Ex.No.: 11		PL SQL PROGRAMS
Date:	22.10.2025	

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.
```

PROGRAM 1

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

```
DECLARE
    v-employee_id      employees.employee-id %TYPE :=110;
    v-salary           employees.employee . salary %TYPE;
    v-incentive        NUMBER (10,2);
BEGIN
    SELECT  salary
    INTO  v-salary
    FROM  employees  WHERE  employee_id = v_employee_id;
    v_incentive = v_salary *0.10;
    DBMS_OUTPUT_LINE ('INCENTIVE of employee ID'' || v_employee_id ||
                          'is:' || TO-CHAR (v-incentive, '99999.9')

EXCEPTION
    WHEN  NO-DATA-FOUND THEN
     DBMS_OUTPUT. PUT_LINE ('Employee with ID' || v_employee_id ||
                                                     'not found');
    WHEN  OTHERS THEN
     DBMS_OUTPUT. PUT_LINE ('An Error excured :'||  SQLERRM);
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_count  NUMER;
BEING
        SELECT  COUNT (*)  INTO  V_COUNT  FROM  mytable;
        DBMS_OUTPUT. PUTLINE ('count:" || v-count);
EXCEPTION
        WHEN  OTHERS  THEN
              DBMS_OUTPUT. PUTLINE ('Error: ' || SQLERRM);
END;
```

PROGRAM 3

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

```
BEGIN
        UPDATE  employees
        SET     salary = salary * 0.10
        WHERE   employeeid  = 122;
        COMMIT;
        DBMS_OUTPUT.PUT_LINE ('salary Updated  for employe
                                                      122');
EXCEPTION
        WHEN  OTHERS  THEN
                DBMS_OUTPUT.PUT_LINE ('Error :'||  SQLERM);
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  check_values (P_val1   IN  VA
                                           P_val2   IN  VARCHAR2
BEGIN
    IF   P_vall   is   NOT NULL  AND  P_val2   IS  NOT   NULL T
             DBMS_OUTPUT.PUT_LINE (' Both  values  are Not
    ELSE
            DBMS_OUTPUT.PUT_LINE ('At least  one  value
                                                    NULL');
    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_name     VARCHAR (50);
BEGIN
        v-name := 'Jonathan';
        IF  v-name  LIKE  'Jon %'  THEN
             DBMS_OUTPUT.PUT_LINE ('Match found  using %
                                                wildcard'');
        END IF;
        v_name:='John';
        If  v-name  LIKE  'John'.THEN
             DBMS_OUTPUT.PUT-LINE ('Match found using wildcard');
        END IF;
        IF  v-name  LIKE  '50 % % ESCAPE '\' THEN
                        DBMS-OUTPUT.PUT-LINE ('Match found using
                                          escabe character').
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 := &num1;
        num2   NUMBER := &num2;
        num_small NUMBER;
        num_Large NUMBER;
BEGIN

        IF   num<num2   THEN

              num_small  :=num1;
              num_large  := num2;
        ELSE
              num_small  := num2;
              num_small := num1;
        END IF;
        DBMS_OUTPUT. PUT_LINE ('small number :' || num_small)
        DBMS_OUTPUT. PUT_LINE ('Large Number :' || num_large);
END;
```

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.

```
CREATE  OR  REPLACE  PROCEDURE  calculate_incentive (
                P_target_achieved IN  NUMBER,
                P_incentive  OUT  NUMBER
)AS
 BEGIN
      IF   P_target_achieved  > 1000  THEN
           P_incentive :=  P_target_achieved * 0.10;
           DBMS.OUTPUT.PUT_LINE ('Record  Not  updated');
      ELSE
           DBMS_OUTPUT_PUTLINE (' Record  updated');
END  calculate_incentive;
```

PROGRAM 8

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

```
CREATE  OR  REPLACE  PROCEDURE  calculate_incenentive (
                P_scale_achieved  IN  NUMBER
)AS
    v_incentive  NUMBER := 0;
BEGIN
     IF   P_scale_achieved  > 44000  THEN
               v_incentive  := 1800;
     ELSE IF  P_scale_achieved  > 32000  THEN
               v_incentive  :=800;
     ELSE
             v_incentive  :=500;
     ENDIF ;
     DBMS_OUTPUT_PUT_LINE ('scale Achieved :'. || v_incentive);
END;
```

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.

```
DECLARE
        emp-count  NUMBER;
        vacancies NUMBER;
BEGIN
        SELECT  COUNT (*) INTO  emp-count FROM  employees whe
                                              department-id = 50;

        vacancies := 45 - emp-count;
        DBMS-OUTPUT. PUT-LINE  ('Employees in Dept 50:' || emp-count);
        DBMS-OUTPUT . PUT- LINE ('vacancies in Dept 50:' || vacancies).
        IF  vacancies > 0  THEN
              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
      dept-id   NUMBER  := & dept-id;
      emp-count   NUMBER;
      vacancies  NUMBER;
BEGIN
      SELECT   COUNT(*) INTO   emp-count  FROM employees
      WHERE    department-id  = dept-id;
      Vacancies  := 45 - emp-count;
      DBMS-OUTPUT . PUT-LINE ('Employees in Dept'|| dept-id );
      DBMS -OUTPUT.PUT-LINE ('vacancies' ||dept- id || vacancies)
      IF    vacancies > 0  THEN
          DBMS-OUTPUT. PUT-LINE ('Department has vacancies' );
      ELSE
          DBMS-OUTPUT . PUT-LINE ('NO vacancies');
      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.

```
BEGIN
    FOR emp_rec IN (SELECT employee_id , first_name,
            last_name ,job_id , hire_date ,salary FROM employees)
     Loop
        DBMS_OUTPUT . PUT_LINE (emp_rec . employee _id || '|' ||
        emp_rec . firstname || '|' || emp_rec . last _name ||
           '|'   || emp_rec .job_id || '|' ||
        to CHAR (emp_rec . hiredate , 'DD - MON - YYYY')||
           '|'   || emp. rec .salary
            ;.
END ;     END LOOP
```

PROGRAM 12

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

```
BEGIN
    FOR   emp_rec   IN (SELECT e.employee_id . e . first _name ||
             ' '  || e.last_name AS full_name, d.dept_name
              FROM   employees   e
              JOIN   departments   d
              ON    e.dept_id   = d.dept_id
              LOOP
                DBMS_OUTPUT.PUT_LINE (emp _ree .employee_id  ||
                                      emp_ree .full _name ||
                                      emp 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.

```
SET  SERVEROUTPUT  ON;
BEGIN
    FOR  Job_rec  IN (SELECT  job_id , job_title,
                          min_salary  FROM  jobs)
LOOP
    DBMS_OUTPUT.PUT_LINE ('Job ID: ' || job_rec.job_id ||
                    'job Title : ' || Job_rec.job.id ||'min sal:
                         ' || job_rec .min_salary );
    END LOOP;
END;
```

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

```
SET  SERVEROUTPUT  ON;
BEGIN
    FOR  emp_rec  IN ( SELECT  e.employee_id , e.first_name
   || ' ' || e.last_name   AS employee_name, jh.strat_date
   FROM  employees  e
   JOIN  job_history jh  ON  e.employee_id = jh.employee_id
   ORDER  BY  e.employee_id
   Loop
       DBMS_OUTPUT.PUT_LINE ('Employee ID: ' || emp_rec.
                                      employees_id
   ||'Name :' || emp_rec. employee_name ||'job:' || temp_rec ||
   TO_CHAR  (emp_rec.strart_date   , 'DD-MON-YYYY')).
         END LOOP;
END;
```

PROGRAM 15

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

```
SET  SERVEROUTPUT ON,
  BEGIN
    FOR emp_rec  IN (
            SELECT  e.employee_id, e.first_name || ' ' ||
    e.last_name  AS employee_name  , jh.end_date
          FROM employees  e
          JOIN  job_history jh  ON emp_id = jh.emp_id
          ORDER BY e.emp_id
          )LOOP
            DBMS_OUTPUT.PUT_LINE ('Employee. ID:' ||emp_rec.
                                                  emp_id
    || 'Name :' || emp_rec . emp_Name ||' Job END Date '||
    To CHAR (emp_rec. end_date , 'DD -MON- YYYY'));
            END LOOP;
END;
/
```

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

[illegible]

RESULT :

thus all the given programs have been executed .