# Full_aidi_project

March 20, 2021

[1]:
```
import requests# to retrieve the html source code of the website
from bs4 import BeautifulSoup# to create beautiful soup object from html source␣
 ↪code
import pandas as pd# to work with dataframes
import schedule# to perform periodic scan
import time# to retrieve the time
from datetime import datetime# to convert time into readable format
```

[40]:
```
import matplotlib.pyplot as plt
import yfinance as yf
```

[3]:
```
web_dict = {'Headline': [], 'Paragraph': []}# dictionary to store headlines and␣
 ↪paragraph of the website.
stock_symbol = []## to store the found symbol
```

[4]:
```
stock_symbol_dic = {'symbol':[], 'occurence':[]}# to store the found symbol and␣
 ↪their occurence.
```

[6]:
```
a=0
def new_parser(pages=2, first=11):
    """This function will extract the headlines and paragraph from the given␣
 ↪website.
        First run, when a=0, it will scan pages from 2 to 10."""

    global a
    if a != 0:# For other runs, a!=0, it will scan periodically the first two␣
 ↪pages.
        first = pages
        pages = 0


    print("Scanning web pages from {} to {}".format(pages, first))

    for i in range(pages, first):# loop to scan through number of pages of␣
 ↪website
```

```
        website = f'https://www.prnewswire.com/news-releases/news-releases-list/
↪?page={i+1}&pagesize=100'

        source = requests.get(website).text# to retrieve the html code
        soup = BeautifulSoup(source, 'lxml')# creating the beautiful soup,␣
↪which represents the document nested as a data structure.

        articles = soup.find('div', class_="col-md-8 col-sm-8 card-list␣
↪card-list-hr")# find div tag with given class

        for article in articles.find_all('div', class_ = 'row'):# loop for all␣
↪the div tag with given class
            head = article.a.h3.text# to extract the headline
            extract = lambda x: x.split('ET')[1].strip()# to grasp the main␣
↪portion, and to remove the time
            headline = extract(head)
            paragraph = article.a.p.text# to extract the paragraph corresponds␣
↪to the heading

            web_dict['Headline'].append(headline)# append the heading to the␣
↪heading list in web_dict
            web_dict['Paragraph'].append(paragraph)# append the heading to the␣
↪heading list in web_dict

            web_dict['Headline'] = list(set(web_dict['Headline']))# Set is used␣
↪to remove duplicacy, and then converted to list.
            web_dict['Paragraph'] = list(set(web_dict['Paragraph']))
    a=1
    print('Found {} news articles till now.'.format(len(web_dict['Headline'])))
    print(' ')
```

[7]:
```
def scheduler(period, epochs):
    """ This function will help to scan the website periodically
    period: it's the time difference between two loops
    epochs: it's the number of time you want to scan the given website"""

    start_time = time.time()# to get the start time of scrapping
    sec_date = lambda x: datetime.fromtimestamp(x).strftime("%B %d, %Y %I:%M")#␣
↪function to convert time (in seconds) to a more readable format.



    j = 0
    schedule.every(period).minutes.do(new_parser)# to run the parser function␣
↪every "period" minutes
```

```
    while j < epochs*period:# this loop will run for total time i.e␣
↪"epochs*period"

        print(sec_date(time.time()))
        print('This code will run for another {} minutes'.format(epochs*period␣
↪- j))# time remaining

        schedule.run_pending()# to run the function which is pending with the␣
↪schedule
        time.sleep(60*period)# this will help to save the computational power,␣
↪and hault the execution for "60*period" minutes

        j+=period# incrementing j with "period"


    end_time = time.time()
#     print(f"Web scrapping end at: {sec_date(end_time)}")# to get the end time␣
↪of scrapping
```

[8]:
```
scheduler(30,6)
```

```
March 19, 2021 06:23
This code will run for another 180 minutes
March 19, 2021 06:53
This code will run for another 150 minutes
Scanning web pages from 2 to 11
Found 899 news articles till now.

March 19, 2021 07:25
This code will run for another 120 minutes
Scanning web pages from 0 to 2
Found 1096 news articles till now.

March 19, 2021 07:55
This code will run for another 90 minutes
Scanning web pages from 0 to 2
Found 1134 news articles till now.

March 19, 2021 08:25
This code will run for another 60 minutes
Scanning web pages from 0 to 2
Found 1154 news articles till now.

March 19, 2021 08:55
This code will run for another 30 minutes
Scanning web pages from 0 to 2
Found 1169 news articles till now.
```

[27]:
```
df = pd.concat([pd.DataFrame(web_dict['Headline']), pd.
↪DataFrame(web_dict['Paragraph'])], ignore_index= True, axis = 1)# create␣
↪dataframe of headings and paragraph scrapped from website.
df.columns = web_dict.keys()# columns name
df.head()
```

[27]:
```
                                            Headline  \
0  Proterra battery technology to power Lightning…
1  Together-Travel Startup, Launchtrip Raises $3…
2  PolyFlex Products to Deliver Highly Advanced P…
3  Lazydays Holdings, Inc. Reports Fourth Quarter…
4  AANP Congratulates Secretary Xavier Becerra of…

                                           Paragraph
0  As Florida lawmakers consider legislation that…
1  Global leader in sustainable technologies John…
2  More than 600 international enterprises are ex…
3  A new survey examining consumer attitudes on c…
4  La nueva compañía líder en el sector de la inv…
```

[10]:
```
df.to_csv('news1111.csv',index=False)# storing the dataframe into csv file.
```

[11]:
```
for para in web_dict['Paragraph']:# to scrap the stock symbol from paragraphs

    if '(TSX:' in para:

#         print(para)
        symbol = para.split('(TSX:')[1].split(')')[0].strip()

        if len(symbol) < 9:# ensuring only stock symbol will be extracted.
            stock_symbol.append(symbol)# appending the stock symbol into list
    #         print(' ')

print(stock_symbol)
```

[11]:
```
['EXF', 'NB', 'BLDP', 'FF', 'VB', 'RFP', 'TA', 'OTEX', 'DOO', 'MAXR']
```

[12]:
```
def occurence(lis):# to find the number of occurence of symbols

    for e in lis:
        stock_symbol_dic['symbol'].append(e)
        stock_symbol_dic['occurence'].append(lis.count(e))# appending the␣
↪number of occurence corresponds to each symbol.
```

[13]:
```
occurence(stock_symbol)
```

```
[37]: print(stock_symbol_dic)

{'symbol': ['EXF', 'NB', 'BLDP', 'FF', 'VB', 'RFP', 'TA', 'OTEX', 'DOO',
'MAXR'], 'occurence': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

```
[29]: out_df = pd.DataFrame(stock_symbol_dic).sort_values(by = 'occurence', ascending␣
       ↪= False)# creating data frame from stock_symbol_dic
      out_df
```

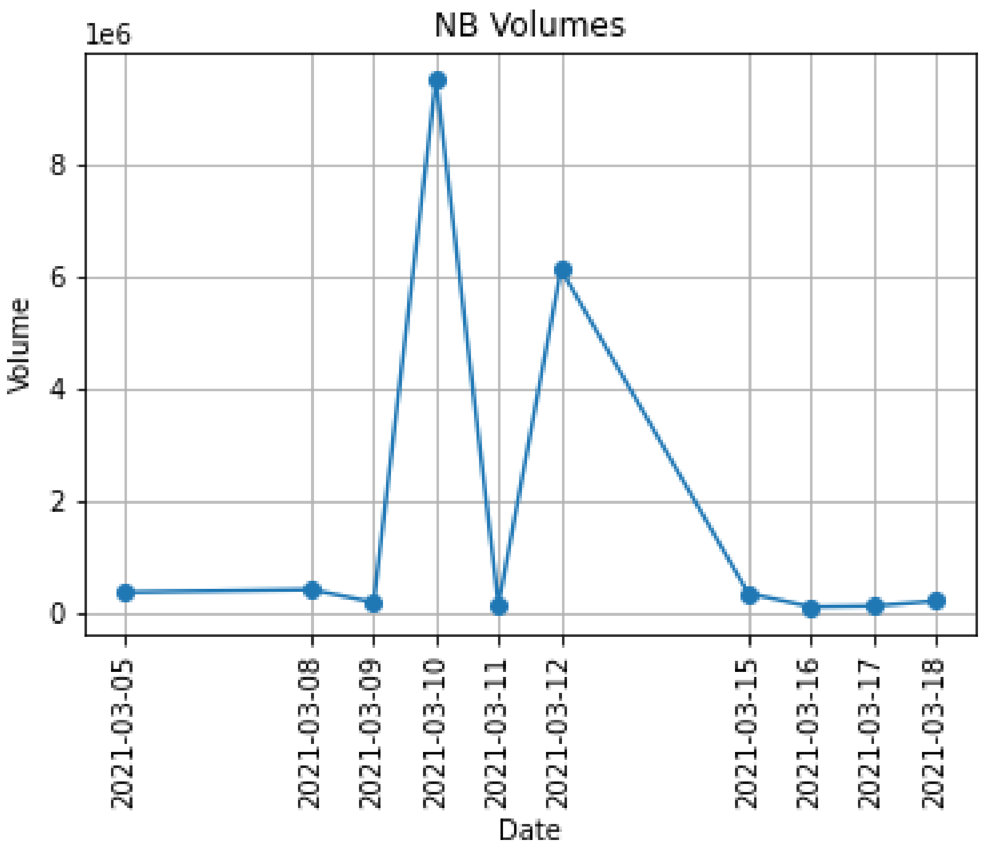
```
[29]:   symbol  occurence
      0    EXF          1
      1     NB          1
      2   BLDP          1
      3     FF          1
      4     VB          1
      5    RFP          1
      6     TA          1
      7   OTEX          1
      8    DOO          1
      9   MAXR          1
```

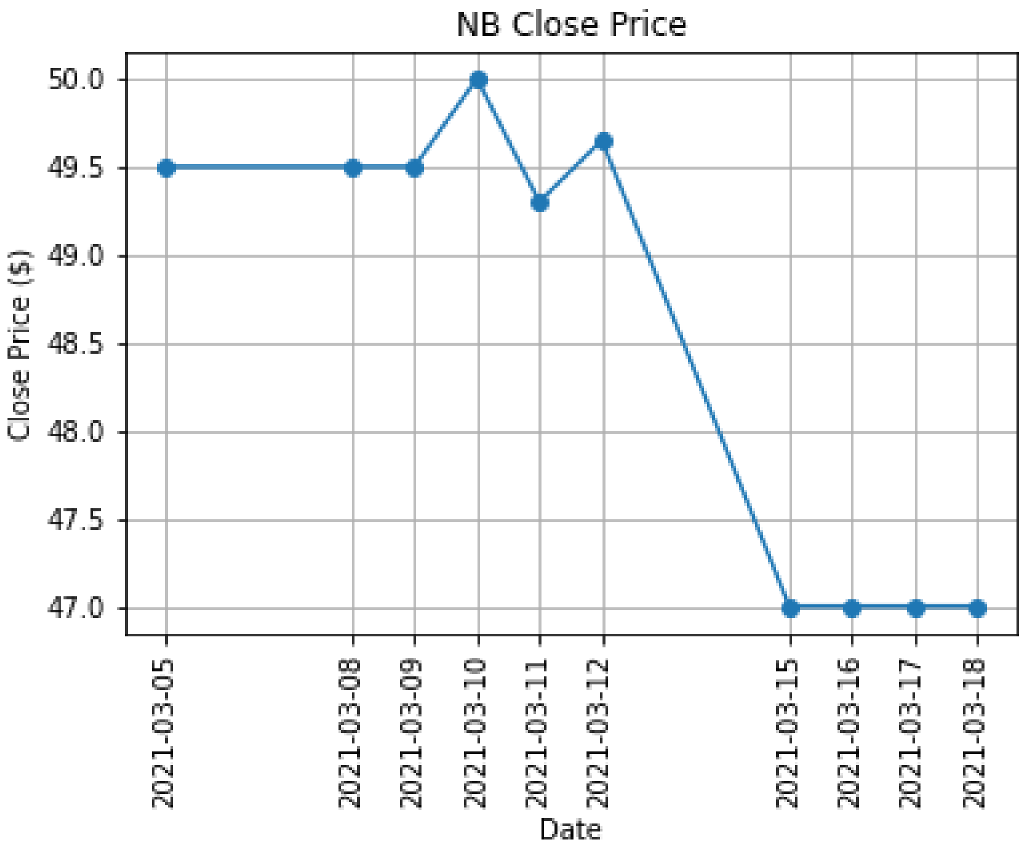
```
[30]: out_df.to_csv('symbol_occurence1111.csv', index=False)# storing the found␣
       ↪symbol and their occurence into a csv file
```

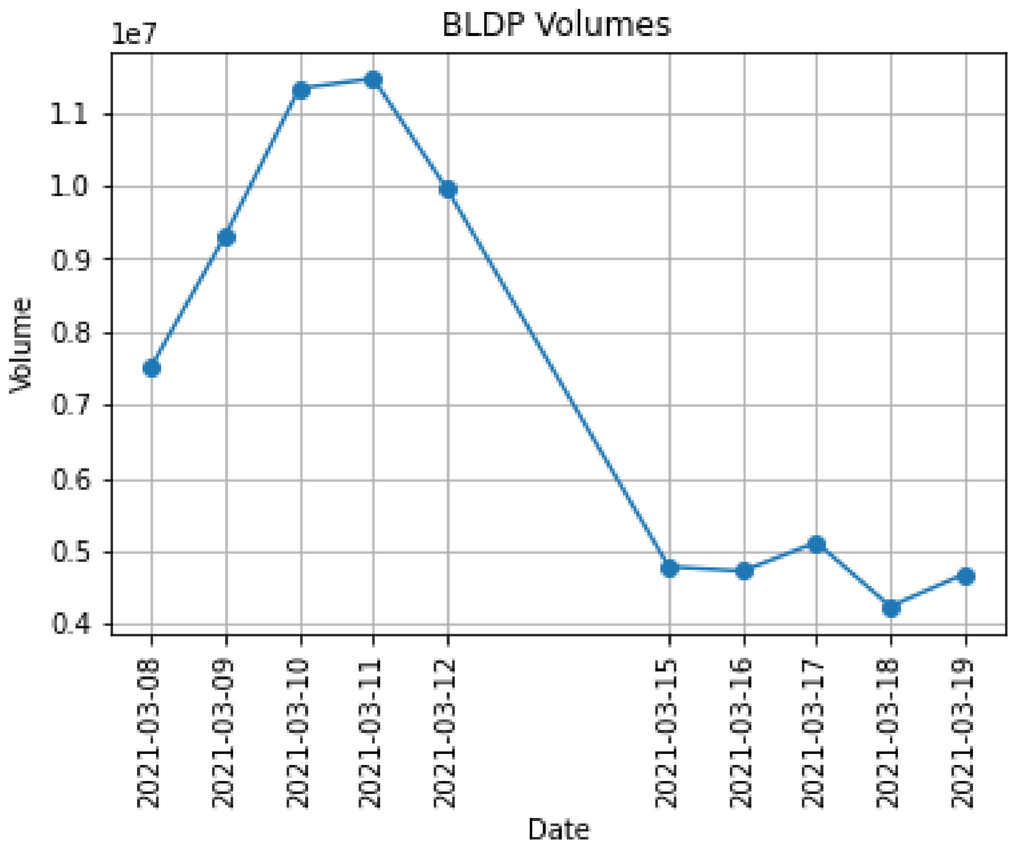
```
[38]: df = pd.read_csv('symbol_occurence1111.csv')
      symbs = df['symbol'][1:6]
```

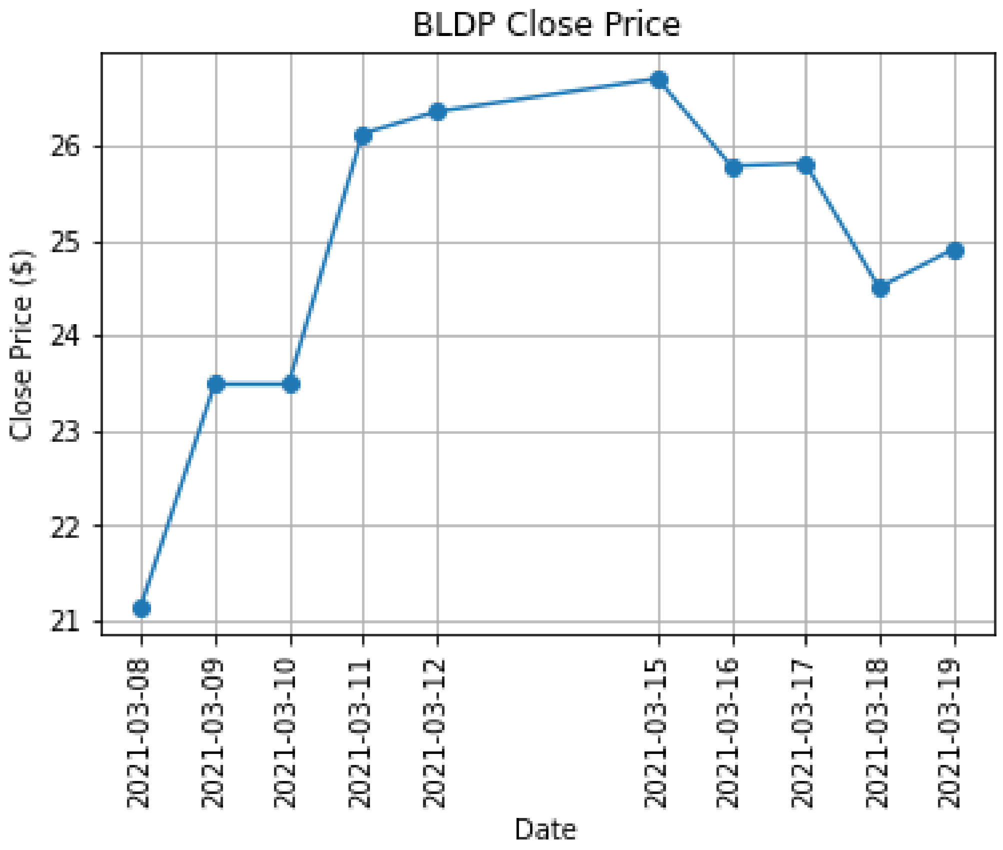
```
[41]: for symb in symbs:
          t = yf.Ticker(symb) # Create ticket object for the symbol
          h = t.history(period="1mo") # Get 1 month of historical data, 1 day␣
       ↪intervals
          # h is a pandas dataframe.
          # We will use integer indexing to get the most recent 10 days
          d10 = h.iloc[-10:]

          # Plot the Volume data
          fig = plt.figure()
          plt.plot(d10.Volume, marker="o")
          plt.title(symb + " Volumes")
          plt.xlabel("Date")
          plt.ylabel("Volume")
          # The xticks should be the dates for each data point
          # In the data, the indexes are dates
          plt.xticks(ticks=d10.index, rotation='vertical')
          plt.grid() # enable grid display
          plt.show()
          fig.savefig(symb+"_Volumes.png", dpi=400, transparent=True,␣
       ↪bbox_inches='tight')
```

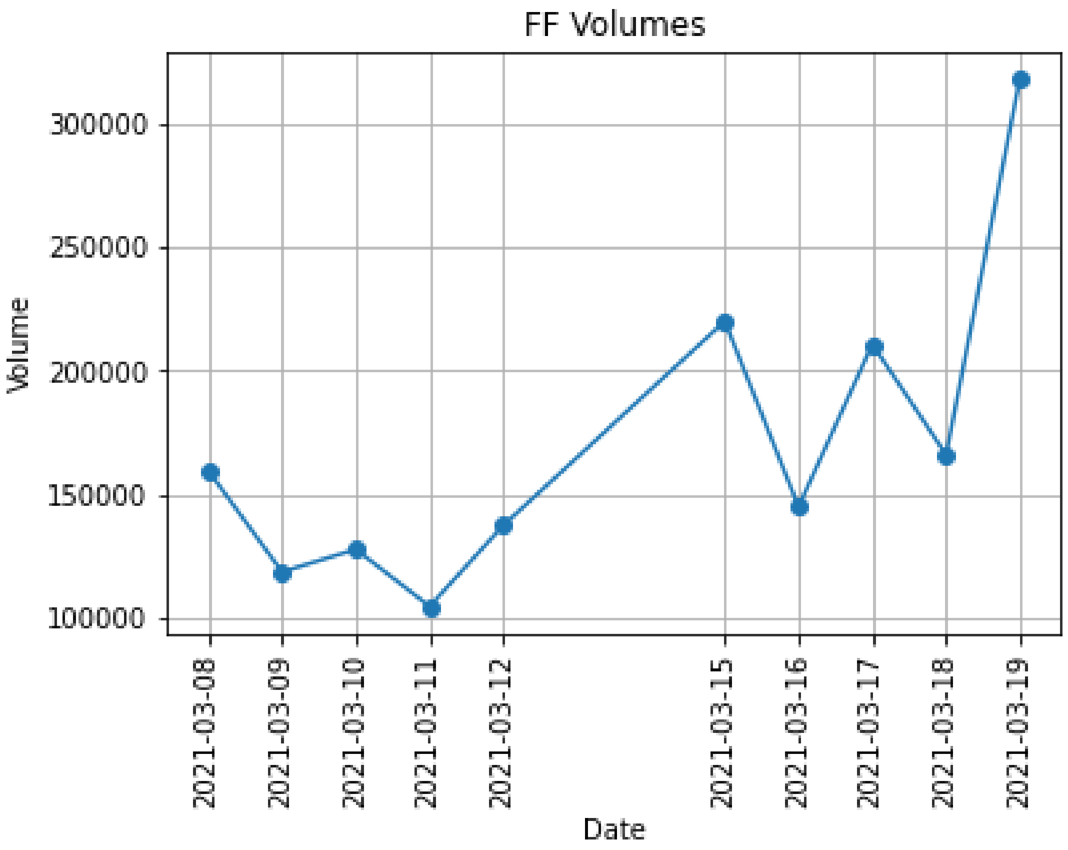
```
    # Plot the Close data
    fig = plt.figure()
    plt.plot(d10.Close, marker="o")
    plt.title(symb + " Close Price")
    plt.xlabel("Date")
    plt.ylabel("Close Price ($)")
    # The xticks should be the dates for each data point
    # In the data, the indexes are dates
    plt.xticks(ticks=d10.index, rotation='vertical')
    plt.grid() # enable grid display
    plt.show()
    fig.savefig(symb+"_Close.png", dpi=400, transparent=True,␣
↪bbox_inches='tight')
```

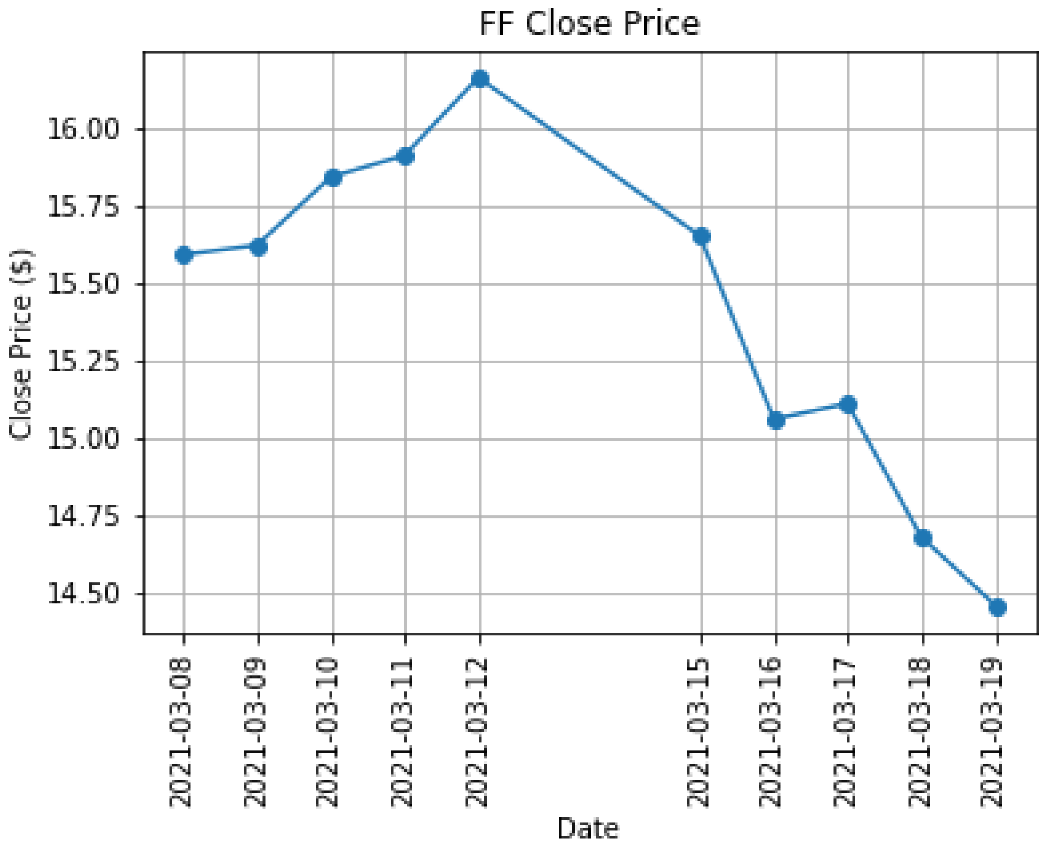


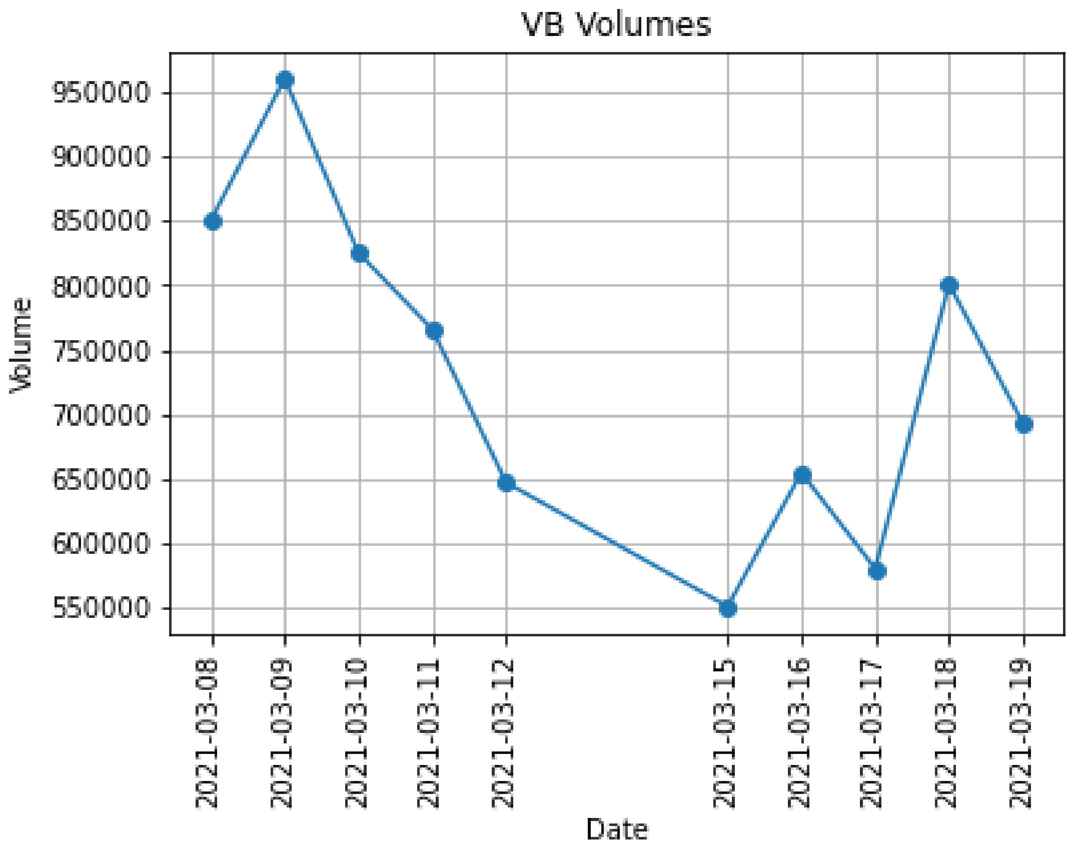

NB Close Price
Close Price ($)
50.0
49.5
49.0
48.5
48.0
47.5
47.0
2021-03-05
2021-03-08
2021-03-09
2021-03-10
2021-03-11
2021-03-12
2021-03-15
2021-03-16
2021-03-17
2021-03-18
Date

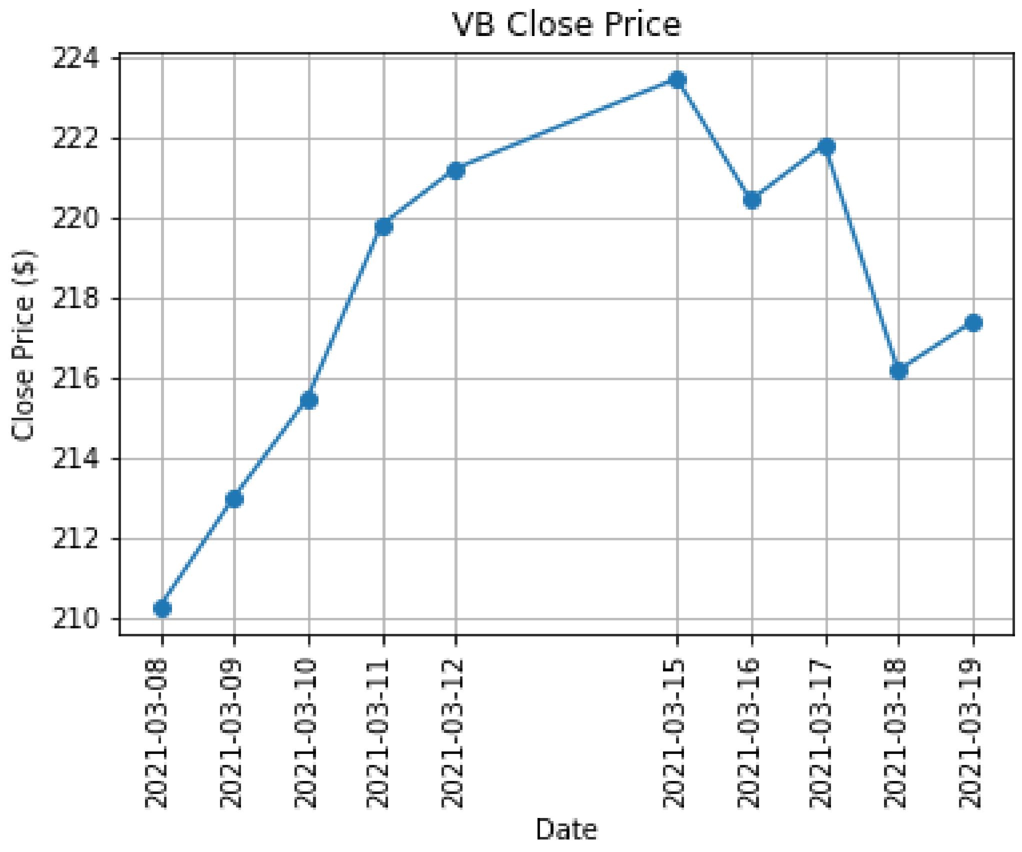

BLDP Volumes
1e7
Volume
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
2021-03-08
2021-03-09
2021-03-10
2021-03-11
2021-03-12
2021-03-15
2021-03-16
2021-03-17
2021-03-18
2021-03-19
Date

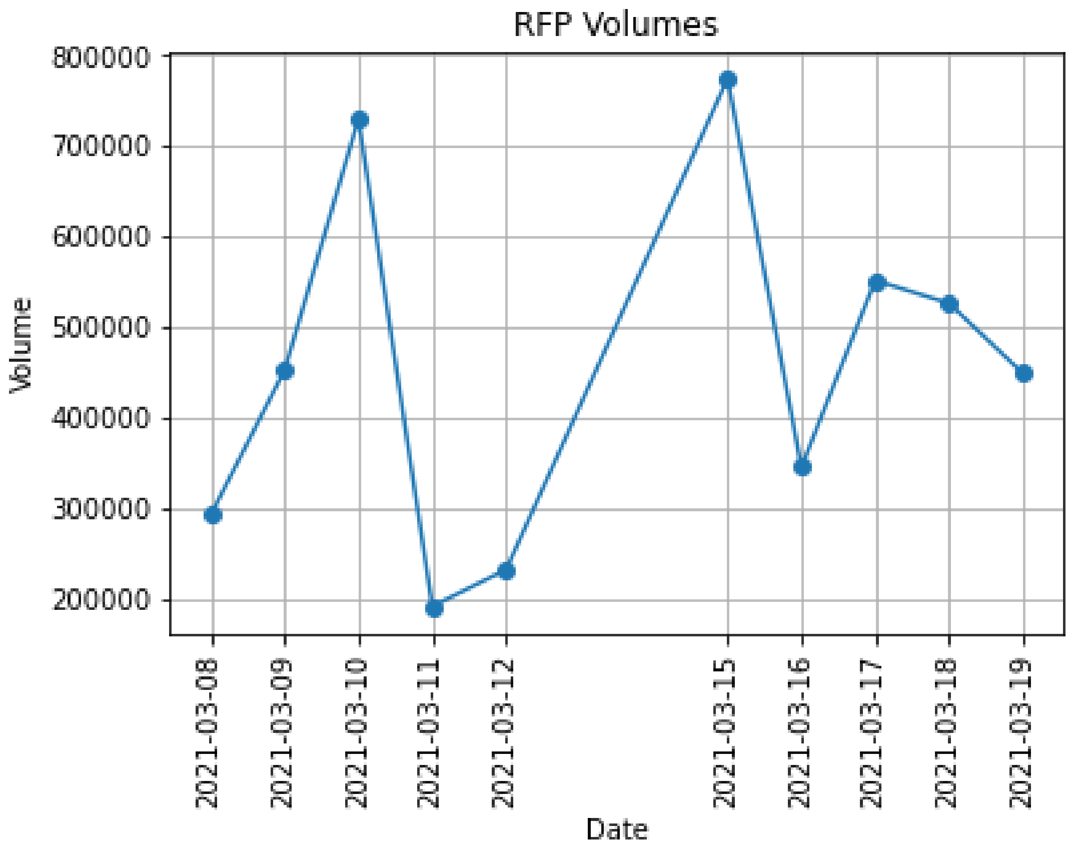

BLDP Close Price
Close Price ($)
21
22
23
24
25
26
2021-03-08
2021-03-09
2021-03-10
2021-03-11
2021-03-12
2021-03-15
2021-03-16
2021-03-17
2021-03-18
2021-03-19
Date

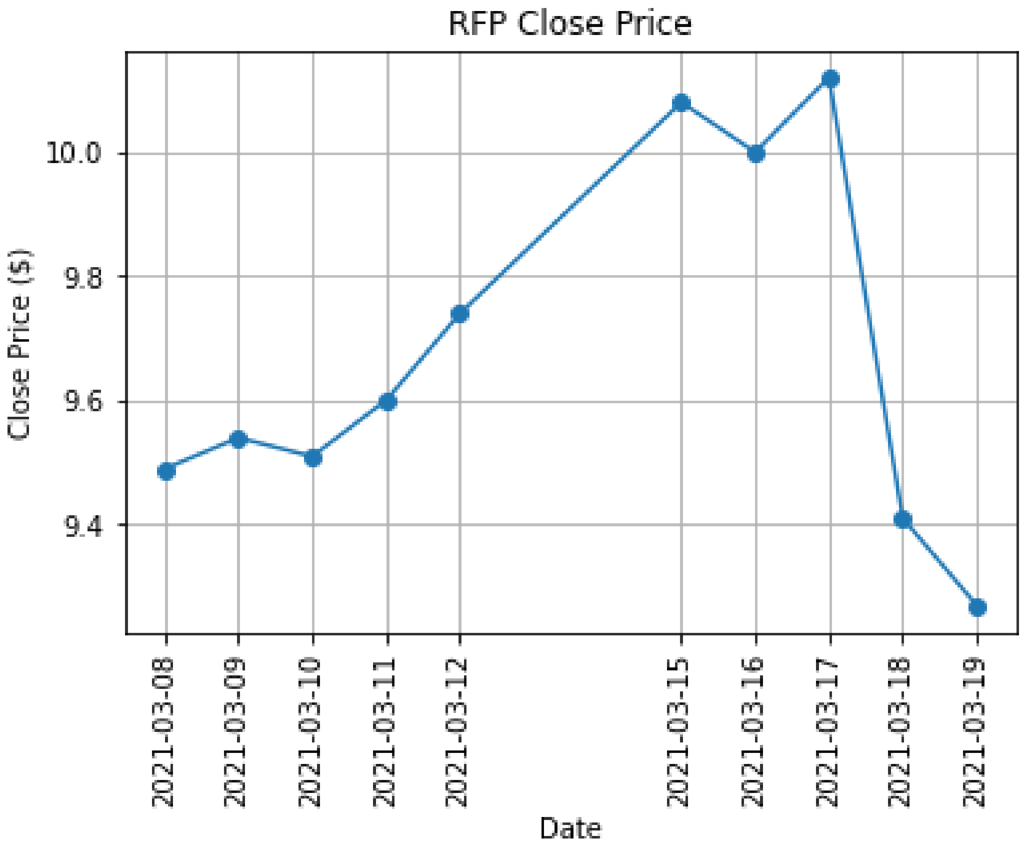

FF Volumes
Volume
300000
250000
200000
150000
100000
Date
2021-03-08
2021-03-09
2021-03-10
2021-03-11
2021-03-12
2021-03-15
2021-03-16
2021-03-17
2021-03-18
2021-03-19


FF Close Price
Close Price ($)
16.00
15.75
15.50
15.25
15.00
14.75
14.50
2021-03-08
2021-03-09
2021-03-10
2021-03-11
2021-03-12
2021-03-15
2021-03-16
2021-03-17
2021-03-18
2021-03-19
Date


VB Volumes
Volume
950000
900000
850000
800000
750000
700000
650000
600000
550000
Date
2021-03-08
2021-03-09
2021-03-10
2021-03-11
2021-03-12
2021-03-15
2021-03-16
2021-03-17
2021-03-18
2021-03-19


VB Close Price
Close Price ($)
224
222
220
218
216
214
212
210
2021-03-08
2021-03-09
2021-03-10
2021-03-11
2021-03-12
2021-03-15
2021-03-16
2021-03-17
2021-03-18
2021-03-19
Date


RFP Volumes
Volume
800000
700000
600000
500000
400000
300000
200000
Date
2021-03-08
2021-03-09
2021-03-10
2021-03-11
2021-03-12
2021-03-15
2021-03-16
2021-03-17
2021-03-18
2021-03-19


RFP Close Price
Close Price ($)
10.0
9.8
9.6
9.4
2021-03-08
2021-03-09
2021-03-10
2021-03-11
2021-03-12
2021-03-15
2021-03-16
2021-03-17
2021-03-18
2021-03-19
Date

```
[ ]:
```