CBD-3335

Data Mining and Analysis

Collecting tweets related to the stock market.

By	To
Britty Bidari C0861113	Ali Nouhi
Sushant Giri C0861112	November 16, 2022
Jaspreet Kaur C0861116	
Vijay Seelam C0857329	

Objective

- **Gain experience collecting data from Twitter using Twitter API**
- **Gain experience in data storage to store the data and query**
- **Gain experience of collecting real-time data**
- **Gain experience of data cleaning**

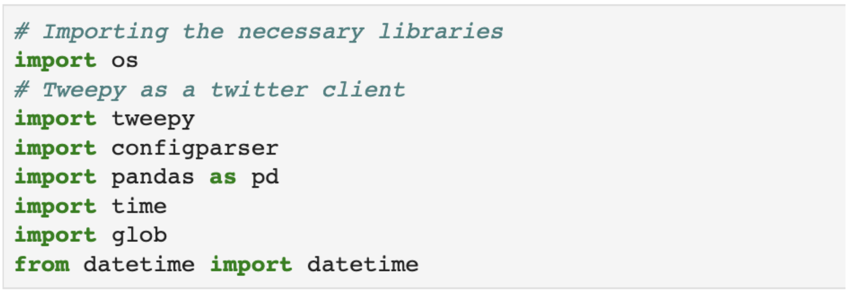
Introduction

Twitter is a social media platform with a vast user base. It is a platform that connects users globally and allows people to share thoughts and media over a variety of topics and among a vast audience. (Wikipedia)

Similarly, for this project, we have collected data from Twitter through Twitter API using tweepy. Further, it cleansed and successfully observed the data through graphical representation. (Tweepy)

Methodology

We started with importing different required machine learning libraries(Fig.1.) as per the requirement of our study. Then further, we followed various steps for performing our observation for the desired result.



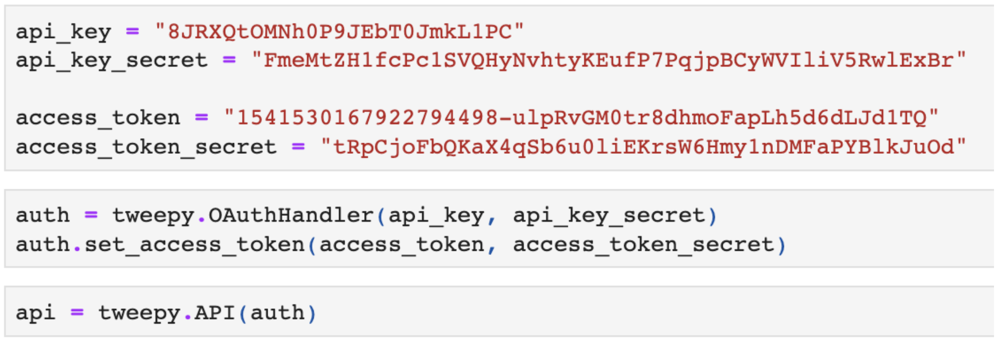
```
# Importing the necessary libraries
import os
# Tweepy as a twitter client
import tweepy
import configparser
import pandas as pd
import time
import glob
from datetime import datetime
```

Fig 1: Import of different required machine learning libraries, including Tweepy

1. **We collected data per our requirement of different keywords from Twitter**

According to the needs of our study. We collected data from Twitter through the use of Twitter API, which is eased by the Twitter API wrapper library for python, known as tweepy (Fig.1).

We also connected with the Twitter API through the consumer key and access token, which helped us for allowing the fetch of data from the Twitter database. Using the OAuthHandler function, we connected to Twitter (Fig. 2.).



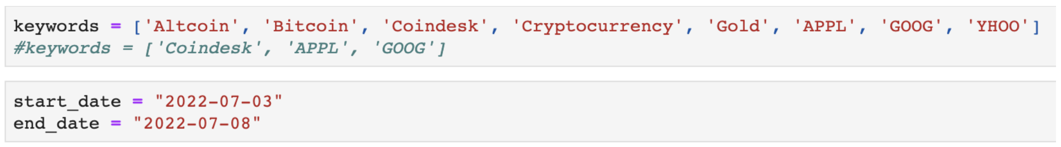
```
api_key = "8JRXQtOMNh0P9JEbT0JmkL1PC"
api_key_secret = "FmeMtZH1fcPc1SVQHyNvhtyKEufP7PqjpBCyWVIliV5RwlExBr"

access_token = "1541530167922794498-ulpRvGM0tr8dhmoFapLh5d6dLJd1TQ"
access_token_secret = "tRpCjoFbQKaX4qSb6u0liEKrsW6Hmy1nDMFaPYBlkJuOd"
```

```
auth = tweepy.OAuthHandler(api_key, api_key_secret)
auth.set_access_token(access_token, access_token_secret)
```

```
api = tweepy.API(auth)
```

Fig 2: Consumer Key and Access Token Declaration, along with a link to Twitter using these keys.

We fetched data per the keyword and did the query for a week, of which we declared the variables (Fig.3.).

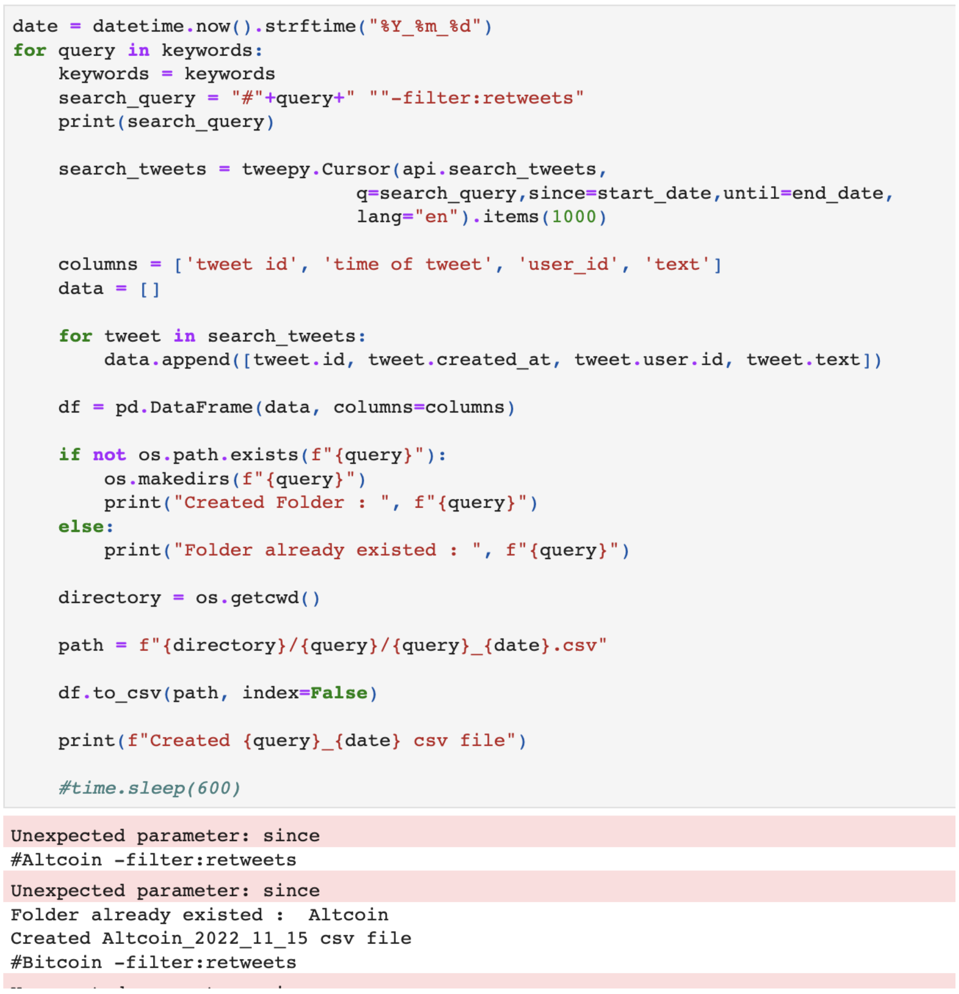


```
keywords = ['Altcoin', 'Bitcoin', 'Coindesk', 'Cryptocurrency', 'Gold', 'APPL', 'GOOG', 'YHOO']
#keywords = ['Coindesk', 'APPL', 'GOOG']
```

```
start_date = "2022-07-03"
end_date = "2022-07-08"
```

Fig 3: Variable for the Keyword and the time intervals.

Then further fetched the data using the tweepy.cursor() function provided by tweepy for the keywords on the given interval. (Fig. 4.).



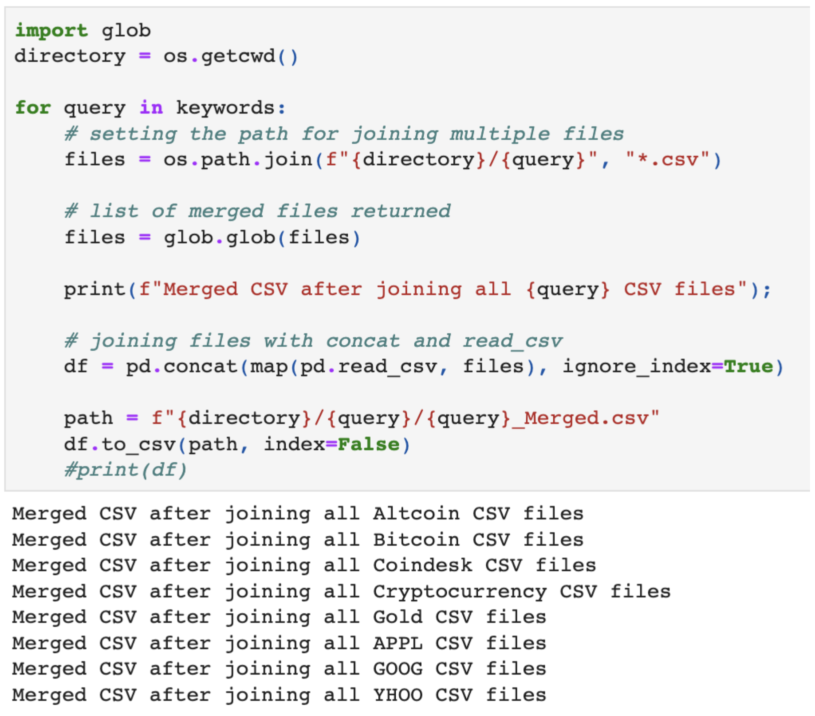
```
date = datetime.now().strftime("%Y_%m_%d")
for query in keywords:
    keywords = keywords
    search_query = "#"+query+" ""-filter:retweets"
    print(search_query)

    search_tweets = tweepy.Cursor(api.search_tweets,
                                  q=search_query,since=start_date,until=end_date,
                                  lang="en").items(1000)

    columns = ['tweet id', 'time of tweet', 'user_id', 'text']
    data = []

    for tweet in search_tweets:
        data.append([tweet.id, tweet.created_at, tweet.user.id, tweet.text])

    df = pd.DataFrame(data, columns=columns)

    if not os.path.exists(f"{query}"):
        os.makedirs(f"{query}")
        print("Created Folder : ", f"{query}")
    else:
        print("Folder already existed : ", f"{query}")

    directory = os.getcwd()

    path = f"{directory}/{query}/{query}_{date}.csv"

    df.to_csv(path, index=False)

    print(f"Created {query}_{date} csv file")

    #time.sleep(600)
```

```
Unexpected parameter: since
#Altcoin -filter:retweets
Unexpected parameter: since
Folder already existed :  Altcoin
Created Altcoin_2022_11_15 csv file
#Bitcoin -filter:retweets
```

Fig 4: Fetched Tweets in JSON format, extracted required feature columns and created Dataframe and stored them in relevant .csv file

2. **We created a data frame for each keyword and stored data collected from Twitter.**

Then, further, we developed individual data frames from each fetched data. The fetched data was in JSON format and had a variety of features in it as per the requirement of our project. We brought features, namely, tweet id, time of the tweet, user id and text, creating a variable named column (Fig.4.). Then, we developed individual CSV for the keywords used in the fetching of data from Twitter (Fig.4). In which for each fetch we checked if the folder for particular keyword existed or not and created or updated the folder for each keyword for a specific interval of time, per fetch and saved the data frame in it. Further, the completed data frame at each particular time created multiple folders for each keyword which were then further merged before further analysis. (Fig.5.)



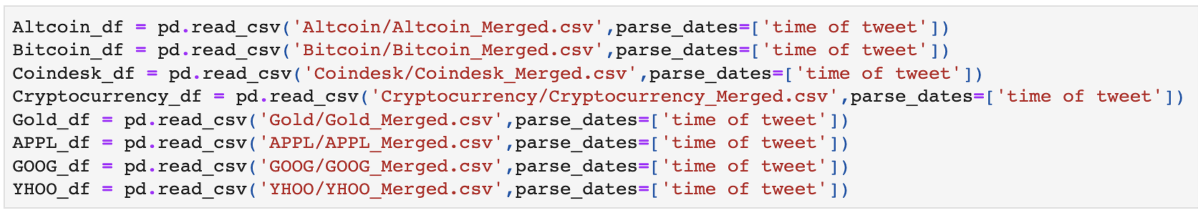
```
import glob
directory = os.getcwd()

for query in keywords:
    # setting the path for joining multiple files
    files = os.path.join(f"{directory}/{query}", "*.csv")

    # list of merged files returned
    files = glob.glob(files)

    print(f"Merged CSV after joining all {query} CSV files");

    # joining files with concat and read_csv
    df = pd.concat(map(pd.read_csv, files), ignore_index=True)

    path = f"{directory}/{query}/{query}_Merged.csv"
    df.to_csv(path, index=False)
    #print(df)
```

```
Merged CSV after joining all Altcoin CSV files
Merged CSV after joining all Bitcoin CSV files
Merged CSV after joining all Coindesk CSV files
Merged CSV after joining all Cryptocurrency CSV files
Merged CSV after joining all Gold CSV files
Merged CSV after joining all APPL CSV files
Merged CSV after joining all GOOG CSV files
Merged CSV after joining all YHOO CSV files
```

Fig 5: Merged multiple CSV of each keyword into one

3. **Performed Cleaning of Data**

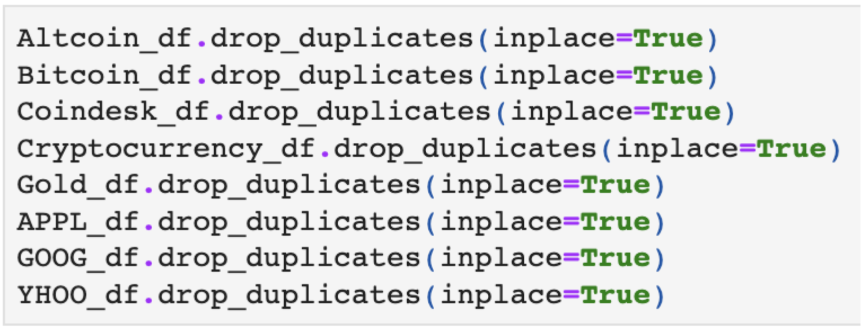
Then, further, we fetched the data from the merged csv for each keyword (Fig.6) for proceeding with data cleaning.



```
Altcoin_df = pd.read_csv('Altcoin/Altcoin_Merged.csv',parse_dates=['time of tweet'])
Bitcoin_df = pd.read_csv('Bitcoin/Bitcoin_Merged.csv',parse_dates=['time of tweet'])
Coindesk_df = pd.read_csv('Coindesk/Coindesk_Merged.csv',parse_dates=['time of tweet'])
Cryptocurrency_df = pd.read_csv('Cryptocurrency/Cryptocurrency_Merged.csv',parse_dates=['time of tweet'])
Gold_df = pd.read_csv('Gold/Gold_Merged.csv',parse_dates=['time of tweet'])
APPL_df = pd.read_csv('APPL/APPL_Merged.csv',parse_dates=['time of tweet'])
GOOG_df = pd.read_csv('GOOG/GOOG_Merged.csv',parse_dates=['time of tweet'])
YHOO_df = pd.read_csv('YHOO/YHOO_Merged.csv',parse_dates=['time of tweet'])
```

Fig 6: Read the merged csv files for data extraction.

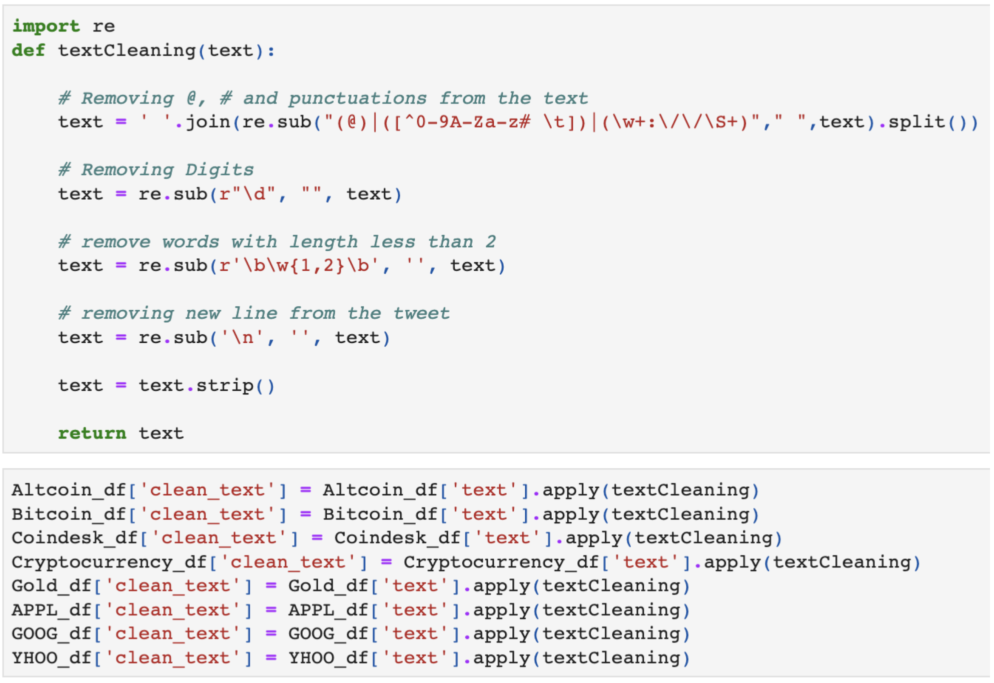
On extracting data from the merged csv we performed EDA and proceeded further. Firstly we dropped the duplicate values of data using the pandas drop_duplicates function. (Fig. 7.).



```
Altcoin_df.drop_duplicates(inplace=True)
Bitcoin_df.drop_duplicates(inplace=True)
Coindesk_df.drop_duplicates(inplace=True)
Cryptocurrency_df.drop_duplicates(inplace=True)
Gold_df.drop_duplicates(inplace=True)
APPL_df.drop_duplicates(inplace=True)
GOOG_df.drop_duplicates(inplace=True)
YHOO_df.drop_duplicates(inplace=True)
```

Fig 7: Dropping duplicate values

Also, for this multi-step procedure of text cleaning, we created a function named **textCleaning(keyword)** (Fig.8.), which was applied to the data frame of all the respective keywords. (Fig. 8.) for performing different cleaning algorithms.

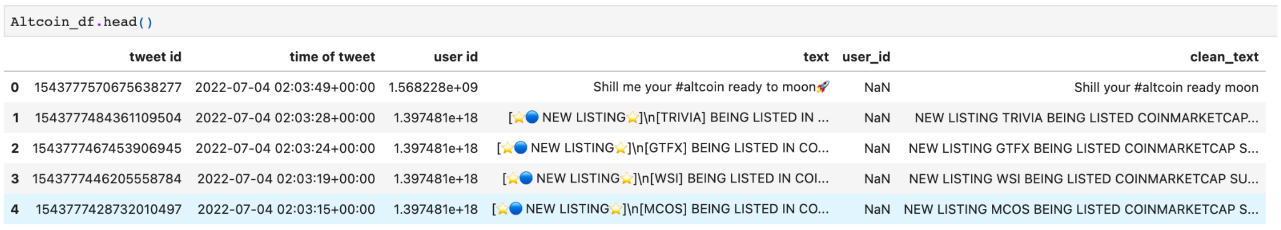


```
import re
def textCleaning(text):

    # Removing @, # and punctuations from the text
    text = ' '.join(re.sub("(@)|([^0-9A-Za-z# \t])|(\w+:\/\/\S+)"," ",text).split())

    # Removing Digits
    text = re.sub(r"\d", "", text)

    # remove words with length less than 2
    text = re.sub(r'\b\w{1,2}\b', '', text)

    # removing new line from the tweet
    text = re.sub('\n', '', text)

    text = text.strip()

    return text
```

```
Altcoin_df['clean_text'] = Altcoin_df['text'].apply(textCleaning)
Bitcoin_df['clean_text'] = Bitcoin_df['text'].apply(textCleaning)
Coindesk_df['clean_text'] = Coindesk_df['text'].apply(textCleaning)
Cryptocurrency_df['clean_text'] = Cryptocurrency_df['text'].apply(textCleaning)
Gold_df['clean_text'] = Gold_df['text'].apply(textCleaning)
APPL_df['clean_text'] = APPL_df['text'].apply(textCleaning)
GOOG_df['clean_text'] = GOOG_df['text'].apply(textCleaning)
YHOO_df['clean_text'] = YHOO_df['text'].apply(textCleaning)
```

Fig 8: Creating a function for data cleaning (textCleaning) and applying it to data frames.

Then further, the cleansed data was observed (Fig.9.).



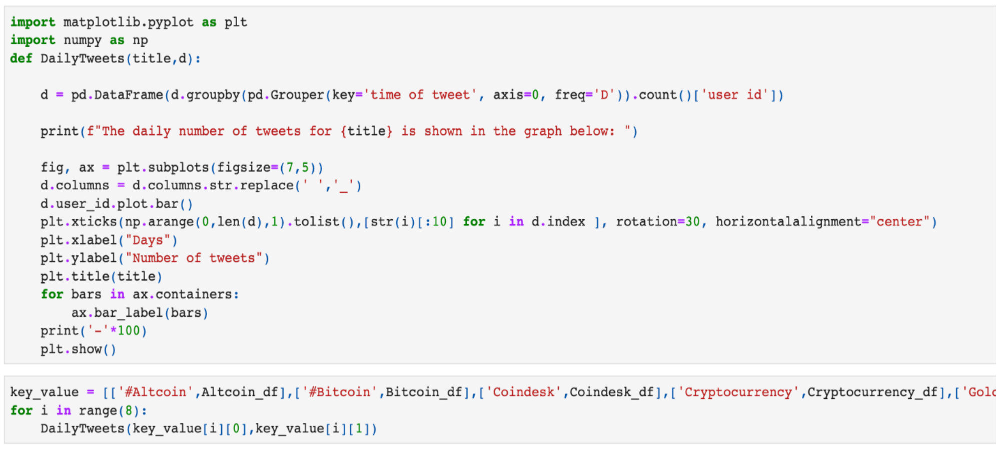
```
Altcoin_df.head()
```

	tweet id	time of tweet	user id	text	user_id	clean_text
0	1543777570675638277	2022-07-04 02:03:49+00:00	1.568228e+09	Shill me your #altcoin ready to moon🚀	NaN	Shill your #altcoin ready moon
1	1543777484361109504	2022-07-04 02:03:28+00:00	1.397481e+18	[⭐🔵 NEW LISTING⭐]\n[TRIVIA] BEING LISTED IN ...	NaN	NEW LISTING TRIVIA BEING LISTED COINMARKETCAP...
2	1543777467453906945	2022-07-04 02:03:24+00:00	1.397481e+18	[⭐🔵 NEW LISTING⭐]\n[GTFX] BEING LISTED IN CO...	NaN	NEW LISTING GTFX BEING LISTED COINMARKETCAP S...
3	1543777446205558784	2022-07-04 02:03:19+00:00	1.397481e+18	[⭐🔵 NEW LISTING⭐]\n[WSI] BEING LISTED IN COI...	NaN	NEW LISTING WSI BEING LISTED COINMARKETCAP SU...
4	1543777428732010497	2022-07-04 02:03:15+00:00	1.397481e+18	[⭐🔵 NEW LISTING⭐]\n[MCOS] BEING LISTED IN CO...	NaN	NEW LISTING MCOS BEING LISTED COINMARKETCAP S...

Fig 9: Observation of cleansed data

4. **Performed visualization of the cleansed Data.**

We graphically represented the cleansed data and observed it. For this, matplotlib was used for the visualization. Similarly, for creating individual graphs of each keyword based on a daily number of tweets, a variable key_value was created (Fig.10.) and passed through a function DailyTweets(), created for the operation of visualization of data for the number of tweets based on days. (Fig.10.). Yielding graphical representation of data as shown below. (Fig. 11.). Along with that, for creating individual graphs of each keyword based on a daily number of users, a variable key_value was created and passed through a function daily users(), created for the operation of visualization of data for several users based on days. (Fig.12). Yielding graphical representation of data as shown below. (Fig. 13.)

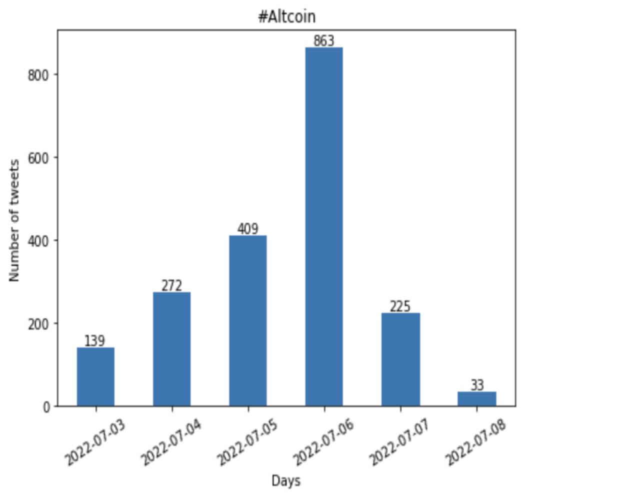


```python
import matplotlib.pyplot as plt
import numpy as np
def DailyTweets(title,d):

    d = pd.DataFrame(d.groupby(pd.Grouper(key='time of tweet', axis=0, freq='D')).count()['user id'])

    print(f"The daily number of tweets for {title} is shown in the graph below: ")

    fig, ax = plt.subplots(figsize=(7,5))
    d.columns = d.columns.str.replace(' ','_')
    d.user_id.plot.bar()
    plt.xticks(np.arange(0,len(d),1).tolist(),[str(i)[:10] for i in d.index ], rotation=30, horizontalalignment="center")
    plt.xlabel("Days")
    plt.ylabel("Number of tweets")
    plt.title(title)
    for bars in ax.containers:
        ax.bar_label(bars)
    print('-'*100)
    plt.show()
```

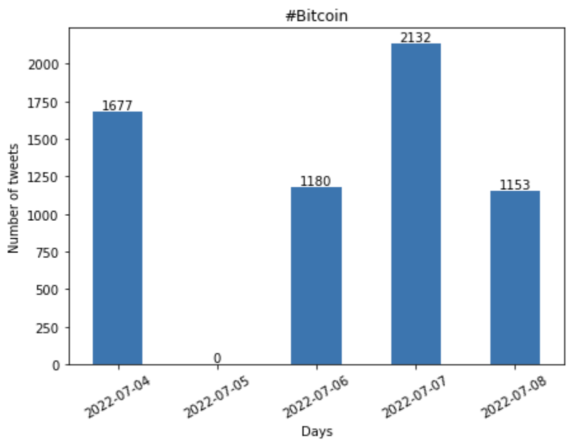
```python
key_value = [['#Altcoin',Altcoin_df],['#Bitcoin',Bitcoin_df],['Coindesk',Coindesk_df],['Cryptocurrency',Cryptocurrency_df],['Gol
for i in range(8):
    DailyTweets(key_value[i][0],key_value[i][1])
```

Fig 10: The function declaration of DailyTweets and key_value variable declaration for keywords (altcoin, bitcoin, coin desk, cryptocurrency, gold, appl, goog and yahoo, respectively)

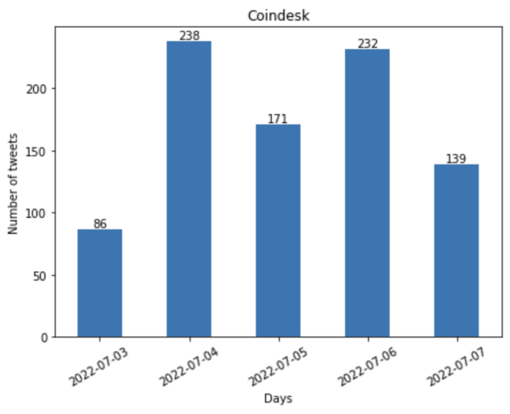
```
The daily number of tweets for #Altcoin is shown in the graph below:
---------------------------------------------------------------------
```



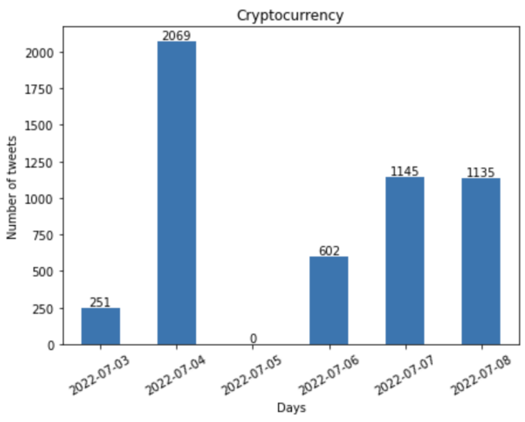
```
The daily number of tweets for #Bitcoin is shown in the graph below:
---------------------------------------------------------------------
```



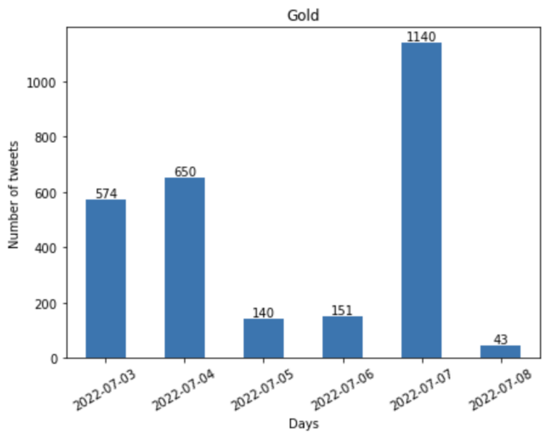
```
The daily number of tweets for Coindesk is shown in the graph below:
---------------------------------------------------------------------
```



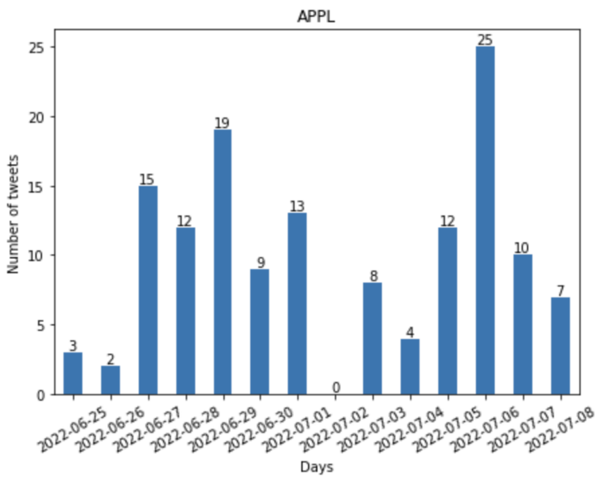
```
The daily number of tweets for Cryptocurrency is shown in the graph below:
--------------------------------------------------------------------------
```



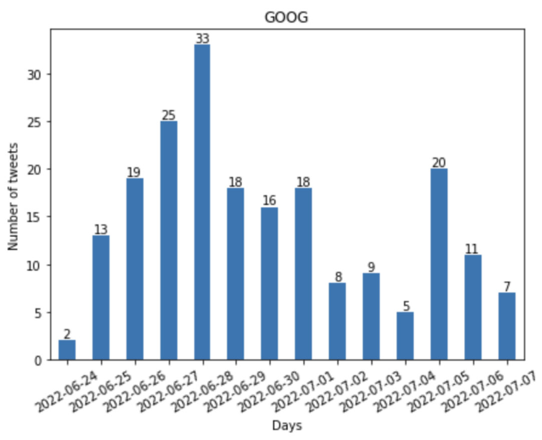
```
The daily number of tweets for Gold is shown in the graph below:
---------------------------------------------------------------------
```



```
The daily number of tweets for APPL is shown in the graph below:
---------------------------------------------------------------------
```



```
The daily number of tweets for GOOG is shown in the graph below:
-----------------------------------------------------------------
```



```
The daily number of tweets for YHOO is shown in the graph below:
-----------------------------------------------------------------
```

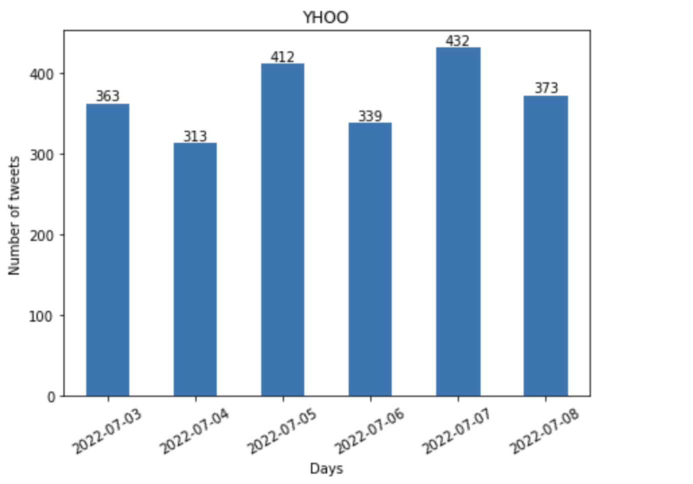
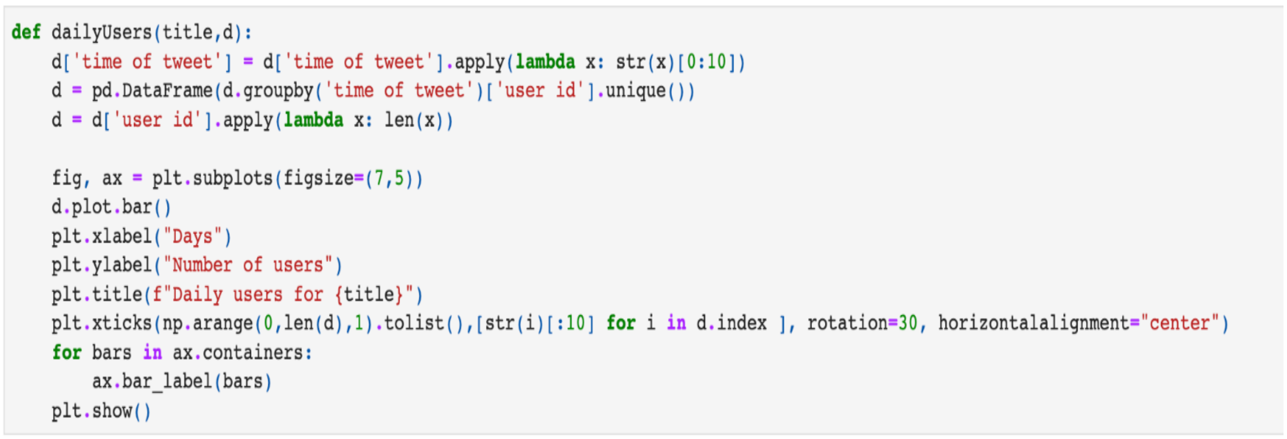
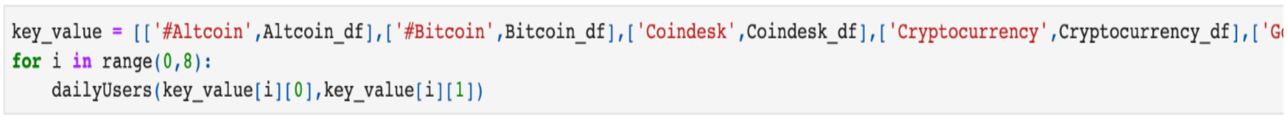


Fig 11: Graphical Representation of the cleansed data based on the number of tweets per day



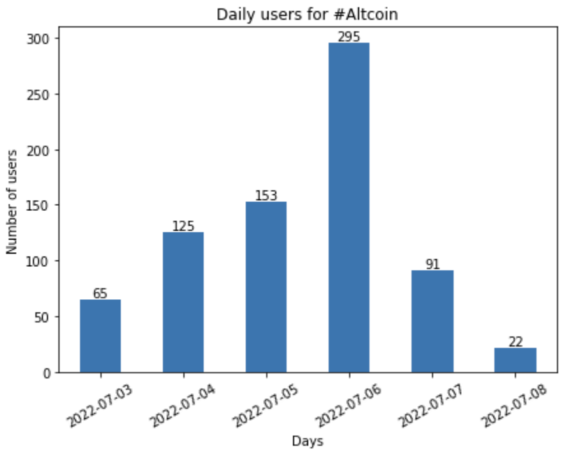
```
def dailyUsers(title,d):
    d['time of tweet'] = d['time of tweet'].apply(lambda x: str(x)[0:10])
    d = pd.DataFrame(d.groupby('time of tweet')['user id'].unique())
    d = d['user id'].apply(lambda x: len(x))

    fig, ax = plt.subplots(figsize=(7,5))
    d.plot.bar()
    plt.xlabel("Days")
    plt.ylabel("Number of users")
    plt.title(f"Daily users for {title}")
    plt.xticks(np.arange(0,len(d),1).tolist(),[str(i)[:10] for i in d.index ], rotation=30, horizontalalignment="center")
    for bars in ax.containers:
        ax.bar_label(bars)
    plt.show()
```

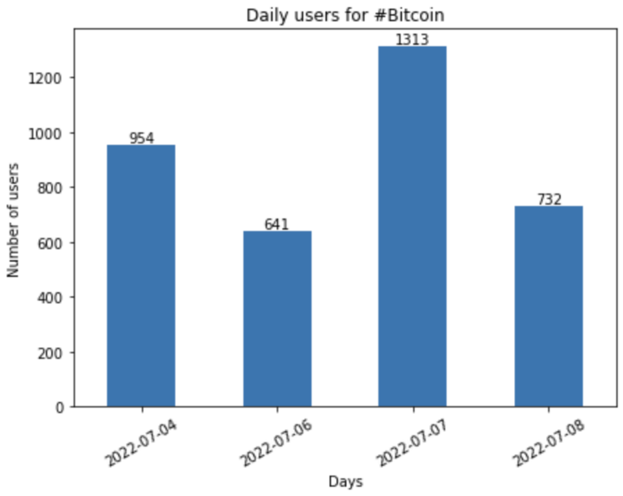


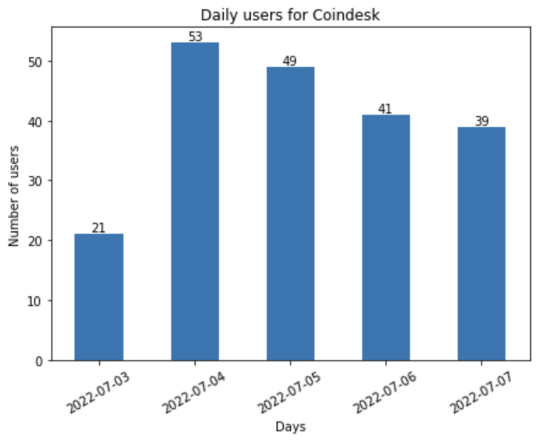
```
key_value = [['#Altcoin',Altcoin_df],['#Bitcoin',Bitcoin_df],['Coindesk',Coindesk_df],['Cryptocurrency',Cryptocurrency_df],['G
for i in range(0,8):
    dailyUsers(key_value[i][0],key_value[i][1])
```

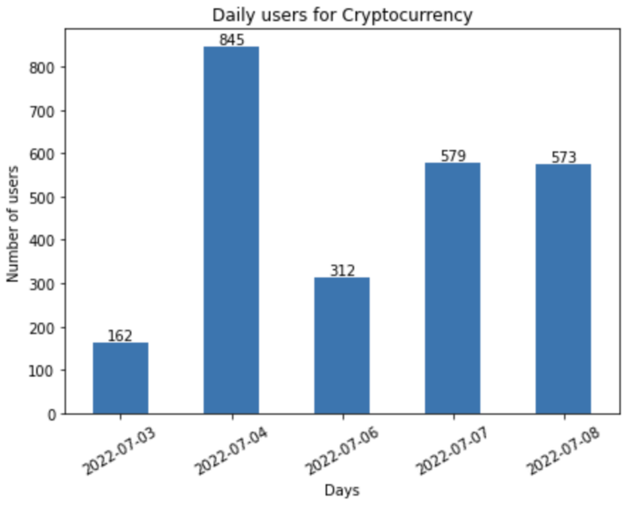
for each

Fig 12: The function declaration of dailyUsers and key_value variable declaration for keywords (altcoin, bitcoin, coindesk, cryptocurrency, gold, appl, goog and yahoo, respectively)


Daily users for #Altcoin
Number of users
Days
65
125
153
295
91
22
2022-07-03
2022-07-04
2022-07-05
2022-07-06
2022-07-07
2022-07-08


Daily users for #Bitcoin
Number of users
Days
954
641
1313
732
2022-07-04
2022-07-06
2022-07-07
2022-07-08


Daily users for Coindesk
Number of users
Days
21
53
49
41
39
2022-07-03
2022-07-04
2022-07-05
2022-07-06
2022-07-07


Daily users for Cryptocurrency
Number of users
Days
162
845
312
579
573
2022-07-03
2022-07-04
2022-07-06
2022-07-07
2022-07-08

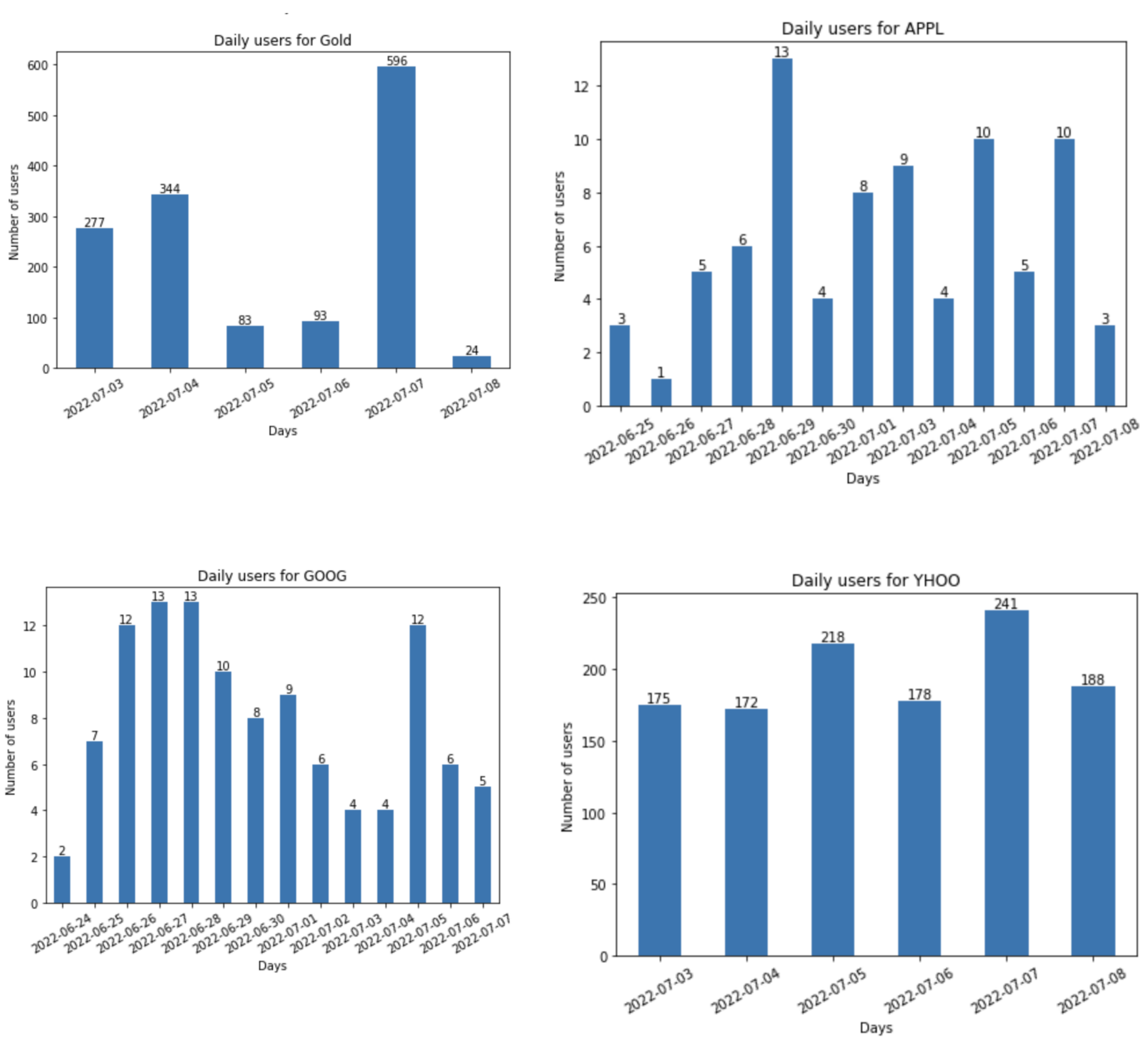


Fig 13 : Graphical Representation of the cleansed data based on number of users per day for each keyword (altcoin, bitcoin, coin desk, cryptocurrency, gold, appl, goog and yahoo, respectively).

Result and Conclusion

Through this project, we learnt several techniques of data collection, data cleansing and visual representation, where we used tweepy and collected data from Twitter.

We then created and stored the data frame in csv format for the JSON extracted data from Twitter for each fetched keyword; we also read data from csv, from which we learnt to store and query the data as per need. We also collected data from the Twitter database using tweepy, which extracted real-time data. Furthermore, after collection and storage with the query of data, we performed different cleaning algorithms on the data of the data frame. From this, we learnt different text cleaning steps like dropping duplicate data, removal of punctuation, space and stop words, and stripping of text.

Hence, we collected Twitter data related to stock, cleansed it and graphically represented it through different libraries available in python. We also gained knowledge of the libraries to be used and the step to be followed for the procedure.

Reference

Tweepy. (n.d.). *Tweepy Documentation.*Tweepy. https://docs.tweepy.org/en/stable/

Wikipedia.(n.d). *Twitter*. Wikipedia.
https://en.wikipedia.org/wiki/Twitter#:~:text=Subsequently%2C%20Musk%20made%20an%20unsolicited,the%20top%20three%20Twitter%20executives.