# Final Assignment

July 22, 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

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

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.

[2]:
```
!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 (0.2.65)
Requirement already satisfied: pandas>=1.3.0 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.3.1)
Requirement already satisfied: numpy>=1.16.5 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.3.1)
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) (0.0.12)
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) (3.18.2)
Requirement already satisfied: beautifulsoup4>=4.11.1 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.12.3)
Requirement already satisfied: curl_cffi>=0.7 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (0.12.0)
Requirement already satisfied: protobuf>=3.19.0 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (6.31.1)
Requirement already satisfied: websockets>=13.0 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (15.0.1)
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: cffi>=1.12.0 in /opt/conda/lib/python3.12/site-
packages (from curl_cffi>=0.7->yfinance) (1.17.1)
Requirement already satisfied: certifi>=2024.2.2 in
/opt/conda/lib/python3.12/site-packages (from curl_cffi>=0.7->yfinance)
(2024.12.14)
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) (2025.2)
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: pycparser in /opt/conda/lib/python3.12/site-
packages (from cffi>=1.12.0->curl_cffi>=0.7->yfinance) (2.22)
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
(6.2.0)
Requirement already satisfied: narwhals>=1.15.1 in
/opt/conda/lib/python3.12/site-packages (from plotly) (1.48.0)
Requirement already satisfied: packaging in /opt/conda/lib/python3.12/site-
packages (from plotly) (24.2)
```

[45]:
```
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
```

[46]:
```
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.

[5]:
```
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.**

[6]:
```
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`.

[7]:
```
#create ticker object for tesla
Tesla = yf.Ticker("TSLA")

# extract historical stock data with the maximum period
Tesla_data = Tesla.history(period ="max")

#Reset the index of the Dataframe
Tesla_data.reset_index(inplace=True)

#Display the first few rows
```

```
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
```

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.

**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.

### 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`.

[39]:
```
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/revenue.htm"

# send GET request to the url
response = requests.get(url)

#save the html content to the variable
html_data = response.text

#print firts few characters to confirm
print(html_data[:500]) #priview only
```

```
<!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`.

[9]:
```
from bs4 import BeautifulSoup

#Parse the html content using BeautifulSoup and the html.parser
Soup = BeautifulSoup(html_data, "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. 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.
```

[35]:
```
from bs4 import BeautifulSoup
import pandas as pd

#create an empty dataframe
tesla_revenue = pd.DataFrame(columns =["Data", "revenue"])

#parse the HTML and find the relevant table
```

```
Soup = BeautifulSoup(html_data,"html.parser")
tables = Soup.find_all("table")

#check for the tesla quarterly revenue table
for table in tables:
    if "Tesla Quarterly Revenue" in table.text:
        revenue_table = table
        break

revenue_data = []

if revenue_table:
    tbody = revenue_table.find("tbody")
    if tbody:
        for row in tbody.find_all("tr"):
            cols = row.find_all("td")
            if len(cols) == 2:
                date = cols[0].text.strip()
                revenue = cols[1].text.strip()
                revenue_data.append({"Date": date, "Revenue": revenue})


# Print the DataFrame
tesla_revenue = pd.DataFrame(revenue_data)

print(tesla_revenue.head())
```

```
         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
```

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

[25]:
```
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.

[24]:
```
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.

[13]:
```
print(tesla_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
```

### 0.4 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`.

[14]:
```
gamestop = 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.

[15]:
```
gme_data = gamestop.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.

[16]:
```
gme_data.reset_index(inplace = True)
gme_data.head()
```

[16]:
```
                       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  1.658001  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`.

[17]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

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
```

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

[18]:
```
Soup = BeautifulSoup(html_data, "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
```

[19]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Download the HTML
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 with BeautifulSoup
soup = BeautifulSoup(html_data_2, "html.parser")

# Extract the second <tbody> as instructed
tbody = soup.find_all("tbody")[1]

# Loop through the rows and extract Date and Revenue
revenue_list = []
for row in tbody.find_all("tr"):
    cols = row.find_all("td")
```

```
    if len(cols) == 2:
        date = cols[0].text.strip()
        revenue = cols[1].text.strip().replace("$", "").replace(",", "")
        if revenue:  # skip empty entries
            revenue_list.append({"Date": date, "Revenue": float(revenue)})

# Create DataFrame
gme_revenue = pd.DataFrame(revenue_list)

# Show the result
print(gme_revenue.head())
```

```
         Date  Revenue
0  2020-04-30   1021.0
1  2020-01-31   2194.0
2  2019-10-31   1439.0
3  2019-07-31   1286.0
4  2019-04-30   1548.0
```

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

[22]:
```
print(gme_revenue.tail())
```

```
          Date  Revenue
57  2006-01-31   1667.0
58  2005-10-31    534.0
59  2005-07-31    416.0
60  2005-04-30    475.0
61  2005-01-31    709.0
```

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.

[41]:
```
import pandas as pd

# Load Tesla stock data
tesla_data_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.
 ↪cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/
 ↪tesla_stock.csv"
tesla_data = pd.read_csv(tesla_data_url)

# Convert date column to datetime format
tesla_data["Date"] = pd.to_datetime(tesla_data["Date"])
tesla_data.sort_values("Date", inplace=True)
```

```
---------------------------------------------------------------------------
HTTPError                                 Traceback (most recent call last)
```

```
Cell In[41], line 5
      3 # Load Tesla stock data
      4 tesla_data_url = "https://cf-courses-data.s3.us.cloud-object-storage.
 ↪appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project
 ↪tesla_stock.csv"
----> 5 tesla_data = pd.read_csv(tesla_data_url)
      7 # Convert date column to datetime format
      8 tesla_data["Date"] = pd.to_datetime(tesla_data["Date"])

File /opt/conda/lib/python3.12/site-packages/pandas/io/parsers/readers.py:1026,
 ↪in read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col,␣
 ↪usecols, dtype, engine, converters, true_values, false_values,␣
 ↪skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na,␣
 ↪na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format,␣
 ↪keep_date_col, date_parser, date_format, dayfirst, cache_dates, iterator,␣
 ↪chunksize, compression, thousands, decimal, lineterminator, quotechar,␣
 ↪quoting, doublequote, escapechar, comment, encoding, encoding_errors, dialect␣
 ↪on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision,␣
 ↪storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (…)
   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File /opt/conda/lib/python3.12/site-packages/pandas/io/parsers/readers.py:620,␣
 ↪in _read(filepath_or_buffer, kwds)
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File /opt/conda/lib/python3.12/site-packages/pandas/io/parsers/readers.py:1620,
 ↪in TextFileReader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File /opt/conda/lib/python3.12/site-packages/pandas/io/parsers/readers.py:1880,
 ↪in TextFileReader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get("encoding", None),
```

```
   1884     compression=self.options.get("compression", None),
   1885     memory_map=self.options.get("memory_map", False),
   1886     is_text=is_text,
   1887     errors=self.options.get("encoding_errors", "strict"),
   1888     storage_options=self.options.get("storage_options", None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File /opt/conda/lib/python3.12/site-packages/pandas/io/common.py:728, in␣
 ↪get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text,␣
 ↪errors, storage_options)
    725     codecs.lookup_error(errors)
    727 # open URLs
--> 728 ioargs = _get_filepath_or_buffer(
    729     path_or_buf,
    730     encoding=encoding,
    731     compression=compression,
    732     mode=mode,
    733     storage_options=storage_options,
    734 )
    736 handle = ioargs.filepath_or_buffer
    737 handles: list[BaseBuffer]

File /opt/conda/lib/python3.12/site-packages/pandas/io/common.py:384, in␣
 ↪_get_filepath_or_buffer(filepath_or_buffer, encoding, compression, mode,␣
 ↪storage_options)
    382 # assuming storage_options is to be interpreted as headers
    383 req_info = urllib.request.Request(filepath_or_buffer,␣
 ↪headers=storage_options)
--> 384 with urlopen(req_info) as req:
    385     content_encoding = req.headers.get("Content-Encoding", None)
    386     if content_encoding == "gzip":
    387         # Override compression based on Content-Encoding header

File /opt/conda/lib/python3.12/site-packages/pandas/io/common.py:289, in␣
 ↪urlopen(*args, **kwargs)
    283 """
    284 Lazy-import wrapper for stdlib urlopen, as that imports a big chunk of
    285 the stdlib.
    286 """
    287 import urllib.request
--> 289 return urllib.request.urlopen(*args, **kwargs)

File /opt/conda/lib/python3.12/urllib/request.py:215, in urlopen(url, data,␣
 ↪timeout, cafile, capath, cadefault, context)
    213 else:
    214     opener = _opener
```

```
--> 215 return opener.open(url, data, timeout)

File /opt/conda/lib/python3.12/urllib/request.py:521, in OpenerDirector.
 ↪open(self, fullurl, data, timeout)
    519 for processor in self.process_response.get(protocol, []):
    520     meth = getattr(processor, meth_name)
--> 521     response = meth(req, response)
    523 return response

File /opt/conda/lib/python3.12/urllib/request.py:630, in HTTPErrorProcessor.
 ↪http_response(self, request, response)
    627 # According to RFC 2616, "2xx" code indicates that the client's
    628 # request was successfully received, understood, and accepted.
    629 if not (200 <= code < 300):
--> 630     response = self.parent.error(
    631         'http', request, response, code, msg, hdrs)
    633 return response

File /opt/conda/lib/python3.12/urllib/request.py:559, in OpenerDirector.
 ↪error(self, proto, *args)
    557 if http_err:
    558     args = (dict, 'default', 'http_error_default') + orig_args
--> 559     return self._call_chain(*args)

File /opt/conda/lib/python3.12/urllib/request.py:492, in OpenerDirector.
 ↪_call_chain(self, chain, kind, meth_name, *args)
    490 for handler in handlers:
    491     func = getattr(handler, meth_name)
--> 492     result = func(*args)
    493     if result is not None:
    494         return result

File /opt/conda/lib/python3.12/urllib/request.py:639, in HTTPDefaultErrorHandle
 ↪http_error_default(self, req, fp, code, msg, hdrs)
    638 def http_error_default(self, req, fp, code, msg, hdrs):
--> 639     raise HTTPError(req.full_url, code, msg, hdrs, fp)

HTTPError: HTTP Error 404: Not Found
```

[43]:
```
tesla_revenue_url = "https://cf-courses-data.s3.us.cloud-object-storage.
 ↪appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/
 ↪project/revenue.htm"
tesla_revenue = pd.read_html(tesla_revenue_url)[1]

# Clean the revenue data
tesla_revenue.columns = ["Date", "Revenue"]
```

```
tesla_revenue["Revenue"] = (
    tesla_revenue["Revenue"].astype(str)
    .str.replace("$", "", regex=False)
    .str.replace(",", "", regex=False)
    .astype(float)
)
tesla_revenue["Date"] = pd.to_datetime(tesla_revenue["Date"])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[43], line 2
      1 tesla_revenue_url = "https://cf-courses-data.s3.us.cloud-object-storage
 ↪appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project
 ↪revenue.htm"
----> 2 tesla_revenue = pd.read_html(tesla_revenue_url)[1]
      4 # Clean the revenue data
      5 tesla_revenue.columns = ["Date", "Revenue"]

TypeError: 'str' object is not callable
```

[44]:
```
plot_tesla = make_graph(tesla_data,tesla_revenue,'Tesla')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[44], line 1
----> 1 plot_tesla = make_graph(tesla_data,tesla_revenue,'Tesla')

Cell In[6], line 3, in make_graph(stock_data, revenue_data, stock)
      1 def make_graph(stock_data, revenue_data, stock):
      2     fig = make_subplots(rows=2, cols=1, shared_xaxes=True,␣
 ↪subplot_titles=("Historical Share Price", "Historical Revenue"),␣
 ↪vertical_spacing = .3)
----> 3     stock_data_specific = stock_data[stock_data.Date <= '2021-06-14']
      4     revenue_data_specific = revenue_data[revenue_data.Date <=␣
 ↪'2021-04-30']
      5     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)

AttributeError: 'str' object has no attribute 'Date'
```

Hint

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

## 0.6 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
```

[32]:
```
plot_gme = make_graph(gme_data,gme_revenue,'GameStop')
```

```
/tmp/ipykernel_2240/109047474.py:5: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.

/tmp/ipykernel_2240/109047474.py:6: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.


<IPython.core.display.HTML object>
```

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

## 0.7 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 |

##