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 employee_id = v_emp_id;

    IF v_salary >= 10000 THEN
        v_incentive := v_salary * 0.10;
    ELSE
        v_incentive := v_salary * 0.05;
    END IF;

    DBMS_OUTPUT.PUT_LINE ('Employee ID: ' || v_emp_id);
    DBMS_OUTPUT.PUT_LINE ('Salary: ' || v_salary);
    DBMS_OUTPUT.PUT_LINE ('Incentive: ' || v_incentive);
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
    "MyVariable" NUMBER := 100;
BEGIN
    DBMS_OUTPUT.PUT_LINE(MyVariable); -- Error
END;
/
```

Corrected :

```
    DBMS_OUTPUT.PUT_LINE("MyVariable");
```

PROGRAM 3

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

Sample table: employees

```
DECLARE
    v_emp_id   employees.employee_id%TYPE := 122;
    v_salary   employees.salary%TYPE;
BEGIN
    SELECT salary INTO v_salary FROM employees WHERE
                                    employee_id = v_emp_id;
    v_salary := v_salary + (v_salary * 0.10);
    UPDATE employees
    SET salary = v_salary
    WHERE employee_id = v_emp_id;
    DBMS_OUTPUT.PUT_LINE ('Salary updated for employee ' || v_emp_id);
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.

```
DECLARE
    v_bonus NUMBER := NULL;
    v_salary NUMBER := 5000;
BEGIN
    IF v_bonus IS NULL THEN
        DBMS_OUTPUT.PUT_LINE ('Bonus is NULL');
    END IF;
    IF (v_salary > 3000) AND (v_salary < 6000) THEN
        DBMS_OUTPUT.PUT_LINE ('Condition TRUE: Salary between
                                3000 and 6000');
    ELSE
        DBMS_OUTPUT.PUT_LINE ('Condition FALSE');
    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  employees.last_name%TYPE;
BEGIN
    FOR rec IN (
        SELECT last_name FROM employees
        WHERE last_name LIKE 'S%' ESCAPE '\'
    ) LOOP
    DBMS_OUTPUT.PUT_LINE ('Name starting with S: ' || rec.last_name);
    END LOOP;
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 :=10;
    num1 NUMBER :=20;
    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 ('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 calc_incentive (p_emp_id NUMBER) IS
    v_salary employees.salary%TYPE;
    v_incentive NUMBER;
BEGIN
    SELECT salary INTO v_salary FROM employees WHERE employee_id
                                                = p_emp_id;
    v_incentive := v_salary * 0.10;
    UPDATE employees
    SET commission_pct = NVL(commission_pct,0) + v_incentive/100
    WHERE employee_id = p_emp_id;
    IF SQL%ROWCOUNT > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Record Updated Successfully');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No Record Found');
    END IF;
END;
/
BEGIN
    Calc_incentive (110);
END;
/
```

PROGRAM 8

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

```
CREATE OR REPLACE PROCEDURE sale_inceptive (p_sales NUMBER) IS
    v_inceptive NUMBER;
BEGIN
    IF p_sales >= 100000 THEN
        v_incentive := 0.20;
    ELSIF p_sales >= 50000 THEN
        v_incentive := 0.10;
    ELSE
        v_incentive := 0.05;
    END IF;
    DBMS_OUTPUT.PUT_LINE ('Incentive Rate : ' || (v_incentive * 100) || '%.')

END;
/
BEGIN
    sale_incentive (60000);
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
    V_count NUMBER;
    V_VACANCIES CONSTANT NUMBER :=45;
BEGIN
    SELECT COUNT(*) INTO v_count FROM employees WHERE
    DBMS_OUTPUT.PUT_LINE('Employees in Dept 50: '|| department_id = 50;
                                                  v_count);
    IF v_count <V_vacancies THEN
        DBMS_OUTPUT.PUT_LINE ('Vacancies available : '|| (V_vacancies
                                                        - v_count));
    ELSE
        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
   v_dept NUMBER := &dept_id;
   v_count NUMBER;
   v_vacancies NUMBER := 45;
BEGIN
   SELECT COUNT(*) INTO v_count FROM employees WHERE department_id
                                                = v_dept;
   IF v_count < v_vacancies THEN
      DBMS_OUTPUT.PUT_LINE('Dept ' || v_dept || ' has ' || (v_vacancies - v_count)
                                                || 'vacancies');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Dept ' || v_dept || ' has 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
                    AS full_name, job_id, hire_date, salary
                    FROM employees)
    LOOP
        DBMS_OUTPUT.PUT_LINE ('ID: ' || emp_rec.employee_id ||
                              ', Name: ' || emp_rec.full_name ||
                              ', Job: ' || emp_rec.job_id ||
                              ', Hire Date: ' || emp_rec.hire_date ||
                              ', Salary: ' || emp_rec.salary);
    END LOOP;
END;
/
```

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_nam
                  AS full_name, d.department_name
                  FROM employees e
                  JOIN departments d ON e.department_id =
                                        d.department_id

  LOOP
    DBMS_OUTPUT.LINE ('ID: ' || emp_rec.employee_id ||
                      ', Name:' || emp_rec.full_name ||
                      ', Department:' || 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.

```
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 ||
                          ', Title : ' || job_rec.job_title ||
                          ', Min salary: ' || job_rec.job_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.

```
BEGIN
   FOR hist_rec IN (SELECT e.employee_id , e.first_name || ' ' || e.last
                           _name AS full_name, h.start_date
                    FROM employees e
                    JOIN job_history h ON e.employee_id=h.employee_id)
   LOOP
      DBMS_OUTPUT.PUT_LINE('ID :' || hist_rec.employee_id ||
                           ', Name :' || hist_rec.full_name ||
                           ', Job History Start :' || hist_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.

```
BEGIN
    FOR list_rec IN (SELECT e.employee_id, e.first_name
                     FROM employees e
                     JOIN job_history h ON e.employee_id
                     = h.employee_id)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Employee Id' || employee_rec.employee
        _id ||
        'First Name' || first_name_rec.employee
        _id);
    END LOOP;
END;
/
```

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