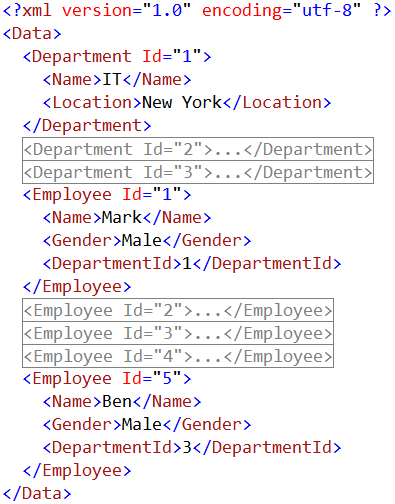
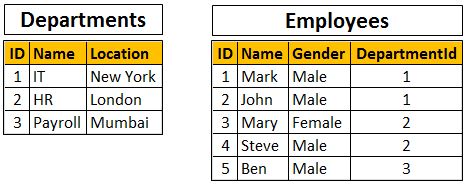
**ADO.NET offers two data access modes**  
**1.** Connection oriented data access  
**2.** Disconnected data access

we will discuss **disconnected data access.**SqlDataAdapter and DataSet objects together provide disconnected data access.  
  
A DataSet is an in-memory data store that can hold one or more tables. DataSets only hold data and do not interact with the underlying database table. The DataSet object has no knowledge of the underlying Data Source. It is the SqlDataAdapter object that retrieves data from the datasource. 

**In this video we will discuss**  
**1.** What are strongly typed datasets  
**2.** Advantage of using strongly typed datasets over untyped datasets  
**3.** Steps to generate a typed dataset using visual studio  
  
**What are strongly typed datasets**  
**1.** Strongly Typed Dataset is generated based on the Database Schema.   
**2.** Strongly Typed Dataset derive form DataSet  
**3.** In a strongly typed dataset the database table columns become properties and the type associated with each column is known at design time  
  
**Advantage of using strongly typed datasets over untyped datasets**  
Since, in a strongly typed dataset the database table columns become properties and the type associated with each column is known at design time,   
**1.** Development is much easier as we will have intellisense   
**2.** Any errors related to misspelt column names can be detected at compile time, rather than at runtime  
  
**Steps to generate a typed dataset using visual studio**  
**1.** Right click on the **Project Name**in solution explorer and select **"Add - New Item"**  
**2.** Select **"DataSet"**, give it a meaningful name and click **"Add"**. This should add a file with **.XSD** extension.  
**3.** Click on **"View"**menu item in Visual Studio and select **"Server Explorer"**  
**4.** In **"Server Explorer"**, expand **"Data Connections"**, then expand the **"Database"**, and then expand **"Tables"**  
**5.** Drag and drop the table based on which you want to generate a strongly typed dataset.  
  
**ASPX code for both WebForm1.aspx and WebForm2.aspx**  
<div style="font-family:Arial">  
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>  
    <asp:Button ID="Button1" runat="server" Text="Button"   
        onclick="Button1\_Click" />  
    <asp:GridView ID="GridView1" runat="server">  
    </asp:GridView>  
</div>  
  
**Student class used in the demo**  
public class Student  
{  
    public int ID { get; set; }  
    public string Name { get; set; }  
    public string Gender { get; set; }  
    public int TotalMarks { get; set; }  
}  
  
Please make sure to include the following using declarations on **WebForm1.aspx.cs**  
using System.Configuration;  
using System.Data;  
using System.Data.SqlClient;  
  
**WebForm1.aspx.cs code:**  
public partial class WebForm1 : System.Web.UI.Page  
{  
    protected void Page\_Load(object sender, EventArgs e)  
    {  
        if (!IsPostBack)  
        {  
            string connectionString =  
            ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;  
            SqlConnection connection = new SqlConnection(connectionString);  
            string selectQuery = "Select \* from tblStudents";  
            SqlDataAdapter dataAdapter = new SqlDataAdapter(selectQuery, connection);  
  
            DataSet dataSet = new DataSet();  
            dataAdapter.Fill(dataSet, "Students");  
  
            Session["DATASET"] = dataSet;  
  
            GridView1.DataSource = from dataRow in dataSet.Tables["Students"].AsEnumerable()   
                select new Student   
                {   
                    ID = Convert.ToInt32(dataRow["Id"]),   
                    Name = dataRow["Name"].ToString(),   
                    Gender = dataRow["Gender"].ToString(),   
                    TotalMarks = Convert.ToInt32(dataRow["TotalMarks"])   
                };  
            GridView1.DataBind();  
        }  
    }  
  
    protected void Button1\_Click(object sender, EventArgs e)  
    {  
        DataSet dataSet = (DataSet)Session["DATASET"];  
  
        if (string.IsNullOrEmpty(TextBox1.Text))  
        {  
            GridView1.DataSource = from dataRow in dataSet.Tables["Students"].AsEnumerable()   
                select new Student   
                {   
                    ID = Convert.ToInt32(dataRow["Id"]),   
                    Name = dataRow["Name"].ToString(),   
                    Gender = dataRow["Gender"].ToString(),   
                    TotalMarks = Convert.ToInt32(dataRow["TotalMarks"])   
                };  
            GridView1.DataBind();  
        }  
        else  
        {  
            GridView1.DataSource = from dataRow in dataSet.Tables["Students"].AsEnumerable()   
                where dataRow["Name"].ToString().ToUpper().StartsWith(TextBox1.Text.ToUpper())   
                select new Student   
                {   
                    ID = Convert.ToInt32(dataRow["Id"]),   
                    Name = dataRow["Name"].ToString(),   
                    Gender = dataRow["Gender"].ToString(),   
                    TotalMarks = Convert.ToInt32(dataRow["TotalMarks"])   
                };  
            GridView1.DataBind();  
        }  
    }  
}  
  
**WebForm2.aspx.cs code:**  
public partial class WebForm2 : System.Web.UI.Page  
{  
    protected void Page\_Load(object sender, EventArgs e)  
    {  
        if (!IsPostBack)  
        {  
            StudentDataSetTableAdapters.StudentsTableAdapter studentsTableAdapter =   
                new StudentDataSetTableAdapters.StudentsTableAdapter();  
            StudentDataSet.StudentsDataTable studentsDataTable =   
                new StudentDataSet.StudentsDataTable();  
            studentsTableAdapter.Fill(studentsDataTable);  
  
            Session["DATATABLE"] = studentsDataTable;  
  
            GridView1.DataSource = from student in studentsDataTable   
                select new { student.ID, student.Name, student.Gender, student.TotalMarks };  
            GridView1.DataBind();  
        }  
    }  
  
    protected void Button1\_Click(object sender, EventArgs e)  
    {  
        StudentDataSet.StudentsDataTable studentsDataTable =   
            (StudentDataSet.StudentsDataTable)Session["DATATABLE"];  
  
        if (string.IsNullOrEmpty(TextBox1.Text))  
        {  
            GridView1.DataSource = from student in studentsDataTable   
                select new { student.ID, student.Name, student.Gender, student.TotalMarks };  
            GridView1.DataBind();  
        }  
        else  
        {  
            GridView1.DataSource = from student in studentsDataTable   
                where student.Name.ToUpper().StartsWith(TextBox1.Text.ToUpper())   
                select new { student.ID, student.Name, student.Gender, student.TotalMarks };  
            GridView1.DataBind();  
        }  
    }  
}

**What is the use of SqlBulkCopy class**  
**SqlBulkCopy class is used to bulk copy data from different data sources to SQL Server database.** This class is present in System.Data.SqlClient namespace. This class can be used to write data only to SQL Server tables. However, the data source is not limited to SQL Server, any data source can be used, as long as the data can be loaded to a DataTable instance or read with a IDataReader instance.   
  
**From a performance standpoint**, SqlBulkCopy makes it very easy and efficient to copy large amounts of data.  
  
**Loading xml data into sql server table using sqlbulkcopy**  
  
We want to load the following XML data into SQL Server database.   
   
  
Notice that we have **Departments**and **Employees**data in the XML file. We would like to load Employees data into Employees table and Departments data into Departments table. After the data is loaded, the database tables should look as shown below.   
   
  
**The following are the steps to achieve this using SqlBulkCopy class**  
  
**Step 1 :**Create the database tables using the following sql script

Create table Departments

(

     ID int primary key,

     Name nvarchar(50),

     Location nvarchar(50)

)

GO

Create table Employees

(

     ID int primary key,

     Name nvarchar(50),

     Gender nvarchar(50),

     DepartmentId int foreign key references Departments(Id)

)

GO

**Step 2 :** Create a new empty asp.net web application project. Name it **Demo**.  
  
**Step 3 :**Add a new xml file to the project. Name it Data.xml. Copy and paste the following XML.

<Data>

  <Department Id="1">

    <Name>IT</Name>

    <Location>New York</Location>

  </Department>

  <Department Id="2">

    <Name>HR</Name>

    <Location>London</Location>

  </Department>

  <Department Id="3">

    <Name>Payroll</Name>

    <Location>Mumbai</Location>

  </Department>

  <Employee Id="1">

    <Name>Mark</Name>

    <Gender>Male</Gender>

    <DepartmentId>1</DepartmentId>

  </Employee>

  <Employee Id="2">

    <Name>John</Name>

    <Gender>Male</Gender>

    <DepartmentId>1</DepartmentId>

  </Employee>

  <Employee Id="3">

    <Name>Mary</Name>

    <Gender>Female</Gender>

    <DepartmentId>2</DepartmentId>

  </Employee>

  <Employee Id="4">

    <Name>Steve</Name>

    <Gender>Male</Gender>

    <DepartmentId>2</DepartmentId>

  </Employee>

  <Employee Id="5">

    <Name>Ben</Name>

    <Gender>Male</Gender>

    <DepartmentId>3</DepartmentId>

  </Employee>

</Data>

**Step 4 :** Include the database connection string in web.config file

<connectionStrings>

  <add name="CS"

           connectionString="server=.;database=Sample;integrated security=true"/>

</connectionStrings>

**Step 5 :** Add a new WebForm to the project. Drag and drop a button control on the webform. Double click the button control to generate the click event handler. Copy and paste the following code in the the click event handler method.

string cs = ConfigurationManager.ConnectionStrings["CS"].ConnectionString;

using (SqlConnection con = new SqlConnection(cs))

{

    DataSet ds = new DataSet();

    ds.ReadXml(Server.MapPath("~/Data.xml"));

    DataTable dtDept = ds.Tables["Department"];

    DataTable dtEmp = ds.Tables["Employee"];

    con.Open();

    using (SqlBulkCopy bc = new SqlBulkCopy(con))

    {

        bc.DestinationTableName = "Departments";

        bc.ColumnMappings.Add("ID", "ID");

        bc.ColumnMappings.Add("Name", "Name");

        bc.ColumnMappings.Add("Location", "Location");

        bc.WriteToServer(dtDept);

    }

    using (SqlBulkCopy bc = new SqlBulkCopy(con))

    {

        bc.DestinationTableName = "Employees";

        bc.ColumnMappings.Add("ID", "ID");

        bc.ColumnMappings.Add("Name", "Name");

        bc.ColumnMappings.Add("Gender", "Gender");

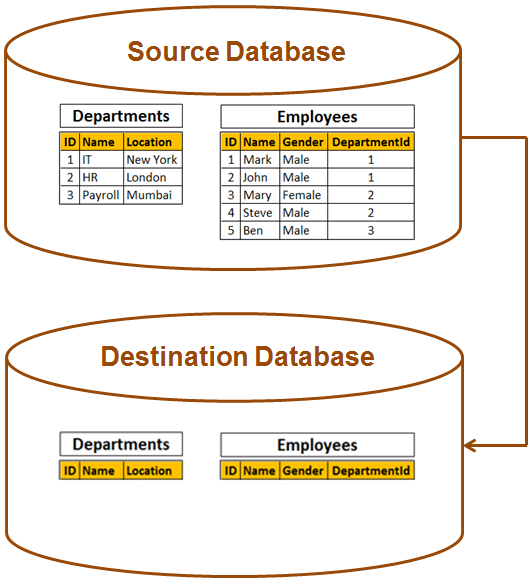
        bc.ColumnMappings.Add("DepartmentId", "DepartmentId");

        bc.WriteToServer(dtEmp);

    }

}

Part19-

**In this video we will discuss copying data from one table to another table.** The source and destination tables may be in the same database or in different databases and these database can be on the same sql server or in different servers. In [Part 18](http://csharp-video-tutorials.blogspot.com/2014/09/part-18-load-xml-data-into-sql-server.html) we discussed, loading xml data into sql server table using sqlbulkcopy. We will be continuing with the example we worked with in [Part 18](http://csharp-video-tutorials.blogspot.com/2014/09/part-18-load-xml-data-into-sql-server.html).   
  
  
  
   
**Step 1 :** Create a new database. Name it **SourceDB.** Execute the following sql script to create Departments and Employees tables, and to populate with data.

Create table Departments

(

     ID int primary key identity,

     Name nvarchar(50),

     Location nvarchar(50)

)

GO

Create table Employees

(

     ID int primary key identity,

     Name nvarchar(50),

     Gender nvarchar(50),

     DepartmentId int foreign key references Departments(Id)

)

GO

Insert into Departments values ('IT', 'New York')

Insert into Departments values ('HR', 'London')

Insert into Departments values ('Payroll', 'Muumbai')

GO

Insert into Employees values ('Mark', 'Male', 1)

Insert into Employees values ('John', 'Male', 1)

Insert into Employees values ('Mary', 'Female', 2)

Insert into Employees values ('Steve', 'Male', 2)

Insert into Employees values ('Ben', 'Male', 3)

GO

**Step 2 :** Create another new database. Name it **DestinationDB.** Execute the just the create sql script to create Departments and Employees tables. Here we have just the structure of the tables and no data. We will be moving data from SourceDB tables to DestinationDB tables.  
  
Step 3 : Include the following 2 connection strings for the Source and Destination databases in the web.config file of the Demo project we created in Part 18.

<connectionStrings>

  <add name="SourceCS"

        connectionString="server=.;database=SourceDB;integrated security=true"/>

  <add name="DestinationCS"

        connectionString="server=.;database=DestinationDB;integrated security=true"/>

</connectionStrings>

**Step 4 :** Copy and paste the following code in the button click event handler method in the code-behind file

string sourceCS =

      ConfigurationManager.ConnectionStrings["SourceCS"].ConnectionString;

string destinationCS =

      ConfigurationManager.ConnectionStrings["DestinationCS"].ConnectionString;

using (SqlConnection sourceCon = new SqlConnection(sourceCS))

{

    SqlCommand cmd = new SqlCommand("Select \* from Departments", sourceCon);

    sourceCon.Open();

    using (SqlDataReader rdr = cmd.ExecuteReader())

    {

        using (SqlConnection destinationCon = new SqlConnection(destinationCS))

        {

            using (SqlBulkCopy bc = new SqlBulkCopy(destinationCon))

            {

                bc.DestinationTableName = "Departments";

                destinationCon.Open();

                bc.WriteToServer(rdr);

            }

        }

    }

    cmd = new SqlCommand("Select \* from Employees", sourceCon);

    using (SqlDataReader rdr = cmd.ExecuteReader())

    {

        using (SqlConnection destinationCon = new SqlConnection(destinationCS))

        {

            using (SqlBulkCopy bc = new SqlBulkCopy(destinationCon))

            {

                bc.DestinationTableName = "Employees";

                destinationCon.Open();

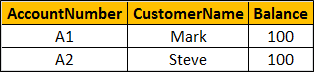
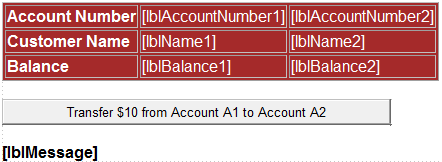
                bc.WriteToServer(rdr);

            }

        }

    }

}

**What is a Transaction**  
A Transaction ensures that either all of the database operations succeed or all of them fail. This means the job is never half done, either all of it is done or nothing is done. Let's understand this with an example.   
  
We will be using the following **Accounts** table in this demo   
   
  
The table has got 2 Accounts (A1 and A2). We want to design a web application to **transfer $10 from Account A1 to Account A2**. The design of the webform should be as shown below.   
   
  
When we click **"Transfer $10 from Account A1 to Account A2"** button, we should subtract 10 from A1 account and add 10 to A2 account. So there will be 2 database UPDATE statements. What do you think will happen if only the first update statement is executed successfully and not the second statement. $10 is deducted from the first account, but not added to the second account. This is definitely not desirable. Either both the statements should succeed or both of them should fail. If one succeeds and other fails we should also rollback the changes made by the first statement to maintain the integrity of the data. This can be achieved using transactions in ado.net.  
  
**Step 1 :** Create the Accounts table using the following SQL script

Create Table Accounts

(

     AccountNumber nvarchar(10) primary key,

     CustomerName nvarchar(50),

     Balance int

)

GO

Insert into Accounts values('A1', 'Mark', 100)

Insert into Accounts values('A2', 'Steve', 100)

GO

**Step 2 :** Create a new empty asp.net web application. Name it Demo.  
  
**Step 3 :** Include connection string in web.config file

<connectionStrings>

  <add name="CS"

           connectionString="server=.;database=Sample;integrated security=SSPI"/>

</connectionStrings>

**Step 4 :** Add a WebForm. Copy and paste the following HTML.

<div style="font-family: Arial">

<table border="1" style="background: brown; color: White">

    <tr>

        <td>

            <b>Account Number </b>

        </td>

        <td>

            <asp:Label ID="lblAccountNumber1" runat="server"></asp:Label>

        </td>

        <td>

            <asp:Label ID="lblAccountNumber2" runat="server"></asp:Label>

        </td>

    </tr>

    <tr>

        <td>

            <b>Customer Name </b>

        </td>

        <td>

            <asp:Label ID="lblName1" runat="server"></asp:Label>

        </td>

        <td>

            <asp:Label ID="lblName2" runat="server"></asp:Label>

        </td>

    </tr>

    <tr>

        <td>

            <b>Balance </b>

        </td>

        <td>

            <asp:Label ID="lblBalance1" runat="server"></asp:Label>

        </td>

        <td>

            <asp:Label ID="lblBalance2" runat="server"></asp:Label>

        </td>

    </tr>

</table>

<br />

<asp:Button ID="btnTransfer" runat="server"

            Text="Transfer $10 from Account A1 to Account A2"

            OnClick="btnTransfer\_Click" />

<br />

<br />

<asp:Label ID="lblMessage" runat="server" Font-Bold="true"></asp:Label>

</div>

**Step 5 :** Copy and paste the following code in the code-behind file.

using System;

using System.Configuration;

using System.Data.SqlClient;

namespace Demo

{

    public partial class WebForm1 : System.Web.UI.Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        {

            if (!IsPostBack)

            {

                GetAccountsData();

            }

        }

        private void GetAccountsData()

        {

            string cs = ConfigurationManager.ConnectionStrings["CS"].ConnectionString;

            using (SqlConnection con = new SqlConnection(cs))

            {

                SqlCommand cmd = new SqlCommand("Select \* from Accounts", con);

                con.Open();

                SqlDataReader rdr = cmd.ExecuteReader();

                while (rdr.Read())

                {

                    if (rdr["AccountNumber"].ToString() == "A1")

                    {

                        lblAccountNumber1.Text = "A1";

                        lblName1.Text = rdr["CustomerName"].ToString();

                        lblBalance1.Text = rdr["Balance"].ToString();

                    }

                    else

                    {

                        lblAccountNumber2.Text = "A2";

                        lblName2.Text = rdr["CustomerName"].ToString();

                        lblBalance2.Text = rdr["Balance"].ToString();

                    }

                }

            }

        }

        protected void btnTransfer\_Click(object sender, EventArgs e)

        {

            string cs = ConfigurationManager.ConnectionStrings["CS"].ConnectionString;

            using (SqlConnection con = new SqlConnection(cs))

            {

                con.Open();

                // Begin a transaction. The connection needs to

                // be open before we begin a transaction

                SqlTransaction transaction = con.BeginTransaction();

                try

                {

                    // Associate the first update command with the transaction

                    SqlCommand cmd = new SqlCommand

                        ("Update Accounts set Balance = Balance - 10 where AccountNumber = 'A1'"

                        , con, transaction);

                    cmd.ExecuteNonQuery();

                    // Associate the second update command with the transaction

                    cmd = new SqlCommand

                        ("Update Accounts set Balance = Balance + 10 where AccountNumber = 'A2'"

                        , con, transaction);

                    cmd.ExecuteNonQuery();

                    // If all goes well commit the transaction

                    transaction.Commit();

                    lblMessage.ForeColor = System.Drawing.Color.Green;

                    lblMessage.Text = "Transaction committed";

                }

                catch

                {

                    // If anything goes wrong, rollback the transaction

                    transaction.Rollback();

                    lblMessage.ForeColor = System.Drawing.Color.Red;

                    lblMessage.Text = "Transaction rolled back";

                }

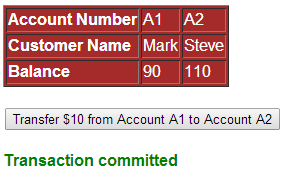
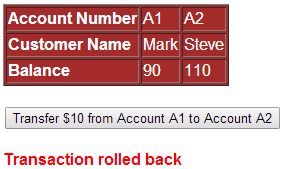
            }

            GetAccountsData();

        }

    }

}

**Testing :** Run the apllication and click the**"Transfer $10 from Account A1 to Account A2"** button. Notice that $10 is deducted from Account A1 and added to Account A2 and the transaction is committed.   
   
  
Let's now deliberately introduce a change that would crash the application at run time after executing the first update statement.   
  
CHANGE THE FOLLOWING LINE  
cmd = new SqlCommand("Update Accounts set Balance = Balance + 10 where AccountNumber = 'A2'", con, transaction);  
  
TO  
cmd = new SqlCommand("Update Accounts1 set Balance = Balance + 10 where AccountNumber = 'A2'", con, transaction);  
  
Run the apllication again and click the **"Transfer $10 from Account A1 to Account A2"** button. Notice that the transaction is rolled back and the data integrity is not lost.   


SqlConnection Object

To interact with a database, you must have a connection to it.

The connection helps identify the database server, the database name, user name, password, and other parameters that are required for connecting to the data base.

A connection object is used by command objects so they will know which database to execute the command on.

SqlConnection object lets you manage a connection to a data source.

you will have to do in code is to instantiate the connection object, open the connection, and then close the connection.

Data Source-- Identifies the server. Could be local machine, machine domain name, or IP Address.

Initial Catalog-- Database name.

Integrated Security-- Set to SSPI to make connection with user’s Windows login

Two ways-

1.SqlConnection connection = new SqlConnection("data source=.; database=SampleDB; integrated security=SSPI");

2.//First create an instance of SqlConnection class using the parameter-less constructor

SqlConnection connection = new SqlConnection();

//Then set the ConnectionString property of the connection object

connection.ConnectionString = "data source=.; database=SampleDB; integrated security=SSPI";

The ConnectionString parameter is a string made up of Key/Value pairs that has the information required to create a connection object.

Steps-

Instantiate the SqlConnection.

Open the connection.

Pass the connection to other ADO.NET objects.

Perform database operations with the other ADO.NET objects.

Close the connection.

Ex.1-

class Program

{

static void Main(string[] args)

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = "Data source=SUNNY;Initial catalog=demo;Integrated security=true";

SqlDataReader rdr;

conn.Open();

string query = "select name from student";

SqlCommand cmd = new SqlCommand(query, conn);

rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Console.WriteLine(rdr[0]);

}

}

}

1-The ExecuteReader method is used to query data.

The ExecuteReader method returns a SqlDataReader object for viewing the results of a select query.

2-ExecuteNonQuery method is used to insert,update and delete data.

The executenonquery is the most frequently used method in sqlcommand object and it is used for executing statements that do not return result set.

It performs ddl(creating and stored procedure,views) and dml(insert,update,delete) operations.

3-ExecuteScalar method is used to return a single value means if you need a single value from database, which could be a count, sum, average, or other aggregated value from a data set then we use execute scalar method.

The SQLCommand Object-

Ex:-1

// 1. Instantiate a new command with a query and connection

SqlCommand cmd = new SqlCommand("select CategoryName from Categories", conn);

// 2. Call Execute reader to get query results

SqlDataReader rdr = cmd.ExecuteReader();

Ex:2-

SqlConnection conn = new SqlConnection();

conn.ConnectionString = "Data source=SUNNY;Initial catalog=demo;Integrated security=true";

conn.Open();

string query = @"insert into student (name,age,city) values('raj', '22','yuganda')";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.ExecuteNonQuery();

conn.Close();

Ex:3-

static void Main(string[] args)

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = "Data source=SUNNY;Initial catalog=demo;Integrated security=true";

conn.Open();

string query = @"update student set name='wer' where name='q'";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.ExecuteNonQuery();

conn.Close();

}

Ex:4-

static void Main(string[] args)

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = "Data source=SUNNY;Initial catalog=demo;Integrated security=true";

conn.Open();

string query = @"delete from student where name='wer'";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.ExecuteNonQuery();

conn.Close();

}

Ex:5-

SqlConnection conn = new SqlConnection();

conn.ConnectionString = "Data source=SUNNY;Initial catalog=demo;Integrated security=true";

conn.Open();

string query = @"select count(\*) from student";

SqlCommand cmd = new SqlCommand(query, conn);

int r=(int)cmd.ExecuteScalar();

conn.Close();

Console.WriteLine(r);

Ex:6-

using System;

using System.Data.SqlClient;

class Program

{

static void Main(string[] args)

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = "Data source=SUNNY;Initial catalog=demo;Integrated security=true";

SqlDataReader rdr = null;

conn.Open();

string query = "select \* from student";

SqlCommand cmd = new SqlCommand(query, conn);

rdr = cmd.ExecuteReader();

while (rdr.Read())

{

string name = (string)rdr["name"];

int age = (int)rdr["age"];

string city = (string)rdr["city"];

Console.Write(name);

Console.Write(age);

Console.Write(city);

Console.WriteLine();

}

}

}

or

using gridview

protected void Button1\_Click(object sender, EventArgs e)

{

SqlConnection conn = new SqlConnection();

conn.ConnectionString = "Data source=SUNNY;Initial catalog=demo;Integrated security=true";

conn.Open();

string query = "select \* from student";

SqlCommand cmd = new SqlCommand(query, conn);

GridView1.DataSource = cmd.ExecuteReader();

GridView1.DataBind();

}

sql injection-

SQL Injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements (also commonly referred to as a malicious payload) that control a web application's database server