Final Assignment

March 27, 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.

[103]:
```
!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 (0.2.55)
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) (2.2.4)
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) (2025.2)
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: 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
(6.0.1)
Requirement already satisfied: narwhals>=1.15.1 in
/opt/conda/lib/python3.12/site-packages (from plotly) (1.32.0)
Requirement already satisfied: packaging in /opt/conda/lib/python3.12/site-
packages (from plotly) (24.2)
```

[104]:

```python
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
```

[105]:

```python
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.

[106]:

```python
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.**

[107]:

```python
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'.

[108]:
```
tesla = yf.Ticker("TSLA")
```

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `tesla_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

[109]:
```
tesla_data = Tesla.history(period="max")
```

Reset the index using the `reset_index(inplace=True)` function on the tesla_data DataFrame and display the first five rows of the `tesla_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 1 to the results below.

[110]:
```
tesla_data.reset_index(inplace=True)
tesla_data.head(5)
```

[110]:
```
                       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`.

[111]:
```
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm'
html_data = requests.get(url).content
```

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

[112]:
```
soup = BeautifulSoup(html_data, "html.parser")
soup.find_all('title')
```

[112]:
```
[<title>Tesla Revenue 2010-2022 | TSLA | MacroTrends</title>]
```

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.

[113]:
```
table = soup.find('table')
table_rows = table.find_all('tr')
data = []
for tr in table_rows:
    td = tr.find_all('td')
    row = [tr.text for tr in td]
    data.append(row)
tesla_revenue = pd.DataFrame(data)
tesla_revenue.columns = ['Date','Revenue']
```

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

[114]:
```
tesla_revenue ["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

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

[115]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
tesla_revenue.dropna(inplace=True)

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

[116]:
```
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.

[117]:
```
tesla_revenue.tail()
```

[117]:

	Date	Revenue
9	2013	$2,013
10	2012	$413
11	2011	$204
12	2010	$117
13	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`.

[94]:
```
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.

[95]:
```python
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.

[96]:
```python
gme_data.reset_index(inplace=True)
gme_data.head(5)
```

[96]:
```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666  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  1.615920  1.662210  1.603296  1.662210   6892800

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

0.5 Question 4: Use Webscraping to Extract GME Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html. Save the text of the response as a variable named `html_data_2`.

[97]:
```python
url2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html'
html_data_2 = requests.get(url2).text
```

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

[98]:
```python
soup_2 = BeautifulSoup(html_data_2,'html.parser')
```

Using `BeautifulSoup` or the `read_html` function extract the table with `GameStop Revenue` and store it into a dataframe named `gme_revenue`. The dataframe should have columns `Date` and `Revenue`. Make sure the comma and dollar sign is removed from the `Revenue` column.

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

Click here if you need help locating the table

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

soup.find_all("tbody")[1]
```

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

[99]:
```python
table2 = soup_2.find('table')
table_rows2 = table2.find_all('tr')
data2 = []
for tr in table_rows2:
    td = tr.find_all('td')
    row = [tr.text for tr in td]
    data2.append(row)
gme_revenue = pd.DataFrame(data2)
gme_revenue.columns = ['Date','Revenue']
gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',|\$',"")
```

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

[100]:
```python
gme_revenue.tail(5)
```

[100]:

	Date	Revenue
12	2009	$8,806
13	2008	$7,094
14	2007	$5,319
15	2006	$3,092
16	2005	$1,843

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.

[118]:
```python
make_graph (tesla_data, tesla_revenue, 'Tesla')
```

```
/tmp/ipykernel_301/109047474.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_301/109047474.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.
```

```
---------------------------------------------------------------------------
```

```
ValueError                                Traceback (most recent call last)
Cell In[118], line 1
----> 1 make_graph (tesla_data, tesla_revenue, 'Tesla')

Cell In[107], line 6, in make_graph(stock_data, revenue_data, stock)
      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)
----> 6 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)
      7 fig.update_xaxes(title_text="Date", row=1, col=1)
      8 fig.update_xaxes(title_text="Date", row=2, col=1)

File /opt/conda/lib/python3.12/site-packages/pandas/core/generic.py:6643, in␣
 ↪NDFrame.astype(self, dtype, copy, errors)
   6637     results = [
   6638         ser.astype(dtype, copy=copy, errors=errors) for _, ser in self.
 ↪items()
   6639     ]
   6641 else:
   6642     # else, only a single dtype is given
-> 6643     new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
   6644     res = self._constructor_from_mgr(new_data, axes=new_data.axes)
   6645     return res.__finalize__(self, method="astype")

File /opt/conda/lib/python3.12/site-packages/pandas/core/internals/managers.py:
 ↪430, in BaseBlockManager.astype(self, dtype, copy, errors)
    427 elif using_copy_on_write():
    428     copy = False
--> 430 return self.apply(
    431     "astype",
    432     dtype=dtype,
    433     copy=copy,
    434     errors=errors,
    435     using_cow=using_copy_on_write(),
    436 )

File /opt/conda/lib/python3.12/site-packages/pandas/core/internals/managers.py:
 ↪363, in BaseBlockManager.apply(self, f, align_keys, **kwargs)
    361         applied = b.apply(f, **kwargs)
    362     else:
--> 363         applied = getattr(b, f)(**kwargs)
    364     result_blocks = extend_blocks(applied, result_blocks)
    366 out = type(self).from_blocks(result_blocks, self.axes)
```

```
File /opt/conda/lib/python3.12/site-packages/pandas/core/internals/blocks.py:
↪758, in Block.astype(self, dtype, copy, errors, using_cow, squeeze)
    755         raise ValueError("Can not squeeze with more than one column.")
    756     values = values[0, :]  # type: ignore[call-overload]
--> 758 new_values = astype_array_safe(values, dtype, copy=copy, errors=errors)
    760 new_values = maybe_coerce_values(new_values)
    762 refs = None

File /opt/conda/lib/python3.12/site-packages/pandas/core/dtypes/astype.py:237,␣
↪in astype_array_safe(values, dtype, copy, errors)
    234     dtype = dtype.numpy_dtype
    236 try:
--> 237     new_values = astype_array(values, dtype, copy=copy)
    238 except (ValueError, TypeError):
    239     # e.g. _astype_nansafe can fail on object-dtype of strings
    240     #  trying to convert to float
    241     if errors == "ignore":

File /opt/conda/lib/python3.12/site-packages/pandas/core/dtypes/astype.py:182,␣
↪in astype_array(values, dtype, copy)
    179     values = values.astype(dtype, copy=copy)
    181 else:
--> 182     values = _astype_nansafe(values, dtype, copy=copy)
    184 # in pandas we don't store numpy str dtypes, so convert to object
    185 if isinstance(dtype, np.dtype) and issubclass(values.dtype.type, str):

File /opt/conda/lib/python3.12/site-packages/pandas/core/dtypes/astype.py:133,␣
↪in _astype_nansafe(arr, dtype, copy, skipna)
    129     raise ValueError(msg)
    131 if copy or arr.dtype == object or dtype == object:
    132     # Explicit copy, or required since NumPy can't view from / to object.
--> 133     return arr.astype(dtype, copy=True)
    135 return arr.astype(dtype, copy=copy)

ValueError: could not convert string to float: '$53,823'
```

Hint

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

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.

[102]:
```
make_graph (gme_data, gme_revenue, 'GameStop')
```

```
/tmp/ipykernel_301/109047474.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_301/109047474.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.
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[102], line 1
----> 1 make_graph (gme_data, gme_revenue, 'GameStop')

Cell In[81], line 6, in make_graph(stock_data, revenue_data, stock)
      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)
----> 6 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)
      7 fig.update_xaxes(title_text="Date", row=1, col=1)
      8 fig.update_xaxes(title_text="Date", row=2, col=1)

File /opt/conda/lib/python3.12/site-packages/pandas/core/generic.py:6643, in␣
 ↪NDFrame.astype(self, dtype, copy, errors)
   6637     results = [
   6638         ser.astype(dtype, copy=copy, errors=errors) for _, ser in self.
 ↪items()
   6639     ]
   6641 else:
   6642     # else, only a single dtype is given
-> 6643     new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
   6644     res = self._constructor_from_mgr(new_data, axes=new_data.axes)
   6645     return res.__finalize__(self, method="astype")

File /opt/conda/lib/python3.12/site-packages/pandas/core/internals/managers.py:
 ↪430, in BaseBlockManager.astype(self, dtype, copy, errors)
    427 elif using_copy_on_write():
    428     copy = False
--> 430 return self.apply(
```

```
    431     "astype",
    432     dtype=dtype,
    433     copy=copy,
    434     errors=errors,
    435     using_cow=using_copy_on_write(),
    436 )

File /opt/conda/lib/python3.12/site-packages/pandas/core/internals/managers.py:
↪363, in BaseBlockManager.apply(self, f, align_keys, **kwargs)
    361         applied = b.apply(f, **kwargs)
    362     else:
--> 363         applied = getattr(b, f)(**kwargs)
    364     result_blocks = extend_blocks(applied, result_blocks)
    366 out = type(self).from_blocks(result_blocks, self.axes)

File /opt/conda/lib/python3.12/site-packages/pandas/core/internals/blocks.py:
↪758, in Block.astype(self, dtype, copy, errors, using_cow, squeeze)
    755         raise ValueError("Can not squeeze with more than one column.")
    756     values = values[0, :]  # type: ignore[call-overload]
--> 758 new_values = astype_array_safe(values, dtype, copy=copy, errors=errors)
    760 new_values = maybe_coerce_values(new_values)
    762 refs = None

File /opt/conda/lib/python3.12/site-packages/pandas/core/dtypes/astype.py:237,␣
↪in astype_array_safe(values, dtype, copy, errors)
    234     dtype = dtype.numpy_dtype
    236 try:
--> 237     new_values = astype_array(values, dtype, copy=copy)
    238 except (ValueError, TypeError):
    239     # e.g. _astype_nansafe can fail on object-dtype of strings
    240     #  trying to convert to float
    241     if errors == "ignore":

File /opt/conda/lib/python3.12/site-packages/pandas/core/dtypes/astype.py:182,␣
↪in astype_array(values, dtype, copy)
    179     values = values.astype(dtype, copy=copy)
    181 else:
--> 182     values = _astype_nansafe(values, dtype, copy=copy)
    184 # in pandas we don't store numpy str dtypes, so convert to object
    185 if isinstance(dtype, np.dtype) and issubclass(values.dtype.type, str):

File /opt/conda/lib/python3.12/site-packages/pandas/core/dtypes/astype.py:133,␣
↪in _astype_nansafe(arr, dtype, copy, skipna)
    129     raise ValueError(msg)
    131 if copy or arr.dtype == object or dtype == object:
    132     # Explicit copy, or required since NumPy can't view from / to object.
--> 133     return arr.astype(dtype, copy=True)
    135 return arr.astype(dtype, copy=copy)
```

```
ValueError: could not convert string to float: '$6,466'
```

Hint

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

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

##