Tesla and GME Share Price and Revenue Data

July 10, 2024

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

```
<ul>
    <li>Define a Function that Makes a Graph</li>
    <li>Question 1: Use yfinance to Extract Stock Data</li>
    <li>Question 2: Use Webscraping to Extract Tesla Revenue Data</li>
    <li>Question 3: Use yfinance to Extract Stock Data</li>
    <li>Question 4: Use Webscraping to Extract GME Revenue Data</li>
    <li>Question 5: Plot Tesla Stock Graph</li>
    <li>Question 6: Plot GameStop Stock Graph</li>
</ul>
```

Estimated Time Needed: 30 min

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

[113]:

```
pip install yfinance
```

```
Requirement already satisfied: yfinance in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (0.2.40)
Requirement already satisfied: pandas>=1.3.0 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from yfinance) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from yfinance) (2.0.0)
Requirement already satisfied: requests>=2.31 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from yfinance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from yfinance) (5.2.2)
Requirement already satisfied: platformdirs>=2.0.0 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from yfinance) (3.10.0)
Requirement already satisfied: pytz>=2022.5 in
```

```
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from yfinance) (2024.1)
Requirement already satisfied: frozendict>=2.3.4 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from yfinance) (2.4.4)
Requirement already satisfied: peewee>=3.16.2 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from yfinance) (3.17.6)
Requirement already satisfied: beautifulsoup4>=4.11.1 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from yfinance) (4.12.2)
Requirement already satisfied: html5lib>=1.1 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: six>=1.9 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
pandas>=1.3.0->yfinance) (2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
pandas>=1.3.0->yfinance) (2024.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
requests>=2.31->yfinance) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
requests>=2.31->yfinance) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
requests>=2.31->yfinance) (1.26.19)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
requests>=2.31->yfinance) (2024.6.2)
Note: you may need to restart the kernel to use updated packages.
```

[114]:

```
pip install bs4
```

```
Requirement already satisfied: bs4 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (0.0.2)
Requirement already satisfied: beautifulsoup4 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from bs4) (4.12.2)
Requirement already satisfied: soupsieve>1.2 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
beautifulsoup4->bs4) (2.5)
Note: you may need to restart the kernel to use updated packages.
```

[115]:

```
pip install nbformat
```

```
Requirement already satisfied: nbformat in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (5.9.2)
Requirement already satisfied: fastjsonschema in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from nbformat) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from nbformat) (4.19.2)
Requirement already satisfied: jupyter-core in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from nbformat) (5.5.0)
Requirement already satisfied: traitlets>=5.1 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from nbformat) (5.7.1)
Requirement already satisfied: attrs>=22.2.0 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
jsonschema>=2.6->nbformat) (23.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
jsonschema>=2.6->nbformat) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
jsonschema>=2.6->nbformat) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from
jsonschema>=2.6->nbformat) (0.10.6)
Requirement already satisfied: platformdirs>=2.5 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from jupyter-
core->nbformat) (3.10.0)
Requirement already satisfied: pywin32>=300 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from jupyter-
core->nbformat) (305.1)
Note: you may need to restart the kernel to use updated packages.
```

[116]:

```
pip install plotly
```

```
Requirement already satisfied: plotly in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (5.22.0)
Requirement already satisfied: tenacity>=6.2.0 in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from plotly) (8.5.0)
Requirement already satisfied: packaging in
c:\users\kavin\anaconda3\envs\rp_env\lib\site-packages (from plotly) (23.2)
Note: you may need to restart the kernel to use updated packages.
```

[117]:

```python
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.

[118]:

```
import warnings
# Ignore all warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

0.1 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.**

[119]:

```
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)
    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_data_specific.Close.astype("float"), name="Share Price"), row=1,␣
 ↪col=1)
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date),␣
 ↪y=revenue_data_specific.Revenue.astype("float"), name="Revenue"), 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()
```

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.**

0.2 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`.

[120]:

```
tesla = yf.Ticker('TSLA')
print(tesla)
```

```
yfinance.Ticker object <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.

```
[121]: tesla_data = tesla.history(period='max')
       tesla_data
```

```
[121]:                                  Open        High         Low       Close  \
       Date
       2010-06-29 00:00:00-04:00    1.266667    1.666667    1.169333    1.592667
       2010-06-30 00:00:00-04:00    1.719333    2.028000    1.553333    1.588667
       2010-07-01 00:00:00-04:00    1.666667    1.728000    1.351333    1.464000
       2010-07-02 00:00:00-04:00    1.533333    1.540000    1.247333    1.280000
       2010-07-06 00:00:00-04:00    1.333333    1.333333    1.055333    1.074000
       …                                   …         …         …         …
       2024-07-03 00:00:00-04:00  234.559998  248.350006  234.250000  246.389999
       2024-07-05 00:00:00-04:00  249.809998  252.369995  242.460007  251.520004
       2024-07-08 00:00:00-04:00  247.710007  259.440002  244.570007  252.940002
       2024-07-09 00:00:00-04:00  251.000000  265.609985  250.300003  262.329987
       2024-07-10 00:00:00-04:00  262.829987  265.720001  257.859985  263.549988

                                     Volume  Dividends  Stock Splits
       Date
       2010-06-29 00:00:00-04:00  281494500        0.0           0.0
       2010-06-30 00:00:00-04:00  257806500        0.0           0.0
       2010-07-01 00:00:00-04:00  123282000        0.0           0.0
       2010-07-02 00:00:00-04:00   77097000        0.0           0.0
       2010-07-06 00:00:00-04:00  103003500        0.0           0.0
       …                                …        …            …
       2024-07-03 00:00:00-04:00  166561500        0.0           0.0
       2024-07-05 00:00:00-04:00  154501200        0.0           0.0
       2024-07-08 00:00:00-04:00  157219600        0.0           0.0
       2024-07-09 00:00:00-04:00  160210900        0.0           0.0
       2024-07-10 00:00:00-04:00   70426997        0.0           0.0

       [3531 rows x 7 columns]
```

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.

```
[122]: tesla_data.reset_index(inplace=True)

       tesla_data.head()
```

[122]:

```
                       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
```

0.3 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`.

[123]:

```
url = " https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
  ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"

html_data = requests.get(url).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` .

[124]:

```
soup = BeautifulSoup(html_data, '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`.

Step-by-step instructions

```
Here are the step-by-step instructions:

1. Find All Tables: Start by searching for all HTML tables on a webpage using `soup.find_all('
2. Identify the Relevant Table: then loops through each table. If a table contains the text "Te
3. Initialize a DataFrame: Create an empty Pandas DataFrame called `tesla_revenue` with columns
4. Loop Through Rows: For each row in the relevant table, extract the data from the first and
5. Clean Revenue Data: Remove dollar signs and commas from the revenue value.
6. Add Rows to DataFrame: Create a new row in the DataFrame with the extracted date and cleaned
7. Repeat for All Rows: Continue this process for all rows in the table.
```

Click here if you need help locating the table

```
Below is the code to isolate the table, you will now need to loop through the rows and columns
```

```
soup.find_all("tbody")[1]

If you want to use the read_html function the table is located at index 1

We are focusing on quarterly revenue in the lab.
```

[125]:
```python
read_html_pandas_data = pd.read_html(str(soup))
tesla_revenue = read_html_pandas_data[1]
tesla_revenue.columns = ['Date', 'Revenue']
tesla_revenue
```

[125]:
```
          Date  Revenue
0   2022-09-30  $21,454
1   2022-06-30  $16,934
2   2022-03-31  $18,756
3   2021-12-31  $17,719
4   2021-09-30  $13,757
5   2021-06-30  $11,958
6   2021-03-31  $10,389
7   2020-12-31  $10,744
8   2020-09-30   $8,771
9   2020-06-30   $6,036
10  2020-03-31   $5,985
11  2019-12-31   $7,384
12  2019-09-30   $6,303
13  2019-06-30   $6,350
14  2019-03-31   $4,541
15  2018-12-31   $7,226
16  2018-09-30   $6,824
17  2018-06-30   $4,002
18  2018-03-31   $3,409
19  2017-12-31   $3,288
20  2017-09-30   $2,985
21  2017-06-30   $2,790
22  2017-03-31   $2,696
23  2016-12-31   $2,285
24  2016-09-30   $2,298
25  2016-06-30   $1,270
26  2016-03-31   $1,147
27  2015-12-31   $1,214
28  2015-09-30     $937
29  2015-06-30     $955
30  2015-03-31     $940
31  2014-12-31     $957
32  2014-09-30     $852
33  2014-06-30     $769
```

```
34   2014-03-31      $621
35   2013-12-31      $615
36   2013-09-30      $431
37   2013-06-30      $405
38   2013-03-31      $562
39   2012-12-31      $306
40   2012-09-30       $50
41   2012-06-30       $27
42   2012-03-31       $30
43   2011-12-31       $39
44   2011-09-30       $58
45   2011-06-30       $58
46   2011-03-31       $49
47   2010-12-31       $36
48   2010-09-30       $31
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
```

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

[126]:
```
tesla_revenue['Revenue'] = tesla_revenue['Revenue'].replace({'\$': '', ',':␣
  ↪''}, regex=True)
```

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

[127]:
```
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.

[128]:
```
last_5_rows = tesla_revenue.tail(5)
last_5_rows
```

[128]:
```
           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
```

0.4 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`.

```
[129]: gme = yf.Ticker("GME")
       gme
```

```
[129]: yfinance.Ticker object <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.

```
[130]: gme_data = gme.history(period="max")
       gme_data
```

```
[130]:                                 Open       High        Low      Close  \
       Date
       2002-02-13 00:00:00-05:00   1.620128   1.693350   1.603296   1.691666
       2002-02-14 00:00:00-05:00   1.712707   1.716074   1.670626   1.683250
       2002-02-15 00:00:00-05:00   1.683251   1.687459   1.658002   1.674834
       2002-02-19 00:00:00-05:00   1.666418   1.666418   1.578047   1.607504
       2002-02-20 00:00:00-05:00   1.615920   1.662210   1.603296   1.662210
       …                                 …        …        …        …
       2024-07-03 00:00:00-04:00  24.030001  24.889999  23.650000  24.370001
       2024-07-05 00:00:00-04:00  24.180000  25.080000  23.820000  24.180000
       2024-07-08 00:00:00-04:00  24.120001  25.139999  23.850000  24.450001
       2024-07-09 00:00:00-04:00  24.600000  25.180000  24.000000  24.600000
       2024-07-10 00:00:00-04:00  25.000000  26.450001  24.938101  25.609800

                                    Volume  Dividends  Stock Splits
       Date
       2002-02-13 00:00:00-05:00  76216000        0.0           0.0
       2002-02-14 00:00:00-05:00  11021600        0.0           0.0
       2002-02-15 00:00:00-05:00   8389600        0.0           0.0
       2002-02-19 00:00:00-05:00   7410400        0.0           0.0
       2002-02-20 00:00:00-05:00   6892800        0.0           0.0
       …                                 …       …          …
       2024-07-03 00:00:00-04:00  11829500        0.0           0.0
       2024-07-05 00:00:00-04:00  11782100        0.0           0.0
       2024-07-08 00:00:00-04:00  11815500        0.0           0.0
       2024-07-09 00:00:00-04:00   9419800        0.0           0.0
       2024-07-10 00:00:00-04:00  14861000        0.0           0.0

       [5639 rows x 7 columns]
```

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.

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

```
[131]:                        Date      Open      High       Low     Close    Volume  \
       0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666  76216000
       1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250  11021600
       2 2002-02-15 00:00:00-05:00  1.683251  1.687459  1.658002  1.674834   8389600
       3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504   7410400
       4 2002-02-20 00:00:00-05:00  1.615920  1.662210  1.603296  1.662210   6892800

          Dividends  Stock Splits
       0        0.0           0.0
       1        0.0           0.0
       2        0.0           0.0
       3        0.0           0.0
       4        0.0           0.0
```

0.5 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`.

```
[132]: url = " https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
        ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"

       html_data_2 = requests.get(url).text
```

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

```
[133]: soup = 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.

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

Click here if you need help locating the table

```
Below is the code to isolate the table, you will now need to loop through the rows and columns

soup.find_all("tbody")[1]

If you want to use the read_html function the table is located at index 1
```

```
[134]: read_html_pandas_data_2 = pd.read_html(str(soup))
       gme_revenue = read_html_pandas_data_2[1]
       gme_revenue.columns = ['Date', 'Revenue']
```

```
gme_revenue['Revenue'] = gme_revenue['Revenue'].replace({'\$': '', ',': ''},␣
 ↪regex=True)
gme_revenue.dropna(inplace=True)
gme_revenue = gme_revenue[gme_revenue['Revenue'] != ""]
gme_revenue
```

[134]:

```
          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
..         …       …
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

[62 rows x 2 columns]
```

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

[135]:

```
last_5_rows_GME = gme_revenue.tail(5)
last_5_rows_GME
```

[135]:

```
          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
```

0.6 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.

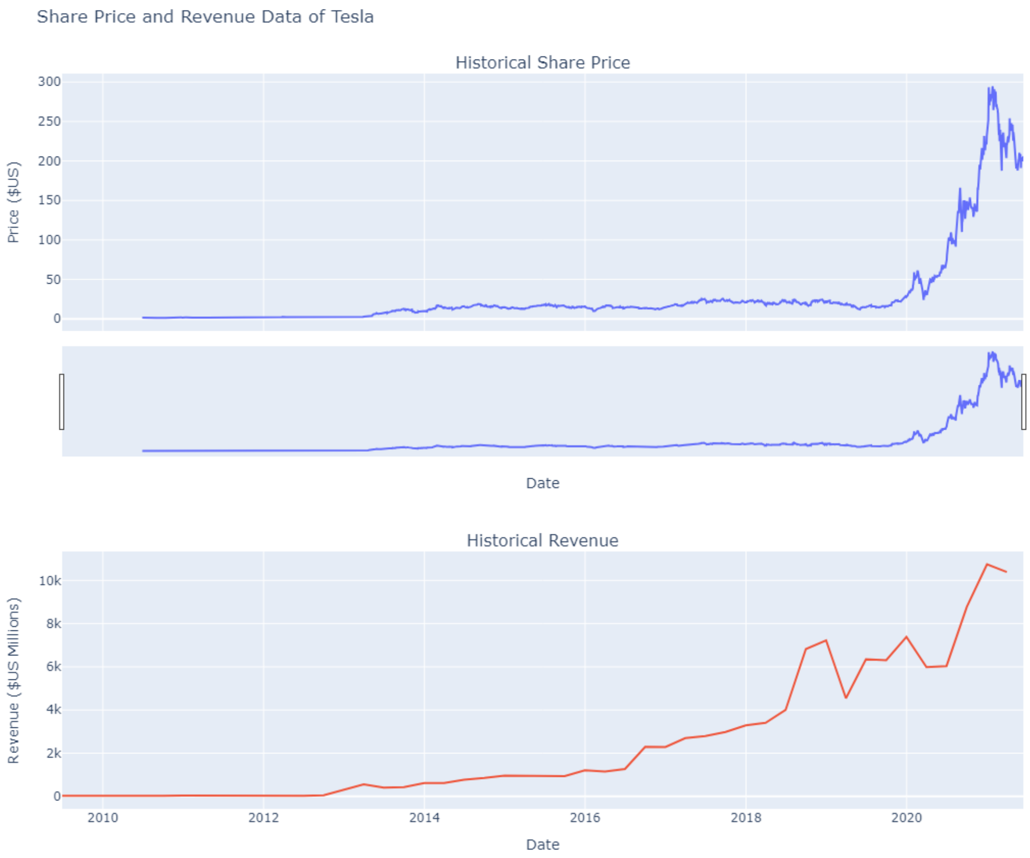
Hint

```
You just need to invoke the make_graph function with the required parameter to print the graph
```

0.6.1 Note: The graph will only show data upto June 2021.

[136]:

```
make_graph(tesla_data, tesla_revenue, 'Share Price and Revenue Data of Tesla')
```



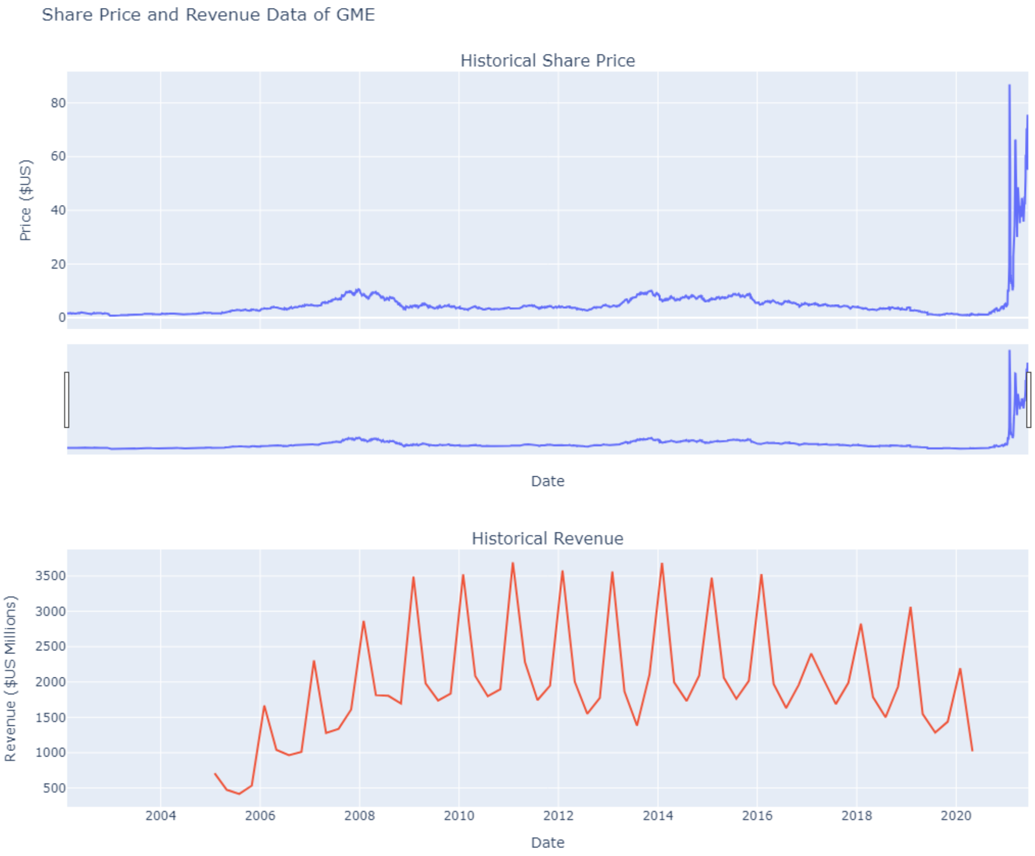
0.7 Question 6: Plot GameStop Stock Graph

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

```
You just need to invoke the make_graph function with the required parameter to print the graphs
```

[137]:
```
make_graph(gme_data, gme_revenue, 'Share Price and Revenue Data of GME')
```



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.

Azim Hirjani

0.8 Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-02-28	1.2	Lakshmi Holla	Changed the URL of GameStop
2020-11-10	1.1	Malika Singla	Deleted the Optional part
2020-08-27	1.0	Malika Singla	Added lab to GitLab

##