**Q] Create two tables – EMP and DEPT**

```
CREATE DATABASE ASSIGN1;

USE ASSIGN1;


-- Create EMP table

CREATE TABLE EMP (

   EMPNO NUMERIC(4) PRIMARY KEY,

   ENAME VARCHAR(20) NOT NULL,

   JOB CHAR(10),

   MGR NUMERIC(4),

   HIREDATE DATETIME,

   SAL NUMERIC(9,2),

   COMM NUMERIC(7,2),

   DEPTNO NUMERIC(2)

);


-- Create DEPT table

CREATE TABLE DEPT (

   DEPTNO NUMERIC(2) PRIMARY KEY,

   DNAME VARCHAR(20) NOT NULL,

   LOC VARCHAR(10)

);

-- Insert records into EMP table
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES
(7369, 'SMITH', 'CLERK', 7902, '1980-12-17', 800, NULL, 20),
(7499, 'ALLEN', 'SALESMAN', 7698, '1981-02-20', 1600, 300, 30),
(7521, 'WARD', 'SALESMAN', 7698, '1981-02-22', 1250, 500, 30),
(7566, 'JONES', 'MANAGER', 7839, '1981-04-02', 2975, NULL, 20),
(7654, 'MARTIN', 'SALESMAN', 7698, '1981-09-28', 1250, 1400, 30),
(7698, 'BLAKE', 'MANAGER', 7839, '1981-05-01', 2850, NULL, 30),
(7782, 'CLARK', 'MANAGER', 7839, '1981-06-09', 2450, NULL, 10),
(7788, 'SCOTT', 'ANALYST', 7566, '1987-04-19', 3000, NULL, 20),
(7839, 'KING', 'PRESIDENT', NULL, '1981-11-17', 5000, NULL, 10),
(7844, 'TURNER', 'SALESMAN', 7698, '1981-09-08', 1500, 0, 30),
(7876, 'ADAMS', 'CLERK', 7788, '1987-05-23', 1100, NULL, 20),
(7900, 'JAMES', 'CLERK', 7698, '1981-12-03', 950, NULL, 30),
(7902, 'FORD', 'ANALYST', 7566, '1981-12-03', 3000, NULL, 20),
(7934, 'MILLER', 'CLERK', 7782, '1982-01-23', 1300, NULL, 10);
-- Insert records into DEPT table
INSERT INTO DEPT (DEPTNO, DNAME, LOC) VALUES
(10, 'ACCOUNTING', 'NEW YORK'),
(20, 'RESEARCH', 'DALLAS'),
(30, 'SALES', 'CHICAGO'),
(40, 'OPERATIONS', 'BOSTON');
```

**Perform the following queries:**

-- 1. SELECT ALL THE RECORDS FROM EMP TABLE

SELECT * FROM EMP;

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7369 | SMITH | CLERK | 7902 | 1980-12-17 00:00:00 | 800.00 | NULL | 20 |
| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00 | 1600.00 | 300.00 | 30 |
| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 00:00:00 | 1250.00 | 500.00 | 30 |
| 7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00 | 2975.00 | NULL | 20 |
| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00 | 1250.00 | 1400.00 | 30 |
| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 00:00:00 | 2850.00 | NULL | 30 |
| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 00:00:00 | 2450.00 | NULL | 10 |
| 7788 | SCOTT | ANALYST | 7566 | 1987-04-19 00:00:00 | 3000.00 | NULL | 20 |
| 7839 | KING | PRESIDENT | NULL | 1981-11-17 00:00:00 | 5000.00 | NULL | 10 |
| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 00:00:00 | 1500.00 | 0.00 | 30 |
| 7876 | ADAMS | CLERK | 7788 | 1987-05-23 00:00:00 | 1100.00 | NULL | 20 |
| 7900 | JAMES | CLERK | 7698 | 1981-12-03 00:00:00 | 950.00 | NULL | 30 |
| 7902 | FORD | ANALYST | 7566 | 1981-12-03 00:00:00 | 3000.00 | NULL | 20 |
| 7934 | MILLER | CLERK | 7782 | 1982-01-23 00:00:00 | 1300.00 | NULL | 10 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

-- 2. SELECT ALL THE RECORDS FROM DEPT TABLE

SELECT * FROM DEPT;

| DEPTNO | DNAME | LOC |
|---|---|---|
| 10 | ACCOUNTING | NEW YORK |
| 20 | RESEARCH | DALLAS |
| 30 | SALES | CHICAGO |
| 40 | OPERATIONS | BOSTON |
| NULL | NULL | NULL |

-- 3. FIND THE EMPLOYEE NAME, SALARY WHO IS WORKING IN DEPT NO 20

SELECT ENAME, SAL FROM EMP WHERE DEPTNO = 20;

| ENAME | SAL |
|---|---|
| SMITH | 800.00 |
| JONES | 2975.00 |
| SCOTT | 3000.00 |
| ADAMS | 1100.00 |
| FORD | 3000.00 |

-- 4. FIND THE NAME, JOB, SALARY OF THE EMPLOYEE WHO IS MANAGER

SELECT ENAME, JOB, SAL FROM EMP WHERE JOB = 'MANAGER';

| ENAME | JOB | SAL |
|---|---|---|
| JONES | MANAGER | 2975.00 |
| BLAKE | MANAGER | 2850.00 |
| CLARK | MANAGER | 2450.00 |

-- 5. FIND THE NAME, JOB, SALARY OF THE EMPLOYEE WHO IS NOT A MANAGER

SELECT ENAME, JOB, SAL FROM EMP WHERE JOB <> 'MANAGER';

| ENAME | JOB | SAL |
|---|---|---|
| SMITH | CLERK | 800.00 |
| ALLEN | SALESMAN | 1600.00 |
| WARD | SALESMAN | 1250.00 |
| MARTIN | SALESMAN | 1250.00 |
| SCOTT | ANALYST | 3000.00 |
| KING | PRESIDENT | 5000.00 |
| TURNER | SALESMAN | 1500.00 |
| ADAMS | CLERK | 1100.00 |
| JAMES | CLERK | 950.00 |
| FORD | ANALYST | 3000.00 |
| MILLER | CLERK | 1300.00 |

-- 6. FIND THOSE EMPLOYEES WHO WERE HIRED BETWEEN 1 MAR 1981 AND 1 JUN 1983

SELECT * FROM EMP WHERE HIREDATE BETWEEN '1981-03-01' AND '1983-06-01';

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00 | 2975.00 | NULL | 20 |
| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00 | 1250.00 | 1400.00 | 30 |
| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 00:00:00 | 2850.00 | NULL | 30 |
| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 00:00:00 | 2450.00 | NULL | 10 |
| 7839 | KING | PRESIDENT | NULL | 1981-11-17 00:00:00 | 5000.00 | NULL | 10 |
| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 00:00:00 | 1500.00 | 0.00 | 30 |
| 7900 | JAMES | CLERK | 7698 | 1981-12-03 00:00:00 | 950.00 | NULL | 30 |
| 7902 | FORD | ANALYST | 7566 | 1981-12-03 00:00:00 | 3000.00 | NULL | 20 |
| 7934 | MILLER | CLERK | 7782 | 1982-01-23 00:00:00 | 1300.00 | NULL | 10 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

-- 7. FIND EMPLOYEE NAME WHO WERE HIRED IN 1981

SELECT ENAME FROM EMP WHERE YEAR(HIREDATE) = 1981;

| ENAME |
|---|
| ALLEN |
| WARD |
| JONES |
| MARTIN |
| BLAKE |
| CLARK |
| KING |
| TURNER |
| JAMES |
| FORD |

-- 8. FIND EMPLOYEE NAME WHOSE NAME STARTS WITH ‘S’

SELECT ENAME FROM EMP WHERE ENAME LIKE 'S%';

| ENAME |
|---|
| SMITH |
| SCOTT |

-- 9. FIND EMPLOYEE NAME WHOSE NAME ENDS WITH ‘S’

SELECT ENAME FROM EMP WHERE ENAME LIKE '%S';

| ENAME |
|---|
| JONES |
| ADAMS |
| JAMES |

-- 10. FIND EMPLOYEE NAME WHO ARE WORKING IN DEPT NO 20 & 40.

SELECT ENAME FROM EMP WHERE DEPTNO IN (20, 40);

| ENAME |
|---|
| SMITH |
| JONES |
| SCOTT |
| ADAMS |
| FORD |

-- 11. FIND ENAME, JOB, AND DEPTNO WHO ARE CLERK & SALESMAN.

SELECT ENAME, JOB, DEPTNO FROM EMP WHERE JOB IN ('CLERK', 'SALESMAN');

| ENAME | JOB | DEPTNO |
|---|---|---|
| SMITH | CLERK | 20 |
| ALLEN | SALESMAN | 30 |
| WARD | SALESMAN | 30 |
| MARTIN | SALESMAN | 30 |
| TURNER | SALESMAN | 30 |
| ADAMS | CLERK | 20 |
| JAMES | CLERK | 30 |
| MILLER | CLERK | 10 |

-- 12. FIND ENAME WHO ARE MANAGER AND GETTING SALARY MORE THAN 2000

SELECT ENAME FROM EMP WHERE JOB = 'MANAGER' AND SAL > 2000;

| ENAME |
|---|
| JONES |
| BLAKE |
| CLARK |

-- 13. FIND ENAME WHO ARE WORKING IN DEPTNO 30 ORDER BY SALARY IN DESC. ORDER

SELECT ENAME FROM EMP WHERE DEPTNO = 30 ORDER BY SAL DESC;

| ENAME |
|---|
| BLAKE |
| ALLEN |
| TURNER |
| WARD |
| MARTIN |
| JAMES |

-- 14. FIND OUT THE TOTAL SALARY OF ALL THE EMPLOYEES

```
SELECT SUM(SAL) AS TOTAL_SALARY FROM EMP;
```

| TOTAL_SALARY |
|---|
| 29025.00 |

-- 15. FIND OUT AVERAGE SALARY OF ALL THE EMPLOYEES WHO ARE WORKING IN DEPTNO 30

```
SELECT AVG(SAL) AS AVG_SALARY FROM EMP WHERE DEPTNO = 30;
```

| AVG_SALARY |
|---|
| 1566.666667 |

-- 16. FIND OUT MINIMUM SALARY OF DEPT NO 20

```
SELECT MIN(SAL) AS MIN_SALARY FROM EMP WHERE DEPTNO = 20;
```

| MIN_SALARY |
|---|
| 800.00 |

-- 17. FIND OUT MAXIMUM HIREDATE

```
SELECT MAX(HIREDATE) AS MAX_HIREDATE FROM EMP;
```

| MAX_HIREDATE |
|---|
| 1987-05-23 00:00:00 |

-- 18. FIND OUT TOTAL NUMBER OF EMPLOYEES WHO ARE WORKING IN DEPT NO 10

```
SELECT COUNT(*) AS TOTAL_EMPLOYEES FROM EMP WHERE DEPTNO = 10;
```

| TOTAL_EMPLOYEES |
|---|
| 3 |

-- 19. FIND OUT DEPTNO, TOTAL SALARY OF THOSE DEPT WHERE THERE IS NO SALESMAN AND TOTAL SALARY OF DEPT IS MORE THAN 8500

```
SELECT DEPTNO, SUM(SAL) AS TOTAL_SALARY FROM EMP WHERE JOB <> 'SALESMAN' GROUP
BY DEPTNO HAVING SUM(SAL) > 8500;
```

| DEPTNO | TOTAL_SALARY |
|---|---|
| 20 | 10875.00 |
| 10 | 8750.00 |

-- 20. FIND ENAME WHO WAS HIRED FIRST

SELECT ENAME FROM EMP ORDER BY HIREDATE LIMIT 1;

| ENAME |
| --- |
| SMITH |

-- 21. FIND TOTAL SALARY FOR THOSE WHO ARE NOT MANAGER

SELECT SUM(SAL) AS TOTAL_SALARY FROM EMP WHERE JOB <> 'MANAGER';

| TOTAL_SALARY |
| --- |
| 20750.00 |

-- 22. FIND OUT JOB AND AVERAGE SALARY FOR ALL THE JOB TYPES WITH MORE THAN 2 EMPLOYEES

SELECT JOB, AVG(SAL) AS AVG_SALARY FROM EMP GROUP BY JOB HAVING COUNT(*) > 2;

| JOB | AVG_SALARY |
| --- | --- |
| CLERK | 1037.500000 |
| SALESMAN | 1400.000000 |
| MANAGER | 2758.333333 |

-- 23. FIND OUT ENAME HAVING MAXIMUM SALARY IN EACH DEPT

SELECT ENAME, DEPTNO, SAL FROM EMP WHERE (DEPTNO, SAL) IN (SELECT DEPTNO, MAX(SAL) FROM EMP GROUP BY DEPTNO);

| ENAME | DEPTNO | SAL |
| --- | --- | --- |
| BLAKE | 30 | 2850.00 |
| SCOTT | 20 | 3000.00 |
| KING | 10 | 5000.00 |
| FORD | 20 | 3000.00 |

-- 24. FIND THE SQUARE ROOT OF THE SALARY IN EMP TABLE

SELECT ENAME, SAL, SQRT(SAL) AS SQRT_SALARY FROM EMP;

| ENAME | SAL | SQRT_SALARY |
|---|---|---|
| SMITH | 800.00 | 28.284271247461902 |
| ALLEN | 1600.00 | 40 |
| WARD | 1250.00 | 35.35533905932738 |
| JONES | 2975.00 | 54.543560573178574 |
| MARTIN | 1250.00 | 35.35533905932738 |
| BLAKE | 2850.00 | 53.38539126015655 |
| CLARK | 2450.00 | 49.49747468305833 |
| SCOTT | 3000.00 | 54.772255750516614 |
| KING | 5000.00 | 70.71067811865476 |
| TURNER | 1500.00 | 38.72983346207417 |
| ADAMS | 1100.00 | 33.166247903554 |
| JAMES | 950.00 | 30.822070014844883 |
| FORD | 3000.00 | 54.772255750516614 |
| MILLER | 1300.00 | 36.05551275463989 |

-- 25. FIND AVG SALARY FOR THOSE EMPLOYEES WHOSE JOB = 'CLERK'

SELECT AVG(SAL) AS AVG_CLERK_SALARY FROM EMP WHERE JOB = 'CLERK';

| AVG_CLERK_SALARY |
|---|
| 1037.500000 |

-- 26. FIND TOTAL SALARY FOR THOSE EMPLOYEES WHO WERE HIRED IN 1981

SELECT SUM(SAL) AS TOTAL_SALARY_1981 FROM EMP WHERE YEAR(HIREDATE) = 1981;

| TOTAL_SALARY_1981 |
|---|
| 22825.00 |

-- 27. CHANGE THE JOB, DEPTNO, SALARY WHERE EMPNO = 7788

UPDATE EMP SET JOB = 'SALES', DEPTNO = 40, SAL = 2000 WHERE EMPNO = 7788;

29 18:15:30 UPDATE EMP SET JOB = 'SALES', DEPTNO = 40, SAL = 2000 WHERE EMPNO = 7788 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

-- 28. CREATE TABLE NEW USING ALL RECORDS FROM EMP

CREATE TABLE NEW AS SELECT * FROM EMP;

30 18:16:22 CREATE TABLE NEW AS SELECT * FROM EMP

-- 29. CHANGE THE JOB OF TABLE NEW TO 'SALES'

UPDATE NEW SET JOB = 'SALES';

```
31  18:16:38  UPDATE NEW SET JOB = 'SALES'
```

-- 30. SELECT ALL RECORDS FROM NEW

SELECT * FROM NEW;

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7369 | SMITH | CLERK | 7902 | 1980-12-17 00:00:00 | 800.00 | NULL | 20 |
| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00 | 1600.00 | 300.00 | 30 |
| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 00:00:00 | 1250.00 | 500.00 | 30 |
| 7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00 | 2975.00 | NULL | 20 |
| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00 | 1250.00 | 1400.00 | 30 |
| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 00:00:00 | 2850.00 | NULL | 30 |
| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 00:00:00 | 2450.00 | NULL | 10 |
| 7788 | SCOTT | SALES | 7566 | 1987-04-19 00:00:00 | 2000.00 | NULL | 40 |
| 7839 | KING | PRESIDENT | NULL | 1981-11-17 00:00:00 | 5000.00 | NULL | 10 |
| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 00:00:00 | 1500.00 | 0.00 | 30 |
| 7876 | ADAMS | CLERK | 7788 | 1987-05-23 00:00:00 | 1100.00 | NULL | 20 |
| 7900 | JAMES | CLERK | 7698 | 1981-12-03 00:00:00 | 950.00 | NULL | 30 |
| 7902 | FORD | ANALYST | 7566 | 1981-12-03 00:00:00 | 3000.00 | NULL | 20 |
| 7934 | MILLER | CLERK | 7782 | 1982-01-23 00:00:00 | 1300.00 | NULL | 10 |

-- 31. ADD A NEW COLUMN ADDRESS VARCHAR(10) TO TABLE NEW

ALTER TABLE NEW ADD COLUMN ADDRESS VARCHAR(10);

```
33  18:17:52  ALTER TABLE NEW ADD COLUMN ADDRESS VARCHAR(10)
```

-- 32. INSERT THE VALUE TO ADDRESS COLUMN IN TABLE NEW

UPDATE NEW SET ADDRESS = 'Ahmednagar';

```
34  18:19:04  UPDATE NEW SET ADDRESS = 'Ahmednagar'
```

-- 33. SELECT ALL RECORDS FROM NEW

SELECT * FROM NEW;

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO | ADDRESS |
|---|---|---|---|---|---|---|---|---|
| 7369 | SMITH | CLERK | 7902 | 1980-12-17 00:00:00 | 800.00 | NULL | 20 | NULL |
| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00 | 1600.00 | 300.00 | 30 | NULL |
| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 00:00:00 | 1250.00 | 500.00 | 30 | NULL |
| 7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00 | 2975.00 | NULL | 20 | NULL |
| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00 | 1250.00 | 1400.00 | 30 | NULL |
| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 00:00:00 | 2850.00 | NULL | 30 | NULL |
| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 00:00:00 | 2450.00 | NULL | 10 | NULL |
| 7788 | SCOTT | SALES | 7566 | 1987-04-19 00:00:00 | 2000.00 | NULL | 40 | NULL |
| 7839 | KING | PRESIDENT | NULL | 1981-11-17 00:00:00 | 5000.00 | NULL | 10 | NULL |
| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 00:00:00 | 1500.00 | 0.00 | 30 | NULL |
| 7876 | ADAMS | CLERK | 7788 | 1987-05-23 00:00:00 | 1100.00 | NULL | 20 | NULL |
| 7900 | JAMES | CLERK | 7698 | 1981-12-03 00:00:00 | 950.00 | NULL | 30 | NULL |
| 7902 | FORD | ANALYST | 7566 | 1981-12-03 00:00:00 | 3000.00 | NULL | 20 | NULL |
| 7934 | MILLER | CLERK | 7782 | 1982-01-23 00:00:00 | 1300.00 | NULL | 10 | NULL |

-- 34. UPDATE THE SIZE OF ADDRESS COLUMN FROM 10 TO 4

ALTER TABLE NEW DROP COLUMN ADDRESS;

ALTER TABLE NEW ADD COLUMN ADDRESS VARCHAR(4);

```
36  18:20:00  ALTER TABLE NEW DROP COLUMN ADDRESS
37  18:20:00  ALTER TABLE NEW ADD COLUMN ADDRESS VARCHAR(4)
```

-- 35. DELETE TABLE NEW

DROP TABLE NEW;

```
38  18:20:15  DROP TABLE NEW
```

**Q] Consider the given database schema:**

**Student (studentid , studentname,instructorid,studentcity)**

**Instructor (instructorid,Instructorname,instructor city,specialization)**

**Use all types of Joins and set operations**

**1. Add primary and foreign keys**

**2. Find the instructor of each student.**

**2. Find the student who is not having any instructor.**

**3. Find the student who is not having any instructor as well as the instructor who is not having a student.**

**4. Find the students whose instructor's specialization is computer.**

**5. Create a view containing the total number of students whose instructor belongs to "Pune".**

**Code:**

```
CREATE DATABASE ASSIGNMENT2;

USE ASSIGNMENT2;



CREATE TABLE Instructor (
   instructorid INT PRIMARY KEY,
   Instructorname VARCHAR(50),
   instructor_city VARCHAR(50),
   specialization VARCHAR(50)
);
```

```
CREATE TABLE Student (
    studentid INT PRIMARY KEY,
    studentname VARCHAR(50),
    instructorid INT,
    studentcity VARCHAR(50),
    FOREIGN KEY (instructorid) REFERENCES Instructor(instructorid)
);
```

```
INSERT INTO Instructor (instructorid, Instructorname, instructor_city, specialization)
VALUES
    (1, 'Deepak Kumar', 'Pune', 'C'),
    (2, 'Yashwant Nagarkar', 'Ayodhya', 'Math'),
    (3, 'Alakh Pandey', 'Delhi', 'Physics');
```

```
INSERT INTO Student (studentid, studentname, instructorid, studentcity)
VALUES
    (101, 'Yashwant', 1, 'Ahmednagar'),
    (102, 'Shreyash', 1, 'Beed'),
    (103, 'Chintu', 2, 'Karjat'),
    (104, 'Mantu', NULL, 'Jamkhed');
```

```
SELECT s.studentid, s.studentname, s.instructorid, i.Instructorname AS instructor_name

FROM Student s

JOIN Instructor i ON s.instructorid = i.instructorid;
```

```
SELECT *

FROM Student

WHERE instructorid IS NULL;
```

```
SELECT *

FROM Student

WHERE instructorid IS NULL;
```

```
SELECT *

FROM Instructor

WHERE instructorid NOT IN (SELECT DISTINCT instructorid FROM Student);
```

```
SELECT s.*
```

```
FROM Student s

JOIN Instructor i ON s.instructorid = i.instructorid

WHERE i.specialization = 'C';



CREATE VIEW Students_Instructors_Pune AS

SELECT COUNT(*) AS num_students

FROM Student s

JOIN Instructor i ON s.instructorid = i.instructorid

WHERE i.instructor_city = 'Pune';
```

**1.)Consider following Database Schemas Account(Acc_no, branch_name,balance) branch(branch_name,branch_city,assets) customer(cust_name,cust_street,cust_city) Depositor(cust_name,acc_no) Loan(loan_no,branch_name,amount)**

**Borrower(cust_name,loan_no) Solve following query: Create above tables with appropriate constraints like primary key, foreign key, check constrains, not null etc.**

```
mysql> create database bank;
Query OK, 1 row affected (0.01 sec)

mysql> use bank;
Database changed
mysql> CREATE TABLE branch (
    ->     branch_name VARCHAR(50) PRIMARY KEY,
    ->     branch_city VARCHAR(50),
    ->     assets DECIMAL(15, 2)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE customer (
    ->     cust_name VARCHAR(50) PRIMARY KEY,
    ->     cust_street VARCHAR(100),
    ->     cust_city VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Loan (
    ->     loan_no INT PRIMARY KEY,
    ->     branch_name VARCHAR(50),
    ->     amount DECIMAL(15, 2),
    ->     FOREIGN KEY (branch_name) REFERENCES branch(branch_name)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Borrower (
    ->     cust_name VARCHAR(50),
    ->     loan_no INT,
    ->     PRIMARY KEY (cust_name, loan_no),
```

```
    ->     FOREIGN KEY (cust_name) REFERENCES customer(cust_name),
    ->     FOREIGN KEY (loan_no) REFERENCES Loan(loan_no)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE Account (
    ->     Acc_no INT PRIMARY KEY,
    ->     branch_name VARCHAR(50),
    ->     balance DECIMAL(15, 2),
    ->     FOREIGN KEY (branch_name) REFERENCES branch(branch_name)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE Depositor (
    ->     cust_name VARCHAR(50),
    ->     acc_no INT,
    ->     PRIMARY KEY (cust_name, acc_no),
    ->     FOREIGN KEY (cust_name) REFERENCES customer(cust_name),
    ->     FOREIGN KEY (acc_no) REFERENCES Account(Acc_no)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO branch (branch_name, branch_city, assets) VALUES
    -> ('Mumbai Central', 'Mumbai', 1500000),
    -> ('Shivaji Nagar', 'Pune', 1200000),
    -> ('Delhi Main', 'Delhi', 1800000),
    -> ('Chennai South', 'Chennai', 1400000),
    -> ('Kolkata North', 'Kolkata', 1600000),
    -> ('Bangalore West', 'Bangalore', 1700000),
    -> ('Hyderabad East', 'Hyderabad', 1300000),
    -> ('Ahmedabad West', 'Ahmedabad', 1100000),
    -> ('Pune Central', 'Pune', 1900000),
```

```
    -> ('Jaipur South', 'Jaipur', 1000000);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO customer (cust_name, cust_street, cust_city) VALUES
    -> ('Rahul Sharma', 'ABC Street', 'Mumbai'),
    -> ('Priya Patel', 'XYZ Street', 'Pune'),
    -> ('Ravi Singh', '123 Street', 'Delhi'),
    -> ('Neha Gupta', '456 Street', 'Chennai'),
    -> ('Amit Kumar', '789 Street', 'Kolkata'),
    -> ('Ananya Das', 'PQR Street', 'Bangalore'),
    -> ('Akash Reddy', 'LMN Street', 'Hyderabad'),
    -> ('Divya Shah', 'UVW Street', 'Ahmedabad'),
    -> ('Sneha Jain', 'JKL Street', 'Pune'),
    -> ('Vivek Verma', 'MNO Street', 'Jaipur');
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Loan (loan_no, branch_name, amount) VALUES
    -> (201, 'Mumbai Central', 25000),
    -> (202, 'Shivaji Nagar', 35000),
    -> (203, 'Delhi Main', 40000),
    -> (204, 'Chennai South', 30000),
    -> (205, 'Kolkata North', 50000),
    -> (206, 'Bangalore West', 45000),
    -> (207, 'Hyderabad East', 28000),
    -> (208, 'Ahmedabad West', 32000),
    -> (209, 'Pune Central', 38000),
    -> (210, 'Jaipur South', 42000);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO Borrower (cust_name, loan_no) VALUES
    -> ('Rahul Sharma', 201),
    -> ('Priya Patel', 202),
    -> ('Ravi Singh', 203),
    -> ('Neha Gupta', 204),
    -> ('Amit Kumar', 205),
    -> ('Ananya Das', 206),
    -> ('Akash Reddy', 207),
    -> ('Divya Shah', 208),
    -> ('Sneha Jain', 209),
    -> ('Vivek Verma', 210);
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Account (Acc_no, branch_name, balance) VALUES
    -> (101, 'Mumbai Central', 50000),
    -> (102, 'Shivaji Nagar', 75000),
    -> (103, 'Delhi Main', 60000),
    -> (104, 'Chennai South', 45000),
    -> (105, 'Kolkata North', 90000),
    -> (106, 'Bangalore West', 80000),
    -> (107, 'Hyderabad East', 70000),
    -> (108, 'Ahmedabad West', 55000),
    -> (109, 'Pune Central', 85000),
    -> (110, 'Jaipur South', 62000);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Depositor (cust_name, acc_no) VALUES
    -> ('Rahul Sharma', 101),
    -> ('Priya Patel', 102),
```

```
    -> ('Ravi Singh', 203),
    -> ('Neha Gupta', 204),
    -> ('Amit Kumar', 205),
    -> ('Ananya Das', 206),
    -> ('Akash Reddy', 207),
    -> ('Divya Shah', 208),
    -> ('Sneha Jain', 209),
    -> ('Vivek Verma', 210);
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Account (Acc_no, branch_name, balance) VALUES
    -> (101, 'Mumbai Central', 50000),
    -> (102, 'Shivaji Nagar', 75000),
    -> (103, 'Delhi Main', 60000),
    -> (104, 'Chennai South', 45000),
    -> (105, 'Kolkata North', 90000),
    -> (106, 'Bangalore West', 80000),
    -> (107, 'Hyderabad East', 70000),
    -> (108, 'Ahmedabad West', 55000),
    -> (109, 'Pune Central', 85000),
    -> (110, 'Jaipur South', 62000);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Depositor (cust_name, acc_no) VALUES
    -> ('Rahul Sharma', 101),
    -> ('Priya Patel', 102),
    -> ('Ravi Singh', 103),
    -> ('Neha Gupta', 104),
    -> ('Amit Kumar', 105),
    -> ('Ananya Das', 106),
    -> ('Akash Reddy', 107),
    -> ('Divya Shah', 108),
    -> ('Sneha Jain', 109),
    -> ('Vivek Verma', 110);
```

## *QUERIES:*

**1)Find the names of all branches in loan relation.**

```
mysql> SELECT DISTINCT branch_name FROM Loan;
+-----------------+
| branch_name     |
+-----------------+
| Ahmedabad West  |
| Bangalore West  |
| Chennai South   |
| Delhi Main      |
| Hyderabad East  |
| Jaipur South    |
| Kolkata North   |
| Mumbai Central  |
| Pune Central    |
| Shivaji Nagar   |
+-----------------+
10 rows in set (0.00 sec)
```

2.)Find all loan numbers for loans made at Shivaji nagar Branch with loan amount > 12000.

```
mysql> SELECT loan_no
    -> FROM Loan
    -> WHERE branch_name = 'Shivaji nagar' AND amount > 12000;
+---------+
| loan_no |
+---------+
|     202 |
+---------+
1 row in set (0.00 sec)
```

**3.)Find all customers who have a loan from bank. Find their names,loan_no and loan amount.**

```
mysql> SELECT c.cust_name, l.loan_no, l.amount
    -> FROM customer c
    -> JOIN Borrower b ON c.cust_name = b.cust_name
    -> JOIN Loan l ON b.loan_no = l.loan_no;
+-------------+---------+----------+
| cust_name   | loan_no | amount   |
+-------------+---------+----------+
| Akash Reddy |     207 | 28000.00 |
| Amit Kumar  |     205 | 50000.00 |
| Ananya Das  |     206 | 45000.00 |
| Divya Shah  |     208 | 32000.00 |
| Neha Gupta  |     204 | 30000.00 |
| Priya Patel |     202 | 35000.00 |
| Rahul Sharma|     201 | 25000.00 |
| Ravi Singh  |     203 | 40000.00 |
| Sneha Jain  |     209 | 38000.00 |
| Vivek Verma |     210 | 42000.00 |
+-------------+---------+----------+
10 rows in set (0.00 sec)
```

**4.)List all customers in alphabetical order who have loan from Shivaji nagar branch.**

```
mysql> SELECT c.cust_name
    -> FROM customer c
    -> JOIN Borrower b ON c.cust_name = b.cust_name
    -> JOIN Loan l ON b.loan_no = l.loan_no
    -> WHERE l.branch_name = 'Shivaji nagar'
    -> ORDER BY c.cust_name;
+-------------+
| cust_name   |
+-------------+
| Priya Patel |
+-------------+
1 row in set (0.00 sec)
```

**5.)Find all customers who have an account or loan or both at bank.**

```
mysql> SELECT DISTINCT cust_name FROM (
    ->     SELECT cust_name FROM Depositor
    ->     UNION
    ->     SELECT cust_name FROM Borrower
    -> ) AS customers;
+--------------+
| cust_name    |
+--------------+
| Rahul Sharma |
| Priya Patel  |
| Ravi Singh   |
| Neha Gupta   |
| Amit Kumar   |
| Ananya Das   |
| Akash Reddy  |
| Divya Shah   |
| Sneha Jain   |
| Vivek Verma  |
+--------------+
10 rows in set (0.00 sec)
```

6.)Find all customers who have both account and loan at bank.

```
mysql> SELECT cust_name
    -> FROM Depositor
    -> WHERE cust_name IN (
    ->     SELECT cust_name
    ->     FROM Borrower
    -> )
    -> GROUP BY cust_name;
+--------------+
| cust_name    |
+--------------+
| Akash Reddy  |
| Amit Kumar   |
| Ananya Das   |
| Divya Shah   |
| Neha Gupta   |
| Priya Patel  |
| Rahul Sharma |
| Ravi Singh   |
| Sneha Jain   |
| Vivek Verma  |
+--------------+
10 rows in set (0.00 sec)
```

**7.)Find all customer who have account but no loan at the bank.**

```
mysql> SELECT cust_name
    -> FROM Depositor
    -> WHERE cust_name NOT IN (
    ->     SELECT cust_name FROM Borrower
    -> );
Empty set (0.01 sec)
```

**8.)Find average account balance at Shivaji nagar branch.**

```
mysql> SELECT AVG(balance)
    -> FROM Account
    -> WHERE branch_name = 'Shivaji nagar';
+--------------+
| AVG(balance) |
+--------------+
| 75000.000000 |
+--------------+
1 row in set (0.00 sec)
```

9.)Find the average account balance at each branch

```
mysql> SELECT branch_name, AVG(balance)
    -> FROM Account
    -> GROUP BY branch_name;
+----------------+--------------+
| branch_name    | AVG(balance) |
+----------------+--------------+
| Ahmedabad West | 55000.000000 |
| Bangalore West | 80000.000000 |
| Chennai South  | 45000.000000 |
| Delhi Main     | 60000.000000 |
| Hyderabad East | 70000.000000 |
| Jaipur South   | 62000.000000 |
| Kolkata North  | 90000.000000 |
| Mumbai Central | 50000.000000 |
| Pune Central   | 85000.000000 |
| Shivaji Nagar  | 75000.000000 |
+----------------+--------------+
10 rows in set (0.00 sec)
```

10.)Find no. of depositors at each branch.

```
mysql> SELECT a.branch_name, COUNT(*) AS num_depositors
    -> FROM Depositor d
    -> JOIN Account a ON d.acc_no = a.Acc_no
    -> GROUP BY a.branch_name;
+----------------+----------------+
| branch_name    | num_depositors |
+----------------+----------------+
| Mumbai Central |              1 |
| Shivaji Nagar  |              1 |
| Delhi Main     |              1 |
| Chennai South  |              1 |
| Kolkata North  |              1 |
| Bangalore West |              1 |
| Hyderabad East |              1 |
| Ahmedabad West |              1 |
| Pune Central   |              1 |
| Jaipur South   |              1 |
+----------------+----------------+
10 rows in set (0.01 sec)
```

11.)Find the branches where average account balance > 12000.

```
mysql> SELECT branch_name
    -> FROM Account
    -> GROUP BY branch_name
    -> HAVING AVG(balance) > 12000;
+----------------+
| branch_name    |
+----------------+
| Ahmedabad West |
| Bangalore West |
| Chennai South  |
| Delhi Main     |
| Hyderabad East |
| Jaipur South   |
| Kolkata North  |
| Mumbai Central |
| Pune Central   |
| Shivaji Nagar  |
+----------------+
10 rows in set (0.01 sec)
```

12.)Find number of tuples in customer relation.

```
mysql> SELECT COUNT(*) FROM customer;
+----------+
| COUNT(*) |
+----------+
|       10 |
+----------+
1 row in set (0.01 sec)
```

13.)Calculate total loan amount given by bank.

```
mysql> SELECT SUM(amount) FROM Loan;
+-------------+
| SUM(amount) |
+-------------+
|   365000.00 |
+-------------+
1 row in set (0.00 sec)
```

14.)Delete all loans with loan amount between 1300 and 1500.

```
mysql> DELETE FROM Loan WHERE amount BETWEEN 1300 AND 1500;
Query OK, 0 rows affected (0.01 sec)
```

15.)Delete all tuples at every branch located in Sharanpur road.

```
mysql> DELETE FROM branch WHERE branch_city = 'Sharanpur road';
Query OK, 0 rows affected (0.00 sec)
```

2.) **Consider the given relational table: employee(empno , empname, designation, city, salary, zipcode, county) 1. Creates a sequence used to generate employee numbers for the empno column of the emp table. 2.Create an Index on county. 3. Find the zipcode whose county = 071 and check whether the query uses the Index and write your observation. 4. Create a view for employees having salary < 50000 and stays in 'Mumbai'.**

```
mysql> CREATE TABLE employee (
    ->     empno INT AUTO_INCREMENT PRIMARY KEY,
    ->     empname VARCHAR(50),
    ->     designation VARCHAR(50),
    ->     city VARCHAR(50),
    ->     salary DECIMAL(10, 2),
    ->     zipcode VARCHAR(10),
    ->     county VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> INSERT INTO employee (empname, designation, city, salary, zipcode, county) VALUES
    ->     ('Ramesh Kumar', 'Manager', 'Mumbai', 60000.00, '400001', 'Mumbai'),
    ->     ('Sita Sharma', 'Developer', 'Delhi', 45000.00, '110001', 'Delhi'),
    ->     ('Amit Singh', 'HR Manager', 'Kolkata', 55000.00, '700001', 'Kolkata'),
    ->     ('Priya Patel', 'Sales Executive', 'Chennai', 48000.00, '600001', 'Chennai'),
    ->     ('Rajesh Gupta', 'Team Lead', 'Mumbai', 70000.00, '400001', 'Mumbai'),
    ->     ('Anita Das', 'Software Engineer', 'Delhi', 42000.00, '110001', 'Delhi'),
    ->     ('Vikram Mishra', 'Consultant', 'Kolkata', 58000.00, '700001', 'Kolkata'),
    ->     ('Geeta Reddy', 'Administrator', 'Chennai', 49000.00, '600001', 'Chennai'),
    ->     ('Kiran Verma', 'Analyst', 'Mumbai', 62000.00, '400001', 'Mumbai'),
    ->     ('Aruna Choudhury', 'Tester', 'Delhi', 43000.00, '110001', 'Delhi');
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

1.Creates a sequence used to generate employee numbers for the empno column of the emp table.

```
mysql>
mysql> SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|               11 |
+------------------+
1 row in set (0.00 sec)
```

2.Create an Index on county.

```
mysql> CREATE INDEX idx_county ON employee(county);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

3.Find the zipcode whose county = 071 and check whether the query uses the Index and write your observation.

```
mysql> EXPLAIN SELECT zipcode FROM employee WHERE county = '071';
+----+-------------+----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
| id | select_type | table    | partitions | type | possible_keys | key        | key_len | ref   | rows | filtered | Extra |
+----+-------------+----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | employee | NULL       | ref  | idx_county    | idx_county | 203     | const |    1 |   100.00 | NULL  |
+----+-------------+----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

4.Create a view for employees having salary < 50000 and stays in ‘Mumbai’.

```
mysql> CREATE VIEW low_salary_mumbai_employees AS
    -> SELECT * FROM employee WHERE salary < 50000 AND city = 'Mumbai';
Query OK, 0 rows affected (0.01 sec)
```

**3.)Consider the given database schema: Student (studentid , studentname,instructorid,studentcity) Instructor(instructorid,Instructorname,instructorcity,specialization) Use all types of Joins**

```
mysql> CREATE TABLE Student (
    ->     studentid INT PRIMARY KEY,
    ->     studentname VARCHAR(50),
    ->     instructorid INT,
    ->     studentcity VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Instructor (
    ->     instructorid INT PRIMARY KEY,
    ->     Instructorname VARCHAR(50),
    ->     instructorcity VARCHAR(50),
    ->     specialization VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> INSERT INTO Student (studentid, studentname, instructorid, studentcity) VALUES
    ->     (1, 'Jiya Dave', 101, 'Mumbai'),
    ->     (2, 'Priya Das', 102, 'Kolkata'),
    ->      (3, 'Praveen Kolhi', 103, 'Pune');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Instructor (instructorid, Instructorname, instructorcity, specialization) VALUES
    ->     (101, 'Reema Ray', 'Mumbai', 'Mathematics'),
    ->     (102, 'Anuj Sharma', 'Kolkata', 'Computer Science'),
    ->     (103, 'Rakesh Singh', 'Pune', 'Physics');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

1.Find the instructor of each student.

```
mysql> SELECT s.studentname, i.Instructorname
    -> FROM Student s
    -> INNER JOIN Instructor i ON s.instructorid = i.instructorid;
+---------------+----------------+
| studentname   | Instructorname |
+---------------+----------------+
| Jiya Dave     | Reema Ray      |
| Priya Das     | Anuj Sharma    |
| Praveen Kolhi | Rakesh Singh   |
+---------------+----------------+
3 rows in set (0.00 sec)
```

2.Find the student who is not having any instructor.

```
mysql> SELECT s.studentname
    -> FROM Student s
    -> LEFT JOIN Instructor i ON s.instructorid = i.instructorid
    -> WHERE i.instructorid IS NULL;
Empty set (0.00 sec)
```

3.Find the student who is not having any instructor as well as instructor who is not having student.

```
mysql> SELECT COALESCE(s.studentname, 'No student') AS StudentName, COALESCE(i.Instructorname, 'No instructor') AS InstructorName
    -> FROM Student s
    -> LEFT JOIN Instructor i ON s.instructorid = i.instructorid
    -> UNION
    -> SELECT COALESCE(s.studentname, 'No student') AS StudentName, COALESCE(i.Instructorname, 'No instructor') AS InstructorName
    -> FROM Instructor i
    -> LEFT JOIN Student s ON s.instructorid = i.instructorid
    -> WHERE s.instructorid IS NULL;
+---------------+----------------+
| StudentName   | InstructorName |
+---------------+----------------+
| Jiya Dave     | Reema Ray      |
| Priya Das     | Anuj Sharma    |
| Praveen Kolhi | Rakesh Singh   |
+---------------+----------------+
3 rows in set (0.00 sec)
```

4.Find the students whose instructor’s specialization is computer.

```
mysql> SELECT s.studentname
    -> FROM Student s
    -> INNER JOIN Instructor i ON s.instructorid = i.instructorid
    -> WHERE i.specialization = 'Computer Science';
+-------------+
| studentname |
+-------------+
| Priya Das   |
+-------------+
1 row in set (0.00 sec)
```

**5.Create a view containing total number of students whose instructor belongs to "Pune".**

```
mysql> CREATE VIEW StudentsWithPuneInstructors AS
    -> SELECT COUNT(*) AS TotalStudentsWithPuneInstructors
    -> FROM Student s
    -> INNER JOIN Instructor i ON s.instructorid = i.instructorid
    -> WHERE i.instructorcity = 'Pune';
Query OK, 0 rows affected (0.01 sec)

mysql> Select * from StudentsWithPuneInstructors;
+----------------------------------+
| TotalStudentsWithPuneInstructors |
+----------------------------------+
|                                1 |
+----------------------------------+
1 row in set (0.00 sec)
```

**4.)Create a database with following schemas Borrower(Rollin, Name, DateofIssue, NameofBook, Status) & Fine(Roll_no,Date,Amt)**

```
mysql> use bakul;
Database changed
mysql> CREATE TABLE Borrower (
    ->     Rollin INT,
    ->     Name VARCHAR(255),
    ->     DateofIssue DATE,
    ->     NameofBook VARCHAR(255),
    ->     Status CHAR(1),
    ->     PRIMARY KEY (Rollin)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Fine (
    ->     Roll_no INT,
    ->     Date DATE,
    ->     Amt DECIMAL(10, 2),
    ->     FOREIGN KEY (Roll_no) REFERENCES Borrower(Rollin)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> INSERT INTO Borrower (Rollin, Name, DateofIssue, NameofBook, Status) VALUES
    ->     (1, 'John Doe', '2024-04-15', 'The Great Gatsby', 'I'),
    ->     (2, 'Jane Smith', '2024-04-20', 'To Kill a Mockingbird', 'I'),
    ->     (3, 'Alice Johnson', '2024-04-10', 'Pride and Prejudice', 'R'),
    ->     (4, 'Bob Brown', '2024-04-25', '1984', 'I'),
    ->     (5, 'Charlie Wilson', '2024-04-18', 'The Catcher in the Rye', 'R');
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Fine (Roll_no, Date, Amt) VALUES
    ->     (1, '2024-05-01', 25.00),
    ->     (2, '2024-05-05', 30.00),
    ->     (4, '2024-05-10', 50.00);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

**1.Write a PL/SQL block to accept input for Borrower table.**

**2.Write a PL/SQL block using control structures to calculate fine by using the following rules:**

**a.check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day**

**b.If no. of days>30, per day fine will be Rs 50 per day**

**c.for days less than 30, Rs. 5 per day. After submitting the book, status will change from I to R. If condition of fine is true, then details will be stored into fine table.**

```
sql> DELIMITER //
sql>
sql> CREATE PROCEDURE InsertBorrower(
  ->     IN v_rollin INT,
  ->     IN v_name VARCHAR(100),
  ->     IN v_date_of_issue DATE,
  ->     IN v_name_of_book VARCHAR(100),
  ->     IN v_status CHAR(1)
  -> )
  -> BEGIN
  ->     INSERT INTO Borrower (Rollin, Name, DateofIssue, NameofBook, Status)
  ->     VALUES (v_rollin, v_name, v_date_of_issue, v_name_of_book, v_status);
  ->     SELECT 'Data inserted successfully into Borrower table.' AS Message;
  -> END //
ery OK, 0 rows affected (0.01 sec)
```

```
mysql>
mysql> DELIMITER ;
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE CalculateFine(
    ->     IN v_roll_no INT
    -> )
    -> BEGIN
    ->     DECLARE v_days_late INT;
    ->     DECLARE v_amt DECIMAL(10, 2);
    ->
    ->     SELECT FLOOR(DATEDIFF(CURDATE(), DateofIssue)) INTO v_days_late FROM Borrower WHERE Rollin = v_roll_no;
    ->
    ->     IF v_days_late BETWEEN 15 AND 30 THEN
    ->         SET v_amt = 5 * v_days_late;
    ->     ELSEIF v_days_late > 30 THEN
    ->         SET v_amt = 50 * v_days_late;
    ->     ELSE
    ->         SET v_amt = 0;
    ->     END IF;
    ->
    ->     IF v_amt > 0 THEN
    ->         INSERT INTO Fine (Roll_no, Date, Amt) VALUES (v_roll_no, CURDATE(), v_amt);
    ->         SELECT CONCAT('Fine calculated and inserted into Fine table: Rs ', v_amt) AS Message;
    ->     ELSE
    ->         SELECT 'No fine applicable.' AS Message;
    ->     END IF;
    -> END //
Query OK, 0 rows affected (0.01 sec)
```

```
mysql>
mysql> DELIMITER ;
mysql> CALL InsertBorrower(123, 'John Doe', '2024-04-25', 'The Book', 'I');
+-------------------------------------------------+
| Message                                         |
+-------------------------------------------------+
| Data inserted successfully into Borrower table. |
+-------------------------------------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL CalculateFine(123);
+----------------------+
| Message              |
+----------------------+
| No fine applicable.  |
+----------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

**5.)Create two tables O_Roll(Rollno,Name,DOB,Phone,address)**

**N_Roll(Rollno,Name,DOB,Phone,address) Write a PL/SQL block using various types of cursor(implicit,Explicit,For, Parameterized) to merge records from O_Roll table with that of N_Roll in such a way duplicate records are to be eliminated.**

```
mysql> use bakul;
Database changed
mysql> desc o_roll;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| Rollno  | int          | NO   | PRI | NULL    |       |
| Name    | varchar(255) | YES  |     | NULL    |       |
| DOB     | date         | YES  |     | NULL    |       |
| Phone   | varchar(20)  | YES  |     | NULL    |       |
| Address | varchar(255) | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

```
mysql> desc n_roll;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| Rollno  | int          | NO   | PRI | NULL    |       |
| Name    | varchar(255) | YES  |     | NULL    |       |
| DOB     | date         | YES  |     | NULL    |       |
| Phone   | varchar(20)  | YES  |     | NULL    |       |
| Address | varchar(255) | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> select * from o_roll;
+--------+---------------+------------+------------+--------------+
| Rollno | Name          | DOB        | Phone      | Address      |
+--------+---------------+------------+------------+--------------+
|      1 | John Doe      | 2000-01-15 | 1234567890 | 123 Main St  |
|      2 | Jane Smith    | 1999-05-20 | 9876543210 | 456 Elm St   |
|      3 | Alice Johnson | 2001-09-10 | 4567890123 | 789 Oak St   |
|      4 | Bob Brown     | 2002-03-25 | 7890123456 | 101 Pine St  |
|      5 | Charlie Wilson| 2000-07-18 | 2345678901 | 202 Maple St |
+--------+---------------+------------+------------+--------------+
5 rows in set (0.00 sec)
```

```
mysql> select * from n_roll;
+--------+-------------+------------+------------+--------------+
| Rollno | Name        | DOB        | Phone      | Address      |
+--------+-------------+------------+------------+--------------+
|      6 | David Lee   | 1998-11-30 | 3456789012 | 303 Cedar St |
|      7 | Emily Davis | 2003-02-05 | 5678901234 | 404 Birch St |
+--------+-------------+------------+------------+--------------+
2 rows in set (0.00 sec)
```

```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE MergeRecords()
    -> BEGIN
    ->     -- Declare variables
    ->     DECLARE v_o_roll_rollno INT;
    ->     DECLARE v_o_roll_name VARCHAR(100);
    ->     DECLARE v_o_roll_dob DATE;
    ->     DECLARE v_o_roll_phone VARCHAR(20);
    ->     DECLARE v_o_roll_address VARCHAR(255);
    ->
    ->     -- Declare cursors
    ->     DECLARE cur_implicit CURSOR FOR SELECT * FROM O_Roll;
    ->     DECLARE cur_explicit CURSOR FOR SELECT * FROM N_Roll;
    ->     DECLARE cur_for CURSOR FOR SELECT * FROM O_Roll;
    ->     DECLARE cur_parameterized CURSOR FOR SELECT * FROM N_Roll WHERE Rollno = p_rollno;
    ->
    ->     -- Implicit Cursor
    ->     OPEN cur_implicit;
    ->     loop_implicit: LOOP
    ->         FETCH cur_implicit INTO v_o_roll_rollno, v_o_roll_name, v_o_roll_dob, v_o_roll_phone, v_o_roll_address;
    ->         IF v_o_roll_rollno IS NULL THEN
    ->             LEAVE loop_implicit;
    ->         END IF;
    ->         -- Check if the record already exists in N_Roll
    ->         SELECT Rollno INTO v_o_roll_rollno FROM N_Roll WHERE Rollno = v_o_roll_rollno;
    ->         IF v_o_roll_rollno IS NULL THEN
    ->             -- Insert the record into N_Roll
    ->             INSERT INTO N_Roll VALUES (v_o_roll_rollno, v_o_roll_name, v_o_roll_dob, v_o_roll_phone, v_o_roll_address);
    ->         END IF;
    ->     END LOOP loop_implicit;
```

```
    ->     CLOSE cur_implicit;
    ->
    ->     -- Explicit Cursor
    ->     OPEN cur_explicit;
    ->     loop_explicit: LOOP
    ->         FETCH cur_explicit INTO v_o_roll_rollno, v_o_roll_name, v_o_roll_dob, v_o_roll_phone, v_o_roll_address;
    ->         IF v_o_roll_rollno IS NULL THEN
    ->             LEAVE loop_explicit;
    ->         END IF;
    ->         -- Check if the record already exists in O_Roll
    ->         SELECT Rollno INTO v_o_roll_rollno FROM O_Roll WHERE Rollno = v_o_roll_rollno;
    ->         IF v_o_roll_rollno IS NULL THEN
    ->             -- Insert the record into O_Roll
    ->             INSERT INTO O_Roll VALUES (v_o_roll_rollno, v_o_roll_name, v_o_roll_dob, v_o_roll_phone, v_o_roll_address);
    ->         END IF;
    ->     END LOOP loop_explicit;
    ->     CLOSE cur_explicit;
    ->
    ->     -- For Cursor
    ->     OPEN cur_for;
    ->     loop_for: LOOP
    ->         FETCH cur_for INTO v_o_roll_rollno, v_o_roll_name, v_o_roll_dob, v_o_roll_phone, v_o_roll_address;
    ->         IF v_o_roll_rollno IS NULL THEN
    ->             LEAVE loop_for;
    ->         END IF;
    ->         -- Check if the record already exists in N_Roll
    ->         SELECT Rollno INTO v_o_roll_rollno FROM N_Roll WHERE Rollno = v_o_roll_rollno;
    ->         IF v_o_roll_rollno IS NULL THEN
    ->             -- Insert the record into N_Roll
    ->             INSERT INTO N_Roll VALUES (v_o_roll_rollno, v_o_roll_name, v_o_roll_dob, v_o_roll_phone, v_o_roll_address);
    ->         END IF;
    ->     END LOOP loop_for;
```

**6)Create a Library database with the schema Books(AccNo,Title,Author,Publisher,Count).**

**a.Create a table Library_Audit with same fiels as of Books.**

```
    ->     END LOOP loop_for;
    ->     CLOSE cur_for;
    ->
    ->     -- Parameterized Cursor
    ->     OPEN cur_parameterized;
    ->     loop_parameterized: LOOP
    ->         FETCH cur_parameterized INTO v_o_roll_rollno, v_o_roll_name, v_o_roll_dob, v_o_roll_phone, v_o_roll_address;
    ->         IF v_o_roll_rollno IS NULL THEN
    ->             LEAVE loop_parameterized;
    ->         END IF;
    ->         -- Check if the record already exists in O_Roll
    ->         SELECT Rollno INTO v_o_roll_rollno FROM O_Roll WHERE Rollno = v_o_roll_rollno;
    ->         IF v_o_roll_rollno IS NULL THEN
    ->             -- Insert the record into O_Roll
    ->             INSERT INTO O_Roll VALUES (v_o_roll_rollno, v_o_roll_name, v_o_roll_dob, v_o_roll_phone, v_o_roll_address);
    ->         END IF;
    ->     END LOOP loop_parameterized;
    ->     CLOSE cur_parameterized;
    ->
    -> END//
Query OK, 0 rows affected (0.01 sec)
```

**b.Create a before trigger to insert records into Librry_Audit table if there is deletion in Books table.**

**c.Create a after trigger to insert records into Librry_Audit table if there is updation in Books table.**

```
mysql> CREATE TABLE Books (
    ->     AccNo INT PRIMARY KEY,
    ->     Title VARCHAR(255),
    ->     Author VARCHAR(255),
    ->     Publisher VARCHAR(255),
    ->     Count INT
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Library_Audit (
    ->     AccNo INT,
    ->     Title VARCHAR(255),
    ->     Author VARCHAR(255),
    ->     Publisher VARCHAR(255),
    ->     Count INT
    -> );
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> INSERT INTO Books (AccNo, Title, Author, Publisher, Count)
    -> VALUES
    ->     (1, 'To Kill a Mockingbird', 'Harper Lee', 'Harper Perennial Modern Classics', 5),
    ->     (2, '1984', 'George Orwell', 'Signet Classic', 8),
    ->     (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Scribner', 6),
    ->     (4, 'Pride and Prejudice', 'Jane Austen', 'Penguin Classics', 7),
    ->     (5, 'The Catcher in the Rye', 'J.D. Salinger', 'Back Bay Books', 4);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> DELIMITER //
mysql>
mysql> CREATE TRIGGER Before_Delete_Books
    -> BEFORE DELETE ON Books
    -> FOR EACH ROW
    -> BEGIN
    ->     INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count)
    ->     VALUES (OLD.AccNo, OLD.Title, OLD.Author, OLD.Publisher, OLD.Count);
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> -- Step 4: Create an after trigger to insert records into Library_Audit table on updation
mysql> DELIMITER //
mysql>
mysql> CREATE TRIGGER After_Update_Books
    -> AFTER UPDATE ON Books
    -> FOR EACH ROW
    -> BEGIN
    ->     INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count)
    ->     VALUES (NEW.AccNo, NEW.Title, NEW.Author, NEW.Publisher, NEW.Count);
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)
```

**7) Create a procedure called USER_QUERY_EMP that accepts three parameters. Parameter p_myeno is of IN parameter mode which provides the empno value. The other two parameters p_myjob and p_mysal are of OUT mode. The procedure retrieves the salary and job of an employee with the provided employee number and assigns those to the two OUT parameters respectively. The procedure should handle the error if the empno does not exist in the EMP table by displaying an appropriate message. Use bind variables for the two OUT Parameters. Compile the code, invoke the procedure, and display the salary and job title for employee number 7839. Do the same for employee number 7123.**

```
mysql> use bakul;
Database changed
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE USER_QUERY_EMP(
    ->     IN p_myeno INT,
    ->     OUT p_myjob VARCHAR(50),
    ->     OUT p_mysal DECIMAL(10,2)
    -> )
    -> BEGIN
    ->     -- Declare variable to store count of matching records
    ->     DECLARE v_count INT;
    ->
    ->     -- Check if employee number exists in the EMP table
    ->     SELECT COUNT(*)
    ->     INTO v_count
    ->     FROM emp
    ->     WHERE empno = p_myeno;
    ->
    ->     -- If employee number does not exist, set OUT parameters to NULL and display message
    ->     IF v_count = 0 THEN
    ->         SET p_myjob = NULL;
    ->         SET p_mysal = NULL;
    ->         SELECT 'Employee not found' AS message;
    ->     ELSE
    ->         -- Retrieve job and salary based on the provided empno
    ->         SELECT job, sal INTO p_myjob, p_mysal
    ->         FROM emp
    ->         WHERE empno = p_myeno;
    ->     END IF;
    ->
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL USER_QUERY_EMP(7839, @job_7839, @sal_7839);
ERROR 1146 (42S02): Table 'bakul.emp' doesn't exist
mysql> SELECT @job_7839 AS job_7839, @sal_7839 AS sal_7839;
+--------------------+--------------------+
| job_7839           | sal_7839           |
+--------------------+--------------------+
| NULL               | NULL               |
+--------------------+--------------------+
1 row in set (0.00 sec)

mysql>
mysql> CALL USER_QUERY_EMP(7123, @job_7123, @sal_7123);
ERROR 1146 (42S02): Table 'bakul.emp' doesn't exist
mysql> SELECT @job_7123 AS job_7123, @sal_7123 AS sal_7123;
+--------------------+--------------------+
| job_7123           | sal_7123           |
+--------------------+--------------------+
| NULL               | NULL               |
+--------------------+--------------------+
1 row in set (0.00 sec)
```

**8)Create a function named USER_ANNUAL_COMP that has three parameters p_eno, p_sal and p_comm for passing on the values of an employee number, the current salary and commission of the employee respectively. The function calculates and returns the annual compensation of the employee by using the following formula. annual_compensation = (p_sal+p_comm)*12 If the salary or commission value is NULL then zero should be substituted for it. Give a call to USER_ANNUAL_COMP from a SELECT statement, against the EMP table**

```
mysql> use bakul;
Database changed
mysql> DELIMITER //
mysql>
mysql> CREATE FUNCTION USER_ANNUAL_COMP(
    ->     p_eno INT,
    ->     p_sal DECIMAL(10,2),
    ->     p_comm DECIMAL(10,2)
    -> )
    -> RETURNS DECIMAL(10,2)
    -> DETERMINISTIC
    -> BEGIN
    ->     DECLARE annual_comp DECIMAL(10,2);
    ->
    ->     -- If salary or commission is NULL, substitute with zero
    ->     IF p_sal IS NULL THEN
    ->         SET p_sal = 0;
    ->     END IF;
    ->
    ->     IF p_comm IS NULL THEN
    ->         SET p_comm = 0;
    ->     END IF;
    ->
    ->     -- Calculate annual compensation
    ->     SET annual_comp = (p_sal + p_comm) * 12;
    ->
    ->     RETURN annual_comp;
    -> END //
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> use bakul;
Database changed
mysql> CREATE TABLE emp (
    ->     empno INT PRIMARY KEY,
    ->     ename VARCHAR(50),
    ->     sal DECIMAL(10,2),
    ->     comm DECIMAL(10,2)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> USE bakul;
Database changed
mysql>
mysql> INSERT INTO emp (empno, ename, sal, comm) VALUES
    -> (7839, 'KING', 5000.00, 1000.00),
    -> (7566, 'JONES', 3000.00, 500.00),
    -> (7698, 'BLAKE', 2850.00, 0),
    -> (7782, 'CLARK', 2450.00, 0),
    -> (7788, 'SCOTT', 3000.00, 0),
    -> (7902, 'FORD', 3000.00, 0),
    -> (7844, 'TURNER', 1500.00, 0),
    -> (7900, 'JAMES', 950.00, 0),
    -> (7654, 'MARTIN', 1250.00, 1400.00),
    -> (7499, 'ALLEN', 1600.00, 300.00),
    -> (7521, 'WARD', 1250.00, 500.00),
    -> (7934, 'MILLER', 1300.00, 0);
Query OK, 12 rows affected (0.01 sec)
Records: 12  Duplicates: 0  Warnings: 0
```

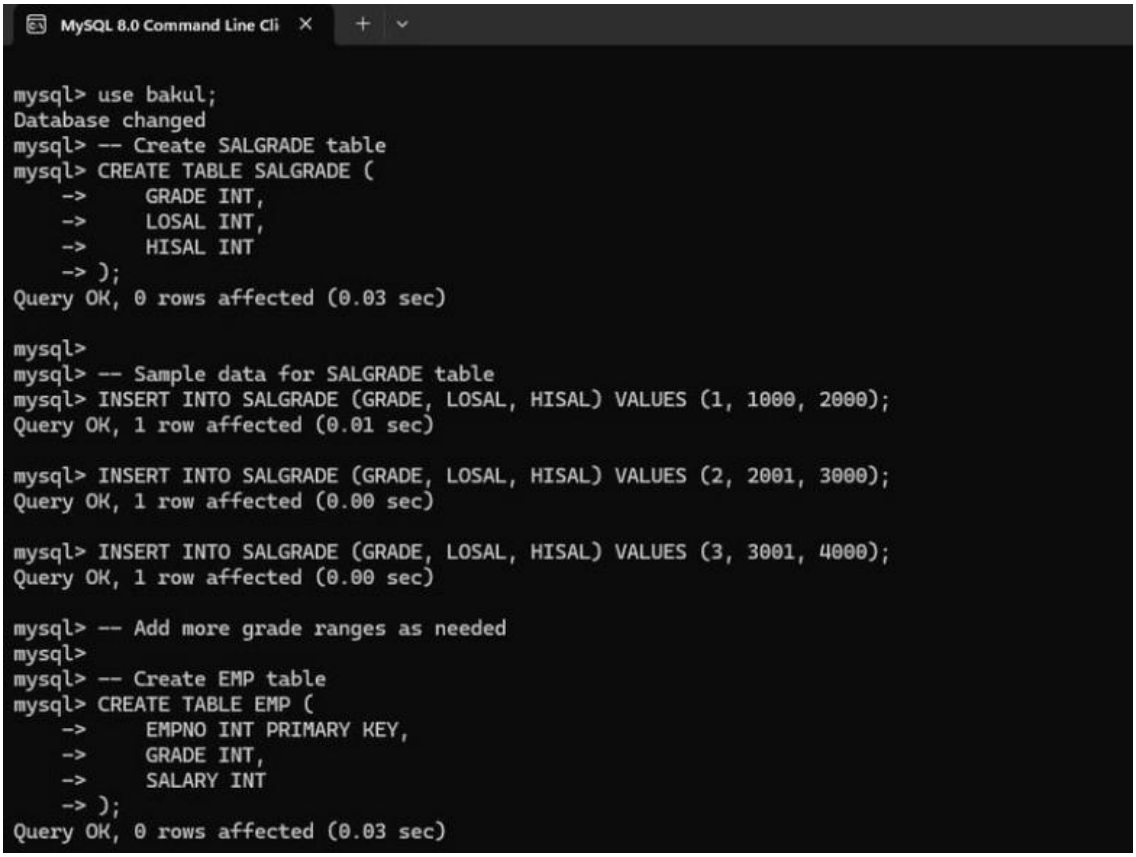
```
mysql> DELIMITER //
mysql>
mysql> CREATE FUNCTION USER_ANNUAL_COMP(
    ->     p_eno INT,
    ->     p_sal DECIMAL(10,2),
    ->     p_comm DECIMAL(10,2)
    -> )
    -> RETURNS DECIMAL(12,2)
    -> DETERMINISTIC
    -> READS SQL DATA
    -> BEGIN
    ->     DECLARE annual_compensation DECIMAL(12,2);
    ->
    ->     -- Substitute NULL values with zero
    ->     IF p_sal IS NULL THEN
    ->         SET p_sal = 0;
    ->     END IF;
    ->
    ->     IF p_comm IS NULL THEN
    ->         SET p_comm = 0;
    ->     END IF;
    ->
    ->     -- Calculate annual compensation
    ->     SET annual_compensation = (p_sal + p_comm) * 12;
    ->
    ->     RETURN annual_compensation;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
```

```
mysql> SELECT empno,
    ->        ename,
    ->        sal,
    ->        comm,
    ->        USER_ANNUAL_COMP(empno, sal, comm) AS annual_compensation
    -> FROM emp;
+-------+--------+---------+---------+----------------------+
| empno | ename  | sal     | comm    | annual_compensation  |
+-------+--------+---------+---------+----------------------+
|  7499 | ALLEN  | 1600.00 |  300.00 |             22800.00 |
|  7521 | WARD   | 1250.00 |  500.00 |             21000.00 |
|  7566 | JONES  | 3000.00 |  500.00 |             42000.00 |
|  7654 | MARTIN | 1250.00 | 1400.00 |             31800.00 |
|  7698 | BLAKE  | 2850.00 |    0.00 |             34200.00 |
|  7782 | CLARK  | 2450.00 |    0.00 |             29400.00 |
|  7788 | SCOTT  | 3000.00 |    0.00 |             36000.00 |
|  7839 | KING   | 5000.00 | 1000.00 |             72000.00 |
|  7844 | TURNER | 1500.00 |    0.00 |             18000.00 |
|  7900 | JAMES  |  950.00 |    0.00 |             11400.00 |
|  7902 | FORD   | 3000.00 |    0.00 |             36000.00 |
|  7934 | MILLER | 1300.00 |    0.00 |             15600.00 |
+-------+--------+---------+---------+----------------------+
12 rows in set (0.00 sec)
```

**9.)Create a function named USER_VALID_DEPTNO that has a single parameter p_dno to accept a department number and returns a BOOLEAN value. The function returns TRUE if the department number exists in the DEPT table else it returns FALSE**

```
mysql> DELIMITER ;
mysql> DELIMITER //
mysql>
mysql> CREATE FUNCTION USER_VALID_DEPTNO(
    ->     p_dno INT
    -> )
    -> RETURNS BOOLEAN
    -> READS SQL DATA
    -> BEGIN
    ->     DECLARE dept_exists BOOLEAN;
    ->
    ->     -- Check if the department number exists in the DEPT table
    ->     SELECT EXISTS (SELECT 1 FROM dept WHERE deptno = p_dno) INTO dept_exists;
    ->
    ->     RETURN dept_exists;
    -> END //
Query OK, 0 rows affected (0.01 sec)
```

**10.)Create a table named salaryLog with the appropriate columns and insert the empno, new grade, old salary and new salary values in salaryLog table if the grade of an employee changes. Create a trigger named TR_CHECK_GRADE that will be fired when a user modifies the EMP table. It will check whether the grade has changed by making use of the SALGRADE table. (Grade is dependent on Salary.) If grade is changed, the trigger will log the corresponding employee number, old salary, new salary and new grade into salaryLog table. Test the working of the trigger by firing an appropriate DML query.**



```
mysql> -- Sample data for EMP table
mysql> INSERT INTO EMP (EMPNO, GRADE, SALARY) VALUES (1, 1, 1500);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO EMP (EMPNO, GRADE, SALARY) VALUES (2, 2, 2500);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> -- Create salaryLog table
mysql> CREATE TABLE salaryLog (
    ->     EMPNO INT,
    ->     OLD_GRADE INT,
    ->     NEW_GRADE INT,
    ->     OLD_SALARY INT,
    ->     NEW_SALARY INT,
    ->     CHANGE_DATE TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> -- Create trigger TR_CHECK_GRADE
mysql> DELIMITER //
mysql> CREATE TRIGGER TR_CHECK_GRADE BEFORE UPDATE ON EMP
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE v_old_grade INT;
    ->     DECLARE v_new_grade INT;
    ->     DECLARE v_old_salary INT;
    ->     DECLARE v_new_salary INT;
    ->
    ->     SELECT GRADE INTO v_old_grade FROM EMP WHERE EMPNO = OLD.EMPNO;
```

```
    ->     SELECT GRADE INTO v_new_grade FROM EMP WHERE EMPNO = NEW.EMPNO;
    ->     SELECT SALARY INTO v_old_salary FROM EMP WHERE EMPNO = OLD.EMPNO;
    ->     SELECT SALARY INTO v_new_salary FROM EMP WHERE EMPNO = NEW.EMPNO;
    ->
    ->     IF v_old_grade != v_new_grade THEN
    ->         INSERT INTO salaryLog (EMPNO, OLD_GRADE, NEW_GRADE, OLD_SALARY, NEW_SALARY)
    ->         VALUES (OLD.EMPNO, v_old_grade, v_new_grade, v_old_salary, v_new_salary);
    ->     END IF;
    -> END//
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql>
mysql> -- Test the trigger by updating the grade of an employee
mysql> UPDATE EMP SET GRADE = 2 WHERE EMPNO = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```