**-SVKM’s NMIMS**

**MPSTME , Shirpur**

A.Y. 2023 - 24

**Course: Database Management Systems Project Report**

|  |  |  |
| --- | --- | --- |
| Program | B.Tech (C.S.) | |
| Semester | IV | |
| Name of the Project: | Banking System | |
|  | | |
| Details of Project Members |  |  |
| Batch | Roll No. | Name |
| A1 | E219 | Mrinal Verma |
| A1  A1 | E228  E201 | Abhishek Pal  Tanishq Tayal |
| Submitted To : Prof. Sachin Bhandari | | |

**Project Report**

***Banking System***

**By**

**Mrinal Verma E219**

**Abhishek Pal E228**

**Tanishq Tayal E201**

**Course: DBMS AY: 2023-24**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Topic** | **Page no.** |
| **1** | Storyline |  |
| **2** | Components of Database Design |  |
| **3** | Entity Relationship Diagram |  |
| **4** | Normalization |  |
| **5** | SQL Queries |  |
| **6** | Learning from the Project |  |
| **7** | Project Demonstration |  |
| **8** | Self-learning beyond classroom |  |
| **9** | Learning from the project |  |

|  |  |  |
| --- | --- | --- |
| **10** | Challenges faced |  |
| **11** | Conclusion |  |

# Storyline

**Introduction**

In today's banking world, fancy computer systems are crucial for handling a lot of information quickly and safely. This story will look at how to make one of these systems for a pretend bank, showing the complicated decisions needed to handle banking data.

**Background:**

The pretend bank we're talking about is big and works all around the world. It does many things, like handling savings and checking accounts, loans, and credit cards. With millions of customers and lots of daily transactions, the bank needs a good way to manage loads of data.

# Components of Database Design

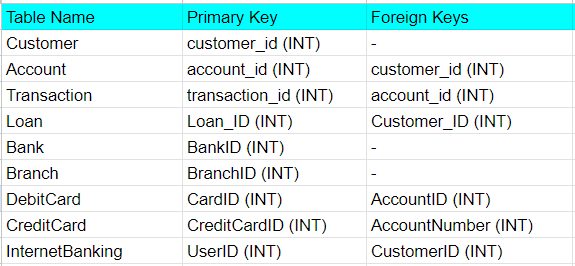
The first thing we do is figure out what the pretend bank needs. It has to keep track of customer info, account details, and transactions. Making sure everything is safe, the data is correct, and the system is always ready to work are very important things to think about.

Functional Requirements:

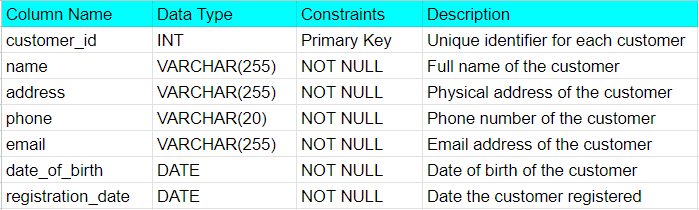
* User Management:
  + Account creation and profile management.
  + Login and authentication with strong security measures.
  + User roles and permissions based on access levels.
* Account Management:
  + Different account types (checking, savings, loans, investments).
  + View account balances and transaction history.
  + Deposit and withdraw funds.
  + Transfer funds between accounts (same and different banks).
  + Schedule recurring transactions.
* Transactions:
  + Secure and reliable transaction processing.
  + Real-time transaction updates across the system.
  + Error handling and rollback mechanisms.
* Other functionalities:
  + Bill payments and online recharges.
  + Loan applications and tracking.
  + Investment planning and management.
  + Reporting and analytics for users and administrators.
  + Integrations with external systems (credit bureaus, payment gateways).

Non-Functional Requirements:

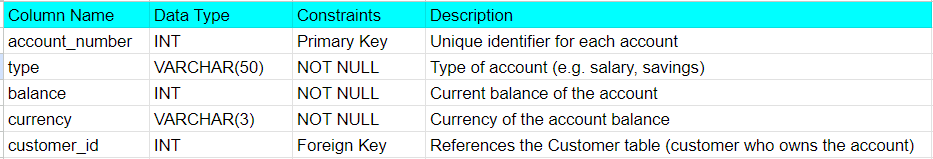
* Security:
  + Data encryption at rest and in transit.
  + Strong authentication and authorization protocols.
  + Access control based on user roles and permissions.
  + Audit trails and logging for security compliance.
* Performance:
  + Fast response times for critical operations (transactions, account balances).
  + High availability and scalability to handle peak loads.
  + Fault tolerance and disaster recovery mechanisms.
* Usability:
  + Simple and intuitive user interface for different user roles.
  + Responsive design for various devices (desktops, mobiles, tablets).
  + Accessibility features for users with disabilities.
* Data Management:
  + Data integrity and consistency across the system.
  + Regular backups and disaster recovery plans.
  + Easy data import and export capabilities.
* **Database Schema:**



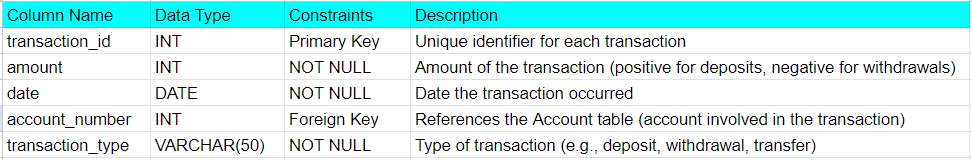
* Customer Table:



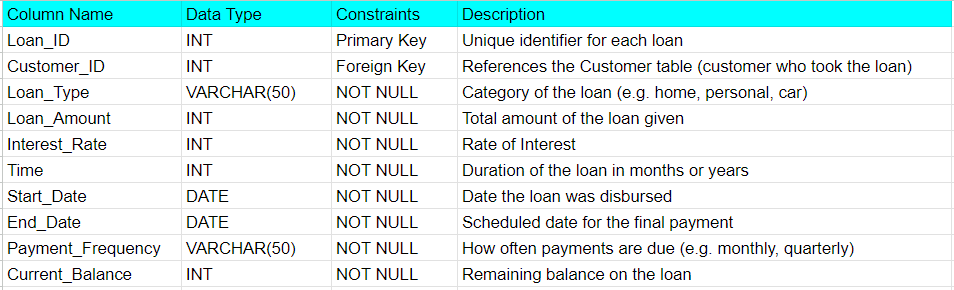
* Account Table:



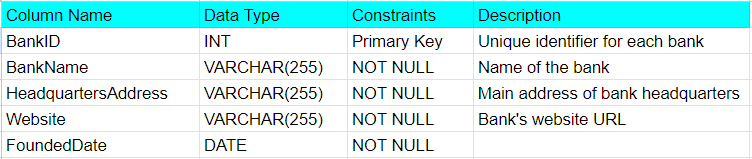
* Transaction Table:



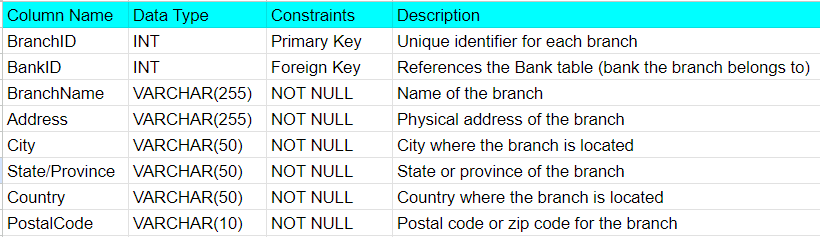
* Loan Table



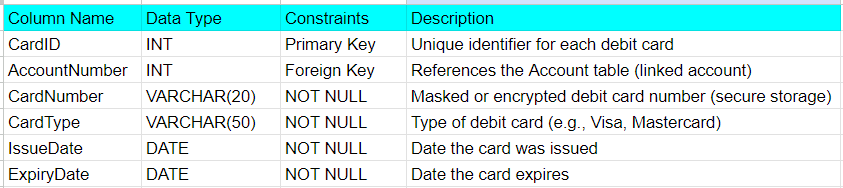
* Bank Table :



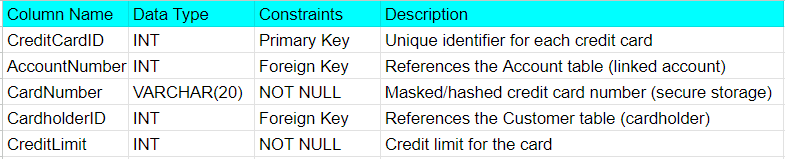
* Branch Table:



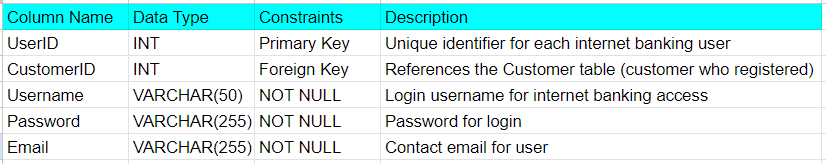
* DebitCard Table:



* CreditCard Table:

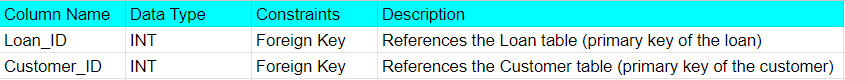


* InternetBanking Table:



* Loan and Customer have M:N cardinality ratio. So we create another table.

Loan\_Customer Table:

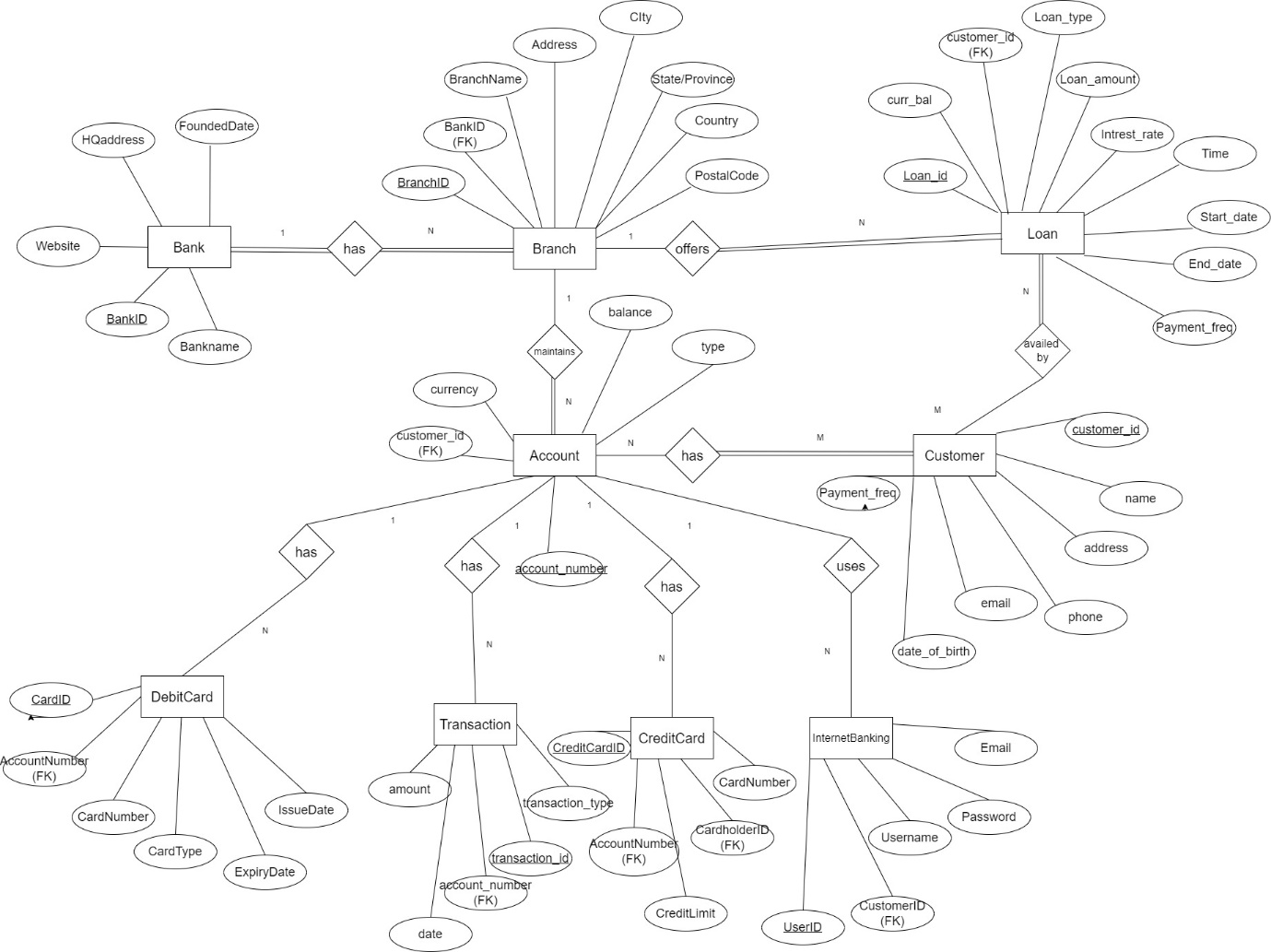


* Account and Customer also have M:N cardinality ratio.

**Account\_Customer Table:**



**Entity Relationship Diagram**



**Normalization**

Normalization is a crucial process in database design that helps ensure data integrity and efficiency. In our banking system project, we have normalized our tables to at least 3rd Normal Form (3NF), which means that all attributes are fully functionally dependent on the primary key, and there are no transitive dependencies.

Here's how normalization applies to some of our tables:

**1.** **Customer Table (Customer):**

- The primary key is customer\_id, and attributes like name, address, phone, email, date of birth, registration date, and password are fully functionally dependent on customer\_id. There are no transitive dependencies in this table.

- Functional Dependencies: customer\_id → {name, address, phone, email, date of birth, registration date, password}

**2.** **Account Table (Account):**

- The primary key is account number, and attributes like type, balance, currency, and customer ID are fully functionally dependent on the account number. There are no transitive dependencies in this table.

- Functional Dependencies: account\_number → {type, balance, currency, customer\_id}

**3.** **Transaction Table (Transaction):**

- The primary key is transaction ID, and attributes like amount, date, account number, and transaction type are fully functionally dependent on the transaction ID. There are no transitive dependencies in this table.

- Functional Dependencies: transaction\_id → {amount, date, account\_number, transaction\_type}

- Functional Dependencies: account\_number → {transaction\_id}

**4. Loan Table (Loan):**

- The primary key is loan ID, and attributes like customer ID, loan type, loan amount, interest rate, time, start date, end date, payment frequency, and current balance are fully functionally dependent on the loan ID. There are no transitive dependencies in this table.

- Functional Dependencies: Loan\_ID → {Customer\_ID, Loan\_Type, Loan\_Amount, Interest\_Rate, Time, Start\_Date, End\_Date, Payment\_Frequency, Current\_Balance}

- Functional Dependencies: Customer\_ID → {Loan\_ID}

**5. Branch Table (Branch):**

- The primary key is branch ID, and attributes like password, branch name, address, city, state/province, country, and postal code are fully functionally dependent on the branch ID. There are no transitive dependencies in this table.

- Functional Dependencies: BranchID → {Password, BranchName, Address, City, State\_Province, Country, PostalCode}

**6. DebitCard Table (DebitCard):**

- The primary keys are card ID and account number, and attributes like card number, card type, issue date, expiry date, and PIN are fully functionally dependent on these primary keys. There are no transitive dependencies in this table.

- Functional Dependencies: CardID → {AccountNumber, CardNumber, CardType, IssueDate, ExpiryDate, Pin}

- Functional Dependencies: AccountNumber → {CardID}

**7. CreditCard Table (CreditCard):**

- The primary keys are credit card ID and account number, and attributes like card number, card type, issue date, expiry date, PIN, and credit limit are fully functionally dependent on these primary keys. There are no transitive dependencies in this table.

- Functional Dependencies: CreditCardID → {AccountNumber, CardNumber, CardType, IssueDate, ExpiryDate, Pin, CreditLimit}

- Functional Dependencies: AccountNumber → {CreditCardID}

**8. InternetBanking Table (InternetBanking):**

- The primary key is user ID, and attributes like customer ID, username, password, and email are fully functionally dependent on the user ID. There are no transitive dependencies in this table.

- Functional Dependencies: UserID → {CustomerID, Username, Password, Email}

- Functional Dependencies: CustomerID → {UserID}

By ensuring that our tables are in 3rd Normal Form, we can maintain data integrity and optimize our database for efficient operations.

**Functional dependancies**

In our banking system project, we have identified the following functional dependencies within our tables:

1. Customer Table:

- Customer ID (C\_Id) → First Name, Last Name, Email, Phone, Address

- Email → Customer ID

- Phone → Customer ID

2. Account Table:

- Account Number → Type, Balance, Currency, Customer ID

3. Transaction Table:

- Transaction ID → Amount, Date, Account Number, Transaction Type

- Account Number → Transaction ID

4. Loan Table:

- Loan ID → Customer ID, Loan Type, Loan Amount, Interest Rate, Time, Start Date, End Date, Payment Frequency, Current Balance

- Customer ID → Loan ID

5. Branch Table:

- Branch ID → Password, Branch Name, Address, City, State/Province, Country, Postal Code

6. DebitCard Table:

- Card ID → Account Number, Card Number, Card Type, Issue Date, Expiry Date, PIN

- Account Number → Card ID

7. CreditCard Table:

- Credit Card ID → Account Number, Card Number, Card Type, Issue Date, Expiry Date, PIN, Credit Limit

- Account Number → Credit Card ID

8. InternetBanking Table:

- User ID → Customer ID, Username, Password, Email

- Customer ID → User ID

9. Loan\_Customer Table (M:N relationship):

- Loan ID, Customer ID → (No specific functional dependency specified)

10. Account\_Customer Table (M:N relationship):

- Account Number, Customer ID → (No specific functional dependency specified)

These functional dependencies help us understand how attributes are related within each table and are crucial for database normalization and query optimization.

# SQL Queries

# CREATE DATABASE BANK\_MANAGEMENT;

# USE BANK\_MANAGEMENT;

# -- Customer Table

# CREATE TABLE Customer (

# customer\_id INT PRIMARY KEY,

# name VARCHAR(255) NOT NULL,

# address TEXT NOT NULL,

# phone VARCHAR(20) NOT NULL,

# email VARCHAR(255) NOT NULL,

# date\_of\_birth DATE NOT NULL,

# registration\_date DATE NOT NULL,

# password VARCHAR(255) NOT NULL

# );

# -- Account Table

# CREATE TABLE Account (

# account\_number INT PRIMARY KEY,

# type VARCHAR(255) NOT NULL,

# balance DECIMAL(10, 2) NOT NULL,

# currency VARCHAR(3) NOT NULL,

# customer\_id INT,

# FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id)

# );

# -- Transaction Table

# CREATE TABLE Transaction1 (

# transaction\_id INT PRIMARY KEY,

# amount DECIMAL(10, 2) NOT NULL,

# date DATE NOT NULL,

# account\_number INT,

# transaction\_type VARCHAR(255) NOT NULL,

# FOREIGN KEY (account\_number) REFERENCES Account(account\_number)

# );

# -- Loan Table

# CREATE TABLE Loan (

# Loan\_ID INT PRIMARY KEY,

# Customer\_ID INT,

# Loan\_Type VARCHAR(255) NOT NULL,

# Loan\_Amount DECIMAL(10, 2) NOT NULL,

# Interest\_Rate DECIMAL(5, 2) NOT NULL,

# Time INT NOT NULL,

# Start\_Date DATE NOT NULL,

# End\_Date DATE NOT NULL,

# Payment\_Frequency VARCHAR(255) NOT NULL,

# Current\_Balance DECIMAL(10, 2) NOT NULL,

# FOREIGN KEY (Customer\_ID) REFERENCES Customer(customer\_id)

# );

# -- Branch Table

# CREATE TABLE Branch (

# BranchID INT PRIMARY KEY,

# Password VARCHAR(255) NOT NULL,

# BranchName VARCHAR(255) NOT NULL,

# Address TEXT NOT NULL,

# City VARCHAR(255) NOT NULL,

# State\_Province VARCHAR(255) NOT NULL,

# Country VARCHAR(255) NOT NULL,

# PostalCode VARCHAR(10) NOT NULL

# );

# -- DebitCard Table

# CREATE TABLE DebitCard (

# CardID INT PRIMARY KEY,

# AccountNumber INT,

# CardNumber VARCHAR(255) NOT NULL,

# CardType VARCHAR(255) NOT NULL,

# IssueDate DATE NOT NULL,

# ExpiryDate DATE NOT NULL,

# Pin VARCHAR(255) NOT NULL,

# FOREIGN KEY (AccountNumber) REFERENCES Account(account\_number)

# );

# -- CreditCard Table

# CREATE TABLE CreditCard (

# CreditCardID INT PRIMARY KEY,

# AccountNumber INT,

# CardNumber VARCHAR(255) NOT NULL,

# CardType VARCHAR(255) NOT NULL,

# IssueDate DATE NOT NULL,

# ExpiryDate DATE NOT NULL,

# Pin VARCHAR(255) NOT NULL,

# CreditLimit DECIMAL(10, 2) NOT NULL,

# FOREIGN KEY (AccountNumber) REFERENCES Account(account\_number)

# );

# -- InternetBanking Table

# CREATE TABLE InternetBanking (

# UserID INT PRIMARY KEY,

# CustomerID INT,

# Username VARCHAR(255) NOT NULL,

# Password VARCHAR(255) NOT NULL,

# Email VARCHAR(255) NOT NULL,

# FOREIGN KEY (CustomerID) REFERENCES Customer(customer\_id)

# );

# -- Account\_Customer Table

# CREATE TABLE Account\_Customer (

# Account\_Number INT,

# Customer\_ID INT,

# PRIMARY KEY (Account\_Number, Customer\_ID),

# FOREIGN KEY (Account\_Number) REFERENCES Account(account\_number),

# FOREIGN KEY (Customer\_ID) REFERENCES Customer(customer\_id)

# );

# -- Loan\_Customer Table

# CREATE TABLE Loan\_Customer (

# Loan\_ID INT,

# Customer\_ID INT,

# PRIMARY KEY (Loan\_ID, Customer\_ID),

# FOREIGN KEY (Loan\_ID) REFERENCES Loan(Loan\_ID),

# FOREIGN KEY (Customer\_ID) REFERENCES Customer(customer\_id)

# );

# --Insertionn here

# INSERT INTO Customer (customer\_id, name, address, phone, email, date\_of\_birth, registration\_date, password)

# VALUES

# (1, 'Aarav Sharma', '123, Main Street, Bangalore', '9876543210', 'aarav.sharma@example.com', '1990-05-15', '2022-01-01', 'password123'),

# (2, 'Ananya Gupta', '456, Park Avenue, Mumbai', '9876543211', 'ananya.gupta@example.com', '1992-08-20', '2022-01-02', 'password456'),

# (3, 'Aaradhya Singh', '789, Lake Road, Kolkata', '9876543212', 'aaradhya.singh@example.com', '1995-02-10', '2022-01-03', 'password789'),

# (4, 'Vihaan Kumar', '321, Hill View, Delhi', '9876543213', 'vihaan.kumar@example.com', '1993-11-25', '2022-01-04', 'password321'),

# (5, 'Saisha Patel', '654, Garden Street, Chennai', '9876543214', 'saisha.patel@example.com', '1994-06-30', '2022-01-05', 'password654'),

# (6, 'Advait Desai', '987, River Road, Pune', '9876543215', 'advait.desai@example.com', '1991-09-12', '2022-01-06', 'password987'),

# (7, 'Zara Khan', '741, Skyline Apartment, Hyderabad', '9876543216', 'zara.khan@example.com', '1996-04-18', '2022-01-07', 'password741'),

# (8, 'Aryan Joshi', '852, Green Valley, Ahmedabad', '9876543217', 'aryan.joshi@example.com', '1997-07-22', '2022-01-08', 'password852'),

# (9, 'Anika Reddy', '963, Ocean View, Kochi', '9876543218', 'anika.reddy@example.com', '1998-10-05', '2022-01-09', 'password963'),

# (10, 'Kabir Verma', '159, Hillcrest, Jaipur', '9876543219', 'kabir.verma@example.com', '1999-03-28', '2022-01-10', 'password159');

# INSERT INTO Account (account\_number, type, balance, currency, customer\_id)

# VALUES

# (101, 'Savings', 50000.00, 'INR', 1),

# (102, 'Current', 100000.00, 'INR', 2),

# (103, 'Savings', 75000.00, 'INR', 3),

# (104, 'Current', 60000.00, 'INR', 4),

# (105, 'Savings', 90000.00, 'INR', 5),

# (106, 'Current', 120000.00, 'INR', 6),

# (107, 'Savings', 80000.00, 'INR', 7),

# (108, 'Current', 110000.00, 'INR', 8),

# (109, 'Savings', 95000.00, 'INR', 9),

# (110, 'Current', 130000.00, 'INR', 10);

# INSERT INTO Transaction1 (transaction\_id, amount, date, account\_number, transaction\_type)

# VALUES

# (1, 500.00, '2022-01-01', 101, 'Credit'),

# (2, 1000.00, '2022-01-02', 102, 'Credit'),

# (3, 200.00, '2022-01-03', 103, 'Debit'),

# (4, 1500.00, '2022-01-04', 104, 'Credit'),

# (5, 300.00, '2022-01-05', 105, 'Debit'),

# (6, 600.00, '2022-01-06', 106, 'Credit'),

# (7, 400.00, '2022-01-07', 107, 'Credit'),

# (8, 800.00, '2022-01-08', 108, 'Debit'),

# (9, 1200.00, '2022-01-09', 109, 'Credit'),

# (10, 700.00, '2022-01-10', 110, 'Debit');

# INSERT INTO Loan (Loan\_ID, Customer\_ID, Loan\_Type, Loan\_Amount, Interest\_Rate, Time, Start\_Date, End\_Date, Payment\_Frequency, Current\_Balance)

# VALUES

# (1, 1, 'Home Loan', 5000000.00, 8.5, 20, '2022-01-01', '2042-01-01', 'Monthly', 4500000.00),

# (2, 2, 'Personal Loan', 200000.00, 12.0, 5, '2022-01-02', '2027-01-02', 'Quarterly', 150000.00),

# (3, 3, 'Car Loan', 800000.00, 9.0, 7, '2022-01-03', '2029-01-03', 'Monthly', 600000.00),

# (4, 4, 'Education Loan', 1000000.00, 7.5, 10, '2022-01-04', '2032-01-04', 'Monthly', 800000.00),

# (5, 5, 'Home Loan', 6000000.00, 8.75, 15, '2022-01-05', '2037-01-05', 'Monthly', 5000000.00),

# (6, 6, 'Personal Loan', 300000.00, 11.0, 3, '2022-01-06', '2025-01-06', 'Quarterly', 250000.00),

# (7, 7, 'Car Loan', 900000.00, 9.25, 5, '2022-01-07', '2027-01-07', 'Monthly', 700000.00),

# (8, 8, 'Education Loan', 1200000.00, 7.25, 8, '2022-01-08', '2030-01-08', 'Monthly', 1000000.00),

# (9, 9, 'Home Loan', 7000000.00, 8.0, 20, '2022-01-09', '2042-01-09', 'Monthly', 6000000.00),

# (10, 10, 'Personal Loan', 400000.00, 10.0, 4, '2022-01-10', '2026-01-10', 'Quarterly', 350000.00);

# INSERT INTO Branch (BranchID, Password, BranchName, Address, City, State\_Province, Country, PostalCode)

# VALUES

# (1, 'branchpass1', 'Main Branch', '1, Bank Road', 'Bangalore', 'Karnataka', 'India', '560001'),

# (2, 'branchpass2', 'Central Branch', '2, Central Street', 'Mumbai', 'Maharashtra', 'India', '400001'),

# (3, 'branchpass3', 'Downtown Branch', '3, Downtown Avenue', 'Kolkata', 'West Bengal', 'India', '700001'),

# (4, 'branchpass4', 'City Branch', '4, City Center', 'Delhi', 'Delhi', 'India', '110001'),

# (5, 'branchpass5', 'Metro Branch', '5, Metro Plaza', 'Chennai', 'Tamil Nadu', 'India', '600001'),

# (6, 'branchpass6', 'Urban Branch', '6, Urban Market', 'Pune', 'Maharashtra', 'India', '411001'),

# (7, 'branchpass7', 'Cosmopolitan Branch', '7, Cosmo Tower', 'Hyderabad', 'Telangana', 'India', '500001'),

# (8, 'branchpass8', 'Green City Branch', '8, Green City Complex', 'Ahmedabad', 'Gujarat', 'India', '380001'),

# (9, 'branchpass9', 'Coastal Branch', '9, Coastal View', 'Kochi', 'Kerala', 'India', '682001'),

# (10, 'branchpass10', 'Royal Branch', '10, Royal Road', 'Jaipur', 'Rajasthan', 'India', '302001');

# INSERT INTO DebitCard (CardID, AccountNumber, CardNumber, CardType, IssueDate, ExpiryDate, Pin)

# VALUES

# (1, 101, '1234567812345678', 'Visa', '2022-01-01', '2025-12-31', '1234'),

# (2, 102, '2345678923456789', 'MasterCard', '2022-01-02', '2026-12-31', '2345'),

# (3, 103, '3456789034567890', 'Visa', '2022-01-03', '2027-12-31', '3456'),

# (4, 104, '4567890145678901', 'MasterCard', '2022-01-04', '2028-12-31', '4567'),

# (5, 105, '5678901256789012', 'Visa', '2022-01-05', '2029-12-31', '5678'),

# (6, 106, '6789012367890123', 'MasterCard', '2022-01-06', '2030-12-31', '6789'),

# (7, 107, '7890123478901234', 'Visa', '2022-01-07', '2031-12-31', '7890'),

# (8, 108, '8901234589012345', 'MasterCard', '2022-01-08', '2032-12-31', '8901'),

# (9, 109, '9012345690123456', 'Visa', '2022-01-09', '2033-12-31', '9012'),

# (10, 110, '0123456701234567', 'MasterCard', '2022-01-10', '2034-12-31', '0123');

# INSERT INTO CreditCard (CreditCardID, AccountNumber, CardNumber, CardType, IssueDate, ExpiryDate, Pin, CreditLimit)

# VALUES

# (1, 101, '1111222233334444', 'Visa', '2022-01-01', '2025-12-31', '1234', 50000.00),

# (2, 102, '2222333344445555', 'MasterCard', '2022-01-02', '2026-12-31', '2345', 100000.00),

# (3, 103, '3333444455556666', 'Visa', '2022-01-03', '2027-12-31', '3456', 75000.00),

# (4, 104, '4444555566667777', 'MasterCard', '2022-01-04', '2028-12-31', '4567', 60000.00),

# (5, 105, '5555666677778888', 'Visa', '2022-01-05', '2029-12-31', '5678', 90000.00),

# (6, 106, '6666777788889999', 'MasterCard', '2022-01-06', '2030-12-31', '6789', 120000.00),

# (7, 107, '7777888899990000', 'Visa', '2022-01-07', '2031-12-31', '7890', 80000.00),

# (8, 108, '8888999900001111', 'MasterCard', '2022-01-08', '2032-12-31', '8901', 110000.00),

# (9, 109, '9999000011112222', 'Visa', '2022-01-09', '2033-12-31', '9012', 95000.00),

# (10, 110, '0000111122223333', 'MasterCard', '2022-01-10', '2034-12-31', '0123', 130000.00);

# INSERT INTO InternetBanking (UserID, CustomerID, Username, Password, Email)

# VALUES

# (1, 1, 'aaravsharma', 'internetpass1', 'aarav.sharma@example.com'),

# (2, 2, 'ananyagupta', 'internetpass2', 'ananya.gupta@example.com'),

# (3, 3, 'aaradhyasingh', 'internetpass3', 'aaradhya.singh@example.com'),

# (4, 4, 'vihaankumar', 'internetpass4', 'vihaan.kumar@example.com'),

# (5, 5, 'saishapatel', 'internetpass5', 'saisha.patel@example.com'),

# (6, 6, 'advaitdesai', 'internetpass6', 'advait.desai@example.com'),

# (7, 7, 'zarakhan', 'internetpass7', 'zara.khan@example.com'),

# (8, 8, 'aryanjoshi', 'internetpass8', 'aryan.joshi@example.com'),

# (9, 9, 'anikareddy', 'internetpass9', 'anika.reddy@example.com'),

# (10, 10, 'kabirverma', 'internetpass10', 'kabir.verma@example.com');

# INSERT INTO Account\_Customer (Account\_Number, Customer\_ID)

# VALUES

# (101, 1),

# (102, 2),

# (103, 3),

# (104, 4),

# (105, 5),

# (106, 6),

# (107, 7),

# (108, 8),

# (109, 9),

# (110, 10);

# INSERT INTO Loan\_Customer (Loan\_ID, Customer\_ID)

# VALUES

# (1, 1),

# (2, 2),

# (3, 3),

# (4, 4),

# (5, 5),

# (6, 6),

# (7, 7),

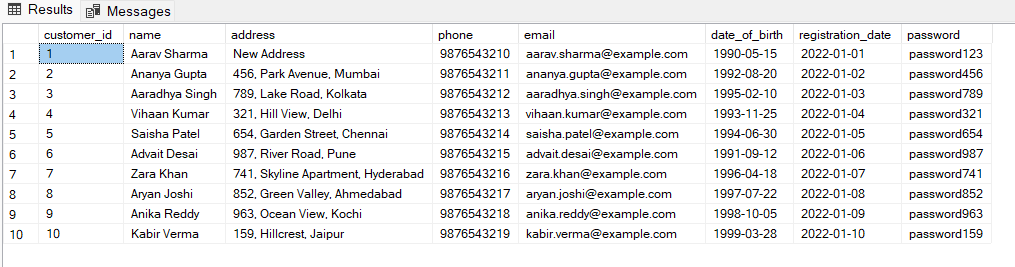
# (8, 8),

# (9, 9),

# (10, 10);

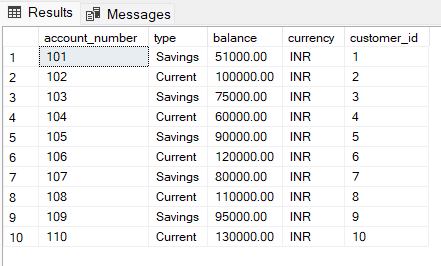
-- 1. Select all customers

SELECT \* FROM Customer;



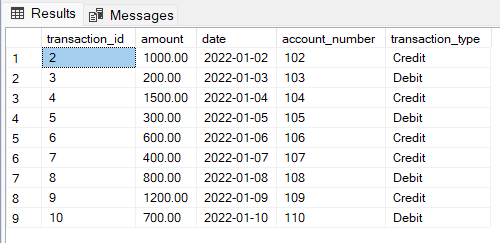
-- 2. Select all accounts

SELECT \* FROM Account;



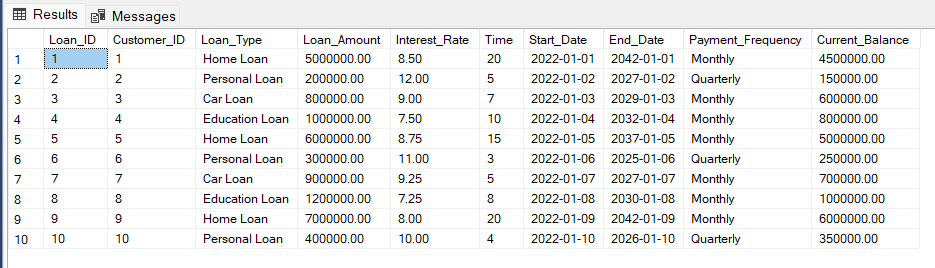
-- 3. Select all transactions

SELECT \* FROM Transaction1;



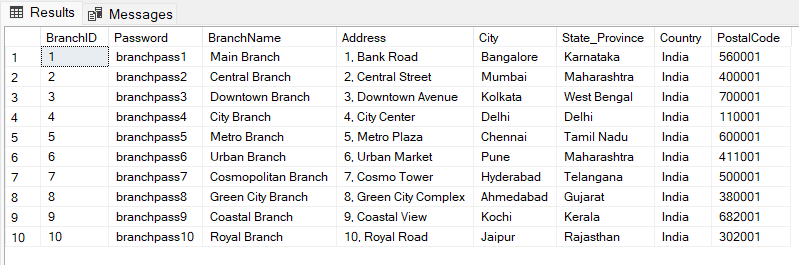
-- 4. Select all loans

SELECT \* FROM Loan;



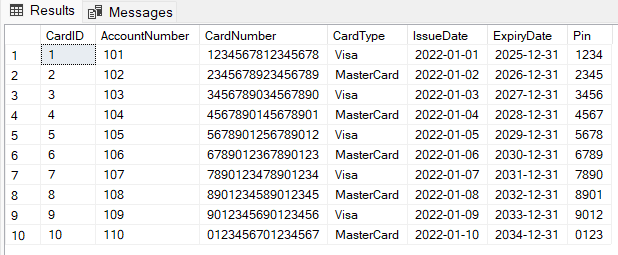
-- 5. Select all branches

SELECT \* FROM Branch;



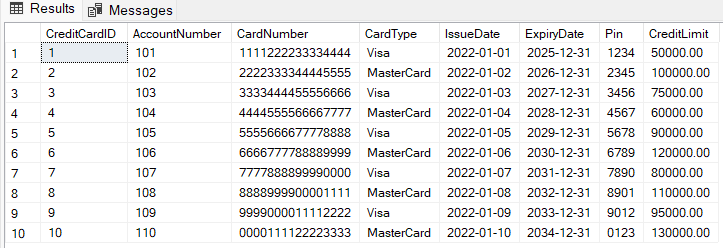
-- 6. Select all debit cards

SELECT \* FROM DebitCard;



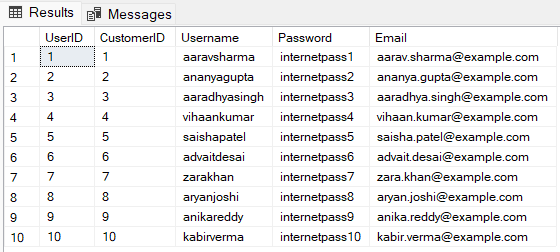
-- 7. Select all credit cards

SELECT \* FROM CreditCard;



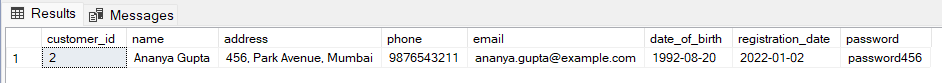
-- 8. Select all internet banking accounts

SELECT \* FROM InternetBanking;



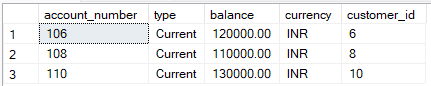
-- 9. Select all customers from a specific city (e.g., Bangalore)

SELECT \* FROM Customer WHERE address LIKE '%Mumbai%';



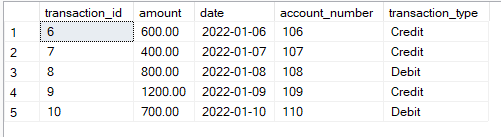
-- 10. Select all accounts with a balance greater than 100000

SELECT \* FROM Account WHERE balance > 100000;



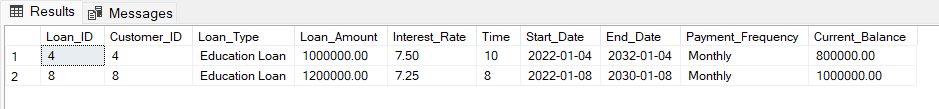
-- 11. Select all transactions after a certain date (e.g., '2022-01-05')

SELECT \* FROM Transaction1 WHERE date > '2022-01-05';



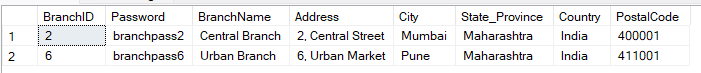
-- 12. Select all loans with an interest rate less than 8.0

SELECT \* FROM Loan WHERE Interest\_Rate < 8.0;



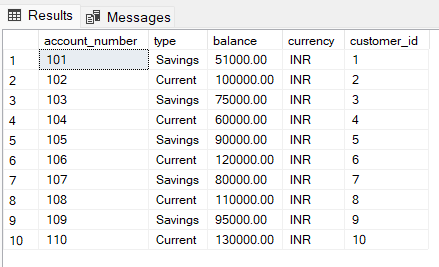
-- 13. Select all branches in a specific state (e.g., Maharashtra)

SELECT \* FROM Branch WHERE State\_Province = 'Maharashtra';



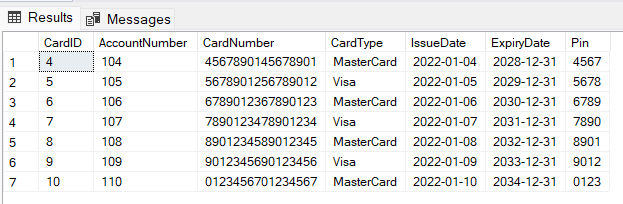
-- 14. Select all accounts with a specific currency (e.g., INR)

SELECT \* FROM Account WHERE currency = 'INR';



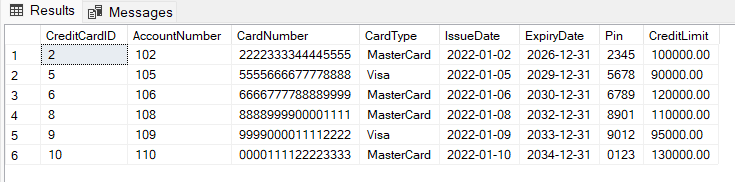
-- 15. Select all debit cards issued after a certain date (e.g., '2022-01-03')

SELECT \* FROM DebitCard WHERE IssueDate > '2022-01-03';



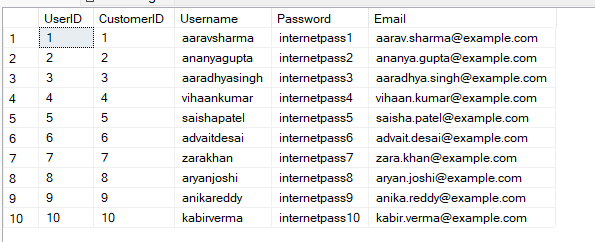
-- 16. Select all credit cards with a credit limit greater than 80000

SELECT \* FROM CreditCard WHERE CreditLimit > 80000;



-- 17. Select all internet banking accounts with a specific email domain (e.g., 'example.com')

SELECT \* FROM InternetBanking WHERE Email LIKE '%example.com';



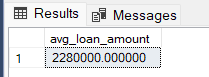
-- 18. Select the total balance of all accounts

SELECT SUM(balance) AS total\_balance FROM Account;



-- 19. Select the average loan amount

SELECT AVG(Loan\_Amount) AS avg\_loan\_amount FROM Loan;

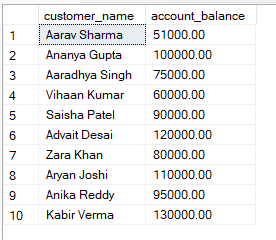


-- 20. Select the customer name and account balance for each account

SELECT c.name AS customer\_name, a.balance AS account\_balance

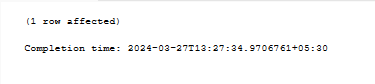
FROM Customer c

JOIN Account a ON c.customer\_id = a.customer\_id;



-- 21. Update the address of a customer (e.g., customer\_id = 1)

UPDATE Customer SET address = 'New Address' WHERE customer\_id = 1;



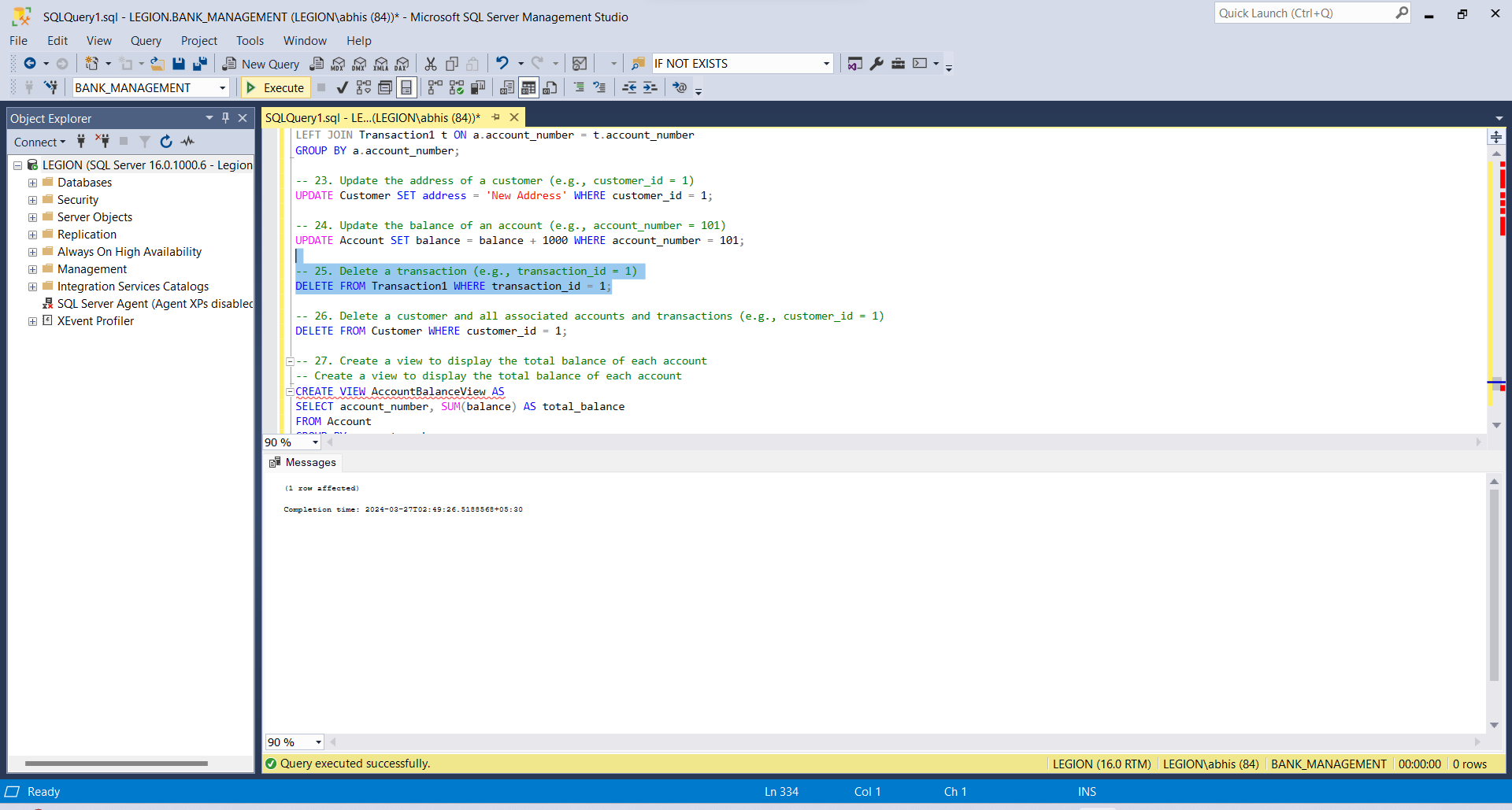
-- 22. Update the balance of an account (e.g., account\_number = 101)

UPDATE Account SET balance = balance + 1000 WHERE account\_number = 101;



-- 23. Delete a transaction (e.g., transaction\_id = 1)

DELETE FROM Transaction1 WHERE transaction\_id = 1;



-- 24. Select the customer name, account number, and total balance for each customer

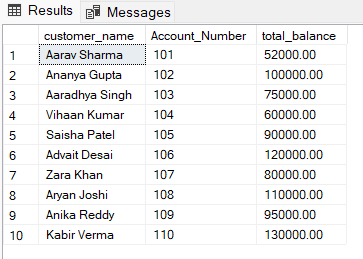
SELECT c.name AS customer\_name, ac.Account\_Number, SUM(a.balance) AS total\_balance

FROM Customer c

JOIN Account\_Customer ac ON c.customer\_id = ac.Customer\_ID

JOIN Account a ON ac.Account\_Number = a.account\_number

GROUP BY c.name, ac.Account\_Number;

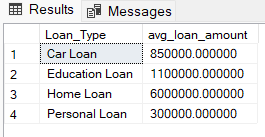


-- 25. Show the average loan amount for each loan type

SELECT Loan\_Type, AVG(Loan\_Amount) AS avg\_loan\_amount

FROM Loan

GROUP BY Loan\_Type;

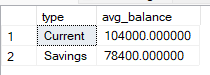


--26 Show the average balance for each type of account

SELECT type, AVG(balance) AS avg\_balance

FROM Account

GROUP BY type;



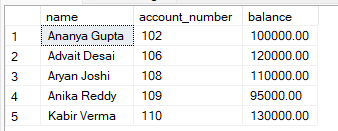
-- 27. Display the customer name, account number, and balance for accounts with a balance greater than the average balance of all accounts

SELECT c.name, a.account\_number, a.balance

FROM Account a

JOIN Customer c ON a.customer\_id = c.customer\_id

WHERE a.balance > (SELECT AVG(balance) FROM Account);

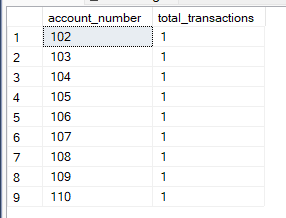


-- 28. Show the total number of transactions for each account

SELECT account\_number, COUNT(\*) AS total\_transactions

FROM Transaction1

GROUP BY account\_number;



-- 29. Show the customer names and the total number of transactions they have made

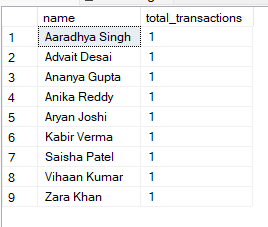
SELECT c.name, COUNT(t.transaction\_id) AS total\_transactions

FROM Customer c

JOIN Account a ON c.customer\_id = a.customer\_id

JOIN Transaction1 t ON a.account\_number = t.account\_number

GROUP BY c.name;

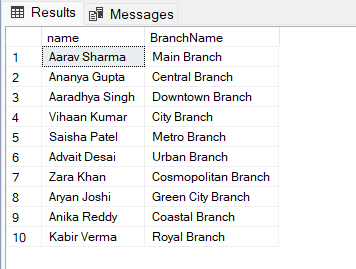


-- 30. Show the customer names and their corresponding branch names

SELECT c.name, b.BranchName

FROM Customer c

JOIN Branch b ON c.customer\_id = b.BranchID;



# ​Project demonstration

Our banking system project utilizes MySQL for database management. In a typical project demonstration, we would showcase the database design, data manipulation, and querying capabilities. This would include presenting the Entity Relationship Diagram (ERD), discussing the normalization process, and demonstrating various SQL queries for data retrieval and manipulation.

# ​Self -Learning beyond classroom

# As a team, we've gone beyond our classroom studies to learn more about managing databases. We've focused on making databases work faster by improving how we write and organize our data. We've also learned about keeping data safe and making sure it stays correct and reliable. This extra learning has helped us understand databases better and be ready for real-world projects.

# In our exploration, we've discovered ways to optimize databases for speed and efficiency. This includes learning about indexing, which helps us find data quickly, and how to structure data so it's easier to work with. We've also learned about security measures to protect data from unauthorized access and techniques to ensure data is stored accurately.

# Our self-learning journey has equipped us with practical skills that directly benefit our project. We're now more proficient in database management, able to create efficient databases that are secure and reliable. This additional knowledge has not only enhanced our understanding but also prepared us for the challenges of working with databases in the professional world.

# Learning from the Project

Our database project taught us a lot about how to design databases well and manage them effectively. We learned about important concepts like normalization, which helps keep data organized, and indexing, which helps us find information quickly. These skills are essential for creating databases that work smoothly and store data correctly.

Besides technical skills, the project also taught us about project management. We learned how to plan tasks, manage our time effectively, and communicate with our team. These skills are important for any project, not just database projects, and will help us in our future work.

In summary, our database project was a great learning experience. It helped us improve our database skills and also taught us valuable project management skills. We're now better prepared to work on similar projects in the future and handle the challenges they bring.

# Challenges Faced

For our database management project, we encountered several challenges related to normalization, creating the ER diagram, and implementing the relational model due to the project's complexity. Normalization involves organizing data to minimize redundancy and dependency, which was challenging given the vast amount of data we had to manage.

Creating the ER diagram was also challenging because it required us to accurately represent the relationships between different entities in our database. This was crucial for designing an efficient and well-structured database.

Additionally, implementing the relational model posed its own challenges, as it required us to translate the ER diagram into a functional database schema. This involved careful planning and attention to detail to ensure that the database met the project requirements and was able to efficiently store and retrieve data.

Furthermore, creating the graphical user interface (GUI) using Tkinter in Python presented its own set of challenges. We had to design a user-friendly interface that allowed users to interact with the database easily while also ensuring that it was visually appealing and responsive.

Overall, overcoming these challenges required a combination of technical expertise, problem-solving skills, and collaboration within our team. It was a valuable learning experience that helped us develop a deeper understanding of database management and software development.

# Conclusion

In summary, our project was a big learning experience. We faced challenges with organizing our data, creating diagrams, and building the database. Despite these challenges, we learned a lot about how databases work and how to manage them well. We also learned about teamwork and planning, which are important skills in any job. Overall, this project has helped us grow and become better at what we do.