Republic of the Philippines
Bicol University Polangui
Polangui, Albay

Name:	**Christian Jeric Non**
Block & Year:	BSIS-2A
Subject:	Information Management I
Instructor:	Guillermo V. Red, Jr.

Week 7

First is creating all the database and use it.Next is to create the necessary tables like Customers , Accounts, Transactions, Loans , Payments.

```sql
CREATE DATABASE BankingSystem;
USE BankingSystem;

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
);
```

```sql
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
);
```

Then inserted data in the tables.

```sql
INSERT INTO Customers (FullName, Email, PhoneNumber, Address)
SELECT
    CONCAT('Customer_', FLOOR(RAND() * 10000)),
    CONCAT('user', FLOOR(RAND() * 1000000), '@bank.com'),
    CONCAT('+639', FLOOR(RAND() * 1000000000)),
    CONCAT('Street_', FLOOR(RAND() * 10000), ', City_', FLOOR(RAND() * 100))
FROM
   information_schema.tables
LIMIT 10000;

SELECT COUNT(*) FROM Customers;
```

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

COUNT(*)
360

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

SELECT COUNT(*) FROM Accounts;

INSERT INTO Transactions (AccountID, TransactionType, Amount)
```

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

COUNT(*)
360

```
71  INSERT INTO Transactions (AccountID, TransactionType, Amount)
72  SELECT
73      AccountID,
74      IF(RAND() > 0.5, 'Deposit', 'Withdrawal'),
75      ROUND(RAND() * 5000, 2)
76  FROM Accounts;
77
78  SELECT COUNT(*) FROM Transactions;
```

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

COUNT(*)
360

```
80  INSERT INTO Loans (CustomerID, LoanAmount, InterestRate, LoanTerm,
81  Status)
82  SELECT
83      CustomerID,
84      ROUND(RAND() * 100000,
85  2),
86      ROUND(RAND() * 10, 2),
87      FLOOR(RAND() * 60) + 12,
88      IF(RAND() > 0.5, 'Active', 'Paid')
89  FROM Customers;
90
91  SELECT COUNT(*) FROM Loans;
```

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

COUNT(*)
360

```
93  INSERT INTO Payments (LoanID, AmountPaid)
94  SELECT
95      LoanID,
96      ROUND(RAND() * 5000, 2) FROM Loans;
97
98  SELECT COUNT(*) FROM Payments;
```

Result Grid | Filter Rows: | Export:

COUNT(*)
360

Next performs two transactions: the first transfers 1000 from account 1 to account 2 while logging the transfer, and the second updates loan 5 to "Paid" and records a 5000 payment.

```
107  START TRANSACTION;
108  UPDATE Accounts SET Balance = Balance - 1000 WHERE AccountID = 1;
109  UPDATE Accounts SET Balance = Balance + 1000 WHERE AccountID = 2;
110  INSERT INTO Transactions (AccountID, TransactionType, Amount)
111  VALUES (1, 'Transfer', 1000), (2, 'Transfer', 1000);
112  COMMIT;
113
114  START TRANSACTION;
115  UPDATE Loans SET Status = 'Paid' WHERE LoanID = 5;
116  INSERT INTO Payments (LoanID, AmountPaid) VALUES (5, 5000);
117  COMMIT;
```

Next is creates two users bank_clerk with granted SELECT and UPDATE access on the BankingSystem.Accounts table and auditor with read-only access to the entire database displays their privileges, and then revokes the bank_clerk's update permission on the Accounts table.

```
119  CREATE USER 'bank_clerk'@'localhost' IDENTIFIED BY 'securepassword';
120
121  GRANT SELECT, UPDATE ON BankingSystem.Accounts TO 'bank_clerk'@'localhost';
122
123  CREATE USER 'auditor'@'localhost' IDENTIFIED BY 'readonlypass';
124  GRANT SELECT ON BankingSystem.* TO 'auditor'@'localhost';
125
126  SHOW GRANTS FOR 'bank_clerk'@'localhost';
127  SHOW GRANTS FOR 'auditor'@'localhost';
128
129  REVOKE UPDATE ON BankingSystem.Accounts FROM
130  'bank_clerk'@'localhost';
```

Simulating an attack this listed all from my accounts table.

```
132 •  SELECT * FROM Accounts WHERE CustomerID = '' OR 1=1;
```

Result Grid | Filter Rows: | Edit: | Export/

AccountID	CustomerID	AccountType	Balance	CreatedAt
1	231	Savings	74243.49	2025-03-11 18:53:14
2	154	Savings	47974.25	2025-03-11 18:53:14
3	289	Savings	15319.25	2025-03-11 18:53:14
4	22	Savings	68297.64	2025-03-11 18:53:14
5	7	Savings	59781.93	2025-03-11 18:53:14
6	228	Checking	21177.83	2025-03-11 18:53:14
7	23	Savings	64341.54	2025-03-11 18:53:14
8	229	Savings	31888.81	2025-03-11 18:53:14
9	187	Savings	26813.52	2025-03-11 18:53:14
10	206	Checking	22457.67	2025-03-11 18:53:14
11	104	Savings	85373.98	2025-03-11 18:53:14
12	88	Savings	19986.60	2025-03-11 18:53:14
13	125	Checking	11379.57	2025-03-11 18:53:14
14	165	Savings	36332.56	2025-03-11 18:53:14
15	42	Checking	87902.58	2025-03-11 18:53:14
16	257	Savings	43156.05	2025-03-11 18:53:14
17	74	Savings	28841.65	2025-03-11 18:53:14
18	183	Checking	59586.96	2025-03-11 18:53:14
19	14	Checking	72577.42	2025-03-11 18:53:14

then securely retrieves data with a prepared statement for 'Alice Johnson',

```
134 •  PREPARE stmt FROM 'SELECT * FROM Accounts WHERE CustomerID = ?';
135 •  SET @holder = 'Alice Johnson';
136 •  EXECUTE stmt USING @holder;
137 •  DEALLOCATE PREPARE stmt;
```

Next is starting a transaction, subtracts 100 from the balances of accounts with ID 1 through 2000, adds 100 to the balances of accounts with ID 2001 through 4000, and then sets a savepoint named "bulk_transaction" to allow for a rollback.

Before: id-1 = 74143.49

Result Grid | Filter Rows: | Edit: | Export/

AccountID	CustomerID	AccountType	Balance	CreatedAt
1	231	Savings	74143.49	2025-03-11 18:53:14
2	154	Savings	47874.25	2025-03-11 18:53:14
3	289	Savings	15219.25	2025-03-11 18:53:14
4	22	Savings	68197.64	2025-03-11 18:53:14
5	7	Savings	59681.93	2025-03-11 18:53:14
NULL	NULL	NULL	NULL	NULL

After: id-1 = 74043.49

Result Grid | Filter Rows: | Edit: | Expo

AccountID	CustomerID	AccountType	Balance	CreatedAt
1	231	Savings	74043.49	2025-03-11 18:53:14
2	154	Savings	47774.25	2025-03-11 18:53:14
3	289	Savings	15119.25	2025-03-11 18:53:14
4	22	Savings	68097.64	2025-03-11 18:53:14
5	7	Savings	59581.93	2025-03-11 18:53:14
NULL	NULL	NULL	NULL	NULL

Then used the rollback to return it from the how it was.

After I used COMMIT to save it.

"I read that this is for setting the highest transaction isolation level SERIALIZABLE".

```
150 •   SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;
151 •   START TRANSACTION;
152 •   UPDATE Accounts SET Balance = Balance - 500 WHERE AccountID = 3;
153 •   UPDATE Accounts SET Balance = Balance + 500 WHERE AccountID = 4;
154 •   COMMIT;
155
156 •   SELECT @@TRANSACTION_ISOLATION;
```

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

@@TRANSACTION_ISOLATION
SERIALIZABLE

ACTION OUTPUT

	#	Time	Action	Message	Duration / Fetch
3	1	18:17:40	CREATE DATABASE BankingSystem	1 row(s) affected	0.000 sec
3	2	18:17:45	USE BankingSystem	0 row(s) affected	0.000 sec
3	3	18:17:59	CREATE TABLE Customers (CustomerID INT PRIMARY KEY AUTO_INCREMENT, FullName VARCHAR(100), Email VARCHAR(100) UNIQUE, PhoneNumber VARCHAR(15), Address TEXT)	0 row(s) affected	0.032 sec
3	4	18:17:59	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)	0 row(s) affected	0.031 sec
3	5	18:17:59	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)	0 row(s) affected	0.031 sec
3	6	18:17:59	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)	0 row(s) affected	0.016 sec

3	7	18:17:59	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)	0 row(s) affected	0.031 sec
3	8	18:34:59	INSERT INTO Customers (FullName, Email, PhoneNumber, Address) SELECT CONCAT('Customer_', FLOOR(RAND() * 10000)), CONCAT('user', FLOOR(RAND() * 1000000), '@bank.com'), CONCAT('+639', FLOOR(RAND() * 1000000000)), CONCAT('Street_', FLOOR(RAND() * 10000), ', City_', FLOOR(RAND() * 100)) FROM information_schema.tables LIMIT 10000	360 row(s) affected Records: 360 Duplicates: 0 Warnings: 0	0.031 sec
3	9	18:35:31	SELECT COUNT(*) FROM Customers LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	10	18:47:29	INSERT INTO Transactions (AccountID, TransactionType, Amount) SELECT AccountID, IF(RAND() > 0.5, 'Deposit', 'Withdrawal'), ROUND(RAND() * 5000, 2) FROM Accounts	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
3	11	18:48:32	SELECT COUNT(*) FROM Accounts LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	12	18:49:04	INSERT INTO Transactions (AccountID, TransactionType, Amount) SELECT AccountID, IF(RAND() > 0.5, 'Deposit', 'Withdrawal'), ROUND(RAND() * 5000, 2) FROM Accounts	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.015 sec
3	13	18:50:45	SELECT * FROM Customers LIMIT 0, 1000	360 row(s) returned	0.000 sec / 0.000 sec
0	14	18:52:53	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)	Error Code: 1050. Table 'accounts' already exists	0.000 sec
3	15	18:53:14	INSERT INTO Accounts (CustomerID, AccountType, Balance) SELECT CustomerID, IF(RAND() > 0.5, 'Savings', 'Checking'), ROUND(RAND() * 100000, 2) FROM Customers	360 row(s) affected Records: 360 Duplicates: 0 Warnings: 0	0.000 sec
3	16	18:53:14	INSERT INTO Transactions (AccountID, TransactionType, Amount) SELECT AccountID, IF(RAND() > 0.5, 'Deposit', 'Withdrawal'), ROUND(RAND() * 5000, 2) FROM Accounts	360 row(s) affected Records: 360 Duplicates: 0 Warnings: 0	0.015 sec
3	17	18:54:30	SELECT COUNT(*) FROM Accounts LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	18	18:56:55	SELECT COUNT(*) FROM Customers LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	19	18:56:55	SELECT COUNT(*) FROM Accounts LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	20	18:56:55	SELECT COUNT(*) FROM Transactions LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

3	21	18:56:55	SELECT COUNT(*) FROM Loans LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	22	18:56:55	SELECT COUNT(*) FROM Payments LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	23	18:57:11	SELECT COUNT(*) FROM Transactions LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	24	18:58:20	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	360 row(s) affected Records: 360 Duplicates: 0 Warnings: 0	0.000 sec
3	25	18:58:29	SELECT COUNT(*) FROM Loans LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	26	19:00:12	INSERT INTO Payments (LoanID, AmountPaid) SELECT LoanID, ROUND(RAND() * 5000, 2) FROM Loans	360 row(s) affected Records: 360 Duplicates: 0 Warnings: 0	0.000 sec
3	27	19:00:15	SELECT COUNT(*) FROM Payments LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	28	19:03:20	START TRANSACTION	0 row(s) affected	0.000 sec
3	29	19:03:48	UPDATE Accounts SET Balance = Balance - 1000 WHERE AccountID = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
3	30	19:03:48	UPDATE Accounts SET Balance = Balance + 1000 WHERE AccountID = 2	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
3	31	19:03:48	INSERT INTO Transactions (AccountID, TransactionType, Amount) VALUES (1, 'Transfer', 1000), (2, 'Transfer', 1000)	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
3	32	19:03:48	COMMIT	0 row(s) affected	0.000 sec
3	33	19:12:07	SELECT * FROM Accounts WHERE CustomerID = '' OR 1=1 LIMIT 0, 1000	360 row(s) returned	0.000 sec / 0.000 sec
3	34	19:14:14	PREPARE stmt FROM 'SELECT * FROM Accounts WHERE CustomerID = ?'	0 row(s) affected Statement prepared	0.016 sec
3	35	19:14:20	SET @holder = 'Alice Johnson'	0 row(s) affected	0.000 sec
3	36	19:14:26	EXECUTE stmt USING @holder	0 row(s) returned	0.000 sec / 0.000 sec
3	37	19:14:31	DEALLOCATE PREPARE stmt	0 row(s) affected	0.000 sec
0	38	19:14:47	EXECUTE stmt USING @holder	Error Code: 1243. Unknown prepared statement handler	0.000 sec

				(stmt) given to EXECUTE	
3	39	19:16:16	START TRANSACTION	0 row(s) affected	0.000 sec
3	40	19:16:16	UPDATE Accounts SET Balance = Balance - 100 WHERE AccountID BETWEEN 1 AND 2000	360 row(s) affected Rows matched: 360 Changed: 360 Warnings: 0	0.000 sec
3	41	19:16:16	UPDATE Accounts SET Balance = Balance + 100 WHERE AccountID BETWEEN 2001 AND 4000	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
3	42	19:16:16	SAVEPOINT bulk_transaction	0 row(s) affected	0.000 sec
3	43	19:18:12	SELECT * FROM Accounts WHERE AccountID BETWEEN 1 AND 5 LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
3	44	19:19:08	START TRANSACTION	0 row(s) affected	0.000 sec
3	45	19:19:08	UPDATE Accounts SET Balance = Balance - 100 WHERE AccountID BETWEEN 1 AND 2000	360 row(s) affected Rows matched: 360 Changed: 360 Warnings: 0	0.000 sec
3	46	19:19:08	UPDATE Accounts SET Balance = Balance + 100 WHERE AccountID BETWEEN 2001 AND 4000	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
3	47	19:19:08	SAVEPOINT bulk_transaction	0 row(s) affected	0.000 sec
3	48	19:19:12	SELECT * FROM Accounts WHERE AccountID BETWEEN 1 AND 5 LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
3	49	19:22:47	ROLLBACK TO bulk_transaction	0 row(s) affected	0.000 sec
3	50	19:23:40	COMMIT	0 row(s) affected	0.000 sec
3	51	19:24:03	SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE	0 row(s) affected	0.000 sec
3	52	19:24:03	START TRANSACTION	0 row(s) affected	0.000 sec
3	53	19:24:03	UPDATE Accounts SET Balance = Balance - 500 WHERE AccountID = 3	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
3	54	19:24:03	UPDATE Accounts SET Balance = Balance + 500 WHERE AccountID = 4	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
3	55	19:24:03	COMMIT	0 row(s) affected	0.016 sec
3	56	19:24:03	SELECT @@TRANSACTION_ISOLATION LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec