Mariam Joan
August 10, 2019
Udacity Data Analyst Nanodegree
Project: Wrangling and Analyze Data

The wrangling portion of this project involved complexity because there were different sources of data we had to gather, which had its own processes, and then a second part which was to assess and clean the data requiring deep understanding pandas and its built in functions. The three sources we were gathering were coming in from different flat file formats including csv, tsv and separately an API although which we then saved to a txt file.

Udacity gave us as part of this project, an archive tsv file that was only available via an HTTP URL. This required the use of Python's request library which we called using a string named url, into requests.get(url). This response is then ready to read to a file which we use the io and os modules for to join our path with our new directory and file, writing our response.content to the file.

Gathering data from an API requires registering first to receive a key, secret and token. Working with Tweepy the Twitter wrapper API worked the same way. After registering with Tweepy, I received a customer key and secret, and then an access token and secret. We use these encrypted strings in our API call with Tweepy which included other parameters related to rate limits of API calls and when to be notified of the rate limit because we were working with a large amount of data.

To then store these tweets coming in we use the Python io module to create a txt file which will be written in JSON given our tweets are in JSON format. From there we run a try except block in this io call so that we try to use json.dump on each tweet to this file and then write a new line however, if a Tweepy API call gives us an error we retreive the Tweepy error, print out a message, and create a dictionary of all failed tweet attempts to keep track of tweets failed.

Next, we use io module again to then read the new txt file of tweets line by line using json.loads and append each line to a list. Now we have our tweets in a list which we convert to a pandas DataFrame. Lastly, we convert both csv, and tsv files to pandas DataFrames as well and our gathering part is done.

Assessing the data required at least 8 quality issues and 2 tidiness issues which follows the rules of tidy data. Some of these rules include, "each variable must have it's own column and each observation must have it's own row" (*Wickham, H., Journal of Statistical Software, Vol VV, Issue 2,* *https://vita.had.co.nz/papers/tidy-data.pdf, pg. 4*), etc. While our data from each source was fairly tidy there were some quality issues to note like, the symbol "_" between word names, grammatical issues like capitalization of names, for example.

Once we define our assessment of the data with our tidy rules, we can start cleaning. Cleaning involves creating a copy of each DataFrame to a new DataFrame so we can save our original data. Cleaning included using pandas built in functions such as drop, dropna, replace, shape, lambda, map, and astype. Once able to view all columns and rows in each DataFrame dropping the columns deemed unusable, we could then merge each DataFrame using pandas merge and reduce function using each dataset's tweet id as the join key.