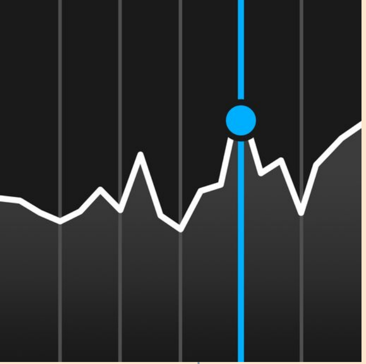


How Weather Affects the Stock Market

Presented by Aiyana Chopra, Anna Rauwerda and Tej Seth

April 21st, 2021

https://github.com/tejseth/SI-206-Final-Project

Agenda

Timeline and Goals: We set the stage for out project.

Visualizations: The data viz that we made.

Code Documentation: The functions that we made for this project.

Analysis: A summary of our findings from BeautifulSoup and API's.

Tables: Insight into the tables that we created in our database.

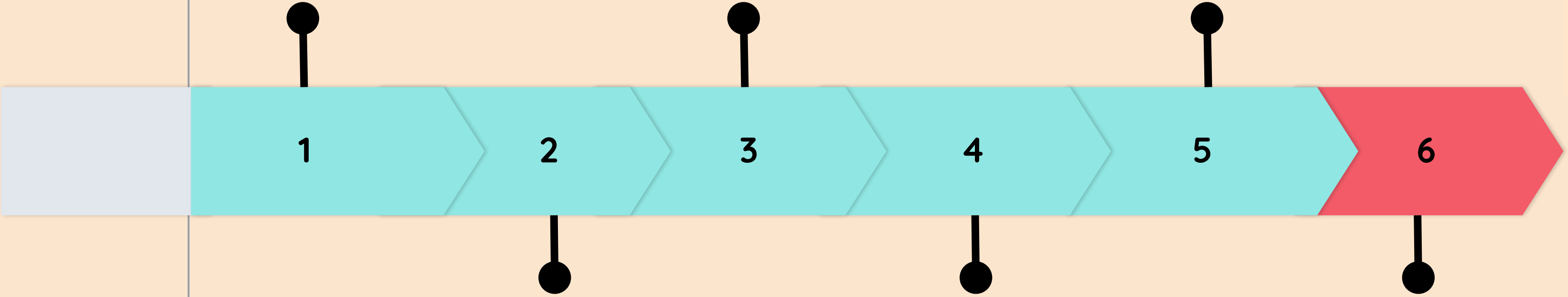
Resources and Conclusion: Wrapping things up.

Timeline

Picking our API's: We picked the MetaWeather and Stock Market API's

Methods I: We did our BeautifulSoup on the articles

Compile: We made our plots and put them in a common area.



Finding articles for Beautiful Soup: We looked in the New York Times and other locations

Methods II: We did SQL on the two API's and joined them.

Finalize: We took all of our findings and put together these slides.

Goals

- (Original Goal: See if there was a correlation between crime rates and weather in New York City with trying to find which crimes were most common with which types of weather).

- Use 2 API's and find at least 1 article for Beautiful Soup

- Make at least 2 visualizations from BeautifulSoup and at least 2 from the API's

Achieved Goals

- We pivoted away from the NYC Crime Data API because the API part of the website got removed and instead compared stock market data to weather trends successfully.

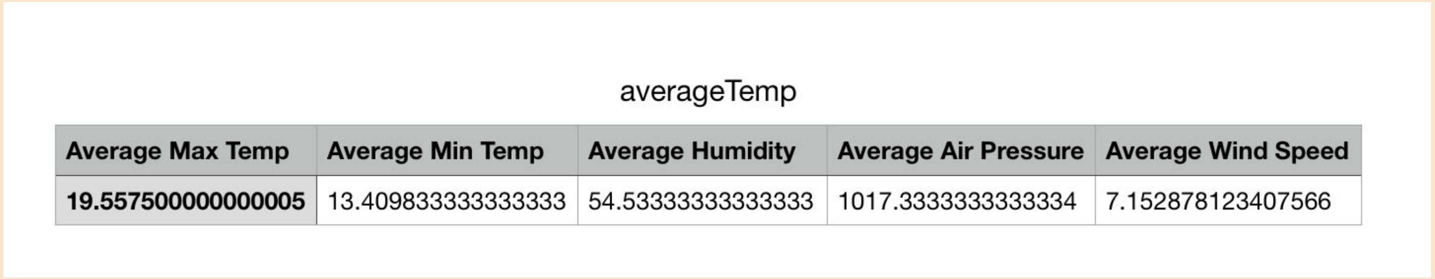
- We achieved our resources goal by using 2 API's and 3 articles for BeautifulSoup

- We achieved our visualization goal by creating 5 visualizations with 2 from the API's and 3 from the BeautifulSoup.

Problems

- The original crime API that we planned on using was no longer offering their API and was only offering CSV files instead so we had to pivot to another API.
- We tried making a word cloud with the text gathered from Beautiful Soup however ‘import WordCloud’ didn’t work
- While utilizing the MetaWeather API, the API key did not initially work and required setup time, in addition each API response took around a second - so loading data for 2 years proved to be time consuming as there was no bulk API operation
- When trying to combine the weather API and stock API data we faced challenges figuring out how to join the the two into one database

Calculations #1 - Weather Table

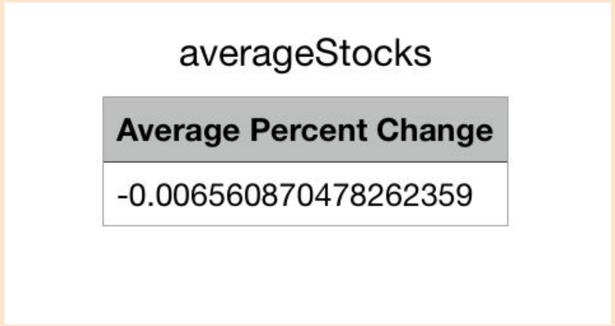


averageTemp

Average Max Temp	Average Min Temp	Average Humidity	Average Air Pressure	Average Wind Speed
19.557500000000005	13.409833333333333	54.53333333333333	1017.3333333333334	7.152878123407566

Data Calculated from MetaWeather API - May 2020 Averages

Calculations #2 - Stocks Table

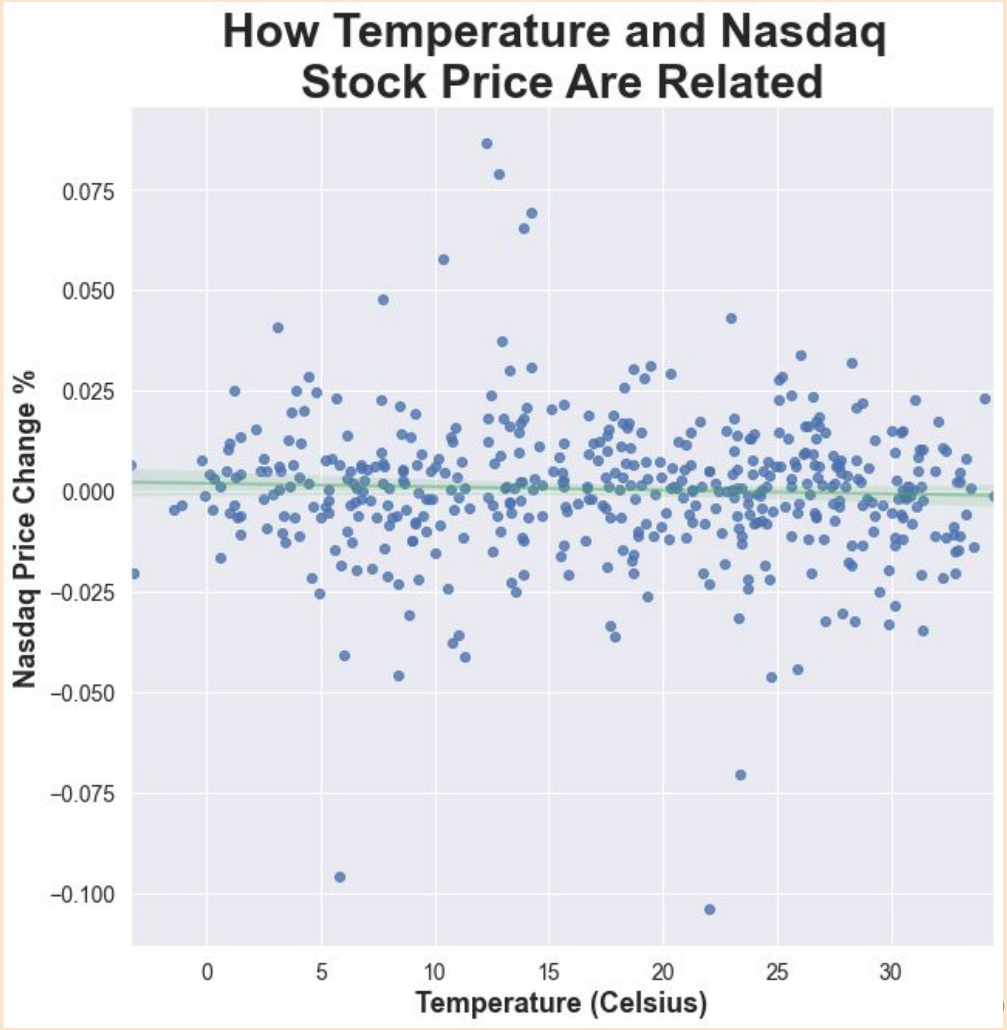

averageStocks

Average Percent Change
-0.006560870478262359

Data Calculated from Stocks API - May 2020 Average of Percent Change

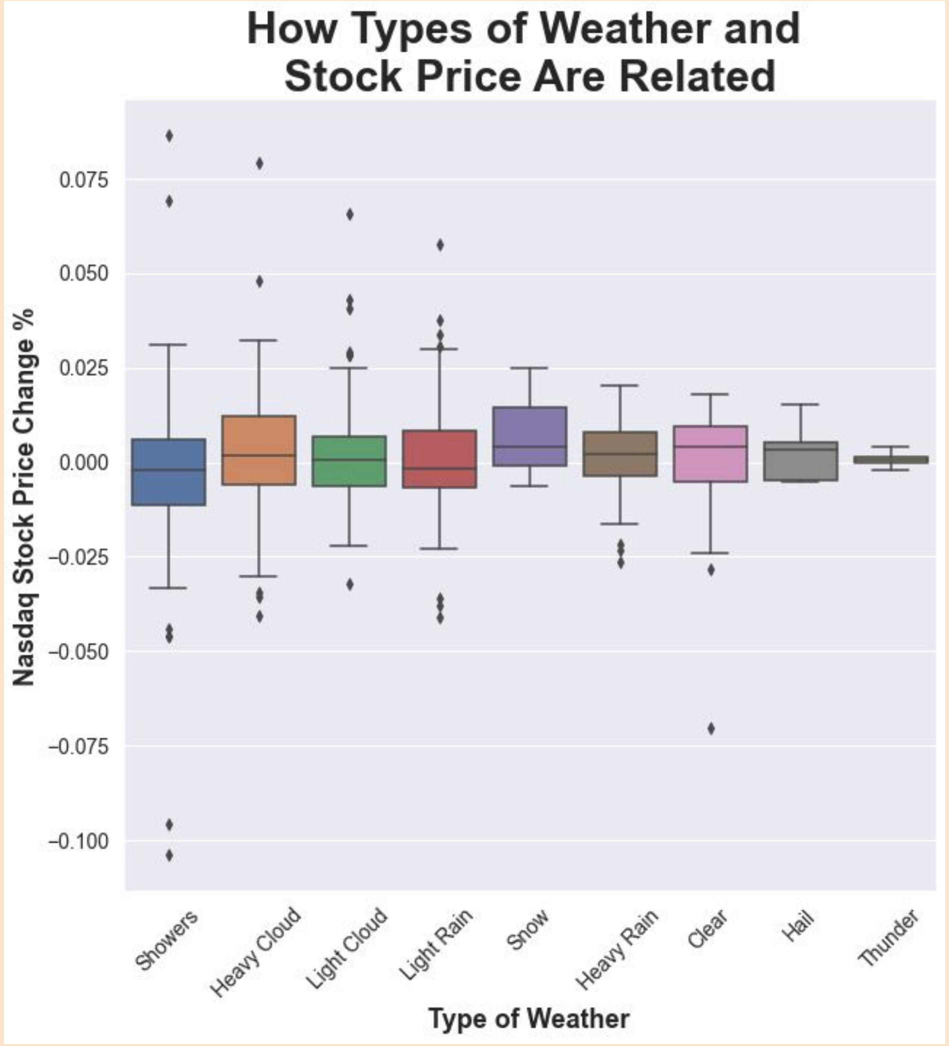
Visualization #1

- This graph illustrates how the Nasdaq price change percentage relates to temperature in degrees Celsius of New York City.



Visualization #2

- This visualization demonstrates how different types of weather impact the Nasdaq price change percentage.



What We Learned From API Analysis

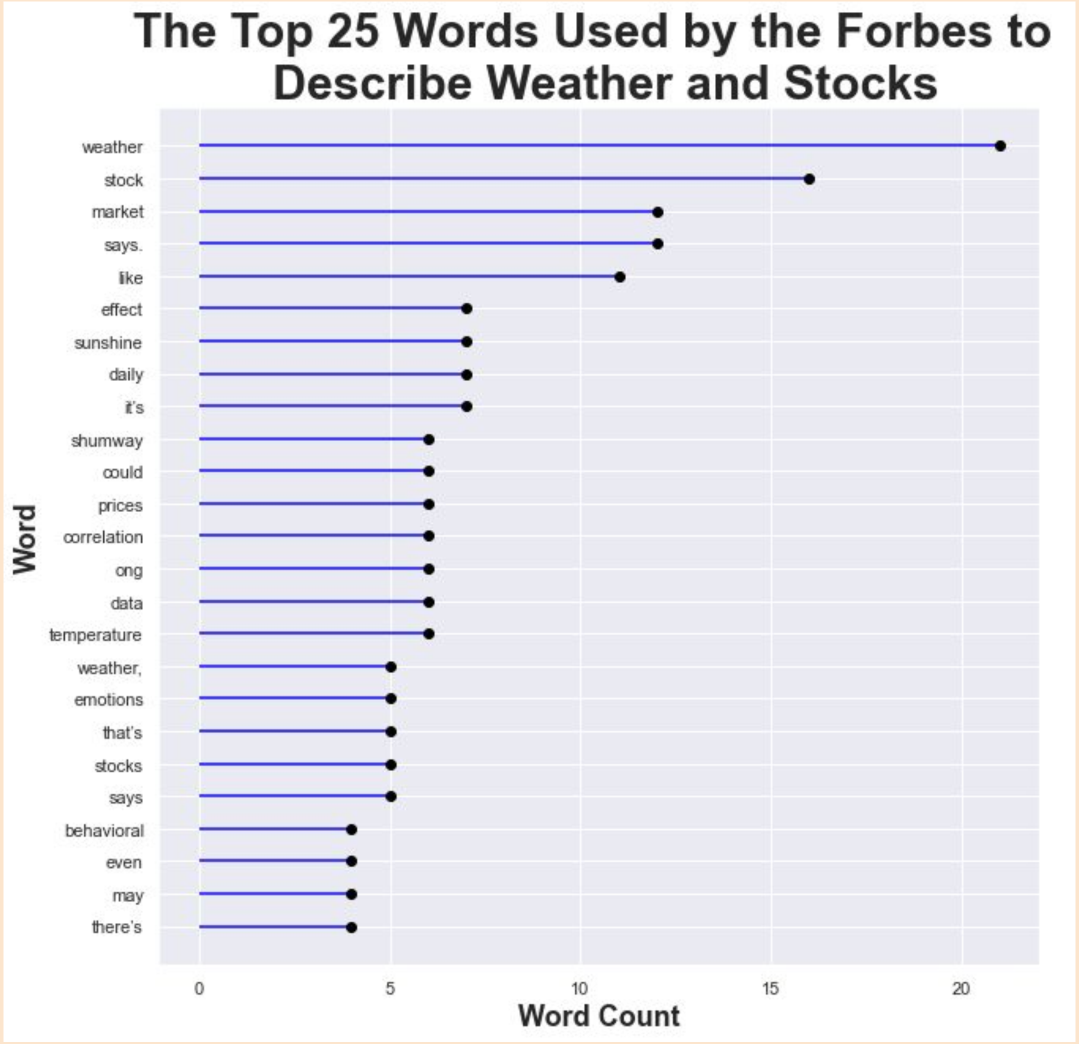
- There is no correlation between NYC temperature and Nasdaq stock price. Instead it is very noisy.

- The findings weren't significant but on days with snow or days with clear skies, there was usually a positive change to the Nasdaq.

- Heavy Clouds and Showers are the weather types that give the highest range of outcomes (variance) for stock price change.

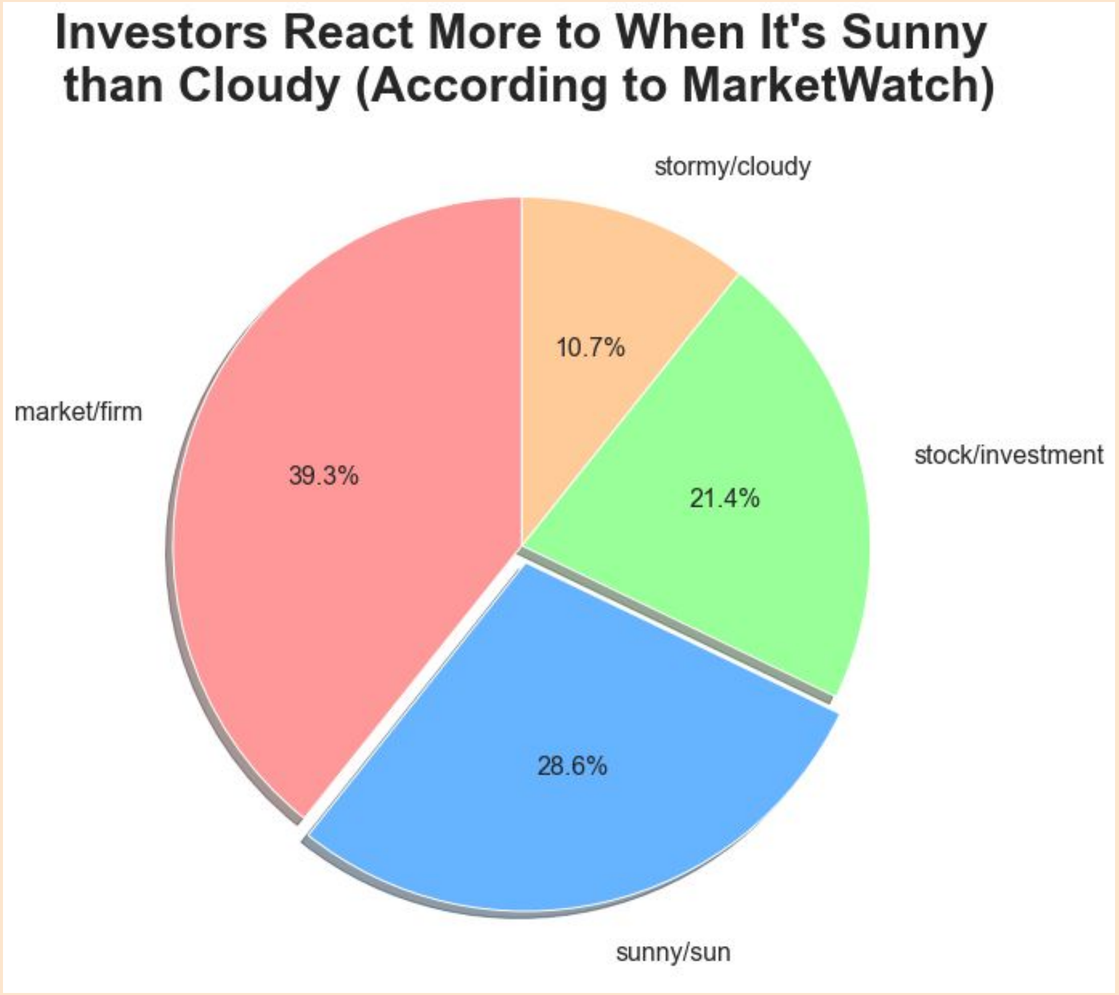
Visualization #3

- In their article, Forbes focuses on how daily weather affects the stock market, mostly how sunshine and temperature work.



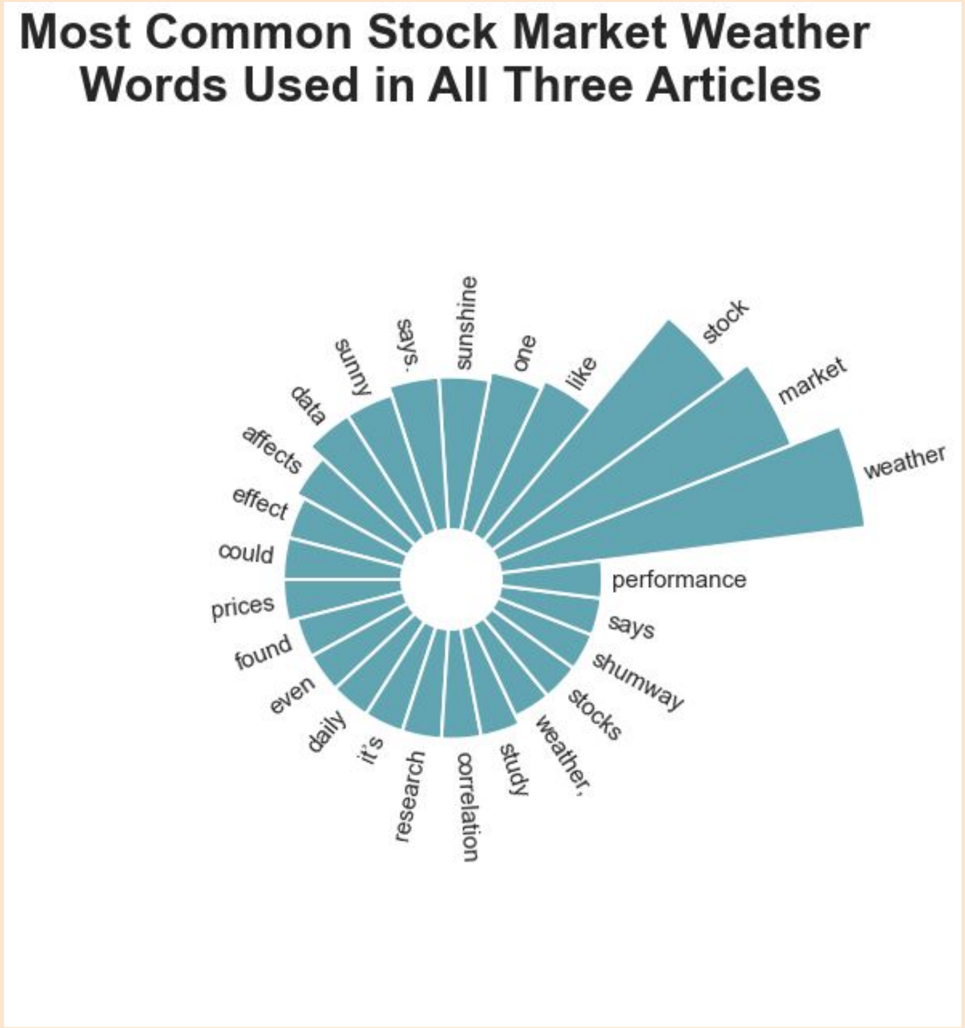
Visualization #4

- MarketWatch did a study on weather and the stock market and mentioned sun or sunny the most of any weather type.



Visualization #5

- When combining Forbes, Investopedia, and MarketWatch's articles together, we can see what words they mentioned the most often.



What We Learned From BeautifulSoup Analysis

- The articles focused on daily correlations between weather and stock prices so we decided to do the same.
- It was interesting that the articles would mention ‘sunny’ and ‘sun’ more often than ‘cloudy’ or ‘rainy’.
- The articles seemed to focus more on types of weather instead of measurable things like temperature.
- Shows how useful analysis like this is because ‘data’ and ‘correlation’ were frequently mentioned.

Instructions

1. Check to see if final_project.db is already not in your files. If it is in your files, delete it so that it can be created.
2. Make sure that the Polygon libraries are installed in your machine. Do this by typing "pip install polygon-api-client" into your terminal
3. The first file that should be run is "final_project.py". This file should be run once and the database will create two tables entitled "Weather" and "Stocks". Weather should be filled with 7 columns (Date, MaxTemp, MinTemp, Humidity, AirPressure, WindSpeed, and WeatherState). The Stocks table should be filled with 4 columns (date, percent_change, open, and stock). Both tables should consist of 731 rows.
4. The second file that should be run is 'stock_weather_scraper.py' which will do Beautiful Soup on two articles and make all the viz shown.
5. To see the calculations, you just need to run the final_project.py and then a CSV called "averageStocks.csv" and "averageTemp.csv" will be created. Open those and you will see the average calculations.
6. Open the database "final_project.db" to see the completed database which should consist of two tables.

Code Documentation #1: BeautifulSoup

```
def get_soup(url):
    '''In this function, a user inputs a URL of an article they want scraped and
        the function uses requests.get, BeautifulSoup and returns so that the use
        can use HTML to parse through the soup object'''

def get_filtered_dict(tags):
    '''This function takes in the tags that came from the url and cleans up the tags by
        just getting the text, putting it into a dictionary, removing common words from
        the dictionary and counting how many times it appears'''

def get_all_text(tags_list):
    '''This function takes in a list of tags that we have accumulated
        from the other articles read in so far and puts them all together
        to eventually create a combined text between articles to analyze'''
```

Code Documentation #2: final_project.py

def add_weather_to_db (locID, year, month, day, cur, conn):

""This function takes in the locID as an integer, year as an integer, month as an integer, day as an integer, cur, and conn and then makes a call to the MetaWeather API. It gets the variables maxtemp, mintemp, humidity, airpressure, windspeed, weatherstate, and date from the json data. Additionally, it inserts my variable columns into the Weather table in the SQL database. This function returns nothing ""

def setUpDatabase(db_name):

"'This function creates a database by taking in a database name, as a string, and then establishing a path and storing the path to the directory. It returns the cursor and connection to the database."

def getAverage(cur, conn):

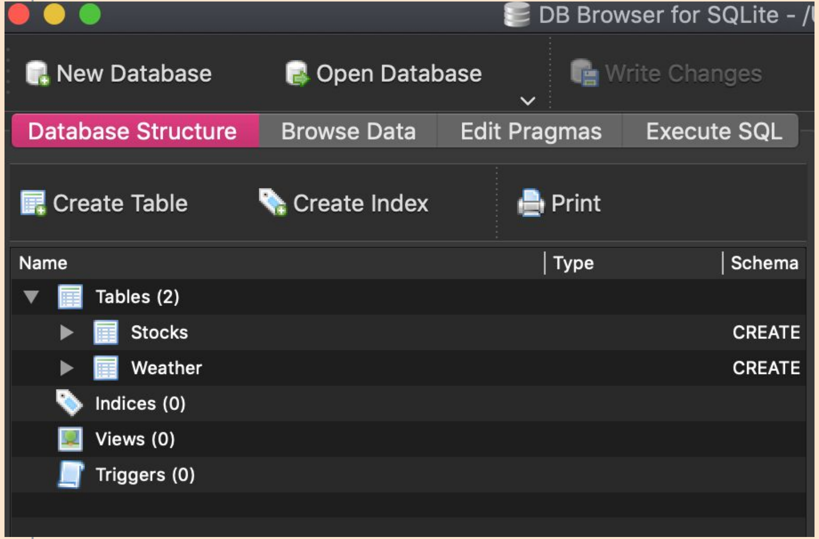
"'This function takes in cur and conn in order to find the average of each column in the weather table of the database. It creates a CSV that contains each variable and the month's average. This function returns nothing""

def getAverageStocks(cur, conn):

"'This function takes in cur and conn in order to find the average of each column in the stocks table of the database. It creates a CSV that contains the variable and the month's percent change average. This function returns nothing""

```
def get_stock_date():
    '''This function doesn’t take anything as an input but it puts a list of dates and returns it'''

def get_stock_data(stock, date):
    '''In this function a stock symbol and date is inputted, and it returns the percent change'''

def add_to_db(date, percent_change, open, stock, cur, conn):
   '''The add_to_db function inserts the data into the stock table'''

def add_weather_to_db(locID, year, month, day, cur, conn):
   '''In this function it uses INNER JOIN to put together the weather database and the stock market
database'''

def write(filename, join_list):
   '''This function takes in a filename join_list for the rows and writes headers and data-rows to a CSV'''

def main():
   '''Main picks a stock symbol like NDAQ and compares it on a day-by-day basis with weather'''

def join_db(cur,conn):
“””In this function it uses INNER JOIN to put together the weather database and the stock market
database. It returns the joined databases as a list'''
```

Weather and Stock tables are created



Name	Type	Schema
Tables (2)		
Stocks		CREATE
Weather		CREATE
Indices (0)		
Views (0)		
Triggers (0)		

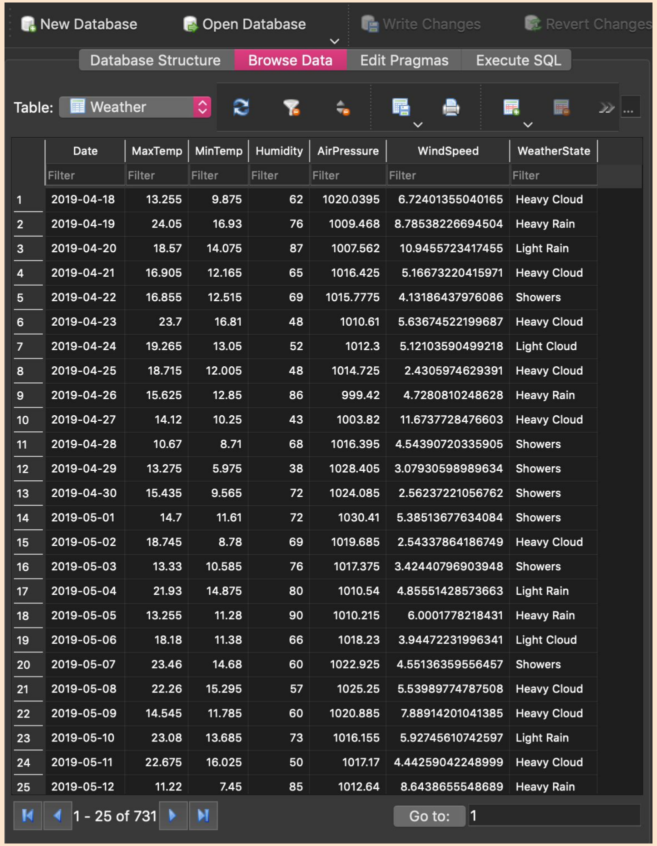


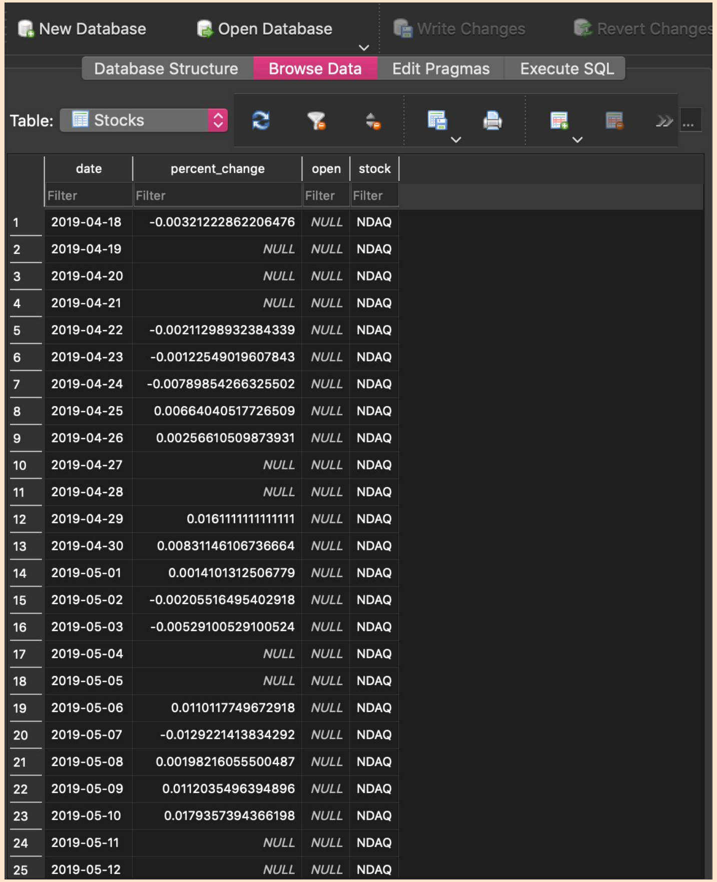
Table: Weather

	Date	MaxTemp	MinTemp	Humidity	AirPressure	WindSpeed	WeatherState
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	2019-04-18	13.255	9.875	62	1020.0395	6.72401355040165	Heavy Cloud
2	2019-04-19	24.05	16.93	76	1009.468	8.78538226694504	Heavy Rain
3	2019-04-20	18.57	14.075	87	1007.562	10.9455723417455	Light Rain
4	2019-04-21	16.905	12.165	65	1016.425	5.16673220415971	Heavy Cloud
5	2019-04-22	16.855	12.515	69	1015.7775	4.13186437976086	Showers
6	2019-04-23	23.7	16.81	48	1010.61	5.63674522199687	Heavy Cloud
7	2019-04-24	19.265	13.05	52	1012.3	5.12103590499218	Light Cloud
8	2019-04-25	18.715	12.005	48	1014.725	2.4305974629391	Heavy Cloud
9	2019-04-26	15.625	12.85	86	999.42	4.7280810248628	Heavy Rain
10	2019-04-27	14.12	10.25	43	1003.82	11.6737728476603	Heavy Cloud
11	2019-04-28	10.67	8.71	68	1016.395	4.54390720335905	Showers
12	2019-04-29	13.275	5.975	38	1028.405	3.07930598989634	Showers
13	2019-04-30	15.435	9.565	72	1024.085	2.56237221056762	Showers
14	2019-05-01	14.7	11.61	72	1030.41	5.38513677634084	Showers
15	2019-05-02	18.745	8.78	69	1019.685	2.54337864186749	Heavy Cloud
16	2019-05-03	13.33	10.585	76	1017.375	3.42440796903948	Showers
17	2019-05-04	21.93	14.875	80	1010.54	4.85551428573663	Light Rain
18	2019-05-05	13.255	11.28	90	1010.215	6.0001778218431	Heavy Rain
19	2019-05-06	18.18	11.38	66	1018.23	3.94472231996341	Light Cloud
20	2019-05-07	23.46	14.68	60	1022.925	4.55136359556457	Showers
21	2019-05-08	22.26	15.295	57	1025.25	5.53989774787508	Heavy Cloud
22	2019-05-09	14.545	11.785	60	1020.885	7.88914201041385	Heavy Cloud
23	2019-05-10	23.08	13.685	73	1016.155	5.92745610742597	Light Rain
24	2019-05-11	22.675	16.025	50	1017.17	4.44259042248999	Heavy Cloud
25	2019-05-12	11.22	7.45	85	1012.64	8.6438655548689	Heavy Rain

1 - 25 of 731

Go to: 1

Stock table created



	date	percent_change	open	stock
	Filter	Filter	Filter	Filter
1	2019-04-18	-0.00321222862206476	NULL	NDAQ
2	2019-04-19	NULL	NULL	NDAQ
3	2019-04-20	NULL	NULL	NDAQ
4	2019-04-21	NULL	NULL	NDAQ
5	2019-04-22	-0.00211298932384339	NULL	NDAQ
6	2019-04-23	-0.00122549019607843	NULL	NDAQ
7	2019-04-24	-0.00789854266325502	NULL	NDAQ
8	2019-04-25	0.00664040517726509	NULL	NDAQ
9	2019-04-26	0.00256610509873931	NULL	NDAQ
10	2019-04-27	NULL	NULL	NDAQ
11	2019-04-28	NULL	NULL	NDAQ
12	2019-04-29	0.0161111111111111	NULL	NDAQ
13	2019-04-30	0.00831146106736664	NULL	NDAQ
14	2019-05-01	0.0014101312506779	NULL	NDAQ
15	2019-05-02	-0.00205516495402918	NULL	NDAQ
16	2019-05-03	-0.00529100529100524	NULL	NDAQ
17	2019-05-04	NULL	NULL	NDAQ
18	2019-05-05	NULL	NULL	NDAQ
19	2019-05-06	0.0110117749672918	NULL	NDAQ
20	2019-05-07	-0.0129221413834292	NULL	NDAQ
21	2019-05-08	0.00198216055500487	NULL	NDAQ
22	2019-05-09	0.0112035496394896	NULL	NDAQ
23	2019-05-10	0.0179357394366198	NULL	NDAQ
24	2019-05-11	NULL	NULL	NDAQ
25	2019-05-12	NULL	NULL	NDAQ

Resources

Date	**Issue Description**	**Location of Resource**	**Result**
4.13.21	I needed to figure out a way to get bold titles on matplotlib plots so that they were easier to read	https://stackoverflow.com/questions/18962063/matplotlib-setting-title-bold-while-using-times-new-roman	i use plt.title(fontweight="bold") and was able to get bold titles!
4.14.21	Needed to find out how to get yesterday's date and format it within python in order to run MetaWeather API for past two years	https://stackoverflow.com/questions/1712116/formatting-yesterdays-date-in-python	Issue was resolved -- I imported from datetime import date, timedelta and was able to get yesterday's date
4.15.21	Had error "JSONDecodeError: Expecting value: line 1 column 1 (char 0)" and couldn't resolve it	https://stackoverflow.com/questions/16573332/jsondecodeerror-expecting-value-line-1-column-1-char-0	Issue was resolved -- added json.loads() and ensured I used JSON compatible character encoding

Resources #2

Date	Issue Description	Location of Resource	Result
4.15.21	TypeError: float() argument must be a string or a number, not 'method'	https://stackoverflow.com/questions/43579753/typeerror-float-argument-must-be-a-string-or-a-number-not-method	I had to use .as_matrix() on my dataset and it worked.
4.15.21	Not being able to import squarify to make waffle plots	https://stackoverflow.com/questions/41400136/how-to-do-waffle-charts-in-python-square-piechart	For some reason I just wasn't able to get that to work on my computer but I had other plots so it was okay.

Conclusion

- We found no significant relationship between the temperature of New York City and the Nasdaq price change.
- Clear and Snow slightly positively correlated to a positive percent change in the Nasdaq stock price.
- BeautifulSoup text analysis turned back that sunshine is very effective in stock price and the API's showed similar things because of the "Clear" weather type.
- Heavy Clouds and Showers had the highest variance of any weather type.

GitHub Repository: https://github.com/tejseth/SI-206-Final-Project