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 in IBM Cloud Watson Studio, please replace the command for installing nbformat from `!pip install nbformat==4.2.0` to simply `!pip install nbformat`

In [101…

```
!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/py
thon/lib/python3.7/site-packages (0.1.67)
Requirement already satisfied: pandas>=0.24 in /home/jupyterlab/conda/envs/pytho
n/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/pyth
on/lib/python3.7/site-packages (from yfinance==0.1.67) (2.29.0)
Requirement already satisfied: multitasking>=0.0.7 in /home/jupyterlab/conda/env
s/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/e
nvs/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/pytho
n/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/cond
a/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/pytho
n/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/en
vs/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/li
b/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
pkgs/main/linux-64 ━━━━━━━━━━━━━━━━━━━━━━━━   0.0 B /  ??.?MB @  ??.?MB/s  0.0sp
kgs/main/noarch                                             No change
```

```
pkgs/main/linux-64                                              No change
pkgs/r/linux-64                                                 No change
pkgs/r/noarch                                                   No change

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/pyt
hon/lib/python3.7/site-packages (4.2.0)
Requirement already satisfied: ipython-genutils in /home/jupyterlab/conda/envs/py
thon/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/pytho
n/lib/python3.7/site-packages (from nbformat==4.2.0) (4.12.0)
Requirement already satisfied: traitlets>=4.1 in /home/jupyterlab/conda/envs/pyth
on/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/pytho
n/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (2
3.1.0)
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/con
da/envs/python/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbforma
t==4.2.0) (5.12.0)
Requirement already satisfied: pkgutil-resolve-name>=1.3.10 in /home/jupyterlab/c
onda/envs/python/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbfor
mat==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/p
ython/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 [103…

```python
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 [104…

```
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 [124…

```
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, infer_da
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date, infer_
    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 [106…

```
import yfinance as yf

# Ticker symbol for Tesla
ticker_symbol = "TSLA"

# Create a ticker object
tesla_ticker = yf.Ticker(ticker_symbol)

# Get historical data for the stock
tesla_data = tesla_ticker.history(period="1d", interval="1m")  # Adjust the peri

# Display the historical data
print(tesla_data)
```

```
                                 Open        High         Low       Close  \
Datetime
2023-11-14 09:30:00-05:00  233.839996  233.889999  233.779999  233.865005
2023-11-14 09:31:00-05:00  233.869995  233.990005  231.880005  232.029999
2023-11-14 09:32:00-05:00  232.054901  233.279999  231.899994  233.259995
2023-11-14 09:33:00-05:00  233.270004  233.899994  232.039993  232.410095
2023-11-14 09:34:00-05:00  232.429993  232.619995  231.350006  231.714996
...                               ...         ...         ...         ...
2023-11-14 15:56:00-05:00  237.080505  237.089996  236.720001  237.014999
2023-11-14 15:57:00-05:00  237.014999  237.070007  236.910004  236.960007
2023-11-14 15:58:00-05:00  236.960007  237.000000  236.800003  236.899994
2023-11-14 15:59:00-05:00  236.889999  237.500000  236.820007  237.404999
2023-11-14 16:00:00-05:00  237.410004  237.410004  237.410004  237.410004

                             Volume  Dividends  Stock Splits
Datetime
2023-11-14 09:30:00-05:00  10521304          0             0
2023-11-14 09:31:00-05:00   1439779          0             0
2023-11-14 09:32:00-05:00   1175673          0             0
2023-11-14 09:33:00-05:00   1372213          0             0
2023-11-14 09:34:00-05:00   1163731          0             0
...                             ...        ...           ...
2023-11-14 15:56:00-05:00    626393          0             0
2023-11-14 15:57:00-05:00    395738          0             0
2023-11-14 15:58:00-05:00    503345          0             0
2023-11-14 15:59:00-05:00    841059          0             0
2023-11-14 16:00:00-05:00         0          0             0

[391 rows x 7 columns]
```

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 [107…

```python
import yfinance as yf
import pandas as pd

# Ticker symbol for Tesla
ticker_symbol = "TSLA"

# Create a ticker object
tesla_ticker = yf.Ticker(ticker_symbol)

# Get historical data for the stock with period set to max
tesla_data = tesla_ticker.history(period="max")

# Display the historical data
print(tesla_data)

# Save the data to a DataFrame
tesla_data_df = pd.DataFrame(tesla_data)

# Display the first few rows of the DataFrame
print(tesla_data_df.head())
```

```
                  Open        High         Low       Close     Volume  \
Date                                                                    
2010-06-29    1.266667    1.666667    1.169333    1.592667  281494500   
2010-06-30    1.719333    2.028000    1.553333    1.588667  257806500   
2010-07-01    1.666667    1.728000    1.351333    1.464000  123282000   
2010-07-02    1.533333    1.540000    1.247333    1.280000   77097000   
2010-07-06    1.333333    1.333333    1.055333    1.074000  103003500   
...                ...         ...         ...         ...        ...   
2023-11-08  223.149994  224.149994  217.639999  222.110001  106584800   
2023-11-09  219.750000  220.800003  206.679993  209.979996  142110500   
2023-11-10  210.029999  215.380005  205.690002  214.649994  130994000   
2023-11-13  215.600006  225.399994  211.610001  223.710007  140447600   
2023-11-14  235.029999  238.139999  230.720001  237.410004  148611400   

            Dividends  Stock Splits  
Date                                 
2010-06-29          0           0.0  
2010-06-30          0           0.0  
2010-07-01          0           0.0  
2010-07-02          0           0.0  
2010-07-06          0           0.0  
...               ...           ...  
2023-11-08          0           0.0  
2023-11-09          0           0.0  
2023-11-10          0           0.0  
2023-11-13          0           0.0  
2023-11-14          0           0.0  

[3369 rows x 7 columns]
                Open      High       Low     Close     Volume  Dividends  \
Date                                                                       
2010-06-29  1.266667  1.666667  1.169333  1.592667  281494500          0   
2010-06-30  1.719333  2.028000  1.553333  1.588667  257806500          0   
2010-07-01  1.666667  1.728000  1.351333  1.464000  123282000          0   
2010-07-02  1.533333  1.540000  1.247333  1.280000   77097000          0   
2010-07-06  1.333333  1.333333  1.055333  1.074000  103003500          0   

            Stock Splits  
Date                      
2010-06-29           0.0  
2010-06-30           0.0  
2010-07-01           0.0  
2010-07-02           0.0  
2010-07-06           0.0  
```

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 [108…

```python
import yfinance as yf
import pandas as pd

# Ticker symbol for Tesla
ticker_symbol = "TSLA"

# Create a ticker object
tesla_ticker = yf.Ticker(ticker_symbol)
```

```
# Get historical data for the stock with period set to max
tesla_data = tesla_ticker.history(period="max")

# Save the data to a DataFrame
tesla_data_df = pd.DataFrame(tesla_data)

# Reset the index in-place
tesla_data_df.reset_index(inplace=True)

# Display the first five rows of the DataFrame
print(tesla_data_df.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`.

In [109…

```
import requests

# URL of the webpage
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDev

# Send a GET request to the URL
response = requests.get(url)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Save the HTML content as a variable named html_data
    html_data = response.text
    print("HTML data downloaded successfully.")
else:
    print(f"Failed to download HTML data. Status code: {response.status_code}")
```

```
HTML data downloaded successfully.
```

Parse the html data using `beautiful_soup`.

In [110…

```
# Parse the HTML using BeautifulSoup
soup = BeautifulSoup(html_data, 'html.parser')
```

```
# Now, you can work with the parsed HTML content using BeautifulSoup functions
# For example, let's print the title of the webpage
title = soup.title
print(f"Webpage Title: {title.text}")
```

```
Webpage Title: Tesla Revenue 2010-2022 | TSLA | MacroTrends
```

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 [111…

```
# Find the table using BeautifulSoup
table = soup.find('table')

# Extract data from the table
data = []
for row in table.find_all('tr')[1:]:  # Skip the header row
    cols = row.find_all(['th', 'td'])
    date = cols[0].text.strip()
    revenue = cols[1].text.strip().replace('$', '').replace(',', '')
    data.append({'Date': date, 'Revenue': revenue})

# Create a DataFrame
tesla_revenue = pd.DataFrame(data)

# Display the DataFrame
print(tesla_revenue)
```

```
    Date Revenue
0   2021   53823
1   2020   31536
2   2019   24578
3   2018   21461
4   2017   11759
5   2016    7000
6   2015    4046
7   2014    3198
8   2013    2013
9   2012     413
10  2011     204
11  2010     117
12  2009     112
```

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

In [112…

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

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

In [113…

```
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 [114…

```
print(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

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 [115…

```
# Ticker symbol for GameStop
ticker_symbol = "GME"

# Create a ticker object
gme_ticker = yf.Ticker(ticker_symbol)

# Get historical data for the stock
gme_data = gme_ticker.history(period="1d", interval="1m")  # Adjust the period a

# Display the historical data
print(gme_data)
```

```
                              Open     High    Low    Close  Volume  \
Datetime                                                              
2023-11-14 09:30:00-05:00  12.7500  12.7900  12.720  12.7503  166652   
2023-11-14 09:31:00-05:00  12.7600  12.7600  12.690  12.7100   29312   
2023-11-14 09:32:00-05:00  12.7085  13.0000  12.695  12.9900  208772   
2023-11-14 09:33:00-05:00  12.9935  13.0600  12.970  12.9950   56890   
2023-11-14 09:34:00-05:00  13.0000  13.0200  12.930  12.9500   35527   
...                            ...      ...     ...      ...     ...   
2023-11-14 15:56:00-05:00  12.9199  12.9199  12.910  12.9150   15141   
2023-11-14 15:57:00-05:00  12.9150  12.9200  12.910  12.9100   28433   
2023-11-14 15:58:00-05:00  12.9100  12.9200  12.910  12.9100   28788   
2023-11-14 15:59:00-05:00  12.9200  12.9200  12.900  12.9000  121918   
2023-11-14 16:00:00-05:00  12.9000  12.9000  12.900  12.9000       0   

                           Dividends  Stock Splits  
Datetime                                            
2023-11-14 09:30:00-05:00          0             0  
2023-11-14 09:31:00-05:00          0             0  
2023-11-14 09:32:00-05:00          0             0  
2023-11-14 09:33:00-05:00          0             0  
2023-11-14 09:34:00-05:00          0             0  
...                              ...           ...  
2023-11-14 15:56:00-05:00          0             0  
2023-11-14 15:57:00-05:00          0             0  
2023-11-14 15:58:00-05:00          0             0  
2023-11-14 15:59:00-05:00          0             0  
2023-11-14 16:00:00-05:00          0             0  

[391 rows x 7 columns]
```

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 [116…

```
# Create a ticker object
gme_ticker = yf.Ticker(ticker_symbol)

# Get historical data for the stock with period set to max
gme_data = gme_ticker.history(period="max")

# Display the historical data
print(gme_data)

# Save the data to a DataFrame
gme_data_df = pd.DataFrame(gme_data)

# Display the first few rows of the DataFrame
print(gme_data_df.head())
```

```
                 Open       High        Low      Close    Volume  Dividends  \
Date
2002-02-13   1.620128   1.693350   1.603296   1.691667  76216000        0.0
2002-02-14   1.712707   1.716074   1.670626   1.683250  11021600        0.0
2002-02-15   1.683250   1.687458   1.658002   1.674834   8389600        0.0
2002-02-19   1.666417   1.666417   1.578047   1.607504   7410400        0.0
2002-02-20   1.615921   1.662210   1.603296   1.662210   6892800        0.0
...               ...        ...        ...        ...       ...        ...
2023-11-08  13.510000  13.760000  13.280000  13.280000   1705600        0.0
2023-11-09  13.250000  13.320000  12.700000  12.700000   2750100        0.0
2023-11-10  12.810000  12.970000  12.350000  12.540000   3872400        0.0
2023-11-13  12.500000  12.530000  11.830000  12.140000   4318500        0.0
2023-11-14  12.750000  13.390000  12.690000  12.900000   5184500        0.0

            Stock Splits
Date
2002-02-13           0.0
2002-02-14           0.0
2002-02-15           0.0
2002-02-19           0.0
2002-02-20           0.0
...                  ...
2023-11-08           0.0
2023-11-09           0.0
2023-11-10           0.0
2023-11-13           0.0
2023-11-14           0.0

[5477 rows x 7 columns]
                Open      High       Low     Close    Volume  Dividends  \
Date
2002-02-13  1.620128  1.693350  1.603296  1.691667  76216000        0.0
2002-02-14  1.712707  1.716074  1.670626  1.683250  11021600        0.0
2002-02-15  1.683250  1.687458  1.658002  1.674834   8389600        0.0
2002-02-19  1.666417  1.666417  1.578047  1.607504   7410400        0.0
2002-02-20  1.615921  1.662210  1.603296  1.662210   6892800        0.0

            Stock Splits
Date
2002-02-13           0.0
2002-02-14           0.0
2002-02-15           0.0
2002-02-19           0.0
2002-02-20           0.0
```

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 [117…

```
# Reset the index in-place
gme_data_df.reset_index(inplace=True)

# Display the first five rows of the DataFrame
print(gme_data_df.head())
```

```
         Date      Open      High       Low     Close    Volume  Dividends  \
0  2002-02-13  1.620128  1.693350  1.603296  1.691667  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  1.666417  1.666417  1.578047  1.607504   7410400        0.0
4  2002-02-20  1.615921  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`.

In [118…

```
# URL of the webpage
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDev

# Send a GET request to the URL
response = requests.get(url)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Save the HTML content as a variable named html_data
    html_data = response.text
    print("HTML data downloaded successfully.")
else:
    print(f"Failed to download HTML data. Status code: {response.status_code}")
```

```
HTML data downloaded successfully.
```

Parse the html data using `beautiful_soup`.

In [119…

```
soup = BeautifulSoup(html_data, 'html.parser')

# Now, you can work with the parsed HTML content using BeautifulSoup functions
# For example, let's print the title of the webpage
title = soup.title
print(f"Webpage Title: {title.text}")
```

```
Webpage Title: GameStop Revenue 2006-2020 | GME | MacroTrends
```

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 [120…

```
# Find the table using BeautifulSoup
table = soup.find('table')

# Extract data from the table
data = []
for row in table.find_all('tr')[1:]:  # Skip the header row
    cols = row.find_all(['th', 'td'])
    date = cols[0].text.strip()
    revenue = cols[1].text.strip().replace('$', '').replace(',', '')
    data.append({'Date': date, 'Revenue': revenue})

# Create a DataFrame
gme_revenue = pd.DataFrame(data)

# Display the DataFrame
print(gme_revenue)
```

```
    Date Revenue
0   2020    6466
1   2019    8285
2   2018    8547
3   2017    7965
4   2016    9364
5   2015    9296
6   2014    9040
7   2013    8887
8   2012    9551
9   2011    9474
10  2010    9078
11  2009    8806
12  2008    7094
13  2007    5319
14  2006    3092
15  2005    1843
```

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

In [121…

```
print(gme_revenue.tail())
```

```
    Date Revenue
11  2009    8806
12  2008    7094
13  2007    5319
14  2006    3092
15  2005    1843
```

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 [125…

```
import matplotlib.pyplot as plt

# Assuming you have tesla_data and tesla_revenue DataFrames
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.

In [126…

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