Experiment 2

Student Name: Nikhil Kumar
UID: 25MCI10036
Branch: MCA (AI/ML)
Section/Group: 25MAM-1(A)
Semester: 2-SEM
Date of Performance: 13/01/2026
Subject Name: Technical Training - 1
Subject Code: 25CAP-652

1. Aim:

To implement and analyze SQL SELECT queries using filtering, sorting, grouping, and aggregation concepts in PostgreSQL for efficient data retrieval and analytical reporting.

2. Tools Used:

- PostgreSQL

3. Objectives:

- To retrieve specific data using filtering conditions
- To sort query results using single and multiple attributes
- To perform aggregation using grouping techniques
- To apply conditions on aggregated data
- To understand real-world analytical queries commonly asked in placement interviews

4. Practical / Experiment Steps

Step 1: Database and Table Preparation

- Start the PostgreSQL server.
- Open the PostgreSQL client tool.
- Create a database for the experiment.

- Prepare a sample table representing customer orders containing details such as customer name, product, quantity, price, and order date.
- Insert sufficient sample records to allow meaningful analysis.

Purpose: To create a realistic dataset for performing analytical queries.

Step 2: Filtering Data Using Conditions

- Execute data retrieval operations to display only those records that satisfy specific conditions, such as higher-priced orders.
- Observe how filtering limits the number of rows returned.

Observation: Filtering reduces unnecessary data processing and improves query efficiency.

Step 3: Sorting Query Results

- Retrieve selected columns from the table and arrange the output based on numerical values such as price.
- Perform sorting using both ascending and descending order.
- Apply sorting on more than one attribute to understand priority-based ordering.

Observation: Sorting is essential for reports, rankings, and ordered displays.

Step 4: Grouping Data for Aggregation

- Group records based on a common attribute such as product.
- Calculate aggregate values like total sales for each group.
- Analyze how multiple rows are combined into summarized results.

Observation: Grouping transforms transactional data into analytical insights.

Step 5: Applying Conditions on Aggregated Data

- Apply conditions on grouped results to retrieve only those groups that satisfy specific aggregate criteria.

- Compare the difference between row-level filtering and group-level filtering.

Observation: Conditions applied after grouping allow refined analytical reporting.

Step 6: Conceptual Understanding of Filtering vs Aggregation Conditions

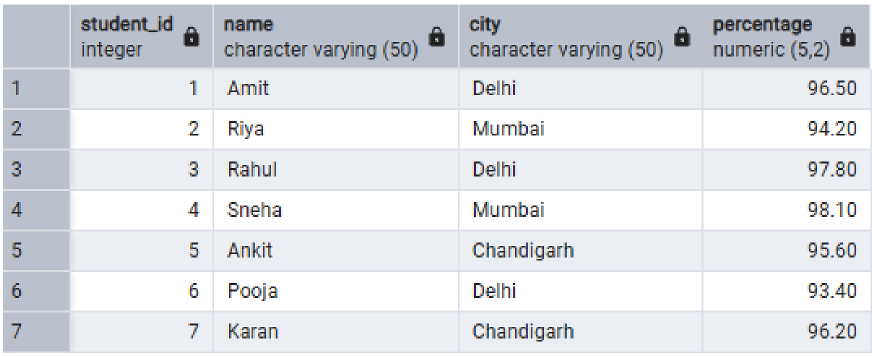
- Analyze scenarios where conditions are incorrectly applied before grouping.
- Correctly apply conditions after grouping to avoid logical errors.

Observation: Understanding execution order prevents common SQL mistakes frequently tested in interviews.

5. Coding / Implementation:

```
CREATE TABLE Students (
student_id INT,
name VARCHAR(50),
city VARCHAR(50),
percentage DECIMAL(5,2)
);


INSERT INTO Students VALUES
(1, 'Amit', 'Delhi', 96.5),
(2, 'Riya', 'Mumbai', 94.2),
(3, 'Rahul', 'Delhi', 97.8),
(4, 'Sneha', 'Mumbai', 98.1),
(5, 'Ankit', 'Chandigarh', 95.6),
(6, 'Pooja', 'Delhi', 93.4),
(7, 'Karan', 'Chandigarh', 96.2);


----Sum of student percentage >95 group by city

---Without Case Statement

SELECT CITY , COUNT(*) AS STUDET_COUNT FROM Students

WHERE percentage> 95

GROUP BY city;


-- WITH CASE STATEMENT

SELECT CITY, SUM(CASE WHEN percentage> 95 THEN 1

ELSE 0 END) AS STUDENT_COUNTS FROM Students

GROUP BY city


-- (II) Average percentage of Student whoes percentage >95 Group by City
With case statement

SELECT CITY, AVG(CASE WHEN PERCENTAGE>95 THEN
PERCENTAGE ELSE

NULL END) AS STUDENT_AVG FROM Students

GROUP BY city

ORDER BY STUDENT_AVG DESC;
```

6. Input Data

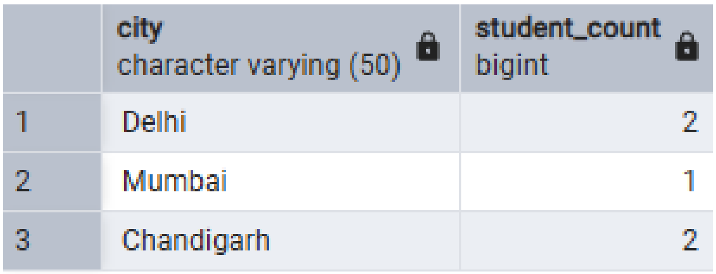


	student_id integer	name character varying (50)	city character varying (50)	percentage numeric (5,2)
1	1	Amit	Delhi	96.50
2	2	Riya	Mumbai	94.20
3	3	Rahul	Delhi	97.80
4	4	Sneha	Mumbai	98.10
5	5	Ankit	Chandigarh	95.60
6	6	Pooja	Delhi	93.40
7	7	Karan	Chandigarh	96.20

7. Output

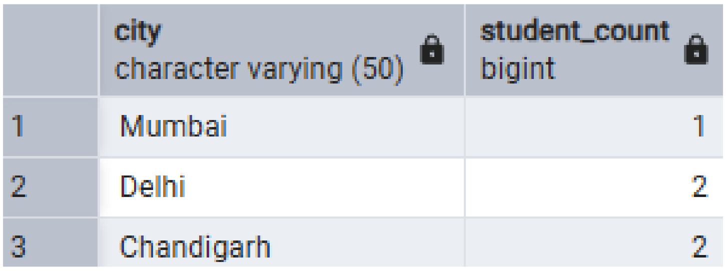
I. Sum of student percentage >95 group by city

i. Without case statement



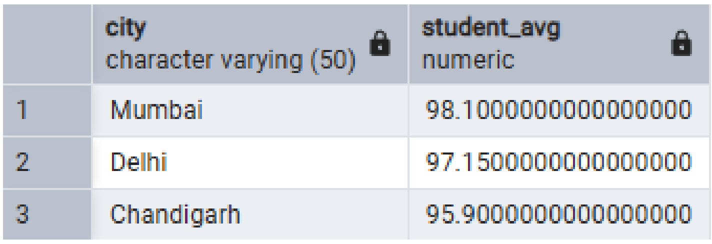
	city character varying (50)	student_count bigint
1	Delhi	2
2	Mumbai	1
3	Chandigarh	2

ii. With case statement



	city character varying (50)	student_count bigint
1	Mumbai	1
2	Delhi	2
3	Chandigarh	2

II . Average percentage of Student whoes percentage >95 Group by City With case statement



	city character varying (50)	student_avg numeric
1	Mumbai	98.1000000000000000
2	Delhi	97.1500000000000000
3	Chandigarh	95.9000000000000000

8. Learning Outcomes (What I have learned):

- Students understand how data can be filtered to retrieve only relevant records from a database.
- Students learn how sorting improves readability and usefulness of query results in reports.
- Students gain the ability to group data for analytical purposes.
- Students clearly differentiate between row-level conditions and group-level conditions.
- Students develop confidence in writing analytical SQL queries used in real-world scenarios.
- Students are better prepared to answer SQL-based placement and interview questions related to filtering, grouping, and aggregation.