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: 5
Given number is Greater than ZERO
PL/SQL procedure successfully completed.
```

2.Simple IF-THEN-ELSE Statement

```
SQL> DECLARE
  n NUMBER;
BEGIN
  n := 12;
  IF n > 10 THEN
    DBMS_OUTPUT.PUT_LINE('Given number is Greater than 10');
```

```
  ELSE
    DBMS_OUTPUT.PUT_LINE('Given number is Less than or Equal to 10');
  END IF;
END;
/
Given number is Greater than 10
PL/SQL procedure successfully completed.
```

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: -4
The number is less than zero
PL/SQL procedure successfully completed.
```

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: 7
Given number is Greater than ZERO
PL/SQL procedure successfully completed.
```

5.Extended IF-THEN Statement

```
SQL> DECLARE
 grade CHAR(1);
BEGIN
  grade := 'C';
  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;
/
Good
PL/SQL procedure successfully completed.
```

6.Simple CASE Statement

```
SQL> DECLARE
  grade CHAR(1);
BEGIN
  grade := 'A';
  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;
/
Excellent
PL/SQL procedure successfully completed.
```

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;
/

Pass

PL/SQL procedure successfully completed.
```

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;
/

No such grade

PL/SQL procedure successfully completed.
```

9.WHILE-LOOP Statement

```
SQL> DECLARE
  a NUMBER;
   i NUMBER := 1;
BEGIN
   a := 6;
   WHILE i < a LOOP
      DBMS_OUTPUT.PUT_LINE('Value: ' || i);
      i := i + 1;
   END LOOP;
END;
/
Value: 1
Value: 2
Value: 3
Value: 4
Value: 5
PL/SQL procedure successfully completed.
```

10.FOR-LOOP Statement

```
SQL> BEGIN
FOR i IN 1..4 LOOP
      DBMS_OUTPUT.PUT_LINE(TO_CHAR(i));
   END LOOP;
END;
/
1
```

```
2
3
4
PL/SQL procedure successfully completed.
```

11.Reverse FOR-LOOP Statement

```
SQL> BEGIN
FOR i IN REVERSE 1..4 LOOP
     DBMS_OUTPUT.PUT_LINE(TO_CHAR(i));
  END LOOP;
END;
/
4
3
2
1
PL/SQL procedure successfully completed.
```

12.Simple GOTO Statement

```
SQL> DECLARE
  p VARCHAR2(30);
   n PLS_INTEGER := 29;
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;
/
29 is a prime number
PL/SQL procedure successfully completed.
```

13.GOTO STATEMENT TO BRANCH TO AN ENCLOSING BLOCK

```
CREATE TABLE employees (
   employee_id NUMBER(6) PRIMARY KEY,
   first_name VARCHAR2(25),
   last_name VARCHAR2(25),
   salary NUMBER(10,2)
);
SET SERVEROUTPUT ON;
DECLARE
   v_last_name  VARCHAR2(25);
   v_emp_id     NUMBER(6) := 205;
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 + 7;

    IF v_emp_id <= 225 THEN
      GOTO get_name;
    END IF;
  END;
END;
/
Employee ID: 205 -> Last Name: Clark
Employee ID: 212 -> Last Name: Lewis
Employee ID: 219 -> Last Name: Baker
PL/SQL procedure successfully completed.
```

14.DO…WHILE STATEMENT

```
DECLARE
  n_num NUMBER := 3;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT(n_num || ', ');
    n_num := n_num + 3;
    EXIT WHEN n_num > 15;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Final: ' || n_num);
```

```
END;
/
3, 6, 9, 12, 15, Final: 18
PL/SQL procedure successfully completed.
```

Example:

FACTORIAL

```
DECLARE
   n_num NUMBER := 7;
   factorial NUMBER := 1;
BEGIN
   FOR i IN 1..n_num LOOP
      factorial := factorial * i;
   END LOOP;

   DBMS_OUTPUT.PUT_LINE('Factorial of ' || n_num || ' is ' || factorial);
END;
/
Factorial of 7 is 5040
PL/SQL procedure successfully completed.
```

PRIME NUMBER GENERATION

```
DECLARE
   n_limit NUMBER := 40;
   is_prime BOOLEAN;
BEGIN
   FOR num IN 2..n_limit 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(num || ' is a prime number');
        END IF;
    END LOOP;
END;
/
2 is a prime number
3 is a prime number
5 is a prime number
7 is a prime number
11 is a prime number
13 is a prime number
17 is a prime number
19 is a prime number
23 is a prime number
29 is a prime number
31 is a prime number
37 is a prime number
PL/SQL procedure successfully completed.
```

FIBONACCI SERIES

```
DECLARE
   num_terms NUMBER := 12;
   a NUMBER := 2;
   b NUMBER := 3;
   c NUMBER;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Fibonacci Series:');

   DBMS_OUTPUT.PUT_LINE(a);
   DBMS_OUTPUT.PUT_LINE(b);

   FOR i IN 3..num_terms LOOP
      c := a + b;
      DBMS_OUTPUT.PUT_LINE(c);
      a := b;
      b := c;
   END LOOP;
END;
/
Fibonacci Series:
2
3
5
8
13
21
```

```
34
55
89
144
233
377
PL/SQL procedure successfully completed.
```

Checking Palindrome

```
DECLARE
   original_string VARCHAR2(100) := 'racecar';
   reversed_string VARCHAR2(100);
BEGIN
   reversed_string := '';

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

   IF original_string = reversed_string THEN
      DBMS_OUTPUT.PUT_LINE(original_string || ' is a palindrome.');
   ELSE
      DBMS_OUTPUT.PUT_LINE(original_string || ' is not a palindrome.');
   END IF;
END;
/
racecar is a palindrome.
PL/SQL procedure successfully completed.
```

PL/SQL BLOCK FOR INSERTION INTO A TABLE

```
CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL,
   salary DECIMAL(10,2) NOT NULL
);
DECLARE
   v_employee_id NUMBER := 110;
   v_first_name  VARCHAR2(50) := 'David';
   v_last_name   VARCHAR2(50) := 'Miller';
   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;
/
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.