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

In [9]:
```
!pip install yfinance bs4 nbformat
```

```
Requirement already satisfied: yfinance in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (0.2.17)
Collecting bs4
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: nbformat in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (5.5.0)
Requirement already satisfied: requests>=2.26 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from yfina
nce) (2.28.1)
Requirement already satisfied: html5lib>=1.1 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from yfinan
ce) (1.1)
Requirement already satisfied: cryptography>=3.3.2 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from
yfinance) (37.0.1)
Requirement already satisfied: numpy>=1.16.5 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from yfinan
ce) (1.21.5)
Requirement already satisfied: lxml>=4.9.1 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from yfinanc
e) (4.9.1)
Requirement already satisfied: pytz>=2022.5 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from yfinanc
e) (2023.3)
Requirement already satisfied: pandas>=1.3.0 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from yfinan
ce) (2.0.0)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from
yfinance) (0.0.11)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from yf
inance) (2.3.7)
Requirement already satisfied: appdirs>=1.4.4 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from yfina
nce) (1.4.4)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (fr
om yfinance) (4.11.1)
Requirement already satisfied: jsonschema>=2.6 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from nbfo
rmat) (4.16.0)
Requirement already satisfied: fastjsonschema in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from nbfor
mat) (2.16.2)
Requirement already satisfied: jupyter_core in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from nbforma
t) (4.11.1)
Requirement already satisfied: traitlets>=5.1 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from nbfor
mat) (5.1.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from beauti
fulsoup4>=4.11.1->yfinance) (2.3.1)
Requirement already satisfied: cffi>=1.12 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from cryptogra
phy>=3.3.2->yfinance) (1.15.1)
Requirement already satisfied: webencodings in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from html5li
b>=1.1->yfinance) (0.5.1)
Requirement already satisfied: six>=1.9 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from html5lib>=
1.1->yfinance) (1.16.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\users\ravena\anaconda3\anaconda
a\lib\site-packages (from jsonschema>=2.6->nbformat) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from jsonsc
hema>=2.6->nbformat) (21.4.0)
Requirement already satisfied: tzdata>=2022.1 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from panda
s>=1.3.0->yfinance) (2022.7)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (fr
om pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from r
equests>=2.26->yfinance) (2022.12.7)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from request
s>=2.26->yfinance) (3.3)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages
(from requests>=2.26->yfinance) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (fro
m requests>=2.26->yfinance) (1.26.11)
Requirement already satisfied: pywin32>=1.0 in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from jupyter
_core->nbformat) (302)
Requirement already satisfied: pycparser in c:\users\ravena\anaconda3\anacondaa\lib\site-packages (from cffi>=1.12
->cryptography>=3.3.2->yfinance) (2.21)
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py): started
  Building wheel for bs4 (setup.py): finished with status 'done'
  Created wheel for bs4: filename=bs4-0.0.1-py3-none-any.whl size=1257 sha256=24655a9dd9ae70037a88b9fa51de391f30f2
9e6193c570b7bb3db8d64e534d2f
  Stored in directory: c:\users\ravena\appdata\local\pip\cache\wheels\73\2b\cb\099980278a0c9a3e57ff1a89875ec07bfa0
b6fcbebb9a8cad3
Successfully built bs4
Installing collected packages: bs4
Successfully installed bs4-0.0.1
```

In [10]:
```
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
```

**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 [11]:
```
def make_graph(stock_data, revenue_data, stock):
    #stock_data is data frame that must contain Date and Close columns
    # revenue_data is data frame with revenue data must contain Date and Revenue cols
    # stock name, a string.
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Historical Share Price", "Historical Rev
    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, infer_datetime_format=True), y=stock_data_sp
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date, infer_datetime_format=True), y=revenue_dat
    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()
```

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 [12]:
```
tsla_ticker = 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 [13]:
```
tesla_data = tsla_ticker.history(period="max")
```

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 [14]:
```
# type(tesla_data) # this is a pd Dataframe
# print(tesla_data.head())
tesla_data.reset_index(inplace=True)
print(tesla_data.head())
```

```
                       Date      Open      High       Low     Close  \
0 2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667
1 2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667
2 2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000
3 2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000
4 2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000

      Volume  Dividends  Stock Splits
0  281494500        0.0           0.0
1  257806500        0.0           0.0
2  123282000        0.0           0.0
3   77097000        0.0           0.0
4  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 [15]:
```
url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNet
html_data = requests.get(url, "html.parser").text
type(html_data)
```

Out[15]: str

Parse the html data using beautiful_soup.

In [17]:
```
soup = BeautifulSoup(html_data, 'html5lib')
type(soup)
```

Out[17]: bs4.BeautifulSoup

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

In [18]:
```
table_element = soup.find_all("table")[1]
tesla_revenue = pd.read_html(str(table_element))
tesla_revenue = tesla_revenue[0]
tesla_revenue.columns = ['Date', 'Revenue']
#print(tesla_revenue.head())
```

In [ ]:
```
#Execute the following line to remove the comma and dollar sign from the Revenue column
```

In [19]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
#print(tesla_revenue.head())
```

```
C:\Users\Ravena\AppData\Local\Temp\ipykernel_15680\492386151.py:1: FutureWarning: The default value of regex will
change from True to False in a future version.
  tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

In [20]:
```
#Execute the following lines to remove an null or empty strings in the Revenue column.
```

In [21]:
```
tesla_revenue.dropna(inplace=True)

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

In [22]:
```
tesla_revenue.tail()
```

Out[22]:

	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 [24]:
```
gme_ticker = 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 [25]:
```
gme_data = gme_ticker.history(period="max")
#print(gme_data)
```

```
#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 [27]:
```
gme_data.reset_index(inplace=True)
gme_data.head()
```

Out[27]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-02-13 00:00:00-05:00	1.620128	1.693350	1.603296	1.691666	76216000	0.0	0.0
1	2002-02-14 00:00:00-05:00	1.712707	1.716073	1.670626	1.683250	11021600	0.0	0.0
2	2002-02-15 00:00:00-05:00	1.683250	1.687458	1.658002	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	1.615920	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 (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.

In [28]:
```
url2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-Skills
html_data = requests.get(url2).text
```

In [29]:
```
#Parse the html data using beautiful_soup.
```

In [30]:
```
soup = BeautifulSoup(html_data, "html5lib")
```

Using BeautifulSoup or the read_html function extract the table with GameStop Quarterly 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 using a method similar to what you did in Question 2.

In [31]:
```
gme_revenue = pd.read_html(url)[1]
#print(gamestop_revenue.head())
gme_revenue.columns = ["Date", "Revenue"]
gme_revenue['Revenue'] = gme_revenue['Revenue'].str.replace(",|\$","")
#print(gamestop_revenue.head())
```

```
C:\Users\Ravena\AppData\Local\Temp\ipykernel_15680\467725191.py:4: FutureWarning: The default value of regex will
change from True to False in a future version.
  gme_revenue['Revenue'] = gme_revenue['Revenue'].str.replace(",|\$","")
```

In [33]:
```
gme_revenue.tail()
```

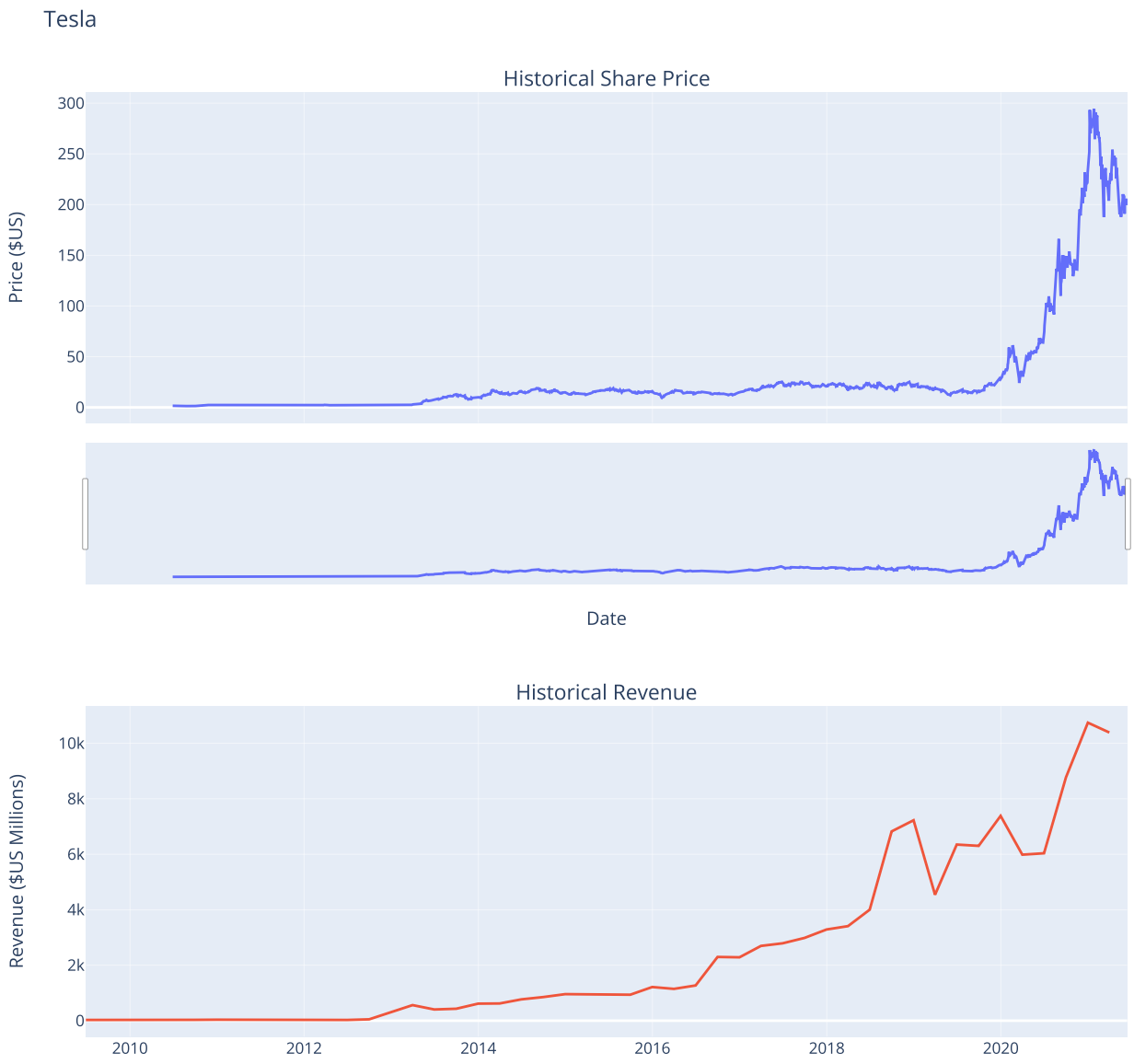
Out[33]:

	Date	Revenue
49	2010-06-30	28
50	2010-03-31	21
51	2009-12-31	NaN
52	2009-09-30	46
53	2009-06-30	27

Question 5: Plot Tesla Stock Graph

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

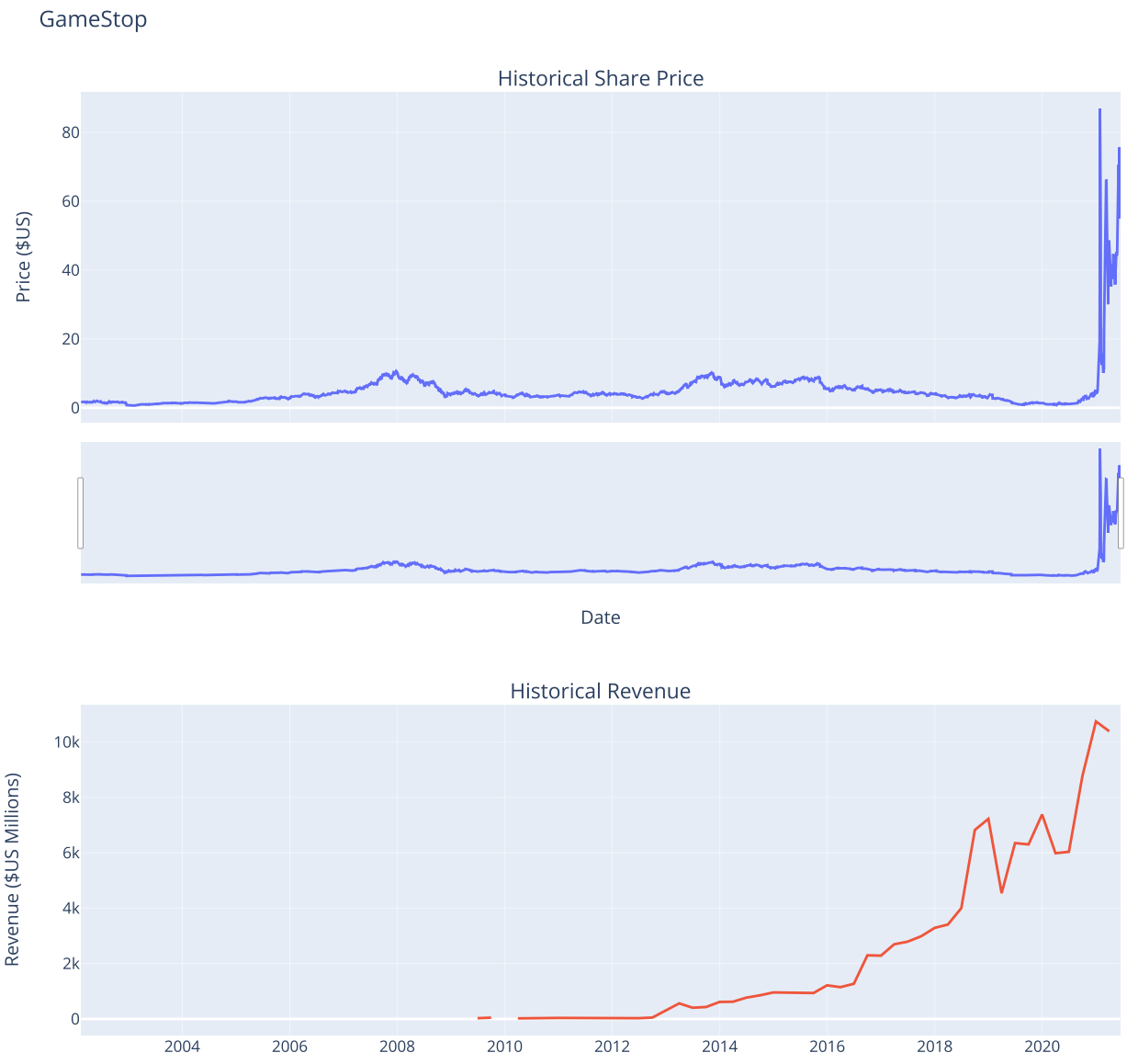
In [35]:
```
make_graph(tesla_data, tesla_revenue, 'Tesla')
```



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.

In [37]:
```
make_graph(gme_data, gme_revenue, 'GameStop')
```



In [ ]:
```

```