```
# Backup-ADGroupMembershipByGroupName.ps1
# Written by: Jeff Brusoe
# Last Updated: September 9, 2021
#
# The purpose of this file is to backup AD Group Membership.
# Backups are written to the O365 SQL Instance and are based on
# the group's name.

[CmdletBinding()]
param(
    [ValidateNotNullOrEmpty()]
    [string]$LogDirectory = "$PSScriptRoot\Logs\",

    [ValidateNotNullOrEmpty()]
    [string]$SQLServer = "hscpowershell.database.windows.net",

    [ValidateNotNullOrEmpty()]
  [string]$DBName = "HSCPowerShell",

    [ValidateNotNullOrEmpty()]
  [string]$DBUsername = "HSCPowerShell",

    [ValidateNotNullOrEmpty()]
  [string]$DBTableName = "ADGroupMembershipByGroupName"
)

try {
    Set-HSCEnvironment -ErrorAction Stop

    $EmptyGroupLog = "$LogDirectory\" +
                        (Get-Date -Format yyyy-MM-dd-HH-mm) +
                        "-EmptyGroups.txt"
    New-Item -ItemType File -Force -Path $EmptyGroupLog

    try {
        Write-Output "Generating SQL Connection String"
        $SQLPassword = Get-HSCSQLPassword -Verbose -ErrorAction Stop

        $GetHSCConnectionStringParams = @{
            DataSource = $SQLServer
            Database = $DBName
            Username = $DBUsername
            SQLPassword = $SQLPassword
            ErrorAction = "Stop"
        }

        $SQLConnectionString = Get-HSCSQLConnectionString
@GetHSCConnectionStringParams

        $InvokeSQLCmdParams = @{
            ConnectionString = $SQLConnectionString
            ErrorAction = "Stop"
        }

        $DeleteQuery = "DELETE FROM $DBTableName"
        $InvokeSQLCmdParams["Query"] = $DeleteQuery
        Invoke-SQLCmd @InvokeSQLCmdParams
    }
    catch {
        Write-Warning "Unable to generate SQL connection string"
```

```powershell
        Invoke-HSCExitCommand -ErrorCount $Error.Count
    }
}
catch {
    Write-Warning "Unable to configure environment"
    Invoke-HSCExitCommand -ErrorCount $Error.Count
}

try {
    Write-Output "Getting list of AD Groups"

    $ADGroups = Get-ADGroup -Filter * -ErrorAction Stop

    Write-Output $("Total number of AD Groups: " + $ADGroups.Count)
}
catch {
    Write-Warning "Unable to get list of AD groups"
    Invoke-HSCExitCommand -ErrorCount $Error.Count
}

Write-Output "Going through list of AD Groups"

foreach ($ADGroup in $ADGroups)
{
    Write-Output $("Group Name: " + $ADGroup.Name)
    Write-Output $("Distinguished Name: " + $ADGroup.DistinguishedName)

    try {
        $ADUsers = $ADGroup |
            Get-ADGroupMember -ErrorAction Stop -Recursive |
            Where-Object {$_.objectClass -eq "user"}

        if ($ADUsers.Count -eq 0) {
            Write-Output $("No users in group: " + $ADGroup.Name)

            Add-Content -Path $EmptyGroupLog -Value $ADGroup.Name -ErrorAction Stop
        }
        else {
            $SamAccountNames = $ADUsers.SamAccountName

            if ($SamAccountNames.Count -eq 0) {
                Write-Output "Group Member has no SamAccountName"
                continue
            }
            else {
                $GroupMembers = $SamAccountNames -join ";"
            }

            $GroupName = $ADGroup.Name
            $GroupDN = $ADGroup.DistinguishedName

            $InsertQuery = "Insert into $DBTableName Values ('" +
                                    "$GroupName','" +
                                    "$GroupDN','" +
                                    "$GroupMembers" + "')"
            Write-Output "SQL Insert Query: $InsertQuery"

            $InvokeSQLCmdParams["Query"] = $InsertQuery
            Invoke-SQLCmd @InvokeSQLCmdParams
        }
    }
    catch {
        Write-Warning "Unable to get list of AD group members"
    }

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

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