Final Project Report - SI 206

Team Members:
Michael DiSanto and Emily Edmunds

GitHub Repository:
https://github.com/mdisant/206_final

Project Goals:
The goal of this project is to create a program that integrates cryptocurrency price data with metrics from social media platform Twitter. Using the basic version of the CoinMarketCap API, we are able to gather the latest market ticker quotes and averages for cryptocurrencies and exchanges. This gives us access to variables such as current price, the percent change of price in the past 24 hours, and the percent change of price in the past 7 days. Through using the Twitter API (accessed through the Tweepy Python library), we gain the capability to look into metadata on Tweets mentioning various cryptocurrencies. This gives us the ability to compare the Twitter data with each cryptocurrency to its current price and price changes over different time periods. We also aim to determine the correlation between the price changes and number of Tweets to gain insight into how the price changes with respect to Twitter popularity.

Achieved Goals:
APIs/websites used: CoinMarketCap, Twitter (tweepy)

We were able to achieve all our goals using CoinMarketCap and the Twitter (tweepy) APIs. We used CoinMarketCap to gather data on 100 cryptocurrencies, including the cryptocurrency ID, name, price, percentage price change over 24 hours, and percentage price change over 7 days. From there, we searched for data points from Twitter to find the tweet count for the amount each of the 100 cryptocurrencies was mentioned in a tweet. We used this to make scatter plots comparing tweet count to percentage change over 24 hours and percentage change over 7 days, and created a bar chart showing the tweet count for each cryptocurrency. Additionally, we calculated the correlation between tweet count and percentage price increase.

Problems:
We faced a few problems when working towards our project goals. These were especially outlined in our initial project presentation, where we then had to go back and rewrite some of our functions to achieve proper functionality of our program. These problems included an inability to only retrieve 25 data entries at a time, finding duplicate data, and joining on text and not a unique ID number. These problems were fixed in the newest iteration of our program, and are further detailed below.

Updated Version Differences (after extra credit presentation):
No Combined/Duplicate Data/New Twtiter Table:
Fixed the problem where there was a combined table being created through the JOIN statement. Now, the Twitter table in the crypto database has a unique ID followed by the Tweet count.

Limiting Data to 25 Entries:
We are now able to limit the data to 25 entries and effectively run the code four times to populate the tables in the database with at least 100 entries. This is done by writing to and reading an auxiliary text file called run_number.txt with the current number of data entries in the table.

New Correlation Calculation/Text File:
The calculation to find the correlation between Tweet count and percent price change is now done by hand (instead of utilizing the built in Numpy function). These correlations are written out to the twitter.txt file instead of writing out the uncalculated price change data.

Fixed JOIN Statement:
Since there is no longer a combined table in the crypto database, the JOIN statements are used to get data joined on a specific, numerical ID. There are multiple JOIN statements, especially when creating the visualizations.

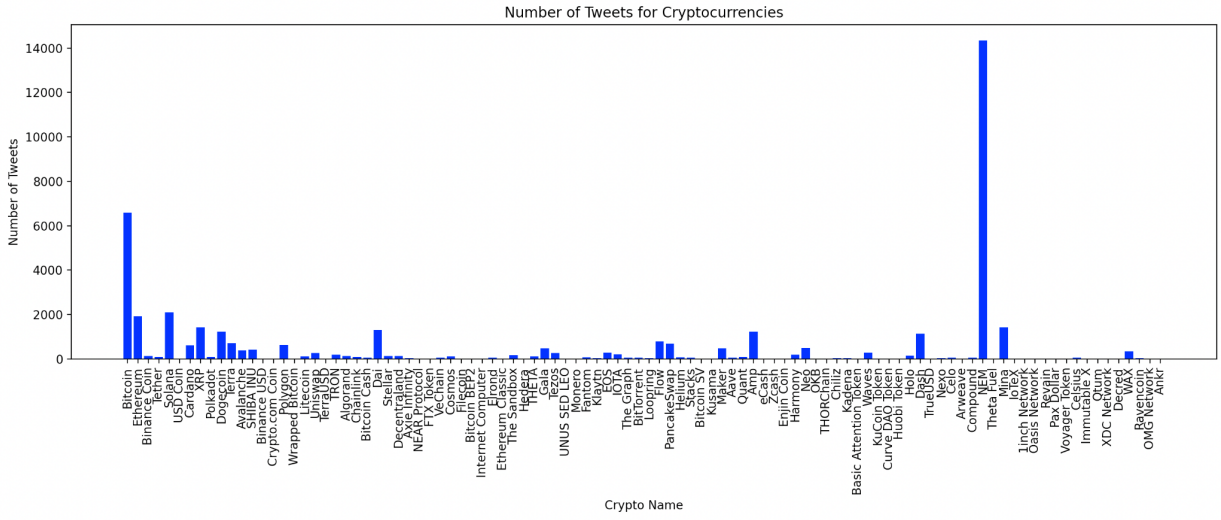
Data Calculations:

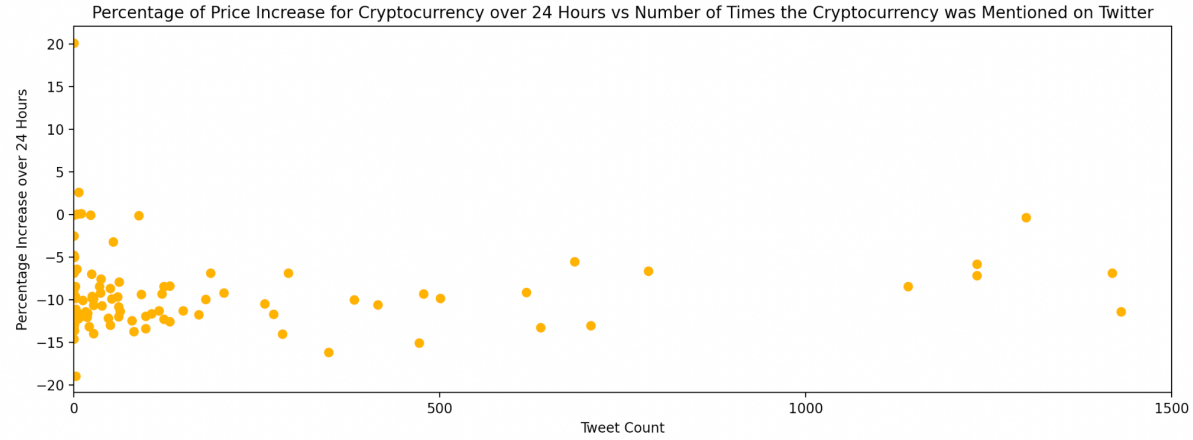
twitter.txt

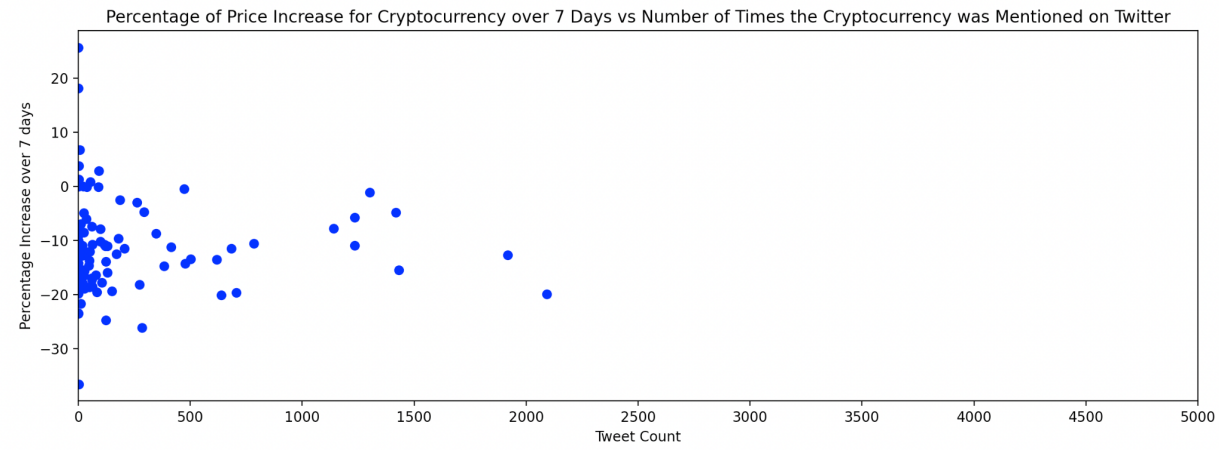
```
The calculated Pearson Correlation Coefficient between number of Tweets and percent change in the past 24 hours is:
0.01224

The calculated Pearson Correlation Coefficient between number of Tweets and percent change in the past 7 days is:
0.02494
```

Visualizations:







Instructions:
To run the github file with the codes, first, open the 206_final github link. Make sure that the "crypto.db" database does not already exist in your files. If it does, delete it so that it can be recreated. For the text files (twitter.txt and run_number) that are already in the 206_final folder, delete those text files to view the recreated text file that each python file generates after running. Since we are told to put the text file in the folder, we have left the text files in the folder to be seen.

Run the crypto.py file 4 times and each time refresh the "crypto.db" database to see the data being put in 25 times each run. This is the data collected from CoinMarketCap, creating a table called "Crypto" which stores the data for 100 cryptocurrencies including the cryptocurrency ID, name, price, percentage change over 24 hours, and percentage change over 7 days. We want to run crypto.py first to get the names for all of the cryptocurrencies we will be searching for on Twitter.

Next, run the twitter.py file 4 times and each time refresh the "crypto.db" database to see the data being put in 25 times each run. This is the data collected from the Twitter tweepy API, creating a table called Tweets, which stores the id and tweet count for the 100 cryptocurrencies in the Crypto table. This file will also create a text file, "twitter.txt," which will show a summary of the calculations run on the data in text format. These calculations are run in the main function through the calculate_correlations() on both "percent_change_7d" and "percent_change_24h".

Finally, run the visualizations.py file which will produce three visualizations.

If you get an “429: Too Many Requests” error you may need to wait a few hours to run the code again using the same API keys.

Function Documentation:

crypto.py:

Function	Input/Output
get_crypto_data(num)	Input: num (integer): the current number of data entries in crypto.db Output: Returns the crypto data dictionary in JSON format to be used to build the cryptocurrency table in the crypto database
setUpDatabase(db_name)	Input: db_name (string): name of the database Output: Returns the cursor and connection to the database
make_crypto_table(data_dict, cur, conn, num)	Input: data_dict (dictionary): the cryptocurrency data returned by the get_crypto_data function cur: cursor to database conn: connection to database Output: Returns nothing. Creates and populates the cryptocurrency table in the crypto database
read_file()	Input: Nothing Output: Returns the number of data entries in

	crypto.db as an integer using the run_number.txt auxiliary text file
write_file(num)	Input: num: the number of data entries in the cryptocurrency table in the crypto database Output: Returns nothing. Writes the number of data entries in crypto.db to the run_number.txt file
main()	Input: Nothing Output: Returns nothing. Reads the number of times that crypto.py has run, creates a cursor and connection to crypto.db, writes to a file the number of data entries in crypto.db, creates/adds to the cryptocurrency table in the crypto database, and closes the connection

twitter.py:

Function	**Input/Output**
create_twitter_tuple(cur)	Input: cur: cursor to database Output: Returns the Twitter data dictionary in JSON format to be used to build the Twitter table in the crypto database
setUpDatabase(db_name)	Input: db_name (string): name of the database Output: Returns the cursor and connection to the database
setUpTweetsTable(tuple_list, cur, conn)	Input: tuple_list: the Twitter data returned by the

	create_twitter_tuple function cur: cursor to database conn: connection to database Output: Returns nothing. Creates and populates the Twitter table in the crypto database
calculate_correlation(cur, timeline)	Input: cur: cursor to database timeline (string): the desired time period for finding the correlation. “percent_change_7d” or “percent_change_24h” Output: Returns the correlation (rounded to 5 decimal places) as a float between the timeline parameter and number of Tweets for all cryptocurrencies in the crypto table
write_out_data(fname, cur)	Input: fname: name of the filename to write out the calculations cur: cursor to database Output: Returns nothing. Writes the correlation between the two timelines to a file called twitter.txt
main()	Input: Nothing Output: Returns nothing. Creates a cursor and connection to crypto.db, creates/adds the Twitter data to the Twitter table in the crypto database, writes out the correlation calculations to the “twitter.txt” file, and closes the connection

visualizations.py:

Function	Input/Output
setUpDatabase(db_name)	Input: db_name (string): name of the database Output: Returns the cursor and connection to the database
bar_one(cur)	Input: cur: cursor to database Output: Returns nothing. Creates and shows a bar chart with the Crypto Name on the x-axis and Number of Tweets on the y-axis.
scatter_one(cur)	Input: cur: cursor to database Output: Returns nothing. Creates and shows a scatterplot with the Tweet Count on the x-axis and Percentage Increase over 24 hours on the y-axis.
scatter_two(cur)	Input: cur: cursor to database Output: Returns nothing. Creates and shows a scatterplot with the Tweet Count on the x-axis and Percentage Increase over 7 days on the y-axis.
main()	Input: Nothing Output: Returns nothing. Creates a cursor and connection to crypto.db, runs the three visualization functions (bar_one, scatter_one, and scatter_two), and closes the connection

Resources:

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
11/27/2021	Tweet text containing many emojis that can't be imported into the database	https://stackoverflow.com/questions/33404752/removing-emojis-from-a-string-in-python	Yes, but we ended up going a different route with the Tweets table (not using the text of tweets)
11/28/2021	Difficulty retrieving data from CoinMarketCap API. A lot of the endpoints require a paid version.	https://rapidapi.com/blog/how-to-use-the-coinmarketcap-api/	We were having trouble making the initial API call to the CoinMarketCap API, so we utilized this resource to get us started. It was very helpful as it provided a way to navigate through the API documentation.
11/29/2021	Receiving 429 error on Twitter data; we pushed too many requests	https://stackoverflow.com/questions/69028757/toomanyrequests-429-too-many-requests-while-running-tweepy	We couldn't make a premium account because it was costly, so we ended up just making a new developer account to reset the amount of tweet data we could collect monthly.
12/3/2021	We were getting an error: " UNIQUE contraists failed for Crypto.id"	https://stackoverflow.com/questions/35415469/sqlite3-unique-constraint-failed-error	This resource helped us understand why we were getting the error. To solve this, we wrote to a file each time the code was ran, which

			changed the start position of the API request, to ensure that we were getting unique data.
12/13/2021	Trying to get CoinMarketCap data inserted into the Crypto table 25 rows at a time	https://stackoverflow.com/questions/1041163/inserting-n-number-of-records-with-t-sql & https://stackoverflow.com/questions/42179757/dynamically-change-the-running-code-by-writing-into-the-file	Yes, we ultimately figured out that we would need to find a way to change the code between program runs. Wrote to a file each time the program ran to make this happen.