



Name	Last Modified
Untitled Fo...	19 hours ago
(t) amd.json	6 months ago
amd.json.1	6 months ago
amd.json.2	6 months ago
amd.json.3	6 months ago
amd.json.4	6 months ago
(t) apple.json	6 months ago
Final Assig...	an hour ago
Final Assig...	2 minutes ago
Final_Assi...	an hour ago
WebScrap...	18 hours ago

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.

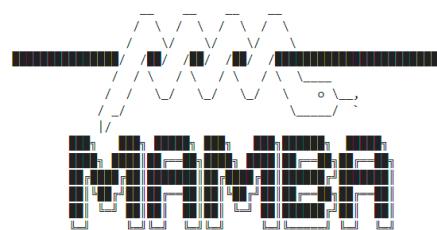
Table of Contents

- Define a Function that Makes a Graph
- Question 1: Use yfinance to Extract Stock Data
- Question 2: Use Webscraping to Extract Tesla Revenue Data
- Question 3: Use yfinance to Extract Stock Data
- Question 4: Use Webscraping to Extract GME Revenue Data
- Question 5: Plot Tesla Stock Graph
- Question 6: Plot GameStop Stock Graph

Estimated Time Needed: 30 min

```
[1]: !pip install yfinance==0.1.67
!mamba install bs4==4.10.0 -y
!pip install nbformat==4.2.0

Collecting yfinance==0.1.67
  Downloading yfinance-0.1.67-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: pandas>=0.24 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (1.3.5)
Requirement already satisfied: requests>=2.20 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (2.28.1)
Requirement already satisfied: lxml<4.5.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (4.6.4)
Collecting multitasking<0.7
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Requirement already satisfied: numpy>=1.15 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (1.21.6)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from pandas>=0.24->yfinance==0.1.67) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from pandas>=0.24->yfinance==0.1.67) (202.6)
Requirement already satisfied: charset-normalizer<3,>=2 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from requests>=2.20->yfinance==0.1.67) (2.1.1)
Requirement already satisfied: certifi>=2017.4.17 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from requests>=2.20->yfinance==0.1.67) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from requests>=2.20->yfinance==0.1.67) (1.26.13)
Requirement already satisfied: idna<4,>=2.5 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from requests>=2.20->yfinance==0.1.67) (3.4)
Requirement already satisfied: six>=1.5 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from python-dateutil>=2.7.3->pandas>=0.24->yfinance==0.1.67) (1.16.0)
Installing collected packages: multitasking, yfinance
Successfully installed multitasking-0.0.11 yfinance-0.1.67
```



mamba (0.15.3) supported by @QuantStack

GitHub: <https://github.com/mamba-org/mamba>
Twitter: <https://twitter.com/QuantStack>

Looking for: ['bs4==4.10.0']

```
pkgs/r/linux-64      [>      ] (----) No change
pkgs/r/linux-64      [=====] (0m:00s) No change
pkgs/r/noarch        [>      ] (----) No change
pkgs/r/noarch        [=====] (0m:00s) No change
pkgs/main/noarch     [>      ] (----) No change
pkgs/main/noarch     [=====] (0m:00s) No change
pkgs/main/linux-64   [<=>    ] (0m:00s)
pkgs/main/linux-64   [>      ] (0m:00s) 720 KB / ?? (2.32 MB/s)
pkgs/main/linux-64   [<=>    ] (0m:00s) 720 KB / ?? (2.32 MB/s)
pkgs/main/linux-64   [<=>    ] (0m:00s) 1 MB / ?? (3.23 MB/s)
pkgs/main/linux-64   [<=>    ] (0m:00s) 1 MB / ?? (3.23 MB/s)
pkgs/main/linux-64   [<=>    ] (0m:00s) 2 MB / ?? (3.78 MB/s)
pkgs/main/linux-64   [<=>    ] (0m:00s) 2 MB / ?? (3.78 MB/s)
pkgs/main/linux-64   [<=>    ] (0m:00s) 3 MB / ?? (4.01 MB/s)
pkgs/main/linux-64   [<=>    ] (0m:00s) 4 MB / ?? (4.24 MB/s)
pkgs/main/linux-64   [<=>    ] (0m:00s) 4 MB / ?? (4.24 MB/s)
pkgs/main/linux-64   [<=>    ] (0m:00s) 5 MB / ?? (4.42 MB/s)
pkgs/main/linux-64   [<=>    ] (0m:00s) Finalizing...
pkgs/main/linux-64   [<=>    ] (0m:01s) Done
```

```

[4]:          _____ (version)  date
Pinned packages:
- python 3.7.*


Transaction

Prefix: /home/jupyterlab/conda/envs/python

All requested packages already installed

Collecting nbformat==4.2.0
  Downloading nbformat-4.2.0-py2.py3-none-any.whl (153 kB)
    153.3/153.3 kB 18.5 MB/s eta 0:00:00
Requirement already satisfied: jupyter-core in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from nbformat==4.2.0) (4.12.0)
Requirement already satisfied: traitlets>=4.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from nbformat==4.2.0) (5.6.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from nbformat==4.2.0) (4.17.3)
Requirement already satisfied: ipython-genutils in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from nbformat==4.2.0) (0.2.0)
Requirement already satisfied: importlib-resources>=1.4.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (1.10.1)
Requirement already satisfied: attrs>=17.4.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (22.1.0)
Requirement already satisfied: pkgutil-resolve-name>=1.3.10 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (1.3.10)
Requirement already satisfied: typing-extensions in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (4.4.0)
Requirement already satisfied: importlib-metadata in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (4.11.4)
Requirement already satisfied: persistent!=0.17.0,!0.17.1,!0.17.2,>=0.14.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (0.19.2)
Requirement already satisfied: zipp=>3.1.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from importlib-resources>=1.4.0->jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (3.11.0)
Requirement already satisfied: nbformat in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages
  Installing collected packages: nbformat
    Attempting uninstall: nbformat
      Found existing installation: nbformat 5.7.0
      Uninstalling nbformat-5.7.0:
        Successfully uninstalled nbformat-5.7.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
nbconvert 7.2.6 requires nbformat>=5.1, but you have nbformat 4.2.0 which is incompatible.
nbclient 0.7.2 requires nbformat>=5.1, but you have nbformat 4.2.0 which is incompatible.
jupyter-server 1.23.3 requires nbformat>=5.2.0, but you have nbformat 4.2.0 which is incompatible.
Successfully installed nbformat-4.2.0

```

```
[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
```

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.

```
[7]: 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="Stock Price"))
    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"))
    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()
```

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`.

```
[9]: 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.

```
[20]: tesla_data = Tesla.history(period="max")
print(tesla_data)

          Open   High    Low   Close   Volume \
Date
2010-06-29  1.266667  1.666667  1.169333  1.592667  281494500
2010-06-30  1.719333  2.028000  1.553333  1.588667  257806500
2010-07-01  1.666667  1.728000  1.351333  1.464000  123282000
2010-07-02  1.533333  1.540000  1.247333  1.280000  77097000
2010-07-06  1.333333  1.333333  1.055333  1.074000  103003500
...
...
...
2023-02-24  196.330002  197.669998  192.800003  196.880005  142228100
2023-02-27  202.029999  209.419998  201.259995  207.630005  161028300
2023-02-28  210.589996  211.229996  203.750000  205.710007  151344900
2023-03-01  206.210007  207.199997  198.520004  202.770004  151617000
2023-03-02  186.740005  193.723694  186.009995  190.110001  140932535

          Dividends  Stock Splits
Date
2010-06-29      0        0.0
2010-06-30      0        0.0
2010-07-01      0        0.0
2010-07-02      0        0.0
2010-07-06      0        0.0
...
...
2023-02-24      0        0.0
2023-02-27      0        0.0
2023-03-01      a        a a
```

```
 2023-03-01      0      0.0
2023-03-02      0      0.0
```

[3191 rows x 7 columns]

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.

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

```
          Date   Open   High    Low   Close  Volume  Dividends  \
0 2010-06-29  1.266667  1.666667  1.169333  1.592667  281494500      0
1 2010-06-30  1.719333  2.028000  1.553333  1.588667  257806500      0
2 2010-07-01  1.666667  1.728000  1.351333  1.464000  123282000      0
3 2010-07-02  1.533333  1.540000  1.247333  1.280000  77097000      0
4 2010-07-06  1.333333  1.333333  1.055333  1.074000  103003500      0

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

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`.

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

Parse the html data using `beautiful_soup`.

```
[26]: soup = BeautifulSoup(html_data, 'html5lib')
```

Using `BeautifulSoup` or the `read_html` function extract the table with `Tesla Quarterly Revenue` and store it into a dataframe named `tesla_revenue`. The dataframe should have columns `Date` and `Revenue`.

▼ 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 like in the previous lab

```
soup.find_all("tbody")[1]
```

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

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

# First we isolate the body of the table which contains all the information
# Then we loop through each row and find all the column values for each row
for row in soup.find("tbody").find_all('tr'):
    col = row.find_all("td")
    date = col[0].text
    revenue = col[1].text

    # Finally we append the data of each row to the table
    tesla_revenue = tesla_revenue.append({"Date":date, "Revenue":revenue}, ignore_index=True)
```

```
[29]: tesla_revenue.head()
```

	Date	Revenue
0	2021	\$53,823
1	2020	\$31,536
2	2019	\$24,578
3	2018	\$21,461
4	2017	\$11,759

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

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

/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning: The default value of regex will change from True to False in a future version.
***Entry point for launching an IPython kernel.

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

```
[32]: 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.

```
[34]: print(tesla_revenue.tail())
```

	Date	Revenue
8	2013	2013
9	2012	413
10	2011	204
11	2010	117
12	2009	112

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`.

```
[35]: 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.

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

          Open    High     Low   Close   Volume Dividends \
Date
2002-02-13  1.620128  1.693350  1.603296  1.691667  76216000      0.0
2002-02-14  1.712708  1.716074  1.670626  1.683251  11021600      0.0
2002-02-15  1.683250  1.687458  1.658002  1.674834  8389600      0.0
2002-02-19  1.666418  1.666418  1.578047  1.607504  74104000      0.0
2002-02-20  1.615920  1.662210  1.603296  1.662210  6892800      0.0
...
2023-02-24  19.709999  20.080000  19.540001  19.540001  2277600      0.0
2023-02-27  19.610001  19.700001  18.889999  19.180000  4006700      0.0
2023-02-28  19.059999  19.520000  18.950001  19.230000  3373800      0.0
2023-03-01  18.920000  19.090000  18.139999  18.150000  4210100      0.0
2023-03-02  18.150000  18.400000  17.620001  18.170000  1922604      0.0

      Stock Splits
Date
2002-02-13      0.0
2002-02-14      0.0
2002-02-15      0.0
2002-02-19      0.0
2002-02-20      0.0
...
2023-02-24      0.0
2023-02-27      0.0
2023-02-28      0.0
2023-03-01      0.0
2023-03-02      0.0

[5299 rows x 7 columns]
```

Reset the index using the `reset_index(inplace=True)` function on the `gme_data` DataFrame and display the first five rows of the `gme_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 3 to the results below.

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

          Date    Open    High     Low   Close   Volume Dividends \
0 2002-02-13  1.620128  1.693350  1.603296  1.691667  76216000      0.0
1 2002-02-14  1.712708  1.716074  1.670626  1.683251  11021600      0.0
2 2002-02-15  1.683250  1.687458  1.658002  1.674834  8389600      0.0
3 2002-02-19  1.666418  1.666418  1.578047  1.607504  74104000      0.0
4 2002-02-20  1.615920  1.662210  1.603296  1.662210  6892800      0.0

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

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`.

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

Parse the html data using `beautiful_soup`.

```
[39]: soup = BeautifulSoup(html_data, 'html5lib')
```

Using `BeautifulSoup` or the `read_html` function extract the table with `GameStop Quarterly 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 using a method similar to what you did in Question 2.

► Click here if you need help locating the table

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

# First we isolate the body of the table which contains all the information
# Then we loop through each row and find all the column values for each row
for row in soup.find("tbody").find_all('tr'):
    col = row.find_all("td")
    date = col[0].text
    revenue = col[1].text

    # Finally we append the data of each row to the table
    gme_revenue = gme_revenue.append({"Date":date, "Revenue":revenue}, ignore_index=True)
```

```
[45]: gme_revenue.head()
```

	Date	Revenue
0	2020	\$6,466
1	2019	\$8,285
2	2018	\$8,547
3	2017	\$7,965
4	2016	\$9,364

```
[46]: gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',', '|\\$|', '#')
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning: The default value of regex will change from True to False in a future version.
    """Entry point for launching an IPython kernel.
```

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

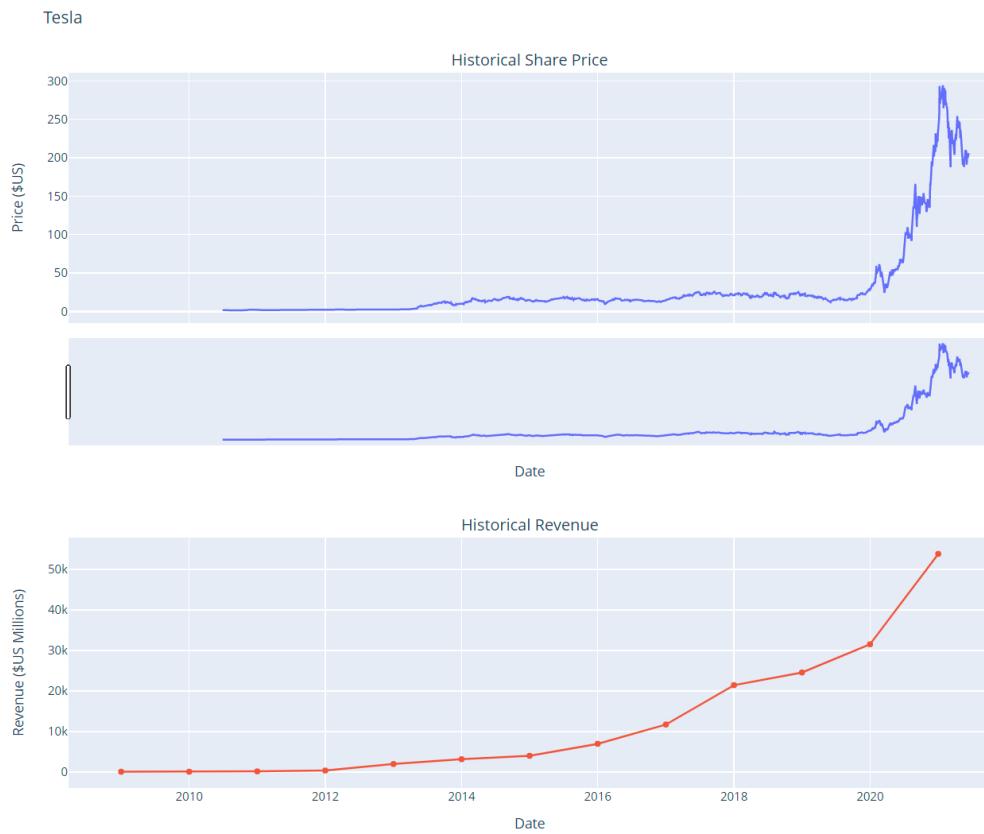
```
[47]: print(gme_revenue.tail())
```

	Date	Revenue
11	2009	8806
12	2008	7094
13	2007	5319
14	2006	3092
15	2005	1843

Question 5: Plot Tesla Stock Graph

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

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

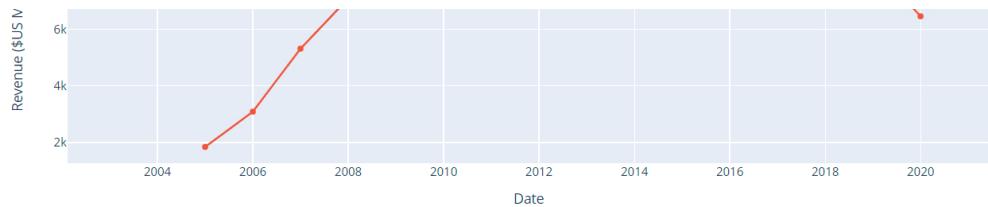


Question 6: Plot GameStop Stock Graph

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.

```
[50]: 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

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

© IBM Corporation 2020. All rights reserved.