DSC540-T302 Data Preparation

Project Milestone 4

Saravanan Janarthanan

In [1]:
```
import json

with open('PolygonKey.json') as f:
    keys = json.load(f)
    Polygon_key = keys['Polygon_key']
```

Details of the json data structure that will be recieved

T*string The exchange symbol that this item is traded under.

c*number The close price for the symbol in the given time period.

h*number The highest price for the symbol in the given time period.

l*number The lowest price for the symbol in the given time period.

ninteger The number of transactions in the aggregate window.

o*number The open price for the symbol in the given time period.

otcboolean Whether or not this aggregate is for an OTC ticker. This field will be left off if false.

t*integer The Unix Msec timestamp for the end of the aggregate window.

v*number The trading volume of the symbol in the given time period.

vwnumber The volume weighted average price.

In [2]:
```
# import modules
import pandas as pd
import requests

# url to access the api,  using 5/15 date data
poly_url = "https://api.polygon.io/v2/aggs/grouped/locale/us/market/stocks/2024-05-15"
# define the url load params
url_param = {}
url_param['apikey'] = Polygon_key
url_param['adjusted'] = 'false'

poly_data = ""

# use the response module to get the JSON data
try:
    response = requests.get(url=poly_url, params=url_param)
```

```
    json_res_data = response.json()
    # check if the response is good,  status attribute has value'OK', if so load the a
    # that contains stock data
    if json_res_data['status'] == 'OK':
        poly_df = pd.DataFrame(json_res_data['results'])
    else:
        err_msg = "Error Occured while searching title " + title_name
        if 'Error' in list(json_res_data.keys()):
            err_msg = err_msg + " Error Message recieved : "+ json_res_data['Error']
        print(err_msg)
except Exception as e:
    print(f"Error while retrieving title details from OMDB : {e}")
    print(e)
```

In [3]:
```
# print the Dataframe
poly_df
```

Out[3]:

	T	v	vw	o	c	h	l	t	n
0	IAT	174090.0	43.5808	43.650	43.6500	43.950	43.330	1715803200000	2198.0
1	ALTI	150409.0	4.4940	4.660	4.4400	4.660	4.350	1715803200000	1836.0
2	POLA	201286.0	0.5142	0.507	0.5369	0.537	0.480	1715803200000	453.0
3	BALT	73089.0	29.4481	29.370	29.4900	29.495	29.370	1715803200000	343.0
4	CRON	2600970.0	2.9527	2.980	2.9000	3.010	2.890	1715803200000	8506.0
...	...	...	...	...	...	...	...	...	...
10495	ZTST	0.0	NaN	40.170	40.1700	40.170	40.170	1715803200000	NaN
10496	ZBZX	0.0	NaN	25.000	25.0000	25.000	25.000	1715803200000	NaN
10497	ZVZZT	211707.0	18.7798	20.250	39.8500	39.850	11.000	1715803200000	1413.0
10498	ZTEST	0.0	NaN	8662.955	8662.9550	8662.955	8662.955	1715803200000	NaN
10499	ZIEXT	1000.0	1.0000	1.000	1.0000	1.000	1.000	1715803200000	1.0

10500 rows × 9 columns

Step - 1

Change the header names

- Change header name 'T' as 'Symbol' in column 1 (index 0)
- Change 'v' as 'Volume' in column 2 (index 1)
- Change 'c' as 'Closing Price' in column 5 (index 4)
- Change 'h' as 'Highest Price' in column 6 (index 5)
- Change 'l' as 'Lowest Price' in column 7 (index 6)
- Change 't' as 'Timestamp' in column 8 (index 7)
- Change 'vw' as 'Vol Wt Avg Price' in column 3 (index 2)
- Change 'o' as 'Open Price' in column 4 (index 3)
- Change 'n' as 'Number of Transactions' in column 9 (index 8)

In [4]:
```
# Retrieve the header names to change them
header_nms = poly_df.columns.tolist()
```

In [5]:
```
header_nms
```

Out[5]:
```
['T', 'v', 'vw', 'o', 'c', 'h', 'l', 't', 'n']
```

In [6]:
```
# Map the new names to the right indexes as intended
header_nms[0] = 'Symbol'
header_nms[1] = 'Volume'
header_nms[2] = 'Vol Wt Avg Price'
header_nms[3] = 'Open Price'
header_nms[4] = 'Closing Price'
header_nms[5] = 'Highest Price'
header_nms[6] = 'Lowest Price'
header_nms[7] = 'Timestamp'
header_nms[8] = 'Number of Transactions'
```

In [7]:
```
# Update the new headernames to dataframe
poly_df.columns = header_nms
for idx, hdr in enumerate(poly_df.columns):
    print(idx, " : ", hdr)
```

```
0  :  Symbol
1  :  Volume
2  :  Vol Wt Avg Price
3  :  Open Price
4  :  Closing Price
5  :  Highest Price
6  :  Lowest Price
7  :  Timestamp
8  :  Number of Transactions
```

In [8]:
```
# print the first 5 rows of DataFrame
poly_df.head()
```

Out[8]:

	Symbol	Volume	Vol Wt Avg Price	Open Price	Closing Price	Highest Price	Lowest Price	Timestamp	Number of Transactions
0	IAT	174090.0	43.5808	43.650	43.6500	43.950	43.33	1715803200000	2198.0
1	ALTI	150409.0	4.4940	4.660	4.4400	4.660	4.35	1715803200000	1836.0
2	POLA	201286.0	0.5142	0.507	0.5369	0.537	0.48	1715803200000	453.0
3	BALT	73089.0	29.4481	29.370	29.4900	29.495	29.37	1715803200000	343.0
4	CRON	2600970.0	2.9527	2.980	2.9000	3.010	2.89	1715803200000	8506.0

In [9]:
```
# check the null value counts for each attribute
poly_df.isnull().sum()
```

Out[9]:
```
Symbol                     0
Volume                     0
Vol Wt Avg Price          74
Open Price                 0
Closing Price              0
Highest Price              0
Lowest Price               0
Timestamp                  0
Number of Transactions    74
dtype: int64
```

Step 2

Round the headers with prices to 2 decimal integers

Use the round method to round the values

Below colums values will be rounded to 2 decimals

- Open Price
- Closing Price
- Highest Price
- Lowest Price

In [10]:
```
# Reassign the column values after rounding them
poly_df['Open Price']    = poly_df['Open Price'].round(2)
poly_df['Closing Price'] = poly_df['Closing Price'].round(2)
poly_df['Highest Price'] = poly_df['Highest Price'].round(2)
poly_df['Lowest Price']  = poly_df['Lowest Price'].round(2)
```

In [12]:
```
poly_df['Vol Wt Avg Price']  = poly_df['Vol Wt Avg Price'].round(2)
```

In [11]:
```
# Check the changes by printing the first 5 rows
poly_df.head()
```

Out[11]:

	Symbol	Volume	Vol Wt Avg Price	Open Price	Closing Price	Highest Price	Lowest Price	Timestamp	Number of Transactions
0	IAT	174090.0	43.5808	43.65	43.65	43.95	43.33	1715803200000	2198.0
1	ALTI	150409.0	4.4940	4.66	4.44	4.66	4.35	1715803200000	1836.0
2	POLA	201286.0	0.5142	0.51	0.54	0.54	0.48	1715803200000	453.0
3	BALT	73089.0	29.4481	29.37	29.49	29.50	29.37	1715803200000	343.0
4	CRON	2600970.0	2.9527	2.98	2.90	3.01	2.89	1715803200000	8506.0

Step 3

Convert 'Volume' and "Number of Transactions" to intger values

In [13]:
```
import math
```

```
# fill the misssing values with 0 to avoid any conversion data format issues
poly_df['Number of Transactions'] = poly_df['Number of Transactions'].fillna(0)

# use astype to change the data type
poly_df['Volume']                 = poly_df['Volume'].round().astype(int)
poly_df['Number of Transactions']    = poly_df['Number of Transactions'].round().astyp
```

In [14]:
```
# Print the rows to check the change
poly_df.head()
```

Out[14]:

	Symbol	Volume	Vol Wt Avg Price	Open Price	Closing Price	Highest Price	Lowest Price	Timestamp	Number of Transactions
0	IAT	174090	43.58	43.65	43.65	43.95	43.33	1715803200000	2198
1	ALTI	150409	4.49	4.66	4.44	4.66	4.35	1715803200000	1836
2	POLA	201286	0.51	0.51	0.54	0.54	0.48	1715803200000	453
3	BALT	73089	29.45	29.37	29.49	29.50	29.37	1715803200000	343
4	CRON	2600970	2.95	2.98	2.90	3.01	2.89	1715803200000	8506

Step 4

Fill the average price of misssing values in 'Vol Wt Avg Price' with average price of 'Open price and 'Closing Price'

- Create a dummy column to fill the avg price for all columns
- Then using fillna method copy the new column values with avg price into na rows of 'Vol Wt Avg Price' column
- Drop the new column

In [15]:
```
# create  a new temp column to store the average price using mean value
poly_df['new_avg_price'] = poly_df[['Open Price', 'Closing Price']].mean(axis=1)
poly_df['new_avg_price'] = poly_df['new_avg_price'].round(2) # round the values to 2 d
```

In [16]:
```
# Copy the average price in the temp column to the null or NA rows values for 'Vol Wt
poly_df['Vol Wt Avg Price'] = poly_df['Vol Wt Avg Price'].fillna(poly_df['new_avg_pric
poly_df = poly_df.drop(columns=['new_avg_price'])
```

In [17]:
```
poly_df.head()
```

Out[17]:

	Symbol	Volume	Vol Wt Avg Price	Open Price	Closing Price	Highest Price	Lowest Price	Timestamp	Number of Transactions
0	IAT	174090	43.58	43.65	43.65	43.95	43.33	1715803200000	2198
1	ALTI	150409	4.49	4.66	4.44	4.66	4.35	1715803200000	1836
2	POLA	201286	0.51	0.51	0.54	0.54	0.48	1715803200000	453
3	BALT	73089	29.45	29.37	29.49	29.50	29.37	1715803200000	343
4	CRON	2600970	2.95	2.98	2.90	3.01	2.89	1715803200000	8506

In [18]:
```
# check if all the null values are filled
poly_df.isnull().sum()
```

Out[18]:
```
Symbol                  0
Volume                  0
Vol Wt Avg Price        0
Open Price              0
Closing Price           0
Highest Price           0
Lowest Price            0
Timestamp               0
Number of Transactions  0
dtype: int64
```

Step 5

Convert 'Timestamp' field with milli second values to a timezone field with date and time

In [19]:
```
# Convert timestamp in millis to regulat date, time format in UTC timezone format  ana
import pytz
poly_df['Timestamp'] = pd.to_datetime(poly_df['Timestamp'], unit='ms')
poly_df['Timestamp'] = poly_df['Timestamp'].dt.tz_localize('UTC')
poly_df['Timestamp'] = poly_df['Timestamp'].dt.tz_convert('US/Eastern')
```

In [20]:
```
# Print the dataframe to check the date format changes
poly_df.head()
```

Out[20]:

	Symbol	Volume	Vol Wt Avg Price	Open Price	Closing Price	Highest Price	Lowest Price	Timestamp	Number of Transactions
0	IAT	174090	43.58	43.65	43.65	43.95	43.33	2024-05-15 16:00:00-04:00	2198
1	ALTI	150409	4.49	4.66	4.44	4.66	4.35	2024-05-15 16:00:00-04:00	1836
2	POLA	201286	0.51	0.51	0.54	0.54	0.48	2024-05-15 16:00:00-04:00	453
3	BALT	73089	29.45	29.37	29.49	29.50	29.37	2024-05-15 16:00:00-04:00	343
4	CRON	2600970	2.95	2.98	2.90	3.01	2.89	2024-05-15 16:00:00-04:00	8506

Changes Made to the data

9 headers of the web content recieved was changed for better context and clarity.

- Change header name 'T' as 'Symbol' in column 1
- Change 'v' as 'Volume' in column 2
- Change 'c' as 'Closing Price' in column 5
- Change 'h' as 'Highest Price' in column 6
- Change 'l' as 'Lowest Price' in column 7
- Change 't' as 'Timestamp' in column 8
- Change 'vw' as 'Vol Wt Avg Price' in column 3
- Change 'o' as 'Open Price' in column 4
- Change 'n' as 'Number of Transactions' in column 9

The data supplied by individual companies to the exchange are utilized by polygon. Polygon API is open for free sunscription and uses the same data feeds recievd by all stock and investment sites

5 price columns values were rounded to 2 decimals as a cosmetic change and does not influence or change major values. Actual transactions use the right decimal values 2 volume and transactions values are converted to integer as these values cannot be in decimals but might have transformed win feeds Average price for missing rows were filled using high and low prices and no arbritarty values used, forward fill or backward fill cannot be used as no timeseries data was used. Tiemstamp changes were made for beetter clarity on date rather than milli seconds

The data originated from a API site whoch can be verified using accredited stock exchange websites

Given that essential data remains unchanged and is solely utilized for academic purposes, it does not impact the trading decisions of stock traders.

In [ ]: