Basic Block

```
begin
    DBMS_OUTPUT.PUT_LINE('Hello World !');
end;
```

Declaring Variables

```
declare
    num1 number:=1;
    num2 number:=2;
    res number:=0;
begin
    res := num1 + num2;
    DBMS_OUTPUT.PUT_LINE('Result = ' || res);
end;
```

If Statement

```
declare
    num number:=1;
begin
    if num > 0 then
        DBMS_OUTPUT.PUT_LINE('Positive');
    end if;
end;
```

If then elseif Statement

```
declare
    num number:=0;
begin
    if num > 0 then
        DBMS_OUTPUT.PUT_LINE('Positive');

    elsif num < 0 then
        DBMS_OUTPUT.PUT_LINE('Negative');

    else
        DBMS_OUTPUT.PUT_LINE('Number is 0');
    end if;
end;
```

CASE Statement

```
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('Fair');
        WHEN 'F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
        ELSE DBMS_OUTPUT.PUT_LINE('No such grade');
      END CASE;
   END;
```

CASE Statement using If-ElsIf-Else

```
DECLARE
      grade CHAR(1);
    BEGIN
      grade := 'B';

      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('Fair');
     ELSIF grade = 'F' THEN
       DBMS_OUTPUT.PUT_LINE('Poor');
     ELSE
       DBMS_OUTPUT.PUT_LINE('No such grade');
     END IF;
   END;
```

Use SQL Statement in PL SQL Blocks

```
DECLARE
  job_count NUMBER;
  emp_count NUMBER;
BEGIN
  SELECT COUNT(DISTINCT job_id) INTO job_count FROM HR.employees;
  SELECT COUNT(*) INTO emp_count FROM HR.employees;
  DBMS_OUTPUT.PUT_LINE(emp_count);
END;
```

Simple Loop

```
declare
    i number :=0;
begin
    loop
        DBMS_Output.PUT_LINE(i);
        i := i + 1;
        exit when i = 10;
    end loop;
end;
```

While Loop

```
DECLARE
i INTEGER := 1;
BEGIN
WHILE i <= 10 LOOP
DBMS_OUTPUT.PUT_LINE(i);
i := i+1;
END LOOP;
END;
```

For Loop

```
BEGIN
FOR i IN 1..10 LOOP
DBMS_OUTPUT.PUT_LINE(i);
END LOOP;
END;
```

Associative array indexed by string

```
DECLARE

      TYPE population IS TABLE OF NUMBER
        INDEX BY VARCHAR2(64);

      city_population  population;
      i VARCHAR2(64);

   BEGIN

     city_population('Pune')  := 20000;
     city_population('Mumbai')     := 30000;
     city_population('Delhi') := 10000;

    i:= city_population.first;
```

```
     loop
       DBMS_Output.PUT_LINE(city_population(i));
       i := city_population.next(i);
       exit when i is null;
     END LOOP;
   END;
```

Nested Table array

```
DECLARE
   TYPE emp_salaries IS TABLE OF number;
   e_sal emp_salaries;

BEGIN
    e_sal:=emp_salaries(10000,20000,13000,12780);

    FOR i in e_sal.first..e_sal.last
        LOOP
            DBMS_Output.PUT_LINE(e_sal(i));
        END LOOP;
END;
```

```
DECLARE
   TYPE emp_salaries IS TABLE OF number;
   e_sal emp_salaries;

BEGIN
    select salary bulk collect into e_sal from HR.employees;

    FOR i in e_sal.first..e_sal.last
        LOOP
            DBMS_Output.PUT_LINE(e_sal(i));
        END LOOP;
END;
```

Varray

```
DECLARE
    TYPE e_salaries IS VARRAY(5) OF INTEGER;
    e_sal e_salaries;
BEGIN
    e_sal:= e_salaries(10000,13000,12000,22000);

    FOR i in e_sal.first..e_sal.last
        LOOP
            DBMS_Output.PUT_LINE(e_sal(i));
        END LOOP;
END;
```

Functions

```
create or replace function print_hello return varchar2
is
begin
    return 'Hello World !';
end print_hello;

-- call the function and assign the value to a variable

declare
    msg varchar(100);
begin
    msg := print_hello();
    dbms_output.put_line(msg);
end;

-- function to find square of a number

create or replace function square(num number) return number
is
begin
    return num * num;
end square;

declare
    num number;
begin
    num := square(10);
    dbms_output.put_line(num);
end;

-- find avg salary for a department

create or replace function findAvg(dept_id number) return number
as
    avgSal number;
    begin
        select avg(salary) into avgSal from HR.employees where
         department_id = dept_id;
        return avgSal;
end findAvg;

declare
    avgSal number;
begin
    avgSal := findAvg(90);
    dbms_output.put_line(avgSal);
end;
```

Procedures

```
create or replace procedure print_hello
is
begin
    dbms_output.put_line('Hello World !');
end print_hello;

-- invoke the procedure

begin
    print_hello();
end;

-- procedure to swap two numbers

create or replace procedure swap (
    num1 in out number,
    num2 in out number
)
is
    temp number;
begin
    temp := num1;
    num1 := num2;
    num2 := temp;
end swap;

-- invoke the procedure

declare
    num1 number;
    num2 number;
begin
    num1 := 10;
    num2 := 20;
    dbms_output.put_line('num1 = ' || num1 || ', num2 = ' || num2);
    swap(num1,num2);
    dbms_output.put_line('num1 = ' || num1 || ', num2 = ' || num2);
end;
```

Defining and using an explicit Cursor

```
DECLARE
 CURSOR c1 IS SELECT last_name, job_id FROM HR.employees;
BEGIN
  FOR item IN c1
  LOOP
    DBMS_OUTPUT.PUT_LINE('Name = ' || item.last_name || ', Job = '
|| item.job_id);
  END LOOP;
END;

-- using arrays %TYPE with cursors

DECLARE
  v_jobid     HR.employees.job_id%TYPE;
  v_lastname  HR.employees.last_name%TYPE;
  CURSOR c1 IS SELECT HR.employees.last_name, HR.employees.job_id
FROM HR.employees;

  BEGIN
    OPEN c1;
        LOOP
            FETCH c1 INTO v_lastname, v_jobid;
            EXIT WHEN c1%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(v_lastname || ',' || v_jobid);
        END LOOP;
    CLOSE c1;
    END;
```

Defining and using a Trigger

```
create table account_details (
    acc_no number constraint account_details_pk primary key,
    acc_name varchar2(50),
    acc_balance number
);

-- creating a trigger to restrict value of balance below 1000

create or replace trigger balance_trigger_account_details
before insert or update on account_details
for each row
begin
    if updating then
        if :new.acc_balance < 1000 then
            Raise_Application_Error (-20100, 'Reached Minimum
Balance.');
```

```
        end if;
    end if;
end;

-- updating to set acc_balance = 100

update account_details set acc_balance = 100 where acc_no = 1;

ORA-20100: Reached Minimum Balance. ORA-06512: at
"SQL_VSKNDCEMZVMMJDZSEULYLTCFD.BALANCE_TRIGGER_ACCOUNT_DETAILS"
```