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.

In [1]:

```
!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
```

```
Collecting yfinance
  Downloading yfinance-0.1.74-py2.py3-none-any.whl (27 kB)
Requirement already satisfied: requests>=2.26 in /opt/conda/lib/python
3.7/site-packages (from yfinance) (2.28.1)
Requirement already satisfied: lxml>=4.5.1 in /opt/conda/lib/python3.
7/site-packages (from yfinance) (4.9.1)
Requirement already satisfied: pandas>=0.24.0 in /opt/conda/lib/python
3.7/site-packages (from yfinance) (1.3.5)
Requirement already satisfied: numpy>=1.15 in /opt/conda/lib/python3.
7/site-packages (from yfinance) (1.21.6)
Collecting multitasking>=0.0.7
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/li
b/python3.7/site-packages (from pandas>=0.24.0->yfinance) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.
7/site-packages (from pandas>=0.24.0->yfinance) (2022.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/li
b/python3.7/site-packages (from requests>=2.26->yfinance) (1.26.11)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.
7/site-packages (from requests>=2.26->yfinance) (3.3)
Requirement already satisfied: charset-normalizer<3,>=2 in /opt/conda/
lib/python3.7/site-packages (from requests>=2.26->yfinance) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/py
thon3.7/site-packages (from requests>=2.26->yfinance) (2022.6.15)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/si
te-packages (from python-dateutil>=2.7.3->pandas>=0.24.0->yfinance)
(1.15.0)
Installing collected packages: multitasking, yfinance
Successfully installed multitasking-0.0.11 yfinance-0.1.74
WARNING: Running pip as the 'root' user can result in broken permissio
ns and conflicting behaviour with the system package manager. It is re
commended to use a virtual environment instead: https://pip.pypa.io/wa
rnings/venv
Collecting bs4
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
  Preparing metadata (setup.py) ... -   done
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python
3.7/site-packages (from bs4) (4.11.1)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python
3.7/site-packages (from beautifulsoup4->bs4) (2.3.1)
```

```
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py) ... -   \   done
  Created wheel for bs4: filename=bs4-0.0.1-py3-none-any.whl size=1272
sha256=59e264c0d5f5d80a9808947248907ea4562182fc5fbbb6fcafd5f5c844e442f
0
  Stored in directory: /root/.cache/pip/wheels/0a/9e/ba/20e5bbc1afef3a
491f0b3bb74d508f99403aabe76eda2167ca
Successfully built bs4
Installing collected packages: bs4
Successfully installed bs4-0.0.1
WARNING: Running pip as the 'root' user can result in broken permissio
ns and conflicting behaviour with the system package manager. It is re
commended to use a virtual environment instead: https://pip.pypa.io/wa
rnings/venv
```

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.

In [2]:

```python
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles
=("Historical Share Price", "Historical Revenue"), vertical_spacing = .3)
    fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data.Date, infer_date
time_format=True), y=stock_data.Close.astype("float"), name="Share Pric
e"), row=1, col=1)
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data.Date, infer_da
tetime_format=True), y=revenue_data.Revenue.astype("float"), name="Revenu
e"), row=2, col=1)
    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.

In [3]:
```
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.

In [4]:
```
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.

In [5]:
```
tesla_data.reset_index(inplace=True)
tesla_data.head()
```

Out[5]:

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

Using Webscraping to Extract Tesla Revenue Data

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

In [6]:
```
url= "https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkPY0220ENSkillsNetwork23455606-2022-01-01"
html_data=requests.get(url).text
```

We will Parse the html data using beautiful soup

In [7]:
```
soup = 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.

In [8]:

```
tesla_revenue= pd.read_html(url, match="Tesla Quarterly Revenue", flavor
='bs4')[0]
tesla_revenue=tesla_revenue.rename(columns = {'Tesla Quarterly Revenue(Mi
llions of US $)': 'Date', 'Tesla Quarterly Revenue(Millions of US $).1':
'Revenue'}, inplace = False)
tesla_revenue["Revenue"] = tesla_revenue["Revenue"].str.replace(",","").s
tr.replace("$","")
tesla_revenue.head()
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:3: Future
Warning: The default value of regex will change from True to False in
a future version. In addition, single character regular expressions wi
ll *not* be treated as literal strings when regex=True.
  This is separate from the ipykernel package so we can avoid doing im
ports until
```

Out[8]:

	Date	Revenue
0	2022-06-30	16934
1	2022-03-31	18756
2	2021-12-31	17719
3	2021-09-30	13757
4	2021-06-30	11958

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

In [9]:

```
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

In [10]:
```
tesla_revenue.tail()
```

Out[10]:

	Date	Revenue
47	2010-09-30	31
48	2010-06-30	28
49	2010-03-31	21
51	2009-09-30	46
52	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.

In [11]:
```
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.

In [12]:
```
gme_data=gamestop.history(period="max")
```

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

In [13]:
```
gme_data.reset_index(inplace=True)
gme_data.head()
```

Out[13]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-02-13	1.620128	1.693350	1.603296	1.691666	76216000	0.0	0.0
1	2002-02-14	1.712707	1.716074	1.670626	1.683250	11021600	0.0	0.0
2	2002-02-15	1.683250	1.687458	1.658002	1.674834	8389600	0.0	0.0
3	2002-02-19	1.666418	1.666418	1.578047	1.607504	7410400	0.0	0.0
4	2002-02-20	1.615920	1.662209	1.603295	1.662209	6892800	0.0	0.0

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 (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.

In [14]:
```
url="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/I
BMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"
html_data=requests.get(url).text
```

Now we will Parse the html data using beautiful soup

In [15]:
```
soup = 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.

In [16]:

```
gme_revenue= pd.read_html(url, match="GameStop Quarterly Revenue", flavor
='bs4')[0]
gme_revenue=gme_revenue.rename(columns = {'GameStop Quarterly Revenue(Mil
lions of US $)': 'Date', 'GameStop Quarterly Revenue(Millions of US $).
1': 'Revenue'}, inplace = False)
gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace(",","").str.r
eplace("$","")
gme_revenue.head()
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:3: Future
Warning: The default value of regex will change from True to False in
a future version. In addition, single character regular expressions wi
ll *not* be treated as literal strings when regex=True.
  This is separate from the ipykernel package so we can avoid doing im
ports until
```

Out[16]:

	Date	Revenue
0	2020-04-30	1021
1	2020-01-31	2194
2	2019-10-31	1439
3	2019-07-31	1286
4	2019-04-30	1548

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

In [17]:
```
gme_revenue.dropna(inplace=True)
gme_revenue.tail()
```

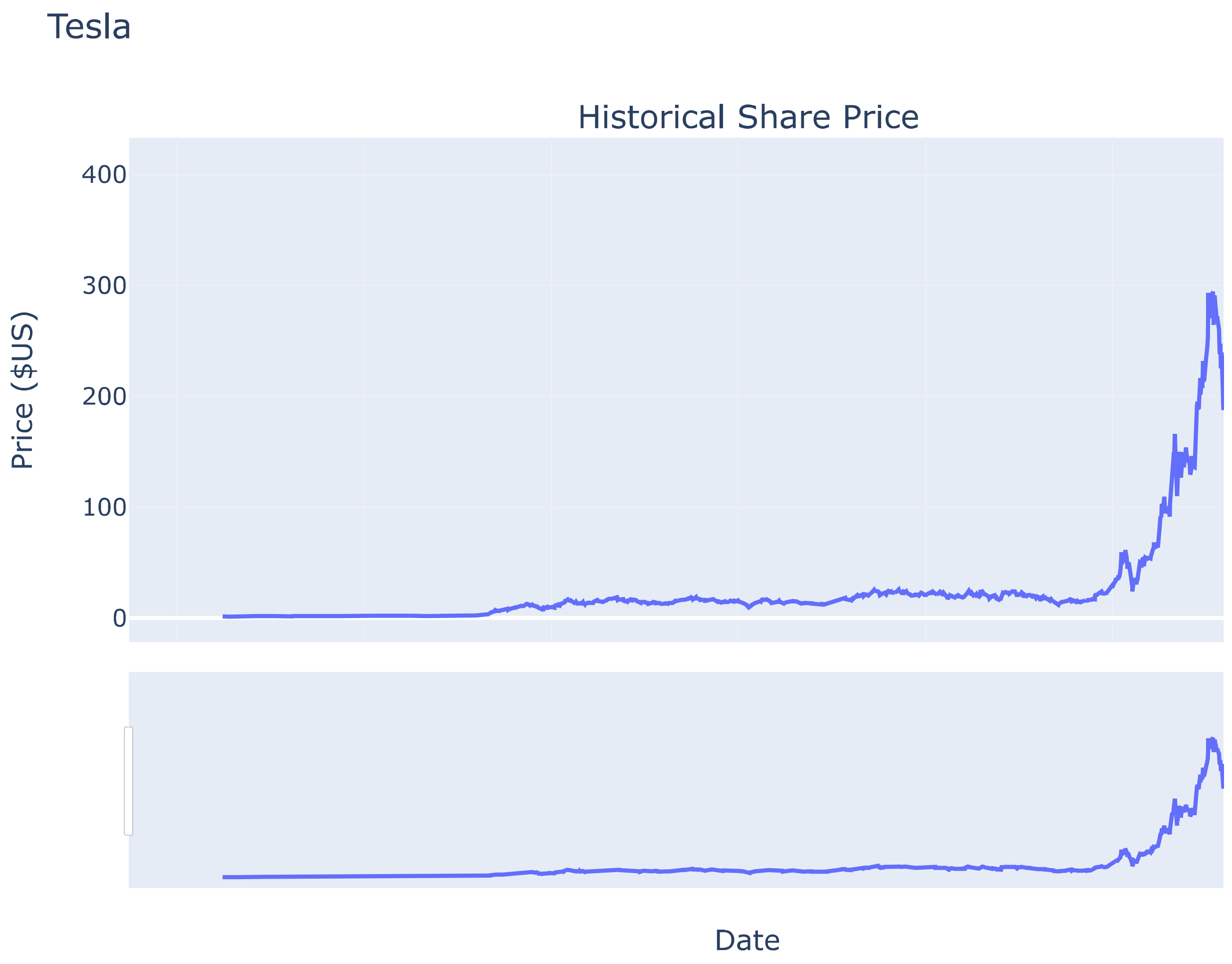
Out[17]:

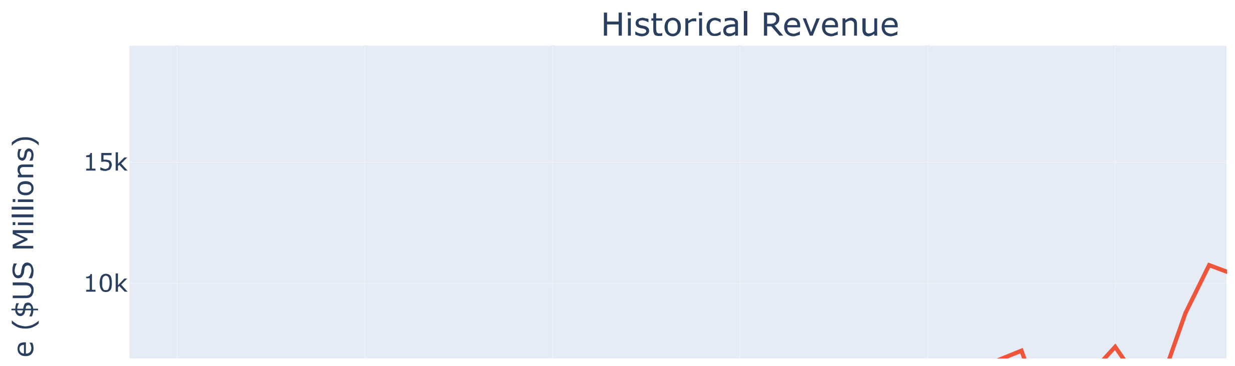
	Date	Revenue
57	2006-01-31	1667
58	2005-10-31	534
59	2005-07-31	416
60	2005-04-30	475
61	2005-01-31	709

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

Note the graph will only show data upto June 2021.

In [18]:
```
make_graph(tesla_data, tesla_revenue, 'Tesla')
```


Tesla
Historical Share Price
Price ($US)
400
300
200
100
0
Date


Historical Revenue
e ($US Millions)
15k
10k

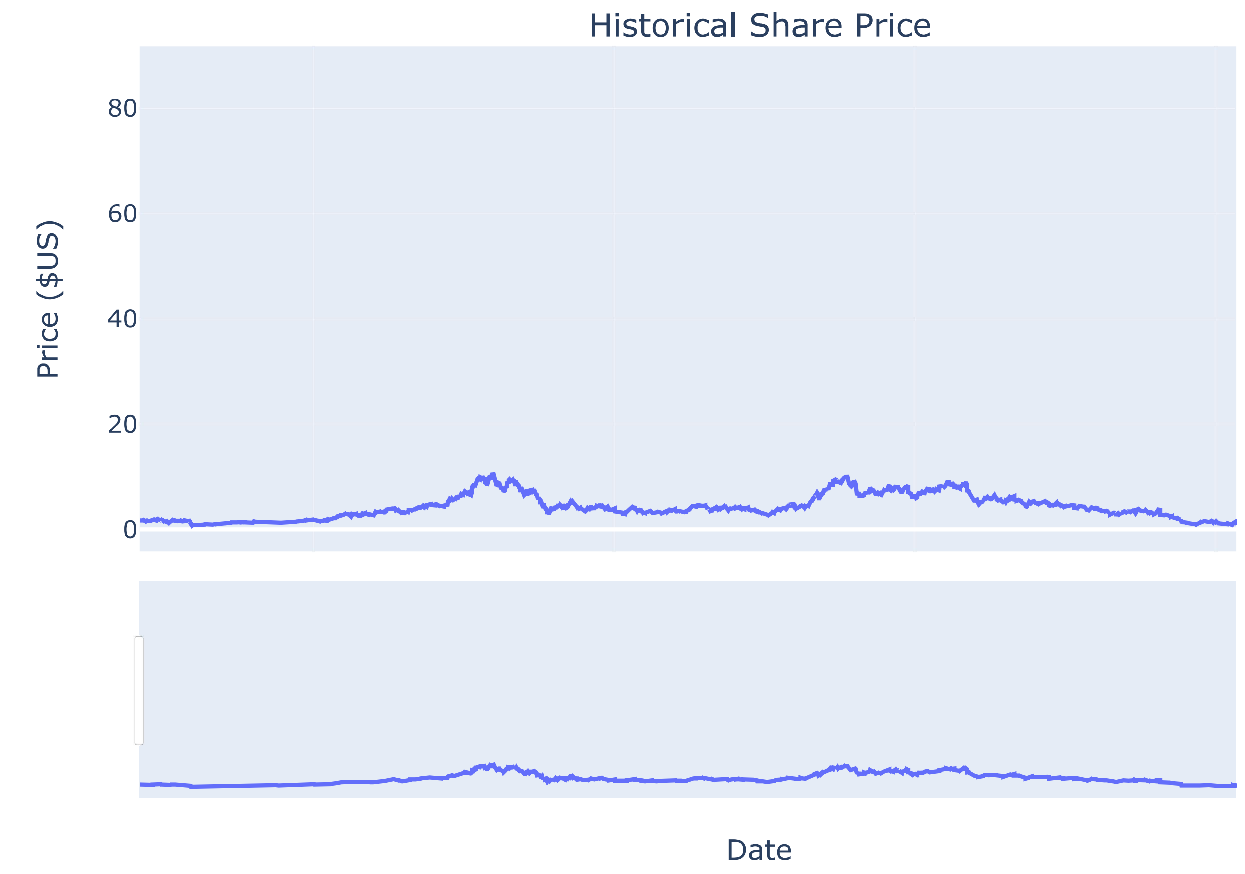
Here too, we will use the make_graph function to graph the GameStop Stock Data.

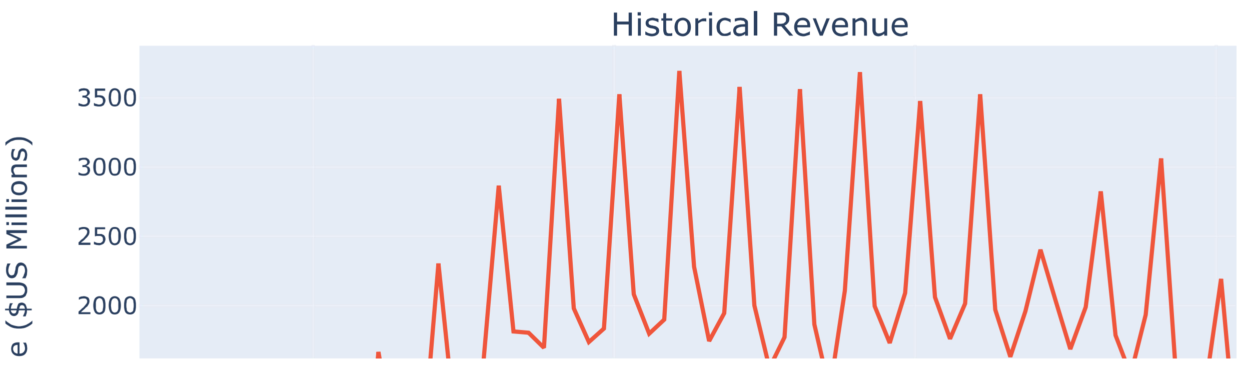
Note the graph will only show data upto June 2021.

In [19]:

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

GameStop


Historical Share Price
Price ($US)
80
60
40
20
0
Date


Historical Revenue
e ($US Millions)
3500
3000
2500
2000