Analysis of Daily Stock Price Data

KATE expects your code to define variables with specific names that correspond to certain things we are interested in.

KATE will run your notebook from top to bottom and check the latest value of those variables, so make sure you don't overwrite them.

- Remember to uncomment the line assigning the variable to your answer and don't change the variable or function names.
- Use copies of the original or previous DataFrames to make sure you do not overwrite them by mistake.

You will find instructions below about how to define each variable.

Once you're happy with your code, upload your notebook to KATE to check your feedback.

In [1]:

```
import pandas as pd
```

First, we will load the dataset from `data/AAPL.csv` into a DataFrame.

In [2]:

```
df = pd.read_csv('data/AAPL.csv')
df.head()
```

Out[2]:

```
         Date        Open        High         Low       Close   Adj Close    Volume
0  2015-06-30  125.570000  126.120003  124.860001  125.430000  115.597382  44370700
1  2015-07-01  126.900002  126.940002  125.989998  126.599998  116.675667  30238800
2  2015-07-02  126.430000  126.690002  125.769997  126.440002  116.528198  27211000
3  2015-07-06  124.940002  126.230003  124.849998  126.000000  116.122704  28060400
4  2015-07-07  125.889999  126.150002  123.769997  125.690002  115.837006  46946800
```

This data, in its raw format, is the same as that which can be retrieved from a number of financial websites.

Before starting the exercise, let's add some additional data columns, calculated from the raw data. Don't worry if you aren't familiar with the methods used in the following cell.

In [3]:

```
df['Date'] = pd.to_datetime(df['Date'])
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df['Weekday'] = df['Date'].dt.day_name()
df['Change %'] = (df['Adj Close'].pct_change() * 100)
```

In [4]:
```
df.head()
```

Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume	Year	Month
0	2015-06-30	125.570000	126.120003	124.860001	125.430000	115.597382	44370700	2015	6
1	2015-07-01	126.900002	126.940002	125.989998	126.599998	116.675667	30238800	2015	7
2	2015-07-02	126.430000	126.690002	125.769997	126.440002	116.528198	27211000	2015	7
3	2015-07-06	124.940002	126.230003	124.849998	126.000000	116.122704	28060400	2015	7
4	2015-07-07	125.889999	126.150002	123.769997	125.690002	115.837006	46946800	2015	7

Avoid modifying `df` itself in the subsequent questions.

Dataset stats

1. What's the mean of the values in the `Adj Close` column?

Store the answer in a variable called `mean_adj_close`

In [5]:
```
# Add your code below
df1 = df.copy()
df1
mean_adj_close = df1['Adj Close'].mean()
mean_adj_close
```

Out[5]:
```
167.04975667513898
```

2. What's the minimum value in the `Low` column?

Store the answer in a variable called `min_low`

In [6]:
```
# Add your code below
min_low = df1['Low'].min()
min_low
```

Out[6]:
```
89.470001
```

3. What's the maximum value in the `High` column?

Store the answer in a variable called `max_high`

In [7]:

```
# Add your code below
max_high = df1['High'].max()
max_high
```

Out[7]:

```
372.380005
```

4. What's the difference between `min_low` and `max_high` ?

Store the answer in a variable called `price_range`

In [8]:

```
# Add your code below
price_range = max_high-min_low
price_range
```

Out[8]:

```
282.91000399999996
```

5. How many rows are there in the DataFrame?

Store the answer in a variable called `entries`

In [9]:

```
# Add your code below
entries = df1.shape[0]
entries
```

Out[9]:

```
1259
```

6. On how many days (i.e. number of rows) was `Change %` greater than zero?

Store the answer in a variable called `positive_days`

In [10]:

```
# Add your code below
df2 = df.copy()
positive = df2['Change %'] >0
df3 = df2[positive]
positive_days = df3.shape[0]
positive_days
```

Out[10]:

```
671
```

7. On how many days (i.e. number of rows) has `Adj Close` been greater than the value in the final row?

Store the answer in a variable called `days_higher`

Hint: we can use list indexing with `.iloc` e.g. `.iloc[-1]` to get the last value in a Series, such as a specific column of a DataFrame

In [11]:

```
# Add your code below
#df2.iloc[-1:,5:6]
df3 = df1['Adj Close']
df4 = df3.iloc[-1]
days = df1['Adj Close']>df4
df5 = df2[days]
days_higher = df5.shape[0]
days_higher
```

Out[11]:

```
2
```

Dataset sorting and filtering

8. Create a new DataFrame called `df_2020` which is the same as `df` but contains only the rows where `Year == 2020`.

Use `set_index('Date', inplace=True)` to set the `Date` column as the row index.

In [12]:

```
# Add your code below
df2=df.copy()
date=df2.set_index('Date', inplace=True)
df2
date_mask = df2['Year']==2020
df_2020 = df2[date_mask]
df_2020
```

Out[12]:

	Open	High	Low	Close	Adj Close	Volume	Year	Month	D
Date									
2020-01-02	296.239990	300.600006	295.190002	300.350006	298.829956	33870100	2020	1	
2020-01-03	297.149994	300.579987	296.500000	297.429993	295.924713	36580700	2020	1	
2020-01-06	293.790009	299.959991	292.750000	299.799988	298.282715	29596800	2020	1	
2020-01-07	299.839996	300.899994	297.480011	298.390015	296.879883	27218000	2020	1	
2020-01-08	297.160004	304.440002	297.160004	303.190002	301.655548	33019800	2020	1	
...	...	...	...	...	...	...	...	...	
2020-06-23	364.000000	372.380005	362.269989	366.529999	366.529999	53038900	2020	6	
2020-06-24	365.000000	368.790009	358.519989	360.059998	360.059998	48155800	2020	6	
2020-06-25	360.700012	365.000000	357.570007	364.839996	364.839996	34380600	2020	6	
2020-06-26	364.410004	365.320007	353.019989	353.630005	353.630005	51314200	2020	6	
2020-06-29	353.250000	362.170013	351.279999	361.779999	361.779999	32579000	2020	6	

124 rows × 11 columns

9. Continuing with `df_2020`, calculate the `.mean()` of `Change %` for entries where `Weekday == Monday`.

Store the value in a variable called `mean_change_mon_2020`.

In [13]:

```
# Add your code below
#df_2020
weekday_mask = df_2020['Weekday']=='Monday'
monday = df_2020[weekday_mask]
monday
mean_change_mon_2020 = monday['Change %'].mean()
```

When you have calculated `mean_change_mon_2020`, uncomment and run the cell below to

In [14]:
```
mean_change_mon_2020
```

Out[14]:
```
0.2918877852311579
```

10. Calculate the sum of the `Volume` column in `df_2020` for entries where `Month == 3`.

Store the value in a variable called `total_volume_march_2020` .

In [15]:
```
# Add your code below
march_mask = df_2020['Month']==3
march = df_2020[march_mask]
march
total_volume_march_2020 = march['Volume'].sum()
```

When you have calculated `total_volume_march_2020` , uncomment and run the cell below to view its value:

In [16]:
```
total_volume_march_2020
```

Out[16]:
```
1570018100
```

11. Using `df_2020` , determine when `Adj Close` was the highest.

- look at the [documentation (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.idxmax.html)](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.idxmax.html) for the `.idxmax()` method and use it for this task
- this will only work if the row index has been set to the `Date` as instructed earlier in the assignment

Store the value in a variable called `year_high_timestamp`

In [17]:
```
# Add your code below
high=df_2020['Adj Close']
year_high_timestamp=high.idxmax()
year_high_timestamp
```

Out[17]:
```
Timestamp('2020-06-23 00:00:00')
```

12. Create a DataFrame called `df_top_10` which contains the 10 entries from `df` with the highest positive `Change %` values.

- consider all entries in `df` rather than `df_2020`
- remember to avoid modifying `df` or any other stored DataFrames
- `.copy()` can be used to copy a DataFrame to a new variable

In [18]:

```
# Add your code below
df3=df.copy()
sort=df3.sort_values(by='Change %', ascending=False)
df_top_10 = sort.head(10)
df_top_10
```

Out[18]:

	Date	Open	High	Low	Close	Adj Close	Volume	Year	M
1184	2020-03-13	264.890015	279.920013	252.949997	277.970001	277.219574	92683000	2020	
1191	2020-03-24	236.360001	247.690002	234.300003	246.880005	246.213516	71882800	2020	
1175	2020-03-02	282.279999	301.440002	277.720001	298.809998	298.003296	85349300	2020	
1200	2020-04-06	250.899994	263.109985	249.380005	262.470001	261.761414	50455100	2020	
1181	2020-03-10	277.140015	286.440002	269.369995	285.339996	284.569672	71322500	2020	
879	2018-12-26	148.300003	157.229996	146.720001	157.169998	154.059814	58582500	2018	
902	2019-01-30	163.250000	166.149994	160.229996	165.250000	161.979935	61109800	2019	
271	2016-07-27	104.269997	104.349998	102.750000	102.949997	96.822357	92344800	2016	
401	2017-02-01	127.029999	130.490005	127.010002	128.750000	122.367752	111985000	2017	
778	2018-08-01	199.130005	201.759995	197.309998	201.500000	196.137955	67935700	2018	

13. How many entries in `df_top_10` were *not* on a Monday?

Store the value in a variable called `top_10_not_mon`

In [19]:

```
# Add your code below
day_mask=df_top_10['Weekday']!='Monday'
frame=df_top_10[day_mask]
#frame
top_10_not_mon=len(frame)
```

When you have calculate `top_10_not_mon` , uncomment and run the cell below to inspect it:

In [20]:

```
top_10_not_mon
```

Out[20]:

```
8
```

Dataset manipulation

14. Create a new DataFrame called `df_var` , which the same as `df` but with an additional column `Variation %` , which is equal to:

((`High` - `Low`)/ `Close`) * 100

- be sure to use `Close` rather than `Adj Close` in this question
- do not modify `df` but create a copy: `df_var = df.copy()`

In [21]:

```
# Add your code below
df_var=df.copy()
df_var['Variation %']=(df_var['High']-df_var['Low'])/df_var['Close'] * 100
df_var
```

Out[21]:

	Date	Open	High	Low	Close	Adj Close	Volume	Year
0	2015-06-30	125.570000	126.120003	124.860001	125.430000	115.597382	44370700	2015
1	2015-07-01	126.900002	126.940002	125.989998	126.599998	116.675667	30238800	2015
2	2015-07-02	126.430000	126.690002	125.769997	126.440002	116.528198	27211000	2015
3	2015-07-06	124.940002	126.230003	124.849998	126.000000	116.122704	28060400	2015
4	2015-07-07	125.889999	126.150002	123.769997	125.690002	115.837006	46946800	2015
...	...	...	...	...	...	...	...	...
1254	2020-06-23	364.000000	372.380005	362.269989	366.529999	366.529999	53038900	2020

Once you have calculated `df_var` , you can uncomment and run the cell below to inspect it:

In [22]:

```
df_var.head()
```

Out[22]:

	Date	Open	High	Low	Close	Adj Close	Volume	Year	Month
0	2015-06-30	125.570000	126.120003	124.860001	125.430000	115.597382	44370700	2015	6
1	2015-07-01	126.900002	126.940002	125.989998	126.599998	116.675667	30238800	2015	7
2	2015-07-02	126.430000	126.690002	125.769997	126.440002	116.528198	27211000	2015	7
3	2015-07-06	124.940002	126.230003	124.849998	126.000000	116.122704	28060400	2015	7
4	2015-07-07	125.889999	126.150002	123.769997	125.690002	115.837006	46946800	2015	7

15. Create a new DataFrame called `df_var_value` , which the same as `df_var` but with an additional column `Traded Value` , equal to:

```
Volume * Adj Close
```

- do not modify `df_var` but create a copy: `df_var_value = df_var.copy()`

In [23]:
```
# Add your code below
df_var_value=df_var.copy()
df_var_value['Traded Value']=df_var_value['Volume']*df_var_value['Adj Close'
df_var_value
```

Out[23]:

	Date	Open	High	Low	Close	Adj Close	Volume	Year	Mc
0	2015-06-30	125.570000	126.120003	124.860001	125.430000	115.597382	44370700	2015	
1	2015-07-01	126.900002	126.940002	125.989998	126.599998	116.675667	30238800	2015	
2	2015-07-02	126.430000	126.690002	125.769997	126.440002	116.528198	27211000	2015	
3	2015-07-06	124.940002	126.230003	124.849998	126.000000	116.122704	28060400	2015	
4	2015-07-07	125.889999	126.150002	123.769997	125.690002	115.837006	46946800	2015	
...	...	...	...	...	...	...	...	...	
1254	2020-06-23	364.000000	372.380005	362.269989	366.529999	366.529999	53038900	2020	
1255	2020-06-24	365.000000	368.790009	358.519989	360.059998	360.059998	48155800	2020	
1256	2020-06-25	360.700012	365.000000	357.570007	364.839996	364.839996	34380600	2020	
1257	2020-06-26	364.410004	365.320007	353.019989	353.630005	353.630005	51314200	2020	
1258	2020-06-29	353.250000	362.170013	351.279999	361.779999	361.779999	32579000	2020	

1259 rows × 14 columns

Now uncomment and run the cell below to view `df_var_value` :

In [24]:

```
df_var_value.head()
```

Out[24]:

	Date	Open	High	Low	Close	Adj Close	Volume	Year	Month
0	2015-06-30	125.570000	126.120003	124.860001	125.430000	115.597382	44370700	2015	6
1	2015-07-01	126.900002	126.940002	125.989998	126.599998	116.675667	30238800	2015	7
2	2015-07-02	126.430000	126.690002	125.769997	126.440002	116.528198	27211000	2015	7
3	2015-07-06	124.940002	126.230003	124.849998	126.000000	116.122704	28060400	2015	7
4	2015-07-07	125.889999	126.150002	123.769997	125.690002	115.837006	46946800	2015	7

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: