Paul and Virginia Engler College of Business

West Texas A&M University

**Finance Industry Database**

Deliverable 2

By: **Farzaneh Noroozi**

Professor: **Dr. Abraham Abby Sen**

Fall Semester 2023

**Chapter 1**

**Introduction**

The finance industry is a dynamic and data-intensive sector that relies heavily on the effective management of vast amounts of information. In this Data & Information Management project, we will explore the challenges and opportunities within the finance industry and propose a database solution to address the data management problems it faces. Our chosen area of application is the finance sector, encompassing banking, investment, and financial services.

**1- Data Management Problems in the Finance Industry:** The finance industry has encountered several data management problems over the years, leading to inefficiencies, errors, and regulatory compliance issues. Some of the prominent challenges include:

1. Data Silos: Financial institutions often have fragmented data stored in various systems and databases, making it difficult to access and consolidate information for accurate decision-making.
2. Data Security: With the increasing frequency of cyberattacks, protecting sensitive financial data has become a paramount concern. Data breaches can lead to financial losses and reputational damage.
3. Compliance and Regulatory Reporting: The finance industry faces stringent regulatory requirements, and ensuring compliance is a complex and resource-intensive process. Inaccurate or incomplete data can result in costly penalties.
4. Data Accuracy and Integrity: Inaccurate data can lead to erroneous financial analyses and investment decisions, potentially causing substantial financial losses.

**2- Motivation for Our Database Solution:** Our proposed database aims to address these data management problems by offering features such as data integrity, reduced data redundancy, centralized data storage, and robust security protocols. By leveraging these qualities, we intend to mitigate the aforementioned challenges:

1. Data Integrity: Our database will implement data validation rules and integrity constraints to ensure that only accurate and reliable data is stored, reducing the risk of errors and misreporting.
2. Reduced Data Redundancy: By providing a single source of truth and implementing normalization techniques, our database will minimize data redundancy, thereby enhancing data consistency and reliability.
3. Enhanced Data Security: Robust security measures, including encryption and access controls, will be implemented to safeguard financial data and protect against cyber threats.

**3- Potential Benefits:** The implementation of our database in the finance industry can yield several benefits:

1. Improved Decision-Making: Access to reliable and consolidated data will enable financial professionals to make informed and timely decisions, reducing the risk of financial losses.
2. Efficient Regulatory Compliance: Our database will streamline compliance efforts by providing a comprehensive view of relevant data, simplifying reporting, and ensuring adherence to regulatory requirements.
3. Cost Reduction: Reduced data redundancy and improved data quality can lead to cost savings associated with data storage and correction of errors.

4- **Potential Users:** The potential users of our database in the finance industry include:

1. Financial Analysts: They can benefit from accurate and up-to-date financial data for investment analysis and risk assessment.
2. Compliance Officers: Our database can assist in simplifying compliance processes, ensuring adherence to financial regulations.
3. Executives and Managers: Access to real-time financial data will aid in strategic planning and decision-making.
4. IT and Security Teams: They will be responsible for maintaining and securing the database to protect against data breaches.

In addition to the challenges mentioned earlier, our database solution can also address emerging issues in the finance industry, such as the growing demand for data analytics and the need for faster data retrieval to support high-frequency trading. By offering a robust and integrated data management solution, we aim to empower financial professionals to thrive in this competitive and data-driven sector.

**Chapter 2**

**Business Rules and User Requirements for the Finance Industry Database**

As the owner of our finance business, it's imperative to provide clear guidance to our hired database developer to ensure that our database meets our specific needs. In this section, I'll outline the key business rules and user requirements that will shape our database.

**Entities and Attributes:**

1. **Customer/Client:**
   * We need to track customer information, including their identification, contact details, and social security number (SSN) for compliance purposes.
     + Customer ID (Unique Identifier)
     + Name (Text)
     + Date of Enrolment (Date/Time)
     + Date of Birth (Date/Time)
     + Contact Information (Phone, Email, Address) (Text)
     + SSN (Text)
2. **Accounts:**
   * Accounts are fundamental to our business. We want to store information about different account types, balances, and opening dates.
     + Account ID (Unique Identifier)
     + Account Type (Text)
     + Balance (Numeric)
     + Opening Date (Date/Time)
   * Each account is linked to a specific customer.
3. **Transactions:**
   * We need to record all financial transactions for auditing and analysis. These include details like transaction type, amount, and date.
     + Transaction ID (Unique Identifier)
     + Transaction Type (Text)
     + Amount (Numeric)
     + Transaction Date and Time (Date/Time)
   * Each transaction is associated with a specific account.
4. **Financial Products:**
   * To manage investments, we need to track various financial products such as stocks and bonds.
     + Product ID (Unique Identifier)
     + Product Type (Text)
     + Product Name (Text)
     + Price (Numeric)
   * These products can be linked to multiple accounts.
5. **Employees:**
   * Employee ID (Unique Identifier)
   * Name (Text)
   * Date of Birth (Date/Time)
   * Contact Information (Phone, Email, Address) (Text)
   * Position (Text)
   * Salary (Numeric)
   * Hire Date (Date/Time)
6. **Loans:**
   * Loan ID (Unique Identifier)
   * Loan Type (Text)
   * Loan Amount (Numeric)
   * Interest Rate (Numeric)
   * Loan Term (Numeric)
   * Approval Date (Date/Time)
   * Customer ID (Foreign Key, linking to Customer entity)
   * Account ID (Foreign Key, linking to Account entity)
7. **Credit Scores:**
   * Credit Score ID (Unique Identifier)
   * Customer ID (Foreign Key, linking to Customer entity)
   * Score (Numeric)
   * Score Date (Date/Time)

**Relationships and Cardinalities:**

* Loyal customer - customer Relationship (Unary)
* Customer-Account Relationship (Binary):
  + Each customer can have multiple accounts.
  + Each account belongs to one customer.
  + Cardinality: One-to-Many (1 Customer to Many Accounts).
* Account-Transaction Relationship (Binary):
  + Each account can have multiple transactions.
  + Each transaction is associated with one account.
  + Cardinality: One-to-Many (1 Account to Many Transactions).
* Record Relationship (Ternary):
  + This is a ternary relationship involving Financial Products, Accounts, and Customers.
  + Each financial product can be linked to multiple accounts.
  + Each account can have multiple financial products.
  + Each financial product can belong to multiple customers.
* Employees-Customer Relationship (Binary):
* Each employee can be associated with multiple customers (e.g., for customer service or relationship management).
* Each customer may be assisted by one or more employees.
* Cardinality: Many-to-Many (Many Employees to Many Customers).
* Loans-Customer Relationship (Binary):
* Each customer can have multiple loans (e.g., mortgages, personal loans).
* Each loan is associated with one customer.
* Cardinality: One-to-Many (1 Customer to Many Loans).
* Credit Scores-Customer Relationship (Binary):
* Each customer can have multiple credit scores, reflecting their credit history over time.
* Each credit score is associated with one customer.
* Cardinality: One-to-Many (1 Customer to Many Credit Scores).
* Account-Loans Relationship (Binary):
* Each Account can have multiple Loans.
* Each Loan is associated with one customer.
* Cardinality: One-to-Many (1 Account to Many Loans).

**Business Rules:**

1. Each Customer after one year of enrolment becomes a loyal customer.
2. Each customer must have a unique Customer ID.
3. Protect the privacy of our customers by securely storing or encrypting SSNs.
4. Ensure the uniqueness of Account IDs.
5. All transactions must be associated with a valid account.
6. Financial products can be linked to multiple accounts to support our investment offerings.
7. Implement data validation rules, like minimum balance requirements for specific account types, to maintain data accuracy.
8. Apply robust access controls to safeguard sensitive financial data.
9. Log and audit all transactions for compliance and security.
10. Ensure that all database operations adhere to relevant financial regulations and industry standards.
11. Each customer is assigned to at least one employee for personalized service. This rule promotes a better customer experience and strengthens customer relationships.
12. Define criteria for loan approval based on attributes like credit scores, loan type, and loan amount. Ensure that loans are approved in compliance with these criteria to manage risk effectively.
13. Regularly update and record credit scores for each customer. Implement alerts for significant changes in credit scores to identify potential issues early.
14. Enforce rules for timely loan repayments and consider automation for reminders and penalties in case of late payments.
15. Implement access controls to ensure that employees can only view and modify customer data necessary for their job roles, protecting customer privacy and data security.

A diagram of a credit score

Description automatically generated

**Chapter 3**

**Enhanced Entity-Relationship Diagram (EERD)**

1. **Entities:**
   * Customer
   * Account
   * Transaction
   * Financial Product
   * Record
   * Employees
   * Loans
   * Credit Scores
2. **Relationships:**

* **Unary Relation:**
  + Loyal Customer with Customer
* **Binary Relation:**
  + Customer- Account Relationship
  + Account-Transaction Relationship
  + Employees-Customer Relationship
  + Loans-Customer Relationship
  + Credit Scores-Customer Relationship
  + Account-Loans Relationship
* **Ternary Relation:**
  + Product ID, Customer ID, and Account ID.

A diagram of a credit score

Description automatically generated

# Create Customers table

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(255),

DateOfEnrollment DATETIME,

DateOfBirth DATETIME,

Phone VARCHAR(20), -- Add Phone

Email VARCHAR(255), -- Add Email

Address TEXT, -- Add Address

SSN TEXT,

loyal\_customer\_id INT,

FOREIGN KEY (loyal\_customer\_id) REFERENCES Customers(CustomerID)

);

#Insert 10 records into the Customers table

INSERT INTO Customers (CustomerID, Name, DateOfEnrollment, DateOfBirth, Phone, Email, Address, SSN, loyal\_customer\_id)

VALUES

(1, 'John Doe', '2022-01-01', '1980-05-15', '123-456-7890', 'john@example.com', '123 Main St', '123-45-6789', NULL),

(2, 'Jane Smith', '2022-01-02', '1985-03-20', '987-654-3210', 'jane@example.com', '456 Elm St', '987-65-4321', NULL),

(3, 'David Johnson', '2022-03-15', '1990-08-10', '555-555-5555', 'david@example.com', '789 Oak St', '555-55-5555', NULL),

(4, 'Sarah Wilson', '2022-03-20', '1987-12-05', '111-222-3333', 'sarah@example.com', '456 Pine St', '111-22-3333', NULL),

(5, 'Michael Brown', '2022-04-10', '1982-04-18', '999-888-7777', 'michael@example.com', '789 Maple St', '999-88-7777', NULL),

(6, 'Emily Davis', '2022-04-20', '1995-01-22', '444-666-8888', 'emily@example.com', '123 Elm St', '444-66-8888', NULL),

(7, 'Daniel Lee', '2022-05-05', '1989-11-03', '777-999-4444', 'daniel@example.com', '456 Oak St', '777-99-4444', NULL),

(8, 'Olivia White', '2022-05-10', '1984-07-15', '333-222-1111', 'olivia@example.com', '789 Cedar St', '333-22-1111', NULL),

(9, 'William Johnson', '2022-06-01', '1981-03-30', '222-333-4444', 'william@example.com', '123 Cedar St', '222-33-4444', NULL),

(10, 'Sophia Martin', '2022-06-10', '1993-09-25', '666-333-1111', 'sophia@example.com', '456 Pine St', '666-33-1111', NULL);

#Insert 10 records into the LoyalCustomers table

INSERT INTO LoyalCustomers (LoyalCustomerID, CustomerID, EnrollmentDate)

VALUES

(1, 1, '2023-01-01'),

(2, 2, '2023-02-01'),

(3, 3, '2023-03-01'),

(4, 4, '2023-04-01'),

(5, 5, '2023-05-01'),

(6, 6, '2023-06-01'),

(7, 7, '2023-07-01'),

(8, 8, '2023-08-01'),

(9, 9, '2023-09-01'),

(10, 10, '2023-10-01');

#Create Accounts table

CREATE TABLE Accounts (

AccountID INT PRIMARY KEY,

AccountType VARCHAR(50),

Balance NUMERIC(12, 2),

OpeningDate DATETIME,

CustomerID INT,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

#Insert 10 records into the Accounts table

INSERT INTO Accounts (AccountID, AccountType, Balance, OpeningDate, CustomerID)

VALUES

(1, 'Savings', 5000.00, '2022-01-01', 1),

(2, 'Checking', 2500.00, '2022-01-01', 2),

(3, 'Savings', 8000.00, '2022-02-15', 3),

(4, 'Checking', 3500.00, '2022-02-15', 4),

(5, 'Savings', 12000.00, '2022-03-20', 5),

(6, 'Checking', 6000.00, '2022-03-20', 6),

(7, 'Savings', 7000.00, '2022-04-10', 7),

(8, 'Checking', 4000.00, '2022-04-10', 8),

(9, 'Savings', 9500.00, '2022-05-01', 9),

(10, 'Checking', 5500.00, '2022-05-01', 10);

#Create Transactions table

CREATE TABLE Transactions (

TransactionID INT PRIMARY KEY,

TransactionType VARCHAR(50),

Amount NUMERIC(12, 2),

TransactionDateTime DATETIME,

AccountID INT,

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

#Insert 10 records into the Transactions table

INSERT INTO Transactions (TransactionID, TransactionType, Amount, TransactionDateTime, AccountID)

VALUES

(1, 'Deposit', 1000.00, '2022-01-03 09:00:00', 1),

(2, 'Withdrawal', 500.00, '2022-01-04 15:30:00', 2),

(3, 'Deposit', 1500.00, '2022-02-16 14:15:00', 3),

(4, 'Withdrawal', 750.00, '2022-02-17 10:45:00', 4),

(5, 'Deposit', 2000.00, '2022-03-21 12:00:00', 5),

(6, 'Withdrawal', 1000.00, '2022-03-21 16:30:00', 6),

(7, 'Deposit', 2500.00, '2022-04-11 08:30:00', 7),

(8, 'Withdrawal', 1200.00, '2022-04-12 17:15:00', 8),

(9, 'Deposit', 3000.00, '2022-05-02 11:00:00', 9),

(10, 'Withdrawal', 1500.00, '2022-05-03 15:45:00', 10);

#Create FinancialProducts table

CREATE TABLE FinancialProducts (

ProductID INT PRIMARY KEY,

ProductType VARCHAR(50),

ProductName VARCHAR(255),

Price NUMERIC(12, 2)

);

#Insert 10 records into the FinancialProducts table

INSERT INTO FinancialProducts (ProductID, ProductType, ProductName, Price)

VALUES

(1, 'Stock', 'TechStock', 100.00),

(2, 'Bond', 'GovernmentBond', 1000.00),

(3, 'Stock', 'PharmaStock', 80.00),

(4, 'Stock', 'AutoStock', 120.00),

(5, 'Bond', 'CorporateBond', 800.00),

(6, 'Stock', 'EnergyStock', 110.00),

(7, 'Stock', 'RetailStock', 90.00),

(8, 'Bond', 'MunicipalBond', 950.00),

(9, 'Stock', 'FinanceStock', 105.00),

(10, 'Stock', 'TelecomStock', 70.00);

# Create Employees table

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(255),

DateOfBirth DATETIME,

Phone VARCHAR(20),

Email VARCHAR(255),

Address TEXT,

Position VARCHAR(50),

Salary NUMERIC(12, 2),

HireDate DATETIME

);

#Insert 10 records into the Employees table

INSERT INTO Employees (EmployeeID, Name, DateOfBirth, Phone, Email, Address, Position, Salary, HireDate)

(1, 'Alice Johnson', '1988-06-10', '555-555-5555, alice@example.com, 789 Oak St', 'Customer Service', 60000.00, '2021-03-15'),

(2, 'Bob Smith', '1990-09-25', '666-666-6666, bob@example.com, 456 Pine St', 'Financial Advisor', 75000.00, '2021-04-01'),

(3, 'Eva Brown', '1985-03-20', '777-777-7777, eva@example.com, 123 Main St', 'Loan Officer', 70000.00, '2021-04-15'),

(4, 'Max Davis', '1980-05-15', '888-888-8888, max@example.com, 456 Elm St', 'Investment Analyst', 85000.00, '2021-05-01'),

(5, 'Grace Wilson', '1989-11-03', '999-999-9999, grace@example.com, 789 Maple St', 'Branch Manager', 90000.00, '2021-05-15'),

(6, 'Oliver Lee', '1981-03-30', '111-111-1111, oliver@example.com, 123 Cedar St', 'Customer Service', 60000.00, '2021-06-01'),

(7, 'Mia Martin', '1984-07-15', '222-222-2222, mia@example.com, 456 Oak St', 'Financial Advisor', 75000.00, '2021-06-15'),

(8, 'Noah White', '1993-09-25', '333-333-3333, noah@example.com, 789 Cedar St', 'Loan Officer', 70000.00, '2021-07-01'),

(9, 'Sophia Johnson', '1995-01-22', '444-444-4444, sophia@example.com, 123 Elm St', 'Investment Analyst', 85000.00, '2021-07-15'),

(10, 'Liam Smith', '1982-04-18', '123-123-1234, liam@example.com, 456 Pine St', 'Branch Manager', 90000.00, '2021-08-01');

#Create EmployeesCustomers table (to represent the many-to-many relationship)

CREATE TABLE EmployeesCustomers (

EmployeeID INT,

CustomerID INT,

FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

#Insert 10 records into the EmployeesCustomers table (sample data)

INSERT INTO EmployeesCustomers (EmployeeID, CustomerID)

VALUES

(1, 1),

(1, 2),

(2, 3),

(2, 4),

(3, 5),

(3, 6),

(4, 7),

(4, 8),

(5, 9),

(5, 10);

#Create Loans table

CREATE TABLE Loans (

LoanID INT PRIMARY KEY,

LoanType VARCHAR(50),

LoanAmount NUMERIC(12, 2),

InterestRate NUMERIC(5, 2),

LoanTerm INT,

ApprovalDate DATETIME,

CustomerID INT,

AccountID INT,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

#Insert 10 records into the Loans table (sample data)

INSERT INTO Loans (LoanID, LoanType, LoanAmount, InterestRate, LoanTerm, ApprovalDate, CustomerID, AccountID)

VALUES

(1, 'Mortgage', 250000.00, 3.5, 30, '2022-01-15', 1, 1),

(2, 'Personal', 10000.00, 7.5, 5, '2022-02-01', 2, 2),

(3, 'Auto', 15000.00, 4.0, 4, '2022-03-15', 3, 3),

(4, 'Mortgage', 300000.00, 3.25, 30, '2022-04-01', 4, 4),

(5, 'Personal', 12000.00, 8.0, 6, '2022-05-15', 5, 5),

(6, 'Auto', 18000.00, 4.25, 4, '2022-06-01', 6, 6),

(7, 'Mortgage', 200000.00, 3.75, 30, '2022-07-15', 7, 7),

(8, 'Personal', 9000.00, 7.0, 5, '2022-08-01', 8, 8),

(9, 'Auto', 16000.00, 4.5, 4, '2022-09-15', 9, 9),

(10, 'Mortgage', 280000.00, 3.6, 30, '2022-10-01', 10, 10);

#Create CreditScores table

CREATE TABLE CreditScores (

CreditScoreID INT PRIMARY KEY,

CustomerID INT,

Score NUMERIC(5, 2),

ScoreDate DATETIME,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

#Insert 10 records into the CreditScores table (sample data)

INSERT INTO CreditScores (CreditScoreID, CustomerID, Score, ScoreDate)

VALUES

(1, 1, 750.0, '2022-01-05'),

(2, 2, 720.0, '2022-02-01'),

(3, 3, 680.0, '2022-03-15'),

(4, 4, 760.0, '2022-04-01'),

(5, 5, 730.0, '2022-05-15'),

(6, 6, 710.0, '2022-06-01'),

(7, 7, 780.0, '2022-07-15'),

(8, 8, 740.0, '2022-08-01'),

(9, 9, 790.0, '2022-09-15'),

(10, 10, 770.0, '2022-10-01');

#Create the Record table (to represent the ternary relationship)

CREATE TABLE Record (

ProductID INT,

AccountID INT,

CustomerID INT,

FOREIGN KEY (ProductID) REFERENCES FinancialProducts(ProductID),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

#Insert 10 records into the Record table

INSERT INTO Record (ProductID, AccountID, CustomerID)

VALUES

(1, 1, 1),

(2, 2, 2),

(3, 3, 3),

(4, 4, 4),

(5, 5, 5),

(6, 6, 6),

(7, 7, 7),

(8, 8, 8),

(9, 9, 9),

(10, 10, 10);

#Customers Table:

-- Show table structure

DESC Customers;

-- Show table records

SELECT \* FROM Customers;

#LoyalCustomers Table:

-- Show table structure

DESC LoyalCustomers;

-- Show table records

SELECT \* FROM LoyalCustomers;

#Accounts Table:

-- Show table structure

DESC Accounts;

-- Show table records

SELECT \* FROM Accounts;

#Transactions Table:

-- Show table structure

DESC Transactions;

-- Show table records

SELECT \* FROM Transactions;

#FinancialProducts Table:

-- Show table structure

DESC FinancialProducts;

-- Show table records

SELECT \* FROM FinancialProducts;

#Employees Table:

-- Show table structure

DESC Employees;

-- Show table records

SELECT \* FROM Employees;

#EmployeesCustomers Table:

-- Show table structure

DESC EmployeesCustomers;

-- Show table records

SELECT \* FROM EmployeesCustomers;

#Loans Table:

-- Show table structure

DESC Loans;

-- Show table records

SELECT \* FROM Loans;

#CreditScores Table:

-- Show table structure

DESC CreditScores;

-- Show table records

SELECT \* FROM CreditScores;

#Record Table:

-- Show table structure

DESC Record;

-- Show table records

SELECT \* FROM Record;

**Chapter 5**

**SQL Code**

A screenshot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Chapter 6**

**DESC and SELECT Commands**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Chapter 7**

**SQL Queries**

1) Retrieve customer information for customers who have opened a savings account.

A screenshot of a computer

Description automatically generated

2) Find the top 3 financial products (by price) purchased by customers, and display the product name and price.

A screenshot of a computer

Description automatically generated

3) Find the loans taken by customers along with their personal information.

A screenshot of a computer

Description automatically generated

4) Retrieve employees who are financial advisors and their customers.

A screenshot of a computer

Description automatically generated

5) Get financial products that customers with a credit score above 750 have purchased.

A screenshot of a computer

Description automatically generated