

<b>Branch:</b> <i>MCA (Data Science)</i>	<b>Semester:</b> <i>2</i>
<b>Student Name:</b> <i>Adarsh Mishra</i>	<b>UID:</b> <i>25MCD10065</i>
<b>Subject Name:</b> <i>Technical Training - I</i>	<b>Subject Code:</b> <i>25CAP-652</i>
<b>Section/Group:</b> <i>25MCD – 1(A)</i>	<b>Date of Performance:</b> <i>27 January, 2026</i>

## **Experiment No. : 3**

### **1. Aim/Overview of the practical:**

To implement conditional decision-making logic in PostgreSQL using IF–ELSE constructs and CASE expressions for classification, validation, and rule-based data processing.

### **2. Objective:**

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

- To understand conditional execution in SQL
- To implement decision-making logic using CASE expressions
- To simulate real-world rule validation scenarios
- To classify data based on multiple conditions

### **3. Software Used:**

- PostgreSQL
- pgAdmin4

### **4. Code for experiment/Practical:**

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

```
INSERT INTO customer_orders (order_id, customer_name,
product, quantity, price, order_date) VALUES
(1001,'Amit', 'Laptop', 1, 55000, '2024-01-10'),
(2002,'Neha', 'Mobile', 2, 30000, '2024-01-12'),
(3003,'Ravi', 'Laptop', 1, 60000, '2024-01-15'),
```

```
(4004,'Priya', 'Tablet', 3, 45000, '2024-01-18'),
(5005,'Karan', 'Mobile', 1, 15000, '2024-01-20'),
(6006,'Anita', 'Laptop', 2, 110000, '2024-01-22');
```

```
SELECT *
FROM customer_orders
WHERE price > 40000;
```

```
SELECT customer_name, product, price
FROM customer_orders
ORDER BY price ASC;
```

```
SELECT customer_name, product, price
FROM customer_orders
ORDER BY price DESC;
```

```
SELECT customer_name, product, price
FROM customer_orders
ORDER BY product ASC, price DESC;
```

```
SELECT product, SUM(price) AS total_sales
FROM customer_orders
GROUP BY product;
```

```
SELECT product, SUM(price) AS total_sales
FROM customer_orders
GROUP BY product
HAVING SUM(price) > 70000;
```

## 5. Result/Output/Writing Summary:

	order_id [PK] integer	customer_name character varying (50)	product character varying (50)	quantity integer	price numeric (10,2)	order_date date
1	1001	Amit	Laptop	1	55000.00	2024-01-10
2	3003	Ravi	Laptop	1	60000.00	2024-01-15
3	4004	Priya	Tablet	3	45000.00	2024-01-18
4	6006	Anita	Laptop	2	110000.00	2024-01-22

	<b>customer_name</b> character varying (50) 🔒	<b>product</b> character varying (50) 🔒	<b>price</b> numeric (10,2) 🔒
1	Anita	Laptop	110000.00
2	Ravi	Laptop	60000.00
3	Amit	Laptop	55000.00
4	Priya	Tablet	45000.00
5	Neha	Mobile	30000.00
6	Karan	Mobile	15000.00

	<b>customer_name</b> character varying (50) 🔒	<b>product</b> character varying (50) 🔒	<b>price</b> numeric (10,2) 🔒
1	Anita	Laptop	110000.00
2	Ravi	Laptop	60000.00
3	Amit	Laptop	55000.00
4	Neha	Mobile	30000.00
5	Karan	Mobile	15000.00
6	Priya	Tablet	45000.00

	<b>product</b> character varying (50) 🔒	<b>total_sales</b> numeric 🔒
1	Mobile	45000.00
2	Tablet	45000.00
3	Laptop	225000.00

	<b>product</b> character varying (50) 🔒	<b>total_sales</b> numeric 🔒
1	Laptop	225000.00

## **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)

### **Learning outcomes (What I have learnt):**

- 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.