IBM Data Science Professional Certificate

Course 5: Python Project for Data Science

Project Overview

Goal:

- Assume the role of data scientist at a startup investment firm.
- Extract financial data like historical share price and quarterly revenue reporting from various sources using Python libraries and webscraping.
- Visualize this data to identify patterns or trends.

Tasks:

- Complete 2 labs to help gather and prep the required data.
- Complete the final analysis and visualization.

LAB 1: Extracting Stock using a Python Library

Extract stock data using the Python yfinance Library, which allows for extracting data and returning stock data in dataframes.

In [2]:
```
# install needed library
# !pip install yfinance
```

```
Requirement already satisfied: yfinance in c:\users\orgil\appdata\local\programs\python
\python39\lib\site-packages (0.1.59)
Requirement already satisfied: pandas>=0.24 in c:\users\orgil\appdata\local\programs\pyt
hon\python39\lib\site-packages (from yfinance) (1.2.4)
Requirement already satisfied: numpy>=1.15 in c:\users\orgil\appdata\local\programs\pyth
on\python39\lib\site-packages (from yfinance) (1.20.3)
Requirement already satisfied: requests>=2.20 in c:\users\orgil\appdata\local\programs\p
ython\python39\lib\site-packages (from yfinance) (2.25.1)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\orgil\appdata\local\progr
ams\python\python39\lib\site-packages (from yfinance) (0.0.9)
Requirement already satisfied: lxml>=4.5.1 in c:\users\orgil\appdata\local\programs\pyth
on\python39\lib\site-packages (from yfinance) (4.6.3)
Requirement already satisfied: pytz>=2017.3 in c:\users\orgil\appdata\local\programs\pyt
hon\python39\lib\site-packages (from pandas>=0.24->yfinance) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\orgil\appdata\local\pr
ograms\python\python39\lib\site-packages (from pandas>=0.24->yfinance) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\users\orgil\appdata\local\programs\python
\python39\lib\site-packages (from python-dateutil>=2.7.3->pandas>=0.24->yfinance) (1.16.
0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\orgil\appdata\local\pro
grams\python\python39\lib\site-packages (from requests>=2.20->yfinance) (1.26.4)
Requirement already satisfied: idna<3,>=2.5 in c:\users\orgil\appdata\local\programs\pyt
hon\python39\lib\site-packages (from requests>=2.20->yfinance) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\orgil\appdata\local\program
s\python\python39\lib\site-packages (from requests>=2.20->yfinance) (4.0.0)
```

```
Requirement already satisfied: certifi>=2017.4.17 in c:\users\orgil\appdata\local\progra
ms\python\python39\lib\site-packages (from requests>=2.20->yfinance) (2020.12.5)
```

In [3]:
```
# import libraries
import yfinance as yf
import pandas as pd
```

In [4]:
```
# use the ticker module to create an object that will allow us to access functions & ex

# apple stock object
apple = yf.Ticker('AAPL')
```

In [11]:
```
# view stock information in a dataframe
apple_info = apple.info
apple_info
```

Out[11]:
```
{'zip': '95014',
 'sector': 'Technology',
 'fullTimeEmployees': 100000,
 'longBusinessSummary': 'Apple Inc. designs, manufactures, and markets smartphones, pers
onal computers, tablets, wearables, and accessories worldwide. It also sells various rel
ated services. The company offers iPhone, a line of smartphones; Mac, a line of personal
computers; iPad, a line of multi-purpose tablets; and wearables, home, and accessories c
omprising AirPods, Apple TV, Apple Watch, Beats products, HomePod, iPod touch, and other
Apple-branded and third-party accessories. It also provides AppleCare support services;
cloud services store services; and operates various platforms, including the App Store,
that allow customers to discover and download applications and digital content, such as
books, music, video, games, and podcasts. In addition, the company offers various servic
es, such as Apple Arcade, a game subscription service; Apple Music, which offers users a
curated listening experience with on-demand radio stations; Apple News+, a subscription
news and magazine service; Apple TV+, which offers exclusive original content; Apple Car
d, a co-branded credit card; and Apple Pay, a cashless payment service, as well as licen
ses its intellectual property. The company serves consumers, and small and mid-sized bus
inesses; and the education, enterprise, and government markets. It sells and delivers th
ird-party applications for its products through the App Store. The company also sells it
s products through its retail and online stores, and direct sales force; and third-party
cellular network carriers, wholesalers, retailers, and resellers. Apple Inc. was founded
in 1977 and is headquartered in Cupertino, California.',
 'city': 'Cupertino',
 'phone': '408-996-1010',
 'state': 'CA',
 'country': 'United States',
 'companyOfficers': [],
 'website': 'http://www.apple.com',
 'maxAge': 1,
 'address1': 'One Apple Park Way',
 'industry': 'Consumer Electronics',
 'previousClose': 130.46,
 'regularMarketOpen': 130.3,
 'twoHundredDayAverage': 128.47574,
 'trailingAnnualDividendYield': 0.006285451,
 'payoutRatio': 0.1834,
 'volume24Hr': None,
 'regularMarketDayHigh': 132.19,
 'navPrice': None,
 'averageDailyVolume10Day': 85083200,
 'totalAssets': None,
 'regularMarketPreviousClose': 130.46,
 'fiftyDayAverage': 127.064705,
 'trailingAnnualDividendRate': 0.82,
```

```
'open': 130.3,
'toCurrency': None,
'averageVolume10days': 85083200,
'expireDate': None,
'yield': None,
'algorithm': None,
'dividendRate': 0.88,
'exDividendDate': 1620345600,
'beta': 1.208152,
'circulatingSupply': None,
'startDate': None,
'regularMarketDayLow': 129.2118,
'priceHint': 2,
'currency': 'USD',
'trailingPE': 29.66959,
'regularMarketVolume': 40544850,
'lastMarket': None,
'maxSupply': None,
'openInterest': None,
'marketCap': 2202763264000,
'volumeAllCurrencies': None,
'strikePrice': None,
'averageVolume': 86313849,
'priceToSalesTrailing12Months': 6.7692766,
'dayLow': 129.2118,
'ask': 132.09,
'ytdReturn': None,
'askSize': 1300,
'volume': 40544850,
'fiftyTwoWeekHigh': 145.09,
'forwardPE': 24.672897,
'fromCurrency': None,
'fiveYearAvgDividendYield': 1.34,
'fiftyTwoWeekLow': 87.7875,
'bid': 132.08,
'tradeable': False,
'dividendYield': 0.0068,
'bidSize': 1100,
'dayHigh': 132.19,
'exchange': 'NMS',
'shortName': 'Apple Inc.',
'longName': 'Apple Inc.',
'exchangeTimezoneName': 'America/New_York',
'exchangeTimezoneShortName': 'EDT',
'isEsgPopulated': False,
'gmtOffSetMilliseconds': '-14400000',
'quoteType': 'EQUITY',
'symbol': 'AAPL',
'messageBoardId': 'finmb_24937',
'market': 'us_market',
'annualHoldingsTurnover': None,
'enterpriseToRevenue': 6.874,
'beta3Year': None,
'profitMargins': 0.23451,
'enterpriseToEbitda': 22.408,
'52WeekChange': 0.45411992,
'morningStarRiskRating': None,
'forwardEps': 5.35,
'revenueQuarterlyGrowth': None,
'sharesOutstanding': 16687599616,
'fundInceptionDate': None,
'annualReportExpenseRatio': None,
'bookValue': 4.146,
'sharesShort': 123121920,
'sharesPercentSharesOut': 0.0074,
```

```
'fundFamily': None,
'lastFiscalYearEnd': 1601078400,
'heldPercentInstitutions': 0.58687997,
'netIncomeToCommon': 76311003136,
'trailingEps': 4.449,
'lastDividendValue': 0.22,
'SandP52WeekChange': 0.33631718,
'priceToBook': 31.83671,
'heldPercentInsiders': 0.00066,
'nextFiscalYearEnd': 1664150400,
'mostRecentQuarter': 1616803200,
'shortRatio': 1.36,
'sharesShortPreviousMonthDate': 1619740800,
'floatShares': 16670609616,
'enterpriseValue': 2236806070272,
'threeYearAverageReturn': None,
'lastSplitDate': 1598832000,
'lastSplitFactor': '4:1',
'legalType': None,
'lastDividendDate': 1620345600,
'morningStarOverallRating': None,
'earningsQuarterlyGrowth': 1.101,
'dateShortInterest': 1622160000,
'pegRatio': 1.45,
'lastCapGain': None,
'shortPercentOfFloat': 0.0074,
'sharesShortPriorMonth': 82710348,
'impliedSharesOutstanding': None,
'category': None,
'fiveYearAverageReturn': None,
'regularMarketPrice': 132,
'logo_url': 'https://logo.clearbit.com/apple.com'}
```


APPLE

In [18]:
```
# use .history() method to get the share price of a stock over a period of time
# the period parameter can be set as 1d, 1mo, 3mo, 6mo, 1y, 2y, 5y, 10y, ytd, max
# shares are the smallest part of a company's stock that can be bought.
apple_share_price_data = apple.history(period = 'max')

# view share price data
apple_share_price_data.head()
```

Out[18]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
1980-12-12	0.100751	0.101189	0.100751	0.100751	469033600	0.0	0.0
1980-12-15	0.095933	0.095933	0.095495	0.095495	175884800	0.0	0.0
1980-12-16	0.088923	0.088923	0.088485	0.088485	105728000	0.0	0.0
1980-12-17	0.090676	0.091114	0.090676	0.090676	86441600	0.0	0.0
1980-12-18	0.093304	0.093742	0.093304	0.093304	73449600	0.0	0.0

In [13]:
```
# reset index of the dataframe
# use inplace parameter so the change takes place to the dataframe itself
apple_share_price_data.reset_index(inplace = True)
```

```
# view data again
apple_share_price_data.head()
```

Out[13]:

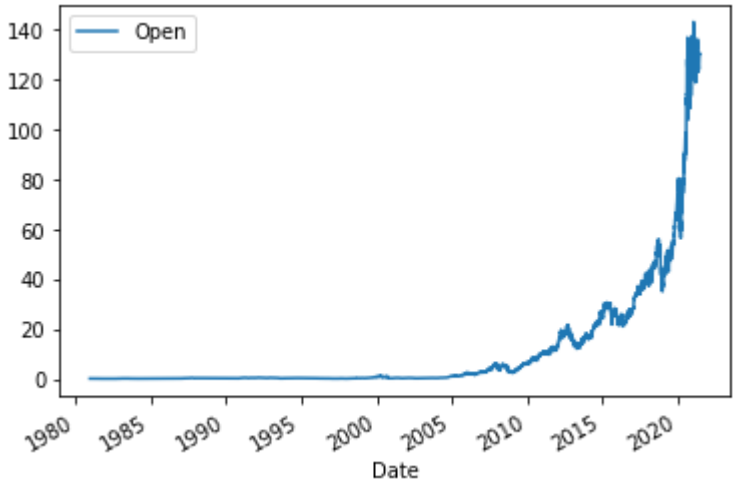
	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	1980-12-12	0.100751	0.101189	0.100751	0.100751	469033600	0.0	0.0
1	1980-12-15	0.095933	0.095933	0.095495	0.095495	175884800	0.0	0.0
2	1980-12-16	0.088923	0.088923	0.088485	0.088485	105728000	0.0	0.0
3	1980-12-17	0.090676	0.091114	0.090676	0.090676	86441600	0.0	0.0
4	1980-12-18	0.093304	0.093742	0.093304	0.093304	73449600	0.0	0.0

In [14]:

```
# plot the open price against the date
apple_share_price_data.plot(x = 'Date', y = 'Open')
```

Out[14]:

```
<AxesSubplot:xlabel='Date'>
```



In [15]:

```
# dividends are hte distribution of a company's profits to shareholders.
# dividends are the amount of money returned per share an investor owns.
# the period is defined by what we set in the .history() parameter
apple.dividends
```

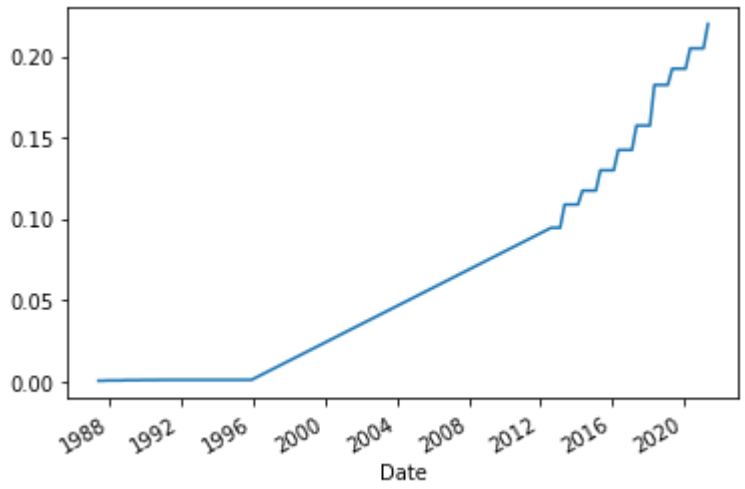
Out[15]:

```
Date
1987-05-11    0.000536
1987-08-10    0.000536
1987-11-17    0.000714
1988-02-12    0.000714
1988-05-16    0.000714
                ...
2020-05-08    0.205000
2020-08-07    0.205000
2020-11-06    0.205000
2021-02-05    0.205000
2021-05-07    0.220000
Name: Dividends, Length: 71, dtype: float64
```

In [16]:

```
# plot the dividends over time
apple.dividends.plot()
```

Out[16]: <AxesSubplot:xlabel='Date'>



AMD

In [22]:
```
# use the ticker module to create an object that will allow us to access functions & ex

# amd stock object
amd = yf.Ticker('AMD')

# get stock info in dataframe
amd_info = amd.info
```

In [25]:
```
# find country of the stock
amd_info['country']
```

Out[25]: 'United States'

In [27]:
```
# find sector of stock
amd_info['sector']
```

Out[27]: 'Technology'

In [29]:
```
# obtain stock data of amd using history function
amd_share_price_data = amd.history(period = 'max')

# view data
amd_share_price_data.head()
```

Out[29]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
1980-03-17	0.0	3.302083	3.125000	3.145833	219600	0	0.0
1980-03-18	0.0	3.125000	2.937500	3.031250	727200	0	0.0
1980-03-19	0.0	3.083333	3.020833	3.041667	295200	0	0.0
1980-03-20	0.0	3.062500	3.010417	3.010417	159600	0	0.0

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
1980-03-21	0.0	3.020833	2.906250	2.916667	130800	0	0.0

In [30]:
```
# fix index
amd_share_price_data.reset_index(inplace = True)
```

In [31]:
```
amd_share_price_data.head()
```

Out[31]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	1980-03-17	0.0	3.302083	3.125000	3.145833	219600	0	0.0
1	1980-03-18	0.0	3.125000	2.937500	3.031250	727200	0	0.0
2	1980-03-19	0.0	3.083333	3.020833	3.041667	295200	0	0.0
3	1980-03-20	0.0	3.062500	3.010417	3.010417	159600	0	0.0
4	1980-03-21	0.0	3.020833	2.906250	2.916667	130800	0	0.0

In [32]:
```
# get volume of first entry
amd_share_price_data.iloc[0][5]
```

Out[32]:
```
219600
```

In [34]:
```
amd_share_price_data.loc[0]['Volume']
```

Out[34]:
```
219600
```

LAB 2: Extracting Stock using Web Scraping

Not all stock data is available via API, so use webscraping and BeautifulSoup to obtain some more data.

In [4]:
```
# install libraries
#!pip install pandas
#!pip install requests
#!pip install bs4
#!pip install plotly
```

```
Requirement already satisfied: pandas in c:\users\orgil\appdata\local\programs\python\py
thon39\lib\site-packages (1.2.4)
Requirement already satisfied: numpy>=1.16.5 in c:\users\orgil\appdata\local\programs\py
thon\python39\lib\site-packages (from pandas) (1.20.3)
Requirement already satisfied: pytz>=2017.3 in c:\users\orgil\appdata\local\programs\pyt
hon\python39\lib\site-packages (from pandas) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\orgil\appdata\local\pr
ograms\python\python39\lib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\users\orgil\appdata\local\programs\python
\python39\lib\site-packages (from python-dateutil>=2.7.3->pandas) (1.16.0)
```

In [5]:
```
# import libraries
import pandas as pd
import requests
from bs4 import BeautifulSoup
```

NETFLIX

In [6]:
```
# use the request library to download the webpage and extract text for Netflix stock da
url = "https://finance.yahoo.com/quote/NFLX/history?period1=1439078400&period2=16231968

data  = requests.get(url).text
```

In [15]:
```
# parse the text into html with BeautifulSoup
soup = BeautifulSoup(data, 'html5lib')

# turn the html table into a dataframe
netflix_data = pd.DataFrame(columns=["Date", "Open", "High", "Low", "Close", "Volume"])
netflix_data
```

Out[15]:

Date	Open	High	Low	Close	Volume

In [16]:
```
# update the body of the dataframe
# isolate the body of the table which has all the info
# then loop through each row and find all the column values
netflix_data = pd.DataFrame(columns=["Date", "Open", "High", "Low", "Close", "Volume"])

# First we isolate the body of the table which contains all the information
# Then we loop through each row and find all the column values for each row
for row in soup.find("tbody").find_all('tr'):
    col = row.find_all("td")
    date = col[0].text
    Open = col[1].text
    high = col[2].text
    low = col[3].text
    close = col[4].text
    adj_close = col[5].text
    volume = col[6].text

    # Finally we append the data of each row to the table
    netflix_data = netflix_data.append({"Date":date, "Open":Open, "High":high, "Low":lo
```

In [17]:
```
# print the dataframe
netflix_data.head()
```

Out[17]:

	Date	Open	High	Low	Close	Volume	Adj Close
0	Jun 01, 2021	504.01	505.41	482.14	497.00	52,223,300	497.00
1	May 01, 2021	512.65	518.95	478.54	502.81	66,927,600	502.81
2	Apr 01, 2021	529.93	563.56	499.00	513.47	111,573,300	513.47
3	Mar 01, 2021	545.57	556.99	492.85	521.66	90,183,900	521.66

	Date	Open	High	Low	Close	Volume	Adj Close
4	Feb 01, 2021	536.79	566.65	518.28	538.85	61,902,300	538.85

In [19]:
```
# also can use pandas's read_html function
read_html_pandas_data = pd.read_html(url)
# since there is only one table on this web page, we just take the first table in the l

# turn it into a dataframe
netflix_data_html = read_html_pandas_data[0]
netflix_data_html
```

Out[19]:

	Date	Open	High	Low	Close*	Adj Close**
0	Jun 01, 2021	504.01	505.41	482.14	497.00	497.00
1	May 01, 2021	512.65	518.95	478.54	502.81	502.81
2	Apr 01, 2021	529.93	563.56	499.00	513.47	513.47
3	Mar 01, 2021	545.57	556.99	492.85	521.66	521.66
4	Feb 01, 2021	536.79	566.65	518.28	538.85	538.85
...	...	...	...	...	...	...
66	Dec 01, 2015	124.47	133.27	113.85	114.38	114.38
67	Nov 01, 2015	109.20	126.60	101.86	123.33	123.33
68	Oct 01, 2015	102.91	115.83	96.26	108.38	108.38
69	Sep 01, 2015	109.35	111.24	93.55	103.26	103.26
70	*Close price adjusted for splits.**Adjusted cl...	*Close price adjusted for splits.**Adjusted cl...	*Close price adjusted for splits.**Adjusted cl...	*Close price adjusted for splits.**Adjusted cl...	*Close price adjusted for splits.**Adjusted cl...	*Close price adjusted for splits.**Adjusted cl...	sp

71 rows × 7 columns

AMAZON

In [17]:
```
# get amazon stock website data
url = 'https://finance.yahoo.com/quote/AMZN/history?period1=1451606400&period2=16121376

data = requests.get(url).text
```

In [18]:
```
# parse the html using beautifulSoup
soup = BeautifulSoup(data, 'html5lib')
```

In [19]:
```
# view title attribute
soup.title
```

Out[19]:
```
<title>Amazon.com, Inc. (AMZN) Stock Historical Prices &amp;amp; Data - Yahoo Finance</title
```

>

In [20]:
```
# use beautifulSoup to extract the data and store it in a dataframe
amazon_data = pd.DataFrame(columns=["Date", "Open", "High", "Low", "Close", "Volume"])

for row in soup.find("tbody").find_all("tr"):
    col = row.find_all("td")
    date = col[0].text
    Open = col[1].text
    high = col[2].text
    low = col[3].text
    close = col[4].text
    adj_close = col[5].text
    volume = col[6].text

    # append the data of each row to the table
    amazon_data = amazon_data.append({"Date":date, "Open":Open, "High":high, "Low":low,
```

In [25]:
```
# view first 5 rows of the amazon dataframe
amazon_data.head()
```

Out[25]:

	Date	Open	High	Low	Close	Volume	Adj Close
0	Jan 01, 2021	3,270.00	3,363.89	3,086.00	3,206.20	71,528,900	3,206.20
1	Dec 01, 2020	3,188.50	3,350.65	3,072.82	3,256.93	77,556,200	3,256.93
2	Nov 01, 2020	3,061.74	3,366.80	2,950.12	3,168.04	90,810,500	3,168.04
3	Oct 01, 2020	3,208.00	3,496.24	3,019.00	3,036.15	116,226,100	3,036.15
4	Sep 01, 2020	3,489.58	3,552.25	2,871.00	3,148.73	115,899,300	3,148.73

In [27]:
```
# view column names of the dataframe
amazon_data.columns
```

Out[27]:
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close'], dtype='object')
```

In [35]:
```
# get value of last row's 'open' column
amazon_data.iloc[-1][1]
```

Out[35]:
```
'656.29'
```

Project: Extracting and Visualizing Stock Data

In [4]:
```
# install needed libraries
#!pip install yfinance
#!pip install pandas
#!pip install requests
#!pip install bs4
#!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\orgil\appdata\local\programs\python\py
thon39\lib\site-packages (4.14.3)
```

```
Requirement already satisfied: retrying>=1.3.3 in c:\users\orgil\appdata\local\programs
\python\python39\lib\site-packages (from plotly) (1.3.3)
Requirement already satisfied: six in c:\users\orgil\appdata\local\programs\python\pytho
n39\lib\site-packages (from plotly) (1.16.0)
```

In [5]:

```
# import libraries
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 [22]:

```
# define make_graph function
# it takes a dataframe with stock data (with date & close info), a dataframe with reven
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Historical
    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_
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date, infer_datetim
    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()
```

In [23]:

```
# use yfinance to extract stock data with the ticker function for tesla
tesla = yf.Ticker('TSLA')
```

In [24]:

```
# view the data
tesla_info = tesla.info
tesla_info
```

Out[24]:

```
{'zip': '94304',
 'sector': 'Consumer Cyclical',
 'fullTimeEmployees': 70757,
 'longBusinessSummary': 'Tesla, Inc. designs, develops, manufactures, leases, and sells
electric vehicles, and energy generation and storage systems in the United States, Chin
a, and internationally. The company operates in two segments, Automotive, and Energy Gen
eration and Storage. The Automotive segment offers electric vehicles, as well as sells a
utomotive regulatory credits. It provides sedans and sport utility vehicles through dire
ct and used vehicle sales, a network of Tesla Superchargers, and in-app upgrades; and pu
rchase financing and leasing services. This segment is also involved in the provision of
non-warranty after-sales vehicle services, sale of used vehicles, retail merchandise, an
d vehicle insurance, as well as sale of products through its subsidiaries to third party
customers; services for electric vehicles through its company-owned service locations, a
nd Tesla mobile service technicians; and vehicle limited warranties and extended service
plans. The Energy Generation and Storage segment engages in the design, manufacture, ins
tallation, sale, and leasing of solar energy generation and energy storage products, and
related services to residential, commercial, and industrial customers and utilities thro
ugh its website, stores, and galleries, as well as through a network of channel partner
s. This segment also offers service and repairs to its energy product customers, includi
```

ng under warranty; and 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 founded in 2003 and is headquartered in Palo Alto, California.',
 'city': 'Palo Alto',
 'phone': '650-681-5000',
 'state': 'CA',
 'country': 'United States',
 'companyOfficers': [],
 'website': 'http://www.tesla.com',
 'maxAge': 1,
 'address1': '3500 Deer Creek Road',
 'industry': 'Auto Manufacturers',
 'previousClose': 623.71,
 'regularMarketOpen': 632,
 'twoHundredDayAverage': 695.76605,
 'trailingAnnualDividendYield': None,
 'payoutRatio': 0,
 'volume24Hr': None,
 'regularMarketDayHigh': 656.0233,
 'navPrice': None,
 'averageDailyVolume10Day': 21644300,
 'totalAssets': None,
 'regularMarketPreviousClose': 623.71,
 'fiftyDayAverage': 610.9986,
 'trailingAnnualDividendRate': None,
 'open': 632,
 'toCurrency': None,
 'averageVolume10days': 21644300,
 'expireDate': None,
 'yield': None,
 'algorithm': None,
 'dividendRate': None,
 'exDividendDate': None,
 'beta': 1.995108,
 'circulatingSupply': None,
 'startDate': None,
 'regularMarketDayLow': 630.13,
 'priceHint': 2,
 'currency': 'USD',
 'trailingPE': 654.12823,
 'regularMarketVolume': 19190547,
 'lastMarket': None,
 'maxSupply': None,
 'openInterest': None,
 'marketCap': 628881096704,
 'volumeAllCurrencies': None,
 'strikePrice': None,
 'averageVolume': 29606878,
 'priceToSalesTrailing12Months': 17.498083,
 'dayLow': 630.13,
 'ask': 652.31,
 'ytdReturn': None,
 'askSize': 1100,
 'volume': 19190547,
 'fiftyTwoWeekHigh': 900.4,
 'forwardPE': 104.95499,
 'fromCurrency': None,
 'fiveYearAvgDividendYield': None,
 'fiftyTwoWeekLow': 187.43,
 'bid': 652.3,
 'tradeable': False,
 'dividendYield': None,
 'bidSize': 900,
 'dayHigh': 656.0233,
 'exchange': 'NMS',

```
'shortName': 'Tesla, Inc.',
'longName': 'Tesla, Inc.',
'exchangeTimezoneName': 'America/New_York',
'exchangeTimezoneShortName': 'EDT',
'isEsgPopulated': False,
'gmtOffSetMilliseconds': '-14400000',
'quoteType': 'EQUITY',
'symbol': 'TSLA',
'messageBoardId': 'finmb_27444752',
'market': 'us_market',
'annualHoldingsTurnover': None,
'enterpriseToRevenue': 16.552,
'beta3Year': None,
'profitMargins': 0.0318,
'enterpriseToEbitda': 130.715,
'52WeekChange': 2.245616,
'morningStarRiskRating': None,
'forwardEps': 6.22,
'revenueQuarterlyGrowth': None,
'sharesOutstanding': 963329984,
'fundInceptionDate': None,
'annualReportExpenseRatio': None,
'bookValue': 23.901,
'sharesShort': 40046181,
'sharesPercentSharesOut': 0.0416,
'fundFamily': None,
'lastFiscalYearEnd': 1609372800,
'heldPercentInstitutions': 0.42664,
'netIncomeToCommon': 1112000000,
'trailingEps': 0.998,
'lastDividendValue': None,
'SandP52WeekChange': 0.39212477,
'priceToBook': 27.313503,
'heldPercentInsiders': 0.19629999,
'nextFiscalYearEnd': 1672444800,
'mostRecentQuarter': 1617148800,
'shortRatio': 1.28,
'sharesShortPreviousMonthDate': 1619740800,
'floatShares': 775057145,
'enterpriseValue': 594882461696,
'threeYearAverageReturn': None,
'lastSplitDate': 1598832000,
'lastSplitFactor': '5:1',
'legalType': None,
'lastDividendDate': None,
'morningStarOverallRating': None,
'earningsQuarterlyGrowth': 26.375,
'dateShortInterest': 1622160000,
'pegRatio': 3.66,
'lastCapGain': None,
'shortPercentOfFloat': 0.051599998,
'sharesShortPriorMonth': 41382433,
'impliedSharesOutstanding': None,
'category': None,
'fiveYearAverageReturn': None,
'regularMarketPrice': 652.82,
'logo_url': 'https://logo.clearbit.com/tesla.com'}
```


In [26]:
```
# use the ticker object & history function to get stock info in a dataframe and set per
tesla_data = tesla.history(period = 'max')
```

In [27]:
```
# reset the index and display the first five rows of tesla_data
```

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

Out[27]:

	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

In [28]:
```
# use webscraping to extract tesla revenue data with the requests library.
url = 'https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue'
html_data = requests.get(url).text
```

In [29]:
```
# parse the html data with BeautifulSoup
soup = BeautifulSoup(html_data, 'html5lib')
```

In [31]:
```
# find all html tables in the page
tables = soup.find_all('table')
len(tables)
```

Out[31]: 6

In [32]:
```
# search the tables to find the correct one we want
for index, table in enumerate(tables):
    if ('Tesla Quarterly Revenue' in str(table)):
        table_index = index
print(table_index)
```

```
1
```

In [33]:
```
# locate the table name & clean it up
print(tables[table_index].prettify())
```

```
<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">
    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>
  <tr>
   <td style="text-align:center">
    2008-12-31
   </td>
   <td style="text-align:center">
   </td>
  </tr>
 </tbody>
</table>
```

In [34]:
```
# use beautifulSoup to extract the data and store it in a dataframe
tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])

for row in tables[table_index].tbody.find_all("tr"):
    col = row.find_all("td")
    date = col[0].text
    revenue = col[1].text


    # append the data of each row to the table
    tesla_revenue = tesla_revenue.append({"Date":date, "Revenue":revenue}, ignore_index
```

In [35]:
```
tesla_revenue
```

Out[35]:

	Date	Revenue
0	2021-03-31	$10,389
1	2020-12-31	$10,744
2	2020-09-30	$8,771
3	2020-06-30	$6,036
4	2020-03-31	$5,985
5	2019-12-31	$7,384
6	2019-09-30	$6,303
7	2019-06-30	$6,350
8	2019-03-31	$4,541
9	2018-12-31	$7,226
10	2018-09-30	$6,824
11	2018-06-30	$4,002
12	2018-03-31	$3,409
13	2017-12-31	$3,288
14	2017-09-30	$2,985
15	2017-06-30	$2,790
16	2017-03-31	$2,696
17	2016-12-31	$2,285
18	2016-09-30	$2,298
19	2016-06-30	$1,270
20	2016-03-31	$1,147
21	2015-12-31	$1,214
22	2015-09-30	$937
23	2015-06-30	$955
24	2015-03-31	$940

	Date	Revenue
25	2014-12-31	$957
26	2014-09-30	$852
27	2014-06-30	$769
28	2014-03-31	$621
29	2013-12-31	$615
30	2013-09-30	$431
31	2013-06-30	$405
32	2013-03-31	$562
33	2012-12-31	$306
34	2012-09-30	$50
35	2012-06-30	$27
36	2012-03-31	$30
37	2011-12-31	$39
38	2011-09-30	$58
39	2011-06-30	$58
40	2011-03-31	$49
41	2010-12-31	$36
42	2010-09-30	$31
43	2010-06-30	$28
44	2010-03-31	$21
45	2009-12-31	
46	2009-09-30	$46
47	2009-06-30	$27
48	2008-12-31	

In [37]:
```
# remove the comma & dollar sign from the revenue column.
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

```
<ipython-input-37-f111bfd476dd>:2: FutureWarning: The default value of regex will change
from True to False in a future version.
  tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

In [39]:
```
# remove any null or empty strings in revenue column
tesla_revenue.dropna(inplace=True)

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

In [40]:
```
# display the last 5 rows of the tesla_revenue dataframe with the .tail function
```

```
tesla_revenue.tail(5)
```

Out[40]:

	Date	Revenue
42	2010-09-30	31
43	2010-06-30	28
44	2010-03-31	21
46	2009-09-30	46
47	2009-06-30	27

In [45]:
```
# use yfinance to extract gamestop stock data with the ticker function
gamestop = yf.Ticker('GME')
```

In [46]:
```
# use the history function to extract stock info with period set to max.
gme_data = gamestop.history(period = 'max')
```

In [47]:
```
# reset the index and display first 5 rows
gme_data.reset_index(inplace = True)
gme_data.head(5)
```

Out[47]:

	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

In [48]:
```
# use webscraping to extract GME Revenue data with the requests library
url = 'https://www.macrotrends.net/stocks/charts/GME/gamestop/revenue'
html_data = requests.get(url).text
```

In [49]:
```
# parse the html data with BeautifulSoup
soup = BeautifulSoup(html_data, 'html5lib')
```

In [50]:
```
# find all html tables in the page so we can identify the gamestop quarterly revenue ta
tables = soup.find_all('table')
len(tables)
```

Out[50]: 6

In [51]:
```
# search the tables to find the correct one we want
for index, table in enumerate(tables):
    if ('GameStop Quarterly Revenue' in str(table)):
```

```
        table_index = index
print(table_index)
```

```
1
```

In [52]:

```
# use beautifulSoup to extract the data and store it in a dataframe
gme_revenue = pd.DataFrame(columns=["Date", "Revenue"])

for row in tables[table_index].tbody.find_all("tr"):
    col = row.find_all("td")
    date = col[0].text
    revenue = col[1].text


    # append the data of each row to the table
    gme_revenue = gme_revenue.append({"Date":date, "Revenue":revenue}, ignore_index=Tru
```

In [53]:

```
# remove the comma & dollar sign from the revenue column.
gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',|\$',"")
```

```
<ipython-input-53-0e12756033e9>:2: FutureWarning: The default value of regex will change
from True to False in a future version.
  gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',|\$',"")
```

In [54]:

```
# remove any null or empty strings in revenue column
gme_revenue.dropna(inplace=True)

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

In [55]:

```
# display the last 5 rows of the dataframe
gme_revenue.tail(5)
```

Out[55]:

	Date	Revenue
61	2006-01-31	1667
62	2005-10-31	534
63	2005-07-31	416
64	2005-04-30	475
65	2005-01-31	709

In [57]:

```
# plot the tesla stock graph with the make_graph function
make_graph(tesla_data, tesla_revenue, 'Tesla Stock')
```

In [58]:

```
# plot gme stock info with make_graph function
make_graph(gme_data, gme_revenue, 'GameStop Stock')
```