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

**AIM**

To create and execute PL/SQL Statements

**Simple IF-THEN Statement**

SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

2 n NUMBER;

3 BEGIN

4 n := &n;

5 IF n > 0 THEN

6 DBMS\_OUTPUT.PUT\_LINE('Given number is Greater than ZERO');

7 END IF;

8 END;

9 /

Enter value for n: 5

Given number is Greater than ZERO

**SIMPLE IF-THEN-ELSE STATEMENT**

SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

2 n NUMBER;

3 BEGIN

4 n := &n;

5 IF n > 0 THEN

6 DBMS\_OUTPUT.PUT\_LINE('Given number is Greater than ZERO');

7 ELSE

8 DBMS\_OUTPUT.PUT\_LINE('Given number is Less than or Equal to ZERO');

9 END IF;

10 END;

11 /

Enter value for n: 5

Given number is Greater than ZERO

**NESTED IF-THEN-ELSE STATEMENTS**

SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

2 n NUMBER;

3 BEGIN

4 n := &n;

5 IF n > 0 THEN

6 DBMS\_OUTPUT.PUT\_LINE('Given number is Greater than ZERO');

7 ELSE

8 IF n = 0 THEN

9 DBMS\_OUTPUT.PUT\_LINE('Given number is Equal to ZERO');

10 ELSE

11 DBMS\_OUTPUT.PUT\_LINE('Given number is Less than ZERO');

12 END IF;

13 END IF;

14 END;

15 /

Enter value for n: 5

Given number is Greater than ZERO

**IF-THEN-ELSIF STATEMENT**

SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

2 n NUMBER;

3 BEGIN

4 n := &n;

5 IF n > 0 THEN

6 DBMS\_OUTPUT.PUT\_LINE('Given number is Greater than ZERO');

7 ELSIF n = 0 THEN

8 DBMS\_OUTPUT.PUT\_LINE('Given number is Equal to ZERO');

9 ELSE

10 DBMS\_OUTPUT.PUT\_LINE('Given number is Less than ZERO');

11 END IF;

12 END;

13 /

Enter value for n: 8

Given number is Greater than ZERO

**EXTENDED IF-THEN STATEMENT**

SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

2 grade CHAR(1);

3 BEGIN

4 grade := 'B';

5 IF grade = 'A' THEN

6 DBMS\_OUTPUT.PUT\_LINE('Excellent');

7 ELSIF grade = 'B' THEN

8 DBMS\_OUTPUT.PUT\_LINE('Very Good');

9 ELSIF grade = 'C' THEN

10 DBMS\_OUTPUT.PUT\_LINE('Good');

11 ELSIF grade = 'D' THEN

12 DBMS\_OUTPUT.PUT\_LINE('Fair');

13 ELSIF grade = 'F' THEN

14 DBMS\_OUTPUT.PUT\_LINE('Poor');

15 ELSE

16 DBMS\_OUTPUT.PUT\_LINE('No such grade');

17 END IF;

18 END;

19 /

Very Good

**SIMPLE CASE STATEMENT**

SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

2 grade CHAR(1);

3 BEGIN

4 grade := 'B';

5 CASE grade

6 WHEN 'A' THEN DBMS\_OUTPUT.PUT\_LINE('Excellent');

7 WHEN 'B' THEN DBMS\_OUTPUT.PUT\_LINE('Very Good');

8 WHEN 'C' THEN DBMS\_OUTPUT.PUT\_LINE('Good');

9 WHEN 'D' THEN DBMS\_OUTPUT.PUT\_LINE('Fair');

10 WHEN 'F' THEN DBMS\_OUTPUT.PUT\_LINE('Poor');

11 ELSE DBMS\_OUTPUT.PUT\_LINE('No such grade');

12 END CASE;

13 END;

14 /

Very Good

**EXCEPTION INSTEAD OF ELSE CLAUSE IN CASE STATEMENT**

SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

2 grade CHAR(1);

3 BEGIN

4 grade := 'B';

5 CASE

6 WHEN grade = 'A' THEN DBMS\_OUTPUT.PUT\_LINE('Excellent');

7 WHEN grade = 'B' THEN DBMS\_OUTPUT.PUT\_LINE('Very Good');

8 WHEN grade = 'C' THEN DBMS\_OUTPUT.PUT\_LINE('Good');

9 WHEN grade = 'D' THEN DBMS\_OUTPUT.PUT\_LINE('Fair');

10 WHEN grade = 'F' THEN DBMS\_OUTPUT.PUT\_LINE('Poor');

11 ELSE

12 DBMS\_OUTPUT.PUT\_LINE('No such grade');

13 END CASE;

14 EXCEPTION

15 WHEN OTHERS THEN

16 DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred');

17 END;

18 /

Very Good

**WHILE-LOOP STATEMENT**

SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

2 A NUMBER;

3 I NUMBER := 1;

4 BEGIN

5 A := 10;

6 WHILE I < A LOOP

7 DBMS\_OUTPUT.PUT\_LINE('VALUE :' || I);

8 I := I + 1;

9 END LOOP;

10 END;

11 /

VALUE :1

VALUE :2

VALUE :3

VALUE :4

VALUE :5

VALUE :6

VALUE :7

VALUE :8

VALUE :9

**FOR-LOOP Statement**

SQL> SET SERVEROUTPUT ON;

SQL> BEGIN

2 FOR i IN 1..3 LOOP

3 DBMS\_OUTPUT.PUT\_LINE (TO\_CHAR(i));

4 END LOOP;

5 END;

6 /

1

2

3

**REVERSE FOR-LOOP STATEMENT**

SQL> SET SERVEROUTPUT ON;

SQL> BEGIN

2 FOR i IN REVERSE 1..3 LOOP

3 DBMS\_OUTPUT.PUT\_LINE (TO\_CHAR(i));

4 END LOOP;

5 END;

6 /

3

2

1

**SIMPLE GOTO STATEMENT**

SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

2 p VARCHAR2(30);

3 n PLS\_INTEGER := 37;

4 BEGIN

5 FOR j IN 2..ROUND(SQRT(n)) LOOP

6 IF n MOD j = 0 THEN

7 p := ' is not a prime number';

8 GOTO print\_now;

9 END IF;

10 END LOOP;

11 p := ' is a prime number';

12

13 <<print\_now>>

14 DBMS\_OUTPUT.PUT\_LINE(TO\_CHAR(n) || p);

15 END;

16 /

37 is a prime number

**GOTO STATEMENT TO BRANCH TO AN ENCLOSING BLOCK**

SET SERVEROUTPUT ON;

DECLARE

v\_last\_name VARCHAR2(25);

v\_emp\_id NUMBER(6) := 120;

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 + 5;

IF v\_emp\_id <= 130 THEN

GOTO get\_name; -- Jump to get\_name for next iteration

END IF;

END;

END;

/

Employee ID: 120 -> Last Name: King

Employee ID: 125 -> Last Name: Hunt

Employee ID: 130 -> Last Name: Oliver

**DO…WHILE STATEMENT**

DECLARE

n\_num NUMBER := 1;

BEGIN

LOOP

DBMS\_OUTPUT.PUT(n\_num || ', ');

n\_num := n\_num + 1;

EXIT WHEN n\_num > 5;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Final: ' || n\_num);

END;

/

1, 2, 3, 4, 5, Final: 6

**FACTORIAL**

DECLARE

n\_num NUMBER := 5;

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 5 is 120

PRIME NUMBER GENERATION

DECLARE

n\_limit NUMBER := 50; -- Change this value to generate primes up to a different limit

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

41 is a prime number

43 is a prime number

47 is a prime number

**FIBONACCI SERIES**

DECLARE

num\_terms NUMBER := 10;

a NUMBER := 0;

b NUMBER := 1;

c NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Fibonacci Series:');

DBMS\_OUTPUT.PUT\_LINE(a); -- First number: 0

DBMS\_OUTPUT.PUT\_LINE(b); -- Second number: 1

FOR i IN 3..num\_terms LOOP

c := a + b;

DBMS\_OUTPUT.PUT\_LINE(c);

a := b;

b := c;

END LOOP;

END;

/

Fibonacci Series:

0

1

1

2

3

5

8

13

21

34

Checking Palindrome

DECLARE

original\_string VARCHAR2(100) := 'madam';

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;

/

madam is a palindrome.

**PL/SQL BLOCK FOR INSERTION INTO A TABLE**

DECLARE

v\_employee\_id NUMBER := 101;

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

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

v\_salary NUMBER := 50000;

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;

/

DECLARE

ENO1 empdet.eno%type;

ENAME1 empdet.name%type;

DEPTNO1 empdet.deptno%type;

BASIC1 empdet.basic%type;

HRA1 empdet.HRA%type;

DA1 empdet.DA%type;

PF1 empdet.pf%type;

NETPAY1 empdet.netpay%type;

BEGIN

ENO1 := &ENO1;

ENAME1 := '&ENAME1';

DEPTNO1 := &DEPTNO1;

BASIC1 := &BASIC1;

HRA1 := (BASIC1 \* 50) / 100;

DA1 := (BASIC1 \* 20) / 100;

PF1 := (BASIC1 \* 7) / 100;

NETPAY1 := BASIC1 + HRA1 + DA1 - PF1;

INSERT INTO EMPDET (eno, name, deptno, basic, hra, da, pf, netpay)

VALUES (ENO1, ENAME1, DEPTNO1, BASIC1, HRA1, DA1, PF1, NETPAY1);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Employee details inserted successfully!');

END;

/

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

**RESULT**

Thus PL/SQL Statements are executed and verified.