```
!pip install yfinance==0.2.38
!pip install pandas==2.2.2
!pip install nbformat

Requirement already satisfied: yfinance==0.2.38 in c:\users\supri\
anaconda3\lib\site-packages (0.2.38)
Requirement already satisfied: pandas>=1.3.0 in c:\users\supri\
anaconda3\lib\site-packages (from yfinance==0.2.38) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in c:\users\supri\
anaconda3\lib\site-packages (from yfinance==0.2.38) (1.26.4)
Requirement already satisfied: requests>=2.31 in c:\users\supri\
anaconda3\lib\site-packages (from yfinance==0.2.38) (2.32.2)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\supri\
anaconda3\lib\site-packages (from yfinance==0.2.38) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in c:\users\supri\
anaconda3\lib\site-packages (from yfinance==0.2.38) (5.2.1)
Requirement already satisfied: appdirs>=1.4.4 in c:\users\supri\
anaconda3\lib\site-packages (from yfinance==0.2.38) (1.4.4)
Requirement already satisfied: pytz>=2022.5 in c:\users\supri\
anaconda3\lib\site-packages (from yfinance==0.2.38) (2024.1)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\supri\
anaconda3\lib\site-packages (from yfinance==0.2.38) (2.4.2)
Requirement already satisfied: peewee>=3.16.2 in c:\users\supri\
anaconda3\lib\site-packages (from yfinance==0.2.38) (3.17.6)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\
supri\anaconda3\lib\site-packages (from yfinance==0.2.38) (4.12.3)
Requirement already satisfied: html5lib>=1.1 in c:\users\supri\
anaconda3\lib\site-packages (from yfinance==0.2.38) (1.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\supri\
anaconda3\lib\site-packages (from beautifulsoup4>=4.11.1-
>yfinance==0.2.38) (2.5)
Requirement already satisfied: six>=1.9 in c:\users\supri\anaconda3\
lib\site-packages (from html5lib>=1.1->yfinance==0.2.38) (1.16.0)
Requirement already satisfied: webencodings in c:\users\supri\
anaconda3\lib\site-packages (from html5lib>=1.1->yfinance==0.2.38)
(0.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
supri\anaconda3\lib\site-packages (from pandas>=1.3.0-
>yfinance==0.2.38) (2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in c:\users\supri\
anaconda3\lib\site-packages (from pandas>=1.3.0->yfinance==0.2.38)
(2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
supri\anaconda3\lib\site-packages (from requests>=2.31-
>yfinance==0.2.38) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\supri\
anaconda3\lib\site-packages (from requests>=2.31->yfinance==0.2.38)
(3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\supri\
anaconda3\lib\site-packages (from requests>=2.31->yfinance==0.2.38)
```

```
(2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\supri\
anaconda3\lib\site-packages (from requests>=2.31->yfinance==0.2.38)
(2024.6.2)
Requirement already satisfied: pandas==2.2.2 in c:\users\supri\
anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in c:\users\supri\
anaconda3\lib\site-packages (from pandas==2.2.2) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
supri\anaconda3\lib\site-packages (from pandas==2.2.2) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\supri\
anaconda3\lib\site-packages (from pandas==2.2.2) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\supri\
anaconda3\lib\site-packages (from pandas==2.2.2) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\supri\anaconda3\
lib\site-packages (from python-dateutil>=2.8.2->pandas==2.2.2)
(1.16.0)
Requirement already satisfied: nbformat in c:\users\supri\anaconda3\
lib\site-packages (5.9.2)
Requirement already satisfied: fastjsonschema in c:\users\supri\
anaconda3\lib\site-packages (from nbformat) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in c:\users\supri\
anaconda3\lib\site-packages (from nbformat) (4.19.2)
Requirement already satisfied: jupyter-core in c:\users\supri\
anaconda3\lib\site-packages (from nbformat) (5.7.2)
Requirement already satisfied: traitlets>=5.1 in c:\users\supri\
anaconda3\lib\site-packages (from nbformat) (5.14.3)
Requirement already satisfied: attrs>=22.2.0 in c:\users\supri\
anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
c:\users\supri\anaconda3\lib\site-packages (from jsonschema>=2.6-
>nbformat) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in c:\users\supri\
anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in c:\users\supri\
anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat) (0.10.6)
Requirement already satisfied: platformdirs>=2.5 in c:\users\supri\
anaconda3\lib\site-packages (from jupyter-core->nbformat) (3.10.0)
Requirement already satisfied: pywin32>=300 in c:\users\supri\
anaconda3\lib\site-packages (from jupyter-core->nbformat) (305.1)
```

```
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
```

Ignoring warnings using the warnings module.Using the filterwarnings function to filter or ignore specific warning messages or categories.

```
import warnings
# Ignore all warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

Define Graphing Function

In this section, we're defining the function `make_graph`. It takes a dataframe with stock data (dataframe containing Date and Close columns), a dataframe with revenue data (dataframe containing Date and Revenue columns), and the name of the stock.

```
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=0.3)

    # Filtering data up to the specific dates
    stock_data_specific = stock_data[stock_data['Date'] <= '2021-06-
14']
    revenue_data_specific = revenue_data[revenue_data['Date'] <=
'2021-04-30']

    # Plotting Share Price on the first row
    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)

    # Plotting Revenue on the second row
    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",
        mode='lines+markers'
    ), row=2, col=1)

    # Updating x-axis labels for both rows
    fig.update_xaxes(title_text="Date", row=1, col=1)
    fig.update_xaxes(title_text="Date", row=2, col=1)

    # Updating y-axis labels for both rows
    fig.update_yaxes(title_text="Price ($US)", row=1, col=1)
    fig.update_yaxes(title_text="Revenue ($US Millions)", row=2,
col=1)

    # Updating layout for the figure
```

```
    fig.update_layout(showlegend=False,
                      height=900,
                      title=stock,
                      xaxis_rangeslider_visible=False)

    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`.

```
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.

```
tesla_history=tesla.history(period="max")
tesla_data=pd.DataFrame(tesla_history)
tesla_data
```

```
                                 Open        High         Low
Close  \
Date

2010-06-29 00:00:00-04:00    1.266667    1.666667    1.169333
1.592667
2010-06-30 00:00:00-04:00    1.719333    2.028000    1.553333
1.588667
2010-07-01 00:00:00-04:00    1.666667    1.728000    1.351333
1.464000
2010-07-02 00:00:00-04:00    1.533333    1.540000    1.247333
1.280000
2010-07-06 00:00:00-04:00    1.333333    1.333333    1.055333
1.074000
...                               ...         ...         ...
...
2024-09-23 00:00:00-04:00  242.610001  250.000000  241.919998
250.000000
2024-09-24 00:00:00-04:00  254.080002  257.190002  249.050003
254.270004
2024-09-25 00:00:00-04:00  252.539993  257.049988  252.279999
257.019989
2024-09-26 00:00:00-04:00  260.600006  261.750000  251.529999
254.220001
2024-09-27 00:00:00-04:00  257.380005  260.700012  254.119995
260.459991
```

```
                                  Volume  Dividends  Stock Splits
Date
2010-06-29 00:00:00-04:00  281494500        0.0           0.0
2010-06-30 00:00:00-04:00  257806500        0.0           0.0
2010-07-01 00:00:00-04:00  123282000        0.0           0.0
2010-07-02 00:00:00-04:00   77097000        0.0           0.0
2010-07-06 00:00:00-04:00  103003500        0.0           0.0
...                              ...        ...           ...
2024-09-23 00:00:00-04:00   86927200        0.0           0.0
2024-09-24 00:00:00-04:00   88491000        0.0           0.0
2024-09-25 00:00:00-04:00   65034300        0.0           0.0
2024-09-26 00:00:00-04:00   67142200        0.0           0.0
2024-09-27 00:00:00-04:00   70729000        0.0           0.0

[3587 rows x 7 columns]
```

Resetting 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.

```
tesla_data.reset_index(inplace=True)
tesla_data
```

```
                       Date        Open        High         Low  \
0    2010-06-29 00:00:00-04:00    1.266667    1.666667    1.169333
1    2010-06-30 00:00:00-04:00    1.719333    2.028000    1.553333
2    2010-07-01 00:00:00-04:00    1.666667    1.728000    1.351333
3    2010-07-02 00:00:00-04:00    1.533333    1.540000    1.247333
4    2010-07-06 00:00:00-04:00    1.333333    1.333333    1.055333
...                        ...         ...         ...         ...
3582 2024-09-23 00:00:00-04:00  242.610001  250.000000  241.919998
3583 2024-09-24 00:00:00-04:00  254.080002  257.190002  249.050003
3584 2024-09-25 00:00:00-04:00  252.539993  257.049988  252.279999
3585 2024-09-26 00:00:00-04:00  260.600006  261.750000  251.529999
3586 2024-09-27 00:00:00-04:00  257.380005  260.700012  254.119995

           Close     Volume  Dividends  Stock Splits
0       1.592667  281494500        0.0           0.0
1       1.588667  257806500        0.0           0.0
2       1.464000  123282000        0.0           0.0
3       1.280000   77097000        0.0           0.0
4       1.074000  103003500        0.0           0.0
...          ...        ...        ...           ...
3582  250.000000   86927200        0.0           0.0
3583  254.270004   88491000        0.0           0.0
3584  257.019989   65034300        0.0           0.0
3585  254.220001   67142200        0.0           0.0
```

```
3586  260.459991   70729000        0.0           0.0

[3587 rows x 8 columns]
```

Question 2: Use Webscraping to Extract Tesla Revenue Data

Using 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 Saving the text of the response as a variable named `html_data`.

```
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
```

Parse the html data using `beautiful_soup`.

```
soup = BeautifulSoup(html_data, 'html.parser')
tables = soup.find_all('table')
tesla_revenue = pd.read_html(str(tables))[0]
tesla_revenue.columns = ['Date', 'Revenue']
print(tesla_revenue.head())
```

```
   Date  Revenue
0  2021  $53,823
1  2020  $31,536
2  2019  $24,578
3  2018  $21,461
4  2017  $11,759
```

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`.

```
read_html_pandas_data = pd.read_html(url)
```

Removing the comma and dollar sign from the `Revenue` column and convert into float

```
tesla_revenue['Revenue'] = tesla_revenue['Revenue'].str.replace(',', '')  # Remove commas
tesla_revenue['Revenue'] = tesla_revenue['Revenue'].str.replace('$', '')  # Remove dollar signs
tesla_revenue['Revenue'] = pd.to_numeric(tesla_revenue['Revenue'], errors='coerce')  # Convert to float
```

Removing an null or empty strings in the Revenue column.

```
tesla_revenue.dropna(inplace=True)

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

Displaying the last 5 row of the `tesla_revenue` dataframe using the `tail` function. Take a screenshot of the results.

```
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`.

```
game_stop=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.

```
history_game_stop=game_stop.history(period="max")
gme_data=pd.DataFrame(history_game_stop)
```

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.

```
gme_data.reset_index(inplace=True)
gme_data.head()
```

```
                       Date      Open      High       Low     Close
Volume  \
0 2002-02-13 00:00:00-05:00  1.620129  1.693350  1.603296  1.691667
76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250
11021600
2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658001  1.674834
8389600
3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504
```

```
7410400
4 2002-02-20 00:00:00-05:00  1.615920  1.662210  1.603296  1.662210
6892800

   Dividends  Stock Splits
0        0.0           0.0
1        0.0           0.0
2        0.0           0.0
3        0.0           0.0
4        0.0           0.0
```

Question 4: Use Webscraping to Extract GME Revenue Data

Using 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`.

```
URL="https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-
SkillsNetwork/labs/project/stock.html"
html_data=requests.get(URL).text
```

Parse the html data using `beautiful_soup`.

```
soup=BeautifulSoup(html_data,'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`.

```
tables=soup.find_all("table")
gme_revenue=pd.read_html(str(tables))[0]
gme_revenue.columns=["Date","Revenue"]
gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',|\\
$',"",regex=True)
gme_revenue.dropna(inplace=True)

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

Displaying the last five rows of the `gme_revenue` dataframe using the `tail` function.

```
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.

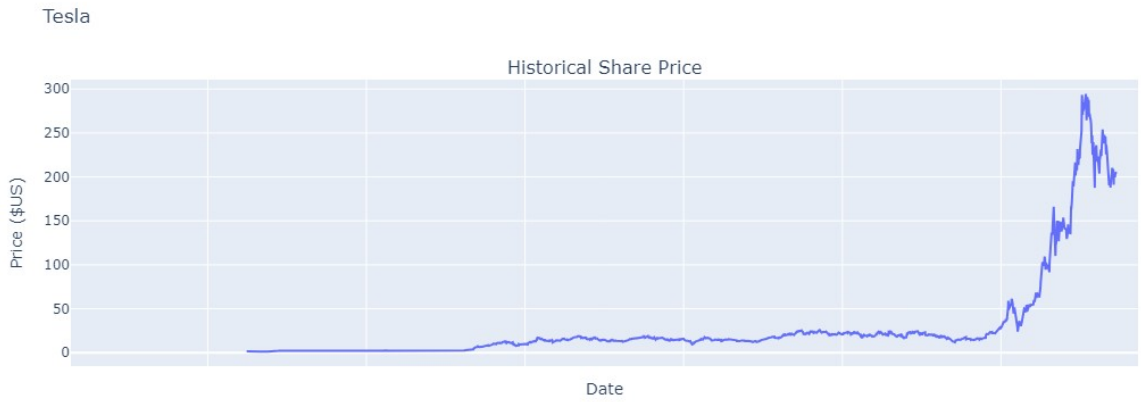
```
tesla_revenue['Date'] = pd.to_datetime(tesla_revenue['Date'],
format='%Y')
make_graph(tesla_data, tesla_revenue, 'Tesla')
```

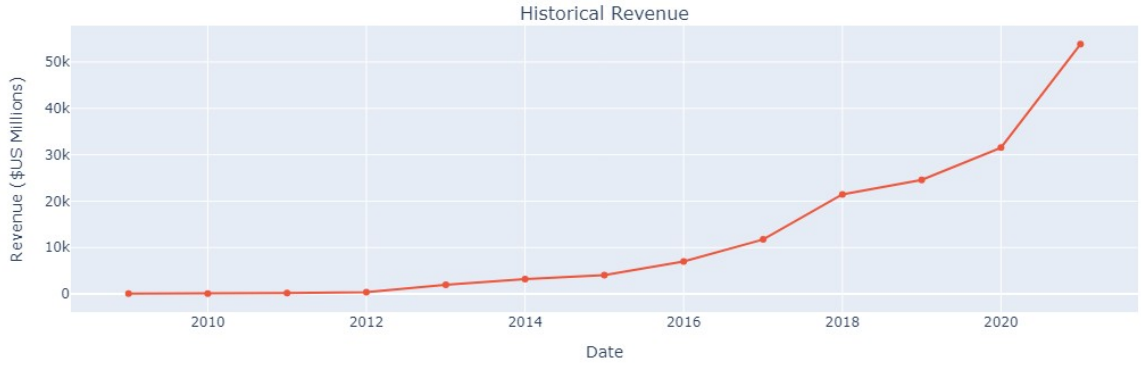
```
C:\Users\supri\AppData\Local\Temp\ipykernel_21472\4012064312.py:12:
UserWarning:

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.

C:\Users\supri\AppData\Local\Temp\ipykernel_21472\4012064312.py:19:
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.

```
gme_revenue['Date'] = pd.to_datetime(gme_revenue['Date'], format='%Y')
make_graph(gme_data, gme_revenue, 'GameStop')
```

```
C:\Users\supri\AppData\Local\Temp\ipykernel_21472\4012064312.py:12:
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.

C:\Users\supri\AppData\Local\Temp\ipykernel_21472\4012064312.py:19:
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.
```