* Group Members:
  + Jorge Torres, Daniel Jo, Nell Meier, Joseph Orona, Elena Katicheva, Chandler Parrott
* Sources:
  + Yahoo Finance
  + Google News API
  + Robinhood Top 100 most Popular Stocks
* Goal Statement: In light of the recent media frenzy surrounding COVID-19 over the last month, not to mention the wild dip on Monday 3/9 for the stock market, we were interested to see what conclusions we could pull from the data for Stock Prices and News Articles related to the stocks. In short, does article count have any correlation to stock price?
* Limitations: The Google News API only allowed data extraction for articles in the last month (for free at least—we didn’t want to pay for this one.) We also used the top 100 most popular stocks, to create a limited scope on the project that we could still have a handle on.
* Extracting:
  + Using the Google News API, extracted news about the stock in review from 30 days prior to search. We used a pip install newsapi-python to pull the data/articles from the website and all signed up for an API key. From there, we were able to grab articles that had matched the parameters we inputted, to match anything related to the stock in review. After successfully pulling all and any articles pertaining to one stock, we created a loop to grab all 100 most popular stocks listed in Robinhood. To do so, we needed to extract the Top 100 most Popular Stocks, using Jupyter notebook. We took the data from the URL and converted it into a list, and then into a data frame. From there, we saved the data as a CSV for further use. After successfully saving the full stock data, we made a separate data table with only the symbol of the stock. We did so by dropping all the unnecessary columns using the .drop function in pandas. We then converted the data frame into a list. Using this list, we were able to use the API call earlier from Google News, that was only able to search for one stock, and loop the symbols into the search engine of the Google News API. Being able to loop through the list while using the Google News API, we only wanted the total results of articles found for each stock, so we created an empty list to house the new information we collected using the .append function. After collecting all data, we saved all information into CSV so that we didn’t have to continuously use the API call from Google News, which had a limit of 500 calls per day.
  + To extract stock information, we used the Yahoo Finance API, and set parameters for the same time period as the Google News API. To call the Yahoo Finance API we had to do a “pip yfinance install” in the Juypter notebook. After installing the yfinance, we used the repurposed list we created from the Robinhood data, and created a loop to call the Yahoo Finance API. Leveraging yahoo finance ticker function allowed us to create a data extract of the historical stock data. To reduce the use of limited API calls, we created data frames while looping into a master data frame, and saved it as a CSV. After extracting all data from Yahoo Finance, Google API, and Robinhood, we stored all the information in CSV files to later clean and transform.
* Transform and Load
  + During the transformation process, we reviewed all three CSV files that were made, and decided to take out unnecessary information for final comparison and review. The data that we decided to keep were: the closed dates of the beginning of the 30 days, the last date, the prices of the two dates, the stock name and symbol, the total amount of articles written about the stock in question, the total volume, and the number of visits that the stock had. We stored and cleaned the CSV/data using PostGRES. We created tables for each data sets and then later used a common table expression function in mySQL to pull certain data/values from a table and later joined other tables under the “Symbol” column. While pulling certain information, we did not need to remove columns but instead we had to pull the information into another final table. All tables were joined under one SQL statement. We then extracted the final table from PostGRES using SQL alchemy and pandas. After extracting the final table, we then had to rename the columns with their respective column headers then saved the table into a CSV file for further analysis.
* Results
  + Using the final table, we wanted to find any correlations between the percent change versus article counts and total volume. Because of the mistake in the data that was realized at the final hour, our descriptive statistics in the Descriptive Statistics Jupyter Notebook are incorrect, which is disappointing because it was the original goal of our experiment. Our original goal was to find out if the number of articles published was in any way tied to the stock price throughout the last month: the month that the Corona Virus (arguably) began to seriously affect the US Stock Market. An area that you can clearly see the mistake for the data (see below for more details on error) in the article count being wrong is the “article count” column in the third output. At 227,487, this is the “WORD” search that was called that wasn’t able to discern between word, the word, and “WORD” the ticker symbol for stock. To some surprise, there was a stock symbol that had a minimum of 0 articles posted in the month long time frame we were able to see data for. Although we weren’t able to find the correlation between stock price and article count, as we originally desired, we were able to learn more about the ETL process and how to be more efficient in future efforts.
* Problems we ran into
  + After creating a creating graphs to find any correlations, we noticed that the article count is not accurate. While running a loop function in python, we used the stock symbols as the search reference but came to realization that it also pulled articles pertaining to the letter or letters in reference. For an example, while searching the stock “WORD”, it not only searched WORD, but also the word “word.” Therefore, we did not use the article count for correlation purposes.
  + Another issue we found was that one of the stocks did not match the Yahoo finance due to a simple special character error. The stock that gave us trouble was the BRK.B stock. Yahoo has the stock listed as BRK-B. We did a simple rename to correct this error.
  + SQL
    - Created initial statement database post gres sql called project dos
    - We created a table
      * Article count, stock results
    - Using sql we constructed common table expressions
      * To iso two data point in our stock result data
        + Using the 1st day and second day of aricles
      * Sticted the data together
      * Created a third table on stack results for total volume and the activit of the two dates (SUM)
      * Join to article counts using the symbol and added the total volume using the symbol under one sql statement
      * Loaded into a new table
      * Then moved into python to extract from postgres
  + Extracting the juypter notebook
    - Stuck it into a data frame and saved it under a CSV file.
    - Using sql achlecmy and pandas we connected to postgres
    - Ran a sql statement on the final table