Extracting & Visualizing Stock Data - Tesla (TSLA) and GameStop (GME)

Description

In ths notebook, I will take historical stock information of Tesla and GameStop and plot it on a graph.

Setting up my environment

Note: setting up my envirnoment by installing packages and loading libraries.

```
!pip install yfinance
#!pip install pandas
#!pip install requests
!pip install bs4
#!pip install plotly

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
```

```
Requirement already satisfied: yfinance in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.10/dist
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.10/dist
Requirement already satisfied: requests>=2.31 in /usr/local/lib/python3.10/dis
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.1
Requirement already satisfied: lxml>=4.9.1 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: platformdirs>=2.0.0 in /usr/local/lib/python3.1
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: frozendict>=2.3.4 in /usr/local/lib/python3.10/
Requirement already satisfied: peewee>=3.16.2 in /usr/local/lib/python3.10/dis
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/python
Requirement already satisfied: html5lib>=1.1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dis
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyth
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10
Collecting bs4
  Downloading bs4-0.0.2-py2.py3-none-any.whl (1.2 kB)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dis
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist
Installing collected packages: bs4
Successfully installed bs4-0.0.2
```

Here, I'll define the function make_graph using a dataframe containing information about the stock, a dataframe containing information about the revenue, and the stock's name.

```
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Hi
    fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data.Date, infer_datetime_
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data.Date, infer_datetim
    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()
```

Using yfinance to Extract Tesla Historical Stock Data

Let's use the Ticker function to build a ticker object by entering the ticker symbol of the stock from which we wish to collect data. Tesla is the company's stock, and TSLA is its ticker.

```
tesla = yf.Ticker("TSLA")
```

Stock data will be extracted and saved in a dataframe called tesla_data using the ticker object and the function history. To obtain data for the longest possible period of time, we shall set the period option to max.

```
tesla_data = tesla.history(period="max")
```

The first five rows of the tesla_data dataframe will be displayed after we reset the index on the dataframe.

```
tesla_data.reset_index(inplace=True)
tesla_data.head()
```

	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

Next steps: Generate code with tesla_data View recommended plots

Using Webscraping to Extract Tesla Revenue Data

Using the requests library to download the webpage https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue and save the text of the response as a variable named html_data.

```
url_tsla = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud
html_data  = requests.get(url_tsla).text
```

We will Parse the html data using beautiful soup

```
soup_tsla = BeautifulSoup(html_data,"html5lib")
```

Using BeautifulSoup function, we will extract the table with Tesla Quarterly Revenue and store it into a dataframe named tesla_revenue showing the first 5 rows and removing the comma and dollar sign from the Revenue column.

```
tesla_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

for table in soup_tsla.find_all('table'):

    if ('Tesla Quarterly Revenue' in table.find('th').text):
        rows = table.find_all('tr')

        for row in rows:
            col = row.find_all('td')

            if col != []:
                date = col[0].text
                revenue = col[1].text.replace("$", "").replace(',','').replace
                tesla_revenue = pd.concat([tesla_revenue, pd.DataFrame({"Date"
```

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

```
tesla_revenue.dropna(inplace=True)

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

Displaying the last 5 row of the tesla_revenue dataframe using the tail function

```
tesla_revenue.tail()
```

	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

Using yfinance to Extract GameStop Historical Stock Data

Again, let's use the Ticker function to enter the ticker symbol of GameStop and its ticker symbol is GME.

```
gamestop=yf.Ticker("GME")
```

Let's extract and save the gamestop data into a dataframe called gme_data using the ticker object and function history.

```
gme_data=gamestop.history(period="max")
```

As before, we will reset the index on the gamestop dataframe and display the first five rows.

```
gme_data.reset_index(inplace=True)
gme_data.head()
```

	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	1.683251	11021600	0.0	0.0

Next steps: Generate code with gme_data View recommended plots

Using Webscraping to Extract GameStop Revenue Data

Using 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 and save the text of the response as a variable named html_data.

```
gme_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/

html_data  = requests.get(gme_url).text
```

Now we will Parse the html data using beautiful soup

```
soup_gme = BeautifulSoup(html_data,"html5lib")
```

Using BeautifulSoup function, we will extract the table with GameStop Quarterly Revenue and store it into a dataframe named gme_revenue while removing the comma and dollar sign from the Revenue column.

```
gme_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

for table in soup_gme.find_all('table'):

    if ('GameStop Quarterly Revenue' in table.find('th').text):
        rows = table.find_all('tr')

        for row in rows:
            col = row.find_all('td')

            if col != []:
                date = col[0].text
                revenue = col[1].text.replace(',','').replace("$", "").replace


                gme_revenue = pd.concat([gme_revenue, pd.DataFrame({"Date":[da
```

Again, Executing the following lines to remove null or empty strings in the Revenue column.

```
gme_revenue.dropna(inplace=True)
gme_revenue.tail()
```

	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

Here we will use the make_graph function to graph the Tesla Stock Data.

Note the graph will only show data upto June 2021.

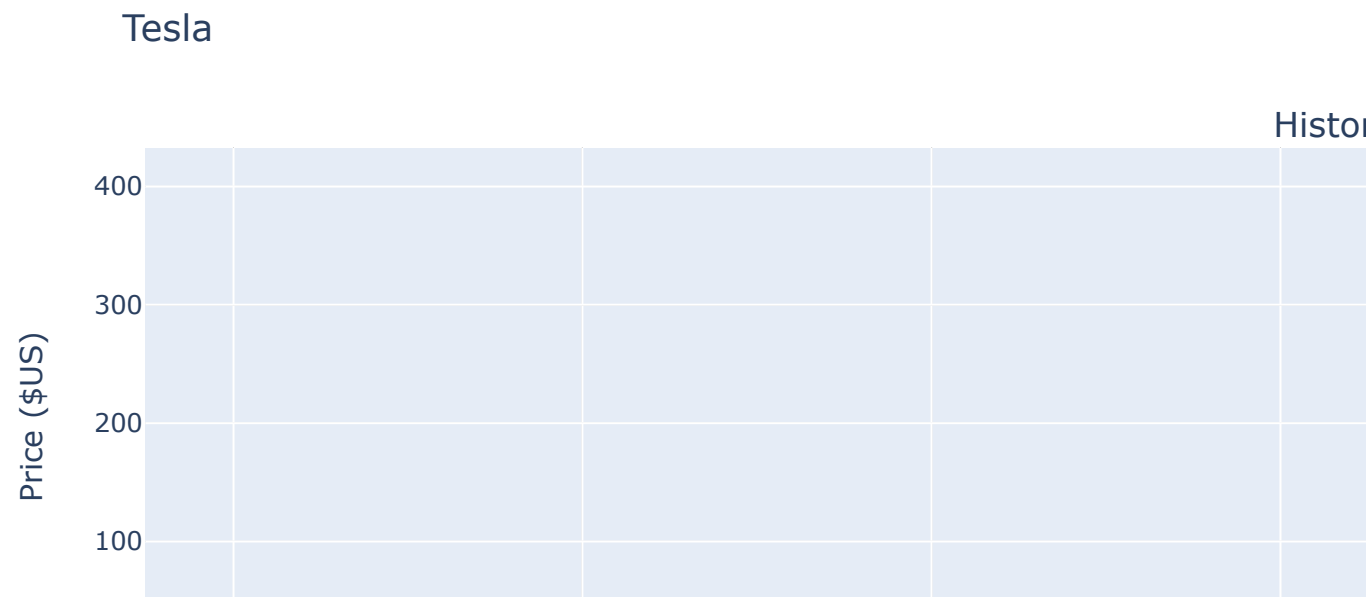
```
make_graph(tesla_data, tesla_revenue, 'Tesla')
```

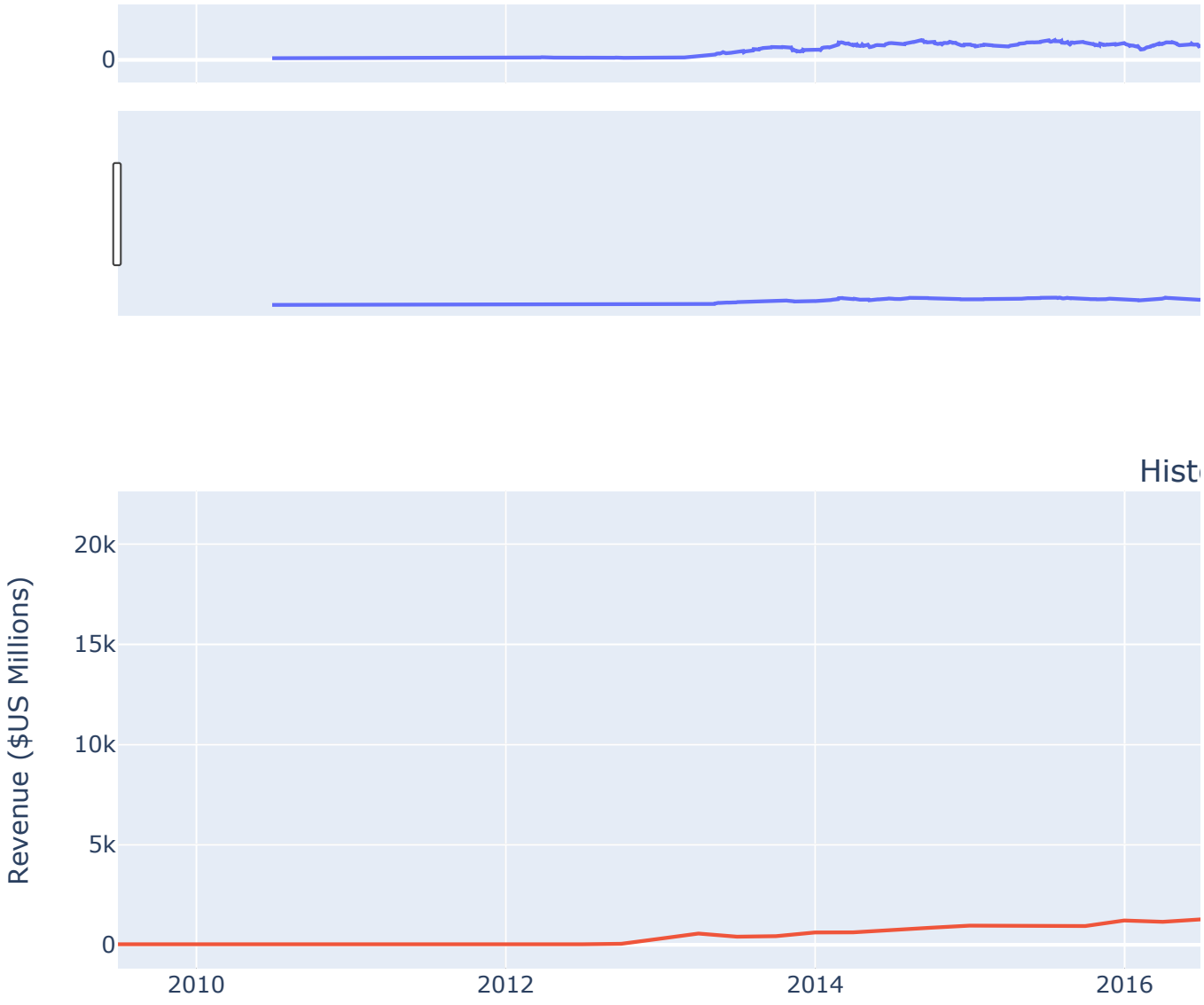
```
<ipython-input-2-1df015be5a96>:3: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a fu

<ipython-input-2-1df015be5a96>:4: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a fu
```





Here too, we will use the make_graph function to graph the GameStop Stock Data.

Note the graph will only show data upto June 2021.

```
1 make_graph(gme_data, gme_revenue, 'GameStop')
```

```
<ipython-input-2-1df015be5a96>:3: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a fu

<ipython-input-2-1df015be5a96>:4: UserWarning:
```

```
The argument 'infer_datetime_format' is deprecated and will be removed in a fu
```