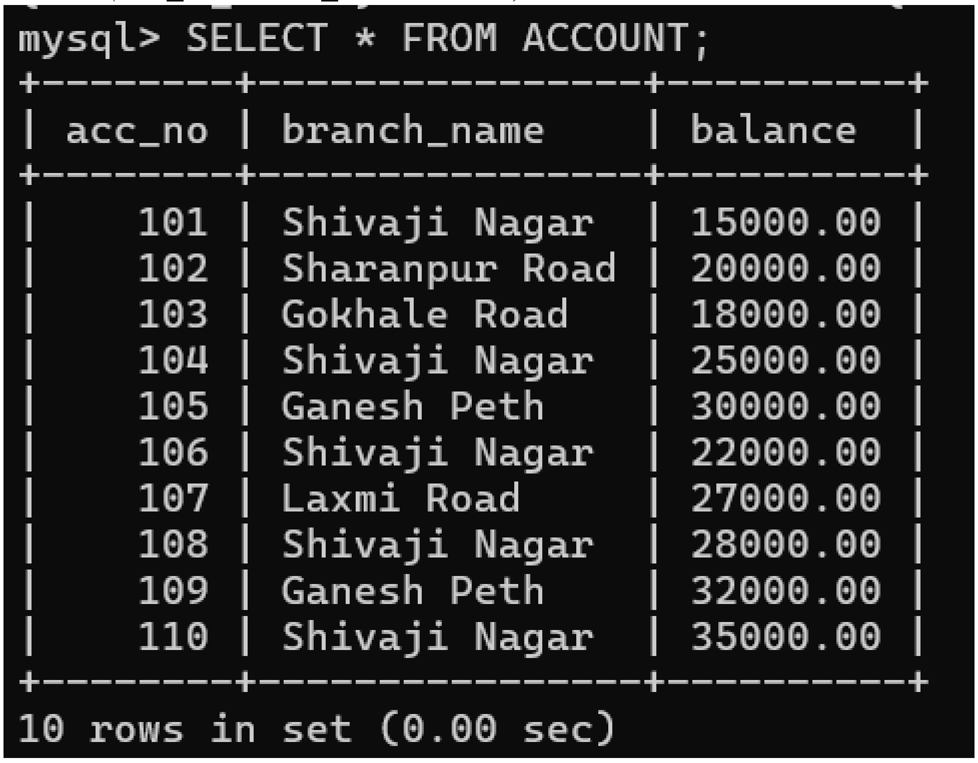
DBMS LAB ASSIGNMENT 3

NAME: NARESH SUTHAR
SRN: 202202174
ROLL NO: 69
DIV: A(A3)

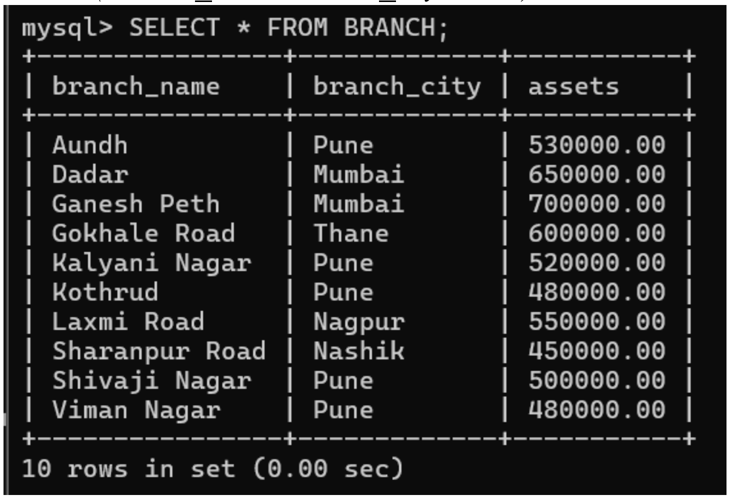
Question 1:

Consider following Database Schemas
Account(Acc_no, branch_name,balance)



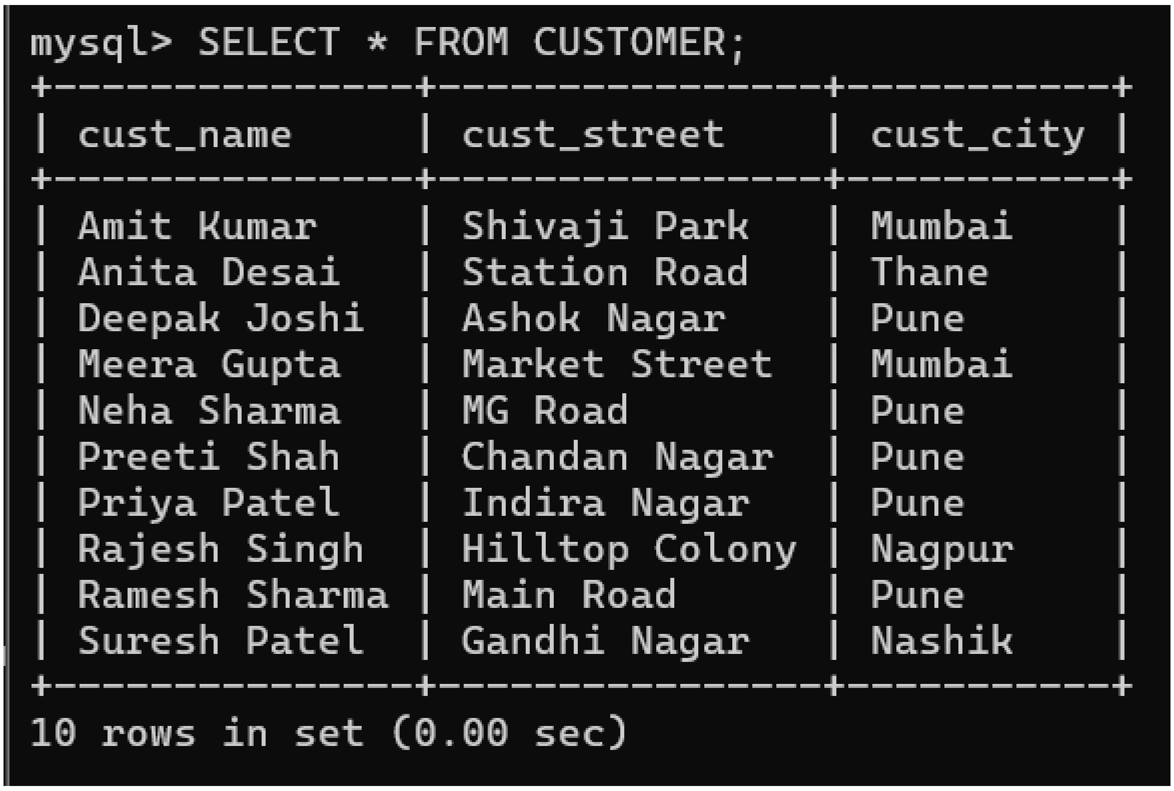
```
mysql> SELECT * FROM ACCOUNT;
+--------+-----------------+----------+
| acc_no | branch_name     | balance  |
+--------+-----------------+----------+
|    101 | Shivaji Nagar   | 15000.00 |
|    102 | Sharanpur Road  | 20000.00 |
|    103 | Gokhale Road    | 18000.00 |
|    104 | Shivaji Nagar   | 25000.00 |
|    105 | Ganesh Peth     | 30000.00 |
|    106 | Shivaji Nagar   | 22000.00 |
|    107 | Laxmi Road      | 27000.00 |
|    108 | Shivaji Nagar   | 28000.00 |
|    109 | Ganesh Peth     | 32000.00 |
|    110 | Shivaji Nagar   | 35000.00 |
+--------+-----------------+----------+
10 rows in set (0.00 sec)
```

branch(branch_name,branch_city,assets)



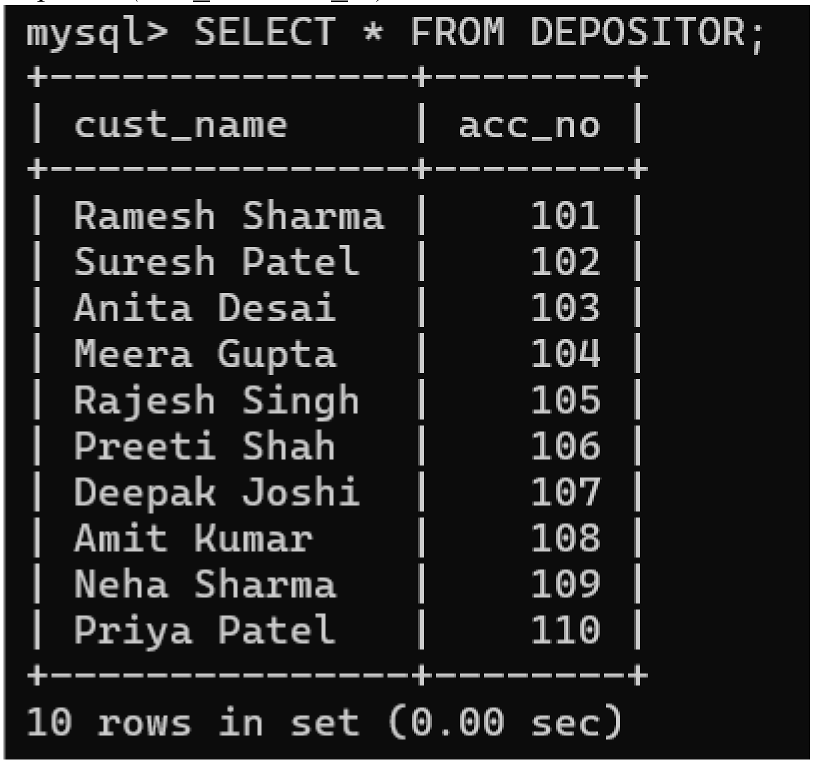
```
mysql> SELECT * FROM BRANCH;
+----------------+-------------+-----------+
| branch_name    | branch_city | assets    |
+----------------+-------------+-----------+
| Aundh          | Pune        | 530000.00 |
| Dadar          | Mumbai      | 650000.00 |
| Ganesh Peth    | Mumbai      | 700000.00 |
| Gokhale Road   | Thane       | 600000.00 |
| Kalyani Nagar  | Pune        | 520000.00 |
| Kothrud        | Pune        | 480000.00 |
| Laxmi Road     | Nagpur      | 550000.00 |
| Sharanpur Road | Nashik      | 450000.00 |
| Shivaji Nagar  | Pune        | 500000.00 |
| Viman Nagar    | Pune        | 480000.00 |
+----------------+-------------+-----------+
10 rows in set (0.00 sec)
```

customer(cust_name,cust_street,cust_city)



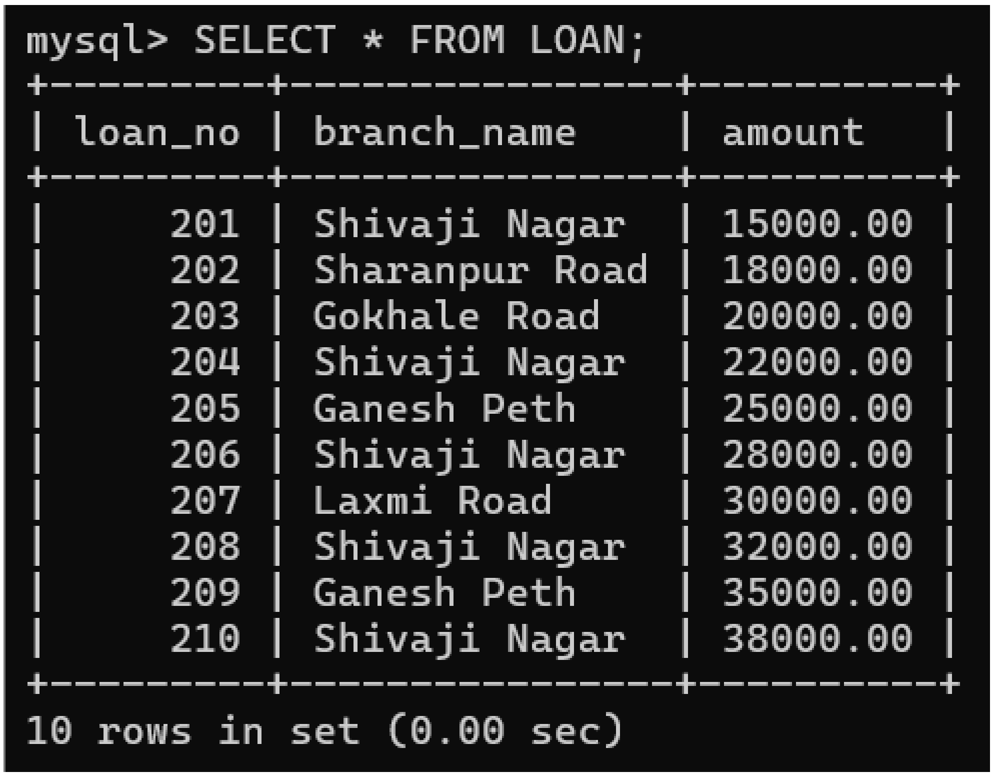
```
mysql> SELECT * FROM CUSTOMER;
+---------------+----------------+-----------+
| cust_name     | cust_street    | cust_city |
+---------------+----------------+-----------+
| Amit Kumar    | Shivaji Park   | Mumbai    |
| Anita Desai   | Station Road   | Thane     |
| Deepak Joshi  | Ashok Nagar    | Pune      |
| Meera Gupta   | Market Street  | Mumbai    |
| Neha Sharma   | MG Road        | Pune      |
| Preeti Shah   | Chandan Nagar  | Pune      |
| Priya Patel   | Indira Nagar   | Pune      |
| Rajesh Singh  | Hilltop Colony | Nagpur    |
| Ramesh Sharma | Main Road      | Pune      |
| Suresh Patel  | Gandhi Nagar   | Nashik    |
+---------------+----------------+-----------+
10 rows in set (0.00 sec)
```

Depositor(cust_name,acc_no)



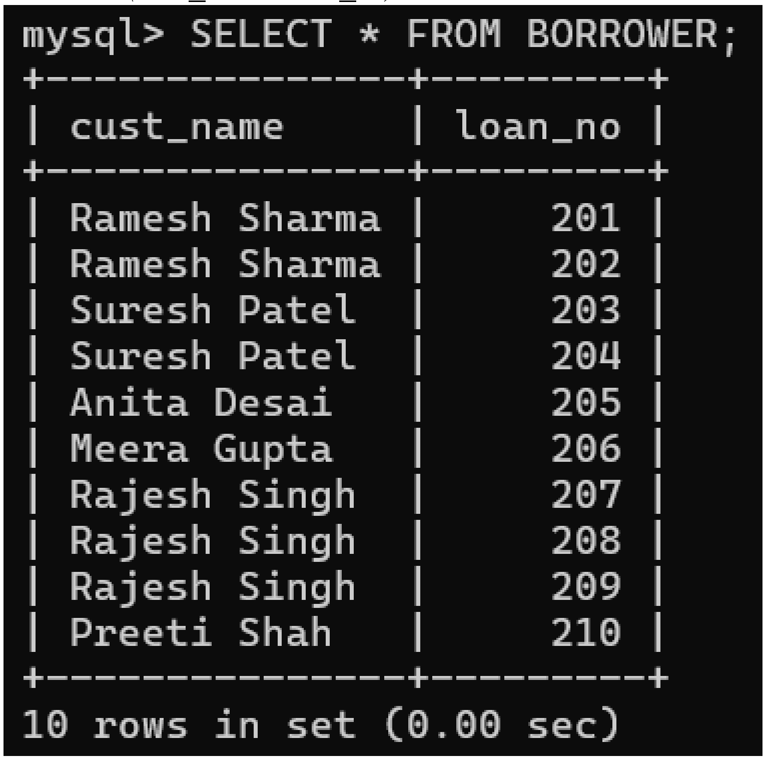
```
mysql> SELECT * FROM DEPOSITOR;
+---------------+--------+
| cust_name     | acc_no |
+---------------+--------+
| Ramesh Sharma |    101 |
| Suresh Patel  |    102 |
| Anita Desai   |    103 |
| Meera Gupta   |    104 |
| Rajesh Singh  |    105 |
| Preeti Shah   |    106 |
| Deepak Joshi  |    107 |
| Amit Kumar    |    108 |
| Neha Sharma   |    109 |
| Priya Patel   |    110 |
+---------------+--------+
10 rows in set (0.00 sec)
```

Loan(loan_no,branch_name,amount)



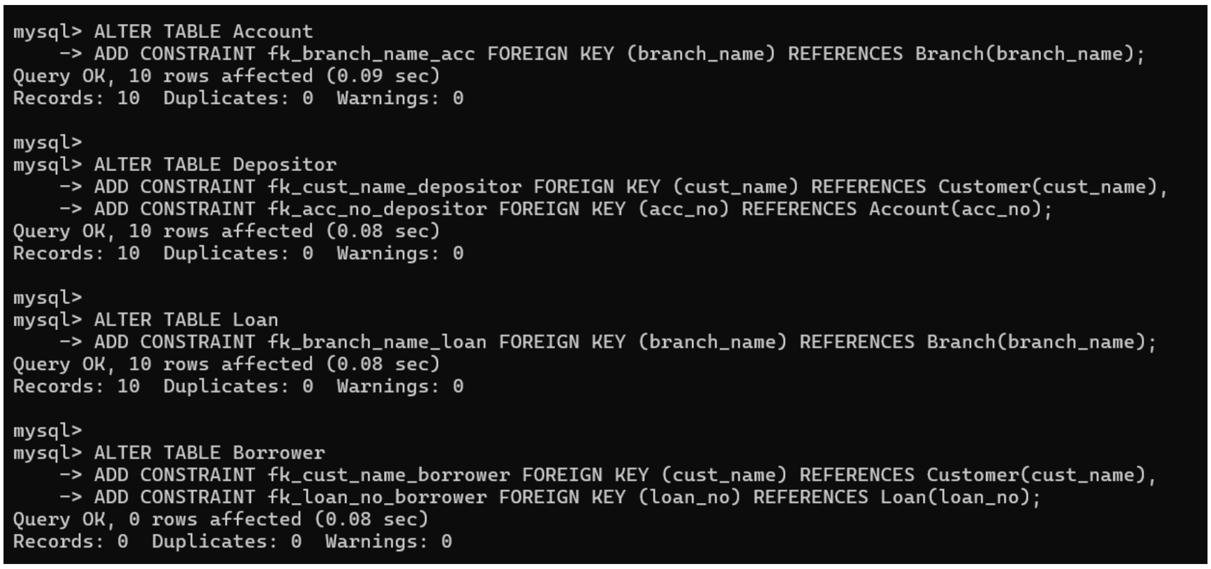
```
mysql> SELECT * FROM LOAN;
+---------+----------------+----------+
| loan_no | branch_name    | amount   |
+---------+----------------+----------+
|     201 | Shivaji Nagar  | 15000.00 |
|     202 | Sharanpur Road | 18000.00 |
|     203 | Gokhale Road   | 20000.00 |
|     204 | Shivaji Nagar  | 22000.00 |
|     205 | Ganesh Peth    | 25000.00 |
|     206 | Shivaji Nagar  | 28000.00 |
|     207 | Laxmi Road     | 30000.00 |
|     208 | Shivaji Nagar  | 32000.00 |
|     209 | Ganesh Peth    | 35000.00 |
|     210 | Shivaji Nagar  | 38000.00 |
+---------+----------------+----------+
10 rows in set (0.00 sec)
```

Borrower(cust_name,loan_no)



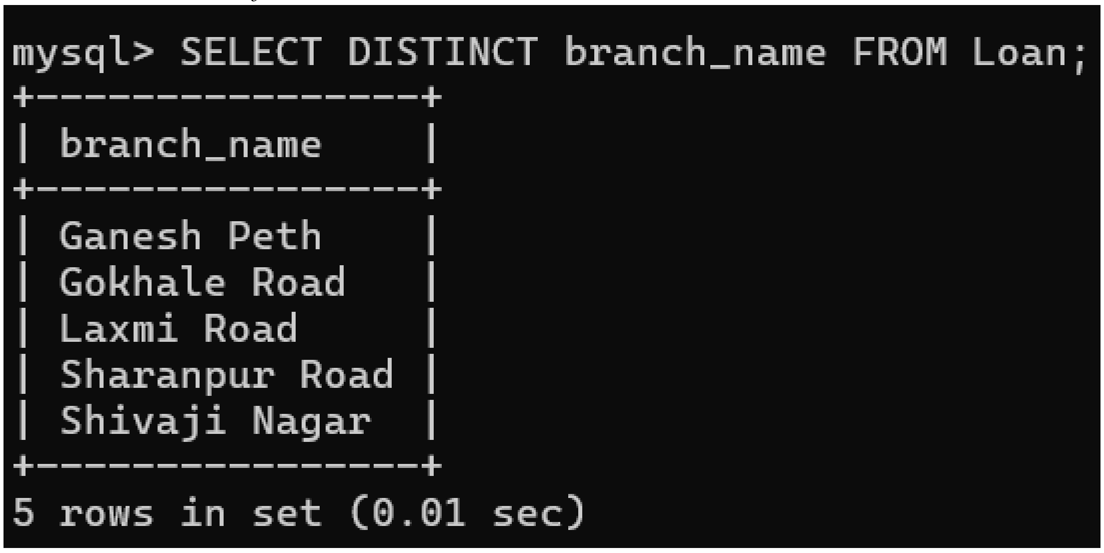
```
mysql> SELECT * FROM BORROWER;
+---------------+---------+
| cust_name     | loan_no |
+---------------+---------+
| Ramesh Sharma |     201 |
| Ramesh Sharma |     202 |
| Suresh Patel  |     203 |
| Suresh Patel  |     204 |
| Anita Desai   |     205 |
| Meera Gupta   |     206 |
| Rajesh Singh  |     207 |
| Rajesh Singh  |     208 |
| Rajesh Singh  |     209 |
| Preeti Shah   |     210 |
+---------------+---------+
10 rows in set (0.00 sec)
```

Solve following query:
Create above tables with appropriate constraints like primary key, foreign key, check constrains,
not null etc.



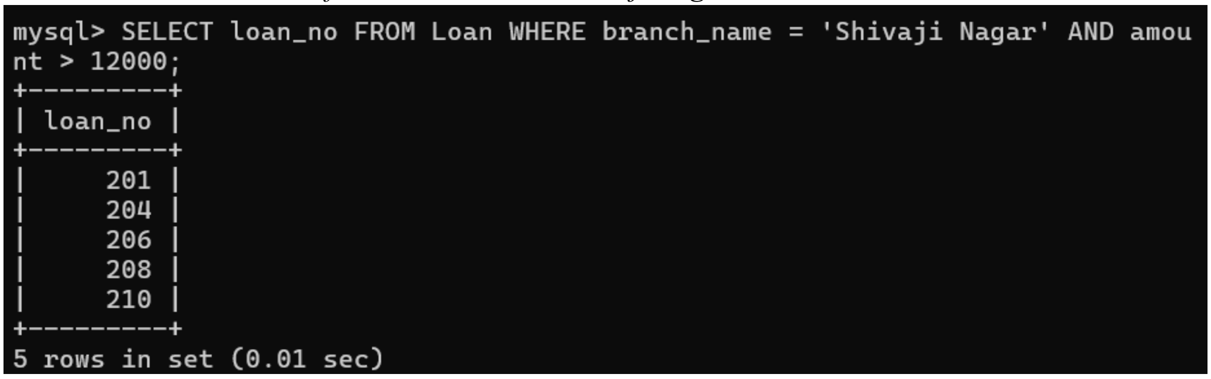
```
mysql> ALTER TABLE Account
    -> ADD CONSTRAINT fk_branch_name_acc FOREIGN KEY (branch_name) REFERENCES Branch(branch_name);
Query OK, 10 rows affected (0.09 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Depositor
    -> ADD CONSTRAINT fk_cust_name_depositor FOREIGN KEY (cust_name) REFERENCES Customer(cust_name),
    -> ADD CONSTRAINT fk_acc_no_depositor FOREIGN KEY (acc_no) REFERENCES Account(acc_no);
Query OK, 10 rows affected (0.08 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Loan
    -> ADD CONSTRAINT fk_branch_name_loan FOREIGN KEY (branch_name) REFERENCES Branch(branch_name);
Query OK, 10 rows affected (0.08 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Borrower
    -> ADD CONSTRAINT fk_cust_name_borrower FOREIGN KEY (cust_name) REFERENCES Customer(cust_name),
    -> ADD CONSTRAINT fk_loan_no_borrower FOREIGN KEY (loan_no) REFERENCES Loan(loan_no);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

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



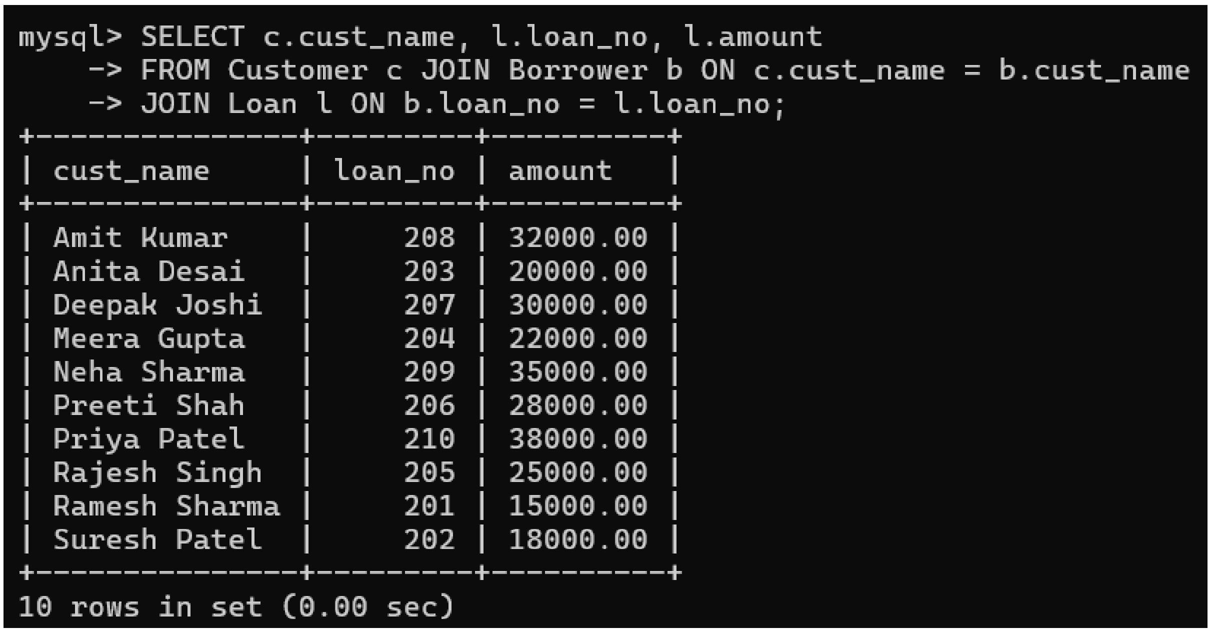
```
mysql> SELECT DISTINCT branch_name FROM Loan;
+----------------+
| branch_name    |
+----------------+
| Ganesh Peth    |
| Gokhale Road   |
| Laxmi Road     |
| Sharanpur Road |
| Shivaji Nagar  |
+----------------+
5 rows in set (0.01 sec)
```

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



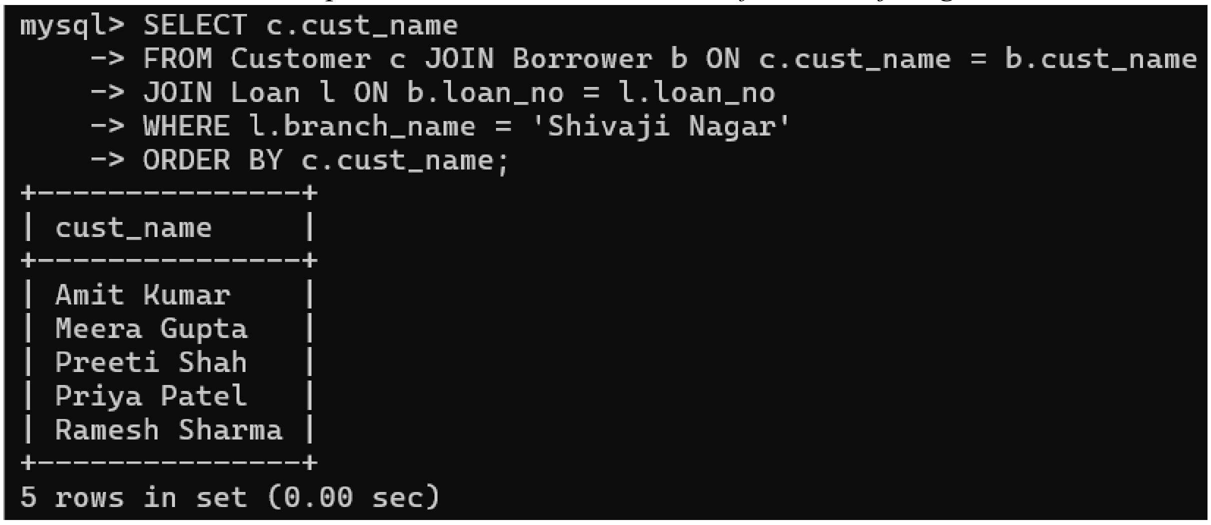
```
mysql> SELECT loan_no FROM Loan WHERE branch_name = 'Shivaji Nagar' AND amou
nt > 12000;
+---------+
| loan_no |
+---------+
|     201 |
|     204 |
|     206 |
|     208 |
|     210 |
+---------+
5 rows in set (0.01 sec)
```

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



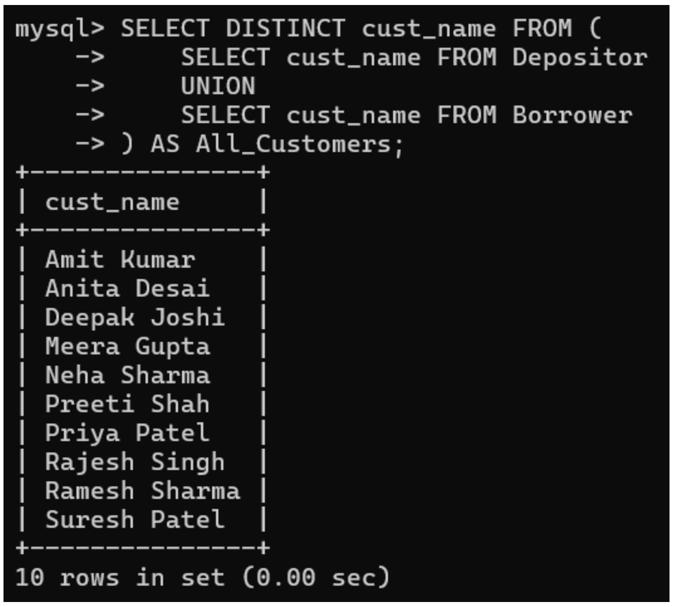
```
mysql> SELECT c.cust_name, l.loan_no, l.amount
    -> FROM Customer c JOIN Borrower b ON c.cust_name = b.cust_name
    -> JOIN Loan l ON b.loan_no = l.loan_no;
+---------------+---------+----------+
| cust_name     | loan_no | amount   |
+---------------+---------+----------+
| Amit Kumar    |     208 | 32000.00 |
| Anita Desai   |     203 | 20000.00 |
| Deepak Joshi  |     207 | 30000.00 |
| Meera Gupta   |     204 | 22000.00 |
| Neha Sharma   |     209 | 35000.00 |
| Preeti Shah   |     206 | 28000.00 |
| Priya Patel   |     210 | 38000.00 |
| Rajesh Singh  |     205 | 25000.00 |
| Ramesh Sharma |     201 | 15000.00 |
| Suresh Patel  |     202 | 18000.00 |
+---------------+---------+----------+
10 rows in set (0.00 sec)
```

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



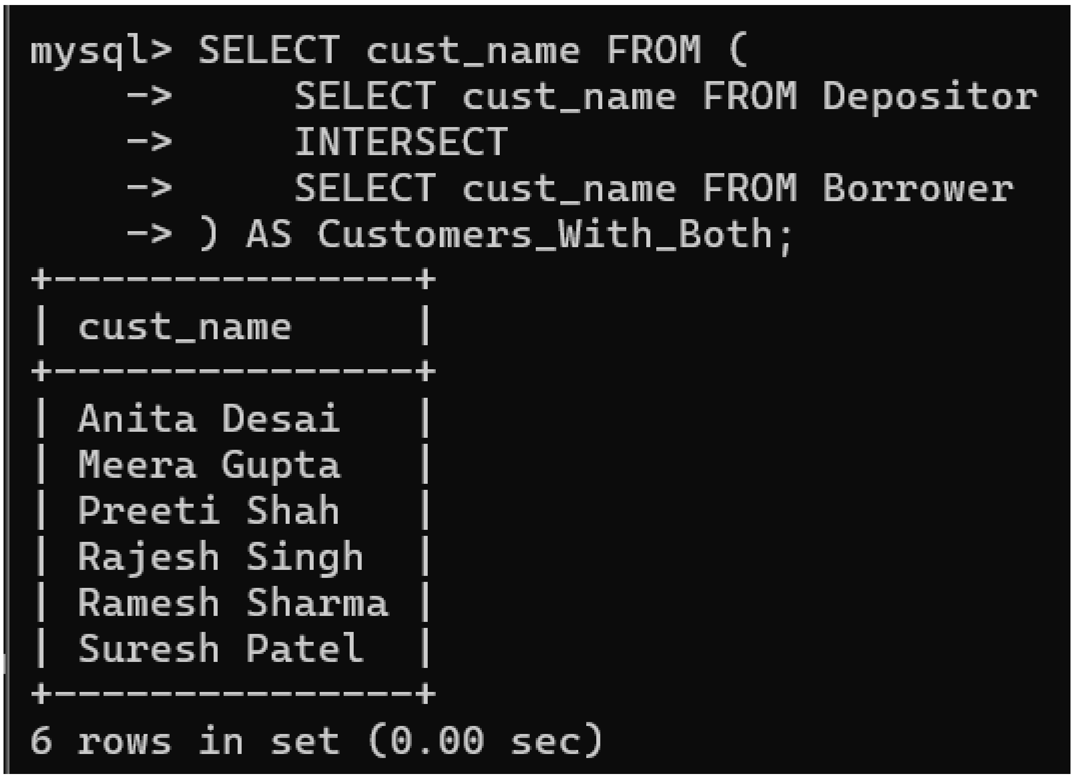
```
mysql> SELECT c.cust_name
    -> FROM Customer c JOIN Borrower b ON c.cust_name = b.cust_name
    -> JOIN Loan l ON b.loan_no = l.loan_no
    -> WHERE l.branch_name = 'Shivaji Nagar'
    -> ORDER BY c.cust_name;
+---------------+
| cust_name     |
+---------------+
| Amit Kumar    |
| Meera Gupta   |
| Preeti Shah   |
| Priya Patel   |
| Ramesh Sharma |
+---------------+
5 rows in set (0.00 sec)
```

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



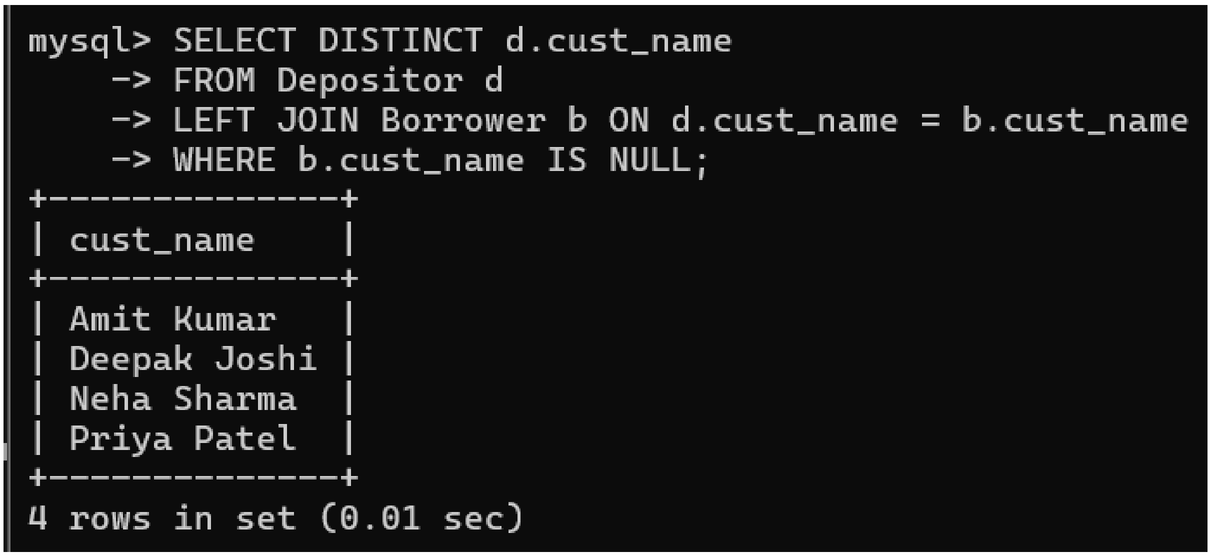
```
mysql> SELECT DISTINCT cust_name FROM (
    ->     SELECT cust_name FROM Depositor
    ->     UNION
    ->     SELECT cust_name FROM Borrower
    -> ) AS All_Customers;
+---------------+
| cust_name     |
+---------------+
| Amit Kumar    |
| Anita Desai   |
| Deepak Joshi  |
| Meera Gupta   |
| Neha Sharma   |
| Preeti Shah   |
| Priya Patel   |
| Rajesh Singh  |
| Ramesh Sharma |
| Suresh Patel  |
+---------------+
10 rows in set (0.00 sec)
```

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



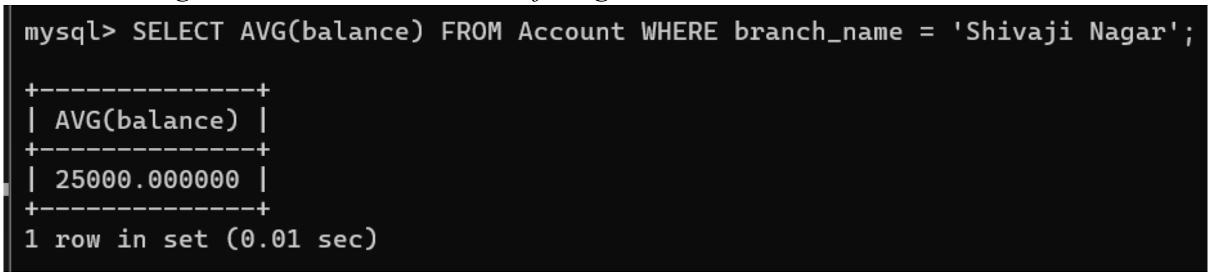
```
mysql> SELECT cust_name FROM (
    ->     SELECT cust_name FROM Depositor
    ->     INTERSECT
    ->     SELECT cust_name FROM Borrower
    -> ) AS Customers_With_Both;
+---------------+
| cust_name     |
+---------------+
| Anita Desai   |
| Meera Gupta   |
| Preeti Shah   |
| Rajesh Singh  |
| Ramesh Sharma |
| Suresh Patel  |
+---------------+
6 rows in set (0.00 sec)
```

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



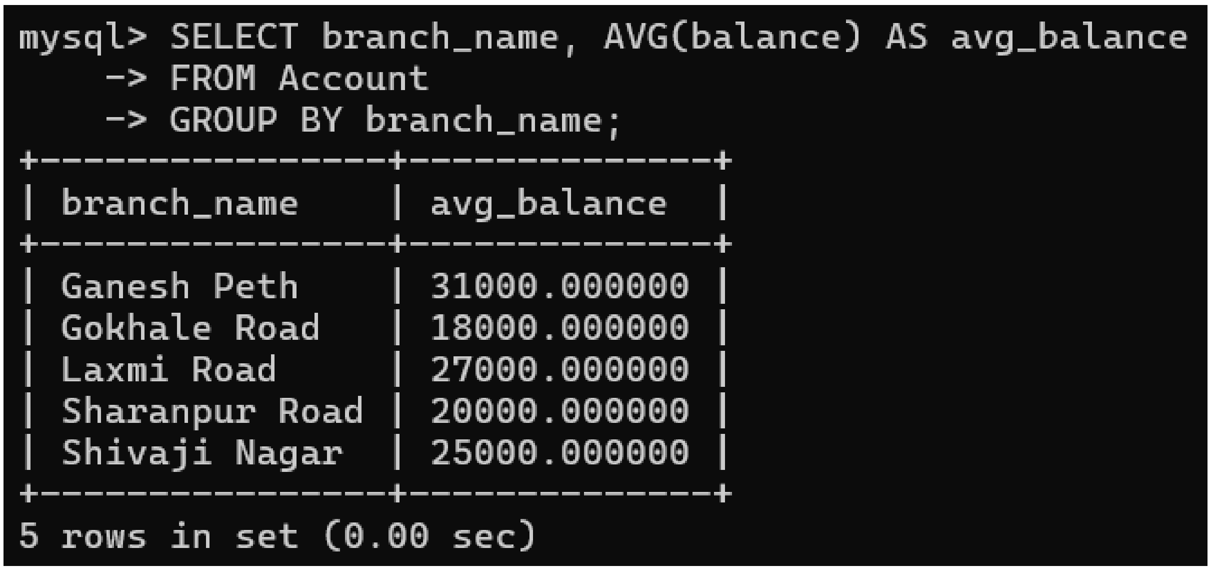
```
mysql> SELECT DISTINCT d.cust_name
    -> FROM Depositor d
    -> LEFT JOIN Borrower b ON d.cust_name = b.cust_name
    -> WHERE b.cust_name IS NULL;
+--------------+
| cust_name    |
+--------------+
| Amit Kumar   |
| Deepak Joshi |
| Neha Sharma  |
| Priya Patel  |
+--------------+
4 rows in set (0.01 sec)
```

8. Find average account balance at Shivaji nagar branch.



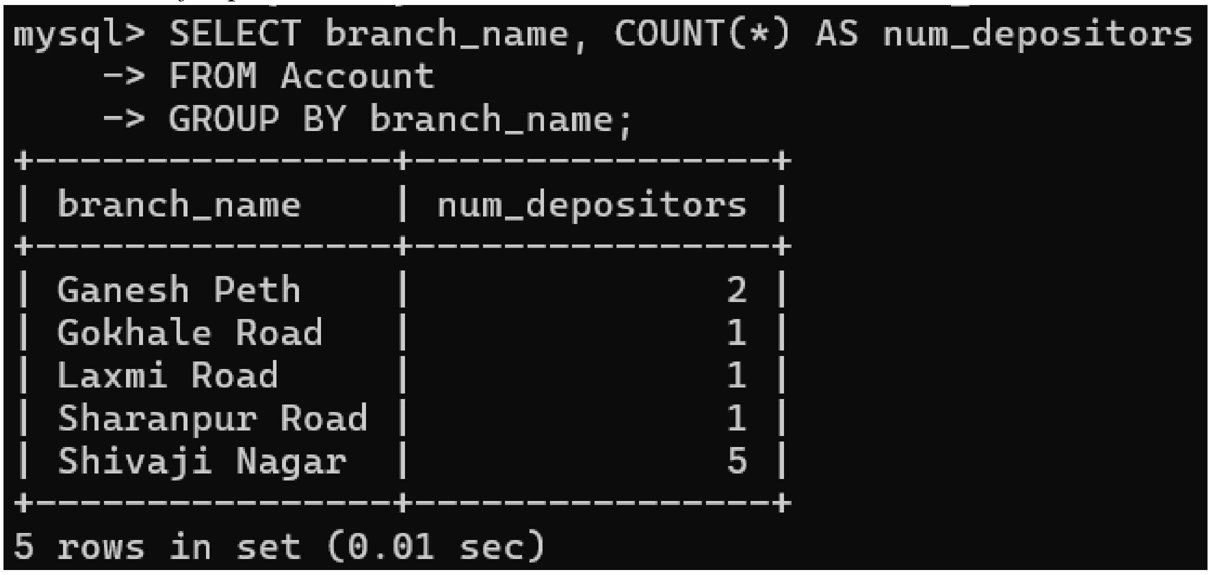
```
mysql> SELECT AVG(balance) FROM Account WHERE branch_name = 'Shivaji Nagar';

+--------------+
| AVG(balance) |
+--------------+
| 25000.000000 |
+--------------+
1 row in set (0.01 sec)
```

9. Find the average account balance at each branch



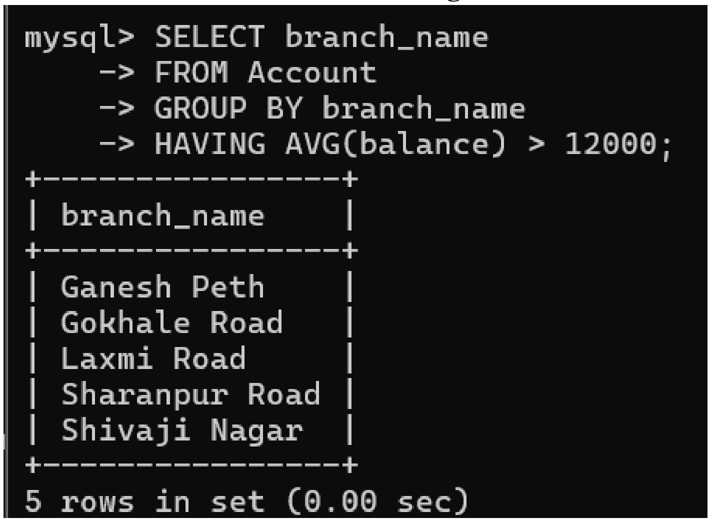
```
mysql> SELECT branch_name, AVG(balance) AS avg_balance
    -> FROM Account
    -> GROUP BY branch_name;
+----------------+--------------+
| branch_name    | avg_balance  |
+----------------+--------------+
| Ganesh Peth    | 31000.000000 |
| Gokhale Road   | 18000.000000 |
| Laxmi Road     | 27000.000000 |
| Sharanpur Road | 20000.000000 |
| Shivaji Nagar  | 25000.000000 |
+----------------+--------------+
5 rows in set (0.00 sec)
```

10. Find no. of depositors at each branch.



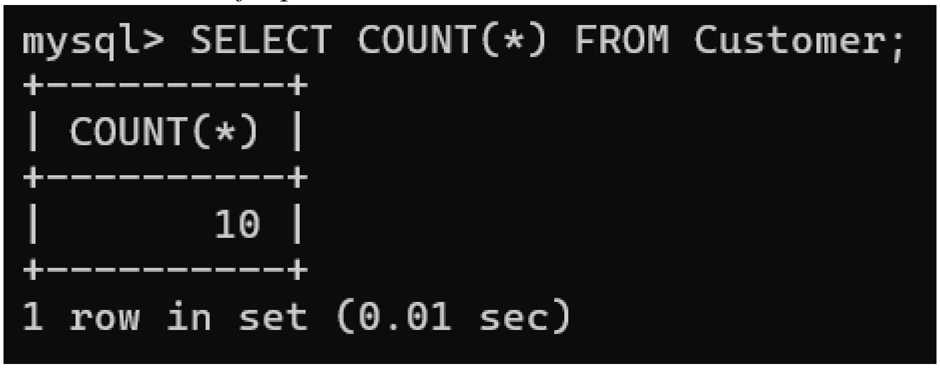
```
mysql> SELECT branch_name, COUNT(*) AS num_depositors
    -> FROM Account
    -> GROUP BY branch_name;
+----------------+----------------+
| branch_name    | num_depositors |
+----------------+----------------+
| Ganesh Peth    |              2 |
| Gokhale Road   |              1 |
| Laxmi Road     |              1 |
| Sharanpur Road |              1 |
| Shivaji Nagar  |              5 |
+----------------+----------------+
5 rows in set (0.01 sec)
```

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



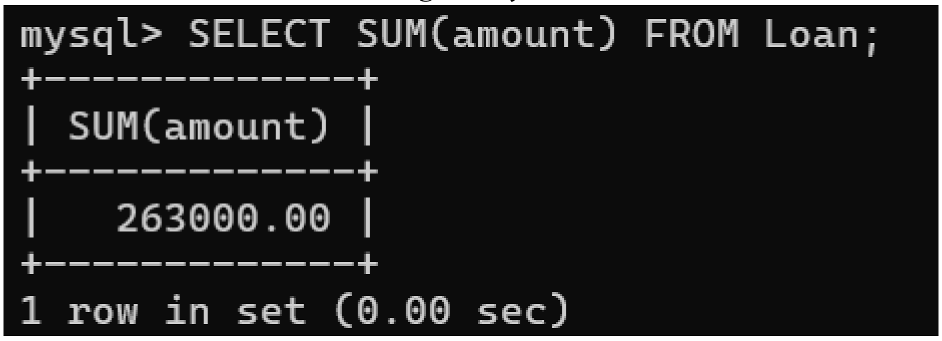
```
mysql> SELECT branch_name
    -> FROM Account
    -> GROUP BY branch_name
    -> HAVING AVG(balance) > 12000;
+----------------+
| branch_name    |
+----------------+
| Ganesh Peth    |
| Gokhale Road   |
| Laxmi Road     |
| Sharanpur Road |
| Shivaji Nagar  |
+----------------+
5 rows in set (0.00 sec)
```

12. Find number of tuples in customer relation.



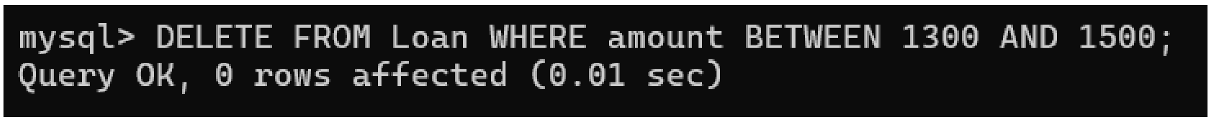
```
mysql> SELECT COUNT(*) FROM Customer;
+----------+
| COUNT(*) |
+----------+
|       10 |
+----------+
1 row in set (0.01 sec)
```

13. Calculate total loan amount given by bank.



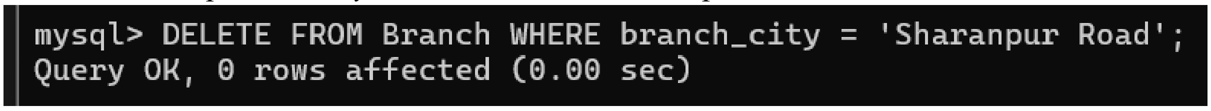
```
mysql> SELECT SUM(amount) FROM Loan;
+-------------+
| SUM(amount) |
+-------------+
|   263000.00 |
+-------------+
1 row in set (0.00 sec)
```

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)
```

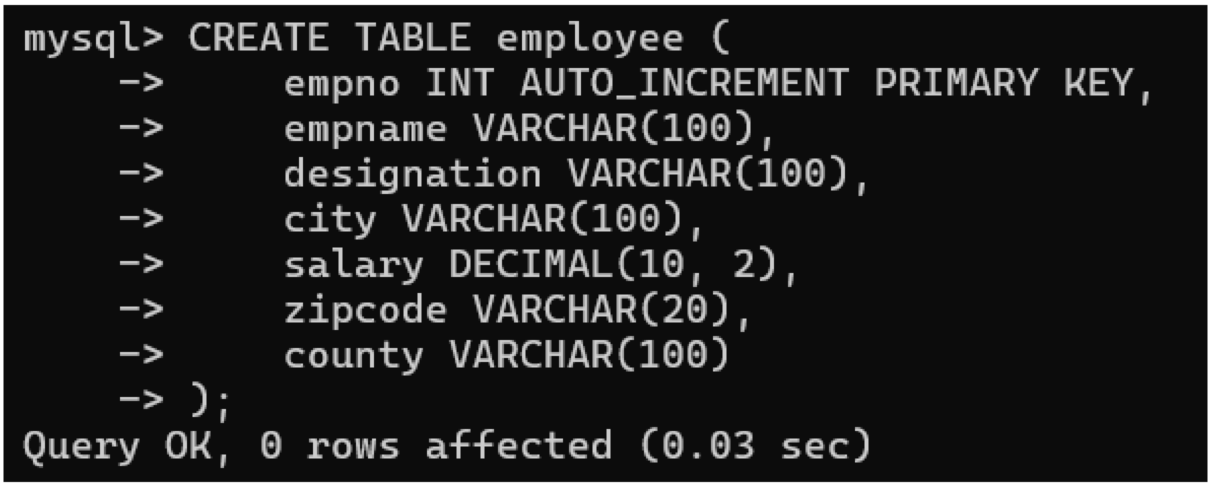
15. Delete all tuples at every branch located in Sharanpur road



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

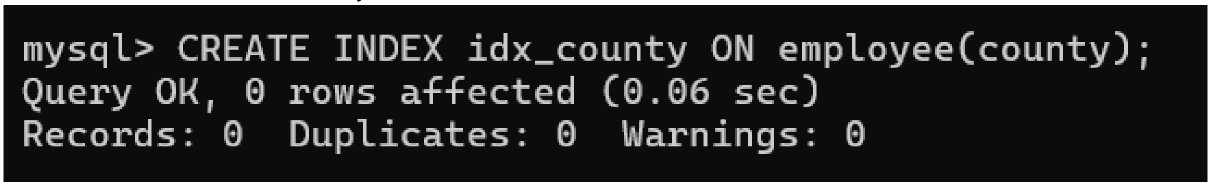
Question 2:

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.



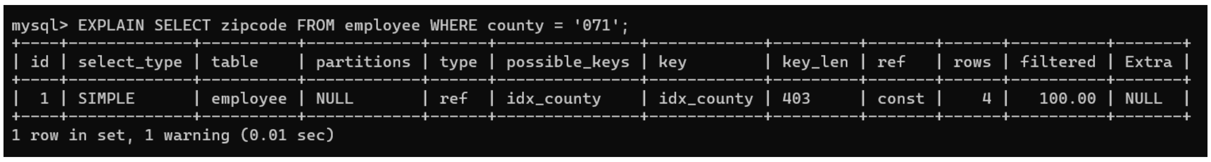
```
mysql> CREATE TABLE employee (
    ->     empno INT AUTO_INCREMENT PRIMARY KEY,
    ->     empname VARCHAR(100),
    ->     designation VARCHAR(100),
    ->     city VARCHAR(100),
    ->     salary DECIMAL(10, 2),
    ->     zipcode VARCHAR(20),
    ->     county VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

2. Create an Index on county.



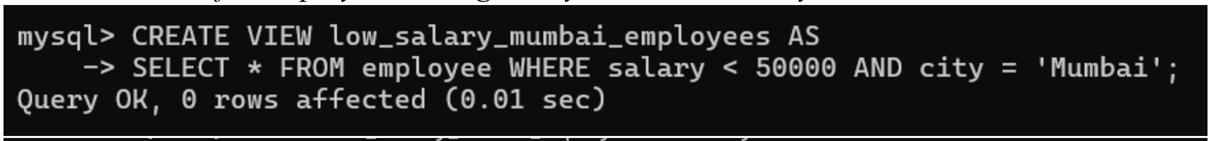
```
mysql> CREATE INDEX idx_county ON employee(county);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

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

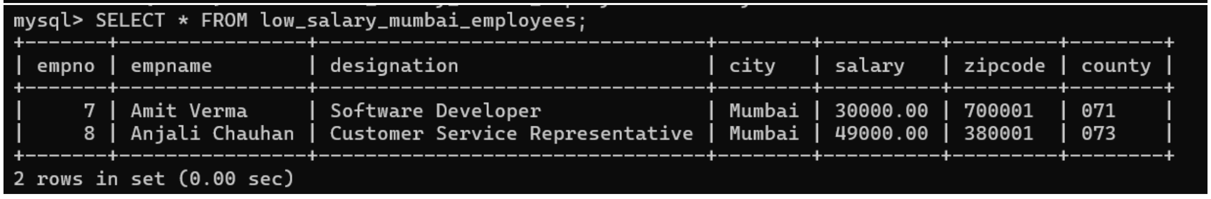


```
mysql> EXPLAIN SELECT zipcode FROM employee WHERE county = '071';
+----+-------------+----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
| id | select_type | table    | partitions | type | possible_keys | key        | key_len | ref   | rows | filtered | Extra |
+----+-------------+----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | employee | NULL       | ref  | idx_county    | idx_county | 403     | const |    4 |   100.00 | NULL  |
+----+-------------+----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

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



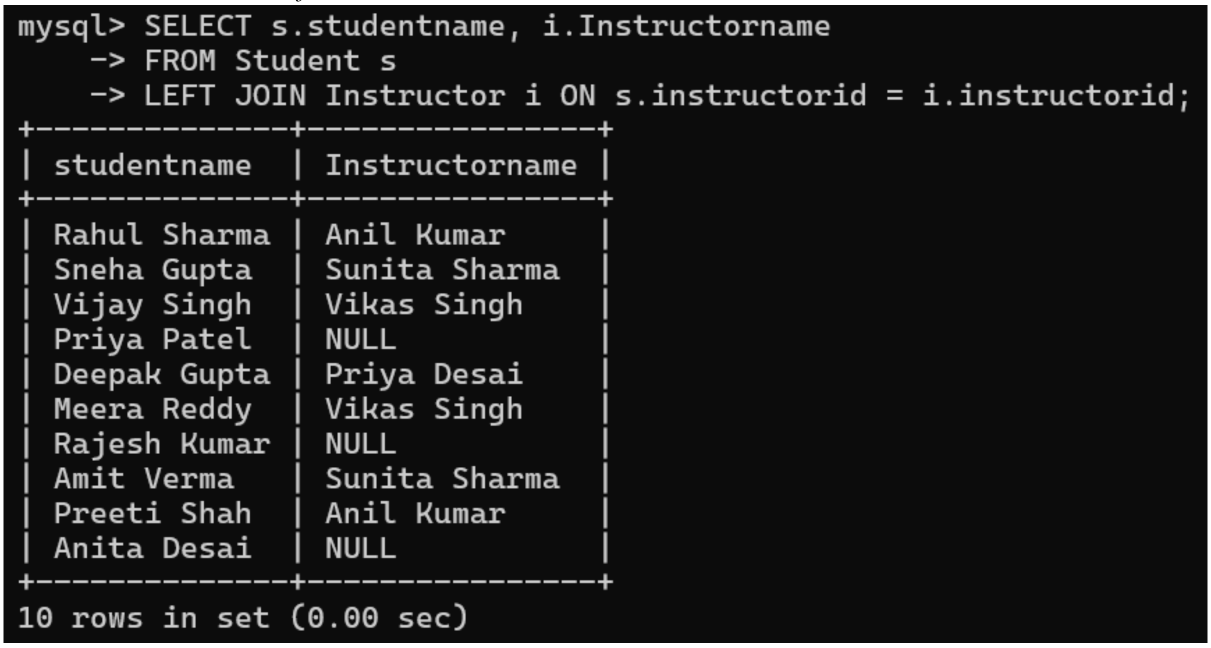
```
mysql> CREATE VIEW low_salary_mumbai_employees AS
    -> SELECT * FROM employee WHERE salary < 50000 AND city = 'Mumbai';
Query OK, 0 rows affected (0.01 sec)
```



```
mysql> SELECT * FROM low_salary_mumbai_employees;
+-------+----------------+---------------------------------+--------+----------+---------+--------+
| empno | empname        | designation                     | city   | salary   | zipcode | county |
+-------+----------------+---------------------------------+--------+----------+---------+--------+
|     7 | Amit Verma     | Software Developer              | Mumbai | 30000.00 | 700001  | 071    |
|     8 | Anjali Chauhan | Customer Service Representative | Mumbai | 49000.00 | 380001  | 073    |
+-------+----------------+---------------------------------+--------+----------+---------+--------+
2 rows in set (0.00 sec)
```

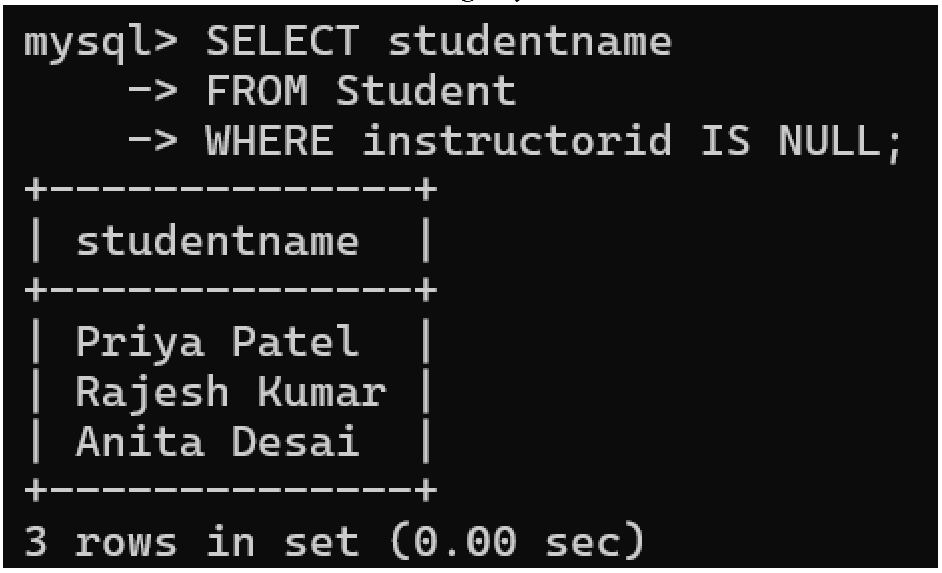
Question 3:

Consider the given database schema:
Student (studentid , studentname,instructorid,studentcity)
Instructor(instructorid,Instructorname,instructorcity,specialization)
Use all types of Joins
1. Find the instructor of each student.



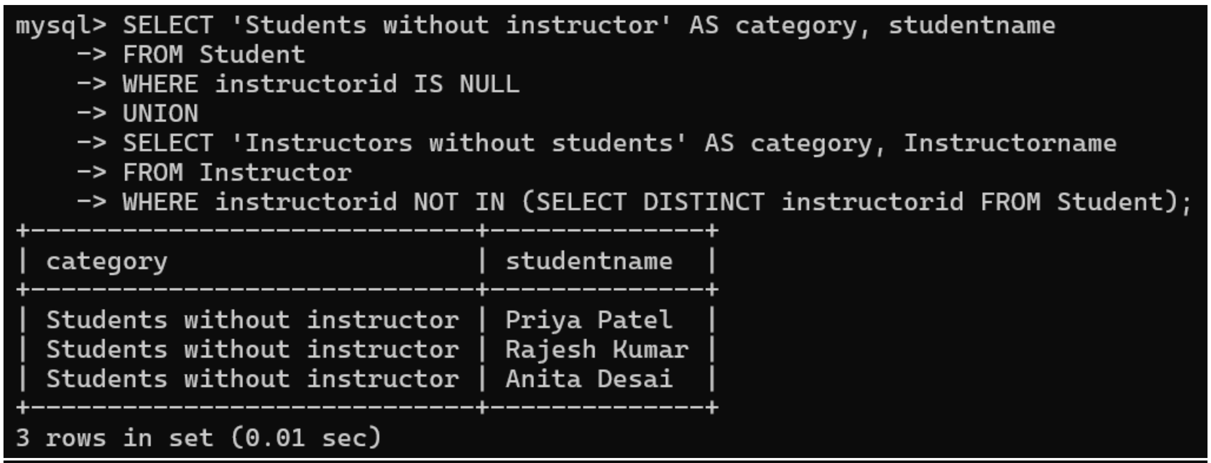
```
mysql> SELECT s.studentname, i.Instructorname
    -> FROM Student s
    -> LEFT JOIN Instructor i ON s.instructorid = i.instructorid;
+--------------+----------------+
| studentname  | Instructorname |
+--------------+----------------+
| Rahul Sharma | Anil Kumar     |
| Sneha Gupta  | Sunita Sharma  |
| Vijay Singh  | Vikas Singh    |
| Priya Patel  | NULL           |
| Deepak Gupta | Priya Desai    |
| Meera Reddy  | Vikas Singh    |
| Rajesh Kumar | NULL           |
| Amit Verma   | Sunita Sharma  |
| Preeti Shah  | Anil Kumar     |
| Anita Desai  | NULL           |
+--------------+----------------+
10 rows in set (0.00 sec)
```

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

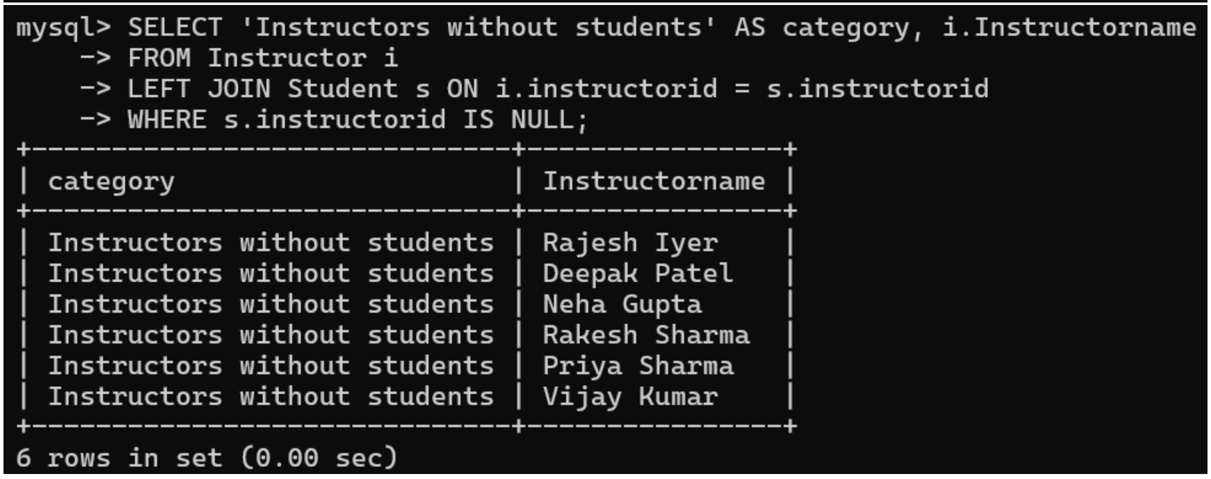


```
mysql> SELECT studentname
    -> FROM Student
    -> WHERE instructorid IS NULL;
+--------------+
| studentname  |
+--------------+
| Priya Patel  |
| Rajesh Kumar |
| Anita Desai  |
+--------------+
3 rows in set (0.00 sec)
```

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

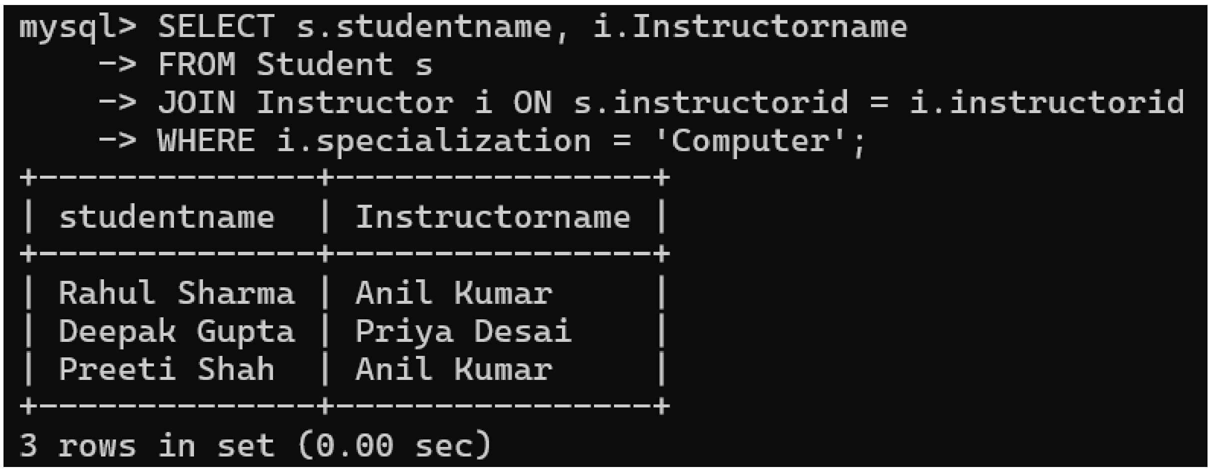


```
mysql> SELECT 'Students without instructor' AS category, studentname
    -> FROM Student
    -> WHERE instructorid IS NULL
    -> UNION
    -> SELECT 'Instructors without students' AS category, Instructorname
    -> FROM Instructor
    -> WHERE instructorid NOT IN (SELECT DISTINCT instructorid FROM Student);
+-----------------------------+--------------+
| category                    | studentname  |
+-----------------------------+--------------+
| Students without instructor | Priya Patel  |
| Students without instructor | Rajesh Kumar |
| Students without instructor | Anita Desai  |
+-----------------------------+--------------+
3 rows in set (0.01 sec)
```



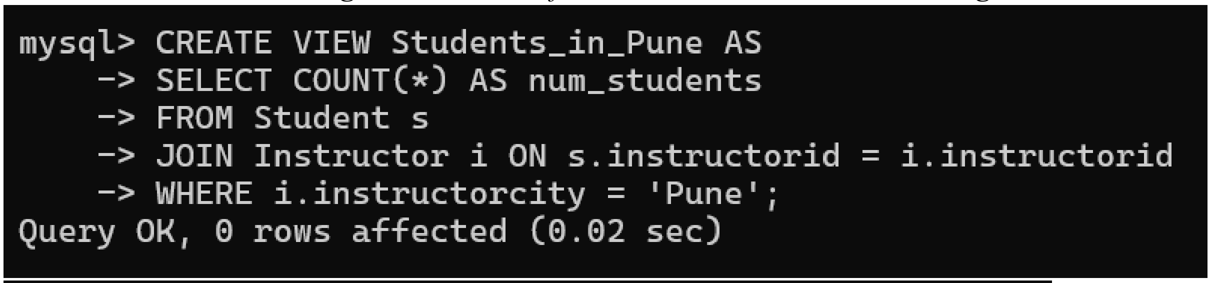
```
mysql> SELECT 'Instructors without students' AS category, i.Instructorname
    -> FROM Instructor i
    -> LEFT JOIN Student s ON i.instructorid = s.instructorid
    -> WHERE s.instructorid IS NULL;
+------------------------------+----------------+
| category                     | Instructorname |
+------------------------------+----------------+
| Instructors without students | Rajesh Iyer    |
| Instructors without students | Deepak Patel   |
| Instructors without students | Neha Gupta     |
| Instructors without students | Rakesh Sharma  |
| Instructors without students | Priya Sharma   |
| Instructors without students | Vijay Kumar    |
+------------------------------+----------------+
6 rows in set (0.00 sec)
```

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

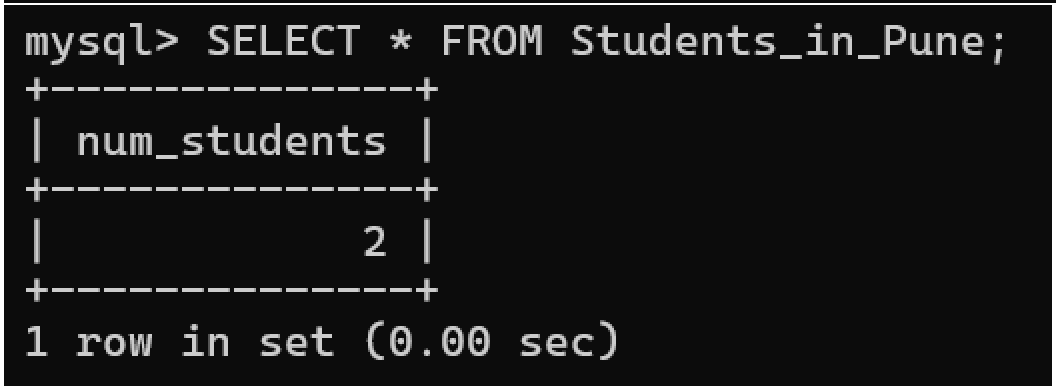


```
mysql> SELECT s.studentname, i.Instructorname
    -> FROM Student s
    -> JOIN Instructor i ON s.instructorid = i.instructorid
    -> WHERE i.specialization = 'Computer';
+--------------+----------------+
| studentname  | Instructorname |
+--------------+----------------+
| Rahul Sharma | Anil Kumar     |
| Deepak Gupta | Priya Desai    |
| Preeti Shah  | Anil Kumar     |
+--------------+----------------+
3 rows in set (0.00 sec)
```

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



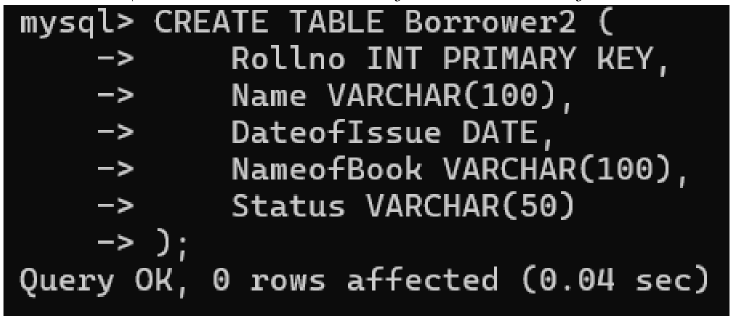
```
mysql> CREATE VIEW Students_in_Pune AS
    -> SELECT COUNT(*) AS num_students
    -> FROM Student s
    -> JOIN Instructor i ON s.instructorid = i.instructorid
    -> WHERE i.instructorcity = 'Pune';
Query OK, 0 rows affected (0.02 sec)
```



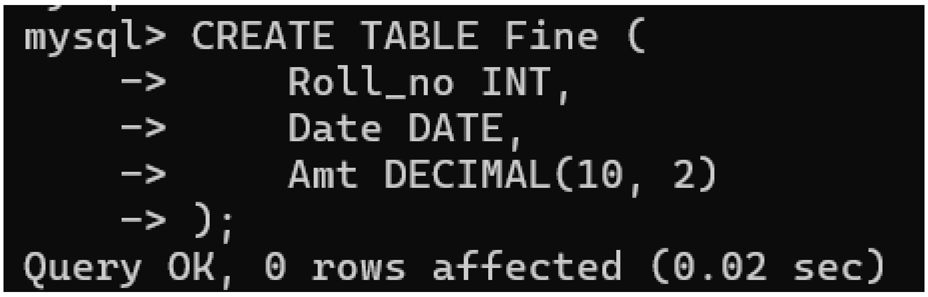
```
mysql> SELECT * FROM Students_in_Pune;
+--------------+
| num_students |
+--------------+
|            2 |
+--------------+
1 row in set (0.00 sec)
```

Question 4:

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

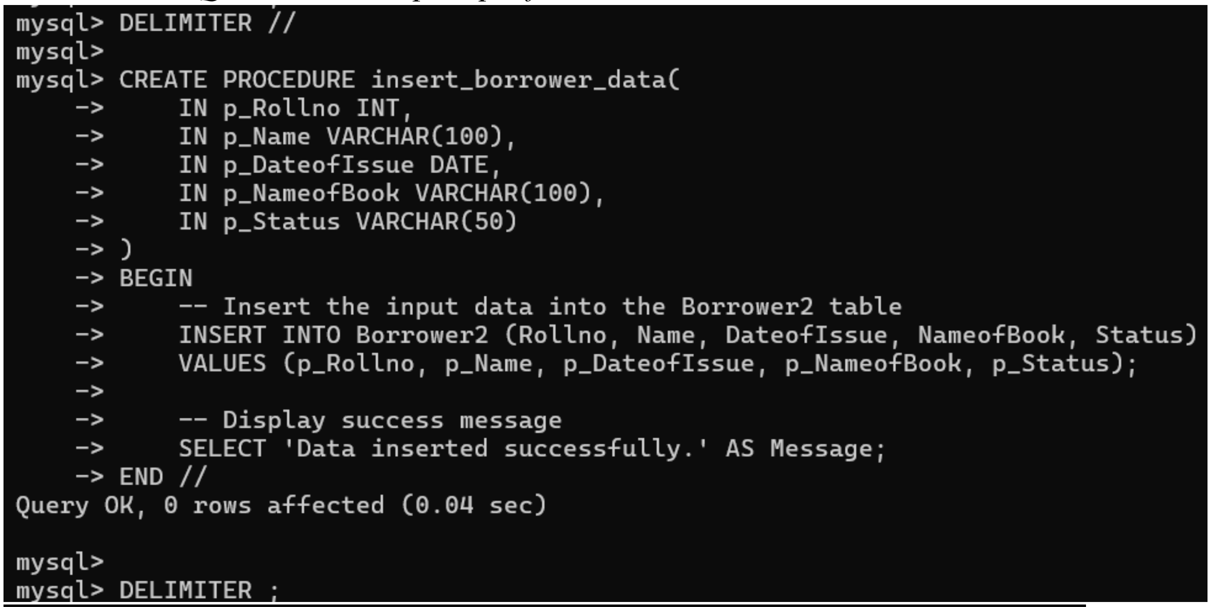


```
mysql> CREATE TABLE Borrower2 (
    ->     Rollno INT PRIMARY KEY,
    ->     Name VARCHAR(100),
    ->     DateofIssue DATE,
    ->     NameofBook VARCHAR(100),
    ->     Status VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

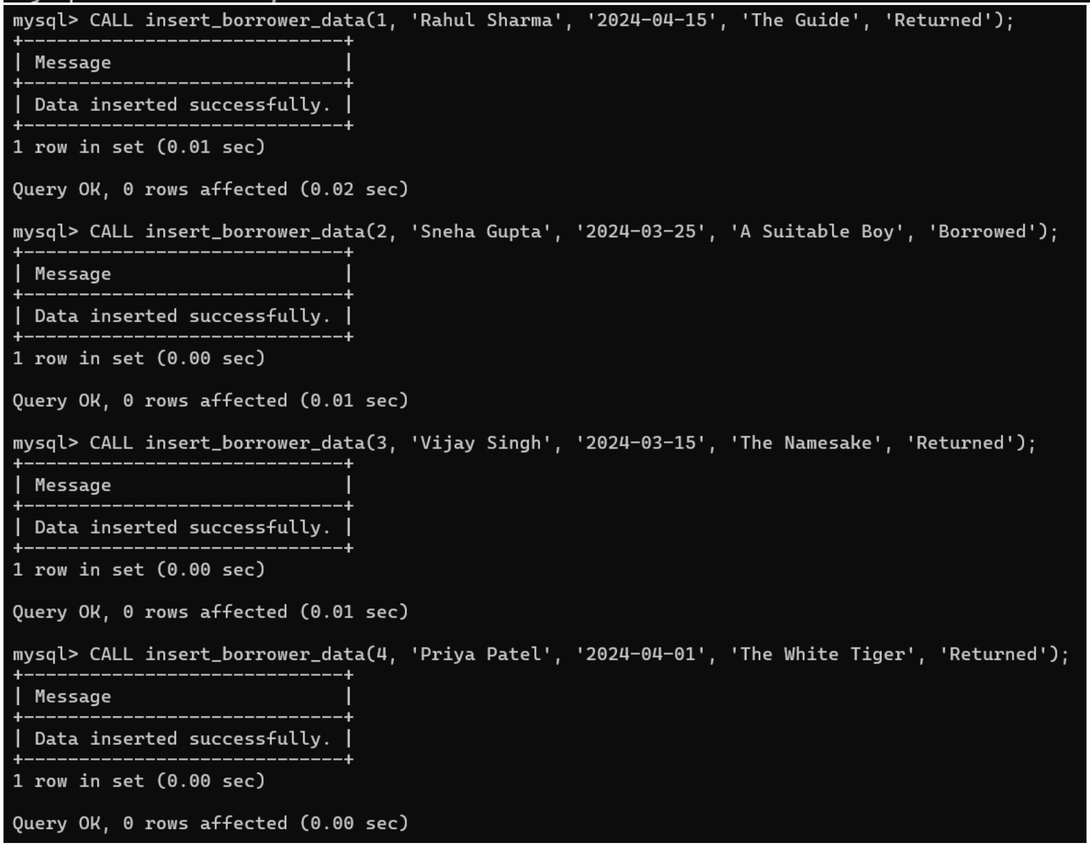


```
mysql> CREATE TABLE Fine (
    ->     Roll_no INT,
    ->     Date DATE,
    ->     Amt DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

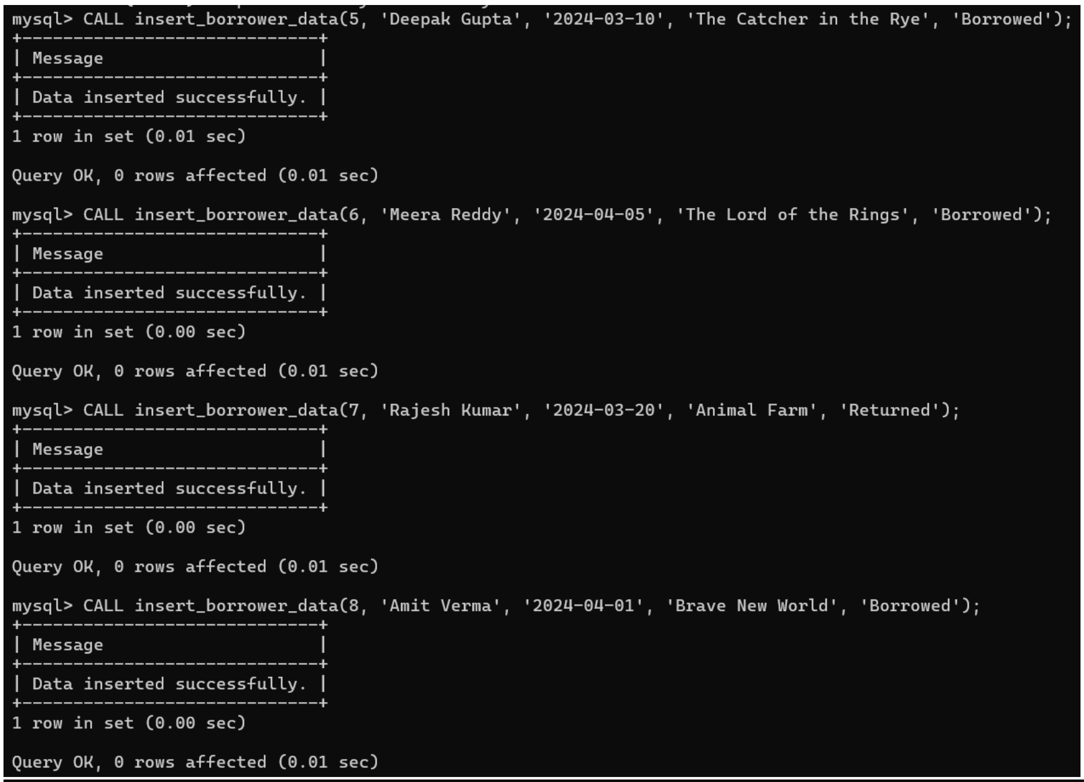
1. Write a PL/SQL block to accept input for Borrower table.



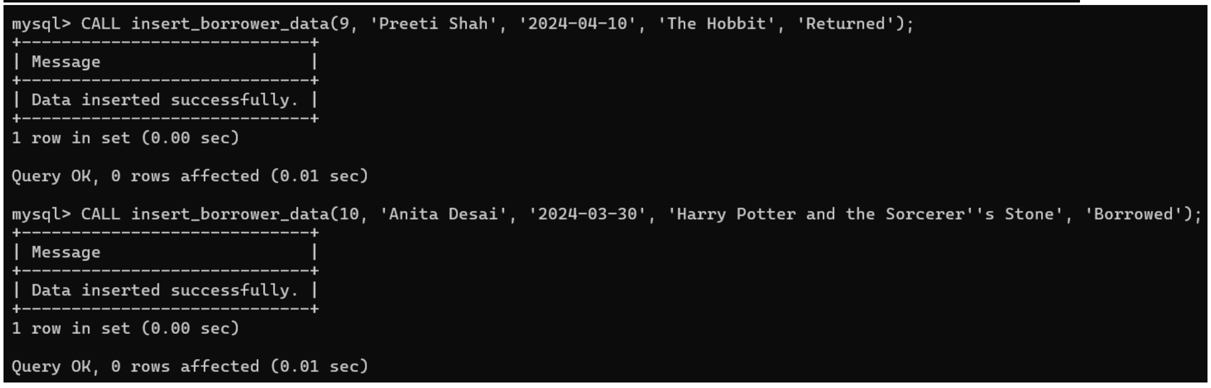
```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE insert_borrower_data(
    ->     IN p_Rollno INT,
    ->     IN p_Name VARCHAR(100),
    ->     IN p_DateofIssue DATE,
    ->     IN p_NameofBook VARCHAR(100),
    ->     IN p_Status VARCHAR(50)
    -> )
    -> BEGIN
    ->     -- Insert the input data into the Borrower2 table
    ->     INSERT INTO Borrower2 (Rollno, Name, DateofIssue, NameofBook, Status)
    ->     VALUES (p_Rollno, p_Name, p_DateofIssue, p_NameofBook, p_Status);
    ->
    ->     -- Display success message
    ->     SELECT 'Data inserted successfully.' AS Message;
    -> END //
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> DELIMITER ;
```



```
mysql> CALL insert_borrower_data(1, 'Rahul Sharma', '2024-04-15', 'The Guide', 'Returned');
+-----------------------------+
| Message                     |
+-----------------------------+
| Data inserted successfully. |
+-----------------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> CALL insert_borrower_data(2, 'Sneha Gupta', '2024-03-25', 'A Suitable Boy', 'Borrowed');
+-----------------------------+
| Message                     |
+-----------------------------+
| Data inserted successfully. |
+-----------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL insert_borrower_data(3, 'Vijay Singh', '2024-03-15', 'The Namesake', 'Returned');
+-----------------------------+
| Message                     |
+-----------------------------+
| Data inserted successfully. |
+-----------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL insert_borrower_data(4, 'Priya Patel', '2024-04-01', 'The White Tiger', 'Returned');
+-----------------------------+
| Message                     |
+-----------------------------+
| Data inserted successfully. |
+-----------------------------+
1 row in set (0.00 sec)

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



```
mysql> CALL insert_borrower_data(5, 'Deepak Gupta', '2024-03-10', 'The Catcher in the Rye', 'Borrowed');
+------------------------------+
| Message                      |
+------------------------------+
| Data inserted successfully. |
+------------------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL insert_borrower_data(6, 'Meera Reddy', '2024-04-05', 'The Lord of the Rings', 'Borrowed');
+------------------------------+
| Message                      |
+------------------------------+
| Data inserted successfully. |
+------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL insert_borrower_data(7, 'Rajesh Kumar', '2024-03-20', 'Animal Farm', 'Returned');
+------------------------------+
| Message                      |
+------------------------------+
| Data inserted successfully. |
+------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL insert_borrower_data(8, 'Amit Verma', '2024-04-01', 'Brave New World', 'Borrowed');
+------------------------------+
| Message                      |
+------------------------------+
| Data inserted successfully. |
+------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```



```
mysql> CALL insert_borrower_data(9, 'Preeti Shah', '2024-04-10', 'The Hobbit', 'Returned');
+------------------------------+
| Message                      |
+------------------------------+
| Data inserted successfully. |
+------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL insert_borrower_data(10, 'Anita Desai', '2024-03-30', 'Harry Potter and the Sorcerer''s Stone', 'Borrowed');
+------------------------------+
| Message                      |
+------------------------------+
| Data inserted successfully. |
+------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

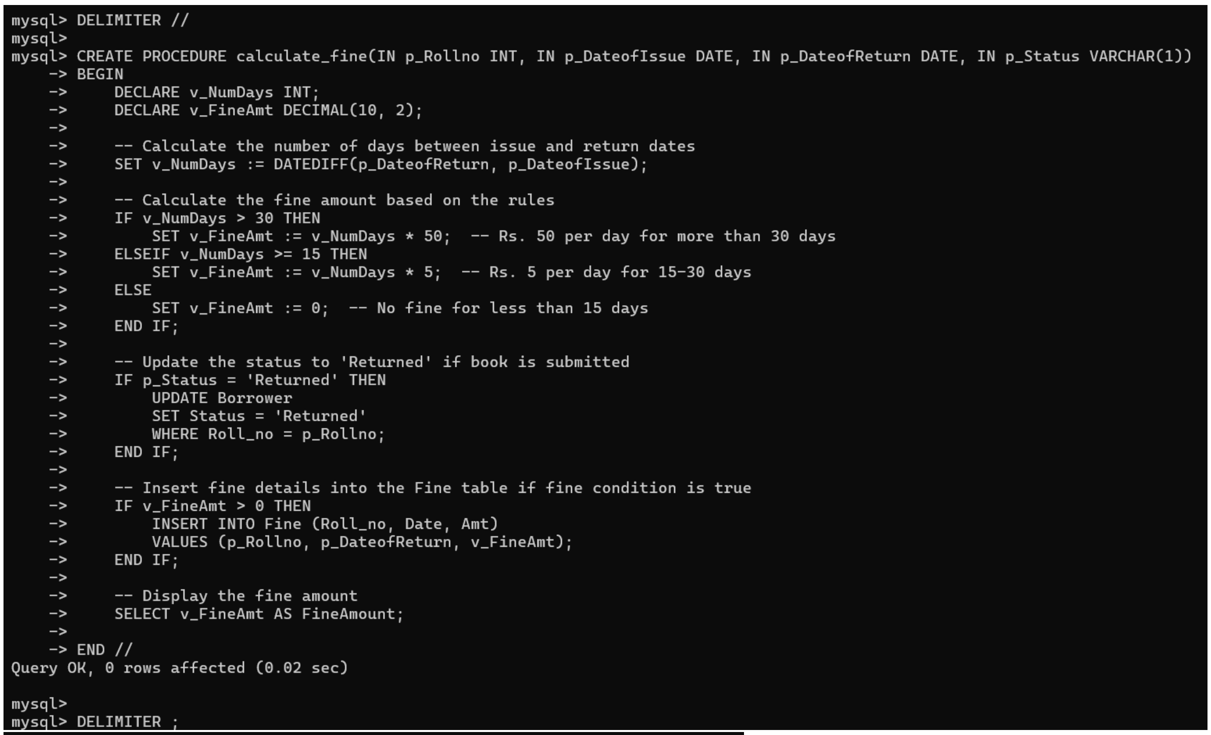
2. Write a PL/SQL block using control structures to calculate fine by using the following rules:

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

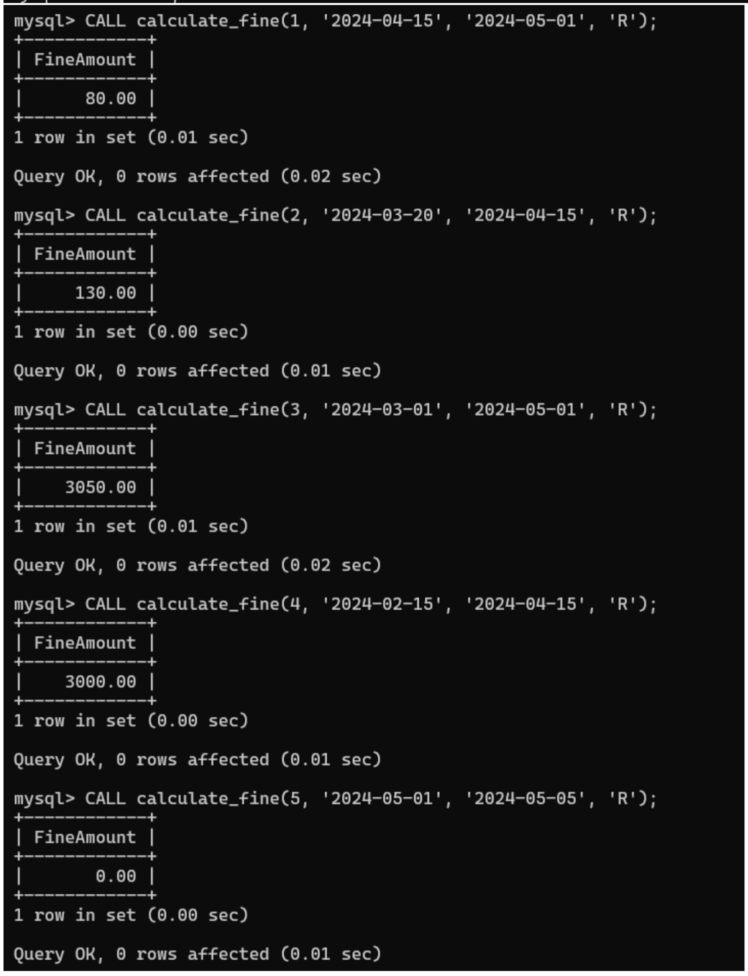
b. If no. of days>30, per day fine will be Rs 50 per day

c. for days less than 30, Rs. 5 per day.

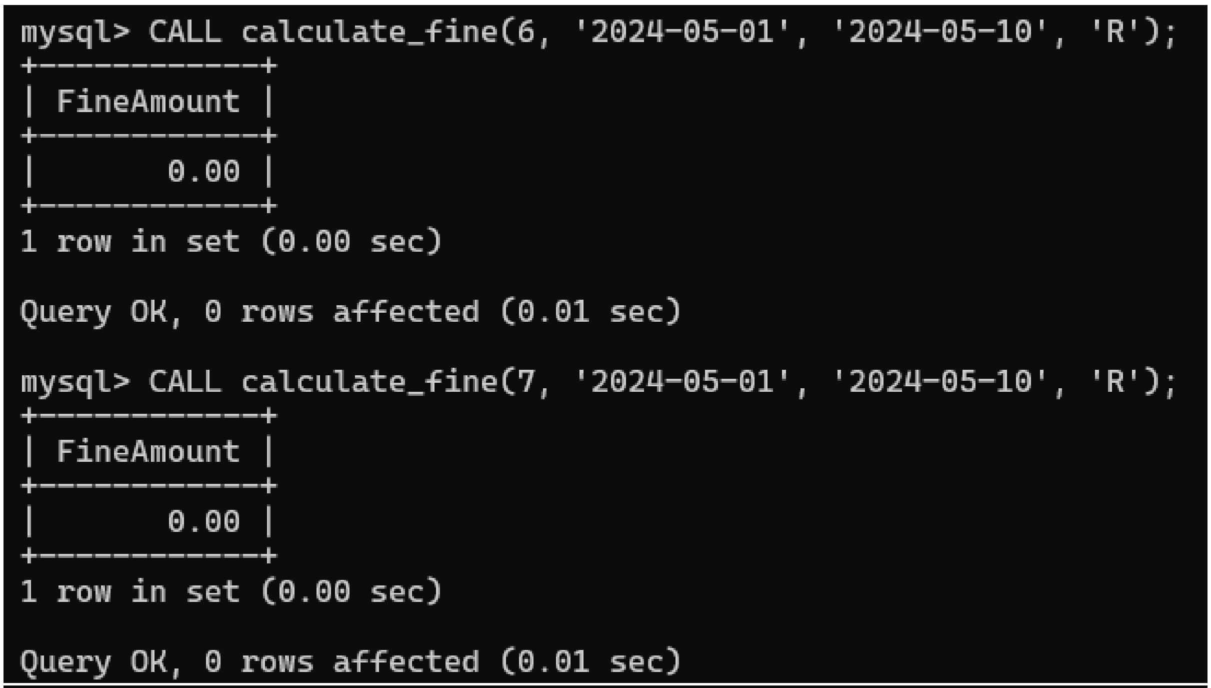
After submitting the book, status will change from I to R. If condition of fine is true, then details will be stored into fine table.



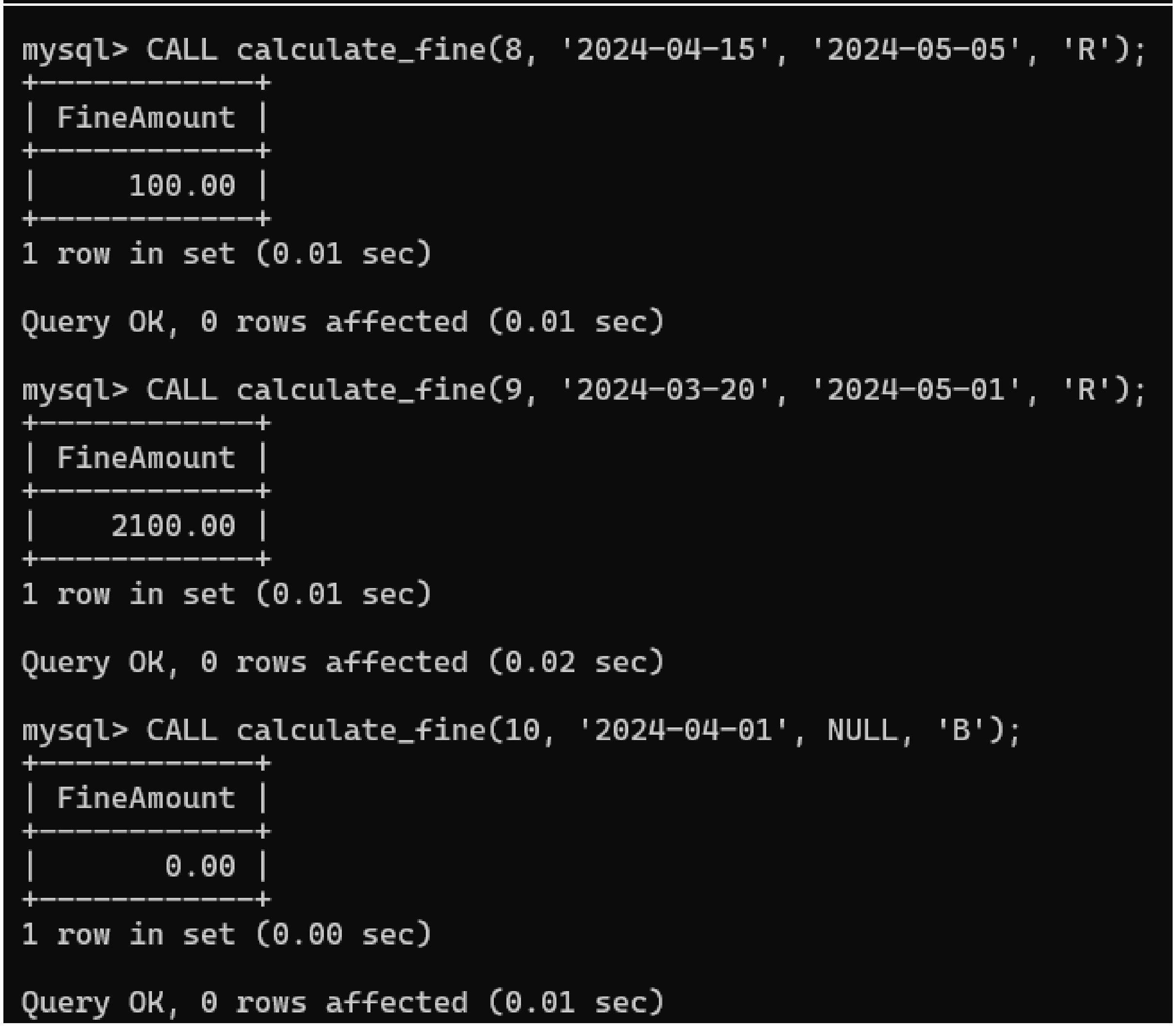
```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE calculate_fine(IN p_Rollno INT, IN p_DateofIssue DATE, IN p_DateofReturn DATE, IN p_Status VARCHAR(1))
    -> BEGIN
    ->     DECLARE v_NumDays INT;
    ->     DECLARE v_FineAmt DECIMAL(10, 2);
    ->
    ->     -- Calculate the number of days between issue and return dates
    ->     SET v_NumDays := DATEDIFF(p_DateofReturn, p_DateofIssue);
    ->
    ->     -- Calculate the fine amount based on the rules
    ->     IF v_NumDays > 30 THEN
    ->         SET v_FineAmt := v_NumDays * 50;  -- Rs. 50 per day for more than 30 days
    ->     ELSEIF v_NumDays >= 15 THEN
    ->         SET v_FineAmt := v_NumDays * 5;  -- Rs. 5 per day for 15-30 days
    ->     ELSE
    ->         SET v_FineAmt := 0;  -- No fine for less than 15 days
    ->     END IF;
    ->
    ->     -- Update the status to 'Returned' if book is submitted
    ->     IF p_Status = 'Returned' THEN
    ->         UPDATE Borrower
    ->         SET Status = 'Returned'
    ->         WHERE Roll_no = p_Rollno;
    ->     END IF;
    ->
    ->     -- Insert fine details into the Fine table if fine condition is true
    ->     IF v_FineAmt > 0 THEN
    ->         INSERT INTO Fine (Roll_no, Date, Amt)
    ->         VALUES (p_Rollno, p_DateofReturn, v_FineAmt);
    ->     END IF;
    ->
    ->     -- Display the fine amount
    ->     SELECT v_FineAmt AS FineAmount;
    ->
    -> END //
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
```



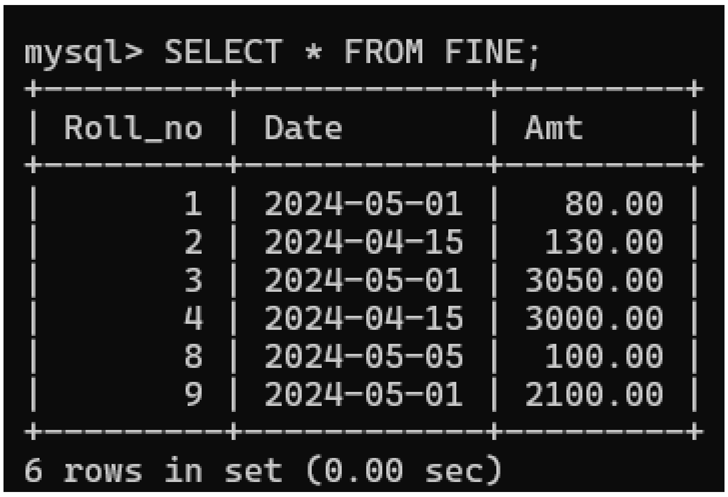
```
mysql> CALL calculate_fine(1, '2024-04-15', '2024-05-01', 'R');
+------------+
| FineAmount |
+------------+
|      80.00 |
+------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> CALL calculate_fine(2, '2024-03-20', '2024-04-15', 'R');
+------------+
| FineAmount |
+------------+
|     130.00 |
+------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL calculate_fine(3, '2024-03-01', '2024-05-01', 'R');
+------------+
| FineAmount |
+------------+
|    3050.00 |
+------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> CALL calculate_fine(4, '2024-02-15', '2024-04-15', 'R');
+------------+
| FineAmount |
+------------+
|    3000.00 |
+------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL calculate_fine(5, '2024-05-01', '2024-05-05', 'R');
+------------+
| FineAmount |
+------------+
|       0.00 |
+------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```



```
mysql> CALL calculate_fine(6, '2024-05-01', '2024-05-10', 'R');
+------------+
| FineAmount |
+------------+
|       0.00 |
+------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL calculate_fine(7, '2024-05-01', '2024-05-10', 'R');
+------------+
| FineAmount |
+------------+
|       0.00 |
+------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```



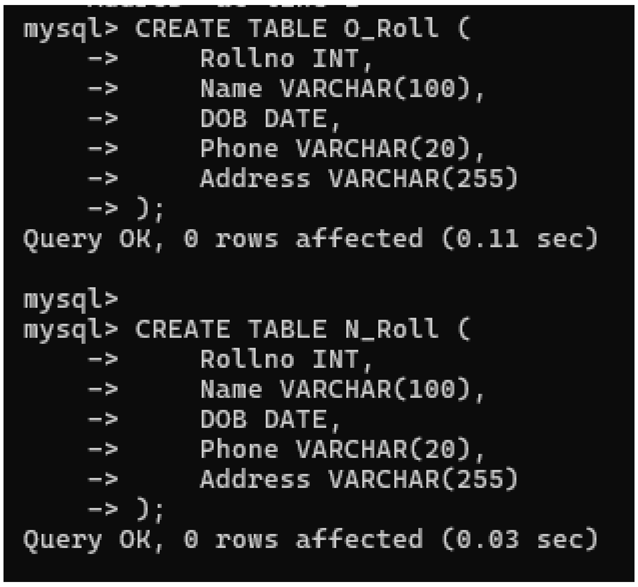
```
mysql> CALL calculate_fine(8, '2024-04-15', '2024-05-05', 'R');
+------------+
| FineAmount |
+------------+
|     100.00 |
+------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL calculate_fine(9, '2024-03-20', '2024-05-01', 'R');
+------------+
| FineAmount |
+------------+
|    2100.00 |
+------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> CALL calculate_fine(10, '2024-04-01', NULL, 'B');
+------------+
| FineAmount |
+------------+
|       0.00 |
+------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```



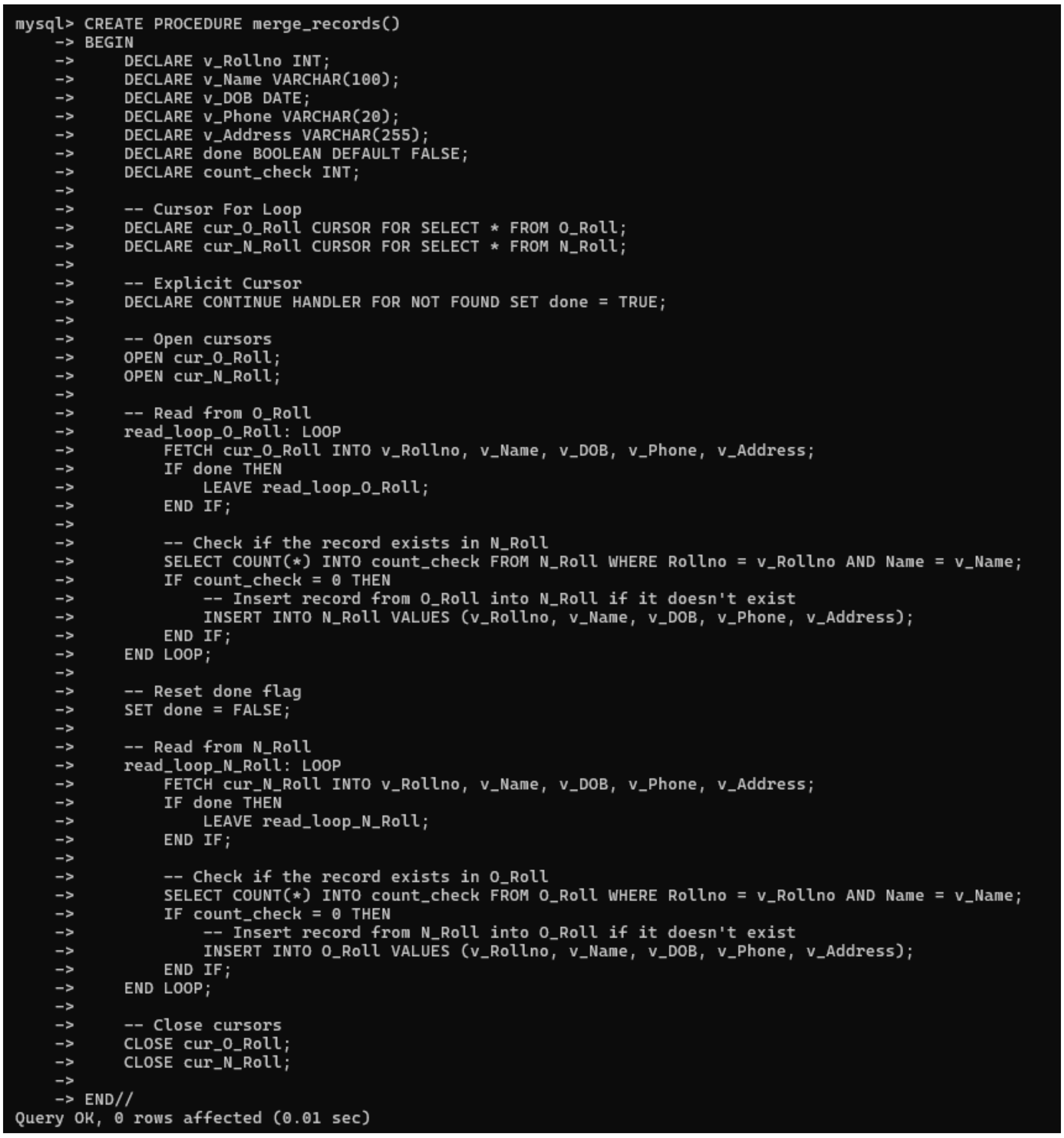
```
mysql> SELECT * FROM FINE;
+---------+------------+---------+
| Roll_no | Date       | Amt     |
+---------+------------+---------+
|       1 | 2024-05-01 |   80.00 |
|       2 | 2024-04-15 |  130.00 |
|       3 | 2024-05-01 | 3050.00 |
|       4 | 2024-04-15 | 3000.00 |
|       8 | 2024-05-05 |  100.00 |
|       9 | 2024-05-01 | 2100.00 |
+---------+------------+---------+
6 rows in set (0.00 sec)
```

Question 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> CREATE TABLE O_Roll (
    ->     Rollno INT,
    ->     Name VARCHAR(100),
    ->     DOB DATE,
    ->     Phone VARCHAR(20),
    ->     Address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.11 sec)

mysql>
mysql> CREATE TABLE N_Roll (
    ->     Rollno INT,
    ->     Name VARCHAR(100),
    ->     DOB DATE,
    ->     Phone VARCHAR(20),
    ->     Address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

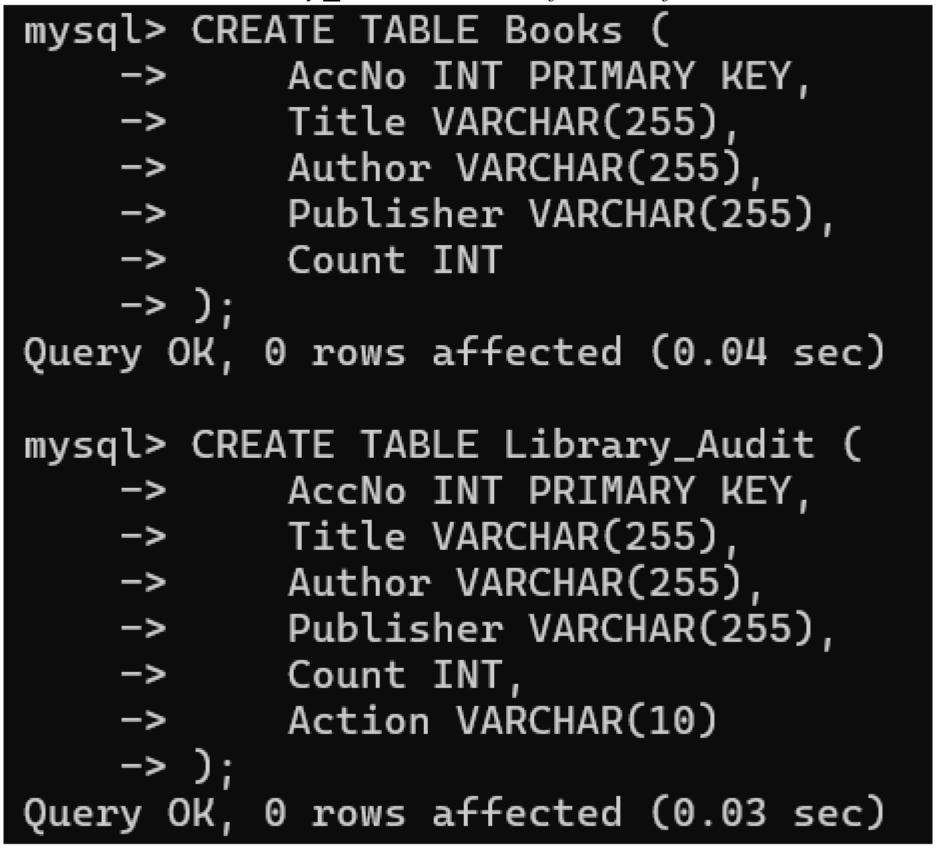


```
mysql> CREATE PROCEDURE merge_records()
    -> BEGIN
    ->     DECLARE v_Rollno INT;
    ->     DECLARE v_Name VARCHAR(100);
    ->     DECLARE v_DOB DATE;
    ->     DECLARE v_Phone VARCHAR(20);
    ->     DECLARE v_Address VARCHAR(255);
    ->     DECLARE done BOOLEAN DEFAULT FALSE;
    ->     DECLARE count_check INT;
    ->
    ->     -- Cursor For Loop
    ->     DECLARE cur_O_Roll CURSOR FOR SELECT * FROM O_Roll;
    ->     DECLARE cur_N_Roll CURSOR FOR SELECT * FROM N_Roll;
    ->
    ->     -- Explicit Cursor
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    ->
    ->     -- Open cursors
    ->     OPEN cur_O_Roll;
    ->     OPEN cur_N_Roll;
    ->
    ->     -- Read from O_Roll
    ->     read_loop_O_Roll: LOOP
    ->         FETCH cur_O_Roll INTO v_Rollno, v_Name, v_DOB, v_Phone, v_Address;
    ->         IF done THEN
    ->             LEAVE read_loop_O_Roll;
    ->         END IF;
    ->
    ->         -- Check if the record exists in N_Roll
    ->         SELECT COUNT(*) INTO count_check FROM N_Roll WHERE Rollno = v_Rollno AND Name = v_Name;
    ->         IF count_check = 0 THEN
    ->             -- Insert record from O_Roll into N_Roll if it doesn't exist
    ->             INSERT INTO N_Roll VALUES (v_Rollno, v_Name, v_DOB, v_Phone, v_Address);
    ->         END IF;
    ->     END LOOP;
    ->
    ->     -- Reset done flag
    ->     SET done = FALSE;
    ->
    ->     -- Read from N_Roll
    ->     read_loop_N_Roll: LOOP
    ->         FETCH cur_N_Roll INTO v_Rollno, v_Name, v_DOB, v_Phone, v_Address;
    ->         IF done THEN
    ->             LEAVE read_loop_N_Roll;
    ->         END IF;
    ->
    ->         -- Check if the record exists in O_Roll
    ->         SELECT COUNT(*) INTO count_check FROM O_Roll WHERE Rollno = v_Rollno AND Name = v_Name;
    ->         IF count_check = 0 THEN
    ->             -- Insert record from N_Roll into O_Roll if it doesn't exist
    ->             INSERT INTO O_Roll VALUES (v_Rollno, v_Name, v_DOB, v_Phone, v_Address);
    ->         END IF;
    ->     END LOOP;
    ->
    ->     -- Close cursors
    ->     CLOSE cur_O_Roll;
    ->     CLOSE cur_N_Roll;
    ->
    -> END//
Query OK, 0 rows affected (0.01 sec)
```

Question 6:

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

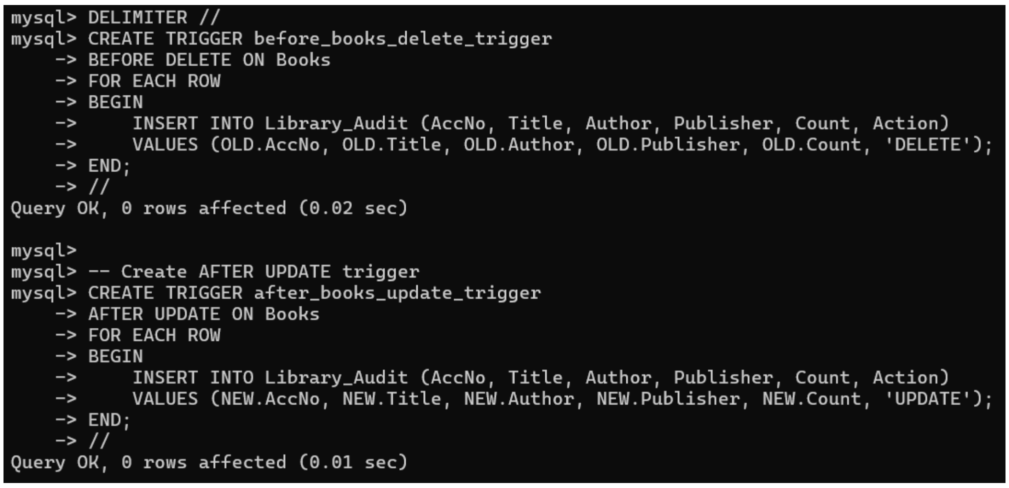
a. Create a table Library_Audit with same fiels as of Books.



```
mysql> CREATE TABLE Books (
    ->     AccNo INT PRIMARY KEY,
    ->     Title VARCHAR(255),
    ->     Author VARCHAR(255),
    ->     Publisher VARCHAR(255),
    ->     Count INT
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Library_Audit (
    ->     AccNo INT PRIMARY KEY,
    ->     Title VARCHAR(255),
    ->     Author VARCHAR(255),
    ->     Publisher VARCHAR(255),
    ->     Count INT,
    ->     Action VARCHAR(10)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

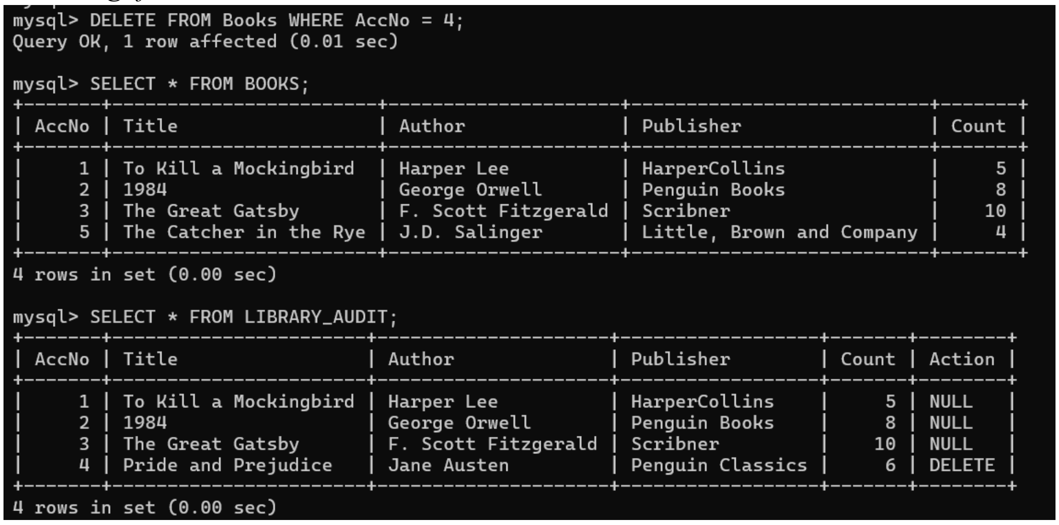
b. Create a before trigger to insert records into Librry_Audit table if there is deletion in Books table.

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



```
mysql> DELIMITER //
mysql> CREATE TRIGGER before_books_delete_trigger
    -> BEFORE DELETE ON Books
    -> FOR EACH ROW
    -> BEGIN
    ->     INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count, Action)
    ->     VALUES (OLD.AccNo, OLD.Title, OLD.Author, OLD.Publisher, OLD.Count, 'DELETE');
    -> END;
    -> //
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> -- Create AFTER UPDATE trigger
mysql> CREATE TRIGGER after_books_update_trigger
    -> AFTER UPDATE ON Books
    -> FOR EACH ROW
    -> BEGIN
    ->     INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count, Action)
    ->     VALUES (NEW.AccNo, NEW.Title, NEW.Author, NEW.Publisher, NEW.Count, 'UPDATE');
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)
```

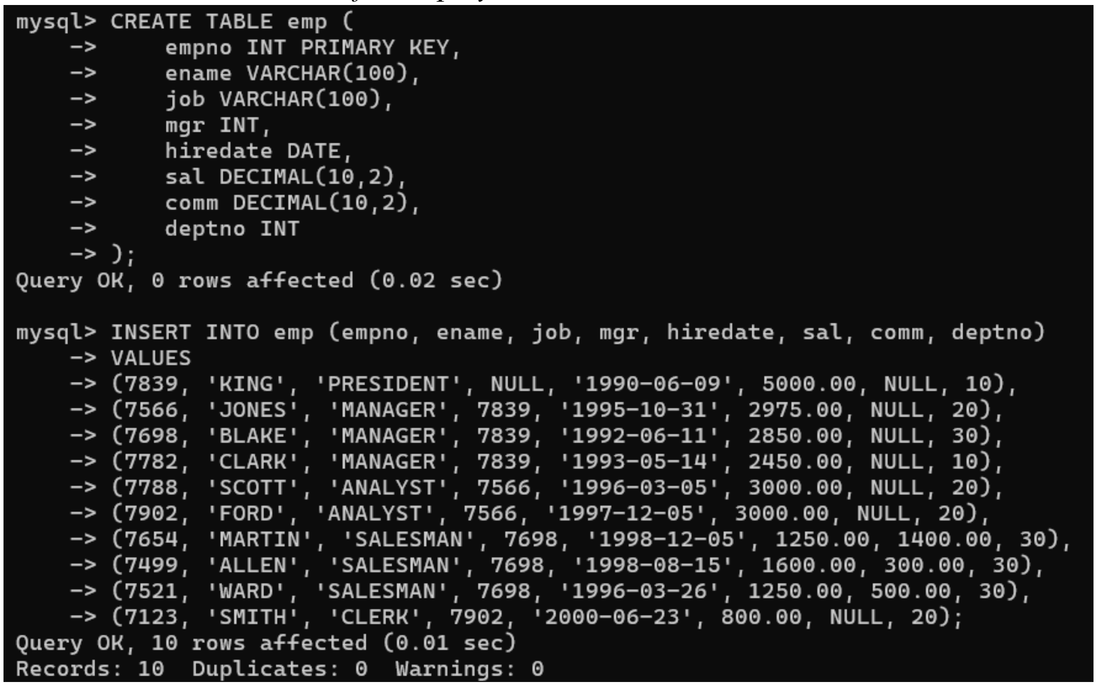
Checking if it worked:



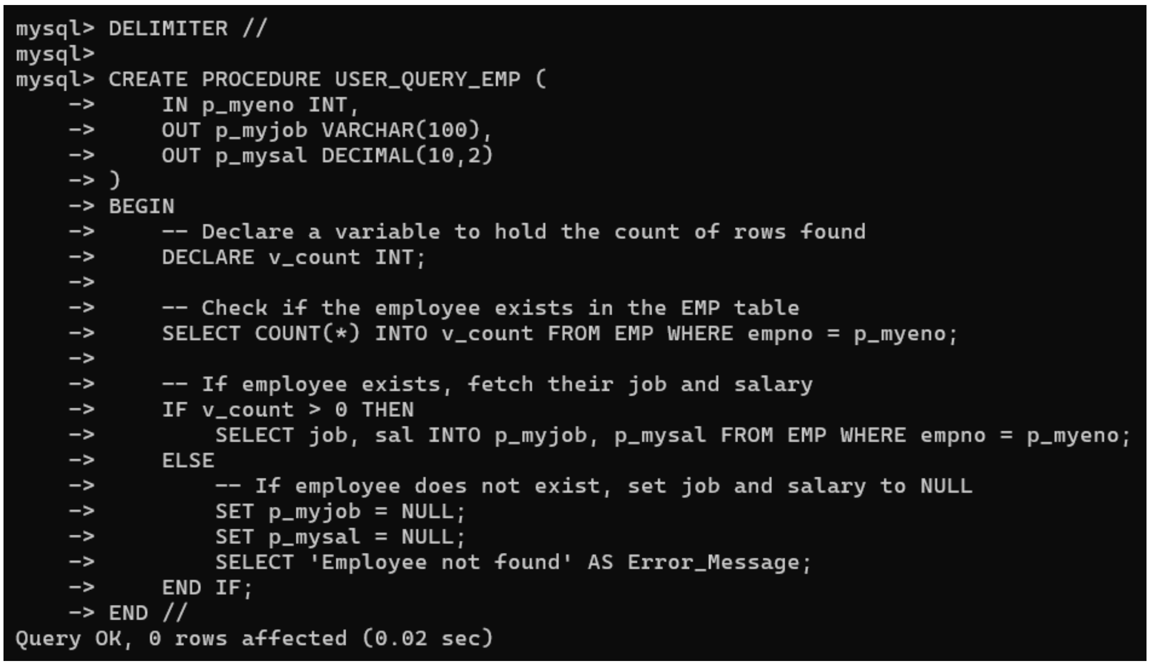
```
mysql> DELETE FROM Books WHERE AccNo = 4;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM BOOKS;
+-------+-------------------------+---------------------+---------------------------+-------+
| AccNo | Title                   | Author              | Publisher                 | Count |
+-------+-------------------------+---------------------+---------------------------+-------+
|     1 | To Kill a Mockingbird   | Harper Lee          | HarperCollins             |     5 |
|     2 | 1984                    | George Orwell       | Penguin Books             |     8 |
|     3 | The Great Gatsby        | F. Scott Fitzgerald | Scribner                  |    10 |
|     5 | The Catcher in the Rye  | J.D. Salinger       | Little, Brown and Company |     4 |
+-------+-------------------------+---------------------+---------------------------+-------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM LIBRARY_AUDIT;
+-------+-----------------------+---------------------+------------------+-------+--------+
| AccNo | Title                 | Author              | Publisher        | Count | Action |
+-------+-----------------------+---------------------+------------------+-------+--------+
|     1 | To Kill a Mockingbird | Harper Lee          | HarperCollins    |     5 | NULL   |
|     2 | 1984                  | George Orwell       | Penguin Books    |     8 | NULL   |
|     3 | The Great Gatsby      | F. Scott Fitzgerald | Scribner         |    10 | NULL   |
|     4 | Pride and Prejudice   | Jane Austen         | Penguin Classics |     6 | DELETE |
+-------+-----------------------+---------------------+------------------+-------+--------+
4 rows in set (0.00 sec)
```

Question 7:

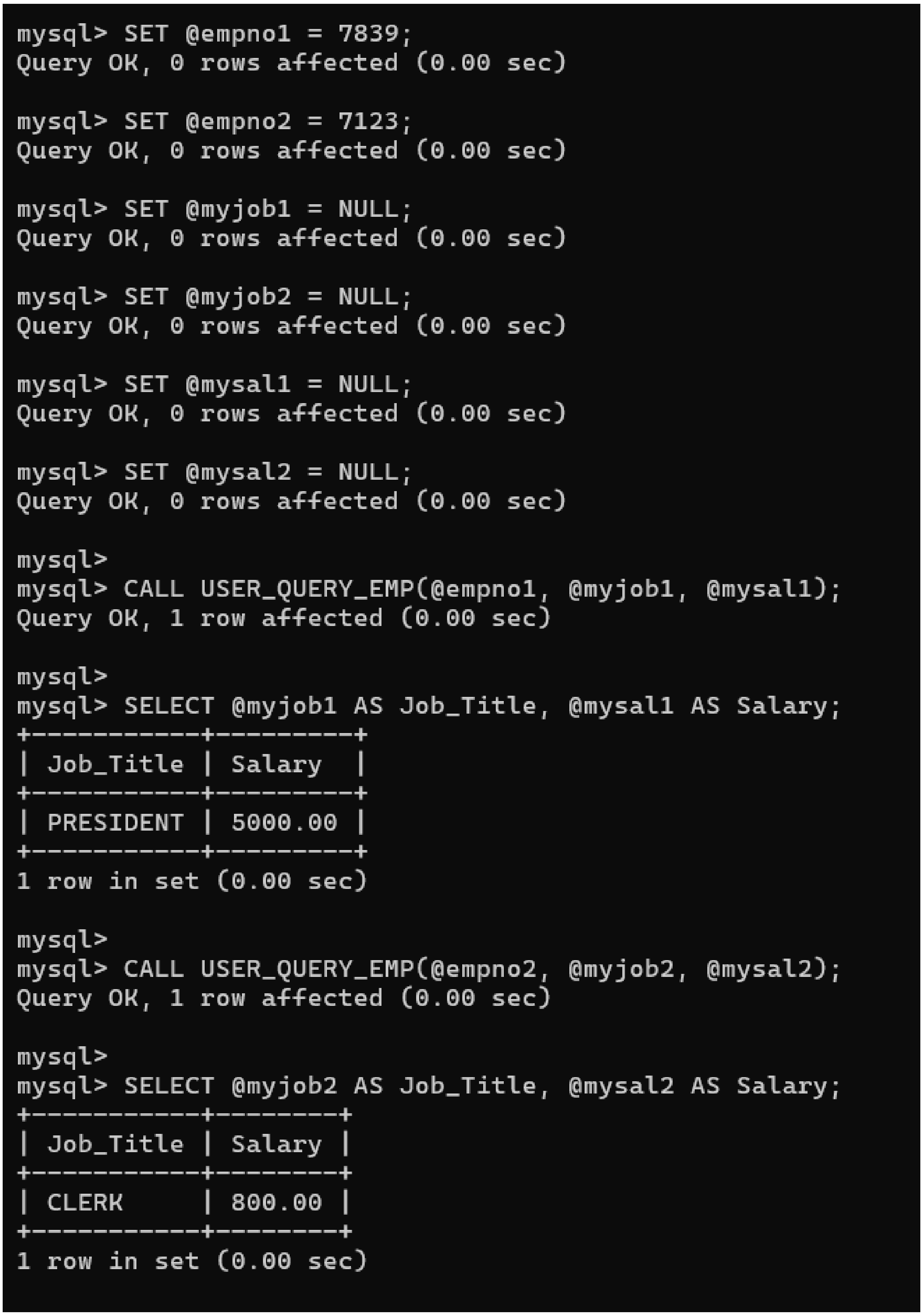
Create a procedure called USER_QUERY_EMP that accepts three parameters. Parameter p_myeno is of IN parameter mode which provides the empno value. The other two parameters p_myjob and p_mysal are of OUT mode. The procedure retrieves the salary and job of an employee with the provided employee number and 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> CREATE TABLE emp (
    ->     empno INT PRIMARY KEY,
    ->     ename VARCHAR(100),
    ->     job VARCHAR(100),
    ->     mgr INT,
    ->     hiredate DATE,
    ->     sal DECIMAL(10,2),
    ->     comm DECIMAL(10,2),
    ->     deptno INT
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
    -> VALUES
    -> (7839, 'KING', 'PRESIDENT', NULL, '1990-06-09', 5000.00, NULL, 10),
    -> (7566, 'JONES', 'MANAGER', 7839, '1995-10-31', 2975.00, NULL, 20),
    -> (7698, 'BLAKE', 'MANAGER', 7839, '1992-06-11', 2850.00, NULL, 30),
    -> (7782, 'CLARK', 'MANAGER', 7839, '1993-05-14', 2450.00, NULL, 10),
    -> (7788, 'SCOTT', 'ANALYST', 7566, '1996-03-05', 3000.00, NULL, 20),
    -> (7902, 'FORD', 'ANALYST', 7566, '1997-12-05', 3000.00, NULL, 20),
    -> (7654, 'MARTIN', 'SALESMAN', 7698, '1998-12-05', 1250.00, 1400.00, 30),
    -> (7499, 'ALLEN', 'SALESMAN', 7698, '1998-08-15', 1600.00, 300.00, 30),
    -> (7521, 'WARD', 'SALESMAN', 7698, '1996-03-26', 1250.00, 500.00, 30),
    -> (7123, 'SMITH', 'CLERK', 7902, '2000-06-23', 800.00, NULL, 20);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```



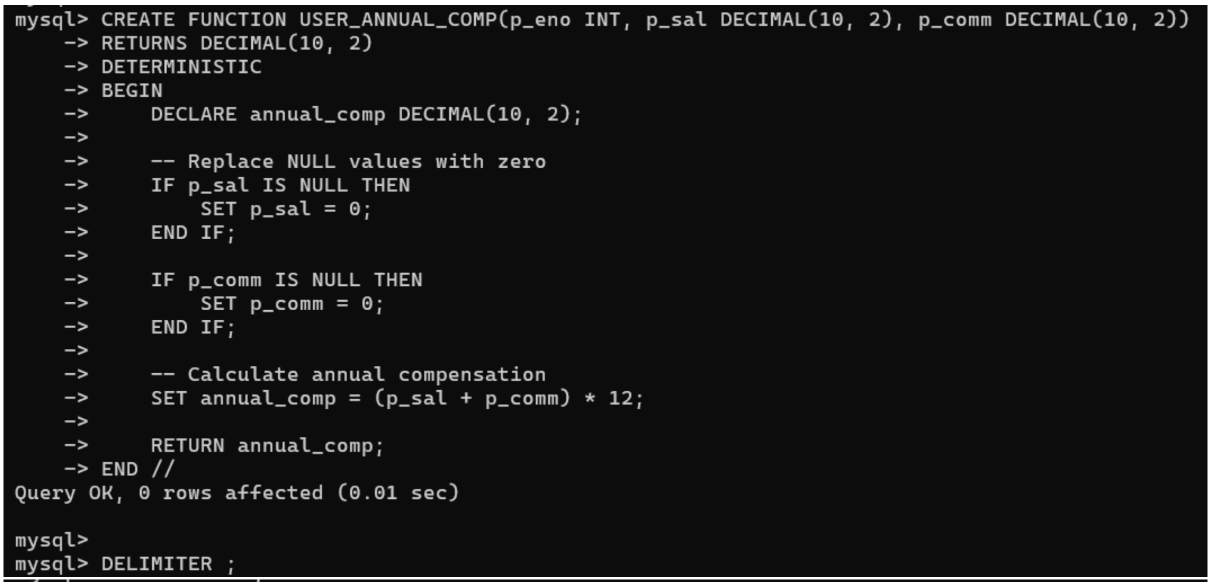
```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE USER_QUERY_EMP (
    ->     IN p_myeno INT,
    ->     OUT p_myjob VARCHAR(100),
    ->     OUT p_mysal DECIMAL(10,2)
    -> )
    -> BEGIN
    ->     -- Declare a variable to hold the count of rows found
    ->     DECLARE v_count INT;
    ->
    ->     -- Check if the employee exists in the EMP table
    ->     SELECT COUNT(*) INTO v_count FROM EMP WHERE empno = p_myeno;
    ->
    ->     -- If employee exists, fetch their job and salary
    ->     IF v_count > 0 THEN
    ->         SELECT job, sal INTO p_myjob, p_mysal FROM EMP WHERE empno = p_myeno;
    ->     ELSE
    ->         -- If employee does not exist, set job and salary to NULL
    ->         SET p_myjob = NULL;
    ->         SET p_mysal = NULL;
    ->         SELECT 'Employee not found' AS Error_Message;
    ->     END IF;
    -> END //
Query OK, 0 rows affected (0.02 sec)
```



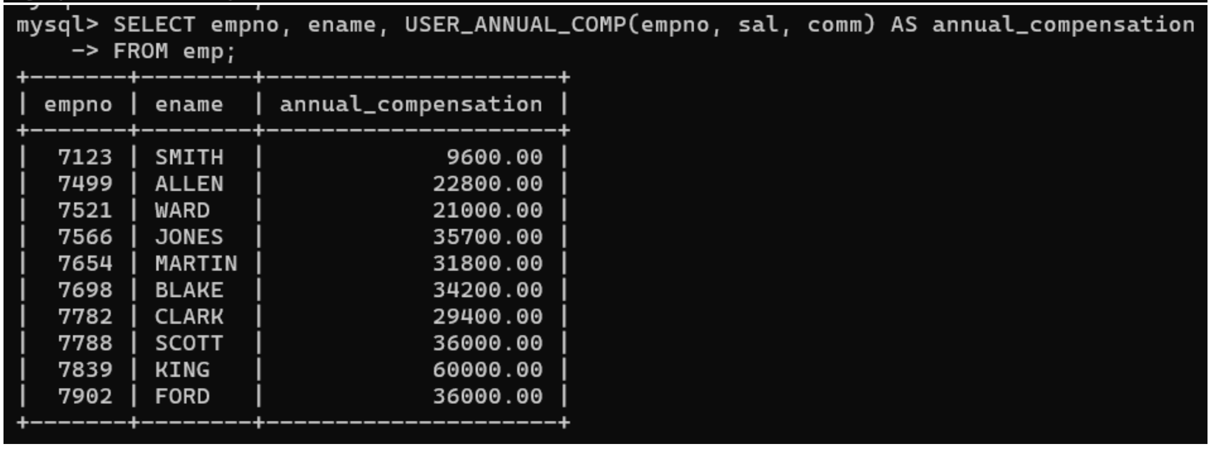
```
mysql> SET @empno1 = 7839;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @empno2 = 7123;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @myjob1 = NULL;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @myjob2 = NULL;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @mysal1 = NULL;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @mysal2 = NULL;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> CALL USER_QUERY_EMP(@empno1, @myjob1, @mysal1);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> SELECT @myjob1 AS Job_Title, @mysal1 AS Salary;
+-----------+---------+
| Job_Title | Salary  |
+-----------+---------+
| PRESIDENT | 5000.00 |
+-----------+---------+
1 row in set (0.00 sec)

mysql>
mysql> CALL USER_QUERY_EMP(@empno2, @myjob2, @mysal2);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> SELECT @myjob2 AS Job_Title, @mysal2 AS Salary;
+-----------+--------+
| Job_Title | Salary |
+-----------+--------+
| CLERK     | 800.00 |
+-----------+--------+
1 row in set (0.00 sec)
```

Question 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.



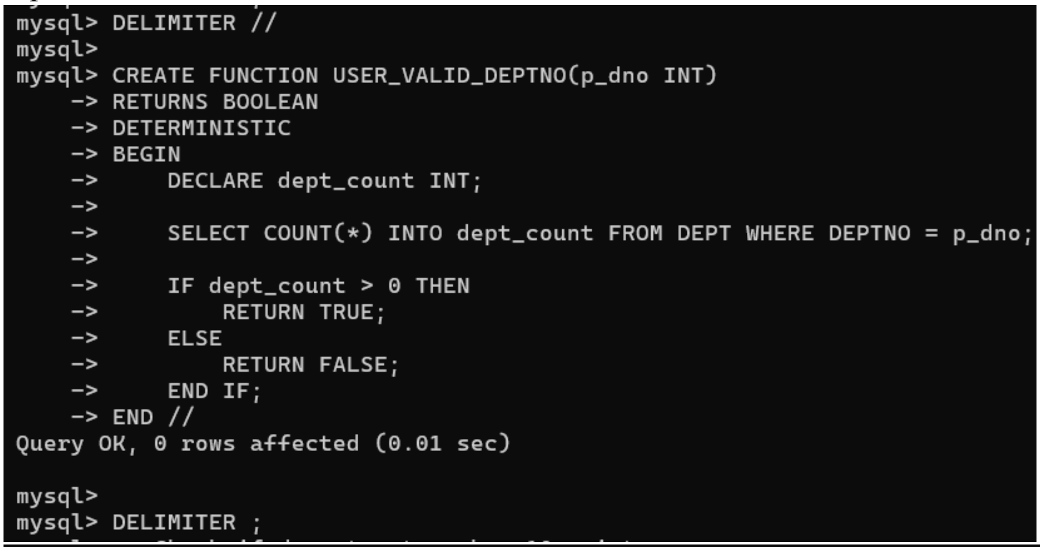
```
mysql> CREATE FUNCTION USER_ANNUAL_COMP(p_eno INT, p_sal DECIMAL(10, 2), p_comm DECIMAL(10, 2))
    -> RETURNS DECIMAL(10, 2)
    -> DETERMINISTIC
    -> BEGIN
    ->     DECLARE annual_comp DECIMAL(10, 2);
    ->
    ->     -- Replace NULL values with zero
    ->     IF p_sal IS NULL THEN
    ->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

mysql>
mysql> DELIMITER ;
```



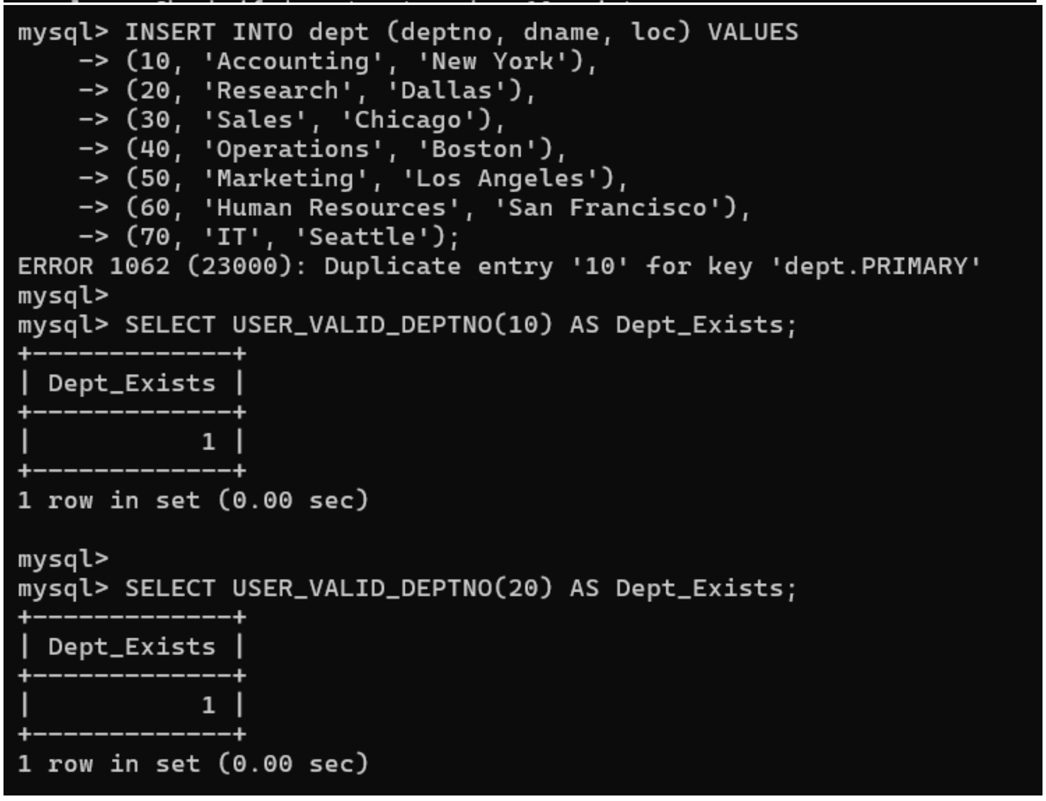
```
mysql> SELECT empno, ename, USER_ANNUAL_COMP(empno, sal, comm) AS annual_compensation
    -> FROM emp;
+-------+--------+---------------------+
| empno | ename  | annual_compensation |
+-------+--------+---------------------+
|  7123 | SMITH  |             9600.00 |
|  7499 | ALLEN  |            22800.00 |
|  7521 | WARD   |            21000.00 |
|  7566 | JONES  |            35700.00 |
|  7654 | MARTIN |            31800.00 |
|  7698 | BLAKE  |            34200.00 |
|  7782 | CLARK  |            29400.00 |
|  7788 | SCOTT  |            36000.00 |
|  7839 | KING   |            60000.00 |
|  7902 | FORD   |            36000.00 |
+-------+--------+---------------------+
```

Question 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



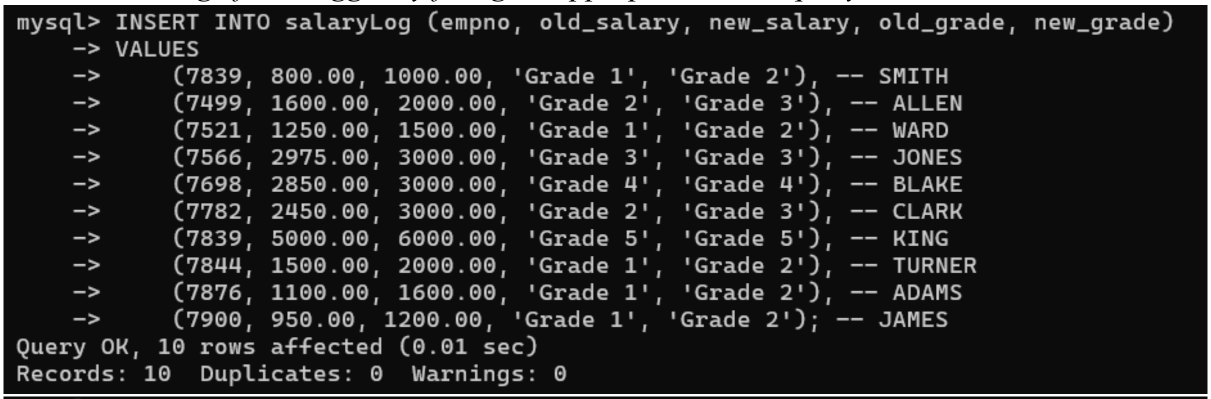
```
mysql> DELIMITER //
mysql>
mysql> CREATE FUNCTION USER_VALID_DEPTNO(p_dno INT)
    -> RETURNS BOOLEAN
    -> DETERMINISTIC
    -> BEGIN
    ->     DECLARE dept_count INT;
    ->
    ->     SELECT COUNT(*) INTO dept_count FROM DEPT WHERE DEPTNO = p_dno;
    ->
    ->     IF dept_count > 0 THEN
    ->         RETURN TRUE;
    ->     ELSE
    ->         RETURN FALSE;
    ->     END IF;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
```



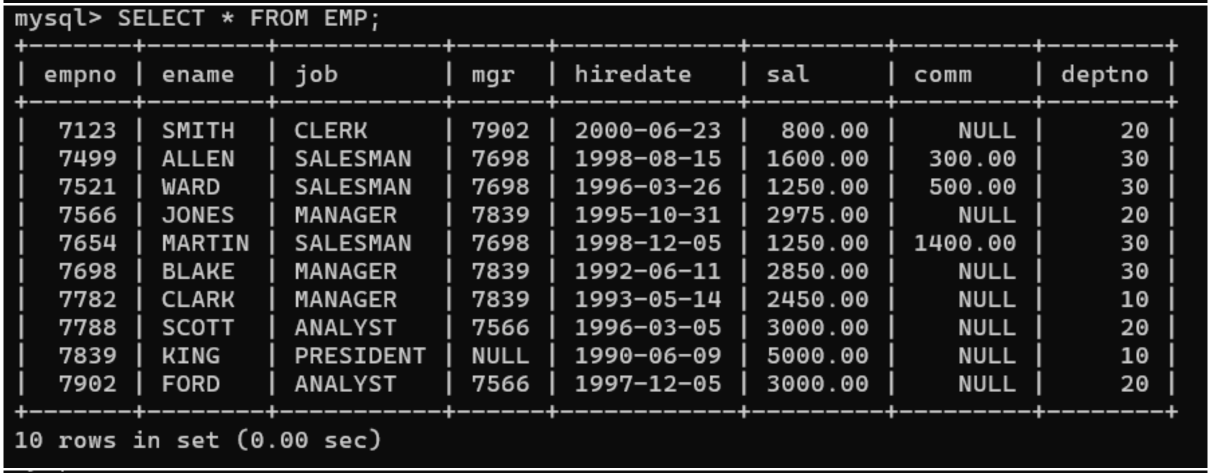
```
mysql> INSERT INTO dept (deptno, dname, loc) VALUES
    -> (10, 'Accounting', 'New York'),
    -> (20, 'Research', 'Dallas'),
    -> (30, 'Sales', 'Chicago'),
    -> (40, 'Operations', 'Boston'),
    -> (50, 'Marketing', 'Los Angeles'),
    -> (60, 'Human Resources', 'San Francisco'),
    -> (70, 'IT', 'Seattle');
ERROR 1062 (23000): Duplicate entry '10' for key 'dept.PRIMARY'
mysql>
mysql> SELECT USER_VALID_DEPTNO(10) AS Dept_Exists;
+-------------+
| Dept_Exists |
+-------------+
|           1 |
+-------------+
1 row in set (0.00 sec)

mysql>
mysql> SELECT USER_VALID_DEPTNO(20) AS Dept_Exists;
+-------------+
| Dept_Exists |
+-------------+
|           1 |
+-------------+
1 row in set (0.00 sec)
```

Question 10:

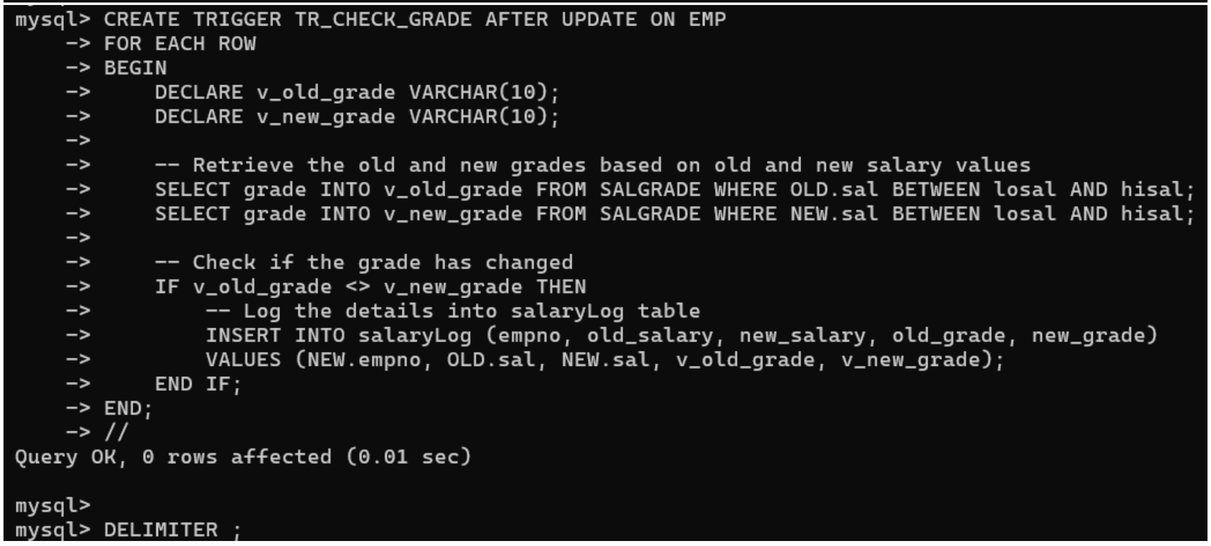
Create a table named salaryLog with the appropriate columns and insert the empno, new grade, old salary and new salary values in salaryLog table if the grade of an employee changes. Create a trigger named TR_CHECK_GRADE that will be fired when a user modifies
the EMP table. It will check whether the 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> INSERT INTO salaryLog (empno, old_salary, new_salary, old_grade, new_grade)
    -> VALUES
    ->     (7839, 800.00, 1000.00, 'Grade 1', 'Grade 2'), -- SMITH
    ->     (7499, 1600.00, 2000.00, 'Grade 2', 'Grade 3'), -- ALLEN
    ->     (7521, 1250.00, 1500.00, 'Grade 1', 'Grade 2'), -- WARD
    ->     (7566, 2975.00, 3000.00, 'Grade 3', 'Grade 3'), -- JONES
    ->     (7698, 2850.00, 3000.00, 'Grade 4', 'Grade 4'), -- BLAKE
    ->     (7782, 2450.00, 3000.00, 'Grade 2', 'Grade 3'), -- CLARK
    ->     (7839, 5000.00, 6000.00, 'Grade 5', 'Grade 5'), -- KING
    ->     (7844, 1500.00, 2000.00, 'Grade 1', 'Grade 2'), -- TURNER
    ->     (7876, 1100.00, 1600.00, 'Grade 1', 'Grade 2'), -- ADAMS
    ->     (7900, 950.00, 1200.00, 'Grade 1', 'Grade 2'); -- JAMES
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

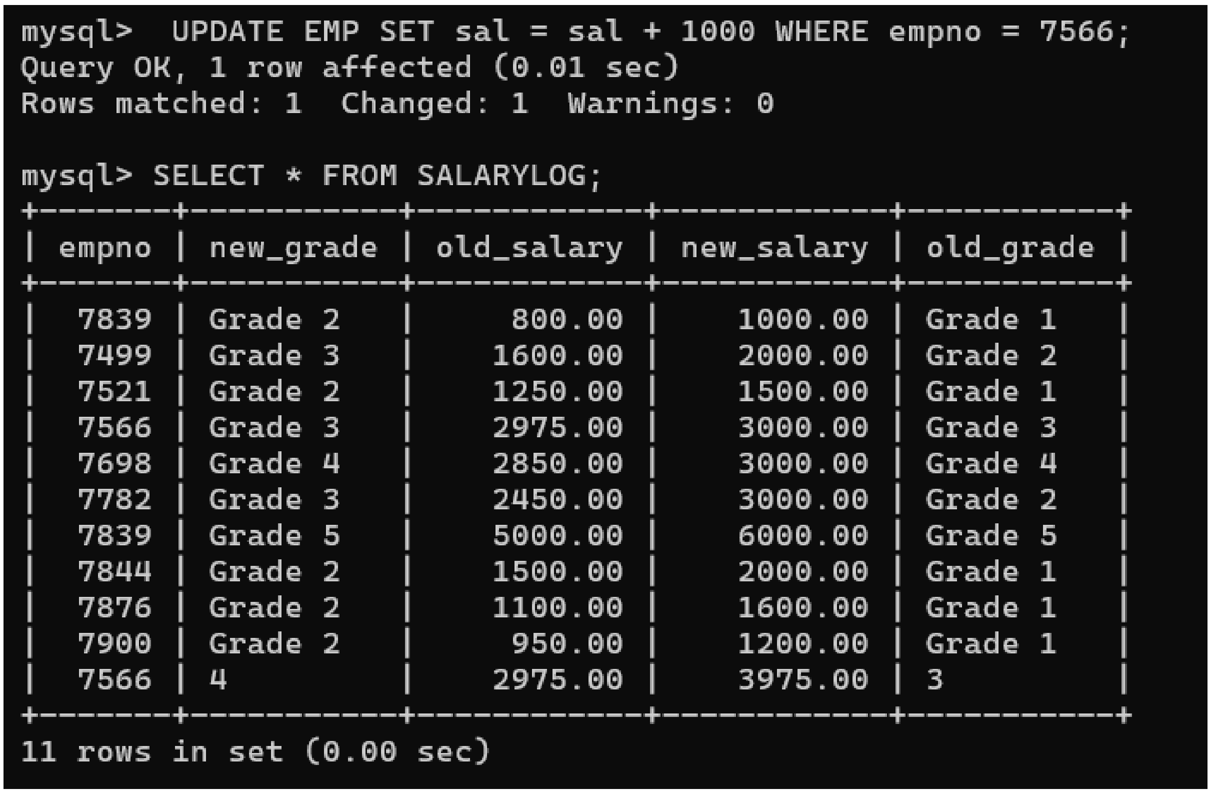


```
mysql> SELECT * FROM EMP;
+-------+--------+-----------+------+------------+---------+---------+--------+
| empno | ename  | job       | mgr  | hiredate   | sal     | comm    | deptno |
+-------+--------+-----------+------+------------+---------+---------+--------+
|  7123 | SMITH  | CLERK     | 7902 | 2000-06-23 |  800.00 |    NULL |     20 |
|  7499 | ALLEN  | SALESMAN  | 7698 | 1998-08-15 | 1600.00 |  300.00 |     30 |
|  7521 | WARD   | SALESMAN  | 7698 | 1996-03-26 | 1250.00 |  500.00 |     30 |
|  7566 | JONES  | MANAGER   | 7839 | 1995-10-31 | 2975.00 |    NULL |     20 |
|  7654 | MARTIN | SALESMAN  | 7698 | 1998-12-05 | 1250.00 | 1400.00 |     30 |
|  7698 | BLAKE  | MANAGER   | 7839 | 1992-06-11 | 2850.00 |    NULL |     30 |
|  7782 | CLARK  | MANAGER   | 7839 | 1993-05-14 | 2450.00 |    NULL |     10 |
|  7788 | SCOTT  | ANALYST   | 7566 | 1996-03-05 | 3000.00 |    NULL |     20 |
|  7839 | KING   | PRESIDENT | NULL | 1990-06-09 | 5000.00 |    NULL |     10 |
|  7902 | FORD   | ANALYST   | 7566 | 1997-12-05 | 3000.00 |    NULL |     20 |
+-------+--------+-----------+------+------------+---------+---------+--------+
10 rows in set (0.00 sec)
```



```
mysql> CREATE TRIGGER TR_CHECK_GRADE AFTER UPDATE ON EMP
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE v_old_grade VARCHAR(10);
    ->     DECLARE v_new_grade VARCHAR(10);
    ->
    ->     -- Retrieve the old and new grades based on old and new salary values
    ->     SELECT grade INTO v_old_grade FROM SALGRADE WHERE OLD.sal BETWEEN losal AND hisal;
    ->     SELECT grade INTO v_new_grade FROM SALGRADE WHERE NEW.sal BETWEEN losal AND hisal;
    ->
    ->     -- Check if the grade has changed
    ->     IF v_old_grade <> v_new_grade THEN
    ->         -- Log the details into salaryLog table
    ->         INSERT INTO salaryLog (empno, old_salary, new_salary, old_grade, new_grade)
    ->         VALUES (NEW.empno, OLD.sal, NEW.sal, v_old_grade, v_new_grade);
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
```

Checking if it worked:



```
mysql>  UPDATE EMP SET sal = sal + 1000 WHERE empno = 7566;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM SALARYLOG;
+-------+-----------+------------+------------+-----------+
| empno | new_grade | old_salary | new_salary | old_grade |
+-------+-----------+------------+------------+-----------+
|  7839 | Grade 2   |     800.00 |    1000.00 | Grade 1   |
|  7499 | Grade 3   |    1600.00 |    2000.00 | Grade 2   |
|  7521 | Grade 2   |    1250.00 |    1500.00 | Grade 1   |
|  7566 | Grade 3   |    2975.00 |    3000.00 | Grade 3   |
|  7698 | Grade 4   |    2850.00 |    3000.00 | Grade 4   |
|  7782 | Grade 3   |    2450.00 |    3000.00 | Grade 2   |
|  7839 | Grade 5   |    5000.00 |    6000.00 | Grade 5   |
|  7844 | Grade 2   |    1500.00 |    2000.00 | Grade 1   |
|  7876 | Grade 2   |    1100.00 |    1600.00 | Grade 1   |
|  7900 | Grade 2   |     950.00 |    1200.00 | Grade 1   |
|  7566 | 4         |    2975.00 |    3975.00 | 3         |
+-------+-----------+------------+------------+-----------+
11 rows in set (0.00 sec)
```