



Name: Dazzly D. Moneda
Course/ Year/ Section: BSIS/ 2/ A

Subject: Information Management
Professor: Mr. Red Guillermo V. Jr.

Laboratory Title: Implementing Transactions and Security in MySQL

Creating a New Database

- I opened *MySQL Workbench* and created a new database for this lab session. I then selected and used the database to begin the activity.

```
1 • CREATE DATABASE BankingSystem;  
2 • USE BankingSystem;  
  
1 14:22:14 CREATE DATABASE BankingSystem  
2 14:22:14 USE BankingSystem
```

Designing a Normalized Banking System

- I designed five normalized tables to simulate a real banking system:
 - Customers Table: Stores personal details.
 - Accounts Table: Links to the Customers table and contains account details such as type and balance.
 - Transactions Table: Records deposits, withdrawals, and transfers.
 - Loans Table: Tracks loan amounts, interest rates, and terms.
 - Payments Table: Records loan payments.
- I implemented ON DELETE CASCADE, ensuring that when a customer is deleted, all associated data is removed.

```
1 • CREATE TABLE Customers (  
2   CustomerID INT PRIMARY KEY AUTO_INCREMENT,  
3   Fullname VARCHAR(100),  
4   Email VARCHAR(100) UNIQUE,  
5   PhoneNumber VARCHAR(15),  
6   Address TEXT  
7 )  
8  
9 • CREATE TABLE Accounts (  
10  AccountID INT PRIMARY KEY AUTO_INCREMENT,  
11  CustomerID INT,  
12  AccountType ENUM('Savings', 'Checking', 'Business'),  
13  Balance DECIMAL(10,2),  
14  CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
15  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE CASCADE  
16 )  
17  
18 • CREATE TABLE Transactions (  
19  TransactionID INT PRIMARY KEY AUTO_INCREMENT,  
20  AccountID INT,  
21  TransactionType ENUM('Deposit', 'Withdrawal', 'Transfer'),  
22  Amount DECIMAL(10,2),  
23  TransactionDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
24  FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID) ON DELETE CASCADE  
25 )  
26  
27 • CREATE TABLE Loans (  
28  LoanID INT PRIMARY KEY AUTO_INCREMENT,  
29  CustomerID INT,  
30  LoanAmount DECIMAL(10,2),  
31  InterestRate DECIMAL(5,2),  
32  LoanTerm INT COMMENT 'loan duration in months',  
33  Status ENUM('Active', 'Paid', 'Defaulted'),  
34  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE CASCADE  
35 )  
36  
37 • CREATE TABLE Payments (  
38  PaymentID INT PRIMARY KEY AUTO_INCREMENT,  
39  LoanID INT,  
40  AmountPaid DECIMAL(10,2),  
41  PaymentDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
42  FOREIGN KEY (LoanID) REFERENCES Loans(LoanID) ON DELETE CASCADE  
43 )  
44  
45 • CREATE TABLE Payments (  
46  PaymentID INT PRIMARY KEY AUTO_INCREMENT,  
47  LoanID INT,  
48  AmountPaid DECIMAL(10,2),  
49  PaymentDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
50  FOREIGN KEY (LoanID) REFERENCES Loans(LoanID) ON DELETE CASCADE  
51 )  
52
```

```

3 14:23:40 CREATE TABLE Customers ( CustomerID INT PRIMARY KEY AUTO_INCREMENT, FullName VARCHAR(100), 0 row(s) affected
4 14:23:40 CREATE TABLE Accounts ( AccountID INT PRIMARY KEY AUTO_INCREMENT, CustomerID INT, Amount DECIMAL(10,2), 0 row(s) affected
5 14:23:40 CREATE TABLE Transactions ( TransactionID INT PRIMARY KEY AUTO_INCREMENT, AccountID INT, Amount DECIMAL(10,2), 0 row(s) affected
6 14:23:40 CREATE TABLE Loans ( LoanID INT PRIMARY KEY AUTO_INCREMENT, CustomerID INT, LoanAmount DECIMAL(10,2), 0 row(s) affected
7 14:23:40 CREATE TABLE Payments ( PaymentID INT PRIMARY KEY AUTO_INCREMENT, LoanID INT, Amount DECIMAL(10,2), 0 row(s) affected

```

Populating the Customers Table with 10,000 Random Entries

- I inserted 10,000 random customer records into the Customers table using a query. The data included randomly generated names, emails, phone numbers, and addresses, formatted appropriately. This approach automated the process, eliminating the need for manual entry.

```

1
2 1 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;

```

14:25:07 INSERT INTO Customers (FullName, Email, PhoneNumber, Address) SELECT CONCAT('Customer_', FLOOR(RAND() * 100000)), CONCAT('user', FLOOR(RAND() * 100000), '@bank.com'), CONCAT('+639', FLOOR(RAND() * 1000000000)), CONCAT('Street_', FLOOR(RAND() * 10000), ', City_', FLOOR(RAND() * 100)) FROM information_schema.tables LIMIT 10000; 350 row(s) affected Records: 350 Duplicates: 0 Warnings: 0

Generating Random Accounts, Transactions, Loans, and Payments

- I populated the other tables with random values:
 - Each customer was assigned a Savings or Checking account with a random balance.
 - The Transactions table recorded either a Deposit or Withdrawal for each customer.
 - The Loans table stored loan details, including amount, interest rate, term, and status (Active or Paid).
 - The Payments table logged random loan payments.
- This ensured that all tables were populated efficiently without manual input.

```

1 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;

```

```

13 1 INSERT INTO Loans (CustomerID, LoanAmount, InterestRate, LoanTerm,
14 Status)
15 SELECT
16     CustomerID,
17     ROUND(RAND() * 100000,
18 2),
19     ROUND(RAND() * 10, 2),
20     FLOOR(RAND() * 60) + 12,
21     IF(RAND() > 0.5, 'Active', 'Paid')
22 FROM Customers;
23 1 INSERT INTO Payments (LoanID, AmountPaid)
24 SELECT
25     LoanID,
26     ROUND(RAND() * 5000, 2) FROM Loans;

```

14:26:14 INSERT INTO Accounts (CustomerID, AccountType, Balance) SELECT CustomerID, IF(RAND() > 0.5, 'Savings', 'Checking'), ROUND(RAND() * 100000, 2) FROM Customers; 350 row(s) affected Records: 350 Duplicates: 0 Warnings: 0
 14:26:14 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') FROM Customers; 350 row(s) affected Records: 350 Duplicates: 0 Warnings: 0
 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

Verifying Inserted Data

- I executed `SELECT COUNT(*)` queries to check the total number of records in the Customers, Accounts, Transactions, Loans, and Payments tables. This verification confirmed that the database was correctly populated.

```
1 * SELECT COUNT(*) FROM Customers;
2 * SELECT COUNT(*) FROM Accounts;
3 * SELECT COUNT(*) FROM Transactions;
4 * SELECT COUNT(*) FROM Loans;
5 * SELECT COUNT(*) FROM Payments;
6
```

Result Grid

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