UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Branch: MCA (Data Science)	**Semester:** 2nd
Student Name: Drishti Bhatia	**UID:** 25MCD10008
Subject Name: Technical Training - I Lab	**Subject Code**: 25CAP-652
Section/Group: 25MCD-1(A)	**Date of Performance:** 21-01-2026

Experiment No.: 2

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

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. **S/W Requirement:** Oracle Database Express Edition and PGAdmin.

3. **Objectives:**
   Implementation of SELECT Queries with Filtering, Grouping and Sorting in PostgreSQL
- 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. **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. Code:

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

INSERT INTO customer_orders (order_id,customer_name, product, quantity, price, order_date) VALUES
(101,'Aman', 'Laptop', 1, 55000, '2024-01-10'),
(102,'Ridhi', 'Mobile', 2, 30000, '2024-01-12'),
(103,'Mohan', 'Laptop', 1, 60000, '2024-01-15'),
(104,'Sakshi', 'Tablet', 3, 15000, '2024-01-18'),
(105,'Gaurav', 'Mobile', 1, 25000, '2024-01-20'),
(106,'Pihu', 'Laptop', 2, 52000, '2024-01-22');
```

```
SELECT * FROM customer_orders;

-- Display high-priced orders
SELECT *
FROM customer_orders
WHERE price > 30000;

-- Filter based on product and price
SELECT customer_name, product, price
FROM customer_orders
WHERE product = 'Laptop' AND price > 55000;

-- Sort by price (ascending)
SELECT customer_name, product, price
FROM customer_orders
ORDER BY price ASC;

-- Sort by price (descending)
SELECT customer_name, product, price
FROM customer_orders
ORDER BY price DESC;

-- Sorting by multiple attributes
SELECT customer_name, product, price
FROM customer_orders
ORDER BY product ASC, price DESC;

-- Total sales per product
SELECT product, SUM(quantity * price) AS total_sales
FROM customer_orders
GROUP BY product;

-- Average price per product
SELECT product, AVG(price) AS avg_price
FROM customer_orders
GROUP BY product;

-- Products with total sales greater than 1,00,000
SELECT product, SUM(quantity * price) AS total_sales
FROM customer_orders
GROUP BY product
HAVING SUM(quantity * price) > 100000;

SELECT product, SUM(quantity * price) AS total_sales
FROM customer_orders
WHERE order_date >= '2024-01-15'
GROUP BY product
HAVING SUM(quantity * price) > 50000;
```

6. **Output:**

Select query:

	order_id [PK] integer	customer_name character varying (50)	product character varying (50)	quantity integer	price numeric (10,2)	order_date date
1	101	Aman	Laptop	1	55000.00	2024-01-10
2	102	Ridhi	Mobile	2	30000.00	2024-01-12
3	103	Mohan	Laptop	1	60000.00	2024-01-15
4	104	Sakshi	Tablet	3	15000.00	2024-01-18
5	105	Gaurav	Mobile	1	25000.00	2024-01-20
6	106	Pihu	Laptop	2	52000.00	2024-01-22

Display high priced orders:

	order_id [PK] integer	customer_name character varying (50)	product character varying (50)	quantity integer	price numeric (10,2)	order_date date
1	101	Aman	Laptop	1	55000.00	2024-01-10
2	103	Mohan	Laptop	1	60000.00	2024-01-15
3	106	Pihu	Laptop	2	52000.00	2024-01-22

Filter based on product and price:

	customer_name character varying (50)	product character varying (50)	price numeric (10,2)
1	Mohan	Laptop	60000.00

Sort by price (ascending):

	customer_name character varying (50)	product character varying (50)	price numeric (10,2)
1	Sakshi	Tablet	15000.00
2	Gaurav	Mobile	25000.00
3	Ridhi	Mobile	30000.00
4	Pihu	Laptop	52000.00
5	Aman	Laptop	55000.00
6	Mohan	Laptop	60000.00

Sort by price (descending):

	customer_name character varying (50)	product character varying (50)	price numeric (10,2)
1	Mohan	Laptop	60000.00
2	Aman	Laptop	55000.00
3	Pihu	Laptop	52000.00
4	Ridhi	Mobile	30000.00
5	Gaurav	Mobile	25000.00
6	Sakshi	Tablet	15000.00

Sorting by multiple attributes:

	customer_name character varying (50)	product character varying (50)	price numeric (10,2)
1	Mohan	Laptop	60000.00
2	Aman	Laptop	55000.00
3	Pihu	Laptop	52000.00
4	Ridhi	Mobile	30000.00
5	Gaurav	Mobile	25000.00
6	Sakshi	Tablet	15000.00

Total sales per product:

	product character varying (50)	total_sales numeric
1	Mobile	85000.00
2	Tablet	45000.00
3	Laptop	219000.00

Average price per product:

	product character varying (50)	avg_price numeric
1	Mobile	27500.000000000000
2	Tablet	15000.0000000000000000
3	Laptop	55666.666666666667

Products with total sales greater than 1,00,000:

	product character varying (50)	total_sales numeric
1	Laptop	219000.00

Using where , group by and having together:

	product character varying (50)	total_sales numeric
1	Laptop	164000.00

7. Learning Outcomes:

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