Final Assignment

August 29, 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.

[2]:
```
!pip install yfinance
!pip install bs4
!pip install nbformat
!pip install --upgrade plotly
```

```
Collecting yfinance
  Downloading yfinance-0.2.65-py2.py3-none-any.whl.metadata (5.8 kB)
Collecting pandas>=1.3.0 (from yfinance)
  Downloading
pandas-2.3.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(91 kB)
Collecting numpy>=1.16.5 (from yfinance)
  Downloading
numpy-2.3.2-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-3.18.2.tar.gz (949 kB)
                           949.2/949.2 kB
55.8 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-6.32.0-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-0.2.65-py2.py3-none-any.whl (119 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
82.2 MB/s eta 0:00:00
Downloading
numpy-2.3.2-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (16.6
MB)
                          16.6/16.6 MB
177.4 MB/s eta 0:00:00
Downloading
pandas-2.3.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.0
MB)
                          12.0/12.0 MB
170.4 MB/s eta 0:00:00
Downloading protobuf-6.32.0-cp39-abi3-manylinux2014_x86_64.whl (322 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=046ca8f2db0bfc7fd114f5c04da253f14c2425bd60abdf8e1ce033d474c2a480
  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-3.18.2-cp312-cp312-linux_x86_64.whl size=303862
sha256=d7499a922c15aec231b0dfa4789e9fb59c050333ad931878712f524fb894d195
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/d1/df/a9/0202b051c65b1
1c992dd6db9f2babdd2c44ec7d35d511be5d3
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-2.3.2
pandas-2.3.2 peewee-3.18.2 protobuf-6.32.0 tzdata-2025.2 websockets-15.0.1
yfinance-0.2.65
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)
Requirement already satisfied: plotly in /opt/conda/lib/python3.12/site-packages
(5.24.1)
Collecting plotly
  Downloading plotly-6.3.0-py3-none-any.whl.metadata (8.5 kB)
Collecting narwhals>=1.15.1 (from plotly)
  Downloading narwhals-2.2.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: packaging in /opt/conda/lib/python3.12/site-
packages (from plotly) (24.2)
Downloading plotly-6.3.0-py3-none-any.whl (9.8 MB)
                         9.8/9.8 MB
142.5 MB/s eta 0:00:00
Downloading narwhals-2.2.0-py3-none-any.whl (401 kB)
Installing collected packages: narwhals, plotly
  Attempting uninstall: plotly
    Found existing installation: plotly 5.24.1
    Uninstalling plotly-5.24.1:
      Successfully uninstalled plotly-5.24.1
Successfully installed narwhals-2.2.0 plotly-6.3.0
```

[3]:
```
import yfinance as yf
import pandas as pd
import requests
from bs4 import BeautifulSoup
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

[4]:
```
import plotly.io as pio
pio.renderers.default = "iframe"
```

In Python, you can ignore warnings using the warnings module. You can use the filterwarnings function to filter or ignore specific warning messages or categories.

[5]:
```
import warnings
# Ignore all warnings

warnings.filterwarnings("ignore", category=FutureWarning)
```

0.1 Define Graphing Function

In this section, we define the function `make_graph`. **You don't have to know how the function works, you should only care about the inputs. It takes a dataframe with stock data (dataframe must contain Date and Close columns), a dataframe with revenue data (dataframe must contain Date and Revenue columns), and the name of the stock.**

[6]:
```
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True,␣
 ↪subplot_titles=("Historical Share Price", "Historical Revenue"),␣
 ↪vertical_spacing = .3)
    stock_data_specific = stock_data[stock_data.Date <= '2021-06-14']
    revenue_data_specific = revenue_data[revenue_data.Date <= '2021-04-30']
    fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific.Date,␣
 ↪infer_datetime_format=True), y=stock_data_specific.Close.astype("float"),␣
 ↪name="Share Price"), row=1, col=1)
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date,␣
 ↪infer_datetime_format=True), y=revenue_data_specific.Revenue.
 ↪astype("float"), name="Revenue"), row=2, col=1)
    fig.update_xaxes(title_text="Date", row=1, col=1)
    fig.update_xaxes(title_text="Date", row=2, col=1)
    fig.update_yaxes(title_text="Price ($US)", row=1, col=1)
    fig.update_yaxes(title_text="Revenue ($US Millions)", row=2, col=1)
    fig.update_layout(showlegend=False,
    height=900,
    title=stock,
    xaxis_rangeslider_visible=True)
    fig.show()
    from IPython.display import display, HTML
    fig_html = fig.to_html()
    display(HTML(fig_html))
```

Use the make_graph function that we've already defined. You'll need to invoke it in questions 5 and 6 to display the graphs and create the dashboard. > **Note: You don't need to redefine the function for plotting graphs anywhere else in this notebook; just use the existing function.**

0.2 Question 1: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is Tesla and its ticker symbol is `TSLA`.

[7]:
```
tesla= yf.Ticker("TSLA")
```

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `tesla_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

[8]:
```
tesla_data= tesla.history(period="max")
```

Reset the index using the `reset_index(inplace=True)` function on the tesla_data DataFrame and display the first five rows of the `tesla_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 1 to the results below.

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

[9]:
```
                       Date      Open      High       Low     Close  \
0 2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667
1 2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667
2 2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000
3 2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000
4 2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000

      Volume  Dividends  Stock Splits
0  281494500        0.0           0.0
1  257806500        0.0           0.0
2  123282000        0.0           0.0
3   77097000        0.0           0.0
4  103003500        0.0           0.0
```

0.3 Question 2: Use Webscraping to Extract Tesla Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm Save the text of the response as a variable named `html_data`.

[10]:
```
URL="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"
response=requests.get(URL)
html_data=response.text
```

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

[11]:
```
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.
```

[12]:
```
tesla_revenue=pd.DataFrame(columns=['Date','Revenue'])
print(tesla_revenue)

tables=soup.find_all("table")
len(tables)

for index, table in enumerate(tables):
    if("Tesla Quarterly Revenue" in str(table)):
        table_index=index
        print(table_index)

for row in tables[table_index].tbody.find_all("tr"):
    col = row.find_all("td")
    if col:
        date = col[0].text.strip()
        revenue = col[1].text.strip()

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

        tesla_revenue=pd.concat([tesla_revenue, new_row], ignore_index=True)
tesla_revenue
```

```
Empty DataFrame
Columns: [Date, Revenue]
Index: []
1
```

[12]:

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

```
34  2014-03-31     $621
35  2013-12-31     $615
36  2013-09-30     $431
37  2013-06-30     $405
38  2013-03-31     $562
39  2012-12-31     $306
40  2012-09-30      $50
41  2012-06-30      $27
42  2012-03-31      $30
43  2011-12-31      $39
44  2011-09-30      $58
45  2011-06-30      $58
46  2011-03-31      $49
47  2010-12-31      $36
48  2010-09-30      $31
49  2010-06-30      $28
50  2010-03-31      $21
51  2009-12-31
52  2009-09-30      $46
53  2009-06-30      $27
```

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

[13]:
```
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.

[14]:
```
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.

[15]:
```
tesla_revenue.tail()
```

[15]:
```
          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`.

[16]:
```
gme=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.

[17]:
```
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.

[18]:
```
gme_data.reset_index(inplace=True)
gme_data.head()
```

[18]:
```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683251  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
```

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`.

[19]:
```
URL="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"
html_data_2=requests.get(URL).text
```

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

[20]:
```
soup=BeautifulSoup(html_data_2,"html.parser")
```

Using `BeautifulSoup` or the `read_html` function extract the table with `GameStop Revenue` and store it into a dataframe named `gme_revenue`. The dataframe should have columns `Date` and `Revenue`. Make sure the comma and dollar sign is removed from the `Revenue` column.

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

Click here if you need help locating the table

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

soup.find_all("tbody")[1]

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

[21]:
```
gme_revenue=pd.DataFrame(columns=['Date','Revenue'])
print(gme_revenue)

tables=soup.find_all("table")
len(tables)

for index, table in enumerate(tables):
    if("Gamestop Revenue" in str(table)):
        table_index=index
        print(table_index)

for row in tables[table_index].tbody.find_all("tr"):
    col = row.find_all("td")
    if col:
        date = col[0].text.strip()
        revenue = col[1].text.strip()

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

        gme_revenue=pd.concat([gme_revenue, new_row], ignore_index=True)

gme_revenue["Revenue"] = gme_revenue['Revenue'].str.
 ↪replace(',|\$',"",regex=True)
gme_revenue

#gme_revenue.dropna(inplace=True)

#gme_revenue = gme_revenue[gme_revenue['Revenue'] != ""]
```

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

[21]:
```
           Date Revenue
0    2020-04-30    1021
1    2020-01-31    2194
2    2019-10-31    1439
3    2019-07-31    1286
4    2019-04-30    1548
..          …       …
```

```
57  2006-01-31    1667
58  2005-10-31     534
59  2005-07-31     416
60  2005-04-30     475
61  2005-01-31     709

[62 rows x 2 columns]
```

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

[22]:
```
gme_revenue.tail()
```

[22]:
```
          Date Revenue
57  2006-01-31    1667
58  2005-10-31     534
59  2005-07-31     416
60  2005-04-30     475
61  2005-01-31     709
```

0.6 Question 5: Plot Tesla Stock Graph

Use the `make_graph` function to graph the Tesla Stock Data, also provide a title for the graph. Note the graph will only show data upto June 2021.

Hint

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

[23]:
```
make_graph(tesla_data, tesla_revenue, "Tesla")
```

```
/tmp/ipykernel_301/109047474.py:5: UserWarning:

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

/tmp/ipykernel_301/109047474.py:6: UserWarning:

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


<IPython.core.display.HTML object>
```

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
```

[24]:
```
make_graph(gme_data,gme_revenue,"Gamestop")
```

```
/tmp/ipykernel_301/109047474.py:5: UserWarning:

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

/tmp/ipykernel_301/109047474.py:6: UserWarning:

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


<IPython.core.display.HTML object>
```

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

##