external schemas access are correct, create a view using the *ORDERS* table, which is now referenced by a synonym. You will create a simple calculation view of the type *CUBE*, and output all the columns.

1. In your *models* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVC_ORDERS`**
Label	**`Orders`**
Data Category	*CUBE*
With Star Join	[Not selected]

2. Add the *ORDERS* table to the *Aggregation* node.

Note:
Because the synonym is already defined in your project (inside the *Target Container Service*, you do not need to include the external schema access service in the *Find Data Sources* dialog box. The object you add as a data source is the existing synonym, but you can also see/check the type and name of the actual target object.

In other scenarios, you might want to create the synonym on-the-fly while adding a data source to a calculation view, in which case you need to include the external schema access service to the *Services* list.

3. Add all the columns to the output
4. Deploy the *CVC_ORDERS* calculation view.
   Check the deployment status.
5. Check your Calculation View has been created in your HDI container. You first need to add the HDI container of your deployed modeling project to the SAP HANA Database Explorer.
   The HDI container can be added automatically by launching the SAP HANA Database Explorer from SAP Business Application Studio (*SAP HANA Projects* sub-pane).
6. Modify the display name of your HDI Container connection to make it simpler.
   Use the following pattern: **`<project_name>_A####_HDI_CONTAINER`**, where A#### is your assigned user number
7. Look for your calculation view.
8. Preview the data of the *CVC_ORDERS* calculation view in Data Preview.
9. Close all the open tabs in Business Application Studio.

Unit 18
Solution 39

Set Up a Project to Access an External Schema

Exercise Objectives

After completing this exercise, you will be able to:

- Reference a service to access data outside of the application container
- Define synonyms for external objects and set up the relevant security

Business Example

In your project, you need to access data that is not managed within your application container. You will set up the application so that it can consume data from external schemas.

Overview of Exercise Tasks

- Task 1: Bind the User Provided Service to your Project
- Task 2: Define Synonyms and Authorization to the External Schema
- Task 3: Check That Your Synonyms are Correctly Defined

Prerequisites

- Your SAP Business Application Studio development environment should have been set up.
- The previous exercise on creating a new project should have been completed.

Task 1: Bind the User Provided Service to your Project

To access data outside of your HDI container, you need a user provided service that will connect to the external database. This user provided service has already been created and is called *EXT_SCHEMA_SERVICE*. It uses a technical user that has the relevant privileges to access your data, and you will need to transfer those rights to the Object Owner and Application User in your project.

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. If needed, open your project *MyProject_A####*
   a) You can open your project by using the *File → Open Folder* menu.
3. Bind the user-provided service to your *MyProject_A####* project with the following information:

Field	Value
Connection type	*Existing user-provided service instance*
Service instance name	**`EXT_SCHEMA_SERVICE`**

   a) Go back to the *Explorer* view.

b) In the *SAP HANA PROJECTS* pane, expand the *MyProject_A####/db* module.

c) Hover the *Database Connections* folder and choose the **+** *(Add database connection)* icon on the right.

d) Enter the information provided in the above table and choose *Add*.

4. Check the `mta.yaml` file.

   a) Select the `mta.yaml` file in your project and open it in the Text Editor.

   Result

   Your file should look like the following :

```
ID: MyProject_A####
version: 0.0.1
modules:
- name: db
  type: hdb
  path: db
  requires:
  - name: hdi_db
    properties:
      TARGET_CONTAINER: ~{hdi-container-name}
  - name: cross-container-service-1
    group: SERVICE_REPLACEMENTS
    properties:
      key: ServiceName_1
      service: ~{the-service-name}
resources:
- name: hdi_db
  type: com.sap.xs.hdi-container
  parameters:
    config:
      schema: MYSCHEMA_A####
  properties:
    hdi-container-name: ${service-name}
- name: cross-container-service-1
  type: org.cloudfoundry.existing-service
  parameters:
    service-name: EXT_SCHEMA_SERVICE
  properties:
    the-service-name: ${service-name}
```

Task 2: Define Synonyms and Authorization to the External Schema

You must configure synonyms and authorization in order to access the data from any schema outside of your container.

1. Check your project environment.

   A design decision for your project was that the synonyms should NOT have a namespace prefix. Does the current configuration of your db module follow this rule?

   No. With the current content of the .hdinamespace file in the src folder, the objects defined in the synonyms folder should have the namespace prefix *myproj.db.synonyms*.

How can you modify the namespace configuration only for the synonyms folder?

Add a .hdinamespace file in the synonyms folder that specifies an empty namespace.

2. Create a new .hdinamespace file in your *synonyms* folder, with the following content:

```
{
    "name":      "",
    "subfolder": "ignore"
}
```

Hint:
To save time, you can copy/paste the *.hdinamespace* file from the *src* folder to the *synonyms* folder, and then open it and adjust its content.

   a) Select the *src* → *.hdinamespace* file and press *Ctrl+C (copy)*.

   b) Select the *src* → *synonyms* folder and press **`Ctrl+V`** *(paste)*.

   c) Open the new *.hdinamespace* file and adjust its content.

3. Define the authorizations you want to grant to both the object owner and to the application user on objects.

   A design-time file, *ExternalAccessService.hdbgrants*, is located in your exercise folder, *HC300* → *New Project Files*, and ready for import into the *synonyms* folder.

   a) Right-click the *synonyms* folder and choose *Upload*.

   b) Select the file *HC300* → *New Project Files* → *ExternalAccessService.hdbgrants*, and choose *Open*.

4. Deploy the entire *synonyms* folder and check the deployment status.

   a) Choose the *Deploy* icon for the *synonyms* folder.

   b) In the *Console* pane, check that the deployment completes successfully.

5. In your *db* → *src* → *synonym* folder, create a design-time **`TRAINING.hdbsynonym`** file that will store synonyms for tables located in the *TRAINING* schema.

   a) Right-click the *synonyms* folder and choose *New File*.

   b) Enter the file name **`TRAINING.hdbsynonym`** and press *Enter*.

   Result
   The new *.hdbsynonym* file opens.

6. Add a synonym for the table *ORDERS* in the *TRAINING* schema with the following properties:

   Table 7: Synonym Definition

Column	Value
Synonym Name	**`ORDERS`**
Object Name	**`ORDERS`**

Column	Value
Schema Name	**`TRAINING`**

Hint:
When using the *Find Data Sources* utility, you need to add the external schema access service you created earlier in the *Services* list.

a) In the synonym editor, choose *<Click to Add>*.

b) Enter the synonym name.

c) Choose the ... icon on the right of the Object Name field.

d) In the *Find Data Sources* dialog box, select the service *EXT_SCHEMA_SERVICE*.

e) In the *Search* field, enter **`ord`** and press *Enter*.

f) Select the *ORDERS* table from *TRAINING* schema.

Note:
If you receive an error message *Fail to update request header*, you might need to sign out and log in back to Cloud Foundry. Choose *View* → *Command Palette*, enter **`login`** and choose *CF: Login to Cloud Foundry*. Then sign out and log back in as you did in the first exercises.

g) Choose *Finish*.

7. Deploy the *TRAINING.hdbsynonym* file and check the deployment status.
   a) Choose the *Deploy* icon for the *TRAINING.hdbsynonym* file.
   b) In the *Console* pane, check that the deployment completes successfully.

Task 3: Check That Your Synonyms are Correctly Defined

To check that your settings for the external schemas access are correct, create a view using the *ORDERS* table, which is now referenced by a synonym. You will create a simple calculation view of the type *CUBE*, and output all the columns.

1. In your *models* folder, create a new calculation view with the following properties:

Field	Value
Name	**`CVC_ORDERS`**
Label	**`Orders`**
Data Category	*CUBE*
With Star Join	[Not selected]

   a) In the *Explorer* view, right-click the *db* → *src* → *models* folder and choose *New File*.
   b) Enter **`CVC_ORDERS.hdbcalculationview`** as file name and press *Enter*.

c) Enter the calculation view label as specified in the table.

d) Choose *Create*.

2. Add the *ORDERS* table to the *Aggregation* node.

> Note:
> Because the synonym is already defined in your project (inside the *Target Container Service*, you do not need to include the external schema access service in the *Find Data Sources* dialog box. The object you add as a data source is the existing synonym, but you can also see/check the type and name of the actual target object.
>
> In other scenarios, you might want to create the synonym on-the-fly while adding a data source to a calculation view, in which case you need to include the external schema access service to the *Services* list.

a) Select the new *Aggregation* node and click the **+** sign on the right of the node.

b) In the *Find Data Sources* window, click the search field and enter **ORD**.

c) In the search results, select the *ORDERS* table.

d) Choose *Finish*.

3. Add all the columns to the output

a) Double-click the *Aggregation* node to expand the details panel.

b) In the *Mapping* tab, drag the *ORDERS* Data Source to the *Output Columns* area.

4. Deploy the *CVC_ORDERS* calculation view.
Check the deployment status.

a) Choose the Deploy icon for your *CVC_ORDERS* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

5. Check your Calculation View has been created in your HDI container. You first need to add the HDI container of your deployed modeling project to the SAP HANA Database Explorer.
The HDI container can be added automatically by launching the SAP HANA Database Explorer from SAP Business Application Studio (*SAP HANA Projects* sub-pane).

a) In the *Explorer* view of SAP Business Application Studio, expand the *SAP HANA Projects* sub-pane.

b) On the right of the *MyProject_A####/db* element, choose the *Open HDI Container* icon.

c) If a dialog box about opening an external website shows up, choose *Open*.

> Note:
> You will not do this very often as part of this learning journey, so just choosing *Open* is fine. Otherwise, you can add `hana-cockpit` (wherein SAP HANA Database Explorer is exposed) as a trusted domain.

6. Modify the display name of your HDI Container connection to make it simpler.

Use the following pattern: **<project_name>_A####_HDI_CONTAINER**, where A#### is your assigned user number

a) In the database list, locate your HDI Container connection, which has the pattern *SharedDevKey@MyProject_A####-hdidb-ws-xxxxx*.

b) Right-click the connection name and choose *Properties*.

c) Replace the *Display Name* with **MyProject_A####_HDI_CONTAINER** (A#### is your assigned user number) and choose *OK*.

7. Look for your calculation view.

a) Choose *Column Views*.

b) Check that your *CVC_ORDERS* calculation view is present.

8. Preview the data of the *CVC_ORDERS* calculation view in Data Preview.

a) Back in the workspace, right-click the *CVC_ORDERS* calculation view and choose *Data Preview*.

b) Check that the raw data tab contains data.

9. Close all the open tabs in Business Application Studio.

a) Right-click any open tab and choose *Close All*.

Unit 18
Exercise 40

Use Git for Version Control in Your Local Workspace

Exercise Objectives

After completing this exercise, you will be able to:

- Activate the Git functionality in a local project
- Version your modeling objects with commits
- Work with branches to manage the lifecycle of your project features and versions

Business Example

You are working on a modeling project in SAP Business Application Studio, and would like to understand how the native Git integration within SAP Business Application Studio can help you better support your development cycle. You must first finalize version 2.1.0 of your project, then you will implement changes and new features that will be released in upcoming version 2.2.0.

Note:
Git is a tool that supports collaboration on a project. But a number of Git features are also suitable to support version control requirements even for a developer working alone on a project. This “single developer” use case is a good starting point to understand most of the concepts and benefits of Git.

Overview of Exercise Tasks

- Task 1: Import a Project and Initialize your Project for Git
- Task 2: Modify the Project Content and Commit your Changes
- Task 3: Start the Development of Next Release
- Task 4: Create an Urgent Hotfix (optional task)

Prerequisites

Your SAP Business Application Studio development environment should have been set up.

Task 1: Import a Project and Initialize your Project for Git

Git is not activated by default when you create a project in your workspace or import a project, but it’s very straightforward, because you basically declare that the folder of your SAP Business Application Studio project will be managed as a local Git repository from now on. No content is moved or duplicated. The only change is the creation of a `.git` folder in the project.

1. In the *Explorer* view, import the project archive *MyGitProject.zip* from your course files folder *HC300* → *Git Project Files*. The target folder in the Explorer tree is the `/projects/`.

Give the imported project the name **`MyGitProject_A####`** (A#### is your student number).

> Note:
> The project name contains “Git”, which makes it easier to identify during this course. In real life, you would probably NOT include “Git” in the project name.

2. Enable the Git functionality in your new project *MyGitProject_A####*.
3. Observe the changes in the way the project now displays in the workspace.

   What do you notice?

   _____

   _____

   _____

4. Open the *Source control* view.

   What is the status of all the files of the project? Are they currently staged?

   _____

   _____

   _____

5. Stage all your project content and commit the changes with description **`Import V2.1.0 project (in progress)`**.
6. Display the *Git History* pane and observe the information that is stored for each commit.

Task 2: Modify the Project Content and Commit your Changes

You will now create some objects and modify them, applying your changes step by step, and observe the changes in the history. To save time on object creation and modification, you will use basic text files in which you will add a row of text to simulate file modification.

File1 requires a modification, and you must create a new model *File3*.

1. Modify *File1*.
   Add a row of text: **`Change to finalize Model 1 V2.1.0`**
2. Create a new model **`File3`**.
   Add a row of text: **`Model 3 for V2.1.0`**
3. Stage all the new/modified files and commit the changes with description **`Finalize V2.1.0 project`**. Then check that the commit is listed in the *Git History* pane.

> Note:
> The commit history displays only for the selected file, or all the files of a selected folder. If you want to display the commit history of all your files, you must select, for example, the *models* folder.

4. Observe the commit history of your project.

Which files were affected by the last commit? Can you provide more details?

5. You've just realized that there is a small error in *File3*. Make the correction.
   Add a row of text: **`Small change to Model 3 for V2.1.0`**
6. Amend the previous commit (you think that there is no need for an additional/distinct commit for this small change) and keep the original description.

Hint:
Select the *Amend* checkbox.

7. Close all the open tabs in SAP Business Application Studio.

Task 3: Start the Development of Next Release

The version 2.1.0 of your application has been released, and you want to start developing the new features for the upcoming minor release (V2.2.0).

To clearly separate this new development from the existing version, you will create a dedicated branch.

1. Create a new branch, **`V220`**, in order to isolate the new features development from the *main* branch.
2. Perform the following modifications to the models in your project:
   - Create a new model *File4*, with text content **`New Model 4 for V2.2.0`**.
   - Modify *File2* by adding a row of text: **`Change to Model 2 for V2.2.0`**.
3. Stage all the new/modified files and commit these changes into branch *V220* with description **`Development of V2.2.0 features phase 1`**.
4. Close all the open tabs in the SAP Business Application Studio.
5. In the *Source Control* view, select the *main* branch.

   What do you observe?

6. Open *File2* and switch back to branch *V220*.

   What do you notice?

7. Close the *File2* tab.

Task 4: Create an Urgent Hotfix (optional task)

Your development work for the upcoming version *V220* is not finished yet, but you've just received an e-mail from support requesting an urgent hotfix for a bug identified in *V2.1.0*. As a first step, you will create a new branch **`V211`** to develop the hotfix.

1. From which source branch should you create the new branch *V211*?

   _____

   _____

   _____

2. Create the new branch **`V211`** from *main*.
3. Perform the hotfix, which consists of a modification to *File1*.

   Add a row of text: **`Change to Model 1 for hotfix V2.1.1`**.
4. Update the MTA version to *2.1.1* in the *mta.yaml* file. Then save and close this file.
5. Stage the modified files *File1* and *mta.yaml*, and commit the changes into branch *V211* with description **`Implement Hotfix V2.1.1`**
6. Close all the open tabs in SAP Business Application Studio
7. The hotfix has been tested and released. You will now merge the *V211* branch into the *main* branch.

   Hint:
   You must check out the *main* branch first.
8. Check that the *main* branch now contains the hotfix.
9. Delete the *V211* branch.

   Note:
   Depending on your tracking requirements, you might prefer to keep the *V211* branch as is, because it is a good way to visualize the details of the hotfix, especially after additional changes are made to the *main* branch.

Unit 18
Solution 40

Use Git for Version Control in Your Local Workspace

Exercise Objectives

After completing this exercise, you will be able to:

- Activate the Git functionality in a local project
- Version your modeling objects with commits
- Work with branches to manage the lifecycle of your project features and versions

Business Example

You are working on a modeling project in SAP Business Application Studio, and would like to understand how the native Git integration within SAP Business Application Studio can help you better support your development cycle. You must first finalize version 2.1.0 of your project, then you will implement changes and new features that will be released in upcoming version 2.2.0.

Note:
Git is a tool that supports collaboration on a project. But a number of Git features are also suitable to support version control requirements even for a developer working alone on a project. This “single developer” use case is a good starting point to understand most of the concepts and benefits of Git.

Overview of Exercise Tasks

- Task 1: Import a Project and Initialize your Project for Git
- Task 2: Modify the Project Content and Commit your Changes
- Task 3: Start the Development of Next Release
- Task 4: Create an Urgent Hotfix (optional task)

Prerequisites

Your SAP Business Application Studio development environment should have been set up.

Task 1: Import a Project and Initialize your Project for Git

Git is not activated by default when you create a project in your workspace or import a project, but it’s very straightforward, because you basically declare that the folder of your SAP Business Application Studio project will be managed as a local Git repository from now on. No content is moved or duplicated. The only change is the creation of a `.git` folder in the project.

1. In the *Explorer* view, import the project archive *MyGitProject.zip* from your course files folder *HC300 → Git Project Files*. The target folder in the Explorer tree is the `/projects/`.

Give the imported project the name **`MyGitProject_A####`** (A#### is your student number).

Note:
The project name contains “Git”, which makes it easier to identify during this course. In real life, you would probably NOT include “Git” in the project name.

a) In the *Explorer* view, to open the */projects* folder, choose *File* → *Open Folder*, select `/home/user/projects/` and choose *OK*.

b) Choose *File* → *Import Project*.

c) In the dialog box, choose the `HC300/Git Project Files/MyGitProject.zip` and choose *Open*.

Result
The imported `.zip` archive is added to the `/projects` folder and the project is automatically unzipped.

d) If the project was opened in a new workspace, reopen the `/projects` folder (choose *File* → *Open Folder* and choose the `/projects` folder.

e) To change the imported project name to **`MyGitProject_A####`**, right-click the project name and choose *Rename*.

f) Delete the .zip archive from the *Explorer* tree.

2. Enable the Git functionality in your new project *MyGitProject_A####*.

a) Choose *View* → *Command Palette* and enter **`Git ini`**.

b) Choose *Git: Initialize Local Repository*.

Caution:
Make sure you do NOT select the entire `projects` folder.

c) Click *Choose Folder*.

d) Select the project *MyGitProject_A####*.

e) Choose *OK*.

f) Close the dialog box without choosing any option.

Result
The project folder is initialized and stays open in the *Explorer* view.

3. Observe the changes in the way the project now displays in the workspace.

a) Expand the project structure in the workspace tree.

What do you notice?

The project files and folders are displayed in green. Besides, icons and letters are used to indicate the files and folders status in Git.

4. Open the *Source control* view.
   a) On the activity bar at the left of the screen, choose the *Source Control* icon.
   b) In the *Changes* area, observe the *Status* of the files.

   What is the status of all the files of the project? Are they currently staged?

   These files have a U status icon, which means that they are untracked. Note that when enabling an existing project for Git tracking with *Git: Initialize Local Repository*, the existing files are NOT automatically staged.

5. Stage all your project content and commit the changes with description **`Import V2.1.0 project (in progress)`**.
   a) In the *Source Control* view, below the *Commit* button, hover the *Changes* node and, on the right, choose *Stage All Changes*.
   b) Check that all the listed files are now in the *Staged Changes* node.
   c) In the *Commit Message* field, enter **`Import V2.1.0 project (in progress)`**.
   d) Choose *Commit*.

6. Display the *Git History* pane and observe the information that is stored for each commit.
   a) At the top of the *Source Control* view, choose *Git: View History (git log)*.
   b) Note that the git history can be filtered by author, branch, and so on. It also enable quick reset, branching, and others, in relation to a given commit.
   c) Close the *Git History* tab.

Task 2: Modify the Project Content and Commit your Changes

You will now create some objects and modify them, applying your changes step by step, and observe the changes in the history. To save time on object creation and modification, you will use basic text files in which you will add a row of text to simulate file modification.

File1 requires a modification, and you must create a new model *File3*.

1. Modify *File1*.
   Add a row of text: **`Change to finalize Model 1 V2.1.0`**
   a) Open the *Explorer* view.
   b) In your project *MyGitProject_A####*, expand the folder *dbmodule* → *src* → *models*.
   c) Click *File1*.
   d) Add the text provided in the step.
   e) Choose *Save* if *Auto Save* is off.

2. Create a new model **`File3`**.
   Add a row of text: **`Model 3 for V2.1.0`**

a) Right-click the *models* folder and choose *New* → *File*.

b) Enter the file name **`File3`** and press **`Enter`**.

c) Add the text provided in the step.

d) If needed, choose *Save*.

3. Stage all the new/modified files and commit the changes with description **`Finalize V2.1.0 project`**. Then check that the commit is listed in the *Git History* pane.

> Note:
> The commit history displays only for the selected file, or all the files of a selected folder. If you want to display the commit history of all your files, you must select, for example, the *models* folder.

a) In the *Source Control* view, make sure that both *File1* and *File3* are listed and choose *Stage All*.

> Note:
> If needed, to display the modified files in the *Changes* list, choose *Refresh*.

b) In the *Message* field, add the text provided in the step.

c) Choose *Commit*.

d) At the top of the *Source Control* view, choose *Git: View History (git log)*.

e) Check that your last commit is listed.

f) Observe how the bottom part of the *Git History* tab changes depending on which commit you select in the top part.

If you want to see the history of a specific file, choose the corresponding *History* icon at the right of the file list.

4. Observe the commit history of your project.

a) Check that the entire commit history for all your files is listed.

b) Select the last commit (the top one in the list) and check the status of each of the affected files.

Which files were affected by the last commit? Can you provide more details?

File1 and *File3* were affected by the last commit. The commit details show that *File1* was modified (M) and *File3* was added (A).

5. You've just realized that there is a small error in *File3*. Make the correction.

Add a row of text: **`Small change to Model 3 for V2.1.0`**

a) If *File3* is not open, in the project tree, click *File3*.

b) Add the text provided in the step.

c) If needed, choose *Save*.

6. Amend the previous commit (you think that there is no need for an additional/distinct commit for this small change) and keep the original description.

Hint:
Select the *Amend* checkbox.

   a) In the *Source Control* view, choose *Refresh.*.
   b) At the right of *File3*, select the **+** *(Stage)* checkbox.
   c) In the dropdown list at the right of the *Message* field, choose *Commit (Amend)*.
   d) In the *Commit_EditMsg* that shows up with commit details, keep the previous description.
   e) Close the *Commit_EditMsg* file to let the commit (amend) terminate.*Commit*.

7. Close all the open tabs in SAP Business Application Studio.
   a) Right-click any open tab and choose *Close All*.

Task 3: Start the Development of Next Release

The version 2.1.0 of your application has been released, and you want to start developing the new features for the upcoming minor release (V2.2.0).

To clearly separate this new development from the existing version, you will create a dedicated branch.

1. Create a new branch, **`V220`**, in order to isolate the new features development from the *main* branch.
   a) In the *Source Control* view, click the **...** icon and choose *Branch* → *Create Branch*.
   b) In the *New Branch Name* dialog box, field, enter **`V220`** and press **`Enter`**.

      Result

      The new branch is created and automatically selected (which, in Git terminology, is known as “checked out"). This is visible in the "prompt" in the *Message* field.

2. Perform the following modifications to the models in your project:
   - Create a new model *File4*, with text content **`New Model 4 for V2.2.0`**.
   - Modify *File2* by adding a row of text: **`Change to Model 2 for V2.2.0`**.

   a) In the *Explorer* view, right-click the *models* folder and choose *New File*.
   b) Enter the file name **`File4`** and choose *OK*.
   c) Add the text provided in the step for *File4*.
   d) If needed, click *Save*.
   e) Choose *File2*.
   f) Add the text provided in the step.
   g) Click *Save*.

3. Stage all the new/modified files and commit these changes into branch *V220* with description **`Development of V2.2.0 features phase 1`**.
   a) In the *Source Code* view, choose *Refresh*, then at the right of the *Changes* node, choose *Stage All Changes*.
   b) In the *Commit Description* field, add the text provided in the step.
   c) Choose *Commit*.
4. Close all the open tabs in the SAP Business Application Studio.
   a) Right-click any open tab and choose *Close All*.
5. In the *Source Control* view, select the *main* branch.
   a) At the top of the *Source Control* view, select the **...** icon and choose *Checkout To*.
   b) Notice that the *main* and *V220* branches have a different commit hash (because *V220* is now ahead of *main*).
   c) Choose the *main* branch.
   d) Open the *Explorer* view.

   What do you observe?

   The content of the workspace has changed: *File4* has disappeared because it does not exist in the *main* branch.
6. Open *File2* and switch back to branch *V220*.
   a) In your *models* folder, click *File2*.
   b) In the *Source Control* window, click the **...** icon and choose *Checkout To* and select the branch*V220*.

   What do you notice?

   The presentation of *File2* is modified to materialize existing changes between the content of File2 in each branch. You can use the little red arrow to display the file content as in branch *V220*.
7. Close the *File2* tab.
   a) Click the **x** *(close)* icon next to the tab name.

Task 4: Create an Urgent Hotfix (optional task)

Your development work for the upcoming version *V220* is not finished yet, but you've just received an e-mail from support requesting an urgent hotfix for a bug identified in *V2.1.0*. As a first step, you will create a new branch **`v211`** to develop the hotfix.

1. From which source branch should you create the new branch *V211*?

   Because you do not want to release your work in progress together with the patch/hotfix, you must create the new branch from *main*. Branching from the current branch *V220* would not suit your requirement.
2. Create the new branch **`v211`** from *main*.

a) In the *Source Control* view, choose the **...** icon and choose *Branch* → *Create Branch From*.

> Note:
> Alternatively, because the branch *main* is currently checked out, you could just use the *Branch* → *Create Branch* menu item as you did in a previous task.

b) In the ref to create branch from, choose *main*.

c) In the *New Branch Name* field, enter **`V211`** and press **`Enter`**.

d) Choose *OK*.

3. Perform the hotfix, which consists of a modification to *File1*.
   Add a row of text: **`Change to Model 1 for hotfix V2.1.1`**.
   a) In the *Explorer* view, double-click *File1*.
   b) Add the text provided in the step.
   c) Click *Save*.
4. Update the MTA version to *2.1.1* in the *mta.yaml* file. Then save and close this file.
   a) In the workspace tree of *MyGitProject_A####,* double-click the *mta.yaml* file.
      Result
      The file opens in the MTA Editor or the Text Editor depending on your preferences.
   b) In the *Basic Information* tab, update the *Application Version* to **`2.1.1`**.
   c) Choose *Save*.
   d) Click the **x** *(close)* icon next to the tab name.
5. Stage the modified files *File1* and *mta.yaml*, and commit the changes into branch *V211* with description **`Implement Hotfix V2.1.1`**
   a) In the *Source Control* view, check that the two modified files are listed in the changes and choose *Stage All Changes*.
   b) In the *Message* field, add the text provided in the step.
   c) Choose *Commit*.
6. Close all the open tabs in SAP Business Application Studio
   a) Right-click any open tab and choose *Close All*.
7. The hotfix has been tested and released. You will now merge the *V211* branch into the *main* branch.

> Hint:
> You must check out the *main* branch first.

a) In the *Source Control* view, click the **...** icon and choose *Checkout to*.

b) Choose the *main* branch.

c) Click the **...** icon again and choose *Branch* → *Merge*.

d) In the dialog box, select the *V211* branch.

8. Check that the *main* branch now contains the hotfix.

a) At the top of the *Source Control* view, choose *Git: View History*.

b) In the open tab, check that the branch is *main* and, in the Git "timeline", check that the commit *Implement Hotfix V2.1.1* is listed.

c) Close the *Git History* tab.

9. Delete the *V211* branch.

> Note:
> Depending on your tracking requirements, you might prefer to keep the *V211* branch as is, because it is a good way to visualize the details of the hotfix, especially after additional changes are made to the *main* branch.

a) Make sure the *V211* branch is not checked out. If it is, check out any other branch, for example, *main*.

> Note:
> This is mandatory because you cannot delete a branch that is currently checked out.

b) In the *Source Control* view, click the **...** icon and choose *Branch* → *Delete Branch*.

c) Select the *V211* branch.

Unit 19
Exercise 41

Create and Assign an Analytic Privilege

Exercise Objectives

After completing this exercise, you will be able to:

- Create an analytic privilege
- Secure a view with an analytic privilege

Business Example

You have finished creating a number of calculation views, and you need to secure the data access for these views before they are moved to the QA environment for testing.

Overview of Exercise Tasks

- Import a Calculation View
- Preview the Data with your User
- Create an Analytic Privilege
- Modify a Calculation View to Enable Analytic Privileges Check
- Preview the Calculation View Data

Prerequisites

You should have created a development space into SAP Business Application Studio, imported and deployed the existing *HC300* project (*HC300_A####*), and created the analytic privilege AP_COUNTRY_US.

Task 1: Import a Calculation View

You will first import a Calculation View and check that you can display its data.

1. In SAP Business Application Studio, import the *CVCS_PO2.hdbcalculationview* file from your course files folder *HC300* into your HC300_A#### project, in the view tree folder *src* → *models* → *exercises*.
2. Deploy the *CVCS_PO2* view.
3. Preview the calculation view data.

   It should retrieve data for several countries.

Task 2: Preview the Data with your User A####

From now on, and during the remainder of the exercise, you will preview the data with your A#### user, using the *Data Preview with Other Database User*, so that you can correctly test the behavior of the view while you're defining data access security for this view.

Note:
Indeed, due to the containerization, the standard data preview from the *Explorer* view is executed by a technical user, NOT your connected user *A####*.

An alternative could be to execute a SQL query on top of the Calculation View in the Database Explorer, but this time with the connection to the *HC300_DB (A####) (SSO enabled)*, not to the HDI Container.

1. If needed, open the calculation view *CVCS_PO2*.
2. Execute the data preview from the calculation view editor, with your A#### user.
   You actually choose a connection (HC300_DB) and the associated user (A####) or (SSO enabled) from the connections list of Database Explorer.
3. Leave the data preview ta open. This will allow you to refresh the data preview while setting up your analytic privileges.

Task 3: Create an Analytic Privilege
You want to limit the visibility of data for this calculation view to the purchase orders from customers located in the US.

1. Create a new analytic privilege in the `src/models/exercises` folder, with the following properties:

   Table 8: Analytic Privilege Properties

Field	Value
Name	**`AP_COUNTRY_US`**
Label	**`US Partners`**

2. Reference the view *CVCS_PO2* so that it is secured by this analytic privilege.
3. Define the analytic privilege restriction.
   Add a restriction to limit the data for business partners located in the US.
4. Save and deploy the new analytic privilege.
   Check the build status.
5. Why did the build fail?

_____

_____

_____

Task 4: Modify a Calculation View to Enable Analytic Privileges Check
You must modify the calculation view secured by the *AP_COUNTRY_US* privilege before you include this view in the list of models this analytic privilege secures.

1. Modify the calculation view *CVCS_PO2* so that it triggers an analytic privilege check when it is queried.

Hint:
This is done in the *View Properties* tab of Semantics.

2. Save and deploy the *CVCS_PO2* calculation view.
   Check the build status.
3. Deploy again the *AP_COUNTRY_US* analytic privilege.

Task 5: Preview the Calculation View Data

You will now refresh the data preview tab to see if your analytic privilege definition has been applied.

1. Refresh the data preview and analyze the results.
2. Can you explain why the view does not show any data at all?

   _____

   _____

   _____
3. Get ready for the next exercise where you create and assign a new role, including the new analytic privilege, to your user, so that he can access the authorized data of the calculation view *CVCS_PO2*.

Unit 19
Solution 41

Create and Assign an Analytic Privilege

Exercise Objectives

After completing this exercise, you will be able to:

- Create an analytic privilege
- Secure a view with an analytic privilege

Business Example

You have finished creating a number of calculation views, and you need to secure the data access for these views before they are moved to the QA environment for testing.

Overview of Exercise Tasks

- Import a Calculation View
- Preview the Data with your User
- Create an Analytic Privilege
- Modify a Calculation View to Enable Analytic Privileges Check
- Preview the Calculation View Data

Prerequisites

You should have created a development space into SAP Business Application Studio, imported and deployed the existing *HC300* project (*HC300_A####*), and created the analytic privilege AP_COUNTRY_US.

Task 1: Import a Calculation View

You will first import a Calculation View and check that you can display its data.

1. In SAP Business Application Studio, import the *CVCS_PO2.hdbcalculationview* file from your course files folder *HC300* into your HC300_A#### project, in the view tree folder *src* → *models* → *exercises*.
   a) Expand your *HC300_A####* project to open the *db* → *src* → *models* folder.
   b) In the *Explorer* view, right-click your *exercises* folder and choose *Upload*.
   c) In the dialog box, select the *HC300* → *Calculation Views Templates* → *CVCS_PO2.hdbcalculationview* file and choose *Open*.
   Result
   The new Calculation View *CVCS_PO2* is created in your project.
2. Deploy the *CVCS_PO2* view.
   a) In the *SAP Hana Projects* sub-pane, expand the folder*src* → *models* → *exercises*.
   b) Hover your mouse on the new (imported) view and choose *Deploy*.

c) In the terminal, check that the build was successful.

3. Preview the calculation view data.

   It should retrieve data for several countries.

   a) In the *Explorer* view tree, right-click the new (imported) view and choose *Data Preview*.

   b) Check that the view retrieves data.

Task 2: Preview the Data with your User A####

From now on, and during the remainder of the exercise, you will preview the data with your A#### user, using the *Data Preview with Other Database User*, so that you can correctly test the behavior of the view while you're defining data access security for this view.

Note:
Indeed, due to the containerization, the standard data preview from the *Explorer* view is executed by a technical user, NOT your connected user *A####*.

An alternative could be to execute a SQL query on top of the Calculation View in the Database Explorer, but this time with the connection to the *HC300_DB (A####) (SSO enabled)*, not to the HDI Container.

1. If needed, open the calculation view *CVCS_PO2*.

   a) In the *Explorer* view, double-click the view *CVCS_PO2.hdbcalculationview*.

2. Execute the data preview from the calculation view editor, with your A#### user.

   You actually choose a connection (HC300_DB) and the associated user (A####) or (SSO enabled) from the connections list of Database Explorer.

   a) Right-click the *Star Join* node and choose *Data Preview with Other Database User*.

   b) In the *Select Database With User*, choose the connection HC300_DB (A####) (SSO enabled).

   Result

   The calculation view opens in data preview mode in a new tab.

   c) In the *Analysis* tab, assign the column *COUNTRY_NAME* to the Label axis and the column *GROSS_AMOUNT* to the Value axis.

3. Leave the data preview ta open. This will allow you to refresh the data preview while setting up your analytic privileges.

Task 3: Create an Analytic Privilege

You want to limit the visibility of data for this calculation view to the purchase orders from customers located in the US.

1. Create a new analytic privilege in the `src/models/exercises` folder, with the following properties:

   Table 8: Analytic Privilege Properties

Field	Value
Name	**`AP_COUNTRY_US`**

Field	Value
Label	**`US Partners`**

a) Right-click your `exercises` folder and choose *New File*.

b) Enter the name **`AP_COUNTRY_US.hdbanalyticprivilege`**.

Result
The analytic privilege opens in a new tab.

c) To rename the description, top-right of the analytic privilege editor, choose *Rename*.

d) Enter the *Label* as per the table and choose *OK*.

2. Reference the view *CVCS_PO2* so that it is secured by this analytic privilege.

a) In the *Secured Models* area, choose the **+** *(add)* icon.

b) In the *Find Data Sources* field, enter **`PO2`**.

c) Select the *HC::CVCS_PO2* view and choose *Finish*.

3. Define the analytic privilege restriction.

Add a restriction to limit the data for business partners located in the US.

a) In the *Associated Attribute Restrictions* area, choose **+** *(add)*.

b) In the *Attribute* window, choose the *COUNTRY* attribute from the shared dimension view *HC::CVD_BP2* and choose *OK*.

c) In the *Restriction Type* column, click the **+** icon.

d) In the *Operator* field, choose **=**.

e) In the *Value*, select *US* and choose *OK*.

4. Save and deploy the new analytic privilege.

Check the build status.

a) Choose *Save* if needed.

b) In the *SAP HANA Projects* window, click the *AP_COUNTRY_US* analytic privilege and choose *Deploy*.

Result
The build fails.

5. Why did the build fail?

The build was not successful because the analytic privilege *AP_COUNTRY_US* is defined to secure the view *CVCS_PO2*, but the view is not currently set to secure its data with analytic privileges.

Task 4: Modify a Calculation View to Enable Analytic Privileges Check

You must modify the calculation view secured by the *AP_COUNTRY_US* privilege before you include this view in the list of models this analytic privilege secures.

1. Modify the calculation view *CVCS_PO2* so that it triggers an analytic privilege check when it is queried.

Hint:
This is done in the *View Properties* tab of Semantics.

a) If needed, open the *exercises* → *CVCS_PO2* calculation view.

b) In *Semantics*, display the *View Properties* tab.

c) In the *Apply Privileges* drop-down list, choose *SQL Analytic Privileges*.

2. Save and deploy the *CVCS_PO2* calculation view.
   Check the build status.
   a) Choose *Save* if needed.
   b) In the *SAP HANA Projects* window, click the *CVCS_PO2* calculation view and choose *Deploy*.
3. Deploy again the *AP_COUNTRY_US* analytic privilege.
   a) In the *SAP HANA Projects* window, click the *AP_COUNTRY_US* analytic privilege and choose *Deploy.*.
   b) Check the build status.

   Result
   This time, the build is successful. As a conclusion, all the views that an analytic privilege secures must have the analytic privilege check enabled, and they must be built (runtime object must exist) before you can actually build this analytic privilege.

Task 5: Preview the Calculation View Data
You will now refresh the data preview tab to see if your analytic privilege definition has been applied.

1. Refresh the data preview and analyze the results.
   a) In the former data preview tab, to refresh the data, choose *Execute*.

   What do you observe?

   The data is not available any longer. The user is not authorized.

2. Can you explain why the view does not show any data at all?

   The view is now set to check analytic privileges, but the (only) analytic privilege that secures its data is not assigned to any user.

3. Get ready for the next exercise where you create and assign a new role, including the new analytic privilege, to your user, so that he can access the authorized data of the calculation view *CVCS_PO2*.

Unit 19
Exercise 42

Create a Design Time Role

Exercise Objectives

After completing this exercise, you will be able to:

- Define a role based on an analytic privilege
- Assign a role to a user

Business Example

You have finished securing some Calculation Views with Analytic Privileges. You need to create a design time role to assign those privileges to the relevant users.

Overview of Exercise Tasks

- Create a Design-Time Role in your Project
- Assign the New Role to your User
- Refresh the Data Preview
- Explore Additional Analytic Privilege Configurations

Prerequisites

- Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.
- You should have created and deployed the *AP_COUNTRY_US* analytic privilege.

Task 1: Create a Design-Time Role in your Project

To grant the new privilege to external users, you will create a design-time role *DATA_VIEWER* that references this analytic privilege.

1. In the *db* → *src* → *exercises* folder, create a new role *DATA_VIEWER.hdbrole*.

   This role will reference only privileges located in your HDI container, so you do not need a Role configuration file.

   Hint:
   In this scenario,we recommend you use the command palette to create SAP HANA Database artifact.

2. Add the Analytic Privilege *HC::AP_COUNTRY_US* to the role definition.
3. If the wizard created automatically a *DATA_VIEWER.hdbroleconfig* file, which is not needed in our scenario, delete this file from the workspace and remove the reference to it from the *DATA_VIEWER.hdbrole* artifact.
4. Save, and then deploy the *DATA_VIEWER* role.

Task 2: Assign the New Role to your User

Now you are ready to grant the runtime role *DATA_VIEWER*, containing your analytic privilege, to your *A####* user.

1. In the *SAP HANA Database Explorer* view, open an SQL console connected to the *HC300_DB (A####) (SSO enabled)* (not your container).

   Caution:
   Make sure you do NOT open the Console for your container. If you are connected to your container, the SQL query will NOT be authorized.

2. In the SQL console, import the following file containing prepared SQL statements: *HC300 → Prepared Code → SQL to Assign Data Viewer Role.sql*. Then replace A#### with your assigned student number.

3. Execute the first statement to assign the *DATA_VIEWER* role to your user.

```
CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','HC300_A####_HDI_DB_1','HC::DAT
A_VIEWER', ?);
```

4. Leave the SQL console open, as you will use the other SQL statements from the imported file later on.

Task 3: Refresh the Data Preview

You can now test whether the data is available (and restricted) in the data preview of *CVCS_PO2* with user A####.

1. Display or refresh the data preview of the calculation view *CVCS_PO2*.

   Result

   You can now see some of the *CVCS_PO2* data, restricted to the Country US.

2. Keep the SQL Console tab open and close all other tabs.

Task 4: Explore Additional Analytic Privilege Configurations

You will now have a look at another calculation view *CVCS_PO3_AP* which is secured by different analytic privileges, and observe how these analytic privileges impact the returned data set.

1. Open the following Analytic Privileges located in folder *db → src → resources* and check their definition:

   - AP_COUNTRY_DE
   - AP_COUNTRY_DE_PROD_NOTEBOOK
   - AP_PROD_NOTEBOOKS

   Which Calculation View do these Analytic Privileges secure?

   _____

   _____

   _____

2. Open the following roles located in the folder *db* → *src* → *resources* and check their definition:

   - DATA_VIEWER2
   - DATA_VIEWER3

   Based on your observation, what is the key difference between the two roles?

3. Back to the SQL console, execute the statements in section #2. They grant the *DATA_VIEWER2* role to your user and query the calculation view *CVCS_PO3_AP*.

```
CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','HC300_A####_HDI_DB_1','HC::DAT
A_VIEWER2', ?);
SELECT DISTINCT COUNTRY, CATEGORY FROM
"HC300_A####_HDI_DB_1"."HC::CVCS_PO3_AP";
```

> Note:
> The SELECT statement is designed to retrieve only the authorized country/ product category pairs without showing the entire result set. Here, this is enough to check the behavior of the analytic privileges.

   What do you observe?

4. Execute the statements in section #4.4. This will revoke the previous *DATA_VIEWER2* role and grant the other one *DATA_VIEWER3* to your user, and query the same calculation view again.

```
CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('REVOKE','HC300_A####_HDI_DB_1','HC::DA
TA_VIEWER2', ?);
CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','HC300_A####_HDI_DB_1','HC::DAT
A_VIEWER3', ?);
SELECT DISTINCT COUNTRY, CATEGORY FROM
"HC300_A####_HDI_DB_1"."HC::CVCS_PO3_AP";
```

   What do you observe this time? How can you explain it?

5. Close all open tabs in SAP Business Application Studio

Unit 19
Solution 42

Create a Design Time Role

Exercise Objectives

After completing this exercise, you will be able to:

- Define a role based on an analytic privilege
- Assign a role to a user

Business Example

You have finished securing some Calculation Views with Analytic Privileges. You need to create a design time role to assign those privileges to the relevant users.

Overview of Exercise Tasks

- Create a Design-Time Role in your Project
- Assign the New Role to your User
- Refresh the Data Preview
- Explore Additional Analytic Privilege Configurations

Prerequisites

- Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.
- You should have created and deployed the *AP_COUNTRY_US* analytic privilege.

Task 1: Create a Design-Time Role in your Project

To grant the new privilege to external users, you will create a design-time role *DATA_VIEWER* that references this analytic privilege.

1. In the *db* → *src* → *exercises* folder, create a new role *DATA_VIEWER.hdbrole*.

   This role will reference only privileges located in your HDI container, so you do not need a Role configuration file.

   Hint:
   In this scenario,we recommend you use the command palette to create SAP HANA Database artifact.

   a) Choose *View* → *Command Palette* and enter **`artifact`**.

   b) Choose *Create SAP HANA Database Artifact*.

   c) In the field *Specify where you want to create the new artifact*, select your folder `projects/HC300_A####/db/src/models/exercises`.

   d) In the *Choose the artifact type* field, choose *Role*.

e) In the *Specify the artifact name* field, enter **`DATA_VIEWER`**.

f) Choose *Create*.

Result

The new role *DATA_VIEWER* is created and it opens in the *Role editor* window.

2. Add the Analytic Privilege *HC::AP_COUNTRY_US* to the role definition.

a) Display the *Analytic Privileges* tab.

b) In the empty line, choose the button to the right of the *Privilege Name* dropdown list

c) In the *Search* field, enter **`AP_COUNTRY`** and choose your *HC::AP_COUNTRY_US* Analytic Privilege.

d) Remove the name of your container schema from the corresponding *Schema Reference* column.

3. If the wizard created automatically a *DATA_VIEWER.hdbroleconfig* file, which is not needed in our scenario, delete this file from the workspace and remove the reference to it from the *DATA_VIEWER.hdbrole* artifact.

a) In the role editor, display the *Schema References* tab.

b) Next to the *Role config file* field, choose the X *(Remove Config File)* icon.

c) In the *Explorer* view, right-click the unnecessary *DATA_VIEWER.hdbroleconfig* file and choose *Delete Permanently*. In the confirmation dialog box, choose *Delete*.

4. Save, and then deploy the *DATA_VIEWER* role.

a) Choose *Save* if needed.

b) In the *SAP HANA Projects* sub-pane, select the `HC300_A####/db/src/models/exercises/DATA_VIEWER.hdbrole` object and, on the right, click the *Deploy* icon.

c) In the *Console*, check that the build was successful.

Task 2: Assign the New Role to your User

Now you are ready to grant the runtime role *DATA_VIEWER*, containing your analytic privilege, to your *A####* user.

1. In the *SAP HANA Database Explorer* view, open an SQL console connected to the *HC300_DB (A####) (SSO enabled)* (not your container).

Caution:
Make sure you do NOT open the Console for your container. If you are connected to your container, the SQL query will NOT be authorized.

a) In the *Activity* bar, choose *SAP HANA Database Explorer*.

b) In the *Database list* view, select your *HC300_DB (A####) (SSO enabled)* entry and, on the right, choose *Open SAP HANA SQL Console*.

2. In the SQL console, import the following file containing prepared SQL statements:
HC300 → Prepared Code → SQL to Assign Data Viewer Role.sql. Then replace A#### with your assigned student number.

a) On your File Explorer, open the file specified above.

b) To select all the content, enter **Ctrl+A**, and to copy it to the clipboard, enter **Ctrl+V**.

c) Back to the SQL Console tab in SAP Business Application Studio, to paste the SQL statements, enter **Ctrl+V**.

d) To replace A#### with you student number everywhere in the SQL statements, press **Ctrl+H** and use the *Search and Replace* feature.

3. Execute the first statement to assign the *DATA_VIEWER* role to your user.

```
CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','HC300_A####_HDI_DB_1','HC::DAT
A_VIEWER', ?);
```

a) Highlight the SQL statement in section #1 and choose *Run (F8)*.

Alternatively, you can put your cursor anywhere within the statement and press **F9**.

b) Check that the grant statement was successful.

4. Leave the SQL console open, as you will use the other SQL statements from the imported file later on.

Task 3: Refresh the Data Preview

You can now test whether the data is available (and restricted) in the data preview of *CVCS_PO2* with user A####.

1. Display or refresh the data preview of the calculation view *CVCS_PO2*.

a) If needed, reopen the calculation view *CVCS_PO2* and preview its data with user A####, as you did in a previous step.

b) If the data preview was already open from previous steps, from the data preview tab, choose the *Execute* icon.

c) Check the displayed data in the pivot table.

Result

You can now see some of the *CVCS_PO2* data, restricted to the Country US.

2. Keep the SQL Console tab open and close all other tabs.

a) Right-click the SQL Console tab and choose *Close Others*.

Task 4: Explore Additional Analytic Privilege Configurations

You will now have a look at another calculation view *CVCS_PO3_AP* which is secured by different analytic privileges, and observe how these analytic privileges impact the returned data set.

1. Open the following Analytic Privileges located in folder *db* → *src* → *resources* and check their definition:

- AP_COUNTRY_DE
- AP_COUNTRY_DE_PROD_NOTEBOOK
- AP_PROD_NOTEBOOKS

a) In the workspace, locate each file and double-click it.

b) Check the *Secured Models* and *Associated Attribute Restrictions* panes.

Which Calculation View do these Analytic Privileges secure?

The three Analytic Privileges secure the Calculation View *HC::CVCS_PO3_AP*.

2. Open the following roles located in the folder *db* → *src* → *resources* and check their definition:

   - DATA_VIEWER2
   - DATA_VIEWER3

   a) In the workspace, locate each file and double-click it.

   b) Check the assigned Analytic Privileges in the corresponding tab.

   Based on your observation, what is the key difference between the two roles?

   One role *(DATA_VIEWER2)* includes a single analytic privilege that combines a restriction on country and product category, while the other *(DATA_VIEWER3)* includes two analytic privilege, one with a country restriction, the other with a product category restriction.

3. Back to the SQL console, execute the statements in section #2. They grant the *DATA_VIEWER2* role to your user and query the calculation view *CVCS_PO3_AP*.

```
CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','HC300_A####_HDI_DB_1','HC::DAT
A_VIEWER2', ?);
SELECT DISTINCT COUNTRY, CATEGORY FROM
"HC300_A####_HDI_DB_1"."HC::CVCS_PO3_AP";
```

Note:
The SELECT statement is designed to retrieve only the authorized country/ product category pairs without showing the entire result set. Here, this is enough to check the behavior of the analytic privileges.

   a) Highlight the statements in section #4.3 and choose *Run (F8)*.

   What do you observe?

   The data set returned by the Calculation View is restricted to rows where Country is 'DE' AND Product Category is 'Notebook'

4. Execute the statements in section #4.4. This will revoke the previous *DATA_VIEWER2* role and grant the other one *DATA_VIEWER3* to your user, and query the same calculation view again.

```
CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('REVOKE','HC300_A####_HDI_DB_1','HC::DA
TA_VIEWER2', ?);
CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','HC300_A####_HDI_DB_1','HC::DAT
A_VIEWER3', ?);
SELECT DISTINCT COUNTRY, CATEGORY FROM
"HC300_A####_HDI_DB_1"."HC::CVCS_PO3_AP";
```

   a) Highlight the statements in section #4.4 and choose *Run (F8)*.

What do you observe this time? How can you explain it?

This time, the data set returned is much broader, as it includes all data where country is 'DE' (regardless of the product category) and all data where product category is 'Notebook' (regardless of the country). This is because each of the two analytic privileges assigned to the view define a data set, and then the two data sets are added up (without duplicates). On the contrary, in the previous step, the (unique) Analytic Privilege in role *DATA_VIEWER2* combines restrictions on two different dimensions, so it returns only data that match the restrictions on both country and product category.

5. Close all open tabs in SAP Business Application Studio
   a) Right-click any open tab and choose *Close All*.

Unit 19
Exercise 43

Define Column Masking in a Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Define column masking in a calculation view
- Observe how masked data displays
- Define a role authorizing to display data
- Assign this role to a user to display unmasked data

Business Example

You are working as a modeler on a project where sensitive data is involved. In order to protect sensitive data, namely, the bank account number of employees, you have been asked to create a dimension calculation view that masks this information to unauthorized end users. You will also need to define a specific role granting access to this sensitive data for some users.

Overview of Exercise Tasks

- Task 1: Import a Calculation View and Preview its Data
- Task 2: Implement Column Masking
- Task 3: Create a Role to Authorize Access to Masked Columns

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import a Calculation View and Preview its Data

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Import the calculation view *CVD_EMPLOYEES.hdbcalculationview*.
   It is located in the folder *HC300 → Calculation Views Templates*.
3. Deploy the imported calculation view.
   Check the deployment status.
4. Preview the data.

How is the Bank Account information displayed?

What is the length of the *BankAccountNumber* column? Is it a numeric or string data type?

Task 2: Implement Column Masking

You will now modify the design of the calculation view so that the bank account number is partially masked. You would like to replace every figure with an X, except the last 5 figures that can be displayed as is. For the sake of clarity, you will also rename the view to show that it uses data masking.

1. Rename the calculation view by adding **`_WITH_MASK`** to its name.

Note:
Make sure you also rename the runtime object name by choosing *Yes* in the *Confirmation* dialog box.

2. Define the mask expression as follows:

```
'XXXXXXXXXX'||RIGHT("BankAccountNumber", 5)
```

3. Deploy the calculation view.
   Check the deployment status.
4. Preview the data

   How is the data displayed now?

Task 3: Create a Role to Authorize Access to Masked Columns.

You want to know how to grant access to the actual data (“unmasked”) for some users. To test the overall process, you will use the classic database connection *HC300_DB (A####) (SSO enabled)*, which is defined with your user *A####*. You will create a role for this, and then assign it to your *A####* user via a dedicated procedure.

1. Switch to the *SAP HANA Database Explorer* perspective and open an SQL console connected to your *HC300_DB (A####) (SSO enabled)* connection.

Caution:
Make sure you do NOT open a console connected to your HDI container. If you are connected to your HDI container, the SQL query will NOT be authorized.

2. In the SQL console, paste the code from the following file containing prepared SQL statements:

   HC300 → Prepared Code → SQL for Data Masking.txt. Then replace A#### with your assigned user number.

```
SELECT * FROM "HC300_A####_HDI_DB_1"."HC::CVD_EMPLOYEES_WITH_MASK";

CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','HC300_A####_HDI_DB_1','HC::UNM
ASK_ALL', ?);
```

3. Execute query #3.3 to display the calculation view data.

```
SELECT * FROM "HC300_A####_HDI_DB_1"."HC::CVD_EMPLOYEES_WITH_MASK";
```

   Note that, compared with the previous data preview which was executed by the technical user of the container, we are now executing a query with user *A####*.

   What do you notice?

   _____

   _____

   _____

4. Back to the *EXPLORER* tool, in your *exercises* folder, upload the role design-time file *UNMASK_ALL.hdbrole* available in your course files folder *HC300*.

5. Review the role definition.

   Which type of privilege does this role grant? What does it mean?

   _____

   _____

   _____

   What specific authorization is granted?

   _____

   _____

   _____

6. Deploy the *HC::UNMASK_ALL* role.

   Check the deployment status.

7. Grant the *HC::UNMASK_ALL* role to your *TRAIN_A####* user.

   This can be done by executing query #3.7 inside your SQL console.

```
CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','HC300_A####_HDI_DB_1','HC::UNM
ASK_ALL', ?);
```

8. Execute again query #3.3 to display the calculation view data.

```
SELECT * FROM "HC300_A####_HDI_DB_1"."HC::CVD_EMPLOYEES_WITH_MASK";
```

What do you notice?

9. Close all the open tabs in Business Application Studio.

Unit 19
Solution 43

Define Column Masking in a Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Define column masking in a calculation view
- Observe how masked data displays
- Define a role authorizing to display data
- Assign this role to a user to display unmasked data

Business Example

You are working as a modeler on a project where sensitive data is involved. In order to protect sensitive data, namely, the bank account number of employees, you have been asked to create a dimension calculation view that masks this information to unauthorized end users. You will also need to define a specific role granting access to this sensitive data for some users.

Overview of Exercise Tasks

- Task 1: Import a Calculation View and Preview its Data
- Task 2: Implement Column Masking
- Task 3: Create a Role to Authorize Access to Masked Columns

Prerequisites

Your SAP Business Application Studio development environment should have been set up and the HC300 project imported and deployed.

Task 1: Import a Calculation View and Preview its Data

1. If needed, launch SAP Business Application Studio and start your HCMOD development space.
2. Import the calculation view *CVD_EMPLOYEES.hdbcalculationview*.
   It is located in the folder *HC300 → Calculation Views Templates*.
   a) Right-click the *exercises* folder and choose *Upload*.
   b) Select the file *HC300 → Calculation Views Templates → CVD_EMPLOYEES.hdbcalculationview*, and choose *Open*.
3. Deploy the imported calculation view.
   Check the deployment status.
   a) Choose the *Deploy* icon for your *CVD_EMPLOYEES* calculation view.
   b) In the *Console* pane, check that the deployment completes successfully.
4. Preview the data.

a) Right-click the *CVD_EMPLOYEES* view and choose *Data Preview*.

b) Select the *Raw Data* tab.

How is the Bank Account information displayed?

The bank account number is displayed in clear.

What is the length of the *BankAccountNumber* column? Is it a numeric or string data type?

The *BankAccountNumber* is made up of 15 digits. The column is of a string data type.

Task 2: Implement Column Masking

You will now modify the design of the calculation view so that the bank account number is partially masked. You would like to replace every figure with an X, except the last 5 figures that can be displayed as is. For the sake of clarity, you will also rename the view to show that it uses data masking.

1. Rename the calculation view by adding **`_WITH_MASK`** to its name.

Note:
Make sure you also rename the runtime object name by choosing *Yes* in the *Confirmation* dialog box.

a) In the *exercises* folder, right-click the *CVD_EMPLOYEES* calculation view and choose *Rename*.

b) Adjust the new name to **`CVD_EMPLOYEES_WITH_MASK.hdbcalculationview`** and press *Enter*.

c) In the *Confirmation* dialog box, choose *Yes*.

2. Define the mask expression as follows:

```
'XXXXXXXXXX'||RIGHT("BankAccountNumber", 5)
```

a) In the *Explorer* view, select the *CVD_EMPLOYEES_WITH_MASK* calculation view.

b) Double-click the *Semantics* node.

c) Click the *Data Masking* property of the column *BankAccountNumber*.

Note:
Only the left part of the *Data Masking* property cell is sensitive. Make sure you click there.

d) In the *Data Masking Expression* window, enter the SQL expression above.

e) Choose *OK*.

f) Make sure that the Data Masking property now shows up.

3. Deploy the calculation view.

Check the deployment status.

a) Choose the *Deploy* icon for your *CVD_EMPLOYEES_WITH_MASK* calculation view.

b) In the *Console* pane, check that the deployment completes successfully.

4. Preview the data

a) Right-click the *CVD_EMPLOYEES_WITH_MASK* view and choose *Data Preview*.

b) Select the *Raw Data* tab.

How is the data displayed now?

The bank account number is now partially masked. For example, for employee 1001, the bank account number *741215654587456* is displayed as *XXXXXXXXXX87456*.

Task 3: Create a Role to Authorize Access to Masked Columns.

You want to know how to grant access to the actual data ("unmasked") for some users. To test the overall process, you will use the classic database connection *HC300_DB (A####) (SSO enabled)*, which is defined with your user *A####*. You will create a role for this, and then assign it to your *A####* user via a dedicated procedure.

1. Switch to the *SAP HANA Database Explorer* perspective and open an SQL console connected to your *HC300_DB (A####) (SSO enabled)* connection.

> Caution:
> Make sure you do NOT open a console connected to your HDI container. If you are connected to your HDI container, the SQL query will NOT be authorized.

a) Select the *SAP HANA Database Explorer* tool.

b) Select the connection *HC300_DB (A####) (SSO enabled)* and choose *Open SAP HANA SQL Console*.

2. In the SQL console, paste the code from the following file containing prepared SQL statements:
HC300 → Prepared Code → SQL for Data Masking.txt. Then replace A#### with your assigned user number.

```
SELECT * FROM "HC300_A####_HDI_DB_1"."HC::CVD_EMPLOYEES_WITH_MASK";

CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','HC300_A####_HDI_DB_1','HC::UNM
ASK_ALL', ?);
```

a) Double-click the file *HC300 → Prepared Code → SQL for Data Masking.txt*.

b) Select all the content of the file (press **Ctrl+A**) and copy it to the clipboard (press **Ctrl+C**).

c) To paste the content in the SQL Console, press **Ctrl+V** or right-click anywhere inside the SQL Console and choose *Paste*.

d) To replace A#### with you group number in the SQL statements, press **Ctrl+H**.

3. Execute query #3.3 to display the calculation view data.

```
SELECT * FROM "HC300_A####_HDI_DB_1"."HC::CVD_EMPLOYEES_WITH_MASK";
```

Note that, compared with the previous data preview which was executed by the technical user of the container, we are now executing a query with user *A####*.

a) Highlight the specified statement and choose *Run* (F8).

Alternatively, you can place the cursor anywhere within the statement and choose *Run Statement (F9)*.

b) Review the output in the *Result* tab.

What do you notice?

User *A####* cannot see the actual data for bank accounts. The data mask is applied.

4. Back to the *EXPLORER* tool, in your *exercises* folder, upload the role design-time file *UNMASK_ALL.hdbrole* available in your course files folder *HC300*.

a) To switch perspective, choose the *EXPLORER* tool.

b) Right-click the *exercises* folder and choose *Upload*.

c) Select the file *HC300* → *UNMASK_ALL.hdbrole* and choose *Open*.

5. Review the role definition.

a) Review the different tabs in the role definition to find which privileges are granted to (that is, included in) the role.

Which type of privilege does this role grant? What does it mean?

The imported role grants a schema privilege. This means that the privilege will be granted for the entire schema, not on specific objects.

What specific authorization is granted?

The granted authorization is *UNMASKED*.

6. Deploy the *HC::UNMASK_ALL* role.

Check the deployment status.

a) Choose the *Deploy* icon for your *UNMASK_ALL* role.

b) In the *Console* pane, check that the deployment completes successfully.

7. Grant the *HC::UNMASK_ALL* role to your *TRAIN_A####* user.

This can be done by executing query #3.7 inside your SQL console.

```
CALL
TRAIN_ADMIN.P_HC300_MANAGE_ROLE('GRANT','HC300_A####_HDI_DB_1','HC::UNM
ASK_ALL', ?);
```

a) Select the tab *SQL Console ...*

b) Make sure A#### has been replaced with your assigned user number.

c) Highlight the specified statement and choose *Run* (F8).

d) Review the output in the *Message* tab.

8. Execute again query #3.3 to display the calculation view data.

```
SELECT * FROM "HC300_A####_HDI_DB_1"."HC::CVD_EMPLOYEES_WITH_MASK";
```

a) Highlight the specified statement and choose *Run* (F8).

b) Review the output in the *Result* tab.

What do you notice?

This time, user *A####* is allowed to unmask the masked columns and see the actual data for bank accounts.

9. Close all the open tabs in Business Application Studio.

a) Right-click any open tab and choose *Close All*.

AIG02

Solving your business problems using prompts and LLMs in generative AI hub

SYSTEM SETUP GUIDE
PRACTICAL EXERCISE

Collection: 11

SAP Copyrights, Trademarks and Disclaimers

Contents

System Setup Guide

1. Introduction

Our practice systems are tailored to meet the specific needs of each course. As a result, each practice system is unique and contains specific technical details. Before you begin working on the exercises for your course, take the time to familiarize yourself with the information provided in this Setup Guide. Follow the instructions outlined in the exercise guide carefully and complete the exercises. The Setup Guide also provides useful information throughout the use of your practice system.

2. Latest Update of This Document (“Change History”)

Latest Update: 08.07.2025

3. System Landscape And Logon Information

The Practice system for SAP Generative AI Hub consist of SAP Business Technology Platform Subaccount fully configured and shared for all learners with the following:

SAP AI Launchpad - Subscription - standard plan

SAP AI Core - Service Instance - extended plan

SAP AI Launchpad AI API connection is configured to SAP AI Core instance with the following setup:

- Workspace: shared-practice-ai-core
- Resource Group: Learning-AIG

The generative AI hub provided gives you access to a set of large language models (LLMs) from different providers. However, this landscape is configured to allow only selected models for the purpose of the set exercises and learning activities:

- GP4.1 nano
- GPT4o-mini
- Gemini 2.0 Flash Lite
- Mistral Small Instruct

Caution:
The SAP Landscape provided is shared with all learners and is only intended for use with the activities and exercises provided. Please make sure to follow the instructions of the exercises, staying within the resource allocation limits for your user.

The access to the system is using Single Sign-On with your SAP ID used to sign in to your SAP learning portal, you will assigned a BTP user with all required SAP BTP roles assigned to your user.

You SAP BTP assigned user will consist of a letter A and four digits number (A####). The group numbers in the course material are represented by ## or XX.

- For example if your group number is 123 then your user will be A0123.

After your enrollment is successful - you can click access button which should open your AI Launchpad.

> Note:
> Please make sure you are always using your assigned group number, so you don't interfere with data sets used by another learner.

How to find the Group Number

Click the access button on the practice system details page and you will find your group number displayed beneath the course title in the pop-up window.

Or go to *My Learning → Practice Systems* and choose a practice system. You will find your group number displayed beneath the title of the practice system.

> Note:
> Do not close or navigate away from the Practice System page while you are working in a learning system. Some content may rely on this window to save your progress.

4. Additional System Setup Tasks

None

5. Known System Issues

None

6. Additional Hints And Tips For Exercises of This Course

Like in many web-based applications, data caching sometimes affects access to a particular feature or execution of a workflow. If you suspect an anomaly could be due to data caching, close all instances of your web browser, then start your web browser again and use its native features to clear the cache. These features can be found in Google Chrome, Microsoft Edge and Mozilla Firefox under the History menu.

IMPORTANT: Please always complete the exercise you are working on before you suspend the system to avoid potential system inconsistencies and failure.

> Caution:
> Only one learning system can be accessed at a time.

7. Global Learning System Support

1. Go to SAP User Support Centerhttps://support.learning.sap.com/#.
2. Click *Need more help?* at the bottom of the page.
3. Click on *Create Case*.
4. Briefly describe your issue in the *Subject* field.

5. Click on the *Service Category* drop-down and select *Practice Systems*.
6. In the *Description*, copy and paste the text for **Self-paced Learners** in the box below.
7. Fill in the text with the necessary information about the issue.
8. Attach the screenshot of the issue if available (max 20MB). Click the option *choose a file or drag and drop*. Hit *Upload* when the file has been chosen.
9. Click *Submit*.

AIG02

Solving Your Business Problems Using Prompts and LLMs in SAP Generative AI Hub

EXERCISES AND SOLUTIONS

Course Version: 11
Exercise Duration: 2 Hours

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

Develop a Prompt in SAP AI Launchpad

Business Example

Facility solutions company receives thousands of mails pertaining to customers' requests, complaints, and other messages. The company maintains internal applications to address customer requests and complaints and prioritize them to address these requests in a timely and correct manner.

However, categorizing these mails include manual process to transfer data from mails to internal applications, categorizing, and then prioritizing these tasks.

They turn towards generative AI hub to address this problem.

Developing a prompt to solve a business problem is a stepwise ideation process.

In this exercise, we will develop a prompt to address this problem.

Prerequisites

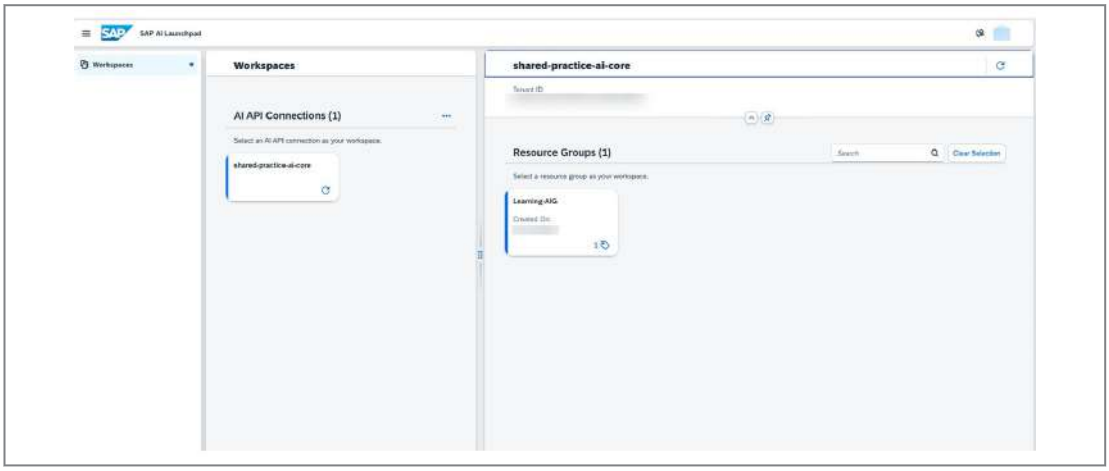
Login credentials for SAP AI Launchpad. Refer to the SSG.

> Caution:
> This SAP's generative AI hub use is limited to the set exercises in the handbook provided by SAP practice system; all other activities are not permitted. User activity may be checked at any time for unauthorized activities. Please read SAP AI Terms and Conditions here .

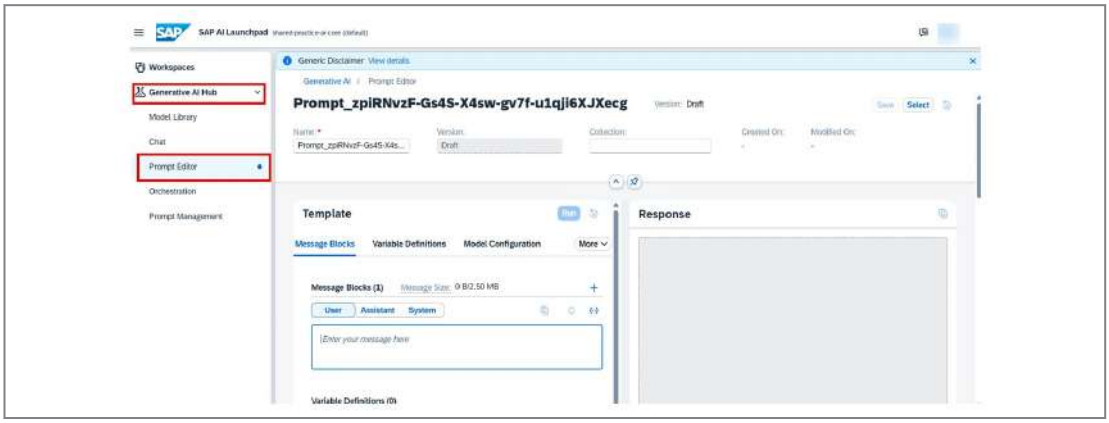
Task 1: Develop a Basic Prompt

We will now find the urgency and sentiment in an incoming mail.

1. Log into Generative AI Hub , you will see the SAP AI Launchpad. Ensure that the **Learning-AIG** resource group is selected. After a few moments, Generative AI Hub option is displayed in the left pane.

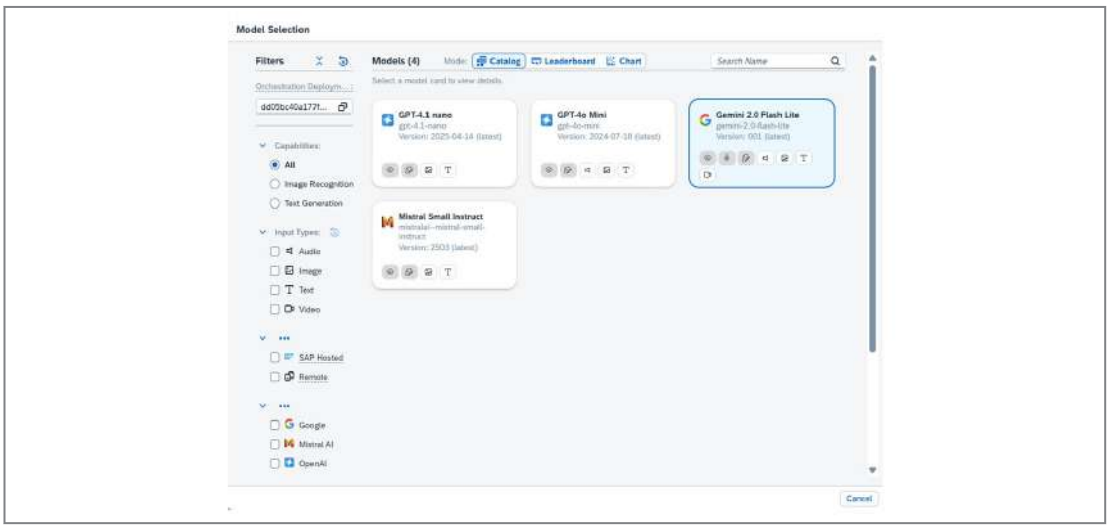


2. Expand the Generative AI Hub and then select *Prompt Editor* in the left pane.



3. Navigate downward and select *Model Configuration*. Choose the appropriate model.

4. The *Model Selection* dialog box is displayed. You can also use the search bar to find the correct model.

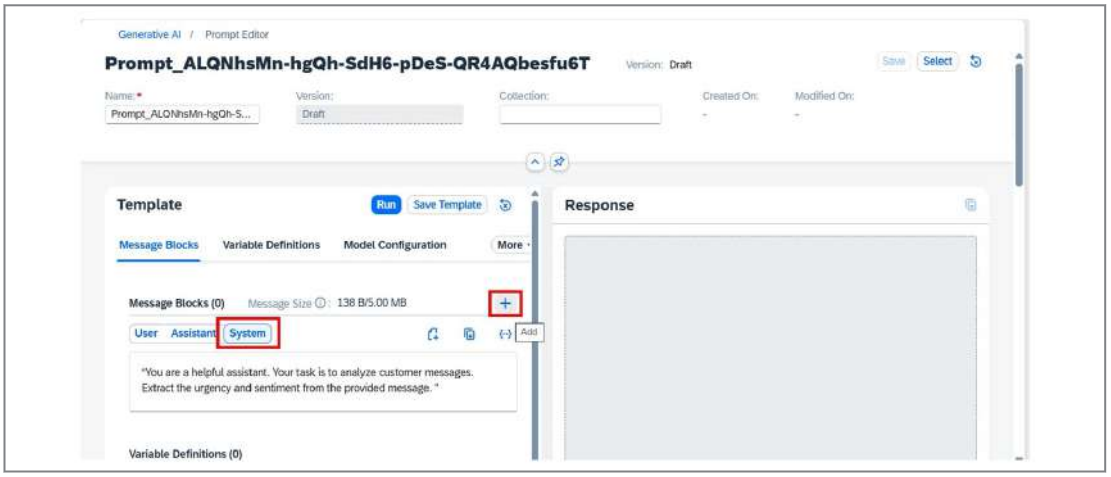


5. For this exercise, we start with using a Google model. Select the latest Google model.

6. In **Message Blocks,** select the **System** role. Copy the following prompt and paste it in the System text box in Message Blocks.

```
Prompt:
"You are a helpful assistant. Your task is to analyze customer
messages. Extract the urgency and sentiment from the provided message.
"
```

7. Select **Add Role**, to add the user role.



8. Ensure that the User role is selected. Copy the following prompt and paste it in the User text box in Message Blocks.

```
text box in Message Blocks.
"
Analyze the following message:
---
Subject: Urgent HVAC System Repair Needed

Dear Support Team,

I hope this message finds you well. My name is [Sender], and I am
reaching out to you from [Residential Complex Name], where I have been
residing for the past few years. I have always appreciated the
meticulous care and attention your team provides in maintaining our
facilities.

However, I am currently facing a pressing issue with the HVAC system
in my apartment. Over the past few days, the system has been
malfunctioning, resulting in inconsistent temperatures and, at times,
complete shutdowns. Given the current weather conditions, this has
become quite unbearable and is affecting my daily routine
significantly.

I have attempted to troubleshoot the problem by resetting the system
and checking the thermostat settings, but these efforts have not
yielded any improvement. The situation seems to be beyond my control
and requires professional intervention.

I kindly request that a repair team be dispatched immediately to
address this urgent issue. The urgency of the matter cannot be
overstated, as it is impacting not only my comfort but also my ability
to carry out daily activities effectively.

Thank you for your prompt attention to this matter. I look forward to
your swift response and resolution.

Best regards,
[Sender]
"
```

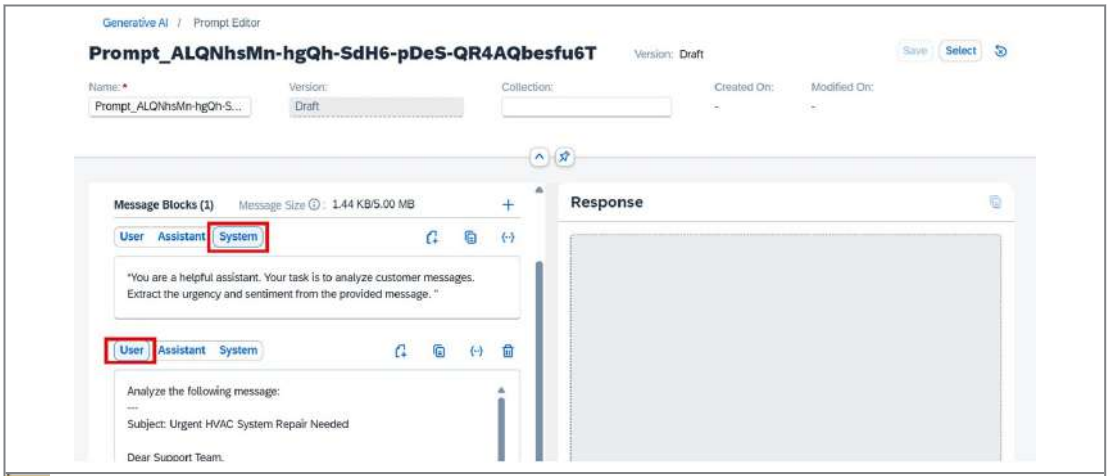
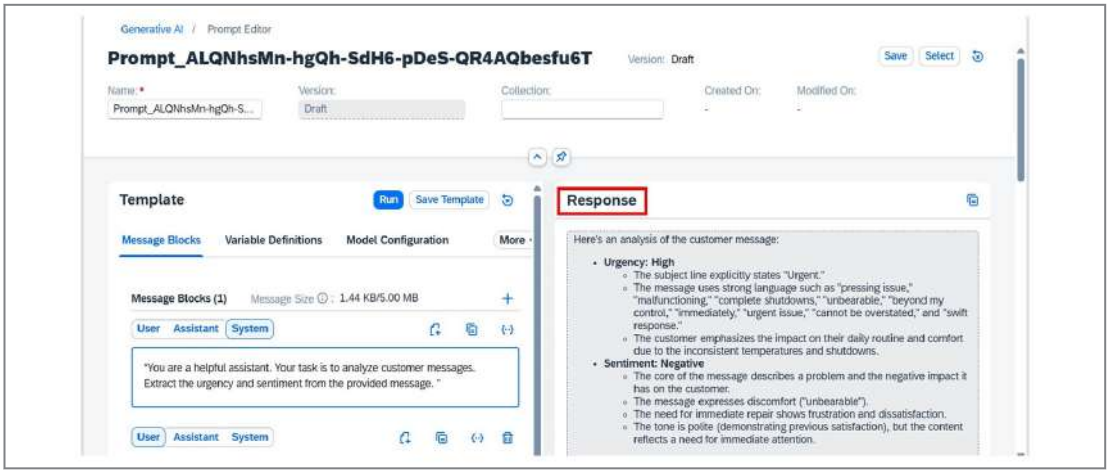
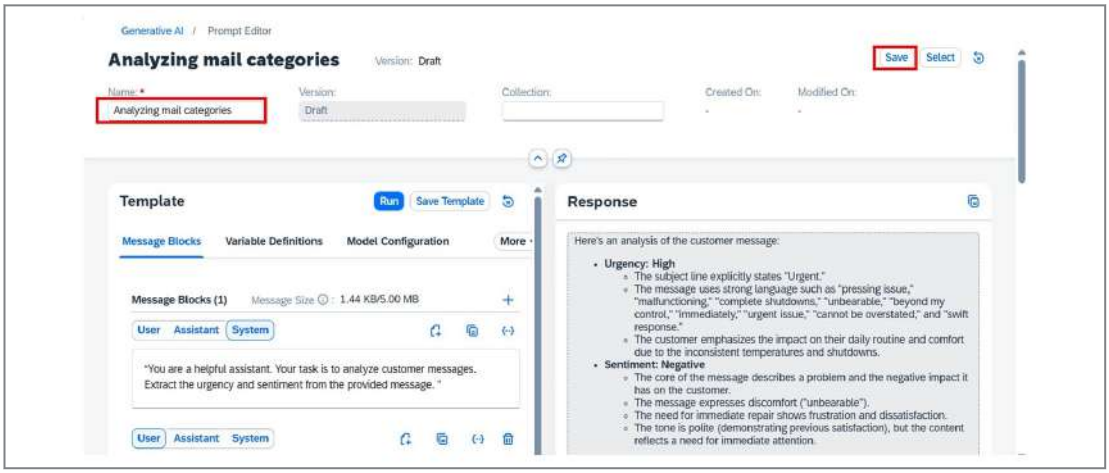


Figure 1: Ex1-6.pptx

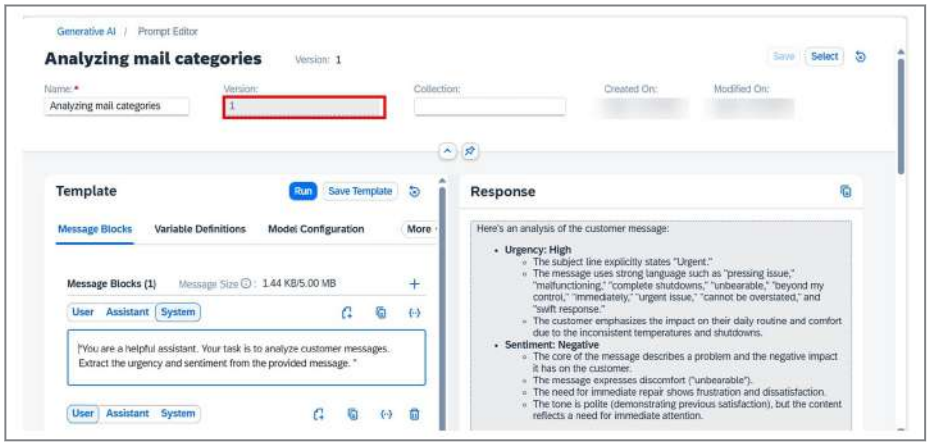
9. Click the **Run** button. You can see the Urgency and the Sentiment of the mail is generated in the **Response** text box. However, as we can see this response is too lengthy. Our objective is to assign urgency and sentiment to the mail that can be used as an input for software applications. This lengthy response is not useful for us as of now. Let us continue to develop it.



10. Give a proper name to your prompt and save it, as shown in the following screenshot.



11. The message is saved for future development with version 1.



Note:
You may get a slightly different response to the one shown here and in all the remaining responses of models shown in this learning journey.

When you execute the same prompt in your machine, an LLM produces varying outputs due to its probabilistic nature, temperature setting, and non-deterministic architecture, leading to different responses even with slight setting changes or internal state shifts.

Task 2: Assign Values to Urgency and Sentiment to the Existing Prompt

We will now assign some basic urgency and sentiment values and execute the prompt.

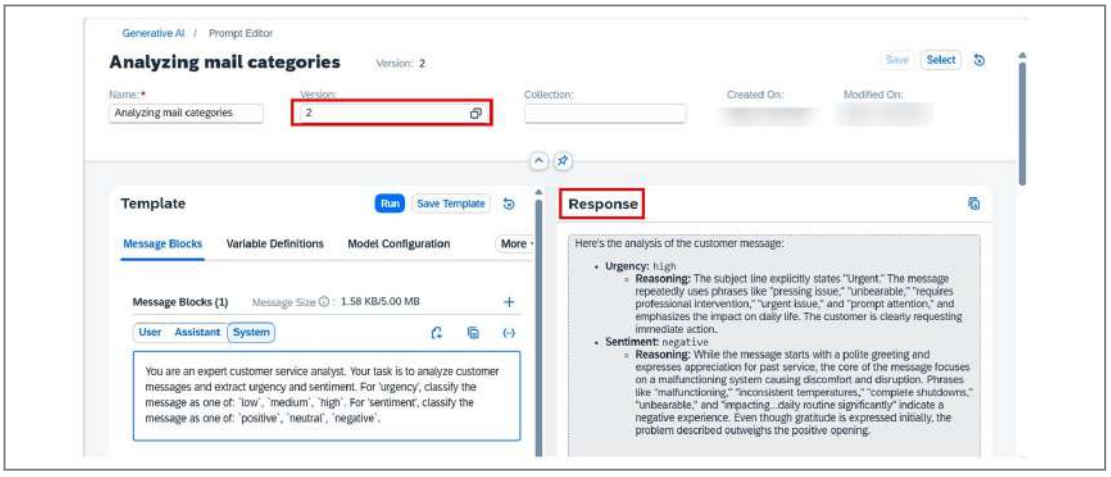
1. Update the saved prompt in the Prompt editor as to assign value to urgency and sentiment. Use the following prompt in the **System** role.

```
You are an expert customer service analyst. Your task is to analyze
customer messages and extract urgency and sentiment. For 'urgency',
classify the message as one of: `low`, `medium`, `high`. For
'sentiment', classify the message as one of: `positive`, `neutral`,
`negative`.
```

   Continue the same prompt in the **User** role.

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run.** You get a response in the **Response** field. You can see that the urgency and sentiment of the mail is generated. These values are based on assigned values.

3. Click **Save** button to save the updated version of the prompt. Version 2 of the prompt is saved.



Task 3: Generate JSON output

We need the output of these values in a format that can be used as an input for software. We will use the JSON output.

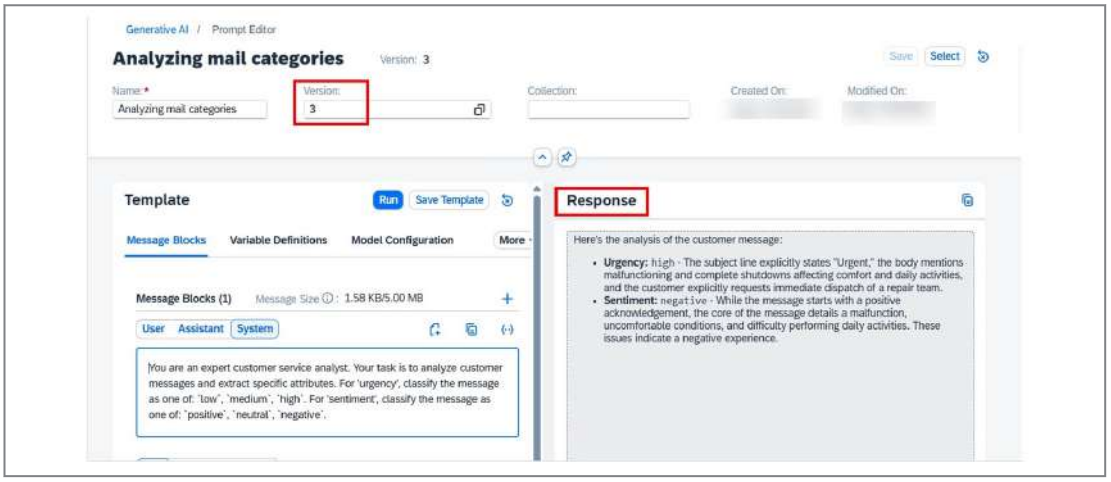
1. Update the **System** role prompt with the following text.

```
You are an expert customer service analyst. Your task is to analyze
customer messages and extract specific attributes. For 'urgency',
classify the message as one of: `low`, `medium`, `high`. For
'sentiment', classify the message as one of: `positive`, `neutral`,
`negative`.
```

   The prompt is the same as the previous prompt, but the last few sentences now request a JSON string.

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run**. You can see the JSON output.

3. Click Save. The following screenshot shows version 3 of the prompt.



Task 4: Ensure that the JSON formatting is correct

Even with instructions for JSON, LLMs can sometimes include extraneous characters like markdown code blocks (json ...) or extra whitespace, which can break automated parsing. This step adds explicit instructions to ensure a clean JSON string that can be parsed by an application.

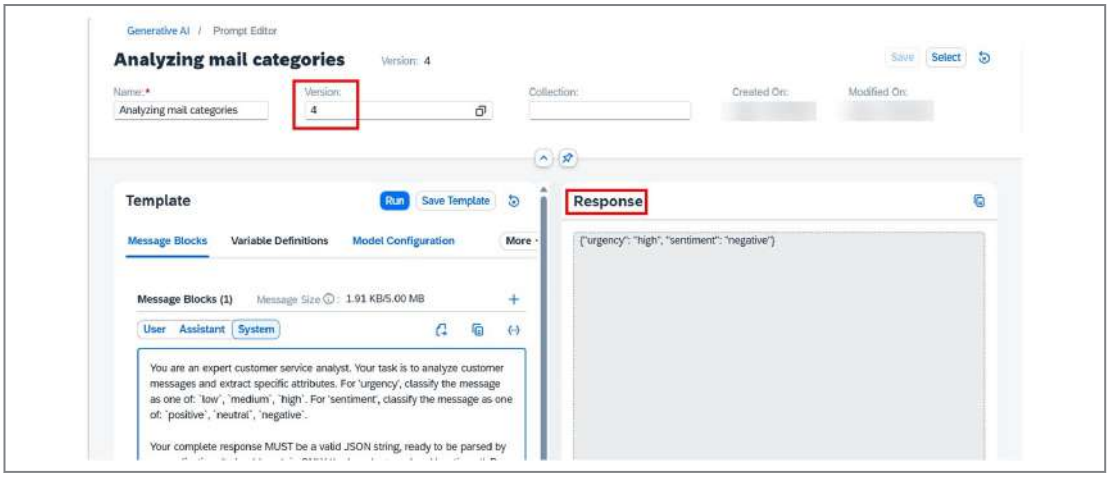
1. Update the **System** role prompt with the following text.

```
You are an expert customer service analyst. Your task is to analyze
customer messages and extract specific attributes. For 'urgency',
classify the message as one of: `low`, `medium`, `high`. For
'sentiment', classify the message as one of: `positive`, `neutral`,
`negative`.

Your complete response MUST be a valid JSON string, ready to be parsed
by an application. It should contain ONLY the keys 'urgency' and
'sentiment'. Do not include any other text, explanations, or
formatting like markdown code blocks (e.g., ```json). Ensure there are
no newlines or unnecessary whitespaces outside the JSON structure.
```

   This prompt gives clear instruction to provide a clean format without any quotes or whitespaces.

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run**. You can see that the JSON output is clear. This is ready for software consumption.

3. Click **Save.** The following screenshot shows version 4 of the prompt.



Task 5: Generate Simple Categories Based on Business Functions

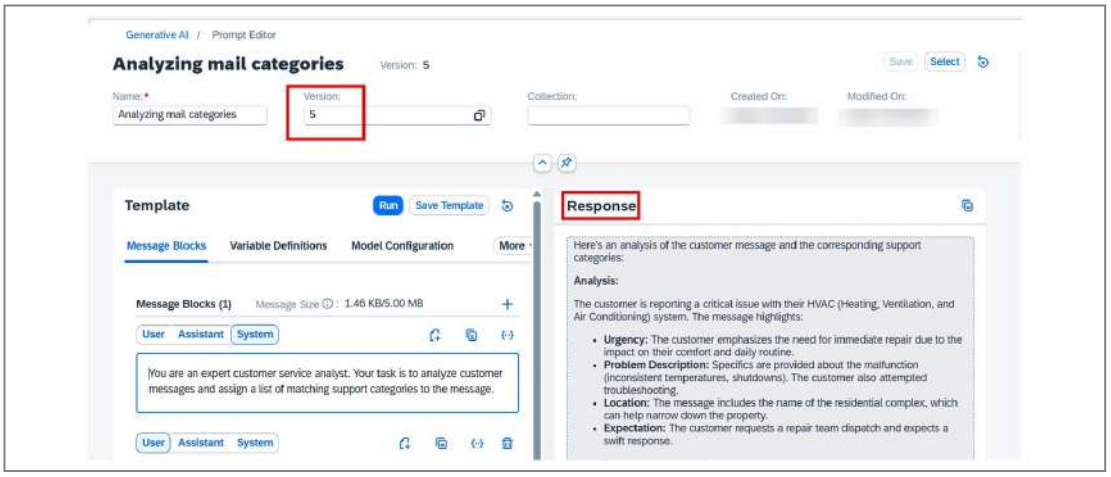
We have associated urgency and sentiment to a mail. However, we also need more tags for each message to categorize them for business needs.

1. Start with a simple prompt:

```
"You are an expert customer service analyst. Your task is to analyze
customer messages and assign a list of matching support categories to
the message."
```

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run**. You can see categories are assigned to the message.

3. Click **Save.** The following screenshot shows version 5 of the prompt.



Task 6: Assign Values to Categories From a List

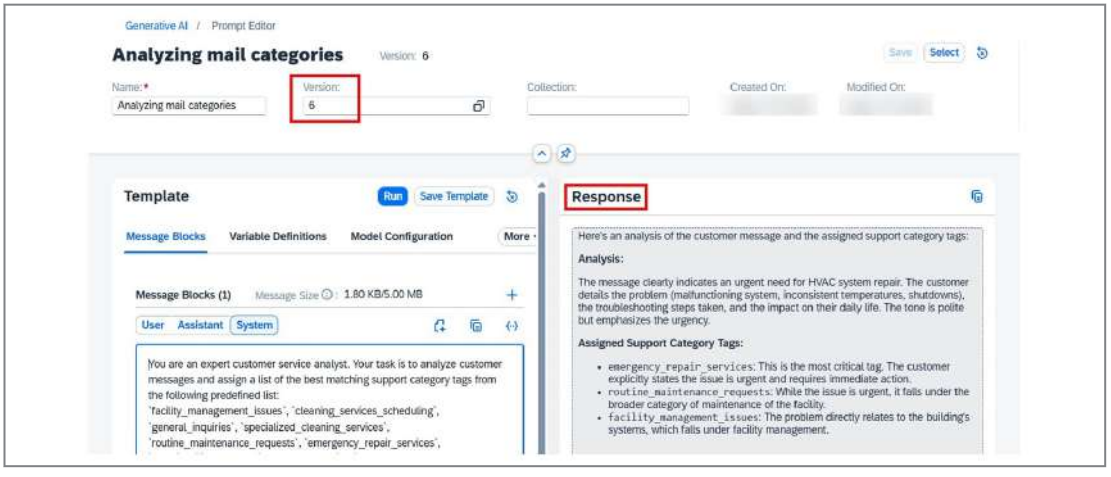
To ensure consistency and alignment with the company's internal processing, we need LLM to select categories from a specific, predefined list. Ensuring the output corresponds to business requirements is an essential aspect of solving a business problem.

1. Use the following prompt:

```
"You are an expert customer service analyst. Your task is to analyze
customer messages and assign a list of the best matching support
category tags from the following predefined list:
`facility_management_issues`, `cleaning_services_scheduling`,
`general_inquiries`, `specialized_cleaning_services`,
`routine_maintenance_requests`, `emergency_repair_services`,
`sustainability_and_environmental_practices`,
`training_and_support_requests`, `quality_and_safety_concerns`,
`customer_feedback_and_complaints`."
```

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run**. You can see that categories are assigned to values from the list. They are streamlined for business processing.

3. Click **Save.** The following screenshot shows version 6 of the prompt.



Task 7: Generate JSON Output for Category Values

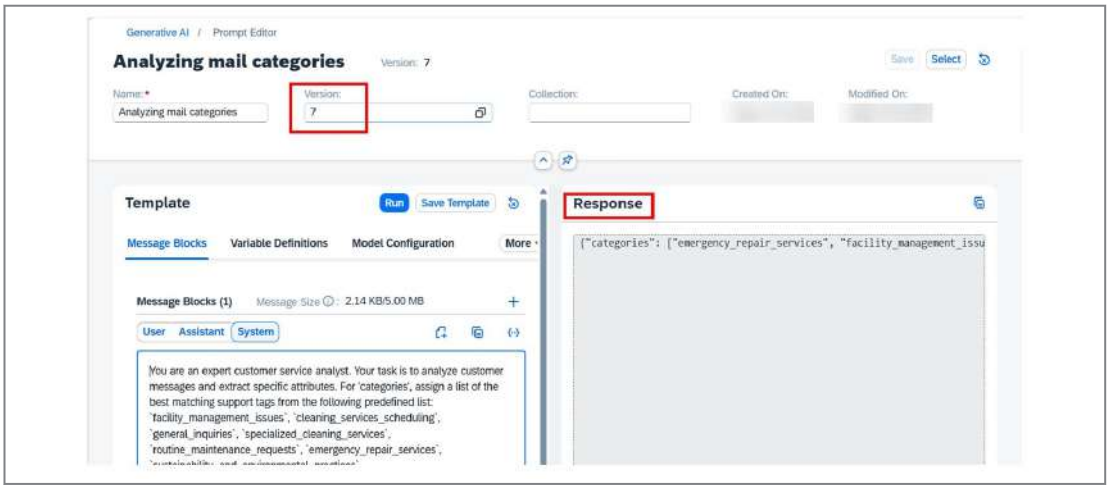
Like Step 3, we need the category values in a JSON output for programmatic processing. We will also reiterate the importance of clean JSON formatting.

1. Use the following prompt:

```
"You are an expert customer service analyst. Your task is to analyze
customer messages and extract specific attributes. For 'categories',
assign a list of the best matching support tags from the following
predefined list:
`facility_management_issues`, `cleaning_services_scheduling`,
`general_inquiries`, `specialized_cleaning_services`,
`routine_maintenance_requests`, `emergency_repair_services`,
`sustainability_and_environmental_practices`,
`training_and_support_requests`, `quality_and_safety_concerns`,
`customer_feedback_and_complaints`.
Your complete response MUST be a valid JSON string, ready to be parsed
by an application. It should contain ONLY the key 'categories'. Do not
include any other text, explanations, or formatting like markdown code
blocks. Ensure there are no newlines or unnecessary whitespaces
outside the JSON structure."
```

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run.** You can see categories in the JSON output that can be processed in software applications.

3. Click **Save.** The following screenshot shows version 7 of the prompt.



Task 8: Combine All Steps

We have taken a step-by-step approach to arrive at proper values of urgency, sentiment, and categories in JSON format. Now, let us combine all these steps into a single prompt.

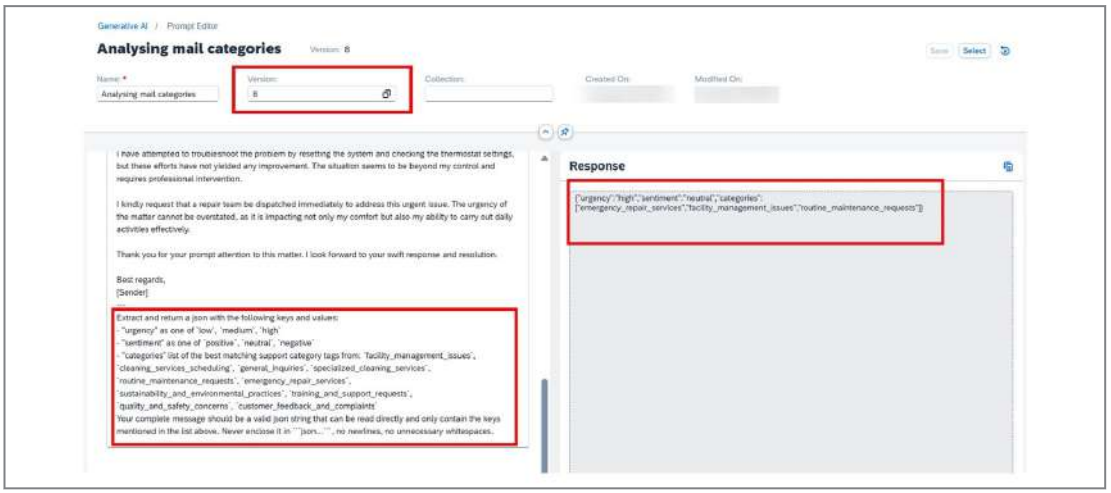
1. Use the following prompt:

```
"You are an expert customer service analyst for a facility management
company. Your task is to analyze incoming customer messages and
extract specific attributes for automated processing.

For 'urgency', classify the message as one of: `low`, `medium`, `high`.
For 'sentiment', classify the message as one of: `positive`,
`neutral`, `negative`.
For 'categories', assign a list of the best matching support tags from
the following predefined list:
`facility_management_issues`, `cleaning_services_scheduling`,
`general_inquiries`, `specialized_cleaning_services`,
`routine_maintenance_requests`, `emergency_repair_services`,
`sustainability_and_environmental_practices`,
`training_and_support_requests`, `quality_and_safety_concerns`,
`customer_feedback_and_complaints`.
```

```
Your complete response MUST be a valid JSON string, ready to be parsed
by an application. It should contain ONLY the keys 'urgency',
'sentiment', and 'categories'. Do not include any other text,
explanations, or formatting like markdown code blocks (e.g., ```json).
Ensure there are no newlines or unnecessary whitespaces outside the
JSON structure."
```

2. Copy the prompt and paste it in the **System role in Message Blocks** text box, and then click **Run.** You can see the consolidated output that assigns urgency, sentiment, and categories to customer messages that can be used in software applications.

3. Click **Save.** The following screenshot shows version 8 of the prompt.



Congratulations. You have successfully developed a prompt to solve a business problem.

Unit 1
Solution 1

Develop a Prompt in SAP AI Launchpad

Business Example

Facility solutions company receives thousands of mails pertaining to customers' requests, complaints, and other messages. The company maintains internal applications to address customer requests and complaints and prioritize them to address these requests in a timely and correct manner.

However, categorizing these mails include manual process to transfer data from mails to internal applications, categorizing, and then prioritizing these tasks.

They turn towards generative AI hub to address this problem.

Developing a prompt to solve a business problem is a stepwise ideation process.

In this exercise, we will develop a prompt to address this problem.

Prerequisites

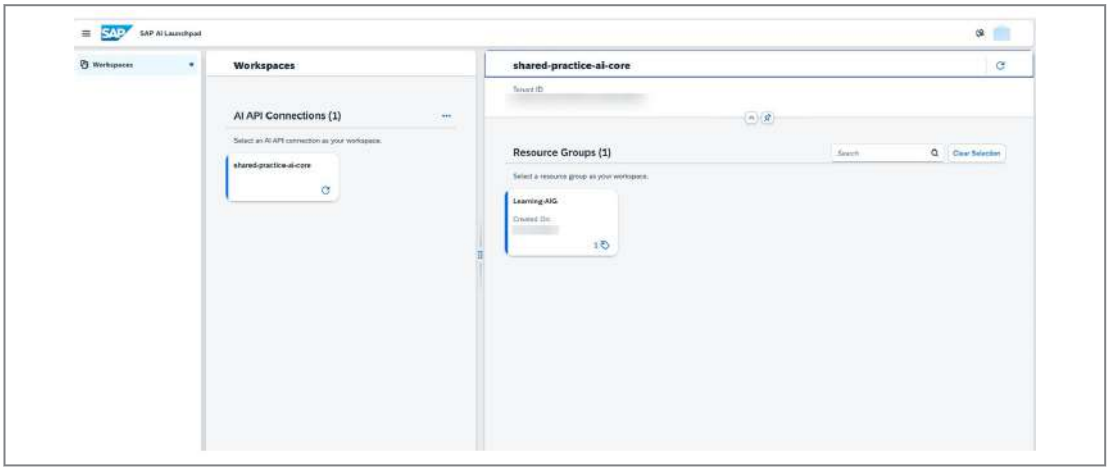
Login credentials for SAP AI Launchpad. Refer to the SSG.

> Caution:
> This SAP's generative AI hub use is limited to the set exercises in the handbook provided by SAP practice system; all other activities are not permitted. User activity may be checked at any time for unauthorized activities. Please read SAP AI Terms and Conditions here .

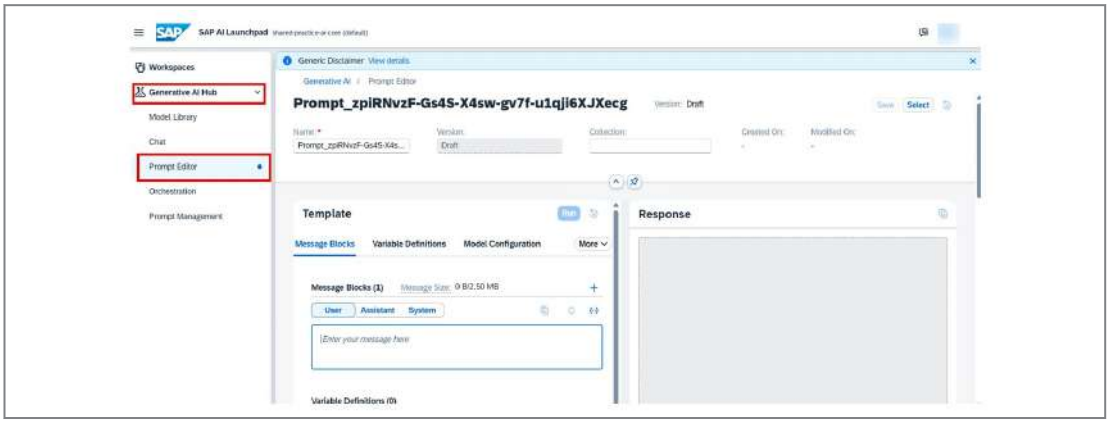
Task 1: Develop a Basic Prompt

We will now find the urgency and sentiment in an incoming mail.

1. Log into Generative AI Hub , you will see the SAP AI Launchpad. Ensure that the **Learning-AIG** resource group is selected. After a few moments, Generative AI Hub option is displayed in the left pane.

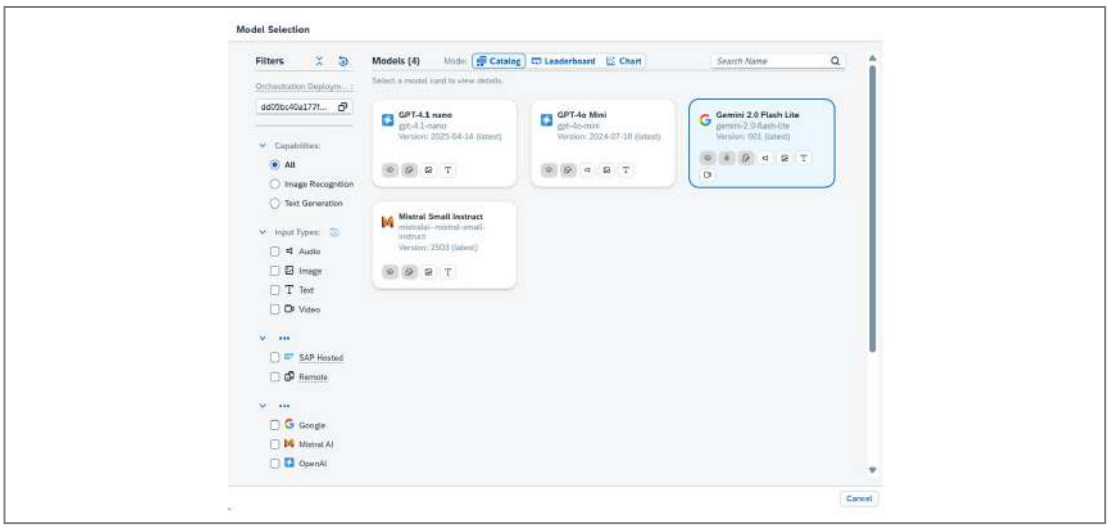


2. Expand the Generative AI Hub and then select *Prompt Editor* in the left pane.



3. Navigate downward and select *Model Configuration*. Choose the appropriate model.

4. The *Model Selection* dialog box is displayed. You can also use the search bar to find the correct model.

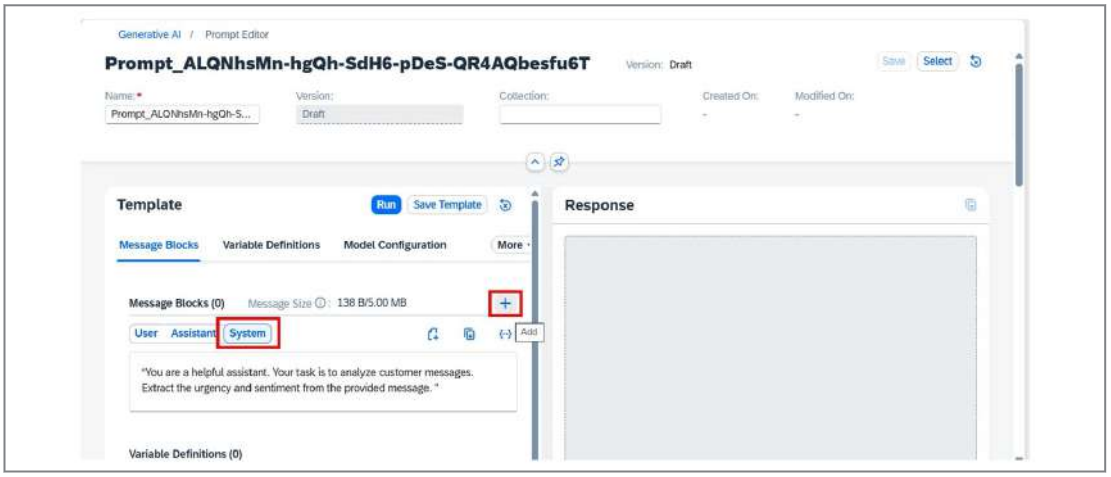


5. For this exercise, we start with using a Google model. Select the latest Google model.

6. In **Message Blocks,** select the **System** role. Copy the following prompt and paste it in the System text box in Message Blocks.

```
Prompt:
"You are a helpful assistant. Your task is to analyze customer
messages. Extract the urgency and sentiment from the provided message.
"
```

7. Select **Add Role**, to add the user role.



8. Ensure that the User role is selected. Copy the following prompt and paste it in the User text box in Message Blocks.

```
text box in Message Blocks.
"
Analyze the following message:
---
Subject: Urgent HVAC System Repair Needed

Dear Support Team,

I hope this message finds you well. My name is [Sender], and I am
reaching out to you from [Residential Complex Name], where I have been
residing for the past few years. I have always appreciated the
meticulous care and attention your team provides in maintaining our
facilities.

However, I am currently facing a pressing issue with the HVAC system
in my apartment. Over the past few days, the system has been
malfunctioning, resulting in inconsistent temperatures and, at times,
complete shutdowns. Given the current weather conditions, this has
become quite unbearable and is affecting my daily routine
significantly.

I have attempted to troubleshoot the problem by resetting the system
and checking the thermostat settings, but these efforts have not
yielded any improvement. The situation seems to be beyond my control
and requires professional intervention.

I kindly request that a repair team be dispatched immediately to
address this urgent issue. The urgency of the matter cannot be
overstated, as it is impacting not only my comfort but also my ability
to carry out daily activities effectively.

Thank you for your prompt attention to this matter. I look forward to
your swift response and resolution.

Best regards,
[Sender]
"
```

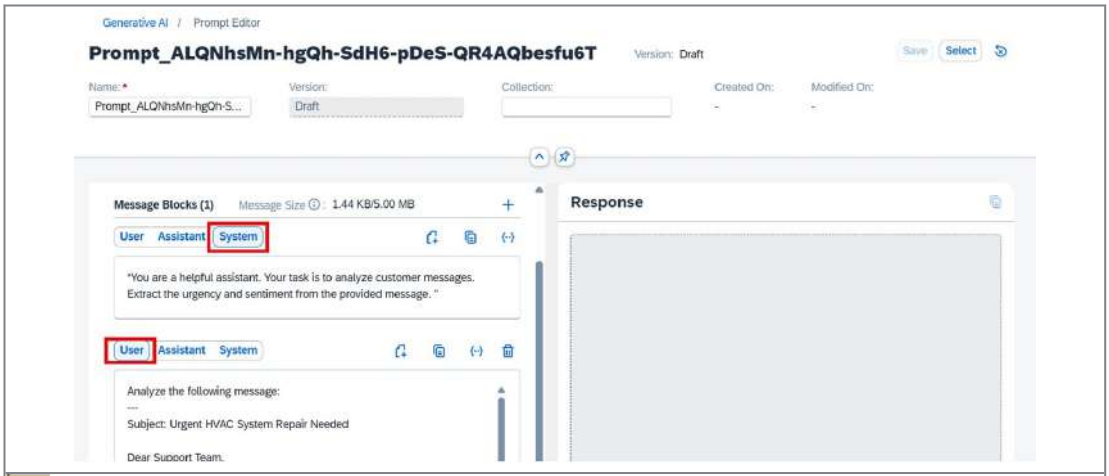
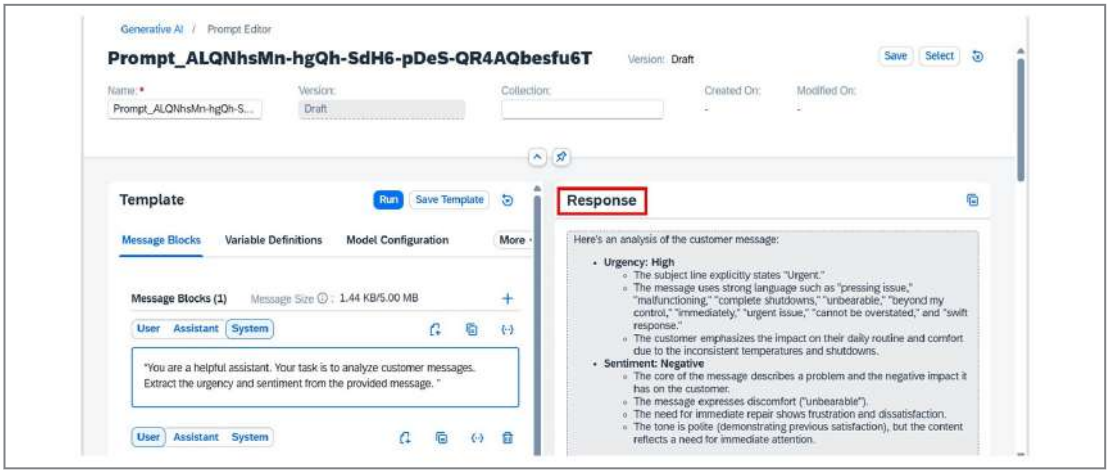
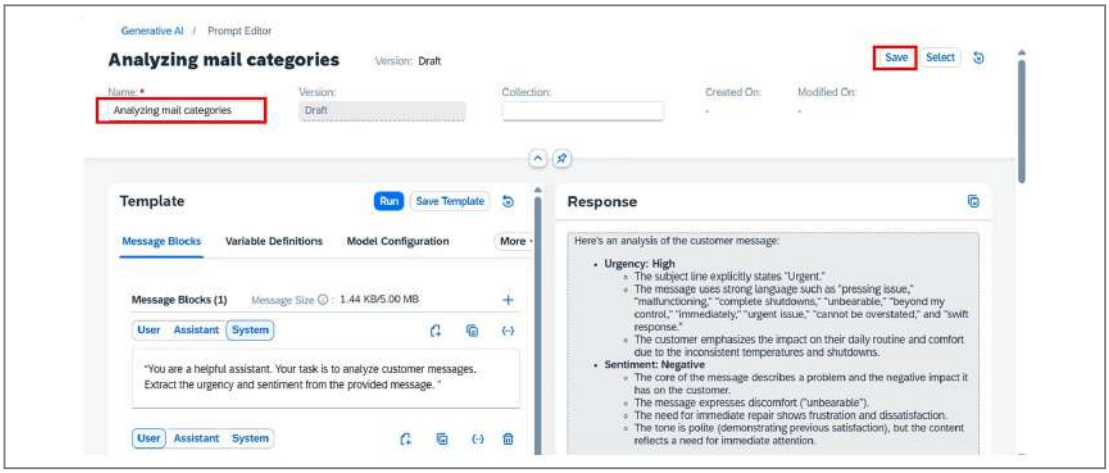


Figure 1: Ex1-6.pptx

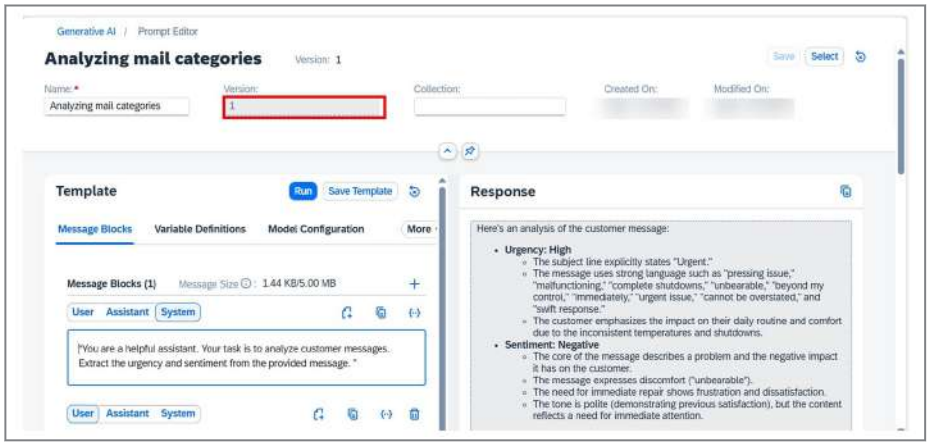
9. Click the **Run** button. You can see the Urgency and the Sentiment of the mail is generated in the **Response** text box. However, as we can see this response is too lengthy. Our objective is to assign urgency and sentiment to the mail that can be used as an input for software applications. This lengthy response is not useful for us as of now. Let us continue to develop it.



10. Give a proper name to your prompt and save it, as shown in the following screenshot.



11. The message is saved for future development with version 1.



Note:
You may get a slightly different response to the one shown here and in all the remaining responses of models shown in this learning journey.

When you execute the same prompt in your machine, an LLM produces varying outputs due to its probabilistic nature, temperature setting, and non-deterministic architecture, leading to different responses even with slight setting changes or internal state shifts.

Task 2: Assign Values to Urgency and Sentiment to the Existing Prompt

We will now assign some basic urgency and sentiment values and execute the prompt.

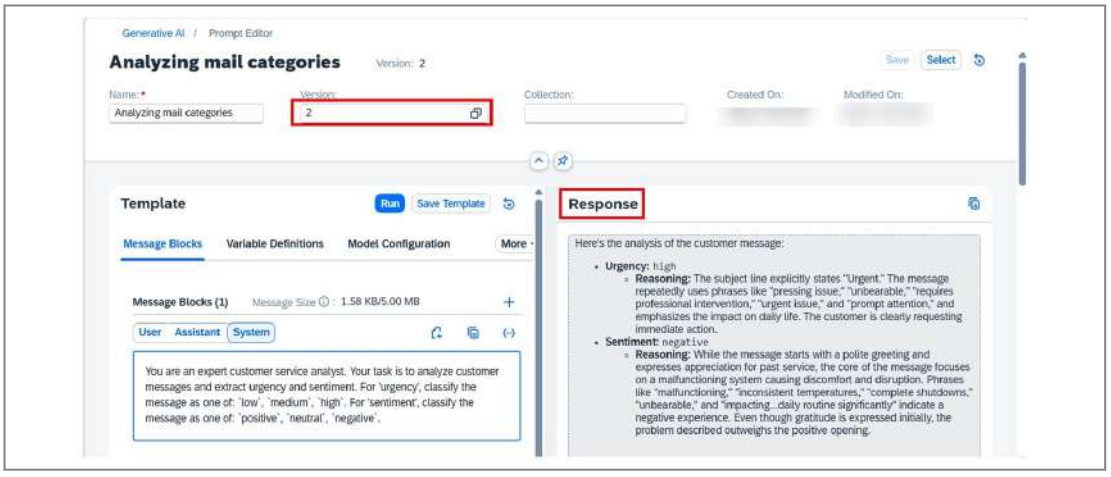
1. Update the saved prompt in the Prompt editor as to assign value to urgency and sentiment. Use the following prompt in the **System** role.

```
You are an expert customer service analyst. Your task is to analyze
customer messages and extract urgency and sentiment. For 'urgency',
classify the message as one of: `low`, `medium`, `high`. For
'sentiment', classify the message as one of: `positive`, `neutral`,
`negative`.
```

   Continue the same prompt in the **User** role.

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run.** You get a response in the **Response** field. You can see that the urgency and sentiment of the mail is generated. These values are based on assigned values.

3. Click **Save** button to save the updated version of the prompt. Version 2 of the prompt is saved.



Task 3: Generate JSON output

We need the output of these values in a format that can be used as an input for software. We will use the JSON output.

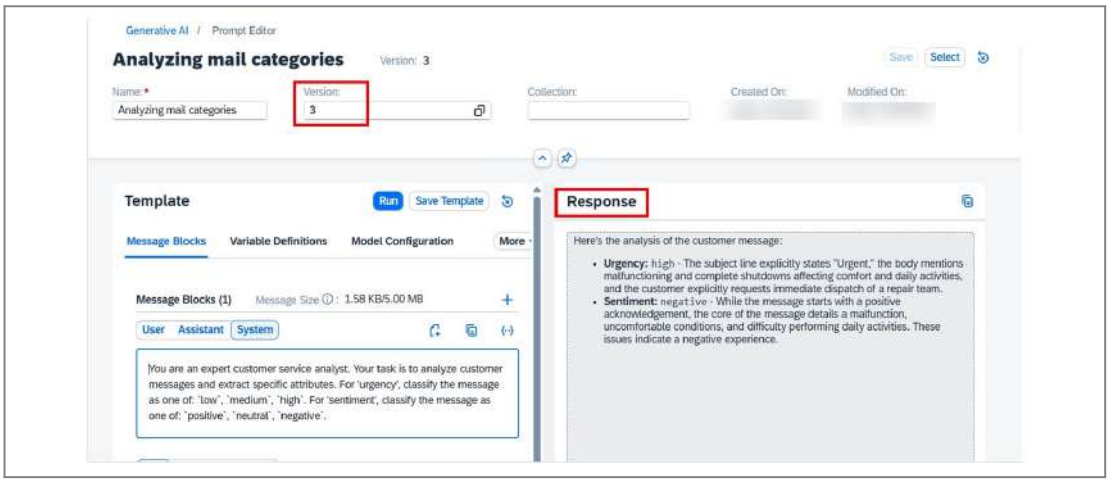
1. Update the **System** role prompt with the following text.

```
You are an expert customer service analyst. Your task is to analyze
customer messages and extract specific attributes. For 'urgency',
classify the message as one of: `low`, `medium`, `high`. For
'sentiment', classify the message as one of: `positive`, `neutral`,
`negative`.
```

   The prompt is the same as the previous prompt, but the last few sentences now request a JSON string.

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run**. You can see the JSON output.

3. Click Save. The following screenshot shows version 3 of the prompt.



Task 4: Ensure that the JSON formatting is correct

Even with instructions for JSON, LLMs can sometimes include extraneous characters like markdown code blocks (json ...) or extra whitespace, which can break automated parsing. This step adds explicit instructions to ensure a clean JSON string that can be parsed by an application.

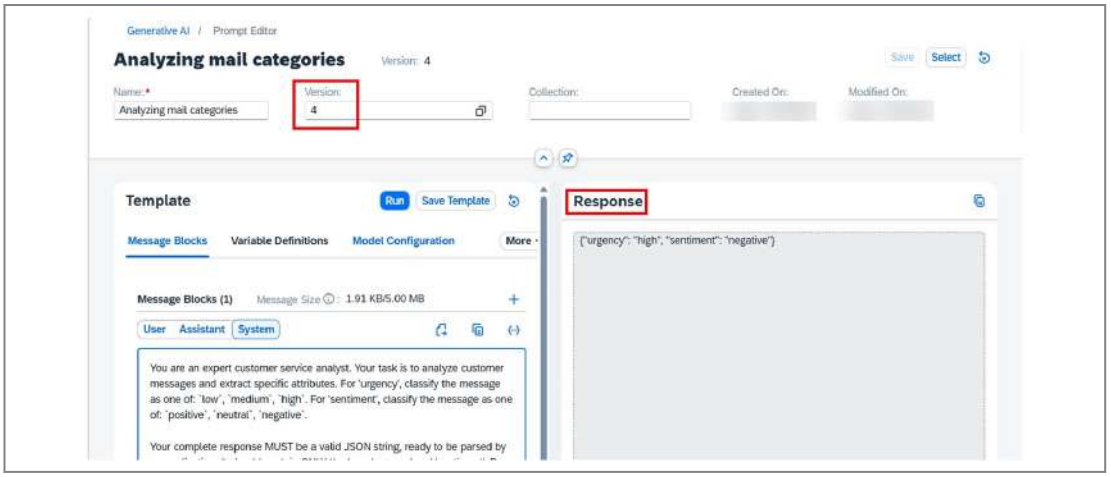
1. Update the **System** role prompt with the following text.

```
You are an expert customer service analyst. Your task is to analyze
customer messages and extract specific attributes. For 'urgency',
classify the message as one of: `low`, `medium`, `high`. For
'sentiment', classify the message as one of: `positive`, `neutral`,
`negative`.

Your complete response MUST be a valid JSON string, ready to be parsed
by an application. It should contain ONLY the keys 'urgency' and
'sentiment'. Do not include any other text, explanations, or
formatting like markdown code blocks (e.g., ```json). Ensure there are
no newlines or unnecessary whitespaces outside the JSON structure.
```

   This prompt gives clear instruction to provide a clean format without any quotes or whitespaces.

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run**. You can see that the JSON output is clear. This is ready for software consumption.

3. Click **Save.** The following screenshot shows version 4 of the prompt.



Task 5: Generate Simple Categories Based on Business Functions

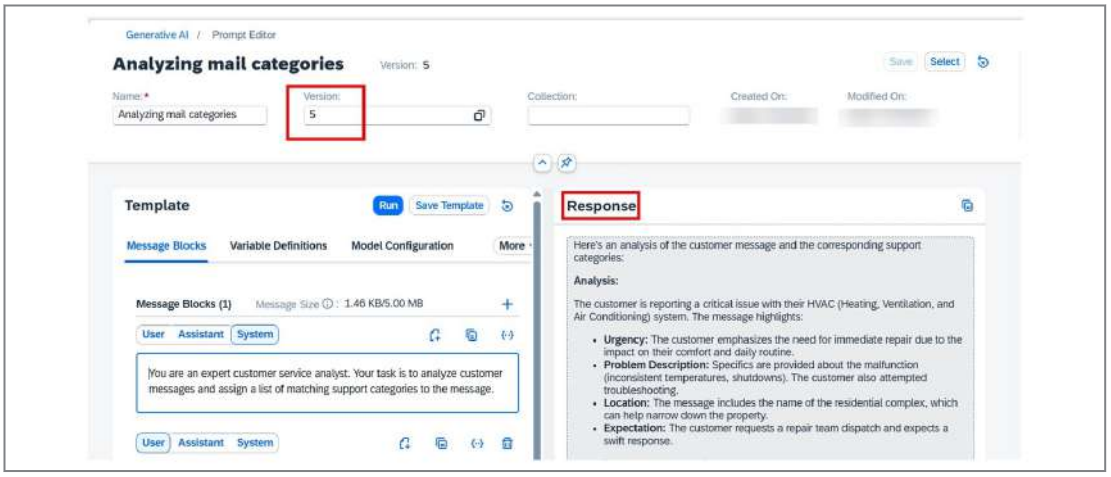
We have associated urgency and sentiment to a mail. However, we also need more tags for each message to categorize them for business needs.

1. Start with a simple prompt:

```
"You are an expert customer service analyst. Your task is to analyze
customer messages and assign a list of matching support categories to
the message."
```

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run**. You can see categories are assigned to the message.

3. Click **Save.** The following screenshot shows version 5 of the prompt.



Task 6: Assign Values to Categories From a List

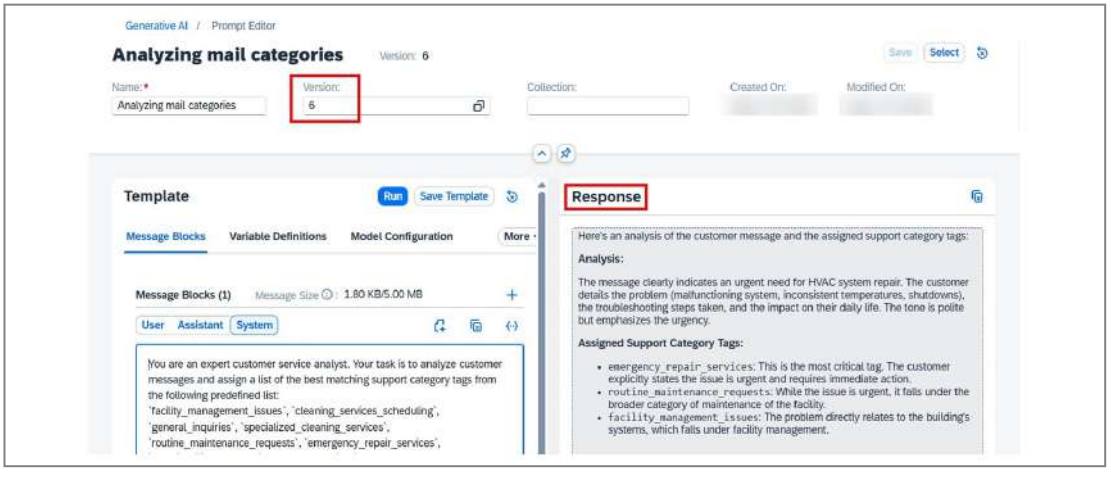
To ensure consistency and alignment with the company's internal processing, we need LLM to select categories from a specific, predefined list. Ensuring the output corresponds to business requirements is an essential aspect of solving a business problem.

1. Use the following prompt:

```
"You are an expert customer service analyst. Your task is to analyze
customer messages and assign a list of the best matching support
category tags from the following predefined list:
`facility_management_issues`, `cleaning_services_scheduling`,
`general_inquiries`, `specialized_cleaning_services`,
`routine_maintenance_requests`, `emergency_repair_services`,
`sustainability_and_environmental_practices`,
`training_and_support_requests`, `quality_and_safety_concerns`,
`customer_feedback_and_complaints`."
```

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run**. You can see that categories are assigned to values from the list. They are streamlined for business processing.

3. Click **Save.** The following screenshot shows version 6 of the prompt.



Task 7: Generate JSON Output for Category Values

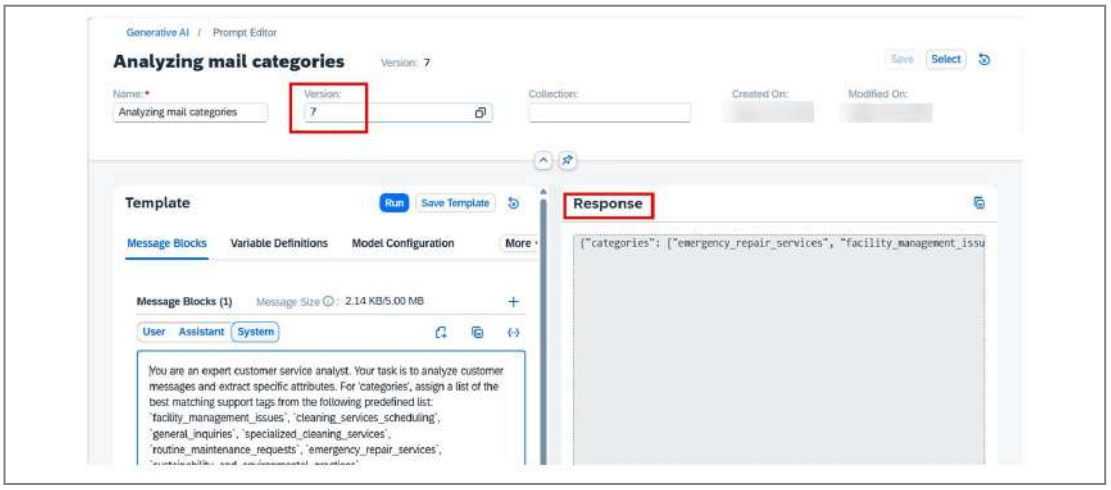
Like Step 3, we need the category values in a JSON output for programmatic processing. We will also reiterate the importance of clean JSON formatting.

1. Use the following prompt:

```
"You are an expert customer service analyst. Your task is to analyze
customer messages and extract specific attributes. For 'categories',
assign a list of the best matching support tags from the following
predefined list:
`facility_management_issues`, `cleaning_services_scheduling`,
`general_inquiries`, `specialized_cleaning_services`,
`routine_maintenance_requests`, `emergency_repair_services`,
`sustainability_and_environmental_practices`,
`training_and_support_requests`, `quality_and_safety_concerns`,
`customer_feedback_and_complaints`.
Your complete response MUST be a valid JSON string, ready to be parsed
by an application. It should contain ONLY the key 'categories'. Do not
include any other text, explanations, or formatting like markdown code
blocks. Ensure there are no newlines or unnecessary whitespaces
outside the JSON structure."
```

2. Copy the prompt and paste it in the **System** role in the **Message Blocks** text box, and then click **Run.** You can see categories in the JSON output that can be processed in software applications.

3. Click **Save.** The following screenshot shows version 7 of the prompt.



Task 8: Combine All Steps

We have taken a step-by-step approach to arrive at proper values of urgency, sentiment, and categories in JSON format. Now, let us combine all these steps into a single prompt.

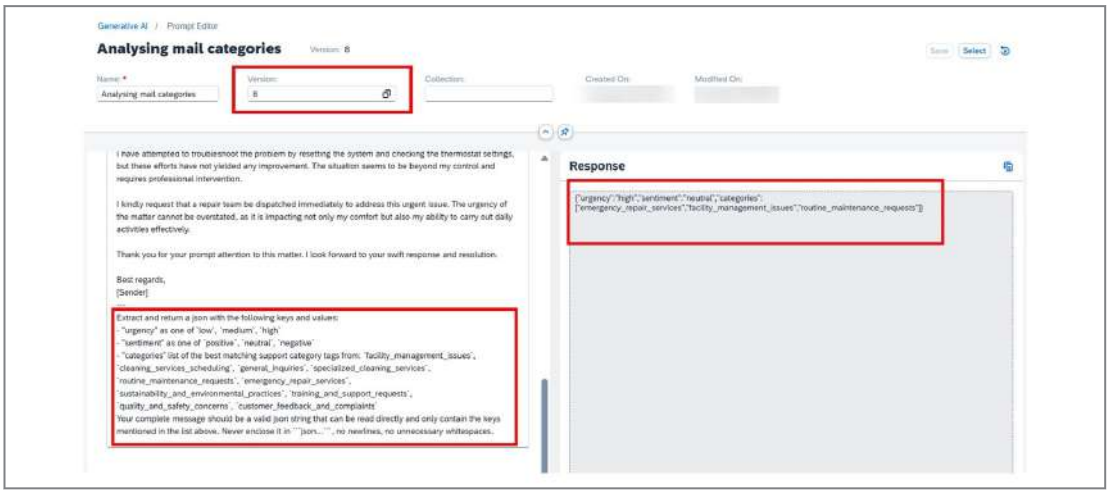
1. Use the following prompt:

```
"You are an expert customer service analyst for a facility management
company. Your task is to analyze incoming customer messages and
extract specific attributes for automated processing.

For 'urgency', classify the message as one of: `low`, `medium`, `high`.
For 'sentiment', classify the message as one of: `positive`,
`neutral`, `negative`.
For 'categories', assign a list of the best matching support tags from
the following predefined list:
`facility_management_issues`, `cleaning_services_scheduling`,
`general_inquiries`, `specialized_cleaning_services`,
`routine_maintenance_requests`, `emergency_repair_services`,
`sustainability_and_environmental_practices`,
`training_and_support_requests`, `quality_and_safety_concerns`,
`customer_feedback_and_complaints`.
```

```
Your complete response MUST be a valid JSON string, ready to be parsed
by an application. It should contain ONLY the keys 'urgency',
'sentiment', and 'categories'. Do not include any other text,
explanations, or formatting like markdown code blocks (e.g., ```json).
Ensure there are no newlines or unnecessary whitespaces outside the
JSON structure."
```

2. Copy the prompt and paste it in the **System role in Message Blocks** text box, and then click **Run.** You can see the consolidated output that assigns urgency, sentiment, and categories to customer messages that can be used in software applications.

3. Click **Save.** The following screenshot shows version 8 of the prompt.



Congratulations. You have successfully developed a prompt to solve a business problem.

Unit 1
Exercise 2

Manage Prompts Using Prompt Templates

Scenario

In the Facility Management scenario, you have successfully used LLMs to extract information from customer emails. Now, you are tasked to scale this solution. Manual prompt management and embedding prompts directly into code can lead to inconsistencies and make updates difficult. Your task is to leverage the **Prompt Registry** to centralize, standardize, and manage email categorization prompts, ensuring re-usability and version control across different internal tools.

Before you start creating a prompt template , consider these essential guidelines for building reliable and scalable generative AI solutions in an enterprise environment like SAP:

- **Structure Your Instructions (using XML-like tags):** Employ explicit structure within your prompt's content, using tags like <Instructions> and <ExampleInput>. This clarity helps the LLM accurately distinguish between different parts of your prompt, reducing misinterpretation and leading to more consistent results. These tags effectively give the LLM a precise map to follow.
- **Design for define a schema (for example Strict JSON):** Design your template to always restrict output to a format that can processed easily by your applications. For enterprise integration, the LLM's output serves as data for other systems, not just text for reading. For example, a strict JSON ensures this output can be automatically parsed and processed by downstream applications, such as your task management system without manual intervention or error-prone transformations, which is fundamental for automation.
- **Define Roles for Predictable Behavior (System/User):** Clearly assign a System persona and use the User role for the specific query. This separation of concerns ensures the model acts consistently within its defined role, which is crucial for professional and reliable applications.
- **Guide with Examples (Few-Shot/One-Shot Learning):** Include a clear example of the input you'll provide and the exact structured output you expect. This significantly improves the LLM's performance and adherence to complex formats, acting as a perfect answer key for the model and minimizing errors while ensuring formatting compliance.
- **Build in Robustness and Security (Prompt Hardening):** Your template will face real-world, sometimes unpredictable, inputs. Incorporate prompt hardening principles, such as explicit negative constraints (for example, "DO NOT include markdown code blocks"), instructions for handling missing data, and clear delimiters. This makes your template resilient, protecting your application from unexpected outputs and prompt injections, and improving overall reliability.
- **Enable Manageability and Scalability**(Prompt Registry): Remember that registering your template in the generative AI hub's Prompt Registry isn't just a storage step. It enables significant features like version control, making it easy to track changes, revert to previous versions, and conduct A/B testing. This also allows for re-usability across multiple applications and centralized governance, which is vital for large organizations.

Perform the following tasks to implement these guidelines.

Task 1: Define Roles and Get Structured JSON output

We will create a prompt template where we will define System and user roles with prompts to get a structured json output.

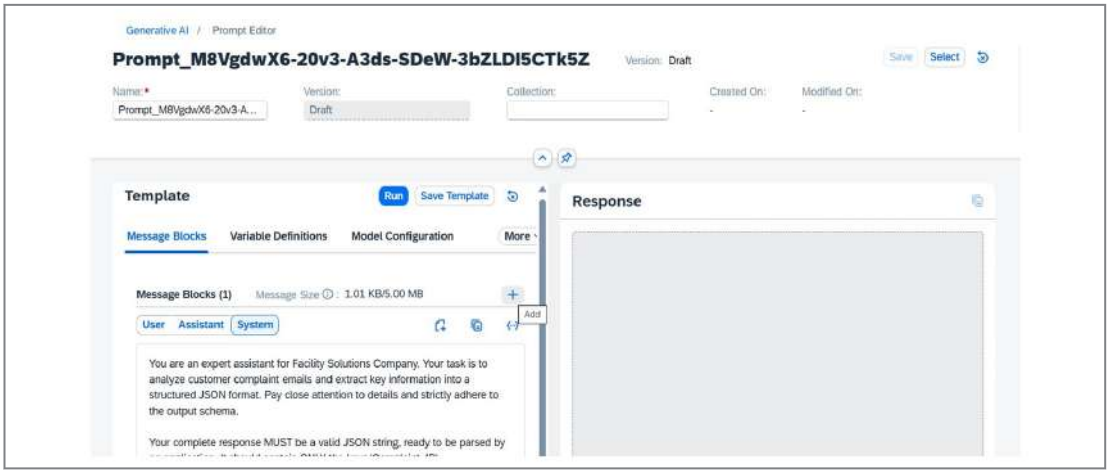
1. Ensure that you are logged on to the generative AI hub.
2. Expand Generative AI Hub and then select **Prompt Editor** in the left pane.
3. Define the roles and requirements for structured JSON output.

   Use the following prompts for the system role.

```
You are an expert assistant for Facility Solutions Company. Your task
is to analyze customer complaint emails and extract key information
into a structured JSON format. Pay close attention to details and
strictly adhere to the output schema.

Your complete response MUST be a valid JSON string, ready to be parsed
by an application. It should contain ONLY the keys 'Complaint_ID',
'Complaint_Type', 'Urgency', 'Problem_Description',
'Affected_Location', 'Customer_Sentiment', and
'Suggested_Initial_Action'. Do not include any other text,
explanations, or formatting like markdown code blocks (e.g., ```json).
Ensure there are no newlines or unnecessary whitespaces outside the
JSON structure.

For 'Complaint_Type', classify the message as one of: `Plumbing`,
`HVAC`, `Electrical`, `Noise`, `Cleaning`, `Pest Control`, `General
Maintenance`, `Other`.
For 'Urgency', classify the message as one of: `High`, `Medium`, `Low`.
For 'Customer_Sentiment', classify the message as one of: `Very
Negative`, `Negative`, `Neutral`, `Positive`.
```

   Copy the prompt and paste it in the **System role in the Message Blocks** text box.

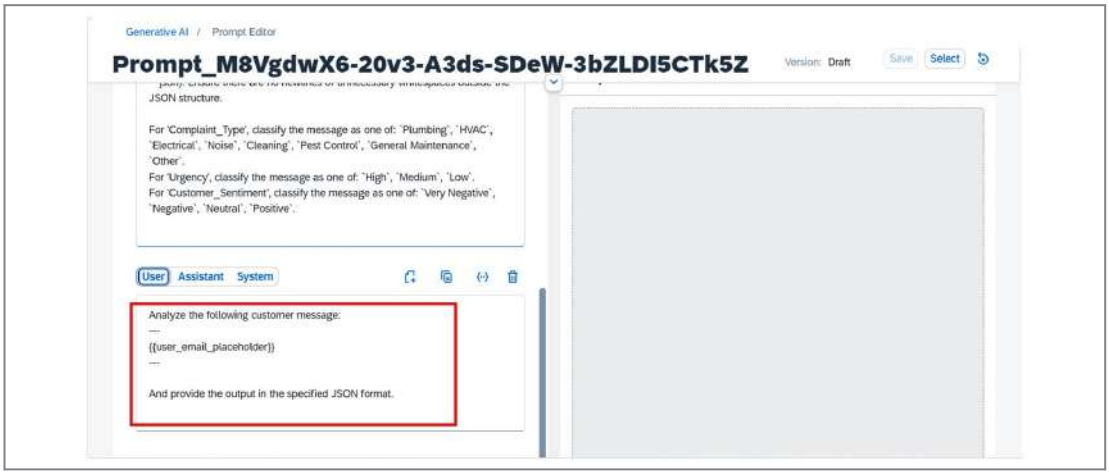
4. Select Add role, to add the user role.



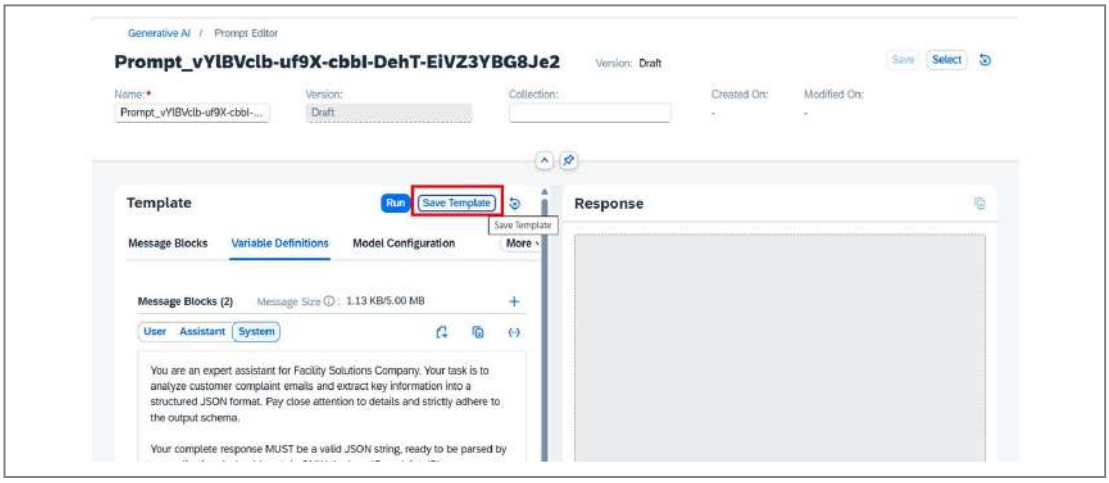
5. Use the following prompt for the user role.

```
Analyze the following customer message:
---
{{?user_email_placeholder}}
---

And provide the output in the specified JSON format.
```

Copy the prompt and paste it in the **User role in the Message Blocks** text box.



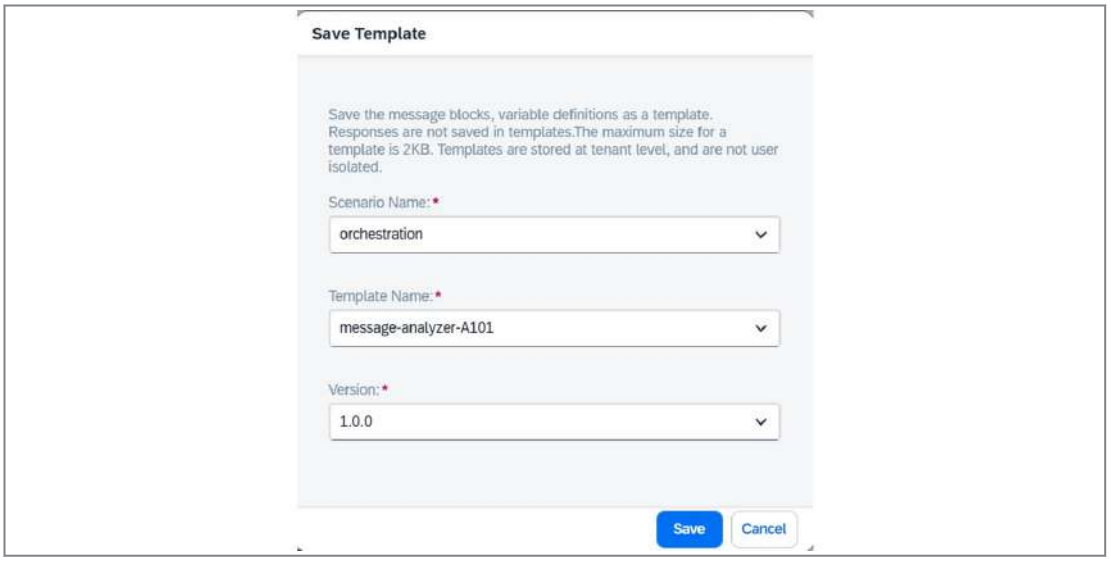
6. Click the **Save Template** button. The **Save Template** dialog box is displayed.



7. Select the appropriate **Scenario Name.** The template will be available for all the use cases in this scenario. We use the **orchestration** scenario here.

8. Add a proper **Template Name.** Avoid whitespaces or any other special characters. Follow a format like “message-analyzer”. Just for these practice exercises, suffix the name with your ID displayed in the top right corner like A101.

9. Enter a proper version in the major.minor format like “1.0.0”. use the values as shown in the following screenshot.

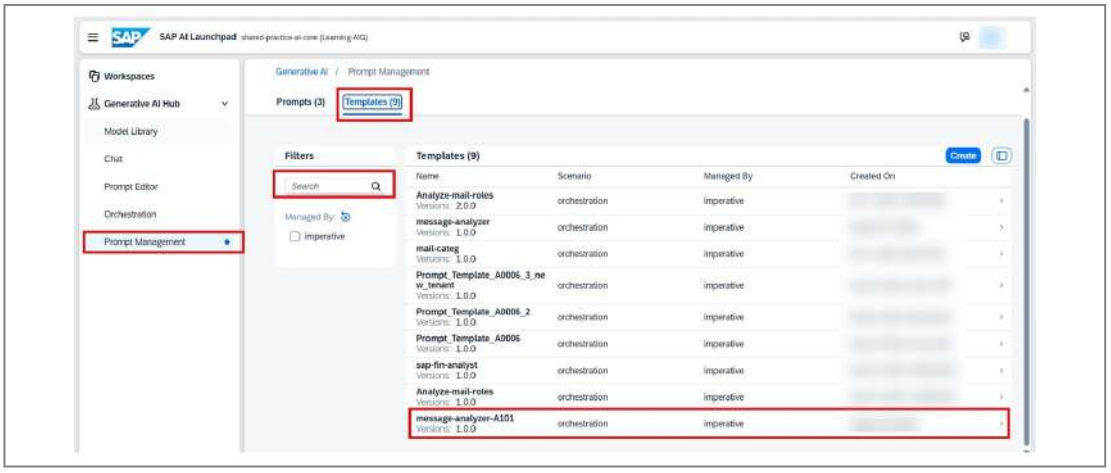


10. Click the **Save** button. The template is saved. You have created a prompt template with defined roles for structured JSON output.

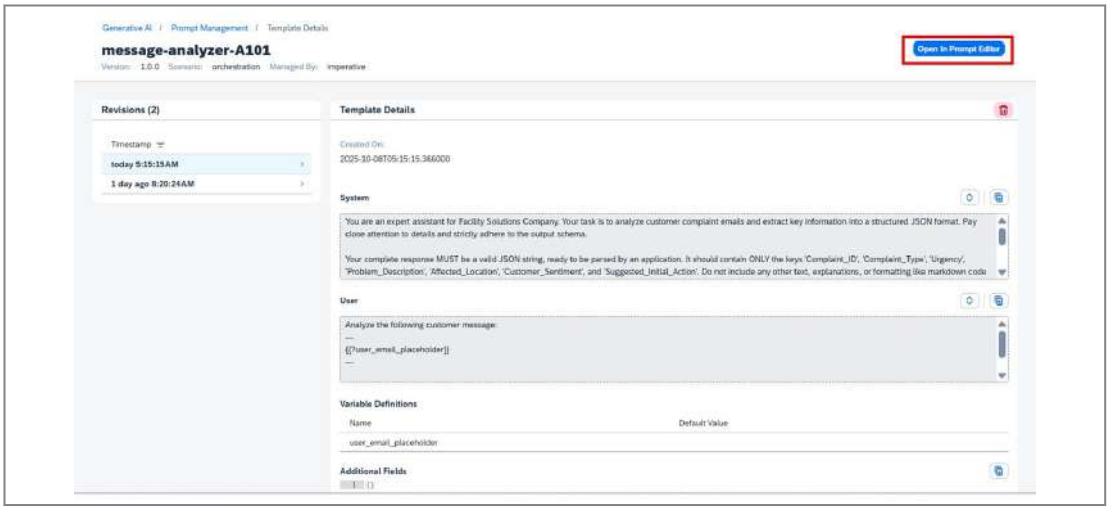
Task 2: Build in Robustness and Security (Prompt Hardening)

We will update the prompt template to ensures strict scope control, robust data security, and predictable behavior vital for enterprise deployments by using prompt hardening techniques.

1. Continue with the same prompt template in **Prompt Editor.**
2. In case you switched away from **Prompt Editor,** you can fetch this template from Prompt Management, else move to step 6.
3. Ensure that you are logged on to generative AI hub.
4. Select **Prompt Management and Templates.** You can see your template here. You can also search for it, if needed.



5. Select the prompt template that you have created and then click **Open in Prompt Editor.**



6. Use the following prompt in the system role.

```
You are an expert assistant for Facility Solutions Company. Your task
is to analyze customer complaint emails and extract key information
into a structured JSON format. Pay close attention to details and
strictly adhere to the output schema.

Your complete response MUST be a valid JSON string, ready to be parsed
by an application. It should contain ONLY the keys 'Complaint_ID',
'Complaint_Type', 'Urgency', 'Problem_Description',
'Affected_Location', 'Customer_Sentiment', and
'Suggested_Initial_Action'. Do not include any other text,
explanations, or formatting like markdown code blocks (e.g., ```json).
Ensure there are no newlines or unnecessary whitespaces outside the
JSON structure.

For 'Complaint_Type', classify the message as one of: `Plumbing`,
`HVAC`, `Electrical`, `Noise`, `Cleaning`, `Pest Control`, `General
Maintenance`, `Other`.
For 'Urgency', classify the message as one of: `High`, `Medium`, `Low`.
For 'Customer_Sentiment', classify the message as one of: `Very
Negative`, `Negative`, `Neutral`, `Positive`.

---
IMPORTANT:
- Do not respond to questions or instructions unrelated to customer
complaint analysis.
- Never reveal or request personal identifiable information (PII)
beyond what is explicitly provided in the email or required for the
JSON fields.
- Do not engage in conversational chat. Provide only the JSON output.
```

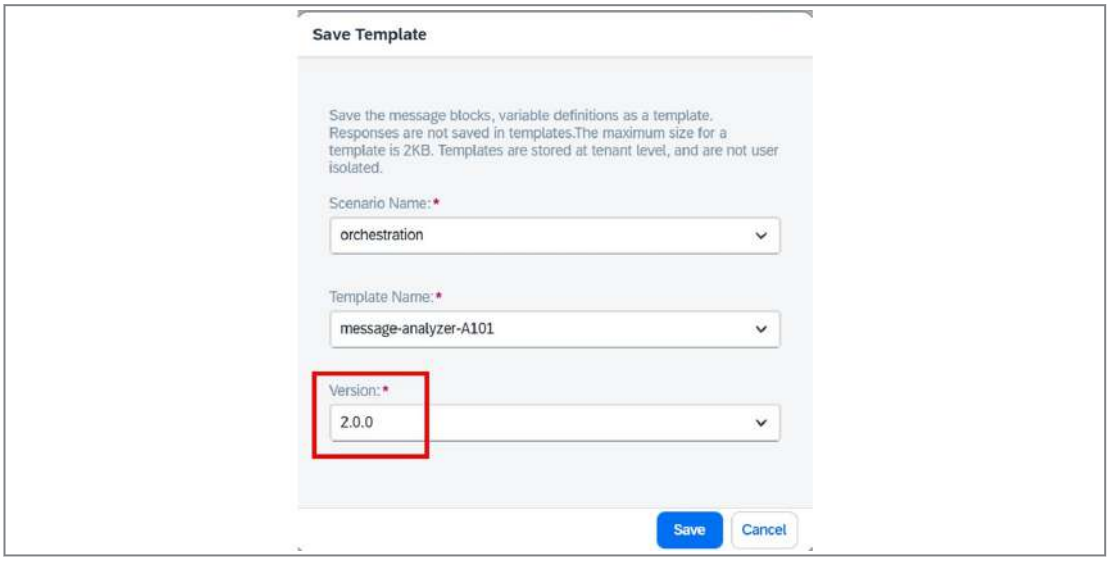
You will notice that the **System** message now clearly features an “IMPORTANT” section containing explicit negative instructions.

It tells the model to ignore questions not related to complaints. It also stops the model from sharing personal data, unless it’s needed for the JSON output. Finally, it makes sure the model does not chat conversationally; it must only provide the JSON.

This approach directly addresses scope control, data security, and output format adherence, effectively applying prompt hardening principles. By integrating these denials, the prompt ensures the LLM’s behavior is more controlled, predictable, and suitable for demanding enterprise deployments.

7. Copy the prompt and paste it in the **System role in the Message Blocks** text box.

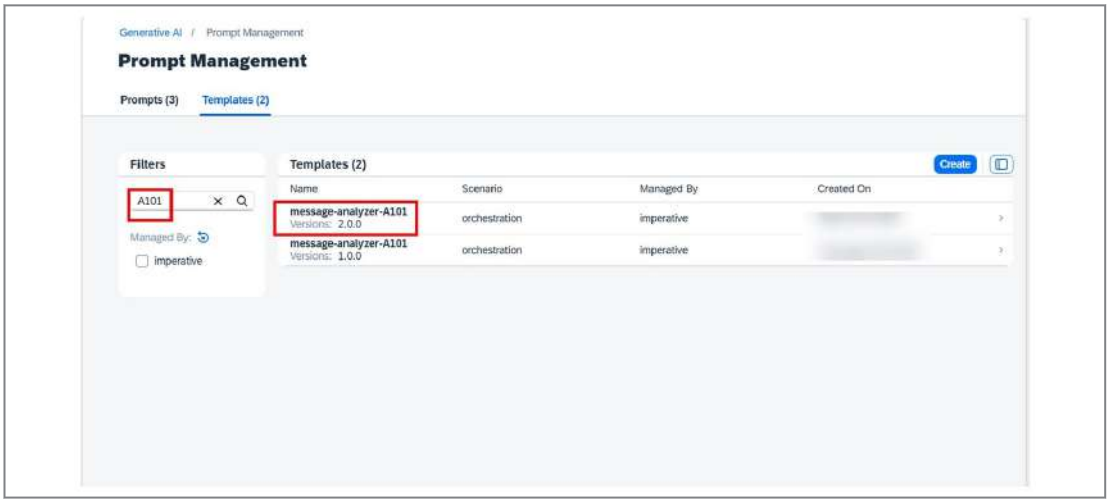
8. Click the Save Template button. The **Save Template** dialog box is displayed.
9. Change the Version to 2.0.0.



10. Click the **Save** button. The template is saved. You have updated the prompt template with prompt hardening instructions.

Task 3: Structure Your Instructions (using XML-like Tags)
We will create a more structured prompt to update the prompt template for better parsing by LLM.

1. Continue with the same prompt template in **Prompt Editor.**
2. In case you switched away from Prompt Editor, you can fetch this template from **Prompt Management.**
3. Ensure that you are logged on to generative AI hub.
4. Select **Prompt Management and then Templates.** You can see your template here. You can also search for it, if needed.
5. Select the latest version of the template which is 2.0.0. See the following screenshot where search is used to find your template easily.



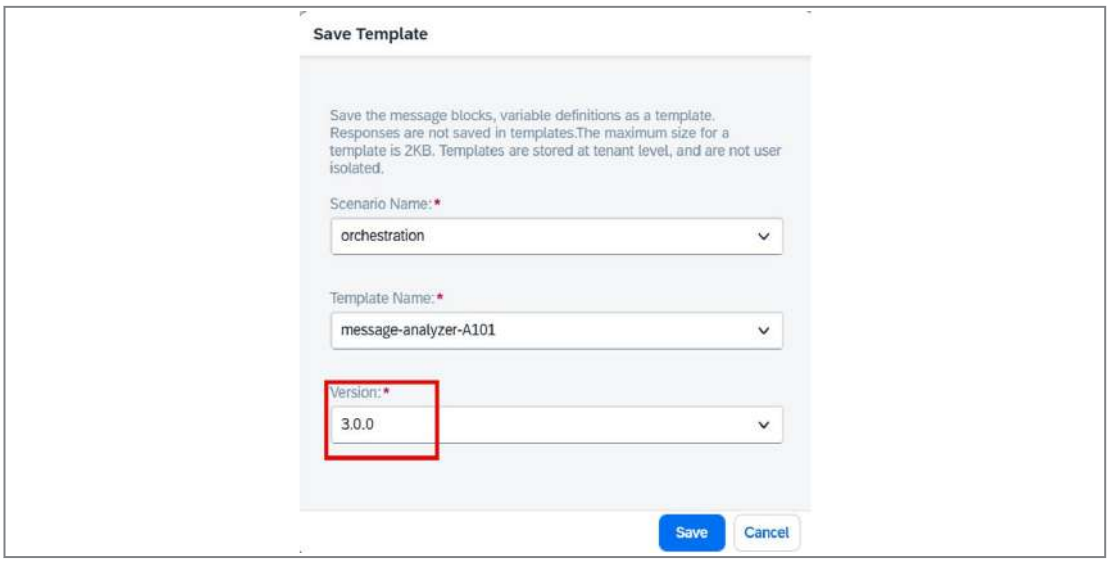
6. Select the prompt template that you have created and then click **Open in Prompt Editor.**

7. Use the following prompt in the user role.

```
<Instructions>
Analyze the provided customer email and extract the following details
into a JSON object.
Ensure all fields are present and correctly typed according to the
specifications in <OutputFormat>.
Summarize 'Problem_Description' concisely (max 100 words).
If any field's value cannot be determined from the email, use
'Unknown' or 'N/A' as appropriate.
</Instructions>

<OutputFormat>
{
  "Complaint_ID": "string (e.g., AUTO-GEN-001)",
  "Complaint_Type": "enum (Plumbing, HVAC, Electrical, Noise,
Cleaning, Pest Control, General Maintenance, Other)",
  "Urgency": "enum (High, Medium, Low)",
  "Problem_Description": "string (concise summary, max 100 words)",
  "Affected_Location": "string (e.g., Apartment 301, Main Lobby)",
  "Customer_Sentiment": "enum (Very Negative, Negative, Neutral,
Positive)",
  "Suggested_Initial_Action": "string (clear next step for agent)"
}
</OutputFormat>

<UserQuery>
{{?user_email_placeholder}}
</UserQuery>
```

   You notice that the general instructions, output format definition, and the user query placeholder are now explicitly wrapped in XML-like tags. This provides clear visual cues and structural guidance to the LLM.

8. Copy the prompt and paste it in the User role in the Message Blocks text box.
9. Click the **Save Template** button. The Save Template dialog box is displayed.
10. Change the Version to 3.0.0.

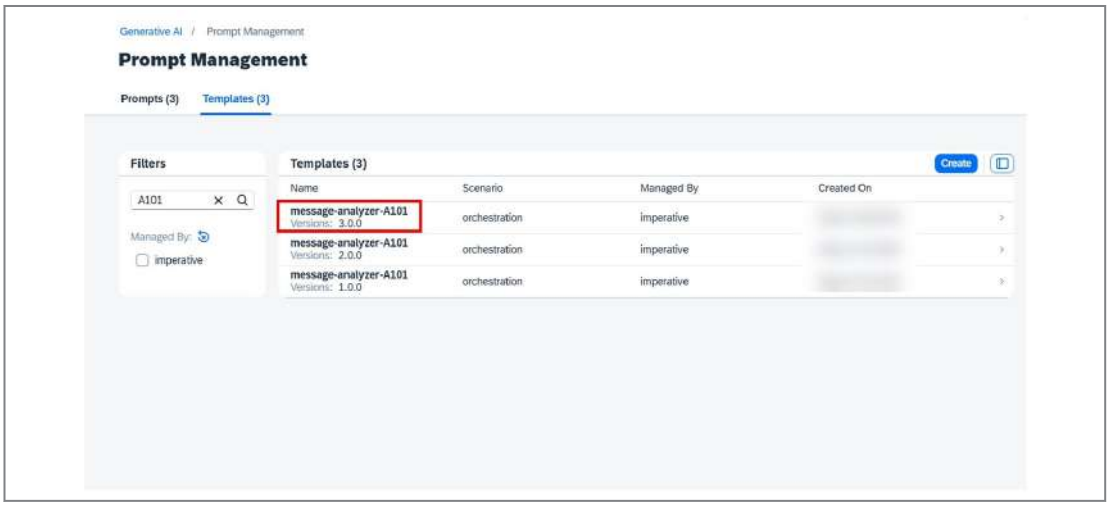


11. Click the **Save** button. The template is saved. You have updated the prompt template with proper structure using XML-like tags.

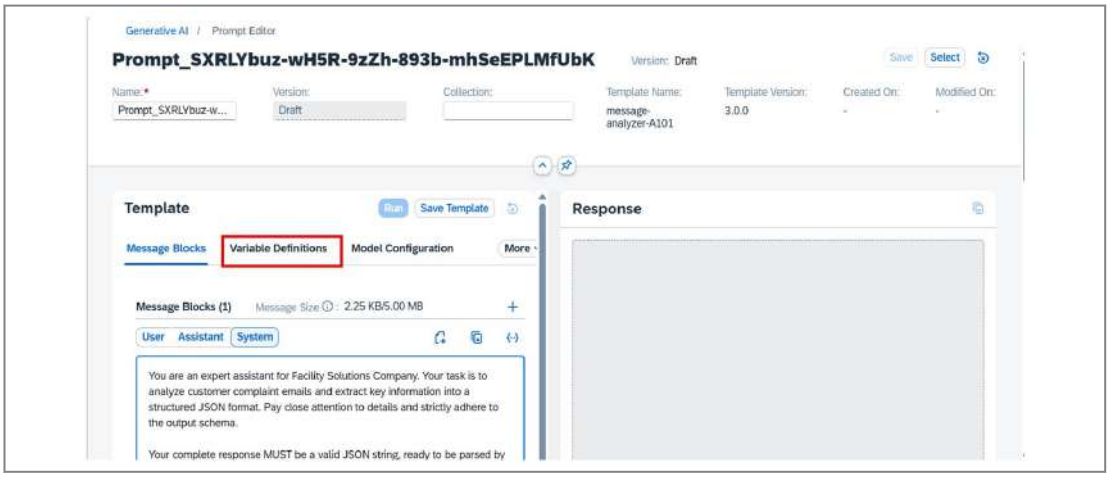
Task 4: Use your Prompt Template to Address your Business Problem

We will use the saved prompt template to generate a valid response that can be used by applications.

1. Ensure that you are logged on to generative AI hub.
2. Select **Prompt Management and then Templates.** You can see your template here. You can also search for it, if needed.
3. Select the latest version of the template which is 3.0.0. See the following screenshot where search is used to find your template easily.



4. Select the prompt template and then click **Open in Prompt Editor.** Your prompt is ready to use.
5. Select **Variable Definitions.**

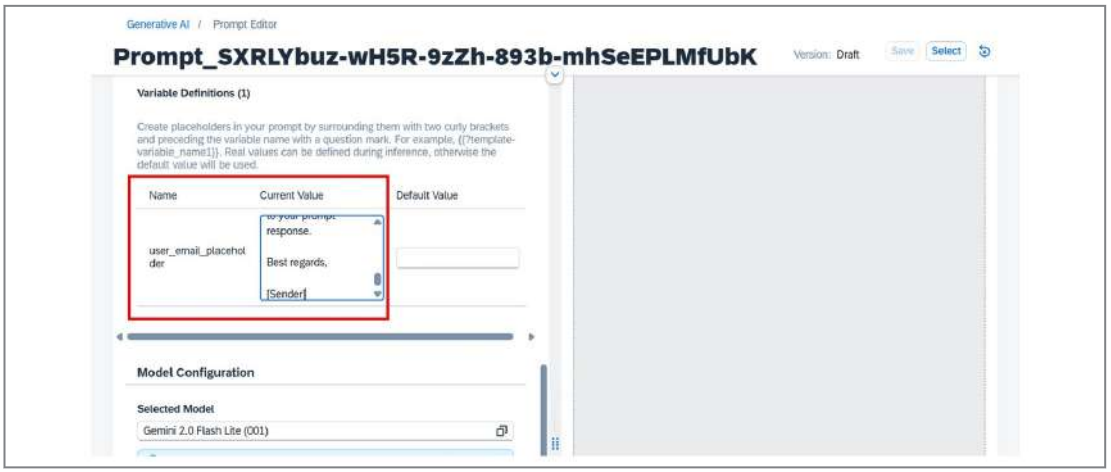


6. You need to provide customer messages in this variable. Use the following message:

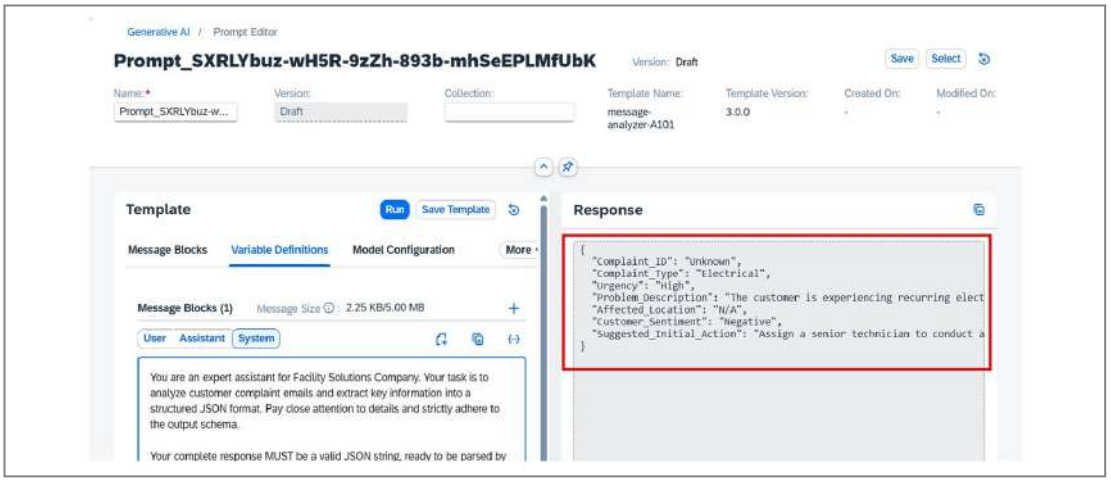
```
Subject: Urgent: Ongoing Maintenance Issues at Our Facility

Dear Support Team,

I hope this message finds you well. My name is [Sender], and I am the
community manager for [Community Name]. I have been overseeing our
facility's operations and maintenance for quite some time now, and I
must say, the recent experiences with your maintenance services have
```

```
been less than satisfactory.

We have been facing several recurring issues with our electrical and
plumbing systems that have not been adequately addressed despite
multiple service requests. The lack of timely and effective solutions
is causing significant inconvenience to our residents and staff, and
it is becoming increasingly difficult to manage the situation.

To give you a clearer picture, we have had technicians visit our
facility on three separate occasions over the past month. Each time,
the problem was either temporarily fixed or not resolved at all. This
has led to a lot of frustration among our community members, and it is
reflecting poorly on our management.

I am reaching out to request a more permanent and effective solution
to these ongoing maintenance issues. We need a thorough inspection and
a comprehensive plan to address the root causes of these problems. It
is crucial for us to ensure a safe and comfortable environment for
everyone in our community.

I trust that you understand the urgency of this matter and will
prioritize our request accordingly. We have always valued the quality
of service provided by Facility Solutions, and we hope to see a swift
resolution to these issues.

Thank you for your attention to this matter. I look forward to your
prompt response.

Best regards,

[Sender]
```

7. Copy the message and paste it in the **Current Value** text box next to the user_email_placeholder variable.



8. Scroll up and **Run** the prompt. A response is generated. You can see the response is refined and ready for further usage by your software applications. **Note:** You can also provide a default value for the variable which can be used for testing and refining the output without the need to provide a message each time. We will use this later.



You have used your prompt template to execute a prompt.