UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

EXPERIMENT 3

Student Name: Jatin Tasoria **UID: 25MCC20054**

Branch: MCA-Cloud Computing **Section/Group: MCC 101-A**

Semester: 2nd **Date of Performance: 27/01/26**

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

1. Aim

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 learner 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
- To strengthen SQL logic skills required in interviews and backend systems

3. Practical / Experiment Steps

1. Create a table to store schema violation details
2. Insert sample data with different violation counts
3. Use CASE expressions to classify violation severity

4. Apply CASE logic inside UPDATE statements
5. Implement IF–ELSE logic using PL/pgSQL blocks
6. Create a grading system using conditional logic
7. Use CASE for custom sorting based on priority

4. Procedure of the Practical

1. Start PostgreSQL and connect to the database
2. Create required tables
3. Insert sample records
4. Execute SELECT queries using CASE expressions
5. Perform UPDATE operations using CASE
6. Execute PL/pgSQL DO blocks using IF–ELSE logic
7. Observe and verify output

5. SQL Queries Used in the Experiment

a. Table Creation and Sample Data

```
CREATE TABLE schema_violations (
    schema_id SERIAL PRIMARY KEY,
    schema_name VARCHAR(50),
    violation_count INT
);

INSERT INTO schema_violations (schema_name, violation_count)
VALUES
('Finance', 0),
```

('HR', 2),
('Sales', 5),
('Audit', 10);

b. Classifying Data Using CASE Expression

```
SELECT schema_name, violation_count,
  CASE
    WHEN violation_count = 0 THEN 'No Violation'
    WHEN violation_count BETWEEN 1 AND 3 THEN 'Minor Violation'
    WHEN violation_count BETWEEN 4 AND 6 THEN 'Moderate Violation'
    ELSE 'Critical Violation'
  END AS violation_status
FROM schema_violations;
```

c. Applying CASE Logic in UPDATE Statement

```
ALTER TABLE schema_violations
ADD COLUMN approval_status VARCHAR(20);

UPDATE schema_violations
```

```
SET approval_status =
  CASE
    WHEN violation_count = 0 THEN 'Approved'
    WHEN violation_count BETWEEN 1 AND 3 THEN 'Needs
Review'
    ELSE 'Rejected'
  END;
```

d. Implementing IF–ELSE Logic Using PL/pgSQL

```
DO $$
DECLARE
  v_count INT := 5;
BEGIN
  IF v_count = 0 THEN
    RAISE NOTICE 'No violations found';
  ELSIF v_count BETWEEN 1 AND 3 THEN
    RAISE NOTICE 'Minor violations detected';
  ELSIF v_count BETWEEN 4 AND 6 THEN
    RAISE NOTICE 'Moderate violations detected';
  ELSE
    RAISE NOTICE 'Critical violations detected';
  END IF;
END $$;
```

e. Real-World Classification Scenario

```
CREATE TABLE students (
    student_id SERIAL PRIMARY KEY,
    student_name VARCHAR(50),
    marks INT
);

INSERT INTO students (student_name, marks) VALUES
('Amit', 85),
('Neha', 72),
('Rahul', 55),
('Sneha', 38);

SELECT
    student_name,
    marks,
    CASE
        WHEN marks >= 80 THEN 'A'
        WHEN marks >= 60 THEN 'B'
        WHEN marks >= 40 THEN 'C'
        ELSE 'Fail'
    END AS grade
FROM students;
```

f. Using CASE for Custom Sorting

```
SELECT
   schema_name,
   violation_count
FROM schema_violations
ORDER BY
   CASE
      WHEN violation_count = 0 THEN 1
      WHEN violation_count BETWEEN 1 AND 3 THEN 2
      WHEN violation_count BETWEEN 4 AND 6 THEN 3
      ELSE 4
   END;
```

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

SQL queries for:

- Table creation

```
CREATE TABLE

Query returned successfully in 46 msec.
```

- Data Insertion

```
INSERT 0 4

Query returned successfully in 80 msec.
```

- Classifying Data Using CASE Expression

	schema_name character varying (50)	violation_count integer	violation_status text
1	Finance	0	No Violation
2	HR	2	Minor Violation
3	Sales	5	Moderate Violati...
4	Audit	10	Critical Violation

- Altering Table

```
ALTER TABLE

Query returned successfully in 52 msec.
```

- Updating

```
UPDATE 4

Query returned successfully in 57 msec.
```

- After altering and updating

	schema_id [PK] integer	schema_name character varying (50)	violation_count integer	approval_status character varying (20)
1	1	Finance	0	Approved
2	2	HR	2	Needs Review
3	3	Sales	5	Rejected
4	4	Audit	10	Rejected

- IF–ELSE Logic Using PL/pgSQL

```
NOTICE:  Moderate violations detected
DO

Query returned successfully in 58 msec.
```

- Real-World Classification Scenario

	student_name character varying (50)	marks integer	grade text
1	Amit	85	A
2	Neha	72	B
3	Rahul	55	C
4	Sneha	38	Fail

- Using CASE for Custom Sorting

	schema_name character varying (50)	violation_count integer
1	Finance	0
2	HR	2
3	Sales	5
4	Audit	10

7. Learning Outcome

- Understood searched CASE and IF–ELSE constructs
- Learned real-world data classification techniques

- Implemented backend-level rule validation
- Reduced dependency on application-side logic
- Gained SQL skills commonly tested in interviews