using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Xml;

using Laserfiche.DocumentServices;

using Laserfiche.RepositoryAccess;

using System.Data.Sql;

using System.Data.SqlClient;

using System.Data;

using System.Net.Mail;

using System.Globalization;

using System.Diagnostics;

namespace LFProcessPendingReceipts

{

    class Program

    {

        public static string gst { get; set; }

        public static string hst { get; set; }

        public static string qst { get; set; }

        public static string ost { get; set; }

        static void Main(string[] args)

        {

           Session MySession;

           MySession = new Session();

           GetTaxGLCodes();

            var repository = new RepositoryRegistration("vanlfi01", "\*\*\*\*\*\*");

            MySession.LogIn("LFWorkflow", "\*\*\*\*\*\*", repository);

            List<int> ilDocs = new List<int>();

            try

            {

                FolderInfo fi = Folder.GetFolderInfo(737757, MySession); // it

              //  FolderInfo fi = Folder.GetFolderInfo(724510, MySession); // Ops

             //   FolderInfo fi = Folder.GetFolderInfo(748033, MySession); // test folder

                EntryListingSettings els = new EntryListingSettings();

                els.AddColumn(SystemColumn.Id);

                FolderListing fl = fi.OpenFolderListing(els);

                if (fl.RowCount > 0)

                {

                    for (int iRow = 1; iRow <= fl.RowCount; iRow++)

                    {

                        if (fl.GetDatumAsString(iRow, SystemColumn.EntryType).Equals("Document"))

                        {

                            ilDocs.Add((int)fl.GetDatum(iRow, SystemColumn.Id));

                        }

                    }

                }

                fl.Close();

                fi.Dispose();

            }

            catch (Exception ex)

            {

                ilDocs = new List<int>();

            }

            for (int i = 0; i <= ilDocs.Count -1; i++)

            {

                CheckForReciepts(ilDocs[i], MySession);

            }

            MySession.LogOut();

        }

        private static void CheckForReciepts(Int32 docid, Session session)

        {

            DocumentInfo doc;

            string CRLF = "\r\n";

            doc = Document.GetDocumentInfo(docid, session);

            // check to see if Doc is Locked or checked out

            // if it is wait till next run

            if (doc.IsLocked || doc.IsCheckedOut) {

                return;

            }

            double InvoiceCalc = 0;

            string SqlId = String.Empty;

            FieldValueCollection fieldValues = doc.GetFieldValues();

            string Notes = "";

            SqlId = fieldValues["SqlId"].ToString();

            if (fieldValues["Reference Number"] != null)

            {

                UpdateRefNumber(fieldValues["Reference Number"].ToString(), SqlId.ToString());

            }

            if (fieldValues["Invoice Total"] != null)

            {

                if (IsNumeric(fieldValues["Invoice Total"].ToString()))

                {

                    InvoiceCalc = double.Parse(fieldValues["Invoice Total"].ToString());

                    UpdateInvoiceTotal(fieldValues["Invoice Total"].ToString(),SqlId);

                }

            }

            if (fieldValues["Pallet Charges"] != null)

            {

                if (IsNumeric(fieldValues["Pallet Charges"].ToString()))

                {

                    InvoiceCalc = InvoiceCalc - double.Parse(fieldValues["Pallet Charges"].ToString());

                    UpdatePalletCharges(fieldValues["Pallet Charges"].ToString(), SqlId);

                }

            }

            if (fieldValues["Freight"] != null)

            {

                if (IsNumeric(fieldValues["Freight"].ToString()))

                {

                    InvoiceCalc = InvoiceCalc - double.Parse(fieldValues["Freight"].ToString());

                    UpdateFreightCharges(fieldValues["Freight"].ToString(), SqlId);

                }

            }

            //////////if (fieldValues["GST"] != null)

            //////////{

            //////////    if (IsNumeric(fieldValues["GST"].ToString()))

            //////////    {

            //////////        InvoiceCalc = InvoiceCalc - Double.Parse(fieldValues["GST"].ToString());

            //////////        UpdateGst(fieldValues["GST"].ToString(), gst, fieldValues["SqlId"].ToString());

            //////////        fieldValues["GstHst"] = fieldValues["Gst"].ToString();

            //////////    }

            //////////}

            //////////if (fieldValues["HST"] != null)

            //////////{

            //////////    if (IsNumeric(fieldValues["HST"].ToString()))

            //////////    {

            //////////        InvoiceCalc = InvoiceCalc - Double.Parse(fieldValues["HST"].ToString());

            //////////        UpdateHst(fieldValues["HST"].ToString(), hst, fieldValues["SqlId"].ToString());

            //////////        fieldValues["GstHst"] = fieldValues["Hst"].ToString();

            //////////    }

            //////////}

            if (fieldValues["GstHst"] != null)

            {

                if (IsNumeric(fieldValues["GstHst"].ToString()))

                {

                    InvoiceCalc = InvoiceCalc - Double.Parse(fieldValues["GstHst"].ToString());

                    UpdateGstHst(fieldValues["GstHst"].ToString(), SqlId);

                }

            }

            if (fieldValues["Invoice Date"] != null)

            {

                if (IsDate(fieldValues["Invoice Date"]))

                {

                    UpdateInvoiceDate(fieldValues["Invoice Date"].ToString(), fieldValues["SqlId"].ToString());

                }

            }

                object QstAmount = "";

                bool FoundQst = fieldValues.TryGetValue("QST", out QstAmount);

                if (FoundQst)

                {

                    if (IsNumeric(FoundQst.ToString()))

                    {

                        InvoiceCalc = InvoiceCalc - Double.Parse(FoundQst.ToString());

                        UpdateQst(FoundQst.ToString(), gst, fieldValues["SqlId"].ToString());

                    }

                }

                string PalletCharges = fieldValues["Pallet Charges"] != null  ? fieldValues["Pallet Charges"].ToString() : "0";

                string freight = fieldValues["freight"] != null ? fieldValues["freight"].ToString() : "0";

                double GstHst = fieldValues["GstHst"] != null ? double.Parse(fieldValues["GstHst"].ToString()) : 0;

                string Total = fieldValues["Invoice Total"] != null ? fieldValues["Invoice Total"].ToString() : "0";

             double CodedAmount = MakeCodedAmount(IsNumeric(PalletCharges) ? double.Parse(PalletCharges) : 0, IsNumeric(freight) ? double.Parse(freight) : 0, GstHst > 0 ? double.Parse(GstHst.ToString()) : 0, IsNumeric(Total.Replace(",", "")) ? double.Parse(Total.Replace(",", "")) : 0);

             fieldValues["Coded Amount"] = CodedAmount.ToString();

            //move notes data

             if (fieldValues["Notes"] != null)

             {

                 StringBuilder sb = new StringBuilder();

                 if (fieldValues["Notes-Log"] != null)

                 {

                     if ((fieldValues["Notes"].ToString().Length + fieldValues["Notes-Log"].ToString().Length) < 4000)

                     {

                         sb.Append(DateTime.Now.ToShortDateString() + ": " + (fieldValues["Notes"] == null ? "" : fieldValues["Notes"].ToString()) + CRLF + fieldValues["Notes-Log"].ToString() + CRLF);

                     }

                     else

                     {

                         string str = fieldValues["Notes"].ToString().Remove(0, 10);

                         sb.Append(DateTime.Now.ToShortDateString() + ": " + fieldValues["Notes"].ToString() + CRLF + str.ToString() + CRLF);

                     }

                     fieldValues["Notes-Log"] = sb.ToString();

                 }

                 fieldValues["Notes"] = "";

             }

             //Check for GLCoding, if found send it

            if(fieldValues["Users GL Coding"] != null )

            {

                string userglfield = fieldValues["Users GL Coding"].ToString();

                if (!userglfield.ToLower().Contains("billback")){

                   string userCoding = "";

                    string ProjectCode = "0000";   //default

                    userCoding = fieldValues["Users GL Coding"].ToString().Substring(0,fieldValues["Users GL Coding"].ToString().IndexOf("|") -1);

                    if(userCoding.StartsWith("\_")){userCoding.Remove(0,1);}

                    if(userCoding.StartsWith("0\_")){userCoding.Remove(0,2);}

                    if (fieldValues["Users Project Codes"] != null)

                    {

                        ProjectCode = fieldValues["Users Project Codes"].ToString().Substring(0, fieldValues["Users Project Codes"].ToString().IndexOf("|") - 1);

                    }

                    fieldValues["GLCoding"] = CodedAmount.ToString() + "\_" + userCoding + "\_" + ProjectCode;

                    UpdateGLCoding(CodedAmount.ToString() + "\_" + userCoding + "\_" + ProjectCode, SqlId);

                    doc.SetFieldValues(fieldValues);

                    doc.Save();

                    doc.MoveTo(@"MagInvoicing\AccountsPayable\Vouchering\" + doc.Name, EntryNameOption.AutoRename);

                    SetToVouchering(100, SqlId);

                    doc.SetFieldValues(fieldValues);

                    doc.Save();

                    doc.Unlock();

                    return;

                }

            }

            if (fieldValues["GLCoding"] != null)

            {

                if (fieldValues["GLCoding"].ToString() != "")

                {

                    UpdateGLCoding(fieldValues["GLCoding"].ToString(), SqlId);

                    doc.MoveTo(@"MagInvoicing\AccountsPayable\Vouchering\" + doc.Name, EntryNameOption.AutoRename);

                    SetToVouchering(100, SqlId);

                    doc.SetFieldValues(fieldValues);

                    doc.Save();

                    doc.Dispose();

                    return;

                }

            }

            try

            {

                DataTable dt = GetReceiptData(SqlId);

                string tempNotesLog = string.Empty;

                if (fieldValues["Notes-Log"] != null)

                {

                    tempNotesLog = CleanNotesField(fieldValues["Notes-Log"].ToString());

                }

                if (fieldValues["Notes"] != null)

                {

                    Notes = fieldValues["Notes"].ToString();

                    fieldValues["Notes"] = "";

                }

                if (dt.Rows.Count > 0)

                {

                    fieldValues["Receiver Number"] = dt.Rows[0]["porct\_receiver"].ToString();

                    fieldValues["Notes-Log"] = "Receiver Found - " + DateTime.Now.ToShortDateString() + CRLF + Notes + CRLF + tempNotesLog + CRLF;

                    SetToVouchering(100, SqlId);

                    doc.MoveTo(@"MagInvoicing\AccountsPayable\Vouchering\" + doc.Name, EntryNameOption.AutoRename);

                    SetToVouchering(100, SqlId);

                }

                else

                {

                    if (tempNotesLog.IndexOf("Receiver Not Found - " + DateTime.Now.ToShortDateString()) != -1)

                    {

                    }

                    else

                    {

                        fieldValues["Notes-Log"] = "Receiver Not Found - " + DateTime.Now.ToShortDateString() + CRLF + Notes + CRLF + tempNotesLog + CRLF;

                    }

                    fieldValues["Receiver Number"] = "";  //clear out any manual data

                    SetToVouchering(90, SqlId);

                    //For Ops flow

                    //if (fieldValues["Invoice Date"] != null)

                    //{

                    //    try

                    //    {

                    //        DateTime InvDate = Convert.ToDateTime(fieldValues["Invoice Date"].ToString());

                    //        if (InvDate.AddDays(7) < DateTime.Today)

                    //        {

                    //            doc.MoveTo(@"MagInvoicing\OPS\ExpiringInvoices\" + doc.Name, EntryNameOption.AutoRename);

                    //        }

                    //    }

                    //    catch (Exception ex)

                    //    {

                    //    }

                    //}

                   // For It flow

                    DateTime docdate = doc.LastModifiedTime;

                    if (docdate.Date < DateTime.Today)

                    {

                        doc.MoveTo(@"MagInvoicing\IT\ITPostOffice\" + doc.Name, EntryNameOption.AutoRename);

                        //  doc.MoveTo(@"MagInvoicing\AccountsPayable\Failed" + doc.Name, EntryNameOption.AutoRename);

                    }

                }

            }

            catch (Exception er)

            {

            }

            doc.SetFieldValues(fieldValues);

            doc.Save();

            doc.Dispose();

        }

        static double MakeCodedAmount(double palletcharges, double freight, double HstGst, double InvoiceTotal)

        {

            double codeamount = 0;

            double subtotal = palletcharges + freight + HstGst;

            codeamount = (InvoiceTotal - subtotal);

            return codeamount;

        }

        static string CleanNotesField(string Notes)

        {

            string[] lines = Notes.Split(new string[] { Environment.NewLine }, StringSplitOptions.None);

            string ret = string.Empty;

            for (int i = 0; i <= lines.Length - 1; i++)

            {

               // Debug.WriteLine(lines[i].ToString());

                if (!lines[i].ToString().StartsWith("Admin On") || lines[i].ToString().StartsWith("Reciever Not Found "))

                {

                    ret += lines[i].ToString();

                }

            }

            return ret.ToString();

        }

        static void UpdateInvoiceDate(string InvoiceDate, string SqlId)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("update kofaxHeader set InvoiceDate = @InvoiceDate where id = @SqlId", con);

            cmd.CommandType = CommandType.Text;

            cmd.Parameters.AddWithValue("@SqlId", SqlId);

            cmd.Parameters.AddWithValue("@InvoiceDate", Convert.ToDateTime(InvoiceDate).ToShortDateString());

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

        static void UpdateGLCoding(string GLCoding, string SqlId)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("update kofaxHeader set kofus\_gl\_coding = @kofus\_gl\_coding where id = @SqlId", con);

            cmd.CommandType = CommandType.Text;

            cmd.Parameters.AddWithValue("@SqlId", SqlId);

            cmd.Parameters.AddWithValue("@kofus\_gl\_coding", GLCoding);

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

        static void SetToVouchering(Int32 status, string Id)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("update kofaxHeader set status = @status where id = @id", con);

            cmd.CommandType = CommandType.Text;

            //cmd.Parameters.AddWithValue("@voucher", voucher.Trim());

            //cmd.Parameters.AddWithValue("@Cheque", cheque.Trim());

            //cmd.Parameters.AddWithValue("@BankCode", BankCode.Trim());

            cmd.Parameters.AddWithValue("@Id", Id);

            cmd.Parameters.AddWithValue("@status", status);

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

        private static Boolean CheckForExpiring(string dateValue)

        {

            Boolean tf = false;

            DateTime InvocieDate = Convert.ToDateTime(dateValue);

            if (InvocieDate.AddDays(7) < DateTime.Now)

                {

                    tf = true;

                }

            else

                //  Console.WriteLine("Unable to convert '{0}' to a date and time.", dateValue);

                tf = false;

                return tf;

        }

        public static bool IsNumeric(string s)

        {

            float output;

            return float.TryParse(s, out output);

        }

        public static bool IsDate(Object obj)

        {

            string strDate = obj.ToString();

            try

            {

                DateTime dt = DateTime.Parse(strDate);

                if (dt != DateTime.MinValue && dt != DateTime.MaxValue)

                    return true;

                return false;

            }

            catch

            {

                return false;

            }

        }

        private static void GetTaxGLCodes()

        {

            string srConnectionString = "server=vansql07;database=kofaxdata;uid=magappuser;pwd=Ivvav!kMes@ch1e;";

            string strQuery = "SELECT [hst],[qst],gst, [ProvStateID] FROM [vTaxesByProv]  where [ProvStateID] = 11";

            SqlDataAdapter DA = new SqlDataAdapter();

            DataTable DT = new DataTable("RecData");

            DataSet dSet = new DataSet();

            SqlCommand cmd = new SqlCommand(strQuery);

            DA.SelectCommand = cmd;

            try

            {

                using (SqlConnection connection = new SqlConnection(srConnectionString))

                {

                    cmd.Connection = connection;

                    cmd.CommandType = CommandType.Text;

                    DA.Fill(DT);

                    if (DT.Rows.Count == 1)

                    {

                        gst = DT.Rows[0]["gst"].ToString();

                        hst = DT.Rows[0]["hst"].ToString();

                        ost = DT.Rows[0]["ost"].ToString();

                        qst = DT.Rows[0]["qst"].ToString();

                    }

                    connection.Close();

                }

            }

            catch (Exception ex)

            {

            }

        }

        private static DataTable GetReceiptData(string sqlid)

        {

            string srConnectionString = "server=vansql07;database=kofaxdata;uid=magappuser;pwd=Ivvav!kMes@ch1e;";

            string strQuery = "spGetPoNumFromRct";

            SqlDataAdapter DA = new SqlDataAdapter();

            DataTable DT = new DataTable("RecData");

            DataSet dSet = new DataSet();

            SqlCommand cmd = new SqlCommand(strQuery);

            DA.SelectCommand = cmd;

            cmd.Parameters.AddWithValue("@SqlId", sqlid);

            try

            {

                using (SqlConnection connection = new SqlConnection(srConnectionString))

                {

                    cmd.Connection = connection;

                    cmd.CommandType = CommandType.StoredProcedure;

                    DA.Fill(DT);

                    connection.Close();

                }

            }

            catch (Exception ex)

            {

            }

            return DT;

        }

   #region Update Taxes

        #endregion

        #region HST

        static void UpdateHst(string HstAmount, string HstCodeing, string SqlId)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("spUpdateTaxes", con);

            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@Mode", 1);

            cmd.Parameters.AddWithValue("@SqlId", SqlId);

            cmd.Parameters.AddWithValue("@HstAmount", HstAmount);

            cmd.Parameters.AddWithValue("@HstCoding", HstCodeing);

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

        #endregion

        #region GST

        static void UpdateGst(string Gst, string GstCodeing, string SqlId)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("spUpdateTaxes", con);

            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@Mode", 1);

            cmd.Parameters.AddWithValue("@SqlId", SqlId);

            cmd.Parameters.AddWithValue("@GstAmount", Gst);

            cmd.Parameters.AddWithValue("@GstCoding", GstCodeing);

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

        static void UpdatePalletCharges(string PalletCharges, string SqlId)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("Update kofax KofaxHeader set PalletCharges = @PalletCharges where id = @SqlId", con);

            cmd.CommandType = CommandType.Text;

            cmd.Parameters.AddWithValue("@SqlId", SqlId);

            cmd.Parameters.AddWithValue("@PalletCharges", PalletCharges);

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

        static void UpdateFreightCharges(string FreightCharges, string SqlId)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("Update KofaxHeader set Freight = @FreightCharges where id = @SqlId", con);

            cmd.CommandType = CommandType.Text;

            cmd.Parameters.AddWithValue("@SqlId", SqlId);

            cmd.Parameters.AddWithValue("@FreightCharges", FreightCharges);

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

        static void UpdateGstHst(string GstHst, string SqlId)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("Update KofaxHeader set gstHst = @GstHst where id = @SqlId", con);

            cmd.CommandType = CommandType.Text;

            cmd.Parameters.AddWithValue("@SqlId", SqlId);

            cmd.Parameters.AddWithValue("@GstHst", GstHst);

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

        static void UpdateInvoiceTotal(string InvoiceTotal,  string SqlId)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("Update kofaxHeader set kofus\_amt = @InvoiceTotal where id = @SqlId", con);

            cmd.CommandType = CommandType.Text;

            cmd.Parameters.AddWithValue("@SqlId", SqlId);

            cmd.Parameters.AddWithValue("@InvoiceTotal", InvoiceTotal);

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

        static void UpdateQst(string Qst, string QstCodeing, string SqlId)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("spUpdateTaxes", con);

            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@Mode", 1);

            cmd.Parameters.AddWithValue("@SqlId", SqlId);

            cmd.Parameters.AddWithValue("@QstAmount", Qst);

            cmd.Parameters.AddWithValue("@QstCoding", QstCodeing);

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

        #endregion

        #region RefNumber

        static void UpdateRefNumber(string RefNumber, string SqlId)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("update kofaxHeader set RefNumber = @RefNumber where id = @SqlId", con);

            cmd.CommandType = CommandType.Text;

            cmd.Parameters.AddWithValue("@SqlId", SqlId);

            cmd.Parameters.AddWithValue("@RefNumber", RefNumber);

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

        #endregion

        static void UpdateInvoiceNumver(string InvoiceNumber, string SqlId)

        {

            SqlConnection con = new SqlConnection("Data Source=vansql07; Initial Catalog=KofaxData; User ID=magappuser;Password=Ivvav!kMes@ch1e");

            SqlCommand cmd = new SqlCommand("Update KofaxData set kofus\_inv = @InvoiceNumber wehere id = @SqlId", con);

            cmd.CommandType = CommandType.StoredProcedure;

            InvoiceNumber = InvoiceNumber.TrimStart('0');

            cmd.Parameters.AddWithValue("@SqlId", SqlId);

            cmd.Parameters.AddWithValue("@HstCoding", InvoiceNumber);

            try

            {

                con.Open();

                Int32 rowsAffected = cmd.ExecuteNonQuery();

            }

            catch (Exception ex)

            {

            }

            finally

            {

                con.Close();

            }

        }

    }

}