MyWebUniversity
python
PowerShell Get-Help cmdlet
PS C:\> Get-Help
FPDF Library
PDF generator
FPDF
Full credit is given to all the above companies including the Operating System that this PDF file was generated!

Windows PowerShell Get-Help on Cmdlet 'Invoke-Sqlcmd'

PS:\>Get-HELP Invoke-Sqlcmd -Full

NAME

Invoke-Sqlcmd

SYNOPSIS

Runs a script containing statements supported by the SQL Server SQLCMD utility.

SYNTAX

Invoke-Sqlcmd [[-Query] <String>] [-AbortOnError] [-AccessToken <String>] [-ApplicationIntent {ReadWrite | ReadOnly}] [-ApplicationName <String>] [-ConnectionTimeout

<Int32>] [-Credential <PSCredential>] [-Database <String>] [-DedicatedAdministratorConnection] [-DisableCommands] [-DisableVariables] [-Encrypt {Mandatory | Optional

| Strict}] [-EncryptConnection] [-ErrorLevel <Int32>] [-FailoverPartner <String>] [-HostName <String>] [-HostNameInCertificate <String>] [-IgnoreProviderContext]

[-IncludeSqlUserErrors] [-InputFile <String>] [-MaxBinaryLength <Int32>] [-MaxCharLength <Int32>] [-MultiSubnetFailover] [-NewPassword <String>] [-OutputAs {DataSet |

DataTables | DataRows}] [-OutputSqlErrors <Boolean>] [-Password <String>] [-ProgressAction <ActionPreference>] [-QueryTimeout <Int32>] [-ServerInstance <PSObject>]

[-SeverityLevel <Int32>] [-StatisticsVariable <String>] [-SuppressProviderContextWarning] [-TrustServerCertificate] [-Username <String>] [-Variable <PSObject>]

[<CommonParameters>]

Invoke-Sqlcmd [[-Query] <String>] [-AbortOnError] [-AccessToken <String>] -ConnectionString <String> [-DisableCommands] [-DisableVariables] [-ErrorLevel <Int32>]

[-IncludeSqlUserErrors] [-InputFile <String>] [-KeyVaultAccessToken <String>] [-ManagedHsmAccessToken <String>] [-MaxBinaryLength <Int32>] [-MaxCharLength <Int32>]

[-OutputAs {DataSet | DataTables | DataRows}] [-OutputSqlErrors <Boolean>] [-ProgressAction <ActionPreference>] [-QueryTimeout <Int32>] [-SeverityLevel <Int32>]

[-StatisticsVariable <String>] [-Variable <PSObject>] [<CommonParameters>]

DESCRIPTION

The Invoke-Sqlcmd cmdlet runs a script containing the languages and commands supported by the SQL Server SQLCMD utility.

The commands supported are Transact-SQL statements and the subset of the XQuery syntax that is supported by the database engine.

This cmdlet also accepts many of the commands supported natively by SQLCMD, such as GO and QUIT.

This cmdlet also accepts the SQLCMD scripting variables, such as SQLCMDUSER. By default, this cmdlet does not set SQLCMD scripting variables.

This cmdlet does not support the use of commands that are primarily related to interactive script editing.

The commands not supported include :!!, :connect, :error, :out, :ed, :list, :listvar, :reset, :perftrace, and :serverlist.

When this cmdlet is run, the first result set that the script returns is displayed as a formatted table.

If subsequent result sets contain different column lists than the first, those result sets are not displayed.

If subsequent result sets after the first set have the same column list, their rows are appended to the formatted table that contains the rows that were returned by the first result set.

You can display SQL Server message output, such as those that result from the SQL PRINT statement, by specifying the Verbose parameter.

PARAMETERS

-AbortOnError [<SwitchParameter>]

Indicates that this cmdlet stops the SQL Server command and returns an error level to the Windows PowerShell ERRORLEVEL variable if this cmdlet encounters an error.

The error level returned is 1 if the error has a severity higher than 10, and the error level is 0 if the error has a severity of 10 or less.

If the ErrorLevel parameter is also specified, this cmdlet returns 1 only if the error message severity is also equal to or higher than the value specified for ErrorLevel.

Required?	false
Position?	named
Default value	False
Accept pipeline input?	False
Accept wildcard characters?	false

-AccessToken <String>

The access token used to authenticate to SQL Server, as an alternative to user/password or Windows Authentication.

This can be used, for example, to connect to `SQL Azure DB` and `SQL Azure Managed Instance` using a `Service Principal` or a `Managed Identity` (see references at the bottom of this page)

In common scenarios, this parameter is obtained with something like `(Get-AzAccessToken -ResourceUrl https://database.windows.net).Token` (requires the Az.Account

    module)

    Do not specify UserName , Password , or Credential when using this parameter.

```
    Required?                    false
    Position?                    named
    Default value                None
    Accept pipeline input?       False
    Accept wildcard characters?  false
```

-ApplicationIntent <ApplicationIntent>

    The application workload type when connecting to a database in an SQL Server Availability Group.

    Allowed values are: ReadOnly and ReadWrite.

```
    Required?                    false
    Position?                    named
    Default value                ReadWrite
    Accept pipeline input?       False
    Accept wildcard characters?  false
```

-ApplicationName <String>

    The name of the application associated with the connection.

```
    Required?                    false
    Position?                    named
    Default value                .NET SqlClient Data Provider
    Accept pipeline input?       False
    Accept wildcard characters?  false
```

-ConnectionString <String>

    Specifies a connection string to connect to the server.

Required?   true

Position?   named

Default value   None

Accept pipeline input?   False

Accept wildcard characters? false

-ConnectionTimeout <Int32>

Specifies the number of seconds when this cmdlet times out if it cannot successfully connect to an instance of the Database Engine. The timeout value must be an

integer value between 0 and 65534. If 0 is specified, connection attempts do not time out.

Required?   false

Position?   named

Default value   None

Accept pipeline input?   False

Accept wildcard characters? false

-Credential <PSCredential>

The PSCredential object whose Username and Password fields will be used to connect to the SQL instance.

Required?   false

Position?   named

Default value   None

Accept pipeline input?   False

Accept wildcard characters? false

-Database <String>

Specifies the name of a database. This cmdlet connects to this database in the instance that is specified in the ServerInstance parameter.

If the Database parameter is not specified, the database that is used depends on whether the current path specifies both the SQLSERVER:\SQL folder and a database name. If the path specifies both the SQL folder and a database name, this cmdlet connects to the database that is specified in the path. If the path is not based

on the SQL folder, or the path does not contain a database name, this cmdlet connects to the default database for the current login ID. If you specify the

IgnoreProviderContext parameter switch, this cmdlet does not consider any database specified in the current path, and connects to the database defined as the

default for the current login ID.

```
Required?                    false
Position?                    named
Default value                None
Accept pipeline input?       False
Accept wildcard characters?  false
```

-DedicatedAdministratorConnection [<SwitchParameter>]

Indicates that this cmdlet uses a Dedicated Administrator Connection (DAC) to connect to an instance of the Database Engine.

DAC is used by system administrators for actions such as troubleshooting instances that will not accept new standard connections.

The instance must be configured to support DAC.

If DAC is not enabled, this cmdlet reports an error and will not run.

```
Required?                    false
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-DisableCommands [<SwitchParameter>]

Indicates that this cmdlet turns off some sqlcmd features that might compromise security when run in batch files.

It prevents Windows PowerShell variables from being passed in to the Invoke-Sqlcmd script.

The startup script specified in the SQLCMDINI scripting variable is not run.

```
Required?                    false
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-DisableVariables [<SwitchParameter>]

Indicates that this cmdlet ignores sqlcmd scripting variables. This is useful when a script contains many INSERT statements that may contain strings that have the

same format as variables, such as $(variable_name).

```
Required?                    false
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-Encrypt <String>

The encryption type to use when connecting to SQL Server.

This value maps to the `Encrypt` property `SqlConnectionEncryptOption` on the SqlConnection object of the Microsoft.Data.SqlClient driver.

When not specified, the default value is `Mandatory`.

> This parameter is new in v22 of the module. For more details, see `Strict Connection Encryption` under Related Links (#related-links).

Required?   false

Position?   named

Default value   None

Accept pipeline input?   False

Accept wildcard characters? false

-EncryptConnection [<SwitchParameter>]

Indicates that this cmdlet uses Secure Sockets Layer (SSL/TLS) encryption for the connection to the instance of the Database Engine specified in the

ServerInstance parameter.

> Starting in v22 of the module, this parameter is deprecated. Connections are encrypted by default. Please, consider using the new -Encrypt parameter instead.

For more details, see `Strict Connection Encryption` under Related Links (#related-links).

Required?   false

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-ErrorLevel <Int32>

Specifies that this cmdlet display only error messages whose severity level is equal to or higher than the value specified. All error messages are displayed if

this parameter is not specified or set to 0. Database Engine error severities range from 1 to 24.

Required?   false

Position?   named

Default value   None

Accept pipeline input?   False

Accept wildcard characters? false

-FailoverPartner <String>

The name or address of the partner server to connect to if the primary server is down.

Required? false

Position? named

Default value ""

Accept pipeline input? False

Accept wildcard characters? false

-HostName <String>

Specifies a workstation name. The workstation name is reported by the sp_who system stored procedure and in the hostname column of the sys.processes catalog view.

If this parameter is not specified, the default is the name of the computer on which Invoke-Sqlcmd is run. This parameter can be used to identify different

Invoke-Sqlcmd sessions.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-HostNameInCertificate <String>

The host name to be used in validating the SQL Server TLS/SSL certificate. You must pass this parameter if your SQL Server instance is enabled for Force

Encryption and you want to connect to an instance using hostname/shortname. If this parameter is omitted then passing the Fully Qualified Domain Name (FQDN) to

-ServerInstance is necessary to connect to a SQL Server instance enabled for Force Encryption.

> This parameter is new in v22 of the module. For more details, see `Strict Connection Encryption` under Related Links (#related-links).

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-IgnoreProviderContext [<SwitchParameter>]

Indicates that this cmdlet ignores the database context that was established by the current SQLSERVER:\SQL path. If the Database parameter is not specified, this

cmdlet uses the default database for the current login ID or Windows account.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-IncludeSqlUserErrors [<SwitchParameter>]

Indicates that this cmdlet returns SQL user script errors that are otherwise ignored by default. If this parameter is specified, this cmdlet matches the default

behavior of the sqlcmd utility.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-InputFile <String>

Specifies a file to be used as the query input to this cmdlet. The file can contain Transact-SQL statements, XQuery statements, and sqlcmd commands and scripting

variables. Specify the full path to the file. Spaces are not allowed in the file path or file name. The file is expected to be encoded using UTF-8.

You should only run scripts from trusted sources. Ensure all input scripts are secured with the appropriate NTFS permissions.

```
        Required?                    false
        Position?                    named
        Default value                None
        Accept pipeline input?       False
        Accept wildcard characters?  false
```

-KeyVaultAccessToken <String>

Specifies an access token for key vaults in Azure Key Vault. Use this parameter if any column to be queried is protected with Always Encrypted using a column

master key stored in a key vault in Azure Key Vault. Alternatively, you can authenticate to Azure with Add-SqlAzureAuthenticationContext before calling this

cmdlet.

```
        Required?                    false
        Position?                    named
        Default value                None
        Accept pipeline input?       False
        Accept wildcard characters?  false
```

-ManagedHsmAccessToken <String>

Specifies an access token for managed HSMs in Azure Key Vault. Use this parameter if any column to be queried is protected with Always Encrypted using a column

master key stored in a managed HSM in Azure Key Vault. Alternatively, you can authenticate to Azure with Add-SqlAzureAuthenticationContext before calling this

cmdlet.

```
        Required?                    false
        Position?                    named
        Default value                None
        Accept pipeline input?       False
        Accept wildcard characters?  false
```

-MaxBinaryLength <Int32>

Specifies the maximum number of bytes returned for columns with binary string data types, such as binary and varbinary. The default value is 1,024 bytes.

```
Required?                    false
Position?                    named
Default value                None
Accept pipeline input?       False
Accept wildcard characters?  false
```

-MaxCharLength <Int32>

Specifies the maximum number of characters returned for columns with character or Unicode data types, such as char, nchar, varchar, and nvarchar. The default

value is 4,000 characters.

```
Required?                    false
Position?                    named
Default value                None
Accept pipeline input?       False
Accept wildcard characters?  false
```

-MultiSubnetFailover [<SwitchParameter>]

If your application is connecting to an AlwaysOn Availability Group (AG) on different subnets, passing this parameter provides faster detection of and connection

to the (currently) active server.

Note: passing -MultiSubnetFailover isn't required with .NET Framework 4.6.1 or later versions.

```
Required?                    false
Position?                    named
Default value                False
Accept pipeline input?       False
```

Accept wildcard characters? false

-NewPassword <String>

Specifies a new password for a SQL Server Authentication login ID. This cmdlet changes the password and then exits. You must also specify the Username and

Password parameters, with Password that specifies the current password for the login.

Required?   false

Position?   named

Default value   None

Accept pipeline input?   False

Accept wildcard characters? false

-OutputAs <OutputType>

Specifies the type of the results this cmdlet gets.

If you do not specify a value for this parameter, the cmdlet sets the value to DataRows.

Required?   false

Position?   named

Default value   None

Accept pipeline input?   False

Accept wildcard characters? false

-OutputSqlErrors <Boolean>

Indicates that this cmdlet displays error messages in the Invoke-Sqlcmd output.

Required?   false

Position?   named

Default value   None

Accept pipeline input?   False

Accept wildcard characters? false

-Password <String>

Specifies the password for the SQL Server Authentication login ID that was specified in the Username parameter. Passwords are case-sensitive. When possible, use

Windows Authentication. Do not use a blank password, when possible use a strong password.

If you specify the Password parameter followed by your password, the password is visible to anyone who can see your monitor.

If you code Password followed by your password in a .ps1 script, anyone reading the script file will see your password.

Assign the appropriate NTFS permissions to the file to prevent other users from being able to read the file.

Required?	false
Position?	named
Default value	None
Accept pipeline input?	False
Accept wildcard characters?	false

-ProgressAction <ActionPreference>

Determines how PowerShell responds to progress updates generated by a script, cmdlet, or provider, such as the progress bars generated by the Write-Progress

cmdlet. The Write-Progress cmdlet creates progress bars that show a command's status.

Required?	false
Position?	named
Default value	None
Accept pipeline input?	False
Accept wildcard characters?	false

-Query <String>

Specifies one or more queries that this cmdlet runs. The queries can be Transact-SQL or XQuery statements, or sqlcmd commands. Multiple queries separated by a

semicolon can be specified. Do not specify the sqlcmd GO separator. Escape any double quotation marks included in the string. Consider using bracketed identifiers

such as [MyTable] instead of quoted identifiers such as "MyTable".

```
Required?                    false
Position?                    0
Default value                None
Accept pipeline input?       False
Accept wildcard characters?  false
```

-QueryTimeout <Int32>

Specifies the number of seconds before the queries time out. If a timeout value is not specified, the queries do not time out. The timeout must be an integer

value between 1 and 65535.

```
Required?                    false
Position?                    named
Default value                None
Accept pipeline input?       False
Accept wildcard characters?  false
```

-ServerInstance <PSObject>

Specifies a character string or SQL Server Management Objects (SMO) object that specifies the name of an instance of the Database Engine. For default instances,

only specify the computer name: MyComputer. For named instances, use the format ComputerName\InstanceName.

```
Required?                    false
Position?                    named
Default value                None
Accept pipeline input?       True (ByValue)
Accept wildcard characters?  false
```

-SeverityLevel <Int32>

Specifies the lower limit for the error message severity level this cmdlet returns to the ERRORLEVEL Windows

PowerShell variable.

This cmdlet returns the highest severity level from the error messages generated by the queries it runs, provided that severity is equal to or higher than

specified in the SeverityLevel parameter.

If SeverityLevel is not specified or set to 0, this cmdlet returns 0 to ERRORLEVEL.

The severity levels of Database Engine error messages range from 1 to 24.

This cmdlet does not report severities for informational messages that have a severity of 10

Required?	false
Position?	named
Default value	None
Accept pipeline input?	False
Accept wildcard characters?	false

-StatisticsVariable <String>

Specify the name of a PowerShell variable that will be assigned the SQL Server run-time statistics when the cmdlet is executed.

Common use for this parameter is to capture the `ExecutionTime` (the cumulative amount of time (in milliseconds) that the provider has spent processing the

cmdlet), or `IduRows` (the total number of rows affected by INSERT, DELETE, and UPDATE statements).

For more details, see Provider Statistics for SQL Server (/dotnet/framework/data/adonet/sql/provider-statistics-for-sql-server).

Required?	false
Position?	named
Default value	None
Accept pipeline input?	False

Accept wildcard characters? false

-SuppressProviderContextWarning [<SwitchParameter>]

Indicates that this cmdlet suppresses the warning that this cmdlet has used in the database context from the current SQLSERVER:\SQL path setting to establish the

database context for the cmdlet.

Required?   false

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-TrustServerCertificate [<SwitchParameter>]

Indicates whether the channel will be encrypted while bypassing walking the certificate chain to validate trust.

> This parameter is new in v22 of the module. For more details, see `Strict Connection Encryption` under Related Links (#related-links).

Required?   false

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-Username <String>

Specifies the login ID for making a SQL Server Authentication connection to an instance of the Database Engine.

The password must be specified through the Password parameter.

If Username and Password are not specified, this cmdlet attempts a Windows Authentication connection using the Windows account running the Windows PowerShell

session. When possible, use Windows Authentication.

Required?   false

Position?   named

Default value   None

Accept pipeline input?   False

Accept wildcard characters? false

-Variable <PSObject>

Specifies a set of sqlcmd scripting variables for use in the sqlcmd script, and sets a values for the variables.

Use a Windows PowerShell array to specify multiple variables and their values; alternatively, use a `Hashtable` where the key represent the variable name and the

value the variable value.

> When using an array, parameter values are trimmed. This behavior was kept in v22 of the module for backward compatibility with v21. It is recommended not to

rely on this behavior, which may change in a future major version of the module.

> The parameter of type `Hashtable` is only available in v22+ of the module.

Required?   false

Position?   named

Default value   None

Accept pipeline input?   False

Accept wildcard characters? false

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug,

ErrorAction, ErrorVariable, WarningAction, WarningVariable,

OutBuffer, PipelineVariable, and OutVariable. For more information, see

about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

INPUTS

System.Management.Automation.PSObject

OUTPUTS

System.Object

NOTES

--- Example 1: Connect to a named instance and run a script ---

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

TimeOfQuery
-----------
9/21/2017 2:48:24 PM
```

This command connects to a named instance of the SQL Database Engine on a computer and runs a basic Transact-SQL script.

Example 2: Invoke commands in a script file and save the output in a text file

```
Invoke-Sqlcmd -InputFile "C:\ScriptFolder\TestSqlCmd.sql" | Out-File -FilePath "C:\ScriptFolder\TestSqlCmd.rpt"
Output sent to TestSqlCmd.rpt.
```

This command reads a file containing Transact-SQL statements and SQLCMD commands, runs the file, and writes the output to another file.

The output file may contain proprietary information, so you should secure the output files with the appropriate NTFS permissions.

Example 3: Invoke a script and pass in variable values from a string

```
$StringArray = "MYVAR1='String1'", "MYVAR2='String2'"
Invoke-Sqlcmd -Query "SELECT `$(MYVAR1) AS Var1, `$(MYVAR2) AS Var2" -Variable $StringArray

Var1     Var2
----     ----
String1  String2
```

This command uses an array of character strings as input to the Variable parameter.

The array defines multiple SQLCMD variables.

The $ signs in the SELECT statement that identify the SQLCMD variables are escaped using the back-tick (`) character.

Example 4: Invoke a script and pass in variables from the SQL database engine

```
Set-Location "SQLSERVER:\SQL\MyComputer\MainInstance"
PS SQLSERVER:\SQL\MyComputer\MainInstance> Invoke-Sqlcmd -Query "SELECT SERVERPROPERTY('MachineName') AS ComputerName" -ServerInstance (Get-Item .)

ComputerName
------------
MyComputer
```

This command uses Set-Location to navigate to the SQL ServerWindows PowerShell provider path for an instance of the SQL Database Engine.

Then it calls Get-Item to retrieve a SQL Management Object Server object for use as the ServerInstance parameter of Invoke-Sqlcmd.

------ Example 5: Run a query and display verbose output ------

```
Set-Location "SQLSERVER:\SQL\MyComputer\MainInstance"
```

```
Invoke-SqlCmd -Query "PRINT N'abc'" -Verbose
VERBOSE: abc
```

This command uses the Windows PowerShellVerbose parameter to return the message output of the SQL PRINT command.

Example 6: Invoke a command using a positional string as input

```
Set-Location "SQLSERVER:\SQL\MyComputer\MainInstance\Databases\MyDatabase"
PS SQLSERVER:\SQL\MyComputer\MainInstance> Invoke-Sqlcmd "SELECT DB_NAME() AS DatabaseName"
WARNING: Using provider context. Server = MyComputer\MainInstance, Database = MyDatabase.

DatabaseName
------------
MyDatabase
```

This command uses a positional string to supply the input to the Query parameter.

It also demonstrates how Invoke-Sqlcmd uses the current path to set the database context to MyDatabase.

-------- Example 7: Capture data into a DataSet object --------

```
$DS = Invoke-Sqlcmd -ServerInstance "MyComputer" -Query "SELECT  ID, Item FROM MyDB.dbo.MyTable" -As DataSet
$DS.Tables[0].Rows | %{ echo "{ $($_['ID']), $($_['Item']) }" }

{ 10, AAA }
{ 20, BBB }
{ 30, CCC }
```

This command uses the As DataSet parameter to capture the data into a .Net System.Data.DataSet object and stores the result in the variable '$DS'. The object can be
used for further processing.

------------- Example 8: Get specific column sets -------------

```
$Tables = Invoke-Sqlcmd -ServerInstance "MyComputer" -Query "SELECT  Item, id FROM MyDatabase.dbo.MyTable; SELECT GETDATE() AS T" -As DataTables
$Tables[0].Rows | %{ echo $_.ID }
$Tables[1].Rows | %{ echo $_.T.DayOfWeek }


10
20
30

Monday
```

The first command uses the As DataTables parameter to capture the data into a collection of .Net System.Data.DataTable objects. The command gets two tables with different column sets.

Each table can be processed individually, based on its own schema.

--------- Example 9: Gain full control of a connection ---------

```
Invoke-Sqlcmd -Query "SELECT COUNT(*) AS Count FROM MyTable" -ConnectionString "Data Source=MYSERVER;Initial Catalog=MyDatabase;Integrated
Security=True;ApplicationIntent=ReadOnly"
Count
-----
127432
```

This command users the -ConnectionString parameter to gain full control of the connection that this cmdlet establishes, instead of the Invoke-Sqlcmd to build the connection string based on the parameters passed on the command line.

This is useful for less-common properties that you may want to use.

Example 10: Execute a stored procedure and capture the SQL errors

```
$script_sp_with_errors = @'
```

```
CREATE PROCEDURE [dbo].[TestProcedure3]
AS
BEGIN
  CREATE TABLE [dbo].[TestTable] (col INT NOT NULL);
  INSERT INTO [dbo].[TestTable] VALUES (NULL); -- will cause an error
END
GO
'@

# Create a test database
Invoke-SqlCmd -ServerInstance MyServer -Query 'CREATE DATABASE TestDB'
# ... adds a stored procedure that has errors in it...
Invoke-SqlCmd -ServerInstance MyServer -Database 'TestDB' -Query $script_sp_with_errors
# ... executes the SP and collected the errors
Invoke-SqlCmd -ServerInstance MyServer -Database 'TestDB' -Query 'EXEC TestProcedure3' -OutputSqlErrors $true
```

Here's the output:

```
 Invoke-SqlCmd : Cannot insert the value NULL into column 'col', table 'TestDB.dbo.TestTable'; column does not allow nulls. INSERT fails.
The statement has been terminated.
 Msg 515, Level 16, State 2, Procedure TestProcedure3, Line 5.
At line:1 char:1
...
```

This command users the -OutputSqlErrors parameter to report the errors to the user. Note that the error message in this case provides extra information like the SP
name and the line number where the error occurred.

Example 11: Connect to Azure SQL Database (or Managed Instance) using an Access Token

```
Import-Module SQLServer
Import-Module Az.Accounts -MinimumVersion 2.2.0

# Note: the sample assumes that you or your DBA configured the server to accept connections using
#       that Service Principal and has granted it access to the database (in this example at least
#       the SELECT permission).

### Obtain the Access Token: this will bring up the login dialog
Connect-AzAccount
$access_token = (Get-AzAccessToken -ResourceUrl https://database.windows.net).Token

# Now that we have the token, we use it to connect to the database 'mydb' on server 'myserver'
Invoke-Sqlcmd -ServerInstance myserver.database.windows.net -Database mydb -AccessToken $access_token`
              -query 'select * from Table1'
```

Example 12: Connect to Azure SQL Database (or Managed Instance) using a Service Principal

```
Import-Module SQLServer

# Note: the sample assumes that you or your DBA configured the server to accept connections using
#       that Service Principal and has granted it access to the database (in this example at least
#       the SELECT permission).

$clientid = "enter application id that corresponds to the Service Principal" # Do not confuse with its display name
$tenantid = "enter the tenant ID of the Service Principal"
$secret = "enter the secret associated with the Service Principal"

$request = Invoke-RestMethod -Method POST `
           -Uri "https://login.microsoftonline.com/$tenantid/oauth2/token"`
                         -Body  @{  resource="https://database.windows.net/";  grant_type="client_credentials";  client_id=$clientid;
client_secret=$secret }`
           -ContentType "application/x-www-form-urlencoded"
$access_token = $request.access_token

# Now that we have the token, we use it to connect to the database 'mydb' on server 'myserver'
Invoke-Sqlcmd -ServerInstance myserver.database.windows.net -Database mydb -AccessToken $access_token
```

```
        -query 'select * from Table1'
```

Example 13: Connect to Azure SQL Database (or Managed Instance) using a System Assigned Managed Identity (SAMI)

```
Import-Module SQLServer

# Note: the sample assumes that you or your DBA configured the server to accept connections using
#       that VM Identity you are running on and has granted it access to the database (in this
#       example at least the SELECT permission).

Connect-AzAccount -Identity
$access_token = (Get-AzAccessToken -ResourceUrl https://database.windows.net).Token

# Now that we have the token, we use it to connect to the database 'mydb' on server 'myserver'
Invoke-Sqlcmd -ServerInstance myserver.database.windows.net -Database mydb -AccessToken $access_token `
        -query 'select * from Table1'
```

Example 14: Connect to Azure SQL Database (or Managed Instance) using a User Assigned Managed Identity (UAMI)

```
Import-Module SQLServer

# Note: the sample assumes that you or your DBA configured the server to accept connections using
#       that VM Identity you are running on and has granted it access to the database (in this
#       example at least the SELECT permission).

Connect-AzAccount -Identity -AccountId '<your-user-assigned-managed-identity-client-id>'
$access_token = (Get-AzAccessToken -ResourceUrl https://database.windows.net).Token

# Now that we have the token, we use it to connect to the database 'mydb' on server 'myserver'
Invoke-Sqlcmd -ServerInstance myserver.database.windows.net -Database mydb -AccessToken $access_token `
        -query 'select * from Table1'
```

Example 15: Connect to an Availability Group configured for Read-Only Routing using -ApplicationIntent

```
# In the following example:
# - MT_2009250511 is a listener for an AG configured for Read-Only Routing (port 5555)
# - AGDB_2_1 is the DB in the AG
# - VLM00226138 is the primary replica configured to only allow ReadWrite connections
# - VLM00226137 is the secondary replica
#
Invoke-Sqlcmd -ServerInstance "MT_2009250511,5555" -Database AGDB_2_1 `
    -HostName "PowershellBox1" -ApplicationName "ReadWrite" -ApplicationIntent ReadWrite `
        -Query "select HOST_NAME() AS HostName, APP_NAME() AS ApplicationIntent, @@SERVERNAME AS
ServerName"

Invoke-Sqlcmd -ServerInstance "MT_2009250511,5555" -Database AGDB_2_1 `
    -HostName "PowershellBox2" -ApplicationName "ReadOnly" -ApplicationIntent ReadOnly `
        -Query "select HOST_NAME() AS HostName, APP_NAME() AS ApplicationIntent, @@SERVERNAME AS
ServerName"

# When you run the 2 cmdlets above, the output is going to be something like this:
#
# HostName       ApplicationIntent ServerName
# --------       ----------------- ----------
# PowershellBox1 ReadWrite         VLM00226138
#
# HostName       ApplicationIntent ServerName
# --------       ----------------- ----------
# PowershellBox2 ReadOnly          VLM00226137
```

which shows that, depending on the value of the `-ApplicationIntent` parameter, the connection is routed to a different server in the AG. Incidentally, observe the

uses of the `-ApplicationName` and `-HostName` parameters to visually differentiate the two results: this is a common technique that can be used to trace connections

and their intents, beyond the -ApplicationIntent example illustrated here.

Example 16: Capture connection statistics via -StatisticsVariable parameter

```
Import-Module SQLServer
Invoke-Sqlcmd -ServerInstance localhost -StatisticsVariable stats `
                      -Query 'CREATE TABLE #Table (ID int); INSERT INTO #Table VALUES(1), (2); INSERT INTO #Table VALUES(3); SELECT * FROM #Table'

Write-Host "Number of rows affected......: $($stats.IduRows)"
Write-Host "Number of insert statements..: $($stats.IduCount)"
Write-Host "Number of select statements..: $($stats.SelectCount)"
Write-Host "Total execution time.........: $($stats.ExecutionTime)ms"

# When you run the code fragment above, is going to be something like this:
#
# Number of rows affected......: 3
# Number of insert statements..: 2
# Number of select statements..: 1
# Total execution time.........: 5ms
```

This example shows how to use the `-StatisticsVariable` parameter to capture informations about the connection, the statements executed, and the execution time when

running some T-SQL that creates a temporary table, insert some value, and finally issues a select to get all the inserted rows.

Note: when the same query is executed against multiple servers (e.g. by piping the server names thru the cmdlet), the `StatisticsVariable` captures an array of

statistics, one for each connection. Results can then be aggregated by using, for example, `($stats.IduRows | Measure-Object -Sum).Sum`.

Refer to Provider Statistics for SQL Server (/dotnet/framework/data/adonet/sql/provider-statistics-for-sql-server)for a more information about the available statistics.

Example 17: Run a query that decrypts data retrieved from columns encrypted using Always Encrypted. Assume the column master key is stored in a key vault in Azure Key Vault.

```
# Connect to Azure account.
Import-Module Az.Accounts -MinimumVersion 2.2.0
Connect-AzAccount

#?Obtain?an?access?token for key vaults.
$keyVaultAccessToken?=?(Get-AzAccessToken?-ResourceUrl?https://vault.azure.net).Token

# Pass the token to the cmdlet, so that it can use it to authenticate to Azure when decrypting data protected with Always Encrypted.
$connString = 'Data?Source=MYSERVER;Initial?Catalog=MyDatabase;Integrated?Security=True;ApplicationIntent=ReadOnly;Column?Encryption?Setting=Enabled'
Invoke-Sqlcmd?-Query?'SELECT?COUNT(*)?AS?Count?FROM?MyTable' -ConnectionString?$connString -KeyVaultAccessToken?$keyVaultAccessToken
```

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/sqlserver/invoke-sqlcmd

SQLServer_Cmdlets

Service Principal

Managed Identity

High Availability

Provider Statistics for SQL Server

Strict Connection Encryption