## 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`.

[14]:
```
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.

[15]:
```
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.

[16]:
```
tesla_data.reset_index(inplace=True)
tesla_data.head()
```

[16]:

| | 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`.

[7]:
```
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`.

[8]:
```
soup = BeautifulSoup(html_data, 'html5lib')
```

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

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

[9]:
```
tesla_revenue= pd.read_html(url, match="Tesla Quarterly Revenue", flavor='bs4')[0]
tesla_revenue=tesla_revenue.rename(columns = {'Tesla Quarterly Revenue(Millions of US $)': 'Date', 'Tesla Quarterly Revenue(Millions of US $).1': 'Revenue'}, inplace = False)
tesla_revenue["Revenue"] = tesla_revenue["Revenue"].str.replace(",","").str.replace("$","")
tesla_revenue.head()
```

[9]:

| | Date | Revenue |
|---|---|---|
| **0** | 2022-09-30 | 21454 |
| **1** | 2022-06-30 | 16934 |
| **2** | 2022-03-31 | 18756 |
| **3** | 2021-12-31 | 17719 |
| **4** | 2021-09-30 | 13757 |

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

[11]:
```
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.

[12]:
```
tesla_revenue.tail()
```

[12]:

| | 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`.

[13]:
```
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.

[14]:
```
gme_data=gamestop.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.

[15]:
```
gme_data.reset_index(inplace=True)
gme_data.head()
```

[15]:

| | Date | Open | High | Low | Close | Volume | Dividends | Stock Splits |
|---|---|---|---|---|---|---|---|---|
| **0** | 2002-02-13 00:00:00-05:00 | 1.620128 | 1.693350 | 1.603296 | 1.691667 | 76216000 | 0.0 | 0.0 |
| **1** | 2002-02-14 00:00:00-05:00 | 1.712707 | 1.716074 | 1.670626 | 1.683251 | 11021600 | 0.0 | 0.0 |
| **2** | 2002-02-15 00:00:00-05:00 | 1.683250 | 1.687458 | 1.658002 | 1.674834 | 8389600 | 0.0 | 0.0 |
| **3** | 2002-02-19 00:00:00-05:00 | 1.666418 | 1.666418 | 1.578047 | 1.607504 | 7410400 | 0.0 | 0.0 |
| **4** | 2002-02-20 00:00:00-05:00 | 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`.

[16]:
```
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`.

[20]:
```
soup = BeautifulSoup(html_data,"html5lib")
```

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

[23]:
```
gme_revenue= pd.read_html(url, match="GameStop Quarterly Revenue", flavor='bs4')[0]
gme_revenue=gme_revenue.rename(columns = {'GameStop Quarterly Revenue(Millions of US $)': 'Date', 'GameStop Quarterly Revenue(Millions of US $).1': 'Revenue'}, inplace = False)
gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace(",","").str.replace("$","")
```

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

[24]:
```
gme_revenue.tail()
```

[24]:

| | 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.

[25]:
```
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.

[26]:
```
make_graph(gme_data, gme_revenue, 'GameStop Stock Graph')
```