

HA300

SAP HANA 2.0 SPS06 - Modeling

EXERCISES AND SOLUTIONS

Course Version: 18
Course Duration: 14 Hours 5 Minutes
Material Number: 50155963

SAP Copyrights, Trademarks and Disclaimers

© 2022 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. Please see <https://www.sap.com/corporate/en/legal/copyright.html> for additional trademark information and notices.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors.

National product specifications may vary.

These materials may have been machine translated and may contain grammatical errors or inaccuracies.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP SE or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP SE or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.

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:	Calculation Views
1	Exercise 1: Prepare your SAP HANA System
14	Exercise 2: Create CUBE and SQL ACCESS ONLY Calculation Views
Unit 2:	Using Nodes in Calculation Views
23	Exercise 3: Create a DIMENSION Calculation View
35	Exercise 4: Join Three Tables in a Single Join Node
42	Exercise 5: Combine Two Data Sources with a Union Node
53	Exercise 6: Use a Minus Node in a Calculation View
59	Exercise 7: Control the Behavior of the Aggregation Node
69	Exercise 8: Create a CUBE with Star Join Calculation View
79	Exercise 9: Use Rank Nodes to Partition, Order, and Extract Values from a Data Set
Unit 3:	Modeling Functions
100	Exercise 10: Create Restricted and Calculated Columns
112	Exercise 11: Define and Use Filters
122	Exercise 12: Define Filtering on SAP Client
130	Exercise 13: Implement Variables
135	Exercise 14: Implement Input Parameters
144	Exercise 15: Implement Value Help Views
149	Exercise 16: Cascade User Prompts
157	Exercise 17: Create a Level Hierarchy
165	Exercise 18: Create a Parent-Child Hierarchy
173	Exercise 19: Implement a Hierarchical Value Help
176	Exercise 20: Implement Currency Conversion in a Calculation View
188	Exercise 21: Define a Time Dimension Calculation View
Unit 4:	Using SQL in Models
195	Exercise 22: Work with the SQL Console
198	Exercise 23: Read a Modeled Hierarchy using SQL
203	Exercise 24: Work with SQLScript
206	Exercise 25: Derive an Input Parameter Value using a Scalar Function
211	Exercise 26: Define a Data Source using a Table Function
Unit 5:	Persistence Layer
219	Exercise 27: Create and Delete Tables Using the SQL Console
225	Exercise 28: Create Tables Using Source Files
227	Exercise 29: Load a Table Using the SQL Console
230	Exercise 30: Load a Table Using Source Files

Unit 6: Optimization of Models

233	Exercise 31: Implement Static Cache
237	Exercise 32: Control Parallelization in a Data Flow
243	Exercise 33: Implement Union Pruning in a Calculation View
252	Exercise 34: Access a Calculation View in Performance Analysis Mode
255	Exercise 35: Use Debug Query Mode

Unit 7: Management and Administration of Models

258	Exercise 36: Audit Calculation Views and their Dependencies
261	Exercise 37: Import, Rename and Copy Models
272	Exercise 38: Create a New Project and an HDB Module
277	Exercise 39: Set Up a Project to Access an External Schema
286	Exercise 40: Use Git for Version Control in Your Local Workspace
297	Exercise 41: Clone the SAP HANA Interactive Education (SHINE) Application from GitHub

Unit 8: Security in SAP HANA Modeling

301	Exercise 42: Create and Assign an Analytic Privilege
314	Exercise 43: Define Column Masking in a Calculation View

Unit 1

Exercise 1

Prepare your SAP HANA System

Exercise Objectives

After completing this exercise, you will be able to:

- Log on to the training landscape
- Start and configure the SAP Web IDE for SAP HANA
- Prepare your system for modeling activities

Business Example

You are at a customer site and want to start modeling in the SAP Web IDE for SAP HANA. The purpose of this exercise is to set up the training environment so that all future exercises in this course can be executed. Some exercises require ready-made files. You will copy these files to your local work area by running a prepared script so they become available to you. You will also set up a project for modeling and connect to database resources.

Overview of Exercise Tasks

- Task 1: Connect to the Training System and Copy Course Files
- Task 2: Log On to the SAP Web IDE for SAP HANA and Import an Existing XSA Project
- Task 3: Add your HDI Container to the Database Explorer
- Task 4: Add the Classical SAP HANA Database to the Database Explorer
- Task 5: Enable External Access to Your Container Models



Note:

In this exercise, when the values include ##, replace ## with the group numbers that your instructor assigned you.

For SAP Learning System Access users, the group number is displayed in the top area of your system's window.

Task 1: Connect to the Training System and Copy Course Files

1. Access the assigned SAP Cloud Training System.



Caution:

If you are an **SAP Learning System Access** customer, follow the provided SAP Learning System Access instructions. You can skip this step and continue with step 2.

2. Initialize your HA300 - SAP Cloud Training System session by running the *Initialize_HANA* script found in the *Initialize Course* folder.
The *Initialize Course* folder is available in the WindowsStart page.
3. Add the course files folder N:\HA300 to your Quick Access shortcuts so that it is easy to reach in future.
4. Check that the course resources are present.

Task 2: Log On to the SAP Web IDE for SAP HANA and Import an Existing XS Advanced Project

1. Launch the SAP Web IDE for SAP HANA.
A shortcut is provided in the folder Quick Access → HA300 → URLs.
2. Enable Google Chrome to save your password.
This makes re-connection to the SAP Web IDE for SAP HANA after time-out easier.



Note:

You do not need to log on to any Google account for that. Passwords will just be saved locally.

3. Log on using the details shown in the table, Logon Details. Allow Google Chrome to save the password.

Table 1: Logon Details

Field	Value
User	STUDENT##
Password	Training1

4. In the *Development* perspective, import an existing XS Advanced project into your workspace.
The project archive, *HA300.zip*, is located in your *HA300* folder, and must be imported to your workspace with the project name **HA300_##**.

5. Review the structure of your *HA300_##* project.

The project is made up of one SAP HANA Database (HDB) module that will contain all your modeling content, including synonyms that are defined to access external data from a few classical SAP HANA schemas in the same H00 database. Within the *HDB → src* folder, the design-time objects will be organized in different subfolders, as follows:

- **config**

Contains the definition of synonyms to access external data (outside of your container schema) and the corresponding authorizations.

- **exercises**

Will be used to store all the modeling artifacts that you will create during the training.

- **resources**

A few ready-made objects that you will reuse in some of your models.

- **solutions**

This folder contains the “solution” objects, that is, 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.

6. Activate the following display options in the SAP Web IDE for SAP HANA:

- Link workspace to Editor
- Show Hidden Files

7. Assign your project to the *DEV* space and check the status of the DI Builder. The status should read "A builder is properly installed in this space".



Caution:

If the status shows something else than "A builder is properly installed in this space", it is likely that this is because the DI Builder timed-out. Please **refer to your instructor** so that he/she restarts the builder, as this only needs to be done by one user. Please do **not** reinstall the builder on your own because this would overload the system.

For **SAP Live Access users**: You can get the builder up and running again by choosing *Reinstall Builder* after assigning the project to the *DEV* space. The console log should report a successful deployment after a few minutes, and the builder status in the Space properties of the project should show in green.

8. Build the *HDB* module of your project so that the database container can be generated.



Caution:

Make sure you execute the build at the *HDB* module level.

If you trigger the build at the project level, the *HDB* module will NOT be built (building a project is what prepares the project archives for deployment, which is not what we want to do for now).

Task 3: Add your HDI Container to the Database Explorer

Once you've built your project successfully, an SAP HANA Deployment Infrastructure (HDI) container is created, containing all the runtime versions of your artefacts. Adding the HDI container to the Database Explorer allows you to view these runtime objects.

1. Switch to the *Database Explorer* perspective.

2. Add your HDI container. This container's name starts with *STUDENT##* and also contains the project name.

Give the following display name to the container: **MY HA300 CONTAINER**.



Hint:

- You can use the search feature and search for **HA300_##**.
- If the container does not show up in the list, choose *Refresh*.

Your HDI Container is now added to the Database Explorer as **MY HA300 CONTAINER**.

Task 4: Add the Classical SAP HANA Database to the Database Explorer

The main source tables you will use during the modeling activities are stored in a couple of classical SAP HANA database schemas, and accessed via synonyms that are already defined in the project you have imported and built earlier in this exercise. To display these source tables easily, you will now add the classical SAP HANA database to the Database Explorer.

1. In the *Database Explorer* perspective, add the tenant database *H00*, using the following information:

Table 2: Database Information

Field	Value
Database Type	SAP HANA Database (Multitenant)
Host	wdf1bmt7215.wdf.sap.corp
Instance number	00
Database name	H00 (Tenant Database)
User	STUDENT## (where ## is your group number)
Password	Training1
Save User and Password	[Selected]
Name to Show in Display	H00 DB (CLASSIC)



Note:

Your training system is configured in *MultiTenant* mode.



Hint:

If the container does not show up in the list, choose *Refresh*.

The catalog of the SAP HANA Database *H00* can now be explored from the Database Explorer.

Task 5: Enable External Access to Your Container Models

A generic role for enabling Data Access to your container's content was created when you built your *HA300_##* application for the first time. This role must be granted to the *TRAINING_ROLE_##*, which is specific to your *STUDENT##* user.

1. In the Database Explorer, open an SQL console connected to the *H00 DB* database (not your own container).



Caution:

Make sure that you do NOT open the console for your container. If you do so, the SQL query will not be authorized.

2. In the SQL console, import the following file containing prepared SQL statements:

HA300 → Prepared Code → SQL for External Container Access.sql.

```
-- 1.  
SELECT * FROM ROLES WHERE ROLE_NAME LIKE '%HA300_##%access_role';  
-- 2.  
GRANT "HA300_##_HDI_HDB_1::access_role" to TRAINING_ROLE##;
```

3. Execute the SQL statements to grant your course participant role the authorization on your container in order to consume the container's data from external tools. You must first replace *##* with your group number.



Note:

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

- Select the statement and 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. Close all open tabs in the SAP Web IDE for SAP HANA.

Unit 1

Solution 1

Prepare your SAP HANA System

Exercise Objectives

After completing this exercise, you will be able to:

- Log on to the training landscape
- Start and configure the SAP Web IDE for SAP HANA
- Prepare your system for modeling activities

Business Example

You are at a customer site and want to start modeling in the SAP Web IDE for SAP HANA. The purpose of this exercise is to set up the training environment so that all future exercises in this course can be executed. Some exercises require ready-made files. You will copy these files to your local work area by running a prepared script so they become available to you. You will also set up a project for modeling and connect to database resources.

Overview of Exercise Tasks

- Task 1: Connect to the Training System and Copy Course Files
- Task 2: Log On to the SAP Web IDE for SAP HANA and Import an Existing XSA Project
- Task 3: Add your HDI Container to the Database Explorer
- Task 4: Add the Classical SAP HANA Database to the Database Explorer
- Task 5: Enable External Access to Your Container Models



Note:

In this exercise, when the values include ##, replace ## with the group numbers that your instructor assigned you.

For SAP Learning System Access users, the group number is displayed in the top area of your system's window.

Task 1: Connect to the Training System and Copy Course Files

1. Access the assigned SAP Cloud Training System.



Caution:

If you are an **SAP Learning System Access** customer, follow the provided SAP Learning System Access instructions. You can skip this step and continue with step 2.

- a) Connect to the assigned SAP Cloud Training System as explained by your instructor.

2. Initialize your HA300 - SAP Cloud Training System session by running the *Initialize_HANA* script found in the *Initialize Course* folder.

The *Initialize Course* folder is available in the WindowsStart page.

- In the Remote Desktop environment, choose *Start*.

- Scroll to locate, then click *Initialize Course*.

- Select *Initialize_HANA*.

- In the *Browse for Folder* dialog box, highlight the *HANA → → HANA_18 → → HA300* folder (do not go any lower) and choose *OK*.

- At the next prompt, choose *OK*.

You have initialized the course landscape, and all the required course files are copied to the *N:\HA300* folder. Have fun with HA300.

3. Add the course files folder *N:\HA300* to your *Quick Access* shortcuts so that it is easy to reach in future.

- In the Windows Explorer window that opened at the previous step, drag the folder *HA300* to the *Quick Access* area on the left. Before releasing the mouse, make sure the proposed action is *Pin to Quick Access*.
- Alternatively, in Windows explorer, double-click the *HA300* folder in the right pane to open it. Then, in the left pane, right-click the *Quick Access* icon and choose *Pin Current Folder to Quick Access*.

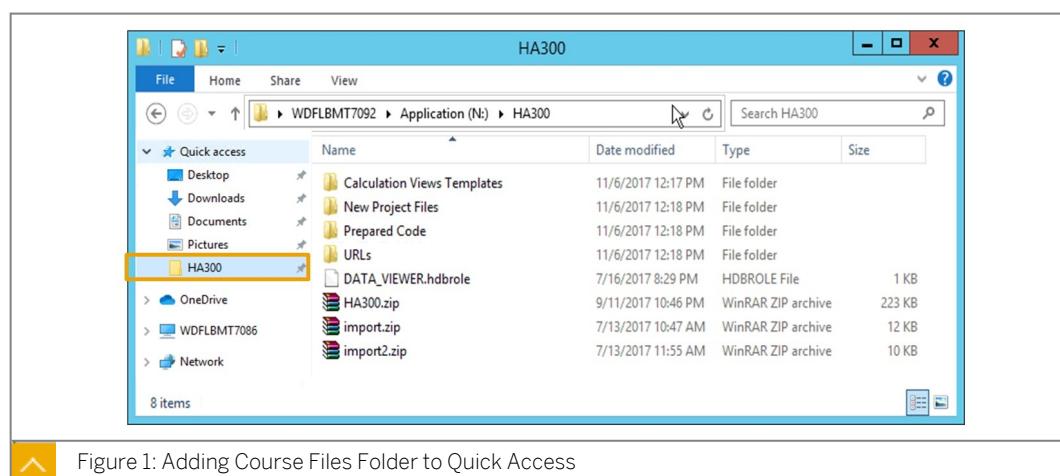


Figure 1: Adding Course Files Folder to Quick Access

4. Check that the course resources are present.

- In *Quick Access* area, select the *HA300* folder and ensure that various ready-made files and shortcuts are present.

Task 2: Log On to the SAP Web IDE for SAP HANA and Import an Existing XS Advanced Project

- Launch the SAP Web IDE for SAP HANA.

A shortcut is provided in the folder *Quick Access → HA300 → URLs*.

- Double-click the shortcut *Web IDE for SAP HANA* in the folder *Quick Access → HA300 → URLs*.

- Enable Google Chrome to save your password.

This makes re-connection to the SAP Web IDE for SAP HANA after time-out easier.



Note:

You do not need to log on to any Google account for that. Passwords will just be saved locally.

- a) At the right of the address bar, click the "3-dots" menu icon and choose *Settings*.
 - b) In the left *Settings* pane, choose *Autofill*.
 - c) Choose *Passwords* and turn on the *Offer to save password* feature.
 - d) Close the *Settings - Password* tab.
3. Log on using the details shown in the table, Logon Details. Allow Google Chrome to save the password.

Table 1: Logon Details

Field	Value
User	STUDENT##
Password	Training1

- a) On the logon screen, enter the details shown in the table, Logon Details, and choose *Log On*.
 - b) In the Google Chrome dialog box proposing to save your password, choose *Save*.
 - c) In the *Tips and Tricks* dialog box, select the *Don't show on startup* checkbox and choose *Close*.
4. In the *Development* perspective, import an existing XS Advanced project into your workspace.
- The project archive, *HA300.zip*, is located in your *HA300* folder, and must be imported to your workspace with the project name **HA300_##**.
- a) In the *Development* perspective, right-click the *Workspace* folder and choose *Import → File or Project*.
 - b) Choose *Browse* and navigate to the Windows folder *Quick Access → HA300*.
 - c) Select the file *HA300.zip* and choose *Open*.
 - d) In the *Import to* field, enter */HA300_##* (## is your group number).
 - e) Make sure the *Extract Archive* checkbox is selected.
 - f) Choose *OK* to begin the import.
5. Review the structure of your *HA300_##* project.
- The project is made up of one SAP HANA Database (HDB) module that will contain all your modeling content, including synonyms that are defined to access external data from a few classical SAP HANA schemas in the same H00 database. Within the *HDB → src* folder, the design-time objects will be organized in different subfolders, as follows:
- config

Contains the definition of synonyms to access external data (outside of your container schema) and the corresponding authorizations.

- **exercises**

Will be used to store all the modeling artifacts that you will create during the training.

- **resources**

A few ready-made objects that you will reuse in some of your models.

- **solutions**

This folder contains the “solution” objects, that is, 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.

6. Activate the following display options in the SAP Web IDE for SAP HANA:

- Link workspace to Editor
- Show Hidden Files

- a) Choose the corresponding options in the *View* menu of the SAP Web IDE for SAP HANA.

7. Assign your project to the *DEV* space and check the status of the DI Builder. The status should read "A builder is properly installed in this space".

- a) In the workspace, right-click your HA300_## project and choose *Project → Project Settings*.
- b) Open the *Space* panel and choose *DEV* from the dropdown list.
- c) Observe the status of the DI Builder in the *DEV* space.
- d) Choose *Save*.
- e) Choose *Close*.

**Caution:**

If the status shows something else than "A builder is properly installed in this space", it is likely that this is because the DI Builder timed-out. Please **refer to your instructor** so that he/she restarts the builder, as this only needs to be done by one user. Please do **not** reinstall the builder on your own because this would overload the system.

For SAP Live Access users: You can get the builder up and running again by choosing *Reinstall Builder* after assigning the project to the *DEV* space. The console log should report a successful deployment after a few minutes, and the builder status in the Space properties of the project should show in green.

8. Build the *HDB* module of your project so that the database container can be generated.

**Caution:**

Make sure you execute the build at the *HDB* module level.

If you trigger the build at the project level, the *HDB* module will NOT be built (building a project is what prepares the project archives for deployment, which is not what we want to do for now).

- a) In your *HA300_##* project, right-click the *HDB* folder and choose *Build → Build*.
- b) Wait a few moments, then check that the log shows a successful build.

Task 3: Add your HDI Container to the Database Explorer

Once you've built your project successfully, an SAP HANA Deployment Infrastructure (HDI) container is created, containing all the runtime versions of your artefacts. Adding the HDI container to the Database Explorer allows you to view these runtime objects.

1. Switch to the *Database Explorer* perspective.
 - a) Choose *Tools → Database Explorer* or click the *Database Explorer* icon on the left toolbar.
 - b) If a dialog box says It looks like you have not added any database to the Database Explorer yet. Do you want to add one now?, choose Yes.
2. Add your HDI container. This container's name starts with *STUDENT##* and also contains the project name.

Give the following display name to the container: **MY HA300 CONTAINER**.

**Hint:**

- You can use the search feature and search for **HA300_##**.
- If the container does not show up in the list, choose *Refresh*.

- a) Click the + (*Add a database to the database explorer*) icon.
(This is not needed if you answered yes at the previous step).

- b) In the *Database Type* dropdown list, choose *HDI Container*.
- c) In the *Search* field, enter **HA300_##**.
- d) Select the *STUDENT##-.....-HA300_##-hdi_HDB* entry.
- e) In the *Name to Show in Display* field, enter **MY HA300 CONTAINER** and choose **OK**.
- f) Choose the *Refresh* button.

Your HDI Container is now added to the Database Explorer as *MY HA300 CONTAINER*.

Task 4: Add the Classical SAP HANA Database to the Database Explorer

The main source tables you will use during the modeling activities are stored in a couple of classical SAP HANA database schemas, and accessed via synonyms that are already defined in the project you have imported and built earlier in this exercise. To display these source tables easily, you will now add the classical SAP HANA database to the Database Explorer.

1. In the *Database Explorer* perspective, add the tenant database *H00*, using the following information:

Table 2: Database Information

Field	Value
Database Type	SAP HANA Database (Multitenant)
Host	wdf1bmt7215.wdf.sap.corp
Instance number	00
Database name	H00 (Tenant Database)
User	STUDENT## (where ## is your group number)
Password	Training1
Save User and Password	[Selected]
Name to Show in Display	H00 DB (CLASSIC)



Note:

Your training system is configured in *MultiTenant* mode.



Hint:

If the container does not show up in the list, choose *Refresh*.

- a) Click the + (*Add a database to the database explorer*) icon.
- b) In the *Database Type* dropdown list, choose *SAP HANA Database (Multitenant)*.
- c) Enter the information shown in the table, Database Information.
- d) Choose **OK**.

- e) If needed, choose the *Refresh* button.

The catalog of the SAP HANA Database *H00* can now be explored from the Database Explorer.

Task 5: Enable External Access to Your Container Models

A generic role for enabling Data Access to your container's content was created when you built your *HA300_##* application for the first time. This role must be granted to the *TRAINING_ROLE_##*, which is specific to your *STUDENT##* user.

1. In the Database Explorer, open an SQL console connected to the *H00 DB* database (not your own container).



Caution:

Make sure that you do NOT open the console for your container. If you do so, the SQL query will not be authorized.

- a) If needed, choose *Tools* → *Database Explorer*.
- b) In the Database/Containers list on the left, select the *H00 DB* entry.
- c) Click the *Open SQL Console* icon.
- d) Alternatively, you can right-click the *H00 DB* entry and choose *Open SQL Console*.

2. In the SQL console, import the following file containing prepared SQL statements:

HA300 → *Prepared Code* → *SQL for External Container Access.sql*.

```
-- 1.  
SELECT * FROM ROLES WHERE ROLE_NAME LIKE '%HA300_##%access_role';  
-- 2.  
GRANT "HA300_##_HDI_HDB_1::access_role" to TRAINING_ROLE##;
```

- a) Choose *Import File*.
- b) Select the file specified above and choose *Open*.
3. Execute the SQL statements to grant your course participant role the authorization on your container in order to consume the container's data from external tools. You must first replace *##* with your group number.



Note:

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

- Select the statement and 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) To replace *##* with your group number everywhere in the SQL statements, press **ctrl +H**.

- b)** Check the actual name of your HA300_## container access role (SQL statement in section #1).

```
SELECT * FROM ROLES WHERE ROLE_NAME LIKE '%HA300_##%access_role';
```

- c)** Execute the SQL statements in section #2.

```
GRANT "HA300_##_HDI_HDB_1::access_role" to TRAINING_ROLE##;
```

- 4.** Close all open tabs in the SAP Web IDE for SAP HANA.

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

Unit 1

Exercise 2

Create CUBE and SQL ACCESS ONLY Calculation Views

Exercise Objectives

After completing this exercise, you will be able to:

- Create a calculation view of type CUBE
- Change the type of a calculation view

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).

Overview of Exercise Tasks

- Task 1: Create a Simple CUBE Calculation View
- Task 2: Create a Calculation View of Type SQL Access Only

Task 1: Create a Simple CUBE Calculation View

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

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: HA300 → URLs.
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 following data source:
Table SNWD_SO (schema EPM_MODEL).



Note:

This table is located in an external schema (not in the container of your application), and is accessed via a synonym HA300::SNWD_SO that has already been defined in the files you imported during the first exercise.

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. Save, and then build the *CVC_SALES_SIMPLE* calculation view.
Check the build status.
12. Preview the data of the *CVC_SALES_SIMPLE* calculation view.

Task 2: Create a Calculation View of Type SQL ACCESS ONLY

You have been asked to create an additional calculation view, similar to the first one, but not meant to be exposed to the end users. Instead, it is planned to reuse this view only as a data source for other calculation views.

1. Which type of view should you use for this purpose?

2. In your *Exercises* folder, copy the *CVC_SALES_SIMPLE* calculation view as ***CVSQL_SALES_SIMPLE***.



Hint:

To rename the new copy more easily (in particular the runtime object name), first keep the proposed copy name, and then use the *Rename* feature.

3. In the *Semantics* node, change the data category to **SQL ACCESS ONLY**.
4. Save, and then build the *CVSQL_SALES_SIMPLE* calculation view.
Check the build status.
5. Check which of the two calculation views are exposed in an external tool such as Microsoft Excel, using the menu *Data → Existing Connections*.

Use the following log-on info:

Table 3: SAP HANA Log-on Information

Field	Value
Host	wdflbmt7215.wdf.sap.corp
Instance number	00
Database Mode	<i>Multi Database</i>
Database	<i>User Database: H00</i>
User	STUDENT##
Password	Training1
Language	EN

6. Close all the open tabs in the SAP Web IDE for SAP HANA.

Unit 1

Solution 2

Create CUBE and SQL ACCESS ONLY Calculation Views

Exercise Objectives

After completing this exercise, you will be able to:

- Create a calculation view of type CUBE
- Change the type of a calculation view

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).

Overview of Exercise Tasks

- Task 1: Create a Simple CUBE Calculation View
- Task 2: Create a Calculation View of Type SQL Access Only

Task 1: Create a Simple CUBE Calculation View

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

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on with your SAP HANA credentials: **STUDENT##** and **Training1** (where ## is your group number).
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 *Workspace* tree, right-click the folder *HA300_## → HDB → src → exercises* folder and choose *New → Calculation View*.

- b) Enter the calculation view name and other properties as specified in the table.
- c) Choose *Create*.
The calculation view graphical editor opens on the right of your screen.
3. In the *Aggregation* node, add the following data source:
Table SNWD_SO (*schema EPM_MODEL*).

**Note:**

This table is located in an external schema (not in the container of your application), and is accessed via a synonym HA300::SNWD_SO that has already been defined in the files you imported during the first exercise.

- a) Select the *Aggregation* node.
- b) Choose **+**, *Add Data Source*.
- c) In the *Find Data Sources* window, click the search field and enter **SNWD_**.
- d) In the search results, select the table **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 *View* → *Maximize Active Editor*.
Alternatively, you can double-click the corresponding tab title **CVC_SALES_SIMPLE.hdbculationview** or press **Ctrl+M**.
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. Save, and then build the *CVC_SALES_SIMPLE* calculation view.

Check the build status.

- a) Choose Save.

- b) Choose *Build* → *Build Selected Files*.

- c) In the *Console* pane, check that the build completes successfully.

12. Preview the data of the *CVC_SALES_SIMPLE* calculation view.

- a) If needed, to display the workspace, choose *View* → *Maximize Active Editor* (toggle to switch off maximized display mode).

Alternatively, you can double-click the corresponding tab title *CVC_SALES_SIMPLE.hdbculationview* or press **ctrl+M**.

- b) Right-click the CVC_SALES_SIMPLE calculation view and choose *Data Preview*.
- c) Display the *Raw Data* tab.
- d) To close the data preview, choose *X (Close)*.

Task 2: Create a Calculation View of Type SQL ACCESS ONLY

You have been asked to create an additional calculation view, similar to the first one, but not meant to be exposed to the end users. Instead, it is planned to reuse this view only as a data source for other calculation views.

1. Which type of view should you use for this purpose?

The calculation view type SQL ACCESS ONLY is what allows you to create a view that is not exposed to end users.

2. In your *Exercises* folder, copy the CVC_SALES_SIMPLE calculation view as **CVSQL_SALES_SIMPLE**.



Hint:

To rename the new copy more easily (in particular the runtime object name), first keep the proposed copy name, and then use the *Rename* feature.

- a) If needed, to display the *Workspace* pane, choose *View → Workspace*
- b) In the *Workspace* pane, expand the *HA300_##HDB → src → exercises* folder.
- c) Select the *CVC_SALES_SIMPLE.hdbculationview* item and press **Ctrl+C** (*copy*).
- d) Select the *Exercises* folder, press **Ctrl+V** and keep the proposed copy name *CopyOfCVC_SALES_SIMPLE...* (temporarily).
- e) Right-click the new file and choose *Rename*.
Alternatively, you can choose *Edit → Rename* or press **F2**.
- f) In the *Rename File* dialog box, adjust the new calculation view name to **CVSQL_SALES_SIMPLE.hdbculationview** and choose *Rename*.
- g) In the *Confirmation* dialog box, choose *Yes* to make sure the runtime object name is also updated.
- h) In the *Refactor Views* window, choose *Refactor*.
- i) Choose *Finish*.
The new calculation view is renamed and a build is automatically triggered. You will get a notification, and you should observe in the console that the build is successful.



Note:

You will learn about refactoring later in this course. For now, just keep in mind that it allows you to rename the runtime object name.

3. In the *Semantics* node, change the data category to **SQL ACCESS ONLY**.
- a) In the *Semantics* node, display the *View Properties* tab.

- b) In the *General* section, in the *Data Category* dropdown list, select **SQL ACCESS ONLY**.
4. Save, and then build the **CVSQL_SALES_SIMPLE** calculation view.
 Check the build status.
- Choose *File* → *Save*.
 - Choose *Build* → *Build Selected Files*.
 - In the *Console* pane, check that the build completes successfully.
5. Check which of the two calculation views are exposed in an external tool such as Microsoft Excel, using the menu *Data* → *Existing Connections*.

Use the following log-on info:

Table 3: SAP HANA Log-on Information

Field	Value
Host	wdf1bmt7215.wdf.sap.corp
Instance number	00
Database Mode	<i>Multi Database</i>
Database	<i>User Database: H00</i>
User	STUDENT##
Password	Training1
Language	EN

- Choose *Data* → *Existing Connections*.
- Choose *Browse for More*.
- Choose +*Connect to New Data Source.odc* and choose *Open*.
- Select *Other/Advanced* and choose *Next*.
- Select *SAP HANA MDX Provider* and choose *Next*.
- Enter the connection details as per the table.
- Choose *OK*.
- At the *Select Database and Table* step, in the drop-down list, select the entry **HA300_##_HDI_HDB_1.HA300**.
- Check which of the calculation views **CVC_SALES_SIMPLE** (type: **CUBE**) and **CVSQL_SALES_SIMPLE** (type: **SQL ACCESS ONLY**) is available in the drop-down list.



Note:

You should only see the **CVC_SALES_SIMPLE** calculation view (plus other ones), but not the **CVSQL_SALES_SIMPLE** calculation view.

- Choose *Cancel*.
- Exit Microsoft Excel.

6. Close all the open tabs in the SAP Web IDE for SAP HANA.
 - a) In the SAP Web IDE for SAP HANA, right-click any open tab and choose *Close All*.

Unit 2

Exercise 3

Create a DIMENSION Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Create a DIMENSION calculation view
- Create a calculated attribute in a calculation view

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.

The first step is 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.

1. Launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: HA300 → URLs.
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 data sources to the new node:
 - Table SNWD_BP (*schema EPM_MODEL*): Business Partners master data
 - Table SNWD_CONTACT (*schema EPM_MODEL*): Contact person details



Note:

These tables are located in an external schema (not in the container of your application), and they are accessed via synonyms that have already been defined in the files you imported during the first exercise.

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*.

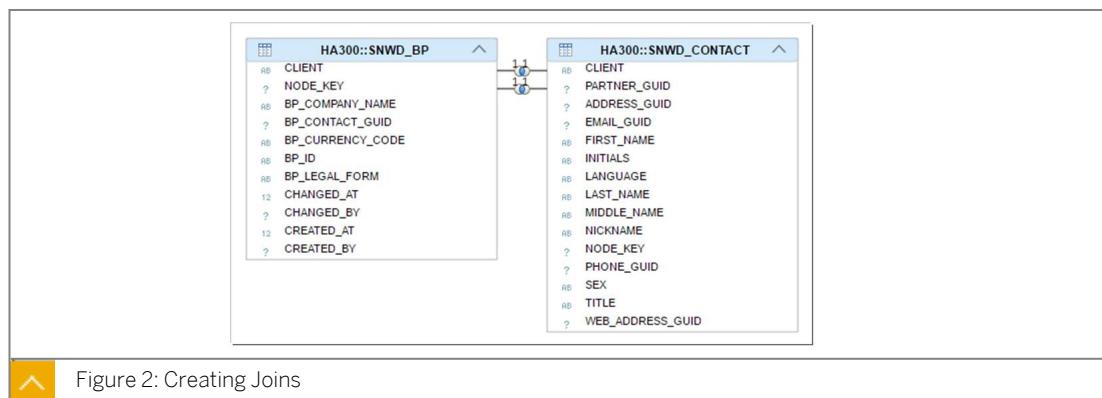


Figure 2: Creating Joins

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

Source table	Columns
SNWD_BP	<ul style="list-style-type: none"> • CLIENT • NODE_KEY • BP_ID • BP_COMPANY_NAME
SNWD_CONTACT	<ul style="list-style-type: none"> • 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 *EPM_MODEL.SNWD_BP_EM* (e-mail address)



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 turns green, 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 *EPM_MODEL.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
<i>Join_1</i>	[All the columns]
<i>SNWD_BP_EM</i>	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 *Semanticsnode*, 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. In the *Projection* node, create a calculated column *FULL_NAME* that combines the first and last name of the business partners.

The new column must be defined as follows:

Field	Value
Name	FULL_NAME
Label	FULL_NAME
Data Type	VARCHAR
Length	100
Expression	" FIRST_NAME " ' ' " LAST_NAME " (language: SQL)

13. Save, and then build the *CVD_BP_JOIN* calculation view.

Check the build status.

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

15. Close all the open tabs in the SAP Web IDE for SAP HANA.

Unit 2 Solution 3

Create a DIMENSION Calculation View

Exercise Objectives

After completing this exercise, you will be able to:

- Create a DIMENSION calculation view
- Create a calculated attribute in a calculation view

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.

The first step is 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.

1. Launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
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 *Workspace* tree, right-click the folder *HA300_## → HDB → src → exercises* folder and choose *New → Calculation View*.
 - b) Enter the calculation view name and other properties as specified in the table.
 - c) Choose *Create*.
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 data sources to the new node:
 - Table *SNWD_BP* (*schema EPM_MODEL*): Business Partners master data

- Table SNWD_CONTACT (schema EPM_MODEL): Contact person details

**Note:**

These tables are located in an external schema (not in the container of your application), and they are accessed via synonyms that have already been defined in the files you imported during the first exercise.

- In the Scenario pane, add a Join node below the top nodes by dragging a Join node from the palette.
 - Select the new *Join_1* node and click the + sign on the right of the node.
 - In the *Find Data Sources* window, click the search field and enter **SNWD_**.
 - In the search results, select the two tables SNWD_BP and SNWD_CONTACT.
 - Choose *Finish*.
4. If needed, expand the *Details* panel.
- 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*.

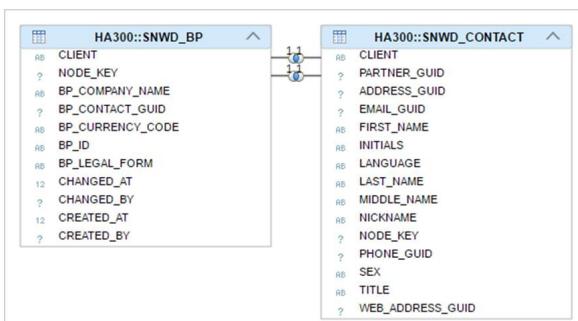


Figure 2: Creating Joins

- In the *Join Definition* tab, 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 *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	<ul style="list-style-type: none"> • CLIENT • NODE_KEY • BP_ID • BP_COMPANY_NAME
SNWD_CONTACT	<ul style="list-style-type: none"> • FIRST_NAME • LAST_NAME • LANGUAGE

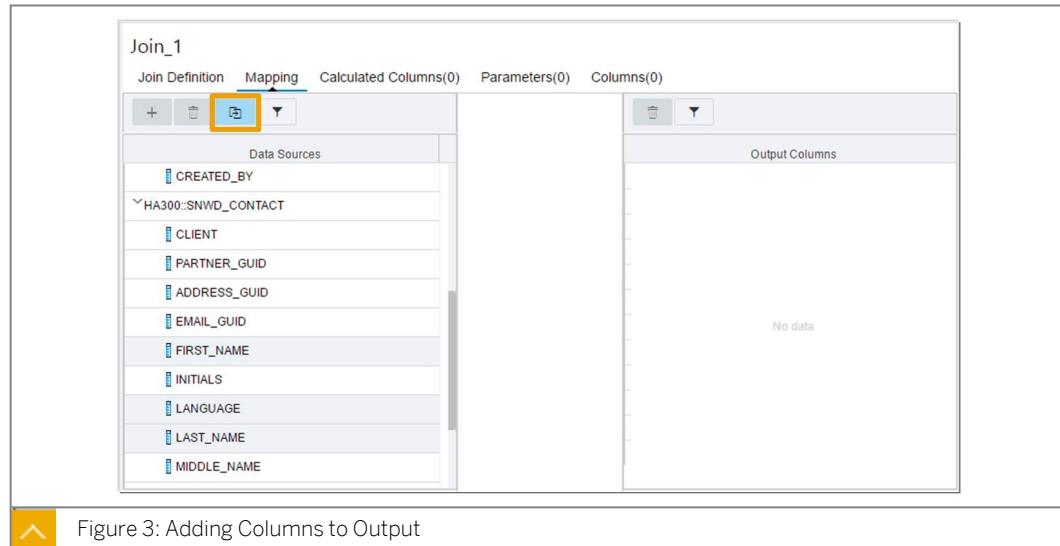


Hint:

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.



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 *EPM_MODEL.SNWD_BP_EM* (e-mail address)



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 turns green, you can drop the arrow icon.

- a) In the *Scenario* pane, add a new *Join* node *Join_2* below the top nodes by dragging a *Join* node from the palette.
 - 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 (EPM_MODEL)* 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 *EPM_MODEL.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.

- 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*).

- To edit the join properties, double-click the new connector.
 - In the *Join Type* dropdown list, choose *Referential*.
 - In the *Cardinality* dropdown list, choose *1..1*.
 - Choose *OK*.
- 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.

- In the *Mapping* tab for *Join_2*, select the *Join_1* item and choose
 - You do not need to select any specific column from this data source.
 - Select the *SNWD_BP_EM → EMAIL_ADDRESS* field and choose
- 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.
 - In the *Scenario* pane, select the *Join_2* node.
 - Drag the arrow icon to the *Projection* node.
 - In the *Mapping* tab of the *Projection* node, select the *Join_2* data source and choose
 - In the *Semanticsnode*, 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* column and choose *OK*.
 - c) Select the *Key* checkbox of the *BP_ID* column.
12. In the *Projection* node, create a calculated column *FULL_NAME* that combines the first and last name of the business partners.

The new column must be defined as follows:

Field	Value
Name	FULL_NAME
Label	FULL_NAME
Data Type	VARCHAR
Length	100
Expression	"FIRST_NAME" ' ' "LAST_NAME" (language: SQL)

- a) Select the *Projection* node.
- b) On the *Calculated Columns* tab, choose the *+* button.
- c) Double-click the new column to display its properties.
- d) Enter the properties as shown in the table above.



Note:

- If it's not visible on your keyboard, the | (pipe) sign can be obtained from the keyboard by entering **Alt+0124**.
- The equivalent syntax in *Column Engine* language would be:
"FIRST_NAME" + ' ' + "LAST_NAME"

- e) In the Expression Editor, choose *Validate Syntax*.

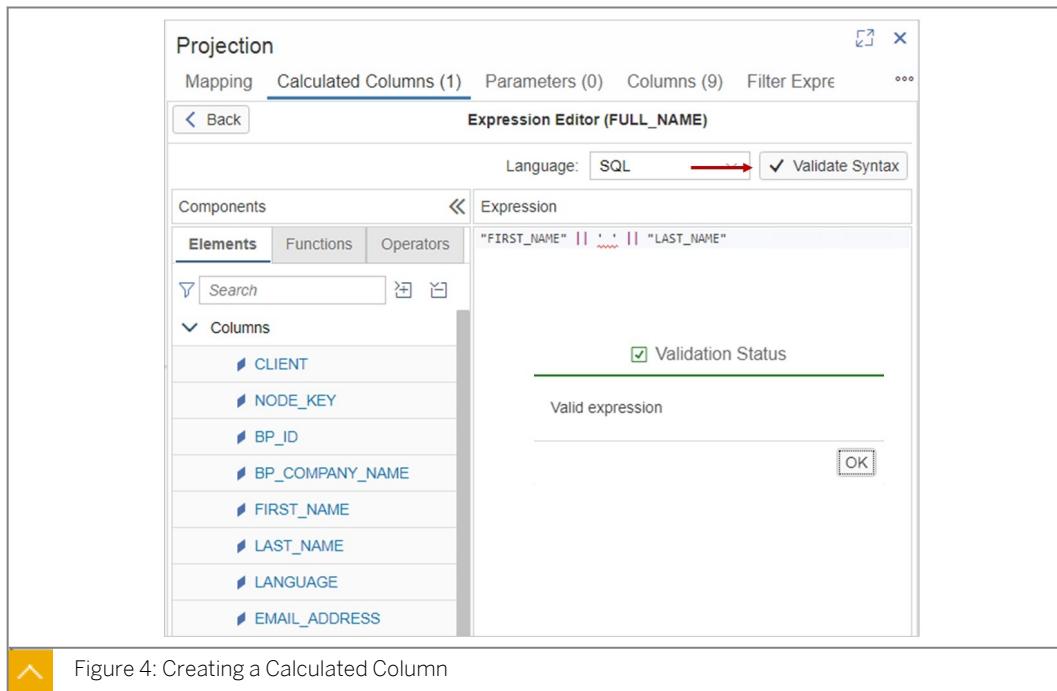


Figure 4: Creating a Calculated Column

f) Choose Back.

13. Save, and then build the CVD_BP_JOIN calculation view.

Check the build status.

a) Click Save.

b) In the Workspace tree, navigate to the folder HA300_## → HDB → src → exercises.

c) Right-click the CVD_BP_JOIN calculation view and choose Build Selected Files.

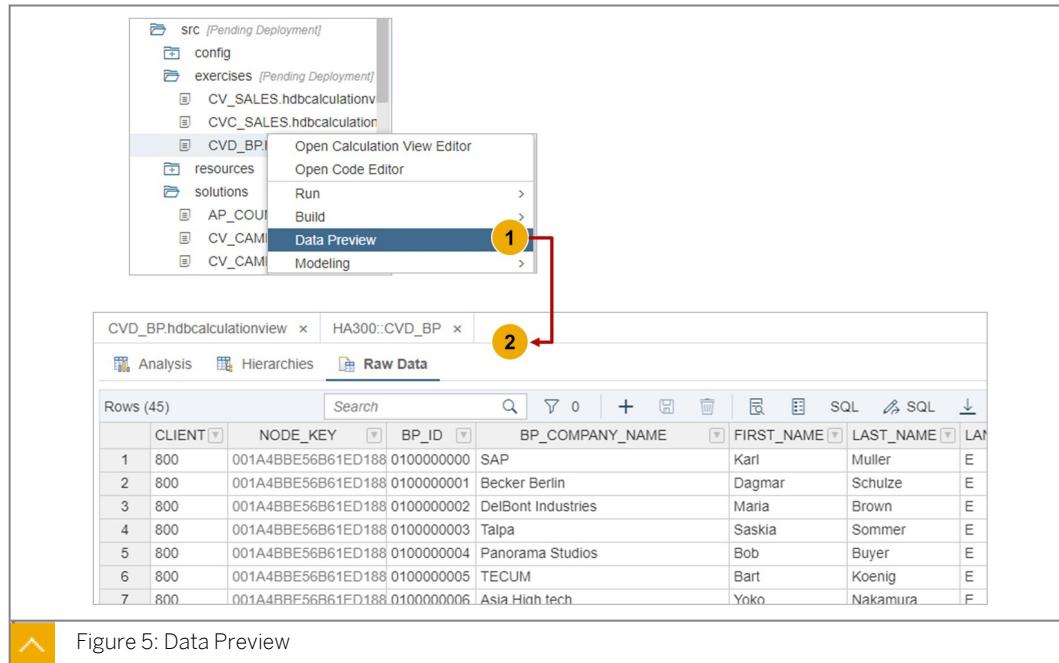
d) In the Console pane, check that the build completes successfully.

14. Preview the data of the CVD_BP_JOIN calculation view.

a) Right-click the CVD_BP_JOIN calculation view and choose Data Preview.

b) Display the Raw Data tab.

c) To close the data preview, choose X (Close).



15. Close all the open tabs in the SAP Web IDE for SAP HANA.

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

Unit 2

Exercise 4

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 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

Task 1: Review the Data Sources Used 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 *HA300::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 the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
2. In the Database Explorer, expand your container *MY HA300 CONTAINER* and open the three following tables to review their structure and content:
 - *HA300::MJ_SALES*
 - *HA300::MJ_CUSTOMERS*

- HA300::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 *Development* perspective.
2. In the workspace, import the *CVC_MJ_SALES.hdbculationview* file from your course files folder *HA300 → Calculation Views Templates* into the folder *HA300_## → HDB → 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 double-clicking the calculation view tab name, 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. Build the *CVC_MJ_SALES* calculation view.

Check the build 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?



Note:

An intermediate solution object is available at this stage:

`solutions → UNIT_2 → CVC_MJ_SALES_00_STAGE1.hdbcalculationview`

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. Save, and then build the *CVC_MJ_SALES* calculation view.
Check the build status.
 3. Based on this change, how many rows do you expect in the calculation view output? Why?
-
-
-

4. Preview the calculation view data on the *Raw Data* tab.

Does the output correspond to the expected behavior?

5. Close all open tabs in the SAP Web IDE for SAP HANA.

Unit 2 Solution 4

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 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

Task 1: Review the Data Sources Used 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 *HA300::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 the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on with your SAP HANA credentials: **STUDENT##** and **Training1** (where ## is your group number).
2. In the Database Explorer, expand your container *MY HA300 CONTAINER* and open the three following tables to review their structure and content:

- HA300::MJ_SALES
 - HA300::MJ_CUSTOMERS
 - HA300::MJ_COUNTRIES
- a) Choose *Tools* → *Database Explorer* or click the corresponding icon on the left toolbar.
- b) Expand the container *MY HA300 CONTAINER* to display the *Tables* folder.
- c) In the *Search* field, bottom-left, enter **MJ**.
- 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 *Development* perspective.
 - a) Choose *Tools* → *Development* or click the corresponding icon on the left toolbar.
2. In the workspace, import the *CVC_MJ_SALES.hdbcalculationview* file from your course files folder *HA300* → *Calculation Views Templates* into the folder *HA300_##* → *HDB* → *src* → *exercises*.
 - a) In the workspace, right-click your *exercises* folder and choose *import* → *File or Project*.
 - b) In the *Import* window, choose *Browse*.
 - c) Select the file *HA300* → *Calculation Views Templates* → *CVC_MJ_SALES.hdbcalculationview* and choose *Open*.
 - d) Make sure that the *Import to* location ends with *.../exercises*.
 - e) Choose *OK*.
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 double-clicking the calculation view tab name, 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) Optionally, double-click the CVC_MJ_SALES.hdbcalculationview tab and/or choose  *Maximize Details Panel*.
- c) In the *Join_1* node, review the *Join Definition* and *Mapping* tabs.
- d) 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 HA300::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. Build the CVC_MJ_SALES calculation view.

Check the build status.

- a) Choose *Build* → *Build Selected Files*.
- b) In the *Console* pane, check that the build 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.



Note:

An intermediate solution object is available at this stage:

solutions → *UNIT_2* → *CVC_MJ_SALES_00_STAGE1.hdbcalculationview*

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 *Mapping* tab of the *Join_1* node.
 - b) In the *PROPERTIES* pane, in the *Multi Join Order* drop-down list, select *Inside Out*.
2. Save, and then build the *CVC_MJ_SALES* calculation view.
Check the build status.
 - a) Choose *Save*.
 - b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build completes successfully.
3. 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).

4. 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.

5. Close all open tabs in the SAP Web IDE for SAP HANA.
 - a) Right-click any open tab and choose *Close All*.

Unit 2

Exercise 5

Combine Two Data Sources with a Union Node

Exercise Objectives

After completing this exercise, you will be able to:

- Create calculated columns in calculation views
- Use *Aggregation* nodes in calculation views
- Work with data types in calculation views
- Use *Union* nodes in calculation views

Business Example

A company wants to analyze its sales 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.

Task 1: Create a Calculation View to Support the Business Requirements

1. Launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: HA300 → URLs.
2. Create a new calculation view using the following details:

Property	Value
Name	CVC_SO_UNION
Label	Summary of Sales Orders from two Source Systems
Data Category	CUBE
With Star Join	[Deselected]

Task 2: Add the Sales Data from the SAP Source

1. In the scenario pane, create a new *Aggregation* node (in addition to the node automatically generated) at the bottom left of the view, and rename it **SAP_ERP_Data**.
2. Add the table SNWD_SO from the schema EPM_MODEL to the SAP_ERP_Data node.
3. Add the following columns from the table SNWD_SO to the output:
 - CLIENT

- *CURRENCY_CODE*
 - *GROSS_AMOUNT*
 - *NET_AMOUNT*
4. Check that the aggregation function *SUM* is assigned to the two measures *GROSS_AMOUNT* and *NET_AMOUNT*.

Task 3: Add the Sales Data from the Third Party Source

1. Create another Aggregation node **3rd_Party_Data** and place it to the right of the *SAP_ERP_Data* node. The data source for the new node is the table *SALES_DATA* from the *TRAINING* schema.
2. Add the following columns from the *3rd_Party_Data* Aggregation node to the output:
 - *CURRENCY*
 - *AMOUNT*

The *AMOUNT* column is a measure that must be aggregated with the *SUM* aggregate function.

Task 4: Create Calculated Columns

The third party data does not contain the net amount, so this measure must be calculated.

Additionally, the columns *GROSS_AMOUNT* and *NET_AMOUNT* from the SAP source are using the data type *Decimal (15,2)*, and you want to ensure that the gross and net amounts from third party data are using the same data type.

1. In the *3rd_Party_Data* Aggregation node, add a calculated column for the net amount, using the following details:

Field	Value
Name	NET_AMOUNT
Data Type	<i>Decimal</i>
Length	15
Scale	2
Expression	"AMOUNT" * 0.9

2. Add another calculated column for the gross amount using the following details:

Field	Value
Name	GROSS_AMOUNT_2
Data Type	<i>Decimal</i>
Length	15
Scale	2
Expression	"AMOUNT"

**Note:**

This calculated column is added for the sole purpose of data type conversion.

3. On the *Columns* tab of the *3rd_Party_Data* node, can you specify the aggregation function to apply to the calculated columns? Can you explain why?

Task 5: Combine the Two Data Sources

Use a *Union* node to combine the result set from the two lower *Aggregation* nodes.

1. In the *Scenario* pane, add a new *Union* node *Union_1* above the *SAP_ERP_Data* and *3rd_Party_Data* nodes, and connect these *Aggregation* nodes to the *Union_1* node.
2. On the *Mappings* tab of the *Union_1* node, specify how the columns from the two data sources must be combined.

All the columns from the *SAP_ERP_Data* must be mapped to the target with the same name.

For the columns of the *3rd_Party_Data* source, define the mapping as follows:

Source column	Target column
CURRENCY	CURRENCY_CODE
AMOUNT	[no mapping]
NET_AMOUNT	NET_AMOUNT
GROSS_AMOUNT_2	GROSS_AMOUNT

**Hint:**

You can use the *Auto Map by Name* feature after selecting the *SAP_ERP_Data* item in the *Data Sources* tree, and then finalize the mapping for remaining columns from the second data source by using drag and drop.

3. The data from the third party system does not contain the *CLIENT* column. To ensure consistent data, assign a constant value for the *CLIENT* target column to all the rows from the *3rd_Party_Data* source.

This value must be set to **800**, which is the value assigned to the SAP system data.

**Hint:**

To manage a column mapping, right-click the *CLIENT* target column and choose *Manage Mappings*.

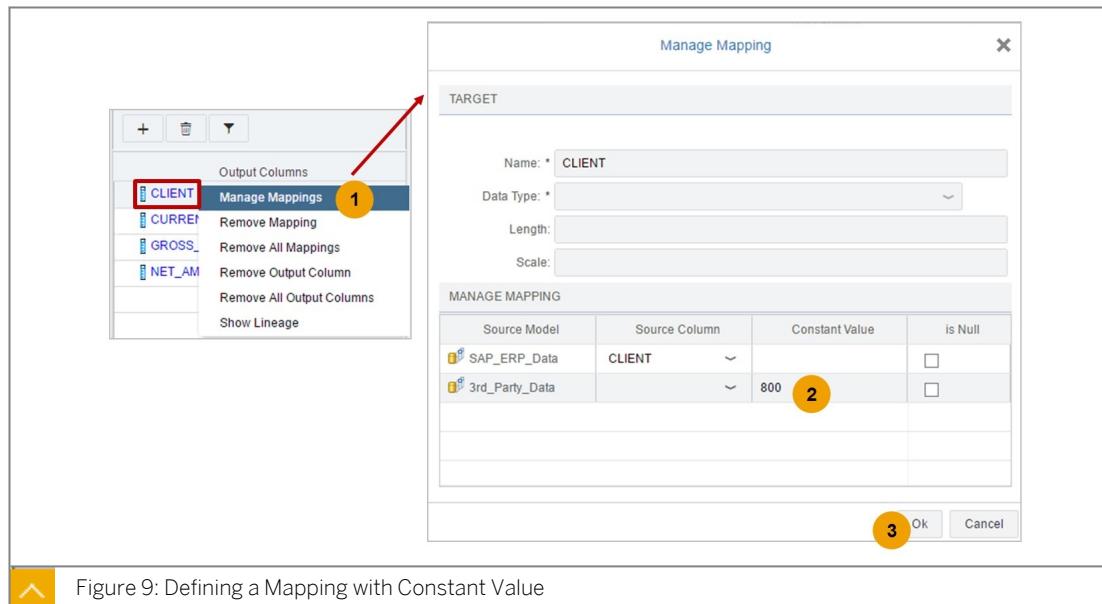


Figure 9: Defining a Mapping with Constant Value

Task 6: Finalize the Calculation View

1. In the scenario, connect the *Union_1* node to the uppermost *Aggregation* node.
2. Add all the columns of the *Aggregation* node to the output. Check that the columns *GROSS_AMOUNT* and *NET_AMOUNT* are defined as measures and aggregated with the *SUM* function, and that the two other columns are defined as attributes.
3. Save, and then build the *CVC_SO_UNION* calculation view.
Check the build status after activation.
4. Preview the data of the calculation view.
5. What is the value of *GROSS_AMOUNT* for *CURRENCY_CODE: USD*?

6. Close all the open tabs in the SAP Web IDE for SAP HANA.

Unit 2 Solution 5

Combine Two Data Sources with a Union Node

Exercise Objectives

After completing this exercise, you will be able to:

- Create calculated columns in calculation views
- Use *Aggregation* nodes in calculation views
- Work with data types in calculation views
- Use *Union* nodes in calculation views

Business Example

A company wants to analyze its sales 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.

Task 1: Create a Calculation View to Support the Business Requirements

1. Launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on with your SAP HANA credentials: **STUDENT##** and **Training1** (where ## is your group number).
2. Create a new calculation view using the following details:

Property	Value
Name	CVC_SO_UNION
Label	Summary of Sales Orders from two Source Systems
Data Category	CUBE
With Star Join	[Deselected]

- a) In the *Workspace* tree, right-click the folder *HA300_## → HDB → src → exercises* folder and choose *New → Calculation View*.
- b) Enter the calculation view name and other properties as specified in the table.

c) Choose *Create*.

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

Task 2: Add the Sales Data from the SAP Source

1. In the scenario pane, create a new *Aggregation* node (in addition to the node automatically generated) at the bottom left of the view, and rename it **SAP_ERP_Data**.
 - a) Drag the *Aggregation* node from the palette to the bottom left of Scenario pane.
 - b) Right-click the new node and choose *Rename*.
 - c) Enter **SAP_ERP_Data**.
2. Add the table **SNWD_SO** from the schema **EPM_MODEL** to the **SAP_ERP_Data** node.
 - a) Select the new **SAP_ERP_Data** node and click the + sign on the right of the node.
 - b) In the *Find Data Sources* window, click the search field and enter **SNWD_SO**.
 - c) In the search results, select the table.
 - d) Choose *Finish*.
3. Add the following columns from the table **SNWD_SO** to the output:
 - **CLIENT**
 - **CURRENCY_CODE**
 - **GROSS_AMOUNT**
 - **NET_AMOUNT**
 - a) In the *Mapping* tab for the **SAP_ERP_Data** Aggregation node, select the columns as per the list above in the *Data Sources*, and choose *Add To Output*.
 - b) Alternatively, you can drag and drop the columns from the *Data Sources* tree to the *Output Columns* area.
4. Check that the aggregation function **SUM** is assigned to the two measures **GROSS_AMOUNT** and **NET_AMOUNT**.
 - a) In the *Columns* tab for the **SAP_ERP_Data** node, select the **GROSS_AMOUNT** column and set the *Aggregation* property to **SUM**.
 - b) Repeat the previous step for the **NET_AMOUNT** column.

Task 3: Add the Sales Data from the Third Party Source

1. Create another *Aggregation* node **3rd_Party_Data** and place it to the right of the **SAP_ERP_Data** node. The data source for the new node is the table **SALES_DATA** from the **TRAINING** schema.
 - a) Drag the *Aggregation* node from the palette to the right of the **SAP_ERP_Data** Aggregation node.
 - b) Right-click the new node and choose *Rename*.
 - c) Enter **3rd_Party_Data**.
 - d) Select the new **3rd_Party_Data** node and click the + sign on the right of the node.

- e) In the *Find Data Sources* window, click the search field and enter **SALES_DATA**.
- f) In the search results, select the **SALES_DATA** table.
- g) Choose *Finish*.

2. Add the following columns from the *3rd_Party_Data* Aggregation node to the output:

- **CURRENCY**
- **AMOUNT**

The **AMOUNT** column is a measure that must be aggregated with the **SUM** aggregate function.

- a) On the *Mapping* tab for the *3rd_Party_Data* Aggregation node, select the columns as per the list above in the *Data Sources*, and choose *Add To Output*.
- b) Alternatively, you can drag and drop the columns from the *Data Sources* tree to the *Output Columns* area.
- c) On the *Columns* tab for the *3rd_Party_Data* node, select the **AMOUNT** column and set the *Aggregation* property to **SUM**.

Task 4: Create Calculated Columns

The third party data does not contain the net amount, so this measure must be calculated.

Additionally, the columns **GROSS_AMOUNT** and **NET_AMOUNT** from the SAP source are using the data type *Decimal (15,2)*, and you want to ensure that the gross and net amounts from third party data are using the same data type.

1. In the *3rd_Party_Data* Aggregation node, add a calculated column for the net amount, using the following details:

Field	Value
Name	NET_AMOUNT
Data Type	<i>Decimal</i>
Length	15
Scale	2
Expression	"AMOUNT" * 0.9

- a) Select the *3rd_Party_Data* Aggregation node.
- b) In the *Calculated Columns* tab, choose **+(Add)**.
- c) Enter the details as listed in the table.
- d) Optionally, to validate the syntax, choose *Expression Editor* and choose *Validate Syntax*. Then choose *Back*.

2. Add another calculated column for the gross amount using the following details:

Field	Value
Name	GROSS_AMOUNT_2
Data Type	<i>Decimal</i>

Field	Value
Length	15
Scale	2
Expression	"AMOUNT"

**Note:**

This calculated column is added for the sole purpose of data type conversion.

- a) Select the *3rd_Party_Data* Aggregation node.
 - b) In the *Calculated Columns* tab, choose +(Add).
 - c) Enter the details as listed in the table.
3. On the *Columns* tab of the *3rd_Party_Data* node, can you specify the aggregation function to apply to the calculated columns? Can you explain why?

It is not possible to define an aggregation function for a calculated column in an Aggregation node. This is because, in an Aggregation node of a calculation view, calculated columns are always computed AFTER the aggregation of the source data. If you need to calculate a column before aggregation, you can do that in an intermediate projection node that is then used as the data source for the Aggregation node.

Task 5: Combine the Two Data Sources

Use a *Union* node to combine the result set from the two lower Aggregation nodes.

1. In the Scenario pane, add a new *Union* node *Union_1* above the *SAP_ERP_Data* and *3rd_Party_Data* nodes, and connect these Aggregation nodes to the *Union_1* node.
- a) In the Scenario pane, drag the *Union* node from the palette to the area above the two Aggregation nodes in the Scenario pane.

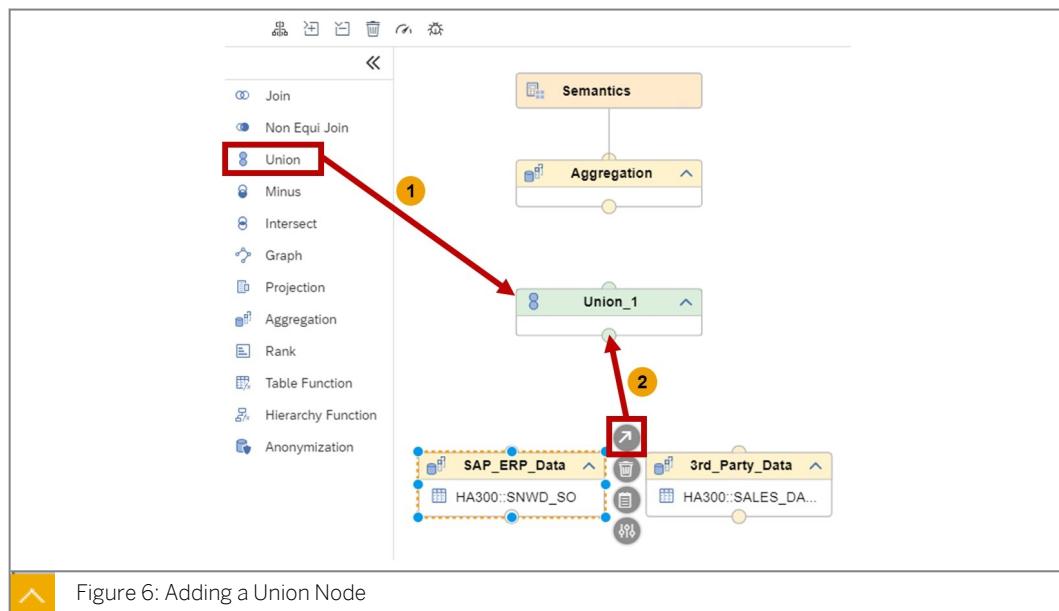


Figure 6: Adding a Union Node

- b) Select the *SAP_ERP_Data* node and drag the arrow icon to the *Union_1* node.
- c) Repeat the previous step for the *3rd_Party_Data* node.
2. On the *Mappings* tab of the *Union_1* node, specify how the columns from the two data sources must be combined.

All the columns from the *SAP_ERP_Data* must be mapped to the target with the same name.

For the columns of the *3rd_Party_Data* source, define the mapping as follows:

Source column	Target column
CURRENCY	CURRENCY_CODE
AMOUNT	[no mapping]
NET_AMOUNT	NET_AMOUNT
GROSS_AMOUNT_2	GROSS_AMOUNT



Hint:

You can use the *Auto Map by Name* feature after selecting the *SAP_ERP_Data* item in the *Data Sources* tree, and then finalize the mapping for remaining columns from the second data source by using drag and drop.

- a) Select the *Union_1* node and display the *Mappings* tab.
- b) In the *Data Sources* tree, select the *SAP_ERP_Data* item and choose *Auto Map by Name*.

Figure 7: Defining Mappings for Source SAP_ERP_Data

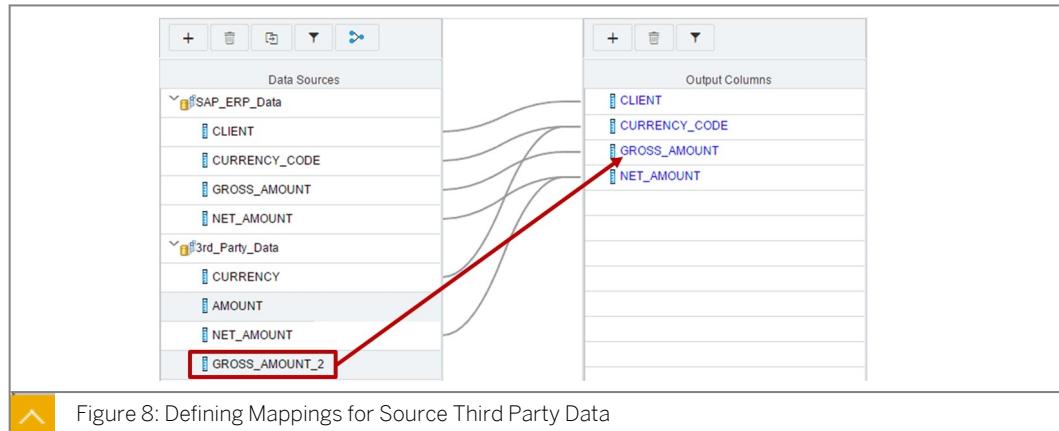
All the columns from the *SAP_ERP_Data* source are added to the target, and the *NET_AMOUNT* from the *3rd_Party_Data* is mapped to the corresponding output column.

- c) Drag the *CURRENCY* column from the *Data Sources* area to the *CURRENCY_CODE* column of the *Output Columns* list.

**Note:**

A warning icon will be displayed in the *Output Columns* pane for the *CURRENCY_CODE* column because the mapped columns do not have the same data types. In the context of this exercise, you can ignore this warning.

- d) Repeat the previous step for the *GROSS_AMOUNT_2* column.

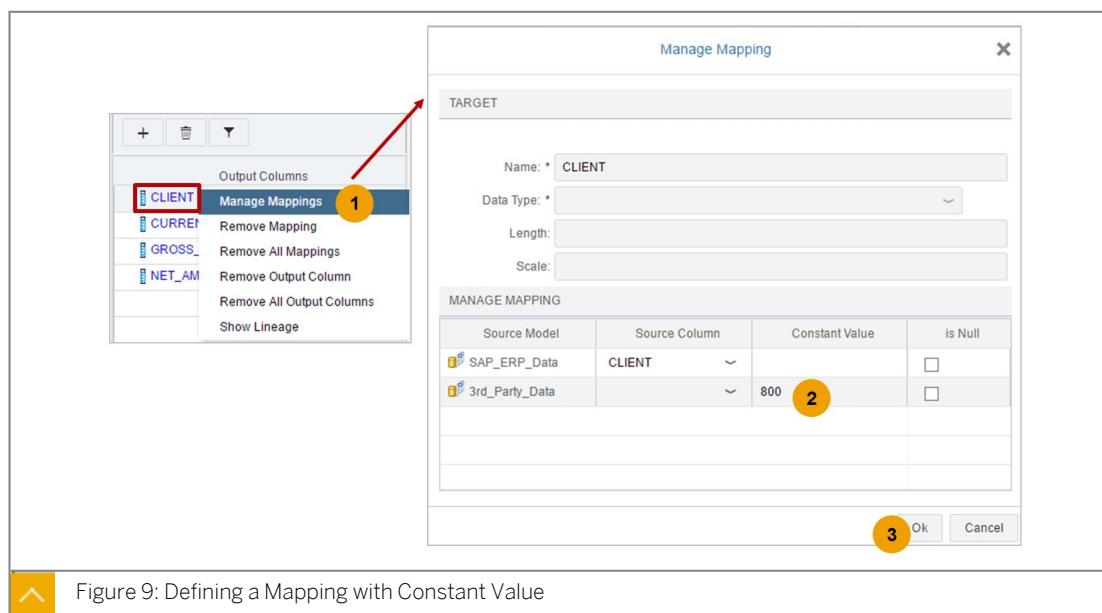


3. The data from the third party system does not contain the *CLIENT* column. To ensure consistent data, assign a constant value for the *CLIENT* target column to all the rows from the *3rd_Party_Data* source.

This value must be set to **800**, which is the value assigned to the SAP system data.

**Hint:**

To manage a column mapping, right-click the *CLIENT* target column and choose *Manage Mappings*.



- a) In the *Output Columns* list, right-click the *CLIENT* column and choose *Manage Mappings*.
- b) In the *Manage Mappings* dialog box, for the Source Model *3rd_Party_Data*, enter **800** in the *Constant Value* column.
- c) Choose *OK*.

Task 6: Finalize the Calculation View

1. In the scenario, connect the *Union_1* node to the uppermost *Aggregation* node.
 - a) Select the *Union_1* node and drag the arrow icon to the *Aggregation* node.
2. Add all the columns of the *Aggregation* node to the output. Check that the columns *GROSS_AMOUNT* and *NET_AMOUNT* are defined as measures and aggregated with the *SUM* function, and that the two other columns are defined as attributes.
 - a) Select the *Aggregation* node and display the *Mappings* tab.
 - b) In the *Data Sources* tree, select the *Union_1* node and choose *Add To Output*.
 - c) On the *Columns* tab, check the *Type* and *Aggregation* properties of the columns.
3. Save, and then build the *CVC_SO_UNION* calculation view.
Check the build status after activation.
 - a) Click *Save*.
 - b) Choose *Build → Build Selected Files*.
 - c) In the *Console* pane, check that the build completes successfully.
4. Preview the data of the calculation view.
 - a) In the workspace, right-click the *CVC_SO_UNION* calculation view and choose *Data Preview*.
5. What is the value of *GROSS_AMOUNT* for *CURRENCY_CODE*: **USD**?

16 913

6. Close all the open tabs in the SAP Web IDE for SAP HANA.
 - a) Right-click any open tab and choose *Close All*.

Unit 2

Exercise 6

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 a CUBE calculation view *HA300::CVCS_SO*. This view is the list of sales orders items, with details about the product category, and also some 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.

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
2. In your exercises folder, create a new calculation **CVSQl_MINUS**. The calculation view type should be **SQL Access Only** 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 a *Projection* node **NB** to filter the data source *HA300:CVCS_SO* for the product category *Notebooks*.

The filter expression should be:

```
"CATEGORY" = 'Notebooks'
```

Observe the available columns from the data source. Which ones do you need to map to fulfill the business requirement?

-
-
-
5. Copy and modify the *NB* node as **sw** and adjust the Filter Expression so that it returns the product category *Software*.



Hint:
You need to right-click the *NB* node first.

6. Connect the two *Projection* nodes *NB* and *SW* to the *Minus* node and map the columns.
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?

7. Add the two required columns to the top *Projection* node.
8. Save, and then build the *CVSQL_MINUS* calculation view
Check the build status.
9. Preview the data of the calculation view.

How many customers are retrieved? Where are they located?

10. 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 "HA300_##_HDI_HDB"."HA300::CVSQL_MINUS"
GROUP BY "COUNTRY";
```

Which countries are returned? What does it mean?

What about the other countries?

11. Close all open tabs in the SAP Web IDE for SAP HANA.

Unit 2 Solution 6

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 a CUBE calculation view *HA300::CVCS_SO*. This view is the list of sales orders items, with details about the product category, and also some 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.

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on with your SAP HANA credentials: **STUDENT##** and **Training1** (where ## is your group number).
2. In your exercises folder, create a new calculation **CVSQL_MINUS**. The calculation view type should be *SQL Access Only* and the description **Customers – NB minus SW**
 - a) Right-click the *exercises* folder and choose *New → Calculation View*.
 - b) Enter the specified details and choose *Create*.
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. The node is called *Minus_1*.
 - b) Select the *Minus_1* node and drag the arrow icon at the right
4. Add a *Projection* node **NB** to filter the data source *HA300:CVCS_SO* for the product category *Notebooks*.
The filter expression should be:

```
"CATEGORY" = 'Notebooks'
```

Observe the available columns from the data source. Which ones do you need to map to fulfill the business requirement?

The required columns are `BP_COMPANY_NAME` and `COUNTRY`, plus the `CATEGORY` column which will be used to filter the data set.

- a) In the palette, choose *Create Projection* and drop the node below the *Minus_1* node.
- b) Right-click the node, choose *Rename* and enter **NB**.
- c) Choose *+ Add Data Source* and enter **cvcs_so** in the *Search* field.
- d) In the search results, select the *HA300:CVCS_SO* view and choose *Finish*.
- e) Double-click the *NB* node to open its details.
- f) In *Mapping* tab, drag and drop the columns mentioned above to the *Output Columns* area.
- g) In the *Filter Expression* tab, enter the expression:

```
"CATEGORY" = 'Notebooks'
```



Hint:
To find the *Filter Expression* tab, depending on the screen size and layout, you might need to click first the **More** button.

- h) Choose *Validate Syntax* and choose *OK* when validity is confirmed.
5. Copy and modify the *NB* node as **sw** and adjust the Filter Expression so that it returns the product category *Software*.



Hint:
You need to right-click the *NB* node first.

- a) Right-click the *NB* node and choose *Copy One*.
Choosing *Copy All Below* in this context would have the same effect because there are no nodes connected to the *NB* one.
- b) Right-click the blank area right next to the *NB* node and choose *Paste*.
- c) Right-click the pasted node, choose *Rename* and enter **sw**.
- d) Adjust the Filter Expression, replacing **Notebooks** with **Software**.
- e) Validate the expression syntax.
6. Connect the two *Projection* nodes *NB* and *SW* to the *Minus* node and map the columns.
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 NB node and drag the arrow on the right to the *Minus_1* node.
- b) Repeat this for the SW node.
- c) Select the *Minus_1* node.
- d) In the *Mapping* tab, choose *Auto Map by Name*.

You might need to click the *Toolbar Overflow* button first.

- e) In the *Output Columns* area, right-click the CATEGORY column and choose *Remove Output Column*.

7. Add the two required columns to the top *Projection* node.

- a) In the *Mapping* tab of the *Projection* node, select the *Minus_1* data source.
- b) Choose *Add To Output*.

8. Save, and then build the *CVSQL_MINUS* calculation view

Check the build status.

- a) Choose *Save*.
- b) Choose *Build* → *Build Selected Files*.
- c) In the *Console* pane, check that the build completes successfully.

9. Preview the data of the calculation view.

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

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).

10. 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 "HA300_##_HDI_HDB"."HA300::CVSQL_MINUS"
GROUP BY "COUNTRY";
```

- a) Remove the two occurrences of "*BP_COMPANY_NAME*", from the *SELECT* and *GROUP BY* clauses of the SQL statement.
- b) 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.

11. Close all open tabs in the SAP Web IDE for SAP HANA.
 - a) Right-click any open tab and choose *Close All*.

Unit 2 Exercise 7

Control the Behavior of the Aggregation Node

Exercise Objectives

After completing this exercise, you will be able to:

- Control the behavior of the Aggregation node when you work with measures such as quantity and unit price.

Business Example

You want to analyze a set of data that represents sales orders for different products, stores, and customers.

The source data includes the quantity and unit price for each sales order, but not the total price: you will have to set up your information view so that it computes correctly the total price regardless of what columns are requested by the client tool query.

In order to better understand how the view behaves, you will execute different SQL queries on top of the calculation view in SAP HANA Studio.



Note:

For the sake of simplicity, there is only one product per sales order.

Task 1: Visualize the Source Data



Note:

Before attempting this task, open a file that contains several SQL statements which you will use during the exercise

1. If necessary, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs*.
2. In the Database Explorer, select your database container and open an SQL console.
3. In the SQL console, import the following file containing prepared SQL statements:
HA300 → Prepared Code → SQL for Aggregation Nodes.sql.
4. Execute the first query (under QUERY 1.a in your SQL console) to visualize the content of the table that will be used during the exercise.
The table is accessed via the synonym *HA300::ORDERS*.

```
SELECT * FROM "HA300::ORDERS";
```

The table contains order details by order ID, product, shop, and customer. The quantity and unit price are in the table, but not the total price by line item (each order has only one line item).

**Caution:**

Until the end of this exercise, make sure that you execute only the SQL statements specified at each step. Do NOT execute the entire set of instructions from the SQL console.

Task 2: Create a Calculation View to Calculate the Total Sales

1. In your HA300_## application, create a new calculation view **CVC_ORDERS_AGGREGATION** in the exercises folder, with the following properties:

Property	Value
Name	CVC_ORDERS_AGGREGATION
Label	Sales Orders with Aggregation
Data Category	CUBE
With Star Join	[Deselected]

2. Add a new Aggregation node **Aggregation_1** to the calculation view scenario, and define it as the data source for the Aggregation node.
3. Add the table **ORDERS** from the schema **TRAINING** as the data source for the **Aggregation_1** node.
Add all the columns from the table to the output, and make sure that only the columns **QUANTITY** and **UNIT_PRICE** are aggregated.
4. Propagate all the columns of the **Aggregation_1** node to the Semantics.
5. In the **Aggregation_1** node, create a calculated column **TOTAL_PRICE**, with the following properties:

Property	Value
Name	TOTAL_PRICE
Data Type	<i>Decimal</i>
Length	15
Scale	2
Expression Language	SQL
Expression	"QUANTITY" * "UNIT_PRICE"

6. Add the **TOTAL_PRICE** column to the output of the Aggregation node.
7. Check that the relevant type is assigned to the columns in the semantics:
The **QUANTITY**, **UNIT_PRICE** and **TOTAL_PRICE** columns are *Measures*, other columns are *Attributes*.
8. Save, and then build the **CVC_ORDERS_AGGREGATION** calculation view.
Check the build status after activation.
9. Preview the data with the standard SAP HANA functionality.

The data preview is consistent and the *TOTAL_PRICE* is correctly calculated.

Task 3: Query the New Calculation View

To further investigate the behavior of the Aggregation node, you will now execute several SQL statements on top of the calculation view *CVC_ORDERS_AGGREGATION*. The main difference between the queries is the list of attribute columns and, in turn, the *GROUP BY* clause.

1. Switch to the *Database Explorer* perspective and select the SQL console that you opened during Task 1.
2. Execute a query to retrieve the quantities and price by order ID and product.

The query is under QUERY 3.a in the SQL console that you opened at the beginning of the exercise.

```
SELECT
    "ORDER_ID",
    "PRODUCT",
    sum("QUANTITY") AS "QUANTITY",
    sum("UNIT_PRICE") AS "UNIT_PRICE",
    sum("TOTAL_PRICE") AS "TOTAL_PRICE"
FROM "HA300::CVC_ORDERS_AGGREGATION"
GROUP BY "ORDER_ID",
    "PRODUCT";
```

3. Execute a query to retrieve the quantities and price only by Product.

The query is under QUERY 3.b in the SQL console.

```
SELECT
    "PRODUCT",
    sum("QUANTITY") AS "QUANTITY",
    sum("UNIT PRICE") AS "UNIT PRICE",
    sum("TOTAL PRICE") AS "TOTAL PRICE"
FROM "HA300::CVC_ORDERS_AGGREGATION"
GROUP BY "PRODUCT";
```

What do you observe? How would you explain it?



Note:

Here, we keep the *UNIT_PRICE* column in the query to better understand how the total price has been computed. In real life, you would not sum up this column because this measure is not relevant when it is aggregated.



Note:

An intermediate solution object is available at this stage:

solutions → *UNIT_2* → *CVC_ORDERS_AGGREGATION_00_STAGE1*

Task 4: Modify Your Calculation View to Return Consistent Data

Modify your *CVC_ORDERS_AGGREGATION* calculation view, to force the *ORDER_ID* column to participate in the *GROUP BY* clause of the view aggregation, even if it is not selected by the top query.

1. In the *Aggregation_1* node of your calculation view, set the *Keep Flag* property for the *ORDER_ID* column to *True*.



Hint:

Select the *ORDER_ID* column in the *Output* pane on the right of your screen.

2. Save, and then build the calculation view.
3. To retrieve the quantities and price only by product, execute the previous SQL query again.

The query is under QUERY 3.b in the SQL console.

```
SELECT
    "PRODUCT",
    sum("QUANTITY") AS "QUANTITY",
    sum("UNIT PRICE") AS "UNIT PRICE",
    sum("TOTAL PRICE") AS "TOTAL PRICE"
FROM "HA300::CVC_ORDERS_AGGREGATION"
GROUP BY "PRODUCT";
```

What do you observe?

4. To retrieve the quantities and total price by product and store, execute a query.

The query is under QUERY 4.a in the SQL console.

```
SELECT
    "PRODUCT",
    "STORE",
    sum("QUANTITY") AS "QUANTITY",
    sum("TOTAL PRICE") AS "TOTAL PRICE"
FROM "HA300::CVC_ORDERS_AGGREGATION"
GROUP BY "PRODUCT",
        "STORE";
```

Is the result still consistent?

5. Close all the open tabs in the SAP Web IDE for SAP HANA.

Control the Behavior of the Aggregation Node

Exercise Objectives

After completing this exercise, you will be able to:

- Control the behavior of the Aggregation node when you work with measures such as quantity and unit price.

Business Example

You want to analyze a set of data that represents sales orders for different products, stores, and customers.

The source data includes the quantity and unit price for each sales order, but not the total price: you will have to set up your information view so that it computes correctly the total price regardless of what columns are requested by the client tool query.

In order to better understand how the view behaves, you will execute different SQL queries on top of the calculation view in SAP HANA Studio.



Note:

For the sake of simplicity, there is only one product per sales order.

Task 1: Visualize the Source Data



Note:

Before attempting this task, open a file that contains several SQL statements which you will use during the exercise

1. If necessary, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on with your SAP HANA credentials: **STUDENT##** and **Training1** (where ## is your group number).
2. In the Database Explorer, select your database container and open an SQL console.
 - a) To open the *Database Explorer* view, choose *Tools → Database Explorer*. Alternatively, you can click the corresponding icon on the left toolbar.
 - b) In the Database Browser, select your HDI Container *MY HA300 CONTAINER*.
 - c) Choose the *Open SQL Console* button in the toolbar.
3. In the SQL console, import the following file containing prepared SQL statements:

HA300 → Prepared Code → SQL for Aggregation Nodes.sql.

- a) Choose *Import File*.
- b) Select the file specified above and choose *Open*.
4. Execute the first query (under QUERY 1.a in your SQL console) to visualize the content of the table that will be used during the exercise.

The table is accessed via the synonym HA300::ORDERS.

```
SELECT * FROM "HA300::ORDERS";
```

- a) Highlight the specified statement and choose *Run (F8)*.
- b) Review the output in the *Result* tab.

The table contains order details by order ID, product, shop, and customer. The quantity and unit price are in the table, but not the total price by line item (each order has only one line item).



Caution:

Until the end of this exercise, make sure that you execute only the SQL statements specified at each step. Do NOT execute the entire set of instructions from the SQL console.

Task 2: Create a Calculation View to Calculate the Total Sales

1. In your HA300_## application, create a new calculation view CVC_ORDERS_AGGREGATION in the exercises folder, with the following properties:

Property	Value
Name	CVC_ORDERS_AGGREGATION
Label	Sales Orders with Aggregation
Data Category	CUBE
With Star Join	[Deselected]

- a) Choose *Tools → Development*.

Alternatively, you can click the corresponding icon on the left toolbar.

- b) Right-click the HA300_## → HDB → src → exercises and choose *New → Calculation View*
- c) Enter the details as shown in the table.
- d) Choose *Create*.

2. Add a new Aggregation node Aggregation_1 to the calculation view scenario, and define it as the data source for the Aggregation node.
- a) Drag the Aggregation node from the palette to the bottom of the Scenario pane.
The new node is automatically named Aggregation_1.
- b) Select the Aggregation_1 node and drag the arrow icon to the Aggregation node.

3. Add the table **ORDERS** from the schema **TRAINING** as the data source for the **Aggregation_1** node.
 Add all the columns from the table to the output, and make sure that only the columns **QUANTITY** and **UNIT_PRICE** are aggregated.
 - a) Select the new **Aggregation_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 **ORDERS**.
 - c) In the search results, select the table **ORDERS**.
 - d) Choose *Finish*.
 - e) In the *Mapping* tab, select the **HA300::ORDERS** object from the *Data Sources* area and choose *Add to Output*.
 Alternatively, you can drag the **HA300::ORDERS** data source to the *Output Columns* area.
 - f) In the *Columns* tab, set the *Aggregation* parameter for column **ORDER_ID** to [blank].
4. Propagate all the columns of the **Aggregation_1** node to the Semantics.
 - a) Select the **Aggregation_1** node, if needed, and display the *Mapping* tab.
 - b) In the *Output Columns* area, select all the columns (click the first one, then on your keyboard, press **Shift** and click the last one).
 - c) Right-click the selected columns and choose *Propagate to Semantics*.
 - d) In the confirmation dialog box, choose *OK*.
5. In the **Aggregation_1** node, create a calculated column **TOTAL_PRICE**, with the following properties:

Property	Value
Name	TOTAL_PRICE
Data Type	<i>Decimal</i>
Length	15
Scale	2
Expression Language	<i>SQL</i>
Expression	"QUANTITY" * "UNIT_PRICE"

 - a) Select the **Aggregation_1** node.
 - b) In the *Calculated Columns* tab, choose **+(Add)**.
 - c) Enter the details as listed in the table.
 - d) Optionally, to validate the syntax, choose *Expression Editor* and choose *Validate Syntax*. Then choose *Back*.
6. Add the **TOTAL_PRICE** column to the output of the Aggregation node.
 - a) Select the Aggregation node.
 - b) In the *Mapping* tab, select the **TOTAL_PRICE** columns and choose *Add to Output*.

Alternatively, you can drag the *TOTAL_PRICE* column to the *Output Columns* area.

7. Check that the relevant type is assigned to the columns in the semantics:
The *QUANTITY*, *UNIT_PRICE* and *TOTAL_PRICE* columns are *Measures*, other columns are *Attributes*.
 - a) Select the *Semantics* node.
 - b) In the *Columns* tab, check the *Type* setting for each column.
8. Save, and then build the *CVC_ORDERS_AGGREGATION* calculation view.
Check the build status after activation.
 - a) Click *Save*.
 - b) In the *Workspace* tree, navigate to the folder *HA300_## → HDB → src → exercises*.
 - c) Right-click the *CVC_ORDERS_AGGREGATION* calculation view and choose *Build Selected Files*.
 - d) In the *Console* pane, check that the build completes successfully.
9. Preview the data with the standard SAP HANA functionality.
 - a) Right-click the *CVC_ORDERS_AGGREGATION* calculation view and choose *Data Preview*.
 - b) Check the data displayed in the *Data Preview* tab.

The data preview is consistent and the *TOTAL_PRICE* is correctly calculated.

Task 3: Query the New Calculation View

To further investigate the behavior of the *Aggregation* node, you will now execute several SQL statements on top of the calculation view *CVC_ORDERS_AGGREGATION*. The main difference between the queries is the list of attribute columns and, in turn, the *GROUP BY* clause.

1. Switch to the *Database Explorer* perspective and select the SQL console that you opened during Task 1.
 - a) Choose *Tools → Database Explorer*.

Alternatively, you can click the corresponding icon in the left toolbar of the SAP Web IDE for SAP HANA.

- b) Display the tab *SQL Console 1.sql* that contains your SQL statements (the tab number could be something else than 1).

2. Execute a query to retrieve the quantities and price by order ID and product.

The query is under *QUERY 3.a* in the SQL console that you opened at the beginning of the exercise.

```
SELECT
  "ORDER_ID",
  "PRODUCT",
  sum("QUANTITY") AS "QUANTITY",
  sum("UNIT PRICE") AS "UNIT PRICE",
  sum("TOTAL PRICE") AS "TOTAL PRICE"
FROM "HA300::CVC_ORDERS_AGGREGATION"
GROUP BY "ORDER_ID",
  "PRODUCT";
```

- a) Highlight the specified statement and choose *Execute (F8)*.

- b) Review the output in the *Result* tab.

This is the same result as what you got from the data preview at the end of Task 2, except the fact that the *STORE* and *CUSTOMER* columns are not queried (which has no impact on the aggregated measures and number of rows).

3. Execute a query to retrieve the quantities and price only by Product.

The query is under QUERY 3.b in the SQL console.

```
SELECT
    "PRODUCT",
    sum("QUANTITY") AS "QUANTITY",
    sum("UNIT_PRICE") AS "UNIT_PRICE",
    sum("TOTAL_PRICE") AS "TOTAL_PRICE"
FROM "HA300::CVC_ORDERS_AGGREGATION"
GROUP BY "PRODUCT";
```

What do you observe? How would you explain it?



Note:

Here, we keep the *UNIT_PRICE* column in the query to better understand how the total price has been computed. In real life, you would not sum up this column because this measure is not relevant when it is aggregated.

- a) Highlight the specified statement and choose *Execute (F8)*.

- b) Review the output in the *Result* tab.

This time, the total price is not correctly calculated because the unit price is inconsistent. Indeed, the *Order ID* attribute was not part of the Select statement, so for each product, the unit price that is returned is the sum of the unit prices of all source rows.



Note:

An intermediate solution object is available at this stage:

solutions → *UNIT_2* → *CVC_ORDERS_AGGREGATION_00_STAGE1*

Task 4: Modify Your Calculation View to Return Consistent Data

Modify your *CVC_ORDERS_AGGREGATION* calculation view, to force the *ORDER_ID* column to participate in the *GROUP BY* clause of the view aggregation, even if it is not selected by the top query.

1. In the *Aggregation_1* node of your calculation view, set the *Keep Flag* property for the *ORDER_ID* column to *True*.



Hint:

Select the *ORDER_ID* column in the *Output* pane on the right of your screen.

- a) In the *Scenario* pane, select the *Aggregation_1* node.

- b) In the *Columns* pane, select the *Keep Flag* checkbox for the *ORDER_ID* column.

2. Save, and then build the calculation view.

- a) Click Save.
 - b) Right-click the CVC_ORDERS_AGGREGATION calculation view and choose *Build Selected Files*.
3. To retrieve the quantities and price only by product, execute the previous SQL query again.

The query is under QUERY 3.b in the SQL console.

```
SELECT
    "PRODUCT",
    sum("QUANTITY") AS "QUANTITY",
    sum("UNIT_PRICE") AS "UNIT_PRICE",
    sum("TOTAL_PRICE") AS "TOTAL_PRICE"
FROM "HA300::CVC_ORDERS_AGGREGATION"
GROUP BY "PRODUCT";
```

What do you observe?

- a) Highlight the specified statement and choose *Execute (F8)*.
- b) Review the output in the *Result* tab.
The total price has been correctly calculated, even with summarized data by product. The *Keep Flag* property has enforced the computation of the total price at the order ID level, where this measure can be calculated as Quantity * Unit Price. Any further aggregation (by product, store or customer) will return consistent results.

4. To retrieve the quantities and total price by product and store, execute a query.

The query is under QUERY 4.a in the SQL console.

```
SELECT
    "PRODUCT",
    "STORE",
    sum("QUANTITY") AS "QUANTITY",
    sum("TOTAL_PRICE") AS "TOTAL_PRICE"
FROM "HA300::CVC_ORDERS_AGGREGATION"
GROUP BY "PRODUCT",
    "STORE";
```

Is the result still consistent?

- a) Highlight the specified statement and choose *Execute (F8)*.
- b) Review the output in the *Result* tab.
The result from the query is still consistent: You can group the result by different attributes in the query you run on top of the calculation view.

5. Close all the open tabs in the SAP Web IDE for SAP HANA.

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

Unit 2

Exercise 8

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, including the one you created for business partners.

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.

In this exercise, when the values include ##, replace the characters with the number that your instructor assigned you.

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.

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 data sources (tables from the *EPM_MODEL* schema) to the new node:

- Table *SNWD_PO* (*schema EPM_MODEL*): Purchase order document headers
- Table *SNWD_PO_I* (*schema EPM_MODEL*): Purchase order line items

**Note:**

These tables are located in an external schema (not in the container of your application), and they are accessed via synonyms that have already been defined in the files you imported during the first exercise.

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	<ul style="list-style-type: none"> • CLIENT • PO_ID • PARTNER_GUID
SNWD_PO_I	<ul style="list-style-type: none"> • 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)

Source Folder	Dimension Calculation View
resources	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.

**Note:**

When joining columns, if a dialog box says that a column from the *Join_1* node will be removed, confirm it by choosing Yes.

9. 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.

10. To avoid conflicting column names during build, give 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

11. In the *Mapping* tab of the *Star Join* node, add all columns from the *Join_1* table to the output, **except** the *PARTNER_GUID* and *PRODUCT_GUID*.



Note:

We make the assumption that these columns, used only to join the fact table to the dimension views, are not required by the end user of the CVCS_PO View.

12. 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.
13. In the *Semantics* node, rename the Client column as **CLIENT**.



Note:

Because the *CLIENT* columns from the dimension tables have been added to the Star Join before defining the mapping within the Star Join node, the *CLIENT* column originating from the *Join_1* node has been renamed *JOIN_2* or something equivalent in a previous step (column mapping in the *Star Join* node).

14. Save, and then build the CVCS_PO calculation view.
Check the build status.
15. Preview the data of the CVCS_PO calculation view.
16. Close all the open tabs in the SAP Web IDE for SAP HANA.

Unit 2 Solution 8

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, including the one you created for business partners.

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.

In this exercise, when the values include ##, replace the characters with the number that your instructor assigned you.

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
 - a) Start Windows Explorer and navigate to the folder HA300 → URLs.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on with your SAP HANA credentials: **STUDENT##** and **Training1** (where ## is your group number).
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 folder HA300_## → HDB → src → exercises folder and choose New → Calculation View.
- b) Enter the calculation view name and other properties as specified in the table.

c) Choose *Create*.

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 data sources (tables from the *EPM_MODEL* schema) to the new node:

- Table *SNWD_PO* (*schema EPM_MODEL*): Purchase order document headers
- Table *SNWD_PO_I* (*schema EPM_MODEL*): Purchase order line items



Note:

These tables are located in an external schema (not in the container of your application), and they are accessed via synonyms that have already been defined in the files you imported during the first exercise.

- a) In the *Scenario* pane, add a *Join* node below the top node by dragging a *Join* node from the palette.

- 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) If the *Join Definition* tab shows only the data source names without the corresponding columns, select the *Star Join* node and then select the *Join_1* node again.

- b) 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*).

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

- d) In the *Join Type* dropdown list, choose *Referential*.
- e) In the *Cardinality* dropdown list, choose *1..n*.
- f) 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	<ul style="list-style-type: none"> • CLIENT • PO_ID • PARTNER_GUID
SNWD_PO_I	<ul style="list-style-type: none"> • 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)
resources	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 **OK**.
 - 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.



Note:

When joining columns, if a dialog box says that a column from the *Join_1* node will be removed, confirm it by choosing Yes.

- 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. 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.

10. To avoid conflicting column names during build, give 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.

11. In the *Mapping* tab of the *Star Join* node, add all columns from the *Join_1* table to the output, **except** the *PARTNER_GUID* and *PRODUCT_GUID*.



Note:

We make the assumption that these columns, used only to join the fact table to the dimension views, are not required by the end user of the *CVCS_PO* View.

a) Display the *Mapping* tab for the *Star Join* node.

b) In the *Data Sources* area, select (click) the specified columns of the *Join_1* source (that is, all columns except *PARTNER_GUID* and *PRODUCT_GUID*) and choose *Add to Output*.

Alternatively, you can drag and drop the entire *Join_1* data source, and then, to remove the two columns that are not needed in the output, select them, right-click the selection and choose *Remove Output Columns*.

12. 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.

13. In the *Semantics* node, rename the Client column as **CLIENT**.



Note:

Because the *CLIENT* columns from the dimension tables have been added to the *Star Join* before defining the mapping within the *Star Join* node, the *CLIENT* column originating from the *Join_1* node has been renamed *JOIN_2* or something equivalent in a previous step (column mapping in the *Star Join* node).

a) In the *Semantics* node, in the *Columns* tab, modify the *Name* and *Label* fields for the *CLIENT_2* column to *CLIENT*.

14. Save, and then build the *CVCS_PO* calculation view.

Check the build status.

- a) Click Save.
 - b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build completes successfully.
15. Preview the data of the *CVCS_PO* calculation view.
- a) In the workspace, right-click the *CVCS_PO* calculation view and choose *Data Preview*.
Alternatively, make sure the focus is on the *Editor* tab for your *CVCS_PO* view and choose *Edit* → *Data Preview*.
 - b) To close the data preview, choose *X (Close)*.
16. Close all the open tabs in the SAP Web IDE for SAP HANA.
- a) Right-click any open tab and choose *Close All*.

Unit 2

Exercise 9

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

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 the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs*.
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*, is generated.

Property	Value
Aggregation	<i>Row</i>
Result Set Direction	<i>Top</i>
Result Set Type	<i>Absolute</i>
Target Value	<i>Fixed</i> Value: 5
Offset	<i>Fixed</i> 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: <i>Descending</i>)

7. In the Aggregation node, add all columns to the output.

8. Save, and then build the CVC_SO_RANK calculation view.
Check the build 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

**Note:**

The following intermediate solution object is available at this stage:
solutions → UNIT_2 → CVC_SO_RANK_00_STAGE1

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. Save and build the CVC_SO_RANK calculation view.
Check the build 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?



Note:

The following intermediate solution object is available at this stage:

solutions → UNIT_2 → CVC_SO_RANK_00_STAGE2.

5. Modify the *CVCS_SO_RANK* calculation view, adding the *BP_COMPANY_NAME* column to the *Logical Partition* of the Rank node. Then, save and build the calculation view.
6. Preview the data of the *CVC_SO_RANK* calculation view.

How does the result set look?

7. 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?

8. Modify the view CVC_SO_RANK to activate the *Dynamic Partition Elements* in the properties of the *Rank_1* node. Save and build the calculation view.
9. Preview the data of the CVC_SO_RANK calculation view. You get the same result, partitioned by *COUNTRY* and *BP_COMPANY_NAME*.
10. 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?



Note:

The following intermediate solution object is available at this stage:

solutions → *UNIT_2* → *CVC_SO_RANK_00_STAGE3*.

11. Close all the open tabs in the SAP Web IDE.

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

Property	Value
Target Value	Fixed Value: 0.1

4. Save and build the CVC_SO_RANK calculation view.

Check the build status after activation.

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. Preview 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. In the definition of the CVCS_SO_RANK calculation view, sort the result set of the calculation view by COUNTRY (asc) and RANK (asc).

9. Save and build the CVC_SO_RANK calculation view.
Check the build status after activation.
10. Preview the CVCS_SO_RANK calculation view and confirm that the result set is now logically ordered.



Note:

The following intermediate solution object is available at this stage:

solutions → UNIT_2 → CVC_SO_RANK_00_STAGE4.

11. 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 CSC_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	<i>Sum</i>
Target Value	<i>Fixed</i> <i>Value: 0.01</i>
Sort Column	<i>GROSS_AMOUNT</i> (direction: <i>Ascending</i>)



Note:

Keep the *Result Set Type* setting *Percentage*.

2. Save and build the CVC_SO_RANK calculation view.
Check the build status after activation.
3. Preview the Calculation View. Check that the result subset for COUNTRY = 'CA' is consistent.

How many rows do you get for Canada?



Note:

The following intermediate solution object is available at this stage:

solutions → UNIT_2 → CVC_SO_RANK_00_STAGE5

4. (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?

5. (Optional) Adjust the Definition of the Rank node as follows (the table only mentions the properties you need to modify).

Property	Value
Aggregation Function	<i>Sum</i>
Result Set Direction	<i>Down</i>
Sort Column	<i>GROSS_AMOUNT</i> (direction: <i>Descending</i>)

6. (Optional) Save and build the *CVC_SO_RANK* calculation view.

Check the build status after activation.

7. (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?

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

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 the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on with the user name **STUDENT##** and the password **Training1** (where ## is your group number).
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 workspace, expand your *HA300_##* project to display the folder *HDB → src → resources*.

- b) Right-click the calculation view CVCS_SO and choose *Data Preview..*

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 *Workspace* tree, choose *HA300_## → HDB → src* and right-click the *exercises* folder. Choose *New → Calculation View*.
- b) Enter the calculation view name and other properties as specified in the table.
- c) Choose *Create*.
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, drag and drop a Rank node from the palette to the area below the Aggregation node.
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 **cvcso_so**.
- c) In the search results, select the view *HA300::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*, is generated.

Property	Value
Aggregation	<i>Row</i>
Result Set Direction	<i>Top</i>
Result Set Type	<i>Absolute</i>
Target Value	<i>Fixed</i> <i>Value: 5</i>
Offset	<i>Fixed</i> <i>Value: [Leave Blank]</i>
Generate Rank Column	[Selected]
Generated rank column name	RANK (do not keep the default column name)
Partition column	<i>COUNTRY</i>
Sort Column	<i>GROSS_AMOUNT</i> (direction: <i>Descending</i>)

- 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. Save, and then build the *CVC_SO_RANK* calculation view.
Check the build status after activation.
- a) Choose *Save*.
 - b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed 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 in the preferences of the Web IDE (*SQL Console* section).

In other scenarios, you might need to pass a filter to the SQL query by choosing *Add Filter*. The query will then be re-executed against the calculation view.

**Note:**

The following intermediate solution object is available at this stage:

solutions → *UNIT_2* → *CVC_SO_RANK_00_STAGE1*

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. Save and build the *CVC_SO_RANK* calculation view.
Check the build status after activation.
 - a) Choose *Save*.
 - b) Choose *Build → Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed 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  **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.



Note:

The following intermediate solution object is available at this stage:

solutions → UNIT_2 → CVC_SO_RANK_00_STAGE2.

5. Modify the **CVCS_SO_RANK** calculation view, adding the **BP_COMPANY_NAME** column to the *Logical Partition* of the **Rank** node. Then, save and build 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 **Save**.
 - d) Choose **Build → Build Selected Files**.
 - e) In the *Console* pane, check that the build was completed successfully.
6. 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.

7. 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.

8. Modify the view *CVC_SO_RANK* to activate the *Dynamic Partition Elements* in the properties of the *Rank_1* node. Save and build 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 *Save*.
 - d) Right-click the *CVC_SO_RANK* calculation view and choose *Build Selected Files*.
 - e) In the *Console* pane, check that the build was completed successfully.
9. 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.
10. 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.



Note:

The following intermediate solution object is available at this stage:
solutions → UNIT_2 → CVC_SO_RANK_00_STAGE3.

11. Close all the open tabs in the SAP Web IDE.
 - 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 workspace, double-click 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	<i>Percentage</i>
Target Value	<i>Fixed</i> Value: 0 . 1

- a) Select the *Rank_1* node.

- b) In the *Definition* tab, adjust the properties as per the table.
4. Save and build the CVC_SO_RANK calculation view.
Check the build status after activation.
- Choose Save.
 - Choose *Build* → *Build Selected Files*.
 - In the *Console* pane, check that the build was completed successfully.
5. Preview the intermediate result set of the *Rank_1* node.
- In the *Scenario* pane, right-click the *Rank_1* node and choose *Data Preview*.
 - 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'.
- Choose the arrow at the right of the COUNTRY column header.
 - 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. Preview the CVCS_SO_RANK calculation view.
- Right-click the CVC_SO_RANK calculation view and choose *Data Preview*.
 - Display the *Raw Data* tab.

What do you notice regarding the result set order?

The result set is ordered by PRODUCT_ID.

What would you suggest to keep the result set ordered logically?

It is possible to add sort criteria in the Semantics.

8. 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.

9. Save and build the *CVC_SO_RANK* calculation view.

Check the build status after activation.

a) Choose *Save*.

b) Choose *Build → Build Selected Files*.

c) In the *Console* pane, check that the build was completed successfully.

10. 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.



Note:

The following intermediate solution object is available at this stage:

solutions → UNIT_2 → CVC_SO_RANK_00_STAGE4.

11. Close all open tabs except the *Calculation View editor* tab for the *CSC_SO_RANK* calculation view.

a) Right-click the tab *CVC_SO_RANK.hdbculationview* 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, `.hdbcaculationview` — 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 `CSC_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	<i>Sum</i>
Target Value	<i>Fixed</i> Value: 0.01
Sort Column	<i>GROSS_AMOUNT</i> (direction: <i>Ascending</i>)

**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. Save and build the `CVC_SO_RANK` calculation view.
Check the build status after activation.
 - a) Choose *Save*.
 - b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed successfully.
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.



Note:

The following intermediate solution object is available at this stage:

solutions → UNIT_2 → CVC_SO_RANK_00_STAGE5

4. (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.

5. (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.
6. (Optional) Save and build the CVC_SO_RANK calculation view.
Check the build status after activation.
 - a) Choose Save.
 - b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed successfully.
7. (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 3

Exercise 10

Create Restricted and Calculated Columns

Exercise Objectives

After completing this exercise, you will be able to:

- Define Restricted Columns
- 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 restricted and calculated 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
- Task 3: Create Calculated Columns for the Sales Percentage by Product Type

Task 1: Create a New Calculation View

1. If needed, launch the SAP HANA Web IDE. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
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 schema TRAINING to the Join_1 node:

- SALES_DATA (synonym: HA300::SALES_DATA)
 - PRODUCT (synonym: HA300:PRODUCT)
5. In your *Join_1* node, create a join between the tables using the following data:

Field	Value
Left Table	HA300::SALES_DATA
Right Table	HA300:PRODUCT
Join Columns	PRODUCT_ID = PRODUCT_ID
Join Type	Left Outer
Cardinality	n..1

**Hint:**

If the list of columns does not show up in the *Join Definition* tab, select another node, and then select the *Join_1* node again.

6. In the *Join_1* node, add the following columns to output:
- HA300::SALES_DATA.CURRENCY
 - HA300::PRODUCT.PRODUCT_TEXT
 - HA300::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. Save, and then build the calculation view.

Check the build 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 4: Restrictions

Column	Operator	Value
PRODUCT_TEXT	EQUAL	[Fixed Type]: Keyboard
PRODUCT_TEXT	EQUAL	[Fixed Type]: 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*, and, from the *Language* dropdown, select *SQL*.

The SQL expression can be written as:

```
"PRODUCT_TEXT" = 'Laptop' OR "PRODUCT_TEXT" LIKE '%station'
```

4. Save, and then build the calculation view.

Check the build 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* screen.



Note:

The following intermediate solution object is available at this stage:

solutions → *UNIT_3* → *CVC_SALES_ANALYSIS_RESTR_00_STAGE1*

Task 3: Create Calculated Columns for the Sales Percentage by Product Type

You will now create two calculated columns in the *Aggregation* node.

- In the Aggregation node, add a calculated column for the input device sales share to the *CV_PRODUCT_SALES_ANALYSIS* calculation view 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
Data Type	<ul style="list-style-type: none"> DECIMAL Length: 4 Scale: 2
Column Type	Measure

- Add the following expression (SQL Language) 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 at the bottom of the screen.

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.

- 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	<ul style="list-style-type: none"> DECIMAL Length: 4 Scale: 2
Column Type	Measure

- Add the following expression (SQL language) to the calculated column and validate the syntax:

```
"COMPUTER_SALES" / "TOTAL_SALES"
```

- Save, and then build the calculation view.

Check the build status.

6. 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)

The screenshot shows the SAP Web IDE interface with the Analysis tab selected. On the left, there are two sections: 'Label Axis' containing 'CURRENCY' and 'Value Axis' containing several measures like 'TOTAL_SALES (SUM)', 'INPUT_DEVICE_SALE...', 'COMPUTER_SALES...', etc. The main area displays a table with two rows of data. Row 1 shows EUR currency with values: TOTAL_SALES (12782.7), INPUT_DEV_ (380), COMPUTER_SALES (9266.4), and COMPUTER_SAL_ (0.029727678815899616). Row 2 shows USD currency with values: TOTAL_SALES (16913), INPUT_DEV_ (576), COMPUTER_SALES (11387.4), and COMPUTER_SAL_ (0.03405664281913321). The table has a header row with columns labeled 'CURRE..', 'TOTAL_SALES', 'INPUT_DEV..', 'COMPUTER...'. A legend at the top right indicates that blue represents currency and green represents measures.

Figure 10: Data Preview (Table Display)

7. Close all open tabs in the SAP Web IDE.

Unit 3

Solution 10

Create Restricted and Calculated Columns

Exercise Objectives

After completing this exercise, you will be able to:

- Define Restricted Columns
- 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 restricted and calculated 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
- Task 3: Create Calculated Columns for the Sales Percentage by Product Type

Task 1: Create a New Calculation View

1. If needed, launch the SAP HANA Web IDE. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on using the user name **STUDENT##** and the password **Training1** (where ## is your group number).
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

Field	Value
With Star Join	[Deselected]

- a) In the *Workspace* tree, choose *HA300 → HDB → src* and right-click the *exercises* folder. Choose *New → Calculation View*.
- b) Enter the calculation view name and other properties as specified in the table.
- c) Choose *Create*.
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, drag a new *Join* node below the *Aggregation* node.
The new join node is automatically named *Join_1*.
- b) Select the *Join_1* node and drag the arrow icon to the *Aggregation_1* node.
4. Add the following tables from the schema *TRAINING* to the *Join_1* node:
- *SALES_DATA* (synonym: *HA300::SALES_DATA*)
 - *PRODUCT* (synonym: *HA300: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 *Searchfield* and enter **SALES**.
- c) Select the *SALES_DATA* (synonym: *HA300::SALES_DATA*) table and choose *Finish*.
- d) Choose *Finish*.
- e) 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	<i>HA300::SALES_DATA</i>
Right Table	<i>HA300:PRODUCT</i>
Join Columns	<i>PRODUCT_ID = PRODUCT_ID</i>
Join Type	<i>Left Outer</i>
Cardinality	<i>n..1</i>



Hint:

If the list of columns does not show up in the *Join Definition* tab, select another node, and then select the *Join_1* node again.

- a) In the *Join Definition* tab of the *Join_1* node, drag a connecting line between the columns *HA300::SALES_DATA.PRODUCT_ID* and *HA300::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:

- HA300::SALES_DATA.CURRENCY
- HA300::PRODUCT.PRODUCT_TEXT
- HA300::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. Save, and then build the calculation view.

Check the build status after activation.

a) Choose *Save*.

b) Choose *Build* → *Build Selected Files*.

c) In the *Console* pane, check that the build was completed 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 4: Restrictions

Column	Operator	Value
PRODUCT_TEXT	EQUAL	[Fixed Type]: Keyboard
PRODUCT_TEXT	EQUAL	[Fixed Type]: Mouse

**Note:**

To assign value text for each of the restrictions, choose *Open Search Help* (or press **F4**) in the *Value* field.

- Select the *Aggregation* node and display the *Restricted Columns* tab.
- Choose +.
- Enter the data as shown in the table, both for the *GENERAL* and *RESTRICTIONS* areas.

**Hint:**

You must add two restrictions.

- 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

- Select the *Aggregation* node and display the *Restricted Columns* tab.
 - Choose +.
 - In the *GENERAL* area, enter the data as shown in the table.
- 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*, and, from the *Language* dropdown, select *SQL*.

The SQL expression can be written as:

```
"PRODUCT_TEXT" = 'Laptop' OR "PRODUCT_TEXT" LIKE '%station'
```

- In *Restrictions*, select *Expression*.

- b) In the *Language* drop-down, select SQL.
- c) Enter the expression.
4. Save, and then build the calculation view.
Check the build status after activation.
- a) Choose Save.
- b) Choose *Build* → *Build Selected Files*.
- c) In the *Console* pane, check that the build was completed successfully.
5. Preview the data of the calculation view and check the behavior of the two restricted columns you have defined.
- a) Choose *Edit* → *Data Preview*.
- b) In the *Raw Data* tab, check that the values of the two restricted columns are consistent for each product.
6. Close the *Data Preview* screen.

**Note:**

The following intermediate solution object is available at this stage:

solutions → *UNIT_3* → *CVC_SALES_ANALYSIS_RESTR_00_STAGE1*

Task 3: Create Calculated Columns for the Sales Percentage by Product Type

You will now create two calculated columns in the *Aggregation* node.

1. In the *Aggregation* node, add a calculated column for the input device sales share to the *CV_PRODUCT_SALES_ANALYSIS* calculation view 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
Data Type	<ul style="list-style-type: none"> • 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) In the *Calculated Column* dialog box, enter the data as shown in the table.

2. Add the following expression (SQL Language) 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 at the bottom of the screen.

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) On the *Expression* tab, check that *SQL* is selected in the language drop-down list.

- b) Enter the listed expression.

Alternatively, choose *Expression Editor* and double-click the column names to select them from the *Elements* list.

- c) Choose *Validate Syntax*.

3. 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	<ul style="list-style-type: none"> • 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.

4. Add the following expression (*SQL* language) to the calculated column and validate the syntax:

```
"COMPUTER_SALES" / "TOTAL_SALES"
```

- a) In the *Expression* tab, check that *SQL* is selected in the language drop-down list.

- b) Enter the listed expression.

Alternatively, choose *Expression Editor* and double-click the column names to select them from the *Elements* list.

- c) Choose *Validate Syntax*.

5. Save, and then build the calculation view.

Check the build status.

- a) Choose Save.
 - b) Choose *Build* → *Build Selected Files*
 - c) In the *Console* pane, check that the build was completed successfully.
6. 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)

	RB_CURRE..	TOTAL_SALES	INPUT_DEV..	COMPU..	INPUT_DEVICE_S..	COMPUTER_SAL..
1	EUR	12782.7	380	9266.4	0.029727678815899616	0.7249172709990847
2	USD	16913	576	11387.4	0.03405664281913321	0.6732927334003429

Figure 10: Data Preview (Table Display)

- a) Choose *Edit* → *Data Preview*.
 - b) Select the *Analysis* tab.
 - c) Right-click the *CURRENCY* attribute and choose *Add to Labels Axis*.
 - d) To select all the measures, select the first one, and then press **Shift** while you select the last one.
 - e) Right-click the selection and choose *Add to Values Axis*.
7. Close all open tabs in the SAP Web IDE.
- a) Right-click any open tab and choose *Close All*.

Unit 3

Exercise 11

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.

Task 1: Import a Calculation View and Filter its Data

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
2. Import a new calculation view CVC_SALES_FILTER into the src → exercises folder of your HA300_## project.
You can find the import file in your course files: HA300 → Calculation View Templates → CVC_SALES_FILTER.hdbculationview.
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. Save, and then build the calculation view.
Check the build 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. Save, and then build the calculation view.
Check the build 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). Save and build the calculation view and preview the results.



Hint:

Use the `LIKE` predicate in the filter expression.

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. Save and build 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. Modify the calculation view nodes so that data can be filtered AFTER aggregation. Save and build 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?

4. 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 → *UNIT3* → → *CVC_SALES_FILTER_00_STAGE2*

5. How could you force the aggregation of the QUANTITY measure by *CUSTOMER_ID* only?

6. 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?

7. Close all open tabs in the SAP Web IDE.

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.

Task 1: Import a Calculation View and Filter its Data

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
 - a) Start Windows Explorer and navigate to the folder HA300 → URLs.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on using **STUDENT##** as the user name and **Training1** as the password (where ## is your group number).
2. Import a new calculation view **CVC_SALES_FILTER** into the *src → exercises* folder of your **HA300##** project.
You can find the import file in your course files: *HA300 → Calculation View Templates → CVC_SALES_FILTER.hdbculationview*.
 - a) In the workspace, right-click your *exercises* folder and choose *import → From File System*.
 - b) In the *Import* window, choose *Browse*.
 - c) Select the *HA300 → Calculation Views Templates → CVC_SALES_FILTER.hdbculationview* file and choose *Open*.

- d) Make sure the *Import to* location ends with .../exercises.
- e) Choose *OK*.
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 HA300::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. Save, and then build the calculation view.
Check the build status after activation.
- a) Choose *Save*.
- b) Choose *Build → Build Selected Files*.
- c) In the *Console* pane, check that the build was completed 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.
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'". Check that the *Language* field is set to SQL, which is the default.
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. Save, and then build the calculation view.
Check the build status after activation.
- a) Choose *Save*.
- b) Choose *Build → Build Selected Files*.

- c) In the *Console* pane, check that the build was completed 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.
 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). Save and build 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 *Save*.
- d) Choose *Build → Build Selected Files*.
- e) In the *Console* pane, check that the build was completed successfully.
- f) Right-click the *CVC_SALES_FILTER* Calculation View and choose *Data Preview*.
- g) Select the *Raw Data* tab and review the contents.



Note:

An intermediate solution object is available at this stage:

`solutions → UNIT3 → → 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. Save and build the calculation view and preview the results.
 - a) Select the *Aggregation* node, and in the *Filter Expression* area, enter "`QUANTITY` \geq 10

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 HA300::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

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 **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 "HA300_##_HDI_HDB_1"."HA300::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. Modify the calculation view nodes so that data can be filtered AFTER aggregation. Save and build 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 green and 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 *Save*.
- f) Choose *Build → Build Selected Files*.
- g) In the *Console* pane, check that the build was completed successfully.
- h) Right-click the *CVC_SALES_FILTER* Calculation View and choose *Data Preview*.
- i) 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.

4. Again, modify the query to remove the *PRODUCT_ID* column.

a) From the last data preview, choose *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 "HA300_##_HDI_HDB_1"."HA300::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 → UNIT3 → → CVC_SALES_FILTER_00_STAGE2

5. 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.

6. Implement this approach in your CVC_SALES_FILTER calculation view. Save and build the calculation view and preview the results.

- a) In the Calculation View editor, select the Aggregation_1 node.
- b) In the Mapping tab, right-click the PRODUCT_ID column in the Output Columns area and choose Remove Output Column.
- c) To confirm the column removal, choose Yes.
- d) In the Aggregation node, check whether the PRODUCT_ID has been removed. If not, repeat the same step.
- e) Choose Save.
- f) Choose Build → Build Selected Files.
- g) In the Console pane, check that the build was completed 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).

7. Close all open tabs in the SAP Web IDE.

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

Unit 3

Exercise 12

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 ##, replace the characters with the number your instructor assigned to you.

Task 1: Import a Calculation View and Preview its Data

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
2. Import a new calculation view CVD_CITIES_CLIENT into the src → exercises folder of your HA300_## project.
You can find the import file in your course files: HA300 → Calculation View Templates → CVD_CITIES_CLIENT.hdbculationview.
3. Review quickly the design of the Calculation View, in particular the *Client Column* setting for the data source HA300::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. Save, and then build the calculation view.
Check the build status after activation.
5. Preview the calculation view data in the *Raw Data* tab.
The column *MANDT* contains the SAP Client number.

**Note:**

An intermediate solution object is available at this stage. This can be found at *solutions* → *UNIT_3* → *CVD_CITIES_CLIENT_00_STAGE1*.

Task 2: Modify the Calculation View to Make it Client-Dependent

1. Define the *MANDT* from the data source *HA300::CITIES* as the Client Column.
2. Save, and then build the calculation view.
Check the build status after activation.
3. Preview the calculation view data in the *Raw Data* tab.

What do you observe? Can you explain why?

4. Modify your calculation view to display only data for SAP Client number **200**.
5. Save, and then build the calculation view.
Check the build status after activation.
6. 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* → *UNIT_3* → *CVD_CITIES_CLIENT_00_STAGE2*.

**Note:**

In the users table of the SAP HANA database, your user *STUDENT##* 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.

In the Data Preview, how many distinct values of the field *CITY* do you see?

7. Modify your calculation view by changing the *Default Client* property to *Session Client*.

8. Save, and then build the calculation view.
Check the build status after activation.
9. Preview the calculation view data in the *Raw Data* tab.

What do you observe? Can you explain why?

Can you explain why?

To be able to actually display the data filtered by the default client of your *STUDENT##* user, you will display the data of the calculation view from an SQL console connected to the database *H00 DB*. In this context, the actual user executing the query will be your *STUDENT##* user.

Alternatively, you could also query the data from an external tool such as SAP BusinessObjects Analysis for MS Excel.



Note:

In the users table of the SAP HANA database, your user *STUDENT##* has been assigned a default client value of 800.

10. Preview the data of the Calculation View *CVD_CITIES_CLIENT* from the *H00 DB* 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?

11. Close all open tabs in the SAP Web IDE.

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 ##, replace the characters with the number your instructor assigned to you.

Task 1: Import a Calculation View and Preview its Data

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
 - a) Start Windows Explorer and navigate to the folder HA300 → URLs.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on using **STUDENT##** as the user name and **Training1** as the password (where ## is your group number).
2. Import a new calculation view **CVD_CITIES_CLIENT** into the **src → exercises** folder of your **HA300_##** project.

You can find the import file in your course files: *HA300 → Calculation View Templates → CVD_CITIES_CLIENT.hdbcalculationview*.

a) In the workspace, right-click your exercises folder and choose *import → From File System*.

b) In the *Import* window, choose *Browse*.

c) Select the *HA300 → Calculation Views Templates → CVD_CITIES_CLIENT.hdbcalculationview* file and choose *Open*.

d) Make sure the *Import to* location ends with .../exercises.

e) Choose *OK*.

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 *HA300::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 *HA300::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 *HA300::CITIES* and choose *Data Preview*.

4. Save, and then build the calculation view.

Check the build status after activation.

a) Choose *Save*.

b) Choose *Build → Build Selected Files*.

c) In the *Console* pane, check that the build was completed successfully.

5. Preview the calculation view data in the *Raw Data* tab.

The column *MANDT* contains the SAP Client number.



Note:

An intermediate solution object is available at this stage. This can be found at *solutions → UNIT_3 → 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?

20

c) In the *Projection_1* node, display the *Mapping* tab.

d) Select the Data Source *HA300::CITIES*.

e) In the *Properties* pane, notice that the *Client Column* property is empty.

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 HA300::CITIES.

Task 2: Modify the Calculation View to Make it Client-Dependent

1. Define the *MANDT* from the data source HA300::CITIES as the Client Column.
 - a) In the *Projection_1* node, display the *Mapping* tab.
 - b) Select the Data Source HA300::CITIES.
 - c) In the *Properties* pane, set the *Client Column* property to *MANDT*.
2. Save, and then build the calculation view.
Check the build status after activation.
 - a) Choose *Save*.
 - b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed successfully.
3. Preview the calculation view data in 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.

- a) Choose *Edit* → *Data Preview*.
- b) Select the *Raw Data* tab.
4. 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**.
5. Save, and then build the calculation view.
Check the build status after activation.
 - a) Choose *Save*.
 - b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed successfully.
6. 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 → UNIT_3 → CVD_CITIES_CLIENT_00_STAGE2*.



Note:

In the users table of the SAP HANA database, your user STUDENT## 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.

- a) 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

7. 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*.
8. Save, and then build the calculation view.
Check the build status after activation.
 - a) Choose *Save*.
 - b) Choose *Build → Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed successfully.
9. Preview the calculation view data in the *Raw Data* tab.
 - a) Choose *Edit → Data Preview*.
 - b) Select the *Raw Data* tab.

What do you observe? Can you explain why?

The view does not retrieve any rows.

Can you explain why?

This is because, in the SAP Web IDE, the actual user executing the data preview is a technical user (not your STUDENT## user) who has no default client number assigned.

To be able to actually display the data filtered by the default client of your STUDENT## user, you will display the data of the calculation view from an SQL console connected to

the database *H00 DB*. In this context, the actual user executing the query will be your *STUDENT##* user.

Alternatively, you could also query the data from an external tool such as SAP BusinessObjects Analysis for MS Excel.



Note:

In the users table of the SAP HANA database, your user *STUDENT##* has been assigned a default client value of 800.

10. Preview the data of the Calculation View *CVD_CITIES_CLIENT* from the *H00 DB* 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) Choose *Tools* → *Database Explorer*
- b) In the Database/Containers list on the left, select the *H00 DB* entry.
- c) Navigate the tree to display *H00 DB* → *Catalog* → *Column Views*.
- d) In the bottom left pane, choose *Choose schema*.
- e) Enter **HA300_##** (## is your assigned group number), select carefully your schema **HA300_##_HDI_HDB_1** and choose **OK**.
- f) In the *Search Column Views* field, enter **CITIES**.
- g) In the search results, right-click the column view **HA300::CVD_CITIES_CLIENT** and choose **Open Data**.
- h) Display the *Raw Data* tab.

What do you observe?

By querying the data with the actual *STUDENT##* user, the data is now filtered by the default client number assigned to this user; that is, 800.

11. Close all open tabs in the SAP Web IDE.
- a) Right-click any open tab and choose *Close All*.

Unit 3

Exercise 13

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.

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 the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
2. In the workspace, import the CVC_CAMPAIGN_ANALYSIS.hdbcCalculationView file from your course files folder HA300 → Calculation Views Templates into the folder HA300_## → HDB → src → exercises.
3. Review the design of the CVC_CAMPAIGN_ANALYSIS calculation view.
4. Build the CVC_CAMPAIGN_ANALYSIS calculation view.
Check the build 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:

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

2. Save, and then build the view.
Check the build status after activation.



Note:

The following intermediate solution object is available at this stage:

solutions → UNIT_3 → CVC_CAMPAIGN_ANALYSIS_STAGE1

3. Preview the calculation view data in the *Raw Data* tab.
4. 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.

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.

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 the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on with the user name **STUDENT##** and the password **Training1** (where ## is your group number).
2. In the workspace, import the *CVC_CAMPAIGN_ANALYSIS.hdbculationview* file from your course files folder *HA300 → Calculation Views Templates* into the folder *HA300_## → HDB → src → exercises*.
 - a) In the workspace, right-click your *exercises* folder and choose *import → File or Project*.
 - b) In the *Import* window, choose *Browse*.
 - c) Select the file *HA300 → Calculation Views Templates → CVC_CAMPAIGN_ANALYSIS.hdbculationview* and choose *Open*.
 - d) Make sure that the *Import to* location ends with *.../exercises*.
 - e) Choose *OK*.
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. Build the *CVC_CAMPAIGN_ANALYSIS* calculation view.
Check the build status.
 - a) Choose *Build → Build Selected Files*.
 - b) In the *Console* pane, check that the build was completed successfully.

5. Preview the calculation view data in the *Raw Data* tab.
- a) Choose *Edit → Data Preview*.

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:

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

- a) In the Semantics node, select the *Parameters* tab.
- b) Choose **+** and choose *Variable*.
- c) Enter the details as shown in the table.
2. Save, and then build the view.
Check the build status after activation.
- a) Choose *Save*.
- b) In the Workspace tree, navigate to the folder *HA300 → HDB → src → exercises*.
- c) Right-click the calculation view and choose *Build Selected Files*.
- d) In the *Console* pane, check that the build was completed successfully.



Note:

The following intermediate solution object is available at this stage:

solutions → UNIT_3 → CVC_CAMPAIGN_ANALYSIS_STAGE1

3. Preview the calculation view data in the *Raw Data* tab.
- a) Right-click the *CVC_CAMPAIGN_ANALYSIS* calculation view.
- b) Choose *Data Preview*.
4. 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) In the *Variables and Input Parameters* screen, for the VAR_CUSTOMER variable, choose *From* and then choose the search help radio button.
- b) From the list, select *Becker Berlin*.
- c) Choose *OK*.
- d) Choose  *Open Content* and select the *Raw Data* tab.
- e) Check that the output includes the specified customer only.

Unit 3

Exercise 14

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.

Task 1: Delete the Variable CUSTOMER and Add a New Column from the Source Table to the Output

The new columns will be used later on to calculate the number of days elapsed since the last marketing campaigns.

1. Delete the variable `VAR_CUSTOMER`, which will no longer be used because you want to visualize data for all the customers.
2. In the `Join_1` node, add the column `SALES_DATA.SQL_DATE` to the output.
3. Add the new column to the `Aggregation` node.
4. In the output of the `Aggregation` node, rename the column `SQL_DATE` as `SALE_DATE`.
5. 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.

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
Data Type	Date
List of values	<ul style="list-style-type: none"> <i>Name:</i> 2019-01-25 <i>Label:</i> First e-mail <i>Name:</i> 2019-02-23 <i>Label:</i> Second e-mail
Default value(s)	[none]



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
Name	DAYS_ELAPSED
Description	Days between Sale Date and e-mail
Data Type	INTEGER
Column Type	Attribute
Expression Language	SQL
Expression	DAYs_BETWEEN("SALE_DATE", '\$\$INP_EMAIL_DATE\$\$')



Note:

The expression may not validate.

3. Save, and then build the calculation view.
Check the build status after activation.



Note:

The following intermediate solution object is available at this stage:

solutions → UNIT_3 → CVC_CAMPAIGN_ANALYSIS_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?

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

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
Default Value(s)	Constant: 2019-01-25

Keep the other parameters as is.

2. Save, and then build the calculation view.
Check the build status after activation.
3. Preview the calculation view data on the *Raw Data* tab.
4. In the *Variables and Input Parameters* screen, keep the default value or enter a different date.



Caution:

Because there is no list of values for the input parameter, do not choose the *Value Help* radio button.

5. In the displayed data, check that the number of elapsed days is correctly calculated based on the input parameter you defined.

6. Close all open tabs in the SAP Web IDE.

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.

Task 1: Delete the Variable CUSTOMER and Add a New Column from the Source Table to the Output

The new columns will be used later on to calculate the number of days elapsed since the last marketing campaigns.

1. 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*.
2. 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, right-click the column *SALES_DATA.SQL_DATE* and choose *Add to Output*.
3. Add the new column to the *Aggregation* node.
 - a) In the *Scenario* tab, select the *Aggregation* node.
 - b) On the *Mapping* tab, right-click the column *SQL_DATE* and select *Add to Output* from the context menu.
4. In the output of the *Aggregation* node, rename the column *SQL_DATE* as ***SALE_DATE***.
 - a) In the *Columns* tab of the *Aggregation* node, change the name and label of the column *SQL_DATE* to ***SALE_DATE***.
5. In the *Semantics* node, check that the new column has the type *Attribute*.
 - a) In the *Scenario* tab, select the *Semantics* node.
 - b) 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.

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
Data Type	Date
List of values	<ul style="list-style-type: none"> • Name: 2019-01-25 Label: First e-mail • Name: 2019-02-23 Label: Second e-mail
Default value(s)	[none]



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) Enter the details as shown in the table.
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
Name	DAYs_ELAPSED
Description	Days between Sale Date and e-mail
Data Type	INTEGER
Column Type	<i>Attribute</i>
Expression Language	SQL
Expression	DAYs_BETWEEN("SALE_DATE", '\$\$INP_EMAIL_DATE\$\$')

**Note:**

The expression may not validate.

- a) On the *Calculated Column* tab of the Aggregation node, choose + and then choose *Create Calculated Column*.
- b) Enter the details as shown in the table.
- c) Choose *BACK*.
3. Save, and then build the calculation view.
Check the build status after activation.
 - a) Choose *Save*.
 - b) In the Workspace tree, navigate to the folder *HA300 → HDB → src → exercises*.
 - c) Right-click the calculation view and choose *Build Selected Files*.
 - d) In the *Console* pane, check that the build was completed successfully.

**Note:**

The following intermediate solution object is available at this stage:

solutions → UNIT_3 → CVC_CAMPAIGN_ANALYSIS_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) 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.

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

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
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.
- c) To define the default value, select the type *Constant* and enter the value **2019-01-25**.
2. Save, and then build the calculation view.
Check the build status after activation.
 - a) Choose Save.
 - b) In the Workspace tree, navigate to the folder *HA300 → HDB → src → exercises*.
 - c) Choose *Raw Data*.
 - d) Choose *Build → Build Selected Files*.
 - e) In the *Console* pane, check that the build was completed successfully.
3. Preview the calculation view data on the *Raw Data* tab.
 - a) Choose *Edit → Data Preview*.

4. In the *Variables and Input Parameters* screen, keep the default value or enter a different date.



Caution:

Because there is no list of values for the input parameter, do not choose the *Value Help* radio button.

- a) For your input parameter *INP_EMAIL_DATE*, keep the default value or enter a different date; for example, **2020-01-01**.
- b) Choose *Open Content*.
5. In the displayed data, check that the number of elapsed days is correctly calculated based on the input parameter you defined.
6. Close all open tabs in the SAP Web IDE.
 - a) Right-click any open tab and choose *Close All*.

Unit 3

Exercise 15

Implement Value Help Views

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, but you could also use Analytic Privileges to implement a dynamic restriction based on a user's permissions.

1. Create a calculation view using the data in the following table:

Field	Value
Name	CVD_COUNTRIES_EMEA_VALUE_HELP
Label	Customers in EMEA
Data Category	DIMENSION

2. Assign the table *HA300::COUNTRIES* as the data source to the default projection node.
3. In the projection node, map all columns to the output.
4. Define a filter expression to restrict countries to those in region *EMEA*.
The resulting expression should look exactly like this : "REGION"='EMEA'.
5. Save and build the calculation view.

Task 2: Consume the External Value Help View

We will now create a calculation view that prompts the user to choose a customer. However, the choices of customers will be restricted to those that are returned from the external value help view. In our case, *EMEA*.

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 *HA300::CUSTOMER* 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	HA300::CVD_COUNTRIES_EMEA_VALUE_HELP
Reference Column	HA300::CVD_COUNTRIES_EMEA_VALUE_HELP.COUNTRY

5. Save and build the calculation view.
6. Preview the results to ensure that your value help for country only returns the countries in EMEA.

Implement Value Help Views

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, but you could also use Analytic Privileges to implement a dynamic restriction based on a user's permissions.

1. Create a calculation view using the data in the following table:

Field	Value
Name	CVD_COUNTRIES_EMEA_VALUE_HELP
Label	Customers in EMEA
Data Category	DIMENSION

- a) In the *Workspace* tree, choose *HA300 → HBD → src* and right-click the *exercises* folder. Choose *New → Calculation View*.
- b) Enter the calculation view name and other properties as specified in the table. Leave the namespace value as it is.
- c) Choose *Create*.
2. Assign the table *HA300::COUNTRIES* 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* (synonym: *HA300::COUNTRIES*) 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 *HA300::COUNTRIES* and choose *Add To Output*.
4. 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*. 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'.

5. Save and build the calculation view.
 - a) Choose Save
 - b) In the *Workspace* tree, right-click the *CVD_COUNTRIES_EMEA_VALUE_HELP* calculation view and choose *Build Selected Files*.

Task 2: Consume the External Value Help View

We will now create a calculation view that prompts the user to choose a customer. However, the choices of customers will be restricted to those that are returned from the external value help view. In our case, EMEA.

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 *Workspace* tree, choose *HA300 → HBD → src* and right-click the *exercises* folder. Choose *New → Calculation View*.
 - b) Enter the calculation view name and other properties as specified in the table. Leave the namespace value as it is.
 - c) Choose *Create*.
2. Assign the table *HA300::CUSTOMER* 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* (synonym: *HA300::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 *HA300::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	HA300::CVD_COUNTRIES_EMEA_VALUE_HELP

Reference Column	HA300::CVD_COUNTRIES_EMEA_VALUE_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) To provide an empty row, choose +.
 - g) Select the *COUNTRY* column.
5. Save and build the calculation view.
- a) Choose *Save*.
 - b) In the *Workspace* tree, right-click the *CVD_CUSTOMERS_EMEA_VALUE_HELP* calculation view and choose *Build Selected Files*.
6. Preview the results to ensure that your value help for country only returns the countries in *EMEA*.
- a) In the *Workspace* tree, 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 countries you have chosen.

Unit 3

Exercise 16

Cascade User Prompts

Exercise Objectives

After completing this exercise, you will be able to:

- Create calculation views with cascading user prompts

Business Example

You have been asked to create a calculation view in which users can analyze data for a specific product, but because there are many products, you want to offer the possibility to choose a product group first, and then a product.

Task 1: Create the Value Help View

First, create the value help view that restricts the list of *products* to a user-specified *product group*.

1. Create a DIMENSION calculation view using the data in the following table:

Field	Value
Name	CVD_PRODUCTS_CASCADE_VHELP
Label	CVD_PRODUCTS_CASCADE_VHELP
Data Category	DIMENSION

2. Add a *join* node to join the tables *PRODUCT* and *PRODUCT_GROUP*, using the *PRODUCT_GROUP* column from both tables.

3. In the join node, map the columns to the output as per the following table:

Data Source	Column
HA300::PRODUCT	PRODUCT_ID
HA300::PRODUCT	PRODUCT_TEXT
HA300::PRODUCT	PRODUCT_GROUP
HA300::PRODUCT_GROUP	PRODUCT_GROUP_TEXT

4. Connect the join node to the projection node and add all columns to the output of the projection node.
5. Create a variable based on *PRODUCT_GROUP_TEXT* using the settings in the following table, so that a user-specified filter is applied to the product group text column in the view (leave all other settings with their default values):

Setting	Value
Name	VAR_VAL_HELP_PRODUCT_GRP

Setting	Value
Label	VAR_VAL_HELP_PRODUCT_GRP
Multiple Entries	check this box
Reference Column	PRODUCT_GROUP_TEXT

6. Save and build the calculation view.

Task 2: Create the Top Level View to Consume the Value Help Using Mapped Variables

1. Create a calculation view using the data in the following table:

Field	Value
Name	CVC_SALES CASCADE_TOP
Label	CVC_SALES CASCADE_TOP
Data Category	CUBE
With Star Join	[Deselected]

2. Add a *join* node to join the table *SALES_DATA* to the calculation view *CVD_PRODUCTS_CASCADE_VHELP*, using the *PRODUCT_ID* column.

3. In the join node, map the columns to the output as per the following table:

Data Source	Column
CVD_PRODUCTS_CASCADE_VHELP	PRODUCT_TEXT
CVD_PRODUCTS_CASCADE_VHELP	PRODUCT_GROUP_TEXT
HA300::SALES_DATA	QUANTITY
HA300::SALES_DATA	QTY_UNIT

4. Connect the join node to the aggregation node and add all columns to the output of the aggregation node.

5. Create a variable based on *PRODUCT_GROUP_TEXT*, using the settings in the following table below, so that a user-specified filter is applied to the product group text column in the view (leave all other settings with their default values):

Setting	Value
Name	VAR_PRODUCT_GROUP_TOP_LEVEL
Label	VAR_PRODUCT_GROUP_TOP_LEVEL
Multiple Entries	check this box
View/Table Value Help	CVD_PRODUCTS_CASCADE_VHELP
Reference Column	PRODUCT_GROUP_TEXT

6. Filter the column *PRODUCT_GROUP_TEXT* using the new variable *VAR_PRODUCT_GROUP_TOP_LEVEL*.

7. Create a variable based on *PRODUCT_TEXT*, using the settings in the following table, so that a user-specified filter is applied to the product text column in the view (leave all other settings with their default values):

Setting	Value
Name	VAR_PRODUCT_TOP_LEVEL
Label	VAR_PRODUCT_TOP_LEVEL
Multiple Entries	check this box
View/Table Value Help	CVD_PRODUCTS CASCADE_VHELP
Reference Column	PRODUCT_TEXT

8. Filter the column *PRODUCT_TEXT* using the new variable *VAR_PRODUCT_TOP_LEVEL*.
9. Map the current view variable *VAR_PRODUCT_GROUP_TOP_LEVEL* to the source variable *VAR_VAL_HELP_PRODUCT_GRP*.
10. Save and build the calculation view.
11. Check the results using *Data Preview* of Web IDE.

Cascade User Prompts

Exercise Objectives

After completing this exercise, you will be able to:

- Create calculation views with cascading user prompts

Business Example

You have been asked to create a calculation view in which users can analyze data for a specific product, but because there are many products, you want to offer the possibility to choose a product group first, and then a product.

Task 1: Create the Value Help View

First, create the value help view that restricts the list of *products* to a user-specified *product group*.

1. Create a DIMENSION calculation view using the data in the following table:

Field	Value
Name	CVD_PRODUCTS_CASCADE_VHELP
Label	CVD_PRODUCTS_CASCADE_VHELP
Data Category	DIMENSION

- a) In the *Workspace* tree, choose *HA300_## → HBD → src* and right-click the *exercises* folder. Choose *New → Calculation View*.
b) Enter the calculation view name and other properties as specified in the table (do not alter the namespace value entry).
c) Choose *Create*.
2. Add a *join* node to join the tables *PRODUCT* and *PRODUCT_GROUP*, using the *PRODUCT_GROUP* column from both tables.
 - a) From the node palette, select the *join* node and drop it to the bottom of the data flow so it is the lowest node.
 - b) Make sure the new *Join* node is selected. Then, choose *+* on the right of the node.
 - c) In the *Find Data Sources* window, select the *Search* field and enter **PRODUCT**.
 - d) In the search result, select the table *PRODUCT* and also the table *PRODUCT_GROUP*, and then choose *Finish*.
 - e) Double-click on the join node to open the properties pane on the right side of the screen.

- f) In the *Join Definition* tab, drag the column *PRODUCT_GROUP* from the left table to the *PRODUCT_GROUP* column in the right table to create the join (you may have to click outside the tables first to deselect them so you can draw the join line).

3. In the join node, map the columns to the output as per the following table:

Data Source	Column
HA300::PRODUCT	PRODUCT_ID
HA300::PRODUCT	PRODUCT_TEXT
HA300::PRODUCT	PRODUCT_GROUP
HA300::PRODUCT_GROUP	PRODUCT_GROUP_TEXT

- a) Select the *Mapping* tab of the join node.
- b) Under the *Data Sources* pane on the left, locate each column specified in the table. Right-click on each column and choose *Add To Output..*
4. Connect the join node to the projection node and add all columns to the output of the projection node.
- a) Select the join node and drag a line from the arrow symbol, which you find on the right side of the node, to the input of the projection node.
- b) Select the projection node. Under the *Mapping* tab, right-click on the data source *Join_1* in the left pane, and then select *Add to Output*.
5. Create a variable based on *PRODUCT_GROUP_TEXT* using the settings in the following table, so that a user-specified filter is applied to the product group text column in the view (leave all other settings with their default values):

Setting	Value
Name	VAR_VAL_HELP_PRODUCT_GRP
Label	VAR_VAL_HELP_PRODUCT_GRP
Multiple Entries	check this box
Reference Column	PRODUCT_GROUP_TEXT

- a) Select the *Semantic* node.
- b) Select the *Parameters* tab.
- c) To open the drop-down list, choose + and then choose *Variable*.
- d) Select the empty variable *VAR_1* and enter the settings from the table.
6. Save and build the calculation view.
- a) Choose *Save*.
- b) In the *Workspace* tree, right-click the *CVD_PRODUCTS_CASCADE_VHELP* calculation view and choose *Build* → *Build Selected Files*.

Task 2: Create the Top Level View to Consume the Value Help Using Mapped Variables

1. Create a calculation view using the data in the following table:

Field	Value
Name	CVC_SALES CASCADE_TOP
Label	CVC_SALES CASCADE_TOP
Data Category	CUBE
With Star Join	[Deselected]

- a) In the *Workspace* tree, choose *HA300_## → HBD → src* and right-click the *exercises* folder. Choose *New → Calculation View*.
 - b) Enter the calculation view name and other properties as specified in the table (do not alter the namespace value entry).
 - c) Choose *Create*.
2. Add a *join* node to join the table *SALES_DATA* to the calculation view *CVD_PRODUCTS_CASCADE_VHELP*, using the *PRODUCT_ID* column.
- a) From the node palette, select the *join* node and drop it to the bottom of the data flow so it is the lowest node.
 - b) Select the new *Join* node and choose *+* on the right of the node.
 - c) In the *Find Data Sources* window, select the *Search* field and enter **SALES DATA**.
 - d) In the search result, select the table *SALES_DATA* and choose *Finish*.
 - e) Choose *+* on the right of the node and, in the *Find Data Sources* window, select the *Search* field. Enter **CVD_PRODUCTS_CASCADE_VHELP**.
 - f) Select the calculation view *CVD_PRODUCTS_CASCADE_VHELP* and then choose *Finish*.
 - g) Double-click on the join node to open the *Properties* pane on the right side of the screen.
 - h) In the *Join Definition* tab, drag the column *PRODUCT_ID* from the left table to the column *PRODUCT_ID* in the right view to create the join (you may have to click outside the tables first to deselect them so you can draw the join line).
3. In the join node, map the columns to the output as per the following table:

Data Source	Column
CVD_PRODUCTS_CASCADE_VHELP	PRODUCT_TEXT
CVD_PRODUCTS_CASCADE_VHELP	PRODUCT_GROUP_TEXT
HA300::SALES_DATA	QUANTITY
HA300::SALES_DATA	QTY_UNIT

- a) Select the *Mapping* tab of the join node.
- b) Under the *Data Sources* pane on the left, locate each column specified in the table. Right-click on each column and choose *Add To Output*.

4. Connect the join node to the aggregation node and add all columns to the output of the aggregation node.
 - a) Select the join node and drag a line from the arrow symbol, which you find on the right side of the node, to the input of the aggregation node.
 - b) Select the aggregation node and, under the *Mapping* tab, right-click on the data source *Join_1* in the left pane. Then, select *Add to Output*.
5. Create a variable based on *PRODUCT_GROUP_TEXT*, using the settings in the following table below, so that a user-specified filter is applied to the product group text column in the view (leave all other settings with their default values):

Setting	Value
Name	VAR_PRODUCT_GROUP_TOP_LEVEL
Label	VAR_PRODUCT_GROUP_TOP_LEVEL
Multiple Entries	check this box
View/Table Value Help	CVD_PRODUCTS CASCADE_VHELP
Reference Column	PRODUCT_GROUP_TEXT

- a) Select the *Semantics* node.
 - b) Select the *Parameters* tab.
 - c) To open the drop-down, choose + and then choose *Variable*.
 - d) Select the empty variable *VAR_1* and enter the *General* settings using the values from the table.
6. Filter the column *PRODUCT_GROUP_TEXT* using the new variable *VAR_PRODUCT_GROUP_TOP_LEVEL*.
 - a) Expand the variable settings section *Apply Filter*.
 - b) To create a new entry, choose +.
 - c) Open the drop-down selector of the new entry and choose *PRODUCT_GROUP_TEXT*.
7. Create a variable based on *PRODUCT_TEXT*, using the settings in the following table, so that a user-specified filter is applied to the product text column in the view (leave all other settings with their default values):

Setting	Value
Name	VAR_PRODUCT_TOP_LEVEL
Label	VAR_PRODUCT_TOP_LEVEL
Multiple Entries	check this box
View/Table Value Help	CVD_PRODUCTS CASCADE_VHELP
Reference Column	PRODUCT_TEXT

- a) Choose *Back* to return to the variables overview screen (you may have to scroll up to see this option).

- b) Choose + and then choose *Variable*.
- c) Select the empty variable *VAR_1* and enter the *General* settings using the values from the table.
8. Filter the column *PRODUCT_TEXT* using the new variable *VAR_PRODUCT_TOP_LEVEL*.

 - a) Expand the variable settings section *Apply Filter*.
 - b) To create a new entry, choose +.
 - c) Open the drop-down selector in the new row and choose *PRODUCT_TEXT*.
9. Map the current view variable *VAR_PRODUCT_GROUP_TOP_LEVEL* to the source variable *VAR_VAL_HELP_PRODUCT_GRP*.

 - a) Choose *Back* to return to the variables overview screen (you may have to scroll up to see this option).
 - b) Choose *Manage Parameter Mapping*.
 - c) In the *Type* selector, choose *Views for value help for variables/input parameters*.
 - d) In the left pane, expand *VAR_PRODUCT_TOP_LEVEL* until you see the variable *VAR_VAL_HELP_PRODUCT_GRP* at the lowest level.
 - e) Drag a line from the source variable *VAR_VAL_HELP_PRODUCT_GRP* to the target variable *VAR_PRODUCT_GROUP_TOP_LEVEL*.
10. Save and build the calculation view.

 - a) Choose *Save*.
 - b) In the *Workspace* tree, right-click the *CVC_SALES CASCADE_TOP* calculation view and choose *Build* → *Build Selected Files*.
11. Check the results using *Data Preview* of Web IDE.

 - a) Right-click on the calculation view *CVC_SALES CASCADE_TOP* in the navigation tree of Web IDE and choose *Data Preview*.
 - b) At the prompt, use the value help selector in the *From* column of the variable *VAR_PRODUCT_GROUP_TOP_LEVEL* and choose any value. Choose *OK*.
 - c) Use the value help selector in the *From* column of the second variable *VAR_PRODUCT_TOP_LEVEL*. Note that the values are restricted to the product group you chose in the first prompt.
 - d) Select one or more values and choose *OK* to view the results.
 - e) Choose *Open Content* and then switch to the *Raw Data* tab to view the results.

Unit 3

Exercise 17

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 the SAP Web IDE
- Explore the data with MS Excel

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
- Task 4 (Optional): Consume the Hierarchy with Microsoft Excel

Task 1: Import the Calculation Views and Review Their Definition

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
2. Import the following calculation views from your course files folder into your workspace folder *exercises*:
 - CVD_PRODUCTS_HIER.hdbcalculationview
 - CVC_SALES_HIER.hdbcalculationviewThey are located in the folder *HA300 → 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. Build the two imported calculation views.

**Note:**

You can build the two calculation views at the same time after selecting both of them. If not, you must build 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 the SAP Web IDE.
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
Node Style	<i>Level Name</i>
Level 1 Element	<i>PROD_GROUP</i>
Level 2 Element	<i>PROD_NAME</i>

Keep the default properties for the hierarchy levels, in particular the level type *REGULAR* and the sort direction *Ascending*.

3. Save, and then build the *CVD_PRODUCTS_HIER* calculation view.
Check the build 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. Save, and then build the *CVC_SALES_HIER* calculation view.
Check the build 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 the SAP Web IDE 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 the SAP Web IDE.

Task 4: Optional: Consume the Hierarchy with Microsoft Excel

Use the SAP HANA MDX Client with Microsoft Excel to query your calculation view data and explore the *PRODUCT* hierarchy that you have implemented.

1. Start Microsoft Excel and create a new workbook based on a blank template.
2. Add a new data source based on the *CVC_SALES_HIER* calculation view as a PivotTable Report in cell A1. You need to use the Data Connection Wizard and connect Microsoft Excel to your calculation view with the following information:

Field	Value
Host	wdf1bmt7215
Instance Number	00
Database Mode	<i>Multi Database > User Database</i> Database name: H00
User	STUDENT##
Password	Training1
Data Source Path	HA300_##_HDI_HDB_1 → HA300

3. Add the following fields to the pivot table:

- QUANTITY
- AMOUNT
- Product Level Hierarchy

**Note:**

In this simple example, the data model does not include the individual product references, and the *PRODUCT_TEXT* column is actually a grouping level in the hierarchy (such as *Keyboards* or *Mouses*). It corresponds to the *PRODUCT_ID* column.

If you add the *PRODUCT_ID* column to the pivot table, you will not get an additional level of detail. You could have defined the *PRODUCT_TEXT* column as the label column for *PRODUCT_ID*.

4. Explore the pivot table to display the details of product groups *Devices* and *Main*.
5. Close the workbook containing the pivot table without saving it and exit Microsoft Excel.
6. If not already done, close all open tabs in the SAP Web IDE.

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 the SAP Web IDE
- Explore the data with MS Excel

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
- Task 4 (Optional): Consume the Hierarchy with Microsoft Excel

Task 1: Import the Calculation Views and Review Their Definition

1. If needed, launch the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on using **STUDENT##** as the user name and **Training1** as the password (where ## is your group number).
2. Import the following calculation views from your course files folder into your workspace folder *exercises*:
 - CVD_PRODUCTS_HIER.hdbcalculationview
 - CVC_SALES_HIER.hdbcalculationviewThey are located in the folder *HA300 → Calculation Views Templates*.
 - a) Right-click the *exercises* folder and choose *Import → From File System*.
 - b) Choose *Browse*, select the file *HA300 → Calculation Views Templates → CVD_PRODUCTS_HIER.hdbcalculationview*, and choose *OK*.

- c) Check the *Import to Location* and choose **OK**.
 - d) 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) Double-click 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. Build the two imported calculation views.

**Note:**

You can build the two calculation views at the same time after selecting both of them. If not, you must build them in the relevant dependency order; that is, the *CVD_PRODUCTS_HIER* calculation view first, and then, the *CVC_SALES_HIER*.

- a) In the workspace, select the two calculation views.
Select the first one, then press **ctrl** while you select the second one.
 - b) Right-click the selection and choose *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed successfully.
 - d) Alternatively, to build one view after another, right-click the *CVD_PRODUCTS_HIER* calculation view and choose *Build Selected Files*. After checking that this first build 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 the SAP Web IDE.
 - 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
Node Style	<i>Level Name</i>
Level 1 Element	<i>PROD_GROUP</i>
Level 2 Element	<i>PROD_NAME</i>

Keep the default properties for the hierarchy levels, in particular the level type *REGULAR* and the sort direction *Ascending*.

- a) In the *Semantics Node*, select the *Hierarchies* tab.
 - b) Choose + to create a new hierarchy and choose *Level Hierarchy*.
 - c) Enter the *Name* and *Label* as provided in the table.
 - d) In the *NODES* pane, choose + to create a new level and select the *PROD_GROUP* column as Level 1.
 - e) Repeat the previous step to add the *PROD_NAME* column as Level 2.
3. Save, and then build the *CVD_PRODUCTS_HIER* calculation view.
Check the build status after activation.
- a) Choose Save.
 - b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed 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 of your workspace, double-click 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. Save, and then build the *CVC_SALES_HIER* calculation view.
Check the build status after activation.
- a) Choose Save.
 - b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed successfully.

Task 3: Preview the Data of the Calculation View in Hierarchy Mode

You will use the *Hierarchy* tab, which is natively integrated in the SAP Web IDE Data Preview.

1. Preview the data of the *CVC_SALES_HIER* calculation view.
 - a) With the calculation view opened, choose *Edit* → *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 the SAP Web IDE.
- a) Right-click any open tab and choose *Close All*.

Task 4: Optional: Consume the Hierarchy with Microsoft Excel

Use the SAP HANA MDX Client with Microsoft Excel to query your calculation view data and explore the *PRODUCT* hierarchy that you have implemented.

1. Start Microsoft Excel and create a new workbook based on a blank template.
 - a) In the Windows *Start* page, enter **Excel** and choose *Excel 2013*.
 - b) Select the *Blank workbook* template.
2. Add a new data source based on the *CVC_SALES_HIER* calculation view as a PivotTable Report in cell A1. You need to use the Data Connection Wizard and connect Microsoft Excel to your calculation view with the following information:

Field	Value
Host	wdf1bmt7215
Instance Number	00
Database Mode	<i>Multi Database > User Database</i> Database name: H00
User	STUDENT##
Password	Training1
Data Source Path	HA300_##_HDI_HDB_1 → HA300

- a) Choose *Data → Existing Connections*.
- b) Choose *Browse for More*.
- c) Choose *+Connect to New Data Source.odc* and choose *Open*.
- d) Select *Other/Advanced* and choose *Next*.
- e) Select *SAP HANA MDX Provider* and choose *Next*.
- f) Enter the connection details as per the table.
- g) Choose *OK*.
If you get a connection error, double-check the credentials and test the connection again.
- h) At the *Select Database and Table* stage, in the drop-down list, select *HA300_##_HDI_HDB_1 → HA300* and select the *Connect to a specific cube* check box.

- i) In the list of views, select the view CVC_SALES_HIER and choose *Finish*.

**Note:**

If you repeat this exercise later and try to add the data source again after modifying it, you might get a dialog box saying that you already have a *CVC_SALES_HIER.odc* (Office Data Connection) file and asking if you want to replace it. Choose *Yes*.

- j) In the *Import Data* dialog box, keep the default options and choose *OK*. You might need to enter your SAP HANA password again.

3. Add the following fields to the pivot table:

- QUANTITY
- AMOUNT
- Product Level Hierarchy

**Note:**

In this simple example, the data model does not include the individual product references, and the *PRODUCT_TEXT* column is actually a grouping level in the hierarchy (such as *Keyboards* or *Mouses*). It corresponds to the *PRODUCT_ID* column.

If you add the *PRODUCT_ID* column to the pivot table, you will not get an additional level of detail. You could have defined the *PRODUCT_TEXT* column as the label column for *PRODUCT_ID*.

- a) In the *Pivot Table Fields* panel, select the QUANTITY, AMOUNT, and Product Level Hierarchy items.
- b) Explore the Product Level Hierarchy with the *Expand* and *Collapse* buttons in column A.
4. Explore the pivot table to display the details of product groups *Devices* and *Main*.
 - a) In the Pivot Table, choose + to expand the *Devices* and *Main* product groups.
5. Close the workbook containing the pivot table without saving it and exit Microsoft Excel.
 - a) Choose *File* → *Close*.
 - b) Choose *Don't Save*.
 - c) To exit Microsoft Excel, choose *File* → *Close*.
6. If not already done, close all open tabs in the SAP Web IDE.
 - a) Right-click any open tab and choose *Close All*.

Unit 3

Exercise 18

Create a Parent-Child Hierarchy

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.

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. 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]

2. Add a Join node *Join_1* to the Scenario pane, and add the following data sources to the new node:
 - *EMPLOYEE* (synonym: *HA300::EMPLOYEE*).
 - *EMPLOYEE_HIERARCHY* (synonym: *HA300::EMPLOYEE_HIERARCHY*).
3. Preview the data source *EMPLOYEE_HIERARCHY* directly from the *Join_1* node to check its structure.

To whom is employee number D100003 reporting?

4. In your *Join_1* node, create a join between the tables using the following properties:

Left Table	HA300::EMPLOYEE
Right Table	HA300::EMPLOYEE_HIERARCHY
Join Columns	EMPLOYEE.DNUMBER = EMPLOYEE_HIERARCHY.CHILD_DNUMBER
Join Type	Left Outer

Cardinality	1..n
-------------	------

5. In the *Join_1* node, add the following columns to output:
 - EMPLOYEE.DNUMBER
 - EMPLOYEE.NAME
 - EMPLOYEE.REMAINDERDAYS
 - EMPLOYEE_HIERARCHY.PARENT_DNUMBER
6. Connect the *Join_1* node to the *Aggregation* node.
7. 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.
8. 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	<i>DNUMBER</i>
Parent	<i>PARENT_DNUMBER</i>
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.
9. On the *Columns* tab of the *Semantics* node, define the *NAME* attribute as the *Label Column* for *DNUMBER*.
 10. Save, and then build the *CVC_EMPLOYEE_HIER* calculation view.
Check the build status after activation.

**Note:**

The hierarchy can be displayed by executing an MDX query on top of the column view, from the *Database Explorer* perspective.

11. 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.

Task 2: Explore Additional Capabilities Relating to Root Nodes and Orphan Nodes

1. Replace the data source *HA300::EMPLOYEE_HIERARCHY* with the table *HA300::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. Save, and then build the *CVC_EMPLOYEE_HIER* calculation view.
Check the build status after activation.
4. Preview the data of the calculation view in hierarchy mode.
5. Close all the open tabs in the SAP Web IDE.

Create a Parent-Child Hierarchy

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.

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. 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 *Workspace* tree, choose *HA300 → HDB → src* and right-click the *exercises* folder. Choose *New → Calculation View*.
b) Enter the calculation view name and other properties as specified in the table.
c) Choose *Create*.
The calculation view graphical editor opens on the right of your screen.
2. Add a Join node *Join_1* to the *Scenario* pane, and add the following data sources to the new node:
 - *EMPLOYEE* (synonym: *HA300::EMPLOYEE*).
 - *EMPLOYEE_HIERARCHY* (synonym: *HA300::EMPLOYEE_HIERARCHY*).
 - a) In the *Scenario* pane, add a join node below the top nodes by dragging a join node from the palette.
 - 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 tables. Press **Ctrl** to perform a multiple selection.
 - e) Choose *Finish*.
3. Preview the data source *EMPLOYEE_HIERARCHY* 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

4. In your *Join_1* node, create a join between the tables using the following properties:

Left Table	HA300::EMPLOYEE
Right Table	HA300::EMPLOYEE_HIERARCHY
Join Columns	EMPLOYEE.DNUMBER = EMPLOYEE_HIERARCHY.CHILD_DNUMBER
Join Type	Left Outer
Cardinality	1..n

- a) Drag a connecting line between the *DNUMBER* of the left table, and the column *CHILD_DNUMBER* of the right table.
 - b) 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 *HA300::EMPLOYEE*. If it is not, you can swap the tables with the *Swap Table* button.

5. 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.

6. 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.

7. 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*.
8. 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	<i>DNUMBER</i>
Parent	<i>PARENT_DNUMBER</i>
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.
- a) Select the *Hierarchies* tab of the *Semantics* node.
- b) Choose + and select *Parent Child Hierarchy*.
- c) Enter the *Name* and *Label* as shown in the table.
- d) In the *NODES* area, choose + and assign the *Child* and *Parent* columns as per the table.
- e) In the *PROPERTIES* area, deselect the *Cache* checkbox.
9. On the *Columns* tab of the *Semantics* node, define the *NAME* attribute as the *Label Column* for *DNUMBER*.
- a) Select the *Semantics* node and display the *Columns* tab.
- b) For the *DNUMBER* row, select the *NAME* attribute from the drop-down menu of *Label Column*.
10. Save, and then build the *CVC_EMPLOYEE_HIER* calculation view.

Check the build status after activation.



Note:

The hierarchy can be displayed by executing an MDX query on top of the column view, from the *Database Explorer* perspective.

- a) Choose Save.
- b) In the *Workspace* tree, navigate to *HA300 → HDB → src → exercises*.
- c) Right-click the *CVC_EMPLOYEE_HIER* calculation view and choose *Build Selected Files*.

- d) In the *Console* pane, check that the build was completed successfully.
11. 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 workspace, 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 *HA300::EMPLOYEES* if you want to check the number of remainder days for employees who are also managers (hierarchy nodes).

Task 2: Explore Additional Capabilities Relating to Root Nodes and Orphan Nodes

1. Replace the data source *HA300::EMPLOYEE_HIERARCHY* with the table *HA300::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 *HA300::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_HIERARCHY2* directly from the *Join_1* node to check the data.
 - a) In the *Join_1* node, right-click the *EMPLOYEE_HIERARCHY2* table and choose *Data Preview*.
 - b) Observe the differences with the previous data set (if needed, it is already opened in a tab of the Web IDE)

What is the main difference compared with the initial data set?

The member *D100012* has no parent.

3. Save, and then build the *CVC_EMPLOYEE_HIER* calculation view.
Check the build status after activation.
 - a) Choose Save.
 - b) Choose *Build* → *Build Selected Files*Right-click the *CVC_EMPLOYEE_HIER* calculation view and choose *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed successfully.
4. Preview the data of the calculation view in hierarchy mode.
 - a) In the workspace, right-click the *CVC_EMPLOYEE_HIER2* 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.
5. Close all the open tabs in the SAP Web IDE.
 - a) Right-click any open tab and choose *Close All*.

Unit 3

Exercise 19

Implement a Hierarchical Value Help

Business Example

User prompts are very popular with your users. However, the value help is cumbersome to work with, as the values are presented in a flat list and some of the lists are very long and contain many irrelevant entries for some users. You have been asked to provide value help that is organized in a hierarchy, so that the user is presented with a compact, collapsed tree. This allows the user to expand the tree at the top level to expose only a subset of values that are relevant for their selection.

1. Create a calculation view using the data in the following table:

Field	Value
Name	CVD_CUSTOMERS_HIER_VHELP
Label	Customers by Country
Data Category	DIMENSION

2. Assign the table *HA300::CUSTOMER* as the data source to the default projection node.
3. In the projection node, map all columns to the output.
4. Define a Level hierarchy with the name and label *CUSTOMER_BY_COUNTRY_HIER* that contains two levels. The top level is the countries and the bottom level is the customers.
5. Define a variable which allows the user to select multiple, single entries of the CUSTOMER dimension. Ensure that the value help for the variable is based on the hierarchy you just created in the previous step. Use the information in the table below.

Field	Value
Name	CHOOSE_CUSTOMER
Label	CHOOSE_CUSTOMER
Multiple Entries	[selected]
View/Table Value Help	<i>HA300::CVD_CUSTOMERS_HIER_VHELP (Current View)</i>
Reference Column	<i>CUSTOMER_ID</i>
Hierarchy	<i>CUSTOMER_BY_COUNTRY_HIER</i>

6. Save and build the calculation view.
7. Launch the calculation view and choose a customer from the hierarchy.

Implement a Hierarchical Value Help

Business Example

User prompts are very popular with your users. However, the value help is cumbersome to work with, as the values are presented in a flat list and some of the lists are very long and contain many irrelevant entries for some users. You have been asked to provide value help that is organized in a hierarchy, so that the user is presented with a compact, collapsed tree. This allows the user to expand the tree at the top level to expose only a subset of values that are relevant for their selection.

1. Create a calculation view using the data in the following table:

Field	Value
Name	CVD_CUSTOMERS_HIER_VHELP
Label	Customers by Country
Data Category	DIMENSION

- a) In the *Workspace* tree, choose *HA300 → HBD → src* and right-click the *exercises* folder. Choose *New → Calculation View*.
- b) Enter the calculation view name and other properties as specified in the table. Leave the namespace value as it is.
- c) Choose *Create*.
2. Assign the table *HA300::CUSTOMER* 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* (synonym: *HA300::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 the data source header *HA300::CUSTOMER* and choose *Add To Output*.
4. Define a Level hierarchy with the name and label *CUSTOMER_BY_COUNTRY_HIER* that contains two levels. The top level is the countries and the bottom level is the customers.
 - a) Select the *Semantics* node.
 - b) Select the *Hierarchies* tab.

- c) Choose +.
 - d) From the drop-down list, choose *Level Hierarchy* and note a template hierarchy is generated with the name *LEVEL_1*.
 - e) Select the generated hierarchy and enter the name and label ***CUSTOMER_BY_COUNTRY_HIER***.
 - f) Expand the section *Nodes*.
 - g) Choose +.
 - h) Using the F4 selection under *Column*, select *COUNTRY*.
 - i) Choose +.
 - j) Using the F4 selection under *Column*, select *CUSTOMER_ID*.
5. Define a variable which allows the user to select multiple, single entries of the CUSTOMER dimension. Ensure that the value help for the variable is based on the hierarchy you just created in the previous step. Use the information in the table below.

Field	Value
Name	CHOOSE_CUSTOMER
Label	CHOOSE_CUSTOMER
Multiple Entries	[selected]
View/Table Value Help	<i>HA300::CVD_CUSTOMERS_HIER_VHELP (Current View)</i>
Reference Column	<i>CUSTOMER_ID</i>
Hierarchy	<i>CUSTOMER_BY_COUNTRY_HIER</i>

- a) Select the *Parameters* tab.
 - b) To open the drop-down list, choose +.
 - c) Select the drop-down option *Variable*. Note 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.
6. Save and build the calculation view.
- a) Choose *Save*.
 - b) Choose *Build* → *Build Selected Files..*
7. Launch the calculation view and choose a customer from the hierarchy.
- a) In the *Workspace* tree, right-click the *CVD_CUSTOMERS_HIER_VHELP* calculation view and choose *Data Preview*.
 - b) At the prompt, use the F4 value help selection (under the *From* column) to expand a country and then choose a customer in that country.
 - c) Choose  *Open Content*.

Unit 3

Exercise 20

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
- Use input parameters to define dynamically conversion settings

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
- Task 4: Refine Your Calculation View by using an Input Parameter for the Conversion Rate Date

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 the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
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, *HA300 → Calculation Views Templates*, and is ready for import.
3. Build 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

1. Modify the output columns of the calculation view to have two amounts, one in the source currency, another in the conversion currency. To do so, add a second instance *GROSS_AMOUNT* to the output, and rename the two amount columns as follows:

Previous Name	New Name and New Label
<i>GROSS_AMOUNT</i>	<i>GROSS_AMOUNT_SRC</i>
<i>GROSS_AMOUNT_1</i>	<i>GROSS_AMOUNT_USD</i>

Check that the new column has the type *Measure*.



Note:

As a first step, you will implement a conversion into a fixed currency: *USD*.

2. At this stage, save and build the view, and preview the data.

Note that the data is the same in both amount columns.

3. In the Calculation View editor tab, assign the following semantics to the column *GROSS_AMOUNT_SRC*:



Note:

Do not enable this column for conversion, as we want to display it in the source currency.

Field	Value
Semantic	Amount with Currency Code
Display Currency	<ul style="list-style-type: none"> • Type: <i>Column</i> • Assigned Column: <i>CURRENCY_CODE</i>

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	<i>Amount with Currency Code</i>

Field	Value
Display Currency	Type: <i>Fixed</i> Assigned currency: <i>USD</i>
Conversion	Yes
Decimal shifts	Yes
Rounding	No
Shift back	No
Reverse Look Up	No
Rates	<i>HA300::TCURR</i>
Configuration	<i>HHA300::TCURV</i>
Prefactors	<i>HA300::TCURF</i>
Precision	<i>HA300::TCURX</i>
Notations	<i>HA300::TCURN</i>
Client for currency conversion	<i>Fixed</i> Client: 800
Source Currency	Type: <i>Column</i> Assigned column: <i>CURRENCY_CODE</i>
Target Currency	Type: <i>Fixed</i> Assigned currency: <i>USD</i>
Exchange Type	<i>Fixed</i> <i>EZB</i>
Conversion Date	<i>Fixed</i> Date: 20121231
Exchange Rate	[blank]
Data Type	<i>Decimal (15,2)</i>
Generate result currency column	No
Upon Failure	<i>Fail</i>
Accuracy	Intermediate Rounding

5. Save, and then build the calculation view.

Check the build status after activation.

6. Preview the data of the calculation view.

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.

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
<i>NET_AMOUNT</i>	<i>NET_AMOUNT_SRC</i>
<i>NET_AMOUNT_1</i>	<i>NET_AMOUNT_USD</i>

2. 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*.
3. Save, and then build the Calculation View.
Check the build status after activation.
4. Preview the data of the calculation view.

Task 4: Refine your Calculation View by Using an Input Parameter for the Conversion Rate Date

The calculation view you imported at the beginning of this exercise includes an input parameter *INP_CONV_DATE* that you have not used so far. You will now modify the calculation view in order to modify and make use of this input parameter.

1. Modify the *INP_CONV_DATE* input parameter so that it suggests a fixed list of dates: *31/12/2012* and *31/12/2013*.

You can define the list of values as follows:

Value	Description
20121231	31/12/2012
20131231	31/12/2013

Entering a value for the input parameter must be mandatory when the view is executed. And **20121231** should be defined as the default conversion date.

2. Modify the semantics assigned to the column *GROSS_AMOUNT_USD* so that the conversion date is not hard-coded in the calculation view, but based on the *INP_CONV_DATE* input parameter.
3. Check that the same change has been automatically applied to the column *NET_AMOUNT_USD*.
4. Save, and then build the Calculation View.
Check the build status after activation.

5. Preview the data.

This time, the *Variables and Input Parameters* dialog box opens. You must set the conversion date value for the *INP_CONV_DATE* input parameter in the *From* column. Choose one of the two possible conversion dates.

Execute a second data preview with the other date and check the data again to ensure the conversion amounts are different.

6. Close all open tabs in the SAP Web IDE.

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
- Use input parameters to define dynamically conversion settings

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
- Task 4: Refine Your Calculation View by using an Input Parameter for the Conversion Rate Date

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 the SAP Web IDE for SAP HANA. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on using **STUDENT##** as the user name and **Training1** as the password (where ## is your group number).
2. Import the template calculation view *CVC_SO_CCY* into your exercises folder.
A design-time file, *CVC_SO_CCY.hdbculationview*, is located in your exercise folder, *HA300 → Calculation Views Templates*, and is ready for import.
 - a) Right-click the exercises folder and choose *Import → File or Project*.

- b) Choose *Browse*, select the file *HA300 → Calculation Views Templates → CVC_SO_CCY.hdbcalculationview* and choose *OK*.
- c) Check the *Import to Location* and choose *OK*.
3. Build the imported calculation view.
- Right-click the imported calculation view *exercises → CVC_SO_CCY* and choose *Build → Build Selected Files*.
 - In the *Console* pane, check that the build was completed 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.
- Right-click the calculation view and choose *Data Preview*.
 - Select the *Raw Data* tab and review the contents.
 - Notice that all the data is expressed in EUR.
 - 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

1. Modify the output columns of the calculation view to have two amounts, one in the source currency, another in the conversion currency. To do so, add a second instance *GROSS_AMOUNT* to the output, and rename the two amount columns as follows:

Previous Name	New Name and New Label
<i>GROSS_AMOUNT</i>	<i>GROSS_AMOUNT_SRC</i>
<i>GROSS_AMOUNT_1</i>	<i>GROSS_AMOUNT_USD</i>

Check that the new column has the type *Measure*.



Note:

As a first step, you will implement a conversion into a fixed currency: *USD*.

- a) In the *Mapping* tab of the *Aggregation* node, right-click the *GROSS_AMOUNT* column and choose *Add to Output*.

- b) Select the *Columns* tab of the Semantics node.
- c) Enter the new names and labels for the two amount columns as shown in the table.
2. At this stage, save and build the view, and preview the data.
Note that the data is the same in both amount columns.
- Choose Save.
 - Choose *Build* → *Build Selected Files*.
 - Choose *Edit* → *Data Preview*.
 - Close the *Data Preview* tab.
3. In the Calculation View editor tab, assign the following semantics to the column *GROSS_AMOUNT_SRC*:

**Note:**

Do not enable this column for conversion, as we want to display it in the source currency.

Field	Value
Semantic	Amount with Currency Code
Display Currency	<ul style="list-style-type: none"> • Type: <i>Column</i> • Assigned Column: <i>CURRENCY_CODE</i>

- In the Semantics node, select the *Columns* tab.
- Open the value help of the *Semantics* property for the *GROSS_AMOUNT_SRC* column.
- In the *Semantic Type* dropdown list, select *Amount with Currency Code*.
- 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	<i>Amount with Currency Code</i>
Display Currency	<ul style="list-style-type: none"> Type: <i>Fixed</i> Assigned currency: <i>USD</i>
Conversion	Yes
Decimal shifts	Yes
Rounding	No
Shift back	No
Reverse Look Up	No

Field	Value
Rates	<i>HA300::TCURR</i>
Configuration	<i>HHA300::TCURV</i>
Prefactors	<i>HA300::TCURF</i>
Precision	<i>HA300::TCURX</i>
Notations	<i>HA300::TCURN</i>
Client for currency conversion	<i>Fixed</i> Client: 800
Source Currency	Type: <i>Column</i> Assigned column: <i>CURRENCY_CODE</i>
Target Currency	Type: <i>Fixed</i> Assigned currency: <i>USD</i>
Exchange Type	<i>Fixed</i> <i>EZB</i>
Conversion Date	<i>Fixed</i> Date: 20121231
Exchange Rate	[blank]
Data Type	<i>Decimal (15,2)</i>
Generate result currency column	No
Upon Failure	<i>Fail</i>
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 all the settings as specified in the table.
- d) Choose *OK*.



Note:

If a dialog box asks for a confirmation to apply identical display currency and target currency settings, choose Yes.

5. Save, and then build the calculation view.

Check the build status after activation.

- a) Choose *Save*.

- b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed successfully.
 - d) Choose *Edit* → *Data Preview*.
 - e) Select the *Raw Data* tab and review the contents.
6. Preview the data of the calculation view.
 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.

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
<i>NET_AMOUNT</i>	<i>NET_AMOUNT_SRC</i>
<i>NET_AMOUNT_1</i>	<i>NET_AMOUNT_USD</i>

- 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. 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) In the *Reuse Semantics* drop-down list, check that the column is *GROSS_AMOUNT_USD*.
 - d) Note that the conversion settings of the column *GROSS_AMOUNT_USD* are displayed, and choose *Next*.
 - e) 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 dropdown 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.

3. Save, and then build the Calculation View.

Check the build status after activation.

- Choose *Build* → *Build Selected Files*.

- In the *Console* pane, check that the build was completed successfully.

4. Preview the data of the calculation view.

- Choose *Edit* → *Data Preview*.

- Check that the conversion of the column *NET_AMOUNT_USD* is consistent with the one applied to the column *GROSS_AMOUNT_USD*.

Task 4: Refine your Calculation View by Using an Input Parameter for the Conversion Rate Date

The calculation view you imported at the beginning of this exercise includes an input parameter *INP_CONV_DATE* that you have not used so far. You will now modify the calculation view in order to modify and make use of this input parameter.

- Modify the *INP_CONV_DATE* input parameter so that it suggests a fixed list of dates; *31/12/2012* and *31/12/2013*.

You can define the list of values as follows:

Value	Description
20121231	31/12/2012
20131231	31/12/2013

Entering a value for the input parameter must be mandatory when the view is executed. And **20121231** should be defined as the default conversion date.

- In the *Semantics* node, select the *Parameters/Variables* tab and select the *INP_CONV_DATE* input parameter.
- Select the *Is Mandatory* checkbox.
- Change the *Parameter Type* to *Static List*.
- For the *Data Type*, choose *DATE*.
- In the *List of values* area, choose *+* to add the two values as specified in the table.
- In the *Default Value* area, choose *+* and enter **20121231**.

- Modify the semantics assigned to the column *GROSS_AMOUNT_USD* so that the conversion date is not hard-coded in the calculation view, but based on the *INP_CONV_DATE* input parameter.

- a) In the *Columns* tab of the *Semantics* node, select the column *GROSS_AMOUNT_USD* and select the *Assign Semantics* icon.
 - b) In the *Conversion Date* field, replace the setting *Fixed* with *Input Parameter*. Assign the parameter *INP_CONV_DATE* and choose *OK*.
 - c) Choose *OK* to finalize the modification of the Semantics for *GROSS_AMOUNT_USD*.
3. Check that the same change has been automatically applied to the column *NET_AMOUNT_USD*.
- a) In the *Columns* tab of the *Semantics* node, select the *NET_AMOUNT_USD* column.
 - b) Select the *Assign Semantics* icon and choose *Assign Semantics*.
 - c) Check that the *Conversion Date* settings refer to the input parameter *INP_CONV_DATE*.
 - d) Choose *Cancel*.
 - e) Alternatively, inside the list of columns, you can choose the *Semantics* property drop-down list for the *NET_AMOUNT_USD*.
If the *Semantics* property is not displayed, select the *Customize Column Display* icon.
4. Save, and then build the Calculation View.
Check the build status after activation.
- a) Choose *Build* → *Build Selected Files*.
 - b) In the *Console* pane, check that the build was completed successfully.
5. Preview the data.
This time, the *Variables and Input Parameters* dialog box opens. You must set the conversion date value for the *INP_CONV_DATE* input parameter in the *From* column. Choose one of the two possible conversion dates.
Execute a second data preview with the other date and check the data again to ensure the conversion amounts are different.
- a) Choose *Edit* → *Data Preview*.
 - b) In the *Variables and Input Parameters* screen, in the *From* drop-down list for *INP_CONV_DATE*, choose *20121231*.
 - c) Choose *Open Content*.
 - d) In the *Raw Data* tab, check the results. They should be identical with what you had at the end of the previous task.
 - e) To execute the view with a different conversion date, choose *Open Parameter Editor* and replace *20121231* with *20131231*.
 - f) Check that the converted amounts have changed.
6. Close all open tabs in the SAP Web IDE.
- a) Right-click any open tab and choose *Close All*.

Unit 3

Exercise 21

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
- Reuse this calculation view in a CUBE calculation view

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
- Task 3: Create a Sales Calculation View Using the TIME Dimension Calculation View

Task 1: Generate Time Data in your HDI Container

1. Generate a new time table with the following settings:

Setting	Value
Calendar Type	Gregorian
Period Unit	Day
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*.

Once you press *OK* the table is built and filled with data but the source file still displays *Pending Deployment*. This is a refresh bug and can be ignored, but if you want to clear this status, simply build the file manually.

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	HA300::M_TIME_DIMENSION
Auto Create	<i>selected</i>

2. Review the generated definition of the calculation view focussing on mapping, filter expression and the hierarchy definition.
3. Select the filter expression tab and observe the filter expression that is already defined.
4. Select the semantics node and review the generated hierarchy.
5. Generate a root node for the hierarchy.
6. Save and build the calculation view.

Task 3: Create a Sales Calculation View Using the TIME Dimension Calculation View

1. In your Exercises folder, create a calculation view of the type cube with star join using the values provided below.

Setting	Value
Name	CVCS_SALES_WITH_TIME_DIM
Label	Sales by Time
Data Category	CUBE
With Star Join	<i>selected</i>

2. Add an aggregation node and assign the table SALES_DATA.
3. Map the columns CUSTOMER_ID, QUANTITY, QTY_UNIT and SAPDATE.
4. Connect the Aggregation_1 node to the Star Join node.
5. Add the time dimension view CVD_DATE_ATTRIBUTES to the Star Join node.
6. Join the time dimension CVD_DATE_ATTRIBUTES to the sales facts (HA300::SALES_DATA) using the columns SAPDATE > DATE_SAP.
7. Map all columns to the output of the Star Join node.
8. Review the time attributes from the time dimension and the time hierarchy.

9. Save and build the calculation view.
10. Check the results using the *Data Preview* option and in particular make sure you are able to navigate the sales data by various time attributes.

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
- Reuse this calculation view in a CUBE calculation view

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
- Task 3: Create a Sales Calculation View Using the TIME Dimension Calculation View

Task 1: Generate Time Data in your HDI Container

1. Generate a new time table with the following settings:

Setting	Value
Calendar Type	Gregorian
Period Unit	Day
From Year	2018
To Year	2019
First Day of Week	Sunday

- a) In your HA300_## project, right-click on the folder *HDB* and choose *Modeling → Maintain Time Tables*
b) In the *Generate Time Data* dialog box, use the settings provided above and choose *OK*.
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*.
Once you press *OK* the table is built and filled with data but the source file still displays *Pending Deployment*. This is a refresh bug and can be ignored, but if you want to clear this status, simply build the file manually.
2. Examine the contents of the generated source file that represents the time table.

- a) In your *HA300_##* 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 *MY HA300 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	HA300::M_TIME_DIMENSION
Auto Create	<i>selected</i>

- a) Right-click your exercises folder and choose *New → Calculation View →*.
 - b) Use the table above to complete the fields and press *Create*.
2. Review the generated definition of the calculation view focussing on mapping, filter expression and the hierarchy definition.
 - a) Select the *Projection_1* node and then choose the filter expression tab to review the filters.
 - b) Select the mapping tab and notice the time of day attributes are not mapped.
3. Select the filter expression tab and observe the filter expression that is already defined.
 - a) In the *Projection_1* node, select the *Filter Expression* tab.
4. Select the semantics node and review the generated hierarchy.
 - a) Select the *Semantics* node.
 - b) In the *Hierarchies* tab, double-click the *Gregorian_Hierarchy* item.
5. Generate a root node for the hierarchy.
 - a) Expand the *Properties* pane.
 - b) In the *Root node Visibility* dropdown list, choose *Add Root node*.
6. Save and build the calculation view.

- a) Choose *File* → *Save*.
- b) Choose *Build* → *Build Selected Files* and check the log to ensure the build is successful.

Task 3: Create a Sales Calculation View Using the TIME Dimension Calculation View

1. In your *Exercises* folder, create a calculation view of the type cube with star join using the values provided below.

Setting	Value
Name	CVCS_SALES_WITH_TIME_DIM
Label	Sales by Time
Data Category	CUBE
With Star Join	<i>selected</i>

- a) Right-click your exercises folder and choose *New* → *Calculation View*.
- b) Use the table above to complete the fields and press *Create*.
2. Add an aggregation node and assign the table *SALES_DATA*.
 - a) In the palette, on the left of the *Scenario* pane, choose *Aggregation* and drag the new node below the *Star Join* node.
 - b) Click on the '+' icon alongside the aggregation node and enter ***sales_data*** so that the *SALES_DATA* table appears in the results.
 - c) Select the *SALES_DATA* table (synonym: *HA300::SALES_DATA*) and choose *Finish*.
3. Map the columns *CUSTOMER_ID*, *QUANTITY*, *QTY_UNIT* and *SAPDATE*.
 - a) In the *Aggregation* node, select the mapping tab and drag the columns *CUSTOMER_ID*, *QUANTITY*, *QTY_UNIT* and *SAPDATE* from the *Data Sources* pane on the left side to the *Output Columns* pane on the right side.
4. Connect the *Aggregation_1* node to the *Star Join* node.
 - a) Select the *Aggregation_1* node and use the connector arrow to drag a line between the *Aggregation_1* node and the *Star Join* node.
5. Add the time dimension view *CVD_DATE_ATTRIBUTES* to the *Star Join* node.
 - a) Select the '+' button next to the star join node and at the dialog enter ***CVD_date*** then select the view when it appears in the results and finally choose *Finish*.
6. Join the time dimension *CVD_DATE_ATTRIBUTES* to the sales facts (*HA300::SALES_DATA*) using the columns *SAPDATE* > *DATE_SAP*.
 - a) Select the join tab and drag a line between the columns *SAPDATE* > *DATE_SAP*.
7. Map all columns to the output of the *Star Join* node.
 - a) In the *Star Join* node, select the *Mapping* tab and drag all columns to the output pane.
8. Review the time attributes from the time dimension and the time hierarchy.
 - a) Select the semantics node and then select the *Shared* sub-tab to display the attributes that are exposed from the time dimension.
 - b) Select the hierarchies tab and then click on the hierarchy to review the settings that are inherited from the time dimension calculation view where it was defined.

9. Save and build the calculation view.
 - a) Choose *File* → *Save*.
 - b) Choose *Build* → *Build Selected Files* and check the log to ensure the build is successful.
10. Check the results using the *Data Preview* option and in particular make sure you are able to navigate the sales data by various time attributes.
 - a) Right-click the calculation view *CVCS_SALES_WITH_TIME_DIM* in the project structure and choose *Data Preview*.
 - b) Select the tab *Hierarchies* and drag *Quantity* to the selected measures pane, so you see the total sales for all customers for all time.
 - c) Drag the hierarchy to the selected hierarchy pane and then explore the quantities by expanding the hierarchy levels through year, quarter, month, week and date.

Unit 4

Exercise 22

Work with the SQL Console

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.

1. Open an SQL Console for your database connection *H00 DB*.
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.



Note:

In the bottom left part of the screen, you will see two search fields. The top search field is where you can enter a schema name so that only tables that belong to the specified schema are displayed. The lower field is where you can enter a full or partial table name so that you can quickly locate your table(s). Using the two search fields helps you find tables quickly, but these fields work independently as well. For example, you might enter a table name but leave the schema field empty so that all tables with the same name, that appear in different schemas, are displayed.

5. Open an SQL Console for your container.
6. Execute a query to read all data from table *EMPLOYEES*.

Unit 4

Solution 22

Work with the SQL Console

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.

1. Open an SQL Console for your database connection *H00 DB*.
 - a) In the SAP Web IDE, switch to the *Database Explorer* view.
 - b) Highlight your SAP HANA classic database connection *H00 DB*, and then choose *Open SQL Console* in the toolbar (or use the right-click menu option).
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.



Note:

In the bottom left part of the screen, you will see two search fields. The top search field is where you can enter a schema name so that only tables that belong to the specified schema are displayed. The lower field is where you can enter a full or partial table name so that you can quickly locate your table(s). Using the two search fields helps you find tables quickly, but these fields work independently as well. For example, you might enter a table name but leave the schema field empty so that all tables with the same name, that appear in different schemas, are displayed.

- a) In the top left pane, navigate to *H00 DB (CLASSIC) → Catalog → Tables*.
- b) In the schema search field, enter **TRAINING**.
You should now see many tables appear in the lower pane. These are all tables in the *TRAINING* schema.
- c) In the table search field, enter **ORDERS**.
You should now see only tables that contain the word *orders*.

d) In the lower left pane, right-click on the table *ORDERS* and choose the context menu option *Generate SELECT statement*.

e) 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 *INSERT*, *CREATE*.

5. Open an SQL Console for your container.

a) Highlight your container and then choose *Open SQL Console* in the toolbar (or use the right-click menu option).

6. Execute a query to read all data from table *EMPLOYEES*.

a) In the empty tab, enter the code:

```
SELECT * FROM "HA300::Employees"
```

Notice the use of the namespace *HA300* 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).

b) Press *Run (F8)*.

Unit 4

Exercise 23

Read a Modeled Hierarchy using SQL

Exercise Objectives

After completing this exercise, you will be able to:

- 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.



Note:

In this exercise, ## represents the two-digit group number assigned to you by the instructor.

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 hierarchy based on **City** and **Country** to organize the employee data. You will access this hierarchy using SQL.

1. In your resources folder, open the Calculation View **CVD_ADDRESSES** 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_EMPLOYEES**. This calculation view joins information for each employee with their address attributes, but most importantly, also enables SQL access to the hierarchy.

1. In your resources folder, open the Calculation View **CVCS_EMPLOYEES** and examine the definition. Pay particular attention to the settings that enable SQL access to the shared hierarchy.

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 sum the salaries of employees by their location using the hierarchy.

1. In the SAP Web IDE, switch to the *Database Explorer* and open an *SQL Console* in **MY HA300 CONTAINER**.
2. From your **HA300** resources folder, open the file *Prepared Code → SQL for reading hierarchy structure.txt*, and copy and paste its entire contents to the SQL console.
3. Review the SQL code to study how the hierarchy is read.
4. Execute the SQL statement.
The result is displayed as shown in the figure, SQL Query Result.

	H_GEOnode	SUM(SALARY_AM.)
1	[CITY].[Antioch, Illinois]	1767002
2	[COUNTRY].[US]	2962927
3	[All].[all]	3565994
4	[CITY].[San Francisco]	576416
5	[CITY].[New York]	619509
6	[CITY].[London]	478449
7	[COUNTRY].[EN]	478449
8	[COUNTRY].[DE]	124618
9	[CITY].[Berlin]	124618



Figure 11: SQL Query Result

5. Remove the comments so that you can execute the query again, this time with the filter set to 'US', to display the aggregated total salaries for all the cities in the US.

Read a Modeled Hierarchy using SQL

Exercise Objectives

After completing this exercise, you will be able to:

- 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.



Note:

In this exercise, ## represents the two-digit group number assigned to you by the instructor.

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 hierarchy based on **City** and **Country** to organize the employee data. You will access this hierarchy using SQL.

1. In your *resources* folder, open the Calculation View **CVD_ADDRESSES** and examine the definition. Pay attention to the hierarchy that is defined in the calculation view.
 - a) If not already done, switch to the *Development* view of SAP Web IDE.
 - b) In your *resources* sub-folder, double-click the calculation view **CVD_ADDRESSES**.
 - c) Review the data source **SNWD_AD** by double-clicking the *Project* node so that you see the columns provided on which 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_GEO** and then expand the nodes section. In particular, note how the hierarchy is a level type (observe the hierarchy icon) and is based on *Country* and *City* attributes in that exact drill-down sequence.

Task 2: Review the Cube Calculation View that Exposes the Shared Hierarchy to SQL

Review the calculation view **CVCS_EMPLOYEES**. This calculation view joins information for each employee with their address attributes, but most importantly, also enables SQL access to the hierarchy.

1. In your *resources* folder, open the Calculation View **CVCS_EMPLOYEES** and examine the definition. Pay particular attention to the settings that enable SQL access to the shared hierarchy.
 - a) In your *resources* folder, double-click the Calculation View **CVCS_EMPLOYEES**.

- b) Review the star join node. Notice how the address and employee data is 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_GEO* and expand the section *SQL ACCESS*. Notice the name *H_GEONode* 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.

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 sum the salaries of employees by their location using the hierarchy.

1. In the SAP Web IDE, switch to the *Database Explorer* and open an *SQL Console* in *MY HA300 CONTAINER*.
 - a) Highlight your database container *MY HA300 CONTAINER* and then choose *Open SQL Console* in the toolbar.
2. From your *HA300* resources folder, open the file *Prepared Code → SQL for reading hierarchy structure.txt*, and copy and paste its entire contents to the SQL console.
 - a) Double-click the file *HA300 → 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*.

```

SELECT
  "H_GEONode",
  SUM("SALARY_AMOUNT")

FROM "HA300::CVCS_EMPLOYEES"

-- WHERE "H_GEONode" = '[COUNTRY].[US]'
-- remove the first two hypens at the very start of this line to
enable the filter

GROUP BY "H_GEONode";
  
```

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_GEONode*, which is the generated column that represents the nodes at all levels of the hierarchy.
4. Execute the SQL statement.
 - a) To execute the script, press **F8** or choose *Run* on the toolbar.

The result is displayed as shown in the figure, SQL Query Result.

	H_GEOnode	SUM(SALARY_AM.)
1	[CITY].[Antioch, Illinois]	1767002
2	[COUNTRY].[US]	2962927
3	[All].[all]	3565994
4	[CITY].[San Francisco]	576416
5	[CITY].[New York]	619509
6	[CITY].[London]	478449
7	[COUNTRY].[EN]	478449
8	[COUNTRY].[DE]	124618
9	[CITY].[Berlin]	124618

Figure 11: SQL Query Result

5. Remove the comments so that you can execute the query again, this time with the filter set to 'US', to display the aggregated total salaries for all the cities in the US.
 - a) Edit the SQL code to remove the two hyphens in front of the 'WHERE' clause.
 - b) To execute the script, press **F8** or choose *Run* on the toolbar.

Unit 4 Exercise 24

Work with SQLScript

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 Web IDE. However, for testing purposes it is possible to write and execute SQLScript directly in the SQL Console of Web IDE 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.

1. If not already there, switch to the *Database Explorer* and open an *SQL Console* in the database connection *H00 DB*.
2. From your *HA300* resources folder, open the file *Prepared Code → SQL 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.

Unit 4

Solution 24

Work with SQLScript

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 Web IDE. However, for testing purposes it is possible to write and execute SQLScript directly in the SQL Console of Web IDE 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.

1. If not already there, switch to the *Database Explorer* and open an *SQL Console* in the database connection *H00 DB*.
 - a) Highlight your database connection *H00 DB* and then choose *Open SQL Console* in the toolbar.
2. From your *HA300* resources folder, open the file *Prepared Code → SQL for Anonymous Block.txt*, and copy and paste its entire contents to the SQL console.
 - a) Double-click the file *HA300 → Prepared Code → SQL 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.

Unit 4

Exercise 25

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.



Note:

For this exercise '##' represents the two-digit group number assigned to you by the instructor.

Task 1: Create a new scalar function to identify the top customer based on sales quantity

1. In the *Development* view of SAP Web IDE, create a new scalar function **SF_GET_BEST_CUSTOMER** in the folder *exercises*.

2. Create the SQLScript of your scalar function using the following code:

```
FUNCTION "HA300::SF_GET_BEST_CUSTOMER"()
RETURNS best_cust nvarchar(10)
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN

SELECT customer_id into best_cust from "HA300::SALES_DATA"
where quantity = (select max(quantity) from "HA300::SALES_DATA");

END;
```



Note:

The SQLScript code can be copied from a text file in your HA300 → Prepared Code folder: *SQL for Scalar Function.txt*. Don't forget to change ## for your own group number.

3. Save and build 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

1. Create a calculation view type cube in your exercises folder with the name and label **CVC_SALES_IP_USING_FUNC**.
2. Maximize the calculation view editor screen so that you are able to view all areas.
3. Assign the table *SALES_DATA* as a data source to the aggregation node.
4. Add the four columns *CUSTOMER_ID* — *QUANTITY* — *QTY_UNIT* — *SQL_DATE* — *PRODUCT_ID* — *AMOUNT* — *CURRENCY* to the output of the aggregation node.
5. Create an input parameter *IP_1* that filters the *CUSTOMER_ID* to the value returned by the scalar function.
6. Define a filter expression that uses the scalar function returned value to select only the best customer.
7. Define the sorting sequence of the column *QUANTITY* as descending so that the highest sales quantity is shown at the top of the display.
8. Save and build your calculation view.
9. Test the calculation view correctly returns the sales orders of the customer who made the biggest quantity purchase.

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.



Note:

For this exercise '##' represents the two-digit group number assigned to you by the instructor.

Task 1: Create a new scalar function to identify the top customer based on sales quantity

1. In the *Development* view of SAP Web IDE, create a new scalar function **SF_GET_BEST_CUSTOMER** in the folder *exercises*.
 - a) If not already open, switch to the *Development* view of SAP Web IDE.
 - b) Navigate to *HDB* → *src* → *exercises..*
 - c) Right-click the *exercises* folder and choose *New* → *Function..*
 - d) In the *File name* field, enter **SF_GET_BEST_CUSTOMER** and choose *Create*.

2. Create the SQLScript of your scalar function using the following code:

```
FUNCTION "HA300::SF_GET_BEST_CUSTOMER" ( )
RETURNS best_cust nvarchar(10)
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN

SELECT customer_id into best_cust from "HA300::SALES_DATA"
where quantity = (select max(quantity) from "HA300::SALES_DATA");

END;
```



Note:

The SQLScript code can be copied from a text file in your *HA300* → *Prepared Code* folder: *SQL for Scalar Function.txt*. Don't forget to change ## for your own group number.

- a) From Windows Explorer, in your HA300 → Prepared Code folder, double-click the file *SQL for Scalar Function.txt*.
The text file opens in *Notepad*.
 - b) Select all the code (**Ctrl+A**) and copy it to the clipboard (**Ctrl+C**).
 - c) In the *Procedure* source file, remove all existing content and then paste the entire code that you just copied (**Ctrl+V**).
3. Save and build the scalar function and check the log to ensure that it was successful.
- a) Choose *Save*.
 - b) Right-click your new function *SF_GET_BEST_CUSTOMER* and choose *Build* → *Build Selected Files*.
 - c) In the *Console* view, check that the job status is *Completed Successfully*.

Task 2: Create a calculation view that includes an input parameter that consumes your scalar function

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* → *Calculation View*.
 - b) Specify the name and the label of the calculation view as **CVC_SALES_IP_USING_FUNC**.
 - c) In the *Data Category*, ensure that *CUBE* is chosen.
 - d) Choose *Create*.
2. Maximize the calculation view editor screen so that you are able to view all areas.
 - a) Double-click in the center of the header tab where you see the title of the calculation view in order to fill the entire screen with the editor.
3. 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**. Two entries appear.
 - c) Select the table *SALES_DATA* and choose *Finish*.
4. Add the four 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.
5. Create an input parameter *IP_1* 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) Ensure that the value for *Parameter Type* is set to *Derived From Procedure/Scalar Function*.
 - f) Expand the *Parameter Type – Derived From Procedure/Scalar Function* section and choose HA300::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.
6. 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.
7. 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*.
8. Save and build your calculation view.
- a) Choose *Save*.
 - b) Choose *Build* → *Build Selected Files*.
 - c) Confirm that the log shows that the calculation view has generated successfully.
9. 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.

Unit 4

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 realize that the only way you can call such fault-tolerant search functionality is by implementing a table function.

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.

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 that the way to achieve this is by defining a table function.

1. In the Development view of SAP Web IDE, create a new function **TF_FUZZY_EMPLOYEES** under the node *exercises*.
2. Copy and paste the function code from the prepared text file that you can find it in your *HA300* folder. The file you need is *Prepared Code → SQL for Function.txt*.

```
FUNCTION "HA300::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 "HA300::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 HA301 — SAP HANA Advanced Modeling.

3. Save and build your table function.

Task 2: Use the Table Function in a Calculation View Type Dimension Using a Constant Mapping

1. Create a calculation view type dimension in your exercises folder with the name and label **CV_FIND_EMPLOYEES**.
2. Maximize the calculation view editor screen.
If you don't do this, then some icons that you will need later may not be easily found.
3. Add a new *Table Function* node and assign the table function *TF_FUZZY_EMPLOYEES* to it.
4. Map the input parameter *LASTNAMEFILTER* of the table function to a constant value **SMIHT**, which represents a misspelled employee name.
5. Connect the *Table Function* node to the *Projection* node.
6. Map all incoming columns to the output columns of the *Projection* node.
7. Save and build your calculation view.
8. 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 → UNIT_4 → CV_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 *CV_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. Save and build 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**.

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 realize that the only way you can call such fault-tolerant search functionality is by implementing a table function.

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.

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 that the way to achieve this is by defining a table function.

1. In the Development view of SAP Web IDE, create a new function **TF_FUZZY_EMPLOYEES** under the node *exercises*.
 - a) Navigate to *HA300* → *HDB* → *src* → *exercises*.
 - b) Right-click the folder *work* and select *New* → *Function*.
 - c) In the *File name* field, enter **TF_FUZZY_EMPLOYEES**.
 - d) Choose *Create*.
2. Copy and paste the function code from the prepared text file that you can find it in your *HA300* folder. The file you need is *Prepared Code* → *SQL for Function.txt*.

```
FUNCTION "HA300::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 "HA300::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 HA301 — 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.
3. Save and build your table function.
- a) Choose Save.
 - b) Right-click the function and choose *Build Selected Files*.
 - c) Check the log to make sure the function built successfully.

Task 2: Use the Table Function in a Calculation View Type Dimension Using a Constant Mapping

1. Create a calculation view type dimension in your exercises folder with the name and label **CV_FIND_EMPLOYEES**.
 - a) Right-click your exercises folder and choose *New → Calculation View*.
 - b) Enter the name and the label of the calculation view as **CV_FIND_EMPLOYEES**.
 - c) In the *Data Category*, choose *DIMENSION*.
 - d) Choose *Create*.
2. Maximize the calculation view editor screen.
 - a) Double-click in the center of the header tab where you see the title of the calculation view in order to fill the entire screen with the editor.

If you don't do this, then some icons that you will need later may not be easily found.
3. 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)
 - a) In the *Search* field, enter **Fuzzy** and you will see only one entry appears.
 - b) Select the table function **HA300::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.
4. Map the input parameter *LASTNAMEFILTER* of the table function to a constant value **SMIHT**, which represents a misspelled employee name.
- a) Select the *Table Function* node and choose the *Input Mapping* tab.
 - b) In the toolbar, choose the blue C icon (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.
5. 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.
6. 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*.
7. Save and build your calculation view.
- a) Choose *Save*.
 - b) Choose *Build* → *Build Selected Files*.
 - c) Confirm that the log shows that the calculation view has generated successfully.
8. Test the table function by using the *Data Preview* function of the calculation view.
- a) If you maximized the calculation view editor you should double-click in the center of its tab so that you restore the screen so that you see the project folder structure on the left of the screen. 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 → *UNIT_4* → *CV_FIND_EMPLOYEES_00_STAGE1.hdbculationview*.

Task 3: Use the Table Function in a Calculation View Type Dimension Using an Input Parameter Mapping

1. From the calculation view *CV_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 mapping 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 with 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) 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. Save and build your calculation view.
- a) Choose Save.
 - b) Choose *Build* → *Build Selected Files*.
 - c) Confirm that the log shows that the calculation view has generated 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) In the top left corner of the toolbar (to the left of the Save button and directly above the project structure), choose *Open Content*, which has now appeared to execute the data preview using your input value.

You can see that SAP HANA still managed to find two employees with the surname *Smith*, despite you typing the surname incorrectly.

Unit 5

Exercise 27

Create and Delete Tables Using the SQL Console

Exercise Objectives

After completing this exercise, you will be able to:

- Create tables by executing SQL statements in the SAP Web IDE
- Create a table as a copy of another table, with or without data
- Delete a table

Business Example

For testing purposes, you need to quickly create some temporary tables.

Task 1: Create Tables Using SQL Statements

1. In the SAP Web IDE for SAP HANA, using the Database Explorer interface, display the tables in your personal *STUDENT##* schema.



Note:

In the bottom left part of the screen, you will see two search fields. The top search field is where you can enter a schema name so that only tables that belong to the specified schema are displayed. The lower field is where you can enter a full or partial table name so that you can quickly locate your table(s). Using the two search fields helps you find tables quickly, but these fields work independently as well. For example, you might enter a table name but leave the schema field empty so that all tables with the same name, that appear in different schemas, are displayed.

2. Open a SQL Console for your database connection *H00 DB (CLASSIC)*.
3. In the SQL Console tab that has opened, import the course file *HA300 → Prepared Code → SQL to Create and Delete Tables.sql*.
4. Replace *##* with your student number.
5. Find and correct the error in the SQL code so that the code is completely error free and you no longer see error symbols.
6. Execute only the code for **Task 1** to create two new column tables *PUBLISHERS* and *BOOKS*.
7. Locate the two newly created tables and review their metadata.

Task 2: Create a Table as a Copy of an Existing Table, First Without Data, Then with Data

1. Execute only the code for **Task 2.1** to create the table *CurrDecimals* as a copy of the existing table *TCURX*, but without its data.

Why do you not see the table that we copy from *TCURX* in your *STUDENT##* schema?

2. Locate the table *CurrDecimals* and confirm that it is empty.
3. Execute only the code for **Task 2.2** to drop, then recreate the table *CurrDecimals* as a copy of the existing table *TCURX*, but this time with the data.
4. Locate the table *CurrDecimals* and confirm it now contains the copied data.
5. Close all tabs.

Create and Delete Tables Using the SQL Console

Exercise Objectives

After completing this exercise, you will be able to:

- Create tables by executing SQL statements in the SAP Web IDE
- Create a table as a copy of another table, with or without data
- Delete a table

Business Example

For testing purposes, you need to quickly create some temporary tables.

Task 1: Create Tables Using SQL Statements

1. In the SAP Web IDE for SAP HANA, using the Database Explorer interface, display the tables in your personal **STUDENT##** schema.



Note:

In the bottom left part of the screen, you will see two search fields. The top search field is where you can enter a schema name so that only tables that belong to the specified schema are displayed. The lower field is where you can enter a full or partial table name so that you can quickly locate your table(s). Using the two search fields helps you find tables quickly, but these fields work independently as well. For example, you might enter a table name but leave the schema field empty so that all tables with the same name, that appear in different schemas, are displayed.

- a) Navigate to **H00 DB (CLASSIC)** → **Catalog** → **Tables**.
b) In the schema search field, enter **STUDENT##** (where ## represents your group number).
You should now see two existing tables in your **STUDENT##** schema (**RESULT_DT** and **STAFF**). You will add two other tables to this schema in the next steps.
2. Open a SQL Console for your database connection **H00 DB (CLASSIC)**.
 - a) Highlight the database connection **H00 DB (CLASSIC)**, and then choose  **Open SQL Console** in the toolbar.
 3. In the SQL Console tab that has opened, import the course file **HA300 → Prepared Code → SQL to Create and Delete Tables.sql**.
 - a) In the SQL Console tab, choose  **Import File**.

- b) Select the file HA300 → Prepared Code → SQL to Create and Delete Tables.sql and choose Open.

The code is as follows:

```
-----
-- TASK 1 - Create two new tables -
-----

-- Preparation:
-- First replace ## everywhere in this script with your group number

SET SCHEMA STUDENT##;

CREATE COLUMN TABLE publishers -- notice how the table will be created
with upper case name!
( pub_id INTEGER PRIMARY KEY,
name VARCHAR (50),
street VARCHAR (50),
post_code VARCHAR (10),
city VARCHAR (50),
country VARCHAR (50));

CREATE COLUMN TABUL BOOKS -- this will cause a design-time error in
Web IDE
( isbn VARCHAR(20) PRIMARY KEY,
title VARCHAR(50),
publisher INTEGER,
edition INTEGER,
year VARCHAR(4),
price DECIMAL(5, 2),
crcy VARCHAR(3));

-----

-- TASK 2.1 - Copy a table definition but not its records -
-----

/*
CREATE COLUMN TABLE "CurrDecimals" LIKE "TCUR"."TCURX";
*/



-----

-- TASK 2.2 - Drop the copied table and then recreate it but this time
with its data -
-----


/*
DROP TABLE "CurrDecimals";
CREATE COLUMN TABLE "CurrDecimals" LIKE "TCUR"."TCURX" WITH DATA;
*/
```

4. Replace ## with your student number.

- a) In the `SET SCHEMA STUDENT##` statement, replace ## with your assigned student number. Be careful not to remove the ; end of statement character.

**Note:**

This statement is not strictly necessary as the current schema is already set by default to *STUDENT##* which is the connected user schema. But it is good practice to explicitly set the schema in case this code is used in other scenarios where the database user, and therefore the default schema, might change, or you want to work in another schema other than your default one.

5. Find and correct the error in the SQL code so that the code is completely error free and you no longer see error symbols.
 - a) Locate the invalid word **TABUL** in the SQL statement and change this to **TABLE**. All design-time error symbols should then disappear.
6. Execute only the code for **Task 1** to create two new column tables *PUBLISHERS* and *BOOKS*.
 - a) Highlight only the code inside **Task 1** and press **F8** (or choose *Run* on the toolbar).
7. Locate the two newly created tables and review their metadata.
 - a) In the Database Explorer, notice two new tables have appeared in your *STUDENT##* schema. In particular, notice how the use of lower case characters without quotes means the table names are created in upper case. If we had used quotes, the tables would have been created with lower case names.
 - b) In the lower left pane, select the table *BOOKS* and you will see a new tab will appear to display the metadata of the table.
 - c) Repeat the last step for the other table.

Task 2: Create a Table as a Copy of an Existing Table, First Without Data, Then with Data

1. Execute only the code for **Task 2.1** to create the table *CurrDecimals* as a copy of the existing table *TCURX*, but without its data.
 - a) Delete the two characters `/*` from the start of **Task 2.1**.
 - b) Delete the two characters `*/` from the end of **Task 2.1**.
 - c) Highlight the code inside **Task 2.1** only and press **F8** or choose *Run* on the toolbar.

Why do you not see the table that we copy from *TCURX* in your *STUDENT##* schema?

The *TCURX* table is located in the schema *TCUR*. Notice this time we explicitly specify the schema name *TCUR* before the table name. When we do this, we override the current *STUDENT##* schema which is the default schema for our database user and is used whenever we do not specify a schema name in an SQL statement.

2. Locate the table *CurrDecimals* and confirm that it is empty.
 - a) In the lower left pane, right-click on the table *CurrDecimals* and choose the context menu *Open Data*. A new tab appears where you can see that the table is empty.
3. Execute only the code for **Task 2.2** to drop, then recreate the table *CurrDecimals* as a copy of the existing table *TCURX*, but this time with the data.

- a) Delete the two characters /* from the start of **Task 2.2**.
 - b) Delete the two characters */ from the end of **Task 2.2**.
 - c) Highlight the two lines of code inside **Task 2.2** only, and press **F8** or choose **Run** on the toolbar.
4. Locate the table *CurrDecimals* and confirm it now contains the copied data.
 - a) In the lower left pane, right-click on the table *CurrDecimals* and choose the context menu option *Open Data*.
A new tab appears where you can confirm the table is now filled with data.
 5. Close all tabs.
 - a) Right-click any tab and choose *Close All*.

Unit 5

Exercise 28

Create Tables Using Source Files

Exercise Objectives

After completing this exercise, you will be able to:

- Create a table based on a source file using .hdbtable

Business Example

The data sources for your calculation views are local tables within your container. You need to create the tables using source files.

Create a Table Using the .hdbtable

1. In your folder *HA300 → HDB → scr → exercises*, create a new source file with the name *Returns.hdbtable* that generates an empty runtime table *Returns*. The source code is available in your local folder *HA300 → Prepared Code → Code to build table using hdbtable.txt*.
2. Check that the new table *Returns* is generated in your local database container.

Create Tables Using Source Files

Exercise Objectives

After completing this exercise, you will be able to:

- Create a table based on a source file using .hdbtable

Business Example

The data sources for your calculation views are local tables within your container. You need to create the tables using source files.

Create a Table Using the .hdbtable

1. In your folder *HA300 → HDB → scr → exercises*, create a new source file with the name *Returns.hdbtable* that generates an empty runtime table *Returns*. The source code is available in your local folder *HA300 → Prepared Code → Code to build table using hdbtable.txt*.
 - a) Confirm that you are in the *Development* view of the SAP Web IDE.
 - b) Expand the folders *HA300 → HDB → scr → exercises*. Then, right-click and choose *New → File*.
 - c) In the *Name* field, enter **Returns.hdbtable** and choose *Create*.
 - d) Open the file *HA300 → Prepared Code → Code to build table using hdbtable.txt*. Then, copy and paste the entire contents to the source file.
 - e) Choose *Save*.
 - f) In the file navigator on the left side of the screen, right-click the new source file and choose *Build Selected Files*.
2. Check that the new table *Returns* is generated in your local database container.
 - a) Switch from the *Development* view to the *Database Explorer* view of SAP Web IDE.
 - b) Expand your database container *My HA300 Container* and select the *Tables* entry. Your new table *Returns* should be visible in the panel below.
 - c) Double-click the table to view the metadata of the new table.

Unit 5

Exercise 29

Load a Table Using the SQL Console

Exercise Objectives

After completing this exercise, you will be able to:

- Fill a table using SQL statements

Business Example

For testing purposes, you need to quickly fill some tables with data.

1. In the SAP Web IDE for SAP HANA, display the *Database Explorer* perspective and open a SQL Console for your database connection *H00 DB (CLASSIC)*.
2. In the new SQL Console tab, import the course file *HA300 → Prepared Code → SQL to Fill Tables BOOKS and PUBLISHERS.sql*.
3. Replace ## with your student number.
4. Execute the complete code in the console to fill the two new column tables *PUBLISHERS* and *BOOKS*.
5. Open the table *BOOKS* and review the records that have been inserted.
6. Check the total record count of the table *BOOKS* by displaying the table properties (Runtime Information).
7. Close all tabs.

Unit 5

Solution 29

Load a Table Using the SQL Console

Exercise Objectives

After completing this exercise, you will be able to:

- Fill a table using SQL statements

Business Example

For testing purposes, you need to quickly fill some tables with data.

1. In the SAP Web IDE for SAP HANA, display the *Database Explorer* perspective and open a SQL Console for your database connection *H00 DB (CLASSIC)*.
 - a) If needed, highlight your database connection *H00 DB (CLASSIC)*, and then choose  *Open SQL Console* in the toolbar.
2. In the new SQL Console tab, import the course file *HA300 → Prepared Code → SQL to Fill Tables BOOKS and PUBLISHERS.sql*.
 - a) In the SQL Console tab, choose  *Import File*.
 - b) Select the file *HA300 → Prepared Code → SQL to Fill Tables BOOKS and PUBLISHERS.sql* and choose *Open*.

The code is as follows:

```
SET SCHEMA STUDENT##;

insert into PUBLISHERS VALUES (1,'Oldenburg Wissenschaftsverlag GmbH','Rosenheimer Strasse 145','81671','Muenchen','Germany');
insert into PUBLISHERS VALUES (2,'Pearson Education Deutschland GmbH','Martin-Kollar-Strasse 10-12','81829','Muenchen','Germany');
insert into PUBLISHERS VALUES (3,'mitp & bhv-Buch','Augustinusstrasse 9d','50226','Frechen','Germany');
insert into PUBLISHERS VALUES (4,'Roof Music','Prinz-Regent-Strasse 50-60','44795','Bochum','Germany');

insert into BOOKS VALUES ('978-3-486-57690-0', 'Datenbanksysteme: Eine Einfuehrung', 1, 6, '2006', '39.80', 'EUR');
insert into BOOKS VALUES ('978-3-86894-012-1', 'Grundlagen von Datenbanken', 2, 3, '2009', '29.95', 'EUR');
insert into BOOKS VALUES ('978-3-8266-1664-8', 'Datenbanken: Konzepte und Sprachen', 3, 3, '2008', '39.95', 'EUR');
insert into BOOKS VALUES ('978-3-486-59002-9', 'Algorithmen: Eine Einführung', 1, 3, '2010', '79.80', 'EUR');
```

3. Replace ## with your student number.

- a) In the *SET SCHEMA STUDENT##* statement, replace ## with your assigned student number. Be careful not to remove the ; end of statement character.

4. Execute the complete code in the console to fill the two new column tables *PUBLISHERS* and *BOOKS*.

- a) Press **F8** or choose  *Run* on the toolbar.

Did you notice how we don't need to specify the column names for each value? That is because we are providing the insert values exactly in the same order as the table columns. If this were not the case, then we would have to specify the column name against its value.

5. Open the table *BOOKS* and review the records that have been inserted.

- a) In the left pane, expand the connection *H00 DB (CLASSIC)* and select the item *Catalog → Tables*.
- b) To display only relevant tables, in the lower-left pane, make sure the schema is set to *STUDENT##*.
- c) In the list of tables (lower-left pane), right-click the table *BOOKS* and choose *Open Data*.

A new tab appears where you see the records that were inserted.

6. Check the total record count of the table *BOOKS* by displaying the table properties (Runtime Information).

- a) In the list of tables, right-click again the table *BOOKS* but this time choose *Open*.
A new tab appears where you see the properties of the table.
- b) Display the tab *Runtime Information* and notice the number of records (*Total Record Count*).

7. Close all tabs.

- a) Right-click any tab and choose the context menu option *Close All*.

Unit 5

Exercise 30

Load a Table Using Source Files

Exercise Objectives

After completing this exercise, you will be able to load data into a table using a source .csv file, together with a .hdbtabledata file defining the mapping of the source data to the target SAP HANA table.

Business Example

You want to define the content of a table in SAP HANA, based on a .csv file.

1. Create the target table **HA300::OrgEmployees** in your HDI Container, by importing and building the file `src → exercises → OrgEmployees.hdbtabledata`. This file can be imported from your course files *HA300 → Prepared Code*.
2. Create a source file with the name **OrgEmployees.csv** that will be used to supply the data to the new table *OrgEmployees*. Copy and paste all records from the supplied csv file *HA300 → Prepared Code → OrgEmployees raw records for import.csv* into the new table *OrgEmployees.csv* file. Then build the source file and check the build status.



Caution:

Do not double-click the *OrgEmployees raw records for import.csv* file, because it would then open in Excel (default .csv file editor) and would parse into columns automatically. We want the raw data view.

3. In your folder *HA300 → HDB → src → exercises*, create a source file with the name *OrgEmployees.hdbtabledata*. This will be used to define the settings for the import of records from the supplied .csv file *HA300 → Prepared Code → OrgEmployees raw records for import.csv* to the new SAP HANA table *OrgEmployees*. The code is supplied in the file *HA300 → Prepared Code → Code to load data to OrgEmployees table.txt*.
4. Check that the new table *OrgEmployees* is now filled with data from the .csv file. The *HA300:OrgEmployees* is now filled with data from the .csv file you imported and build in Step 2.
5. Close all open tabs in the SAP Web IDE.

Unit 5

Solution 30

Load a Table Using Source Files

Exercise Objectives

After completing this exercise, you will be able to load data into a table using a source .csv file, together with a .hdbtabledata file defining the mapping of the source data to the target SAP HANA table.

Business Example

You want to define the content of a table in SAP HANA, based on a .csv file.

1. Create the target table **HA300 : :OrgEmployees** in your HDI Container, by importing and building the file *src → exercises → OrgEmployees.hdbtabledata*. This file can be imported from your course files *HA300 → Prepared Code*.
 - a) Make sure that you are in the *Development* view of the SAP Web IDE for SAP HANA.
 - b) In your *HA300_##* project, right-click the folder *HDB → src → exercises* and choose *Import → Files or Project*.
 - c) In the *Import* window, choose *Browse*, select the file *HA300 → Prepared Code → OrgEmployees.hdbtable* and choose *Open*.
 - d) Choose *OK*.
 - e) Right-click the imported file and choose *Build → Build Selected Files*.
 - f) In the *Console* pane, check that the build was completed successfully.
2. Create a source file with the name **OrgEmployees.csv** that will be used to supply the data to the new table *OrgEmployees*. Copy and paste all records from the supplied csv file *HA300 → Prepared Code → OrgEmployees raw records for import.csv* into the new table *OrgEmployees.csv* file. Then build the source file and check the build status.



Caution:

Do not double-click the *OrgEmployees raw records for import.csv* file, because it would then open in Excel (default .csv file editor) and would parse into columns automatically. We want the raw data view.

- a) Expand the folders *HA300 → HDB → scr → exercises*. Then right-click and choose *New → File*.
- b) In the *File Name* field, enter **OrgEmployees.csv** and choose *Create*.
- c) In Windows Explorer, right-click the file *HA300 → Prepared Code → OrgEmployees raw records for import.csv* and choose *Open with → Notepad*.
- d) Copy the entire content from Notepad and paste it into the file *OrgEmployees.csv* opened in the SAP Web IDE.

- e) Choose Save.
 - f) Right-click on this source file and choose *Build Selected Files*.
 - g) In the *Console* pane, check that the build was completed successfully.
The table *HA300::OrgEmployees* is now created in your HDI container's schema.
3. In your folder *HA300 → HDB → src → exercises*, create a source file with the name *OrgEmployees.hdbtabledata*. This will be used to define the settings for the import of records from the supplied .csv file *HA300 → Prepared Code → OrgEmployees raw records for import.csv* to the new SAP HANA table *OrgEmployees*. The code is supplied in the file *HA300 → Prepared Code → Code to load data to OrgEmployees table.txt*.
- a) Confirm that you are in the *Development* view of the SAP Web IDE.
 - b) Right-click the folder *HA300 → HDB → scr → exercises* and choose *New → File*.
 - c) In the *File Name* field, enter **OrgEmployees.hdbtabledata** and choose *Create*.
 - d) In Windows Explorer, open the file *HA300 → Prepared Code → Code to load data to OrgEmployees table.txt* and copy all its content (press **Ctrl + A** then **Ctrl + V**).
 - e) Back to the Web IDE for SAP HANA, paste the entire contents to the source file (press **Ctrl + V**).
 - f) Choose Save.
 - g) Right-click this source file and choose *Build → Build Selected Files*.
4. Check that the new table *OrgEmployees* is now filled with data from the .csv file.
- a) Switch from the *Development* view to the *Database Explorer* view of SAP Web IDE.
 - b) Expand your database container *My HA300 Container* and select the *Tables* entry.
Your new table *HA300::OrgEmployees* should be visible in the panel below.
 - c) Right-click on the table and choose *Open Data* to view the records.
- The *HA300:OrgEmployees* is now filled with data from the .csv file you imported and build in Step 2.
5. Close all open tabs in the SAP Web IDE.
- a) Right-click any open tab and choose *Close All*.

Unit 6

Exercise 31

Implement Static Cache

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.

Task 1: Create a Calculation View which Caches All Data

1. 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]

2. Assign the table *CACHED_TABLE* to the default aggregation node.
3. Map all the columns of the table.
4. Enable the static cache.
5. Save and build the calculation view.
6. Execute a data preview of the calculation view in order to fill the cache.
7. 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 *CACHEDCOLUMN* and *MEASURE* are to be cached.
2. Save and build 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 Web IDE.

Implement Static Cache

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.

Task 1: Create a Calculation View which Caches All Data

1. 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 → Calculation View*.
- b) Use the table above to complete the fields and press *Create*.
2. 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 *HA300::CACHED_TABLE* and choose *Finish*.
3. 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.
4. 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*.
 - e) Accept the pop-up warning
5. Save and build the calculation view.
 - a) Choose *File → Save*.
 - b) Choose *Build → Build Selected Files* and check the log to ensure the build is successful.
6. 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.
7. 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* (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.

Task 2: Define Specific Columns for Cache

1. Adjust the calculation view *CVC_STATIC_CACHE* by specifying that only the columns *CACHEDCOLUMN* and *MEASURE* are to be cached.
 - a) Select the tab for the definition of the *CVC_STATIC_CACHE* calculation view
 - b) Double-click on the *Semantics* node.
 - c) Click *View Properties*.
 - d) Click *Static Cache*.
 - e) in the *Columns* section, click the + button to add a new row.
 - f) Click on the drop-down selector for the new row
 - g) Enable the check-box for the column *CACHEDCOLUMN* and *MEASURE* and choose *Select* to return to the main screen.
2. Save and build the calculation view.
 - a) Choose *File* → *Save*.
 - b) Choose *Build* → *Build Selected Files* and check the log to ensure the build is successful.
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 *NOTCACHEDCOLUMN* 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 NOTCACHEDCOLUMN. The final code should look like this:

```
SELECT TOP 1000
"CACHEDCOLUMN",
-- "NOTCACHEDCOLUMN",
SUM("MEASURE") AS "MEASURE"
FROM "HA300##_HDI_HDB_1"."HA300::CVC_STATIC_CACHE"
GROUP BY "CACHEDCOLUMN"--, "NOTCACHEDCOLUMN"
```

- b) Choose *Analyze* → *Explain Plan*.
- c) 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 Web IDE.
- a) Close all tabs ready for the next exercise.

Unit 6 Exercise 32

Control Parallelization in a Data Flow

Exercise Objectives

After completing this exercise, you will be able to:

- Control parallelization in a data flow of a calculation view

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.

1. If needed, launch the SAP HANA Web IDE. A shortcut to this application is available in the following folder: HA300 → URLs.
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 Empl_Salary_Current
5. Expand the *Details* panel.
6. On the *Mapping* tab of the new *Projection* 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 column engine calculated column using the information in the table below:

Setting	Value
Name	partition_id

Column Engine Expression	partitionid()
--------------------------	---------------



Caution:

Don't forget to switch the expression type to *Column Engine* or the build will fail.



Note:

Do not use the *Validate Syntax* button as this will report an error. The *partitionid()* expression is actually valid and your calculation view will build but the Web IDE does not yet validate this new expression. It will be added in a future release of Web IDE. For now, just type carefully!

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. Save, and then build the *CVD_CONTROL_PARALLELIZATION* calculation view.
16. Preview the data of the *CVD_CONTROL_PARALLELIZATION* calculation view.

Unit 6

Solution 32

Control Parallelization in a Data Flow

Exercise Objectives

After completing this exercise, you will be able to:

- Control parallelization in a data flow of a calculation view

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.

1. If needed, launch the SAP HANA Web IDE. A shortcut to this application is available in the following folder: *HA300 → URLs*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on with your SAP HANA credentials: **STUDENT##** and **Training1** (where ## is your group number).
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 Workspace tree, right-click the folder *HA300_## → HDB → src → exercises* folder and choose *New → Calculation View*.
 - b) Enter the calculation view name and other properties as specified in the table.
 - c) Choose *Create*.
3. Add a new *Projection* node and rename the node label as **Start_Parallel**.
 - a) Drag and drop a *Projection* node from the palette to 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 Empl_Salary_Current

- a) Select the *Projection* node that you just added.
 - b) Choose *+ Add Data Source*.
 - c) In the *Find Data Sources* window, click the search field and enter **emp1**.
 - d) In the search results, select the table *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 *Projection* node, add all columns to the output.
- a) Select the new *Projection* node and on the *Mapping* tab, right-click the data source *HA300::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 *HA300::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) Drag and drop a *Projection* node from the palette to 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 column engine calculated column using the information in the table below:

Setting	Value
Name	partition_id
Column Engine Expression	partitionid()

**Caution:**

Don't forget to switch the expression type to *Column Engine* or the build will fail.

**Note:**

Do not use the *Validate Syntax* button as this will report an error. The *partitionid()* expression is actually valid and your calculation view will build but the Web IDE does not yet validate this new expression. It will be added in a future release of Web IDE. For now, just type carefully!

- 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) Drag 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. Save, and then build the *CVD_CONTROL_PARALLELIZATION* calculation view.
 - a) Choose *Save*.

- b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed successfully.
- 16. Preview the data of the *CVD_CONTROL_PARALLELIZATION* calculation view.
 - a) Right-click the *CVD_CONTROL_PARALLELIZATION* calculation view and choose *Data Preview*.
 - b) Display the *Raw Data* tab and check that you can see the partition ID generated for each distinct value found in the partition column.
 - c) To close the data preview, choose *X (Close)*.

Unit 6

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 “warm” 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.

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 the SAP HANA Web IDE. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
2. Open the Calculation View *HA300_## → HDB → 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 2018 to each employee.

1. Execute the default Data Preview and display the row data.
2. Adapt the default SQL query to return only the data for 2018, and execute it again.



Hint:

Because the data preview tab and the corresponding SQL session are 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 "HA300::CVC_EMPL_SALARY"
WHERE "Year" = 2018
GROUP BY "Year", "EmployeeID";
```

3. Analyze the SQL query.

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 2017 or 2018, 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 the SAP Web IDE.
2. Copy the original Calculation View **CVC_EMPL_SALARY** from the resources folder to your exercises folder. Rename the copy **CVC_EMPL_SALARY_PRUNING**.
3. Open the new (copied) calculation view and add a pruning configuration table with the name *Empl_Salary_Pruning*. Define the pruning conditions as follows:

Table 5: Pruning Configuration Entries

Input	Column	Operators	Low Value
HA300::Empl_Salary_Current	Year	>=	2017
HA300::Empl_Salary_Historical	Year	<	2017



Caution:

Note that 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. Save, and then build the Calculation View.

Check the build status.



Note:

If the blue progress bar is still showing up, refresh the browser tab in Chrome.

5. From the default data preview, adapt the SQL query to return only the data for 2018, as you did in Task 2, and execute 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 "HA300::CVC_EMPL_SALARY_PRUNING"
WHERE "Year" = 2018
GROUP BY "Year", "EmployeeID";
```

6. Analyze the SQL query.

How many tables have been used during the execution of the query?

7. Close all open tabs in the SAP Web IDE.

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 “warm” 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.

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 the SAP HANA Web IDE. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
 - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
 - b) Double-click the shortcut *Web IDE for SAP HANA*.
 - c) Log on with your SAP HANA credentials: **STUDENT##** and **Training1** (where ## is your group number).
2. Open the Calculation View *HA300_## → HDB → src → resources → CVC_EMPL_SALARY*.

- a) In the *Development* perspective, navigate to the folder *HA300_## → HDB → src → resources*.
 - b) Double-click 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: *HA300::Empl_Salary_Current* and *HA300::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, right-click the data source *HA300::Empl_Salary_Current* and choose *Data Preview*.
- b) Repeat the previous step for the other data source, *HA300::Empl_Salary_Historical*.

To which years does the data from each data source correspond?

Current data corresponds to the years 2017 and 2018, while historical data corresponds to the years 2015 and 2016.

Task 2: Execute a Query on Top of the Calculation View

You want to query the total wages paid in 2018 to each employee.

1. Execute the default Data Preview and display the row data.
 - a) From the *Workspace* navigator on the left of the screen, select the calculation view *CVC_EMPL_SALARY* and choose *Data Preview*.
 - b) Select the *Raw Data* tab.
 - c) Note that the data from both sources is combined together in the result set (salaries for the years 2015 to 2018).
2. Adapt the default SQL query to return only the data for 2018, and execute it again.



Hint:

Because the data preview tab and the corresponding SQL session are 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 "HA300::CVC_EMPL_SALARY"
WHERE "Year" = 2018
GROUP BY "Year", "EmployeeID";
```

- From the Raw Data tab of the default data preview that you opened in the previous step, choose *Edit SQL Statement In SQL Console*.
- Remove the `TOP 1000` clause.
- Add the following `WHERE` clause to the SQL query, between the `FROM` and the `GROUP BY` lines.
`WHERE "Year"=2018`
- Optionally, remove the container schema name `"HA300_##_HDI_HDB_1"`, including the `.` (dot) separator.
- To execute the query, choose *Run*.
- Check that the query result is now filtered, showing only the data for 2018.

3. Analyze the SQL query.

- Select the small arrow at the right of the *(Run)* icon and choose *Analyze SQL*.
- Choose the *Tables in Use* tile.

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 2018 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 2017 or 2018, 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 the SAP Web IDE.
 - 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**.
 - a) In the workspace of the *Development* perspective, 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) At the *Naming Conflict* pop-up, change the name of the calculation view to **CVC_EMPL_SALARY_PRUNING.hdbcalculationview**.
3. Open the new (copied) calculation view and add a pruning configuration table with the name *Empl_Salary_Pruning*. Define the pruning conditions as follows:

Table 5: Pruning Configuration Entries

Input	Column	Operators	Low Value
HA300::Empl_Salary_Current	Year	>=	2017
HA300::Empl_Salary_Historical	Year	<	2017



Caution:

Note that 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) In your workspace, double-click 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*.
4. Save, and then build the Calculation View.

Check the build status.



Note:

If the blue progress bar is still showing up, refresh the browser tab in Chrome.

- a) Choose Save.
 - b) Choose *Build* → *Build Selected Files*.
 - c) In the *Console* pane, check that the build was completed successfully.
5. From the default data preview, adapt the SQL query to return only the data for 2018, as you did in Task 2, and execute 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 "HA300::CVC_EMPL_SALARY_PRUNING"
WHERE "Year" = 2018
GROUP BY "Year", "EmployeeID";
```

- a) From the *Workspace* navigator on the left of the screen, select the calculation view *CVC_EMPL_SALARY_PRUNING* and choose *Data Preview*.
 - b) On the *Raw Data* tab, choose *Edit SQL Statement In SQL Console*.
 - c) Remove the `TOP 1000` clause.
 - d) Add the following *WHERE* clause to the SQL query, between the *FROM* and the *GROUP BY* lines:
- ```
WHERE "Year"=2018
```
- e) Optionally, remove the container schema name "`HA300##HDI_HDB_1`"., including the `.` (dot) separator.
  - f) To execute the query, choose *Run*.
  - g) Check that the query result is filtered, showing only the data for 2018.

6. Analyze the SQL query.

- a) Select the small arrow at the right of the *(Run)* icon and choose *Analyze SQL*.
- b) Choose the *Tables in Use* tile.

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.

7. Close all open tabs in the SAP Web IDE.

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

## Unit 6

### Exercise 34

# Access a Calculation View in 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.

1. If needed, launch the SAP HANA Web IDE. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
2. In the *Preferences* of SAP Web IDE, set the *Performance Analysis Mode* threshold value for the number of table records to 20000.
3. View the definition of the calculation view *resources* → CVCS\_DELAY in *Performance Analysis Mode*.
4. Why do you now see a warning triangle inside the join node?
5. How many rows does the large *On\_Time* data source contain?
6. How might you optimize this table?
7. There are two more warnings relating to potential performance issues. Can you locate them and describe the issues?
8. View the definition of the calculation view *resources* → CVC\_BANK\_CHURN using the *Performance Analysis Mode*.
9. Which is the largest partition in the table *BANK\_WITH\_PARTITIONS*?
10. There is a performance validation warning that informs you that you have not followed an SAP modeling recommendation. What is this warning?
11. In the *Preferences* of SAP Web IDE, set the *Performance Analysis Mode* threshold value for the number of table records back to 20000.
12. Close all open tabs in the SAP Web IDE.

# Access a Calculation View in 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.

1. If needed, launch the SAP HANA Web IDE. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
  - a) Start Windows Explorer and then navigate to the folder HA300 → URLs.
  - b) Double-click the shortcut *Web IDE for SAP HANA*.
  - c) Log on with your SAP HANA credentials: **STUDENT##** and **Training1** (where ## is your group number).
2. In the *Preferences* of SAP Web IDE, set the *Performance Analysis Mode* threshold value for the number of table records to 20000.
  - a) From the menu of SAP Web IDE, select *Tool* → *Preferences* → *Modeler*.
  - b) In the *Threshold Value* field, enter **20000**.
  - c) Choose *Save*.
3. View the definition of the calculation view *resources* → CVCS\_DELAY in *Performance Analysis Mode*.
  - a) Double-click the calculation view *resources* → CVCS\_DELAY 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*'.
4. Why do you now see a warning triangle inside the join node?
  - a) This is a warning to let you know that the table *On\_Time* exceeds the number of records defined in the modeler preferences.
5. How many rows does the large *On\_Time* data source contain?
  - a) From the menu of SAP Web IDE, select *Tool* → *Preferences* → *Modeler*.
  - b) In the *Threshold Value* field, enter **50000000** (50 millions).
  - c) Choose *Save*.
  - d) Close and reopen the CVCS\_DELAY calculation view and enable *Performance Analysis mode*.
  - e) Double-click the header for the *Join\_1* node and in the *Data Source Details* pane notice the number of rows for the *On\_Time* table is 45,619,531.

6. How might you optimize this table?
  - a) Consider partitioning the table according to the data selections that are made by users, for example, by region / year..
7. There are two more warnings relating to potential performance issues. Can you 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.
  - b) Make sure the *Join\_1* node is selected and switch to the *Join Definition* tab and you will see that an orange triangle appears over the join to indicate that you did not use the proposed join cardinality. Expand the *Properties* section to view the chosen cardinality.
8. View the definition of the calculation view resources → *CVC\_BANK\_CHURN* using the *Performance Analysis Mode*.
  - a) Double-click the calculation view resources → *CVC\_BANK\_CHURN* 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*'.
9. Which is the largest partition in the table *BANK\_WITH\_PARTITIONS*?
  - a) Double-click the node *Join\_1*.
  - b) Select the tab *Performance Analysis*.
  - c) In the *Partition Detail* section of the screen, you will see there are 6,373 records in the partition Range: 35–50 which makes it the largest partition.
10. There is a performance validation warning that informs you that you have not followed an SAP modeling recommendation. What is this warning?
  - a) 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.
11. In the *Preferences* of SAP Web IDE, set the *Performance Analysis Mode* threshold value for the number of table records back to 20000.
  - a) From the menu of SAP Web IDE, select *Tool* → *Preferences* → *Modeler*.
  - b) In the *Threshold Value* field, enter **20000**.
  - c) Choose *Save*.
12. Close all open tabs in the SAP Web IDE.
  - a) Right-click any open tab and choose *Close All*.

# Unit 6

## Exercise 35

### Use 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.

1. From the folder *resources*, open the calculation view CVCS\_SO in Debug Query mode.
2. Launch the default debug query to generate SQL at each node.
3. Switch to the *Join\_1* node and execute the generated SQL to view the intermediate results.
4. 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.
5. Close the Calculation View CVCS\_SO before the next step (you will reopen it and use a different query).
6. 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.
7. Review the columns that are grayed out due to pruning at the Semantics node and also at the Star Join node.
8. Display the list of columns in the semantics.

What do you notice about the columns?

---

---

---

9. Close all open tabs in the SAP Web IDE.

# Unit 6

## Solution 35

### Use 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.

1. From the folder *resources*, open the calculation view CVCS\_SO in Debug Query mode.
  - a) In the *Development* view of SAP Web IDE, expand the folder *resources*.
  - b) Double-click 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).
2. 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 wont see any data yet as this step simply generates the SQL at each node of the calculation view.
3. Switch to the *Join\_1* node and execute the generated SQL to view the intermediate results.
  - a) Highlight the *Join\_1* node.
  - b) Just above the SQL code and to the right, choose  *Execute*.
  - c) View the results below the SQL code.
4. 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.
5. Close the Calculation View CVCS\_SO before the next step (you will reopen it and use a different query).
  - a) Click the  *Close* button on the corresponding tab.
6. 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) Double-click 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 "HA300_##_HDI_HDB_1"."HA300::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
7. 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*.
8. Display the list of columns in the semantics.
- a) Select the *Semantic* node and then select the *Columns* tab.

What do you notice about the columns?

The columns that are not required by the debug query are grayed out in the *Columns* tab.

9. Close all open tabs in the SAP Web IDE.
- a) Right-click any open tab and choose *Close All*.

## Unit 7

### 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

#### Business Example

You are working on a dimension with calculation view 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.

#### 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 *HA300::SNWD\_SO* table or the *HA300::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 view resources → CVD\_PD (Products). 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\_PD*.
2. What other models reference the *CVD\_PD* view?
3. Close all the open tabs in the SAP Web IDE.

# 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

## Business Example

You are working on a dimension with calculation view 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.

### 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 HA300\_## project, navigate to the resources folder.
  - b) Double-click the CVCS\_SO calculation view.
  - c) In the Semantics, display the *Columns* tab.
  - d) If needed, use the  *Toolbar Overflow* button.
  - e) 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 HA300::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 *HA300::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 *HA300::SNWD\_SO* table or the *HA300::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 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 *HA300::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 *CVC\_SO* and choose *Close*.
- Task 2: Analyze the Impact of a View Modification on Other Views**  
You have been asked to modify the view *resources* → *CVD\_PD* (Products). 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\_PD*.  
a) In your *HA300\_##* project, right-click the *resources* → *CVD\_PD* view and choose *Modeling* → *Impact Analysis*.
  2. What other models reference the *CVD\_PD* view?  
a) Choose the *Expand* button.  
b) The *CVD\_PD* is consumed by the *CVCS\_PO\_00* and the *CVCS\_SO* view. Depending on other exercises you have already completed, you might see other models that depend on the *CVD\_PD* view.
  3. Close all the open tabs in the SAP Web IDE.  
a) Right-click any open tab and choose *Close All*.

## Unit 7

### Exercise 37

# Import, Rename and Copy Models

### Exercise Objectives

After completing this exercise, you will be able to:

- Import models
- Build models
- Rename a model
- Identify common build errors when importing, moving, and copying models

### Business Example

You are working as a modeler in the SAP Web IDE for SAP HANA, 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 build issues that can occur when building 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

### 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 the SAP HANA SAP Web IDE. A shortcut to this application is available in the following folder: *HA300 → URLs → SAP Web IDE for SAP HANA*.
2. Clear the SAP Web IDE console.
3. In the workspace, import the *import.zip* file from your course files folder *HA300* into your workspace folder *HA300\_## → HDB → src → exercises*.



Caution:

You must add the destination folder */import* to the *Import to* field.

4. Build the *import* folder.

5. What do you observe?

---

---

---

6. Analyze the root cause by checking the log from the start of the build until you see an error report.

```
Error: "calc.scenario://HA300::CVCS_SO": the object cannot be provided more than once [8212002]
"src/exercises/import/CVCS_SO.hdbcalculationview": the file would provide it
"src/resources/CVCS_SO.hdbcalculationview": the deployed file already provides it
```

- Why did the build fail?

---

---

---

7. In the Database Explorer, check whether the view *CVD\_PD2* has been created in your HA300 container.

8. Has the *CVD\_PD2* view been created? Why?

---

---

---

9. Try to build the calculation view *CVD\_PD2* separately.

10. Has the build been successful? Why?

---

---

---

### Task 2: Rename a View

To solve the issue for the view CVCS\_SO, you will rename it.

1. Rename the import CVCS\_SO Calculation View as **cvcso\_so2**. You will only rename the design-time file and adjust the runtime object name accordingly.
2. (Optional) Analyze the console log to check that the build of CVCS\_SO2 has been successful.

```
(...)
Deploying "src/exercises/import/CVCS_SO2.hdbcalculationview"...
(...)
Deployment to container HA300_##_HDI_HDB_1 done
```



Note:

A flag *Pending Development* might still be shown next to the *CVCS\_SO2* calculation view even if it has been successfully built. If you want to remove this flag, just trigger a build of the view from the workspace.

3. In the Database Explorer, confirm that the view CVCS\_SO2 is now created in your HA300\_## container.
4. Close all open tabs in the SAP Web IDE.

### 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 *Development* perspective.
2. Clear the SAP Web IDE console.
3. Import the *import2.zip* file from your course files folder HA300 into your workspace folder HA300\_## → HDB → src → exercises → *import2*.



**Caution:**

Similarly to task 1, you must add the destination folder `/import` to the *Import to* field.

4. Build the *import2* folder.

Check the build status.

5. Was the build successful?

---



---



---

6. Observe carefully the content of the *import2* folder.



**Hint:**

Make sure the hidden files display in the workspace (*View* → *Show hidden files*).

The imported folder contains a *CVCS\_SO.hdbculationview* file, but this time the build 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 Web IDE 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) and check the adjusted name in the *Refactor Views* window.



Hint:

You can use the Cut/Paste shortcuts or the context menu.

3. In the Database Explorer, confirm that the view *HA300.import2::CVCS\_SO2* is now created in your *HA300\_##* container.
4. Search the list of column views for the old *CVCS\_SO2* view.

Was the *HA300::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, build the import folder.  
Check the build status.
6. In the Database Explorer, confirm that the view *HA300::CVCS\_SO2* has been deleted.
7. Close all open tabs in the SAP Web IDE.

# Import, Rename and Copy Models

## Exercise Objectives

After completing this exercise, you will be able to:

- Import models
- Build models
- Rename a model
- Identify common build errors when importing, moving, and copying models

## Business Example

You are working as a modeler in the SAP Web IDE for SAP HANA, 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 build issues that can occur when building 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

### 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 the SAP HANA SAP Web IDE. A shortcut to this application is available in the following folder: *HA300 → URLs → SAP Web IDE for SAP HANA*.
  - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
  - b) Double-click the shortcut *SAP Web IDE for SAP HANA*.
  - c) Log on with your SAP HANA credentials: **STUDENT## / Training1** (## is your group number).
2. Clear the SAP Web IDE console.
  - a) Choose *View → Clear Console*.
3. In the workspace, import the *import.zip* file from your course files folder *HA300* into your workspace folder *HA300\_## → HDB → src → exercises*.

**Caution:**

You must add the destination folder `/import` to the *Import to* field.

- a) In the workspace, right-click your exercises folder and choose *import* → *File or Project*.
  - b) In the *Import* window, choose *Browse*.
  - c) Select the *HA300* → *import.zip* file and choose *Open*.
  - d) Add the folder `/import` at the end of the *Import to* location and check that it ends with `.../exercises/import`.
  - e) Check that the *Extract Archive* checkbox is selected and choose *OK*.  
An *import* folder is created in your project. It will be used specifically for this exercise.
4. Build the *import* folder.
- a) Right-click the new *Import* folder and choose *Build Selected Files*.
5. What do you observe?

The build fails.

---

6. Analyze the root cause by checking the log from the start of the build until you see an error report.

```
Error: "calc.scenario://HA300::CVCS_SO": the object cannot be provided more than once [8212002]
"src/exercises/import/CVCS_SO.hdbcalculationview": the file would provide it
"src/resources/CVCS_SO.hdbcalculationview": the deployed file already provides it
```

Why did the build fail?

A *CVCS\_SO* object already exists with the *HA300* namespace.

---

7. In the Database Explorer, check whether the view *CVD\_PD2* has been created in your *HA300* container.
- a) Choose *Tools* → *Database Explorer*.  
Alternatively, click the *Database Explorer* icon in the left toolbar.
  - b) In the top-left pane, select *MY HA300 CONTAINER* → *ColumnViews*.
  - c) In the *Search* field, enter ***CVD\_PD2***.
8. Has the *CVD\_PD2* view been created? Why?

The *CVD\_PD2* view has not been built. This is because when you request the build of an entire folder in your project, either all the runtime files can be built successfully, or no file at all is built.

---

9. Try to build the calculation view *CVD\_PD2* separately.

- a) Choose *Tools → Development*.  
Alternatively, click the *Development* icon in the left toolbar.
  - b) Right-click the *import → CVD\_PD2* view and choose *Build Selected Files*.
10. Has the build been successful? Why?

Yes. The *CVD\_PD2* view has been created because it was the only design-time file included in the scope of the build, and this one does not generate any build error (conflict in runtime object name or other build issue).

### Task 2: Rename a View

To solve the issue for the view *CVCS\_SO*, you will rename it.

1. Rename the import *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) In the *Rename File* dialog box, modify the new name to  
***CVCS\_SO2.hdbcaculationview***
  - c) Choose *Rename*.
  - d) In the confirmation dialog box, choose *Yes*.  
The *Refactor Views* window shows that there are no views in which references to *CVCS\_SO* should be adapted.
  - e) Choose *Refactor*.  
Again, the *Refactor Views* window shows that no impacted views have been modified.
  - f) Keep the *Build impacted views* checkbox selected.



#### Note:

This setting applies not only to the "impacted views" (views in which references to the renamed view should be adjusted) but also to the view that you are renaming. So you must select it if you want the build of the renamed view to be triggered as part of the renaming process. If you do not, you can of course build the renamed view later on.

- g) Choose *Finish*.
2. (Optional) Analyze the console log to check that the build of *CVCS\_SO2* has been successful.

```
(...)
Deploying "src/exercises/import/CVCS_SO2.hdbcaculationview"...
(...)
Deployment to container HA300_##_HDI_HDB_1 done
```

- a) Consult the last 10–15 rows of the log in the console and confirm you see the entries provided.

**Note:**

A flag *Pending Development* might still be shown next to the CVCS\_SO2 calculation view even if it has been successfully built. If you want to remove this flag, just trigger a build of the view from the workspace.

3. In the Database Explorer, confirm that the view CVCS\_SO2 is now created in your HA300\_## container.
  - a) Choose *Tools* → *Database Explorer*.  
Alternatively, click the *Database Explorer* icon in the left toolbar.
  - b) In the top-left pane, select *MY HA300 CONTAINER* → *ColumnViews*.
  - c) In the Search field, enter **cvcs\_so2**.
  - d) Make sure there is a view *HA300::CVCS\_SO2* in the results.
4. Close all open tabs in the SAP Web IDE.
  - a) 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 *Development* perspective.
  - a) Choose *Tools* → *Development*.  
Alternatively, click the *Development* icon in the left toolbar.
2. Clear the SAP Web IDE console.
  - a) Choose *View* → *Clear Console*.
3. Import the *import2.zip* file from your course files folder HA300 into your workspace folder HA300\_## → HDB → src → exercises → import2.

**Caution:**

Similarly to task 1, you must add the destination folder **/import** to the *Import to* field.

- a) In the workspace, right-click your *exercises* folder and choose *Import* → *File or Project*.
  - b) In the *Import* window, choose *Browse*.
  - c) Select the *HA300* → *import2.zip* file and choose *Open*.
  - d) Add the folder **/import2** at the end of the *Import to* location and check that it ends with .../exercises/import2.
  - e) Check that the *Extract Archive* checkbox is selected and choose *OK*.  
An *import2* folder is created in your project. It will be used specifically for this exercise.
4. Build the *import2* folder.  
Check the build status.

- a) Right-click the new *import2* folder and choose *Build Selected Files*.
  - b) Consult the build log results in the *Console* pane.
5. Was the build successful?

Yes.

---

6. Observe carefully the content of the *import2* folder.



**Hint:**

Make sure the hidden files display in the workspace (*View → Show hidden files*).

The imported folder contains a *CVCS\_SO.hdbculationview* file, but this time the build was successful. Can you explain why?

The *import2* folder contains a *.hdinamespace* file with a namespace prefix *HA300.import2*, which is different from the namespace *HA300* 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 runtime object has the same name (apart from the namespace).

---

- a) If needed, choose *View → Show hidden files*.
- b) Double-click the *import2 → .hdinamespace* file and observe its content.
- c) Double-click 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 *HA300.import2*.
- d) Alternatively, you can right-click the *import2 → CVCS\_SO* view and choose *Open With → Code Editor*. Then check the identifier of the runtime view in the second row.



**Note:**

The *.hdinamespace* file must be built before, or at least at the same time as, the content of the corresponding folder. For example, if you had built 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 Web IDE console.

- a) Choose *View* → *Clear 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) and check the adjusted name in the *Refactor Views* window.



Hint:

You can use the Cut/Paste shortcuts or the context menu.

- a) In the workspace, 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.
  - e) In the *Refactor Views* window, check that the namespace of the view has been adjusted to *HA300.import2*, according to the namespace rule of the destination folder.
  - f) Choose *Refactor*.
  - g) Choose *Finish*.  
The refactored calculation view is built automatically.
3. In the Database Explorer, confirm that the view *HA300.import2::CVCS\_SO2* is now created in your *HA300\_##* container.
    - a) Choose *Tools* → *Database Explorer*.  
Alternatively, click the *Database Explorer* icon in the left toolbar.
    - b) In the top-left pane, select *MY HA300\_## CONTAINER* → *ColumnViews*.
    - c) In the *Search* field, enter **so2**.
    - d) Make sure there is a view *HA300.import2::CVCS\_SO2*.
  4. Search the list of column views for the old *CVCS\_SO2* view.
    - a) In the Database Explorer, refresh the search results for SO2 that you did at the previous step.

Was the *HA300::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 built successfully.

5. To finalize the move of the view, that is, delete the original runtime object, build the import folder.  
Check the build status.
  - a) In the *Development* perspective, select the import folder and choose *Build selected files*.
  - b) In the *Console* pane, check that the build was completed successfully.

6. In the Database Explorer, confirm that the view *HA300::CVCS\_SO2* has been deleted.
  - a) Choose *Tools* → *Database Explorer*.
  - b) If needed, in the top-left pane, select *MY HA300\_## CONTAINER* → *ColumnViews*.
  - c) If needed, in the *Search* field, enter **so2**.
  - d) Note that the view *HA300::CVCS\_SO2* is no longer listed in the search results.
7. Close all open tabs in the SAP Web IDE.
  - a) Right-click any open tab and choose *Close All*.

## Unit 7

### 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
- Build the project

#### Business Example

You have been working on an SAP HANA project as a modeler in the SAP Web IDE 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 DB module, and namespace configuration, by creating a simple project and reviewing its key properties.

#### Task 1: Create the Project and Database Module

1. If needed, launch the SAP HANA Web IDE. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
2. In the workspace, create a new project based on the *SAP HANA Database Application* template, with the following properties:

Table 6: New Project Properties

| Field               | Value                                         |
|---------------------|-----------------------------------------------|
| Project Name        | <b>MyProject_##</b> (## is your group number) |
| Application ID      | <b>MyApplication_##</b>                       |
| Application Version | <b>0.0.1</b>                                  |
| Description         | <b>My new app</b>                             |
| Space               | <b>DEV</b>                                    |

Table 7: New Database Module Properties

| Field                     | Value                                         |
|---------------------------|-----------------------------------------------|
| Namespace                 | <b>myproj.db</b>                              |
| Schema Name               | <b>MYSCHHEMA_##</b> (## is your group number) |
| SAP HANA Database Version | <b>2.0 SPS 06</b>                             |

#### Task 2: Finalize the Project and Build the HDB Module

1. Review the content of the *mta.yaml* located at the root of your project.
2. Adjust the *db → package.json* file so that the most recent HDI-Deploy application can be used, which requires a minimum version of the NodeJS runtime.  
You need to add an "**engines**" entry, and the modified content of the *package.json* file should be as follows:

```
{
 "name": "deploy",
 "engines": {
 "node": "^14"
 },
 "dependencies": {
 "@sap/hdi-deploy": "^4"
 },
 "scripts": {
 "start": "node node_modules/@sap/hdi-deploy/deploy.js"
 }
}
```

3. Create the following sub-folders within the *src* folder to organize the design-time content of your HDB module:
  - *data*
  - *models*
  - *synonyms*
  - *test*
4. Build the HDB Module *db*.



Caution:

Make sure you select the *db* folder. Building the entire project does NOT build the HDB module.

## Unit 7 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
- Build the project

### Business Example

You have been working on an SAP HANA project as a modeler in the SAP Web IDE 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 DB module, and namespace configuration, by creating a simple project and reviewing its key properties.

#### Task 1: Create the Project and Database Module

1. If needed, launch the SAP HANA Web IDE. A shortcut to this application is available in the following folder: *HA300 → URLs → Web IDE for SAP HANA*.
  - a) Start Windows Explorer and navigate to the folder *HA300 → URLs*.
  - b) Double-click the shortcut *Web IDE for SAP HANA*.
  - c) Log on with your SAP HANA credentials: **STUDENT## / Training1** (## is your group number).
2. In the workspace, create a new project based on the *SAP HANA Database Application* template, with the following properties:

Table 6: New Project Properties

| Field               | Value                                         |
|---------------------|-----------------------------------------------|
| Project Name        | <b>MyProject_##</b> (## is your group number) |
| Application ID      | <b>MyApplication_##</b>                       |
| Application Version | <b>0.0.1</b>                                  |
| Description         | <b>My new app</b>                             |
| Space               | <b>DEV</b>                                    |

Table 7: New Database Module Properties

| Field     | Value            |
|-----------|------------------|
| Namespace | <b>myproj.db</b> |

| Field                     | Value                                         |
|---------------------------|-----------------------------------------------|
| Schema Name               | <b>MYSCHHEMA_##</b> (## is your group number) |
| SAP HANA Database Version | 2.0 SPS 06                                    |

- Right-click your workspace root (*Workspace*) and choose *New → Project from Template*.
- Select the *SAP HANA Database Application* project template and choose *Next*.
- Enter the project properties according to the table and choose *Next*.
- Confirm the project name and choose *Finish*.

### Task 2: Finalize the Project and Build the HDB Module

- Review the content of the *mta.yaml* located at the root of your project.

- Right-click the file *mta.yaml* and choose *Open with → Code Editor*.



Note:

You can also double-click the *mta.yaml* file, which opens it with a dedicated user interface.

- 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-container*.
- Adjust the *db → package.json* file so that the most recent HDI-Deploy application can be used, which requires a minimum version of the NodeJS runtime.

You need to add an **"engines"** entry, and the modified content of the *package.json* file should be as follows:

```
{
 "name": "deploy",
 "engines": {
 "node": "^14"
 },
 "dependencies": {
 "@sap/hdi-deploy": "^4"
 },
 "scripts": {
 "start": "node node_modules/@sap/hdi-deploy/deploy.js"
 }
}
```

- In the left pane, in *MyProject\_##*, open the *db → package.json → file*.
- Modify the *package.json* file by adding, after the *"name"* entry, the following **"engines"** entry to specify the nodeJS engine :

```
"engines": {
 "node": "^14"
},
```

- Choose *File → Save*.

3. 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 choose *OK*.

c) Repeat the previous steps for the other folders.

4. Build the HDB Module *db*.



Caution:

Make sure you select the *db* folder. Building the entire project does NOT build the HDB module.

a) Right-click the folder *MyProject\_##* → *db* and choose *Build*.

b) Check the build status in the Console.

## Unit 7

### 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: Configure the Connection to External DB Schemas
- Task 2: Define Synonyms and Authorization to the External Schemas
- Task 3: Check That Your Synonyms are Correctly Defined

#### Task 1: Configure the Connection to External DB Schemas

A dedicated service has been deployed in the *DEV* space by an administrator, in order to access data from classic database schemas in your database. This service name is *CROSS\_SCHEMA\_ACCESS*.

In order to reference this service in your project, you must adapt the *mta.yaml* file.



##### Note:

A copy of the final *mta.yaml* file is located in your *HA300 → New Project Files* folder, for your reference.

1. If needed, open the *mta.yaml* file located at the root of your *MyProject\_##* application.
2. Declare the resource for the external schema access by adding the following code at the very bottom of the *mta.yaml* file, in the *resources* section:

```
- name: ext-schema-service
 type: org.cloudfoundry(existing-service)
 parameters:
 service-name: CROSS_SCHEMA_SERVICE
```



##### Caution:

The *.yaml* file type relies on spaces to indent the sections and sub-sections of the code. You must NOT use the **Tab** key. Make sure that you always align the code as indicated. This is valid whenever you modify the *mta.yaml* file in this exercise. Most of the indentation errors are materialized in the code editor.

3. Declare that the HDB module depends on this resource by adding the following code to the *requires* section of the HDB module (just after the `- name: hdi_db` row):

```
- name: ext-schema-service
```

4. Adjust the properties of the HDI container service in the *requires* section of the HANA DB module to specify the *TARGET\_CONTAINER*. This is needed because the db module now has more than one dependency, so you must specify which of them is the actual hdi container service. To do so, add the following code, just after the `- name: hdi_db` row:

```
properties:
 TARGET_CONTAINER: ~{hdi-container-name}
```

5. Check carefully the content of the *mta.yaml* file.

The code should look like this (## is your group number):

```
ID: MyApplication_##
schema-version: '2.0'
description: My new app
version: 0.0.1

modules:
- name: db
 type: hdb
 path: db
 requires:
 - name: hdi_db
 properties:
 TARGET_CONTAINER: ~{hdi-container-name}
 - name: ext-schema-service

resources:
- name: hdi_db
 parameters:
 config:
 schema: MYSHEMA_##
 properties:
 hdi-container-name: ${service-name}
 type: com.sap.xs.hdi-container
- name: ext-schema-service
 type: org.cloudfoundry.existing-service
 parameters:
 service-name: CROSS_SCHEMA_SERVICE
```

6. Save the *mta.yaml* file

7. Build the *db* module and check the build status.



**Note:**

At this stage, the access to an external schema is not yet operational, because you have only referenced the corresponding service, but not defined any synonym or authorization.

### Task 2: Define Synonyms and Authorization to the External Schemas

You must configure synonyms and authorization in order to access the data from any schema outside of your container.

1. 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?

---



---



---

2. How can you modify the namespace configuration only for the *synonyms* folder?

---



---



---

3. 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.

4. In your *db* → *src* → *synonym* folder, create a design-time **TRAINING.hdbsynonym** file that will store synonyms for tables located in the *TRAINING* schema.
5. Add a synonym for the table *SALES\_DATA* in the *TRAINING* schema with the following properties:

Table 8: Synonym Definition

| Column        | Value             |
|---------------|-------------------|
| Synonym Name  | <b>SALES_DATA</b> |
| Object Name   | <b>SALES_DATA</b> |
| Schema Name   | <b>TRAINING</b>   |
| Database Name | <b>H00</b>        |

6. Import another set of synonyms, for two tables of the *EPM\_MODEL* schema.  
A design-time file, *EPM\_MODEL.hdbsynonym*, is located in your exercise folder, *HA300* → *New Project Files*, and ready for import.
7. Define the authorizations you want to grant to both the object owner and to the application user on objects.  
A design-time file, *ExternalSchemaAccess.hdbgrants*, is located in your exercise folder, *HA300* → *New Project Files*, and ready for import into the *synonyms* folder.
8. Build the entire *synonyms* folder and check the build 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 *SALES\_DATA* 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           | <b>SALES</b>   |
| Label          | <b>Sales</b>   |
| Data Category  | <i>CUBE</i>    |
| With Star Join | [Not selected] |

2. Add the table *SALES\_DATA* table to the *Aggregation* node.
3. Add all the columns to the output
4. Save, and then build the new calculation view.
5. Preview the data of the *SALES* calculation view.
6. Close all open tabs in the SAP Web IDE.

# 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: Configure the Connection to External DB Schemas
- Task 2: Define Synonyms and Authorization to the External Schemas
- Task 3: Check That Your Synonyms are Correctly Defined

### Task 1: Configure the Connection to External DB Schemas

A dedicated service has been deployed in the *DEV* space by an administrator, in order to access data from classic database schemas in your database. This service name is *CROSS\_SCHEMA\_ACCESS*.

In order to reference this service in your project, you must adapt the *mta.yaml* file.



#### Note:

A copy of the final *mta.yaml* file is located in your *HA300 → New Project Files* folder, for your reference.

1. If needed, open the *mta.yaml* file located at the root of your *MyProject\_##* application.
  - a) Double-click the *mta.yaml* file and display the *Code Editor* tab.
2. Declare the resource for the external schema access by adding the following code at the very bottom of the *mta.yaml* file, in the *resources* section:

```
- name: ext-schema-service
 type: org.cloudfoundry(existing-service
 parameters:
 service-name: CROSS_SCHEMA_SERVICE
```

**Caution:**

The .yaml file type relies on spaces to indent the sections and sub-sections of the code. You must NOT use the **Tab** key. Make sure that you always align the code as indicated. This is valid whenever you modify the mta.yaml file in this exercise. Most of the indentation errors are materialized in the code editor.

3. Declare that the HDB module depends on this resource by adding the following code to the *requires* section of the HDB module (just after the `- name: hdi_db` row):
 

```
- name: ext-schema-service
```
4. Adjust the properties of the HDI container service in the *requires* section of the HANA DB module to specify the *TARGET\_CONTAINER*. This is needed because the db module now has more than one dependency, so you must specify which of them is the actual hdi container service. To do so, add the following code, just after the `- name: hdi_db` row:
 

```
properties:
 TARGET_CONTAINER: ~{hdi-container-name}
```

5. Check carefully the content of the *mta.yaml* file.

The code should look like this (## is your group number):

```
ID: MyApplication_##
schema-version: '2.0'
description: My new app
version: 0.0.1

modules:
- name: db
 type: hdb
 path: db
 requires:
 - name: hdi_db
 properties:
 TARGET_CONTAINER: ~{hdi-container-name}
 - name: ext-schema-service

resources:
- name: hdi_db
 parameters:
 config:
 schema: MYSHEMA_##
 properties:
 hdi-container-name: ${service-name}
 type: com.sap.xs.hdi-container
- name: ext-schema-service
 type: org.cloudfoundry(existing-service)
 parameters:
 service-name: CROSS_SCHEMA_SERVICE
```

6. Save the *mta.yaml* file
  - a) Choose *Save*.
7. Build the *db* module and check the build status.
  - a) Right-click the *db* module and choose *Build*.
  - b) In the Console, check that the build was successful.

**Note:**

At this stage, the access to an external schema is not yet operational, because you have only referenced the corresponding service, but not defined any synonym or authorization.

**Task 2: Define Synonyms and Authorization to the External Schemas**

You must configure synonyms and authorization in order to access the data from any schema outside of your container.

1. 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.

2. 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.

3. 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.

- Select the *src* → *.hdinamespace* file and press **Ctrl+C** (copy).
  - Select the *src* → *synonyms* folder and press **Ctrl+V** (paste).
  - Open the new *.hdinamespace* file and adjust its content.
  - Choose Save.
- In your *db* → *src* → *synonym* folder, create a design-time **TRAINING.hdbsynonym** file that will store synonyms for tables located in the *TRAINING* schema.
    - Right-click the *synonyms* folder and choose *New* → *File*.
    - Enter the file name **TRAINING.hdbsynonym** and choose *OK*.  
The new *.hdbsynonym* file opens.
  - Add a synonym for the table *SALES\_DATA* in the *TRAINING* schema with the following properties:

Table 8: Synonym Definition

| Column       | Value             |
|--------------|-------------------|
| Synonym Name | <b>SALES_DATA</b> |

| Column        | Value             |
|---------------|-------------------|
| Object Name   | <b>SALES_DATA</b> |
| Schema Name   | <b>TRAINING</b>   |
| Database Name | <b>H00</b>        |

- a) In the synonym editor, choose <Click to Add>.
- b) Enter the synonym properties as per the table.
- c) Choose Save.
6. Import another set of synonyms, for two tables of the *EPM\_MODEL* schema.  
A design-time file, *EPM\_MODEL.hdbsynonym*, is located in your exercise folder, *HA300 → New Project Files*, and ready for import.
- a) Right-click the *synonyms* folder and choose *Import → File or Project*.
  - b) Click *Browse*, select the file *HA300 → New Project Files → EPM\_MODEL.hdbsynonym*, and choose *Open*.
  - c) Check that the *Import to* location is your *synonyms* folder and choose *OK*.
7. Define the authorizations you want to grant to both the object owner and to the application user on objects.  
A design-time file, *ExternalSchemaAccess.hdbgrants*, is located in your exercise folder, *HA300 → New Project Files*, and ready for import into the *synonyms* folder.
- a) Right-click the *synonyms* folder and choose *Import → File or Project*.
  - b) Click *Browse*, select the file *HA300 → New Project Files → ExternalSchemaAccess.hdbgrants*, and choose *Open*.
  - c) Check *Import to Location* and choose *OK*.
8. Build the entire *synonyms* folder and check the build status.
- a) Right-click the *synonyms* folder and choose *Build Selected Files*.
  - b) In the Console, make sure the build was successful.

### 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 *SALES\_DATA* 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           | <b>SALES</b>   |
| Label          | <b>Sales</b>   |
| Data Category  | <i>CUBE</i>    |
| With Star Join | [Not selected] |

- a) In the *Workspace* tree, right-click the *db* → *src* → *models* folder and choose *New* → *Calculation View*.
  - b) Enter the Calculation View name and other properties as specified in the table.
  - c) Choose *Create*.
2. Add the table *SALES\_DATA* table to the *Aggregation* node.
- 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 **SALES**.
  - c) In the search results, select the *SALES\_DATA* 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 *SALES\_DATA* Data Source to the *Output Columns* area.
4. Save, and then build the new calculation view.
- a) Choose *Save*.
  - b) Choose *Build* → *Build Selected Files*.
  - c) In the *Console* pane, check that the build was completed successfully.
5. Preview the data of the *SALES* calculation view.
- a) In the workspace, right-click the *SALES* calculation view and choose *Data Preview*.
  - b) Check that the raw data tab contains data.
6. Close all open tabs in the SAP Web IDE.
- a) Right-click any open tab and choose *Close All*.

## Unit 7

### 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 the SAP Web IDE, and would like to understand how the native Git integration within the SAP Web IDE for SAP HANA can help you better support your development cycle. You must first finalize version 2.1.0, then you will implement changes that will be released in upcoming version 2.2.0.



#### Note:

Git is a tool that supports collaboration on a project. 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)

#### 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 WebIDE project will be managed as a local Git repository from now on. No content is moved or duplicated.

1. If needed, launch the SAP HANA SAP Web IDE. A shortcut to this application is available in the following folder: *HA300 → URLs → SAP Web IDE for SAP HANA*.
2. In the workspace, import the project archive *MyGitProject.zip* from your course files folder *HA300 → Git Project Files*. Give the imported project the name **MyGitProject\_##** (## is your group 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.

3. Enable the Git functionality in your new project *MyGitProject\_##*.  
You need to confirm the proposed email and user name, which are the ones assigned to your SAP HANA user.
4. Observe the changes in the way the project now displays in the workspace.

What do you notice?

---



---



---

5. Open the *Git* pane.

What is the status icon of all the files of the project, and the folders that contain them? Are they currently staged?

---



---



---

6. Open the *Git History* pane.

Is the history of your project totally empty?

---



---



---

7. Make sure all your project content is staged and commit the existing content of your project with description **Import v2.1.0 project (in progress)**.

### **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. Display the commit history of the entire *models* folder.

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 the SAP Web IDE.

### **Task 3: Start the Development of Next Release**

The version 2.1.0 or 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 *master* 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 Web IDE.
5. In the *Git Pane*, select the *master* branch.

What do you observe?

---



---



---

6. If the `project.json` file appears as modified, stage it and commit this change into branch `master` with description `Commit project.json`.
7. Open `File2` and switch back to branch `V220`.

What do you notice?

---



---



---

8. Again, if the `project.json` file appears as modified, stage it and commit this change into branch `V220` with description `Commit project.json`.

9. 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 `master`.

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 the SAP Web IDE.
7. The hotfix has been tested and released. Merge the `V211` branch into the `master` branch.



Hint:

You must check out the `master` branch first.

8. 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 `master` branch.

# 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 the SAP Web IDE, and would like to understand how the native Git integration within the SAP Web IDE for SAP HANA can help you better support your development cycle. You must first finalize version 2.1.0, then you will implement changes that will be released in upcoming version 2.2.0.



### Note:

Git is a tool that supports collaboration on a project. 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)

### 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 WebIDE project will be managed as a local Git repository from now on. No content is moved or duplicated.

1. If needed, launch the SAP HANA SAP Web IDE. A shortcut to this application is available in the following folder: HA300 → URLs → SAP Web IDE for SAP HANA.
  - a) Start Windows Explorer and navigate to the folder HA300 → URLs.
  - b) Double-click the shortcut SAP Web IDE for SAP HANA.
  - c) Log on with your SAP HANA credentials: **STUDENT## / Training1** (## is your group number).

2. In the workspace, import the project archive *MyGitProject.zip* from your course files folder *HA300 → Git Project Files*. Give the imported project the name **MyGitProject\_##** (## is your group 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 *Development* perspective, right-click the *Workspace* folder and choose *Import → File or Project*.
  - b) Choose *Browse* and navigate to the Windows folder *Quick Access → HA300 → Git Project Files*.
  - c) Select the file *MyGitProject.zip* and choose *Open*.
  - d) In the *Import to* field, enter **/MyGitProject\_##** (## is your group number).
  - e) Choose *OK* to begin the import.
3. Enable the Git functionality in your new project *MyGitProject\_##*.  
You need to confirm the proposed email and user name, which are the ones assigned to your SAP HANA user.
    - a) In your workspace, right-click the *MyGitProject\_##* and choose *Git → Initialize Local Repository*.
    - b) In the *Git User Information* dialog box, choose *Save*.
  4. 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?

Icons are used to indicate the files and folders status in Git, and the project name is followed by (*master*), which is the name of the main branch of the project and, for now, the unique one.

5. Open the *Git* pane.
  - a) On the toolbar at the right of the screen, choose the *Git Pane* icon.
  - b) In the *Changes* area, observe the *Status* icon and the *Stage* checkbox.

What is the status icon of all the files of the project, and the folders that contain them? Are they currently staged?

These files and folders have a + status icon, which means that they are new, and staged. Note that when enabling an existing project for Git tracking with *Initialize Local Repository*, all the existing files are automatically staged.

6. Open the *Git History* pane.
  - a) On the toolbar at the right of the screen, choose the *Git History* icon.

Is the history of your project totally empty?

No, it contains an initial commit which corresponds to the initialization of the empty project. This commit, in particular, does not include any of the imported files, even if they were there BEFORE the activation of Git tracking.

---

7. Make sure all your project content is staged and commit the existing content of your project with description **Import v2.1.0 project (in progress)**.
  - a) To display the Git pane, choose the *Git Pane* icon.
  - b) Check that all the listed files are staged. If not, choose *Stage All*.
  - c) In the *Commit Message* field, enter **Import v2.1.0 project (in progress)**.
  - d) Choose *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**
  - a) In your project *MyGitProject\_##*, expand the folder *dbmodule* → *src* → *models*.
  - b) Double-click *File1*.
  - c) Add the text provided in the step.
  - d) Choose *Save*.
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 choose *OK*.
  - c) Add the text provided in the step.
  - d) 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 *Git* pane, make sure that both *File1* and *File3* are listed and choose *Stage All*.
- b) In the *Commit Description* field, add the text provided in the step..

- c) Choose *Commit*.
  - d) On the toolbar at the right of the screen, choose the *Git History* icon.
  - e) Check that your last commit is listed.  
If *File3* is selected, you only see one commit because this file was affected by one commit only.
4. Display the commit history of the entire *models* folder.
- a) In the workspace, select the *models* folder.
  - b) Check that the entire commit history for all your files is listed.
  - c) 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 and *File3* was added. Note that, in addition, the *project.json* file was modified ; it is an important technical file that governs, for example, the list of modules in an MTA project.

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, double-click *File3*.
  - b) Add the text provided in the step.
  - c) 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 *Git Pane*, select the *Stage* checkbox for *File3*.
  - b) Select the *Amend* checkbox.
  - c) Keep the previous description that automatically displays.
  - d) Choose *Commit*.
7. Close all the open tabs in the SAP Web IDE.
- a) Right-click any open tab and choose *Close All*.

### Task 3: Start the Development of Next Release

The version 2.1.0 or 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 *master* branch.
  - a) In the *Git Pane*, choose **+** (*Create Local Branch*).
  - b) In the dialog box, in the *Branch Name* field, enter **v220**.
  - c) Choose **OK**.  
The new branch is created and automatically selected (which, in Git terminology, is known as “checked out”).
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) 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) Click **Save**.
  - e) Double-click *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) Choose **Stage All**.
  - 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 Web IDE.
  - a) Right-click any open tab and choose **Close All**.
5. In the *Git Pane*, select the *master* branch.
  - a) In the *Branch* dropdown list, select *master*.

What do you observe?

The content of the workspace changes (*File4* disappears because it does not exist in the *master* branch.)

---

6. If the *project.json* file appears as modified, stage it and commit this change into branch *master* with description **Commit project.json**.
  - a) Choose **Stage All**.
  - b) In the *Commit Description* field, add the text provided in the step.
  - c) Choose **Commit**.
7. Open *File2* and switch back to branch *V220*.

- a) In your *models* folder, double-click *File2*.
- b) In the *Branch* dropdown list, select V220.

What do you notice?

The content of *File2* is changed according to its content in each branch. This means that your workspace content is adapted automatically to reflect the status of the branch you have selected/checked out.

8. Again, if the *project.json* file appears as modified, stage it and commit this change into branch V220 with description **Commit project.json**.
  - a) Choose *Stage All*.
  - b) In the *Commit Description* field, add the text provided in the step.
  - c) Choose *Commit*.
9. 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 *master*. Branching from the current branch V220 would not suit your requirement.

2. Create the new branch **v211** from *master*.
  - a) In the *Branch* dropdown list, select *master*.



Note:

Selecting the source branch from which you want to create a new one is not mandatory (because you can also choose it from within the dialog box). The small benefit is that the current branch (*master*) will be proposed by default as the source branch to create a new one.

- b) Choose **+** (*Create Local Branch*).
- c) In the dialog box, in the *Branch Name* field, enter **v211**.
- 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 *models* folder, 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.
- In the workspace tree of *MyGitProject\_##*, double-click the *mta.yaml* file.  
The file opens in the MTA Editor.
  - In the *Basic Information* tab, update the *Application Version* to **2.1.1**.
  - Choose Save.
  - 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**
- Choose *Stage All*.
  - In the *Commit Description* field, add the text provided in the step.
  - Check that the selected (checked out) branch is *V211*.
  - Choose *Commit*.
6. Close all the open tabs in the SAP Web IDE.
- Right-click any open tab and choose *Close All*.
7. The hotfix has been tested and released. Merge the *V211* branch into the *master* branch.



Hint:

You must check out the *master* branch first.

- In the *Branch* dropdown list, select *master*.
  - Choose *Merge*.
  - In the dialog box, select the *V211* branch and choose *OK*.
8. 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 *master* branch.

- In the *Branch* dropdown list, check that the *V211* is NOT selected.



Note:

This is mandatory because you cannot delete a branch that is currently checked out.

- Click the delete branch.
- In the *Delete Branch* dialog box, select the *V211* branch and choose *Delete*.

## Unit 7

### Exercise 41

# Clone the SAP HANA Interactive Education (SHINE) Application from GitHub

#### Exercise Objectives

After completing this exercise, you will be able to:

- Use the native Git connectivity of the Web IDE for SAP HANA to replicate a project from GitHub.

#### Business Example

You are working as a modeler on an SAP HANA Project, and would like to benefit from the SHINE model, in particular its HDB modules, to help you during the design phase of the project. In order to access easily all the design-time artifacts of SHINE for XSA, you have decided to clone the corresponding repository from GitHub.



##### Note:

You will NOT build the SHINE application in your system. This application is already deployed in your system and shared between the instructor and all the participants.

1. Check from the GitHub repository which version of the SHINE Model suits your training system, which runs SAP HANA 2.0 SPS05.

The repository URL is: <https://github.com/SAP/hana-shine-xsa>. The *master* branch always contains the last released version of SHINE, and the previous versions are kept in different branches.

2. Clone the Git Repository of SHINE for XSA, which is available at the following URL:  
<https://github.com/SAP/hana-shine-xsa>



##### Note:

You will be asked to adjust the settings for Git system files that should be ignored (not tracked). Ignore this step by choosing *Do It Later*. Indeed, the GitHub repository for SHINE content is read-only and you are not allowed to push any modification to this repository.

3. (Optional) In case the branch *master* of the GitHub repository contains a more recent version of SHINE for XSA –refer to Step 1 above– but you still want to consult specifically the SHINE for XSA that suits SAP HANA 2.0 SPS05, create an additional local branch from the remote branch *HANA2.0-SPS05* and check out (select) this branch.



Note:

A *Checkout failed* dialog box might show up, asking you whether you want to reset the local master branch which contains a few uncommitted changes (mainly *project.json* files). These are minor changes that you can discard, so choose *Reset and Checkout*.

4. You are now ready to explore the content of the cloned repository, in particular the *core-db* module, which is the main HDB module of SHINE, and the most relevant content for the application modeler.

In this *core-db* module, the calculation views are located in the *src → models* folder, and the persistence layer in the *src → data*.

5. Close all open tabs in the SAP Web IDE.

## Unit 7 Solution 41

# Clone the SAP HANA Interactive Education (SHINE) Application from GitHub

### Exercise Objectives

After completing this exercise, you will be able to:

- Use the native Git connectivity of the Web IDE for SAP HANA to replicate a project from GitHub.

### Business Example

You are working as a modeler on an SAP HANA Project, and would like to benefit from the SHINE model, in particular its HDB modules, to help you during the design phase of the project. In order to access easily all the design-time artifacts of SHINE for XSA, you have decided to clone the corresponding repository from GitHub.



#### Note:

You will NOT build the SHINE application in your system. This application is already deployed in your system and shared between the instructor and all the participants.

1. Check from the GitHub repository which version of the SHINE Model suits your training system, which runs SAP HANA 2.0 SPS05.

The repository URL is: <https://github.com/SAP/hana-shine-xsa>. The *master* branch always contains the last released version of SHINE, and the previous versions are kept in different branches.

- a) In Google Chrome, open a new tab and enter the SHINE for XSA URL.  
The *master* branch displays.
- b) Check the release information displayed for the different folders. This indicates for which version of SAP HANA the SHINE for XSA code is valid.
- c) In case a more recent version of SHINE for XSA has been released, for example, for HANA 2.0 SPS06, check the name of the branch that contains the version of SHINE for SAP HANA 2.0 SPS05. It should be *HANA2.0-SPS05*.

2. Clone the Git Repository of SHINE for XSA, which is available at the following URL:

<https://github.com/SAP/hana-shine-xsa>



#### Note:

You will be asked to adjust the settings for Git system files that should be ignored (not tracked). Ignore this step by choosing *Do It Later*. Indeed, the GitHub repository for SHINE content is read-only and you are not allowed to push any modification to this repository.

- a) In the SAP Web IDE for SAP HANA, right-click the workspace and choose *Git → Clone Repository*.
  - b) Enter the provided GitHub URL and choose *Clone*.  
After a while, a new folder `** hana-shine-xsa (master)` is added to your workspace, and populated with all the SHINE design-time files.
  - c) In the *Git Ignore System Files* dialog box, choose *Do It Later*.
3. (Optional) In case the branch *master* of the GitHub repository contains a more recent version of SHINE for XSA –refer to Step 1 above– but you still want to consult specifically the SHINE for XSA that suits SAP HANA 2.0 SPS05, create an additional local branch from the remote branch *HANA2.0-SPS05* and check out (select) this branch.

**Note:**

A *Checkout failed* dialog box might show up, asking you whether you want to reset the local master branch which contains a few uncommitted changes (mainly `project.json` files). These are minor changes that you can discard, so choose *Reset and Checkout*.

- a) If needed, open the *Git Pane* from the toolbar on the right.
  - b) Next to the *Branch* dropdown list, choose **+** (*Create Local Branch*).
  - c) In the dialog box, select the *Source Branch* (remote branch) *origin/HANA2.0-SPS05*.
  - d) Keep the proposed *Branch Name*.
  - e) In the *Checkout failed* window (if any), choose *Reset and Checkout* and confirm by choosing *OK*.  
The local branch *HANA2.0-SPS05* is created and checked out.
4. You are now ready to explore the content of the cloned repository, in particular the *core-db* module, which is the main HDB module of SHINE, and the most relevant content for the application modeler.  
In this *core-db* module, the calculation views are located in the *src → models* folder, and the persistence layer in the *src → data*.
5. Close all open tabs in the SAP Web IDE.
- a) Right-click any open tab and choose *Close All*.

## Unit 8

### Exercise 42

# 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
- Define a role based on an analytic privilege
- Assign a role to a user

#### Business Example

You have finished creating a number of information models, and you need to secure the data access for these models before they are moved to the QA environment for testing.

#### Task 1: Import a View

You will first import a Calculation View and check that you can display its data.

1. In the SAP Web IDE, import the *CVCS\_PO2.hdbcalculationview* file from your course files folder *HA300 → CVCS\_PO2.hdbcalculationview* into your workspace folder *exercises*.
2. Build the *CVCS\_PO2* view.
3. Preview the calculation view data.

It should retrieve data for several countries.

#### Task 2: Preview the Data in Microsoft Excel

From now on, and during the remainder of the exercise, you will use an external tool to check the behavior of the view while you're defining data access security for this view.



#### Note:

Indeed, due to the containerization, the data preview in the Web IDE is executed by a technical user, not your connected user *STUDENT##*. An alternative could be to execute a SQL query on top of the Calculation View in the Database Explorer, but with the connection to the *H00 DB* (not to the container).

1. If needed, start Microsoft Excel.
2. To add the *CVCS\_PO2* data source, use the *Data → Existing Connections* menu of Excel and first define the connection to SAP HANA.  
Use the following log-on info:

Table 9: SAP HANA Log-on Information

| Field           | Value                           |
|-----------------|---------------------------------|
| Host            | <b>wdflbmt7215.wdf.sap.corp</b> |
| Instance number | <b>00</b>                       |
| Database Mode   | <i>Multi Database</i>           |
| Database        | <i>User Database: HA00</i>      |
| User            | <b>STUDENT##</b>                |
| Password        | <b>Training1</b>                |
| Language        | <b>EN</b>                       |

3. Choose the data source *CVCS\_PO2*, located in the “database” *HA300\_##\_HDI\_HDB\_1.HA300*.

**Note:**

This is not actually a database, but more specifically the *HA300* namespace that identifies DB objects in your container schema *HA300\_##\_HDI\_HDB\_1*.

The Pivot Table tools open in the worksheet.

4. Define your pivot table to display the gross amount analyzed by region and country.  
These two characteristics of the Business Partners dimension are structured into a hierarchy called Geographical Hierarchy.
5. Leave the Excel workbook open. You will refresh it while setting up your analytic privileges.

**Task 3: Create an Analytic Privilege**

You want to limit the visibility of data for this view to the purchase orders from customers located in the US.

1. In the SAP Web IDE for SAP HANA, create a new analytic privilege in the models folder, with the following properties:

Table 10: Analytic Privilege Properties

| Field | Value                |
|-------|----------------------|
| Name  | <b>AP_COUNTRY_US</b> |
| Label | <b>US Partners</b>   |

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 then build 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. Open the calculation view CVCS\_PO2 and modify it 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 then build the CVCS\_PO2 calculation view.

Check the build status.

3. Build again the AP\_COUNTRY\_US analytic privilege.

#### Task 5: Preview the Calculation View in Excel

You will now refresh the Excel workbook to see if your analytic privilege definition has been applied.

1. Back in Excel, refresh your Excel workbook.

What do you observe?

---



---



---

2. Can you explain why the view does not show any data at all?

---



---



---

#### Task 6: 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.
2. Add the Analytic Privilege HA300::AP\_COUNTRY\_US to the role definition.
3. Save, and then build the DATA\_VIEWER role.
4. Build the DATA\_VIEWER role and check the build status.

### Task 7: Assign the New Role to your User

Now you are ready to grant the runtime role *DATA\_VIEWER*, containing your analytic privilege, to the role *TRAINING\_ROLE\_##* that is already assigned to your user *STUDENT##*.

1. In the Database Explorer, open a SQL console connected to the *H00 DB* (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:  
*HA300 → Prepared Code → SQL to Assign Data Viewer Role.sql*. Then replace *##* with your group number.
3. Execute the first statement to assign the *DATA\_VIEWER* role to your user.  
GRANT "HA300\_##\_HDI\_HDB\_1"."HA300::DATA\_VIEWER" TO TRAINING\_ROLE##;
4. Leave the SQL console open, as you will use the rest of the other SQL statements from the imported file later on.

### Task 8: Refresh the Excel Workbook

You can now test whether the data is available (and restricted) in the MS Excel workbook.

1. In MS Excel, refresh the workbook.  
You can now see some of the *CVCS\_PO2* data, restricted to the Country US.

### Task 9: 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. Back to the Web IDE, open the following Analytic Privileges located in folder *hdb → 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 folder *hdb → 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*.

```
GRANT "HA300_##_HDI_HDB_1"."HA300::DATA_VIEWER2" TO TRAINING_ROLE_#;
SELECT DISTINCT COUNTRY, CATEGORY FROM
"HA300_##_HDI_HDB_1"."HA300::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 #3. 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.

```
REVOKE "HA300_##_HDI_HDB_1"."HA300::DATA_VIEWER2" FROM
TRAINING_ROLE_#;
GRANT "HA300_##_HDI_HDB_1"."HA300::DATA_VIEWER3" TO TRAINING_ROLE_#;
SELECT DISTINCT COUNTRY, CATEGORY FROM
"HA300_##_HDI_HDB_1"."HA300::CVCS_PO3_AP";
```

What do you observe this time? How can you explain it?

---



---



---

5. Close all open tabs in the SAP Web IDE.

# 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
- Define a role based on an analytic privilege
- Assign a role to a user

## Business Example

You have finished creating a number of information models, and you need to secure the data access for these models before they are moved to the QA environment for testing.

### Task 1: Import a View

You will first import a Calculation View and check that you can display its data.

1. In the SAP Web IDE, import the *CVCS\_PO2.hdbcalculationview* file from your course files folder *HA300 → CVCS\_PO2.hdbcalculationview* into your workspace folder *exercises*.
  - a) In the workspace, right-click your *exercises* folder and choose *import → From File System*.
  - b) In the *Import* window, choose *Browse*.
  - c) Select the *HA300 → Calculation Views Templates → CVCS\_PO2.hdbcalculationview* file and choose *Open*.
  - d) Make sure the *Import to* location ends with *.../exercises*.
  - e) Choose *OK*.  
The new Calculation View *CVCS\_PO2* is created in your project.
2. Build the *CVCS\_PO2* view.
  - a) Right-click the new (imported) view and choose *Build → Selected Files*.
  - b) Check that the build was successful.
3. Preview the calculation view data.  
It should retrieve data for several countries.
  - a) Right-click the new (imported) view and choose *Data Preview*.
  - b) Check that the view retrieves data.

### Task 2: Preview the Data in Microsoft Excel

From now on, and during the remainder of the exercise, you will use an external tool to check the behavior of the view while you're defining data access security for this view.



## Note:

Indeed, due to the containerization, the data preview in the Web IDE is executed by a technical user, not your connected user **STUDENT##**. An alternative could be to execute a SQL query on top of the Calculation View in the Database Explorer, but with the connection to the **H00 DB** (not to the container).

1. If needed, start Microsoft Excel.
- a) In the Windows Start menu, enter **Excel** and in the search results choose *Microsoft Excel*.
2. To add the CVCS\_PO2 data source, use the *Data → Existing Connections* menu of Excel and first define the connection to SAP HANA.

Use the following log-on info:

Table 9: SAP HANA Log-on Information

| Field           | Value                           |
|-----------------|---------------------------------|
| Host            | <b>wdf1bmt7215.wdf.sap.corp</b> |
| Instance number | <b>00</b>                       |
| Database Mode   | <i>Multi Database</i>           |
| Database        | <i>User Database: H00</i>       |
| User            | <b>STUDENT##</b>                |
| Password        | <b>Training1</b>                |
| Language        | EN                              |

- a) Choose *Data → Existing Connections*.
- b) Choose *Browse for More*.
- c) Choose *+Connect to New Data Source.odc* and choose *Open*.
- d) Select *Other/Advanced* and choose *Next*.
- e) Select *SAP HANA MDX Provider* and choose *Next*.
- f) Enter the connection details as per the table.
- g) Choose *OK*.
3. Choose the data source **CVCS\_PO2**, located in the “database” **HA300\_##\_HDI\_HDB\_1.HA300**.



## Note:

This is not actually a database, but more specifically the **HA300** namespace that identifies DB objects in your container schema **HA300\_##\_HDI\_HDB\_1**.

a) At the *Select Database and Table* step, in the drop-down list, select the entry *HA300\_##\_HDI\_HDB\_1.HA300*.

b) Select the *Connect to a specific cube* checkbox.

c) Select the *CVCS\_PO2* cube and choose *Next*.

d) Note that the definition of the connection will be saved in a .odc file.

This file can be used to access this view directly, if needed, without providing all the connection details apart from the user password.

e) Choose *Finish*.

f) In the *Import Data* window, keep the default settings and choose *OK*.

The Pivot Table tools open in the worksheet.

4. Define your pivot table to display the gross amount analyzed by region and country.

These two characteristics of the Business Partners dimension are structured into a hierarchy called Geographical Hierarchy.

a) In the *PivotTable Fields* pane, select the *GROSS\_AMOUNT* value and the *Geographical Hierarchy*.

The gross amount by region display in the table. You can drill down to the country level.

5. Leave the Excel workbook open. You will refresh it while setting up your analytic privileges.

### Task 3: Create an Analytic Privilege

You want to limit the visibility of data for this view to the purchase orders from customers located in the US.

1. In the SAP Web IDE for SAP HANA, create a new analytic privilege in the models folder, with the following properties:

Table 10: Analytic Privilege Properties

| Field | Value                |
|-------|----------------------|
| Name  | <b>AP_COUNTRY_US</b> |
| Label | <b>US Partners</b>   |

a) In the SAP Web IDE, right-click your exercises folder and choose *New → Analytic Privilege*.

b) Enter the fields as per the table and choose *Create*.

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 *HA300::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 *HA300::CVD\_BP2* and choose *OK*.
  - c) In the *Restriction column*, click the *+* icon.
  - d) In the *Operator* field, choose *=*.
  - e) In the *Value*, select *US* and choose *OK*.
4. Save, and then build the new analytic privilege.  
Check the build status.
- a) Choose *Save*.
  - b) Choose *Build* → *Build Selected Files*.  
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. Open the calculation view *CVCS\_PO2* and modify it so that it triggers an analytic privilege check when it is queried.



Hint:

This is done in the *View Properties* tab of Semantics.

- a) In your workspace, 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 then build the *CVCS\_PO2* calculation view.  
Check the build status.
- a) Choose *Save*.
  - b) Choose *Build* → *Build Selected Files*.
3. Build again the *AP\_COUNTRY\_US* analytic privilege.
- a) Right-click the *AP\_COUNTRY\_US* analytic privilege and choose *Build* → *Build Selected Files*.
  - b) Check the build status.  
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 in Excel

You will now refresh the Excel workbook to see if your analytic privilege definition has been applied.

1. Back in Excel, refresh your Excel workbook.

What do you observe?

The data is not available any longer. The user is not authorized.

---

- a) In the Excel workbook, choose *Data → Refresh All*.

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.

---

### Task 6: 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.

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

- b) In the *Role name* field, enter **DATA\_VIEWER**.

- c) Choose the *Clear config input* icon.

- d) Choose *Create*.

The new role *DATA\_VIEWER* is created and it opens in the *Role editor* window.

2. Add the Analytic Privilege *HA300::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 *HA300::AP\_COUNTRY\_US* Analytic Privilege.

- d) Remove the name of your container schema from the corresponding *Schema Reference* column.

3. Save, and then build the *DATA\_VIEWER* role.

- a) Choose *Save*.

- b) Right-click the *exercises* → *DATA\_VIEWER* role and choose *Build → Build Selected Files*.

- c) In the *Console*, check that the build was successful.

4. Build the *DATA\_VIEWER* role and check the build status.

- a) Right-click the exercises → *DATA\_VIEWER* role and choose *Build* → *Build Selected Files*.
- b) In the Console, check that the build was successful.

### Task 7: Assign the New Role to your User

Now you are ready to grant the runtime role *DATA\_VIEWER*, containing your analytic privilege, to the role *TRAINING\_ROLE\_##* that is already assigned to your user *STUDENT##*.

1. In the Database Explorer, open a SQL console connected to the *H00 DB* (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) Choose *Tools* → *Database Explorer*
  - b) In the Database/Containers list on the left, select the *H00 DB* entry.
  - c) Click the *Open SQL Console* icon.
2. In the SQL console, import the following file containing prepared SQL statements:  
*HA300* → *Prepared Code* → *SQL to Assign Data Viewer Role.sql*. Then replace *##* with your group number.
  - a) Choose *Import File*.
  - b) Select the file specified above and choose *Open*.
  - c) To replace *##* with your group number everywhere in the SQL statements, press **ctrl +H**.
3. Execute the first statement to assign the *DATA\_VIEWER* role to your user.  

```
GRANT "HA300_##_HDI_HDB_1"."HA300::DATA_VIEWER" TO TRAINING_ROLE_##;
```

  - a) Highlight the SQL statement in section #1 and choose *Run (F8)*.
  - b) Check that the grant statement was successful.
4. Leave the SQL console open, as you will use the rest of the other SQL statements from the imported file later on.

### Task 8: Refresh the Excel Workbook

You can now test whether the data is available (and restricted) in the MS Excel workbook.

1. In MS Excel, refresh the workbook.
  - a) Choose *Data* → *Refresh All*.
  - b) Check the displayed data in the pivot table.

You can now see some of the *CVCS\_PO2* data, restricted to the Country US.

### Task 9: 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.

- Back to the Web IDE, open the following Analytic Privileges located in folder *hdb → src → resources* and check their definition:

- AP\_COUNTRY\_DE
  - AP\_COUNTRY\_DE\_PROD\_NOTEBOOK
  - AP\_PROD\_NOTEBOOKS
- a) In the Web IDE, choose *Tools → Development*.
  - b) In the workspace, locate each file and double-click it.
  - c) 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 HA300::CVCS\_PO3\_AP.

- Open the following roles located in folder *hdb → 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.

- 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*.

```
GRANT "HA300_##_HDI_HDB_1"."HA300::DATA_VIEWER2" TO TRAINING_ROLE_##;
SELECT DISTINCT COUNTRY, CATEGORY FROM
"HA300_##_HDI_HDB_1"."HA300::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 #2 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 #3. 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.

```
REVOKE "HA300_##_HDI_HDB_1"."HA300::DATA_VIEWER2" FROM
TRAINING_ROLE_##;
GRANT "HA300_##_HDI_HDB_1"."HA300::DATA_VIEWER3" TO TRAINING_ROLE_##;
SELECT DISTINCT COUNTRY, CATEGORY FROM
"HA300_##_HDI_HDB_1"."HA300::CVCS_PO3_AP";
```

- a) Highlight the statements in section #3 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 both the restrictions on country and product category.

5. Close all open tabs in the SAP Web IDE.
- a) Right-click any open tab and choose *Close All*.

## Unit 8

### 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

### Task 1: Import a Calculation View and Preview its Data

1. If needed, launch the SAP HANA Web IDE. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
  2. Import the calculation view *CVD\_EMPLOYEES.hdbculationview*.  
It is located in the folder HA300 → Calculation Views Templates.
  3. Build the imported calculation view.  
Check the build status.
  4. Preview the data.
  5. 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. Save, and then build the calculation view.

Check the build 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 *H00 DB*, which is defined with your user *STUDENT##*. You will create a role for this, and then assign it to *STUDENT##* via a dedicated role *TRAINING\_ROLE\_##*.

1. Switch to the *Database Explorer* perspective and open an SQL console connected to your *H00 DB* connection.



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:

*HA300 → Prepared Code → SQL for Data Masking.sql*. Then replace **##** everywhere with your group number.

---

3. Execute query #1 to display the calculation view data.

```
SELECT * FROM "HA300_##_HDI_HDB_1"."HA300::CVD_EMPLOYEES_WITH_MASK";
```

What do you notice?

---

---

---

4. Back to the *Development* perspective, in your exercises folder, import the role design-time file *UNMASK\_ALL.hdbrole* available in your course files folder *HA300*.

5. Review the role definition.

Which type of privilege does this role grant? What does it mean?

---

---

---

What specific authorization is granted?

---

---

---

6. Build the *UNMASK\_ALL* role.

Check the build status.

7. Grant the *HA300::UNMASK\_ALL* role to the role *TRAINING\_ROLE\_##*.

This can be done by executing query #2 inside your SQL console.

```
GRANT "HA300_##_HDI_HDB_1"."HA300::UNMASK_ALL" TO TRAINING_ROLE_##;
```

8. Execute again query #1 to display the calculation view data.

```
SELECT * FROM "HA300_##_HDI_HDB_1"."HA300::CVD_EMPLOYEES_WITH_MASK";
```

What do you notice?

---

---

---

9. Close all open tabs in the SAP Web IDE.

# 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

### Task 1: Import a Calculation View and Preview its Data

1. If needed, launch the SAP HANA Web IDE. A shortcut to this application is available in the following folder: HA300 → URLs → Web IDE for SAP HANA.
  - a) Start Windows Explorer and navigate to the folder HA300 → URLs.
  - b) Double-click the shortcut *Web IDE for SAP HANA*.
  - c) Log on with your SAP HANA credentials: **STUDENT##** and **Training1** (where ## is your group number).
2. Import the calculation view *CVD\_EMPLOYEES.hdbculationview*.  
It is located in the folder HA300 → Calculation Views Templates.
  - a) Right-click the exercises folder and choose *Import → File or Project*.
  - b) Click *Browse*, select the file HA300 → Calculation Views Templates → *CVD\_EMPLOYEES.hdbculationview*, and choose *OK*.
  - c) Check the *Import to* location and choose *OK*.  
The Calculation View opens in the editor.
3. Build the imported calculation view.  
Check the build status.

- a) In the exercises folder, right-click the calculation view *CVD\_EMPLOYEES* and choose *Build Selected Files*.
  - b) In the *Console* pane, check that the build was completed successfully.
4. Preview the data.
- a) Right-click the *CVD\_EMPLOYEES* view and choose *Data Preview*.
  - b) Select the *Raw Data* tab.
5. 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 figures. 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 choose *Rename*.
- c) In the *Confirmation* dialog box, choose *Yes*.
- d) In the *Refactor Views* dialog box, choose *Refactor*.
- e) Choose *Finish*.

2. Define the mask expression as follows:

```
'XXXXXXXXXX' || RIGHT("BankAccountNumber", 5)
```

- a) In the *Workspace*, double-click 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. Save, and then build the calculation view.
- Check the build status.
- a) Choose *Save*.
  - b) Choose *Build* → *Build Selected Files*.
  - c) In the *Console* pane, check that the build was completed successfully.
4. Preview the data
- a) Choose *Edit* → *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 *H00 DB*, which is defined with your user *STUDENT##*. You will create a role for this, and then assign it to *STUDENT##* via a dedicated role *TRAINING\_ROLE\_##*.

1. Switch to the *Database Explorer* perspective and open an SQL console connected to your *H00 DB* connection.

**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) Choose *Tools* → *Database Explorer*.
- Alternatively, you can click the corresponding icon  on the left toolbar.
- b) Select the entry *H00 DB* (classic database) and choose  *Open SQL Console*.
2. In the SQL console, import the following file containing prepared SQL statements: *HA300 → Prepared Code → SQL for Data Masking.sql*. Then replace *##* everywhere with your group number.

- a) Choose *Import File*.
- b) Select the file specified above and choose *Open*.
- c) To replace ## with your group number everywhere in the SQL statements, press **ctrl +H**.

3. Execute query #1 to display the calculation view data.

```
SELECT * FROM "HA300_##_HDI_HDB_1"."HA300::CVD_EMPLOYEES_WITH_MASK";
```

- 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?

The user STUDENT## cannot see the actual data for bank accounts. The data mask is applied.

---

4. Back to the *Development* perspective, in your *exercises* folder, import the role design-time file *UNMASK\_ALL.hdbrole* available in your course files folder *HA300*.

- a) To switch perspective, choose *Tools → Development*.  
Alternatively, you can click the corresponding icon  on the left toolbar.
- b) Right-click the *exercises* folder and choose *Import → File or Project*.
- c) Click *Browse*, select the file *HA300 → UNMASK\_ALL.hdbrole* and choose *OK*.
- d) Check the *Import to* location and choose *OK*.  
The imported role opens in the role editor.

5. Review the role definition.

- a) Review the different tabs in the role definition to find which privileges are granted to (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. Build the *UNMASK\_ALL* role.

Check the build status.

- a) In the *exercises* folder, right-click the role *UNMASK\_ALL* and choose *Build Selected Files*.
- b) In the *Console* pane, check that the build was completed successfully.

7. Grant the HA300::UNMASK\_ALL role to the role TRAINING\_ROLE\_##.

This can be done by executing query #2 inside your SQL console.

```
GRANT "HA300_##_HDI_HDB_1"."HA300::UNMASK_ALL" TO TRAINING_ROLE_##;
```

- a) Select the tab SQL Console ...
- b) Make sure ## has been replaced everywhere with your group number.
- c) Highlight the specified statement and choose Run (F8).
- d) Review the output in the Message tab.

8. Execute again query #1 to display the calculation view data.

```
SELECT * FROM "HA300_##_HDI_HDB_1"."HA300::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, the user STUDENT## is allowed to unmask the masked columns and see the actual data for bank accounts.

9. Close all open tabs in the SAP Web IDE.

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