**How do I load text or csv file data into SQL Server?**

If you need to load data into SQL Server (e.g. from log files, csv files, chat transcripts etc), then chances are, you're going to be making good friends with the BULK INSERT command.   
   
You can specify row and column delimiters, making it easy to import files in comma-separated values (CSV), Tab-separated values (TSV), or any other delimiter (e.g. the pipe character is commonly used, since it rarely exists in valid data). You can also tell it to skip any number of header rows (these are fairly common in many log file formats).   
   
So, let's run through a few examples. We have this table: 

|  |
| --- |
| CREATE TABLE OrdersBulk  (      CustomerID INT,      CustomerName VARCHAR(32),      OrderID INT,      OrderDate SMALLDATETIME  ) |

And let's say we have this CSV file, saced as c:\file1.csv: 

|  |
| --- |
| 1,foo,5,20031101  3,blat,7,20031101  5,foobar,23,20031104 |

The command to bulk insert this data would be: 

|  |
| --- |
| BULK INSERT OrdersBulk      FROM 'c:\file.csv'      WITH      (          FIELDTERMINATOR = ',',          ROWTERMINATOR = '\n'      ) |

Now, if we had tab-separated values, it would look like this: 

|  |
| --- |
| BULK INSERT OrdersBulk      FROM 'c:\file.csv'      WITH      (          FIELDTERMINATOR = '\t',          ROWTERMINATOR = '\n'      ) |

Note that a row delimiter often has to be experimented with, because often log files produce "columns" in the table by putting the value and then the delimiter. So, in many cases, you might actually be looking to separate rows by column delimiter + row delimiter. So, the last column has a dangling column delimiter that isn't really necessary (but we have to deal with it). I currently have processes that look like this, to deal with files that have trailing tab characters after every column: 

|  |
| --- |
| BULK INSERT OrdersBulk      FROM 'c:\file.csv'      WITH      (          FIELDTERMINATOR = '\t',          ROWTERMINATOR = '\t\n'      ) |

And earlier I mentioned the case where you have header rows that are almost never going to match the data types of the target table (and you wouldn't want the headers mixed in with the data, even if all the data types were character). So, let's say the CSV file actually looked like this: 

|  |
| --- |
| CustomerID,CustomerName,OrderID,OrderDate  1,foo,5,20031101  3,blat,7,20031101  5,foobar,23,20031104 |

If you try the BULK INSERT command from above, you will get this error: 

|  |
| --- |
| Server: Msg 4864, Level 16, State 1, Line 1  Bulk insert data conversion error (type mismatch) for row 1, column 1 (CustomerID). |

This is because the first row doesn't contain valid data, and the insert fails. We could tell BULK INSERT to ignore the first row in the data file, by specifying the starting row with the FIRSTROW parameter: 

|  |
| --- |
| BULK INSERT OrdersBulk      FROM 'c:\file.csv'      WITH      (          **FIRSTROW = 2**,          FIELDTERMINATOR = ',',          ROWTERMINATOR = '\n'      ) |

Finally, you can also specify how many errors you want to allow before considering that the BULK INSERT failed. We can never really be confident that a log file from some external source will always have "perfect" data in it. Failures can occur when you have character data that is too big for the column in the table, or if you have a string in an integer column, or if your date is malformed, or several other potential scenarios. So, you probably want some threshold where more than n rows will cause the BULK INSERT to fail. You can specify this limit using the MAXERRORS parameter. (I've already requested for Yukon that they add a MAXERRORPERCENT so that there will be an appropriate failure ratio for both large and small files.) 

|  |
| --- |
| BULK INSERT OrdersBulk      FROM 'c:\file.csv'      WITH      (          FIRSTROW = 2,          **MAXERRORS = 0**,          FIELDTERMINATOR = ',',          ROWTERMINATOR = '\n'      ) |

You can also use bcp to move data from a flat file into an existing table. BULK INSERT and the bcp utility are documented in Books Online.