
Assessment 2: Advanced Database - Bank Management System

Sachin: 23235298 — Yash Mehta: 23145127 — Mohammed Mufid Shaikh: 23228726^{* 1}

1. Introduction

The Bank Management System project is a very important case study for the financial services domain, especially in banking operations which are not limited to customer transactions and deposits & withdrawal functionalities but beyond it. As well as standard flexibility and performance concerns, the project has highlighted the need for secure and accurate data management a key factor in economic stability and customer trust governance. The database schema to supply a removable banking management system has been designed and adapted for a guarantee of unique customer identification, and data validation with enforcement business rules. It includes the core customer, account, loan, and service tables along with audit (for logging) and role/permission for security. It offers scalable, secure flexibility to operate and adapt to all of the intricate requirements unique in modern banking while also preparing for AI implementation along with its related scrutiny.

2. Domain Description

This project is concerned with the domain – financial services where banking operations represent a sizable portion. In the banking domain, everything is related to customers, accounts, branches card operations, transactions, loans, etc. It contains secure information and financial data and, therefore needs proper management the accurate records. A little bit about this domain is economic stability and customer trust. Further, we are entering the era of AI for which the sector is heavily regulated to avoid financial crimes ([Post, 2009](#)) .

3. Database analysis

The database schema supports a robust Banking Database Management System, ensuring unique customer identification, strict validation for email and age, and compliance with business rules like account ownership and transaction recording. Key tables include customers, accounts, loans, employees, and services, with an Audit table for security tracking and Role/Permission tables for access control. The structure facilitates efficient data management, regulatory compliance, and security while allowing flexibility in managing banking products and offers. This design provides a scalable, secure, and comprehensive solution for modern banking needs.

3.1. Business Situation

The bank has a large system in place to manage the financial activities of both its customers and employees. This system handles everything from customer accounts and loans to day-to-day transactions. The bank provides a variety of services, including account management, loan processing, and transaction handling. It is divided into several branches, each of which is designated by a special code. Every branch contains staff members with distinct responsibilities, including as managers, tellers, and loan officers. Each staff member has a position that limits their access to certain parts of the system. Each branch has at least one person in charge of operations who answers to the central office.

Customers have unique profiles that include their personal information, and they can have multiple accounts with the bank. Each account is linked to only one customer. Customers also interact with the bank's services through transactions, and they might receive special offers based on their profiles. The bank also manages ATMs, which are connected to branches, allowing customers to access their accounts conveniently. The system also keeps track of important details like audit logs, permissions for employees, and compliance with regulations. It ensures that transactions are secure, records are accurate, and that the bank follows all necessary rules. Overall, this system helps the bank operate smoothly while keeping customer data secure and making sure all financial activities are properly managed ([Hogan, 2018](#)).

3.2. Business Rules

3.2.1. CUSTOMER MANAGEMENT

- Unique Identification: An SSN and Customer ID must be assigned to each individual customer. The SSN is subject to security and privacy laws and is used for validation.
- Email validation: Every customer's Email_ID needs to be distinct and well formatted.
- Age Requirement: To open an account, a customer must be at least eighteen years old.

3.2.2. ACCOUNT AND TRANSACTIONS

- Account Ownership: A customer may have more than one account, but an account can only be linked to a single customer.
- Account Balance: Unless the account type expressly permits overdrafts, account balances cannot drop below zero.
- Transaction Recording: Each and every transaction needs to be documented using a distinct Transaction_ID that includes information about the type, date, and amount of the transaction.
- Transaction Limits: In accordance with regulatory compliance (such as anti-money laundering laws), transactions exceeding a specific threshold need to be noted and reported.

3.2.3. LOAN MANAGEMENT

- Loan Association: Each loan must be associated with one customer and one account.
- Interest Calculation: The interest on a loan must be computed using the terms that were agreed upon when the loan was created, and it must be updated on a regular basis.

3.2.4. EMPLOYEE AND USER ROLES

- Employee Roles: Using Role_ID, each employee is given a role that establishes their access levels and rights within the system.
- Access Control: Workers cannot access a customer's financial information without permission and may only access information that is pertinent to their job.

3.2.5. AUDIT AND SECURITY

- Audit Trail: Every modification to important data, such as account or customer details, needs to be recorded in the Audit table along with the person who did it and when.
- Data Security: Passwords and SSNs, among other sensitive data, need to be encrypted in the database.
- User Authentication: Before gaining access to the system, users must authenticate themselves using safe techniques (passwords, two-factor authentication, etc.).

3.2.6. SERVICE AND FEES

- Service Registration: A distinct Service ID must be assigned to each banking service that is provided and registered in the Service organization.
- Fee Application: Service-related fees have to be automatically applied in accordance with established guidelines (e.g., monthly maintenance fees for accounts)

3.2.7. MARKETING AND OFFERS

- Offer Eligibility: Offers can be associated with particular consumer segments and must have explicit eligibility requirements.
- Offer Expiration: Every offer has an End Date and a Start Date, after which it can no longer be applied.

3.2.8. COMPLIANCE AND REGULATORY

- **Data Retention:** As mandated by law, customer and transaction data must be kept in the database for a minimum amount of time (e.g., five years).
- **Regulatory Reporting:** The system needs to be able to generate reports that regulatory authorities need, including those that include client identity information and transactions that have been flagged.

3.2.9. BRANCH AND ATMs

- **Branch Identification:** Each branch must have a unique IFSC_code.
- **ATM Management:** Each ATM is linked to a branch and must be uniquely identified.

3.3. List of Entity and Attributes

3.3.1. ENTITIES

Within a relational database, an entity refers to a distinct object, concept, or occurrence from the real world that can be identified separately from other objects. Each entity is represented in the database as a table, where the name of the table reflects the type of entity, and each row within that table corresponds to an instance of the entity. Entities are key components in data modeling, helping to structure data into logical groupings that align with real-world scenarios.

Entities are typically divided into the following categories:

1. **Core Entities:** These are the main objects, like `Customer`, `Employee`, or `Product`, that play a crucial role in the business framework.
2. **Contextual Entities:** Entities such as `Departments`, `Branches`, and `Roles` fall into this category, providing essential structure and context to the core entities.
3. **Event-Based Entities:** These include entities like `Transactions`, `Orders`, or `Invoices`, which capture interactions or events related to the core entities (Palakurthi et al., 2015).

3.3.2. ATTRIBUTES

Attributes are the specific details or features that define and describe an entity. In the context of a database, these attributes are represented as columns within a table, where each column corresponds to a particular attribute, containing data that relates directly to the entity (Laender et al., 1994).

Attributes generally fall into these categories:

1. **Key Attributes:** These serve as unique identifiers for each instance of an entity, such as `Customer_ID` or `Employee_ID`. These are critical in databases and are typically used as primary keys.
2. **Descriptive Attributes:** These provide descriptive details about the entity. Examples include `customer_name`, `address`, and `DOB` (date of birth).
3. **Derived Attributes:** These are attributes that can be computed or inferred from other attributes. For instance, `age` can be calculated based on the `DOB`.

- **Entity: CUSTOMER**

- Attributes: `Customer_ID`, `customer_name`, `gender`, `DOB`, `age`, `address`, `email_ID`, `SSN`

- **Entity: BRANCH**

- Attributes: `branch_id`, `sort_code`, `branch_address`, `phoneNo`, `region`, `zip_code`

- **Entity: DEPARTMENTS**

- Attributes: dept_id, dept_name
- **Entity: ATM**
 - Attributes: atm_ID, atm_address, sort_code
- **Entity: LOCKERS**
 - Attributes: locker_ID, customer_id, sort_code
- **Entity: ACCOUNTS**
 - Attributes: customer_ID, acc_no, acc_type, open_date, current_balance, sort_code
- **Entity: EMPLOYEE**
 - Attributes: employee_ID, sort_code, dept_ID, join_date, employee_position
- **Entity: CARD.INFO**
 - Attributes: customer_ID, card_number, cardtype, acc_no, sort_code, issue_date
- **Entity: LOAN**
 - Attributes: customer_ID, loan_id, sort_code, date_initiated, loan_duration, interest, acc_no
- **Entity: TRANSACTIONS**
 - Attributes: customer_ID, acc_no, transaction_ID, amount, transaction_date, transaction_type
- **Entity: AUDIT**
 - Attributes: Audit_ID, Entity_Type, Entity_ID, Changed_By, Change_Date, Change_Type, Change_Details
- **Entity: ROLE**
 - Attributes: Role_ID, Role_Name
- **Entity: PERMISSION**
 - Attributes: Permission_ID, Permission_Name
- **Entity: ROLE.PERMISSION**
 - Attributes: Role_ID, Permission_ID
- **Entity: USER**
 - Attributes: User_ID, Username, Password.Hash, Employee_ID, Role_ID
- **Entity: SERVICE**
 - Attributes: Service_ID, Service_Name, Description, Fee_Amount
- **Entity: SERVICE_TRANSACTION**
 - Attributes: Transaction_ID, Service_ID
- **Entity: OFFER**
 - Attributes: Offer_ID, Description, Start_Date, End_Date, Eligible_Customers
- **Entity: CUSTOMER.OFFER**
 - Attributes: Customer_ID, Offer_ID, Status

3.4. Simple Relationships

- [CUSTOMER] 1 <owns> M [ACCOUNTS]
- [BRANCH] 1 <has> M [EMPLOYEE]
- [DEPARTMENTS] 1 <manages> M [EMPLOYEE]
- [CUSTOMER] 1 <has> M [LOCKERS]
- [CUSTOMER] 1 <has> M [CARD_INFO]
- [CUSTOMER] 1 <takes> M [LOAN]
- [EMPLOYEE] M <reports> 1 [DEPARTMENTS]
- [TRANSACTIONS] M <related to> 1 [ACCOUNTS]
- [SERVICE_TRANSACTION] M <performed for> M [SERVICE]
- [CUSTOMER_OFFER] M <applies to> M [OFFER]
- [USER] <assigned to> M [ROLE]
- [ROLE_PERMISSION] M <grants> M [PERMISSION]
- [CUSTOMER] 1 <receives> M [OFFER]

3.5. Connectivity's, Cardinalities and Participation

- **USER**
 - A USER is assigned a minimum of 1 ROLE.
 - A USER is assigned a maximum of 1 ROLE.
 - Reverse:
 - * A ROLE is assigned to a minimum of 0 USERS.
 - * A ROLE is assigned to a maximum of M USERS.
 - A USER is linked to a minimum of 1 EMPLOYEE.
 - A USER is linked to a maximum of 1 EMPLOYEE.
 - Reverse:
 - * An EMPLOYEE is linked to a minimum of 0 USERS.
 - * An EMPLOYEE is linked to a maximum of M USERS.
- **ROLE**
 - A ROLE is associated with a minimum of 0 ROLE_PERMISSIONS.
 - A ROLE is associated with a maximum of M ROLE_PERMISSIONS.
 - Reverse:
 - * A ROLE_PERMISSION is linked to a minimum of 1 ROLE.
 - * A ROLE_PERMISSION is linked to a maximum of 1 ROLE.
- **PERMISSION**
 - A PERMISSION is assigned to a minimum of 0 ROLE_PERMISSIONS.
 - A PERMISSION is assigned to a maximum of M ROLE_PERMISSIONS.
 - Reverse:
 - * A ROLE_PERMISSION is linked to a minimum of 1 PERMISSION.
 - * A ROLE_PERMISSION is linked to a maximum of 1 PERMISSION.

- **EMPLOYEE**

- An EMPLOYEE works in a minimum of 1 DEPARTMENT.
- An EMPLOYEE works in a maximum of 1 DEPARTMENT.
- Reverse:
 - * A DEPARTMENT has a minimum of 0 EMPLOYEES.
 - * A DEPARTMENT has a maximum of M EMPLOYEES.
- An EMPLOYEE is assigned to a minimum of 1 BRANCH.
- An EMPLOYEE is assigned to a maximum of 1 BRANCH.
- Reverse:
 - * A BRANCH has a minimum of 0 EMPLOYEES.
 - * A BRANCH has a maximum of M EMPLOYEES.

- **DEPARTMENT**

- A DEPARTMENT is associated with a minimum of 0 TRANSACTIONS.
- A DEPARTMENT is associated with a maximum of M TRANSACTIONS.
- Reverse:
 - * A TRANSACTION is linked to a minimum of 1 DEPARTMENT.
 - * A TRANSACTION is linked to a maximum of 1 DEPARTMENT.

- **SERVICE**

- A SERVICE is linked to a minimum of 0 SERVICE_TRANSACTIONS.
- A SERVICE is linked to a maximum of M SERVICE_TRANSACTIONS.
- Reverse:
 - * A SERVICE_TRANSACTION is linked to a minimum of 1 SERVICE.
 - * A SERVICE_TRANSACTION is linked to a maximum of 1 SERVICE.

- **TRANSACTION**

- A TRANSACTION is linked to a minimum of 1 CUSTOMER.
- A TRANSACTION is linked to a maximum of 1 CUSTOMER.
- Reverse:
 - * A CUSTOMER has a minimum of 0 TRANSACTIONS.
 - * A CUSTOMER has a maximum of M TRANSACTIONS.

- **ACCOUNT**

- An ACCOUNT is associated with a minimum of 1 CUSTOMER.
- An ACCOUNT is associated with a maximum of 1 CUSTOMER.
- Reverse:
 - * A CUSTOMER has a minimum of 0 ACCOUNTS.
 - * A CUSTOMER has a maximum of M ACCOUNTS.

- **LOAN**

- A LOAN is linked to a minimum of 1 CUSTOMER.
- A LOAN is linked to a maximum of 1 CUSTOMER.
- Reverse:
 - * A CUSTOMER has a minimum of 0 LOANS.
 - * A CUSTOMER has a maximum of M LOANS.

- **CARD_INFO**

- A CARD_INFO entry is associated with a minimum of 1 CUSTOMER.

- A CARD_INFO entry is associated with a maximum of 1 CUSTOMER.
- Reverse:
 - * A CUSTOMER has a minimum of 0 CARD_INFO entries.
 - * A CUSTOMER has a maximum of M CARD_INFO entries.

• **CUSTOMER_OFFER**

- A CUSTOMER_OFFER is linked to a minimum of 1 CUSTOMER.
- A CUSTOMER_OFFER is linked to a maximum of 1 CUSTOMER.
- Reverse:
 - * A CUSTOMER has a minimum of 0 CUSTOMER_OFFERS.
 - * A CUSTOMER has a maximum of M CUSTOMER_OFFERS.

• **LOCKERS**

- A LOCKER is associated with a minimum of 1 CUSTOMER.
- A LOCKER is associated with a maximum of 1 CUSTOMER.
- Reverse:
 - * A CUSTOMER has a minimum of 0 LOCKERS.
 - * A CUSTOMER has a maximum of M LOCKERS.

• **BRANCH**

- A BRANCH is associated with a minimum of 0 ATMs.
- A BRANCH is associated with a maximum of M ATMs.
- Reverse:
 - * An ATM is assigned to a minimum of 1 BRANCH.
 - * An ATM is assigned to a maximum of 1 BRANCH.
- A BRANCH has a minimum of 0 EMPLOYEES.
- A BRANCH has a maximum of M EMPLOYEES.
- Reverse:
 - * An EMPLOYEE is linked to a minimum of 1 BRANCH.
 - * An EMPLOYEE is linked to a maximum of 1 BRANCH.

• **AUDIT**

- An AUDIT record is linked to a minimum of 1 ENTITY.
- An AUDIT record is linked to a maximum of 1 ENTITY.
- Reverse:
 - * An ENTITY is linked to a minimum of 0 AUDIT records.
 - * An ENTITY is linked to a maximum of M AUDIT records.

• **ATM**

- An ATM is associated with a minimum of 1 BRANCH.
- An ATM is associated with a maximum of 1 BRANCH.
- Reverse:
 - * A BRANCH has a minimum of 0 ATMs.
 - * A BRANCH has a maximum of M ATMs.

3.6. ERD Mapping

3.6.1. ONE-TO-ONE RELATIONSHIPS

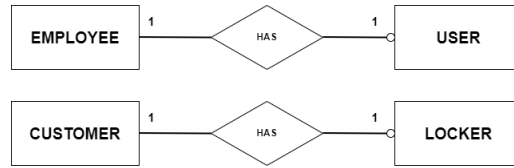


Figure 1. One-to-One Relationships

3.6.2. ONE-TO-MANY RELATIONSHIPS

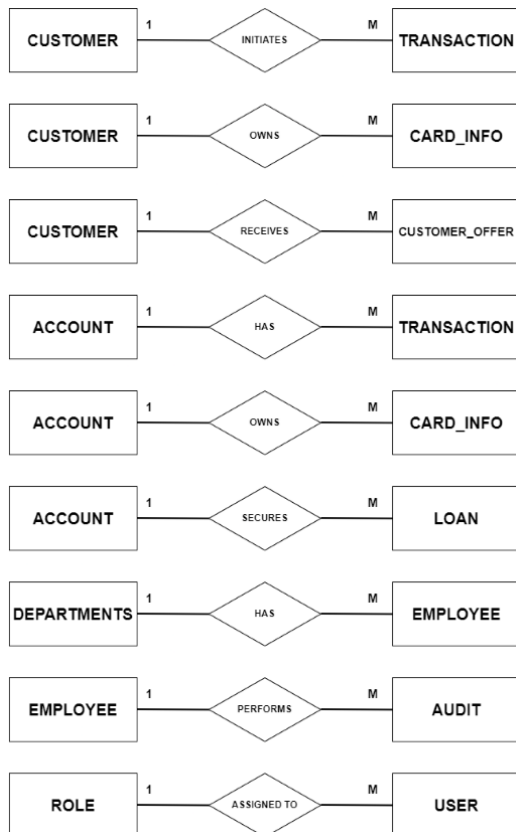


Figure 2.

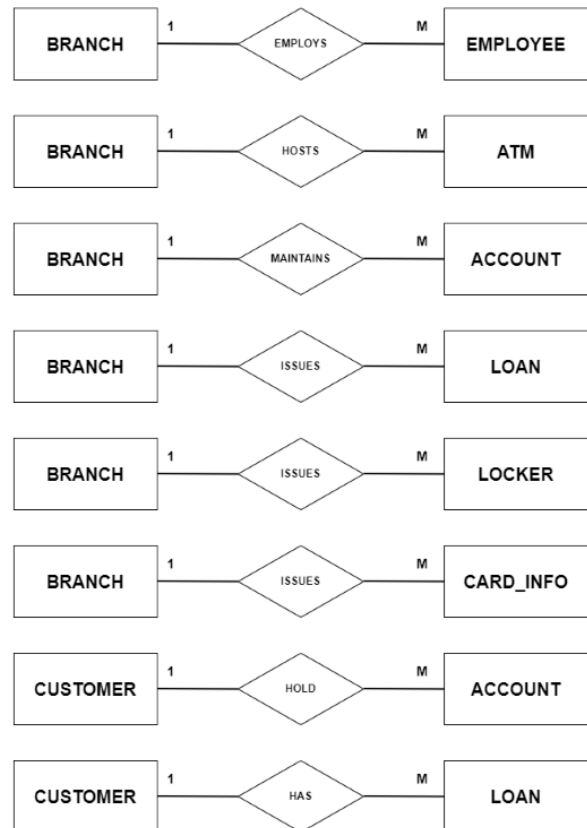


Figure 3.

3.6.3. MANY-TO-MANY RELATIONSHIPS

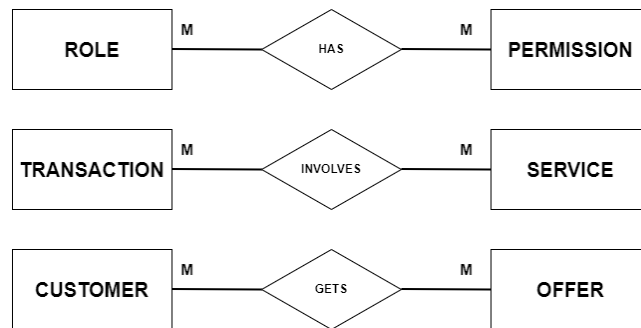


Figure 4. Many-to-Many Relationships

4. Database Normalization

The normalization in this database is up to 3NF. Each table follows 1NF since every column is atomic, with no repeating groups. Moving on to 2NF, there are no partial dependencies; therefore, all the non-key attributes depend fully on the primary key. Similarly, the database also follows 3NF, since for any three attributes 'A', 'B', and 'C', if $A \rightarrow B$ then B does not imply C. Put differently, there are no transitive dependencies; the non-key attribute depends only on the primary key. In tables like 'Customer', 'Branch', 'Accounts', and many more, it is good with clean primary and foreign key relationships, database design is optimized and normalized (Beer et al., 1989).

5. Database Implementation

5.1. DDL Commands

1. Customer Table

```
CREATE TABLE customer (  
    Customer_ID VARCHAR(8) NOT NULL PRIMARY KEY,  
    customer_name VARCHAR(80) NOT NULL,  
    gender VARCHAR(10) NOT NULL,  
    DOB DATE NOT NULL,  
    age INT NOT NULL CHECK (age >= 18),  
    address VARCHAR(100) NOT NULL,  
    email_ID VARCHAR(80) NOT NULL,  
    SSN VARCHAR(30) UNIQUE NOT NULL  
);
```

2. Branch Table

```
CREATE TABLE branch (  
    sort_code VARCHAR(8) NOT NULL PRIMARY KEY,  
    branch_address VARCHAR(100) NOT NULL,  
    phone_No VARCHAR(15) NOT NULL,  
    region VARCHAR(50) NOT NULL,  
    zip_code VARCHAR(10) NOT NULL  
);
```

3. Departments Table

```
CREATE TABLE departments (  
    dept_id VARCHAR(20) NOT NULL PRIMARY KEY,  
    dept_name VARCHAR(80) NOT NULL  
);
```

4. ATM Table

```
CREATE TABLE atm (  
    atm_ID VARCHAR(20) NOT NULL PRIMARY KEY,  
    atm_address VARCHAR(100) NOT NULL,  
    sort_code VARCHAR(8) NOT NULL,  
    FOREIGN KEY (sort_code) REFERENCES branch(sort_code) ON DELETE CASCADE  
);
```

5. Lockers Table

```
CREATE TABLE lockers (  
    locker_ID INT NOT NULL PRIMARY KEY,  
    customer_id VARCHAR(8) NOT NULL,  
    sort_code VARCHAR(8) NOT NULL,  
    FOREIGN KEY (sort_code) REFERENCES branch(sort_code) ON DELETE CASCADE,  
    FOREIGN KEY (customer_ID) REFERENCES customer(customer_ID) ON DELETE CASCADE  
);
```

6. Accounts Table

```
CREATE TABLE accounts (  
    customer_ID VARCHAR(8) NOT NULL,  
    acc_no INT NOT NULL PRIMARY KEY,  
    acc_type VARCHAR(15) NOT NULL,  
    open_date DATE NOT NULL,  
    current_balance FLOAT NOT NULL,  
    sort_code VARCHAR(8) NOT NULL,  
    FOREIGN KEY (sort_code) REFERENCES branch(sort_code) ON DELETE CASCADE,  
    FOREIGN KEY (customer_ID) REFERENCES customer(customer_ID) ON DELETE CASCADE  
);
```

7. Employee Table

```
CREATE TABLE employee (  
    employee_ID VARCHAR(6) NOT NULL PRIMARY KEY,  
    sort_code VARCHAR(8) NOT NULL,  
    dept_ID VARCHAR(20) NOT NULL,  
    join_date DATE NOT NULL,  
    employee_position VARCHAR(50) NOT NULL,  
    FOREIGN KEY (sort_code) REFERENCES branch(sort_code) ON DELETE CASCADE,  
    FOREIGN KEY (dept_ID) REFERENCES departments(dept_id) ON DELETE CASCADE  
);
```

8. Card.Info Table

```
CREATE TABLE card_info (  
    customer_ID VARCHAR(8) NOT NULL,  
    card_number BIGINT NOT NULL PRIMARY KEY,  
    card_type VARCHAR(20) NOT NULL,  
    acc_no INT NOT NULL,  
    sort_code VARCHAR(8) NOT NULL,  
    issue_date DATE NOT NULL,  
    FOREIGN KEY (sort_code) REFERENCES branch(sort_code) ON DELETE NO ACTION,  
    FOREIGN KEY (customer_ID) REFERENCES customer(customer_ID) ON DELETE NO ACTION,  
    FOREIGN KEY (acc_no) REFERENCES accounts(acc_no) ON DELETE NO ACTION  
);
```

9. Loan Table

```
CREATE TABLE loan (
    customer_ID VARCHAR(8) NOT NULL,
    loan_id VARCHAR(10) NOT NULL PRIMARY KEY,
    sort_code VARCHAR(8) NOT NULL,
    date_initiated DATE NOT NULL,
    loan_duration INT NOT NULL,
    interest FLOAT NOT NULL,
    acc_no INT NOT NULL,
    FOREIGN KEY (sort_code) REFERENCES branch(sort_code) ON DELETE NO ACTION,
    FOREIGN KEY (customer_ID) REFERENCES customer(customer_ID) ON DELETE NO ACTION,
    FOREIGN KEY (acc_no) REFERENCES accounts(acc_no) ON DELETE NO ACTION
);
```

10. Transactions Table

```
CREATE TABLE transactions (
    customer_ID VARCHAR(8) NOT NULL,
    acc_no INT NOT NULL,
    transaction_ID VARCHAR(8) NOT NULL PRIMARY KEY,
    amount FLOAT NOT NULL,
    transaction_date DATE NOT NULL,
    transaction_type VARCHAR(15) NOT NULL,
    FOREIGN KEY (customer_ID) REFERENCES customer(customer_ID),
    FOREIGN KEY (acc_no) REFERENCES accounts(acc_no)
);
```

11. Audit Table

```
CREATE TABLE Audit (
    Audit_ID INT IDENTITY(1,1) PRIMARY KEY,
    Entity_Type NVARCHAR(50),
    Entity_ID INT,
    Changed_By VARCHAR(6),
    Change_Date DATETIME DEFAULT GETDATE(),
    Change_Type NVARCHAR(50),
    Change_Details NVARCHAR(MAX),
    FOREIGN KEY (Changed_By) REFERENCES employee(employee_ID)
);
```

12. Role Table

```
CREATE TABLE Role (
    Role_ID INT IDENTITY(1,1) PRIMARY KEY,
    Role_Name NVARCHAR(50) NOT NULL UNIQUE
);
```

13. Permission Table

```
CREATE TABLE Permission (
    Permission_ID INT IDENTITY(1,1) PRIMARY KEY,
```

```
        Permission_Name NVARCHAR(50) NOT NULL UNIQUE
    );
```

14. Role_Permission Table

```
CREATE TABLE Role_Permission (
    Role_ID INT NOT NULL,
    Permission_ID INT NOT NULL,
    PRIMARY KEY (Role_ID, Permission_ID),
    FOREIGN KEY (Role_ID) REFERENCES Role(Role_ID),
    FOREIGN KEY (Permission_ID) REFERENCES Permission(Permission_ID)
);
```

15. User Table

```
CREATE TABLE [User] (
    User_ID INT IDENTITY(1,1) PRIMARY KEY,
    Username NVARCHAR(50) NOT NULL UNIQUE,
    Password_Hash NVARCHAR(255) NOT NULL,
    Employee_ID VARCHAR(6),
    Role_ID INT,
    FOREIGN KEY (Employee_ID) REFERENCES employee(employee_ID),
    FOREIGN KEY (Role_ID) REFERENCES Role(Role_ID)
);
```

16. Service Table

```
CREATE TABLE Service (
    Service_ID INT IDENTITY(1,1) PRIMARY KEY,
    Service_Name NVARCHAR(100) NOT NULL,
    Description NVARCHAR(MAX),
    Fee_Amount DECIMAL(18, 2)
);
```

17. Service_Transaction Table

```
CREATE TABLE Service_Transaction (
    Transaction_ID VARCHAR(8) NOT NULL,
    Service_ID INT NOT NULL,
    PRIMARY KEY (Transaction_ID, Service_ID),
    FOREIGN KEY (Transaction_ID) REFERENCES transactions(transaction_ID),
    FOREIGN KEY (Service_ID) REFERENCES Service(Service_ID)
);
```

18. Offer Table

```
CREATE TABLE Offer (
    Offer_ID INT IDENTITY(1,1) PRIMARY KEY,
    Description NVARCHAR(MAX),
```

```
        Start_Date DATE,  
        End_Date DATE,  
        Eligible_Customers NVARCHAR(MAX)  
    );
```

19. Customer_Offer Table

```
CREATE TABLE Customer_Offer (  
    Customer_ID VARCHAR(8) NOT NULL,  
    Offer_ID INT NOT NULL,  
    Status NVARCHAR(50),  
    PRIMARY KEY (Customer_ID, Offer_ID),  
    FOREIGN KEY (Customer_ID) REFERENCES customer(Customer_ID),  
    FOREIGN KEY (Offer_ID) REFERENCES Offer(Offer_ID)  
);
```

5.2. Additional Commands for constraints

1. Alter Table to Add Gender Check Constraint

```
ALTER TABLE customer  
ADD CONSTRAINT CHK_Gender CHECK (gender IN ('Male', 'Female', 'Other'));
```

2. Create Index on Transaction Date

```
CREATE INDEX IDX_TransactionDate ON transactions(transaction_date);
```

3. Alter Table to Add Unique Constraint on Email

```
ALTER TABLE customer  
ADD CONSTRAINT UQ_Email UNIQUE (email_ID);
```

4. Alter Table to Add Balance Check Constraint

```
ALTER TABLE accounts  
ADD CONSTRAINT CHK_Balance CHECK (current_balance >= 0);
```

5. Alter Table to Add Default Status Constraint

```
ALTER TABLE Customer_Offer  
ADD CONSTRAINT DF_Status DEFAULT 'Pending' FOR Status;
```

6. Alter Table to Add Unique Constraint on Locker and Sort Code

```
ALTER TABLE lockers  
ADD CONSTRAINT UQ_Locker_SortCode UNIQUE (locker_ID, sort_code);
```

7. Alter Table to Add Default Join Date Constraint

```
ALTER TABLE employee  
ADD CONSTRAINT DF_JoinDate DEFAULT GETDATE() FOR join_date;
```

5.3. Insertion Commands

To maintain clarity and conciseness in the presentation of the data, only the first five INSERT commands from each table have been displayed. This approach is adopted due to the length and repetition of the INSERT statements, ensuring that the essential structure and format of the data are clear. Insert Commands for 19 Tables are as follows:

5.3.1. CUSTOMER

```
INSERT INTO customer (Customer_ID, customer_name, gender, DOB, age,
address, email_ID, SSN)
VALUES
('C000001', 'James Smith', 'Male', '1980-05-21', 44, '123 Baker Street,
London', 'james.smith@email.com', 'SSN12345601'),
('C000002', 'Emma Johnson', 'Female', '1992-08-15', 32, '45 Oxford Road,
Manchester', 'emma.johnson@email.com', 'SSN12345602'),
('C000003', 'Oliver Williams', 'Male', '1975-11-11', 49, '22 Queen Square,
Bristol', 'oliver.williams@email.com', 'SSN12345603'),
('C000004', 'Isabella Brown', 'Female', '1985-04-28', 39, '88 King Street,
Edinburgh', 'isabella.brown@email.com', 'SSN12345604'),
('C000005', 'George Jones', 'Male', '1960-09-30', 63, '16 Elm Street,
Cardiff', 'george.jones@email.com', 'SSN12345605');
```

| | Customer_ID | customer_name | gender | DOB | age | address | email_ID | SSN |
|----|-------------|------------------|--------|------------|-----|----------------------------|----------------------------|-------------|
| 1 | C000001 | James Smith | Male | 1980-05-21 | 44 | 123 Baker Street, London | james.smith@email.com | SSN12345601 |
| 2 | C000002 | Emma Johnson | Female | 1992-08-15 | 32 | 45 Oxford Road, Manchester | emma.johnson@email.com | SSN12345602 |
| 3 | C000003 | Oliver Williams | Male | 1975-11-11 | 49 | 22 Queen Square, Bristol | oliver.williams@email.com | SSN12345603 |
| 4 | C000004 | Isabella Brown | Female | 1985-04-28 | 39 | 88 King Street, Edinburgh | isabella.brown@email.com | SSN12345604 |
| 5 | C000005 | George Jones | Male | 1960-09-30 | 63 | 16 Elm Street, Cardiff | george.jones@email.com | SSN12345605 |
| 6 | C000006 | Mia Davis | Female | 1995-12-03 | 28 | 71 High Street, Belfast | mia.davis@email.com | SSN12345606 |
| 7 | C000007 | Lucas Wilson | Male | 1988-07-17 | 36 | 34 Park Avenue, Glasgow | lucas.wilson@email.com | SSN12345607 |
| 8 | C000008 | Charlotte Miller | Female | 1970-03-22 | 54 | 19 Broad Street, Newcastle | charlotte.miller@email.com | SSN12345608 |
| 9 | C000009 | Liam Taylor | Male | 2000-01-10 | 24 | 7 Victoria Road, Liverpool | liam.taylor@email.com | SSN12345609 |
| 10 | C000010 | Sophia Anderson | Female | 1967-02-14 | 57 | 10 Hill Street, Birmingham | sophia.anderson@email.com | SSN12345610 |

Figure 5. Customer Table

5.3.2. BRANCH

```
INSERT INTO branch (branch_id, sort_code, branch_address, phone_No, region,
zip_code)
VALUES
(1, 'SC001234', '123 Bank St, London', '02079460001', 'London', 'WC1A
1AA'),
(2, 'SC001234', '456 Finance St, Birmingham', '01216060002', 'West
Midlands', 'B1 1AA'),
(3, 'SC001234', '789 Money Ave, Manchester', '01618360003', 'Greater
Manchester', 'M1 1AA'),
(4, 'SC001234', '101 Savings Rd, Leeds', '01133600004', 'West Yorkshire',
'LS1 1AA'),
(5, 'SC001234', '202 Investment Blvd, Liverpool', '01512360005',
'Merseyside', 'L1 1AA');
```

Bank Database Management System

| | branch_id | sort_code | branch_address | phone_No | region | zip_code |
|----|-----------|-----------|--------------------------------|-------------|--------------------|----------|
| 1 | 1 | SC001234 | 123 Bank St, London | 02079460001 | London | WC1A 1AA |
| 2 | 2 | SC001234 | 456 Finance St, Birmingham | 01216060002 | West Midlands | B1 1AA |
| 3 | 3 | SC001234 | 789 Money Ave, Manchester | 01618360003 | Greater Manchester | M1 1AA |
| 4 | 4 | SC001234 | 101 Savings Rd, Leeds | 01133600004 | West Yorkshire | LS1 1AA |
| 5 | 5 | SC001234 | 202 Investment Blvd, Liverpool | 01512360005 | Merseyside | L1 1AA |
| 6 | 6 | SC001234 | 303 Wealth St, Newcastle | 01912600006 | Tyne and Wear | NE1 1AA |
| 7 | 7 | SC001234 | 404 Capital Ln, Bristol | 01173000007 | South West | BS1 1AA |
| 8 | 8 | SC001234 | 505 Prosperity Sq, Edinburgh | 01312000008 | Scotland | EH1 1AA |
| 9 | 9 | SC001234 | 606 Fortune St, Cardiff | 02920600009 | Wales | CF1 1AA |
| 10 | 10 | SC001234 | 707 Equity Rd, Glasgow | 01414000010 | Scotland | G1 1AA |

Figure 6. Branch Table

5.3.3. DEPARTMENTS

```
INSERT INTO departments (dept_id, dept_name)
VALUES
('D001', 'Finance'),
('D002', 'Customer Service'),
('D003', 'IT'),
('D004', 'Marketing'),
('D005', 'Operations');
```

| | dept_id | dept_name |
|----|---------|--------------------|
| 1 | D001 | Finance |
| 2 | D002 | Customer Service |
| 3 | D003 | IT |
| 4 | D004 | Marketing |
| 5 | D005 | Operations |
| 6 | D006 | HR |
| 7 | D007 | Legal |
| 8 | D008 | Compliance |
| 9 | D009 | Risk Management |
| 10 | D010 | Corporate Strategy |

Figure 7. Departments Table

5.3.4. ATM

```
INSERT INTO atm (atm_ID, atm_address, sort_code)
VALUES
('ATM001', '10 Downing St, London', 'SC001234'),
('ATM002', '20 Westminster St, Birmingham', 'SC001234'),
('ATM003', '30 Piccadilly, Manchester', 'SC001234'),
('ATM004', '40 Hyde Park, Leeds', 'SC001234'),
('ATM005', '50 Liverpool St, Liverpool', 'SC001234');
```


| | atm_ID | atm_address | sort_code |
|----|--------|---------------------------------|-----------|
| 1 | ATM001 | 10 Downing St, London | SC001234 |
| 2 | ATM002 | 20 Westminster St, Birmingham | SC001234 |
| 3 | ATM003 | 30 Piccadilly, Manchester | SC001234 |
| 4 | ATM004 | 40 Hyde Park, Leeds | SC001234 |
| 5 | ATM005 | 50 Liverpool St, Liverpool | SC001234 |
| 6 | ATM006 | 60 Northumberland St, Newcastle | SC001234 |
| 7 | ATM007 | 70 Cabot Circus, Bristol | SC001234 |
| 8 | ATM008 | 80 Princes St, Edinburgh | SC001234 |
| 9 | ATM009 | 90 Queen St, Cardiff | SC001234 |
| 10 | ATM010 | 100 Buchanan St, Glasgow | SC001234 |
| 11 | ATM011 | 110 Whitehall, London | SC001234 |
| 12 | ATM012 | 120 Broad St, Birmingham | SC001234 |
| 13 | ATM013 | 130 Deansgate, Manchester | SC001234 |
| 14 | ATM014 | 140 Briggate, Leeds | SC001234 |

Figure 8. ATM Table

5.3.5. LOCKERS

```
INSERT INTO lockers (locker_ID, customer_id, sort_code)
VALUES
(1, 'C000001', 'SC001234'),
(2, 'C000003', 'SC001234'),
(3, 'C000005', 'SC001234'),
(4, 'C000007', 'SC001234'),
(5, 'C000009', 'SC001234');
```

| | locker_ID | customer_id | sort_code |
|----|-----------|-------------|-----------|
| 1 | 1 | C000001 | SC001234 |
| 2 | 2 | C000003 | SC001234 |
| 3 | 3 | C000005 | SC001234 |
| 4 | 4 | C000007 | SC001234 |
| 5 | 5 | C000009 | SC001234 |
| 6 | 6 | C000002 | SC001234 |
| 7 | 7 | C000004 | SC001234 |
| 8 | 8 | C000006 | SC001234 |
| 9 | 9 | C000008 | SC001234 |
| 10 | 10 | C000010 | SC001234 |
| 11 | 11 | C000011 | SC001234 |
| 12 | 12 | C000013 | SC001234 |
| 13 | 13 | C000015 | SC001234 |

Figure 9. Lockers Table

5.3.6. ACCOUNTS

```
INSERT INTO accounts (customer_ID, acc.no, acc.type, open.date,
current_balance, sort_code)
VALUES
('C000001', 10000001, 'Savings', '2020-01-01', 2500.00, 'SC001234'),
('C000002', 10000002, 'Current', '2020-01-02', 3500.00, 'SC001234'),
('C000003', 10000003, 'Savings', '2020-01-03', 1500.00, 'SC001234'),
('C000004', 10000004, 'Current', '2020-01-04', 4500.00, 'SC001234'),
('C000005', 10000005, 'Savings', '2020-01-05', 5500.00, 'SC001234');
```

Bank Database Management System

| | customer_ID | acc_no | acc_type | open_date | current_balance | sort_code |
|----|-------------|----------|----------|------------|-----------------|-----------|
| 1 | C000001 | 10000001 | Savings | 2020-01-01 | 2500 | SC001234 |
| 2 | C000002 | 10000002 | Current | 2020-01-02 | 3500 | SC001234 |
| 3 | C000003 | 10000003 | Savings | 2020-01-03 | 1500 | SC001234 |
| 4 | C000004 | 10000004 | Current | 2020-01-04 | 4500 | SC001234 |
| 5 | C000005 | 10000005 | Savings | 2020-01-05 | 5500 | SC001234 |
| 6 | C000006 | 10000006 | Current | 2020-01-06 | 6500 | SC001234 |
| 7 | C000007 | 10000007 | Savings | 2020-01-07 | 7500 | SC001234 |
| 8 | C000008 | 10000008 | Current | 2020-01-08 | 8500 | SC001234 |
| 9 | C000009 | 10000009 | Savings | 2020-01-09 | 9500 | SC001234 |
| 10 | C000010 | 10000010 | Current | 2020-01-10 | 10500 | SC001234 |
| 11 | C000011 | 10000011 | Savings | 2020-01-11 | 11500 | SC001234 |
| 12 | C000012 | 10000012 | Current | 2020-01-12 | 12500 | SC001234 |
| 13 | C000013 | 10000013 | Savings | 2020-01-13 | 13500 | SC001234 |
| 14 | C000014 | 10000014 | Current | 2020-01-14 | 14500 | SC001234 |

Figure 10. Accounts Table

5.3.7. EMPLOYEE

```
INSERT INTO employee (employee_ID, sort_code, dept_ID, join_date,
employee_position)
VALUES
('E00001', 'SC001234', 'D001', '2015-05-10', 'Manager'),
('E00002', 'SC001234', 'D002', '2016-07-20', 'Customer Service
Representative'),
('E00003', 'SC001234', 'D003', '2017-09-15', 'IT Specialist'),
('E00004', 'SC001234', 'D004', '2018-11-05', 'Compliance Officer'),
('E00005', 'SC001234', 'D005', '2019-02-25', 'Operations Manager');
```

| | employee_ID | sort_code | dept_ID | join_date | employee_position |
|----|-------------|-----------|---------|------------|---------------------------------|
| 1 | E00001 | SC001234 | D001 | 2015-05-10 | Manager |
| 2 | E00002 | SC001234 | D002 | 2016-07-20 | Customer Service Representative |
| 3 | E00003 | SC001234 | D003 | 2017-09-15 | IT Specialist |
| 4 | E00004 | SC001234 | D004 | 2018-11-05 | Compliance Officer |
| 5 | E00005 | SC001234 | D005 | 2019-02-25 | Operations Manager |
| 6 | E00006 | SC001234 | D006 | 2020-04-18 | Legal Advisor |
| 7 | E00007 | SC001234 | D007 | 2021-06-12 | Risk Analyst |
| 8 | E00008 | SC001234 | D008 | 2022-08-30 | HR Manager |
| 9 | E00009 | SC001234 | D009 | 2023-10-01 | Marketing Analyst |
| 10 | E00010 | SC001234 | D010 | 2023-12-12 | Corporate Strategist |
| 11 | E00011 | SC001234 | D001 | 2015-06-15 | Operations Manager |
| 12 | E00012 | SC001234 | D002 | 2016-08-20 | Customer Service Representative |
| 13 | E00013 | SC001234 | D003 | 2017-10-18 | Legal Advisor |
| 14 | E00014 | SC001234 | D004 | 2018-12-01 | IT Specialist |

Figure 11. Employee Table

5.3.8. CARD INFO

```
INSERT INTO cardinfo (customer_ID, card_number, card_type, acc.no,
sort_code, issue_date)
VALUES
('C000001', 234567890123, 'Credit', 10000001, 'SC001234', '2021-02-01'),
```

```
('C000001', 234567890124, 'Debit', 10000001, 'SC001234', '2021-02-01'),
('C000005', 345678901234, 'Credit', 10000005, 'SC001234', '2021-03-01'),
('C000005', 345678901235, 'Debit', 10000005, 'SC001234', '2021-03-01'),
('C000010', 456789012345, 'Credit', 10000010, 'SC001234', '2021-04-01');
```

| | customer_ID | card_number | card_type | acc_no | sort_code | issue_date |
|----|-------------|--------------|-----------|----------|-----------|------------|
| 1 | C000040 | 123456789001 | Credit | 10000040 | SC001234 | 2021-10-01 |
| 2 | C000040 | 123456789002 | Debit | 10000040 | SC001234 | 2021-10-01 |
| 3 | C000009 | 123456789003 | Debit | 10000009 | SC001234 | 2021-10-15 |
| 4 | C000021 | 123456789004 | Credit | 10000021 | SC001234 | 2022-07-01 |
| 5 | C000032 | 123456789005 | Debit | 10000032 | SC001234 | 2023-04-01 |
| 6 | C000001 | 234567890123 | Credit | 10000001 | SC001234 | 2021-02-01 |
| 7 | C000001 | 234567890124 | Debit | 10000001 | SC001234 | 2021-02-01 |
| 8 | C000045 | 234567890125 | Credit | 10000045 | SC001234 | 2021-11-01 |
| 9 | C000045 | 234567890126 | Debit | 10000045 | SC001234 | 2021-11-01 |
| 10 | C000011 | 234567890127 | Credit | 10000011 | SC001234 | 2021-11-20 |
| 11 | C000022 | 234567890128 | Debit | 10000022 | SC001234 | 2022-08-01 |
| 12 | C000033 | 234567890129 | Credit | 10000033 | SC001234 | 2023-05-01 |
| 13 | C000005 | 345678901234 | Credit | 10000005 | SC001234 | 2021-03-01 |
| 14 | C000005 | 345678901235 | Debit | 10000005 | SC001234 | 2021-03-01 |

Figure 12. Card-Information Table

5.3.9. LOAN

```
INSERT INTO loan (customer_ID, loan_id, sort_code, date_initiated,
loan_duration, interest, acc_no)
VALUES
('C000005', 'L000001', 'SC001234', '2020-02-15', 60, 3.50, 10000005),
('C000020', 'L000002', 'SC001234', '2020-03-20', 48, 4.00, 10000020),
('C000030', 'L000003', 'SC001234', '2020-04-25', 36, 4.50, 10000030),
('C000042', 'L000004', 'SC001234', '2020-05-30', 60, 3.75, 10000042),
('C000057', 'L000005', 'SC001234', '2020-06-10', 48, 4.25, 10000057);
```

| | customer_ID | loan_id | sort_code | date_initiated | loan_duration | interest | acc_no |
|----|-------------|---------|-----------|----------------|---------------|----------|----------|
| 1 | C000005 | L000001 | SC001234 | 2020-02-15 | 60 | 3.5 | 10000005 |
| 2 | C000020 | L000002 | SC001234 | 2020-03-20 | 48 | 4 | 10000020 |
| 3 | C000030 | L000003 | SC001234 | 2020-04-25 | 36 | 4.5 | 10000030 |
| 4 | C000042 | L000004 | SC001234 | 2020-05-30 | 60 | 3.75 | 10000042 |
| 5 | C000057 | L000005 | SC001234 | 2020-06-10 | 48 | 4.25 | 10000057 |
| 6 | C000063 | L000006 | SC001234 | 2020-07-15 | 36 | 3.8 | 10000063 |
| 7 | C000076 | L000007 | SC001234 | 2020-08-20 | 60 | 4.1 | 10000076 |
| 8 | C000081 | L000008 | SC001234 | 2020-09-25 | 48 | 3.9 | 10000081 |
| 9 | C000091 | L000009 | SC001234 | 2020-10-30 | 36 | 4.2 | 10000091 |
| 10 | C000100 | L000010 | SC001234 | 2020-11-10 | 60 | 4 | 10000100 |
| 11 | C000013 | L000011 | SC001234 | 2020-12-15 | 60 | 3.6 | 10000013 |
| 12 | C000024 | L000012 | SC001234 | 2021-01-20 | 48 | 3.95 | 10000024 |
| 13 | C000037 | L000013 | SC001234 | 2021-02-25 | 36 | 4.05 | 10000037 |
| 14 | C000045 | L000014 | SC001234 | 2021-03-30 | 60 | 3.7 | 10000045 |

Figure 13. Loan Table

5.3.10. TRANSACTIONS

```

INSERT INTO transactions (customer_ID, acc_no, transaction_ID, amount,
transaction_date, transaction_type)
VALUES
('C000001', 10000001, 'T000001', 250.00, '2023-01-01', 'Deposit'),
('C000001', 10000001, 'T000002', 150.00, '2023-01-02', 'Withdrawal'),
('C000002', 10000002, 'T000003', 500.00, '2023-01-03', 'Deposit'),
('C000002', 10000002, 'T000004', 300.00, '2023-01-04', 'Withdrawal'),
('C000003', 10000003, 'T000005', 400.00, '2023-01-05', 'Deposit');

```

| | customer_ID | acc_no | transaction_ID | amount | transaction_date | transaction_type |
|----|-------------|----------|----------------|--------|------------------|------------------|
| 1 | C000001 | 10000001 | T000001 | 61658 | 2023-01-02 | Deposit |
| 2 | C000001 | 10000001 | T000002 | 63131 | 2023-01-03 | Withdrawal |
| 3 | C000001 | 10000001 | T000003 | 214900 | 2023-01-04 | Deposit |
| 4 | C000001 | 10000001 | T000004 | 183647 | 2023-01-05 | Deposit |
| 5 | C000002 | 10000002 | T000005 | 187003 | 2023-01-06 | Withdrawal |
| 6 | C000002 | 10000002 | T000006 | 293813 | 2023-01-07 | Withdrawal |
| 7 | C000002 | 10000002 | T000007 | 171257 | 2023-01-08 | Deposit |
| 8 | C000002 | 10000002 | T000008 | 178000 | 2023-01-09 | Deposit |
| 9 | C000002 | 10000002 | T000009 | 25317 | 2023-01-10 | Withdrawal |
| 10 | C000003 | 10000003 | T000010 | 60254 | 2023-01-11 | Deposit |
| 11 | C000003 | 10000003 | T000011 | 230679 | 2023-01-12 | Deposit |
| 12 | C000003 | 10000003 | T000012 | 30457 | 2023-01-13 | Withdrawal |
| 13 | C000003 | 10000003 | T000013 | 59534 | 2023-01-14 | Withdrawal |
| 14 | C000004 | 10000004 | T000014 | 53615 | 2023-01-15 | Deposit |

Figure 14. Transaction Table

5.3.11. AUDIT

```

INSERT INTO Audit (Entity_Type, Entity_ID, Changed_By, Change_Date,
Change_Type, Change_Details)
VALUES
('Customer', 1, 'E00001', GETDATE(), 'Insert', 'Inserted new customer
record'),
('Customer', 2, 'E00002', GETDATE(), 'Update', 'Updated customer record'),
('Customer', 3, 'E00003', GETDATE(), 'Delete', 'Deleted customer record'),
('Customer', 4, 'E00004', GETDATE(), 'Insert', 'Inserted new customer
record'),
('Customer', 5, 'E00005', GETDATE(), 'Update', 'Updated customer record');

```

Bank Database Management System

| | Audit_ID | Entity_Type | Entity_ID | Changed_By | Change_Date | Change_Type | Change_Details |
|----|----------|-------------|-----------|------------|-------------------------|-------------|------------------------------|
| 1 | 1 | Customer | 1 | E00042 | 2024-08-25 14:24:46.477 | Delete | Deleted customer record |
| 2 | 2 | Customer | 2 | E00064 | 2024-08-25 14:24:46.480 | Update | Updated customer record |
| 3 | 3 | Customer | 3 | E00005 | 2024-08-25 14:24:46.480 | Insert | Inserted new customer record |
| 4 | 4 | Customer | 4 | E00018 | 2024-08-25 14:24:46.480 | Delete | Deleted customer record |
| 5 | 5 | Customer | 5 | E00007 | 2024-08-25 14:24:46.480 | Insert | Inserted new customer record |
| 6 | 6 | Customer | 6 | E00005 | 2024-08-25 14:24:46.480 | Delete | Deleted customer record |
| 7 | 7 | Customer | 7 | E00035 | 2024-08-25 14:24:46.480 | Delete | Deleted customer record |
| 8 | 8 | Customer | 8 | E00047 | 2024-08-25 14:24:46.480 | Update | Updated customer record |
| 9 | 9 | Customer | 9 | E00028 | 2024-08-25 14:24:46.480 | Insert | Inserted new customer record |
| 10 | 10 | Customer | 10 | E00046 | 2024-08-25 14:24:46.480 | Update | Updated customer record |
| 11 | 11 | Customer | 11 | E00021 | 2024-08-25 14:24:46.483 | Delete | Deleted customer record |
| 12 | 12 | Customer | 12 | E00086 | 2024-08-25 14:24:46.483 | Insert | Inserted new customer record |
| 13 | 13 | Customer | 13 | E00053 | 2024-08-25 14:24:46.483 | Delete | Deleted customer record |
| 14 | 14 | Customer | 14 | E00100 | 2024-08-25 14:24:46.483 | Update | Updated customer record |

Figure 15. Audit Table

5.3.12. ROLE

```

INSERT INTO Role (Role_Name)
VALUES
('Legal Role'),
('Manager Role'),
('Support Role'),
('Strategy Role'),
('Analyst Role');

```

| | Role_ID | Role_Name |
|---|---------|-----------------|
| 1 | 8 | Admin Role |
| 2 | 5 | Analyst Role |
| 3 | 7 | Compliance Role |
| 4 | 6 | HR Role |
| 5 | 1 | Legal Role |
| 6 | 2 | Manager Role |
| 7 | 4 | Strategy Role |
| 8 | 3 | Support Role |

Figure 16. Role Table

5.3.13. PERMISSION

```

INSERT INTO Permission (Permission_Name)
VALUES
('View Customers'),
('Edit Customers'),
('Delete Customers'),
('View Accounts'),
('Edit Accounts');

```

| | Permission_ID | Permission_Name |
|----|---------------|---------------------|
| 1 | 6 | Delete Accounts |
| 2 | 3 | Delete Customers |
| 3 | 9 | Delete Transactions |
| 4 | 5 | Edit Accounts |
| 5 | 2 | Edit Customers |
| 6 | 8 | Edit Transactions |
| 7 | 4 | View Accounts |
| 8 | 1 | View Customers |
| 9 | 10 | View Reports |
| 10 | 7 | View Transactions |

Figure 17. Permission Table

5.3.14. ROLE_PERMISSION

```
INSERT INTO Role_Permission (Role_ID, Permission_ID)
VALUES
(8, 4),
(8, 1),
(8, 7),
(8, 10),
(8, 5);
```

| | Role_ID | Permission_ID |
|----|---------|---------------|
| 1 | 1 | 1 |
| 2 | 1 | 4 |
| 3 | 1 | 7 |
| 4 | 1 | 10 |
| 5 | 2 | 1 |
| 6 | 2 | 2 |
| 7 | 2 | 4 |
| 8 | 2 | 5 |
| 9 | 2 | 7 |
| 10 | 2 | 8 |
| 11 | 2 | 10 |
| 12 | 3 | 1 |
| 13 | 3 | 2 |
| 14 | 3 | 4 |

Figure 18. Role_Permission Table

5.3.15. USER

```
INSERT INTO [User] (Username, Password_Hash, Employee_ID, Role_ID)
VALUES
('manager_e00001', '5f4dcc3b5aa765d61d8327deb882cf99', 'E00001', 2),
('customer_service_representative_e00002', '6cb75f652a9b52798eb6cf2201057c73', 'E00002', 2),
('it_specialist_e00003', '8d969eef6ecad3c29a3a629280e686cf', 'E00003', 2),
('compliance_officer_e00004', 'e99a18c428cb38d5f260853678922e03', 'E00004', 7),
('operations_manager_e00005', 'f6fdffe48c908deb0f4c3bd36c032e72', 'E00005',
```

2);

| | User_ID | Username | Password_Hash | Employee_ID | Role_ID |
|----|---------|--|----------------------------------|-------------|---------|
| 1 | 1 | manager_e00001 | 5f4dcc3b5aa765d61d8327deb882cf99 | E00001 | 2 |
| 2 | 2 | customer_service_representative_e00002 | 6cb75f652a9b52798eb6cf2201057c73 | E00002 | 2 |
| 3 | 3 | it_specialist_e00003 | 8d969eef6ecad3c29a3a629280e686cf | E00003 | 2 |
| 4 | 4 | compliance_officer_e00004 | e99a18c428cb38d5f260853678922e03 | E00004 | 7 |
| 5 | 5 | operations_manager_e00005 | f6fdffe48c908deb0f4c3bd36c032e72 | E00005 | 2 |
| 6 | 6 | legal_advisor_e00006 | 9a1158154dfa42caddbd0694a4e9bdc8 | E00006 | 1 |
| 7 | 7 | risk_analyst_e00007 | 6c97424dc02e4c125e4f4ea8e0f6599a | E00007 | 5 |
| 8 | 8 | hr_manager_e00008 | 5d41402abc4b2a76b9719d911017c592 | E00008 | 6 |
| 9 | 9 | marketing_analyst_e00009 | 7c6a180b36896a0a8c02787eeafb0e4c | E00009 | 5 |
| 10 | 10 | corporate_strategist_e00010 | 098f6bcd4621d373cade4e832627b4f6 | E00010 | 4 |
| 11 | 11 | operations_manager_e00011 | 5f4dcc3b5aa765d61d8327deb882cf99 | E00011 | 2 |
| 12 | 12 | customer_service_representative_e00012 | 6cb75f652a9b52798eb6cf2201057c73 | E00012 | 2 |
| 13 | 13 | legal_advisor_e00013 | 8d969eef6ecad3c29a3a629280e686cf | E00013 | 1 |
| 14 | 14 | it_specialist_e00014 | e99a18c428cb38d5f260853678922e03 | E00014 | 2 |

Figure 19. User Table

5.3.16. SERVICE

```

INSERT INTO Service (Service_Name, Description, Fee_Amount)
VALUES
('Online Banking', 'Access to online banking services', 5.00),
('Mobile Banking', 'Access to mobile banking services', 3.00),
('ATM Withdrawal', 'Withdrawal at ATMs', 1.50),
('Cheque Book', 'Request for cheque book', 10.00),
('Account Statement', 'Request for account statement', 2.00);

```

| | Service_ID | Service_Name | Description | Fee_Amount |
|----|------------|-------------------|-----------------------------------|------------|
| 1 | 1 | Online Banking | Access to online banking services | 5.00 |
| 2 | 2 | Mobile Banking | Access to mobile banking services | 3.00 |
| 3 | 3 | ATM Withdrawal | Withdrawal at ATMs | 1.50 |
| 4 | 4 | Cheque Book | Request for cheque book | 10.00 |
| 5 | 5 | Account Statement | Request for account statement | 2.00 |
| 6 | 6 | Loan Processing | Processing of loan applications | 50.00 |
| 7 | 7 | Credit Card | Issuance of credit card | 25.00 |
| 8 | 8 | Debit Card | Issuance of debit card | 10.00 |
| 9 | 9 | Overdraft | Overdraft facility | 15.00 |
| 10 | 10 | Account Closure | Closure of bank account | 20.00 |

Figure 20. Service Table

5.3.17. SERVICE_TRANSACTION

```

INSERT INTO Service_Transaction (Transaction_ID, Service_ID)
VALUES
('T000005', 5),

```



```
( 'T000006', 6),
( 'T000007', 7),
( 'T000008', 8),
( 'T000009', 9);
```

| | Transaction_ID | Service_ID |
|---|----------------|------------|
| 1 | T000005 | 5 |
| 2 | T000006 | 6 |
| 3 | T000007 | 7 |
| 4 | T000008 | 8 |
| 5 | T000009 | 9 |
| 6 | T000010 | 10 |

Figure 21. Service_Transaction Table

5.3.18. OFFER

```
INSERT INTO Offer (Description, Start_Date, End_Date, Eligible_Customers)
VALUES
('10% discount on all services for premium customers', '2023-01-01',
'2023-12-31', 'Premium Customers'),
('5% cashback on online transactions', '2023-02-01', '2023-11-30', 'All
Customers'),
('Free ATM withdrawals for the first month', '2023-03-01', '2023-03-31',
'New Customers'),
('No processing fee for personal loans', '2023-04-01', '2023-06-30', 'All
Customers'),
('Double reward points on credit card usage', '2023-05-01', '2023-07-31',
'Credit Card Holders');
```

| | Offer_ID | Description | Start_Date | End_Date | Eligible_Customers |
|----|----------|--|------------|------------|--------------------------|
| 1 | 1 | 10% discount on all services for premium customers | 2023-01-01 | 2023-12-31 | Premium Customers |
| 2 | 2 | 5% cashback on online transactions | 2023-02-01 | 2023-11-30 | All Customers |
| 3 | 3 | Free ATM withdrawals for the first month | 2023-03-01 | 2023-03-31 | New Customers |
| 4 | 4 | No processing fee for personal loans | 2023-04-01 | 2023-06-30 | All Customers |
| 5 | 5 | Double reward points on credit card usage | 2023-05-01 | 2023-07-31 | Credit Card Holders |
| 6 | 6 | Complimentary cheque book for premium customers | 2023-06-01 | 2023-08-31 | Premium Customers |
| 7 | 7 | Reduced interest rates on home loans | 2023-07-01 | 2023-09-30 | Home Loan Applicants |
| 8 | 8 | Zero maintenance fees on savings accounts | 2023-08-01 | 2023-10-31 | Savings Account Holders |
| 9 | 9 | Higher overdraft limit for business accounts | 2023-09-01 | 2023-11-30 | Business Account Holders |
| 10 | 10 | Exclusive investment offers for VIP customers | 2023-10-01 | 2023-12-31 | VIP Customers |

Figure 22. Offer Table

5.3.19. CUSTOMER_OFFER

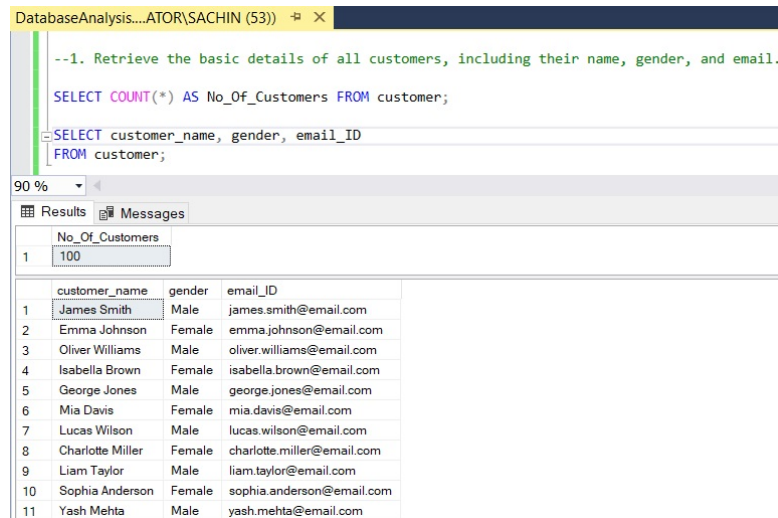
```
INSERT INTO Customer_Offer (Customer_ID, Offer_ID, Status)
VALUES
('C000045', 7, 'Redeemed'),
('C000018', 2, 'Active'),
('C000039', 4, 'Expired'),
('C000056', 9, 'Active'),
('C000067', 1, 'Redeemed');
```

| | Customer_ID | Offer_ID | Status |
|----|-------------|----------|----------|
| 1 | C000001 | 1 | Redeemed |
| 2 | C000003 | 9 | Active |
| 3 | C000004 | 3 | Active |
| 4 | C000005 | 1 | Active |
| 5 | C000006 | 10 | Expired |
| 6 | C000007 | 10 | Expired |
| 7 | C000008 | 4 | Active |
| 8 | C000009 | 3 | Active |
| 9 | C000010 | 9 | Active |
| 10 | C000011 | 6 | Redeemed |
| 11 | C000012 | 2 | Active |
| 12 | C000013 | 8 | Active |
| 13 | C000014 | 8 | Active |
| 14 | C000015 | 1 | Active |

Figure 23. Customer_Offer Table

5.4. Individual Analysis - (Sachin_23235298)

5.4.1. BASIC QUERIES



DatabaseAnalysis...ATOR\SACHIN (53))

```
--1. Retrieve the basic details of all customers, including their name, gender, and email.  
  
SELECT COUNT(*) AS No_Of_Customers FROM customer;  
  
SELECT customer_name, gender, email_ID  
FROM customer;
```

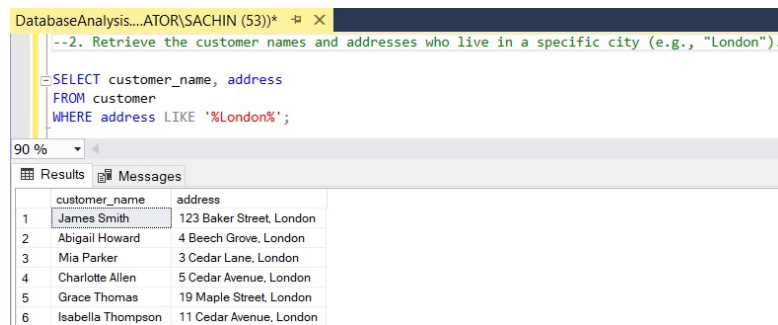
90 %

Results Messages

| No_Of_Customers |
|-----------------|
| 100 |

| customer_name | gender | email_ID |
|--------------------|--------|----------------------------|
| 1 James Smith | Male | james.smith@email.com |
| 2 Emma Johnson | Female | emma.johnson@email.com |
| 3 Oliver Williams | Male | oliver.williams@email.com |
| 4 Isabella Brown | Female | isabella.brown@email.com |
| 5 George Jones | Male | george.jones@email.com |
| 6 Mia Davis | Female | mia.davis@email.com |
| 7 Lucas Wilson | Male | lucas.wilson@email.com |
| 8 Charlotte Miller | Female | charlotte.miller@email.com |
| 9 Liam Taylor | Male | liam.taylor@email.com |
| 10 Sophia Anderson | Female | sophia.anderson@email.com |
| 11 Yash Mehta | Male | yash.mehta@email.com |

Figure 24. Fetch customer details: name, gender, email



DatabaseAnalysis...ATOR\SACHIN (53))

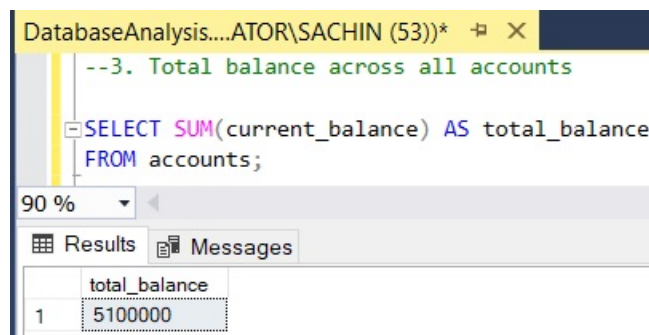
```
--2. Retrieve the customer names and addresses who live in a specific city (e.g., "London").  
  
SELECT customer_name, address  
FROM customer  
WHERE address LIKE '%London%';
```

90 %

Results Messages

| customer_name | address |
|---------------------|--------------------------|
| 1 James Smith | 123 Baker Street, London |
| 2 Abigail Howard | 4 Beech Grove, London |
| 3 Mia Parker | 3 Cedar Lane, London |
| 4 Charlotte Allen | 5 Cedar Avenue, London |
| 5 Grace Thomas | 19 Maple Street, London |
| 6 Isabella Thompson | 11 Cedar Avenue, London |

Figure 25. Customer names and addresses in a specific city



DatabaseAnalysis...ATOR\SACHIN (53))

```
--3. Total balance across all accounts  
  
SELECT SUM(current_balance) AS total_balance  
FROM accounts;
```

90 %

Results Messages

| total_balance |
|---------------|
| 1 5100000 |

Figure 26. Total balance across all accounts

5.4.2. INTERMEDIATE QUERIES

DatabaseAnalysis....ATOR\SACHIN (53))*

```
-- 4. Retrieve the details of customers who have taken out a loan with details.
SELECT c.customer_name, l.loan_id, l.date_initiated, l.loan_duration, l.interest
FROM customer c
JOIN loan l ON c.Customer_ID = l.customer_ID;
```

90 %

Results Messages

| | customer_name | loan_id | date_initiated | loan_duration | interest |
|----|------------------|---------|----------------|---------------|----------|
| 1 | George Jones | L000001 | 2020-02-15 | 60 | 3.5 |
| 2 | Ella Roberts | L000002 | 2020-03-20 | 48 | 4 |
| 3 | Mukesh Ambani | L000003 | 2020-04-25 | 36 | 4.5 |
| 4 | Charlotte Ward | L000004 | 2020-05-30 | 60 | 3.75 |
| 5 | Thomas Scott | L000005 | 2020-06-10 | 48 | 4.25 |
| 6 | Liam Kelly | L000006 | 2020-07-15 | 36 | 3.8 |
| 7 | Matthew Robinson | L000007 | 2020-08-20 | 60 | 4.1 |
| 8 | Emily Johnson | L000008 | 2020-09-25 | 48 | 3.9 |
| 9 | Emily Lewis | L000009 | 2020-10-30 | 36 | 4.2 |
| 10 | Emma King | L000010 | 2020-11-10 | 60 | 4 |
| 11 | Benjamin Scott | L000011 | 2020-12-15 | 60 | 3.6 |
| 12 | Chloe Wright | L000012 | 2021-01-20 | 48 | 3.95 |
| 13 | Christopher Cox | L000013 | 2021-02-25 | 36 | 4.05 |
| 14 | Benjamin Carter | L000014 | 2021-03-30 | 60 | 3.7 |
| 15 | Harry Mitchell | L000015 | 2021-04-10 | 48 | 4 |
| 16 | Henry Lewis | L000016 | 2021-05-15 | 36 | 4.2 |
| 17 | Liam Hill | L000017 | 2021-06-20 | 60 | 3.85 |
| 18 | Lily Roberts | L000018 | 2021-07-25 | 48 | 4.1 |
| 19 | James Smith | L000019 | 2021-08-30 | 36 | 4 |
| 20 | Grace Harris | L000020 | 2021-09-10 | 60 | 3.75 |
| 21 | Lucy Carter | L000021 | 2021-10-15 | 48 | 4 |
| 22 | Ella Adams | L000022 | 2021-11-20 | 36 | 4.5 |
| 23 | Sophie Wood | L000023 | 2021-12-25 | 60 | 4.1 |
| 24 | Megan King | L000024 | 2022-01-10 | 48 | 3.95 |
| 25 | Amelia Young | L000025 | 2022-02-15 | 36 | 4 |
| 26 | Ella Turner | L000026 | 2022-03-20 | 60 | 3.8 |
| 27 | Chloe Adams | L000027 | 2022-04-25 | 48 | 4.25 |
| 28 | Charlotte Miller | L000028 | 2022-05-30 | 36 | 4 |
| 29 | Emily Moore | L000029 | 2022-06-10 | 60 | 3.75 |
| 30 | Grace Ward | L000030 | 2022-07-15 | 48 | 4 |

Query... PREDATOR\SQLEXPRESS (16.0 RTM) PREDATOR\SACHIN (53) Bank 00:00:00 45 rows

Figure 27. Customer details with loan information

DatabaseAnalysis....ATOR\SACHIN (53))*

```
--5. Retrieve the top 5 customers with the highest account balances.
SELECT TOP (5) c.customer_name, a.current_balance
FROM customer c
JOIN accounts a ON c.Customer_ID = a.customer_ID
ORDER BY a.current_balance DESC;
```

90 %

Results Messages

| | customer_name | current_balance |
|---|---------------|-----------------|
| 1 | Emma King | 100500 |
| 2 | James Parker | 99500 |
| 3 | Grace Hall | 98500 |
| 4 | Harry Walker | 97500 |
| 5 | Olivia Davis | 96500 |

Figure 28. Top 5 customers by account balance

Bank Database Management System

| | | | |
|--|------|-------|--------------|
| DatabaseAnalysis....ATOR\SACHIN (53))* ✕ | | | |
| --6. Retrieve a summary of the total transaction amount per month for each year. | | | |
| <pre> SELECT YEAR(transaction_date) AS year, MONTH(transaction_date) AS month, SUM(amount) AS total_amount FROM transactions GROUP BY YEAR(transaction_date), MONTH(transaction_date) ORDER BY year, month; </pre> | | | |
| 90 % | | | |
| Results Messages | | | |
| | year | month | total_amount |
| 1 | 2023 | 1 | 4197882 |
| 2 | 2023 | 2 | 4042157 |
| 3 | 2023 | 3 | 5229164 |
| 4 | 2023 | 4 | 4755377 |
| 5 | 2023 | 5 | 5424603 |
| 6 | 2023 | 6 | 4037336 |
| 7 | 2023 | 7 | 5837558 |
| 8 | 2023 | 8 | 4681845 |
| 9 | 2023 | 9 | 4459243 |
| 10 | 2023 | 10 | 4703778 |
| 11 | 2023 | 11 | 4350376 |
| 12 | 2023 | 12 | 5336760 |
| 13 | 2024 | 1 | 3744693 |

Figure 29. Monthly transaction summary per year

| | | | | | |
|---|-------------|-----------|------------|-------------------------|-------------------------|
| DatabaseAnalysis....ATOR\SACHIN (53))* ✕ | | | | | |
| <pre> --7. Retrieve a detailed audit trail for all updates made to customer records, -- including who made the changes and when. SELECT a.Entity_Type, a.Entity_ID, a.Changed_By, a.Change_Date, a.Change_Details FROM Audit a WHERE a.Entity_Type = 'Customer' AND a.Change_Type = 'Update' ORDER BY a.Change_Date DESC; </pre> | | | | | |
| 90 % | | | | | |
| Results Messages | | | | | |
| | Entity_Type | Entity_ID | Changed_By | Change_Date | Change_Details |
| 1 | Customer | 83 | E00088 | 2024-08-24 21:40:58.883 | Updated customer record |
| 2 | Customer | 86 | E00027 | 2024-08-24 21:40:58.883 | Updated customer record |
| 3 | Customer | 87 | E00020 | 2024-08-24 21:40:58.883 | Updated customer record |
| 4 | Customer | 75 | E00006 | 2024-08-24 21:40:58.843 | Updated customer record |
| 5 | Customer | 76 | E00093 | 2024-08-24 21:40:58.843 | Updated customer record |
| 6 | Customer | 77 | E00013 | 2024-08-24 21:40:58.843 | Updated customer record |
| 7 | Customer | 78 | E00076 | 2024-08-24 21:40:58.843 | Updated customer record |
| 8 | Customer | 63 | E00045 | 2024-08-24 21:40:58.840 | Updated customer record |
| 9 | Customer | 67 | E00037 | 2024-08-24 21:40:58.840 | Updated customer record |
| 10 | Customer | 69 | E00023 | 2024-08-24 21:40:58.840 | Updated customer record |
| 11 | Customer | 73 | E00045 | 2024-08-24 21:40:58.840 | Updated customer record |
| 12 | Customer | 50 | E00046 | 2024-08-24 21:40:58.837 | Updated customer record |
| 13 | Customer | 53 | E00093 | 2024-08-24 21:40:58.837 | Updated customer record |
| 14 | Customer | 55 | E00019 | 2024-08-24 21:40:58.837 | Updated customer record |
| 15 | Customer | 60 | E00005 | 2024-08-24 21:40:58.837 | Updated customer record |
| 16 | Customer | 36 | E00047 | 2024-08-24 21:40:58.833 | Updated customer record |
| 17 | Customer | 37 | E00023 | 2024-08-24 21:40:58.833 | Updated customer record |
| 18 | Customer | 39 | E00080 | 2024-08-24 21:40:58.833 | Updated customer record |
| 19 | Customer | 40 | E00004 | 2024-08-24 21:40:58.833 | Updated customer record |
| 20 | Customer | 21 | E00043 | 2024-08-24 21:40:58.830 | Updated customer record |
| 21 | Customer | 28 | E00076 | 2024-08-24 21:40:58.830 | Updated customer record |
| 22 | Customer | 5 | E00069 | 2024-08-24 21:40:58.827 | Updated customer record |
| 23 | Customer | 6 | E00051 | 2024-08-24 21:40:58.827 | Updated customer record |
| 24 | Customer | 7 | E00093 | 2024-08-24 21:40:58.827 | Updated customer record |
| 25 | Customer | 9 | E00056 | 2024-08-24 21:40:58.827 | Updated customer record |
| 26 | Customer | 12 | E00098 | 2024-08-24 21:40:58.827 | Updated customer record |
| 27 | Customer | 18 | E00003 | 2024-08-24 21:40:58.827 | Updated customer record |
| 28 | Customer | 19 | E00012 | 2024-08-24 21:40:58.827 | Updated customer record |

Figure 30. Customer update audit trail: who and when

5.4.3. ADVANCED QUERIES

DatabaseAnalysis...ATOR\SACHIN (53))* ✕

```
--8. Top 10 Rank customers based on their account balance within each branch
-- and return their rank along with their details.

SELECT TOP (10)
    c.customer_name,
    b.branch_address,
    a.current_balance,
    RANK() OVER (PARTITION BY b.branch_id ORDER BY a.current_balance DESC) AS balance_rank
FROM customer c
JOIN accounts a ON c.Customer_ID = a.customer_ID
JOIN branch b ON a.sort_code = b.sort_code;
```

90 %

Results Messages

| | customer_name | branch_address | current_balance | balance_rank |
|----|-------------------|---------------------|-----------------|--------------|
| 1 | Emma King | 123 Bank St. London | 100500 | 1 |
| 2 | James Parker | 123 Bank St. London | 99500 | 2 |
| 3 | Grace Hall | 123 Bank St. London | 98500 | 3 |
| 4 | Harry Walker | 123 Bank St. London | 97500 | 4 |
| 5 | Olivia Davis | 123 Bank St. London | 96500 | 5 |
| 6 | George White | 123 Bank St. London | 95500 | 6 |
| 7 | Isabella Thompson | 123 Bank St. London | 94500 | 7 |
| 8 | Daniel Martin | 123 Bank St. London | 93500 | 8 |
| 9 | Megan Harris | 123 Bank St. London | 92500 | 9 |
| 10 | Emily Lewis | 123 Bank St. London | 91500 | 10 |

Figure 31. Top 10 customers by account balance per branch with rank and details

DatabaseAnalysis...ATOR\SACHIN (53))* ✕

```
--9. Retrieves customer information along with their account type
-- and a status indicator based on their current balance.

-- The status should be as follows:
-- "Low Balance" if the balance is below 5000,
-- "Medium Balance" if the balance is between 5000 and 20000,
-- "High Balance" if the balance is above 20000.

SELECT
    c.customer_name,
    a.acc_type,
    a.current_balance,
    CASE
        WHEN a.current_balance < 5000 THEN 'Low Balance'
        WHEN a.current_balance BETWEEN 5000 AND 20000 THEN 'Medium Balance'
        ELSE 'High Balance'
    END AS balance_status
FROM
    customer c
INNER JOIN
    accounts a
    ON c.Customer_ID = a.customer_ID
WHERE
    a.acc_type = 'Savings';
```

90 %

Results Messages

| | customer_name | acc_type | current_balance | balance_status |
|----|-----------------|----------|-----------------|----------------|
| 1 | James Smith | Savings | 2000 | Low Balance |
| 2 | Oliver Williams | Savings | 1500 | Low Balance |
| 3 | George Jones | Savings | 5500 | Medium Balance |
| 4 | Lucas Wilson | Savings | 7500 | Medium Balance |
| 5 | Liam Taylor | Savings | 9500 | Medium Balance |
| 6 | Yash Mehta | Savings | 11500 | Medium Balance |
| 7 | Benjamin Scott | Savings | 13500 | Medium Balance |
| 8 | Henry Lewis | Savings | 15500 | Medium Balance |
| 9 | Samuel White | Savings | 17500 | Medium Balance |
| 10 | Jack Baker | Savings | 19500 | Medium Balance |
| 11 | Mufid Shaikh | Savings | 21500 | High Balance |
| 12 | Jacob Young | Savings | 23500 | High Balance |
| 13 | William Martin | Savings | 25500 | High Balance |

Q... PREDATOR\SQLEXPRESS (16.0 RTM) PREDATOR\SACHIN (53) Bank 00:00:00 50 rows

Figure 32. Customer info with account type and balance status (Low, Medium, High)

Using Stored Procedure

```

CREATE PROCEDURE TransferFunds
    @FromAccountID NVARCHAR(8),
    @ToAccountID NVARCHAR(8),
    @Amount DECIMAL(18, 2)
AS
BEGIN
    BEGIN TRY
        -- Start the transaction
        BEGIN TRANSACTION;

        -- Debit the amount from the sender's account
        UPDATE accounts
        SET current_balance = current_balance - @Amount
        WHERE customer_ID = @FromAccountID;

        -- Credit the amount to the recipient's account
        UPDATE accounts
        SET current_balance = current_balance + @Amount
        WHERE customer_ID = @ToAccountID;

        -- If everything is successful, commit the transaction
        COMMIT TRANSACTION;

        PRINT 'Transaction completed successfully.';
    END TRY
    BEGIN CATCH
        -- Rollback the transaction if there is any error
        ROLLBACK TRANSACTION;

        PRINT 'Transaction failed. Rolled back.';
    END CATCH
END;

```

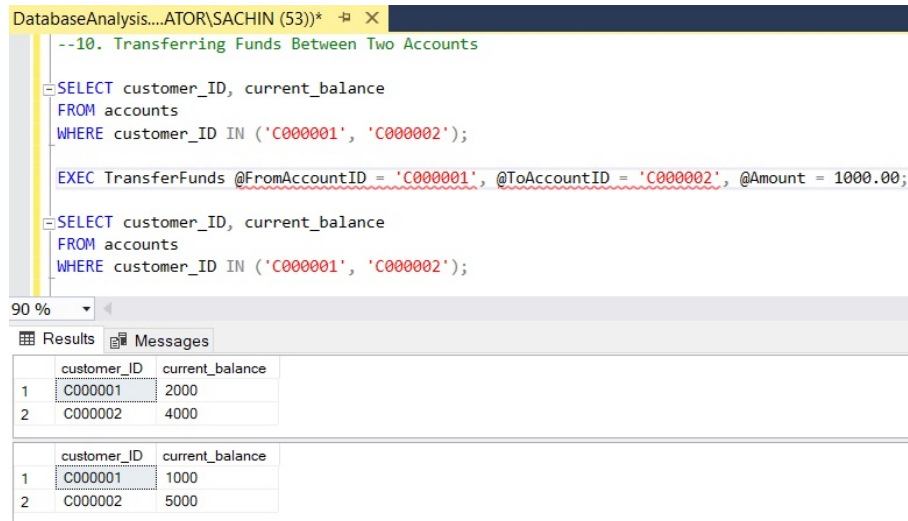


Figure 33. Funds transfer between two accounts

5.5. Query Performance and Planning

Query Performance and Planning focuses on ensuring that database queries are executed efficiently. This involves several key practices:

- **Optimization:** Fine-tuning queries to minimize execution time and resource consumption.
- **Execution Plans:** Analyzing how a database processes a query to identify potential bottlenecks.
- **Indexing:** Creating indexes to accelerate data retrieval, while balancing the potential performance impact of excessive indexing.
- **Partitioning:** Dividing large datasets into smaller, more manageable segments to improve query performance.
- **Caching:** Storing the results of frequent queries to reduce repeated processing.
- **Concurrency Management:** Ensuring that multiple queries can run simultaneously without causing conflicts.

Regular monitoring and tuning are crucial to maintaining optimal performance as data volumes and usage patterns evolve (Fritchey & Fritchey, 2018; Boggiano & Fritchey, 2019).

5.6. Query optimization Technique

SQL query optimization is the process of improving the performance and efficiency of SQL queries. Query Answering: Optimization techniques are used to query and retrieve the data quickly as well as securely. These queries would be like searching this data unsorted and improperly, making time and resources waste away without the proper optimization (Shankar et al., 2012).

5.6.1. QUERY OPTIMIZATION USING INDEXING

The purpose of an SQL index is to get the data quickly from a database. This is the biggest way by which we can boost a server's performance for queries and applications. It will be to index the table or view. On the other hand, an SQL index is a lightning-fast lookup table for searches users may perform often. An index is a succinct, efficient way of looking up rows on the database quickly. It is a really good SQL for connecting the relational tables and also can search large tables (Maesaroh et al., 2022).

Query

```
SELECT *  
FROM transactions  
WHERE customer_ID = 'C000001';
```

Step 1: Measure Query Performance Before Indexing

| Key Lookup (Clustered) | |
|--|-----------------|
| Uses a supplied clustering key to lookup on a table that has a clustered index. | |
| Physical Operation | Key Lookup |
| Logical Operation | Key Lookup |
| Estimated Execution Mode | Row |
| Storage | RowStore |
| Estimated I/O Cost | 0.003125 |
| Estimated Operator Cost | 0.0083137 (72%) |
| Estimated CPU Cost | 0.0001581 |
| Estimated Subtree Cost | 0.0083137 |
| Estimated Number of Executions | 5 |
| Estimated Number of Rows for All Executions | 5 |
| Estimated Number of Rows Per Execution | 1 |
| Estimated Row Size | 33 B |
| Ordered | True |
| Node ID | 3 |
| Object | |
| [Bank].[dbo].[transactions].[PK__transact__85C90537C368F4E4] | |
| Output List | |
| [Bank].[dbo].[transactions].acc_no, [Bank].[dbo].[transactions].amount, [Bank].[dbo].[transactions].transaction_date, [Bank].[dbo].[transactions].transaction_type | |
| Seek Predicates | |
| Seek Keys[1]: Prefix: [Bank].[dbo].[transactions].transaction_ID = Scalar Operator([Bank].[dbo].[transactions].[transaction_ID]) | |

Figure 34. Execution Plan Before Optimization

Step 2: Create the Index

```
CREATE NONCLUSTERED INDEX idx_transactions_customer_id
ON transactions (customer_ID);
```

Step 3: Measure Query Performance After Indexing

| Index Seek (NonClustered) | |
|--|-----------------|
| Scan a particular range of rows from a nonclustered index. | |
| Physical Operation | Index Seek |
| Logical Operation | Index Seek |
| Estimated Execution Mode | Row |
| Storage | RowStore |
| Estimated Operator Cost | 0.0032875 (28%) |
| Estimated I/O Cost | 0.003125 |
| Estimated Subtree Cost | 0.0032875 |
| Estimated CPU Cost | 0.0001625 |
| Estimated Number of Executions | 1 |
| Estimated Number of Rows to be Read | 5 |
| Estimated Number of Rows for All Executions | 5 |
| Estimated Number of Rows Per Execution | 5 |
| Estimated Row Size | 24 B |
| Ordered | True |
| Node ID | 1 |
| Object | |
| [Bank].[dbo].[transactions].[idx_transactions_customer_id] | |
| Output List | |
| [Bank].[dbo].[transactions].customer_ID, [Bank].[dbo].[transactions].transaction_ID | |
| Seek Predicates | |
| Seek Keys[1]: Prefix: [Bank].[dbo].[transactions].customer_ID = Scalar Operator('C000001') | |

Figure 35. Execution Plan After Optimization

In Figure 34 & 35 The enhancement, from optimizing the query was notable after adding a clustered index to the 'customer_ID' column. The cost of the 'Index ' operation now stands at **28%** with the ' Non-Index ' cost, at **72%**. This adjustment decreased the estimated I/O cost to **0.003125** from **0.0046065** leading to a smoother query execution process.

5.7. Simulating Real-Life Operations: Concurrency Control

A scenario was created to test the concurrency control of the database. At first, the balance of £20,500 belonged to account number 10000020. Two simultaneous transactions took place: a £500 withdrawal and a £300 deposit.

| Results | | Messages | | | | |
|---------|-------------|----------|----------|------------|-----------------|-----------|
| | customer_ID | acc_no | acc_type | open_date | current_balance | sort_code |
| 1 | C000020 | 10000020 | Current | 2020-01-20 | 20500 | SC001234 |

Figure 36. Before Applying Scenario

During Session 1, the withdrawal was conducted by running the SQL query provided.

```
BEGIN TRANSACTION;
UPDATE Accounts
SET Current_balance = Current_balance - 500
WHERE acc_no = 10000020;
```

Initiating this query triggered a transaction, securing the row linked to account number 10000020, and decreased the balance by £500. The balance stood at £20,000 following the operation, however, the transaction had not been executed.

During Session 2, the deposit was carried out by running the given SQL query.

```
BEGIN TRANSACTION;
UPDATE Accounts
SET Current_balance = Current_balance + 300
WHERE acc_no = 10000020;
```

This query also began a transaction, locked the same row, but this transaction was not made yet even though it was committed, since the previous transaction was still running this transaction was put on hold.

After Session 1 was then committed as below:

```
COMMIT;
```

Both the transactions effected the account after committing the second account.

| Results | | Messages | | | | |
|---------|-------------|----------|----------|------------|-----------------|-----------|
| | customer_ID | acc_no | acc_type | open_date | current_balance | sort_code |
| 1 | C000020 | 10000020 | Current | 2020-01-20 | 20300 | SC001234 |

Figure 37. After Applying Scenario

This scenario demonstrates that the database's concurrency control mechanism correctly managed the simultaneous operations, ensuring the final balance accurately reflected both the withdrawal and deposit. The final result was a net decrease of £200 from the initial balance of £20,500, confirming that the database effectively handled concurrent transactions and maintained data integrity.

6. Conclusion

A Bank Management System is able to manage the main functions of a bank, data security related to accounts and transactions, loan timings as well all types of banking services. This is built with an organized, normalized database schema and future-ready design to adhere to standards and regulatory board compliances. This could be applied around several business rules and lead to a rich entity-relationship model. The system has been optimized for extensive query optimization and is able to respond in real-time while providing high throughput, greater than the speed of equivalent systems with a true concurrency-control mechanism. It is affordable, scalable conforms to all modern banking standards without betraying the faith and stability of customers.

References

- Beeri, C., Bernstein, P. A., and Goodman, N. A sophisticate's introduction to database normalization theory. In *Readings in artificial intelligence and databases*, pp. 468–479. Elsevier, 1989.
- Boggiano, T. and Fritchey, G. *Query Store for SQL Server 2019*. Springer, 2019.
- Fritchey, G. and Fritchey, G. Sql query performance tuning. *SQL Server 2017 Query Performance Tuning: Troubleshoot and Optimize Query Performance*, pp. 1–22, 2018.
- Hogan, R. *A practical guide to database design*. CRC Press, 2018.
- Laender, A. H. F., Casanova, M. A., de Carvalho, A. P., and Ridolfi, L. An analysis of sql integrity constraints from an entity-relationship model perspective. *Information Systems*, 19(4):331–358, 1994.
- Maesaroh, S., Gunawan, H., Lestari, A., Tsaaurie, M. S. A., and Fauji, M. Query optimization in mysql database using index. *International Journal of Cyber and IT Service Management*, 2(2):104–110, 2022.
- Palakurthi, A., Ruthu, S., Akula, A., and Mamidi, R. Classification of attributes in a natural language query into different sql clauses. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pp. 497–506, 2015.
- Post, G. V. *Database management systems*. PHI Learning Pvt. Limited, 2009.
- Shankar, S., Nehme, R., Aguilar-Saborit, J., Chung, A., Elhemali, M., Halverson, A., Robinson, E., Subramanian, M. S., DeWitt, D., and Galindo-Legaria, C. Query optimization in microsoft sql server pdw. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 767–776, 2012.