Final Assignment

July 30, 2024

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

Note:- If you are working Locally using anaconda, please uncomment the following code and execute it.

[6]:
```
#!pip install yfinance==0.2.38
#!pip install pandas==2.2.2
#!pip install nbformat
```

[7]:
```
!pip install yfinance
!pip install bs4
!pip install nbformat
```

```
Requirement already satisfied: yfinance in d:\code\anaconda3\lib\site-packages
(0.2.41)
Requirement already satisfied: pandas>=1.3.0 in d:\code\anaconda3\lib\site-
packages (from yfinance) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in d:\code\anaconda3\lib\site-
packages (from yfinance) (1.26.4)
Requirement already satisfied: requests>=2.31 in d:\code\anaconda3\lib\site-
packages (from yfinance) (2.32.2)
Requirement already satisfied: multitasking>=0.0.7 in
```

```
d:\code\anaconda3\lib\site-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in d:\code\anaconda3\lib\site-
packages (from yfinance) (5.2.1)
Requirement already satisfied: platformdirs>=2.0.0 in
d:\code\anaconda3\lib\site-packages (from yfinance) (3.10.0)
Requirement already satisfied: pytz>=2022.5 in d:\code\anaconda3\lib\site-
packages (from yfinance) (2024.1)
Requirement already satisfied: frozendict>=2.3.4 in d:\code\anaconda3\lib\site-
packages (from yfinance) (2.4.2)
Requirement already satisfied: peewee>=3.16.2 in d:\code\anaconda3\lib\site-
packages (from yfinance) (3.17.6)
Requirement already satisfied: beautifulsoup4>=4.11.1 in
d:\code\anaconda3\lib\site-packages (from yfinance) (4.12.3)
Requirement already satisfied: html5lib>=1.1 in d:\code\anaconda3\lib\site-
packages (from yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in d:\code\anaconda3\lib\site-
packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: six>=1.9 in d:\code\anaconda3\lib\site-packages
(from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in d:\code\anaconda3\lib\site-
packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
d:\code\anaconda3\lib\site-packages (from pandas>=1.3.0->yfinance) (2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in d:\code\anaconda3\lib\site-
packages (from pandas>=1.3.0->yfinance) (2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
d:\code\anaconda3\lib\site-packages (from requests>=2.31->yfinance) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in d:\code\anaconda3\lib\site-
packages (from requests>=2.31->yfinance) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\code\anaconda3\lib\site-
packages (from requests>=2.31->yfinance) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in d:\code\anaconda3\lib\site-
packages (from requests>=2.31->yfinance) (2024.7.4)
Requirement already satisfied: bs4 in d:\code\anaconda3\lib\site-packages
(0.0.2)
Requirement already satisfied: beautifulsoup4 in d:\code\anaconda3\lib\site-
packages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in d:\code\anaconda3\lib\site-
packages (from beautifulsoup4->bs4) (2.5)
Requirement already satisfied: nbformat in d:\code\anaconda3\lib\site-packages
(5.9.2)
Requirement already satisfied: fastjsonschema in d:\code\anaconda3\lib\site-
packages (from nbformat) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in d:\code\anaconda3\lib\site-
packages (from nbformat) (4.19.2)
Requirement already satisfied: jupyter-core in d:\code\anaconda3\lib\site-
packages (from nbformat) (5.7.2)
Requirement already satisfied: traitlets>=5.1 in d:\code\anaconda3\lib\site-
```

```
packages (from nbformat) (5.14.3)
Requirement already satisfied: attrs>=22.2.0 in d:\code\anaconda3\lib\site-
packages (from jsonschema>=2.6->nbformat) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
d:\code\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in
d:\code\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in d:\code\anaconda3\lib\site-
packages (from jsonschema>=2.6->nbformat) (0.10.6)
Requirement already satisfied: platformdirs>=2.5 in d:\code\anaconda3\lib\site-
packages (from jupyter-core->nbformat) (3.10.0)
Requirement already satisfied: pywin32>=300 in d:\code\anaconda3\lib\site-
packages (from jupyter-core->nbformat) (305.1)
```

[8]:
```
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
```

In Python, you can ignore warnings using the warnings module. You can use the filterwarnings function to filter or ignore specific warning messages or categories.

[10]:
```
import warnings
# Ignore all warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

0.1 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.**

[13]:
```
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),␣
 ↪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),␣
 ↪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()
```

Use the make_graph function that we've already defined. You'll need to invoke it in questions 5 and 6 to display the graphs and create the dashboard. > **Note: You don't need to redefine the function for plotting graphs anywhere else in this notebook; just use the existing function.**

0.2 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`.

[17]:
```
import yfinance as yf
import pandas as pd

tsla = yf.Ticker("TSLA")
tsla.info
```

[17]:
```
{'address1': '1 Tesla Road',
 'city': 'Austin',
 'state': 'TX',
 'zip': '78725',
 'country': 'United States',
 'phone': '512 516 8177',
 'website': 'https://www.tesla.com',
 'industry': 'Auto Manufacturers',
 'industryKey': 'auto-manufacturers',
 'industryDisp': 'Auto Manufacturers',
 'sector': 'Consumer Cyclical',
 'sectorKey': 'consumer-cyclical',
 'sectorDisp': 'Consumer Cyclical',
 'longBusinessSummary': 'Tesla, Inc. designs, develops, manufactures, leases,
and sells electric vehicles, and energy generation and storage systems in the
United States, China, and internationally. The company operates in two segments,
Automotive, and Energy Generation and Storage. The Automotive segment offers
electric vehicles, as well as sells automotive regulatory credits; and non-
warranty after-sales vehicle, used vehicles, body shop and parts, supercharging,
retail merchandise, and vehicle insurance services. This segment also provides
sedans and sport utility vehicles through direct and used vehicle sales, a
network of Tesla Superchargers, and in-app upgrades; purchase financing and
leasing services; services for electric vehicles through its company-owned
```

service locations and Tesla mobile service technicians; and vehicle limited warranties and extended service plans. The Energy Generation and Storage segment engages in the design, manufacture, installation, sale, and leasing of solar energy generation and energy storage products, and related services to residential, commercial, and industrial customers and utilities through its website, stores, and galleries, as well as through a network of channel partners; and provision of service and repairs to its energy product customers, including under warranty, as well as various financing options to its solar customers. The company was formerly known as Tesla Motors, Inc. and changed its name to Tesla, Inc. in February 2017. Tesla, Inc. was incorporated in 2003 and is headquartered in Austin, Texas.',
 'fullTimeEmployees': 140473,
 'companyOfficers': [{'maxAge': 1,
   'name': 'Mr. Elon R. Musk',
   'age': 51,
   'title': 'Co-Founder, Technoking of Tesla, CEO & Director',
   'yearBorn': 1972,
   'fiscalYear': 2023,
   'exercisedValue': 1861335,
   'unexercisedValue': 68433694720},
  {'maxAge': 1,
   'name': 'Mr. Vaibhav Taneja',
   'age': 45,
   'title': 'Chief Financial Officer',
   'yearBorn': 1978,
   'fiscalYear': 2023,
   'totalPay': 278000,
   'exercisedValue': 8517957,
   'unexercisedValue': 202075632},
  {'maxAge': 1,
   'name': 'Mr. Xiaotong Zhu',
   'age': 43,
   'title': 'Senior Vice President of Automotive',
   'yearBorn': 1980,
   'fiscalYear': 2023,
   'totalPay': 926877,
   'exercisedValue': 0,
   'unexercisedValue': 344144320},
  {'maxAge': 1,
   'name': 'Mr. Martin Viecha',
   'title': 'Vice President of Investor Relations',
   'fiscalYear': 2023,
   'exercisedValue': 0,
   'unexercisedValue': 0},
  {'maxAge': 1,
   'name': 'Brian Scelfo',
   'title': 'Senior Director of Corporate Development',

```
  'fiscalYear': 2023,
  'exercisedValue': 0,
  'unexercisedValue': 0},
 {'maxAge': 1,
  'name': 'Mr. Franz  von Holzhausen',
  'title': 'Chief Designer',
  'fiscalYear': 2023,
  'exercisedValue': 0,
  'unexercisedValue': 0},
 {'maxAge': 1,
  'name': 'Mr. John  Walker',
  'age': 60,
  'title': 'Vice President of Sales - North America',
  'yearBorn': 1963,
  'fiscalYear': 2023,
  'totalPay': 121550,
  'exercisedValue': 0,
  'unexercisedValue': 0},
 {'maxAge': 1,
  'name': 'Mr. Peter  Bannon',
  'title': 'Chip Architect',
  'fiscalYear': 2023,
  'exercisedValue': 0,
  'unexercisedValue': 0},
 {'maxAge': 1,
  'name': 'Mr. Turner  Caldwell',
  'title': 'Engineering Manager',
  'fiscalYear': 2023,
  'exercisedValue': 0,
  'unexercisedValue': 0},
 {'maxAge': 1,
  'name': 'Mr. Rodney D. Westmoreland Jr.',
  'title': 'Director of Construction Management',
  'fiscalYear': 2023,
  'exercisedValue': 0,
  'unexercisedValue': 0}],
'auditRisk': 7,
'boardRisk': 9,
'compensationRisk': 10,
'shareHolderRightsRisk': 9,
'overallRisk': 10,
'governanceEpochDate': 1719792000,
'compensationAsOfEpochDate': 1703980800,
'maxAge': 86400,
'priceHint': 2,
'previousClose': 232.1,
'open': 232.25,
```

```
'dayLow': 220.01,
'dayHigh': 232.41,
'regularMarketPreviousClose': 232.1,
'regularMarketOpen': 232.25,
'regularMarketDayLow': 220.01,
'regularMarketDayHigh': 232.41,
'beta': 2.313,
'trailingPE': 62.709858,
'forwardPE': 70.44936,
'volume': 99883197,
'regularMarketVolume': 99883197,
'averageVolume': 95949636,
'averageVolume10days': 114538170,
'averageDailyVolume10Day': 114538170,
'bid': 221.58,
'ask': 222.69,
'bidSize': 400,
'askSize': 100,
'marketCap': 711190708224,
'fiftyTwoWeekLow': 138.8,
'fiftyTwoWeekHigh': 278.98,
'priceToSalesTrailing12Months': 7.4612427,
'fiftyDayAverage': 205.1582,
'twoHundredDayAverage': 204.68265,
'currency': 'USD',
'enterpriseValue': 724066041856,
'profitMargins': 0.12996,
'floatShares': 2777707894,
'sharesOutstanding': 3194639872,
'sharesShort': 104460574,
'sharesShortPriorMonth': 105382772,
'sharesShortPreviousMonthDate': 1718323200,
'dateShortInterest': 1721001600,
'sharesPercentSharesOut': 0.0327,
'heldPercentInsiders': 0.12986,
'heldPercentInstitutions': 0.45926997,
'shortRatio': 0.86,
'shortPercentOfFloat': 0.0375,
'impliedSharesOutstanding': 3194639872,
'bookValue': 20.81,
'priceToBook': 10.6977415,
'lastFiscalYearEnd': 1703980800,
'nextFiscalYearEnd': 1735603200,
'mostRecentQuarter': 1719705600,
'earningsQuarterlyGrowth': -0.453,
'netIncomeToCommon': 12389999616,
'trailingEps': 3.55,
```

```
'forwardEps': 3.16,
'pegRatio': 32.64,
'lastSplitFactor': '3:1',
'lastSplitDate': 1661385600,
'enterpriseToRevenue': 7.596,
'enterpriseToEbitda': 59.457,
'52WeekChange': -0.110966384,
'SandP52WeekChange': 0.19376493,
'exchange': 'NMS',
'quoteType': 'EQUITY',
'symbol': 'TSLA',
'underlyingSymbol': 'TSLA',
'shortName': 'Tesla, Inc.',
'longName': 'Tesla, Inc.',
'firstTradeDateEpochUtc': 1277818200,
'timeZoneFullName': 'America/New_York',
'timeZoneShortName': 'EDT',
'uuid': 'ec367bc4-f92c-397c-ac81-bf7b43cffaf7',
'messageBoardId': 'finmb_27444752',
'gmtOffSetMilliseconds': -14400000,
'currentPrice': 222.62,
'targetHighPrice': 310.0,
'targetLowPrice': 24.86,
'targetMeanPrice': 203.45,
'targetMedianPrice': 212.9,
'recommendationMean': 2.8,
'recommendationKey': 'hold',
'numberOfAnalystOpinions': 42,
'totalCash': 30720000000,
'totalCashPerShare': 9.616,
'ebitda': 12177999872,
'totalDebt': 12515000320,
'quickRatio': 1.249,
'currentRatio': 1.911,
'totalRevenue': 95317999616,
'debtToEquity': 18.606,
'revenuePerShare': 29.932,
'returnOnAssets': 0.044159997,
'returnOnEquity': 0.20861,
'freeCashflow': -907249984,
'operatingCashflow': 11532000256,
'earningsGrowth': -0.462,
'revenueGrowth': 0.023,
'grossMargins': 0.17719999,
'ebitdaMargins': 0.12776,
'operatingMargins': 0.0858,
'financialCurrency': 'USD',
```

```
 'trailingPegRatio': 4.3061}
```

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.

[19]:
```
tesla_data = tsla.history(period="max")
print (tesla_data.head())
```

```
                                 Open      High       Low     Close     Volume  \
Date
2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667  281494500
2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667  257806500
2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000  123282000
2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000   77097000
2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000  103003500

                           Dividends  Stock Splits
Date
2010-06-29 00:00:00-04:00        0.0           0.0
2010-06-30 00:00:00-04:00        0.0           0.0
2010-07-01 00:00:00-04:00        0.0           0.0
2010-07-02 00:00:00-04:00        0.0           0.0
2010-07-06 00:00:00-04:00        0.0           0.0
```

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.

[21]:
```
tesla_data.reset_index(inplace=True)
print (tesla_data.head())
```

```
                       Date      Open      High       Low     Close  \
0 2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667
1 2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667
2 2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000
3 2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000
4 2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000

      Volume  Dividends  Stock Splits
0  281494500        0.0           0.0
1  257806500        0.0           0.0
2  123282000        0.0           0.0
3   77097000        0.0           0.0
4  103003500        0.0           0.0
```

0.3 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`.

[24]:
```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"
html_data = requests.get(url)
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`. Make sure to use the `html_data` with the content parameter as follow `html_data.content` .

[26]:
```
html_data2 = html_data.content
soup = BeautifulSoup(html_data2, 'html.parser')
```

Using `BeautifulSoup` or the `read_html` function extract the table with `Tesla Revenue` and store it into a dataframe named `tesla_revenue`. The dataframe should have columns `Date` and `Revenue`.

Step-by-step instructions

```
Here are the step-by-step instructions:

1. Find All Tables: Start by searching for all HTML tables on a webpage using `soup.find_all('t
2. Identify the Relevant Table: then loops through each table. If a table contains the text "Te
3. Initialize a DataFrame: Create an empty Pandas DataFrame called `tesla_revenue` with columns
4. Loop Through Rows: For each row in the relevant table, extract the data from the first and s
5. Clean Revenue Data: Remove dollar signs and commas from the revenue value.
6. Add Rows to DataFrame: Create a new row in the DataFrame with the extracted date and cleaned
7. Repeat for All Rows: Continue this process for all rows in the table.
```

Click here if you need help locating the table

```
Below is the code to isolate the table, you will now need to loop through the rows and columns

soup.find_all("tbody")[1]

If you want to use the read_html function the table is located at index 1

We are focusing on quarterly revenue in the lab.
> Note: Instead of using the deprecated pd.append() method, consider using pd.concat([df, pd.Da
```

[30]:
```
tables = soup.find_all('table')
relevant_tables = [table for table in tables if "Tesla Quarterly Revenue" in␣
↪table.text]
relevant_tables
```

[30]:
```
[<table class="historical_data_table table">
 <thead>
 <tr>
 <th colspan="2" style="text-align:center">Tesla Quarterly Revenue<br/><span
```

```
style="font-size:14px;">(Millions of US $)</span></th>
 </tr>
 </thead>
 <tbody>
 <tr>
 <td style="text-align:center">2022-09-30</td>
 <td style="text-align:center">$21,454</td>
 </tr>
 <tr>
 <td style="text-align:center">2022-06-30</td>
 <td style="text-align:center">$16,934</td>
 </tr>
 <tr>
 <td style="text-align:center">2022-03-31</td>
 <td style="text-align:center">$18,756</td>
 </tr>
 <tr>
 <td style="text-align:center">2021-12-31</td>
 <td style="text-align:center">$17,719</td>
 </tr>
 <tr>
 <td style="text-align:center">2021-09-30</td>
 <td style="text-align:center">$13,757</td>
 </tr>
 <tr>
 <td style="text-align:center">2021-06-30</td>
 <td style="text-align:center">$11,958</td>
 </tr>
 <tr>
 <td style="text-align:center">2021-03-31</td>
 <td style="text-align:center">$10,389</td>
 </tr>
 <tr>
 <td style="text-align:center">2020-12-31</td>
 <td style="text-align:center">$10,744</td>
 </tr>
 <tr>
 <td style="text-align:center">2020-09-30</td>
 <td style="text-align:center">$8,771</td>
 </tr>
 <tr>
 <td style="text-align:center">2020-06-30</td>
 <td style="text-align:center">$6,036</td>
 </tr>
 <tr>
 <td style="text-align:center">2020-03-31</td>
 <td style="text-align:center">$5,985</td>
```

```
</tr>
<tr>
<td style="text-align:center">2019-12-31</td>
<td style="text-align:center">$7,384</td>
</tr>
<tr>
<td style="text-align:center">2019-09-30</td>
<td style="text-align:center">$6,303</td>
</tr>
<tr>
<td style="text-align:center">2019-06-30</td>
<td style="text-align:center">$6,350</td>
</tr>
<tr>
<td style="text-align:center">2019-03-31</td>
<td style="text-align:center">$4,541</td>
</tr>
<tr>
<td style="text-align:center">2018-12-31</td>
<td style="text-align:center">$7,226</td>
</tr>
<tr>
<td style="text-align:center">2018-09-30</td>
<td style="text-align:center">$6,824</td>
</tr>
<tr>
<td style="text-align:center">2018-06-30</td>
<td style="text-align:center">$4,002</td>
</tr>
<tr>
<td style="text-align:center">2018-03-31</td>
<td style="text-align:center">$3,409</td>
</tr>
<tr>
<td style="text-align:center">2017-12-31</td>
<td style="text-align:center">$3,288</td>
</tr>
<tr>
<td style="text-align:center">2017-09-30</td>
<td style="text-align:center">$2,985</td>
</tr>
<tr>
<td style="text-align:center">2017-06-30</td>
<td style="text-align:center">$2,790</td>
</tr>
<tr>
<td style="text-align:center">2017-03-31</td>
```

```
<td style="text-align:center">$2,696</td>
</tr>
<tr>
<td style="text-align:center">2016-12-31</td>
<td style="text-align:center">$2,285</td>
</tr>
<tr>
<td style="text-align:center">2016-09-30</td>
<td style="text-align:center">$2,298</td>
</tr>
<tr>
<td style="text-align:center">2016-06-30</td>
<td style="text-align:center">$1,270</td>
</tr>
<tr>
<td style="text-align:center">2016-03-31</td>
<td style="text-align:center">$1,147</td>
</tr>
<tr>
<td style="text-align:center">2015-12-31</td>
<td style="text-align:center">$1,214</td>
</tr>
<tr>
<td style="text-align:center">2015-09-30</td>
<td style="text-align:center">$937</td>
</tr>
<tr>
<td style="text-align:center">2015-06-30</td>
<td style="text-align:center">$955</td>
</tr>
<tr>
<td style="text-align:center">2015-03-31</td>
<td style="text-align:center">$940</td>
</tr>
<tr>
<td style="text-align:center">2014-12-31</td>
<td style="text-align:center">$957</td>
</tr>
<tr>
<td style="text-align:center">2014-09-30</td>
<td style="text-align:center">$852</td>
</tr>
<tr>
<td style="text-align:center">2014-06-30</td>
<td style="text-align:center">$769</td>
</tr>
<tr>
```

```
<td style="text-align:center">2014-03-31</td>
<td style="text-align:center">$621</td>
</tr>
<tr>
<td style="text-align:center">2013-12-31</td>
<td style="text-align:center">$615</td>
</tr>
<tr>
<td style="text-align:center">2013-09-30</td>
<td style="text-align:center">$431</td>
</tr>
<tr>
<td style="text-align:center">2013-06-30</td>
<td style="text-align:center">$405</td>
</tr>
<tr>
<td style="text-align:center">2013-03-31</td>
<td style="text-align:center">$562</td>
</tr>
<tr>
<td style="text-align:center">2012-12-31</td>
<td style="text-align:center">$306</td>
</tr>
<tr>
<td style="text-align:center">2012-09-30</td>
<td style="text-align:center">$50</td>
</tr>
<tr>
<td style="text-align:center">2012-06-30</td>
<td style="text-align:center">$27</td>
</tr>
<tr>
<td style="text-align:center">2012-03-31</td>
<td style="text-align:center">$30</td>
</tr>
<tr>
<td style="text-align:center">2011-12-31</td>
<td style="text-align:center">$39</td>
</tr>
<tr>
<td style="text-align:center">2011-09-30</td>
<td style="text-align:center">$58</td>
</tr>
<tr>
<td style="text-align:center">2011-06-30</td>
<td style="text-align:center">$58</td>
</tr>
```

```
<tr>
<td style="text-align:center">2011-03-31</td>
<td style="text-align:center">$49</td>
</tr>
<tr>
<td style="text-align:center">2010-12-31</td>
<td style="text-align:center">$36</td>
</tr>
<tr>
<td style="text-align:center">2010-09-30</td>
<td style="text-align:center">$31</td>
</tr>
<tr>
<td style="text-align:center">2010-06-30</td>
<td style="text-align:center">$28</td>
</tr>
<tr>
<td style="text-align:center">2010-03-31</td>
<td style="text-align:center">$21</td>
</tr>
<tr>
<td style="text-align:center">2009-12-31</td>
<td style="text-align:center"></td>
</tr>
<tr>
<td style="text-align:center">2009-09-30</td>
<td style="text-align:center">$46</td>
</tr>
<tr>
<td style="text-align:center">2009-06-30</td>
<td style="text-align:center">$27</td>
</tr>
</tbody>
</table>]
```

[31]:
```
tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])

tesla_revenue
```

[31]:
```
Empty DataFrame
Columns: [Date, Revenue]
Index: []
```

[32]:
```
for table in relevant_tables:
    for row in table.tbody.find_all('tr'):
        columns = row.find_all('td')
        if columns:
```

```
            date = columns[0].text.strip()
            revenue = columns[1].text.strip()
            new_row = pd.DataFrame({"Date": [date], "Revenue": [revenue]})
            tesla_revenue = pd.concat([tesla_revenue, new_row],␣
 ↪ignore_index=True)
print(tesla_revenue)
```

```
          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
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.

[34]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\\$',"",␣
↪regex=True)
```

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

[36]:
```
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.

[38]:
```
tesla_revenue.head()
```

[38]:

	Date	Revenue
0	2022-09-30	21454
1	2022-06-30	16934
2	2022-03-31	18756
3	2021-12-31	17719
4	2021-09-30	13757

0.4 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`.

[41]:
```
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.

[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.

[45]:
```
gme_data.reset_index(inplace=True)
print (gme_data.head())
```

```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620129  1.693350  1.603296  1.691667  76216000
1 2002-02-14 00:00:00-05:00  1.712708  1.716074  1.670626  1.683251  11021600
2 2002-02-15 00:00:00-05:00  1.683251  1.687459  1.658002  1.674834   8389600
3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504   7410400
4 2002-02-20 00:00:00-05:00  1.615920  1.662210  1.603296  1.662210   6892800

   Dividends  Stock Splits
0        0.0           0.0
1        0.0           0.0
2        0.0           0.0
3        0.0           0.0
4        0.0           0.0
```

0.5 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_2`.

[48]:
```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
  ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"
html_data_2 = requests.get(url).text
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`.

[50]:
```
soup_2 = BeautifulSoup(html_data_2, 'html.parser')
```

Using `BeautifulSoup` or the `read_html` function extract the table with `GameStop 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.

Note: Use the method similar to what you did in question 2.

Click here if you need help locating the table

```
Below is the code to isolate the table, you will now need to loop through the rows and columns

soup.find_all("tbody")[1]

If you want to use the read_html function the table is located at index 1
```

[54]:
```
tables_2 = soup_2.find_all('table')
relevant_tables_2 = []
for table in tables_2:
    if "GameStop Revenue" in table.text:
        relevant_tables.append(table)
relevant_tables_2
```

[54]:
```
[]
```

[55]:
```
gme_revenue = pd.DataFrame(columns=["Date", "Revenue"])

for table in relevant_tables:
    for row in table.tbody.find_all('tr'):
        columns = row.find_all('td')
        if columns:
            date = columns[0].text.strip()
            revenue = columns[1].text.strip()
            new_row = pd.DataFrame({"Date": [date], "Revenue": [revenue]})
            gme_revenue = pd.concat([gme_revenue, new_row], ignore_index=True)

gme_revenue['Revenue'] = gme_revenue['Revenue'].str.replace(',', '').str.
 ↪replace('$', '')
gme_revenue.dropna(inplace=True)
gme_revenue = gme_revenue[gme_revenue['Revenue'] != ""]

print(gme_revenue.head())
```

```
         Date Revenue
0  2022-09-30   21454
1  2022-06-30   16934
2  2022-03-31   18756
3  2021-12-31   17719
4  2021-09-30   13757
```

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

[57]:
```
print(gme_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
```

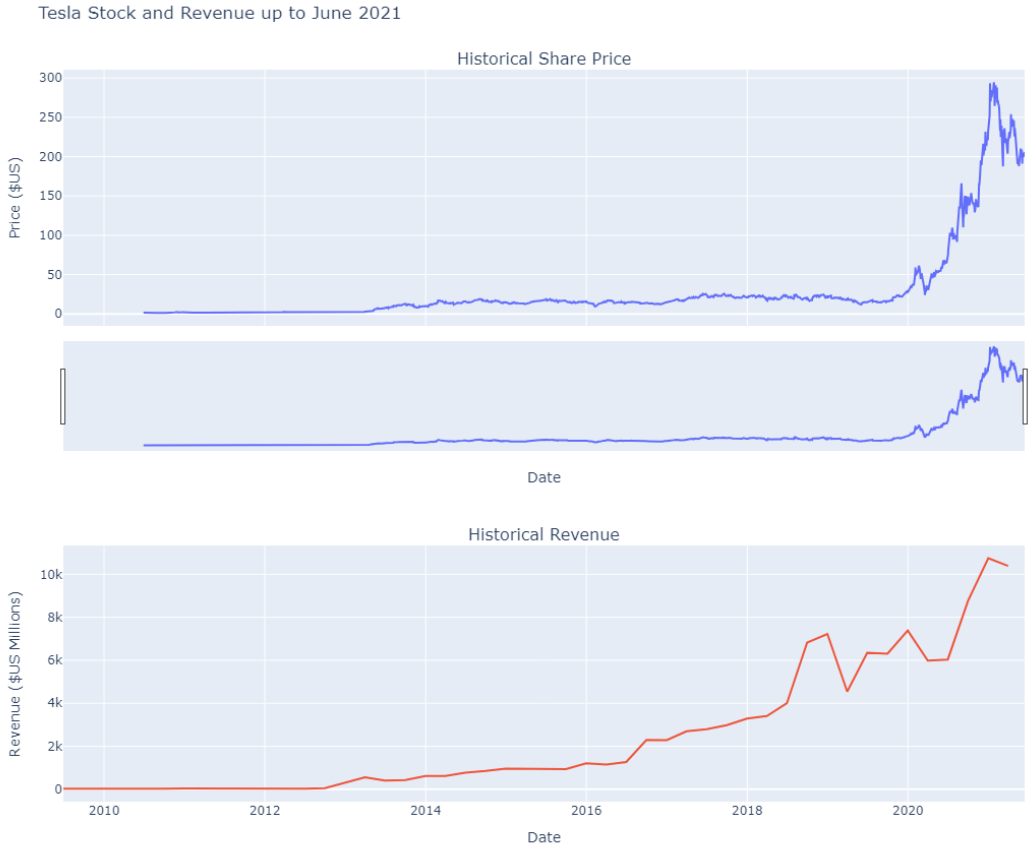
0.6 Question 5: Plot Tesla Stock Graph

Use the `make_graph` function to graph the Tesla Stock Data, also provide a title for the graph. Note the graph will only show data upto June 2021.

Hint

`You just need to invoke the make_graph function with the required parameter to print the graphs`

[61]:
```
tesla_data.index = pd.to_datetime(tesla_data.index)
tesla_data_filtered = tesla_data[tesla_data.index <= '2021-06-30']
make_graph(tesla_data_filtered, tesla_revenue, 'Tesla Stock and Revenue up to␣
↪June 2021')
```



0.7 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.

Hint

You just need to invoke the make_graph function with the required parameter to print the graphs

[65]:
```
gme_data.index = pd.to_datetime(gme_data.index)
gme_data_filtered = gme_data[gme_data.index <= '2021-06-30']
make_graph(gme_data_filtered, gme_revenue, 'GameStop Stock and Revenue up to␣
↪June 2021')
```

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.

##


```
toggle ## Change Log          toggle | Date (YYYY-MM-DD) | Version | Changed By     |
Change Description          | toggle | ----------------- | ------- | ------------- |
```

```
------------------------- | toggle | 2022-02-28        | 1.2     | Lakshmi Holla |
Changed the URL of GameStop | toggle | 2020-11-10        | 1.1     | Malika Singla
| Deleted the Optional part | toggle | 2020-08-27        | 1.0     | Malika Singla
| Added lab to GitLab       |
```

[ ]: