Automating Flat File SQL Server Imports With PowerShell

By: [Tim Smith](https://www.mssqltips.com/sqlserverauthor/92/tim-smith/)

FROM: <https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/>

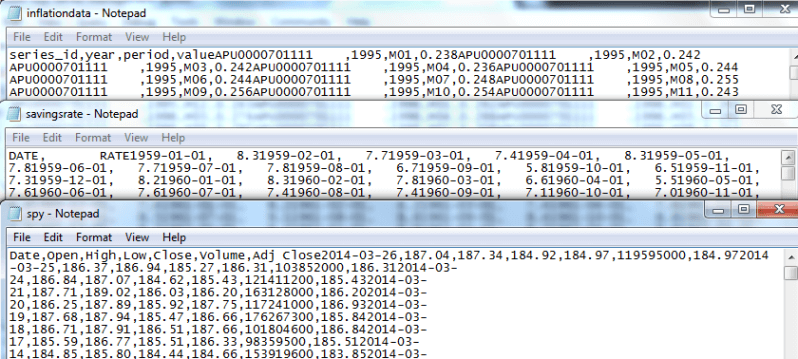
Problem

We're looking to minimize ETL costs by importing as much data as possible using a automated system. A data provider sends us flat file data and we extract, analyze and report on these data. Due to our limited budget and time, how can we automate this?

Solution

[Click here for video version!](https://www.mssqltips.com/sqlservertip/3263/sql-server-video-automating-flat-file-imports-using-powershell/)

Before we begin, if your data provider possesses an API, it may be simpler to grab live data. For the sake of this example, we'll assume that the data provider sends only flat file data and that these files are named in an appropriate manner for their data (meaning, if it's stock data, the data file is "stock.txt"). If this is not the case, you will need to spend time re-naming the files you receive. I've provided three different files - savings.txt, inflationdata.csv and spy.csv as examples for this tip, but will refer to savings.txt for the written example (screenshots of the results are seen for all three).



Before importing flat file data into a table, a table needs to exist for these data. Because we are focusing on extraction, we will import these data into a staging table, accepting everything in the file. We can later filter the results. We will make some assumptions about these files and if any of these assumptions are incorrect, the below code must be changed to reflect it:

The flat files columns are delimited by a character (in this example, a comma).

The column names are listed in the first line of the file.

New line characters demarcate new rows.

The name of the file will be the name of the staging table.

These are important to note because for every exception to the rule, an exception must be noted in code. In a world of competitive data providers, most small startup shops have a large supply of who they want to work with if they don't like a specific company's data. Remember, you're paying for it, so make sure that it's what you want.

PowerShell Code to Automatically Import Data

PowerShell will automatically create our staging table using the above assumptions by reading from the file we want. We must tell PowerShell the name of the file and where the file is located for it to do this. For now, we will code this directly and later turn it into a function:

$location = "C:\files\"

$file = "savingsrate"

$extension = ".txt"

$full = $location + $file + $extension

Write-Host $full

In this step, we've stored the file location in the variable $location, the file name in the variable $file, and the extension of the file in the variable $extension. We then created a $full variable and created the full file path by concatenating the other variables and wrote out the result. This confirms that we're pointing to the correct file. Next, we want to get the first line of the file because this first line, as shown in our above assumptions, will determine the columns in the table.

$location = "C:\files\"

$file = "savingsrate"

$extension = ".txt"

$full = $location + $file + $extension

$all = Get-Content $full

$columns = $all[0]

$columns

Note, that if you're importing a large file, I would suggest the below syntax for columns - and this syntax will work with smaller files as well. It selects the first line from the file without loading the entire file's content to the $columns variable.

$full = $location + $file + $extension

$columns = Get-Content $full | Select -First 1

$columns = $columns.Replace(" ","")

$columns = $columns.Replace(",","] VARCHAR(100), [")

$table = "CREATE TABLE " + $file + "([" + $columns + "] VARCHAR(100))"

In this step, we add two new variables, $all which grabs all the data in the file, and $columns, which grabs only the first line of the file, which will become our columns. We'll notice that $columns returns "DATE, RATE" and we want to create a table with these two columns. Because this will be a staging table, we will use VARCHAR(100) to store the values (but, depending on their size, these can be automated to larger values) and make the name of the table to be the file name. This is shown in this next step with comments:

## Note that we want the comma to be replaced with the syntax for T-SQL column creation

## First, this will remove any unnecessary spaces so that our columns look like ColumnOne,ColumnTwo

$columns = $columns.Replace(" ","")

## Next, note that brackets are used in case of a keyword

$columns = $columns.Replace(",","] VARCHAR(100), [")

## This variable grabs CREATE TABLE, concatenates it with the file name, then concatenates it with the

## columns and ends it with the appropriate syntax to give the last column a VARCHAR of 100. Note that

## if we have larger data sets, we will need to expand it to larger values such as VARCHAR(250)

## or VARCHAR(500).

$table = "CREATE TABLE " + $file + "([" + $columns + "] VARCHAR(100))"

## Let's see our result:

Write-Host $table

We place column names in this syntax between brackets because the column names may be keywords (in this case, DATE is a keyword) and the brackets prevent our code from breaking. When we look at the results, we see that we have the appropriate syntax for creating a table in T-SQL. PowerShell reads the first line of the file, concatenates the necessary pieces and produces a table. Now we need PowerShell to actually create the table. For this we will need a connection to SQL Server and a SQL Server command; first, let's create both of those.

## $connection creates our Connection to SQL Server

$connection = New-Object System.Data.SqlClient.SqlConnection

## $buildTable creates our Sql Server command we'll later execute

$buildTable = New-Object System.Data.SqlClient.SqlCommand

## This establishes our connection string

$connection.ConnectionString = "Data Source=OURSERVER\OURINSTANCE;Database=StagingDatabase;integrated security=true"

## This is command we'll execute when we connect to SQL Server, which as we know creates a table.

$buildTable.CommandText = $table

## This ties the SQL Server connection to the SQL Server command

$buildTable.Connection = $connection

Next, we need to open the connection and execute the command.

$connection.Open()

$buildTable.ExecuteNonQuery()

$connection.Close()

Our staging table is created! We now have a script that can read from the first line of a file and auto-create a table from it. If our delimiters are different, for instance a vertical bar (|) instead of a comma, we can adjust the script to replace "|" with "] VARCHAR(100), [" after removing the empty spaces. Now, let's wrap our code in a function so that we can re-use it:

Function CreateStagingTable($location, $file, $extension, $server, $database)

{

$full = $location + $file + $extension

$all = Get-Content $full

$columns = $all[0]

$columns = $columns.Replace(" ","")

$columns = $columns.Replace(",","] VARCHAR(100), [")

$table = "CREATE TABLE " + $file + "([" + $columns + "] VARCHAR(100))"

$connection = New-Object System.Data.SqlClient.SqlConnection

$buildTable = New-Object System.Data.SqlClient.SqlCommand

$connection.ConnectionString = "Data Source=" + $server + ";Database=" + $database + ";integrated security=true"

$buildTable.CommandText = $table

$buildTable.Connection = $connection

$connection.Open()

$buildTable.ExecuteNonQuery()

$connection.Close()

}

CreateStagingTable -location "C:\files\" -file "savingsrate" -extension ".txt" -server "OURSERVER\OURINSTANCE" -database "StagingDatabase"

Finally, we want to INSERT the data from the file to the staging table. At this point, there are numerous ways to do this. My preferred method is BULK INSERT because I've found it to be one of the most reliable tools when compared with SSIS, BCP, etc. However, feel free to use the tool that works for you. Below, I rename the function, add another SQL Command which passes two parameters into a stored procedure, shown after the PowerShell function:

Function AutoImportCommaFlatFiles($location, $file, $extension, $server, $database)

{

$full = $location + $file + $extension

$all = Get-Content $full

$columns = $all[0]

$columns = $columns.Replace(" ","")

$columns = $columns.Replace(",","] VARCHAR(100), [")

$table = "CREATE TABLE " + $file + "([" + $columns + "] VARCHAR(100))"

$connection = New-Object System.Data.SqlClient.SqlConnection

$buildTable = New-Object System.Data.SqlClient.SqlCommand

$insertData = New-Object System.Data.SqlClient.SqlCommand

$connection.ConnectionString = "Data Source=" + $server + ";Database=" + $database + ";integrated security=true"

$buildTable.CommandText = $table

$buildTable.Connection = $connection

## Added to function

$x = 0

$insertData.CommandText = "EXECUTE stp\_CommaBulkInsert @1,@2"

$insertData.Parameters.Add("@1", $full)

$insertData.Parameters.Add("@2", $file)

$insertData.Connection = $connection

$connection.Open()

$buildTable.ExecuteNonQuery()

$connection.Close()

## Added to function

$x = 1

if ($x = 1)

{

$connection.Open()

$insertData.ExecuteNonQuery()

$connection.Close()

}

}

AutoImportCommaFlatFiles -location "C:\files\" -file "savingsrate" -extension ".txt" -server "OURSERVER\OURINSTANCE" -database "StagingDatabase"

CREATE PROCEDURE stp\_CommaBulkInsert

@file NVARCHAR(250), @table NVARCHAR(250)

AS

BEGIN

DECLARE @f NVARCHAR(250), @t NVARCHAR(250), @s NVARCHAR(MAX)

SET @f = @file

SET @t = @table

SET @s = N'BULK INSERT ' + @t + '

FROM ''' + @f + '''

WITH (

FIELDTERMINATOR = '',''

,ROWTERMINATOR = ''0x0a''

,FIRSTROW=2

)'

EXEC sp\_executesql @s

END

Check the Imported Data

When we execute the below, we see all the data:

SELECT \*

FROM StagingDatabase.dbo.savingsrate

Importing Other Files Using PowerShell Function

Now, let's call the same function for our inflationdata.csv and spy.csv files:

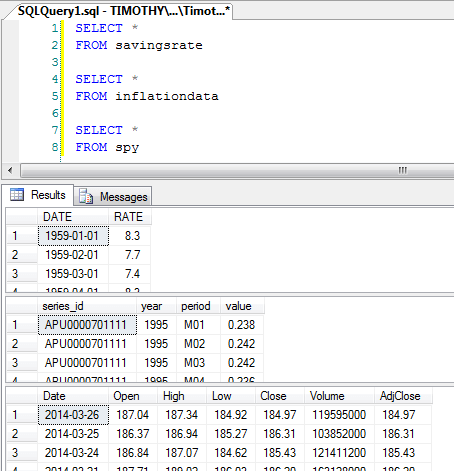
AutoImportCommaFlatFiles -location "C:\files\" -file "inflationdata" -extension ".csv" -server "OURSERVER\OURINSTANCE" -database "StagingDatabase"   
  
AutoImportCommaFlatFiles -location "C:\files\" -file "spy" -extension ".csv" -server "OURSERVER\OURINSTANCE" -database "StagingDatabase"

SELECT \*

FROM StagingDatabase.dbo.inflationdata

SELECT \*

FROM StagingDatabase.dbo.spy



Summary

Finally, we only need to make a few changes if our delimiters are different. You'll notice that my final naming conventions specify what type of flat file data it deals with - stp\_CommaBulkInsert and AutoImportCommaFlatFiles. I can re-name and add to my functions and procedures as needed, and ideally, the only thing that should ever change when calling the function is the name. Depending on your company, you may only deal with a few file formats such as commas and vertical bars and thus will only need a few scripts. From there, we can just change the file name, and everything will be done for us.

While this tip is initially intended for small or startup shops, I've implemented this same code at large shops too, and they've seen the benefit of automatic imports without the use of tools that require hours to debug, or need to be rebuilt for a new set of data. Provided that our flat files are similar in column and row format delimiters, we can re-use this for more than one or two files.

Watch Video Version of Tip

[Click Here](https://www.mssqltips.com/sqlservertip/3263/sql-server-video-automating-flat-file-imports-using-powershell/)

Next Steps

Test comma delimited files and other formats, noting that you may want to expand the VARCHAR length above 100 if the data are large.

Use the [Rename and Move file script](https://www.mssqltips.com/sqlservertip/3207/rename-and-move-a-file-with-powershell-in-a-sql-server-agent-job-/) from the previous tip I wrote to archive the imported files.

Last Update: 2014-04-30

[](https://www.mssqltips.com/get-free-sql-server-tips/?ref=yellow)

[](https://www.mssqltips.com/sql-server-tip-category/115/import-and-export/)

About the author

[](https://www.mssqltips.com/sqlserverauthor/92/tim-smith/)Tim Smith works as a DBA and developer and also teaches Automating ETL on Udemy.   
  
[View all my tips](https://www.mssqltips.com/sqlserverauthor/92/tim-smith/)

Related Resources

[More Database Developer Tips...](https://www.mssqltips.com/sql-server-developer-resources/)

[inShare](javascript:void(0);)14

[Become a paid author](https://www.mssqltips.com/contribute/)

Top of Form



Post a comment or let the author know this tip helped.

All comments are reviewed, so stay on subject or we may delete your comment. Note: your email address is not published. Required fields are marked with an asterisk (\*).

\*Name    \*Email    Notify for updates 

p

  
SQL tips: 

|  |  |
| --- | --- |
| \*Enter Code https://www.mssqltips.com/captcha.asp[refresh code](javascript:void(0)) |  |

Bottom of Form

|  |  |
| --- | --- |
| Saturday, May 27, 2017 - 9:44:32 AM - venkat | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Hello,  Excellent Article..  I am using BULK INSERT for the data import as I believe this is easiest and efficient of all the other options and also I need not request for any additional access to the DBA team. Now I came across a csv file with 4M rows and there are commas as part of the column value enclosed in double quotes. When the SQL BULK INSERT finds a comma as part of csv it assumes a new column value is starting from there on and inputs the data next column. Here goes the problem. I am now trying to use the powershell script to remove those additional commas as part of the enclosed strings and then load the data. Can we do the same with this script? I have around on google for last 3 days but unforunately didn't find the solution. I even tried to do it using Macros but I believe powershell can automated, not sure of macros.  One example for this    I have csv file that contain data like below  1, "abc,xyz", gfgf, "def,ghi"  21, "abc,xyz", gfgf, "def,ghi"    As you can see there additional commas as part of the enclosed double quotes.. I want to replace them with either spaces or any other character.    desired output  1, "abc xyz", gfgf, "def ghi"  21, "abc xyz", gfgf, "def ghi"    Thanks  Venkat | |

|  |  |
| --- | --- |
| Friday, May 05, 2017 - 5:50:36 PM - Anne | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Same question:    Why not use a SSIS package? What is the Powershell advantage? | |

|  |  |
| --- | --- |
| Friday, March 03, 2017 - 12:28:39 PM - Maria Hernandez | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Hi Tim,  Worked like a charm!  Thanks so much!  Hope you are doing well. | |

|  |  |
| --- | --- |
| Wednesday, February 01, 2017 - 6:43:04 PM - Ryan A. | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| The following lines of code from the AutoImportCommaFlatFiles Function should be removed from your example. Nothing productive is done with the variable $X.    ## Added to function  $x = 0  .  ## Added to function  $x = 1  .  if ($x = 1)  {  .  }    The IF statement is ALWAYS TRUE as the assignment of 1 to variable $X always succeeds ($X = 1). To evaluate if $X is \*equal to\* 1 it should be written as ($X -eq 1) in PowerShell. | |

|  |  |
| --- | --- |
| Thursday, January 22, 2015 - 11:51:30 AM - Kevin Smith | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Very nice article. Thank you. I have had several issues where formats within the file are not advertised. (multiple record formats within a file (some identified by prefix, others by number of delimiters, and others with just bad data/delimiters). The bigges issue is getting a delimited file and not knowing the possible data type and max possible number of characters within a column. I'm going to check your validation article Pre-Validating Data with PowerShell Before Importing to SQL Server, but updating this code to include setting the column width would be very cool. | |

|  |  |
| --- | --- |
| Wednesday, December 17, 2014 - 1:32:20 PM - pam | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Probably this is a dummy question.... Where Does the function need to be saved? in SQL server? | |

|  |  |
| --- | --- |
| Thursday, November 27, 2014 - 12:08:48 PM - Rollin Shultz | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Thanks for the excellent article. I find this a valuable method to get unkown data from a file. I have added it to my toolbox and I plan to use it whenever I need it. I amalso using it to help me gain powershell scripting knowledge. | |

|  |  |
| --- | --- |
| Tuesday, July 01, 2014 - 1:55:34 PM - Tim | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| You use the existing command and you can manually name it by removing the " + $file + " and replacing it with a table name that's manual.  Make sure to drop the table after you import to your MainTable. | |

|  |  |
| --- | --- |
| Monday, June 30, 2014 - 2:24:12 PM - Khanh Le | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Hi Tim,  Thank you for your reply. When I create a new table on the fly, is that I use # for temporary? Where should I put in the line $table = "CREATE TABLE " + $file + " ([" + $columns + "] VARCHAR(100))"? or I just do that line the same and in the $buildTable.CommandText = $table, I would try to put some key word here?  Thank you. | |

|  |  |
| --- | --- |
| Sunday, June 29, 2014 - 7:29:53 PM - Tim | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Khanh - Yes, you don't have to go through the process of creating the table, or you can have the script auto create the table and do a INSERT INTO ... SELECT \* FROM ...  In practice how that looks is that we have a FinalTable (formatted and everything) and in our automate import script, we import the data by creating a new table on the fly.  From there, we automatically have a stored procedure which moves the data from the newly created table to the FinalTable and drops the newly created table.  This helps because you never know if some of the data is bad, so you can do a data validation step with the data before inserting it into the FinalTable. | |

|  |  |
| --- | --- |
| Sunday, June 29, 2014 - 1:42:04 PM - Khanh Le | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Hi Tim,  I have the database that already had the table. I have to transfer the fie to the table in that database. I don't think we need the step in creating the table, cause the table is there. But the bulkinsert can be used again for inserting data. Am I right that I just need to delete the step of creating table, so I will not have $buildTable = New-Object System.Data.SqlClient.SqlCommand? But then how do I connect the table in database and the one in the file I want to get the content? Please give me some hints if it's possible.  Thank you very much for all the help. | |

|  |  |
| --- | --- |
| Friday, June 27, 2014 - 12:45:51 PM - Khanh Le | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Hi Tim,  Thank you very much for your quick reply. I am able to fix my problems now. Thanks a lot. I've been reading a lot of examples and watching a lot of video clip in youtube, but I found that your instruction is the most helpful. You're a great teacher!!!  Best,  Khanh Le. | |

|  |  |
| --- | --- |
| Thursday, June 26, 2014 - 5:06:03 PM - Tim | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| The first error indicates that you didn't pass any columns into the table.  From looking at your information, I don't see your delimiter and in the example, I use a comma delimiter which will demarcate the columns - $columns = $columns.Replace(",","] VARCHAR(100), [").  The second error indicates that you don't have stp\_BulkInsert in your database, which you'll need to build and make sure that it also matches your delimiter.  If your file isn't comma delimited, then you'll need to change the script to specify your delimiter.  For further information, [see this video](https://www.mssqltips.com/sqlservertip/3263/sql-server-video-automating-flat-file-imports-using-powershell/). | |

|  |  |
| --- | --- |
| Thursday, June 26, 2014 - 3:09:36 PM - Khanh Le | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Hi Tim,  Thank you very much for the post. But when I copy the code, AutoImportCommaFlatFiles to PS, and it shows errors. Everything I did the same, except the last line I changed a little bit:  AutoImportCommaFlatFiles -location "C:\PS\" -file "processes" -extension ".txt" -server "KHANHLEE\SQL2012XP" -database "Sensor"  My processes.txt looks like this:    Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName  -------  ------    -----      ----- -----   ------     -- -----------       34       4      484       2348    12     0.00   1508 AERTSr64      167      12    10432      13608    80   587.77   1656 audiodg     1090      64    59684      96804   341   351.72    512 chrome      199      53   103936     101920   290    33.83   3472 chrome      191      27    46312      44388   216     3.59   4564 chrome      326      29    77680     104028   340   256.47   5104 chrome      227      57   200208     198416   406   293.16   5332 chrome      194      27    21372      22608   236    37.38   6200 chrome      190      26    39436      35976   204     2.64   6964 chrome      248      45   176560     176084   355   129.53   7884 chrome       42       4      552       2484    23     0.02   1260 conhost      527      15     1944       4480    48     2.06    556 csrss      479      27     2384      36872   174     7.45   6820 csrss      195      13     3412      10164    44     1.30   1640 dasHost      177      14     3016       7120    64     0.36   3660 devmonsrv      117       7     1512       6604    54     0.05   2128 dllhost      243      28    21264      31216   187    91.91   6072 dwm      286      17     4020      10872    93     0.28   1560 EvtEng     2414     149    85812     148032  1149    36.31   2456 explorer      164      13     2956       8408   116     0.16   7404 FlashUtil\_ActiveX      118       8     1260       5044    54     0.05   1684 HeciServer      112       9     1496       5844    87     0.02   5128 hkcmd      414      47    23988      51148   171     5.45   3168 IAStorDataMgrSvc      259      22    21952      26848   268     0.53   4212 IAStorIcon      112       8     1408       4664    56     0.06   1756 ibtrksrv        0       0        0          4     0               0 Idle     1133     123   163764     193860   558   193.48    180 iexplore      816      80    58964      84832   369    37.13    704 iexplore      707      49    16000      41388   247    13.17   4276 iexplore      687      46    34944      60468   297     4.77   7140 iexplore      123      10     1588       6080    89     0.03   6368 igfxpers      113      11     2260       6432    80     0.14   3544 igfxsrvc      112       9     1520       5872    92     0.05   5348 igfxtray       64       7      784       3532    26     0.00   1440 IntelMeFWService      333      21     3884      11696   103     0.13   1792 iSCTAgent       91       9     1148       4652    84     0.02   6704 iSCTsysTray8      612      57    16196       3260   222     1.89   7296 ismagent       73       8      940       4224    29     0.02   2480 jhi\_service      637      30    14800      19440   853     0.30   6104 livecomm      304      14     3488       9604    62     0.34   1268 LMS     1417      21     7232      15132    51    12.86    764 lsass       85       7     4624       6524    57     0.09   2576 McAPExe      768      39   269432     177788   461   415.06   2612 mcshield     1354     130    48228      31400   337    87.14   2920 McSvHost      875      78    47784      47492   368    11.16   6760 McUICnt      130      14     3008       6520    42    24.27   2868 mfefire      190       7     2676       5444    25     1.38   1888 mfevtps      183     167     2896       7032    63     0.34   3932 obexsrv      448      56    29660      51864   602    10.52   3256 PocketCloudService      613      60   152396     173968   922    48.83   6300 powershell\_ise      213      14     2568       9064   107     0.16   7724 quickset      205      13     4940       9452   110     0.09   3460 RAVBg64      205      13     5272       9908   111     0.05   4716 RAVBg64      228      14     5172      10168   113     0.08   6028 RAVBg64      229      15     6828      11064   120     0.08   6032 RAVBg64      127       9     1576       6328    65     0.05   2024 RegSrvc      111       8     1320       4900    50     0.03    384 RtkAudioService64      308      17     4612      10084   126     0.09   7400 RtkNGUI64      204      16     2752      11136   126     0.14   5148 rundll32      237      16     3800      13576   112     1.20   5896 RuntimeBroker      676      69    82836      88348   393    37.77   4044 SearchIndexer      308      11     4080       7120    25    25.84    756 services      475      23     6396       4440   141     0.86   3340 SettingSyncHost      172      14     2460       8368    80     8.59   1080 SftService      583      29     8040      12740   157     0.86   5544 SkyDrive       44       2      276       1032     4     0.23    340 smss      398      22     5196      11940    75     6.28   1364 spoolsv      101       9     1480       5552    40     0.03   1600 sqlwriter      888      48    15532      24444   127    13.70    380 svchost      886      38    80472      94600   171   253.30    468 svchost      503      24     7956      14148    63    58.81    828 svchost      692      16     6528       9600    33    12.25    876 svchost      983      33    23452      31500   134    17.55    952 svchost     2092      66    33664      49432   259   130.70   1008 svchost      934     143    14120      23116  1248    20.94   1068 svchost      633      40    23160      27904   107    38.98   1404 svchost      119      10     3504       7656    38     0.58   1532 svchost      161      14     4644       8816    46     1.41   1588 svchost      106       8     2352       5244    22     5.16   2760 svchost      483      29     6540      14276    90     5.75   2932 svchost      340      14     4732      11220    45     9.66   3672 svchost      443      14     4792       5108   105   546.05   5380 SynTPEnh       42       6      764        212    53     0.03   8040 SynTPHelper     1616       0    35148       8072    45   275.19      4 System      416      25     7704      19564   204     5.92   4164 TabTip       43       6      680       2852    40     0.02   4560 TabTip32      115       9     1344       4916    85     0.03   7916 taskhost      311      26     5872      12892   298     2.72    176 taskhostex      286      27    22540      27188   276     0.78   1932 ThinkGear Connector       94       6     1308       4556    27     0.06   2252 unsecapp      281      24     9952       4400   130     0.63   6972 updateui       81       8      924       3988    40     0.72    660 wininit      156       7     1208       5256    49     0.17   4020 winlogon      444      19     4152      13004    87     4.81   1236 wlanext      188      15     5684      12060    74     1.44   2340 WmiPrvSE      415      25     6188       4172    95     0.89   3040 wmpnetwk      292      14     3048       8632    67     0.55   3084 WUDFHost      368      29    10624      22312   185     0.41   4656 WWAHost      707      72    63300      91956   355     4.36   5200 WWAHost      170      14     1908       5248    60     0.05   2460 WyseRemoteAccess      336      20     4980      15112   104     2.80   2524 ZeroConfigService  Here is the errors:  Exception calling "ExecuteNonQuery" with "0" argument(s): "An object or column  name is missing or empty. For SELECT INTO statements, verify each column has a  name. For other statements, look for empty alias names. Aliases defined as ""  or [] are not allowed. Change the alias to a valid name."  At C:\Users\Khanh\Documents\examplePowershell\AutoImportCommaFlatFiles.ps1:21  char:5  +     $buildTable.ExecuteNonQuery()  +     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~      + CategoryInfo          : NotSpecified: (:) [], MethodInvocationException      + FullyQualifiedErrorId : SqlException    Exception calling "ExecuteNonQuery" with "0" argument(s): "Could not find  stored procedure 'stp\_CommaBulkInsert'."  At C:\Users\Khanh\Documents\examplePowershell\AutoImportCommaFlatFiles.ps1:29  char:9  +         $insertData.ExecuteNonQuery()  +         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~      + CategoryInfo          : NotSpecified: (:) [], MethodInvocationException      + FullyQualifiedErrorId : SqlException    I don't know what I did wrong, could you please help me?  Again, thank you so much. | |

|  |  |
| --- | --- |
| Monday, June 23, 2014 - 3:39:24 AM - RS | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Hi Tim : As always, you write for the novice which is really wonderful.  All my excel files have plenty of empty rows at the end of the data rows. How do i get rid of them just before importing via SSIS.  Ifyou can write an article on that for excel, it would be very helpful    Thank you | |

|  |  |
| --- | --- |
| Saturday, June 21, 2014 - 10:22:20 AM - MDC | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Great article!  How do you modify the function to read a text like below?  [Cars]  Toyota = 3  Nissan = 0  [Dealers]  SouthDealer = 4,0 (\*this is a comma)  WestDealer = 3,0    [Cars] and [Dealers] are the table names  Cars table columns below:  ID    carName   required  1      Toyota     3  2      Nissan      0    Dealers table columns below:  ID    dealerName    value  1      SouthDealer   4,0  2      WesDealer     3,0 | |

|  |  |
| --- | --- |
| Thursday, June 12, 2014 - 9:07:45 PM - Tim | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| @Gordon Definitely look at [this article here on MSSQLTips about pre-validating data](https://www.mssqltips.com/sqlservertip/3236/prevalidating-data-with-powershell-before-importing-to-sql-server/) because if you have a line with extra delimiters, meaning you may have more rows than the first line dictates, if enough space is provided, BULK INSERT will add those to the final column, as you can see in the first example.  If your final column is VARCHAR(MAX), then the lines with extra delimiters - often wrapped between quotes like you showed - will cause the row to be moved over based on delimiters and in the final column you will find those extra values in last column.  In other words, the above script with a final column of VARCHAR(MAX) may not break, though you should be aware that some of your rows will have incorrect values and (1) you can move things over using TSQL, or (2) handle those issues before importing, such as writing good data to a valid file, then building a process to handle the exceptions. | |

|  |  |
| --- | --- |
| Wednesday, June 11, 2014 - 9:07:37 AM - Gordon | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| I meant multiple row lengths in terms of the number of delimiters per line (apologies this wasn't clear). Also, I had to handle some text cells/entries that were covered by "" (so that the text could include a comma without that being a delimiter) and then handle entries that included a " within the text!  For example: an original cell in the spreadsheet that was: 4, Belmont Drive (and so in the csv file was "4, Belmont Drive")  or a tougher single cell that was: "The Landings", 4 Belmont Drive (and so in the csv file hopefully was """The Landings"", 4 Belmont Drive")  But the last, at least, might have to wait for another look.  Thanks | |

|  |  |
| --- | --- |
| Wednesday, June 11, 2014 - 8:46:10 AM - Tim | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| @Gordon Thanks for the question; when you wrote multiple row lengths in the file, do you mean the number of delimiters in the CSV changed per line, such as the first line might have 7 commas, the next line would have 4 commas, the next line would have 6 commas, etc?  Or do you mean that the rows had the same number of delimiters, but were just very long?  See below examples:    Multiple row lengths (ie: delimiter amount changes per line)  1,2,3,4,5,6,7,8 1,2,3,4 1,2,3,4,5,6,7  Some rows are longer than others:  RowOneColumnOne,RowTwoColumnTwo RowOneAndVERYVERYVERYVERYVERYVERYVERYLongColumnOne,RowTwoAndVERYVERYVERYVERYVERYVERYVERYLong  Thanks! | |

|  |  |
| --- | --- |
| Tuesday, June 10, 2014 - 11:04:32 AM - Gordon | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Will this work properly if the original spreadsheet has lines of differing lengths?  I had to work recently with some large csv files which had multiple row lengths within each file, making most tools (like batchimport) fail; I landed doing a wizard-based import for each. Thanks. | |

|  |  |
| --- | --- |
| Tuesday, May 06, 2014 - 12:26:48 PM - ken ambrose | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Hi Frank,  Have you ever programmed an SSIS package that can process \_any arbitrary text file structure\_, parse the columns, and load into table ?  I've used SSIS for large and small projects for many years.  SSIS is optomized for performance, and there are trade offs in the optomizations that Microsoft made.  One of the tradeoffs is that SSIS fights you big time if you want to work with data structure that is not "cast in stone" at design time.  It can be done sort of, by bringing in each line from the text file as one record, then using code - I used tsql - to parse out arbitary column structure from the one column "landing" table, based on the delimeter, and then dynamically creating the appropriate DDL to create a target table and "shred" each row into the target table.  However when you end up writing that much code, the benefits of using SSIS are much diminished vs. just doing everything in code.  If you have any ideas about how to use SSIS to load text files with arbitrary column structure into a db table, without using code to do 90% of the work, please post an example, I'd love to learn how...  ken | |

|  |  |
| --- | --- |
| Monday, May 05, 2014 - 6:18:15 PM - FrankQ | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Why not use a SSIS package? What is the Powershell advantage? | |

|  |  |
| --- | --- |
| Monday, May 05, 2014 - 3:31:04 PM - Hank Freeman | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Tim, I use a ton of SQL Server BULK INSERT statements and this article looks to be a really good post. Thanks for that...  Once I get feel for your code I willl respond back with more info... I right now I have several input file issue I have had to program around so this will be helpful in allowing me to think outside my own box..  Hank Freeman  SQL Server DBA / Data Architect - Atlanta, GA. | |

|  |  |
| --- | --- |
| Thursday, May 01, 2014 - 10:20:08 AM - Tim | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| @felix With outside type files, I use format files which I can repeat use for the process.  In the code above, PowerShell would perform a loop (if there were multiple files) and truncate the staging table on each pass.  I would add a parameter for the format file if I had several different "types" of files.  Because format files allow for re-use, there would be no need to re-construct a table.  The procedure for BULK INSERT could accept any format file I passed in from PowerShell. | |

|  |  |
| --- | --- |
| Wednesday, April 30, 2014 - 6:26:12 PM - felix | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Thank you Tim for the article.  How you change your code for the Ragged, fixed column width file? | |

|  |  |
| --- | --- |
| Wednesday, April 30, 2014 - 1:39:27 PM - TimothyAWiseman | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Excellent article.  I have created similarly automated processes that work in pure T-SQL, but this certainly shows the flexibility of powershell. | |

|  |  |
| --- | --- |
| Wednesday, April 30, 2014 - 12:23:09 PM - ken ambrose | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Excellent article that clearly shows how incredibly \_useful\_ powershell is using just a few lines of code!  After 20+ years in this profession working with Microsoft technologies, in my opinion there is no better tool one can use to get infrastructure work done quickly, efficiently, robustly... | |

|  |  |
| --- | --- |
| Wednesday, April 30, 2014 - 12:21:12 PM - Tim | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| @Jon Lellelid Hey thanks for calling attention to that error - it should be "grabs." | |

|  |  |
| --- | --- |
| Wednesday, April 30, 2014 - 11:29:40 AM - Jon Lellelid | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| How timely, I imported a file, just yesterday; this would have made the task so much easier.  In the 4th "code paragraph" you mention the word "graps" in this statement: " ## This variable graps CREATE TABLE," Is this a typo or a command? I did not find it in Wikipedia or in Powershell?  Thanks. | |

|  |  |
| --- | --- |
| Wednesday, April 30, 2014 - 10:09:18 AM - Darek | [Back To Top](https://www.mssqltips.com/sqlservertip/3208/automating-flat-file-sql-server-imports-with-powershell/#top) |
| Pretty nice. With PowerShell one can do everything and anything these days... PowerShell is ENORMOUSLY POWERFUL. Thanks for the article. | |