**Batch information:**

* **Batch Start Date: 2025-08-04**
* **Batch Name: WiproNGA\_DWS\_B5\_25VID2550**
* **First Name: Seetal**
* **Last Name: Biswal**
* **User Id: 34933**
* **Batch ID: 25VID2550**
* **Topic:** Introducing to Cmdlets

The PowerShell Pipeline

Key Cmdlets

WMI & PowerShell

Pipeline Filtering & Operators

Input, Output & Formatting

Scripting Overview

**Introducing to Cmdlets**:- Cmdlets (pronounced "command-lets") are the building blocks of PowerShell scripting. They're specialized .NET classes that implement specific functions, allowing you to perform various tasks.

**Characteristics of Cmdlets:**

1. Single-purpose: Each cmdlet performs a single, well-defined task.

2. Verb-Noun syntax: Cmdlets follow a consistent naming convention, making them easy to discover and use.

3. Pipelining: Cmdlets can be combined in a pipeline to perform complex tasks.

**Common Cmdlet Verbs:**

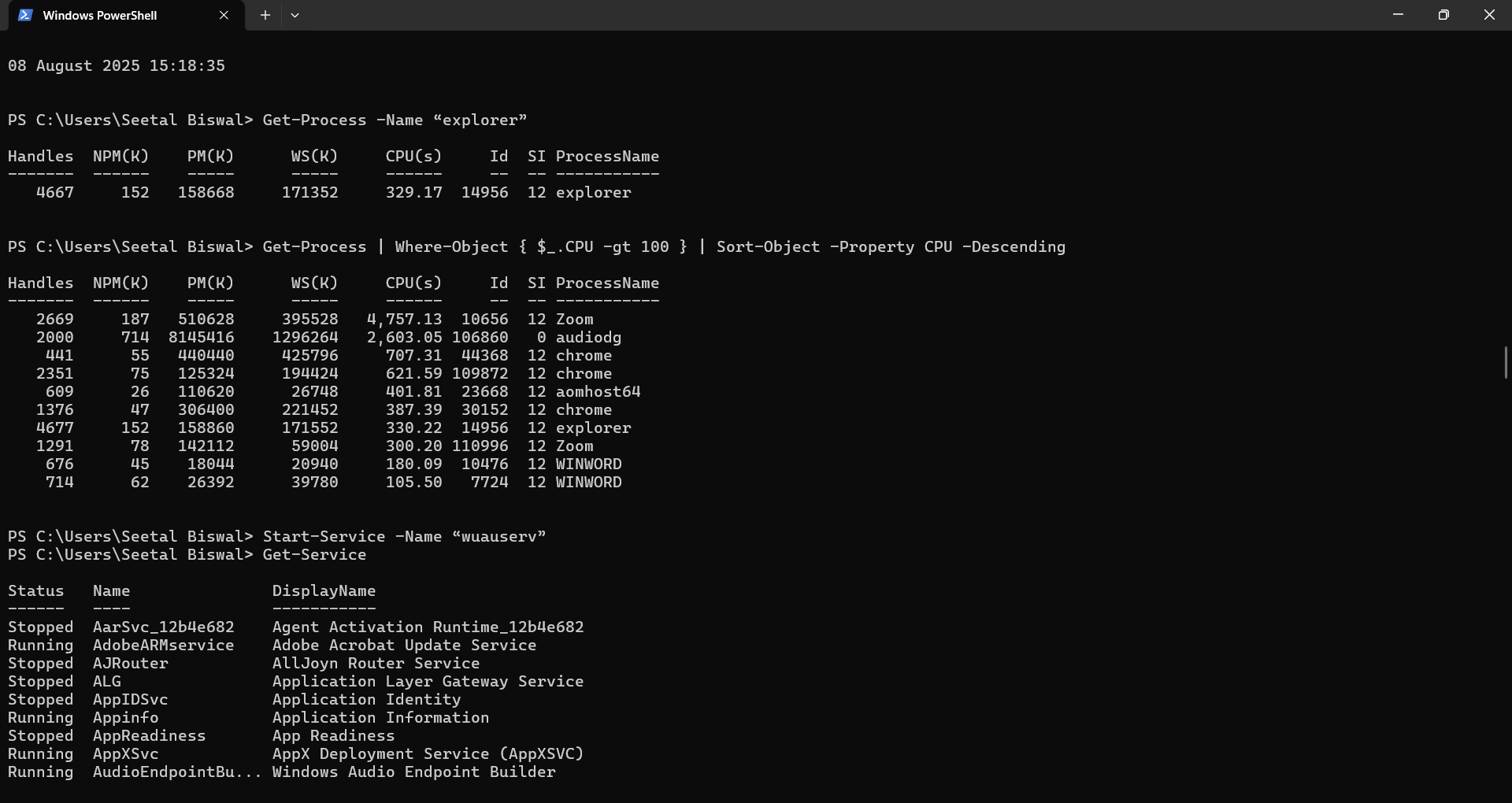
1. Get: Retrieves information or data.

2. Set: Configures or modifies settings.

3. New: Creates new objects or resources.

4. Remove: Deletes objects or resources.

5. Invoke: Executes a command or script.



**The PowerShell Pipeline:** The PowerShell pipeline is a powerful feature that allows you to chain multiple cmdlets together to perform complex tasks. Here's how it works:

**Pipeline Basics:**

1. Cmdlet output: Each cmdlet produces output that can be piped to the next cmdlet.

2. Pipeline operator: The pipeline operator (|) is used to pass output from one cmdlet to another.

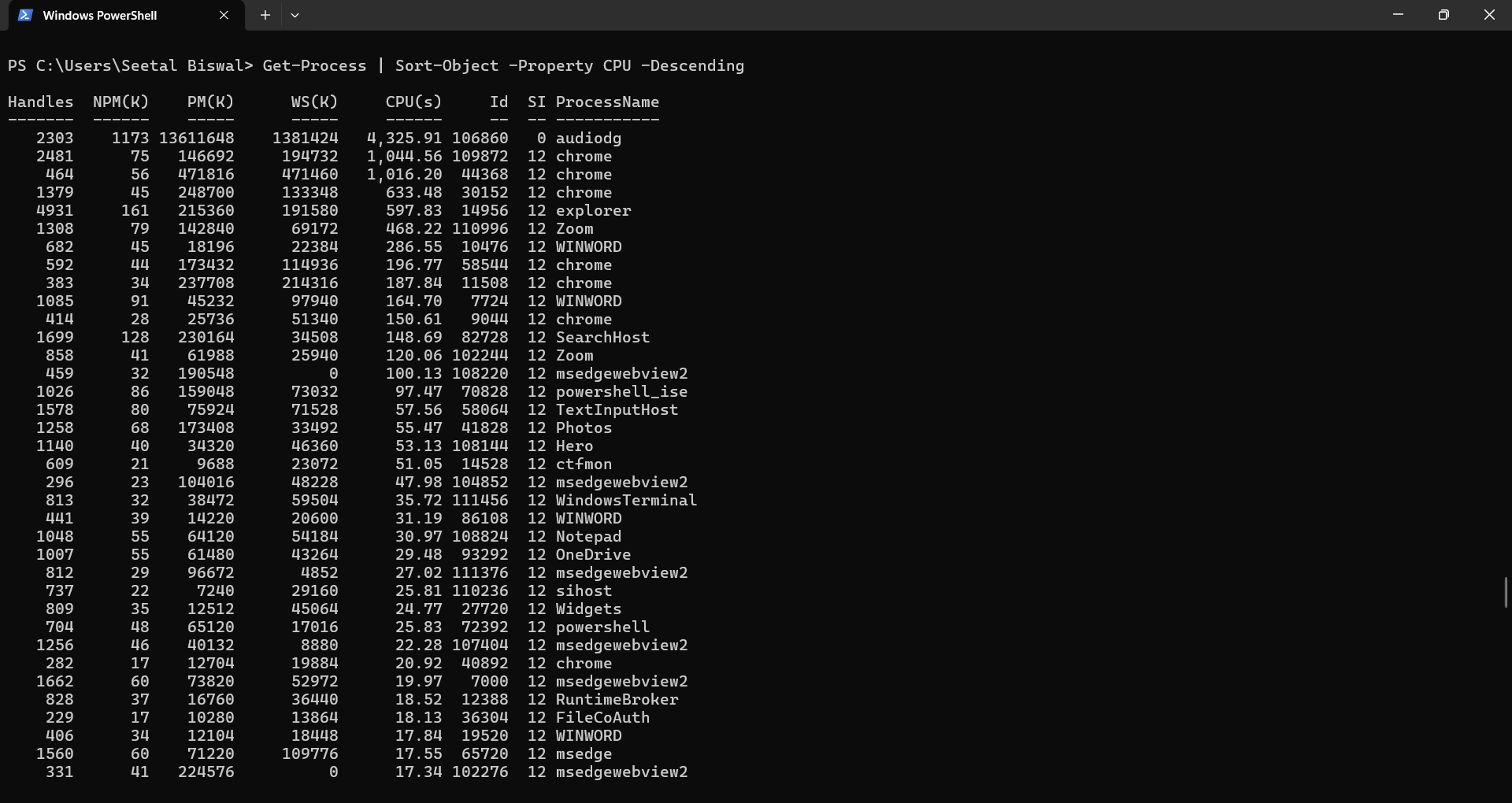
**How Pipelining Works:**

1. Cmdlet 1 output: The first cmdlet produces output.

2. Pipeline operator: The output is passed to the next cmdlet using the pipeline operator (|).

3. Cmdlet 2 processing: The second cmdlet processes the output from the first cmdlet.

4. Output: The final output is displayed or passed to the next cmdlet.

****

**Key Cmdlets:**

**Navigation and File Management:**

1. Get-Location: Gets the current directory.

2. Set-Location: Changes the current directory.

3. Get-ChildItem: Gets a list of files and folders in a directory.

4. New-Item: Creates a new file or folder.

5. Remove-Item: Deletes a file or folder.

**Process Management:**

1. Get-Process: Gets a list of running processes.

2. Stop-Process: Stops a running process.

3. Start-Process: Starts a new process.

**System Information:**

1. Get-ComputerInfo: Gets information about the computer.

2. Get-EventLog: Gets event log entries.

3. Get-Service: Gets a list of services.

**Networking:**

1. Get-NetIPConfiguration: Gets network IP configuration.

2. Test-NetConnection: Tests network connectivity.

3. Invoke-WebRequest: Sends an HTTP request.

**Active Directory:**

1. Get-ADUser: Gets Active Directory user information.

2. Get-ADGroup: Gets Active Directory group information.

3. Get-ADComputer: Gets Active Directory computer information.

**PowerShell Management:**

1. Get-Command: Gets a list of available cmdlets.

2. Get-Help: Displays help information for a cmdlet.

3. Get-Member: Gets information about an object's properties and methods.

**WMI & PowerShell:** WMI (Windows Management Instrumentation) is a powerful tool for managing and monitoring Windows systems. PowerShell provides a rich set of cmdlets for interacting with WMI.

**WMI Basics:**

1. WMI Classes: WMI classes represent managed objects, such as processes, services, and hardware.

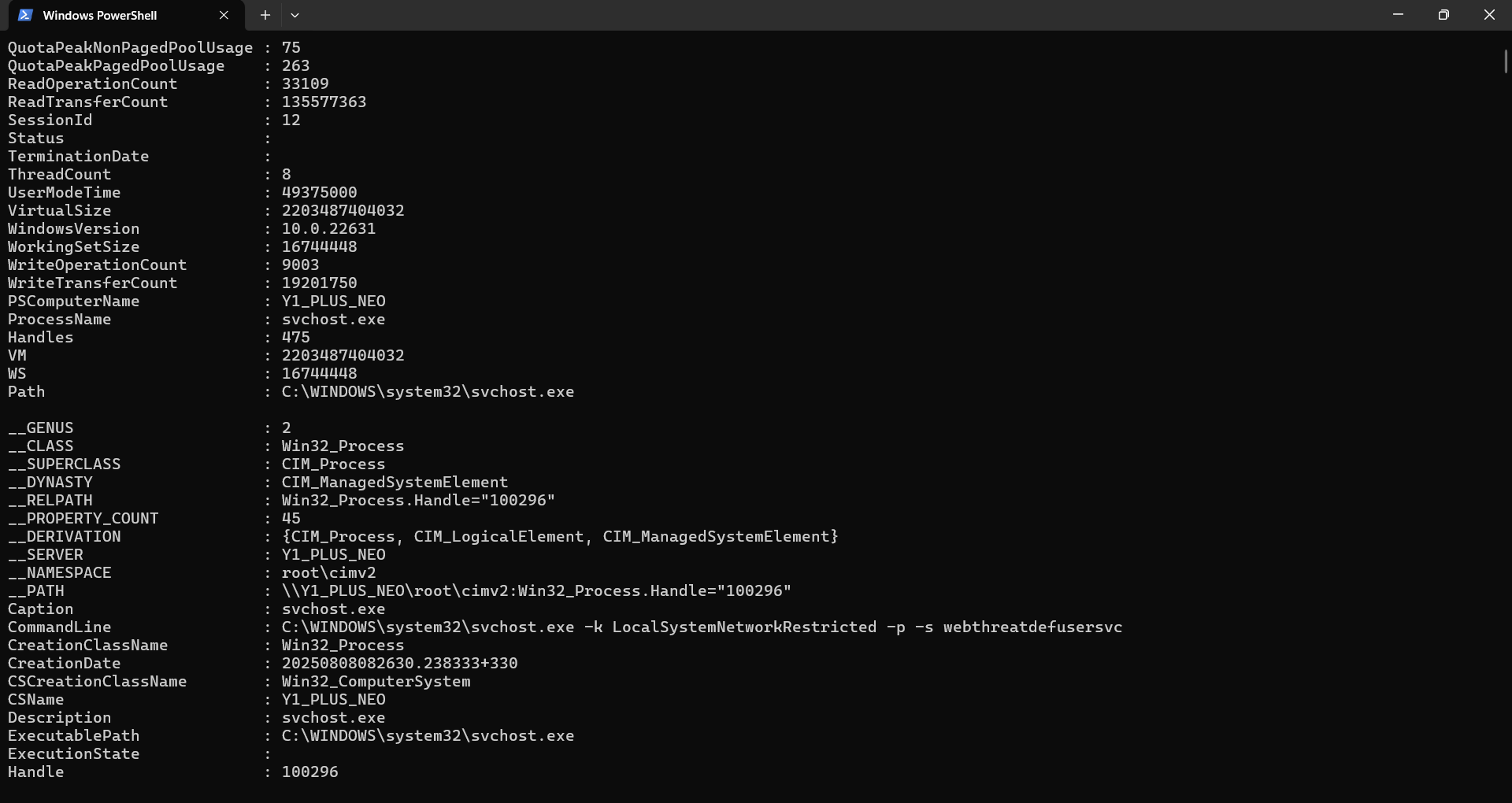
2. WMI Namespaces: WMI namespaces organize WMI classes and instances.

**PowerShell WMI Cmdlets:**

1. Get-WmiObject: Retrieves WMI objects.

2. Invoke-WmiMethod: Invokes WMI methods.

3. Set-WmiInstance: Modifies WMI instances.



**Pipeline Filtering & Operators:**

**Pipeline Filtering:**

1. Where-Object: Filters objects based on specific conditions.

2. Select-Object: Selects specific properties or objects.

Examples:

1. Filter processes by name:

Get-Process | Where-Object {$\_.ProcessName -eq "explorer"}

2. Select specific properties:

Get-Process | Select-Object -Property ProcessName, Id

**Operators:**

1. Comparison operators:

- -eq (equal)

- -ne (not equal)

- -gt (greater than)

- -lt (less than)

- -ge (greater than or equal)

- -le (less than or equal)

2. Logical operators:

- -and (logical and)

- -or (logical or)

- -not (logical not)

3. String operators:

- -like (wildcard match)

- -match (regular expression match)

**Input, Output & Formatting:**

**Input:**

1. Read-Host: Reads input from the console.

2. $input: Automatic variable containing input from the pipeline.

3. Param(): Declares parameters for scripts and functions.

**Output:**

1. Write-Host: Writes output to the console.

2. Write-Output: Writes output to the pipeline.

3. Return: Returns output from a function.

**Formatting:**

1. Format-Table: Formats output as a table.

2. Format-List: Formats output as a list.

3. Format-Wide: Formats output in a wide table.

**Scripting Overview:**

**Scripting Basics:**

1. Scripts: Text files containing PowerShell code, saved with a .ps1 extension.

2. Functions: Reusable blocks of code that perform specific tasks.

3. Variables: Store and manipulate data in scripts.

**Scripting Benefits:**

1. Automation: Automate repetitive tasks, reducing manual effort.

2. Efficiency: Streamline processes, improving productivity.

3. Consistency: Ensure consistent results, reducing errors.

**Scripting Best Practices:**

1. Use meaningful variable names: Descriptive names make code easier to understand.

2. Comment code: Comments explain code purpose and functionality.

3. Error handling: Handle errors gracefully, ensuring script reliability.

4. Test scripts: Thoroughly test scripts before deployme