#######################   
<#   
.SYNOPSIS   
Runs a T-SQL script.   
.DESCRIPTION   
Runs a T-SQL script. Invoke-Sqlcmd2 only returns message output, such as the output of PRINT statements when -verbose parameter is specified   
.INPUTS   
None   
 You cannot pipe objects to Invoke-Sqlcmd2   
.OUTPUTS   
 System.Data.DataTable   
.EXAMPLE   
Invoke-Sqlcmd2 -ServerInstance "MyComputer\MyInstance" -Query "SELECT login\_time AS 'StartTime' FROM sysprocesses WHERE spid = 1"   
This example connects to a named instance of the Database Engine on a computer and runs a basic T-SQL query.   
StartTime   
-----------   
2010-08-12 21:21:03.593   
.EXAMPLE   
Invoke-Sqlcmd2 -ServerInstance "MyComputer\MyInstance" -InputFile "C:\MyFolder\tsqlscript.sql" | Out-File -filePath "C:\MyFolder\tsqlscript.rpt"   
This example reads a file containing T-SQL statements, runs the file, and writes the output to another file.   
.EXAMPLE   
Invoke-Sqlcmd2 -ServerInstance "MyComputer\MyInstance" -Query "PRINT 'hello world'" -Verbose   
This example uses the PowerShell -Verbose parameter to return the message output of the PRINT command.   
VERBOSE: hello world   
.NOTES   
Version History   
v1.0 - Chad Miller - Initial release   
v1.1 - Chad Miller - Fixed Issue with connection closing   
v1.2 - Chad Miller - Added inputfile, SQL auth support, connectiontimeout and output message handling. Updated help documentation   
v1.3 - Chad Miller - Added As parameter to control DataSet, DataTable or array of DataRow Output type   
#>   
**function** Invoke-Sqlcmd2   
{   
 [CmdletBinding()]   
 **param**(   
 [Parameter(Position=0, Mandatory=$true)] [string]$ServerInstance,   
 [Parameter(Position=1, Mandatory=$false)] [string]$Database,   
 [Parameter(Position=2, Mandatory=$false)] [string]$Query,   
 [Parameter(Position=3, Mandatory=$false)] [string]$Username,   
 [Parameter(Position=4, Mandatory=$false)] [string]$Password,   
 [Parameter(Position=5, Mandatory=$false)] [Int32]$QueryTimeout=600,   
 [Parameter(Position=6, Mandatory=$false)] [Int32]$ConnectionTimeout=15,   
 [Parameter(Position=7, Mandatory=$false)] [ValidateScript({**test-path** $\_})] [string]$InputFile,   
 [Parameter(Position=8, Mandatory=$false)] [ValidateSet("DataSet", "DataTable", "DataRow")] [string]$As="DataRow"   
 )   
   
 **if** ($InputFile)   
 {   
 $filePath = $(**resolve-path** $InputFile).path   
 $Query = [System.IO.File]::ReadAllText("$filePath")   
 }   
   
 $conn=**new-object** System.**Data**.SqlClient.SQLConnection   
   
 **if** ($Username)   
 { $ConnectionString = "Server={0};Database={1};User ID={2};Password={3};Trusted\_Connection=False;Connect Timeout={4}" -f $ServerInstance,$Database,$Username,$Password,$ConnectionTimeout }   
 **else**   
 { $ConnectionString = "Server={0};Database={1};Integrated Security=True;Connect Timeout={2}" -f $ServerInstance,$Database,$ConnectionTimeout }   
   
 $conn.ConnectionString=$ConnectionString   
   
 #Following EventHandler is used for PRINT and RAISERROR T-SQL statements. Executed when -Verbose parameter specified by caller   
 **if** ($PSBoundParameters.Verbose)   
 {   
 $conn.FireInfoMessageEventOnUserErrors=$true   
 $handler = [System.**Data**.SqlClient.SqlInfoMessageEventHandler] {**Write-Verbose** "$($\_)"}   
 $conn.add\_InfoMessage($handler)   
 }   
   
 $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)   
 $conn.Close()   
 **switch** ($As)   
 {   
 'DataSet' { Write-Output ($ds) }   
 'DataTable' { Write-Output ($ds.Tables) }   
 'DataRow' { Write-Output ($ds.Tables[0]) }   
 }   
   
} #Invoke-Sqlcmd2