

# *Experiment-4*

Student Name: Harshit Kumawat  
Branch: CSE - AIML  
Semester: 4  
Subject Name: DBMS

UID: 24BAI70025  
Section/Group: 24AIT\_KRG G1  
Date of Performance: 04/02/26  
Subject Code: 24CSH-298

## Aim

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 analyse decision-making capabilities in PL/SQL blocks.

## Software Requirements

- Database Management System:
  - PostgreSQL
- Database Administration Tool:
  - pgAdmin

## Objectives

- Implement control structures in PL/SQL (IF-ELSE, ELSE-IF, ELSE-IF LADDER, CASE STATEMENTS in PL-SQL BLOCK).

## Problem Statement

Develop and execute PL/SQL programs that demonstrate the use of conditional control statements. The programs should employ IF-ELSE, ELSIF, ELSIF ladder, and CASE statements to evaluate given conditions and control the flow of execution accordingly, thereby illustrating decision-making capabilities in PL/SQL blocks.

### 1. Problem Statement – IF-ELSE Statement

Write a PL/SQL program to check whether a given number is positive or non-positive using the IF–ELSE conditional control statement and display an appropriate message.

## **2. Problem Statement – IF–ELSIF–ELSE Statement**

Write a PL/SQL program to evaluate the grade of a student based on the obtained marks using the IF–ELSIF–ELSE statement and display the corresponding grade.

## **3. Problem Statement – ELSIF Ladder**

Write a PL/SQL program to determine the performance status of a student based on marks using an ELSIF ladder and display the appropriate result.

## **4. Problem Statement – CASE Statement**

Write a PL/SQL program to display the name of the day based on a given day number using the CASE conditional statement.

# **Practical/Experiment Steps**

- **Control Blocks:**
  - Design and implement distinct PL/SQL program blocks to explore and demonstrate the functionality of each conditional control structure: simple **IF–ELSE**, the **ELSIF Ladder** (IF–ELSIF–ELSE), and the **CASE statement**.
  - Focus on creating diverse conditional logic formats, including simple binary branching and complex multi-path evaluation.
- **Apply Logic Branching for Data Categorization:**
  - Utilize the **IF–ELSE** and **ELSIF Ladder** structures to categorize numerical input data (e.g., student marks) into predefined logical ranges.
  - This enables structured decision-making to automate outcomes such as assigning specific student grades or determining tiered performance statuses.
- **Optimize Selection Logic with CASE Statements:**
  - Implement the **CASE conditional statement** as a streamlined and efficient alternative for mapping discrete input values (e.g., integer day numbers 1 through 7) to their corresponding output names (e.g., 'SUNDAY', 'MONDAY').

- Include a default handler (the `ELSE` clause) to manage invalid or out-of-range inputs gracefully.
- **Integrate Dynamic Messaging for Real-Time Feedback:**
  - Incorporate variable-driven output strings using the `DBMS_OUTPUT.PUT_LINE` procedure within each block.
  - This provides immediate, real-time feedback and clear, descriptive messages to the console based on the logical evaluation of the input conditions.
- **Validate Execution Flow and Decision Logic:**
  - Perform rigorous testing by initializing and executing the blocks with diverse test values (e.g., negative numbers, boundary case marks, and specific day numbers).
  - Rigorously validate the decision-making capabilities of the PL/SQL engine to ensure that the correct logical branch is activated for every test case and the resulting output is as expected.
- 

## Procedure

- **Environment Preparation:**
  - Enable the output server environment (e.g., using `SET SERVEROUTPUT ON` in SQL clients) to ensure all procedural results from `DBMS_OUTPUT.PUT_LINE` are visible in the console window.
- **PL/SQL Block Implementation:**
  - **IF-ELSE:** Construct a basic **IF-ELSE** block to perform a binary check on an input numerical variable, categorizing it as either positive or non-positive.
  - **IF-ELSIF-ELSE (Grading):** Develop an **IF-ELSIF-ELSE** structure to map student marks to specific letter grades based on defined percentage thresholds.
  - **ELSIF Ladder (Performance):** Expand the conditional logic into a comprehensive **ELSIF ladder** to categorize performance into tiered statuses (e.g., Distinction, First Class, Pass).
  - **CASE Statement:** Implement a **CASE statement** block to translate integer inputs into corresponding day names, ensuring a default handler is included for invalid entries.
- **Adherence to Structure:**
  - Nest all procedural logic within the standard `DECLARE`, `BEGIN`, and `END;` blocks to maintain structured programming principles.

- **Testing and Validation:**
  - Initialize diverse test values for each block's variable (e.g., negative numbers, boundary case marks, specific day numbers) to verify the logic accuracy across all conditional branches.
  - Execute each individual block sequentially, monitoring the DBMS output console for the expected string concatenations.
- **Result Verification:**
  - Verify that the output correctly reflects the logic branch associated with the assigned variable values and document the results.
  - Cross-check the console output against manual calculations to ensure the logic and variable handling were correct.
- 

## **Input/Output Analysis**

### SQL Input Queries

```

DECLARE
NUM NUMBER:=-21;

BEGIN
IF NUM>0 THEN
DBMS_OUTPUT.PUT_LINE('IT IS A POSITIVE NUMBER');
ELSE
DBMS_OUTPUT.PUT_LINE('IT IS A NON-POSITIVE NUMBER');
END IF;
END;

```

### **Output**

experiment4.sql\*

```

1  DECLARE
2  NUM NUMBER:=10;
3
4  BEGIN
5  ... IF NUM>0 THEN
6  ... DBMS_OUTPUT.PUT_LINE('IT IS A POSITIVE NUMBER');
7  ... ELSE
8  ... DBMS_OUTPUT.PUT_LINE('IT IS A NON-POSITIVE NUMBER');
9  ... END IF;
10 END;
11
12
13 DECLARE
14 MARKS NUMBER:=48;
15 GRADE VARCHAR(1);
16
17 BEGIN

```

Query result    Script output    DBMS output    Explain Plan    SQL history

SQL> DECLARE  
NUM NUMBER:=10;  
  
BEGIN...  
Show more...

IT IS A POSITIVE NUMBER

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.003

experiment4.sql\*

```

1  DECLARE
2  NUM NUMBER:=-21;
3
4  BEGIN
5  ... IF NUM>0 THEN
6  ... DBMS_OUTPUT.PUT_LINE('IT IS A POSITIVE NUMBER');
7  ... ELSE
8  ... DBMS_OUTPUT.PUT_LINE('IT IS A NON-POSITIVE NUMBER');
9  ... END IF;
10 END;
11
12
13 DECLARE
14 MARKS NUMBER:=48;
15 GRADE VARCHAR(1);
16
17 BEGIN

```

Query result    Script output    DBMS output    Explain Plan    SQL history

SQL> DECLARE  
NUM NUMBER:=-21;  
  
BEGIN...  
Show more...

IT IS A NON-POSITIVE NUMBER

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.005

## SQL Queries Input

```

DECLARE
MARKS NUMBER:=68;
GRADE VARCHAR(1);
```

```

BEGIN
IF MARKS>=90 THEN
GRADE:='A';
ELSIF MARKS>=80 THEN
GRADE:='B';
ELSIF MARKS>=70 THEN
GRADE:='C';
ELSIF MARKS>=60 THEN
GRADE:='D';
ELSE
GRADE:='F';
END IF;
```

```
DBMS_OUTPUT.PUT_LINE('MARKS ='||MARKS||', GRADE ='||GRADE||');
```

```
END;
```

## Output

The image shows two separate sessions in Oracle SQL Developer. Each session has a code editor at the top with the same PL/SQL procedure. Below the code editor is a toolbar with icons for Query result, Script output, DBMS output, Explain Plan, and SQL history. Underneath the toolbar is a status bar with icons for trash, download, and 'Show more...'. The main area below the status bar displays the execution results.

**Session 1 (Left):**

```
12
13  DECLARE
14  MARKS NUMBER:=98;
15  GRADE VARCHAR(1);
16
17  BEGIN
18    ... IF MARKS>=90 THEN
19    ...   GRADE:='A';
20    ... ELSIF MARKS>=80 THEN
21    ...   GRADE:='B';
22    ... ELSIF MARKS>=70 THEN
23    ...   GRADE:='C';
24    ... ELSIF MARKS>=60 THEN
25    ...   GRADE:='D';
26    ... ELSE
27    ...   GRADE:='F';
28  END IF;
29
30  DBMS_OUTPUT.PUT_LINE('MARKS = ' || MARKS || ', GRADE = ' || GRADE);
31
32 END;
33
```

Query result: MARKS = 98, GRADE = A

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.007

**Session 2 (Right):**

```
13  DECLARE
14  MARKS NUMBER:=68;
15  GRADE VARCHAR(1);
16
17  BEGIN
18    ... IF MARKS>=90 THEN
19    ...   GRADE:='A';
20    ... ELSIF MARKS>=80 THEN
21    ...   GRADE:='B';
22    ... ELSIF MARKS>=70 THEN
23    ...   GRADE:='C';
24    ... ELSIF MARKS>=60 THEN
25    ...   GRADE:='D';
26    ... ELSE
27    ...   GRADE:='F';
28  END IF;
29
30  DBMS_OUTPUT.PUT_LINE('MARKS = ' || MARKS || ', GRADE = ' || GRADE);
31
32 END;
33
```

Query result: MARKS = 68, GRADE = D

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.008

## SQL Queries Input

```
DECLARE
```

```
MARKS NUMBER:=58;
```

```
PERFORMANCE VARCHAR(20);
```

```
BEGIN
```

```
  IF MARKS>=75 THEN
```

```
    PERFORMANCE:='DISTINCTION';
```

```
  ELSIF MARKS>=60 THEN
```

```
    PERFORMANCE:='FIRST CLASS';
```

```
  ELSIF MARKS>=50 THEN
```

```
    PERFORMANCE:='SECOND CLASS';
```

```
  ELSIF MARKS>=35 THEN
```

```

PERFORMANCE:='PASS';
ELSE
PERFORMANCE:='FAIL';
END IF;

DBMS_OUTPUT.PUT_LINE('MARKS ='||MARKS|| AND PERFORMANCE
=||PERFORMANCE);
END;

```

## Output

The screenshot shows the Oracle SQL Developer interface with two panes. The left pane displays the PL/SQL code for experiment4.sql, which declares variables MARKS and PERFORMANCE, initializes them, and uses an IF-ELSE-IF-ELSIF-ELSE-END IF construct to determine the performance grade based on marks. It then prints the result using DBMS\_OUTPUT.PUT\_LINE. The right pane shows the results of running the procedure with MARKS set to 38 and PERFORMANCE set to PASS. The output pane displays the message "MARKS = 38 AND PERFORMANCE = PASS" and indicates the procedure was successfully completed.

```

35 DECLARE
36   MARKS NUMBER:=38;
37   PERFORMANCE VARCHAR(20);
38
39 BEGIN
40   . . . IF MARKS>=75 THEN
41   . . .   PERFORMANCE:='DISTINCTION';
42   . . . ELSIF MARKS>=60 THEN
43   . . .   PERFORMANCE:='FIRST CLASS';
44   . . . ELSIF MARKS>=50 THEN
45   . . .   PERFORMANCE:='SECOND CLASS';
46   . . . ELSIF MARKS>=35 THEN
47   . . .   PERFORMANCE:='PASS';
48   . . . ELSE
49   . . .   PERFORMANCE:='FAIL';
50   . . . END IF;
51
52   . . . DBMS_OUTPUT.PUT_LINE('MARKS = '||MARKS|| ' AND PERFORMANCE = '||PERFORMANCE);
53 END;
54

```

Query result    Script output    DBMS output    Explain Plan    SQL history

trash refresh

...  
Show more...

MARKS = 38 AND PERFORMANCE = PASS

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.007

The screenshot shows the Oracle SQL Developer interface with two panes. The left pane displays the same PL/SQL code as the first screenshot, but with MARKS set to 88 and PERFORMANCE set to DISTINCTION. The right pane shows the results of running the procedure with these values. The output pane displays the message "MARKS = 88 AND PERFORMANCE = DISTINCTION" and indicates the procedure was successfully completed.

```

35 DECLARE
36   MARKS NUMBER:=88;
37   PERFORMANCE VARCHAR(20);
38
39 BEGIN
40   . . . IF MARKS>=75 THEN
41   . . .   PERFORMANCE:='DISTINCTION';
42   . . . ELSIF MARKS>=60 THEN
43   . . .   PERFORMANCE:='FIRST CLASS';
44   . . . ELSIF MARKS>=50 THEN
45   . . .   PERFORMANCE:='SECOND CLASS';
46   . . . ELSIF MARKS>=35 THEN
47   . . .   PERFORMANCE:='PASS';
48   . . . ELSE
49   . . .   PERFORMANCE:='FAIL';
50   . . . END IF;
51
52   . . . DBMS_OUTPUT.PUT_LINE('MARKS = '||MARKS|| ' AND PERFORMANCE = '||PERFORMANCE);
53 END;
54

```

Query result    Script output    DBMS output    Explain Plan    SQL history

trash refresh

...  
Show more...

MARKS = 88 AND PERFORMANCE = DISTINCTION

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.011

## SQL Queries Input

```

DECLARE
DAYNUM NUMBER:=3;
DAYNAME VARCHAR(20);

BEGIN
DAYNAME:=CASE DAYNUM
WHEN 1 THEN 'SUNDAY'

```

```

WHEN 2 THEN 'MONDAY'
WHEN 3 THEN 'TUESDAY'
WHEN 4 THEN 'WEDNESDAY'
WHEN 5 THEN 'THURSDAY'
WHEN 6 THEN 'FRIDAY'
WHEN 7 THEN 'SATURDAY'
ELSE 'INVALID DAY'
END;

```

```

DBMS_OUTPUT.PUT_LINE('IT IS'||DAYNAME);
END;

```

## Output

experiment4.sql\*

```

55
56  DECLARE
57  DAYNUM NUMBER:=3;
58  DAYNAME VARCHAR(20);
59
60  BEGIN
61  ... DAYNAME:=CASE DAYNUM
62  ... WHEN 1 THEN 'SUNDAY'
63  ... WHEN 2 THEN 'MONDAY'
64  ... WHEN 3 THEN 'TUESDAY'
65  ... WHEN 4 THEN 'WEDNESDAY'
66  ... WHEN 5 THEN 'THURSDAY'
67  ... WHEN 6 THEN 'FRIDAY'
68  ... WHEN 7 THEN 'SATURDAY'
69  ... ELSE 'INVALID DAY'
70  ...
71  ...
72  ... DBMS_OUTPUT.PUT_LINE('IT IS'||DAYNAME);
73 END;

```

[Query result](#) [Script output](#) [DBMS output](#) [Explain Plan](#) [SQL history](#)



Show more...

IT IS TUESDAY

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.006

experiment4.sql\*

```

55
56  DECLARE
57  DAYNUM NUMBER:=9;
58  DAYNAME VARCHAR(20);
59
60  BEGIN
61  ... DAYNAME:=CASE DAYNUM
62  ... WHEN 1 THEN 'SUNDAY'
63  ... WHEN 2 THEN 'MONDAY'
64  ... WHEN 3 THEN 'TUESDAY'
65  ... WHEN 4 THEN 'WEDNESDAY'
66  ... WHEN 5 THEN 'THURSDAY'
67  ... WHEN 6 THEN 'FRIDAY'
68  ... WHEN 7 THEN 'SATURDAY'
69  ... ELSE 'INVALID DAY'
70  ...
71  ...
72  ... DBMS_OUTPUT.PUT_LINE('IT IS'||DAYNAME);
73 END;

```

[Query result](#) [Script output](#) [DBMS output](#) [Explain Plan](#) [SQL history](#)



Show more...

IT IS INVALID DAY

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.008

## **Learning Outcomes**

- Gained proficiency in using IF-ELSE, ELSIF ladders, and CASE statements to control program execution flow.
- Evaluated data variables to automate specific outcomes, such as student grading or performance status.
- Using CASE statements as a streamlined method for mapping discrete values like day numbers to names.
- Skills in setting logical thresholds to categorize raw numerical marks into descriptive classifications