

Experiment 4

Student Name: Bhavik Pahuja
Branch: CSE - AIML
Semester: 4
Subject Name: DBMS

UID: 24BAI70522
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:= -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;

```

```

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;
```

```
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;
```

```
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

| | |
|---|---|
| SQL> DECLARE DAYNUM NUMBER := 3; DAYNAME VARCHAR(20); BEGIN... Show more... | SQL> DECLARE MARKS NUMBER := 58; PERFORMANCE VARCHAR(20); BEGIN... Show more... |
| IT IS TUESDAY | MARKS = 58 AND PERFORMANCE = SECOND CLASS |
| PL/SQL procedure successfully completed. | PL/SQL procedure successfully completed. |
| Elapsed: 00:00:00.007 | Elapsed: 00:00:00.006 |
| SQL> DECLARE MARKS NUMBER := 68; GRADE VARCHAR(1); BEGIN... Show more... | SQL> DECLARE NUM NUMBER := -21; BEGIN IF NUM > 0 THEN... Show more... |
| MARKS = 68, GRADE = D | IT IS A NON-POSITIVE NUMBER |
| PL/SQL procedure successfully completed. | PL/SQL procedure successfully completed. |
| Elapsed: 00:00:00.007 | Elapsed: 00:00:00.006 |

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