

# **Experiment 4: PL/SQL Conditional Control Statements**

**Name: Arnav Prajapati**

**UID: 24BAI70131**

**Course: BE-CSE (AI&ML)**

**Subject: Database Management System**

## **1. Aim of the Session**

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

## **2. Software Requirements**

- Database: PostgreSQL
- Database Administration Tool: pgAdmin

## **3. Objective of the Session**

To implement control structures in PL/SQL (IF-ELSE, ELSE-IF, ELSE-IF LADDER, CASE STATEMENTS in PL-SQL BLOCK) to handle procedural logic within the database environment.

## **4. Practical / Experiment Steps**

The work was carried out through the following activities:

1. Program Structure Definition: Designed basic PL/SQL blocks (anonymous blocks using DO statements) consisting of declaration and execution sections.
2. Variable Declaration: Declared required variables (Integer, Varchar) in the DECLARE section to store input values for testing conditions.
3. Logic Implementation: Wrote executable statements using IF-THEN-ELSE, ELSIF, and CASE constructs to perform logical operations.
4. Output Display: Used the RAISE NOTICE procedure to display results and formatted messages to the console.
5. Execution and Verification: Executed the code in the pgAdmin Query Tool and verified the outputs against expected logical results.

## **5. Procedure of the Practical**

- 1. Environment Initialization:** Opened pgAdmin and connected to the PostgreSQL server.
- 2. Session Configuration:** Ensured the "Messages" tab was visible to view RAISE NOTICE outputs.
- 3. Program Preparation:** Formatted the block using DO \$\$... END\$\$; syntax.
- 4. Variable Setup:** Assigned test values to variables (e.g., marks, numbers, or day IDs) in the declaration section.
- 5. Logic Execution:** Implemented conditional branches to evaluate the declared variables.
- 6. Output Handling:** Added notification triggers for each logic branch.
- 7. Result Verification:** Ran the scripts and documented the performance of each conditional construct.

## **6. I/O Analysis (Input / Output Analysis)**

### **Query 1: IF-ELSE (Check Positive/Non-Positive)**

## *DECLARE*

*num* INTEGER := -5;

*BEGIN*

*IF num > 0 THEN*

```
DBMS_OUTPUT.PUT_LINE('The number ' || num || ' is positive.');
```

*ELSE*

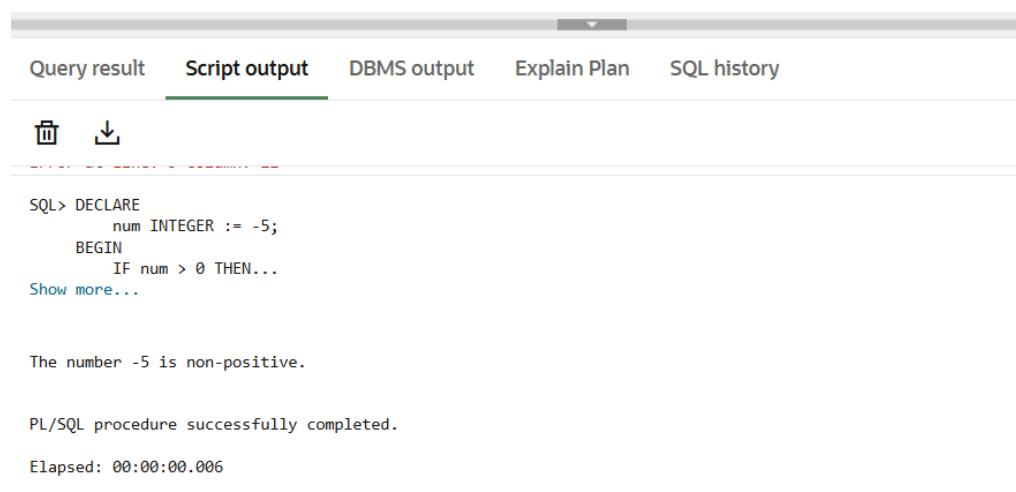
```
DBMS_OUTPUT.PUT_LINE('The number ' || num || ' is non-positive.');
```

*END IF;*

END;

```
[SQL Worksheet] ✓ => ⌂ L ↴ ⌂ Md ⌂ W

1  DECLARE
2  num INTEGER := -5;
3  BEGIN
4  IF num >> 0 THEN
5  DBMS_OUTPUT.PUT_LINE('The number ' || num || ' is positive.');
6  ELSE
7  DBMS_OUTPUT.PUT_LINE('The number ' || num || ' is non-positive.');
8  END IF;
9
END;
```



## **Query 2 & 3: ELSIF Ladder (Student Grading & Performance)**

```
DECLARE
    marks INTEGER := 85;
BEGIN
    IF marks >= 90 THEN
        DBMS_OUTPUT.PUT_LINE('Grade: A+ | Status: Excellent');
    ELSIF marks >= 80 THEN
        DBMS_OUTPUT.PUT_LINE('Grade: A | Status: Very Good');
    ELSIF marks >= 60 THEN
        DBMS_OUTPUT.PUT_LINE('Grade: B | Status: Satisfactory');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Grade: F | Status: Needs Improvement');
    END IF;
END;
```

```
14
15  DECLARE
16      marks INTEGER := 85;
17  BEGIN
18      IF marks >= 90 THEN
19          DBMS_OUTPUT.PUT_LINE('Grade: A+ | Status: Excellent');
20      ELSIF marks >= 80 THEN
21          DBMS_OUTPUT.PUT_LINE('Grade: A | Status: Very Good');
22      ELSIF marks >= 60 THEN
23          DBMS_OUTPUT.PUT_LINE('Grade: B | Status: Satisfactory');
24      ELSE
25          DBMS_OUTPUT.PUT_LINE('Grade: F | Status: Needs Improvement');
26      END IF;
27  END;
28
29
```

Query result    Script output    DBMS output    Explain Plan    SQL history

SQL> DECLARE  
 marks INTEGER := 85;  
 BEGIN  
 IF marks >= 90 THEN...  
Show more...

Grade: A | Status: Very Good

PL/SQL procedure successfully completed.

***Query 4: CASE Statement (Day of the Week)***

DECLARE

    day\_num INTEGER := 3;

    day\_name VARCHAR2(20);

BEGIN

    day\_name :=

        CASE day\_num

            WHEN 1 THEN 'Monday'

            WHEN 2 THEN 'Tuesday'

            WHEN 3 THEN 'Wednesday'

            WHEN 4 THEN 'Thursday'

            WHEN 5 THEN 'Friday'

            WHEN 6 THEN 'Saturday'

            WHEN 7 THEN 'Sunday'

            ELSE 'Invalid Day Number'

        END;

    DBMS\_OUTPUT.PUT\_LINE(

        'Input Number: ' || day\_num || ', Day Name: ' || day\_name

    );

END;

The screenshot shows the Oracle SQL Developer interface. On the left, there's a tree view with a single node expanded. The main area contains PL/SQL code:

```
29
30
31  DECLARE
32  ... day_num INTEGER := 3;
33  ... day_name VARCHAR2(20);
34  BEGIN
35  ... day_name :=
36  ... CASE day_num
37  ... WHEN 1 THEN 'Monday'
38  ... WHEN 2 THEN 'Tuesday'
39  ... WHEN 3 THEN 'Wednesday'
40  ... WHEN 4 THEN 'Thursday'
41  ... WHEN 5 THEN 'Friday'
42  ... WHEN 6 THEN 'Saturday'
43  ... WHEN 7 THEN 'Sunday'
44  ... ELSE 'Invalid Day Number'
45  ... END;
46
47  DBMS_OUTPUT.PUT_LINE(
48  ... 'Input Number: ' || day_num || ', Day Name: ' || day_name
49  );
50
51 /
52
```

Below the code, there are tabs: "Query result", "Script output" (which is selected), "DBMS output", "Explain Plan", and "SQL history". The "Script output" tab shows the results of running the procedure:

Input Number: 3, Day Name: Wednesday

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.008

## 7. Learning Outcome

- Understood the implementation of conditional branching in PostgreSQL's PL/pgSQL.
- Learned the syntax differences between Oracle and PostgreSQL (e.g., using RAISE NOTICE instead of DBMS\_OUTPUT).
- Gained proficiency in using ELSIF ladders for multi-condition evaluations.
- Mastered the CASE statement for handling discrete value mappings efficiently.
- Developed the ability to debug and verify procedural logic within a database administration tool.