```powershell
# Enable-AccountSecurityCompliance.ps1
# Written by: Jeff Brusoe
# Last Updated: January 4, 2021
#
# This account is designed to enable an Office365 mailbox
# after a user has passed their security compliance (HIPAA or IT Security) training.

[CmdletBinding()]
param (
    [ValidateNotNullOrEmpty()]
    [string]$DBName = "SecurityAwareness",

    [ValidateNotNullOrEmpty()]
    [string]$DBTableName = "EnableExchange",

    [ValidateNotNullOrEmpty()]
    [string]$SaveLogDirectory = "$PSScriptRoot\LogsToSave\"
  )

try {
  Write-Output "Configuring HSC Environment"
  $SessionTranscriptFile = Set-HSCEnvironment -ErrorAction Stop

  Write-Output "Connecting to Exchange Online"
  Connect-HSCExchangeOnlineV1 -ErrorAction Stop

  $NewDisable = $false
}
catch {
  Write-Warning "Unable to configure environment. Program is exiting."
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

try {
  Write-Output "Decrypting SQL Password & generating connection string"

    $SQLPassword = Get-HSCSQLPassword -SOLEDB -Verbose -ErrorAction Stop

  $GetHSCSQLConnectionStringParams = @{
    Password = $SQLPassword
    Database = $DBName
    ErrorAction = "Stop"
  }

    $SQLConnectionString = Get-HSCSQLConnectionString @GetHSCSQLConnectionStringParams
}
catch {
  Write-Warning "Unable to decrypt SQL Passowrd"
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

try {
  #Query table for any users who have passed compliance training
  $Query = "select * from $DBTableName"

  $InvokeSQLCmdParams = @{
    Query = $Query
    ConnectionString = $SQLConnectionString
```

```
    ErrorAction = "Stop"
  }

  $AccountsToEnable = Invoke-SqlCmd @InvokeSQLCmdParams
}
catch {
  Write-Warning "Unable to query SQL DB"
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

$TotalCount = 1

if (($AccountsToEnable | Measure-Object).Count -gt 0)
{
  foreach ($AccountToEnable in $AccountsToEnable)
  {
    Write-Output "Total Count: $TotalCount"
    $TotalCount++
    $NewDisable = $false

    Write-Output "New Disable Count: $NewDisableCount"

    Write-Output $("Current email: " + $AccountToEnable.Email)
    Write-Output $("Backup Email: " + $AccountToEnable.BackupEmail)

    if (($AccountToEnable.Email -like "*mix.wvu.edu*") -AND
      ([string]::IsNullOrEmpty($AccountToEnable.BackupEmail)))
    {
      Write-Output "Primary email is @mix.wvu.edu"

      $DeleteQuery = "Delete From $DBTableName Where Email='" +
              $AccountToEnable.Email + "'"
      Write-Output "Delete Query: $DeleteQuery"

      $InvokeSQLCmdParams.Query = $DeleteQuery
      try {
        Invoke-SqlCmd @InvokeSQLCmdParams
      }
      catch {
        Write-Warning "Unable top delete from DB"
      }

    }
    else
    {
      try
      {
        $CasMailbox = Get-CASMailbox $AccountToEnable.Email -ErrorAction Stop

        if ($CasMailbox.OWAEnabled -AND
          $CasMailbox.MAPIEnabled -AND
          $CasMailbox.ActiveSyncEnabled)
        {
          Write-Output $("MapiEnabled: " + $CasMailbox.MapiEnabled)
          Write-Output $("ActiveSyncEnabled: " + $CasMailbox.ActiveSyncEnabled)
          Write-Output $("OWAEnabled: " + $CasMailbox.OWAEnabled)

          Write-Output "Mailbox is enabled."

          $DeleteQuery = "Delete From $DBTableName Where Email='" +
```

```
                $AccountToEnable.Email + "'"
          Write-Output "Delete Query: $DeleteQuery"

          $InvokeSQLCmdParams.Query = $DeleteQuery

          Invoke-SqlCmd @InvokeSQLCmdParams
        }
        else
        {
          #Newly enabled mailbox

          $SetCASMailBoxParams = @{
            Identity = $AccountToEnable.Email
            OWAEnabled = $true
            MAPIEnabled = $true
            ActiveSyncEnabled = $true
            ErrorAction = "Stop"
          }

          Set-CASMailbox @SetCASMailBoxParams

          $NewDisable = $true
        }
      }
      catch
      {
        if ([string]::IsNullOrEmpty($AccountToEnable.BackupEmail)) {
          Write-Output "Backup email is empty"
        }
        else
        {
          try {
            $SetCASMailBoxParams.Identity = $AccountToEnable.BackupEmail

            Set-CASMailbox @SetCASMailBoxParams
            Get-CASMailbox $AccountToEnable.BackupEmail -ErrorAction Stop
          }
          catch {
            Write-Warning "Unable to enable account"
          }
        }

        try {
          $DeleteQuery = "Delete From $DBTableName Where Email='" +
                  $AccountToEnable.Email + "'"
          Write-Output "Delete Query: $DeleteQuery"

          Write-Output "Running Delete Query"

          $InvokeSQLCmdParams.Query = $DeleteQuery
          Invoke-SqlCmd @InvokeSQLCmdParams
        }
        catch {
          Write-Warning "Unable to execute delete query"
        }
      }
    }

    Write-Output "**********************"
  }
```

```
}
else {
  Write-Output "No updates available at $(Get-Date)"
}

if ($NewDisable)
{
  #An account is newly disabled. Being moved to Logs to keep directory.
  try {
    Copy-Item -Path $SessionTranscriptFile -Destination $SaveLogDirectory
  }
  catch {
    Write-Warning "Error moving session transcript to logs to save directory"
  }
}

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