Bank Management System

MySql Project

By- Ankita

BANK MANAGEMENT SYSTEM

The bank management system is an application for maintaining a personal account in a bank . The system provides the access to the customer to create an account, deposit/withdraw the cash from his account, also to view reports of all accounts present.

Banking Management System thus ensures smooth operation of the Real-Estate management tasks as well as keeps the information about the employees and their salary.

Bank is the place where customers feel the sense of safety for their property. In the bank, customers deposit and withdraw their money. Transaction of money also is a part where customer takes shelter of the bank. Now to keep the belief and trust of customers, there is the positive need for management of the bank, which can handle all this with comfort and ease. Smooth and efficient management affects the satisfaction of the customers and staff members, indirectly. And of course, it encourages management committee in taking some needed decision for future enhancement of the bank. Now a days, managing a bank is tedious job upto certain limit. So software that reduces the work is essential. Also today’s world is a genuine computer world and is getting faster and faster day-by-day. Thus, considering above necessities, the software for bank management has became necessary which would be useful in managing the bank more efficiently.

# OBJECTIVE OF THE PROJECT

A computer-based management system is designed to handle all the primary information required to calculate monthly statements of customer account which include monthly statement of any month. Separate database is maintained to handle all the details required for the correct statement calculation and generation.

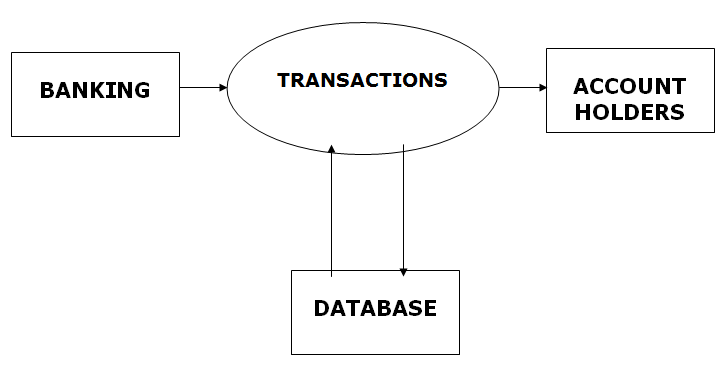
This project intends to introduce more user friendliness in the various activities such as record updation, maintenance, and searching. The searching of record has been made quite simple as all the details of the customer can be obtained by simply keying in the identification or account number of that customer. Similarly, record maintenance and updation can also be accomplished by using the account number with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.

The main objective of our project is providing the different typed of customers facility, the main objective of this system is to find out the actual customer service. Etc.

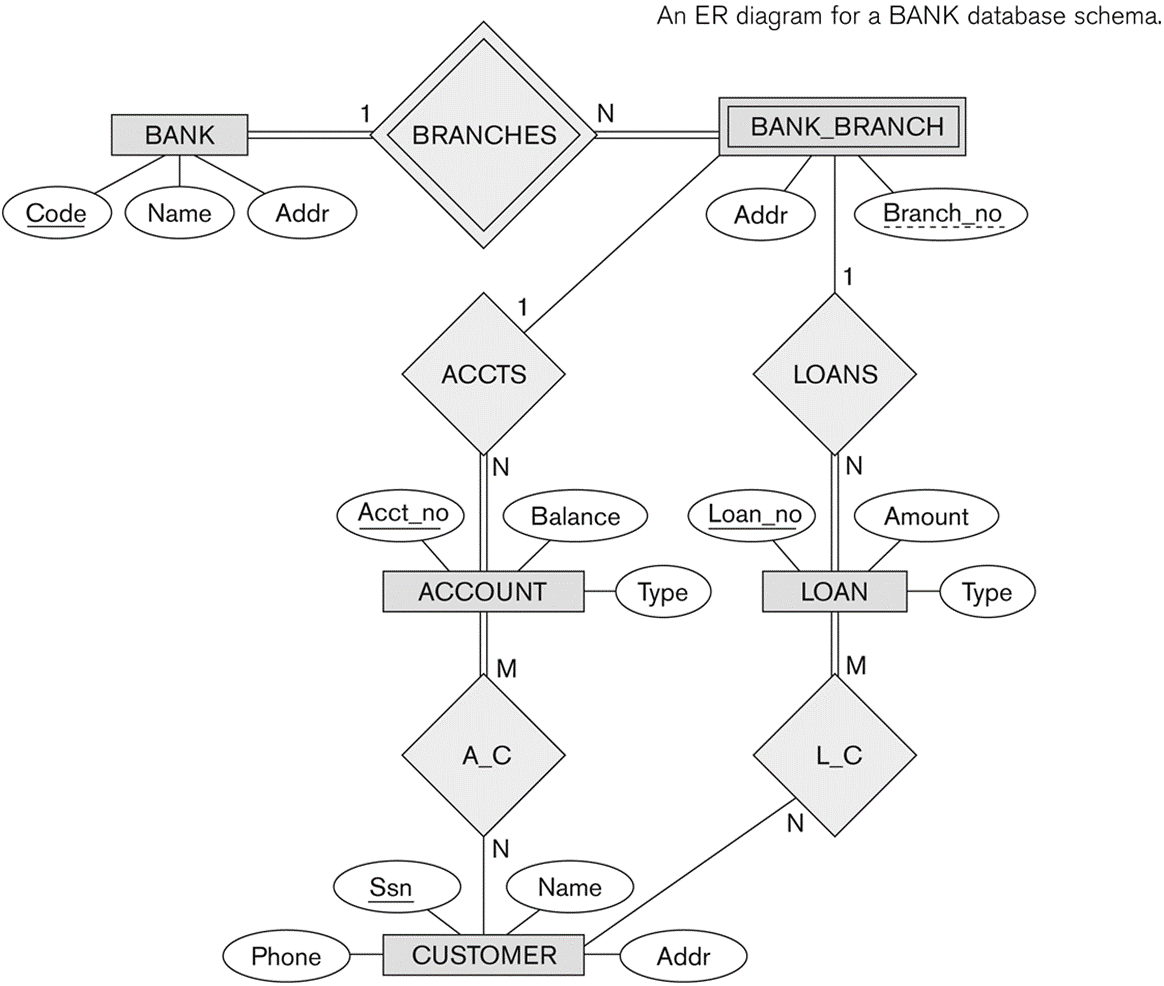
## It should fulfill almost all the process requirements of any Bank.

* It should increase the productivity of bank by utilizing the working hours more and more, with minimum manpower.

# SYSTEM DATA FLOW DIAGRAM



# **ER-DIAGRAM**



DATABASE SPECIFICATIONS

**“Account\_info” Table: -**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Constraints** |
| account\_number | char(5) | Primary Key |
| branch\_name | Varchar(7) | Not null |
| balance | double | Not Null |

**“Branch\_info” Table: -**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Constraints** |
| branch\_name | Varchar(10) | Primary Key, not null |
| branch\_city | Varchar(10) | Not null |
| assets | double | Not null |

**“Depositor\_info” Table: -**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Constraints** |
| account\_number | Char(5) | Not null, foreign key |
| customer\_name | Varchar(20) | Not null , foreign key |

**“Loan\_info” Table: -**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Constraints** |
| loan\_number | Varchar(5) | Not null , primary key |
| branch\_name | Varchar(10) | Not null |
| amount | Double | Not null |

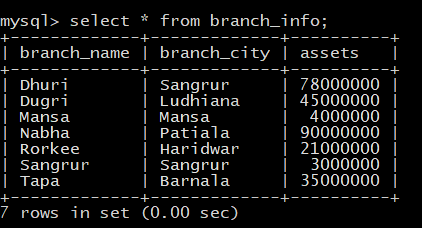
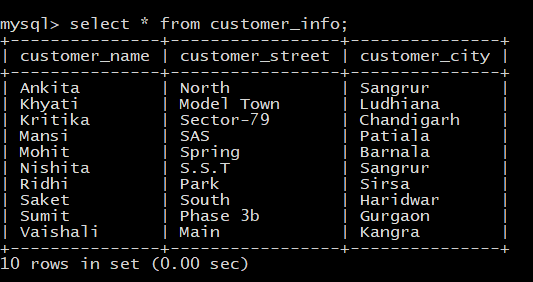
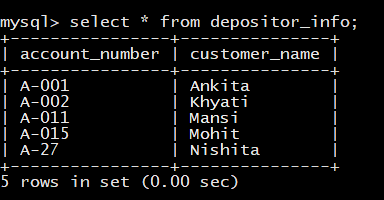
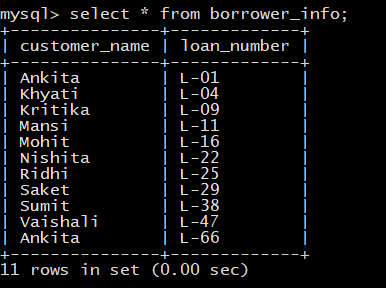
**“Customer\_info” Table: -**

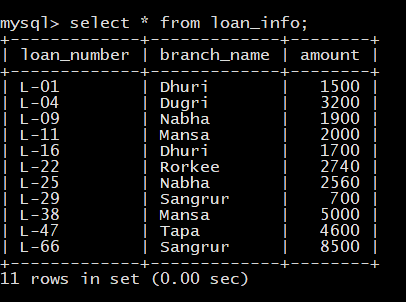
|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Constraints** |
| customer\_name | Varchar(20) | Not null , primary key |
| customer\_street | Varchar(20) | Not null |
| customer\_city | Varchar(10) | Not null |

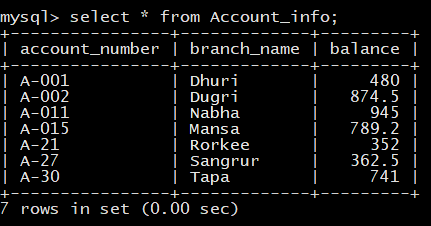
**“Borrower\_info” Table: -**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Constraints** |
| customer\_Name | Varchar(20) | Not null, foreign key |
| loan\_number | Varchar(5) | Not null, foreign key |

Queries

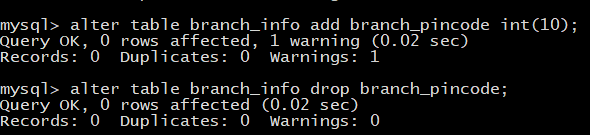


****

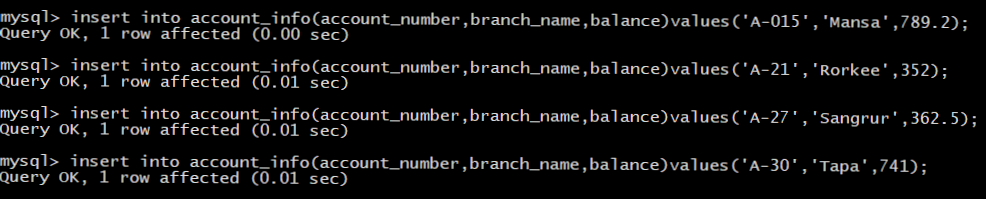
****

1. **ADD/REMOVE ATTRIBUTE**

We use the alter table command to add/remove attributes to an existing relation. When we add an attribute all tuples in the relation are assigned null as the value for the new attribute.

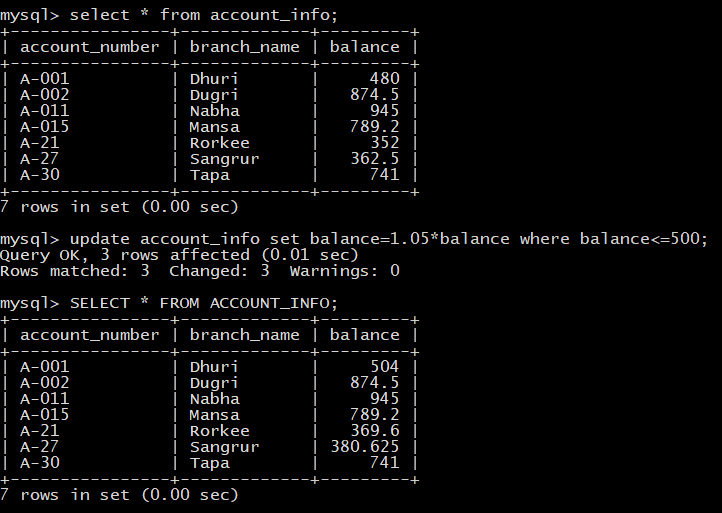


1. **INSERT QUERY**

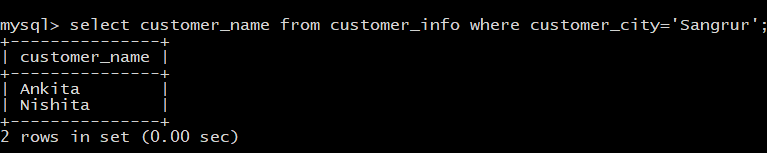


1. **UPDATE QUERY**

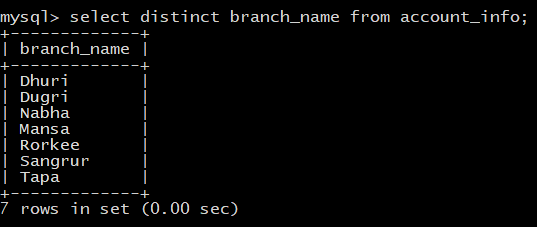
If interest is to be paid only to accounts with a balance of Rs. 500 or less, we can write



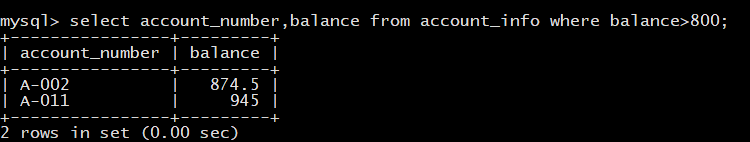
1. **SELECT QUERY**
2. Find all name of customers whose city is in ‘Sangrur’.



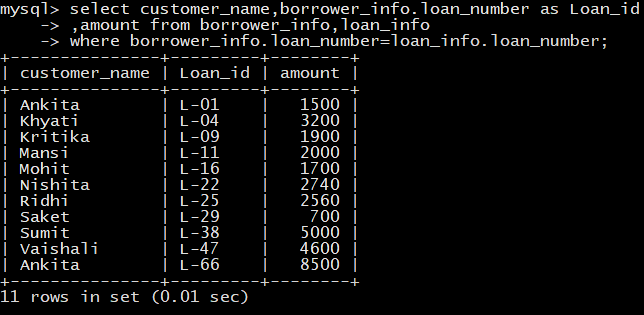
1. Find the names of all branches in the account information, don't display duplicates.



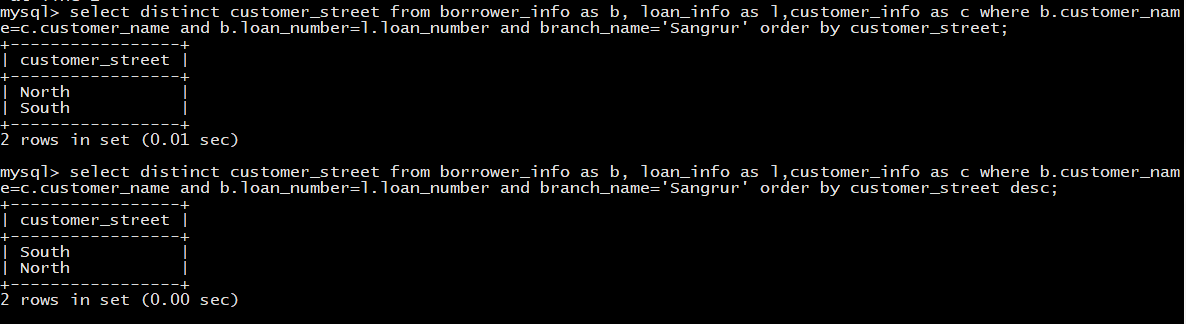
1. Find the account number and balance for all accounts from Account\_info where the balance is greater than INR 800.

****

1. SQL provides a mechanism for renaming both relations and attributes. It uses the **as** clause, taking the form *old-name****as****new-name.* The **as** clause can appear in both the **select** and **from** clauses.



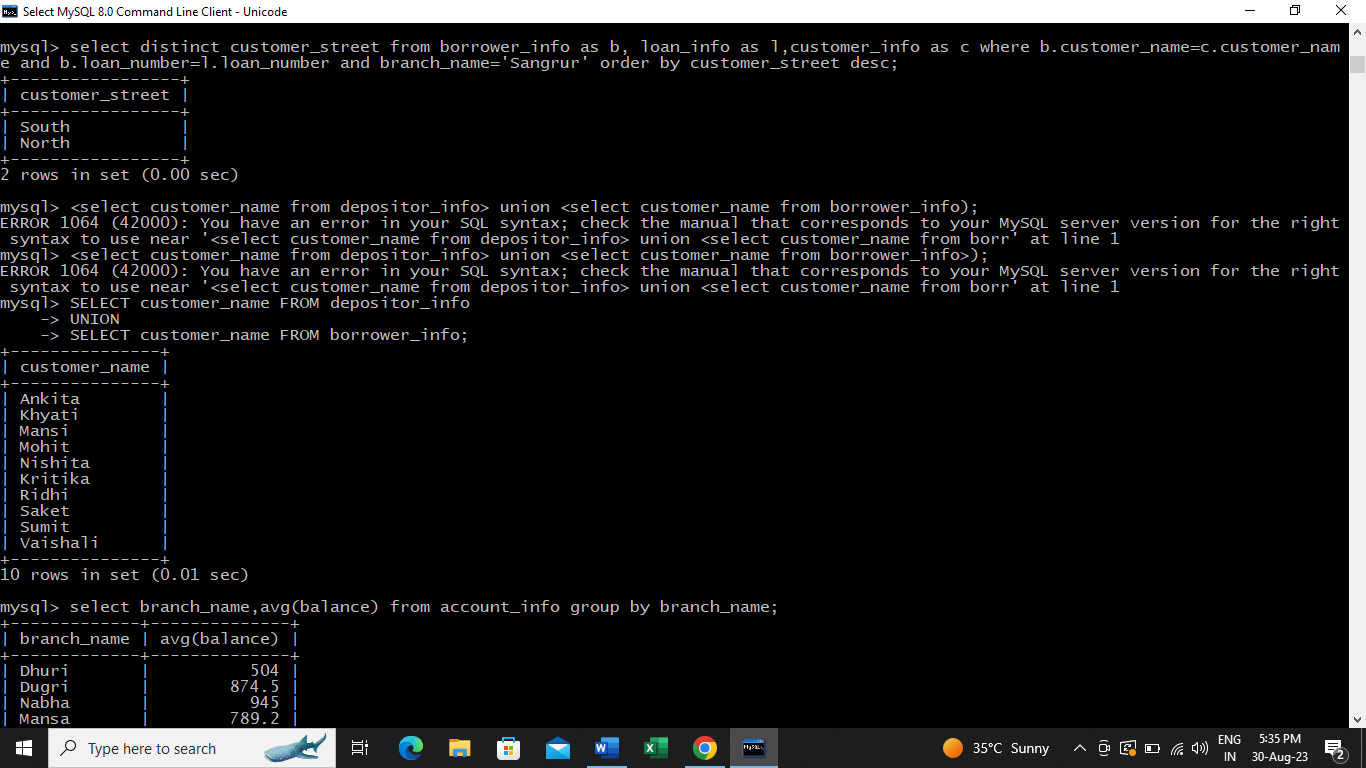
1. Find the list of all streets in alphabetic order who have a loan at the downtown branch in ASCENDING and DESCENDING order



1. **SET OPERATIONS**

* **UNION OPERATION:** The union operation automatically eliminates duplicates. If we want to retain all duplicates, we must write **union all** in place of **union.**

To find all customers having a loan, an account, or both at the bank, we write:



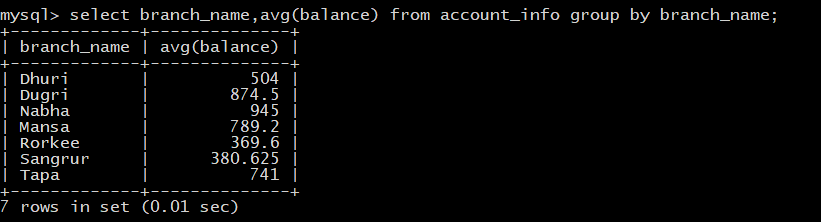
1. **AGGREGATE FUNCTIONS**

Aggregate functions are functions that take a collection (a set or multiset) of values as input and return a single value. SQL offers five built-in aggregate functions:

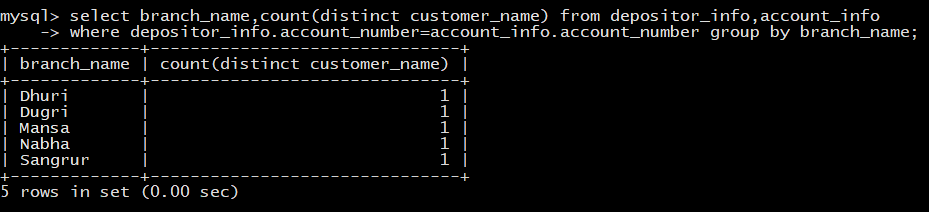
* Average : AVG
* Minimum : MIN
* Maximum : MAX
* Total : SUM
* Count : COUNT

The input to sum and avg must be a collection of numbers, but the other operators can operate on collections of nonnumeric data types, such as strings, as well.

1. Find the average account balance at each branch.

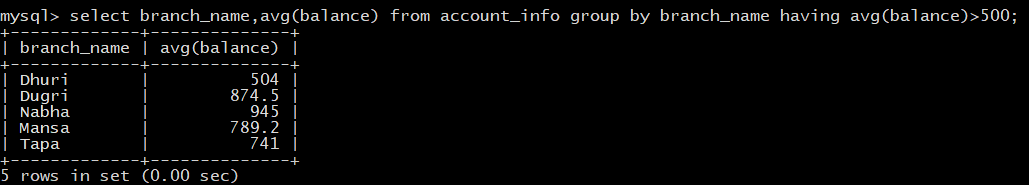


1. Find the number of depositors for each branch.

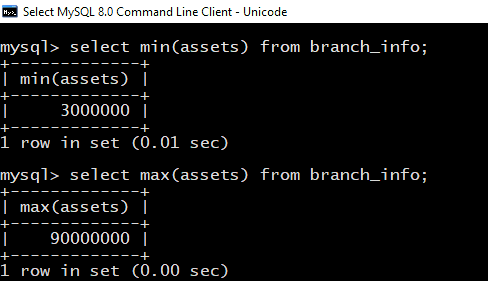


1. **HAVING :**

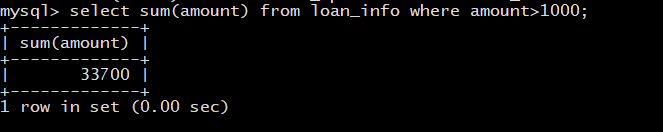
Find all the branches where the average account balance is more than Rs. 500



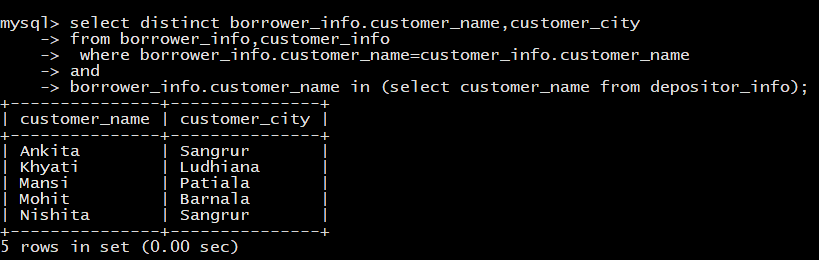
1. **MIN and MAX:**



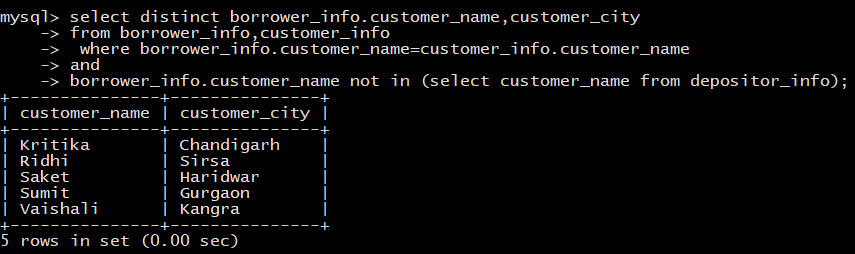
1. **SUM:**

****

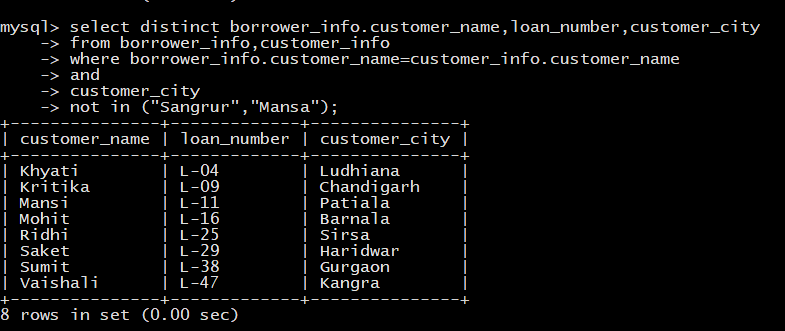
1. **NESTED SUBQUERIES**
2. Find all customers who have both an account and a loan at the bank



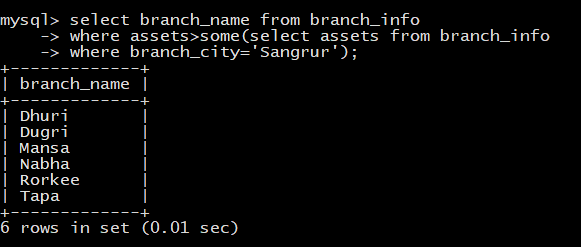
1. Find all the customer who have at last one loan but not have any account



1. Find the names of customers who have a loan at the bank, and whose cities are neither ‘Sangrur and Mansa’

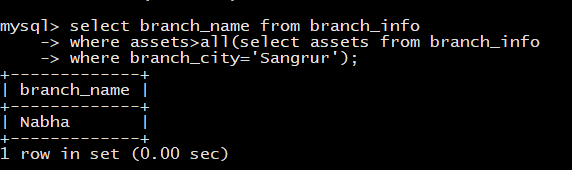


1. **SET COMPARISON**
   1. Find the names of all branches that have assets greater than those of at least one branch located in Sangrur.



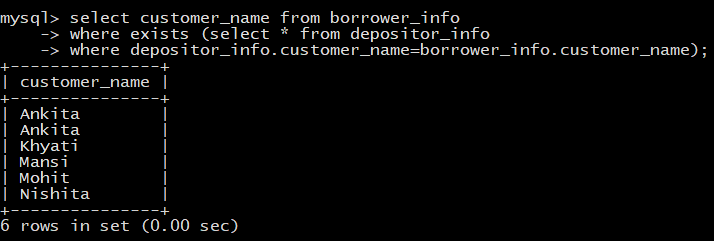
SQL also allows < **some**, <= **some**, >= **some**, = **some**, and <> **some** comparisons. As an exercise, verify that = **some** is identical to **in**, whereas <> **some** is not the same as **not in**. The keyword **any** is synonymous to some in SQL. Early versions of SQL allowed only **any**.

* 1. Find the names of all branches that have an asset value greater than that of each branch in ‘Sangrur’

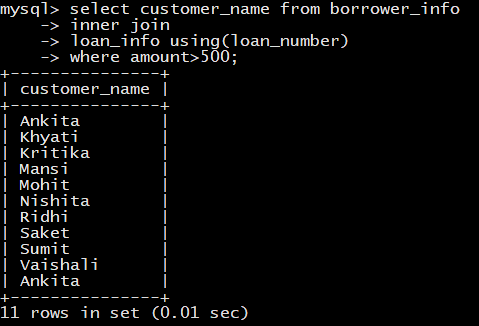


SQL includes a feature for testing whether a subquery has any tuples in its result. The **exists** construct returns the value true if the argument subquery is nonempty.

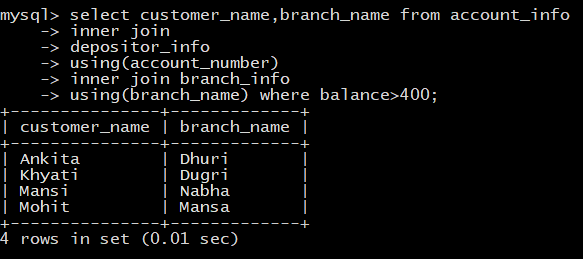
* 1. Find all customers who have both an account and a loan at the bank



1. **JOINTS**
2. Select customer name which has loan amount more than 500



1. Select customer name and branch name where account balance is more than 400



CONCLUSION

This banking system project will serve as a useful approach to data base dialog box to deposit and withdraw the money for the person. It serves as a helpful approach for the users. It provides easy way of the deposit and withdraws the money. It reduces the time taken by the user to save the money. Thus, the project is the user -friendly approach.