PL/SQL PROGRAMS

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

```
DECLARE
   v_employee_id NUMBER := 110;
   v_salary      NUMBER;
   v_incentive   NUMBER(10, 2);
BEGIN
      SELECT salary
   INTO v_salary
   FROM employees
   WHERE employee_id = v_employee_id;

     v_incentive := v_salary * 0.10;

      DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);
   DBMS_OUTPUT.PUT_LINE('Salary: ' || TO_CHAR(v_salary,
'FM999,999.00'));
   DBMS_OUTPUT.PUT_LINE('Calculated Incentive: ' || v_incentive);

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Employee with ID ' || v_employee_id || ' not
found.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
```

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 := 10;
```

```
    myvar NUMBER := 20;
BEGIN
  DBMS_OUTPUT.PUT_LINE(MyVar);
  DBMS_OUTPUT.PUT_LINE("myvar");
END;
```

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

```
DECLARE
  v_old_salary employees.salary%TYPE;
  v_new_salary employees.salary%TYPE;
BEGIN
  SELECT salary
  INTO v_old_salary
  FROM employees
  WHERE employee_id = 122;

  v_new_salary := v_old_salary * 1.10;

  UPDATE employees
  SET salary = v_new_salary
  WHERE employee_id = 122;

  DBMS_OUTPUT.PUT_LINE('Salary adjusted successfully.');
  DBMS_OUTPUT.PUT_LINE('Old Salary: ' || v_old_salary);
  DBMS_OUTPUT.PUT_LINE('New Salary: ' || v_new_salary);

 END;
```

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_null_and_condition IS
  v_num1 NUMBER := 10;
  v_num2 NUMBER := NULL;
BEGIN
  IF v_num1 IS NOT NULL AND v_num1 > 5 THEN
    DBMS_OUTPUT.PUT_LINE('Condition 1: TRUE (v_num1 is not null
and greater than 5)');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Condition 1: FALSE');
  END IF;

  IF v_num2 IS NOT NULL AND v_num1 > 5 THEN
    DBMS_OUTPUT.PUT_LINE('Condition 2: TRUE');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Condition 2: FALSE (v_num2 is null, so
AND condition fails)');
  END IF;
END;

Begin
Exec check_null_and_condition;
```

5.Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

```
DECLARE
  v_name VARCHAR2(50) := 'JOHN_SMITH';
BEGIN
  IF v_name LIKE 'JOHN%' THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Matches pattern: JOHN%');
  END IF;

  IF v_name LIKE '%SMITH' THEN
    DBMS_OUTPUT.PUT_LINE('Matches pattern: %SMITH');
  END IF;

  IF v_name LIKE 'J_HN%' ESCAPE '\' THEN
    DBMS_OUTPUT.PUT_LINE('Matches pattern: J_HN% with escape
character');
  END IF;
END;
```

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:=20;
  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: '||num_small);
  DBMS_OUTPUT.PUT_LINE('Large: '||num_large);
END;
```

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 calc_incentive(p_emp_id
NUMBER,p_target NUMBER) IS
  v_incentive NUMBER;
BEGIN
  IF p_target>=10000 THEN
    v_incentive:=p_target*0.1;
    DBMS_OUTPUT.PUT_LINE('Incentive: '||v_incentive);
    DBMS_OUTPUT.PUT_LINE('Record updated');
  ELSE
    DBMS_OUTPUT.PUT_LINE('No incentive, record not updated');
  END IF;
END;



BEGIN
  calc_incentive(101,12000);
END;
```

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

```
CREATE OR REPLACE PROCEDURE incentive_by_sale(p_emp_id
NUMBER,p_sales NUMBER) IS
  v_incentive NUMBER:=0;
BEGIN
```

```
  IF p_sales>=20000 THEN
    v_incentive:=p_sales*0.15;
  ELSIF p_sales>=10000 THEN
    v_incentive:=p_sales*0.10;
  ELSIF p_sales>=5000 THEN
    v_incentive:=p_sales*0.05;
  ELSE
    v_incentive:=0;
  END IF;
  DBMS_OUTPUT.PUT_LINE('Incentive: '||v_incentive);
END;


BEGIN
  incentive_by_sale(101,15000);
END;
```

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_count NUMBER;
  v_vac NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM employees WHERE
department_id=50;
  v_vac:=45-v_count;
  IF v_vac>0 THEN
    DBMS_OUTPUT.PUT_LINE('Vacancies available: '||v_vac);
  ELSE
    DBMS_OUTPUT.PUT_LINE('No vacancies');
  END IF;
```

```
END;
```

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_count NUMBER;
  v_vac NUMBER;
  v_dept NUMBER:=60;
  v_total NUMBER:=45;
BEGIN
  SELECT COUNT(*) INTO v_count FROM employees WHERE
department_id=v_dept;
  v_vac:=v_total-v_count;
  IF v_vac>0 THEN
    DBMS_OUTPUT.PUT_LINE('Department '||v_dept||' has '||v_vac||'
vacancies');
  ELSE
    DBMS_OUTPUT.PUT_LINE('No vacancies in department '||v_dept);
  END IF;
END;
```

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

```
BEGIN
```

```
  FOR r IN (SELECT employee_id, first_name ||' '|| last_name AS name,
job_id, hire_date, salary FROM employees)
  LOOP
    DBMS_OUTPUT.PUT_LINE(r.employee_id ||' '|| r.name ||' '|| r.job_id ||'
'|| r.hire_date ||' '|| r.salary);
  END LOOP;
END;
```

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

```
BEGIN
  FOR r IN (SELECT e.employee_id, e.first_name||' '||e.last_name AS
name, d.department_name
          FROM employees e JOIN departments d ON
e.department_id=d.department_id)
  LOOP
    DBMS_OUTPUT.PUT_LINE(r.employee_id ||' '|| r.name ||' '||
r.department_name);
  END LOOP;
END;
```

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

```
BEGIN
  FOR r IN (SELECT job_id, job_title, min_salary FROM jobs)
```

```
  LOOP
    DBMS_OUTPUT.PUT_LINE(r.job_id ||' '| |r.job_title ||' '|| r.min_salary);
  END LOOP;
END;
```

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

```
BEGIN
  FOR r IN (SELECT e.employee_id, e.first_name||' '||e.last_name AS
name, jh.start_date
          FROM employees e JOIN job_history jh ON
e.employee_id=jh.employee_id)
  LOOP
    DBMS_OUTPUT.PUT_LINE(r.employee_id ||' '|| r.name ||' '||
r.start_date);
  END LOOP;
END;
```

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

```
BEGIN
  FOR r IN (SELECT e.employee_id, e.first_name||' '||e.last_name AS
name, jh.end_date
          FROM employees e JOIN job_history jh ON
e.employee_id=jh.employee_id)
  LOOP
```

```
      DBMS_OUTPUT.PUT_LINE(r.employee_id ||' '| |r.name ||' '||
r.end_date);
   END LOOP;
END;
```