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

In [ ]:
```
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 [ ]:
```
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.

In [ ]:
```
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 [ ]:
```
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles
    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, i
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date,
    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.**

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 [1]:
```
import yfinance as yf
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.

In [2]:
```
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.

In [4]:
```
tesla_data.reset_index(inplace=True)
```

In [5]:
```
tesla_data.head()
```

Out[5]:

	index	Date	Open	High	Low	Close	Volume	Dividend
0	0	2010-06-29 00:00:00-04:00	1.266667	1.666667	1.169333	1.592667	281494500	0.
1	1	2010-06-30 00:00:00-04:00	1.719333	2.028000	1.553333	1.588667	257806500	0.
2	2	2010-07-01 00:00:00-04:00	1.666667	1.728000	1.351333	1.464000	123282000	0.
3	3	2010-07-02 00:00:00-04:00	1.533333	1.540000	1.247333	1.280000	77097000	0.
4	4	2010-07-06 00:00:00-04:00	1.333333	1.333333	1.055333	1.074000	103003500	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 [13]:
```
import requests
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud
html_data = requests.get(url).text
```

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

```
In [14]: from bs4 import BeautifulSoup

         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

```
In [15]: !pip install lxml html5lib beautifulsoup4
```

```
Collecting lxml
  Downloading lxml-6.0.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_
x86_64.whl.metadata (6.6 kB)
Collecting html5lib
  Downloading html5lib-1.1-py2.py3-none-any.whl.metadata (16 kB)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.1
2/site-packages (4.12.3)
Requirement already satisfied: six>=1.9 in /opt/conda/lib/python3.12/site-
packages (from html5lib) (1.17.0)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.12/s
ite-packages (from html5lib) (0.5.1)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/
site-packages (from beautifulsoup4) (2.5)
Downloading lxml-6.0.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x8
6_64.whl (5.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.3/5.3 MB 94.2 MB/s eta 0:00:
00
Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
Installing collected packages: lxml, html5lib
Successfully installed html5lib-1.1 lxml-6.0.0
```

```
In [16]: import pandas as pd

         tables = pd.read_html(html_data)
         tesla_revenue = tables[1]  # this is usually the correct table
```

```
/tmp/ipykernel_865/3313150106.py:3: FutureWarning: Passing literal html to
'read_html' is deprecated and will be removed in a future version. To read
from a literal string, wrap it in a 'StringIO' object.
  tables = pd.read_html(html_data)
```

▶ Click here if you need help locating the table

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

```
In [18]: tesla_revenue.columns
```

```
Out[18]: Index(['Tesla Quarterly Revenue (Millions of US $)', 'Tesla Quarterly Re
venue (Millions of US $).1'], dtype='object')
```

```
In [22]: tesla_revenue.columns= ['Data','Revenue']
```

```
In [23]: tesla_revenue["Revenue"] = tesla_revenue["Revenue"].str.replace(r"\$|,",
```

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

```
In [24]: 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 [25]: tesla_revenue.tail()
```

Out[25]:

	Data	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`.

```
In [26]: import yfinance as yf

         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 [27]: 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 [28]: gme_data.reset_index(inplace=True)
```

```
In [29]: gme_data.head()
```

Out[29]:

	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	1.683250	11021600	0.0	0.0
2	2002-02-15 00:00:00-05:00	1.683250	1.687458	1.658001	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_2`.

In [30]:
```
import requests

url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud
html_data_2 = requests.get(url).text
```

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

In [31]:
```
from bs4 import BeautifulSoup

soup = 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 [32]:
```
import pandas as pd

tables = pd.read_html(html_data_2)
gme_revenue = tables[1]  # Usually the second table is GameStop revenue
```

```
/tmp/ipykernel_865/3209352396.py:3: FutureWarning: Passing literal html to
'read_html' is deprecated and will be removed in a future version. To read
from a literal string, wrap it in a 'StringIO' object.
  tables = pd.read_html(html_data_2)
```

In [33]:
```
gme_revenue.columns = ['Date', 'Revenue']
```

In [34]:
```
gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace(r"\$|,", "",
```

In [35]:
```
gme_revenue.dropna(inplace=True)
gme_revenue = gme_revenue[gme_revenue["Revenue"] != ""]
```

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

In [36]:
```
gme_revenue.tail()
```

Out[36]:

	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. Note the graph will only show data upto June 2021.

▶ Hint

In [39]:
```
import matplotlib.pyplot as plt

def make_graph(stock_data, revenue_data, stock):
    fig, ax1 = plt.subplots(figsize=(14, 6))

    # Plot stock data (Stock price over time)
    ax1.plot(stock_data['Date'], stock_data['Close'], 'b-', label="Stock
    ax1.set_xlabel("Date")
    ax1.set_ylabel("Stock Price ($)", color="b")
    ax1.tick_params(axis='y', labelcolor="b")

    # Second Y axis for revenue
    ax2 = ax1.twinx()
    ax2.plot(revenue_data['Date'], revenue_data['Revenue'].astype(float),
    ax2.set_ylabel("Revenue ($ Millions)", color="r")
    ax2.tick_params(axis='y', labelcolor="r")

    plt.title(f"{stock} Stock Price and Revenue")
    fig.tight_layout()
    plt.show()
```

In [42]:
```
tesla_revenue.columns
```

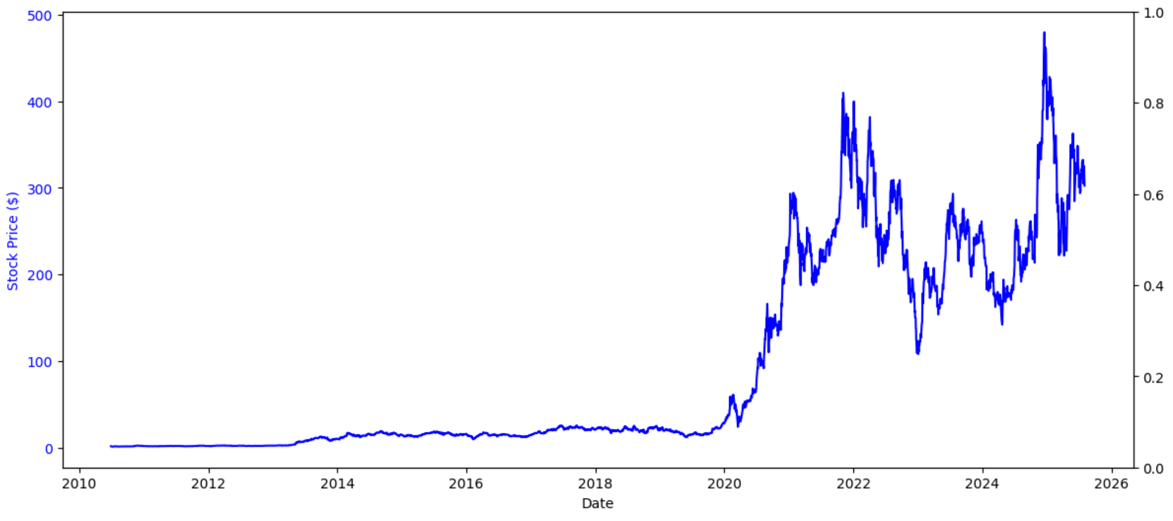
Out[42]:
```
Index(['Data', 'Revenue'], dtype='object')
```

In [43]:
```
tesla_revenue.columns = ['Date', 'Revenue']
```

In [44]:
```
make_graph(tesla_data, tesla_revenue, 'Tesla')
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
Cell In[44], line 1
----> 1 make_graph(tesla_data, tesla_revenue, 'Tesla')

Cell In[39], line 14, in make_graph(stock_data, revenue_data, stock)
     12 # Second Y axis for revenue
     13 ax2 = ax1.twinx()
---> 14 ax2.plot(revenue_data['Date'], revenue_data['Revenue'].astype(floa
t), 'r-', label="Revenue")
     15 ax2.set_ylabel("Revenue ($ Millions)", color="r")
     16 ax2.tick_params(axis='y', labelcolor="r")

File /opt/conda/lib/python3.12/site-packages/matplotlib/axes/_axes.py:177
7, in Axes.plot(self, scalex, scaley, data, *args, **kwargs)
   1534 """
   1535 Plot y versus x as lines and/or markers.
   1536
   (...)
   1774 (``'green'``) or hex strings (``'#008000'``).
   1775 """
   1776 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1777 lines = [*self._get_lines(self, *args, data=data, **kwargs)]
   1778 for line in lines:
   1779     self.add_line(line)

File /opt/conda/lib/python3.12/site-packages/matplotlib/axes/_base.py:297,
in _process_plot_var_args.__call__(self, axes, data, return_kwargs, *args,
**kwargs)
    295     this += args[0],
    296     args = args[1:]
--> 297 yield from self._plot_args(
    298     axes, this, kwargs, ambiguous_fmt_datakey=ambiguous_fmt_datake
y,
    299     return_kwargs=return_kwargs
    300 )

File /opt/conda/lib/python3.12/site-packages/matplotlib/axes/_base.py:489,
in _process_plot_var_args._plot_args(self, axes, tup, kwargs, return_kwarg
s, ambiguous_fmt_datakey)
    486     x, y = index_of(xy[-1])
    488 if axes.xaxis is not None:
--> 489     axes.xaxis.update_units(x)
    490 if axes.yaxis is not None:
    491     axes.yaxis.update_units(y)

File /opt/conda/lib/python3.12/site-packages/matplotlib/axis.py:1754, in A
xis.update_units(self, data)
   1752 neednew = self._converter != converter
   1753 self._set_converter(converter)
-> 1754 default = self._converter.default_units(data, self)
   1755 if default is not None and self.units is None:
   1756     self.set_units(default)

File /opt/conda/lib/python3.12/site-packages/matplotlib/category.py:108, i
n StrCategoryConverter.default_units(data, axis)
    106     axis.set_units(UnitData(data))
    107 else:
```

```
--> 108     axis.units.update(data)
    109 return axis.units

AttributeError: 'America/New_York' object has no attribute 'update'
```



Question 6: Plot GameStop Stock Graph

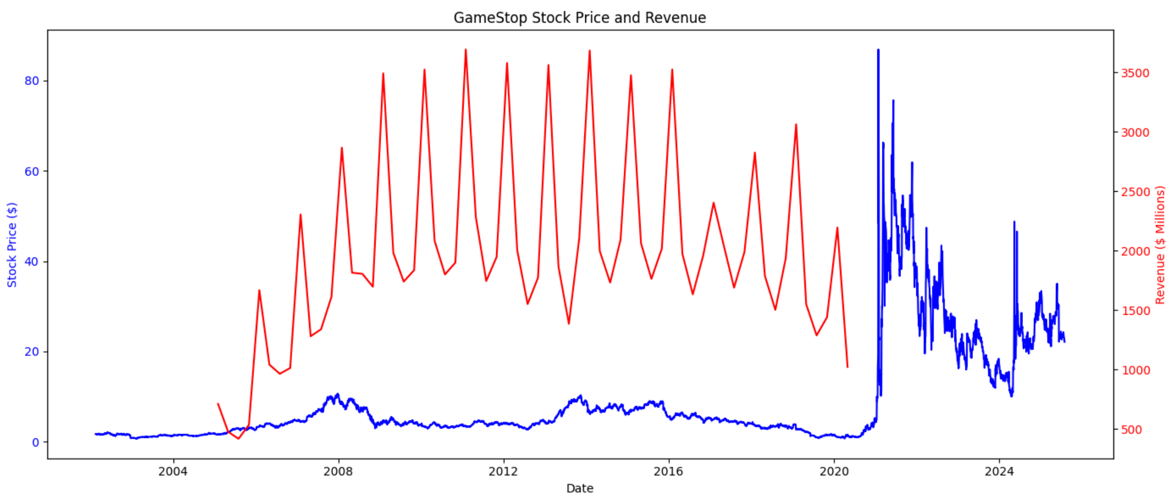
Use the `make_graph` function to graph the GameStop Stock Data, also provide a title for the graph. The structure to call the `make_graph` function is `make_graph(gme_data, gme_revenue, 'GameStop')`. Note the graph will only show data upto June 2021.

▶ Hint

In [47]:
```
# Fix for tesla_data
tesla_data['Date'] = pd.to_datetime(tesla_data['Date']).dt.tz_localize(No

# Fix for gme_data
gme_data['Date'] = pd.to_datetime(gme_data['Date']).dt.tz_localize(None)
```

In [48]:
```
# For tesla_revenue
tesla_revenue['Date'] = pd.to_datetime(tesla_revenue['Date'])

# For gme_revenue
gme_revenue['Date'] = pd.to_datetime(gme_revenue['Date'])
```

In [49]:
```
make_graph(gme_data, gme_revenue, 'GameStop')
```



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