Final Assignment

January 6, 2025

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. Use the version as per your python version.

[3]:
```
!pip install yfinance
!pip install bs4
!pip install nbformat
```

```
Collecting yfinance
  Downloading yfinance-0.2.51-py2.py3-none-any.whl.metadata (5.5 kB)
Collecting pandas>=1.3.0 (from yfinance)
  Downloading
pandas-2.2.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(89 kB)
                                   89.9/89.9 kB
13.8 MB/s eta 0:00:00
Collecting numpy>=1.16.5 (from yfinance)
  Downloading
numpy-2.2.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(62 kB)
```

```
                              62.0/62.0 kB
10.2 MB/s eta 0:00:00
Requirement already satisfied: requests>=2.31 in
/opt/conda/lib/python3.11/site-packages (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)
Collecting lxml>=4.9.1 (from yfinance)
  Downloading lxml-5.3.0-cp311-cp311-manylinux_2_28_x86_64.whl.metadata (3.8 kB)
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-
packages (from yfinance) (2024.1)
Collecting frozendict>=2.3.4 (from yfinance)
  Downloading frozendict-2.4.6-py311-none-any.whl.metadata (23 kB)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.17.8.tar.gz (948 kB)
                             948.2/948.2 kB
47.3 MB/s eta 0:00:00
  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/site-packages (from yfinance) (4.12.3)
Collecting html5lib>=1.1 (from yfinance)
  Downloading html5lib-1.1-py2.py3-none-any.whl.metadata (16 kB)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.11/site-
packages (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-
packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.11/site-packages (from pandas>=1.3.0->yfinance) (2.9.0)
Collecting tzdata>=2022.7 (from pandas>=1.3.0->yfinance)
  Downloading tzdata-2024.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.11/site-packages (from requests>=2.31->yfinance) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.11/site-
packages (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)
(2024.12.14)
Downloading yfinance-0.2.51-py2.py3-none-any.whl (104 kB)
                            104.7/104.7 kB
17.3 MB/s eta 0:00:00
Downloading frozendict-2.4.6-py311-none-any.whl (16 kB)
Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
                         112.2/112.2 kB
19.8 MB/s eta 0:00:00
Downloading lxml-5.3.0-cp311-cp311-manylinux_2_28_x86_64.whl (5.0 MB)
                         5.0/5.0 MB
125.2 MB/s eta 0:00:0000:01
Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Downloading
numpy-2.2.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.4 MB)
                         16.4/16.4 MB
118.6 MB/s eta 0:00:0000:0100:01
Downloading
pandas-2.2.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.1
MB)
                         13.1/13.1 MB
131.8 MB/s eta 0:00:0000:010:01
Downloading tzdata-2024.2-py2.py3-none-any.whl (346 kB)
                         346.6/346.6 kB
48.3 MB/s eta 0:00:00
Building wheels for collected packages: peewee
  Building wheel for peewee (pyproject.toml) … done
  Created wheel for peewee:
filename=peewee-3.17.8-cp311-cp311-linux_x86_64.whl size=300750
sha256=dfcabe536f0455a2ee04b47426bf99d7eade619e883b85957122f0a7802d9bbc
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/ff/6c/15/506e25bc390de
450a7fa53c155cd9b0fbd13ad3e84a9abc183
Successfully built peewee
Installing collected packages: peewee, multitasking, tzdata, numpy, lxml,
html5lib, frozendict, pandas, yfinance
Successfully installed frozendict-2.4.6 html5lib-1.1 lxml-5.3.0
multitasking-0.0.11 numpy-2.2.1 pandas-2.2.3 peewee-3.17.8 tzdata-2024.2
yfinance-0.2.51
Collecting bs4
  Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.11/site-
packages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.11/site-
packages (from beautifulsoup4->bs4) (2.5)
Downloading bs4-0.0.2-py2.py3-none-any.whl (1.2 kB)
Installing collected packages: bs4
Successfully installed bs4-0.0.2
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/site-packages (from nbformat) (2.19.1)
Requirement already satisfied: jsonschema>=2.6 in
/opt/conda/lib/python3.11/site-packages (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-
packages (from nbformat) (5.14.3)
Requirement already satisfied: attrs>=22.2.0 in /opt/conda/lib/python3.11/site-
packages (from jsonschema>=2.6->nbformat) (23.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/opt/conda/lib/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-
packages (from jsonschema>=2.6->nbformat) (0.18.0)
Requirement already satisfied: platformdirs>=2.5 in
/opt/conda/lib/python3.11/site-packages (from jupyter-
core!=5.0.*,>=4.12->nbformat) (4.2.1)
```

[4]:
```
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.

[5]:
```
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.**

[6]:
```
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,␣
 ↪infer_datetime_format=True), 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,␣
↪infer_datetime_format=True), 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`.

[7]:
```
tesla_stock = 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.

[8]:
```
tesla_data = tesla_stock.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.

[9]:
```
tesla_data.reset_index(inplace=True)
tesla_data.head()
```

[9]:
```
                       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`.

[10]:
```
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`.

[27]:
```
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. Create an Empty DataFrame
2. Find the Relevant Table
3. Check for the Tesla Quarterly Revenue Table
4. Iterate Through Rows in the Table Body
5. Extract Data from Columns
6. Append Data to the DataFrame
```

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.
```

[33]:
```
tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])

for row in soup.find_all("tbody")[1].find_all("tr"):
    col = row.find_all("td")
    if col != []:
        date = col[0].text
        revenue = col[1].text
        tesla_revenue = pd.concat([tesla_revenue, pd.DataFrame({"Date":[date],␣
↪"Revenue":[revenue]})], ignore_index=True)

tesla_revenue
```

[33]:

```
          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
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.

```
[34]: tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace('$',"").str.
        ↪replace(',',"")
      tesla_revenue.head()
```

```
[34]:          Date Revenue
      0  2022-09-30   21454
      1  2022-06-30   16934
      2  2022-03-31   18756
      3  2021-12-31   17719
      4  2021-09-30   13757
```

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

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

      tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
      tesla_revenue.head()
```

```
[35]:          Date Revenue
      0  2022-09-30   21454
      1  2022-06-30   16934
      2  2022-03-31   18756
      3  2021-12-31   17719
      4  2021-09-30   13757
```

```
[ ]:
```

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

[36]:
```
tesla_revenue.tail()
```

[36]:
```
          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`.

[37]:
```
gamestop_stock = 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.

[39]:
```
gme_data = gamestop_stock.history(period="max")
gme_data
```

[39]:
```
                                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   1.683250   1.687458   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
…                                …        …         …        …
2024-12-30 00:00:00-05:00  31.799999  32.880001  31.610001  32.009998
2024-12-31 00:00:00-05:00  32.060001  32.439999  31.100000  31.340000
2025-01-02 00:00:00-05:00  31.840000  32.049999  30.370001  30.660000
2025-01-03 00:00:00-05:00  30.799999  32.139999  30.570000  31.650000
2025-01-06 00:00:00-05:00  31.700001  33.490002  30.760000  32.820000

                             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
…                               …        …           …
2024-12-30 00:00:00-05:00   9593100        0.0           0.0
2024-12-31 00:00:00-05:00   7395300        0.0           0.0
2025-01-02 00:00:00-05:00   7979700        0.0           0.0
```

```
2025-01-03 00:00:00-05:00   7425800        0.0           0.0
2025-01-06 00:00:00-05:00  12154409        0.0           0.0

[5763 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.

[40]:
```
gme_data.reset_index(inplace=True)
gme_data.head()
```

[40]:
```
                       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  1.683250  1.687458  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`.

[41]:
```
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`.

[42]:
```
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
```

[45]:
```python
gme_revenue = pd.DataFrame(columns=["Date", "Revenue"])

for row in soup.find_all("tbody")[1].find_all("tr"):
    col = row.find_all("td")
    date = col[0].text
    revenue = col[1].text
    gme_revenue = pd.concat([gme_revenue, pd.DataFrame({"Date":[date],␣
 ↪"Revenue":[revenue]})], ignore_index=True)


gme_revenue
```

[45]:
```
          Date Revenue
0   2020-04-30  $1,021
1   2020-01-31  $2,194
2   2019-10-31  $1,439
3   2019-07-31  $1,286
4   2019-04-30  $1,548
..         ...     ...
57  2006-01-31  $1,667
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]
```

[46]:
```python
gme_revenue["Revenue"] = gme_revenue["Revenue"].astype(str).str.replace('$',␣
 ↪"").str.replace(',' , "")
gme_revenue = gme_revenue[gme_revenue["Revenue"] != ""]
gme_revenue.head()
```

[46]:
```
         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
```

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

[47]:
```
gme_revenue.tail()
```

[47]:
```
          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
```

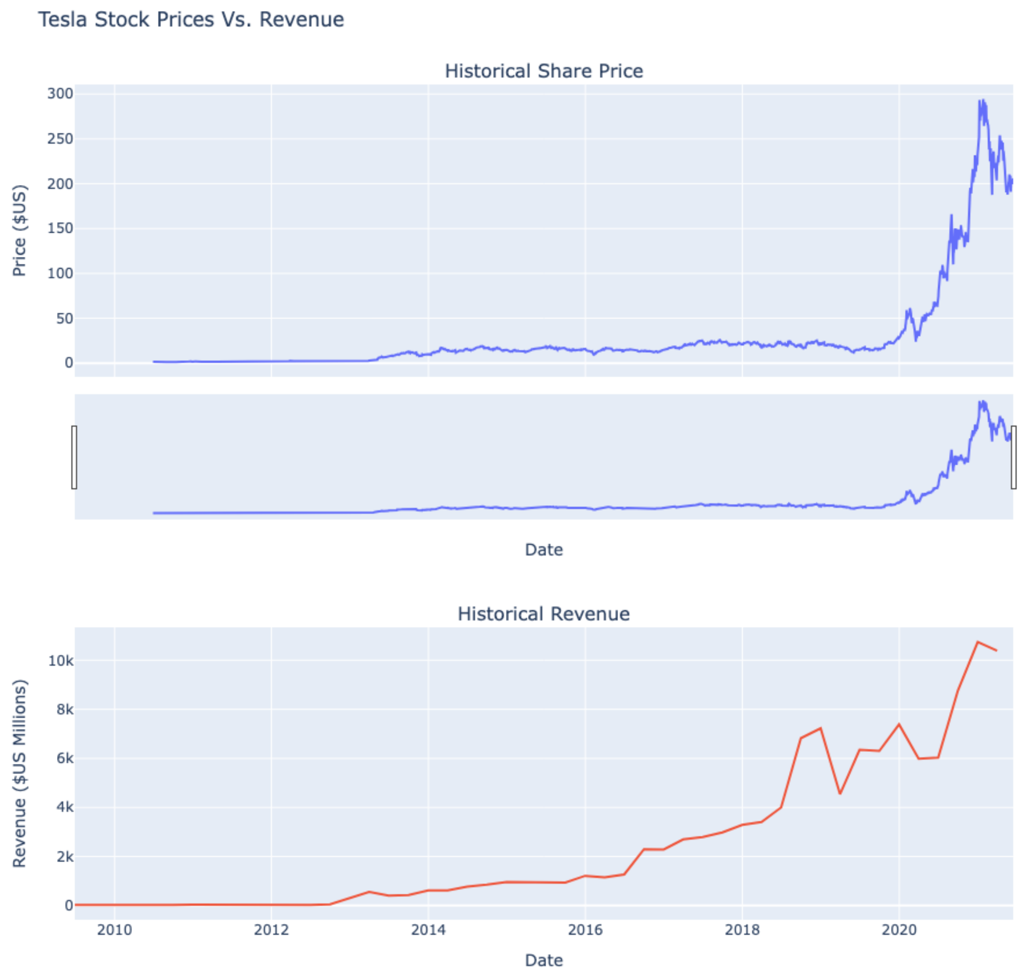
[48]:
```
make_graph(tesla_data, tesla_revenue, 'Tesla Stock Prices Vs. Revenue')
```

```
/tmp/ipykernel_83/3316612210.py:5: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.

/tmp/ipykernel_83/3316612210.py:6: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.
```



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
```

[49]:
```
make_graph(gme_data, gme_revenue, 'GameStop Stock Prices Vs. Revenue')
```

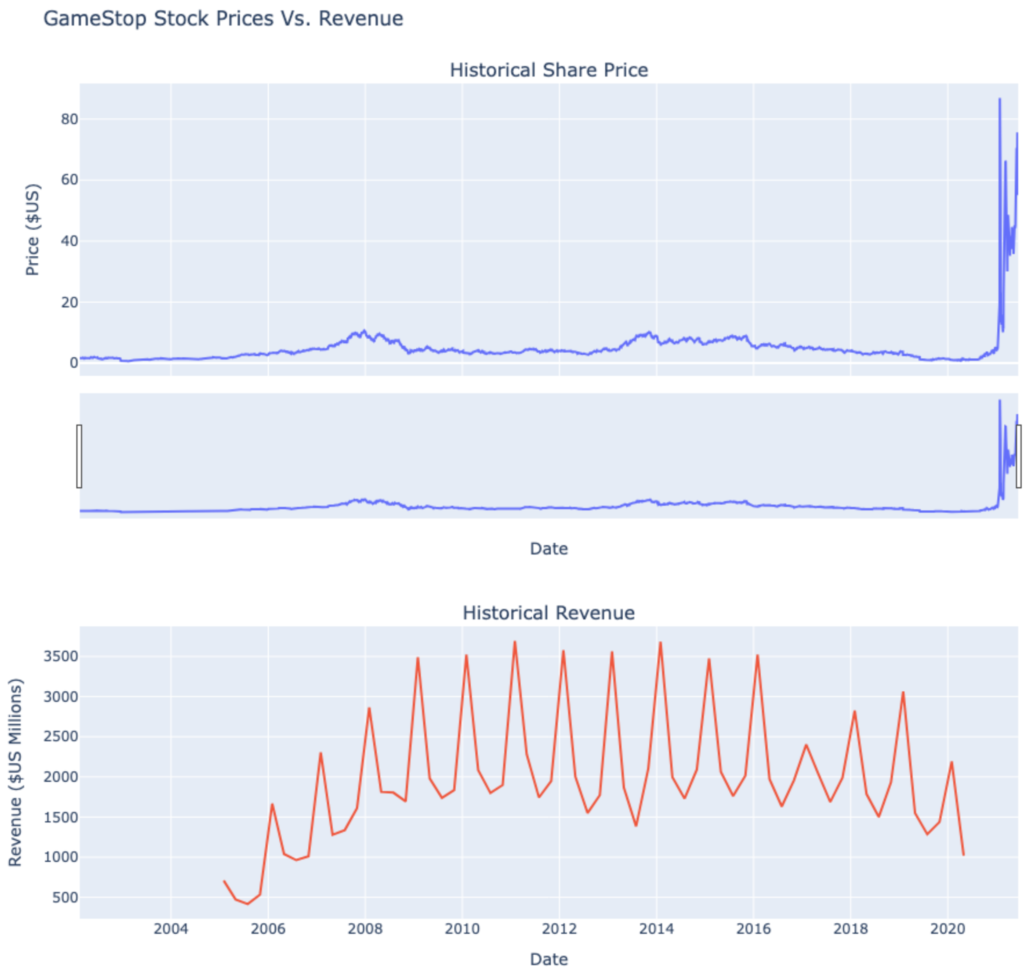
```
/tmp/ipykernel_83/3316612210.py:5: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
```

```
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.

/tmp/ipykernel_83/3316612210.py:6: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.
```



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

##