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 bs4
!pip install nbformat
!pip install --upgrade plotly
```

```
Collecting yfinance
  Downloading yfinance-0.2.64-py2.py3-none-any.whl.metadata (5.8 kB)
Collecting pandas>=1.3.0 (from yfinance)
  Downloading pandas-2.3.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (91 kB)
Collecting numpy>=1.16.5 (from yfinance)
  Downloading numpy-2.3.1-cp312-cp312-manylinux_2_28_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-3.18.1.tar.gz (3.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.0/3.0 MB 114.3 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)
Collecting curl_cffi>=0.7 (from yfinance)
  Downloading curl_cffi-0.11.4-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (14 kB)
Collecting protobuf>=3.19.0 (from yfinance)
  Downloading protobuf-6.31.1-cp39-abi3-manylinux2014_x86_64.whl.metadata (593 bytes)
Collecting websockets>=13.0 (from yfinance)
  Downloading websockets-15.0.1-cp312-cp312-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.8 kB)
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: cffi>=1.12.0 in /opt/conda/lib/python3.12/site-packages (from curl_cffi>=0.7->yfinance) (1.17.1)
Requirement already satisfied: certifi>=2024.2.2 in /opt/conda/lib/python3.12/site-packages (from curl_cffi>=0.7->yfinance) (2024.12.14)
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-2025.2-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: pycparser in /opt/conda/lib/python3.12/site-packages (from cffi>=1.12.0->curl_cffi>=0.7->yfinance) (2.22)
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-0.2.64-py2.py3-none-any.whl (119 kB)
Downloading curl_cffi-0.11.4-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.5/8.5 MB 165.9 MB/s eta 0:00:00
Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Downloading numpy-2.3.1-cp312-cp312-manylinux_2_28_x86_64.whl (16.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.6/16.6 MB 183.2 MB/s eta 0:00:00
Downloading pandas-2.3.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.0/12.0 MB 175.9 MB/s eta 0:00:00
Downloading protobuf-6.31.1-cp39-abi3-manylinux2014_x86_64.whl (321 kB)
Downloading websockets-15.0.1-cp312-cp312-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (182 kB)
Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Building wheels for collected packages: peewee
  Building wheel for peewee (pyproject.toml) ... one
  Created wheel for peewee: filename=peewee-3.18.1-cp312-cp312-linux_x86_64.whl size=303801 sha256=bbbb6f028e710710416f5e623f49f22b162a188e4bfea0c6610f87cc2b6
```

```
56606
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/1a/57/6a/bb71346381d0d911cd4ce3026f1fa720da76707e4f01cf27dd
Successfully built peewee
Installing collected packages: peewee, multitasking, websockets, tzdata, protobuf, numpy, pandas, curl_cffi, yfinance
Successfully installed curl_cffi-0.11.4 multitasking-0.0.11 numpy-2.3.1 pandas-2.3.0 peewee-3.18.1 protobuf-6.31.1 tzdata-2025.2 websockets-15.0.1 yfinance-0.
2.64
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) (4.1
2.2)
Requirement already satisfied: plotly in /opt/conda/lib/python3.12/site-packages (5.24.1)
Collecting plotly
  Downloading plotly-6.2.0-py3-none-any.whl.metadata (8.5 kB)
Collecting narwhals>=1.15.1 (from plotly)
  Downloading narwhals-1.45.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: packaging in /opt/conda/lib/python3.12/site-packages (from plotly) (24.2)
Downloading plotly-6.2.0-py3-none-any.whl (9.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.6/9.6 MB 163.0 MB/s eta 0:00:00
Downloading narwhals-1.45.0-py3-none-any.whl (371 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-1.45.0 plotly-6.2.0
```

In [15]:

```
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 [16]:
```
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 [17]:
```
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 [18]:
```
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
    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=
    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 [19]:
```
# use of ticker function for extrating data
tsle = 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 [20]:
```
#use of history and period

tesla_data = tsle.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 [21]:
```
# use of reset index

tesla_data.reset_index(inplace=True)

tesla_data.head()  #view the data
```

Out[21]:

	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	257806500	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

In [22]:
```
tesla_data.tail()
```

Out[22]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
3772	2025-06-27 00:00:00-04:00	324.510010	329.339996	317.500000	323.630005	89067000	0.0	0.0
3773	2025-06-30 00:00:00-04:00	319.899994	325.579987	316.600006	317.660004	76695100	0.0	0.0
3774	2025-07-01 00:00:00-04:00	298.459991	305.890015	293.209991	300.709991	145085700	0.0	0.0
3775	2025-07-02 00:00:00-04:00	312.630005	316.829987	303.820007	315.649994	119483700	0.0	0.0
3776	2025-07-03 00:00:00-04:00	317.989990	318.450012	312.760010	315.350006	58042300	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 [38]:
```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"

response = requests.get(url)
html_data = response.text
html_data[:500]
```

Out[38]:
```
'\n<!DOCTYPE html>\n<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->\n<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8">
<![endif]-->\n<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->\n<!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->\n    <head>\n
<meta charset="utf-8">\n        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">\n\t\t<link rel="canonical" href="https://www.macrotrends.net/s
tocks/charts/TSLA/tesla/revenue" />\n\t'
```

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

In [40]:
```
soup = BeautifulSoup(html_data,"html.parser")
soup.title.text
```

Out[40]:
```
'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`.

▶ Step-by-step instructions

▶ Click here if you need help locating the table

In [45]:
```
#create an empty Data frame for store data
tesla_revenue = pd.DataFrame(columns=["Date","Revenue"])

#viwe the created dataframe
tesla_revenue

tables = soup.find_all("table")

for table in tables:
    if "Tesla Quarterly Revenue" in table.text:
        target_table = table
        break

for row in target_table.find_all("tr")[1:]:
    cols = row.find_all("td")
```

```
    if len(cols)==2:
        date = cols[0].text
        revenue = cols[1].text
        tesla_revenue = pd.concat([tesla_revanue,pd.DataFrame([{"Date": date,"Revenue": revenue}])],ignore_index=True)

tesla_revenue.head()
```

Out[45]:

	Date	Revenue
0	2022-09-30	$21,454
1	2022-06-30	$16,934
2	2022-03-31	$18,756
3	2021-12-31	$17,719
4	2021-09-30	$13,757

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

In [46]:
```
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 [47]:
```
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 [48]:
```
tesla_revenue.tail()
```

Out[48]:

	Date	Revenue
49	2010-06-30	28
50	2010-03-31	21
52	2009-09-30	46
53	2009-06-30	27
54	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 [50]:
```
gme = yf.Ticker("GME")
gme
```

Out[50]:
```
yfinance.Ticker object <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 [52]:
```
gme_data = gme.history(period ="max")
gme_data
```

Out[52]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2002-02-13 00:00:00-05:00	1.620128	1.693350	1.603296	1.691667	76216000	0.0	0.0
2002-02-14 00:00:00-05:00	1.712707	1.716074	1.670626	1.683250	11021600	0.0	0.0
2002-02-15 00:00:00-05:00	1.683250	1.687458	1.658001	1.674834	8389600	0.0	0.0
2002-02-19 00:00:00-05:00	1.666418	1.666418	1.578047	1.607504	7410400	0.0	0.0
2002-02-20 00:00:00-05:00	1.615921	1.662210	1.603296	1.662210	6892800	0.0	0.0
...	...	...	...	...	...	...	...
2025-06-27 00:00:00-04:00	23.990000	24.260000	23.459999	23.590000	11638200	0.0	0.0
2025-06-30 00:00:00-04:00	23.639999	24.400000	23.540001	24.389999	10439300	0.0	0.0
2025-07-01 00:00:00-04:00	24.150000	24.500000	23.680000	23.680000	8308000	0.0	0.0
2025-07-02 00:00:00-04:00	23.900000	24.100000	23.750000	23.950001	6428600	0.0	0.0
2025-07-03 00:00:00-04:00	23.850000	24.160000	23.490000	23.590000	5566300	0.0	0.0

5885 rows × 7 columns

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 [73]:
```
.reset_index(drop=True, inplace=True) # here i add drop becuse this is showng lever_0 and index
gme_data.head()
```

Out[73]:

	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	1.615921	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 [78]:
```
url_2 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"

response_2 = requests.get(url)
html_data_2 = response_2.text
html_data_2[:500]
```

Out[78]:
```
'<!DOCTYPE html>\n<!-- saved from url=(0105)https://web.archive.org/web/20200814131437/https://www.macrotrends.net/stocks/charts/GME/gamestop/revenue -->\n<h
tml class=" js flexbox canvas canvastext webgl no-touch geolocation postmessage websqldatabase indexeddb hashchange history draganddrop websockets rgba hsla
multiplebgs backgroundsize borderimage borderradius boxshadow textshadow opacity cssanimations csscolumns cssgradients cssreflections csstransforms csstransf
orms3d csstransitions fontface g'
```

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

In [82]:
```
soup_2 = BeautifulSoup(html_data_2,"html.parser")
soup_2.title.text
```

Out[82]:
```
'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.

> **Note: Use the method similar to what you did in question 2.**

▶ Click here if you need help locating the table

In [96]:
```
# create a dataframe date and revenue
```

```
gme_revenue = pd.DataFrame(columns = ["Date","Revenue"])
gme_revenue
```

Out[96]:

Date	Revenue

In [98]:

```
tables_2 = soup_2.find_all("table")

for table_2 in tables_2:
    if "GameStop Quarterly Revenue" in table_2.text:
        target_table_gme = table_2
        break

for row in target_table_gme.find_all("tr")[1:]:
    cols = row.find_all("td")
    if len(cols)==2:
        date_2 = cols[0].text
        revenue_2 = cols[1].text
        gme_revenue = pd.concat([gme_revenue, pd.DataFrame([{"Date":date_2,"Revenue":revenue_2}])], ignore_index = True)

gme_revenue['Revenue'] = gme_revenue['Revenue'].str.replace(',|\$',"",regex=True)

tesla_revenue.dropna(inplace=True)

tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
```

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

In [99]:

```
gme_revenue.tail()
```

Out[99]:

	Date	Revenue
119	2006-01-31	1667
120	2005-10-31	534
121	2005-07-31	416
122	2005-04-30	475
123	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 [101…
```
make_graph(tesla_data, tesla_revenue, 'Tesla')
```

```
/tmp/ipykernel_300/109047474.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.pyda
ta.org/pdeps/0004-consistent-to-datetime-parsing.html. You can safely remove this argument.

/tmp/ipykernel_300/109047474.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.pyda
ta.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 [102…

```
make_graph(gme_data, gme_revenue, 'GameStop')
```

```
/tmp/ipykernel_300/109047474.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.pyda
ta.org/pdeps/0004-consistent-to-datetime-parsing.html. You can safely remove this argument.

/tmp/ipykernel_300/109047474.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.pyda
ta.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
