# CIS 218 Lab 07: Text data, mini-project (70 points)

Please leave the lab questions/instructions/rubrics/etc. in place. Just paste your screenshots and code below my instructions.

Purpose/knowledge/skills: Working with live data in PowerShell only gets you so far. At some point, most administrators have stored information they re-used to automate administrative tasks. In this lab, you’ll have a chance to work with previously-stored information, as well as create a mini-project to flex your new PowerShell skills.

## Task 1: Use Get-Content on a complex input file (10 points)

Your network engineer has arranged computers in blocks of 32 addresses:

192.168.0.000 - 192.168.0.031  
 192.168.0.032 - .063  
 192.168.0.064 - .095  
 192.168.0.096 - .127  
 192.168.0.128 - .159  
 192.168.0.160 - .191  
 192.168.0.192 - .223  
 192.168.0.224 - .255

Create a command that will list one of:

1. The computer names in a given block of 32 addresses, OR  
 2. The users of computers in a given block of 32 addresses  
Which option you choose is up to you.

A .CSV file is included in the lab files for the week, with computer names, IP addresses, and user names for each computer.

Steps to complete the task:

1. Read the file into a PowerShell variable using Import-CSV.

2. Explore the structure of the variable until you think you understand how the data is organized and accessed.

3. Pick a specific network address block to display. The .000-.031 block is one good option.

4. Create a command that will list the computer names or user names only in that block.

5. Save the command as a .ps1 file.

6. Fill out the PowerShell Documentation Template below the rubric. There are no parameters required for this task.

7. Document the command using a <# #> block.

9. When complete, capture a screenshot of a successful program run, with output showing either computer names or user names, but not both.

Rubric:

* Screenshot of the output created: 1 points
* PowerShell command copy-and-pasted: 1 points
* Correct program operation: 5 points
* PowerShell command documentation template: 2 points total, about 0.5 points per section

Please paste a screenshot of a successful program run **and** copy-and-paste the source code of your PowerShell command(s) here:

|  |  |
| --- | --- |
| PowerShell command documentation template | |
| Name |  |
| Synopsis |  |
| Description |  |
| Parameters |  |
| Examples |  |

## Task 2: PowerShell mini-project (60 points)

Create a PowerShell script to solve a problem. A list of sample topics is included below, but you are welcome to solve another problem so long as your project meets all requirements.

Your project must:

1. Get information to/from a file. This can be a text file, CSV file, XML file, or other.

2. Be implemented as a fully-documented script.

2.1. The PowerShell command documentation template is included below. Please use this to plan your project.

2.2. Include full documentation in a <# #> block. Points are awarded for clear and concise documentation. Use previous lab files and Microsoft documentation as a model.

2.3. Command names should be Verb-Noun and family-friendly.

2.4. Commands should have at least one parameter. Please contact your instructor if you can’t think of a practical parameter for your command.

3. Do enough to be worth a mini-project. Examples follow. Your project would only need to do one of these, or something roughly comparable:

3.1. Split information out from a sub-field of an input file.

3.2. Process an XML file in a way we haven’t done in class

3.3. Convert output to a new, useful format – such as text to HTML

3.4. Do calculations or limits on input data – selecting only entries with a property in a given range

4. Some of file input or output is required.  
 You may need to create sample input data for your project to work with. If so . . .

4.1. Sample data can be a .CSV file, a text file, or other source.

4.2. Sample data should include at least 10 objects/items.

4.3. It is good if sample data has several fields per item, but this is not a strict requirement.

4.4. You are welcome to reuse sample data from lab files, but most samples will need to be expanded for the project.

Sample project topics:

A. Given a list of computer names with IP addresses and user lists, find the addresses of all computers where a particular user has an account.

B. Given a list of computer names with IP addresses and user lists, group the computer names by address ranges (such as blocks of 32).

C. Parameterize the PowerShell script from task 1 so the user can specify a block of computer addresses. (Specifying the starting address (such as “32”) is one good way to go. There are others.) Change the output in some way, such as creating an alphabetical deduplicated list of users, or listing computer names with only the last three digits of the IP address.

D. Given a list of computer names and programs installed, find all computers that have a specific program installed.

E. On the currently-running computer, get a list of all drives. Report drives with more than/less than some percent of free space left. For example: Report-DrivePercent 5 will show all drives with less than 5% of free space left in one list, and more than 5% free space left in another list. Your textbook can help with initial coding here.

F. Find all installed products using WMI and with Win32\_Product class. List all products created by a specific company. Your textbook can help with the initial coding here: see review labs in the appendix.

G. List all folders, recursively, under C:\Users\username for a given user name on the current computer. On SPSCC lab computers, this will only work as expected if you use your own user name.

H. Other ideas are welcome, but must be approved by the professor by Wednesday of project week.

Steps to complete the task:

1. Identify the problem to solve. It should be complex enough to require a three-stage PowerShell pipeline with at least one parameter.

2. Identify whether data can come in from a file to start, or should be written to a file after completion.

3. Document how you *expect* to code the project using the command documentation template below the rubric.

4. Do minor exploratory coding to think about the problem – how is this likely to work? What pieces do you need? Use get-help to find more information on topics as needed.

5. Update the documentation from step 3 if needed.

6. Code your project in PowerShell using a pipeline or command sequence.

7. Update the documentation from step 3 if needed.

8. Add <# #> documentation to your PowerShell pipeline/command sequence.

9. Run the project, observing correct output.

10. Capture the project for submission:

10.1. Your PowerShell command documentation template should already be complete.

10.2. Take a screenshot showing a successful program run.

10.3. In the spaces indicated below the rubric, copy/paste the code from:

10.3.1. Your .PS1 file

10.3.2. The text from your input and/or output file(s), whichever your project uses.

Rubric:

Submission pieces: 15 points

* PowerShell command documentation template: 10 points total, about 2 points per section
* Screenshot of successful program run: 2 points
* PowerShell command copy-and-pasted: 2 points
* Input/output file(s) copy-and-pasted: 1 points

Design pieces: 45 points

* PowerShell (.ps1) file source code included: 0 points, but required to earn any design points
* Correct program operation: 20 points
* Program sophistication (“does enough”): 10 points
* Command has a parameter: 5 points
* File input or output: 5 points
* <# #> documentation clear and concise, roughly matches documentation template: 5 points

Submission 2.1: Please paste a screenshot of a successful program run **and** copy-and-paste the source code of your PowerShell command(s) here:

Submission 2.2: PowerShell command documentation template

|  |  |
| --- | --- |
| PowerShell command documentation template | |
| Name |  |
| Synopsis |  |
| Description |  |
| Parameters |  |
| Examples |  |

Submission 2.3: Please copy-and-paste the text of your PowerShell .ps1 script file here:

Submission 2.4: Please copy-and-paste the text of your input and/or output files here: