Ex.No.: 11		PL SQL PROGRAMS
Date:	24/9/2024	

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.
```

PROGRAM 1

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

PROGRAM 2

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

```
SET SERVEROUTPUT ON;
DECLARE
   employee VARCHAR (50):= 'John Doe';
  "Employee" VARCHAR (50):= 'Jane Doe';
BEGIN
    DBMS_OUTPUT.PUT_LINE('case-Insentive (employee Name'): '||
DBMS_OUTPUT. PUT_LINE ('case_sensentive ("employee Name"): '||
EXCEPTION
    DBMS_OUTPUT.PUT_LINE ('Error: '||SQLERRM);
END;
```

Program 1:

```
DECLARE
emp_id employees.emp_id %TYPE := 110;
emp_name employees.name %TYPE;
emp_salary employees.salary %TYPE;
incentive NUMBER(7,2);
BEGIN
SELECT name, salary
INTO emp_name, emp_salary
FROM employees
WHERE emp_id = 110;
incentive := emp_salary * 0.10;
DBMS_OUTPUT.PUT_LINE('Employee Name : ' || emp_name);
DBMS_OUTPUT.PUT_LINE('Employee Salary : ' || emp_salary);
DBMS_OUTPUT.PUT_LINE('Incentive (10%) : ' || incentive);
EXCEPTION
 WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Employee with ID 110 not found');
 WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

PROGRAM 3

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

```
SET SERVER OUTPUT ON;
BEGIN
   UPDATE employees
   SET salary = salary + (salary * 0.10)
   WHERE emp_id = 12
   RETURNING salary INTO :new_salary;
   DBMS_OUTPUT.PUT_LINE ('New salary: '|| :new_salary);
   EXCEPTION
       WHEN NO_DATA_FOUND THEN
          DBMS_OUTPUT.PUT_LINE ('Employee with ID122 not found:');
       WHEN OTHERS THEN
          DBMS_OUTPUT.PUT_LINE ('Error:' || SQLERRM);
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.

```
SET SERVEROUTPUT.ON;
BEGIN
   IF('Hello' IS NOT NULL AND NULL IS NOT NULL) THEN
     DBMS_OUTPUT.PUT_LINE ('Both are not NULL');
   ELSE
     DBMS_OUTPUT.PUT_LINE ('Atlast One is NULL :);
   END IF;
END;
```

OUTPUT: Atleast one is NULL.

PROGRAM 5

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

OUTPUT:
Pattern 1 matched
Pattern 2 matched
Pattern 3 matched with escape

```
SET SERVEROUTPUT ON;
BEGIN
  IF 'Hello World' LIKE 'H%W%' THEN
  DBMS_OUTPUT.PUT_LINE ('Pattern 1 matched.');
  END IF;
  IF 'Hello 123' LIKE 'Hello_23' THEN
   DBMS_OUTPUT.PUT_LINE('Pattern 2 matched.');
  END IF;
  IF '50% discount' LIKE '50\%%' ESCAPE '\' THEN
   DBMS_OUTPUT.PUT_LINE('Pattern 3 matched with escape.');
```

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.

```
SET SERVEROUTPUT ON;
DECLARE
    num1 NUMBER := 10;
    num2 NUMBER := 20;
    num_small NUMBER := LEAST (num1, num2);
    num_large NUMBER := GREATEST (num1, num2);
BEGIN
  DBMS_OUTPUT.PUT_LINE ('Small: ' || num_small || ; Large: ' ||
END;
```

OUTPUT : Small : 10,
Large : 20

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.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE calc_incentive (emp_id_IN number) IS
BEGIN
UPDATE employees SET incentive = target_achieved * 0.10 WHERE
emp_id = emp_id AND TARGET
DBMS_OUTPUT.PUT_LINE ('Record' || CASE WHEN SQL%. ROW count 70
THEN ' Updated.'
ELSE 'not updated.' END);
END;
```

PROGRAM 8
Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE calc_incentive(emp_id IN NUMBER) IS
Sales_limit_ NUMBER = 1000;
  incentive
BEGIN
  SELECT CASE WHEN total_sales >= Sales_limit THEN total_sales* 0.10
UPDATE employees SET incentive= incentive_amount WHERE emp_id = emp_id;
DBMS_OUTPUT. PUT_LINE ('Incentive for ID' || emp_id || ':' || incentive_amount);
EXCEPTION
  WHEN NO_DATA_FOUND THEN DBMS_OUTPUT. PUT_LINE ('Employee not found');
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.

```
SET SERVEROUTPUT ON';
DECLARE
   emp_count NUMBER ;
BEGIN
  SELECT COUNT (*)INTO emp_count FROM employees WHERE department_id = 50';
DBMS_OUTPUT.PUT_LINE ('employees in DEPT 50:' || emp_count) ;
DBMS_OUTPUT.PUT_LINE (IIF( emp_count < 45 , 'Vacancies available.',)) ;
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.

```
SET SERVEROUTPUT ON';
DECLARE
  emp_count NUMBER ;
  Vacancies NUMBER := 45';
BEGIN
  SELECT COUNT (*) INTO emp_count FROM employees WHERE department=50,
DBMS_OUTPUT.PUT_LINE ('Employees in DEPT 50:' || emp_count || ' ,vacancies ||
(vacancies- emp_count)) ;
  END',
```

PROGRAM 11

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

```
SET SERVEROUTPUT ON;
BEGIN
FOR rec IN (SELECT employee_id, name, job_title, hire_date, salary FROM
  DBMS_OUTPUT.PUT_LINE('ID: ' || rec.employee_id ||
                ', Name: ' || rec.name ||
                ', Job title: ' || rec.job_title ||
                ', Hire Date: ' || rec.hire_date ||
                ', Salary: ' || rec.salary);
END LOOP;
END;
```

PROGRAM 12

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

```
SET SERVEROUTPUT ON;
BEGIN
FOR rec IN (SELECT e.employee_id, e.name, d.department_name
          FROM employees e
          JOIN departments d ON e.department_id = d.department
DBMS_OUTPUT.PUT_LINE ('ID: ' || rec.employees_id ||
                      ', Name: ' || rec.name ||
                      ', Department: ' || rec.department_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 rec IN (SELECT job_id , job_title, min_salary FROM) LOOP
DBMS_OUTPUT.PUT_LINE ('JobID :' || rec.job_id ||
                      ', Title :' || rec.job_title ||
                      ', min salary :' || 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 rec(SELECT e.employee_id, e.name, j.start_date
        FROM employees e
        JOIN job_history j ON e.employee_id=j.employee_id
DBMS_OUTPUT.PUT_LINE ('ID :' || rec.employee_id ||
                      ', Name :' || rec.name ||
                      ', Job start Date :' || rec.start_date);
  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 rec IN (SELECT e.employee_id, e.name, j.end_date
        FROM employees e
        JOIN job_history j ON e.employee_id = j.employee_id)
DBMS_OUTPUT.PUT_LINE ('ID:' || rec.employee_id ||
                      ', Name :' || rec.name ||
                      ', Job End Date :' || rec.end_date);
  END LOOP';
END';
```

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