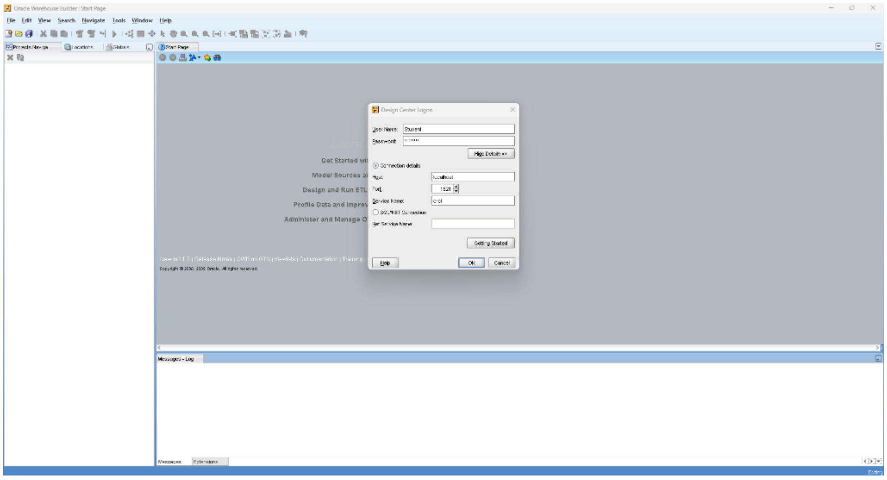
PRACTICAL NO. 1

Aim: Deployment of data cube using OLAP operations

Following are the steps to create a project:

1.Login into the Design Center and right click inside project explorer, select New and select project



Design Center Logon

User Name: Student

Password: ********

Connection details: ●

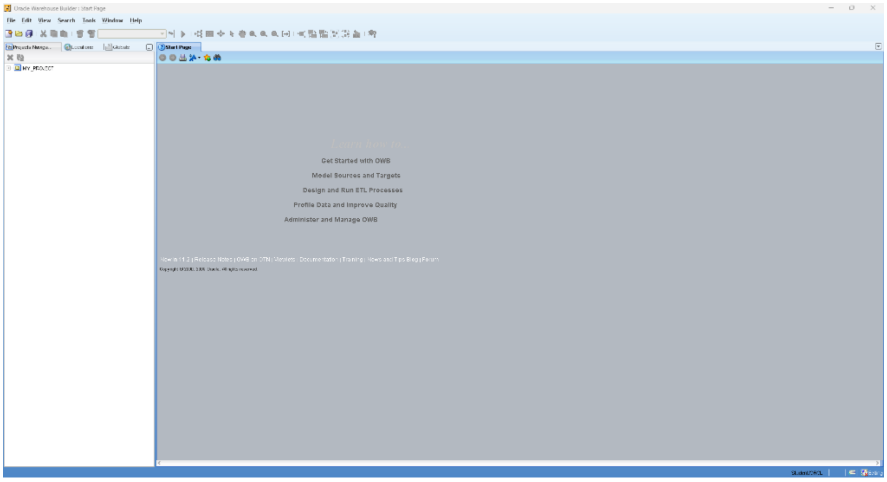
Host: localhost

Port: 1521

Service Name: orcl

SQL*NET Connection: ○

Net Service Name:

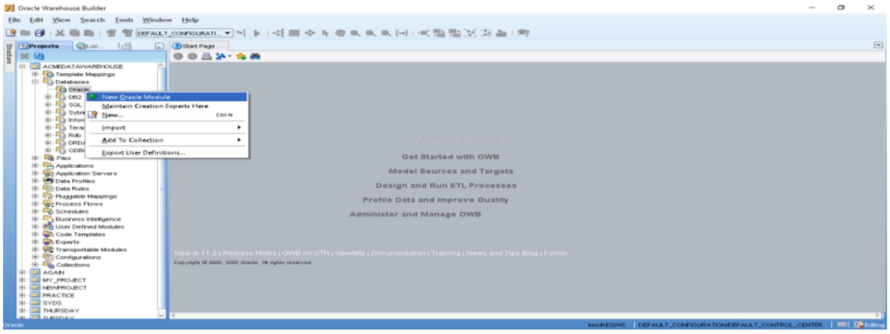


Oracle Warehouse Builder : Start Page
File Edit View Search Tools Window Help
Start Page
MY_PROJECT
Get Started with OWB
Model Sources and Targets
Design and Run ETL Processes
Profile Data and Improve Quality
Administer and Manage OWB

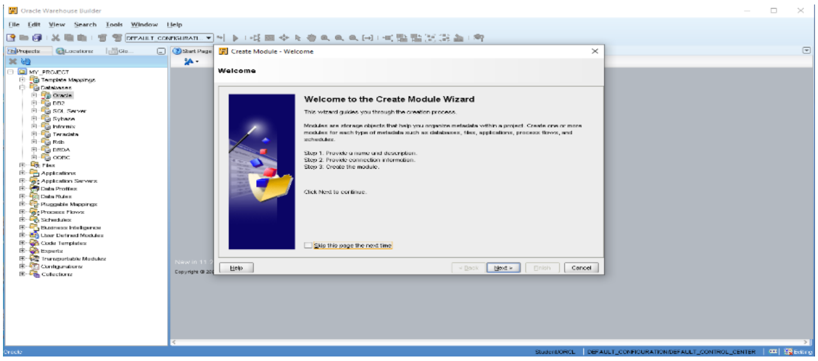
The project appears in the project list:

Following are the steps to create a module in the project :

1. Expand the project, go to Databases and expand it , select Oracle -> right click -> New Oracle Module



2.Set Name of the module


Oracle Warehouse Builder
File Edit View Search Tools Window Help
MY_PROJECT
Create Module - Welcome
Help
< Back
Next >
Finish
Cancel
Student/ORCL | DEFAULT_CONFIGURATION/DEFAULT_CONTROL_CENTER

Welcome

Welcome to the Create Module Wizard

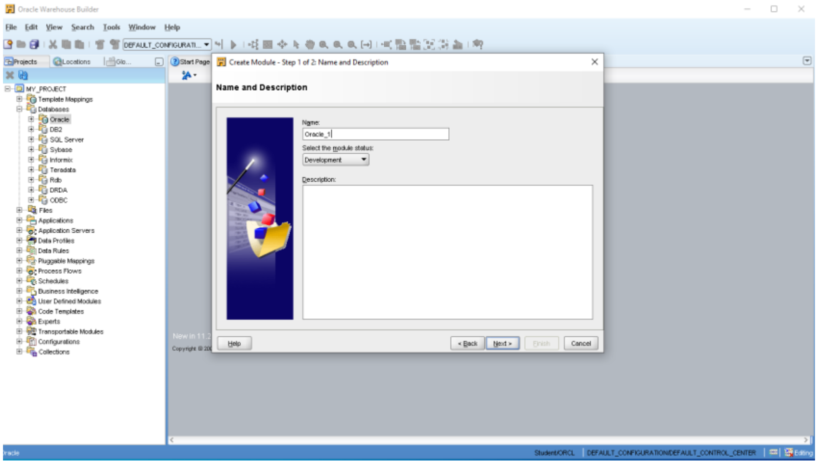
This wizard guides you through the creation process.

Modules are storage objects that help you organize metadata within a project. Create one or more modules for each type of metadata such as databases, files, applications, process flows, and schedules.

Step 1: Provide a name and description.
Step 2: Provide connection information.
Step 3: Create the module.

Click Next to continue.

□ Skip this page the next time


Oracle Warehouse Builder
File Edit View Search Tools Window Help
MY_PROJECT
Create Module - Step 1 of 2: Name and Description
Help
< Back
Next >
Finish
Cancel
Student/ORCL | DEFAULT_CONFIGURATION/DEFAULT_CONTROL_CENTER

Name and Description

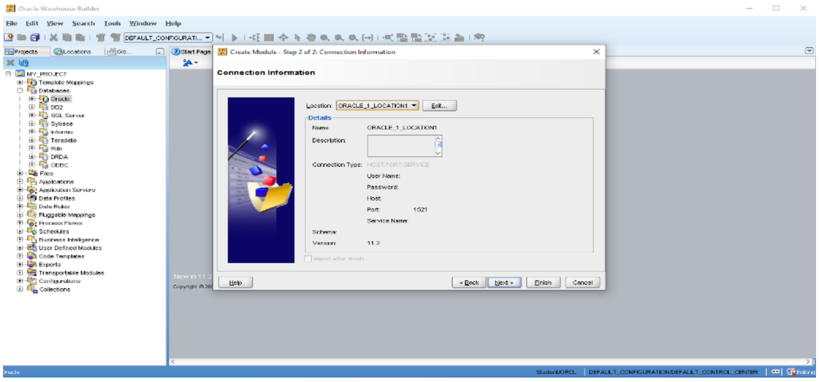
Name: Oracle_1

Select the module status: Development

Description:

3.Click Next

4.At step 2 , click the EDIT button and provide the connection information :


Oracle Warehouse Builder
File Edit View Search Tools Window Help
MY_PROJECT
Create Module - Step 2 of 2: Connection Information
Edit...
Help
< Back
Next >
Finish
Cancel
Student/ORCL | DEFAULT_CONFIGURATION/DEFAULT_CONTROL_CENTER

Connection Information

Location: ORACLE_1_LOCATION1

Details

Name: ORACLE_1_LOCATION1

Description:

Connection Type: HOST:PORT:SERVICE

User Name:

Password:

Host:

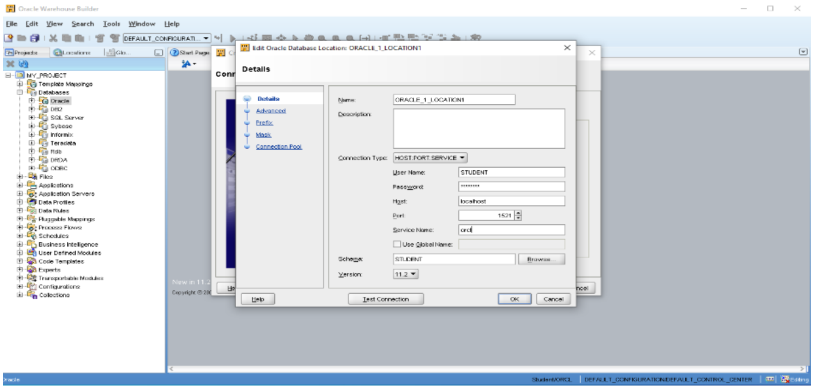
Port: 1521

Service Name:

Schema:

Version: 11.2

□ Import after finish



Edit Oracle Database Location: ORACLE_1_LOCATION1

Details

Name: ORACLE_1_LOCATION1

Description:

Connection Type: HOST:PORT:SERVICE

User Name: STUDENT

Password: ********

Host: localhost

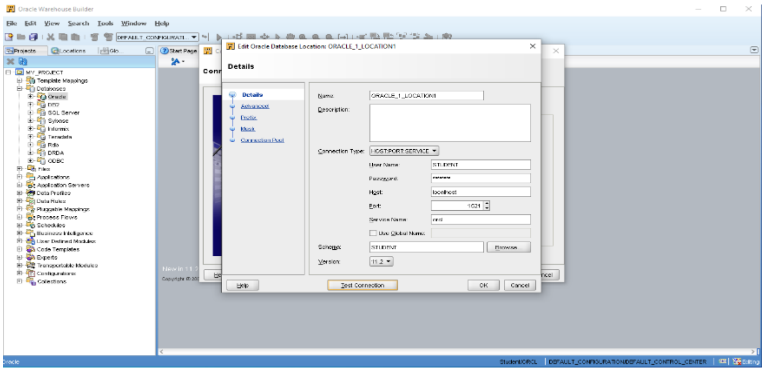
Port: 1521

Service Name: orcl

□ Use Global Name:

Schema: STUDENT

Version: 11.2



Edit Oracle Database Location: ORACLE_1_LOCATION1

Details

Name: ORACLE_1_LOCATION1

Description:

Connection Type: HOST:PORT:SERVICE

User Name: STUDENT

Password: ******

Host: localhost

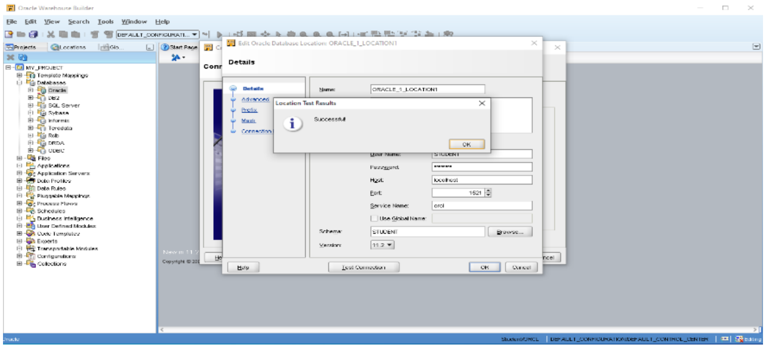
Port: 1521

Service Name: orcl

□ Use Global Name:

Schema: STUDENT

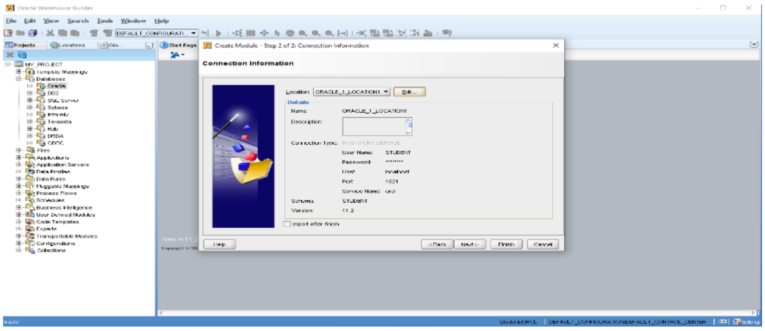
Version: 11.2



Location Test Results

Successful

5.Click on Test Connection -> OK -> Next -> Finish



Create Module - Step 2 of 2: Connection Information

Connection Information

Location: ORACLE_1_LOCATION1

Details

Name: ORACLE_1_LOCATION1

Description:

Connection Type: HOST:PORT:SERVICE

User Name: STUDENT

Password: ********

Host: localhost

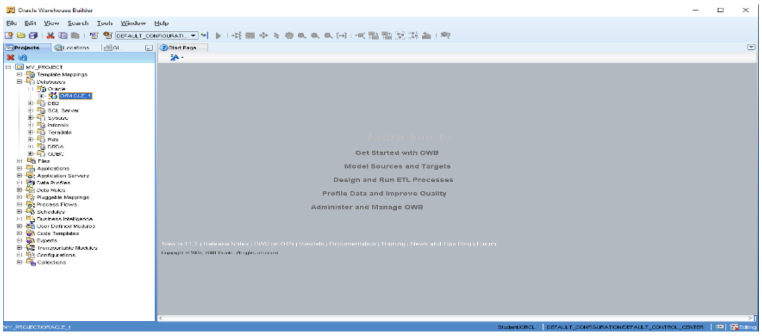
Port: 1521

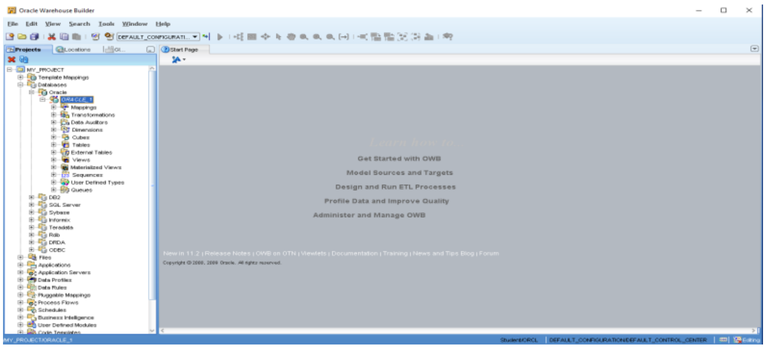
Service Name: orcl

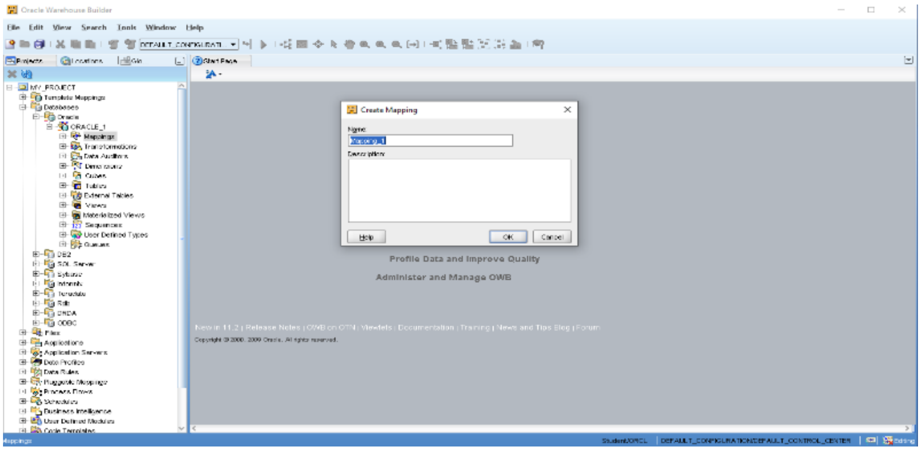
Schema: STUDENT

Version: 11.2

□ Import after finish

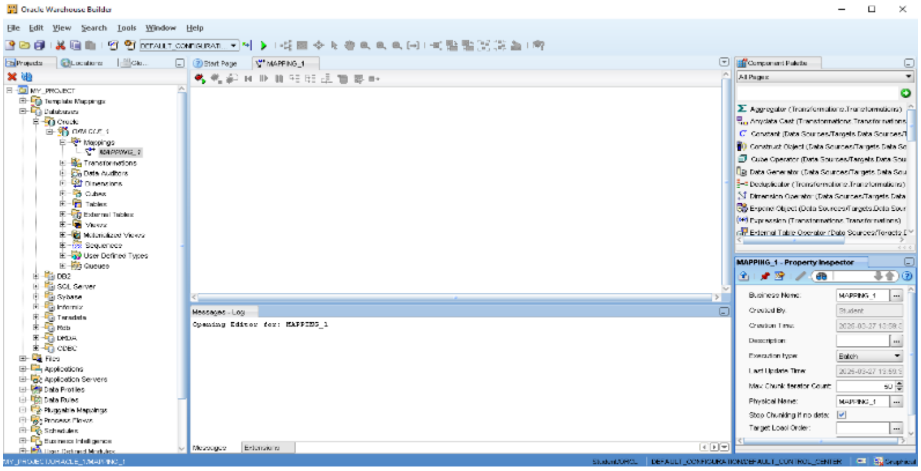






Name:
Mapping_1

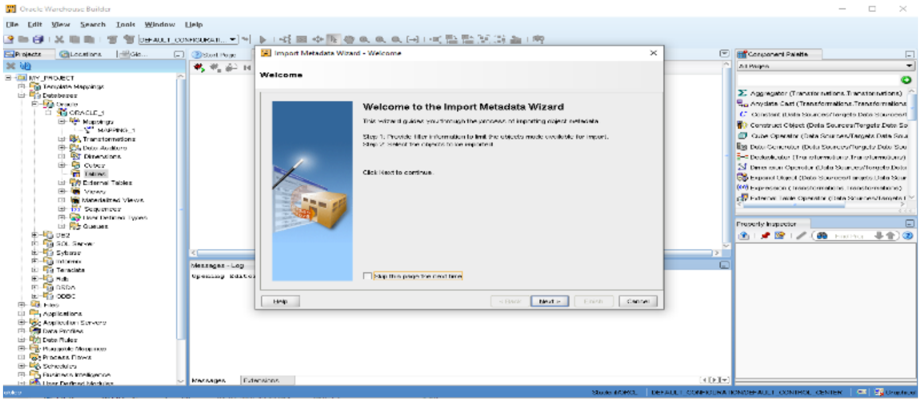
Description:


Oracle Warehouse Builder
File Edit View Search Tools Window Help
DEFAULT_CONFIGURATI...
Projects
Locations
Start Page
MAPPING_1
MY_PROJECT
Template Mappings
Databases
Oracle
ORACLE_1
Mappings
MAPPING_1
Transformations
Data Auditors
Dimensions
Cubes
Tables
External Tables
Views
Materialized Views
Sequences
User Defined Types
Queues
DB2
SQL Server
Sybase
Informix
Teradata
Rdb
DRDA
ODBC
Files
Applications
Application Servers
Data Profiles
Data Rules
Pluggable Mappings
Process Flows
Schedules
Business Intelligence
User Defined Modules
Component Palette
All Pages
Aggregator (Transformations.Transformations)
Anydata Cast (Transformations.Transformations)
Constant (Data Sources/Targets.Data Sources/
Construct Object (Data Sources/Targets.Data So
Cube Operator (Data Sources/Targets.Data Sou
Data Generator (Data Sources/Targets.Data Sou
Deduplicator (Transformations.Transformations)
Dimension Operator (Data Sources/Targets.Data
Expand Object (Data Sources/Targets.Data Sour
Expression (Transformations.Transformations)
External Table Operator (Data Sources/Targets.I
Messages - Log
Messages
Extensions
MY_PROJECT/ORACLE_1/MAPPING_1
Student/ORCL
DEFAULT_CONFIGURATION/DEFAULT_CONTROL_CENTER
Graphical

```
Opening Editor for: MAPPING_1
```

MAPPING_1 - Property Inspector

Business Name:	MAPPING_1
Created By:	Student
Creation Time:	2026-03-27 13:59:3
Description:	
Execution type:	Batch
Last Update Time:	2026-03-27 13:59:5
Max Chunk Iterator Count:	50
Physical Name:	MAPPING_1
Stop Chunking if no data:	☑
Target Load Order:	



Import Metadata Wizard - Welcome

Welcome

Welcome to the Import Metadata Wizard

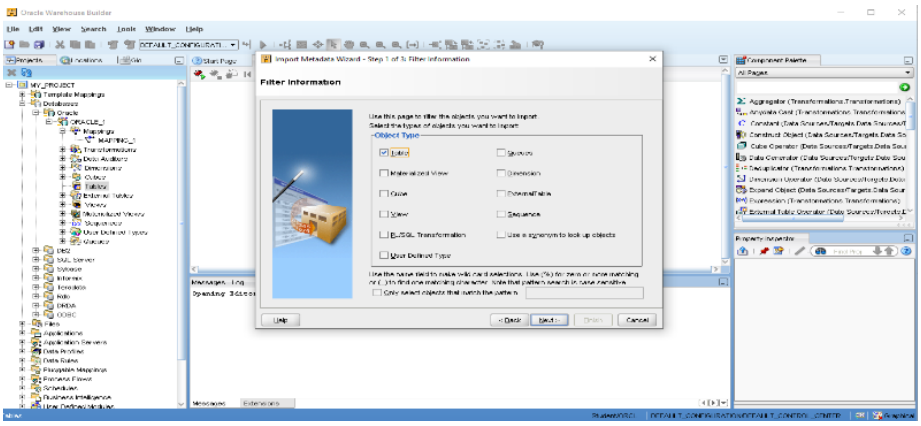
This wizard guides you through the process of importing object metadata.

Step 1: Provide filter information to limit the objects made available for import.
Step 2: Select the objects to be imported.

Click Next to continue.

□ Skip this page the next time

2.Select Tables and click next:



Import Metadata Wizard - Step 1 of 3: Filter Information

Filter Information

Use this page to filter the objects you want to import.
Select the types of objects you want to import.

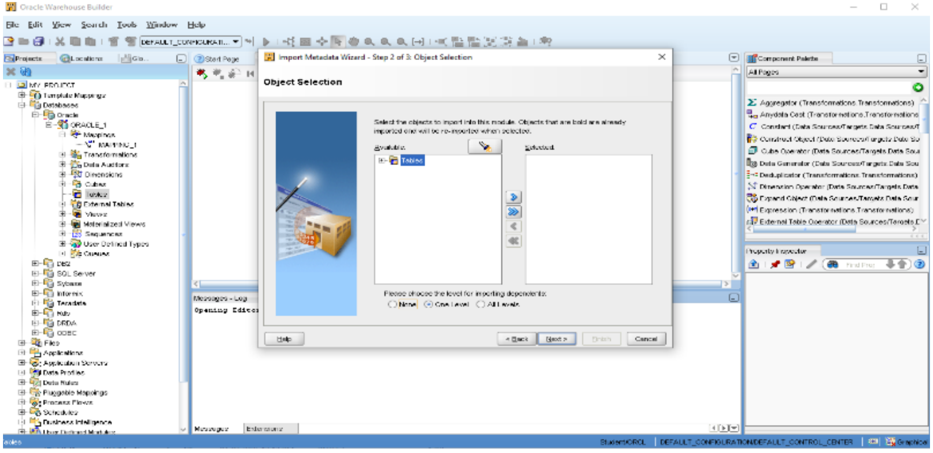
Object Type

☑ Table
□ Queues
□ Materialized View
□ Dimension
□ Cube
□ ExternalTable
□ View
□ Sequence
□ PL/SQL Transformation
□ Use a synonym to look up objects
□ User Defined Type

Use the name field to make wild card selections. Use (%) for zero or more matching or (_) to find one matching character. Note that pattern search is case sensitive.

□ Only select objects that match the pattern

3.Expand the option Tables and select the 5 acme tables



Import Metadata Wizard - Step 2 of 3: Object Selection

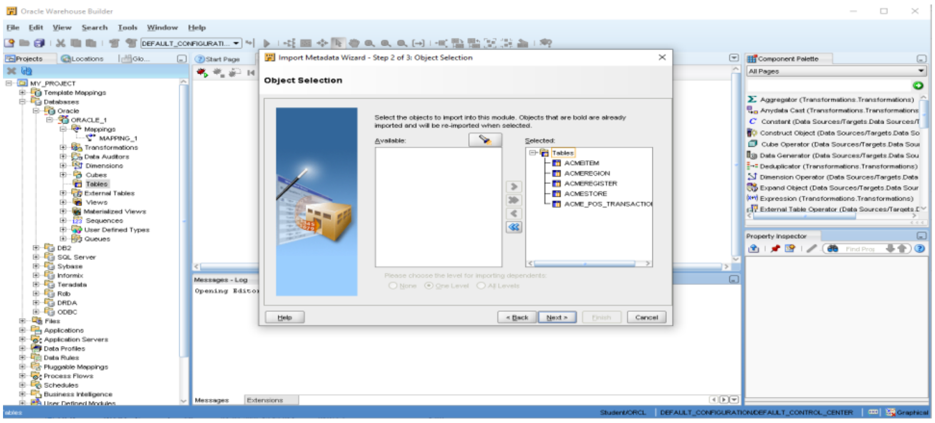
Object Selection

Select the objects to import into this module. Objects that are bold are already imported and will be re-imported when selected.

Available:
Tables

Selected:

Please choose the level for importing dependents:
○ None ● One Level ○ All Levels



Import Metadata Wizard - Step 2 of 3: Object Selection

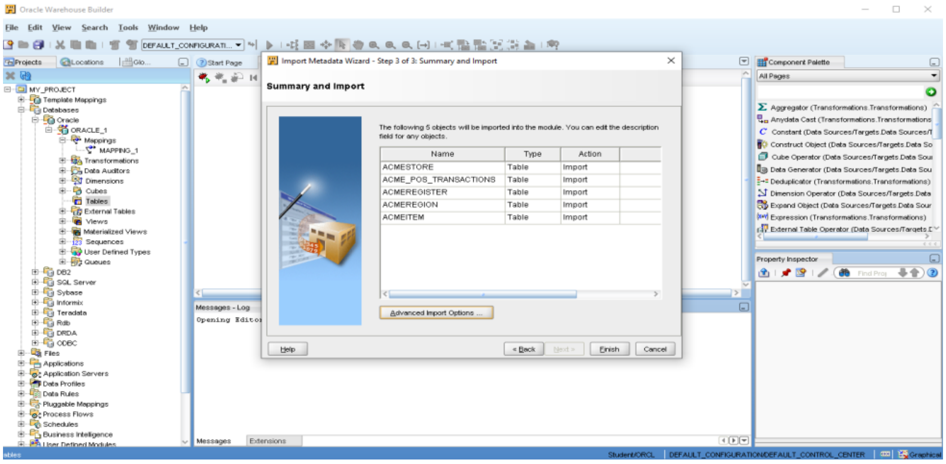
Object Selection

Select the objects to import into this module. Objects that are bold are already imported and will be re-imported when selected.

Available:

Selected:
Tables
ACMEITEM
ACMEREGION
ACMEREGISTER
ACMESTORE
ACME_POS_TRANSACTIO

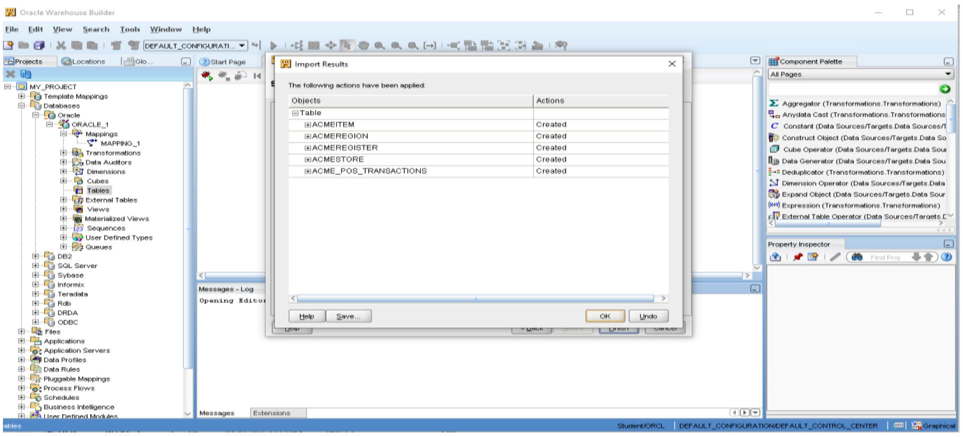
Please choose the level for importing dependents:
○ None ● One Level ○ All Levels


Import Metadata Wizard - Step 3 of 3: Summary and Import

Summary and Import

The following 5 objects will be imported into the module. You can edit the description field for any objects.

Name	Type	Action
ACMESTORE	Table	Import
ACME_POS_TRANSACTIONS	Table	Import
ACMEREGISTER	Table	Import
ACMEREGION	Table	Import
ACMEITEM	Table	Import


Import Results

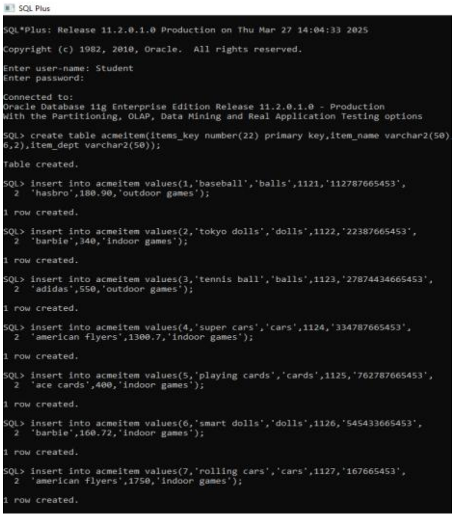
The following actions have been applied:

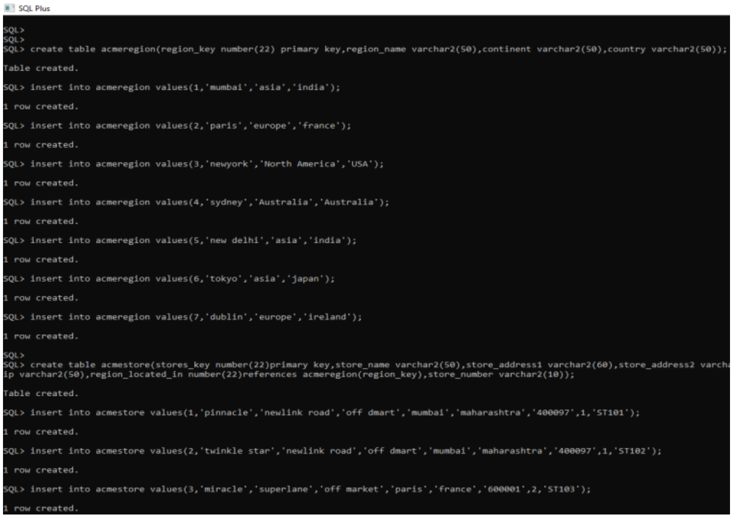
Objects	Actions
⊟Table	
⊞ACMEITEM	Created
⊞ACMEREGION	Created
⊞ACMEREGISTER	Created
⊞ACMESTORE	Created
⊞ACME_POS_TRANSACTIONS	Created

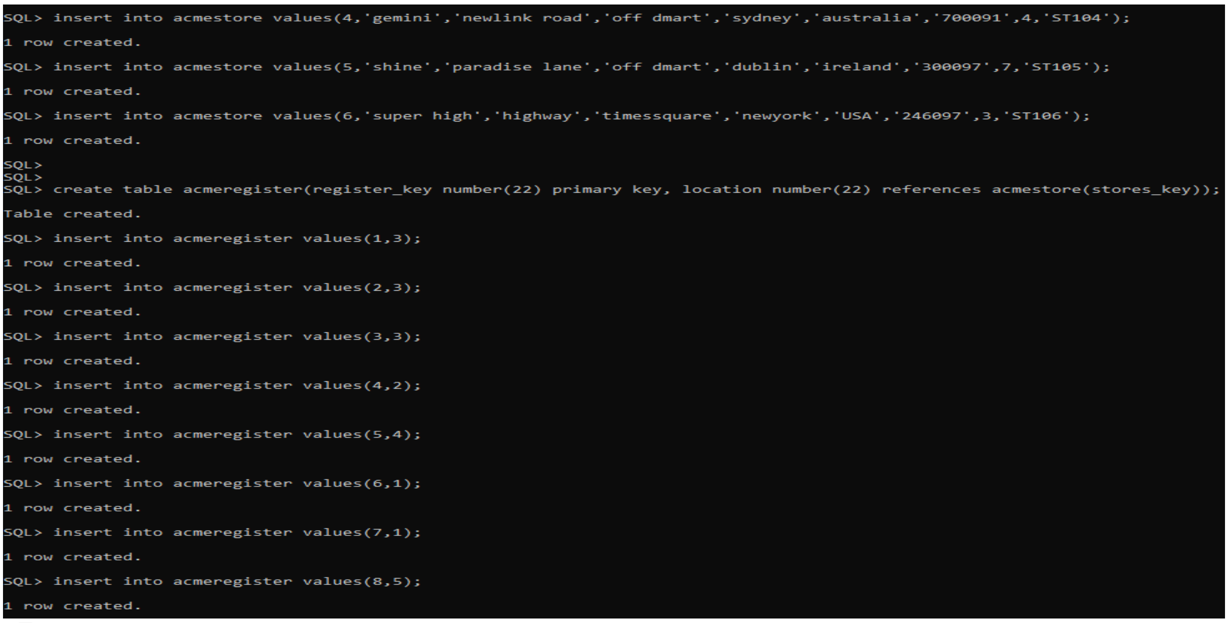
4.Click next and Finish

Creating, Mapping and Deploying Stage Table

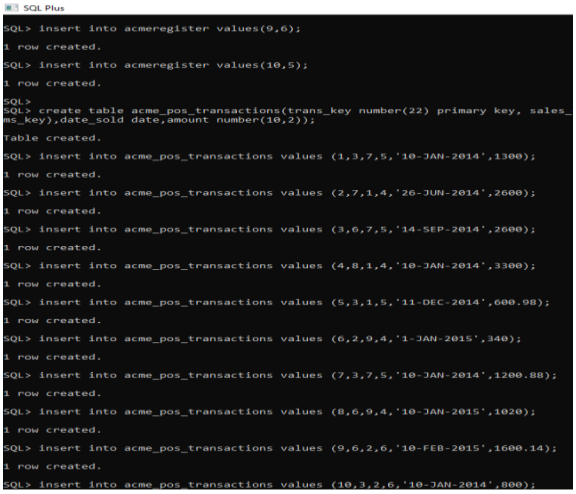
Go to SQL*Plus enter username and password correctly and then SQL interpreter will open then cut paste the queries from acmepioneerdatabase sql file control+A+C by clicking right selecting edit option along with notepad


```
SQL*Plus: Release 11.2.0.1.0 Production on Thu Mar 27 14:04:33 2025

Copyright (c) 1982, 2010, Oracle.  All rights reserved.

Enter user-name: Student
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> create table acmeitem(items_key number(22) primary key,item_name varchar2(50)
6,2),item_dept varchar2(50));

Table created.

SQL> insert into acmeitem values(1,'baseball','balls',1121,'112787665453',
  2  'hasbro',180.90,'outdoor games');

1 row created.

SQL> insert into acmeitem values(2,'tokyo dolls','dolls',1122,'22387665453',
  2  'barbie',340,'indoor games');

1 row created.

SQL> insert into acmeitem values(3,'tennis ball','balls',1123,'27874434665453',
  2  'adidas',550,'outdoor games');

1 row created.

SQL> insert into acmeitem values(4,'super cars','cars',1124,'334787665453',
  2  'american flyers',1300.7,'indoor games');

1 row created.

SQL> insert into acmeitem values(5,'playing cards','cards',1125,'762787665453',
  2  'ace cards',400,'indoor games');

1 row created.

SQL> insert into acmeitem values(6,'smart dolls','dolls',1126,'545433665453',
  2  'barbie',160.72,'indoor games');

1 row created.

SQL> insert into acmeitem values(7,'rolling cars','cars',1127,'167665453',
  2  'american flyers',1750,'indoor games');

1 row created.
```

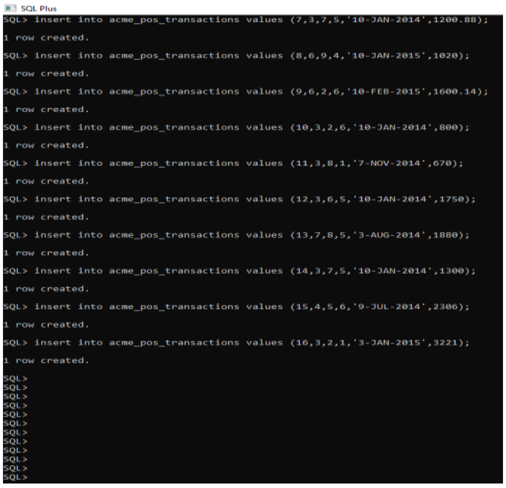

```
SQL>
SQL>
SQL> create table acmeregion(region_key number(22) primary key,region_name varchar2(50),continent varchar2(50),country varchar2(50));

Table created.

SQL> insert into acmeregion values(1,'mumbai','asia','india');

1 row created.

SQL> insert into acmeregion values(2,'paris','europe','france');

1 row created.

SQL> insert into acmeregion values(3,'newyork','North America','USA');

1 row created.

SQL> insert into acmeregion values(4,'sydney','Australia','Australia');

1 row created.

SQL> insert into acmeregion values(5,'new delhi','asia','india');

1 row created.

SQL> insert into acmeregion values(6,'tokyo','asia','japan');

1 row created.

SQL> insert into acmeregion values(7,'dublin','europe','ireland');

1 row created.

SQL>
SQL> create table acmestore(stores_key number(22)primary key,store_name varchar2(50),store_address1 varchar2(60),store_address2 varcha
ip varchar2(50),region_located_in number(22)references acmeregion(region_key),store_number varchar2(10));

Table created.

SQL> insert into acmestore values(1,'pinnacle','newlink road','off dmart','mumbai','maharashtra','400097',1,'ST101');

1 row created.

SQL> insert into acmestore values(2,'twinkle star','newlink road','off dmart','mumbai','maharashtra','400097',1,'ST102');

1 row created.

SQL> insert into acmestore values(3,'miracle','superlane','off market','paris','france','600001',2,'ST103');

1 row created.
```



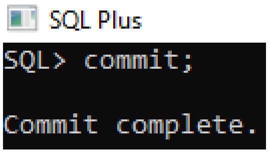
```
SQL> insert into acmestore values(4,'gemini','newlink road','off dmart','sydney','australia','700091',4,'ST104');

1 row created.

SQL> insert into acmestore values(5,'shine','paradise lane','off dmart','dublin','ireland','300097',7,'ST105');

1 row created.

SQL> insert into acmestore values(6,'super high','highway','timessquare','newyork','USA','246097',3,'ST106');

1 row created.

SQL>
SQL>
SQL> create table acmeregister(register_key number(22) primary key, location number(22) references acmestore(stores_key));

Table created.

SQL> insert into acmeregister values(1,3);

1 row created.

SQL> insert into acmeregister values(2,3);

1 row created.

SQL> insert into acmeregister values(3,3);

1 row created.

SQL> insert into acmeregister values(4,2);

1 row created.

SQL> insert into acmeregister values(5,4);

1 row created.

SQL> insert into acmeregister values(6,1);

1 row created.

SQL> insert into acmeregister values(7,1);

1 row created.

SQL> insert into acmeregister values(8,5);

1 row created.
```



```
SQL> insert into acmeregister values(9,6);

1 row created.

SQL> insert into acmeregister values(10,5);

1 row created.

SQL>
SQL> create table acme_pos_transactions(trans_key number(22) primary key, sales_
ms_key),date_sold date,amount number(10,2));

Table created.

SQL> insert into acme_pos_transactions values (1,3,7,5,'10-JAN-2014',1300);

1 row created.

SQL> insert into acme_pos_transactions values (2,7,1,4,'26-JUN-2014',2600);

1 row created.

SQL> insert into acme_pos_transactions values (3,6,7,5,'14-SEP-2014',2600);

1 row created.

SQL> insert into acme_pos_transactions values (4,8,1,4,'10-JAN-2014',3300);

1 row created.

SQL> insert into acme_pos_transactions values (5,3,1,5,'11-DEC-2014',600.98);

1 row created.

SQL> insert into acme_pos_transactions values (6,2,9,4,'1-JAN-2015',340);

1 row created.

SQL> insert into acme_pos_transactions values (7,3,7,5,'10-JAN-2014',1200.88);

1 row created.

SQL> insert into acme_pos_transactions values (8,6,9,4,'10-JAN-2015',1020);

1 row created.

SQL> insert into acme_pos_transactions values (9,6,2,6,'10-FEB-2015',1600.14);

1 row created.

SQL> insert into acme_pos_transactions values (10,3,2,6,'10-JAN-2014',800);
```



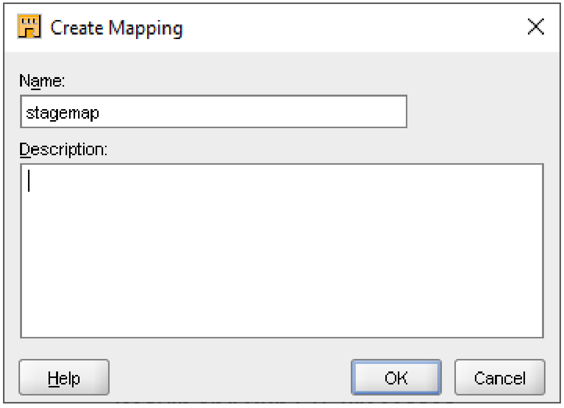
```
SQL> insert into acme_pos_transactions values (7,3,7,5,'10-JAN-2014',1200.88);

1 row created.

SQL> insert into acme_pos_transactions values (8,6,9,4,'10-JAN-2015',1020);

1 row created.

SQL> insert into acme_pos_transactions values (9,6,2,6,'10-FEB-2015',1600.14);

1 row created.

SQL> insert into acme_pos_transactions values (10,3,2,6,'10-JAN-2014',800);

1 row created.

SQL> insert into acme_pos_transactions values (11,3,8,1,'7-NOV-2014',670);

1 row created.

SQL> insert into acme_pos_transactions values (12,3,6,5,'10-JAN-2014',1750);

1 row created.

SQL> insert into acme_pos_transactions values (13,7,8,5,'3-AUG-2014',1880);

1 row created.

SQL> insert into acme_pos_transactions values (14,3,7,5,'10-JAN-2014',1300);

1 row created.

SQL> insert into acme_pos_transactions values (15,4,5,6,'9-JUL-2014',2306);

1 row created.

SQL> insert into acme_pos_transactions values (16,3,2,1,'3-JAN-2015',3221);

1 row created.

SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
```



```
SQL> commit;

Commit complete.
```

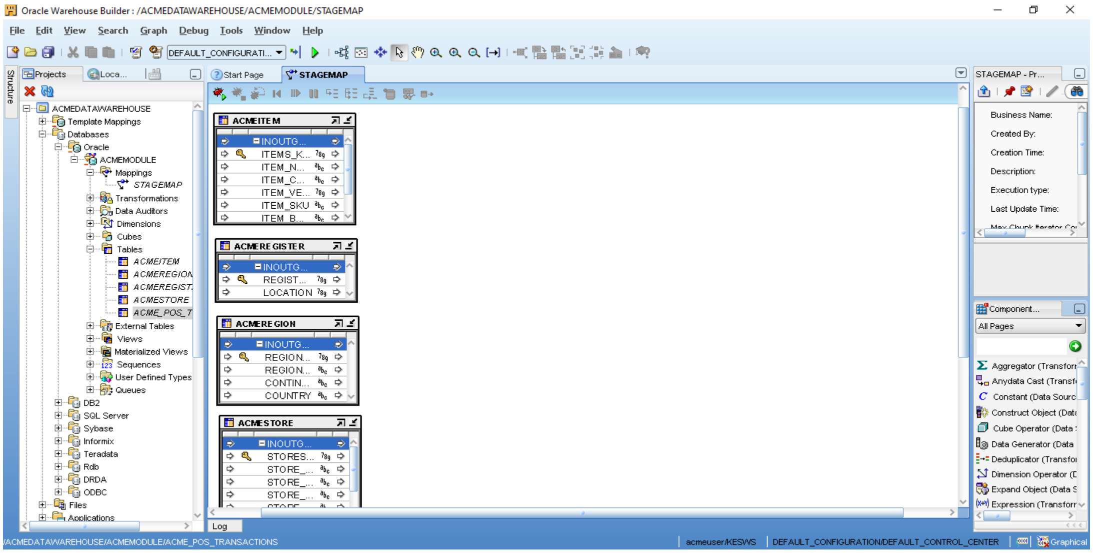
Following are the steps to create a stage mapping

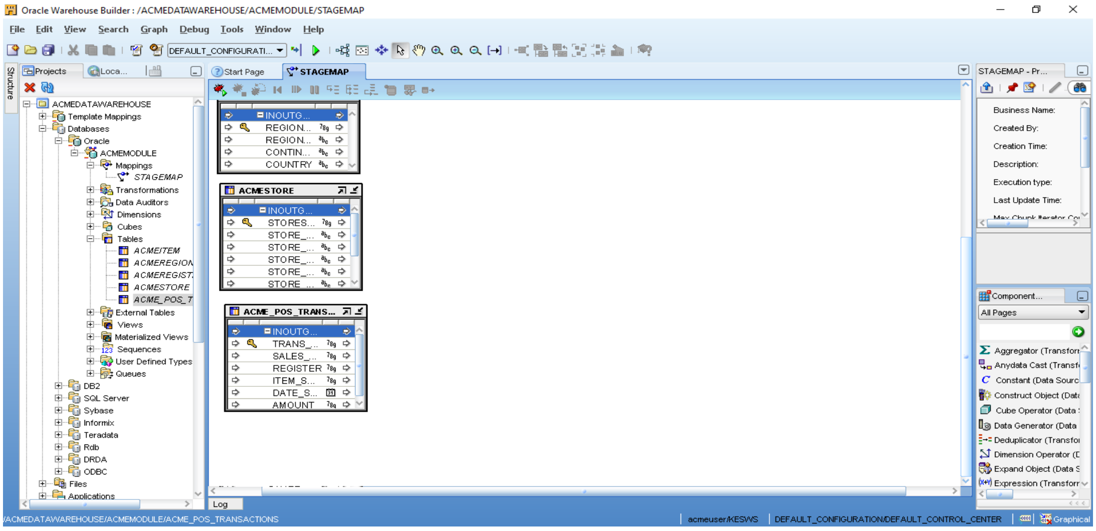
Expand module -> Right click mapping -> Select New Mapping and set name to the mapping



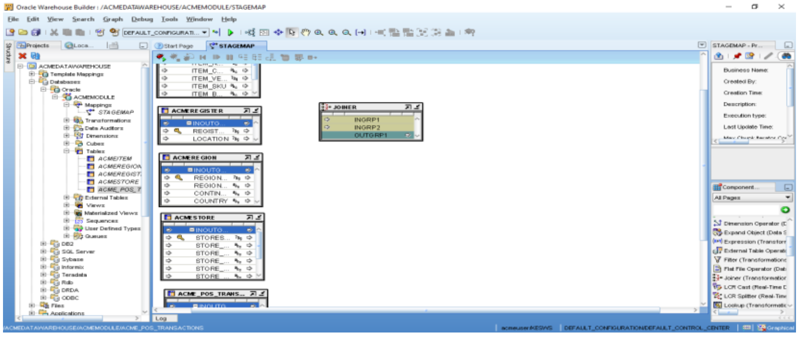
Name: stagemap

2.Drag the acme tables on the left of the canvas

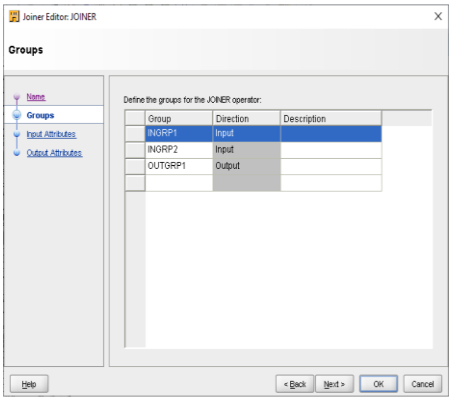




3.Drag Joiner from the component palette and drop in the middle of the canvas



4.Double click Joiner -> Go to Groups -> Add 3 more input groups

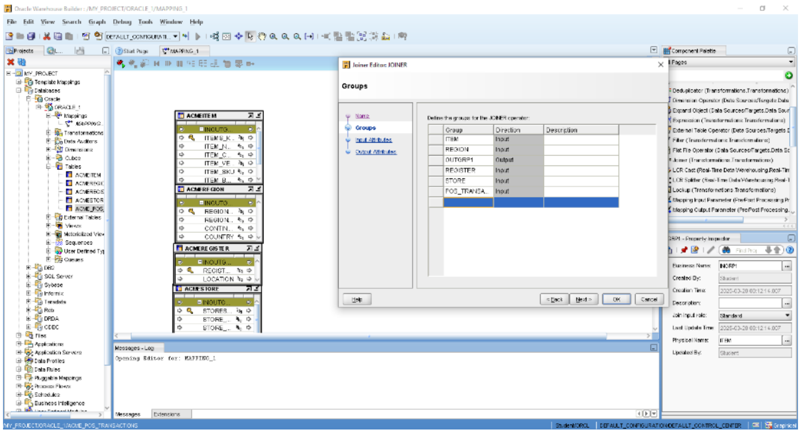


Joiner Editor: JOINER

Groups

Define the groups for the JOINER operator:

Group	Direction	Description
INGRP1	Input	
INGRP2	Input	
OUTGRP1	Output	



Joiner Editor: JOINER

Groups

Define the groups for the JOINER operator:

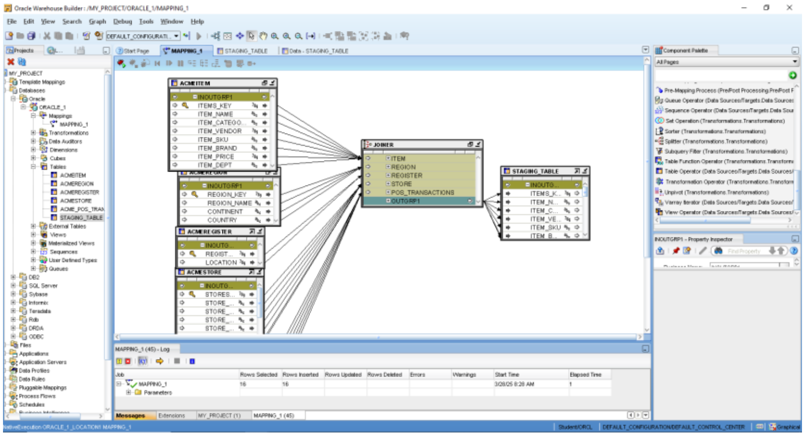
Group	Direction	Description
ITEM	Input	
REGION	Input	
OUTGRP1	Output	
REGISTER	Input	
STORE	Input	
POS_TRANSA...	Input	

```
Opening Editor for: MAPPING_1
```

Oracle Warehouse Builder : /MY_PROJECT/ORACLE_1/MAPPING_1
File Edit View Search Graph Debug Tools Window Help
ACMEITEM
ACMEREGION
ACMEREGISTER
ACMESTORE
Component Palette
Help
< Back
Next >
OK
Cancel

5.Click on OK

6.Map INOUTGRP of tables to the input groups of Joiner respectively

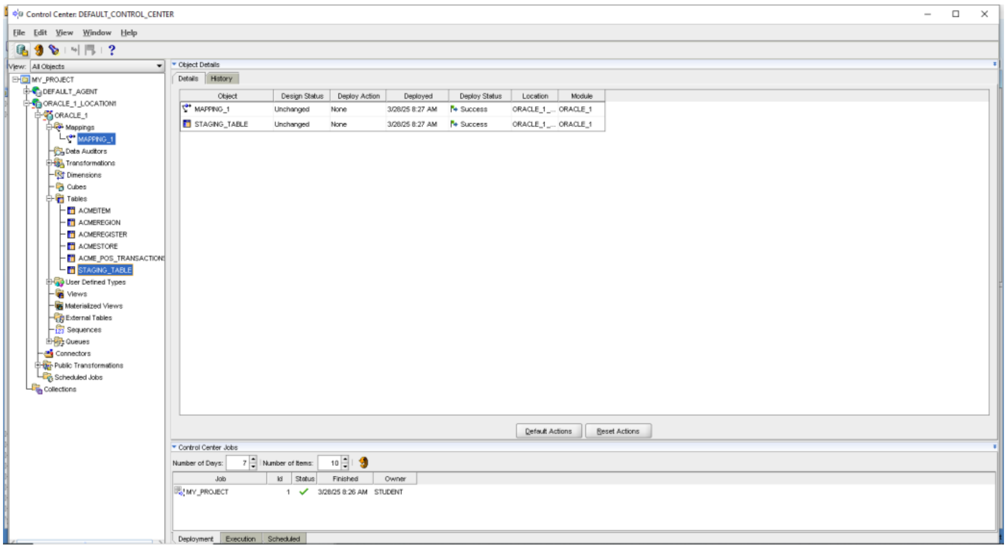


MAPPING_1 (45) - Log

Job	Rows Selected	Rows Inserted	Rows Updated	Rows Deleted	Errors	Warnings	Start Time	Elapsed Time
MAPPING_1	16	16					3/28/25 8:28 AM	1
Parameters								

7.Drag Aggregator from component palette and drop it next to Joiner

8.Connect output of Joiner as input to staging table.



Control Center: DEFAULT_CONTROL_CENTER

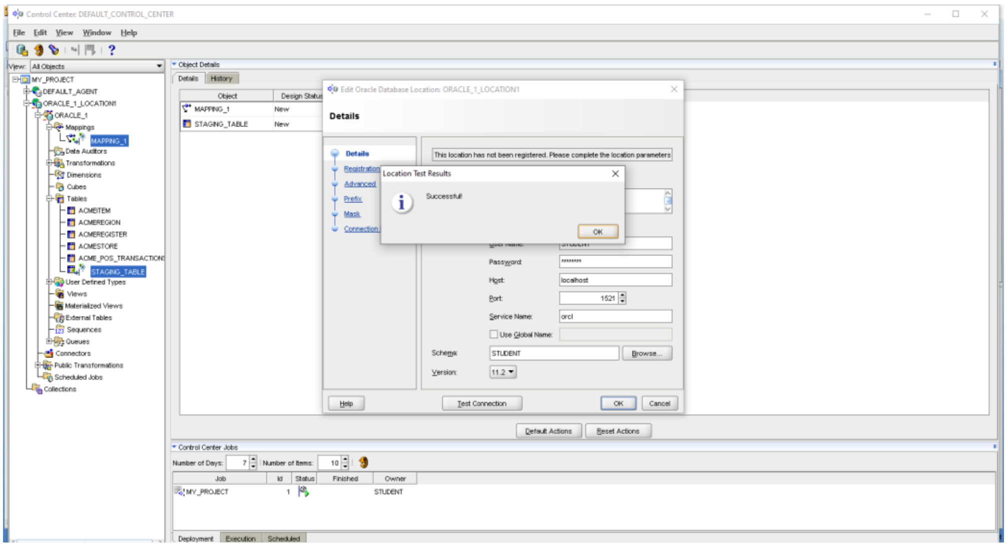
Object Details

Object	Design Status	Deploy Action	Deployed	Deploy Status	Location	Module
MAPPING_1	Unchanged	None	3/28/25 8:27 AM	Success	ORACLE_1_...	ORACLE_1
STAGING_TABLE	Unchanged	None	3/28/25 8:27 AM	Success	ORACLE_1_...	ORACLE_1

Control Center Jobs

Number of Days: 7 Number of Items: 10

Job	Id	Status	Finished	Owner
MY_PROJECT	1	✓	3/28/25 8:26 AM	STUDENT



Edit Oracle Database Location: ORACLE_1_LOCATION1

Details

This location has not been registered. Please complete the location parameters

Location Test Results

Successful!

Password: ********

Host: localhost

Port: 1521

Service Name: orcl

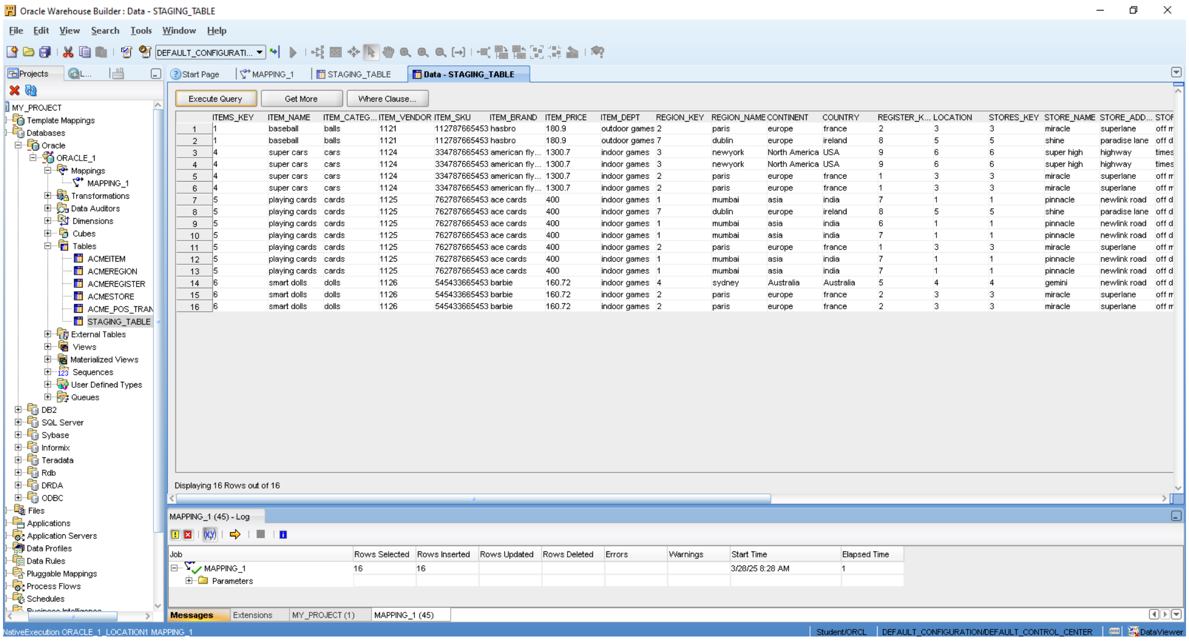
□ Use Global Name:

Schema: STUDENT

Version: 11.2

Object	Design Status
MAPPING_1	New
STAGING_TABLE	New

Job	Id	Status	Finished	Owner
MY_PROJECT	1			STUDENT



Oracle Warehouse Builder : Data - STAGING_TABLE

	ITEMS_KEY	ITEM_NAME	ITEM_CATEG...	ITEM_VENDOR	ITEM_SKU	ITEM_BRAND	ITEM_PRICE	ITEM_DEPT	REGION_KEY	REGION_NAME	CONTINENT	COUNTRY	REGISTER_K...	LOCATION	STORES_KEY	STORE_NAME	STORE_ADD...	STOF
1	1	baseball	balls	1121	112787665453	hasbro	180.9	outdoor games	2	paris	europe	france	2	3	3	miracle	superlane	off m
2	1	baseball	balls	1121	112787665453	hasbro	180.9	outdoor games	7	dublin	europe	ireland	8	5	5	shine	paradise lane	off d
3	4	super cars	cars	1124	334787665453	american fly...	1300.7	indoor games	3	newyork	North America	USA	9	6	6	super high	highway	times
4	4	super cars	cars	1124	334787665453	american fly...	1300.7	indoor games	3	newyork	North America	USA	9	6	6	super high	highway	times
5	4	super cars	cars	1124	334787665453	american fly...	1300.7	indoor games	2	paris	europe	france	1	3	3	miracle	superlane	off m
6	4	super cars	cars	1124	334787665453	american fly...	1300.7	indoor games	2	paris	europe	france	1	3	3	miracle	superlane	off m
7	5	playing cards	cards	1125	762787665453	ace cards	400	indoor games	1	mumbai	asia	india	7	1	1	pinnacle	newlink road	off d
8	5	playing cards	cards	1125	762787665453	ace cards	400	indoor games	7	dublin	europe	ireland	8	5	5	shine	paradise lane	off d
9	5	playing cards	cards	1125	762787665453	ace cards	400	indoor games	1	mumbai	asia	india	6	1	1	pinnacle	newlink road	off d
10	5	playing cards	cards	1125	762787665453	ace cards	400	indoor games	1	mumbai	asia	india	7	1	1	pinnacle	newlink road	off d
11	5	playing cards	cards	1125	762787665453	ace cards	400	indoor games	2	paris	europe	france	1	3	3	miracle	superlane	off m
12	5	playing cards	cards	1125	762787665453	ace cards	400	indoor games	1	mumbai	asia	india	7	1	1	pinnacle	newlink road	off d
13	5	playing cards	cards	1125	762787665453	ace cards	400	indoor games	1	mumbai	asia	india	7	1	1	pinnacle	newlink road	off d
14	6	smart dolls	dolls	1126	545433665453	barbie	160.72	indoor games	4	sydney	Australia	Australia	5	4	4	gemini	newlink road	off d
15	6	smart dolls	dolls	1126	545433665453	barbie	160.72	indoor games	2	paris	europe	france	2	3	3	miracle	superlane	off m
16	6	smart dolls	dolls	1126	545433665453	barbie	160.72	indoor games	2	paris	europe	france	2	3	3	miracle	superlane	off m

Displaying 16 Rows out of 16

MAPPING_1 (45) - Log

Job	Rows Selected	Rows Inserted	Rows Updated	Rows Deleted	Errors	Warnings	Start Time	Elapsed Time
MAPPING_1	16	16					3/28/25 8:28 AM	1
Parameters								

NativeExecution ORACLE_1_LOCATION1 MAPPING_1 Student/ORCL DEFAULT_CONFIGURATION/DEFAULT_CONTROL_CENTER DataViewer

PRACTICAL NO. 2

Aim: Basic exercise on python packages such as numpy, pandas and matplot

Code 1: Num py

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 1. Creating Arrays
print("NumPy Array Creation:")
arr_zeros = np.zeros(5)
arr_ones = np.ones(5)
arr_range = np.arange(0, 10, 2)
arr_linspace = np.linspace(0, 10, 5)

print("Zeros Array:", arr_zeros)
print("Ones Array:", arr_ones)
print("Range Array:", arr_range)
print("Linspace Array:", arr_linspace)

# 2. Array Operations
print("\nNumPy Array Operations:")
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

# Element-wise operations
sum_arr = arr1 + arr2
product_arr = arr1 * arr2

print("Array 1:", arr1)
print("Array 2:", arr2)
print("Sum of Arrays:", sum_arr)
print("Product of Arrays:", product_arr)

# 3. Shape and Resizing
print("\nNumPy Array Resizing and
Shape:")
arr_reshaped = np.arange(12).reshape(3, 4)
print("Original Array (0 to 11):",
np.arange(12))
print("Reshaped Array (3x4):\n",
arr_reshaped)
print("Shape of Reshaped Array:",
arr_reshaped.shape)

# 4. Statistical Operations
print("\nNumPy Statistical Operations:")
mean_arr = np.mean(arr_reshaped)
std_arr = np.std(arr_reshaped)
```

Code 2: Num py, Pandas, Matplot

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 1. NumPy Operations
print("NumPy Operations:")
arr = np.array([10, 20, 30, 40, 50])
print("Original Array:", arr)
print("Array Mean:", np.mean(arr))
print("Array Sum:", np.sum(arr))
print("Array Squared:", arr ** 2)  #
Element-wise squaring

# 2. Pandas DataFrame
print("\nPandas DataFrame Example:")
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'Salary': [50000, 60000, 70000]}
df = pd.DataFrame(data)
print(df)


# 3. Matplotlib Plot
print("\nPlotting Data using Matplotlib...")
x = np.linspace(0, 10, 5)
y = x ** 2  # Squaring each value

plt.plot(x, y, marker='o', linestyle='-',
color='b', label='y = x^2')
plt.xlabel("X values")
plt.ylabel("Y values")
plt.title("Simple Plot using Matplotlib")
plt.legend()
plt.grid(True)
plt.show()
```

Output:

```
NumPy Operations:
Original Array: [10 20 30 40 50]
Array Mean: 30.0
Array Sum: 150
Array Squared: [ 100  400  900 1600 2500]

Pandas DataFrame Example:
    Name  Age  Salary
0   Alice   25   50000
```

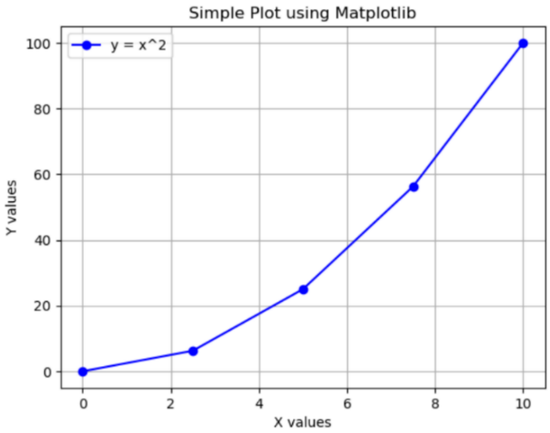
```
print("Mean of Array:", mean_arr)
print("Standard Deviation of Array:",
std_arr)
```

Output:

```
NumPy Array Creation:
Zeros Array: [0. 0. 0. 0. 0.]
Ones Array: [1. 1. 1. 1. 1.]
Range Array: [0 2 4 6 8]
Linspace Array: [ 0.   2.5  5.   7.5 10. ]

NumPy Array Operations:
Array 1: [1 2 3]
Array 2: [4 5 6]
Sum of Arrays: [5 7 9]
Product of Arrays: [ 4 10 18]

NumPy Array Resizing and Shape:
Original Array (0 to 11): [ 0  1  2  3  4  5  6
7  8  9 10 11]
Reshaped Array (3x4):
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

Shape of Reshaped Array: (3, 4)
NumPy Statistical Operations:
Mean of Array: 5.5
Standard Deviation of Array:
3.452052529534663
```

```
1      Bob   30   60000
2  Charlie   35   70000
```

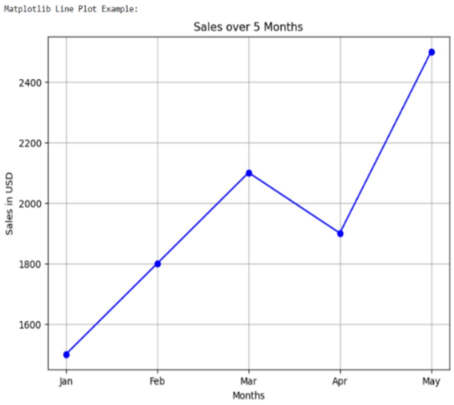
Plotting Data using Matplotlib



Code 3: Matplotlib

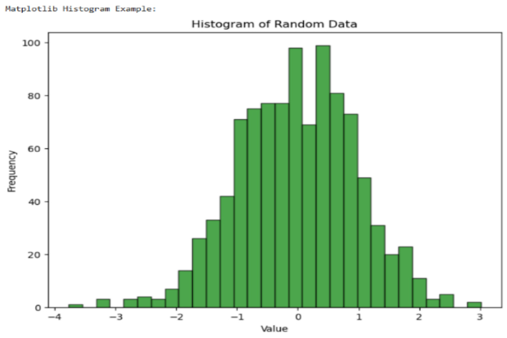
```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 1. Line Plot
print("\nMatplotlib Line Plot Example:")
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']
sales = [1500, 1800, 2100, 1900, 2500]

plt.figure(figsize=(8, 6))
plt.plot(months, sales, marker='o', color='b')
plt.title('Sales over 5 Months')
plt.xlabel('Months')
plt.ylabel('Sales in USD')
plt.grid(True)
```

OUTPUT:-

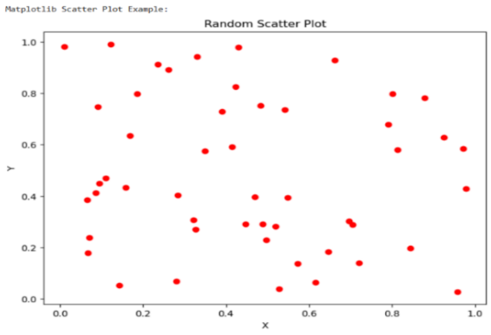


```
Matplotlib Line Plot Example:
```

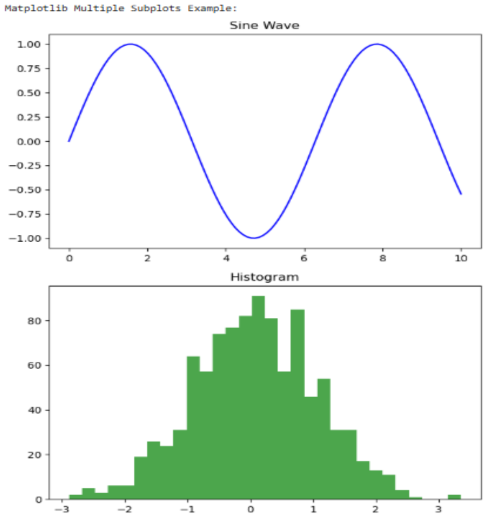
```
plt.show()

# 2. Histogram
print("\nMatplotlib Histogram Example:")
random_data = np.random.randn(1000)

plt.figure(figsize=(8, 6))
plt.hist(random_data, bins=30,
edgecolor='black', color='g', alpha=0.7)
plt.title('Histogram of Random Data')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()

# 3. Scatter Plot
print("\nMatplotlib Scatter Plot Example:")
x = np.random.rand(50)
y = np.random.rand(50)

plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='r')
plt.title('Random Scatter Plot')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()

# 4. Multiple Subplots
print("\nMatplotlib Multiple Subplots
Example:")
x = np.linspace(0, 10, 100)

plt.figure(figsize=(12, 8))

# Subplot 1: Line Plot
plt.subplot(2, 2, 1)
plt.plot(x, np.sin(x), color='b')
plt.title('Sine Wave')

# Subplot 2: Bar Plot
plt.subplot(2, 2, 2)
plt.bar(['A', 'B', 'C', 'D'], [4, 7, 3, 6], color='y')
plt.title('Bar Plot')

# Subplot 3: Histogram
plt.subplot(2, 2, 3)
plt.hist(np.random.randn(1000), bins=30,
color='g', alpha=0.7)
plt.title('Histogram')

# Subplot 4: Scatter Plot
```



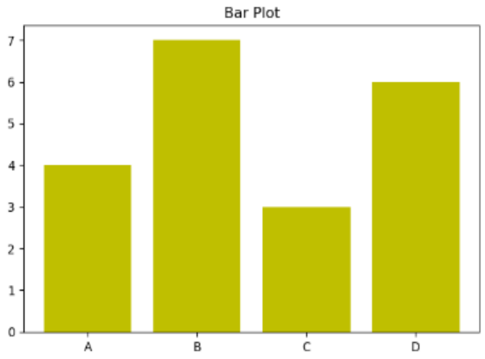
```
Matplotlib Histogram Example:
```



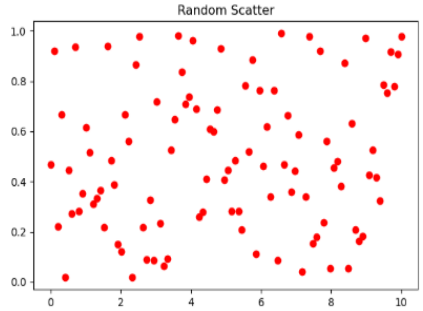
```
Matplotlib Scatter Plot Example:
```



```
Matplotlib Multiple Subplots Example:
```



```
plt.subplot(2, 2, 4)
plt.scatter(x, np.random.rand(100), color='r')
plt.title('Random Scatter')

plt.tight_layout()
plt.show()
```



Code 4:

```
import pandas as pd
import numpy as np
# Create DataFrame
df = pd.DataFrame({'Name': ['Alice', 'Bob',
'Charlie', 'David'],
                'Age': [24, 27, 22, 32],
                'City': ['New York', 'Los Angeles',
'Chicago', 'Houston']})

print(df.head(2), df.info(), df.describe())  #
Display first rows, info, and stats

# Handling Missing Values
df.loc[2, 'Age'] = np.nan  # Introduce NaN
print(df.dropna(), df.fillna({'Age':
df['Age'].mean()}))  # Drop & Fill NaN

# Statistical Operations
print(f"Mean: {df['Age'].mean()}, Median:
{df['Age'].median()}, Mode:
{df['Age'].mode()[0]}")

# Grouping Data
df2 = pd.DataFrame({'Department': ['HR', 'IT',
'HR', 'IT'],
                'Salary': [60000, 80000, 65000,
85000]})
print(df2.groupby('Department')['Salary'].mean())
# Grouped Mean Salary
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    4 non-null      object
 1   Age     4 non-null      int64
 2   City    4 non-null      object
dtypes: int64(1), object(2)
memory usage: 228.0+ bytes
    Name  Age         City
0  Alice   24     New York
1    Bob   27  Los Angeles None
Age
count   4.000000
mean   26.250000
std     4.349329
min    22.000000
25%    23.500000
50%    25.500000
75%    28.250000
max    32.000000
Name   Age         City
0  Alice  24.0     New York
1    Bob  27.0  Los Angeles
3  David  32.0      Houston       Name
Age         City
0    Alice  24.000000     New York
1      Bob  27.000000  Los Angeles
2  Charlie  27.666667      Chicago
3    David  32.000000      Houston
Mean: 27.666666666666668, Median:
27.0, Mode: 24.0
Department
HR    62500.0
IT    82500.0
Name: Salary, dtype: float64
```

PRACTICAL NO. 3

Aim: Given a dataset. Write a program to compute the mean, median, mode.

Code 1:

```
import pandas as pd
import statistics

# Load Speed dataset (example data)
data = {'Speed': [55, 60, 62, 58, 57, 63, 65,
60, 58, 62, 61, 64, 59, 55, 60, 61, 62, 63, 64,
58]}
df = pd.DataFrame(data)

def compute_statistics(data):
   mean_value = data.mean()
   median_value = data.median()
   mode_value = data.mode().iloc[0]  #
Mode may return multiple values, take the
first one

   return mean_value, median_value,
mode_value

# Compute statistics
mean, median, mode =
compute_statistics(df['Speed'])

print(f"Mean Speed: {mean}")
print(f"Median Speed: {median}")
print(f"Mode Speed: {mode}")
```

Output:

```
Mean Speed: 60.35
Median Speed: 60.5
Mode Speed: 58
```

Code 2:

```
import pandas as pd
import statistics
from sklearn.datasets import load_iris

def compute_statistics(data):
   mean_value = data.mean()
   median_value = data.median()
   mode_value = data.mode().iloc[0]  #
Mode may return multiple values, take the
first one

   return mean_value, median_value,
mode_value

# Load Iris dataset
iris = load_iris()
df = pd.DataFrame(iris.data,
columns=iris.feature_names)

# Select a specific column for analysis
column_name = 'sepal length (cm)'  #
Example column
data = df[column_name]

mean, median, mode =
compute_statistics(data)

print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Mode: {mode}")
```

Output:

```
Mean: 5.843333333333334
Median: 5.8
Mode: 5.0
```

PRACTICAL NO. 4

Aim: Given a dataset. Write a program to compute standard deviation, covariance, co-relation between pair of attributes.

Code:

```
import numpy as np
import pandas as pd
# Sample dataset
data = pd.DataFrame({
   "Height": [170, 165, 180, 175, 160],
   "Weight": [65, 70, 75, 80, 60],
   "Age": [25, 30, 35, 28, 22]
})

def compute_standard_deviation(data):
   """ Compute standard deviation of each attribute """
   return data.std()
def compute_covariance_matrix(data):
   """ Compute covariance matrix """
   return data.cov()
def compute_correlation_matrix(data):
   """ Compute correlation matrix """
   return data.corr()

# Display results
print("\nStandard Deviation of each attribute:")
print(compute_standard_deviation(data))
print("\nCovariance Matrix:")
print(compute_covariance_matrix(data))
print("\nCorrelation Matrix:")
print(compute_correlation_matrix(data))
```

Output:

```
Standard Deviation of each attribute:
Height    7.905694
Weight    7.905694
Age       4.949747
dtype:  float64
Covariance Matrix:
           Height  Weight  Age
Height   62.5      50.0     30.0
Weight   50.0      62.5     27.5
Age        30.0     27.5     24.5

Correlation Matrix:
           Height      Weight      Age
Height  1.000000   0.800000   0.766652
Weight  0.800000   1.000000   0.702764
Age       0.766652   0.702764   1.000000
```

PRACTICAL NO. 5

Aim: Write a program to implement data pre-processing techniques

Code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection
import train_test_split
from sklearn.preprocessing
import StandardScaler, LabelEncoder
from sklearn.impute import
SimpleImputer from sklearn.datasets
import load_iris

# Load dataset
data = load_iris()
df = pd.DataFrame(data.data,
columns=data.feature_names)
df['target'] = data.target

# Introduce missing values for
df.iloc[5:10, 1] = np.nan
# Handle missing values
df.iloc[:, 1:5] =
SimpleImputer(strategy='mean').
fit_transform(df.iloc[:, 1:5])
# Encode categorical data (if applicable)
df['target'] =
LabelEncoder().fit_transform(df['target'])
# Feature scaling
scaler = StandardScaler()
df.iloc[:, :-1] =
scaler.fit_transform(df.iloc[:, :-1])
# Train-Test Split
X_train, X_test, y_train, y_test =
train_test_split(df.drop('target', axis=1),
df['target'], test_size=0.3,
random_state=42)
# Output sample data
print("Data after Preprocessing:")
print(df.head())

print("\nTrain-Test Split:")
print(f"Training
Features:\n{X_train.head()}")
print(f"Testing
Features:\n{X_test.head()}")
```

Output:

```
  sepal length (cm)  sepal width (cm)  petal
length (cm)  petal width (cm)  \
0      -0.900681      1.060054      -
1.340227      -1.315444
1      -1.143017      -0.111500      -
1.340227      -1.315444
2      -1.385353      0.357122      -
1.397064      -1.315444
3      -1.506521      0.122811      -
1.283389      -1.315444
4      -1.021849      1.294365      -
1.340227      -1.315444
  target
0    0
1    0
2    0
3    0
4    0
Train-Test Split:
Training Features:
   sepal length (cm)  sepal width (cm)  petal
length (cm)  petal width (cm)
81       -0.416010      -1.517364      -
0.032966      -0.262387
133       0.553333      -0.580121
0.762758       0.395774
137       0.674501       0.122811
0.990108       0.790671
75        0.916837      -0.111500
0.364896       0.264142
109       1.643844       1.294365
1.331133       1.712096
Testing Features:
   sepal length (cm)   sepal width (cm)  petal
length (cm)  petal width (cm)
73        0.310998      -0.580121
0.535409       0.000878
18       -0.173674       1.762986      -
1.169714      -1.183812
78        0.189830      -0.345810
0.421734       0.395774
76        1.159173      -0.580121
0.592246       0.264142
```

PRACTICAL NO. 6

Aim: Write a program to implement data transformation using different normalization techniques

Code:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler,
StandardScaler, MaxAbsScaler, RobustScaler

# Generating a sample dataset
def generate_sample_data():
    # Create a sample dataframe with some random data
    data = {
        'Feature1': [10, 20, 30, 40, 50],
        'Feature2': [5, 10, 15, 20, 25],
        'Feature3': [100, 200, 300, 400, 500]
    }
    df = pd.DataFrame(data)
    return df

# Apply different normalization techniques
def normalize_data(df):
    # Min-Max Normalization (scales data to [0, 1])
    min_max_scaler = MinMaxScaler()
    min_max_normalized = min_max_scaler.fit_transform(df)

    # Z-Score Normalization (Standardization)
    standard_scaler = StandardScaler()
    z_score_normalized = standard_scaler.fit_transform(df)

    # Max Abs Normalization (scales data by dividing by the
maximum absolute value)
    max_abs_scaler = MaxAbsScaler()
    max_abs_normalized = max_abs_scaler.fit_transform(df)

    # Robust Scaler (uses median and interquartile range)
    robust_scaler = RobustScaler()
    robust_normalized = robust_scaler.fit_transform(df)

    # Create DataFrames for each normalization technique to
display
    df_min_max = pd.DataFrame(min_max_normalized,
columns=df.columns)
    df_z_score = pd.DataFrame(z_score_normalized,
columns=df.columns)
    df_max_abs = pd.DataFrame(max_abs_normalized,
columns=df.columns)
```

Output:

```
Original Data:
  Feature1  Feature2
Feature3
0      10        5      100
1      20       10      200
2      30       15      300
3      40       20      400
4      50       25      500

Min-Max Normalized
Data:
  Feature1  Feature2
Feature3
0     0.00      0.00
0.00
1     0.25      0.25
0.25
2     0.50      0.50
0.50
3     0.75      0.75
0.75
4     1.00      1.00
1.00

Z-Score Normalized
Data:
  Feature1  Feature2
Feature3
0 -1.414214 -1.414214 -
1.414214
1 -0.707107 -0.707107 -
0.707107
2  0.000000  0.000000
0.000000
3  0.707107  0.707107
0.707107
4  1.414214  1.414214
1.414214

Max Abs Normalized
Data:
  Feature1  Feature2
Feature3
```

```
    df_robust = pd.DataFrame(robust_normalized,
columns=df.columns)

    return df_min_max, df_z_score, df_max_abs, df_robust

# Displaying the results
def display_results(df_original, df_min_max, df_z_score,
df_max_abs, df_robust):
    print("Original Data:")
    print(df_original)
    print("\nMin-Max Normalized Data:")
    print(df_min_max)
    print("\nZ-Score Normalized Data:")
    print(df_z_score)
    print("\nMax Abs Normalized Data:")
    print(df_max_abs)
    print("\nRobust Scaler Normalized Data:")
    print(df_robust)

if __name__ == "__main__":
    # Generate sample data
    df_original = generate_sample_data()

    # Normalize data using different techniques
    df_min_max, df_z_score, df_max_abs, df_robust =
normalize_data(df_original)

    # Display the results
    display_results(df_original, df_min_max, df_z_score,
df_max_abs, df_robust)
```

```
0    0.2     0.2     0.2
1    0.4     0.4     0.4
2    0.6     0.6     0.6
3    0.8     0.8     0.8
4    1.0     1.0     1.0

Robust Scaler
Normalized Data:
  Feature1  Feature2
Feature3
0   -1.0    -1.0    -1.0
1   -0.5    -0.5    -0.5
2    0.0     0.0     0.0
3    0.5     0.5     0.5
4    1.0     1.0     1.0
```

PRACTICAL NO. 7

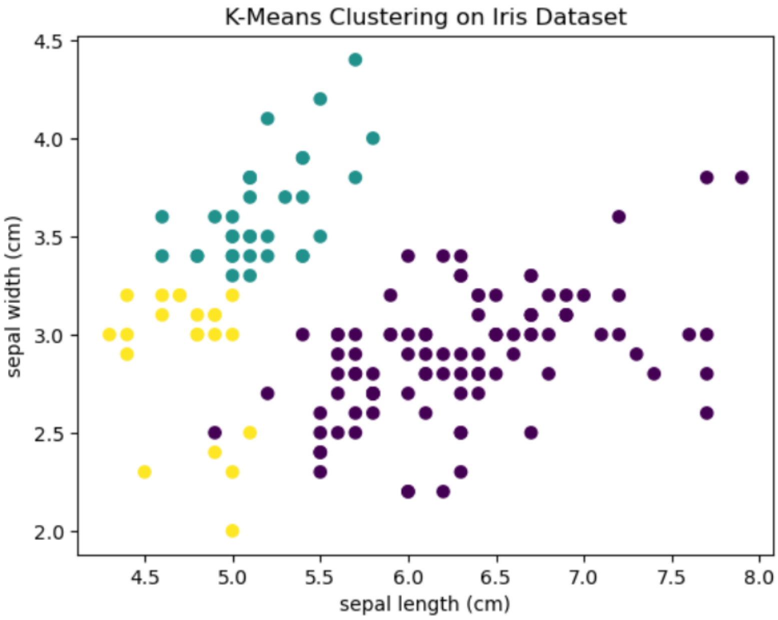
Aim: Write a program to implement K-Mean clustering algorithm. Select your own dataset to test the program

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler

# Load Iris dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Standardize the data
scaler = StandardScaler()
```

```
df_scaled = scaler.fit_transform(df)

# Apply K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(df_scaled)

# Plot the clusters
plt.scatter(df.iloc[:, 0], df.iloc[:, 1], c=df['Cluster'], cmap='viridis')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.title('K-Means Clustering on Iris Dataset')
plt.show()
```

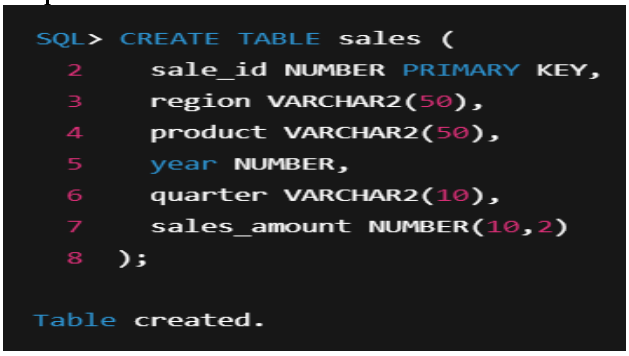
Output:



PRACTICAL NO. 8

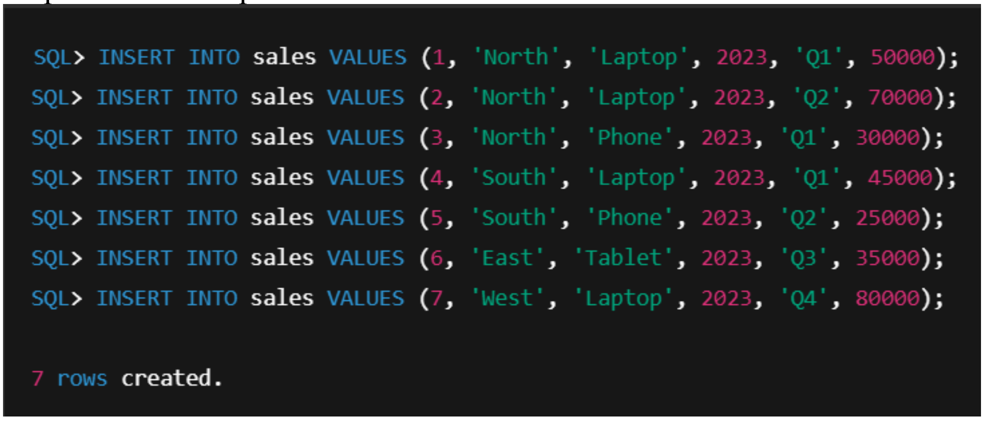
Aim: OLAP Operations using PL/SQL
Performed in: Oracle SQL*Plus

Step 1: Create the Sales Table



```
SQL> CREATE TABLE sales (
  2     sale_id NUMBER PRIMARY KEY,
  3     region VARCHAR2(50),
  4     product VARCHAR2(50),
  5     year NUMBER,
  6     quarter VARCHAR2(10),
  7     sales_amount NUMBER(10,2)
  8  );

Table created.
```

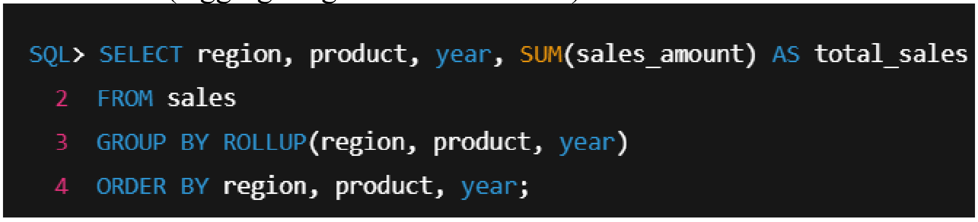
Step 2: Insert Sample Data



```
SQL> INSERT INTO sales VALUES (1, 'North', 'Laptop', 2023, 'Q1', 50000);
SQL> INSERT INTO sales VALUES (2, 'North', 'Laptop', 2023, 'Q2', 70000);
SQL> INSERT INTO sales VALUES (3, 'North', 'Phone', 2023, 'Q1', 30000);
SQL> INSERT INTO sales VALUES (4, 'South', 'Laptop', 2023, 'Q1', 45000);
SQL> INSERT INTO sales VALUES (5, 'South', 'Phone', 2023, 'Q2', 25000);
SQL> INSERT INTO sales VALUES (6, 'East', 'Tablet', 2023, 'Q3', 35000);
SQL> INSERT INTO sales VALUES (7, 'West', 'Laptop', 2023, 'Q4', 80000);

7 rows created.
```

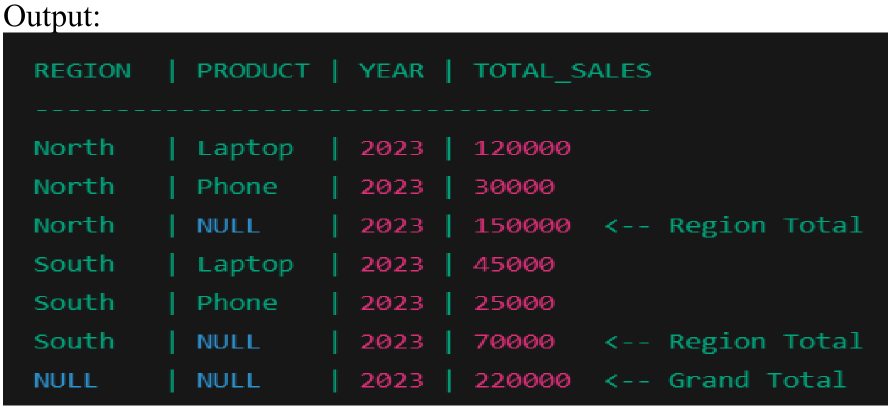
Performing OLAP Operations in SQL*Plus

1. ROLLUP (Aggregating Hierarchical Data)



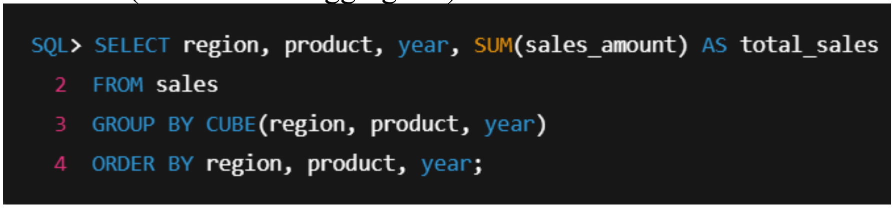
```
SQL> SELECT region, product, year, SUM(sales_amount) AS total_sales
  2  FROM sales
  3  GROUP BY ROLLUP(region, product, year)
  4  ORDER BY region, product, year;
```

Output:



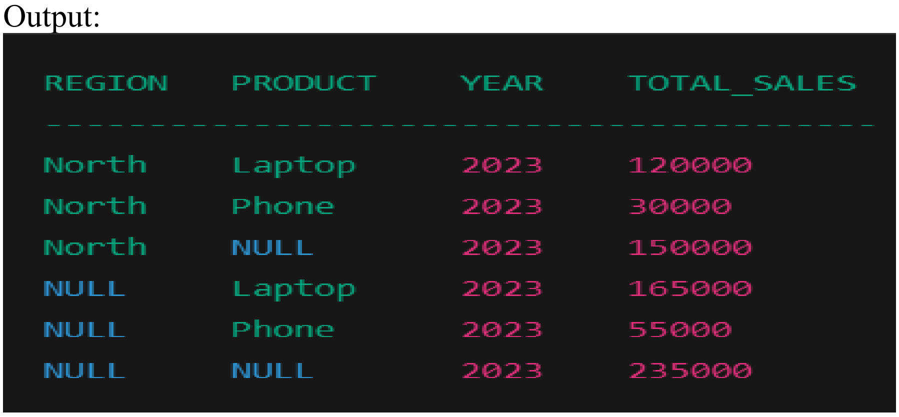
```
REGION   | PRODUCT | YEAR | TOTAL_SALES
---------------------------------------
North    | Laptop  | 2023 | 120000
North    | Phone   | 2023 | 30000
North    | NULL    | 2023 | 150000  <-- Region Total
South    | Laptop  | 2023 | 45000
South    | Phone   | 2023 | 25000
South    | NULL    | 2023 | 70000   <-- Region Total
NULL     | NULL    | 2023 | 220000  <-- Grand Total
```

2. CUBE (All Possible Aggregates)



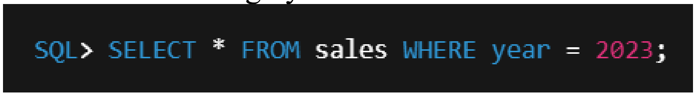
```
SQL> SELECT region, product, year, SUM(sales_amount) AS total_sales
  2  FROM sales
  3  GROUP BY CUBE(region, product, year)
  4  ORDER BY region, product, year;
```

Output:

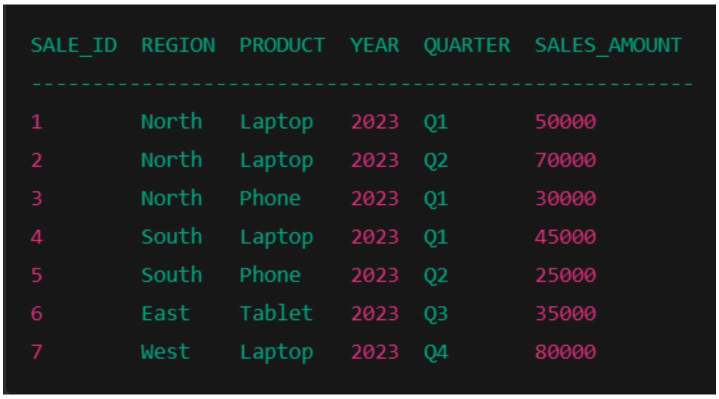


```
REGION    PRODUCT     YEAR     TOTAL_SALES
----------------------------------------
North     Laptop      2023     120000
North     Phone       2023     30000
North     NULL        2023     150000
NULL      Laptop      2023     165000
NULL      Phone       2023     55000
NULL      NULL        2023     235000
```

3. SLICE - Filtering by Year

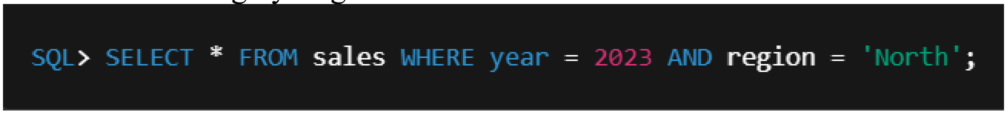


```
SQL> SELECT * FROM sales WHERE year = 2023;
```



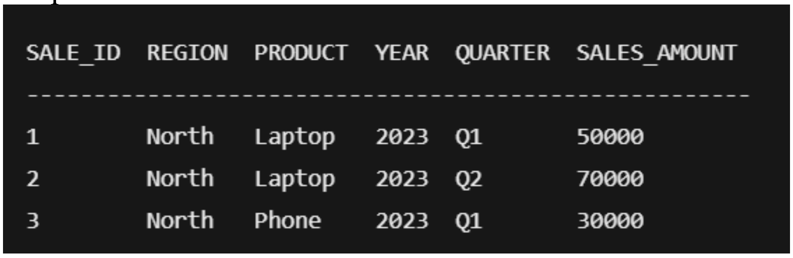
```
SALE_ID  REGION  PRODUCT  YEAR  QUARTER  SALES_AMOUNT
------------------------------------------------------
1        North   Laptop   2023  Q1       50000
2        North   Laptop   2023  Q2       70000
3        North   Phone    2023  Q1       30000
4        South   Laptop   2023  Q1       45000
5        South   Phone    2023  Q2       25000
6        East    Tablet   2023  Q3       35000
7        West    Laptop   2023  Q4       80000
```

4. DICE - Filtering by Region & Year



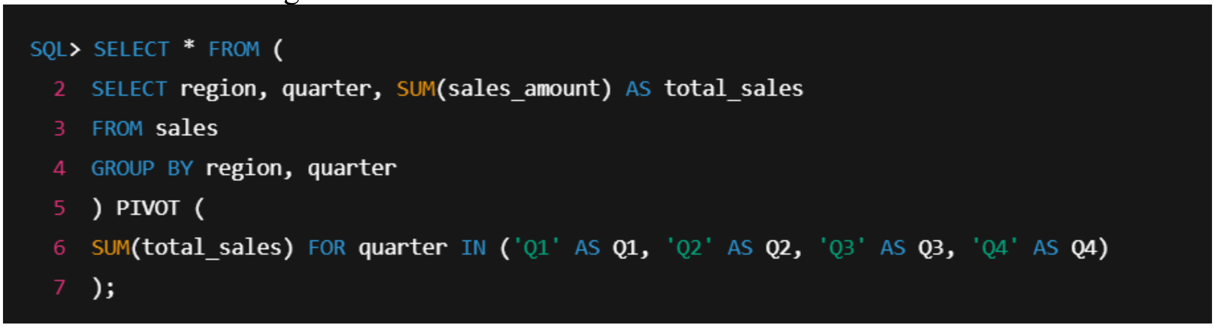
```
SQL> SELECT * FROM sales WHERE year = 2023 AND region = 'North';
```

Output:



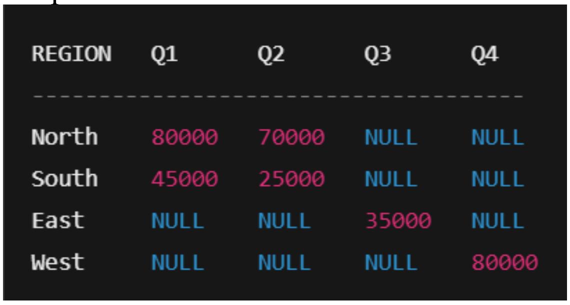
```
SALE_ID  REGION  PRODUCT  YEAR  QUARTER  SALES_AMOUNT
------------------------------------------------------
1        North   Laptop   2023  Q1       50000
2        North   Laptop   2023  Q2       70000
3        North   Phone    2023  Q1       30000
```

5. PIVOT - Converting Rows into Columns



```
SQL> SELECT * FROM (
  2  SELECT region, quarter, SUM(sales_amount) AS total_sales
  3  FROM sales
  4  GROUP BY region, quarter
  5  ) PIVOT (
  6  SUM(total_sales) FOR quarter IN ('Q1' AS Q1, 'Q2' AS Q2, 'Q3' AS Q3, 'Q4' AS Q4)
  7  );
```

Output:



```
REGION   Q1      Q2      Q3      Q4
------------------------------------
North    80000   70000   NULL    NULL
South    45000   25000   NULL    NULL
East     NULL    NULL    35000   NULL
West     NULL    NULL    NULL    80000
```