Swift Bank Management System

Problem Statement

The management of customer accounts, balances, and deposits poses a complex and time-consuming challenge for banking institutions, with potential financial losses and reputational damage resulting from inadequate management. Banks face several obstacles when managing customer accounts, balances, and deposits, including:

- Managing multiple accounts: Banks must manage multiple accounts for a single customer, such as checking accounts, savings accounts, and credit card accounts, which can be particularly challenging and time-consuming, especially if manual data entry and paper records are used.

- Large customer base: Banks have thousands or even millions of customers, each with their own unique account details. Tracking all of this information can be overwhelming without the use of a centralized database system.

- Complex transactions: Banks must manage a wide range of complex transactions, including wire transfers, loan disbursements, and foreign currency exchanges. Such transactions can be challenging to manage, especially if they involve multiple accounts and currencies.

- Security risks: Ensuring data security and protection from unauthorized access and hacking attempts is critical.

- Compliance requirements: Banks must comply with a range of regulatory requirements, such as Anti-Money Laundering (AML) and Know Your Customer (KYC) regulations. Failure to comply with these requirements can result in hefty fines and reputational damage.

To manage customer accounts, balances, and deposits more efficiently, reduce errors, and provide better customer service, it is essential for banks to use a centralized database system. The proposed solution is the implementation of a Swift Bank Management System to streamline banking operations and enhance the customer experience.

Functionality

Note - Certain assumptions were made regarding the structure and content of the tables to improve the functionalities for the bank, all of which are clearly mentioned below.

The Swift Bank Management System is designed to provide an array of advanced functionalities to aid the bank in **effectively** managing customer accounts, balances, and deposits. These functionalities are based on certain assumptions and include the following:

- Customer Management: The system enables the bank to maintain a centralized database of customer information, such as name, address, contact details, and account details, allowing it to track all customers and their associated accounts and transactions. This helps to reduce the risk of errors and ensures that all customer information is up-to-date. We have assumed that multiple account holders can share the same address, such as individuals from the same family residing together.

- Account Management: The system allows the bank to maintain a centralized database of account information, including account type, debit card information, transaction status, and transaction history. This helps the bank to track all accounts and associated transactions in one place, reducing the risk of errors and ensuring that all account information is up-to-date. We have also assumed that the system provides functionality to allow customers to obtain add-on debit cards. Additionally, the system tracks the status of debit cards as Active or Expired. The transaction status is dynamic and can change from Initiated to Pending to On Hold to Failed or Successful.

- Deposit and Withdrawal Management: The system facilitates the bank in managing customer deposits and withdrawals with advanced features for real-time transactions, enabling faster and more accurate processing of transactions, reducing the risk of errors, and ensuring prompt disbursement of funds.

- Transfer Management: The system allows the bank to manage transfers made between different accounts within the bank and includes features for merchant details addition. The database design includes the storage of unique MerchantID, Merchant name, phone, email, and address. The system also tracks the MerchantID for each transaction.

- Reporting: The system provides reports on customer information, account information, and transaction history, with customizable reporting features, enabling the bank to generate reports on various aspects of its operations, such as customer demographics, account balances, and transaction history. This allows the bank to make data-driven decisions and provide better customer service.

In summary, the Swift Bank Management System provides a comprehensive solution for managing customer accounts, balances, and deposits, improving operational efficiency, and enhancing customer service quality, all based on the aforementioned assumptions.

Entities

1) CUSTOMER – Contains all customer information uniquely identified by *CUSTOMERID*

2) ACCOUNT – Contains all account information held by customers uniquely identified by *ACCOUNTID.* This is also a supertype for 2 entities following total disjoint specialization

   2.1) SAVING_ACCOUNT – This is the first ACCOUNT subtype which stores the savings interest offered to the account

   2.2) CHECKING_ACCOUNT – This is the second ACCOUNT subtype which stores the withdrawal limit for the account

3) DEBIT_CARD – Contains the debit card details for the checking accounts uniquely identified by the *DEBITCARDNUMBER*

4) ADDRESS – Contains the address of customers and merchants of the bank uniquely identified by the *ADDRESSID*

5) LOGIN – Contains the login details for the customers of the bank uniquely identified by the *LOGINID*

6) MERCHANT – Contains all merchant information uniquely identified by *MERCHANTID*

7) TRANSACTION – Contains all the transactions between a customer and merchant uniquely identified by the *TRANSATIONID*

8) TRANSACTION_STATUS – Contains the status of all transactions uniquely identified by the *TRANSACTIONSTATUSID*

9) TRANSACTION_STATUS_TYPE – Contains the type of status of transactions to be used in the TRANSACTION_STATUS table uniquely identified by the *TRANSACTIONSTATUSTYPEID*

Note: In creating this database, certain assumptions were made regarding the structure and content of the tables, all of which are clearly mentioned in the functionalities

Relationship between Entities

CUSTOMER	⇄	ACCOUNT
CUSTOMER	⇄	LOGIN
ADDRESS	⇄	CUSTOMER
ADDRESS	⇄	MERCHANT
ACCOUNT	→	CHECKING_ACCOUNT
ACCOUNT	→	SAVING_ACCOUNT
CHECKING_ACCOUNT	⇄	DEBIT_CARD
CHECKING_ACCOUNT	⇄	TRANSACTION
TRANSACTION	⇄	MERCHANT
TRANSACTION	⇄	TRANSACTION_STATUS
TRANSACTION_STATUS_TYPE	⇄	TRANSACTION_STATUS

Cardinalities of Relationship amongst Entities

CUSTOMER (mandatory one)	⇄	ACCOUNT (mandatory many)
CUSTOMER (mandatory one)	⇄	LOGIN (optional one)
ADDRESS (mandatory one)	⇄	CUSTOMER (mandatory many)
ADDRESS (mandatory one)	⇄	MERCHANT (mandatory many)
ACCOUNT (total disjoint)	→	CHECKING_ACCOUNT
ACCOUNT (total disjoint)	→	SAVING_ACCOUNT
CHECKING_ACCOUNT (mandatory one)	⇄	DEBIT_CARD (optional mandatory)
CHECKING_ACCOUNT (mandatory one)	⇄	TRANSACTION (optional mandatory)
TRANSACTION (optional mandatory)	⇄	MERCHANT (mandatory one)
TRANSACTION (mandatory one)	⇄	TRANSACTION_STATUS (mandatory many)
TRANSACTION_STATUS_TYPE (mandatory one)	⇄	TRANSACTION_STATUS (mandatory many)

Attributes of all Entities

The following are essential attributes that have been identified from the business rules. They have been grouped based on domain knowledge and may be further refined as the project progresses.

- CUSTOMERID
- CUSTOMERNAME
- CUSTOMERPHONE
- CUSTOMEREMAIL
- ADDRESSID

- ACCOUNTID
- ACCOUNTNAME
- CUSTOMERID
- ACCOUNTBALANCE
- ACCOUNTOPENINGDATE
- ROUTINGNUMBER

- INTERESTRATE
- ACCOUNTID

- ATMWITHDRAWALCAP
- ACCOUNTID

- DEBITCARDNUMBER
- DEBITCARDEXPIRYMONTH
- DEBITCARDEXPIRYYEAR
- DEBITCARDPIN
- DEBITCARDACTIVEFLAG
- ACCOUNTID

- ADDRESSID
- STREET1
- STREET2
- CITY
- STATE
- COUNTRY
- ZIP

- LOGINID
- USERNAME
- PASSWORD
- CUSTOMERID

- MERCHANTID
- MERCHANTNAME
- MERCHANTPHONE
- MERCHANTEMAIL
- ADDRESSID

- TRANSACTIONID
- TRANSACTIONAMOUNT
- TRASNACTIONDATE
- ACCOUNTID
- MERCHANTID

- TRANSACTIONSTATUSID
- TRANSDATE
- TRANSACTIONID
- TRANSACTIONSTATUSTYPEID

- TRANSACTIONSTATUSTYPEID
- TRANSACTIONSTATUS

Entity Relationship Diagram

To design the ER Diagram, I began by segregating each entity and its attributes, and after several iterations, I arrived at the outcome. Please find below the Entities and the ER-Diagram for the overall use case.

Entities: -

TRANSACTION_STATUS		
FK	TRANSACTIONID	VARCHAR(50)
FK	TRANSACTION STATUSTYPEID	INT
	TRANSDATE	TIMESTAMP
PK	TRANSACTION STATUSID	INT

ADDRESS		
PK	ADDRESS ID	INT
	STREET1	VARCHAR(50)
	STREET2	VARCHAR(50)
	CITY	VARCHAR(50)
	STATE	VARCHAR(50)
	COUNTRY	VARCHAR(50)
	ZIP	VARCHAR(5)

CUSTOMER		
PK	CUSTOMERID	INT
	CUSTOMERNAME	VARCHAR(50)
	CUSTOMERPHONE	VARCHAR(10)
	CUSTOMEREMAIL	VARCHAR(50)
FK	ADDRESSID	INT

TRANSACTION		
PK	TRANSACTIONID	VARCHAR(50)
	TRANSACTION AMOUNT	FLOAT
	TRANSACTIONDATE	TIMESTAMP
FK	ACCOUNTID	INT
FK	MERCHANTID	INT

LOGIN		
PK	LOGINID	INT
	USERNAME	VARCHAR(50)
	PASSWORD	VARCHAR(50)
FK	CUSTOMERID	INT

ACCOUNT		
PK	ACCOUNTID	INT
	ACCOUNTNAME	VARCHAR(50)
FK	CUSTOMERID	INT
	ACCOUNTBALANCE	FLOAT
	ACCOUNTOPENING DATE	TIMESTAMP
	ROUTINGNUMBER	VARCHAR(20)

TRANSACTION_STATUS_TYPE		
PK	TRANSACTION STATUSTYPEID	INT
	TRANSACTION STATUS	VARCHAR(50)

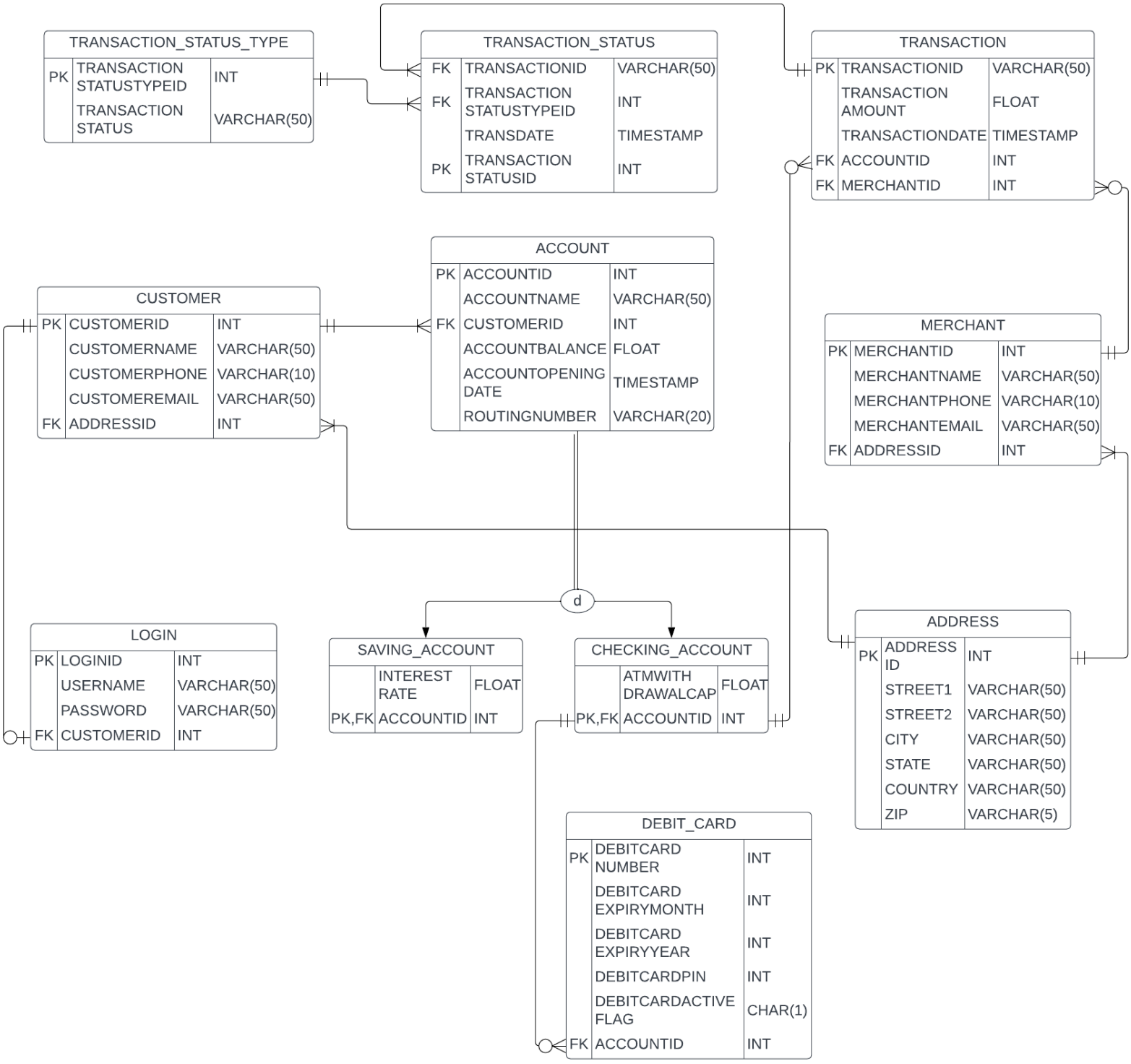
SAVING_ACCOUNT		
	INTEREST RATE	FLOAT
PK,FK	ACCOUNTID	INT

DEBIT_CARD		
PK	DEBITCARD NUMBER	INT
	DEBITCARD EXPIRYMONTH	INT
	DEBITCARD EXPIRYYEAR	INT
	DEBITCARDPIN	INT
	DEBITCARDACTIVE FLAG	CHAR(1)
FK	ACCOUNTID	INT

MERCHANT		
PK	MERCHANTID	INT
	MERCHANTNAME	VARCHAR(50)
	MERCHANTPHONE	VARCHAR(10)
	MERCHANTEMAIL	VARCHAR(50)
FK	ADDRESSID	INT

CHECKING_ACCOUNT		
	ATMWITH DRAWALCAP	FLOAT
PK,FK	ACCOUNTID	INT

ER-Diagram: -

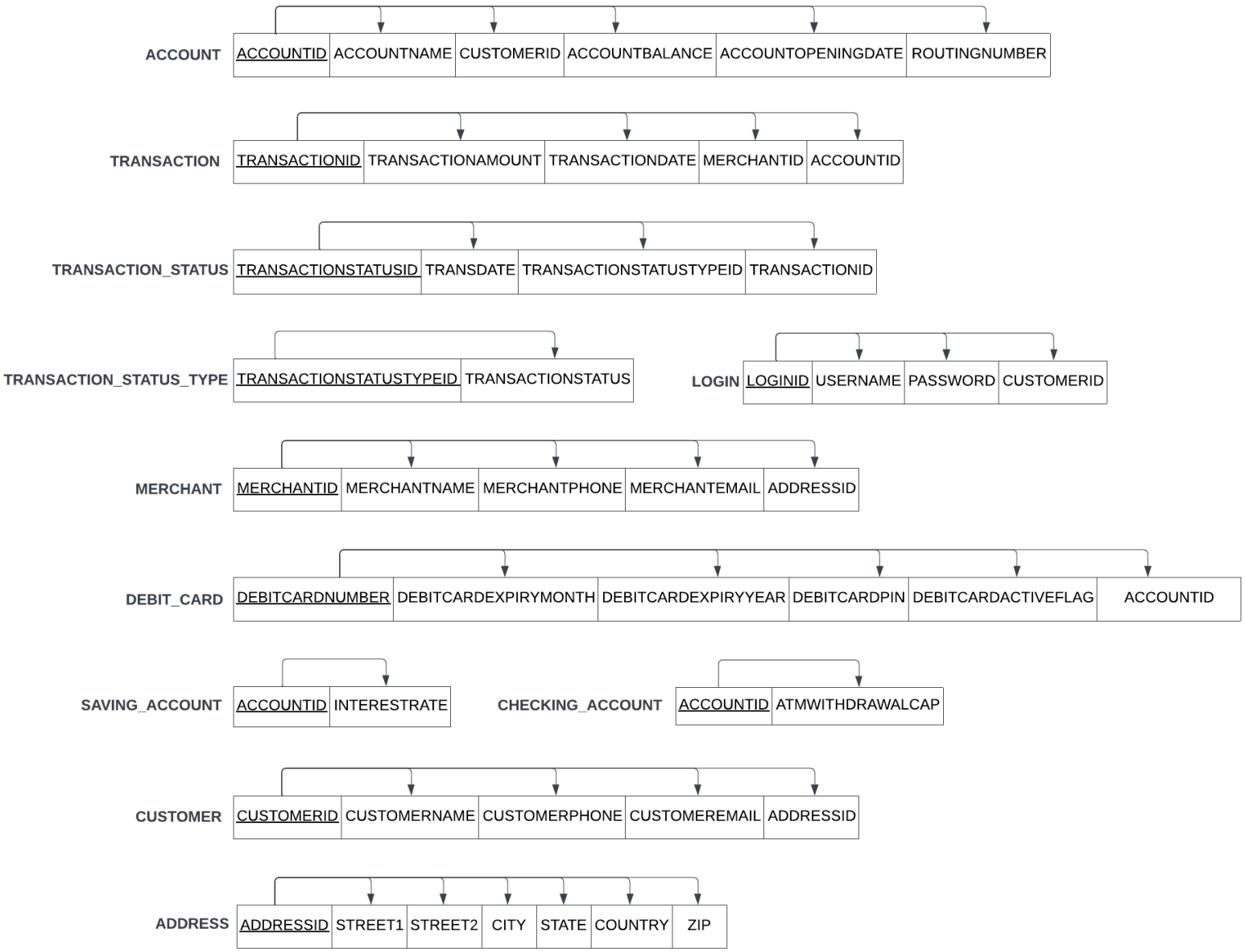

TRANSACTION_STATUS_TYPE
PK TRANSACTION STATUSTYPEID INT
TRANSACTION STATUS VARCHAR(50)
TRANSACTION_STATUS
FK TRANSACTIONID VARCHAR(50)
FK TRANSACTION STATUSTYPEID INT
TRANSDATE TIMESTAMP
PK TRANSACTION STATUSID INT
TRANSACTION
PK TRANSACTIONID VARCHAR(50)
TRANSACTION AMOUNT FLOAT
TRANSACTIONDATE TIMESTAMP
FK ACCOUNTID INT
FK MERCHANTID INT
ACCOUNT
PK ACCOUNTID INT
ACCOUNTNAME VARCHAR(50)
FK CUSTOMERID INT
ACCOUNTBALANCE FLOAT
ACCOUNTOPENING DATE TIMESTAMP
ROUTINGNUMBER VARCHAR(20)
CUSTOMER
PK CUSTOMERID INT
CUSTOMERNAME VARCHAR(50)
CUSTOMERPHONE VARCHAR(10)
CUSTOMEREMAIL VARCHAR(50)
FK ADDRESSID INT
MERCHANT
PK MERCHANTID INT
MERCHANTNAME VARCHAR(50)
MERCHANTPHONE VARCHAR(10)
MERCHANTEMAIL VARCHAR(50)
FK ADDRESSID INT
d
LOGIN
PK LOGINID INT
USERNAME VARCHAR(50)
PASSWORD VARCHAR(50)
FK CUSTOMERID INT
SAVING_ACCOUNT
INTEREST RATE FLOAT
PK,FK ACCOUNTID INT
CHECKING_ACCOUNT
ATMWITH DRAWALCAP FLOAT
PK,FK ACCOUNTID INT
ADDRESS
PK ADDRESS ID INT
STREET1 VARCHAR(50)
STREET2 VARCHAR(50)
CITY VARCHAR(50)
STATE VARCHAR(50)
COUNTRY VARCHAR(50)
ZIP VARCHAR(5)
DEBIT_CARD
PK DEBITCARD NUMBER INT
DEBITCARD EXPIRYMONTH INT
DEBITCARD EXPIRYYEAR INT
DEBITCARDPIN INT
DEBITCARDACTIVE FLAG CHAR(1)
FK ACCOUNTID INT

ER-Diagram to Relational Schema

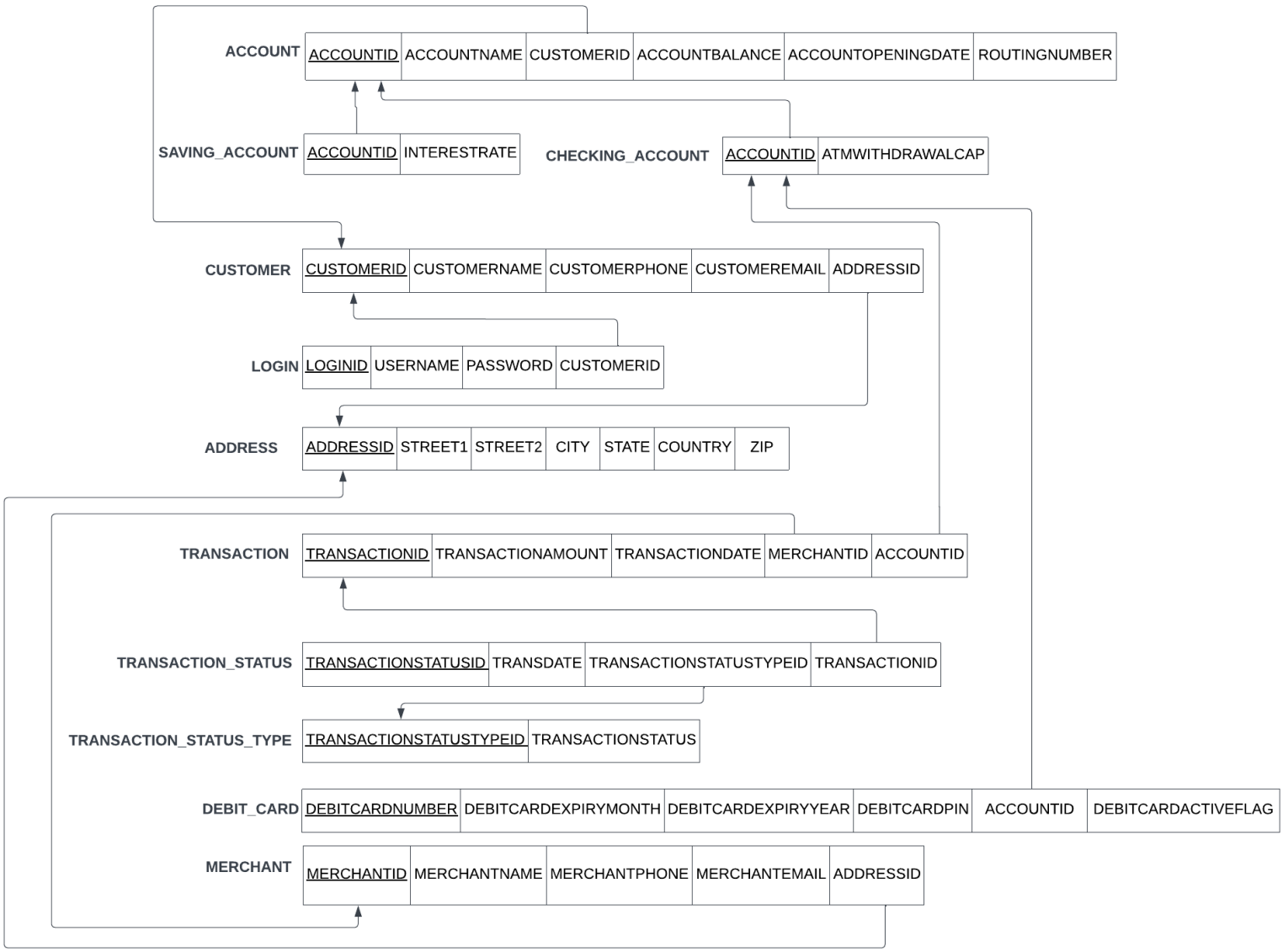
Normalization: 1st Normal Form (1NF)

PK	CUSTOMERID
PK	ACCOUNTID
PK	LOGINID
PK	ADDRESSID
PK	DEBITCARDNUMBER
PK	MERCHANTID
PK	TRANSACTIONID
PK	TRANSACTIONSTATUSID
PK	TRANSACTIONSTATUSTYPEID
	CUSTOMERNAME
	CUSTOMERPHONE
	CUSTOMEREMAIL
	ACCOUNTNAME
	ACCOUNTBALANCE
	ACCOUNTOPENINGDATE
	ROUTINGNUMBER
	INTERESTRATE
	ATMWITHDRAWALCAP
	DEBITCARDEXPIRYMONTH
	DEBITCARDEXPIRYYEAR
	DEBITCARDPIN
	DEBITCARDACTIVEFLAG
	STREET1
	STREET2
	CITY
	STATE
	COUNTRY
	ZIP
	USERNAME
	PASSWORD
	MERCHANTNAME
	MERCHANTPHONE
	MERCHANTEMAIL
	TRANSACTIONAMOUNT
	TRASNACTIONDATE
	TRANSDATE
	TRANSACTIONSTATUS

Normalization: 2nd Normal Form (2NF)



Normalization: 3rd Normal Form (3NF)



Summary Table for Each Entity

Customer Table
CUSTOMER (CUSTOMERID, CUSTOMERNAME, CUSTOMERPHONE, CUSTOMEREMAIL, ADDRESSID)
Datatype: INT, VARCHAR(50), VARCHAR(10), VARCHAR(50), INT
Additional Details: All fields required, ADDRESSID is FK.

Account Table
ACCOUNT (ACCOUNTID, ACCOUNTNAME, ACCOUNTBALANCE, CUSTOMERID, ACCOUNTBALANCE, ACCOUNTOPENINGDATE, ROUTINGNUMBER)
Datatype: INT, VARCHAR(50), INT, FLOAT, TIMESTAMP, VARCHAR(20)
Additional Details: All fields required, CUSTOMERID is FK.

Saving Account Table
SAVING_ACCOUNT (INTERESTRATE, ACCOUNTID)
Datatype: FLOAT, INT
Additional Details: All fields required, ACCOUNTID is FK.

Checking Account Table
CHECKING_ACCOUNT (ATMWITHDRAWALCAP, ACCOUNTID)
Datatype: FLOAT, INT
Additional Details: All fields required, ACCOUNTID is FK.

Debit Card Table
DEBIT_CARD (DEBITCARDNUMBER, DEBITCARDEXPIRYMONTH, DEBITCARDEXPIRYYEAR, DEBITCARDPIN, DEBITCARDACTIVEFLAG, ACCOUNTID)
Datatype: INT, INT, INT, INT, CHAR(1), INT
Additional Details: All fields required, ACCOUNTID is FK.

Address Table
ADDRESS (ADDRESSID, STREET1, STREET2, CITY, STATE, COUNTRY, ZIP)
Datatype: INT, VARCHAR(50), VARCHAR(50), VARCHAR(50), VARCHAR(50), VARCHAR(50), VARCHAR(5)
Additional Details: All fields except STREET2 required.

Login Table
LOGIN (LOGINID, USERNAME, PASSWORD, CUSTOMERID)
Datatype: INT, VARCHAR(50), VARCHAR(50), INT
Additional Details: All fields required, CUSTOMERID is FK.

Merchant Table
MERCHANT (MERCHANTID, MERCHANTNAME, MERCHANTPHONE, MERCHANTEMAIL, ADDRESSID)
Datatype: INT, VARCHAR(50), VARCHAR(10), VARCHAR(50), INT
Additional Details: All fields required, ADDRESSID is FK.

Transaction Table
TRANSACTION (TRANSACTIONID, TRANSACTIONAMOUNT, TRANSACTIONDATE, ACCOUNTID, MERCHANTID)
Datatype: VARCHAR(50), FLOAT, TIMESTAMP, INT, INT
Additional Details: All fields required, ACCOUNTID and MERCHANTID are FKs.

Transaction Status Type Table
TRANSACTION_STATUS_TYPE (TRANSACTIONSTATUSTYPEID, TRANSACTIONSTATUS)
Datatype: INT, VARCHAR(50)
Additional Details: All fields required.

Transaction Status Table
TRANSACTION_STATUS (TRANSACTIONID, TRANSACTIONSTATUSTYPEID, TRANSDATE, TRANSACTIONSTATUSID)
Datatype: VARCHAR(50), INT, TIMESTAMP, INT
Additional Details: All fields required, TRANSACTIONID and TRANSACTIONSTATUSTYPEID are FKs.

Creation of Tables

Customer Table
CUSTOMER (CUSTOMERID, CUSTOMERNAME, CUSTOMERPHONE, CUSTOMEREMAIL, ADDRESSID)
Datatype: INT, VARCHAR(50), VARCHAR(10), VARCHAR(50), INT
Additional Details: All fields required, ADDRESSID is FK.

```
CREATE TABLE Customer (
  CustomerID INT NOT NULL,
  CustomerName VARCHAR(50) NOT NULL,
  CustomerPhone VARCHAR(50) NOT NULL,
  CustomerEmail VARCHAR(50) NOT NULL,
  AddressID INT NOT NULL,
  CONSTRAINT Customer_PK
    PRIMARY KEY (CustomerID),
  CONSTRAINT Customer_FK
    FOREIGN KEY (AddressID)
      REFERENCES Address(AddressID)
);
```

Account Table
ACCOUNT (ACCOUNTID, ACCOUNTNAME, ACCOUNTBALANCE, CUSTOMERID, ACCOUNTBALANCE, ACCOUNTOPENINGDATE, ROUTINGNUMBER)
Datatype: INT, VARCHAR(50), INT, FLOAT, TIMESTAMP, VARCHAR(20)
Additional Details: All fields required, CUSTOMERID is FK.

```
CREATE TABLE Account(
  AccountID INT NOT NULL,
  AccountName VARCHAR(50) NOT NULL,
  CustomerID INT NOT NULL,
  AccountBalance FLOAT NOT NULL,
  AccountOpeningDate TIMESTAMP NOT NULL,
  RoutingNo VARCHAR(20) NOT NULL,
  CONSTRAINT Account_PK
    PRIMARY KEY (AccountID),
  CONSTRAINT Account_FK
    FOREIGN KEY (CustomerID)
       REFERENCES Customer(CustomerID)
);
```

Saving Account Table

SAVING_ACCOUNT (INTERESTRATE, ACCOUNTID)
Datatype: FLOAT, INT
Additional Details: All fields required, ACCOUNTID is FK.

```
CREATE TABLE Saving_Account (
  AccountID INT NOT NULL,
  INTerestRate FLOAT NOT NULL,
  CONSTRAINT Saving_Account_PK
    PRIMARY KEY (AccountID),
  CONSTRAINT Saving_Account_FK
    FOREIGN KEY (AccountID)
       REFERENCES Account(AccountID)
);
```

Checking Account Table

CHECKING_ACCOUNT (ATMWITHDRAWALCAP, ACCOUNTID)
Datatype: FLOAT, INT
Additional Details: All fields required, ACCOUNTID is FK.

```
CREATE TABLE Checking_Account(
  AccountID INT NOT NULL,
  ATMWithdrawalCap FLOAT NOT NULL,
  CONSTRAINT Checking_Account_PK
    PRIMARY KEY (AccountID),
  CONSTRAINT Checking_Account_FK
    FOREIGN KEY (AccountID)
       REFERENCES Account(AccountID)
);
```

Debit Card Table

DEBIT_CARD (DEBITCARDNUMBER, DEBITCARDEXPIRYMONTH, DEBITCARDEXPIRYYEAR, DEBITCARDPIN, DEBITCARDACTIVEFLAG, ACCOUNTID)
Datatype: INT, INT, INT, INT, CHAR(1), INT
Additional Details: All fields required, ACCOUNTID is FK.

```
CREATE TABLE Debit_Card (
  DebitCardNumber INT NOT NULL,
  DebitCardExpiryMonth INT NOT NULL,
  DebitCardExpiryYear INT NOT NULL,
  DebitCardPin INT NOT NULL,
  DebitCardActiveFlag CHAR(1) NOT NULL,
  AccountID INT NOT NULL,
  CONSTRAINT Debit_Card_PK
    PRIMARY KEY (DebitCardNumber),
  CONSTRAINT Debit_Card_FK
    FOREIGN KEY (AccountID)
     REFERENCES Checking_Account(AccountID)
);
```

Address Table
ADDRESS (ADDRESSID, STREET1, STREET2, CITY, STATE, COUNTRY, ZIP)
Datatype: INT, VARCHAR(50), VARCHAR(50), VARCHAR(50), VARCHAR(50), VARCHAR(50), VARCHAR(5)
Additional Details: All fields except STREET2 required.

```
CREATE TABLE Address
(
  AddressID INT NOT NULL,
  Street1 VARCHAR(50) NOT NULL,
  Street2 VARCHAR(50),
  City VARCHAR(50) NOT NULL,
  State VARCHAR(50) NOT NULL,
  Country VARCHAR(50) NOT NULL,
  ZIP VARCHAR(5) NOT NULL,
  CONSTRAINT Address_PK
    PRIMARY KEY (AddressID)
);
```

Login Table
LOGIN (LOGINID, USERNAME, PASSWORD, CUSTOMERID)
Datatype: INT, VARCHAR(50), VARCHAR(50), INT
Additional Details: All fields required, CUSTOMERID is FK.

```
CREATE TABLE Login (
  LoginID INT NOT NULL,
  Username VARCHAR(50) NOT NULL,
  Password VARCHAR(50) NOT NULL,
  CustomerID INT NOT NULL,
  CONSTRAINT Login_PK
  PRIMARY KEY (LoginID),
  CONSTRAINT Login_FK
    FOREIGN KEY(CustomerID)
     REFERENCES Customer(CustomerID)
);
```

Merchant Table
MERCHANT (MERCHANTID, MERCHANTNAME, MERCHANTPHONE, MERCHANTEMAIL, ADDRESSID)
Datatype: INT, VARCHAR(50), VARCHAR(10), VARCHAR(50), INT
Additional Details: All fields required, ADDRESSID is FK.

```
CREATE TABLE Merchant (
  MerchantID INT NOT NULL,
  MerchantName VARCHAR(50) NOT NULL,
  MerchantPhone VARCHAR(10) NOT NULL,
  MerchantEmail VARCHAR(50) NOT NULL,
  AddressID INT NOT NULL,
  CONSTRAINT Merchant_PK
    PRIMARY KEY (MerchantID),
  CONSTRAINT Merchant_FK
    FOREIGN KEY (AddressID)
      REFERENCES Address(AddressID)
);
```

Transaction Table

TRANSACTION (TRANSACTIONID, TRANSACTIONAMOUNT, TRANSACTIONDATE, ACCOUNTID, MERCHANTID)
Datatype: VARCHAR(50), FLOAT, TIMESTAMP, INT, INT
Additional Details: All fields required, ACCOUNTID and MERCHANTID are FKs.

```
CREATE TABLE Transaction (
  TransactionID VARCHAR(50) NOT NULL,
  TransactionAmount FLOAT NOT NULL,
  TransactionDate TIMESTAMP NOT NULL,
  AccountID INT NOT NULL,
  MerchantID INT NOT NULL,
  CONSTRAINT Transaction_PK
    PRIMARY KEY (TransactionID),
  CONSTRAINT Transaction_FK1
    FOREIGN KEY (AccountID)
      REFERENCES Checking_Account(AccountID),
  CONSTRAINT Transaction_FK2
    FOREIGN KEY (MerchantID)
      REFERENCES Merchant(MerchantID)
);
```

Transaction Status Type Table

TRANSACTION_STATUS_TYPE (TRANSACTIONSTATUSTYPEID, TRANSACTIONSTATUS)
Datatype: INT, VARCHAR(50)
Additional Details: All fields required.

```
CREATE TABLE Transaction_Status_Type (
  TransactionStatusTypeID INT NOT NULL,
  TransactionStatus VARCHAR(50) NOT NULL,
  CONSTRAINT Transaction_Status_Type_PK
    PRIMARY KEY (TransactionStatusTypeID)
);
```

Transaction Status Table

TRANSACTION_STATUS (TRANSACTIONID, TRANSACTIONSTATUSTYPEID, TRANSDATE, TRANSACTIONSTATUSID)
Datatype: VARCHAR(50), INT, TIMESTAMP, INT
Additional Details: All fields required, TRANSACTIONID and TRANSACTIONSTATUSTYPEID are FKs.

```
CREATE TABLE Transaction_Status (
  TransactionStatusID INT NOT NULL,
  TransactionID VARCHAR(50) NOT NULL,
  TransactionStatusTypeID INT NOT NULL,
  TransDate TIMESTAMP NOT NULL,
  CONSTRAINT Transaction_Status_PK
    PRIMARY KEY (TransactionStatusID),
  CONSTRAINT Transaction_Status_FK1
    FOREIGN KEY (TransactionStatusTypeID)
      REFERENCES Transaction_Status_Type(TransactionStatusTypeID),
  CONSTRAINT Transaction_Status_FK2
    FOREIGN KEY (TransactionID)
      REFERENCES Transaction(TransactionID)
);
```

Insertion of Data in Tables

Customer Table: -

```
INSERT INTO C##USER01.Customer (CustomerID, CustomerName, CustomerPhone, CustomerEmail, AddressID)
VALUES (10001, 'Ritvik Wadhwa', '857123456', 'wadhwa.rit@northeastern.edu', 1);

INSERT INTO C##USER01.Customer (CustomerID, CustomerName, CustomerPhone, CustomerEmail, AddressID)
VALUES (10002, 'Shahbaz Aslam', '857123654', 'aslam.sh@northeastern.edu', 2);

INSERT INTO C##USER01.Customer (CustomerID, CustomerName, CustomerPhone, CustomerEmail, AddressID)
VALUES (10003, 'Shivani Bajaj', '857123321', 'bajaj.shi@northeastern.edu', 3);
```

Account Table: -

```
INSERT INTO C##USER01.Account (AccountID, AccountName, CustomerID, AccountBalance, AccountOpeningDate,
RoutingNo)
VALUES (100011, 'Ritvik Wadhwa', 10001, 9843579.69, timestamp '2007-05-11 14:34:35', 12345678);

INSERT INTO C##USER01.Account (AccountID, AccountName, CustomerID, AccountBalance, AccountOpeningDate,
RoutingNo)
VALUES (100021, 'Shahbaz Aslam', 10002, 539.69, timestamp '2013-09-29 10:12:58', 87654321);

INSERT INTO C##USER01.Account (AccountID, AccountName, CustomerID, AccountBalance, AccountOpeningDate,
RoutingNo)
VALUES (100022, 'Moeez Aslam', 10002, 5343259.69, timestamp '2014-02-14 12:30:12', 12344321);

INSERT INTO C##USER01.Account (AccountID, AccountName, CustomerID, AccountBalance, AccountOpeningDate,
RoutingNo)
VALUES (100031, 'Shivani Bajaj', 10003, 23945.69, timestamp '2011-04-15 13:35:02', 56788765);
```

Saving Table: -

```
INSERT INTO C##USER01.Saving_Account (InterestRate, AccountID)
VALUES (2.5, 100011);

INSERT INTO C##USER01.Saving_Account (InterestRate, AccountID)
VALUES (1.05, 100022);

INSERT INTO C##USER01.Saving_Account (InterestRate, AccountID)
VALUES (1.65, 100031);
```

Checking Table: -

```
INSERT INTO C##USER01.Checking_Account (ATMWithdrawalCap, AccountID)
VALUES (20000, 100011);

INSERT INTO C##USER01.Checking_Account (ATMWithdrawalCap, AccountID)
VALUES (1000, 100021);

INSERT INTO C##USER01.Checking_Account (ATMWithdrawalCap, AccountID)
VALUES (1500, 100022);

INSERT INTO C##USER01.Checking_Account (ATMWithdrawalCap, AccountID)
VALUES (2000, 100031);
```

Debit Card Table: -

INSERT INTO C##USER01.Debit_Card (DebitCardNumber, DebitCardExpiryMonth, DebitCardExpiryYear, DebitCardPin, DebitCardActiveFlag, AccountID)
VALUES (1234567887654321, 11, 2028, 1769, 'Y', 100011);

INSERT INTO C##USER01.Debit_Card (DebitCardNumber, DebitCardExpiryMonth, DebitCardExpiryYear, DebitCardPin, DebitCardActiveFlag, AccountID)
VALUES (4321123487655678, 02, 2026, 1111, 'Y', 100021);

INSERT INTO C##USER01.Debit_Card (DebitCardNumber, DebitCardExpiryMonth, DebitCardExpiryYear, DebitCardPin, DebitCardActiveFlag, AccountID)
VALUES (5678876543211234, 04, 2027, 1221, 'N', 100022);

INSERT INTO C##USER01.Debit_Card (DebitCardNumber, DebitCardExpiryMonth, DebitCardExpiryYear, DebitCardPin, DebitCardActiveFlag, AccountID)
VALUES (8765567812344321, 07, 2025, 1122, 'Y', 100031);

Address Table: -

INSERT INTO C##USER01.Address (AddressID, Street1, Street2, City, State, Country, ZIP)
VALUES ('1', '360 Huntington Ave', 'Dodge Hall', 'Boston', 'MA', 'United States', '02115');

INSERT INTO C##USER01.Address (AddressID, Street1, Street2, City, State, Country, ZIP)
VALUES ('2', '77 Massachusetts Ave', 'Matt Hall', 'Cambridge', 'MA', 'United States', '02139');

INSERT INTO C##USER01.Address (AddressID, Street1, Street2, City, State, Country, ZIP)
VALUES ('3', '220 Pawtucket St', 'Sean Hall', 'Lowell', 'MA', 'United States', '01854');

INSERT INTO C##USER01.Address (AddressID, Street1, Street2, City, State, Country, ZIP)
VALUES ('4', '2201, Tremont St', 'JVUE Apartments', 'Boston', 'MA', 'United States', '02120');

INSERT INTO C##USER01.Address (AddressID, Street1, Street2, City, State, Country, ZIP)
VALUES ('5', '1575, Lacoste St', '', 'Natick', 'MA', 'United States', '01760');

Login Table: -

INSERT INTO C##USER01.Login (LoginID, Username, Password, CustomerID)
VALUES (99001, 'moneyman', 'iamrich', 10001);

INSERT INTO C##USER01.Login (LoginID, Username, Password, CustomerID)
VALUES (99002, 'richierich', 'iamaliar', 10002);

INSERT INTO C##USER01.Login (LoginID, Username, Password, CustomerID)
VALUES (99003, 'selfie_girl', 'instaqueen', 10003);

Merchant Table: -

```
INSERT INTO C##USER01.Merchant (MerchantID, MerchantName, MerchantPhone, MerchantEmail, AddressID)
VALUES (50001, 'Aslam Bros', 8578578578, 'aslam_bros@gmail.com', 4);

INSERT INTO C##USER01.Merchant (MerchantID, MerchantName, MerchantPhone, MerchantEmail, AddressID)
VALUES (50002, 'Bajaj Cakes', 6173213213, 'bajaj_tasty@hotmail.com', 5);

INSERT INTO C##USER01.Merchant (MerchantID, MerchantName, MerchantPhone, MerchantEmail, AddressID)
VALUES (50003, 'Wadhwa Bikes', 9999999999, '2wheelsmovethesoul@gmail.com', 4);
```

Transaction Table: -

```
INSERT INTO C##USER01.Transaction (TransactionID, TransactionAmount, TransactionDate, AccountID,
MerchantID)
VALUES ('T001', 959.58, timestamp '2023-03-12 20:37:55', 100011, 50001);

INSERT INTO C##USER01.Transaction (TransactionID, TransactionAmount, TransactionDate, AccountID,
MerchantID)
VALUES ('T002', 200, timestamp '2023-03-10 10:21:22', 100021, 50003);

INSERT INTO C##USER01.Transaction (TransactionID, TransactionAmount, TransactionDate, AccountID,
MerchantID)
VALUES ('T003', 55, timestamp '2023-03-13 23:55:59', 100022, 50002);
```

Transaction Status Type Table: -

```
INSERT INTO C##USER01.Transaction_Status_Type (TransactionStatusTypeID, TransactionStatus)
VALUES (1, 'Initiated');

INSERT INTO C##USER01.Transaction_Status_Type (TransactionStatusTypeID, TransactionStatus)
VALUES (2, 'Pending');

INSERT INTO C##USER01.Transaction_Status_Type (TransactionStatusTypeID, TransactionStatus)
VALUES (3, 'On Hold');

INSERT INTO C##USER01.Transaction_Status_Type (TransactionStatusTypeID, TransactionStatus)
VALUES (4, 'Successful');

INSERT INTO C##USER01.Transaction_Status_Type (TransactionStatusTypeID, TransactionStatus)
VALUES (5, 'Failed');
```

Transaction Status Table: -

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40001, 'T001', 1, timestamp '2023-03-12 08:37:50');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40002, 'T001', 4, timestamp '2023-03-12 08:37:55');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40003, 'T002', 1, timestamp '2023-03-10 10:20:23');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40004, 'T002', 2, timestamp '2023-03-10 10:20:27');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40005, 'T002', 3, timestamp '2023-03-10 10:20:39');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40006, 'T002', 4, timestamp '2023-03-10 10:21:22');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40007, 'T003', 1, timestamp '2023-03-13 11:51:54');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40008, 'T003', 2, timestamp '2023-03-13 11:52:03');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40009, 'T003', 3, timestamp '2023-03-13 11:52:29');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40010, 'T003', 5, timestamp '2023-03-13 11:53:59');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40011, 'T003', 1, timestamp '2023-03-13 11:54:15');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40012, 'T003', 2, timestamp '2023-03-13 11:54:23');

INSERT INTO C##USER01.Transaction_Status (TransactionStatusID, TransactionID, TransactionStatusTypeID, TransDate)
VALUES (40013, 'T003', 4, timestamp '2023-03-13 11:55:59');

Data Loaded in Database

```
306  SELECT * FROM Customer;
```

CUSTOMERID	CUSTOMERNAME	CUSTOMERPHONE	CUSTOMEREMAIL	ADDRESSID
10001	Ritvik Wadhwa	857123456	wadhwa.rit@northeastern.edu	1
10002	Shahbaz Aslam	857123654	aslam.sh@northeastern.edu	2
10003	Shivani Bajaj	857123321	bajaj.shi@northeastern.edu	3

```
307  SELECT * FROM Account;
```

ACCOUNTID	ACCOUNTNAME	CUSTOMERID	ACCOUNTBALANCE	ACCOUNTOPENINGDATE	ROUTINGNO
100011	Ritvik Wadhwa	10001	9843579.69	11-MAY-07 02.34.35.000000000 PM	12345678
100022	Moeez Aslam	10002	5343259.69	14-FEB-14 12.30.12.000000000 PM	12344321
100031	Shivani Bajaj	10003	23945.69	15-APR-11 01.35.02.000000000 PM	56788765
100021	Shahbaz Aslam	10002	539.69	29-SEP-13 10.12.58.000000000 AM	87654321

```
308  SELECT * FROM Saving_Account;
```

ACCOUNTID	INTERESTRATE
100011	2.5
100022	1.05
100031	1.65

```
309  SELECT * FROM Checking_Account;
```

ACCOUNTID	ATMWITHDRAWALCAP
100011	20000
100021	1000
100022	1500
100031	2000

```
310 SELECT * FROM Debit_Card;
```

DEBITCARDNUMBER	DEBITCARDEXPIRYMONTH	DEBITCARDEXPIRYYEAR	DEBITCARDPIN	DEBITCARDACTIVEFLAG	ACCOUNTID
1234567887654321	11	2028	1769	Y	100011
4321123487655678	2	2026	1111	Y	100021
5678876543211234	4	2027	1221	N	100022
8765567812344321	7	2025	1122	Y	100031

```
311 SELECT * FROM Address;
```

ADDRESSID	STREET1	STREET2	CITY	STATE	COUNTRY	ZIP
1	360 Huntington Ave	Dodge Hall	Boston	MA	United States	02115
2	77 Massachusetts Ave	Matt Hall	Cambridge	MA	United States	02139
3	220 Pawtucket St	Sean Hall	Lowell	MA	United States	01854
4	2201, Tremont St	JVUE Apartments	Boston	MA	United States	02120
5	1575, Lacoste St	(null)	Natick	MA	United States	01760

```
312 SELECT * FROM Login;
```

LOGINID	USERNAME	PASSWORD	CUSTOMERID
99001	moneyman	iamrich	10001
99002	richierich	iamaliar	10002
99003	selfie_girl	instaqueen	10003

```
313 SELECT * FROM Merchant;
```

MERCHANTID	MERCHANTNAME	MERCHANTPHONE	MERCHANTEMAIL	ADDRESSID
50001	Aslam Bros	8578578578	aslam_bros@gmail.com	4
50002	Bajaj Cakes	6173213213	bajaj_tasty@hotmail.com	5
50003	Wadhwa Bikes	9999999999	2wheelsmovethesoul@gmail.com	4

```
314 SELECT * FROM Transaction;
```

TRANSACTIONID	TRANSACTIONAMOUNT	TRANSACTIONDATE	ACCOUNTID	MERCHANTID
T001	959.58	12-MAR-23 08.37.55.000000000 PM	100011	50001
T002	200	10-MAR-23 10.21.22.000000000 AM	100021	50003
T003	55	13-MAR-23 11.55.59.000000000 PM	100022	50002

```
315 SELECT * FROM Transaction_Status_Type;
```

TRANSACTIONSTATUSTYPEID	TRANSACTIONSTATUS
1	Initiated
2	Pending
3	On Hold
4	Successful
5	Failed

```
316 SELECT * FROM Transaction_Status;
```

TRANSACTIONSTATUSID	TRANSACTIONID	TRANSACTIONSTATUSTYPEID	TRANSDATE
40001	T001	1	12-MAR-23 08.37.50.000000000 AM
40002	T001	4	12-MAR-23 08.37.55.000000000 AM
40003	T002	1	10-MAR-23 10.20.23.000000000 AM
40004	T002	2	10-MAR-23 10.20.27.000000000 AM
40005	T002	3	10-MAR-23 10.20.39.000000000 AM
40006	T002	4	10-MAR-23 10.21.22.000000000 AM
40007	T003	1	13-MAR-23 11.51.54.000000000 AM
40008	T003	2	13-MAR-23 11.52.03.000000000 AM
40009	T003	3	13-MAR-23 11.52.29.000000000 AM
40010	T003	5	13-MAR-23 11.53.59.000000000 AM
40011	T003	1	13-MAR-23 11.54.15.000000000 AM
40012	T003	2	13-MAR-23 11.54.23.000000000 AM
40013	T003	4	13-MAR-23 11.55.59.000000000 AM

Queries

Find names and addresses of customers residing in Cambridge, who possess multiple accounts and have experienced transaction go on hold.

Query: -

```
318 SELECT acc.CustomerID, c.CustomerName, a.Street1, a.Street2, a.City, a.State, a.Country, a.ZIP
319 FROM C##USER01.Customer c
320 JOIN C##USER01.Address a
321     ON c.AddressID = a.AddressID
322 JOIN C##USER01.Account acc
323     ON c.CustomerID = acc.CustomerID
324 JOIN C##USER01.Transaction t
325     ON t.AccountID = acc.AccountID
326 JOIN C##USER01.Transaction_Status ts
327     on t.TransactionID = ts.TransactionID
328 WHERE ts.TransactionStatusTypeID = 3
329     AND UPPER(a.city) = 'CAMBRIDGE'
330 GROUP BY acc.CustomerID, c.CustomerName, a.Street1, a.Street2, a.City, a.State, a.Country, a.ZIP
331     HAVING COUNT(acc.CustomerID) > 1;
```

Output: -

CUSTOMERID	CUSTOMERNAME	STREET1	STREET2	CITY	STATE	COUNTRY	ZIP
10002	Shahbaz Aslam	77 Massachusetts Ave	Matt Hall	Cambridge	MA	United States	02139

Retrieve the names and email addresses of both accounts and merchants, along with the date and time of transactions, for failed transactions below $100 occurring after March 10, 2023.

Query: -

```
334 SELECT acc.AccountID, acc.AccountName, c.CustomerEmail, m.MerchantName, m.MerchantEmail, ts.TransDate
335 FROM Customer c
336 JOIN Account acc
337     ON acc.CustomerID = c.CustomerID
338 JOIN Transaction t
339     ON t.AccountID = acc.AccountID
340 JOIN Transaction_Status ts
341     ON t.TransactionID = ts.TransactionID
342 JOIN Merchant m
343     ON m.MerchantID = t.MerchantID
344 WHERE ts.TransactionStatusTypeID = 5
345     AND t.TransactionAmount < 100
346     AND ts.TransDate > to_timestamp('2023-03-10', 'YYYY-MM-DD');
```

Output: -

ACCOUNTID	ACCOUNTNAME	CUSTOMEREMAIL	MERCHANTNAME	MERCHANTEMAIL	TRANSDATE
100022	Moeez Aslam	aslam.sh@northeastern.edu	Bajaj Cakes	bajaj_tasty@hotmail.com	13-MAR-23 11.53.59.000000000 AM

The bank has introduced a new policy that prohibits the use of only lowercase letters in net banking passwords. Find the names, email addresses, and physical addresses of all customers whose passwords contain only lowercase letters, so they can be informed of the new guideline and asked to change their passwords.

Query: -

```
349  SELECT c.CustomerName, c.CustomerEmail, a.Street1, a.Street2, a.City, a.State, a.Country, a.ZIP
350  FROM Customer c
351  JOIN Login l
352      ON l.CustomerID = c.CustomerID
353  JOIN Address a
354      ON a.AddressID = c.AddressID
355  WHERE l.Password = LOWER(l.Password);
```

Output: -

CUSTOMERNAME	CUSTOMEREMAIL	STREET1	STREET2	CITY	STATE	COUNTRY	ZIP
Ritvik Wadhwa	wadhwa.rit@northeastern.edu	360 Huntington Ave	Dodge Hall	Boston	MA	United States	02115
Shahbaz Aslam	aslam.sh@northeastern.edu	77 Massachusetts Ave	Matt Hall	Cambridge	MA	United States	02139
Shivani Bajaj	bajaj.shi@northeastern.edu	220 Pawtucket St	Sean Hall	Lowell	MA	United States	01854

Learning

- Importance of Proper Planning: Developing a database system requires thorough planning to ensure all functionalities and assumptions are considered, and potential issues are addressed. Careful planning helps to reduce errors and ensure that the project meets the desired outcomes.

- Collaboration is Key: Developing a database system requires collaboration and effective communication among team members to ensure that all aspects of the project are properly implemented. Proper communication helps to ensure that all team members are on the same page and that the project progresses smoothly.

- Attention to Detail: Developing a database system requires a high level of attention to detail, especially when dealing with sensitive financial information. Ensuring that all data is entered accurately and protected from unauthorized access is critical to the system's success.

- Knowledge of Regulations: Developing a database system for a bank requires a good understanding of the regulatory environment in which the bank operates. This understanding is essential to ensure that the system is compliant with all relevant regulations.

- Customer-Centric Approach: Developing a database system for a bank requires a customer-centric approach to ensure that the system meets the needs of the bank's customers. This includes features such as easy-to-use interfaces, customized reporting, and prompt transaction processing.

- Understanding of Financial Domain: Developing a database system for a bank allowed us to gain a better understanding of the financial domain. This included learning about different account types, transaction processes, and security measures employed by banks. It also exposed us to the challenges and complexities involved in managing customer accounts, balances, and deposits in a secure and compliant manner. This knowledge can be useful for future projects in the financial industry or related fields.