# Extracting and Visualizing Stock Data

## Description

Extracting essential data from a dataset and displaying it is a necessary part of data science; therefore individuals can make correct decisions based on the data. In this assignment, you will extract some stock data, you will then display this data in a graph.

## Table of Contents



Estimated Time Needed: **30 min**

---

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

```
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbformat) (0
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbformat) (0.20.0
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dist-packages (from jupyter-core!=5.0.*,>=4.12->nbf
Downloading nbformat-5.10.4-py3-none-any.whl (78 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.5/78.5 kB 1.8 MB/s eta 0:00:00
Installing collected packages: nbformat
  Attempting uninstall: nbformat
    Found existing installation: nbformat 4.2.0
    Uninstalling nbformat-4.2.0:
      Successfully uninstalled nbformat-4.2.0
Successfully installed nbformat-5.10.4
```

```
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.

```
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_spa
    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(
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date, infer_datetime_format=True), y=revenue_data_specific.Revenue.a
    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`.

```
ticker = 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_data = ticker.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.

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

Next steps: Generate code with tesla_data | View recommended plots | New interactive sheet

## 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`.

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

Parse the html data using `beautiful_soup`.

```
soup = BeautifulSoup(html_data, 'html5lib')
```

Using `BeautifulSoup` or the `read_html` function extract the table with `Tesla Quarterly 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

```
tables = soup.find_all('table')
table_index = None

# Find the index of the table with "Tesla Quarterly Revenue" in its content
for index, table in enumerate(tables):
    if "Tesla Quarterly Revenue" in str(table):
        table_index = index

if table_index is not None:
    tesla_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

    for row in tables[table_index].tbody.find_all('tr'):
        col = row.find_all('td')
        if col:
            date = col[0].text
            revenue = col[1].text
            tesla_revenue = pd.concat([tesla_revenue, pd.DataFrame({'Date': [date], 'Revenue': [revenue]})], ignore_index=True)

    print(tesla_revenue.head())
else:
    print("Table not found")
```

```
         Date  Revenue
0  2022-09-30  $21,454
1  2022-06-30  $16,934
2  2022-03-31  $18,756
3  2021-12-31  $17,719
4  2021-09-30  $13,757
```

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

```
tesla_revenue["Revenue"] = tesla_revenue["Revenue"].str.replace(',', '').str.replace('$', '')
```

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

```
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.

```
tesla_revenue.tail()
```

| | 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`.

```
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.

```
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.

```
gme_data.reset_index(inplace=True)
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 | 1.716074 | 1.670626 | 1.683250 | 11021600 | 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 | 1.666417 | 1.666417 | 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 |

Next steps: Generate code with `gme_data` View recommended plots New interactive sheet

## 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`.

```
html_data = requests.get("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNe
```

Parse the html data using `beautiful_soup`.

```
soup = BeautifulSoup(html_data)
```

Using `BeautifulSoup` or the `read_html` function 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.

▶ Click here if you need help locating the table

```
tables = soup.find_all('table')
table_index = None

for index, table in enumerate(tables):
    if "GameStop Quarterly Revenue" in str(table):
```

```
        table_index = index

if table_index is not None:
    gme_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

    for row in tables[table_index].tbody.find_all('tr'):
        col = row.find_all('td')
        if col:
            date = col[0].text
            revenue = col[1].text
            revenue = revenue.replace(',', '').replace('$', '')
            gme_revenue = pd.concat([gme_revenue, pd.DataFrame({'Date': [date], 'Revenue': [revenue]})], ignore_index=True)
    gme_revenue.dropna(inplace=True)
    gme_revenue = gme_revenue[gme_revenue['Revenue'] != ""]

    print(gme_revenue.head())
else:
    print("Table not found")
```

```
         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
```

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

```
gme_revenue.tail()
```

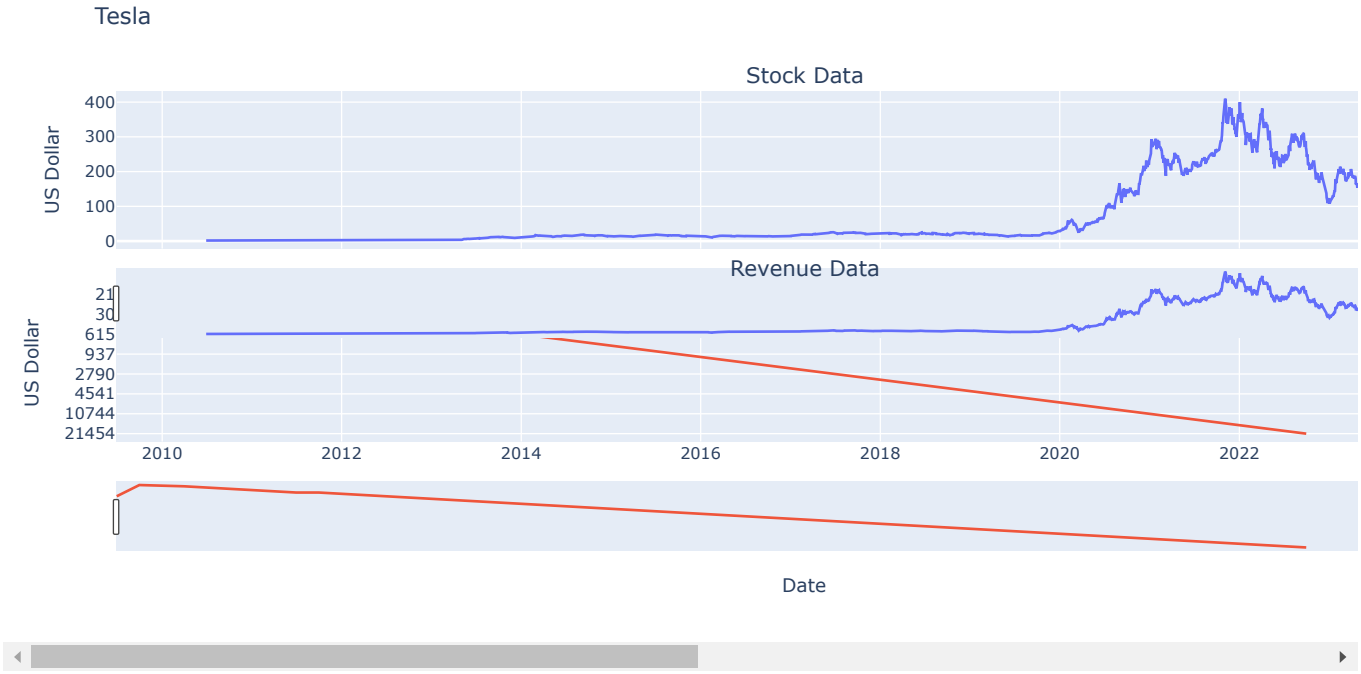
| | Date | Revenue |
|---|---|---|
| **57** | 2006-01-31 | 1667 |
| **58** | 2005-10-31 | 534 |
| **59** | 2005-07-31 | 416 |
| **60** | 2005-04-30 | 475 |
| **61** | 2005-01-31 | 709 |

## 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.

```
def make_graph(data, revenue, title):
    # Create a subplot with Plotly
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, vertical_spacing=0.1,
                        subplot_titles=['Stock Data', 'Revenue Data'])

    # Create traces for stock data
    trace_stock = go.Scatter(x=data['Date'], y=data['Close'], mode='lines', name='Stock Data')
    fig.add_trace(trace_stock, row=1, col=1)

    # Create traces for revenue data
    trace_revenue = go.Scatter(x=revenue['Date'], y=revenue['Revenue'], mode='lines', name='Revenue Data')
    fig.add_trace(trace_revenue, row=2, col=1)

    # Update the layout
    fig.update_layout(title_text=title)
    fig.update_xaxes(title_text='Date', row=2, col=1)
    fig.update_yaxes(title_text='US Dollar', row=1, col=1)
    fig.update_yaxes(title_text='US Dollar', row=2, col=1)

    # Add a range slider for date selection
    fig.update_xaxes(rangeslider_visible=True, row=1, col=1)
    fig.update_xaxes(rangeslider_visible=True, row=2, col=1)

    fig.show()
```

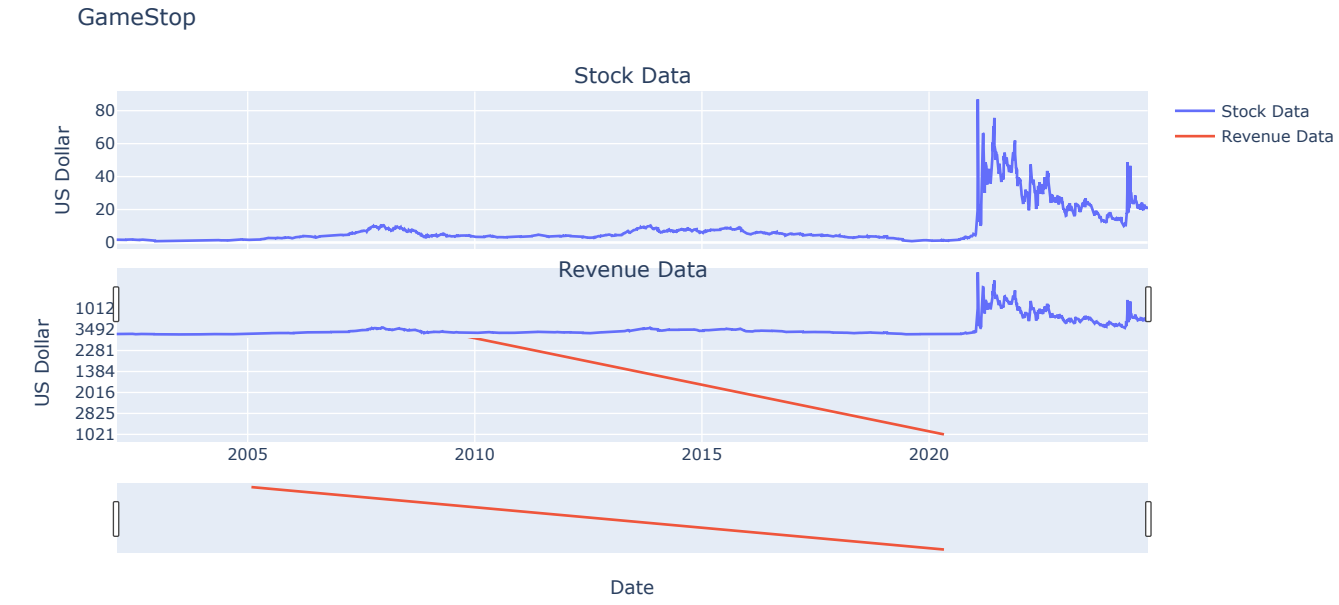
```
make_graph(tesla_data, tesla_revenue, 'Tesla')
```




## 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.

```
g = make_graph(gme_data, gme_revenue, 'GameStop')
```




## About the Authors:

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Azim Hirjani

## Change Log

| Date (YYYY-MM-DD) | Version | Changed By | Change Description |
|---|---|---|---|
| 2022-02-28 | 1.2 | Lakshmi Holla | Changed the URL of GameStop |
| 2020-11-10 | 1.1 | Malika Singla | Deleted the Optional part |
| 2020-08-27 | 1.0 | Malika Singla | Added lab to GitLab |