IBM Developer SKILLS NETWORK

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**

In [10]:
```
!pip install yfinance
#!pip install pandas
#!pip install requests
!pip install bs4
#!pip install plotly
```

```
Requirement already satisfied: yfinance in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (0.1.55)
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: requests>=2.20 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from yfinance) (2.24.0)
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: multitasking>=0.0.7 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from yfinance) (0.0.9)
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: 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: 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->yf
inance) (1.25.9)
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: 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: 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)
Requirement already satisfied: bs4 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (0.0.1)
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)
```

In [11]:
```
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 [12]:
```
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
    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, c
    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 [13]:
```
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 [14]:
```
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 [15]:
```
tesla_data.reset_index(inplace=True)
tesla_data.head()
```

Out[15]:

	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

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 [55]:
```
url = 'https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue'
html_data = requests.get(url).text
```

Parse the html data using `beautiful_soup`.

In [56]:
```
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 [57]:
```
tesla_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

for table in soup.find_all('table'):

    if ('Tesla Quarterly Revenue' in table.find('th').text):
        rows = table.find_all('tr')

        for row in rows:
            col = row.find_all('td')

            if col != []:
                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 [58]:
```
tesla_revenue
```

Out[58]:

	Date	Revenue
0	2020-12-31	10744
1	2020-09-30	8771
2	2020-06-30	6036
3	2020-03-31	5985
4	2019-12-31	7384
5	2019-09-30	6303
6	2019-06-30	6350
7	2019-03-31	4541
8	2018-12-31	7226
9	2018-09-30	6824
10	2018-06-30	4002
11	2018-03-31	3409
12	2017-12-31	3288
13	2017-09-30	2985
14	2017-06-30	2790
15	2017-03-31	2696
16	2016-12-31	2285
17	2016-09-30	2298
18	2016-06-30	1270
19	2016-03-31	1147
20	2015-12-31	1214
21	2015-09-30	937
22	2015-06-30	955
23	2015-03-31	940
24	2014-12-31	957
25	2014-09-30	852
26	2014-06-30	769
27	2014-03-31	621
28	2013-12-31	615
29	2013-09-30	431
30	2013-06-30	405
31	2013-03-31	562
32	2012-12-31	306
33	2012-09-30	50
34	2012-06-30	27
35	2012-03-31	30
36	2011-12-31	39
37	2011-09-30	58
38	2011-06-30	58
39	2011-03-31	49
40	2010-12-31	36
41	2010-09-30	31
42	2010-06-30	28
43	2010-03-31	21
44	2009-12-31	
45	2009-09-30	46
46	2009-06-30	27
47	2008-12-31	

In [59]:
```
tesla_revenue = tesla_revenue[tesla_revenue['Revenue'].astype(bool)]
```

▶ 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 [60]:
```
tesla_revenue.tail()
```

Out[60]:

	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 [37]:
```
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 [43]:
```
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 [44]:
```
gme_data.reset_index(inplace=True)
gme_data.head()
```

Out[44]:

	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

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 [45]:
```
url = 'https://www.macrotrends.net/stocks/charts/GME/gamestop/revenue'
html_data = requests.get(url).text
```

Parse the html data using `beautiful_soup`.

In [46]:
```
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 [49]:
```
gme_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

for table in soup.find_all('table'):

    if ('GameStop Quarterly Revenue' in table.find('th').text):
        rows = table.find_all('tr')

        for row in rows:
            col = row.find_all('td')

            if col != []:
                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 [50]:
```
gme_revenue.tail()
```

Out[50]:

	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

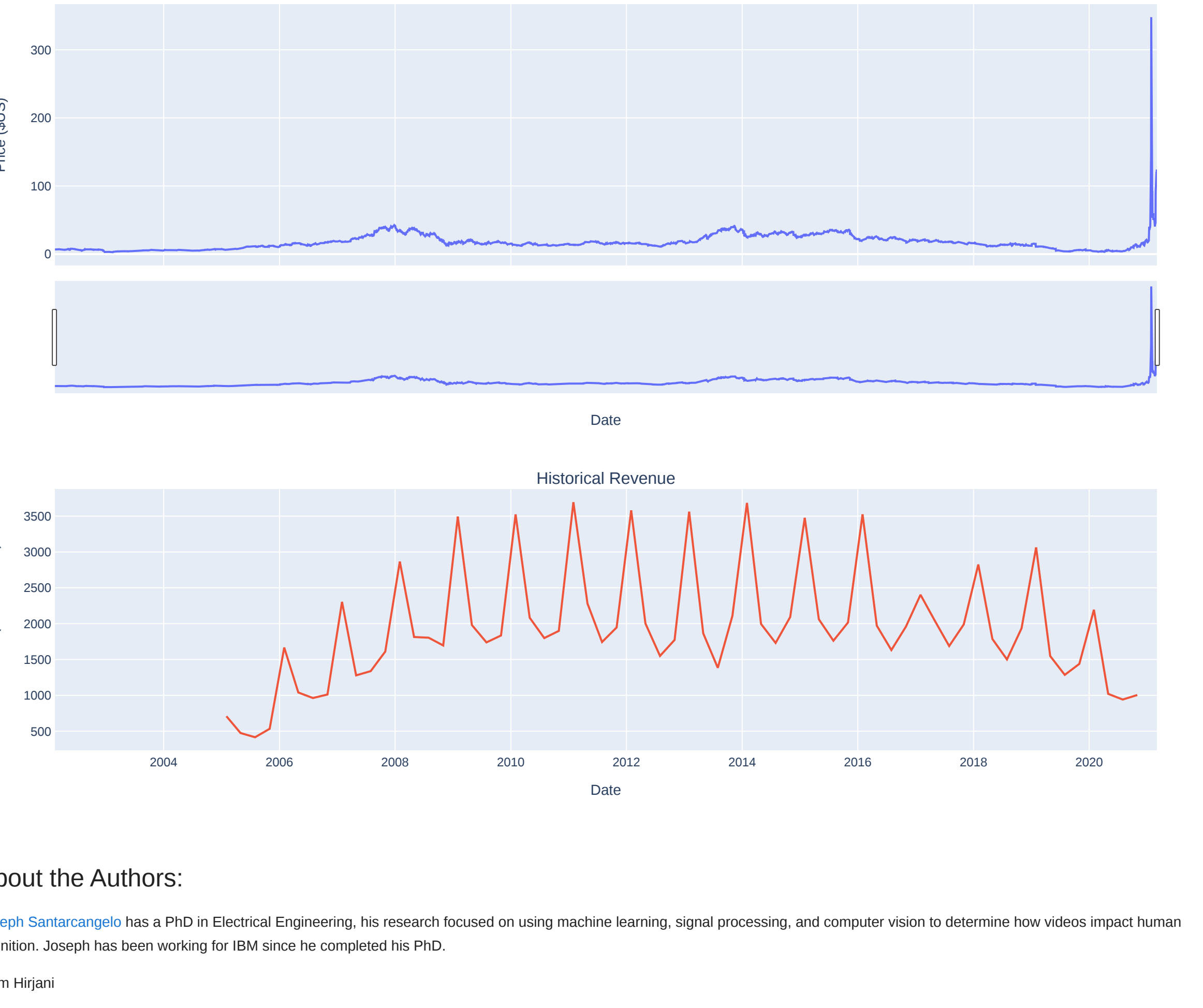
Use the `make_graph` function to graph the Tesla Stock Data, also provide a title for the graph. The structure to call the `make_graph` function is `make_graph(tesla_data, tesla_revenue, 'Tesla')`

In [61]:
```
make_graph(tesla_data[['Date','Close']], 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')`.

In [62]:
```
make_graph(gme_data[['Date','Close']], 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
2020-11-10	1.1	Malika Singla	Deleted the Optional part
2020-08-27	1.0	Malika Singla	Added lab to GitLab