LAB - 08

Data Handling using Warehouse

Aim : Handling of data in a multidimensional view-point using the concept of data warehouses. To be aware of a form of analytics also known as data mining.

Description:

The primary level of analytics started with aggregation functionalities. i.e find maximum, minimum, average, mode, mean, median, etc from mathematics. The flat/relational storage supports transactions and is able to perform aggregation like primitives. The need for a multidimensional view point of summarized data and being able to perform operations on the same runtime like slicing, dicing, drill down, etc is accomplished via data warehousing and data mining. Many legacy products are having their own establishment and it is important to be aware of the same. Know that Data Warehouse is in itself a separate data storage in parallel with raw data storage. It is built and maintained (nightly/weekly/monthly/yearly) as per requirements. This again builds the base of data handling of modern times which are a lot advanced and sustainable to meet Millennials needs.

During the development, the following are types and purposes of analytics observed.
It also has of Industrial Revolutions i.e. Industry 1.0. 2.0, etc, community wise.
HindSight to Insight to ForeSight

- What happened? (Descriptive Analytics)
- Why did it happen? (Diagnostic Analytics)
- What will happen? (Predictive Analytics)
- How can we make it happen? (Prescriptive Analytics)
- The most recent and advanced is "COGNITIVE ANALYTICS", using AI, ML, DL and such advanced technologies learning and improving to infer better and better and act autonomously.

Methodology:

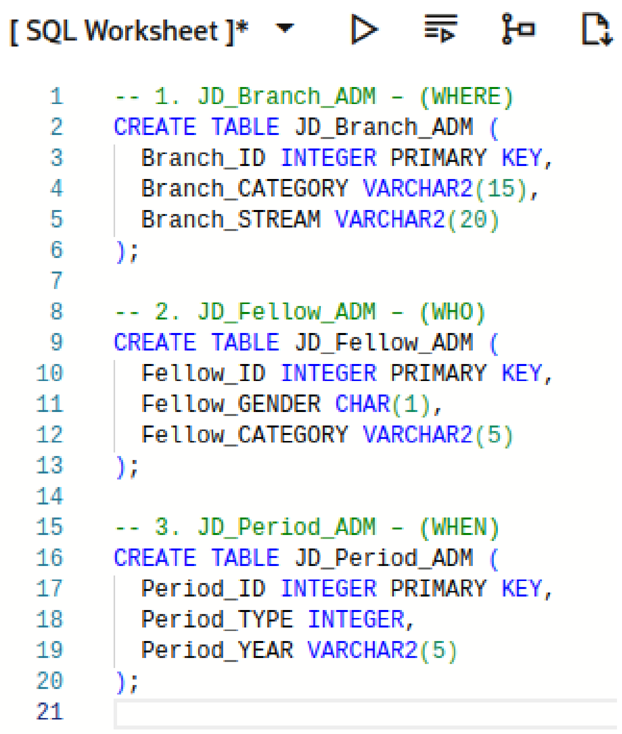
Understand the multi-dimensionality and utilize the concept of star schema to create relations within Oracle/MySQL like relational databases. Understand clearly that either from internal tables or external data sources i.e. .csv,.xml, web service responses, the establishment of data warehousing is carried out after running some or other aggregation like queries.

Example:

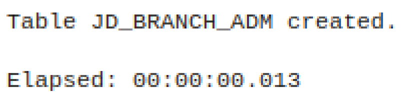
Dimensional modeling for business process of “STUDENT ADMISSION FOR GRADUATION, AFTER 12TH SCIENCE”

Here, I’m working on https://livesql.oracle.com/

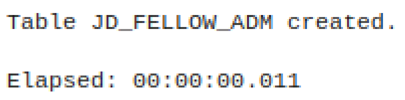
Step 1: Create Dimension Tables



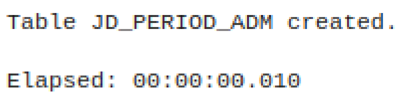
```
-- 1. JD_Branch_ADM - (WHERE)
CREATE TABLE JD_Branch_ADM (
  Branch_ID INTEGER PRIMARY KEY,
  Branch_CATEGORY VARCHAR2(15),
  Branch_STREAM VARCHAR2(20)
);

-- 2. JD_Fellow_ADM - (WHO)
CREATE TABLE JD_Fellow_ADM (
  Fellow_ID INTEGER PRIMARY KEY,
  Fellow_GENDER CHAR(1),
  Fellow_CATEGORY VARCHAR2(5)
);

-- 3. JD_Period_ADM - (WHEN)
CREATE TABLE JD_Period_ADM (
  Period_ID INTEGER PRIMARY KEY,
  Period_TYPE INTEGER,
  Period_YEAR VARCHAR2(5)
);
```



```
Table JD_BRANCH_ADM created.

Elapsed: 00:00:00.013
```

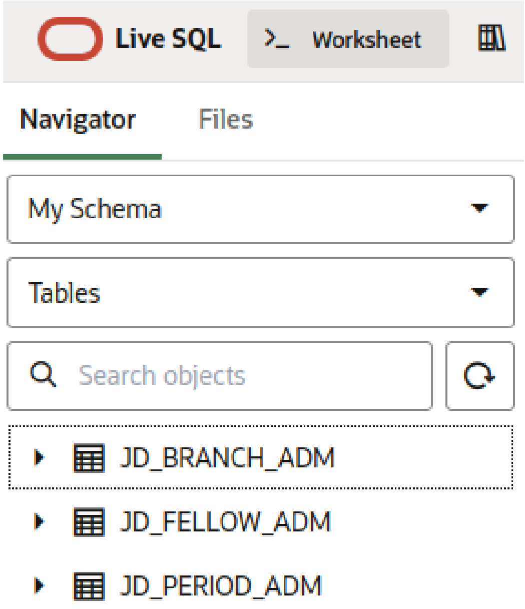


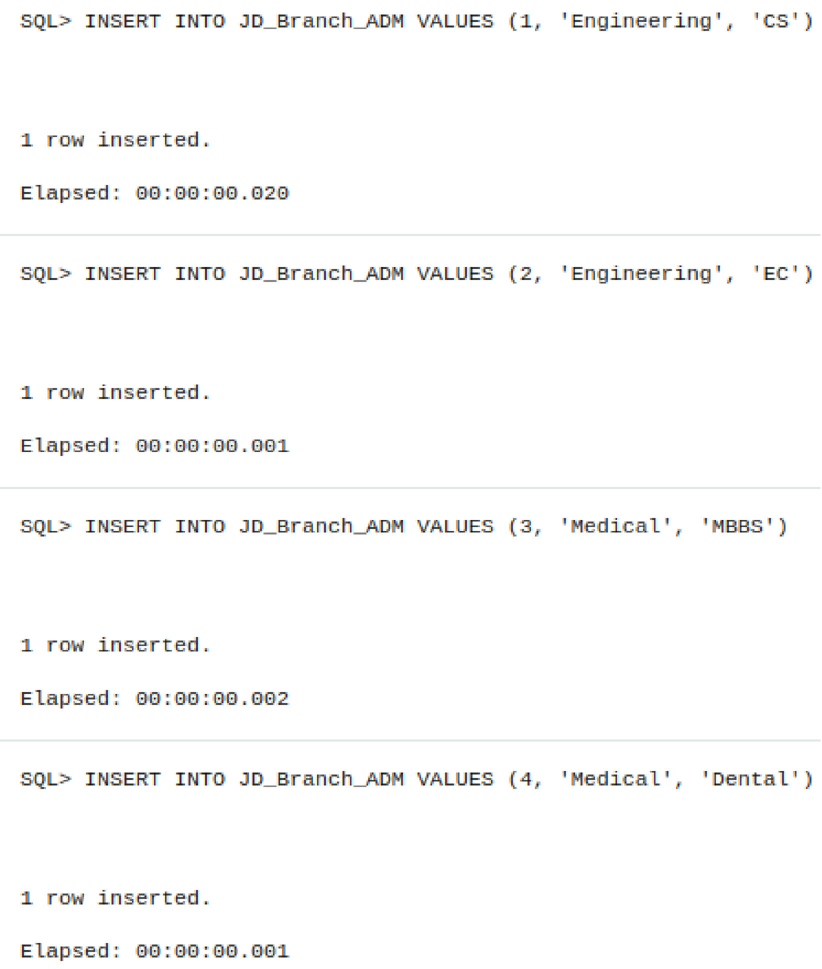
```
Table JD_FELLOW_ADM created.

Elapsed: 00:00:00.011
```



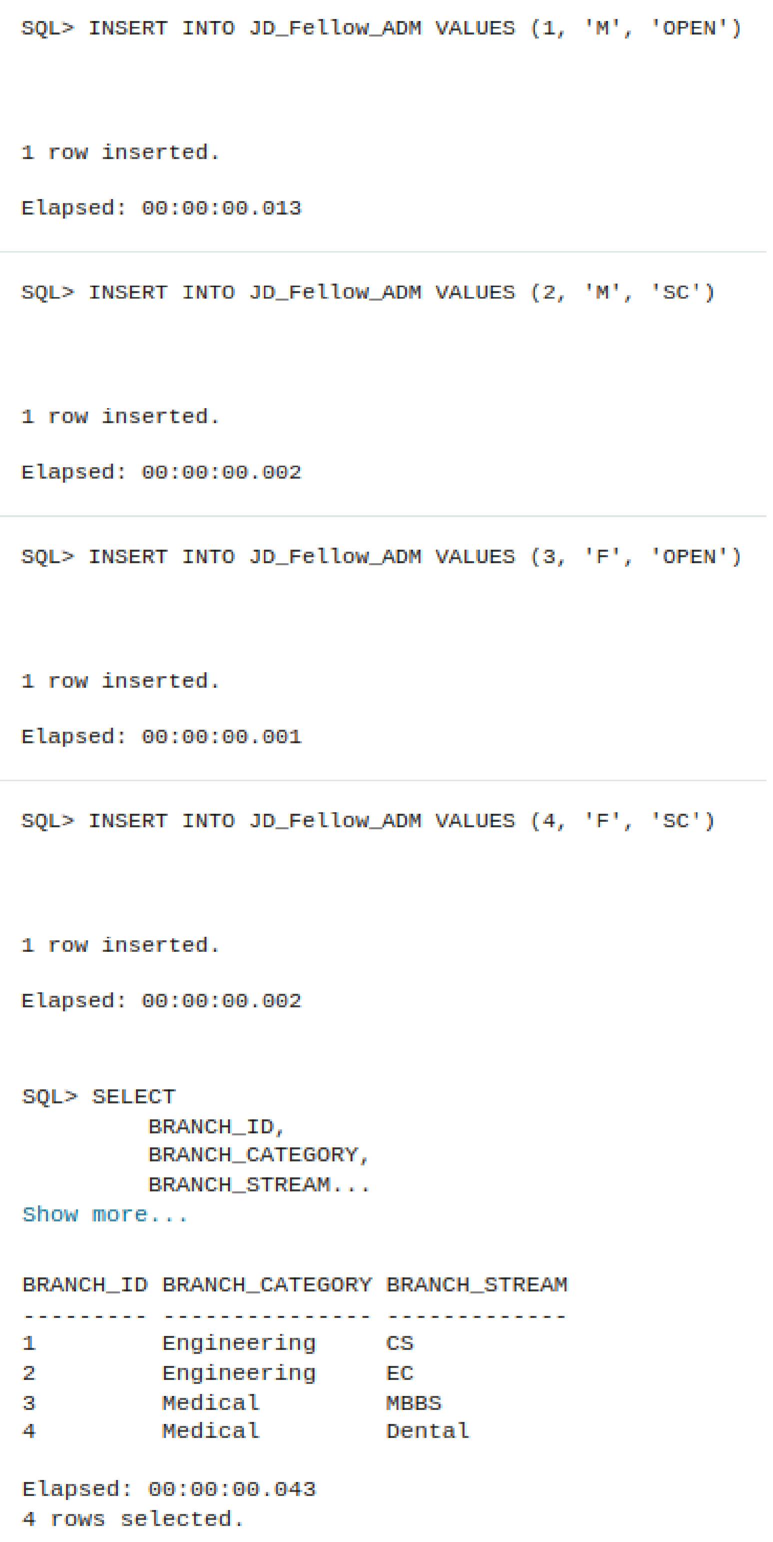
```
Table JD_PERIOD_ADM created.

Elapsed: 00:00:00.010
```

Step 2: Insert Sample Data into Dimension Tables





```
SQL> INSERT INTO JD_Branch_ADM VALUES (1, 'Engineering', 'CS')


1 row inserted.

Elapsed: 00:00:00.020

SQL> INSERT INTO JD_Branch_ADM VALUES (2, 'Engineering', 'EC')


1 row inserted.

Elapsed: 00:00:00.001

SQL> INSERT INTO JD_Branch_ADM VALUES (3, 'Medical', 'MBBS')


1 row inserted.

Elapsed: 00:00:00.002

SQL> INSERT INTO JD_Branch_ADM VALUES (4, 'Medical', 'Dental')


1 row inserted.

Elapsed: 00:00:00.001
```



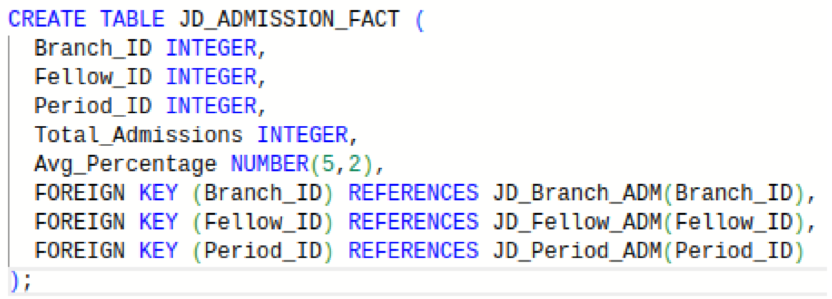
```
SQL> INSERT INTO JD_Fellow_ADM VALUES (1, 'M', 'OPEN')


1 row inserted.

Elapsed: 00:00:00.013
```

```
SQL> INSERT INTO JD_Fellow_ADM VALUES (2, 'M', 'SC')


1 row inserted.

Elapsed: 00:00:00.002
```

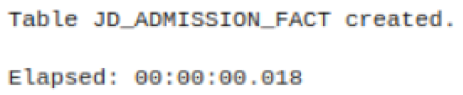
```
SQL> INSERT INTO JD_Fellow_ADM VALUES (3, 'F', 'OPEN')


1 row inserted.

Elapsed: 00:00:00.001
```

```
SQL> INSERT INTO JD_Fellow_ADM VALUES (4, 'F', 'SC')


1 row inserted.

Elapsed: 00:00:00.002


SQL> SELECT
         BRANCH_ID,
         BRANCH_CATEGORY,
         BRANCH_STREAM...
Show more...

BRANCH_ID BRANCH_CATEGORY BRANCH_STREAM
--------- --------------- -------------
1         Engineering     CS
2         Engineering     EC
3         Medical         MBBS
4         Medical         Dental

Elapsed: 00:00:00.043
4 rows selected.
```

Step 3: Create the Fact Table

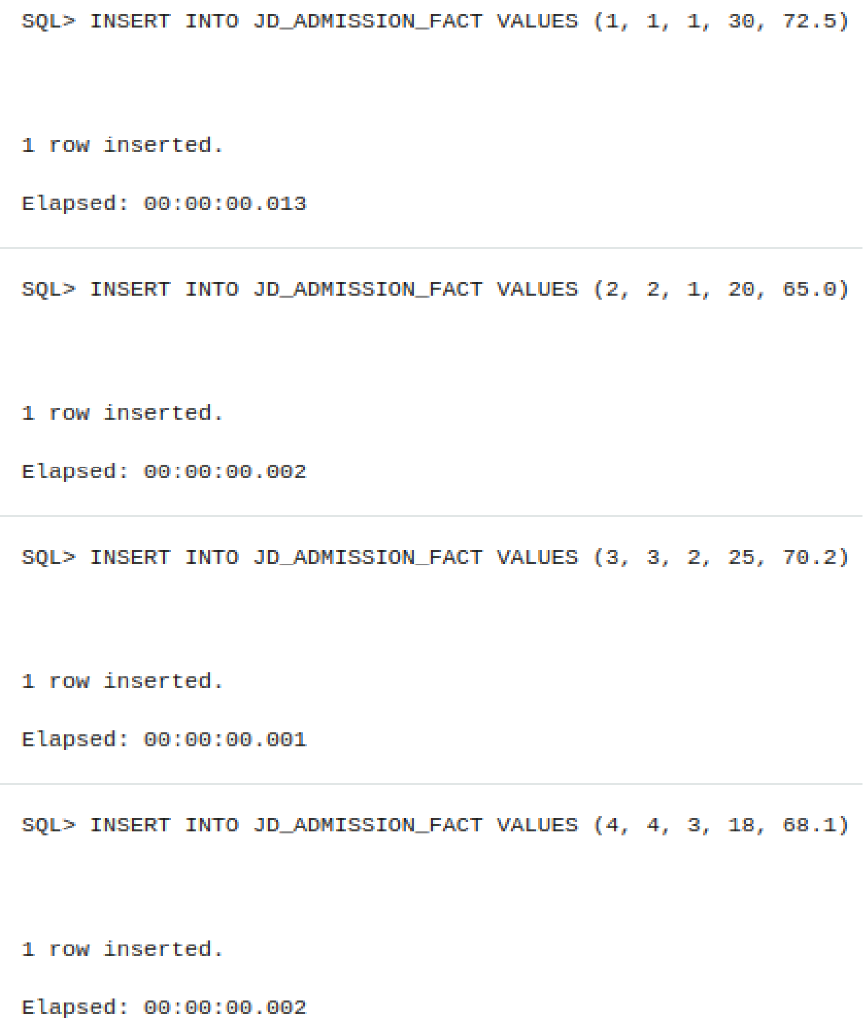


```
CREATE TABLE JD_ADMISSION_FACT (
  Branch_ID INTEGER,
  Fellow_ID INTEGER,
  Period_ID INTEGER,
  Total_Admissions INTEGER,
  Avg_Percentage NUMBER(5,2),
  FOREIGN KEY (Branch_ID) REFERENCES JD_Branch_ADM(Branch_ID),
  FOREIGN KEY (Fellow_ID) REFERENCES JD_Fellow_ADM(Fellow_ID),
  FOREIGN KEY (Period_ID) REFERENCES JD_Period_ADM(Period_ID)
);
```



```
Table JD_ADMISSION_FACT created.

Elapsed: 00:00:00.018
```

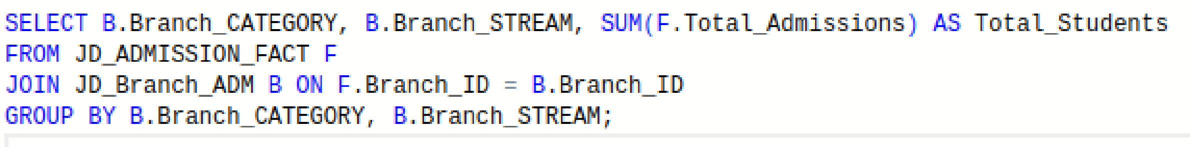
Step 4: Insert Sample Data into Fact Table



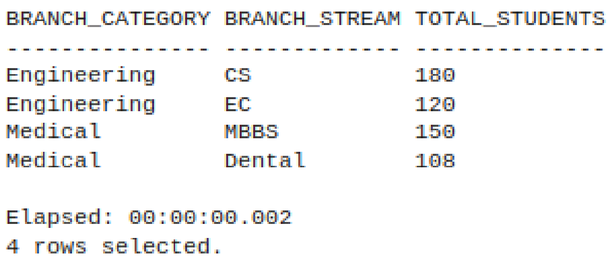
```
SQL> INSERT INTO JD_ADMISSION_FACT VALUES (1, 1, 1, 30, 72.5)


1 row inserted.

Elapsed: 00:00:00.013

SQL> INSERT INTO JD_ADMISSION_FACT VALUES (2, 2, 1, 20, 65.0)


1 row inserted.

Elapsed: 00:00:00.002

SQL> INSERT INTO JD_ADMISSION_FACT VALUES (3, 3, 2, 25, 70.2)


1 row inserted.

Elapsed: 00:00:00.001

SQL> INSERT INTO JD_ADMISSION_FACT VALUES (4, 4, 3, 18, 68.1)


1 row inserted.

Elapsed: 00:00:00.002
```

Step 5: Querying: Aggregation, Slicing, Dicing

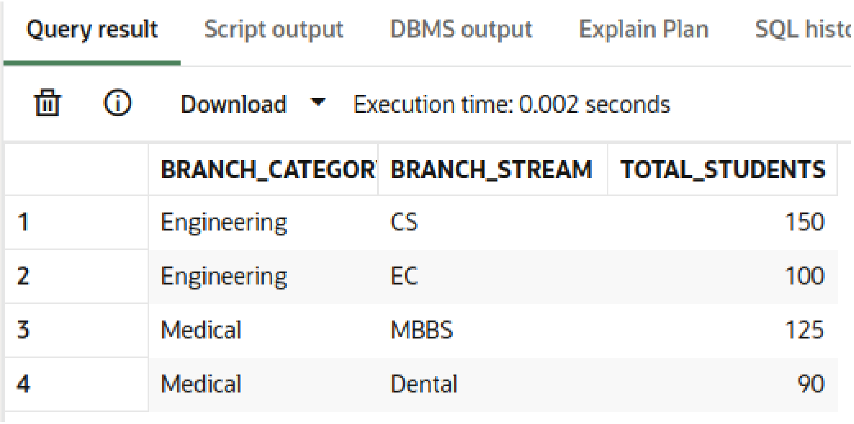
A. Get total admissions by branch:



```
SELECT B.Branch_CATEGORY, B.Branch_STREAM, SUM(F.Total_Admissions) AS Total_Students
FROM JD_ADMISSION_FACT F
JOIN JD_Branch_ADM B ON F.Branch_ID = B.Branch_ID
GROUP BY B.Branch_CATEGORY, B.Branch_STREAM;
```



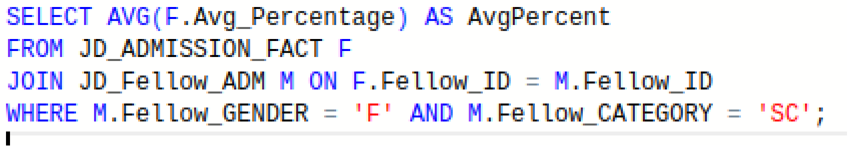
```
BRANCH_CATEGORY BRANCH_STREAM TOTAL_STUDENTS
--------------- ------------- --------------
Engineering     CS            180
Engineering     EC            120
Medical         MBBS          150
Medical         Dental        108

Elapsed: 00:00:00.002
4 rows selected.
```



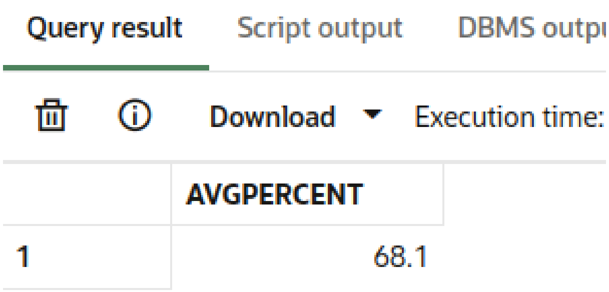
Execution time: 0.002 seconds

	BRANCH_CATEGOR	BRANCH_STREAM	TOTAL_STUDENTS
1	Engineering	CS	150
2	Engineering	EC	100
3	Medical	MBBS	125
4	Medical	Dental	90

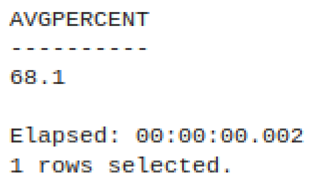
B. Average percentage for Female-SC students:



```
SELECT AVG(F.Avg_Percentage) AS AvgPercent
FROM JD_ADMISSION_FACT F
JOIN JD_Fellow_ADM M ON F.Fellow_ID = M.Fellow_ID
WHERE M.Fellow_GENDER = 'F' AND M.Fellow_CATEGORY = 'SC';
```

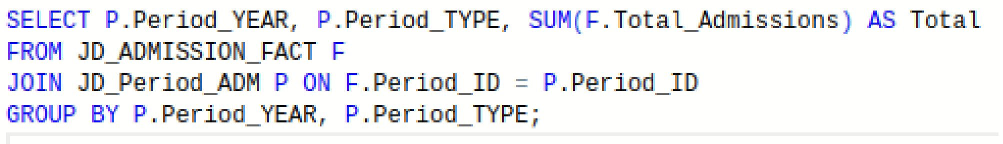


	AVGPERCENT
1	68.1

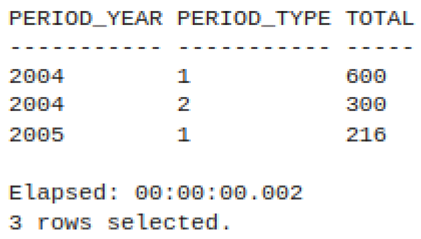


```
AVGPERCENT
----------
68.1

Elapsed: 00:00:00.002
1 rows selected.
```

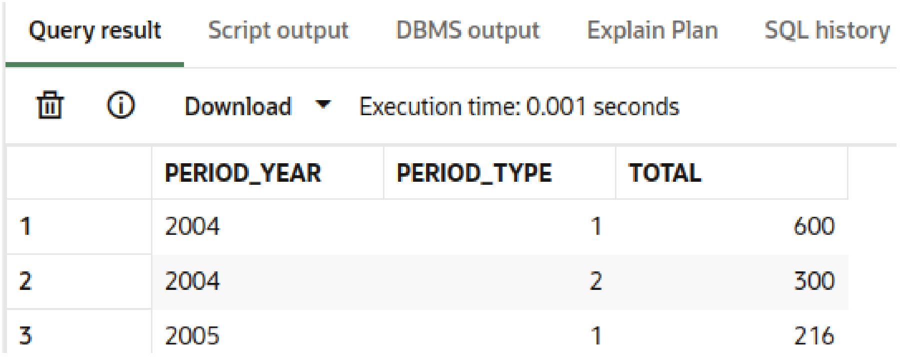
C. Admissions by year and reshuffling type:



```
SELECT P.Period_YEAR, P.Period_TYPE, SUM(F.Total_Admissions) AS Total
FROM JD_ADMISSION_FACT F
JOIN JD_Period_ADM P ON F.Period_ID = P.Period_ID
GROUP BY P.Period_YEAR, P.Period_TYPE;
```

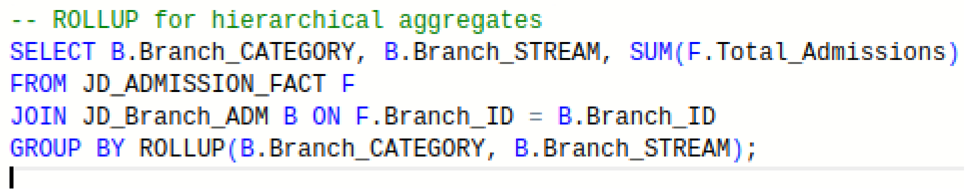


```
PERIOD_YEAR PERIOD_TYPE TOTAL
----------- ----------- -----
2004        1           600
2004        2           300
2005        1           216

Elapsed: 00:00:00.002
3 rows selected.
```

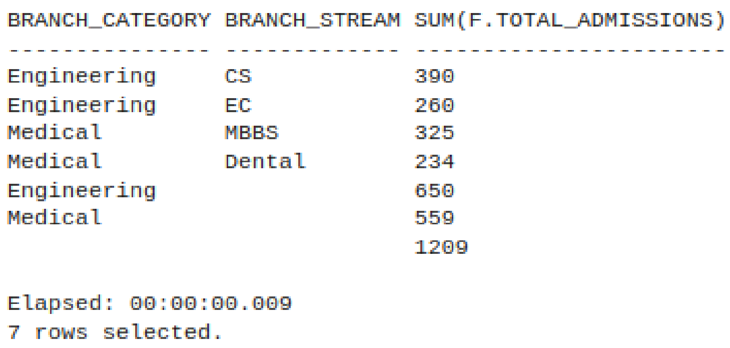


	PERIOD_YEAR	PERIOD_TYPE	TOTAL
1	2004	1	600
2	2004	2	300
3	2005	1	216

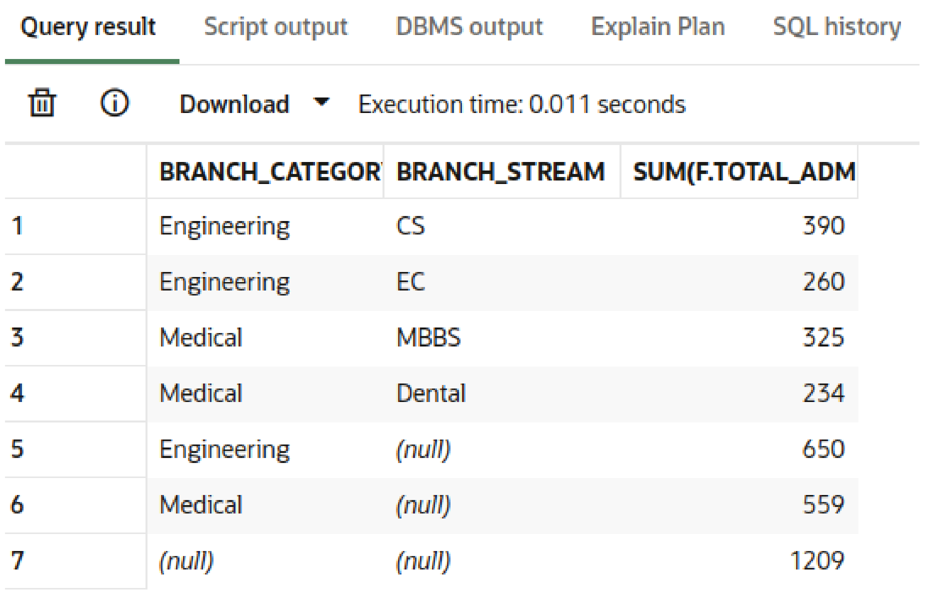
ROLLUP and CUBE for OLAP-style aggregations



```
-- ROLLUP for hierarchical aggregates
SELECT B.Branch_CATEGORY, B.Branch_STREAM, SUM(F.Total_Admissions)
FROM JD_ADMISSION_FACT F
JOIN JD_Branch_ADM B ON F.Branch_ID = B.Branch_ID
GROUP BY ROLLUP(B.Branch_CATEGORY, B.Branch_STREAM);
```

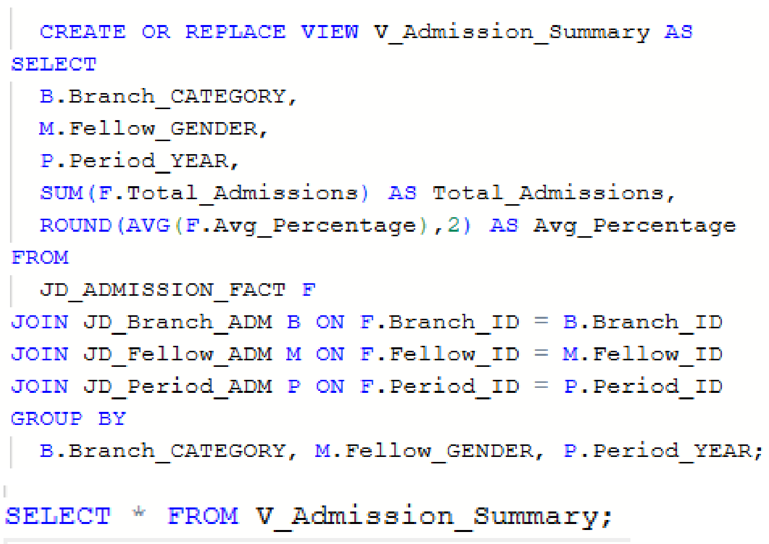


```
BRANCH_CATEGORY BRANCH_STREAM SUM(F.TOTAL_ADMISSIONS)
--------------- ------------- -----------------------
Engineering     CS            390
Engineering     EC            260
Medical         MBBS          325
Medical         Dental        234
Engineering                   650
Medical                       559
                              1209

Elapsed: 00:00:00.009
7 rows selected.
```

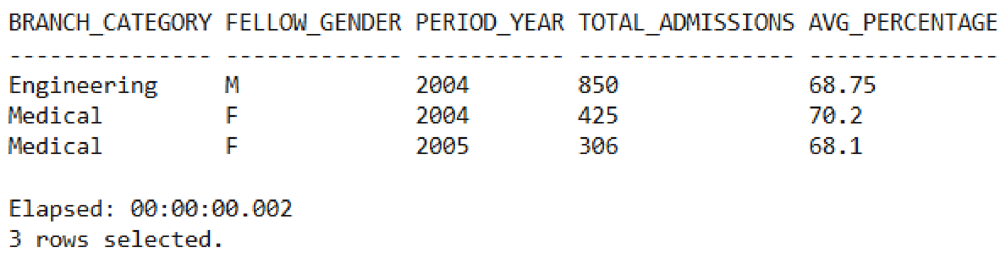


	BRANCH_CATEGOR	BRANCH_STREAM	SUM(F.TOTAL_ADM
1	Engineering	CS	390
2	Engineering	EC	260
3	Medical	MBBS	325
4	Medical	Dental	234
5	Engineering	*(null)*	650
6	Medical	*(null)*	559
7	*(null)*	*(null)*	1209

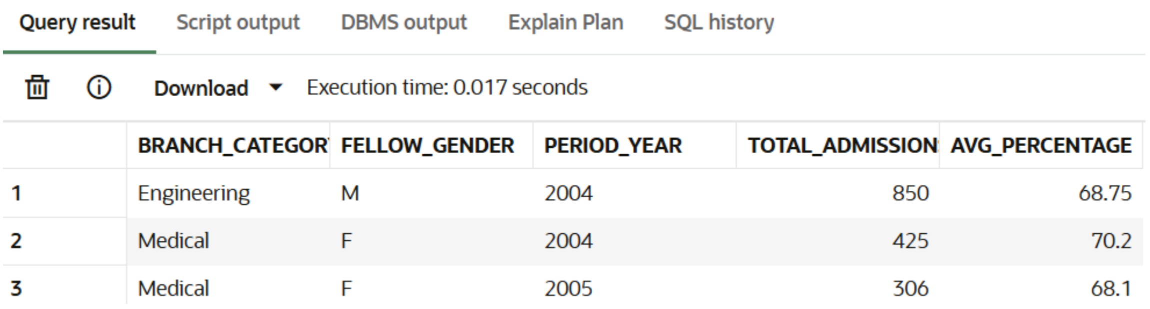
Seeing output from the View:



```
  CREATE OR REPLACE VIEW V_Admission_Summary AS
SELECT
  B.Branch_CATEGORY,
  M.Fellow_GENDER,
  P.Period_YEAR,
  SUM(F.Total_Admissions) AS Total_Admissions,
  ROUND(AVG(F.Avg_Percentage),2) AS Avg_Percentage
FROM
  JD_ADMISSION_FACT F
JOIN JD_Branch_ADM B ON F.Branch_ID = B.Branch_ID
JOIN JD_Fellow_ADM M ON F.Fellow_ID = M.Fellow_ID
JOIN JD_Period_ADM P ON F.Period_ID = P.Period_ID
GROUP BY
  B.Branch_CATEGORY, M.Fellow_GENDER, P.Period_YEAR;

SELECT * FROM V_Admission_Summary;
```



```
BRANCH_CATEGORY FELLOW_GENDER PERIOD_YEAR TOTAL_ADMISSIONS AVG_PERCENTAGE
--------------- ------------- ----------- ---------------- --------------
Engineering     M             2004        850              68.75
Medical         F             2004        425              70.2
Medical         F             2005        306              68.1

Elapsed: 00:00:00.002
3 rows selected.
```



	BRANCH_CATEGOR	FELLOW_GENDER	PERIOD_YEAR	TOTAL_ADMISSION	AVG_PERCENTAGE
1	Engineering	M	2004	850	68.75
2	Medical	F	2004	425	70.2
3	Medical	F	2005	306	68.1

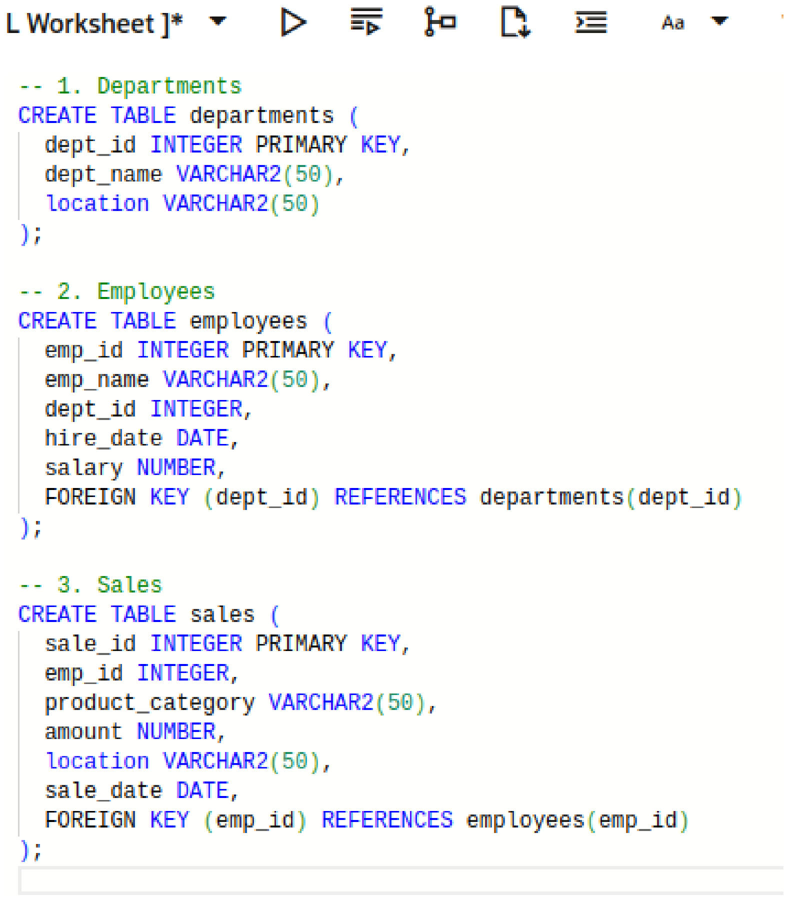
EXERCISE:

1. Differentiate OLAP vs OLTP.

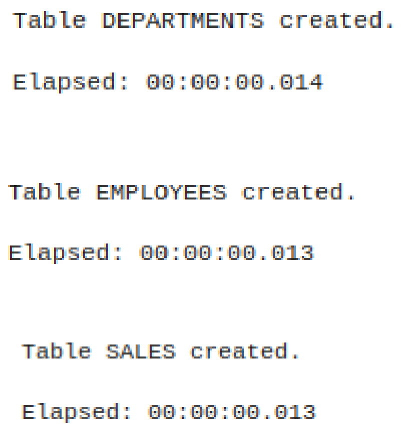
Feature	OLTP (Online Transaction Processing)	OLAP (Online Analytical Processing)
Purpose	Day-to-day operations	Analytical queries and decision making
Data	Current, detailed	Historical, summarized, multidimensional
Operations	INSERT, UPDATE, DELETE	SELECT with aggregations
Speed	Fast for read/write	Optimized for complex queries
Example	Banking transactions	Sales trend analysis
Normalization	Highly normalized	Denormalized/star schema
Queries	Simple, short	Complex with aggregations

2. Create a schema with three tables: employees, departments, and sales with following entries.

Creating tables

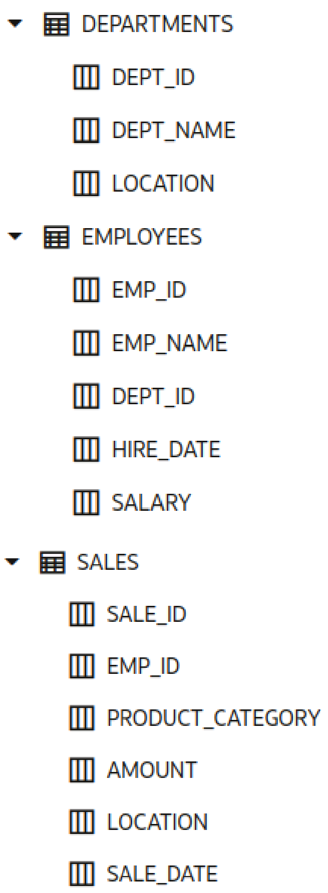


```
-- 1. Departments
CREATE TABLE departments (
  dept_id INTEGER PRIMARY KEY,
  dept_name VARCHAR2(50),
  location VARCHAR2(50)
);

-- 2. Employees
CREATE TABLE employees (
  emp_id INTEGER PRIMARY KEY,
  emp_name VARCHAR2(50),
  dept_id INTEGER,
  hire_date DATE,
  salary NUMBER,
  FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);

-- 3. Sales
CREATE TABLE sales (
  sale_id INTEGER PRIMARY KEY,
  emp_id INTEGER,
  product_category VARCHAR2(50),
  amount NUMBER,
  location VARCHAR2(50),
  sale_date DATE,
  FOREIGN KEY (emp_id) REFERENCES employees(emp_id)
);
```



```
Table DEPARTMENTS created.

Elapsed: 00:00:00.014


Table EMPLOYEES created.

Elapsed: 00:00:00.013


Table SALES created.

Elapsed: 00:00:00.013
```

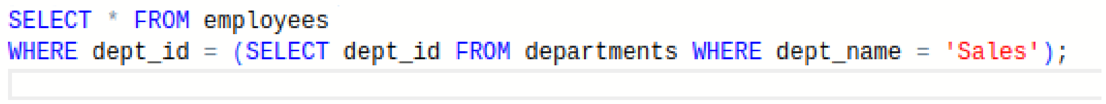
The tables look like these:



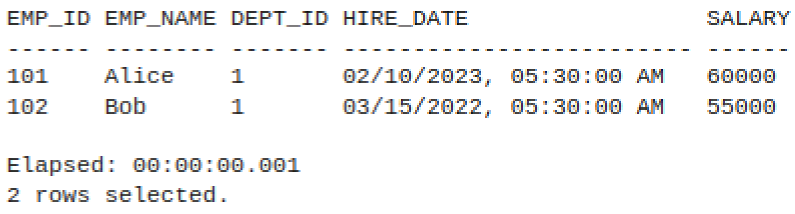
- DEPARTMENTS
  - DEPT_ID
  - DEPT_NAME
  - LOCATION
- EMPLOYEES
  - EMP_ID
  - EMP_NAME
  - DEPT_ID
  - HIRE_DATE
  - SALARY
- SALES
  - SALE_ID
  - EMP_ID
  - PRODUCT_CATEGORY
  - AMOUNT
  - LOCATION
  - SALE_DATE

Perform the following OLAP operations query on it:

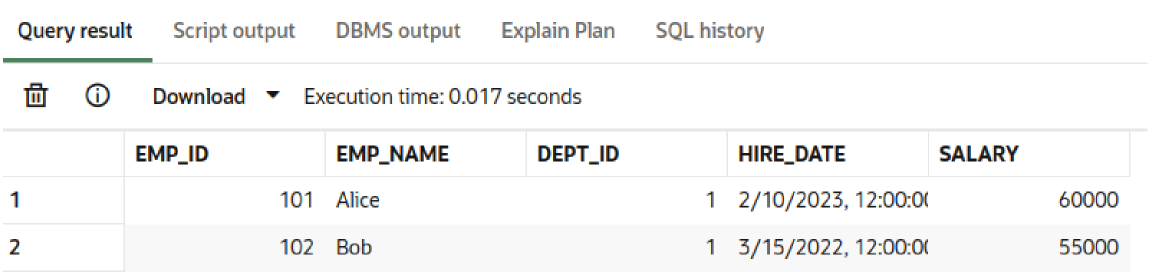
a. Slice operation for selecting a single dimension from the cube. Slice by department 'Sales'.



```
SELECT * FROM employees
WHERE dept_id = (SELECT dept_id FROM departments WHERE dept_name = 'Sales');
```

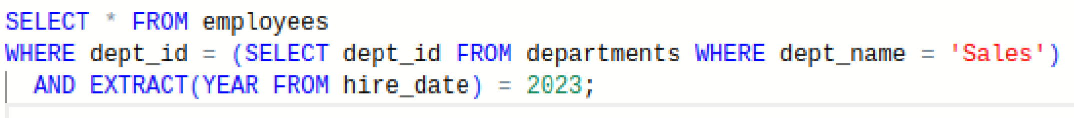


```
EMP_ID EMP_NAME DEPT_ID HIRE_DATE                 SALARY
------ -------- ------- ------------------------- ------
101    Alice    1       02/10/2023, 05:30:00 AM   60000
102    Bob      1       03/15/2022, 05:30:00 AM   55000

Elapsed: 00:00:00.001
2 rows selected.
```

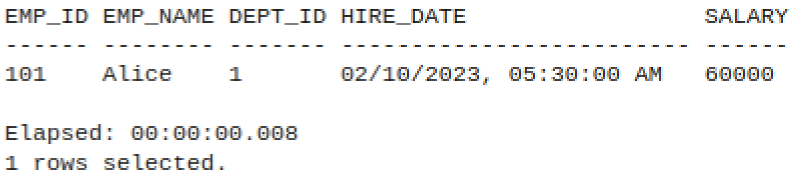


	EMP_ID	EMP_NAME	DEPT_ID	HIRE_DATE	SALARY
1	101	Alice	1	2/10/2023, 12:00:0(	60000
2	102	Bob	1	3/15/2022, 12:00:0(	55000

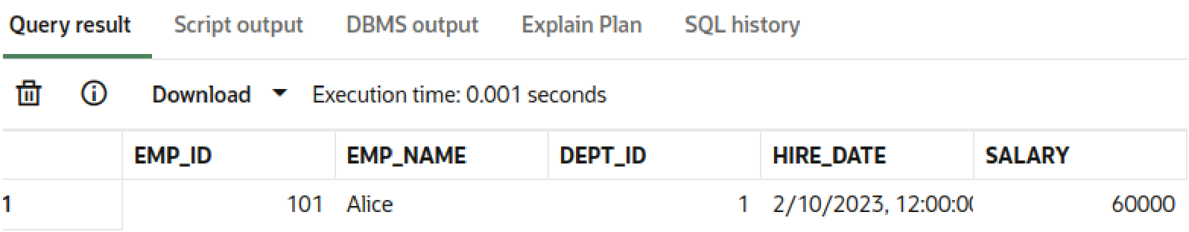
b. Dice by department 'Sales' and hire year 2023.



```
SELECT * FROM employees
WHERE dept_id = (SELECT dept_id FROM departments WHERE dept_name = 'Sales')
 AND EXTRACT(YEAR FROM hire_date) = 2023;
```

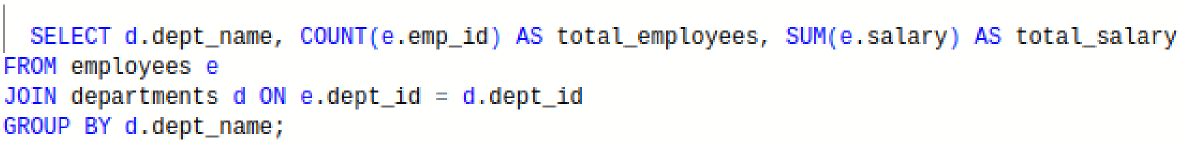


```
EMP_ID EMP_NAME DEPT_ID HIRE_DATE                SALARY
------ -------- ------- ------------------------ ------
101    Alice    1       02/10/2023, 05:30:00 AM  60000

Elapsed: 00:00:00.008
1 rows selected.
```

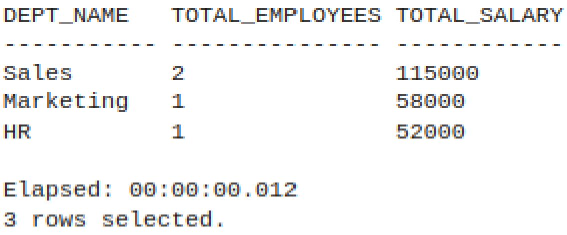


	EMP_ID	EMP_NAME	DEPT_ID	HIRE_DATE	SALARY
1	101	Alice	1	2/10/2023, 12:00:0(	60000

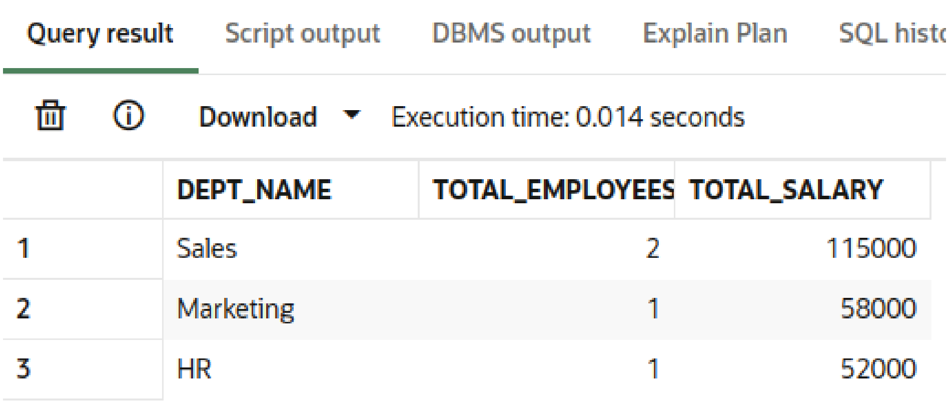
c. Roll-up by department to get total number of employees and total salary.



```
 SELECT d.dept_name, COUNT(e.emp_id) AS total_employees, SUM(e.salary) AS total_salary
FROM employees e
JOIN departments d ON e.dept_id = d.dept_id
GROUP BY d.dept_name;
```



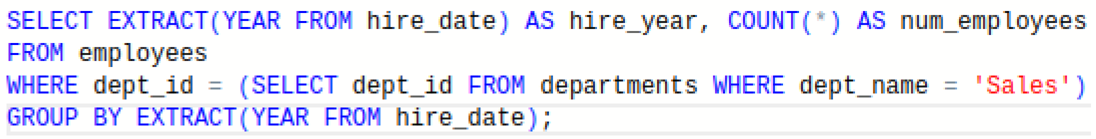
```
DEPT_NAME   TOTAL_EMPLOYEES TOTAL_SALARY
----------- --------------- ------------
Sales       2               115000
Marketing   1               58000
HR          1               52000

Elapsed: 00:00:00.012
3 rows selected.
```



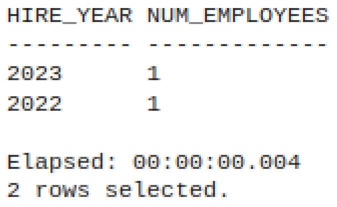
Execution time: 0.014 seconds

	DEPT_NAME	TOTAL_EMPLOYEES	TOTAL_SALARY
1	Sales	2	115000
2	Marketing	1	58000
3	HR	1	52000

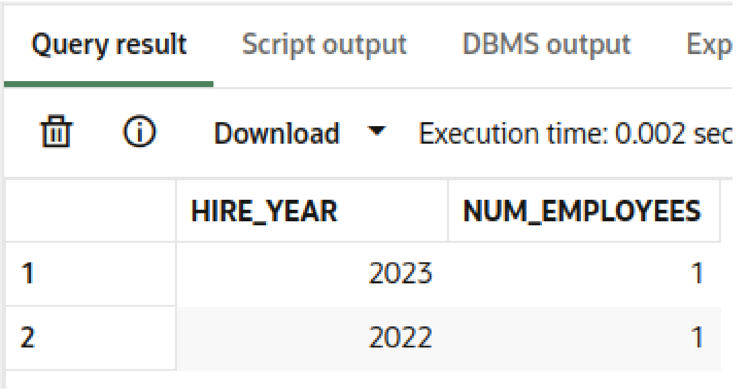
d. Drill-down by department 'Sales' to see data by hire year.



```
SELECT EXTRACT(YEAR FROM hire_date) AS hire_year, COUNT(*) AS num_employees
FROM employees
WHERE dept_id = (SELECT dept_id FROM departments WHERE dept_name = 'Sales')
GROUP BY EXTRACT(YEAR FROM hire_date);
```



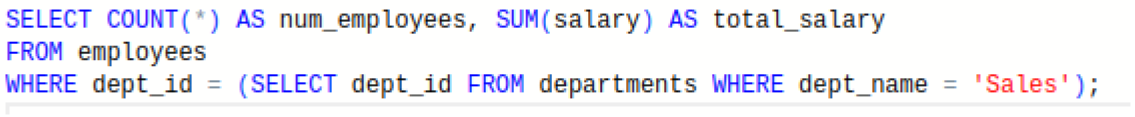
```
HIRE_YEAR NUM_EMPLOYEES
--------- -------------
2023      1
2022      1

Elapsed: 00:00:00.004
2 rows selected.
```



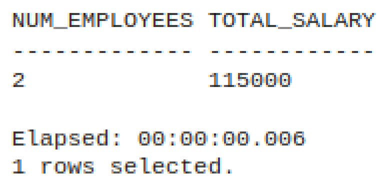
Execution time: 0.002 sec

	HIRE_YEAR	NUM_EMPLOYEES
1	2023	1
2	2022	1

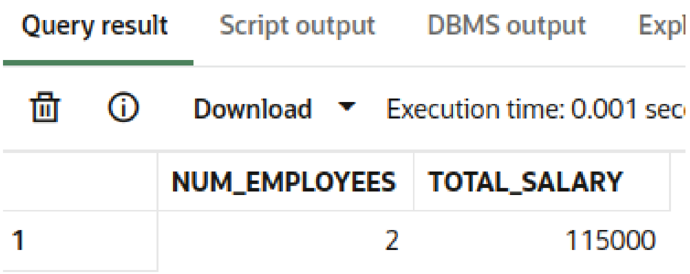
e. User Query: "Show me the total salary and number of employees in the Sales department."



```
SELECT COUNT(*) AS num_employees, SUM(salary) AS total_salary
FROM employees
WHERE dept_id = (SELECT dept_id FROM departments WHERE dept_name = 'Sales');
```

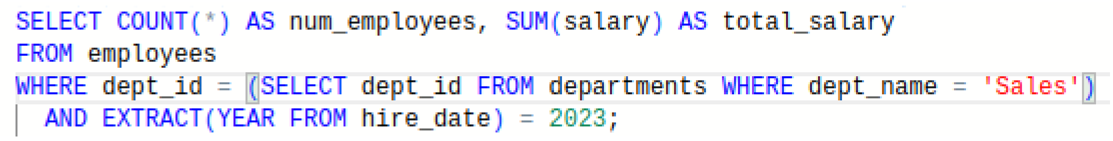


```
NUM_EMPLOYEES TOTAL_SALARY
------------- ------------
2             115000

Elapsed: 00:00:00.006
1 rows selected.
```

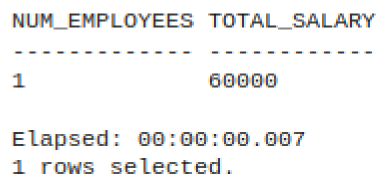


	NUM_EMPLOYEES	TOTAL_SALARY
1	2	115000

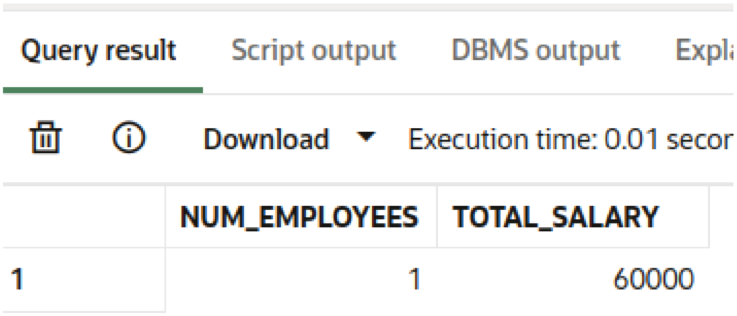
f. User Query: "Show me the total salary and number of employees hired in 2023 in the Sales department."



```
SELECT COUNT(*) AS num_employees, SUM(salary) AS total_salary
FROM employees
WHERE dept_id = (SELECT dept_id FROM departments WHERE dept_name = 'Sales')
  AND EXTRACT(YEAR FROM hire_date) = 2023;
```

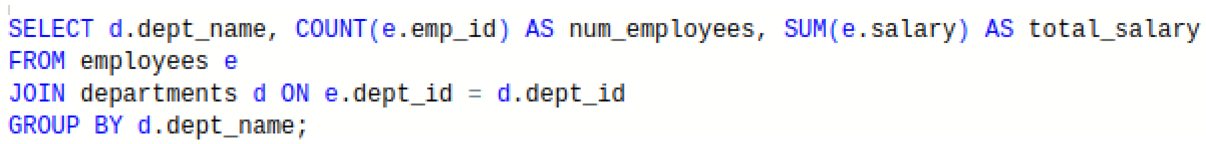


```
NUM_EMPLOYEES TOTAL_SALARY
------------- ------------
1             60000

Elapsed: 00:00:00.007
1 rows selected.
```

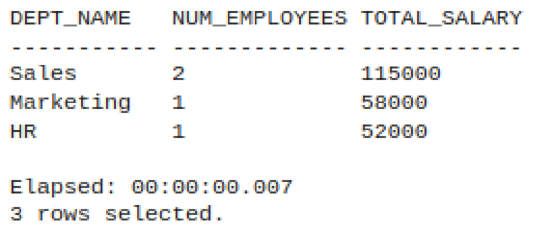


	NUM_EMPLOYEES	TOTAL_SALARY
1	1	60000

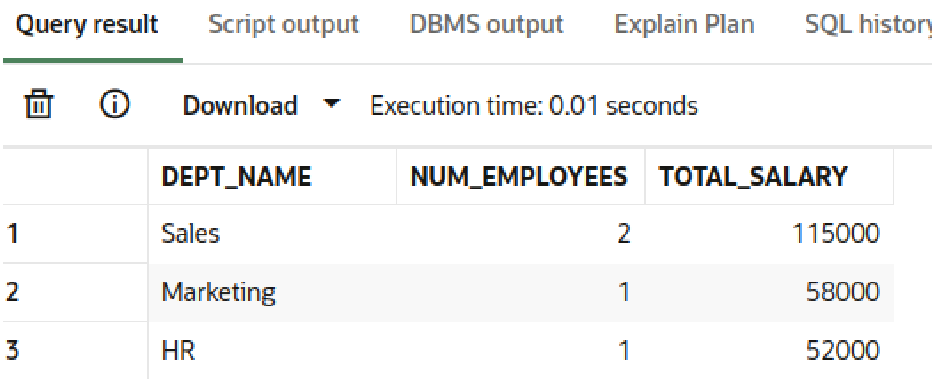
g. User Query: "Show me the total salary and number of employees aggregated by department."



```
SELECT d.dept_name, COUNT(e.emp_id) AS num_employees, SUM(e.salary) AS total_salary
FROM employees e
JOIN departments d ON e.dept_id = d.dept_id
GROUP BY d.dept_name;
```



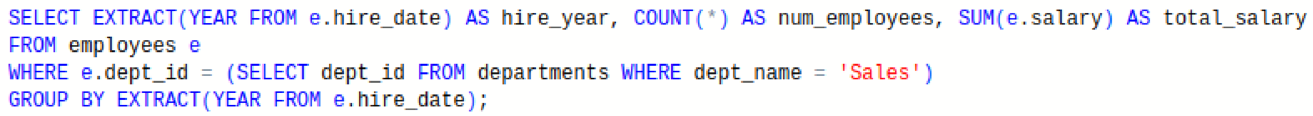
```
DEPT_NAME   NUM_EMPLOYEES TOTAL_SALARY
----------- ------------- ------------
Sales       2             115000
Marketing   1             58000
HR          1             52000

Elapsed: 00:00:00.007
3 rows selected.
```



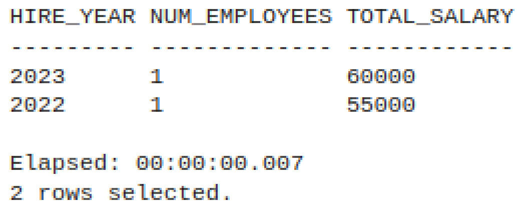
Execution time: 0.01 seconds

	DEPT_NAME	NUM_EMPLOYEES	TOTAL_SALARY
1	Sales	2	115000
2	Marketing	1	58000
3	HR	1	52000

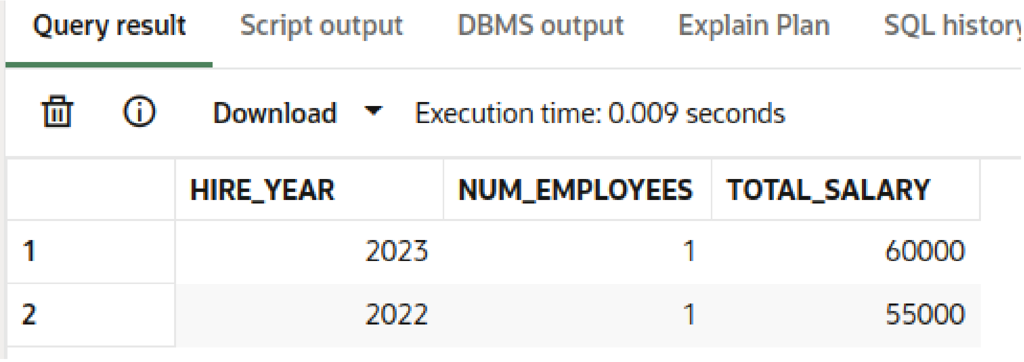
h. User Query: "Show me the total salary and number of employees in the Sales department, broken down by hire year."



```
SELECT EXTRACT(YEAR FROM e.hire_date) AS hire_year, COUNT(*) AS num_employees, SUM(e.salary) AS total_salary
FROM employees e
WHERE e.dept_id = (SELECT dept_id FROM departments WHERE dept_name = 'Sales')
GROUP BY EXTRACT(YEAR FROM e.hire_date);
```



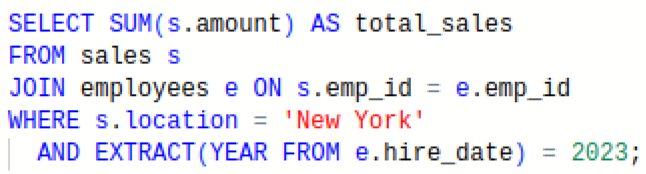
```
HIRE_YEAR NUM_EMPLOYEES TOTAL_SALARY
--------- ------------- ------------
2023      1             60000
2022      1             55000

Elapsed: 00:00:00.007
2 rows selected.
```



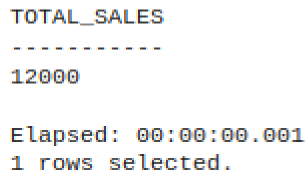
Execution time: 0.009 seconds

	HIRE_YEAR	NUM_EMPLOYEES	TOTAL_SALARY
1	2023	1	60000
2	2022	1	55000

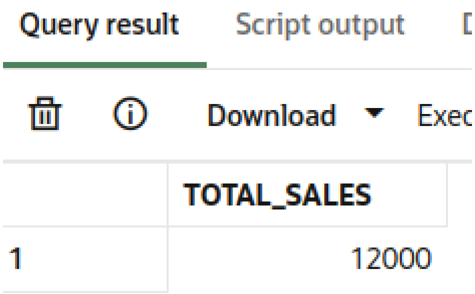
i. User Query: "Show me the total sales amount for products sold in New York by employees hired in 2023."



```
SELECT SUM(s.amount) AS total_sales
FROM sales s
JOIN employees e ON s.emp_id = e.emp_id
WHERE s.location = 'New York'
  AND EXTRACT(YEAR FROM e.hire_date) = 2023;
```

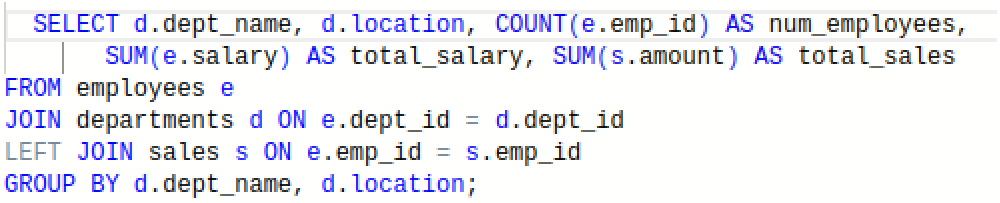


```
TOTAL_SALES
-----------
12000

Elapsed: 00:00:00.001
1 rows selected.
```

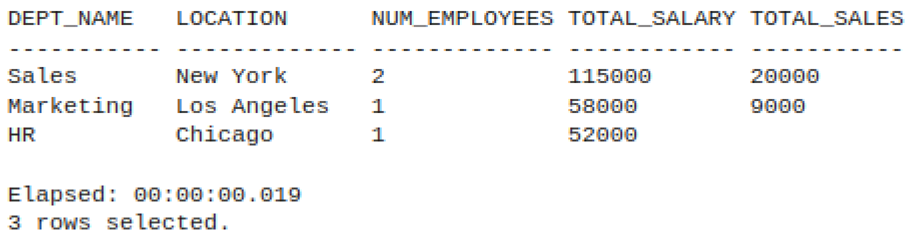


	TOTAL_SALES
1	12000

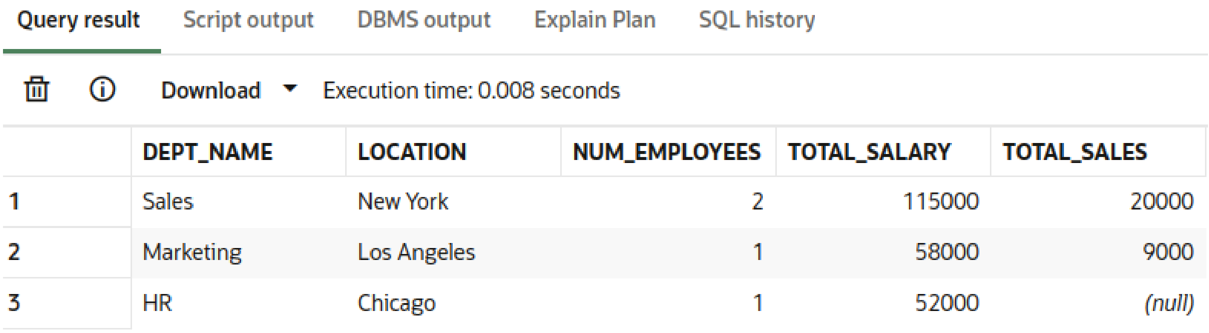
j. User Query: "Show me the total salary, number of employees, and total sales amount aggregated by department and location."



```
SELECT d.dept_name, d.location, COUNT(e.emp_id) AS num_employees,
     SUM(e.salary) AS total_salary, SUM(s.amount) AS total_sales
FROM employees e
JOIN departments d ON e.dept_id = d.dept_id
LEFT JOIN sales s ON e.emp_id = s.emp_id
GROUP BY d.dept_name, d.location;
```



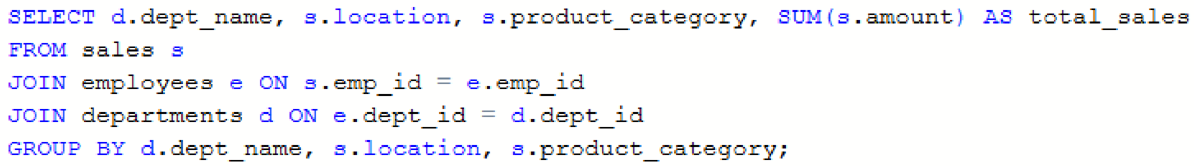
```
DEPT_NAME   LOCATION     NUM_EMPLOYEES TOTAL_SALARY TOTAL_SALES
----------- ------------ ------------- ------------ -----------
Sales       New York     2             115000       20000
Marketing   Los Angeles  1             58000        9000
HR          Chicago      1             52000

Elapsed: 00:00:00.019
3 rows selected.
```



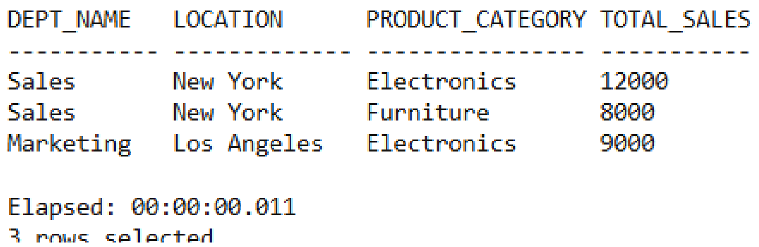
Execution time: 0.008 seconds

	DEPT_NAME	LOCATION	NUM_EMPLOYEES	TOTAL_SALARY	TOTAL_SALES
1	Sales	New York	2	115000	20000
2	Marketing	Los Angeles	1	58000	9000
3	HR	Chicago	1	52000	*(null)*

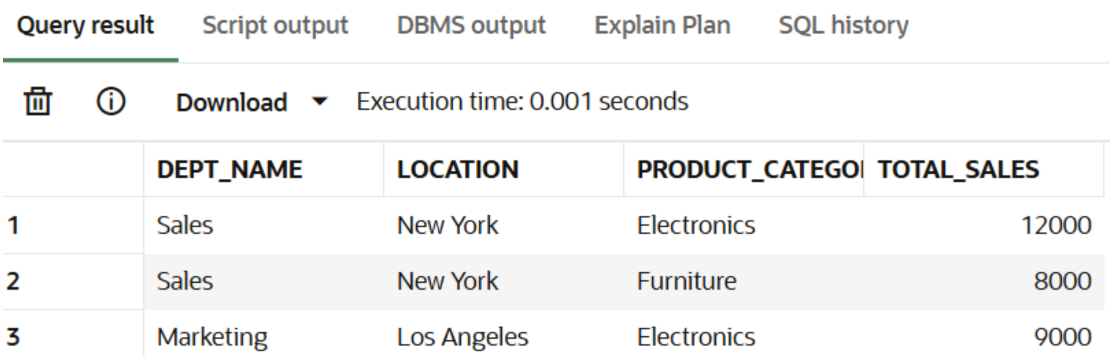
k. User Query: "Show me the total sales amount by department, location, and product category."



```
SELECT d.dept_name, s.location, s.product_category, SUM(s.amount) AS total_sales
FROM sales s
JOIN employees e ON s.emp_id = e.emp_id
JOIN departments d ON e.dept_id = d.dept_id
GROUP BY d.dept_name, s.location, s.product_category;
```



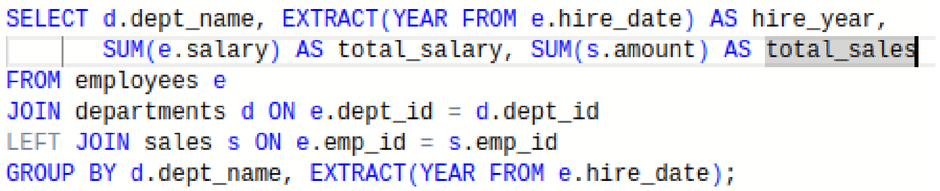
```
DEPT_NAME   LOCATION      PRODUCT_CATEGORY TOTAL_SALES
----------- ------------- ---------------- -----------
Sales       New York      Electronics      12000
Sales       New York      Furniture        8000
Marketing   Los Angeles   Electronics      9000

Elapsed: 00:00:00.011
3 rows selected
```



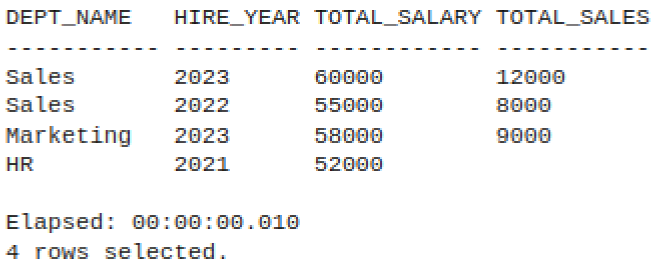
Execution time: 0.001 seconds

	DEPT_NAME	LOCATION	PRODUCT_CATEGOI	TOTAL_SALES
1	Sales	New York	Electronics	12000
2	Sales	New York	Furniture	8000
3	Marketing	Los Angeles	Electronics	9000

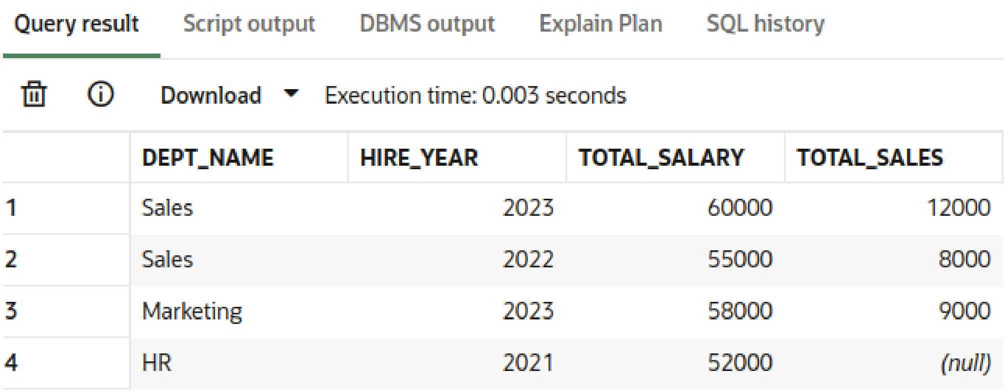
l. User Query: "Show me the total salary and total sales amount for all combinations of departments and hire years."



```
SELECT d.dept_name, EXTRACT(YEAR FROM e.hire_date) AS hire_year,
       SUM(e.salary) AS total_salary, SUM(s.amount) AS total_sales
FROM employees e
JOIN departments d ON e.dept_id = d.dept_id
LEFT JOIN sales s ON e.emp_id = s.emp_id
GROUP BY d.dept_name, EXTRACT(YEAR FROM e.hire_date);
```



```
DEPT_NAME   HIRE_YEAR TOTAL_SALARY TOTAL_SALES
----------- --------- ------------ -----------
Sales       2023      60000        12000
Sales       2022      55000        8000
Marketing   2023      58000        9000
HR          2021      52000

Elapsed: 00:00:00.010
4 rows selected.
```

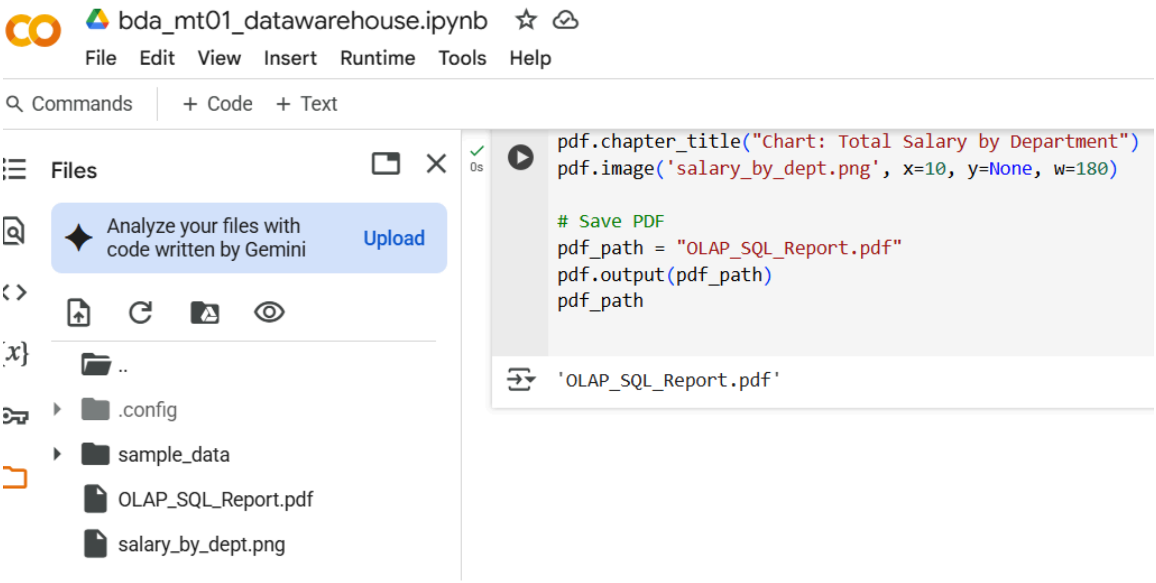


	DEPT_NAME	HIRE_YEAR	TOTAL_SALARY	TOTAL_SALES
1	Sales	2023	60000	12000
2	Sales	2022	55000	8000
3	Marketing	2023	58000	9000
4	HR	2021	52000	(null)

3. Learn and explore tools/api to be able to generate pdf-like reports containing tabular/visualizations dashboards, etc.

I’ve exported the data from Oracle live sql to python colab.
The code can be seen here: MT01_Bda_Datawarehouse.ipynb

Thus the pdf gets generated and I’ve downloaded it from the files section.



```
pdf.chapter_title("Chart: Total Salary by Department")
pdf.image('salary_by_dept.png', x=10, y=None, w=180)

# Save PDF
pdf_path = "OLAP_SQL_Report.pdf"
pdf.output(pdf_path)
pdf_path
```

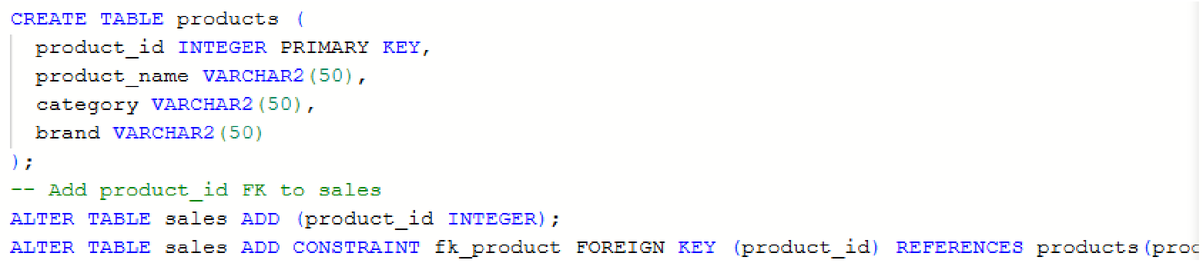
```
'OLAP_SQL_Report.pdf'
```

The copy of pdf-report is attached along with this lab work.

Extra exercise:

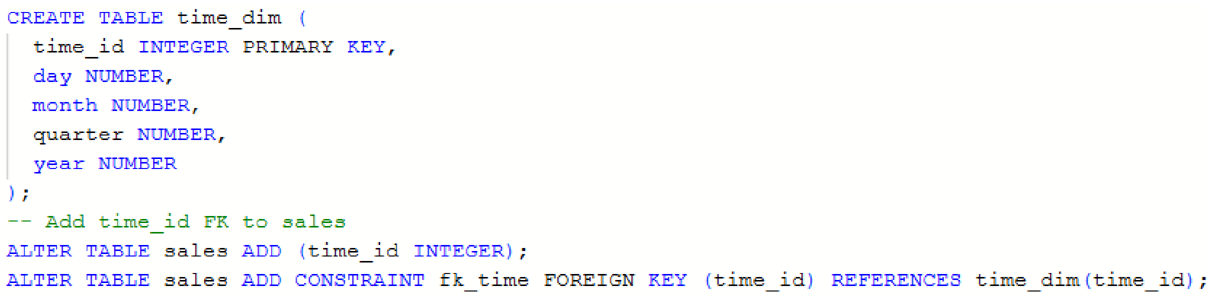
Extend the business process to admit 4th and 5th dimensional data and idealize the changes/additions required to achieve more dimensionality in Data warehouse.

Add a Product dimension (4th)



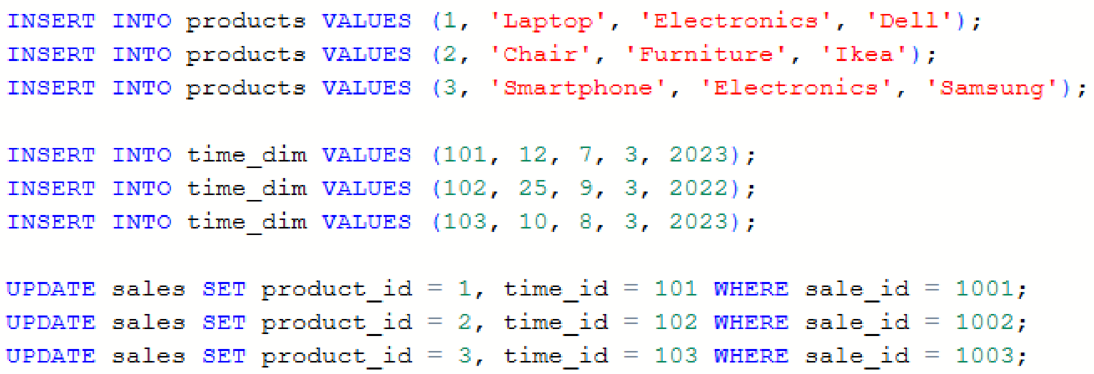
```
CREATE TABLE products (
  product_id INTEGER PRIMARY KEY,
  product_name VARCHAR2(50),
  category VARCHAR2(50),
  brand VARCHAR2(50)
);
-- Add product_id FK to sales
ALTER TABLE sales ADD (product_id INTEGER);
ALTER TABLE sales ADD CONSTRAINT fk_product FOREIGN KEY (product_id) REFERENCES products(proc
```

Add a Customer or Time Dimension Table (5th)



```
CREATE TABLE time_dim (
  time_id INTEGER PRIMARY KEY,
  day NUMBER,
  month NUMBER,
  quarter NUMBER,
  year NUMBER
);
-- Add time_id FK to sales
ALTER TABLE sales ADD (time_id INTEGER);
ALTER TABLE sales ADD CONSTRAINT fk_time FOREIGN KEY (time_id) REFERENCES time_dim(time_id);
```

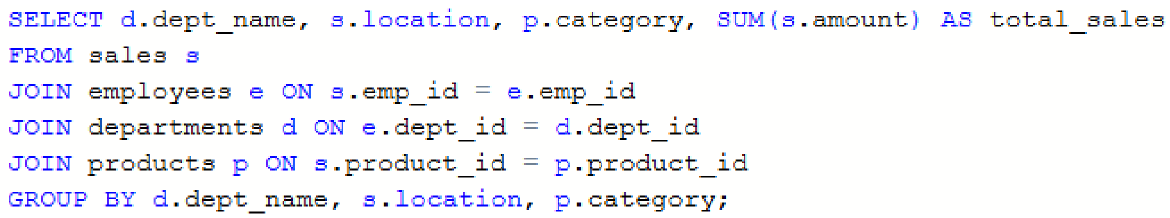
Sample Data to Insert in products, time_dim & update sales table:



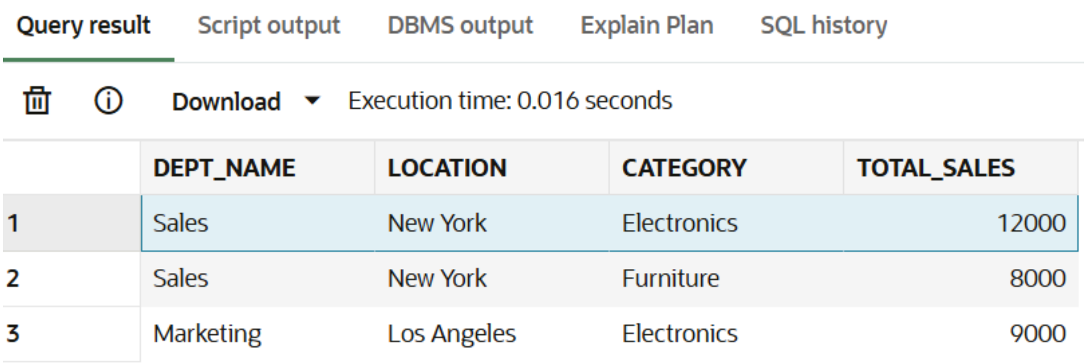
```
INSERT INTO products VALUES (1, 'Laptop', 'Electronics', 'Dell');
INSERT INTO products VALUES (2, 'Chair', 'Furniture', 'Ikea');
INSERT INTO products VALUES (3, 'Smartphone', 'Electronics', 'Samsung');

INSERT INTO time_dim VALUES (101, 12, 7, 3, 2023);
INSERT INTO time_dim VALUES (102, 25, 9, 3, 2022);
INSERT INTO time_dim VALUES (103, 10, 8, 3, 2023);

UPDATE sales SET product_id = 1, time_id = 101 WHERE sale_id = 1001;
UPDATE sales SET product_id = 2, time_id = 102 WHERE sale_id = 1002;
UPDATE sales SET product_id = 3, time_id = 103 WHERE sale_id = 1003;
```

Now we can see query across 5 dimensions!

Query 1: Total sales by department, location, and product category



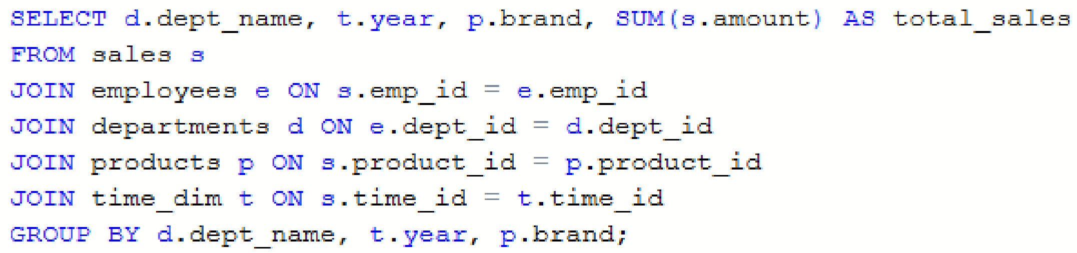
```sql
SELECT d.dept_name, s.location, p.category, SUM(s.amount) AS total_sales
FROM sales s
JOIN employees e ON s.emp_id = e.emp_id
JOIN departments d ON e.dept_id = d.dept_id
JOIN products p ON s.product_id = p.product_id
GROUP BY d.dept_name, s.location, p.category;
```



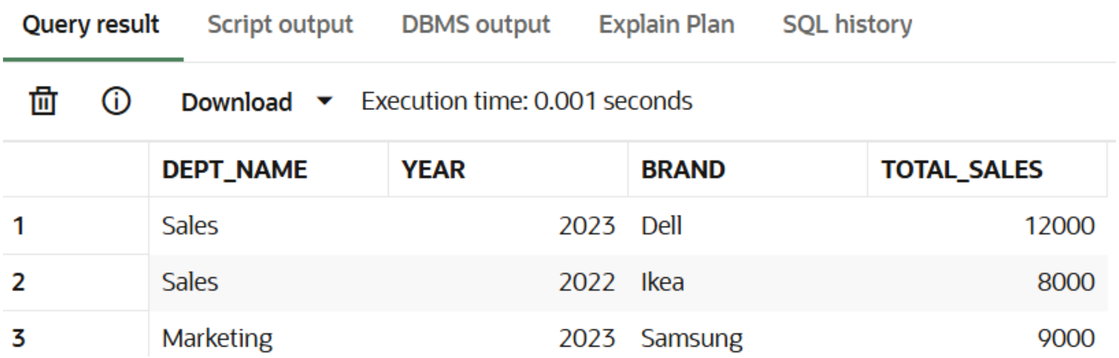
Execution time: 0.016 seconds

	DEPT_NAME	LOCATION	CATEGORY	TOTAL_SALES
1	Sales	New York	Electronics	12000
2	Sales	New York	Furniture	8000
3	Marketing	Los Angeles	Electronics	9000

Query 2: Total sales by department, year, and product brand



```sql
SELECT d.dept_name, t.year, p.brand, SUM(s.amount) AS total_sales
FROM sales s
JOIN employees e ON s.emp_id = e.emp_id
JOIN departments d ON e.dept_id = d.dept_id
JOIN products p ON s.product_id = p.product_id
JOIN time_dim t ON s.time_id = t.time_id
GROUP BY d.dept_name, t.year, p.brand;
```



Execution time: 0.001 seconds

	DEPT_NAME	YEAR	BRAND	TOTAL_SALES
1	Sales	2023	Dell	12000
2	Sales	2022	Ikea	8000
3	Marketing	2023	Samsung	9000

Summarised learning:

This lab explored key OLAP operations like slice, dice, roll-up, and drill-down using SQL over a dimensional data warehouse. I generated a structured PDF report of results and visualizations. The warehouse was further extended to support 4th and 5th dimensions, enabling richer multi-dimensional analytics for business insights.