UNIT-IV

Databases:

- A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.
- Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data.

Configuring Database:

- To connect with asp.net web form with database we need a class named SqlConnection and in order to use this class a namespace must be imported i.e System.Data.SqlClient;
  - SqlConnection class accepts several parameter that is called connection string.Parameters are:
    1. Data source: It defines server name
    2. Initial Catalog:name of the database.
    3. Uid: userid of sql server
    4. Pswd: password to connect with sql server
    5. Integrated security: used for windows authentication

For sql server authentication:

```
string cs = "data source=DESKTOP-16RE47P\\SQLEXPRESS; initial catalog=college1; uid=sa;pwd=deepa123";
```

For Windows authentication:

```
string cs = "data source=DESKTOP-16RE47P;initial catalog=college; integrated security=true";
```

Understanding SQL basics:

- Create database college
- create table student
  (roll int,name varchar(20),course varchar(10))
- insert into student values(101,'deepa','bca')
- select * from student
- delete from student where roll=1
- update student set name=’nisha’ where roll=10

ADO.NET:

- ADO.NET stands for ActiveX Data Object.
- It is a module of .Net Framework which is used to establish connection between application and data sources.
- Data sources can be such as SQL Server and XML.
- ADO.NET consists of classes that can be used to connect, retrieve, insert and delete data.
- All the ADO.NET classes are located into **System.Data.dll** and integrated with XML classes located into **System.Xml.dll.**
- ADO.NET has two main components that are used for accessing and manipulating data are the .NET Framework data provider and the DataSet.

Data Provider Model:

Data provider is used to connect to the database, execute commands and retrieve the record. It is lightweight component with better performance. It also allows us to place the data into DataSet to use it further in our application.

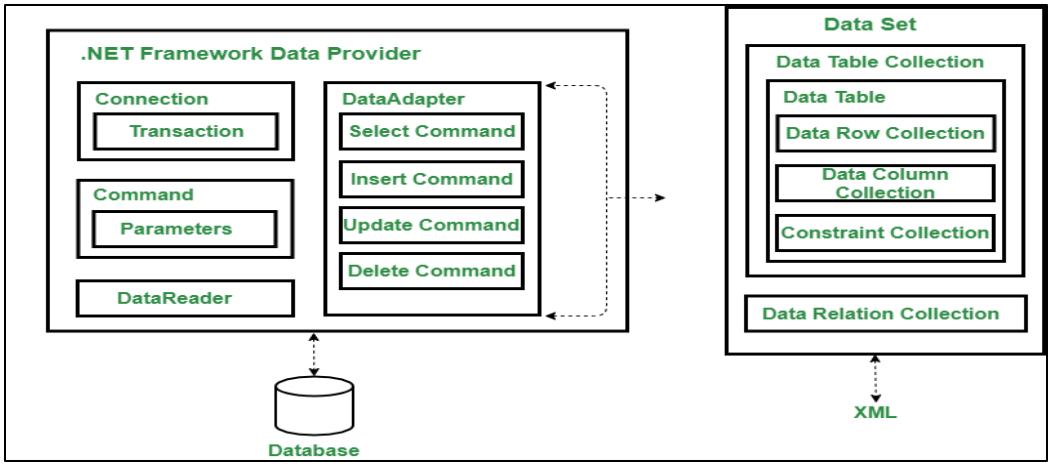
The .NET Framework provides the following data providers that we can use in our application

.NET Framework data providers	Description
SQL Server	It provides data access for Microsoft SQL Server. It requires the **System.Data.SqlClient** namespace.
OLEDB	It is used to connect with OLE DB. It requires the **System.Data.OleDb** namespace.
ODBC	It is used to connect to data sources by using ODBC. It requires the **System.Data.Odbc** namespace.
Oracle	It is used for Oracle data sources. It uses the **System.Data.OracleClient** namespace.

Core Objects of data providers:

Following are the core object of Data Providers.

Object	Description
Connection	It is used to establish a connection to a specific data source.
Command	It is used to execute queries to perform database operations.
DataReader	It is used to read data from data source. The DbDataReader is a base class for all DataReader objects.
DataAdapter	It populates a DataSet and resolves updates with the data source. The base class for all DataAdapter objects is the DbDataAdapter class.



Namespaces for SQL Server Data Access:

1. **System.Data.SqlClient:**

- It contains SqlConnection and SqlCommand,SqlDataReader & SqlDataAdapter class.

2. **System.Data:**

- It contains DataSet Class.

Important Classes used while dealing with database

1. SqlConnection:

- It is used to establish an open connection to the SQL Server database. It is a sealed class so that cannot be inherited.
- SqlConnection class uses SqlDataAdapter and SqlCommand classes together to increase performance when connecting to a Microsoft SQL Server database.
- Connection does not close explicitly even it goes out of scope. Therefore, you must explicitly close the connection by calling Close() method.
- To use SqlConnection Class we have use namespace Using System.Data.SqlClient;

SqlConnection Methods

Method	Description
Close()	It is used to close the connection to the database.
Open()	It is used to open a database connection.

2. SqlCommand Class:

- This class is used to store and execute SQL statement for SQL Server database. It is a sealed class so that cannot be inherited.
- SqlCommand Class accepts 2 argument first the query string and second connection object or connection string.

Methods:

Method	Description
Cancel()	It tries to cancel the execution of a SqlCommand.
ExecuteReader()	It is used to send the CommandText to the Connection and builds a SqlDataReader.
ExecuteXmlReader()	It is used to send the CommandText to the Connection and builds an XmlReader object.
ExecuteScalar()	It executes the query and returns the first column of the first row in the result set. Additional columns or rows are ignored.
ExecuteNonQuery():	This method executes a Transact-SQL statement against the connection and returns the number of rows affected

3. SqlDataReader class:

- SqlDataReader class in C# is used to read data from the SQL Server database in the most efficient manner.
- It reads data in the forward-only direction. It means once it reads a record, it will then read the next record; there is no way to go back and read the previous record.

a. **SqlDataReader is Connection-Oriented.** It means it requires an open or active connection to the data source while reading the data. The data is available as long as the connection with the database exists.
b. **SqlDataReader is Read-Only.** It means it is also not possible to change the data using SqlDataReader. You also need to open and close the connection explicitly.

SqlDataReader Class Properties in C#:

The SqlDataReader class provides the following properties.

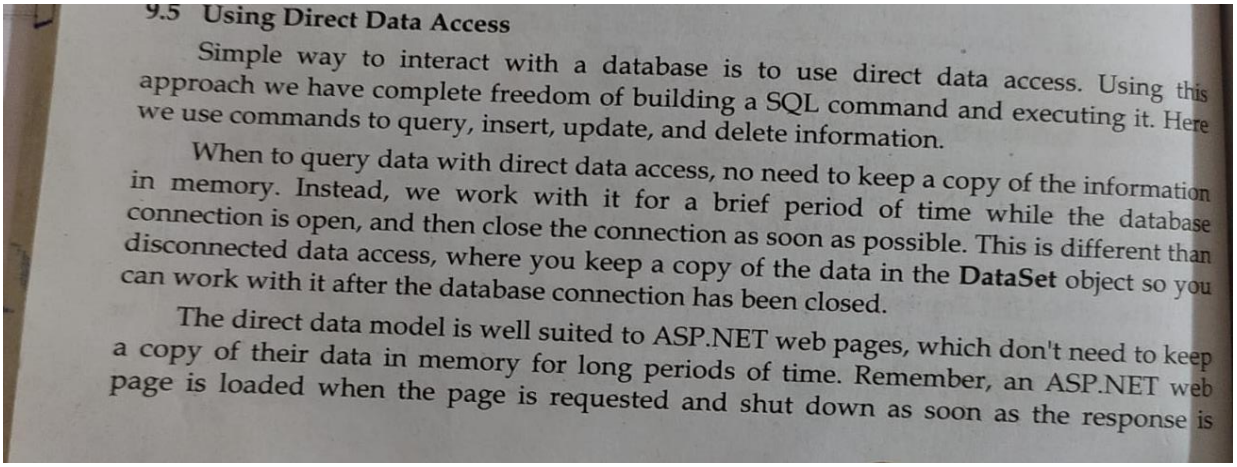
1. **Connection**: It gets the System.Data.SqlClient.SqlConnection associated with the System.Data.SqlClient.SqlDataReader.

2. **FieldCount**: It gets the number of columns in the current row.
3. **HasRows**: It gets a value that indicates whether the System.Data.SqlClient.SqlDataReader contains one or more rows.
4. **Item[String]**: It gets the specified column's value in its native format, given the column name.

ADO.NET SqlDataReader Class Methods in C#:
The SqlDataReader class provides the following methods.

1. **Close():** It closes the SqlDataReader object.
2. **Read():** It Advances the System.Data.SqlClient.SqlDataReader to the next record and returns true if there are more rows; otherwise, it is false.

Using Direct Access:

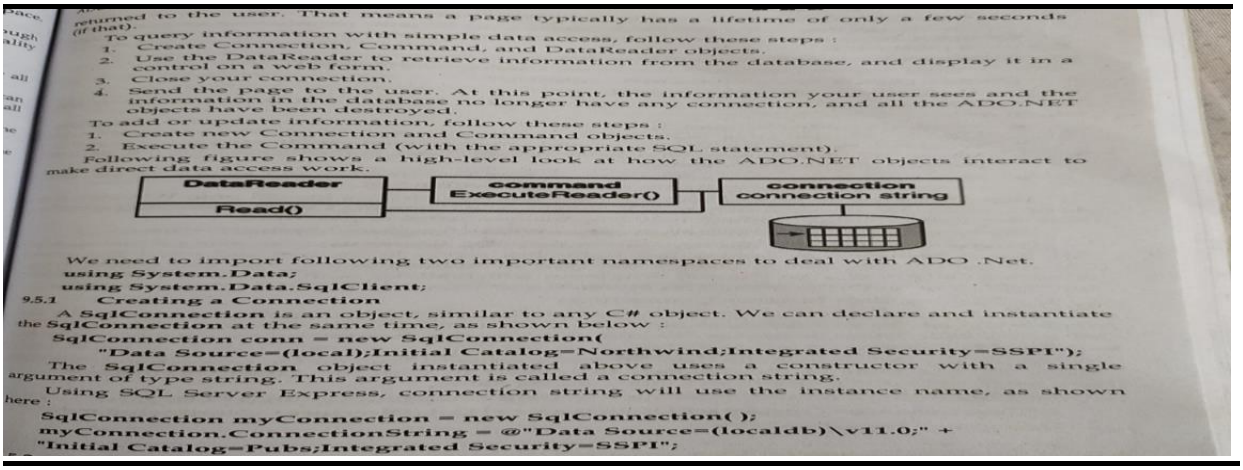


9.5 Using Direct Data Access

Simple way to interact with a database is to use direct data access. Using this approach we have complete freedom of building a SQL command and executing it. Here we use commands to query, insert, update, and delete information.

When to query data with direct data access, no need to keep a copy of the information in memory. Instead, we work with it for a brief period of time while the database connection is open, and then close the connection as soon as possible. This is different than disconnected data access, where you keep a copy of the data in the **DataSet** object so you can work with it after the database connection has been closed.

The direct data model is well suited to ASP.NET web pages, which don't need to keep a copy of their data in memory for long periods of time. Remember, an ASP.NET web page is loaded when the page is requested and shut down as soon as the response is



returned to the user. That means a page typically has a lifetime of only a few seconds (if that).

To query information with simple data access, follow these steps :

1. Create Connection, Command, and DataReader objects.
2. Use the DataReader to retrieve information from the database, and display it in a control on a web form.
3. Close your connection.
4. Send the page to the user. At this point, the information your user sees and the information in the database no longer have any connection, and all the ADO.NET objects have been destroyed.

To add or update information, follow these steps :

1. Create new Connection and Command objects.
2. Execute the Command (with the appropriate SQL statement).

Following figure shows a high-level look at how the ADO.NET objects interact to make direct data access work.

We need to import following two important namespaces to deal with ADO .Net.

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

9.5.1 **Creating a Connection**

A **SqlConnection** is an object, similar to any C# object. We can declare and instantiate the **SqlConnection** at the same time, as shown below :

```
SqlConnection conn = new SqlConnection(
    "Data Source=(local);Initial Catalog=Northwind;Integrated Security=SSPI");
```

The **SqlConnection** object instantiated above uses a constructor with a single argument of type string. This argument is called a connection string.

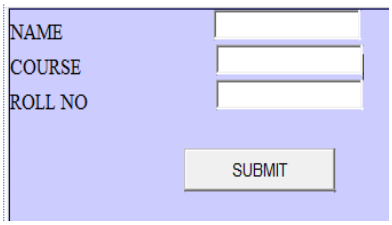
Using SQL Server Express, connection string will use the instance name, as shown here :

```
SqlConnection myConnection = new SqlConnection();
myConnection.ConnectionString = @"Data Source=(localdb)\v11.0;" +
"Initial Catalog=Pubs;Integrated Security=SSPI";
```

Program to insert a record

- We have taken 3 textboxes and on submit button click all the 3 data should get inserted into student table.
- To execute insert query "ExecuteNonQuery()" method has been used.

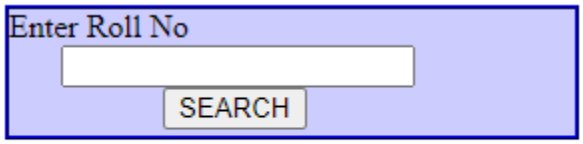
```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace database1
{
   public partial class DB1 : System.Web.UI.Page
   {
      SqlConnection conn;
      SqlCommand cmd;
      protected void Page_Load(object sender, EventArgs e)
      {
         string cs = "data source=DESKTOP-16RE47P\\SQLEXPRESS;initial
catalog=college1;uid=sa;pwd=deepa123";
         conn = new SqlConnection(cs);
      }
      protected void insert_Click1(object sender, EventArgs e)
      {
         string q = "insert into student values('" + TextBox4.Text + "','" + TextBox1.Text +
"','" + TextBox2.Text + "')";

         cmd = new SqlCommand(q, conn);

         conn.Open();

         cmd.ExecuteNonQuery();

         Response.Write("data stored");

      }
```



Search A Record From Table & Display It

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace database1
{
   public partial class DB1 : System.Web.UI.Page
   {
      SqlConnection conn;
      SqlCommand cmd;
      protected void Page_Load(object sender, EventArgs e)
      {
         string cs = "data source=DESKTOP-16RE47P\\SQLEXPRESS; initial
catalog=college1; uid=sa;pwd=deepa123";
         conn = new SqlConnection(cs);
      }

      protected void search_Click1(object sender, EventArgs e)
      {
         String q1 = "select * from student where roll='" + TextBox3.Text + "'";
         conn.Open();
         cmd = new SqlCommand(q1, conn);
         SqlDataReader dr = cmd.ExecuteReader();
         if (dr.HasRows)
         {
            dr.Read();

            Response.Write(dr["name"] + " " + dr["course"]);
```

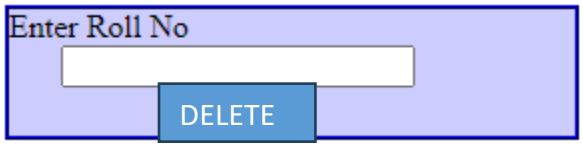
```
        }
        else
        {
            Response.Write("data not available");
            conn.Close();
    } } }
```



Delete A Record:

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace database1
{
    public partial class DB1 : System.Web.UI.Page
    {
        SqlConnection conn;
        SqlCommand cmd;
        protected void Page_Load(object sender, EventArgs e)
        {
            string cs = "data source=DESKTOP-16RE47P\\SQLEXPRESS;initial
catalog=college1;uid=sa;pwd=deepa123";
            conn = new SqlConnection(cs);
        }
        protected void del_Click1(object sender, EventArgs e)
        {
            String q1 = "select * from student where roll='" + txt_roll.Text + "'";
            conn.Open();
            cmd = new SqlCommand(q1, conn);
            SqlDataReader dr = cmd.ExecuteReader();
```

```
        if (dr.HasRows)
        {
         string q = "delete from student where roll='"txt_roll.Text+"'";

        SqlCommand cmd1 = new SqlCommand(q, conn);

        cmd1.ExecuteNonQuery();

       Response.Write("data deleted");
        }
        else
        {
           Response.Write("data not found to delete");
        }

                con.close();

  }
```



Display data in gridview using connected architecture

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
namespace database1
{
    public partial class DB1 : System.Web.UI.Page
    {
        SqlConnection conn;
        SqlCommand cmd;
        protected void Page_Load(object sender, EventArgs e)
        {
            string cs = "data source=DESKTOP-
16RE47P\\SQLEXPRESS;initial catalog=college1;uid=sa;pwd=deepa123";
            conn = new SqlConnection(cs);

            string s = "select * from student";
            SqlCommand md = new SqlCommand(s, conn);
```

```
            conn.Open();
            SqlDataReader dr = md.ExecuteReader();
            DataTable dt = new DataTable();
            dt.Load(dr);
            GridView1.DataSource = dt;
            GridView1.DataBind();
            conn.Close();
        }
}
```

roll	name	course
101	deepa	bca
12	sahyog	MBA
54	deepa	MBA
32	poonam	bba
13	neha	mbbs
13	riya	bams
54	giya	ba
12	priya	bba
100	xyz	bca

Connected and Disconnected Architecture in ADO.NET

- The ADO.NET is one of the Microsoft data access technologies which is used to establish a connection between the .NET Application (Console, WCF, WPF, Windows, MVC, Web Form, etc.) and different data sources such as SQL Server, Oracle, MySQL, XML, etc.
- The ADO.NET framework access the data from data sources in two different ways.
- The models are Connection Oriented Data Access Architecture and Disconnected Data Access Architecture. In this article, I will explain both these architectures in detail with Examples.

Types of Architecture to Access the Data using ADO.NET:

The Architecture supported by ADO.NET for communicating with data sources is categorized into two models. They are as follows:

1. Connected Oriented Architecture
2. Disconnected Oriented Architecture

- So, ADO.NET supports both Connection-Oriented Architectures as well as Disconnection-Oriented Architecture.

❖ ADO.NET Connection-Oriented Data Access Architecture:

- In the case of Connection Oriented Data Access Architecture, always an open and active connection is required in between the .NET Application and the database.
- An example is Data Reader and when we are accessing the data from the database, the Data Reader object requires an active and open connection to access the data, If the

connection is closed then we cannot access the data from the database and in that case, we will get the runtime error.

- The Connection Oriented Data Access Architecture is always forward only. That means using this architecture mode, we can only access the data in the forward direction. Once we read a row, then it will move to the next data row and there is no chance to move back to the previous row.

❖ ADO.NET Disconnection-Oriented Data Access Architecture:

- In the case of Disconnection Oriented Data Access Architecture, always an open and active connection is not required in between the .NET Application and the database. In this architecture, Connectivity is required only to read the data from the database and to update the data within the database.
- An example is DataAdapter and DataSet or DataTable classes. Here, using the DataAdapter object and using an active and open connection, we can retrieve the data from the database and store the data in a DataSet or DataTable. The DataSets or DataTables are in-memory objects or you can say they store the data temporarily within .NET Application. Then whenever required in our .NET Application, we can fetch the data from the dataset or data table and process the data. Here, we can modify the data, we can insert new data, can delete the data from within the dataset or data tables. So, while processing the data within the .NET Application using DataSet or Datatable, we do not require an active and open connection.

Differences Between Connected-Oriented Architecture and Disconnected-Oriented Architecture:

Let us see the Differences Between ADO.NET Connected Oriented Architecture and Disconnected Oriented Architecture.

ADO.NET Connected Oriented Architecture:

1. It is connection-oriented data access architecture. It means always an active and open connection is required.
2. Using the DataReader object we can implement the Connected Oriented Architecture.
3. Connected Oriented Architecture gives a faster performance.
4. Connected Oriented Architecture can hold the data of a single table only.
5. You can access the data in a forward-only and read-only manner.
6. Using Data Reader, we cannot persist the data in the database.

ADO.NET Disconnected Oriented Architecture:

1. It is disconnection-oriented data access architecture. It means always an active and open connection is not required.
2. Using DataAdapter and DataSet or Datatable we can implement Dis-Connected Oriented Architecture.

3. Disconnected Oriented Architecture gives a lower performance.
4. Disconnected Oriented Architecture can hold the data of multiple tables using dataset and single table data using Datatable.
5. You can access the data in forward and backward directions and you can also modify the data.
6. Using Data Adapter, we can persist the DataSet or DataTable data into the database.

Program to display data in gridview using datatable for disconnected architecture

```
protected void Page_Load(object sender, EventArgs e)
    {
        string cs = "data source=DESKTOP-16RE47P\\SQLEXPRESS; initial catalog=
college1; uid=sa;pwd=deepa123";
        SqlConnection conn = new SqlConnection(cs);
        SqlDataAdapter da = new SqlDataAdapter("select * from student", conn);
         DataTable dt = new DataTable();
         da.Fill(dt);
         GridView1.DataSource = dt;
         GridView1.DataBind();
    }
```

roll	name	course
101	deepa	bca
12	sahyog	MBA
54	deepa	MBA
32	poonam	bba
13	neha	mbbs
13	riya	bams
54	giya	ba
12	priya	bba
100	xyz	bca

Program to display individual value from data table and also total number of records.

```
protected void Page_Load(object sender, EventArgs e)
    {
        string cs = "data source=DESKTOP-16RE47P\\SQLEXPRESS;initial
catalog=college1;uid=sa;pwd=deepa123";
       SqlConnection conn = new SqlConnection(cs);
        SqlDataAdapter da = new SqlDataAdapter("select * from student", conn);
        DataTable dt = new DataTable();
        da.Fill(dt);
        Response.Write("Total number of records are:" + dt.Rows.Count + "<br>");
        Response.Write(dt.Rows[0][0] + " " + dt.Rows[0][1]);
}
```

```
Total number of records are:9
101 deepa
```

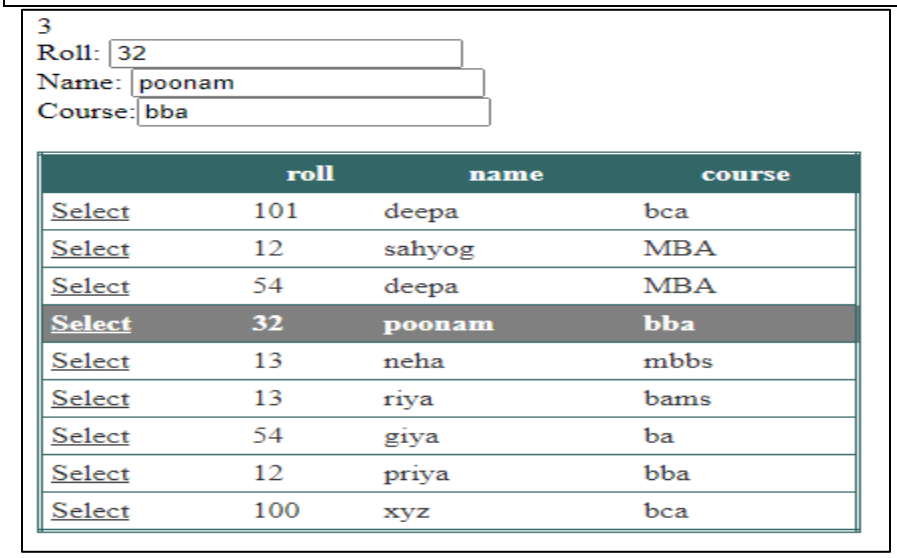
❖ Select gridview row and display in textbox or label control.

- Set AutoGenerateSelectButton to true to select a true.

```
GridView1.AutoGenerateSelectButton = true;
```

```
protected void Page_Load(object sender, EventArgs e)
    {
        GridView1.AutoGenerateSelectButton = true;
     }
protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
    {
        GridViewRow g = GridView1.SelectedRow;
        GridView1.SelectedRow.BackColor = System.Drawing.Color.Gray;
        Response.Write(g.RowIndex); //it display rows index
        TextBox1.Text = g.Cells[1].Text;
```

```
        TextBox2.Text = g.Cells[2].Text;
        TextBox3.Text = g.Cells[3].Text;
    }
```



What is ADO.NET DataSet?

- The DataSet represents a subset of the database in memory. That means the ADO.NET DataSet is a collection of tables containing relational data in memory in tabular format.
- It does not require a continuous open or active connection to the database. The DataSet is based on the disconnected architecture. This is the reason why it is used to fetch the data without interacting with any data source. We will discuss the disconnected architecture of the data set in our upcoming articles.

Note: The ADO.NET DataSet class is the core component for providing data access in a distributed and disconnected environment. The ADO.NET DataSet class belongs to the **System.Data** namespace.

- Dataset is a collection of table.
- We can store multiple table records in dataset.
- Name of first table is Table and index 0 ,second table is Table1 and index 1 and so on.
- We can access table information either by table name or by index number.

Program to display record in gridview using dataset.

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
```

```
using System.Data;
namespace database1
{
    public partial class dataset : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            string cs = "data source=DESKTOP-16RE47P\\SQLEXPRESS;initial
catalog=college1;uid=sa;pwd=deepa123";
            SqlConnection conn = new SqlConnection(cs);
            SqlCommand cmd = new SqlCommand("select * from student", conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            da.Fill(ds);
            GridView1.DataSource = ds;
            GridView1.DataBind();
        }
    }
}
```

roll	name	course
101	deepa	bca
12	sahyog	MBA
54	deepa	MBA
32	poonam	bba
13	neha	mbbs
13	riya	bams
54	giya	ba
12	priya	bba
100	xyz	bca

Program to display individual value from data set and also total number of records.

```
  protected void Page_Load(object sender, EventArgs e)
        {
            string cs = "data source=DESKTOP-16RE47P\\SQLEXPRESS;initial
catalog=college1;uid=sa;pwd=deepa123";
            SqlConnection conn = new SqlConnection(cs);
            SqlCommand cmd = new SqlCommand("select * from student", conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            da.Fill(ds);

            Response.Write(ds.Tables[0].Rows[0][0] + "<br>");
            Response.Write(ds.Tables[0].Rows[2][1] + "<br>");
            Response.Write(ds.Tables[0].Rows[1][2] + "<br>");
        }
```

101
deepa
MBA

Storing multiple tables in dataset and displaying in different gridview.

- THERE ARE 3 tables in college datatbase
- Student,course and staff

Create a subprocedure

```
CREATE PROCEDURE SAHYOG
AS
BEGIN
select * from student
SELECT * FROM COURSE
sELECT * FROM STAFF
END
```

- Sahyog is a procedure, whenever we will execute sahyog procedure, records of all the 3 table will be displayed.

```
protected void Page_Load(object sender, EventArgs e)
        {
            string cs = "data source=DESKTOP-
16RE47P\\SQLEXPRESS;initial catalog=college1;uid=sa;pwd=deepa123";
            SqlConnection conn = new SqlConnection(cs);
            SqlCommand cmd = new SqlCommand("sahyog", conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            da.Fill(ds);
            GridView1.DataSource = ds.Tables[0];//first table
            GridView2.DataSource = ds.Tables[1];//second table
            GridView3.DataSource = ds.Tables[2];//third table
            GridView1.DataBind();
            GridView2.DataBind();
            GridView3.DataBind();
}
```

Specifying name to each table

```
protected void Page_Load(object sender, EventArgs e)
{
string cs = "data source=DESKTOP-16RE47P\\SQLEXPRESS;initial
catalog=college1;uid=sa;pwd=deepa123";
            SqlConnection conn = new SqlConnection(cs);
```

```
            SqlCommand cmd = new SqlCommand("sahyog", conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            da.Fill(ds);
            ds.Tables[0].TableName = "student1";
            ds.Tables[1].TableName = "staff1";
            ds.Tables[2].TableName = "course1";

            GridView1.DataSource = ds.Tables["student1"];
            GridView2.DataSource = ds.Tables["staff1"];
            GridView3.DataSource = ds.Tables["course1"];
            GridView1.DataBind();
            GridView2.DataBind();
            GridView3.DataBind();
}
```

roll	name	course
101	deepa	bca
12	sahyog	MBA
54	deepa	MBA
32	poonam	bba
13	neha	mbbs
13	riya	bams
54	giya	ba
12	priya	bba
100	xyz	bca

cid	cname	duration	sub
10	BSC IT	3	15
11	BCA	3	10

id	sname	fees
1001	naira	54000
1002	priya	12000
1003	nisha	87000

Data Binding:

It is an ASP.NET feature that allows you to pop data directly into HTML elements and fully formatted controls.

- **Single-Value Data Binding (simple)**: Allows you to take a Page variable, property, or an expression and insert it dynamically into a page. In other words, it allows you to pop data into HTML elements. To implement single-value Data Binding, use a Binding expression like <%# Var_Prop_Expr %> within your .aspx file. Then, convert all the data binding expressions on the Page via the DataBind() method of the Page object.

Program:

singleValueDataBinding .aspx file

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="singleValueDataBinding.aspx.cs" Inherits="database1.singleValueDataBinding"
%>
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div>
         <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
     <asp:Label ID="Label1" runat="server" Text="<%# first %>"></asp:Label>
      </div>
   </form>
</body>
</html>
```

singleValueDataBinding.aspx.cs file

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace database1
{
   public partial class singleValueDataBinding : System.Web.UI.Page
   {
      public static int first = 54;
      protected void Page_Load(object sender, EventArgs e)
      {
         DataBind();
      }
      protected void Button1_Click(object sender, EventArgs e)
      {
         first++;
```

```
        DataBind();
    }}}
```

In the above program whenever we will click on button, value of first variable will get increased and will be displayed on the label1.

- **Repeated-Value Data Binding (list)**: Bind data to List controls (e.g. ListBox) or Rich Data controls (e.g. GridView), which are all controls with a DataSource property. To use Repeated-Value Data Binding, create / fill / retrieve your data object (i.e. SqlDataReader or DataSet) and then assign it as your control's DataSource (often done in the Page.Load event or in the aspx file). When you call DataBind(), the control automatically creates a full list using all the corresponding values.

Program to display data in dropdownlist.

```
    protected void Page_Load(object sender, EventArgs e)
    {
            string cs = "data source=DESKTOP-16RE47P\\SQLEXPRESS; initial
    catalog=college1;uid=sa;pwd=deepa123";
            SqlConnection conn = new SqlConnection(cs);
            SqlCommand cmd = new SqlCommand("select * from student", conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            da.Fill(ds);

            DropDownList1.DataSource = ds;
            DropDownList1.DataTextField = "roll";
            DropDownList1.DataValueField = "name";
            DataBind();
    }
```

Program to display data in listbox.

```
protected void Page_Load(object sender, EventArgs e)
{
            string cs = "data source=DESKTOP-16RE47P\\SQLEXPRESS;initial
catalog=college1;uid=sa;pwd=deepa123";
            SqlConnection conn = new SqlConnection(cs);
            SqlCommand cmd = new SqlCommand("select * from student", conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            da.Fill(ds);

            ListBox1.DataSource = ds;
```

```
            ListBox1.DataTextField = "roll";
            ListBox1.DataValueField = "name";

            DataBind();
}
```

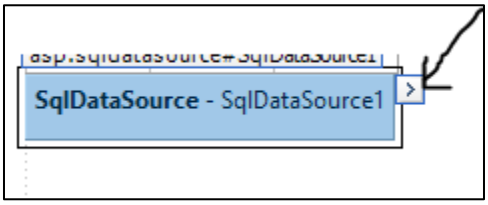
Data Source Control

- Using the ASP.NET Data Source controls, you can retrieve data from a database without writing a line of ADO.NET code.
- Data Source controls work particularly well with List and Rich Data controls.
- One example is SqlDataSource, for which command logic is supplied through four properties — SelectCommand, InsertCommand, UpdateCommand, and DeleteCommand — each of which takes a string.

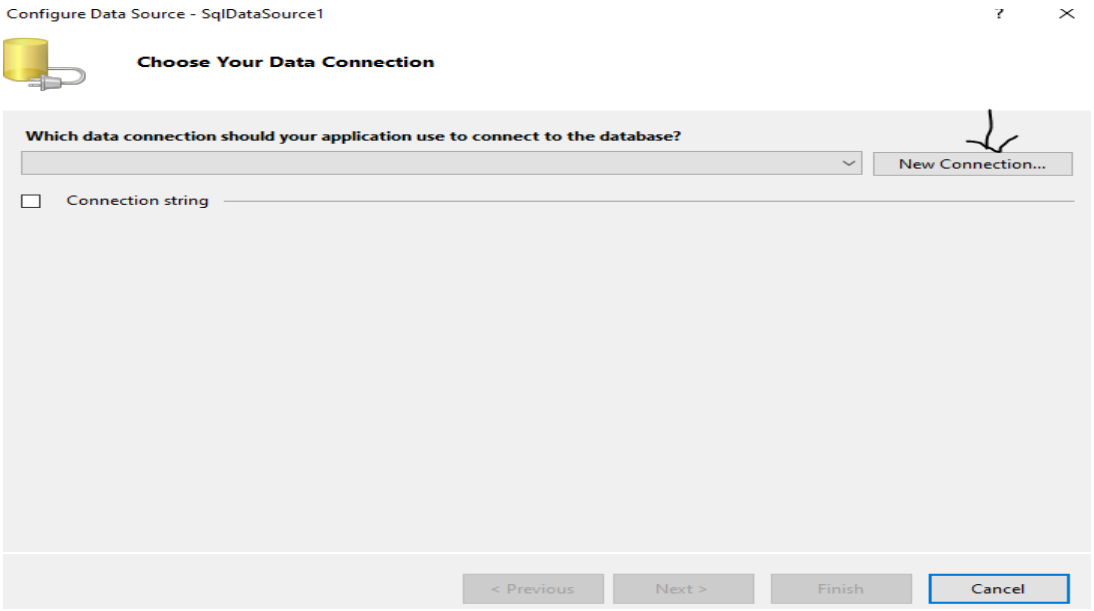
Note that with the SqlDataSource that you can still use parameterized commands (e.g. ControlParameter, QueryStringParameter, etc.). Also note that many Rich Data controls have a DataKeyNames and Auto Generate Insert/Delete/Edit Button properties.

Configuring datasource control and display record in gridview.

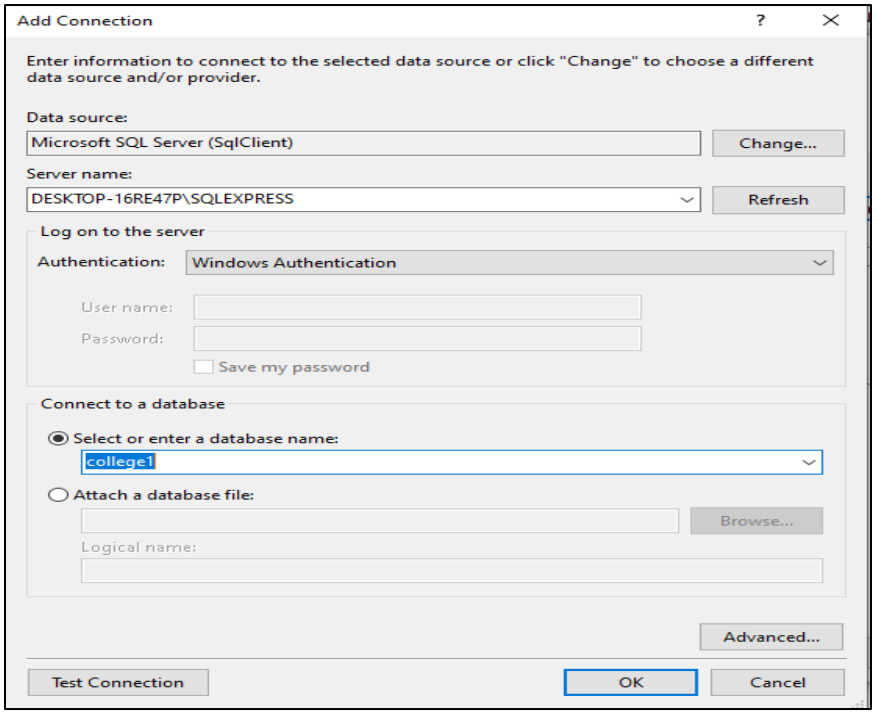
- Drag a datasource control from toolbox



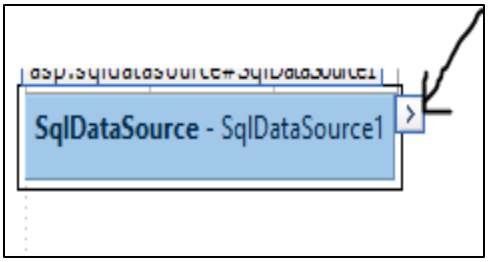
- Click on the arrow -> configure database

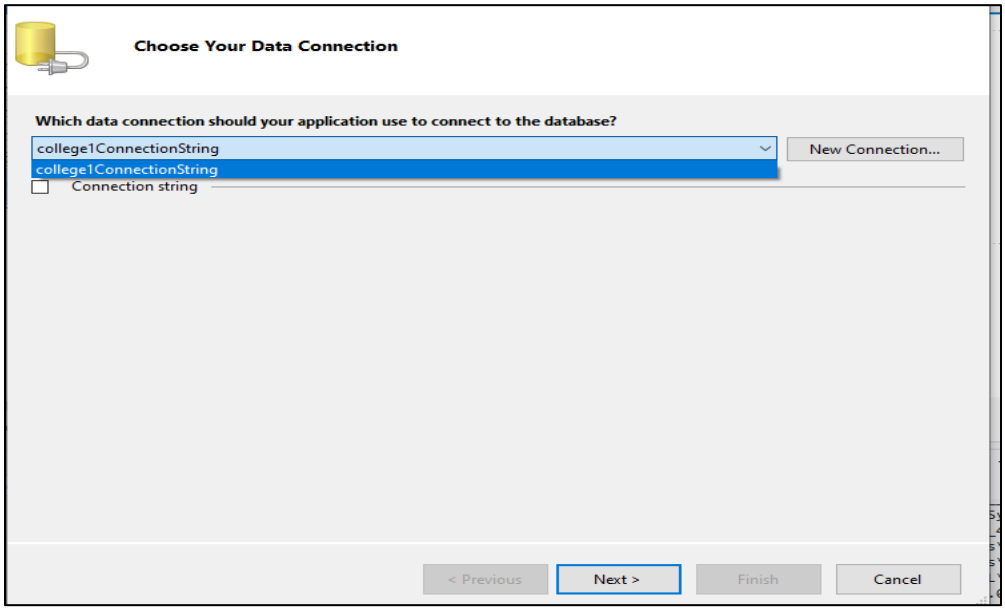


- Click on new connection, set server name and select database and click on test connection.

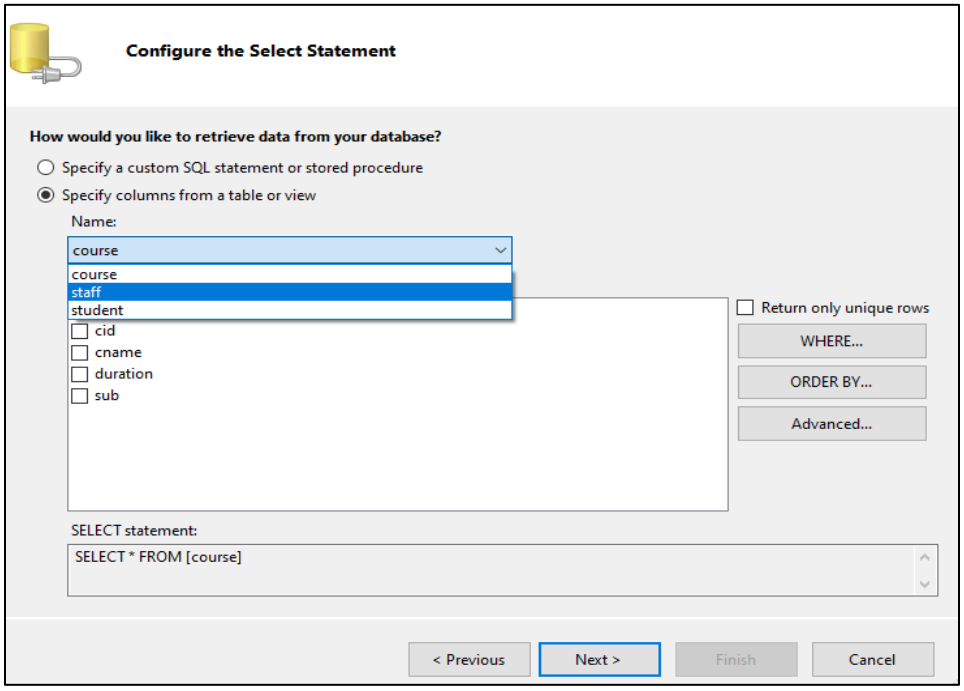


- Again click here and choose college1connectionstring and click on next

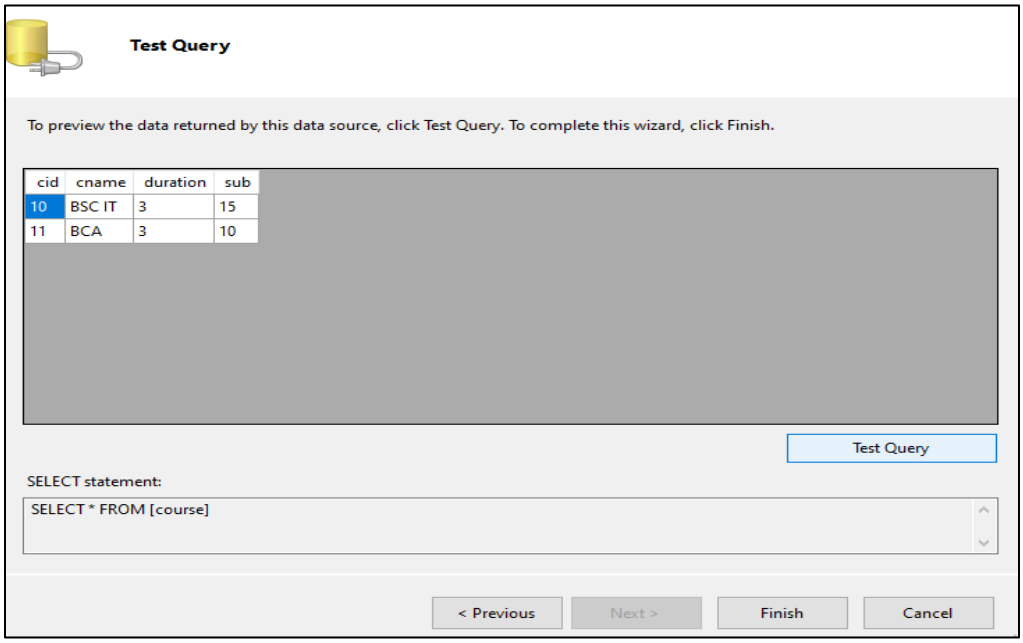




Select name of table and column that you want to display and click next

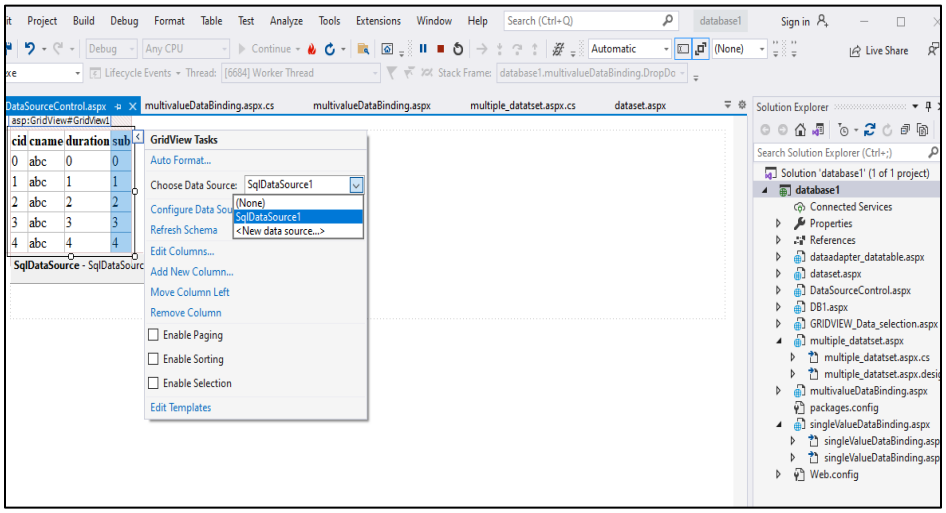


Click on test query and finish



In this way we can configure data source control.

Now set the data source of gridview control.



Output:

cid	cname	duration	sub
10	BSC IT	3	15
11	BCA	3	10

In this way we can also display data in any control using sqldatasource control.

The Data Controls:

Data controls play an important role in this purpose. Data controls used to display the records in the form of reports.

Some of the Data controls are:

1. Detailsview data control
2. GridView data control
3. DataList data control
4. FormView data control

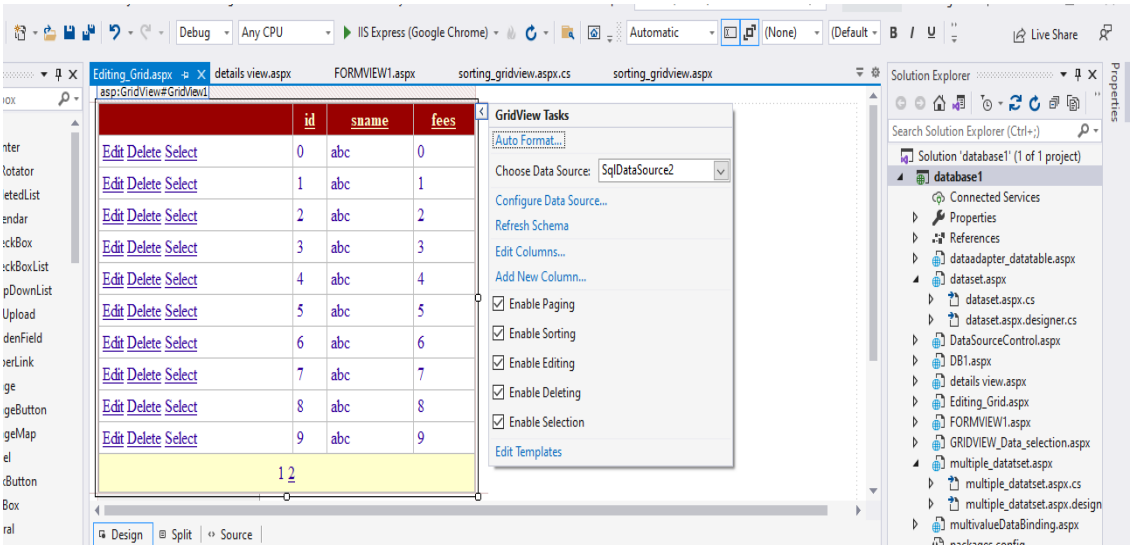
1. GridView data control:

- This control displays the data in tabular form.
- It not only support the editing and deleting of data, it also support sorting and paging of data.

Name	Addres	Phone
Aman	C-8/55 Pitampura	9876543210
Sumit	cd Block jahangirpuri	8800567489
Anshika	House No. 123 Laxmi Nagar	9999687540
Ria	B-90/12 Sector 6 Rohini	9899949770

EDITING,SORTING,UPATING,DELETING GRIDVIEW

- Drag a GRIDVIEW and sqldatasource control.
- There must a primary key in table to do insertion, updation and deletion.
- Configure the datasource same as we configured for formview control.



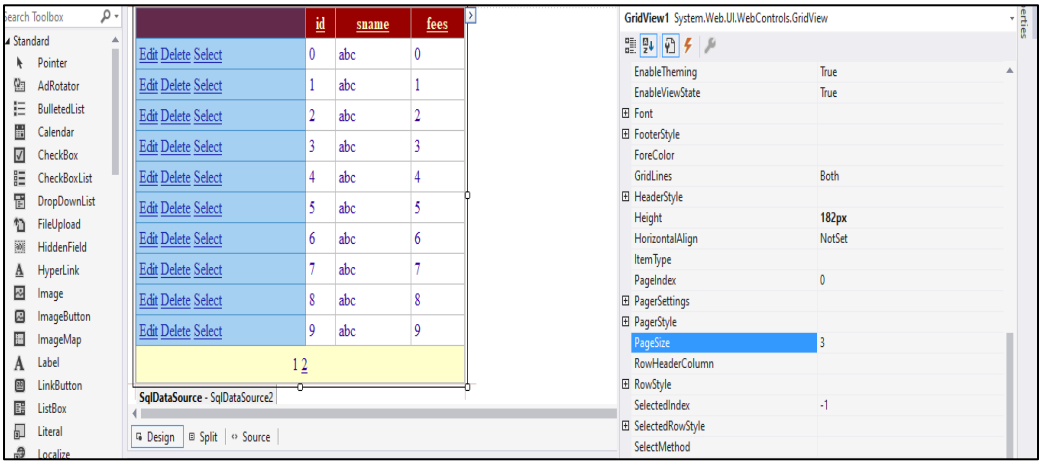
- Select the options that you want to perform.

Output:

	id	sname	fees
Edit Delete Select	1001	NNN	54000
Edit Delete Select	1002	priya	12000
Edit Delete Select	1003	nisha	87000
Edit Delete Select	1004	neha	7000
Edit Delete Select	1005	karan	87000
Edit Delete Select	1006	kahsi	7000
Edit Delete Select	1007	rohit	77000
Edit Delete Select	1008	neha	76000

Note:

- We have applied paging option but I didn't find that in my output.its because by default a web page can display 10 records and we have 8 record in our table.
- To get paging option set page size of gridview.



- We have set page size to 3 it means now 3 records will be displayed in gridview.

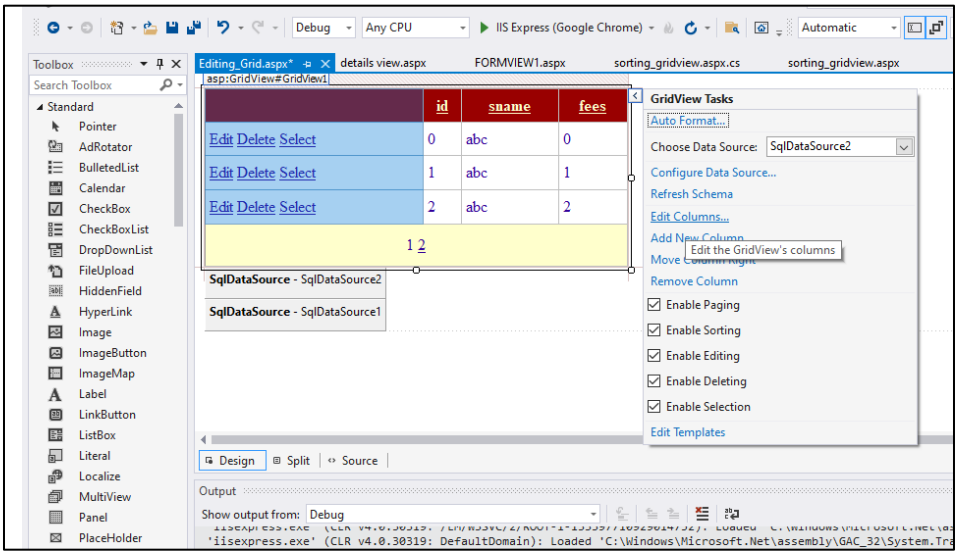
	id	sname	fees
Edit Delete Select	1004	neha	7000
Edit Delete Select	1005	karan	87000
Edit Delete Select	1006	kahsi	7000
1 2 3			

- Also we have allowed sorting. So if we click on columns it will be sorted.

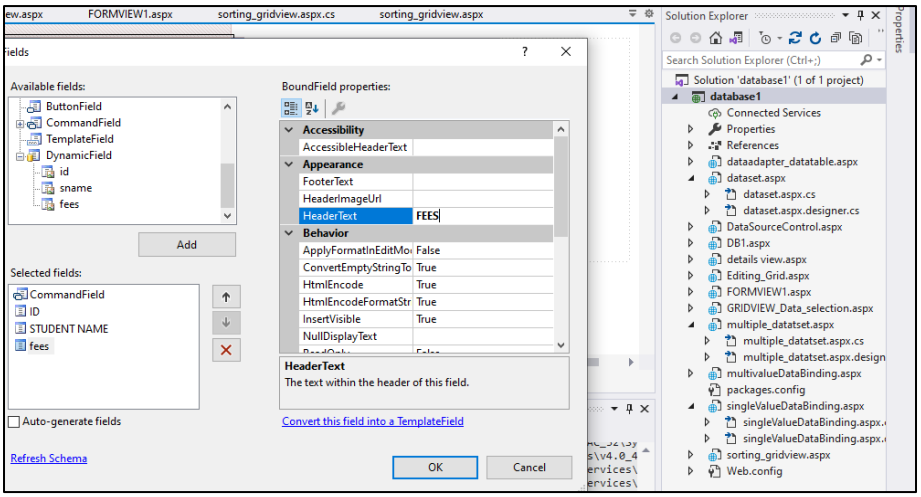
Change column name of gridview

Solution:

- Click on gridview -> edit columns



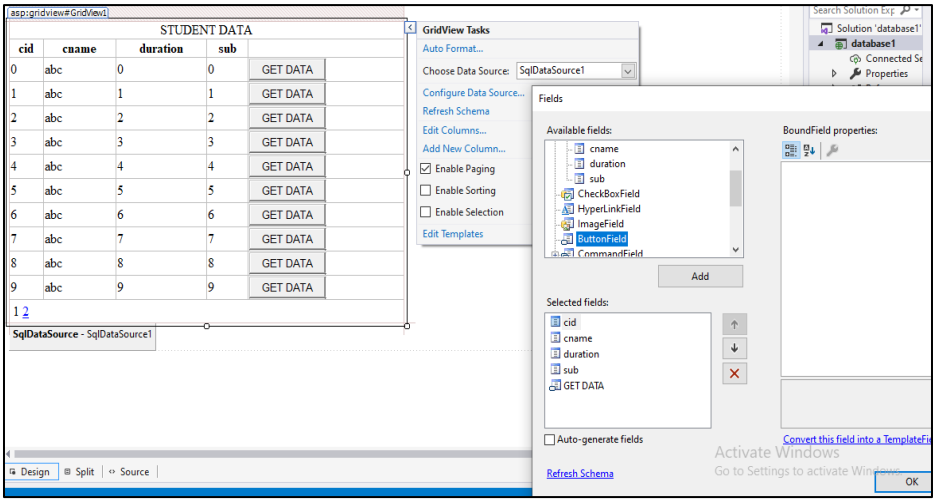
- Select the field that you want to change and set its header text.

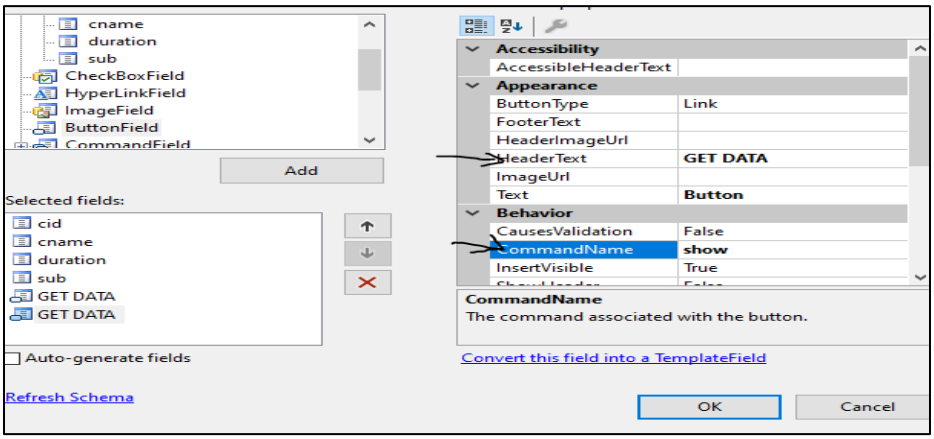


	ID	STUDENT NAME	FEES
Edit Delete Select	1001	NNN	54000
Edit Delete Select	1002	priya	12000
Edit Delete Select	1003	nisha	87000
1 2 3			

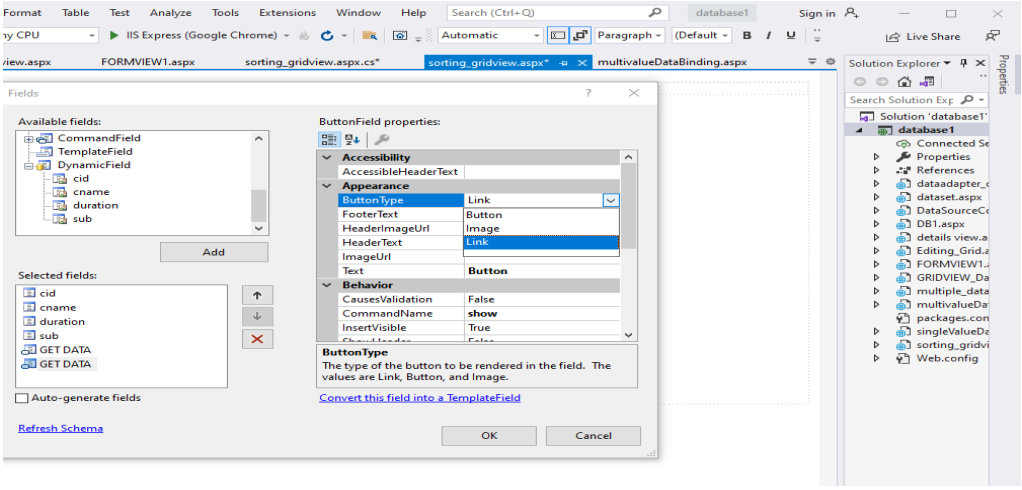
Adding button in gridview control and display records.

Click on edit columns->button field->Add->Set header text and Command name->ok





Set ButtonType to Button

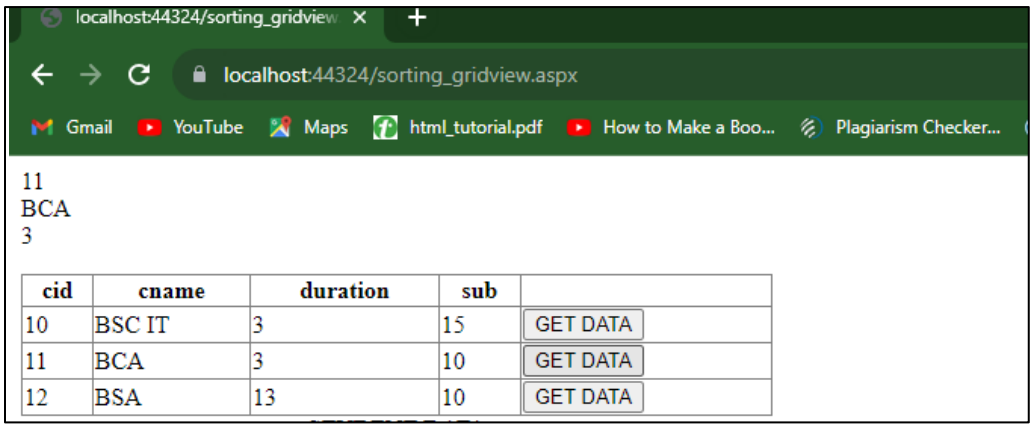


- **On RowCommand write down following code:**

```
protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
    {
      if(e.CommandName=="show")
       {
         int i = Convert.ToInt32(e.CommandArgument);

          Response.Write(GridView1.Rows[i].Cells[0].Text);
          Response.Write(" <BR> " + GridView1.Rows[i].Cells[1].Text);
          Response.Write(" <BR> " + GridView1.Rows[i].Cells[2].Text);          }

    }
```

- e. CommandName will return the command name we have set.
- e. CommandArugument return row index number.

OUTPUT:



2. FormView Control

- This control displays a single record of data at a time like DetailsView control and supports the editing of record.
- This control requires the use of template to define the rendering of each item.
- Developer can completely customize the appearance of the record.
- We can use data source control to display data in formview.

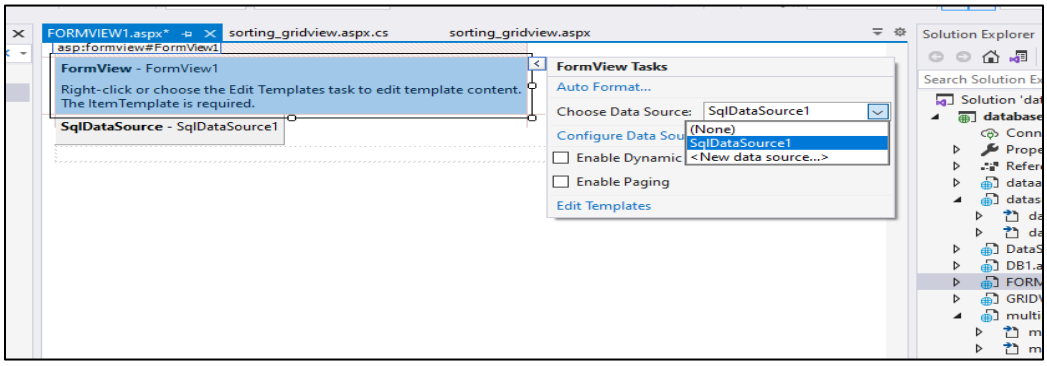
display data in formview.

Solution:

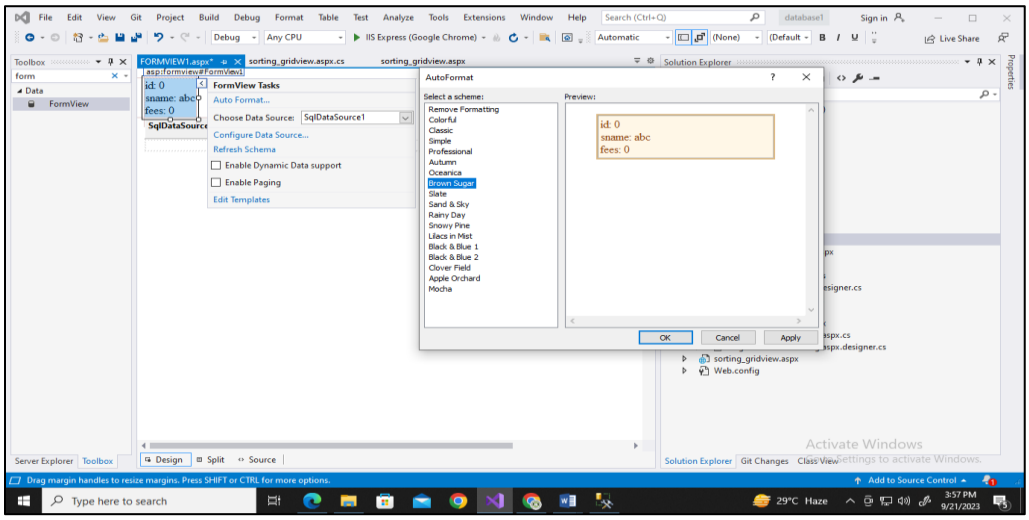
Step1: drag a formview and datasource control.

Step2: Configure data source control.

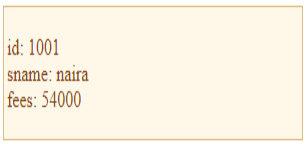
Step3: Set data source of formview control.



Step 4 [optional]: Click on auto format to pick different styles.



Output:

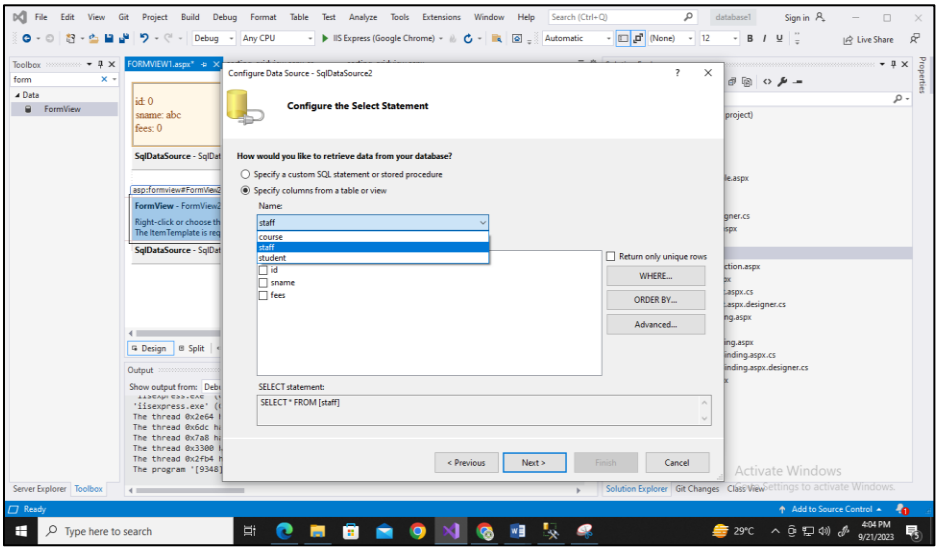


Program to allow paging, insert,deletion and updation in table using form view control.

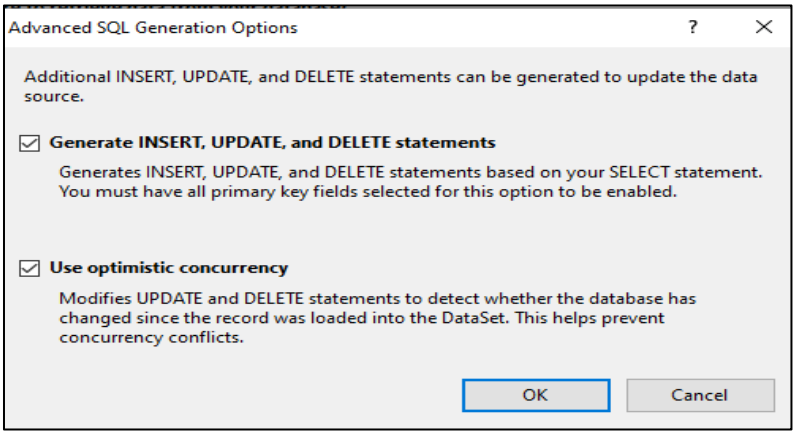
Note: to perform insertion,deletion,updation there must be a primary key in table.

Solution:

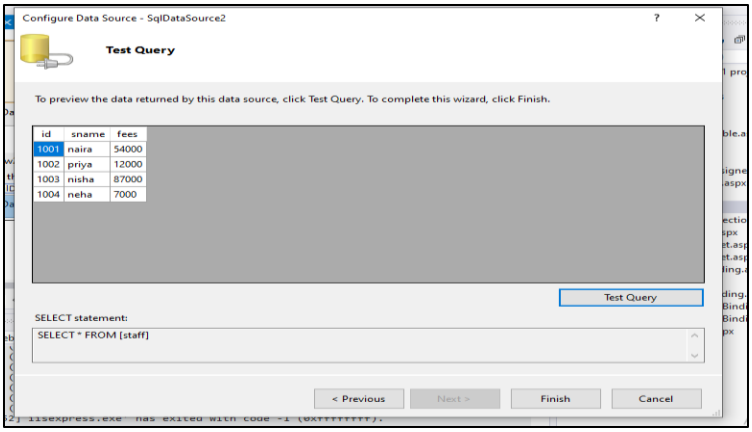
1. Drag formview and datasource control.
2. Configure datasource,select the table having primary key and click on advanced option.



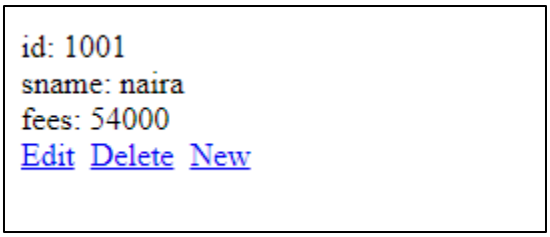
3. Select both the checkboxes ,click ok,then click on next



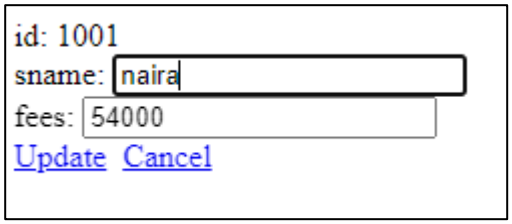
4. **Click test query then finish**



5. **Set datasource of form view control and run the code.**

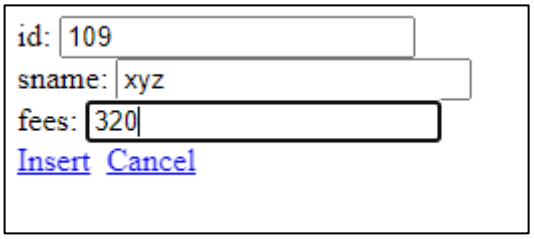


- On edit button click will get 2 options update and cancel



Here we can only change name and fees.

- On new button click will get 2 options insert and cancel



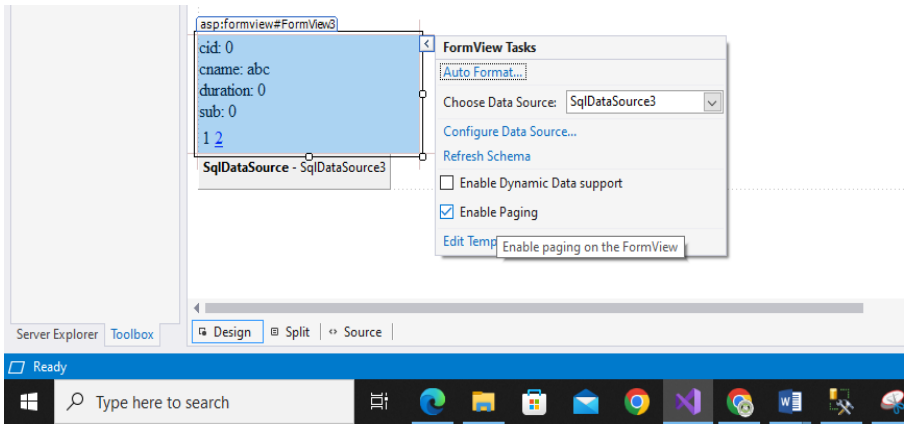
- On delete click..record will be deleted from the table

Program to allow paging,change column name [header] in form view control.

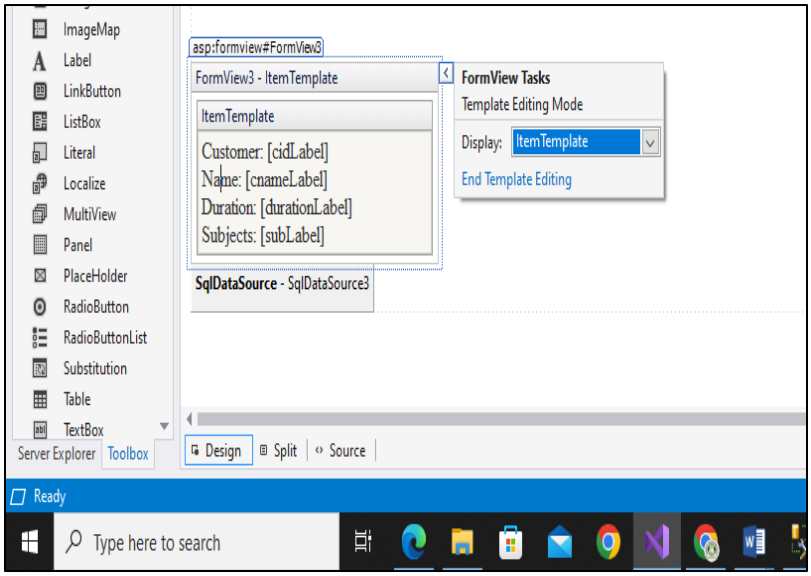
Solution:

- Drag formview control and datasource

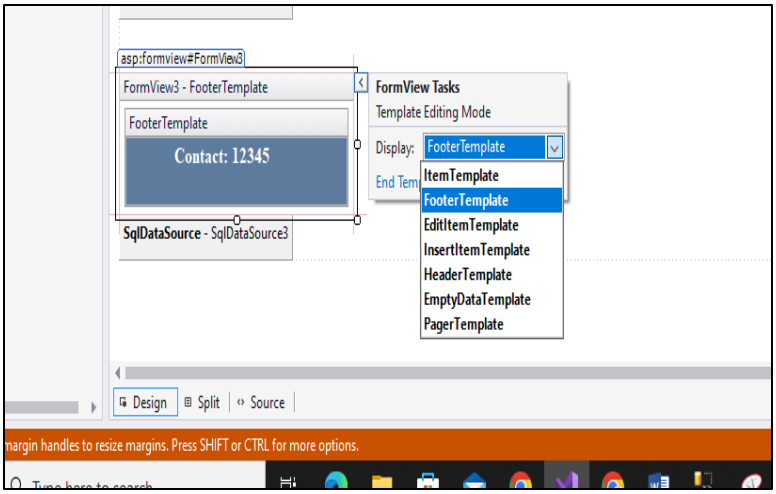
- Set enabling paging property of control



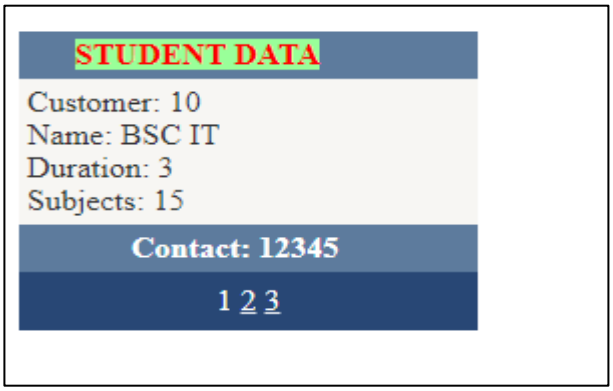
- To Change column name click on formcontrol.select ItemTemplate and make changes in column which you want to display on web page.
- ItemTemplate decides how the data will be visible.



- We can also set header,footer template and also we can add controls from toolbox and modify them inside form view control template.



Output:



3. DetailsView Control

- DetailsView control used to display a single database record of table layout in ASP.Net.
- Means it works with a single data item at a time. DetailsView control used to display record, insert, update, and delete records from database.

DetailsView Control Syntax :

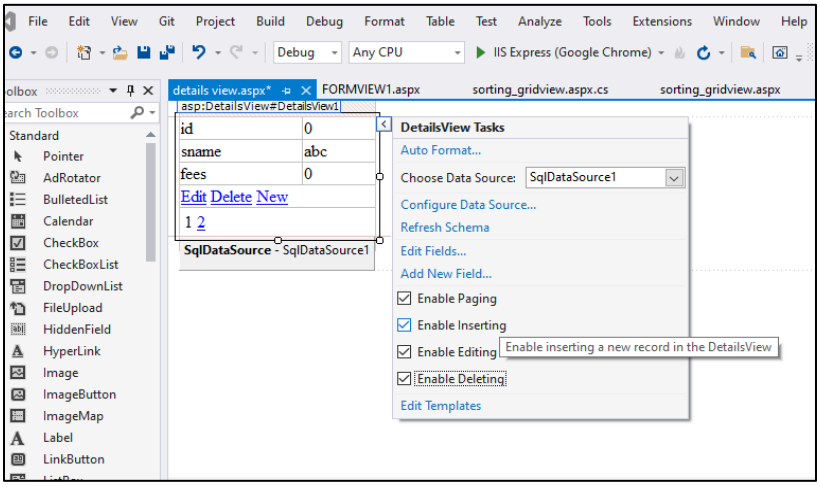
```
<asp:DetailsView ID="DetailsView1" runat="server"></asp:DetailsView>
```

Display data in details view,allow paging,insertion,deletion and updation.

Solution:

- Drag a details view and sqldatasource control.
- There must a primary key in details to do insertion, updationa nd deletion.

- Configure the datasource same as we configured for formview control.



- Select all 4 options

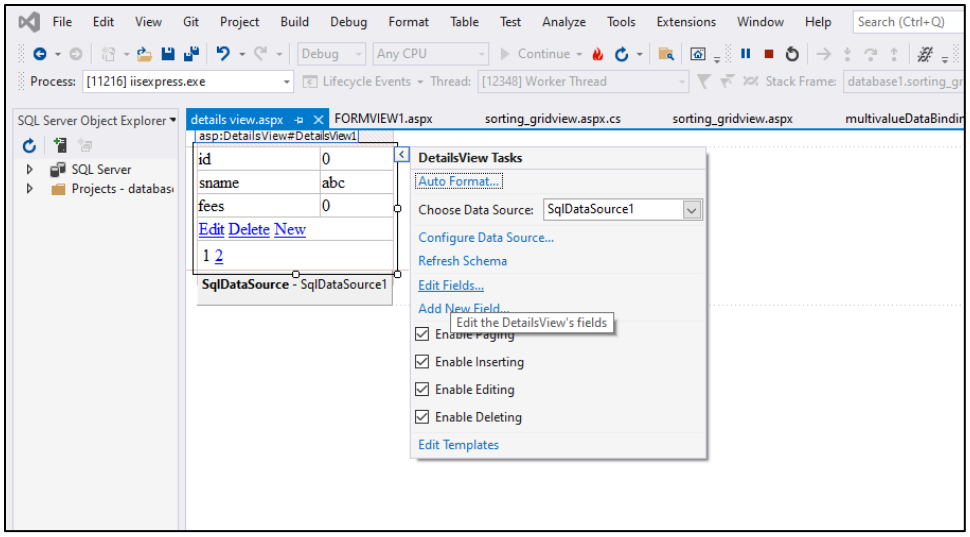
Output:

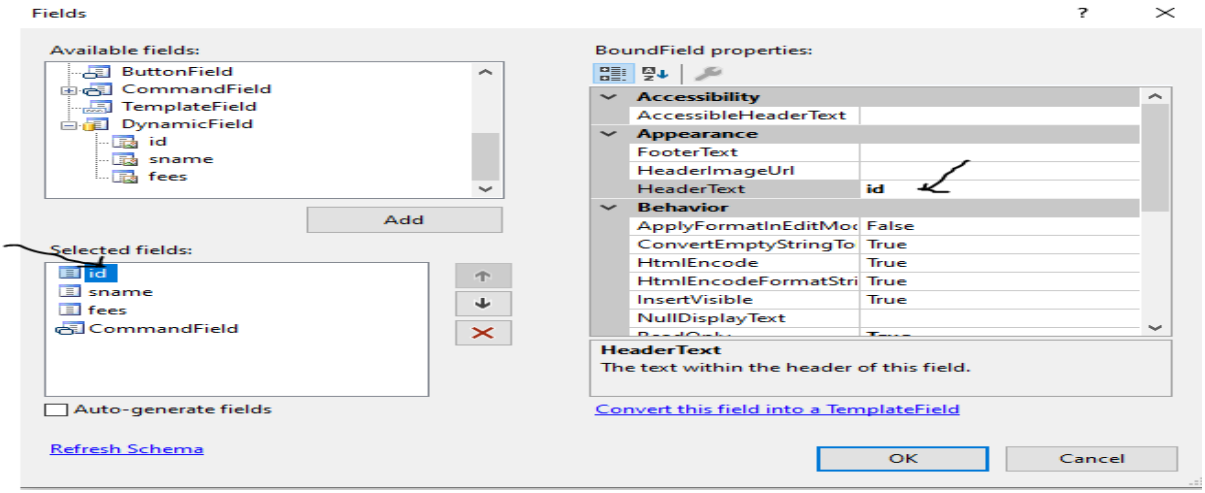
id	1003
sname	nisha
fees	87000
Edit Delete New	
1 2 3 4	

Changing field name/column name in details view.

Solution:

- Click on details view then click edit fields

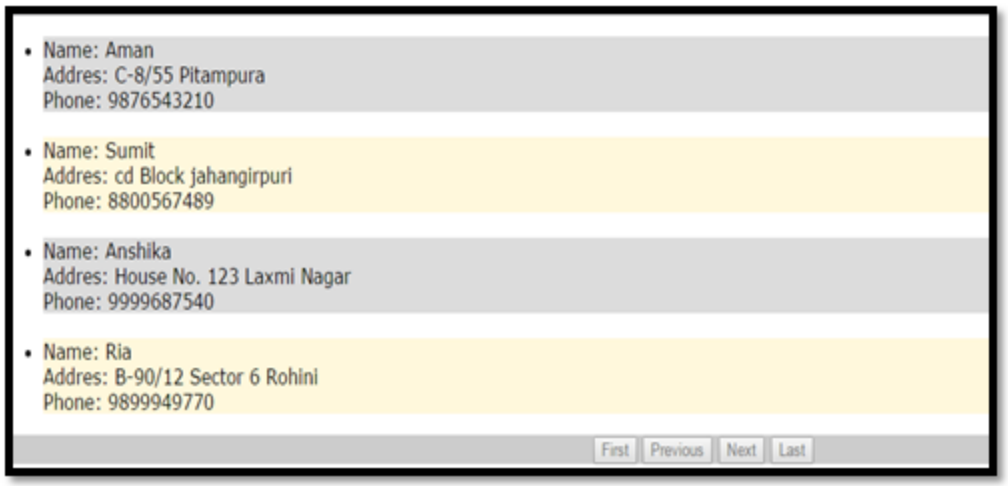




- Select the field that you want to change and change its header text and click ok.

STUDENT ID	1001
STUDENT NAME	nisha
FEES	54000
Edit Delete New	
1 2 3 4	

4. **DataListView control:**
   - This control displays data as items in a list.
   - Presentation of the data can be customized via templates.
   - Inline editing and deleting of data is supported by this control.


Name: Aman
Addres: C-8/55 Pitampura
Phone: 9876543210
Name: Sumit
Addres: cd Block jahangirpuri
Phone: 8800567489
Name: Anshika
Addres: House No. 123 Laxmi Nagar
Phone: 9999687540
Name: Ria
Addres: B-90/12 Sector 6 Rohini
Phone: 9899949770
First
Previous
Next
Last

****

XML