```
public static bool IsValid(string EmailAddress)
 {
     string strPattern = @"+
  "^([\w-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([\w-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";

     bool bFlag = false;
     System.Text.RegularExpressions.Regex regex = new System.Text.RegularExpressions.Regex(strPattern);
     System.Text.RegularExpressions.Match match = regex.Match(EmailAddress);
     if (match.Success) bFlag = true;

     return bFlag;
 }
 static void TestSeparatingValidTextData()
 {
     try
     {
         System.IO.StreamReader objReadFile = new System.IO.StreamReader(@"+
                                "C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\Bellevue\"+
                                "Bellevue_EmailsToValidate\Bellevue_EmailToValidateData.csv");
         System.IO.StreamWriter objWriteValidFile = new System.IO.StreamWriter(@"+
                                "C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\Bellevue\"+
                                "Bellevue_ValidEmail\Bellevue_ValidEmailData.csv", false);
         System.IO.StreamWriter objWriteInValidFile = new System.IO.StreamWriter(@"+
                                "C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\Bellevue\"+
                                "Bellevue_InValidEmail\Bellevue_InValidEmailData.csv", false);
         string strLineData = "";
         var firstLine = objReadFile.ReadLine(); // pop off the header line.
         objWriteValidFile.WriteLine(firstLine);
         objWriteInValidFile.WriteLine(firstLine);
         while ((strLineData = objReadFile.ReadLine()) != null)
         {
             //Separate the string of the comma delimited string targeting the third column [0,1,2].
             var parts = strLineData.Split(','); // assuming that ',' is your delimiter character
             if (parts.Length >= 2 && IsValid(parts[2]))
             {
                 objWriteValidFile.WriteLine(strLineData);
             }
```

```
            else
            {
                objWriteInValidFile.WriteLine(strLineData);
            }
        }
        objReadFile.Close();
        objWriteValidFile.Close();
        objWriteInValidFile.Close();
    }
    catch (System.Exception objException)
    {
        System.Console.WriteLine(objException.ToString());
        throw objException; //Must add this for the Main method to catch this exception.
    }
}
/// <summary>
/// This method is called when this script task executes in the control flow.
/// Before returning from this method, set the value of Dts.TaskResult to indicate success or failure.
/// To open Help, press F1.
/// </summary>
public void Main()
{
    try
    {
        TestSeparatingValidTextData();
        Dts.TaskResult = (int)ScriptResults.Success;
    }
    catch (Exception objException)
    {
        string strMessage = @"On Error, please check the file is in C:\_BISolutions\"+
                             "Module08 The Final\Final\ClinicDailyData\Bellevue\ Folder";
        Dts.Events.FireError(0, strMessage, objException.ToString(), string.Empty, 0);
        Dts.TaskResult = (int)ScriptResults.Failure;
    }
}
```

```
#region ScriptResults declaration
/// <summary>
/// This enum provides a convenient shorthand within the scope of this class for setting the
/// result of the script.
///
/// This code was generated automatically.
/// </summary>
enum ScriptResults
{
    Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
    Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
};
#endregion
```