Revenue Data and Building a Dashboard

December 1, 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.

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

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

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.

[ ]:
```
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.**

```
[ ]: # The make_graph function has been modified to use Matplotlib for static graphs.
     ↪ Earlier, it used Plotly to generate interactive dashboards, which caused␣
     ↪issues when uploading the notebook in the MARK assignment submission.




     import matplotlib.pyplot as plt

     def make_graph(stock_data, revenue_data, stock):
         stock_data_specific = stock_data[stock_data.Date <= '2021-06-14']
         revenue_data_specific = revenue_data[revenue_data.Date <= '2021-04-30']

         fig, axes = plt.subplots(2, 1, figsize=(12, 8), sharex=True)

         # Stock price
         axes[0].plot(pd.to_datetime(stock_data_specific.Date), stock_data_specific.
     ↪Close.astype("float"), label="Share Price", color="blue")
         axes[0].set_ylabel("Price ($US)")
         axes[0].set_title(f"{stock} - Historical Share Price")

         # Revenue
         axes[1].plot(pd.to_datetime(revenue_data_specific.Date),␣
     ↪revenue_data_specific.Revenue.astype("float"), label="Revenue",␣
     ↪color="green")
         axes[1].set_ylabel("Revenue ($US Millions)")
         axes[1].set_xlabel("Date")
         axes[1].set_title(f"{stock} - Historical Revenue")

         plt.tight_layout()
         plt.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`.

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.

# Extract stock history tesla_data = tesla.history(period="max")

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

```
Collecting yfinance
  Downloading yfinance-0.2.66-py2.py3-none-any.whl.metadata (6.0 kB)
Collecting pandas>=1.3.0 (from yfinance)
  Downloading pandas-2.3.3-cp312-cp312-
manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (91 kB)
Collecting numpy>=1.16.5 (from yfinance)
  Downloading
numpy-2.3.5-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata
(62 kB)
Requirement already satisfied: requests>=2.31 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.32.3)
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.12.tar.gz (19 kB)
  Preparing metadata (setup.py) … done
Requirement already satisfied: platformdirs>=2.0.0 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.3.6)
Requirement already satisfied: pytz>=2022.5 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2024.2)
Requirement already satisfied: frozendict>=2.3.4 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (2.4.6)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.18.3.tar.gz (3.0 MB)
                           3.0/3.0 MB
135.7 MB/s eta 0:00:00
  Installing build dependencies … one
  Getting requirements to build wheel … done
  Preparing metadata (pyproject.toml) … done
Requirement already satisfied: beautifulsoup4>=4.11.1 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.12.3)
Collecting curl_cffi>=0.7 (from yfinance)
  Downloading curl_cffi-0.13.0-cp39-abi3-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Collecting protobuf>=3.19.0 (from yfinance)
  Downloading protobuf-6.33.1-cp39-abi3-manylinux2014_x86_64.whl.metadata (593
bytes)
Collecting websockets>=13.0 (from yfinance)
  Downloading websockets-15.0.1-cp312-cp312-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl.metadata (6.8 kB)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-
packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: cffi>=1.12.0 in /opt/conda/lib/python3.12/site-
```

```
packages (from curl_cffi>=0.7->yfinance) (1.17.1)
Requirement already satisfied: certifi>=2024.2.2 in
/opt/conda/lib/python3.12/site-packages (from curl_cffi>=0.7->yfinance)
(2024.12.14)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.12/site-packages (from pandas>=1.3.0->yfinance)
(2.9.0.post0)
Collecting tzdata>=2022.7 (from pandas>=1.3.0->yfinance)
  Downloading tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: charset_normalizer<4,>=2 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.12/site-
packages (from requests>=2.31->yfinance) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (2.3.0)
Requirement already satisfied: pycparser in /opt/conda/lib/python3.12/site-
packages (from cffi>=1.12.0->curl_cffi>=0.7->yfinance) (2.22)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance) (1.17.0)
Downloading yfinance-0.2.66-py2.py3-none-any.whl (123 kB)
Downloading
curl_cffi-0.13.0-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.3
MB)
                         8.3/8.3 MB
96.2 MB/s eta 0:00:00
Downloading
numpy-2.3.5-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (16.6
MB)
                         16.6/16.6 MB
178.9 MB/s eta 0:00:00
Downloading
pandas-2.3.3-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (12.4
MB)
                         12.4/12.4 MB
189.6 MB/s eta 0:00:00
Downloading protobuf-6.33.1-cp39-abi3-manylinux2014_x86_64.whl (323 kB)
Downloading websockets-15.0.1-cp312-cp312-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (182 kB)
Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Building wheels for collected packages: multitasking, peewee
  Building wheel for multitasking (setup.py) … one
  Created wheel for multitasking: filename=multitasking-0.0.12-py3-none-
any.whl size=15605
sha256=43eaead317b4fe8e318d6ce869086281f74138462cf32f502e1d3bfda54fbc2e
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/cc/bd/6f/664d62c99327a
beef7d86489e6631cbf45b56fbf7ef1d6ef00
  Building wheel for peewee (pyproject.toml) … one
  Created wheel for peewee:
filename=peewee-3.18.3-cp312-cp312-linux_x86_64.whl size=303891
sha256=87e27ee2610a40210dca4ec33c31d9a4d29bd03df490b23940b5f3c42d6559fa
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/e2/48/b6/675a31c56e50b
8b343e1ffbb1d9209f0d95025e2cfa0bbeeed
Successfully built multitasking peewee
Installing collected packages: peewee, multitasking, websockets, tzdata,
protobuf, numpy, pandas, curl_cffi, yfinance
Successfully installed curl_cffi-0.13.0 multitasking-0.0.12 numpy-2.3.5
pandas-2.3.3 peewee-3.18.3 protobuf-6.33.1 tzdata-2025.2 websockets-15.0.1
yfinance-0.2.66
```

[3]:
```python
import yfinance as yf

# Create ticker object
tesla = yf.Ticker("TSLA")

# Extract stock history
tesla_data = tesla.history(period="max")

# Reset index
tesla_data.reset_index(inplace=True)

# Display first 5 rows
tesla_data.head()
```

[3]:
```
                       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
```

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.

0.3 Question 2: Use Webscraping to Extract Tesla Revenue Data

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

```
[9]: import yfinance as yf

     # Create ticker object
     tesla = yf.Ticker("TSLA")

     # Extract stock history
     tesla_data = tesla.history(period="max")

     # Reset index safely
     tesla_data.reset_index(drop=True, inplace=True)

     # View result
     tesla_data.head()
```

```
[9]:        Open      High       Low     Close     Volume  Dividends  Stock Splits
     0  1.266667  1.666667  1.169333  1.592667  281494500        0.0           0.0
     1  1.719333  2.028000  1.553333  1.588667  257806500        0.0           0.0
     2  1.666667  1.728000  1.351333  1.464000  123282000        0.0           0.0
     3  1.533333  1.540000  1.247333  1.280000   77097000        0.0           0.0
     4  1.333333  1.333333  1.055333  1.074000  103003500        0.0           0.0
```

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`.

```
[12]: import requests

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

      # Download page
      html_data = requests.get(url).text
```

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

```
[14]: from bs4 import BeautifulSoup

      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.
```

[15]:
```python
import pandas as pd

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

[16]:
```python
soup.find_all("tbody")[1]
```

[16]:
```
<tbody>
<tr>
<td style="text-align:center">2022-09-30</td>
<td style="text-align:center">$21,454</td>
</tr>
<tr>
<td style="text-align:center">2022-06-30</td>
<td style="text-align:center">$16,934</td>
</tr>
<tr>
<td style="text-align:center">2022-03-31</td>
<td style="text-align:center">$18,756</td>
</tr>
<tr>
<td style="text-align:center">2021-12-31</td>
<td style="text-align:center">$17,719</td>
</tr>
<tr>
<td style="text-align:center">2021-09-30</td>
<td style="text-align:center">$13,757</td>
</tr>
<tr>
```

```
<td style="text-align:center">2021-06-30</td>
<td style="text-align:center">$11,958</td>
</tr>
<tr>
<td style="text-align:center">2021-03-31</td>
<td style="text-align:center">$10,389</td>
</tr>
<tr>
<td style="text-align:center">2020-12-31</td>
<td style="text-align:center">$10,744</td>
</tr>
<tr>
<td style="text-align:center">2020-09-30</td>
<td style="text-align:center">$8,771</td>
</tr>
<tr>
<td style="text-align:center">2020-06-30</td>
<td style="text-align:center">$6,036</td>
</tr>
<tr>
<td style="text-align:center">2020-03-31</td>
<td style="text-align:center">$5,985</td>
</tr>
<tr>
<td style="text-align:center">2019-12-31</td>
<td style="text-align:center">$7,384</td>
</tr>
<tr>
<td style="text-align:center">2019-09-30</td>
<td style="text-align:center">$6,303</td>
</tr>
<tr>
<td style="text-align:center">2019-06-30</td>
<td style="text-align:center">$6,350</td>
</tr>
<tr>
<td style="text-align:center">2019-03-31</td>
<td style="text-align:center">$4,541</td>
</tr>
<tr>
<td style="text-align:center">2018-12-31</td>
<td style="text-align:center">$7,226</td>
</tr>
<tr>
<td style="text-align:center">2018-09-30</td>
<td style="text-align:center">$6,824</td>
</tr>
```

```
<tr>
<td style="text-align:center">2018-06-30</td>
<td style="text-align:center">$4,002</td>
</tr>
<tr>
<td style="text-align:center">2018-03-31</td>
<td style="text-align:center">$3,409</td>
</tr>
<tr>
<td style="text-align:center">2017-12-31</td>
<td style="text-align:center">$3,288</td>
</tr>
<tr>
<td style="text-align:center">2017-09-30</td>
<td style="text-align:center">$2,985</td>
</tr>
<tr>
<td style="text-align:center">2017-06-30</td>
<td style="text-align:center">$2,790</td>
</tr>
<tr>
<td style="text-align:center">2017-03-31</td>
<td style="text-align:center">$2,696</td>
</tr>
<tr>
<td style="text-align:center">2016-12-31</td>
<td style="text-align:center">$2,285</td>
</tr>
<tr>
<td style="text-align:center">2016-09-30</td>
<td style="text-align:center">$2,298</td>
</tr>
<tr>
<td style="text-align:center">2016-06-30</td>
<td style="text-align:center">$1,270</td>
</tr>
<tr>
<td style="text-align:center">2016-03-31</td>
<td style="text-align:center">$1,147</td>
</tr>
<tr>
<td style="text-align:center">2015-12-31</td>
<td style="text-align:center">$1,214</td>
</tr>
<tr>
<td style="text-align:center">2015-09-30</td>
<td style="text-align:center">$937</td>
```

```
</tr>
<tr>
<td style="text-align:center">2015-06-30</td>
<td style="text-align:center">$955</td>
</tr>
<tr>
<td style="text-align:center">2015-03-31</td>
<td style="text-align:center">$940</td>
</tr>
<tr>
<td style="text-align:center">2014-12-31</td>
<td style="text-align:center">$957</td>
</tr>
<tr>
<td style="text-align:center">2014-09-30</td>
<td style="text-align:center">$852</td>
</tr>
<tr>
<td style="text-align:center">2014-06-30</td>
<td style="text-align:center">$769</td>
</tr>
<tr>
<td style="text-align:center">2014-03-31</td>
<td style="text-align:center">$621</td>
</tr>
<tr>
<td style="text-align:center">2013-12-31</td>
<td style="text-align:center">$615</td>
</tr>
<tr>
<td style="text-align:center">2013-09-30</td>
<td style="text-align:center">$431</td>
</tr>
<tr>
<td style="text-align:center">2013-06-30</td>
<td style="text-align:center">$405</td>
</tr>
<tr>
<td style="text-align:center">2013-03-31</td>
<td style="text-align:center">$562</td>
</tr>
<tr>
<td style="text-align:center">2012-12-31</td>
<td style="text-align:center">$306</td>
</tr>
<tr>
<td style="text-align:center">2012-09-30</td>
```

```
<td style="text-align:center">$50</td>
</tr>
<tr>
<td style="text-align:center">2012-06-30</td>
<td style="text-align:center">$27</td>
</tr>
<tr>
<td style="text-align:center">2012-03-31</td>
<td style="text-align:center">$30</td>
</tr>
<tr>
<td style="text-align:center">2011-12-31</td>
<td style="text-align:center">$39</td>
</tr>
<tr>
<td style="text-align:center">2011-09-30</td>
<td style="text-align:center">$58</td>
</tr>
<tr>
<td style="text-align:center">2011-06-30</td>
<td style="text-align:center">$58</td>
</tr>
<tr>
<td style="text-align:center">2011-03-31</td>
<td style="text-align:center">$49</td>
</tr>
<tr>
<td style="text-align:center">2010-12-31</td>
<td style="text-align:center">$36</td>
</tr>
<tr>
<td style="text-align:center">2010-09-30</td>
<td style="text-align:center">$31</td>
</tr>
<tr>
<td style="text-align:center">2010-06-30</td>
<td style="text-align:center">$28</td>
</tr>
<tr>
<td style="text-align:center">2010-03-31</td>
<td style="text-align:center">$21</td>
</tr>
<tr>
<td style="text-align:center">2009-12-31</td>
<td style="text-align:center"></td>
</tr>
<tr>
```

```
<td style="text-align:center">2009-09-30</td>
<td style="text-align:center">$46</td>
</tr>
<tr>
<td style="text-align:center">2009-06-30</td>
<td style="text-align:center">$27</td>
</tr>
</tbody>
```

[18]:
```
tbody = soup.find_all("tbody")[1]
```

[19]:
```
rows = tbody.find_all("tr")
```

[21]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Step 1: Download webpage
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

# Step 2: Parse HTML
soup = BeautifulSoup(html_data, "html.parser")

# Step 3: Create empty DataFrame
tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])

# Step 4: Locate the revenue table (tbody index 1)
tbody = soup.find_all("tbody")[1]

# Step 5: Loop through rows and extract Date + Revenue
rows = tbody.find_all("tr")

for row in rows:
    cols = row.find_all("td")
    date = cols[0].text.strip()
    revenue = cols[1].text.strip()

    new_row = pd.DataFrame({"Date": [date], "Revenue": [revenue]})
    tesla_revenue = pd.concat([tesla_revenue, new_row], ignore_index=True)

# Step 6: Display the result
tesla_revenue.head()
```

[21]:
```
         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
```

[22]:
```
tesla_revenue["Revenue"] = tesla_revenue["Revenue"].str.replace(",", "").str.
 ↪replace("$", "")
tesla_revenue.dropna(inplace=True)
tesla_revenue = tesla_revenue[tesla_revenue["Revenue"] != ""]
tesla_revenue.head()
```

[22]:
```
         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 line to remove the comma and dollar sign from the `Revenue` column.

[23]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.
 ↪replace(',|\$',"",regex=True)
```

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

[24]:
```
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.

[25]:
```
tesla_revenue.tail()
```

[25]:
```
          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`.

[26]:
```
# Question 3: Use yfinance to extract GameStop stock data (GME)
# Paste this into a single Jupyter cell.
```

```
import sys

# 1) Try to import yfinance, install if missing (works in many notebook␣
↪environments)
try:
    import yfinance as yf
    print("Imported yfinance.")
except Exception as e:
    print("yfinance not found, trying to install it...")
    try:
        !pip install yfinance --quiet
        import importlib
        importlib.reload(sys.modules.get('pip', None) or sys)
        import yfinance as yf
        print("Installed and imported yfinance.")
    except Exception as e2:
        print("Automatic install failed. Please ask instructor or use a␣
↪different environment.")
        raise

# 2) Create ticker object and download history
try:
    gme = yf.Ticker("GME")
    gme_data = gme.history(period="max")
    print("Downloaded GME history via yfinance.Ticker.history().")
except Exception as e:
    # fallback: try the pdr override if pandas_datareader + yfinance present
    try:
        from pandas_datareader import data as pdr
        yf.pdr_override()
        gme_data = pdr.get_data_yahoo("GME", start="2000-01-01")
        print("Downloaded GME history via pandas_datareader.get_data_yahoo()␣
↪fallback.")
    except Exception as e2:
        print("Failed to download GME with both methods. Error:")
        raise

# 3) Reset index safely (drop=True avoids duplicate 'level_0' or 'index' column␣
↪issues)
# If you want to preserve the date as a column, do not drop; the lab expects a␣
↪reset index.
gme_data.reset_index(drop=True, inplace=True)

# 4) Display first 5 rows (head)
print("\nGME Data - first 5 rows:")
display(gme_data.head())
```

```
Imported yfinance.
Downloaded GME history via yfinance.Ticker.history().

GME Data - first 5 rows:
       Open      High       Low     Close    Volume  Dividends  Stock Splits
0  1.620128  1.693350  1.603296  1.691667  76216000        0.0           0.0
1  1.712707  1.716074  1.670626  1.683250  11021600        0.0           0.0
2  1.683250  1.687458  1.658002  1.674834   8389600        0.0           0.0
3  1.666418  1.666418  1.578047  1.607504   7410400        0.0           0.0
4  1.615920  1.662210  1.603296  1.662210   6892800        0.0           0.0
```

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.

[27]:
```python
import yfinance as yf

# Create ticker object for GameStop
gme = yf.Ticker("GME")

# Extract stock history data
gme_data = gme.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]:
```python
# Reset index safely
gme_data.reset_index(drop=True, inplace=True)

# Display first 5 rows
gme_data.head()
```

[29]:
```
       Open      High       Low     Close    Volume  Dividends  Stock Splits
0  1.620129  1.693350  1.603296  1.691667  76216000        0.0           0.0
1  1.712707  1.716074  1.670626  1.683250  11021600        0.0           0.0
2  1.683250  1.687458  1.658002  1.674834   8389600        0.0           0.0
3  1.666418  1.666418  1.578047  1.607504   7410400        0.0           0.0
4  1.615920  1.662210  1.603296  1.662210   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`

[35]:
```
# Try to install the parsers pandas uses
!pip install --quiet lxml html5lib beautifulsoup4
```

[41]:
```
url2 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"
```

[42]:
```
import requests
from bs4 import BeautifulSoup

html_data_2 = requests.get(url2).text
soup2 = BeautifulSoup(html_data_2, "html.parser")

tables = soup2.find_all("table")
print("Number of tables found:", len(tables))
```

```
Number of tables found: 6
```

[43]:
```
target_table = tables[1]   # GameStop table
rows = target_table.find_all("tr")

records = []
for row in rows[1:]:
    cols = row.find_all("td")
    if len(cols) >= 2:
        date = cols[0].text.strip()
        revenue = cols[1].text.strip()
        records.append((date, revenue))

import pandas as pd
gme_revenue = pd.DataFrame(records, columns=["Date", "Revenue"])

gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace(r"[\$,]", "",␣
 ↪regex=True)
gme_revenue = gme_revenue[gme_revenue["Revenue"] != ""].reset_index(drop=True)

gme_revenue.head()
```

[43]:
```
         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
```

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

[44]:
```
# Question 4 - Extract GameStop revenue table (robust: works with or without pd.
 ↪read_html/lxml)
```

```python
import requests
import pandas as pd
from bs4 import BeautifulSoup

url2 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"

html_data_2 = requests.get(url2).text

# Try the easy read_html path first (works if pandas has a parser available)
gme_revenue = None
try:
    tables = pd.read_html(html_data_2, flavor="bs4")  # flavor='bs4' avoids␣
 ↪lxml when possible
    # lab's GameStop revenue table is table index 1
    if len(tables) > 1:
        gme_revenue = tables[1].copy()
        # Ensure column names
        if len(gme_revenue.columns) >= 2:
            gme_revenue.columns = ["Date", "Revenue"]
except Exception:
    # fallback to BeautifulSoup manual parsing (guaranteed)
    soup2 = BeautifulSoup(html_data_2, "html.parser")
    all_tables = soup2.find_all("table")
    if len(all_tables) < 2:
        raise Exception("Could not find the GameStop revenue table on the page␣
 ↪(no tables found).")
    target_table = all_tables[1]   # in this course page the GME table is table␣
 ↪index 1
    records = []
    rows = target_table.find_all("tr")
    for row in rows[1:]:   # skip header if present
        cols = row.find_all("td")
        if len(cols) >= 2:
            date = cols[0].get_text(strip=True)
            revenue = cols[1].get_text(strip=True)
            records.append((date, revenue))
    gme_revenue = pd.DataFrame(records, columns=["Date", "Revenue"])

# Final cleaning steps (remove $ and commas, drop empty/null, reset index)
gme_revenue["Date"] = gme_revenue["Date"].astype(str).str.strip()
gme_revenue["Revenue"] = gme_revenue["Revenue"].astype(str).str.strip()
# remove $ and commas
gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace(r"[\$,]", "",␣
 ↪regex=True)
# drop empty or missing
gme_revenue.dropna(subset=["Revenue", "Date"], inplace=True)
```

```
gme_revenue = gme_revenue[gme_revenue["Revenue"] != ""].reset_index(drop=True)

# (optional) convert Revenue to numeric for later steps
gme_revenue["Revenue"] = pd.to_numeric(gme_revenue["Revenue"], errors="coerce")
gme_revenue.dropna(subset=["Revenue"], inplace=True)
gme_revenue = gme_revenue.reset_index(drop=True)

# Show last 5 rows for the screenshot required by the lab
gme_revenue.tail()
```

```
/tmp/ipykernel_408/1043474677.py:13: FutureWarning: Passing literal html to
'read_html' is deprecated and will be removed in a future version. To read from
a literal string, wrap it in a 'StringIO' object.
  tables = pd.read_html(html_data_2, flavor="bs4")  # flavor='bs4' avoids lxml
when possible
```

[44]:

```
          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
```

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]:

```
import requests
from bs4 import BeautifulSoup

url2 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"

html_data_2 = requests.get(url2).text

soup = BeautifulSoup(html_data_2, "html.parser")
```

Remove the comma and dollar sign, an null or empty strings from the Revenue column.

```
[46]: import pandas as pd

      # Find all tables
      tables = soup.find_all("table")

      # GameStop table is ALWAYS the second table on this page (index 1)
      table = tables[1]

      # Extract rows
      records = []
      rows = table.find_all("tr")

      for row in rows[1:]:  # skip header
          cols = row.find_all("td")
          if len(cols) >= 2:
              date = cols[0].get_text(strip=True)
              revenue = cols[1].get_text(strip=True)
              records.append((date, revenue))

      # Convert to DataFrame
      gme_revenue = pd.DataFrame(records, columns=["Date", "Revenue"])
```

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

```
[48]: # Remove $ and comma
      gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace(r"[\$,]", "",␣
        ↪regex=True)

      # Remove null or empty strings
      gme_revenue = gme_revenue[gme_revenue["Revenue"] != ""]
      gme_revenue.dropna(inplace=True)

      # Reset index
      gme_revenue = gme_revenue.reset_index(drop=True)
```

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
```

```
[56]: !pip install matplotlib
```

```
Collecting matplotlib
  Downloading matplotlib-3.10.7-cp312-cp312-
```

```
manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.3-cp312-cp312-
manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (5.5 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.61.0-cp312-cp312-
manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_5_x86_6
4.whl.metadata (113 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.9-cp312-cp312-
manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (6.3 kB)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.12/site-
packages (from matplotlib) (2.3.5)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (24.2)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-12.0.0-cp312-cp312-
manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (8.8 kB)
Collecting pyparsing>=3 (from matplotlib)
  Downloading pyparsing-3.2.5-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Downloading
matplotlib-3.10.7-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl
(8.7 MB)
                         8.7/8.7 MB
102.4 MB/s eta 0:00:00
Downloading
contourpy-1.3.3-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (362
kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.61.0-cp312-cp312-
manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_5_x86_6
4.whl (4.9 MB)
                         4.9/4.9 MB
114.7 MB/s eta 0:00:00
Downloading
kiwisolver-1.4.9-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (1.5
MB)
                         1.5/1.5 MB
77.5 MB/s eta 0:00:00
Downloading
pillow-12.0.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (7.0
MB)
```

```
                          7.0/7.0 MB
126.5 MB/s eta 0:00:00
Downloading pyparsing-3.2.5-py3-none-any.whl (113 kB)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler,
contourpy, matplotlib
Successfully installed contourpy-1.3.3 cycler-0.12.1 fonttools-4.61.0
kiwisolver-1.4.9 matplotlib-3.10.7 pillow-12.0.0 pyparsing-3.2.5
```

[58]:
```python
import matplotlib.pyplot as plt
import pandas as pd

def make_graph(stock_data, revenue_data, company):
    # Convert the Date column to datetime in both DataFrames
    stock_data["Date"] = pd.to_datetime(stock_data["Date"])
    revenue_data["Date"] = pd.to_datetime(revenue_data["Date"])

    # Filter both datasets to only include data up to June 2021
    stock_data = stock_data[stock_data["Date"] <= "2021-06-01"]
    revenue_data = revenue_data[revenue_data["Date"] <= "2021-06-01"]

    # Create Figure
    fig, ax1 = plt.subplots(figsize=(14, 6))

    # Plot stock price
    ax1.plot(stock_data["Date"], stock_data["Close"], color="tab:blue")
    ax1.set_xlabel("Date")
    ax1.set_ylabel("Stock Price (USD)", color="tab:blue")
    ax1.tick_params(axis="y", labelcolor="tab:blue")

    # Create second axis
    ax2 = ax1.twinx()

    # Plot revenue
    ax2.plot(revenue_data["Date"], revenue_data["Revenue"], color="tab:red")
    ax2.set_ylabel("Revenue (USD Millions)", color="tab:red")
    ax2.tick_params(axis="y", labelcolor="tab:red")

    # Title
    plt.title(f"{company} Stock Price vs Revenue")

    plt.show()
```

[60]:
```python
import matplotlib.pyplot as plt
import pandas as pd

def make_graph(stock_data, revenue_data, company):
    # Convert the Date column to datetime in both DataFrames
    stock_data["Date"] = pd.to_datetime(stock_data["Date"])
    revenue_data["Date"] = pd.to_datetime(revenue_data["Date"])

    # Filter both datasets to only include data up to June 2021
    stock_data = stock_data[stock_data["Date"] <= "2021-06-01"]
    revenue_data = revenue_data[revenue_data["Date"] <= "2021-06-01"]

    # Create Figure
    fig, ax1 = plt.subplots(figsize=(14, 6))

    # Plot stock price
    ax1.plot(stock_data["Date"], stock_data["Close"], color="tab:blue")
    ax1.set_xlabel("Date")
    ax1.set_ylabel("Stock Price (USD)", color="tab:blue")
    ax1.tick_params(axis="y", labelcolor="tab:blue")

    # Create second axis
    ax2 = ax1.twinx()

    # Plot revenue
    ax2.plot(revenue_data["Date"], revenue_data["Revenue"], color="tab:red")
    ax2.set_ylabel("Revenue (USD Millions)", color="tab:red")
    ax2.tick_params(axis="y", labelcolor="tab:red")

    # Title
    plt.title(f"{company} Stock Price vs Revenue")
    plt.show()
```

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
```

[62]:
```
# Diagnostic: check that variables and function exist
missing = []
try:
    make_graph
except NameError:
    missing.append("make_graph (function)")

if 'gme_data' not in globals():
    missing.append("gme_data (stock dataframe)")
```

```
if 'gme_revenue' not in globals():
    missing.append("gme_revenue (revenue dataframe)")

if missing:
    print("Missing items:", missing)
    # show heads if present
    if 'gme_data' in globals():
        print("\ngme_data preview:")
        display(gme_data.head())
    if 'gme_revenue' in globals():
        print("\ngme_revenue preview:")
        display(gme_revenue.head())
else:
    print("All items present. You can re-run the make_graph call.")
```

```
All items present. You can re-run the make_graph call.
```

[64]:
```
gme_data.head()
gme_data.columns
```

[64]:
```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Dividends', 'Stock Splits'],
dtype='object')
```

[65]:
```
gme_revenue.head()
gme_revenue.columns
```

[65]:
```
Index(['Date', 'Revenue'], dtype='object')
```

[67]:
```
gme_data = gme_data.reset_index()
```

[70]:
```
gme_data = gme_data.rename(columns={
    "date": "Date",
    "Date ": "Date",
    "Unnamed: 0": "Date"
})
```

[73]:
```
# Diagnostic + automatic fix for missing "Date" column, then call␣
 ↪make_graph(gme_data,gme_revenue,"GameStop")
import pandas as pd
from IPython.display import display

# 1) show current state for debugging (printed so you can paste back if it␣
 ↪fails)
print("==== gme_data.columns ====")
print(gme_data.columns.tolist() if 'gme_data' in globals() else "gme_data not␣
 ↪found")
if 'gme_data' in globals():
    display(gme_data.head())
```

```
print("\n==== gme_revenue.columns ====")
print(gme_revenue.columns.tolist() if 'gme_revenue' in globals() else␣
 ↪"gme_revenue not found")
if 'gme_revenue' in globals():
    display(gme_revenue.head())

# 2) helper to find/normalize a date column in a dataframe
def ensure_date_column(df, df_name="df"):
    df = df.copy()
    # if there is already a Date column, normalize name
    cols_lower = [c.lower() if isinstance(c, str) else c for c in df.columns]
    if "date" in cols_lower:
        # find actual column name (case-insensitive)
        date_col = [c for c in df.columns if str(c).lower() == "date"][0]
        df = df.rename(columns={date_col: "Date"})
        df["Date"] = pd.to_datetime(df["Date"], errors="coerce")
        return df

    # If date is index, reset index (index name might be 'Date' or unnamed)
    if not "Date" in df.columns:
        # reset_index will create a column from the index; safe to do
        df = df.reset_index()
        # After reset, check again
        cols_lower = [c.lower() if isinstance(c, str) else c for c in df.
 ↪columns]
        if "date" in cols_lower:
            date_col = [c for c in df.columns if str(c).lower() == "date"][0]
            df = df.rename(columns={date_col: "Date"})
            df["Date"] = pd.to_datetime(df["Date"], errors="coerce")
            return df

    # Try to find a column with many parseable datetimes
    candidate = None
    for c in df.columns:
        try:
            parsed = pd.to_datetime(df[c], errors="coerce")
            non_na = parsed.notna().sum()
            if non_na > len(df) * 0.5:
                candidate = c
                break
        except Exception:
            continue

    if candidate is not None:
        df = df.rename(columns={candidate: "Date"})
        df["Date"] = pd.to_datetime(df["Date"], errors="coerce")
        return df

    # If nothing found, raise informative error with samples
    raise RuntimeError(f"No Date-like column found automatically in {df_name}. "
                       "Please paste the output of gme_data.columns and␣
 ↪gme_data.head() here.")

# 3) Apply fixes to gme_data and gme_revenue
if 'gme_data' not in globals():
    raise RuntimeError("gme_data not present in the notebook. Run the GME␣
 ↪extraction cell first.")

if 'gme_revenue' not in globals():
    raise RuntimeError("gme_revenue not present in the notebook. Run the␣
 ↪GameStop revenue extraction cell first.")

# Normalize both DataFrames
gme_data = ensure_date_column(gme_data, "gme_data")
gme_revenue = ensure_date_column(gme_revenue, "gme_revenue")

# Drop rows where Date could not be parsed
gme_data = gme_data.dropna(subset=["Date"]).reset_index(drop=True)
gme_revenue = gme_revenue.dropna(subset=["Date"]).reset_index(drop=True)

print("\nAfter normalization:")
print("gme_data.columns:", gme_data.columns.tolist())
print("gme_revenue.columns:", gme_revenue.columns.tolist())
display(gme_data.head())
display(gme_revenue.head())

# 4) final quick clean on revenue (ensure numeric)
gme_revenue["Revenue"] = gme_revenue["Revenue"].astype(str).str.
 ↪replace(r"[\$,]", "", regex=True)
gme_revenue["Revenue"] = pd.to_numeric(gme_revenue["Revenue"], errors="coerce")
gme_revenue = gme_revenue.dropna(subset=["Revenue"]).reset_index(drop=True)

# 5) Call the lab-provided make_graph function
try:
    make_graph(gme_data, gme_revenue, "GameStop")
except NameError:
    raise RuntimeError("Function make_graph is not defined. Run or paste the␣
 ↪make_graph definition cell provided by the lab above this cell.")
except Exception as e:
    # if plotting fails, show helpful debug info
    raise RuntimeError(f"make_graph raised an error: {e}")
```

```
==== gme_data.columns ====
['level_0', 'index', 'Open', 'High', 'Low', 'Close', 'Volume', 'Dividends',
'Stock Splits']
```

```
   level_0  index      Open      High       Low     Close    Volume  \
0        0      0  1.620129  1.693350  1.603296  1.691667  76216000
1        1      1  1.712707  1.716074  1.670626  1.683250  11021600
2        2      2  1.683250  1.687458  1.658002  1.674834   8389600
3        3      3  1.666418  1.666418  1.578047  1.607504   7410400
4        4      4  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
```

```
==== gme_revenue.columns ====
['Date', 'Revenue']
```

```
         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
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_408/1738603066.py in ?()
     65 if 'gme_revenue' not in globals():
     66     raise RuntimeError("gme_revenue not present in the notebook. Run the␣
  ↪GameStop revenue extraction cell first.")
     67
     68 # Normalize both DataFrames
---> 69 gme_data = ensure_date_column(gme_data, "gme_data")
     70 gme_revenue = ensure_date_column(gme_revenue, "gme_revenue")
     71
     72 # Drop rows where Date could not be parsed

/tmp/ipykernel_408/1738603066.py in ?(df, df_name)
     27
     28     # If date is index, reset index (index name might be 'Date' or␣
  ↪unnamed)
     29     if not "Date" in df.columns:
     30         # reset_index will create a column from the index; safe to do
---> 31         df = df.reset_index()
     32         # After reset, check again
     33          cols_lower = [c.lower() if isinstance(c, str) else c for c in d
 ↪columns]
     34          if "date" in cols_lower:

/opt/conda/lib/python3.12/site-packages/pandas/core/frame.py in ?(self, level,␣
 ↪drop, inplace, col_level, col_fill, allow_duplicates, names)
   6490                      level_values = algorithms.take(
   6491                          level_values, lab, allow_fill=True,␣
 ↪fill_value=lev._na_value
   6492                      )
   6493
-> 6494                  new_obj.insert(
   6495                      0,
   6496                      name,
   6497                      level_values,

/opt/conda/lib/python3.12/site-packages/pandas/core/frame.py in ?(self, loc,␣
 ↪column, value, allow_duplicates)
   5176                  "'self.flags.allows_duplicate_labels' is False."
   5177              )
   5178          if not allow_duplicates and column in self.columns:
   5179              # Should this be a different kind of error??
-> 5180              raise ValueError(f"cannot insert {column}, already exists")
   5181          if not is_integer(loc):
   5182              raise TypeError("loc must be int")
   5183          # convert non stdlib ints to satisfy typing checks

ValueError: cannot insert level_0, already exists
```

[74]:
```python
# Diagnostic: is gme_data present? show a preview if it is.
from IPython.display import display
if 'gme_data' in globals():
    print("gme_data exists. Columns:")
    print(gme_data.columns.tolist())
    display(gme_data.head())
else:
    print("gme_data is NOT present in the notebook.")
```

```
gme_data exists. Columns:
['level_0', 'index', 'Open', 'High', 'Low', 'Close', 'Volume', 'Dividends',
'Stock Splits']

   level_0  index      Open      High       Low     Close    Volume  \
0        0      0  1.620129  1.693350  1.603296  1.691667  76216000
1        1      1  1.712707  1.716074  1.670626  1.683250  11021600
2        2      2  1.683250  1.687458  1.658002  1.674834   8389600
3        3      3  1.666418  1.666418  1.578047  1.607504   7410400
```

```
4         4      4  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
```

[75]:
```
# Fix for gme_data: re-download proper GameStop history and call the lab␣
↪make_graph
import sys
import pandas as pd
from IPython.display import display

# 1) ensure we can fetch yfinance
try:
    import yfinance as yf
except Exception:
    try:
        !pip install --quiet yfinance
        import yfinance as yf
    except Exception as e:
        raise RuntimeError("yfinance not available and pip install failed.␣
↪Paste the pip error here.") from e

# 2) download GME history and create a clean gme_data with Date column
try:
    gme = yf.Ticker("GME")
    gme_data = gme.history(period="max").reset_index()   # Date will be a column
except Exception as e:
    # fallback to pandas_datareader if yfinance.history fails
    try:
        from pandas_datareader import data as pdr
        yf.pdr_override()
        gme_data = pdr.get_data_yahoo("GME", start="2000-01-01").reset_index()
    except Exception as e2:
        raise RuntimeError("Failed to download GME data with both yfinance and␣
↪pandas_datareader.") from e2

# 3) normalize Date and show preview
gme_data['Date'] = pd.to_datetime(gme_data['Date'], errors='coerce')
gme_data = gme_data.dropna(subset=['Date']).reset_index(drop=True)

print("gme_data columns:", gme_data.columns.tolist())
display(gme_data.head())
```

```
# 4) sanity-check: is gme_revenue present and OK?
if 'gme_revenue' not in globals():
    print("\nNote: gme_revenue not found. Run the GameStop revenue extraction␣
↪cell first.")
else:
    print("\ngme_revenue preview:")
    display(gme_revenue.head())

# 5) Try to call the lab-provided plotting function (if it exists)
try:
    make_graph(gme_data, gme_revenue, "GameStop")
except NameError:
    print("\nmake_graph is not defined in your notebook. If the lab provided a␣
↪make_graph cell, run it first.")
except Exception as e:
    print("\nmake_graph raised an exception:", e)
```

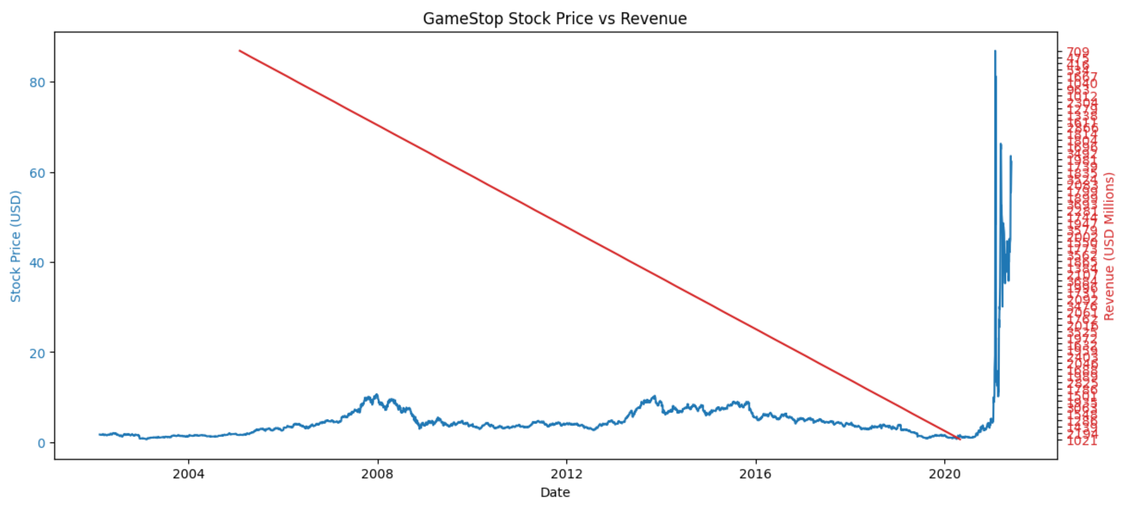
```
gme_data columns: ['Date', 'Open', 'High', 'Low', 'Close', 'Volume',
'Dividends', 'Stock Splits']

                       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  1.716073  1.670625  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  1.615921  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


gme_revenue preview:

         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
```



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