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 :=110;
        v_salary.
employees.salary%.TYPE
        v_incentive   NUMBER;
BEGIN
    SELECT salary
    INTO v_salary
    FROM employees
    WHERE employees_id = v_emp_id;
    v_incentive := v_salary * 0.10;
    DBMS_OUTPUT.PUT_LINE ('Employee ID: ' || v_emp_id);
    DBMS_OUTPUT.PUT_LINE ('Salary: ' || v_salary);
    DBMS_OUTPUT.PUTLINE ('Incentive: ' || v_incentive);

EXCEPTION
        WHEN  NO_DATA_FOUND THEN
              DBMS_OUTPUT.PUT_LINE ('No employee found with
                                     ID' || v_emp_id);
        WHEN  OTHERSTHEN
              DBMS_OUTPUT.PUTLINE ('An error occured: ' ||
                                    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.

```
SET SERVEROUTPUT ON;

DECLARE
    "Name" VARCHAR2(20) := 'MESSI';

BEGIN
    DBMS_OUTPUT.PUT_LINE (name);

    DBMS_OUTPUT.PUT_LINE ('Name');

END;
```

PROGRAM 3

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

Sample table: employees

```
DECLARE
        V_old_salary
employees.salary%.TYPE;
        V_new_salary
employees.salary%.TYPE;
BEGIN
    SELECT salary INTO V_old_salary
    FROM employees
    WHERE employee_id =122;

    V_new_salary:= V_old_salary *1.10;

    UPDATE employees
    SET salary =V_new_salary
    WHERE employee_id =122;
    DBMS_OUTPUT.PUT_LINE('salary updated for Employee ID 122.');
    DBMS_OUTPUT.PUT_LINE('Old salary:'|| V_old_salary);
    DBMS_OUTPUT.PUT_LINE('New salary:' || V_new_salary);
    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No employee found with ID 122.');
    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;
DECLARE
    PROCEDURE check_null_and_condition (P_num1 NUMBER, P_num2 NUMBER)
    BEGIN
        DBMS_OUTPUT.PUT_LINE ('Input values: P_num1 = ' ||
        NVL(TO_CHAR(P_num1), 'NULL') || ', P_num2 = ' || NVL(TO_CHAR
                                    (P_num2), 'NULL'));
    IF(P_num1 IS NOT NULL) AND (P_num2 IS NOT NULL) THEN
        DBMS_OUTPUT.PUT_LINE('Both operands are NOT NULL
                                    -> AND condition is TRUE');
    ELSE
        DBMS_OUTPUT.PUT_LINE ('One or both operands are NULL
                                    -> AND condition is FALSE');
    END IF;
END;
BEGIN
    check_null_and_condition (10,20);
    check_null_and_condition (NULL,15);
    check_null_and_condition (25, NULL);
    check_null_and_condition (NULL, NULL);
END;
/
```

PROGRAM 5

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

```
SET SERVEROUTPUT ON;

DECLARE
    Vname. VARCHAR(30);
BEGIN
    Vname := 'SMITH';
    IF Vname LIKE 'S%' THEN
        DBMS_OUTPUT.PUT_LINE ('Pattern "S%" matched : Name starts
                                                    with S');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Pattern "S%" did not match');
    END IF;

  IF Vname LIKE 'S_I%' THEN
     DBMS_OUTPUT.PUT_LINE ('Pattern 'S_I%' matched: Second letter
                                can be any single character');
  ELSE
     DBMS_OUTPUT.PUT_LINE ('Pattern 'S_I%' did not match');
  END IF;

  Vname := 'JOHN_DOE';
  IF Vname LIKE 'JOHN\_DOE' ESCAPE '\' THEN
     DBMS_OUTPUT.PUT_LINE ('Pattern 'JOHN_DOE' matched :
                                Actual underscore matched using ESCAPE');
  ELSE
     DBMS_OUTPUT.PUT_LINE ('Pattern "JOHN_DOE" did not match');
  END IF;

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.

```
SET SERVEROUTPUT ON;

DECLARE
    num1 NUMBER := 25;
    num2 NUMBER := 10;
    num_small NUMBER;
    num_large NUMBER;

BEGIN
    IF num1 < num2 THEN
        num_small := num1;
        num_large := num2;
    ELSE
        num_small := num2;
        num_large := num1;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Smaller 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.

```
SET SERVEROUTPUT ON;
DECLARE
    emp_id NUMBER := 122;
    target NUMBER := 75000;
    incentive NUMBER;
    rows_updated NUMBER;
BEGIN
    IF Target >= 100000 THEN
        incentive := 5000;
    ELSIF target >= 50000 THEN
        incentive := 2000;
    ELSE
        incentive := 0;
    END IF;
    UPDATE employee
    SET incentive = incentive
    WHERE employee_id = emp_id;
    rows_updated := SQL%ROWCOUNT;
    IF rowsupdated > 0 THEN.
```

PROGRAM 8

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

```
SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE
calc_incentive (
    p_emp_id IN NUMBER,
    p_sales IN NUMBER.
)IS
    v_incentive NUMBER;
BEGIN.
    IF p_sales >=100000 THEN
        v_incentive :=5000;
    ELSIF p_sales >=50000 THEN
        v_incentive := 2000;
    ELSE
        v_incentive := 0;
    END IF;

    UPDATE employee
    SET incentive = v_incentive
    WHERE employee_id = p_emp_id;

    IF SQL%. ROWCOUNT > 0 THEN.
DBMS_OUTPUT. PUT_LINE ('Record updated. Incentive = ' || v_incentive
ELSE.
DBMS_OUTPUT. PUT_LINE ('No Record found for employee ID')
```

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
    v_emp_count NUMBER;
    v_vacancies NUMBER := 45;
    v_remaining NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_emp_count
    FROM employee
    WHERE department_id = 50;
    v_remaining := v_vacancies - v_emp_count;
    DBMS_OUTPUT.PUT_LINE('Number of employees in Dept 50: ');
    IF v_remaining > 0 THEN
DBMS_OUTPUT.PUT_LINE('Vacancies available:' || v_remaining);
ELSE
    DBMS_OUTPUT.PUT_LINE('No vacancies in Dept 50.');
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.

```
SET SERVEROUTPUT ON;
DECLARE
    V_dept_id    NUMBER: =&dept_id;
    V_emp_count  NUMBER;
    V_total_post NUMBER:. =45;
    V_vacancies  NUMBER;
BEGIN.
    SELECT COUNT(*) INTO V_emp_count.
    FROM employee
    WHERE department_id = V_dept_id;
    V_ vacancies ':. = v_total_posts -V_emp_count;
DBMS_OUTPUT. PUT_LINE ('Department ID:' || V.dept_id);
DBMS_OUTPUT.PUT_LINE ('Number of Employees:' || v_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;

DECLARE
    v_emp_id
employee.employee_id%TYPE;
    v_name
employee.first_name%TYPE;
    v_job
employee.job_id%TYPE;
    v_hire_date
employee.hire_date%TYPE;
    v_salary
employee.salary%TYPE;

CURSOR emp_cur IS
        SELECT employee_id, first_date, job_id, hire_date
salary
        FROM employee;

BEGIN
    DBMS_OUTPUT.PUT_LINE('EMP_ID NAME JOB_ID HIRE_DATE');

END;
```

PROGRAM 12

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

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR emp_cur IS
        SELECT e.employee_id,
               e.first_name,
               d.department_name
        FROM employee e
        JOIN department d
          ON e.department_id = d.department_id;

BEGIN
    DBMS_OUTPUT.PUT_LINE('EMP_ID  NAME  DEPT');
    DBMS_OUTPUT.PUT_LINE('----------------');

  FOR emp_rec IN emp_cur LOOP
      DBMS_OUTPUT.PUT_LINE(emp_rec.employee_id || ' ' ||
      RPAD(emp_rec.first_name, 12) || ' ' ||
      emp_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;
DECLARE
    CURSOR job_cur IS
      SELECT JobId, job_title, min_salary
      FROM Jobs;
BEGIN
    DBMS_OUTPUT.PUT_LINE ('JOB_ID  JOB_TITLE  MIN_SALARY');
DBMS_OUTPUT.PUT_LINE ('---------- ---------- ----------');

FOR job_rec IN job_cur LOOP
    DBMS_OUTPUT.PUT_LINE (RPAD(job_rec.job_id, 10)
    || ' ' || RPAD (job_rec.job_title, 25) || ' ' ||
     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;
DECLARE
    CURSOR emp_cur IS
        SELECT e.employee_id,
               e.first_name
               jh.start_date
        FROM employee e
        JOIN job_history jh.
        ON e.employee_id = jh.employee_id;
BEGIN
    DBMS_OUTPUT.PUT_LINE ('EMP_ID  NAME  START_DATE');
    DBMS_OUTPUT.PUT_LINE ('- - - - - - - - - - - - -');
    FOR emp_rec. IN emp_cur LOOP
        DBMS_OUTPUT.PUT_LINE (emp_rec.employee_id || '  ' ||
        RPAD(emp_rec.first_name, 10) || '  ' ||
        TOCHAR(emp_rec, start_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;
DECLARE
    CURSOR emp_cur IS
        SELECT e.employee_id;
               e.first_name
               jh.end_date
        FROM employee e
        JOIN job_history jh
ON e.employee_id =
        jh.employee_id;
BEGIN
  DBMS_OUTPUT.PUT_LINE
('EMP_ID   NAME
          END_DATE').

FOR emp_rec IN emp_cur
 LOOP DBMS_OUTPUT.PUT_LINE
(emp_rec.employee_id ||'   '||
 RPAD (emp_rec.first_name)||'  '||
  TO_CHAR(emp_rec.end_date, 'DD-MON-YYYY'));
  END LOOP;
END;
```

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