

Branch: MCA (Data Science)	Semester: 2 nd
Student Name: Chaitanya Sharma	UID: 25MCD10056
Subject Name: Technical Training - I Lab	Subject Code: 25CAP-652
Section/Group: 25MCD-1(A)	Date of Performance: 27-Jan-2026

Experiment No.: 3

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

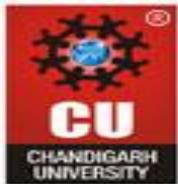
2. S/W Requirement: PostgreSQL, PG Admin

3. Objectives:

- 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. Task to be done:

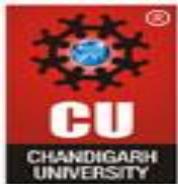
- Step 1: Classifying Data Using CASE Expression
 - Retrieve schema names and their violation counts.
 - Use conditional logic to classify each schema into categories such as
 - No Violation
 - Minor Violation
 - Moderate Violation
 - Critical Violation



- 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.

○ **Table:**

	record_id [PK] integer	entity_name character varying (50)	violation_count integer
1	1	Auth_Service	0
2	2	Payment_Service	1
3	3	Order_Service	2
4	4	Audit_Service	3
5	5	Admin_Service	5



o **Code:**

```
CREATE TABLE violation_review (
    record_id SERIAL PRIMARY KEY,
    entity_name VARCHAR(50) NOT NULL,
    violation_count INT NOT NULL CHECK (violation_count >= 0)
);
INSERT INTO violation_review (entity_name, violation_count) VALUES
('Auth_Service', 0),
('Payment_Service', 1),
('Order_Service', 2),
('Audit_Service', 3),
('Admin_Service', 5);
```

```
SELECT * FROM violation_review;
```

```
-- case statement to classify violations:
```

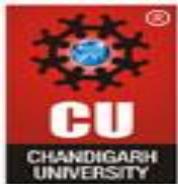
```
SELECT *,
CASE WHEN violation_count = 0 THEN 'No Violations'
WHEN violation_count BETWEEN 1 and 2 THEN 'Moderate Violations'
ELSE 'Critical Violations'
END AS violations_level
FROM violation_review;
```

```
-- Adding a status column:
```

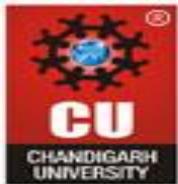
```
ALTER TABLE violation_review
ADD COLUMN status VARCHAR(20);
```

```
-- case statement to update status column:
```

```
UPDATE violation_review
SET status =
CASE WHEN violation_count = 0 THEN 'Accepted'
WHEN violation_count BETWEEN 1 AND 2 THEN 'Reviewing'
ELSE 'Rejected'
END
WHERE status IS NULL;
```



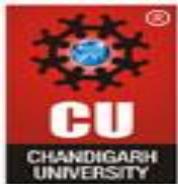
```
-- If Else:  
DO $$  
DECLARE  
    v_count INT;  
BEGIN  
    SELECT violation_count  
    INTO v_count  
    FROM violation_review  
    WHERE entity_name = 'Payment_Service';  
  
    IF v_count = 0 THEN  
        RAISE NOTICE 'Payment_Service: Accepted';  
  
    ELSIF v_count = 1 THEN  
        RAISE NOTICE 'Payment_Service: Needs Review';  
  
    ELSE  
        RAISE NOTICE 'Payment_Service: Rejected';  
    END IF;  
END $$;  
  
-- step 4  
CREATE TABLE student (  
    student_name VARCHAR(50),  
    marks INT  
);  
  
INSERT INTO student VALUES  
('Chai', 92),  
('Coffee', 78),  
('Tea', 65),  
('Latte', 48),  
('Chai-Latte', 33);
```



```
SELECT
student_name,
marks,
CASE
WHEN marks >= 90 THEN 'A'
WHEN marks >= 75 THEN 'B'
WHEN marks >= 60 THEN 'C'
WHEN marks >= 40 THEN 'D'
ELSE 'Fail'
END AS grade
FROM student;
```

-- custom order filtering:

```
SELECT
entity_name,
violation_count
FROM violation_review
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,
violation_count DESC;
```



	record_id [PK] integer	entity_name character varying (50)	violation_count integer	violations_level text
1	1	Auth_Service	0	No Violations
2	2	Payment_Service	1	Moderate Violations
3	3	Order_Service	2	Moderate Violations
4	4	Audit_Service	3	Critical Violations
5	5	Admin_Service	5	Critical Violations

	record_id [PK] integer	entity_name character varying (50)	violation_count integer	status character varying (20)
1	1	Auth_Service	0	[null]
2	2	Payment_Service	1	[null]
3	3	Order_Service	2	[null]
4	4	Audit_Service	3	[null]
5	5	Admin_Service	5	[null]

	record_id [PK] integer	entity_name character varying (50)	violation_count integer	status character varying (20)
1	1	Auth_Service	0	Accepted
2	2	Payment_Service	1	Reviewing
3	3	Order_Service	2	Reviewing
4	4	Audit_Service	3	Rejected
5	5	Admin_Service	5	Rejected

NOTICE: Payment_Service: Needs Review
DO

	student_name character varying (50) 	marks integer 	grade text 
1	Chai	92	A
2	Coffee	78	B
3	Tea	65	C
4	Latte	48	D
5	Chai-Latte	33	Fail

	entity_name character varying (50) 	violation_count integer 
1	Auth_Service	0
2	Audit_Service	3
3	Order_Service	2
4	Payment_Service	1
5	Admin_Service	5

- **Learning Outcomes:**

- Understand and implement conditional decision-making logic in PostgreSQL using CASE expressions and IF-ELSE constructs.
- Apply rule-based classification and validation logic directly at the database level using SELECT, UPDATE, and DO blocks.
- Demonstrate backend procedural control by using PL/pgSQL to evaluate conditions and automate status assignment based on violation severity.