Author: Alan Danque

Class: DSC 540

Week 12: Final Term Project

Purpose: Summarize all steps taken and reflect on any challenges encountered during this project.

For my project, since I have always been interested in how to find trends in stocks - I decided I would begin to reflect my back-burner passion onto this project. My hopes are to use this project after completing this master’s program to build a larger framework that would be helpful in later providing more insights on the relationships between different stocks and external pressures to be later identified and added. Below are my steps taken (including the names of the related python scripts created per each step) and the challenges I have encountered.

**Steps taken:**

1. **Web Page analysis and review:**

Being a customer of Fidelity, Van Guard and Morgan Stanley, I am one who is prone to access their stock data using my credentials. For this project, I really didn’t want to tie my personal credentials to my code. Beside some of it can be a little embarrassing. Meh, all kidding aside I really don’t have the time to keep a trained eye on the stocks selected by my younger self. Albeit, a year ago, week ago or even an hour ago. Outside the login required stock sites, I tried to use financial sites like finance.yahoo.com and finance.google.com. The only issue I had with those sites was there was not enough metrics available to collect. After reviewing a few other stock sites and finding either I needed to pay for access or not have enough metrics available, I was able to find a site that worked well for my needs. This site is finviz.com.

A screenshot of a computer

Description automatically generated

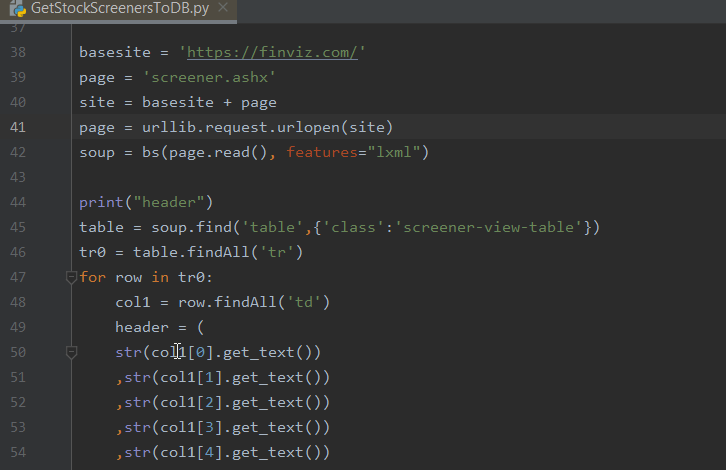
**The following two steps were coded into the script named, “GetStockScreenersToDB.py“.** The code will loop through all available ticker stock pages to collect a total of 7762 stocks in 42minutes. It builds generates a dataset that consists of 17 fields.

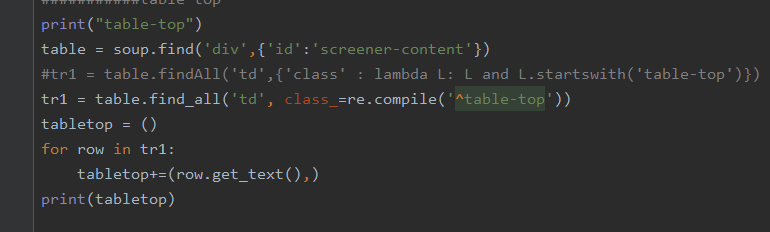
1. **Identify web scrape data retrieval strategy steps:**

To find the metrics I needed, I reviewed all the pages available. I was primarily looking for where the data was being delivered within the html tags of the sites. I was especially interested in looking for specific tags that were accompanied by unique or somewhat unique class or id names using the code inspector within Firefox. Once I was able to find a tag and class name or ID that presented the data I needed. I created the code needed to use the python library Beautiful soup to retrieve the values being delivered.

A screenshot of a computer

Description automatically generated



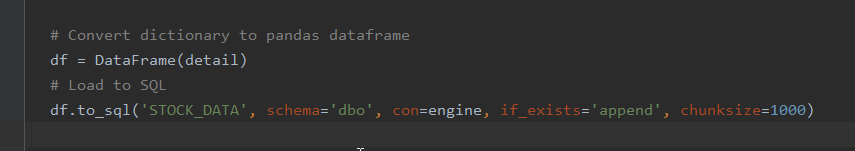


Above is the regular expression used to capture the class name that started with “table-top” as shown in the html inspector screen shot.

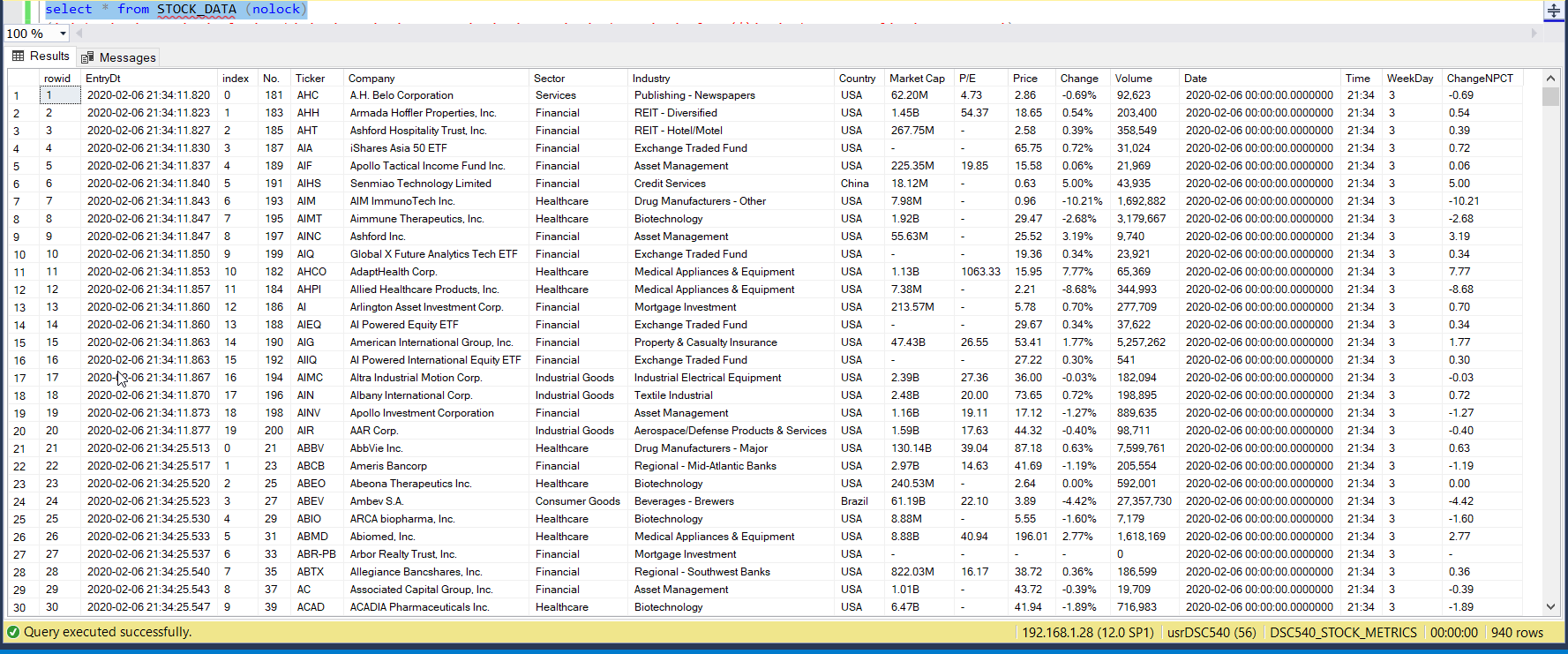
1. **Formatting data and ingestion steps:**

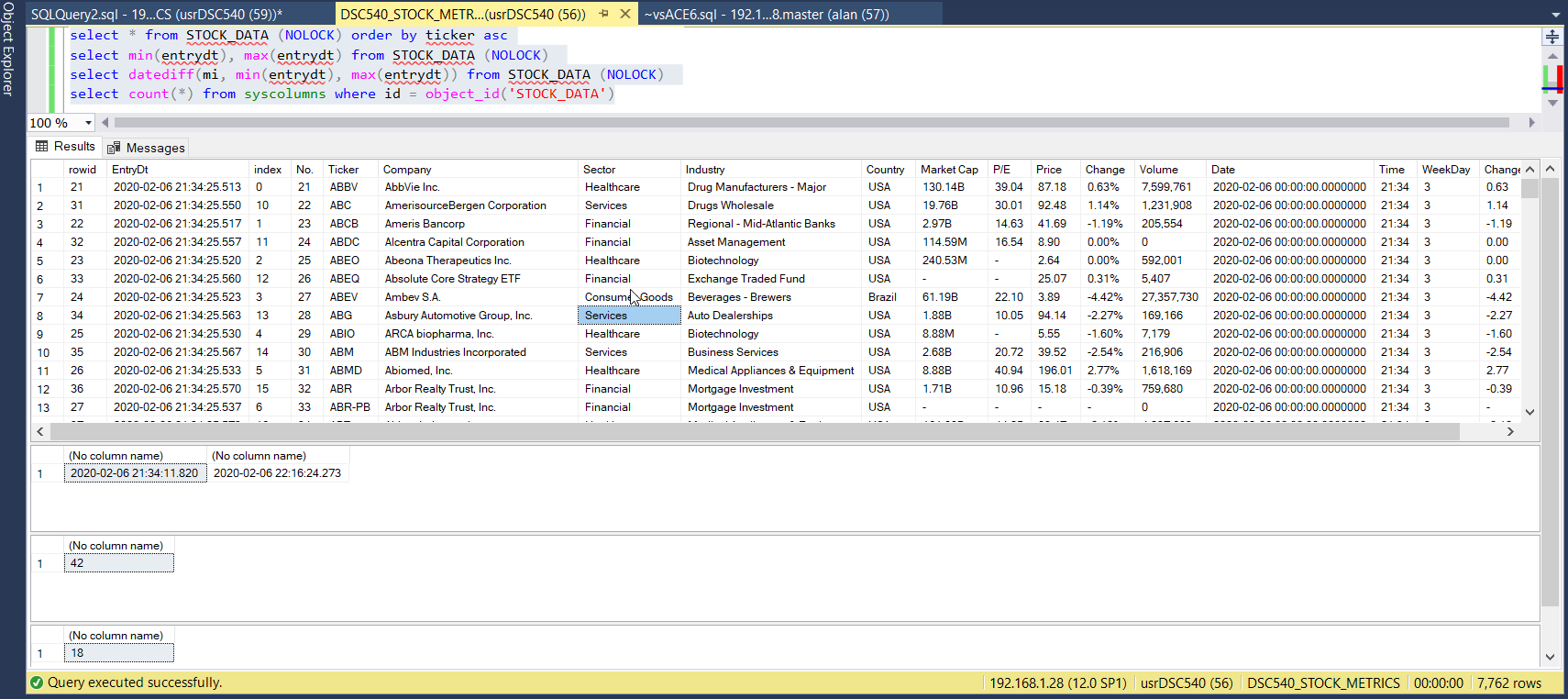
After creating the code needed to extract the headers and values, I needed using the BeautifulSoup and re libraries, I added the code to save the field headers to a tuple. Then for the detailed data row, I wrote a loop to reiteratively collected the rows of data into a tuple and then zipped them to combine with field header tuple to add to a dictionary. Then appended the dictionary to a list so to ensure the column fields are aligned. Lastly using the python libraries Pandas and SQLAlchemy, I converted my list of dictionaries into a Pandas dataframe and then used the to\_sql method to write the dataframe to a MSSQL Database table.





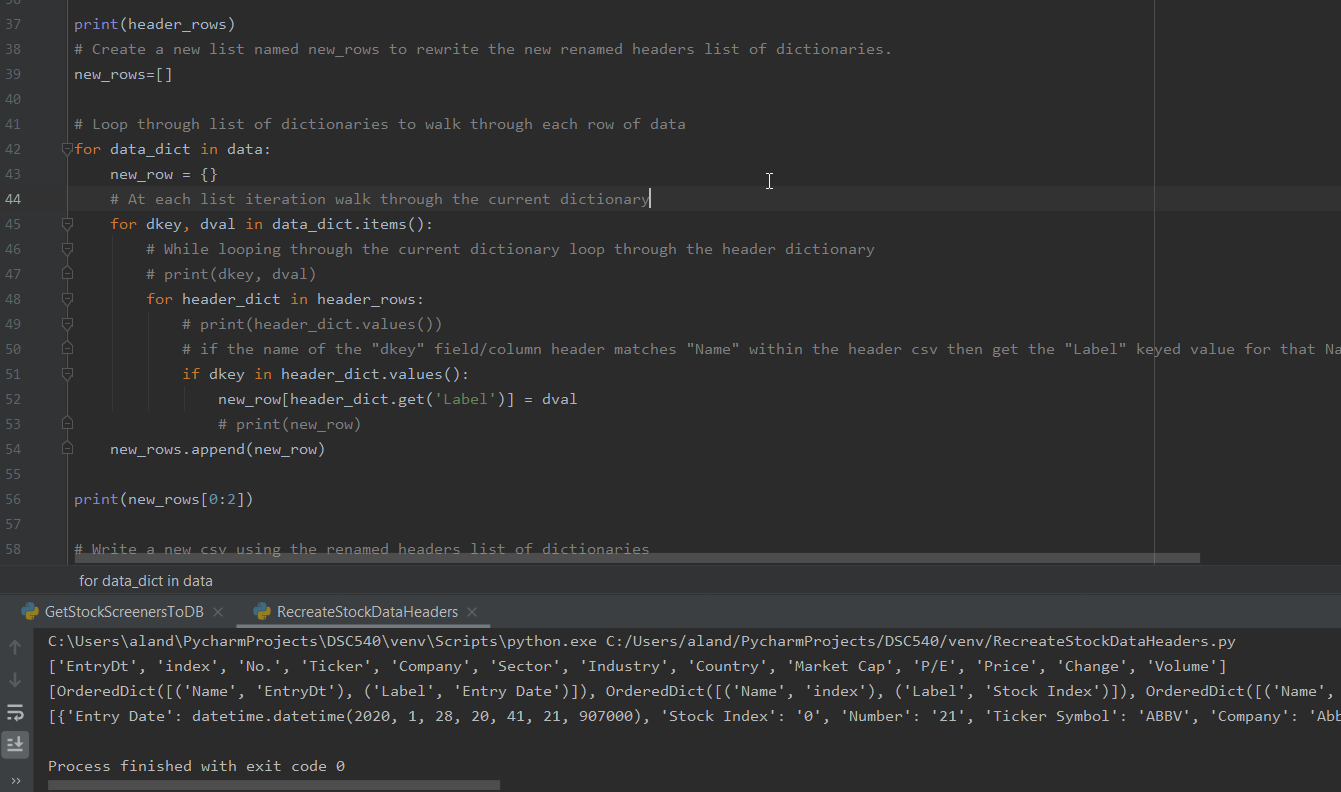
Screen shot of my dataset.

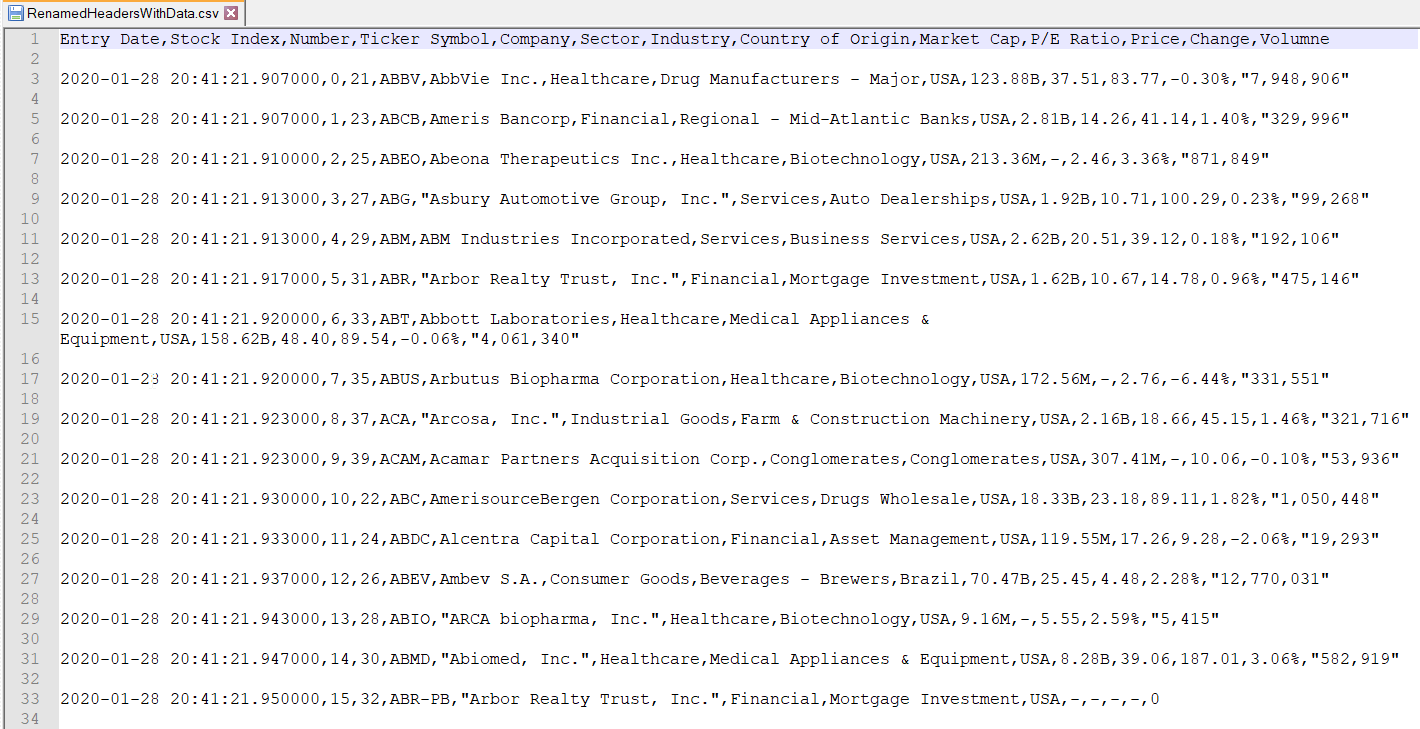




1. **Inventory, Test, Clean-up and Data Transformation Steps:**
   1. **ReviewStockData.py**
   2. **RecreateStockDataHeaders.py**

After ingesting the web scrapped data onto a MS SQL Server database table, using the python script named, “ReviewStockData.py” I inventoried the data types of each of the fields of ingested data, as well as inventoried the unique ticker values while counting how many duplicates there were. After reviewing the duplicate counts and data type counts, I reviewed the data to verify that the headers described the contents of the values they contained. Then using the python script named, “RecreateStockDataHeaders.py“ I replaced the headers to make them as intuitive as possible. Then I reviewed for Outliers and Bad Data that needed to be cleaned. After completing all of the steps above, I then imported the dataset back to a final relational structured MS SQL table.

****

****

**Challenges encountered.**

In retrospect of this first time of creating python code to scrape a website for helpful metrics, I found this project to be very fun and challenging. The things I found both fun and challenging is:

1. The week(s) long hunt for a site that has at least 15-20 metrics and did not require a pay for subscript or credentials. This was mostly because I believe I was picky however; I believe in this case the site selected worked out well for my data needs.
2. Reviewing and identifying the correct html tags and classes was slightly daunting as can see how each site can easily either purposefully or accidentally change a tag, I scripted to use to then break my code.
3. Identifying and analyzing the correct label names and the related data for my purpose of this data was a small challenge as I wanted to ensure the data was related and fit my needs.
4. Creating the necessary python code to loop through the web site and pages to scrape the data while adding them to a dictionary and then to zip with the header. This is per the regular expressions I used to find variations of unique named classes tied directly to the html tags that stored the data I sought after.
5. Reviewing, discovering, validating and cleansing the data contained in my dataset was slightly challenging as I wanted to make sure I added a good set of validation techniques that I can later reuse for more similar projects.