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 Locally using anaconda, please uncomment the following code and execute it. Use the version as per your python version.

In [1]:
```
!pip install yfinance
!pip install pandas
!pip install nbformat
```


```
Requirement already satisfied: yfinance in /opt/conda/lib/python3.11/site-packages (0.2.36)
Requirement already satisfied: pandas>=1.3.0 in /opt/conda/lib/python3.11/site-packages (from yfinance) (2.2.1)
Requirement already satisfied: numpy>=1.16.5 in /opt/conda/lib/python3.11/site-packages (from yfinance) (2.1.3)
Requirement already satisfied: requests>=2.31 in /opt/conda/lib/python3.11/site-packages (from yfinance) (2.31.0)
Requirement already satisfied: multitasking>=0.0.7 in /opt/conda/lib/python3.11/site-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in /opt/conda/lib/python3.11/site-packages (from yfinance) (5.1.0)
Requirement already satisfied: platformdirs>=2.0.0 in /opt/conda/lib/python3.11/site-packages (from yfinance) (4.2.1)
Requirement already satisfied: pytz>=2022.5 in /opt/conda/lib/python3.11/site-packages (from yfinance) (2024.1)
Requirement already satisfied: frozendict>=2.3.4 in /opt/conda/lib/python3.11/site-packages (from yfinance) (2.4.4)
Requirement already satisfied: peewee>=3.16.2 in /opt/conda/lib/python3.11/site-packages (from yfinance) (3.17.8)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /opt/conda/lib/python3.11/site-packages (from yfinance) (4.12.3)
Requirement already satisfied: html5lib>=1.1 in /opt/conda/lib/python3.11/site-packages (from yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.11/site-packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: six>=1.9 in /opt/conda/lib/python3.11/site-packages (from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.11/site-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.11/site-packages (from pandas>=1.3.0->yfinance) (2.9.0)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.11/site-packages (from pandas>=1.3.0->yfinance) (2024.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python3.11/site-packages (from requests>=2.31->yfinance) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.11/site-packages (from requests>=2.31->yfinance) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.11/site-packages (from requests>=2.31->yfinance) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.11/site-packages (from requests>=2.31->yfinance) (2024.8.30)
Requirement already satisfied: toml in /opt/conda/lib/python3.11/site-packages (0.8.2)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.11/site-packages (from toml) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.11/site-packages (from beautifulsoup4->toml) (2.5)
Requirement already satisfied: fastjsonschema>=2.15 in /opt/conda/lib/python3.11/site-packages (2.19.1)
Requirement already satisfied: jsonschema>=2.6 in /opt/conda/lib/python3.11/site-packages (from nbformat) (2.19.1)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /opt/conda/lib/python3.11/site-packages (from jsonschema>=2.6->nbformat) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in /opt/conda/lib/python3.11/site-packages (from jsonschema>=2.6->nbformat) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /opt/conda/lib/python3.11/site-packages (from jsonschema>=2.6->nbformat) (0.18.0)
Requirement already satisfied: platformdirs>=2.5 in /opt/conda/lib/python3.11/site-packages (from jupyter-core!=5.0.*,>=4.12->nbformat) (4.2.1)
```


In [3]:
```
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 [5]:
```
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 Revenue columns), a dataframe with revenue data (dataframe must contain Date and Revenue columns), and the name of the stock.**

In [7]:
```
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 <= '2021-06-30']
    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(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,
    xaxis_rangeslider_visible=True)
    fig.show()
```

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.**

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 [9]:
```
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.

In [11]:
```
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.

In [10]:
```
tesla_data.reset_index(inplace=True)
tesla_data.head()
```

Out[10]:

	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	257626500	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`.

In [12]:
```
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` using parser i.e `html5lib` or `html.parser`.

In [13]:
```
soup = BeautifulSoup(html_data, 'html.parser')
```

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

- Click here if you need help locating the table

In [15]:
```
tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])
for row in soup.find("tbody").find_all("tr"):
    col = row.find_all("td")
    date = col[0].text
    revenue = col[1].text
    tesla_revenue = pd.concat([tesla_revenue, pd.DataFrame({"Date":[date], "Revenue":[revenue]})], ignore_index=True)
```

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

In [16]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"",regex=True)
```

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

In [17]:
```
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 [18]:
```
tesla_revenue.tail()
```

Out[18]:

	Date	Revenue
8	2013	$2,013
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 [19]:
```
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.

In [20]:
```
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.

In [21]:
```
gme_data.reset_index(inplace=True)
gme_data.head()
```

Out[21]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-02-13 00:00:00-05:00	1.620128	1.693350	1.603296	1.691666	76216000	0.0	0.0
1	2002-02-14 00:00:00-05:00	1.712707	1.716074	1.670626	1.683250	11021600	0.0	0.0
2	2002-02-15 00:00:00-05:00	1.683251	1.687459	1.658062	1.674834	8389600	0.0	0.0
3	2002-02-19 00:00:00-05:00	1.666418	1.666418	1.578047	1.607504	7414400	0.0	0.0
4	2002-02-20 00:00:00-05:00	1.615921	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_2`.

In [22]:
```
url2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html'
html_data_2 = requests.get(url2).text
```

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

In [23]:
```
soup2 = BeautifulSoup(html_data_2, 'html.parser')
```

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

In [24]:
```
gme_revenue = pd.DataFrame(columns=["Date", "Revenue"])
for row in soup2.find("tbody").find_all("tr"):
    col = row.find_all("td")
    date = col[0].text
    revenue = col[1].text
    gme_revenue = pd.concat([gme_revenue, pd.DataFrame({"Date":[date], "Revenue":[revenue]})], ignore_index=True)
```

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

In [25]:
```
gme_revenue.tail()
gme_revenue
```

Out[25]:

	Date	Revenue
0	2020	$6,466
1	2019	$8,285
2	2018	$8,547
3	2017	$7,965
4	2016	$9,364
5	2015	$9,296
6	2014	$9,040
7	2013	$8,887
8	2012	$9,551
9	2011	$9,474
10	2010	$9,078
11	2009	$8,806
12	2008	$7,094
13	2007	$5,319
14	2006	$3,092
15	2005	$1,843

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

In [27]:
```
make_graph(tesla_data, tesla_revenue, 'Tesla')
```


```
/tmp/ipykernel_2145/3316612228.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_2145/3316612228.py:6: 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.
```


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

In [28]:
```
gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',|\$',"",regex=True)
make_graph(gme_data, gme_revenue, 'GameStop')
```


```
/tmp/ipykernel_2145/3316612228.py:1: 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_2145/3316612228.py:6: 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.
```


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