Open your PowerShell console and execute the below statements one by one.

**Exercise 1: List All Running Processes**

Question: Write a cmdlet that lists all currently running processes on your system.

**Get-Process**

**Exercise 2: Get System Information**

Question: Retrieve detailed information about your system, including the operating system version and architecture.

**Get-ComputerInfo**

**Exercise 3: Get your PowerShell version**

**PSVersionTable**

**Exercise 4: Get Current date & time**

**Get-Date**

**get-date | select Year**

**Exercise 5: Gets one or more disks visible to the operating system**

**Get-Disk**

**Exercise 6: Returns a list of all partition objects visible on all disks**

**Get-Partition**

**Exercise 7: Get Your System Information**

**Get-ComputerInfo**

**Get-ComputerInfo | Select CsManufacturer, CsModel, CsName**

**Exercise 8: Get OS Details of your system**

**Get-ComputerInfo**

**Get-ComputerInfo | select OsName, OsType, OsVersion, OsArchitecture**

**Get-ComputerInfo | select OsName, OsType, OsVersion, OsArchitecture | Format-List**

**Get-ComputerInfo | select OsName, OsType, OsVersion, OsArchitecture | Format-Table**

**Exercise 9: Get disk information**

**get-disk**

**Exercise 10: Retrieve detailed network configuration**

**Get-NetIPAddress**

**File manipulation cmdlet activity**

**Exercise 1: Create a New File**

Question: Create a new text file named demo.txt in the directory D:\Day1, and write the text "**This is a sample file content"** into it.

Answer:

New-Item -Path **"D:\Day1\demo.txt"** -ItemType File

Set-Content -Path **"D:\Day1\demo.txt"** -Value "This is a sample file content"

**Exercise 2: Read Content from a File**

Question: Read the content of the file **demo.txt** located at **D:\Day1**.

Answer:

Get-Content -Path "D:\Day1\demo.txt"

**Exercise 3: Append Text to a File**

Question: Append the text "Adding more data to the file" to the D:\Day1\demo.txt file.

Add-Content -Path "D:\Day1\demo.txt" -Value "Adding more data to the file"

**Exercise 4: Copy a File**

Question: Copy demo.txt from **D:\Day1** to **D:\Day1\_Backup**.

Answer:

Copy-Item -Path "D:\Day1\demo1.txt" -Destination "D:\Day1\_Backup\demo1.txt"

**Exercise 5: Move a File**

Question: Move the demo.txt file from D:\Day1 to E:\

Answer:

Move-Item -Path "D:\Day1\demo.txt" -Destination "E:\"

**Exercise 6: Rename a File**

Question: Rename the file demo.txt to new\_demo.txt in the E:\.

Answer:

Rename-Item -Path "E:\demo.txt" -NewName "new\_demo.txt"

**Exercise 7: Delete a File**

Question: Delete the file new\_demo.txt located at E:\.

Answer:

Remove-Item -Path "E:\new\_demo.txt”

**Process manipulation cmdlet activity**

**Exercise 1: Get a List of All Running Processes**

**Question: Get a list of all currently running processes on your machine.**

Answer:

**Get-Process**

**Exercise 2: Get Information for a Specific Process**

**Question: Get information about the process notepad (or any other process you know is running on your system).**

Answer:

**Get-Process -Name "notepad"**

**Exercise 3: Get Process by Process ID (PID)**

**Question: Get information about the process with a specific Process ID (PID), for example, PID 1234.**

Answer:

**Get-Process -Id 1234**

**Exercise 4: Sort Processes by Memory Usage**

**Question: Get a list of processes and sort them by memory usage in descending order.**

Answer:

**Get-Process | Sort-Object -Property WorkingSet -Descending**

**Exercise 5: Display Process Details (Including CPU and Memory)**

**Question: Get a detailed list of processes, including CPU usage and memory usage (WorkingSet) for each process.**

**Answer:**

**Get-Process | Select-Object Name, Id, CPU, WorkingSet**

**Exercise 6: Stop a Running Process**

**Question: Stop the process with the name notepad (or another process running on your system).**

Answer:

**Stop-Process -Name "notepad"**

**Exercise 7: Stop a Process-by-Process ID (PID)**

**Question: Stop the process with PID 1234.**

Answer:

**Stop-Process -Id 1234**

**Exercise 8: Start a New Process**

**Question: Start a new instance of notepad (or any other program you want to launch).**

Answer:

**Start-Process "notepad"**

**Exercise 9: Start a Process with Arguments**

**Question: Start the notepad process and open a specific file (D:\Day1\demo.txt).**

Answer:

**Start-Process "notepad" -ArgumentList "D:Day1\demo.txt"**

**Exercise 10: Get Process CPU Time**

**Question: Get the total CPU time used by the process chrome.**

Answer:

**Get-Process -Name "chrome" | Select-Object Name, CPU**

**Disk cmdlet activity**

**Exercise 1: Get Information About All Disks**

**Question: Get a list of all physical disks on your system, including their size, status, and model.**

Answer:

**Get-Disk**

**Exercise 2: Get Information About a Specific Disk**

**Question: Get detailed information about a disk with DiskNumber 1.**

Answer:

**Get-Disk -Number 1**

**Exercise 3: Check Disk Health (Operational Status)**

**Question: Check the operational status of all disks on your system to see if they are online or offline.**

Answer:

**Get-Disk | Select-Object Number, FriendlyName, OperationalStatus**

**Exercise 4: Export Disk Information to a File**

**Question: Export the information about all disks and volumes on the system to a text file named DiskInfo.txt.**

**Answer:**

Get-Disk | Out-File -FilePath "D:\Day1\DiskInfo.txt"

Get-Volume | Out-File -FilePath “D:\Day1\DiskInfo.txt" -Append

**PowerShell script – Activity**

Exercise 8: Check if a File Exists

Question: Check if the file sample.txt exists in the C:\Temp directory. If it does, output "File exists."

Answer:

if (Test-Path -Path "C:\Temp\sample.txt") {

Write-Output "File exists"

}

else {

Write-Output "File does not exist"

}

Exercise 9: Get a List of Files in a Directory

Question: List all the files in the C:\Temp directory.

Answer:

Get-ChildItem -Path "C:\Temp"

Exercise 10: Delete All Files in a Directory

Question: Delete all .txt files in the C:\Temp directory.

Answer:

Get-ChildItem -Path "C:\Temp" -Filter "\*.txt" | Remove-Item

Note: delete operation

---------------------------------------------------

Exercise 11: Check File Size

Question: Check the size of the file sample.txt located at C:\Temp.

Answer:

(Get-Item -Path "C:\Temp\sample.txt").Length

Exercise 12: Read the First 5 Lines of a File

Question: Read the first 5 lines of the sample.txt file located at C:\Temp.

Answer:

Get-Content -Path "C:\Temp\sample.txt" | Select-Object -First 5

Exercise 13: Check if a File is Empty

Question: Check if the file sample.txt is empty or not.

Answer:

if ((Get-Content -Path "C:\Temp\sample.txt").Length -eq 0) {

Write-Output "The file is empty."

} else {

Write-Output "The file is not empty."

}

Exercise 14: Monitor a Process's CPU and Memory Usage Over Time

Question: Monitor the CPU and memory usage of the notepad process every 5 seconds for 30 seconds.

Answer:

$process = Get-Process -Name "notepad"

while ($true) {

$process | Select-Object Name, CPU, WorkingSet

Start-Sleep -Seconds 5

$process = Get-Process -Name "notepad"

if ($process) {

break

}

}

Exercise 15: Get Processes Consuming High CPU

Question: Get a list of processes consuming more than 50% CPU.

Answer:

**Get-Process | Where-Object { $\_.CPU -gt 50 }**

Exercise 16: Get All Processes Started by a Specific User

Question: Get all processes running on your system that were started by the user Administrator.

**Answer:**

**Get-Process | Where-Object { $\_.StartInfo.UserName -eq "Administrator" }**

Exercise 17: Get Process Handles

Question: Get the handle count (number of handles) for the notepad process.

**Answer:**

**(Get-Process -Name "notepad").Handles**

Exercise 18: Get the Start Time of a Process

Question: Get the start time of the notepad process.

Answer:

**(Get-Process -Name "notepad").StartTime**

Exercise 19: Find Processes with a Specific Module (DLL)

Question: Find all processes that have loaded the kernel32.dll module.

Answer:

**Get-Process | Where-Object { $\_.Modules.ModuleName -contains "kernel32.dll" }**

Exercise 20: List Process Owner Information

Question: Get the owner (user) of the notepad process.

Answer:

Get-WmiObject -Class Win32\_Process -Filter "Name = 'notepad.exe'" | Select-Object ProcessId, @{Name="Owner";Expression={(Get-WmiObject -Class Win32\_ComputerSystem).UserName}}

Exercise 21: Get a List of Running Processes with a Specific Name Pattern

Question: Get a list of processes whose names start with chrome.

Answer:

**Get-Process | Where-Object { $\_.Name -like "chrome\*" }**

----------------------------------------------------------------------

Note: Don't try the below cmdlets

Exercise 6: Format a Partition

Question: Format the partition on DiskNumber 1 that was just created with the file system NTFS.

Answer:

Format-Volume -DriveLetter D -FileSystem NTFS -Confirm:$false

(Note: Replace D with the correct drive letter assigned by the New-Partition cmdlet.)

Exercise 7: Create a Simple Volume

Question: Create a simple volume on DiskNumber 1 with a size of 100GB.

Answer:

New-Volume -DiskNumber 1 -Size 100GB -FileSystem NTFS -DriveLetter E

Exercise 8: Check Free Space on a Drive

Question: Check the free space and used space on drive C:.

Answer:

Get-PSDrive -Name C

Exercise 9: Resize a Partition

Question: Resize the partition on DiskNumber 1 to 200GB.

Answer:

Resize-Partition -DriveLetter D -Size 200GB

(Note: Ensure that you have the necessary space available on the disk before resizing.)

Exercise 10: Set a Drive Label

Question: Set the label of drive D: to "BackupDrive".

Answer:

Set-Volume -DriveLetter D -NewFileSystemLabel "BackupDrive"

Exercise 11: List All Mounted Volumes

Question: List all the volumes (partitions) that are currently mounted on your system.

Answer:

Get-Volume

Exercise 12: Set a Drive to Offline Status

Question: Set DiskNumber 2 to offline status.

Answer:

**Set-Disk -Number 2 -IsOffline $true**

Exercise 13: Set a Disk to Online Status

Question: Set DiskNumber 2 to online status.

Answer:

**Set-Disk -Number 2 -IsOffline $false**

Exercise 14: Get Detailed Disk Information Using WMI

Question: Get detailed disk information (including the serial number, model, etc.) for DiskNumber 1 using WMI.

Answer:

Get-WmiObject -Class Win32\_DiskDrive -Filter "DeviceID='\\\\.\\PHYSICALDRIVE1'" | Select-Object Model, SerialNumber, Size, MediaType

Exercise 15: Clean a Disk (Remove All Partitions)

Question: Clean DiskNumber 1 (removes all partitions and data) and prepare it for a fresh setup.

Answer:

Clear-Disk -Number 1 -RemoveData -Confirm:$false

-------------------------------------------------------------------------------

Exercise 16: Monitor Disk Space Usage Continuously

Question: Monitor the free space on C: drive and display updates every 5 seconds.

Answer:

while ($true) {

Get-PSDrive -Name C | Select-Object Name, @{Name="Used(GB)";Expression={[math]::round($\_.Used/1GB,2)}}, @{Name="Free(GB)";Expression={[math]::round($\_.Free/1GB,2)}}

Start-Sleep -Seconds 5

}

Exercise 17: Check Disk SMART Status

Question: Check the SMART status (Self-Monitoring, Analysis, and Reporting Technology) for all disks on your system.

Answer:

Get-PhysicalDisk | Select-Object FriendlyName, OperationalStatus, HealthStatus

Exercise 18: Create a New Local User

Question: Write a script that creates a new local user account named NewUser with the password P@ssw0rd!. Ensure the account is not set to expire and is enabled.

Answer:

# Define user details

$username = "NewUser"

$password = ConvertTo-SecureString "P@ssw0rd!" -AsPlainText -Force

$fullName = "New User"

$description = "New User Account"

$userPrincipalName = "$username@domain.com"

# Create the new user

New-LocalUser -Name $username -Password $password -FullName $fullName -Description $description -Enabled $true

Write-Output "User account '$username' has been created successfully."

Exercise 19: Modify User Attributes

Question: Write a script that modifies the FullName and Description for an existing user ExistingUser in Active Directory.

Answer:

# Modify user attributes

$username = "ExistingUser"

$fullName = "Updated Full Name"

$description = "Updated Description"

# Modify the user in Active Directory

Set-ADUser -Identity $username -FullName $fullName -Description $description

Write-Output "User '$username' attributes have been updated."

(Note: This requires the ActiveDirectory module.)

--------------

Exercise 20: Disable a User Account

Question: Write a script that disables the user account UserToDisable in Active Directory.

Answer:

# Disable the user account

$username = "UserToDisable"

Disable-ADAccount -Identity $username

Write-Output "User '$username' account has been disabled."

-----------------

Exercise 21: Add User to a Group

Question: Write a script that adds the user UserToAdd to the group GroupName in Active Directory.

Answer:

# Add user to group

$username = "UserToAdd"

$groupName = "GroupName"

Add-ADGroupMember -Identity $groupName -Members $username

Write-Output "User '$username' has been added to the group '$groupName'."

----------------------------

Exercise 22: Remove User from a Group

Question: Write a script that removes the user UserToRemove from the group GroupName in Active Directory.

Answer:

# Remove user from group

$username = "UserToRemove"

$groupName = "GroupName"

Remove-ADGroupMember -Identity $groupName -Members $username -Confirm:$false

Write-Output "User '$username' has been removed from the group '$groupName'."

Exercise 23: List All Active Directory Users

Question: Write a script that lists all active users in Active Directory and displays their Name, SamAccountName, and Enabled status.

Answer:

# Get all active users

Get-ADUser -Filter {Enabled -eq $true} -Properties SamAccountName, Enabled | Select-Object Name, SamAccountName, Enabled

Exercise 24: Delete a Local User Account

Question: Write a script that deletes the local user account UserToDelete.

Answer:

# Delete the user account

$username = "UserToDelete"

Remove-LocalUser -Name $username

Write-Output "User '$username' has been deleted."

---------------------------

Exercise 25: Export Active Directory Users to CSV

Question: Write a script that exports the Name, SamAccountName, and Enabled status of all users in Active Directory to a CSV file.

Answer:

# Export AD user information to CSV

Get-ADUser -Filter \* -Properties SamAccountName, Enabled | Select-Object Name, SamAccountName, Enabled |

Export-Csv -Path "C:\ADUsers.csv" -NoTypeInformation

Write-Output "User information has been exported to C:\ADUsers.csv"

---------------------------

Exercise 26: Reset a User Password

Question: Write a script that resets the password for the user UserToReset to NewP@ssw0rd! in Active Directory.

Answer:

# Define the new password

$username = "UserToReset"

$newPassword = ConvertTo-SecureString "NewP@ssw0rd!" -AsPlainText -Force

# Reset the user's password

Set-ADAccountPassword -Identity $username -NewPassword $newPassword -Reset

Write-Output "Password for user '$username' has been reset."

------------

Exercise 27: Find Expired User Accounts

Question: Write a script that finds all user accounts in Active Directory that have expired (i.e., their AccountExpirationDate is in the past).

Answer:

# Find expired user accounts

Get-ADUser -Filter {AccountExpirationDate -lt (Get-Date)} -Properties AccountExpirationDate | Select-Object Name, AccountExpirationDate

------------

Exercise 28 : Create a User in Active Directory with Multiple Attributes

Question: Write a script that creates a user JohnDoe in Active Directory with the following attributes: GivenName = John, Surname = Doe, SamAccountName = jdoe, and Department = IT.

Answer:

# Define user attributes

$username = "JohnDoe"

$givenName = "John"

$surname = "Doe"

$samAccountName = "jdoe"

$department = "IT"

$password = ConvertTo-SecureString "P@ssw0rd!" -AsPlainText -Force

# Create the user

New-ADUser -GivenName $givenName -Surname $surname -SamAccountName $samAccountName -UserPrincipalName "$samAccountName@domain.com" `

-Department $department -Name "$givenName $surname" -AccountPassword $password -Enabled $true

Write-Output "User '$username' has been created successfully in Active Directory."

---------

Exercise 29: Export Group Membership to CSV

Question: Write a script that exports all members of a specific group GroupName in Active Directory to a CSV file.

Answer:

# Export group membership to CSV

Get-ADGroupMember -Identity "GroupName" | Select-Object Name, SamAccountName |

Export-Csv -Path "C:\GroupMembership.csv" -NoTypeInformation

Write-Output "Group membership has been exported to C:\GroupMembership.csv"

-----------------

Exercise 30: Check Disk Space Usage

Question: Check the free and used space on your system's drives (e.g., C: drive).

Answer:

Get-PSDrive -Name C

--------------------

Exercise 31: List All Installed Software

Question: List all installed software on your system (from the registry) with their name and version.

Answer:

Get-WmiObject -Class Win32\_Product | Select-Object Name, Version

Exercise 32: Create a New Folder

Question: Create a new folder called NewFolder on your desktop.

Answer:

New-Item -Path "C:\Users\$env:UserName\Desktop" -Name "NewFolder" -ItemType Directory

-------------------------------------

Exercise 33: Copy Files to a Folder

Question: Copy all .txt files from C:\Source to C:\Backup.

Answer:

Copy-Item -Path "C:\Source\\*.txt" -Destination "C:\Backup" -Recurse

-------------------------------------

Exercise 34: Rename a File

Question: Rename the file OldName.txt to NewName.txt in the C:\Documents folder.

Answer:

Rename-Item -Path "C:\Documents\OldName.txt" -NewName "NewName.txt"

---------------------------

Exercise 35: Start a Service

Question: Start the Windows Update service on your system.

Answer:

Start-Service -Name wuauserv

Exercise 36: Stop a Service

Question: Stop the Spooler (Print Spooler) service on your system.

Answer:

Stop-Service -Name Spooler

--------------------------------

Exercise 37: Get CPU Usage Statistics

Question: Get the current CPU usage statistics.

Answer:

Get-Counter -Counter "\Processor(\_Total)\% Processor Time"

Exercise 38: Find Files Larger Than 1 GB

Question: Find all files larger than 1 GB in the C:\Documents folder.

Answer:

Get-ChildItem -Path "C:\Documents" -Recurse | Where-Object { $\_.Length -gt 1GB }

---------------

Exercise 39: Display Last Boot Time

Question: Display the last boot time of your system.

Answer:

(Get-CimInstance -ClassName Win32\_OperatingSystem).LastBootUpTime

-------------------

Exercise 40: Monitor a Folder for Changes

Question: Monitor the folder C:\Documents for any changes (such as file creation, modification, or deletion).

Answer:

$folderPath = "C:\Documents"

$filter = "\*.\*"

$fsw = New-Object IO.FileSystemWatcher $folderPath, $filter

$fsw.EnableRaisingEvents = $true

Register-ObjectEvent $fsw "Changed" -Action {

Write-Host "Change detected in $($EventArgs.FullPath) at $($EventArgs.ChangeType)"

}

Write-Host "Monitoring changes in $folderPath. Press any key to exit."

[System.Console]::ReadKey($true) | Out-Null

Exercise 41:

Question: Write a script that copies all files from C:\Source to C:\Backup and adds a timestamp to the backup folder name. If the C:\Backup folder does not exist, create it.

Answer:

$source = "C:\Source"

$backupDir = "C:\Backup"

$timestamp = Get-Date -Format "yyyyMMdd\_HHmmss"

$backupPath = "$backupDir\Backup\_$timestamp"

# Check if the backup directory exists, if not, create it

if (-not (Test-Path -Path $backupDir)) {

New-Item -Path $backupDir -ItemType Directory

}

# Create a new backup folder with a timestamp

New-Item -Path $backupPath -ItemType Directory

# Copy all files from Source to Backup

Copy-Item -Path "$source\\*" -Destination $backupPath -Recurse

Write-Output "Backup completed successfully to $backupPath"

Exercise 42:

Question: Write a script that checks if the free space on the C: drive is less than 10 GB. If it is, send an alert message to the user.

Answer:

$drive = Get-PSDrive -Name C

$freeSpace = $drive.Free / 1GB

# Check if free space is less than 10GB

if ($freeSpace -lt 10) {

Write-Warning "Warning: C: drive has less than 10GB of free space. Free space: $([math]::round($freeSpace, 2)) GB"

}

else {

Write-Output "C: drive has sufficient free space: $([math]::round($freeSpace, 2)) GB"

}

Exercise 42:

Question: Write a script that checks whether a user account (e.g., username) is locked or disabled. If the account is disabled or locked, display a message indicating the status.

Answer:

$username = "username"

$user = Get-ADUser -Identity $username -Properties LockedOut, Enabled

if ($user.Enabled -eq $false) {

Write-Output "User account '$username' is disabled."

}

if ($user.LockedOut -eq $true) {

Write-Output "User account '$username' is locked."

}

if ($user.Enabled -eq $true -and $user.LockedOut -eq $false) {

Write-Output "User account '$username' is active and unlocked."

}

(Note: This script requires the ActiveDirectory module to run, which is available on domain-joined machines or with Remote Server Administration Tools (RSAT) installed.)

Exercise 43: Create Multiple Directories

Question: Write a script that creates a set of directories based on a list of folder names: ["Reports", "Logs", "Backups", "Archives"]. The script should create these directories in C:\Company.

Answer:

$folders = @("Reports", "Logs", "Backups", "Archives")

$basePath = "C:\Company"

# Loop through the folder names and create the directories

foreach ($folder in $folders) {

$folderPath = Join-Path -Path $basePath -ChildPath $folder

if (-not (Test-Path -Path $folderPath)) {

New-Item -Path $folderPath -ItemType Directory

Write-Output "Created directory: $folderPath"

}

else {

Write-Output "Directory already exists: $folderPath"

}

}

Exercise 43: Rename Files in a Directory

Question: Write a script that renames all .txt files in the C:\Documents directory by adding the current date at the beginning of each filename.

Answer:

$directory = "C:\Documents"

$datePrefix = Get-Date -Format "yyyyMMdd"

$files = Get-ChildItem -Path $directory -Filter "\*.txt"

# Loop through all text files and rename them

foreach ($file in $files) {

$newName = "$datePrefix" + "\_" + $file.Name

$newPath = Join-Path -Path $directory -ChildPath $newName

Rename-Item -Path $file.FullName -NewName $newPath

Write-Output "Renamed '$($file.Name)' to '$newName'"

}

Exercise 44: Check for Running Processes

Question: Write a script that checks if a specific process (e.g., notepad.exe) is running. If it is, display a message saying that the process is running. If it's not running, display a message saying it's not running.

Answer:

$processName = "notepad"

# Check if the process is running

$process = Get-Process -Name $processName -ErrorAction SilentlyContinue

if ($process) {

Write-Output "$processName is running."

}

else {

Write-Output "$processName is not running."

}

Exercise 45: Export User Information to CSV

Question: Write a script that exports a list of all users in Active Directory (with Name, SamAccountName, and Enabled properties) to a CSV file.

Answer:

Get-ADUser -Filter \* -Properties SamAccountName, Enabled | Select-Object Name, SamAccountName, Enabled |

Export-Csv -Path "C:\UsersList.csv" -NoTypeInformation

Write-Output "User information has been exported to C:\UsersList.csv"

(Note: This script requires the ActiveDirectory module.)

Exercise 46: Scheduled Task to Run a Script

Question: Write a script that creates a scheduled task to run a PowerShell script (C:\Scripts\Backup.ps1) every day at 8:00 AM.

Answer:

$action = New-ScheduledTaskAction -Execute "PowerShell.exe" -Argument "-File C:\Scripts\Backup.ps1"

$trigger = New-ScheduledTaskTrigger -Daily -At "08:00AM"

$taskName = "DailyBackup"

Register-ScheduledTask -Action $action -Trigger $trigger -TaskName $taskName -User "SYSTEM" -Force

Write-Output "Scheduled task '$taskName' has been created to run daily at 8:00 AM."

Exercise 47: Send an Email Notification

Question: Write a script that sends an email notification when a file (C:\Logs\logfile.txt) is modified. Use SMTP server smtp.example.com with the sender sender@example.com and recipient recipient@example.com.

Answer:

$logFile = "C:\Logs\logfile.txt"

$smtpServer = "smtp.example.com"

$sender = "sender@example.com"

$recipient = "recipient@example.com"

$subject = "File Modified"

$body = "The file '$logFile' was modified at $(Get-Date)."

$lastModified = (Get-Item $logFile).LastWriteTime

$lastChecked = $lastModified

while ($true) {

$currentModified = (Get-Item $logFile).LastWriteTime

if ($currentModified -ne $lastChecked) {

# Send email notification

Send-MailMessage -From $sender -To $recipient -Subject $subject -Body $body -SmtpServer $smtpServer

Write-Output "Email sent for modification at $(Get-Date)."

# Update the last modified timestamp

$lastChecked = $currentModified

}

Start-Sleep -Seconds 10

}

**Exercise 48: Monitor and Log System CPU Usage**

**Question: Write a script that monitors the CPU usage every 5 seconds and logs it to C:\Logs\cpu\_log.txt.**

Answer:

$logFile = "C:\Logs\cpu\_log.txt"

# Create log file if it doesn't exist

if (-not (Test-Path $logFile)) {

New-Item -Path $logFile -ItemType File

}

# Loop to log CPU usage every 5 seconds

while ($true) {

$cpuUsage = Get-Counter '\Processor(\_Total)\% Processor Time'

$cpuUsageValue = $cpuUsage.CounterSamples[0].CookedValue

$timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"

"$timestamp - CPU Usage: $cpuUsageValue%" | Out-File -Append -FilePath $logFile

Start-Sleep -Seconds 5

}