## Practice Activity 4

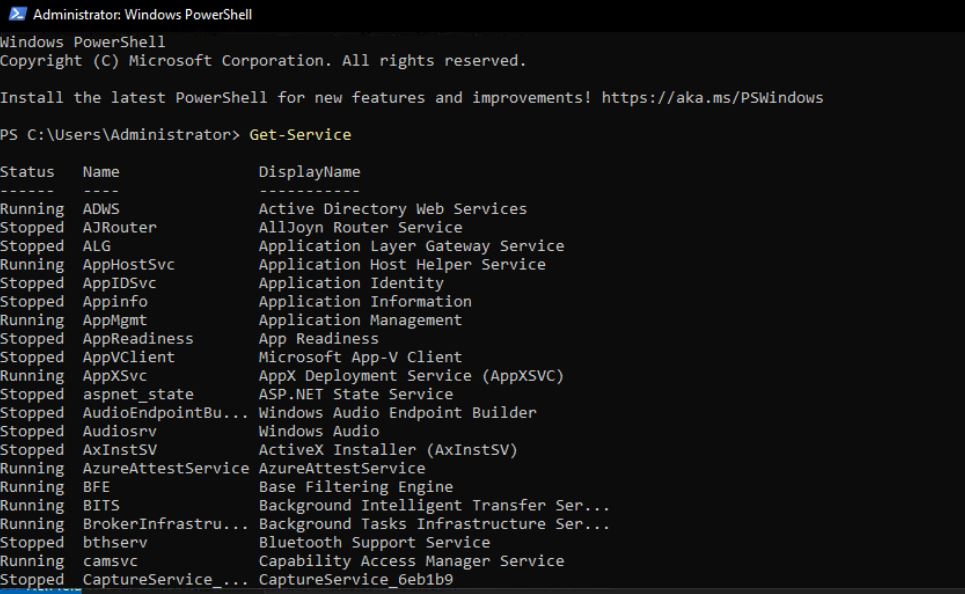
Name: Zahida

Mail id: zahidajmi1809@gmail.com

## Summary

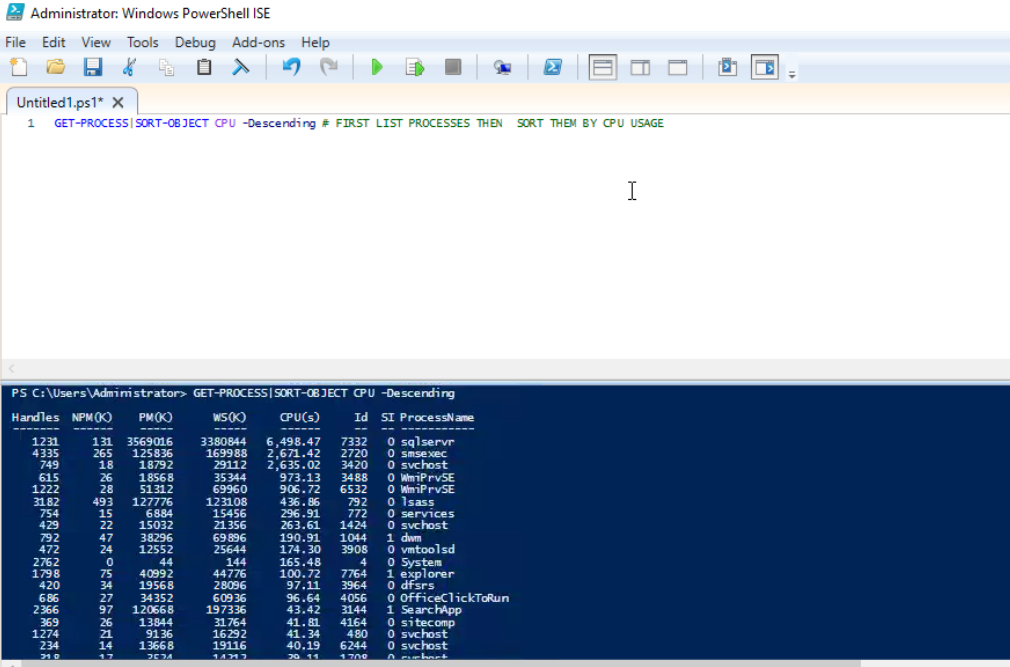
**Topics:** Introducing to Cmdlets, The powershell pipeline, Key Cmdlets, WMI & Powershell, Pipeline filtering & operators, Input output & Formatting, Scripting Overview.

### Introduction to Cmdlets

Cmdlets are specialized commands in PowerShell designed to perform specific tasks efficiently. Unlike traditional commands, they follow a verb-noun naming pattern (e.g., Get-Process, Set-Service) and output objects rather than plain text, which makes them highly versatile for automation.

### The PowerShell Pipeline

The pipeline allows the output of one cmdlet to be passed directly as input to another, enabling chaining of commands. This object-based pipeline lets you manipulate and filter data step-by-step without needing intermediate files or manual intervention.



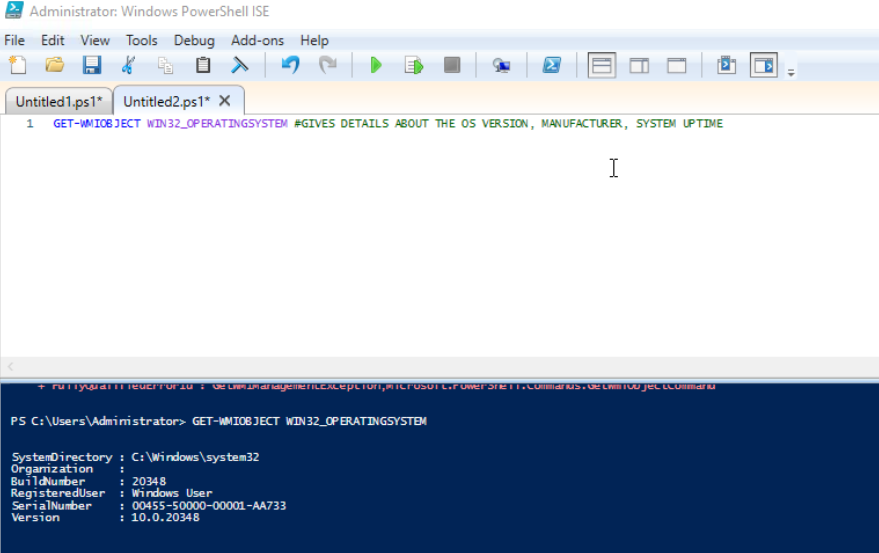
### Key Cmdlets

Some fundamental cmdlets every PowerShell user should know include:

* Get-Help: Displays documentation and usage information.
* Get-Command: Lists available cmdlets and functions.
* Get-Service: Retrieves status of Windows services.
* Get-Process: Shows running processes on the system.  
  These cmdlets help explore and manage Windows environments effectively.

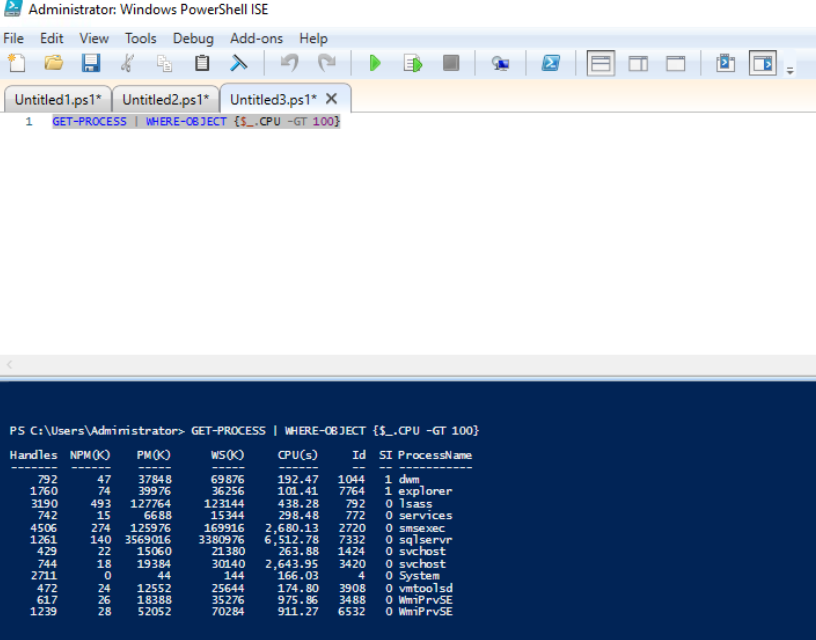
### WMI & PowerShell

Windows Management Instrumentation (WMI) is a powerful interface for accessing detailed system and hardware information. PowerShell uses cmdlets like Get-WmiObject or Get-CimInstance to query WMI classes, allowing administrators to retrieve and configure system settings programmatically.



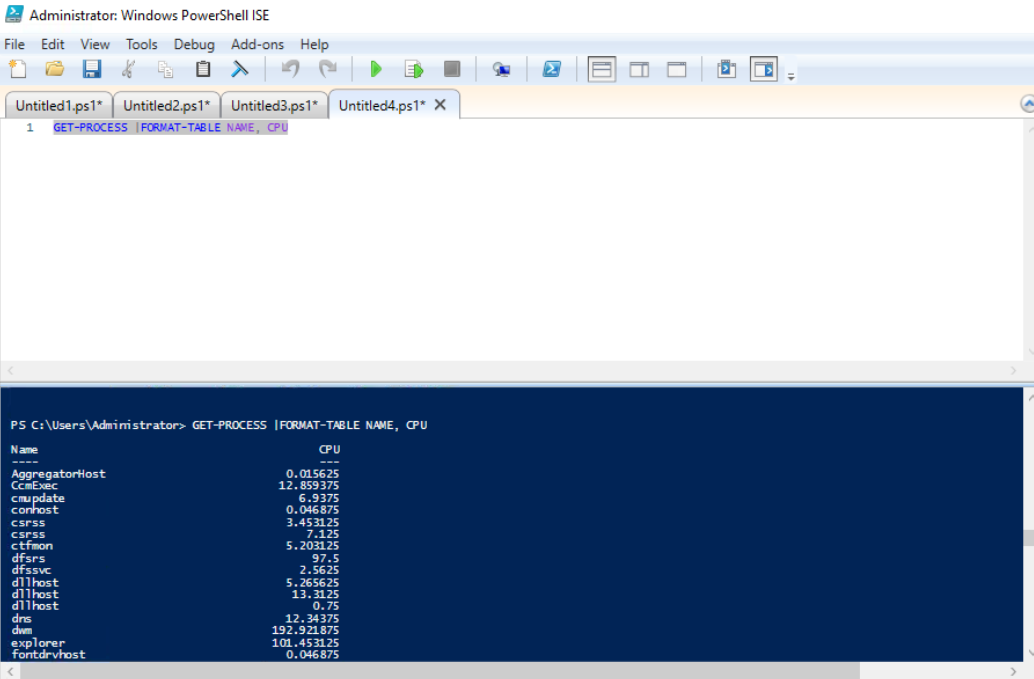
### Pipeline Filtering & Operators

Filtering is crucial for narrowing down data. The Where-Object cmdlet combined with comparison operators like -eq (equals), -ne (not equals), -gt (greater than), and -like (pattern matching) helps refine the output within the pipeline, making it easier to work with only relevant information.



1. **Input, Output & Formatting**

PowerShell accepts input from users, files, or other commands. Its output is in the form of objects that can be formatted for display using Format-Table, Format-List, or exported to CSV, JSON, or XML. Proper formatting improves readability and facilitates data exchange.



### Scripting Overview

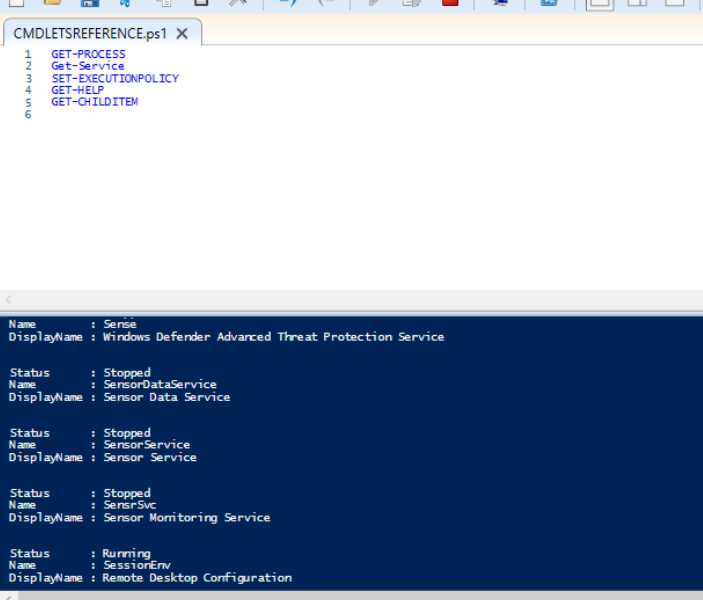
PowerShell scripting extends command-line capabilities by enabling the creation of scripts that include variables, loops, conditionals, and functions. This allows automation of complex workflows, repetitive tasks, and configuration management, making system administration more efficient.

**Project 1:**

**Exploring Cmdlet Syntax**

This project focuses on understanding how PowerShell cmdlets are written and used. A **Cmdlet Reference Guide** is created by opening PowerShell ISE or any editor and making a script named CmdletReference.ps1. Choose five cmdlets like Get-Process, Get-Service, Set-ExecutionPolicy, Get-Help, and Get-ChildItem. For each one, write its description, syntax, common parameters, and a usage example. After writing, run the script to test them.  
**Outcome:** Improves understanding of cmdlet structure, parameters, and documentation for better use in real tasks.

(screenshot of Get-Process, Get-Service, Set-ExecutionPolicy, Get-Help, and Get-ChildItem command)



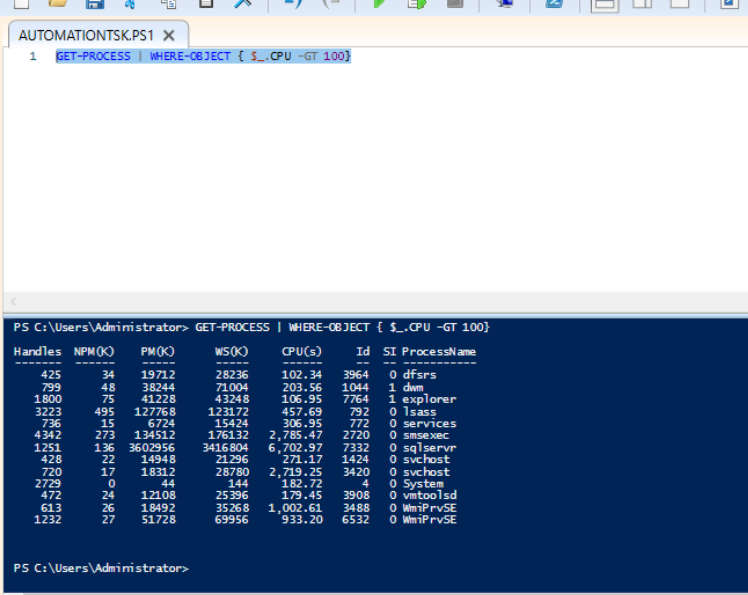
## ****Project 2:****

## ****Automate a Task with a Cmdlet Script****

Here, the aim is to create a script to perform a routine task automatically. For example, checking for processes with high CPU usage or cleaning temporary files. Create a script named AutomationTask.ps1 that uses:

* Get-Process | Where-Object {$\_.CPU -gt 100} to find high CPU processes.
* Remove-Item to delete temporary files.  
  Combine them and run the script with proper permissions.  
  **Outcome:** Teaches how to chain cmdlets, write automation scripts, and manage system tasks efficiently.

(screenshot of Get-Process | Where-Object {$\_.CPU -gt 100} command)



## ****Project 3:****

## ****Create a PowerShell Cmdlet Cheat Sheet****

This project involves creating a quick reference for cmdlets. Make a file named PowerShellCheatSheet.md and organize it into:

* **Basic Cmdlets** – e.g., Get-Help, Get-Command, Get-Content.
* **File System Cmdlets** – e.g., New-Item, Copy-Item, Remove-Item.
* **Network Cmdlets** – e.g., Test-Connection, Get-NetIPAddress.  
  Include short examples for each cmdlet.  
  **Outcome:** Helps store and organize cmdlet knowledge for quick access during scripting and troubleshooting.

**(screenshot of New-Item -Path "C:\Users\test.txt" -ItemType "File" command)**

