UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+
Accredited University

Experiment- 2

Branch: MCA (AI&ML)	**Semester: 2**
Student Name: Ajay Rakwal	**UID:** 25MCI10329
Subject Name: Technical Training	**Subject Code**: 25CAP-652
Section/Group: MAM-1(A)	**Date of Performance:** 20-01-2026

Aim of the program:

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

- **Operating System:** Windows / Linux
- **Database Management System:** MySQL / Oracle / PostgreSQL
- **SQL Interface:** MySQL Workbench /Web Based / pgAdmin

Objective

- To study and implement SQL SELECT queries using PostgreSQL
- To retrieve required data by applying appropriate filtering conditions
- To organize query results using sorting in ascending and descending order
- To analyze data by grouping records and applying aggregate functions
- To apply conditions on grouped data using the HAVING clause and understand the difference between row-level and group-level filtering

Procedure of the Practical

i. Start the PostgreSQL server.
ii. Open the PostgreSQL client tool such as **pgAdmin** or **psql**.
iii. Create a new database for the experiment using the SQL command: CREATE DATABASE practical_db;
iv. Connect to the created database
v. Create a table to store customer order details with attributes such as customer name, product, quantity, price, and order date.
vi. Insert sufficient sample records into the table to allow meaningful data analysis.
vii. Execute SQL SELECT queries with WHERE clause to filter records based on given conditions.
viii. Execute ORDER BY queries to sort the results in ascending and descending order and using multiple columns.
ix. Execute GROUP BY queries along with aggregate functions to summarize data.
x. Verify the output after executing each query.
xi. Save the work and take screenshots of the executed queries and outputs for record submission.

Practical / Experiment Steps

```
-- Create table
CREATE TABLE orders (
    cust_name VARCHAR(50),
    product VARCHAR(50),
    quantity INT,
    price INT,
    order_date DATE);


-- Insert sample records
```

```
INSERT INTO orders VALUES
('Ajay', 'Laptop', 1, 60000, '2025-01-10'),
('Jatin', 'Mobile', 2, 25000, '2025-01-12'),
('Harkirat', 'Laptop', 1, 70000, '2024-01-15'),
('Purnima', 'Tablet', 3, 15000, '2025-01-18'),
('Rohit', 'Mobile', 1, 30000, '2025-01-20'),
('Neha', 'Tablet', 2, 14000, '2025-01-22');

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

-- Sort by price (ascending)
SELECT cust_name, product, price
FROM orders
ORDER BY price ASC;

-- Sort by price (descending)
SELECT cust_name, product, price
FROM orders
ORDER BY price DESC;

-- Sort by product, then price
SELECT cust_name, product, price
```

```
FROM orders

ORDER BY product ASC, price DESC;


-- Total sales per product

SELECT product, SUM(quantity * price) AS total_sales

FROM orders

GROUP BY product;


--Step 5

SELECT product, SUM(quantity*price) AS total_sales

FROM orders

GROUP BY product

HAVING SUM(quantity*price) > 75000;


-- Step 6 Products with total sales greater than 75000

SELECT product, SUM(quantity * price) AS total_sales

FROM orders

WHERE order_date >= '2025-01-01'

GROUP BY product

HAVING SUM(quantity * price) > 75000;
```

I/O Analysis (Input / Output)

Input

- Orders table creation query

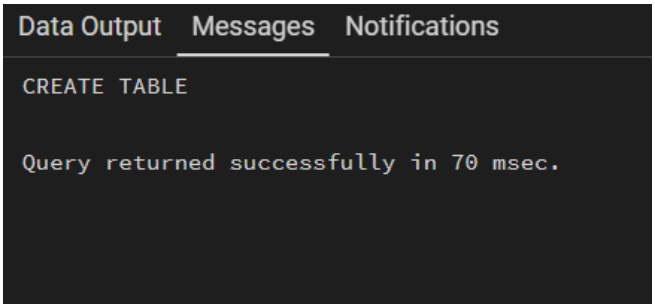
- Sample records inserted into the Orders table using INSERT commands
- SELECT queries to display orders with high prices
- ORDER BY queries to sort records in ascending and descending order
- GROUP BY queries to calculate total sales per product
- HAVING clause queries to filter products based on total sales

Output

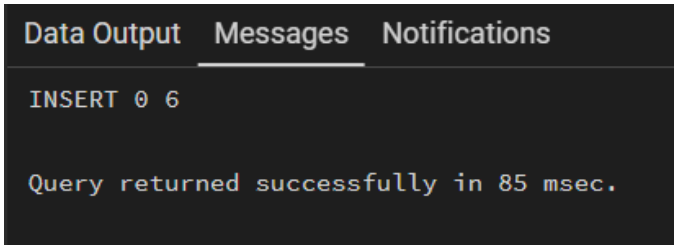
- Orders table created successfully
- Records inserted and stored correctly in the table
- High-priced orders displayed accurately based on the given condition
- Records sorted correctly by price and product
- Total sales calculated correctly for each product
- Products with total sales greater than the specified value displayed successfully

OUTPUT:

Table Created:



Insert sample records:



Display high priced orders

Data Output | Messages | Notifications

	cust_name character varying (50)	product character varying (50)	quantity integer	price integer	order_date date
1	Ajay	Laptop	1	60000	2025-01-10
2	Harkirat	Laptop	1	70000	2025-01-15

Sort by price (ascending)

Data Output | Messages | Notifications

	cust_name character varying (50)	product character varying (50)	price integer
1	Neha	Tablet	14000
2	Purnima	Tablet	15000
3	Jatin	Mobile	25000
4	Rohit	Mobile	30000
5	Ajay	Laptop	60000
6	Harkirat	Laptop	70000

Sort by price (descending)

Data Output | Messages | Notifications

	cust_name character varying (50)	product character varying (50)	price integer
1	Harkirat	Laptop	70000
2	Ajay	Laptop	60000
3	Rohit	Mobile	30000
4	Jatin	Mobile	25000
5	Purnima	Tablet	15000
6	Neha	Tablet	14000

Sort by product, then price

Data Output | Messages | Notifications

	cust_name character varying (50)	product character varying (50)	price integer
1	Harkirat	Laptop	70000
2	Ajay	Laptop	60000
3	Rohit	Mobile	30000
4	Jatin	Mobile	25000
5	Purnima	Tablet	15000
6	Neha	Tablet	14000

Total sales per product

Data Output | Messages | Notifications

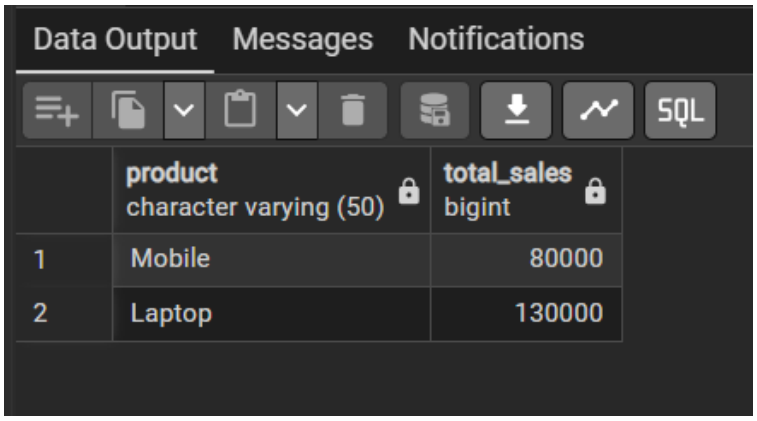
	product character varying (50)	total_sales bigint
1	Mobile	80000
2	Tablet	73000
3	Laptop	130000

Products with total sales greater than 75000

Data Output | Messages | Notifications

	product character varying (50)	total_sales bigint
1	Mobile	80000
2	Laptop	130000

Implementing Step 6 by applying WHERE and HAVING



	product character varying (50)	total_sales bigint
1	Mobile	80000
2	Laptop	130000

Learning Outcome:

- Understood the use of SQL SELECT queries in PostgreSQL
- Learned how to filter records using the WHERE clause
- Gained knowledge of sorting data using ORDER BY
- Understood grouping of data using the GROUP BY clause
- Learned to apply aggregate functions for data analysis
- Gained practical experience in executing and analyzing SQL queries.