PL/SQL

PROGRAM 1

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

```
DECLARE
    v_salary employees.salary%type;
    v_incentive NUMBER
BEGIN
    SELECT salary INTO v_salary FROM employees
    WHERE employee_id = 110;
IF
    v_salary > 10000 THEN
    v_incentive := v_salary * 0.10;
ELSE IF
        v_salary BETWEEN 5000 AND 10000 THEN
            v_incentive := v_salary * 0.07;
    ELSE
        v_incentive := v_salary * 0.05;
    END IF;
    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
   "Name" VARCHAR2(20) := 'Alice';
BEGIN
   DBMS_OUTPUT.PUT_LINE("Name");
END;
/
```

PROGRAM 3

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

Sample table: employees

```
DECLARE
  v_salary employees.salary%TYPE;
BEGIN
   SELECT salary INTO v_salary
   FROM employees
   WHERE employee_id = 122;
  v_salary := v_salary * 0.10;
  UPDATE employees
  SET salary = v_salary
WHERE employee_id = 122;
DBMS_OUTPUT.PUT_LINE('salary updated');
   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 SERVER OUTPUT ON;
/
CREATE OR REPLACE PROCEDURE check_null_and
condition IS
   V_num1 NUMBER = 10;
   V_num2 NUMBER = NULL;
BEGIN
    IF V_num1 IS NOT NULL THEN
   DBMS_OUTPUT.PUT_LEN('V_num1 is NOT NULL)
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(20) := 'A_Rajit';
BEGIN
   IF v_name LIKE 'A%' THEN
DBMS_OUTPUT.PUT_LINE('Name starts with A');
END IF;
 IF v_name LIKE 'A_%' THEN
  DBMS_OUTPUT.PUT_LINE('Name starts with A
                  in atleast two');
  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 := 25;
    num2 NUMBER := 10;
    num_small NUMBER;
    num_larger NUMBER;
BEGIN
    IF num1 < num2 THEN
        num_small := num1;
        num_larger := num2;
    ELSE
        num_small := num2;
        num_larger := num1;
    END IF;
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.

```
SET SERVEROUTPUT ON;
/
CREATE OR REPLACE PROCEDURE calc (
    P_emp_id IN employees_id %. TYPE
    P_target IN NUMBER ) IS
    v_incentive NUMBER;
    v_count NUMBER;
BEGIN
    IF P_target >= 100 THEN
        v_incentive := 5000;
    ELSE
        v_incentive := 1500;
    END IF;
END;
/
```

PROGRAM 8

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

```
SET SERVEROUTPUT ON;
/
CREATE OR REPLACE PROCEDURE CALL (
    P_sales IN NUMBER ) IS
  V_incentive NUMBER := 0;
BEGIN
   IF P_sales >= 10000 THEN
      V_incentive := 1000;
  ELSE IF P_sales >= 75000 THEN
     V_incentive := 7500;
  ELSE IF P_sales >= 50000 THEN
     V_incentive := 1000;
  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_empl_count NUMBER;
   v_vacancies NUMBER := 45;
BEGIN
    SELECT COUNT (*) INTO v_empl_count
   FROM employees
   WHERE dept_id := 50;
DBMS_OUTPUT.PUT_LINE ('Employees');
IF v_empl_count < v_vaccancies> THEN
   DBMS_OUTPUT.PUT_LINE ('Available');
ELSE
   DBMS_OUTPUT.PUT_LINE ('Not 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
  v_dept_id  NUMBER := 45;
  v_emp_count  NUMBER;
  v_total_positions  NUMBER := 45;
  v_vacancies  NUMBER;
BEGIN
   SELECT  COUNT(*) INTO  v_emp_count
   FROM  employees;
v_vacancies := v_total_positions - v_emp_count;
IF  v_vacancies THEN
    DBMS_OUTPUT.PUT_LINE('Vacancies');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Not Available');
END;
/
```

PROGRAM 11

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

```
SET SERVER OUTPUT ON;
/
BEGIN
  FOR emp1 IN (SELECT employee_id, firstname||" "||
last_name AS emp_name, job_id, hire_date
  FROM employees LOOP
  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;
/
BEGIN
  FOR emp1 IN (SELECT e.employee_id, e.first_name
|| ' ' || e.last_name AS emp_name, d.department
  FROM employees) LOOP
  DBMS_OUTPUT.PUT_LINE(emp1);
  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;
/
BEGIN
  FOR job_rec IN (SELECT job_id, job_title, min_
FROM jobs) LOOP
DBMS_OUTPUT.PUT_LINE (job_rec);
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 SERVER OUTPUT ON;
/
BEGIN
 FOR emp_list IN (SELECT e.employee_id, e.fir
 e.last_name, e.start FROM employee e) name
LOOP
DBMS_OUTPUT.PUT_LINE (emp_list);
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
  FOR emp_list IN (SELECT e.employee_id,
e.first_name, e.last_name,
e.end_of_job FROM employee e)
    DBMS_OUTPUT.PUT_LINE (emp_list);
  END LOOP;
  END;
/
```

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