Assignment - 9

Unit - 5

* Program : 1

-- Write a program to calculate the area of a circle and store that value in the table. C_area (RADIUS NUMBER (5), AREA NUMBER (14,2)).

```
SQL > Set Serveroutput on ;
Declare

    r  Number (5) ;
    a  Number (14,2) ;

Begin

    r := &RADIUS ;
    a := 3.14159 * r * r ;

    Insert into C_area values (r,a) ;
    dbms_output.put_line ('Radius : ' ||r) ;
    dbms_output.put_line ('Area of circle :'
        ||a) ;

End ;
/
```

→ Table query :

```
Create table C_area (
        RADIUS Number (5),
        AREA Number (14, 2)
);
```

* Program : 2

Write a program to calculate the Square and Cube of the given number. Write the program that accepts a value from the user then print that value with and without using third variable.

```
SQL> Set Serveroutput on ;

Declare

    num Number ;
    Sq Number ;
    Cube Number ;

Begin

    num := &num ;

    Sq := num * num ;
    Cube := num * num * num ;

    dbms_output.put_line ('Number : ' ||
        num) ;
    dbms_output.put_line ('Square : ' ||
        Sq) ;
    dbms_output.put_line ('Cube : ' || Cube) ;

End ;
/
```

→ using a Third Variable :

```
SQL > Set Serveroutput on ;

Declare

    a Number ;
    b Number ;
    temp Number ;

Begin

    a := &a ;
    b := &b ;

    dbms_output.put_line ('Before
        Swapping : A = ' || a || ',
        B = ' || b );

    temp := a ;
    a := b ;
    b := temp ;

    dbms_output.put_line ('after
        Swapping (using third variable):
        A = ' || a || ', B = ' || b );

End ;
/
```

→ Without using third variable

```
SQL> Set Serveroutput on ;

Declare

    a Number ;
    b Number ;

Begin

    a := &a ;
    b := &b ;

    dbms_output.put_line ('Before
      Swapping : A = ' || a || ', B =
      || b ) ;

    a := a + b ;
    b := a - b ;
    a := a - b ;

    dbms_output.put_line ('after
      Swapping (without third variable):
      A = ' || a || ', B = ' || b ) ;

End ;
/
```

* Program : 3

-- write a program of mark sheet with displays the Seat_No, Name of Student, marks of 5 Subjects, total of 5 Subjects and percentage, also display the class of Student based on the value of percentage.

```
SQL > Set serveroutput on ;

Declare

        Seat_no Number(5);
        name_Student Varchar2(30);
        m1 Number(3);
        m2 Number(3);
        m3 Number(3);
        m4 Number(3);
        m5 Number(3);
        total Number(5);
        percentage Number(5,2);
        class_Student Varchar2(20);

Begin

        Seat_no := &Seat_no;
        name_Student := '&name_Student';
        m1 := &m1;
        m2 := &m2;
```

```
m3 := &m3;
m4 := &m4;
m5 := &m5;

total := m1 + m2 + m3 + m4 + m5;
percentage := (total / 5);

If percentage >= 75 Then
    class_student := 'Distinction';
elsif percentage >= 60 Then
    class_student := 'First class';
elsif percentage >= 50 Then
    class_student := 'Second class';
elsif percentage >= 40 Then
    class_student := 'Pass class';
else
    class_student := 'Fail';
End if;

dbms_output.put_line ('Seat No :'
    || seat_no);
dbms_output.put_line ('Name :' ||
    name_student);
dbms_output.put_line ('Subject 1 :'
    || m1);
dbms_output.put_line ('Subject2 :'
    || m2);
dbms_output.put_line ('Subject 3 :'
    || m3);
```

```
dbms_output.put_line ('Subject 4 :'
    || m4 );
dbms_output.put_line ('Subject 5 :'
    || m5 );
dbms_output.put_line ('Total Marks :
    ' || total );
dbms_output.put_line ('Percentage :'
    || percentage || '%.' );
dbms_output.put_line ('class :'
    || class_student );

End ;
/
```

* Program : 4

-- Write a program that prints value 1 to 100 numbers using FOR loop, using loop command, using WHILE loop command.

→ using FOR Loop :

```
SQL > Set serveroutput on ;

Begin
    FOR i IN 1..100 LOOP
        dbms_output.put_line (i);
    End LOOP ;
End ;
/
```

→ Using LOOP command :

```
SQL > Set serveroutput on ;

Declare

    i Number := 1 ;

Begin

    LOOP
        dbms_output.put_line (i);
        i := i + 1 ;

        EXIT When i > 100 ;
    End LOOP ;

End ;
/
```

→ using while loop command :

```
SQL > Set Serveroutput on ;

Declare

        i Number := 1 ;

Begin

        while i <= 100 LOOP

                dbms_output.put_line (i);
                i := i + 1 ;

        END LOOP ;

End ;
/
```

* Program : 5

-- Write a program that displays the use of %Type variable. This program stores the values of the columns in the memory variables using %Type and %Rowtype variables.

Pending

ma'am will give demo.

shape

* Program : 6

-- Write a simple procedure without any parameter that updates the values in the EMP table.

```
SQL> Set Serveroutput on ;

Create or replace procedure Comm_update
is

Begin

        update emp Set Comm = 500 ;

end Comm_update ;
/
```

* Program : 7

-- Write a simple procedure that increases by the salary of the given department no. by percentage inputted by the user using IN parameter.

```
SQL> Set serveroutput on;

Create or replace procedure inc_sal_emp
  (xdeptno IN number, per IN number)
is

Begin

    update emp set salary = salary +
      (salary * (per/100)) where
      deptno = xdeptno;

    dbms_output.put_line ('Salary
      increased by given per');

end inc_sal_emp;
/
```

→ Calling program :

```
Declare
        d number(2) := &d ;
        p number(2) := &p ;
Begin
        inc_sal_emp (d,p) ;
End ;
/
```

* Program : 8

-- Write a program / procedure that search's whether the given employee number is present or not in the table. (use both IN and OUT mode variables) and also write a PL/SQL block to call the SEARCH_EMP procedure.

```
SQL > Set serveroutput on ;

Create or replace procedure Search_Emp (id
  IN number, xename OUT char, xdeptno
  OUT number, xSalary OUT number)

is
Begin

      Select ename, deptno, Salary INTO
       xename, xdeptno, xSalary from
       emp where empid = id ;

Exception

      when NO_DATA_FOUND Then

          dbms_output.put_line ('Invalid
                employee id ');

End Search_Emp ;
/
```

→ Calling program :

```
Declare

    xid Number(4) := &xid ;
    enm char(15) ;
    d number(2) ;
    Sal number(6) ;

Begin

    Search_Emp(xid, enm, d, Sal) ;
    dbms_output.put_line('Name of
        Employee : ' || enm) ;
    dbms_output.put_line('Deptno :'
        || d) ;
    dbms_output.put_line('Salary :'
        || Sal) ;

End ;
```

* Program : 9

-- write a function that returns the Square of the given number. Execute this function using Separate PL/SQL block and also without using PL/SQL block on command line.

→ Create function that returns the Square

```
Create or Replace Function Squre_num
(n Number) Return Number

is
    Sq Number ;

Begin

    Sq := n * n ;
    Return Sq ;

End ;
/
```

→ Execute using a Separate PL/SQL Block :

```
Declare

    num Number ;
    result Number ;
```

```
Begin

    num :- &num;
    result := Square_num (num);

    dbms_output.put_line ('Square
      of ' || num || 'is ' || result);

End ;
/
```

→ Execute directly (without PL/SQL block) (on command line)

```
Select Square_num (&num) AS
  Square_value FROM dual;
```

* Program : 10

-- Write a function that returns balance for given account number.

```
Create or Replace Function get_balance
(acc_no Number) Return Number

is
    bal Number(10,2);

Begin

    Select balance INTO bal From
      account where accno = acc_no;

    Return bal;

Exception

    When NO_DATA_FOUND Then

        dbms_output.put_line('No
          account found with account
          number :' || acc_no);
        Return NULL;

End;
/
```

→ Calling program:

```
SQL > Set Serveroutput on ;

Declare
        accno Number ;
        bal Number (10, 2) ;

Begin

        accno := &accno ;
        bal := get_balance (accno) ;

        If bal is NOT NULL Then

            dbms_output.put_line
            ('Balance for account No'
            || accno || 'is : ' || bal);
        End if ;

End ;
/
```

→ Table query :

```
Create table account (accno Number,
  Name Varchar2 (20), Balance
  Number (10,2)) ;
```

* Program : 11

// write a trigger to insert the existing values of the EMP table into NewEmp table when the record is deleted from EMP table.

```
SQL > Set Serveroutput on ;

Create or Replace Trigger trg_emp_delete
Before Delete ON EMP

FOR EACH ROW

Begin

        Insert into NewEmp (Empno,
          Ename, Job, Sal ) values
          (:OLD.Empno, :OLD.Ename,
          :OLD.Job, :OLD.Sal );

End ;
/
```

→ Table query :

```
Create table EMP
(
    Empno Number(5),
    Ename Varchar2(30),
    Job Varchar2(20),
    Sal Number(10,2)
);
```

```
Create table NewEmp AS
Select * from Emp ;
```

* Program : 12

-- Write a trigger to insert the existing value of the EMP table into NewEmp table when the record is updated in EMP table.

```
Create or Replace Trigger trg_emp_update
Before update on Emp

For each Row

Begin

    Insert into NewEmp (Empno,
    Ename, Job, Sal) values
    (:OLD.Empno, :OLD.Ename,
    :OLD.Job, :OLD.Sal);

End;
/
```

* Program : 13

-- Write a trigger to insert the values into the NEWEMP table when the records are inserted into the EMP table.

```
Create or Replace Trigger trg_emp_insert
after Insert on Emp

For Each Row

Begin

    Insert into NEWEMP (Empno,
    Empname, Job, Sal) Values
    (:NEW.Empno, :NEW.Ename,
    :NEW.Job, :NEW.Sal);

End;
/
```

* Program : 14

-- Write a trigger that restricts the entry of record if Salary is greater than Rs. 50000.

```
Create or Replace Trigger trg_restrict_high_
Salary Before Insert or update ON EMP
FOR EACH ROW
Begin
    If :NEW.Sal > 50000 Then
        Raise_application_Error
        (-20001, 'Salary cannot be
            greater than Rs. 50000');
    End if;
End;
/
```

* Program : 15

-- Write a trigger that identifies the gender of the employee and according to the gender sets Mr. in front of MALE employee and Ms. in front of FEMALE employee.

```
Create or Replace Trigger trg_emp_
gender_prefix Before insert or update
ON EMP

FOR EACH ROW

Begin

    If UPPER (:NEW. Gender) = 'MALE' Then
        :NEW. Ename := 'Mr. ||
            Initcap (:NEW. Ename);
    Elsif UPPER (:NEW. Gender) = 'FEMALE' Then
        :NEW. Ename := 'Ms. ||
            Initcap (:NEW. Ename);
    Else
        :NEW. Ename := Initcap (:NEW.
            Ename);

    End if;

End;
/
```

* Program : 16

-- write a trigger to restrict user form using the table on Sunday.

```
Create or Replace Trigger trg_no_sunday_use
Before Insert or update or Delete on EMP

Begin

    If TO_CHAR (Sysdate, 'DY') =
      'Sun' Then

        Raise_Application_Error (-20002,
          'operation not allowed on
            Sunday.');

    End if ;

End ;
/
```