1)Demonstrate the use of group by and order by clause in rdbms

****

```
CREATE TABLE Sales1 (id INT, ProductName VARCHAR(50), Quantity INT, Price
DECIMAL(10, 2));

insert into Sales1 values(1,'laptop',2000,2000);

select *from Sales1;

insert into Sales1 values(2,'mobile',100,10000);

insert into Sales1 values(3,'tab',200,20000);

select *from Sales1;

SELECT ProductName, SUM(Quantity) AS TotalQuantity FROM Sales1 GROUP BY
ProductName ORDER BY TotalQuantity DESC;
```

---

2) .Consider the following schema for a hospital database: DOCTOR(Did , Dname , DAddress,Qualification)PATIENTMASTER(Pcode , EntryDate , DischargeDate,WardNo , Disease) a) find the deatil of the doctor who is treating the patient of ward no3 b)Find the detail of patient who are admitted within period 03/03/2020 to 25/05/2020 c)Find the deatil of patient who are suffered from blood cancer d)create view on DOCTOR And PATIENTMASTER tables

```
CREATE TABLE DOCTOR(DID NUMBER , DNAME VARCHAR(20) , DADDRESS VARCHAR(20)
,QUALIFICATION VARCHAR(20));

INSERT INTO DOCTOR VALUES(1,'VISHAL','JALGAON','MD');

INSERT INTO DOCTOR VALUES(2,'DIPTI','PUNE','MBBS');

INSERT INTO DOCTOR VALUES(3,'NITIN','MUMBAI','BMS');

INSERT INTO DOCTOR VALUES(4,'AKSHAY','NASHIK','MD');


SELECT *FROM DOCTOR;


CREATE TABLE PATIENTMASTER(PCODE NUMBER , ENTRYDATE DATE , DISCHARGEDATE DATE
,WARDNO NUMBER  , DISEASE VARCHAR(20));

INSERT INTO PATIENTMASTER VALUES(101,'19/JAN/2025','20/JAN/2025',1,'FEVER');

INSERT INTO PATIENTMASTER VALUES(102,'03/MAR/2020','25/MAR/2020',2,'CANCER');

INSERT INTO PATIENTMASTER VALUES(101,'19/OCT/2023','20/FEB/2024',3,'BLOOD-CANCER');
```

```
SELECT *FROM DOCTOR D WHERE EXISTS (SELECT 1 FROM PATIENTMASTER P WHERE WARDNO=3);
SELECT *FROM PATIENTMASTER WHERE ENTRYDATE BETWEEN '03/MAR/2020' AND '25/MAR/2020';
SELECT *FROM PATIENTMASTER WHERE DISEASE='BLOOD-CANCER';

CREATE VIEW DR AS
SELECT *FROM DOCTOR D JOIN PATIENTMASTER P ON 1=1;

SELECT *FROM DR;
```

---

3) Create a department table

a)Add colunm designation to the department table

b)insert values into table

c)List the record of dept table grouped by deptno

d)update record where deptno is 9

e)delete any column data from the table

```
CREATE TABLE DEPT5(DEPTNO NUMBER,DEPTNAME VARCHAR(20));
ALTER TABLE DEPT5 ADD DESIGNATION VARCHAR(20);

INSERT INTO DEPT5 VALUES(1,'HR','MANAGER');
INSERT INTO DEPT5 VALUES(2,'ENGINEER','PROGRAMMER');

SELECT *FROM DEPT5;

SELECT *FROM DEPT5 GROUP BY DEPTNO ,DEPTNAME , DESIGNATION;

UPDATE DEPT5 SET DEPTNAME='SENIOR' , DESIGNATION='DEVELOPER' WHERE DEPTNO=1;

ALTER TABLE DEPT5 DROP COLUMN DESIGNATION;
```

4) Create database using following schema apply integrity constraint and answer the following queries using SQL . DOCTOR(Did,Dname , DAddress , qualification) PATIENT(Pid,Pname,age,gender)

integrity constraint : 1)the values of any attribute should not be null 2)Did should be unique constraint 3)Pid should be unique constraint 4)gendr values should be Male or female

queries: a)insert at least 10 record in table b)find deatil of all table c)delete record from DOCTORS where qualification is male or female d)find detail of patient where age is less than 40 e)update the patient name where patient id is 5.

```
insert into dr values(3,'siya','nashik','bms');

insert into dr values(4,'reyu','mumbai','md');

insert into dr values(5,'sonu','pune','mbbs');

insert into dr values(6,'monu','hydrabad','bms');

insert into dr values(7,'pia','nashik','md');

insert into dr values(8,'kitu','mumbai','mbbs');

insert into dr values(9,'shobhu','pune','bms');

insert into dr values(10,'paru','sambhajinagar','md');


select *from dr;


create table patien(id int ,pname varchar(20),age int CHECK(age>=0),gender varchar(20)
CHECK(gender in('female','male','other')));

insert into patien values(101, 'James Wilson', 30, 'male');

insert into patien values(102, 'aaa', 20, 'female');

insert into patien values(103, 'bbb', 10, 'other');

insert into patien values(104, 'ccc', 18, 'male');

insert into patien values(105, 'ddd', 24, 'other');

insert into patien values(106, 'eee', 27, 'female');

insert into patien values(107, 'fff', 45, 'female');

insert into patien values(108, 'ggg', 33, 'male');

insert into patien values(109, 'hhh', 36, 'other');

insert into patien values(110, 'iii', 22, 'other');


select *from patien;
```

```
delete from dr where qualification in('male','female');

delete from patien where gender in('male','female');

select *from patien where age<40;

update patien set pname='pooja' where id=105;
```

---

5) . write a PL/SQL code to create an employee database with the table and field specified as bellow. Employee[emp no Employee name Street City] WORKS[EMP NO COMPANY_NAME_JOINING_DATE DESIGNATION SALARY] COMPANY[EMP NO CITY] MANAGES [EMP NO MANAGER_NAME , MANG_NO]

```
-- Creating the tables


CREATE TABLE Employee (

   Emp_no       NUMBER PRIMARY KEY,

   Employee_name  VARCHAR2(50),

   Street       VARCHAR2(50),

   City        VARCHAR2(50)

);


CREATE TABLE Works (

   Emp_no       NUMBER,

   Company_name   VARCHAR2(50),

   Joining_date   DATE,

   Designation    VARCHAR2(50),

   Salary       NUMBER(10,2),

   FOREIGN KEY (Emp_no) REFERENCES Employee(Emp_no)

);


CREATE TABLE Company (

   Emp_no       NUMBER,

   City        VARCHAR2(50),

   FOREIGN KEY (Emp_no) REFERENCES Employee(Emp_no)

);
```

```
CREATE TABLE Manages (
   Emp_no        NUMBER,
   Manager_name   VARCHAR2(50),
   Mang_no       NUMBER,
   FOREIGN KEY (Emp_no) REFERENCES Employee(Emp_no)
);

-- Inserting sample data

INSERT INTO Employee VALUES (101, 'Alice Smith', '123 Main St', 'New York');
INSERT INTO Employee VALUES (102, 'Bob Johnson', '456 Oak Ave', 'Los Angeles');
INSERT INTO Employee VALUES (103, 'Carol White', '789 Pine Rd', 'Chicago');

INSERT INTO Works VALUES (101, 'TechCorp', TO_DATE('2020-05-10', 'YYYY-MM-DD'), 'Engineer',
75000);
INSERT INTO Works VALUES (102, 'InnoSoft', TO_DATE('2021-07-15', 'YYYY-MM-DD'), 'Analyst',
65000);
INSERT INTO Works VALUES (103, 'WebSolutions', TO_DATE('2019-03-20', 'YYYY-MM-DD'), 'Manager',
85000);

INSERT INTO Company VALUES (101, 'New York');
INSERT INTO Company VALUES (102, 'Los Angeles');
INSERT INTO Company VALUES (103, 'Chicago');

INSERT INTO Manages VALUES (101, 'David Miller', 201);
INSERT INTO Manages VALUES (103, 'Sandra Lee', 202);

-- Commit the changes
COMMIT;
```

---

6) PL/SQL code to retrive the employee name , join date and designation from employee database of an employee whose number is input by the user

```
create table em(empno int , name varchar(20) , joidate date ,designation varchar(20), salary int);
```

```
insert into em values(1,'pooja','19/jan/2025','manager',55000);
insert into em values(2,'siya','15/mar/2025','it',45000);
insert into em values(3,'reyansh','25/june/2025','sale',50000);

select *from em;

declare
eno em.empno%type:=:employee_number;
ename em.name%type;
jdate em.joidate%type;
job em.designation%type;

begin
select name,joidate,designation into ename , jdate , job from em where empno=eno;

dbms_output.put_line('employee name : '|| ename);
dbms_output.put_line('joining date : '|| jdate);
dbms_output.put_line('designation : '|| job);
end;
```

---

7) write a pl/sql code to update the salary of employees who earn less than the average salary using cursor.

```
-- Step 1: Create the table
CREATE TABLE em1 (
   EMPLOYEE_ID NUMBER PRIMARY KEY,
   NAME VARCHAR2(50),
   SALARY NUMBER
);

-- Step 2: Insert sample data
INSERT INTO em1 VALUES (1, 'Alice', 3000);
INSERT INTO em1 VALUES (2, 'Bob', 4000);
INSERT INTO em1 VALUES (3, 'Charlie', 5000);


select *from em1;

COMMIT;


-- Step 3: PL/SQL block to update salaries using cursor
DECLARE
   avg_salary NUMBER;
   CURSOR c1 IS
      SELECT EMPLOYEE_ID FROM em1 WHERE SALARY < avg_salary;
BEGIN
   -- Calculate average salary
   SELECT AVG(SALARY) INTO avg_salary FROM em1;


   -- Loop through employees earning below average
   FOR rec IN c1 LOOP
      UPDATE em1
      SET SALARY = avg_salary
      WHERE EMPLOYEE_ID = rec.EMPLOYEE_ID;
   END LOOP;


   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Salaries updated successfully.');
END;
/
```

---

8) Write a row trigger to insert the existing values of the salary table in to a new table when the salary table is updated.

```
CREATE TABLE salary (
   empno    INT PRIMARY KEY,
   name     VARCHAR2(50),
   salary   NUMBER
);
INSERT INTO salary VALUES (1, 'Pooja', 55000);
INSERT INTO salary VALUES (2, 'Siya', 45000);
INSERT INTO salary VALUES (3, 'Reyansh', 50000);
select *from salary;

CREATE TABLE salary_backup (
   empno     INT,
   name      VARCHAR2(50),
   salary    NUMBER,
   updated_on DATE
);

CREATE OR REPLACE TRIGGER trg_salary_update_backup
BEFORE UPDATE ON salary
FOR EACH ROW
BEGIN
   INSERT INTO salary_backup (empno, name, salary, updated_on)
   VALUES (:OLD.empno, :OLD.name, :OLD.salary, SYSDATE);
END;
/
UPDATE salary
SET salary = 60000
WHERE empno = 1;

COMMIT;
SELECT * FROM salary_backup;
```

---

9) Write a trigger on the employee table which shows the old values and new values of Ename after any updation on Ename on Empolyee table.

```
CREATE TABLE em19 (
   EMPLOYEE_ID NUMBER PRIMARY KEY,
   ENAME VARCHAR2(50),
   SALARY NUMBER
);
```

```
INSERT INTO em19 VALUES (1, 'Alice', 3000);
INSERT INTO em19 VALUES (2, 'Bob', 4000);

select *from em19;

CREATE OR REPLACE TRIGGER trg_show_ename_change
BEFORE UPDATE OF ENAME ON em19
FOR EACH ROW
BEGIN
   DBMS_OUTPUT.PUT_LINE('Old ENAME: ' || :OLD.ENAME);
   DBMS_OUTPUT.PUT_LINE('New ENAME: ' || :NEW.ENAME);
END;
/

UPDATE em19 SET ENAME = 'Alicia' WHERE EMPLOYEE_ID = 1;
```

---

10) Write PL/SQL procedure to find the number of students ranging from 100- 70%, 69-60%, 59-50% & below 49% in each course from the student_course table given by the procedure as parameter.

```
CREATE TABLE STUDENT_COURSES (
   STUDENT_ID NUMBER,
   COURSE_NAME VARCHAR2(20),
   PERCENTAGE NUMBER
);
INSERT INTO STUDENT_COURSES VALUES (1, 'Math', 85);
INSERT INTO STUDENT_COURSES VALUES (2, 'Math', 65);
INSERT INTO STUDENT_COURSES VALUES (3, 'Math', 55);
INSERT INTO STUDENT_COURSES VALUES (4, 'Math', 40);

SELECT *FROM STUDENT_COURSES;

CREATE OR REPLACE PROCEDURE count_students(p_course VARCHAR2) IS
   c1 NUMBER;
```

```
    c2 NUMBER;
    c3 NUMBER;
    c4 NUMBER;
BEGIN
    SELECT COUNT(*) INTO c1 FROM STUDENT_COURSE WHERE COURSE_NAME = p_course AND
PERCENTAGE BETWEEN 70 AND 100;
    SELECT COUNT(*) INTO c2 FROM STUDENT_COURSE WHERE COURSE_NAME = p_course AND
PERCENTAGE BETWEEN 60 AND 69;
    SELECT COUNT(*) INTO c3 FROM STUDENT_COURSE WHERE COURSE_NAME = p_course AND
PERCENTAGE BETWEEN 50 AND 59;
    SELECT COUNT(*) INTO c4 FROM STUDENT_COURSE WHERE COURSE_NAME = p_course AND
PERCENTAGE < 50;

    DBMS_OUTPUT.PUT_LINE('70-100%: ' || c1);
    DBMS_OUTPUT.PUT_LINE('60-69% : ' || c2);
    DBMS_OUTPUT.PUT_LINE('50-59% : ' || c3);
    DBMS_OUTPUT.PUT_LINE('<50%   : ' || c4);
END;
/



BEGIN
    count_students('Math');
END;
/
```

11) Create a store function that accepts 2 number and returns the addition of passed values. Also, write the code to call your function

```
CREATE OR REPLACE FUNCTION add_numbers(a NUMBER, b NUMBER)
RETURN NUMBER
IS
BEGIN
    RETURN a + b;
END;
/
```

```
DECLARE
   result NUMBER;
BEGIN
   result := add_numbers(10, 20);
   DBMS_OUTPUT.PUT_LINE('Sum is: ' || result);
END;
/
```

12) Write a PL/SQL function that accepts the department number and returns the total salary of the department. Also, write a function to call the function.

```
CREATE TABLE DEPT3 (ID NUMBER , SALARY NUMBER)


INSERT INTO DEPT3 VALUES(1 , 10000)
INSERT INTO DEPT3 VALUES(2 , 20000)
INSERT INTO DEPT3 VALUES(3 , 30000)
INSERT INTO DEPT3 VALUES(4 , 40000)


SELECT *FROM DEPT3;


CREATE OR REPLACE FUNCTION getsalary(did IN NUMBER)
RETURN NUMBER
IS
   total NUMBER;
BEGIN
   SELECT SUM(SALARY) INTO total
   FROM DEPT3
   WHERE ID = did;
   RETURN NVL(total , 0);
END;
/
```

```
DECLARE
  dept_salary NUMBER;
BEGIN
   dept_salary := getsalary(2);
DBMS_OUTPUT.PUT_LINE('DEPARTMENT OF 2 SALARY IS '||dept_salary);
END;
/
```

---

13) Write a PL/SQL code to create,

1. Package specification

2. Package body.

For the insert, retrieve, update, and delete operations on a student table.

```
CREATE TABLE STU1 (ID NUMBER , NAME VARCHAR(20) , MARKS NUMBER)

INSERT INTO STU1 VALUES(1,'ABC',20)
INSERT INTO STU1 VALUES(2,'PQR',19)
INSERT INTO STU1 VALUES(3,'XYZ',18)

SELECT *FROM STU1;

CREATE OR REPLACE PACKAGE stu1pkg AS
PROCEDURE insert_student(PID NUMBER , PNAME VARCHAR2 , PMARKS NUMBER);
PROCEDURE get_student(PID NUMBER);
PROCEDURE update_student(PID NUMBER,PMARKS NUMBER);
PROCEDURE delete_student(PID NUMBER);
END stu1pkg;
/
```

```
CREATE OR REPLACE PACKAGE BODY stu1pkg AS
PROCEDURE insert_student(PID NUMBER , PNAME VARCHAR2 , PMARKS NUMBER)IS
BEGIN
INSERT INTO STU1 VALUES(PID , PNAME , PMARKS);
END;

PROCEDURE get_student(PID NUMBER) IS
v_name STU1.NAME%TYPE;
v_marks STU1.MARKS%TYPE;
BEGIN
  SELECT NAME , MARKS INTO v_name , v_marks
  FROM STU1
  WHERE ID = PID;
DBMS_OUTPUT.PUT_LINE('NAME '||v_name|| ' MARKS' ||v_marks);
END;

PROCEDURE update_student(PID NUMBER , PMARKS NUMBER) IS
BEGIN
 UPDATE STU1 SET MARKS=PMARKS WHERE  ID=PID ;
END;

PROCEDURE delete_student(PID NUMBER) IS
BEGIN
DELETE FROM STU1 WHERE ID = PID;
END;

END stu1pkg;
/


BEGIN
```

```
 stu1pkg.insert_student(4,'lmn',15);
END;
/

BEGIN
 stu1pkg.get_student(4);
END;
/

BEGIN
 stu1pkg.update_student(4,12);
END;
/
BEGIN
 stu1pkg.delete_student(4);
END;
/
```

---

14) Write a program to illustrate user-defined exceptions, built-in exceptions, and raise application error exceptions

```
DECLARE
myexc EXCEPTION;
x NUMBER :=10;
y NUMBER :=0;
BEGIN
  DBMS_OUTPUT.PUT_LINE('RESULT IS : '||(x/y));

  EXCEPTION
  WHEN ZERO_DIVIDE THEN
  DBMS_OUTPUT.PUT_LINE('DIVIDE ZERO EXCEPTION ');

  WHEN myexc THEN
```

```
 DBMS_OUTPUT.PUT_LINE('other error');
END;
/
```

15) Write a program Reserving a string using PL/SQL block

```
BEGIN
  DECLARE
    str VARCHAR2(50) := 'Hello';
    rev VARCHAR2(50) := '';
  BEGIN
    FOR i IN REVERSE 1 .. LENGTH(str) LOOP
      rev := rev || SUBSTR(str, i, 1);
    END LOOP;

    DBMS_OUTPUT.PUT_LINE(rev);
  END;
END;
/
```

16) Trigger for Auditing Table Changes

- **Create a trigger that records changes to an EMPLOYEES table (insert , update, delete) into an employees_audit table, include details like employee_id, operation_type, timestamp.**

```
CREATE TABLE EMP_AUDIT (EMPID NUMBER , OPERATION_TYPE VARCHAR(20) , OPERATION_TIMESTAMP TIMESTAMP DEFAULT SYSTIMESTAMP);


CREATE SEQUENCE sequence1 START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;


INSERT INTO EMP_AUDIT (EMPID , OPERATION_TYPE ) VALUES (101 , 'INSERT');

INSERT INTO EMP_AUDIT (EMPID , OPERATION_TYPE ) VALUES (102 , 'UPDATE');

INSERT INTO EMP_AUDIT (EMPID , OPERATION_TYPE ) VALUES (104 , 'DELETE');
```

```
SELECT *FROM EMP_AUDIT;


CREATE OR REPLACE TRIGGER trgnm
BEFORE INSERT ON EMP_AUDIT
FOR EACH ROW
BEGIN
  SELECT sequence1.NEXTVAL INTO :NEW.EMPID FROM DUAL;
END;
/


UPDATE EMP_AUDIT SET EMPID=1 WHERE OPERATION_TYPE='INSERT';
DELETE FROM EMP_AUDIT WHERE OPERATION_TYPE='DELETE';
```

17) Employee Bonus Calculation Using Cursor

- **Write a PL/SQL program using an explicit cursor to calculate and display a 10% bonus for all employees whose salary is greater than 50,000. Assume a table EMPLOYEES with columns EMPLOYEE_ID, Name, and Salary.**

```
CREATE TABLE EMP3 (EMPID NUMBER , NAME VARCHAR(20) , SALARY NUMBER);
INSERT INTO EMP3  VALUES (101 , 'ABC',40000);
INSERT INTO EMP3  VALUES (102 , 'PQR',35000);
INSERT INTO EMP3  VALUES (103 , 'XYZ',50000);


SELECT *FROM EMP3;


DECLARE
CURSOR C2 IS
SELECT EMPID , NAME , SALARY FROM EMP3 WHERE SALARY<40000;
```

```
v_id EMP3.EMPID%TYPE;
v_name EMP3.NAME%TYPE;
v_salary EMP3.SALARY%TYPE;
v_bonus NUMBER;

BEGIN
OPEN C2;
LOOP
 FETCH C2 INTO v_id , v_name , v_salary;
EXIT WHEN C2%NOTFOUND;

v_bonus := v_salary *0.10;

   DBMS_OUTPUT.PUT_LINE('ID: ' || v_id || ' NAME: ' || v_name || ' SALARY: ' || v_salary || ' 
BONUS: ' || v_bonus);
END LOOP;
CLOSE C2;
END;
/
```

---

18) Write a SQL Program to implement Aggregate Functions.

```
CREATE TABLE EMPLOYEE1 ( SALARY NUMBER);
INSERT INTO EMPLOYEE1  VALUES (40000);
INSERT INTO EMPLOYEE1  VALUES (35000);
INSERT INTO EMPLOYEE1  VALUES (50000);
SELECT *FROM EMPLOYEE1;

SELECT
COUNT(*) AS TOTAL,
SUM(SALARY) AS SUM_SALARY,
MAX(SALARY) AS MAX_SALARY,
MIN(SALARY) AS MIN_SALARY,
AVG(SALARY) AS AVG_SALARY
FROM EMPLOYEE1;
```

19) Write PL/SQL code for finding Even Numbers.

```
DECLARE
  i NUMBER;
BEGIN
  FOR i IN 1..10 LOOP
    IF MOD(i, 2) = 0 THEN
      DBMS_OUTPUT.PUT_LINE(i);
    END IF;
  END LOOP;
END;
/
```

20) Write PL/SQL code to find Larger of three numbers

```
DECLARE
  num1 NUMBER := 10;
  num2 NUMBER := 20;
  num3 NUMBER := 15;
BEGIN
  DBMS_OUTPUT.PUT_LINE('The largest number is: ' || GREATEST(num1, num2, num3));
END;
/
```

21) Write PL/SQL code to accept the text and reserve the text and test whether the given character is Palindrome or not.

```
DECLARE
ORIGINAL_TEXT VARCHAR2(20) := 'POOJA';
REVERSE_TEXT VARCHAR2(20) := '';

BEGIN
```

```
FOR i IN REVERSE 1..LENGTH(ORIGINAL_TEXT) LOOP
REVERSE_TEXT := REVERSE_TEXT || SUBSTR(ORIGINAL_TEXT , i,1);
END LOOP;

IF REVERSE_TEXT = ORIGINAL_TEXT THEN
DBMS_OUTPUT.PUT_LINE('POOJA IS PALINDROME');
ELSE
DBMS_OUTPUT.PUT_LINE('POOJA IS NOT PALINDROME');
END IF;
END;
/
```

22) Write PL/SQL code to Insert values in created tables.

```
CREATE TABLE EMP5(ID NUMBER , NAME VARCHAR(20) , SALARY NUMBER);
BEGIN
INSERT INTO EMP5 (ID , NAME , SALARY) VALUES(1,'ABC',50000);
INSERT INTO EMP5 (ID , NAME , SALARY) VALUES(2,'PQR',40000);
INSERT INTO EMP5 (ID , NAME , SALARY) VALUES(3,'XYZ',30000);
END;
/
SELECT *FROM EMP5;
```

23) Write PL/SQL code to UPDATE values in created tables by using implicit Cursors

```
CREATE TABLE employees5 (
  id NUMBER PRIMARY KEY,
  name VARCHAR2(50),
  salary NUMBER
);
BEGIN

  FOR emp_record IN (SELECT id, salary FROM employees5 WHERE salary < 60000) LOOP

    UPDATE employees5
    SET salary = salary + 5000
```

```
        WHERE id = emp_record.id;
    END LOOP;


    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Employee salaries updated successfully!');
END;
/
select *from employees5;
```

24) Write PL/SQL code to display Employee detail using explicit cursor.

```
CREATE TABLE employees0 (
    id NUMBER PRIMARY KEY,
    name VARCHAR2(50),
    salary NUMBER
);
insert into employees0 values(1,'pooja',30000);
insert into employees0 values(2,'siya',40000);
DECLARE
    CURSOR emp_cursor2 IS
        SELECT id, name, salary FROM employees0;


    v_id employees0.id%TYPE;
    v_name employees0.name%TYPE;
    v_salary employees0.salary%TYPE;
BEGIN
    FOR emp_record IN emp_cursor2 LOOP
        DBMS_OUTPUT.PUT_LINE('ID: ' || emp_record.id || ', Name: ' || emp_record.name || ', Salary: 
' || emp_record.salary);
    END LOOP;
END;
/
```

25) Write PL/SQL code in cursor to display employee names and salary

```
-- 1. Create Table
CREATE TABLE employees4 (
   name   VARCHAR2(50),
   salary NUMBER
);

-- 2. Insert Values
INSERT INTO employees4 (name, salary) VALUES ('Alice', 30000);
INSERT INTO employees4 (name, salary) VALUES ('Bob', 40000);
select *from employees4;
COMMIT;

-- 3. PL/SQL Block to Display Names and Salaries Using Cursor
DECLARE
   CURSOR emp_cursor4 IS
      SELECT name, salary FROM employees4;
BEGIN
   FOR emp_rec IN emp_cursor4 LOOP
      DBMS_OUTPUT.PUT_LINE('Name: ' || emp_rec.name || ', Salary: ' || emp_rec.salary);
   END LOOP;
END;
/
```

26) Write PL/SQL Programs in cursor using two cursor at a time.

```
-- Create tables
CREATE TABLE departments5 (
   dept_id NUMBER,
   dept_name VARCHAR2(50)
);

CREATE TABLE employee5 (
   emp_id NUMBER,
   emp_name VARCHAR2(50),
   dept_id NUMBER
);

INSERT INTO departments5 VALUES (1, 'HR');
INSERT INTO departments5 VALUES (2, 'IT');

INSERT INTO employee5 VALUES (101, 'Alice', 1);
INSERT INTO employee5 VALUES (102, 'Bob', 2);

DECLARE
   CURSOR emp_cur IS SELECT emp_name, dept_id FROM employee5;
   CURSOR dept_cur IS SELECT dept_id, dept_name FROM departments5;

   v_emp_name employee5.emp_name%TYPE;
   v_emp_dept employee5.dept_id%TYPE;

   v_dept_id departments5.dept_id%TYPE;
   v_dept_name departments5.dept_name%TYPE;
BEGIN
   FOR emp_rec IN emp_cur LOOP
      FOR dept_rec IN dept_cur LOOP
         IF emp_rec.dept_id = dept_rec.dept_id THEN
            DBMS_OUTPUT.PUT_LINE(emp_rec.emp_name || ' works in ' || dept_rec.dept_name);
         END IF;
      END LOOP;
   END LOOP;
END;
/
```

27) Write PL/SQL code in Procedure to find reverse number.

```
CREATE OR REPLACE PROCEDURE reverse_number(n IN NUMBER) IS
```

```
   r NUMBER := 0;
   x NUMBER := n;
BEGIN
   WHILE x > 0 LOOP
      r := r * 10 + MOD(x, 10);
      x := TRUNC(x / 10);
   END LOOP;

   DBMS_OUTPUT.PUT_LINE('Reverse: ' || r);
END;
/
BEGIN
   reverse_number(1234);
END;
/
```

---

28) Write PL/SQL code in Procedure to find factorial of a given number by using call Procedure

```
DECLARE
   v_input NUMBER := 5;
    v_output NUMBER;
BEGIN
     find_factorial(v_input, v_output);
   DBMS_OUTPUT.PUT_LINE('The factorial of ' || v_input || ' is ' || v_output);
END;
/
```

---

29) Write a procedure to retrieve the salary of a particular employee.

```
CREATE TABLE emp4 (
   id NUMBER,
   name VARCHAR2(20),
   sal NUMBER
);
```

```
INSERT INTO emp4 VALUES (1, 'Amit', 10000);
COMMIT;
CREATE OR REPLACE PROCEDURE get_sal (
   eid IN NUMBER,
   esal OUT NUMBER
) IS
BEGIN
   SELECT sal INTO esal FROM emp4 WHERE id = eid;
END;
/
DECLARE
   s NUMBER;
BEGIN
   get_sal(1, s);
   DBMS_OUTPUT.PUT_LINE('Salary: ' || s);
END;
/
```

---

30) Write PL/SQL code in trigger not to accept the existing Empno(Unique no).

```
CREATE TABLE Employee (
   Empno NUMBER,
   Ename VARCHAR2(50)
);
INSERT INTO Employee VALUES (101, 'John');
INSERT INTO Employee VALUES (102, 'Alice');
COMMIT;

CREATE OR REPLACE TRIGGER trg_prevent_duplicate_empno
BEFORE INSERT ON Employee
FOR EACH ROW
DECLARE
   v_count NUMBER;
BEGIN
```

```
   SELECT COUNT(*) INTO v_count
   FROM Employee
   WHERE Empno = :NEW.Empno;

   IF v_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Duplicate Empno not allowed.');
   END IF;
END;
/
INSERT INTO Employee VALUES (101, 'Bob');
```