Worksheet 2

Student Name:Suyash **UID:25MCI10054**
Branch:MCA (AI&ML) **Section/Group:MAM-1 A**
Semester:2nd **Date of Performance:13/01/2026**
Subject Name:- DBMS LAB **Subject Code:**

1. Aim of the Session

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

2. Software Requirements

- PostgreSQL (Database Server)
- pgAdmin
- Windows Operating System

3. Objective of the Session

After completing this practical, the student will be able to:

- 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

- Create a sample table representing customer orders
- Insert realistic records into the table
- Retrieve filtered data using WHERE clause
- Sort query results using ORDER BY

- Group records and apply aggregate functions
- Apply conditions on grouped data using HAVING
- Analyze execution order of WHERE and HAVING clauses

5. Procedure of the Practical

(i) Start the system and log in to the computer.

(ii) Open PostgreSQL software.

iii) Create and select the database.

(iv) Create table using DDL command.

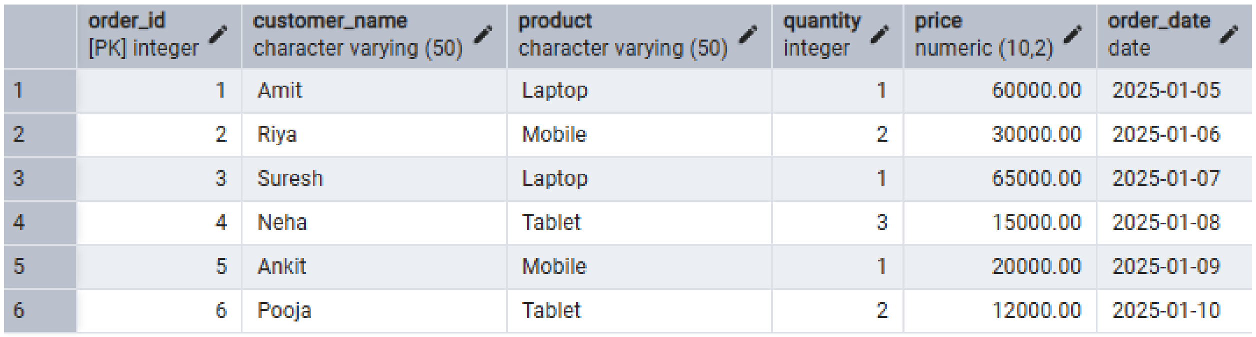
```
CREATE TABLE customer_orders (
    order_id SERIAL PRIMARY KEY,
    customer_name VARCHAR(50),
    product VARCHAR(50),
    quantity INT,
    price NUMERIC(10,2),
    order_date DATE
);
```

(v) Insert records into the table.

```
INSERT INTO customer_orders (customer_name, product, quantity, price, order_date) VALUES
('Amit', 'Laptop', 1, 60000, '2025-01-05'),
('Riya', 'Mobile', 2, 30000, '2025-01-06'),
('Suresh', 'Laptop', 1, 65000, '2025-01-07'),
('Neha', 'Tablet', 3, 15000, '2025-01-08'),
('Ankit', 'Mobile', 1, 20000, '2025-01-09'),
('Pooja', 'Tablet', 2, 12000, '2025-01-10');
```

(vi) Display all records.

```
select * from customer_orders;
```



	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	60000.00	2025-01-05
2	2	Riya	Mobile	2	30000.00	2025-01-06
3	3	Suresh	Laptop	1	65000.00	2025-01-07
4	4	Neha	Tablet	3	15000.00	2025-01-08
5	5	Ankit	Mobile	1	20000.00	2025-01-09
6	6	Pooja	Tablet	2	12000.00	2025-01-10

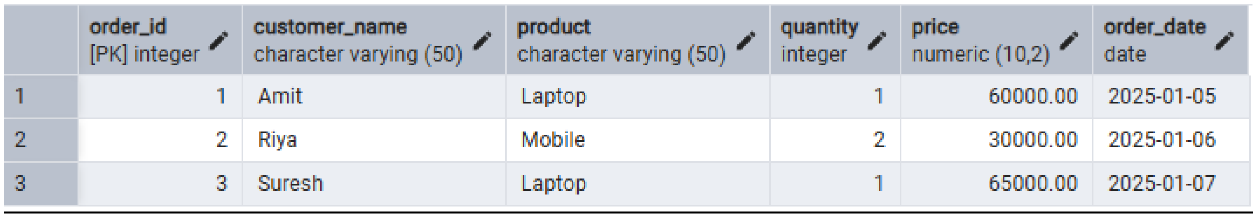
(vii) Filtering Data Using WHERE clause.

Query(Without case Statment)

```
SELECT *

FROM customer_orders

WHERE price > 25000;
```

<u>Query with case statement:</u>

```
SELECT *

FROM customer_orders

WHERE

   CASE

      WHEN Price > 25000 THEN 1

      ELSE 0

   END = 1;
```



	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	60000.00	2025-01-05
2	2	Riya	Mobile	2	30000.00	2025-01-06
3	3	Suresh	Laptop	1	65000.00	2025-01-07

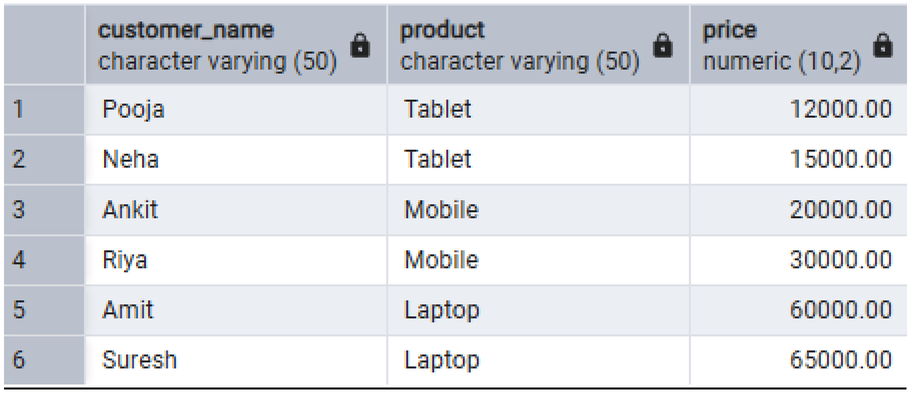
(viii) Sorting Query Results.

Ascending Order

SELECT customer_name, product, price

FROM customer_orders

ORDER BY price ASC;



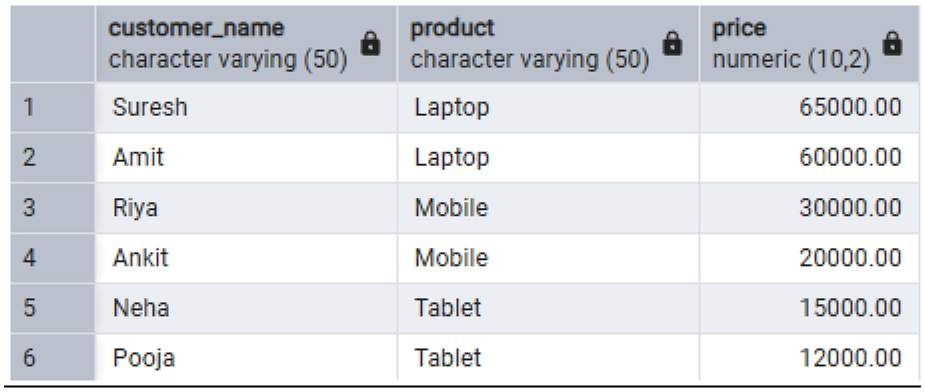
	customer_name character varying (50)	product character varying (50)	price numeric (10,2)
1	Pooja	Tablet	12000.00
2	Neha	Tablet	15000.00
3	Ankit	Mobile	20000.00
4	Riya	Mobile	30000.00
5	Amit	Laptop	60000.00
6	Suresh	Laptop	65000.00

Descending Order

SELECT customer_name, product, price

FROM customer_orders

ORDER BY price DESC;



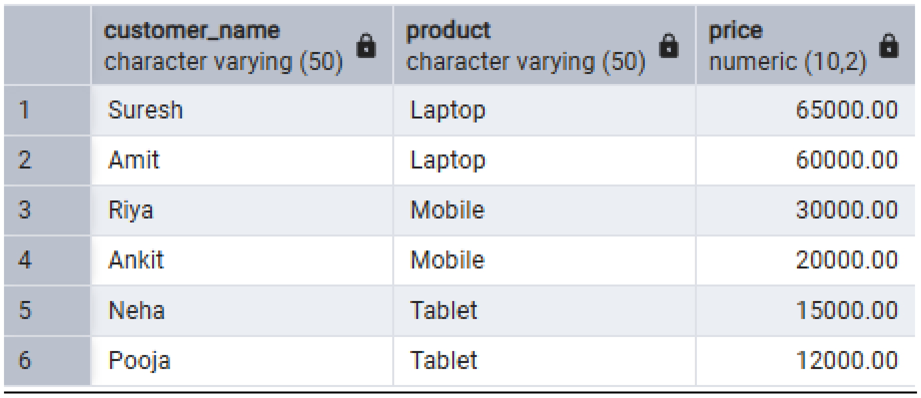
	customer_name character varying (50)	product character varying (50)	price numeric (10,2)
1	Suresh	Laptop	65000.00
2	Amit	Laptop	60000.00
3	Riya	Mobile	30000.00
4	Ankit	Mobile	20000.00
5	Neha	Tablet	15000.00
6	Pooja	Tablet	12000.00

Sort by Product, then Price

SELECT customer_name, product, price

FROM customer_orders

ORDER BY product ASC, price DESC;



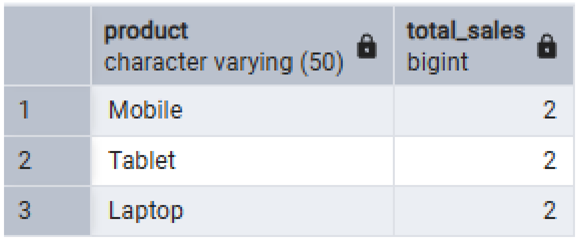
	customer_name character varying (50)	product character varying (50)	price numeric (10,2)
1	Suresh	Laptop	65000.00
2	Amit	Laptop	60000.00
3	Riya	Mobile	30000.00
4	Ankit	Mobile	20000.00
5	Neha	Tablet	15000.00
6	Pooja	Tablet	12000.00

(ix) Grouping Data for Aggregation.

SELECT product, Count(*) AS total_sales

FROM customer_orders

GROUP BY product;



	product character varying (50)	total_sales bigint
1	Mobile	2
2	Tablet	2
3	Laptop	2

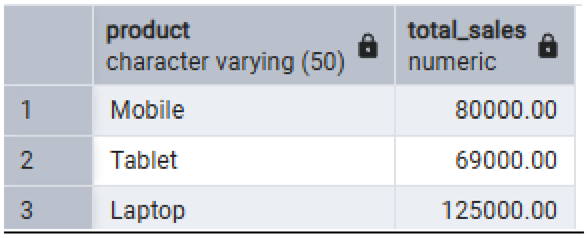
(x) Applying conditions on aggregated data (HAVING).

SELECT product, SUM(price * quantity) AS total_sales

FROM customer_orders

GROUP BY product

HAVING SUM(price * quantity) > 50000;



	product character varying (50)	total_sales numeric
1	Mobile	80000.00
2	Tablet	69000.00
3	Laptop	125000.00

(xi) Using WHERE and HAVING together.

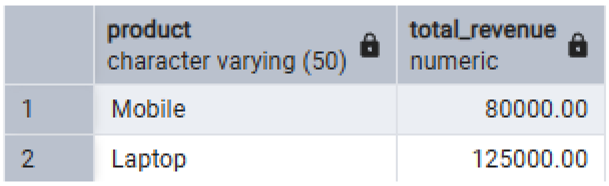
select product, sum(quantity*price) as total_revenue

from customer_orders

where order_date >= '2025-01-01'

group by product

having sum(quantity*price) > 50000;



	product character varying (50)	total_revenue numeric
1	Mobile	80000.00
2	Laptop	125000.00

(x) Incorrect usage:

SELECT product, SUM(price)

FROM customer_orders

WHERE SUM(price) > 50000

GROUP BY product;

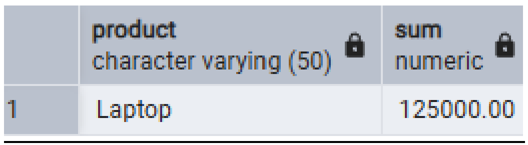
Correct Usage:

SELECT product, SUM(price)

FROM customer_orders

GROUP BY product

HAVING SUM(price) > 50000;



	product character varying (50)	sum numeric
1	Laptop	125000.00

6. I/O Analysis (Input / Output)

Input:

- Customer order details
- Filtering, sorting, grouping, and aggregation queries

Output:

- Filtered customer records
- Sorted result sets
- Group-wise sales summary
- Aggregated revenue reports

(Screenshots of execution and output attached)

7. Learning Outcomes

- Understand how to create relational database tables using appropriate data types and constraints
- Learn to retrieve required data from a table using row-level filtering with the WHERE clause.
- Gain the ability to apply column-level (group-level) filtering using the HAVING clause.
- Develop practical knowledge of using CASE statements for conditional logic in SQL queries.
- Understand the use of aggregate functions such as SUM(), AVG(), and COUNT() for analytical reporting.
- Clearly differentiate between row-level filtering and group-level filtering, and apply them correctly in real-world SQL scenarios.