|  |  |
| --- | --- |
| **Ex.No.6**  **DATE:** | **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 Less 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;

/

Very Good

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 := 88;

v\_first\_name VARCHAR2(50) := 'Streav';

v\_last\_name VARCHAR2(50) := 'Tony';

v\_salary NUMBER := 99920;

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.

SQL>SET SERVEROUTPUT ON;

SQL> SELECT \* FROM employees;

**EMPLOYEE\_ID FIRST\_NAME LAST\_NAME SALARY**

--------------- ----------- ----------- ----------

88 Streav Tony 99920.00

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

# RESULT

Thus, the PL/SQL Control Statements where implemented successfully. PL/SQL Programs were executed and the expected results were obtained.