UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+
Accredited University

Experiment- 3

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:** 27-01-2026

Aim of the program:

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

Software Requirements

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

Objective

- 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

Procedure of the Practical

i. Start the PostgreSQL server.

ii. Open pgAdmin or psql client.

iii. Create a new database for the experiment.

iv. Connect to the database.

v. Create required tables to store schema and violation details.

vi. Insert sample records with varying violation counts.

vii. Apply CASE expressions to classify data.

viii. Use CASE logic inside UPDATE statements.

ix. Implement IF–ELSE logic using PL/pgSQL blocks.

x. Execute queries and verify outputs.

xi. Save results and capture screenshots for submission.

Practical / Experiment Steps

```
-- Create table

CREATE TABLE schema_violations (

    schema_id INT PRIMARY KEY,

    schema_name VARCHAR(50),

    violation_count INT

);

-- Insert sample records

INSERT INTO schema_violations VALUES

(1, 'Finance_Schema', 0),

(2, 'HR_Schema', 2),

(3, 'Sales_Schema', 5),

(4, 'Audit_Schema', 9),

(5, 'Admin_Schema', 12);
```

```
-- 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 7 THEN 'Moderate Violation'
    ELSE 'Critical Violation'
END AS violation_status
FROM schema_violations;

-- 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 5 THEN 'Needs Review'
    ELSE 'Rejected'
END;

-- Implementing IF–ELSE Logic Using PL/pgSQL
DO $$
DECLARE
```

```
    v_count INT := 6;
BEGIN
    IF v_count = 0 THEN
        RAISE NOTICE 'No violations detected';
    ELSIF v_count <= 5 THEN
        RAISE NOTICE 'Minor violations found';
    ELSE
        RAISE NOTICE 'Critical violations found';
    END IF;
END $$;

-- Real-World Classification Scenario (Grading System)
CREATE TABLE student_marks (
    student_name VARCHAR(50),
    marks INT
);

INSERT INTO student_marks VALUES
('Ajay', 85),
('Neha', 72),
('Rohit', 61),
('Purnima', 48);

-- using CASE on Step 5
```

```
SELECT student_name, marks,
CASE
    WHEN marks >= 80 THEN 'A'
    WHEN marks >= 65 THEN 'B'
    WHEN marks >= 50 THEN 'C'
    ELSE 'Fail'
END AS grade
FROM student_marks;


-- 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 7 THEN 3
    ELSE 4
END;
```

I/O Analysis (Input / Output)

Input

- Table creation queries
- Sample data insertion commands

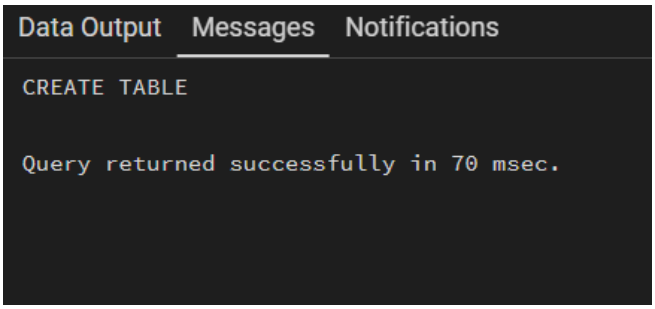
- CASE expression queries for classification
- UPDATE queries using CASE logic
- PL/pgSQL block using IF–ELSE
- SELECT queries for grading and sorting

Output

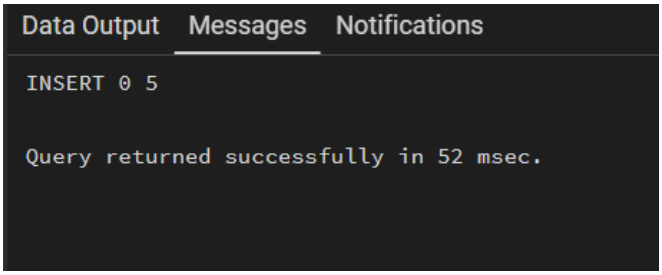
- Tables created successfully
- Records inserted correctly
- Data classified based on violation severity
- Approval status updated accurately
- IF–ELSE logic executed with correct messages
- Grades and sorted results displayed correctly

OUTPUT:

Table Created:



Insert sample records:



Classifying Data Using CASE Expression

Data Output | Messages | Notifications

	schema_name character varying (50)	violation_count integer	violation_status text
1	Finance_Schema	0	No Violation
2	HR_Schema	2	Minor Violation
3	Sales_Schema	5	Moderate Violati...
4	Audit_Schema	9	Critical Violation
5	Admin_Schema	12	Critical Violation

Applying CASE Logic in UPDATE Statement

Data Output | Messages | Notifications

```
UPDATE 5

Query returned successfully in 64 msec.
```

Implementing IF–ELSE Logic Using PL/pgSQL

Data Output | Messages | Notifications

```
NOTICE:  Critical violations found
DO

Query returned successfully in 58 msec.
```

Real-World Classification Scenario (Grading System)

Data Output | Messages | Notifications

```
INSERT 0 4

Query returned successfully in 58 msec.
```

using CAS

Data Output | Messages | Notifications

	student_name character varying (50)	marks integer	grade text
1	Ajay	85	A
2	Neha	72	B
3	Rohit	61	C
4	Purnima	48	Fail

Using CASE for Custom Sorting

Data Output | Messages | Notifications

	schema_name character varying (50)	violation_count integer
1	Finance_Schema	0
2	HR_Schema	2
3	Sales_Schema	5
4	Audit_Schema	9
5	Admin_Schema	12

Learning Outcome:

- Understood conditional logic in PostgreSQL
- Learned to use CASE expressions for classification
- Implemented IF–ELSE logic using PL/pgSQL
- Applied real-world decision-making rules in SQL
- Gained practical experience in backend rule validation and interview-oriented SQL logic