Extracting and Visualizing Stock Data

Description

Extracting essential data from a dataset and displaying it is a necessary part of data science; therefore individuals can make correct decisions based on the data. In this assignment, you will extract some stock data, you will then display this data in a graph.

Table of Contents

Estimated Time Needed: **30 min**

Note:- If you are working Locally using anaconda, please uncomment the following code and execute it.

In [1]:
```
#!pip install yfinance==0.2.38
#!pip install pandas==2.2.2
#!pip install nbformat
```

In [2]:
```
!pip install yfinance
!pip install bs4
!pip install nbformat
```

```
Collecting yfinance
  Downloading yfinance-0.2.43-py2.py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: pandas>=1.3.0 in /opt/conda/lib/python3.11/site-packa
ges (from yfinance) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in /opt/conda/lib/python3.11/site-packa
ges (from yfinance) (2.1.0)
Requirement already satisfied: requests>=2.31 in /opt/conda/lib/python3.11/site-pack
ages (from yfinance) (2.31.0)
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.11-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: lxml>=4.9.1 in /opt/conda/lib/python3.11/site-package
s (from yfinance) (5.3.0)
Requirement already satisfied: platformdirs>=2.0.0 in /opt/conda/lib/python3.11/site
-packages (from yfinance) (4.2.1)
Requirement already satisfied: pytz>=2022.5 in /opt/conda/lib/python3.11/site-packag
es (from yfinance) (2024.1)
Collecting frozendict>=2.3.4 (from yfinance)
  Downloading frozendict-2.4.4-py311-none-any.whl.metadata (23 kB)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.17.6.tar.gz (3.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.0/3.0 MB 78.1 MB/s eta 0:00:00:00:01
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: beautifulsoup4>=4.11.1 in /opt/conda/lib/python3.11/s
ite-packages (from yfinance) (4.12.3)
Requirement already satisfied: html5lib>=1.1 in /opt/conda/lib/python3.11/site-packa
ges (from yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.11/site-packa
ges (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: six>=1.9 in /opt/conda/lib/python3.11/site-packages
(from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.11/site-packag
es (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.11/s
ite-packages (from pandas>=1.3.0->yfinance) (2.9.0)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.11/site-pack
ages (from pandas>=1.3.0->yfinance) (2024.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python3.1
1/site-packages (from requests>=2.31->yfinance) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.11/site-packag
es (from requests>=2.31->yfinance) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.11/site-
packages (from requests>=2.31->yfinance) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.11/site-
packages (from requests>=2.31->yfinance) (2024.6.2)
Downloading yfinance-0.2.43-py2.py3-none-any.whl (84 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.6/84.6 kB 11.4 MB/s eta 0:00:00
Downloading frozendict-2.4.4-py311-none-any.whl (16 kB)
Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Building wheels for collected packages: peewee
  Building wheel for peewee (pyproject.toml) ... done
  Created wheel for peewee: filename=peewee-3.17.6-py3-none-any.whl size=138890 sha2
56=ec8a22afa142e8f2de0ebdd63b51ab76ccf451944edde1daab9d658667241a3d
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/1c/09/7e/9f659fde248ecdc17
22a142c1d744271aad3914a0afc191058
```

```
Successfully built peewee
Installing collected packages: peewee, multitasking, frozendict, yfinance
Successfully installed frozendict-2.4.4 multitasking-0.0.11 peewee-3.17.6 yfinance-
0.2.43
Requirement already satisfied: bs4 in /opt/conda/lib/python3.11/site-packages (0.0.
2)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.11/site-pack
ages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.11/site-packa
ges (from beautifulsoup4->bs4) (2.5)
Requirement already satisfied: nbformat in /opt/conda/lib/python3.11/site-packages
(5.10.4)
Requirement already satisfied: fastjsonschema>=2.15 in /opt/conda/lib/python3.11/sit
e-packages (from nbformat) (2.19.1)
Requirement already satisfied: jsonschema>=2.6 in /opt/conda/lib/python3.11/site-pac
kages (from nbformat) (4.22.0)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in /opt/conda/lib/python3.
11/site-packages (from nbformat) (5.7.2)
Requirement already satisfied: traitlets>=5.1 in /opt/conda/lib/python3.11/site-pack
ages (from nbformat) (5.14.3)
Requirement already satisfied: attrs>=22.2.0 in /opt/conda/lib/python3.11/site-packa
ges (from jsonschema>=2.6->nbformat) (23.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /opt/conda/li
b/python3.11/site-packages (from jsonschema>=2.6->nbformat) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in /opt/conda/lib/python3.11/site
-packages (from jsonschema>=2.6->nbformat) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /opt/conda/lib/python3.11/site-pack
ages (from jsonschema>=2.6->nbformat) (0.18.0)
Requirement already satisfied: platformdirs>=2.5 in /opt/conda/lib/python3.11/site-p
ackages (from jupyter-core!=5.0.*,>=4.12->nbformat) (4.2.1)
```

In [3]:
```
import yfinance as yf
import pandas as pd
import requests
from bs4 import BeautifulSoup
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

In Python, you can ignore warnings using the warnings module. You can use the filterwarnings function to filter or ignore specific warning messages or categories.

In [4]:
```
import warnings
# Ignore all warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

Define Graphing Function

In this section, we define the function `make_graph`. **You don't have to know how the function works, you should only care about the inputs. It takes a dataframe with stock data (dataframe must contain Date and Close columns), a dataframe with revenue data (dataframe must contain Date and Revenue columns), and the name of the stock.**

In [5]:
```
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Histori
    stock_data_specific = stock_data[stock_data.Date <= '2021--06-14']
    revenue_data_specific = revenue_data[revenue_data.Date <= '2021-04-30']
    fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific.Date), y=stock_da
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date), y=revenu
    fig.update_xaxes(title_text="Date", row=1, col=1)
    fig.update_xaxes(title_text="Date", row=2, col=1)
    fig.update_yaxes(title_text="Price ($US)", row=1, col=1)
    fig.update_yaxes(title_text="Revenue ($US Millions)", row=2, col=1)
    fig.update_layout(showlegend=False,
    height=900,
    title=stock,
    xaxis_rangeslider_visible=True)
    fig.show()
```

Use the make_graph function that we've already defined. You'll need to invoke it in questions 5 and 6 to display the graphs and create the dashboard.

> **Note: You don't need to redefine the function for plotting graphs anywhere else in this notebook; just use the existing function.**

Question 1: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is Tesla and its ticker symbol is `TSLA`.

In [13]:
```
tesla = yf.Ticker('TSLA')
```

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `tesla_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

In [27]:
```
tesla_data = tesla.history(period='max')
#Show results before reset index
tesla_data.head()
```

Out[27]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2010-06-29 00:00:00-04:00	1.266667	1.666667	1.169333	1.592667	281494500	0.0	0.0
2010-06-30 00:00:00-04:00	1.719333	2.028000	1.553333	1.588667	257806500	0.0	0.0
2010-07-01 00:00:00-04:00	1.666667	1.728000	1.351333	1.464000	123282000	0.0	0.0
2010-07-02 00:00:00-04:00	1.533333	1.540000	1.247333	1.280000	77097000	0.0	0.0
2010-07-06 00:00:00-04:00	1.333333	1.333333	1.055333	1.074000	103003500	0.0	0.0

Reset the index using the `reset_index(inplace=True)` function on the tesla_data DataFrame and display the first five rows of the `tesla_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 1 to the results below.

In [28]:
```
tesla_data.reset_index(inplace=True)
tesla_data.head()
```

Out[28]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2010-06-29 00:00:00-04:00	1.266667	1.666667	1.169333	1.592667	281494500	0.0	0.0
1	2010-06-30 00:00:00-04:00	1.719333	2.028000	1.553333	1.588667	257806500	0.0	0.0
2	2010-07-01 00:00:00-04:00	1.666667	1.728000	1.351333	1.464000	123282000	0.0	0.0
3	2010-07-02 00:00:00-04:00	1.533333	1.540000	1.247333	1.280000	77097000	0.0	0.0
4	2010-07-06 00:00:00-04:00	1.333333	1.333333	1.055333	1.074000	103003500	0.0	0.0

Question 2: Use Webscraping to Extract Tesla Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm Save the text of the response as a variable named `html_data`.

In [36]:
```
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDevelo
response = requests.get(url)
html_data = response.text
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`. Make sure to use the `html_data` with the content parameter as follow `html_data.content`.

In [40]:
```
soup = BeautifulSoup(response.content,'html.parser')
```

Using `BeautifulSoup` or the `read_html` function extract the table with `Tesla Revenue` and store it into a dataframe named `tesla_revenue`. The dataframe should have columns `Date` and `Revenue`.

In [42]:
```
#find all html tables in the web page
tables = soup.find_all('table')
#see how many tables were found by checking the length of the tables list
len(tables)
```

Out[42]: 6

In [45]:
```
#search for the table Tesla Revenue
for index,table in enumerate(tables):
     if ("Tesla Quarterly Revenue" in str(table)):
         table_index = index
print(table_index)
```

1

In [61]:
```
#so Tesla Revenue is table with index 1
```

In [103…
```
#return list of all dataframes inside table with index 1
frames = pd.read_html(str(tables[1]), flavor='bs4')
```

In [104…
```
# we see that data frame with index 0 is the correct data frame
tesla_revenue_frame = frames[0]
```

In [93]:
```
#now we create empty frame with columns date and revenue
tesla_revenue = pd.DataFrame()
tesla_revenue['Date']=[]
```

```
tesla_revenue['Revenue']=[]
tesla_revenue
```

Out[93]:

	Date	Revenue

In [105…
```
tesla_revenue = tesla_revenue_frame
tesla_revenue.columns=['Date','Revenue']
```

▶ Step-by-step instructions

▶ Click here if you need help locating the table

Execute the following line to remove the comma and dollar sign from the `Revenue` column.

In [106…
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"", regex=Tr
```

Execute the following lines to remove an null or empty strings in the Revenue column.

In [107…
```
tesla_revenue.dropna(inplace=True)

tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
```

Display the last 5 row of the `tesla_revenue` dataframe using the `tail` function. Take a screenshot of the results.

In [146…
```
tesla_revenue.tail()
```

Out[146…

	Date	Revenue
48	2010-09-30	31
49	2010-06-30	28
50	2010-03-31	21
52	2009-09-30	46
53	2009-06-30	27

Question 3: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is GameStop and its ticker symbol is `GME`.

In [111…
```
gamestop = yf.Ticker('GME')
```

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `gme_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

In [118…
```
gme_data = gamestop.history(period='max')
gme_data.head()
```

Out[118…

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2002-02-13 00:00:00-05:00	1.620129	1.693350	1.603296	1.691667	76216000	0.0	0.0
2002-02-14 00:00:00-05:00	1.712707	1.716074	1.670626	1.683250	11021600	0.0	0.0
2002-02-15 00:00:00-05:00	1.683250	1.687458	1.658001	1.674834	8389600	0.0	0.0
2002-02-19 00:00:00-05:00	1.666418	1.666418	1.578047	1.607504	7410400	0.0	0.0
2002-02-20 00:00:00-05:00	1.615921	1.662210	1.603296	1.662210	6892800	0.0	0.0

Reset the index using the `reset_index(inplace=True)` function on the gme_data DataFrame and display the first five rows of the `gme_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 3 to the results below.

In [119…
```
gme_data.reset_index(inplace=True)
gme_data.head()
```

Out[119…

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-02-13 00:00:00-05:00	1.620129	1.693350	1.603296	1.691667	76216000	0.0	0.0
1	2002-02-14 00:00:00-05:00	1.712707	1.716074	1.670626	1.683250	11021600	0.0	0.0
2	2002-02-15 00:00:00-05:00	1.683250	1.687458	1.658001	1.674834	8389600	0.0	0.0
3	2002-02-19 00:00:00-05:00	1.666418	1.666418	1.578047	1.607504	7410400	0.0	0.0
4	2002-02-20 00:00:00-05:00	1.615921	1.662210	1.603296	1.662210	6892800	0.0	0.0

Question 4: Use Webscraping to Extract GME Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html. Save the text of the response as a variable named `html_data_2`.

In [120…
```
url4 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDevel
html_data_2 = requests.get(url4).text
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`.

In [125…
```
soup4 = BeautifulSoup(html_data_2, 'html.parser')
```

Using `BeautifulSoup` or the `read_html` function extract the table with `GameStop Revenue` and store it into a dataframe named `gme_revenue`. The dataframe should have columns `Date` and `Revenue`. Make sure the comma and dollar sign is removed from the `Revenue` column.

In [139…
```
#find all html tables in the web page
tables = soup4.find_all('table')
#see how many tables were found by checking the length of the tables list
len(tables)
#len function returns 6
#search for the table GameStop Revenue
for index,table in enumerate(tables):
     if ("GameStop%Revenue" in str(table)):
         table_index = index
print(table_index)
```

```
1
```

In [140…
```
frames = pd.read_html(str(tables[1]), flavor='bs4')
gme_revenue_frame = frames[0]
#now we create empty frame with columns date and revenue
gme_revenue = pd.DataFrame()
gme_revenue['Date']=[]
gme_revenue['Revenue']=[]
gme_revenue
gme_revenue = gme_revenue_frame
gme_revenue.columns=['Date','Revenue']
gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',|\$',"", regex=True)
```

> **Note: Use the method similar to what you did in question 2.**

▶ Click here if you need help locating the table

Display the last five rows of the `gme_revenue` dataframe using the `tail` function. Take a screenshot of the results.

In [141…
```
gme_revenue.tail()
```

Out[141…

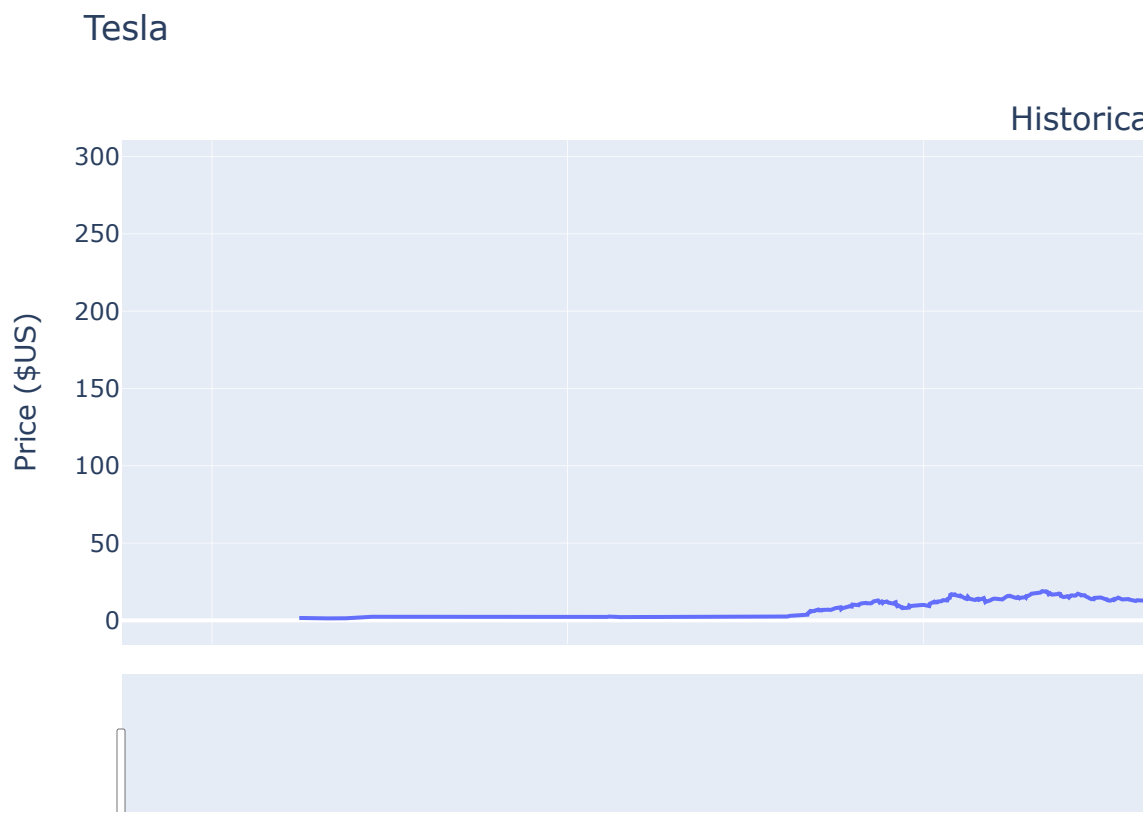
	Date	Revenue
57	2006-01-31	1667
58	2005-10-31	534
59	2005-07-31	416
60	2005-04-30	475
61	2005-01-31	709

Question 5: Plot Tesla Stock Graph

Use the `make_graph` function to graph the Tesla Stock Data, also provide a title for the graph. Note the graph will only show data upto June 2021.

▶ Hint

In [157…
```
make_graph(tesla_data, tesla_revenue, 'Tesla')
```



In [143…

```
!pip install matplotlib
```

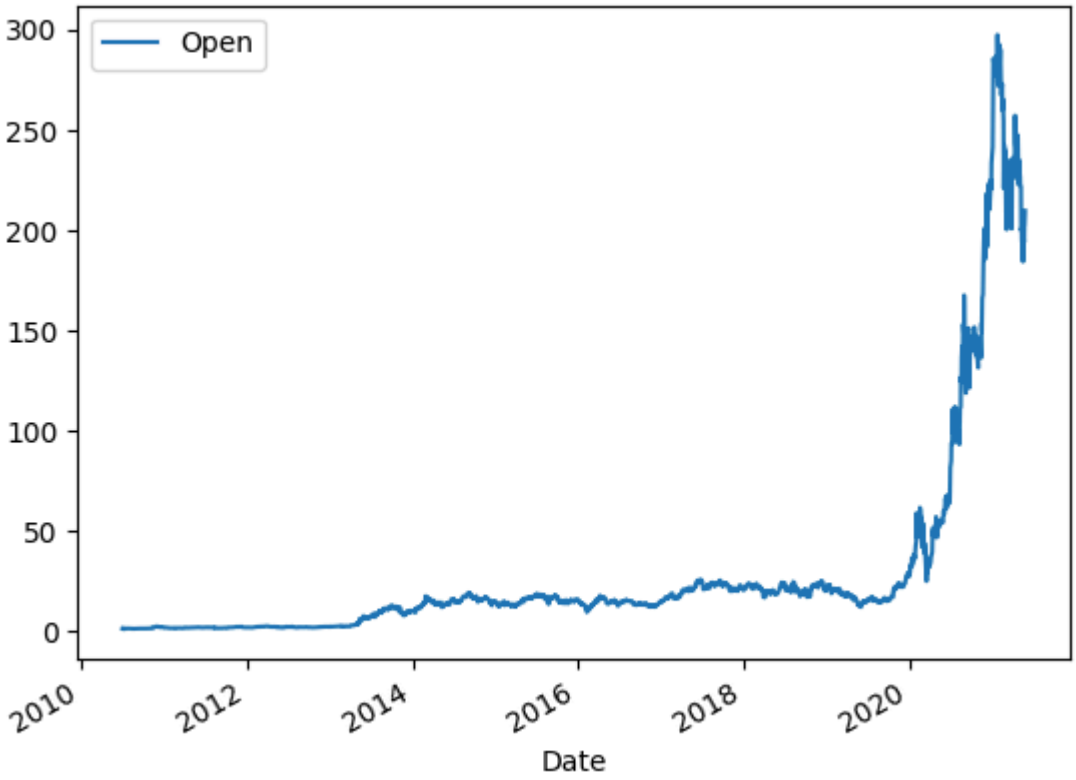
```
Collecting matplotlib
  Downloading matplotlib-3.9.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl.metadata (5.4 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.53.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl.metadata (162 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 162.6/162.6 kB 14.7 MB/s eta 0:00:00
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl.metadata (6.4 kB)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.11/site-package
s (from matplotlib) (2.1.0)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.11/site-pac
kages (from matplotlib) (24.0)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-10.4.0-cp311-cp311-manylinux_2_28_x86_64.whl.metadata (9.2 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.1.4-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.11/sit
e-packages (from matplotlib) (2.9.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.11/site-packages
(from python-dateutil>=2.7->matplotlib) (1.16.0)
Downloading matplotlib-3.9.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.
whl (8.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.3/8.3 MB 99.4 MB/s eta 0:00:00:00:010
0:01
Downloading contourpy-1.3.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.w
hl (323 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 323.2/323.2 kB 33.2 MB/s eta 0:00:00
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.53.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.
whl (4.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.9/4.9 MB 122.0 MB/s eta 0:00:0000:01
Downloading kiwisolver-1.4.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.
whl (1.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 75.5 MB/s eta 0:00:00
Downloading pillow-10.4.0-cp311-cp311-manylinux_2_28_x86_64.whl (4.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.5/4.5 MB 110.5 MB/s eta 0:00:0000:01
Downloading pyparsing-3.1.4-py3-none-any.whl (104 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 104.1/104.1 kB 12.9 MB/s eta 0:00:00
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler, con
tourpy, matplotlib
Successfully installed contourpy-1.3.0 cycler-0.12.1 fonttools-4.53.1 kiwisolver-1.
4.5 matplotlib-3.9.2 pillow-10.4.0 pyparsing-3.1.4
```

In [156…

```python
#limit graph to June 2021
tesla_data=tesla_data[tesla_data['Date']<'2021-06-01']
tesla_data.plot(x="Date", y="Open")
```

Out[156…

```
<Axes: xlabel='Date'>
```



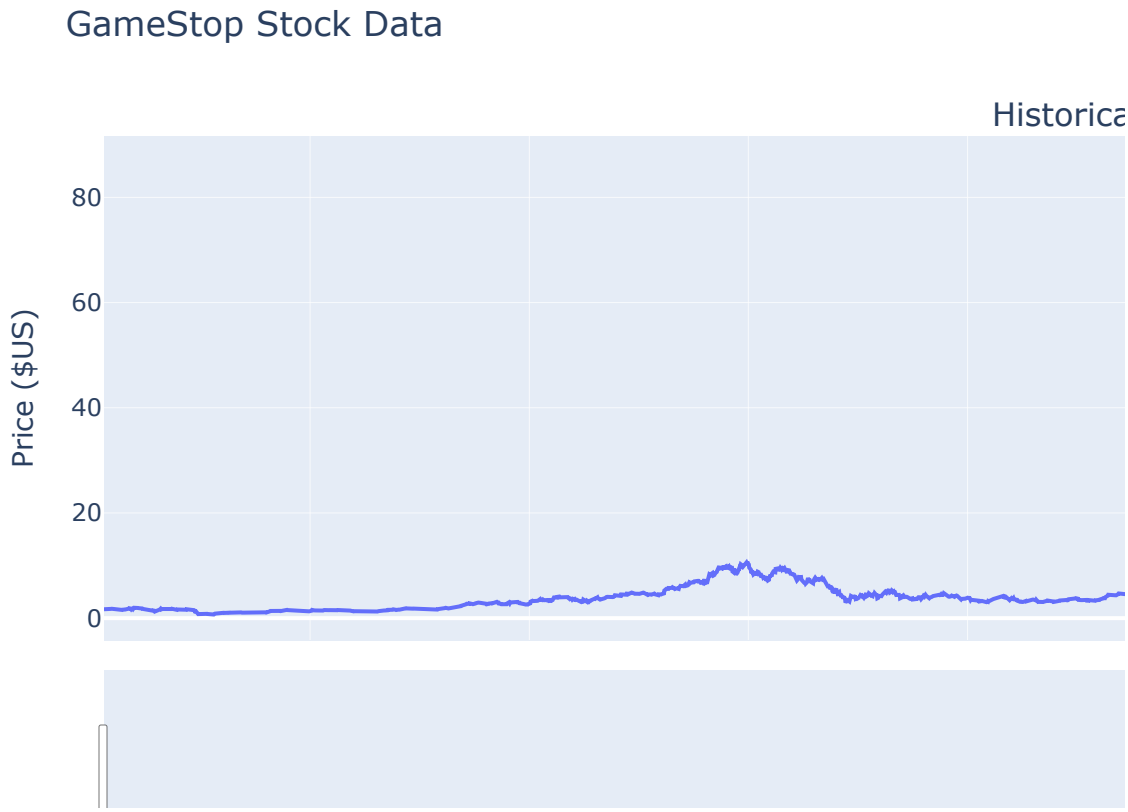
Question 6: Plot GameStop Stock Graph

Use the `make_graph` function to graph the GameStop Stock Data, also provide a title for the graph. The structure to call the `make_graph` function is `make_graph(gme_data, gme_revenue, 'GameStop')`. Note the graph will only show data upto June 2021.

▶ Hint

In [159…

```
make_graph(gme_data, gme_revenue, 'GameStop Stock Data')
```

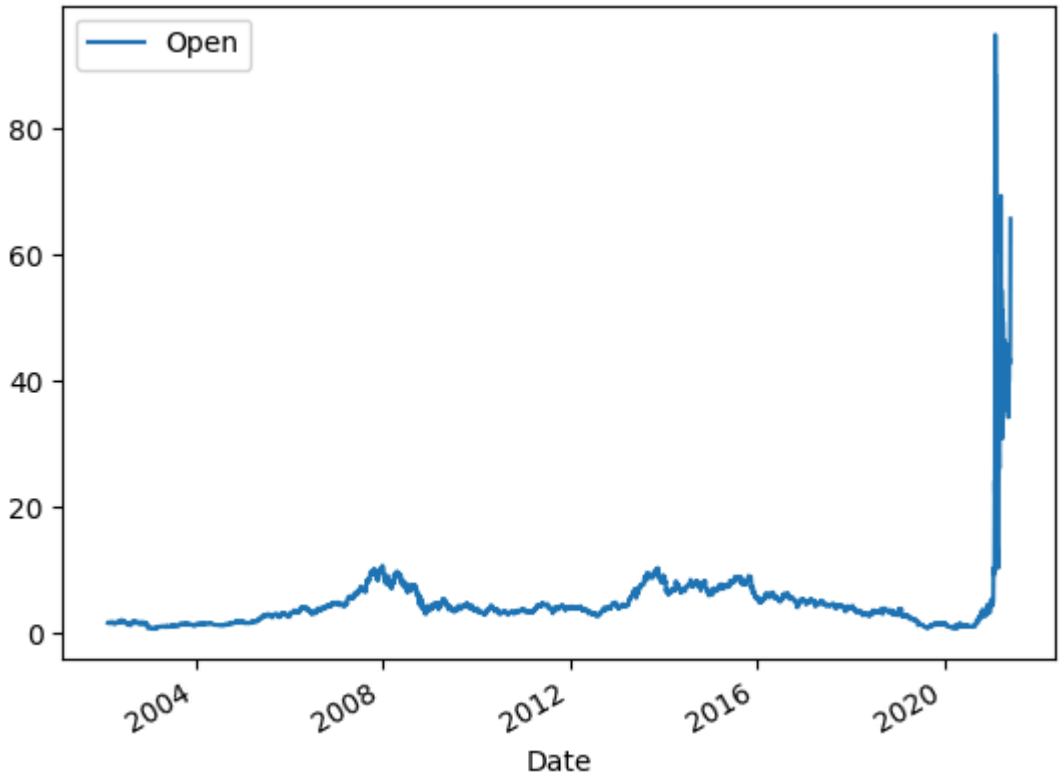


In [160…

```
#limit graph to June 2021
gme_data=gme_data[gme_data['Date']<'2021-06-01']
```

```
gme_data.plot(x="Date", y="Open")
```

Out[160… <Axes: xlabel='Date'>



About the Authors:

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

toggle

toggle

toggle

toggle

toggle

toggle