**Login**

**// Create SQL Command & Parameters to prevent SQL Injection**

SqlCommand cmd = new SqlCommand();

cmd.CommandText = @"

SELECT EmployeeID, EmployeeFirstName + ' ' + EmployeeLastName AS FullName, Position

FROM Employee

WHERE EmployeeEmailAddress = @Email AND Password = @Password;

";

cmd.Parameters.Add("@Email", SqlDbType.VarChar, 50).Value = txtEmail.Text;

cmd.Parameters.Add("@Password", SqlDbType.Int).Value = parsedPassword;

**Deposit**

**// Prepare SQL command using parameters**

SqlCommand cmd = new SqlCommand();

cmd.CommandText = @"

UPDATE Account

SET Balance = Balance + @Amount

WHERE AccountID = @AccountID;

INSERT INTO [Transaction] (

EmployeeID,

LocationID,

AccountID,

TransactionDate,

TransactionType,

AmountTransferred

)

VALUES (

@EmployeeID,

@LocationID,

@AccountID,

GETDATE(),

'Deposit',

@Amount

);

";

cmd.Parameters.Add("@AccountID", SqlDbType.Int).Value = Convert.ToInt32(txtAccountID.Text);

cmd.Parameters.Add("@Amount", SqlDbType.Decimal).Value = depositAmount;

cmd.Parameters["@Amount"].Precision = 10;

cmd.Parameters["@Amount"].Scale = 2;

cmd.Parameters.Add("@EmployeeID", SqlDbType.Int).Value = Session.EmployeeID;

cmd.Parameters.Add("@LocationID", SqlDbType.Int).Value = Convert.ToInt32(txtLocationID.Text);

**Withdrawal**

**// SQL Command (Checks for insufficient funds)**

SqlCommand cmd = new SqlCommand();

cmd.CommandText = @"

IF EXISTS (

SELECT 1 FROM Account

WHERE AccountID = @AccountID AND Balance >= @Amount

)

BEGIN

UPDATE Account

SET Balance = Balance - @Amount

WHERE AccountID = @AccountID;

INSERT INTO [Transaction] (

EmployeeID,

LocationID,

AccountID,

TransactionDate,

TransactionType,

AmountTransferred

)

VALUES (

@EmployeeID,

@LocationID,

@AccountID,

GETDATE(),

'Withdrawal',

@Amount

);

END

ELSE

BEGIN

RAISERROR('Insufficient funds.', 16, 1);

END;

";

// Add SQL parameters

cmd.Parameters.Add("@AccountID", SqlDbType.Int).Value = Convert.ToInt32(txtAccountID.Text);

cmd.Parameters.Add("@Amount", SqlDbType.Decimal).Value = withdrawalAmount;

cmd.Parameters["@Amount"].Precision = 10;

cmd.Parameters["@Amount"].Scale = 2;

cmd.Parameters.Add("@EmployeeID", SqlDbType.Int).Value = Session.EmployeeID;

cmd.Parameters.Add("@LocationID", SqlDbType.Int).Value = Convert.ToInt32(txtLocationID.Text);

**Create Account**

**// Constructing SQL command**

try

{

SqlCommand cmd = new SqlCommand();

cmd.CommandText = @"

DECLARE @NewAccountID INT;

INSERT INTO Account (

CustomerID,

AccountTypeID,

Balance,

StartDate,

Description,

Principal,

[Interest/Return],

Term

)

VALUES (

@CustomerID,

@AccountTypeID,

@Balance,

GETDATE(),

@Description,

@Principal,

@InterestReturn,

@Term

);

SET @NewAccountID = SCOPE\_IDENTITY();

INSERT INTO [Transaction] (

EmployeeID,

LocationID,

AccountID,

TransactionDate,

TransactionType,

AmountTransferred

)

VALUES (

@EmployeeID,

@LocationID,

@NewAccountID,

GETDATE(),

@TransactionType,

@AmountTransferred

);

";

// Shared parameters

cmd.Parameters.Add("@CustomerID", SqlDbType.Int).Value = Convert.ToInt32(txtCustomerID.Text);

cmd.Parameters.Add("@AccountTypeID", SqlDbType.Int).Value = Convert.ToInt32(txtAccountType.Text);

cmd.Parameters.Add("@Balance", SqlDbType.Decimal).Value = balance;

cmd.Parameters["@Balance"].Precision = 10;

cmd.Parameters["@Balance"].Scale = 2;

cmd.Parameters.Add("@Description", SqlDbType.VarChar, 200).Value = txtDescription.Text;

cmd.Parameters.Add("@EmployeeID", SqlDbType.Int).Value = Session.EmployeeID;

cmd.Parameters.Add("@LocationID", SqlDbType.Int).Value = Convert.ToInt32(txtLocation.Text);

cmd.Parameters.Add("@TransactionType", SqlDbType.VarChar, 30).Value = "Deposit";

cmd.Parameters.Add("@AmountTransferred", SqlDbType.Decimal).Value = balance;

cmd.Parameters["@AmountTransferred"].Precision = 10;

cmd.Parameters["@AmountTransferred"].Scale = 2;

// If its checking, set all non checking fields to null

if (isChecking)

{

cmd.Parameters.Add("@Principal", SqlDbType.Decimal).Value = DBNull.Value;

cmd.Parameters["@Principal"].Precision = 10;

cmd.Parameters["@Principal"].Scale = 2;

cmd.Parameters.Add("@InterestReturn", SqlDbType.Decimal).Value = DBNull.Value;

cmd.Parameters["@InterestReturn"].Precision = 4;

cmd.Parameters["@InterestReturn"].Scale = 4;

cmd.Parameters.Add("@Term", SqlDbType.Int).Value = DBNull.Value;

}

// if its non checking, process entered values

else

{

cmd.Parameters.Add("@Principal", SqlDbType.Decimal).Value = principal;

cmd.Parameters["@Principal"].Precision = 10;

cmd.Parameters["@Principal"].Scale = 2;

cmd.Parameters.Add("@InterestReturn", SqlDbType.Decimal).Value = interestReturn;

cmd.Parameters["@InterestReturn"].Precision = 4;

cmd.Parameters["@InterestReturn"].Scale = 4;

cmd.Parameters.Add("@Term", SqlDbType.Int).Value = term;

}

**Create Customer**

**// SQL commands**

SqlCommand cmd = new SqlCommand();

cmd.CommandText = @"

INSERT INTO Customer (

CustomerFirstName,

CustomerLastName,

CustomerAddress,

CustomerPhoneNumber,

CustomerEmailAddress,

JoinDate

)

VALUES (

@FirstName,

@LastName,

@Address,

@PhoneNumber,

@Email,

@JoinDate

);

";

cmd.Parameters.Add("@FirstName", System.Data.SqlDbType.VarChar, 50).Value = txtFirstName.Text;

cmd.Parameters.Add("@LastName", System.Data.SqlDbType.VarChar, 50).Value = txtLastName.Text;

cmd.Parameters.Add("@Address", System.Data.SqlDbType.VarChar, 200).Value = txtAddress.Text;

cmd.Parameters.Add("@PhoneNumber", System.Data.SqlDbType.VarChar, 15).Value = txtPhone.Text;

cmd.Parameters.Add("@Email", System.Data.SqlDbType.VarChar, 50).Value = txtEmail.Text;

cmd.Parameters.Add("@JoinDate", System.Data.SqlDbType.Date).Value = DateTime.Now.Date;

**View Customer Info**

**// Construct SQL Command for getting customer names**

SqlCommand cmd = new SqlCommand();

cmd.CommandText = @"

SELECT CustomerID, CustomerFirstName + ' ' + CustomerLastName AS FullName

FROM Customer

ORDER BY CustomerFirstName, CustomerLastName

";

**Customer Account Info**

**// Customer Info dgv**

SqlCommand cmd = new SqlCommand();

cmd.CommandText = @"

SELECT CustomerID, JoinDate, CustomerPhoneNumber, CustomerEmailAddress, CustomerAddress

FROM Customer

WHERE CustomerID = @CustomerID

";

cmd.Parameters.Add("@CustomerID", SqlDbType.Int).Value = selectedCustomerId;

**// Account List dgv**

cmd = new SqlCommand();

cmd.CommandText = @"

SELECT AccountID, Description, Balance

FROM Account

WHERE CustomerID = @CustomerID

";

cmd.Parameters.Add("@CustomerID", SqlDbType.Int).Value = selectedCustomerId;

**Account Info**

**// Account Details dgv**

SqlCommand cmd = new SqlCommand();

cmd.CommandText = @"

SELECT

a.AccountID,

at.AccountType,

a.Balance,

a.StartDate,

a.Description,

a.Principal,

a.[Interest/Return],

a.Term

FROM Account a

JOIN AccountType at ON a.AccountTypeID = at.AccountTypeID

WHERE a.AccountID = @AccountID

";

cmd.Parameters.Add("@AccountID", SqlDbType.Int).Value = selectedAccountId;

**// Transaction dgv**

cmd = new SqlCommand();

cmd.CommandText = @"

SELECT

t.TransactionDate,

t.TransactionType,

t.AmountTransferred,

l.LocationAddress,

l.RoutingNumber,

e.EmployeeLastName

FROM [Transaction] t

JOIN Location l ON t.LocationID = l.LocationID

JOIN Employee e ON t.EmployeeID = e.EmployeeID

WHERE t.AccountID = @AccountID

ORDER BY t.TransactionDate DESC

";

cmd.Parameters.Add("@AccountID", SqlDbType.Int).Value = selectedAccountId;