PROGRAM 1

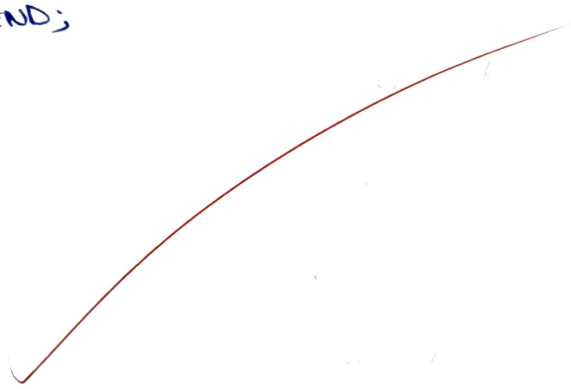
Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

```
DECLARE
    v_salary NUMBER;
    v_incentive NUMBER;
BEGIN
    Select Salary INTO v_salary
    FROM employees
    WHERE employee_id = 110;

    v_incentive := v_salary * 0.10;

    DBMS_OUTPUT.PUT_LINE ('Incentive for employee 110 is:
                          ' || 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
        "Myvar" NUMBER := 100;
BEGIN
        DBMS_OUTPUT.PUT_LINE (myvar);
END;
```



PROGRAM 3

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

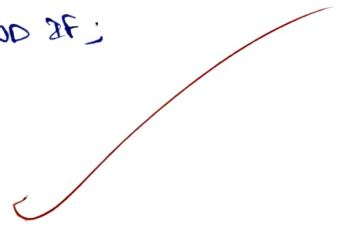
Sample table: employees

```
DECLARE
        v_salary NUMBER;
BEGIN
        SELECT salary INTO v_salary
        FROM employees
        WHERE employee_id = 122;

        UPDATE employees
        SET salary = v_salary * 1.10
        WHERE employee_id = 122;

        DBMS_OUTPUT.PUT_LINE('Salary updated for employee 122');
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.

```
CREATE OR REPLACE PROCEDURE check_conditions IS
    v_name VARCHAR2(50) := 'priya';
    v_status VARCHAR2(10) := NULL;
BEGIN
    IF v_name IS NOT NULL AND v_status IS NOT
    NULL THEN
        DBMS_OUTPUT.PUT_LINE ('Both Conditions are TRUE');
    ELSE
        DBMS_OUTPUT.PUT_LINE ('AND operator returned
                               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  VARCHAR2(50);
BEGIN
    Select Name INTO v_name
    From employees
    WHERE name LIKE 'A%';

    DBMS_OUTPUT.PUT_LINE ('Name Starting with A:' || v_name);

    Select name into v_name
    from employees
    where name LIKE 'John';

    DBMS_OUTPUT.PUT_LINE ('Name matching John : ' || v_name);

    Select name into v_name
    from employees
    where name LIKE 'A\_%' Escape '\';

    DBMS_OUTPUT.PUT_LINE ('name Starting with A_ (Literal
                          underscore): ' || v_name);
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 := 45;
        num2   NUMBER := 30;
        num_small   NUMBER;
        num_large   NUMBER;
BEGIN
        IF   num1 < num2   THEN
             num_small  :=  num1;
             num_large  := num 2;
        ELSE
             num_small   := num2;
             num_large   :=num1;
        END IF;
        DBMS_OUTPUT.PUT_LINE ('Smaller number : ' ||num_small);
        DBMS_OUTPUT.PUT_LINE ('Larger 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 update_incentive (p_emp_id NUMBER,
    p_target NUMBER ) AS
    v_incentive NUMBER;
    v_rows_updated NUMBER;
BEGIN
    v_incentive := p_target * 0.15;
    UPDATE employees
    SET incentive = v_incentive
    WHERE employee_id = p_emp_id;
    v_rows_updated := SQL%ROWCOUNT;
    IF v_rows_updated > 0 THEN
        DBMS_OUTPUT.PUT_LINE ('Record updated for employee'
            || p_emp_id);
    ELSE
        DBMS_OUTPUT.PUT_LINE ('No record updated. Employee
            ID not found.');
    END IF
END;
/
BEGIN
    update_incentive (110, 50000);
END;
```

PROGRAM 8

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

```
CREATE OR REPLACE PROCEDURE Calculate_incentive (P_emp_id NUMBER,
                                                 P_Sales  NUMBER) AS

    V_incentive  NUMBER := 0;
BEGIN
    IF P_Sales >= 100000 THEN
        v_incentive := P_Sales * 0.10;
    ELSIF P_Sales >= 50000 THEN
        v_incentive := P_Sales * 0.05;
    ELSE
        V_incentive := P_Sales * 0.02;
    END IF;
    Update employees
    SET incentive = V_incentive
    WHERE employee_id = P_emp_id;
    IF SQL%rowcount > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Incentive updated for employee'||
                             P_emp_id);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No matching employee found.
                              Record not updated.');
    END IF
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_emp_Count   NUMBER;
    v_Vacancies   NUMBER := 45;
BEGIN
    Select  Count(*)  INTO  v_emp_Count
    from    employees
    where   department_id = 50;
    DBMS_OUTPUT.PUT_LINE ('Number of employees in department 50: ' ||
                          v_emp_Count)
    IF  v_emp_Count  < v_ Vacancies  THEN
        DBMS_OUTPUT.PUT_LINE ('Vacancies available : ' || (v_Vacancies -
        v_emp_Count));
    ELSE
        DBMS_OUTPUT.PUT_LINE ('No vacancies available in department
                              50.');
    END IF;
END;
```

PROGRAM 14

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

```
DECLARE
    CURSOR  emp_job_Cursor  IS
        Select  e.employee_id
                e.first_name || ' ' || e.last_name AS full_name,
                j.start_date
        from    employees e
        JOIN    job_history j on  e.employee_id = j.employee_id;

    v_emp_id        employees.employee_id %TYPE;
    v_emp_name      VARCHAR2(100);
    v_start_date    job_history.start_date %TYPE;
BEGIN
    OPEN  emp_job_Cursor;
    Loop
        FETCH  emp_job_cursor  INTO  v_emp_id, v_emp_name, v_start_
        date;
        EXIT  WHEN  emp_job_Cursor %NOT  FOUND;

        DBMS_OUTPUT.PUT_LINE ('ID: ' || v_emp_id || ', Name: ' ||
            v_emp_name || ', Job Start Date : ' || TO_CHAR
            (v_start_date, 'DD-MON-YYYY'));

    END  loop;
    CLOSE  emp_job_Cursor;

END;
```

PROGRAM 15

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

DECLARE

Cursor emp_job_Cursor IS

Select e.employee_id,
e.first_name || ' ' || e.last_name AS full_name,
j.end_date
FROM employees e
JOIN job_history j ON e.employee_id = j.employee_id;

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

v_emp_id employees.employee_id %TYPE;
v_emp_name VARCHAR2(100);
v_end_date job_history.end_date %TYPE;

BEGIN

OPEN emp_job_Cursor;
LOOP
FETCH emp_job_Cursor INTO v_emp_id, v_emp_name,
v_end_date;
END LOOP;
END; CLOSE emp_id job_Cursor;