NAME:=Shivansh Iohani

PRN:=23070521141

Introduction to PL/SQL *Conditions*

In PL/SQL, conditions allow decision-making in programs. The two main types of conditional statements are: **IF-THEN**
IF-THEN-ELSE
IF-THEN-ELSIF-ELSE
CASE Statement

IF-THEN Statement

Executes a block of code if the condition is `TRUE`.

Example: Check if a number is positive

```
SET SERVEROUTPUT ON;
 DECLARE
    num NUMBER := 10;
BEGIN
    IF num > 0 THEN
        DBMS_OUTPUT.PUT_LINE('The number is positive.');
    END IF;
END; /
```

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  num NUMBER := 10;
  3  BEGIN
  4  IF num > 0 THEN
  5  DBMS_OUTPUT.PUT_LINE('The number is positive.');
  6  END IF;
  7  END;
  8  /
The number is positive.

PL/SQL procedure successfully completed.

SQL>
```

IF-THEN-ELSE Statement

Executes one block if the condition is `TRUE`, otherwise executes another block.

Example: Check if a number is even or odd

```
SET SERVEROUTPUT ON;

DECLARE     num
NUMBER := 7;
BEGIN
    IF MOD(num, 2) = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Even number');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Odd number');
    END IF;
END; /
```

```
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> DECLARE     num NUMBER := 7;
  2  BEGIN
  3      IF MOD(num, 2) = 0 THEN
  4          DBMS_OUTPUT.PUT_LINE('Even number');
  5      ELSE
  6          DBMS_OUTPUT.PUT_LINE('Odd number');
  7      END IF;
  8  END;
  9  /
Odd number

PL/SQL procedure successfully completed.
```

IF-THEN-ELSIF-ELSE Statement

Check multiple conditions one by one.

Example: Check if a number is positive, negative, or zero

```
SET SERVEROUTPUT ON;
 DECLARE
    num NUMBER := -5; BEGIN
    IF num > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Positive number');
    ELSIF num < 0 THEN
        DBMS_OUTPUT.PUT_LINE('Negative number');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Zero');
    END IF;
END; /
```

```
SQL> SET SERVEROUTPUT ON;
SQL>  DECLARE
  2       num NUMBER := -5; BEGIN
  3       IF num > 0 THEN
  4           DBMS_OUTPUT.PUT_LINE('Positive number');
  5       ELSIF num < 0 THEN
  6           DBMS_OUTPUT.PUT_LINE('Negative number');
  7       ELSE
  8           DBMS_OUTPUT.PUT_LINE('Zero');
  9       END IF;
 10  END;
 11  /
Negative number

PL/SQL procedure successfully completed.
```

CASE Statement

The `CASE` statement is used to handle multiple conditions more efficiently.

Example: Grade Calculation Using CASE

```
SET SERVEROUTPUT ON;
 DECLARE
    marks NUMBER := 85;
grade VARCHAR2(10); BEGIN
    grade := CASE
                WHEN marks >= 90 THEN 'A'
                WHEN marks >= 80 THEN 'B'
                WHEN marks >= 70 THEN 'C'
                ELSE 'Fail'
             END;

    DBMS_OUTPUT.PUT_LINE('Grade: ' || grade);
END; /
```

```
SQL> SET SERVEROUTPUT ON;
SQL>  DECLARE
  2      marks NUMBER := 85;     grade VARCHAR2(10); BEGIN
  3      grade := CASE
  4                  WHEN marks >= 90 THEN 'A'
  5                  WHEN marks >= 80 THEN 'B'
  6                  WHEN marks >= 70 THEN 'C'
  7                  ELSE 'Fail'
  8                END;
  9
 10      DBMS_OUTPUT.PUT_LINE('Grade: ' || grade);
 11  END;
 12  /
Grade: B

PL/SQL procedure successfully completed.
```

BBB B B

Simple Tasks for Practice

1. Write a PL/SQL program to check whether a number is **divisible by 5**.

```
SQL> DECLARE
  2    num NUMBER := 25; -- Change the number to test
  3  BEGIN
  4    IF MOD(num, 5) = 0 THEN
  5      DBMS_OUTPUT.PUT_LINE(num || ' is divisible by 5.');
  6    ELSE
  7      DBMS_OUTPUT.PUT_LINE(num || ' is not divisible by 5.');
  8    END IF;
  9  END;
 10  /
25 is divisible by 5.

PL/SQL procedure successfully completed.
```

2. Modify the **grade program** to include more conditions (e.g., `60-70` for **D**, `below 60` for **F**).

```
SQL> DECLARE
  2    marks NUMBER := 65; -- Change the marks to test
  3    grade CHAR(1);
  4  BEGIN
  5    IF marks >= 90 THEN
  6      grade := 'A';
  7    ELSIF marks >= 80 THEN
  8      grade := 'B';
  9    ELSIF marks >= 70 THEN
 10      grade := 'C';
 11    ELSIF marks >= 60 THEN
 12      grade := 'D';
 13    ELSE
 14      grade := 'F';
 15    END IF;
 16
 17    DBMS_OUTPUT.PUT_LINE('The grade is: ' || grade);
 18  END;
 19  /
The grade is: D

PL/SQL procedure successfully completed.
```

3. Write a **CASE statement** to display the day of the week based on a number input (1 = Monday, 2 = Tuesday, etc.).

```
SQL> DECLARE
  2    day_num NUMBER := 3; -- Change the number to test
  3    day_name VARCHAR2(15);
  4  BEGIN
  5    day_name := CASE day_num
  6      WHEN 1 THEN 'Monday'
  7      WHEN 2 THEN 'Tuesday'
  8      WHEN 3 THEN 'Wednesday'
  9      WHEN 4 THEN 'Thursday'
 10      WHEN 5 THEN 'Friday'
 11      WHEN 6 THEN 'Saturday'
 12      WHEN 7 THEN 'Sunday'
 13      ELSE 'Invalid Day'
 14    END;
 15
 16    DBMS_OUTPUT.PUT_LINE('Day: ' || day_name);
 17  END;
 18  /
Day: Wednesday

PL/SQL procedure successfully completed.
```

4. Create a program that **checks the largest of three numbers** using `IF-THEN-ELSIF`.

```
SQL> DECLARE
  2    a NUMBER := 15;
  3    b NUMBER := 30;
  4    c NUMBER := 25;
  5    largest NUMBER;
  6  BEGIN
  7    IF a >= b AND a >= c THEN
  8      largest := a;
  9    ELSIF b >= a AND b >= c THEN
 10      largest := b;
 11    ELSE
 12      largest := c;
 13    END IF;
 14
 15    DBMS_OUTPUT.PUT_LINE('The largest number is: ' || largest);
 16  END;
 17  /
The largest number is: 30

PL/SQL procedure successfully completed.
```