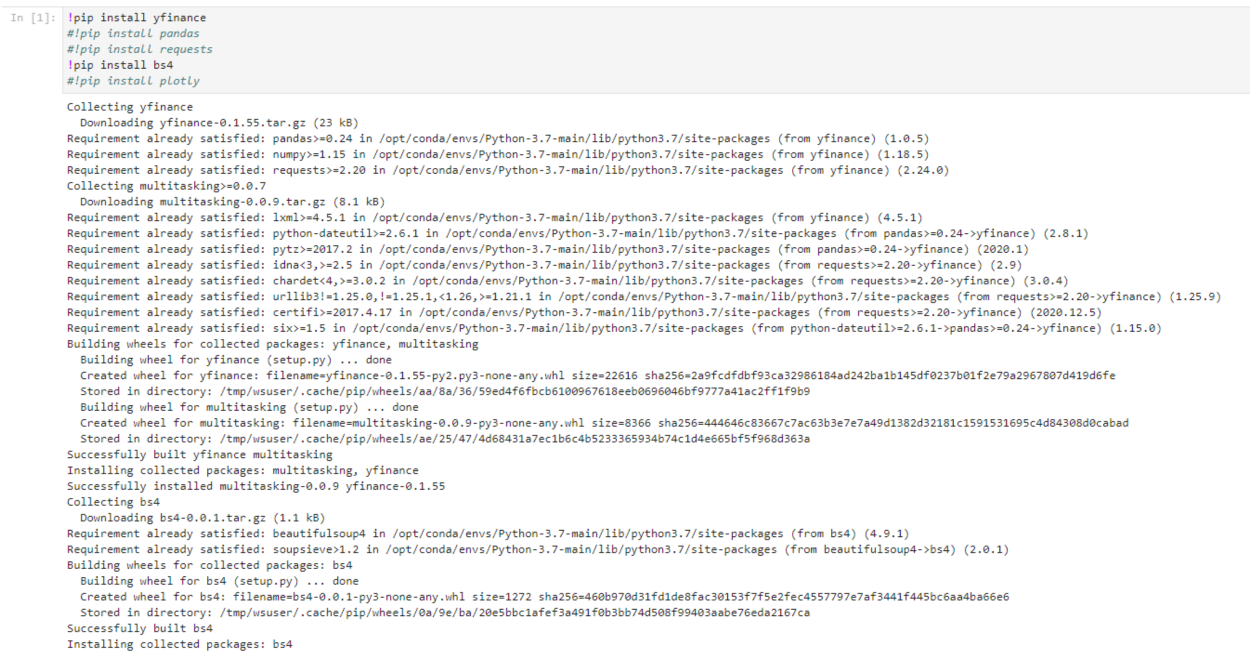
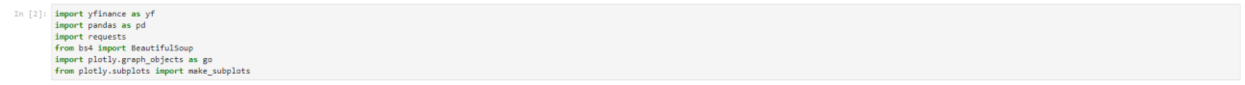
In [1]:



```
In [1]: !pip install yfinance
        #!pip install pandas
        #!pip install requests
        !pip install bs4
        #!pip install plotly
```

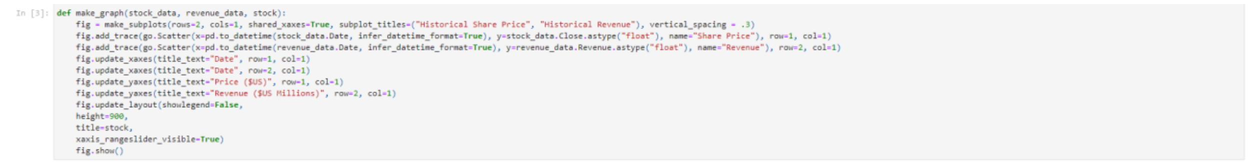
```
Collecting yfinance
  Downloading yfinance-0.1.55.tar.gz (23 kB)
Requirement already satisfied: pandas>=0.24 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from yfinance) (1.0.5)
Requirement already satisfied: numpy>=1.15 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from yfinance) (1.18.5)
Requirement already satisfied: requests>=2.20 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from yfinance) (2.24.0)
Collecting multitasking>=0.0.7
  Downloading multitasking-0.0.9.tar.gz (8.1 kB)
Requirement already satisfied: lxml>=4.5.1 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from yfinance) (4.5.1)
Requirement already satisfied: python-dateutil>=2.6.1 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from pandas>=0.24->yfinance) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from pandas>=0.24->yfinance) (2020.1)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests>=2.20->yfinance) (2.9)
Requirement already satisfied: chardet<4,>=3.0.2 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests>=2.20->yfinance) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests>=2.20->yfinance) (1.25.9)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests>=2.20->yfinance) (2020.12.5)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from python-dateutil>=2.6.1->pandas>=0.24->yfinance) (1.15.0)
Building wheels for collected packages: yfinance, multitasking
  Building wheel for yfinance (setup.py) ... done
  Created wheel for yfinance: filename=yfinance-0.1.55-py2.py3-none-any.whl size=22616 sha256=2a9fcdfdbf93ca32986184ad242ba1b145df0237b01f2e79a2967807d419d6fe
  Stored in directory: /tmp/wsuser/.cache/pip/wheels/aa/8a/36/59ed4f6fbcb6100967618eeb0696046bf9777a41ac2ff1f9b9
  Building wheel for multitasking (setup.py) ... done
  Created wheel for multitasking: filename=multitasking-0.0.9-py3-none-any.whl size=8366 sha256=444646c83667c7ac63b3e7e7a49d1382d32181c1591531695c4d84308d0cabad
  Stored in directory: /tmp/wsuser/.cache/pip/wheels/ae/25/47/4d68431a7ec1b6c4b5233365934b74c1d4e665bf5f968d363a
Successfully built yfinance multitasking
Installing collected packages: multitasking, yfinance
Successfully installed multitasking-0.0.9 yfinance-0.1.55
Collecting bs4
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
Requirement already satisfied: beautifulsoup4 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from bs4) (4.9.1)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from beautifulsoup4->bs4) (2.0.1)
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py) ... done
  Created wheel for bs4: filename=bs4-0.0.1-py3-none-any.whl size=1272 sha256=460b970d31fd1de8fac30153f7f5e2fec4557797e7af3441f445bc6aa4ba66e6
  Stored in directory: /tmp/wsuser/.cache/pip/wheels/0a/9e/ba/20e5bbc1afef3a491f0b3bb74d508f99403aabe76eda2167ca
Successfully built bs4
Installing collected packages: bs4
```

In [2]:



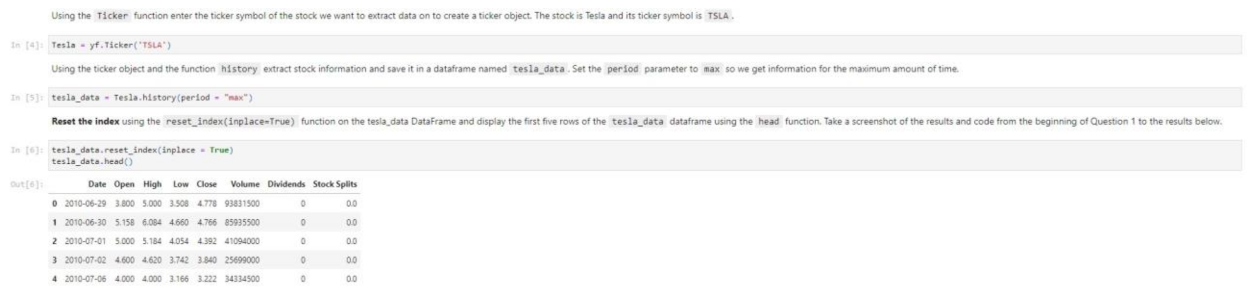
```
In [2]: import yfinance as yf
        import pandas as pd
        import requests
        from bs4 import BeautifulSoup
        import plotly.graph_objects as go
        from plotly.subplots import make_subplots
```

Define Graphing Function



```
In [3]: 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)
            fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data.Date, infer_datetime_format=True), y=stock_data.Close.astype("float"), name="Share Price"), row=1, col=1)
            fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data.Date, infer_datetime_format=True), y=revenue_data.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()
```

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 [4]: 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 [5]: tesla_data = Tesla.history(period = "max")
```

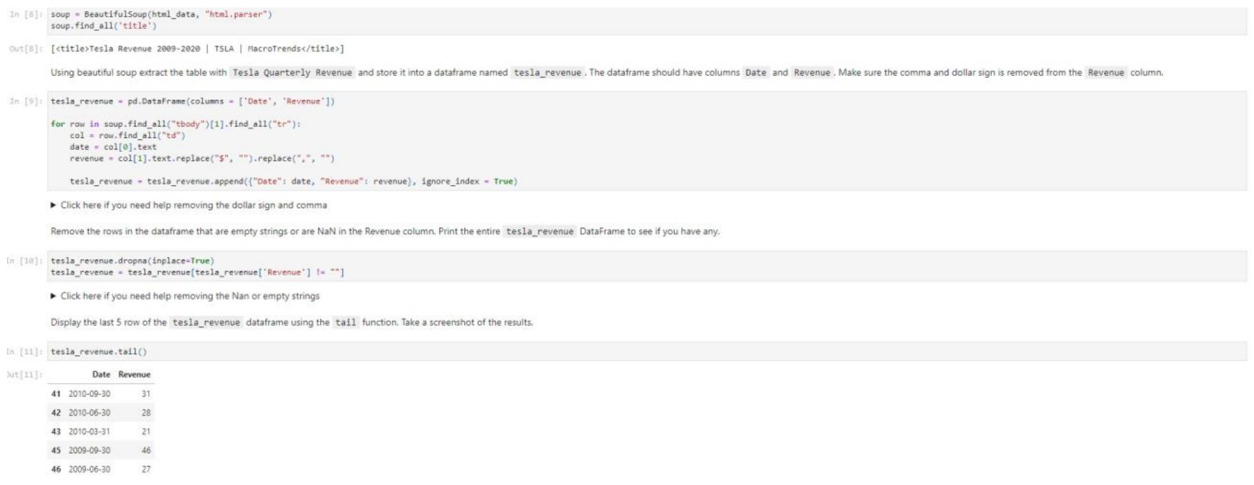
Reset the index using the `reset_index(inplace=True)` function on the tesla_data DataFrame and display the first five rows of the `tesla_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 1 to the results below.

```
In [6]: tesla_data.reset_index(inplace = True)
        tesla_data.head()
```

Out[6]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2010-06-29	3.800	5.000	3.508	4.778	93831500	0	0.0
1	2010-06-30	5.158	6.084	4.660	4.766	85935500	0	0.0
2	2010-07-01	5.000	5.184	4.054	4.392	41094000	0	0.0
3	2010-07-02	4.600	4.620	3.742	3.840	25699000	0	0.0
4	2010-07-06	4.000	4.000	3.166	3.222	34334500	0	0.0

Question 2: Use Webscraping to Extract Tesla Revenue Data



```
In [8]: soup = BeautifulSoup(html_data, "html.parser")
        soup.find_all('title')

Out[8]: [<title>Tesla Revenue 2009-2020 | TSLA | MacroTrends</title>]
```

Using beautiful soup extract the table with `Tesla Quarterly Revenue` and store it into a dataframe named `tesla_revenue`. The dataframe should have columns `Date` and `Revenue`. Make sure the comma and dollar sign is removed from the `Revenue` column.

```
In [9]: tesla_revenue = pd.DataFrame(columns = ['Date', 'Revenue'])

        for row in soup.find_all("tbody")[1].find_all("tr"):
            col = row.find_all("td")
            date = col[0].text
            revenue = col[1].text.replace("$", "").replace(",", "")

            tesla_revenue = tesla_revenue.append({"Date": date, "Revenue": revenue}, ignore_index = True)
```

▶ Click here if you need help removing the dollar sign and comma

Remove the rows in the dataframe that are empty strings or are NaN in the Revenue column. Print the entire `tesla_revenue` DataFrame to see if you have any.

```
In [10]: tesla_revenue.dropna(inplace=True)
         tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
```

▶ Click here if you need help removing the Nan or empty strings

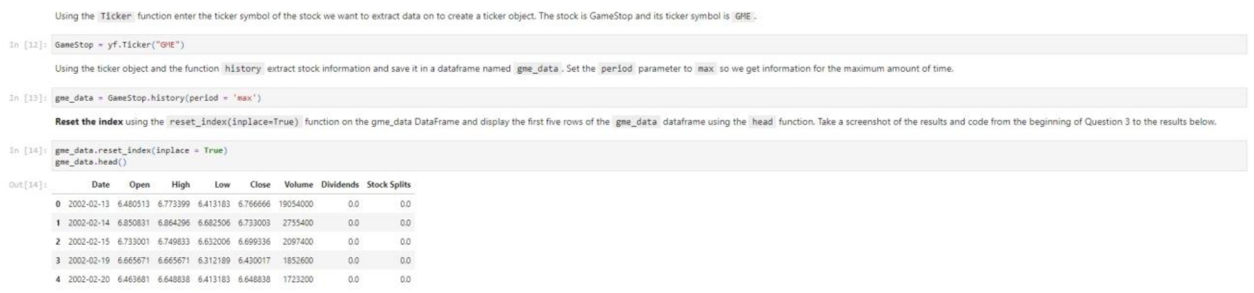
Display the last 5 row of the `tesla_revenue` dataframe using the `tail` function. Take a screenshot of the results.

```
In [11]: tesla_revenue.tail()
```

Out[11]:

	Date	Revenue
41	2010-09-30	31
42	2010-06-30	28
43	2010-03-31	21
45	2009-09-30	46
46	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 [12]: GameStop = 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 [13]: gme_data = GameStop.history(period = 'max')
```

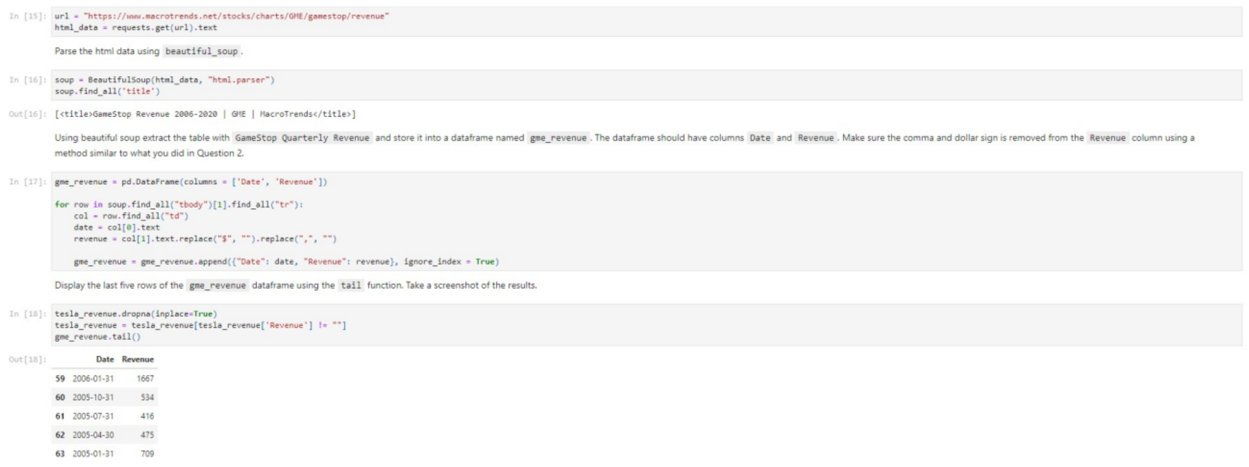
Reset the index using the `reset_index(inplace=True)` function on the gme_data DataFrame and display the first five rows of the `gme_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 3 to the results below.

```
In [14]: gme_data.reset_index(inplace = True)
         gme_data.head()
```

Out[14]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-02-13	6.480513	6.773399	6.413183	6.766666	19054000	0.0	0.0
1	2002-02-14	6.850831	6.864296	6.682506	6.733003	2755400	0.0	0.0
2	2002-02-15	6.733001	6.749833	6.632006	6.699336	2097400	0.0	0.0
3	2002-02-19	6.665671	6.665671	6.312189	6.430017	1852600	0.0	0.0
4	2002-02-20	6.463681	6.648838	6.413183	6.648838	1723200	0.0	0.0

Question 4: Use Webscraping to Extract GME Revenue Data



```
In [15]: url = "https://www.macrotrends.net/stocks/charts/GME/gamestop/revenue"
         html_data = requests.get(url).text
```

Parse the html data using `beautiful_soup`.

```
In [16]: soup = BeautifulSoup(html_data, "html.parser")
         soup.find_all('title')

Out[16]: [<title>GameStop Revenue 2006-2020 | GME | MacroTrends</title>]
```

Using beautiful soup extract the table with `GameStop Quarterly 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.

```
In [17]: gme_revenue = pd.DataFrame(columns = ['Date', 'Revenue'])

         for row in soup.find_all("tbody")[1].find_all("tr"):
             col = row.find_all("td")
             date = col[0].text
             revenue = col[1].text.replace("$", "").replace(",", "")

             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.

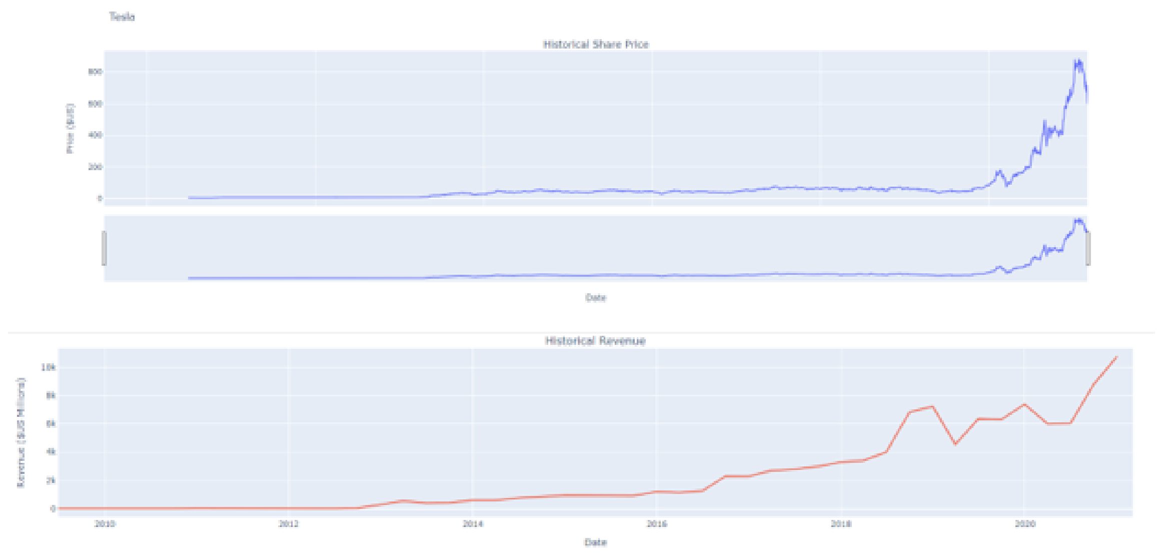
```
In [18]: tesla_revenue.dropna(inplace=True)
         tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
         gme_revenue.tail()
```

Out[18]:

	Date	Revenue
59	2006-01-31	1667
60	2005-10-31	534
61	2005-07-31	416
62	2005-04-30	475
63	2005-01-31	709

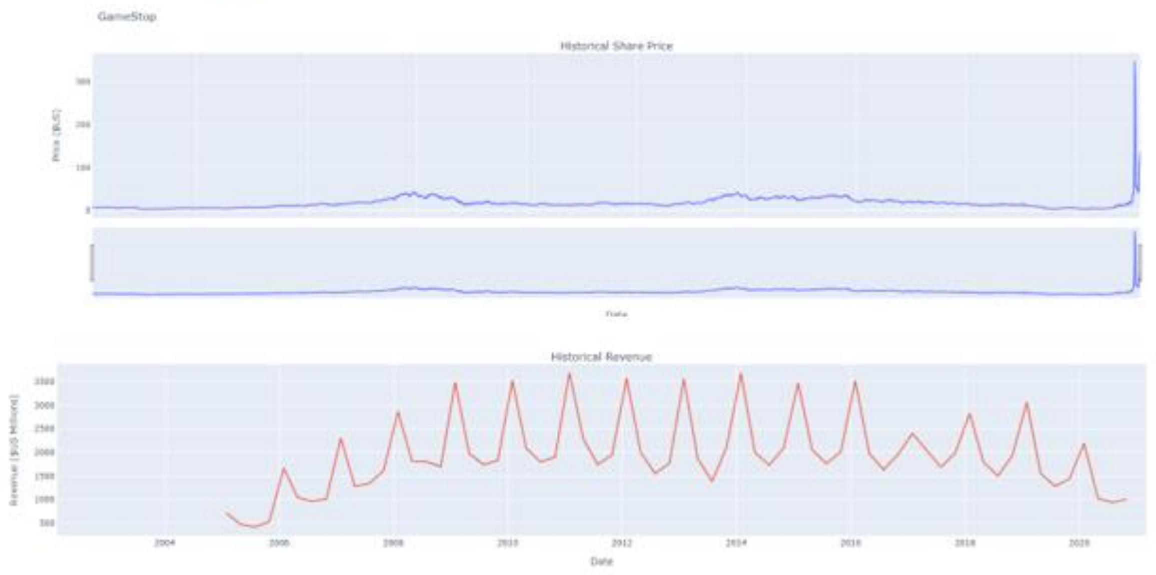
Question 5: Plot Tesla Stock Graph

```
In [18]: make_graph(tesla_data, tesla_revenue, 'Tesla')
```



Question 6: Plot GameStop Stock Graph

```
In [30]: make_graph(gme_data, gme_revenue, 'GameStop')
```


GameStop
Historical Share Price
Price ($US)
Date
Historical Revenue
Revenue ($US Millions)
Date