



# 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**

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

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

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

```
In [1]: 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
```

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 [2]: 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 [3]: def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Hist
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), y=stock
fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date), y=rev
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,
axis_rangeslider_visible=True)
fig.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.

## 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 [58]: 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 [59]: 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 [60]: tesla_data.reset_index(inplace=True)
tesla_data.head(5)
```

```
Out[60]:
```

|   | Date                      | Open     | High     | Low      | Close    | Volume    | Dividends | Stock Splits |
|---|---------------------------|----------|----------|----------|----------|-----------|-----------|--------------|
| 0 | 2010-06-29 00:00:00-04:00 | 1.266667 | 1.666667 | 1.169333 | 1.592667 | 281494500 | 0.0       | 0.0          |
| 1 | 2010-06-30 00:00:00-04:00 | 1.719333 | 2.028000 | 1.553333 | 1.588667 | 257806500 | 0.0       | 0.0          |
| 2 | 2010-07-01 00:00:00-04:00 | 1.666667 | 1.728000 | 1.351333 | 1.464000 | 123282000 | 0.0       | 0.0          |
| 3 | 2010-07-02 00:00:00-04:00 | 1.533333 | 1.540000 | 1.247333 | 1.280000 | 77097000  | 0.0       | 0.0          |
| 4 | 2010-07-06 00:00:00-04:00 | 1.333333 | 1.333333 | 1.055333 | 1.074000 | 103003500 | 0.0       | 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 [96]: url="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDevel
html_data=requests.get(url).text
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`. Make sure to use the `html_data` with the content parameter as follow `html_data.content`.

```
In [104... 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

► [Click here if you need help locating the table](#)

```
In [92]: # As per the hint, the table is located at tbody index 1
revenue_table = soup.find_all("tbody")[1] #tbody has date and revenue

# Step 4: Initialize a list to hold the revenue data
revenue_data = []

# Step 5: Loop through each row in the relevant table
for row in revenue_table.find_all('tr'):
    columns = row.find_all('td')
    if len(columns) >= 2:
        # Extract Date and Revenue (column 1 and column 2)
        date = columns[0].text.strip()
        revenue = columns[1].text.strip()

        # Step 6: Clean Revenue data (remove dollar signs and commas)
        revenue = revenue.replace('$', '').replace(',', '')

        # Step 7: Append the cleaned data to the list
        revenue_data.append({'Date': date, 'Revenue': revenue})

# Step 8: Convert the list of dictionaries to a DataFrame
tesla_revenue = pd.DataFrame(revenue_data)
```

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

```
In [94]: tesla_revenue.head()
```

```
Out[94]:
```

|   | 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    |

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

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

```
In [95]: tesla_revenue.tail(5)
```

Out[95]:

|    | 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 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 [84]: 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 [85]: 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 [86]: gme_data.reset_index(inplace=True)
gme_data.head(5)
```

Out[86]:

|   | Date                         | Open     | High     | Low      | Close    | Volume   | Dividends | Stock Splits |
|---|------------------------------|----------|----------|----------|----------|----------|-----------|--------------|
| 0 | 2002-02-13<br>00:00:00-05:00 | 1.620129 | 1.693350 | 1.603296 | 1.691667 | 76216000 | 0.0       | 0.0          |
| 1 | 2002-02-14<br>00:00:00-05:00 | 1.712707 | 1.716073 | 1.670626 | 1.683250 | 11021600 | 0.0       | 0.0          |
| 2 | 2002-02-15<br>00:00:00-05:00 | 1.683250 | 1.687458 | 1.658001 | 1.674834 | 8389600  | 0.0       | 0.0          |
| 3 | 2002-02-19<br>00:00:00-05:00 | 1.666417 | 1.666417 | 1.578047 | 1.607504 | 7410400  | 0.0       | 0.0          |
| 4 | 2002-02-20<br>00:00:00-05:00 | 1.615921 | 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_2`.

```
In [74]: url="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDevel
html_data_2=requests.get(url).text
```

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

```
In [75]: soup=BeautifulSoup(html_data_2)
```

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

```
In [76]: # Step 3: Find the specific table that contains GameStop Revenue
# As per the hint, the table is located at tbody index 1
revenue_table = soup.find_all("tbody")[1]
```

```

# Step 4: Initialize a list to hold the revenue data
revenue_data = []

# Step 5: Loop through each row in the relevant table
for row in revenue_table.find_all('tr'):
    columns = row.find_all('td')
    if len(columns) >= 2:
        # Extract Date and Revenue (column 1 and column 2)
        date = columns[0].text.strip()
        revenue = columns[1].text.strip()

        # Step 6: Clean Revenue data (remove dollar signs and commas)
        revenue = revenue.replace('$', '').replace(',', '')

        # Step 7: Append the cleaned data to the list
        revenue_data.append({'Date': date, 'Revenue': revenue})

# Step 8: Convert the list of dictionaries to a DataFrame
gme_revenue = pd.DataFrame(revenue_data)

```

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

```
In [77]: gme_revenue.tail(5)
```

```
Out[77]:
```

|    | 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     |

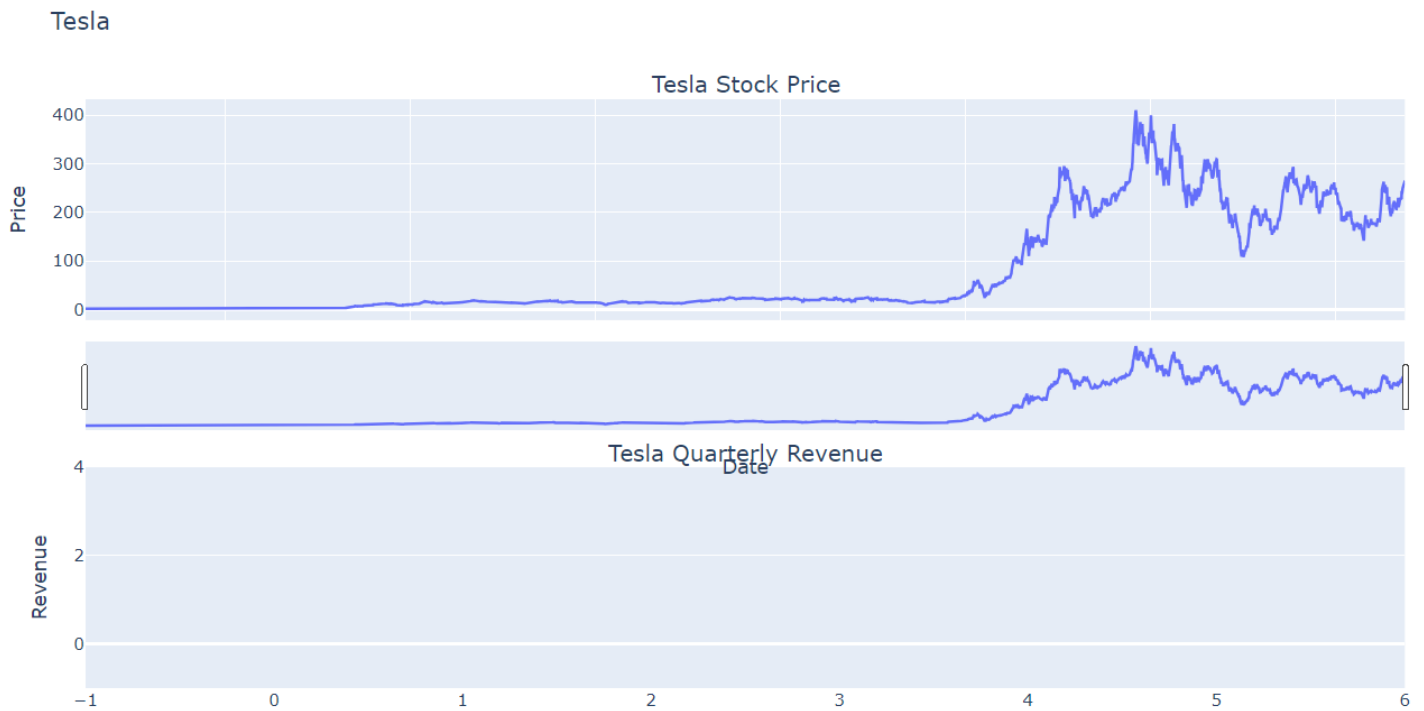
## Question 5: Plot Tesla Stock Graph

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

► Hint

```
In [81]: make_graph(tesla_data, tesla_revenue, 'Tesla')
```

```
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.

► Hint

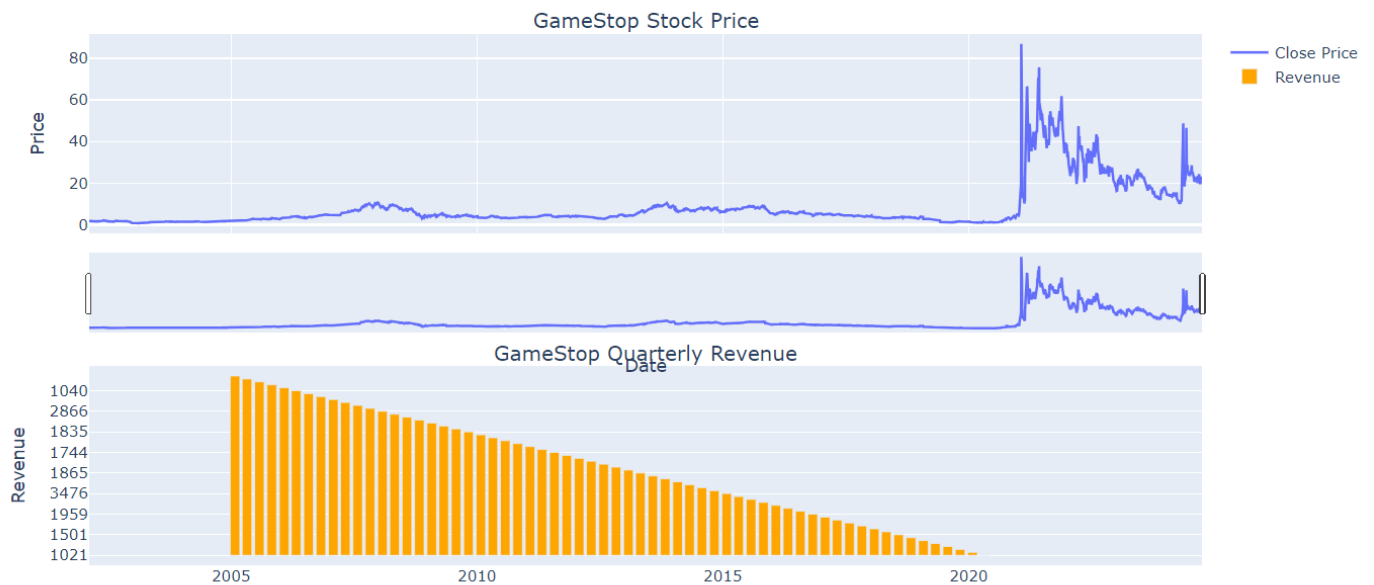
```
In [87]: make_graph(gme_data, gme_revenue, 'GameStop')
```



```
make_graph(gme_data, gme_revenue, 'GameStop')
```



## 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.

© IBM Corporation 2020. All rights reserved.

