

# 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.

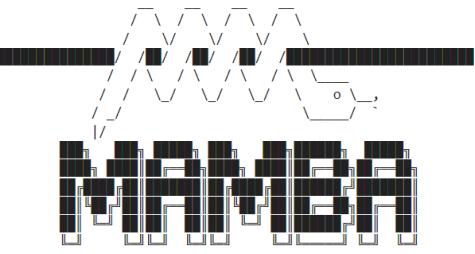
## Table of Contents

- Define a Function that Makes a Graph
- Question 1: Use yfinance to Extract Stock Data
- Question 2: Use Webscraping to Extract Tesla Revenue Data
- Question 3: Use yfinance to Extract Stock Data
- Question 4: Use Webscraping to Extract GME Revenue Data
- Question 5: Plot Tesla Stock Graph
- Question 6: Plot GameStop Stock Graph

Estimated Time Needed: **30 min**

```
In [64]: !pip install yfinance==0.1.67
!mamba install bs4==4.10.0 -y
!pip install nbformat==4.2.0
!mamba install html5lib==1.1 -y

Requirement already satisfied: yfinance==0.1.67 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (0.1.67)
Requirement already satisfied: pandas>=0.24 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (1.3.5)
Requirement already satisfied: requests>=2.20 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (2.28.1)
Requirement already satisfied: lxml>=4.5.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (4.9.1)
Requirement already satisfied: multitasking>=0.0.7 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (0.0.11)
Requirement already satisfied: numpy>=1.15 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (1.21.6)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from pandas>=0.24->yfinance==0.1.67) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from pandas>=0.24->yfinance==0.1.67) (2022.6)
Requirement already satisfied: charset-normalizer<3,>= in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from requests>=2.20->yfinance==0.1.67) (2.1.1)
Requirement already satisfied: certifi>=2017.4.17 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from requests>=2.20->yfinance==0.1.67) (2022.12.7)
Requirement already satisfied: idna<4,>=2.5 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from requests>=2.20->yfinance==0.1.67) (3.4)
Requirement already satisfied: six>=1.5 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from python-dateutil>=2.7.3->pandas>=0.24->yfinance==0.1.67) (1.16.0)

mamba (0.15.3) supported by @QuantStack  

GitHub: https://github.com/mamba-org/mamba  

Twitter: https://twitter.com/QuantStack  

Looking for: ['bs4==4.10.0']

pkgs/main/linux-64      [>                ] (----) No change
pkgs/main/linux-64      [=*=*=*=*=*=*=*=] (00m:00s) No change
pkgs/main/noarch         [>                ] (----) No change
pkgs/main/noarch         [=*=*=*=*=*=*=*=] (00m:00s) No change
pkgs/r/linux-64          [>                ] (----) No change
pkgs/r/linux-64          [=*=*=*=*=*=*=*=] (00m:00s) No change
pkgs/r/noarch            [>                ] (----) No change
pkgs/r/noarch            [=*=*=*=*=*=*=*=] (00m:00s) No change

Pinned packages:
- python 3.7.*

Transaction
Prefix: /home/jupyterlab/conda/envs/python

All requested packages already installed

Requirement already satisfied: nbformat==4.2.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (4.2.0)
Requirement already satisfied: jupyter-core in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from nbformat==4.2.0) (4.12.0)
Requirement already satisfied: traitlets>=4.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from nbformat==4.2.0) (5.6.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from nbformat==4.2.0) (4.17.3)
Requirement already satisfied: ipython-genutils in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from nbformat==4.2.0) (0.2.0)
Requirement already satisfied: importlib-resources>=1.4.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (5.1.0)
Requirement already satisfied: attrs>=17.4.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (22.1.0)
Requirement already satisfied: tunnia-extensions in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (4.4.0)
```

```

Requirement already satisfied: pkgutil-resolve-name>=1.3.10 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema>=2.5.0,>=2.4->nbformat==4.2.0) (1.3.10)
Requirement already satisfied: importlib-metadata in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema>=2.5.0,>=2.4->nbformat==4.2.0) (4.11.4)
Requirement already satisfied: pyrsistent!=0.17.0,!0.17.1,!0.17.2,>=0.14.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from jsonschema>=2.5.0,>=2.4->nbformat==4.2.0) (0.19.2)
Requirement already satisfied: zipp>=3.1.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from importlib-resources>=1.4.0->jsonschema>=2.5.0,>=2.4->nbformat==4.2.0) (3.11.0)

```



mamba (0.15.3) supported by @QuantStack

GitHub: <https://github.com/mamba-org/mamba>  
Twitter: <https://twitter.com/QuantStack>



Looking for: ['html5lib==1.1']

```

pkgs/main/linux-64      Using cache
pkgs/main/noarch        Using cache
pkgs/r/linux-64         Using cache
pkgs/r/noarch          Using cache

```

Pinned packages:  
- python 3.7.\*

Transaction

Prefix: /home/jupyterlab/conda/envs/python

All requested packages already installed

```
In [66]: 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 [67]: 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)
    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=1, col=1)
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date, infer_datetime_format=True), y=revenue_data_specific.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 [68]: tsla = 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 [69]: tesla_data = tsla.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 [70]: tesla_data.reset_index(inplace=True)
tesla_data.head()
```

	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`.

```
In [88]: url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm'  
html_data = requests.get(url).text
```

Parse the html data using `beautiful_soup`.

```
In [89]: beautiful_soup = BeautifulSoup(html_data, 'html.parser')
```

```
In [90]: #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
```

► Click here if you need help locating the table

```
In [91]: tables = beautiful_soup.find_all('table')  
  
len(tables)  
for index, table in enumerate(tables):  
    if ('Tesla Quarterly Revenue' in str(table)):  
        table_index = index  
print(table_index)  
#print(tables[1].prettyify())  
  
tesla_revenue = pd.DataFrame(columns=['Date', 'Revenue'])  
  
for row in tables[1].tbody.find_all('tr'):.  
    col = row.find_all('td')  
    if (col != []):  
        date = col[0].text.strip()  
        revenue = col[1].text.strip()  
        tesla_revenue = tesla_revenue.append({'Date':date, 'Revenue':revenue}, ignore_index = True)  
  
print(tesla_revenue)
```

```
1  
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  
5 2021-06-30 $11,958  
6 2021-03-31 $10,389  
7 2020-12-31 $10,744  
8 2020-09-30 $8,771  
9 2020-06-30 $6,036  
10 2020-03-31 $5,985  
11 2019-12-31 $7,384  
12 2019-09-30 $6,303  
13 2019-06-30 $6,350  
14 2019-03-31 $4,541  
15 2018-12-31 $7,226  
16 2018-09-30 $6,824  
17 2018-06-30 $4,002  
18 2018-03-31 $3,409  
19 2017-12-31 $3,288  
20 2017-09-30 $2,985  
21 2017-06-30 $2,790  
22 2017-03-31 $2,696  
23 2016-12-31 $2,285  
24 2016-09-30 $2,298  
25 2016-06-30 $1,270  
26 2016-03-31 $1,147  
27 2015-12-31 $1,214  
28 2015-09-30 $937  
29 2015-06-30 $955  
30 2015-03-31 $940  
31 2014-12-31 $957  
32 2014-09-30 $852  
33 2014-06-30 $769  
34 2014-03-31 $621  
35 2013-12-31 $615  
36 2013-09-30 $431  
37 2013-06-30 $405  
38 2013-03-31 $562  
39 2012-12-31 $306  
40 2012-09-30 $50  
41 2012-06-30 $27  
42 2012-03-31 $30  
43 2011-12-31 $39  
44 2011-09-30 $58  
45 2011-06-30 $58  
46 2011-03-31 $49  
47 2010-12-31 $36  
48 2010-09-30 $31  
49 2010-06-30 $28  
50 2010-03-31 $21  
51 2009-12-31 $46  
52 2009-09-30 $46  
53 2009-06-30 $27
```

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

```
In [94]: tesla_revenue['Revenue'] = tesla_revenue['Revenue'].str.replace(',|\$', "")
```

```
/home/jupyterlab/conda/envs/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
"""Entry point for launching an IPython kernel.
```

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

```
In [95]: tesla_revenue.dropna(inplace=True)

tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
```

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

```
In [96]: tesla_revenue.tail()
```

Out[96]:

	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`.

```
In [80]: 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.

```
In [81]: 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.

```
In [82]: gme_data.reset_index(inplace=True)
gme_data.head()
```

Out[82]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-02-13	1.620128	1.693349	1.603295	1.691666	76216000	0.0	0.0
1	2002-02-14	1.712707	1.716073	1.670626	1.683250	11021600	0.0	0.0
2	2002-02-15	1.683251	1.687459	1.658002	1.674834	8389600	0.0	0.0
3	2002-02-19	1.666418	1.666418	1.578047	1.607504	7410400	0.0	0.0
4	2002-02-20	1.615920	1.662209	1.603296	1.662209	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/IBMDriverSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html>. Save the text of the response as a variable named `html_data`.

```
In [105]: url_4 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDriverSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html'
html_data = requests.get(url_4).text
```

Parse the html data using `beautiful_soup`.

```
In [107]: soup_4 = BeautifulSoup(html_data, 'html.parser')
```

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

```
In [108]: tables = soup_4.find_all('table')

len(tables)
for index, table in enumerate(tables):
    if ('GameStop Quarterly Revenue' in str(table)):
        table_index = index
print(table_index)
#print(tables[1].prettify())

gme_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

for row in tables[1].tbody.find_all('tr'):
    col = row.find_all('td')
    if (col != []):
        date = col[0].text.strip()
        revenue = col[1].text.strip()
        gme_revenue = gme_revenue.append({'Date':date, 'Revenue':revenue}, ignore_index = True)

print(gme_revenue)
```

	Date	Revenue
0	2020-04-30	\$1,021
1	2020-01-31	\$2,194
2	2019-10-31	\$1,439
3	2019-07-31	\$1,286
4	2019-04-30	\$1,548
..	...	...
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	\$700

```

In [109..]: [62 rows x 2 columns]
gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',', '$', '')
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning: The default value of regex will change from True to False in a future version.
    """Entry point for launching an IPython kernel.

In [110..]: gme_revenue.dropna(inplace=True)
gme_revenue = gme_revenue[gme_revenue['Revenue'] != ""]

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

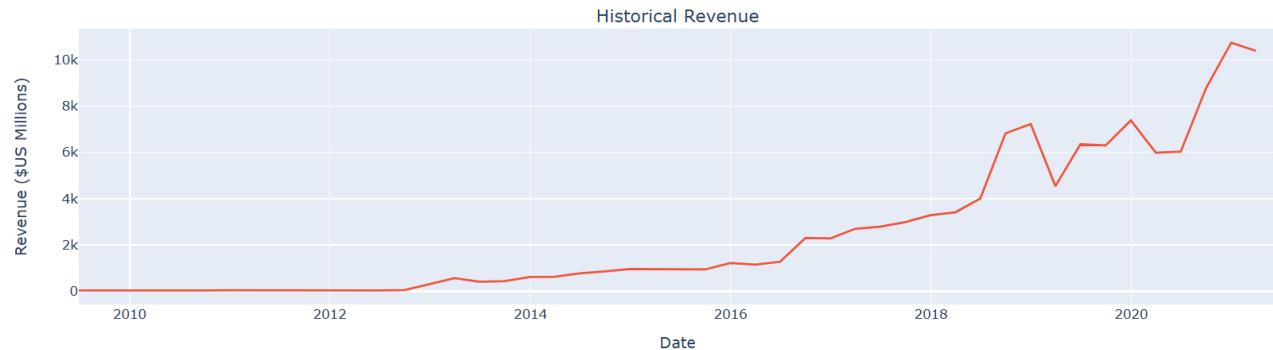
In [111..]: gme_revenue.tail()
Out[111]:
```

	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.

```
In [112..]: make_graph(tesla_data, tesla_revenue, 'Tesla')
```



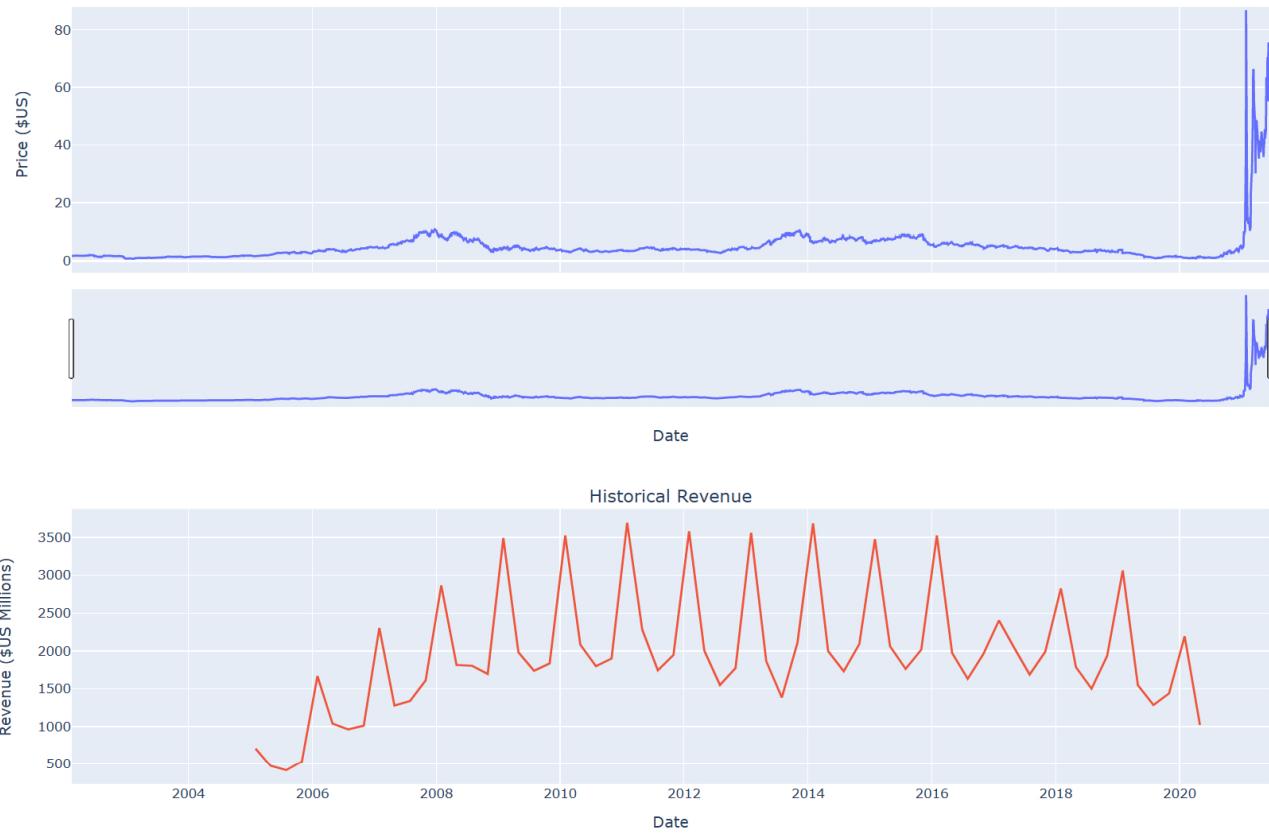
## Question 6: Plot GameStop Stock Graph

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.

```
In [113..]: make_graph(gme_data, gme_revenue, 'GameStop')
```

GameStop

Historical Share Price



In [ ]:

### 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