CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

Worksheet No. 2

Student Name: ANINDITA DHAR **UID:** 25MCA20259
Branch: MCA (GENERAL) **Section/Group:** MCA- 1-A
Semester: II **Date of Performance:** 17/01/2026

Subject Name: Technical Training **Subject Code:** 25CAP-652

1.Aim/Overview of the practical:

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

2. Objectives:

After completing this practical, the student will be able

- 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.

3. Input/Apparatus Used:

- PostgreSQL
- pgAdmin

4. Practical / Experiment Steps

- Create a sample table representing customer orders
- Insert realistic records into the table
- Retrieve filtered data using the WHERE clause
- Sort query results using ORDER BY
- Group records and apply aggregate functions
- Apply conditions on grouped data using HAVING
- Analyse execution order of WHERE and HAVING clauses

4.Procedure/Algorithm/Code:

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 a table using the 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 named customer_orders.

```
insert into customer_orders(customer_name,product,quantity,price,order_date)
values
('Amit', 'Laptop', 1, 55000, '2025-01-05'),
('Amit', 'Mouse', 2, 800, '2025-01-06'),
('Riya', 'Mobile', 1, 22000, '2025-01-10'),
('Riya', 'Headphones', 1, 2000, '2025-01-10'),
('Karan', 'Laptop', 1, 60000, '2025-02-02'),
('Karan', 'Keyboard', 1, 1500, '2025-02-05'),
('Neha', 'Mobile', 2, 21000, '2025-02-15'),
('Neha', 'Charger', 3, 900, '2025-02-18');
```

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	Amit	Laptop	1	55000.00	2025-01-05
2	2	Amit	Mouse	2	800.00	2025-01-06
3	3	Riya	Mobile	1	22000.00	2025-01-10
4	4	Riya	Headphones	1	2000.00	2025-01-10
5	5	Karan	Laptop	1	60000.00	2025-02-02
6	6	Karan	Keyboard	1	1500.00	2025-02-05
7	7	Neha	Mobile	2	21000.00	2025-02-15
8	8	Neha	Charger	3	900.00	2025-02-18

vii. Filtering table data using the 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	Amit	Laptop	1	55000.00
2	3	Riya	Mobile	1	22000.00
3	5	Karan	Laptop	1	60000.00
4	7	Neha	Mobile	2	21000.00

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	7	Neha	Mobile	2	21000.00
2	3	Riya	Mobile	1	22000.00
3	1	Amit	Laptop	1	55000.00
4	5	Karan	Laptop	1	60000.00

(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	5	Karan	Laptop	1	60000.00
2	1	Amit	Laptop	1	55000.00
3	3	Riya	Mobile	1	22000.00
4	7	Neha	Mobile	2	21000.00

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	Charger	1
2	Mobile	2
3	Mouse	1
4	Keyboard	1
5	Laptop	2
6	Headphones	1

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	64000.00
2	Laptop	115000.00

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	64000.00
2	Laptop	115000.00

5. I/O Analysis (Input / Output Analysis)

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 from a database.
- Students learn how sorting improves the readability and usefulness of query results in reports.
- Students gain the ability to group data for analytical purposes.
- Students clearly differentiate between row-level conditions and group-level conditions.
- Students develop confidence in writing analytical SQL queries used in real-world scenarios.
- Students are better prepared to answer SQL-based placement and interview questions related to filtering, grouping, and aggregation