IT1753 Principles of Computing PCMP

# Topic 6A

Storing Data
Using SQL Database

## Topics

- Create database in Visual Studio
- Working with database in C# application

**Objectives:**

- Be able to work with database

# Working with Database
using SQL database

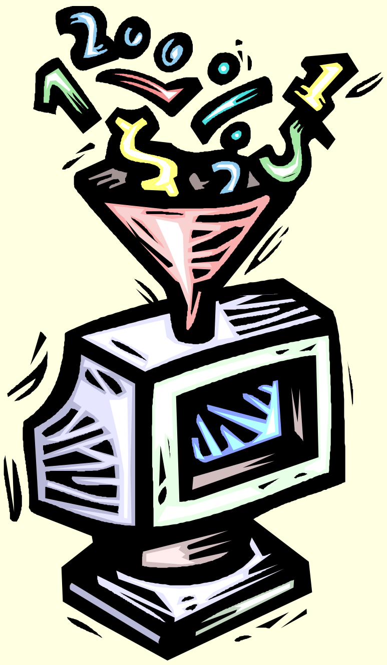
- Why do we need a database?
  - Recap the concept you have learnt in Database Management System module
  - Need a database to store our data
    - Helps to organise large amount of data
    - Stores it for long term usage (unlike class variable, information is gone when the application ends)

## Working with Database
using SQL database

- How to work with database in C# using Visual Studio?

**In 3 simple steps:**

1. Create a new SQL database
2. Connect database objects with a data source
3. Create controls to bind the table



## Example1: Application to store customer data in SQL database

- Create a simple database to store Customer data.
- Every customer has:
  - Customer ID (unique primary key)
  - Name
  - Birthday
  - Telephone number

| Customer ID | Name | Birthday | Tele Phone |
|---|---|---|---|
| 1 | Mary | 1 jan 1970 | 1234567 |
| 2 | Sally | 9 jan 1970 | 5534567 |

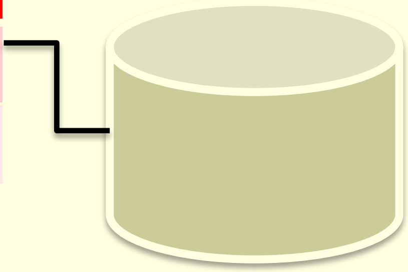
- Navigate the customer data using a window form.

## About SQL database

- It stores data for you in an organised and interrelated way
- Visual Studio provides tools to help you to maintain your data in database
- Data in a SQL database are stored in **tables**
- A database can have 1 or many tables.

| Customer ID | Name | Birthday | Tele Phone |
|---|---|---|---|
| 1 | Mary | 1 jan 1970 | 1234567 |
| 2 | Sally | 9 jan 1970 | 5534567 |

Customer table



## Accessing Records in Database

### ❑ Creating Client Data Applications

Most applications revolve around:

- ❑ To read and update information in databases.
- ❑ To allow data integration in applications

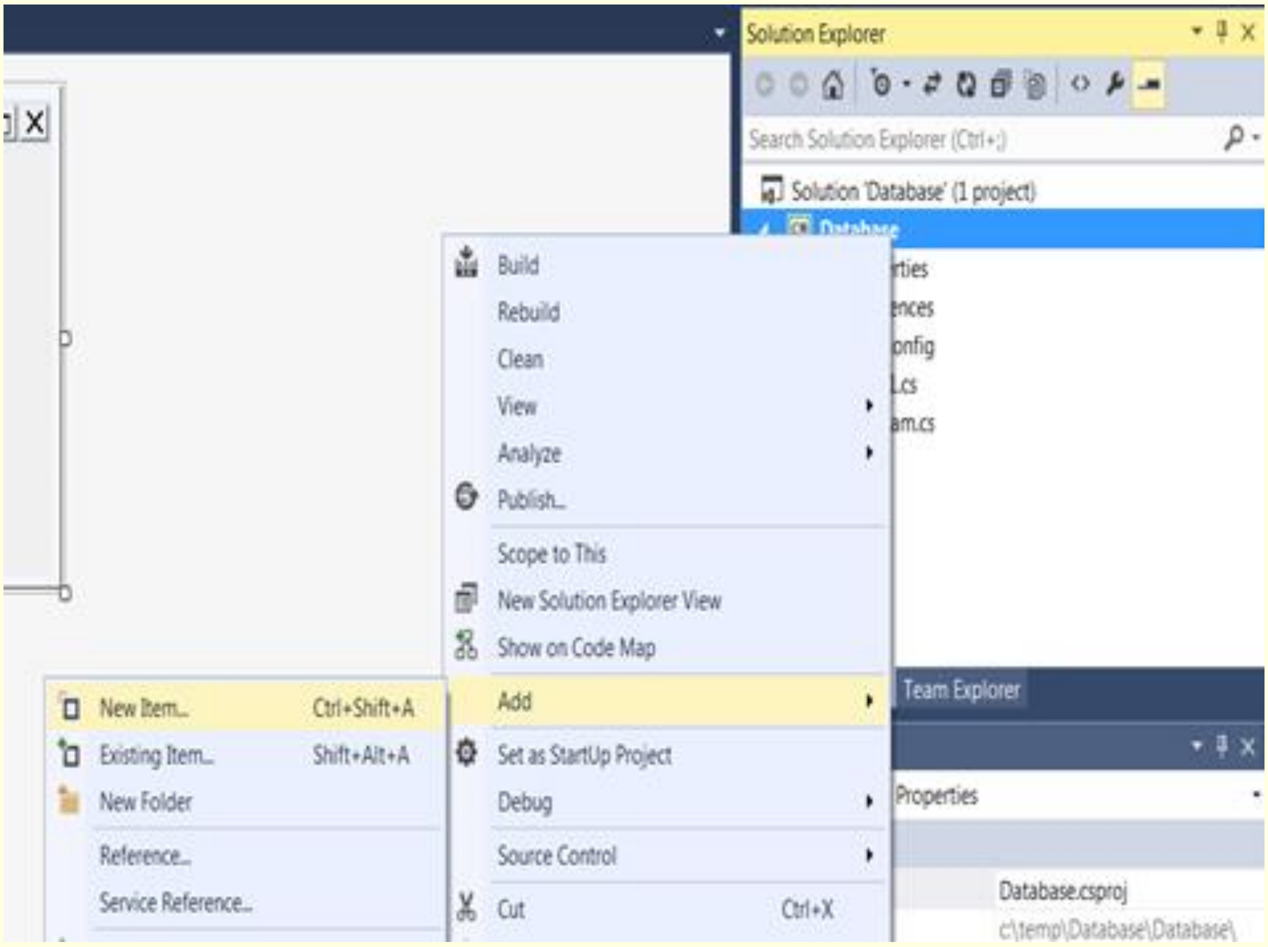
Visual Studio provides tools and support for working with data using **ADO.NET** and Windows Forms Data Binding.

What is **ADO.NET**?

- ❑ ADO stands for ActiveX Data Objects
- ❑ It provides consistent access to data sources such as SQL Server and XML, and to data sources exposed through OLE DB and ODBC.
- ❑ Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, handle, and update the data that they contain.

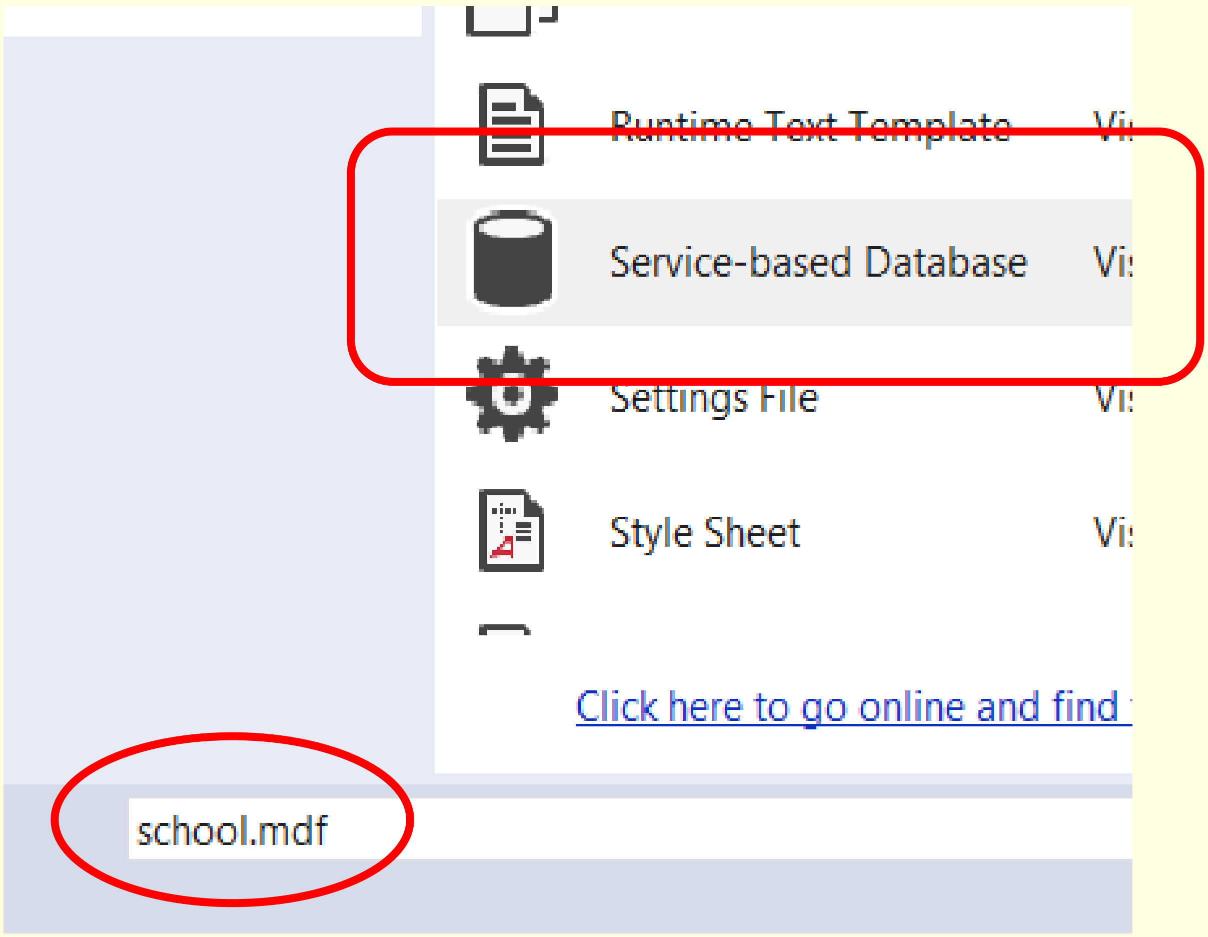
## Example1- Step1: Create a new SQL database

1. Create a new project in C:\Temp and name it as DATABASE.
2. In Solution Explorer, right-click the project. Select Add, then choose New Item.



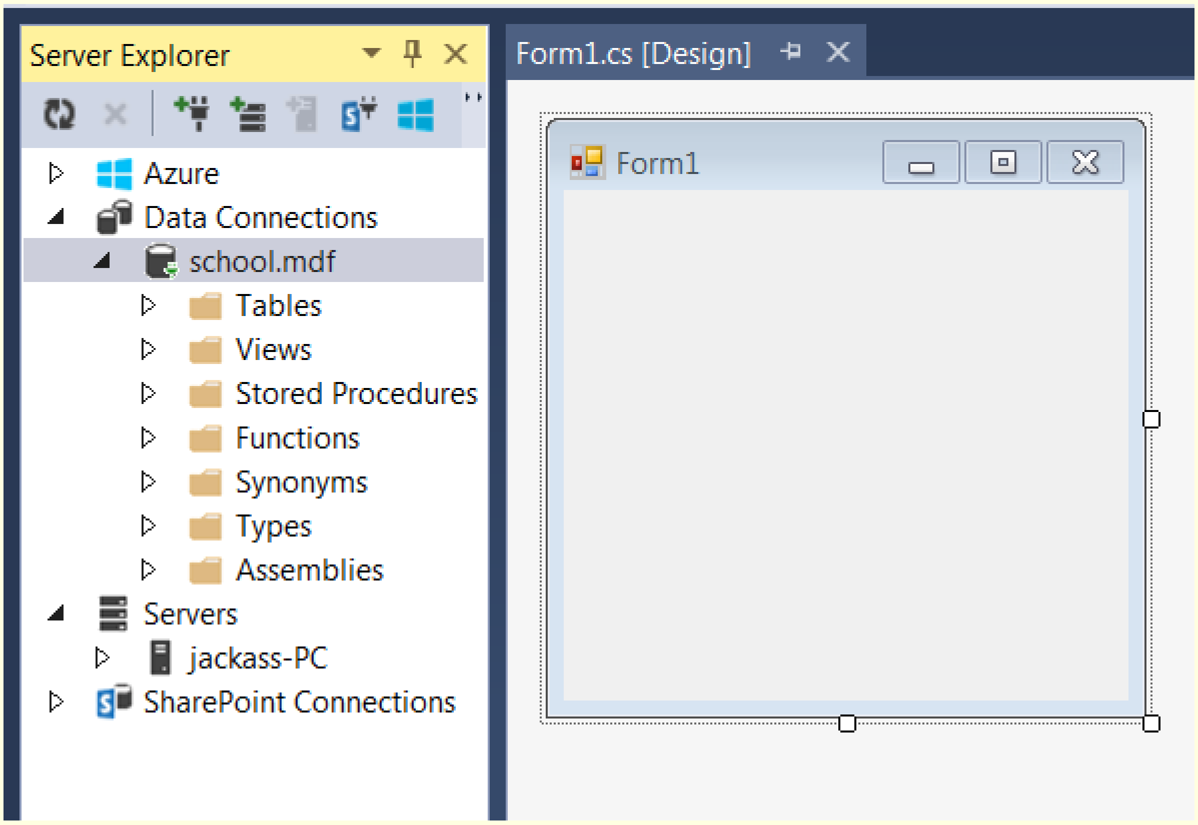
# Example1- Step1: Create a new SQL database

1. Choose the Service-based Database icon and name it as school.mdf
2. Click on the Add button



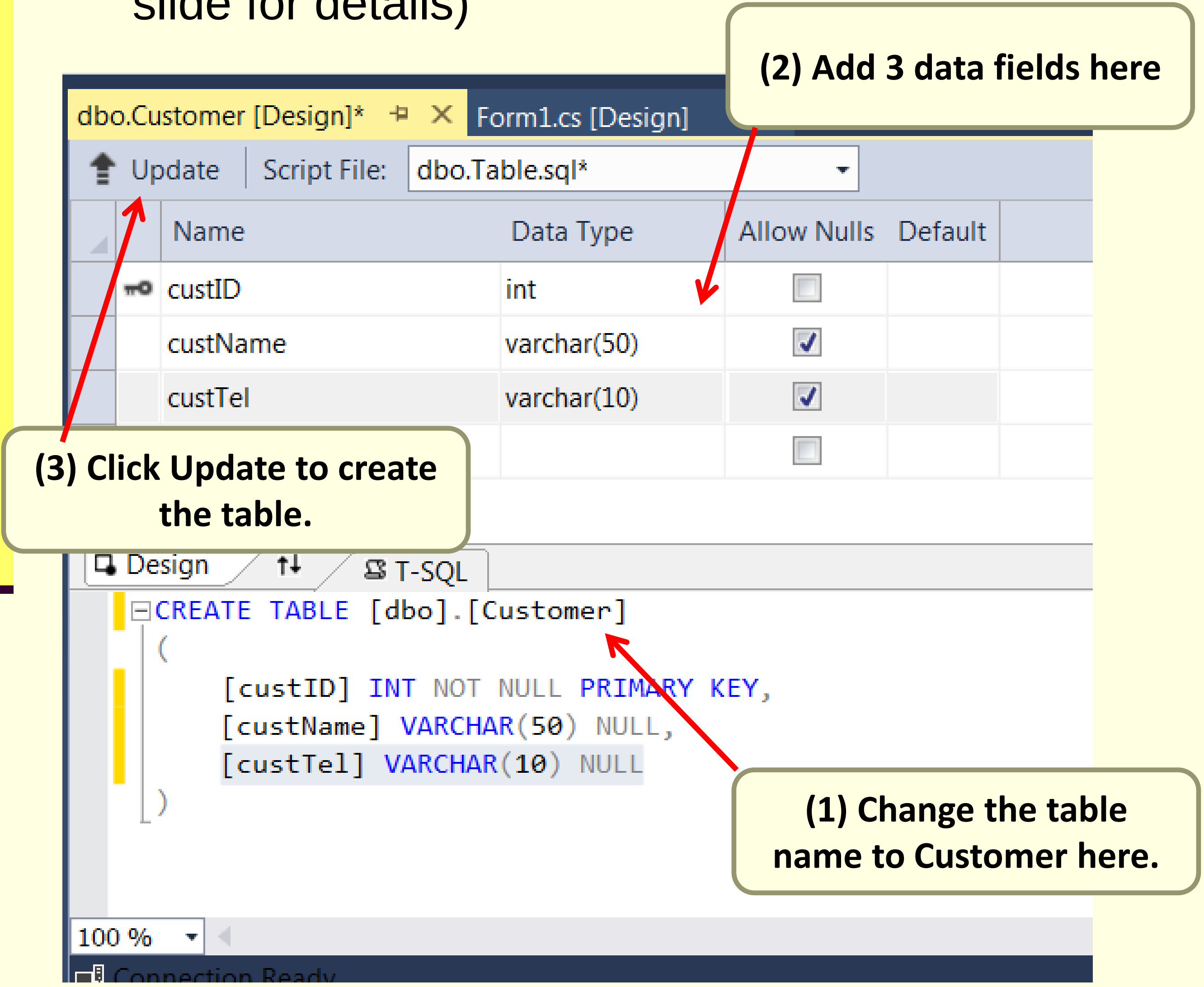
# Example1- Step1: Create a new SQL database

4. Open the Server Explorer (View → Server Explorer)
5. Expand Data Connections if necessary.
6. You may see a red cross at the school.mdf. Double click on it and the red cross should disappear.



# Example1: Create Table in SQL database

1. Add a table to the database. Right click on Tables and select Add New Table. In the CREATE TABLE window change the table name to Customer.
2. Define the column (fields) of the table (see next slide for details)


dbo.Customer [Design]*
Form1.cs [Design]
Update
Script File:
dbo.Table.sql*
Design
T-SQL
100 %
Connection Ready
(2) Add 3 data fields here
(3) Click Update to create the table.
(1) Change the table name to Customer here.

| Name | Data Type | Allow Nulls | Default |
|---|---|---|---|
| custID | int | ☐ | |
| custName | varchar(50) | ✓ | |
| custTel | varchar(10) | ✓ | |
| | | ☐ | |

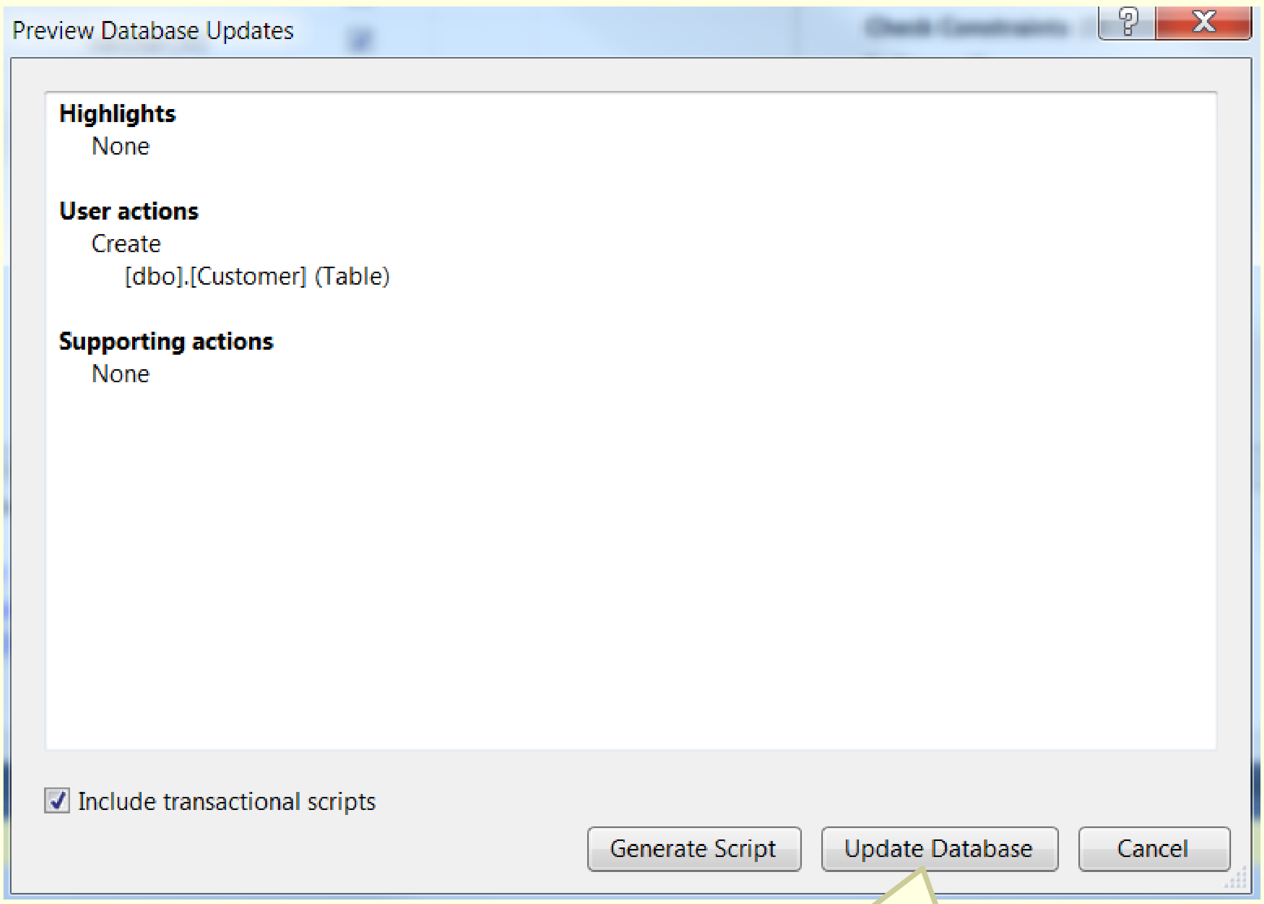
```
CREATE TABLE [dbo].[Customer]
(
    [custID] INT NOT NULL PRIMARY KEY,
    [custName] VARCHAR(50) NULL,
    [custTel] VARCHAR(10) NULL
)
```

# Create Table in SQL database: Data Field Definitions

| | |
|---|---|
| custID field | – Data Type → int<br>- Allow Nulls → no tick<br>- Set as Primary Key |
| custName field | - Data Type = varchar(50)<br>- Allow Nulls → ticked |
| custTel field | - Data Type = varchar(10)<br>- Allow Nulls → ticked |

- int → integer (ie. whole number).
- varchar(50) → Variable Character type can hold letters and numbers. 50 means it can hold max 50 characters.
- Allow Nulls → if ticked (ie. selected) means the data field can be empty. If not ticked, then the data field must have a value. So in our table, when you create a Customer record, you must provide a custID but the Name and Tel can be blank (ie. empty)
- Primary Key → use as Index; must be unique.

# Example1: Create Table in SQL database


Preview Database Updates

**Highlights**
None

**User actions**
Create
[dbo].[Customer] (Table)

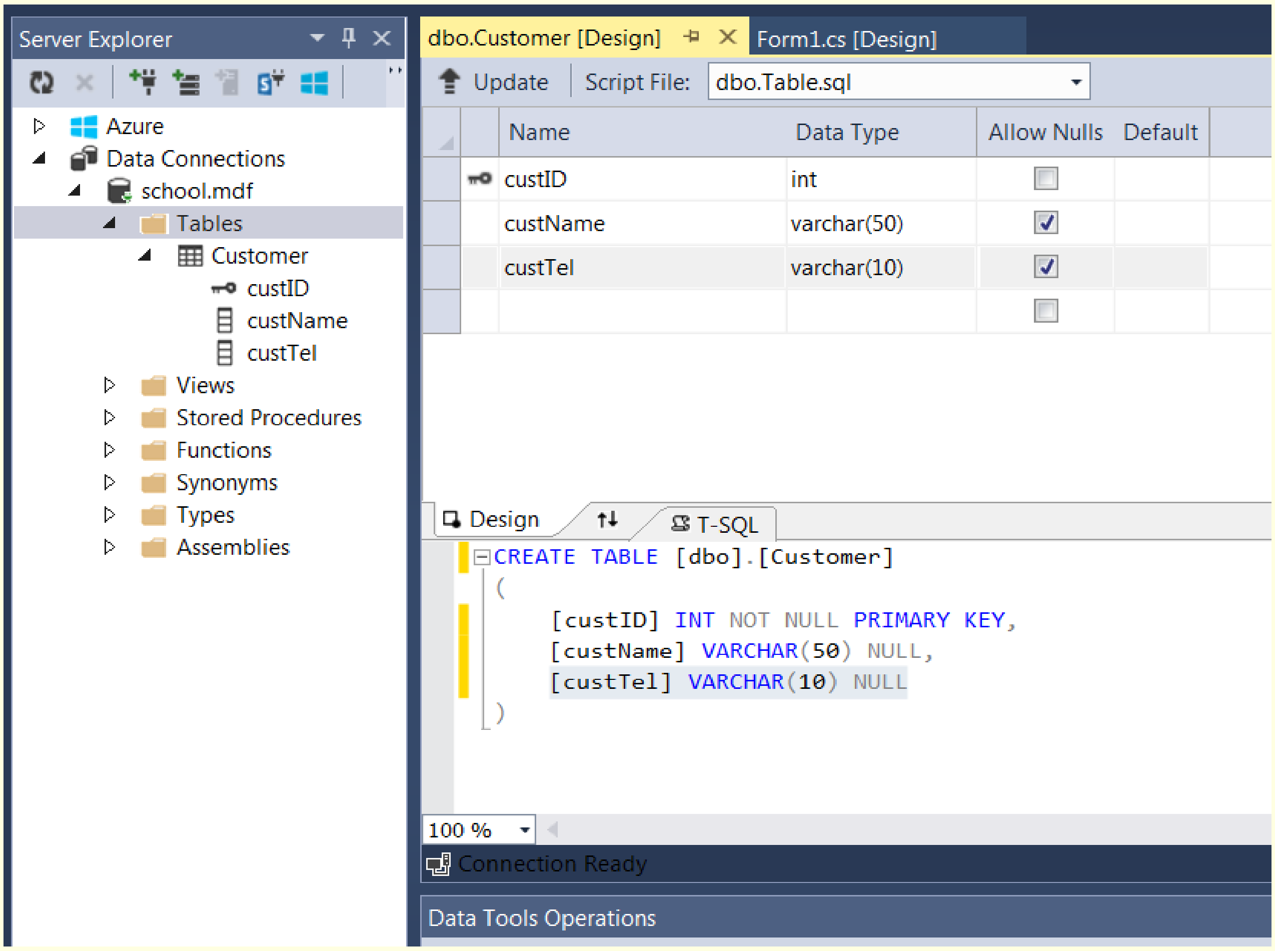
**Supporting actions**
None

☑ Include transactional scripts

**(4) Click Update Database**

# Example1:
# Create table in SQL database

Completed table shown below.



| Name | Data Type | Allow Nulls | Default |
|---|---|---|---|
| custID | int | ☐ | |
| custName | varchar(50) | ✓ | |
| custTel | varchar(10) | ✓ | |
| | | ☐ | |

```
CREATE TABLE [dbo].[Customer]
(
    [custID] INT NOT NULL PRIMARY KEY,
    [custName] VARCHAR(50) NULL,
    [custTel] VARCHAR(10) NULL
)
```

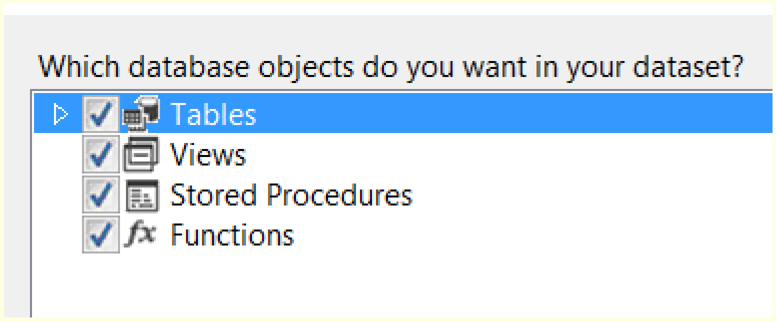
Please make sure you have the screen above.
Call your tutor if your screen looks different.

## Example1- Step2: Add database driven control

1. Select the Windows Form Form1.
2. Select View → Other Windows → Data Sources
3. Select Add New Data Source.

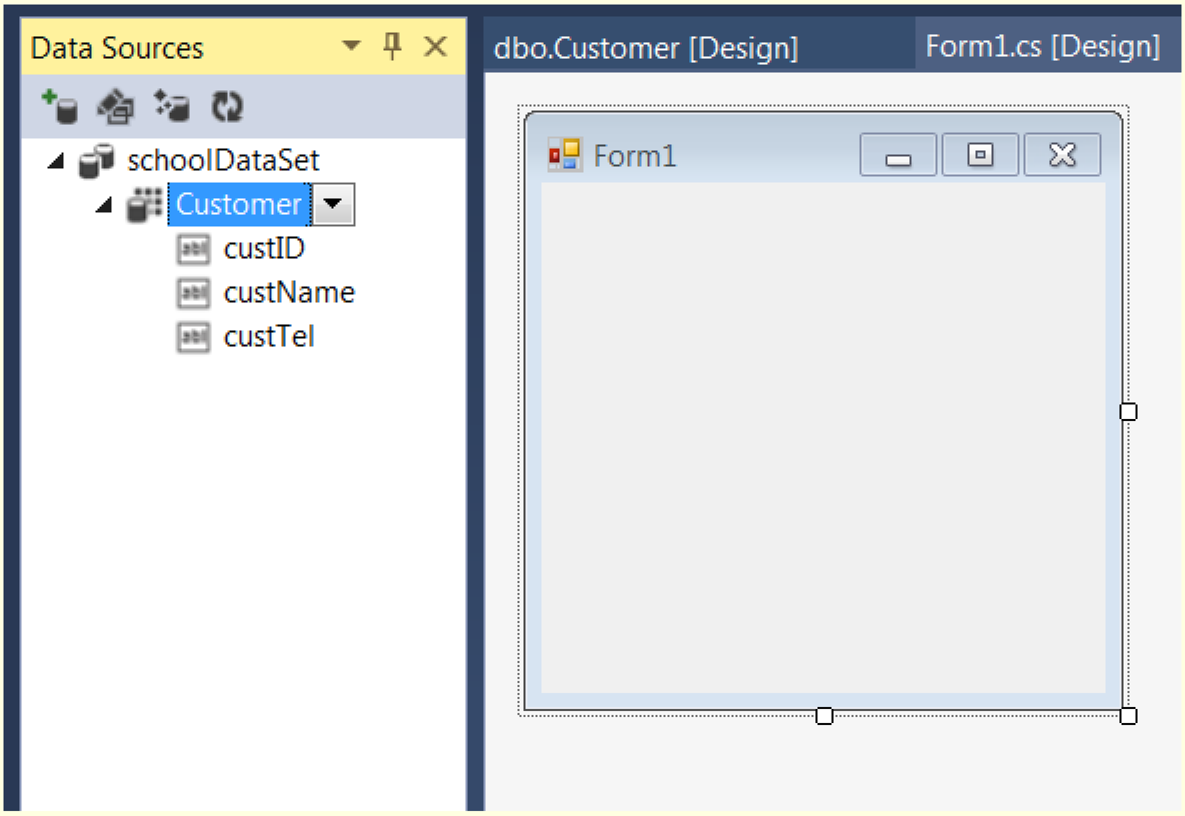
(if you don't see it, right-click schoolDataSet)

4. Select Database → Next
5. Select Dataset → Next
6. Select school.mdf → Next
7. Click Next to accept the default schoolConnectionString
8. Select everything (see picture below) in the [Choose Your Database Objects] window → Finish.


Which database objects do you want in your dataset?

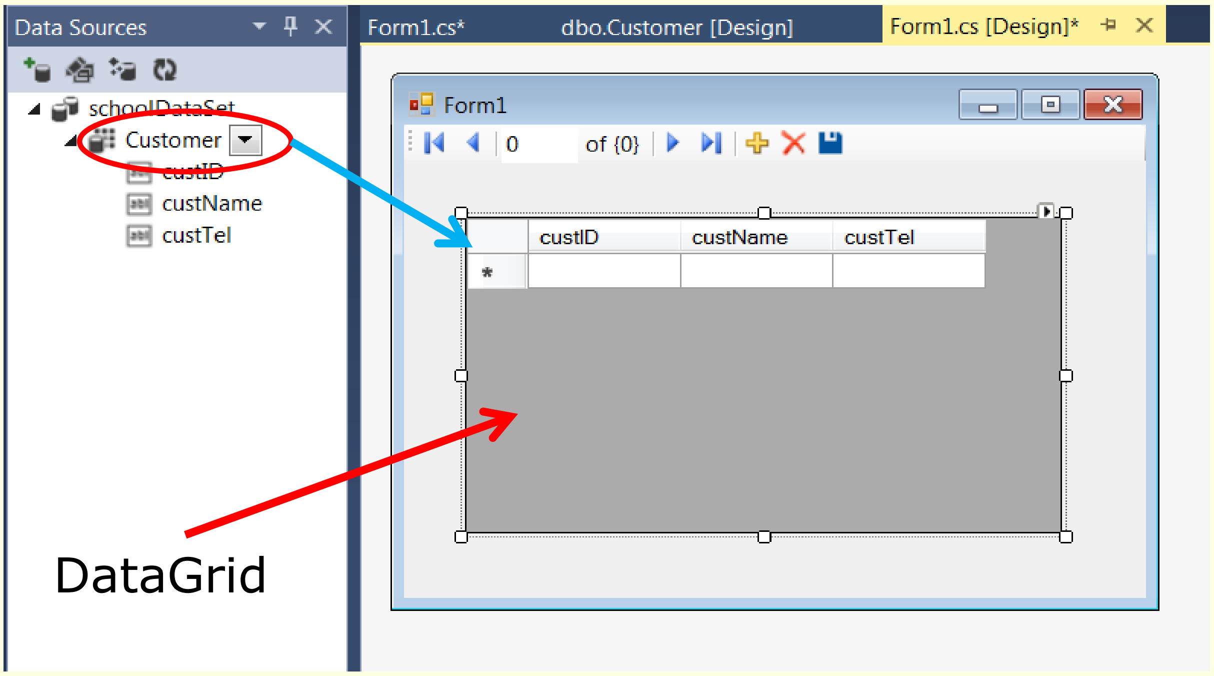
- ✓ Tables
- ✓ Views
- ✓ Stored Procedures
- ✓ Functions

# Example1- Step2: Add database driven control



- You should see the above screen.
- Call your tutor if you do not see it.

# Example1- Step3: Create controls to bind the table

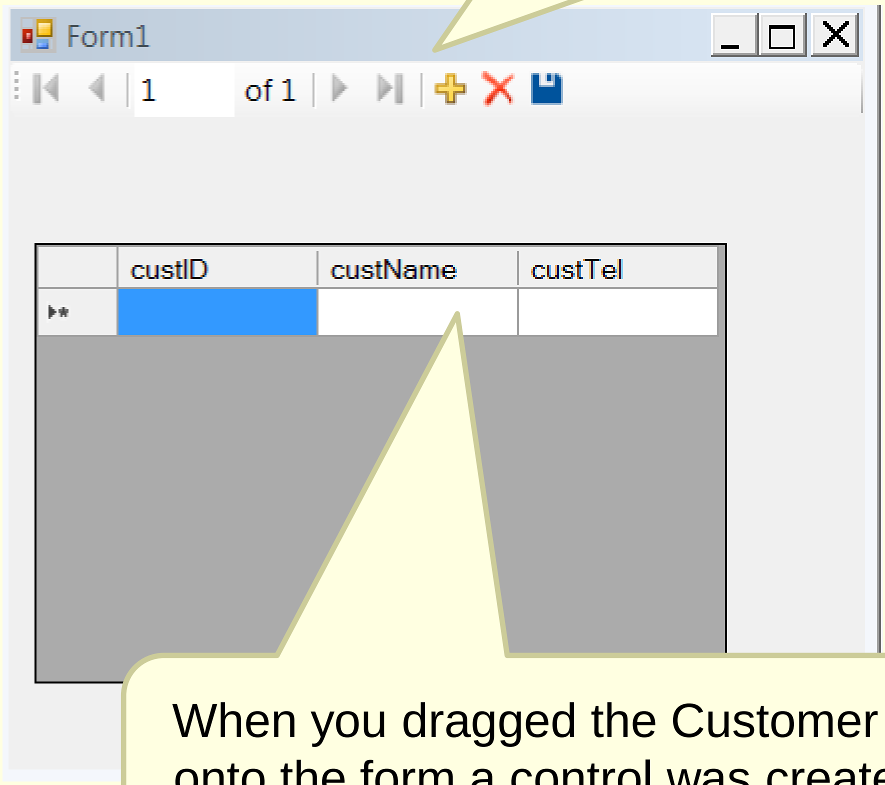


- Click on the Customer object and drag it into the Form1.
- You should get a table as shown above.

# Example1- Step3: Create controls to bind the table

3. Press Ctrl F5 and run the application.

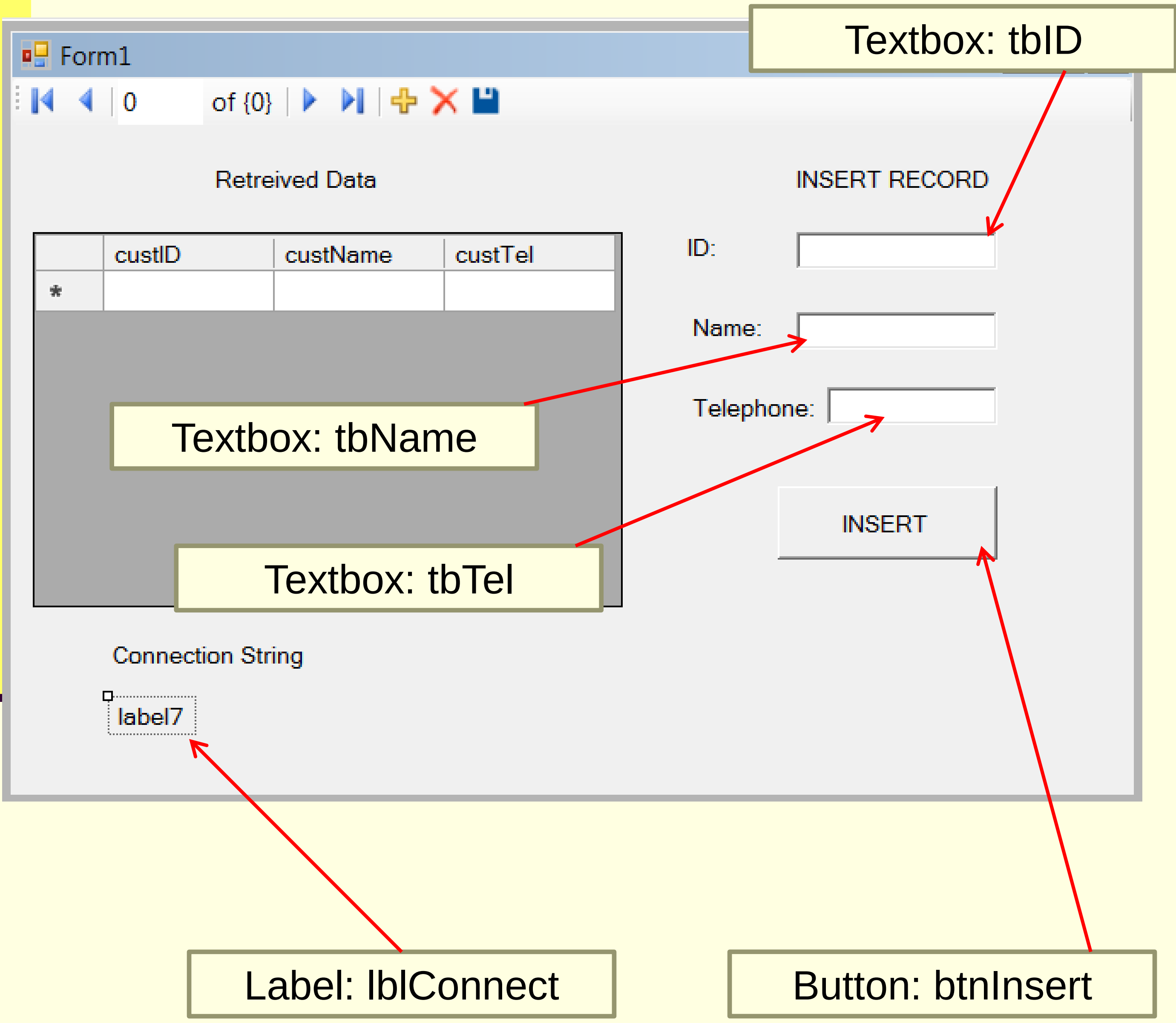
VS creates this toolbar for navigating through the Customer table



When you dragged the Customer table onto the form a control was created for each column in the table

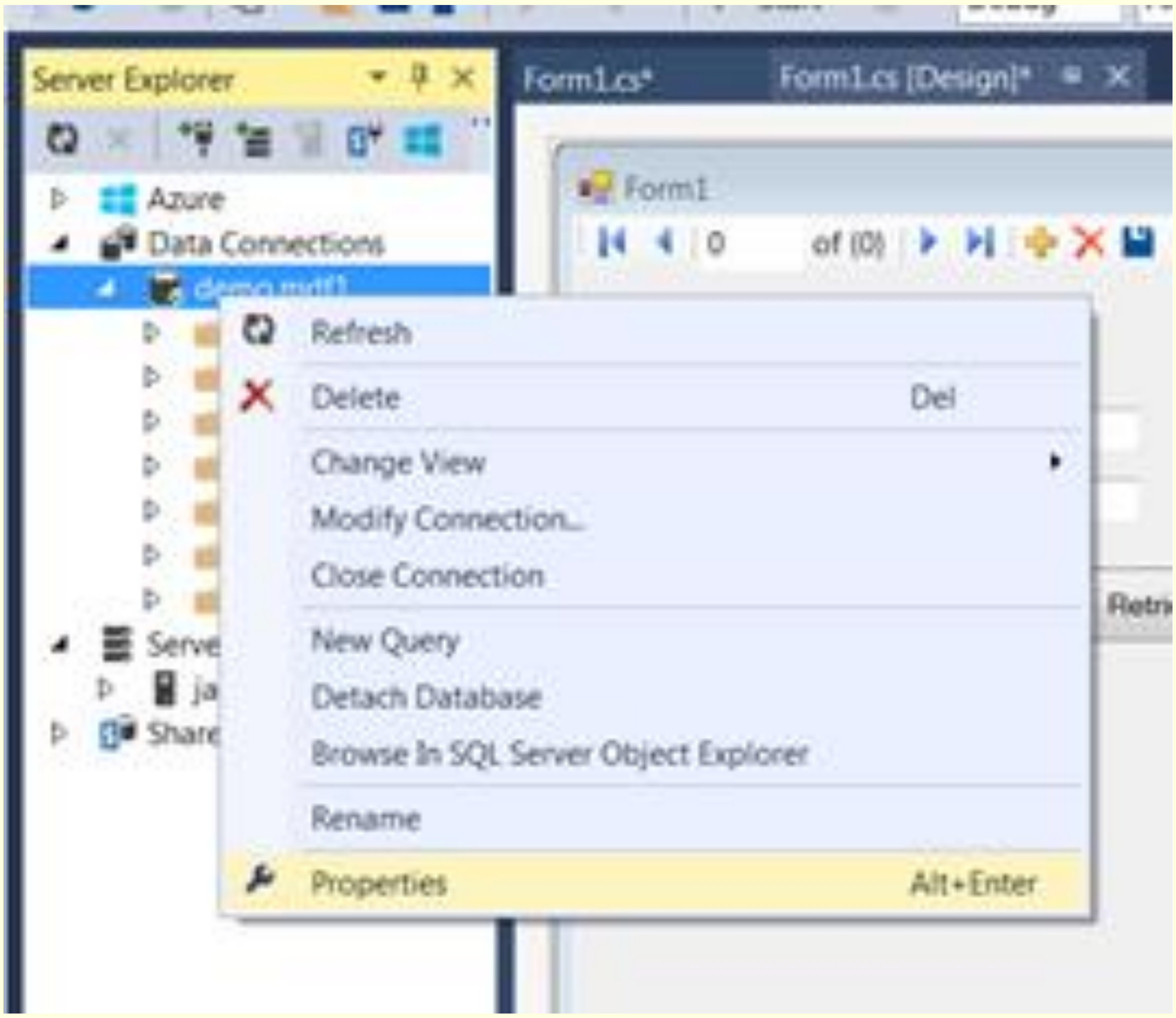
# Example1- Step4: Create the Form to Insert and Retrieve data

2. Amend the form as follows:



## Example1- Step 4: Connection String

1. Open the Server Explorer.
2. Right click on school.mdf and select Properties.

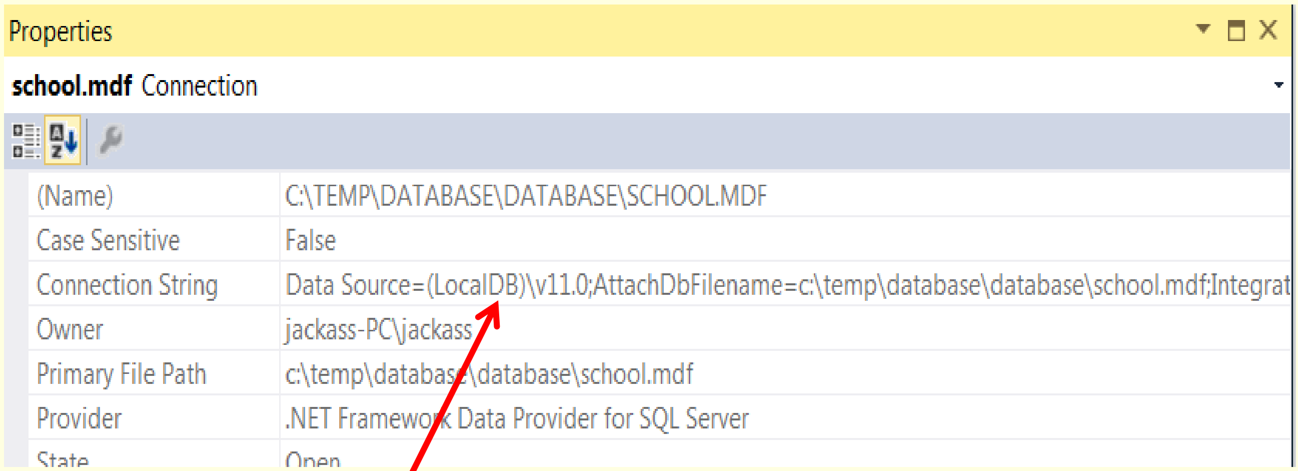


## Connection String

1. Copy the entire Connection String text:

Data Source……………………..Integrated……

You will paste it in the codes later.


Properties

school.mdf Connection

| | |
|---|---|
| (Name) | C:\TEMP\DATABASE\DATABASE\SCHOOL.MDF |
| Case Sensitive | False |
| Connection String | Data Source=(LocalDB)\v11.0;AttachDbFilename=c:\temp\database\database\school.mdf;Integrat |
| Owner | jackass-PC\jackass |
| Primary File Path | c:\temp\database\database\school.mdf |
| Provider | .NET Framework Data Provider for SQL Server |
| State | Open |

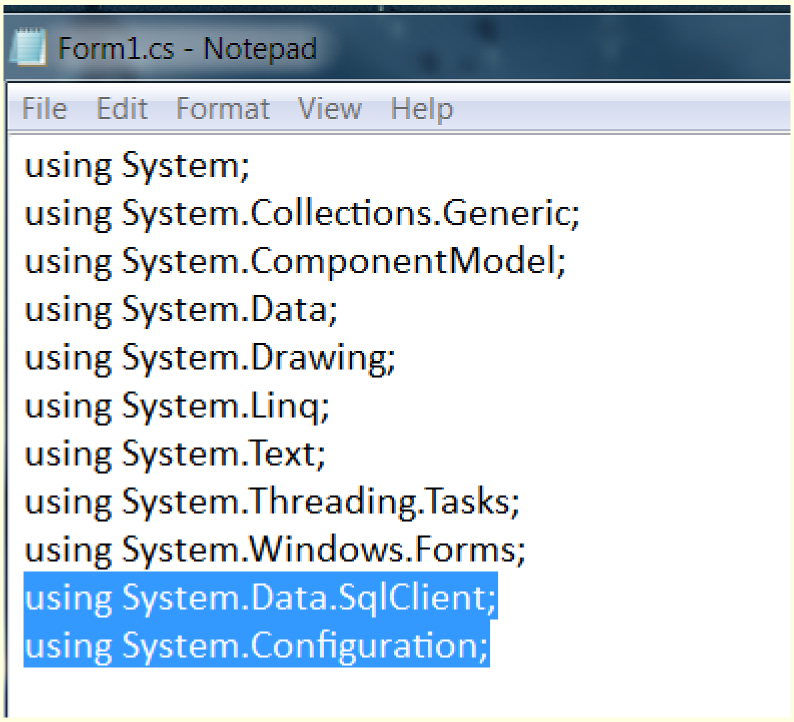
Copy this ENTIRE string

The connection string specifies the location of the database.

## Insert Record - Additional Libraries Needed

- Double click the INSERT button on the form.
- You need to use some additional System files.
- Go to Form1.cs and scroll to the top.
- Add below 2 lines:

```
using System.Data.SqlClient;
using System.Configuration;
```



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Configuration;
```

## Explanation:

using System.Data.SqlClient;

Add the SqlClient namespace which interacts with the SQL Server and allows the development of data-driven applications.

- It creates database connections with SqlConnection.
- It inserts data with SqlCommand.
- It handles rows with SqlDataReader.

using System.Configuration;

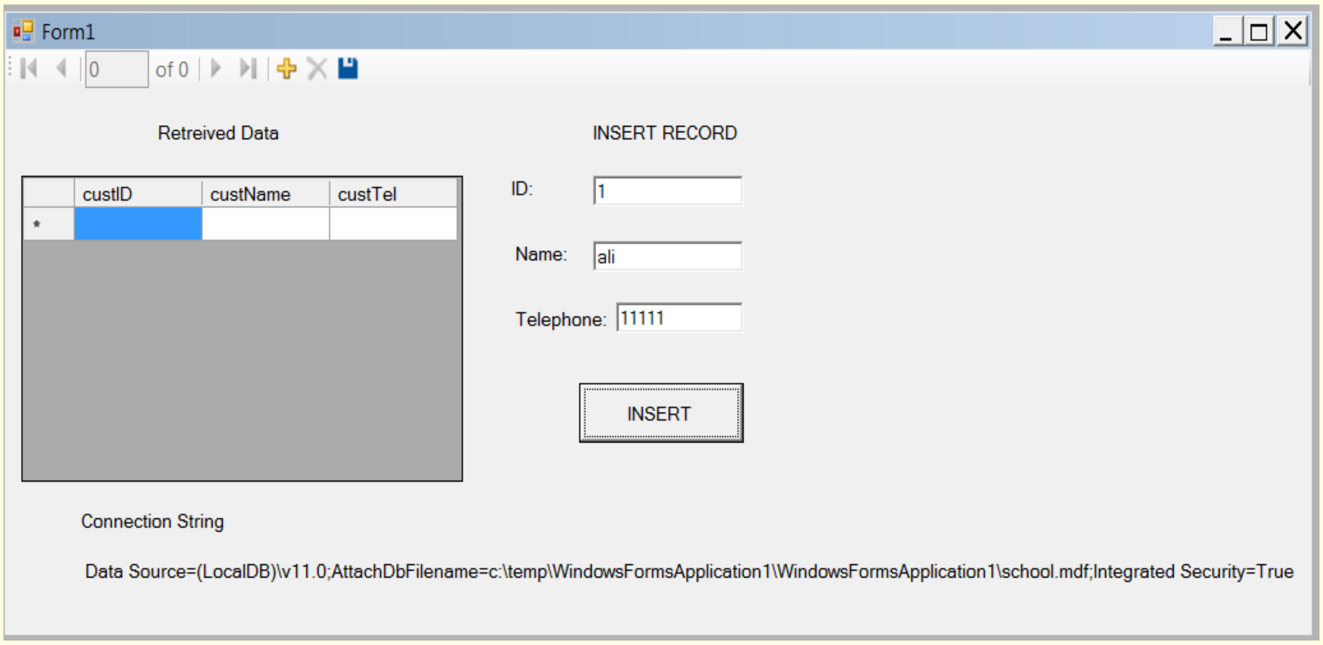
Access the ConfigurationManager class which provides methods to access configuration information.

# Example 1: Application to Insert Record to Database

```
private void btnInsert_Click(object sender, EventArgs e)
{
            int varID;          //create variables first
            string varName;
            string varTel;

            // read in name and telephone fields
            varID = int.Parse(tbID.Text);
            varName = tbName.Text;
            varTel = tbTel.Text;

      // paste the connection string you copied earlier to below
      // add another '\' to all the back-slashes
            string connect = "Data
Source=(LocalDB)\\v11.0;AttachDbFilename=c:\\temp\\d
atabase\\database\\school.mdf;Integrated
Security=True";
            // Display the connection string to check
            lblConnect.Text = connect;

      // → 1
 }
```

*Run the program to make sure your Connection String is working before continuing*

## Insert Record - Testing the Connection String



Retreived Data

INSERT RECORD

| | |
|---|---|
| ID: | 1 |
| Name: | ali |
| Telephone: | 11111 |

Connection String

```
Data Source=(LocalDB)\v11.0;AttachDbFilename=c:\temp\WindowsFormsApplication1\WindowsFormsApplication1\school.mdf;Integrated Security=True
```

- ❑ Enter some data and click Insert button.
- ❑ You should see the Connection String as shown above.
- ❑ Note the example above has the location in C:\temp, the Project name is database and the database is called school.mdf
- ❑ Your connection string may be different.

## Insert Record – Continue After Connection String is Working

**Add the codes below at 1 in slide 24.**

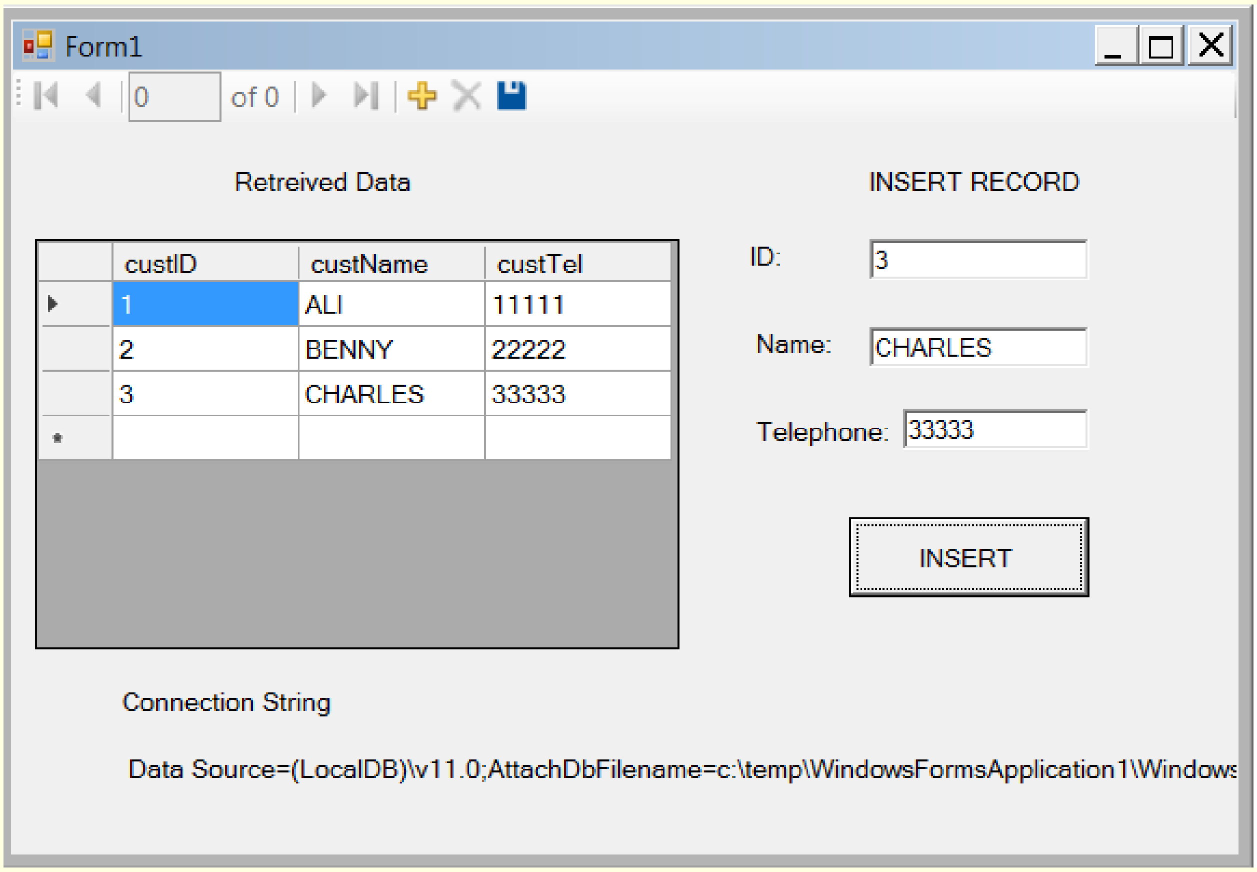
```
//creates a SQL connection object and opens
//the connection
    SqlConnection myConnect = new
            SqlConnection(connect);
    myConnect.Open();

//creates a SQL command object
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = myConnect;

//forms the SQL command and parameters
cmd.CommandText = "INSERT INTO
Customer(custID, custName, custTel) values (@xid,
@xname, @xtel)";

cmd.Parameters.Add(new SqlParameter("@xid",
varID));

cmd.Parameters.Add(new SqlParameter("@xname",
varName));


//continue next slide
```

## Insert Record …. continue

```
cmd.Parameters.Add(new SqlParameter("@xtel",
varTel));

//runs the SQL command
cmd.ExecuteNonQuery();

//Refresh the DataGrid
DataTable dt = new DataTable();

SqlDataAdapter orange = new
SqlDataAdapter("Select * from Customer",
myConnect);

orange.Fill(dt);
customerDataGridView.DataSource = dt;
customerDataGridView.Update();
customerDataGridView.Refresh();

//close database connection
myConnect.Close();

} // end of Insert button
```

*The entire program is in Annex A*

# Test Your Application



Retreived Data

| | custID | custName | custTel |
|---|---|---|---|
| ▶ | 1 | ALI | 11111 |
| | 2 | BENNY | 22222 |
| | 3 | CHARLES | 33333 |
| * | | | |

INSERT RECORD

| | |
|---|---|
| ID: | 3 |
| Name: | CHARLES |
| Telephone: | 33333 |

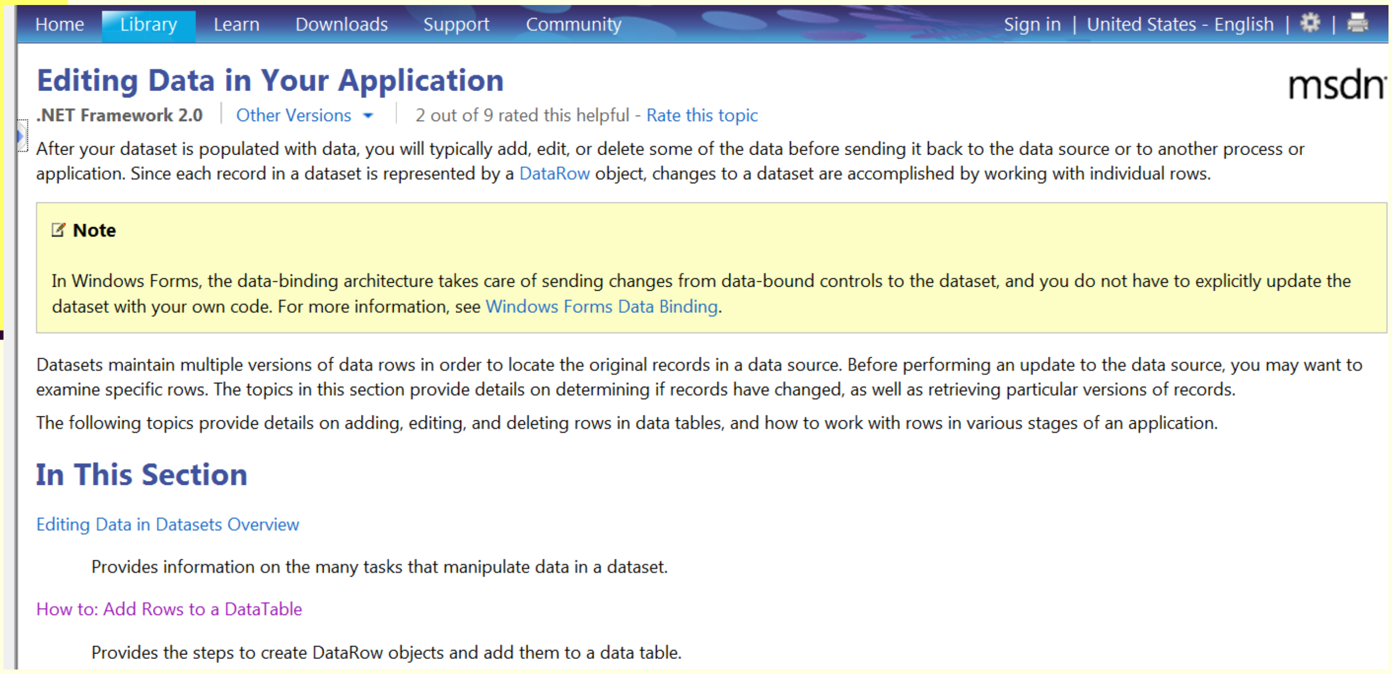
Connection String

Data Source=(LocalDB)\v11.0;AttachDbFilename=c:\temp\WindowsFormsApplication1\Windows

- ❑ Enter a record and click Insert button.
- ❑ Your data should appear in the table on the left.
- ❑ For more records, make sure your Customer ID is UNIQUE (ie. cannot be same)

# More methods for accessing database using C# code

- To learn more about editing data in your application
- Refer to MSDN(Microsoft Developer Network) website:
  - http://msdn.microsoft.com/en-us/library/ms171928(v=vs.80)



Editing Data in Your Application

.NET Framework 2.0 | Other Versions | 2 out of 9 rated this helpful - Rate this topic

After your dataset is populated with data, you will typically add, edit, or delete some of the data before sending it back to the data source or to another process or application. Since each record in a dataset is represented by a DataRow object, changes to a dataset are accomplished by working with individual rows.

**Note**

In Windows Forms, the data-binding architecture takes care of sending changes from data-bound controls to the dataset, and you do not have to explicitly update the dataset with your own code. For more information, see Windows Forms Data Binding.

Datasets maintain multiple versions of data rows in order to locate the original records in a data source. Before performing an update to the data source, you may want to examine specific rows. The topics in this section provide details on determining if records have changed, as well as retrieving particular versions of records.

The following topics provide details on adding, editing, and deleting rows in data tables, and how to work with rows in various stages of an application.

In This Section

Editing Data in Datasets Overview

Provides information on the many tasks that manipulate data in a dataset.

How to: Add Rows to a DataTable

Provides the steps to create DataRow objects and add them to a data table.

## Summary

- Working with database in C# application
  - In 3 simple steps:
    1. Create a new SQL database
    2. Connect database objects with a data source
    3. Create controls to blind the table

- MSDN
  - MS development network
  - http://msdn.microsoft.com/en-us/default.aspx
  - Learn more about C#

## Practical 6A

**Project Work**

- Apply the lesson learnt into your project.
- Add a customer database to store customer details.
- Retrieve the customer data from the database for existing customer.

## Annex A: Entire Program of Insert Button

```
private void btnInsert_Click(object sender, EventArgs e)
 {
           int varID;
           string varName;
           string varTel;

           varID = int.Parse(tbID.Text);
           varName = tbName.Text;
           varTel = tbTel.Text;

string connect = "Data Source = (LocalDB)\\v11.0;
AttachDbFilename=c:\\temp\\database\\database\\school.mdf; Integrated
Security=True";
           lblConnect.Text = connect;

           SqlConnection myConnect = new SqlConnection(connect);
           myConnect.Open();
           SqlCommand cmd = new SqlCommand();
           cmd.Connection = myConnect;
           cmd.CommandText = "INSERT INTO Customer(custID, custName, custTel)
values (@xid, @xname, @xtel)";
           cmd.Parameters.Add(new SqlParameter("@xid", varID));
           cmd.Parameters.Add(new SqlParameter("@xname", varName));
           cmd.Parameters.Add(new SqlParameter("@xtel", varTel));
           cmd.ExecuteNonQuery();

           DataTable dt = new DataTable();
           SqlDataAdapter orange = new SqlDataAdapter("Select * from Customer",
myConnect);

           orange.Fill(dt);
           customerDataGridView.DataSource = dt;
           customerDataGridView.Update();
           customerDataGridView.Refresh();
           myConnect.Close();
}
```

# End of Topic 6A

Storing Data
Using SQL Database