```
<#
.SYNOPSIS
   This files disables people who are out of security compliance training
requirements. AD is not affected.

.DESCRIPTION
    This file connects to a database to get a list of people who are not in
compliance with security awareness training.
    If they are not in compliance with training, their email access (OWA, MAPI,
ActiveSync) will be disabled.
    This file does not affect AD.

    The database is maintained by the SOLE group, and is populated at 7:07 am
every day. This file runs
    as a scheduled task at 7:15 am every day.

.PARAMETER sqlPasswordPath
  Stores encrypted pasword used to connect to the SOLE database

.PARAMETER MaximumDisables
  This is a safety parameter to make sure too many accounts aren't disabled.

.PARAMETER DisableAccounts
  Specifies whether accounts that are actually disabled

.PARAMETER SQLServer
  The ip address of the sql server

.PARAMETER DBName
  DB name to query

.PARAMETER DBUserName
  Username to use when executing query

.PARAMETER DBTableName
  Table name to query

.NOTES
  Author: Jeff Brusoe
    Last Updated by: Jeff Brusoe
  Last Updated: September 1, 2020

  Version Updates:
    - March 25, 2020
      - Cleaned up file output
      - Modified to work with GitHub directory
      - Added new common code parameters
      - Removed old commmon code to initialize environment
      - Added new common code to initialize environment
      - Used SQL Module for SQL function calls

    - October 14, 2019
      - Changed DB calls to use parameters

    - March 25, 2020
      - Switched to using Invoke-SqlCmd call instead of .NET methods

    - September 1, 2020
      - Moved to updated HSC modules
      - Changed to new Get-HSCSPOExclusionList function with these cmdlets:
        https://docs.microsoft.com/en-us/powershell/sharepoint/sharepoint-pnp/sharepoint-pnp-cmdlets?view=sharepoint-ps
#>

[CmdletBinding()]
param (
    [string]$sqlPasswordPath=$((Get-HSCEncryptedDirectoryPath) + "SOLESQL4.txt"),
    [int]$MaximumDisables = 20, #Safety measure to prevent too many accounts from being disabled
    [switch]$DisableAccounts, #Must use this switch to do account disables
  [string]$SQLServer = "sql01.hsc.wvu.edu",
  [string]$DBName = "SecurityAwareness",
  [string]$DBUsername = "itsnetworking",
  [string]$DBTableName = "DisabledUser"
)

#Configure environment and get exclusion list
$Error.Clear()

try {
  Write-Verbose "Configuring environment"

  Set-HSCEnvironment -ErrorAction Stop
  Connect-HSCExchangeOnlineV1 -ErrorAction Stop

  $DoNotDisable = Get-HSCSPOExclusionList

  Write-Output "`n`n---------------------------------------------------"
  Write-Output "Processing will not be done on these accounts:"
  $DoNotDisable
  Write-Output "---------------------------------------------------`n`n"
}
catch {
  Write-Warning "There was an error configuring the environment. Program is exiting."
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

try {
  #Decrypt SQL Password
  $sqlSecureStringPassword = Get-Content $sqlPasswordPath  | ConvertTo-SecureString
  $sqlPassword = [System.Runtime.InteropServices.marshal]::PtrToStringAuto([System.Runtime.InteropServices.marshal]::SecureStringToBSTR($sqlSecureStringPassword))

  #Get SQL Connection String
  $SQLConnectionString = Get-HSCConnectionString -Datasource $SQLServer -Database $DBName -Username $DBUsername -Password $sqlPassword

  #Query database to get users who should be disabled
  $query = "select * from $DBTableName"
  $SQLData = Invoke-SQLCmd -Query $query -ConnectionString $SQLConnectionString

  $Count = 0 #Keeps track of current user for visual output
  $DisableCount = 0 #Keeps track of new accounts being disabled.
}
catch {
  Write-Warning "There was an error reading the SOLE database. Program is exiting."
```

```powershell
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

$SQLRowCount = ($SQLData | Measure-Object).Count

while ($Count -lt $SQLRowCount)
{
  Write-Output "User number: $Count"
  Write-Output "Disable Count: $DisableCount"
  Write-Output $("Error Count: " + $Error.Count)

  $user = $SQLData[$Count].Username.Trim()
  Write-Output "User to be disabled: $user"

  if ($DisableCount -lt $MaximumDisables)
  {
        if ($DoNotDisable -notcontains $user)
        {
      #This if statement ensures that nobody in the exclusion list will be disabled.
      Write-Output "User is not in exclusion list"

      if($DisableAccounts)
      {
        try
        {
          $MbxExist = $false

          #This line just checks to see if the mailbox exists.
          #Get-Mailbox throws an error that is really just cosmetic if a mailbox
          #doesn't exist. This line will prevent that from being displayed. Instead, it will
          #just output on the screen that the mailbox doesn't exist.
          $MbxExist = [bool](Get-Mailbox $user -ErrorAction SilentlyContinue)

          if ($MbxExist)
          {
            $Mbx = Get-Mailbox $user

            $CasMailboxInfoFound = $true
            try
            {
              $CasMbx = $Mbx | Get-CasMailbox -ErrorAction Stop
              Write-Verbose "Successfully pulled CasMailbox information"
            }
            catch
            {
              #I'm not sure why certain accounts need this instead of the previous
              #code, but there are a small number that do.
              try
              {
                $CasMbx = Get-CasMailbox -Identity $Mbx.PrimarySMTPAddress -ErrorAction Stop
                Write-Verbose "Successfully pulled CasMailbox information with PrimarySMTPAddress"
              }
              catch
              {
                Write-Warning "Unable to pull CasMailbox information for this user"
                $CasMailboxInfoFound = $false
              }
            }

            if (!$CasMbx.OWAEnabled -AND !$CasMbx.MAPIEnabled -AND !$CasMbx.ActiveSyncEnabled -AND $CasMailboxInfoFound)
            {
              Write-Output "Mailbox is already disabled"
            }
            else
            {
              if (($Mbx | Measure-Object).Count -eq 1)
              {
                #This if statement just makes sure that one uniqe mailbox is found.
                $Mbx | Set-CasMailbox -OWAEnabled $false -MAPIEnabled $false -ActiveSyncEnabled $false

                Write-Output "User has been disabled: $user"

                $DisableCount++
              }
              else
              {
                Write-Warning "Unable to find one unique mailbox for this user."
                Write-Warning "User should be disabled, but isn't due to this issue."
              }
            }
          }
          else
          {
            Write-Output "No mailbox for this user: $user"
          }
        }
        catch
        {
          Write-Warning "An error happened trying to disable mailbox"
        }
      }
      else
      {
        Write-Warning "User should be disabled"
        Write-Warning $("DisableAccounts: $DisableAccounts")
      }
        }
    else
    {
      Write-Output "The user is in the exclusion list and will not be disabled."
    }

  }
    else
    {
    Write-Warning "The maximum number of disables has been reached."
    Write-Warning "No further accounts will be disabled."
    }

  $Count++

  Write-Output "******************************"
} #End main while loop

Write-Output "`nProcessing has been completed."
Write-Output "Accounts Processed: $Count"
Write-Output "New Accounts Disabled: $DisableCount"

Invoke-HSCExitCommand -ErrorCount $Error.Count
```