|  |  |
| --- | --- |
| CMPG 223  Group 15 | All SQL  This document contains all the SQL used in the system |

Forms Where SQL is Used

[Table Creation 2](#_Toc144645395)

[frmWelcome 5](#_Toc144645396)

[frmOrders 8](#_Toc144645397)

[frmAdmin 11](#_Toc144645398)

[frmAddDriver 16](#_Toc144645399)

[frmProfile 17](#_Toc144645400)

# Table Creation

This is the code used to generate the tables in visual studio:

Users:

CREATE TABLE [dbo].[Users] (

[User\_ID] INT IDENTITY (1, 1) NOT NULL,

[User\_Name] NVARCHAR (30) NULL,

[User\_Surname] NVARCHAR (30) NULL,

[User\_Email] NVARCHAR (50) NULL,

[User\_Password] NVARCHAR (30) NULL,

[User\_Role] NVARCHAR (20) NULL,

PRIMARY KEY CLUSTERED ([User\_ID] ASC)

);

Restaurants:

CREATE TABLE [dbo].[Restaurants] (

[Restaurant\_ID] INT NOT NULL,

[Restaurant\_Name] NVARCHAR (50) NULL,

[Restaurant\_Location] NVARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([Restaurant\_ID] ASC)

);

Menu\_Items:

CREATE TABLE [dbo].[MenuItems] (

[Menu\_ID] INT NOT NULL,

[Item\_Desc] NVARCHAR (150) NULL,

[Item\_Price] MONEY NULL,

PRIMARY KEY CLUSTERED ([Menu\_ID] ASC)

);

Restaurant-Menu Items:

CREATE TABLE [dbo].[RestaurantMenu] (

[RestMenu\_ID] INT NOT NULL,

[Restaurant\_ID] INT NULL,

[Menu\_ID] INT NULL,

PRIMARY KEY CLUSTERED ([RestMenu\_ID] ASC),

FOREIGN KEY ([Restaurant\_ID]) REFERENCES [dbo].[Restaurants] ([Restaurant\_ID]),

FOREIGN KEY ([Menu\_ID]) REFERENCES [dbo].[MenuItems] ([Menu\_ID])

);

Delivery Points:

CREATE TABLE [dbo].[DeliveryLocations] (

[Location\_ID] INT NOT NULL,

[Location\_Name] NVARCHAR(50) NULL,

[Longitude] FLOAT (53) NULL,

[Latidue] FLOAT (53) NULL,

PRIMARY KEY CLUSTERED ([Location\_ID] ASC)

);

Orders:

CREATE TABLE [dbo].[Orders] (

[Order\_ID] INT IDENTITY (1, 1) NOT NULL,

[Customer\_ID] INT NULL,

[Driver\_ID] INT NULL,

[DP\_ID] INT NULL,

[Restaurant\_ID] INT NULL,

[Order\_Price] MONEY NULL,

[Menu\_Items\_IDs] NVARCHAR (150) NULL,

[Date] DATE NULL,

[Status] NVARCHAR (25) NULL,

PRIMARY KEY CLUSTERED ([Order\_ID] ASC),

FOREIGN KEY ([Customer\_ID]) REFERENCES [dbo].[Users] ([User\_ID]),

FOREIGN KEY ([Driver\_ID]) REFERENCES [dbo].[Users] ([User\_ID]),

FOREIGN KEY ([Restaurant\_ID]) REFERENCES [dbo].[Restaurants] ([Restaurant\_ID])

);

# frmWelcome

This form was used for logging in, signing up and giving access to users:

private bool CheckIfUserExists(string email)

{

// Open the connection

Con.Open();

// Check if the user is already in the system

SqlCommand cmd = new SqlCommand("SELECT COUNT(\*) FROM Users WHERE User\_Email = @Email", Con);

cmd.Parameters.AddWithValue("@Email", email);

// Execute the command and get the inserted user ID

int iID = Convert.ToInt32(cmd.ExecuteScalar());

// Close the connection

Con.Close();

// Return true if the user exists, false otherwise

return iID > 0;

}

private int GetHighestUserID()

{

int highestUserID = 0;

try

{

// Open the connection

Con.Open();

// Execute the command to retrieve the highest User\_ID

SqlCommand cmd = new SqlCommand("SELECT MAX(User\_ID) FROM Users", Con);

highestUserID = Convert.ToInt32(cmd.ExecuteScalar());

// Close the connection

Con.Close();

}

catch (Exception ex)

{

// Handle any errors

MessageBox.Show(ex.Message.ToString(), "Database Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

return highestUserID;

}

private int AddUserToDatabase(string name, string surname, string email, string password)

{

// Open the connection

Con.Open();

// Add user to Users Table

SqlCommand cmd = new SqlCommand("INSERT INTO Users (User\_Name, User\_Surname, User\_Email, User\_Password, User\_Role) VALUES (@Name, @Surname, @Email, @Password, @Role)", Con);

cmd.Parameters.AddWithValue("@Name", name);

cmd.Parameters.AddWithValue("@Surname", surname);

cmd.Parameters.AddWithValue("@Email", email);

cmd.Parameters.AddWithValue("@Password", password);

cmd.Parameters.AddWithValue("@Role", "Customer"); //By default a user is given the Customer role

// Execute the command

int iResult = cmd.ExecuteNonQuery();

// Close the connection

Con.Close();

return iResult;

}

**Log In Functionality:**

private void btnLogIn\_Click(object sender, EventArgs e)

{

// Get user inputs

string sEmail = txtLogInEmail.Text.ToLower(); //all emails are forced lowercase

string sPassword = txtLogInPassword.Text;

try

{

// Open the connection

Con.Open();

// Check if the user exists and retrieve the user's role and ID

SqlCommand cmd = new SqlCommand("SELECT User\_ID, User\_Role FROM Users WHERE User\_Email = @Email AND User\_Password = @Password", Con);

cmd.Parameters.AddWithValue("@Email", sEmail);

cmd.Parameters.AddWithValue("@Password", sPassword);

// Execute the command and get the user's role and ID

SqlDataReader reader = cmd.ExecuteReader();

if (reader.Read())

{

int userId = Convert.ToInt32(reader["User\_ID"]);

string sRole = Convert.ToString(reader["User\_Role"]);

// Close the reader

reader.Close();

// Close the connection

Con.Close();

// Display a success message

MessageBox.Show("Logged in successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

// Sign User In

ApproveUser(userId, sRole);

}

else

{

// Close the reader

reader.Close();

// Close the connection

Con.Close();

// Display an error message if the user does not exist or the password is incorrect

MessageBox.Show("Invalid email or password.", "Login Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch (Exception ex)

{

// Display database errors

MessageBox.Show(ex.Message.ToString(), "Database Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

# frmOrders

This form is used by customers to place orders and have the data be stored in the database:

public void ComboUpdate(System.Windows.Forms.ComboBox comboBox, string SQL, string columnName)

{

comboBox.Items.Clear();

comm = new SqlCommand(SQL, con);

dataReader = comm.ExecuteReader();

while (dataReader.Read())

{

comboBox.Items.Add(dataReader[columnName].ToString());

}

dataReader.Close();

}

Ordering Functionality:

private void btnOrder\_Click(object sender, EventArgs e)

{

try

{

if (string.IsNullOrEmpty(cbRestaurants.Text) || string.IsNullOrEmpty(cbMenuItems.Text) || string.IsNullOrEmpty(cbDeliveryPoints.Text))

{

MessageBox.Show("Please fill all the necessary requirements", "Error");

}

else

{

Date = DateTime.Now;

SQL = "INSERT INTO Orders(Customer\_ID,Driver\_ID, DP\_ID, Restaurant\_ID, Order\_Price, Menu\_Items\_IDs, Date, Status) " +

"VALUES(@Customer\_ID,@Driver\_ID, @DP\_ID, @Restaurant\_ID, @Order\_Price, @Menu\_Items\_IDs, @Date, @Status)";

if (con.State == ConnectionState.Open)

{

con.Close();

}

con.Open();

using (SqlCommand cmd = new SqlCommand(SQL, con))

{

cmd.Parameters.AddWithValue("@Customer\_ID", Customer\_ID);

cmd.Parameters.AddWithValue("@Driver\_ID", 2);

cmd.Parameters.AddWithValue("@DP\_ID", DP\_ID);

cmd.Parameters.AddWithValue("@Restaurant\_ID", Restaurant\_ID);

cmd.Parameters.AddWithValue("@Order\_Price", Order\_Price);

cmd.Parameters.AddWithValue("@Menu\_Items\_IDs", Menu\_Items\_IDs);

cmd.Parameters.AddWithValue("@Date", Date);

cmd.Parameters.AddWithValue("@Status", "Order Placed");

cmd.ExecuteNonQuery();

}

con.Close();

// Ask the user to confirm the order

string confirmationMessage = $"Order Details:\n\n" +

$"Restaurant: {cbRestaurants.Text}\n" +

$"Total Price: " + Order\_Price.ToString("c") +"\n" +

$"Delivery Point: {cbDeliveryPoints.Text}\n" +

$"Date: {Date}\n\n" +

"Do you want to confirm this order?";

DialogResult confirmationResult = MessageBox.Show(confirmationMessage, "Order Confirmation", MessageBoxButtons.YesNo);

if (confirmationResult == DialogResult.Yes)

{

// User confirmed the order, process the payment

ProcessPayment();

//Clear the order:

lstOutput.Items.Clear();

cbRestaurants.Enabled = true;

cbDeliveryPoints.SelectedIndex = -1;

cbRestaurants.SelectedIndex = -1;

cbMenuItems.SelectedIndex = -1;

}

}

}

catch (Exception ex)

{

MessageBox.Show("An error occurred: " + ex.Message, "Error");

}

}

Updating the form with relevant restaurants and menu items:

private void cbRestaurants\_SelectedIndexChanged(object sender, EventArgs e)

{

//When a restaurant is selected, its matching menu items are retrieved from RestaurantMenu table

//Then each menu item id from the restaurant is stored in an array

//The array is then used to add each item to the combo box

try

{

if (con.State == ConnectionState.Open)

{

con.Close();

}

con.Open();

// Get the selected restaurant's Menu\_ID

tempID = cbRestaurants.SelectedIndex;

Restaurant\_ID = tempID + 1;

SQL = "SELECT Menu\_ID FROM RestaurantMenu WHERE Restaurant\_ID = " + Restaurant\_ID;

comm = new SqlCommand(SQL, con);

dataReader = comm.ExecuteReader();

int[] MenuIDs = new int[50];

int iCounter = 0;

while (dataReader.Read())

{

int menuID = dataReader.GetInt32(0);

MenuIDs[iCounter++] = menuID;

}

dataReader.Close();

cbMenuItems.Items.Clear();

//Add each item to combobox:

for (int i = 0; i < iCounter; i++)

{

SqlCommand cmd = new SqlCommand("SELECT Item\_Desc FROM MenuItems WHERE Menu\_ID = " + MenuIDs[i], con);

dataReader = cmd.ExecuteReader();

if (dataReader.HasRows) // Check if there are rows to read

{

while (dataReader.Read()) // Iterate through the rows

{

cbMenuItems.Items.Add(dataReader.GetString(0));

}

}

dataReader.Close();

}

con.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occurred: " + ex.Message, "Error");

}

}

# frmAdmin

This form is the admin control panel, used to generate reports and perform certain actions on the database.

private void generateAdminReport()

{

try

{

// Create a SaveFileDialog to allow the user to choose the file location

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "Excel Files|\*.xlsx",

FileName = "FoodOnCampusReport" + DateTime.Now.ToString("yyyyMMdd") + ".xlsx"

};

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

// Define the connection string and SQL query

string conString = @"Data Source=(LocalDB)\MSSQLLocalDB; AttachDbFilename=|DataDirectory|DBMain.mdf; Integrated Security=True";

string sqlQuery = "SELECT Order\_ID as 'Order ID', Customer\_ID as 'Customer ID', Driver\_ID as 'Driver ID', Restaurant\_ID as 'Restaurant ID', Order\_Price as 'Order Price (R)', Menu\_Items\_IDs as 'Menu Items', Date, Status as 'Order Status' FROM Orders";

// Create a new SqlConnection and SqlDataAdapter

using (SqlConnection con = new SqlConnection(conString))

using (SqlDataAdapter adapter = new SqlDataAdapter(sqlQuery, con))

{

con.Open();

// Create a new DataTable to hold the data

DataTable dt = new DataTable();

adapter.Fill(dt);

// Calculate Total Order Price

double totalOrderPrice = dt.AsEnumerable().Sum(row => Convert.ToDouble(row["Order Price (R)"]));

// Calculate Average Order Total

double averageOrderTotal = totalOrderPrice / dt.Rows.Count;

// Create a new Excel workbook and worksheet

using (var workbook = new XLWorkbook())

{

var worksheet = workbook.Worksheets.Add("Orders");

// Add the data from the DataTable to the worksheet

worksheet.Cell(1, 1).InsertTable(dt);

// Add Total Order Price and Average Order Total to the worksheet

worksheet.Cell(dt.Rows.Count + 3, 2).Value = "Total Order Price (R)";

worksheet.Cell(dt.Rows.Count + 3, 3).Value = totalOrderPrice;

worksheet.Cell(dt.Rows.Count + 4, 2).Value = "Average Order Total (R)";

worksheet.Cell(dt.Rows.Count + 4, 3).Value = averageOrderTotal;

// Auto-fit columns and rows to content

worksheet.Columns().AdjustToContents();

worksheet.Rows().AdjustToContents();

// Save the workbook to the user-selected file location

workbook.SaveAs(saveFileDialog.FileName);

MessageBox.Show("Report exported successfully!", "Export Report", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Error exporting the report: " + ex.Message, "Export Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ShowDrivers(string sql)

{

try

{

using (SqlConnection con = new SqlConnection(conString))

{

con.Open();

// Create a SqlDataAdapter to fetch data from the database

SqlDataAdapter adapter = new SqlDataAdapter(sql, con);

// Create a DataSet to store the data

DataSet dataSet = new DataSet();

// Fill the DataSet with the data from the database

adapter.Fill(dataSet);

// Set the DataGridView's data source to the DataSet's first table (assuming there's only one table)

dgvDrivers.DataSource = dataSet.Tables[0];

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error Loading Drivers Data", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void LoadOrderDetails()

{

//This method records all the relevant info about all orders:

int TotalOrders = 0;

int OrdersPlaced = 0;

int InProgress = 0;

int Completed = 0;

double AverageMoneySpent = 0;

double TotalMoneySpent = 0;

try

{

using (SqlConnection con = new SqlConnection(conString))

{

con.Open();

// Total Orders: Count all rows in the Orders table

SqlCommand cmdTotalOrders = new SqlCommand("SELECT COUNT(\*) FROM Orders", con);

TotalOrders = (int)cmdTotalOrders.ExecuteScalar();

// Orders Placed: Count rows with status 'Order Placed'

SqlCommand cmdOrdersPlaced = new SqlCommand("SELECT COUNT(\*) FROM Orders WHERE Status = 'Order Placed'", con);

OrdersPlaced = (int)cmdOrdersPlaced.ExecuteScalar();

// In Progress: Count rows with status 'In Progress'

SqlCommand cmdInProgress = new SqlCommand("SELECT COUNT(\*) FROM Orders WHERE Status = 'In Progress'", con);

InProgress = (int)cmdInProgress.ExecuteScalar();

// Completed: Count rows with status 'Completed'

SqlCommand cmdCompleted = new SqlCommand("SELECT COUNT(\*) FROM Orders WHERE Status = 'Complete'", con);

Completed = (int)cmdCompleted.ExecuteScalar();

// Average Money Spent: Calculate average amount spent in all orders

SqlCommand cmdAvgMoneySpent = new SqlCommand("SELECT AVG(Order\_Price) FROM Orders", con);

AverageMoneySpent = Convert.ToDouble(cmdAvgMoneySpent.ExecuteScalar());

// Total Money Spent: Calculate sum of TotalAmount in all orders

SqlCommand cmdTotalMoneySpent = new SqlCommand("SELECT SUM(Order\_Price) FROM Orders", con);

TotalMoneySpent = Convert.ToDouble(cmdTotalMoneySpent.ExecuteScalar());

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error Reading Orders Data", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

//Show data:

lblTotalOrders.Text = TotalOrders.ToString();

lblPlaced.Text = OrdersPlaced.ToString();

lblInProgress.Text = InProgress.ToString();

lblCompleted.Text = Completed.ToString();

lblAverageSpent.Text = AverageMoneySpent.ToString("c");

lblTotalSpent.Text = TotalMoneySpent.ToString("c");

}

Delete A Driver:

private void btnDeleteDriver\_Click(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(DriverEmail))

{

try

{

using (SqlConnection con = new SqlConnection(conString))

{

con.Open();

// Query to retrieve the user's name and surname for confirmation

string selectQuery = "SELECT User\_Name, User\_Surname FROM Users WHERE User\_Email = @Email";

// Create a command to execute the query

using (SqlCommand selectCommand = new SqlCommand(selectQuery, con))

{

selectCommand.Parameters.AddWithValue("@Email", DriverEmail);

// Execute the query and read the user's name and surname

using (SqlDataReader reader = selectCommand.ExecuteReader())

{

if (reader.Read())

{

string userName = reader["User\_Name"].ToString();

string userSurname = reader["User\_Surname"].ToString();

// Ask for user confirmation

string confirmationMessage = $"Are you sure you want to delete {userName} {userSurname}?";

DialogResult confirmationResult = MessageBox.Show(confirmationMessage, "Confirm Deletion", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (confirmationResult == DialogResult.Yes)

{

// Delete the driver record

string deleteQuery = "DELETE FROM Users WHERE User\_Email = @Email";

using (SqlCommand deleteCommand = new SqlCommand(deleteQuery, con))

{

deleteCommand.Parameters.AddWithValue("@Email", DriverEmail);

int rowsAffected = deleteCommand.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Driver deleted successfully.", "Deletion Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);

// Refresh the driver list after deletion if necessary

ShowDrivers("SELECT User\_Name AS Name, User\_Surname AS Surname, User\_Email AS Email FROM Users WHERE User\_Role = 'Driver' ORDER BY User\_Name DESC");

}

else

{

MessageBox.Show("Driver deletion failed.", "Deletion Failed", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

else

{

MessageBox.Show("Driver not found.", "Driver Not Found", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Error deleting driver: " + ex.Message, "Deletion Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Please select a driver to delete.", "No Driver Selected", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

# frmAddDriver

This form is used by an admin to add a new driver to the database:

//The main part of the method is excluded for brevity and is not relevant to the SQL

if (isValid)

{

try

{

// Insert the new driver into the "Users" table with the "Driver" role

using (SqlConnection con = new SqlConnection(conString))

{

con.Open();

string insertQuery = "INSERT INTO Users (User\_Name, User\_Surname, User\_Email, User\_Password, User\_Role) VALUES (@Name, @Surname, @Email, @Password, 'Driver')";

using (SqlCommand insertCommand = new SqlCommand(insertQuery, con))

{

insertCommand.Parameters.AddWithValue("@Name", name);

insertCommand.Parameters.AddWithValue("@Surname", surname);

insertCommand.Parameters.AddWithValue("@Email", email);

insertCommand.Parameters.AddWithValue("@Password", password);

int rowsAffected = insertCommand.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Driver added successfully.", "Add Driver", MessageBoxButtons.OK, MessageBoxIcon.Information);

ClearForm();

DialogResult = DialogResult.OK;

Close();

}

else

{

MessageBox.Show("Failed to add driver.", "Add Driver", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Error adding driver: " + ex.Message, "Add Driver Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show(errorMessage, "Input Verification Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

# frmProfile

Any user can access this form and see their personal details and order history.

private void frmProfile\_Load(object sender, EventArgs e)

{

conn = new SqlConnection(@"Data Source = (LocalDB)\MSSQLLocalDB; AttachDbFilename = |DataDirectory|DBMain.mdf; Integrated Security = True");

//----------------------Get user name and set label-------------------

lblHello.Text = "";

conn.Open();

cmd = new SqlCommand("SELECT User\_Name FROM Users WHERE User\_ID = "+UserId+" ", conn);

rdr = cmd.ExecuteReader();

while (rdr.Read())

{

lblHello.Text = "Hello, " + rdr.GetValue(0).ToString();

}

conn.Close();

//--------------------------------------------------------------------

//-------------------------Fill Orders list box-----------------------

conn.Open();

cmd = new SqlCommand("SELECT Restaurant\_Name FROM Restaurants", conn);

rdr = cmd.ExecuteReader();

int countName = 0;

while (rdr.Read())

{

restaurantName[countName] = rdr.GetValue(0).ToString();

countName++;

}

conn.Close();

conn.Open();

cmd = new SqlCommand("SELECT Restaurant\_ID FROM Orders WHERE Customer\_ID = "+UserId+" ", conn);

rdr = cmd.ExecuteReader();

int count = 0;

while (rdr.Read())

{

restaurantId[count] = Convert.ToInt32(rdr.GetValue(0));

count++;

}

conn.Close();

for (int i = 0; i < count; i++)

{

restaurants[i] = restaurantName[restaurantId[i]-1];

}

lbxOrders.Items.Clear();

lbxOrders.Items.Add("Order ID\tRestaurant\tPrice\tDate");

lbxOrders.Items.Add("");

conn.Open();

cmd = new SqlCommand("SELECT Order\_ID, Order\_Price, Date FROM Orders WHERE Customer\_ID = "+UserId+" ", conn);

rdr = cmd.ExecuteReader();

int k = 0;

while (rdr.Read())

{

lbxOrders.Items.Add(rdr.GetValue(0).ToString()+"\t"+ restaurants[k] +"\t"+rdr.GetValue(1).ToString()+"\t"+rdr.GetValue(2).ToString());

k++;

}

conn.Close();

//-------------------------------------------------------------

}

private void btnUpdatePassword\_Click(object sender, EventArgs e)

{

if(tbxPassword.Text=="")

{

MessageBox.Show("Please enter a new password");

}

else

{

conn.Open();

string sql = "UPDATE Users SET User\_Password = '"+tbxPassword.Text+"' WHERE User\_ID = "+UserId+"";

cmd = new SqlCommand(sql,conn);

cmd.ExecuteNonQuery();

conn.Close();

}

}

private void btnUpdateEmail\_Click(object sender, EventArgs e)

{

if (tbxEmail.Text == "")

{

MessageBox.Show("Please enter a new Email");

}

else

{

conn.Open();

string sql = "UPDATE Users SET User\_Email = '" + tbxEmail.Text + "' WHERE User\_ID = " + UserId + "";

cmd = new SqlCommand(sql, conn);

cmd.ExecuteNonQuery();

conn.Close();

}

}

}