CU CHANDIGARH UNIVERSITY

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Worksheet 3

Student Name: Aryan **UID: 25MCI10082**
Branch: MCA (AI&ML) **Section/Group: 1/A**
Semester: 2nd **Date of Performance: 22/01/2026**
Subject Name: DBMS LAB **Subject Code:**

1. Aim of the Session

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

2. Tools Used

- PostgreSQL

3. Objective of the Session

- 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

4. Practical / Experiment Steps

- Design the database schema for implementing conditional logic.
- Create tables using appropriate constraints.
- Insert sample records into tables.
- Perform data classification using CASE expressions.
- Perform conditional data updates using CASE logic.
- Implement IF–ELSE logic using PL/pgSQL blocks.
- Perform custom sorting using CASE expressions.

- Display and verify results.

5. Procedure of the Practical

(i) Start the system and log in to the computer.

(ii) Open PostgreSQL software.

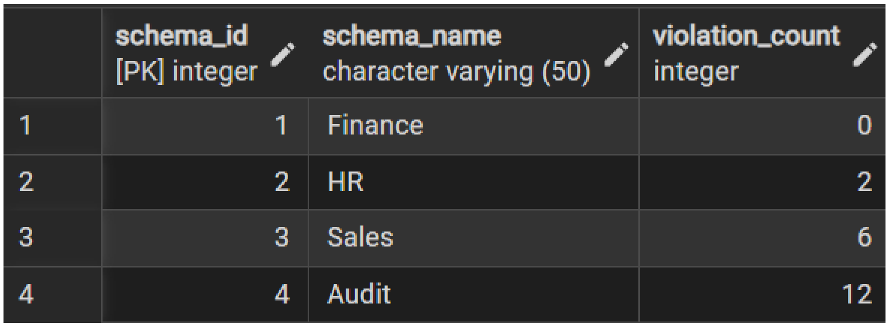
(iii) create database Experiment3;

(iv) Create a table to store schema-level violations.

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

(v) Insert records using DML commands.

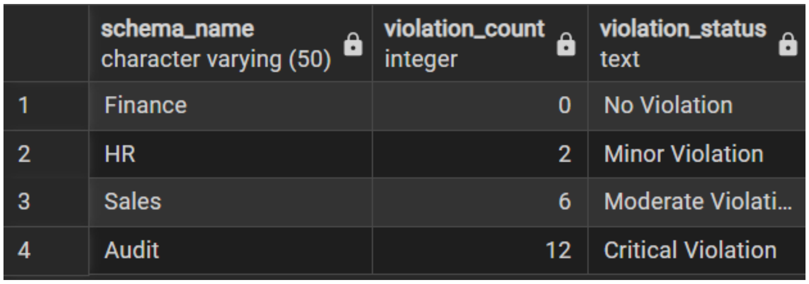
```
INSERT INTO schema_violations VALUES
(1, 'Finance', 0),
(2, 'HR', 2),
(3, 'Sales', 6),
(4, 'Audit', 12);
```



	schema_id [PK] integer	schema_name character varying (50)	violation_count integer
1	1	Finance	0
2	2	HR	2
3	3	Sales	6
4	4	Audit	12

(vi) 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;
```

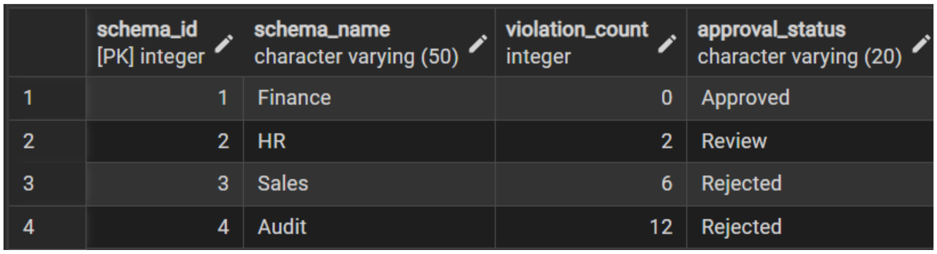


	schema_name character varying (50)	violation_count integer	violation_status text
1	Finance	0	No Violation
2	HR	2	Minor Violation
3	Sales	6	Moderate Violati...
4	Audit	12	Critical Violation

(vii) Applying CASE Logic in Data Updates

```
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 <= 5 THEN 'Needs Review'
```

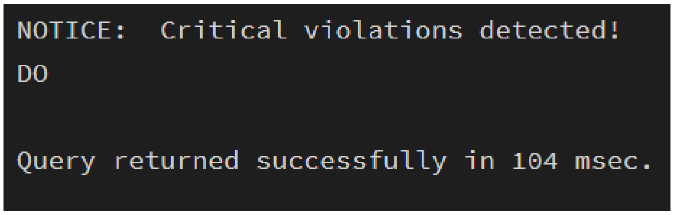
```
ELSE 'Rejected'
   END;
```



	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	Review
3	3	Sales	6	Rejected
4	4	Audit	12	Rejected

(viii) Implementing IF–ELSE Logic Using PL/pgSQL.

```
DO $$
DECLARE
   v_count INT := 6;
BEGIN
   IF v_count = 0 THEN
      RAISE NOTICE 'No violations detected.';
   ELSIF v_count <= 5 THEN
      RAISE NOTICE 'Minor issues found. Review required.';
   ELSE
      RAISE NOTICE 'Critical violations detected!';
   END IF;
END $$;
```

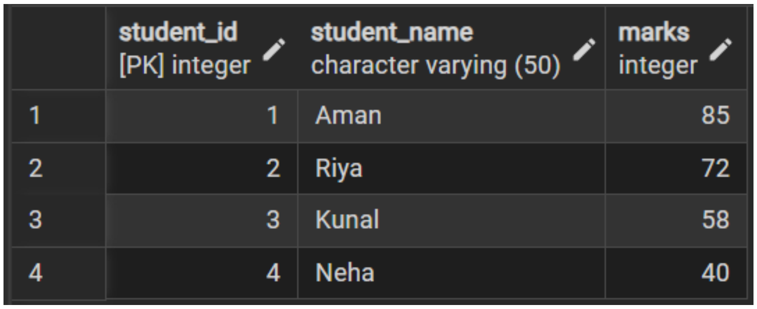


```
NOTICE:  Critical violations detected!
DO

Query returned successfully in 104 msec.
```

(ix) Real-World Classification Scenario (Grading System).

```
CREATE TABLE students (
    student_id INT PRIMARY KEY,
    student_name VARCHAR(50),
    marks INT
);
INSERT INTO students VALUES
(1, 'Aman', 85),
(2, 'Riya', 72),
(3, 'Kunal', 58),
(4, 'Neha', 40);
```

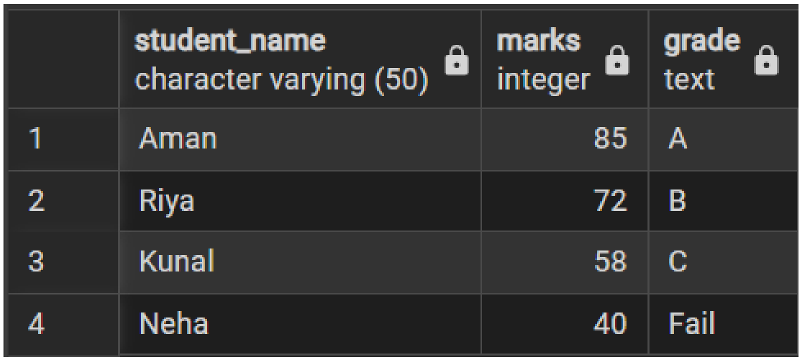


	student_id [PK] integer	student_name character varying (50)	marks integer
1	1	Aman	85
2	2	Riya	72
3	3	Kunal	58
4	4	Neha	40

Grade Classification.

```
SELECT
    student_name,
    marks,
    CASE
        WHEN marks >= 80 THEN 'A'
        WHEN marks >= 60 THEN 'B'
        WHEN marks >= 50 THEN 'C'
        ELSE 'Fail'
```

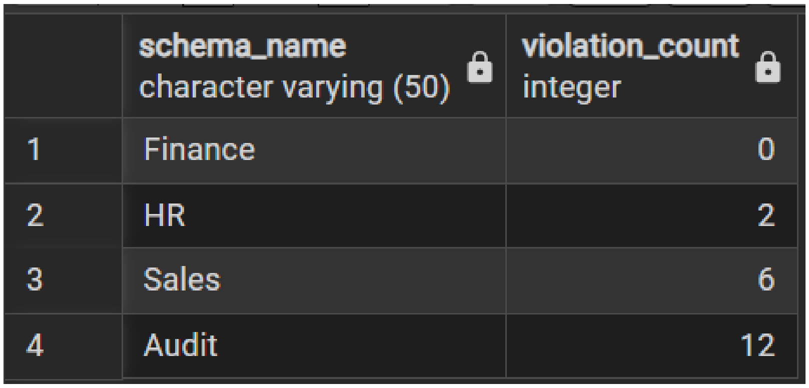
```
 END AS grade
FROM students;
```



	student_name character varying (50)	marks integer	grade text
1	Aman	85	A
2	Riya	72	B
3	Kunal	58	C
4	Neha	40	Fail

(xi) 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 <= 3 THEN 2
      WHEN violation_count <= 7 THEN 3
      ELSE 4
   END;
```



	schema_name character varying (50)	violation_count integer
1	Finance	0
2	HR	2
3	Sales	6
4	Audit	12

Learning Outcomes

- How data can be filtered to retrieve only relevant records from a database.
- Learn how sorting improves readability and usefulness of query results in reports.
- Gain the ability to group data for analytical purposes.
- Differentiate between row-level conditions and group-level conditions.
- Develop confidence in writing analytical SQL queries used in real-world scenarios.
- Better prepared to answer SQL-based placement and interview questions related to filtering, grouping, and aggregation.