Final_Exam

December 7, 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. Use the version as per your python version.

[2]:
```
!pip install yfinance
!pip install bs4
!pip install nbformat
```

```
Collecting yfinance
  Downloading yfinance-0.2.50-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
8.4 MB/s eta 0:00:00
Collecting numpy>=1.16.5 (from yfinance)
  Downloading
numpy-2.1.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(62 kB)
```

```
                            62.0/62.0 kB
9.2 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
66.8 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)
(2024.8.30)
Downloading yfinance-0.2.50-py2.py3-none-any.whl (102 kB)
                          102.2/102.2 kB
10.2 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
13.3 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
109.0 MB/s eta 0:00:0000:01
Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Downloading
numpy-2.1.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.3 MB)
                         16.3/16.3 MB
119.8 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
132.5 MB/s eta 0:00:0000:010:01
Downloading tzdata-2024.2-py2.py3-none-any.whl (346 kB)
                         346.6/346.6 kB
38.1 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-py3-none-any.whl
size=138964
sha256=49436628d98ad7041e5abec0519e2f9817da7c4d71680527808500829516e440
  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-2.1.3 pandas-2.2.3 peewee-3.17.8 tzdata-2024.2
yfinance-0.2.50
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)
```

[3]:
```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.

[4]:
```python
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.**

[5]:
```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)
    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`.

[6]:
```
tesla = 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.

[7]:
```
tesla_data= tesla.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.

[8]:
```
tesla_data.reset_index(inplace=True)

# Display the first five rows of the tesla_data DataFrame
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
```

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`.

[11]:
```python
import requests
import pandas as pd
from bs4 import BeautifulSoup

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`.

[12]:
```python
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.
```

[44]:
```
# Step 3: Find the relevant table (Tesla Quarterly Revenue)
tables = soup.find_all("table")
tesla_revenue_table = tables[1]  # Assuming the second table contains quarterly␣
 ↪revenue

# Step 4: Create a list to store the revenue data
revenue_data = []

# Step 5: Iterate through rows in the table body
for row in tesla_revenue_table.find_all("tr")[1:]:  # Skip header row
    cols = row.find_all("td")  # Get all columns in this row
    if len(cols) >= 2:  # Ensure there are enough columns
        date = cols[0].text.strip()  # Extract date from first column
        revenue = cols[1].text.strip()  # Extract revenue from second column

        # Append data as a dictionary to the list
        revenue_data.append({"Date": date, "Revenue": revenue})

# Step 6: Convert the list of dictionaries into a DataFrame
tesla_revenue = pd.DataFrame(revenue_data)

# Display the resulting DataFrame
print(tesla_revenue.tail())
```

```
          Date Revenue
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.

[45]:
```
# Clean the Revenue column by removing dollar signs and commas
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(r'[,\$]', '',␣
 ↪regex=True)

# Convert the cleaned Revenue column to numeric type
tesla_revenue["Revenue"] = pd.to_numeric(tesla_revenue["Revenue"])

# Display the resulting DataFrame
print(tesla_revenue.tail())
```

```
          Date  Revenue
49  2010-06-30     28.0
50  2010-03-31     21.0
51  2009-12-31      NaN
52  2009-09-30     46.0
```

```
53  2009-06-30     27.0
```

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

[47]:
```python
# Remove rows where Revenue is null or an empty string
tesla_revenue = tesla_revenue[tesla_revenue["Revenue"].notna() &␣
 ↪(tesla_revenue["Revenue"] != '')]

# Display the cleaned DataFrame
print(tesla_revenue.tail())
```

```
          Date  Revenue
48  2010-09-30     31.0
49  2010-06-30     28.0
50  2010-03-31     21.0
52  2009-09-30     46.0
53  2009-06-30     27.0
```

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

[25]:
```python
# Display the last 5 rows of the tesla_revenue DataFrame
last_five_rows = tesla_revenue.tail()
print(last_five_rows)
```

```
          Date  Revenue
48  2010-09-30     31.0
49  2010-06-30     28.0
50  2010-03-31     21.0
52  2009-09-30     46.0
53  2009-06-30     27.0
```

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`.

[26]:
```python
import yfinance as yf
import pandas as pd

# Create a Ticker object for GameStop (GME)
game_stop = 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.

[28]:
```python
# Step 2: Extract stock information and save it in a DataFrame named gme_data
# Set the period parameter to max to get information for the maximum amount of␣
 ↪time
```

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

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.

[29]:
```
# Step 3: Reset the index of the gme_data DataFrame
gme_data.reset_index(inplace=True)

# Step 4: Display the first five rows of the gme_data DataFrame
print(gme_data.head())
```

```
   index                      Date      Open      High       Low     Close  \
0      0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666
1      1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250
2      2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658001  1.674834
3      3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504
4      4 2002-02-20 00:00:00-05:00  1.615921  1.662210  1.603296  1.662210

     Volume  Dividends  Stock Splits
0  76216000        0.0           0.0
1  11021600        0.0           0.0
2   8389600        0.0           0.0
3   7410400        0.0           0.0
4   6892800        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`.

[30]:
```
import requests
import pandas as pd
from bs4 import BeautifulSoup

# Step 1: Define the URL and download the webpage
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`.

[32]:
```
# Step 2: Parse the HTML data using BeautifulSoup
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
```

[35]:
```python
# Step 3: Isolate the relevant table (GameStop Quarterly Revenue)
table_body = soup.find_all("tbody")[1]  # Get the second tbody which contains␣
↪GME revenue

# Step 4: Create a list to store the revenue data
revenue_data = []

# Step 5: Iterate through rows in the table body
for row in table_body.find_all("tr"):  # Loop through each row in tbody
    cols = row.find_all("td")  # Get all columns in this row
    if len(cols) >= 2:  # Ensure there are enough columns
        date = cols[0].text.strip()  # Extract date from first column
        revenue = cols[1].text.strip()  # Extract revenue from second column

        # Append data as a dictionary to the list
        revenue_data.append({"Date": date, "Revenue": revenue})

# Step 6: Convert the list of dictionaries into a DataFrame
gme_revenue = pd.DataFrame(revenue_data)

# Clean the Revenue column by removing dollar signs and commas
gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(r'[,\$]', '',␣
↪regex=True)

# Convert the cleaned Revenue column to numeric type
gme_revenue["Revenue"] = pd.to_numeric(gme_revenue["Revenue"])
```

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

[36]:
```python
# Display the last 5 rows of the gme_revenue DataFrame
last_five_rows = gme_revenue.tail()
print(last_five_rows)
```

```
          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
```

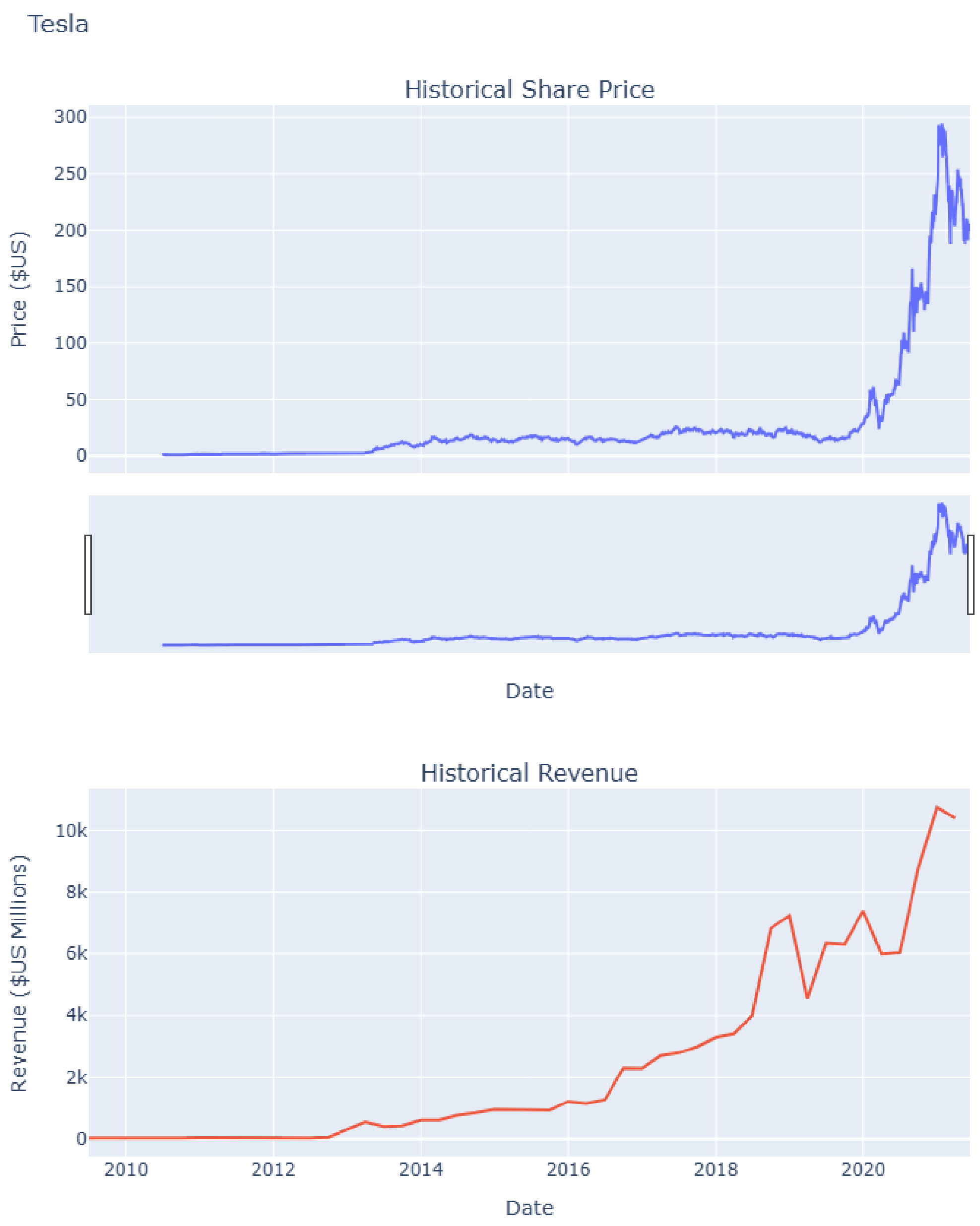
[38]:
```
make_graph(tesla_data, tesla_revenue, 'Tesla')
```

```
/tmp/ipykernel_146/3316612210.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_146/3316612210.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 graph

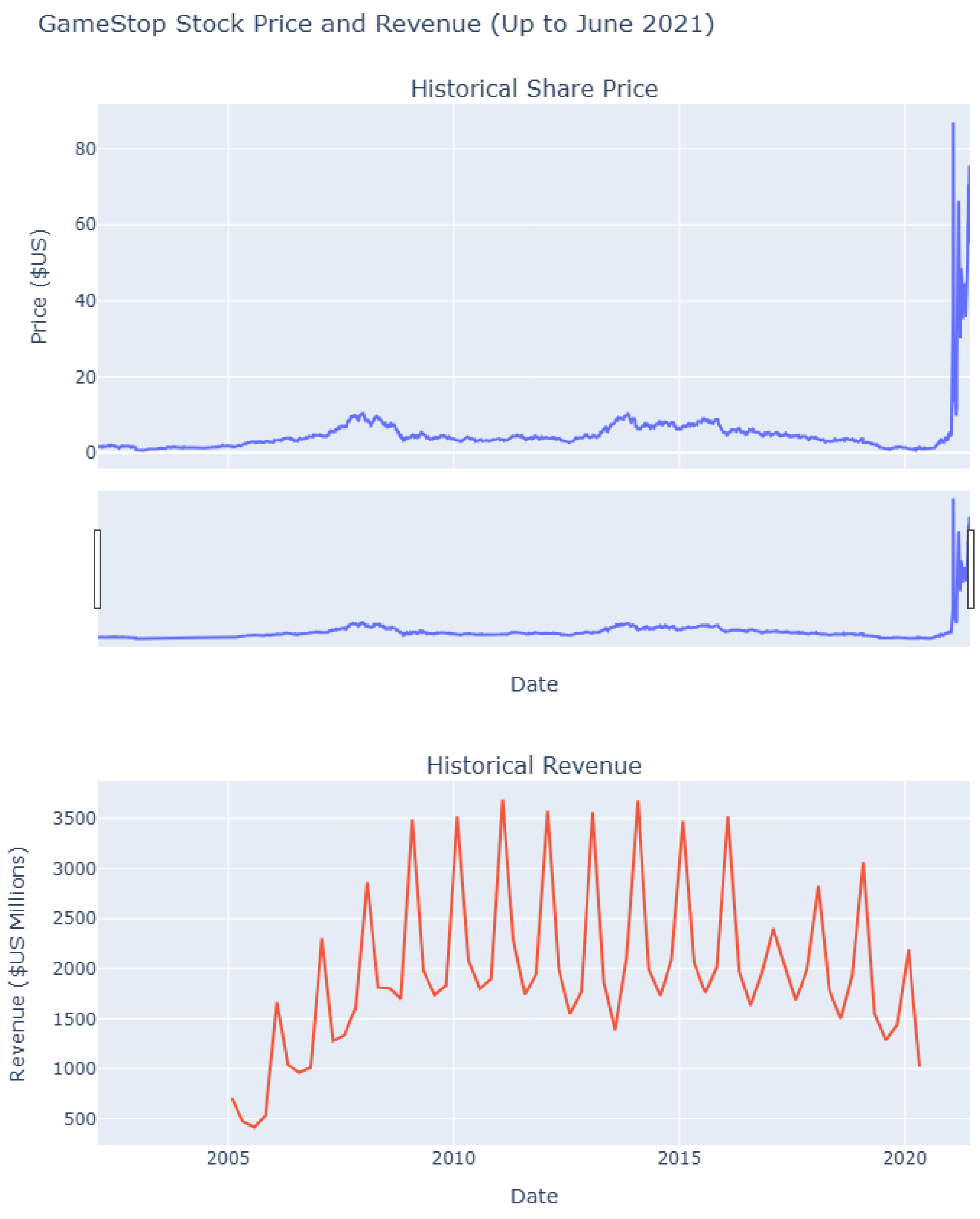
[39]:
```
make_graph(gme_data, gme_revenue, 'GameStop Stock Price and Revenue (Up to June␣
↪2021)')
```

```
/tmp/ipykernel_146/3316612210.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_146/3316612210.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

##