# Scripting Project 3 – Due Tuesday 5/9 11:59 pm CST

THIS IS AN EXAM. Any collaboration between students will result in academic integrity charges for all participating students. You may only use the code that I've covered in class to date and your code structure must match mine. You may not seek any outside help. The sample output video is the definitive word on what to do. Your script must run like mine and you must follow my logic. Remember that your script must run to be graded, so it's better to fully finish a function rather than halfway complete them all. You have done everything required for this project in previous tutorials and homework exercises.

Late projects will not be graded. Any script with a modification date later than the due date and time will be considered late.

There is enough flexibility in these instructions that no two scripts should be identical. Using commands not taught in this class will result in penalties. Code that is wildly different than what has been introduced in class will be assumed to be a product of AI generation and will receive no credit.

Cleanly comment your code and use white space to break up blocks of code into logical, commented sections.

Download aggregate_complaints_001.zip and aggregate_complaints_002.zip from the sftp server. I found it handy to make a copy of these two files in a file on my desktop. That allowed me to replace them on the desktop as needed.

CRITICAL: On Wednesday, 5/10 you must download the file aggregate_complaints_003.zip from the sftp server. Your desktop should have all three complaint zip folders along with your script clearly named "Project3.sh". Your complaints folder must be on the desktop. Failure to have this exact setup by 5/10 will result in a 10% grade penalty.

We are translating my Python Project 1 solution into PowerShell. As you look at the solution for Project 1 keep in mind that we are not doing a line by line translation, but rather a process to process translation. I'll give you some top-down guidance here:

Declarations

Declare all of your paths as static variables. Be sure to put the paths in quotes. You don't need to double backslash. `$ZIP_ARCHIVES = "Complaints\Zip Archives"`

Move to the desktop

Be sure to use $env:Username and set the location to the desktop

Setup Function

- If the complaints directory exists, delete it.
  ```
  Remove-Item $MAIN_DIRECTORY -Recurse
  ```
- Build the necessary directories using New-Item. Ensure any messages are not displayed.
- Set up the master file with the appropriate header

Unpacking Function

- Declare any necessary arrays
- Add * to your normal path before using Get-ChildItem
- Load file names in a directory to the array
  ```
  $somearray = Get-ChildItem -path $check_path -include aggregate_complaints*.zip
  ```
- Extract each zip file contents and move zip files to archives
  - Use -Force when unzipping with Expand-Archive
- Notify the user of a wait
- Populate arrays of files names in the processing and the file archives
  - Remember the issue with Get-ChildItem
- Iterate through your array of files to process
  - Iterate through the lines in each file using Get-Content
  - Skip lines with a length <= 2
  - Use split, trim, replace to make variables for the csv file
  - Use Out-File to write the data to the csv file
    - Remember the encoding
- Move the processed files to the file archives

Cleanup Function

- Declare any arrays
- Iterate through the csv file to make an array of the lines from the file
  - Skip the header
- Get a count of records before deleting duplicates
- Make the array unique
- Get a count of records after deleting duplicates
- Report on the number of current records
- Report on the number of records after removing duplicates

```
    Report on the number of duplicates removed.
    Replace the csv file.  Write a new header and then the non-duplicate data.

Report Function
    Make any array declarations
    Load an array of all the csv file records, skipping the header
    Iterate through that array to make an array of products
    In a while true loop:
        Declare variables and arrays
        Write “available products” header to the screen
        Remove duplicate products from the product array
        Write the products, preceded with a number starting at 1
            Don’t use foreach
        Get the user number choice
            Break if zero
            Error message if > the number of products shown
        Get the proper product based on the user entry
        Get the matching issues and companies
            See the Python script for flow
        Get a count of matching records
            Issue array length with duplicates
        Remove duplicates from the issue array
        Remove duplicates from the company array
        Show the report elements per the demo
Call the setup function
Menu
    Just like the Python script.
```

Remember, if you are writing an element of an array you have to enclose that in $()

Get-ChildItem

```
    # When using the -Include parameter, if you don't include an asterisk in the path
    # the command returns no output.
    Get-ChildItem -Path C:\Test\ -Include *.txt (THIS WILL NOT WORK)
    Get-ChildItem -Path C:\Test\* -Include *.txt (MUST BE LIKE THIS)
```

Third time’s a charm!  Good luck and have fun.  Thanks for taking my scripting class.