UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+

Student Name: Yash Karde **UID: 25MCI10090**
Branch: M.C.A(Al & ML) **Section/Group: 25MAM-1 A**
Semester: 2nd **Date of Performance:27/01/2026**
Subject Name: TECHINCAL SKILLS **Subject Code:**

WORKSHEET 3

AIM: To implement conditional decision-making logic in PostgreSQL using IF–ELSE constructs and CASE expressions for classification, validation, and rule-based data processing. S/W Requirement: Oracle Database Express Edition and pgAdmin

OBJECTIVES:

- To understand conditional execution in SQL
- To implement decision-making logic using CASE expressions
- To classify data based on multiple conditions
- To strengthen SQL logic skills required in interviews and backend systems
- To simulate real-world rule validation scenarios

Given:

Practical / Experiment Steps

Step 1: Classifying Data Using CASE ExpressionStart the PostgreSQL server.

- Retrieve schema names and their violation counts.
- Use conditional logic to classify each schema into categories such as:
    - No Violation
    - Minor Violation
    - Moderate Violation
    - Critical Violation

Querry:

Table Creation:

```
CREATE TABLE compliance_report (
   id SERIAL PRIMARY KEY,
   schema_name VARCHAR(50) NOT NULL,
    violation_count INT NOT NULL );
```

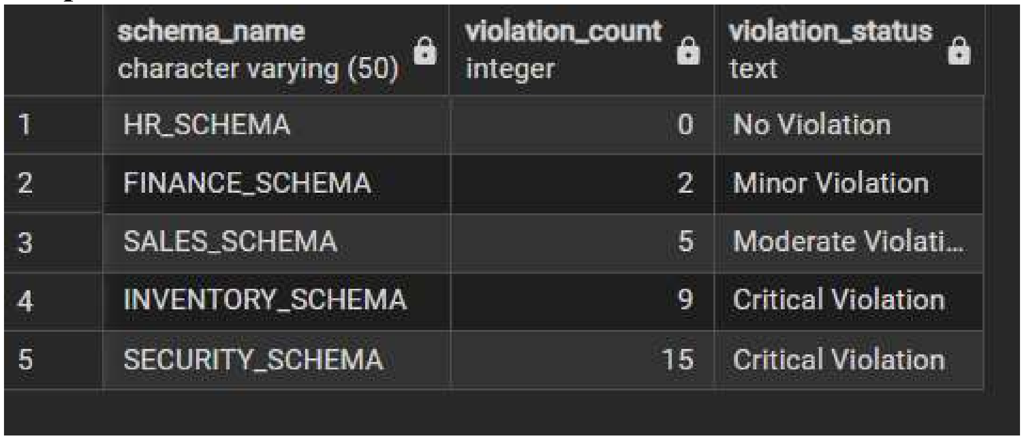
Insertion Of Records

```
INSERT INTO compliance_report (schema_name, violation_count) VALUES
('HR_SCHEMA', 0),
('FINANCE_SCHEMA', 2),
('SALES_SCHEMA', 5),
('INVENTORY_SCHEMA', 9),
('SECURITY_SCHEMA', 15);
```

Code :-

```
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 compliance_report;
```

Output:-



	schema_name character varying (50)	violation_count integer	violation_status text
1	HR_SCHEMA	0	No Violation
2	FINANCE_SCHEMA	2	Minor Violation
3	SALES_SCHEMA	5	Moderate Violati...
4	INVENTORY_SCHEMA	9	Critical Violation
5	SECURITY_SCHEMA	15	Critical Violation

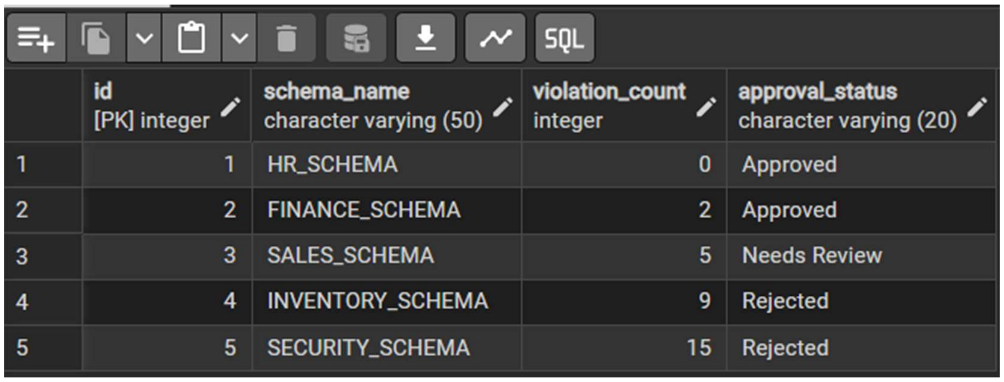
Step 2: Applying CASE Logic in Data Updates

- Add a new column to store approval status.
- Update this column based on violation count using conditional rules such as:
    - Approved
    - Needs Review
    - Rejected

Querry:-

```
UPDATE compliance_report
SET approval_status =
    CASE
        WHEN violation_count <= 2 THEN 'Approved'
        WHEN violation_count BETWEEN 3 AND 7 THEN 'Needs Review'
        ELSE 'Rejected'
    END;
```

Output:



	id [PK] integer	schema_name character varying (50)	violation_count integer	approval_status character varying (20)
1	1	HR_SCHEMA	0	Approved
2	2	FINANCE_SCHEMA	2	Approved
3	3	SALES_SCHEMA	5	Needs Review
4	4	INVENTORY_SCHEMA	9	Rejected
5	5	SECURITY_SCHEMA	15	Rejected

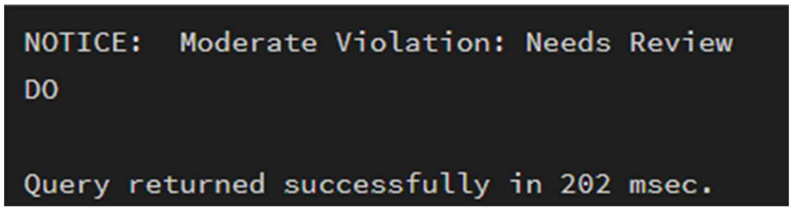
Step 3: Implementing IF–ELSE Logic Using PL/pgSQL

- Use a procedural block instead of a SELECT statement.
- Declare a variable representing violation count.
- Display different messages based on the value of the variable using IF–ELSE logic.

Querry:

```
DO $$
DECLARE
    v_violation_count INT := 5;
BEGIN
    IF v_violation_count = 0 THEN
        RAISE NOTICE 'No Violation: Schema is fully compliant';
    ELSIF v_violation_count BETWEEN 1 AND 3 THEN
        RAISE NOTICE 'Minor Violation: Approval Granted';
    ELSIF v_violation_count BETWEEN 4 AND 7 THEN
        RAISE NOTICE 'Moderate Violation: Needs Review';
    ELSE
        RAISE NOTICE 'Critical Violation: Approval Rejected';
    END IF;
END $$;
```

Output:



```
NOTICE:  Moderate Violation: Needs Review
DO

Query returned successfully in 202 msec.
```

Step 4: Real-World Classification Scenario (Grading System)

- Create a table to store student names and marks.
- Classify students into grades based on their marks using conditional logic.

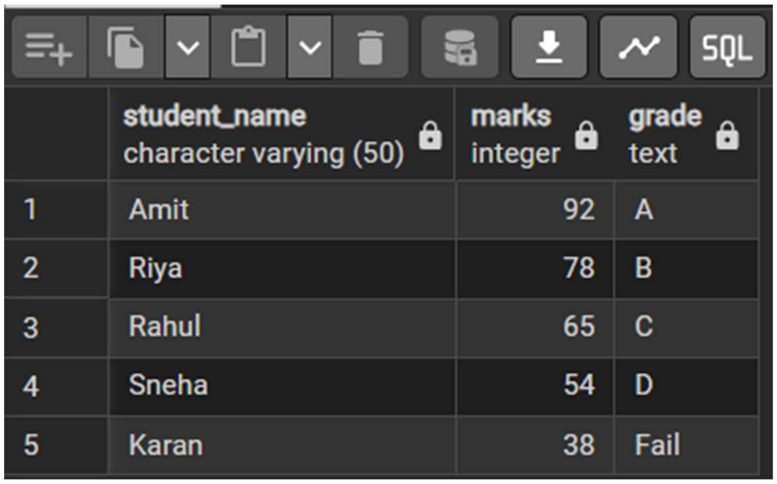
Querry:

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

```
INSERT INTO student_grades (student_name, marks) VALUES
('Amit', 92),
('Riya', 78),
('Rahul', 65),
('Sneha', 54),
('Karan', 38);

SELECT
    student_name,
    marks,
    CASE
        WHEN marks >= 90 THEN 'A'
        WHEN marks BETWEEN 75 AND 89 THEN 'B'
        WHEN marks BETWEEN 60 AND 74 THEN 'C'
        WHEN marks BETWEEN 40 AND 59 THEN 'D'
        ELSE 'Fail'
    END AS grade
FROM student_grades;
```

Output :-



	student_name character varying (50)	marks integer	grade text
1	Amit	92	A
2	Riya	78	B
3	Rahul	65	C
4	Sneha	54	D
5	Karan	38	Fail

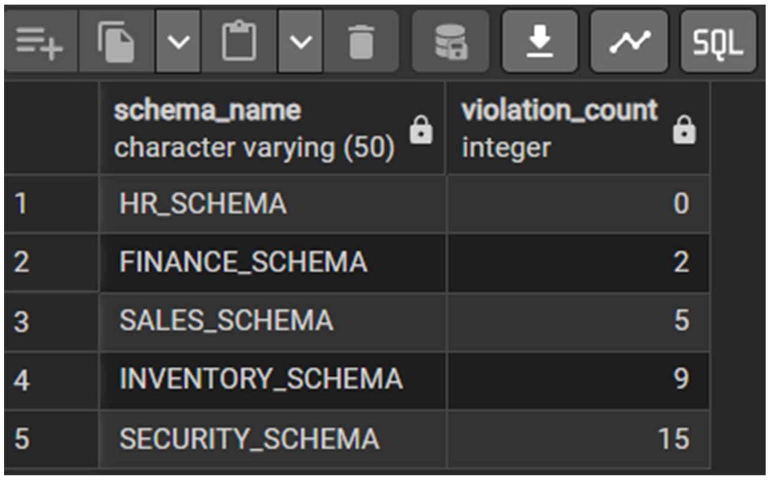
Step 5: Using CASE for Custom Sorting

- Retrieve schema details.
- Apply conditional priority while sorting records based on violation severity.

Querry:-

```
SELECT
   schema_name,
   violation_count
FROM compliance_report
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;
```

Output:-



	schema_name character varying (50)	violation_count integer
1	HR_SCHEMA	0
2	FINANCE_SCHEMA	2
3	SALES_SCHEMA	5
4	INVENTORY_SCHEMA	9
5	SECURITY_SCHEMA	15

Learning Outcomes:

1. Design and create database tables in PostgreSQL using appropriate data types and constraints.
2. Insert and manage multiple records representing real-world data scenarios.
3. Apply conditional logic using CASE expressions to classify and categorize data based on business rules.
4. Implement rule-based data validation and approval workflows using UPDATE statements with CASE.
5. Use PL/pgSQL procedural blocks to implement IF–ELSE decision-making logic with variables.
6. Display system messages using RAISE NOTICE for validation and status reporting.
7. Solve real-world classification problems such as compliance reporting and grading systems.
8. Perform custom sorting of records using CASE expressions in ORDER BY clauses.
9. Differentiate between CASE (SQL level) and IF–ELSE (procedural level) and apply them appropriately.
10. Develop logical thinking for database-driven decision systems used in real-world applications.