How to export data as CSV format from SQL Server using sqlcmd?

Asked 15 years, 11 months ago Modified 6 months ago Viewed 530k times

174

I can quite easily dump data into a text file such as:

```
sqlcmd -S myServer -d myDB -E -Q "select col1, col2, col3 from SomeTable"
     -o "MyData.txt"
```

However, I have looked at the help files for `SQLCMD` but have not seen an option specifically for CSV.

Is there a way to dump data from a table into a CSV text file using `SQLCMD`?

sql-server file csv sqlcmd

Share
Improve this question
Follow

edited May 1, 2012 at 14:41

SteveC
16.7k ● 25 ● 112 ● 179

asked Jan 8, 2009 at 18:49

Ray
192k ● 99 ● 227 ● 206

Must this be via sqlcmd, or could you use another program such as the following: codeproject.com/KB/aspnet/ImportExportCSV.aspx – Bernhard Hofmann Jan 8, 2009 at 19:06

1 It doesn't have to be but I wanted to know for certain whether or not sqlcmd could actually do this before diving into some other export utility. One thing to mention is that it does need to be scriptable. – Ray Jan 8, 2009 at 20:05

There is a SSMS 2008 addin tool that does CSV output from your tables that can be customized by where and order by clauses. store.nmally.com/software/sql-server-management-studio-addons/… – user975249 Jul 14, 2013 at 6:35

12 Answers

Sorted by: Highest score (default)

168

You can run something like this:

```
sqlcmd -S MyServer -d myDB -E -Q "select col1, col2, col3 from SomeTable"
       -o "MyData.csv" -h-1 -s"," -w 700
```

- `-h-1` removes column name headers from the result

- `-s","` sets the column separator to `,`
- `-w 700` sets the row width to 700 chars (this will need to be as wide as the longest row or it will wrap to the next line)

Share

Improve this answer

Follow

edited Feb 13 at 17:05

mklement0

434k ● 68 ● 698 ● 903

answered Jan 8, 2009 at 19:08

scottm

28.7k ● 22 ● 108 ● 161

37 The caveat with doing it this way is that your data may not contain any commas. – Sarel Botha Oct 30, 2012 at 16:57

3 @SarelBotha, you can get around that problem with `'""' + col1 + '""' AS col1`, wrapped in (doubled) double quotes or just call a stored procedure. – MisterIsaak Dec 11, 2013 at 17:17

3 @JIsaak Then make sure your data does not have any double quotes or make sure to replace your double quotes with two double quotes. – Sarel Botha Dec 12, 2013 at 19:23

7 Downvoted because this does not produce valid csv formatted files. See the answer which uses `Export-csv`. – Jthorpe Aug 8, 2017 at 17:12

4 This is not outdated (there's no PowerShell on Linux!) but it doesn't work correctly for several reasons including that without NOCOUNT you'll have the rows affected in your data. – Hack-R Mar 29, 2018 at 22:28

119

With PowerShell you can solve the problem neatly by piping `Invoke-Sqlcmd` into `Export-Csv`.

```
#Requires -Module SqlServer
Invoke-Sqlcmd -Query "SELECT * FROM DimDate;" `
              -Database AdventureWorksDW2012 `
              -Server localhost |
Export-Csv -NoTypeInformation `
           -Path "DimDate.csv" `
           -Encoding UTF8
```

`Invoke-Sqlcmd` is the PowerShell equivalent of sqlcmd.exe. Instead of text it outputs `System.Data.DataRow` objects.

The `-Query` parameter works like the `-Q` parameter of sqlcmd.exe. Pass it a SQL query that describes the data you want to export.

The `-Database` parameter works like the `-d` parameter of sqlcmd.exe. Pass it the name of the database that contains the data to be exported.

The `-Server` parameter works like the `-S` parameter of sqlcmd.exe. Pass it the name of the server that contains the data to be exported.

`Export-CSV` is a PowerShell cmdlet that serializes generic objects to CSV. It ships with PowerShell.

The `-NoTypeInformation` parameter suppresses extra output that is not part of the CSV format. By default the cmdlet writes a header with type information. It lets you know the type of the object when you deserialize it later with `Import-Csv`, but it confuses tools that expect standard CSV.

The `-Path` parameter works like the `-o` parameter of sqlcmd.exe.

The `-Encoding` parameter works like the `-f` or `-u` parameters of sqlcmd.exe. By default Export-Csv outputs only ASCII characters and replaces all others with question marks. Use UTF8 instead to preserve all characters and stay compatible with most other tools.

The main advantage of this solution over sqlcmd.exe or bcp.exe is that you don't have to hack the command to output valid CSV. The Export-Csv cmdlet handles it all for you.

The main disadvantage is that `Invoke-Sqlcmd` reads the whole result set before passing it along the pipeline. Make sure you have enough memory for the whole result set you want to export.

It may not work smoothly for billions of rows. If that's a problem, you could try the other tools, or roll your own efficient version of `Invoke-Sqlcmd` using System.Data.SqlClient.SqlDataReader class.

Differences between SQL Server versions

As of SQL Server 2016, `Invoke-Sqlcmd` ships as part of the `SqlServer` module.

SQL Server 2012 instead has the old SQLPS module. When the module is imported, it changes the current location to `SQLSERVER:\`. So you'll need to change the `#Requires` line above to:

```
Push-Location $PWD
Import-Module -Name SQLPS
# dummy query to catch initial surprise directory change
Invoke-Sqlcmd -Query "SELECT 1" `
              -Database  AdventureWorksDW2012 `
              -Server localhost |Out-Null
Pop-Location
# actual Invoke-Sqlcmd |Export-Csv pipeline
```

A full path for `Export-Csv`'s `-Path` parameter is safest if you are stuck using the old SQLPS module.

To adapt the example for SQL Server 2008 and 2008 R2, remove the `#Requires` line entirely and use the sqlps.exe utility instead of the standard PowerShell host.

Share
Improve this answer
Follow

edited Mar 24, 2022 at 15:46

community wiki
7 revs, 3 users 81%
Iain Samuel McLean Elder

1 On SQL 2008 R2, I had to run the "sqlps.exe" tool to use the Invoke-Sqlcmd. Apparently I need SQL 2012 to use Import-Module? Anway it works within "sqlps.exe" - see this thread for details. – Mister_Tom Feb 3, 2015 at 20:11

1 @Mister_Tom good point. The SQLPS module was introduced with SQL 2012. The answer now explains how to adapt the example for older editions. – Iain Samuel McLean Elder Feb 3, 2015 at 23:11

1 @JasonMatney PowerShell is the new administrative interface to Windows systems, but a lot of SQL Server advice was published before it became standard. Spread the word! :-) – Iain Samuel McLean Elder Jul 2, 2015 at 19:25

4 This answer provides useful information and a powerful and flexible ALTERNATIVE way to handle the issue, however it does *completely fail to answer* the original question as it was specifically asked. I'm a powershell fan too, but let's not let evangelism turn into hysteria. SQLCMD is not going away any time soon. – barbecue Sep 4, 2016 at 18:34

2 The **SQLPS** module has now been supplanted by the superior **SqlServer** module which installs with SSMS 2016, and fixes several serious bugs (including ambiguous versioning, and changing to the `SQLSERVER:` location at module import). – brianary Mar 28, 2017 at 22:57

84

```
sqlcmd -S myServer -d myDB -E -o "MyData.txt" ^
    -Q "select bar from foo" ^
    -W -w 999 -s","
```

The last line contains CSV-specific options.

- `-W` remove trailing spaces from each individual field
- `-s","` sets the column seperator to the comma (,)
- `-w 999` sets the row width to 999 chars

scottm's answer is very close to what I use, but I find the `-W` to be a really nice addition: I needn't trim whitespace when I consume the CSV elsewhere.

Also see the MSDN sqlcmd reference. It puts the `/?` option's output to shame.

Share
Improve this answer
Follow

edited May 23, 2017 at 12:17

Community Bot
1 • 1

answered Mar 11, 2010 at 16:42

ESV
7,730 • 4 • 40 • 29

22 @sims "set nocount on" in the beginning of the query/inputfile – d-_-b Mar 31, 2010 at 7:54

9 How can I remove underlining on the Headers? – ntombela May 26, 2010 at 9:06

@gugulethun : You can do a union in your query to put the column name on the first line. – Nordes Apr 20, 2011 at 9:16

2 This works like a charm, but if you column contains the seperator i get a corrupted csv file... – Peter Apr 12, 2013 at 9:19

1 Added this comment to the accepted answer as well but... you can get around that problem with `'"' + col1 + '"' AS col1`, wrapped in (doubled) double quotes or just call a stored procedure. – MisterIsaak Dec 11, 2013 at 17:19

67

Is this not `bcp` was meant for?

```
bcp "select col1, col2, col3 from database.schema.SomeTable" queryout
"c:\MyData.txt"  -c -t"," -r"\n" -S ServerName -T
```

Run this from your command line to check the syntax.

```
bcp /?
```

For example:

```
usage: bcp {dbtable | query} {in | out | queryout | format} datafile
  [-m maxerrors]            [-f formatfile]          [-e errfile]
  [-F firstrow]             [-L lastrow]             [-b batchsize]
  [-n native type]          [-c character type]      [-w wide character type]
  [-N keep non-text native] [-V file format version] [-q quoted identifier]
  [-C code page specifier]  [-t field terminator]    [-r row terminator]
  [-i inputfile]            [-o outfile]             [-a packetsize]
  [-S server name]          [-U username]            [-P password]
  [-T trusted connection]   [-v version]             [-R regional enable]
  [-k keep null values]     [-E keep identity values]
  [-h "load hints"]         [-x generate xml format file]
  [-d database name]
```

Please, note that `bcp` cannot output column headers.

See: bcp Utility docs page.

Example from the above page:

> ```
> bcp.exe MyTable out "D:\data.csv" -T -c -C 65001 -t , ...
> ```

Share

Improve this answer

Follow

edited May 29 at 19:08

TylerH

21.2k • 76 • 79 • 110

answered Mar 25, 2011 at 5:19

john.da.costa

4,802 • 4 • 30 • 30

1 ServerName = YourcomputerName\SQLServerName, only then it executes otherwise error – TheTechGuy Oct 18, 2011 at 12:40

2 In case you want to see the full documentation for bcp.exe: technet.microsoft.com/en-us/library/ms162802.aspx – Robert Bernstein Nov 12, 2013 at 14:42

5 What do you do if you need to export the column names as a header as well? Is there a straightforward generic solution using bcp? – Iain Samuel McLean Elder Jun 1, 2014 at 3:19

1 It looks like there is no way to get column names with bcp. And that makes it useless for the case. – Andriy K Mar 2, 2016 at 17:24

3 `bcp` doesn't includer header (column names), but it's ~10X faster than `sqlcmd` (from my experience). For a really big data, you can use `bcp` to get the data, and use `sqlcmd` (select top 0 * from ...) to get the header, and then combine them. – YJZ Jun 4, 2018 at 23:12

14

A note for anyone looking to do this but also have the column headers, this is the solution that I used an a batch file:

```
sqlcmd -S servername -U username -P password -d database -Q "set nocount on;
set ansi_warnings off; sql query here;" -o output.tmp -s "," -W
type output.tmp | findstr /V \-\,\- > output.csv
del output.tmp
```

This outputs the initial results (including the ----,---- separators between the headers and data) into a temp file, then removes that line by filtering it out through findstr. Note that it's not perfect since it's filtering out `-,-`—it won't work if there's only one column in the output, and it will also filter out legitimate lines that contain that string.

Share Improve this answer Follow

answered Aug 31, 2011 at 16:04

Rudism

1,655 • 11 • 13

1 Use following filter instead: findstr /r /v ^\-[,\-]*$ > output.csv For some reason simle ^[,\-]*$ matches all lines. – Volodymyr Korolyov Sep 8, 2011 at 15:27

5 Always put a regex with ^ inside double quotes, or you get weird results, because ^ is an escape character for cmd.exe. Both regexes above don't work properly, as far as I can tell, but this does: findstr /r /v "^-[-,]*-.$" (the . before the $ seems to be needed when testing with echo, but might not for sqlcmd output) – JimG Mar 9, 2012 at 7:01

There's also a problem with dates having 2 spaces between date and time instead of one. When you try to open the CSV in Excel, it show as 00:00.0. An easy way to resolve this would be to search and replace all " " with " " in-place using SED. The command to add to your

script would be: SED -i "s/ / /g" output.csv. More about SED gnuwin32.sourceforge.net/packages/sed.htm – PollusB Jun 13, 2012 at 16:04

this is the correct answer, works perfectly with @JimG's addition. I used semicolon for separator, and an sql file for input, the file contained the noncount on and ansi_warning off part, and the -W switch for space removal – robotik Nov 17, 2015 at 21:07

2

This answer builds on the solution from @iain-elder, which works well except for the large database case (as pointed out in his solution). The entire table needs to fit in your system's memory, and for me this was not an option. I suspect the best solution would use the System.Data.SqlClient.SqlDataReader and a custom CSV serializer (see here for an example) or another language with an MS SQL driver and CSV serialization. In the spirit of the original question which was probably looking for a no dependency solution, the PowerShell code below worked for me. It is very slow and inefficient especially in instantiating the $data array and calling Export-Csv in append mode for every $chunk_size lines.

```
$chunk_size = 10000
$command = New-Object System.Data.SqlClient.SqlCommand
$command.CommandText = "SELECT * FROM <TABLENAME>"
$command.Connection = $connection
$connection.open()
$reader = $command.ExecuteReader()

$read = $TRUE
while($read){
    $counter=0
    $DataTable = New-Object System.Data.DataTable
    $first=$TRUE;
    try {
        while($read = $reader.Read()){

            $count = $reader.FieldCount
            if ($first){
                for($i=0; $i -lt $count; $i++){
                    $col = New-Object System.Data.DataColumn
$reader.GetName($i)
                    $DataTable.Columns.Add($col)
                }
                $first=$FALSE;
            }

            # Better way to do this?
            $data=@()
            $emptyObj = New-Object System.Object
            for($i=1; $i -le $count; $i++){
                $data +=  $emptyObj
            }

            $reader.GetValues($data) | out-null
            $DataRow = $DataTable.NewRow()
            $DataRow.ItemArray = $data
            $DataTable.Rows.Add($DataRow)
            $counter += 1
```

```
            if ($counter -eq $chunk_size){
                break
            }
        }
        $DataTable | Export-Csv "output.csv" -NoTypeInformation -Append
    }catch{
        $ErrorMessage = $_.Exception.Message
        Write-Output $ErrorMessage
        $read=$FALSE
        $connection.Close()
        exit
    }
}
$connection.close()
```

Share

Improve this answer

Follow

edited Jun 28, 2017 at 15:12

answered Jun 27, 2017 at 21:05

jeffmax

469 ● 4 ● 14

1

Alternate option with BCP:

```
exec master..xp_cmdshell 'BCP "sp_who" QUERYOUT C:\av\sp_who.txt -S MC0XENTC -T
-c '
```

Share

Improve this answer

Follow

edited Oct 30, 2017 at 16:54

kenorb

166k ● 94 ● 706 ● 773

answered Nov 16, 2016 at 5:04

Dinesh vishe

3,598 ● 1 ● 26 ● 25

1

Usually `sqlcmd` comes with `bcp` utility (as part of `mssql-tools`) which exports into CSV by default.

Usage:

```
bcp {dbtable | query} {in | out | queryout | format} datafile
```

For example:

```
bcp.exe MyTable out data.csv
```

To dump all tables into corresponding CSV files, here is the *Bash* script:

```
#!/usr/bin/env bash
# Script to dump all tables from SQL Server into CSV files via bcp.
# @file: bcp-dump.sh
server="sql.example.com" # Change this.
```

```
user="USER" # Change this.
pass="PASS" # Change this.
dbname="DBNAME" # Change this.
creds="-S '$server' -U '$user' -P '$pass' -d '$dbname'"
sqlcmd $creds -Q 'SELECT * FROM sysobjects sobjects' > objects.lst
sqlcmd $creds -Q 'SELECT * FROM information_schema.routines' > routines.lst
sqlcmd $creds -Q 'sp_tables' | tail -n +3 | head -n -2 > sp_tables.lst
sqlcmd $creds -Q 'SELECT name FROM sysobjects sobjects WHERE xtype = "U"' |
tail -n +3 | head -n -2 > tables.lst

for table in $(<tables.lst); do
  sqlcmd $creds -Q "exec sp_columns $table" > $table.desc && \
  bcp $table out $table.csv -S $server -U $user -P $pass -d $dbname -c
done
```

Share

Improve this answer

Follow

edited Feb 26, 2018 at 17:02

answered Feb 25, 2018 at 19:16

kenorb

166k ● 94 ● 706 ● 773

6 BCP allows specifying field and row delimiters, but does not escape these delimiters if they appear in the data. So it isn't any better than SQLCmd for creating CSV files. – David Ching Jan 8, 2021 at 6:15

1

An answer above almost solved it for me but it does not correctly create a parsed CSV.

Here's my version:

```
sqlcmd -S myurl.com -d MyAzureDB -E -s, -W -i mytsql.sql | findstr /V /C:"-" /B
> parsed_correctly.csv
```

Someone saying that `sqlcmd` is outdated in favor of some PowerShell alternative is forgetting that `sqlcmd` isn't just for Windows. I'm on Linux (and when on Windows I avoid PS anyway).

Having said all that, I do find `bcp` easier.

Share Improve this answer Follow

answered Mar 29, 2018 at 22:29

Hack-R

23.2k ● 15 ● 80 ● 138

0

Since following 2 reasons, you should run my solution in CMD:

1. There may be double quotes in the query
2. Login username & password is sometimes necessary to query a remote SQL Server instance

```
sqlcmd -U [your_User]  -P[your_password] -S [your_remote_Server] -d
[your_databasename]  -i "query.txt" -o "output.csv" -s"," -w 700
```

Share

Improve this answer

Follow

edited Dec 22, 2018 at 5:05

answered Dec 20, 2018 at 12:30

محسن عباسی

2,412 ● 32 ● 35

0

Try the python package `sqlcmd-csv` to postprocess the comma-separated output to valid csv.

https://github.com/shadiakiki1986/sqlcmd-csv

```
sqlcmd ... -s, ...
pip install git+https://github.com/shadiakiki1986/sqlcmd-csv.git
sqlcmd_csv out.txt out.csv
```

Share Improve this answer Follow

answered Feb 17, 2022 at 23:22

Shadi

10.3k ● 5 ● 47 ● 73

-1

You can do it in a hackish way. Careful using the `sqlcmd` hack. If the data has double quotes or commas you will run into trouble.

You can use a simple script to do it properly:

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Data Exporter                                                         '
'                                                                      '
' Description: Allows the output of data to CSV file from a SQL        '
'       statement to either Oracle, SQL Server, or MySQL               '
' Author: C. Peter Chen, http://dev-notes.com                           '
' Version Tracker:                                                     '
'       1.0   20080414 Original version                                '
'       1.1   20080807 Added email functionality                       '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
option explicit
dim dbType, dbHost, dbName, dbUser, dbPass, outputFile, email, subj, body,
smtp, smtpPort, sqlstr

''''''''''''''''''
' Configuration '
''''''''''''''''''
dbType = "oracle"                ' Valid values: "oracle", "sqlserver",
"mysql"
dbHost = "dbhost"                ' Hostname of the database server
dbName = "dbname"                ' Name of the database/SID
dbUser = "username"              ' Name of the user
dbPass = "password"              ' Password of the above-named user
```

```
outputFile = "c:\output.csv"      ' Path and file name of the output CSV fil
email = "email@me.here"           ' Enter email here should you wish to emai
the CSV file (as attachment); if no email, leave it as empty string ""
  subj = "Email Subject"          ' The subject of your email; required only
you send the CSV over email
  body = "Put a message here!"    ' The body of your email; required only if
you send the CSV over email
  smtp = "mail.server.com"        ' Name of your SMTP server; required only
you send the CSV over email
  smtpPort = 25                   ' SMTP port used by your server, usually 2
required only if you send the CSV over email
sqlStr = "select user from dual"  ' SQL statement you wish to execute
''''''''''''''''''''
' End Configuration '
''''''''''''''''''''



dim fso, conn

'Create filesystem object
set fso = CreateObject("Scripting.FileSystemObject")

'Database connection info
set Conn = CreateObject("ADODB.connection")
Conn.ConnectionTimeout = 30
Conn.CommandTimeout = 30
if dbType = "oracle" then
  conn.open("Provider=MSDAORA.1;User ID=" & dbUser & ";Password=" & dbPass &
";Data Source=" & dbName & ";Persist Security Info=False")
elseif dbType = "sqlserver" then
  conn.open("Driver={SQL Server};Server=" & dbHost & ";Database=" & dbName &
";Uid=" & dbUser & ";Pwd=" & dbPass & ";")
elseif dbType = "mysql" then
  conn.open("DRIVER={MySQL ODBC 3.51 Driver}; SERVER=" & dbHost &
";PORT=3306;DATABASE=" & dbName & "; UID=" & dbUser & "; PASSWORD=" & dbPass
"; OPTION=3")
end if

' Subprocedure to generate data.  Two parameters:
'   1. fPath=where to create the file
'   2. sqlstr=the database query
sub MakeDataFile(fPath, sqlstr)
  dim a, showList, intcount
  set a = fso.createtextfile(fPath)

  set showList = conn.execute(sqlstr)
  for intcount = 0 to showList.fields.count -1
      if intcount <> showList.fields.count-1 then
          a.write """" & showList.fields(intcount).name & ""","
      else
          a.write """" & showList.fields(intcount).name & """"
      end if
  next
  a.writeline ""

  do while not showList.eof
      for intcount = 0 to showList.fields.count - 1
          if intcount <> showList.fields.count - 1 then
              a.write """" & showList.fields(intcount).value & ""","
          else
              a.write """" & showList.fields(intcount).value & """"
```

```
            end if
        next
        a.writeline ""
        showList.movenext
    loop
    showList.close
    set showList = nothing

    set a = nothing
end sub

' Call the subprocedure
call MakeDataFile(outputFile,sqlstr)

' Close
set fso = nothing
conn.close
set conn = nothing

if email <> "" then
    dim objMessage
    Set objMessage = CreateObject("CDO.Message")
    objMessage.Subject = "Test Email from vbs"
    objMessage.From = email
    objMessage.To = email
    objMessage.TextBody = "Please see attached file."
    objMessage.AddAttachment outputFile

    objMessage.Configuration.Fields.Item
("http://schemas.microsoft.com/cdo/configuration/sendusing") = 2
    objMessage.Configuration.Fields.Item
("http://schemas.microsoft.com/cdo/configuration/smtpserver") = smtp
    objMessage.Configuration.Fields.Item
("http://schemas.microsoft.com/cdo/configuration/smtpserverport") = smtpPort

objMessage.Configuration.Fields.Update

    objMessage.Send
end if

'You're all done!!  Enjoy the file created.
msgbox("Data Writer Done!")
```

Source: Writing SQL output to CSV with VBScript.

Share

Improve this answer

Follow

edited May 29 at 19:05

TylerH

21.2k ● 76 ● 79 ● 110

answered Jan 8, 2009 at 18:50

Sarel Botha

12.7k ● 7 ● 57 ● 61

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.