Final Assignment

March 3, 2025

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

```
<ul>
    <li>Define a Function that Makes a Graph</li>
    <li>Question 1: Use yfinance to Extract Stock Data</li>
    <li>Question 2: Use Webscraping to Extract Tesla Revenue Data</li>
    <li>Question 3: Use yfinance to Extract Stock Data</li>
    <li>Question 4: Use Webscraping to Extract GME Revenue Data</li>
    <li>Question 5: Plot Tesla Stock Graph</li>
    <li>Question 6: Plot GameStop Stock Graph</li>
</ul>
```

Estimated Time Needed: 30 min

Note:- If you are working Locally using anaconda, please uncomment the following code and execute it. Use the version as per your python version.

```
[1]:  !pip install yfinance
      !pip install bs4
      !pip install nbformat
      !pip install --upgrade plotly
```

Collecting yfinance
  Downloading yfinance-0.2.54-py2.py3-none-any.whl.metadata (5.8 kB)
Collecting pandas>=1.3.0 (from yfinance)
  Downloading
pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(89 kB)
Collecting numpy>=1.16.5 (from yfinance)
  Downloading
numpy-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(62 kB)
Requirement already satisfied: requests>=2.31 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.32.3)

```
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.11-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: platformdirs>=2.0.0 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.3.6)
Requirement already satisfied: pytz>=2022.5 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2024.2)
Requirement already satisfied: frozendict>=2.3.4 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (2.4.6)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.17.9.tar.gz (3.0 MB)
                            3.0/3.0 MB
106.3 MB/s eta 0:00:00
  Installing build dependencies … one
  Getting requirements to build wheel … done
  Preparing metadata (pyproject.toml) … done
Requirement already satisfied: beautifulsoup4>=4.11.1 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-
packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.12/site-packages (from pandas>=1.3.0->yfinance)
(2.9.0.post0)
Collecting tzdata>=2022.7 (from pandas>=1.3.0->yfinance)
  Downloading tzdata-2025.1-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: charset_normalizer<4,>=2 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.12/site-
packages (from requests>=2.31->yfinance) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance)
(2024.12.14)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance) (1.17.0)
Downloading yfinance-0.2.54-py2.py3-none-any.whl (108 kB)
Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Downloading
numpy-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.1 MB)
                            16.1/16.1 MB
108.4 MB/s eta 0:00:00
Downloading
pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.7
MB)
                            12.7/12.7 MB
103.7 MB/s eta 0:00:00
Downloading tzdata-2025.1-py2.py3-none-any.whl (346 kB)
Building wheels for collected packages: peewee
```

```
  Building wheel for peewee (pyproject.toml) … one
  Created wheel for peewee:
filename=peewee-3.17.9-cp312-cp312-linux_x86_64.whl size=303802
sha256=3c4fae939d19614d2e6d5d57ec234c050032216b4db2511ba6a40260d291f177
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/43/ef/2d/2c51d496bf084
945ffdf838b4cc8767b8ba1cc20eb41588831
Successfully built peewee
Installing collected packages: peewee, multitasking, tzdata, numpy, pandas,
yfinance
Successfully installed multitasking-0.0.11 numpy-2.2.3 pandas-2.2.3
peewee-3.17.9 tzdata-2025.1 yfinance-0.2.54
Collecting bs4
  Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.12/site-
packages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-
packages (from beautifulsoup4->bs4) (2.5)
Downloading bs4-0.0.2-py2.py3-none-any.whl (1.2 kB)
Installing collected packages: bs4
Successfully installed bs4-0.0.2
Requirement already satisfied: nbformat in /opt/conda/lib/python3.12/site-
packages (5.10.4)
Requirement already satisfied: fastjsonschema>=2.15 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (2.21.1)
Requirement already satisfied: jsonschema>=2.6 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (4.23.0)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (5.7.2)
Requirement already satisfied: traitlets>=5.1 in /opt/conda/lib/python3.12/site-
packages (from nbformat) (5.14.3)
Requirement already satisfied: attrs>=22.2.0 in /opt/conda/lib/python3.12/site-
packages (from jsonschema>=2.6->nbformat) (25.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/opt/conda/lib/python3.12/site-packages (from jsonschema>=2.6->nbformat)
(2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in
/opt/conda/lib/python3.12/site-packages (from jsonschema>=2.6->nbformat)
(0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in /opt/conda/lib/python3.12/site-
packages (from jsonschema>=2.6->nbformat) (0.22.3)
Requirement already satisfied: platformdirs>=2.5 in
/opt/conda/lib/python3.12/site-packages (from jupyter-
core!=5.0.*,>=4.12->nbformat) (4.3.6)
Requirement already satisfied: typing-extensions>=4.4.0 in
/opt/conda/lib/python3.12/site-packages (from
referencing>=0.28.4->jsonschema>=2.6->nbformat) (4.12.2)
Requirement already satisfied: plotly in /opt/conda/lib/python3.12/site-packages
(5.24.1)
```

```
Collecting plotly
  Downloading plotly-6.0.0-py3-none-any.whl.metadata (5.6 kB)
Collecting narwhals>=1.15.1 (from plotly)
  Downloading narwhals-1.28.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: packaging in /opt/conda/lib/python3.12/site-
packages (from plotly) (24.2)
Downloading plotly-6.0.0-py3-none-any.whl (14.8 MB)
                         14.8/14.8 MB
184.8 MB/s eta 0:00:00
Downloading narwhals-1.28.0-py3-none-any.whl (308 kB)
Installing collected packages: narwhals, plotly
  Attempting uninstall: plotly
    Found existing installation: plotly 5.24.1
    Uninstalling plotly-5.24.1:
      Successfully uninstalled plotly-5.24.1
Successfully installed narwhals-1.28.0 plotly-6.0.0
```

```python
[111]: 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
```

```python
[112]: import plotly.io as pio
       pio.renderers.default = "iframe"
```

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.

```python
[113]: import warnings
       # Ignore all warnings
       warnings.filterwarnings("ignore", category=FutureWarning)
```

0.1 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.**

```python
[114]: 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 = .3)
           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"),␣
↪name="Share Price"), row=1, col=1)
    #fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific.Date),␣
↪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.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date),␣
↪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,
    xaxis_rangeslider_visible=True)
    fig.show()
    from IPython.display import display, HTML
    fig_html = fig.to_html()
    display(HTML(fig_html))
```

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.**

0.2 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`.

```
[115]: 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.

```
[116]: 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.

```
[117]: tesla_data.reset_index(inplace=True)
       # Display the downloaded data
       tesla_data.head()
```

```
[117]:                        Date      Open      High       Low     Close  \
       0 2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667
       1 2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667
       2 2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000
       3 2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000
       4 2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000

             Volume  Dividends  Stock Splits
       0  281494500        0.0           0.0
       1  257806500        0.0           0.0
       2  123282000        0.0           0.0
       3   77097000        0.0           0.0
       4  103003500        0.0           0.0
```

0.3 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`.

```
[118]: 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
       #print(html_data)
```

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

```
[119]: soup = BeautifulSoup(html_data, 'html.parser')
       #print(soup)
```

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

Here are the step-by-step instructions:

1. Create an Empty DataFrame
2. Find the Relevant Table
3. Check for the Tesla Quarterly Revenue Table
4. Iterate Through Rows in the Table Body
5. Extract Data from Columns
6. Append Data to the DataFrame

Click here if you need help locating the table

Below is the code to isolate the table, you will now need to loop through the rows and columns

`soup.find_all("tbody")[1]`

If you want to use the `read_html` function the table is located at index 1

We are focusing on quarterly revenue in the lab.

```
[120]: tesla_revenue = pd.DataFrame()
# soup.find(td)


# First we isolate the body of the table which contains all the information
# Then we loop through each row and find all the column values for each row

#for row in soup.find("historical_data_table table").find_all('tr'):

foo=soup.find_all("tbody")[1]
#print(foo)

for row in soup.find_all("tbody")[1].find_all('tr'):
    col = row.find_all("td")
    Date = col[0].text
    Revenue = col[1].text
    #high = col[2].text
    #low = col[3].text
    #close = col[4].text
    #adj_close = col[5].text
    #volume = col[6].text
    tesla_revenue = pd.concat([tesla_revenue,pd.DataFrame({"Date":[Date],␣
    ↪"Revenue":[Revenue]})], ignore_index=True)

    #LOOPS through and print each item and date


# Finally we append the data of each row to the table
#tesla_revenue = pd.concat([tesla_revenue,pd.DataFrame({"Date":[date], "Open":
    ↪[Open], "High":[high], "Low":[low], "Close":[close], "Adj Close":
    ↪[adj_close], "Volume":[volume]})], ignore_index=True)

#tesla_revenue
```

```
[ ]:
```

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

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

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

```
[122]: 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.

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

```
[123]:         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
```

0.4 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`.

```
[124]: 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.

```
[73]: 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.

```
[75]: gme_data.reset_index(inplace=True)
      # Display the downloaded data
      gme_data.head()
```

```
[75]:    index                      Date      Open      High       Low     Close  \
       0      0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666
       1      1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250
       2      2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658001  1.674834
       3      3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504
       4      4 2002-02-20 00:00:00-05:00  1.615920  1.662210  1.603296  1.662210
```

```
      Volume  Dividends  Stock Splits
0  76216000        0.0           0.0
1  11021600        0.0           0.0
2   8389600        0.0           0.0
3   7410400        0.0           0.0
4   6892800        0.0           0.0
```

0.5 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`.

```
[76]: url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
      ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"
      html_data2 = requests.get(url).text
      # print(html_data2)
```

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

```
[77]: soup1 = BeautifulSoup(html_data2, 'html.parser')
      #print(soup1)
```

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

Below is the code to isolate the table, you will now need to loop through the rows and columns

soup.find_all("tbody")[1]

If you want to use the read_html function the table is located at index 1

```
[85]: gme_revenue = pd.DataFrame()
      # soup.find(td)


      # First we isolate the body of the table which contains all the information
      # Then we loop through each row and find all the column values for each row

      #for row in soup.find("historical_data_table table").find_all('tr'):
```

```
bar=soup1.find_all("tbody")[1]
#print(bar)

for row in soup1.find_all("tbody")[1].find_all('tr'):
    col = row.find_all("td")
    Date = col[0].text
    Revenue = col[1].text.replace('$','').replace(',','')
    #high = col[2].text
    #low = col[3].text
    #close = col[4].text
    #adj_close = col[5].text
    #volume = col[6].text
    gme_revenue = pd.concat([gme_revenue,pd.DataFrame({"Date":[Date], "Revenue":
 ↪[Revenue]})], ignore_index=True)

    #LOOPS through and print each item and date


# Finally we append the data of each row to the table
#tesla_revenue = pd.concat([tesla_revenue,pd.DataFrame({"Date":[date], "Open":
 ↪[Open], "High":[high], "Low":[low], "Close":[close], "Adj Close":
 ↪[adj_close], "Volume":[volume]})], ignore_index=True)

gme_revenue
```

[85]:
```
          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
..         ...      ...
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

[62 rows x 2 columns]
```

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

[87]: `gme_revenue.tail()`

[87]:
```
          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
```

0.6 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

You just need to invoke the make_graph function with the required parameter to print the graphs

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

```
/tmp/ipykernel_300/655126992.py:5: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.

/tmp/ipykernel_300/655126992.py:7: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.


<IPython.core.display.HTML object>
```

0.7 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

You just need to invoke the make_graph function with the required parameter to print the graphs

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

```
/tmp/ipykernel_300/655126992.py:5: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
```

```
can safely remove this argument.

/tmp/ipykernel_300/655126992.py:7: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.


<IPython.core.display.HTML object>
```

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

0.8   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

##