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;
        v_incentive := v_salary * 0.10;
        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
    "MyVar" NUMBER := 100;
BEGIN
    DBMS_OUTPUT.PUT_LINE (MyVar);
END;
/
```

(Incorrect)

```
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_emp-id  employees.employee_id %TYPE := 122;
  v_old-sal  employees.salary %TYPE;
  v_new_sal  employees.salary %TYPE;

BEGIN
  Select salary into v_old-sal from employees where
    employee_id = v_emp-id;
    v_new-sal := v_old-sal *1.10;
    UPDATE employees set salary = v_new-sal where
    employee_id = v_emp-id;
    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Not found');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error');
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_Condition (
                              P_val1 varchar2,
                              P_val2 varchar2) AS

BEGIN

    DBMS_OUTPUT.PUT_LINE (NVL (P_val1, 'NUL'));

    DBMS_OUTPUT.PUT_LINE (NVL (P_val2, 'NULL'));

    IF (P_val1 IS NOT NULL) AND (P_val2 IS NOT NULL)

  THEN
      DBMS_OUTPUT.PUT_LINE ('Both values are NOT NULL and
  returned True');

  ELSE
      DBMS_OUTPUT.PUT_LINE('One or both values are NULL
  AND 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(30);

BEGIN
    v_name := 'JOHN_SMITH';

    DBMS_OUTPUT.PUT_LINE('Value:' || v_name);

    IF v_name LIKE 'JOHN%' THEN
            DBMS_OUTPUT.PUT_LINE('LIKE "JOHN%" -> TRUE
    (Starts with JOHN)');

    ELSE
      DBMS_OUTPUT.PUT_LINE('LIKE "JOHN%" -> FALSE');

    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.

```
DECLARE

    num1 NUMBER := &num1;
    num2 NUMBER := &num2;
    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 cdc_incentive (
        emp_id IN NUMBER,
        target IN NUMBER,
        achieved in NUMBER )

AS

incentive NUMBER;

BEGIN

    IF achieved >= target then
        incentive := (achieved- target ) * 0.10;
        DBMS_OUTPUT. PUT_LINE ('Incentive : '|| incentive);

    ELSE
        DBMS_OUTPUT. PUT_LINE( 'No records updated');

    ENDIF;

END;
/
```

PROGRAM 8

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

```
Create or replace Procedure Sale_incentive(
        emp_id IN NUMBER,
        Sale_amount in NUMBER )
AS
incentive NUMBER;
BEGIN
    IF Sale_amount >=100000 then
        incentive := Sale_amount * 0.10;
    ELSIF Sale_amount >50000 THEN
        incentive := Sale_amount * 0.07;
    ELSIF Sale_amount >= 20000 THEN
        incentive := Sale_amount * 0.05;
    ELSE
        incentive := 0;
    END IF;
    DBMS_OUTPUT.PUT_LINE( 'Incentive : '||incentive);
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
        emp-Count NUMBER;
        vocancies NUMBER := 45;

BEGIN
        Select Count(*) INTO emp-Count from employees where
        department id = 50;

        If emp-count < vocancies then
                DBMS-OUTPUT. PUT-LINE ('Vocancies available :' ||(vocancies-
                                                                  empcount))

        ELSE
                DBMS OUTPUT. PUT-LINE ('No vocancies available');

        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

    dept_id number := &dept_id;

    emp_count NUMBER;

    total_vacancies NUMBER := 45;

BEGIN
    Select count(*) INTO emp_count from employees where
    department_id = dept_id;
    IF emp_count < vacancies THEN
        DBMS_OUTPUT.PUT_LINE('Vacancies Available :'||
                                (vacancies - emp_count));
    ELSE
        DBMS_OUTPUT.PUT_LINE('Vacancies Not Available');

    ENDIF;

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
    Cursor c is Select e.employee_id , e.firstname || ' ' ||e.last_na
as emp-name, j.job_title , e.hire date , e.Salary from employees e
JOIN Jobs J ON e.job-id = j.job_id;

BEGIN
    for r in c Loop
        DBMS_OUTPUT.PUT_LINE( 'ID': || r.employee_id || 'Name:'
|| r.emp-name || 'Job' || r. job_title || 'Hiredate' ||
TO_CHAR(r.hire-date, 'DD-MON-YYYY') || 'Salary' || r.Salary);

END Loop;

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 c is Select e.emp-id, e.emp-name, d.dept-name
    from employee e JOIN department d ON d.dept-id=e.dept-id;

BEGIN
    for r in c Loop
        DBMS_OUTPUT.PUT-LINE('ID:'||r.emp-id ||'Name:'||
r.emp-name || ' Dept Name: ' || r.dept_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 c is Select job_id, job_title , min_salary from
    job;

BEGIN
    for r in c Loop
        DBMS_OUTPUT.PUT_LINE('Job_id: ||r.job_id ||
( Job_title :' || r.job_title || 'Min Salary:'|| r.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 C is Select e.emp_id, e.emp_name, jh.Start_date,
from employee e JOIN job history jh ON e.emp_id=jh.emp_id;

BEGIN
    for r in C Loop
        DBMS_OUTPUT.PUT_LINE('ID:'||r.emp_id||'Name:'
|| r.emp_name || 'Job Start Date:' || To_CHAR(r.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 c is Select e.emp_id, e.emp_name, jh.end_date from
  employees e JOIN job_history jh ON e.emp_id = jh.emp_id;

BEGIN
  for r in c loop

DBMS_OUTPUT.PUT_LINE
('ID: '||r.emp_id||
', Name: '||r.emp_name||
', Job End Date: '||
TO_CHAR(r.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	