Bicol University
Bicol University Polangui Campus
Polangui, Albay

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.**

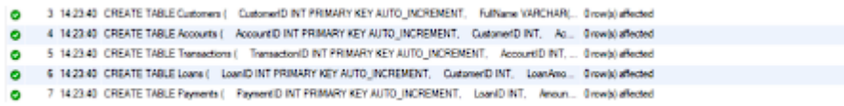
```
1 • CREATE DATABASE BankingSystem;
2 • USE BankingSystem;
```

```
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.**

```
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
);
```



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.**

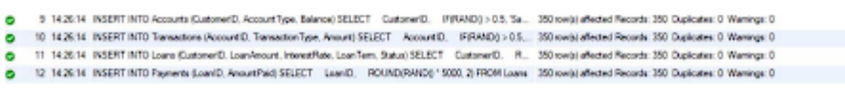
```
1
2 •  INSERT INTO Customers (FullName, Email, PhoneNumber, Address)
3    SELECT
4        CONCAT('Customer_', FLOOR(RAND() * 100000)),
5        CONCAT('user', FLOOR(RAND() * 100000), '@bank.com'),
6        CONCAT('+639', FLOOR(RAND() * 1000000000)),
7        CONCAT('Street_', FLOOR(RAND() * 10000),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


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 •  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 •  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 •  INSERT INTO Payments (LoanID, AmountPaid)
24    SELECT
25        LoanID,
26        ROUND(RAND() * 5000, 2) FROM Loans;
```



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.**