REPUBLIC OF THE PHILIPPINES
BICOL UNIVERSITY
POLANGUI

ISO 9001:2015
SOCOTEC Certificate No. SCP000722Q

NAME: JOHN ELMER BLANQUISCO BOBIER JR. **SUBJECT:** Information Management I

COURSE & SECTION: BSIS-2A **PROFFESOR:** Red, Guillermo Jr. V.

Midterm Laboratory for Week 7

Database Transactions and Security

Laboratory Title: Implementing Transactions and Security in MySQL

Created new database for this lab session:

- Firstly, I opened the software 'My SQL Workbench' and proceeded to Create and use database starting this activity.

```
1 • CREATE DATABASE BankingSystem;
2 • USE BankingSystem;
```

✓	1	14:22:14	CREATE DATABASE BankingSystem
✓	2	14:22:14	USE BankingSystem

Five normalized tables to simulate a real banking system

- I created the tables needed for the Banking System database. The Customers table stores personal details, while the Accounts table links to it and keeps account details like type and balance. The Transactions table records deposits, withdrawals, and transfers. The Loans table tracks loans with amount, interest, and term, and the Payments table records payments made on loans. I also used ON DELETE CASCADE so when a customer is deleted, all their related data is removed.

```sql
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY AUTO_INCREMENT,
    FullName VARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(15),
    Address TEXT
);

CREATE TABLE Accounts (
    AccountID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerID INT,
    AccountType ENUM('Savings', 'Checking', 'Business'),
    Balance DECIMAL(10,2),
    CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE CASCADE
);

CREATE TABLE Transactions (
    TransactionID INT PRIMARY KEY AUTO_INCREMENT,
    AccountID INT,
    TransactionType ENUM('Deposit', 'Withdrawal', 'Transfer'),
    Amount DECIMAL(10,2),
    TransactionDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID) ON DELETE CASCADE
);

CREATE TABLE Loans (
    LoanID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerID INT,
    LoanAmount DECIMAL(12,2),
    InterestRate DECIMAL(5,2),
    LoanTerm INT COMMENT 'Loan duration in months',
    Status ENUM('Active', 'Paid', 'Defaulted'),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE CASCADE
);

CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY AUTO_INCREMENT,
    LoanID INT,
    AmountPaid DECIMAL(10,2),
    PaymentDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (LoanID) REFERENCES Loans(LoanID) ON DELETE CASCADE
);
```

```
3  14:23:40  CREATE TABLE Customers (  CustomerID INT PRIMARY KEY AUTO_INCREMENT,  FullName VARCHAR(...  0 row(s) affected
4  14:23:40  CREATE TABLE Accounts (  AccountID INT PRIMARY KEY AUTO_INCREMENT,  CustomerID INT,  Ac...  0 row(s) affected
5  14:23:40  CREATE TABLE Transactions (  TransactionID INT PRIMARY KEY AUTO_INCREMENT,  AccountID INT, ...  0 row(s) affected
6  14:23:40  CREATE TABLE Loans (  LoanID INT PRIMARY KEY AUTO_INCREMENT,  CustomerID INT,  LoanAmo...  0 row(s) affected
7  14:23:40  CREATE TABLE Payments (  PaymentID INT PRIMARY KEY AUTO_INCREMENT,  LoanID INT,  Amoun...  0 row(s) affected
```

Populate the Customers table with 10,000 random customers

- I inserted random customer records into the Customers table using a query. It generated random names, emails, phone numbers, and addresses using the RAND() function. The email was formatted. This method allowed me to quickly add 10,000 customers instead of entering them one by one.

```
1
2  INSERT INTO Customers (FullName, Email, PhoneNumber, Address)
3  SELECT
4      CONCAT('Customer_', FLOOR(RAND() * 100000)),
5      CONCAT('user', FLOOR(RAND() * 100000), '@bank.com'),
6      CONCAT('+639', FLOOR(RAND() * 1000000000)),
7      CONCAT('Street_', FLOOR(RAND() * 10000), ', City_', FLOOR(RAND() * 100))
8  FROM
9     information_schema.tables
10 LIMIT 10000;
```

```
8  14:25:07  INSERT INTO Customers (FullName, Email, PhoneNumber, Address) SELECT  CONCAT('Customer_', FLOOR...  350 row(s) affected Records: 350  Duplicates: 0  Warnings: 0
```

generate random accounts, transactions, loans, and payments for customers

- I added data to other tables using random values. Customers got a random Savings or Checking account with a balance. The Transactions table recorded either a Deposit or Withdrawal. The Loans table stored loan details like amount, interest, and term, with status set as Active or Paid. Lastly, the Payments table recorded random loan payments. This made sure all tables had data without manual entry.

```
1  INSERT INTO Accounts (CustomerID, AccountType, Balance)
2  SELECT
3      CustomerID,
4      IF(RAND() > 0.5, 'Savings', 'Checking'),
5      ROUND(RAND() * 100000, 2)
6  FROM Customers;
```

```
INSERT INTO Loans (CustomerID, LoanAmount, InterestRate, LoanTerm,
Status)
SELECT
    CustomerID,
    ROUND(RAND() * 100000,
2),
    ROUND(RAND() * 10, 2),
    FLOOR(RAND() * 60) + 12,
    IF(RAND() > 0.5, 'Active', 'Paid')
```

```
    FLOOR(RAND() * 60) + 12,
    IF(RAND() > 0.5, 'Active', 'Paid')
FROM Customers;
INSERT INTO Payments (LoanID, AmountPaid)
SELECT
    LoanID,
    ROUND(RAND() * 5000, 2) FROM Loans;
```

```
9  14:26:14  INSERT INTO Accounts (CustomerID, AccountType, Balance) SELECT  CustomerID,  IF(RAND() > 0.5, 'Sa...  350 row(s) affected Records: 350  Duplicates: 0  Warnings: 0
10 14:26:14  INSERT INTO Transactions (AccountID, TransactionType, Amount) SELECT  AccountID,  IF(RAND() > 0.5,...  350 row(s) affected Records: 350  Duplicates: 0  Warnings: 0
11 14:26:14  INSERT INTO Loans (CustomerID, LoanAmount, InterestRate, LoanTerm, Status) SELECT  CustomerID,  R...  350 row(s) affected Records: 350  Duplicates: 0  Warnings: 0
12 14:26:14  INSERT INTO Payments (LoanID, AmountPaid) SELECT  LoanID,  ROUND(RAND() * 5000, 2) FROM Loans  350 row(s) affected Records: 350  Duplicates: 0  Warnings: 0
```

Verify the inserted data

- I used SELECT COUNT(*) queries to check the total number of Customers, Accounts, Transactions, Loans, and Payments. This confirmed that the database was properly filled with data.

```
SELECT COUNT(*) FROM Customers;
SELECT COUNT(*) FROM Accounts;
SELECT COUNT(*) FROM Transactions;
SELECT COUNT(*) FROM Loans;
SELECT COUNT(*) FROM Payments;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

COUNT(*)
350

✓	13	14:27:46	SELECT COUNT(*) FROM Customers LIMIT 0, 1000	1 row(s) returned
✓	14	14:27:46	SELECT COUNT(*) FROM Accounts LIMIT 0, 1000	1 row(s) returned
✓	15	14:27:46	SELECT COUNT(*) FROM Transactions LIMIT 0, 1000	1 row(s) returned
✓	16	14:27:46	SELECT COUNT(*) FROM Loans LIMIT 0, 1000	1 row(s) returned
✓	17	14:27:46	SELECT COUNT(*) FROM Payments LIMIT 0, 1000	1 row(s) returned

Reference:

Red, Guillermo Jr. V. (2025). Implementing Transactions and Security in MySQL. https://bulms.bicol-u.edu.ph/mod/page/view.php?id=26216.