Final Assignment

March 14, 2025

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. Use the version as per your python version.

[3]:
```
!pip install yfinance
!pip install bs4
!pip install nbformat
!pip install --upgrade plotly
```

```
Requirement already satisfied: yfinance in /opt/conda/lib/python3.12/site-
packages (0.2.54)
Requirement already satisfied: pandas>=1.3.0 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.2.3)
Requirement already satisfied: numpy>=1.16.5 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.2.3)
Requirement already satisfied: requests>=2.31 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (0.0.11)
Requirement already satisfied: platformdirs>=2.0.0 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.3.6)
```

```
Requirement already satisfied: pytz>=2022.5 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2024.2)
Requirement already satisfied: frozendict>=2.3.4 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (3.17.9)
Requirement already satisfied: beautifulsoup4>=4.11.1 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-
packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.12/site-packages (from pandas>=1.3.0->yfinance)
(2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-
packages (from pandas>=1.3.0->yfinance) (2025.1)
Requirement already satisfied: charset_normalizer<4,>=2 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.12/site-
packages (from requests>=2.31->yfinance) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance)
(2024.12.14)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance) (1.17.0)
Requirement already satisfied: bs4 in /opt/conda/lib/python3.12/site-packages
(0.0.2)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.12/site-
packages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-
packages (from beautifulsoup4->bs4) (2.5)
Requirement already satisfied: nbformat in /opt/conda/lib/python3.12/site-
packages (5.10.4)
Requirement already satisfied: fastjsonschema>=2.15 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (2.21.1)
Requirement already satisfied: jsonschema>=2.6 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (4.23.0)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (5.7.2)
Requirement already satisfied: traitlets>=5.1 in /opt/conda/lib/python3.12/site-
packages (from nbformat) (5.14.3)
Requirement already satisfied: attrs>=22.2.0 in /opt/conda/lib/python3.12/site-
packages (from jsonschema>=2.6->nbformat) (25.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/opt/conda/lib/python3.12/site-packages (from jsonschema>=2.6->nbformat)
(2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in
```

```
/opt/conda/lib/python3.12/site-packages (from jsonschema>=2.6->nbformat)
(0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in /opt/conda/lib/python3.12/site-
packages (from jsonschema>=2.6->nbformat) (0.22.3)
Requirement already satisfied: platformdirs>=2.5 in
/opt/conda/lib/python3.12/site-packages (from jupyter-
core!=5.0.*,>=4.12->nbformat) (4.3.6)
Requirement already satisfied: typing-extensions>=4.4.0 in
/opt/conda/lib/python3.12/site-packages (from
referencing>=0.28.4->jsonschema>=2.6->nbformat) (4.12.2)
Requirement already satisfied: plotly in /opt/conda/lib/python3.12/site-packages
(6.0.0)
Requirement already satisfied: narwhals>=1.15.1 in
/opt/conda/lib/python3.12/site-packages (from plotly) (1.30.0)
Requirement already satisfied: packaging in /opt/conda/lib/python3.12/site-
packages (from plotly) (24.2)
```

[5]:
```
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
```

[6]:
```
import plotly.io as pio
pio.renderers.default = "iframe"
```

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.

[ ]:
```
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.**

[ ]:
```
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()
    from IPython.display import display, HTML
    fig_html = fig.to_html()
    display(HTML(fig_html))
```

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`.

[12]:
```
!pip install yfinance --upgrade --no-cache-dir
import yfinance as yf

# Create a ticker object for Tesla (TSLA)
tesla_ticker = yf.Ticker("TSLA")

# Display the object to check the data
tesla_ticker.info
```

```
Requirement already satisfied: yfinance in /opt/conda/lib/python3.12/site-
packages (0.2.54)
Requirement already satisfied: pandas>=1.3.0 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.2.3)
Requirement already satisfied: numpy>=1.16.5 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.2.3)
Requirement already satisfied: requests>=2.31 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (0.0.11)
```

```
Requirement already satisfied: platformdirs>=2.0.0 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.3.6)
Requirement already satisfied: pytz>=2022.5 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2024.2)
Requirement already satisfied: frozendict>=2.3.4 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (3.17.9)
Requirement already satisfied: beautifulsoup4>=4.11.1 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-
packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.12/site-packages (from pandas>=1.3.0->yfinance)
(2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-
packages (from pandas>=1.3.0->yfinance) (2025.1)
Requirement already satisfied: charset_normalizer<4,>=2 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.12/site-
packages (from requests>=2.31->yfinance) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance)
(2024.12.14)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance) (1.17.0)
```

[12]:

```
{'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. This segment also provides services and repairs to its energy product customers, including under warranty; and various financing options to its residential 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': 125665,
 'companyOfficers': [{'maxAge': 1,
   'name': 'Mr. Elon R. Musk',
   'age': 53,
   'title': 'Co-Founder, Technoking of Tesla, CEO & Director',
   'yearBorn': 1971,
   'fiscalYear': 2023,
   'exercisedValue': 0,
   'unexercisedValue': 0},
  {'maxAge': 1,
   'name': 'Mr. Vaibhav Taneja',
   'age': 46,
   'title': 'Chief Financial Officer',
   'yearBorn': 1978,
   'fiscalYear': 2023,
   'totalPay': 278000,
   'exercisedValue': 8517957,
   'unexercisedValue': 202075632},
  {'maxAge': 1,
   'name': 'Mr. Xiaotong Zhu',
   'age': 44,
   'title': 'Senior Vice President of Automotive',
   'yearBorn': 1980,
   'fiscalYear': 2023,
   'totalPay': 926877,
   'exercisedValue': 0,
   'unexercisedValue': 344144320},
  {'maxAge': 1,
   'name': 'Travis Axelrod',
   'title': 'Head 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': 61,
  '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': 5,
'boardRisk': 9,
'compensationRisk': 10,
'shareHolderRightsRisk': 9,
'overallRisk': 10,
'governanceEpochDate': 1740787200,
```

```
'compensationAsOfEpochDate': 1703980800,
'executiveTeam': [],
'maxAge': 86400,
'priceHint': 2,
'previousClose': 248.09,
'open': 248.125,
'dayLow': 233.53,
'dayHigh': 248.29,
'regularMarketPreviousClose': 248.09,
'regularMarketOpen': 248.125,
'regularMarketDayLow': 233.53,
'regularMarketDayHigh': 248.29,
'payoutRatio': 0.0,
'beta': 2.507,
'trailingPE': 117.40488,
'forwardPE': 74.28395,
'volume': 113936193,
'regularMarketVolume': 113936193,
'averageVolume': 90317664,
'averageVolume10days': 127330500,
'averageDailyVolume10Day': 127330500,
'bid': 236.6,
'ask': 236.81,
'bidSize': 1,
'askSize': 1,
'marketCap': 774151995392,
'fiftyTwoWeekLow': 138.8,
'fiftyTwoWeekHigh': 488.54,
'priceToSalesTrailing12Months': 7.9245777,
'fiftyDayAverage': 354.9698,
'twoHundredDayAverage': 282.32556,
'trailingAnnualDividendRate': 0.0,
'trailingAnnualDividendYield': 0.0,
'currency': 'USD',
'tradeable': False,
'enterpriseValue': 751978348544,
'profitMargins': 0.07259,
'floatShares': 2799688594,
'sharesOutstanding': 3216519936,
'sharesShort': 67129145,
'sharesShortPriorMonth': 59612004,
'sharesShortPreviousMonthDate': 1738281600,
'dateShortInterest': 1740700800,
'sharesPercentSharesOut': 0.0209,
'heldPercentInsiders': 0.12887,
'heldPercentInstitutions': 0.49293,
'shortRatio': 0.82,
```

```
'shortPercentOfFloat': 0.0239,
'impliedSharesOutstanding': 3216519936,
'bookValue': 22.672,
'priceToBook': 10.615737,
'lastFiscalYearEnd': 1735603200,
'nextFiscalYearEnd': 1767139200,
'mostRecentQuarter': 1735603200,
'earningsQuarterlyGrowth': -0.708,
'netIncomeToCommon': 7129999872,
'trailingEps': 2.05,
'forwardEps': 3.24,
'lastSplitFactor': '3:1',
'lastSplitDate': 1661385600,
'enterpriseToRevenue': 7.698,
'enterpriseToEbitda': 57.725,
'52WeekChange': 0.47141898,
'SandP52WeekChange': 0.07903516,
'quoteType': 'EQUITY',
'currentPrice': 240.68,
'targetHighPrice': 550.0,
'targetLowPrice': 120.0,
'targetMeanPrice': 343.39215,
'targetMedianPrice': 376.5,
'recommendationMean': 2.64583,
'recommendationKey': 'hold',
'numberOfAnalystOpinions': 42,
'totalCash': 36563001344,
'totalCashPerShare': 11.367,
'ebitda': 13027000320,
'totalDebt': 13623000064,
'quickRatio': 1.427,
'currentRatio': 2.025,
'totalRevenue': 97690001408,
'debtToEquity': 18.489,
'revenuePerShare': 30.557,
'returnOnAssets': 0.04186,
'returnOnEquity': 0.1042,
'grossProfits': 17450000384,
'freeCashflow': -826875008,
'operatingCashflow': 14922999808,
'earningsGrowth': -0.709,
'revenueGrowth': 0.021,
'grossMargins': 0.17863001,
'ebitdaMargins': 0.13335,
'operatingMargins': 0.06158,
'financialCurrency': 'USD',
'symbol': 'TSLA',
```

```
'language': 'en-US',
'region': 'US',
'typeDisp': 'Equity',
'quoteSourceName': 'Nasdaq Real Time Price',
'triggerable': True,
'customPriceAlertConfidence': 'HIGH',
'marketState': 'PREPRE',
'messageBoardId': 'finmb_27444752',
'exchangeTimezoneName': 'America/New_York',
'exchangeTimezoneShortName': 'EDT',
'gmtOffSetMilliseconds': -14400000,
'market': 'us_market',
'esgPopulated': False,
'regularMarketChangePercent': -2.9868207,
'regularMarketPrice': 240.68,
'shortName': 'Tesla, Inc.',
'longName': 'Tesla, Inc.',
'exchange': 'NMS',
'corporateActions': [],
'postMarketTime': 1741910399,
'regularMarketTime': 1741896000,
'earningsTimestamp': 1738184940,
'earningsTimestampStart': 1745438400,
'earningsTimestampEnd': 1746302400,
'earningsCallTimestampStart': 1738189800,
'earningsCallTimestampEnd': 1738189800,
'isEarningsDateEstimate': True,
'epsTrailingTwelveMonths': 2.05,
'epsForward': 3.24,
'epsCurrentYear': 2.75035,
'priceEpsCurrentYear': 87.50886,
'fiftyDayAverageChange': -114.289795,
'fiftyDayAverageChangePercent': -0.3219705,
'twoHundredDayAverageChange': -41.64557,
'twoHundredDayAverageChangePercent': -0.14750902,
'sourceInterval': 15,
'exchangeDataDelayedBy': 0,
'averageAnalystRating': '2.6 - Hold',
'cryptoTradeable': False,
'hasPrePostMarketData': True,
'firstTradeDateMilliseconds': 1277818200000,
'postMarketChangePercent': 0.174511,
'postMarketPrice': 241.1,
'postMarketChange': 0.420013,
'regularMarketChange': -7.4100037,
'regularMarketDayRange': '233.53 - 248.29',
'fullExchangeName': 'NasdaqGS',
```

```
 'averageDailyVolume3Month': 90317664,
 'fiftyTwoWeekLowChange': 101.87999,
 'fiftyTwoWeekLowChangePercent': 0.7340057,
 'fiftyTwoWeekRange': '138.8 - 488.54',
 'fiftyTwoWeekHighChange': -247.86002,
 'fiftyTwoWeekHighChangePercent': -0.5073485,
 'fiftyTwoWeekChangePercent': 47.1419,
 'displayName': 'Tesla',
 'trailingPegRatio': 3.5311}
```

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.

[13]:
```
import yfinance as yf

# Create a ticker object for Tesla (TSLA)
tesla_ticker = yf.Ticker("TSLA")

# Extract stock data for Tesla using the history function and set the period to␣
 ↪"max"
tesla_data = tesla_ticker.history(period="max")

# Display the first few rows of the dataframe to verify
tesla_data.head()
```

[13]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2010-06-29 00:00:00-04:00	1.266667	1.666667	1.169333	1.592667	281494500	0.0	0.0
2010-06-30 00:00:00-04:00	1.719333	2.028000	1.553333	1.588667	257806500	0.0	0.0
2010-07-01 00:00:00-04:00	1.666667	1.728000	1.351333	1.464000	123282000	0.0	0.0
2010-07-02 00:00:00-04:00	1.533333	1.540000	1.247333	1.280000	77097000	0.0	0.0
2010-07-06 00:00:00-04:00	1.333333	1.333333	1.055333	1.074000	103003500	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.

[14]:
```
import yfinance as yf
```

```
# Create a ticker object for Tesla (TSLA)
tesla_ticker = yf.Ticker("TSLA")

# Extract stock data for Tesla using the history function and set the period to␣
 ↪"max"
tesla_data = tesla_ticker.history(period="max")

# Reset the index
tesla_data.reset_index(inplace=True)

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

[14]:

	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`.

[15]:
```
import requests

# URL for the webpage
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"

# Use the requests library to download the webpage
response = requests.get(url)

# Save the text of the response as a variable named html_data
html_data = response.text

# Display the first 500 characters of the html_data to verify
```

```
print(html_data[:500])  # Display the first 500 characters of the HTML content
```

```
<!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/stocks/charts/TSLA/tesla/revenue" />
```

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

[16]:
```
from bs4 import BeautifulSoup

# Parse the html_data using BeautifulSoup and the 'html.parser'
soup = BeautifulSoup(html_data, 'html.parser')

# Display the parsed HTML
print(soup.prettify()[:500])  # Display the first 500 characters of the␣
 ↪prettified HTML
```

```
<!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 content="IE=edge,chrome=1" http-equiv="X-UA-Compatible"/>
  <link href="https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue"
rel="canonical"/>
  <title>
   Te
```

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. Create an Empty DataFrame

2. Find the Relevant Table
3. Check for the Tesla Quarterly Revenue Table
4. Iterate Through Rows in the Table Body
5. Extract Data from Columns
6. Append Data to the DataFrame

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.

[18]:
```
       import pandas as pd
from bs4 import BeautifulSoup

# Parse the html_data using BeautifulSoup and 'html.parser'
soup = BeautifulSoup(html_data, 'html.parser')

# Find the table containing the Tesla Revenue
table = soup.find('table')  # Assuming the table is the first one on the page

# Extract the data into a list of rows
rows = table.find_all('tr')

# Create lists for the data
dates = []
revenues = []

# Loop through rows to extract date and revenue
for row in rows[1:]:  # Skip the header row
    cols = row.find_all('td')
    dates.append(cols[0].text.strip())  # Extract date
    revenues.append(cols[1].text.strip())  # Extract revenue

# Create a DataFrame from the extracted data
tesla_revenue = pd.DataFrame({
    'Date': dates,
    'Revenue': revenues
})

# Display the first few rows of the dataframe
tesla_revenue.head()
```

[18]:

	Date	Revenue
0	2021	$53,823
1	2020	$31,536
2	2019	$24,578
3	2018	$21,461
4	2017	$11,759

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

[ ]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].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.

[19]:
```
# Display the last 5 rows of the tesla_revenue DataFrame
tesla_revenue.tail()
```

[19]:

	Date	Revenue
8	2013	$2,013
9	2012	$413
10	2011	$204
11	2010	$117
12	2009	$112

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`.

[22]:
```
import yfinance as yf

# Create a ticker object for GameStop (GME)
gme = yf.Ticker("GME")
# Get historical market data for GameStop
gme_data = gme.history(period="max")
```

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.

[23]:
```
import yfinance as yf

# Create a ticker object for GameStop (GME)
gme = yf.Ticker("GME")
```

```
# Fetch historical stock data for the maximum available period
gme_data = gme.history(period="max")

# Display the first few rows of the data to verify
print(gme_data.head())
```

```
                                 Open      High       Low     Close    Volume  \
Date
2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691667  76216000
2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250  11021600
2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658002  1.674834   8389600
2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504   7410400
2002-02-20 00:00:00-05:00  1.615921  1.662210  1.603296  1.662210   6892800

                           Dividends  Stock Splits
Date
2002-02-13 00:00:00-05:00        0.0           0.0
2002-02-14 00:00:00-05:00        0.0           0.0
2002-02-15 00:00:00-05:00        0.0           0.0
2002-02-19 00:00:00-05:00        0.0           0.0
2002-02-20 00:00:00-05:00        0.0           0.0
```

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.

[24]:
```
# Reset the index of gme_data
gme_data.reset_index(inplace=True)

# Display the first 5 rows of gme_data
print(gme_data.head())
```

```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691667  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250  11021600
2 2002-02-15 00:00:00-05:00  1.683250  1.687458  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  1.615921  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`.

[25]:
```
import requests

# URL of the webpage to download
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html'

# Send a GET request to the URL
response = requests.get(url)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Save the HTML content to html_data_2
    html_data_2 = response.text
else:
    print(f"Failed to retrieve the webpage. Status code: {response.
 ↪status_code}")
```

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

[26]:
```
from bs4 import BeautifulSoup
soup = 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
```

[27]:
```
import pandas as pd
import requests
from bs4 import BeautifulSoup

# Step 1: Fetch the HTML content from the URL
```

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

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(html_data_2, 'html.parser')

# Step 3: Extract the revenue data from the HTML table
data = []
table = soup.find('table')  # Assuming the first table contains the revenue data
for row in table.find_all('tr')[1:]:  # Skip the header row
    cols = row.find_all('td')
    if len(cols) > 1:
        date = cols[0].text.strip()
        revenue = cols[1].text.strip()
        data.append([date, revenue])

# Step 4: Create a DataFrame from the extracted data
gme_revenue = pd.DataFrame(data, columns=["Date", "Revenue"])

# Step 5: Clean the 'Revenue' column by removing dollar signs and commas
gme_revenue['Revenue'] = gme_revenue['Revenue'].replace({'\$': '', ',': ''},␣
 ↪regex=True)

# Step 6: Convert the 'Revenue' column to numeric values
gme_revenue['Revenue'] = pd.to_numeric(gme_revenue['Revenue'], errors='coerce')

# Step 7: Handle any missing or invalid data
gme_revenue = gme_revenue.dropna(subset=['Revenue'])

# Step 8: Display the first five rows of the DataFrame
print(gme_revenue.head())
```

```
   Date  Revenue
0  2020     6466
1  2019     8285
2  2018     8547
3  2017     7965
4  2016     9364
```

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

[28]:
```
gme_revenue.tail()
```

[28]:
```
    Date  Revenue
11  2009     8806
```

```
12  2008      7094
13  2007      5319
14  2006      3092
15  2005      1843
```

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
```

[29]:
```
import pandas as pd
import yfinance as yf
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Fetch Tesla stock data up to June 2021
tesla_data = yf.download('TSLA', start='2020-01-01', end='2021-06-30').
 ↪reset_index()

# Define Tesla's quarterly revenue data
revenue_dates = pd.date_range(start='2020-01-01', periods=7, freq='Q')
revenue_values = [10, 12, 14, 16, 18, 20, 22]  # Example values

tesla_revenue = pd.DataFrame({'Date': revenue_dates, 'Revenue': revenue_values})
tesla_revenue['Date'] = pd.to_datetime(tesla_revenue['Date'])

# Function to plot stock and revenue data
def make_graph(stock_data, revenue_data, company_name):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, vertical_spacing=0.1,
                        subplot_titles=(f'{company_name} Stock Price',␣
 ↪f'{company_name} Revenue'),
                        row_heights=[0.7, 0.3])

    fig.add_trace(go.Candlestick(x=stock_data['Date'],
                                 open=stock_data['Open'],
                                 high=stock_data['High'],
                                 low=stock_data['Low'],
                                 close=stock_data['Close'],
                                 name=f'{company_name} Stock Price'),
                  row=1, col=1)

    fig.add_trace(go.Bar(x=revenue_data['Date'],
                         y=revenue_data['Revenue'],
```

```
                        name=f'{company_name} Revenue'),
                  row=2, col=1)

    fig.update_layout(title=f'{company_name} Stock Price and Revenue',
                      xaxis_title='Date',
                      yaxis_title='Stock Price (USD)',
                      yaxis2_title='Revenue (USD)',
                      xaxis_rangeslider_visible=False)
    fig.show()

# Plot Tesla's stock and revenue data
make_graph(tesla_data, tesla_revenue, 'Tesla')
```

```
[*********************100%**********************]  1 of 1 completed

YF.download() has changed argument auto_adjust default to True


/tmp/ipykernel_2512/2003233015.py:10: FutureWarning:

'Q' is deprecated and will be removed in a future version, please use 'QE'
instead.
```

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
```

[30]:
```
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

0.8 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

##