Experiment 2

Student Name: Shubham Agarwal **UID: 25MCI10091**
Branch: MCA (AI & ML) **Section/Group: 25MAM_KAR-1_A**
Semester: 2 **Date of Performance: 20/01/26**
Subject Name: Technical Training **Subject Code: 25CAP-652**

Title

Implementation of SELECT Queries with Filtering, Grouping and Sorting in PostgreSQL

Aim

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

Software Requirements:

• PostgreSQL
• pgAdmin
• Windows Operating System

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

Procedure of the practical

- Create a sample table representing Employee details
- Insert realistic records into the table
- Retrieve filtered data using WHERE clause
- Sort query results using ORDER BY clause
- Group records using GROUP BY clause
- Apply conditions on grouped data using HAVING clause
- Analyze execution order of WHERE and HAVING clauses

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.

Practical:

Step 1: Database and Table Preparation

```
CREATE TABLE orders (
   order_id SERIAL PRIMARY KEY,
   customer_name VARCHAR(50),
   product VARCHAR(50),
   quantity INT,
   price NUMERIC(10,2),
   order_date DATE
);

INSERT INTO orders (customer_name, product, quantity, price, order_date) VALUES
('Amit', 'Laptop', 1, 65000, '2024-01-10'),
('Neha', 'Mobile', 2, 40000, '2024-01-12'),
('Rohan', 'Tablet', 1, 25000, '2024-01-15'),
('Simran', 'Laptop', 1, 70000, '2024-01-18'),
('Ankit', 'Mobile', 3, 60000, '2024-01-20'),
('Pooja', 'Headphones', 2, 5000, '2024-01-22'),
('Rahul', 'Laptop', 1, 68000, '2024-01-25');
```

	order_id [PK] integer	customer_name character varying (50)	product character varying (50)	quantity integer	price numeric (10,2)	order_date date
1	1	Amit	Laptop	1	65000.00	2024-01-10
2	2	Neha	Mobile	2	40000.00	2024-01-12
3	3	Rohan	Tablet	1	25000.00	2024-01-15
4	4	Simran	Laptop	1	70000.00	2024-01-18
5	5	Ankit	Mobile	3	60000.00	2024-01-20
6	6	Pooja	Headphones	2	5000.00	2024-01-22
7	7	Rahul	Laptop	1	68000.00	2024-01-25

Step 2: Filtering Data Using Conditions

SELECT * FROM orders WHERE price > 50000;

	order_id [PK] integer	customer_name character varying (50)	product character varying (50)	quantity integer	price numeric (10,2)	order_date date
1	1	Amit	Laptop	1	65000.00	2024-01-10
2	4	Simran	Laptop	1	70000.00	2024-01-18
3	5	Ankit	Mobile	3	60000.00	2024-01-20
4	7	Rahul	Laptop	1	68000.00	2024-01-25

Step 3: Sorting Query Results

SELECT order_id, customer_name, product, price FROM orders
ORDER BY price ASC;

	customer_name character varying (50)	product character varying (50)	price numeric (10,2)
1	Pooja	Headphones	5000.00
2	Rohan	Tablet	25000.00
3	Neha	Mobile	40000.00
4	Ankit	Mobile	60000.00
5	Amit	Laptop	65000.00
6	Rahul	Laptop	68000.00
7	Simran	Laptop	70000.00

SELECT customer_name, product, price, quantity FROM orders
ORDER BY product ASC, price DESC;

Step 4: Grouping Data for Aggregation

SELECT product,
    SUM(price * quantity) AS total_sales
FROM orders
GROUP BY product;

	product character varying (50)	total_sales numeric
1	Mobile	260000.00
2	Tablet	25000.00
3	Laptop	203000.00
4	Headphones	10000.00

Step 5: Applying Conditions on Aggregated Data

```
SELECT product,
    SUM(price * quantity) AS total_sales
FROM orders
WHERE price > 30000
GROUP BY product;
```

	product character varying (50)	total_sales numeric
1	Mobile	260000.00
2	Laptop	203000.00

Step 6: Conceptual Understanding of Filtering vs Aggregation Conditions

```
SELECT product,
    SUM(quantity * price) AS sales
FROM orders
WHERE order_date >= '2024-01-01'
AND order_date <= '2024-01-31'
GROUP BY product;
```

	character varying (50)	numeric
1	Headphones	1200.00
2	Laptop	112000.00
3	Mobile	48000.00

Learning Outcomes

- Understand how conditional filtering is used to retrieve only relevant records from a database.
- Explain how sorting enhances the readability and usefulness of query results in reports.
- Apply grouping techniques to organize data for analytical and summary purposes.
- Distinguish clearly between row-level conditions and group-level conditions using appropriate sql clauses.
- Develop confidence in writing analytical sql queries applicable to real-world database scenarios.
- Demonstrate improved readiness for placement and interview questions related to filtering, grouping, and aggregation concepts.