

Revenue Data and Building a Dashboard

January 22, 2026

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.

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



```
Collecting yfinance
  Downloading yfinance-1.0-py2.py3-none-any.whl.metadata (6.0 kB)
Collecting pandas>=1.3.0 (from yfinance)
  Downloading pandas-3.0.0-cp312-cp312-
manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (79 kB)
Collecting numpy>=1.16.5 (from yfinance)
  Downloading
numpy-2.4.1-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata
(6.6 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-3.19.0-py3-none-any.whl.metadata (7.0 kB)
Requirement already satisfied: beautifulsoup4>=4.11.1 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.12.3)
Collecting curl_cffi<0.14,>=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-6.33.4-cp39-abi3-manylinux2014_x86_64.whl.metadata (593
bytes)
Collecting websockets>=13.0 (from yfinance)
  Downloading websockets-16.0-cp312-cp312-
manylinux1_x86_64.manylinux_2_28_x86_64.manylinux_2_5_x86_64.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<0.14,>=0.7->yfinance) (1.17.1)
Requirement already satisfied: certifi>=2024.2.2 in
/opt/conda/lib/python3.12/site-packages (from curl_cffi<0.14,>=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)
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<0.14,>=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-1.0-py2.py3-none-any.whl (127 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

82.4 MB/s eta 0:00:00

```
Downloading
numpy-2.4.1-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (16.4
MB)
    16.4/16.4 MB
173.4 MB/s eta 0:00:00
Downloading
pandas-3.0.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (10.9
MB)
    10.9/10.9 MB
183.3 MB/s eta 0:00:00
Downloading peewee-3.19.0-py3-none-any.whl (411 kB)
Downloading protobuf-6.33.4-cp39-abi3-manylinux2014_x86_64.whl (323 kB)
Downloading websockets-16.0-cp312-cp312-
manylinux1_x86_64.manylinux_2_28_x86_64.manylinux_2_5_x86_64.whl (184 kB)
Building wheels for collected packages: multitasking
    Building wheel for multitasking (setup.py) ... one
        Created wheel for multitasking: filename=multitasking-0.0.12-py3-none-
any.whl size=15605
sha256=ebb4a4ce8aebe8a1332c094d5c3df664a16bdf8a88a5ef9e32df44d01d0cc577
    Stored in directory: /home/jupyterlab/.cache/pip/wheels/cc/bd/6f/664d62c99327a
beef7d86489e6631cbf45b56fbf7ef1d6ef00
Successfully built multitasking
Installing collected packages: peewee, multitasking, websockets, protobuf,
numpy, pandas, curl_cffi, yfinance
Successfully installed curl_cffi-0.13.0 multitasking-0.0.12 numpy-2.4.1
pandas-3.0.0 peewee-3.19.0 protobuf-6.33.4 websockets-16.0 yfinance-1.0
Collecting bs4
    Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.12/site-
packages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/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/python3.12/site-
packages (5.10.4)
Requirement already satisfied: fastjsonschema>=2.15 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (2.21.1)
Requirement already satisfied: jsonschema>=2.6 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (4.23.0)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (5.7.2)
Requirement already satisfied: traitlets>=5.1 in /opt/conda/lib/python3.12/site-
packages (from nbformat) (5.14.3)
Requirement already satisfied: attrs>=22.2.0 in /opt/conda/lib/python3.12/site-
packages (from jsonschema>=2.6->nbformat) (25.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
```

```
/opt/conda/lib/python3.12/site-packages (from jsonschema>=2.6->nbformat)
(2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in
/opt/conda/lib/python3.12/site-packages (from jsonschema>=2.6->nbformat)
(0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in /opt/conda/lib/python3.12/site-
packages (from jsonschema>=2.6->nbformat) (0.22.3)
Requirement already satisfied: platformdirs>=2.5 in
/opt/conda/lib/python3.12/site-packages (from jupyter-
core!=5.0.*,>=4.12->nbformat) (4.3.6)
Requirement already satisfied: typing-extensions>=4.4.0 in
/opt/conda/lib/python3.12/site-packages (from
referencing>=0.28.4->jsonschema>=2.6->nbformat) (4.12.2)
Collecting matplotlib
    Downloading matplotlib-3.10.8-cp312-cp312-
manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (52 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-4.61.1-cp312-cp312-
manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_5_x86_6
4.whl.metadata (114 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) (2.4.1)
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-12.1.0-cp312-cp312-
manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (8.8 kB)
Collecting pyparsing>=3 (from matplotlib)
    Downloading pyparsing-3.3.2-py3-none-any.whl.metadata (5.8 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-3.10.8-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl
(8.7 MB)
    8.7/8.7 MB
168.7 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-4.61.1-cp312-cp312-
manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_5_x86_6
4.whl (5.0 MB)                                5.0/5.0 MB
154.6 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
99.7 MB/s eta 0:00:00
Downloading
pillow-12.1.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (7.0
MB)                                         7.0/7.0 MB
162.8 MB/s eta 0:00:00
Downloading pyparsing-3.3.2-py3-none-any.whl (122 kB)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler,
contourpy, matplotlib
Successfully installed contourpy-1.3.3 cycler-0.12.1 fonttools-4.61.1
kiwisolver-1.4.9 matplotlib-3.10.8 pillow-12.1.0 pyparsing-3.3.2

```

```
[14]: 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.

```
[15]: 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.

```
[16]: # 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`.

[17]: `tesla = yf.Ticker("TSLA")`

[11]:

```
/opt/conda/lib/python3.12/site-packages/yfinance/scrapers/history.py:201:
PandasWarning: Timestamp.utcnow is deprecated and will be removed in a future
version. Use Timestamp.now('UTC') instead.
dt_now = pd.Timestamp.utcnow()
```

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.

[18]: `tesla_data = tesla.history(period="max")`

```
/opt/conda/lib/python3.12/site-packages/yfinance/scrapers/history.py:201:  
PandasWarning: Timestamp.utcnow is deprecated and will be removed in a future  
version. Use Timestamp.now('UTC') instead.  
    dt_now = pd.Timestamp.utcnow()
```

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.

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

```
[19]:          Date      Open      High      Low     Close \\\n0 2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667\n1 2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667\n2 2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000\n3 2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000\n4 2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000\n\n          Volume  Dividends  Stock Splits\n0   281494500        0.0        0.0\n1   257806500        0.0        0.0\n2   123282000        0.0        0.0\n3   77097000         0.0        0.0\n4   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/IBMDriverSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm>. Save the text of the response as a variable named `html_data`.

```
[20]: url = "https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue"  
html_data = requests.get(url).text
```

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

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

```
-----  
FeatureNotFound                                     Traceback (most recent call last)  
Cell In[21], line 1  
----> 1 soup = BeautifulSoup(html_data, "html5lib")  
  
File /opt/conda/lib/python3.12/site-packages/bs4/_init_.py:250, in __  
    247     self._parser = self._get_parser()  
    248     builder_class = builder_registry.lookup(*features)  
    249     if self._parser == 'lxml':  
    250         builder_class = LXMLBuilder  
    251     else:  
    252         builder_class = STANDARD_BUILDERS.get(self._parser)  
    253     if builder_class is None:  
    254         raise FeatureNotFound("Parser %s is not supported" % self._parser)  
    255     self._builder = builder_class()  
    256     self._parser = self._parser.lower()  
    257     self._features = features  
    258     self._feature_set = set(features)  
    259     self._feature_set.add("html5lib")  
    260     self._feature_set.add("xml")  
    261     self._feature_set.add("xmlns")  
    262     self._feature_set.add("standalone")  
    263     self._feature_set.add("base")  
    264     self._feature_set.add("schemas")  
    265     self._feature_set.add("noempty")  
    266     self._feature_set.add("minmaxint")  
    267     self._feature_set.add("minmaxfloat")  
    268     self._feature_set.add("minmaxdouble")  
    269     self._feature_set.add("minmaxlong")  
    270     self._feature_set.add("minmaxint32")  
    271     self._feature_set.add("minmaxfloat32")  
    272     self._feature_set.add("minmaxdouble32")  
    273     self._feature_set.add("minmaxint64")  
    274     self._feature_set.add("minmaxfloat64")  
    275     self._feature_set.add("minmaxdouble64")  
    276     self._feature_set.add("minmaxint128")  
    277     self._feature_set.add("minmaxfloat128")  
    278     self._feature_set.add("minmaxdouble128")  
    279     self._feature_set.add("minmaxint256")  
    280     self._feature_set.add("minmaxfloat256")  
    281     self._feature_set.add("minmaxdouble256")  
    282     self._feature_set.add("minmaxint512")  
    283     self._feature_set.add("minmaxfloat512")  
    284     self._feature_set.add("minmaxdouble512")  
    285     self._feature_set.add("minmaxint1024")  
    286     self._feature_set.add("minmaxfloat1024")  
    287     self._feature_set.add("minmaxdouble1024")  
    288     self._feature_set.add("minmaxint2048")  
    289     self._feature_set.add("minmaxfloat2048")  
    290     self._feature_set.add("minmaxdouble2048")  
    291     self._feature_set.add("minmaxint4096")  
    292     self._feature_set.add("minmaxfloat4096")  
    293     self._feature_set.add("minmaxdouble4096")  
    294     self._feature_set.add("minmaxint8192")  
    295     self._feature_set.add("minmaxfloat8192")  
    296     self._feature_set.add("minmaxdouble8192")  
    297     self._feature_set.add("minmaxint16384")  
    298     self._feature_set.add("minmaxfloat16384")  
    299     self._feature_set.add("minmaxdouble16384")  
    300     self._feature_set.add("minmaxint32768")  
    301     self._feature_set.add("minmaxfloat32768")  
    302     self._feature_set.add("minmaxdouble32768")  
    303     self._feature_set.add("minmaxint65536")  
    304     self._feature_set.add("minmaxfloat65536")  
    305     self._feature_set.add("minmaxdouble65536")  
    306     self._feature_set.add("minmaxint131072")  
    307     self._feature_set.add("minmaxfloat131072")  
    308     self._feature_set.add("minmaxdouble131072")  
    309     self._feature_set.add("minmaxint262144")  
    310     self._feature_set.add("minmaxfloat262144")  
    311     self._feature_set.add("minmaxdouble262144")  
    312     self._feature_set.add("minmaxint524288")  
    313     self._feature_set.add("minmaxfloat524288")  
    314     self._feature_set.add("minmaxdouble524288")  
    315     self._feature_set.add("minmaxint1048576")  
    316     self._feature_set.add("minmaxfloat1048576")  
    317     self._feature_set.add("minmaxdouble1048576")  
    318     self._feature_set.add("minmaxint2097152")  
    319     self._feature_set.add("minmaxfloat2097152")  
    320     self._feature_set.add("minmaxdouble2097152")  
    321     self._feature_set.add("minmaxint4194304")  
    322     self._feature_set.add("minmaxfloat4194304")  
    323     self._feature_set.add("minmaxdouble4194304")  
    324     self._feature_set.add("minmaxint8388608")  
    325     self._feature_set.add("minmaxfloat8388608")  
    326     self._feature_set.add("minmaxdouble8388608")  
    327     self._feature_set.add("minmaxint16777216")  
    328     self._feature_set.add("minmaxfloat16777216")  
    329     self._feature_set.add("minmaxdouble16777216")  
    330     self._feature_set.add("minmaxint33554432")  
    331     self._feature_set.add("minmaxfloat33554432")  
    332     self._feature_set.add("minmaxdouble33554432")  
    333     self._feature_set.add("minmaxint67108864")  
    334     self._feature_set.add("minmaxfloat67108864")  
    335     self._feature_set.add("minmaxdouble67108864")  
    336     self._feature_set.add("minmaxint134217728")  
    337     self._feature_set.add("minmaxfloat134217728")  
    338     self._feature_set.add("minmaxdouble134217728")  
    339     self._feature_set.add("minmaxint268435456")  
    340     self._feature_set.add("minmaxfloat268435456")  
    341     self._feature_set.add("minmaxdouble268435456")  
    342     self._feature_set.add("minmaxint536870912")  
    343     self._feature_set.add("minmaxfloat536870912")  
    344     self._feature_set.add("minmaxdouble536870912")  
    345     self._feature_set.add("minmaxint1073741824")  
    346     self._feature_set.add("minmaxfloat1073741824")  
    347     self._feature_set.add("minmaxdouble1073741824")  
    348     self._feature_set.add("minmaxint2147483648")  
    349     self._feature_set.add("minmaxfloat2147483648")  
    350     self._feature_set.add("minmaxdouble2147483648")  
    351     self._feature_set.add("minmaxint4294967296")  
    352     self._feature_set.add("minmaxfloat4294967296")  
    353     self._feature_set.add("minmaxdouble4294967296")  
    354     self._feature_set.add("minmaxint8589934592")  
    355     self._feature_set.add("minmaxfloat8589934592")  
    356     self._feature_set.add("minmaxdouble8589934592")  
    357     self._feature_set.add("minmaxint17179869184")  
    358     self._feature_set.add("minmaxfloat17179869184")  
    359     self._feature_set.add("minmaxdouble17179869184")  
    360     self._feature_set.add("minmaxint34359738368")  
    361     self._feature_set.add("minmaxfloat34359738368")  
    362     self._feature_set.add("minmaxdouble34359738368")  
    363     self._feature_set.add("minmaxint68719476736")  
    364     self._feature_set.add("minmaxfloat68719476736")  
    365     self._feature_set.add("minmaxdouble68719476736")  
    366     self._feature_set.add("minmaxint137438953472")  
    367     self._feature_set.add("minmaxfloat137438953472")  
    368     self._feature_set.add("minmaxdouble137438953472")  
    369     self._feature_set.add("minmaxint274877906944")  
    370     self._feature_set.add("minmaxfloat274877906944")  
    371     self._feature_set.add("minmaxdouble274877906944")  
    372     self._feature_set.add("minmaxint549755813888")  
    373     self._feature_set.add("minmaxfloat549755813888")  
    374     self._feature_set.add("minmaxdouble549755813888")  
    375     self._feature_set.add("minmaxint1099511627776")  
    376     self._feature_set.add("minmaxfloat1099511627776")  
    377     self._feature_set.add("minmaxdouble1099511627776")  
    378     self._feature_set.add("minmaxint2199023255552")  
    379     self._feature_set.add("minmaxfloat2199023255552")  
    380     self._feature_set.add("minmaxdouble2199023255552")  
    381     self._feature_set.add("minmaxint4398046511104")  
    382     self._feature_set.add("minmaxfloat4398046511104")  
    383     self._feature_set.add("minmaxdouble4398046511104")  
    384     self._feature_set.add("minmaxint8796093022208")  
    385     self._feature_set.add("minmaxfloat8796093022208")  
    386     self._feature_set.add("minmaxdouble8796093022208")  
    387     self._feature_set.add("minmaxint17592186044416")  
    388     self._feature_set.add("minmaxfloat17592186044416")  
    389     self._feature_set.add("minmaxdouble17592186044416")  
    390     self._feature_set.add("minmaxint35184372088832")  
    391     self._feature_set.add("minmaxfloat35184372088832")  
    392     self._feature_set.add("minmaxdouble35184372088832")  
    393     self._feature_set.add("minmaxint70368744177664")  
    394     self._feature_set.add("minmaxfloat70368744177664")  
    395     self._feature_set.add("minmaxdouble70368744177664")  
    396     self._feature_set.add("minmaxint140737488355328")  
    397     self._feature_set.add("minmaxfloat140737488355328")  
    398     self._feature_set.add("minmaxdouble140737488355328")  
    399     self._feature_set.add("minmaxint281474976710656")  
    400     self._feature_set.add("minmaxfloat281474976710656")  
    401     self._feature_set.add("minmaxdouble281474976710656")  
    402     self._feature_set.add("minmaxint562949953421312")  
    403     self._feature_set.add("minmaxfloat562949953421312")  
    404     self._feature_set.add("minmaxdouble562949953421312")  
    405     self._feature_set.add("minmaxint1125899906842624")  
    406     self._feature_set.add("minmaxfloat1125899906842624")  
    407     self._feature_set.add("minmaxdouble1125899906842624")  
    408     self._feature_set.add("minmaxint2251799813685248")  
    409     self._feature_set.add("minmaxfloat2251799813685248")  
    410     self._feature_set.add("minmaxdouble2251799813685248")  
    411     self._feature_set.add("minmaxint4503599627368496")  
    412     self._feature_set.add("minmaxfloat4503599627368496")  
    413     self._feature_set.add("minmaxdouble4503599627368496")  
    414     self._feature_set.add("minmaxint9007199254736992")  
    415     self._feature_set.add("minmaxfloat9007199254736992")  
    416     self._feature_set.add("minmaxdouble9007199254736992")  
    417     self._feature_set.add("minmaxint18014398509473984")  
    418     self._feature_set.add("minmaxfloat18014398509473984")  
    419     self._feature_set.add("minmaxdouble18014398509473984")  
    420     self._feature_set.add("minmaxint36028797018947968")  
    421     self._feature_set.add("minmaxfloat36028797018947968")  
    422     self._feature_set.add("minmaxdouble36028797018947968")  
    423     self._feature_set.add("minmaxint72057594037895936")  
    424     self._feature_set.add("minmaxfloat72057594037895936")  
    425     self._feature_set.add("minmaxdouble72057594037895936")  
    426     self._feature_set.add("minmaxint14411518807579187")  
    427     self._feature_set.add("minmaxfloat14411518807579187")  
    428     self._feature_set.add("minmaxdouble14411518807579187")  
    429     self._feature_set.add("minmaxint28823037615158374")  
    430     self._feature_set.add("minmaxfloat28823037615158374")  
    431     self._feature_set.add("minmaxdouble28823037615158374")  
    432     self._feature_set.add("minmaxint57646075230316748")  
    433     self._feature_set.add("minmaxfloat57646075230316748")  
    434     self._feature_set.add("minmaxdouble57646075230316748")  
    435     self._feature_set.add("minmaxint115292150460633496")  
    436     self._feature_set.add("minmaxfloat115292150460633496")  
    437     self._feature_set.add("minmaxdouble115292150460633496")  
    438     self._feature_set.add("minmaxint230584300921266992")  
    439     self._feature_set.add("minmaxfloat230584300921266992")  
    440     self._feature_set.add("minmaxdouble230584300921266992")  
    441     self._feature_set.add("minmaxint461168601842533984")  
    442     self._feature_set.add("minmaxfloat461168601842533984")  
    443     self._feature_set.add("minmaxdouble461168601842533984")  
    444     self._feature_set.add("minmaxint922337203685067968")  
    445     self._feature_set.add("minmaxfloat922337203685067968")  
    446     self._feature_set.add("minmaxdouble922337203685067968")  
    447     self._feature_set.add("minmaxint1844674407370135936")  
    448     self._feature_set.add("minmaxfloat1844674407370135936")  
    449     self._feature_set.add("minmaxdouble1844674407370135936")  
    450     self._feature_set.add("minmaxint3689348814740271872")  
    451     self._feature_set.add("minmaxfloat3689348814740271872")  
    452     self._feature_set.add("minmaxdouble3689348814740271872")  
    453     self._feature_set.add("minmaxint7378697629480543744")  
    454     self._feature_set.add("minmaxfloat7378697629480543744")  
    455     self._feature_set.add("minmaxdouble7378697629480543744")  
    456     self._feature_set.add("minmaxint14757395258961087488")  
    457     self._feature_set.add("minmaxfloat14757395258961087488")  
    458     self._feature_set.add("minmaxdouble14757395258961087488")  
    459     self._feature_set.add("minmaxint29514790517922174976")  
    460     self._feature_set.add("minmaxfloat29514790517922174976")  
    461     self._feature_set.add("minmaxdouble29514790517922174976")  
    462     self._feature_set.add("minmaxint59029581035844349952")  
    463     self._feature_set.add("minmaxfloat59029581035844349952")  
    464     self._feature_set.add("minmaxdouble59029581035844349952")  
    465     self._feature_set.add("minmaxint11805916207168869984")  
    466     self._feature_set.add("minmaxfloat11805916207168869984")  
    467     self._feature_set.add("minmaxdouble11805916207168869984")  
    468     self._feature_set.add("minmaxint23611832414337739968")  
    469     self._feature_set.add("minmaxfloat23611832414337739968")  
    470     self._feature_set.add("minmaxdouble23611832414337739968")  
    471     self._feature_set.add("minmaxint47223664828675479936")  
    472     self._feature_set.add("minmaxfloat47223664828675479936")  
    473     self._feature_set.add("minmaxdouble47223664828675479936")  
    474     self._feature_set.add("minmaxint94447329657350959872")  
    475     self._feature_set.add("minmaxfloat94447329657350959872")  
    476     self._feature_set.add("minmaxdouble94447329657350959872")  
    477     self._feature_set.add("minmaxint18889465931470191976")  
    478     self._feature_set.add("minmaxfloat18889465931470191976")  
    479     self._feature_set.add("minmaxdouble18889465931470191976")  
    480     self._feature_set.add("minmaxint37778931862940383952")  
    481     self._feature_set.add("minmaxfloat37778931862940383952")  
    482     self._feature_set.add("minmaxdouble37778931862940383952")  
    483     self._feature_set.add("minmaxint75557863725880767904")  
    484     self._feature_set.add("minmaxfloat75557863725880767904")  
    485     self._feature_set.add("minmaxdouble75557863725880767904")  
    486     self._feature_set.add("minmaxint15111572745176153808")  
    487     self._feature_set.add("minmaxfloat15111572745176153808")  
    488     self._feature_set.add("minmaxdouble15111572745176153808")  
    489     self._feature_set.add("minmaxint30223145490352307616")  
    490     self._feature_set.add("minmaxfloat30223145490352307616")  
    491     self._feature_set.add("minmaxdouble30223145490352307616")  
    492     self._feature_set.add("minmaxint60446290980704615232")  
    493     self._feature_set.add("minmaxfloat60446290980704615232")  
    494     self._feature_set.add("minmaxdouble60446290980704615232")  
    495     self._feature_set.add("minmaxint12089258196140923046")  
    496     self._feature_set.add("minmaxfloat12089258196140923046")  
    497     self._feature_set.add("minmaxdouble12089258196140923046")  
    498     self._feature_set.add("minmaxint24178516392281846092")  
    499     self._feature_set.add("minmaxfloat24178516392281846092")  
    500     self._feature_set.add("minmaxdouble24178516392281846092")  
    501     self._feature_set.add("minmaxint48357032784563692184")  
    502     self._feature_set.add("minmaxfloat48357032784563692184")  
    503     self._feature_set.add("minmaxdouble48357032784563692184")  
    504     self._feature_set.add("minmaxint96714065569127384368")  
    505     self._feature_set.add("minmaxfloat96714065569127384368")  
    506     self._feature_set.add("minmaxdouble96714065569127384368")  
    507     self._feature_set.add("minmaxint19342813113825476872")  
    508     self._feature_set.add("minmaxfloat19342813113825476872")  
    509     self._feature_set.add("minmaxdouble19342813113825476872")  
    510     self._feature_set.add("minmaxint38685626227650953744")  
    511     self._feature_set.add("minmaxfloat38685626227650953744")  
    512     self._feature_set.add("minmaxdouble38685626227650953744")  
    513     self._feature_set.add("minmaxint77371252455301907488")  
    514     self._feature_set.add("minmaxfloat77371252455301907488")  
    515     self._feature_set.add("minmaxdouble77371252455301907488")  
    516     self._feature_set.add("minmaxint15474250491060381496")  
    517     self._feature_set.add("minmaxfloat15474250491060381496")  
    518     self._feature_set.add("minmaxdouble15474250491060381496")  
    519     self._feature_set.add("minmaxint30948500982120762992")  
    520     self._feature_set.add("minmaxfloat30948500982120762992")  
    521     self._feature_set.add("minmaxdouble30948500982120762992")  
    522     self._feature_set.add("minmaxint61897001964241525984")  
    523     self._feature_set.add("minmaxfloat61897001964241525984")  
    524     self._feature_set.add("minmaxdouble61897001964241525984")  
    525     self._feature_set.add("minmaxint12379400392848305192")  
    526     self._feature_set.add("minmaxfloat12379400392848305192")  
    527     self._feature_set.add("minmaxdouble12379400392848305192")  
    528     self._feature_set.add("minmaxint24758800785696610384")  
    529     self._feature_set.add("minmaxfloat24758800785696610384")  
    530     self._feature_set.add("minmaxdouble24758800785696610384")  
    531     self._feature_set.add("minmaxint49517601571393220768")  
    532     self._feature_set.add("minmaxfloat49517601571393220768")  
    533     self._feature_set.add("minmaxdouble49517601571393220768")  
    534     self._feature_set.add("minmaxint99035203142786441536")  
    535     self._feature_set.add("minmaxfloat99035203142786441536")  
    536     self._feature_set.add("minmaxdouble99035203142786441536")  
    537     self._feature_set.add("minmaxint19807040628557288304")  
    538     self._feature_set.add("minmaxfloat19807040628557288304")  
    539     self._feature_set.add("minmaxdouble19807040628557288304")  
    540     self._feature_set.add("minmaxint39614081257114576608")  
    541     self._feature_set.add("minmaxfloat39614081257114576608")  
    542     self._feature_set.add("minmaxdouble39614081257114576608")  
    543     self._feature_set.add("minmaxint79228162514229153216")  
    544     self._feature_set.add("minmaxfloat79228162514229153216")  
    545     self._feature_set.add("minmaxdouble79228162514229153216")  
    546     self._feature_set.add("minmaxint15845632528845830640")  
    547     self._feature_set.add("minmaxfloat15845632528845830640")  
    548     self._feature_set.add("minmaxdouble15845632528845830640")  
    549     self._feature_set.add("minmaxint31691265057691661280")  
    550     self._feature_set.add("minmaxfloat31691265057691661280")  
    551     self._feature_set.add("minmaxdouble31691265057691661280")  
    552     self._feature_set.add("minmaxint63382530115383322560")  
    553     self._feature_set.add("minmaxfloat63382530115383322560")  
    554     self._feature_set.add("minmaxdouble63382530115383322560")  
    555     self._feature_set.add("minmaxint12676506023076644520")  
    556     self._feature_set.add("minmaxfloat12676506023076644520")  
    557     self._feature_set.add("minmaxdouble12676506023076644520")  
    558     self._feature_set.add("minmaxint25353012046153289040")  
    559     self._feature_set.add("minmaxfloat25353012046153289040")  
    560     self._feature_set.add("minmaxdouble25353012046153289040")  
    561     self._feature_set.add("minmaxint50706024092306578080")  
    562     self._feature_set.add("minmaxfloat5070602409
```

```

249     if builder_class is None:
--> 250         raise FeatureNotFound(
251             "Couldn't find a tree builder with the features you "
252             "requested: %s. Do you need to install a parser library?" %
253             ", ".join(features))
255 # At this point either we have a TreeBuilder instance in
256 # builder, or we have a builder_class that we can instantiate
257 # with the remaining **kwargs.
258 if builder is None:

```

`FeatureNotFound`: Couldn't find a tree builder with the features you requested:
 ↪html5lib. Do you need to install a parser library?

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.

```
[23]: tesla_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

for table in soup.find_all('table'):

    if ('Tesla Quarterly Revenue' in table.find('th').text):
        rows = table.find_all('tr')

        for row in rows:
            col = row.find_all('td')
```

```

if col != []:
    date = col[0].text
    revenue = col[1].text.replace(',', '').replace('$', '')

    tesla_revenue = tesla_revenue.append({"Date":date, "Revenue":revenue}, ignore_index=True)

```

```

-----
NameError                                 Traceback (most recent call last)
Cell In[23], line 3
      1 tesla_revenue = pd.DataFrame(columns=['Date', 'Revenue'])
----> 3 for table in soup.find_all('table'):
      5     if ('Tesla Quarterly Revenue' in table.find('th').text):
      6         rows = table.find_all('tr')

NameError: name 'soup' is not defined

```

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

```
[24]: tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',', '$', regex=True)
```

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

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

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

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

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

```
[27]: Empty DataFrame
Columns: [Date, Revenue]
Index: []
```

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.

```
[28]: GameStop = 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.

```
[29]: gme_data = GameStop.history(period="max")
```

```
/opt/conda/lib/python3.12/site-packages/yfinance/scrapers/history.py:201:
PandasWarning: Timestamp.utcnow is deprecated and will be removed in a future
version. Use Timestamp.now('UTC') instead.

    dt_now = pd.Timestamp.utcnow()
```

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.

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

```
[30]:          Date      Open      High      Low     Close   Volume \
0 2002-02-13 00:00:00-05:00  1.620129  1.693350  1.603296  1.691667  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250  11021600
2 2002-02-15 00:00:00-05:00  1.683251  1.687459  1.658002  1.674834  8389600
3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504  7410400
4 2002-02-20 00:00:00-05:00  1.615920  1.662210  1.603296  1.662210  6892800

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

0.5 Question 4: Use Webscraping to Extract GME Revenue Data

Use the `requests` library to download the webpage <https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDriverSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html>. Save the text of the response as a variable named `html_data_2`

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

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

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

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

```
-----
FeatureNotFound                                     Traceback (most recent call last)
Cell In[32], line 1
----> 1 soup = BeautifulSoup(html_data, "html5lib")
```

```

File /opt/conda/lib/python3.12/site-packages/bs4/__init__.py:250, in_
  ↪BeautifulSoup.__init__(self, markup, features, builder, parse_only,_
  ↪from_encoding, exclude_encodings, element_classes, **kwargs)
  248     builder_class = builder_registry.lookup(*features)
  249     if builder_class is None:
--> 250         raise FeatureNotFound(
  251             "Couldn't find a tree builder with the features you "
  252             "requested: %s. Do you need to install a parser library?" %
  253             ", ".join(features))
  255 # At this point either we have a TreeBuilder instance in
  256 # builder, or we have a builder_class that we can instantiate
  257 # with the remaining **kwargs.
  258 if builder is None:

FeatureNotFound: Couldn't find a tree builder with the features you requested:_
  ↪html5lib. Do you need to install a parser library?

```

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

```
[33]: gme_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

for table in soup.find_all('table'):

    if ('GameStop Quarterly Revenue' in table.find('th').text):
        rows = table.find_all('tr')

        for row in rows:
            col = row.find_all('td')

            if col != []:
                date = col[0].text
                revenue = col[1].text.replace(',', '').replace('$', '')

                gme_revenue = gme_revenue.append({'Date':date, 'Revenue':
                    ↪revenue}, ignore_index=True)
```

```
NameError Traceback (most recent call last)
Cell In[33], line 3
      1 gme_revenue = pd.DataFrame(columns=['Date', 'Revenue'])
----> 3 for table in soup.find_all('table'):
      5     if ('GameStop Quarterly Revenue' in table.find('th').text):
      6         rows = table.find_all('tr')

NameError: name 'soup' is not defined
```

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

[34]: `gme_revenue.tail()`

[34]: Empty DataFrame
Columns: [Date, Revenue]
Index: []

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

[36]: `gme_revenue.tail()`

[36]: Empty DataFrame
Columns: [Date, Revenue]
Index: []

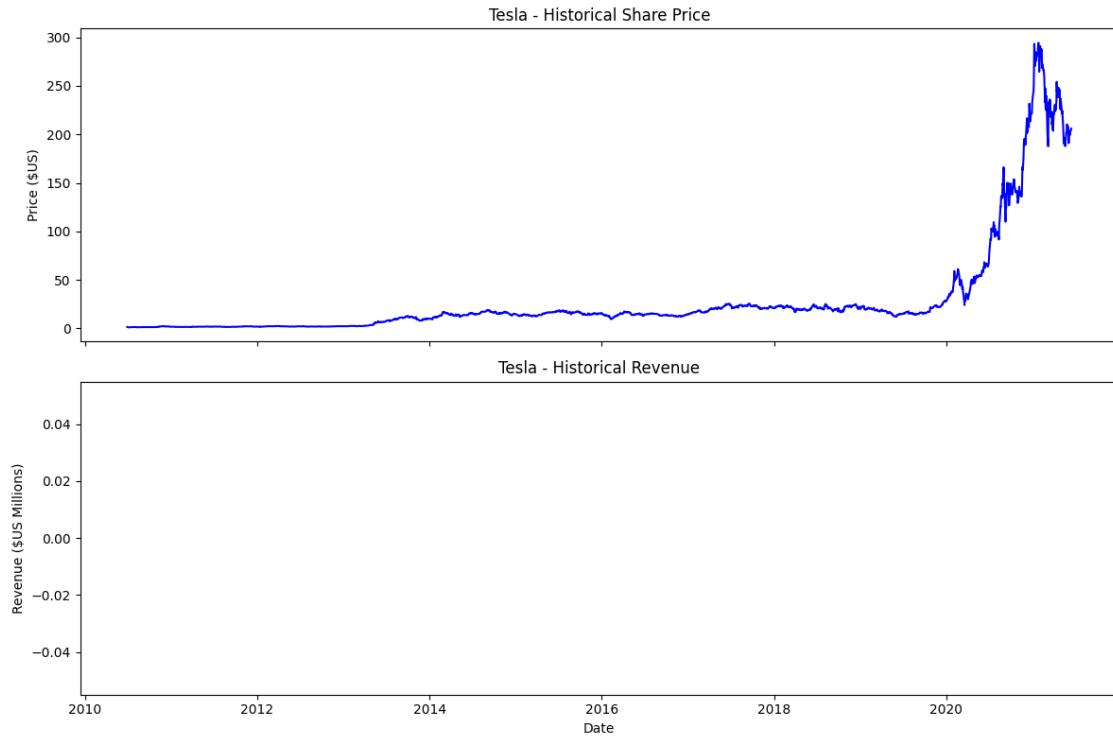
0.6 Question 5: Plot Tesla Stock Graph

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.

[37]: `make_graph(tesla_data, tesla_revenue, "Tesla")`



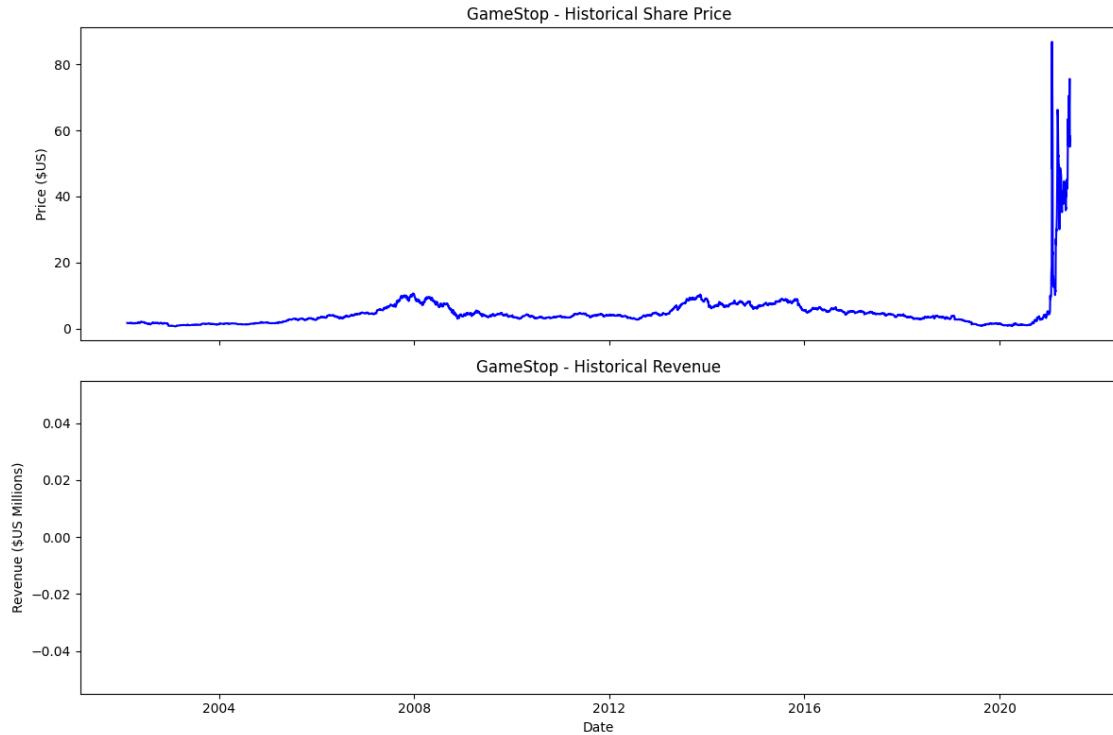
0.7 Question 6: Plot GameStop Stock Graph

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

Hint

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

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



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

Copyright © 2020 IBM Corporation. All rights reserved.