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);
```

Program 1:-

```
DECLARE
emp_id employees.emp_id %TYPE:=110;
emp_home employees.name %TYPE;
emp_Salary employee.salary %TYPE;
incentive NUMBER(7,2);

BEGIN
SELECT name, Salary
INTO emp_home, 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:'||
END; SQLERRM);
```

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 - Insertive
("employee Name");'|| DBMS_OUTPUT.PUT_LINE
('Case _Sensitive ("employee Name");'||
     EXCEPTION
          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:'||;
newSalary)';
EXCEPTION
        WHEN NO-DATA-FOUND THEN
            DBMS-OUTPUT.PUT-LINE('Employee with
ID 122 not found.')';
         WHEN OTHERS - THEN
END;        DBMS-OUTPUT. PUT-LINE('Error:'||SQLERRM)';
```

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('Atleast 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.

```
SET SERVEROUTPUT ON;
BEGIN
      IF 'Helloworld' LIKE 'H%W%' THEN
      DBMS_OUTPUT.PUT_LINE('Pattern1 matched.');
      END IF;
   IF 'Hello123' 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;
/
```

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
(empid IN number) IS
 BEGIN
UPDATE employees SET incentive=target achieved*0.10
WHERE emp_id =emp_id ANDD TARGET DBMS_
OUTPUT_PUT_LINE ('Reward'|| CASE WHEN SQL% ROW
COUNT > 0 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 = empid; 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-OUTPUTPUT-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 SERVER OUT PUT 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  SERVER OUTPUT  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 SERVER OUTPUT ON;
 BEGIN
 FOR rec IN(SELECT e.employee_id e.name, d.
department_name FROM employee JOIN departments
d ON e.department_id = d.department DBMS_OUTPUT.
PUT_LINE ('ID: '|| rec.employee_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   SERVER   OUTPUT. ON;
BEGIN
        FOR  rec  IN(SELECT Job - id, job_title, min_
Salary FROM) LOOP
        DBMS_OUTPUT. PUT_LINE('Job ID:') || 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 SERVER OUTPUT  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)	4
Total (15)	14
Faculty Signature	[illegible]

Scanned with OKEN Scanner