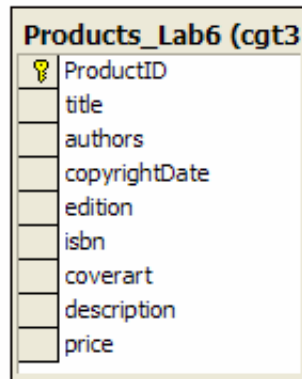


Lab06.sql - Use this database table

```
-- Create the table to interact with
CREATE TABLE Products_Lab6(
ProductID      int IDENTITY PRIMARY KEY,
title          varchar(100),
authors        varchar(100),
copyrightDate  varchar(6),
edition        varchar(6),
isbn           varchar(25),
coverart       varchar(100),
description    varchar(100),
price          varchar(100)
);
```



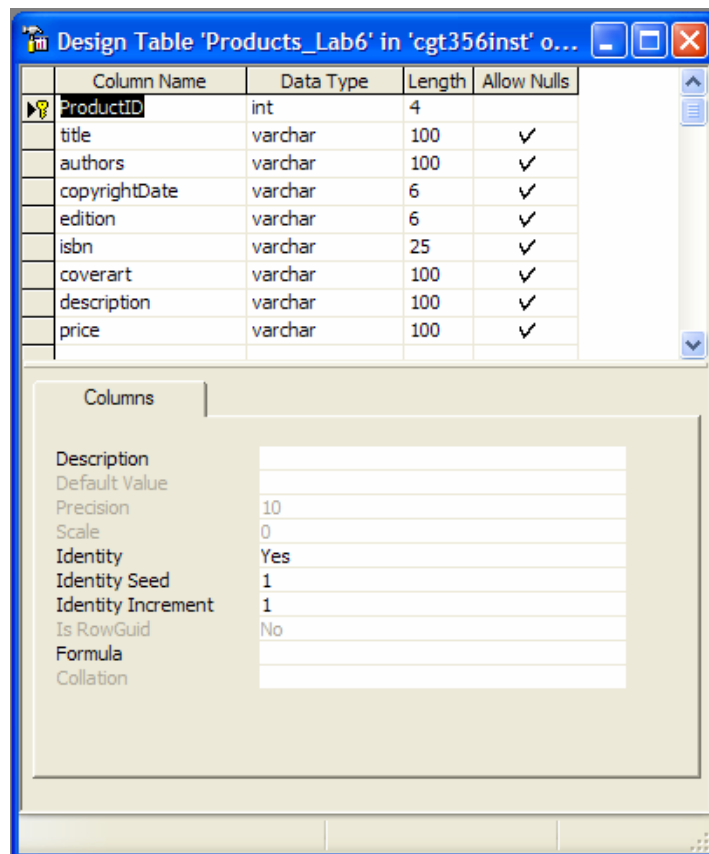
The screenshot shows a table structure for 'Products_Lab6 (cgt3'. The columns are listed in a single column, with 'ProductID' marked as the primary key with a key icon.

ProductID
title
authors
copyrightDate
edition
isbn
coverart
description
price

```
-- Insert some data to begin with
INSERT INTO Products_Lab6(title, authors, copyrightDate, edition, isbn, coverart, description,
price) VALUES('Visual Basic .NET How to Program: Second Edition', 'Harvey M. Deitel, Paul J. Deitel
& Tem R. Nieto', '2002', 2, '0-13-029363-6', 'vbnethp2.png', 'Microsoft Visual Basic .NET',
76.00);
```

```
INSERT INTO Products_Lab6(title, authors, copyrightDate, edition, isbn, coverart, description,
price) VALUES('C++ How to Program: Fourth Edition', 'Harvey M. Deitel & Paul J. Deitel', '2002', 4,
'0-13-038474-7', 'cpphtp4.png', 'Introduces Web Programming with CGI', 76.00);
```

```
INSERT INTO Products_Lab6(title, authors, copyrightDate, edition, isbn, coverart, description,
price) VALUES('C# How to Program: First Edition', 'Harvey M. Deitel, Paul J. Deitel, Jeff
Listfield, Tem R. Nieto, Cheryl Yaeger & Marina Zlatkina', '2002', 1, '0-13-062221-4',
'csharphtp1.png', 'Introduces .NET and Web services', 76.00);
```



The screenshot shows the 'Design Table' window for 'Products_Lab6' in 'cgt356inst' o... The window has a table with columns: Column Name, Data Type, Length, and Allow Nulls. The columns are: ProductID (int, 4, No), title (varchar, 100, Yes), authors (varchar, 100, Yes), copyrightDate (varchar, 6, Yes), edition (varchar, 6, Yes), isbn (varchar, 25, Yes), coverart (varchar, 100, Yes), description (varchar, 100, Yes), and price (varchar, 100, Yes). Below the table is a 'Columns' section with a list of properties for the selected column (ProductID): Description, Default Value, Precision (10), Scale (0), Identity (Yes), Identity Seed (1), Identity Increment (1), Is RowGuid (No), Formula, and Collation.

Column Name	Data Type	Length	Allow Nulls
ProductID	int	4	
title	varchar	100	✓
authors	varchar	100	✓
copyrightDate	varchar	6	✓
edition	varchar	6	✓
isbn	varchar	25	✓
coverart	varchar	100	✓
description	varchar	100	✓
price	varchar	100	✓

Columns

Description

Default Value

Precision 10

Scale 0

Identity Yes

Identity Seed 1

Identity Increment 1

Is RowGuid No

Formula

Collation

Lab06.aspx Browser View

Select	Insert	Update	Delete					
ProductID	title	authors	copyrightDate	edition	isbn	coverart	description	price
7	Microsoft Visual	Jon Jagger	2003	2003	0735619093	Microsoft	Teach yourself	26.39
8	Microsoft Visual	Jon Jagger	2003	2003	0735619093	Microsoft	Teach yourself	26.39
9	New Title Value	Jon Jagger	2003	2003	0735619093	Microsoft	Teach yourself	26.39

Lab06.aspx

```
<%@ Page language="c#" Codebehind="Lab06.aspx.cs" AutoEventWireup="false" Inherits="Lab06.Lab06" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
  <HEAD>
    <title>Lab06</title>
    <meta name="CODE_LANGUAGE" Content="C#">
    <meta name="vs_defaultClientScript" content="JavaScript">
    <meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5">
  </HEAD>
  <body MS_POSITIONING="GridLayout">
    <form id="Form1" method="post" runat="server">
      <asp:Button Runat="server" ID="select" Text="Select" OnClick="Select_Click" />
      <asp:Button Runat="server" ID="insert" Text="Insert" OnClick="Insert_Click" />
      <asp:Button Runat="server" ID="update" Text="Update" OnClick="Update_Click" />
      <asp:Button Runat="server" ID="delete" Text="Delete" OnClick="Delete_Click" />

      <!-- A DataGrid is a data bound list control that displays the items from a
      data source in a table. Use the DataGrid control to display the fields of a
      data source as columns in a table. Each row in the DataGrid control represents
      a record in the data source. The DataGrid control supports selection, editing,
      deleting, paging, and sorting. -->
      <asp:DataGrid Runat="server" ID="results" />
    </form>
  </body>
</HTML>
```

Lab06.aspx.cs

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

namespace Lab06
{
    /// <summary>
    /// //////////////////////////////////////
    ///     Author: Ronald J. Glotzbach
    ///
    ///     Date: February 16, 2005
    ///
    ///     Project: CGT 456 Lab06 Solution A
    /// //////////////////////////////////////
    /// Summary description for Lab06 class.
    /// Lab06 uses ExecuteReader and ExecuteNonQuery to execute SELECT,
    /// INSERT, UPDATE, and DELETE statements.
    /// </summary>
    public class Lab06 : System.Web.UI.Page
    {
        // Declare global variables
        // Pull controls over from aspx
        protected System.Web.UI.WebControls.DataGrid results;
        protected System.Web.UI.WebControls.Button select;
        protected System.Web.UI.WebControls.Button insert;
        protected System.Web.UI.WebControls.Button update;
        protected System.Web.UI.WebControls.Button delete;
        // Declare global objects to be used
        public String ConnStr = "server=ipaddy;uid=xxx;pwd=yyy;database=zzz";
        public SqlConnection oConn;
        public SqlCommand myCommand;
        public SqlDataReader myReader;
        public SqlDataAdapter sda;
        public DataSet ds;
        public DataTable dt;
        public String Sql;

        /// <summary>
        /// function Select_Click()
        /// When the Select All Button is clicked, all of the records will be
        /// selected and binded to the DataGrid.
        /// </summary>
        /// <param name="Source"></param>
        /// <param name="E"></param>
        protected void Select_Click(Object Source, EventArgs E)
        {
            // Create the SQL statement
            Sql= "SELECT * FROM Products_Lab6";
            // Create a new connection object using the connection string above.
            // oConn represents an open connection to a SQL Server database.
            oConn = new SqlConnection(ConnStr);
            // Create a new SQL command using the SQL command and connection object
            myCommand = new SqlCommand(Sql, oConn);

            // Open the connection to the database
            oConn.Open();

            // set the data source
            results.DataSource = myCommand.ExecuteReader();
        }
    }
}
```

```

    // bind the data source to the datagrid
    results.DataBind();

    // Close the connection to the database
    oConn.Close();
}

/// <summary>
/// function Insert_Click()
/// When the Insert Button is clicked, a new record will be inserted
/// into the database. The Primary Key is set to IDENTITY, so it will
/// auto-increment.
/// </summary>
/// <param name="Source"></param>
/// <param name="E"></param>
protected void Insert_Click(Object Source, EventArgs E)
{
    // Create the SQL statement
    Sql = "INSERT INTO Products_Lab6 (title, authors, copyrightDate, edition, isbn, ";
    Sql += "coverart, description, price) VALUES('Microsoft Visual', 'Jon Jagger', ";
    Sql += "'2003', '2003', '0735619093', 'Microsoft', 'Teach yourself', '26.39')";
    // Create a new connection object using the connection string above.
    // oConn represents an open connection to a SQL Server database.
    oConn = new SqlConnection(ConnStr);
    // Create a new SQL command using the SQL command and connection object.
    // myCommand represents a SQL statement or stored procedure to execute
    // against a SQL Server database.
    myCommand = new SqlCommand(Sql, oConn);

    // Open the connection to the database
    oConn.Open();
    // Execute the SQL statement against the connection. ExecuteNonQuery will
    // return the number of rows affected, but does not return a recordset.
    myCommand.ExecuteNonQuery();
    // Close the connection to the database
    oConn.Close();

    // Call the Select_Click() function to redisplay the data in the browser.
    Select_Click(Source, E);
}

/// <summary>
/// function Update_Click()
/// When the Update Button is clicked, the record with the maximum
/// ProductID is updated with a different Title.
/// </summary>
/// <param name="Source"></param>
/// <param name="E"></param>
protected void Update_Click(Object Source, EventArgs E)
{
    // Get the max id
    // Create the SQL statement
    Sql= "SELECT Max(ProductID) AS MaxID FROM Products_Lab6";
    // Create a new connection object using the connection string above.
    // oConn represents an open connection to a SQL Server database.
    oConn = new SqlConnection(ConnStr);
    // Create a new SQL command using the SQL command and connection object.
    // myCommand represents a SQL statement or stored procedure to execute
    // against a SQL Server database.
    myCommand = new SqlCommand(Sql, oConn);

    // Open the connection to the database
    oConn.Open();

    // Instantiate an instance of the SqlDataAdapter class
    // The SqlDataAdapter, serves as a bridge between a DataSet and SQL Server

```

```

// for retrieving and saving data. The SqlDataAdapter provides this bridge
// by mapping Fill, which changes the data in the DataSet to match the data
// in the data source, and Update, which changes the data in the data source
// to match the data in the DataSet, using the appropriate Transact-SQL
// statements against the data source.
sda = new SqlDataAdapter();
// Set the value of the SDA select command equal to the command object.
// SelectCommand gets or sets a SQL statement used to select records in the data source.
sda.SelectCommand = myCommand;

// Instantiate an instance of the DataSet class which represents
// an in-memory cache of data.
ds = new DataSet();
// When the SqlDataAdapter fills a DataSet, it will create the necessary
// tables and columns for the returned data if they do not already exist.
sda.Fill(ds);

// Instantiate an instance of the DataTable class which represents
// one table of in-memory data. Additional Info: To add tables to the
// collection, use Add method of the DataTableCollection. To remove
// tables, use the Remove method.
dt = new DataTable();
// Get the first array index of the collection of tables contained in the DataSet
dt = ds.Tables[0];

// If a record exists, Do the update
if(dt.Rows[0][0].ToString() != "")
{
    // Create the SQL statement
    Sql = "UPDATE Products_Lab6 SET Title='New Title Value' WHERE ProductID=";
    Sql += dt.Rows[0][0].ToString();
    // Create a new SQL command using the SQL command and connection object
    myCommand = new SqlCommand(Sql, oConn);

    // Execute the SQL statement against the connection. ExecuteNonQuery will
    // return the number of rows affected, but does not return a recordset.
    myCommand.ExecuteNonQuery();
}
// Close the connection to the database
oConn.Close();

// Call the Select_Click() function to redisplay the data in the browser.
Select_Click(Source, E);
}

/// <summary>
/// function Delete_Click()
/// When the Delete Button is clicked, the record with the maximum
/// ProductID is deleted from the database.
/// </summary>
/// <param name="Source"></param>
/// <param name="E"></param>
protected void Delete_Click(Object Source, EventArgs E)
{
    // Get the max id
    // Create the SQL statement
    Sql= "SELECT Max(ProductID) AS MaxID FROM Products_Lab6";
    // Create a new connection object using the connection string above.
    // oConn represents an open connection to a SQL Server database.
    oConn = new SqlConnection(ConnStr);
    // Create a new SQL command using the SQL command and connection object.
    // myCommand represents a SQL statement or stored procedure to execute
    // against a SQL Server database.
    myCommand = new SqlCommand(Sql, oConn);

    // Open the connection to the database

```

```

oConn.Open();

// Instantiate an instance of the SqlDataAdapter class
// The SqlDataAdapter, serves as a bridge between a DataSet and SQL Server
// for retrieving and saving data. The SqlDataAdapter provides this bridge
// by mapping Fill, which changes the data in the DataSet to match the data
// in the data source, and Update, which changes the data in the data source
// to match the data in the DataSet, using the appropriate Transact-SQL
// statements against the data source.
sda = new SqlDataAdapter();
// Set the value of the SDA select command equal to the command object.
// SelectCommand gets or sets a SQL statement used to select records in the data source.
sda.SelectCommand = myCommand;

// Instantiate an instance of the DataSet class which represents
// an in-memory cache of data.
ds = new DataSet();
// When the SqlDataAdapter fills a DataSet, it will create the necessary
// tables and columns for the returned data if they do not already exist.
sda.Fill(ds);

// Instantiate an instance of the DataTable class which represents
// one table of in-memory data. Additional Info: To add tables to the
// collection, use Add method of the DataTableCollection. To remove
// tables, use the Remove method.
dt = new DataTable();
// Get the first array index of the collection of tables contained in the DataSet
dt = ds.Tables[0];

// If a record exists, Do the delete
if(dt.Rows[0][0].ToString() != "")
{
    // Create the SQL statement
    Sql = "DELETE FROM Products_Lab6 WHERE ProductID=" + dt.Rows[0][0].ToString();
    // Create a new SQL command using the SQL command and connection object
    myCommand = new SqlCommand(Sql, oConn);

    // Execute the SQL statement against the connection. ExecuteNonQuery will
    // return the number of rows affected, but does not return a recordset.
    myCommand.ExecuteNonQuery();
}
// Close the connection to the database
oConn.Close();

// Call the Select_Click() function to redisplay the data in the browser.
Select_Click(Source, E);
}

private void Page_Load(object sender, System.EventArgs e)
{
    // Put user code to initialize the page here
    if(!Page.IsPostBack)
    {
        //do nothing
    }
}

#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    //
    // CODEGEN: This call is required by the ASP.NET Web Form Designer.
    //
    InitializeComponent();
    base.OnInit(e);
}

```

```
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#endregion
}
```

Lab06params.aspx

Select All	Select 1	Insert	Update	Delete	Delete All	Truncate		
ProductID	title	authors	copyrightDate	edition	isbn	coverart	description	price
7	Microsoft Visual	Jon Jagger	2003	2003	0735619093	Microsoft	Teach yourself	26.39
8	Microsoft Visual	Jon Jagger	2003	2003	0735619093	Microsoft	Teach yourself	26.39
9	New Title Value	Jon Jagger	2003	2003	0735619093	Microsoft	Teach yourself	26.39

Lab06params.aspx

```
<%@ Page language="c#" Codebehind="Lab06params.aspx.cs" AutoEventWireup="false" Inherits="Lab06.Lab06params" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
  <HEAD>
    <title>Lab06params</title>
    <meta name="CODE_LANGUAGE" Content="C#">
    <meta name="vs_defaultClientScript" content="JavaScript">
    <meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5">
  </HEAD>
  <body MS_POSITIONING="GridLayout">
    <form id="Form1" method="post" runat="server">
      <asp:Button Runat="server" ID="select" Text="Select All" OnClick="Select_Click" />
      <asp:Button Runat="server" ID="selectOne" Text="Select 1" OnClick="SelectOne_Click" />
      <asp:Button Runat="server" ID="insert" Text="Insert" OnClick="Insert_Click" />
      <asp:Button Runat="server" ID="update" Text="Update" OnClick="Update_Click" />
      <asp:Button Runat="server" ID="delete" Text="Delete" OnClick="Delete_Click" />
      <asp:Button Runat="server" ID="deleteAll" Text="Delete All" OnClick="DeleteAll_Click" />
      <asp:Button Runat="server" ID="truncate" Text="Truncate" OnClick="Truncate_Click" />

      <!-- A DataGrid is a data bound list control that displays the items from a
      data source in a table. Use the DataGrid control to display the fields of a
      data source as columns in a table. Each row in the DataGrid control represents
      a record in the data source. The DataGrid control supports selection, editing,
      deleting, paging, and sorting. -->
      <asp:DataGrid Runat="server" ID="results" />
    </form>
  </body>
</HTML>
```


Lab06params.aspx.cs

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

namespace Lab06
{
    /// <summary>
    /// //////////////////////////////////////
    ///     Author: Ronald J. Glotzbach
    ///
    ///     Date: February 18, 2005
    ///
    ///     Project: CGT 456 Lab06 Solution B using Parameters
    /// //////////////////////////////////////
    /// Summary description for Lab06params class.
    /// Lab06params uses the Parameters property of the SqlCommand class to
    /// allow parameters (or placeholders) to appear in SQL statements and
    /// the values of those placeholders to be passed in later.
    /// </summary>
    public class Lab06params : System.Web.UI.Page
    {
        // Declare global variables
        // Pull controls over from aspx
        protected System.Web.UI.WebControls.DataGrid results;
        protected System.Web.UI.WebControls.Button select;
        protected System.Web.UI.WebControls.Button selectOne;
        protected System.Web.UI.WebControls.Button insert;
        protected System.Web.UI.WebControls.Button update;
        protected System.Web.UI.WebControls.Button delete;
        protected System.Web.UI.WebControls.Button deleteAll;
        protected System.Web.UI.WebControls.Button truncate;
        // Declare global objects to be used
        public String ConnStr = "server=ipaddy;uid=xxx;pwd=yyy;database=zzz";
        public SqlConnection oConn;
        public SqlCommand myCommand;
        public SqlDataReader myReader;
        public SqlDataAdapter sda;
        public DataSet ds;
        public DataTable dt;
        public String Sql;

        /// <summary>
        /// function Select_Click()
        /// When the Select All Button is clicked, all of the records will be
        /// selected and binded to the DataGrid.
        /// </summary>
        /// <param name="Source"></param>
        /// <param name="E"></param>
        protected void Select_Click(Object Source, EventArgs E)
        {
            // Create the SQL statement
            Sql = "SELECT * FROM Products_Lab6";
            // Create a new connection object using the connection string above.
            // oConn represents an open connection to a SQL Server database.
            oConn = new SqlConnection(ConnStr);
            // Create a new SQL command using the SQL command and connection object
            myCommand = new SqlCommand(Sql, oConn);

            // Open the connection to the database
            oConn.Open();
        }
    }
}
```

```

// set the data source
results.DataSource = myCommand.ExecuteReader();

// bind the data source to the DataGrid
results.DataBind();

// Close the connection to the database
oConn.Close();
}

/// <summary>
/// function SelectOne_Click()
/// When the SelectOne Button is clicked, the record with the maximum
/// ProductID will be selected and binded to a DataGrid.
/// </summary>
/// <param name="Source"></param>
/// <param name="E"></param>
protected void SelectOne_Click(Object Source, EventArgs E)
{
    // Get the max id
    // Create the SQL statement
    Sql      = "SELECT Max(ProductID) AS MaxID FROM Products_Lab6";
    // Create a new connection object using the connection string above.
    // oConn represents an open connection to a SQL Server database.
    oConn     = new SqlConnection(ConnStr);
    // Create a new SQL command using the SQL command and connection object.
    // myCommand represents a SQL statement or stored procedure to execute
    // against a SQL Server database.
    myCommand = new SqlCommand(Sql, oConn);

    // Open the connection to the database
    oConn.Open();

    // Instantiate an instance of the SqlDataAdapter class
    // The SqlDataAdapter, serves as a bridge between a DataSet and SQL Server
    // for retrieving and saving data. The SqlDataAdapter provides this bridge
    // by mapping Fill, which changes the data in the DataSet to match the data
    // in the data source, and Update, which changes the data in the data source
    // to match the data in the DataSet, using the appropriate Transact-SQL
    // statements against the data source.
    sda = new SqlDataAdapter();
    // Set the value of the SDA select command equal to the command object.
    // SelectCommand gets or sets a SQL statement used to select records in the data source.
    sda.SelectCommand = myCommand;

    // Instantiate an instance of the DataSet class which represents
    // an in-memory cache of data.
    ds = new DataSet();
    // When the SqlDataAdapter fills a DataSet, it will create the necessary
    // tables and columns for the returned data if they do not already exist.
    sda.Fill(ds);

    // Instantiate an instance of the DataTable class which represents
    // one table of in-memory data. Additional Info: To add tables to the
    // collection, use Add method of the DataTableCollection. To remove
    // tables, use the Remove method.
    dt = new DataTable();
    // Get the first array index of the collection of tables contained in the DataSet
    dt = ds.Tables[0];

    // If a record exists, Do the select
    if(dt.Rows[0][0].ToString() != "")
    {
        // Create the SQL statement
        // The @prodID represents a parameter that serves as a placeholder thats value

```

```

        // can be passed into the SQL statement later.
        Sql = "SELECT * FROM Products_Lab6 WHERE ProductID = @prodID";
        // Create a new SQL command using the SQL command and connection object
        myCommand = new SqlCommand(Sql, oConn);

        // Parameters gets the SqlParameterCollection.
        // Add a new parameter matching the parameter(s) used in the above SQL statement
        myCommand.Parameters.Add(new SqlParameter("@prodID", System.Data.SqlDbType.Int, 4));
        // Assign a value to the @prodID placeholder in the above SQL statement
        myCommand.Parameters["@prodID"].Value = dt.Rows[0][0].ToString();

        // set the data source
        results.DataSource = myCommand.ExecuteReader();

        // bind the data source to the DataGrid
        results.DataBind();
    }
    // Close the connection to the database
    oConn.Close();
}

/// <summary>
/// function Insert_Click()
/// When the Insert Button is clicked, a new record will be inserted
/// into the database. The Primary Key is set to IDENTITY, so it will
/// auto-increment.
/// </summary>
/// <param name="Source"></param>
/// <param name="E"></param>
protected void Insert_Click(Object Source, EventArgs E)
{
    // Create the SQL statement
    // The @xxxx represents parameters that serve as a placeholders that's value
    // can be passed into the SQL statement later.
    Sql = "INSERT INTO Products_Lab6 (title, authors, copyrightDate, edition, ";
    Sql += "isbn, coverart, description, price) VALUES(@title, @authors, ";
    Sql += "@copyrightDate, @edition, @isbn, @coverart, @description, @price)";
    // Create a new connection object using the connection string above.
    // oConn represents an open connection to a SQL Server database.
    oConn = new SqlConnection(ConnStr);
    // Create a new SQL command using the SQL command and connection object.
    // myCommand represents a SQL statement or stored procedure to execute
    // against a SQL Server database.
    myCommand = new SqlCommand(Sql, oConn);

    // Parameters gets the SqlParameterCollection.
    // Add a new parameter matching the parameter(s) used in the above SQL statement
    myCommand.Parameters.Add(new SqlParameter("@title", System.Data.SqlDbType.VarChar, 100));
    myCommand.Parameters.Add(new SqlParameter("@authors", System.Data.SqlDbType.VarChar, 100));
    myCommand.Parameters.Add(new SqlParameter("@copyrightDate", System.Data.SqlDbType.VarChar, 6));
    myCommand.Parameters.Add(new SqlParameter("@edition", System.Data.SqlDbType.VarChar, 6));
    myCommand.Parameters.Add(new SqlParameter("@isbn", System.Data.SqlDbType.VarChar, 25));
    myCommand.Parameters.Add(new SqlParameter("@coverart", System.Data.SqlDbType.VarChar, 100));
    myCommand.Parameters.Add(new SqlParameter("@description", System.Data.SqlDbType.VarChar, 100));
    myCommand.Parameters.Add(new SqlParameter("@price", System.Data.SqlDbType.VarChar, 100));
    // Assign a value to the placeholders in the above SQL statement
    myCommand.Parameters["@title"].Value = "Microsoft Visual";
    myCommand.Parameters["@authors"].Value = "Jon Jagger";
    myCommand.Parameters["@copyrightDate"].Value = "2003";
    myCommand.Parameters["@edition"].Value = "2003";
    myCommand.Parameters["@isbn"].Value = "0735619093";
    myCommand.Parameters["@coverart"].Value = "Microsoft";
    myCommand.Parameters["@description"].Value = "Teach yourself";
    myCommand.Parameters["@price"].Value = "26.39";
}

```

```

// Open the connection to the database
oConn.Open();
// Execute the SQL statement against the connection. ExecuteNonQuery will
// return the number of rows affected, but does not return a recordset.
myCommand.ExecuteNonQuery();
// Close the connection to the database
oConn.Close();

// Call the Select_Click() function to redisplay the data in the browser.
Select_Click(Source, E);
}

/// <summary>
/// function Update_Click()
/// When the Update Button is clicked, the record with the maximum
/// ProductID is updated with a different Title.
/// </summary>
/// <param name="Source"></param>
/// <param name="E"></param>
protected void Update_Click(Object Source, EventArgs E)
{
    // Get the max id
    // Create the SQL statement
    Sql      = "SELECT Max(ProductID) AS MaxID FROM Products_Lab6";
    // Create a new connection object using the connection string above.
    // oConn represents an open connection to a SQL Server database.
    oConn     = new SqlConnection(ConnStr);
    // Create a new SQL command using the SQL command and connection object.
    // myCommand represents a SQL statement or stored procedure to execute
    // against a SQL Server database.
    myCommand = new SqlCommand(Sql, oConn);

    // Open the connection to the database
    oConn.Open();

    // Instantiate an instance of the SqlDataAdapter class
    // The SqlDataAdapter, serves as a bridge between a DataSet and SQL Server
    // for retrieving and saving data. The SqlDataAdapter provides this bridge
    // by mapping Fill, which changes the data in the DataSet to match the data
    // in the data source, and Update, which changes the data in the data source
    // to match the data in the DataSet, using the appropriate Transact-SQL
    // statements against the data source.
    sda = new SqlDataAdapter();
    // Set the value of the SDA select command equal to the command object.
    // SelectCommand gets or sets a SQL statement used to select records in the data source.
    sda.SelectCommand = myCommand;

    // Instantiate an instance of the DataSet class which represents
    // an in-memory cache of data.
    ds = new DataSet();
    // When the SqlDataAdapter fills a DataSet, it will create the necessary
    // tables and columns for the returned data if they do not already exist.
    sda.Fill(ds);

    // Instantiate an instance of the DataTable class which represents
    // one table of in-memory data. Additional Info: To add tables to the
    // collection, use Add method of the DataTableCollection. To remove
    // tables, use the Remove method.
    dt = new DataTable();
    // Get the first array index of the collection of tables contained in the DataSet
    dt = ds.Tables[0];

    // If a record exists, Do the update
    if(dt.Rows[0][0].ToString() != "")
    {
        // Create the SQL statement

```

```

// The @prodID represents a parameter that serves as a placeholder that's value
// can be passed into the SQL statement later.
Sql      = "UPDATE Products_Lab6 SET Title=@title WHERE ProductID=@prodID";
// Create a new SQL command using the SQL command and connection object
myCommand = new SqlCommand(Sql, oConn);

// Parameters gets the SqlParameterCollection.
// Add a new parameter matching the parameter(s) used in the above SQL statement
myCommand.Parameters.Add(new SqlParameter("@title", System.Data.SqlDbType.VarChar, 100));
myCommand.Parameters.Add(new SqlParameter("@prodID", System.Data.SqlDbType.Int, 4));
// Assign a values to the placeholders in the above SQL statement
myCommand.Parameters["@title"].Value = "New Title Value";
myCommand.Parameters["@prodID"].Value = dt.Rows[0][0].ToString();

// Execute the SQL statement against the connection. ExecuteNonQuery will
// return the number of rows affected, but does not return a recordset.
myCommand.ExecuteNonQuery();
}
// Close the connection to the database
oConn.Close();

// Call the Select_Click() function to redisplay the data in the browser.
Select_Click(Source, E);
}

/// <summary>
/// function Delete_Click()
/// When the Delete Button is clicked, the record with the maximum
/// ProductID is deleted from the database.
/// </summary>
/// <param name="Source"></param>
/// <param name="E"></param>
protected void Delete_Click(Object Source, EventArgs E)
{
    // Get the max id
    // Create the SQL statement
    Sql      = "SELECT Max(ProductID) AS MaxID FROM Products_Lab6";
    // Create a new connection object using the connection string above.
    // oConn represents an open connection to a SQL Server database.
    oConn    = new SqlConnection(ConnStr);
    // Create a new SQL command using the SQL command and connection object.
    // myCommand represents a SQL statement or stored procedure to execute
    // against a SQL Server database.
    myCommand = new SqlCommand(Sql, oConn);

    // Open the connection to the database
    oConn.Open();

    // Instantiate an instance of the SqlDataAdapter class
    // The SqlDataAdapter, serves as a bridge between a DataSet and SQL Server
    // for retrieving and saving data. The SqlDataAdapter provides this bridge
    // by mapping Fill, which changes the data in the DataSet to match the data
    // in the data source, and Update, which changes the data in the data source
    // to match the data in the DataSet, using the appropriate Transact-SQL
    // statements against the data source.
    sda = new SqlDataAdapter();
    // Set the value of the SDA select command equal to the command object.
    // SelectCommand gets or sets a SQL statement used to select records in the data source.
    sda.SelectCommand = myCommand;

    // Instantiate an instance of the DataSet class which represents
    // an in-memory cache of data.
    ds = new DataSet();
    // When the SqlDataAdapter fills a DataSet, it will create the necessary
    // tables and columns for the returned data if they do not already exist.
    sda.Fill(ds);
}

```

```

// Instantiate an instance of the DataTable class which represents
// one table of in-memory data. Additional Info: To add tables to the
// collection, use Add method of the DataTableCollection. To remove
// tables, use the Remove method.
dt = new DataTable();
// Get the first array index of the collection of tables contained in the DataSet
dt = ds.Tables[0];

// If there is a record to delete, Do the delete
if(dt.Rows[0][0].ToString() != "")
{
    // Create the SQL statement
    // The @prodID represents a parameter that serves as a placeholder thats value
    // can be passed into the SQL statement later.
    Sql      = "DELETE FROM Products_Lab6 WHERE ProductID=@prodID";
    // Create a new SQL command using the SQL command and connection object
    myCommand = new SqlCommand(Sql, oConn);

    // Parameters gets the SqlParameterCollection.
    // Add a new parameter matching the parameter(s) used in the above SQL statement
    myCommand.Parameters.Add(new SqlParameter("@prodID", System.Data.SqlDbType.Int, 4));
    // Assign a values to the placeholders in the above SQL statement
    myCommand.Parameters["@prodID"].Value = dt.Rows[0][0].ToString();

    // Execute the SQL statement against the connection. ExecuteNonQuery will
    // return the number of rows affected, but does not return a recordset.
    myCommand.ExecuteNonQuery();
}
// Close the connection to the database
oConn.Close();

// Call the Select_Click() function to redisplay the data in the browser.
Select_Click(Source, E);
}

/// <summary>
/// function DeleteAll_Click()
/// When the Delete All Button is clicked, all of the records will be
/// deleted from the database.
/// </summary>
/// <param name="Source"></param>
/// <param name="E"></param>
protected void DeleteAll_Click(Object Source, EventArgs E)
{
    // Create the SQL statement
    Sql      = "DELETE FROM Products_Lab6";
    // Create a new connection object using the connection string above.
    // oConn represents an open connection to a SQL Server database.
    oConn    = new SqlConnection(ConnStr);
    // Create a new SQL command using the SQL command and connection object
    myCommand = new SqlCommand(Sql, oConn);

    // Open the connection to the database
    oConn.Open();
    // Execute the SQL statement against the connection. ExecuteNonQuery will
    // return the number of rows affected, but does not return a recordset.
    myCommand.ExecuteNonQuery();
    // Close the connection to the database
    oConn.Close();

    // Call the Select_Click() function to redisplay the data in the browser.
    Select_Click(Source, E);
}

/// <summary>

```

```

/// function Truncate_Click()
/// When the Truncate Button is clicked, all of the records will be
/// deleted from the database and the table will be truncated. The
/// reason for this example is to show how TRUNCATE resets the
/// IDENTITY counter in SQL Server.
/// </summary>
/// <param name="Source"></param>
/// <param name="E"></param>
protected void Truncate_Click(Object Source, EventArgs E)
{
    // Create the SQL statement
    Sql      = "TRUNCATE TABLE Products_Lab6";
    // Create a new connection object using the connection string above.
    // oConn represents an open connection to a SQL Server database.
    oConn     = new SqlConnection(ConnStr);
    // Create a new SQL command using the SQL command and connection object
    myCommand = new SqlCommand(Sql, oConn);

    // Open the connection to the database
    oConn.Open();
    // Execute the SQL statement against the connection. ExecuteNonQuery will
    // return the number of rows affected, but does not return a recordset.
    myCommand.ExecuteNonQuery();
    // Close the connection to the database
    oConn.Close();

    // Call the Select_Click() function to redisplay the data in the browser.
    Select_Click(Source, E);
}

private void Page_Load(object sender, System.EventArgs e)
{
    // Put user code to initialize the page here
}

#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    //
    // CODEGEN: This call is required by the ASP.NET Web Form Designer.
    //
    InitializeComponent();
    base.OnInit(e);
}

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#endregion
}
}

```