PROGRAM 1

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

```
SET SERVEROUTPUT ON;
DECLARE
 v_employee_id          CONSTANT NUMBER := 100;
 v_salary               employees.salary%TYPE;
 v_commission_pct       employees.commission_pct%TYPE;
 v_incentive_amount     NUMBER(10,2);
 c_incentive_rate       CONSTANT NUMBER := 0.10;
BEGIN
    SELECT
        salary,
        NVL(commission_pct, 0)
    INTO
        v_salary,
        v_commission_pct
    FROM
        employees
    WHERE
        employee_id = v_employee_id;

END;
```

PROGRAM 2

(161)

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

```
DECLARE
    "eMp_Sal" NUMBER (8,2) := 5000;
     v_new_salary NUMBER (8,2);
BEGIN
    v_new_salary := EMP_SAL * 1.10;
DBMS_OUTPUT.PUT_LINE ('New Salary: ' || v_new_salary);
    END;
```

PROGRAM 3

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

Sample table: employees

```
DECLARE
  v_employee_id  CONSTANT NUMBER := 122;
  c_raise        CONSTANT NUMBER := 0.05 ;

BEGIN
      UPDATE employees
      SET salary = salary * (1+ c_raise)
      WHERE employee_id = v_employee_id ;

    COMMIT ;

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
    A NUMBER := 10 ;
    B VARCHAR2(10) := 'Data' ;
BEGIN
  IF A IS NOT NULL AND B IS NOT NULL THEN
    DBMS_OUTPUT_PUT_LINE ('Result : TRUE')
  ELSE
    DBMS_OUTPUT_PUT_LINE ('Result': FALSE)

    END IF ;

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
     v_count NUMBER;
  BEGIN
    SELECT COUNT (last_name) INTO v_count
    FROM employees
    WHERE last_name LIKE '%a%' ;

  DBMS_OUTPUT. PUT_LINE ('Searching for literal
    50%: ' || v_count);

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
number 1   NUMBER := 45;
number 2   NUMBER := &;
        num_small  NUMBER;
        num_large  NUMBER;
BEGIN
    num_small := LEAST (number 1, number 2);
    num_large := GREATEST (number 1, number 2);

DBMS_OUTPUT.PUT_LINE ('Small Number (num_small)
                      '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 Calculate_incentive (
    p_emp_id IN NUMBER)
AS
  v_sales_data employee_sales%ROWTYPE;
BEGIN
    SELECT * INTO v_sales_data
    FROM employee_sales
    WHERE emp_id = p_emp_id;
IF v_sales_data.sales_amount >= v_sales_data.sales_target THE
      UPDATE employee_sales
  SET incentive = v_sales_data.sales_amount * 0.10
  WHERE emp_id = p_emp_id;
  DBMS_OUTPUT.PUT_LINE ('Record Updated')
```

PROGRAM 8

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

```
CREATE OR REPLACE PROCEDURE calculate (
    p_emp_id IN NUMBER,
    p_sales_amount IN NUMBER )
AS
    v_rate NUMBER ;
BEGIN
    IF p_sales_amount > 15000 THEN
        v_rate := 0.10;
    ELSIF p_sales_amount > 10000 THEN
        v_rate := 007;
    ELSE
        v_rate := 0.05;
    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_dept_id           CONSTANT NUMBER := 50;
    c_total_vacancies   CONSTANT NUMBER := 45;
    v_remaining_vacancies NUMBER;
BEGIN
    SELECT c_total_vacancies - COUNT(*)
    INTO   v_remaining_vacancies
    FROM   employees
    WHERE  department_id = v_dept_id
IF v_remaining_vacancies > 0 THEN
DBMS_OUTPUT.PUT_LINE('Department 50 HAS';
    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.

```
CREATE OR REPLACE PROCEDURE check_dept(
    p_dept_id   IN NUMBER,
    p_vacancy_limit IN NUMBER
)
AS
    v_remaining_vacancies NUMBER;
BEGIN
SELECT p_vacancy_limit - COUNT(*)
INTO   v_remaining_vacancies
FROM   employees
WHERE  department_id = p_dept_id;

ELSE
    DBMS_OUTPUT.PUT_LINE('Dept'||p_de
                         )

    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
DBMS_OUTPUT.PUTLINE('---Employee Roster---');
FOR emp_rec IN (
    SELECT
        employee_id,
        job_id,
        hiredate
    FROM
        employee
    ORDER BY
        employee_id
);
END LOOP;
END;
```

PROGRAM 12

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

```
BEGIN
    DBMS_OUTPUT.PUTLINE ('--- Roster ---')
FOR rec IN (
    SELECT
        e.employee_id,
        d.department_name
    FROM
        employees e
    JOIN
        departments d ON e.department_id = d.department_id
    ORDER BY
        e.employee_id

END LOOP;

END;
```

PROGRAM 13

Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```
BEGIN
  DBMS_OUTPUT.PUT_LINE('--- Job Title and Minimum Salaries ---');
FOR job_rec IN (
    SELECT
        job_id,
        job_title,
        min_salary
    FROM
        jobs
    ORDER BY
        job_id
  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;

BEGIN
    DBMS_OUTPUT.PUT_LINE (
'Job history Start Date');

FOR emp_rec IN (
    SELECT
        e.employee_id,
        jh_start_date
    FROM
        employees e
    ORDER BY
        e.employee_id, jh_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
    DBMS_OUTPUT.PUT_LINE (
    'Job History End Date');
FOR emp_rec IN (
    SELECT e.employee_id,
           jh.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	