

Worksheet -3

Student Name: SANCHIT KATOCH

Branch: MCA

Semester: 2nd

Subject Name: TECHNICAL TRAINING I

UID: 25MCA20059

Section/Group: 25MCA-1-A

Date of Performance: 27/01/2026

Subject Code: 25CAP-652

1. Aim/Overview of the practical:

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:

- 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 back-end systems.

3. S/W Requirement:

- Oracle Database Express Edition
- pgAdmin.

Procedure:

Start PostgreSQL Environment

Prerequisite Understanding

First create a table that stores:

- A unique identifier
- A schema or entity name
- A numeric count representing violations or issues

Populate the table with multiple records having different violation counts.

Step 1: Classifying Data Using CASE Expression

- 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

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

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.

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.

Step 5: Using CASE for Custom Sorting

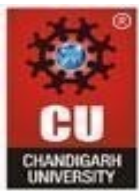
- Retrieve schema details.
- Apply conditional priority while sorting records based on violation severity.

4.Code:

```
CREATE TABLE schema_violations (  
    schema_id INT PRIMARY KEY,  
    schema_name VARCHAR(30),  
    violation_count INT  
);
```

```
INSERT INTO schema_violations VALUES  
(1,'Sanchit', 0),  
(2,'Mandeep', 2),  
(3,'Digam', 4),  
(4,'Vanshaj', 7),  
(5,'Divanshu', 8);
```

```
-- step1  
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;  
  
-- step2  
  
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;  
  
SELECT * FROM schema_violations;  
  
--step 3
```



```
DO $$  
DECLARE  
    v_count INT := 6;  
BEGIN  
    IF v_count = 0 THEN  
        RAISE NOTICE 'No violations detected';  
    ELSIF v_count <= 5 THEN  
        RAISE NOTICE 'Minor violations review required';  
    ELSE  
        RAISE NOTICE 'Critical violations access denied';  
    END IF;  
END $$;
```

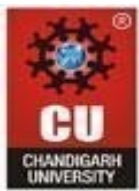
```
--step 4  
CREATE TABLE students (  
    student_id INT PRIMARY KEY,  
    student_name VARCHAR(30),  
    marks INT  
);
```

```
INSERT INTO students VALUES  
(1,'Sanchit', 88),  
(2,'Mandeep', 80),  
(3,'Digam', 72),  
(4,'Vanshaj',45),  
(5,'Divanshu', 30);
```

```
SELECT student_name,marks,  
CASE  
    WHEN marks >= 80 THEN 'A'  
    WHEN marks >= 70 THEN 'B'  
    WHEN marks >= 60 THEN 'C'  
    WHEN marks >= 40 THEN 'D'  
    ELSE 'F'  
END AS grade  
FROM students;
```

--step 5

```
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;
```

5.Output:

Prerequisite table creation

pgAdmin 4

File Object Tools Edit View Window Help

Experiment_3/postgres@PostgreSQL 18*

Experiment_3/postgres@PostgreSQL 18

No limit

Query Query History

```
1 CREATE TABLE schema_violations (  
2     schema_id INT PRIMARY KEY,  
3     schema_name VARCHAR(30),  
4     violation_count INT  
5 );  
6  
7 INSERT INTO schema_violations VALUES  
8 (1,'Sanchit', 0),  
9 (2,'Mandeep', 2),  
10 (3,'Digam', 4),  
11 (4,'Vanshaj', 7),  
12 (5,'Divanshu', 8);  
13  
14 -- step1  
15 SELECT  
16     schema_name,  
17     violation_count,  
18     CASE
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 125 msec.

Total rows: Query complete 00:00:00.125 CRLF Ln 5, Col 3

Step1

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- Servers (1)
 - PostgreSQL 18
 - Databases (7)
 - DATB1
 - EXP1.2
 - EXPERIMENT_1
 - Experiment_2
 - Experiment_3
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas
 - Subscriptions
 - college_db
 - postgres
 - Login/Group Roles (19)
 - Tablespaces

Experiment_3/postgres@PostgreSQL 18*

Experiment_3/postgres@PostgreSQL 18

Query

```

13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
-- step1
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;

-- step2
ALTER TABLE schema_violations
ADD COLUMN approval_status VARCHAR(20);

UPDATE schema_violations

```

Data Output

Messages

Notifications

INSERT 0 5

Query returned successfully in 99 msec.

Total rows: Query complete 00:00:00.099 CRLF Ln 22, Col 24

Step2 add – update - alter column

pgAdmin 4

File Object Tools Edit View Window Help

Experiment_3/postgres@PostgreSQL 18*

Experiment_3/postgres@PostgreSQL 18

Query

```

20
21
22
23
24
25
26
27
28
29
30
31
-- step2
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'

```

Data Output

Messages

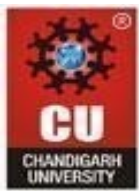
Notifications

UPDATE 5

Query returned successfully in 115 msec.

Query returned successfully in 115 msec. X

Total rows: Query complete 00:00:00.115 CRLF Ln 31, Col 135



Step3

pgAdmin 4

File Object Tools Edit View Window Help

Experiment_3/postgres@PostgreSQL 18*

Experiment_3/postgres@PostgreSQL 18

Query Query History

Execute script (F5)

```
DO $$
DECLARE
  v_count INT := 6;
BEGIN
  IF v_count = 0 THEN
    RAISE NOTICE 'No violations detected';
  ELSIF v_count <= 5 THEN
    RAISE NOTICE 'Minor violations review required';
  ELSE
    RAISE NOTICE 'Critical violations access denied';
  END IF;
END $$;
```

Data Output Messages Notifications

NOTICE: Critical violations access denied
DO

Query returned successfully in 157 msec.

Total rows: Query complete 00:00:00.157

✓ Query returned successfully in 157 msec. ✕

CRLF Ln 49, Col 1

Step4



pgAdmin 4

File Object Tools Edit View Window Help

pgAdmin 4 interface showing the Query Editor for PostgreSQL 18. The left sidebar shows the database structure, including the 'Experiment_3' database. The main pane displays the following SQL script:

```
--step 4
CREATE TABLE students (
  student_id INT PRIMARY KEY,
  student_name VARCHAR(30),
  marks INT
);
INSERT INTO students VALUES
(1,'Sanchit', 88),
(2,'Mandeep', 80),
(3,'Digam', 72),
(4,'Vanshaj',45),
(5,'Divanshu', 30);
```

The 'Data Output' tab shows the result of the INSERT statement:

```
INSERT 0 5
Query returned successfully in 100 msec.
```

A green notification bar at the bottom right states: "Query returned successfully in 100 msec." The status bar at the bottom indicates "Total rows: Query complete 00:00:00.100" and "Ln 64, Col 1".

pgAdmin 4

File Object Tools Edit View Window Help

pgAdmin 4 interface showing the Query Editor for PostgreSQL 18. The left sidebar shows the database structure, including the 'Experiment_3' database. The main pane displays the following SQL script:

```
(5,'Divanshu', 30);

SELECT student_name,marks,
CASE
  WHEN marks >= 80 THEN 'A'
  WHEN marks >= 70 THEN 'B'
  WHEN marks >= 60 THEN 'C'
  WHEN marks >= 40 THEN 'D'
  ELSE 'F'
END AS grade
FROM students;
```

The 'Data Output' tab shows the result of the SELECT statement:

student_name	marks	grade
Sanchit	88	A
Mandeep	80	A
Digam	72	B
Vanshaj	45	D
Divanshu	30	F

A green notification bar at the bottom right states: "Successfully run. Total query runtime: 146 msec. 5 rows affected." The status bar at the bottom indicates "Total rows: 5" and "Query complete 00:00:00.146".

Step5

pgAdmin 4

File Object Tools Edit View Window Help

Experiment_3/postgres@PostgreSQL 18+ x

Experiment_3/postgres@PostgreSQL 18

No limit

Execute script (F5)

```

73 FROM students;
74
75 --step 5
76
77 SELECT schema_name,violation_count FROM schema_violations
78 ORDER BY CASE
79     WHEN violation_count = 0 THEN 1
80     WHEN violation_count <= 3 THEN 2
81     WHEN violation_count <= 7 THEN 3
82     ELSE 4
83 END;
84

```

Data Output Messages Notifications

Showing rows: 1 to 5 Page No: 1 of 1

	schema_name character varying (30)	violation_count integer
1	Sanchit	0
2	Mandeep	2
3	Digam	4
4	Vanshaj	7
5	Divanshu	8

Total rows: 5 Query complete 00:00:00.160

Successfully run. Total query runtime: 160 msec. 5 rows affected.

CRLF Ln 83, Col 7

6.Learning Outcomes:

- * Understood how conditional logic is implemented in PostgreSQL and using CASE expressions and IF-ELSE constructs.
- * Learnt how rule-based SQL logic helps in data classification and validation.
- * Gained the ability to apply conditional statements
- * Clearly able to use CASE-based logic for analytics and compliance reporting scenarios.