

INFS3200 Advanced Database Systems

Prac 2: Data Warehousing (5%)

*Semester 2, 2022***Due:** Week 8 – Friday September 16th at 4pm**Submission:** via Blackboard → [Assessment](#) > [Practicals](#) > [Practical Two](#) > [Submission](#)

Introduction

Learning objectives

- Learn how to create a cube and its dependent components in Oracle OLAP. The tasks include:
 - To identify and build the dimensions,
 - To define measures (both stored and calculated facts).
- Learn how to map an OLAP model to source data and load data into the dimensions.
- Learn how to view an OLAP dataset. The tasks include:
 - To query through hierarchies of dimensions,
 - To perform DW operations such as *roll-up*, *drill-down*, and *pivot*.
- Understand how data is organized and stored in dimensions and data cubes,
- Learn how to query a data cube using SQL queries.
- Understand how materialized views are used in cube query optimization.

Marking Scheme

- **1 mark:** Complete the cube creation in Task 1.
- **1 mark:** Maintain the cube successfully in Task 2.
- **2 marks:** Complete Task 3, each bullet point is worth 1 mark.
- **1 mark:** Complete Task 4.

Screenshot your results for each task and compile them into a document with appropriate explanations. You should screenshot both the data cubes you have created, as well as the result of your OLAP operations. Keep your explanations terse but make sure you contain all necessary information. Make sure your screenshots contain **your student ID** (since your student ID will be included in the name of the users you created) as proof of originality. Export your report into a **PDF** document. **If you just submit a PDF report**, name it like: [JohnnyExample_s1234567890_prac2.pdf](#). Submitting scripts is **optional**. Copy your scripts into a subdirectory named [scripts](#) and pack your scripts and report into a zip file. Your zip file should be named appropriately, eg [JohnnyExample_s1234567890_prac2.zip](#). Please format your document and code nicely to assist the tutor's marking process. A poorly formatted document may receive a reduced mark. **Due 4pm on Friday the 16th of September, 2022.**

Late Penalties (from ECP): “Where an assessment item is submitted after the deadline, without an approved extension, a late penalty will apply. The late penalty shall be 10% of the maximum possible mark for the assessment item will be deducted per calendar day (or part thereof), up to a maximum of seven (7) days. After seven days, no marks will be awarded for the item. A day is considered to be a 24 hour block from the assessment item due time. Negative marks will not be awarded.”

Tips for Completing this Practical

This practical sheet is quite long and will require you to use software which you are not yet familiar with (the Oracle OLAP engine). Due to the way the lab machines are configured (including with the RDP connection), please be aware that your work **may be lost after you disconnect from the lab machine**. There are a few options to deal with this. Firstly, it is recommended that you take local backups of your work as you go (saving the scripts so you can execute them rapidly). You may also wish to set aside time to complete the prac in a single session to avoid disruption. Secondly, the same troubleshooting tips from Prac 1 may assist you here. If you cannot connect to the database, **make sure the oracle services are running**. Please see the prac sheet 1, the discussion board, and the RDP connection guide for reference.

Preparation: Data Warehouse Setup

System overview

Oracle OLAP: Oracle OLAP is a multidimensional analytics engine embedded in the Oracle database system. It supports online analytics based on data warehousing techniques. In Oracle OLAP, a *cube* provides a convenient way of collecting stored and calculated measures with similar characteristics, including dimensionality, aggregation rules and so on. A particular analytic workspace (AW) may contain more than one cube, and each cube may describe a different dimensional shape. Multiple cubes in the same AW may share one or more dimensions (Fact constellation; see the *week 5 lecture notes*).

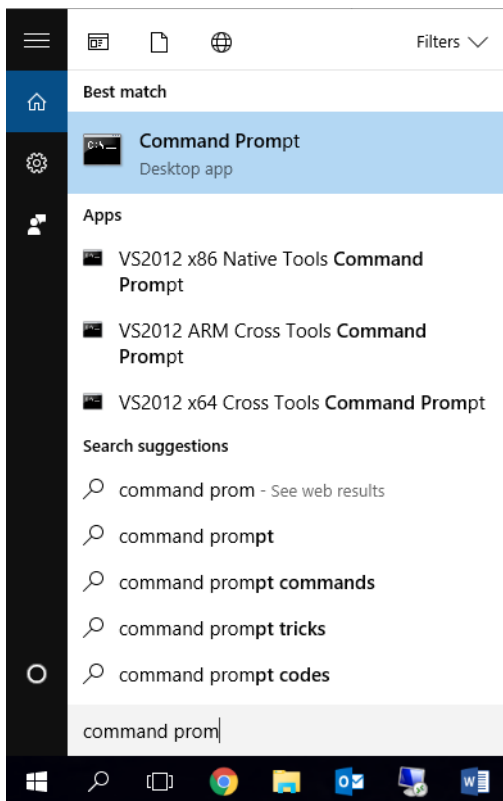
Analytic Workspace Manager (AWM): In Oracle OLAP, **AWM** is an easy-to-use GUI tool for creating, developing, and managing multidimensional data in an Oracle data warehouse. Refer to: <https://www.oracle.com/database/technologies/olap-12101b-readme.html>

Description of the Data Warehouse

You are given a Data Warehouse, namely **OLAPTRAIN** which has a **star schema**. **OLAPTRAIN** was sourced from a transactional database system, which contains data for a fictional electronics store. The following descriptions outline the content of the tables (including *four dimension tables* and *one fact table*), intended for analysis. These tables have already gone through the ETL (Extraction, Transformation, and Loading of heterogeneous data) process:

Table	Description
CHANNELS	This table contains distribution channels for customers' purchases.
CUSTOMERS	This shows who purchased products, and where products were sold for Geographical dimensions of locations.
PRODUCTS	This table contains product categories sold by the company.
TIMES	This table contains time periods when products were sold.
SALES_FACT	This table stores purchases in dollars, quantity and unit price, by the channel of distribution, product item, day, and customer.

2.3 Data import



Download the `P2.zip` package from the LMS and extract it to a local directory, for example “`C:\app\P2\`”. The given package contains the *AWM software*, the *OLAPTRAIN schema*, *cube templates* and *query scripts* used in the following tasks. **Note:** The location of the directory here needs to be on the local disk, C:, or oracle will not have the permissions required to modify the data.

Before analyzing the data, we need to import the OLAPTRAIN schema into the Oracle database. Open a **Command Prompt** by searching ‘*command prompt*’ in the search window. Please **run it as administrator** to avoid privilege issues. Then, complete the following steps.

Enter the directory that has the installation files, in my case it is “`C:\app\P2\olaptrain_install`”

```
cd YOUR_P2_FOLDER\olaptrain_install
```

Login to SQL*Plus

```
> sqlplus  
sys as sysdba
```

Enter password: `Password1`

NOTE: If the above password fails, try `Password1!`

Set a system parameter to avoid a future error when creating users

```
SQL> alter session set "_ORACLE_SCRIPT"=true;
```

Run the OLAPTRAIN installation script

```
SQL> @install_olaptrain_student
```

Enter the install directory and password

Directory: `YOUR_P2_FOLDER\olaptrain_install`

Password: `w`

StudentID: `S1234567`

Note that the directory should be the same as above and you can choose your own password for user “`OLAPTRAIN_S1234567`” (Please **DO NOT** include any special characters in your password – we recommend just using something simple like `w`), which will be the main user throughout this practical. Please change “`S1234567`” to your student ID and make sure “`S`” is upper case, this is crucial.

The process is shown below [continued next page]:

```
Administrator: Command Prompt - sqlplus
C:\Windows\system32>cd C:\app\P2\olaptrain_install
C:\app\P2\olaptrain_install>sqlplus
SQL*Plus: Release 12.1.0.2.0 Production on Wed Aug 24 14:43:28 2022
Copyright (c) 1982, 2014, Oracle. All rights reserved.
Enter user-name: sys as sysdba
Enter password:
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options
SQL> alter session set "_ORACLE_SCRIPT"=true;
Session altered.
SQL> @install_olaptrain_student
This procedure installs the OLAPTRAIN schema for the Oracle OLAP 12c
training class, samples and Oracle by Examples.
1. You should be logged on as a DBA to execute this procedure.
2. Ensure any existing OLAPTRAIN user sessions are disconnected before proceeding with the installation (this install will
recreate the user if it exists)
Specify file system directory containing this installation program.
Default:
c:\app\P2\olaptrain_install
Directory: C:\app\P2\olaptrain_install
Specify a password for the OLAPTRAIN user.
Password:
Please type in your student ID, e.g., S1234567, and your final username will be OLAPTRAIN_S1234567
PLEASE MAKE SURE ID IS UPPER CASE.
Student ID: S1234567
Begin installation
```

Note: The default path is **wrong**; the "C" should be capitalized. Please type in the correct path based on where you put the P2 files on your own system

2.4 Troubleshooting:

A **successful** installation should end up with **6 errors**, shown as follows:

```
Failing sql is:
BEGIN
  SYS.DBMS_SNAPSHOT_UTL.SYNC_UP_LOG('OLAPTRAIN','CUSTOMERS');
END;

Processing object type SCHEMA_EXPORT/TABLE/MATERIALIZED_VIEW_LOG
Processing object type SCHEMA_EXPORT/DIMENSION
ORA-39083: Object type DIMENSION:"OLAPTRAIN_S1234567"."CHANNELS" failed to create with error:
ORA-00942: table or view does not exist

Failing sql is:
CREATE DIMENSION "OLAPTRAIN_S1234567"."CHANNELS" LEVEL "CHANNEL" IS ("OLAPTRAIN"."CHANNELS"."CHANNEL_KEY") LEVEL "CLASS"
IS ("OLAPTRAIN"."CHANNELS"."CLASS_KEY") HIERARCHY "SALES_CHANNEL" ("CHANNEL" CHILD OF "CLASS") ATTRIBUTE "CHANNEL" LEV
EL "CHANNEL" DETERMINES "OLAPTRAIN"."CHANNELS"."CHANNEL_NAME" ATTRIBUTE "CHANNEL" LEVEL "CHANNEL" DETERMINES "OLAPTRAIN"
"."CHANNELS"."CHANNEL_TYPE" ATTRIBUTE "CLASS" LEVEL "CLASS" DETERMINES "OLAPTRAIN"."CHANNELS"."CLASS_NAME"

Processing object type SCHEMA_EXPORT/POST_SCHEMA/PROCACT_SCHEMA
Job "OLAPTRAIN_S1234567"."SYS_IMPORT_FULL_01" completed with 6 error(s) at Fri Apr 17 21:55:16 2020 elapsed 0 00:00:42

... dropping OLAPTRAIN_INSTALL directory
Installation complete.
SQL>
```

However, if the installation ends quickly with the following message, it is usually caused by specifying an incorrect directory. **Please check your path and try again.**

```
please type in your student ID, e.g.:S1234567, and your final username will be OLAPTRAIN_S1234567  
PLEASE MAKE SURE 'S' IS UPPER CASE.  
Student ID: S1234567
```

```
Begin installation
```

```
... creating OLAPTRAIN user  
... creating OLAPTRAIN_INSTALL directory  
... importing dump file into OLAPTRAIN_S1234567 schema.
```

```
Import: Release 12.2.0.1.0 - Production on Fri Apr 17 21:44:31 2020
```

```
Copyright (c) 1982, 2017, Oracle and/or its affiliates. All rights reserved.
```

```
Connected to: Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
```

```
ORA-39002: invalid operation
```

```
ORA-39070: Unable to open the log file.
```

```
ORA-29283: invalid file operation
```

```
ORA-29283: invalid file operation
```

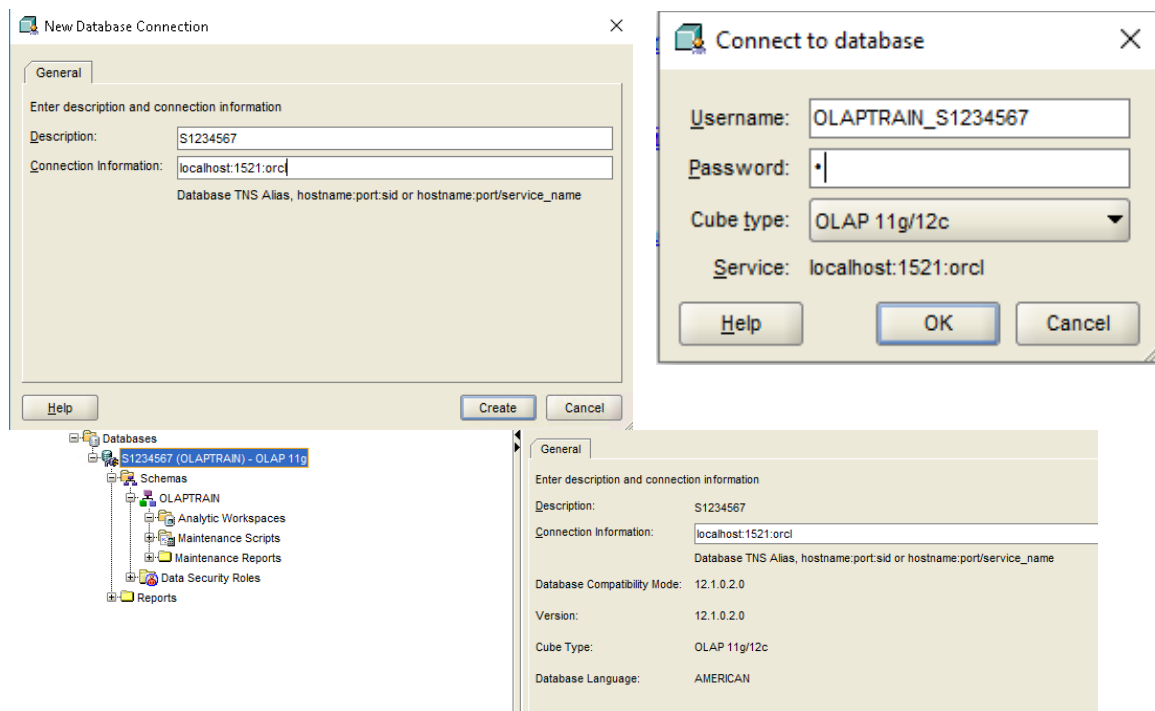
```
... dropping OLAPTRAIN_INSTALL directory
```

```
Installation complete.
```

```
SQL>
```

Connect to data warehouse using AWM

Click the “**awm.bat**” in the “*awm122010_Standalone*” folder to open the AWM (if there is no response after a click, please check if you have **Java JDK** installed in your system, Java 8 recommended). Create a new connection to the Oracle database and set the **Description** as your student ID and the **Connection Information** as ‘localhost:1521:orcl’. Click the connection and log in using the username “**OLAPTRAIN_S1234567**” and the password you set.



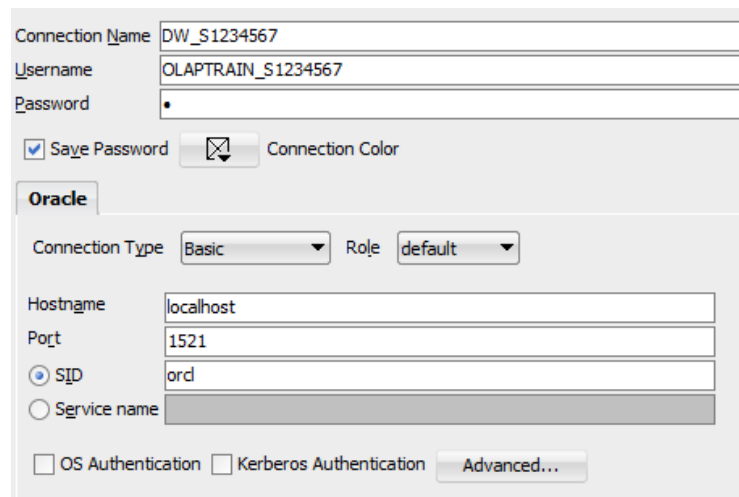
Troubleshooting

- If the **awm.bat** file does not run correctly, try launching AWM by double clicking the **awm12.2.0.1.0.jar** file.

Part 1: Create Logical Data Model

1. Understanding the data

Before designing the data model, it is highly suggested to first understand the *OLAPTRAIN* schema. Use **SQL developer** to connect to the database and check the following tables: *channels*, *customers*, *products*, *times* and *sales_fact*. The connection name should include your student ID, like “DW_S1234567”.



After examining the OLAPTRAIN schema, we need to identify the dimensions, hierarchies and attributes of our data model. In addition, we define various measures based on business interests. **If you get an error** when you try to connect to your data warehouse, make sure the oracle services are running, make sure the SID above is “orcl”, and make sure the username and password is correct.

(1) Identifying Dimensions

Using the source data tables as the primary input, the following dimensions are identified as requirements for the OLAP data model:

- Channel
- Geography
- Product
- Time

Each of the dimensions contains a hierarchical structure. For example, in the CHANNELS table, shown in the following figure, we can identify four hierarchies:

1. The actual channels (channel_name/channel_key)
2. The channel type (channel_type)
3. The channel class (class_name/class_key)
4. \emptyset (not chosen in group-by query, refer to *question 2 in tutorial 5*). Here, we name it all_channel).

	CHANNEL_KEY	CHANNEL_NAME	CHANNEL_TYPE	CLASS_KEY	CLASS_NAME
1	18	Catalog	Mail Order Catalog	-4	Indirect
2	19	New York Retail	Boutique	-3	Direct
3	20	Lisbon Retail	Boutique	-3	Direct
4	21	San Francisco Retail	Mall Store	-3	Direct
5	22	Internet	Web Store	-4	Indirect
6	23	London Retail	Boutique	-3	Direct
7	24	Shanghai Retail	Airport Shop	-3	Direct
8	25	Los Angeles Retail	Boutique	-3	Direct
9	26	Paris Retail	Boutique	-3	Direct

(2) Identifying Measures

The *measures* are defined based on common business interests, each of which is equivalent to an SQL aggregation query. The measures include both stored and calculated measures. Stored measures are facts acquired from the fact table directly, while the calculated measures require complicated calculations over one or multiple facts. In this dataset, we focus on the following measures:

Stored Measures

- Sales
- Quantity

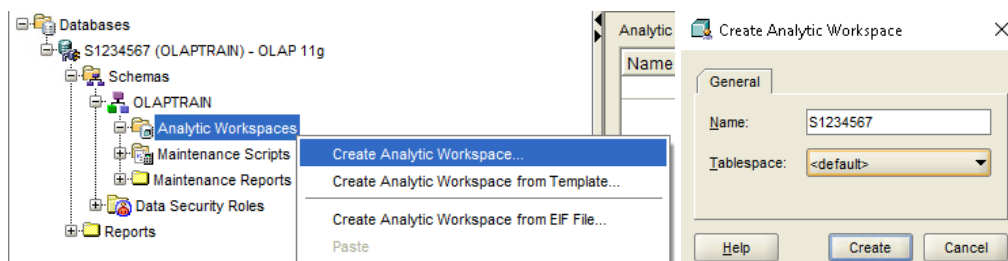
Calculated Measures

- Sales Year-to-Date
- Sales Year-to-Date Prior Year

The measures will be defined during the creation of cubes, which will be introduced later.

2. Create analytic workspace

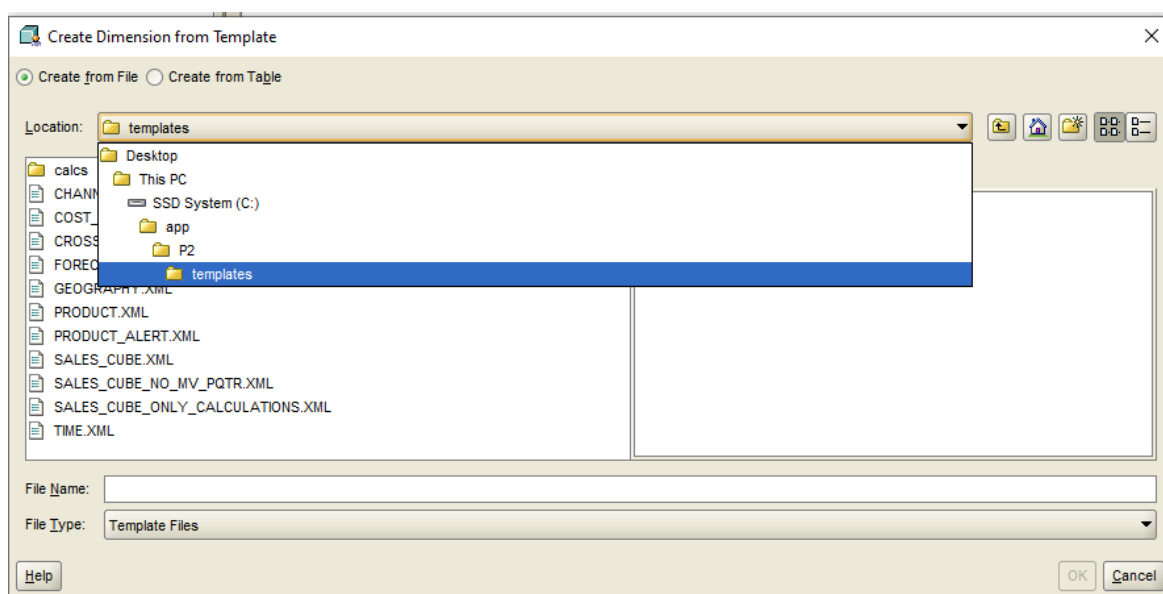
Go back to the Analytic Workspace Manager. Next, right-click **Analytic Workspaces** and select **Create Analytic Workspace** to create a new analytic workspace under the name of your student ID.



3. Create dimensions using templates

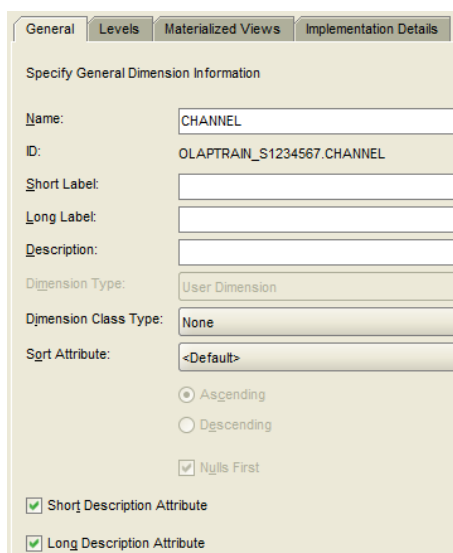
We provide *templates* for all four dimensions so that you do not need to define them manually. The template feature in Analytic Workspace Manager saves the definition of the OLAP data objects as an XML file. Using a saved template, you can create a new analytic workspace, dimension, cube, and measure exactly like an existing object, with or without mappings. Templates do not include the data, only the definition of the object.

In order to import a template, right-click the **Dimensions** folder, then select **Create Dimension from Template**. The templates are stored in the *templates* folder in the extracted folder. **Remember, the path depends on where you stored the data you downloaded from the LMS!**



Import *Channel*, *Geography*, *Product* and *Time* dimensions in the same way. Check the settings of these dimensions and make sure all members are mapped to the data source (no modification is needed). For example, in the *Channel* dimension, the template contains the following settings:

Click the **Dimensions** folder and select *Channel*. In the General tab, we can see the dimension name (Channel) and the dimension type (User Dimension).



In the Levels tab, we can see three levels are defined, which correspond to three of the four possible dimensions mentioned above (channel type not included):

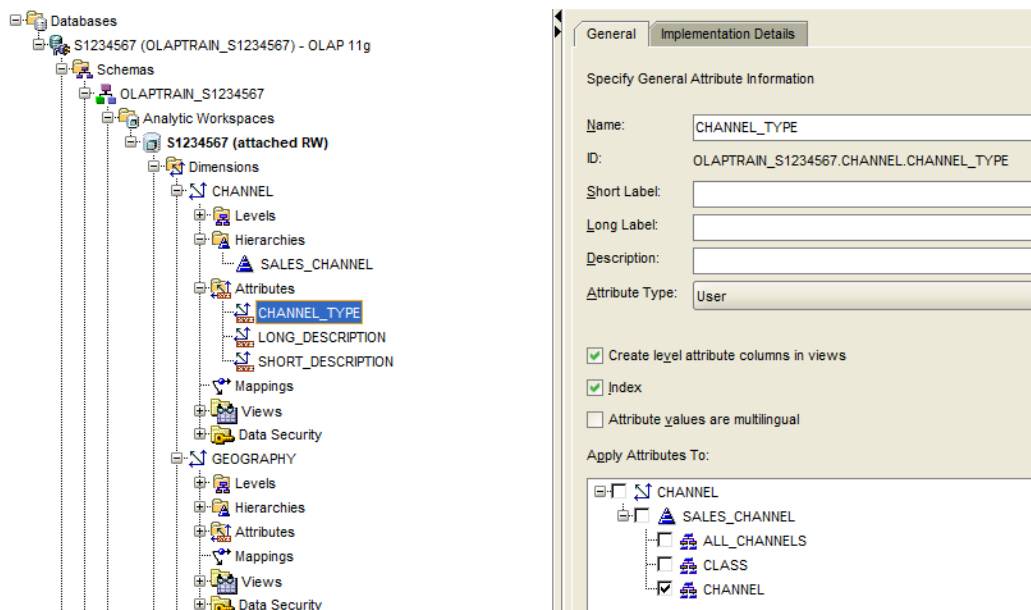
- **ALL_CHANNELS** = \emptyset
- **CLASS** = channel class
- **CHANNEL** = the actual channels

General Levels Materialized Views Implementation Details			
Name	Long Label	Short Label	Description
ALL_CHANNELS			
CLASS			
CHANNEL			

Under the Hierarchies in *Channel* dimension, we can see the **SALES_CHANNEL** hierarchy. These three levels are ordered in the following way, which is consistent with our understanding of the data:

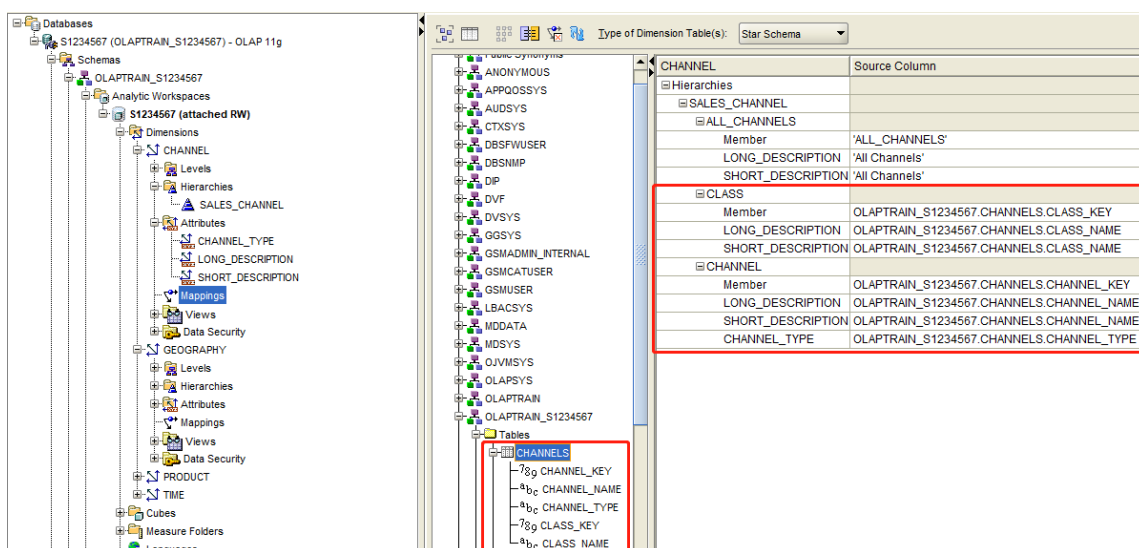
The screenshot displays the Oracle BI Enterprise Desktop interface. On the left, a tree view shows the hierarchy: Databases > S1234567 (OLAPTRAIN_S1234567) - OLAP 11g > Schemas > OLAPTRAIN_S1234567 > Analytic Workspaces > S1234567 (attached RW) > Dimensions > CHANNEL > Hierarchies > SALES_CHANNEL. The SALES_CHANNEL hierarchy is highlighted. On the right, the 'Implementation Details' tab is active, showing the configuration for the SALES_CHANNEL hierarchy. The 'Name' is 'SALES_CHANNEL' and the 'ID' is 'OLAPTRAIN_S1234567.CHANNEL.SALES_CHANNEL'. The 'Set as Default Hierarchy' checkbox is checked. The 'Available Levels' list on the right contains 'ALL_CHANNELS', 'CLASS', and 'CHANNEL'.

Under the Attributes dialogue, we can see that *channel type* has been defined as an attribute attached to **CHANNEL** level, instead of being another hierarchy level, which means it is outside of the business interest and cannot be queried on.



After defining a dimension, those defined components should also be mapped to the existing tables and views in Oracle Database. Specifically, the *Member* attributes in the OLAP dimension should be mapped to the *key* columns in the dimension tables, while the attribute columns should also be mapped to the appropriate OLAP dimension attributes.

Click the **Mappings** in *Channel* dimension and make sure the *source columns* (left) appears in the *mapping pane* (right) correctly. Note that there is no such source data column for **ALL_CHANNELS** level, therefore, for "All/Total" hierarchy levels, the descriptions are typed manually:

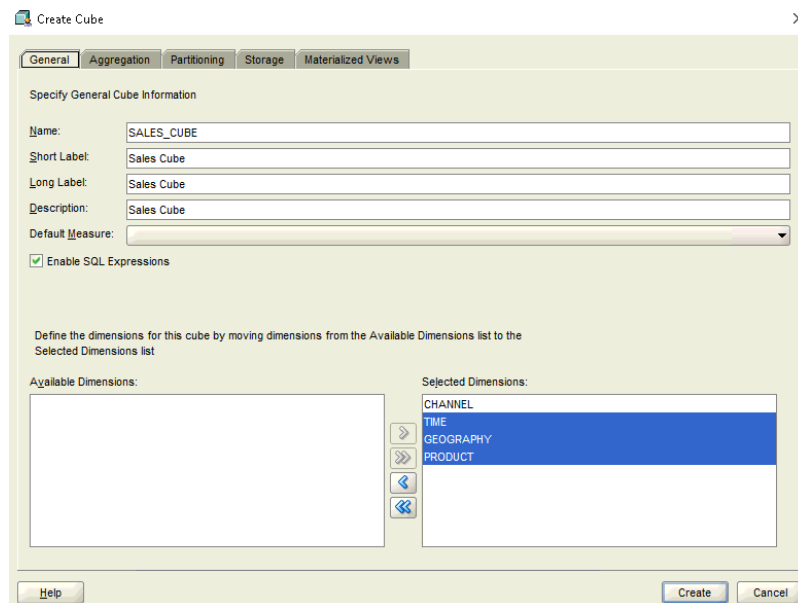


Perform the same inspection to the other imported dimensions and understand how they are defined.

4. Create the data Cube

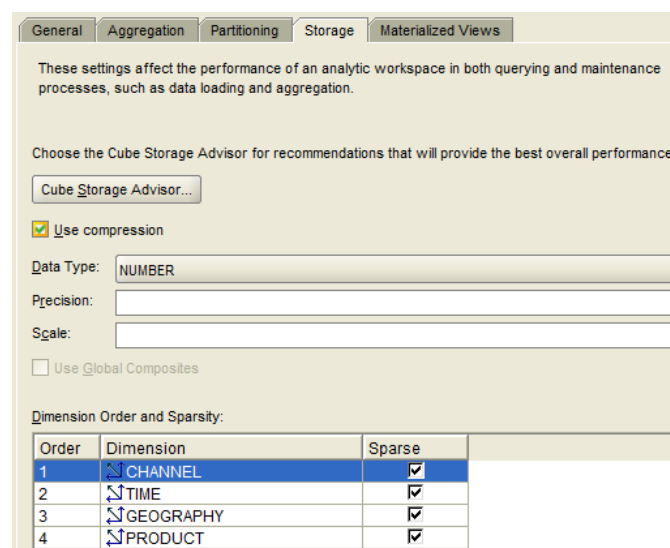
After all dimensions are defined, we are ready to create our cube. Right-click the **Cubes** folder, then click **Create Cube**. In the General tab of the **Create Cube** window, specify the following:

- Name: SALES_CUBE
- Use the Add button (>) to select dimensions in the following order:
 - CHANNEL
 - TIME
 - GEOGRAPHY
 - PRODUCT



Notes: The order in which the dimensions are listed in a cube may affect performance because it determines the way the data is stored on disk.

Next, select the Storage tab. The Storage tab helps you manage the data compression strategy. By default, we choose **Use compression**, and then enable the Sparse option for all dimensions, as shown below:



Finally, click **Apply** to finish the dialogue.

5. Create measures

You can create two types of measures in a cube: *Stored* (or Base) measures, and *Calculated* measures. Every measure that belongs to a particular cube shares the characteristics that were defined for the cube.

(1) Stored Measures

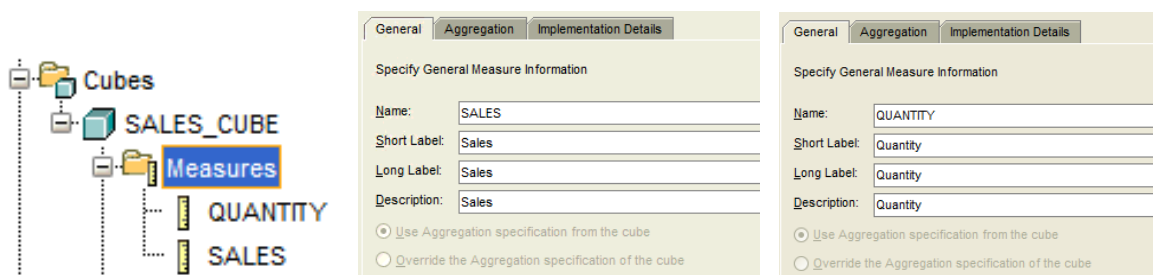
Stored measures store the facts collected about your business. When you create stored measures in your OLAP data model, you will map them to the source data just like what you have done with the dimensions.

(2) Calculated Measures

One of the powerful features of the Oracle OLAP technology is the ability to efficiently and easily generate business calculations of data held in the database. In any OLAP implementation, the number of calculated measures greatly exceeds the number of stored measures.

OLAP calculated measures are derived from stored measures or other calculated measures. These calculations are computed dynamically when users query the data. Calculations are automatically exposed as columns in a cube view, just like the facts.

According to our design, we define two measures, i.e., *Sales* and *Quantity*, and two calculated measures. Two measures are created as follows (right click **Measures** > **Create measure**).



Now we create the first calculated measures by entering or selecting the following (right click **Calculated Measures** > **Create Calculated Measure**):

- Name = **SALES_YTD** (the Label and Description fields are auto-filled)
- Calculation Type = **Period to Date**
- In the Calculation inputs section, select the following:
 - First hyperlink = **Ancestor At Level**
 - Second hyperlink = **TIME.CALENDAR_YEAR**

Finally, the window should look like this:

Name:	SALES_YTD
ID:	OLAPTRAIN_S1234567.SALES_CUBE.SALES_YTD
Short Label:	sales ytd
Long Label:	sales ytd
Description:	sales ytd
Calculation Type:	Period To Date

Calculation:
[Ancestor At Level](#) [TIME.CALENDAR_YEAR](#) to date for [SALES \(...\)](#) in the [TIME](#) dimension and [TIME.CALENDAR](#) hierarchy. Aggregate using [SUM](#) from the [beginning](#) of the period.

Similarly, we create a YTD calculation for the previous year. This measure facilitates year-to-year comparisons. The settings are shown below:

- Name = **SALES_YTD_PY**
- Calculation Type = **Parallel Period**
- In the Calculation inputs section, select the following:
 - Second hyperlink = **SALES_YTD**
 - Fifth hyperlink = **TIME.CALENDAR_YEAR**

General	
Specify General Calculated Measure Information	
Name:	SALES_YTD_PY
Short Label:	Sales Ytd Pr Year
Long Label:	Sales Ytd Pr Year
Description:	Sales Ytd Pr Year
Calculation Type:	Parallel Period

Calculation:
[Parallel period](#) for [SALES_YTD \(...\)](#) in the [TIME](#) dimension and [TIME.CALENDAR](#) hierarchy [1](#) [TIME.CALENDAR_YEAR](#)
 ago based on position from [beginning to ending](#) of period.

6. Map the cube

Same as the dimension mapping, we need to map our cube to the existing data source. In a data cube, we need to map the following fields:

- The stored measures that are defined within the cube.
- The lowest level of detail for each dimension hierarchy.
- The Join Condition field. This field associates the foreign key column from the fact table to the primary key column from the dimension table.

Note that the mapping is done by dragging the corresponding column from the *source columns* (left) to the correct spot in the *mapping pane* (right). A Join condition can be achieved by dragging both joining columns to that slot and the “=” will be added automatically. Please DO NOT type those values manually as it will cause unexpected problems in the future. Eventually, the mapping result should look like the following image (see next page). Click **Apply** to complete the mapping.

SALES_CUBE	Source Column
Measures	
SALES	OLAPTRAIN.SALES_FACT.SALES
QUANTITY	OLAPTRAIN.SALES_FACT.QUANTITY
Dimensions	
TIME	
ALL_YEARS	
CALENDAR_YEAR	
CALENDAR_QUARTER	
MONTH $\Sigma \uparrow$	OLAPTRAIN.TIMES.MONTH_ID
Join Condition	OLAPTRAIN.SALES_FACT.DAY_KEY=OLAPTRAIN.TIMES.DAY_KEY
CHANNEL	
ALL_CHANNELS	
CLASS	
CHANNEL $\Sigma \uparrow$	OLAPTRAIN.CHANNELS.CHANNEL_KEY
Join Condition	OLAPTRAIN.SALES_FACT.CHANNEL=OLAPTRAIN.CHANNELS.CHANNEL_KEY
GEOGRAPHY	
ALL_REGIONS	
REGION	
COUNTRY	
STATE_PROVINCE $\Sigma \uparrow$	OLAPTRAIN.CUSTOMERS.STATE_PROVINCE_KEY
Join Condition	OLAPTRAIN.SALES_FACT.CUSTOMER=OLAPTRAIN.CUSTOMERS.CUSTOMER_KEY
PRODUCT	
ALL_PRODUCTS	
DEPARTMENT	
CATEGORY	
TYPE	
SUBTYPE	
ITEM $\Sigma \uparrow$	OLAPTRAIN.PRODUCTS.ITEM_KEY
Join Condition	OLAPTRAIN.SALES_FACT.PRODUCT=OLAPTRAIN.PRODUCTS.ITEM_KEY
Filter	

Assessment Task 1

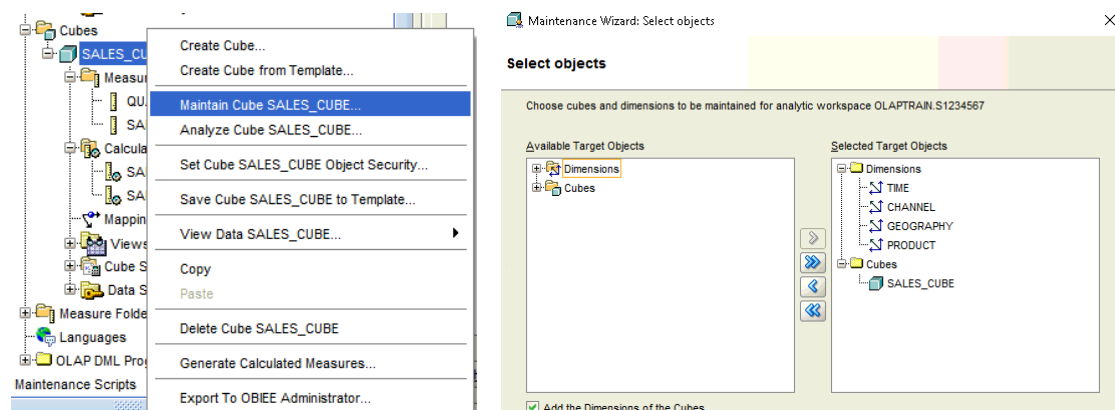
Complete all steps introduced above until you successfully create and map the cube.
Take three screenshots of the cube you have created, including the mapping, and the definition of two calculated measures. Include those screenshots in your document submission.

NOTE: Please make sure **your student ID** appears in every screenshot you take to show the originality of your work. This rule applies to **ALL screenshots** taken throughout this prac unless otherwise specified. Screenshots without student ID may be regarded as invalid and receive reduced marks. An example screenshot of the mapping should look like the following image:

Part 2: Load and View Cube Data

The Maintenance Wizard loads and aggregates the data in a single step. We can load all mapped objects in the analytic workspace, or individual dimensions and measures. We can also choose to run the job immediately, enter it in the Oracle Job Queue, or save it as a SQL script. The materialized views, calculated measures are preprocessed during this maintenance phase. After the maintenance, we can use SQL queries to retrieve the result. Also, we can view the cube data using operators like Roll-up, Drill-down and Pivot.

1. Maintain the cube

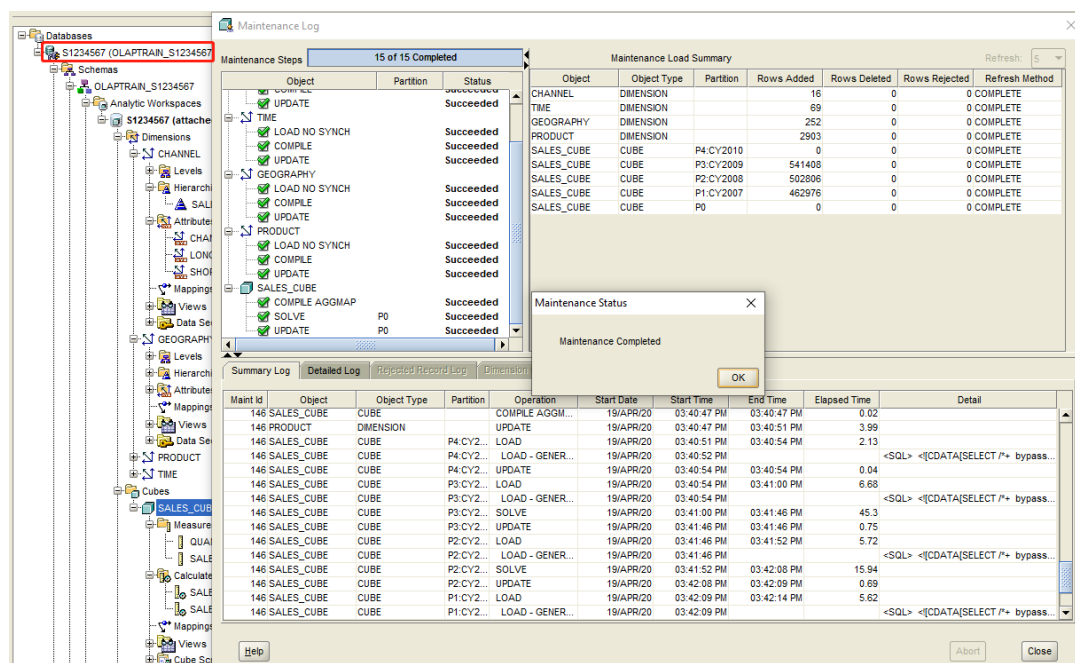


The cube maintenance will load data to the cube. By default, the dimensions of that cube are also processed before the cube. If you have already loaded dimension data, you can specify only to load measure data.

Assessment Task 2

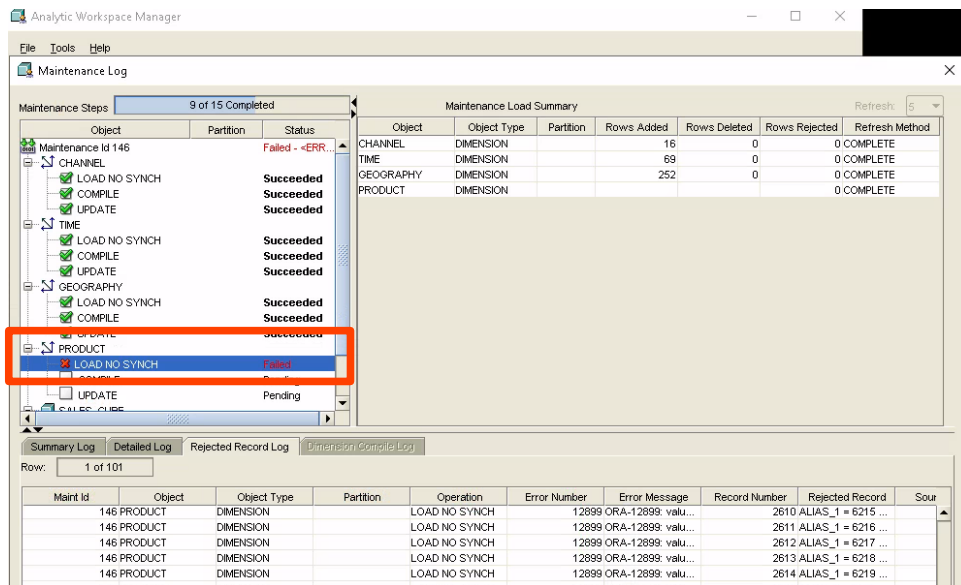
We start to maintain the cube. It will take quite a while to finish the maintenance.

Please take a screenshot of your maintenance result, as shown below, and include it in your report.

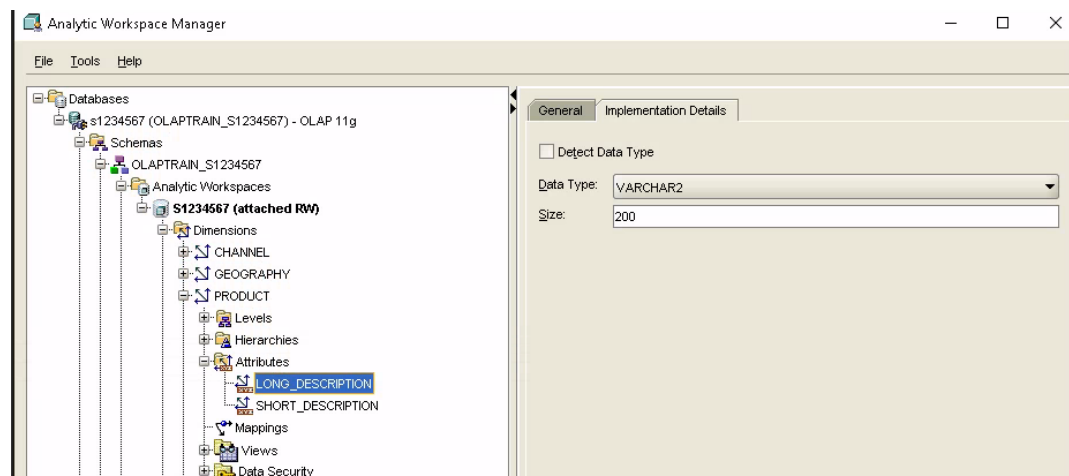


Troubleshooting

If the maintenance task results in errors like the following one, you can fix them by modifying the data storage properties inside the AWM.



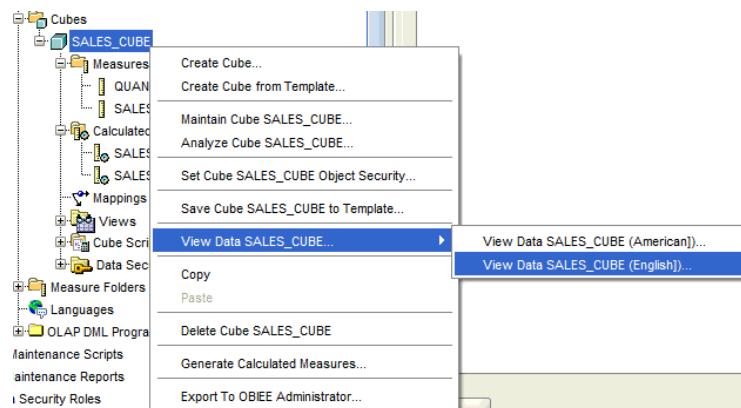
1. Find which cube/tuple is giving the error by clicking "rejected record log" and examining the OBJECT and then clicking the error message. In this case, it is the "PRODUCT" one.
2. Close the maintenance window. Navigate to the "Dimensions" folder in the database.
3. Find the corresponding Dimension to the Object above; here that is PRODUCT.
4. Click PRODUCT and go to "Attributes" > LONG_DESCRIPTION
5. Go to "Implementation Details" and change the datatype size to 200. Make sure you click "apply" at the bottom right corner of the AWM window.



6. Repeat for any more of the same error.

2. View the cube data

Click **View Data SALES_CUBE** to start the Data Viewer.

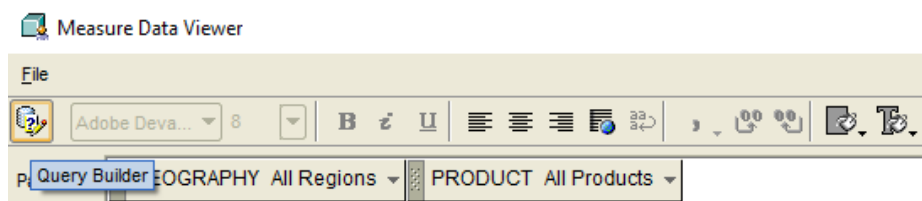


Assessment Task 3

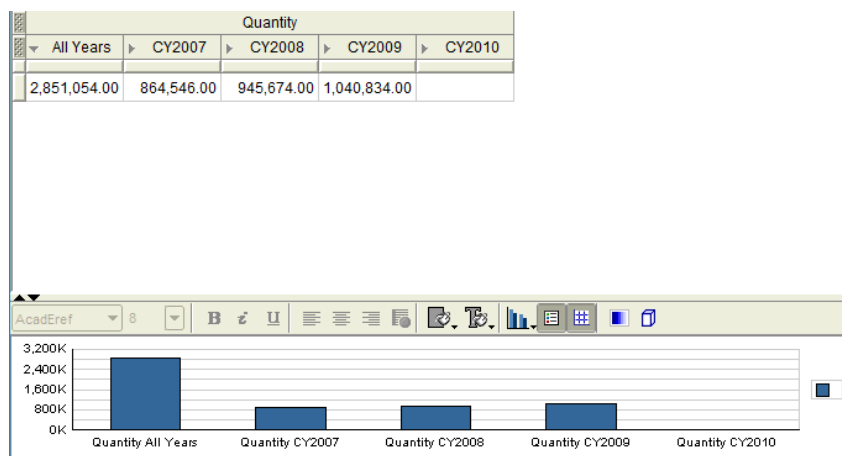
Now you can play with this tool to explore the data. Meanwhile, you are required to achieve a few goals:

- Perform *roll-up*, *drill-down* and *pivot* operations in Data Viewer, respectively (review the lecture slides and tutorials for a reminder about these operations). Each operation includes three parts:
 - (1) the screenshot of the view before the operation;
 - (2) a sentence describing which operation you are performing on which columns/levels; and
 - (3) the screenshot of the view after the operation. Therefore, there are 6 screenshots and 3 lines of description in total. Include the screenshots and descriptions in your document.
- Adjust your Data Viewer window until it is identical (the value can be different) to the following two views (**Hint:** your viewer windows must be identical to get a full mark and you may need help from the **Query Builder**, shown below). Take a screenshot for each view you made and put them in your document.

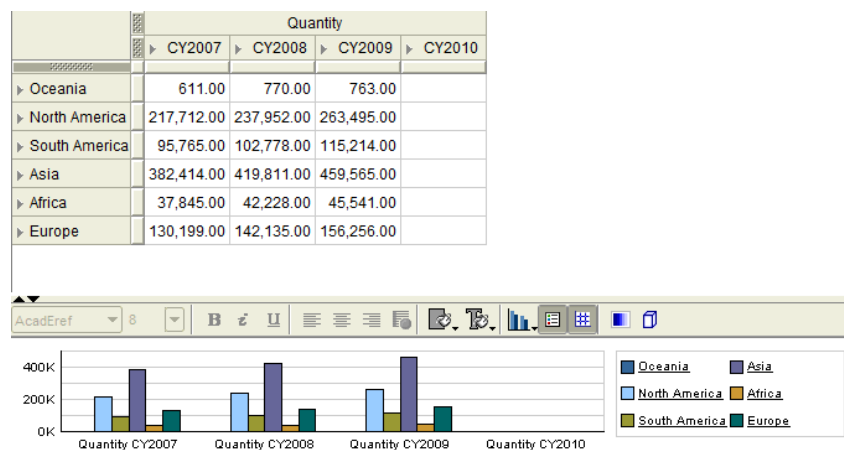
The query builder is the icon on the top left (see the following image).



View one: Show the Quantity across all years, and broken down by each CY (2007–2010).



View two: Show the Quantity broken down by each CY (2007–2010), and grouped by the location.



Part 3: Understand Data Warehouse Design Mechanism

We have learnt how to create dimensions and cubes using Oracle OLAP and AWM. Now, we will try to understand the mechanism behind these operations.

NOTE: There is NO mark in this part, but it will help you understand the whole process of how data cubes are built, queried and managed by Oracle OLAP, which will make the subsequent tasks easier to follow. Therefore, **this part is optional but recommended**. Please follow the instructions below.

1. View the dimension tables and fact table

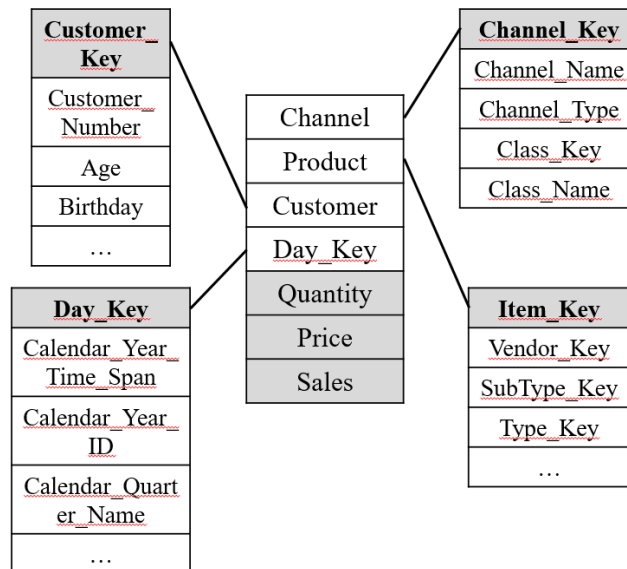
Connect to the olaptrain schema via **SQL Developer** using your “OLAPTRAIN_S1234567” (Once again, S1234567 refers to your student number) user, and then check the tables “CHANNELS” and “SALES_FACT”.

The screenshot shows the SQL Developer interface. On the left, the 'Connections' pane shows a connection to 'DW'. Below it, the 'Tables (Filtered)' pane shows a list of tables, including 'CHANNELS' and 'SALES_FACT'. The 'CHANNELS' table is expanded, showing its columns: 'CHANNEL_KEY', 'CHANNEL_NAME', 'CHANNEL_TYPE', 'CLASS_KEY', and 'CLASS_NAME'. The 'SALES_FACT' table is also expanded, showing its columns: 'QUANTITY', 'PRICE', 'SALES', 'DAY_KEY', 'PRODUCT', 'CHANNEL', and 'CUSTOMER'.

CHANNEL_KEY	CHANNEL_NAME	CHANNEL_TYPE	CLASS_KEY	CLASS_NAME
1	18 Catalog	Mail Order Catalog	-4	Indirect
2	19 New York Retail	Boutique	-3	Direct
3	20 Lisbon Retail	Boutique	-3	Direct
4	21 San Francisco Retail	Mall Store	-3	Direct
5	22 Internet	Web Store	-4	Indirect
6	23 London Retail	Boutique	-3	Direct

QUANTITY	PRICE	SALES	DAY_KEY	PRODUCT	CHANNEL	CUSTOMER
1	105	105	16/JAN/07	4644	23	237353
1	49.99	49.99	21/JAN/07	4888	18	451245
1	74.95	74.95	30/JAN/07	5518	18	451984
1	29	29	28/JAN/07	3962	22	327660
1	119	119	29/JAN/07	4061	22	383753
1	329	329	28/JAN/07	6200	26	331926
1	22	22	06/JAN/07	5928	18	322508
1	45	45	06/JAN/07	3784	18	322508
1	29	29	28/JAN/07	3962	22	315919
1	39.95	39.95	24/JAN/07	5242	27	124966
1	82	82	28/JAN/07	5409	28	215504
1	105	105	31/JAN/07	6098	22	289674
1	289	289	28/JAN/07	4395	20	123542
1	45	45	14/JAN/07	4281	22	523457

Just like what we have learnt from the lecture and tutorial, the dimension table “CHANNELS” contains dimension key (*Channel_Key*), hierarchy (*Channel* → *Class* → *All Channels*) and attributes (*Channel_Name*, *Channel_Type*), while the fact table consists of the dimension key for each dimension (*Channel*, *Product*, *Customer*, etc.) and the facts (*Quantity*, *Price* and *Sales*). The dimension tables and fact table constitute a star schema as follows:



2. Compare the views with dimension tables and fact table

Now, let us view the data in the “CHANNEL” dimension in our analytic workspace. Connect to our analytic workspace through **AWM** and open the *views* in the “CHANNEL” dimension. Here we have two views: the dimension view “CHANNEL_VIEW” and the hierarchy view “CHANNEL_SALES_CHANNEL_VIEW”.



In addition to the raw dimension table “CHANNELS”, the “CHANNEL_VIEW” stores the hierarchy information, and the “CHANNEL_SALES_CHANNEL_VIEW” contains additional columns indicating the hierarchy level of each record.

	DIM_KEY	LEVEL_NAME	MEMBER_TYPE	DIM_ORDER
1	18	CHANNEL	L	3
2	19	CHANNEL	L	4
3	20	CHANNEL	L	5
4	21	CHANNEL	L	6
5	22	CHANNEL	L	7
6	23	CHANNEL	L	8
7	24	CHANNEL	L	9
8	25	CHANNEL	L	10
9	26	CHANNEL	L	11
10	27	CHANNEL	L	12
11	28	CHANNEL	L	13
12	29	CHANNEL	L	14
13	30	CHANNEL	L	15
14	-4	CLASS	L	2
15	-3	CLASS	L	1
16	ALL_CHANNELS	ALL_CHANNELS	A	16

PARENT	DEPTH	ALL_CHANNELS	CLASS	CHANNEL
-4		2ALL_CHANNELS	-4	18
-3		2ALL_CHANNELS	-3	19
-3		2ALL_CHANNELS	-3	20
-3		2ALL_CHANNELS	-3	21
-4		2ALL_CHANNELS	-4	22
-3		2ALL_CHANNELS	-3	23
-3		2ALL_CHANNELS	-3	24
-3		2ALL_CHANNELS	-3	25
-3		2ALL_CHANNELS	-3	26
-3		2ALL_CHANNELS	-3	27
-3		2ALL_CHANNELS	-3	28
-4		2ALL_CHANNELS	-4	29
-3		2ALL_CHANNELS	-3	30
ALL_CHANNE...		1ALL_CHANNELS	-4	
ALL_CHANNE...		1ALL_CHANNELS	-3	
		0ALL_CHANNELS		

By storing the hierarchy information as separate records in dimension tables, we can easily store the summarization of each hierarchy level when maintaining a cube. Check the materialized view “SALES_CUBE_VIEW” in the cube “SALES_CUBE” and see such results:

	SALES	QUANTITY	SALES_YTD	SALES_YTD_PY	CHANNEL	TIME	GEOGRAPHY	PRODUCT
1	417515017.27	2851054			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	ALL_PRODUCTS
2	49286079.48	440239			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	-520
3	49286079.48	440239			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	-530
4	31773651.06	179710			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	-592
5	28498878.54	164197			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	-891
6	6809412.34	41367			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	3926
7	771311.15	4685			ALL_CHANNELS	ALL_YEARS	-12	3926
8	6765.59	41			ALL_CHANNELS	ALL_YEARS	-69	3926
9	6765.59	41			ALL_CHANNELS	ALL_YEARS	-229	3926
10	1631.9	10	1631.9		ALL_CHANNELS	CY2007	-69	3926
11	1631.9	10	1631.9		ALL_CHANNELS	CY2007	-229	3926
12	849.95	5	849.95		ALL_CHANNELS	Q1CY2007	-69	3926
13	849.95	5	849.95		ALL_CHANNELS	Q1CY2007	-229	3926
14	339.98	2	509.97		ALL_CHANNELS	FEB2007	-69	3926
15	339.98	2	509.97		ALL_CHANNELS	FEB2007	-229	3926
16	169.99	1	169.99		-4	FEB2007	-69	3926
17	169.99	1	169.99		-4	FEB2007	-229	3926

Therefore, given above materialized views, we can answer the aggregation queries quickly. Assuming the user is viewing the total sales on “All_Channels”, “All_Years”, “All_Regions” and “All_Products”, we can easily perform a drill-down operation from “All_Channels” to “Class” by SELECT the Sales from “SALES_CUBE_VIEW” where:

- Time = “All_Years”
- Geography = “All_Regions”
- Product = “All_Products”
- Channel in {-3,-4}

as we can identify that -3 and -4 in “CHANNEL_VIEW” refer to the two possible classes. With the help of the materialized view “SALES_CUBE_VIEW” and the dimension tables, the complex aggregation queries are converted to simple selections.

3. Cube optimization

With the help of the “SALES_CUBE_VIEW”, we can easily get the results of the aggregation queries. However, maintaining such a cube view is not easy. If you open an SQL Plus terminal and type in the following two queries, you will find out that the number of records in the fact table and the cube are 2,811,097 and 24,440,506 respectively (since the second query may take more than one hour, it is not mandatory to run these queries).

```
/* Check the count of records in the fact table. */
SELECT count(*) FROM SALES_FACT;

/* Check the count of records in the cube view*/
SELECT count(*) FROM SALES _CUBE_VIEW;
```

Therefore, a cube view contains many more records than the original table since the cube size is determined by the size of dimensions. In our case, although each dimension only contains tens to thousands of rows, the cube size already exceeds the million level. Hence, it may be impractical to pre-compute all 24 million aggregation results in the maintenance step.

In order to achieve a balance between maintenance cost and query performance, the analytic workspace provides a bunch of configurations to control the cost of maintenance and query:

(1) Data compression

When we built our cube in Part 1, we leveraged the data compression feature and specified the sparse dimensions as shown below:

The screenshot shows the 'Storage' tab in the Oracle OLAP configuration window. It includes a 'Cube Storage Advisor...' button, a checked 'Use compression' checkbox, and input fields for 'Data Type' (NUMBER), 'Precision', and 'Scale'. There is also an unchecked 'Use Global Composites' checkbox. Below these is a section titled 'Dimension Order and Sparsity' containing a table with four dimensions: CHANNEL, TIME, GEOGRAPHY, and PRODUCT. Each dimension is marked as 'Sparse' with a checked checkbox.

Order	Dimension	Sparse
1	CHANNEL	<input checked="" type="checkbox"/>
2	TIME	<input checked="" type="checkbox"/>
3	GEOGRAPHY	<input checked="" type="checkbox"/>
4	PRODUCT	<input checked="" type="checkbox"/>

The data sparsity is very common in dimensional data models. When there are a large number of empty cells in a cube, the cube is said to be “sparse”. In our data cube, if we define multiple dimensions as “sparse” dimensions, the Oracle OLAP will create a special index for the cube so as to automatically manage sparsity. Compression on sparse dimensions can also significantly reduce the size of cubes and improve the performance of both data loads and queries. Do not try to uncheck the “Use compression” and maintain the cube, it will take days!

(2) Cost-based aggregation

Cost-based aggregation enables you to balance the maintenance cost and query performance directly. Specify a percentage value and the database will pre-compute and store the most costly aggregate values based on your input. That is to say, if we set the value to 30%, then 30% of the aggregate values will be calculated and stored during data maintenance, and 70% will be calculated in response to a query. Try to change the value into something between 1% to 100%, and you will see a time difference in maintaining the cube. Please maintain the cube again to apply the changes you made each time. However, the query performance varies slightly as the cube is relatively small. (**Note:** “Partition Cube” should be unchecked to enable cost-based aggregation).

The screenshot shows the 'Aggregation' tab in the Oracle OLAP configuration window. It has sub-tabs for 'Rules' and 'Precompute'. Under the 'Precompute' sub-tab, there is a section for 'Cost-based aggregation (recommended for compressed cubes)' with a 'Bottom Partition' input field set to 30.

(3) Dimension partitioning and ordering

Partitioning is a method of physically storing the measures in a cube. It improves the performance of large measures in the following ways:

- Improves scalability by keeping data structures small. Each partition function is like a smaller measure.
- Keeps the working set of data smaller both for queries and maintenance, since the relevant data are stored together.
- Enables parallel aggregation during data maintenance. Each partition can be aggregated by a separate process.
- Simplifies removal of old data from storage. Old partitions can be dropped, and new partitions can be added.

You can activate the partitioning as below, and choose a proper partitioning strategy to accelerate your maintenance and queries.

The screenshot shows the 'Partitioning' tab of a configuration window. At the top, there are tabs for 'General', 'Aggregation', 'Partitioning', 'Storage', and 'Materialized Views'. Below the tabs, the text 'Specify Cube Partition Information' is followed by a checked checkbox labeled 'Partition cube'. Underneath, there are two sub-tabs: 'Select Partitions' and 'Partition Member Analysis'. The 'Select Partitions' sub-tab is active. It contains a text box 'Dimension:' with 'TIME' selected in a dropdown menu. Below this, the text 'Aggregation Hierarchies:' is followed by two buttons: 'Order Hierarchies' and 'Clear Selections'. A list box titled 'CALENDAR' contains four items: 'ALL_YEARS : 1 members' (unchecked), 'CALENDAR_YEAR : 4 members' (checked), 'CALENDAR_QUARTER : 16 members' (unchecked), and 'MONTH : 48 members' (unchecked).

Also, the order of the dimensions in a cube may affect performance. In general, when you dimension a cube, the first dimension in a cube should have the smallest cardinality and the last dimension has the largest.

Part 4: Query Data Cube via SQL

As mentioned above, the views provided by Oracle data cube are similar to traditional table-based star models. However, there are two key differences:

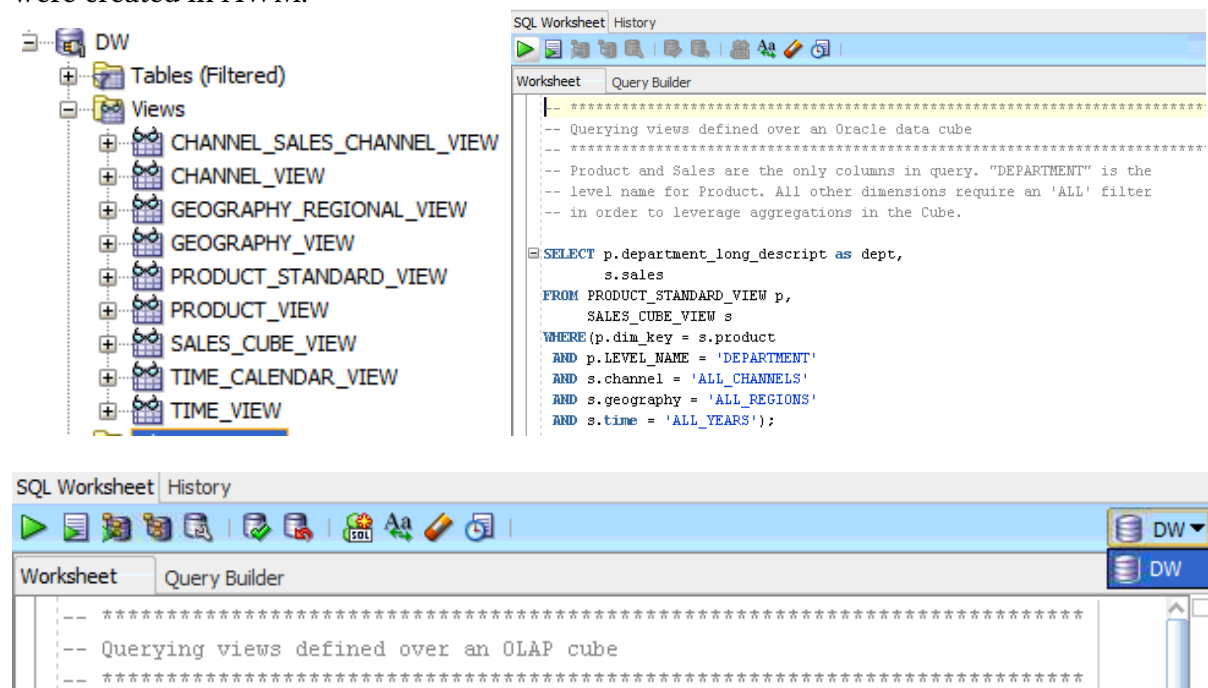
- Fact tables in a star schema store detailed data, while the cube views reveal many summary levels.
- In addition to the facts in fact tables, the cube provides additional measures and calculations, which are calculated and materialized as columns in the cube view.

These differences impact the way you query data. With star queries, you aggregate the data by combining aggregation functions (such as sum) and the **GROUP BY** clause. With cube queries, if the cube has been fully calculated (cube view is fully materialized), you simply select the data you want (either stored or calculated measures) as a column. Typically, no aggregation function is necessary since the data has already been summarized by the cube.

Since the cube data is made directly accessible to SQL by a set of relational views, in this part, we will write some SQL queries to query the cube views.

1. Simple Cube Queries

Get back to the **SQL Developer** and connect to the **olaptrain** schema. Check the views that were created in AWM.



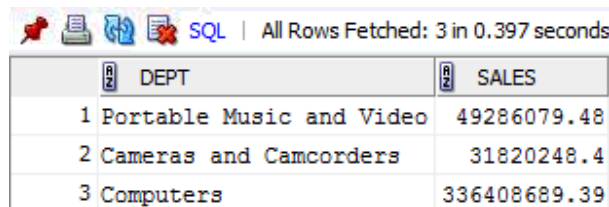
Select **File > Open** and open the “cube_queries.sql” file in the *queries* folder from the extracted files. Check the following query:

```
SELECT p.department_long_descript as dept,  
       s.sales  
FROM PRODUCT_STANDARD_VIEW p,  
      SALES_CUBE_VIEW s  
WHERE (p.dim_key = s.product  
AND p.LEVEL_NAME = 'DEPARTMENT'  
AND s.channel = 'ALL_CHANNELS'  
AND s.geography = 'ALL_REGIONS'  
AND s.time = 'ALL_YEARS');
```

Query Notes:

- “Sales” is simply selected as a column. There is no SQL aggregation operation applied.
- A level within the Product dimension hierarchy – **Department** – is used to filter product members.
- All of the dimensions are qualified in the WHERE clause, even though only the Product dimension is selected. In OLAP cube queries, dimensions that are not selected in the query require an “ALL” condition – which specifies the top-level hierarchy value for each of the dimension columns – in order to leverage summaries that are already computed by the cube.

Execute the query using your connection (“DW_S1234567” in the above example). The query should return three rows almost instantaneously. The results should look like this:



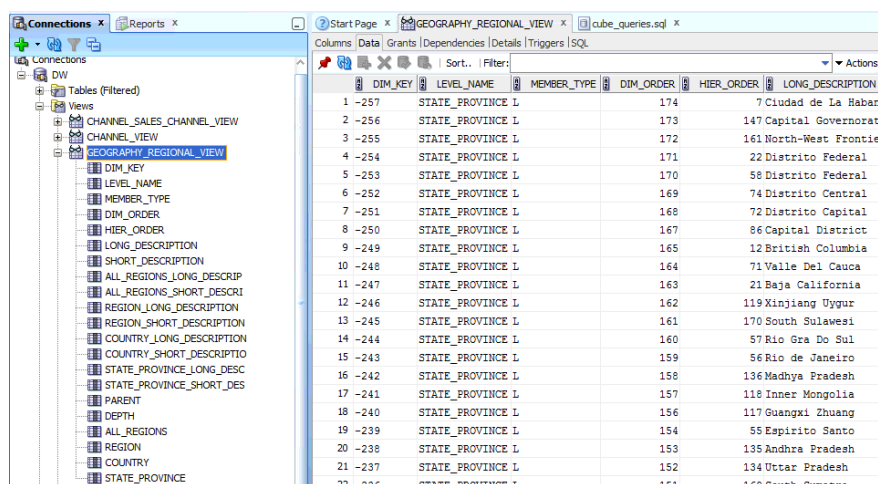
DEPT	SALES
1 Portable Music and Video	49286079.48
2 Cameras and Camcorders	31820248.4
3 Computers	336408689.39

2. Advanced aggregation query

Now you are asked to write a similar query as above. In the previous query, a “**Level**” Condition was used for the Product dimension (which was the only dimension selected). Here, you need to apply level conditions to multiple dimensions in a cube query.

Every hierarchy and dimension view contains a “**LEVEL_NAME**” column. The value in this column is the name of the OLAP hierarchy Level object that you created when modelling the dimension in AWM. By simply specifying a value for this column in the WHERE clause, you filter the data to include only those dimension members at the specified level in the hierarchy.

Since the Oracle OLAP truncates column names at 24 characters, in order to get the exact name of a column, simply drill on the view that you want to examine by using SQL Developer’s Connections navigator. You can also view the data to see the exact values in a column.



DIM_KEY	LEVEL_NAME	MEMBER_TYPE	DIM_ORDER	HIER_ORDER	LONG_DESCRIPTION
1 -257	STATE_PROVINCE L		174	7	Ciudad de La Habana
2 -256	STATE_PROVINCE L		173	147	Capital Governorate
3 -255	STATE_PROVINCE L		172	161	North-West Frontier
4 -254	STATE_PROVINCE L		171	22	Distrito Federal
5 -253	STATE_PROVINCE L		170	58	Distrito Federal
6 -252	STATE_PROVINCE L		169	74	Distrito Central
7 -251	STATE_PROVINCE L		168	72	Distrito Capital
8 -250	STATE_PROVINCE L		167	86	Capital District
9 -249	STATE_PROVINCE L		165	12	British Columbia
10 -248	STATE_PROVINCE L		164	71	Valle Del Cauca
11 -247	STATE_PROVINCE L		163	21	Baja California
12 -246	STATE_PROVINCE L		162	119	Xinjiang Uygur
13 -245	STATE_PROVINCE L		161	170	South Sulawesi
14 -244	STATE_PROVINCE L		160	57	Rio Gra Do Sul
15 -243	STATE_PROVINCE L		159	56	Rio de Janeiro
16 -242	STATE_PROVINCE L		158	136	Madhya Pradesh
17 -241	STATE_PROVINCE L		157	118	Inner Mongolia
18 -240	STATE_PROVINCE L		156	117	Guangxi Zhuang
19 -239	STATE_PROVINCE L		154	55	Espirito Santo
20 -238	STATE_PROVINCE L		153	135	Andhra Pradesh
21 -237	STATE_PROVINCE L		152	134	Uttar Pradesh
22 -236	STATE_PROVINCE L		151	169	South Sumatra

Assessment Task 4

Now, complete the following SQL query:

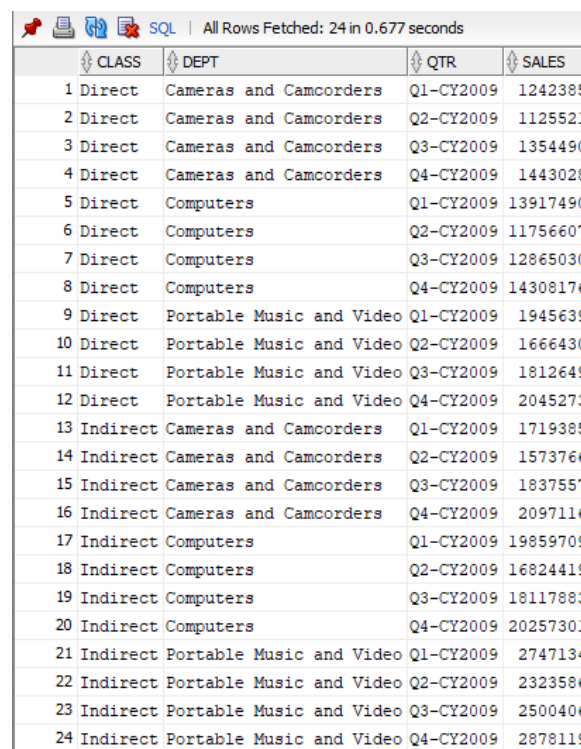
```
SELECT      c.class_short_description as class,
            p.department_long_descript as dept,
            t.calendar_quarter_long_de as qtr,
            round(s.sales) as sales
FROM        ?
WHERE       ?
```

In this query we need to use **c**, **p**, **g**, **t**, **s** to denote the the views from **channel**, **product**, **geography**, **time** dimensions and **Sales_cube**, respectively.

Please perform a summarization on dimension levels:

- channel.CLASS,
- product.DEPARTMENT,
- geography.ALL_REGION, and
- time.CALENDAR_QUARTER.

Also ensure that the time is filtered for “CY2009” only. The query result should look like this (the record order doesn’t matter):



CLASS	DEPT	QTR	SALES
1 Direct	Cameras and Camcorders	Q1-CY2009	1242385
2 Direct	Cameras and Camcorders	Q2-CY2009	1125521
3 Direct	Cameras and Camcorders	Q3-CY2009	1354490
4 Direct	Cameras and Camcorders	Q4-CY2009	1443028
5 Direct	Computers	Q1-CY2009	13917490
6 Direct	Computers	Q2-CY2009	11756607
7 Direct	Computers	Q3-CY2009	12865030
8 Direct	Computers	Q4-CY2009	14308176
9 Direct	Portable Music and Video	Q1-CY2009	1945639
10 Direct	Portable Music and Video	Q2-CY2009	1666430
11 Direct	Portable Music and Video	Q3-CY2009	1812649
12 Direct	Portable Music and Video	Q4-CY2009	2045273
13 Indirect	Cameras and Camcorders	Q1-CY2009	1719385
14 Indirect	Cameras and Camcorders	Q2-CY2009	1573766
15 Indirect	Cameras and Camcorders	Q3-CY2009	1837557
16 Indirect	Cameras and Camcorders	Q4-CY2009	2097116
17 Indirect	Computers	Q1-CY2009	19859709
18 Indirect	Computers	Q2-CY2009	16824419
19 Indirect	Computers	Q3-CY2009	18117883
20 Indirect	Computers	Q4-CY2009	20257301
21 Indirect	Portable Music and Video	Q1-CY2009	2747134
22 Indirect	Portable Music and Video	Q2-CY2009	2323586
23 Indirect	Portable Music and Video	Q3-CY2009	2500406
24 Indirect	Portable Music and Video	Q4-CY2009	2878119

Tip: If the width of the column is too wide to display your results, you can shorten it via the SUBSTR operator: `SELECT SUBSTR(p.department_long_descript, 0, 20) as dept`

Include your query and a screenshot of the results in your submitted document.

Changelog

- v1: The original prac sheet.
- v2: Changed the description of where data should be stored due to permissions issues.
- v3: Clarified submission format - Submitting the scripts is optional if they are shown in your PDF report. Added tips for shortening output width.
- v4: Added troubleshooting details for cube maintenance