```
!pip install yfinance==0.1.67
!mamba install bs4==4.10.0 -y
!pip install nbformat==4.2.0
```

```
Collecting yfinance==0.1.67
  Downloading yfinance-0.1.67-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: pandas>=0.24 in /opt/python/envs/default/li
Requirement already satisfied: numpy>=1.15 in /opt/python/envs/default/lib
Requirement already satisfied: requests>=2.20 in /opt/python/envs/default/
Collecting multitasking>=0.0.7 (from yfinance==0.1.67)
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Requirement already satisfied: lxml>=4.5.1 in /opt/python/envs/default/lib
Requirement already satisfied: python-dateutil>=2.8.1 in /opt/python/envs/
Requirement already satisfied: pytz>=2020.1 in /opt/python/envs/default/li
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/python/env
Requirement already satisfied: idna<4,>=2.5 in /opt/python/envs/default/li
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/python/envs/defa
Requirement already satisfied: certifi>=2017.4.17 in /opt/python/envs/defa
Requirement already satisfied: six>=1.5 in /opt/python/envs/default/lib/py
Installing collected packages: multitasking, yfinance
Successfully installed multitasking-0.0.11 yfinance-0.1.67

[notice] A new release of pip is available: 23.1.2 -> 23.3.1
[notice] To update, run: pip install --upgrade pip
```

```
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
```

```
ModuleNotFoundError: No module named 'yfinance'
```

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

```
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("H
    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
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date, inf
    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()
```

```
import yf
# use the ticker function to enter the ticker symbol 'TSLA' and create a tick
# Use yfinance's Ticker function to create a Ticker object for Tesla
tesla_ticker:  = yf.Ticker('TSLA')
```

```
ModuleNotFoundError: No module named 'yf'
```

```
# import yfinance as yf
# Import the yfinance library with alias yf
import yfinance as yf

# If we want to use it for another stock, we can follow the same procedure as
apple_ticker = yf.Ticker('AAPL')
```

```
tesla_ticker = yf.Ticker("TSLA")
```

```
# use the ticker object an the function history to extract stock information
# Use the 'history' function of the tesla_ticker object, set period to 'max'
tesla_data = tesla_ticker.history(period="max")
```

```
# reset the index using the reset_index(inplace=True) function on the tesla_a
# Reset the index of the DataFrame
tesla_data.reset_index(inplace=True)

# Display the first five rows of the DataFrame
tesla_data.head()
```

	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

```
# use requests library to download a webpage
# Import the library
import requests

# Specify the URL of the page you want to access
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM

# Send HTTP request
response = requests.get("https://cf-courses-data.s3.us.cloud-object-storage.a

# Confirm that the request was successful
if response.status_code == 200:
    print("Webpage downloaded successfully")
    # Access the content with response.content or response.text
else:
    print("Failed to download webpage")
```

```
Webpage downloaded successfully
```

```
# save the text of the response as a variable named html_data
# Save the content of the webpage to a variable
html_data = response.text
```

```
print(response.text)
```

```
<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
```

```
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
                <link rel="canonical" href="https://www.macrotrends.net/st
                <title>Tesla Revenue 2010-2022 | TSLA | MacroTrends</title
                <meta name="description" content="Tesla annual/quarterly r

                                    &lt;ul style='margin-top:10px;'&gt;
                                    &lt;li&gt;Tesla revenue for the quarter en
                                    &lt;li&gt;Tesla revenue for the twelve mon
                                    &lt;li&gt;Tesla annual revenue for 2021 wa
                                    &lt;li&gt;Tesla annual revenue for 2020 wa
                                    &lt;li&gt;Tesla annual revenue for 2019 wa
```

```
# parse the html data using beautiful_soup
# Import the Beautiful Soup library
from bs4 import BeautifulSoup

# Parse the HTML content using Beautiful Soup
soup = BeautifulSoup(html_data, "html.parser")
```

```
import pandas as pd
# extract the table with Tesla Revenue and store it into a dataframe named te
# Using BeautifulSoup object soup to extract the table and convert it into a
table = soup.find_all('table')[0]
tesla_revenue = pd.read_html(str(table))[0]
tesla_revenue.head()
```

	Tesla Annual Revenue (Millions of US $)	Tesla Annual Revenue (Millions of US $).1
0	2021	$53,823
1	2020	$31,536
2	2019	$24,578
3	2018	$21,461
4	2017	$11,759

```
# remove the comma and dollar signs from the revenue column
# removing the comma and dollar signs from the revenue column
tesla_revenue['Tesla Annual Revenue (Millions of US $).1'] = tesla_revenue['T
```

```
# remove any null or empty strings in the Revenue column
# Removing the null values
tesla_revenue = tesla_revenue[tesla_revenue['Tesla Annual Revenue (Millions o

# Removing the empty strings
tesla_revenue = tesla_revenue[tesla_revenue['Tesla Annual Revenue (Millions o

# Display the updated DataFrame
tesla_revenue.head()
```

	Tesla Annual Revenue (Millions of US $)	Tesla Annual Revenue (Millions of US $).1
0	2021	53823
1	2020	31536
2	2019	24578
3	2018	21461
4	2017	11759

```
print(tesla_revenue)
```

```
    Tesla Annual Revenue (Millions of US $)  \
0                                      2021
1                                      2020
2                                      2019
3                                      2018
4                                      2017
5                                      2016
6                                      2015
7                                      2014
8                                      2013
9                                      2012
10                                     2011
11                                     2010
12                                     2009

   Tesla Annual Revenue (Millions of US $).1
0                                     53823
1                                     31536
2                                     24578
3                                     21461
```

```
# display the last 5 rows of the tesla_revenue dataframe using the tail funct
# Displaying last 5 rows of tesla_revenue DataFrame
tesla_revenue.tail()
```

	Tesla Annual Revenue (Millions of US $)	Tesla Annual Revenue (Millions of US $).1
8	2013	2013
9	2012	413
10	2011	204
11	2010	117
12	2009	112

```
# create a ticker object for gamestop. the ticker symbol is GME
# Import yfinance using an alias
import yfinance as yf

# Using the Ticker function to create a ticker object for GameStop
gme_ticker = yf.Ticker('GME')
```

```
# use the ticker object and the history function to extract stock information
# Extract historical stock information for GameStop
gme_data = gme_ticker.history(period="max")

# Display the first five rows of DataFrame
gme_data.head()
```

	Open	High	Low	Close	Volume	Dividends	Stock Splits
Date							
2002-02-13 00:00:00-05:00	1.620129	1.693350	1.603296	1.691667	76216000	0.0	0.0
2002-02-14 00:00:00-05:00	1.712707	1.716073	1.670625	1.683250	11021600	0.0	0.0
2002-02-15 00:00:00-05:00	1.683250	1.687458	1.658001	1.674833	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	1.615920	1.662210	1.603296	1.662210	6892800	0.0	0.0

```
# reset the index and display the first five rows of the gme_data dataframe u
# Reset the index of the DataFrame
gme_data.reset_index(inplace=True)

# Display the first five rows of the DataFrame
gme_data.head()
```

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-02-13 00:00:00-05:00	1.620129	1.693350	1.603296	1.691667	76216000	0.0	0.0
1	2002-02-14 00:00:00-05:00	1.712707	1.716073	1.670625	1.683250	11021600	0.0	0.0
2	2002-02-15 00:00:00-05:00	1.683250	1.687458	1.658001	1.674833	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	1.615920	1.662210	1.603296	1.662210	6892800	0.0	0.0

```
# use the requests library to download a webpage and save the text of the res
# Import the requests library
import requests

# Use the get method in the requests library to access the webpage
response = requests.get("https://cf-courses-data.s3.us.cloud-object-storage.a

# Store the content of the response in a variable named html_data as a text f
html_data = response.text
```

```
# parse the html data using beautiful soup
# Import Beautiful Soup
from bs4 import BeautifulSoup

# Parse the html data with Beautiful Soup
soup = BeautifulSoup(html_data, 'html.parser')
```

```
# use the BeautifulSoup function to extract the table with GameStop Revenue
# Using BeautifulSoup object 'soup' to extract the correct table and convert
table = soup.find_all('table')[1] # Modify this index if the table for GameSt
gme_revenue = pd.read_html(str(table))[0] # Converting the parsed HTML table

# Renaming the columns
gme_revenue.columns = ['Date', 'Revenue']

# Cleaning the 'Revenue' column by removing the comma and dollar signs
gme_revenue['Revenue'] = gme_revenue['Revenue'].replace({'\$': '', ',': ''},
gme_revenue
```

	Date	Revenue
0	2020-04-30	1021
1	2020-01-31	2194
2	2019-10-31	1439
3	2019-07-31	1286
4	2019-04-30	1548
...	...	...
57	2006-01-31	1667
58	2005-10-31	534
59	2005-07-31	416
60	2005-04-30	475
61	2005-01-31	709

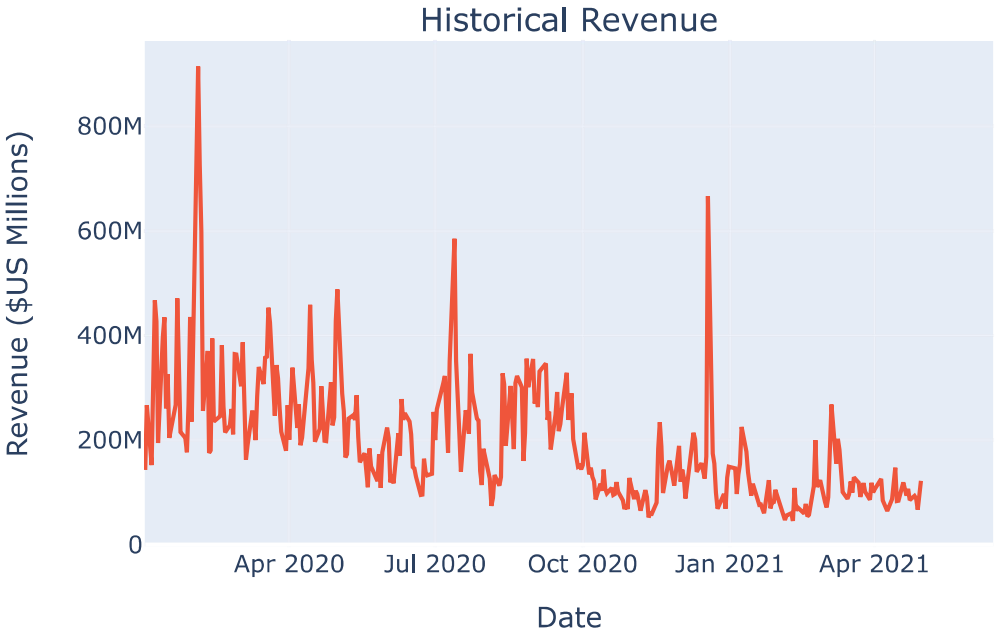
62 rows × 2 columns

```
# Display the last five rows of the gme_revenue dataframe using the tail func
# The last 5 rows of the gme_revenue DataFrame
gme_revenue.tail()
```

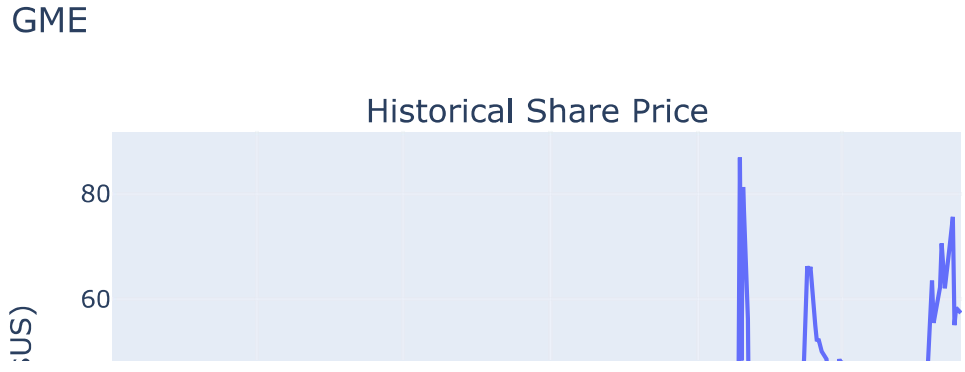
	Date	Revenue
57	2006-01-31	1667
58	2005-10-31	534
59	2005-07-31	416
60	2005-04-30	475
61	2005-01-31	709

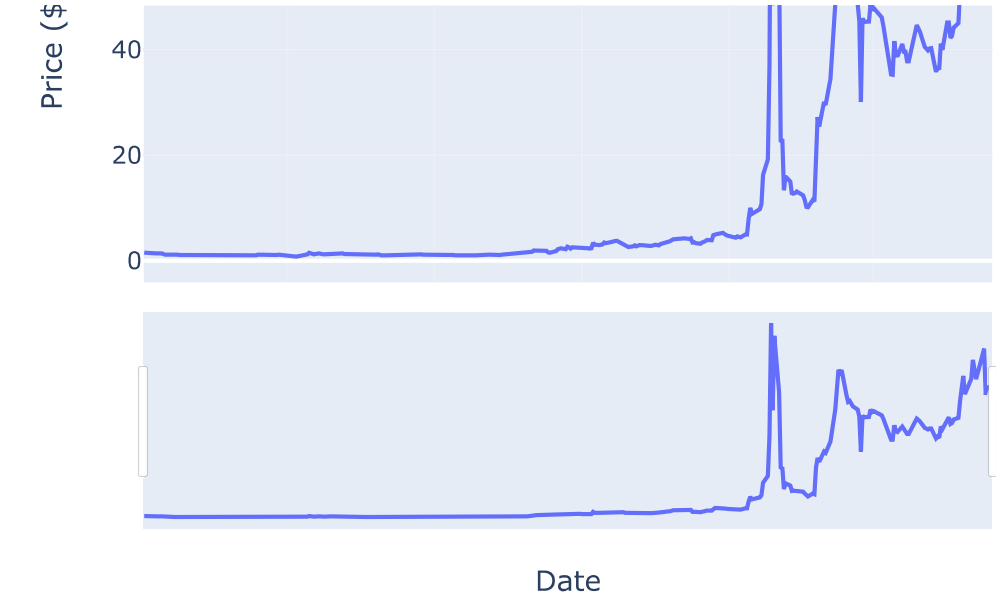
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import yfinance as yf
import pandas as pd

stock_data = yf.download("TSLA", start="2020-01-01", end="2021-09-30", progre
revenue_data = yf.download("TSLA", start="2020-01-01", end="2021-09-30", prog
stock_data.reset_index(inplace=True)
revenue_data.reset_index(inplace=True)

def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1,
                        shared_xaxes=True,
                        subplot_titles=("Historical Share Price", "Historical
                        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 Price"), row

    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date, inf
                             y=revenue_data_specific.Volume.astype("float"),
                             name="Volume"), 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()

make_graph(stock_data, revenue_data, 'TSLA')
```

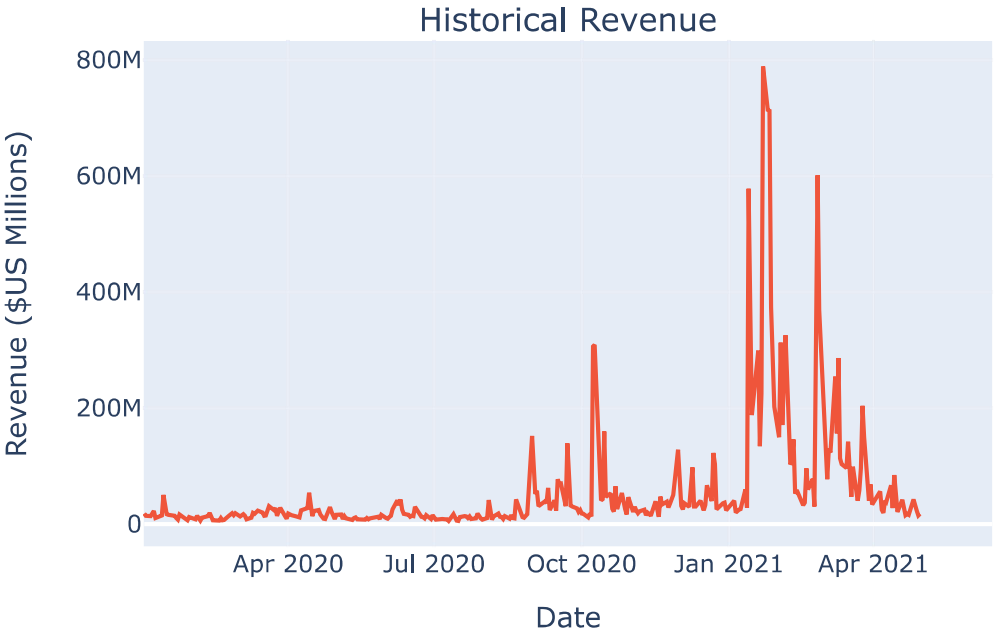

Price ($
150
100
50
Date


Historical Revenue
Revenue ($US Millions)
800M
600M
400M
200M
0
Apr 2020
Jul 2020
Oct 2020
Jan 2021
Apr 2021
Date

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import yfinance as yf
import pandas as pd

stock_data = yf.download("GME", start="2020-01-01", end="2021-09-30", progres
revenue_data = yf.download("GME", start="2020-01-01", end="2021-09-30", progr
stock_data.reset_index(inplace=True)
revenue_data.reset_index(inplace=True)

def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1,
                        shared_xaxes=True,
                        subplot_titles=("Historical Share Price", "Historical
                        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 Price"), row

    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date, inf
                             y=revenue_data_specific.Volume.astype("float"),
                             name="Volume"), 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()

make_graph(stock_data, revenue_data, 'GME')
```




Price ($
40
20
0
Date


Historical Revenue
Revenue ($US Millions)
800M
600M
400M
200M
0
Apr 2020
Jul 2020
Oct 2020
Jan 2021
Apr 2021
Date