How do you run a SQL Server query from PowerShell?

Asked 9 years, 2 months ago Active 1 month ago Viewed 352k times

171

Is there a way to execute an arbitrary query on a SQL Server using Powershell on my local machine?

powershell sql

39

Share Improve this question Follow

asked Dec 7 '11 at 22:27
Tigger78

8 Answers

Active | Oldest | Votes

175

For others who need to do this with just stock .NET and PowerShell (no additional SQL tools installed) here is the function that I use:

```
function Invoke-SQL {
    param(
        [string] $dataSource = ".\SQLEXPRESS",
        [string] $database = "MasterData",
        [string] $sqlCommand = $(throw "Please specify a query.")
      )

    $connectionString = "Data Source=$dataSource; " +
            "Integrated Security=SSPI; " +
            "Initial Catalog=$database"

    $connection = new-object system.data.SqlClient.SQLConnection($connectionString)
    $command = new-object system.data.sqlclient.sqlcommand($sqlCommand,$connection)
    $connection.Open()

    $adapter = New-Object System.Data.sqlclient.sqlDataAdapter $command
    $dataset = New-Object System.Data.DataSet
    $adapter.Fill($dataSet) | Out-Null

    $connection.Close()
    $dataSet.Tables

}
```

I have been using this so long I don't know who wrote which parts. This was distilled from others' examples, but simplified to be clear and just what is needed without extra dependencies or features.

I use and share this often enough that I have turned this into a script module on GitHub so that you can now go to your modules directory and execute `git clone` [illegible] and from that point forward invoke-sql will

Share Improve this answer Follow

edited Sep 16 '20 at 18:55

TylerH
18.7k 46 64 84

answered Sep 21 '13 at 17:35

Chris Magnuson
5,110 6 27 32

2 @Maslow I couldn't say for sure, I know that this works fine without disposing of the objects but if you have a single powershell.exe process that will call this multiple times over weeks without closing then it might eventually be an issue but you would have to test that. – Chris Magnuson May 28 '14 at 14:36

1 Note that this forces you to write scripts that may be vulnerable to sql injection attacks, if they depend on reading data for the query from a source that relies on user input. – Joel Coehoorn Jun 17 '14 at 21:38

1 @JoelCoehoorn could you explain a bit more for uninitiated among us? – AllTradesJack Jan 13 '17 at 20:00

4 @AllTradesJack Google Sql Injection. The Invoke-Sql command doesn't have a way to include parameters separate from the command text. This pretty much guarantees you used string concatenation to build the queries, and that's a big no-no. – Joel Coehoorn Jan 14 '17 at 1:36

2 Works well for me. For anyone wondering, to dispose an object, just add `$connection.dispose()` etc. I don't know if it makes any difference though – Nick.McDermaid Aug 15 '17 at 11:34

110

You can use the `Invoke-Sqlcmd` cmdlet

```
Invoke-Sqlcmd -Query "SELECT GETDATE() AS TimeOfQuery;" -ServerInstance
"MyComputer\MyInstance"
```

http://technet.microsoft.com/en-us/library/cc281720.aspx

Share Improve this answer Follow

answered Dec 7 '11 at 22:36

manojlds
254k 56 435 398

25 Someone should mention this may be great if you are in the context of the sql server, but not so much if you are using your workstation ... – aikeru Jun 27 '13 at 22:07

12 You can run this anywhere the SQL Server client tools (SSMS) are installed. It works fine from any workstation, whether it's running SQL Server or not. – alroc Sep 21 '13 at 17:56

3 Use the following import to have the cmdlet available: `Import-Module "sqlps" -DisableNameChecking` – xx1xx Mar 21 '14 at 5:52

1 If you're still on SQL 2008 R2 you need to use a work around module: sev17.com/2010/07/10/making-a-sqlps-module – Vincent De Smet Dec 21 '15 at 9:10

2 Invoke-SqlCmd is an endless nightmare of bizarre edge-cases and inconsistent behavior. Why is it outputting columns sometimes and not other times? Where are my error messages? Why is it on one computer or not another? How do I install it? The answer to each question is worse than the last. – Pxtl Aug 7 '18 at 21:29

32

This function will return the results of a query as an array of powershell objects so you can use them in filters and access columns easily:

```
Source=$server;Integrated Security=SSPI;Initial Catalog=$database");
    $cmd = new-object System.Data.SqlClient.SqlCommand($sqlText, $connection);

    $connection.Open();
    $reader = $cmd.ExecuteReader()

    $results = @()
    while ($reader.Read())
    {
        $row = @{}
        for ($i = 0; $i -lt $reader.FieldCount; $i++)
        {
            $row[$reader.GetName($i)] = $reader.GetValue($i)
        }
        $results += new-object psobject -property $row
    }
    $connection.Close();

    $results
}
```

Share Improve this answer Follow

answered Aug 1 '13 at 10:13

mcobrien
1,060 1 9 16

Why is this preferable over filling a `DataTable` (see Adam's answer)? – alroc Aug 1 '13 at 11:20

2 There probably isn't a huge difference, but SqlDataReaders are generally preferred because they consume less resources. That isn't likely to be relevant here but it is nice to get real objects back instead of a datatable that you can use in foreach and where clauses without worrying about the source of the data. – mcobrien Aug 1 '13 at 12:08

1 an example usage would be nice. – Eric Schneider May 24 '18 at 13:38

Sometimes there are not enough stars – Fred B May 29 '19 at 15:07

28

Here's an example I found on this blog.

```
$cn2 = new-object system.data.SqlClient.SQLConnection("Data Source=machine1;Integrated
Security=SSPI;Initial Catalog=master");
$cmd = new-object system.data.sqlclient.sqlcommand("dbcc freeproccache", $cn2);
$cn2.Open();
if ($cmd.ExecuteNonQuery() -ne -1)
{
    echo "Failed";
}
$cn2.Close();
```

Presumably you could substitute a different TSQL statement where it says `dbcc freeproccache` .

Share Improve this answer Follow

answered Dec 7 '11 at 22:34

dmc
2,476 19 23

1 Seems it returns the number of lines impacted. docs.microsoft.com/en-us/dotnet/api/… – NicolasW Feb 2 '19 at 0:55

13

If you want to do it on your ***local machine*** instead of in the context of SQL server then I would use the following. It is what we use at my company.

```
$ServerName = "_ServerName_"
$DatabaseName = "_DatabaseName_"
$Query = "SELECT * FROM Table WHERE Column = ''"

#Timeout parameters
$QueryTimeout = 120
$ConnectionTimeout = 30

#Action of connecting to the Database and executing the query and returning results if
there were any.
$conn=New-Object System.Data.SqlClient.SQLConnection
$ConnectionString = "Server={0};Database={1};Integrated Security=True;Connect Timeout=
{2}" -f $ServerName,$DatabaseName,$ConnectionTimeout
$conn.ConnectionString=$ConnectionString
$conn.Open()
$cmd=New-Object system.Data.SqlClient.SqlCommand($Query,$conn)
$cmd.CommandTimeout=$QueryTimeout
$ds=New-Object system.Data.DataSet
$da=New-Object system.Data.SqlClient.SqlDataAdapter($cmd)
[void]$da.fill($ds)
$conn.Close()
$ds.Tables
```

Just fill in the **$ServerName**, **$DatabaseName** and the **$Query** variables and you should be good to go.

I am not sure how we originally found this out, but there is something very similar here.

Share Improve this answer Follow

answered Jul 5 '13 at 22:57

Adam
441 1 5 8

13

```
Invoke-Sqlcmd -Query "sp_who" -ServerInstance . -QueryTimeout 3
```

Share Improve this answer Follow

edited Nov 9 '16 at 15:38

Bhargav Rao ♦
39.6k 26 111 127

answered Nov 9 '16 at 15:37

arnav
2,601 1 18 19

it will show number of connection in sql used by powershell command – arnav Nov 9 '16 at 19:21

There isn't a built-in "PowerShell" way of running a SQL query. If you have the SQL Server tools installed, you'll get an Invoke-SqlCmd cmdlet.

Share Improve this answer Follow

answered Aug 1 '13 at 18:46

Aaron Jensen
22.3k 14 71 82

1

To avoid SQL Injection with varchar parameters you could use

```
function sqlExecuteRead($connectionString, $sqlCommand, $pars) {

    $connection = new-object system.data.SqlClient.SQLConnection($connectionString)
    $connection.Open()
    $command = new-object system.data.sqlclient.sqlcommand($sqlCommand, $connection)

    if ($pars -and $pars.Keys) {
        foreach($key in $pars.keys) {
            # avoid injection in varchar parameters
            $par = $command.Parameters.Add("@$key", [system.data.SqlDbType]::VarChar,
512);
            $par.Value = $pars[$key];
        }
    }

    $adapter = New-Object System.Data.sqlclient.sqlDataAdapter $command
    $dataset = New-Object System.Data.DataSet
    $adapter.Fill($dataset) | Out-Null
    $connection.Close()
    return $dataset.tables[0].rows

}

$connectionString = "connectionstringHere"
$sql = "select top 10 Message, TimeStamp, Level from dbo.log " +
    "where Message = @MSG and Level like @LEVEL"
$pars = @{
    MSG = 'this is a test from powershell'
    LEVEL = 'aaa%'
};
sqlExecuteRead $connectionString $sql $pars
```

Share Improve this answer

Follow

edited Dec 8 '20 at 20:06

2b77bee6-5445-4c77-b1eb-4df3e5
722 12 24

answered May 28 '20 at 18:16

Piero
194 12

I can't image a scenario where a PS script would be susceptible to SQL Injection, but I prefer this to manually building the query. I think it is more readable. Thanks! – 2b77bee6-5445-4c77-b1eb-4df3e5 Dec 9 '20 at 21:38