MISM 6213 – BUSINESS INFO DESIGN, QUALITY, STRATEGY

SUBMITTED TO:

PROF. TAREQ NASRALAH

SUBMITTED BY:

AESHA GABANI, HADDYJATOU NDIMBALAN & SHELLY GARG

MS BUSINESS ANALYTICS

D’AMORE – MCKIM SCHOOL OF BUSINESS

NORTHEASTERN UNIVERSITY

DATE – MARCH 14, 2023

PROJECT 1

CASE 2: SWIFT BANK MANAGEMENT SYSTEM

Problem Statement:

The Swift Bank Management System is a database system that is being developed by a team of entrepreneurs to support residents with real-time transaction processing. Since the database system will maintain customer accounts, balances, deposits, and transactions, it is crucial to the banking industry. The key elements of the database system and the information that must be stored have been determined upon by the entrepreneurs.

The database system will store information about customers, including their names, the date they opened an account with the bank, phone number, address, and email. Also, the system will save details regarding customer accounts, such as account IDs, names of customers to whom they belong, account balances, routing numbers, account types, and dates of account opening. Transactions will also be monitored by the system, along with the date, amount, account ID used to complete the transaction, and a special identifier for every transaction.

The database system will also hold information about businesses and the transactions that were made in their area, in addition to customer and account information. The program will save the merchant's name, phone number, email address, and special Merchant ID. Each transaction's Merchant ID and status, such as "Cancelled," "Success," "Disputed," "Disputed then Resolved," or "Declined," will also be monitored by the system. To ensure the efficient and effective operation of the bank, the business owners have determined all of the critical elements of the database system and the data that must be stored.

Functionality:

Swift Bank is a new private bank that requires an internal Bank Management System database to function. Customer data, including names, addresses, phone numbers, and account information, will be saved to the database. For security reasons, each customer will have an own login username and password. Also, the database will contain details about transactions, such as dates, amounts, and statuses. The database will also include details of the merchants' individual ID, name, contact information (including email and phone), and address.

End-users will be able to manage customer and account information, transaction monitoring, and merchant details in an organized and secure way with the help of the Swift Bank Management System.

Entities:

- Customer (Unique Identifier Customer_ID)
- Login info (Customer_ID)
- Account (Unique Identifier Account_ID)
- Savings Account (Subtype)
- Checking Account (Subtype)
- Transaction (Unique Identifier Transaction_ID)
- Merchant (Unique Identifier Merchant_ID)

Relationship between entities

Below are the relationships between above entities that one can infer from the rules defined:

- Customer ⇌ Login info
- Customer ⇌Account
- Account ⇌Savings Account — Checking Account
- Checking Account ⇌Transaction
- Transaction ⇌Merchant

Cardinalities of relationships among entities

- Customer ⇌Login info: one-to-one (1:1). A customer can have only one set of login information, and each set of login information belongs to only one customer.
- Customer ⇌Account: one-to-many (1:N). A customer can have multiple accounts, but each account belongs to only one customer.
- Account ⇌Savings Account — Checking Account: one-to-one or one-to-many (1:1 or 1:N). An account can have either one savings account or one checking account or both. Therefore, the cardinality can be either 1:1 or 1:N, depending on the design.
- Checking Account ⇌Transaction: one-to-many (1:N). A checking account can have multiple transactions, but each transaction belongs to only one checking account.
- Transaction ⇌Merchant: one-to-one (1:1). Each transaction is associated with only one merchant, and each merchant is associated with multiple transactions.

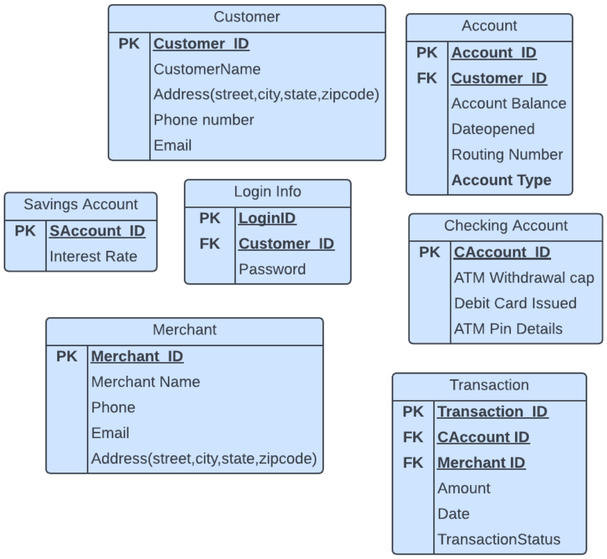
Attributes of all entities

Based on the given case, the following attributes are mandatory as per the business rules. These attributes have been loosely grouped according to domain understanding and may undergo further refinement in the future.

● **Customer ID** ● Customer Name ● Date opened ● Address(street,city,state,zipcode) ● Phone number ● Email	● **Transaction ID** ● Amount ● Date ● Transaction Status
● **Account ID** ● Account Balance ● Routing Number ● Account type	● **SAccountID** ● Interest rate ● **CAccountID** ● ATM Withdrawal ● Debit Card Issued ● ATM Pin Details
● **Merchant ID** ● Merchant Name ● Phone Number ● Email ● Address(street,city,state,zipcode)	● **Login ID** ● Password

ER-Diagram

To design the ER Diagram, I began by separating each entity and its attributes. The result of this process is presented below, along with an ER Diagram that covers the overall use case.



Customer	
PK	Customer_ID
	CustomerName
	Address(street,city,state,zipcode)
	Phone number
	Email

Account	
PK	Account_ID
FK	Customer_ID
	Account Balance
	Dateopened
	Routing Number
	Account Type

Savings Account	
PK	SAccount_ID
	Interest Rate

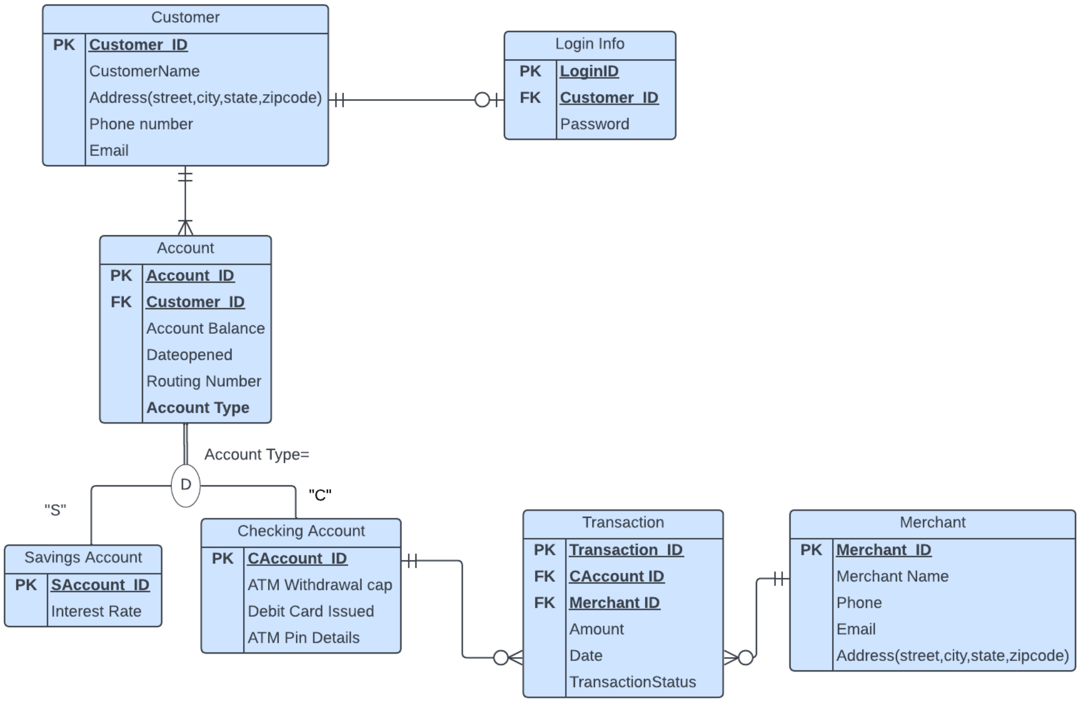
Login Info	
PK	LoginID
FK	Customer_ID
	Password

Checking Account	
PK	CAccount_ID
	ATM Withdrawal cap
	Debit Card Issued
	ATM Pin Details

Merchant	
PK	Merchant_ID
	Merchant Name
	Phone
	Email
	Address(street,city,state,zipcode)

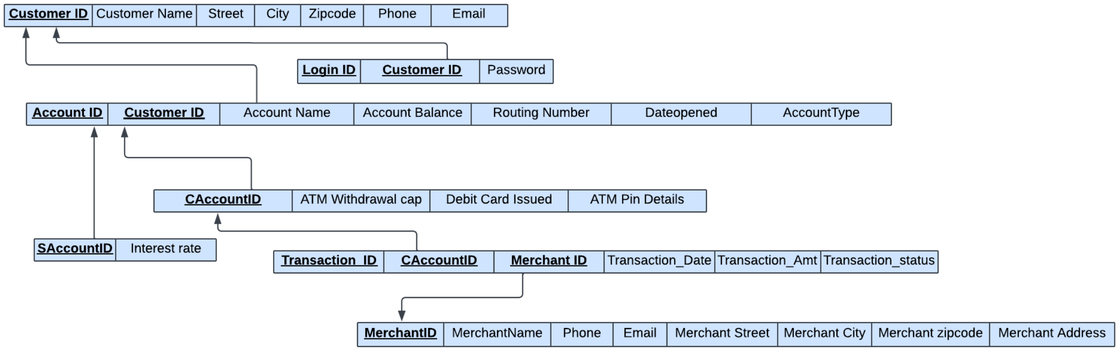
Transaction	
PK	Transaction_ID
FK	CAccount ID
FK	Merchant ID
	Amount
	Date
	TransactionStatus

Diagram:



ER Diagram to Relational Schema

The process involves converting a conceptual schema of the application domain into a data model schema that can be used by a specific DBMS, such as a relational or object-oriented data model.



Customer ID	Customer Name	Street	City	Zipcode	Phone	Email

Login ID	**Customer ID**	Password

Account ID	**Customer ID**	Account Name	Account Balance	Routing Number	Dateopened	AccountType

CAccountID	ATM Withdrawal cap	Debit Card Issued	ATM Pin Details

SAccountID	Interest rate

Transaction ID	**CAccountID**	**Merchant ID**	Transaction_Date	Transaction_Amt	Transaction_status

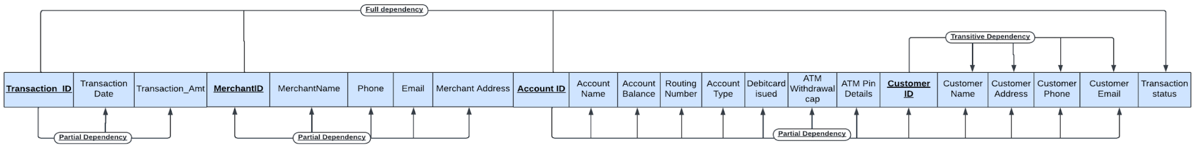
MerchantID	MerchantName	Phone	Email	Merchant Street	Merchant City	Merchant zipcode	Merchant Address

Data Normalization

A constraint has been set to identify the account type as either checking or savings, which helps in achieving normalization from 1NF to 3NF. This constraint is implemented in our SQL code and can be referenced to validate our normalization process.

Normalization : 1st normal form (1NF)

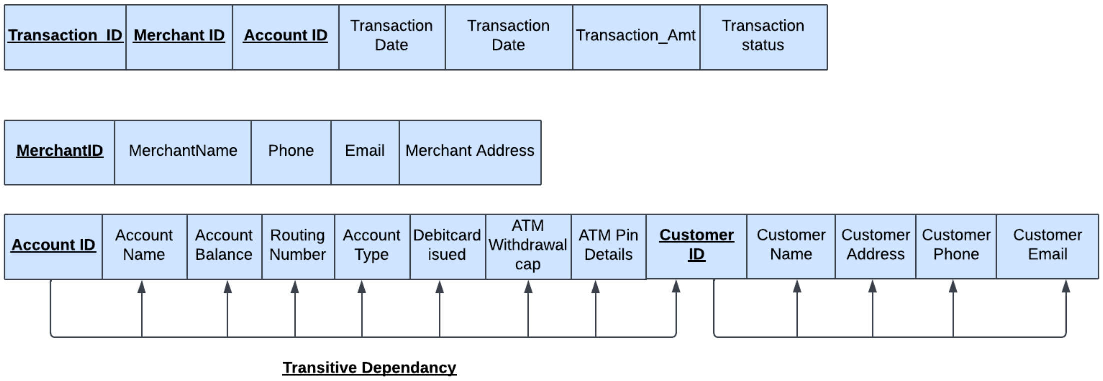
The given data can be converted to 1NF (First Normal Form) by eliminating repeating groups and identifying a primary key.



Transaction ID	Transaction Date	Transaction_Amt	**MerchantID**	MerchantName	Phone	Email	Merchant Address	**Account ID**	Account Name	Account Balance	Routing Number	Account Type	Debitcard isued	ATM Withdrawal cap	ATM Pin Details	**Customer ID**	Customer Name	Customer Address	Customer Phone	Customer Email	Transaction status

Normalization: 2nd Normal Form (2 NF)

To convert the data into 2NF, we need to ensure that there are no partial dependencies. From 1NF, its observed that Transaction ID, Merchant ID, Account ID and Customer ID have partial dependancies on certain attributes, which is required to be removed.



Transaction ID	**Merchant ID**	**Account ID**	Transaction Date	Transaction Date	Transaction_Amt	Transaction status

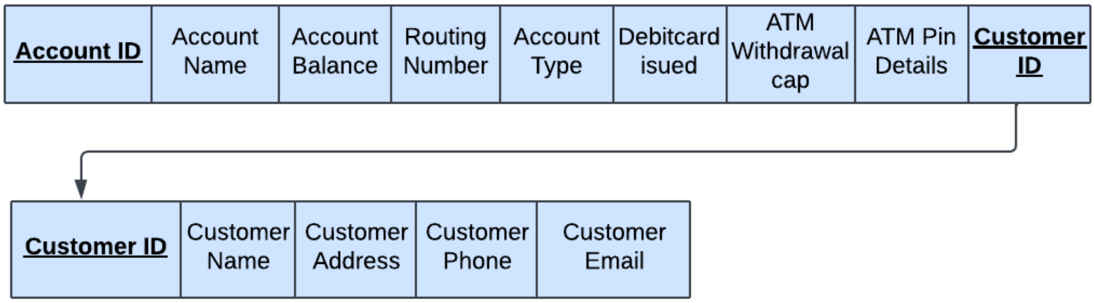
MerchantID	MerchantName	Phone	Email	Merchant Address

Account ID	Account Name	Account Balance	Routing Number	Account Type	Debitcard isued	ATM Withdrawal cap	ATM Pin Details	**Customer ID**	Customer Name	Customer Address	Customer Phone	Customer Email

In this new schema, all partial dependencies have been removed and it is in 2NF.

Normalization: 3rd Normal Form (3 NF)

To convert the above data to 3NF, we need to eliminate all transitive dependencies and ensure that every non-key attribute is functionally dependent on the primary key. Following is the 3NF diagram for the Swift Bank Management System:



Account ID	Account Name	Account Balance	Routing Number	Account Type	Debitcard isued	ATM Withdrawal cap	ATM Pin Details	**Customer ID**

Customer ID	Customer Name	Customer Address	Customer Phone	Customer Email

In this new schema, all transitive dependencies have been removed and the schema is in 3NF. The Account and Customer tables have been separated into their own tables. The Merchant and Transaction tables have been left as they were in the 2NF schema since there were no transitive dependencies involving them.

Summary table for each entity

Customer Table

Customer (Customer_ID, CustomerName, Address[Street, city, State, Zipcode], Phone Number, Email)
Datatype: VARCHAR(10), VARCHAR(30), VARCHAR(40), VARCHAR(20), VARCHAR(20), CHAR(5), VARCHAR(10), VARCHAR(35) respectively.
Additional Details: All fields are required. Address is a composite attribute. Customer_ID must be unique.

Login_info Table

Login_info (LoginID, Customer_ID, Password)
Datatype: VARCHAR(10), VARCHAR(10), VARCHAR(10) respectively.
Additional Details: All fields are required. LoginID is the primary key and Customer_ID is a foreign key referencing the Customer table.

Account Table

Account (Account_ID, Customer_ID, AccountBalance, RoutingNumber, DateOpened, AccountType)
Datatype: VARCHAR(10), VARCHAR(10), FLOAT, VARCHAR(15), DATE, VARCHAR(20) respectively.
Additional Details: All fields are required. Account_ID is the primary key and Customer_ID is a foreign key referencing Customer table. AccountType can be either SAVINGS or CHECKING.

SavingsAccount Table

SavingsAccount (Account_ID, InterestRate, AccountType)
Datatype: CHAR(10), FLOAT, VARCHAR(20) respectively.
Additional Details: All fields are required. Account_ID is a foreign key referencing Account table. InterestRate is the interest rate for the savings account. This table has a one-to-one relationship with the Account table.

CheckingAccount Table

CheckingAccount (Account_ID, ATMWithdrawalCap, DebitCardIssued, ATMPinDetails, AccountType)

Datatype: CHAR(10), VARCHAR(20), VARCHAR(3), VARCHAR(20), VARCHAR(20) respectively.

Additional Details: All fields are required. Account_ID is a foreign key referencing Account table. ATMWithdrawalCap is the maximum amount that can be withdrawn from an ATM in a day. DebitCardIssued indicates whether a debit card has been issued for the account (YES or NO). ATMPinDetails contains the details of the ATM PIN for the account. This table has a one-to-one relationship with the Account table.

Merchant Table

Merchant (Merchant_ID, Merchant_Name, Merchant_Email, Address[Street, city, State, Zipcode])

Datatype: VARCHAR(10), VARCHAR(50), VARCHAR(20), VARCHAR(40), VARCHAR(20), VARCHAR(20), VARCHAR(5) respectively.

Additional Details: All fields are required. Merchant_ID is the primary key. Address is a composite attribute which will be broken down.

Transaction Table

Transaction (Transaction_ID, Account_ID, Merchant_ID, AmountTransaction, Transdate, TransactionStatus)

Datatype: VARCHAR(10), VARCHAR(10), VARCHAR(10), FLOAT, DATE, VARCHAR(50) respectively.

Additional Details: All fields are required. Transaction_ID is the primary key. Account_ID is a foreign key referencing an Account table (either SavingsAccount or CheckingAccount). Merchant_ID is a foreign key referencing the Merchant table. AmountTransaction is the amount of the transaction. Transdate is the date of the transaction. TransactionStatus can be CANCELED, SUCCESSFUL, DISPUTED THEN RESOLVED, or DECLINED. This table has a many-to-many relationship with both SavingsAccount and Merchant tables

Creation of Tables

```
CREATE TABLE Customer
 (
  Customer_ID VARCHAR(10) NOT NULL,
  CustomerName VARCHAR(30) NOT NULL,
  Street VARCHAR(40) NOT NULL,
  City VARCHAR(20) NOT NULL,
   State VARCHAR(20) NOT NULL,
   Zipcode CHAR(5) NOT NULL,
   PhoneNumber VARCHAR(10) NOT NULL,
   Email VARCHAR(35) NOT NULL,
  CONSTRAINT Customer_PK PRIMARY KEY (Customer_ID)
  );



CREATE TABLE Logininfo
(
    LoginID VARCHAR(10) NOT NULL,
    Customer_ID VARCHAR(10) NOT NULL,
     Passwords VARCHAR(10) NOT NULL,
     CONSTRAINT Loginfo_PK PRIMARY KEY ( LoginID),
     CONSTRAINT Customer_FK FOREIGN KEY (Customer_ID)
     REFERENCES Customer(Customer_ID)
 );



CREATE TABLE Account
(
     Account_ID VARCHAR(10) NOT NULL,
     Customer_ID VARCHAR(10) NOT NULL,
     AccountBalance FLOAT NOT NULL,
     RoutingNumber VARCHAR(15) NOT NULL,
     DateOpened DATE NOT NULL,
    AccountType VARCHAR(20) NOT NULL CHECK(AccountType IN ('SAVINGS', 
'CHECKING')),
    CONSTRAINT Account_PK PRIMARY KEY (Account_ID),
     CONSTRAINT Account_FK1 FOREIGN KEY (Customer_ID)
     REFERENCES Customer(Customer_ID)
);


CREATE TABLE SavingsAccount
(
     Account_ID VARCHAR(10) NOT NULL,
     InterestRate FLOAT NOT NULL,
     AccountType VARCHAR(20) NOT NULL,
     CONSTRAINT SavingsAccount_PK PRIMARY KEY (Account_ID),
     CONSTRAINT SavingsAccount_FK1 FOREIGN KEY (Account_ID)
      REFERENCES Account(Account_ID),
      CONSTRAINT SavingsAccount_CHK CHECK (AccountType = 'SAVINGS')
);


 CREATE TABLE CheckingAccount
(
      Account_ID VARCHAR(10) NOT NULL,
     ATMWithdrawalCap VARCHAR(20),
     DebitCardIssued VARCHAR(3) CHECK(DebitCardIssued IN ('YES','NO')),
    AccountType VARCHAR(20) NOT NULL,
    ATMPinDetails VARCHAR(20),
     CONSTRAINT CheckingAccount_PK PRIMARY KEY (Account_ID),
     CONSTRAINT CheckingAccount_FK1 FOREIGN KEY (Account_ID)
     REFERENCES Account(Account_ID),
    CONSTRAINT CheckingAccount_CHK CHECK (AccountType = 'CHECKING')
);


CREATE TABLE Merchant
 (
     Merchant_ID VARCHAR(10) NOT NULL,
     Merchant_Name VARCHAR(50) NOT NULL,
     Merchant_Email VARCHAR(20) NOT NULL,
     Street VARCHAR(40) NOT NULL,
     City VARCHAR(20) NOT NULL,
     State VARCHAR(20) NOT NULL,
     Zipcode VARCHAR(5) NOT NULL,
     CONSTRAINT Merchant_PK PRIMARY KEY (Merchant_ID),
);



CREATE TABLE Transactions
 (
    Transaction_ID VARCHAR(10) NOT NULL,
     Account_ID VARCHAR(10) NOT NULL,
     Merchant_ID VARCHAR(10) NOT NULL,
      AmountTransaction FLOAT NOT NULL,
      Transdate DATE NOT NULL,
      TransactionStatus VARCHAR(50) CHECK(TransactionStatus IN
('CANCELLED','SUCCESSFUL','DISPUTED THEN RESOLVED','DECLINED')),
    CONSTRAINT Transactions_PK PRIMARY KEY (Transaction_ID),
    CONSTRAINT Transactions_FK1 FOREIGN KEY (Account_ID)
     REFERENCES SavingsAccount(Account_ID),
   CONSTRAINT Transactions_FK2 FOREIGN KEY (Merchant_ID)
    REFERENCES Merchant(Merchant_ID)
);
```

<u>Insertion of Data in Tables:</u>

*********Customer Table********

```
INSERT INTO Customer (Customer_ID, CustomerName, Street, City, State, Zipcode,
PhoneNumber, Email) VALUES
('C001', 'John Doe', '123 Main St', 'New York', 'NY', '10001', '555-1234', 'johndoe@gmail.com'),
INSERT INTO Customer (Customer_ID, CustomerName, Street, City, State, Zipcode,
PhoneNumber, Email) VALUES('C002', 'Jane Smith', '456 Elm St', 'Los Angeles', 'CA', '90001',
'555-5678', 'janesmith@yahoo.com'),
INSERT INTO Customer (Customer_ID, CustomerName, Street, City, State, Zipcode,
PhoneNumber, Email) VALUES('C003', 'Bob Johnson', '789 Oak St', 'Chicago', 'IL', '60601',
'555-9876', 'bobjohnson@hotmail.com'),
INSERT INTO Customer (Customer_ID, CustomerName, Street, City, State, Zipcode,
PhoneNumber, Email) VALUES('C004', 'Mary Williams', '1010 Pine St', 'Houston', 'TX',
'77002', '555-4321', 'marywilliams@gmail.com');

********Login in Table*******
INSERT INTO LoginInfo (LoginID, Customer_ID, Password) VALUES
('L001', 'C001', 'password123'),
INSERT INTO LoginInfo (LoginID, Customer_ID, Password) VALUES
('L002', 'C002', 'abcde12345'),
INSERT INTO LoginInfo (LoginID, Customer_ID, Password) VALUES
('L003', 'C003', 'qwertyuiop'),
INSERT INTO LoginInfo (LoginID, Customer_ID, Password) VALUES
('L004', 'C004', 'asdfghjkl;');

**********Account Table******
INSERT INTO Account (Account_ID, Customer_ID, AccountBalance, RoutingNumber,
AccountType,DateOpened) VALUES
('A001', 'C001', 1000.00, '123456789', 'SAVINGS','2022-01-01'),
INSERT INTO Account (Account_ID, Customer_ID, AccountBalance, RoutingNumber,
AccountType) VALUES('A002', 'C001', 2000.00, '123456789', 'CHECKING','2022-01-01'),
INSERT INTO Account (Account_ID, Customer_ID, AccountBalance, RoutingNumber,
AccountType) VALUES('A003', 'C002', 1500.00, '234567890', 'SAVINGS','2022-01-01'),
INSERT INTO Account (Account_ID, Customer_ID, AccountBalance, RoutingNumber,
AccountType) VALUES('A004', 'C002', 3000.00, '234567890', 'CHECKING','2022-01-01');

******Savings_Account******
INSERT INTO SavingsAccount (Account_ID, InterestRate) VALUES('A001', 0.01),
INSERT INTO SavingsAccount (Account_ID, InterestRate) VALUES('A003', 0.02),
INSERT INTO SavingsAccount (Account_ID, InterestRate) VALUES('A005', 0.03),
INSERT INTO SavingsAccount (Account_ID, InterestRate) VALUES('A007', 0.04);

*****Checking_Account*****
INSERT INTO CheckingAccount (Account_ID, ATMWithdrawalCap, DebitCardIssued,
ATMPinDetails) VALUES ('CA0001', '$500', 'YES', '1234'),
INSERT INTO CheckingAccount (Account_ID, ATMWithdrawalCap, DebitCardIssued,
ATMPinDetails) VALUES('CA0002', '$200', 'YES', '5678'),
INSERT INTO CheckingAccount (Account_ID, ATMWithdrawalCap, DebitCardIssued,
ATMPinDetails) VALUES('CA0003', '$1000', 'NO', NULL),
INSERT INTO CheckingAccount (Account_ID, ATMWithdrawalCap, DebitCardIssued,
ATMPinDetails) VALUES('CA0004', '$300', 'YES', '2468');
```

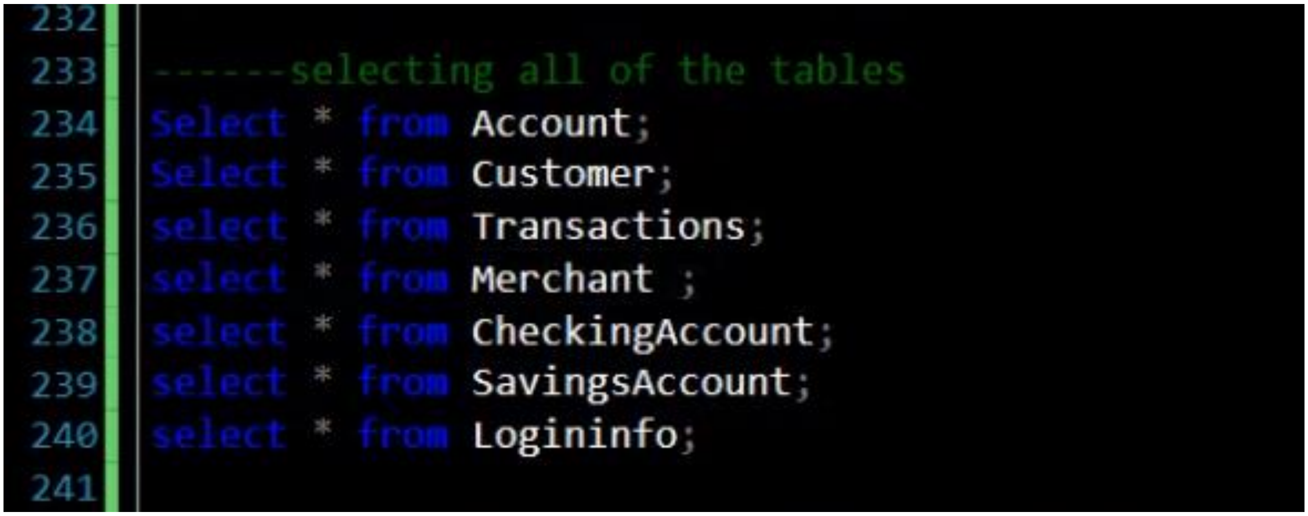
*******Transaction Table********

INSERT INTO Transaction (Transaction_ID, Account_ID, Merchant_ID, AmountTransaction, Transdate, TransactionStatus) VALUES('T001', 'A001', 'M001', 50.00, '2022-01-15', 'SUCCESSFUL'),
INSERT INTO Transaction (Transaction_ID, Account_ID, Merchant_ID, AmountTransaction, Transdate, TransactionStatus) VALUES('T002', 'A002', 'M002', 25.00, '2022-02-28', 'SUCCESSFUL'),
INSERT INTO Transaction (Transaction_ID, Account_ID, Merchant_ID, AmountTransaction, Transdate, TransactionStatus) VALUES('T003', 'A003', 'M003', 75.00, '2022-03-15', 'DISPUTED THEN RESOLVED'),
INSERT INTO Transaction (Transaction_ID, Account_ID, Merchant_ID, AmountTransaction, Transdate, TransactionStatus) VALUES('T004', 'A004', 'M004', 100.00, '2022-04-30', 'DECLINED');

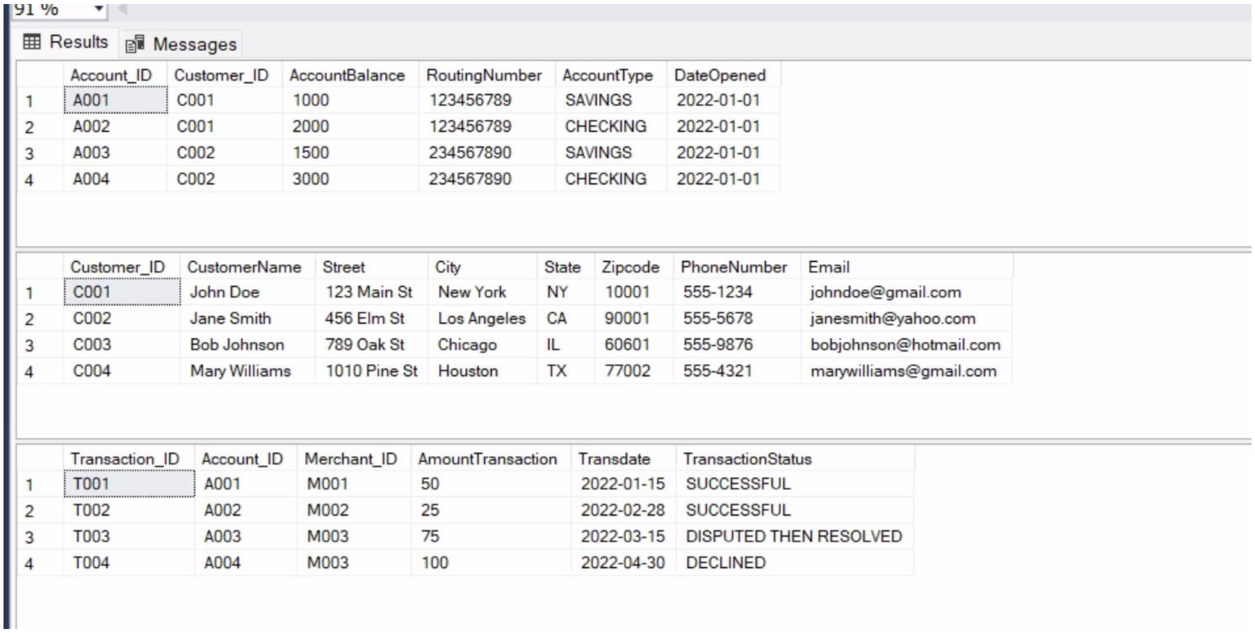
**********Merchant Table***********

INSERT INTO Merchant (Merchant_ID, Merchant_Name, Merchant_Email, Street, City, State, Zipcode) VALUES('M001', 'Amazon', 'contact@amazon.com', '410 Terry Ave', 'Seattle', 'WA', '98109'),
INSERT INTO Merchant (Merchant_ID, Merchant_Name, Merchant_Email, Street, City, State, Zipcode) VALUES('M002', 'Walmart', 'contact@walmart.com', '702 SW 8th St', 'Bentonville', 'AR', '72716'),
INSERT INTO Merchant (Merchant_ID, Merchant_Name, Merchant_Email, Street, City, State, Zipcode) VALUES('M003', 'Target', 'contact@target.com', '1000 Nicollet Mall', 'Minneapolis',

<u>Data loaded in DB:</u>



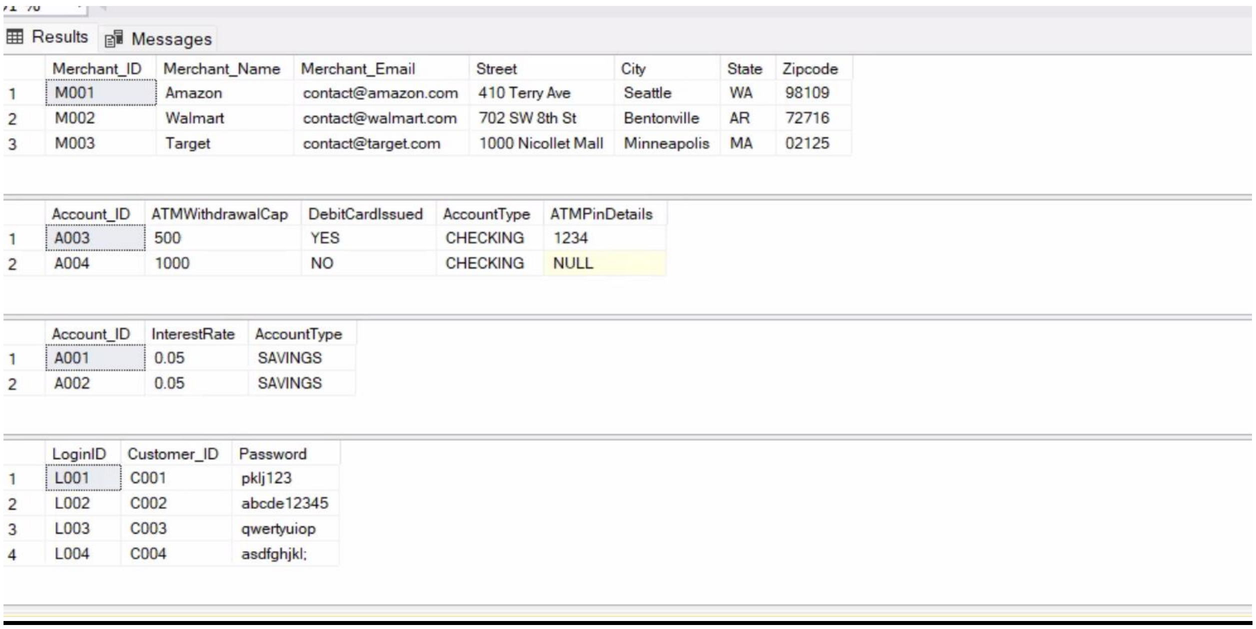
```

------selecting all of the tables
Select * from Account;
Select * from Customer;
select * from Transactions;
select * from Merchant ;
select * from CheckingAccount;
select * from SavingsAccount;
select * from Logininfo;

```



	Account_ID	Customer_ID	AccountBalance	RoutingNumber	AccountType	DateOpened
1	A001	C001	1000	123456789	SAVINGS	2022-01-01
2	A002	C001	2000	123456789	CHECKING	2022-01-01
3	A003	C002	1500	234567890	SAVINGS	2022-01-01
4	A004	C002	3000	234567890	CHECKING	2022-01-01

	Customer_ID	CustomerName	Street	City	State	Zipcode	PhoneNumber	Email
1	C001	John Doe	123 Main St	New York	NY	10001	555-1234	johndoe@gmail.com
2	C002	Jane Smith	456 Elm St	Los Angeles	CA	90001	555-5678	janesmith@yahoo.com
3	C003	Bob Johnson	789 Oak St	Chicago	IL	60601	555-9876	bobjohnson@hotmail.com
4	C004	Mary Williams	1010 Pine St	Houston	TX	77002	555-4321	marywilliams@gmail.com

	Transaction_ID	Account_ID	Merchant_ID	AmountTransaction	Transdate	TransactionStatus
1	T001	A001	M001	50	2022-01-15	SUCCESSFUL
2	T002	A002	M002	25	2022-02-28	SUCCESSFUL
3	T003	A003	M003	75	2022-03-15	DISPUTED THEN RESOLVED
4	T004	A004	M003	100	2022-04-30	DECLINED



	Merchant_ID	Merchant_Name	Merchant_Email	Street	City	State	Zipcode
1	M001	Amazon	contact@amazon.com	410 Terry Ave	Seattle	WA	98109
2	M002	Walmart	contact@walmart.com	702 SW 8th St	Bentonville	AR	72716
3	M003	Target	contact@target.com	1000 Nicollet Mall	Minneapolis	MA	02125

	Account_ID	ATMWithdrawalCap	DebitCardIssued	AccountType	ATMPinDetails
1	A003	500	YES	CHECKING	1234
2	A004	1000	NO	CHECKING	NULL

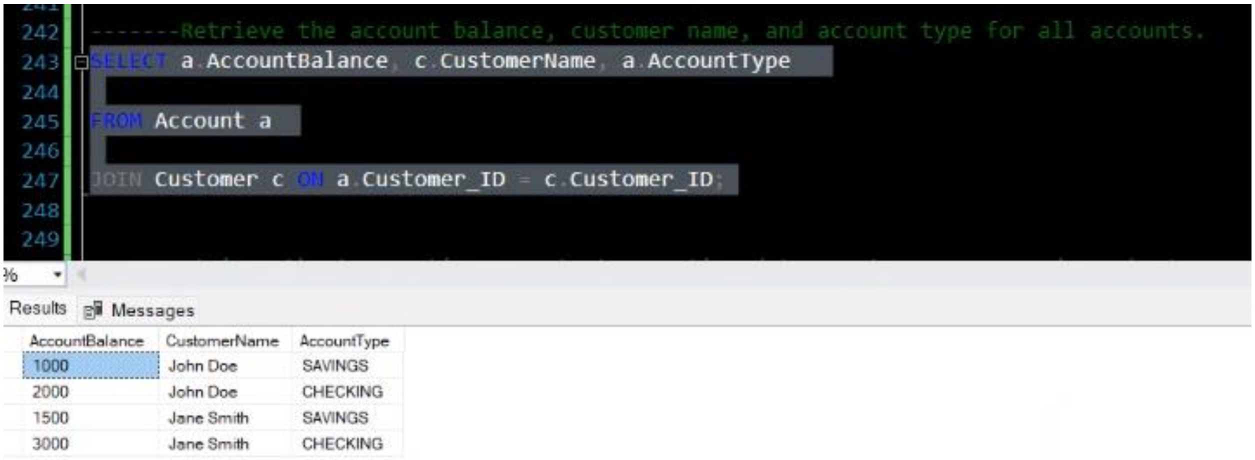
	Account_ID	InterestRate	AccountType
1	A001	0.05	SAVINGS
2	A002	0.05	SAVINGS

	LoginID	Customer_ID	Password
1	L001	C001	pklj123
2	L002	C002	abcde12345
3	L003	C003	qwertyuiop
4	L004	C004	asdfghjkl;

Queries:

- Retrieve the account balance, customer name, and account type for all accounts:

```
SELECT a.AccountBalance, c.CustomerName, a.AccountType
FROM Account a
JOIN Customer c ON a.Customer_ID=c.Customer_ID;
```

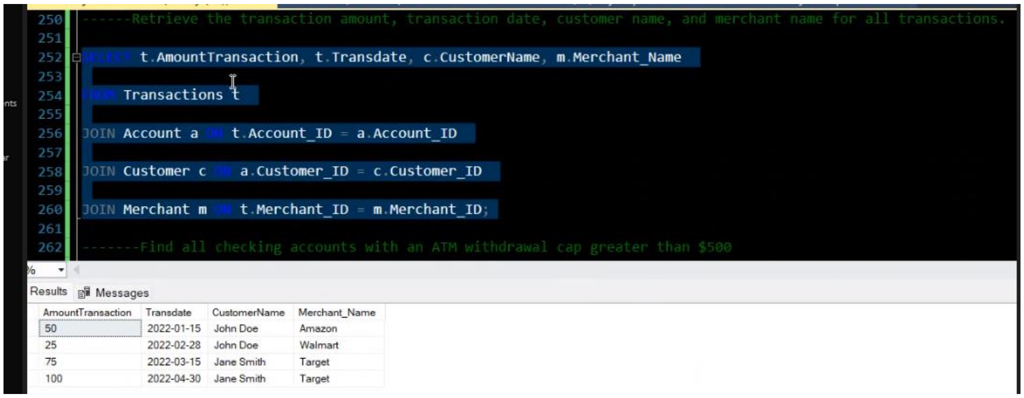


```
-------Retrieve the account balance, customer name, and account type for all accounts.
SELECT a.AccountBalance, c.CustomerName, a.AccountType

FROM Account a

JOIN Customer c ON a.Customer_ID = c.Customer_ID;
```

AccountBalance	CustomerName	AccountType
1000	John Doe	SAVINGS
2000	John Doe	CHECKING
1500	Jane Smith	SAVINGS
3000	Jane Smith	CHECKING

- Retrieve the transaction amount, transaction date, customer name, and merchant name for all transactions:

```
SELECT t.AmountTransaction, t.Transdate, c.CustomerName, m.Merchant_Name
FROM Transactions
JOIN Account a ON t.Account_ID = a.Account_ID
JOIN Customer c ON a.Customer_ID = c.Customer_ID
JOIN Merchant m ON t.Merchant_ID = m.Merchant_ID;
```

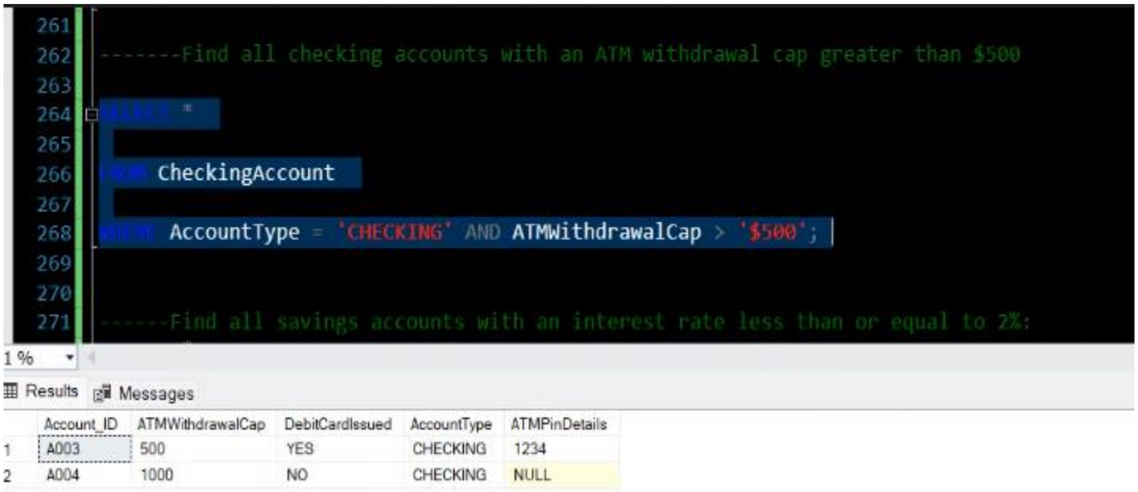


```
------Retrieve the transaction amount, transaction date, customer name, and merchant name for all transactions.

SELECT t.AmountTransaction, t.Transdate, c.CustomerName, m.Merchant_Name

FROM Transactions t

JOIN Account a ON t.Account_ID = a.Account_ID

JOIN Customer c ON a.Customer_ID = c.Customer_ID

JOIN Merchant m ON t.Merchant_ID = m.Merchant_ID;

-------Find all checking accounts with an ATM withdrawal cap greater than $500
```

AmountTransaction	Transdate	CustomerName	Merchant_Name
50	2022-01-15	John Doe	Amazon
25	2022-02-28	John Doe	Walmart
75	2022-03-15	Jane Smith	Target
100	2022-04-30	Jane Smith	Target

- Find all checking accounts with an ATM withdrawal cap greater than $500:

  SELECT *
  FROM CheckingAccount
  WHERE AccountType = ‘CHECKING’ AND ATMWithdrawalCap > ‘$500’;

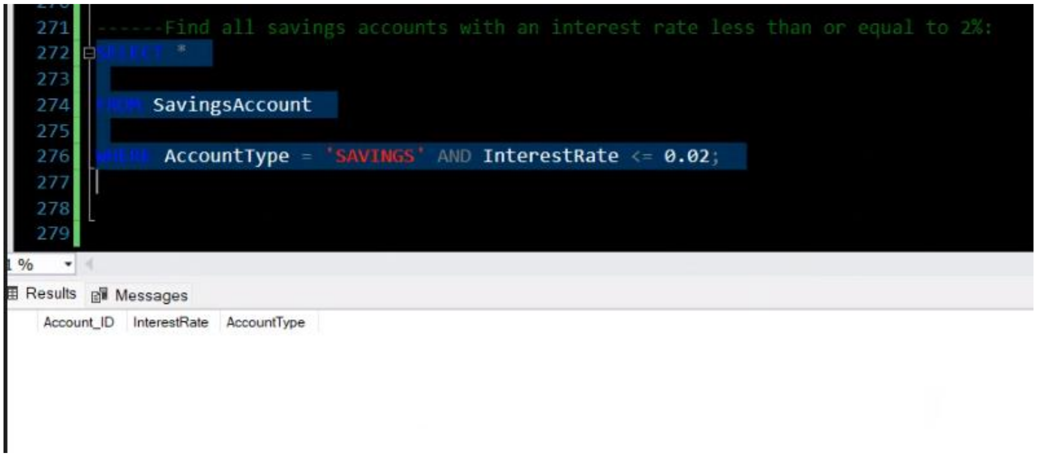


```
-------Find all checking accounts with an ATM withdrawal cap greater than $500

SELECT *

FROM CheckingAccount

WHERE AccountType = 'CHECKING' AND ATMWithdrawalCap > '$500';


------Find all savings accounts with an interest rate less than or equal to 2%:
```

	Account_ID	ATMWithdrawalCap	DebitCardIssued	AccountType	ATMPinDetails
1	A003	500	YES	CHECKING	1234
2	A004	1000	NO	CHECKING	NULL

- Find all savings account with an interest rate less than or equal to 2%:

  SELECT *
  FROM SavingsAccount
  WHERE AccountType = ‘SAVINGS’ AND InterestRate<=0.02;



```
------Find all savings accounts with an interest rate less than or equal to 2%:
SELECT *

FROM SavingsAccount

WHERE AccountType = 'SAVINGS' AND InterestRate <= 0.02;
```

Account_ID	InterestRate	AccountType

Learning Outcomes:

Through completing this assignment, We have:

- Gained an understanding of how to organize and structure data using normalization techniques to improve data integrity and reduce redundancy.
- Learned how to identify and separate data into different tables based on functional dependencies and eliminate data anomalies.
- Formulated queries using SQL syntax and commands, such as SELECT, FROM, WHERE, and JOIN, to retrieve and manipulate data from databases.
- Analyzed data and interpreted the results to solve problems and answer questions.
- Improved our ability to handle errors and exceptions in SQL queries.
- Developed a general knowledge of data analysis and management techniques and how they relate to databases.
- Applied the process of data normalization and SQL queries to a given business problem to design efficient and well-structured databases.
- Enhanced our critical thinking skills by evaluating the effectiveness of normalization in reducing redundancy, improving data integrity, and enhancing data maintenance and updateability.
- Gained familiarity with the specific database used in the queries.