|  |  |
| --- | --- |
|  |  |

DECLARE

var number(3) := 50;

BEGIN

**IF** (var = 10) THEN

dbms\_output.put\_line('Value of var is 10');

ELSIF (var = 20) THEN

dbms\_output.put\_line('Value of var is 20');

ELSIF (var = 30) THEN

dbms\_output.put\_line('Value of var is 30');

**ELSE**

dbms\_output.put\_line('None of the above condition is true.');

END **IF**;

dbms\_output.put\_line('Exact value of var is: '|| var);

END;

/

DECLARE

nameChar **char**(1) := 'J';

BEGIN

**CASE** nameChar

when 'B' then dbms\_output.put\_line('B');

when 'R' then dbms\_output.put\_line('R');

when 'S' then dbms\_output.put\_line('S');

when 'V' then dbms\_output.put\_line('V');

when 'J' then dbms\_output.put\_line('J');

**else** dbms\_output.put\_line('No such name');

END **CASE**;

END;

/

DECLARE

num NUMBER := 1;

BEGIN

LOOP

DBMS\_OUTPUT.PUT\_LINE(num);

**IF** num = 10 THEN

EXIT;

END **IF**;

num := num+1;

END LOOP;

END;

/

DECLARE

num NUMBER := 1;

BEGIN

**WHILE** num <= 10

LOOP

DBMS\_OUTPUT.PUT\_LINE(num);

num := num+1;

END LOOP;

END;

/

DECLARE

BEGIN

**FOR** var IN 1..10

LOOP

DBMS\_OUTPUT.PUT\_LINE(var);

END LOOP;

END;

/

DECLARE

num NUMBER := 0;

BEGIN

**WHILE** num < 10

LOOP

num := num +1;

**IF** num = 5 THEN

**CONTINUE**;

END **IF**;

DBMS\_OUTPUT.PUT\_LINE(num);

END LOOP;

END;

/

DECLARE

i number(1);

j number(1);

BEGIN

<< outer\_loop >>

**FOR** i IN 1..5 LOOP

<< inner\_loop >>

**FOR** j IN 1..5 LOOP

dbms\_output.put\_line('i is: '|| i || ' and j is: ' || j);

END loop inner\_loop;

END loop outer\_loop;

END;

/

DECLARE

num number := 1;

BEGIN

<<loop1>>

-- **while** loop execution

**WHILE** num <= 10 LOOP

dbms\_output.put\_line ('Value of num: ' || num);

num := num + 1;

**IF** num = 5 THEN

num := num + 1;

**GOTO** loop1;

END **IF**;

END LOOP;

END;

/

Stored Procedure:

CREATE OR REPLACE PROCEDURE hello\_world

AS

BEGIN

dbms\_output.put\_line('Hello World!');

END;

/

Before below, you should already have a table called students

CREATE OR REPLACE PROCEDURE add\_student(rollNo IN NUMBER, name IN VARCHAR2)

IS

BEGIN

insert into students values(rollNo,name);

END;

/

Exec/execute add\_student;

Drop procedure add\_student;

Function:

create or replace function getMultiple(num1 in number, num2 in number)

**return** number

is

num3 number(8);

begin

num3 :=num1\*num2;

**return** num3;

end;

/

Implicit cursor: ROWCOUNT is an implicit cursor.

You should have an employees table in prior!

DECLARE var\_rows number(2);

BEGIN

UPDATE employees

SET salary = salary + 2000;

**IF** SQL%NOTFOUND THEN

dbms\_output.put\_line('No record updated.');

ELSIF SQL%FOUND THEN

var\_rows := SQL%ROWCOUNT;

dbms\_output.put\_line(var\_rows || ' records are updated.');

END **IF**;

END;

/

Explicit cursor:

Should have a students table already!

CURSOR cur\_students IS

SELECT rollNo, name, address FROM students;

Using explicit cursor in SP (Stored procedure):

DECLARE

s\_rollNo students.rollNo%type;

s\_name students.name%type;

s\_address students.address%type;

CURSOR cur\_students is

SELECT rollNo, name, address FROM students;

BEGIN

OPEN cur\_students;

LOOP

FETCH cur\_students into s\_rollNo, s\_name, s\_address;

EXIT WHEN cur\_students%notfound;

dbms\_output.put\_line(s\_rollNo || ' ' || s\_name || ' ' || s\_address);

END LOOP;

CLOSE cur\_students;

END;

/

Exceptions: System defined and user defined exceptions:

DECLARE

s\_rollNo students.rollNo%type := 10;

s\_name students.name%type;

s\_address students.address%type;

BEGIN

SELECT rollNo, name, address FROM students WHERE rollNo = s\_rollNo;

dbms\_output.put\_line(s\_rollNo || ' ' || s\_name || ' ' || s\_address);

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line('No such student!');

WHEN others THEN

dbms\_output.put\_line('Error!');

END;

/

Trigger: to automatically trigger some stored procedure!

Select \* from employees;

|  |
| --- |
| EMP\_ID NAME AGE ADDRESS SALAR |

CREATE OR REPLACE TRIGGER show\_salary\_difference

BEFORE DELETE OR INSERT OR UPDATE ON employees

**FOR** EACH ROW

WHEN (**NEW**.EMP\_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;

/