| Ex.No.6 | PL/SQL STATEMENTS |
|---|---|

## AIM:

To execute and analyze various PL/SQL Control Statements and Programs**,** demonstrating structured programming concepts like loops

## PL/SQL Control Statements

## 1.Simple IF-THEN Statement:

```
SQL> DECLARE
n NUMBER;
BEGIN
  n := &n;
  IF n > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Given number is Greater than ZERO');
  END IF;
END;
/
Enter value for n: 8
Given number is Greater than ZERO
```

## 2.Simple IF-THEN-ELSE Statement:

```
SQL> DECLARE
  n NUMBER;
BEGIN
  n := 20;
  IF n > 15 THEN
    DBMS_OUTPUT.PUT_LINE('Given number is Greater than 15');
  ELSE
```

```
    DBMS_OUTPUT.PUT_LINE('Given number is Less than or Equal to 15');
  END IF;
END;
/
Given number is Greater than 15
```

### 3.Nested IF-THEN-ELSE Statement:

```
SQL> DECLARE
 n NUMBER;
BEGIN
  n := &n;
  IF n > 0 THEN
    DBMS_OUTPUT.PUT_LINE('The number is greater than zero');
  ELSE
    IF n = 0 THEN
      DBMS_OUTPUT.PUT_LINE('The number is zero');
    ELSE
      DBMS_OUTPUT.PUT_LINE('The number is less than zero');
    END IF;
  END IF;
END;
/
Enter value for n: -48
The number is less than zero
```

### 4.IF-THEN-ELSIF Statement:

```
SQL> DECLARE
```

```
n NUMBER;
BEGIN
  n := &n;
  IF n > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Given number is Greater than ZERO');
  ELSIF n = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Given number is Equal to ZERO');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Given number is Less than ZERO');
  END IF;
END;
/
Enter value for n: 40
Given number is Greater than ZERO
```

## 5.Extended IF-THEN Statement:

```
SQL> DECLARE
 grade CHAR(1);
BEGIN
  grade := 'A';
  IF grade = 'A' THEN
    DBMS_OUTPUT.PUT_LINE('Excellent');
  ELSIF grade = 'B' THEN
    DBMS_OUTPUT.PUT_LINE('Very Good');
  ELSIF grade = 'C' THEN
    DBMS_OUTPUT.PUT_LINE('Good');
  ELSIF grade = 'D' THEN
    DBMS_OUTPUT.PUT_LINE('Average');
```

```
  ELSE
    DBMS_OUTPUT.PUT_LINE('No such grade');
  END IF;
END;
/
```

Output:

Excellent

## 6.Simple CASE Statement:

```
SQL> DECLARE
  grade CHAR(1);
  BEGIN
  grade := 'B';
  CASE grade
    WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
    WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
    WHEN 'C' THEN DBMS_OUTPUT.PUT_LINE('Good');
    WHEN 'D' THEN DBMS_OUTPUT.PUT_LINE('Average');
    ELSE
      DBMS_OUTPUT.PUT_LINE('No such grade');
  END CASE;
END;
/
```

Output:
Very Good

## 7.Searched CASE Statement:

```
SQL> DECLARE
 grade CHAR(1);
BEGIN
  grade := 'D';
  CASE
    WHEN grade = 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
    WHEN grade = 'B' THEN DBMS_OUTPUT.PUT_LINE('Good');
```

```
    WHEN grade = 'D' THEN DBMS_OUTPUT.PUT_LINE('Pass');
    ELSE
      DBMS_OUTPUT.PUT_LINE('No such grade');
  END CASE;
END;
/
```

Output:
Pass

## 8.EXCEPTION Instead of ELSE Clause in CASE Statement:

```
SQL> DECLARE

 grade CHAR(1);
BEGIN
  grade := 'X';
  CASE
    WHEN grade = 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
    WHEN grade = 'B' THEN DBMS_OUTPUT.PUT_LINE('Good');
    WHEN grade = 'C' THEN DBMS_OUTPUT.PUT_LINE('Fail');
  END CASE;
EXCEPTION
  WHEN CASE_NOT_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No such grade');
END;
/
```

Output:
No such grade

## 9.WHILE-LOOP Statement:

```
SQL> DECLARE

 a NUMBER;

  i NUMBER := 1111;

BEGIN

  a := 1116;

  WHILE i < a LOOP

    DBMS_OUTPUT.PUT_LINE('Value: ' || i);
```

```
    i := i + 1;
  END LOOP;
END;
/
Value: 1111
Value: 1112
Value: 1113
Value: 1114
Value: 1115
```

## 10.FOR-LOOP Statement:

```
SQL> BEGIN
FOR i IN 1..8 LOOP
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(i));
  END LOOP;
END;
/
Output:
1
2
3
4
5
6
7
8
```

## 11.Reverse FOR-LOOP Statement:

```
SQL> BEGIN
FOR i IN REVERSE 1..6 LOOP
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(i));
  END LOOP;
END;
/
```

Output:

```
6
5
4
3
2
1
```

## 12.Simple GOTO Statement:

```
SQL> DECLARE
 p VARCHAR2(30);
  n PLS_INTEGER := 91;
BEGIN
  FOR j IN 2..ROUND(SQRT(n)) LOOP
    IF n MOD j = 0 THEN
      p := ' is not a prime number';
      GOTO print_now;
    END IF;
  END LOOP;
  p := ' is a prime number';
```

```
  <<print_now>>
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(n) || p);
END;
/
```

91 is a prime number

## 13.GOTO STATEMENT TO BRANCH TO AN ENCLOSING BLOCK:

```
SET SERVEROUTPUT ON;
DECLARE
   v_last_name  VARCHAR2(25);
   v_emp_id     NUMBER(6) := 25;
BEGIN
   <<get_name>>

   SELECT last_name INTO v_last_name
   FROM employees
   WHERE employee_id = v_emp_id;

   BEGIN
      DBMS_OUTPUT.PUT_LINE ('Employee ID: ' || v_emp_id || ' -> Last Name: ' ||
v_last_name);

      v_emp_id := v_emp_id + 3;

      IF v_emp_id <= 33 THEN
         GOTO get_name;
      END IF;
```

```
    END;
END;
/
Employee ID: 25 -> Last Name: Clark
Employee ID: 28 -> Last Name: Lewis
Employee ID: 31 -> Last Name: Baker
```

## 14.DO…WHILE STATEMENT:

```
DECLARE
   n_num NUMBER := 4;
BEGIN
   LOOP
      DBMS_OUTPUT.PUT(n_num || ', ');
      n_num := n_num + 4;
      EXIT WHEN n_num > 20;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Final: ' || n_num);
END;
/
4, 8, 12, 16, 20, Final: 24
```

## Example:

## PRIME NUMBER GENERATION

```
DECLARE
   n NUMBER := 50;
   is_prime BOOLEAN;
BEGIN
```

```
    FOR num IN 2..n LOOP
        is_prime := TRUE;

        FOR i IN 2..FLOOR(SQRT(num)) LOOP
            IF num MOD i = 0 THEN
                is_prime := FALSE;
                EXIT;
            END IF;
        END LOOP;

        IF is_prime THEN
            DBMS_OUTPUT.PUT_LINE('The Prime numbers are:');
             DBMS_OUTPUT.PUT_LINE(num);
        END IF;
    END LOOP;
END;
/
```

Output:

The Prime numbers are:

2

3

5

7

9

11

13

17

19

23

29

31

37

41

43

47

## Greatest of Three Number:

```
DECLARE
   a NUMBER := 25;
   b NUMBER := 18;
   c NUMBER := 42;
   max NUMBER;
BEGIN
   max := a;

   IF b > max THEN
      max := b;
   END IF;

   IF c > max THEN
      max := c;
   END IF;

   DBMS_OUTPUT.PUT_LINE('Greatest among ' || a || ', ' || b || ', and ' || c || ' is: ' || max);
END;
/
```

Output:

Greatest among 25, 18, 42 is: 42

## Check if a Number is Even or Odd

```
DECLARE
   num NUMBER := 29;
BEGIN
   IF MOD(num, 2) = 0 THEN
      DBMS_OUTPUT.PUT_LINE(num || ' is Even.');
   ELSE
      DBMS_OUTPUT.PUT_LINE(num || ' is Odd.');
   END IF;
END;
/
```

Output:

29 is Odd.

## Checking Palindrome

```
DECLARE
   a VARCHAR2(100) := 'kabesh';
   b VARCHAR2(100);
BEGIN
   b := '';

   FOR i IN REVERSE 1..LENGTH(a) LOOP
      b := b|| SUBSTR(a, i, 1);
   END LOOP;

   IF a = b THEN
      DBMS_OUTPUT.PUT_LINE(a || ' is a palindrome.');
```

```
    ELSE
        DBMS_OUTPUT.PUT_LINE(a || ' is not a palindrome.');
    END IF;
END;
/
```

kabesh is not a palindrome.

## PL/SQL BLOCK FOR INSERTION INTO A TABLE

```
DECLARE
    v_employee_id NUMBER := 75;
    v_first_name  VARCHAR2(50) := 'Kabesh';
    v_last_name   VARCHAR2(50) := 'M';
    v_salary      NUMBER := 75000;
BEGIN
    INSERT INTO employees (employee_id, first_name, last_name, salary)
    VALUES (v_employee_id, v_first_name, v_last_name, v_salary);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Record inserted successfully.');
END;
/
```

Output:

Record inserted successfully.

| CONTENTS | MARKS ALLOTED | MARKS OBTAINED |
|---|---|---|
| Aim,Algorithm,SQL,PL/SQL | 30 | |
| Execution and Result | 20 | |
| Viva | 10 | |
| Total | 60 | |

**RESULT**

Thus PL/SQL Control Statements and PL/SQL Programs were executed.