Worksheet-3

Name: TANUJ JOSHI
Branch: MCA General
Semester: II
Subject Name: Technical Training

UID: 25MCA20081
Section/Group: 25MCA_KAR-1
Date of Performance: 27-01-26
Subject Code: 25CAP-652

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.

Tools Used:

PostgreSQL — Powerful open-source relational database for storing and managing data efficiently

Objective of the Session:

The objectives of this practical session are:

- 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

Practical / Experiment Steps:

Classifying Data Using CASE Expression

Steps

1. Write a SELECT query to retrieve schema names and violation counts.
2. Use a searched CASE expression in the SELECT clause.
3. Define conditions for:
   a. No Violation
   b. Minor Violation
   c. Moderate Violation

   d. Critical Violation
4. Assign appropriate labels for each condition.
5. Execute the query and observe the classification result.

Applying CASE Logic in Data Updates

Steps

1. Alter the existing table to add a new column named approval_status.
2. Write an UPDATE statement using a CASE expression.
3. Define approval rules such as:
   a. Approved
   b. Needs Review
   c. Rejected
4. Update all rows based on violation count.
5. Verify the update using a SELECT query.

Implementing IF–ELSE Logic Using PL/pgSQL

Steps

1. Begin a PL/pgSQL DO block.
2. Declare a variable to store violation count.
3. Assign a value to the variable.
4. Use IF–ELSE IF–ELSE conditions to check violation levels.
5. Display appropriate messages using RAISE NOTICE.
6. Execute the block and observe the output.

Real-World Classification Scenario (Grading System)

Steps

1. Create a table to store student names and marks.
2. Insert sample student records with varying marks.
3. Write a SELECT query using a CASE expression.
4. Define grade categories (such as A, B, C, Fail).
5. Execute the query and verify grade classification.

Using CASE for Custom Sorting

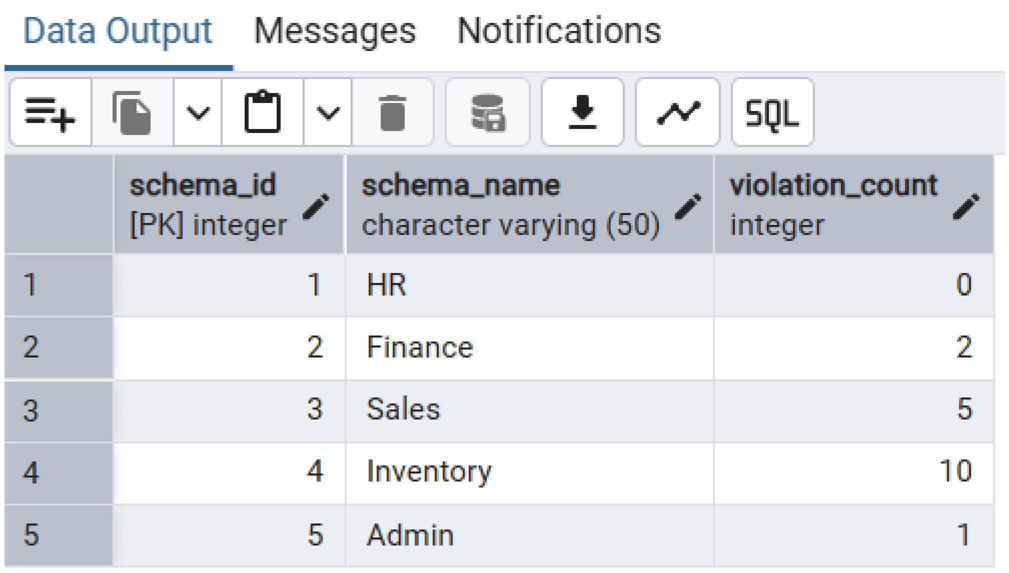
Steps

1. Write a SELECT query to retrieve schema details.
2. Use a CASE expression inside the ORDER BY clause.
3. Assign priority values to violation severity levels.
4. Execute the query to sort records based on severity.
5. Analyze the ordered output.

Procedure of the Practical:

1. Start the system and log in to the computer.
2. Open the PostgreSQL client tool (psql / pgAdmin).
3. Create a table to store schema violations details.

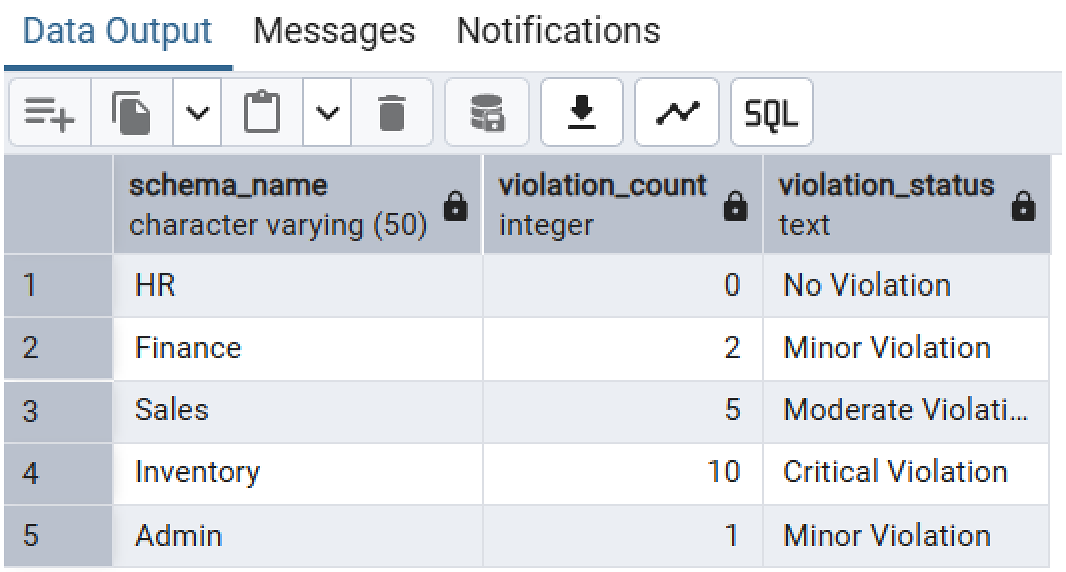
```
CREATE TABLE schema_violations (
    schema_id SERIAL PRIMARY KEY,
    schema_name VARCHAR(50),
    violation_count INT
);

INSERT INTO schema_violations (schema_name, violation_count) VALUES
('HR', 0),
('Finance', 2),
('Sales', 5),
('Inventory', 10),
('Admin', 1);

SELECT * FROM schema_violations;
```


Data Output Messages Notifications

	schema_id [PK] integer	schema_name character varying (50)	violation_count integer
1	1	HR	0
2	2	Finance	2
3	3	Sales	5
4	4	Inventory	10
5	5	Admin	1

4. Classifying Data Using CASE Expression.

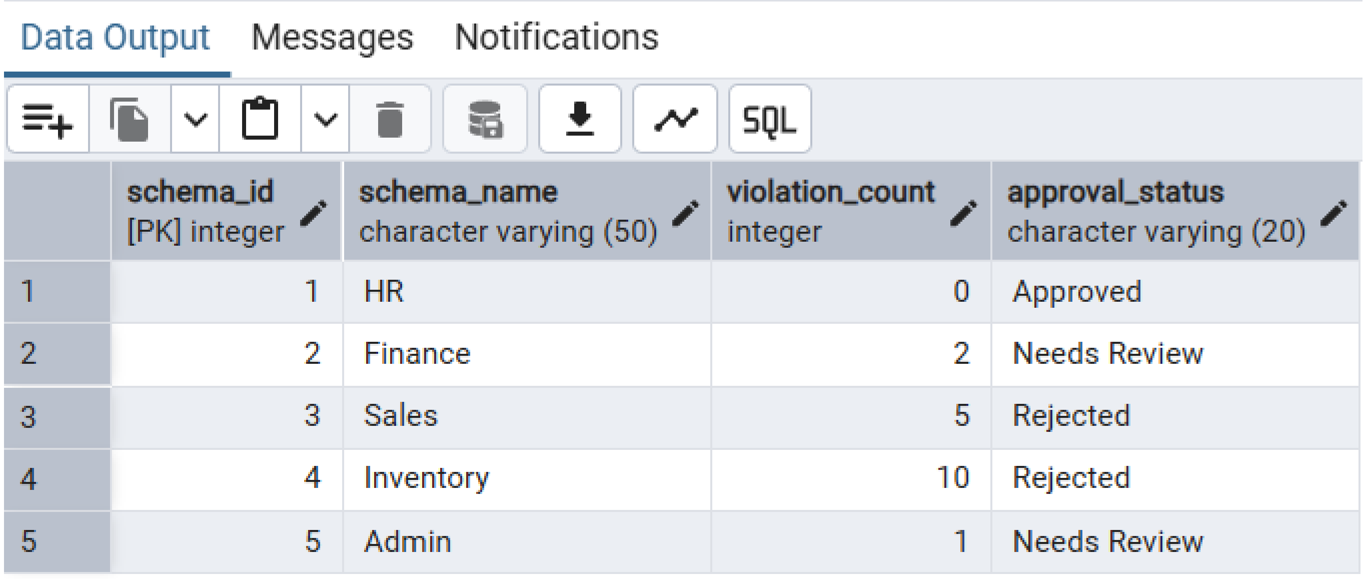
```
SELECT
   schema_name,
   violation_count,
   CASE
      WHEN violation_count = 0 THEN 'No Violation'
      WHEN violation_count BETWEEN 1 AND 3 THEN 'Minor Violation'
      WHEN violation_count BETWEEN 4 AND 7 THEN 'Moderate Violation'
      ELSE 'Critical Violation'
   END AS violation_status
FROM schema_violations;
```



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

5. Applying CASE Logic in Data Updates.

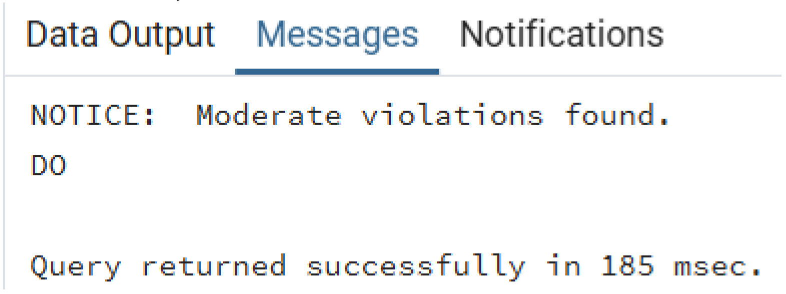
```
ALTER TABLE schema_violations
ADD COLUMN approval_status VARCHAR(20);

UPDATE schema_violations
SET approval_status =
   CASE
      WHEN violation_count = 0 THEN 'Approved'
      WHEN violation_count BETWEEN 1 AND 4 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	HR	0	Approved
2	2	Finance	2	Needs Review
3	3	Sales	5	Rejected
4	4	Inventory	10	Rejected
5	5	Admin	1	Needs Review

6. Implementing IF–ELSE Logic Using PL/pgSQL.

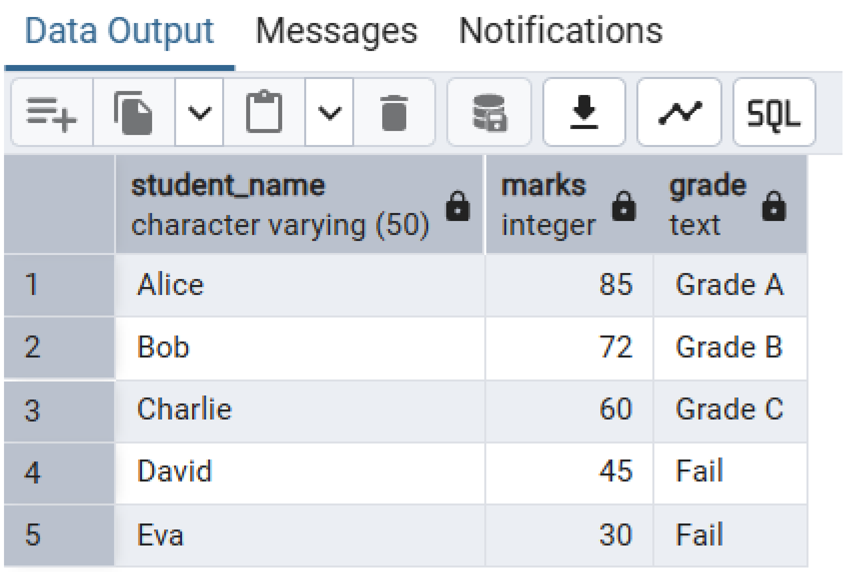
```
DO $$
DECLARE
   v_violation_count INT := 5;
BEGIN
   IF v_violation_count = 0 THEN
      RAISE NOTICE 'No violations detected.';
   ELSIF v_violation_count BETWEEN 1 AND 3 THEN
      RAISE NOTICE 'Minor violations found.';
   ELSIF v_violation_count BETWEEN 4 AND 7 THEN
      RAISE NOTICE 'Moderate violations found.';
   ELSE
      RAISE NOTICE 'Critical violations detected.';
   END IF;
END $$;
```



```
NOTICE:  Moderate violations found.
DO


Query returned successfully in 185 msec.
```

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

```
CREATE TABLE students (
   student_id SERIAL PRIMARY KEY,
    student_name VARCHAR(50),
    marks INT
);
INSERT INTO students (student_name, marks) VALUES
('Alice', 85),
('Bob', 72),
('Charlie', 60),
('David', 45),
('Eva', 30);
SELECT
    student_name,
    marks,
    CASE
        WHEN marks >= 80 THEN 'Grade A'
        WHEN marks >= 65 THEN 'Grade B'
        WHEN marks >= 50 THEN 'Grade C'
        ELSE 'Fail'
    END AS grade
FROM students;
```

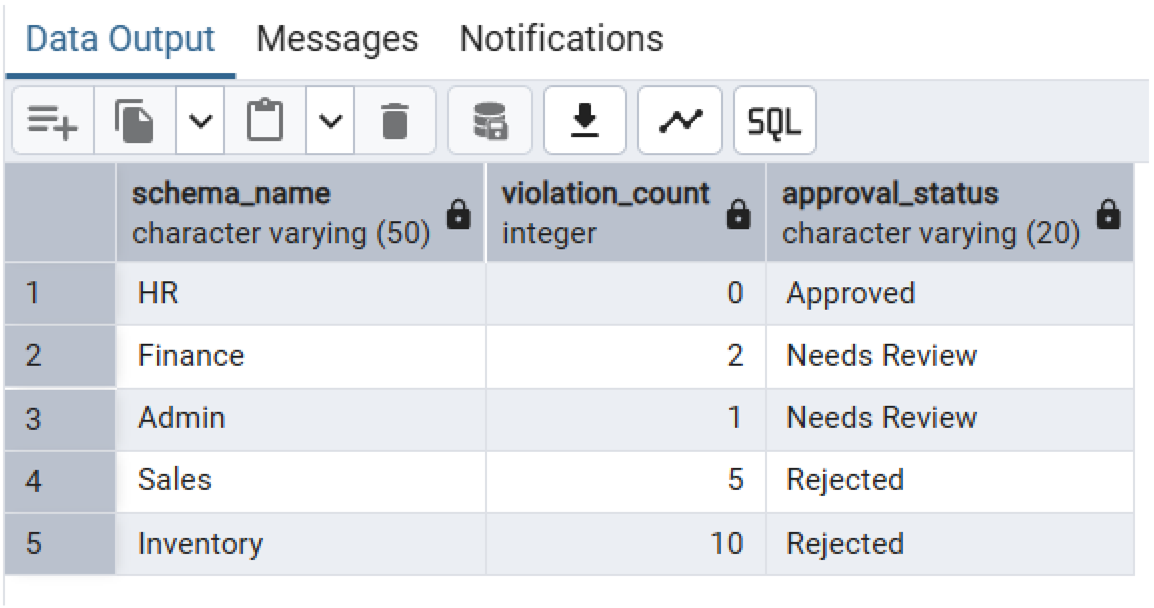


	student_name character varying (50)	marks integer	grade text
1	Alice	85	Grade A
2	Bob	72	Grade B
3	Charlie	60	Grade C
4	David	45	Fail
5	Eva	30	Fail

8. Using CASE for Custom Sorting.

```
SELECT
    schema_name,
    violation_count,
    approval_status
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;
```



	schema_name character varying (50)	violation_count integer	approval_status character varying (20)
1	HR	0	Approved
2	Finance	2	Needs Review
3	Admin	1	Needs Review
4	Sales	5	Rejected
5	Inventory	10	Rejected

9. Verify the output after execution.
10. Note down the results obtained.
11. Save the work and take screenshots for record.

I/O Analysis:

Input Provided

- SQL queries using CASE expressions in SELECT, UPDATE, and ORDER BY clauses
- PL/pgSQL DO block implementing IF–ELSE conditional logic
- Table data containing:
- Schema names
- Violation counts
- Sample student records with marks inserted into the grading table

Output Generated

- Classified schema records based on violation severity
- Automatically assigned approval status for each schema
- Conditional messages displayed using IF–ELSE logic

- Student grades generated based on marks
- Custom-sorted output prioritizing records by violation severity

Learning Outcomes:

- Apply CASE expressions for data classification and updates
- Implement IF–ELSE logic using PL/pgSQL
- Perform rule-based data processing within the database
- Solve real-world SQL interview and backend scenarios