

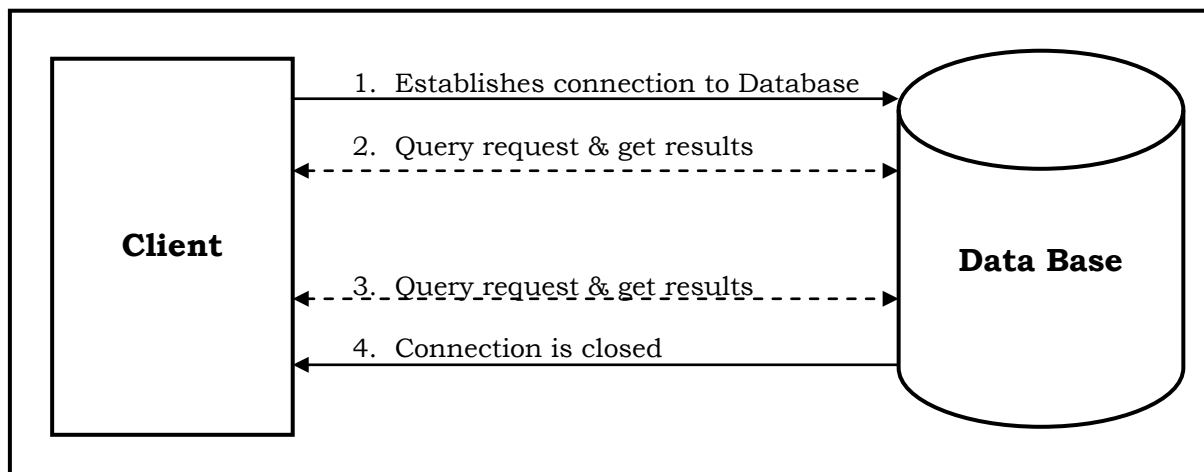
## Chapter -6

# Connecting Database with ADO

### 6.1 ADO.NET Architecture

#### Traditional ADO :

When we are working with dynamic web application the data needs to be stored in a centralized location i.e. database. Data is stored in database and whenever data is needed in a web page it is requested from database. To get data from database or to manipulate/insert data in database, the mechanism is illustrated in diagram below and it is known as **ADO (ActiveX Data Object)**.



**Figure: Working mechanism of classical ADO**

As shown in diagram above, to work with the data stored in database, following steps are processed:

**Step 1:** Connection to database is established.

**Step 2:** Query processing takes place and appropriate result is delivered, if the connection is established successfully.

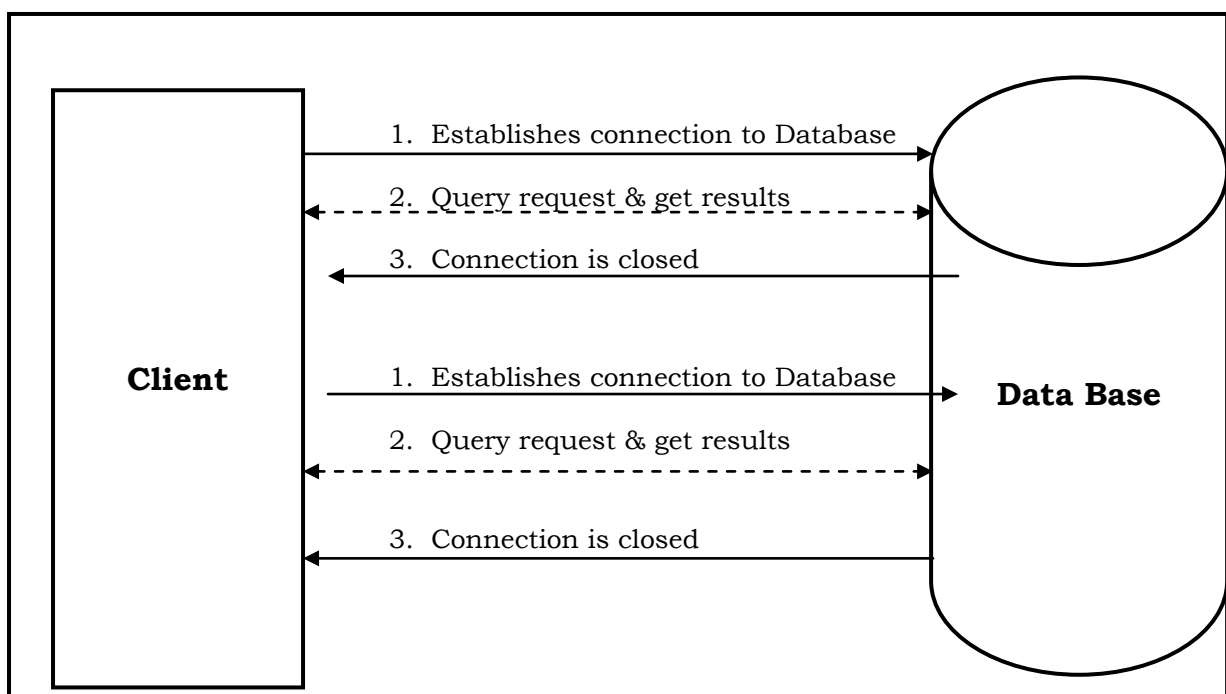
**Step 3:** Step 2 is repeated till there is a need of working on data. Any numbers of queries can be processed one after another.

**Step 4:** The connection is cancelled/closed when the need of data is finished.

There are some drawbacks of classical ADO mechanism. Once the connection has been established successfully, it remains open till it is not closed. The time span of closing the connection is not fixed and any number of queries can be performed on open connection. The time span between two consecutive queries is also not fixed and can be a long duration. So throughout this time the connection remains open, which in fact is not appropriate from data security point of view as the data becomes vulnerable.

### **ADO.NET :**

To outcome the drawbacks of classical ADO, ADO.Net is used. It is an extended version of ADO. ADO stands for **ActiveX Data Object** here as well, but the working mechanism is changed. The diagram given below illustrates the working mechanism of ADO.Net.



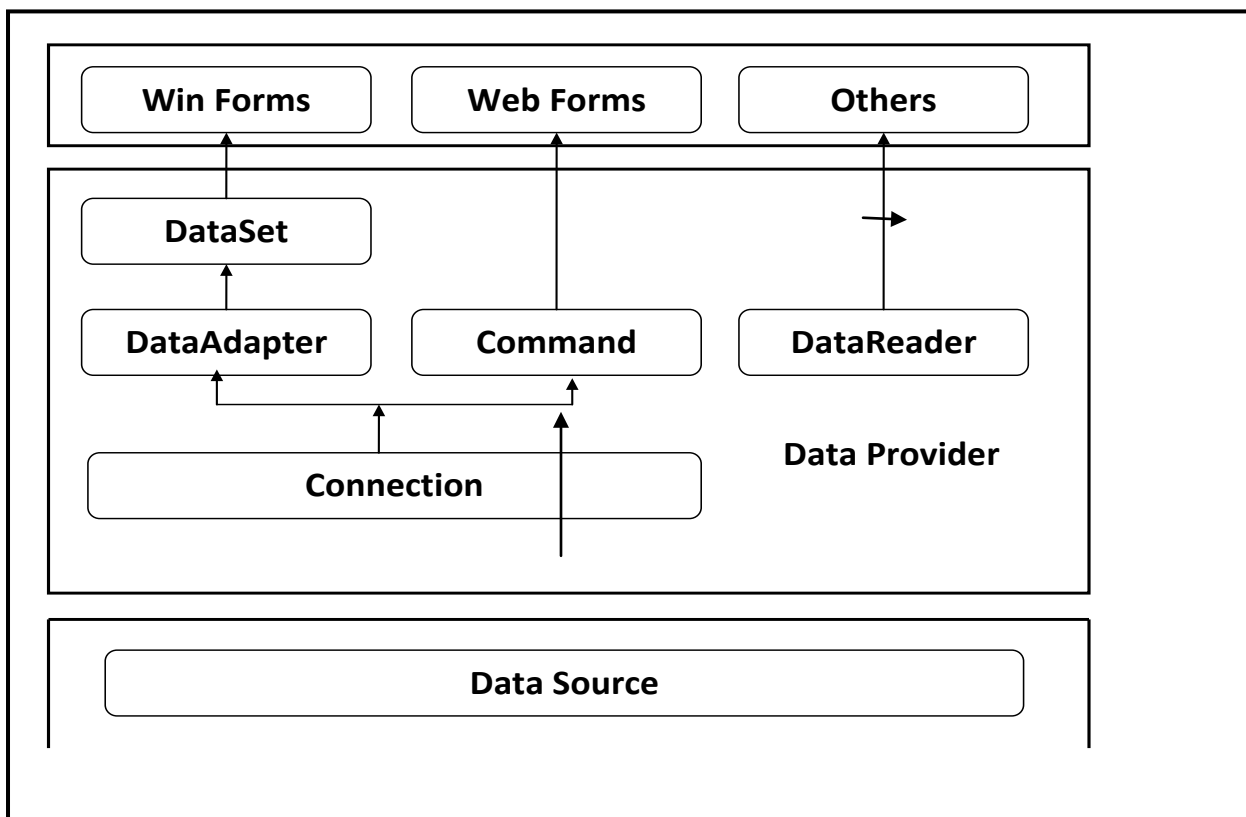
**Figure: Working mechanism of ADO.Net**

The only difference in ADO.Net is the connection to the database. As shown in figure, the connection is established, query is processed & result is delivered. After one query is processed, the connection is closed immediately.

So, connection doesn't remain open till all query processing is finished. Instead connection is opened & closed for each query. Whole process is repeated every time there is a need of working on data. This increases the security of data because as soon as query processing is finished, connection is closed.

ADO.Net works with two different ways and they are **Connected Environment** and **Disconnected Environment**. In **connected environment**, database connection is established while query is being processed. It means till the time query is being processed the database is accessible. While in **disconnected environment**, a copy of data from database is stored in local buffer and is used for query processing. This buffer can be DataSet.

#### ADO.NET Architecture:



ADO.Net is a data access technology which handles the data manipulation and regarding query processing. It is used with ASP.Net for storage and usage of data. It is used in visual studio i.e. IDE for developing .Net applications. The

structure of ADO.Net is given in the diagram below which is known as ADO.Net architecture.

### 6.1.1 Data Provider

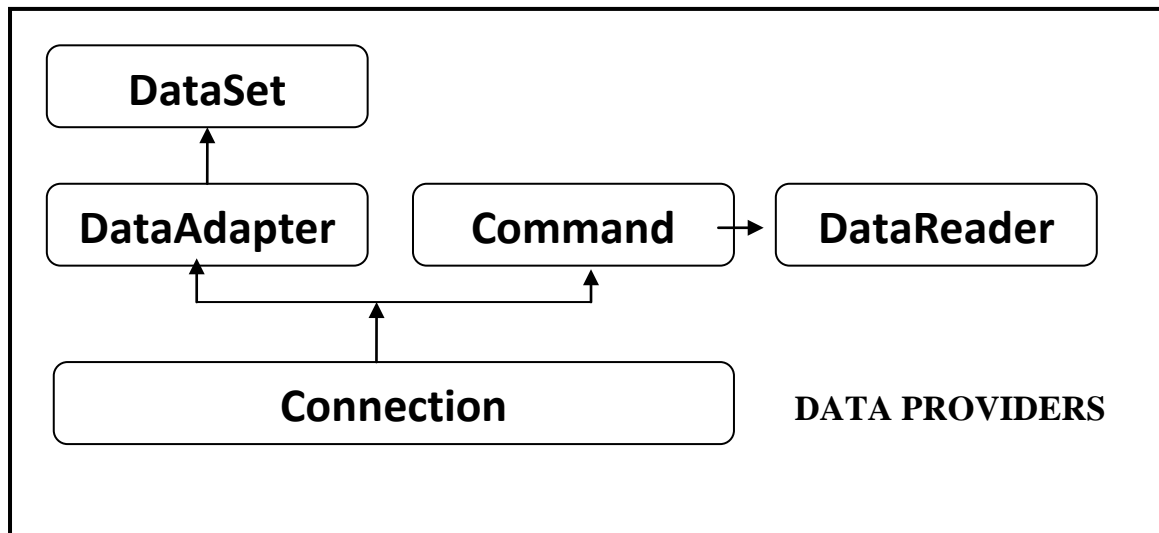
ADO.Net works with three layers and all layers are linked with each other. The layers are **Data Source, Data Provider and Forms**.

- a. **Data Source:** It is the storage of actual data i.e. Database.
- b. **Data Provider:** it is provider of data, depending upon the need and type of database. It contains built-in classes for each type of database used in .Net. OLEDB for MS Access, SQL for SQL Database, ODP for Oracle and ODBC for other DBMSs.
- c. **Forms:** Top level layer of ADO.Net using which data is manipulated or retrieved from Database.

As shown in diagram above, connection object is used to connect application to database. Connection object can be used with two data object component of ADO.Net i.e. **DataAdapter** and **DataCommand**. Data containers (local buffers) are available with both components. If you are using DataAdapter, DataSet is used to store the data while DataReader is used with DataCommand. ADO.Net provides set of classes and objects for data access from database. There are main four components of ADO.Net.

- 1. **Connection Object**
- 2. **Command Object**
- 3. **DataAdapter Object**
- 4. **DataReader Object**

The .Net Framework includes mainly three Data Providers for ADO.Net i.e. Microsoft SQL Server Data Provider, OLEDB Data Provider & ODBC Data Provider. SQL Server uses **SqlConnection** object, OLEDB uses **OleDbConnection** object & ODBC uses **OdbcConnection** object respectively.



#### ADO.NET Objects :

Connection	Used for connecting data source to application. Uses connection string for locating data source (database).
Command	Used for executing database queries.
DataReader	Used as a data container for data retrieval. Used with Data Command.
DataAdapter	Works same as command object but works in disconnected environment.

#### Database Providers and their according namespaces:

MS Access	Imports System.Data.OleDb
SQL	Imports System.Data.Sql
Oracle	Imports System.Data.Oracle

#### ADO.Net Components :

Dataset	Local data buffer of retrieved records from database. Used with DataAdapter.
Data Table	Used to contain the data in tabular form with rows & columns.
Data Row	Represents the single row in a data table.

Data Column	Represents the single column or field of a data table.
Data Relation	Represents the relations between the different data tables or datasets.
Constrains	Represent the constrains applied to the field of databale.

### 6.1.2 Connection Object

The ADO Connection Object is used to create an open connection to a data source. Through this connection, you can access and manipulate a database.

If you want to access a database multiple times, you should establish a connection using the Connection object. You can also make a connection to a database by passing a connection string via a Command or Recordset object.

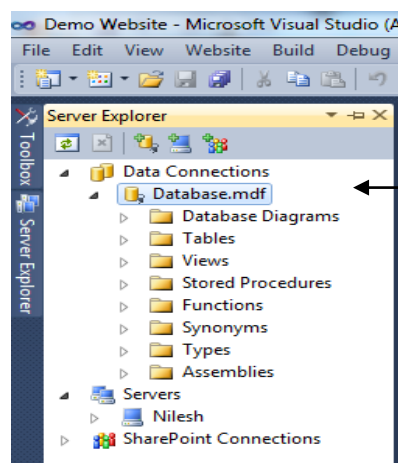
Before going into details of connection object, the basic primary thing needed is location of database. The actual location of database is called **Connection String**.

#### Syntax:

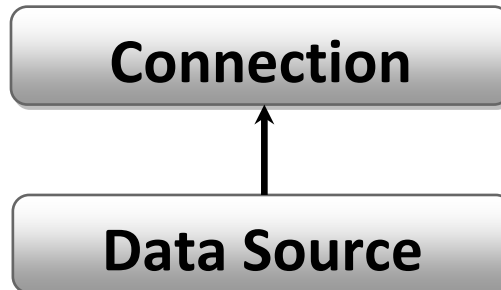
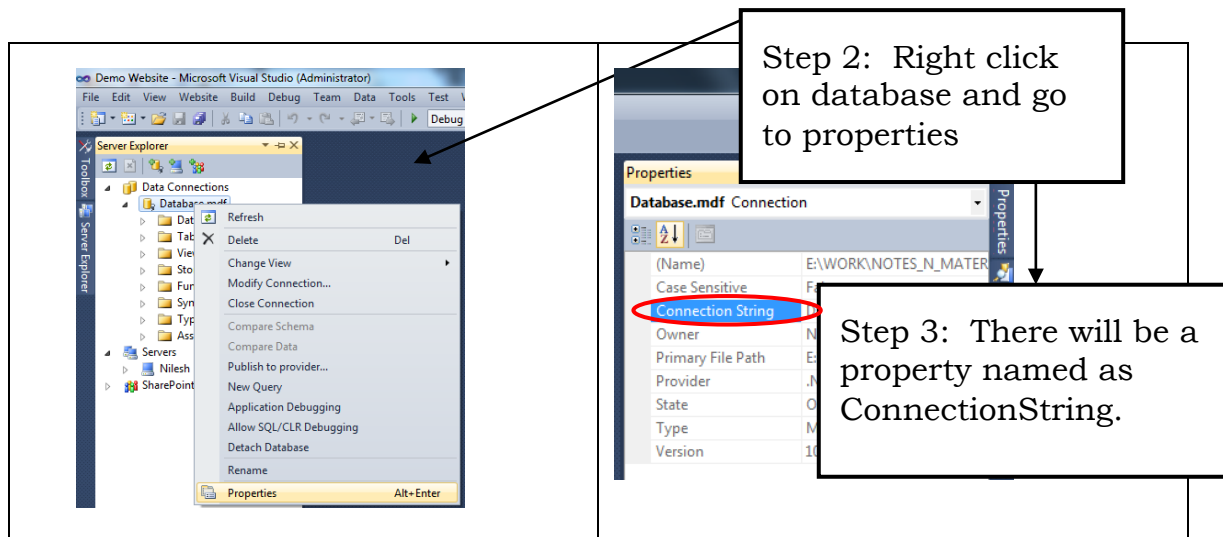
```
Dim str as String = "Connection string data"
```

#### How to get Connection String:

There are numbers of ways using which connection string can be achieved. One of them is using database in server explorer in visual studio after database has been created.



Step 1: Go to server explorer in visual studio



**Figure: Connection Object**

**Syntax for Connection object:**

```
Dim obj_name as providerConnection(connection string)
```

where obj\_name = name of the object for database classes to perform queries.

Provider = database provider (SQL, Oracle, Access, etc).

There are many data providers like SQL, Oracle and Access.

First you have to get connection string as shown above.

**Example:****1. SQL as data provider**

```
Dim str As String
str = "Data Source=.\SQLEXPRESS;AttachDbFilename='E:\Demo
      Website\App_Data\Database.mdf';IntegratedSecurity=True;
      User Instance=True"

Dim cn As New SqlConnection(str)
```

**2. Access as data provider**

```
Dim str As String

str = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\myFolder\
      myAccessFile.accdb; Persist Security Info=False"

Dim cn As New OleDbConnection(str)
```

Property	Description
DataSource	Name of the data source for data manipulation
Database	Name of the database in data source for establishing connection and access data.
ConnectionString	Actual location of database for connecting and data access.
ConnectionTimeout	A time span after which the connection will be expired.

Method	Description
Open()	Used to establish the connection to the database.
Close()	Used to close the connection to the database after query execution.



**Example : [using SQL as data provider]****VB Code :** conn\_demo.aspx.vb**Imports System.Data.SqlClient**

Partial Class conn\_demo Inherits System.Web.UI.Page

Protected Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

Dim str As String

```
str = "Data Source=. \SQLEXPRESS;AttachDbFilename='E:\Demo
      Website\App_Data\Database.mdf;IntegratedSecurity=True;
      User Instance=True"
```

Dim cn As New SqlConnection(str)

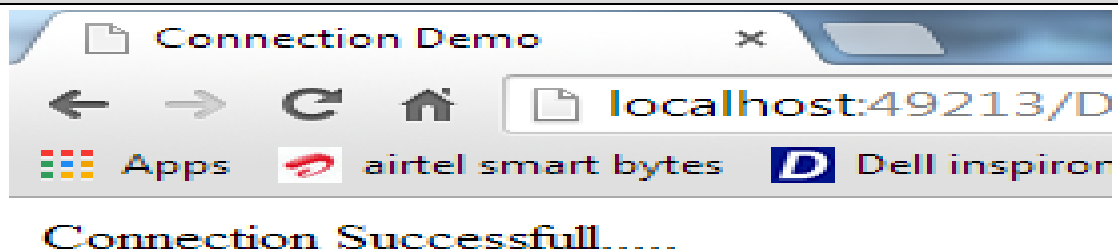
**cn.Open()** ← Establishes Connection

Response.Write("Connection Successfull.....")

**cn.Close()** ← Closes Connection

End Sub

End Class

**Output View :****Steps for defining & using Connection Object:**

1. Add the namespace of Data Provider you want to use and that can be SQL (Sql), Oracle (ODBC) or Access (OLEDB).
2. Write a connection string in a string variable;
3. Create an instance of the connection class of your selected Data Provider.
4. Give the connection string variable as an argument in it.
5. Open the connection using Open() method [ Syntax : object.Open(); ]
6. Execution Code
7. Close the connection using Close() method. [ Syntax : object.Close(); ]

**In the Code Blocks illustrated above,**

1. **An import is** the keyword that is used to include/import namespace to the application.
2. System.Data.Sql is a namespace that contains all the classes & definitions of Database connections.
3. For MS-ACCESS database connection, OLEDB is used.
4. For Sql Server Database connection, Sql, SqlConnection, SqlTypes are used.
5. If MS-ACCESS is a database then database extension will be .accdb/.mdb.
6. If Sql Server is a database then database extension will be .mdf.

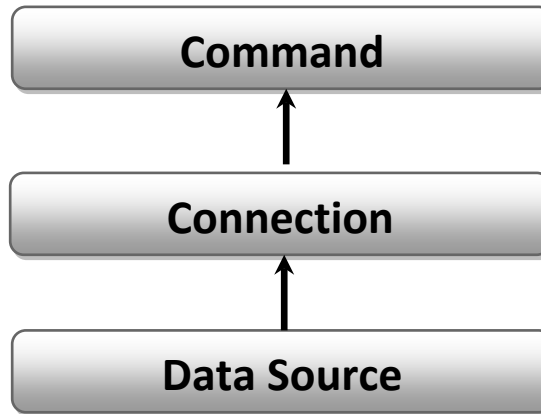
OleDbConnection and SqlConnection are the classes for defining connection object for MS-Access & Sql Server, respectively.

### 6.1.3 Command Object

The ADO.Net Command Object is used to execute single query against database and retrieve appropriate results. The query to be executed can be of inserting, deleting, updating or selecting data record.

If data is retrieved from database using Command Object, the result has to be contained somewhere and it is local data container **DataReader**. Otherwise in

any other queries, the code is executed and number of rows affected is returned.



**Figure: Command Object**

**Syntax for Command object:**

```
Dim obj_name as new providerCommand(query, connection string)
```

Where obj\_name = name of the object for DataCommand class.

Provider = database provider (SQL, Oracle, Access, etc).

There are many data providers like SQL, Oracle and Access.

**Example:**

**1. SQL as data provider**

```
Dim cmd As New SqlCommand("Select * from Table1",cn)
```

Here, cn is connection object and cmd is instance of SqlCommand class.  
SQL is used as Data Provider.

**2. Access as data provider**

```
Dim cmd As New OleDbCommand("Select * from Table1",cn)
```

Here, cn is connection object and cmd is instance of OleDbCommand class.  
Access is used as Data Provider.

As command object works in connected environment, the query has to be executed in an open connection. So first we have to open the connection using `Open()` method and then query is executed.

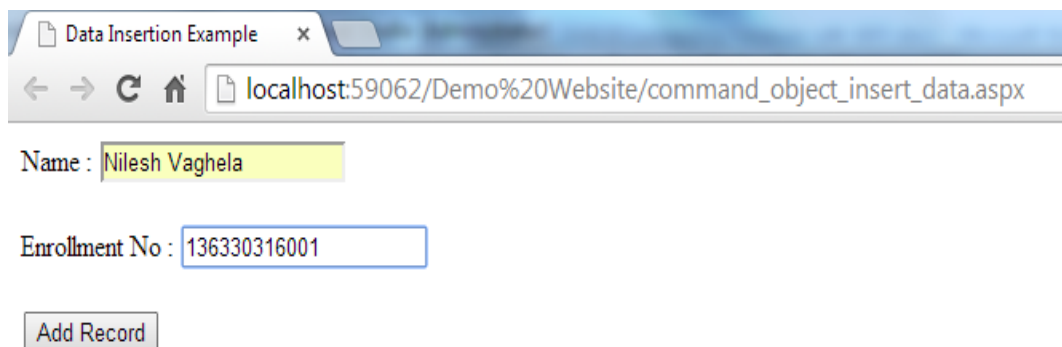
An instance of Command object has to be created first. Each .NET Framework data provider included with the .NET Framework has its own command object that inherits from `DbCommand`.

The .NET Framework Data Provider for OLE DB is an **OleDbCommand** object, for SQL Server is a **SqlCommand** object, for ODBC is an **OdbcCommand** object.

The query is executed using any of the available methods with command object.

Method	Description
<code>ExecuteReader()</code>	Returns a <code>DataReader</code> object.
<code>ExecuteScalar()</code>	Returns a single scalar value.
<code>ExecuteNonQuery()</code>	Executes a Transact-SQL statement against the connection and returns the number of rows affected
<code>ExecuteXMLReader()</code>	Returns an <code>XmlReader</code> . Available for <code>SqlCommand</code> object only.
<b>Example: Inserting data record into database using Command Object.</b>	
<b>Design Code:</b> <code>command_object_insert_data.aspx</code>	
<pre> &lt;%@ Page Language="VB" AutoEventWireup="false" CodeFile="command_object_insert_data.aspx.vb" Inherits="command_object_insert_data" %&gt;  &lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"&gt;  &lt;html xmlns="http://www.w3.org/1999/xhtml"&gt; &lt;head runat="server"&gt; </pre>	

```
<title>Data Insertion Example</title>
</head>
<body>
  <form id="form1" runat="server">
    Name :
    <asp:TextBox ID="txt_name" runat="server"></asp:TextBox>
    <br /><br />
    Enrollment No :
    <asp:TextBox ID="txt_enroll" runat="server"></asp:TextBox>
    <br /><br />
    <asp:Button ID="btn_add_record" runat="server" Text="Add Record" />
  </form>
</body>
</html>
```



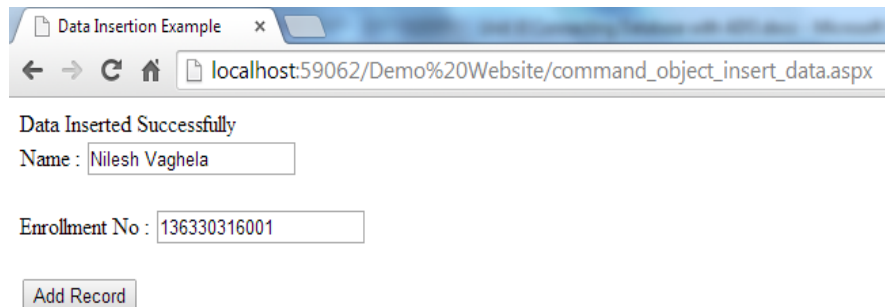
**VB Code :** command\_object\_insert\_data.aspx.vb

**Imports System.Data.SqlClient**

Partial Class command\_object\_insert\_data Inherits System.Web.UI.Page  
Protected Sub btn\_add\_record\_Click(ByVal sender As Object, ByVal e As  
System.EventArgs) Handles btn\_add\_record.Load

```
    Dim str As String
    str = "Data Source=.\SQLEXPRESS;AttachDbFilename='E:\Demo
        Website\App_Data\Database.mdf';IntegratedSecurity=True;
        User Instance=True"
    Dim cn As New SqlConnection(str)
```

```
Dim q As String = "insert into stdent_details_master values(' " +  
txt_name.Text.Trim() + " ',' " + txt_enroll.Text.Trim() + " ')"  
  
Dim cmd As New SqlCommand(q, cn)  
  
cn.Open()  
cmd.ExecuteNonQuery() ← Executes Query  
Response.Write("Data Inserted Successfully")  
cn.Close()  
End Sub  
End Class
```

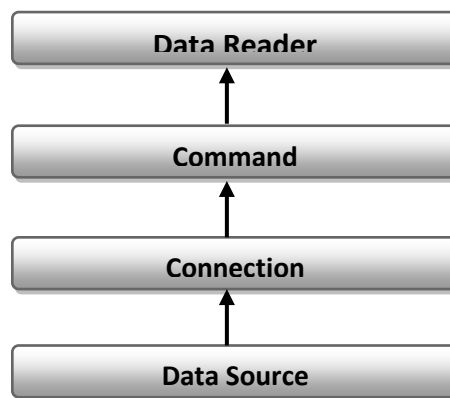
**Output View :**

### 6.1.4 DataReader Object

ADO.Net command object executes the query and returns appropriate result. If the query to be executed is select statement then data record(s) will be retrieved. These data record(s) has to be contained somewhere for access and manipulation.

**DataReader is local buffer or data container that contains the result of select statements executed by command object.** DataReader creates a temporary data table depending upon results retrieved from database. DataReader works only in connected environment so it has to be created in open connection.

DataReader is created when a particular select statement is executed with the help of Command Object. **ExecuteReader()** is the method that executes Select statement and stores the result data into created DataReader.



**Figure: DataReader Object**

**Example: Search a data record from database using DataReader Object.**

**Design Code:** datareader\_demo.aspx

**VB Code :** datareader\_demo.aspx.vb

**Imports System.Data.SqlClient**

Partial Class datareader\_demo Inherits System.Web.UI.Page

Protected Sub btn\_search\_Click (ByVal sender As Object, ByVal e As System.EventArgs) Handles btn\_search.Load

```
    Dim str As String
    str = "Data Source=.\SQLEXPRESS;AttachDbFilename='E:\Demo
        Website\App_Data\Database.mdf;IntegratedSecurity=True;
        User Instance=True"
    Dim cn As New SqlConnection(str)
    Dim q As String = "select * from stdent_details_master where
        name = ' " + txt_name.Text.Trim() + " ' "

    Dim cmd As New SqlCommand(q, cn)

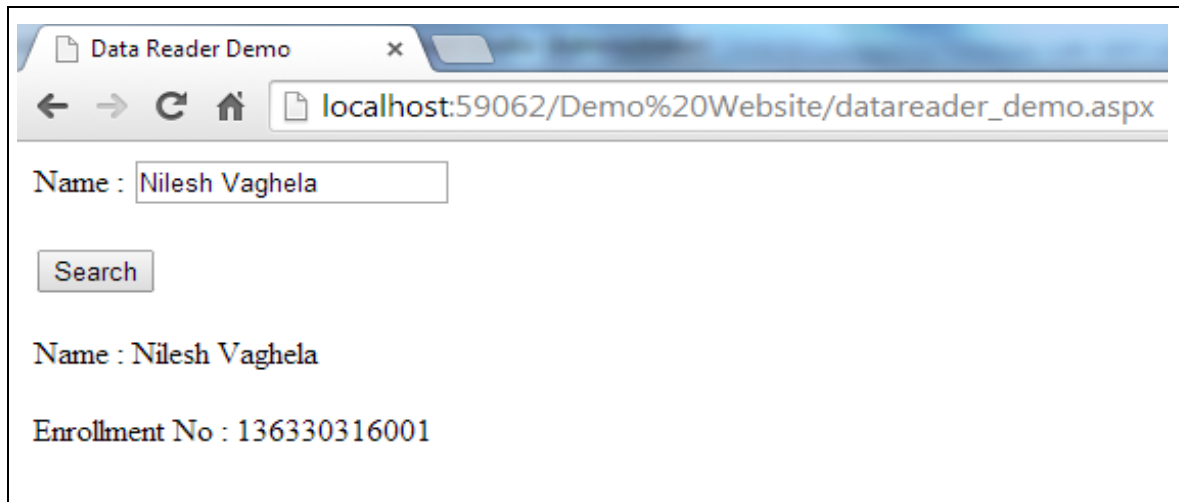
    cn.Open()
    Dim dr As SqlDataReader
    dr = cmd.ExecuteReader()
    If dr.Read() Then
        lbl_name.Text = dr("name").ToString()
        lbl_enroll.Text = dr("enrollment_no").ToString()
        cn.Close()
    Else
        Response.Write("No Such Data Found")
        cn.Close()
    End If
End Sub

End Class
```

**Note:** Read() is a method that checks if data is available in data reader or not and advances the counter by +1. By default counter is set to -1 when data reader is created.

**Output View :**

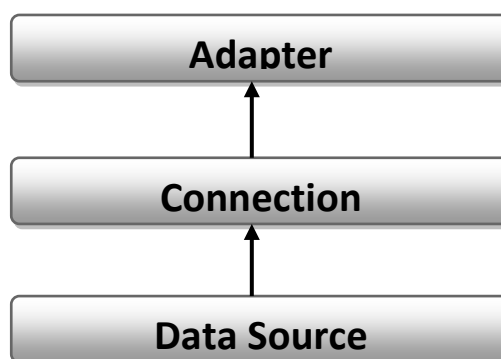




### 6.1.5 DataAdapter Object

ADO.Net provides disconnected data environment. When data is contained in local buffer/container from actual database and all SQL statement execution is done on the local buffer data, this is known as **disconnected data environment**. Because web page is not connected to the actual database for data retrieval or manipulation, instead it works on local copy of database. Hence it is known as disconnected data environment.

To work with disconnected data, first it has to be contained in local buffer. This local buffer is DataSet. To bridge the gap between Database and DataSet, DataAdapter is used.



**Figure: DataAdapter Object**

ADO.Net has a class of DataAdapter which is associated with DataSet. DataAdapter works the same way as DataCommand works but DataAdapter provides disconnected data that is the only difference

**Syntax for DataAdapter object:**

```
Dim obj_name as New providerDataAdapter(query, connection string)
Or
Dim obj_name as New providerDataAdapter(command_object)
```

Where obj\_name = name of the object for DataAdapter class.

Query = SQL statement to be executed

Connection string = location of database

Provider = database provider (SQL, Oracle, Access, etc).

**Example:****1. SQL as data provider**

```
a) Dim cmd As New SqlCommand("Select * from Table1",cn)
```

```
Dim da As New SqlDataAdapter(cmd)
```

Or

```
b) Dim da As New SqlDataAdapter("Select * from Table1",cn)
```

**Note: Similarly it can be created for OleDb Data Provider as well**

DataAdapter provides four properties that indicate database commands and are:

1. SelectCommand
2. InsertCommand
3. UpadteCommnad
4. DeleteCommand

DataAdapter works with Dataset to contain the results retrieved from SelectCommand.

DataSet is a collection of DataRows, DataColumnms, and DataConstrains.

**Example: Illustartion of DataAdapter and DataSet Object.****VB Code :** dataadapter\_demo.aspx.vb**Imports System.Data.SqlClient**

Partial Class dataadapter\_demo Inherits System.Web.UI.Page

Protected Sub Page\_Load (ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

```
Dim str As String
str = "Data Source=. \SQLEXPRESS;AttachDbFilename='E:\Demo
      Website\App_Data\Database.mdf';IntegratedSecurity=True;
      User Instance=True"
```

```
Dim cn As New SqlConnection(str)
```

```
Dim q As String = "select * from stdent_details_master"
```

```
Dim da As New SqlDataAdapter(q, cn)
```

```
Dim ds As New DataSet
```

```
Da.Fill(ds)
```

```
End Sub
```

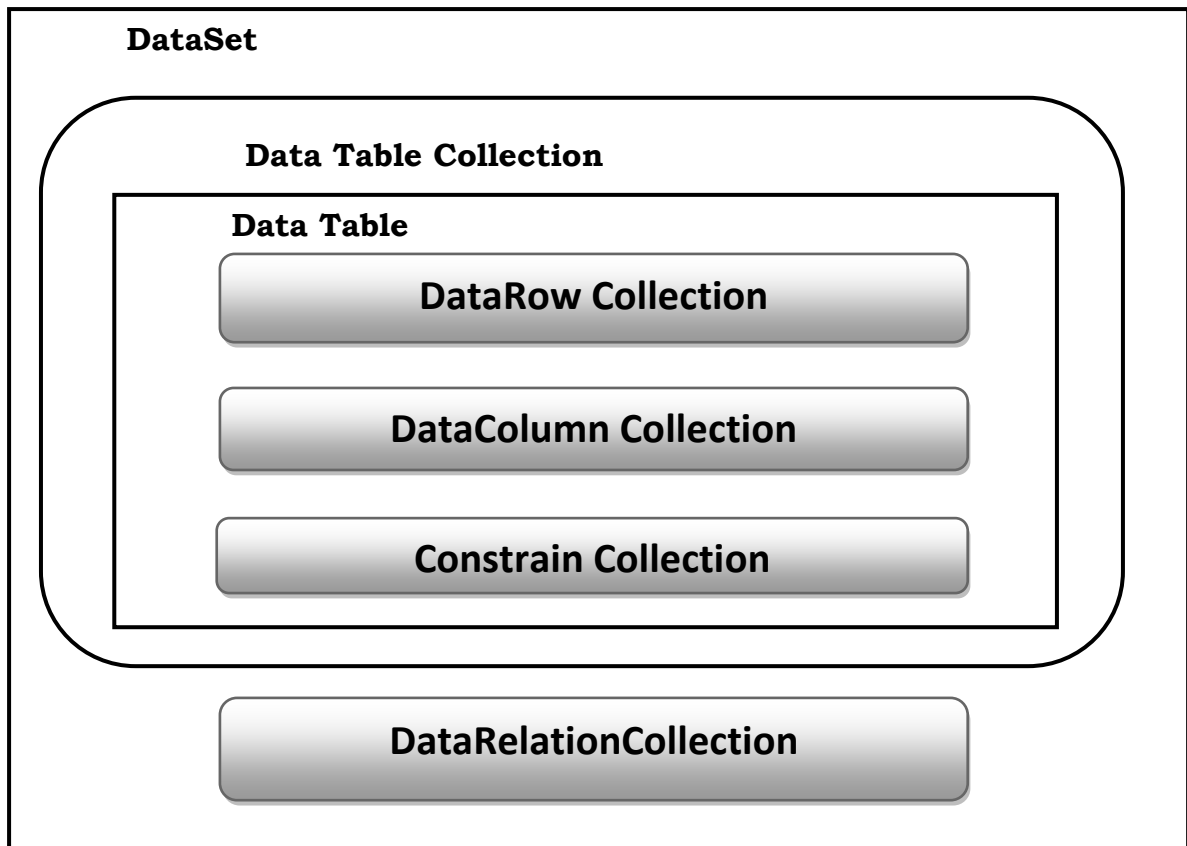
```
End Class
```

**Note:** Fill() is a method that executes the select query, retrieves the result and stores data in local buffer i.e. DataSet. It accepts dataset as argument.

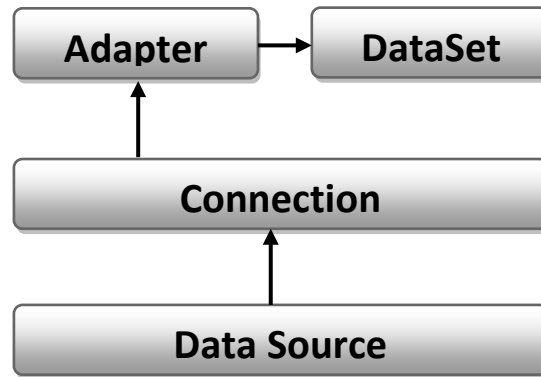
### 6.1.6 DataSet Object

ADO.Net works in disconnected environment. To work in disconnected environment, first it has to be contained in local buffer. This local buffer is DataSet. To bridge the gap between Database and DataSet, DataAdapter is used.

DataSet is a collection of **DataRow**s, **DataColumn**s, and **DataConstraints**. DataSet can be created explicitly by instantiating with DataSet Class. Empty DataSet is created when we make an object of DataSet class. Using Fill() the result of Select statement is stored in it.



**Figure: DataSet Architecture**



**Figure: DataSet Object**

As shown in DataSet architecture, it consists of DataTable Collection. There can be one or more DataTables in a single DataSet. Going further in detail, DataTable consist of DataRow Collection, DataColumn Collection and Constrains Collection. DataRow indicates data records of data row. DataColumn indicates data of a single record.

Constrains Collection is set of constrains that are applied to DataTable. DataRelation Collection is used to indicate how two or more DataTables are related to each other.

DataSet is used to populate DataTable from the result retrieved from Select query. DataSet has a DataTable collection, so it will be indexed and index will start from 0.

DataSet can be assigned to any of the controls that can have data collection. These ASP server controls can be DropDownList, RadioButtonList, CheckBoxList, GridView, ListBox, DataList, etc. DataSet acts as a source of data for these controls to contain data from database dynamically.

**Syntax for DataSet object:**

```
Dim obj_name as New DataSet
```

Where obj\_name = name of the object for DataSet class.

DataSet is always used with DataAdapter for any Data Provider.

**Example:****1. DataSet with SQL as data provider**

a) Dim cmd As New SqlCommand("Select \* from Table1",cn)

Dim da As New SqlDataAdapter(cmd)

Dim ds As New DataSet

da.Fill(ds)

controlid.DataSource = ds

Or

b) Dim da As New SqlDataAdapter("Select \* from Table1",cn)

Dim ds As New DataSet

da.Fill(ds)

controlid.DataSource = ds

**Note:** Similarly it can be created for OleDb Data Provider as well

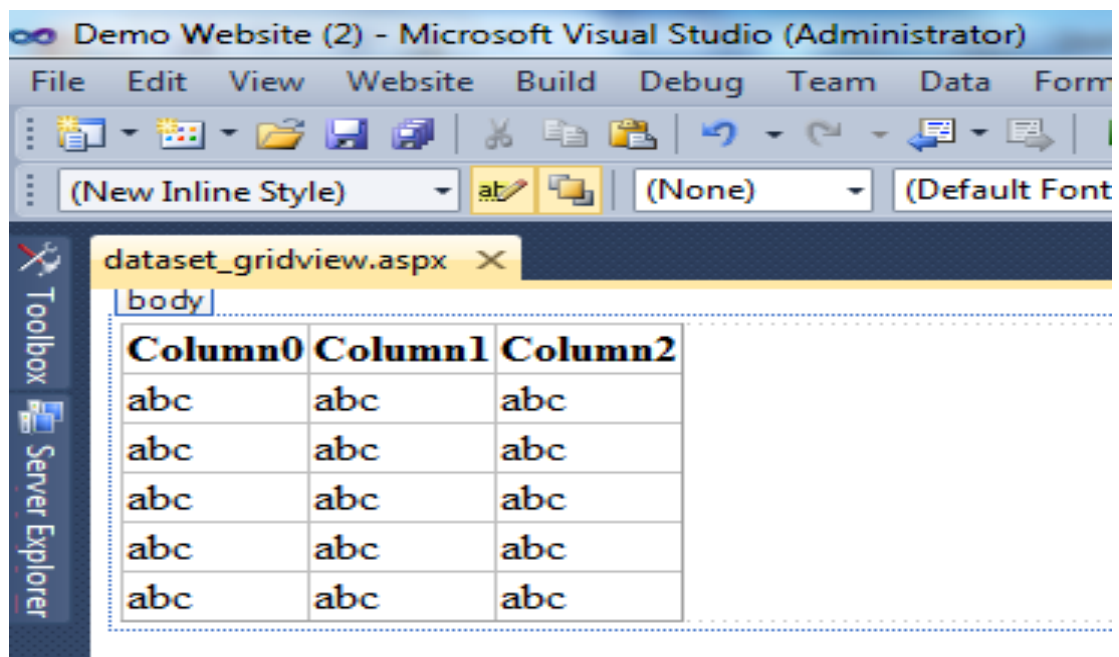
**Example: Filling GridView using DataAdapter and DataSet Object.****Design Code:** dataset\_gridview.aspx

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="dataset_gridview.aspx.vb" Inherits="dataset_gridview" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>DataSet Demo</title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:GridView ID="GridView1" runat="server">
```

```
</asp:GridView>  
</form>  
</body>  
</html>
```



**VB Code :** dataset\_gridview.aspx.vb

**Imports System.Data.SqlClient**  
**Imports System.Data**

Partial Class dataset\_gridview Inherits System.Web.UI.Page

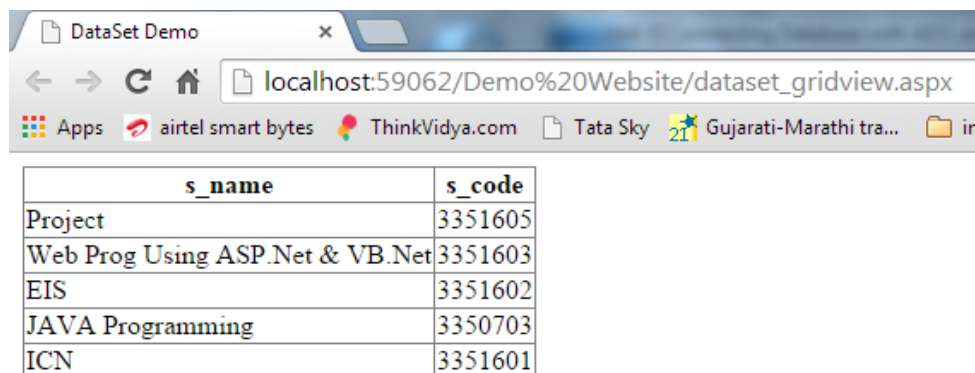
Protected Sub Page\_Load(ByVal sender As Object, ByVal e As  
 System.EventArgs) Handles Me.Load

```

    Dim str As String
    str = "Data Source=.\SQLEXPRESS;AttachDbFilename='E:\Demo
        Website\App_Data\Database.mdf';IntegratedSecurity=True;
        User Instance=True"
    Dim cn As New SqlConnection(str)
    Dim q As String = "select * from subject_master"
    Dim da As New SqlDataAdapter(q, cn)
    Dim ds As New DataSet
    da.Fill(ds)
    GridView1.DataSource = ds
    GridView1.DataBind()
  
```

End Sub  
 End Class

#### Output View :



s_name	s_code
Project	3351605
Web Prog Using ASP.Net & VB.Net	3351603
EIS	3351602
JAVA Programming	3350703
ICN	3351601



### 6.1.7 DataView

A **DataView** enables you to create different views of the data stored in a **DataTable**, a capability that is often used in data-binding applications. Using a **DataView**, you can expose the data in a table with different sort orders, and you can filter the data by row state or based on a filter expression.

A **DataView** provides a dynamic view of data in the underlying **DataTable**: the content, ordering, and membership reflect changes as they occur. This behavior differs from the **Select** method of the **DataTable**, which returns a **DataRow** array from a table based on a particular filter and/or sort order: this content reflects changes to the underlying table, but its membership and ordering remain static. The dynamic capabilities of the **DataView** make it ideal for data-binding applications.

A **DataView** provides you with a dynamic view of a single set of data, much like a database view, to which you can apply different sorting and filtering criteria. Unlike a database view, however, a **DataView** cannot be treated as a table and cannot provide a view of joined tables. You also cannot exclude columns that exist in the source table, nor can you append columns, such as computational columns, that do not exist in the source table.

To use **DataView**, you have to make an object of the class **DataView**. This object then can be used with different available properties and methods of **DataView**. **DataView** works with **DataSet** and the data records of **DataSet** can be manipulated for displaying using it.

**DataView** contains two main properties for use: **Sort** and **RowFilter**.

#### a) Sort

This property of **DataView** provides the functionality of sorting data by specifying column name. Data can be displayed in ascending or descending order. **DataSet** works with data of **DataTable** available in **DataSet**.

**Syntax:**

```
Dim obj_name as New DataView(ds.Tables(0))  
obj_name.sort = "column_name"  
Or  
obj_name.sort = "column_name DESC/ASC"  
Or  
obj_name.sort = "column_name1,column_name2"  
Or  
obj_name.sort = "column_name1 DESC/ASC, column_name2 ASC/DESC"
```

Where obj\_name = name of the object for DataView class.

Sort = property for DataView

Multiple data columns can be used for sorting in sequential order.

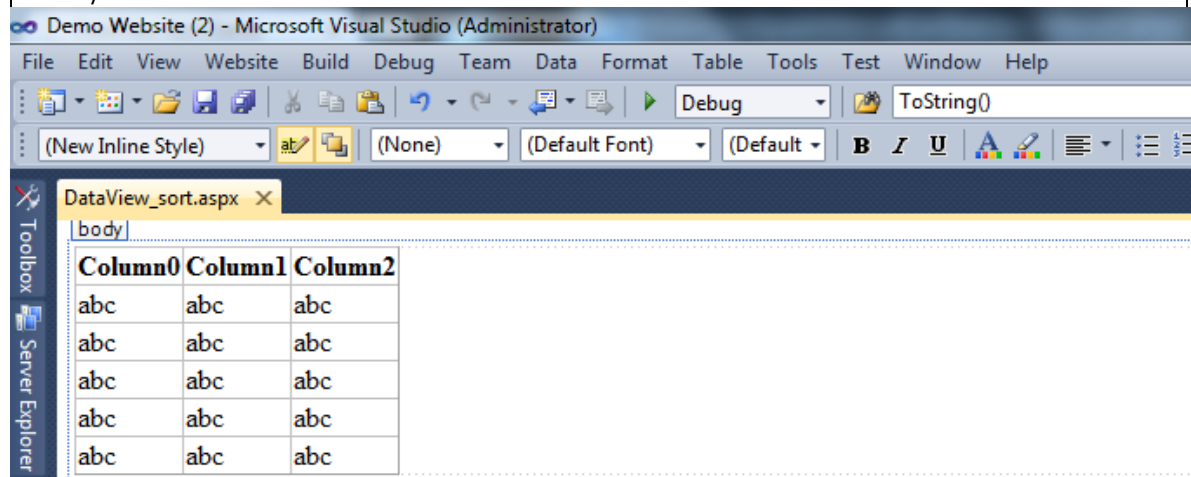
**Example: Filling GridView using DataView Object and sorting data.**

**Design Code:** DataView\_sort.aspx

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="DataView_sort.aspx.vb" Inherits="DataView_sort" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>DataView Sort Demo</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:GridView ID="GridView1" runat="server"></asp:GridView>
    </form>
</body>
</html>
```



**VB Code :** DataView\_sort.aspx.vb

```
Imports System.Data.SqlClient
Imports System.Data
```

```
Partial Class DataView_sort Inherits System.Web.UI.Page
```

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
```

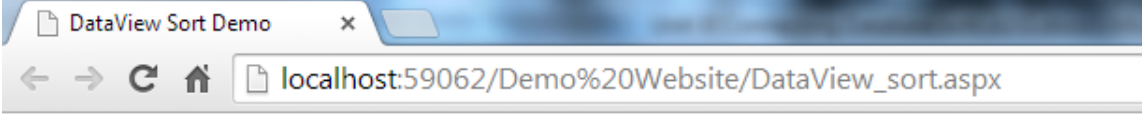
```
    Dim str As String
    str = "Data Source=.\SQLEXPRESS;AttachDbFilename='E:\Demo
        Website\App_Data\Database.mdf;IntegratedSecurity=True;
        User Instance=True"
```

```
    Dim cn As New SqlConnection(str)
    Dim q As String = "select * from subject_master"
    Dim da As New SqlDataAdapter(q, cn)
    Dim ds As New DataSet
    da.Fill(ds)
```

```
    Dim dv As New DataView(ds.Tables(0))
    dv.Sort = "s_code"
```

```
    GridView1.DataSource = dv
    GridView1.DataBind()
```

```
End Sub
End Class
```

**Output View :**


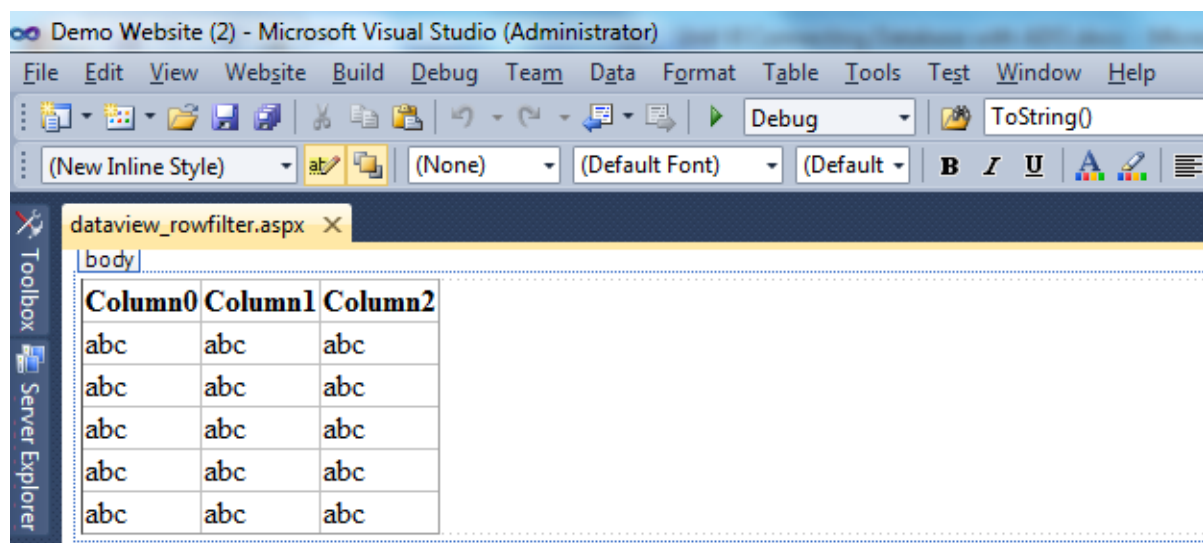
s_name	s_code
JAVA Programming	3350703
ICN	3351601
EIS	3351602
Web Prog Using ASP.Net & VB.Net	3351603
Project	3351605

**b) RowFilter**

If we want data to be displayed upon certain conditions, DataView has a property of RowFilter. RowFilter provides conditional filtering with RowFilter.

**Example: Filling GridView using DataView Object and filtering data.**

**Design Code:** dataview\_rowfilter.aspx



```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile=" dataview_rowfilter.aspx.vb" Inherits=" dataview_rowfilter" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>DataView Row Filter Demo</title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:GridView ID="GridView1" runat="server"></asp:GridView>
  </form>
</body>
</html>
```

**VB Code :** dataview\_rowfilter.aspx.vb**Imports System.Data.SqlClient****Imports System.Data**

Partial Class dataview\_rowfilter Inherits System.Web.UI.Page

Protected Sub Page\_Load(ByVal sender As Object, ByVal e As

System.EventArgs) Handles Me.Load

Dim str As String

str = "Data Source=.\SQLEXPRESS;AttachDbFilename='E:\Demo  
Website\App\_Data\Database.mdf;IntegratedSecurity=True;  
User Instance=True"

Dim cn As New SqlConnection(str)

Dim q As String = "select \* from student\_detail\_master"

Dim da As New SqlDataAdapter(q, cn)

Dim ds As New DataSet

da.Fill(ds)

Dim dv As New DataView(ds.Tables(0))

dv.Sort = "s\_code"

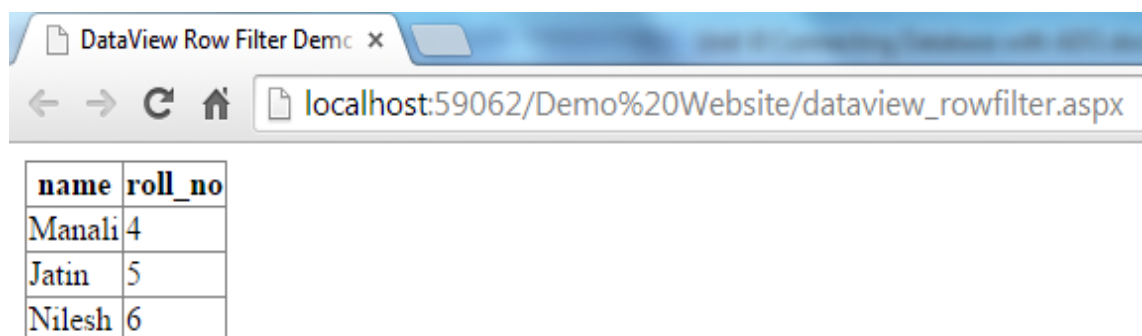
**dv.RowFilter = "roll\_no>3"**

GridView1.DataSource = dv

GridView1.DataBind()

End Sub

End Class

**Output View :**


name	roll_no
Manali	4
Jatin	5
Nilesh	6

## 6.2 Data Binding

### Types of Data Binding:

Data binding is used to bind soft data into ASP Server controls dynamically. If data is loaded into ASP Control at run time, it is called soft coded data. Soft coded data can be changed dynamically using VB code. Depending upon type of control used, different data is filled in it.

**Data Binding is a process of filling up an ASP control dynamically at run time using VB code or any control wizard.** Data binding can be used in two different ways depending upon which type of control is being used. For Data Binding, we need Data Source Controls. There are two types of data source controls.

#### 1. Tabular Data:

SQLDataSource

AccessDataSource

ObjectDataSource

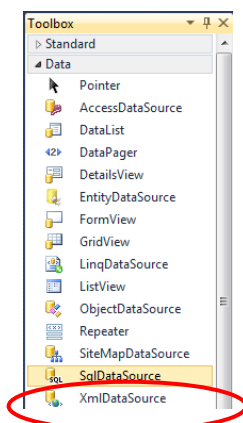
#### 2. Tabular Data and hierarchical Data:

XMLDataSource

SiteMapDataSource

### SQLDataSource:

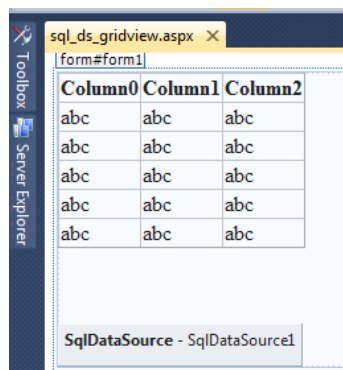
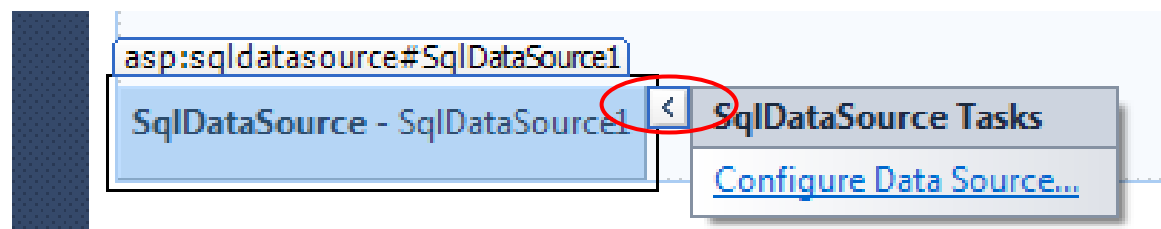
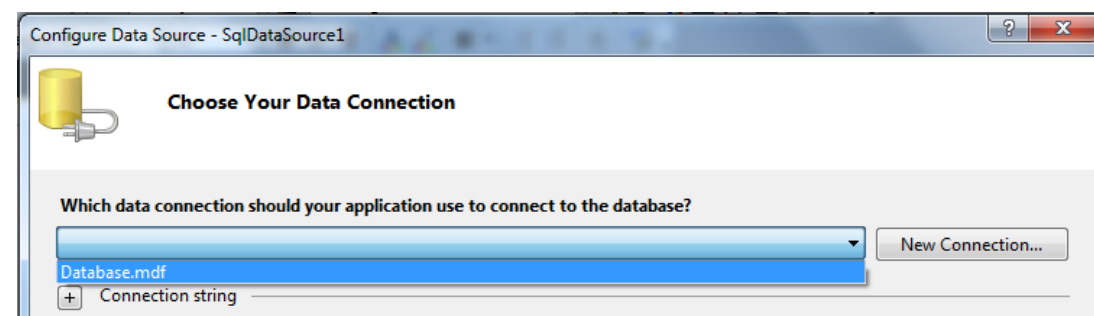
The SQLDataSource control enables you to use a Web server control to access data that is located in a relational data base. This can include Microsoft SQL Server and Oracle databases, as well as OLE DB and ODBC data sources. It is used when we are working with SQL database.

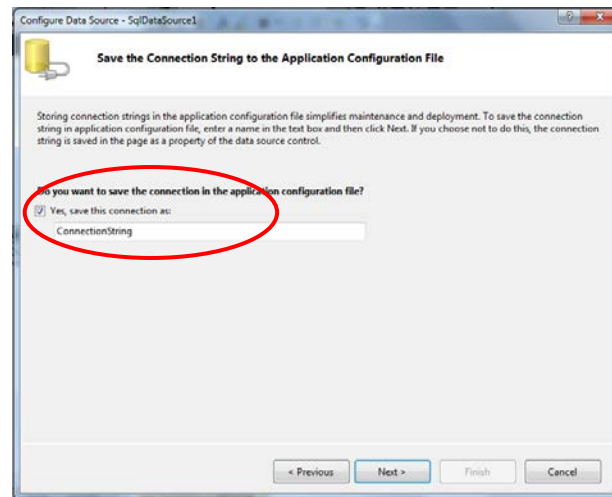


SQLDataSource is found in Data Tab of Toobox in Visual Studio.

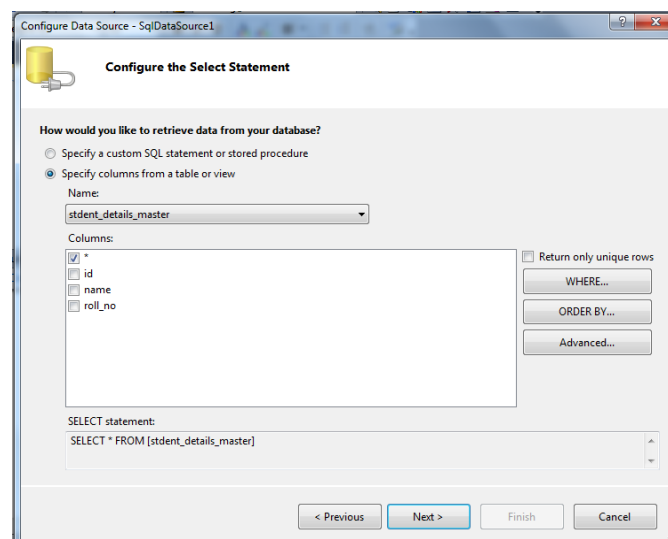
SQLDataSource enables the developers to bind the data to the controls from database. SQLDataSource can be used to fill ASP controls such as GridView, ListBox, DataList, DropDownList, RadioButtonList, etc. to display and manipulate data on an ASP.NET Web page, using little or no code. All the controls that contain data as a collection can be used with SQLDataSource.

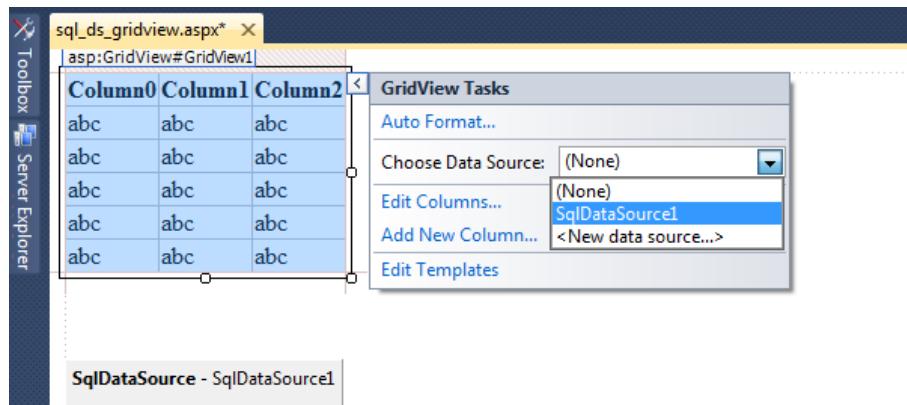


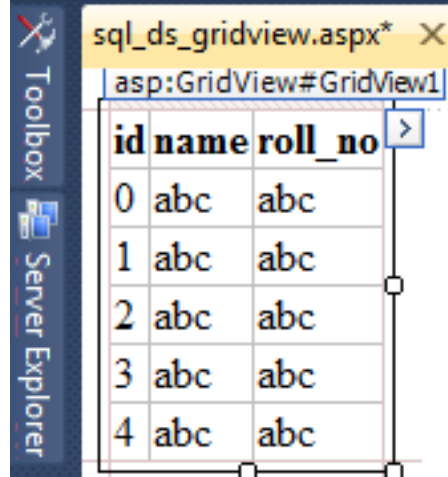
**Example: Filling GridView using SQLDataSource.****Design View Before Configuring SQLDataSource:****Configuring SQLDataSource:****Step 1 : Click on Side Button and click on Configure Data Source****Step 2 : Select The Data Base [we have created in built database already]****Step 3 : Click on Next Button**

**Step 4 : Save Connection with name****Step 5 : Configure Select Statement**

**You either can write query by selecting first option or can configure it by using second option**



**Step 5: Click Finish.****Assigning configured DataSource to GridView:****After DataSource is assigned:**



id	name	roll_no
0	abc	abc
1	abc	abc
2	abc	abc
3	abc	abc
4	abc	abc

**Design Code:** sql\_ds\_gridview.aspx

```

<%@ Page Language="VB" AutoEventWireup="false"
CodeFile=" sql_ds_gridview.aspx.vb" Inherits=" sql_ds_gridview" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>SQL Data Source-GridView </title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False" DataKeyNames="id"
DataSourceID="SqlDataSource1" EnableModelValidation="True">
            <Columns>
                <asp:BoundField DataField="id" HeaderText="id"
InsertVisible="False" ReadOnly="True" SortExpression="id" />
                <asp:BoundField DataField="name" HeaderText="name"
SortExpression="name" />
                <asp:BoundField DataField="roll_no" HeaderText="roll_no"

```

```
SortExpression="roll_no" />

</Columns>
</asp:GridView>
<br /><br />
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%%$ ConnectionStrings:ConnectionString %>"
SelectCommand="SELECT * FROM [stdent_details_master]">
</asp:SqlDataSource>
</form>
</body>
</html>
```

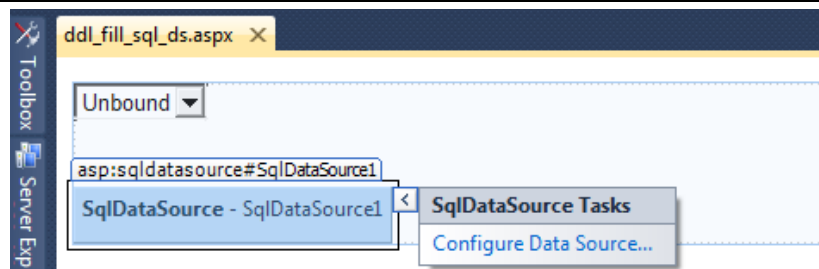
**VB Code :** sql\_ds\_gridview.aspx.vb

**No Code**

**Output View :**

**Example: Filling DropDownList using SqlDataSource.**

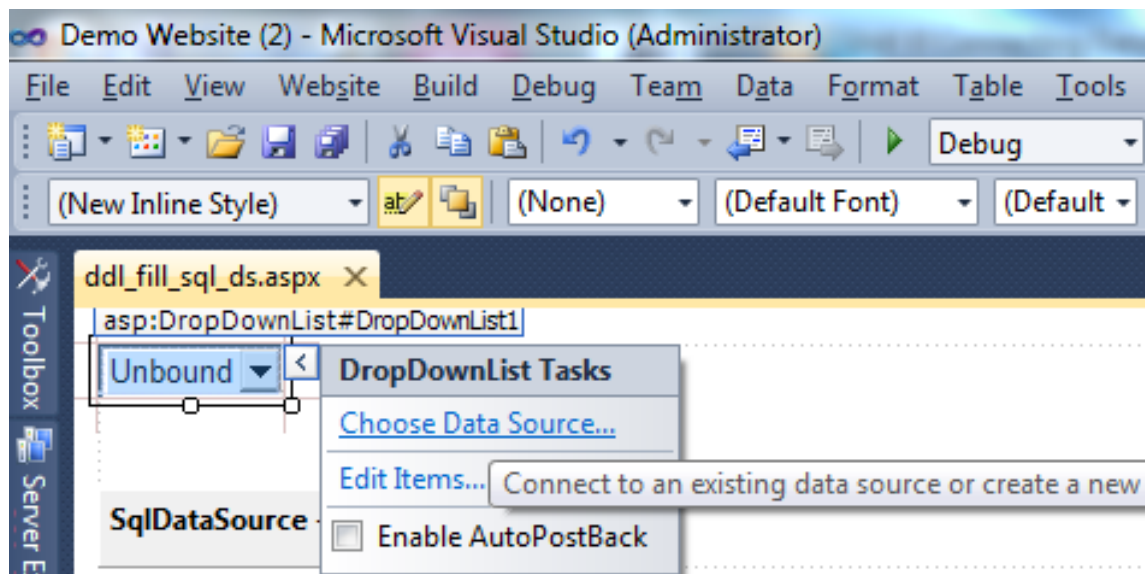
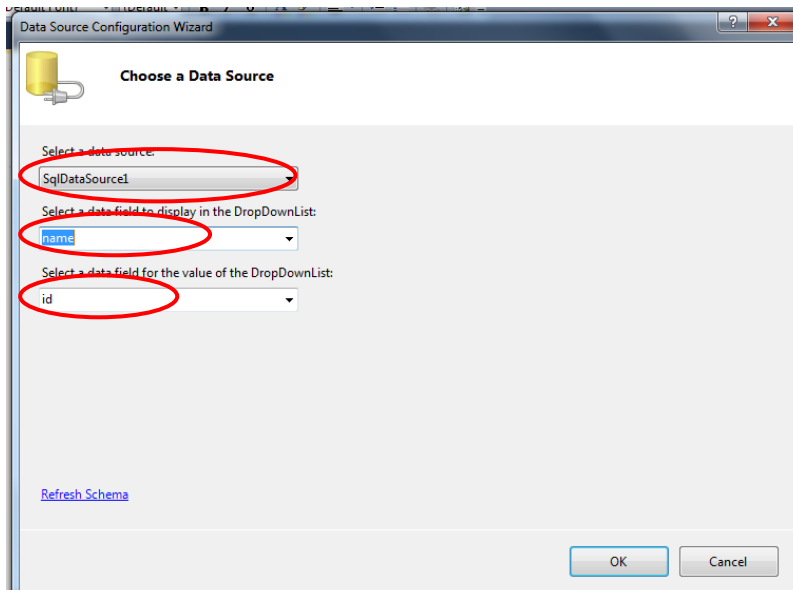
**Design View:**



**Configuring SqlDataSource:**

**Steps for configuring SqlDataSource is same as shown in above example**

**Assigning configured DataSource to DropDownList:**

**Step 1: Choose DataSource****Step 2: Choose DataSource and Data field and Value field.**

**Data field has to be that column which is to be displayed in control.**

**Value field has to be value for that control. it can be primary key or**

**anything.**

**Design Code:** ddl\_fill\_sql\_ds.aspx

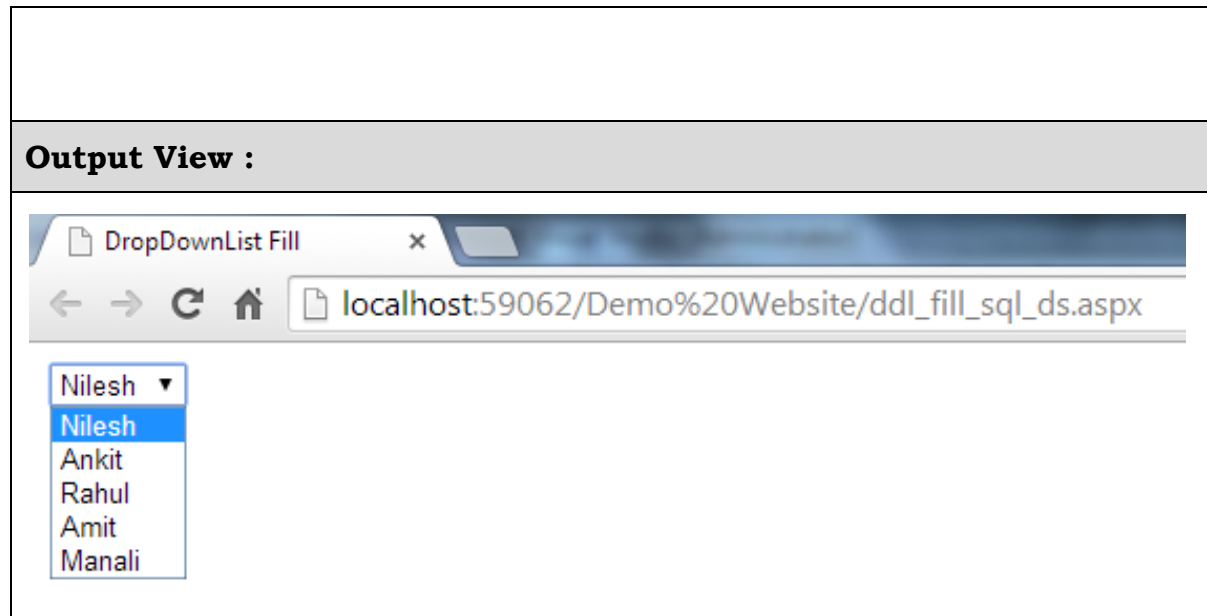
```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile=" ddl_fill_sql_ds.aspx.vb" Inherits="ddl_fill_sql_ds " %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>DropDownList Fill </title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:DropDownList ID="DropDownList1" runat="server"
DataSourceID="SqlDataSource1" DataTextField="name"
DataValueField="id"> </asp:DropDownList>
        <br /><br />

        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%%$ ConnectionStrings:ConnectionString %>"
SelectCommand="SELECT * FROM [stdnt_details_master]">
        </asp:SqlDataSource>

    </form>
</body>
</html>
```

**VB Code :** sql\_ds\_gridview.aspx.vb

**No Code**

**ObjectDataSource:**

The ASP.NET ObjectDataSource control represents a middle-tier object with data retrieval and update capabilities. The ObjectDataSource control acts as a data interface for data-bound controls such as the GridView, FormView, or DetailsView controls. You can use these controls to display and edit data from a middle-tier business object on an ASP.NET Web page.

Most ASP.NET data source controls, such as the SQLDataSource, are used in a two-tier application architecture where the presentation layer (the ASP.NET Web page) communicates directly with the data tier (the database, an XML file, and so on). However, a common application design practice is to separate the presentation layer from business logic and encapsulate the business logic in business objects. These business objects form a layer between the presentation layer and the data tier, resulting in three-tier application architecture.

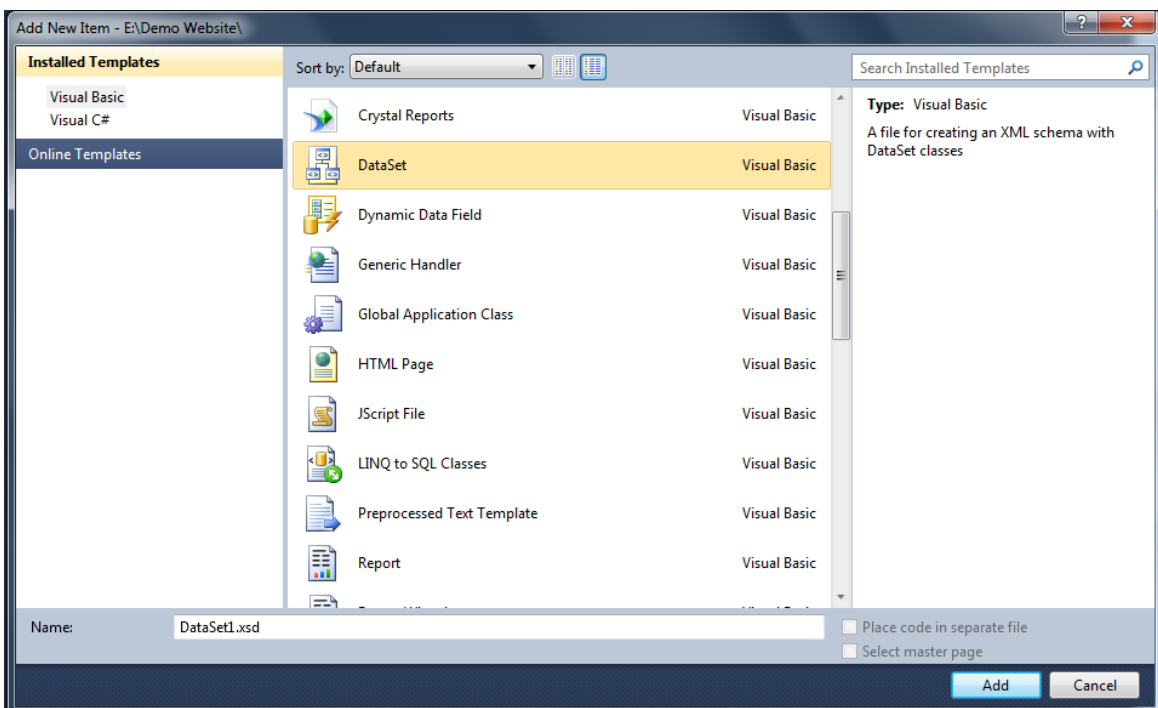
The ObjectDataSource control supports three-tier architecture by providing a way for you to bind data controls on the page to a middle-tier business object. The ObjectDataSource works with a middle-tier business object to select, insert, update, delete, page, sort, cache, and filter data declaratively without extensive code.



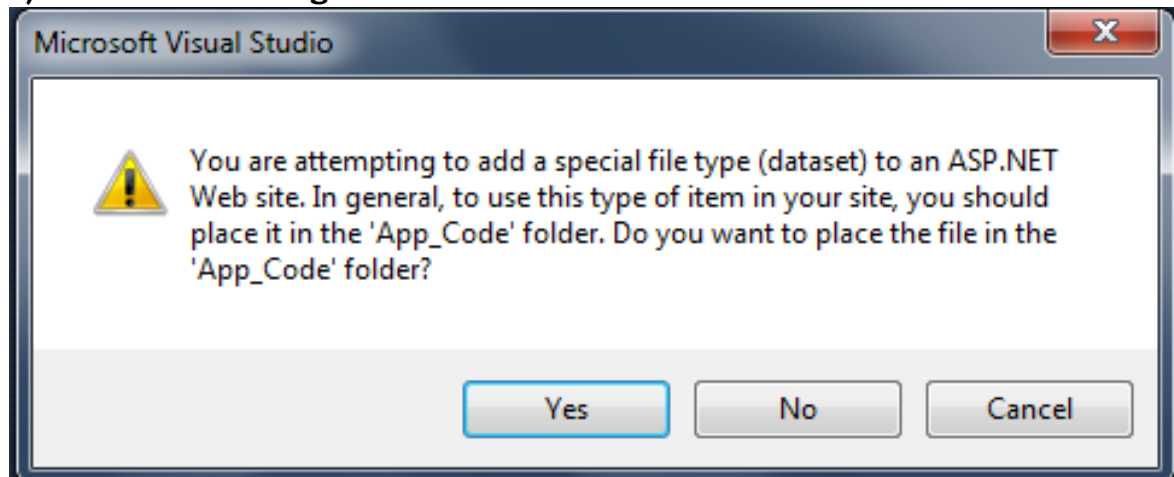
The ObjectDataSource control uses reflection to call methods of a business object to select, update, insert, and delete data. You set the ObjectDataSource control's TypeName property to specify the name of the class to use as a source object. For details on how to create a source data object to be used with the ObjectDataSource control, see Creating an ObjectDataSource Control Source Object.

**Example: Filling GridView using ObjectDataSource.****Step 1: Create business logic in different layer i.e. Dataset (.xsd) file**

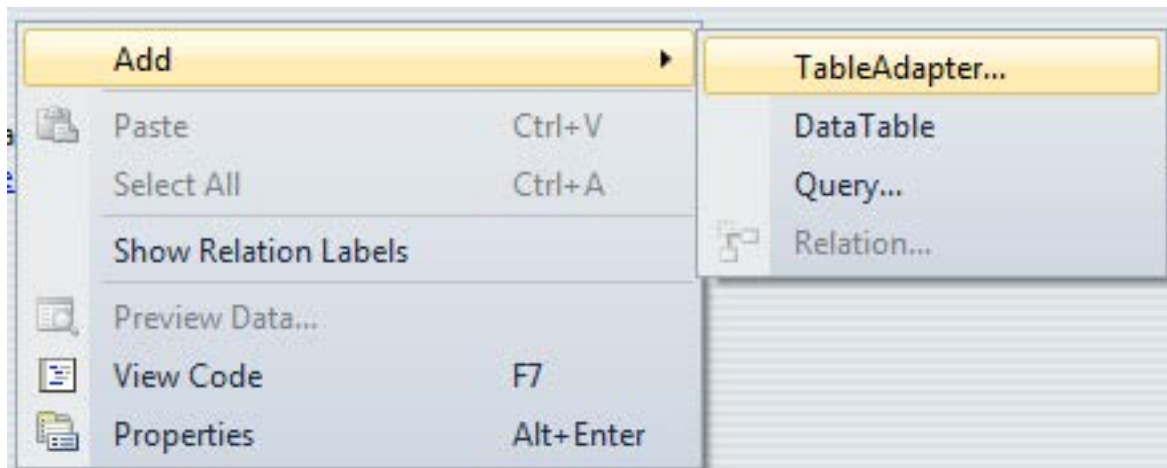
- a) Right click on root directory and click on Add New Item
- b) select dataset file (.xsd) and name it properly and click Add.

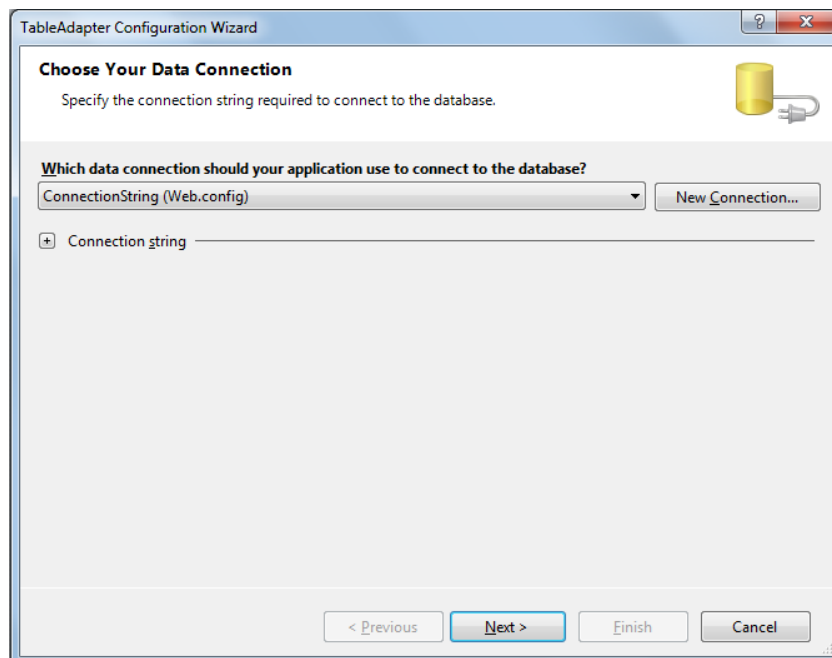
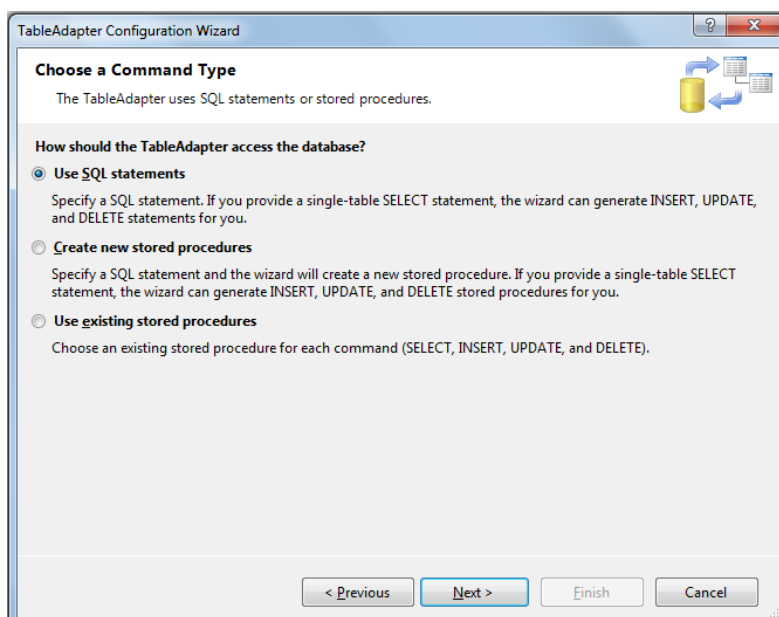


**c) Click Yes in dialogue box.**



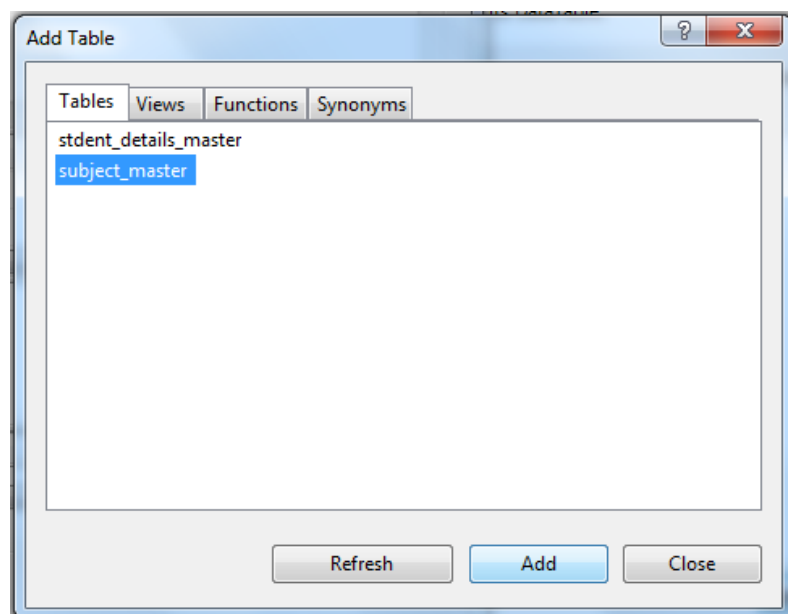
**d) Right Click in file and Create TableAdapter for DataSet File**



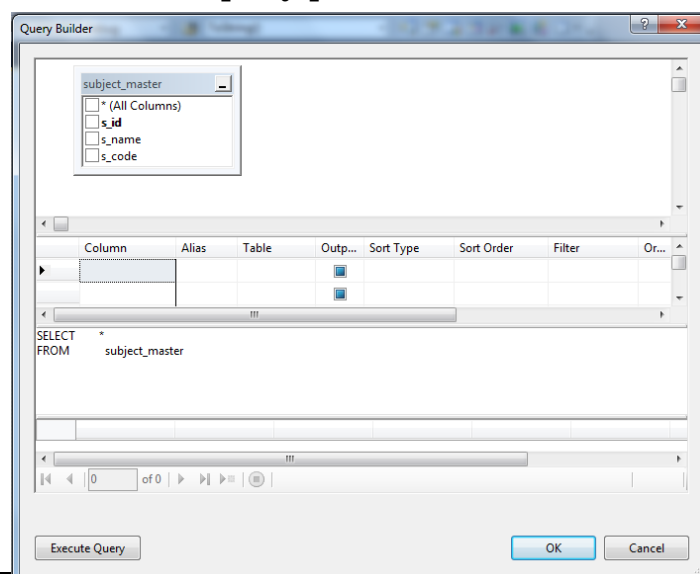
**e) Choose the data connection and click Next.****f) Choose Command Type. [Can be SQL statements or Stored Procedures]**

**g) Enter SQL Statement as we have selected SQL above. Query Builder can be used as well by clicking on it.**

**h) Select data table for retrieval of data and click Add.**



**i) Write SQL statement in query pane and click OK.**



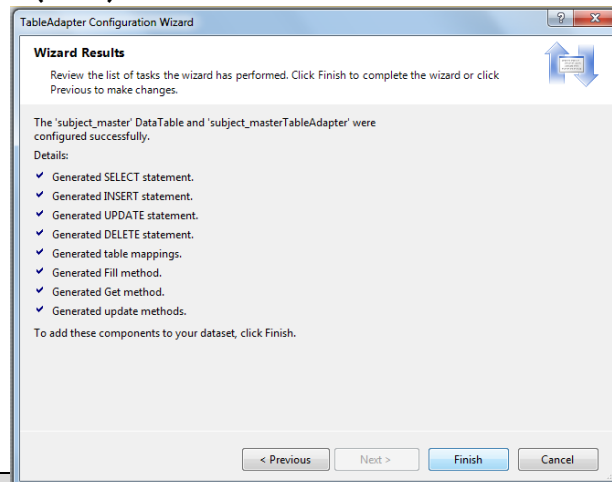
**j) Complete wizard by clicking Next every time and then click Finish.**

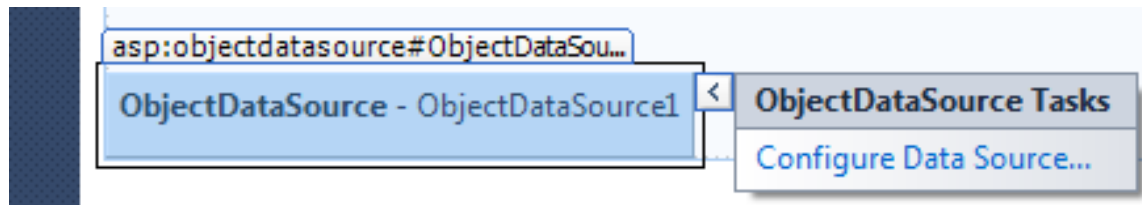
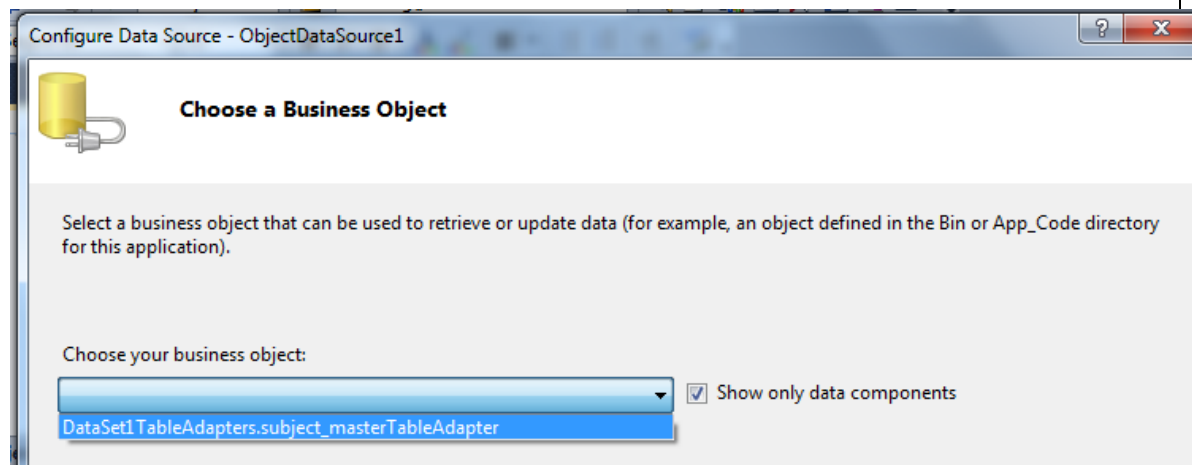
The screenshot shows the 'Enter a SQL Statement' step of the TableAdapter Configuration Wizard. The title bar reads 'TableAdapter Configuration Wizard'. The main heading is 'Enter a SQL Statement'. Below it, a subtitle states: 'The TableAdapter uses the data returned by this statement to fill its DataTable.' A blue double-headed arrow icon is in the top right corner. The instruction says: 'Type your SQL statement or use the Query Builder to construct it. What data should be loaded into the table? What data should be loaded into the table?'. A text box contains the SQL statement: 

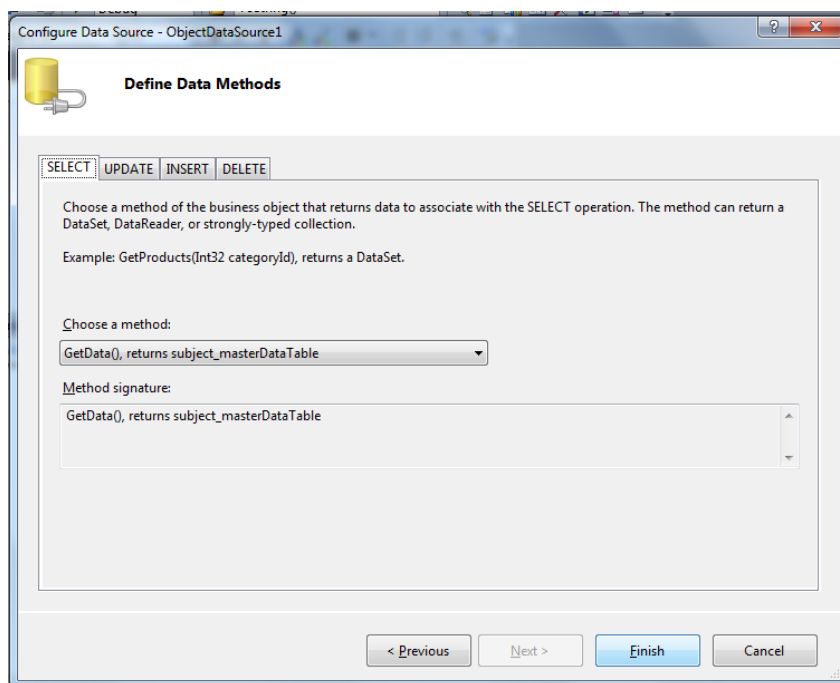
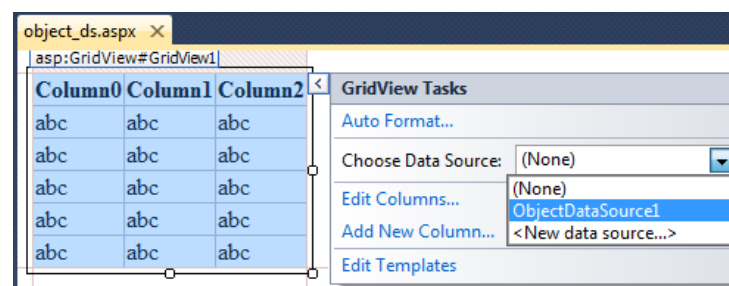
```
SELECT s.id, s.name, s.code
FROM subject-master
```

. Below the text box are two buttons: 'Advanced Options...' and 'Query Builder...'. At the bottom are four navigation buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

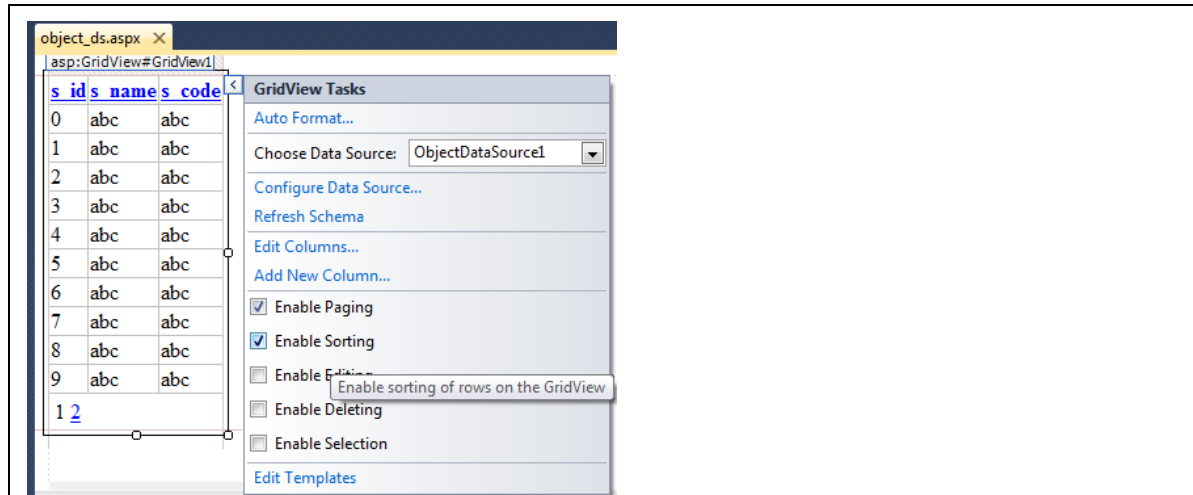
The screenshot shows the 'Choose Methods to Generate' step of the TableAdapter Configuration Wizard. The title bar reads 'TableAdapter Configuration Wizard'. The main heading is 'Choose Methods to Generate'. Below it, a subtitle states: 'The TableAdapter methods load and save data between your application and the database.' A blue double-headed arrow icon is in the top right corner. The instruction says: 'Which methods do you want to add to the TableAdapter?'. There are three checked options:   
1. ☒ **Fill a DataTable**: 'Creates a method that takes a DataTable or DataSet as a parameter and executes the SQL statement or SELECT stored procedure entered on the previous page.' The 'Method name' field contains 'Fill'.   
2. ☒ **Return a DataTable**: 'Creates a method that returns a new DataTable filled with the results of the SQL statement or SELECT stored procedure entered on the previous page.' The 'Method name' field contains 'GetData'.   
3. ☒ **Create methods to send updates directly to the database (GenerateDBDirectMethods)**: 'Creates Insert, Update, and Delete methods that can be called to send individual row changes directly to the database.' At the bottom are four navigation buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

**k) Save DataSet (.xsd) File.****Configuring ObjectDataSource:**

**Step 1 : Click on Side Button and click on Configure Data Source****Step 2: Choose Business Logic [.xsd file created earlier] and click Next.**

**Step 3: Click on Finish.****Assigning configured DataSource to GridView:****After DataSource is assigned: Enabling Paging and Sorting options.**



**Design Code:** object\_ds.aspx

```

<%@ Page Language="VB" AutoEventWireup="false"
CodeFile=" object_ds.aspx.vb" Inherits="object_ds " %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Object DataSource Demo </title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False" DataKeyNames="s_id"
DataSourceID="ObjectDataSource1" EnableModelValidation="True">

            <Columns>
                <asp:BoundField DataField="s_id" HeaderText="s_id"
InsertVisible="False" ReadOnly="True" SortExpression="s_id" />
                <asp:BoundField DataField="s_name" HeaderText="s_name"
SortExpression="s_name" />
                <asp:BoundField DataField="s_code" HeaderText="s_code"

```

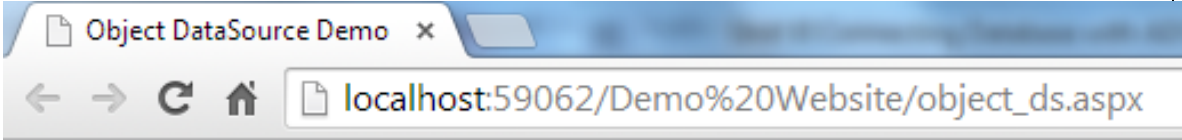
```
SortExpression="s_code" />
</Columns>
</asp:GridView>
<br /> <br />
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
DeleteMethod="Delete" InsertMethod="Insert"
OldValuesParameterFormatString="original_{0}" SelectMethod="GetData"
TypeName="DataSet1TableAdapters.subject_masterTableAdapter"
UpdateMethod="Update">

    <DeleteParameters>
        <asp:Parameter Name="Original_s_id" Type="Int32" />
    </DeleteParameters>
    <InsertParameters>
        <asp:Parameter Name="s_name" Type="String" />
        <asp:Parameter Name="s_code" Type="String" />
    </InsertParameters>
    <UpdateParameters>
        <asp:Parameter Name="s_name" Type="String" />
        <asp:Parameter Name="s_code" Type="String" />
        <asp:Parameter Name="Original_s_id" Type="Int32" />
    </UpdateParameters>
</asp:ObjectDataSource>
</form>
</body>
</html>
```

**VB Code :** object\_ds.aspx.vb

**No Code**

**Output View :**



The screenshot shows a web browser window with the title 'Object DataSource Demo'. The address bar displays 'localhost:59062/Demo%20Website/object\_ds.aspx'. Below the browser window, a table is displayed with three columns: 's\_id', 's\_name', and 's\_code'. The table contains five rows of data.

<u>s_id</u>	<u>s_name</u>	<u>s_code</u>
1	Project	3351605
2	Web Prog Using ASP.Net & VB.Net	3351603
3	EIS	3351602
4	JAVA Programming	3350703
5	ICN	3351601

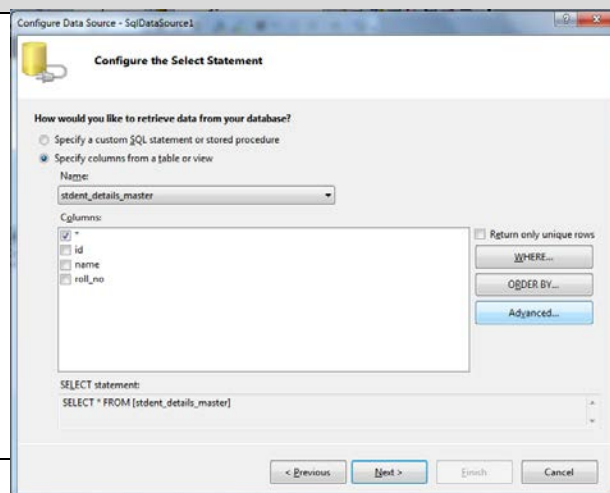
## 6.3 SQL DataSource

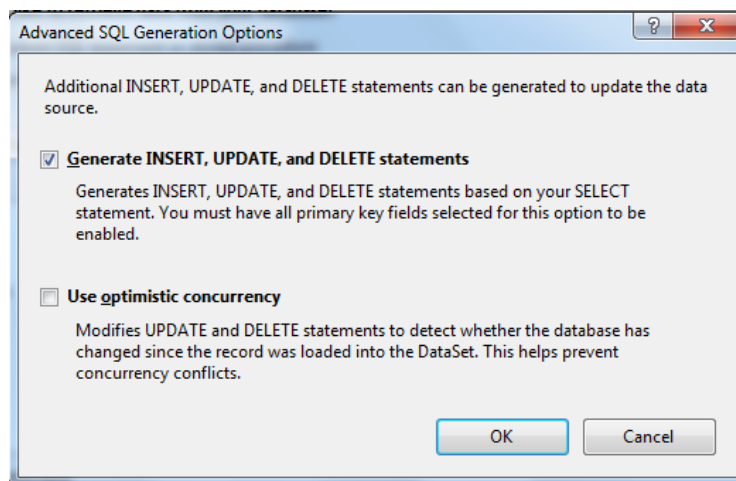
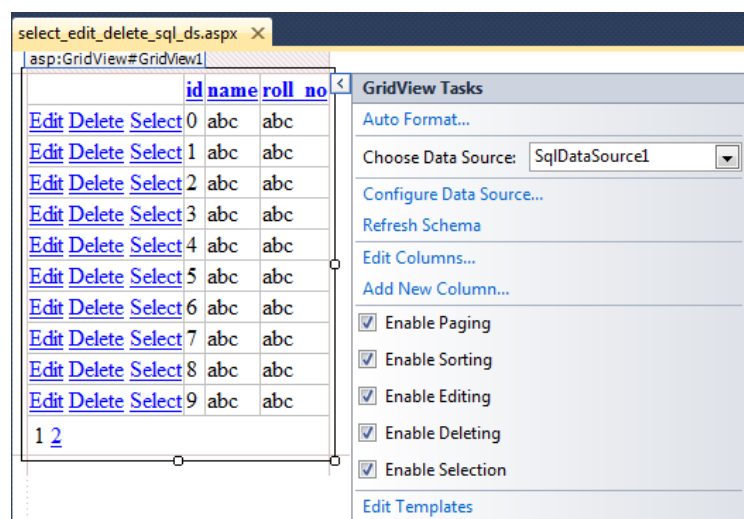
### Selecting, Editing, Deleting Records

**Example: Selecting, Deleting, Updating Data using SQL DataSource in GridView.**

**Step 1: Configuration steps till query is same as illustrated in earlier example**

**Step 2: Click on Advance Option while configuring SQL statement.**



**Step 3: Check first option to generate SELECT, INSERT, DELETE statements.****Step 4: Click Next and then click Finish.****Assigning DataSource to GridView and enabling all options to edit, delete data.**

**Design Code:** select\_edit\_delete\_sql\_ds.aspx

```

<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="select_edit_delete_sql_ds.aspx.vb"
Inherits=" select_edit_delete_sql_ds" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Manipulating Data - SQL DataSource</title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:GridView ID="GridView1" runat="server" AllowPaging="True"
      AllowSorting="True" AutoGenerateColumns="False" DataKeyNames="id"
      DataSourceID="SqlDataSource1" EnableModelValidation="True">

      <Columns>
        <asp:CommandField ShowDeleteButton="True" ShowEditButton="True"
          ShowSelectButton="True" />

        <asp:BoundField DataField="id" HeaderText="id" InsertVisible="False"
          ReadOnly="True" SortExpression="id" />

        <asp:BoundField DataField="name" HeaderText="name"
          SortExpression="name" />

        <asp:BoundField DataField="roll_no" HeaderText="roll_no"
          SortExpression="roll_no" />
      </Columns>
    </asp:GridView>
    <br /> <br />
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%%$ ConnectionStrings:ConnectionString %>"
      DeleteCommand="DELETE FROM [stdnt_details_master] WHERE [id] =
        @id" InsertCommand="INSERT INTO [stdnt_details_master] ([name],

```

```

[roll_no]) VALUES (@name, @roll_no)" SelectCommand="SELECT * FROM
[stdnt_details_master]"

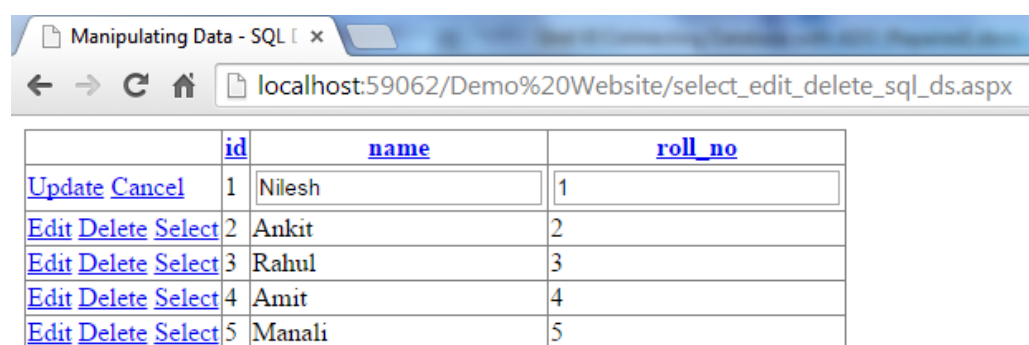
UpdateCommand="UPDATE [stdnt_details_master] SET [name] =
@name, [roll_no] = @roll_no WHERE [id] = @id">
    <DeleteParameters>
        <asp:Parameter Name="id" Type="Int32" />
    </DeleteParameters>
    <InsertParameters>
        <asp:Parameter Name="name" Type="String" />
        <asp:Parameter Name="roll_no" Type="String" />
    </InsertParameters>
    <UpdateParameters>
        <asp:Parameter Name="name" Type="String" />
        <asp:Parameter Name="roll_no" Type="String" />
        <asp:Parameter Name="id" Type="Int32" />
    </UpdateParameters>
</asp:SqlDataSource>
</form>
</body>
</html>

```

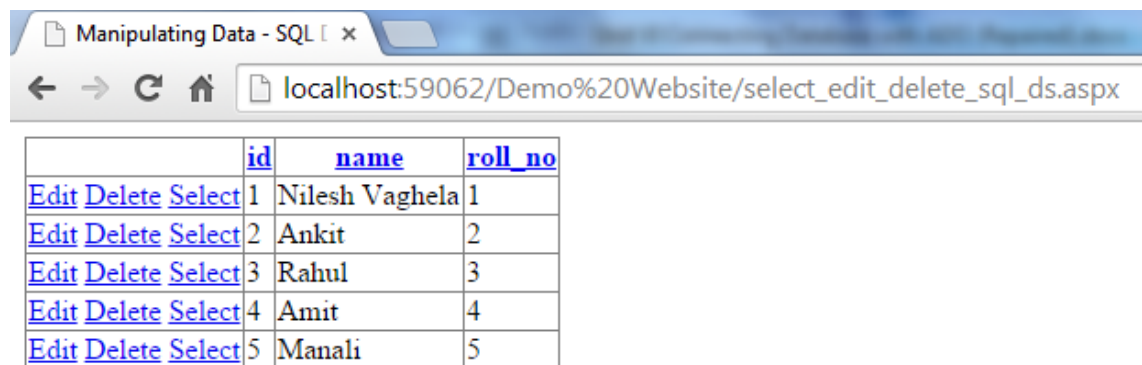
**VB Code :** object\_ds.aspx.vb

**No Code**

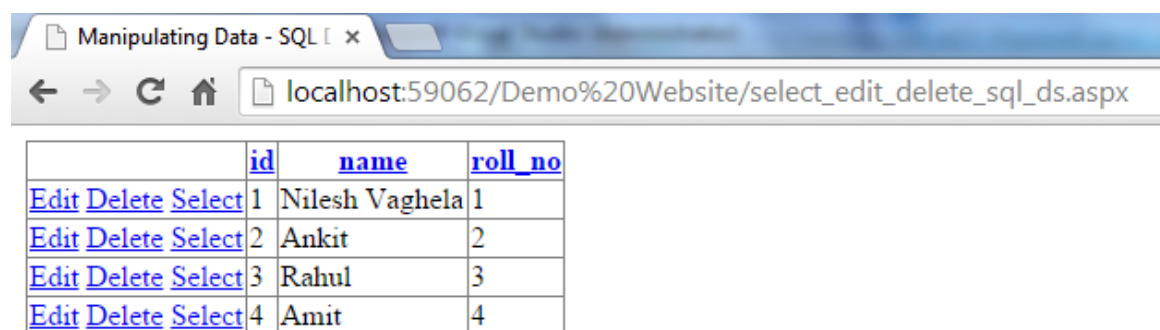
**Output View : Editing Data**



	id	name	roll_no
<a href="#">Update</a> <a href="#">Cancel</a>	1	Nilesh	1
<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Select</a>	2	Ankit	2
<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Select</a>	3	Rahul	3
<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Select</a>	4	Amit	4
<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Select</a>	5	Manali	5

**Output View : Updated Data after editing it**

	<u>id</u>	<u>name</u>	<u>roll_no</u>
<a href="#">Edit Delete Select</a>	1	Nilesh Vaghela	1
<a href="#">Edit Delete Select</a>	2	Ankit	2
<a href="#">Edit Delete Select</a>	3	Rahul	3
<a href="#">Edit Delete Select</a>	4	Amit	4
<a href="#">Edit Delete Select</a>	5	Manali	5

**Output View : After Deleting Data**

	<u>id</u>	<u>name</u>	<u>roll_no</u>
<a href="#">Edit Delete Select</a>	1	Nilesh Vaghela	1
<a href="#">Edit Delete Select</a>	2	Ankit	2
<a href="#">Edit Delete Select</a>	3	Rahul	3
<a href="#">Edit Delete Select</a>	4	Amit	4

**Exercise**

1. What is ADO.NET?
2. What is disconnected environment?
3. Explain Connection object.
4. Explain command object with its various methods.
5. Compare DataReader & DataSet.
6. Explain DataAdapter with suitable example.
7. Explain DataTable & DataRow with example.
8. What is DataView?
9. Explain SqlDataSource.
10. Explain ObjectDataSource.
11. Compare SqlDataSource & ObjectDataSource.

\*\*\*\*\*END OF CHAPTER\*\*\*\*\*



\$ END OF BOOK\$

\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$ END OF BOOK\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$