



Experiment 4 — Conditional Control Statements in PL/pgSQL (PostgreSQL)



Aim

To design and implement PL/pgSQL programs using conditional control statements such as IF-ELSE, IF-ELSIF-ELSE, ELSIF ladder, and CASE constructs to control the flow of execution based on logical conditions and analyze decision-making capabilities in PL/pgSQL blocks.



Objectives

- Understand decision-making constructs in PL/pgSQL
- Implement IF-ELSE statements
- Implement IF-ELSIF-ELSE and ELSIF ladder
- Apply CASE statements for multi-way branching
- Analyze program flow based on conditional evaluation
- Gain hands-on experience with PostgreSQL procedural blocks



Software Requirements

- PostgreSQL
- pgAdmin / psql

► Practical Steps

1. Open pgAdmin
2. Connect to PostgreSQL database
3. Open Query Tool
4. Write PL/pgSQL block using:

```
DO $$  
DECLARE  
    variable declarations  
BEGIN  
    executable statements  
END $$;
```

5. Use RAISE NOTICE to display output
6. Execute and observe results in Messages tab

Theory

Conditional control statements allow programs to make decisions and execute different blocks of code depending on logical conditions.

- ◇ IF-ELSE

Used when two outcomes exist.

- ◇ IF-ELSIF-ELSE

Used when multiple mutually exclusive conditions exist.

- ◇ ELSIF Ladder

Checks conditions sequentially and stops at the first true condition.

- ◇ CASE

Cleaner alternative for multi-way branching.

Programs (PostgreSQL)

Program 1 — IF-ELSE

Check whether a number is positive or non-positive

```
DO $$  
DECLARE  
    num INTEGER := 5;  
BEGIN  
    IF num > 0 THEN  
        RAISE NOTICE 'The number that is Positive is %', num;  
    ELSE  
        RAISE NOTICE 'The number that is Non-Positive is %', num;  
    END IF;  
END $$;
```

☒ Program 2 — IF-ELSIF-ELSE

Evaluate grade based on marks

DO \$\$

DECLARE

marks INTEGER := 82;

BEGIN

IF marks >= 90 THEN

RAISE NOTICE 'Grade : A';

ELSIF marks >= 75 THEN

RAISE NOTICE 'Grade : B';

ELSIF marks >= 60 THEN

RAISE NOTICE 'Grade : C';

ELSE

RAISE NOTICE 'Grade : D';

END IF;

END \$\$;

☒ Program 3 — ELSIF Ladder

Determine performance status

DO \$\$

DECLARE

marks INTEGER := 68;

BEGIN

IF marks >= 85 THEN

RAISE NOTICE 'Performance : Excellent';

ELSIF marks >= 70 THEN

RAISE NOTICE 'Performance : Very Good';

ELSIF marks >= 55 THEN

RAISE NOTICE 'Performance : Good';

ELSIF marks >= 40 THEN

RAISE NOTICE 'Performance : Average';

ELSE

RAISE NOTICE 'Performance : Fail';

END IF;

END \$\$;

☒ Program 4 — CASE Statement

Display day name based on day number

DO \$\$

DECLARE

day_no INTEGER := 3;

BEGIN

```

CASE day_no
  WHEN 1 THEN RAISE NOTICE 'Monday';
  WHEN 2 THEN RAISE NOTICE 'Tuesday';
  WHEN 3 THEN RAISE NOTICE 'Wednesday';
  WHEN 4 THEN RAISE NOTICE 'Thursday';
  WHEN 5 THEN RAISE NOTICE 'Friday';
  WHEN 6 THEN RAISE NOTICE 'Saturday';
  WHEN 7 THEN RAISE NOTICE 'Sunday';
  ELSE RAISE NOTICE 'Invalid Day Number';
END CASE;
END $$;

```

Input / Output

Program	Input	Output
1	num = 5	The number 5 is Positive
2	marks = 82	Grade: B
3	marks = 68	Performance: Good
4	day_no = 3	Wednesday

Output Screenshot:

```

psql:commands.sql:10: NOTICE:  The number is positive.
psql:commands.sql:27: NOTICE:  Grade: B
psql:commands.sql:44: NOTICE:  Performance: Very Good

```

Analysis

- IF-ELSE provides binary decision control
- IF-ELSIF-ELSE handles multiple conditions
- ELSIF ladder improves efficiency
- CASE improves readability
- PL/pgSQL closely follows structured programming concepts

Learning Outcomes

After completing this experiment, you will be able to:

- Write PL/pgSQL blocks
- Declare variables
- Use conditional statements
- Display output using RAISE NOTICE
- Develop decision-based database programs confidently

Conclusion

This experiment demonstrates how conditional constructs help implement logical decision-making and control program flow effectively in PostgreSQL using PL/pgSQL.