

Experiment 4

Student Name: Krish Latawa
Branch: CSE - AIML
Semester: 4
Subject Name: DBMS

UID: 24BAI70541
Section/Group: 24AIT_KRG G1
Date of Performance: 4/2/26
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 Structure Implementation: Designed multiple PL/SQL blocks to explore diverse conditional logic formats, including simple branching and multi-path evaluation.
- Logic Branching Analysis: Utilised IF-ELSE and ELSIF ladders to categorize numerical data into specific ranges, such as student grades and performance statuses.
- Selection Optimisation: Implemented the CASE statement as a streamlined alternative to multiple conditional checks for mapping discrete values like day numbers to names.
- Dynamic Messaging: Integrated variable-driven output strings to provide real-time feedback based on the evaluation of input conditions.
- Execution Flow Control: Validated the decision-making capabilities of the PL/SQL engine by testing various input scenarios to ensure the correct code path was activated.

Procedure

- Enabled the output server environment to ensure all procedural results would be visible in the console window.
- Constructed a basic IF-ELSE block to perform a binary check on a numerical variable for positive or non-positive properties.
- Developed an IF-ELSIF-ELSE structure to map student marks to specific letter grades based on defined percentage thresholds.
- Expanded the conditional logic into a comprehensive ELSIF ladder to categorise performance into tiers such as Distinction, First Class, and Pass.
- Implemented a CASE statement block to translate integer inputs into corresponding day names, including a default handler for invalid entries.
- Initialised diverse test values for each variable, such as negative numbers for sign checks and specific marks for grading, to verify logic accuracy.
- Nested the procedural logic within standard BEGIN...END; blocks to maintain structured programming principles.
- Executed each individual block sequentially and monitored the DBMS output console for the expected string concatenations.
- Verified that the output correctly reflected the logic branch associated with the assigned variable values and documented the results.
- Verified the console output against the manual calculations to ensure the logic and variables were handled correctly.

Input/Output Analysis

SQL Input Queries

```

DECLARE
NUM NUMBER:= -2;

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

```
[SQL Worksheet]*
1 DECLARE
2   NUM NUMBER:=2;
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;
```

Query result Script output DBMS output Explain Plan SQL history

SQL> DECLARE
NUM NUMBER:=2;

BEGIN...

IT IS A NON-POSITIVE NUMBER

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.007

```
[SQL Worksheet]*
1 DECLARE
2   NUM NUMBER:=69;
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;
```

Query result Script output DBMS output Explain Plan SQL history

SQL> DECLARE
NUM NUMBER:=69;

BEGIN...

IT IS A POSITIVE NUMBER

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.005

SQL Queries Input

```
DECLARE
MARKS NUMBER:=52;
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

```
experiment4.sql*  
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 **Script output** DBMS output Explain Plan SQL history

Show more...

MARKS = 98, GRADE = A

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.007

```
[SQL Worksheet]*  
1 DECLARE  
2 MARKS NUMBER:=52;  
3 GRADE VARCHAR(1);  
4  
5 BEGIN  
6 ... IF MARKS>=90 THEN  
7 ... GRADE:='A';  
8 ... ELSIF MARKS>=80 THEN  
9 ... GRADE:='B';  
10 ... ELSIF MARKS>=70 THEN  
11 ... GRADE:='C';  
12 ... ELSIF MARKS>=60 THEN  
13 ... GRADE:='D';  
14 ... ELSE  
15 ... GRADE:='F';
```

Query result **Script output** DBMS output Explain Plan SQL history

Show more...

SQL> DECLARE
MARKS NUMBER:=52;
GRADE VARCHAR(1);
...
Show more...

MARKS = 52, GRADE = F

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.009

SQL Queries Input

```
DECLARE  
MARKS NUMBER:=38;  
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 a SQL Worksheet interface. The top pane contains the following PL/SQL code:

```

1 DECLARE
2 MARKS NUMBER:=38;
3 PERFORMANCE VARCHAR(20);
4
5 BEGIN
6 IF MARKS>=75 THEN
7 PERFORMANCE:='DISTINCTION';
8 ELSIF MARKS>=60 THEN
9 PERFORMANCE:='FIRST CLASS';
10 ELSIF MARKS>=50 THEN
11 PERFORMANCE:='SECOND CLASS';
12 ELSIF MARKS>=35 THEN
13 PERFORMANCE:='PASS';
14 ELSE
15 PERFORMANCE:='FAIL';

```

The bottom pane shows the 'Script output' tab with the following output:

```

SQL> DECLARE
MARKS NUMBER:=38;
PERFORMANCE VARCHAR(20);
...
Show more...

MARKS = 38 AND PERFORMANCE = PASS

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.006

```

The screenshot shows a SQL Worksheet interface for a file named 'experiment4.sql*'. The top pane contains the following PL/SQL code:

```

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

```

The bottom pane shows the 'Script output' tab with the following output:

```

...
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;

```

```

experiment4.sql*  ▶ ⌵ ⌵ ⌵ ⌵ ⌵ Aa ⌵
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 ... END;
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*  ▶ ⌵ ⌵ ⌵ ⌵ ⌵ Aa ⌵
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 ... END;
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

Output

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