**NAME:** - ANSHUL SAINI  
**MAIL:** - [sainianshul7565@gmail.com](mailto:sainianshul7565@gmail.com)

**User ID**: - 34732

**Date: -** 09/08/2025

**Batch: -** DWS\_B5\_25VID2550

**Topic Assignment: -**

* Powershell cmdlets commands

1. **Get-NetIPConfiguration | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt"**

This command retrieves the computer’s current IP configuration details, including adapter names, IP addresses, DNS, and gateways. The | (pipeline) sends the output to Out-File, saving it to the specified text file on D: drive. It overwrites the file if it already exists.

1. **Get-PsDrive | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**

This lists all available PowerShell drives (such as C:, Env:, HKLM:) showing their type, free space, and root location. The -Append parameter adds this output to the existing file instead of overwriting, so the previous IP configuration remains intact.

1. **Get-NetIPConfiguration | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**

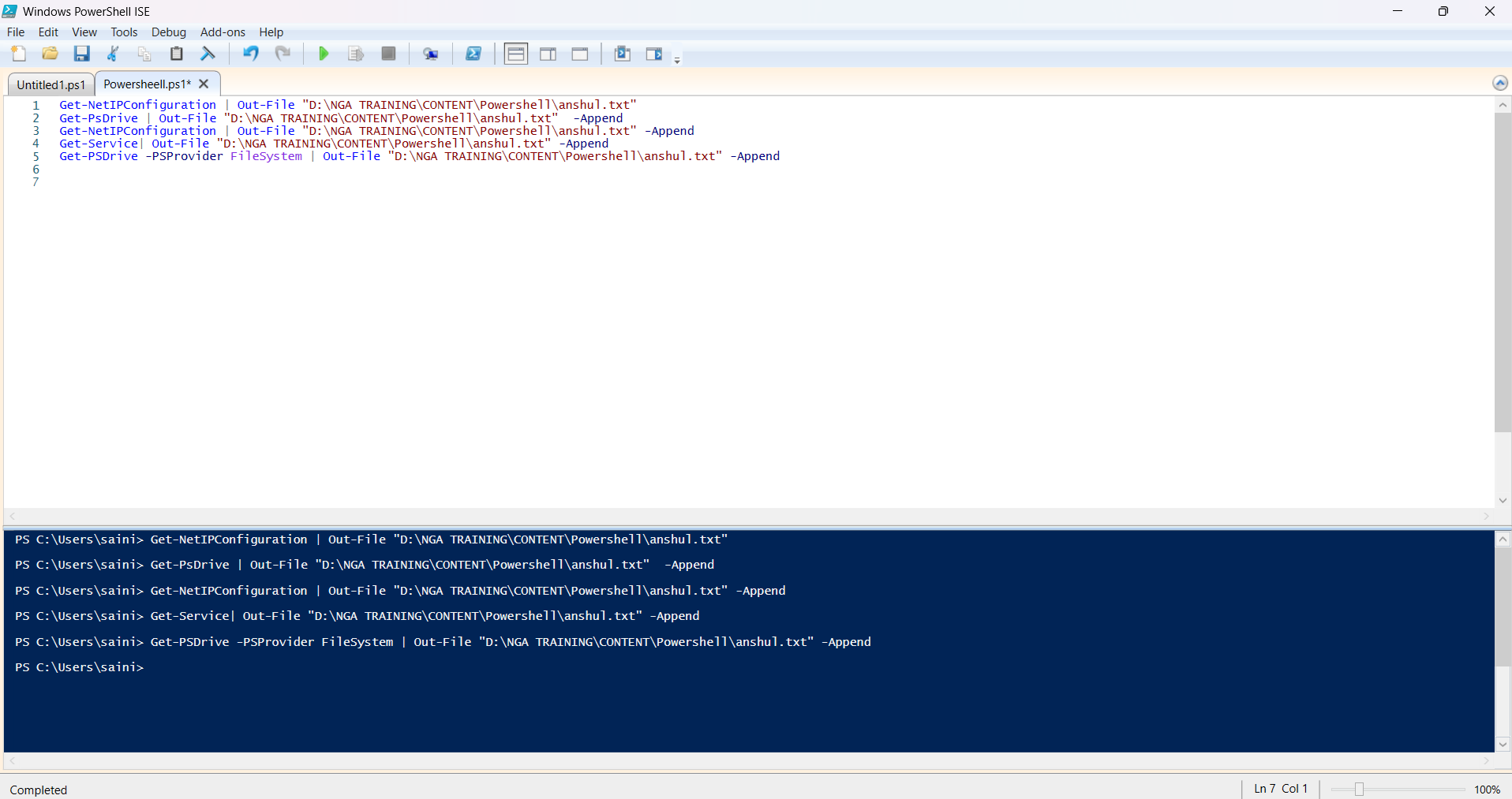
Again retrieves IP configuration (network adapter settings), but this time it appends the result to the same file. This ensures the data is saved below the previously written drive information rather than replacing it.

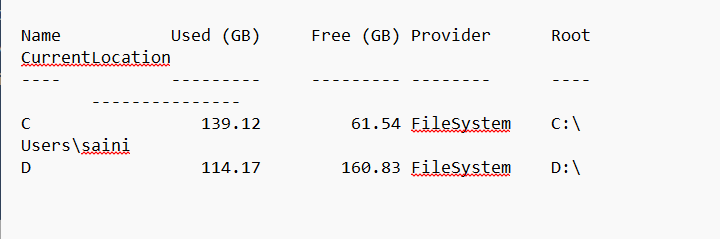
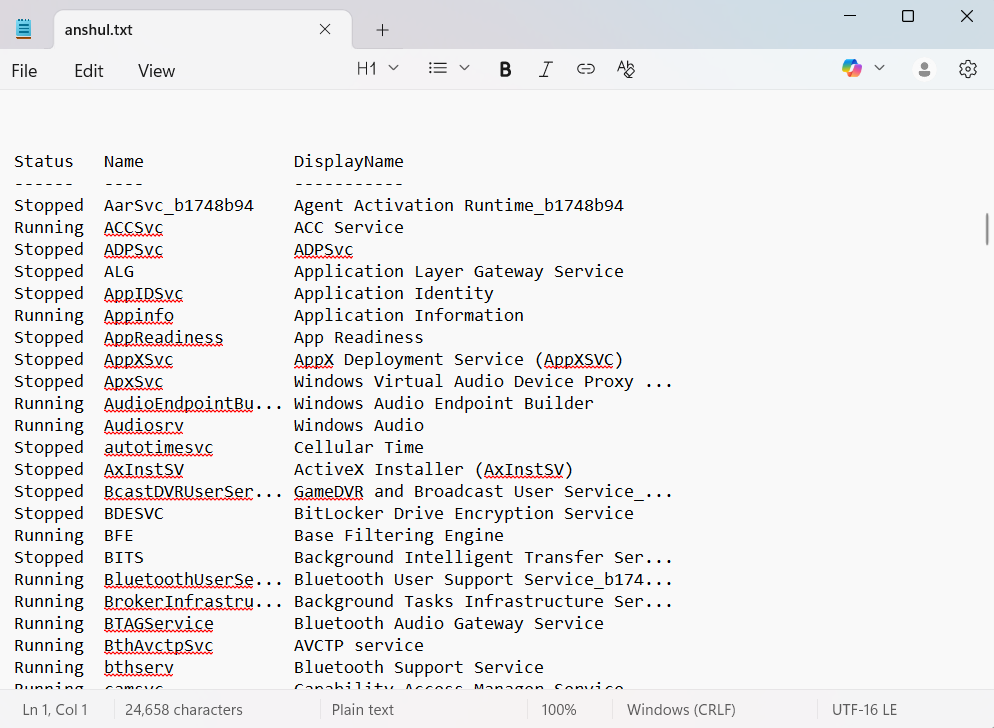
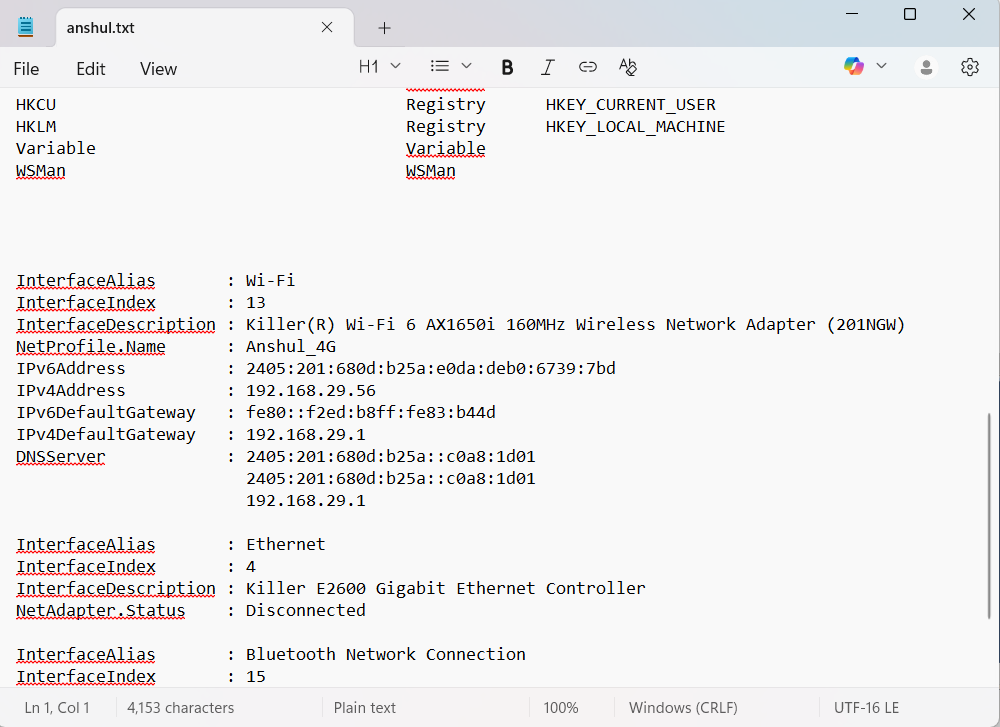
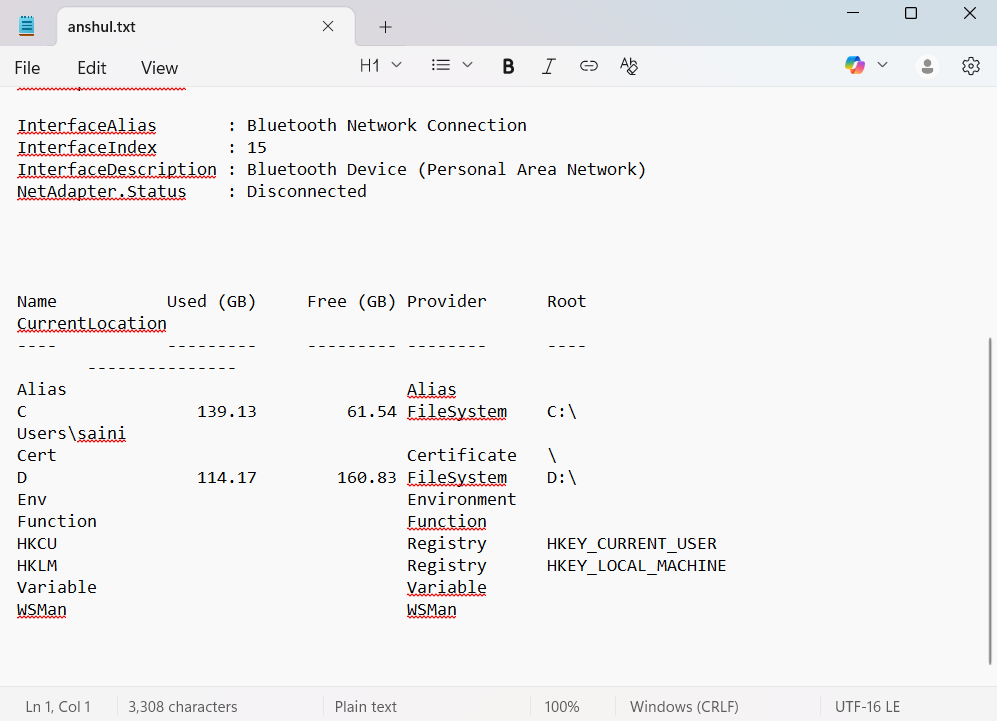
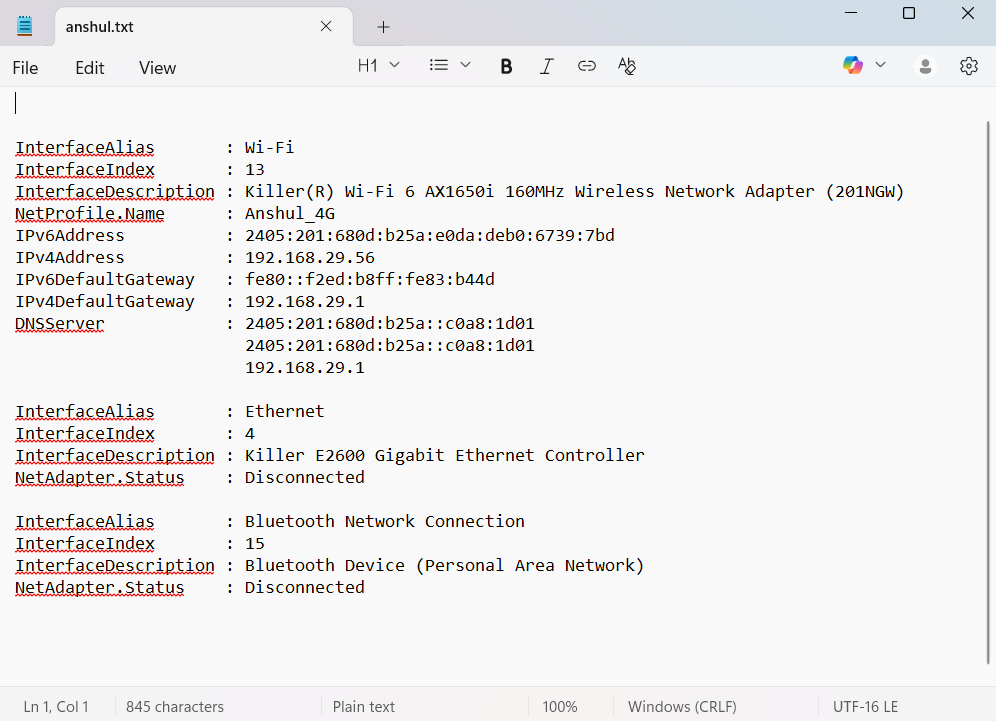
1. **Get-Service| Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**

This fetches a list of all services on the system along with their status (Running, Stopped, etc.) and service names. The output is appended to the file, allowing for later review of service health and availability.

1. **Get-PSDrive -PSProvider FileSystem | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**

This lists only file system drives (like C:, D:) from PowerShell’s drive list, excluding other providers such as Registry or Environment. The output, containing drive size and free space, is appended to the existing text file.

**Practice Activity: -  
  
**



1. **$name = Read-Host "Please enter your name"**

**Write-Host "Hello, $name!"**

This prompts the user to type their name into the console. The Read-Host cmdlet takes the user’s input and stores it in the $name variable. The quoted string is the prompt message displayed to guide the user on what to enter.

This displays the string Hello, followed by the value stored in $name. PowerShell replaces $name with the user’s input from the previous command. Write-Host outputs text directly to the console without sending it through the pipeline.

**Output: -**

PS C:\Users\saini> $name = Read-Host "Please enter your name"

Write-Host "Hello, $name!"

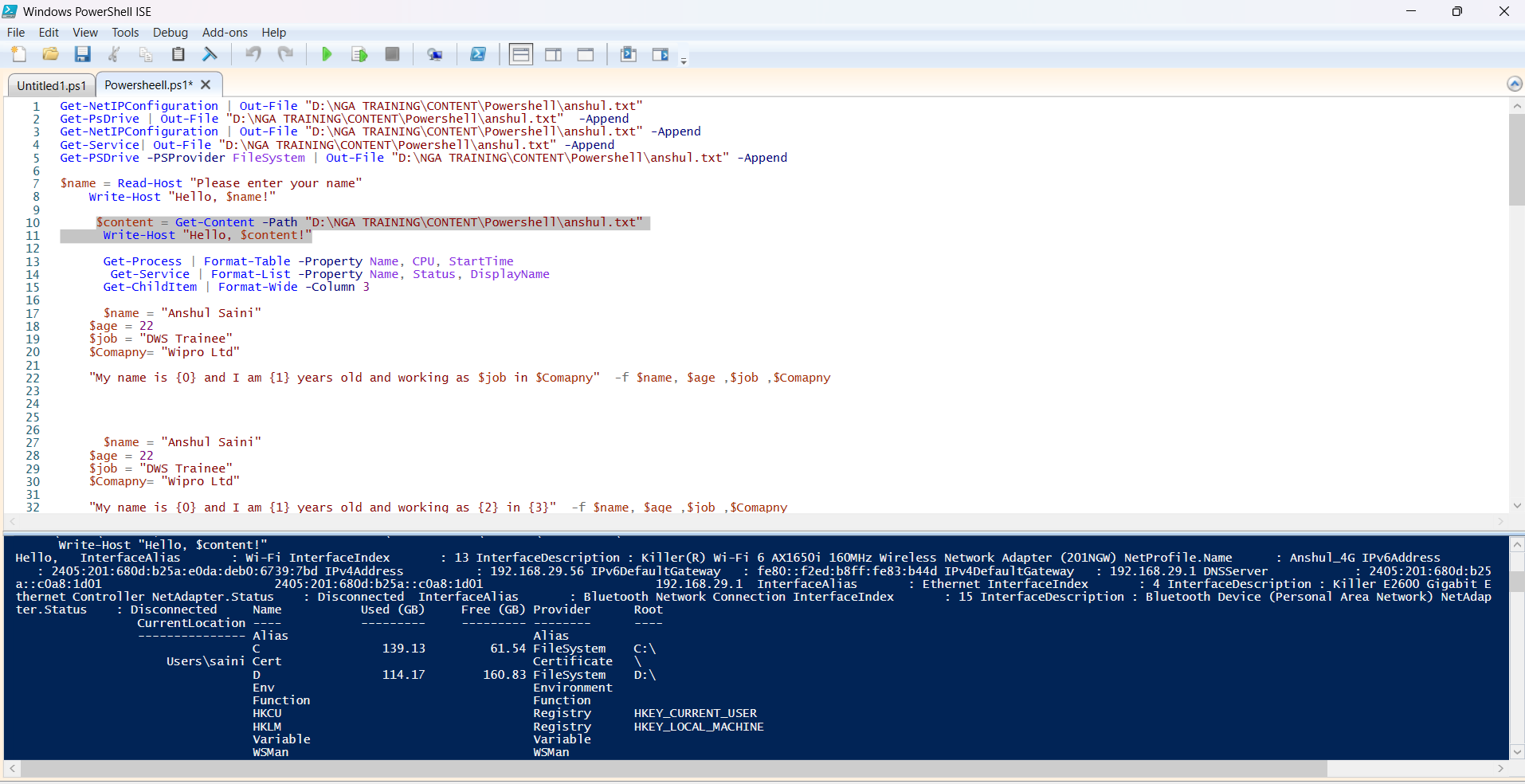
Please enter your name: Anshul

Hello, Anshul!

1. **$content = Get-Content -Path "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt"**

**Write-Host "Hello, $content!"**

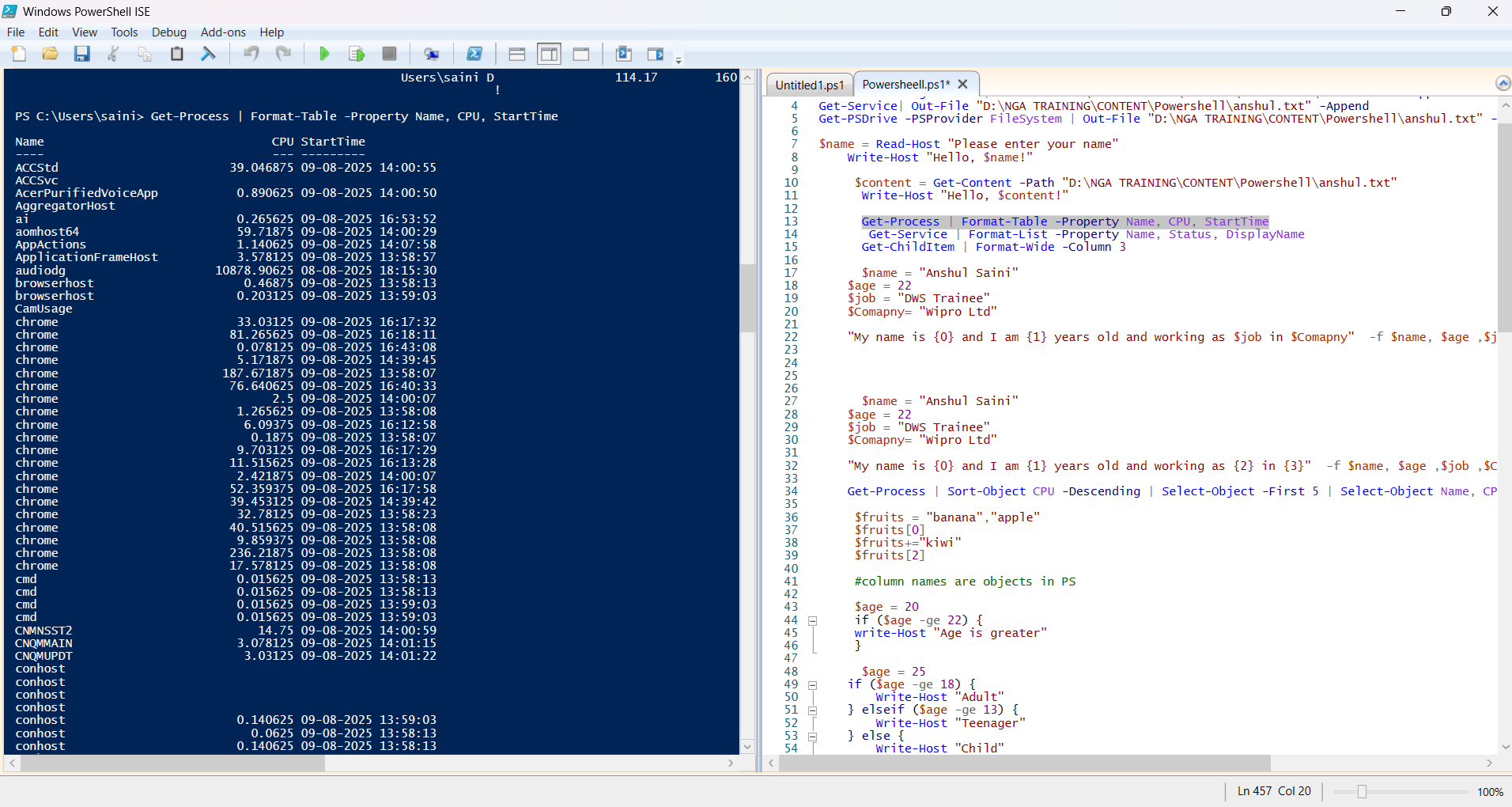
**Practice Activity: -**



1. **Get-Process | Format-Table -Property Name, CPU, StartTime**

This lists all running processes on the system, showing only their Name, CPU time used, and StartTime. The Format-Table cmdlet arranges these properties into neat table columns for easy reading.

**Practice Activity: -**

****

1. **Get-Service | Format-List -Property Name, Status, DisplayName**

This lists services but formats the output as a detailed list instead of a table. Only the service Name, its Status, and the DisplayName are shown for each service, making it easier to read when there’s more descriptive text.

**Output: -**

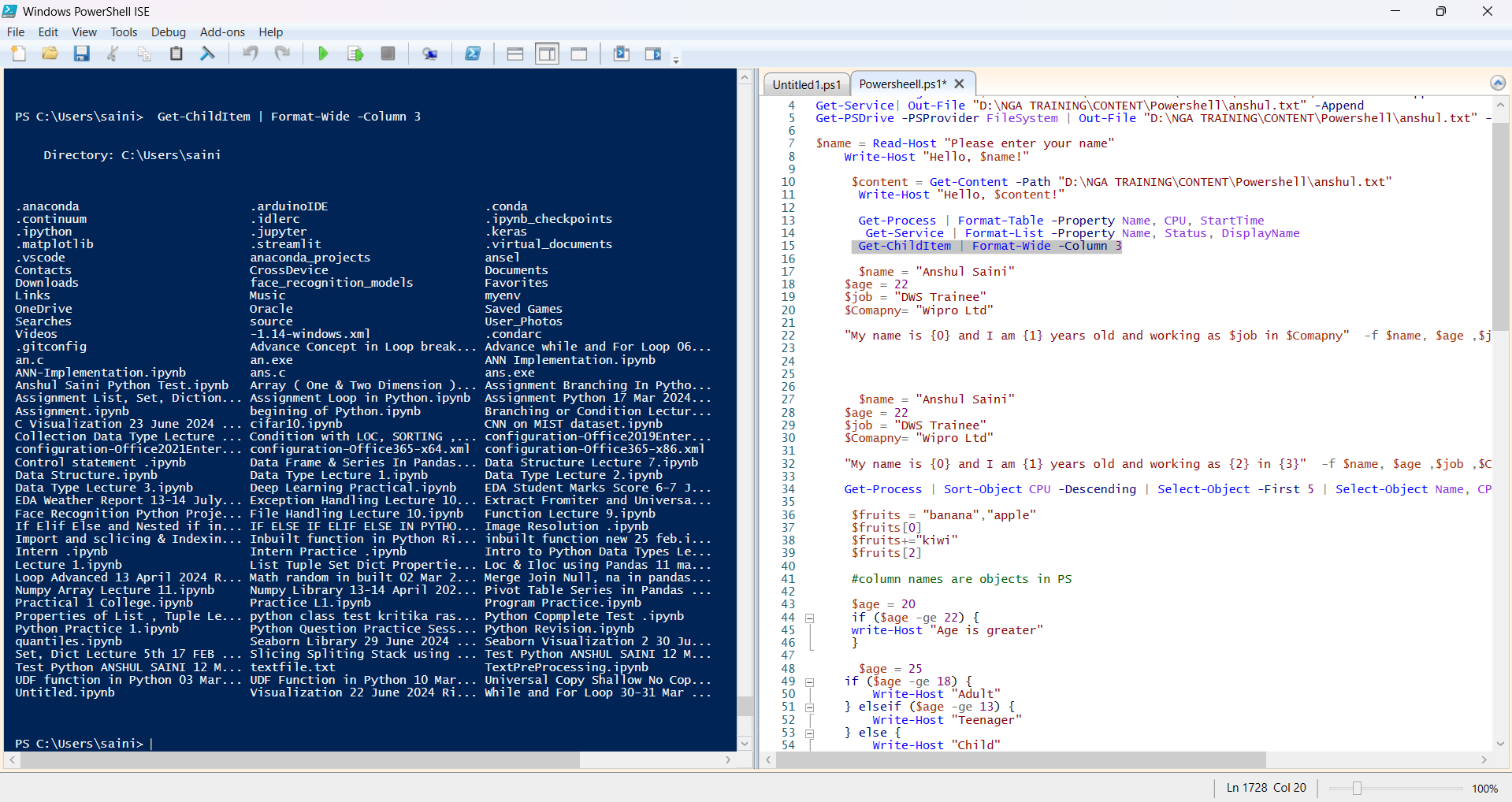
Name : XboxNetApiSvc

Status : Stopped

DisplayName : Xbox Live Networking Service

1. **Get-ChildItem | Format-Wide -Column 3**

**Practice Activity: -**



1. **11.$name = "Anshul Saini"**

**$age = 22**

**$job = "DWS Trainee"**

**$Comapny= "Wipro Ltd"**

**"My name is {0} and I am {1} years old and working as $job in $Comapny" -f $name, $age ,$job ,$Comapny**

**Output: -**

PS C:\Users\saini> $name = "Anshul Saini"

$age = 22

$job = "DWS Trainee"

$Comapny= "Wipro Ltd"

"My name is {0} and I am {1} years old and working as $job in $Comapny" -f $name, $age ,$job ,$Comapny

My name is Anshul Saini and I am 22 years old and working as DWS Trainee in Wipro Ltd

1. **Get-Process | Sort-Object CPU -Descending | Select-Object -First 5 | Select-Object Name, CPU**

This sorts all running processes by CPU usage in descending order, selects the top 5 processes, and displays only their Name and CPU usage. It’s useful for identifying the most resource-hungry processes.

**Output: -**

PS C:\Users\saini> Get-Process | Sort-Object CPU -Descending | Select-Object -First 5 | Select-Object Name, CPU

Name CPU

---- ---

audiodg 11030.140625

Zoom 3191.640625

WhatsApp 475.671875

WINWORD 253.640625

chrome 240.765625

1. **$fruits = "banana","apple"**

**$fruits[0]**

**$fruits+="kiwi"**

**$fruits[2]**

**Output: -**

PS C:\Users\saini> $fruits = "banana","apple"

$fruits[0]

$fruits+="kiwi"

$fruits[2]

banana

kiwi

1. **$age = 20**

**if ($age -ge 22) {**

**write-Host "Age is greater"**

**}**

**$age = 25**

**if ($age -ge 18) {**

**Write-Host "Adult"**

**} elseif ($age -ge 13) {**

**Write-Host "Teenager"**

**} else {**

**Write-Host "Child"**

**}**

**Output: -**

PS C:\Users\saini> $age = 20

if ($age -ge 22) {

write-Host "Age is greater"

}

$age = 25

if ($age -ge 18) {

Write-Host "Adult"

} elseif ($age -ge 13) {

Write-Host "Teenager"

} else {

Write-Host "Child"

}

Adult

1. **Get-Service | Get-Member | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
2. **Get-Date | Get-Member | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
3. **Get-Date | Select-Object –Property Second**
4. **Get-Date | Select-Object –Property TimeofDay**
5. **Get-Command \*hotfix\* Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
6. **Get-Hotfix | Get-Member**

Get-Service retrieves service objects, Get-Member lists their properties and methods, and the results are appended to the file. This is useful for learning what information you can access about each service.

This retrieves the current date and time object from Get-Date, then passes it to Get-Member to display all its properties and methods (like Day, Month, Year). The results are appended to the specified file for reference.

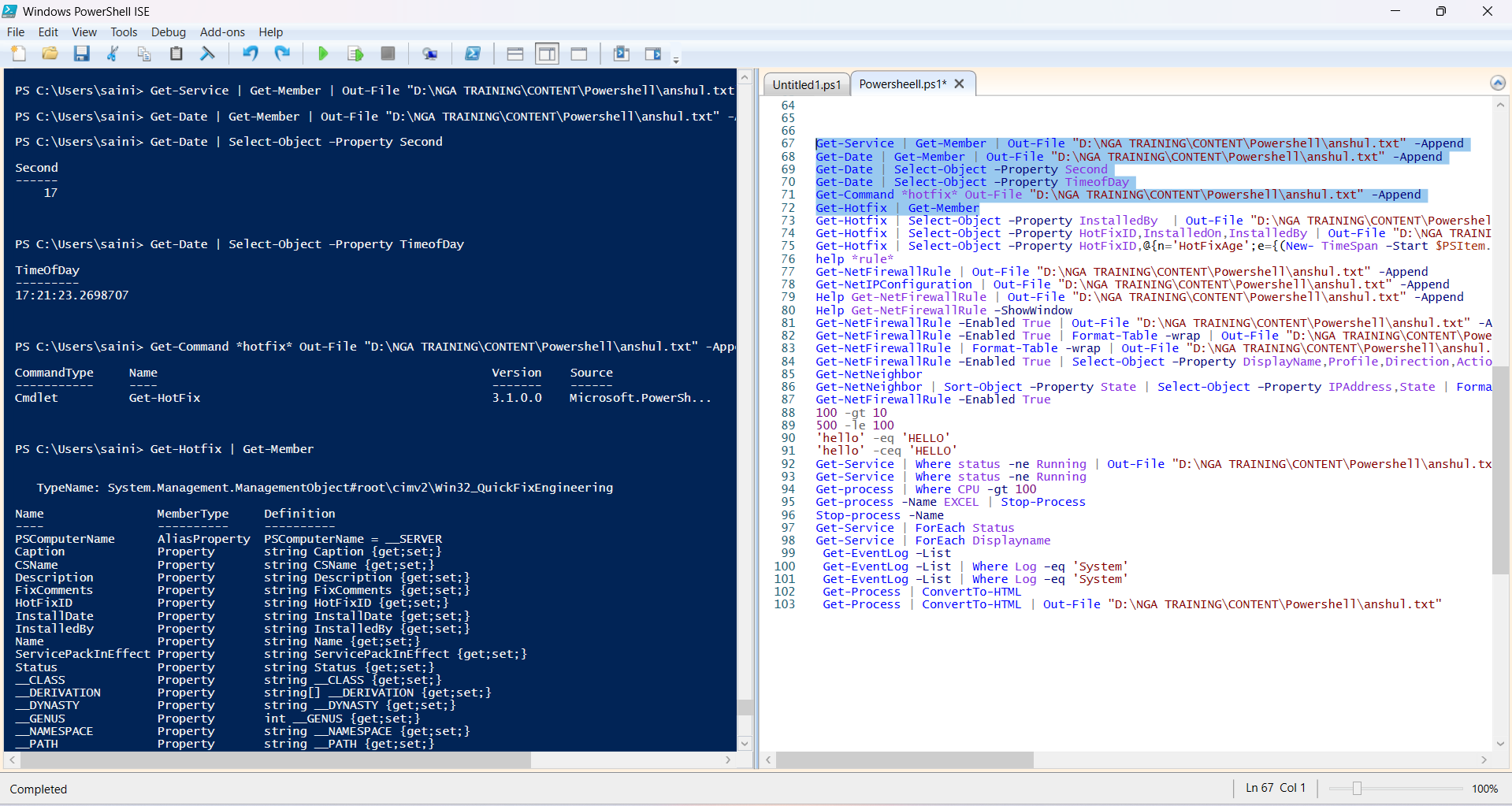
This retrieves the current date/time but only displays the Second property (value from 0 to 59). Select-Object filters the object to show just this property, hiding all other date and time details.

This returns the current time of day as a TimeSpan object, showing hours, minutes, seconds, and fractions of a second since midnight. It’s useful for precise timing or duration calculations.

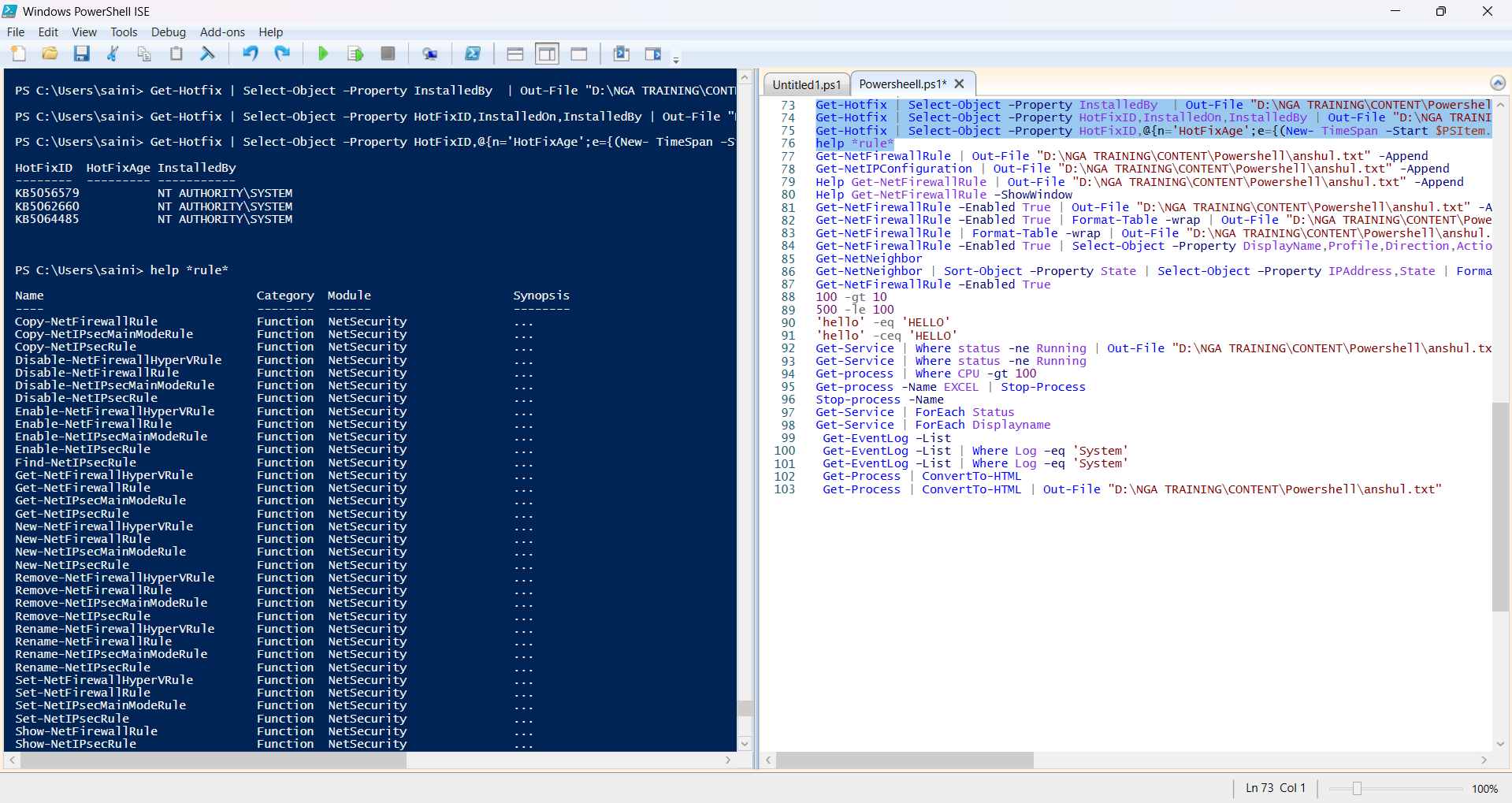
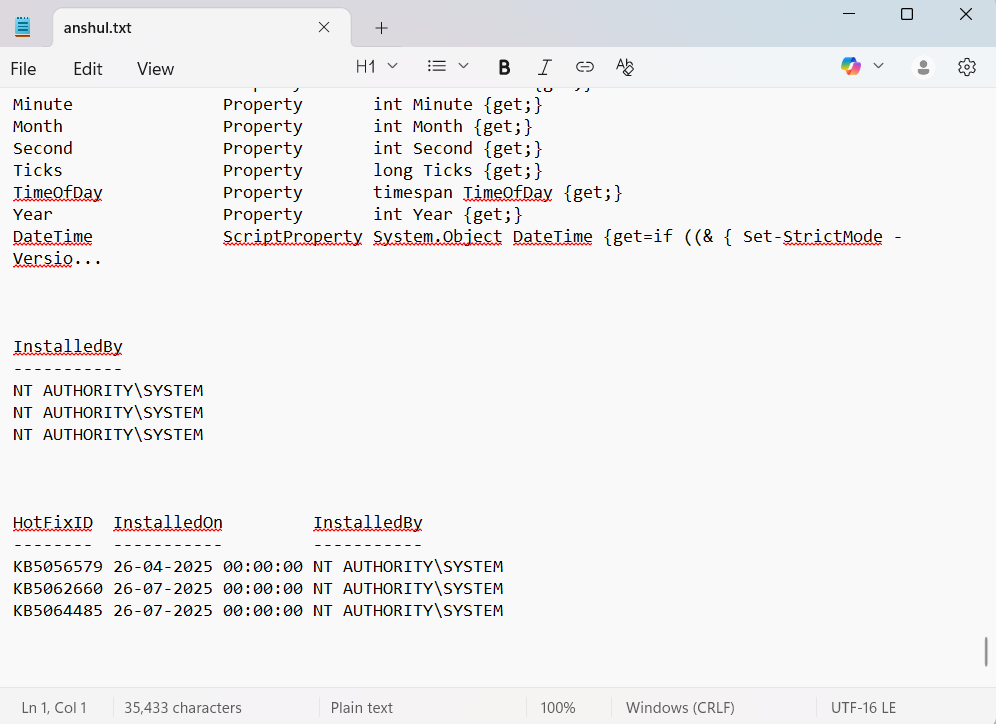
This searches for all available PowerShell commands whose names contain the word “hotfix” and appends the list to the file. It’s a way to find all cmdlets or functions related to hotfix management.

This retrieves all installed Windows updates (hotfixes), then pipes them to Get-Member to display the object’s properties and methods, showing what data you can access about each hotfix.

**Practice Activity: -**



1. **Get-Hotfix | Select-Object –Property InstalledBy | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
2. **Get-Hotfix | Select-Object –Property HotFixID,InstalledOn,InstalledBy | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
3. **Get-Hotfix | Select-Object –Property HotFixID,@{n='HotFixAge';e={(New- TimeSpan –Start $PSItem.InstalledOn).Days}},InstalledBy**
4. **help \*rule\***

**Practice Activity: -**

1. **Get-NetFirewallRule | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
2. **Get-NetIPConfiguration | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
3. **Help Get-NetFirewallRule | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
4. **Help Get-NetFirewallRule –ShowWindow**

**Output: -**

NAME

Get-NetFirewallRule

SYNTAX

Get-NetFirewallRule [-All] [-PolicyStore <string>] [-TracePolicyStore] [-CimSession

<CimSession[]>] [-ThrottleLimit <int>] [-AsJob] [<CommonParameters>]

Get-NetFirewallRule [-Name] <string[]> [-PolicyStore <string>] [-TracePolicyStore]

[-CimSession <CimSession[]>] [-ThrottleLimit <int>] [-AsJob] [<CommonParameters>]

Get-NetFirewallRule -DisplayName <string[]> [-PolicyStore <string>] [-TracePolicyStore]

[-CimSession <CimSession[]>] [-ThrottleLimit <int>] [-AsJob] [<CommonParameters>]

Get-NetFirewallRule [-Description <string[]>] [-DisplayGroup <string[]>] [-Group <string[]>]

[-Enabled {True | False}] [-Direction {Inbound | Outbound}] [-Action {NotConfigured | Allow |

Block}] [-EdgeTraversalPolicy {Block | Allow | DeferToUser | DeferToApp}]

[-LooseSourceMapping <bool[]>] [-LocalOnlyMapping <bool[]>] [-Owner <string[]>]

[-PrimaryStatus {Unknown | OK | Inactive | Error}] [-Status <string[]>] [-PolicyStoreSource

<string[]>] [-PolicyStoreSourceType {None | Local | GroupPolicy | Dynamic | Generated |

Hardcoded | MDM | HostFirewallLocal | HostFirewallGroupPolicy | HostFirewallDynamic |

HostFirewallMDM}] [-PolicyAppId <string[]>] [-PackageFamilyName <string[]>] [-PolicyStore

<string>] [-TracePolicyStore] [-CimSession <CimSession[]>] [-ThrottleLimit <int>] [-AsJob]

[<CommonParameters>]

Get-NetFirewallRule -AssociatedNetFirewallAddressFilter <CimInstance#MSFT\_NetAddressFilter>

[-PolicyStore <string>] [-TracePolicyStore] [-CimSession <CimSession[]>] [-ThrottleLimit

<int>] [-AsJob] [<CommonParameters>]

Get-NetFirewallRule -AssociatedNetFirewallApplicationFilter

<CimInstance#MSFT\_NetApplicationFilter> [-PolicyStore <string>] [-TracePolicyStore]

[-CimSession <CimSession[]>] [-ThrottleLimit <int>] [-AsJob] [<CommonParameters>]

Get-NetFirewallRule -AssociatedNetFirewallInterfaceFilter

<CimInstance#MSFT\_NetInterfaceFilter> [-PolicyStore <string>] [-TracePolicyStore]

[-CimSession <CimSession[]>] [-ThrottleLimit <int>] [-AsJob] [<CommonParameters>]

Get-NetFirewallRule -AssociatedNetFirewallInterfaceTypeFilter

<CimInstance#MSFT\_NetInterfaceTypeFilter> [-PolicyStore <string>] [-TracePolicyStore]

[-CimSession <CimSession[]>] [-ThrottleLimit <int>] [-AsJob] [<CommonParameters>]

Get-NetFirewallRule -AssociatedNetFirewallPortFilter <CimInstance#MSFT\_NetProtocolPortFilter>

[-PolicyStore <string>] [-TracePolicyStore] [-CimSession <CimSession[]>] [-ThrottleLimit

<int>] [-AsJob] [<CommonParameters>]

Get-NetFirewallRule -AssociatedNetFirewallSecurityFilter

<CimInstance#MSFT\_NetNetworkLayerSecurityFilter> [-PolicyStore <string>] [-TracePolicyStore]

[-CimSession <CimSession[]>] [-ThrottleLimit <int>] [-AsJob] [<CommonParameters>]

Get-NetFirewallRule -AssociatedNetFirewallServiceFilter <CimInstance#MSFT\_NetServiceFilter>

[-PolicyStore <string>] [-TracePolicyStore] [-CimSession <CimSession[]>] [-ThrottleLimit

<int>] [-AsJob] [<CommonParameters>]

Get-NetFirewallRule -AssociatedNetFirewallProfile <CimInstance#MSFT\_NetFirewallProfile>

[-PolicyStore <string>] [-TracePolicyStore] [-CimSession <CimSession[]>] [-ThrottleLimit

<int>] [-AsJob] [<CommonParameters>]

ALIASES

None

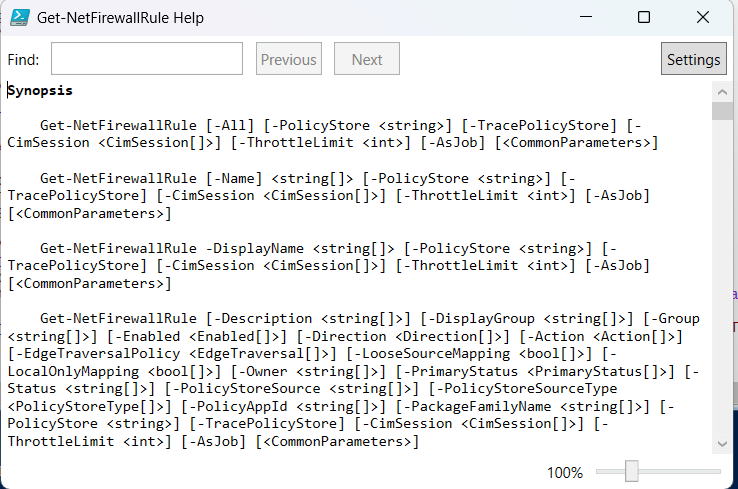
REMARKS

Get-Help cannot find the Help files for this cmdlet on this computer. It is displaying only

partial help.

-- To download and install Help files for the module that includes this cmdlet, use

Update-Help.



1. **Get-NetFirewallRule –Enabled True | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
2. **Get-NetFirewallRule –Enabled True | Format-Table -wrap | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
3. **Get-NetFirewallRule | Format-Table -wrap | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
4. **Get-NetFirewallRule –Enabled True | Select-Object –Property DisplayName,Profile,Direction,Action | Sort-Object –Property Profile, DisplayName | Format-Table -wrap | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
5. **Get-NetNeighbor**
6. **Get-NetNeighbor | Sort-Object –Property State | Select-Object –Property IPAddress,State | Format-Wide -GroupBy State - | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
7. **Get-NetFirewallRule –Enabled True**
8. **100 -gt 10**

**500 -le 100**

**'hello' -eq 'HELLO'**

**'hello' -ceq 'HELLO'**

**Output: -**

PS C:\Users\saini> 100 -gt 10

500 -le 100

'hello' -eq 'HELLO'

'hello' -ceq 'HELLO'

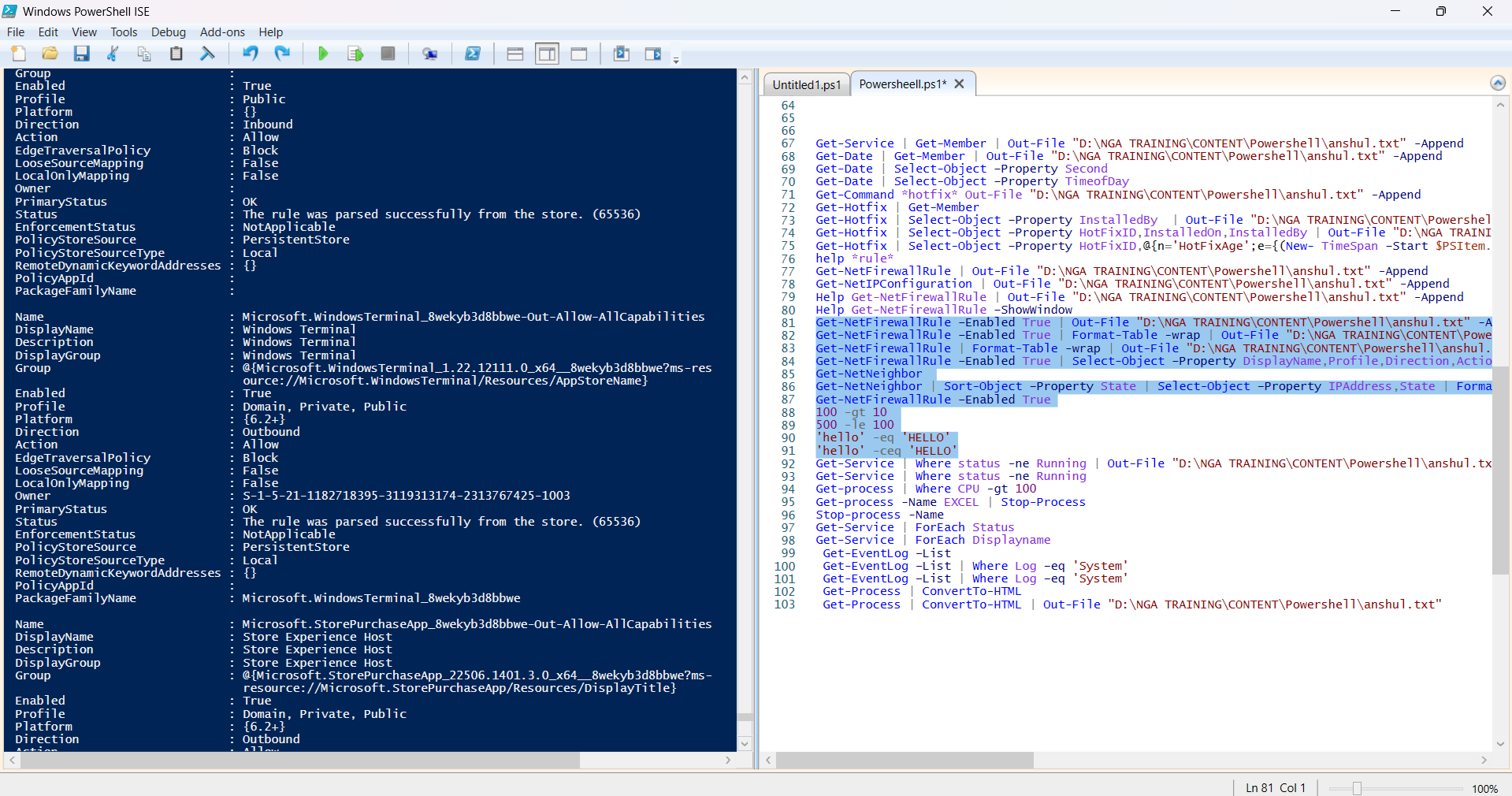
True

False

True

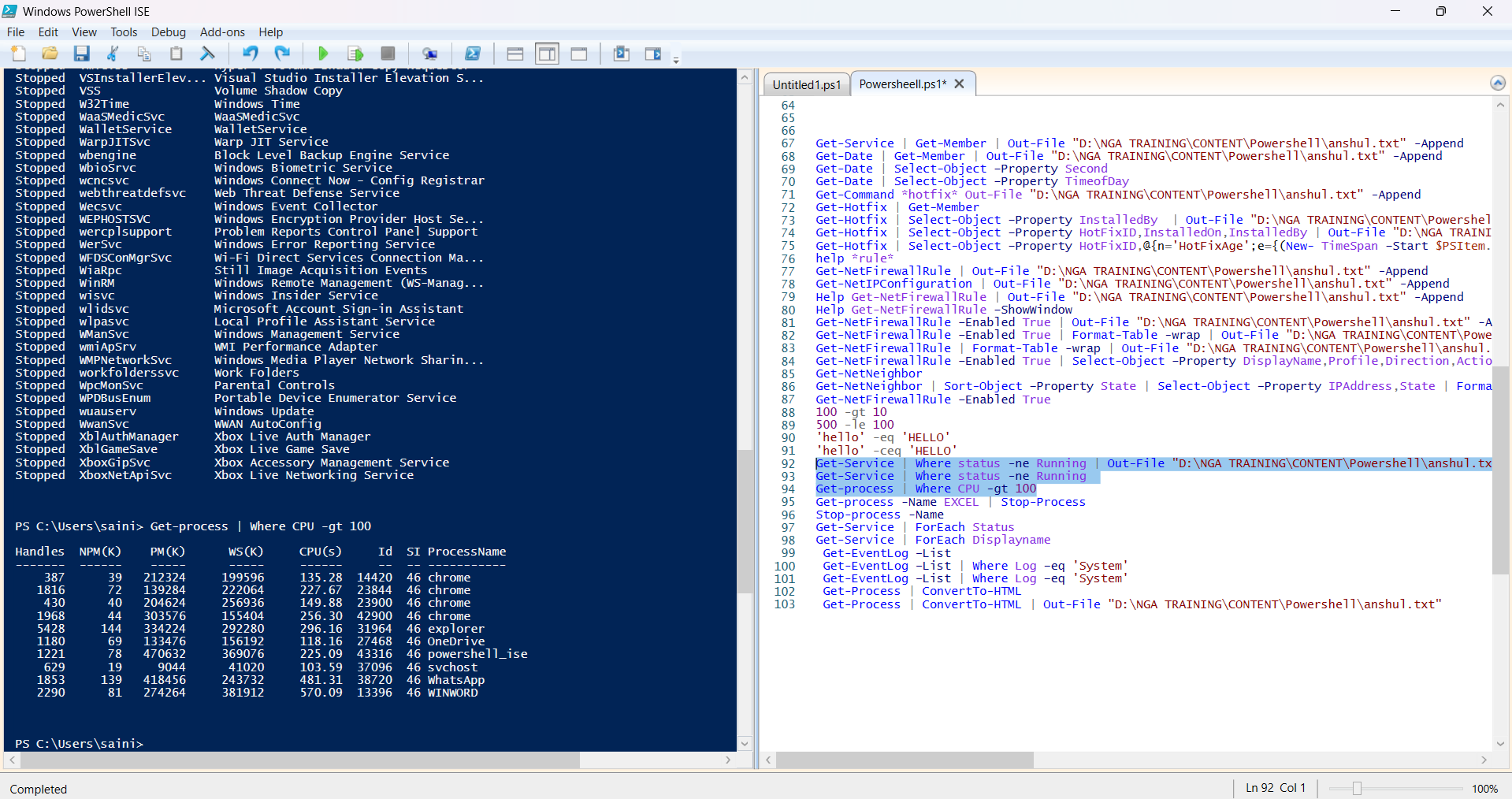
False

**Practice Activity: -**

****

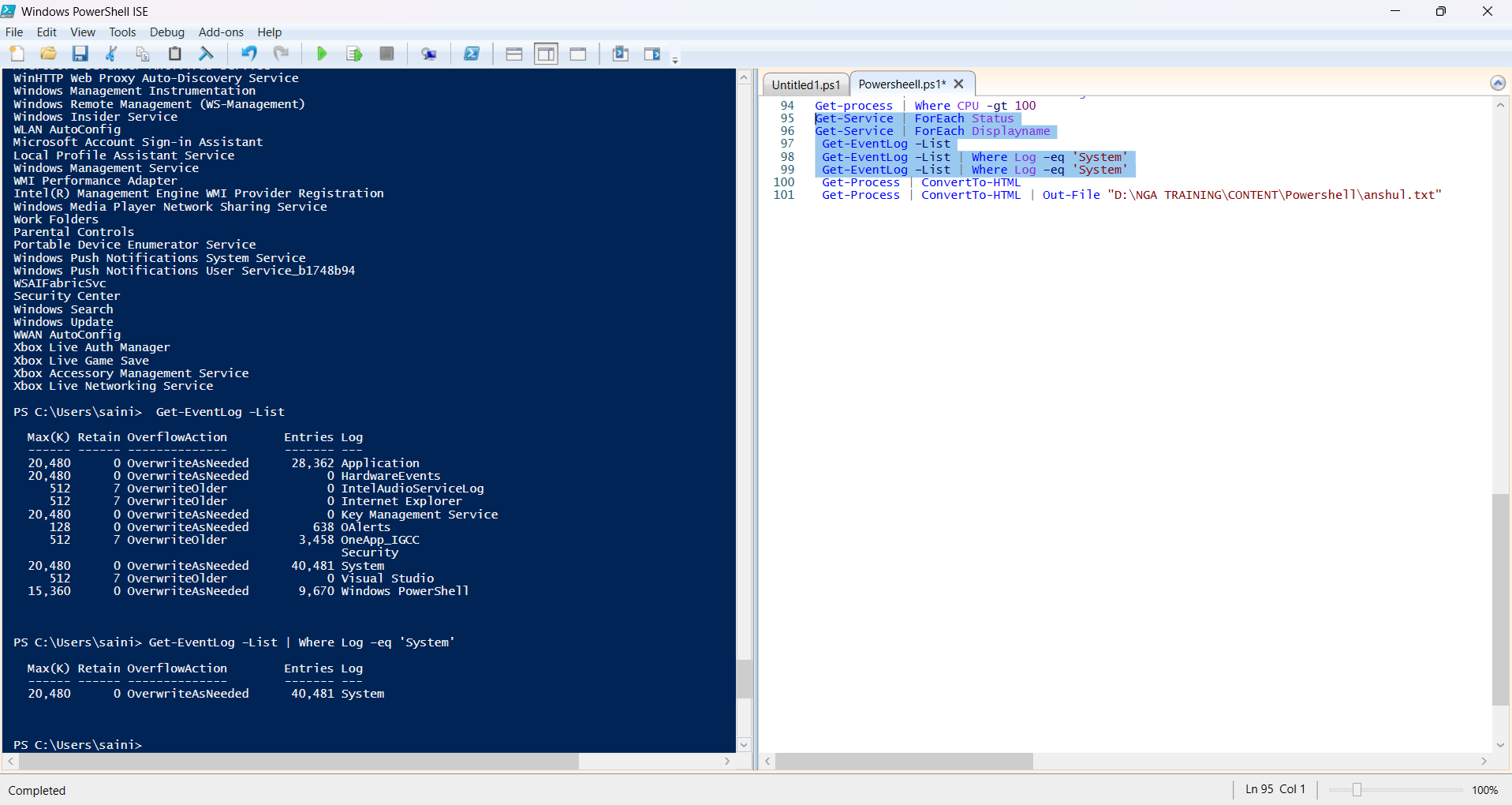
1. **Get-Service | Where status -ne Running | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt" -Append**
2. **Get-Service | Where status -ne Running**
3. **Get-process | Where CPU -gt 100**

**Practice Activity: -**

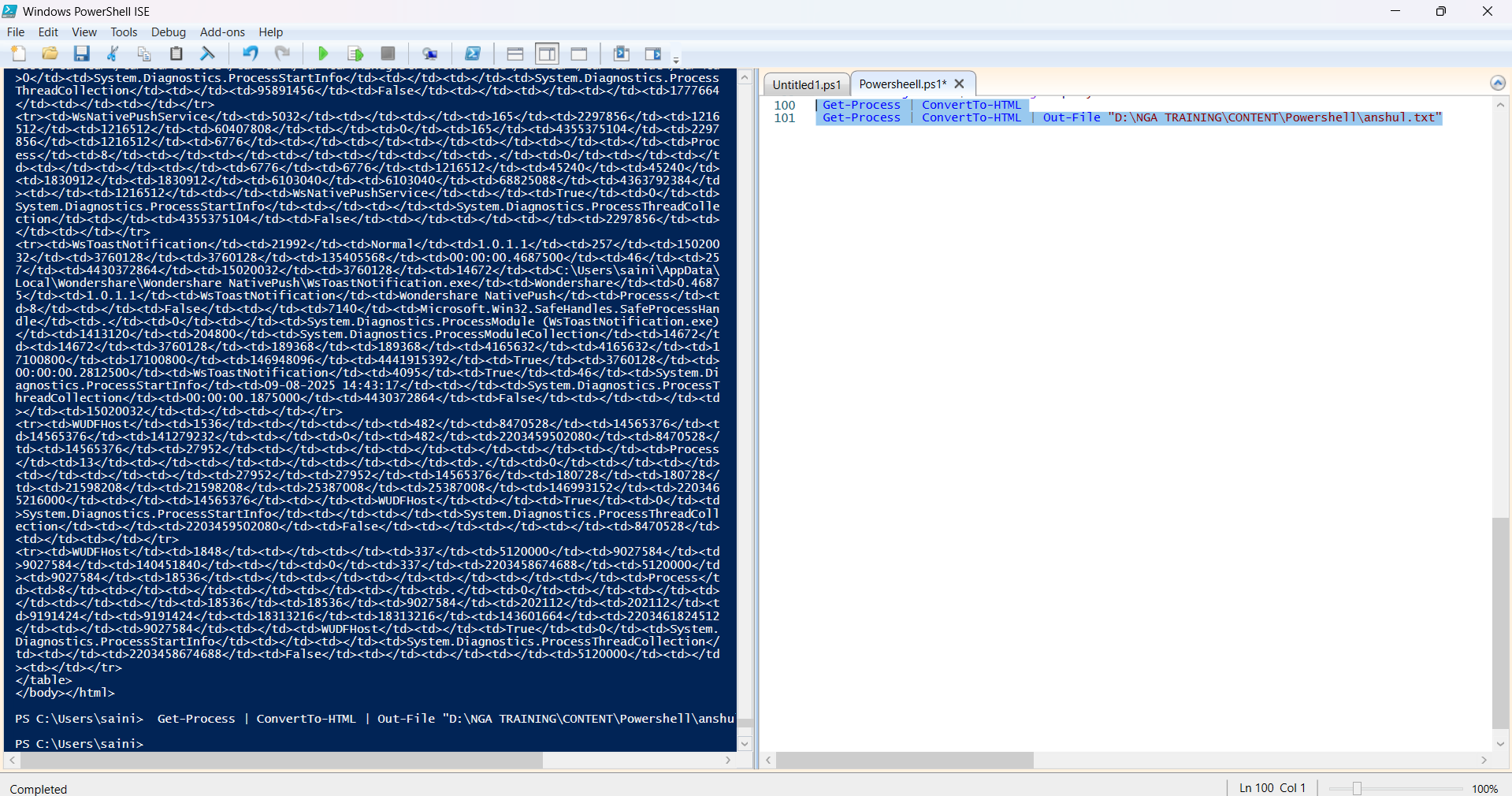
****

1. **Get-Service | ForEach Status**
2. **Get-Service | ForEach Displayname**
3. **Get-EventLog –List**
4. **Get-EventLog –List | Where Log –eq 'System'**

**Practice Activity: -**

****

1. **Get-Process | ConvertTo-HTML**
2. **Get-Process | ConvertTo-HTML | Out-File "D:\NGA TRAINING\CONTENT\Powershell\anshul.txt"**

**Practice Activity: -**

