NAME: Shreya Bansal
PRN: 23070521144
SEC: B2

Introduction to PL/SQL Conditions

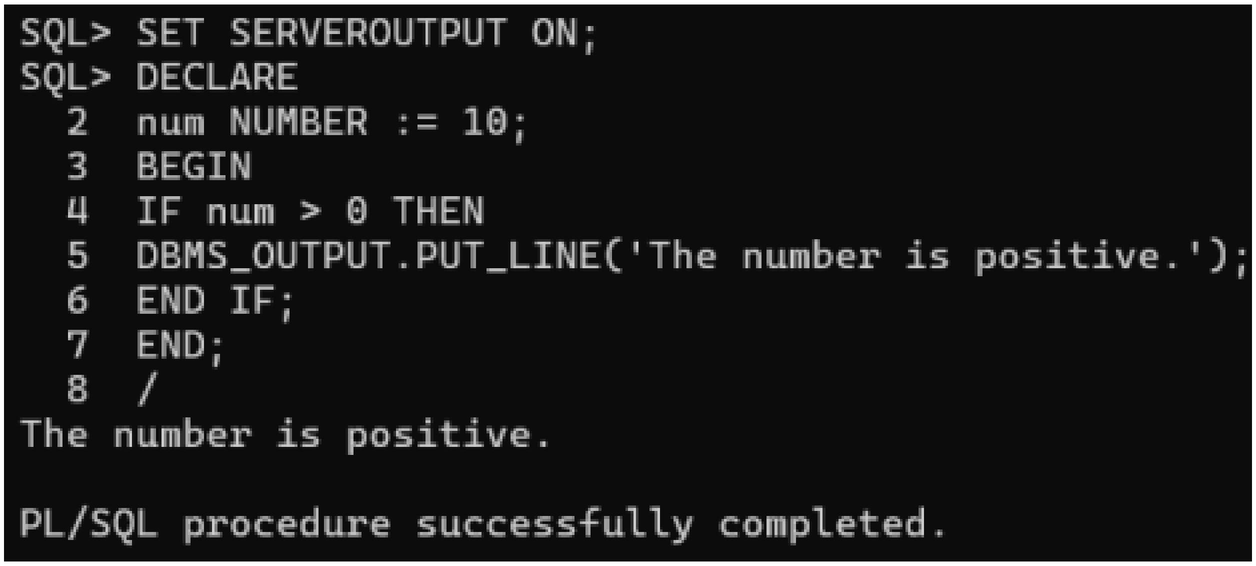
In PL/SQL, conditions allow decision-making in programs. The two main types of conditional statements are:
IF-THEN
IF-THEN-ELSE
IF-THEN-ELSIF-ELSE
CASE Statement

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;
/
```



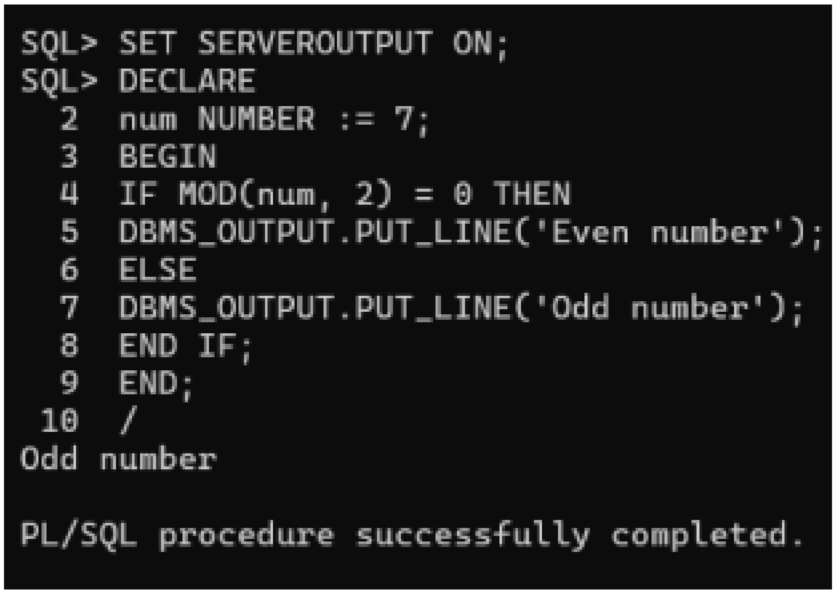
```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  num NUMBER := 10;
  3  BEGIN
  4  IF num > 0 THEN
  5  DBMS_OUTPUT.PUT_LINE('The number is positive.');
  6  END IF;
  7  END;
  8  /
The number is positive.

PL/SQL procedure successfully completed.
```

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;
/
```



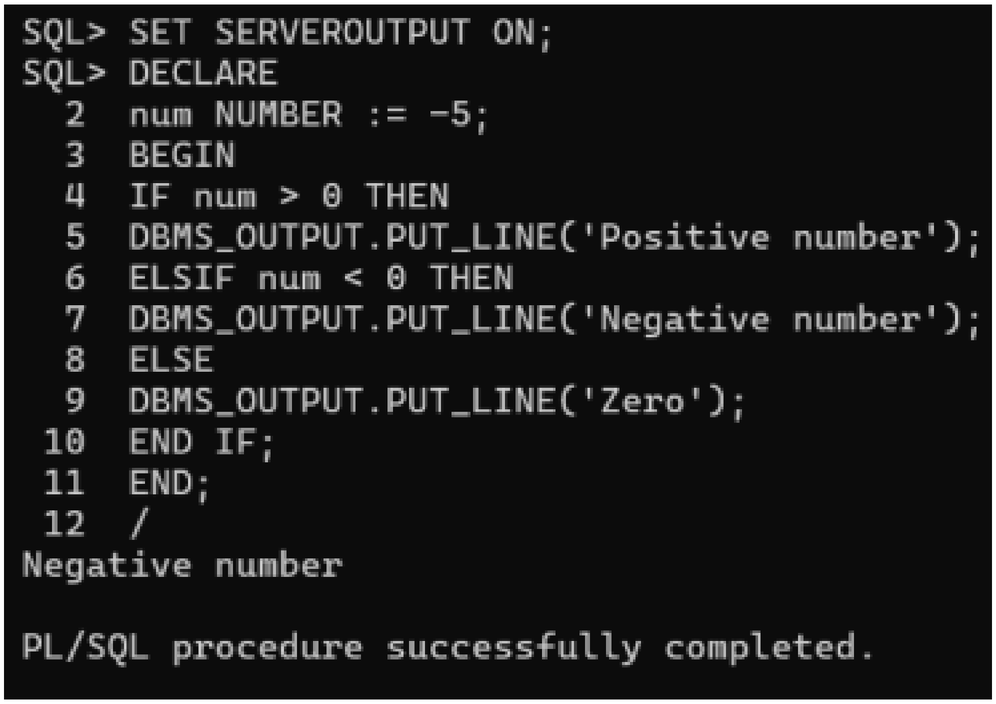
```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  num NUMBER := 7;
  3  BEGIN
  4  IF MOD(num, 2) = 0 THEN
  5  DBMS_OUTPUT.PUT_LINE('Even number');
  6  ELSE
  7  DBMS_OUTPUT.PUT_LINE('Odd number');
  8  END IF;
  9  END;
 10  /
Odd number

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;
/
```

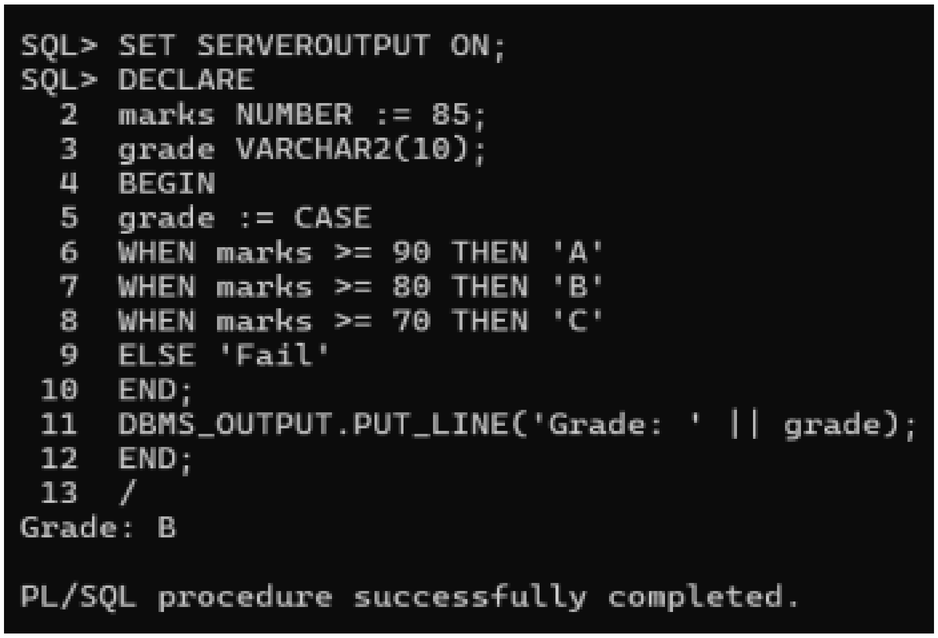

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  num NUMBER := -5;
  3  BEGIN
  4  IF num > 0 THEN
  5  DBMS_OUTPUT.PUT_LINE('Positive number');
  6  ELSIF num < 0 THEN
  7  DBMS_OUTPUT.PUT_LINE('Negative number');
  8  ELSE
  9  DBMS_OUTPUT.PUT_LINE('Zero');
 10  END IF;
 11  END;
 12  /
Negative number

PL/SQL procedure successfully completed.
```

CASE Statement

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

Example: Grade Calculation Using CASE

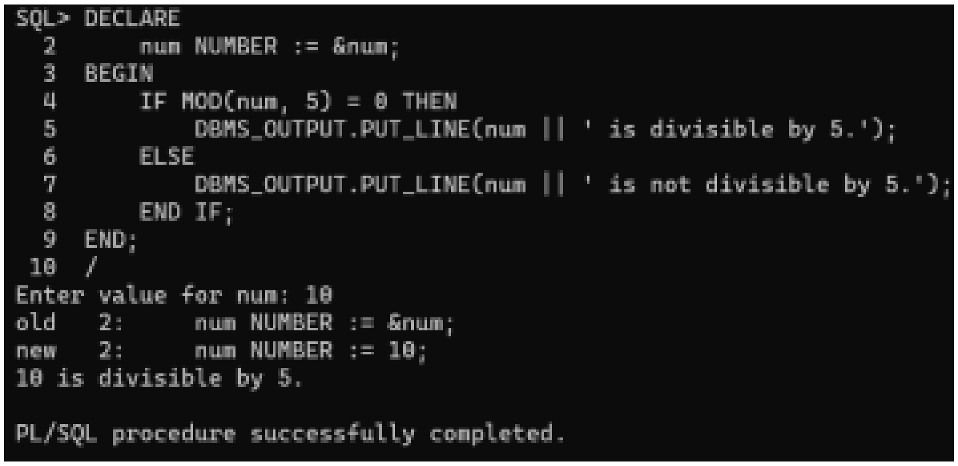
```
SET SERVEROUTPUT ON;

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
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;
/
Grade: B

PL/SQL procedure successfully completed.
```

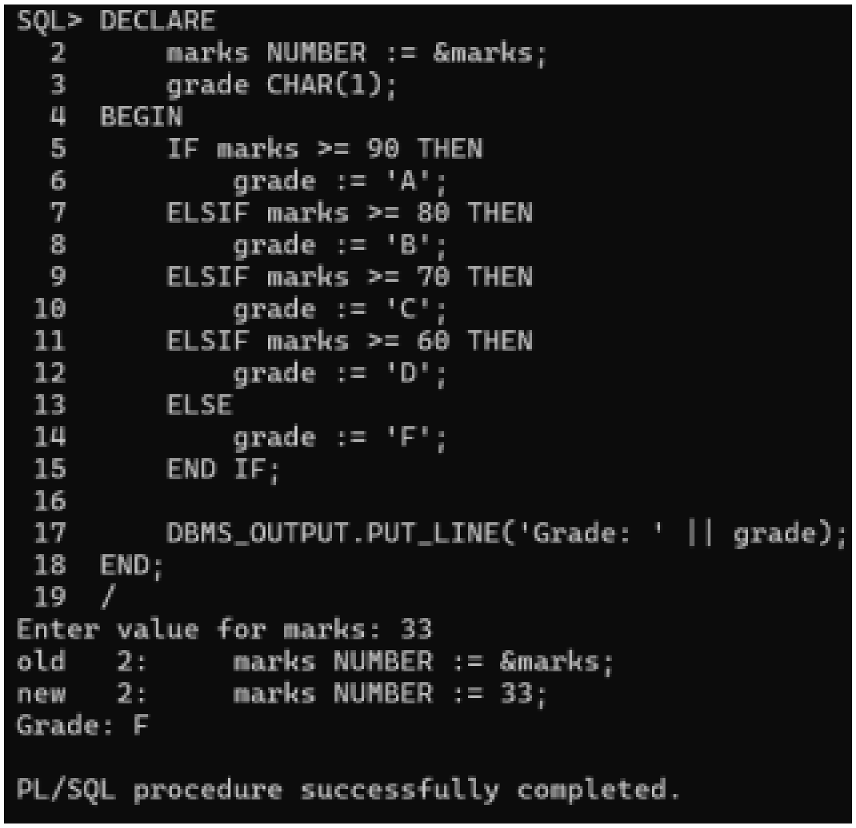
Simple Tasks for Practice

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



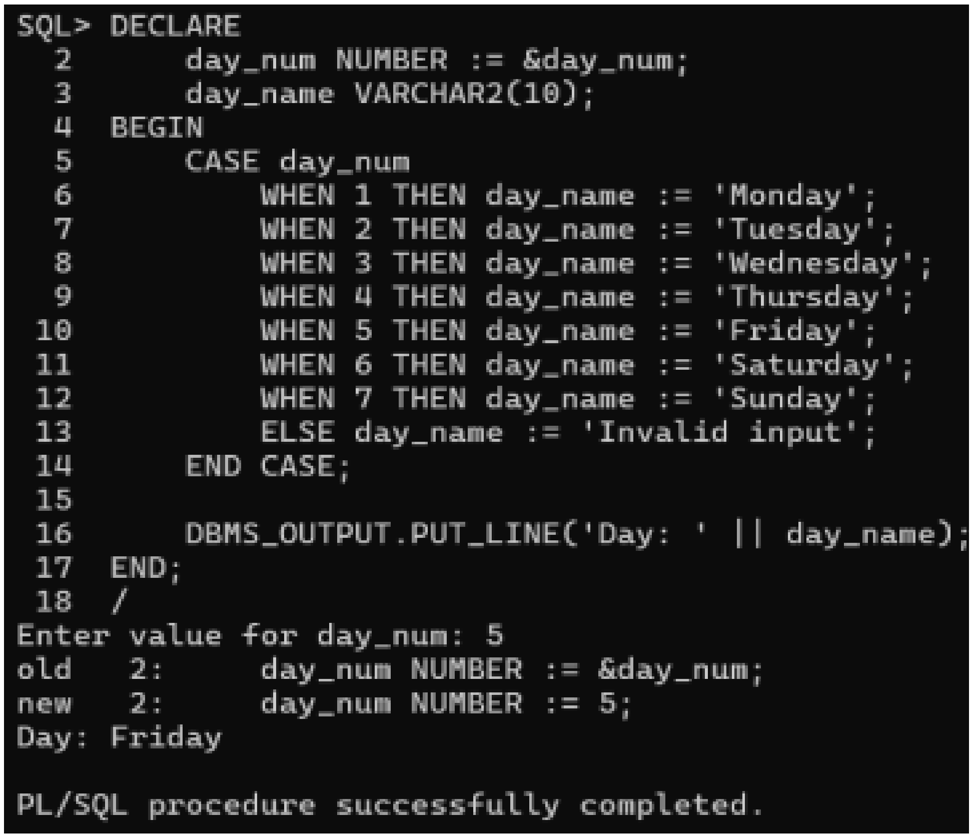
```
SQL> DECLARE
  2      num NUMBER := &num;
  3  BEGIN
  4      IF MOD(num, 5) = 0 THEN
  5          DBMS_OUTPUT.PUT_LINE(num || ' is divisible by 5.');
  6      ELSE
  7          DBMS_OUTPUT.PUT_LINE(num || ' is not divisible by 5.');
  8      END IF;
  9  END;
 10  /
Enter value for num: 10
old   2:     num NUMBER := &num;
new   2:     num NUMBER := 10;
10 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**).



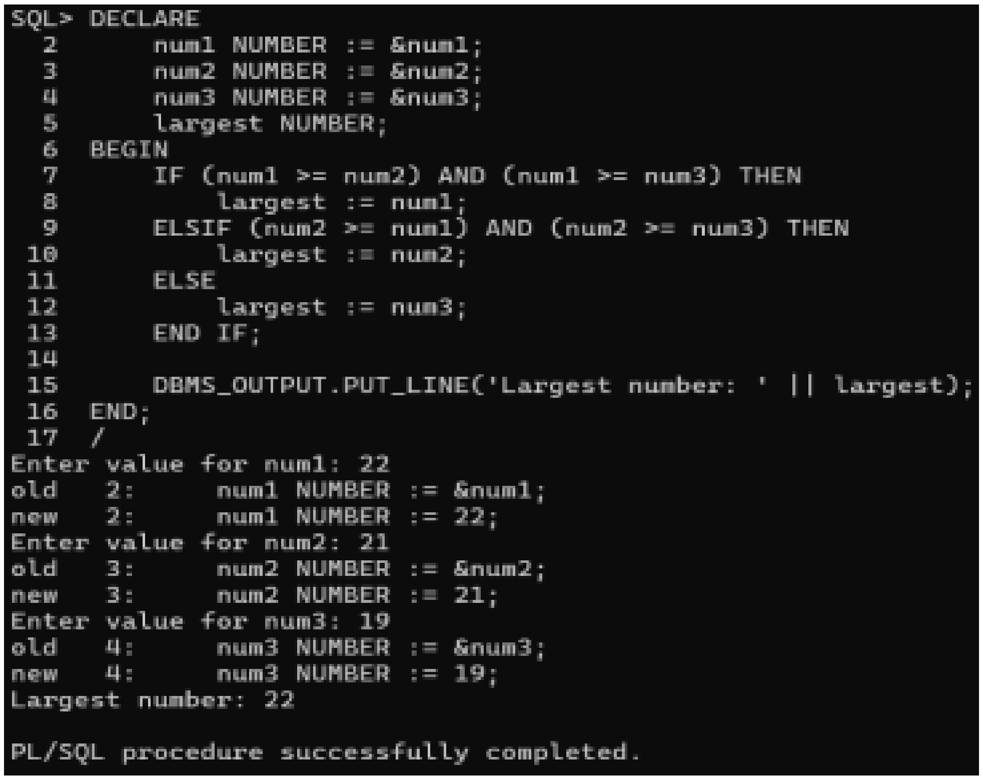
```
SQL> DECLARE
  2      marks NUMBER := &marks;
  3      grade CHAR(1);
  4  BEGIN
  5      IF marks >= 90 THEN
  6          grade := 'A';
  7      ELSIF marks >= 80 THEN
  8          grade := 'B';
  9      ELSIF marks >= 70 THEN
 10          grade := 'C';
 11      ELSIF marks >= 60 THEN
 12          grade := 'D';
 13      ELSE
 14          grade := 'F';
 15      END IF;
 16
 17      DBMS_OUTPUT.PUT_LINE('Grade: ' || grade);
 18  END;
 19  /
Enter value for marks: 33
old   2:     marks NUMBER := &marks;
new   2:     marks NUMBER := 33;
Grade: F

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
        day_num NUMBER := &day_num;
        day_name VARCHAR2(10);
    BEGIN
        CASE day_num
            WHEN 1 THEN day_name := 'Monday';
            WHEN 2 THEN day_name := 'Tuesday';
            WHEN 3 THEN day_name := 'Wednesday';
            WHEN 4 THEN day_name := 'Thursday';
            WHEN 5 THEN day_name := 'Friday';
            WHEN 6 THEN day_name := 'Saturday';
            WHEN 7 THEN day_name := 'Sunday';
            ELSE day_name := 'Invalid input';
        END CASE;

        DBMS_OUTPUT.PUT_LINE('Day: ' || day_name);
    END;
    /
Enter value for day_num: 5
old   2:     day_num NUMBER := &day_num;
new   2:     day_num NUMBER := 5;
Day: Friday

PL/SQL procedure successfully completed.
```

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



```
SQL> DECLARE
        num1 NUMBER := &num1;
        num2 NUMBER := &num2;
        num3 NUMBER := &num3;
        largest NUMBER;
    BEGIN
        IF (num1 >= num2) AND (num1 >= num3) THEN
            largest := num1;
        ELSIF (num2 >= num1) AND (num2 >= num3) THEN
            largest := num2;
        ELSE
            largest := num3;
        END IF;

        DBMS_OUTPUT.PUT_LINE('Largest number: ' || largest);
    END;
    /
Enter value for num1: 22
old   2:     num1 NUMBER := &num1;
new   2:     num1 NUMBER := 22;
Enter value for num2: 21
old   3:     num2 NUMBER := &num2;
new   3:     num2 NUMBER := 21;
Enter value for num3: 19
old   4:     num3 NUMBER := &num3;
new   4:     num3 NUMBER := 19;
Largest number: 22

PL/SQL procedure successfully completed.
```