UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Worksheet 2

Student Name:Gaurang Garg **UID:25MCI1079**
Branch: MCA(AI/ML) **Section/Group:1/A**
Semester: II **Date of Performance:13/01/2026**
Subject Name:- Technical Training **Subject Code: 25CAP-652**

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. Objective of the Session

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

- Retrieve specific data using filtering conditions
- Sort query results using single and multiple attributes
- Perform aggregation using grouping techniques
- Apply conditions on aggregated data using HAVING clause
- Understand real-world analytical queries commonly asked in placement interviews

3. 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

4. Procedure of the Practical

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

(ii) Open PostgreSQL software. **iii) Create and select the database.**

create database CompanyDB;

(iv) Create table using DDL command.

```
create table
customer_orders( order_id
serial primary key,
customer_name varchar(20),
product varchar(20), 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
('Aman',  'Laptop', 1, 55000.00, '2025-01-10'),
('Rohit', 'Mobile', 2, 20000.00, '2025-01-11'),
('Neha',  'Tablet',1, 30000.00, '2025-01-12'),
('Priya', 'Laptop',1, 56000.00, '2025-01-13'),
('Rahul', 'Mobile', 3, 18000.00, '2025-01-14'),
('Kiran', 'Headphone',2, 3000.00, '2025-01-15'),
('Suman', 'Keyboard', 1,1500.00, '2025-01-16'),
('Ankit', 'Mouse',  2,800.00, '2025-01-17');
```

(vi) Display all records. select * from customer_orders;

	order_id [PK] integer	customer_name character varying (20)	product character varying (20)	quantity integer	price numeric (10,2)	order_date date
1	1	Aman	Laptop	1	55000.00	2025-01-10
2	2	Rohit	Mobile	2	20000.00	2025-01-11
3	3	Neha	Tablet	1	30000.00	2025-01-12
4	4	Priya	Laptop	1	56000.00	2025-01-13
5	5	Rahul	Mobile	3	18000.00	2025-01-14
6	6	Kiran	Headphone	2	3000.00	2025-01-15
7	7	Suman	Keyboard	1	1500.00	2025-01-16
8	8	Ankit	Mouse	2	800.00	2025-01-17

(vii) Filtering Data Using WHERE clause.

select order_id, customer_name, product, quantity, price

from customer_orders where price > 20000;

	order_id [PK] integer	customer_name character varying (20)	product character varying (20)	quantity integer	price numeric (10,2)
1	1	Aman	Laptop	1	55000.00
2	3	Neha	Tablet	1	30000.00
3	4	Priya	Laptop	1	56000.00

(viii) Sorting Query Results. Ascending Order

select order_id, customer_name, product, quantity, price

from customer_orders where price > 20000 order by

price;

	order_id [PK] integer	customer_name character varying (20)	product character varying (20)	quantity integer	price numeric (10,2)
1	3	Neha	Tablet	1	30000.00
2	1	Aman	Laptop	1	55000.00
3	4	Priya	Laptop	1	56000.00

Descending Order

select order_id, customer_name, product, quantity, price

from customer_orders where price > 20000 order by

price desc;

	order_id [PK] integer	customer_name character varying (20)	product character varying (20)	quantity integer	price numeric (10,2)
1	4	Priya	Laptop	1	56000.00
2	1	Aman	Laptop	1	55000.00
3	3	Neha	Tablet	1	30000.00

(ix) Grouping Data for Aggregation.

select product, count(*) as total_product_sale

from customer_orders group by product;

	product character varying (20)	total_product_sale bigint
1	Mobile	2
2	Mouse	1
3	Tablet	1
4	Headphone	1
5	Keyboard	1
6	Laptop	2

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

```
select product,
sum(quantity*price) as total_revenue
from customer_orders group by
product having sum(quantity*price)
> 50000;
```

	product character varying (20)	total_revenue numeric
1	Mobile	94000.00
2	Laptop	111000.00

(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 (20)	total_revenue numeric
1	Mobile	94000.00
2	Laptop	111000.00

5. 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)

6. Learning Outcomes

- Students understand how data can be filtered to retrieve only relevant records.
- Students learn how sorting improves readability and usefulness of reports.
- Students gain the ability to group data for analytical purposes.
- Students clearly differentiate between WHERE and HAVING clauses.
- Students develop confidence in writing analytical SQL queries.
- Students are better prepared for SQL-based placement and interview questions.