Print Statement

```
SQL> DECLARE
  2    -- variable declaration
  3  message  varchar2(20):= 'Hello World!';
  4  BEGIN
  5  --output
  6  dbms_output.put_line(message);
  7  END;
  8  /
Hello World!
```

Initialising Variables

```
SQL> DECLARE
  2     a integer := 30;
  3     b integer := 40;
  4     c integer;
  5     f real;
  6  BEGIN
  7     c := a + b;
  8     dbms_output.put_line('Value of c: ' || c);
  9  --To show the floating type usage
 10     f := 100.0/3.0;
 11     dbms_output.put_line('Value of f: ' || f);
 12  END;
 13  /
Value of c: 70
Value of f: 33.333333333333333333333333333333333333

PL/SQL procedure successfully completed.
```

Global and Local Declarations

```
SQL> DECLARE
  2   -- Global variables
  3     num1 number := 95;
  4     num2 number := 85;
  5  BEGIN
  6     dbms_output.put_line('Outer Variable num1: ' || num1);
  7     dbms_output.put_line('Outer Variable num2: ' || num2);
  8     DECLARE
  9        -- Local variables
 10        num1 number := 195;
 11        num2 number := 185;
 12     BEGIN
 13        dbms_output.put_line('Inner Variable num1: ' || num1);
 14        dbms_output.put_line('Inner Variable num2: ' || num2);
 15     END;
 16
 17  END;
 18  /
Outer Variable num1: 95
Outer Variable num2: 85
Inner Variable num1: 195
Inner Variable num2: 185

PL/SQL procedure successfully completed.
```

Getting Input from the prompt

```
SQL> DECLARE
  2  SALARY1 employee1.salary % TYPE;
  3  ECODE employee1.emp_id % TYPE;
  4  BEGIN
  5  Ecode :=&Ecode;
  6  Select salary into salary1  from employee1 where emp_id = ECODE;
  7  dbms_output.put_line('Salary of ' || ECODE || 'is = || salary1');
  8  END;
  9  /
Enter value for ecode: 11
old   5: Ecode :=&Ecode;
new   5: Ecode :=11;
Salary of 11is = || salary1

PL/SQL procedure successfully completed.
```

Conditional Statements

Program to check if number is even or odd

```
SQL> declare
  2  n number:=&n;
  3
  4  begin
  5  if mod(n,2)=0
  6  then
  7  dbms_output.put_line('number is even');
  8  else
  9  dbms_output.put_line('number is odd');
 10  end if;
 11  end;
 12  /
Enter value for n: 11
old   2: n number:=&n;
new   2: n number:=11;
number is odd

PL/SQL procedure successfully completed.
```

Program to find the maximum of three numbers

```
SQL> Declare
  2  a number;
  3  b number;
  4  c number;
  5  Begin
  6  dbms_output.put_line('Enter a:');
  7  a:=&a;
  8  dbms_output.put_line('Enter b:');
  9  b:=&b;
 10  dbms_output.put_line('Enter c:');
 11  c:=&c;
 12  if (a>b) and (a>c) then
 13  dbms_output.put_line('A is Maximum');
 14  elsif (b>a) and (b>c) then
 15  dbms_output.put_line('B is Maximum');
 16  else
 17  dbms_output.put_line('C is Maximum');
 18  end if;
 19  End;
 20  /
Enter value for a: 10
old   7: a:=&a;
new   7: a:=10;
Enter value for b: 20
old   9: b:=&b;
new   9: b:=20;
Enter value for c: 30
old  11: c:=&c;
new  11: c:=30;
Enter a:
Enter b:
Enter c:
C is Maximum

PL/SQL procedure successfully completed.
```

Iterations in PL/SQL

```
SQL> DECLARE
  2     x number := 10;
  3  BEGIN
  4     LOOP
  5        dbms_output.put_line(x);
  6        x := x + 10;
  7        IF x > 50 THEN
  8           exit;
  9        END IF;
 10     END LOOP;
 11     -- after exit, control resumes here
 12     dbms_output.put_line('After Exit x is: ' || x);
 13  END;
 14  /
10
20
30
40
50
After Exit x is: 60
```

Prime Numbers

```
SQL> DECLARE
  2     i number(3);
  3     j number(3);
  4  BEGIN
  5     i := 2;
  6     LOOP
  7        j:= 2;
  8        LOOP
  9           exit WHEN ((mod(i, j) = 0) or (j = i));
 10           j := j +1;
 11        END LOOP;
 12     IF (j = i ) THEN
 13        dbms_output.put_line(i || ' is prime');
 14     END IF;
 15     i := i + 1;
 16     exit WHEN i = 50;
 17     END LOOP;
 18  END;
 19  /
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
```

For Loop

```
SQL> DECLARE
  2     a number(2);
  3  BEGIN
  4     FOR a in 10 .. 20 LOOP
  5        dbms_output.put_line('value of a: ' || a);
  6    END LOOP;
  7  END;
  8  /
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
value of a: 20
```

Procedure

Find the Minimum Number

```
SQL> DECLARE
  2     a number;
  3     b number;
  4     c number;
  5  PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
  6  BEGIN
  7     IF x < y THEN
  8        z:= x;
  9     ELSE
 10        z:= y;
 11     END IF;
 12  END;
 13  BEGIN
 14     a:= 23;
 15     b:= 45;
 16     findMin(a, b, c);
 17     dbms_output.put_line(' Minimum of (23, 45) : ' || c);
 18  END;
 19  /
Minimum of (23, 45) : 23

PL/SQL procedure successfully completed.
```

Square of a Number

```
SQL> DECLARE
  2     a number;
  3  PROCEDURE squareNum(x IN OUT number) IS
  4  BEGIN
  5    x := x * x;
  6  END;
  7  BEGIN
  8     a:= 23;
  9     squareNum(a);
 10     dbms_output.put_line(' Square of (23): ' || a);
 11  END;
 12  /
Square of (23): 529

PL/SQL procedure successfully completed.
```

Function

```
SQL> DECLARE
  2     num number;
  3     factorial number;
  4
  5  FUNCTION fact(x number)
  6  RETURN number
  7  IS
  8     f number;
  9  BEGIN
 10     IF x=0 THEN
 11        f := 1;
 12     ELSE
 13        f := x * fact(x-1);
 14     END IF;
 15  RETURN f;
 16  END;
 17
 18  BEGIN
 19     num:= 6;
 20     factorial := fact(num);
 21     dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
 22  END;
 23  /
Factorial 6 is 720

PL/SQL procedure successfully completed.
```

Implicit Cursors

```
SQL> DECLARE
  2     total_rows number(2);
  3  BEGIN
  4     UPDATE customers
  5     SET salary = salary + 500;
  6     IF sql%notfound THEN
  7        dbms_output.put_line('no customers selected');
  8     ELSIF sql%found THEN
  9        total_rows := sql%rowcount;
 10        dbms_output.put_line( total_rows || ' customers selected ');
 11     END IF;
 12  END;
 13  /
3 customers selected

PL/SQL procedure successfully completed.
```

Explicit Cursors

```
1
2  DECLARE
3  c_id customers.id%type;
4  c_name customers.name%type;
5  c_addr customers.address%type;
6  CURSOR c_customers is
7  SELECT id, name,address FROM customers;
8
9  BEGIN
10 OPEN c_customers;
11 LOOP
12 FETCH c_customers into c_id, c_name, c_addr;
13 EXIT WHEN c_customers%notfound;
14 dbms_output.put_line(c_id || ' '|| c_name || ' '||c_addr);
15 END LOOP;
16 CLOSE c_customers;
17 END;
18 /
```

```
Statement processed.
101 Alex US
102 Mia Uk
103 Sia Italy
```

Exception Handling

```
1  DECLARE
2  c_id customers.id%type :=8;
3  c_name customers.name%type;
4  c_addr customers.address%type;
5  BEGIN
6  SELECT name, address INTO c_name,c_addr
7  FROM customers
8  WHERE id = c_id;
9  DBMS_OUTPUT.PUT_LINE('Name: '||c_name);
10 DBMS_OUTPUT.PUT_LINE('Address: ' || c_addr);
11 EXCEPTION
12 WHEN no_data_found THEN
13 dbms_output.put_line('No such customer!');
14 WHEN others THEN
15 dbms_output.put_line('Error!');
16 END;
17 /
```

```
Statement processed.
No such customer!
```

Triggers

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
sal_diff number;
BEGIN
 sal_diff := :NEW.salary  - :OLD.salary;
dbms_output.put_line('Old salary: ' || :OLD.salary);
dbms_output.put_line('New salary: ' || :NEW.salary);
dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/



DECLARE
total_rows number(2);
BEGIN
UPDATE  customers
SET salary = salary + 5000;
IF sql%notfound THEN
dbms_output.put_line('no customers updated');
ELSIF sql%found THEN
total_rows := sql%rowcount;
dbms_output.put_line( total_rows || ' customers updated ');
END IF;
END;
 /
```

```
Trigger created.


Statement processed.
Old salary: 10000
New salary: 15000
Salary difference: 5000
Old salary: 15000
New salary: 20000
Salary difference: 5000
Old salary: 16000
New salary: 21000
Salary difference: 5000
3 customers updated
```