PROGRAM 1

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

```
DECLARE
    v_employee_id       employees.employee_id %TYPE := 100;
    v_salary            employees.salary %TYPE;
    v_incentive         NUMBER;
    v_incentive_pct     CONSTANT NUMBER := 0.10
BEGIN
    SELECT salary into v_salary from employees
    where employee_id = v_employee_id;
    v_incentive := v_salary * v_incentive_pct;

    DBMS_OUTPUT.PUT_LINE ('Incentive for empID' || v_employee_id ||
                          '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
    v_test_variable   NUMBER := 100;
BEGIN
    EXECUTE IMMEDIATE 'CREATE OR REPLACE FUNCTION
"MyFunction" RETURN NUMBER IS  BEGIN RETURN 1; END;';
DECLARE
    v_result  NUMBER;
BEGIN
    v_result := my function;
    DBMS_OUTPUT.PUT_LINE ('RESULT:' ||v_result);
EXCEPTION
    WHEN OTHERS THEN
                DBMS_OUTPUT_LINE ('ERROR:' ||SQLERRM);
END;
DBMS_OUTPUT.PUT_LINE ('value:' || V_TEST_VARIABLE);
```

PROGRAM 3

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

```
DECLARE
        v_employee_id       employees.employee_id%TYPE := 122;
        v_newsalary         employees.salary%TYPE;
        v_current_sal       employees.salary%TYPE;
        v_adj               NUMBER := 500;
BEGIN
   Select salary into v_currentsal from employees where
    employee_id = v_employee_id;
   v_new_sal := v_current_sal + v_adj;
   UPDATE employees SET salary = v_newsalary where
   employee_id = v_employee_id;
   COMMIT;
   DBMS_OUTPUT.PUT_LINE ('Salary Updated!');
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 update_emp_status(p_emp_id IN
   employees.empid%TYPE ) AS
  v_sal      employees.salary%TYPE
  v_dept_id     employees.department_id%TYPE;
BEGIN
 Select sal, dept_id into v_sal, v_dpid from employees
 where emp_id = p_emp_id;
 IF v_sal IS NOT NULL AND v_dpid IS NOT NULL THEN
       UPDATE employees SET status='Active' where emp_id =
       p_emp_id;
```

```
        DBMS_OUTPUT.PUT_LINE('Status updated to Active')
    ELSE
        DBMS_OUTPUT.PUT_LINE("Status cannot be updated");
    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_emp_name    emp.name%TYPE;
BEGIN
    FOR rec IN (select name from emp where name LIKE 'J_%'
                ESCAPE '\')
    LOOP
        DBMS_OUTPUT.PUT_LINE(rec.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 := 25;
    num2        NUMBER := 10;
    num_small   NUMBER;
    num_large   NUMBER;
BEGIN
    IF num1 < num2 THEN
        num_s := num1;
        num_l := num2;
    ELSE
        num_s := num2;
        num_l := num1;
    END IF;
    DBMS_OUTPUT.PUT_LINE (num_s)
    DBMS_OUTPUT.PUT_LINE (num_l);
```

```
CREATE OR REPLACE cal_in ( p_emp_id IN emp.emp_id %TYPE,
p_tar_achevd IN NUMBER ) AS v_inc NUMBER; v_tar
NUMBER := 1000;
BEGIN
   IF p_tar_achevd >= v_target THEN
            v_incentive := p_target_achevd * 0.10;
   ELSE
        v_incentive := 0;
```

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.

```
END IF;
UPDATE emp SET inc = v_inc where emp_id = p_empid;
IF SQL%ROWCOUNT > 0 THEN
     DBMS_OUTPUT.PUTLINE ("Record updated");
ELSE
    DBMS_OUTPUT.PUT_LINE (" NO record updated");
END IF;
END;
```

PROGRAM 8

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

```
Create or replace Procedure cal_inc ( p_sal_amt IN
NUMBER ) AS v_inc_num;
BEGIN
    IF p_sal_amt >= 5000 THEN
          v_inc := p_sal_amt * 0.15;
    ELSE
        v_inc := p_sal_amt * 0.05;
    END IF;
    DBMS_OUTPUT.PUT_LINE ( p_sal_amt || v_inc);
```

```
DECLARE
    v_emp_count    NUMBER;
    v_vacancies    CONSTANT NUMBER := 45;
BEGIN
    Select  count(*) into v_emp_count  FROM employees where
    dept_id = 50;
```

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.

```
DBMS_OUTPUT. PUT_LINE ( v_emp_count) ;
IF  v_emp_count < v_vacancies  THEN
        DBMS_OUTPUT. PUT_LINE ('50 vacancies available') ;
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_id  NUMBER := 50;
    v_emp_count   NUMBER;
    v_total_pos   NUMBER := 100;
    v_vac         NUMBER;
BEGIN
    Select  COUNT(*) INTO v_emp_count FROM employees where
    dept_id = v_dept_id ;
    v_vacancies := v_total_pos -v_emp_count;
    DBMS_OUTPUT. PUT_LINE  ( v_emp_count);
   IF v_vac >0 THEN
    DBMS_OUTPUT. PUT_LINE( " vacancies available ");
```

```
ELSE
  DBMS_OUTPUT.PUT_LINE ("IS FULLY STAFFED");
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.

```
DECLARE
BEGIN
   FOR rec IN( select emp_id , first_name || ' ' || last_name
   AS emp_name, job_title , hire_date , salary from employees
   JOIN jobs ON employees.job_id = jobs.job_id )
   LOOP
   DBMS_OUTPUT.PUT_LINE (rec.emp_id || rec.emp_name || rec.job_title ||
   TO_CHAR(rec.hire_date, 'DD-MON-YYYY') || rec.salary );
   END LOOP;
END;
```

PROGRAM 12

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

```
DECLARE
BEGIN
   FOR rec IN ( SELECT e.emp_id, e.f_name || ' ' ||
   e.l_name AS emp_name, d.dept_name FROM employees e
   JOIN departments d ON e.dept_id = d.dept_id )
   LOOP
      DBMS_OUTPUT.PUT_LINE ( rec.emp_id || rec.emp_name ||
                             rec.dept_name );
   END LOOP;
END;
```

PROGRAM 13
Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```
DECLARE
BEGIN
    FOR rec IN ( SELECT job_id, job_title, min_sal from
   jobs)
   LOOP
      DBMS_OUTPUT.PUT_LINE(rec.job_id || rec.job_title ||
       rec.min_sal);
    END LOOP;

END;
```

PROGRAM 14
Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

```
DECLARE
BEGIN
    FOR rec IN ( SELECT e.emp_id, e.first_name || ' ' ||
    e.last_name AS emp_name, jh.start_date FROM
    employees e JOIN job_history jh ON e.emp_id = jh.emp_id)
   LOOP
    DBMS_OUTPUT.PUT_LINE(rec.emp_id, rec.emp_name,
              TO_CHAR(rec.start_date, 'DD-MON-YYYY'));
     END LOOP;
END;
```

15)

```
DECLARE
    v_employee_id    employees.employee_id %TYPE;
    v_first_name     employees.first_name %TYPE;
    v_last_name      employees.last_name %TYPE;
    v_end_date       job_history.end_date %TYPE;

    CURSOR emp_cur IS
        SELECT e.employee_id, e.first_name, e.last_name,
               j.end_date FROM employees e JOIN
               job_history j ON e.employee_id = j.employee_id;

BEGIN
    FOR emp_record IN emp_cursor LOOP
        v_employee_id := emp_record.employee_id;
        v_firstname := emp_record.first_name;
        v_last_name := emp_record.last_name;
        v_end_date := emp_record.end_date;

        DBMS_OUTPUT.PUT_LINE('Employee ID :' || v_employee_id ||
        'Name :' || v_firstname || ' ' || v_lastname || 'Job History End
        Date :' || TO_CHAR(v_end_date, 'YYYY-MM-DD'));
    END LOOP;

END;
```

PROGRAM 15

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

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