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

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_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 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-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 SERVEROUTPUT 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 ID 122 not
      WHEN OTHERS_THEN                                    found:);
    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 ('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.

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 'Hell 0_23' THEN
    DBMS_OUTPUT. PUT_LINE ('Pattern 2 matched.');
     END IF;
IF '50%. discount' LIKE '50\%.%.' ESCAPE '\' THEN
   DBMS_OUTPUT. PUTLINE ('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 SERVER OUTPUT ON;
DECLARE
      num 1 NUMBER := 10;
      num 2 NUMBER := 20;
   num_small NUMBER := LEAST (num 1, num 2);
   num_large NUMBER := GREATEST (num 1, num 2);
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 SERVER OUTPUT 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.PUTLINE ('Record' || CASE WHEN SQL %. ROW COUNT > 0 THEN
   ELSE 'not updated.' END);                              'Updated'.
   END;
```

PROGRAM 8

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

```
SET SERVER OUTPUT 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.PUTLINE ('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.PUTLINE('employees in DEPT 50:' || emp_count);
DBMS_OUTPUT.PUTLINE(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.emp-wunt');                    vacancies||
  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 SERVER OUTPUT 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.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 SERVER OUTPUT 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_OUT.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	