```powershell
<#
  .SYNOPSIS
        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.

    .PARAMETER SharedUserPath
        This parameter is the path to where the shared user file is stored.

    .PARAMETER Testing
        Used to give debug output for code development.

  .NOTES
    Last Modified by: Jeff Brusoe
    Last Modified: October 27, 2020
#>

[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

    $ConnectionStringParams = @{
        DataSource = "hscpowershell.database.windows.net"
        Database = "HSCPowerShell"
        UserName = "HSCPowerShell"
        SQLPassword = $SQLPassword
    }
    $ConnectionString = Get-HSCSQLConnectionString @ConnectionStringParams
}
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 -ErrorAction Stop
}
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

    if ([string]::IsNullOrEmpty($SharedUser.email) -AND $SharedUser.Status -eq
"Accepted")
    {
        Write-Output "Shared user is accepted but with blank email field"

        Write-Output "User has a blank field and will not processed"
        $SharedUser | Export-Csv $NotProcessedUsers -Append

        continue
    }
    else
    {
        foreach ($SharedUserProperty in $SharedUser.PSObject.Properties)
        {
            Write-Verbose $("Checking Property " + $SharedUserProperty.Name)

            if ([string]::IsNullOrEmpty($SharedUserProperty.Value) -AND
$SharedUserProperty.Name -ne "mid")
            {
                Write-Output "Field is blank"
                Write-Output "User will not processed"

                $SharedUser | Export-Csv $NotProcessedUsers -Append

                continue
            }
            else {
                Write-Verbose "Field is not blank"
            }
        }
    }

    if ($SharedUser.Status -eq "Accepted")
    {
        Write-Output $("Processing: " + $SharedUser.wvu_id)
        Write-Output "User will be added to database"
```

```powershell

        #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)
        {
        $LastName = $LastName -Replace "'",""
        }

        $FirstName = $SharedUser.firstname
        if ($FirstName.indexOf("'") -gt 0)
        {
            $FirstName = $FirstName -Replace "'",""
        }

        Write-Output "Count: $Count"
        $Count++

        $InsertQuery = "Insert into SharedUserTable (Status,wvu_id,uid,firstname,lastname,email,HSCEmail) " +
        "Values ('$($SharedUser.Status)','$($SharedUser.wvu_id)','$($SharedUser.uid)'," +
        "'$FirstName','$LastName','$($SharedUser.email)','$HSCEmail');"

        Write-Output "Insert Query:"
        Write-Output $InsertQuery

        try {
            Write-Output "Attempting to write to DB"
            Invoke-SQLCmd -Query $InsertQuery -ConnectionString $ConnectionString -ErrorAction Stop
            Write-Output "Successfuly wrote user to DB"
        }
        catch {
            #The update query probably isn't needed. I'm keeping it here because it likely will be needed
            #in the future when the delete query is removed.
            #Should get here if user exists in DB already. Insert query errors out and must run update query.
            $Error.Clear()

            Write-Output "Error running insert query. Trying SQL update query"
            $UpdateQuery = "Update SharedUserTable " +
                            "SET uid = '$($SharedUser.uid)',firstname='$FirstName'," +
                            "lastname='$LastName',email='$($SharedUser.email)',HSCEmail='$HSCEmail' " +
                            "where wvu_id = '$($SharedUser.wvu_id)'"

            Write-Output "Update Query:"
            Write-Output $UpdateQuery

            try {
                Invoke-SQLCmd -Query $UpdateQuery -ConnectionString $ConnectionString -ErrorAction Stop
                Write-Output "Successfully ran update query"
            }
            catch
            {
                Write-Warning "Error running update query."

                if ($Testing) {
                    Write-Warning "Program is exiting."
                    Invoke-HSCExitCommand -ErrorCount $Error.Count
                }

            }
        }
    }

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

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