

ADVANCED DATA MODELING IN SAP HANA

HBD362

Exercises / Solutions

Bangalore	Rutika Bodas	SAP Labs India
Bangalore	Rajesh Panditi	SAP Labs India
Bangalore	Karthick TSB	SAP Labs India
Barcelona	Yves Augustin	SAP SE
Barcelona	Jan Zwickel	SAP SE
Las Vegas	Nancy Chen	SAP Labs US
Las Vegas	Werner Steyn	SAP Labs US

TABLE OF CONTENTS

SETUP - (10 MINUTES, 20 STEPS)3

EXERCISE 1 - MULTI LEVEL AGGREGATION (15 MINUTES, 30 STEPS)..... 11

EXERCISE 2 - SCRIPTED MODELS VS MODELED MODELS (20 MINUTES, 35 STEPS)21

EXERCISE 3 - DYNAMIC RANKING (10 MINUTES, 14 STEPS).....31

EXERCISE 4 - EXPLICIT VS IMPLICIT UNION PRUNING (20 MINUTES, 34 STEPS).....37

EXERCISE 5 - NON-CUMULATIVE KEY FIGURES (15 MINUTES, 28 STEPS)47

EXERCISE 6 - CUMULATIVE SLOWLY CHANGING DIMENSIONS (20 MINUTES, 30 STEPS).....56

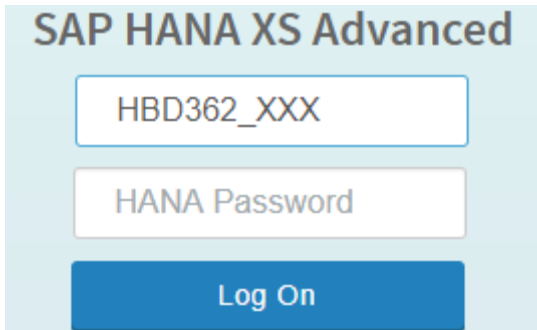
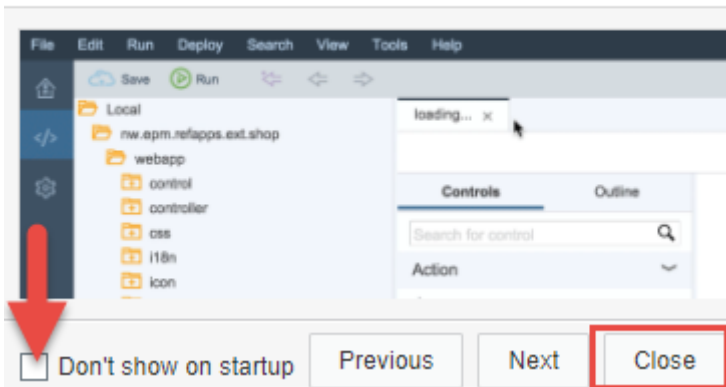
EXERCISE 7 - SQL HIERARCHIES AND SQL ANALYTICAL PRIVILEGES (20 MINUTES, 30 STEPS)69

SETUP - (10 Minutes, 20 steps)

During the exercises, you will be using the SAP Web IDE:

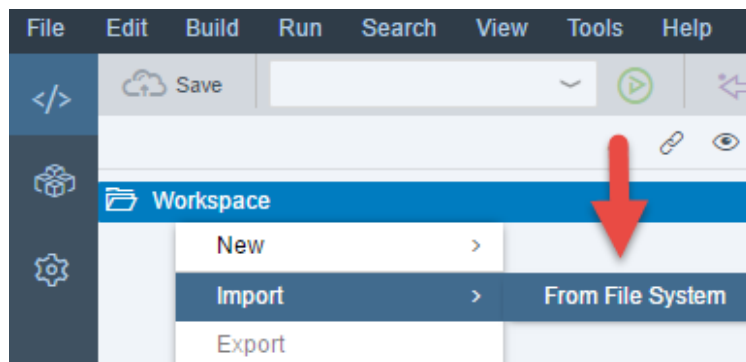
- SAP Web IDE for URL `https://lt5123.wdf.sap.corp:53075`
- User ID `HBD362_###`
- Password `${assigned by instructor}`
- Session `${assigned by instructor}`

User numbers are printed on a card next to your workstation, select the user number for your session and append the 3-digit user number after HBD362_ (i.e. HBD362_001 ... HBD362_120)

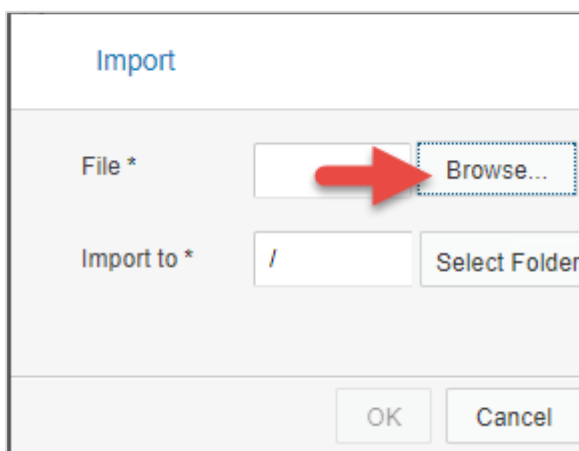
Explanation	Screenshot
1. Logon to the SAP Web IDE using <u>Google Chrome</u>	
2. Click 'Don't show' the Tips and Tricks window on startup > Close	

3. Start by importing the workshop material

Workspace > right click >
Import from File System

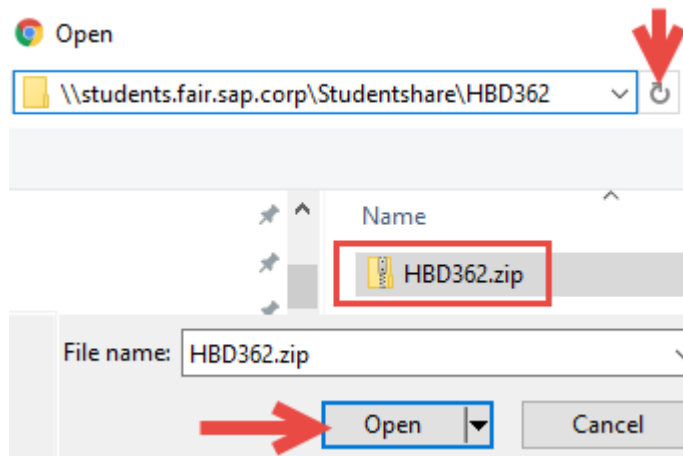


4. Click Browse to select the ZIP file to import



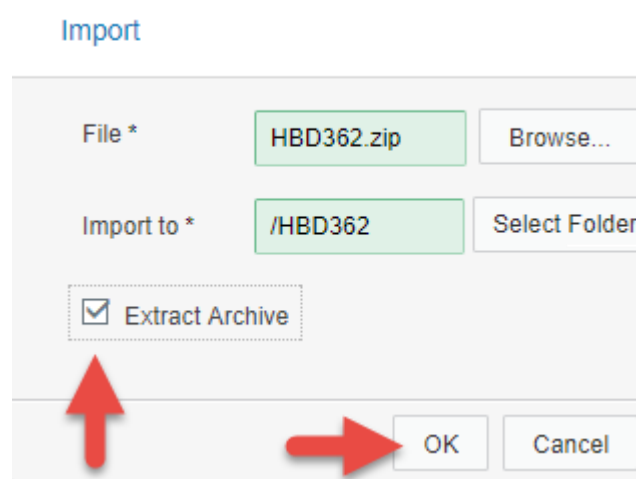
5. When prompted for a file to import, locate the ZIP file located on the student share:

\\students.fair.sap.corp\Student
share\HBD362



6. Important: Check Extract Archive

Click OK

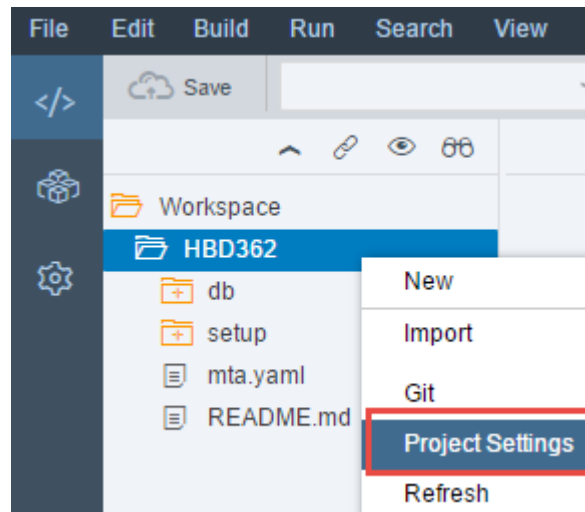


7. On the bottom-right side of the screen open the console and confirm that the import succeeded

```
7:15:35 AM (import) Import request sent
7:15:52 AM (import) Import request completed
successfully
```

8. Set the development SPACE

Right click on the project (HBD362) > Project Settings > Space



9. Select HDB362

Hint: To support isolation of the development environment, dedicated spaces are used for building and running projects

Click Save and Close

Space

Select a space in SAP HANA XS Advan

Space

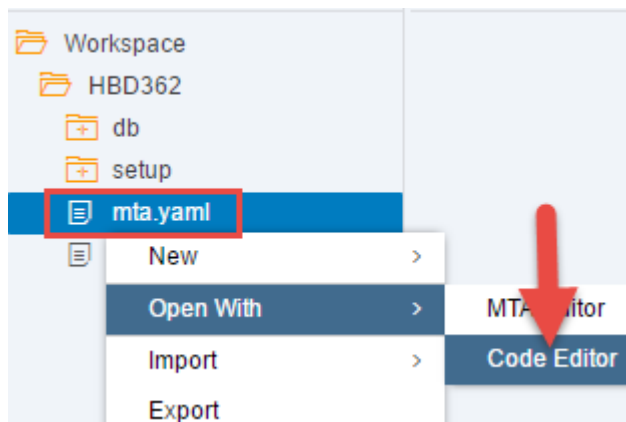
HBD362

Save

Close

10. Edit the MTA descriptor (aka yaml file) and add your User ID

Note: The multi-target application (MTA) descriptor file contains the metadata of all entities used in an application or used by it during deployment or runtime, and the dependencies between them



11. Delete \${user} and replace with your assigned User ID

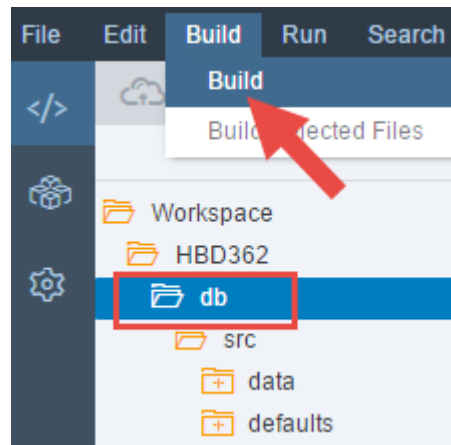
Hint: Replace XXX with your assigned User ID

```
resources:
- name: hdb362.srv
  parameters:
    config:
      schema: $(user) HBD362_XXX
  properties:
```

12. Save and Build the database module

Select db > Build

Hint: You can create a deployment archive by selecting and building the root folder HBD362. In this workshop, you only need to build the database module (db) project. This will generate the runtime database objects



13. Ensure the build completed successfully!

Hint: Open the console to see detailed messages

```
Deployment to container HBD362_XXX_1 done
[Deployment ID: none].
(2s 686ms)

7:23:11 AM (DIBuild) ***** End of
/HBD362/db Build Log *****

7:23:11 AM (DIBuild) Build results link:
https://ld9993.wdf.sapcloud.net:53075/che/build
all/7?arch=zip

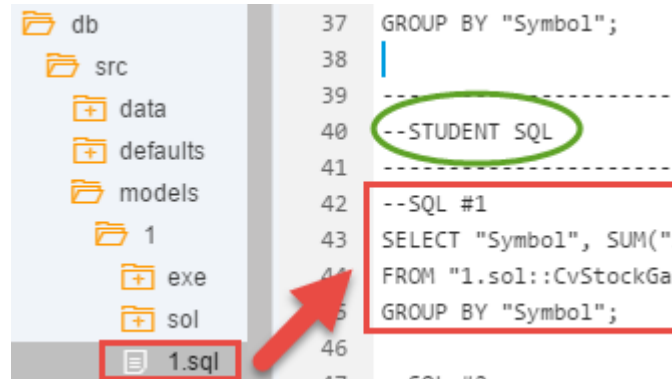
7:23:12 AM (Builder) Build of /HBD362/db
completed successfully.
```



14. Each exercise contains sample SQL statements (located in the root folder of each exercise)

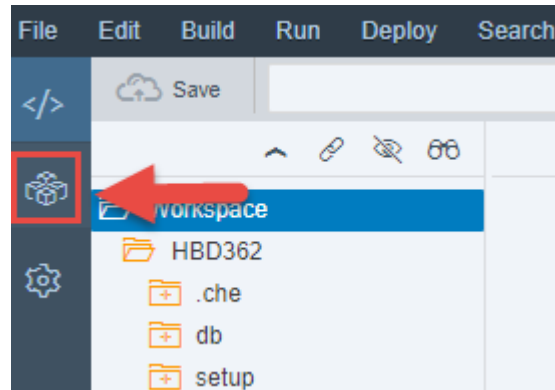
In the workspace tile open the content tree and Expand db > src > models > 1 > open 1.sql

Each file contains both solution and student SQL statements. Scroll down and always look for the STUDENT SQL section > copy the SQL #1 statement

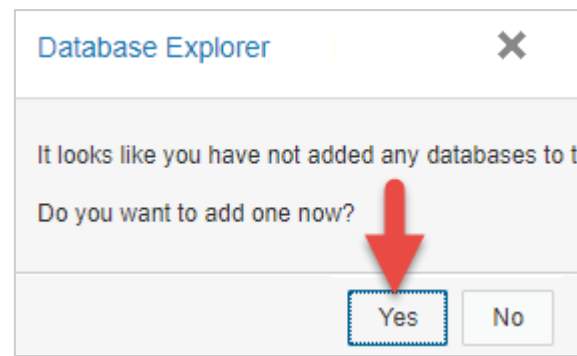


15. Open the Database Explorer and the SQL Console

Hint: On the left of the screen click on the Database Explorer icon



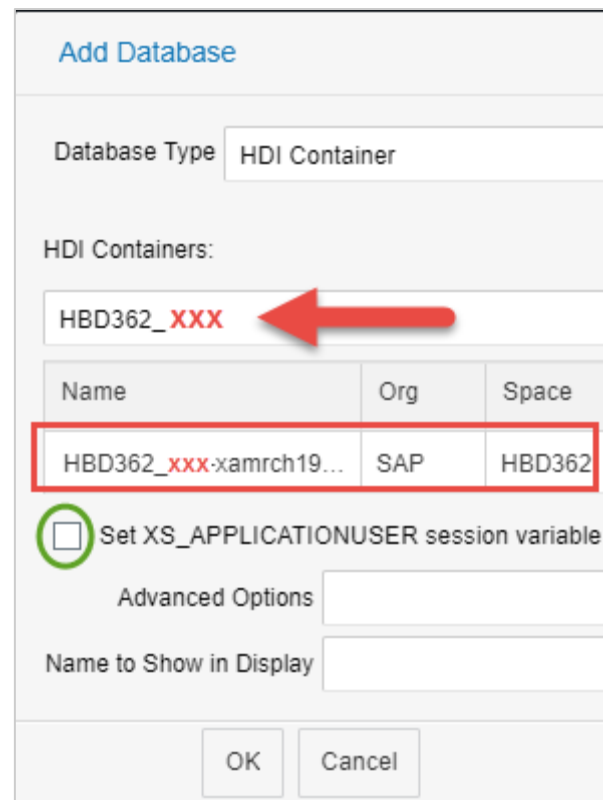
16. If prompted to add a database click Yes; (otherwise click + sign)



17. Search and select your HDI container

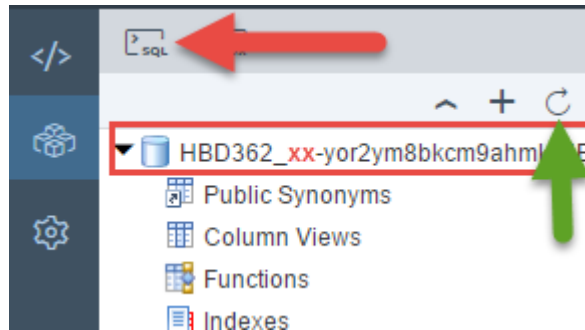
Hint: Make sure to select your own container, replace XXX with your assigned User ID

Important: uncheck "Set XS_APPLICATIONUSER"



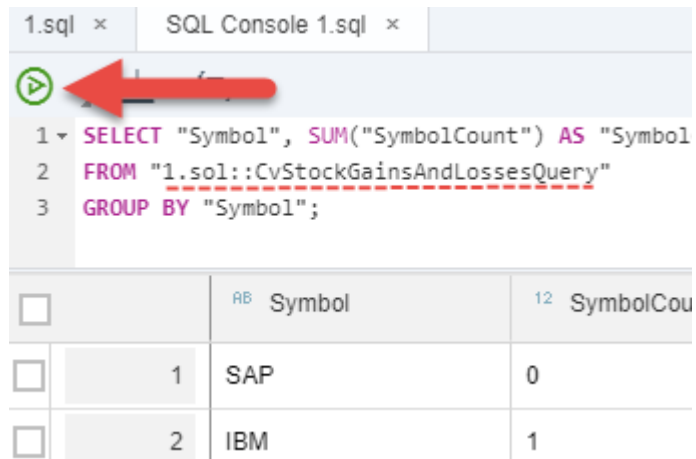
18. Select your container > click
Open SQL Console

Hint: If you do not see your
container, click the refresh icon



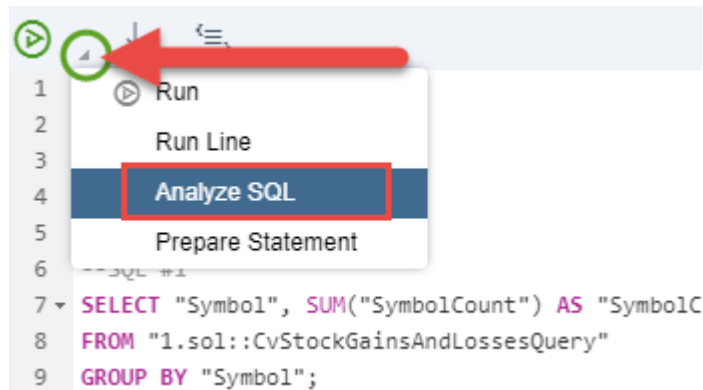
19. Paste the previously copied
SQL statement > Execute

Hint: Notice models are not
deployed to the <_sys_bic>
schema, but instead to an
isolated container-schema

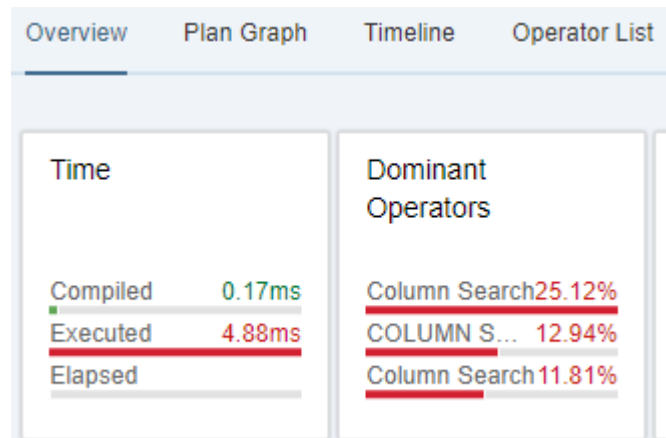


20. Query analysis is an important
best practice during modeling.
Several exercises require you
to analyze the SQL as follows:

Click on the small drop-down-
list icon to the right of the green
execute button > Analyze SQL

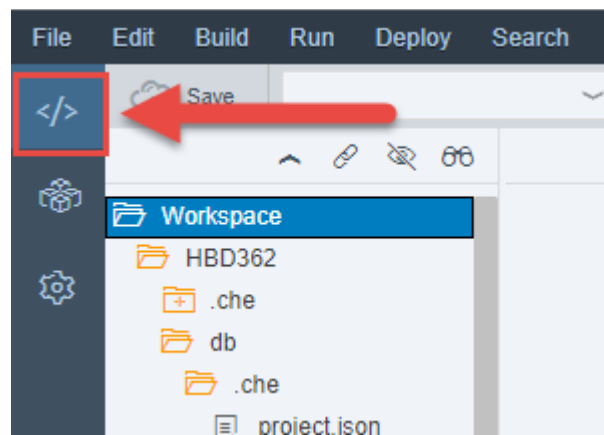


21. Result: You are presented with the execution overview.



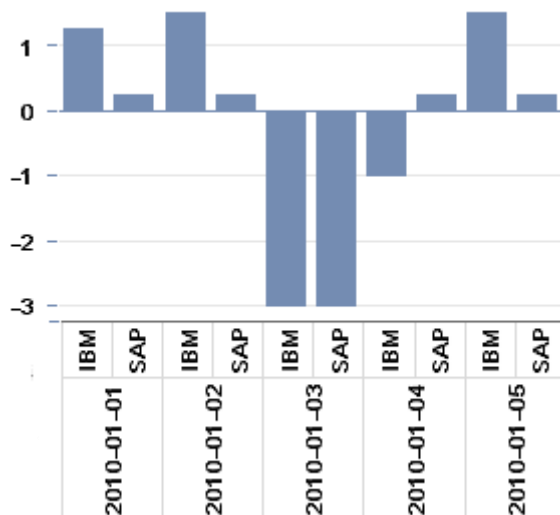
22. You can stop here for now, setup is complete!

Hint: Close any opened .sql files and click on the Development icon to go back to the development perspective



EXERCISE 1 - Multi level aggregation (15 Minutes, 30 steps)

In this exercise, you will compare 2 stock prices after 1 week of trading. The requirements are to show if the stock price ended up higher or lower by the end of the week and to show how many days the stock price increased

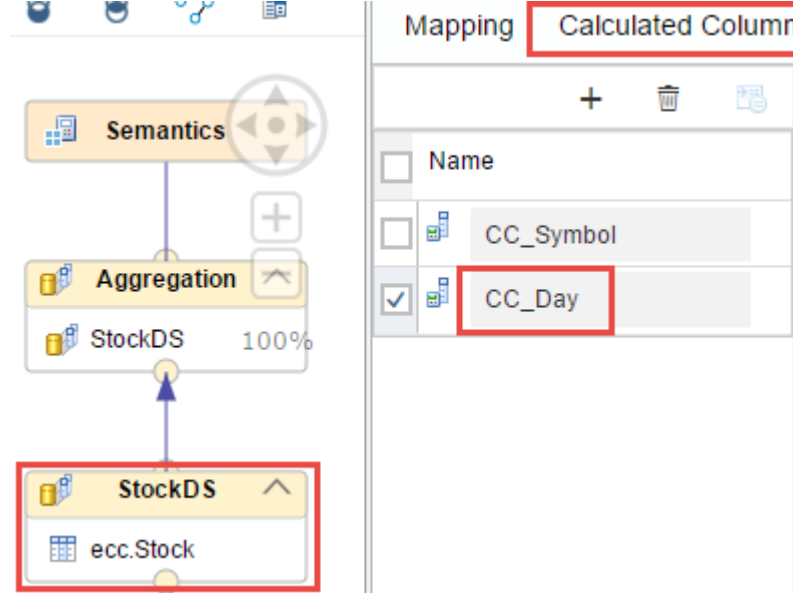
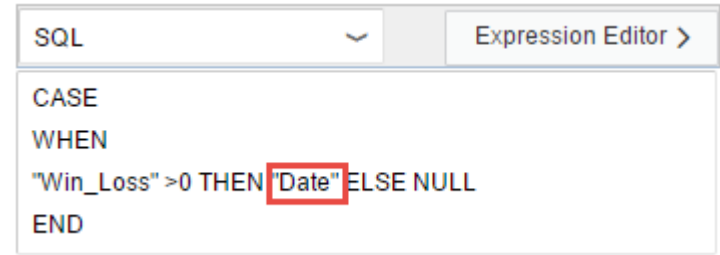
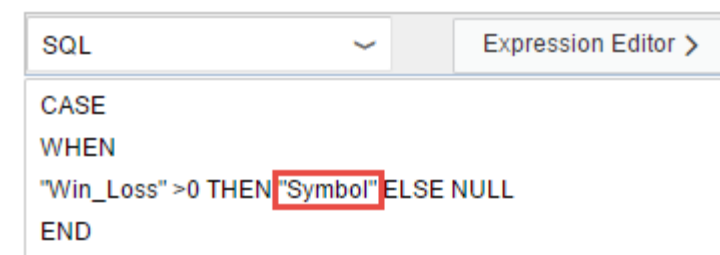
Expected Results

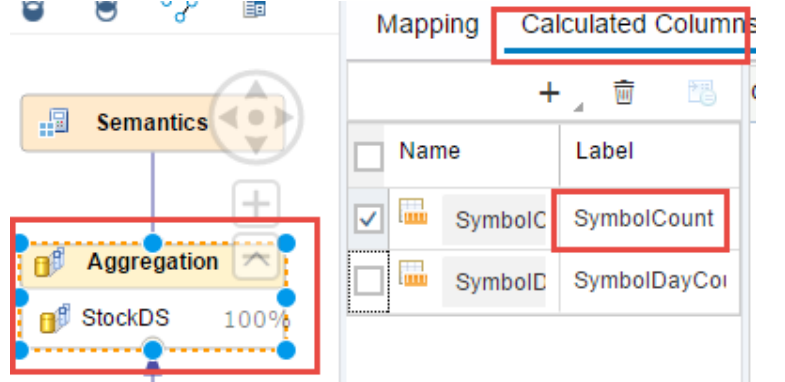
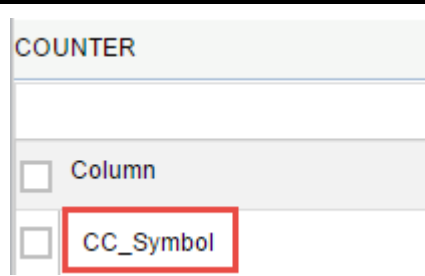
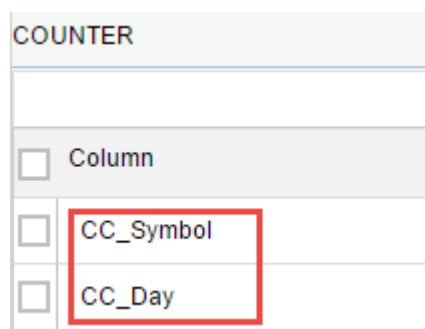
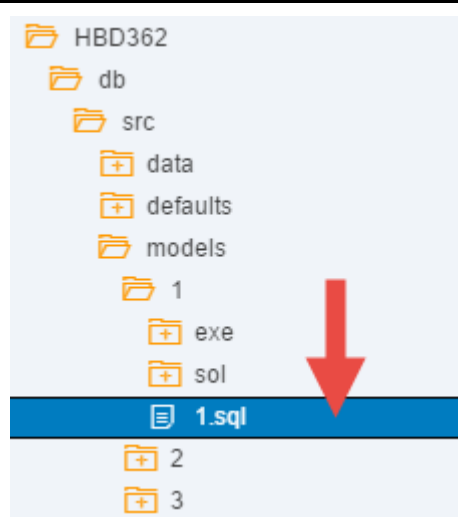
Symbol	Trending Up	Days
IBM	1	3
SAP	0	4

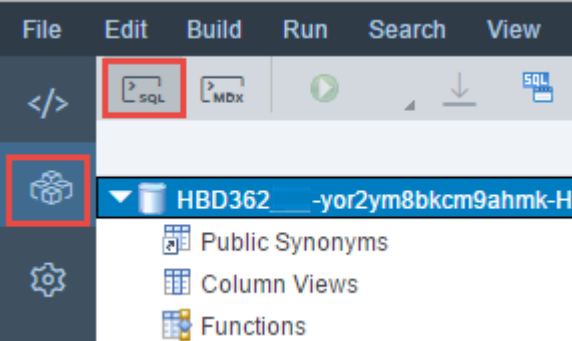
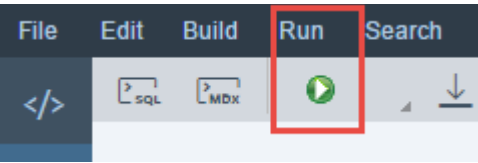
SAP rose 4 days, had a single bad day that brought the stock down for the entire week. IBM had 2 bad days but recovered the losses by end of week

Your assignment is to review the model and to determine why unexpected results are shown for certain queries. When each measure is individually queried the calculations are correct, however when both measures are queried simultaneous the calculations are incorrect. After you have analysed the situation proceed to fix the model

Explanation	Screenshot
<p>1. Open the solution calculation model and start your analysis</p> <p>HDB362 > db > src > models > 1 > sol > Stock Gains and Losses Query</p>	<pre> HDB362 ├── db │ ├── src │ │ ├── data │ │ ├── defaults │ │ ├── models │ │ │ ├── 1 │ │ │ │ ├── exe │ │ │ │ └── sol │ │ └── CvStockGainsAndLossesQuery.hdbcalculationview │ │ ├── CvStockGainsAndLossesStackedSolutionQuery.hdb │ │ ├── CvStockGainsAndLossesUnionSolutionQuery.hdbca │ │ └── 1.sql </pre>

<p>2. Select the Stock data source table and review the 2 calculated columns</p>	 <p>The screenshot shows the Qlik Sense data model. On the left, a flow diagram shows 'ecc.Stock' (a table icon) connected to 'StockDS' (a data source icon), which is connected to 'Aggregation' (an aggregation icon), which is finally connected to 'Semantics' (a semantic layer icon). The 'StockDS' node is highlighted with a red box. On the right, the 'Mapping' pane is open, showing a table with columns 'Name', 'CC_Symbol', and 'CC_Day'. The 'CC_Day' column is highlighted with a red box.</p>
<p>3. The CC_Day calculated column serves as an intermediate step used in a counter later in the flow. If the total trades for a given day's stock are positive, then record the date. The distinct counter will sum-up all the unique positive days</p>	 <p>The screenshot shows the SQL Expression Editor. It has a dropdown menu set to 'SQL' and a button 'Expression Editor >'. The SQL code is: <code>CASE WHEN 'Win_Loss' > 0 THEN 'Date' ELSE NULL END</code>. The word 'Date' is highlighted with a red box.</p>
<p>4. Similarly, the CC_Symbol calculated column serves as an intermediate step used in a counter later in the flow. If the total trades for a given week and stock are positive, then record the stock symbol. The distinct counter result will then either be 1 or 0 for each stock symbol</p>	 <p>The screenshot shows the SQL Expression Editor. It has a dropdown menu set to 'SQL' and a button 'Expression Editor >'. The SQL code is: <code>CASE WHEN 'Win_Loss' > 0 THEN 'Symbol' ELSE NULL END</code>. The word 'Symbol' is highlighted with a red box.</p>

<p>5. Review the counters:</p> <p>Select the topmost Aggregation node > Calculated columns</p>	 <p>The screenshot shows the Semantic Model editor. The 'Aggregation' node is selected and highlighted with a red box. The 'Calculated Columns' tab is active, showing a table with columns 'Name' and 'Label'. The table contains two rows: 'SymbolC' with 'SymbolCount' and 'SymbolD' with 'SymbolDayCo'. The 'SymbolCount' cell is highlighted with a red box.</p>
<p>6. Select SymbolCount > notice the distinct calculation is based on the individual Symbol calculated column</p>	 <p>The screenshot shows the 'COUNTER' configuration panel. The 'Column' section is expanded, and 'CC_Symbol' is selected and highlighted with a red box.</p>
<p>7. Select the SymbolDayCount > notice the distinct calculation is based on multiple calculated columns (Symbol and Day)</p> <p>Hint: Each of these calculated columns have different levels of granularity</p>	 <p>The screenshot shows the 'COUNTER' configuration panel. The 'Column' section is expanded, and both 'CC_Symbol' and 'CC_Day' are selected and highlighted with a red box.</p>
<p>8. Review the results by executing SQL statements > open the sample SQL statement file > db > src > models > 1 > 1.sql</p>	 <p>The screenshot shows a file explorer view. The path 'HBD362 > db > src > models > 1 > 1.sql' is highlighted. A red arrow points to the '1.sql' file.</p>

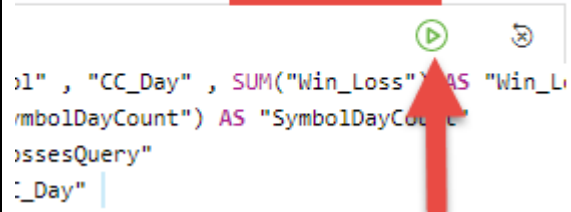
9. Under the Student SQL section > Copy SQL #1	<pre> ----- --STUDENT SQL ----- --SQL #1 SELECT "Symbol", SUM("SymbolCount") AS "SymbolCount" FROM "1.sol::CvStockGainsAndLossesQuery" GROUP BY "Symbol"; </pre>						
10. Open the Database Explore > select your container and click on SQL Console							
11. Paste the SQL statement into the SQL Console copied earlier > click execute							
12. Notice: The Symbol count is correct. SAP had a single bad day that brought the stock down for the week. IBM was slightly higher for the same week	<table border="1"> <thead> <tr> <th>Symbol</th><th>SymbolCount</th></tr> </thead> <tbody> <tr> <td>SAP</td><td>0</td></tr> <tr> <td>IBM</td><td>1</td></tr> </tbody> </table>	Symbol	SymbolCount	SAP	0	IBM	1
Symbol	SymbolCount						
SAP	0						
IBM	1						
13. Repeat the steps, copy SQL #2 into the SQL Console > Execute Hint: If COPY is not visible as a menu option use CTRL+C	<pre> --SQL #2 SELECT "Symbol", SUM("SymbolDayCount") AS "SymbolDayCount" FROM "1.sol::CvStockGainsAndLossesQuery" GROUP BY "Symbol"; </pre>						
14. Notice: The Symbol Day count is correct. SAP's stock was positive for 4 days' vs IBM 3 days	<table border="1"> <thead> <tr> <th>Symbol</th><th>SymbolDayCount</th></tr> </thead> <tbody> <tr> <td>SAP</td><td>4</td></tr> <tr> <td>IBM</td><td>3</td></tr> </tbody> </table>	Symbol	SymbolDayCount	SAP	4	IBM	3
Symbol	SymbolDayCount						
SAP	4						
IBM	3						

15. Repeat the steps, copy SQL #3 into the SQL Console > Execute	--SQL #3 SELECT "Symbol", SUM("SymbolCount") AS "SymbolCount", SUM("SymbolDayCount") AS "SymbolDayCount" FROM "1.sol::CvStockGainsAndLossesQuery" GROUP BY "Symbol";																		
16. Notice: When both measures are simultaneously requested the Symbol Count for SAP is <u>incorrect</u> !	<table><tr><th>RB</th><th>Symbol</th><th>12</th><th>SymbolCount</th><th>12</th><th>SymbolDay</th></tr><tr><td></td><td>SAP</td><td></td><td>1</td><td></td><td>4</td></tr><tr><td></td><td>IBM</td><td></td><td>1</td><td></td><td>3</td></tr></table>	RB	Symbol	12	SymbolCount	12	SymbolDay		SAP		1		4		IBM		1		3
RB	Symbol	12	SymbolCount	12	SymbolDay														
	SAP		1		4														
	IBM		1		3														
17. Your assignment starts: Debug the model to see which data is brought into context. Select the Semantics node, then click on the little bug icon to start the debugger																			
18. When the Debug Query tab appears showing the SQL statement > Click execute																			
19. Click on the Debug Query tab again to look at the data																			

20. Click Execute again

Parameters (0)

Debug Query



```

1" , "CC_Day" , SUM("Win_Loss") AS "Win_L
mbolDayCount") AS "SymbolDayCount"
ssesQuery"
:_Day"

```

21. Analysis: The following data is brought into context when both measures are queried:

The CC_Day calculated column is using (and therefore bringing into context) a finer grain Date column that is affecting the granularity of the coarser grain CC_Symbol calculation. As a result, the CC_Symbol calculation is calculated daily and before aggregation (instead of on the entire week and after aggregation)

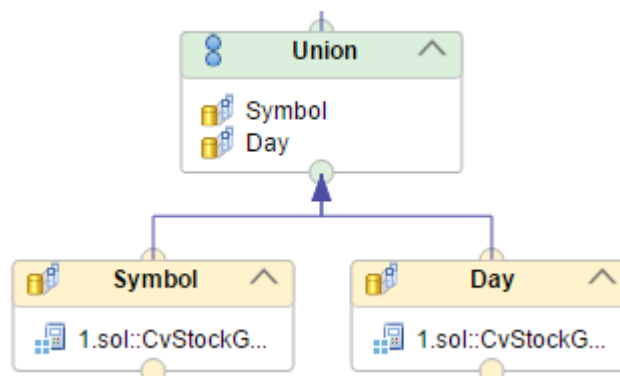
Date	Symbol	CC_Symbol	CC_Day	Win_Loss
201...	IBM	IBM	2010-0...	1.25
201...	IBM	IBM	2010-0...	1.5
201...	IBM	IBM	2010-0...	1.5
201...	SAP	SAP	2010-0...	0.25
201...	SAP	SAP	2010-0...	0.25
201...	SAP	SAP	2010-0...	0.25
201...	SAP	SAP	2010-0...	0.25
201...	IBM			-3
201...	IBM			-1
201...	SAP			-3

22. Informational: When the date column is not brought into context the CC_Symbol executes after aggregation resulting in the counter to be 1 for IBM and 0 for SAP

Symbol	Win_Loss	CC_Symbol
SAP	-2	?
IBM	0.25	IBM

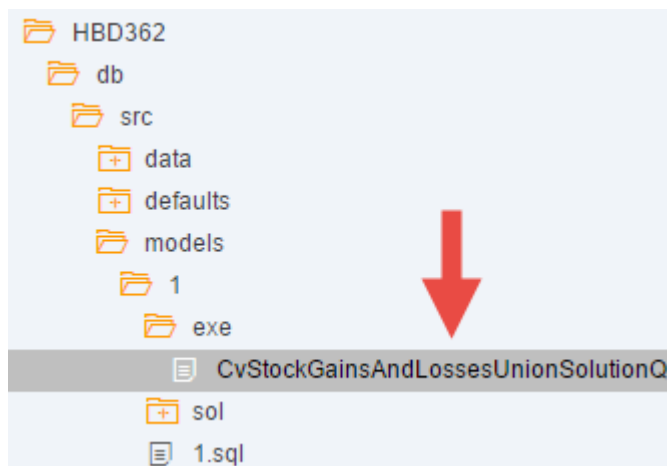
23. Recommendation 1: Union

Model instead so that each counter uses its own context. This can be done by placing the calculations into separate branches (keeping same-granularity-level calculations on the same branch)

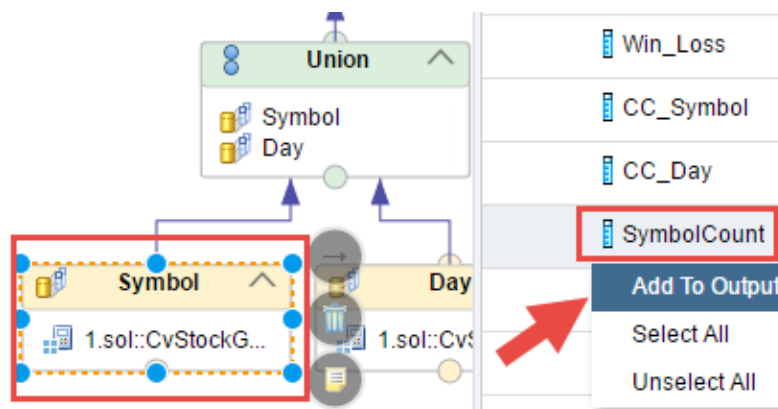


24. Your assignment starts:

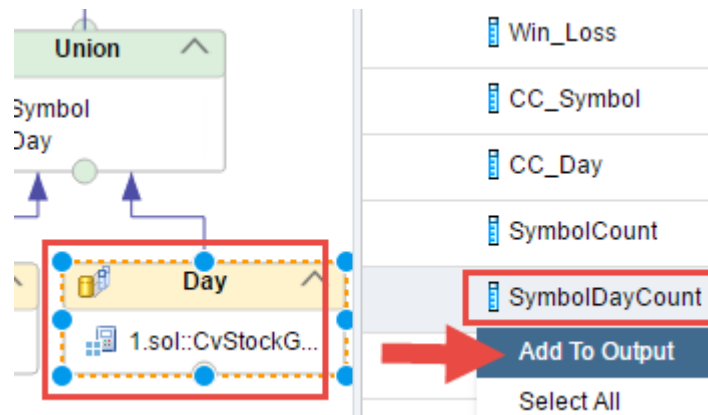
Expand models > 1 > exe > open Stock Gains and Losses Union Solution Query



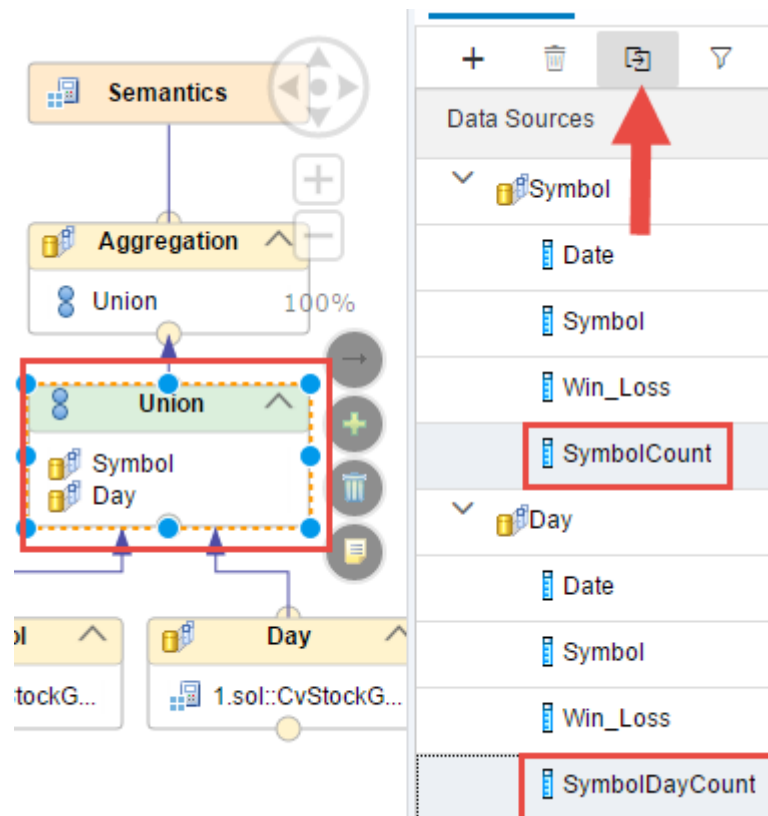
25. Underneath the Union, select the left branch (Symbol) > mapping > select SymbolCount > Add to Output



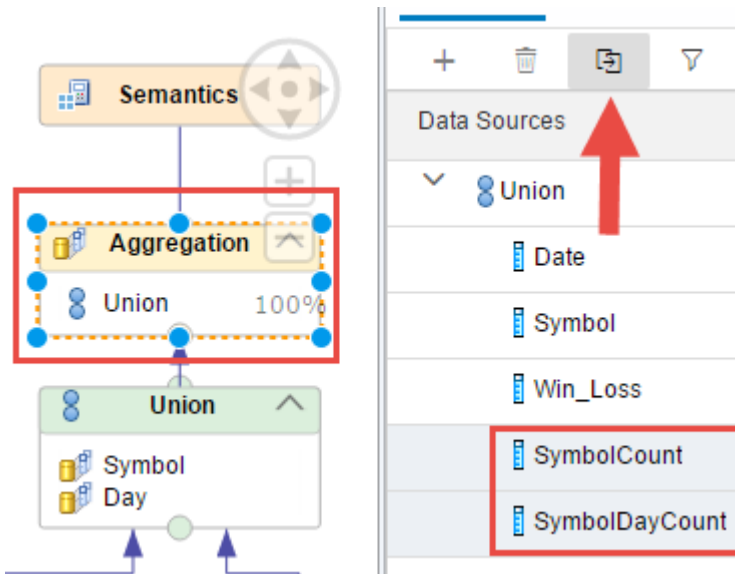
26. Similarly select the right branch (Day) > mapping > select SymbolDayCount > Add to Output



27. Select the Union > Mapping > Add both Symbol Count and Symbol Day Count to the output

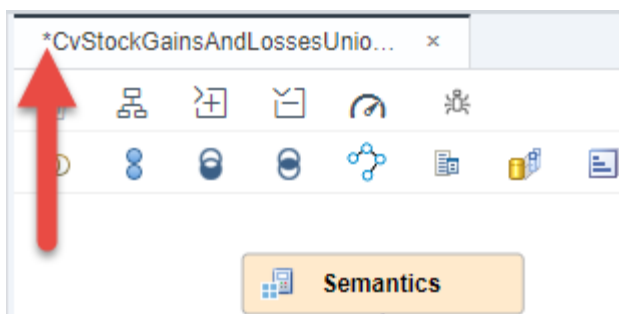


28. Similar within the topmost aggregation node add both measures to the output as well

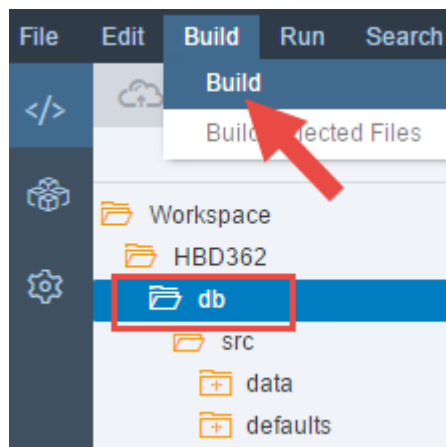


29. Important. Save your work!

Hint: The star is an indication of unsaved work!



30. Save & build the db project



31. Ensure the build finished successful

Build of /HBD362/db completed successfully

32. Test the model > copy SQL #5 into the SQL Console > Execute

```
--SQL #5
SELECT "Symbol", SUM("SymbolCount") AS "SymbolCount",
FROM "1.exe::CvStockGainsAndLossesUnionSolutionQuery"
GROUP BY "Symbol";
```

33. Notice: The counters are correct!

```
1 SELECT "Symbol", SUM("SymbolCount") AS "SymbolCount", SUM
2 FROM "1.exe::CvStockGainsAndLossesUnionSolutionQuery"
3 GROUP BY "Symbol";
```

Result x Messages x

Rows (2)

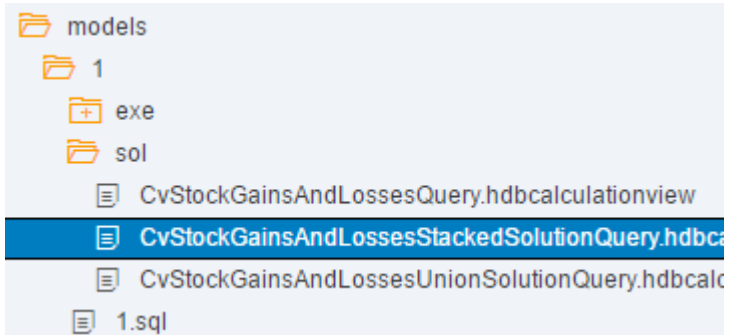
	AB	Symbol	12 SymbolCount	12 SymbolDa
<input type="checkbox"/>	1	SAP	0	4
<input type="checkbox"/>	2	IBM	1	3

34. Recommendation 2:

Stacking models:

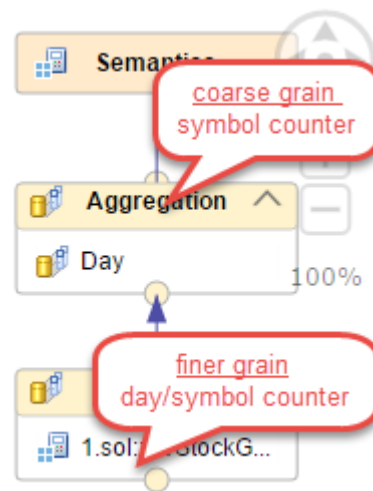
open 1 > sol > Stacked
Solution Query

Instead of using a union a
stacked approach would
similarly work



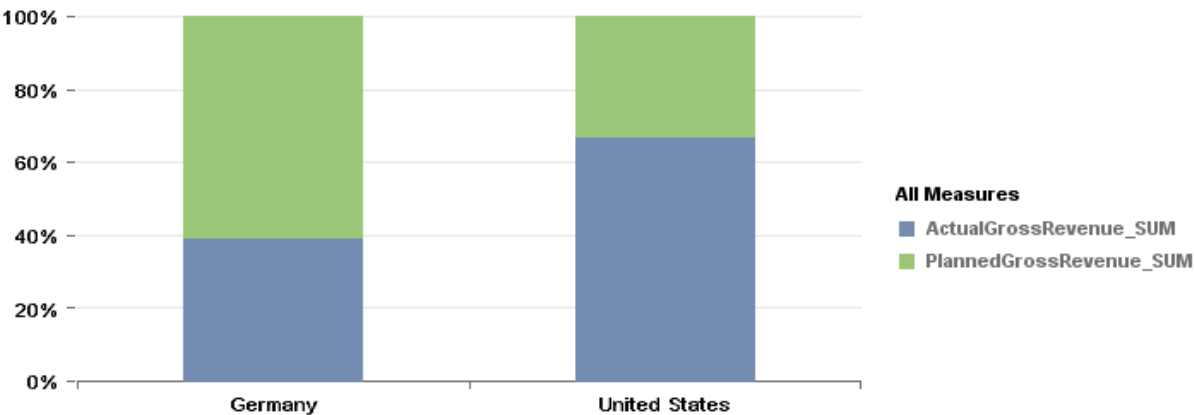
35. Strictly informational:

Notice the finer grain day-
and-symbol counter in
modeled in the child
calculation model; and then
coarser grain symbol
counter is modeled in a
wrapper calculation model



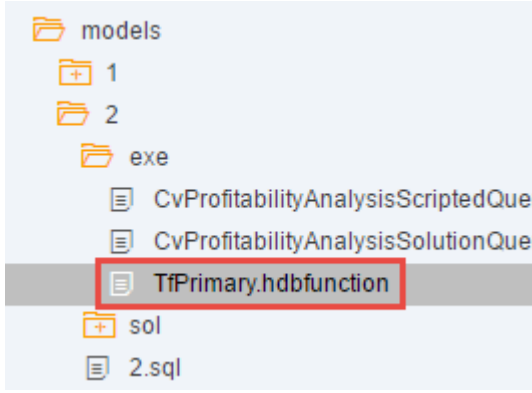
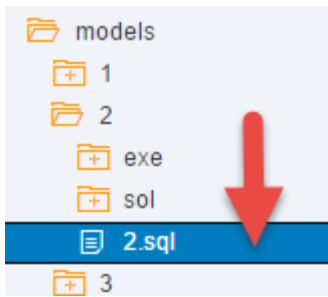
EXERCISE 2 - Scripted models vs Modeled models (20 MINUTES, 35 steps)

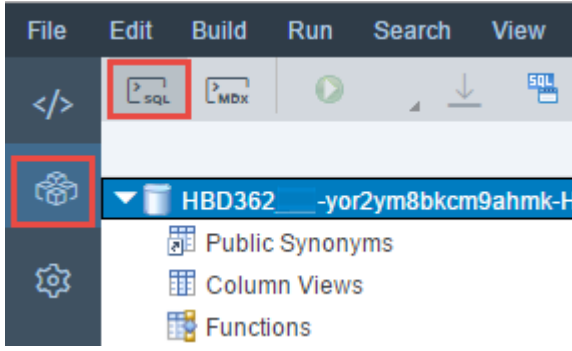
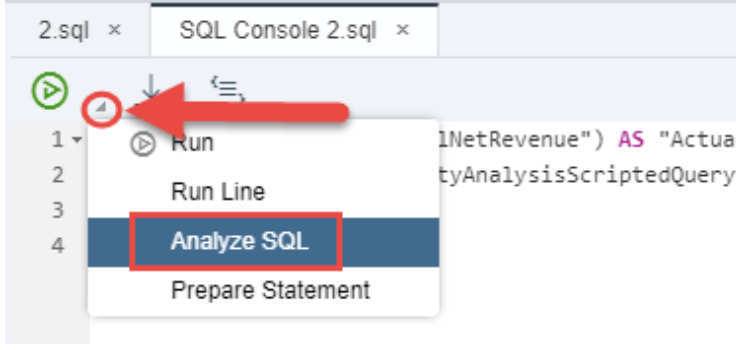
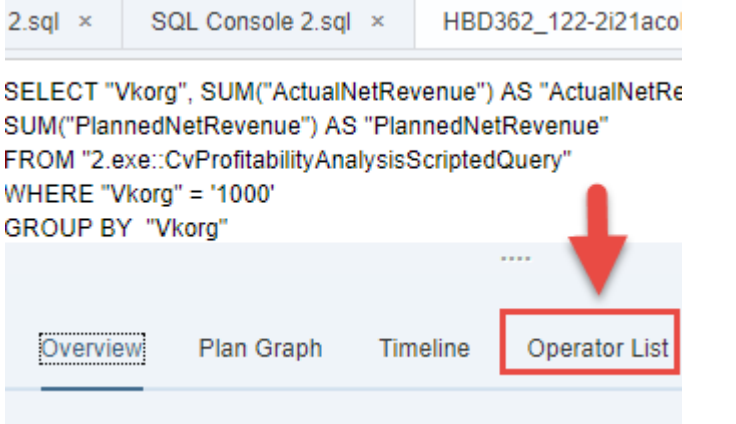
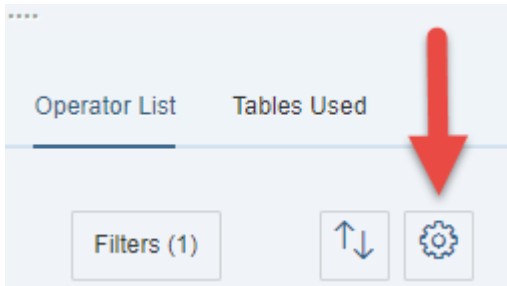
This exercise showcases the classical ERP CO-PA probability analysis scenario that provides multi-dimensional insights into a company's product profitability. The final solution implements various best practice techniques such as the use of calculation views, star joins, join optimizations and unions to compare actual vs planned data



The CO-PA solution model needs to support dynamic ad-hoc query analyses. It is implemented using 2 different approaches, first using modeled calculation views and second using SQLScript. You will compare the execution of both approaches and at the end of the exercise you will understand the various optimizations that are native to graphical calculation models to support dynamic ad-hoc analyses.

Explanation	Screenshot
1. Expand models > 2 > exe > open the Scripted calculation model query	
2. Notice a table function is used as a data source, proceed to open the table function	

<p>3. Within the exe directory open the TfPrimary table function</p>	
<p>4. Review the code:</p> <p>Notice the actuals-query, followed by the planned-query which is then combined using a union operator</p> <p>Note: The transaction tables are directly queried, and the dimensional tables are wrapped inside other calculation models</p>	<pre> 7 8 ITAB_ACTUALS = SELECT 9 "FACT"."Perio" AS "Perbl", "FACT"."Gjahr" AS 10 "PRODUCT"."Matnr" AS "Matnr", "PRODUCT"."Mak 11 "LOCATION"."Kunnr" AS "Kunnr", "LOCATION"."L 12 "LOCATION"."Regio" AS "Regio", "LOCATION"."B 13 SUM("FACT"."Vv010") AS "ActualGrossRevenue", 14 SUM("FACT"."Vv070") AS "ActualSalesDeduction 15 SUM("FACT"."Vv290") AS "ActualProductVarianc 16 SUM("FACT"."Vv010" - "FACT"."Vv070") AS "Act 17 FROM "ecc.CopaCelIdea" "FACT" </pre>
<p>5. Proceed to analyze the execution of the scripted model</p> <p>models > 2 > open 2.sql</p>	
<p>6. Copy SQL #1, open the Database Explorer</p> <p>Hint: Notice the WHERE clause filter</p>	<pre> ----- --STUDENT SQL ----- --SQL #1 SELECT "Vkorg", SUM("ActualNetRevenue") AS "A FROM "2.exe::CvProfitabilityAnalysisScriptedQ WHERE "Vkorg" = '1000' GROUP BY "Vkorg"; </pre>

7. Select your container and open the SQL Console	
8. Paste the SQL > then click on the small icon next to the green execute button > Analyze SQL	
9. Before starting the analysis, first click on the Operator List tab	
10. Then click on settings to personalize the display	

11. Unselect all columns and only select the following 3 columns:

- Operator Name
- Accessed DB Objects
- Details

Columns

↑ ↓ Search

☐ All

☐ CPU Time

☒ Operator Name

☒ Accessed DB Objects

☐ Output Rows

☐ Output Bytes

☒ Details

OK Cancel

12. Then click on the up and down (Sort) icon

Filters (1) ↑↓ ⚙

13. Sort by Operator Name

Sort Settings


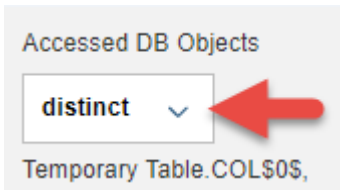
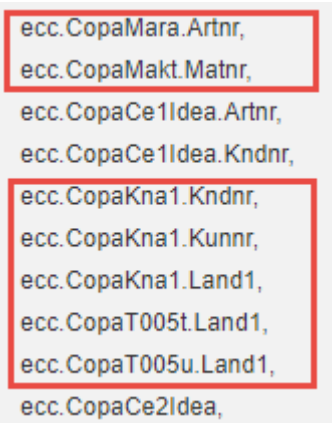
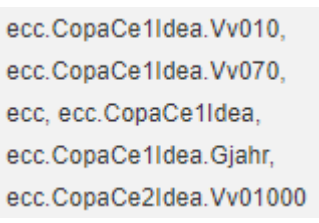
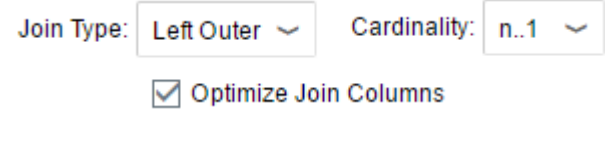
☐ Offset (ms)

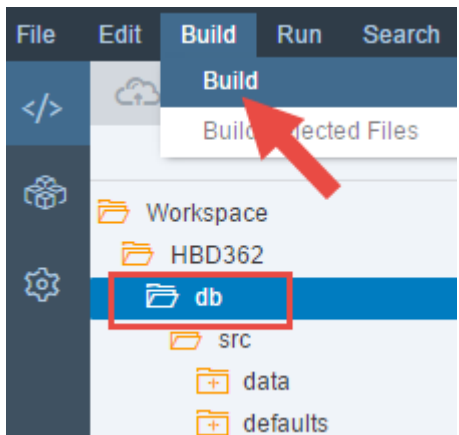
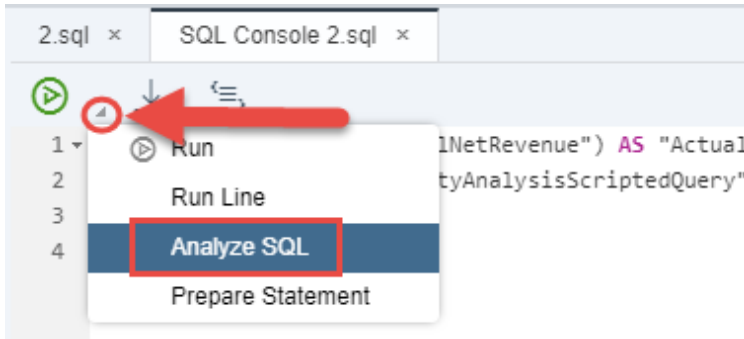
☐ Execution Time (ms)

☐ CPU Time (ms)

☒ Operator Name

OK Cancel

<p>14. At first glance it appears that the query is showing optimal execution:</p> <p>The model is unfolded (meaning that the underlying table details are shown without references to any calculation models). Filters are pushed down (i.e. VKORG) adequately</p>	
<p>15. Under Accessed DB Objects change the display to distinct</p>	
<p>16. Notice the first optimization difference when using scripts. <u>Additional</u> tables are accessed even though the query requested columns from only the 2 transaction tables</p> <p>Hint: Graphical models contain additional optimizations that can prune un-used tables and joined columns (i.e. using <u>cardinality</u> / <u>optimize join settings</u>)</p>	
<p>17. Informational: Later in the exercise you will see how graphical models can ignore unused tables resulting in more optimal execution</p>	
<p>18. Informational: Spoiler alert: Specifying join properties within graphical calculation models can enable pruning</p>	

<p>19. Next, the usage of scalar variables may block optimizations</p> <p>Edit the table function > 2.exe::TfPrimary > uncomment theses 8 and 9 (the scalar variable declaration line and the assignment line)</p>	<pre>*TfPrimary.hdbfunction x 1 FUNCTION "2.sol::TfPrimary" () 2 RETURNS "ecc.ActualsPlannedT" 3 LANGUAGE SQLSCRIPT 4 SQL SECURITY INVOKER AS 5 BEGIN 6 7 --# blocker 8 DECLARE V_YEAR NVARCHAR(4); 9 SELECT YEAR(ADD_YEARS(CURRENT_DATE, -13)) 10</pre>
<p>20. Towards the end of the function uncomment the WHERE clause on line 87</p>	<pre>84 SUM("PlannedOtherExpenses") AS "PlannedOtherExpenses" 85 SUM("PlannedNetRevenue") AS "PlannedNetRevenue" 86 FROM :ITAB_RESULT 87 WHERE "Gjahr" = :V_YEAR 88 GROUP BY "Perb1", "Gjahr", "Kndnr", " --</pre>
<p>21. Save & build the db project</p>	
<p>22. Ensure the build is succeeded</p>	<p>Build of /HBD362/db completed successfully.</p>
<p>23. Analyze SQL #1 again > click on the small icon next to the green execute button > Analyze SQL</p> <p>Hint: Find out if both filters are pushed down</p>	

24. Under the Operator List >
Notice the model is not
unfolded (instead of table
details the calculation model is
shown), potentially blocking
optimizations

Note: Scalar variables used
inside the SQLScript body is
not yet fully optimized and may
adversely impact performance

Overview Plan Graph Timeline **Operator List** Tables L

Operators (234)

Operator Name	Accessed DB Objects
none	none

Calculation Model 2.exe::CvProfitabilityAnalysisScriptedQuery

Execute Model 2.exe::CvProfitabilityAnalysisScriptedQuery

25. Proceed with additional
analysis to confirm if the filters
(VKORG and YEAR) are
pushed down

Sort by the Operator Name

Sort Settings

☐ Offset (ms)

☐ Execution Time (ms)

☐ CPU Time (ms)

☒ Operator Name

OK Cancel

26. The execution of the model is
not optimal; YEAR is pushed
down but VKORG is blocked

Potential solutions:
Use explicit and mandatory
input parameters for static
scenarios or ...
Create modeled calculation
views for dynamic scenarios
that prohibit mandatory
parameters

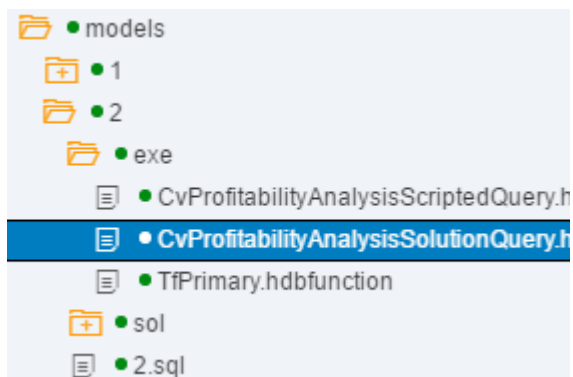
Basic predicate _SYS_CE__popid_3_6f549 Vkorg = '1000'
17223635D5_29

Basic predicate ecc.CopaMakt Spras = 'E'

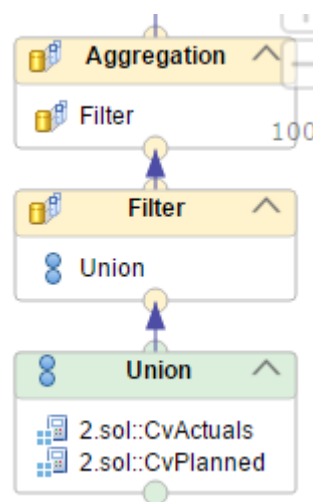
Basic predicate ecc.CopaCe2Idea Gjahr = '2004'

27. Your next assignment starts:
Analyze the execution of
graphical calculation models:

Within the exe folder open the
Profitability Analysis Solution
Query calculation model (aka
graphical calculation model)



28. Notice instead of using
SQLScript the solution is
created entirely using
calculation models with unions
and star joins, etc.

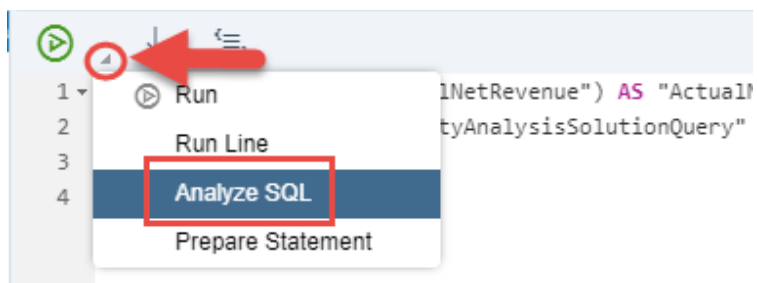


29. Analyze the execution against
the graphical calculation model

As before copy SQL #2 and
paste it into the SQL Console

```
--SQL #2
SELECT "Vkorg", SUM("ActualNetRevenue") AS "ActualNetRevenue"
FROM "2.exe::CvProfitabilityAnalysisSolutionQuery"
WHERE "Vkorg" = '1000'
GROUP BY "Vkorg";
```

30. Analyze the SQL



31. The execution of this model is optimal

Operators > Accessed DB Objects [distinct]

- The model is unfolded (only tables are referenced)
- Any unused tables and columns are ignored

Accessed DB Objects

distinct

ecc.CopaCe1Idea.Vv010,
ecc.CopaCe1Idea.Vv070,
ecc, ecc.CopaCe1Idea,
ecc.CopaCe1Idea.Gjahr,
ecc.CopaCe2Idea.Vv01000
ecc.CopaCe2Idea.Vv07000
1, ecc.CopaCe2Idea,
ecc.CopaCe2Idea.Gjahr

32. Under Operators > Sort by Operator Name > Both filters (YEAR and VKORG) are passed down to the fact tables

Note: If you recall earlier in the scripted scenario, only VKORG was pushed down, YEAR was blocked due to the usage of scalar variables. See next how the YEAR filter is modeled instead in graphical calculation models to allow push down

Operators (59)

Filter

Operator Name

Accessed DB Objects

Details

none

none

Basic predicate

ecc.CopaCe2Idea

Gjahr = '2004'

Basic predicate

ecc.CopaCe2Idea

Vkorg = '1000'

Basic predicate

ecc.CopaCe1Idea

Gjahr = '2004'

Basic predicate

ecc.CopaCe1Idea

Vkorg = '1000'

33. Within the Semantics area > Parameters > open the IP_YEAR_DUMMY input parameter

Hint: Parameters can be maintained from all nodes not only in the Semantics area

Semantics

View Properties

Columns (21)

Hierarch



GENERAL



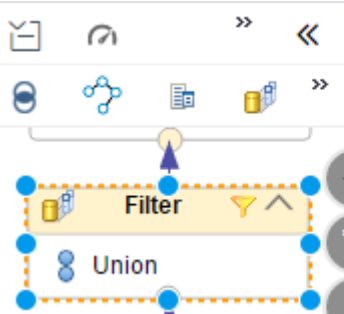
Name

Label



IP_YEAR_

IP_YEAR_DUMM

34. Notice, the dummy input parameter contains the expression to calculate 13 years in the past	<div><div>DEFAULT VALUE(S)</div><table><tr><td><input type="checkbox"/> Type</td><td>Value</td></tr><tr><td><input type="checkbox"/> Expression</td><td>YEAR(ADD_YEARS(CURRENT_DATE,-13))</td></tr></table></div>	<input type="checkbox"/> Type	Value	<input type="checkbox"/> Expression	YEAR(ADD_YEARS(CURRENT_DATE,-13))
<input type="checkbox"/> Type	Value				
<input type="checkbox"/> Expression	YEAR(ADD_YEARS(CURRENT_DATE,-13))				
35. The input parameter is then used as a YEAR filter (in the aggregation node above the union)	<div><div>The diagram shows a workflow with two nodes: a 'Filter' node (yellow) and a 'Union' node (blue). The 'Filter' node is positioned above the 'Union' node, and a blue arrow points from the 'Filter' node to the 'Union' node. Both nodes are enclosed in a dashed orange box.</div><div><div>Filter</div><div>MappingCalculated Columns</div><div>Language: SQL</div><div>"Gjahr"='\$\$IP_YEAR_DUMMY\$\$'</div></div></div>				

EXERCISE 3 - Dynamic ranking (10 MINUTES, 14 steps)

In this exercise, the rank node is used to determine the most valuable soccer players. The requirements are, to display the best 2 players per a team and the best 3 players in the league

Original data set:

Player	Team	Goals
Arjen Robben	Bayern Munich	3
Max Kruse	Borussia M.Glad...	5
Karim Bellar...	Bayer Leverkusen	4
Richardo R...	Wolfsburg	3
Mario Götze	Bayern Munich	6
Naldo	Wolfsburg	2
Uwe Hune...	Paderborn	2
Abdul Rah...	Augsburg	0
Kevin De Br...	Wolfsburg	0
Xabi Alonso	Bayern Munich	1

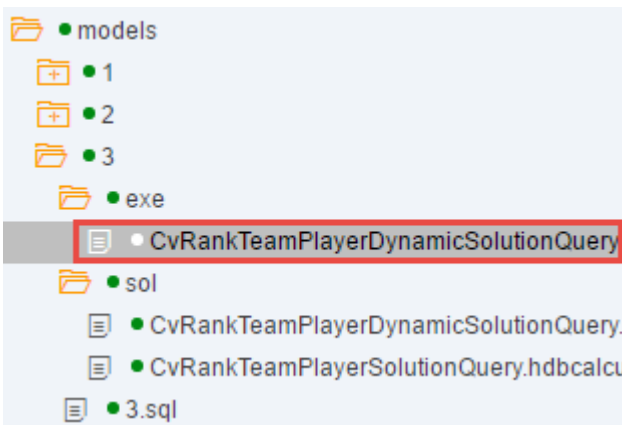
Expected results: top 2 players in a team

Player	Team	Goals
Mario Götze	Bayern Munich	6
Arjen Robben	Bayern Munich	3
Max Kruse	Borussia M.G...	5
Karim Bellar...	Bayer Leverk...	4

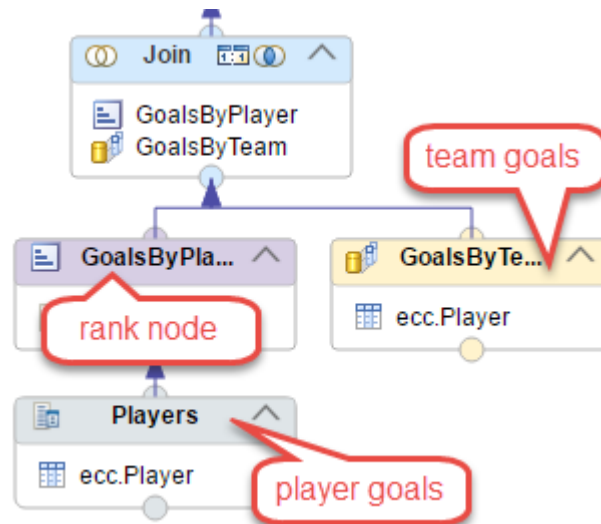
Expected results: top 3 players in league

Player	Goals	Team
Mario Götze	6	10
Max Kruse	5	5
Karim Bellarabi	4	4

Your task is to review the model and to determine why unexpected results are shown for certain queries. The ranking is correct when querying the best 2 players per team, however the ranking is wrong when querying the best 3 players across the league. After you have analysed the situation proceed to enhance and fix the model

Explanation	Screenshot
1. The model is already prepared for you. Navigate to models > 3 > exe > open the calculation model	

2. Notice the 2 separate branches to calculate the player and team goals



3. Select the rank node > definition. The rank node is configured to sort the top N player goals within a team from high to low

Note: The player column is not a partitioned column, instead the parent column (team) acts as the window frame within which players will be ranked based on their goals

Sort Direction :

Threshold :

Order By :

☐ Dynamic Partition Elements

Partition By

☐ Partition By Column

☐

4. Start by executing SQL statements against the model

Open 3.sql and copy both SQL #1 and #2 into the SQL Console

```
--STUDENT SQL

--SQL #1
SELECT "TeamName", "PlayerName", SUM("PlayerGoals")
FROM "3.exe::CvRankTeamPlayerDynamicSolutionQuery"
GROUP BY "TeamName", "PlayerName";

--SQL #2
SELECT "PlayerName", SUM("PlayerGoals") AS "PlayerGo
FROM "3.exe::CvRankTeamPlayerDynamicSolutionQuery"
GROUP BY "PlayerName";
```


5. Execute SQL #1 > The results are correct.

Note: The input parameter to display only the top 2 players per team

3.sql x	untitled 4.sql x
1	--SQL #1
2	SELECT "TeamName", "PlayerName", SUM("PlayerGoals")
3	FROM "3.exe::CvRankTeamPlayerDynamicSolutionQuery"
4	(PLACEHOLDER."\$\$TOP_N_PLAYERS\$\$"=>2) GROUP BY "TeamName"

Result x				
Rows (8)				
<input type="checkbox"/>		TeamName	PlayerName	PlayerGoals
<input type="checkbox"/>	1	Bayern Munich	Mario Götze	6
<input type="checkbox"/>	2	Bayern Munich	Arjen Robben	3
<input type="checkbox"/>	3	Borussia M.Glad	Max Kruse	5

6. Execute SQL #2 > The results are wrong!

The top 3 players across the league is incorrect; Instead of returning 3 rows (Mario, Max, Karim) all the players are returned

ProfitabilityAnalysisSolution...	SQL Console 1.sql x
<div> </div>	
--SQL #2 SELECT "PlayerName", SUM("PlayerGoals") AS "PlayerGoals" FROM "3.exe::CvRankTeamPlayerDynamicSolutionQuery" (PLACEHOLDER."\$\$TOP_N_PLAYERS\$\$"=>3) GROUP BY "PlayerName";	

Messages x		
(10)		
	PlayerName	PlayerGoals
1	Mario Götze	6
2	Xabi Alonso	1
3	Arjen Robben	3
4	Max Kruse	5
5	Karim Bellarabi	4

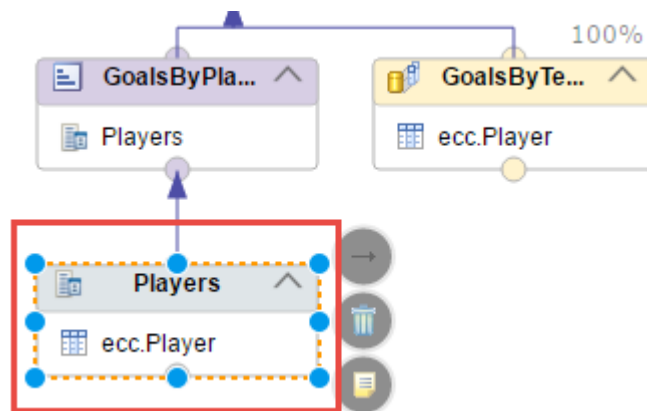
7. Explanation: The rank node requires a parent column to act as the window frame. In our example the team (parent) is used, the players within each team is then ranked. The model currently does not support a grand-parent (league) concept to rank individual players across the entire league.

These situations call for a dummy calculated column to act as a temporary window frame column

12 X	AB Player	12 Go...	AB Team
1	Arjen Robbe	3	Bayern Munich
1	Max Kruse	5	Borussia M.Gladbach
1	Karim Bellarbi	4	Bayer Leverkusen
1	Richard Roth	3	Wolfsburg
1	Mario Götze	6	Bayern Munich
1	Naldo	2	Wolfsburg

8. Your assignment starts:

If not already opened navigate to models > 3 > exe > open the calculation model > underneath the rank node > select the player data source at the bottom of the model > then over on the right > select calculated columns



9. Click the + icon and create a new integer DUMMY calculated column (default value of 1)

GENERAL

Name: * DUMMY

Data Type: * INTEGER

Length:

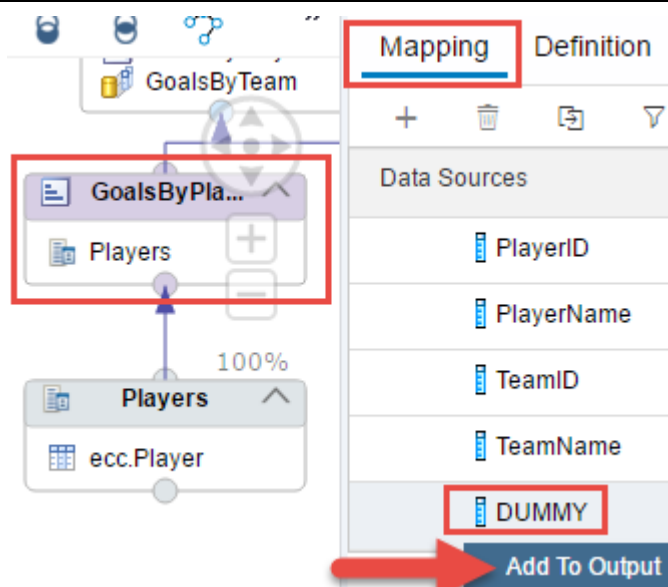
Scale:

EXPRESSION

SQL E

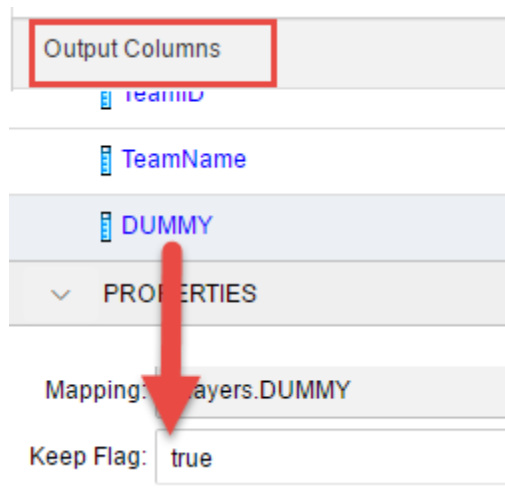
1

10. Select the rank node > Mapping > add the DUMMY calculated column to the output



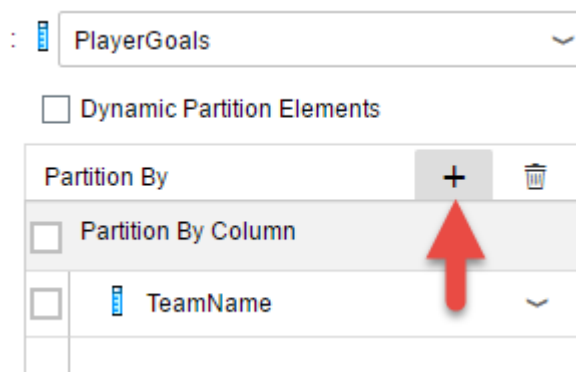
11. Within Mapping > Output Columns > select the DUMMY column

Force DUMMY to be brought into context by setting the keep flag to true



12. With the rank node selected > select the Definition area

Add the DUMMY column as a partition column



13. Make sure to select Dynamic Partition Elements

Note: Thus, the team name column will not be brought into context when ranking Players across the league

☒ Dynamic Partition Elements

Partition By +

☐ Partition By Column

☐ TeamName

☐ DUMMY

14. Save, build and execute SQL #2 again. The results are now correct for both queries

3.sql x untitled 1.sql x

```
1 SELECT "PlayerName", SUM("PlayerGoals") /
2 FROM "3.exe::CvRankTeamPlayerDynamicSolut
3 (PLACEHOLDER."$$TOP_N_PLAYERS$$"=>3)
4 GROUP BY "PlayerName";
```

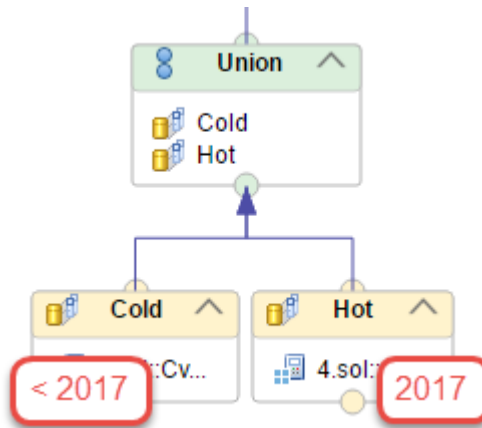
Result x

Rows (3)

<input type="checkbox"/>	PlayerName	PlayerGoals
<input type="checkbox"/>	1 Mario Götze	6
<input type="checkbox"/>	2 Max Kruse	5
<input type="checkbox"/>	3 Karim Bellarabi	4

EXERCISE 4 - Explicit vs Implicit union pruning (20 MINUTES, 34 steps)

This exercise demonstrates how to model scenarios that supports hot data (that exists in SAP HANA) and cold data (that exist in an external database) without having to expose the technical location details to end users. In these types of situations, current data is most often queried, it is crucial that union pruning is enabled so that the large amounts of historical data are ignored by default unless explicitly or implicitly requested by the user

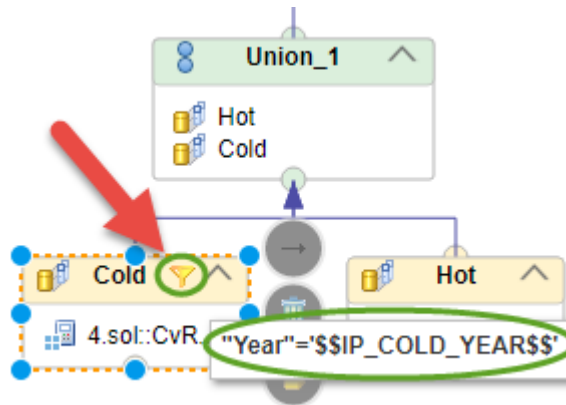


Your assignment is to model the union and to implement both explicit and implicit input source pruning

Explanation	Screenshot
1. Within models > 4 > exe > open CvSalesByYearComparison Query	

2. Select the cold branch, notice that a filter is defined. With the mouse hover over the filter to see the details.

The cold historical data is always filtered using a mandatory input parameter



3. Next, click on the parameters tab (over on the right) to see the details of the input parameter. Notice that that input parameter contains a default expression – the previous year is calculated unless a user overrides the expression by supplying a specific historical year

DEFAULT VALUE(S)	
<input type="checkbox"/> Type	Value
<input type="checkbox"/> Expression	YEAR(NOW())-1

4. Execute SQL statements against the model

Open 4.sql > copy SQL #1 and past it into the SQL Console > execute

Note: No input parameter is supplied, therefore the current year and the previous year is automatically calculated

```
--SQL #1
SELECT "Country",
SUM("PreviousAmount") AS "2016",
SUM("CurrentAmount") AS "2017"
FROM "4.exe::CvSalesByYearComparisonQuery"
WHERE "Country" IN ('USA', 'Germany', 'France',
GROUP BY "Country" ORDER BY "Country";
```

	RB Country	12 2016	12 2017
1	Brazil	46568.67	9165.4
2	France	19963.06	2825.4
3	Germany	82665.7	18099
4	Sweden	24084.4	2826.5
5	USA	103372.18	4692.4

5. To confirm that the results are correct execute SQL #2, this will force 2016 using an input parameter

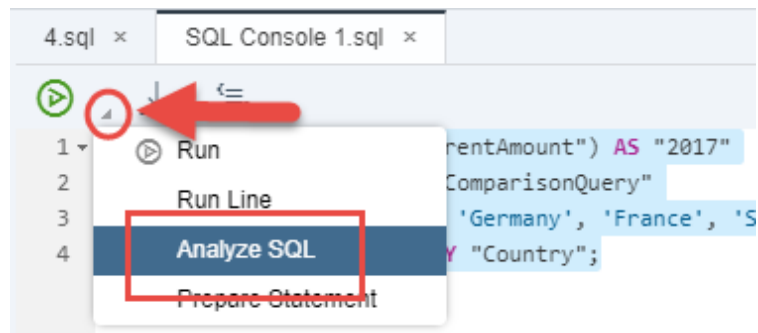
```
--SQL #2
SELECT "Country",
SUM("PreviousAmount") AS "2016",
SUM("CurrentAmount") AS "2017"
FROM "4.exe::CvSalesByYearComparisonQuery"
('PLACEHOLDER' = ('$$IP_COLD_YEAR$$', '2016'))
WHERE "Country" IN ('USA', 'Germany', 'France',
GROUP BY "Country" ORDER BY "Country";
```

	AB Country	12 2016	12 2017
1	Brazil	46568.67	9165.4
2	France	19963.06	2825.4
3	Germany	82665.7	18099
4	Sweden	24084.4	2826.5
5	USA	103372.18	4692.4

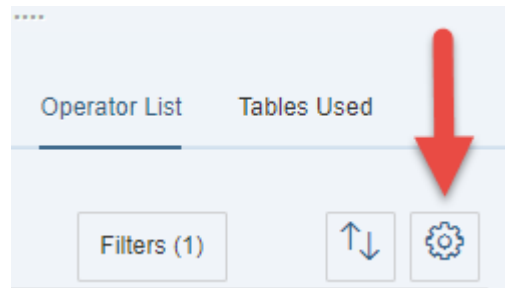
6. Copy and analyze SQL #3, this query is calculating hot data (2017), the expectation is that cold/historical data would be ignored

Note: Current data is loaded into the Orders table and all older historical data is loaded into a table called IQOrders

Note: In reality IQOrders will be a virtual table pointing to an external database table



7. Click on the Operator List tab > then to make analysis easier, click on settings to personalize the display



8. Unselect all columns and only select the following 3 columns:

- Operator Name
- Accessed DB Objects
- Details

Columns

↑ ↓ Search

☐ All

☐ CPU Time

☒ Operator Name

☒ Accessed DB Objects

☐ Output Rows

☐ Output Bytes

☒ Details

OK Cancel

9. Change the drop-down list to show the distinct database objects

Analysis: The execution shows that the model is not optimal.

Even though the end results are correct the historical data (IQOrders) was unnecessarily queried

Accessed DB Objects

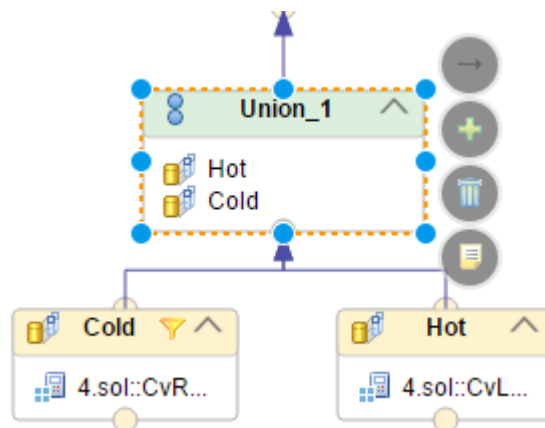
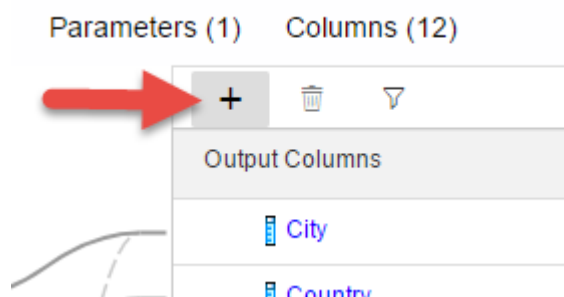
distinct

ecc.Orders.TotalAmount,
ecc.Customers.CustomerID,
ecc.Orders.CustomerID,
ecc.ecc.Orders,
ecc.Customers.Country,
6, ecc, ecc.Time,
ecc.IQOrders OrderDate,
ecc.Time.Date,
ecc.IQOrders.CustomerID,
ecc.Customers

10. Your assignment starts.

First enhance the model to support explicit union pruning

If not already opened. Open
models > 4 > exe >
CvSalesByYearComparison
Query > select the Union

11. Create a constant. On the right side of the screen > Mappings > In the Output Columns > Click +

12. Define the constant as follows:

Name: SOURCE VARCHAR(4)

Add a constant value for each
input source:

Hot = HANA
Cold = IQ

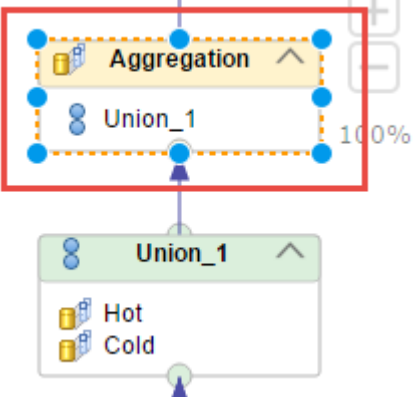
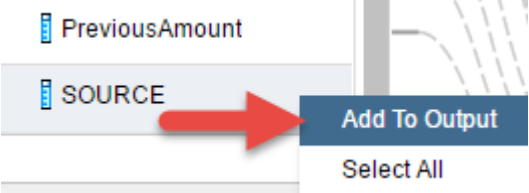
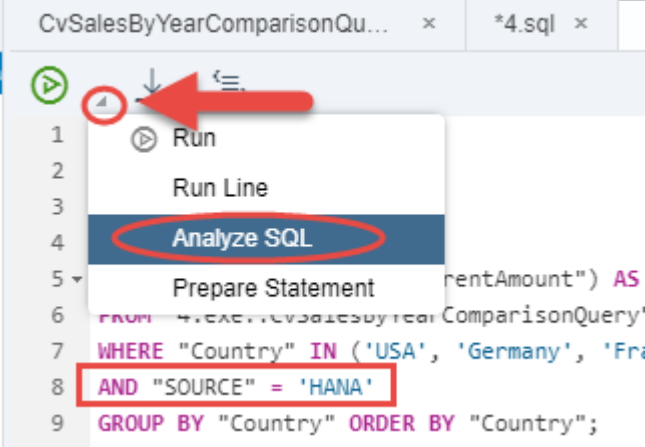
Name: * SOURCE

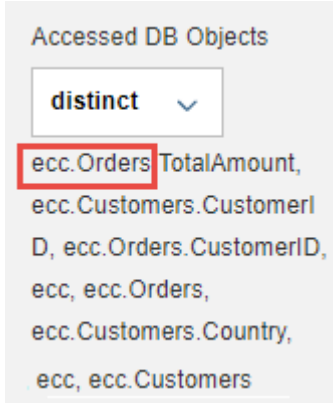
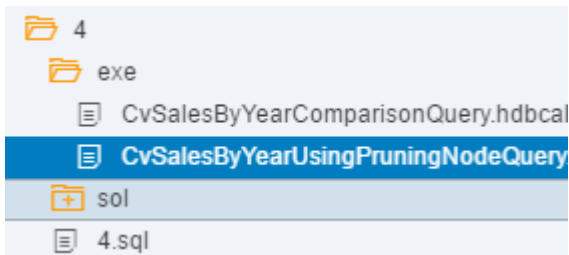
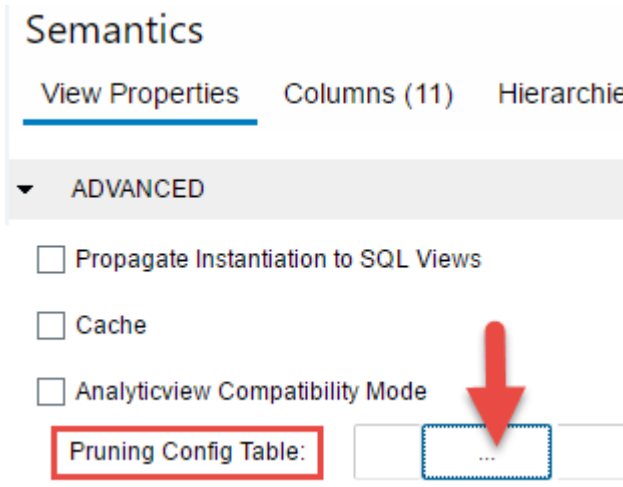
Data Type: * VARCHAR

Length: 4

MANAGE MAPPING

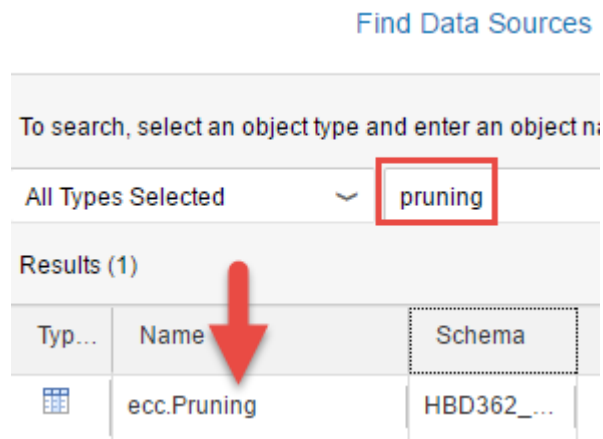
Source Model	Sourc...	Constant Value
Hot	▼	HANA
Cold	▼	IQ

<p>13. Click on the Aggregation node above the Union > Mapping</p>	
<p>14. On the left side of the mappings screen under data sources > add the SOURCE column to the output</p>	
<p>15. Save & build the db project.</p> <p>Make sure that the project compiled successfully</p>	<p>Build of /HBD362/db completed successfully</p>
<p>16. Copy and analyze SQL #4</p> <p>Notice hot data is explicitly queried using the SOURCE filter</p>	 <pre>1 2 3 4 5 6 7 8 9 CvSalesByYearComparisonQu... x *4.sql x Run Run Line Analyze SQL Prepare Statement FROM ...CvSalesByYearComparisonQuery' WHERE "Country" IN ('USA', 'Germany', 'Fri AND "SOURCE" = 'HANA' GROUP BY "Country" ORDER BY "Country";</pre>

<p>17. The result shows optimal execution, using explicit input source pruning all historical data is ignored</p> <p>Hint: IQOrders is not queried</p>	
<p>18. Your next assignment: Situations that demand more flexible predicates can instead enable implicit pruning</p> <p>Open 4 > exe > CvSalesByYear UsingPruningNodeQuery</p>	
<p>19. This mechanism does not use union constants nor input parameters. Instead a pruning configuration table with WHERE clause filters are used</p> <p>Select Semantics > View Properties > then scroll down to <u>Advanced</u> > Pruning Config Table > click on the icon to search for and select pruning table</p>	

20. When the search window appears, type in pruning and click search. Then select the Pruning table

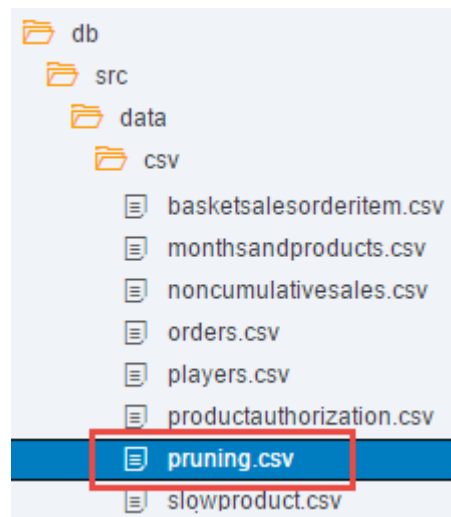
Save the model



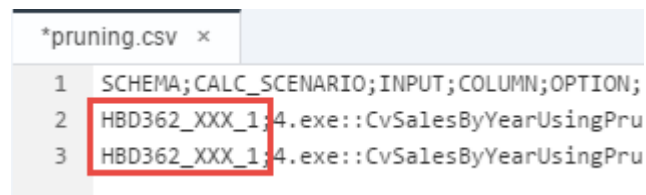
21. Next. Add your own model and schema specific information to the pruning configuration table

Expand db > src > data > csv > then edit pruning.csv

Hint: The next build will automatically import the data from the .csv file into the pruning table



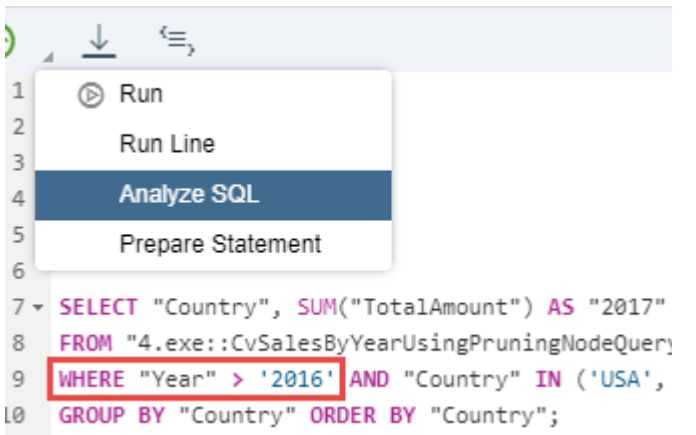
22. The pruning configuration mechanism relies on predicates defined in this table. Only 2 entries exist for your model, 1 for each branch in the union (Hot and Cold). Add your schema name by replacing XXX with your assigned User ID/schema name. See next

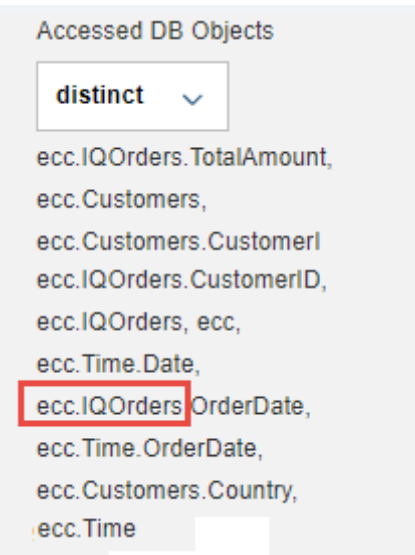


23. Find your schema in the build console trace, usually HBD362_XXX_1

Update the pruning.csv file by replace XXX with your physical schema name, then save the file

```
Deployment to container HBD362_XXX_1 done  
[Deployment ID: none].  
(2s 686ms)  
  
7:23:12 AM (Builder) Build of /HBD362/db  
completed successfully.
```

<p>24. Strictly informational: Within the pruning file/table notice the rules for each modeled branch</p> <p>Hint: The Cold and Hot names represent the input source names defined within the calculation model</p>	<pre>Cold;Year;<=;2016; Hot;Year;=;2017;</pre>
<p>25. Save and build the db project > Analyze SQL #6</p> <p>Notice the WHERE clause. The expectation is that all historical data would be ignored</p>	 <pre>1 2 3 4 5 6 7 SELECT "Country", SUM("TotalAmount") AS "2017" 8 FROM "4.exe::CvSalesByYearUsingPruningNodeQuery" 9 WHERE "Year" > '2016' AND "Country" IN ('USA', 10 GROUP BY "Country" ORDER BY "Country";</pre>
<p>26. Optimal execution > historical data is ignored and only current data is queried</p>	<p>Accessed DB Objects</p> <p>distinct ▾</p> <p>ecc.Orders.TotalAmount, ecc.Customers, ecc.Customers.CustomerID, ecc.Orders.CustomerID, ecc.Orders, ecc, ecc.Time.Date, ecc.Orders.OrderDate, ecc.Time.OrderDate, ecc.Customers.Country, 0, ecc, ecc.Time</p>
<p>27. Analyze SQL #7</p> <p>Notice the WHERE clause. The expectation is to query only historical data</p>	<pre>--SQL #7 SELECT "Country", SUM("TotalAmount") AS "2017" FROM "4.exe::CvSalesByYearUsingPruningNodeQuery" WHERE "Year" <= '2016' AND "Country" IN ('USA', 'CAN', 'MEX') GROUP BY "Country" ORDER BY "Country";</pre>

<p>28. Optimal execution. Only historical data (2015-2016) are queried, hot data is ignored</p>	 <p>Accessed DB Objects</p> <p>distinct ▾</p> <p>ecc.IQOrders.TotalAmount, ecc.Customers, ecc.Customers.CustomerID, ecc.IQOrders.CustomerID, ecc.IQOrders, ecc, ecc.Time.Date, ecc.IQOrders OrderDate, ecc.Time.OrderDate, ecc.Customers.Country, ecc.Time</p>
<p>29. Final important information:</p>	<p>Pruning optimization are currently blocked for queries without any aggregation. To enable tracing for pruning processing: Set the trace level for ceoptimizer to DEBUG in the index server section</p>

EXERCISE 5 - NON-cumulative key figures (15 MINUTES, 28 steps)

Non-cumulative key figures are calculated based on other key figures and characteristics, values are not stored but are calculated during runtime. Non-cumulative key figures are used in applications where users want to know the daily stock level or account balance. In the exercise the daily balance calculation depends on the previous day's daily balance calculation and therefore needs to be carried over to the next day

Data-set

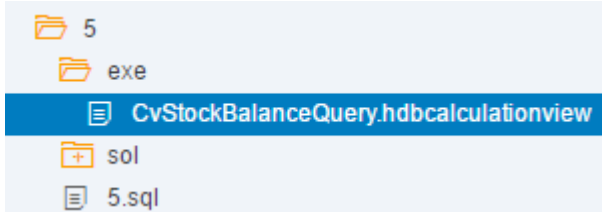
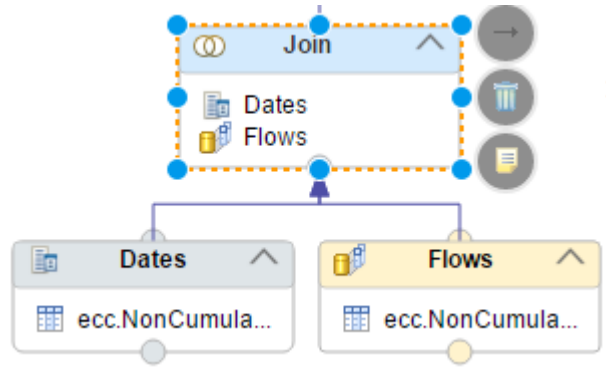
Date	Material	IN	OUT
20160501	Apple	25	0
20160502	Banana	50	0
20160503	Apple	30	20
20160504	Banana	40	15
20160505	Apple	25	15
20160506	Apple	20	10
20160507	Apple	35	25

Expected end of day results

Daily Balance
25
50
35
75
45
55
65

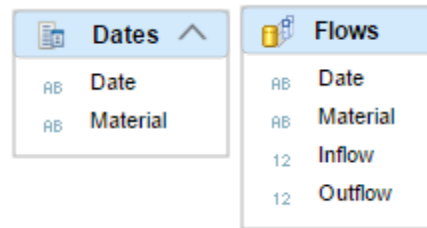
The solution consists of 4 steps:

1. Explode the data using a cross join
2. Calculate daily running balances
3. Implode/aggregate the data
4. Pick the original daily in/out values using a self-join

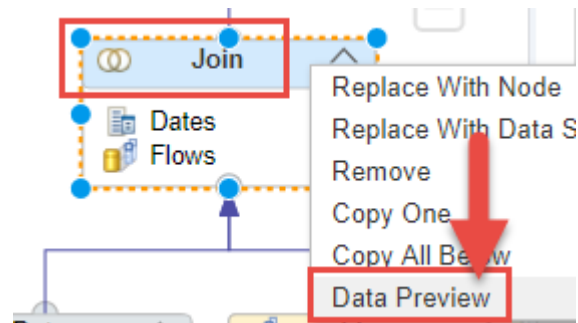
Explanation	Screenshot
1. Open > 5 > exe > CvStockBalanceQuery	
2. Select the Join node	

3. On the right of the screen select Join Definition. Notice the cross join (absence of physical join lines)

Important: Cross joins can produce result sets that take-up large amounts of memory that may adversely impact query performance if not filtered appropriately



4. Proceed to preview the join node: Select Join node > Right Click > Data Preview

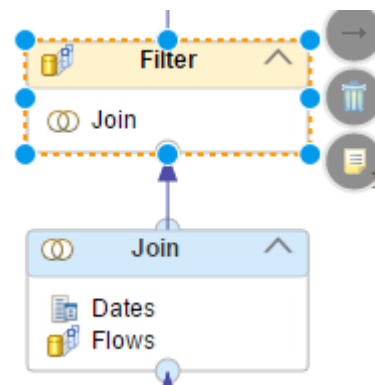


5. The cross join produces 7x7=49 records. End of day stock values can then be calculated using a combination of filters followed by aggregation (for example: end of day balance for 20160503 would be: 25+30-20=35)

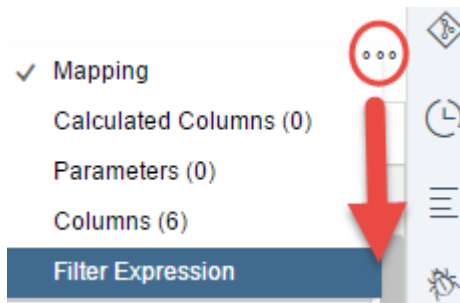
20160503	20160501	Apple	Apple	25	0
20160503	20160502	Apple	Banana	50	0
20160503	20160503	Apple	Apple	30	20
20160503	20160504	Apple	Banana	40	15
20160503	20160505	Apple	Apple	25	15
20160503	20160506	Apple	Apple	20	10
20160503	20160507	Apple	Apple	35	25

6. Your assignment starts:

Apply a filter to strip out the exploded data that is not needed. Above the join node > select the Filter aggregation node

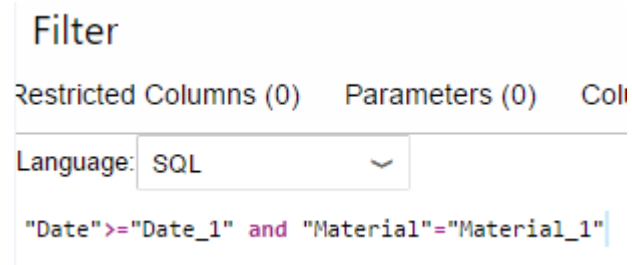


7. Select Filter Expressions

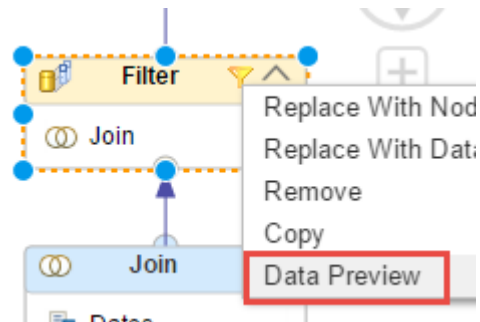


8. Add the following filter:

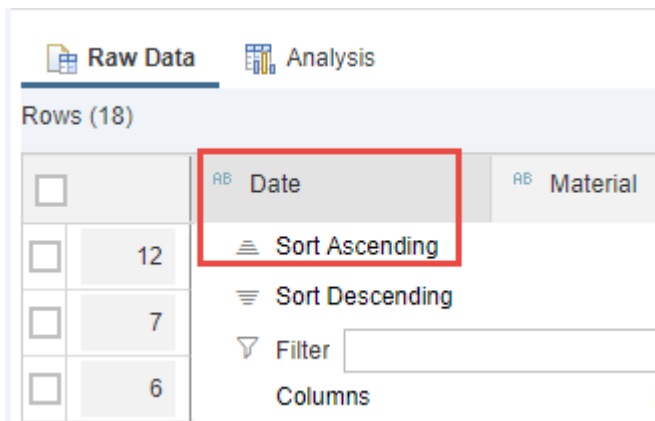
"Date">="Date_1" and
"Material"="Material_1"



9. Save & build the project > then preview the Filter node



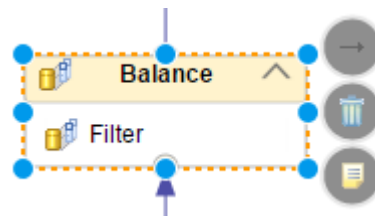
10. When the results appear sort the Date column (Ascending)



11. Notice the entries for 20160504 and specifically (50 Bananas) – this amount is carried over from 20160502. In the next step both records will get aggregated (resulting in 90), subtracting 15 outflows will produce a correct daily balance of 75 for 20160504

AB	Date	AB	D...	AB	M...	12	I...	12	O...
20160501	20160501		Apple				25		0
20160502	20160502		Banana				50		0
20160503	20160501		Apple				25		0
20160503	20160503		Apple				30		20
20160504	20160502		Banana				50		0
20160504	20160504		Banana				40		15

12. Select the Balance node



13. Create a Balance calculated column

Balance

Mapping Calculated Columns (0)

☐ Name

A red arrow points to the '+' icon in the mapping area.

14. Subtract the Outflows from the Inflows

"Inflow"-"Outflow"

Name: * Balance

Data Type: * DECIMAL

Len... 12

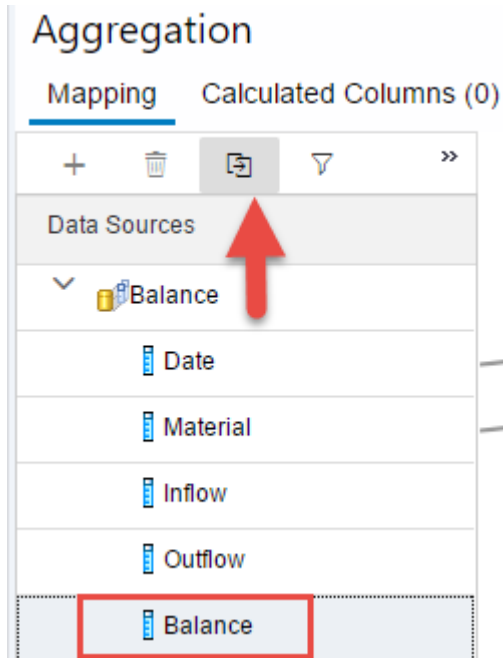
Scale: 2

EXPRESSION

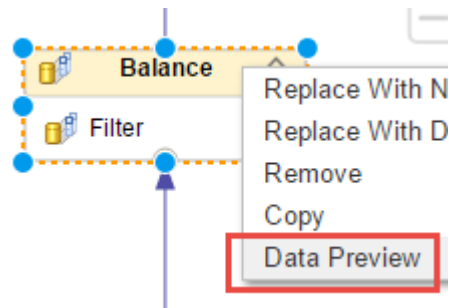
SQL

"Inflow"-"Outflow"

15. Select the topmost Aggregation node and add Balance to the Output



16. Save and build the db project
17. Then Preview the Balance



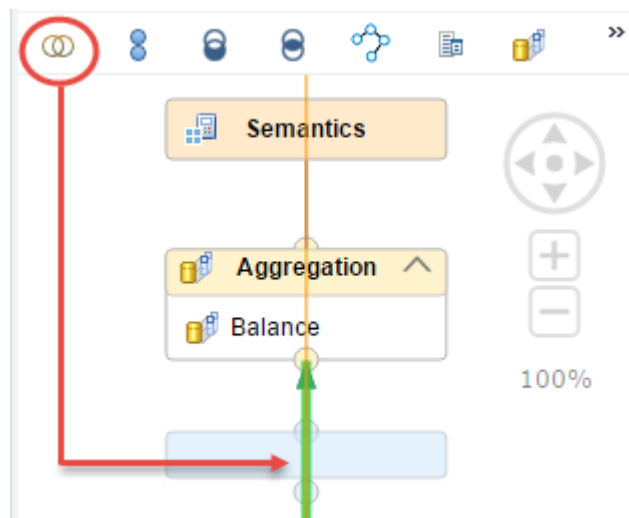
18. The daily stock balance/level is correct.

However, notice the data explosion invalidated the granularity levels of the inflow and outflows. The correct values can be obtained by joining back to the original data-set

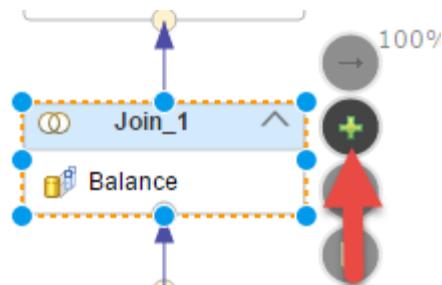
AB	Date	AB	Ma...	12	L...	12	O...	Balan...
20160501	Apple			25		0		25
20160502	Banana			50		0		50
20160503	Apple			55		20		35
20160504	Banana			90		15		75
20160505	Apple			80		35		45
20160506	Apple			100		45		55
20160507	Apple			135		70		65

19. Your next assignment: Fix the daily inflows and outflows. Insert a Join node between the topmost Aggregation node and the Balance node.

Hint: Make sure to drop the join node precisely on the connection line



20. Add another table to the Join node. Click on the + Icon.



21. Search and select the NonCumulativeSales table

Find Data Sources

To search, select an object type and enter an object name

All Types Selected NonCumulative

Results (1)

Typ...	Name	Schema	S...
	ecc.NonCumulativeSales	HBD362_...	

22. Within Join Definition > create a join between Date and Material (LO, N:1)

Hint: Since we are only interested in the inflow and outflow columns the optimize join checkbox can be checked to disregard the concatenated joined columns from being brought into context

The screenshot shows the 'Join Definition' window. On the left, the 'Balance' table is listed with columns: Date (RB), Material (RB), Inflow (12), Outflow (12), and Balance (12). On the right, the 'ecc.NonCum' table is listed with columns: Date (RB), Material (RB), Inflow (12), Outflow (12), and Unit (RB). A join is defined between 'Date' and 'Material' with a cardinality of 'n..1'. The 'Optimize Join Columns' checkbox is checked.

PROPERTIES

Balance Right: ecc.NonCumulative

Left Outer Cardinality: n..1

☐ Dynamic Join

☒ Optimize Join Columns

23. Click on Mapping > Output Columns > Delete both the inflow and outflow columns (if you recall both columns were incorrect, in the next step you will replace them with the recently joined table columns instead)

The screenshot shows the 'Join_1 Mapping' window. Under the 'Output Columns' section, the columns 'Date', 'Material', 'Inflow', 'Outflow', and 'Balance' are listed. The 'Inflow' and 'Outflow' columns are highlighted with a red box. A red arrow points to the delete icon (a trash can) at the top of the list.

Join_1 Mapping

Output Columns

Date

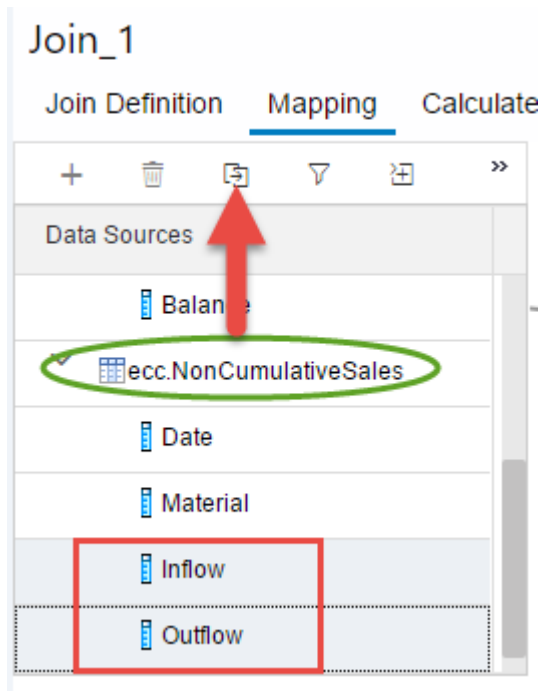
Material

Inflow

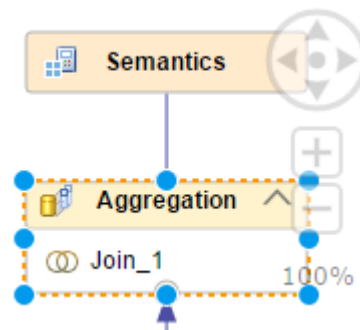
Outflow

Balance

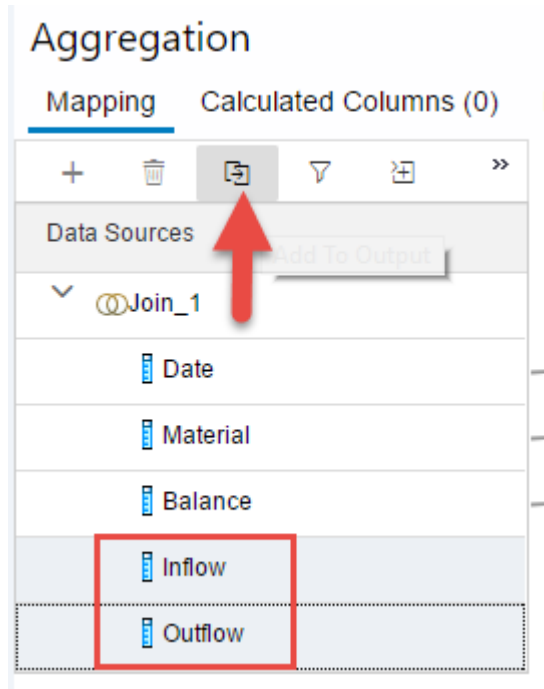
24. Under Data Sources > Add both Inflow and Outflow columns from the previously added NonCumulativeSales table



25. Select the topmost aggregation node



26. If not already added proceed to add both columns to the output of the topmost Aggregation node as well



27. Save & build the project

Open 5.sql > copy SQL #1 > execute

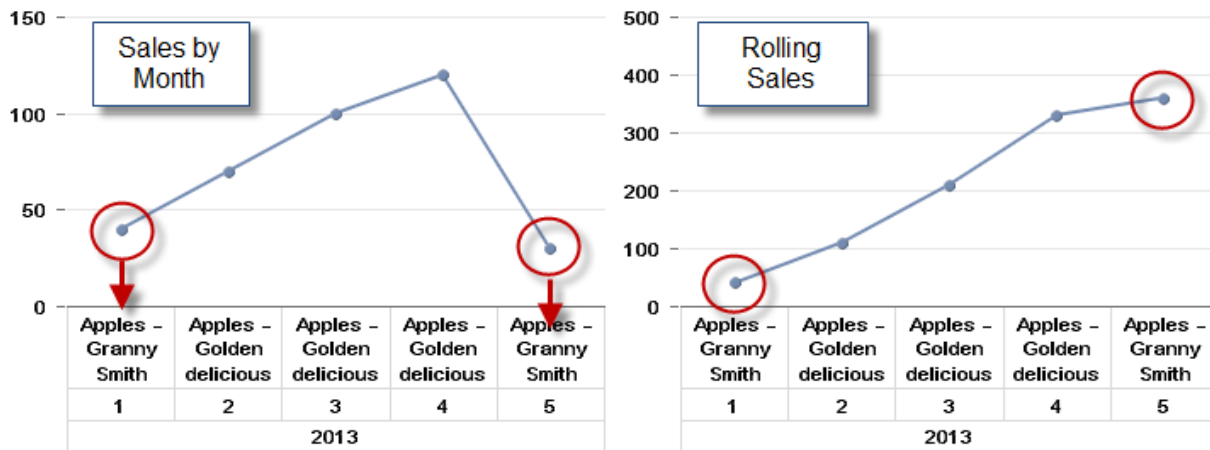
```
--STUDENT SQL
--SQL #1
SELECT "Date", "Material", "Inflow", "Outflow", "Ba
FROM "5.exe::CvStockBalanceQuery" ORDER BY "Date";
```

28. Correct results are shown

AB Date	AB M...	12 In...	12 Out...	12 ...
20160501	Apple	25	0	25
20160502	Banana	50	0	50
20160503	Apple	30	20	35
20160504	Banana	40	15	75
20160505	Apple	25	15	45
20160506	Apple	20	10	55
20160507	Apple	35	25	65

EXERCISE 6 - Cumulative slowly changing dimensions (20 MINUTES, 30 steps)

This exercise showcases sales by month & cumulative slowly changing dimensions. An optical illusion is visible, even though the rolling sales appears to be rising each month the sales by month report shows that sales are declining each time granny-smith (apples) were sold



This solution uses a physical helper table to assist with the monthly rolling totals, in addition a range of advanced modeling features are used such as a star-join, temporal join, dynamic join, transparent filters and keep flags

Explanation	Screenshot
1. Expand models > 6 > exe > open CvSales	
2. Review the sample sales dataset > select the JoinMonths data source join node at the bottom	
3. In the Join Definition Details area to the right > Select the CumulativeSales table > Right Click > Data Preview	

4. Strictly informational:

Notice ...

Total sales for January = \$50

Total sales for February = \$80

Notice ...

Product 1 sales January = \$40

Product 2 sales January = \$10

AB Year	AB Month	AB ProductID	12 Sale
2013	1	1	10
2013	1	1	10
2013	1	1	10
2013	1	1	10
2013	1	2	10
2013	2	1	10
2013	2	1	10
2013	2	1	10
2013	2	1	10
2013	2	1	10
2013	2	1	10
2013	2	1	10
2013	2	1	10
2013	2	2	10

5. Strictly informational

Based on the sample dataset
the expected rolling monthly
sales would be as follows

AB Year	AB Month	12 Sale	12 Rolling
2013	1	50	50
2013	2	80	130
2013	3	110	240
2013	4	130	370
2013	5	40	410

6. Strictly informational:

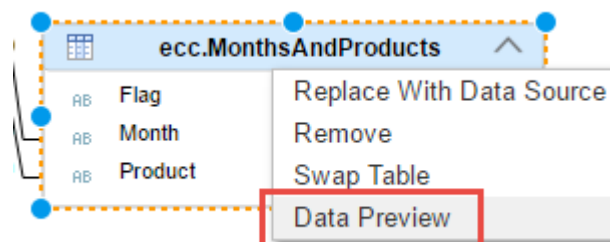
Based on the sample dataset the expected rolling monthly sales by product would be as follows

AB Year	AB Month	AB Name	¹² Rolling
2013	1	Apples	40
2013	1	Banana	10
2013	2	Apples	110
2013	2	Banana	20
2013	3	Apples	210
2013	3	Banana	30
2013	4	Apples	330
2013	4	Banana	40
2013	5	Apples	360
2013	5	Banana	50

7. Review the rolling sales helper table.

Select the MonthsAndProducts table > right click > Data Preview

Notice the concatenated join – Month & Product



8. Sort the Flag column

Informational: Pay attention to the Flag column and take March as an example, the Month column contains Month 1-3 (since the rolling sales for March will include the sales from all 3 months)

The helper table also helps with rolling product sales thus the product column is included in the matrix

AB	Month	AB	Flag	AB	Prod...
	1		1		1
	1		1		2
	1		2		1
	1		2		2
	2		2		1
	2		2		2
1			3		1
1			3		2
2			3		1
2			3		2
3			3		1
3			3		2

9. Strictly informational: Notice the data generator utility that was used to populate the physical helper table

Note: Instead of materializing the matrix into a physical table as in this exercise, the data can dynamically be exploded in memory during runtime as well

10. Review the results of the join between the sales table and the helper table matrix. Right click on JoinMonths > Data Preview

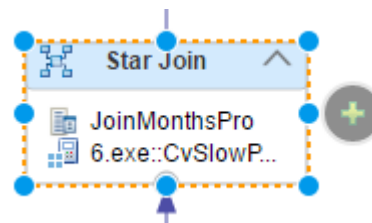
11. Notice February (flag column-red-value -> 2) includes both Month 1 and Month 2 (red Month column)

Subsequently January (flag column 1 orange) includes only Month 1 (yellow month 1 column)

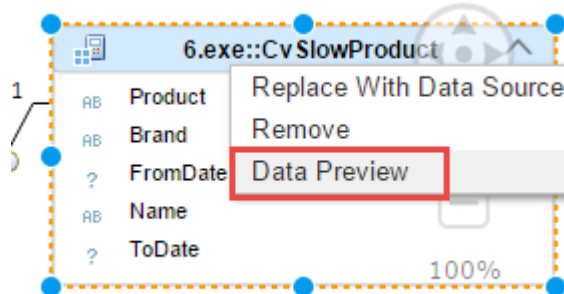
AB	Year	AB	Month	AB	Prod...	12	Sale	Flag
2013	1	1	1				10	1
2013	1	1	1				10	1
2013	1	1	1				10	1
2013	1	1	1				10	1
2013	1	1	2				10	1
2013	1	1	1				10	2
2013	1	1	1				10	2
2013	1	1	1				10	2
2013	1	1	1				10	2
2013	1	2	1				10	2
2013	2	1	1				10	2
2013	2	1	1				10	2
2013	2	1	1				10	2
2013	2	1	1				10	2
2013	2	1	1				10	2
2013	2	1	1				10	2
2013	2	1	1				10	2
2013	2	2	1				10	2
2013	2	2	2				10	2

12. Review the slowly changing product dimensions

Select the StarJoin node



13. Preview the Slow Products Dimensional Calculation Model



14. Notice the validity period for Apples. Second, notice that a different brand of apples was sold between Feb-April

Take note of Granny Smith apples, it was sold in January and May

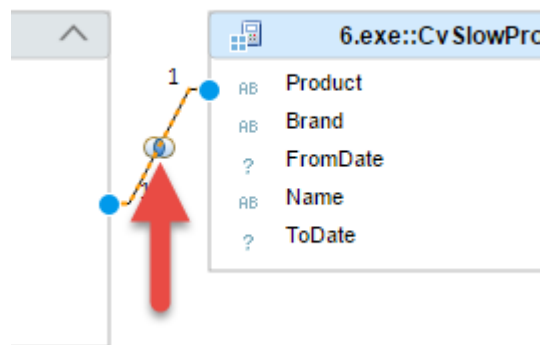
From...	ToDate	AB Brand	AB N...	AB ..
2013-01-01	2013-01-31	Granny Smith	Apples	1
2013-02-01	2013-04-30	Golden Delicio	Apples	1
2013-05-01	2099-01-31	Granny Smith	Apples	1
2013-01-01	2099-01-02	Chiquita	Banana	2

15. Your assignment starts: Model the temporal join:

Join products and transactions, using the product id and the product validity date

Within the Star-Join node > double click on the Join line between the slow-products dimension and the transactional data

Note: The Inner Join is based on Product ID



16. Scroll down to the Temporal Properties and define the properties as follows:

Hint: The Temporal Join (aka between join) is on Order Date

TEMPORAL PROPERTIES

Temporal Column:	OrderDate	From Column:	FromDate
Temporal Condition:	Include Both	To Column:	ToDate

17. Save & build the db project

Open 6.sql, copy and paste SQL #2 and #3 into the SQL console > execute SQL #2

Rolling sales by month is shown

```
--SQL-2
SELECT "Year", "Flag" as "Month", sum("Sale")
FROM "6.exe::CvSales"
WHERE "OrderDate" BETWEEN '20130101' AND '201
```

AB Year	AB Month	12 Rolling	
2013	1	50	
2013	2	130	
2013	3	240	
2013	4	370	
2013	5	410	

18. Execute SQL #3 to see the rolling sales by month and product

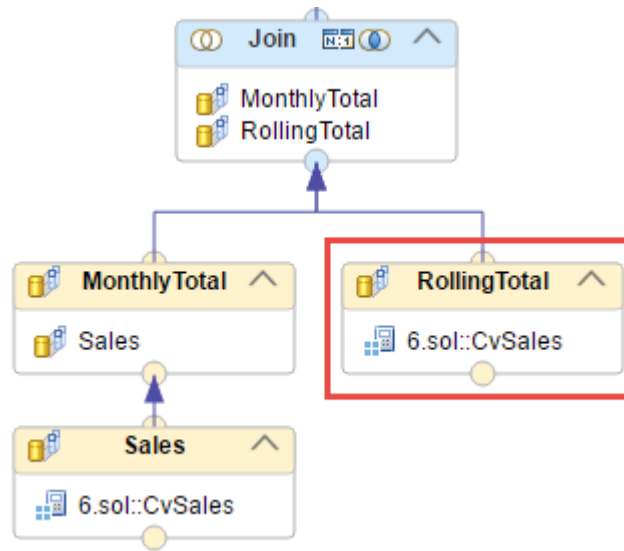
```
--SQL #3
SELECT "Year", "Flag" as "Month", "Name",
sum("Sale") AS "Rolling"
FROM "6.exe::CvSales"
WHERE "OrderDate" BETWEEN '20130101' AND '
```

AB Y..	AB Mo..	AB Name	12 Rolling
2013	1	Apples	40
2013	1	Banana	10
2013	2	Apples	110
2013	2	Banana	20
2013	3	Apples	210
2013	3	Banana	30
2013	4	Apples	330
2013	4	Banana	40
2013	5	Apples	360
2013	5	Banana	50

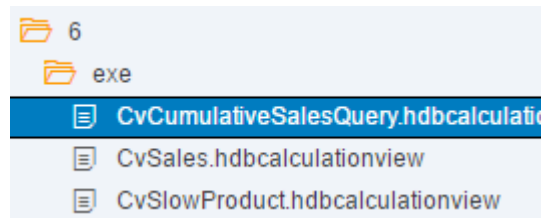
19. Your next assignment starts:

Enhance the model so that the report not only shows rolling sales but also total sales for each individual month

A sneak preview is shown – the solution includes 2 branches (Monthly Total and Rolling Total)

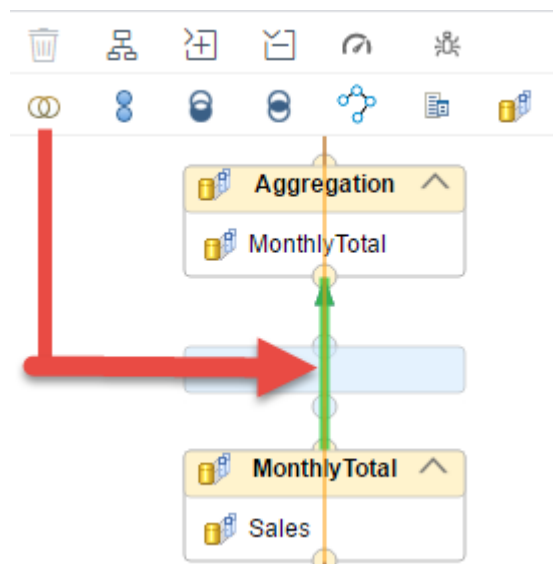


20. Expand models > 6 > exe > open CvCumulativeSalesQuery

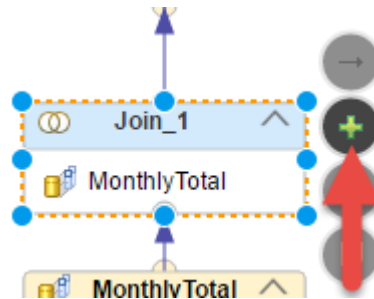


21. Insert a Join node between the Aggregation node and the Monthly Total node

Hint: Make sure to drop the Join node precisely on the connection line



22. Click on + icon to add a join partner data source



23. Search and select 6.exe::CvSales within your own schema

Find Data Sources

To search, select an object type and enter an object name

All Types Selected

CvSales

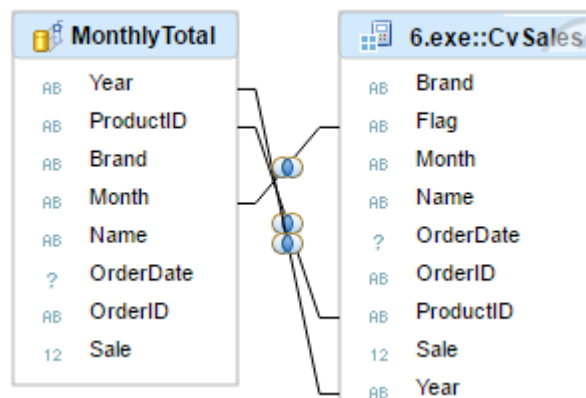
Results (8)

Type...	Name	Schema	Database
	6.exe::CvSales	HBD362_XXX_1	M45

24. Model the following concatenated inner Join:

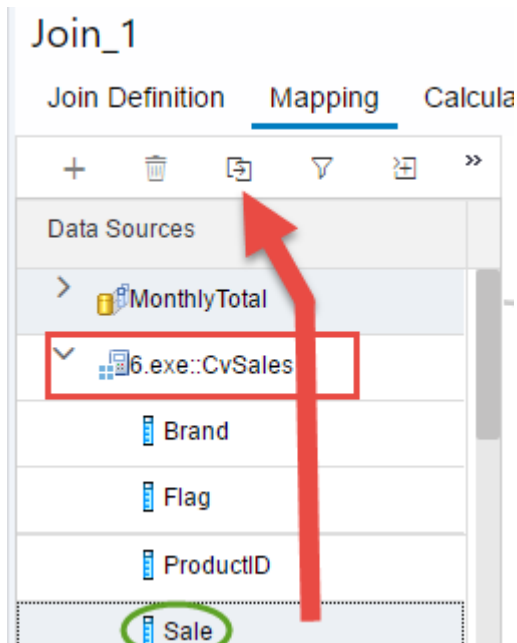
Year -> Year
Product ID -> Product ID
Month -> **Flag**

Important: Set the cardinality (N:1) to enable pruning

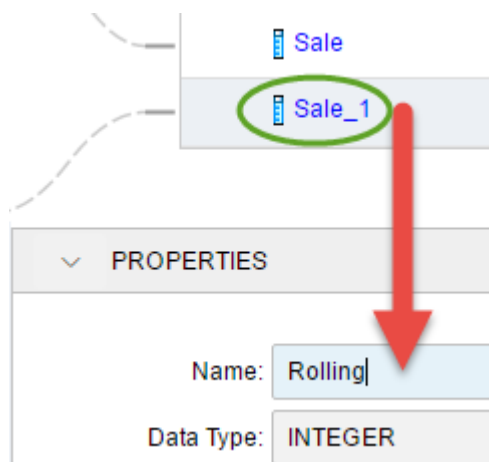


25. Select Mapping and add only the column Sale from the recently added CvSales to the output

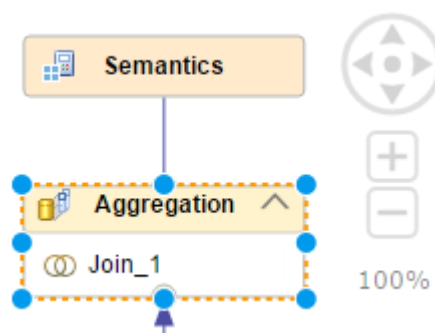
Hint: Since a column called Sale already exists the Sale column that you added are named Sale_1



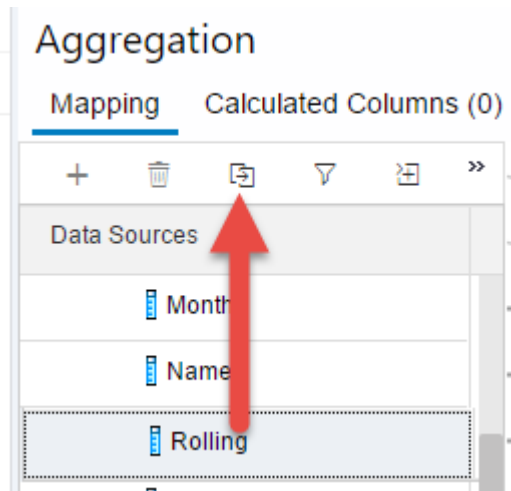
26. Within Mapping > Output > select Sale_1 > in the properties rename to Rolling



27. Add the Rolling column to the output of the Aggregation node



28. Select Rolling > click the Add to Output icon.



29. Save & build the db project.

Copy and execute SQL #4

Note: Even though the rolling sales increase each month, the sales for May is significantly down

```
--SQL #4
SELECT "Year", "Month", sum("Sale") AS "Sale"
FROM "6.exe::CvCumulativeSalesQuery"
WHERE "OrderDate" BETWEEN '20130101' AND '20130501'
GROUP BY "Year", "Month";
```

AB Year	AB Month	12 S...	12 Rolling
2013	1	50	50
2013	2	80	130
2013	3	110	240
2013	4	130	370
2013	5	40	410

30. Drill down into the data,
execute SQL #5 to include
product name

Note: Apple sales are lower in
January and May

```
SELECT "Year", "Month", "Name", sum("Sale") AS
FROM "6.exe::CvCumulativeSalesQuery"
WHERE "OrderDate" BETWEEN '20130101' AND '2013
GROUP BY "Year", "Month", "Name";
```

AB ...	AB ...	AB Na...	12 Sa...	12 Rolling
2013	1	Apples	40	40
2013	1	Banana	10	10
2013	2	Apples	70	110
2013	2	Banana	10	20
2013	3	Apples	100	210
2013	3	Banana	10	30
2013	4	Apples	120	330
2013	4	Banana	10	40
2013	5	Apples	30	360
2013	5	Banana	10	50

31. Drill down into the data even further, execute SQL #6 to include the product band name

Note: Every time Granny Smith Apples were sold the monthly sales were down.

--SQL #6

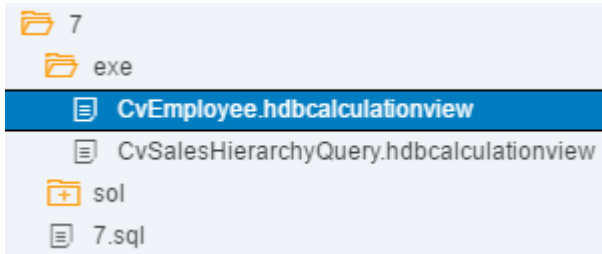
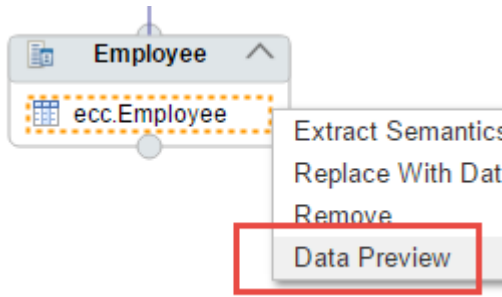
```
SELECT "Year", "Month", "Name", "Brand", sum("Sale") /
FROM "6.exe::CvCumulativeSalesQuery"
WHERE "OrderDate" BETWEEN '20130101' AND '20131231'
GROUP BY "Year", "Month", "Name", "Brand";
```

AB ...	AB ...	AB ...	AB Brand	12 S...	12 Rolli
2013	1	Apple:	Granny Smith	40	40
2013	1	Banar	Chiquita	10	10
2013	2	Apple:	Golden Deliciol	70	110
2013	2	Banar	Chiquita	10	20
2013	3	Apple:	Golden Deliciol	100	210
2013	3	Banar	Chiquita	10	30
2013	4	Apple:	Golden Deliciol	120	330
2013	4	Banar	Chiquita	10	40
2013	5	Apple:	Granny Smith	30	360
2013	5	Banar	Chiquita	10	50

EXERCISE 7 - SQL hierarchies and SQL analytical privileges (20 MINUTES, 30 steps)

This exercise consists of 2 parts; SQL hierarchies and SQL analytical privileges. Your assignment is to create an employee parent child sales hierarchy and to calculate margin at each level of the hierarchy. In addition, you will learn how to incorporate design time roles and how to test analytical privileges

Employee	Revenue	Cost	Margin
[-] Pointed-Haired	30,960	5,046.00	83.70
[-] Alice	20,370	3,463.00	82.99
Loud	12,360	3,057.00	75.26
Wally	7,310	304.00	95.84
[-] Dilbert	10,590	1,583.00	85.05
Dogbert	2,570	314.00	87.78
Ted	8,020	1,269.00	84.17

Explanation	Screenshot
1. Expand models > 7 > exe > open CvEmployee dimensional model	
2. Review the sample Dilbert employee organization dataset > Select the Employee node > right click > Data Preview	

3. Notice the parent child hierarchy (Manager, Employee)

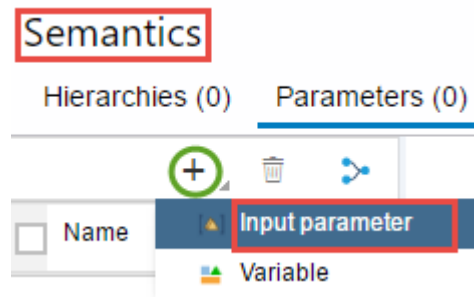
Notice the upcoming organization change – in January 2018 Loud will no longer be reporting to Alice but to a new manager Mordac

Emp...	ValidF...	ValidTo	Man...	NickNa...
Pointed-Hi	2010-01-01	2020-12-31	NULL	Big Boss
Dilbert	2010-01-01	2020-12-31	Pointed-Hi	Dilbert
Dogbert	2010-01-01	2020-12-31	Dilbert	Dog
Ted	2010-01-01	2020-12-31	Dilbert	Engineer
Alice	2010-01-01	2020-12-31	Pointed-Hi	Hairstylist
Wally	2010-01-01	2020-12-31	Alice	Old Timer
Loud	2010-01-01	2017-12-31	Alice	Intern
Mordac	2018-01-01	2020-12-31	Pointed-Hi	The Refuser
Loud	2018-01-01	2020-12-31	Mordac	Loud Howard

4. Your assignment starts:

Create an input parameter (current date) that will be used for both the temporal and hierarchy calculations

Semantics > Parameters > click on the + icon > Input Parameter



- 5.

Name: KeyDate
 Label: KeyDate
 Type: Direct
 Semantic: Date
 Type: Date

Name: * KeyDate

Label: KeyDate

Parameter Type: Direct

Semantic Type: Date

Data Type: * DATE

6. Add a default expression value and use the NOW SQL function

DEFAULT VALUE(S)

Type	Value
Expression	NOW()

Annotations: A red arrow points to the '+' icon at the top right. A red box highlights the 'Expression' dropdown. A green circle highlights the copy icon at the bottom right.

7. Your next assignment: Add a temporal filter:

Select the Employee Data Source > Filter Expression

"ValidFrom" <= '\$\$KeyDate\$\$' and '\$\$KeyDate\$\$' <= "ValidTo"

Employee

Mapping Filter Expression

Language: SQL

"ValidFrom" <= '\$\$KeyDate\$\$' and '\$\$KeyDate\$\$' <= "ValidTo"

Annotations: A red box highlights the 'Filter Expression' dropdown. A red arrow points down towards the SQL expression.

8. Your next assignment: Create a Parent Child Hierarchy

Select the top Semantics node > Hierarchies > click the + icon > select Parent Child Hierarchy

Semantics

Columns (6) Hierarchies (0)

Annotations: A green circle highlights the '+' icon. A red box highlights the 'Parent Child Hierarchy' option.

9. Use the following settings:
- Name: OrgHierarchy
- Label: OrgHierarchy

Click on the + Icon

Child: EmployeeID

Parent: ManagerID

GENERAL

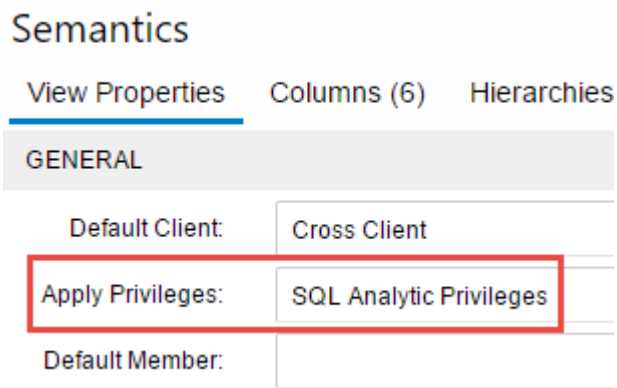
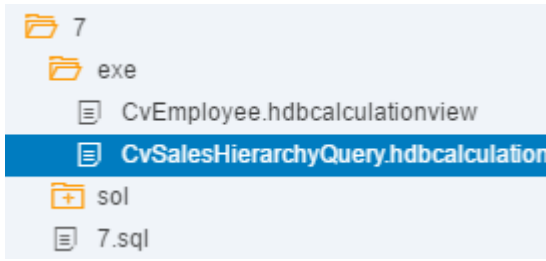
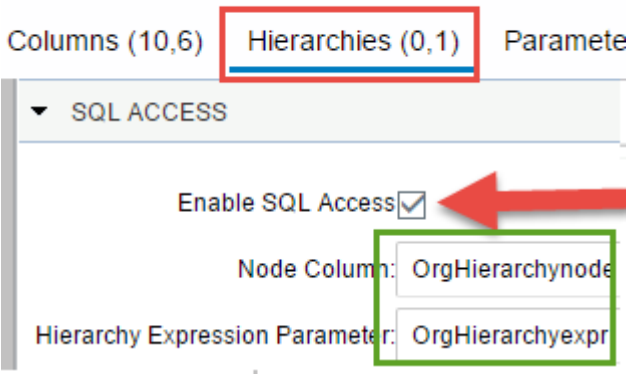
Name: * OrgHierarchy

Label: OrgHierarchy

NODES

Child	Parent	Step
EmployeeID	ManagerID	

Annotations: A red arrow points to the '+' icon at the top right. A red box highlights the 'EmployeeID' and 'ManagerID' entries in the table.

<p>10. Apply SQL Analytical Privileges</p> <p>Semantics > View Properties > General > Apply Privileges > SQL Analytical Privileges</p>	
<p>11. Save & build the db project</p>	<p>Build of /HBD362/db completed successfully</p>
<p>12. Your next assignment: Enable SQL hierarchy access for the calculation model (aka cube) that uses the dimensional employee hierarchy model</p> <p>7 > exe > Open Cv Sales Hierarchy Query</p>	
<p>13. Semantics > Hierarchies > scroll down to SQL Access > click the checkbox to Enable SQL Access > notice the default values</p> <p>Important: Notice the additional OrgHierarchynode column, this column can be used via SQL group by statements and/or within SQL Analytical Privileges to filter the data (which you will do later)</p>	

14. Semantics > View Properties > Enable SQL Analytical Privileges for this CUBE model

Ensure to enabled hierarchy access for SQL

Semantics

View Properties Columns (10,6) Hierarchies (0)

Apply Privileges:

SQL Analytic Privileges



☒ Enable Hierarchies for SQL access

15. Confirm Yes to enable default names for Hierarchy SQL Access

Confirm

This operation will set default names for the model and for the hierarchies. Do you want to continue?

Yes

No

16. Save & build the db project

Build of /HBD362/db completed successfully

17. Open 7.sql and copy both SQL #1 and #2 into the SQL Console > Execute SQL #1

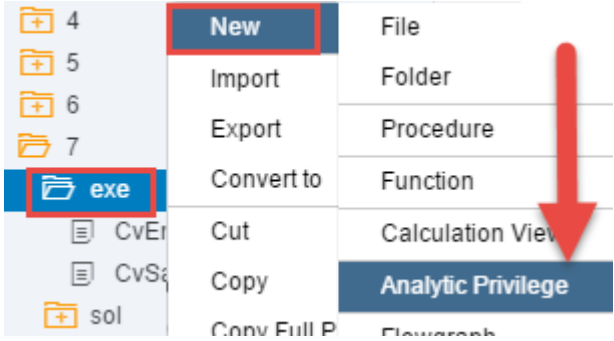
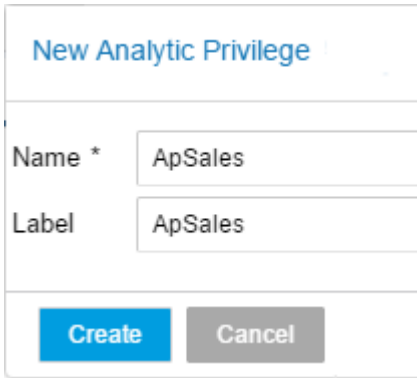
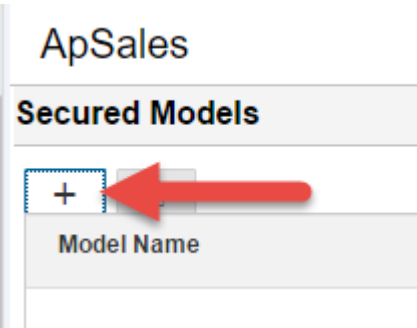
Note: At this point you will not be able to query the model; SQL analytical privileges require structured privileges which we will create next

```
--SQL #1
SELECT "OrgHierarchynode", sum("Revenue") AS "Revenue"
FROM "7.exe::CvSalesHierarchyQuery"
GROUP BY "OrgHierarchynode" ORDER BY "OrgHierarchynode"
```

SQL Console

Could not execute 'SELECT "OrgHierarchynode", sum("Revenue") AS "Revenue", sum("Cost"), sum("Margin") AS "Margin" ...'
Error: (dberror) insufficient privilege



<p>18. Create a new Analytic Privilege</p> <p>7 > exe > right click > New > Analytical Privilege</p>	
<p>19. Name: ApSales > Create</p>	
<p>20. Click on the + icon to add the Sales Hierarchy cube model</p>	



21. Search and select
CvSalesHierarchy in your exe
folder

Find Data Sources

To search, select an object type and enter an object name


All Types Selected CvSalesHier

Results (2)

Typ...	Name	Schema
	7.exe::CvSalesHierarchy..	HBD362_...
	7.sol::CvSalesHierarchy...	HBD362_...



22. Change the privilege type to
SQL Expression > confirm OK

Rename

Attribute 

PRIVILEGE VALIDITY

ASSOCIATED ATTRIBUTE RES

Attribute /value

Attribute
SQL Expressio
Dynamic

23. Enter the following SQL
Expression:
- "OrgHierarchynode" = 'Alice'
- Save your work!

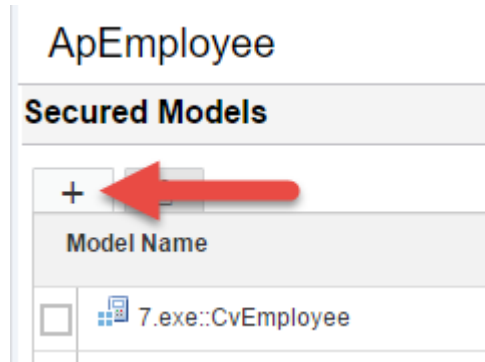
SQL Expression

"OrgHierarchynode" = 'Alice'

24. Repeat the steps and create another SQL analytical privilege (ApEmployee) > add CvEmployee as a secured model

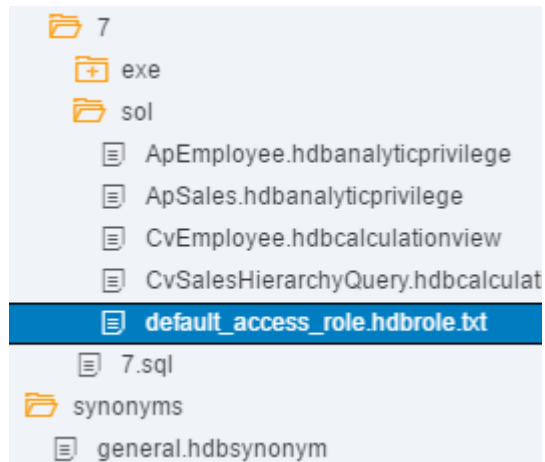
Hint: The OrgHierarchynode filter column exists only on the parent Sales model cube, the Employee dimensional model requires its own independent privilege

Save your work!



25. Your next assignment: create a role that uses the analytical privileges.

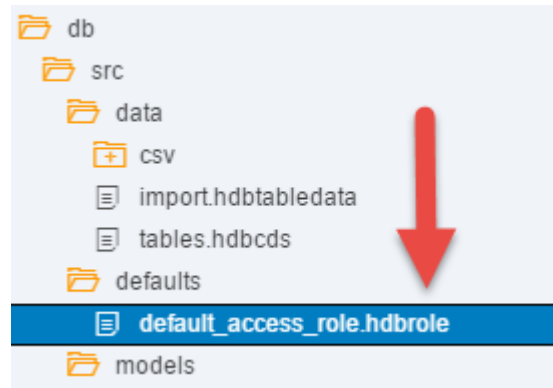
Expand 7 > sol > open default_access_role



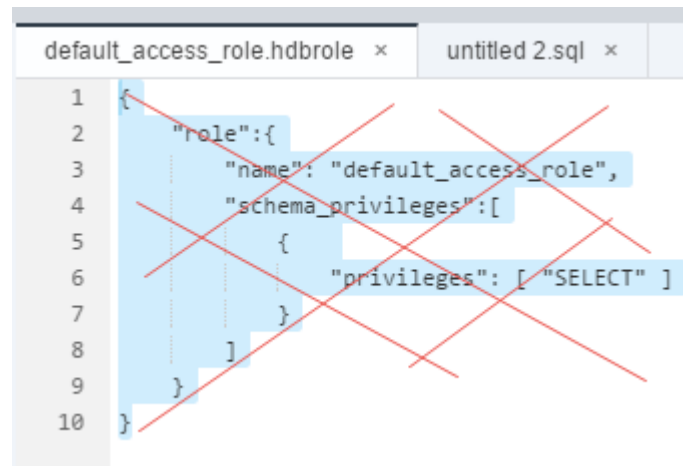
26. Copy all the text > CNTRL+A > CNTRL + C



27. Open db > src > defaults > default_access_role.hdbrole >



28. Delete all the text and override / paste it with the text you copied previously



29. Notice the parent sales hierarchy node is exposed with SQL view access

Notice both analytical privileges created previously are included in this role

Hint: We added both the object and analytical privileges to the default role, this is a convenience way for you as a developer to test analytical privileges

```
"object_privileges": [
  {
    "name": "7.exe::CvSalesHierarchyQuery",
    "type": "VIEW",
    "privileges": [ "SELECT" ]
  }
],

"schema_analytic_privileges": [
  {
    "privileges": [ "7.exe::ApSales" ]
  },
  {
    "privileges": [ "7.exe::ApEmployee" ]
  }
]
```

30. Save & build the db project

31. Execute SQL #1

Notice that Alice is the only employee returned since your user is restricted to Alice via the analytical privilege

Hint: Revenue amount includes combined sales for herself + her direct reports (Loud + Wally)

untitled 2.sql ×			
1	--SQL #1		
2	SELECT "OrgHierarchynode", sum("Revenue") AS "R		
3	FROM "7.exe::CvSalesHierarchyQuery"		
4	GROUP BY "OrgHierarchynode" ORDER BY "OrgHierar		
5			
AB	OrgHierarchynode	12 Reven...	12 Cost
	Alice	20370	3463
			82.99

32. Execute SQL #2

Notice the above numbers are correct, the query returned individual sales for Alice + her direct reports

AB	Emp...	12 Revenue	12 Cost	12 Margin
	Loud	12360	3057	75.26
	Wally	7310	304	95.84
	Alice	700	102	85.42

www.sap.com/contactsap

© 2017 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.

The information contained herein may be changed without prior notice. 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 are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP 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, and they should not be relied upon in making purchasing decisions.

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. All other product and service names mentioned are the trademarks of their respective companies. See <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.