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`.

[7]:
```
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.

[8]:
```
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.

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

[10]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2010-06-29	1.266667	1.666667	1.169333	1.592667	281494500	0	0.0
1	2010-06-30	1.719333	2.028000	1.553333	1.588667	257806500	0	0.0
2	2010-07-01	1.666667	1.728000	1.351333	1.464000	123282000	0	0.0
3	2010-07-02	1.533333	1.540000	1.247333	1.280000	77097000	0	0.0
4	2010-07-06	1.333333	1.333333	1.055333	1.074000	103003500	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`.

[45]:
```
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue
html_data = requests.get(url).text
```

Parse the html data using `beautiful_soup`.

[87]:
```
beautiful_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`.

▶ Click here if you need help locating the table

[144]:
```
tables = beautiful_soup.find_all('tbody')[1]
tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])
for row in tables.find_all("tr"):
    col = row.find_all("td")
    date = col[0].text
    revenue = col[1].text
    tesla_revenue = tesla_revenue.append({"Date":date, "Revenue":revenue}, ignore_index=True)
```

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

[145]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

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

[146]:
```
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.

[153]:
```
tesla_revenue.tail()
```

[153]:

	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

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`.

[41]:
```
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.

[42]:
```
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.

[44]:
```
gme_data.reset_index(inplace=True)
gme_data.head()
```

[44]:

	index	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	0	2002-02-13	1.620129	1.693350	1.603296	1.691667	76216000	0.0	0.0
1	1	2002-02-14	1.712707	1.716074	1.670626	1.683251	11021600	0.0	0.0
2	2	2002-02-15	1.683250	1.687458	1.658001	1.674834	8389600	0.0	0.0
3	3	2002-02-19	1.666418	1.666418	1.578047	1.607504	7410400	0.0	0.0
4	4	2002-02-20	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`.
```

[148]:
```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.h
html_data = requests.get(url).text
```

Parse the html data using `beautiful_soup`.

[149]:
```
beautiful_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`. Make sure the comma and dollar sign is removed from the `Revenue` column using a method similar to what you did in Question 2.

▶ Click here if you need help locating the table

[150]:
```
tables = beautiful_soup.find_all('tbody')[1]
gme_revenue = pd.DataFrame(columns=["Date", "Revenue"])
for row in tables.find_all("tr"):
    col = row.find_all("td")
    date = col[0].text
    revenue = col[1].text
    gme_revenue = gme_revenue.append({"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.

[151]:
```
gme_revenue.tail()
```

[151]:

	Date	Revenue
57	2006-01-31	$1,667
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. 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.

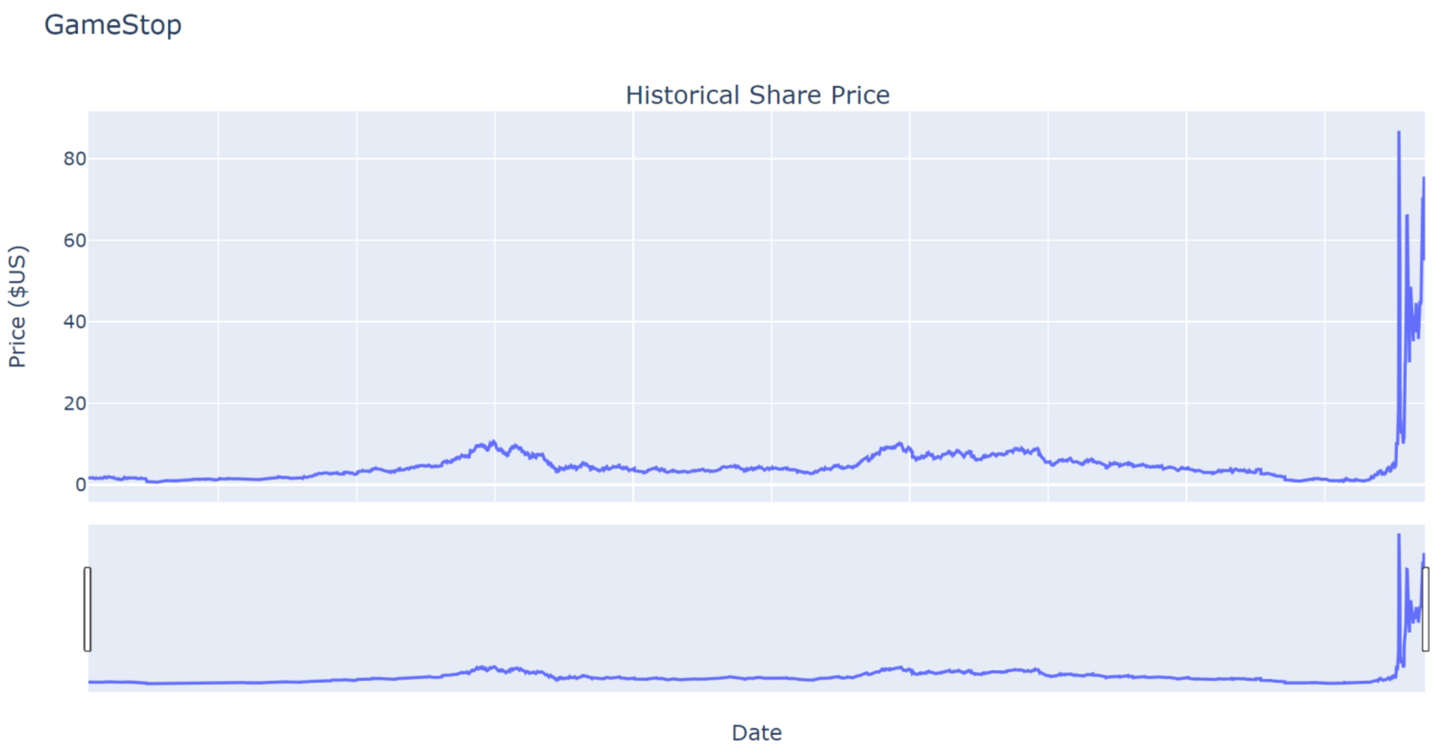
```
[167]: make_graph(tesla_data, tesla_revenue, 'Tesla')
```

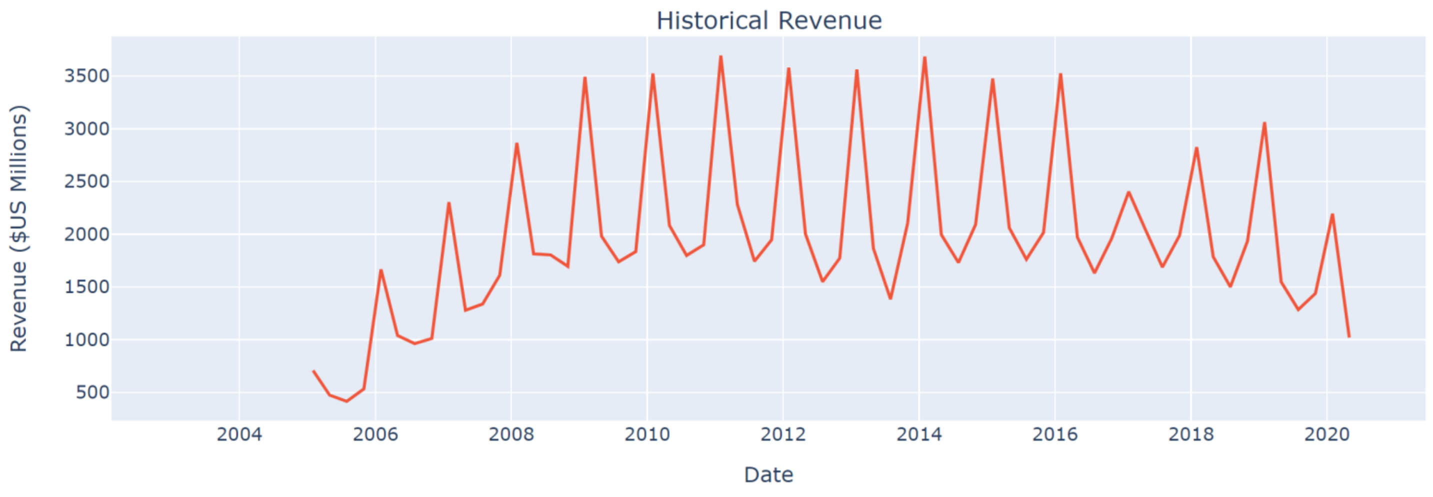
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_681/851608974.py in <module>
----> 1 make_graph(tesla_data, tesla_revenue, 'Tesla')

/tmp/ipykernel_681/2068038883.py in make_graph(stock_data, revenue_data, stock)
      3     stock_data_specific = stock_data[stock_data.Date <= '2021--06-14']
      4     revenue_data_specific = revenue_data[revenue_data.Date <= '2021-04-30']
----> 5     fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific.Date, infer_datetime_format=True), y=stock_data_specific.Close.astyp
e("float"), name="Share Price"), row=1, col=1)
      6     fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date, infer_datetime_format=True), y=revenue_data_specific.Revenu
e.astype("float"), name="Revenue"), row=2, col=1)
      7     fig.update_xaxes(title_text="Date", row=1, col=1)

~/conda/envs/python/lib/python3.7/site-packages/pandas/core/generic.py in __getattr__(self, name)
   5485         ):
   5486             return self[name]
-> 5487         return object.__getattribute__(self, name)
   5488 
   5489     def __setattr__(self, name: str, value) -> None:

AttributeError: 'DataFrame' object has no attribute 'Close'
```

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.

```
168]: gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',|~~\$',"")~~
      make_graph(gme_data, gme_revenue, 'GameStop')
```




Historical Revenue
Revenue ($US Millions)
500
1000
1500
2000
2500
3000
3500
2004
2006
2008
2010
2012
2014
2016
2018
2020
Date