Data Processing in Shell notes

Note: examples shown from datacamp terminal due to the complexity of systems involved, I was not aware of the docker container or other options at the time of assembling this report

- **Downloading Data on the Command Line + Data Cleaning and "Munging" on the Command Line**
    - **Curl**
        - **"Option flags," before the URL**
            - **-C resumes file transfer in case of timeout**
            - **-O save file with original name**
            - **-o [name] specify file name**

```
1  # Download and rename the file in the same step
2  curl -o Spotify201812.zip -L https://assets.datacamp.com/production/repositories/4180/datasets/eb1d6a36fa3039e4e00064797e1a1600d267b135/
   201812SpotifyData.zip
```

```
Terminal  Slides
$ {
> # Download and rename the file in the same step
es/4180/datasets/eb1d6a36fa3039e4e00064797e1a1600d267b135/201812SpotifyData.zipie
>  :; }
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1944k  100 1944k    0     0  31.1M      0 --:--:-- --:--:-- --:--:-- 31.1M
$
```

- **"curl –help" or "man curl"**

```
1  curl --help
```

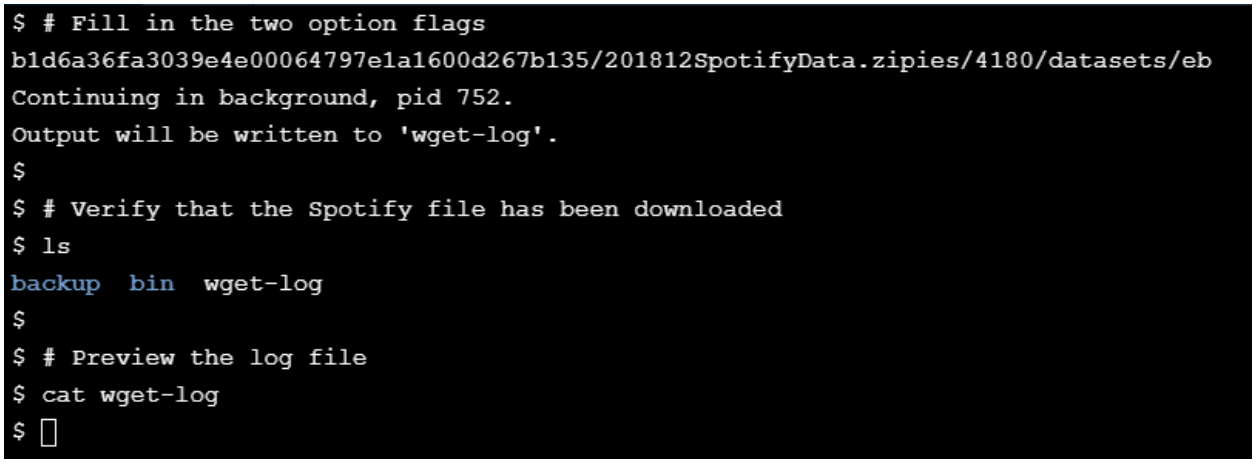
```
$ curl --help
Usage: curl [options...] <url>
     --abstract-unix-socket <path> Connect via abstract Unix domain socket
     --anyauth       Pick any authentication method
 -a, --append        Append to target file when uploading
     --basic         Use HTTP Basic Authentication
     --cacert <file> CA certificate to verify peer against
     --capath <dir>  CA directory to verify peer against
 -E, --cert <certificate[:password]> Client certificate file and password
```

- **Wget [option flags] [url]**
  - **-q: turn off output**
  - **-c:resume broken download**
  - **Option flags all together, "-bqc" for example**

```
# Fill in the two option flags
wget -c -b https://assets.datacamp.com/production/repositories/4180/datasets/eb1d6a36fa3039e4e00064797e1a1600d267b135/201812SpotifyData.zip

# Verify that the Spotify file has been downloaded
ls

# Preview the log file
cat wget-log
```



- Create wait times, “wget --wait=1 -i ‘filename’.txt,” to avoid overloading servers, this creates 1 second pause between each file

```
# View url_list.txt to verify content
cat url_list.txt

# Create a mandatory 1 second pause between downloading all files in url_list.txt
wget --wait=1 -i url_list.txt

# Take a look at all files downloaded
ls
```

```
--2022-05-05 23:02:36--  https://assets.datacamp.com/production/repositories/4180/datasets/
Reusing existing connection to assets.datacamp.com:443.
HTTP request sent, awaiting response... 200 OK
Length: 4293170 (4.1M) [text/csv]
Saving to: 'Spotify201809.csv'

Spotify201809.csv   100%[===================>]   4.09M  --.-KB/s    in 0.05s

2022-05-05 23:02:36 (79.4 MB/s) - 'Spotify201809.csv' saved [4293170/4293170]

FINISHED --2022-05-05 23:02:36--
Total wall clock time: 2.2s
Downloaded: 3 files, 13M in 0.1s (107 MB/s)
$
$ # Take a look at all files downloaded
$ ls
Spotify201809.csv  Spotify201811.csv  bin
Spotify201810.csv  backup             url_list.txt
$
```

- Csvkit
  - Installing csvkit with Python package manager “pip”

```
# Upgrade csvkit using pip
pip install --upgrade csvkit

# Print manual for in2csv
in2csv -h

# Print manual for csvlook
csvlook -h
```

- **In2csv –help, in2scv -h**

```
$ in2csv -h
usage: in2csv [-h] [-d DELIMITER] [-t] [-q QUOTECHAR] [-u {0,1,2,3}] [-b]
              [-p ESCAPECHAR] [-z FIELD_SIZE_LIMIT] [-e ENCODING] [-L LOCALE]
              [-S] [--blanks] [--date-format DATE_FORMAT]
              [--datetime-format DATETIME_FORMAT] [-H] [-K SKIP_LINES] [-v]
              [-l] [--zero] [-V] [-f FILETYPE] [-s SCHEMA] [-k KEY] [-n]
              [--sheet SHEET] [--write-sheets WRITE_SHEETS]
              [--encoding-xls ENCODING_XLS] [-y SNIFF_LIMIT] [-I]
              [FILE]

Convert common, but less awesome, tabular data formats to CSV.

positional arguments:
  FILE                  The CSV file to operate on. If omitted, will accept
                        input on STDIN.

optional arguments:
  -h, --help            show this help message and exit
```

- **in2csv > [file.xlsx] > [file.csv]**
  - **> directs output and saves as following argument**

```
# Convert SpotifyData.xlsx to csv
in2csv SpotifyData.xlsx > SpotifyData.csv
```

```
> # Convert SpotifyData.xlsx to csv
> in2csv SpotifyData.xlsx > SpotifyData.csv
```

- **Database Operations on the Command Line**
  - **When encountering errors, -v or -verbose for detailed logs**

- **Examples**
  - **sql2csv --db "sqlite:///SpotifyDatabase.db" **

    **--query "SELECT * FROM Spotify_Popularity LIMIT 5" **

```
# Verify database name
ls

# Save query to new file Spotify_Popularity_5Rows.csv
sql2csv --db "sqlite:///SpotifyDatabase.db" \
        --query "SELECT * FROM Spotify_Popularity LIMIT 5" \
        > Spotify_Popularity_5Rows.csv

# Verify newly created file
ls

# Print preview of newly created file
csvlook Spotify_Popularity_5Rows.csv
```

```
$ {
> # Verify database name
> ls
>
> # Save query to new file Spotify_Popularity_5Rows.csv
> sql2csv --db "sqlite:///SpotifyDatabase.db" \
>         --query "SELECT * FROM Spotify_Popularity LIMIT 5" \
>         > Spotify_Popularity_5Rows.csv
>
> # Verify newly created file
> ls
>
> # Print preview of newly created file
> csvlook Spotify_Popularity_5Rows.csv
>  :; }
SpotifyDatabase.db  backup  bin
SpotifyDatabase.db  Spotify_Popularity_5Rows.csv  backup  bin
| track_id               | popularity |
| ---------------------- | ---------- |
```

- **To reference a query stored as a shell variable, [--query "$sqlquery"]**
- **Upload a file to a database:**

  csvsql --db "sqlite:///SpotifyDatabase.db" --insert Spotify_MusicAttributes.csv

```
# Preview file
ls

# Upload Spotify_MusicAttributes.csv to database
csvsql --db "sqlite:///SpotifyDatabase.db" --insert Spotify_MusicAttributes.csv

# Store SQL query as shell variable
sqlquery="SELECT * FROM Spotify_MusicAttributes"

# Apply SQL query to re-pull new table in database
sql2csv --db "sqlite:///SpotifyDatabase.db" --query "$sqlquery"
```

```
$ {
> # Preview file
> ls
>
> # Upload Spotify_MusicAttributes.csv to database
> csvsql --___ "sqlite:///SpotifyDatabase.db" --___ Spotify_MusicAttributes.csv
>
> # Store SQL query as shell variable
> sqlquery="SELECT * FROM Spotify_MusicAttributes"
>
> # Apply SQL query to re-pull new table in database
> sql2csv --db "sqlite:///SpotifyDatabase.db" --query "$___"
>  :; }
Spotify_MusicAttributes.csv  backup  bin
usage: csvsql [-h] [-d DELIMITER] [-t] [-q QUOTECHAR] [-u {0,1,2,3}] [-b]
              [-p ESCAPECHAR] [-z FIELD_SIZE_LIMIT] [-e ENCODING] [-L LOCALE]
              [-S] [--blanks] [--date-format DATE_FORMAT]
              [--datetime-format DATETIME_FORMAT] [-H] [-K SKIP_LINES] [-v]
              [-l] [--zero] [-V]
```

- **Data Pipeline on the Command Line**
  - **-V or -version, prints python version**
  - **To start python on the terminal, just type python (if pathing is correct for installation)**

- **exit() - does what you would expect**
  - **print("hello world"), etc.**

- **Run a .py file by navigating to directory in command line, then "python *filename*.py"**

```
# in one step, create a new file and pass the print function into the file
echo "print('This is my first Python script')" > my_first_python_script.py

# check file location
ls

# check file content
cat my_first_python_script.py

# execute Python script file directly from command line
python my_first_python_script.py
```

```
$ {
> # in one step, create a new file and pass the print function into the file
> echo "print('This is my first Python script')" > my_first_python_script.py
>
> # check file location
> ls
>
> # check file content
> cat my_first_python_script.py
>
> # execute Python script file directly from command line
> python my_first_python_script.py
>  :; }
backup  bin  my_first_python_script.py
print('This is my first Python script')
This is my first Python script
$
```

- **Echo "(line that you want to add to .py file)" > "file destination"**

- **pip - package installation**
  - **pip –version**
  - **pip list**
    - **Displays all packages installed**

```
> # Install the required dependencies
> pip install -r requirements.txt
>
> # Verify that Scikit-Learn is now installed
> pip list
>  :; }
scikit-learn
Collecting scikit-learn
  Downloading scikit_learn-0.24.2-cp36-cp36m-manylinux2010_x86_64.whl (22.2 MB)
     |                                | 22.2 MB 18.7 MB/s
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-3.1.0-py3-none-any.whl (14 kB)
Collecting joblib>=0.11
  Downloading joblib-1.1.0-py2.py3-none-any.whl (306 kB)
     |                                | 306 kB 78.1 MB/s
Collecting scipy>=0.19.1
  Downloading scipy-1.5.4-cp36-cp36m-manylinux1_x86_64.whl (25.9 MB)
     |                                | 25.9 MB 30.4 MB/s
```

  - **To install non-current version of a library:**
    - **pip install [package name]==[version]**
    - **Example: pip install scikit-learn==0.19.2**
  - **Multiple packages**
    - **pip install [package1] [package2] etc.**

- **cron: a scheduler**
  - *** * * * ***
  - **Minute, hour, day of month, month(1-12, or jan, fab, mar, etc.), day of week(0-6, sunday 0 or 7) or sun,mon,tue,etc.**
  - **Leaving * blank will run it every increment of that level**

```
# Verify that there are no CRON jobs currently scheduled
crontab -l

# Create Python file hello_world.py
echo "print('hello world')" > hello_world.py

# Preview Python file
cat hello_world.py

# Add as job that runs every minute on the minute to crontab
echo "* * * * * python hello_world.py" | crontab

# Verify that the CRON job has been added
crontab -l
```

```
> # Verify that there are no CRON jobs currently scheduled
> crontab -l
>
> # Create Python file hello_world.py
> echo "print('hello world')" > hello_world.py
>
> # Preview Python file
> cat hello_world.py
>
> # Add as job that runs every minute on the minute to crontab
> echo "* * * * * python hello_world.py" | crontab
>
> # Verify that the CRON job has been added
> crontab -l
>  :; }
* * * * * python hello_world.py
print('hello world')
* * * * * python hello_world.py
$
```