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

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: yfinance in c:\users\hp\appdata\roaming\python\python39\site-packages (0.2.18)
Requirement already satisfied: html5lib>=1.1 in c:\users\hp\appdata\roaming\python\python39\site-packages (from yfinance) (1.1)
Requirement already satisfied: requests>=2.26 in c:\programdata\anaconda3\lib\site-packages (from yfinance) (2.27.1)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\hp\appdata\roaming\python\python39\site-packages (from yfinance) (0.0.11)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\hp\appdata\roaming\python\python39\site-packages (from yfinance) (2.3.8)
Requirement already satisfied: lxml>=4.9.1 in c:\users\hp\appdata\roaming\python\python39\site-packages (from yfinance) (4.9.2)
Requirement already satisfied: cryptography>=3.3.2 in c:\programdata\anaconda3\lib\site-packages (from yfinance) (3.4.8)
Requirement already satisfied: appdirs>=1.4.4 in c:\programdata\anaconda3\lib\site-packages (from yfinance) (1.4.4)
Requirement already satisfied: pandas>=1.3.0 in c:\programdata\anaconda3\lib\site-packages (from yfinance) (1.4.2)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\programdata\anaconda3\lib\site-packages (from yfinance) (4.11.1)
Requirement already satisfied: pytz>=2022.5 in c:\users\hp\appdata\roaming\python\python39\site-packages (from yfinance) (2023.3)
Requirement already satisfied: numpy>=1.16.5 in c:\programdata\anaconda3\lib\site-packages (from yfinance) (1.21.5)
Requirement already satisfied: soupsieve>1.2 in c:\programdata\anaconda3\lib\site-packages (from beautifulsoup4>=4.11.1->yfinance) (2.3.1)
Requirement already satisfied: cffi>=1.12 in c:\programdata\anaconda3\lib\site-packages (from cryptography>=3.3.2->yfinance) (1.15.0)
Requirement already satisfied: pycparser in c:\programdata\anaconda3\lib\site-packages (from cffi>=1.12->cryptography>=3.3.2->yfinance) (2.21)
Requirement already satisfied: webencodings in c:\programdata\anaconda3\lib\site-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: six>=1.9 in c:\programdata\anaconda3\lib\site-packages (from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\programdata\anaconda3\lib\site-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.26->yfinance) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.26->yfinance) (2021.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.26->yfinance) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.26->yfinance) (3.3)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: bs4 in c:\users\hp\appdata\roaming\python\python39\site-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in c:\programdata\anaconda3\lib\site-packages (from bs4) (4.11.1)
Requirement already satisfied: soupsieve>1.2 in c:\programdata\anaconda3\lib\site-packages (from beautifulsoup4->bs4) (2.3.1)
```

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 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 [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 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	257060500	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

Question 2: Use Webscraping to Extract Tesla Revenue Data

Use the requests library to download the webpage https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue. Save the text of the response as a variable named html_data.

In [7]:
```
url= "https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue"
html_data=requests.get(url).text
```

Parse the html data using beautiful_soup.

In [8]:
```
soup = BeautifulSoup(html_data, "html5lib")
```

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.read_html(url, match="Tesla Quarterly Revenue", flavor='bs4')[0]
tesla_revenue=tesla_revenue.rename(columns = {'Tesla Quarterly Revenue(Millions of US $)': 'Date', 'Tesla Quarterly Revenue(Millions of US $).1': 'Revenue'})
tesla_revenue["Revenue"] = tesla_revenue["Revenue"].str.replace(",","").str.replace("$","")
tesla_revenue.head()
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_20540\961858309.py:3: FutureWarning: The default value of regex will change from True to False in a future version. I
n addition, single character regular expressions will *not* be treated as literal strings when regex=True.
  tesla_revenue["Revenue"] = tesla_revenue["Revenue"].str.replace(",","").str.replace("$","")
```

Out[9]:

	Date	Revenue
0	2023-03-31	23329
1	2022-12-31	24318
2	2022-09-30	21454
3	2022-06-30	16934
4	2022-03-31	18756

In [10]:
```
tesla_revenue.head()
```

Out[10]:

	Date	Revenue
0	2023-03-31	23329
1	2022-12-31	24318
2	2022-09-30	21454
3	2022-06-30	16934
4	2022-03-31	18756

In [11]:
```
tesla_revenue.dropna(inplace=True)
tesla_revenue.tail()
```

Out[11]:

	Date	Revenue
50	2010-09-30	31
51	2010-06-30	28
52	2010-03-31	21
54	2009-09-30	46
55	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 00:00:00-05:00	1.620128	1.693349	1.603296	1.691666	76216000	0.0	0.0
1	2002-02-14 00:00:00-05:00	1.712707	1.716074	1.670626	1.683250	11021000	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	7414000	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://www.macrotrends.net/stocks/charts/GME/gamestop/revenue. Save the text of the response as a variable named html_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,"html5lib")
```

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.read_html(url, match="GameStop Quarterly Revenue", flavor='bs4')[0]
gme_revenue=gme_revenue.rename(columns = {'GameStop Quarterly Revenue(Millions of US $)': 'Date', 'GameStop Quarterly Revenue(Millions of US $).1': 'Revenue'})
gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace(",","").str.replace("$","")
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_20540\955646331.py:3: FutureWarning: The default value of regex will change from True to False in a future version. I
n addition, single character regular expressions will *not* be treated as literal strings when regex=True.
  gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace(",","").str.replace("$","")
```

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

In [18]:
```
gme_revenue.dropna(inplace=True)
gme_revenue.tail()
```

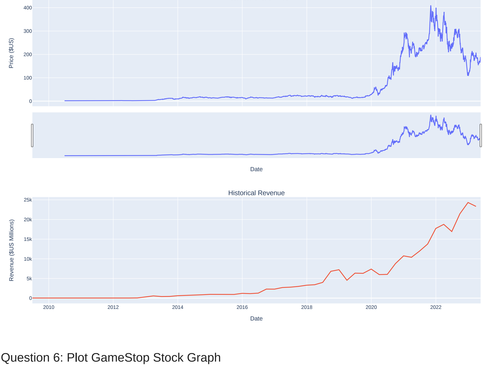
Out[18]:

	Date	Revenue
52	2010-01-31	3524
53	2009-10-31	1935
54	2009-07-31	1739
55	2009-04-30	1981
56	2009-01-31	3492

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')

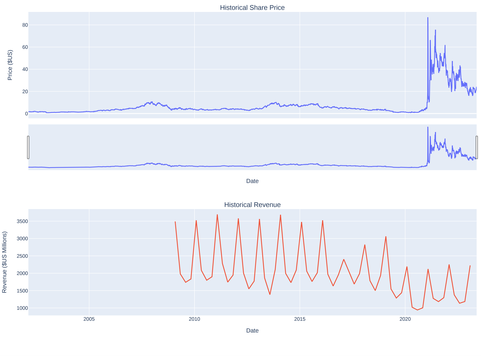
In [19]:
```
make_graph(tesla_data, tesla_revenue, 'Tesla Stock Data Graph')
```



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').

In [20]:
```
make_graph(gme_data, gme_revenue, 'GameStop Stock Data Graph')
```



In [ ]:
```
```