

# Peer-graded Assignment: Analyzing Historical Stock/Revenue Data and Building a Dashboard

*Name: Ayushi Tyagi*

<b>Summary Table</b>	Beginning of code to answer to the different questions: .....	Error! Bookmark not defined.
<b>Question 1: Use yfinance to Extract Stock Data (Extracting Tesla Stock Data Using yfinance).....</b>		<b>2</b>
<b>Question 2: Use Webscraping to Extract Tesla Revenue Data (Extracting Tesla Revenue Data Using Webscraping).....</b>		<b>3</b>
<b>Question 3: Use yfinance to Extract Stock Data (Extracting GameStop Stock Data Using yfinance).....</b>		<b>6</b>
<b>Question 4: Use Webscraping to Extract GME Revenue Data (Extracting GameStop Revenue Data Using Webscraping).....</b>		<b>7</b>
<b>Question 5: Plot Tesla Stock Graph (Tesla Stock and Revenue Dashboard) .....</b>		<b>9</b>
<b>Question 6: Plot GameStop Stock Graph (GameStop Stock and Revenue Dashboard) .....</b>		<b>10</b>

### Beginning of code to answer to the different questions:

```
!pip install yfinance==0.1.67
!mamba install bs4==4.10.0 -y
!pip install nbformat==4.2.0
```

```
#!pip install pandas==1.3.3
#!pip install requests==2.26.0
!mamba install bs4==4.10.0 -y
!mamba install html5lib==1.1 -y
!pip install lxml==4.6.4
#!pip install plotly==5.3.1
```

```
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
```

```
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=0.05)
    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"), mode="lines", 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"), mode="lines", 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 (Extracting Tesla Stock Data Using yfinance)

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`.

```
#Question 1
tsla = 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.

```
tesla_data = tsla.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.

```
tesla_data.reset_index(inplace=True)
```

```
tesla_data.head()
```

	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 (Extracting Tesla Revenue Data Using Webscraping)

Use the `requests` library to download the webpage <https://cf-courses-data.s3.us.cloudobject-storage.appdomain.cloud/IBMDDeveloperSkillsNetwork-PY0220ENSkillsNetwork/labs/project/revenue.htm> . Save the text of the response as a variable named `html_data`.

```
# Question 2
# Step 1 Send HTTP requests to the web page
url = "https://cf-courses-data.s3.us.cloud-object-
```

```
age
ud-object-storage.appdomain.cloud/IBMDDeveloperSkil
```

```
IBMDDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/l
```

```
k-PY0220EN-SkillsNetwork/labs/project/revenue.htm"
```

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

Parse the `html` data using `beautiful_soup`.

```
# Step 2 Parse the HTML content
beautiful_soup = BeautifulSoup(html_data, )
```

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 3 Identify the HTML tags
tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])
```

```
# Step 4 Use a BeautifulSoup method for extracting data
for row in beautiful_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})
```

```
method for extracting data
tbody').find_all('tr'):

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

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

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

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

```
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.

```
tesla_revenue.dropna(inplace=True)

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

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 (Extracting GameStop Stock Data Using *yfinance*)

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`.

```
#Question_3
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.

```
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.

```
gme_data.reset_index(inplace=True)
```

```
gme_data.head()
```



	index	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	0	2002-02-13	1.620128	1.693350	1.603296	1.691666	76216000	0.0	0.0
1	1	2002-02-14	1.712707	1.716074	1.670626	1.683250	11021600	0.0	0.0
2	2	2002-02-15	1.683251	1.687459	1.658002	1.674834	8389600	0.0	0.0
3	3	2002-02-19	1.666418	1.666418	1.578047	1.607504	7410400	0.0	0.0
4	4	2002-02-20	1.615921	1.662210	1.603296	1.662210	6892800	0.0	0.0

## Question 4: Use Webscraping to Extract GME Revenue Data (Extracting GameStop Revenue Data Using Webscraping)

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

```
#Question_4
# Step 1 Send HTTP requests to the web page
url1 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"
```

```
e
d-object-storage.appdomain.cloud/IBMDDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"
```

```
html_data = requests.get(url1).text
print(html_data)
```

Parse the html data using `beautiful_soup`.

```
# Step 2 Parse the HTML content
beautiful_soup = BeautifulSoup(html_data,)
```

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.

```
# Step 3 Identify the HTML tags
gme_revenue = pd.DataFrame(columns=["Date", "Revenue"])
```

```
# Step 4 Use a BeautifulSoup method for extracting data
for row in beautiful_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})
```

```
method for extracting data
('tbody').find_all('tr'):

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

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

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



```
gme_revenue.dropna(inplace=True)
```

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

```
gme_revenue.tail()
```

	Date	Revenue
8	2012	203
9	2011	413
10	2010	204
11	2009	117
12	2008	112

### Question 5: Plot Tesla Stock Graph (Tesla Stock and Revenue Dashboard)

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.

```
# Question 5
def make_graph(tesla_data, tesla_revenue, 'Tesla'):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Historical Share Price", "Historical Revenue"), vertical_spacing=.3)
    stock_data_specific = tesla_data[tesla_data.Date <= '2021-06-14']
    revenue_data_specific = tesla_revenue[revenue_data.Date <= '2021-06-14']
    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_data_specific.Date,
                      xaxis_rangeslider_visible=True)
    fig.show()
```

```
):
    xaxes=True, subplot_titles=("Historical Share Price", "Historical Revenue"), vertical_spacing=.3)
    .Date <= '2021-06-14']
    ue_data.Date <= '2021-06-14']
    tock_data_specific.Date, infer_datetime_format=True), y=stock_data_specific.Close.astype("float"), name="Share Price"), row=1, col=1)
    evenue_data_specific.Date, infer_datetime_format=True), y=revenue_data_specific.Revenue.astype("float"), name="Revenue"), row=2, col=1)
    col=1)
    row=1, col=1)
    illions)", row=2, col=1)
```

## Question 6: Plot GameStop Stock Graph (GameStop Stock and Revenue Dashboard)

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.

```
# Question 6
def make_graph(gme_data,gme_revenue,'GameStop'):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Historical Share Price", "Historical Revenue"), vertical_spacing=.3)
    stock_data_specific1 = gme_data[tesla_data.Date <= '2021-06-14']
    revenue_data_specific1 = gme_revenue[revenue_data.Date <= '2021-06-14']
    fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific1.Date, infer_datetime_format=True), y=stock_data_specific1.Close.astype("float"), name="Share Price"), row=1, col=1)
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific1.Date, infer_datetime_format=True), y=revenue_data_specific1.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()
```

```
fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Historical Share Price", "Historical Revenue"), vertical_spacing=.3)
stock_data_specific1 = gme_data[tesla_data.Date <= '2021-06-14']
revenue_data_specific1 = gme_revenue[revenue_data.Date <= '2021-06-14']
fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific1.Date, infer_datetime_format=True), y=stock_data_specific1.Close.astype("float"), name="Share Price"), row=1, col=1)
fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific1.Date, infer_datetime_format=True), y=revenue_data_specific1.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()
```