

```

using System;
using System.Data;
using System.Data.SqlClient;
using CustomersBus;

namespace CustomersData
{
    /// <summary>
    /// CustData Class designed to handle
    /// all database activities for Customers table.
    /// Author: Doug Streitenberger
    /// </summary>
    public class CustData
    {
        /// <summary>
        /// Default Instantiation of Class
        /// </summary>
        public CustData()
        {

        }

        /// <summary>
        /// Populates a DataTable with all Customer Information
        /// Author: Doug Streitenberger
        /// </summary>
        /// <returns>DataTable of all Customer Information.</returns>
        public static DataTable CustomersEnum()
        {
            using (SqlConnection cn = new SqlConnection(GetConnectionString()))
            {
                using (SqlCommand cmd = new SqlCommand("spCustomersEnum", cn))
                {
                    cmd.CommandType = CommandType.StoredProcedure;

                    SqlDataAdapter da = new SqlDataAdapter(cmd);
                    DataTable dt = new DataTable();

                    da.Fill(dt);
                    return dt;
                }
            }
        }

        /// <summary>
        /// Populates Customer Object with Customer Information based on param CustomerId.
        /// Author: Doug Streitenberger

```

```

/// </summary>
/// <param name="CustomerId">Requested CustomerId.</param>
/// <returns>Customer Object</returns>
public static Customer CustomersEnumByID(string CustomerId)
{
    using (SqlConnection cn = new SqlConnection(GetConnectionString()))
    {
        using (SqlCommand cmd = new SqlCommand("spCustomersEnumByID", cn))
        {
            cmd.CommandType = CommandType.StoredProcedure;

            SqlParameter prmsCustomerID = cmd.Parameters.Add("@sCustomerID", SqlDbType.VarChar, 5);
            prmsCustomerID.Value = CustomerId;

            cn.Open();
            SqlDataReader rdr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
            if (rdr.Read())
            {
                Customer customer = new Customer();
                customer.CustomerId = CustomerId;
                customer.CompanyName = rdr["CompanyName"].ToString();
                customer.ContactName = rdr["ContactName"].ToString();
                customer.PhoneNumber = rdr["Phone"].ToString();
                customer.FaxNumber = rdr["Fax"].ToString();
                customer.ContactTitle = rdr["ContactTitle"].ToString();
                customer.Address = rdr["Address"].ToString();
                customer.City = rdr["City"].ToString();
                customer.Region = rdr["Region"].ToString();
                customer.PostalCode = rdr["PostalCode"].ToString();
                customer.Country = rdr["Country"].ToString();
                try
                {
                    customer.LastUpdated = Convert.ToDateTime(rdr["Last_Updated"].ToString());
                }
                catch
                {
                    customer.LastUpdated = DateTime.Today;
                }
                return customer;
            }
            rdr.Close();
        }
    }
    return null;
}

/// <summary>

```

```

/// Determines if CustomerID already exists in the Customers table based on param CustomerId.
/// Author: Doug Streitenberger
/// </summary>
/// <param name="CustomerId">Requested CustomerId</param>
/// <returns>Boolean whether CustomerID already exists in the database.</returns>
public static bool UniqueCustomerID(string CustomerId)
{
    bool bUnique = false;
    using (SqlConnection cn = new SqlConnection(GetConnectionString()))
    {
        using (SqlCommand cmd = new SqlCommand("spCustomersEnumByID", cn))
        {
            cmd.CommandType = CommandType.StoredProcedure;

            SqlParameter prmsCustomerID = cmd.Parameters.Add("@sCustomerID", SqlDbType.VarChar, 5);
            prmsCustomerID.Value = CustomerId;

            cn.Open();
            SqlDataReader rdr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
            if (rdr.Read())
            {
                bUnique = true;
            }
            rdr.Close();
        }
    }
    return bUnique;
}

/// <summary>
/// Populates a DataTable with all Customer Information based on search params entered.
/// Search Criteria are formatted such that if values are entered, the returned datatable
/// will be filtered according to those entries. Otherwise, all records will be returned.
/// Author: Doug Streitenberger
/// </summary>
/// <param name="CustomerId"></param>
/// <param name="CompanyName"></param>
/// <param name="ContactName"></param>
/// <returns>DataTable of all Customer Information.</returns>
public static DataTable CustomerSearchEnum(string CustomerId, string CompanyName, string ContactName)
{
    CustomerId = CustomerId.Trim();
    CompanyName = CompanyName.Trim();
    ContactName = ContactName.Trim();

    using (SqlConnection cn = new SqlConnection(GetConnectionString()))
    {

```

```

using (SqlCommand cmd = new SqlCommand("spCustomersEnumSearch", cn))
{
    cmd.CommandType = CommandType.StoredProcedure;

    SqlParameter prmsCustomerID = cmd.Parameters.Add("@sCustomerID", SqlDbType.VarChar, 6);
    prmsCustomerID.Value = FormatSearchCriteria(CustomerId);

    SqlParameter prmsCompanyName = cmd.Parameters.Add("@sCompanyName", SqlDbType.VarChar, 40);
    prmsCompanyName.Value = FormatSearchCriteria(CompanyName);

    SqlParameter prmsContactName = cmd.Parameters.Add("@sContactName", SqlDbType.VarChar, 30);
    prmsContactName.Value = FormatSearchCriteria(ContactName);

    SqlDataAdapter da = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();

    da.Fill(dt);
    return dt;
}
}

/// <summary>
/// If an entry is made to a search criteria, the search criteria
/// will be concatenated as follows: '%SearchCriteria%'
/// (a % on each end of the string).
/// Otherwise an '%' is returned for that search criteria.
/// Author: Doug Streitenberger
/// </summary>
/// <param name="SearchCriteria">Search Criteria as entered by the user.</param>
/// <returns>Formatted Search Criteria String.</returns>
private static string FormatSearchCriteria(string SearchCriteria)
{
    if (SearchCriteria.Length == 0)
    {
        SearchCriteria = "%";
    }
    else
    {
        SearchCriteria = "%" + SearchCriteria + "%";
    }
    return SearchCriteria;
}

/// <summary>
/// Populates a DataTable of the Top 5 Customers By Sales
/// based on Start And End Dates entered.

```

```

/// Author: Doug Streitenberger
/// </summary>
/// <returns>DataTable of Top 5 Customers</returns>
public static DataSet CustomersEnumTop5(string sStartDate, string sEndDate)
{
    using (SqlConnection cn = new SqlConnection(GetConnectionString()))
    {
        using (SqlCommand cmd = new SqlCommand("spCustomersEnumTop5", cn))
        {
            cmd.CommandType = CommandType.StoredProcedure;
            SqlParameter prmdtStartDate =
                cmd.Parameters.Add("@dtStartDate", SqlDbType.DateTime, 8);
            SqlParameter prmdtEndDate =
                cmd.Parameters.Add("@dtEndDate", SqlDbType.DateTime, 8);
            try
            {
                prmdtStartDate.Value = Convert.ToDateTime(sStartDate);
                prmdtEndDate.Value = Convert.ToDateTime(sEndDate);
            }
            catch { }
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();

            da.Fill(ds);
            return ds;
        }
    }
}
/// <summary>
/// Inserts Customer Record based on Customer object.
/// Author: Doug Streitenberger
/// </summary>
/// <param name="customer">Customer Object</param>
public static void CustomerAdd(Customer customer)
{
    using (SqlConnection cn = new SqlConnection(GetConnectionString()))
    {
        using (SqlCommand cmd = new SqlCommand("spCustomersInsert", cn))
        {
            cmd.CommandType = CommandType.StoredProcedure;

            SqlParameter prmsCustomerID = cmd.Parameters.Add("@sCustomerID", SqlDbType.VarChar, 5);
            prmsCustomerID.Value = customer.CustomerId;
            SqlParameter prmsCompanyName = cmd.Parameters.Add("@sCompanyName", SqlDbType.VarChar, 40);
            prmsCompanyName.Value = customer.CompanyName;
            SqlParameter prmsContactName = cmd.Parameters.Add("@sContactName", SqlDbType.VarChar, 30);
            prmsContactName.Value = customer.ContactName;
        }
    }
}

```

```

        SqlParameter prmsPhone = cmd.Parameters.Add("@sPhone", SqlDbType.VarChar, 24);
        prmsPhone.Value = customer.PhoneNumber;
        SqlParameter prmsFax = cmd.Parameters.Add("@sFax", SqlDbType.VarChar, 24);
        prmsFax.Value = customer.FaxNumber;

        cn.Open();
        cmd.ExecuteNonQuery();

    }
}

/// <summary>
/// Updates Customer Record based on Customer Object.
/// Author: Doug Streitenberger
/// </summary>
/// <param name="oldCustomer">Customer Object Before Update For Data Concurrency</param>
/// <param name="customer">Customer Object of Updated Data</param>
/// <returns>int iCount;
/// Should return value of '1' record updated.
/// If '99': Database Error, Data Concurrency Error.
/// </returns>

public static int CustomerUpdate(Customer oldCustomer, Customer customer)
{
    int iCount = 0;
    using (SqlConnection cn = new SqlConnection(GetConnectionString()))
    {
        using (SqlCommand cmd = new SqlCommand("spCustomersUpdate", cn))
        {
            cmd.CommandType = CommandType.StoredProcedure;

            SqlParameter prmsCustomerID = cmd.Parameters.Add("@sCustomerID", SqlDbType.VarChar, 5);
            prmsCustomerID.Value = customer.CustomerId;

            SqlParameter prmsOldCompanyName =
                cmd.Parameters.Add("@sOldCompanyName", SqlDbType.VarChar, 50);
            prmsOldCompanyName.Value = oldCustomer.CompanyName;
            SqlParameter prmsOldContactName =
                cmd.Parameters.Add("@sOldContactName", SqlDbType.VarChar, 50);
            prmsOldContactName.Value = oldCustomer.ContactName;
            SqlParameter prmsOldPhone = cmd.Parameters.Add("@sOldPhone", SqlDbType.VarChar, 50);
            prmsOldPhone.Value = oldCustomer.PhoneNumber;
            SqlParameter prmsOldFax = cmd.Parameters.Add("@sOldFax", SqlDbType.VarChar, 50);
            prmsOldFax.Value = oldCustomer.FaxNumber;

```

```

        SqlParameter prmsCompanyName = cmd.Parameters.Add("@sCompanyName", SqlDbType.VarChar, 50);
        prmsCompanyName.Value = customer.CompanyName;
        SqlParameter prmsContactName = cmd.Parameters.Add("@sContactName", SqlDbType.VarChar, 50);
        prmsContactName.Value = customer.ContactName;
        SqlParameter prmsPhone = cmd.Parameters.Add("@sPhone", SqlDbType.VarChar, 50);
        prmsPhone.Value = customer.PhoneNumber;
        SqlParameter prmsFax = cmd.Parameters.Add("@sFax", SqlDbType.VarChar, 50);
        prmsFax.Value = customer.FaxNumber;

        try
        {
            cn.Open();
            iCount = Convert.ToInt32(cmd.ExecuteScalar().ToString());
        }
        catch(Exception ex)
        {
            iCount = 99;

            CustDefects objDefects = new CustDefects();
            objDefects.InsertDefect("Customer Update Error", ex.ToString());
            objDefects = null;
        }
    }
}
return iCount;
}

/// <summary>
/// Deletes Customer record based upon CustomerId.
/// Author: Doug Streitenberger
/// </summary>
/// <param name="CustomerId">CustomerId of the selected record</param>
public static void CustomerDelete(string CustomerId)
{
    using (SqlConnection cn = new SqlConnection(GetConnectionString()))
    {
        using (SqlCommand cmd = new SqlCommand("spCustomersDelete", cn))
        {
            cmd.CommandType = CommandType.StoredProcedure;

            SqlParameter prmsCustomerID = cmd.Parameters.Add("@sCustomerID", SqlDbType.VarChar, 5);
            prmsCustomerID.Value = CustomerId;

            cn.Open();
            cmd.ExecuteNonQuery();
        }
    }
}

```

```
}  
  
private static string GetConnectionString()  
{  
    // Establish a database connection string for  
    // the CustData Class using the CustDataCommon Class.  
    CustDataCommon objDataCommon = new CustDataCommon();  
    string sConn = objDataCommon.sConnStr;  
    objDataCommon = null;  
    return sConn;  
}  
}
```