TU2983: Advanced Databases

Lab Notes 3:

Searching for Data and Displaying Search Results in Visual Basic .NET Objects

Written by:

Hafiz Mohd Sarim

Centre of Artificial Intelligence Technology
Fakulti Teknologi dan Sains Maklumat
Universiti Kebangsaan Malaysia

Table of Contents

PART 1: Introduction

A. Lab Objectives

1. This TU2983 lab module aims to fulfil the following objectives:

   i. To teach students how to display SQL SELECT results on different Visual Basic .NET form objects and data structures.

   ii. To teach students how to use Visual Basic .NET form objects as parameters in a dynamic SQL SELECT statement.

2. The programming language used for the TU2983 lab modules is Visual Basic .NET, while database used to hold project data will be either of the following
   a. Microsoft Access 2007 or above as the portable database
   b. IBM DB2 Enterprise Server Edition v9.7 or above as the server database.
   Students will be instructed on the exact database to use for each lab exercise and lab assignment.

3. These lab notes were written using the "**Visual Studio 2019 Enterprise**" integrated development environment (IDE) for the **Visual Basic .NET** programming language. However, the steps demonstrated in these lab notes can also be used for all versions of Visual Basic .NET in Visual Studio IDEs starting from Visual Studio .NET 2002 and above.

B. Student Requirements for this Lab

1. All students must bring the following equipment for the lab:

   - **MANDATORY REQUIREMENT:** One USB hard drive / thumb drive pen drive (to save your work)

2. All students must have the following requisite knowledge:

   - **MANDATORY REQUIREMENT:** Using the Microsoft Windows operating system.
   - **MANDATORY REQUIREMENT:** Understand the fundamentals of programming and recognize the different control statements and data structures that are generally available in all programming languages.
   - **MANDATORY REQUIREMENT:** Writing Structured Query Language code, or SQL.
   - **MANDATORY REQUIREMENT:** Creating a sample database in Microsoft Access (see TU2983 Lab Notes 1).
   - **MANDATORY REQUIREMENT:** Building windows forms in Visual Basic .NET and connecting to a Microsoft Access database (see TU2983 Lab Notes 2).

PART 2: Displaying SQL SELECT Results in VB .NET Form Objects

A. Opening an Existing VB project

1. Start Visual Studio. In the Windows desktop, click **'Start' > 'Microsoft Visual Studio 2019'**.

2. We will use the Visual Basic .NET project you created in TU2983 Lab Notes 2. In the Visual Studio start page, click on **'prj_facultyrecords_a123456'**, which is listed under the **'Recent Projects'** list. Your project will open.

3. Alternatively, you can also browse to the project folder contained in your working folder, for example **'C:\a123456_facultyrecords\prj_facultyrecords_a123456'**, and double-click on the ***.sln** file (solution file), for example **'prj_facultyrecords_a123456.sln'**.

4. Verify that you have the following forms and modules:

   - A splash screen form (**frm_splashscreen_a123456.vb**)
   - A main menu form (**frm_mainmenu_a123456.vb**)
   - A form for browsing the student list (**frm_studentlist_a123456**)
   - A module containing your global variables (**mod_globals_a123456**)

5. If you do not have all these forms and modules, please complete TU2983 Lab Notes 1 before proceeding any further.

B. Declaring the Database Connection String as a Global Variable.

1. In TU2983 Lab Notes 2, the connection string to your database is currently stored as a local variable in the **Form_Load** event in your student list form, **frm_student_list_a123456**, as follows:

```
Private Sub frm_studentlist_a123456_Load(sender As System.Object, e as System.EventArgs) Handles
MyBase.Load

        Dim myconnection as String = "Provider=Microsoft.ACE.OLEDB.12.0;Data
        Source=DB_FACULTYRECORDS_A123456.accdb;Persist Security Info=False;"

        ...

End Sub
```

2. Having this database connection string stored as a local variable in this form event, means that you will have to declare the same database connection repeatedly in other events for other VB forms. Since you will very likely connect to only one database for the entirety of your TU2983 labs and lab assignments, you can store this database connection string as a global variable.

3. Copy the connection string variable above, and paste it to the module containing your global variables, **mod_globals_a123456**.

4. Replace the 'Dim' declaration statement with 'Public' as follows:

```
Module mod_globals_a123456

        Public myconnection as String = "Provider=Microsoft.ACE.OLEDB.12.0;Data
        Source=DB_FACULTYRECORDS_A123456.accdb;Persist Security Info=False;"

End Module
```

C. Displaying Data from a SELECT Statement in a DataGridView Object

1. In the Solution Explorer list, add a new form called '**frm_coursesbydept_a123456.vb**'.

2. Double-click on the 'My Project' item in the Solution Explorer list and change your **'Startup form'** to '**frm_coursesbydept_a123456.vb**'.

3. From your ToolBox, add a **Label** to the top of the '**frm_coursesbydept_a123456.vb**' form. Change the **(Name)** property of this Label to '**lbl_title'**. Change the 'Text' property of this Label to 'Courses by Department', and redesign the font of this label as you see fit.

4. From your ToolBox, add a **DataGridView** object to the '**frm_coursesbydept_a123456.vb**' form. Change the **(Name)** property of this DataGridView object to '**grd_courses'**.

5. Open the **Form_Load** event for this form, by double-clicking on any blank space in the '**frm_coursesbydept_a123456.vb**' form.

6. Add the following code to the Form_Load event:

```
Private Sub frm_coursesbydept_a123456_Load(sender As System.Object, e as System.EventArgs)
Handles MyBase.Load

        Dim mysql as String = "SELECT * FROM TBL_COURSES_A123456"

        Dim mydatatable as New DataTable

        Dim myreader as New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        grd_courses.DataSource = mydatatable

End Sub
```

NOTE: You do not need to declare the database connection string, 'myconnection', in the code above because you have already stored the database connection string as a global variable in the module **'mod_globals_a123456.vb'**.

D. Creating a Sub-procedure

1. We would like to re-use the data access code that you entered in the Form_Load event into other object events. However, if the code remains in its current location, we would need to rewrite this code repeatedly if want to access the database in other events.

2. A sub-procedure, or Sub, is type of Function that encapsulates any block of code that you would like to call repeatedly. Unlike Functions, a sub-procedure does not return a value.

3. Underneath the '**End Sub**' of the **Form_Load** event, write the following to create a sub-procedure:

```
Private Sub refresh_grid()

End Sub
```

4. Copy the database access code from the **Form_Load** event and paste it the '**refresh_grid'** sub-procedure as follows:

```
Private Sub refresh_grid()

        Dim mysql as String = "SELECT * FROM TBL_COURSES_A123456"

        Dim mydatatable as New DataTable

        Dim myreader as New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        grd_courses.DataSource = mydatatable

End Sub
```

5. In the Form_Load event, delete the existing database access code and replace it the following function call:

```
Private Sub frm_coursesbydept_a123456_Load(sender As System.Object, e as System.EventArgs)
Handles MyBase.Load

        refresh_grid()

End Sub
```

6. Run the program. You will observe that moving the data access code to a sub-procedure will have the same result as placing your code in form events.

E. Displaying Data Manually in a ComboBox Object

1. Expand the '**frm_coursesbydept_a123456.vb**' form by dragging the form borders, and make some space in between **'lbl_title'** and **'grd_courses'**.

2. From your ToolBox, add a **ComboBox** object to the '**frm_coursesbydept_a123456.vb**' form. Change the **(Name)** property of this ComboBox object to '**cmb_dept'**.

3. A ComboBox is a drop-down list that can contain a single column of data. We would like to populate the **'cmb_dept'** ComboBox with a list of the department ID codes.

4. If we have a very short list that never changes, we can manually add values to the 'cmb_dept' ComboBox:

```
Private Sub frm_coursesbydept_a123456_Load(sender As System.Object, e as System.EventArgs)
Handles MyBase.Load

        cmb_dept.Items.Add("TU")
        cmb_dept.Items.Add("TK")
        cmb_dept.Items.Add("TR")

        refresh_grid()

End Sub
```

F. Displaying Data from a SELECT statement in a ComboBox Object

1. If we have a long list of data to display in a ComboBox object, or if the data in that list is expected to change frequently, we will have a difficult time manually changing the list whenever our data changes. In this case it is easier to lookup the most current data from a database table.

2. Delete or comment out the **cmb_dept.Items.Add** lines from the **Form_Load** event.

3. Replace these lines with the following data access code in the Form_Load event:

```
Private Sub frm_coursesbydept_a123456_Load(sender As System.Object, e as System.EventArgs)
Handles MyBase.Load

        Dim mysql As String = "SELECT FLD_DEPT_ID FROM TBL_DEPARTMENT_A123456"

        Dim mydatatable As New DataTable

        Dim myreader As New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        cmb_dept.DataSource = mydatatable
        cmb_dept.DisplayMember = "FLD_DEPT_ID"

        refresh_grid()

End Sub
```

NOTE: The only differences in the data access code for a ComboBox compared to the data access code for a DataGridView object are:

i. the SQL string, and
ii. the display object.

Unlike a DataGridView, a ComboBox can only display one column of data. Therefore you will need to indicate which column or attribute you would like to display in the ComboBox using the DisplayMember property.

4. Run the Program. You will observe that when the form loads, 'cmb_dept' will be filled with the department IDs.

G. Using Object Values as Parameters in a Dynamic SQL Statement

1. Now that we have **'cmb_dept'** as a way to select Departments, we would like to use this ComboBox to filter the courses displayed in our **'grd_courses'** DataGridView.

2. Go to your **'refresh_grid()'** sub-procedure and edit the sub-procedure declaration by adding the string input parameter **'dept_id'**:

```
Private Sub refresh_grid(dept_id as String)
```

3. Change the SQL SELECT statement in the 'refresh_grid()' sub-procedure to accept the 'dept_id' input parameter as part a WHERE condition in the SELECT statement"

```
Dim mysql As String = "SELECT * FROM TBL_COURSES_A123456 where FLD_MANAGING_DEPT='" &
dept_id & "'"
```

4. Your **'refresh_grid()'** sub-procedure should now look like the following:

```
Private Sub refresh_grid(dept_id as String)

        Dim mysql as String = "SELECT * FROM TBL_COURSES_A123456 WHERE
        FLD_MANAGING_DEPT='" & dept_id & "'"

        Dim mydatatable as New DataTable

        Dim myreader as New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        grd_courses.DataSource = mydatatable

End Sub
```

5. In your Form_Load event, change the function call for 'refresh_grid()' to take the currently displayed value in 'cmb_dept' as its input:

```
refresh_grid(cmb_dept.Text)
```

6. Your Form_Load code should now look like the following:

```
Private Sub frm_coursesbydept_a123456_Load(sender As System.Object, e as System.EventArgs)
Handles MyBase.Load

        Dim mysql As String = "SELECT FLD_DEPT_ID FROM TBL_DEPARTMENT_A123456"

        Dim mydatatable As New DataTable

        Dim myreader As New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        cmb_dept.DataSource = mydatatable
        cmb_dept.DisplayMember = "FLD_DEPT_ID"

        refresh_grid(cmb_dept.Text)

End Sub
```

7. In your Form Design View, double-click on the 'cmb_dept' ComboBox object to generate the code snippet for its default event, 'SelectedIndexChanged'.

8. Copy the same 'refresh_grid()' function call from your Form_Load event into the 'SelectedIndexChanged' code snippet as follows:

```
Private Sub cmb_dept_SelectedIndexChanged(sender As System.Object, e As System.EventArgs)
Handles cmb_dept.SelectedIndexChanged

        refresh_grid(cmb_dept.Text)

End Sub
```

9. Run the program, and change the selected Department ID in the **'cmb_dept'** ComboBox. You will observe that the data in the **'grd_courses'** DataGridView will change in tandem with the selected Department ID.

H. Changing the Appearance of a DataGridView Object

1. You will note that the **'grd_courses'** DataGridView has the attribute names of the **'tbl_courses_a123456'** table as its headers. You can change this by specifying the **Columns.HeaderText** property for each displayed column.

2. Add the following lines of code at the end of the **'refresh_grid()'** subprocedure:

```
Private Sub refresh_grid(dept_id as String)

        Dim mysql as String = "SELECT * FROM TBL_COURSES_A123456 WHERE
        FLD_MANAGING_DEPT='" & dept_id & "'"

        Dim mydatatable as New DataTable

        Dim myreader as New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        grd_courses.DataSource = mydatatable

        grd_courses.Columns(0).HeaderText = "Course Code"
        grd_courses.Columns(1).HeaderText = "Course Name"
        grd_courses.Columns(2).HeaderText = "Credits"
        grd_courses.Columns(3).HeaderText = "Managing Department"

End Sub
```

3. To change the width of each column to follow width of the data and headers, in the Form Design view, click on **'grd_courses'** and change the **'AutoSizeColumns'** property to **'AllCells'**.

4. Since this **'grd_courses'** will only be used to display existing data, change the **'AllowUsersToAddRows'** property to 'False'. This will remove the new record placeholder from the DataGridView.

5. To prevent users from altering the data in the DataGridView, click on **'grd_courses'** and change the **'ReadOnly'** property to **'True'**.

I. Displaying Data from a SELECT Statement in a ListBox Object

1. A ListBox is like a DataGridView that can only display one column of data at a time.

2. Displaying data in a ListBox, either manually or by using data access to lookup data in a database table, is identical to the method for displaying data in a ComboBox. You will need to specify the DataSource property and DisplayMember property for ListBoxes, just as you would for ComboBoxes.

3. In the Solution Explorer list, add a new form called '**frm_studentdetails_a123456.vb**'.

4. Double-click on the 'My Project' item in the Solution Explorer list and change your **'Startup form'** to '**frm_studentdetails_a123456.vb**'.

5. Add the following objects to the form:

   - On the top of the form, add a **Label**. Change its **(Name)** property to **'lbl_title'**. Change the **'Text'** property of this Label to **'Student Details'**, and redesign the font of this label as you see fit.

   - Underneath **'lbl_title'** to the left side of the form, add a **ListBox** object. Change its **(Name)** property to **'lst_matric'**.

   - To the right of **'lst_matric'**, in the middle of the form, add three **TextBoxes**. Change the **(Name)** property of the first TextBox to **'txt_matric'**. Change the **(Name)** property of the second TextBox to **'txt_name'**. Finally, change the **(Name)** property of the third TextBox to **'txt_dept'**.

6. Open the **Form_Load** event for this form, by double-clicking on any blank space in the '**frm_studentdetails_a123456.vb**' form.

7. Add the following code to the Form_Load event:

```
Private Sub frm_studentdetails_a123456_Load(sender As System.Object, e as System.EventArgs)
Handles MyBase.Load

        Dim mysql As String = "SELECT FLD_MATRIC FROM TBL_STUDENTS_A123456"

        Dim mydatatable As New DataTable

        Dim myreader As New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        lst_matric.DataSource = mydatatable
        lst_matric.DisplayMember = "FLD_MATRIC"

End Sub
```

8. Run the Program. You will observe that when the form loads, **'lst_matric'** will be filled with the student matric numbers.

J. Displaying Data from a SELECT Statement in a TextBox Object

1. The method for displaying data from an SQL SELECT statement in a **TextBox** Object is slightly different from DataGridViews, ComboBoxes, and List Boxes, because each textbox can only display the contents of a single cell of data.

2. To display the contents of a single cell, you will need to specify the **column index** and **row index** of the SQL SELECT output. The intersection of a column index address and row index address is a single cell:

   - The **column index** can be easily specified by stating the '**attribute name'** of the SQL SELECT output as the column index address.

   - However, the **row index** cannot be easily specified because the row index address is an arbitrary integer value of the SQL SELECT output rows in the order that they are extracted.

3. To overcome the difficulty of identifying the correct row index address, we need to specify the primary key value of the data row that we want as a **WHERE** condition, so that our SQL SELECT statement will always return only one row. This way, the row index address will always be **'0'** (**zero**).

4. Below the **Form_Load** event in '**frm_studentdetails_a123456'**, create a new sub-procedure called '**refresh_text()'** that takes the currently selected 'lst_matric' ListBox value as a string input.

5. Since the matric number is a primary key for the **'tbl_students_a123456'** table, we can use this input as a WHERE condition to ensure that only one row is returned from the SQL SELECT string.

6. The code for the **'refresh_text()'** sub-procedure is as follows:

```
Private Sub refresh_text(matric As String)

        Dim mysql As String = "SELECT * FROM TBL_STUDENTS_A123456 WHERE FLD_MATRIC='" &
        matric & "'"

        Dim mydatatable As New DataTable

        Dim myreader As New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        txt_matric.Text = mydatatable.Rows(0).Item("FLD_MATRIC")
        txt_name.Text = mydatatable.Rows(0).Item("FLD_NAME")
        txt_dept.Text = mydatatable.Rows(0).Item("FLD_DEPT")

End Sub
```

NOTE: Since the data for all three TextBoxes need to come for the same row, we can use the same SQL SELECT output to fill in each TextBox.

7. Add the **'refresh_text()'** function call to the end on the **Form_Load** event, and use the 'lst_matric' value as input, as follows:

```
Private Sub frm_studentdetails_a123456_Load(sender As System.Object, e as System.EventArgs)
Handles MyBase.Load

        Dim mysql As String = "SELECT FLD_MATRIC FROM TBL_STUDENTS_A123456"

        Dim mydatatable As New DataTable

        Dim myreader As New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        lst_matric.DataSource = mydatatable
        lst_matric.DisplayMember = "FLD_MATRIC"

        refresh_text(lst_matric.Text)

End Sub
```

8. Double-click on 'lst_matric'. When the code snippet for its default event 'SelectedIndexChanged' is generated, change the event at the top of the main panel (with the lightning symbol) to 'MouseClick'. This will generate the code snippet for the 'MouseClick' event.

9. Add the same 'refresh_text()' function call to the 'MouseClick' code snippet as follows:

```
Private Sub lst_matric_MouseClick(sender As Object, e As
System.Windows.Forms.MouseEventArgs) Handles lst_matric.MouseClick

        refresh_text(lst_matric.Text)

End Sub
```

   NOTE: We use the '**MouseClick**' event instead of the '**SelectedIndexedChanged**' event because VB automatically detects a '**SelectedIndexedChanged**' on all ListBoxes immediately upon initialization, **before** the actual **Form_Load** event. Since the **'lst_matric'** ListBox will still be empty at this point, calling the **'refresh_text()'** sub-procedure in the '**SelectedIndexedChanged**' event will produce an error because the 'lst_matric' has not been filled with valid matric numbers. The **'MouseClick'** event is a safer event because it requires an active 'click' by the user before the sub-procedure is called.

10. Delete the code snippet for the 'SelectedIndexChanged' event from its 'Private Sub...' to its 'End Sub'.

11. Run the program and click on the matric numbers in the list boxes. The text all three TextBoxes will change accordingly.

K. Displaying Pictures to Match Displayed Database Records

1. In your **'Bin/Debug'** folder, create a subfolder called 'pictures'

2. In your web browser, open http://images.google.com and Google "passport photo". Save the pictures of enough people to match the number records you have in your 'tbl_students_a123456' table, and place it in the **'pictures'** subfolder.

3. Rename each individual photo to match exactly one matric number in your table. For example, if you have a record with the matric '**A101'** rename one photo to become **'A101.jpg'**. Repeat this process until you have one photo for each matric number in your table.

4. On the right side of your **'frm_studentdetails_a123456.vb'** form, to the right of the TextBoxes, add a PictureBox object from your ToolBox. Change its (Name) property to **'pic_student'**. Change its **'BackgroundImageLayout'** property to **'Zoom'**.

5. Add the following line of code at the end of your '**refresh_text()**' sub-procedure.

```
Private Sub refresh_text(matric As String)

        Dim mysql As String = "SELECT * FROM TBL_STUDENTS_A123456 WHERE FLD_MATRIC='" &
        matric & "'"

        Dim mydatatable As New DataTable

        Dim myreader As New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        txt_matric.Text = mydatatable.Rows(0).Item("FLD_MATRIC")
        txt_name.Text = mydatatable.Rows(0).Item("FLD_NAME")
        txt_dept.Text = mydatatable.Rows(0).Item("FLD_DEPT")

        pic_student.BackgroundImage = Image.FromFile("pictures/" & txt_matric.Text & ".jpg")

End Sub
```

6. Run the program. You will observe that the picture displayed will change according to the matric that you clicked in the '**lst_matric'** ListBox.

PART 3: Searching Through Data

A. Using SQL SELECT to Perform Partial String Searches

1. Previously we have used WHERE conditions in SQL SELECT statements of match exact records. We can also perform partial matching for attributes with the 'String' data type, by using the LIKE statement. To do this we will need to include SQL wildcards as part of the dynamic search string.

2. In the Solution Explorer list, add a new form called '**frm_searchcourses_a123456.vb**'.

3. Double-click on the 'My Project' item in the Solution Explorer list and change your **'Startup form'** to '**frm_searchcourses_a123456.vb**'.

4. Add a **Label** to the top of the '**frm_searchcourses_a123456.vb**' form. Change the **(Name)** property of this Label to '**lbl_title'**. Change the 'Text' property of this Label to **'Search for a Course'**, and redesign the font of this label as you see fit.

5. Underneath this Label, add a TextBox. Change the **(Name)** property of this TextBox to '**txt_search'**.

6. Underneath the TextBox, add a **DataGridView** object. Change the **(Name)** property of this DataGridView object to '**grd_courses'**.

7. Double-click on any blank area on the 'frm_searchstring_a123456' form to get the Form_Load event.

8. Underneath the Form_Load event, after the 'End Sub', create a new sub-procedure called '**refresh_grid()**' that does not take any input. Add the following code in the sub-procedure:

```
Private Sub refresh_grid()

        Dim mysql As String = "SELECT * FROM TBL_COURSES_A123456 WHERE
        FLD_COURSE_NAME like '%" & txt_search.Text & "%'"

        Dim mydatatable As New DataTable

        Dim myreader As New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        grd_courses.DataSource = mydatatable

End Sub
```

NOTE: The wild card for SQL string conditions in VB .NET are as follows:

- **% (percent sign)** → The wildcard character to indicate any number of characters or none at all.
- **_ (underscore)** → The wildcard character to indicate exactly one character.

9. Double-click on the **'txt_search'** TextBox to generate the code snippet for its default event '**TextChanged**'. Add the function call for '**refresh_grid()**' to this event:

```
Private Sub txt_search_TextChanged(sender As System.Object, e As System.EventArgs) Handles
txt_search.TextChanged

        refresh_grid()

End Sub
```

10. Double-click on any blank area on the 'frm_searchstring_a123456' form. Add the function call for 'refresh_grid()' to the Form_Load event:

```
Private Sub frm_searchcourses_a123456_Load(sender As System.Object, e As System.EventArgs)
Handles MyBase.Load

        refresh_grid()

End Sub
```

11. Run the program. As you type in the TextBox, you will observe that the data in the **'grd_courses'** changes as well.

B. Using SQL SELECT for Data Validation

1. Apart from searching through data, SQL SELECT statements can also be used to validate data. For example, we can create a 'Login' form to check for valid usernames and passwords before users can be authorized to use your system.

2. First we will need to add a table called **'tbl_users_a123456'** in your Microsoft Access database. This table should have the following data:

FLD_USERNAME	FLD_PASSWORD
admin	abc123

TIP: In the example above, we store the password as plaintext to simplify the login process. In real-world password systems, the plaintext password is never kept in the database. Instead a one-way salted-hash value for the password is generated during user registration, using hashing algorithms such as MD5 or SHA. Only this hash value is kept in the database. Later, during login, whatever password that is entered goes through the same hashing algorithm, and only the hash results will be compared. This method prevents theft of the user passwords from users that have direct access to the database tables.

3. Return to Visual Studio. In the Solution Explorer list, add a new form called '**frm_login_a123456.vb**'.

4. Double-click on the 'My Project' item in the Solution Explorer list and change your **'Startup form'** to '**frm_login_a123456.vb**'.

5. Add two Labels, two TextBoxes, and one Button to '**frm_login_a123456.vb**'. Change the properties as follows, and arrange on the form accordingly:

   - First Label: **(Name) = 'lbl_username', Text = 'Username'**

   - Second Label: **(Name) = 'lbl_password', Text = 'Password'**

- First TextBox: **(Name) = 'txt_username'**
- Second TextBox: **(Name) = 'txt_password', PasswordChar = *(asterisk)**
- Button: **(Name) = 'btn_login', Text = 'LOGIN'**

6. Double-click on '**btn_login'** to generate the code snippet for its **Button_Click** event, and type in the following code:

```
Private Sub btn_login_Click(sender As System.Object, e As System.EventArgs) Handles
btn_login.Click

        Dim mysql As String = "SELECT COUNT(*) AS NUM_MATCHES FROM TBL_USERS_A123456
        WHERE FLD_USERNAME='" & txt_username.Text & "' and FLD_PASSWORD='" &
        txt_password.Text & "'"

        Dim mydatatable As New DataTable

        Dim myreader As New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        Dim num_matches As String = mydatatable.Rows(0).Item("NUM_MATCHES")

        If num_matches = 1 Then

                frm_mainmenu_a123456.Show()
                Me.Hide()

        Else

                txt_username.Text = ""
                txt_password.Text = ""
                MsgBox("Incorrect Username or Password")

        End If

End Sub
```

7. Run the Program. If you type in the username 'admin' and password 'abc123', your main menu form will open. If you type in anything other than this, the form will reset.

IMPORTANT: These notes cover Dynamic SQL to familiarize students with the fundamental concept of customizing SQL commands to meet the current user needs. However, the Dynamic SQL method shown in these notes are highly susceptible to "SQL Injection Attacks". Furthermore, sensitive data such as passwords in the example above is kept in plaintext – making it subject to data theft through "SQL Injection Attacks". These attacks can be prevented by:

- Minimizing direct TextBox input, and instead using ComboBoxes, ListBoxes or other restrictive controls. Or...
- '**Sanitizing'** all TextBox input fields by stripping any and all special characters from user entered input. Or...
- Parameterizing SQL inputs

Read up on "SQL Injection Attacks" to know more of the risks and methods of prevention. Protection from SQL Injection Attacks and protection of sensitive data will be covered in detail in future Lab Notes.

PART 4: Before Leaving the Lab

1. Copy your entire working folder, e.g **'c:\a123456_facultyrecords'**, to your USB drive. This folder will be used again for the next lab session.

2. Delete the existing **'c:\a123456_facultyrecords'** folder in your lab computer's **'C:\'** drive.

Additional Lab Exercises

1. Create a form that can filter courses based on its number of credit hours.

2. Create a form that can filter students based on 2 attribute values, which are:

   i. the student's department, and
   ii. a partial string search of the student's name.

3. Create a form that can list the name and matric number of students that have registered for a course during a specific semester and session.

4. Create a form that lists all the courses registered by a specific student during a specific semester and session, and also display the total credit hours registered for that semester & session WITHOUT using the 'CALCULATED' data type in Microsoft Access.