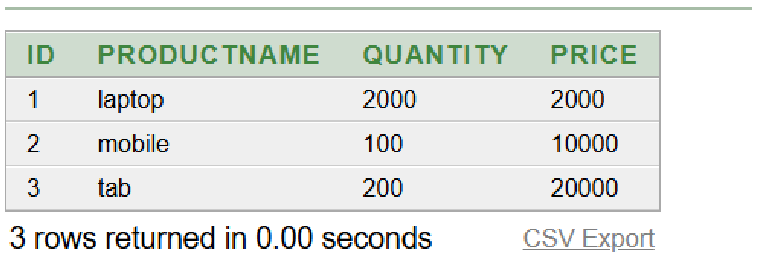
1.Demonstrate the use of group by and order by clause in rdbms

****

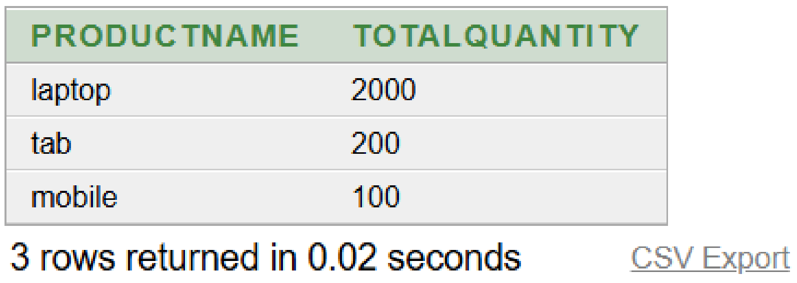
```
CREATE TABLE Sales1 (id INT, ProductName VARCHAR(50), Quantity INT, Price DECIMAL(10, 2));

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;
```

OUTPUT :-



ID	PRODUCTNAME	QUANTITY	PRICE
1	laptop	2000	2000
2	mobile	100	10000
3	tab	200	20000

3 rows returned in 0.00 seconds   CSV Export



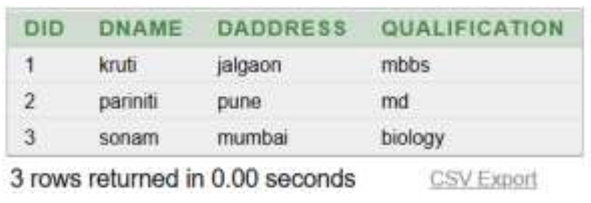
PRODUCTNAME	TOTALQUANTITY
laptop	2000
tab	200
mobile	100

3 rows returned in 0.02 seconds   CSV Export

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

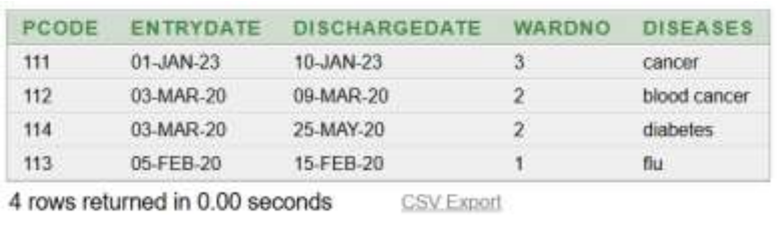
****

```
CREATE TABLE DOCTOR (Did INT PRIMARY KEY, Dname VARCHAR2(100),DAddress VARCHAR2(255),Qualification VARCHAR(100));

insert into DOCTOR values(1,'kruti','jalgaon','mbbs');

insert into DOCTOR values(2,'pariniti','pune','md');

insert into DOCTOR values(3,'sonam','mumbai','biology');

select *from DOCTOR;


CREATE TABLE PATIENTMASTERS (Pcode INT PRIMARY KEY, EntryDate DATE,DischargeDate DATE WardNo INT,Disease VARCHAR(100));


insert into PATIENTMASTERS values(111,'01-JAN-23' ,'10-JAN-23', 3,'cancer');

insert into PATIENTMASTERS values(112,'03-MAR-20' ,'09-MAR-20', 2,'blood cancer');

insert into PATIENTMASTERS values(114,'03-MAR-20' ,'25-MAY-20', 2,'diabetes');

insert into PATIENTMASTERS values(113,'05-FEB-20' ,'15-FEB-20', 1,'flu');

select *from PATIENTMASTERS;


SELECT *FROM DOCTOR D WHERE EXISTS (SELECT 1 FROM PATIENTMASTERS P WHERE WardNo = 3);

SELECT *FROM PATIENTMASTERS WHERE EntryDate BETWEEN TO_DATE('2020-03-03', 'YYYY-MM-DD') AND TO_DATE('2020-05-25', 'YYYY-MM-DD');

SELECT *FROM PATIENTMASTERS WHERE Disease = 'blood cancer';

CREATE VIEW DoctorPatientView1 AS

SELECT *FROM DOCTOR D JOIN PATIENTMASTERS P ON 1=1;

select *from DoctorPatientView1;
```



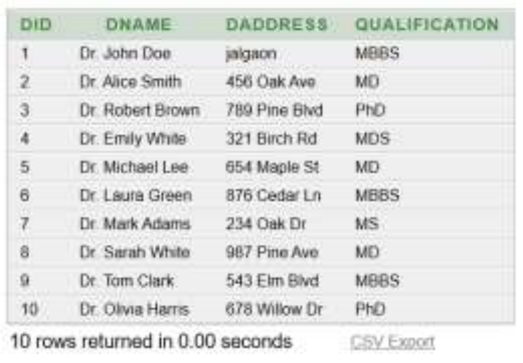
DID	DNAME	DADDRESS	QUALIFICATION
1	kruti	jalgaon	mbbs
2	pariniti	pune	md
3	sonam	mumbai	biology

3 rows returned in 0.00 seconds CSV Export



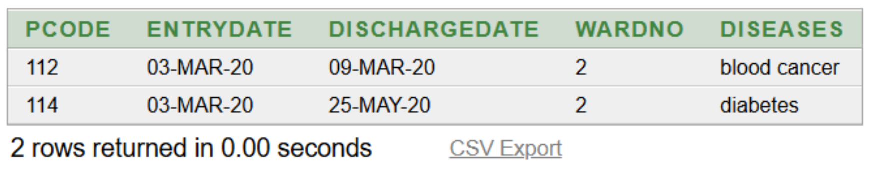
PCODE	ENTRYDATE	DISCHARGEDATE	WARDNO	DISEASES
111	01-JAN-23	10-JAN-23	3	cancer
112	03-MAR-20	09-MAR-20	2	blood cancer
114	03-MAR-20	25-MAY-20	2	diabetes
113	05-FEB-20	15-FEB-20	1	flu

4 rows returned in 0.00 seconds CSV Export



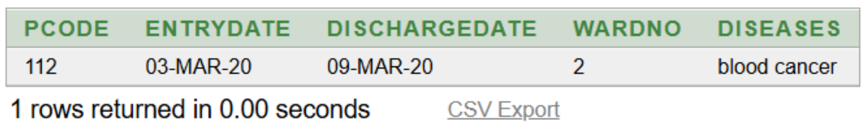
DID	DNAME	DADDRESS	QUALIFICATION
1	Dr. John Doe	jalgaon	MBBS
2	Dr. Alice Smith	456 Oak Ave	MD
3	Dr. Robert Brown	789 Pine Blvd	PhD
4	Dr. Emily White	321 Birch Rd	MDS
5	Dr. Michael Lee	654 Maple St	MD
6	Dr. Laura Green	876 Cedar Ln	MBBS
7	Dr. Mark Adams	234 Oak Dr	MS
8	Dr. Sarah White	987 Pine Ave	MD
9	Dr. Tom Clark	543 Elm Blvd	MBBS
10	Dr. Olivia Harris	678 Willow Dr	PhD

10 rows returned in 0.00 seconds CSV Export



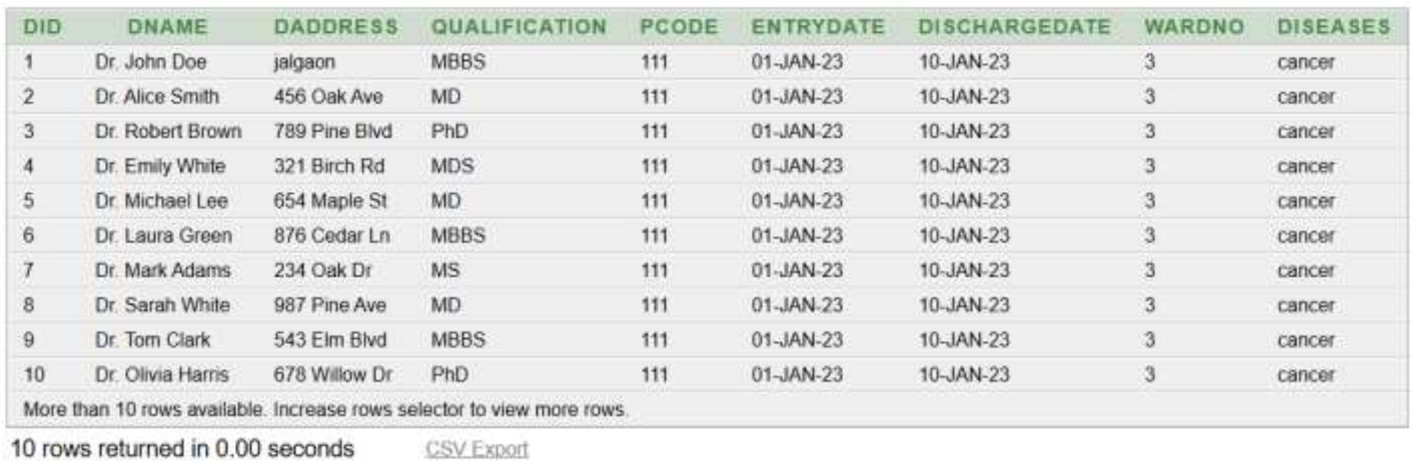
PCODE	ENTRYDATE	DISCHARGEDATE	WARDNO	DISEASES
112	03-MAR-20	09-MAR-20	2	blood cancer
114	03-MAR-20	25-MAY-20	2	diabetes

2 rows returned in 0.00 seconds CSV Export



PCODE	ENTRYDATE	DISCHARGEDATE	WARDNO	DISEASES
112	03-MAR-20	09-MAR-20	2	blood cancer

1 rows returned in 0.00 seconds CSV Export



DID	DNAME	DADDRESS	QUALIFICATION	PCODE	ENTRYDATE	DISCHARGEDATE	WARDNO	DISEASES
1	Dr. John Doe	jalgaon	MBBS	111	01-JAN-23	10-JAN-23	3	cancer
2	Dr. Alice Smith	456 Oak Ave	MD	111	01-JAN-23	10-JAN-23	3	cancer
3	Dr. Robert Brown	789 Pine Blvd	PhD	111	01-JAN-23	10-JAN-23	3	cancer
4	Dr. Emily White	321 Birch Rd	MDS	111	01-JAN-23	10-JAN-23	3	cancer
5	Dr. Michael Lee	654 Maple St	MD	111	01-JAN-23	10-JAN-23	3	cancer
6	Dr. Laura Green	876 Cedar Ln	MBBS	111	01-JAN-23	10-JAN-23	3	cancer
7	Dr. Mark Adams	234 Oak Dr	MS	111	01-JAN-23	10-JAN-23	3	cancer
8	Dr. Sarah White	987 Pine Ave	MD	111	01-JAN-23	10-JAN-23	3	cancer
9	Dr. Tom Clark	543 Elm Blvd	MBBS	111	01-JAN-23	10-JAN-23	3	cancer
10	Dr. Olivia Harris	678 Willow Dr	PhD	111	01-JAN-23	10-JAN-23	3	cancer

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds CSV Export

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

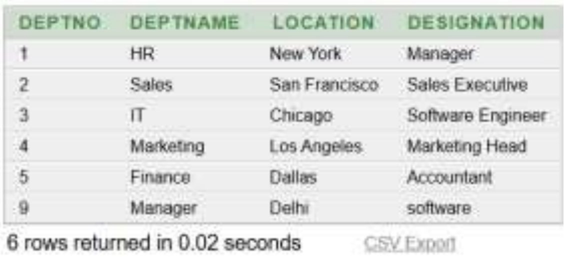
d)update record where deptno is 9

e)delete any column data from the table

****

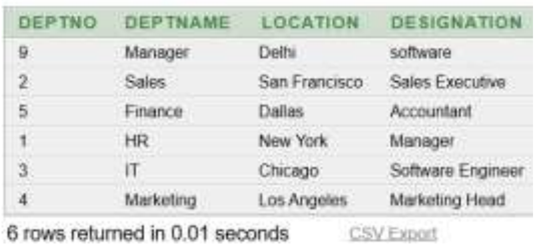
```
CREATE TABLE DEPARTMENT (DeptNo INT PRIMARY KEY, DeptName VARCHAR(30) , Location VARCHAR(30));

ALTER TABLE DEPARTMENT ADD Designation VARCHAR(50);

insert into DEPARTMENT values(1 , 'HR', 'New York', 'Manager');

insert into DEPARTMENT values(2, 'Sales', 'San Francisco', 'Sales Executive');

insert into DEPARTMENT values(3, 'IT', 'Chicago', 'Software Engineer');

insert into DEPARTMENT values(4, 'Marketing', 'Los Angeles', 'Marketing Head');

insert into DEPARTMENT values(5, 'Finance', 'Dallas', 'Accountant');

insert into DEPARTMENT values(9, 'Manager', 'Delhi', 'software');

select *from DEPARTMENT;

SELECT *FROM DEPARTMENT GROUP BY DeptNo, DeptName, Location, Designation;

UPDATE DEPARTMENT SET Location = 'Miami', Designation = 'Regional Manager' WHERE DeptNo = 9;

select *from  DEPARTMENT;


ALTER TABLE DEPARTMENT DROP COLUMN Location;

select *from DEPARTMENT;
```

OUTPUT:



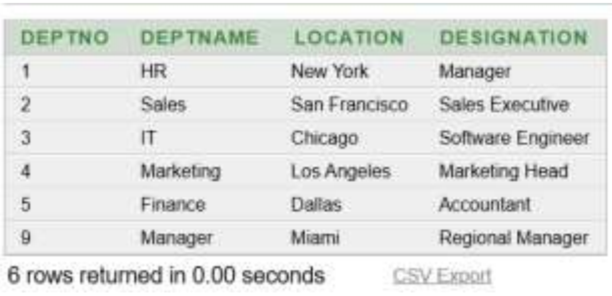
DEPTNO	DEPTNAME	LOCATION	DESIGNATION
1	HR	New York	Manager
2	Sales	San Francisco	Sales Executive
3	IT	Chicago	Software Engineer
4	Marketing	Los Angeles	Marketing Head
5	Finance	Dallas	Accountant
9	Manager	Delhi	software

6 rows returned in 0.02 seconds CSV Export



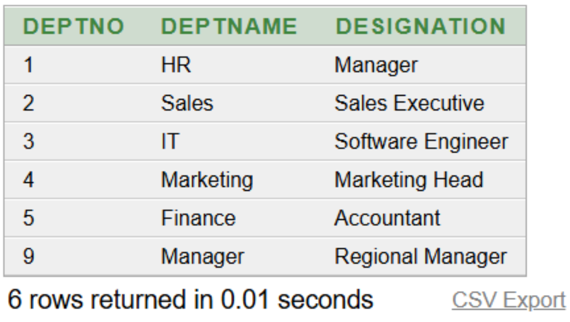
DEPTNO	DEPTNAME	LOCATION	DESIGNATION
9	Manager	Delhi	software
2	Sales	San Francisco	Sales Executive
5	Finance	Dallas	Accountant
1	HR	New York	Manager
3	IT	Chicago	Software Engineer
4	Marketing	Los Angeles	Marketing Head

6 rows returned in 0.01 seconds CSV Export



DEPTNO	DEPTNAME	LOCATION	DESIGNATION
1	HR	New York	Manager
2	Sales	San Francisco	Sales Executive
3	IT	Chicago	Software Engineer
4	Marketing	Los Angeles	Marketing Head
5	Finance	Dallas	Accountant
9	Manager	Miami	Regional Manager

6 rows returned in 0.00 seconds CSV Export



DEPTNO	DEPTNAME	DESIGNATION
1	HR	Manager
2	Sales	Sales Executive
3	IT	Software Engineer
4	Marketing	Marketing Head
5	Finance	Accountant
9	Manager	Regional Manager

6 rows returned in 0.01 seconds CSV Export

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)

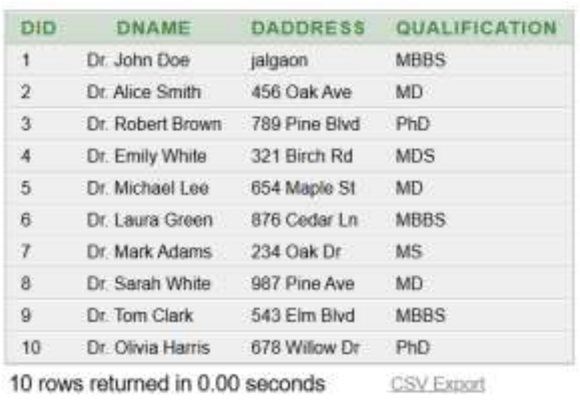
integrity constraint : 1)the values of any attribute should not be null 2)Did should be unique constraint 3)Pid should be unique constraint 4)gendr values should be Male or female

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.

****

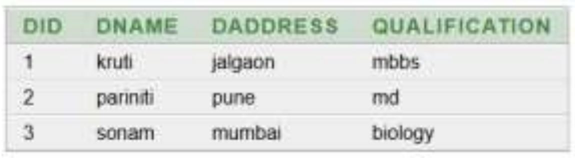
```
CREATE TABLE DOCTOR1 (Did INT PRIMARY KEY, Dname VARCHAR(50) NOT NULL,
DAddress VARCHAR(100), Qualification VARCHAR(50));

insert into DOCTOR1 values(1, 'Dr. John Doe', 'jalgaon', 'MBBS');

insert into DOCTOR1 values(2, 'Dr. Alice Smith', '456 Oak Ave', 'MD');

insert into DOCTOR1 values(3, 'Dr. Robert Brown', '789 Pine Blvd', 'PhD');

insert into DOCTOR1 values(4, 'Dr. Emily White', '321 Birch Rd', 'MDS');

insert into DOCTOR1 values(5, 'Dr. Michael Lee', '654 Maple St', 'MD');

insert into DOCTOR1 values(6, 'Dr. Laura Green', '876 Cedar Ln', 'MBBS');

insert into DOCTOR1 values(7, 'Dr. Mark Adams', '234 Oak Dr', 'MS');

insert into DOCTOR1 values(8, 'Dr. Sarah White', '987 Pine Ave', 'MD');

insert into DOCTOR1 values(9, 'Dr. Tom Clark', '543 Elm Blvd', 'MBBS');

insert into DOCTOR1 values(10, 'Dr. Olivia Harris', '678 Willow Dr', 'PhD');


select *from DOCTOR1;


CREATE TABLE PATIENT (Pid INT PRIMARY KEY, Pname VARCHAR(50) NOT NULL,
Age INT CHECK (Age >= 0),Gender VARCHAR(10) CHECK (Gender IN ('Male', 'Female',
'Other')));

insert into PATIENT values(101, 'James Wilson', 30, 'Male');

insert into PATIENT values(102, 'aditi', 20, 'Female');

insert into PATIENT values(103, 'babli', 25, 'Female');

insert into PATIENT values(104, 'chotu', 30, 'Male');

insert into PATIENT values(105, 'dhruv', 25, 'Male');
insert into PATIENT values(106, 'eshan', 28, 'Male');

insert into PATIENT values(107, 'hindvi', 24, 'Female');

insert into PATIENT values(108, 'ishani', 22, 'Female');

insert into PATIENT values(109, 'Jiyan', 27, 'Male');

insert into PATIENT values(110, 'moni', 21, 'Female');


select *from PATIENT;


DELETE FROM DOCTOR WHERE Qualification IN ('Male', 'Female');

select *from DOCTOR;

DELETE FROM PATIENT WHERE Gender IN ('Male', 'Female');

select *from PATIENT;

SELECT * FROM PATIENT WHERE Age < 40;

UPDATE PATIENT SET Pname = 'New Name' WHERE Pid = 105;

select *from PATIENT;
```

OUTPUT :-

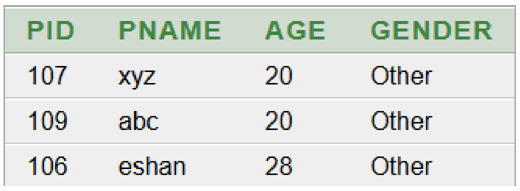


DID	DNAME	DADDRESS	QUALIFICATION
1	Dr. John Doe	jalgaon	MBBS
2	Dr. Alice Smith	456 Oak Ave	MD
3	Dr. Robert Brown	789 Pine Blvd	PhD
4	Dr. Emily White	321 Birch Rd	MDS
5	Dr. Michael Lee	654 Maple St	MD
6	Dr. Laura Green	876 Cedar Ln	MBBS
7	Dr. Mark Adams	234 Oak Dr	MS
8	Dr. Sarah White	987 Pine Ave	MD
9	Dr. Tom Clark	543 Elm Blvd	MBBS
10	Dr. Olivia Harris	678 Willow Dr	PhD

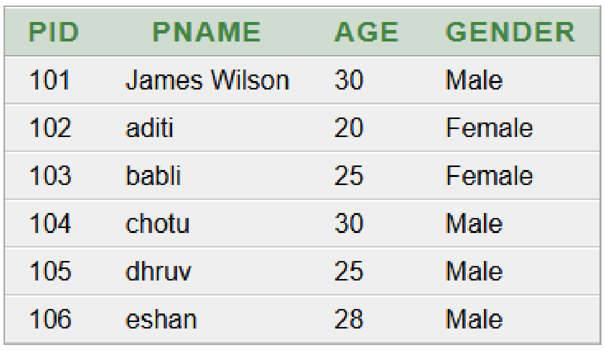
10 rows returned in 0.00 seconds CSV Export



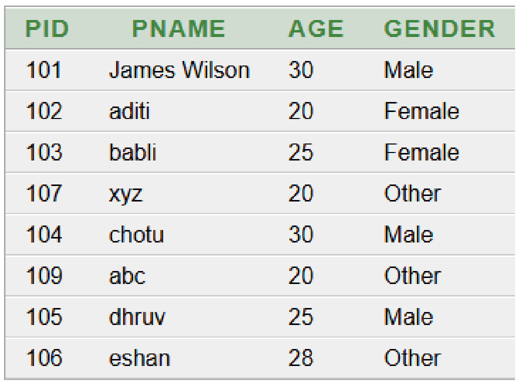
DID	DNAME	DADDRESS	QUALIFICATION
1	kruti	jalgaon	mbbs
2	pariniti	pune	md
3	sonam	mumbai	biology



PID	PNAME	AGE	GENDER
107	xyz	20	Other
109	abc	20	Other
106	eshan	28	Other



PID	PNAME	AGE	GENDER
101	James Wilson	30	Male
102	aditi	20	Female
103	babli	25	Female
104	chotu	30	Male
105	dhruv	25	Male
106	eshan	28	Male

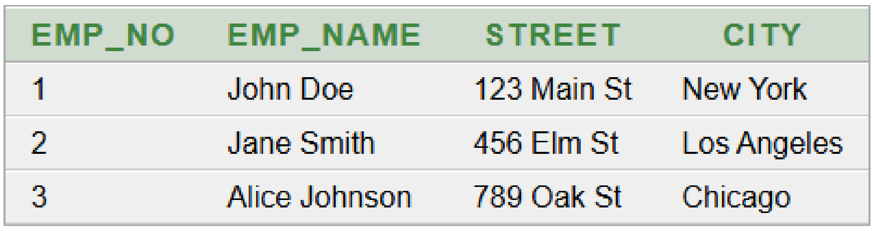


PID	PNAME	AGE	GENDER
101	James Wilson	30	Male
102	aditi	20	Female
103	babli	25	Female
107	xyz	20	Other
104	chotu	30	Male
109	abc	20	Other
105	dhruv	25	Male
106	eshan	28	Other

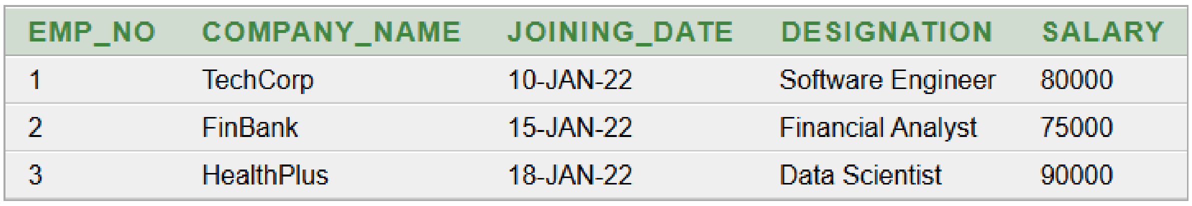
5. write a PL/SQL code to create an employee database with the table and field specified as bellow. Employee[emp no Employee name Street City] Works[Emp no Company_name_joining_date Designation Salary] Company[Emp no City] Manages [emp no Manager_name , Mang_no]

****

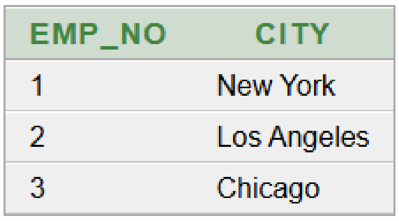
```
CREATE TABLE Employee (emp_no INT PRIMARY KEY, emp_name VARCHAR2(100) NOT NULL, street VARCHAR2(100), city VARCHAR2(100));

insert into Employee values(101 , 'pooja','123 Main St', 'New York');

insert into Employee values(102, 'Jane Smith', '456 Oak Rd', 'Los Angeles');

select *from Employee;


CREATE TABLE Works (emp_no INT,company_name VARCHAR2(100) NOT NULL, joining_date DATE NOT NULL, designation VARCHAR2(100),salary DECIMAL(10, 2),CONSTRAINT fk_emp_no FOREIGN KEY (emp_no) REFERENCES Employee(emp_no));

insert into Works values(101, 'TechCorp', '10/JAN/22' , 'Software Engineer', 75000);

insert into Works values(102, 'DataSolutions', '19/MAY/22', 'Data Analyst', 68000);

select *from Works;


CREATE TABLE Company ( emp_no INT, company_city VARCHAR2(100), CONSTRAINT fk_emp_no_company FOREIGN KEY (emp_no) REFERENCES Employee(emp_no));

insert into Company values(101, 'New York');

insert into Company values(102, 'Los Angeles');

select *from Company;


CREATE TABLE Manages (emp_no INT , manager_name VARCHAR2(100), mang_no INT, CONSTRAINT fk_emp_no_manager FOREIGN KEY (emp_no) REFERENCES Employee(emp_no));

insert into Manages values(101, 'Alice White', 201);

insert into Manages values(102, 'Tom Green', 202);

select *from manages;
```



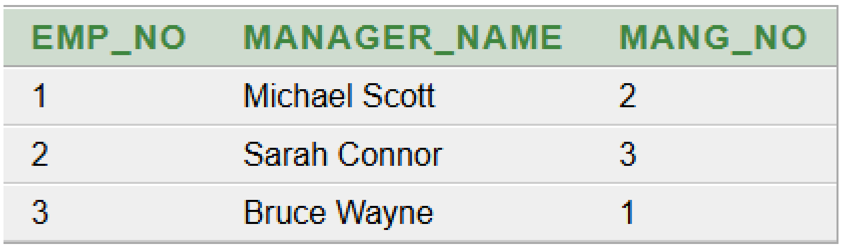
EMP_NO	EMP_NAME	STREET	CITY
1	John Doe	123 Main St	New York
2	Jane Smith	456 Elm St	Los Angeles
3	Alice Johnson	789 Oak St	Chicago



EMP_NO	COMPANY_NAME	JOINING_DATE	DESIGNATION	SALARY
1	TechCorp	10-JAN-22	Software Engineer	80000
2	FinBank	15-JAN-22	Financial Analyst	75000
3	HealthPlus	18-JAN-22	Data Scientist	90000



EMP_NO	CITY
1	New York
2	Los Angeles
3	Chicago

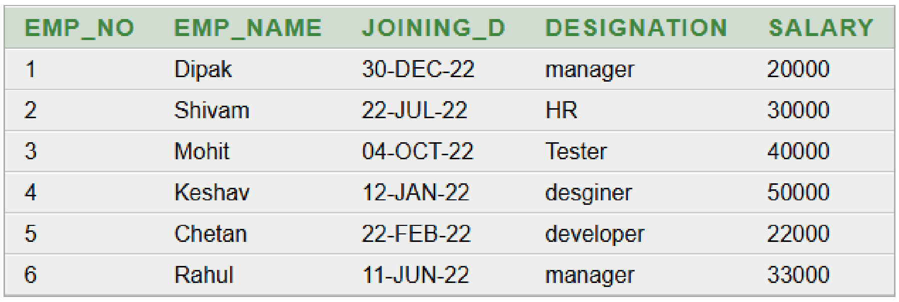


EMP_NO	MANAGER_NAME	MANG_NO
1	Michael Scott	2
2	Sarah Connor	3
3	Bruce Wayne	1

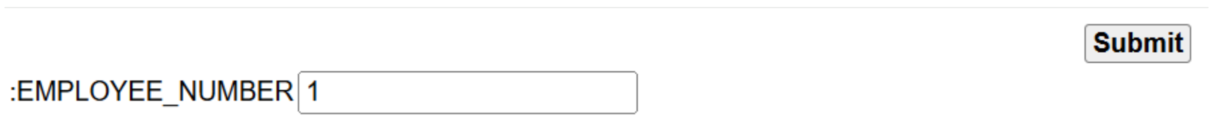
6 . PL/SQL code to retrive the employee name , join date and designation from employee database of an employee whose number is input by the user

****

```
create table employee1(emp_no number, emp_name varchar(30),joining_d date, designation
varchar(30), salary number);

insert into employee1 values(1,'Dipak','30-dec-2022','manager',20000);

insert into employee1 values(2,'Shivam','22-july-2022','HR',30000);

 insert into employee1 values(3,'Mohit','4-oct-2022','Tester',40000);

 insert into employee1 values(4,'Keshav','12-jan-2022','desginer',50000);

insert into employee1 values(5,'Chetan','22-feb-2022','developer',22000);

 insert into employee1 values(6,'Rahul','11-june-2022','manager',33000);

select*from employee1;


declare

 eno employee1.emp_no%type:= :employee_number;

 enm employee1.emp_name%type;

 joining_d employee1.joining_d%type;

ejob employee1.designation%type;


begin

select emp_name, joining_d, designation into enm , joining_d,

 ejob from employee1 where emp_no=eno;

dbms_output.put_line('employee name:'||enm);

dbms_output.put_line('joining date:'||joining_d);

 dbms_output.put_line('Designation:'||ejob);

 end;
```



EMP_NO	EMP_NAME	JOINING_D	DESIGNATION	SALARY
1	Dipak	30-DEC-22	manager	20000
2	Shivam	22-JUL-22	HR	30000
3	Mohit	04-OCT-22	Tester	40000
4	Keshav	12-JAN-22	desginer	50000
5	Chetan	22-FEB-22	developer	22000
6	Rahul	11-JUN-22	manager	33000



:EMPLOYEE_NUMBER 1

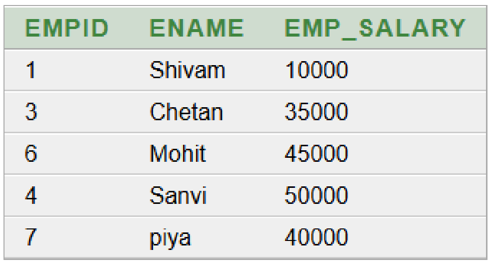
```
employee name:Dipak
joining date:30-DEC-22
Designation:manager

Statement processed.
```

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

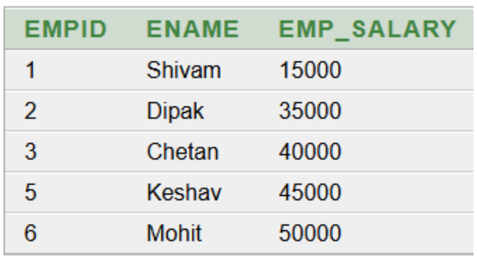
****

```
create table emp9(empid number, ename varchar(30), emp_salary number);
 insert into emp9 values(1,'Shivam',10000);
 insert into emp9 values(2,'Dipak',30000);
insert into emp9 values(3,'Chetan',35000);
 insert into emp9 values(5,'Keshav',40000);
insert into emp9 values(6,'Mohit',45000);
select*from emp9;


DECLARE
   total_rows number(2);
BEGIN
   UPDATE  emp9
   SET emp_salary = emp_salary + 5000;
   IF sql%notfound
THEN
 dbms_output.put_line('no employee salary updated');
 ELSIF sql%found
THEN
 total_rows := sql%rowcount;
 dbms_output.put_line( total_rows || ' salary updated ');
   END IF;
END;
```

OUTPUT :-



EMPID	ENAME	EMP_SALARY
1	Shivam	10000
3	Chetan	35000
6	Mohit	45000
4	Sanvi	50000
7	piya	40000



EMPID	ENAME	EMP_SALARY
1	Shivam	15000
2	Dipak	35000
3	Chetan	40000
5	Keshav	45000
6	Mohit	50000

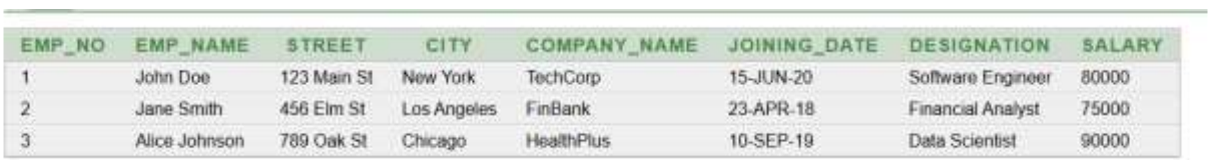
8. Write a row trigger to insert the existing values of the salary table in to a new table when the salary table is updated.

****

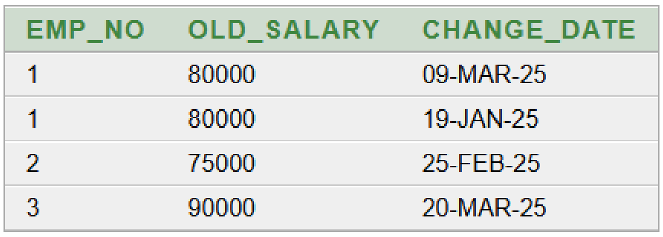
```
CREATE TABLE EmployeeDetails (emp_no NUMBER PRIMARY KEY,
   emp_name VARCHAR2(100),street VARCHAR2(100),
    city VARCHAR2(50),company_name VARCHAR2(100),
   joining_date DATE,designation VARCHAR2(50),salary NUMBER(10,2));


INSERT INTO EmployeeDetails VALUES (1, 'John Doe', '123 Main St', 'New York', 'TechCorp',
TO_DATE('2020-06-15', 'YYYY-MM-DD'), 'Software Engineer', 80000);

INSERT INTO EmployeeDetails VALUES (2, 'Jane Smith', '456 Elm St', 'Los Angeles', 'FinBank',
TO_DATE('2018-04-23', 'YYYY-MM-DD'), 'Financial Analyst', 75000);

INSERT INTO EmployeeDetails VALUES (3, 'Alice Johnson', '789 Oak St', 'Chicago', 'HealthPlus',
TO_DATE('2019-09-10', 'YYYY-MM-DD'), 'Data Scientist', 90000);


select *from EmployeeDetails;


CREATE TABLE SalaryHistory( emp_no NUMBER, old_salary NUMBER(10,2),
change_date DATE,CONSTRAINT fk_salaryhistory_employee FOREIGN KEY (emp_no)
REFERENCES EmployeeDetails(emp_no));


INSERT INTO SalaryHistory VALUES (1, 80000, '19/JAN/25');
INSERT INTO SalaryHistory VALUES (2, 75000, '25/FEB/25');
INSERT INTO SalaryHistory VALUES (3, 90000, '20/MAR/25');


select *from SalaryHistory;
UPDATE EmployeeDetails SET salary = 85000 WHERE emp_no = 1;


-- Create Trigger to Capture Salary Changes
CREATE OR REPLACE TRIGGER trg_salary_update
BEFORE UPDATE ON EmployeeDetails
FOR EACH ROW
BEGIN
```

```
    INSERT INTO SalaryHistory (emp_no, old_salary, change_date)
    VALUES (:OLD.emp_no, :OLD.salary, SYSDATE);
END;
/
```

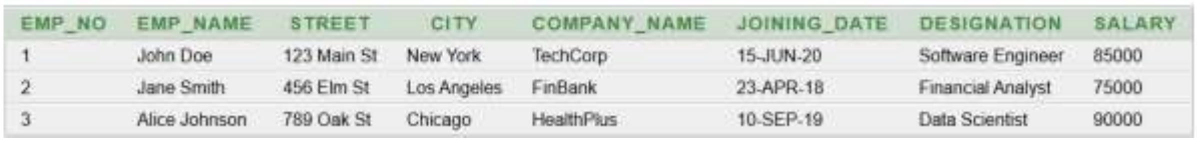
OUTPUT :



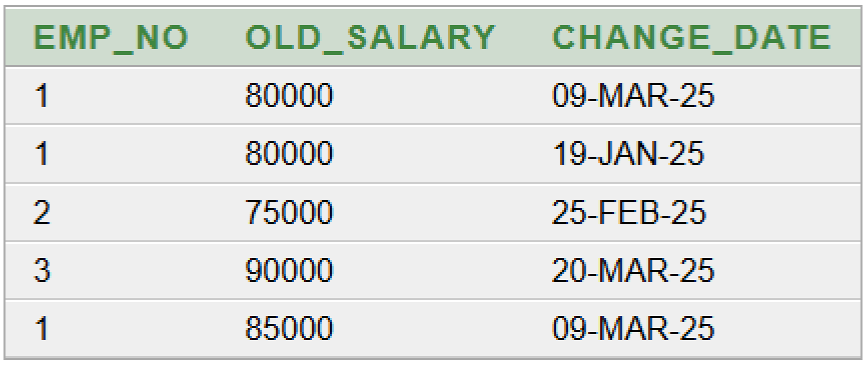
EMP_NO	EMP_NAME	STREET	CITY	COMPANY_NAME	JOINING_DATE	DESIGNATION	SALARY
1	John Doe	123 Main St	New York	TechCorp	15-JUN-20	Software Engineer	80000
2	Jane Smith	456 Elm St	Los Angeles	FinBank	23-APR-18	Financial Analyst	75000
3	Alice Johnson	789 Oak St	Chicago	HealthPlus	10-SEP-19	Data Scientist	90000



EMP_NO	OLD_SALARY	CHANGE_DATE
1	80000	09-MAR-25
1	80000	19-JAN-25
2	75000	25-FEB-25
3	90000	20-MAR-25



EMP_NO	EMP_NAME	STREET	CITY	COMPANY_NAME	JOINING_DATE	DESIGNATION	SALARY
1	John Doe	123 Main St	New York	TechCorp	15-JUN-20	Software Engineer	85000
2	Jane Smith	456 Elm St	Los Angeles	FinBank	23-APR-18	Financial Analyst	75000
3	Alice Johnson	789 Oak St	Chicago	HealthPlus	10-SEP-19	Data Scientist	90000



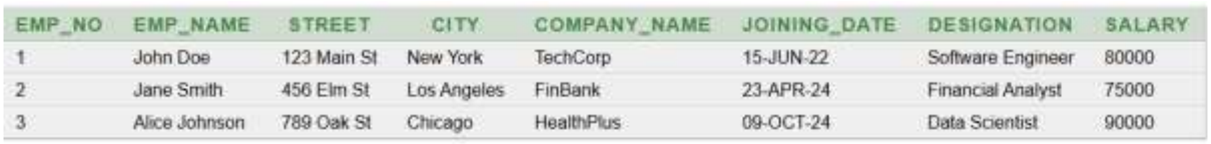
EMP_NO	OLD_SALARY	CHANGE_DATE
1	80000	09-MAR-25
1	80000	19-JAN-25
2	75000	25-FEB-25
3	90000	20-MAR-25
1	85000	09-MAR-25

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.

****

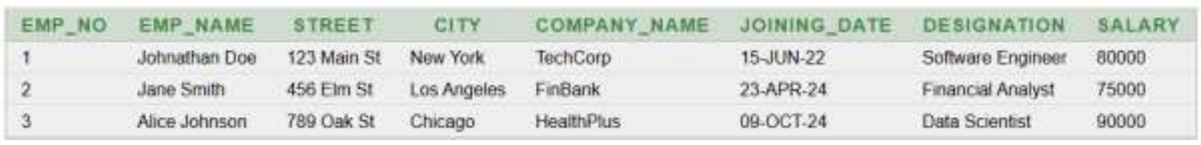
```
CREATE TABLE StaffRecords ( emp_no NUMBER PRIMARY KEY, emp_name VARCHAR2(100),
    street VARCHAR2(100),city VARCHAR2(50),company_name VARCHAR2(100),
    joining_date DATE,designation VARCHAR2(50),salary NUMBER(10,2));


INSERT INTO StaffRecords VALUES (1, 'John Doe', '123 Main St', 'New York', 'TechCorp',
'15/JUN/22', 'Software Engineer', 80000);
INSERT INTO StaffRecords VALUES (2, 'Jane Smith', '456 Elm St', 'Los Angeles', 'FinBank',
'23/APRIL/24', 'Financial Analyst', 75000);
INSERT INTO StaffRecords VALUES (3, 'Alice Johnson', '789 Oak St', 'Chicago', 'HealthPlus',
'9/OCT/24', 'Data Scientist', 90000);


select *from StaffRecords;
UPDATE StaffRecords SET emp_name = 'Johnathan Doe' WHERE emp_no = 1;


-- Create Trigger to Capture Name Changes
CREATE OR REPLACE TRIGGER trg_emp_name_update
BEFORE UPDATE OF emp_name ON StaffRecords
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('Employee Name Changed: Old Value = ' || :OLD.emp_name || ', New
Value = ' || :NEW.emp_name);
END;
/
```

OUTPUT



EMP_NO	EMP_NAME	STREET	CITY	COMPANY_NAME	JOINING_DATE	DESIGNATION	SALARY
1	John Doe	123 Main St	New York	TechCorp	15-JUN-22	Software Engineer	80000
2	Jane Smith	456 Elm St	Los Angeles	FinBank	23-APR-24	Financial Analyst	75000
3	Alice Johnson	789 Oak St	Chicago	HealthPlus	09-OCT-24	Data Scientist	90000

```
Employee Name Changed: Old Value = John Doe, New Value = Johnathan Doe
```

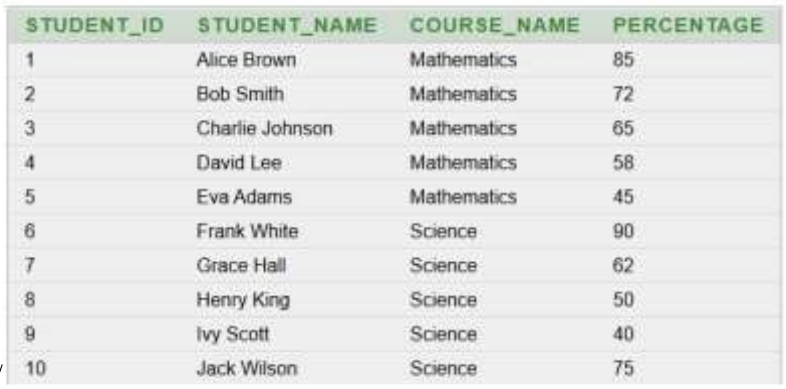


EMP_NO	EMP_NAME	STREET	CITY	COMPANY_NAME	JOINING_DATE	DESIGNATION	SALARY
1	Johnathan Doe	123 Main St	New York	TechCorp	15-JUN-22	Software Engineer	80000
2	Jane Smith	456 Elm St	Los Angeles	FinBank	23-APR-24	Financial Analyst	75000
3	Alice Johnson	789 Oak St	Chicago	HealthPlus	09-OCT-24	Data Scientist	90000

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 Student_Course Table

CREATE TABLE student_course ( student_id NUMBER PRIMARY KEY, student_name
VARCHAR2(100), course_name VARCHAR2(100), percentage NUMBER(5,2));


-- Insert Sample Data

INSERT INTO student_course VALUES (1, 'Alice Brown', 'Mathematics', 85);

INSERT INTO student_course VALUES (2, 'Bob Smith', 'Mathematics', 72);

INSERT INTO student_course VALUES (3, 'Charlie Johnson', 'Mathematics', 65);

INSERT INTO student_course VALUES (4, 'David Lee', 'Mathematics', 58);

INSERT INTO student_course VALUES (5, 'Eva Adams', 'Mathematics', 45);

INSERT INTO student_course VALUES (6, 'Frank White', 'Science', 90);

INSERT INTO student_course VALUES (7, 'Grace Hall', 'Science', 62);

INSERT INTO student_course VALUES (8, 'Henry King', 'Science', 50);

INSERT INTO student_course VALUES (9, 'Ivy Scott', 'Science', 40);

INSERT INTO student_course VALUES (10, 'Jack Wilson', 'Science', 75);


select *from student_course;

-- Create Procedure to Count Students in Percentage Ranges

CREATE OR REPLACE PROCEDURE Count_Students (p_course_name IN VARCHAR2)

IS

   v_high NUMBER := 0;

   v_mid NUMBER := 0;

   v_low NUMBER := 0;

   v_fail NUMBER := 0;

BEGIN

   SELECT

      COUNT(CASE WHEN percentage >= 70 THEN 1 END),

      COUNT(CASE WHEN percentage BETWEEN 60 AND 69 THEN 1 END),

      COUNT(CASE WHEN percentage BETWEEN 50 AND 59 THEN 1 END),
      COUNT(CASE WHEN percentage < 50 THEN 1 END)
   INTO v_high, v_mid, v_low, v_fail
   FROM student_course
   WHERE course_name = p_course_name;

   DBMS_OUTPUT.PUT_LINE('Course: ' || p_course_name);
   DBMS_OUTPUT.PUT_LINE('70% and above: ' || v_high);
   DBMS_OUTPUT.PUT_LINE('60-69%: ' || v_mid);
   DBMS_OUTPUT.PUT_LINE('50-59%: ' || v_low);
   DBMS_OUTPUT.PUT_LINE('Below 50%: ' || v_fail);
END;
/
```



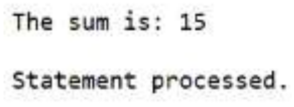
STUDENT_ID	STUDENT_NAME	COURSE_NAME	PERCENTAGE
1	Alice Brown	Mathematics	85
2	Bob Smith	Mathematics	72
3	Charlie Johnson	Mathematics	65
4	David Lee	Mathematics	58
5	Eva Adams	Mathematics	45
6	Frank White	Science	90
7	Grace Hall	Science	62
8	Henry King	Science	50
9	Ivy Scott	Science	40
10	Jack Wilson	Science	75

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

****

```
-- Function to add two numbers in PL/SQL
CREATE OR REPLACE FUNCTION add_numbers(
   num1 IN NUMBER,
   num2 IN NUMBER
) RETURN NUMBER IS
   result NUMBER;
BEGIN
   result := num1 + num2;
   RETURN result;
END add_numbers;
/

-- Calling the function and storing the result
DECLARE
   sum_result NUMBER;
BEGIN
   sum_result := add_numbers(5, 10);
   DBMS_OUTPUT.PUT_LINE('The sum is: ' || sum_result);
END;
/
```

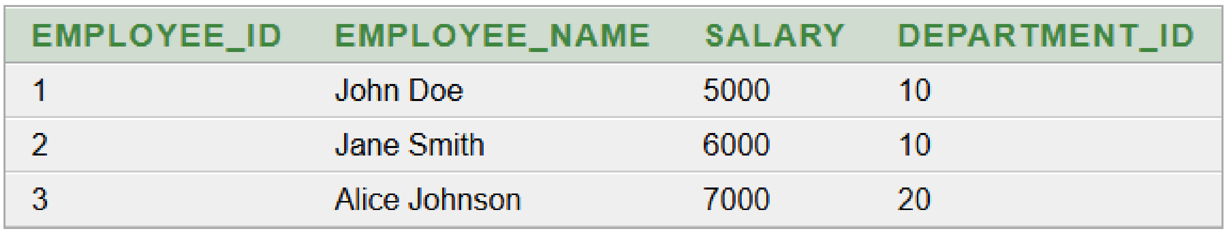
OUTPUT


```
The sum is: 15
Statement processed.
```

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 staff table

CREATE TABLE staff (   employee_id NUMBER PRIMARY KEY,   employee_name VARCHAR2(100), salary NUMBER, department_id NUMBER);

INSERT INTO staff VALUES (1, 'John Doe', 5000, 10);

INSERT INTO staff VALUES (2, 'Jane Smith', 6000, 10);

INSERT INTO staff VALUES (3, 'Alice Johnson', 7000, 20);


select *from staff;


-- Function to calculate total salary of a department in PL/SQL

CREATE OR REPLACE FUNCTION get_total_salary(

   dept_no IN NUMBER

) RETURN NUMBER IS

   total_salary NUMBER := 0;

BEGIN

   SELECT SUM(salary) INTO total_salary FROM staff WHERE department_id = dept_no;

   RETURN total_salary;

END get_total_salary;

/

-- Calling the function and displaying the result

DECLARE

   dept_salary NUMBER;

BEGIN

   dept_salary := get_total_salary(10);

   DBMS_OUTPUT.PUT_LINE('Total salary for department 10: ' || dept_salary);

END;

/
```



EMPLOYEE_ID	EMPLOYEE_NAME	SALARY	DEPARTMENT_ID
1	John Doe	5000	10
2	Jane Smith	6000	10
3	Alice Johnson	7000	20

```
Total salary for department 10: 11000

Statement processed.
```

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.

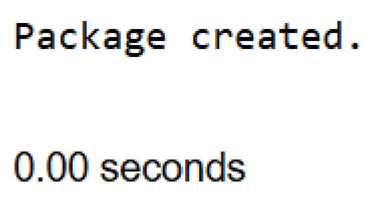
****

```
-- Creating Student Table
create table student ( student_id number primary key, name varchar2(100), age number, course varchar2(100));


-- Inserting Sample Data
insert into student (student_id, name, age, course) values (1, 'john doe', 20, 'computer science');
insert into student (student_id, name, age, course) values (2, 'jane smith', 22, 'mathematics');
insert into student (student_id, name, age, course) values (3, 'robert brown', 21, 'physics');
commit;
select *from student;


create or replace package student_pkg as
   procedure insert_student(p_id number, p_name varchar2, p_age number, p_course varchar2);
   procedure update_student(p_id number, p_name varchar2, p_age number, p_course varchar2);
   procedure delete_student(p_id number);
   procedure get_student(p_id number);
end student_pkg;
/


-- Package Body
create or replace package body student_pkg as
```

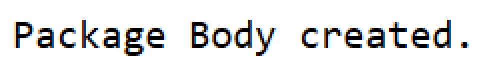
```
   procedure insert_student(p_id number, p_name varchar2, p_age number, p_course
varchar2) is
   begin
      insert into student (id, name, age, course)
      values (p_id, p_name, p_age, p_course);
      commit;
   end insert_student;

   procedure update_student(p_id number, p_name varchar2, p_age number, p_course
varchar2) is
   begin
      update student
      set name = p_name, age = p_age, course = p_course
      where id = p_id;
      commit;
   end update_student;

   procedure delete_student(p_id number) is
   begin
      delete from student where id = p_id;
      commit;
   end delete_student;

   procedure get_student(p_id number) is
      v_name student.name%type;
      v_age student.age%type;
      v_course student.course%type;
   begin
      select name, age, course into v_name, v_age, v_course
```

```
        from student where id = p_id;

        dbms_output.put_line('id: ' || p_id || ', name: ' || v_name || ', age: ' || v_age || ', course:
' || v_course);

    end get_student;

end student_pkg;

/
```

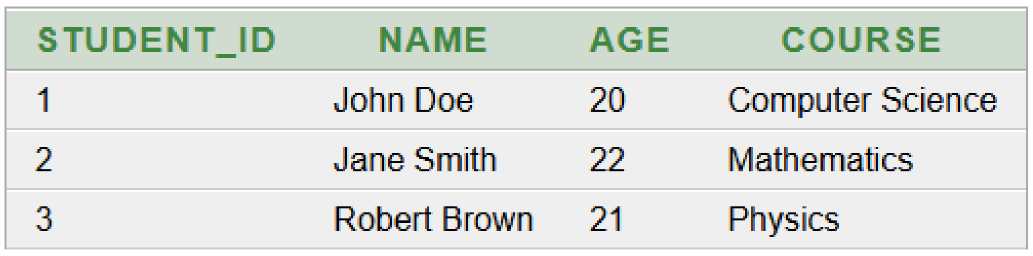
OUTPUT :



```
Package created.

0.00 seconds
```



```
Package Body created.
```



STUDENT_ID	NAME	AGE	COURSE
1	John Doe	20	Computer Science
2	Jane Smith	22	Mathematics
3	Robert Brown	21	Physics

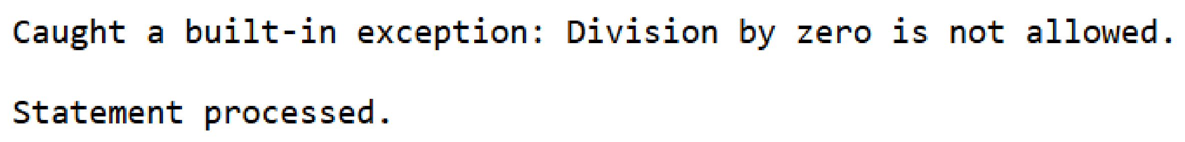
14. Write a program to illustrate user-defined exceptions, built-in exceptions, and raise application error exceptions.

****

```
declare
   -- user-defined exception
   negative_value exception;
   pragma exception_init(negative_value, -20001);

   -- built-in exception (for division by zero)
   v_divisor number := 0;

   -- variable to test exceptions
   v_value number := -5;
begin
   -- handling built-in exception (zerodivisionerror equivalent)
   if v_divisor = 0 then
      raise zero_divide;
   end if;

   -- handling user-defined exception
   if v_value < 0 then
      raise negative_value;
   end if;

   dbms_output.put_line('valid input received: ' || v_value);

exception
   when zero_divide then
```

```
        dbms_output.put_line('caught a built-in exception: division by zero is not
allowed.');

    when negative_value then
        dbms_output.put_line('caught a user-defined exception: negative values are not
allowed.');

    when others then
        dbms_output.put_line('an unexpected error occurred: ' || sqlerrm);
end;
```

OUTPUT:



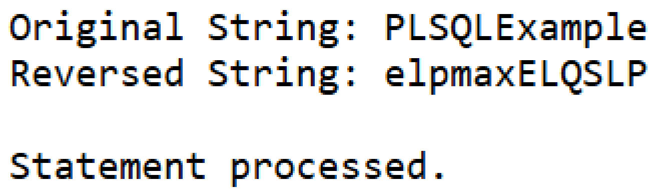
```
Caught a built-in exception: Division by zero is not allowed.

Statement processed.
```

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

****

```
DECLARE
   v_input_string  VARCHAR2(100) := 'PLSQLExample'; -- Input string
   v_reversed_string VARCHAR2(100) := '';
   v_length NUMBER;
BEGIN
   -- Get the length of the input string
   v_length := LENGTH(v_input_string);

   -- Loop through the string in reverse order
   FOR i IN REVERSE 1..v_length LOOP
      v_reversed_string := v_reversed_string || SUBSTR(v_input_string, i, 1);
   END LOOP;

   -- Output the reversed string
   DBMS_OUTPUT.PUT_LINE('Original String: ' || v_input_string);
   DBMS_OUTPUT.PUT_LINE('Reversed String: ' || v_reversed_string);
END;
```

OUTPUT:



```
Original String: PLSQLExample
Reversed String: elpmaxELQSLP

Statement processed.
```

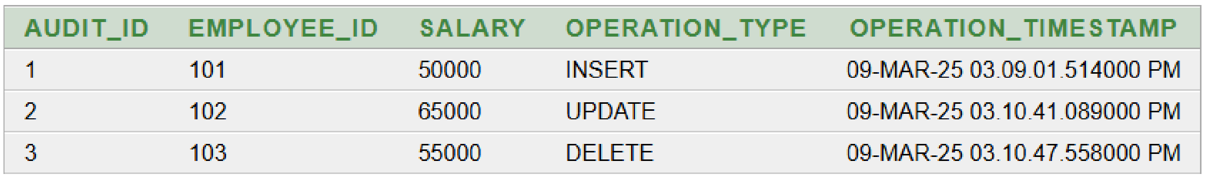
16. Trigger for Auditing Table Changes

- **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.**

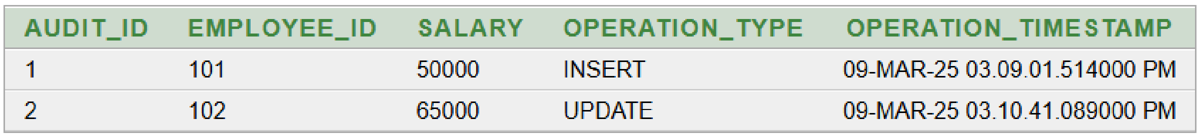
****

```
create table employees_audit1 ( audit_id number primary key, employee_id number not null, salary number , operation_type varchar2(10) not null, operation_timestamp timestamp default systimestamp not null);


insert into employees_audit1 values (1, 101, 50000, 'insert');

insert into employees_audit1 (audit_id, employee_id, salary, operation_type)

values (2, 102, 60000, 'update');


insert into employees_audit1 (audit_id, employee_id, salary, operation_type)

values (3, 103, 55000, 'delete');


update employees_audit1

set salary = 65000, operation_type = 'update'

where employee_id = 102;


delete from employees_audit1

where employee_id = 103;


select * from employees_audit1;

select * from employees_audit;


create sequence employees_audit_seq

start with 1

increment by 1

nocache
```

```
nocycle;


-- create a trigger to assign next sequence value to audit_id
create or replace trigger employees_audit_trg
before insert on employees_audit
for each row
begin
   select employees_audit_seq.nextval into :new.audit_id from dual;
end;
/
```



AUDIT_ID	EMPLOYEE_ID	SALARY	OPERATION_TYPE	OPERATION_TIMESTAMP
1	101	50000	INSERT	09-MAR-25 03.09.01.514000 PM
2	102	65000	UPDATE	09-MAR-25 03.10.41.089000 PM
3	103	55000	DELETE	09-MAR-25 03.10.47.558000 PM



AUDIT_ID	EMPLOYEE_ID	SALARY	OPERATION_TYPE	OPERATION_TIMESTAMP
1	101	50000	INSERT	09-MAR-25 03.09.01.514000 PM
2	102	65000	UPDATE	09-MAR-25 03.10.41.089000 PM

17. Employee Bonus Calculation Using Cursor

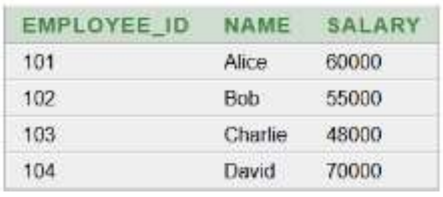
- **Write a PL/SQL program using an explicit cursor to calculate and display a 10% bonus for all employees whose salary is greater than 50,000. Assume a table EMPLOYEES with columns EMPLOYEE_ID, Name, and Salary.**

****

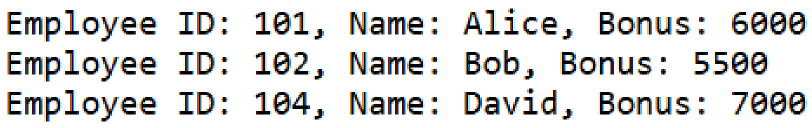
```
create table employee_info (
   employee_id number primary key,
   name varchar2(100),
   salary number(10,2)
);

insert into employee_info values (101, 'alice', 60000);
insert into employee_info  values (102, 'bob', 55000);
insert into employee_info  values (103, 'charlie', 48000);
insert into employee_info  values (104, 'david', 70000);

select *from employee_info;
commit;

declare
   -- declare cursor
   cursor emp_cursor is
      select employee_id, name, salary from employee_info where salary > 50000;

   -- declare variables to hold employee details
   v_emp_id employee_info.employee_id%type;
   v_name employee_info.name%type;
   v_salary employee_info.salary%type;
```

```
    v_bonus number(10,2);
begin
    -- open the cursor
    open emp_cursor;

    loop
        -- fetch employee data
        fetch emp_cursor into v_emp_id, v_name, v_salary;
        exit when emp_cursor%notfound;

        -- calculate bonus (10% of salary)
        v_bonus := v_salary * 0.10;

        -- display the result
        dbms_output.put_line('employee id: ' || v_emp_id || ', name: ' || v_name || ', bonus: ' || v_bonus);
    end loop;

    -- close the cursor
    close emp_cursor;
end;
```

OUTPUT :



EMPLOYEE_ID	NAME	SALARY
101	Alice	60000
102	Bob	55000
103	Charlie	48000
104	David	70000



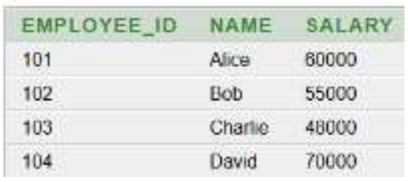
```
Employee ID: 101, Name: Alice, Bonus: 6000
Employee ID: 102, Name: Bob, Bonus: 5500
Employee ID: 104, Name: David, Bonus: 7000
```

18. Write a SQL Program to implement Aggregate Functions.

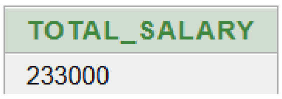
****

```
create table employeetable ( employee_id number primary key, name varchar2(100),
  salary number(10,2));

insert into employeetable  values (101, 'alice', 60000);
insert into employeetable  values (102, 'bob', 55000);
insert into employeetable  values (103, 'charlie', 48000);
insert into employeetable  values (104, 'david', 70000);

select *from employeetable;
commit;

-- aggregate functions
select count(*) as total_employees from employeetable;
select avg(salary) as average_salary from employeetable;
select sum(salary) as total_salary from employeetable;
select max(salary) as highest_salary from employeetable;
select min(salary) as lowest_salary from employeetable;
```

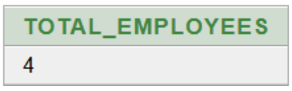
OUTPUT:



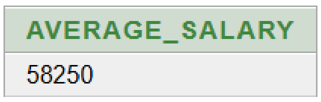
EMPLOYEE_ID	NAME	SALARY
101	Alice	60000
102	Bob	55000
103	Charlie	48000
104	David	70000



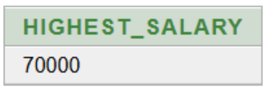
TOTAL_SALARY
233000



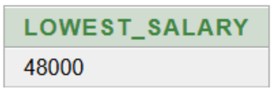
TOTAL_EMPLOYEES
4



AVERAGE_SALARY
58250



HIGHEST_SALARY
70000



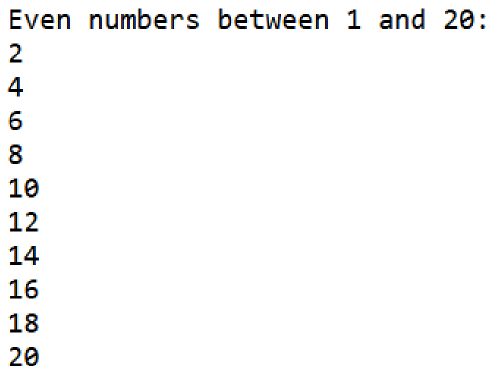
LOWEST_SALARY
48000

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

****

```
declare
   v_start number := 1; -- starting number
   v_end number := 20;  -- ending number
begin
   dbms_output.put_line('even numbers between ' || v_start || ' and ' || v_end || ':');

   for i in v_start..v_end loop
      if mod(i, 2) = 0 then
         dbms_output.put_line(i);
      end if;
   end loop;
end;
```

OUTPUT:



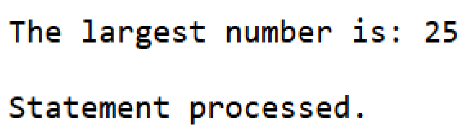
```
Even numbers between 1 and 20:
2
4
6
8
10
12
14
16
18
20
```

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

****

```
declare
   num1 number := 15;
   num2 number := 25;
   num3 number := 10;
   largest number;
begin
   if num1 >= num2 and num1 >= num3 then
      largest := num1;
   elsif num2 >= num1 and num2 >= num3 then
      largest := num2;
   else
      largest := num3;
   end if;

   dbms_output.put_line('the largest number is: ' || largest);
end;
```

OUTPUT:-



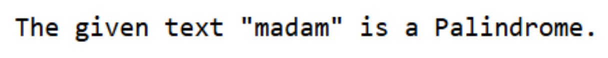
```
The largest number is: 25

Statement processed.
```

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

****

```
declare
   v_text varchar2(100);
   v_reversed_text varchar2(100) := '';
   v_length number;
begin
   -- accepting input
   v_text := 'madam'; -- you can replace it with any input
   v_length := length(v_text);

   -- reversing the text
   for i in reverse 1..v_length loop
      v_reversed_text := v_reversed_text || substr(v_text, i, 1);
   end loop;

   -- checking if the original text is equal to the reversed text
   if v_text = v_reversed_text then
      dbms_output.put_line('the given text "' || v_text || '" is a palindrome.');
   else
      dbms_output.put_line('the given text "' || v_text || '" is not a palindrome.');
   end if;
end;
```

OUTPUT:



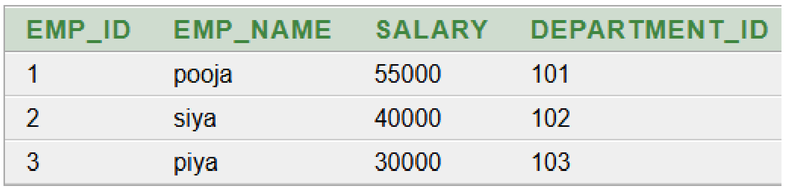
```
The given text "madam" is a Palindrome.
```

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

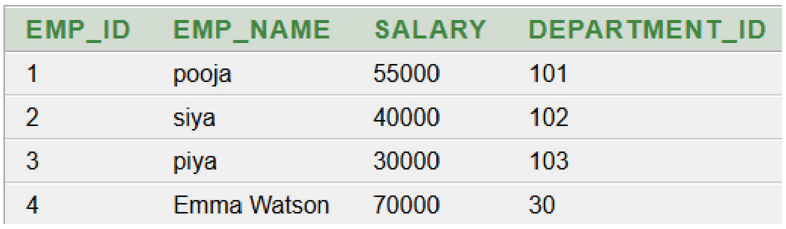
****

```
create table employees (emp_id      number primary key,  emp_name     varchar2(100),
   salary       number(10,2),   department_id number);
insert into employees values(1,'pooja',50000,101);
insert into employees values(2,'siya',40000,102);
insert into employees values(3,'piya',30000,103);


select *from employees;
declare
   v_emp_id number := 4;
   v_emp_name varchar2(100) := 'emma watson';
   v_salary number := 70000;
   v_department_id number := 30;
begin
   insert into employees (emp_id, emp_name, salary, department_id)
   values (v_emp_id, v_emp_name, v_salary, v_department_id);


   dbms_output.put_line('record inserted successfully.');
   commit;
end;
/
```

OUTPUT:



EMP_ID	EMP_NAME	SALARY	DEPARTMENT_ID
1	pooja	55000	101
2	siya	40000	102
3	piya	30000	103



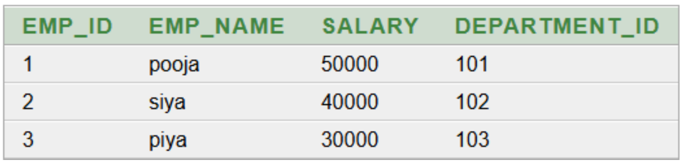
EMP_ID	EMP_NAME	SALARY	DEPARTMENT_ID
1	pooja	55000	101
2	siya	40000	102
3	piya	30000	103
4	Emma Watson	70000	30

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

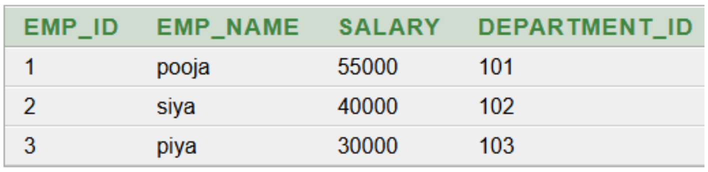
****

```
create table employees (emp_id      number primary key,   emp_name      varchar2(100),
   salary       number(10,2),  department_id number);
insert into employees values(1,'pooja',50000,101);
insert into employees values(2,'siya',40000,102);
insert into employees values(3,'piya',30000,103);

select *from employees;
declare
   v_dept_id number := 101;  -- update employees in this department
   v_increment number := 5000;  -- salary increment amount
begin
   -- implicit cursor for updating salary
   update employees
   set salary = salary + v_increment
   where department_id = v_dept_id;

   -- display number of rows updated
   dbms_output.put_line(sql%rowcount || ' record(s) updated.');
   commit; -- save changes
end;
/
```

OUTPUT:



EMP_ID	EMP_NAME	SALARY	DEPARTMENT_ID
1	pooja	50000	101
2	siya	40000	102
3	piya	30000	103



EMP_ID	EMP_NAME	SALARY	DEPARTMENT_ID
1	pooja	55000	101
2	siya	40000	102
3	piya	30000	103

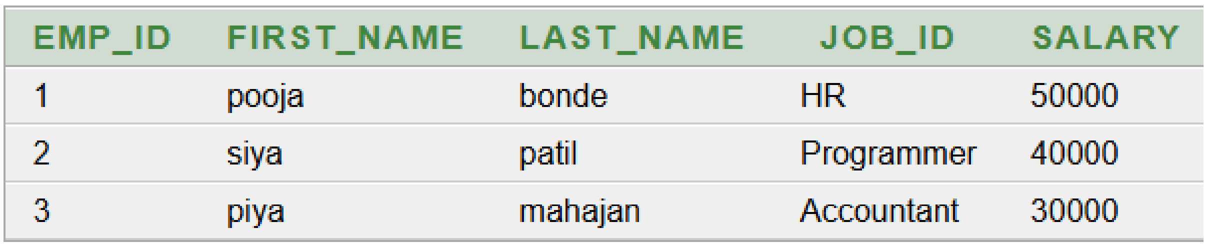
24. Write PL/SQL code to display Employee detail using explicit cursor.

****

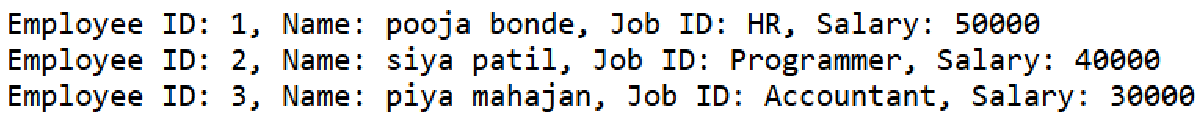
```
create table emp (emp_id      number primary key,  first_name  varchar2(50),
last_name   varchar2(50), job_id      varchar2(20),  salary      number(10,2) );
insert into emp values(1,'pooja','bonde','hr',50000);
insert into emp values(2,'siya','patil','programmer',40000);
insert into emp values(3,'piya','mahajan','accountant',30000);

select *from emp;
declare
   -- declare an explicit cursor for selecting employee details
   cursor emp_cursor is
      select emp_id, first_name, last_name, job_id, salary from emp;

   -- declare variables to hold fetched data
   v_emp_id     emp.emp_id%type;
   v_first_name emp.first_name%type;
   v_last_name  emp.last_name%type;
   v_job_id     emp.job_id%type;
   v_salary     emp.salary%type;

begin
   -- open the cursor
   open emp_cursor;

   loop
      -- fetch data into variables
      fetch emp_cursor into v_emp_id, v_first_name, v_last_name, v_job_id, v_salary;
```

```
        -- exit loop when no more records are found
        exit when emp_cursor%notfound;

        -- display employee details
        dbms_output.put_line('employee id: ' || v_emp_id ||
                             ', name: ' || v_first_name || ' ' || v_last_name ||
                             ', job id: ' || v_job_id ||
                             ', salary: ' || v_salary);
    end loop;

    -- close the cursor
    close emp_cursor;
end;
/
```

OUTPUT:



EMP_ID	FIRST_NAME	LAST_NAME	JOB_ID	SALARY
1	pooja	bonde	HR	50000
2	siya	patil	Programmer	40000
3	piya	mahajan	Accountant	30000



```
Employee ID: 1, Name: pooja bonde, Job ID: HR, Salary: 50000
Employee ID: 2, Name: siya patil, Job ID: Programmer, Salary: 40000
Employee ID: 3, Name: piya mahajan, Job ID: Accountant, Salary: 30000
```

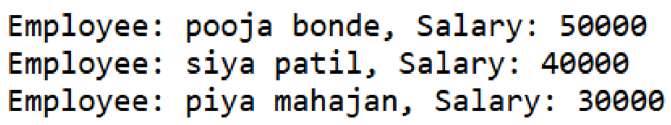
25. Write PL/SQL code in cursor to display employee names and salary.

****

```
create table empp(emp_id      number primary key,  first_name  varchar2(50)
 last_name   varchar2(50), salary      number(10,2) );

insert into empp values(1,'pooja','bonde',800000);
insert into empp values(2,'reyansh','bonde',500000);
insert into empp values(3,'khushi','mahajan',70000);

declare
   -- declare an explicit cursor to fetch employee names and salary
   cursor emp_cursor is
      select first_name, last_name, salary from emp;

   -- declare variables to hold the fetched data
   v_first_name emp.first_name%type;
   v_last_name  emp.last_name%type;
   v_salary     emp.salary%type;

begin
   -- open the cursor
   open emp_cursor;

   loop
      -- fetch data into variables
      fetch emp_cursor into v_first_name, v_last_name, v_salary;

      -- exit loop when no more records
      exit when emp_cursor%notfound;
```

```
        -- display employee names and salary
        dbms_output.put_line('employee: ' || v_first_name || ' ' || v_last_name ||
                             ', salary: ' || v_salary);
    end loop;

    -- close the cursor
    close emp_cursor;
end;
/
```

OUTPUT :



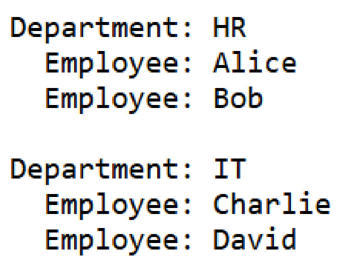
```
Employee: pooja bonde, Salary: 50000
Employee: siya patil, Salary: 40000
Employee: piya mahajan, Salary: 30000
```

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

****

```
-- Create DEPARTMENT table

CREATE TABLE DEPARTMENT28 (dept_id    NUMBER PRIMARY KEY, dept_name VARCHAR2(50));


-- Create EMPLOYEE table

CREATE TABLE EMPLOYEE28 (emp_id    NUMBER PRIMARY KEY, emp_name VARCHAR2(100), dept_id  NUMBER REFERENCES DEPARTMENT28(dept_id));


-- Insert data into DEPARTMENT table

INSERT INTO DEPARTMENT28 VALUES (1, 'HR');

INSERT INTO DEPARTMENT28 VALUES (2, 'IT');


-- Insert data into EMPLOYEE table

INSERT INTO EMPLOYEE28 VALUES (101, 'Alice', 1);

INSERT INTO EMPLOYEE28 VALUES (102, 'Bob', 1);

INSERT INTO EMPLOYEE28 VALUES (201, 'Charlie', 2);

INSERT INTO EMPLOYEE28 VALUES (202, 'David', 2);


-- Commit the changes

COMMIT;

select *from DEPARTMENT28;

select *from EMPLOYEE28;


DECLARE

  -- Cursor for Department

  CURSOR dept_cursor IS

    SELECT dept_id, dept_name FROM DEPARTMENT28;


  -- Cursor for Employee (Parameterized Cursor)

  CURSOR emp_cursor (p_dept_id NUMBER) IS

    SELECT emp_name FROM EMPLOYEE28 WHERE dept_id = p_dept_id;
    v_dept_id NUMBER;
    v_dept_name VARCHAR2(50);
    v_emp_name VARCHAR2(100);

BEGIN
    OPEN dept_cursor;
    LOOP
        FETCH dept_cursor INTO v_dept_id, v_dept_name;
        EXIT WHEN dept_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('Department: ' || v_dept_name);

        OPEN emp_cursor(v_dept_id);
        LOOP
            FETCH emp_cursor INTO v_emp_name;
            EXIT WHEN emp_cursor%NOTFOUND;

            DBMS_OUTPUT.PUT_LINE('  Employee: ' || v_emp_name);
        END LOOP;
        CLOSE emp_cursor;

        DBMS_OUTPUT.PUT_LINE(' ');
    END LOOP;
    CLOSE dept_cursor;
END;
/
```

OUTPUT:

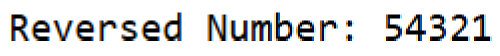


```
Department: HR
  Employee: Alice
  Employee: Bob

Department: IT
  Employee: Charlie
  Employee: David
```

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

****

```
DECLARE
   v_input NUMBER := 12345;
   v_output NUMBER;
BEGIN
   -- Call the procedure to reverse the number
   reverse_number(v_input, v_output);

   -- Display the reversed number
   DBMS_OUTPUT.PUT_LINE('Reversed Number: ' || v_output);
END;
/
```



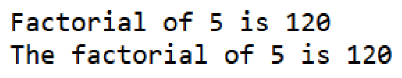
```
Reversed Number: 54321
```

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

****

```
DECLARE
   v_input NUMBER := 5;      -- Change this number to find factorial of a different number
   v_output NUMBER;
BEGIN
   -- Call the procedure
   find_factorial(v_input, v_output);

   -- Display the result
   DBMS_OUTPUT.PUT_LINE('The factorial of ' || v_input || ' is ' || v_output);
END;
/
```

OUTPUT:



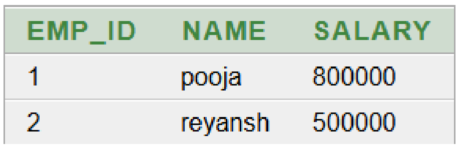
```
Factorial of 5 is 120
The factorial of 5 is 120
```

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

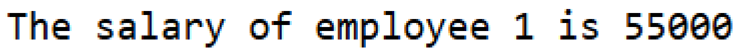
****

```
create table emppl(emp_id number , name varchar(20) ,salary number );

insert into emppl values(1,'pooja',800000);
insert into emppl values(2,'reyansh',500000);
select *from emppl;

declare
   v_emp_id number := 1;  -- change this to the employee id you want to check
   v_salary number;
begin
   -- call the procedure to retrieve the salary
   get_employee_salary(v_emp_id, v_salary);

   -- display the result
   if v_salary is not null then
      dbms_output.put_line('the salary of employee ' || v_emp_id || ' is ' || v_salary);
   else
      dbms_output.put_line('salary not found for employee ' || v_emp_id);
   end if;
end;
/
```

OUTPUT :



EMP_ID	NAME	SALARY
1	pooja	800000
2	reyansh	500000



```
The salary of employee 1 is 55000
```

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

****

```
CREATE TABLE employees20 (
   empno      NUMBER PRIMARY KEY,    -- Employee Number
(Primary Key)     emp_name   VARCHAR2(100),         --
Employee Name     hire_date  DATE,                  -- Hire Date
salary     NUMBER(8, 2),          -- Salary
   dept_id    NUMBER                 -- Department ID
);
INSERT INTO employees20 (empno, emp_name, hire_date, salary, dept_id)
VALUES (1001, 'John Doe', TO_DATE('2020-01-01', 'YYYY-MM-DD'), 50000,
10);

INSERT INTO employees20 (empno, emp_name, hire_date, salary, dept_id)
VALUES (1002, 'Jane Smith', TO_DATE('2021-03-15', 'YYYY-MM-DD'),
60000, 20);

select *from employees20;
```

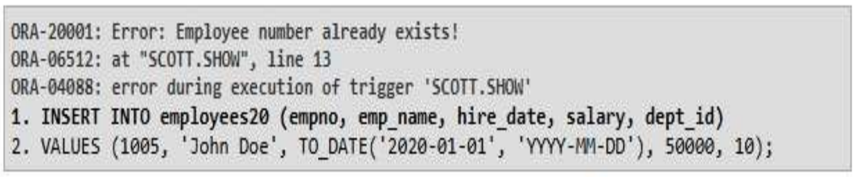
```
CREATE OR REPLACE TRIGGER show
BEFORE INSERT ON employees20
FOR EACH ROW
DECLARE

   v_count NUMBER;
BEGIN

   SELECT COUNT(*)
   INTO v_count
   FROM employees20
   WHERE empno = :NEW.empno;

   -- If empno already exists, raise an error
   IF v_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Error: Employee number
already exists!');     END IF;
END;
/
```

INSERT INTO employees20 (empno, emp_name, hire_date, salary, dept_id)
VALUES (1005, 'John Doe', TO_DATE('2020-01-01', 'YYYY-MM-DD'), 50000,
10);

INSERT INTO employees20 (empno, emp_name, hire_date, salary, dept_id)
VALUES (1001, 'John Doe', TO_DATE('2020-01-01', 'YYYY-MM-DD'), 50000,
10);

INSERT INTO employees20 (empno, emp_name, hire_date, salary, dept_id)
VALUES (1005, 'John Doe', TO_DATE('2020-01-01', 'YYYY-MM-DD'), 50000,
10);

OUTPUT



```
ORA-20001: Error: Employee number already exists!
ORA-06512: at "SCOTT.SHOW", line 13
ORA-04088: error during execution of trigger 'SCOTT.SHOW'
1. INSERT INTO employees20 (empno, emp_name, hire_date, salary, dept_id)
2. VALUES (1005, 'John Doe', TO_DATE('2020-01-01', 'YYYY-MM-DD'), 50000, 10);
```