SWIFT BANK MANAGEMENT SYSTEM – FINAL REPORT

Group 15 – Jonathan Gabriel, Sai Pranay Reddy Rachumalla and Mohsin Arshad Qazi

Problem Statement:

People in today's fast-paced world need a quick and secure way to manage their banking activities. With the growing popularity of online banking, a comprehensive system that provides customers with a user-friendly interface to manage their banking transactions is required. A banking management system database is required to centrally store and manage customer information, account details, transaction history, and merchant information.

With a user-friendly interface, the Banking Management System Database for Swift Bank aims to provide a solution that simplifies banking activities for customers and improves their experience. The project will provide a solution that caters to the needs of customers and merchants, enhancing their banking experience, by developing a comprehensive banking management system database.

The project will concentrate on creating a database system that is dependable, scalable, and adaptable to changing requirements. The system should be able to manage large amounts of data efficiently and provide customers with real-time updates. The database will store and provide easy access to entities such as User Credentials, Customer, Account, Savings Account, Checking Account, Merchant, and Transactions.

Functionality:

The Swift Banking Management System Database seeks to provide a comprehensive online banking solution that enables customers to manage their banking activities from any location at any time. User registration and login, customer and account management, merchant management, and transaction management are all available through the system. Customers can open various types of accounts, such as Savings Accounts and Checking Accounts, and conduct transactions such as depositing, withdrawing, transferring, and paying bills. Customers can use the system to make payments to merchants, and merchants can register and manage their payment details. The system records all transactions, keeps a transaction history, and provides real-time updates to customers. The project provides a user-friendly interface that improves the customer's banking experience and ensures that all customer information is secure

Entities:

1) Customers (has a unique customer ID)
2) Merchants (has a unique merchant ID)
3) User Credential (has a unique Login ID)
4) Transactions (has a unique transaction ID, records a payment every time a customer pays a merchant)
5) Account (has a unique Account ID)
6) SavingAccount (SAccountID)
7) CheckingAccount (CAccountID)

Relationship between entities:

UserCredentials	(1:1)	Customer
Customer	(1:M)	Account
Account	(1:M)	Transactions
Transactions	(M:1)	Merchant

Cardinalities of Relationships among entities:

User credentials (mandatory one)	Customers (optional one)
Customers (mandatory many)	Accounts (mandatory one)
Account type savings checking (optional many)	Transactions (mandatory one)
Transactions (mandatory one)	Merchants (optional many)

Attributes of all entities:

• CustID	• AccountType	• TransID
• CustName	• AccountBal	• TransDate
• CustPhone	• RoutingNum	• TransAmount
• CustAddress		• TransStatus
• CustEmail	• Min_Bal	
• DateRegistered	• Interest_Rate	• Merchant_ID
		• Merchant_Name
• LoginID	• Monthly_fee	• Merchant_Phone
• Passwd	• ATM_withdrawalcap	• Merchant_email
	• DebitCardNum	• Merchant_address
• AccountID	• PIN	
• AccountName		
• DateOpened		

ER – Diagram:

Below is the final outcome of all the entities we chose along with the attribute names for the overall Use Case.

Entities:

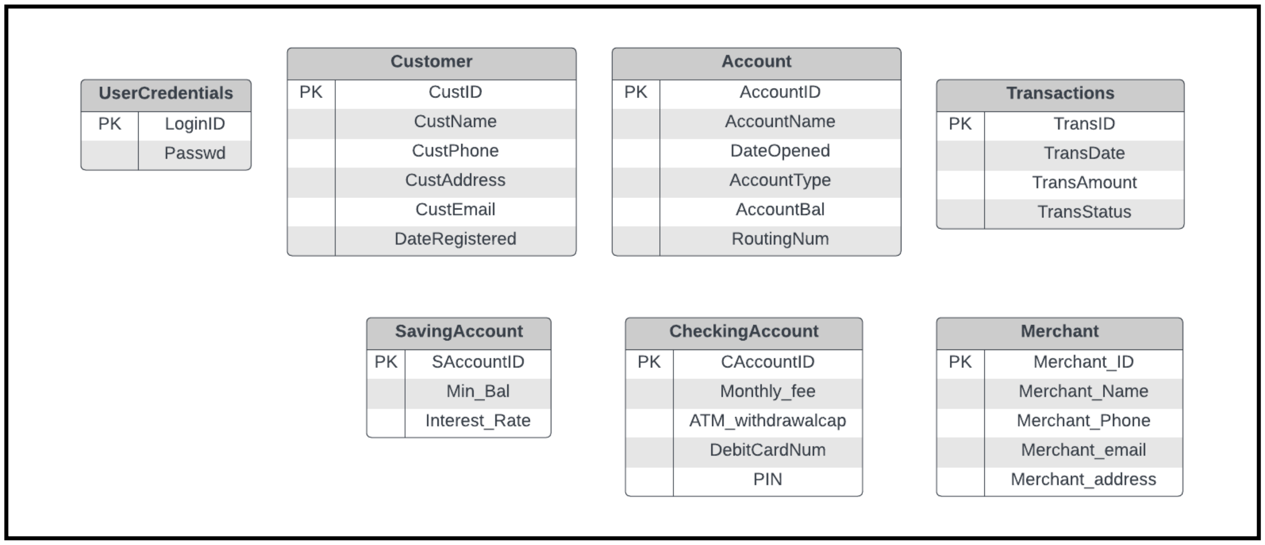
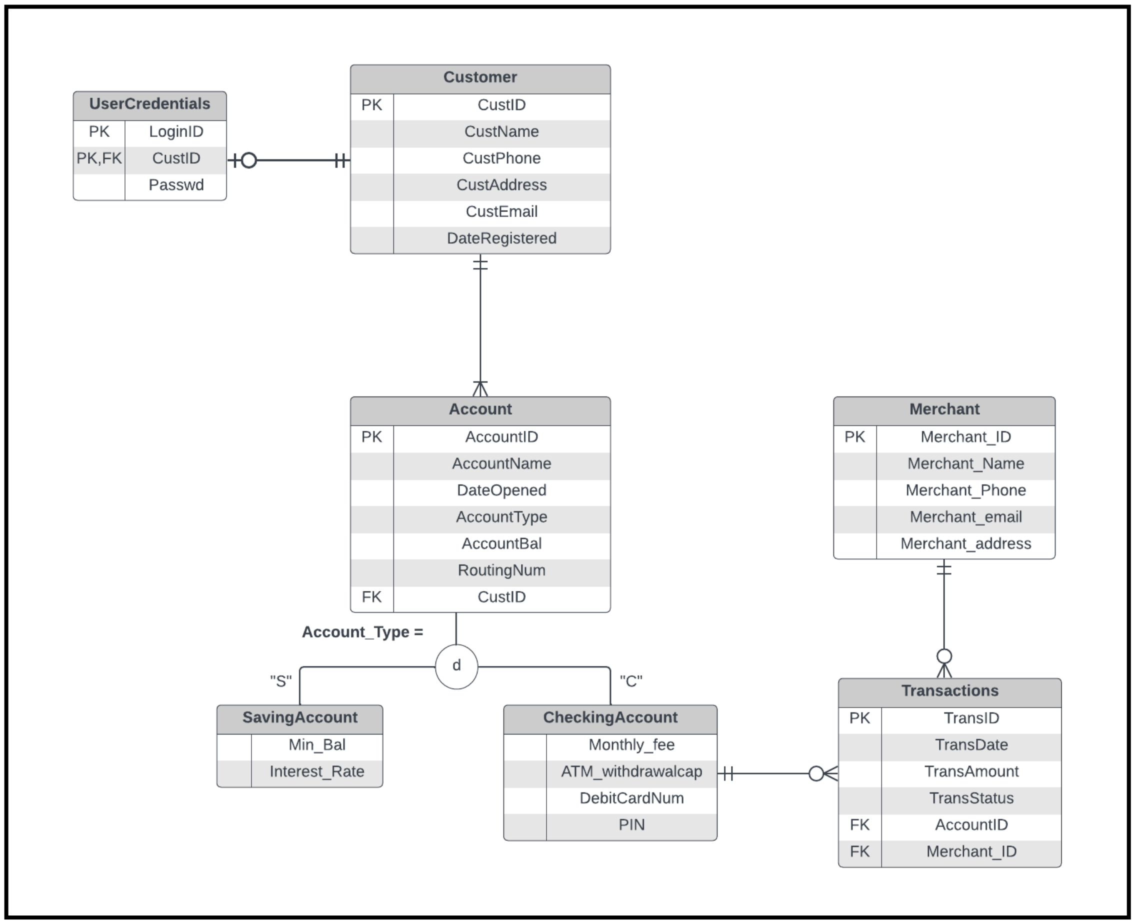
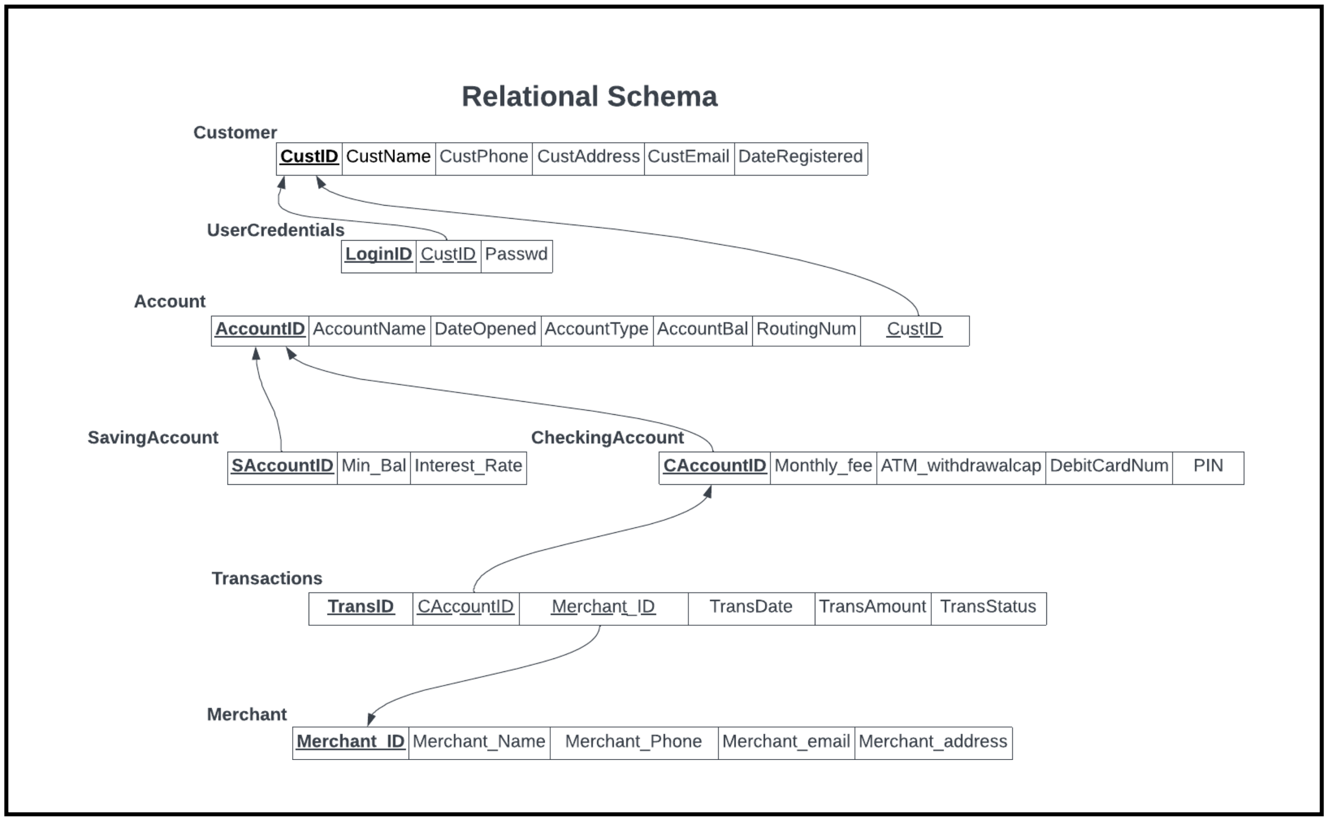


Diagram:



ER Diagram to Relational Schema:

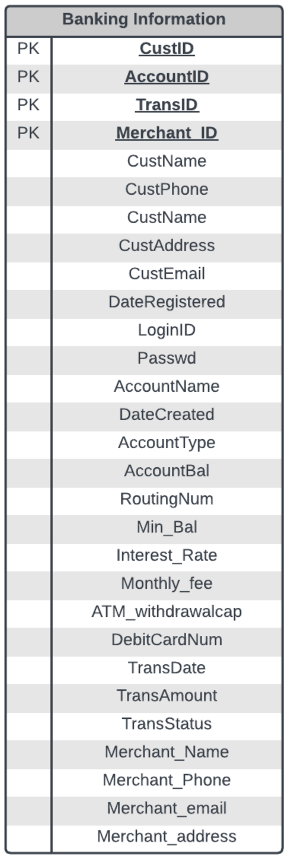
Converting an ER diagram to a relational schema helps identify and mapping the entities and relationships in the ER diagram to tables, columns, and relationships in the relational schema. The goal of converting an ER diagram to a relational schema is to create a well-structured and normalized database schema that a database management system can easily implement and use. This enables developers and users to interact with data in a consistent and efficient manner, while reducing the risk of data redundancy and inconsistency. The relational schema can also be used to create the database tables and indexes in the database management system.


Relational Schema
Customer
CustID | CustName | CustPhone | CustAddress | CustEmail | DateRegistered
UserCredentials
LoginID | CustID | Passwd
Account
AccountID | AccountName | DateOpened | AccountType | AccountBal | RoutingNum | CustID
SavingAccount
SAccountID | Min_Bal | Interest_Rate
CheckingAccount
CAccountID | Monthly_fee | ATM_withdrawalcap | DebitCardNum | PIN
Transactions
TransID | CAccountID | Merchant_ID | TransDate | TransAmount | TransStatus
Merchant
Merchant_ID | Merchant_Name | Merchant_Phone | Merchant_email | Merchant_address

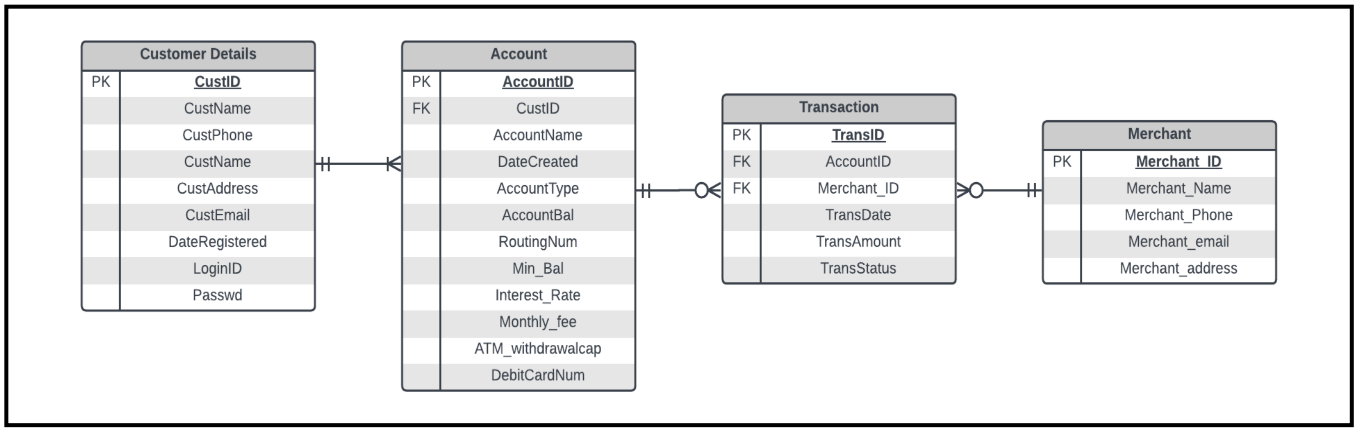
Data Normalization:

The process of organizing data in a database so that it is structured, efficient, and easy to use is known as data normalization. The primary goal of data normalization is to reduce data redundancy and improve data integrity, resulting in a more efficient and reliable database system. Data normalization is required because databases frequently contain a large amount of redundant data, which can lead to inconsistencies, errors, and inefficiencies. By normalizing data, redundant data can be eliminated or minimized, reducing storage requirements and improving database performance.

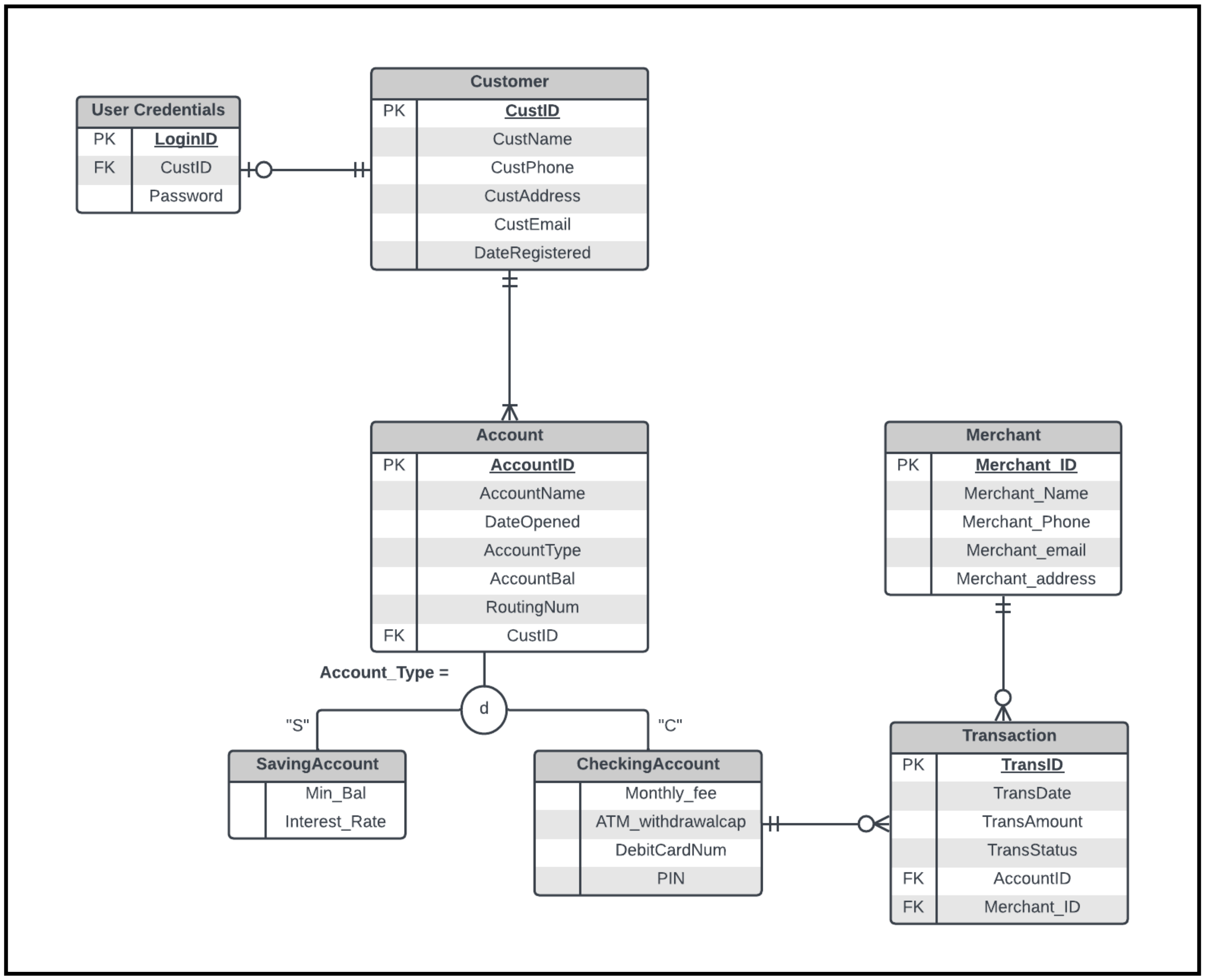
Normalization: 1st Normal Form (1NF) - contains only atomic values and ensures that each row in the table is unique.



Normalization: 2nd Normal Form (2NF) – eliminate partial dependencies and ensure that each column in the table is functionally dependent on the entire key.



Normalization: 3rd Normal Form (3NF) – eliminates transitive dependencies and ensure that each column in the table is directly related or has full dependency to the key.



Summary Table for Each Entity:

Customer Table Customer(CustID, CustName, CustPhone, CustAddress, CustEmail, DateRegistered) **Datatype**: INT, VARCHAR(50), VARCHAR(20), VARCHAR(50), VARCHAR(50), DATE respectively **Additional Details**: CustID is auto generated and unique, all other fields are required.
Account Table Account(AccountID, AccountName, Dateopened, AccountType, AccountBalance, RoutingNum, CustID) **Datatype**: INT, VARCHAR(50), DATE, VARCHAR(20), DECIMAL(10,2), VARCHAR(20), INT respectively **Additional Details**: AccountID is auto generated and unique, all other fields are required.

UserCredentials Table
UserCredentials (LoginId, CustID, Passwd)
Datatype: VARCHAR(10), INT, VARCHAR(20) respectively
Additional Details: All fields are required, and LoginId is unique for each customer.

Saving Account Table
SavingAccount(AccountID, Min_Bal, Interest_Rate)
Datatype: INT, DECIMAL(10,2), Decimal(5,2) respectively
Additional Details: AccountID is a foreign key referencing the Accounts table, all other fields are required.

CheckingAccount Table
CheckingAccount(AccountID, Monthly_Fee ATM_WithdrawalCap, DebitCardNum, PIN)
Datatype: INT, DECIMAL(10,2), INT, VARCHAR(20),VARCHAR(4) respectively
Additional Details: AccountID is a foreign key referencing the Accounts table, all other fields are required.

Merchant Table
Merchant(Merchant_ID, Merchant_Name, Merchant_Phone, Merchant_email, Merchant_address)
Datatype: INT,VARCHAR(50), VARCHAR(15), VARCHAR(50), VARCHAR(100)
Additional Details: All these attributes are defined as NOT NULL, which means that they are required for every record in the table.

Transaction Table
Transaction(TransID, TransDate, TransAmount,TransStatus,AccountID, Merchant_ID)
Datatype: INT, DATE, Decimal(10,2), Varchar(20), INT, INT
Additional Details: All these attributes are defined as NOT NULL, which means that they are required for every record in the table. Additionally, we have added two foreign key constraints to ensure that the AccountID and Merchant_ID values in the Transactions table correspond to valid entries in the Accounts and Merchants tables, respectively.

Creation of Tables:

UserCredentials Table

UserCredentials (LoginId, CustID, Passwd)

Datatype: VARCHAR(10), INT, VARCHAR(20) respectively

Additional Details: All fields are required, and LoginId is unique for each customer.

```
CREATE TABLE UserCredentials
(
LoginId VARCHAR(10) NOT NULL UNIQUE,
CustID INT NOT NULL,
Password VARCHAR(20) NOT NULL,
CONSTRAINT UserCredentials_PK PRIMARY KEY(LoginId),
CONSTRAINT UserCredentials_FK FOREIGN KEY(CustID) REFERENCES Customer(CustID)
) ;
```

Customer Table

Customer(CustID, CustName, CustPhone, CustAddress, CustEmail, DateRegistered)

Datatype: INT, VARCHAR(50), VARCHAR(20), VARCHAR(50), VARCHAR(50), DATE respectively

Additional Details: CustID is auto-generated and unique, all other fields are required.

```
CREATE TABLE Customer
(
CustID INT NOT NULL PRIMARY KEY Auto_Increment,
CustName VARCHAR(50) NOT NULL,
CustPhone VARCHAR(20) NOT NULL,
CustAddress VARCHAR(50) NOT NULL,
CustEmail VARCHAR(50) NOT NULL,
DateRegistered DATE NOT NULL
) ;
```

Account Table

Account(AccountID, AccountName, Dateopened, AccountType, AccountBalance, RoutingNum, CustID)

Datatype: INT, VARCHAR(50), DATE, VARCHAR(20), DECIMAL(10,2), VARCHAR(20), INT respectively

Additional Details: AccountID is auto-generated and unique, all other fields are required.

```
CREATE TABLE Account
(
AccountID INT NOT NULL PRIMARY KEY Auto_Increment,
AccountName VARCHAR(50) NOT NULL,
DateOpened DATE NOT NULL,
AccountType VARCHAR(20) NOT NULL,
AccountBalance DECIMAL(10,2) NOT NULL,
RoutingNum VARCHAR(20) NOT NULL,
CustID INT NOT NULL,
CONSTRAINT Account_FK FOREIGN KEY(CustID) REFERENCES Customer(CustID)
) ;
```

Saving Account Table

SavingAccount(AccountID, Min_Bal, Interest_Rate)

Datatype: INT, DECIMAL(10,2), Decimal(5,2) respectively

Additional Details: AccountID is a foreign key referencing the Accounts table, all other fields are required.

```
CREATE TABLE SavingAccount
(
AccountID INT NOT NULL PRIMARY KEY,
Min_Bal DECIMAL(10,2) NOT NULL,
Interest_Rate DECIMAL(5,2) NOT NULL,
CONSTRAINT Saving Account_FK FOREIGN KEY(AccountID) REFERENCES Account(AccountID)
);
```

CheckingAccount Table

CheckingAccount(AccountID, Monthly_Fee ATM_WithdrawalCap, DebitCardNum, PIN)

Datatype: INT, DECIMAL(10,2), INT, VARCHAR(20),VARCHAR(4) respectively

Additional Details: AccountID is a foreign key referencing the Accounts table, all other fields are required.

```
CREATE TABLE CheckingAccount
(
AccountID INT NOT NULL PRIMARY KEY,
Monthly_Fee DECIMAL(10,2) NOT NULL,
ATM_WithdrawalCap INT NOT NULL,
DebitCardNum VARCHAR(20) NOT NULL,
PIN VARCHAR(4) NOT NULL,
CONSTRAINT CheckingAccount_FK FOREIGN KEY(AccountID) REFERENCES
Account(AccountID)
);
```

Merchant Table

Merchant(Merchant_ID,
Merchant_Name,Merchant_Phone,Merchant_email,Merchant_address)

Datatype: INT,VARCHAR(50), VARCHAR(15), VARCHAR(50), VARCHAR(100)

Additional Details: All these attributes are defined as NOT NULL, which means that they are required for every record in the table.

```
CREATE TABLE Merchant (
 Merchant_ID INT NOT NULL PRIMARY KEY,
 Merchant_Name VARCHAR(50) NOT NULL,
 Merchant_Phone VARCHAR(15) NOT NULL,
 Merchant_email VARCHAR(50) NOT NULL,
 Merchant_address VARCHAR(100) NOT NULL
);
```

Transaction Table
Transaction(TransID, TransDate, TransAmount,TransStatus,AccountID, Merchant_ID)
Datatype: INT, DATE, Decimal(10,2), Varchar(20), INT, INT
Additional Details: All these attributes are defined as NOT NULL, which means that they are required for every record in the table. Additionally, we have added two foreign key constraints to ensure that the AccountID and Merchant_ID values in the Transactions table correspond to valid entries in the Accounts and Merchants tables, respectively.

```
CREATE TABLE Transaction (
   TransID INT NOT NULL PRIMARY KEY,
   TransDate DATE NOT NULL,
   TransAmountDECIMAL(10,2) NOT NULL,
   TransStatus VARCHAR(20) NOT NULL,
   AccountID INT NOT NULL,
   Merchant_ID INT NOT NULL,
   CONSTRAINT FK_Transaction_Account FOREIGN KEY (AccountID) REFERENCES
Accounts(AccountID),
   CONSTRAINT FK_Transaction_Merchant FOREIGN KEY (Merchant_ID) REFERENCES
Merchant(Merchant_ID)
);
```

Insertion of Data in Tables:

```
/************************App      Credentials**********************************/

INSERT INTO UserCredentials (LoginId, CustID, Passwd)
VALUES
   ('user1', 1, 'password1'),
   ('user2', 2, 'password2'),
   ('user3', 3, 'password3'),
   ('user4', 4, 'password4'),
   ('user5', 5, 'password5'),
   ('user6', 6, 'password6'),
   ('user7', 7, 'password7'),
   ('user8', 8, 'password8'),
   ('user9', 9, 'password9'),
   ('user10', 10, 'password10');

/************************Customers    **********************************/

INSERT INTO Customer (CustName, CustPhone, CustAddress, CustEmail, DateRegistered)
VALUES
('John Doe', '123-456-7890', '123 Main St, Anytown, USA', 'johndoe@email.com', '2020-01-01'),
('Jane Smith', '987-654-3210', '456 Oak Ave, Somecity, USA', 'janesmith@email.com', '2019-05-
15'),
```

('Bob Johnson', '555-123-4567', '789 Elm St, Anothercity, USA', 'bobjohnson@email.com', '2022-02-10'),
('Samantha Brown', '555-555-1212', '432 Pine St, Bigcity, USA', 'samanthabrown@email.com', '2021-12-01'),
('Tom Wilson', '555-555-5555', '111 Cherry Ave, Smalltown, USA', 'tomwilson@email.com', '2018-10-20'),
('Mary Jackson', '555-789-1234', '222 Cedar Blvd, Nowhereville, USA', 'maryjackson@email.com', '2023-01-01'),
('David Lee', '555-888-7777', '444 Maple Dr, Anytown, USA', 'davidlee@email.com', '2020-03-15'),
('Karen Davis', '555-444-5555', '567 Birch St, Somecity, USA', 'karendavis@email.com', '2022-05-01'),
('James Brown', '555-123-7890', '999 Oak Ln, Anothercity, USA', 'jamesbrown@email.com', '2019-08-15'),
('Megan Williams', '555-321-4567', '333 Pine Dr, Bigcity, USA', 'meganwilliams@email.com', '2022-01-01');

/************************Accounts ***********************************/

INSERT INTO Account (AccountName, DateOpened, AccountType, AccountBalance, RoutingNum, CustID)
VALUES
('John Doe', '2020-01-01', 'Checking', 5000.00, '123456789', 1),
('Jane Smith', '2019-05-15', 'Savings', 10000.00, '234567890', 2),
('Bob Johnson', '2022-02-10', 'Checking', 7500.00, '345678901', 3),
('Samantha Brown', '2021-12-01', 'Savings', 15000.00, '456789012', 4),
('Tom Wilson', '2018-10-20', 'Checking', 1000.00, '567890123', 5),
('Mary Jackson', '2023-01-01', 'Savings', 20000.00, '678901234', 6),
('David Lee', '2020-03-15', 'Checking', 3000.00, '789012345', 7),
('Karen Davis', '2022-05-01', 'Savings', 5000.00, '890123456', 8),
('James Brown', '2019-08-15', 'Checking', 2500.00, '901234567', 9),
('Megan Williams', '2022-01-01', 'Savings', 12000.00, '012345678', 10)

/************************Savings Accounts ***********************************/

INSERT INTO SavingAccount (AccountID, Min_Bal, Interest_Rate)
VALUES
(1, 1000.00, 0.50),
(2, 5000.00, 0.75),
(3, 2500.00, 0.25),
(4, 10000.00, 1.00),
(5, 500.00, 0.50),
(6, 10000.00, 0.75),
(7, 1000.00, 0.25),
(8, 2000.00, 0.50),

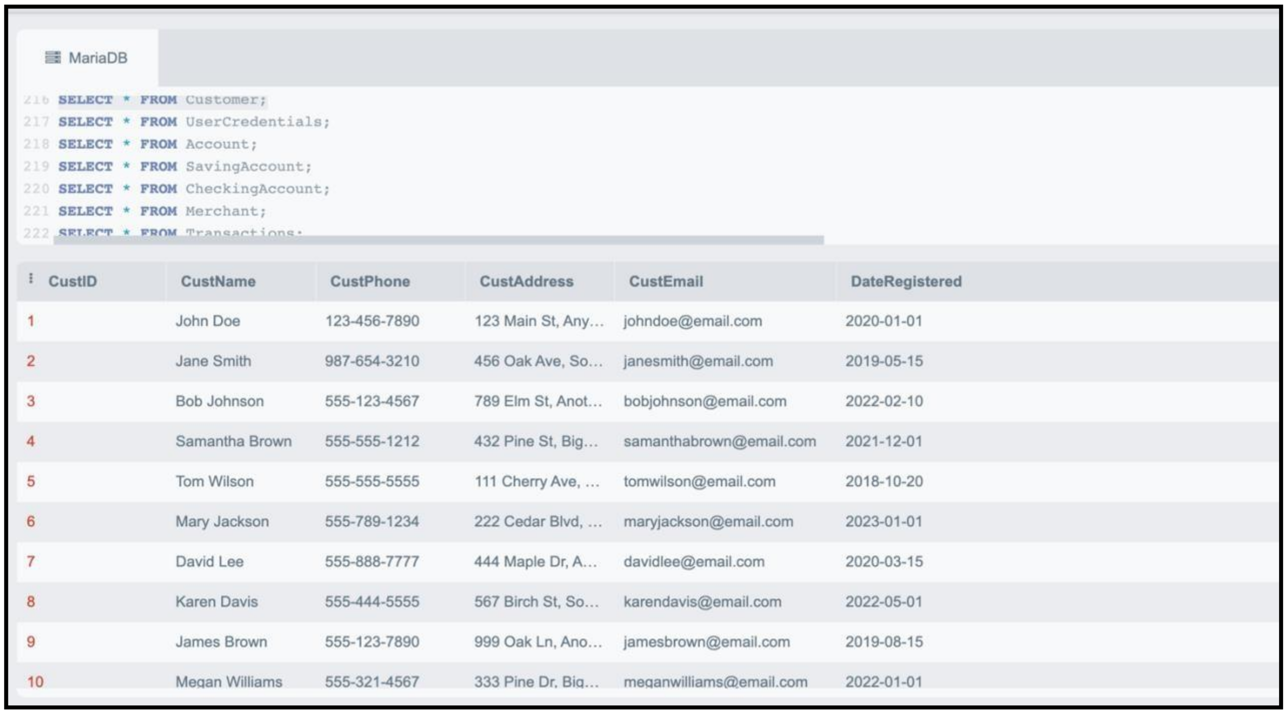
```
(9, 1500.00, 0.25),
(10, 8000.00, 0.75);

/************************Checking   Accounts   **********************************/

INSERT INTO CheckingAccount (AccountID, Monthly_Fee, ATM_WithdrawalCap, DebitCardNum,
PIN)
VALUES
(1, 10.00, 500, '1234567890123456', '1234'),
(2, 5.00, 250, '2345678901234567', '2345'),
(3, 15.00, 750, '3456789012345678', '3456'),
(4, 10.00, 500, '4567890123456789', '4567'),
(5, 0.00, 0, '5678901234567890', '5678'),
(6, 5.00, 250, '6789012345678901', '6789'),
(7, 10.00, 500, '7890123456789012', '7890'),
(8, 5.00, 250, '8901234567890123', '8901'),
(9, 15.00, 750, '9012345678901234', '9012'),
(10, 5.00, 250, '0123456789012345', '0123');

/************************Merchants     **********************************/

INSERT INTO Merchant (Merchant_ID, Merchant_Name, Merchant_Phone, Merchant_email,
Merchant_address)
VALUES
(1, 'Amazon', '+1-800-201-7575', 'support@amazon.com', '410 Terry Ave. North Seattle, WA
98109'),
(2, 'Walmart', '+1-800-925-6278', 'help@walmart.com', '702 SW 8th St, Bentonville, AR 72712'),
(3, 'Target', '+1-800-440-0680', 'guest.service@target.com', '1000 Nicollet Mall, Minneapolis,
MN 55403'),
(4, 'Best Buy', '+1-888-237-8289', 'customerservice@bestbuy.com', '7601 Penn Ave S, Richfield,
MN 55423'),
(5, 'Apple Inc.', '+1-800-275-2273', 'feedback@apple.com', '1 Apple Park Way, Cupertino, CA
95014'),
(6, 'Microsoft Corporation', '+1-800-642-7676', 'support@microsoft.com', 'One Microsoft Way,
Redmond, WA 98052'),
(7, 'Nike', '+1-800-344-6453', 'nikestore@nike.com', '1 Bowerman Dr, Beaverton, OR 97005'),
(8, 'Adidas', '+1-800-448-1796', 'customercare@adidas.com', '5055 N Greeley Ave, Portland, OR
97217'),
(9, 'Starbucks', '+1-800-782-7282', 'info@starbucks.com', '2401 Utah Ave S, Seattle, WA
98134'),
(10, 'McDonald''s', '+1-800-244-6227', 'customerservice@mcdonalds.com', '2111 McDonald''s
Dr, Oak Brook, IL 60523');
```

/************************Transactions ***********************************/

```
INSERT INTO Transaction(TransID, TransDate, TransAmount, TransStatus, AccountID,
Merchant_ID)
VALUES
(1, '2023-03-01', 100.00, 'Approved', 1, 1),
(2, '2023-03-02', 50.00, 'Approved', 2, 2),
(3, '2023-03-03', 200.00, 'Approved', 3, 3),
(4, '2023-03-04', 150.00, 'Approved', 4, 4),
(5, '2023-03-05', 75.00, 'Approved', 5, 5),
(6, '2023-03-06', 300.00, 'Approved', 6, 6),
(7, '2023-03-07', 25.00, 'Declined', 7, 7),
(8, '2023-03-08', 400.00, 'Approved', 8, 8),
(9, '2023-03-09', 80.00, 'Approved', 9, 9),
(10, '2023-03-10', 500.00, 'Declined', 10, 10);
```

*/************************INSERTION OF DATA COMPLETED************************/*

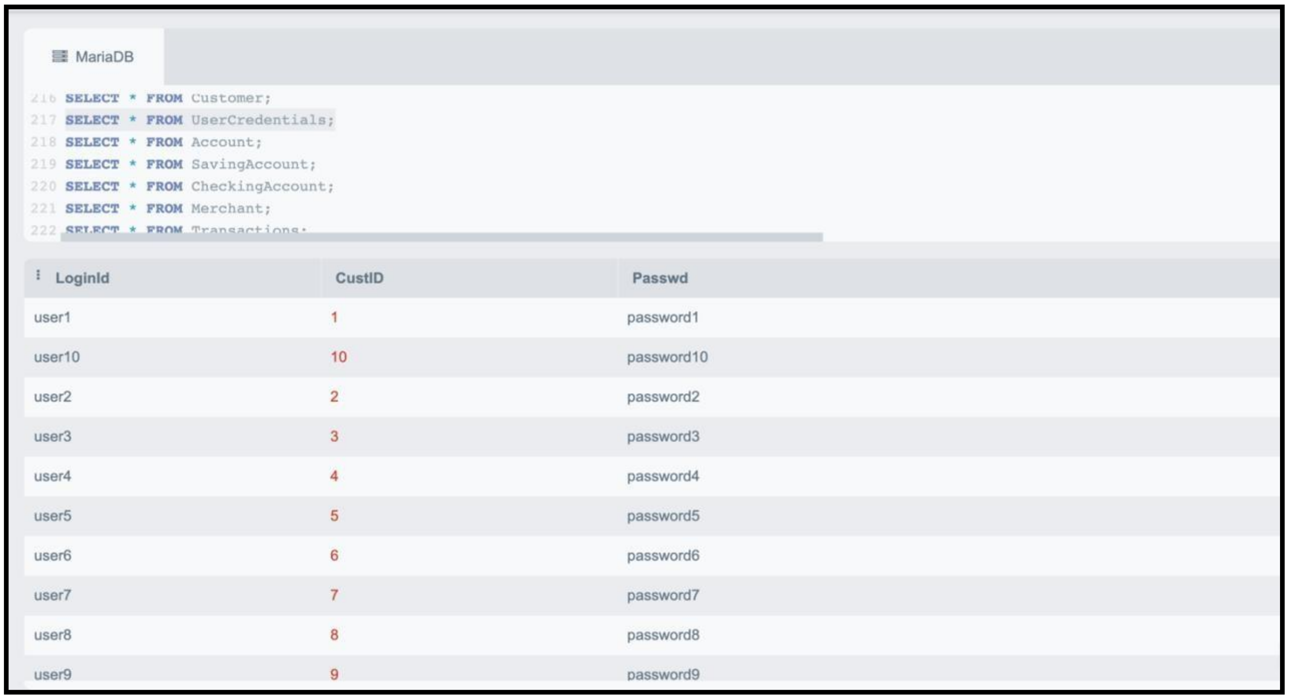
Data Loaded in DB:



MariaDB

```
SELECT * FROM Customer;
SELECT * FROM UserCredentials;
SELECT * FROM Account;
SELECT * FROM SavingAccount;
SELECT * FROM CheckingAccount;
SELECT * FROM Merchant;
SELECT * FROM Transactions;
```

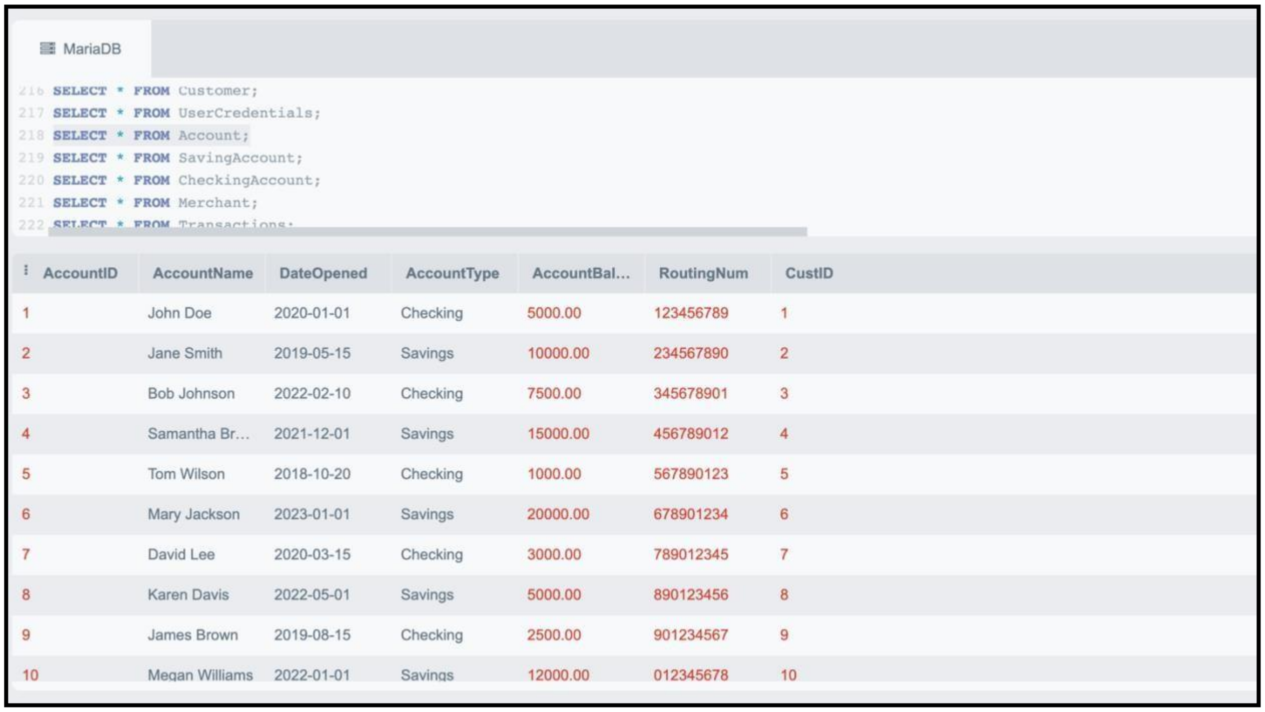
CustID	CustName	CustPhone	CustAddress	CustEmail	DateRegistered
1	John Doe	123-456-7890	123 Main St, Any…	johndoe@email.com	2020-01-01
2	Jane Smith	987-654-3210	456 Oak Ave, So…	janesmith@email.com	2019-05-15
3	Bob Johnson	555-123-4567	789 Elm St, Anot…	bobjohnson@email.com	2022-02-10
4	Samantha Brown	555-555-1212	432 Pine St, Big…	samanthabrown@email.com	2021-12-01
5	Tom Wilson	555-555-5555	111 Cherry Ave, …	tomwilson@email.com	2018-10-20
6	Mary Jackson	555-789-1234	222 Cedar Blvd, …	maryjackson@email.com	2023-01-01
7	David Lee	555-888-7777	444 Maple Dr, A…	davidlee@email.com	2020-03-15
8	Karen Davis	555-444-5555	567 Birch St, So…	karendavis@email.com	2022-05-01
9	James Brown	555-123-7890	999 Oak Ln, Ano…	jamesbrown@email.com	2019-08-15
10	Megan Williams	555-321-4567	333 Pine Dr, Big…	meganwilliams@email.com	2022-01-01



MariaDB

```
SELECT * FROM Customer;
SELECT * FROM UserCredentials;
SELECT * FROM Account;
SELECT * FROM SavingAccount;
SELECT * FROM CheckingAccount;
SELECT * FROM Merchant;
SELECT * FROM Transactions;
```

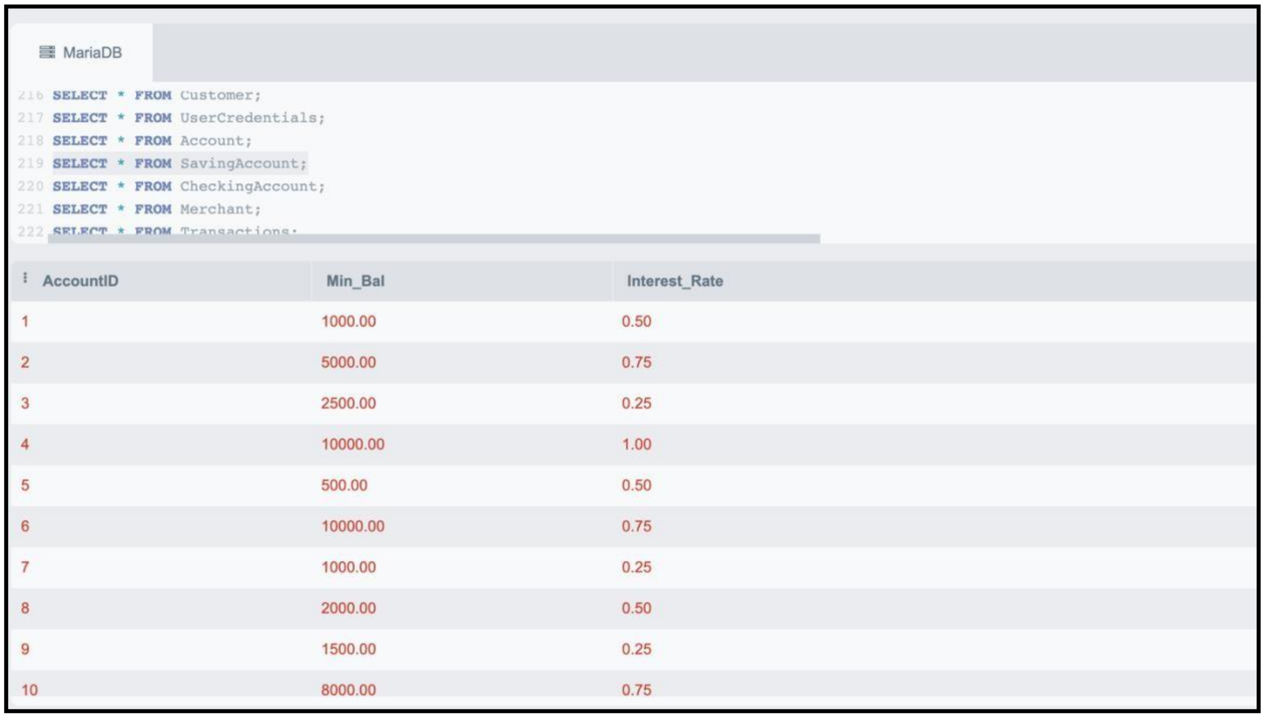
LoginId	CustID	Passwd
user1	1	password1
user10	10	password10
user2	2	password2
user3	3	password3
user4	4	password4
user5	5	password5
user6	6	password6
user7	7	password7
user8	8	password8
user9	9	password9



MariaDB

```
SELECT * FROM Customer;
SELECT * FROM UserCredentials;
SELECT * FROM Account;
SELECT * FROM SavingAccount;
SELECT * FROM CheckingAccount;
SELECT * FROM Merchant;
SELECT * FROM Transactions;
```

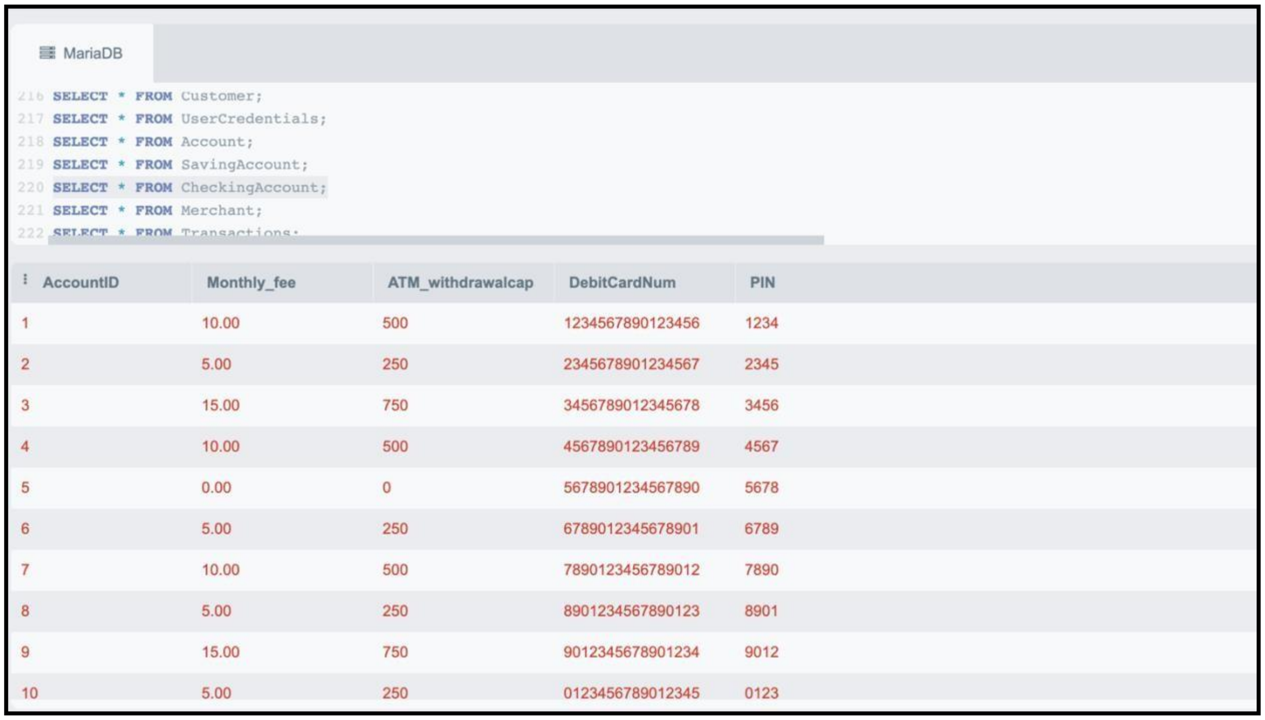
AccountID	AccountName	DateOpened	AccountType	AccountBal...	RoutingNum	CustID
1	John Doe	2020-01-01	Checking	5000.00	123456789	1
2	Jane Smith	2019-05-15	Savings	10000.00	234567890	2
3	Bob Johnson	2022-02-10	Checking	7500.00	345678901	3
4	Samantha Br...	2021-12-01	Savings	15000.00	456789012	4
5	Tom Wilson	2018-10-20	Checking	1000.00	567890123	5
6	Mary Jackson	2023-01-01	Savings	20000.00	678901234	6
7	David Lee	2020-03-15	Checking	3000.00	789012345	7
8	Karen Davis	2022-05-01	Savings	5000.00	890123456	8
9	James Brown	2019-08-15	Checking	2500.00	901234567	9
10	Megan Williams	2022-01-01	Savings	12000.00	012345678	10



MariaDB

```
SELECT * FROM Customer;
SELECT * FROM UserCredentials;
SELECT * FROM Account;
SELECT * FROM SavingAccount;
SELECT * FROM CheckingAccount;
SELECT * FROM Merchant;
SELECT * FROM Transactions;
```

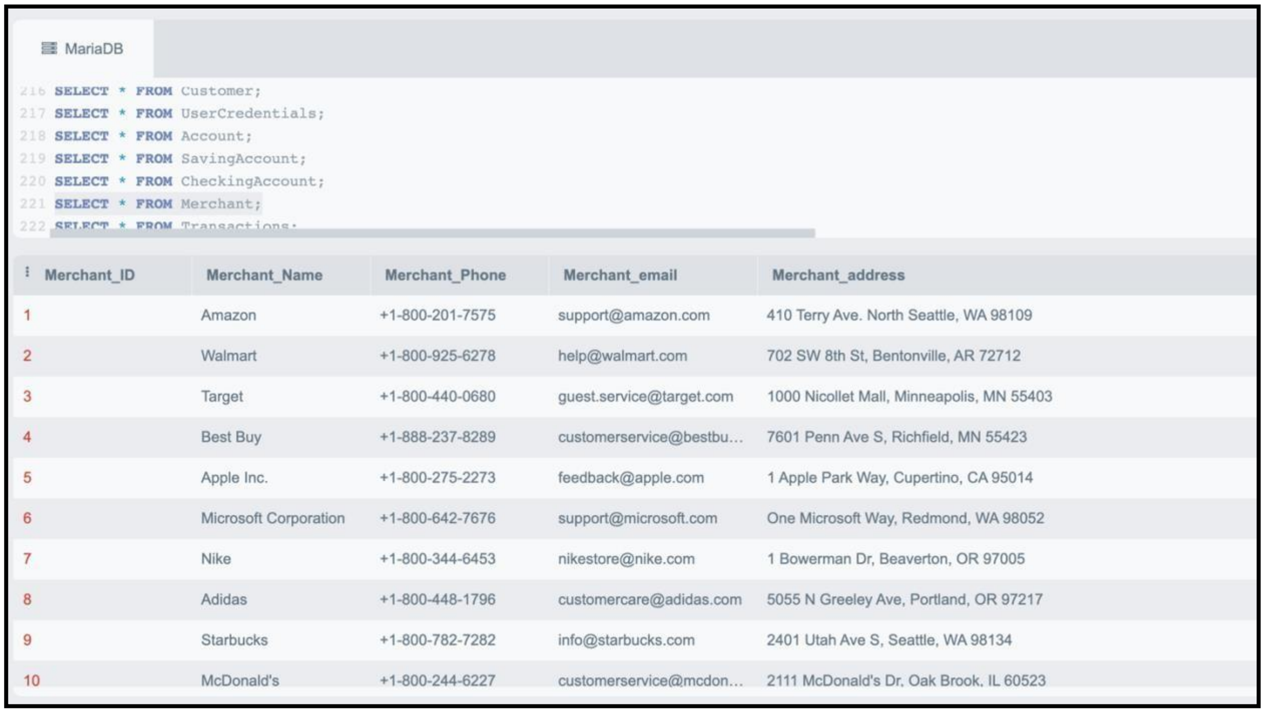
AccountID	Min_Bal	Interest_Rate
1	1000.00	0.50
2	5000.00	0.75
3	2500.00	0.25
4	10000.00	1.00
5	500.00	0.50
6	10000.00	0.75
7	1000.00	0.25
8	2000.00	0.50
9	1500.00	0.25
10	8000.00	0.75



MariaDB

```
SELECT * FROM Customer;
SELECT * FROM UserCredentials;
SELECT * FROM Account;
SELECT * FROM SavingAccount;
SELECT * FROM CheckingAccount;
SELECT * FROM Merchant;
SELECT * FROM Transactions;
```

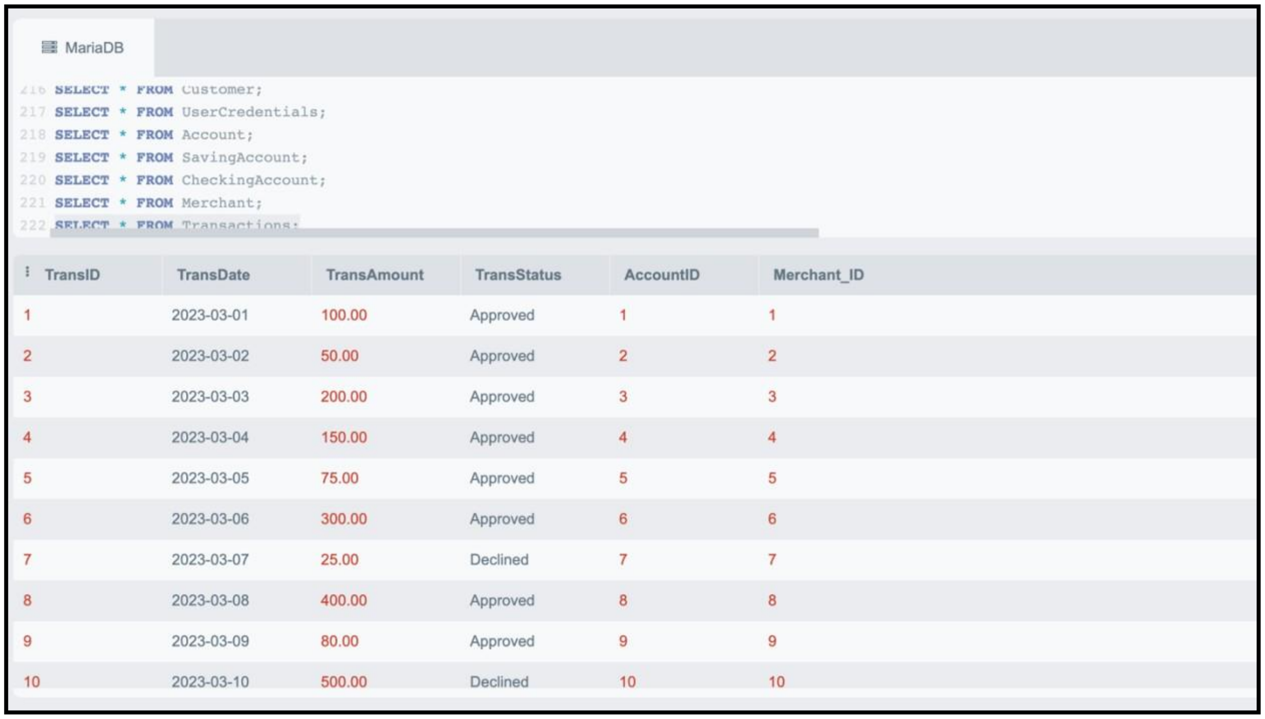
AccountID	Monthly_fee	ATM_withdrawalcap	DebitCardNum	PIN
1	10.00	500	1234567890123456	1234
2	5.00	250	2345678901234567	2345
3	15.00	750	3456789012345678	3456
4	10.00	500	4567890123456789	4567
5	0.00	0	5678901234567890	5678
6	5.00	250	6789012345678901	6789
7	10.00	500	7890123456789012	7890
8	5.00	250	8901234567890123	8901
9	15.00	750	9012345678901234	9012
10	5.00	250	0123456789012345	0123



MariaDB

```
SELECT * FROM Customer;
SELECT * FROM UserCredentials;
SELECT * FROM Account;
SELECT * FROM SavingAccount;
SELECT * FROM CheckingAccount;
SELECT * FROM Merchant;
SELECT * FROM Transactions;
```

Merchant_ID	Merchant_Name	Merchant_Phone	Merchant_email	Merchant_address
1	Amazon	+1-800-201-7575	support@amazon.com	410 Terry Ave. North Seattle, WA 98109
2	Walmart	+1-800-925-6278	help@walmart.com	702 SW 8th St, Bentonville, AR 72712
3	Target	+1-800-440-0680	guest.service@target.com	1000 Nicollet Mall, Minneapolis, MN 55403
4	Best Buy	+1-888-237-8289	customerservice@bestbu…	7601 Penn Ave S, Richfield, MN 55423
5	Apple Inc.	+1-800-275-2273	feedback@apple.com	1 Apple Park Way, Cupertino, CA 95014
6	Microsoft Corporation	+1-800-642-7676	support@microsoft.com	One Microsoft Way, Redmond, WA 98052
7	Nike	+1-800-344-6453	nikestore@nike.com	1 Bowerman Dr, Beaverton, OR 97005
8	Adidas	+1-800-448-1796	customercare@adidas.com	5055 N Greeley Ave, Portland, OR 97217
9	Starbucks	+1-800-782-7282	info@starbucks.com	2401 Utah Ave S, Seattle, WA 98134
10	McDonald's	+1-800-244-6227	customerservice@mcdon…	2111 McDonald's Dr, Oak Brook, IL 60523



MariaDB

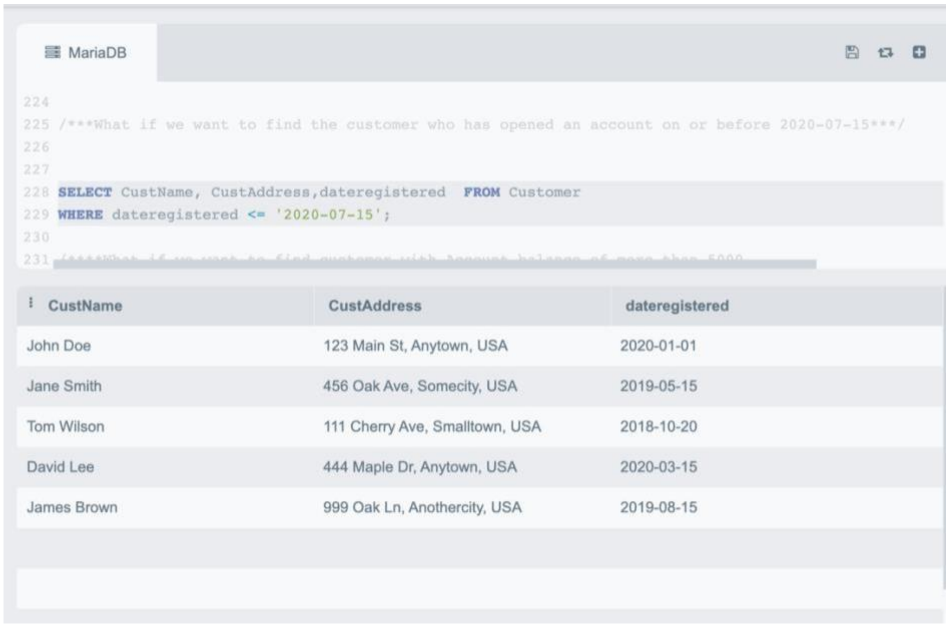
```
SELECT * FROM Customer;
SELECT * FROM UserCredentials;
SELECT * FROM Account;
SELECT * FROM SavingAccount;
SELECT * FROM CheckingAccount;
SELECT * FROM Merchant;
SELECT * FROM Transactions;
```

TransID	TransDate	TransAmount	TransStatus	AccountID	Merchant_ID
1	2023-03-01	100.00	Approved	1	1
2	2023-03-02	50.00	Approved	2	2
3	2023-03-03	200.00	Approved	3	3
4	2023-03-04	150.00	Approved	4	4
5	2023-03-05	75.00	Approved	5	5
6	2023-03-06	300.00	Approved	6	6
7	2023-03-07	25.00	Declined	7	7
8	2023-03-08	400.00	Approved	8	8
9	2023-03-09	80.00	Approved	9	9
10	2023-03-10	500.00	Declined	10	10

Queries:

- **What if we want to find the customer who opened an account on or before July 15, 2020? Show CustName, CustAddress, dateregistered**

⇒ select CustName, CustAddress,dateregistered from Customer where dateregistered <= '2020-07-15';

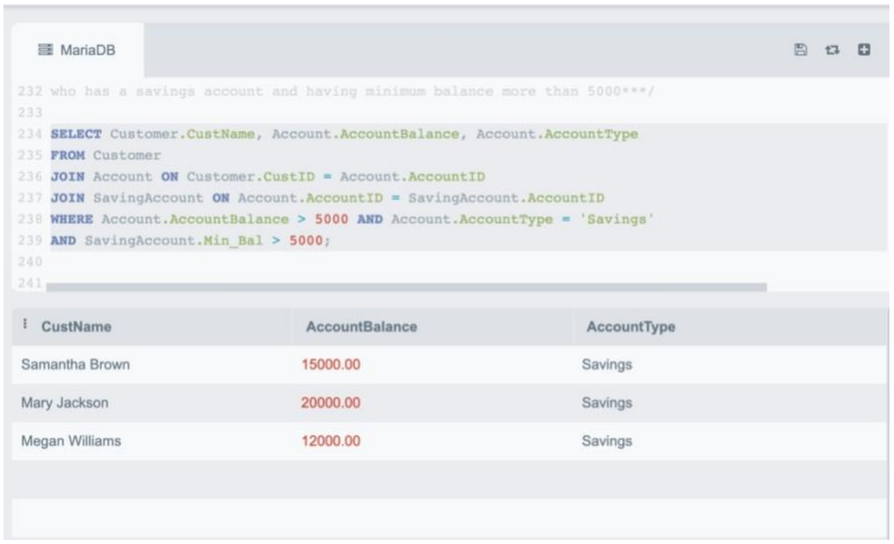


```
/***What if we want to find the customer who has opened an account on or before 2020-07-15***/


SELECT CustName, CustAddress,dateregistered  FROM Customer
WHERE dateregistered <= '2020-07-15';

/***What if we want to find customer with Account balance of more than 5000
```

CustName	CustAddress	dateregistered
John Doe	123 Main St, Anytown, USA	2020-01-01
Jane Smith	456 Oak Ave, Somecity, USA	2019-05-15
Tom Wilson	111 Cherry Ave, Smalltown, USA	2018-10-20
David Lee	444 Maple Dr, Anytown, USA	2020-03-15
James Brown	999 Oak Ln, Anothercity, USA	2019-08-15

- **What if we want to find a customer with an account balance of more than $5,000? who has a savings account with a minimum balance of more than $5,000. Show custName, AccountBalance, AccountType.**

⇒ SELECT Customer.CustName, Account.AccountBalance, Account.AccountType
FROM Customer
JOIN Account ON Customer.CustID = Account.AccountID
JOIN SavingAccount ON Account.AccountID = SavingAccount.AccountID
WHERE Account.AccountBalance > 5000 AND Account.AccountType = 'Savings'
AND SavingAccount.Min_Bal > 5000;



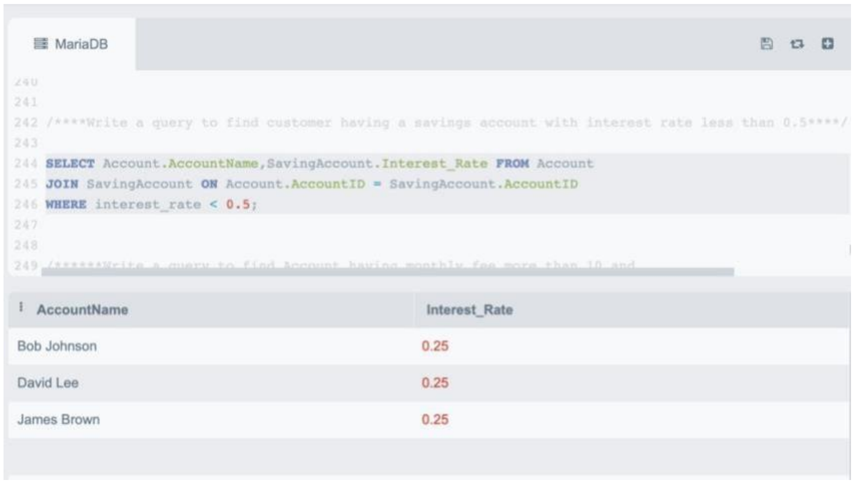
MariaDB

```sql
who has a savings account and having minimum balance more than 5000***/

SELECT Customer.CustName, Account.AccountBalance, Account.AccountType
FROM Customer
JOIN Account ON Customer.CustID = Account.AccountID
JOIN SavingAccount ON Account.AccountID = SavingAccount.AccountID
WHERE Account.AccountBalance > 5000 AND Account.AccountType = 'Savings'
AND SavingAccount.Min_Bal > 5000;
```

CustName	AccountBalance	AccountType
Samantha Brown	15000.00	Savings
Mary Jackson	20000.00	Savings
Megan Williams	12000.00	Savings

- **Write a query to find customers having a savings account with an interest rate less than 0.5. show AccountName, Interest_Rate**

⟹ SELECT Account.AccountName,SavingAccount.Interest_Rate from Account
JOIN SavingAccount on Account.AccountID = SavingAccount.AccountID
WHERE interest_rate < 0.5;



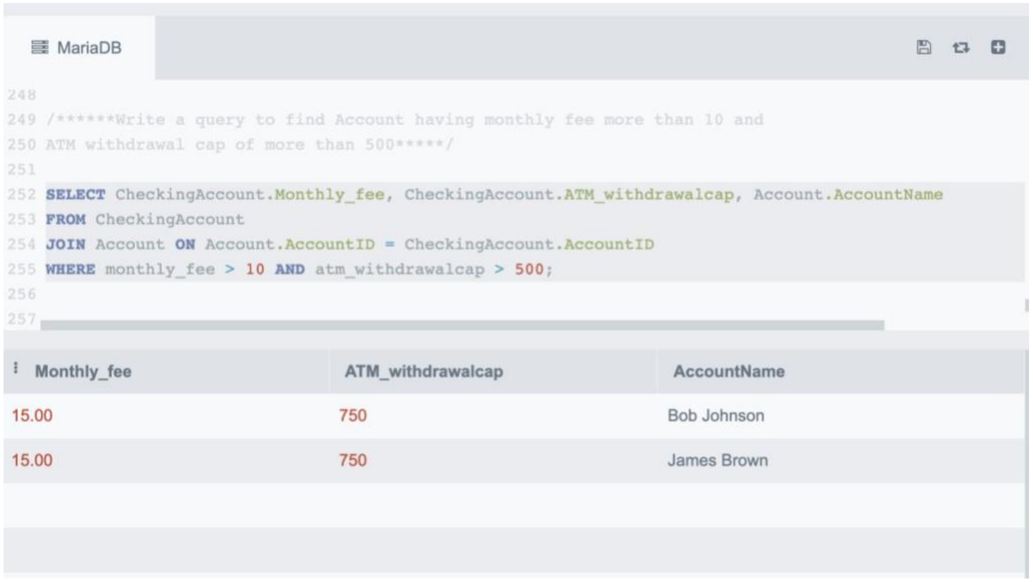
MariaDB

```sql


/****Write a query to find customer having a savings account with interest rate less than 0.5****/

SELECT Account.AccountName,SavingAccount.Interest_Rate FROM Account
JOIN SavingAccount ON Account.AccountID = SavingAccount.AccountID
WHERE interest_rate < 0.5;


/*****Write a query to find Account having monthly fee more than 10 and
```

AccountName	Interest_Rate
Bob Johnson	0.25
David Lee	0.25
James Brown	0.25

- **Write a query to find accounts with a monthly fee of more than $10 and ATM withdrawal cap of more than 500. Show Monthly_fee, ATM_withdrawalcap, AccountName.**

⇒ SELECT CheckingAccount.Monthly_fee, CheckingAccount.ATM_withdrawalcap, Account.AccountName
from CheckingAccount
JOIN Account ON Account.AccountID = CheckingAccount.AccountID
WHERE monthly_fee > 10 and atm_withdrawalcap > 500;

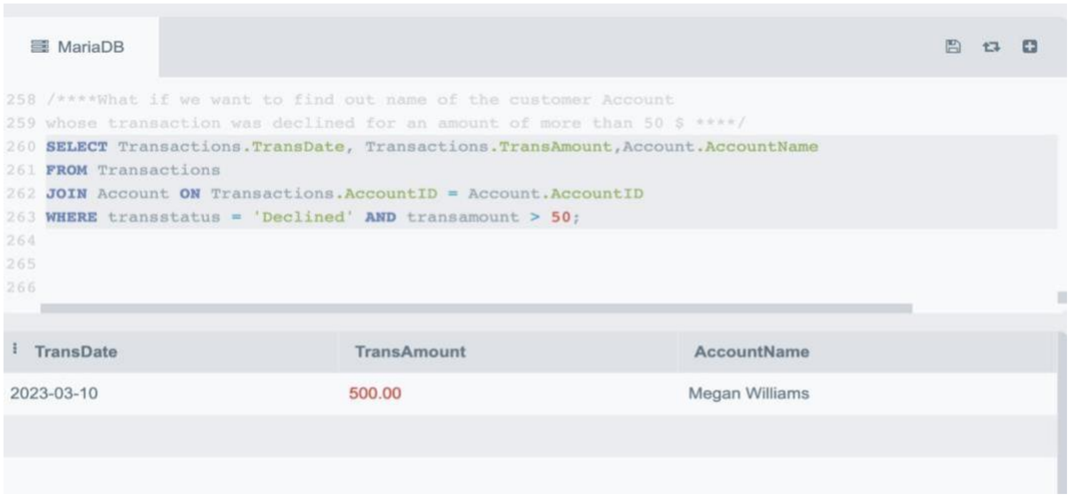

MariaDB

```
/******Write a query to find Account having monthly fee more than 10 and
ATM withdrawal cap of more than 500*****/

SELECT CheckingAccount.Monthly_fee, CheckingAccount.ATM_withdrawalcap, Account.AccountName
FROM CheckingAccount
JOIN Account ON Account.AccountID = CheckingAccount.AccountID
WHERE monthly_fee > 10 AND atm_withdrawalcap > 500;
```

Monthly_fee	ATM_withdrawalcap	AccountName
15.00	750	Bob Johnson
15.00	750	James Brown

- **What if we want to find out the name of the customer account?**
whose transaction was declined for a sum greater than $50 ? Show TransDate, TransAmount, and AccountName.

⇒ select Transactions.TransDate, Transactions.TransAmount,Account.AccountName
FROM Transactions
JOIN Account ON Transactions.AccountID = Account.AccountID
WHERE transstatus = 'Declined' and transamount > 50;


MariaDB

```
/****What if we want to find out name of the customer Account
whose transaction was declined for an amount of more than 50 $ ****/
SELECT Transactions.TransDate, Transactions.TransAmount,Account.AccountName
FROM Transactions
JOIN Account ON Transactions.AccountID = Account.AccountID
WHERE transstatus = 'Declined' AND transamount > 50;
```

TransDate	TransAmount	AccountName
2023-03-10	500.00	Megan Williams

Learnings:

The following were our group’s learnings from starting every phase, working on it and to ending each phase in a proper required format.

1. Conceptual understanding: The project aided us in better understanding the principles underlying the banking management systems. Reporting, account management, transaction processing, and customer data management are just a few of the components of the system about which we learned more.
2. Logical design: The project also taught us about a bank management system's logical design. To model the behavior and data flow of the system, we learned how to create use case diagrams and Enhanced Entity-Relationship Diagrams. To represent the various data entities and relationships present in the system, we also learned how to construct a logical data model.
3. Physical implementation: The project helped us gain practical experience in implementing a bank management system using a database management system such as SQL. We learned how to create tables, columns, and relationships to store and manage customer data, account data, and transaction data. We also learned how to automate certain tasks and ensure data integrity.
4. Querying: Finally, the project taught us how to perform queries on the SQL database to retrieve data and produce reports. We also learn how to retrieve data from multiple tables and filter it according to predetermined criteria using SQL commands like SELECT, FROM, WHERE, and JOIN.

Overall, the project was a great learning opportunity for a team like who are interested in database management systems and banking as well. It might give them the abilities and information required to use SQL to design, implement, and query a bank management system. Apart from this we also were able to work as a team, share ideas, bring up discussion all throughout our time dealing with the project which enables us to produce a successful documented project.