**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]:



**import** yfinance **as** yf

​

*# Create a Ticker object for Tesla*

ticker **=** 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.

[10]:



*# Extract stock information and save it in a dataframe*

tesla\_data **=** ticker.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.

[11]:



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.

[20]:



**import** requests

**from** bs4 **import** BeautifulSoup

**import** pandas **as** pd

​

*# Download the webpage*

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.

[21]:



*# Parse the HTML data*

soup **=** BeautifulSoup(html\_data, "html.parser")

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

[22]:



*# Extract the table with Tesla Quarterly Revenue and store it in a dataframe*

table **=** soup.find("table")

tesla\_revenue **=** pd.read\_html(str(table))[0]

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

[26]:



*# Extract the table with Tesla Quarterly Revenue and store it in a dataframe*

table **=** soup.find("table")

tesla\_revenue **=** pd.read\_html(str(table))[0]

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

[27]:



*# Remove 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.

[24]:



*# Display the last 5 rows of the tesla\_revenue dataframe*

print(tesla\_revenue.tail())

Tesla Annual Revenue(Millions of US $) \

8 2013

9 2012

10 2011

11 2010

12 2009

Tesla Annual Revenue(Millions of US $).1

8 $2,013

9 $413

10 $204

11 $117

12 $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.

[28]:



**import** yfinance **as** yf

​

*# Create a ticker object*

ticker **=** 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.

[29]:



*# Extract stock information and save it in a dataframe*

gme\_data **=** ticker.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.

[30]:



*# Reset the index*

gme\_data.reset\_index(inplace**=True**)

​

*#Display the first five rows of the gme\_data dataframe*

print(gme\_data.head())

​

Date Open High Low Close Volume Dividends \

0 2002-02-13 1.620128 1.693350 1.603296 1.691666 76216000 0.0

1 2002-02-14 1.712707 1.716074 1.670626 1.683250 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 7410400 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.

[33]:



**import** requests

**import** pandas **as** pd

**from** bs4 **import** BeautifulSoup

​

*# download webpage content*

url **=** 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html'

response **=** requests.get(url)

Parse the html data using beautiful\_soup.

[34]:



*# parse html data*

html\_data **=** BeautifulSoup(response.content, 'html.parser')

​

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

[35]:



*# extract GameStop Quarterly Revenue table*

gme\_revenue **=** pd.read\_html(str(html\_data.find\_all('table')[1]))[0]

gme\_revenue.columns **=** ['Date', 'Revenue']

gme\_revenue['Revenue'] **=** gme\_revenue['Revenue'].str.replace(',', '').str.replace('$', '').astype(float)

​

/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/ipykernel\_launcher.py:4: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will \*not\* be treated as literal strings when regex=True.

after removing the cwd from sys.path.

Display the last five rows of the gme\_revenue dataframe using the tail function. Take a screenshot of the results.

[36]:



*# display last five rows of gme\_revenue*

gme\_revenue.tail()

[36]:

|  | **Date** | **Revenue** |
| --- | --- | --- |
| **57** | 2006-01-31 | 1667.0 |
| **58** | 2005-10-31 | 534.0 |
| **59** | 2005-07-31 | 416.0 |
| **60** | 2005-04-30 | 475.0 |
| **61** | 2005-01-31 | 709.0 |

**Question 5: Plot Tesla Stock Graph**

**Question 6: Plot GameStop 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.

[45]:



make\_graph(tesla\_data, tesla\_revenue, 'Tesla')

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.

[ ]:



make\_graph(gme\_data, gme\_revenue, 'GameStop')