```powershell
#Update-SharedUserDB.ps1
#Written by: Jeff Brusoe
#Last Updated: October 13, 2020

#The purpose of this file is to update the shared user database based on information
#in the shared user file that is sent over daily.

[CmdletBinding()]
param (
    [ValidateNotNullOrEmpty()]
    [string]$SharedUserPath = "\\hs\public\tools\SharedUsersRpt\",

    [switch]$Testing
)

#Configure Environment
Set-HSCEnvironment

$NotProcessedUsers = "$PSScriptRoot\Logs\" + (Get-Date -Format yyyy-MM-dd-HH-mm) + "-NotProcessedUsers.csv"
Write-Verbose "Not Processed User File $NotProcessedUsers"

$Count = 0

#Get O365 SQL instance connection string
try {
    $SQLPassword = Get-HSCSQLPassword -Verbose

    $ConnectionString = Get-HSCSQLConnectionString -DataSource "hscpowershell.database.windows.net" -Database "HSCPowerShell" -Username "HSCPowerShell" -SQLPassword $SQLPassword
}
catch {
    Write-Warning "Unable to generate SQL connection string"
    Invoke-HSCExitCommand -ErrorCount $Error.Count
}

#Remove entries from shared user DB table
try {
    Write-Verbose "Deleting current shared user DB entries"

    $DeleteQuery = "Delete from SharedUserTable"
    Invoke-Sqlcmd -Query $DeleteQuery -ConnectionString $ConnectionString
}
catch {
    Write-Warning "Unable to remove old entries from DB table"
    Invoke-HSCExitCommand -ErrorCount $Error.Count
}

if (Test-Path $SharedUserPath)
{
    $SharedFile = Get-HSCLastFile -DirectoryPath $SharedUserPath
    Write-Output "SharedFile: $SharedFile"

    #Begin looping through shared user file
    $SharedUsers = Import-Excel $SharedFile

    #Copy shared user file
    Copy-Item -Path $SharedFile -Destination "$PSScriptRoot\SharedUserFiles\"
```

```
}
else {
    Write-Warning "Unable to reach shared user file path. Program is exisitng."
    Invoke-HSCExitCommand -ErrorCount $Error.Count
}

foreach ($SharedUser in $SharedUsers)
{
    #Search for blank fields in shared user file
    $BlankField = $false
    foreach ($SharedUserProperty in $SharedUser.PSObject.Properties)
    {
        Write-Verbose $("Checking Property " + $SharedUserProperty.Name)

        if ([string]::IsNullOrEmpty($SharedUserProperty.Value) -AND
$SharedUserProperty.Name -ne "mid")
        {
            $BlankField = $true
            Write-Verbose "Field is blank"
            break
        }
        else {
            Write-Verbose "Field is not blank"
        }
    }

    if ($SharedUser.Status -eq "Accepted")
    {
         Write-Output $("Processing: " + $SharedUser.wvu_id)

         if ($BlankField)
         {
             Write-Output "User has a blank field and will not processed"
             $SharedUser | Export-Csv $NotProcessedUsers -Append
         }
         else {
             Write-Output "User will be added to database"

             #Determine if they are HSC or WVUM users
             $HSCEmail = $false

             if ($SharedUser.email.ToLower().indexOf("hsc.wvu.edu") -gt 0)
             {
                 Write-Verbose "User has an HSC email address"
                 $HSCEmail = $true
             }
             elseif ($SharedUser.cost_center_name.indexOf("HVI") -ge 0)
             {
                 Write-Verbose "User is in HVI"
                 $HSCEmail = $true
             }
             else {
                 Write-Verbose "Hospital User"
             }

             $LastName = $SharedUser.lastname
             Write-Output "Last Name: $LastName"

             if ($LastName.indexOf("'") -gt 0)
             {
```

```
116                 $LastName = $LastName -Replace "'",""
117             }
118
119             $FirstName = $SharedUser.firstname
120             if ($FirstName.indexOf("'") -gt 0)
121             {
122                 $FirstName = $FirstName -Replace "'",""
123             }
124
125             Write-Output "Count: $Count"
126             $Count++
127
128             $InsertQuery = "Insert into SharedUserTable
(Status,wvu_id,uid,firstname,lastname,email,HSCEmail) " +
129             "Values
('$($SharedUser.Status)','$($SharedUser.wvu_id)','$($SharedUser.uid)'," +
130             "'$FirstName','$LastName','$($SharedUser.email)','$HSCEmail');"
131
132             Write-Output "Insert Query:"
133             Write-Output $InsertQuery
134
135             try {
136                 Write-Output "Attempting to write to DB"
137                 Invoke-SQLCmd -Query $InsertQuery -ConnectionString
$ConnectionString -ErrorAction Stop
138                 Write-Output "Successfuly wrote user to DB"
139             }
140             catch {
141                 #Should get here if user exists in DB already. Insert query errors
out and must run update query.
142                 $Error.Clear()
143
144                 Write-Output "Error running insert query. Trying SQL update query"
145                 $UpdateQuery = "Update SharedUserTable " +
146                                "SET uid =
'$($SharedUser.uid)',firstname='$FirstName'," +
147
 "lastname='$LastName',email='$($SharedUser.email)',HSCEmail='$HSCEmail' " +
148                                "where wvu_id = '$($SharedUser.wvu_id)'"
149
150                 Write-Output "Update Query:"
151                 Write-Output $UpdateQuery
152
153                 try {
154                     Invoke-SQLCmd -Query $UpdateQuery -ConnectionString
$ConnectionString -ErrorAction Stop
155                     Write-Output "Successfully ran update query"
156                 }
157                 catch {
158                     Write-Warning "Error running update query."
159
160                     if ($Testing) {
161                         Write-Warning "Program is exiting."
162                         Invoke-HSCExitCommand -ErrorCount $Error.Count
163                     }
164
165                 }
166             }
167         }
168     }
```

```
169
170     Write-Output "*****************************"
171 }
172
173 Invoke-HSCExitCommand -ErrorCount $Error.Count
```