Ex.No.: 11		PL SQL PROGRAMS
Date:	24/09/2024	

PROGRAMS

TO DISPLAY HELLO MESSAGE

```
SQL> set serveroutput on;
SQL> declare
  2  a varchar2(20);
  3  begin
  4  a:='Hello';
  5  dbms_output.put_line(a);
  6  end;
  7  /
Hello

PL/SQL procedure successfully completed.
```

TO INPUT A VALUE FROM THE USER AND DISPLAY IT

```
SQL> set serveroutput on;
SQL> declare
  2  a varchar2(20);
  3  begin
  4  a:=&a;
  5  dbms_output.put_line(a);
  6  end;
  7  /
Enter value for a: 5
old   4: a:=&a;
new   4: a:=5;
5

PL/SQL procedure successfully completed.
```

GREATEST OF TWO NUMBERS

```
SQL> set serveroutput on;

SQL> declare
  2  a number(7);
```

```
3  b number(7);
4  begin
5  a:=&a;
6  b:=&b;
7  if(a>b) then
8  dbms_output.put_line (' The grerater of the two is'|| a);
9  else
10  dbms_output.put_line (' The grerater of the two is'|| b);
11  end if;
12  end;
13  /
Enter value for a: 5
old   5: a:=&a;
new   5: a:=5;
Enter value for b: 9
old   6: b:=&b;
new   6: b:=9;
The grerater of the two is9

PL/SQL procedure successfully completed.
```

GREATEST OF THREE NUMBERS

```
SQL> set serveroutput on;

SQL> declare
  2  a number(7);
  3  b number(7);
  4  c number(7);
  5  begin
  6  a:=&a;
  7  b:=&b;
  8  c:=&c;
  9  if(a>b and a>c) then
 10  dbms_output.put_line (' The greatest of the three is ' || a);
 11  else if (b>c) then
 12  dbms_output.put_line (' The greatest of the three is ' || b);
 13  else
 14  dbms_output.put_line (' The greatest of the three is ' || c);
 15  end if;
 16  end if;
 17  end;
 18  /
Enter value for a: 5
old   6: a:=&a;
new   6: a:=5;
```

```
Enter value for b: 7
old   7: b:=&b;
new   7: b:=7;
Enter value for c: 1
old   8: c:=&c;
new   8: c:=1;
The greatest of the three is 7

PL/SQL procedure successfully completed.
```

PRINT NUMBERS FROM 1 TO 5 USING SIMPLE LOOP

```
SQL> set serveroutput on;

SQL> declare
  2  a number:=1;
  3  begin
  4  loop
  5  dbms_output.put_line (a);
  6  a:=a+1;
  7  exit when a>5;
  8  end loop;
  9  end;
 10  /
1
2
3
4
5

PL/SQL procedure successfully completed.
```

PRINT NUMBERS FROM 1 TO 4 USING WHILE LOOP

```
SQL> set serveroutput on;

SQL> declare
  2  a number:=1;
  3  begin
  4  while(a<5)
  5  loop
  6  dbms_output.put_line (a);
  7  a:=a+1;
  8  end loop;
```

```
 9 end;
10 /
1
2
3
4
```
PL/SQL procedure successfully completed.

PRINT NUMBERS FROM 1 TO 5 USING FOR LOOP

```
SQL> set serveroutput on;

SQL> declare
 2 a number:=1;
 3 begin
 4 for a in 1..5
 5 loop
 6 dbms_output.put_line (a);
 7 end loop;
 8 end;
 9 /
1
2
3
4
5
```
PL/SQL procedure successfully completed.

PRINT NUMBERS FROM 1 TO 5 IN REVERSE ORDER USING FOR LOOP

```
SQL> set serveroutput on;
SQL> declare
 2 a number:=1;
 3 begin
 4 for a in reverse 1..5
 5 loop
 6 dbms_output.put_line (a);
 7 end loop;
 8 end;
 9 /
5
4
3
2
1
```
PL/SQL procedure successfully completed.

TO CALCULATE AREA OF CIRCLE

```
SQL> set serveroutput on;
SQL> declare
 2 pi constant number(4,2):=3.14;
```

```
 3  a number(20);
 4  r number(20);
 5  begin
 6  r:=&r;
 7  a:= pi* power(r,2);
 8  dbms_output.put_line (' The area of circle is ' || a);
 9  end;
10  /
Enter value for r: 2
old   6: r:=&r;
new   6: r:=2;
The area of circle is 13
PL/SQL procedure successfully completed.
```

TO CREATE SACCOUNT TABLE

```
SQL> create table saccount  ( accno number(5), name varchar2(20), bal number(10));
Table created.
SQL> insert into saccount values ( 1,'mala',20000);
1 row created.
SQL> insert into saccount values (2,'kala',30000);
1 row created.
SQL> select * from saccount;

    ACCNO NAME                 BAL
--------- -------------------- --------
        1 mala                 20000
        2 kala                 30000
SQL> set serveroutput on;
SQL> declare
 2  a_bal number(7);
 3  a_no varchar2(20);
 4  debit number(7):=2000;
 5  minamt number(7):=500;
 6  begin
 7  a_no:=&a_no;
 8  select bal into a_bal from saccount where accno= a_no;
 9  a_bal:= a_bal-debit;
10  if (a_bal > minamt) then
11  update  saccount  set bal=bal-debit where accno=a_no;
12  end if;
13  end;
14
15  /
Enter value for a_no: 1
old   7: a_no:=&a_no;
new   7: a_no:=1;
```

PL/SQL procedure successfully completed.

```
SQL> select * from saccount;
 ACCNO NAME                 BAL
--------- -------------------- ---------
     1 mala              18000
     2 kala              30000
```

TO CREATE TABLE SROUTES

```
SQL> create table sroutes ( rno number(5), origin varchar2(20), destination varchar2(20), fare
numbe
r(10), distance number(10));
Table created.
SQL> insert into sroutes values ( 2, 'chennai', 'dindugal', 400,230);
1 row created.
SQL> insert into sroutes values ( 3, 'chennai', 'madurai', 250,300);
1 row created.
SQL> insert into sroutes values ( 6, 'thanjavur', 'palani', 350,370);
1 row created.
SQL> select * from sroutes;

    RNO ORIGIN          DESTINATION          FARE  DISTANCE
-------- -------------------- -------------------- --------- ---------
       2 chennai              dindugal               400       230
       3 chennai              madurai                250       300
       6 thanjavur            palani                 350       370

SQL> set serveroutput on;

SQL> declare
  2  route sroutes.rno % type;
  3  fares sroutes.fare % type;
  4  dist sroutes.distance % type;
  5  begin
  6  route:=&route;
  7  select fare, distance into fares , dist from sroutes where rno=route;
  8  if (dist < 250) then
  9  update sroutes set fare=300 where rno=route;
 10  else if dist between 250 and 370 then
 11  update sroutes set fare=400 where rno=route;
 12  else if (dist > 400) then
 13  dbms_output.put_line('Sorry');
 14  end if;
 15  end if;
 16  end if;
 17  end;
 18  /
Enter value for route: 3
```

```
old  6: route:=&route;
new  6: route:=3;

PL/SQL procedure successfully completed.

SQL> select * from sroutes;

     RNO ORIGIN          DESTINATION          FARE  DISTANCE
-------- --------------- -------------------- --------- ---------
       2 chennai         dindugal               400       230
       3 chennai         madurai                400       300
       6 thanjavur       palani                 350       370
```

TO CREATE SCA LCULATE TABLE

```
SQL> create table scalculate ( radius number(3), area number(5,2));
Table created.
SQL> desc scalculate;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 RADIUS                                             NUMBER(3)
 AREA                                               NUMBER(5,2)

SQL> set serveroutput on;

SQL> declare
  2  pi constant number(4,2):=3.14;
  3  area number(5,2);
  4  radius number(3);
  5  begin
  6  radius:=3;
  7  while (radius <=7)
  8  loop
  9  area:= pi* power(radius,2);
 10  insert into scalculate values (radius,area);
 11  radius:=radius+1;
 12  end loop;
 13  end;
 14  /

PL/SQL procedure successfully completed.

SQL> select * from scalculate;
   RADIUS     AREA
```

```
---------- ----------
         3      28.26
         4      50.24
         5       78.5
         6     113.04
         7     153.86
```

TO CALCULATE FACTORIAL OF A GIVEN NUMBER

```
SQL> set serveroutput on;
SQL> declare
  2  f number(4):=1;
  3  i number(4);
  4  begin
  5  i:=&i;
  6  while(i>=1)
  7  loop
  8  f:=f*i;
  9  i:=i-1;
 10  end loop;
 11  dbms_output.put_line('The value is ' || f);
 12  end;
 13  /
Enter value for i: 5
old   5: i:=&i;
new   5: i:=5;
The value is 120

PL/SQL procedure successfully completed.
```

```
--------- ---------
        3    28.26
        4    50.24
        5     78.5
        6   113.04
        7   153.86
```

TO CALCULATE FACTORIAL OF A GIVEN NUMBER

```
SQL> set serveroutput on;
SQL> declare
  2  f number(4):=1;
  3  i number(4);
  4  begin
  5  i:=&i;
  6  while(i>=1)
  7  loop
  8  f:=f*i;
  9  i:=i-1;
 10  end loop;
 11  dbms_output.put_line('The value is ' || f);
 12  end;
 13  /
Enter value for i: 5
old   5: i:=&i;
new   5: i:=5;
The value is 120

PL/SQL procedure successfully completed.
```

PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

```
DECLARE
  v_emp_id        employees.employee_id%TYPE := 100;
  v_salary        employees.salary%TYPE;
  v_incentive     NUMBER;
  v_incentive_pct CONSTANT NUMBER := 0.10
BEGIN
  SELECT salary into v_salary from employees where emp_id =
  v_emp_id;
  v_incentive := v_salary + v_incentive_pct;
  DBMS_OUTPUT.PUT_LINE ("Incentive for empID " || v_id || v_incentive);
END;
```

PROGRAM 2

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

```
DECLARE
   v_test_variable NUMBER := 100;
BEGIN
   Execute immediate 'create or replace function "myfunction" return
   number is begin return 1; end;'
Declare
     v_result number;
BEGIN
     v_result := myfunction;
     DBMS_OUTPUT.PUT_LINE ('RESULT' || v_result);
EXCEPTION
     when others then DBMS_OUTPUT.PUT_LINE ('Error:' || sqlerrm);
End;
     DBMS_OUTPUT.PUT_LINE ('value:' || v_test_variable);
```

PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122.
Sample table: employees

```
Declare
    v_emp_id      employees.emp_id%TYPE:=22;
    v_newsalary   employees.salary%TYPE;
    v_current_sal employees.salary%TYPE;
    v_adj         number:=800;
Begin
    select salary into v_current_sal from employees where emp_id=
    v_emp_id;
    v_newsal := v_current_sal + v_adj;
    UPDATE employees SET salary = v_newsalary where emp_id=
    v_emp_id;
    Commit;
    DBMS_OUTPUT.PUTLINE ("Salary updated!");
End;
```

PROGRAM 4

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

```
Create or replace procedure update_emp_status ( p_emp_id
IN employees.empid%TYPE) AS
v_sal     employees.salary%TYPE
v_dept_id employees.dept_id%TYPE;
BEGIN
select sal, dept_id into v_sal, v_depid from employees where
emp_id = p_emp_id;
if v_sal isnotnull and v_depid is not null then update
employees SET status = 'active' where empid = p_emp_id;
DBMS_OUTPUT.PUT_LINE ('status updated to Active')
ELSE
DBMS_OUTPUT.PUT_LINE ("status cannot be updated");
END IF;
END;
```

PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

```
DECLARE
    V_emp_name  emp.name%TYPE;
BEGIN
    FOR rec IN (select name from emp where name
     LIKE 'J_x' ESCAPE '\')
     LOOP
        DBMS_OUTPUT.PUTLINE (rec.name);
     END LOOP;
END;
```

PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.

```
DECLARE
    num1    NUMBER := 25;
    num2    NUMBER := 10;
    num_small  NUMBER;
    num_large  NUMBER;
BEGIN
    IF num1 < num2 THEN
            num_s := num1;
            num_l := num2;
    ELSE
            num_s := num2;
            num_l := num1;
    END IF;
    DBMS_OUTPUT.PUTLINE (num_s);
    DBMS_OUTPUT.PUTLINE (num_l);
    CREATE OR REPLACE cal_in (p_emp_id IN emp.empid%type,
    p_tar_achieve IN NUMBER) AS inc NUMBER; v_tar
    NUMBER := 1000;
    BEGIN
      IF p_tar_achieved >= v_target THEN v_incentive :=
          p_target_achieved * 0.10;
      ELSE
          v_incentive := 0;
```

PROGRAM 7

Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

```
END IF;
UPDATE emp SET inc = v_inc where emp_id = p_empid
IF SQL%ROWCOUNT > 0 THEN
    DBMS_OUTPUT.PUT_LINE ("Record updated");
ELSE
    DBMS_OUTPUT.PUT_LINE ("NO record updated");
END IF;
END;
```

PROGRAM 8

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

```
Create or replace procedure cal_inc (p_sdant IN
NUMBER) AS   v_inc_num;
BEGIN
   IF p_sal_amt >= 5000 THEN
          v_inc := p_sal_amt * 0.15;
   ELSE
        v_inc := p_sal_amt * 0.05;
   ENDIF;
   DBMS_OUTPUT.PUT_LINE (p_sal_amt || v_inc);
   DECLARE
       v_emp_count  NUMBER;
       v_vacancies  CONSTANT NUMBER := 45;
   BEGIN
       select count(*) into v_emp_count FROM employees
   where dept_id = 50;
```

PROGRAM 9

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

```
DBMS_OUTPUT.PUT_LINE ( V.emp_count);
IF V-emp_count < V-vacancies THEN
      DBMS_OUTPUT.PUT_LINE ('80 vacancies available');
ELSE
      DBMS_OUTPUT.PUT_LINE ('no vacancies');
END IF;
END;
```

PROGRAM 10

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

```
DECLARE
   V-dept_id  NUMBER := 50;
   V-emp_count  NUMBER;
   V-total_pos  NUMBER := 150;
   V-vac        NUMBER;
BEGIN
   select count(*) INTO V-emp_count FROM employees whose
   dept_id = V-dept_id;
   V-vacancies := V-total_pos - V-emp_count;
   DBMS_OUTPUT.PUT_LINE (V-emp_count);
   If V-vac > 0 Then
        DBMS_OUTPUT.PUT_LINE ("vacancies available");
   ELIE
        DBMS_OUTPUT.PUT_LINE ("It is full Housed");
   END IF;
   END;
```

PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees.

```
DECLARE
BEGIN
   FOR rec IN (Select emp_id, Firstname || ' ' || last_name as
   emp_name, jobtitle, hiredate, salary from employees
    Join jobs on employees.jobid = jobs.job_id)
LOOP
   DBMS_OUTPUT.PUT_LINE (rec.empid || rec.emp name
                         || rec.job_title || TOCHAR
   (rec.hire_date, 'DD-MON-YYYY') || rec.salary);
END LOOP;
END;
```

PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

```
DECLARE
BEGIN
   FOR rec IN (Select e.empid, e.fname || ' ' || e.lname
   as empname, deptname from employees e JOIN
   departments d ON e.dept_id = d.dept_id)
LOOP
     DBMS_OUTPUT.PUT_LINE (rec.empid || rec.emp_name ||
                           rec.dept_name),
END LOOP;
END;
```

PROGRAM 13
Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```
DECLARE
BEGIN
   FOR rec IN (SELECT job_id, job_title, min_sal FROM jobs)
   LOOP
      DBMS_OUTPUT.PUT_LINE(rec.job_id || rec.job_title ||
         rec.min_sal);
   END LOOP;
END;
```

PROGRAM 14
Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

```
DECLARE
BEGIN
   FOR rec IN (SELECT e.emp_id, e.first_name || ' ' || e.last_name AS
   emp_name, jh.start_date FROM employees e JOIN job_history jh
   ON e.emp_id = jh.emp_id)
   LOOP
      DBMS_OUTPUT.PUT_LINE(rec.emp_id, rec.emp_name, TO_CHAR
         (rec.start_date, 'DD-Mon-YYYY'));
   END LOOP;
   END;
```

15.

```
DECLARE
V_employee_id     employees_emp_id %.TYPE;
V-first-name      employee.firstname %.TYPE;
V-last-name       employee.lastname %.TYPE;
V-end-date        job history.end-date %.TYPE;

CURSOR emp_cur IS
  SELECT e.employee id, e.first-name, e.last-name, j.end-date
      from employee e JOIN job history j on e.employee_id=
        j.employee-id;

BEGIN
  FOR emp_record IN emp_cursor Loop
      v-employee_id := emp_record.employee_id;
      v-firstname := emp_record.first_name;
      v-last-name := emp_record.last_name;
      v-end-date := emp_record.end_date;

      DBMS_OUTPUT.PUT_LINE ('Employee ID:' || v-employee_id ||
      'name :' || v-firstname || ' ' || v-lastname ||, ' Job history
       End Date :' || TO_CHAR(v_end_date, 'YYYY-MM-DD'));

END LOOP;

END;
```

PROGRAM 15

Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	
Program/Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	