In [1]:
```
#With faang, use type conversion to cast the values of the date column intodatetimes d
#load pd
import pandas as pd
#read faang
faang = pd.read_csv('faang.csv')

#make date column into datetime format
faang['date'] = pd.to_datetime(faang['date'])

#make volume column into integers
faang['volume'] = faang['volume'].astype(int)

#sort by date and ticker
faang = faang.sort_values(by=['date', 'ticker'])
print(faang.head())
```

```
           date         high          low         open        close  \
505  2019-01-02    39.712502    38.557499    38.722500    39.480000
0    2019-01-02  1553.359985  1460.930054  1465.199951  1539.130005
1010 2019-01-02  1052.319946  1015.710022  1016.570007  1045.849976
2020 2019-01-02   137.509995   128.559998   128.990005   135.679993
1515 2019-01-02   269.750000   256.579987   259.279999   267.660004

         volume    adj_close ticker
505   148158800    38.562561   AAPL
0       7983100  1539.130005   AMZN
1010    1532600  1045.849976   GOOG
2020   28146200   135.679993   META
1515   11679500   267.660004   NFLX
```

In [2]:
```
#Find the seven rows in faang with the lowest value for volume.
#load pd
import pandas as pd

#read faaang
faang = pd.read_csv('faang.csv')

#volume column to integers
faang['volume'] = faang['volume'].astype(int)

#seven rows with the lowest value for volume
lowest_volume_rows = faang.nsmallest(7, 'volume')
print(lowest_volume_rows)
```

```
            date         high          low         open        close  volume  \
1510  12/24/2020  1746.000000  1729.109985  1735.000000  1738.849976  346800
1257  12/24/2019  1350.260010  1342.780029  1348.500000  1343.560059  347500
1240  11/29/2019  1310.204956  1303.969971  1307.119995  1304.959961  587000
1258  12/26/2019  1361.327026  1344.469971  1346.170044  1360.400024  667500
1079   4/11/2019  1207.959961  1200.130005  1203.959961  1204.619995  710200
1078   4/10/2019  1203.785034  1196.435059  1200.680054  1202.160034  724600
1170   8/21/2019  1199.000000  1187.430054  1193.150024  1191.250000  740700

        adj_close ticker
1510  1738.849976   GOOG
1257  1343.560059   GOOG
1240  1304.959961   GOOG
1258  1360.400024   GOOG
1079  1204.619995   GOOG
1078  1202.160034   GOOG
1170  1191.250000   GOOG
```

In [3]:

```
#Right now, the data is somewhere between long and wide format. Use melt()to make it c
#Load pd
import pandas as pd

#read faang
faang = pd.read_csv('faang.csv')

#melt into long
faang_melted = faang.melt(id_vars=['date', 'ticker'],  value_vars=['open', 'high', 'lo
var_name='measurement', value_name='value')
print(faang_melted.head())
```

```
       date ticker measurement        value
0  1/2/2019   AMZN        open  1465.199951
1  1/3/2019   AMZN        open  1520.010010
2  1/4/2019   AMZN        open  1530.000000
3  1/7/2019   AMZN        open  1602.310059
4  1/8/2019   AMZN        open  1664.689941
```

Suppose we found out that on July 26, 2018 there was a glitch in how the data was recorded. How should we handle this? Note that there is no coding required for this exercise.

- We could start this with verifying the authenticity of the information regarding the glitch, sometimes false alarms can arise from misinformation. If it is found to be true, we would determine the extent and impact of the glitch. We would ask questions like does it affect just one data point, multiple points, or entire columns. We could then look unto different data correction options such as delete, imputation, back or forward fill ,or we can leave it as it just with added annotations. We would then document everything we have done up to that point. We would finish out by making sure if a fix was implemented, it was correct, and then if there were steps that could've been done to implement prevention, that we attempt t implement that every time.

In [ ]: