1.Write a Pl/SQL program using FOR loop to insert ten rows into a database table.

```
CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   employee_name VARCHAR2(50)
);

DECLARE
   v_employee_name VARCHAR2(50);
BEGIN
   FOR i IN 1..10 LOOP
      v_employee_name := 'Employee_' || i;
      INSERT INTO employees (employee_id, employee_name)
      VALUES (i, v_employee_name);
 END LOOP;
   DBMS_OUTPUT.PUT_LINE('10 rows have been inserted successfully.');
END;
/
select * from employees
```

OUTPUT:

```
10 rows have been inserted successfully.

EMPLOYEE_ID EMPLOYEE_NAME
----------- ------------------------------------
          1 Employee_1
          2 Employee_2
          3 Employee_3
          4 Employee_4
          5 Employee_5
          6 Employee_6
          7 Employee_7
          8 Employee_8
          9 Employee_9
         10 Employee_10
```

2. Given the table EMPLOYEE (EmpNo, Name, Salary, Designation, DeptID), write a cursor to select the five highest paid employees from the table.

```
CREATE TABLE employee (
   empno INT PRIMARY KEY,
   name VARCHAR2(100),
   salary NUMBER(10, 2),
   designation VARCHAR2(50),
   deptid INT
);
INSERT INTO employee (empno, name, salary, designation, deptid) VALUES (1, 'John Doe', 80000, 'Manager', 10);
INSERT INTO employee (empno, name, salary, designation, deptid) VALUES (2, 'Jane Smith', 95000, 'Engineer', 20);
INSERT INTO employee (empno, name, salary, designation, deptid) VALUES (3, 'Alice Johnson', 120000, 'Director', 10);
INSERT INTO employee (empno, name, salary, designation, deptid) VALUES (4, 'Bob Brown', 75000, 'Analyst', 30);
INSERT INTO employee (empno, name, salary, designation, deptid) VALUES (5, 'Charlie Davis', 110000, 'Developer', 20);
INSERT INTO employee (empno, name, salary, designation, deptid) VALUES (6, 'Eve White', 130000, 'Manager', 10);
INSERT INTO employee (empno, name, salary, designation, deptid) VALUES (7, 'Grace Harris', 70000, 'Analyst', 20);
INSERT INTO employee (empno, name, salary, designation, deptid) VALUES (8, 'Frank Clark', 85000, 'Technician', 30);
INSERT INTO employee (empno, name, salary, designation, deptid) VALUES (9, 'Helen Adams', 105000, 'Consultant', 20);
INSERT INTO employee (empno, name, salary, designation, deptid) VALUES (10, 'Ivy Lewis', 95000, 'Developer', 10);
```

```
DECLARE
  CURSOR highest_paid_employees IS
    SELECT empno, name, salary, designation, deptid
    FROM employee
    ORDER BY salary DESC
    FETCH FIRST 5 ROWS ONLY;
  v_empno employee.empno%TYPE;
  v_name employee.name%TYPE;
  v_salary employee.salary%TYPE;
  v_designation employee.designation%TYPE;
  v_deptid employee.deptid%TYPE;
BEGIN
  OPEN highest_paid_employees;
  LOOP
    FETCH highest_paid_employees INTO v_empno, v_name, v_salary, v_designation, v_deptid;
    EXIT WHEN highest_paid_employees%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Empno: ' || v_empno || ', Name: ' || v_name ||
                ', Salary: ' || v_salary || ', Designation: ' || v_designation ||
                ', Deptid: ' || v_deptid);
  END LOOP;
  CLOSE highest_paid_employees;
END;
/
```

OUTPUT:

```
Empno: 6, Name: Eve White, Salary: 130000, Designation: Manager, Deptid: 10
Empno: 3, Name: Alice Johnson, Salary: 120000, Designation: Director, Deptid: 10
Empno: 5, Name: Charlie Davis, Salary: 110000, Designation: Developer, Deptid:
20
Empno: 9, Name: Helen Adams, Salary: 105000, Designation: Consultant, Deptid: 20
Empno: 2, Name: Jane Smith, Salary: 95000, Designation: Engineer, Deptid: 20
```

3. Given an integer i, write a PL/SQL procedure to insert the tuple (i, 'xxx') into a given relation.

```
CREATE TABLE sample (
   col1 INT,
   col2 VARCHAR2(50)
);
CREATE OR REPLACE PROCEDURE insert_tuple(i IN INT) IS
BEGIN
   INSERT INTO sample (col1, col2)
   VALUES (i, 'xxx');
   DBMS_OUTPUT.PUT_LINE('Tuple (' || i || ', ''xxx'') inserted successfully.');
END;
/
EXEC insert_tuple(5);
SELECT * FROM sample;
```

OUTPUT:

```
Tuple (5, 'xxx') inserted successfully.

      COL1 COL2
---------- --------------------------------------------------
         5 xxx
```

4. Write a PL/SQL program to demonstrate Exceptions.

- User defined exception

```
DECLARE
  e_invalid_age EXCEPTION;
  v_age NUMBER := -5;
BEGIN
  IF v_age < 0 THEN
    RAISE e_invalid_age;
  ELSE
    DBMS_OUTPUT.PUT_LINE('Age is valid: ' || v_age);
  END IF;
EXCEPTION
  WHEN e_invalid_age THEN
    DBMS_OUTPUT.PUT_LINE('Error: Age cannot be negative!');
END;
/
```

OUTPUT:

```
Error: Age cannot be negative!
```

-Pre- defined Exceptions

```
DECLARE
  v_num1 NUMBER := 10;
  v_num2 NUMBER := 0;
  v_result NUMBER;
BEGIN
  v_result := v_num1 / v_num2;
  DBMS_OUTPUT.PUT_LINE('Result: ' || v_result);
EXCEPTION
  WHEN ZERO_DIVIDE THEN
    DBMS_OUTPUT.PUT_LINE('Error: Division by zero!');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred!');
END;
/
```

OUTPUT:

```
Error: Division by zero!
```

.

5. Write a PL/SQL program to demonstrate Cursors.

```
CREATE TABLE employees (
  employee_id   NUMBER PRIMARY KEY,
  first_name    VARCHAR2(50),
  last_name     VARCHAR2(50),
  department_id NUMBER
);
INSERT INTO employees (employee_id, first_name, last_name, department_id) VALUES (6,
'Alice', 'Davis', 10);
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, first_name, last_name
    FROM employees
    WHERE department_id = 10;
  v_employee_id employees.employee_id%TYPE;
  v_first_name employees.first_name%TYPE;
  v_last_name employees.last_name%TYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO v_employee_id, v_first_name, v_last_name;
    EXIT WHEN emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id ||
                    ', First Name: ' || v_first_name ||
                    ', Last Name: ' || v_last_name);
  END LOOP;
  CLOSE emp_cursor;
END;
/
```

OUTPUT:

```
Employee ID: 6, First Name: Alice, Last Name: Davis
```

6. Write a PL/SQL program to demonstrate Functions.

```
CREATE OR REPLACE FUNCTION calculate_area(length IN NUMBER, width IN NUMBER)
RETURN NUMBER IS
  area NUMBER;
BEGIN
  area := length * width;
  RETURN area;
END calculate_area;
/
DECLARE
  v_length NUMBER := 5;
  v_width  NUMBER := 12;
  v_area   NUMBER;
BEGIN
  v_area := calculate_area(v_length, v_width);
  DBMS_OUTPUT.PUT_LINE('The area of the rectangle is: ' || v_area);
END;
/
```

OUTPUT:

```
The area of the rectangle is: 60
```

7. Write a PL/SQL program to demonstrate Packages.

```
CREATE OR REPLACE PACKAGE employee_pkg AS
  c_bonus_percentage CONSTANT NUMBER := 0.10;
  FUNCTION calculate_total_salary(p_basic_salary IN NUMBER) RETURN NUMBER;
  PROCEDURE print_salary_details(p_employee_name IN VARCHAR2, p_basic_salary IN
NUMBER);
END employee_pkg;
/
CREATE OR REPLACE PACKAGE BODY employee_pkg AS
  FUNCTION calculate_total_salary(p_basic_salary IN NUMBER) RETURN NUMBER IS
    v_total_salary NUMBER;
  BEGIN
    v_total_salary := p_basic_salary + (p_basic_salary * c_bonus_percentage);
    RETURN v_total_salary;
  END calculate_total_salary;
  PROCEDURE print_salary_details(p_employee_name IN VARCHAR2, p_basic_salary IN
NUMBER) IS
    v_total_salary NUMBER;
  BEGIN
    v_total_salary := calculate_total_salary(p_basic_salary);
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || p_employee_name);
    DBMS_OUTPUT.PUT_LINE('Basic Salary: ' || p_basic_salary);
    DBMS_OUTPUT.PUT_LINE('Total Salary (with bonus): ' || v_total_salary);
  END print_salary_details;
END employee_pkg;
/
BEGIN
 employee_pkg.print_salary_details('John Doe', 5000);
END;
```

/

OUTPUT:

```
Employee Name: John Doe
Basic Salary: 5000
Total Salary (with bonus): 5500
```

8. Write PL/SQL queries to create Procedures.

```
CREATE OR REPLACE PROCEDURE calculate_rectangle_area(length IN NUMBER, width IN
NUMBER) IS
  area NUMBER;
BEGIN
  area := length * width;
  DBMS_OUTPUT.PUT_LINE('The area of the rectangle is: ' || area);
END calculate_rectangle_area;
/
BEGIN
  calculate_rectangle_area(5, 10);
END;
/
```

OUTPUT:

```
The area of the rectangle is: 50
```

9. Write PL/SQL queries to create Triggers.

```
CREATE TABLE users (
  user_id     NUMBER PRIMARY KEY,
  user_name   VARCHAR2(50),
  email       VARCHAR2(100),
  created_at  DATE
);
CREATE OR REPLACE TRIGGER trg_set_created_at
BEFORE INSERT ON users
FOR EACH ROW
BEGIN
  :NEW.created_at := SYSDATE;
END;
/
INSERT INTO users (user_id, user_name, email)
VALUES (1, 'John Doe', 'johndoe@example.com');
SELECT * FROM users
```

OUTPUT:

USER_ID	USER_NAME	EMAIL	CREATED_A
1	John Doe	johndoe@example.com	23-JAN-25