DATABASE MANAGEMENT REPORT

PROJECT TITLE: Online Banking System

ORGANISATION: Secure Bank Operation

To
Dr. Yibai Li
BUAN 576 Business Database Management Systems
December 11, 2023

INTRODUCTION:

Secure Bank Corporation is a leading financial institution that specializes in providing a wide range of banking and financial services to individuals, businesses, and institutions. Established in 1975, Secure Bank has a long-standing tradition of excellence and commitment to its customers.
Secure Bank Corporation is a prominent financial institution offering a wide array of services. Their retail banking services encompass savings and checking accounts, personal loans, and credit cards, accessible through both online and mobile platforms, as well as physical branches. They excel in investment and wealth management, delivering informed financial decisions through portfolio management and advisory services. Secure Bank supports businesses of all sizes with business loans, merchant services, and customized banking solutions. Their focus on technology ensures a secure, user-friendly online banking platform. They prioritize security and compliance, adhering to all regulations. Exceptional customer service is a fundamental value, providing support and guidance. Secure Bank's vision is to empower customers with innovative solutions, and this project aims to enhance their online banking system for a seamless and secure user experience.

BUSINESS RULES:

Every client has personal information, including name, address, and phone number.

- **Clients:** Clients can maintain many accounts (savings, checking, etc.).
- **Accounts:** Each consumer has an individual account.

There are several kinds of accounts, such as credit card, savings, investment, and checking.

- **Transactions:** Transactions comprise amounts, dates, and kinds (deposit, withdrawal, and transfer) and are made on certain accounts.
- **Security and Compliance:** To access any account, you must first authenticate.

Two-factor authentication and transaction verification for significant sums are part of security protocols.

- **Online and Mobile Banking:** These systems allow cash transfers, bill payments, account management, and more.

Entity-Relationship-DIAGRAM [ER]:

Entity Relationship Diagram is used in modern database software engineering to illustrate logical structure of database.
It is a relational schema database modelling method used to Model a system and approach.
This approach commonly used in database design. The diagram created using this method is called ER-diagram.
The ER-diagram depicts the various relationships among entities, considering each object as entity.
Entity is represented as rectangle shape and relationship represented as diamond shape.
It depicts the relationship between data objects. The ER-diagram is the notation that is used to conduct the data modelling activity.

ER diagram of Bank has the following description:

- Banks have Customer.
- Banks are identified by a name, code, address of main office.
- Banks have branches.
- Branches are identified by a branch_no., branch_name, address.
- Customers are identified by name, cust-id, phone number, address.
- Customer can have one or more accounts.
- Accounts are identified by account_no., acc_type, balance.
- Customer can avail loans.
- Loans are identified by loan_id, loan_type and amount.
- Account and loans are related to bank's branch.

Entities and their Attributes are:

- Customer Entity: Attributes of Customer Entity are Customer_id, Name, Phone Number and Address.
- Customer_id is Primary Key for Customer Entity.
- Branch Entity: Attributes of Branch Entity are Branch_id, Name and Address.
- Branch_id is Primary Key for Branch Entity.
- Account Entity: Attributes of Account Entity are Account_number, Account_Type and Balance.
- Account_number is Primary Key for Account Entity.

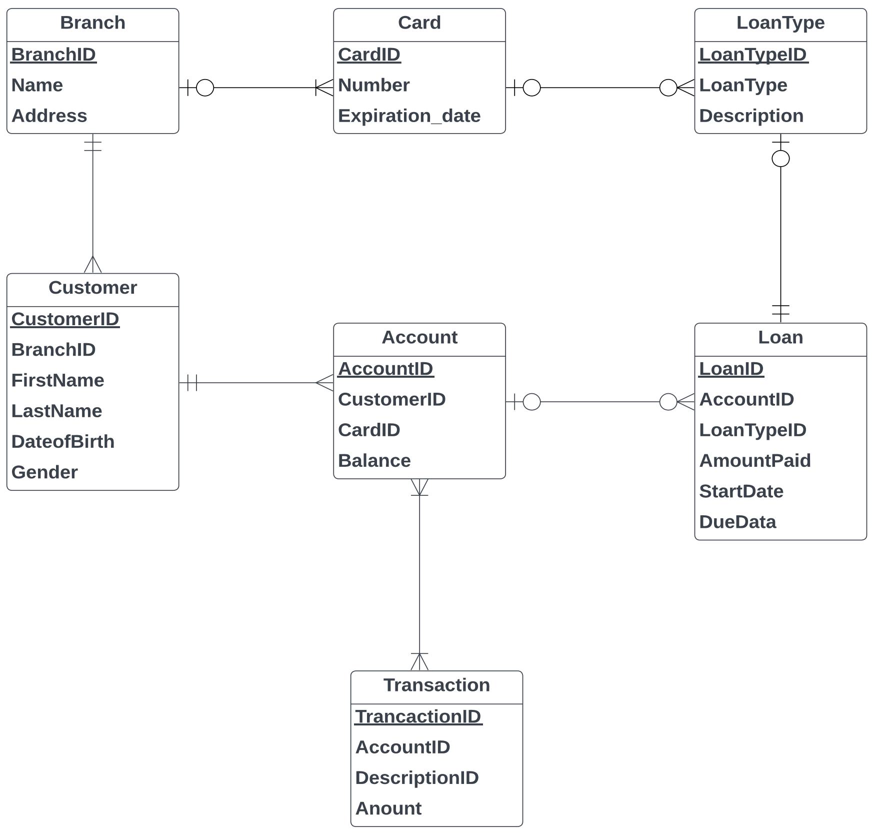
- Loan Entity: Attributes of Loan Entity are Loan_id, Loan_Type and Amount.
- Loan_id is Primary Key for Loan Entity.

Relationships – cardinalities

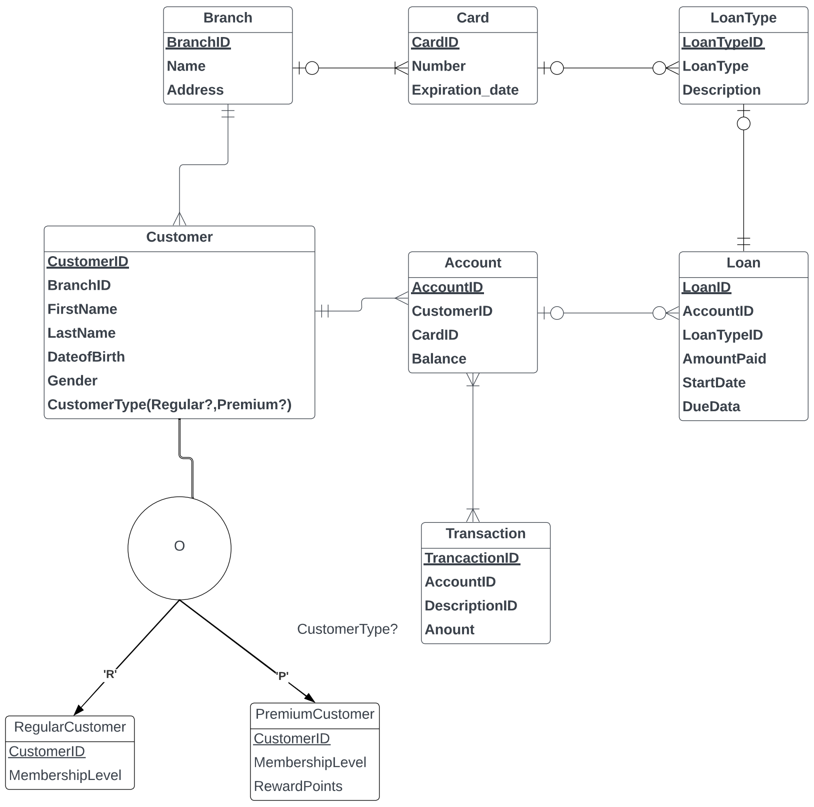
- Bank has Branches => 1 : N
- Branch maintain Accounts => 1 : N
- Branch offer Loans => 1 : N
- Account held by Customers => M : N
- Account held by Customers => M : N

Branch entity and the Customer entity -This means that one branch can have many customers (each customer is associated with a branch), but each customer belongs to only one branch.
Card entity and the Customer entity: This means that one customer can have multiple cards (credit or debit cards), but each card belongs to only one customer.
Customer entity and the Account entity: This means that one customer can have multiple accounts, but each account belongs to only one customer.
Account entity and the Loan entity: This means that one account can have multiple loans associated with it, but each loan is related to a specific account.
Account entity and the Transaction entity: This means that one account can have multiple transactions, but each transaction is linked to a specific account.

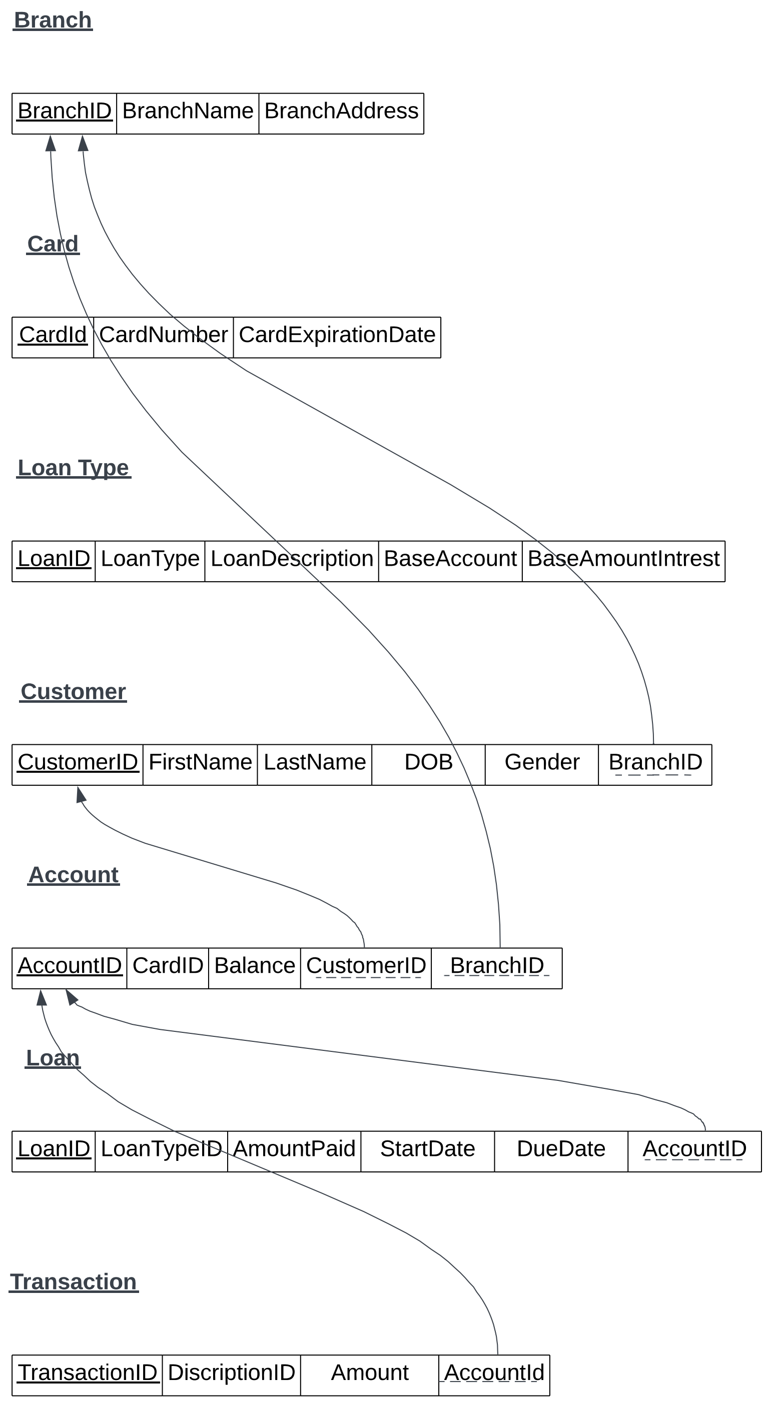
ENTITY RELATIONSHIP[ER]



ENHANCED ENTITY RELATIONSHIP[EER]



LOGICAL DESIGN (3rd NF):


Branch
BranchID | BranchName | BranchAddress
Card
CardId | CardNumber | CardExpirationDate
Loan Type
LoanID | LoanType | LoanDescription | BaseAccount | BaseAmountIntrest
Customer
CustomerID | FirstName | LastName | DOB | Gender | BranchID
Account
AccountID | CardID | Balance | CustomerID | BranchID
Loan
LoanID | LoanTypeID | AmountPaid | StartDate | DueDate | AccountID
Transaction
TransactionID | DiscriptionID | Amount | AccountId

The 3NF logical design for an online banking system involves multiple interconnected tables:

Branch: The branch holds the different branches of the bank like BranchID, Branch Name & and Branch Address.

Card: Stores information on bank cards, such as the card number and expiration date.

Loan Type: Contains information about different types of loans offered by the bank.

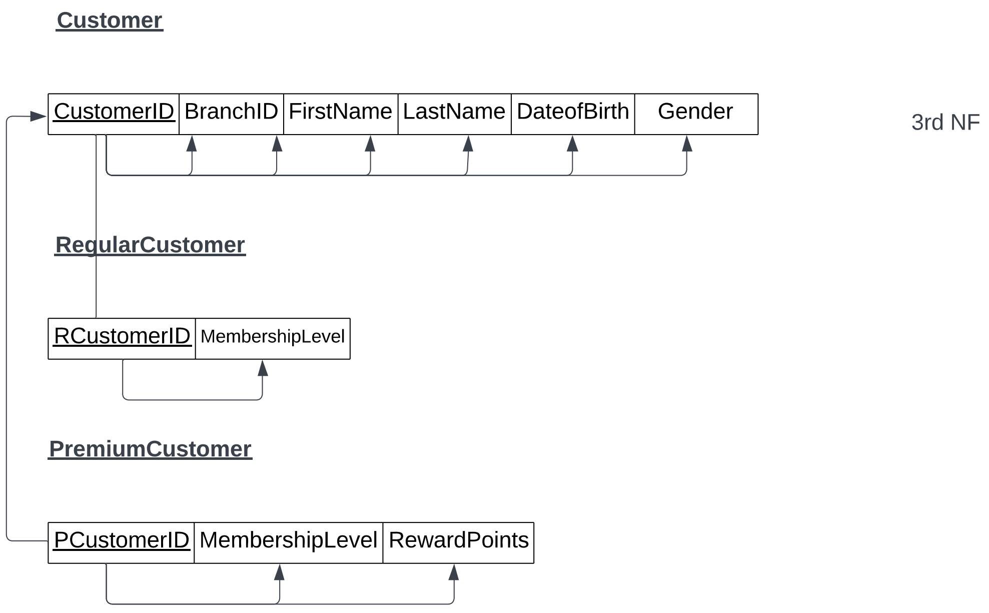
Customer: Stores customer details like customer ID, first name, Last name, DOB, Gender, including their branch affiliation.

Account: Manages customer accounts, linking both customers and cards.

Loan: Tracks information about loans associated with customer accounts.

Transaction: Records transactions related to customer accounts.

FUNCTIONAL DEPENDENCY:



Functional dependencies describe the relationships between attributes in a relational database. These functional dependencies represent the links between supertype properties and their subtypes. They describe how changes in an attribute affect the values of other attributes in the same table. In this case for the supertype and the subtype Customer is both foreign key and primary key.

DATA TYPES USED AND THEIR DESCRIPTION

Integer: one optional sign character (+ or -) followed by at least one digit (0-9). Leading and trailing blanks are ignored. No other character is allowed.

Varchar: It is used to store alpha numeric characters. In this data type we can set the maximum number of characters up to 8000 ranges by defaults SQL server will set the size to 50 characters range.

Date: The DATE data type accepts date values. No parameters are required when declaring a DATE data type. Date values should be specified in the form: YYYY-MM- DD. However, Point Base will also accept single digits entries for month and day values.

Time: The TIME data type accepts time values. No parameters are required when declaring a TIME data type. Time values should be specified in the form: HH:MM: SS. An optional fractional value can be used to represent nanoseconds.

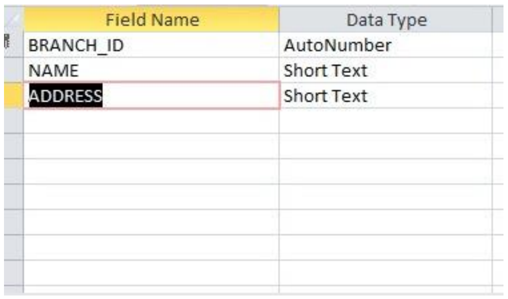
DIFFERENCE BETWEEN SQL AND ACCESS

- SQL Server is a server-based database while Microsoft Access is used as local database.
- Access is used for small desktop size databases used by less than 5 users at the same time and have a front-end GUI system to design applications quickly while SQL Server is a more robust system and is able to handle large amounts of users as well as data sizes and does not have front end GUI system so it will require other development tools (Visual Studios, .NET, VB, C++, etc.)
- Access employs security based on the actual database settings. Each database can have different security settings and employee lists. Logins are not tied to a users' Windows

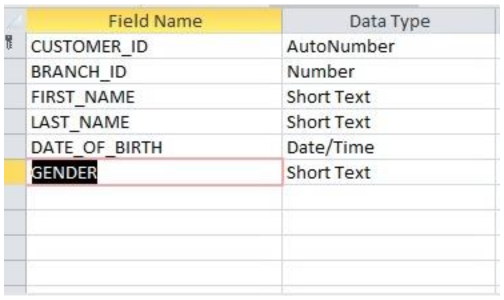
logon while in SQL, On Center Software recommends using Windows Authentication when setting up an SQL database, that way, the users' Windows security is transferred to the database and only valid users can access the database AND the security is centralized.

TABLES IN MICROSOFT ACCESS

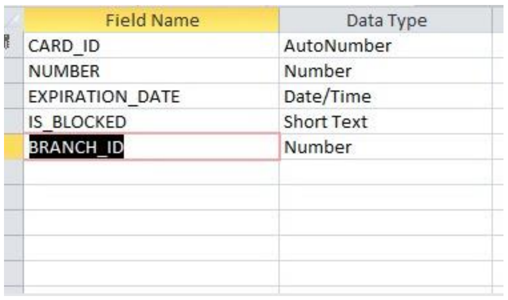
Below We have designed the data base using Microsoft Access and My SQL.



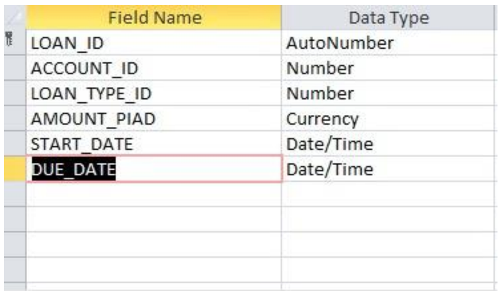
Field Name	Data Type
BRANCH_ID	AutoNumber
NAME	Short Text
ADDRESS	Short Text



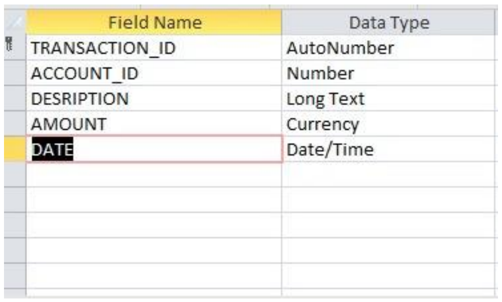
Field Name	Data Type
CUSTOMER_ID	AutoNumber
BRANCH_ID	Number
FIRST_NAME	Short Text
LAST_NAME	Short Text
DATE_OF_BIRTH	Date/Time
GENDER	Short Text



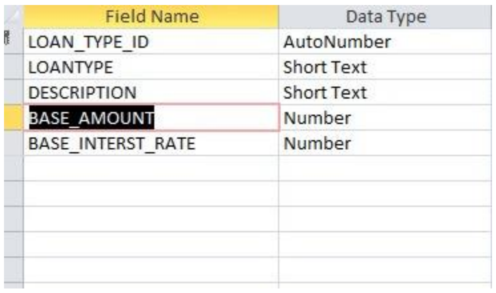
Field Name	Data Type
CARD_ID	AutoNumber
NUMBER	Number
EXPIRATION_DATE	Date/Time
IS_BLOCKED	Short Text
BRANCH_ID	Number



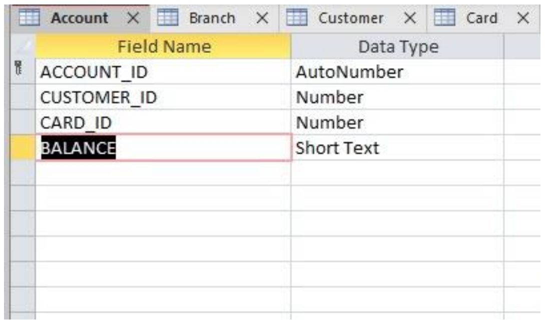
Field Name	Data Type
LOAN_ID	AutoNumber
ACCOUNT_ID	Number
LOAN_TYPE_ID	Number
AMOUNT_PIAD	Currency
START_DATE	Date/Time
DUE_DATE	Date/Time



Field Name	Data Type
TRANSACTION_ID	AutoNumber
ACCOUNT_ID	Number
DESRIPTION	Long Text
AMOUNT	Currency
DATE	Date/Time

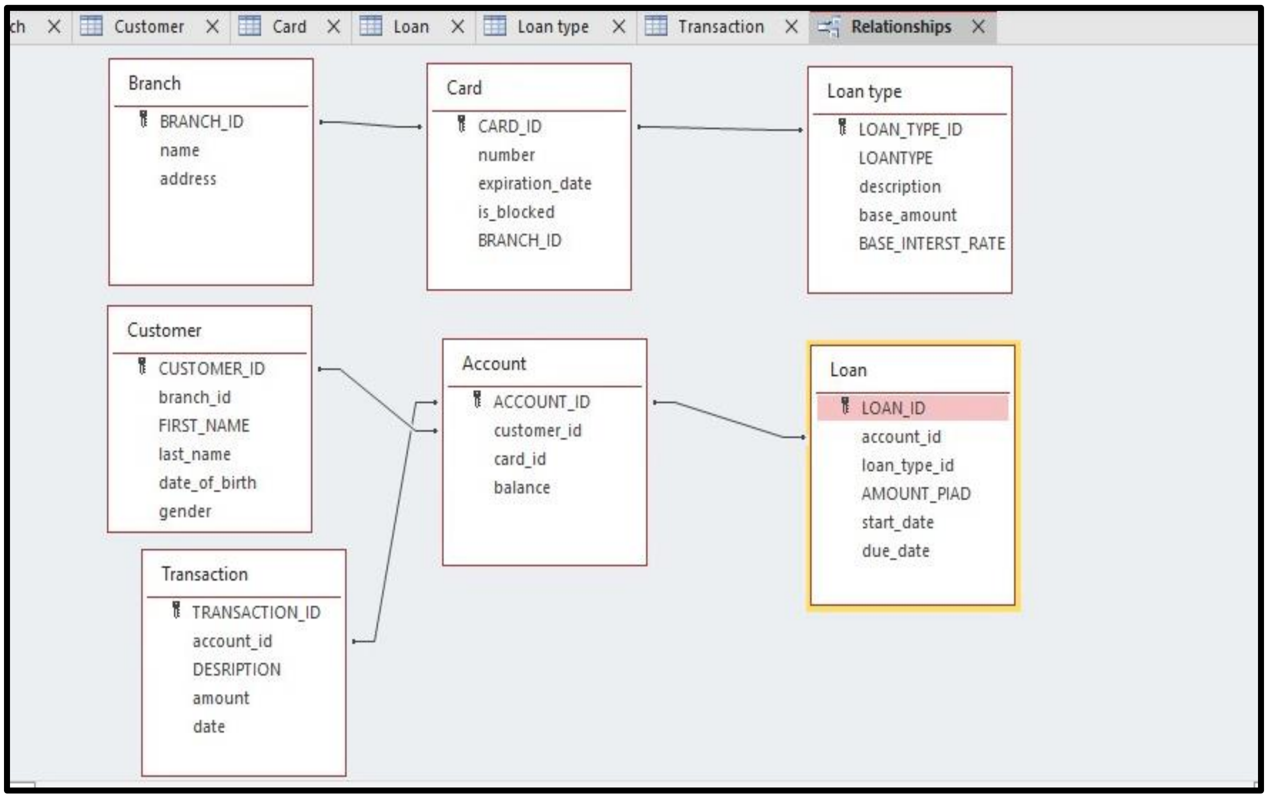


Field Name	Data Type
LOAN_TYPE_ID	AutoNumber
LOANTYPE	Short Text
DESCRIPTION	Short Text
BASE_AMOUNT	Number
BASE_INTERST_RATE	Number



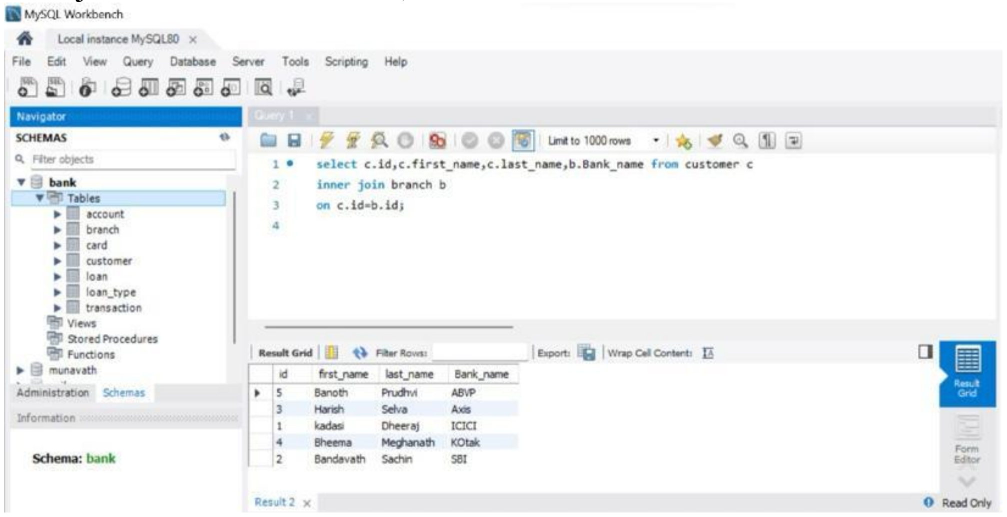
Field Name	Data Type
ACCOUNT_ID	AutoNumber
CUSTOMER_ID	Number
CARD_ID	Number
BALANCE	Short Text

RELATIONAL MODEL [ACCESS]



<u>SQL</u>

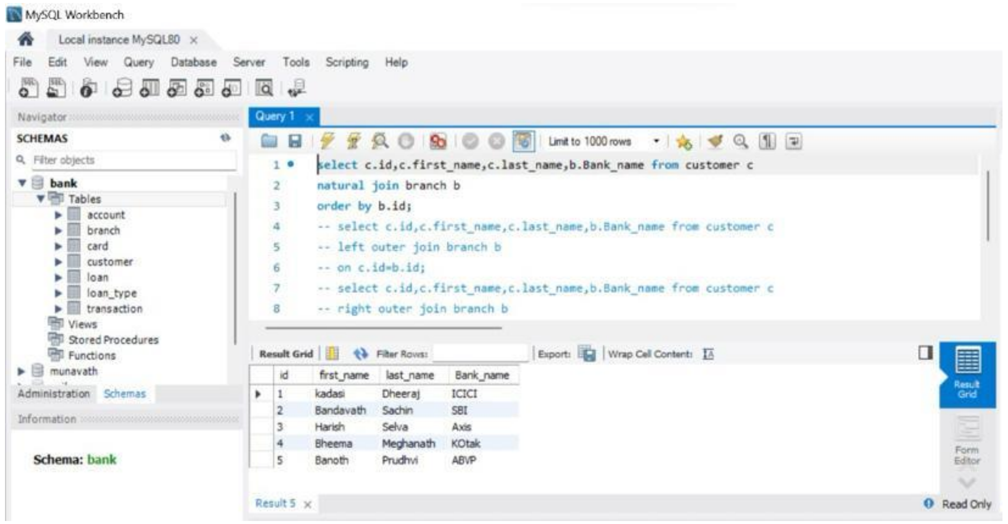
-- performed inner join on customer and branch table
select c.id,c.first_name,c.last_name,b.Bank_name from customer c
inner join branch bon c.id=b.id;



```
select c.id,c.first_name,c.last_name,b.Bank_name from customer c
inner join branch b
on c.id=b.id;
```

id	first_name	last_name	Bank_name
5	Banoth	Prudhvi	ABVP
3	Harish	Selva	Axis
1	kadasi	Dheeraj	ICICI
4	Bheema	Meghanath	KOtak
2	Bandavath	Sachin	SBI

-- performed natural join on customer and branch table
select c.id,c.first_name,c.last_name,b.Bank_name from customer c
natural join branch b
order by b.id;



```
select c.id,c.first_name,c.last_name,b.Bank_name from customer c
natural join branch b
order by b.id;
-- select c.id,c.first_name,c.last_name,b.Bank_name from customer c
-- left outer join branch b
-- on c.id=b.id;
-- select c.id,c.first_name,c.last_name,b.Bank_name from customer c
-- right outer join branch b
```

id	first_name	last_name	Bank_name
1	kadasi	Dheeraj	ICICI
2	Bandavath	Sachin	SBI
3	Harish	Selva	Axis
4	Bheema	Meghanath	KOtak
5	Banoth	Prudhvi	ABVP

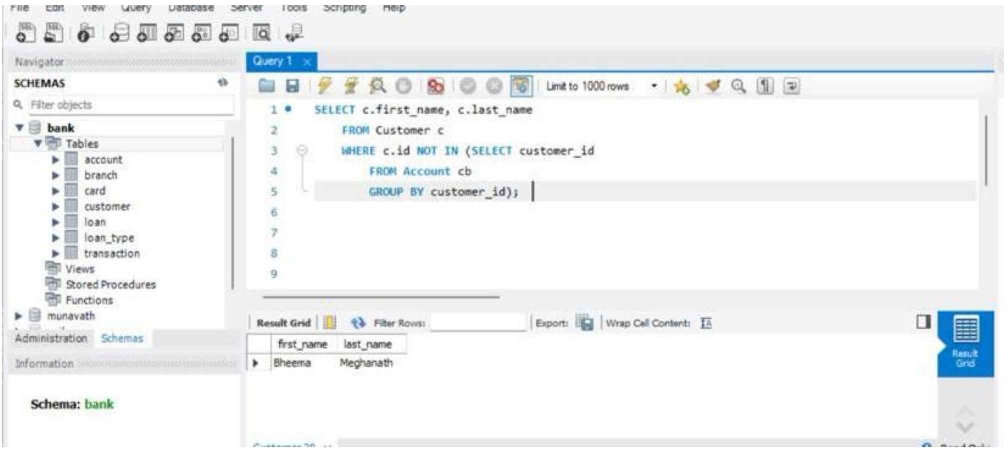
-- performed left outer join on customer and branch table
select c.id,c.first_name,c.last_name,b.Bank_name from customer c
left outer join branch b on c.id=b.id;

-- performed right outer join on customer and branch table

```
select c.id,c.first_name,c.last_name,b.Bank_name from customer c
right outer join branch b
on c.id=b.id;
```

-- performed cross join on customer and branch table

```
select c.id,c.first_name,c.last_name,b.Bank_name from customer c
cross join branch b
 on c.id=b.id
```



```
SELECT c.first_name, c.last_name
    FROM Customer c
    WHERE c.id NOT IN (SELECT customer_id
        FROM Account cb
        GROUP BY customer_id);
```

```
  first_name  last_name
▸ Bheema      Meghanath
```

-- list of customer that have accounts in two or more branches

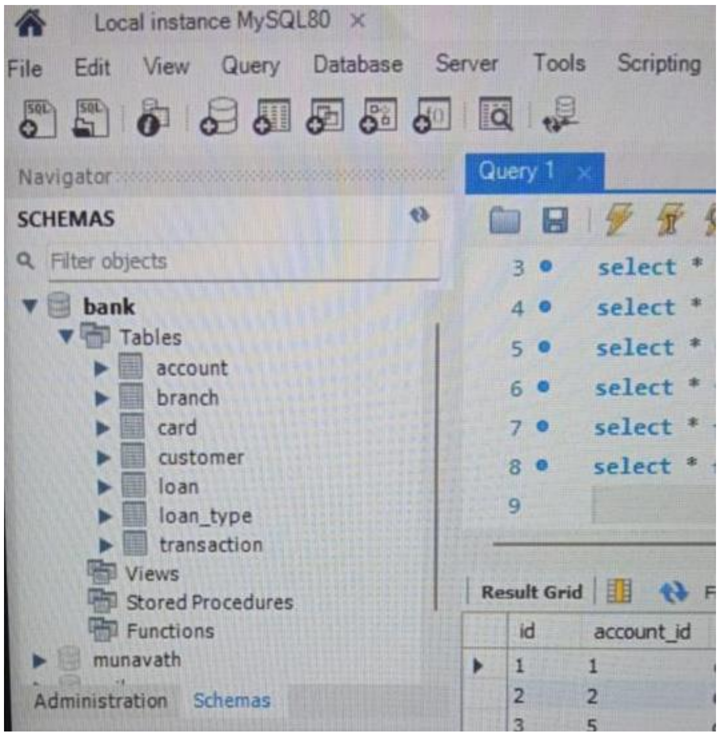
```
SELECT c.first_name, c.last_name
     FROM Customer c
     WHERE c.id IN (SELECT customer_id
         FROM Customer_Branch cb
      GROUP BY customer_id
      HAVING COUNT(*) >= 2);
```

-- who takes loans more often men or women

```
SELECT gender, COUNT(*) AS count
FROM Customer AS c
WHERE c.id IN (
     SELECT customer_id
   FROM Account AS a
   WHERE a.id IN (
         SELECT account_id
         FROM Loan AS l))
GROUP BY gender
ORDER BY count DESC;
```

```
-- list of customer that never had a loan
SELECT c.first_name, c.last_name
      FROM Customer c
      WHERE c.id IN (SELECT a.customer_id
           FROM Account a
           WHERE a.id NOT IN (SELECT l.account_id
                                 FROM Loan l));
```

```
-- find customer who have no open accounts
SELECT c.first_name, c.last_name
      FROM Customer c
      WHERE c.id NOT IN (SELECT customer_id
           FROM Account cb
       GROUP BY customer_id);
```



```
select *
select *
select *
select *
select *
select *
```

```
id  account_id
1   1
2   2
3   5
```

CONCLUSION:
This project is all about creating and using databases. It starts with drawing ER diagrams to plan the database, then by using Logical design we are organizing the data in a smart way to make it reliable and fast. Finally, we use SQL, to find and analyze the data. This helps us manage information better and make smart decisions. The project shows how important good database design and the right tools to get data in various fields.