Skills Network

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

Estimated Time Needed: **30 min**

Note:- If you are working Locally using anaconda, please uncomment the following code and execute it.

In [1]:
```
#!pip install yfinance==0.2.38
#!pip install pandas==2.2.2
#!pip install nbformat
```

In [2]:
```
!pip install yfinance==0.1.67
!mamba install bs4==4.10.0 -y
!pip install nbformat==4.2.0
```

```
Requirement already satisfied: yfinance==0.1.67 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (0.1.67)
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: 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: requests>=2.20 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (2.29.0)
Requirement already satisfied: multitasking>=0.0.7 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (0.0.11)
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)
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) (2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from requests>=2.20->yfinance==0.1.67) (3.1.0)
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: 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) (1.26.15)
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) (2023.5.7)
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)

                  __    __    __    __
                 /  \  /  \  /  \  /  \
                /    \/    \/    \/    \
███████████████/  /██/  /██/  /██/  /████████████████████████
              /  / \   / \   / \   / \  \____
             /  /   \_/   \_/   \_/   \    o \__,
            / _/                       \_____/  `
            |/
        ███╗   ███╗ █████╗ ███╗   ███╗██████╗  █████╗
        ████╗ ████║██╔══██╗████╗ ████║██╔══██╗██╔══██╗
        ██╔████╔██║███████║██╔████╔██║██████╔╝███████║
        ██║╚██╔╝██║██╔══██║██║╚██╔╝██║██╔══██╗██╔══██║
        ██║ ╚═╝ ██║██║  ██║██║ ╚═╝ ██║██████╔╝██║  ██║
        ╚═╝     ╚═╝╚═╝  ╚═╝╚═╝     ╚═╝╚═════╝ ╚═╝  ╚═╝

        mamba (1.4.2) supported by @QuantStack

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

█████████████████████████████████████████████████████████████


Looking for: ['bs4==4.10.0']

[+] 0.0s
[+] 0.1s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   0.0 B /  ??.?MB @  ??.?MB/s  0.1s
pkgs/main/noarch   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   0.0 B /  ??.?MB @  ??.?MB/s  0.1s
pkgs/r/linux-64    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   0.0 B /  ??.?MB @  ??.?MB/s  0.1s
pkgs/r/noarch      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   0.0 B /  ??.?MB @  ??.?MB/s  0.1spkgs/r/linux-64                                  No change
pkgs/r/noarch                                                No change
[+] 0.2s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   0.0 B /  ??.?MB @  ??.?MB/s  0.2s
pkgs/main/noarch   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   0.0 B /  ??.?MB @  ??.?MB/s  0.2s[+] 0.3s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 184.3kB /  ??.?MB @ 716.4kB/s  0.3s
pkgs/main/noarch   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 405.5kB /  ??.?MB @   1.5MB/s  0.3s[+] 0.4s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 491.5kB @   1.4MB/s             0.4s
pkgs/main/noarch   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 873.1kB @   2.2MB/s Finalizing  0.4spkgs/main/noarch                        @   2.2MB/s  0.4s
[+] 0.5s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 790.5kB /  ??.?MB @   1.7MB/s  0.5s[+] 0.6s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   1.2MB /  ??.?MB @   2.1MB/s  0.6s[+] 0.7s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   1.6MB /  ??.?MB @   2.4MB/s  0.7s[+] 0.8s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   1.9MB /  ??.?MB @   2.5MB/s  0.8s[+] 0.9s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   2.3MB /  ??.?MB @   2.6MB/s  0.9s[+] 1.0s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   2.7MB /  ??.?MB @   2.7MB/s  1.0s[+] 1.1s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   2.8MB /  ??.?MB @   2.7MB/s  1.1s[+] 1.2s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   3.3MB /  ??.?MB @   2.8MB/s  1.2s[+] 1.3s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   3.7MB /  ??.?MB @   2.9MB/s  1.3s[+] 1.4s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   4.0MB /  ??.?MB @   2.9MB/s  1.4s[+] 1.5s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   4.4MB /  ??.?MB @   3.0MB/s  1.5s[+] 1.6s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   4.7MB /  ??.?MB @   3.0MB/s  1.6s[+] 1.7s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   5.1MB /  ??.?MB @   3.1MB/s  1.7s[+] 1.8s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   5.6MB /  ??.?MB @   3.1MB/s  1.8s[+] 1.9s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   6.1MB /  ??.?MB @   3.2MB/s  1.9s[+] 2.0s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   6.4MB /  ??.?MB @   3.2MB/s  2.0s[+] 2.1s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   6.9MB /  ??.?MB @   3.3MB/s  2.1s[+] 2.2s
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━   7.1MB @   3.3MB/s Finalizing  2.2spkgs/main/linux-64                        @   3.3MB/s  2.3s

Pinned packages:
  - python 3.7.*


Transaction

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

  All requested packages already installed

Requirement already satisfied: nbformat==4.2.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (4.2.0)
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: 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: 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) (5.9.0)
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) (23.1.0)
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: 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) (5.12.0)
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: pyrsistent!=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) (0.19.3)
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) (4.5.0)
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) (3.15.0)
```

In [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
import pandas as pd
```

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.

In [4]:
```
import warnings
# Ignore all warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

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.

In [5]:
```
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()
```

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`.

In [6]:
```
tesla = yf.Ticker("TSLA")
```

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

In [7]:
```
tesla_data = tesla.history(period="max")
```

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

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

Out[8]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2010-06-29	1.266667	1.666667	1.169333	1.592667	281494500	0	0.0
1	2010-06-30	1.719333	2.028000	1.553333	1.588667	257806500	0	0.0
2	2010-07-01	1.666667	1.728000	1.351333	1.464000	123282000	0	0.0
3	2010-07-02	1.533333	1.540000	1.247333	1.280000	77097000	0	0.0
4	2010-07-06	1.333333	1.333333	1.055333	1.074000	103003500	0	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`.

In [9]:
```
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`.

In [10]:
```
!pip install html5lib
beautiful_soup = BeautifulSoup(html_data, 'lxml')
```

```
Requirement already satisfied: html5lib in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (1.1)
Requirement already satisfied: six>=1.9 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from html5lib) (1.16.0)
Requirement already satisfied: webencodings in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from html5lib) (0.5.1)
```

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`.

▶ Click here if you need help locating the table

In [11]:
```
#Initialize an empty DataFrame
tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])

# Extract table data into a list
for row in beautiful_soup.find_all("tbody")[1].find_all('tr'):
    cols = row.find_all('td')
    date = cols[0].text.strip()  # Corrected variable name and added .strip() to remove extra spaces
    revenue = cols[1].text.strip()  # Corrected variable name and added .strip() to remove extra spaces
    # Append data to the DataFrame
    tesla_revenue = tesla_revenue.append({"Date": date, "Revenue": revenue}, ignore_index=True)

# Display the first few rows of the dataframe
print(tesla_revenue.head())
```

```
         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
```

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

In [12]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

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

In [13]:
```
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.

In [14]:
```
tesla_revenue.tail()
```

Out[14]:

	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

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`.

In [15]:
```
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.

In [16]:
```
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.

In [17]:
```
gme_data.reset_index(inplace=True)
gme_data.head()
```

Out[17]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-02-13	1.620129	1.693350	1.603296	1.691667	76216000	0.0	0.0
1	2002-02-14	1.712707	1.716074	1.670626	1.683250	11021600	0.0	0.0
2	2002-02-15	1.683251	1.687459	1.658002	1.674834	8389600	0.0	0.0
3	2002-02-19	1.666418	1.666418	1.578047	1.607504	7410400	0.0	0.0
4	2002-02-20	1.615920	1.662210	1.603296	1.662210	6892800	0.0	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`.

In [18]:
```
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
#print(html_data)
```

Parse the html data using `beautiful_soup`.

In [19]:
```
gme_soup = BeautifulSoup(html_data, 'lxml')
```

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 using a method similar to what you did in Question 2.

▶ Click here if you need help locating the table

In [20]:
```
#Initialize an empty DataFrame
gme_revenue = pd.DataFrame(columns=["Date", "Revenue"])

# Extract table data into a list
for row in gme_soup.find_all("tbody")[1].find_all('tr'):
    cols = row.find_all('td')
    date = cols[0].text.strip()  # Corrected variable name and added .strip() to remove extra spaces
    revenue = cols[1].text.strip()  # Corrected variable name and added .strip() to remove extra spaces
    # Append data to the DataFrame
    gme_revenue = gme_revenue.append({"Date": date, "Revenue": revenue}, ignore_index=True)

# Display the first few rows of the dataframe
print(gme_revenue.head())

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

# Drop rows with missing or empty 'Revenue' values
gme_revenue.dropna(inplace=True)
gme_revenue = gme_revenue[gme_revenue['Revenue'] != ""]
```

```
         Date Revenue
0  2020-04-30  $1,021
1  2020-01-31  $2,194
2  2019-10-31  $1,439
3  2019-07-31  $1,286
4  2019-04-30  $1,548
```

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

In [21]:
```
gme_revenue.tail()
```

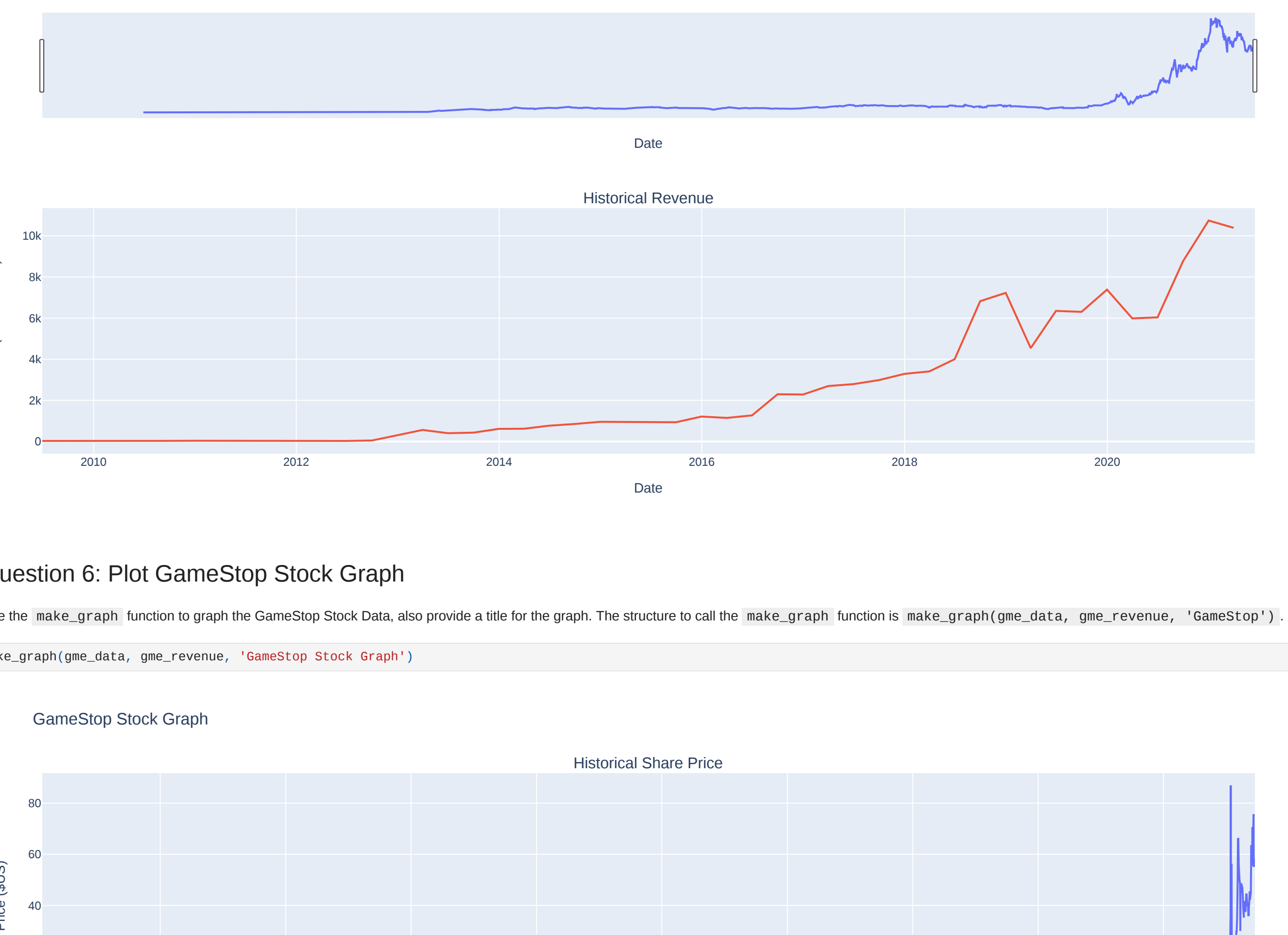
Out[21]:

	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

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.

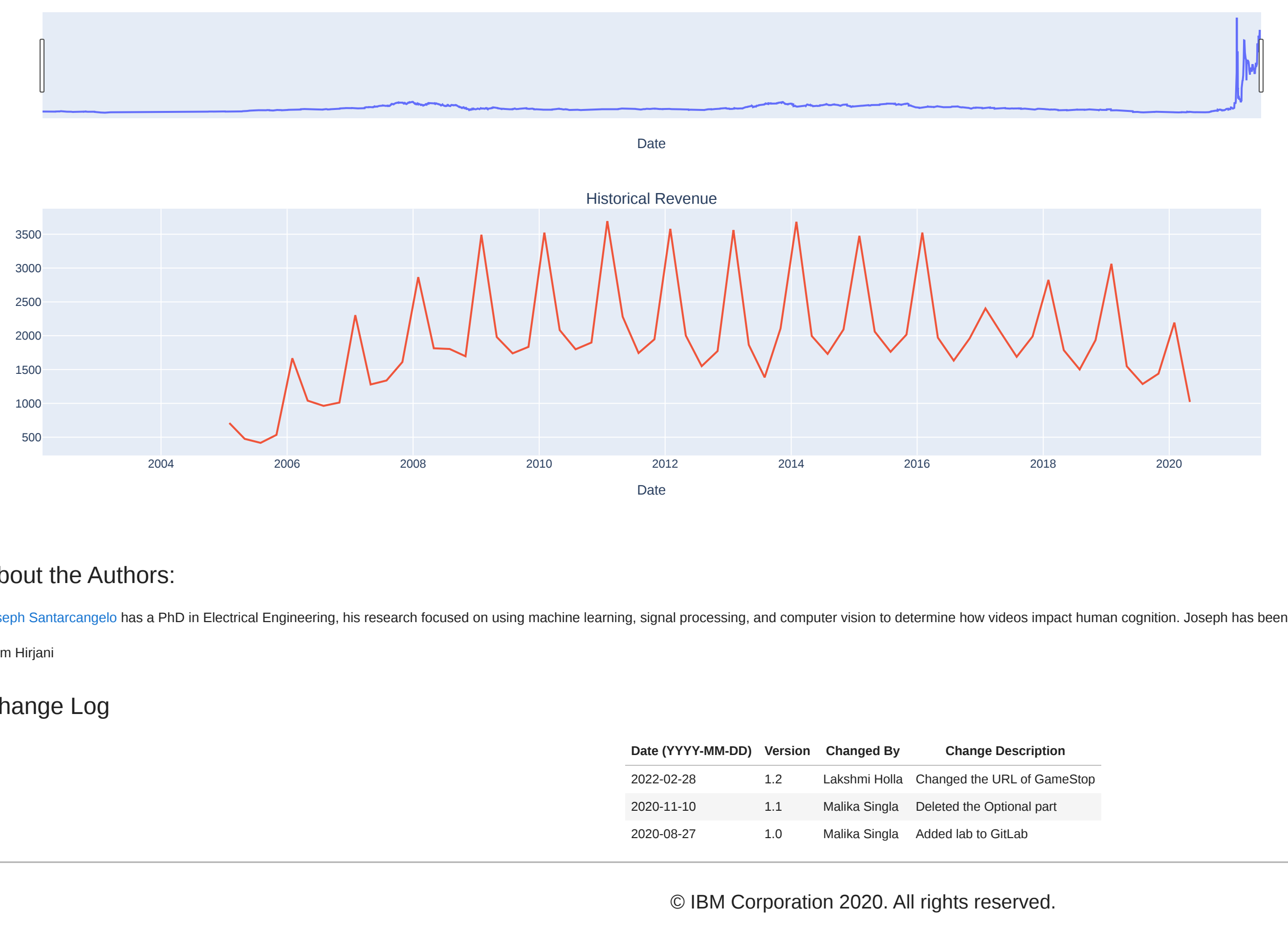
In [22]:
```
make_graph(tesla_data, tesla_revenue, 'Tesla Stock Graph')
```



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.

In [23]:
```
make_graph(gme_data, gme_revenue, 'GameStop Stock Graph')
```



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
