


QF 627 Programming and Computational Finance

Problem-Sets for Exercise 3 | Questions

"Hi Team! 🙋"

Here we go again! The current script is prepared to help you to further exercise course content (But, again, do note that these are not assessment questions. They are for your exercise.)

Some of the questions ask you to perform `reverse-engineering` . Instead of a question written in text format, you will be given the end results of some lines of code. Your task is to fill in the input cell with lines of code to create the output cell. Such a reverse-engineering exercise in computational programming will maximize your knowledge and skills in Computational Finance.

The questions below won't be that difficult, as you have learned how to answer them from the lessons and the assigned readings. As you will notice from the questions, my intention is not just to give questions, but also to give additional learning pointers. I hope this helps 😊

Enjoy!

First, let's do some quick exercise with stock prices data.

Activation of necessary libraries.

In [2]:

```
1 import numpy as np
2 import pandas as pd
3
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6
7 import datetime
8
9 import warnings
10 warnings.filterwarnings("ignore")
```

Let's try to import historical stock data from the web and from your machine.

How to import from the Web

You need to have `pandas_datareader` and `yfinance` module.

In [3]:

```
1 import pandas_datareader.data as web
2 import yfinance as yf
3
4 yf.pdr_override()
```

Let's start & end dates--we are interested in year 2023 until August.

In [4]:

```
1 start = datetime.datetime(2023, 1, 2)
2 end = datetime.datetime(2023, 8, 31)
```

Then, let's use [Yahoo! Finance \(https://sg.finance.yahoo.com/\)](https://sg.finance.yahoo.com/) for extracting historical stock data.

In [5]:

```
1 apple = \
2 ( web
3   .get_data_yahoo("AAPL",
4                   start,
5                   end)
6 )
7
8 google = \
9 ( web
10  .get_data_yahoo("GOOGL",
11                 start,
12                 end)
13 )
```

```
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
```

You may save the data to csv files.

In [6]:

```
1 apple.to_csv("apple_2023_stock.csv")
2 google.to_csv("google_2023_stock.csv")
```

Let's read the stored files.

In [7]:

```
1 apple_2023 =\
2 (pd
3  .read_csv("apple_2023_stock.csv",
4            index_col = 0,
5            parse_dates = True)
6  )
7
8 google_2023 =\
9 (pd
10  .read_csv("google_2023_stock.csv",
11           index_col = 0,
12           parse_dates = True)
13  )
```

Have a quick look at your data.

In [8]:

```
1 apple.head()
```

Out[8]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2023-01-03	130.279999	130.899994	124.169998	125.070000	124.538658	112117500
2023-01-04	126.889999	128.660004	125.080002	126.360001	125.823189	89113600
2023-01-05	127.129997	127.769997	124.760002	125.019997	124.488876	80962700
2023-01-06	126.010002	130.289993	124.889999	129.619995	129.069321	87754700
2023-01-09	130.470001	133.410004	129.889999	130.149994	129.597061	70790800

In [9]:

```
1 google.head()
```

Out[9]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2023-01-03	89.589996	91.050003	88.519997	89.120003	89.120003	28131200
2023-01-04	90.349998	90.650002	87.269997	88.080002	88.080002	34854800
2023-01-05	87.470001	87.570000	85.900002	86.199997	86.199997	27194400
2023-01-06	86.790001	87.690002	84.860001	87.339996	87.339996	41381500
2023-01-09	88.360001	90.050003	87.860001	88.019997	88.019997	29003900

In [10]:

```
1 apple.describe()
```

Out[10]:

	Open	High	Low	Close	Adj Close	Volume
count	166.000000	166.000000	166.000000	166.000000	166.000000	1.660000e+02
mean	166.993555	168.662289	165.748073	167.331205	166.976434	6.066744e+07
std	18.429099	18.195184	18.451313	18.227959	18.348919	1.815773e+07
min	126.010002	127.769997	124.169998	125.019997	124.488876	3.145820e+07
25%	152.405006	154.277496	150.852505	152.899998	152.482895	4.849780e+07
50%	169.389999	170.400002	167.709999	169.064995	168.603790	5.586760e+07
75%	181.210003	182.905003	179.082497	180.957504	180.713474	6.777278e+07
max	196.240005	198.229996	195.279999	196.449997	196.185074	1.543573e+08

In [11]:

```
1 google.describe()
```

Out[11]:

	Open	High	Low	Close	Adj Close	Volume
count	166.000000	166.000000	166.000000	166.000000	166.000000	1.660000e+02
mean	110.625181	112.190723	109.509759	110.915843	110.915843	3.517107e+07
std	14.212845	14.274273	14.131140	14.109641	14.109641	1.380574e+07
min	85.980003	87.570000	84.860001	86.199997	86.199997	1.446790e+07
25%	97.597502	100.032499	96.914999	98.972498	98.972498	2.670200e+07
50%	106.864998	108.084999	105.385002	107.614998	107.614998	3.156025e+07
75%	123.129999	124.792501	122.135000	123.517500	123.517500	3.751230e+07
max	134.779999	136.570007	134.070007	135.880005	135.880005	1.194550e+08

We will re-organize and re-shape stock data

Let's concatenate data in multiple DataFrame objects

Let's obtain Apple adjusted close data for July and August 2022

In [463]:

```
1 apple.columns
```

Out[463]:

```
Index(['Symbol', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

In [13]:

```
1 start_07 = datetime.datetime(2023, 7, 1)
2 end_07 = datetime.datetime(2023, 8, 1)
3 start_08 = datetime.datetime(2023, 8, 1)
4 end_08 = datetime.datetime(2023, 9, 1)
```

In [14]:

```
1 apple_07 = apple.loc[start_07:end_07][['Adj Close']]
2 apple_08 = apple.loc[start_08:end_08][['Adj Close']]
```

In [15]:

```
1 apple_07.head()
```

Out[15]:

	Adj Close
Date	
2023-07-03	192.200470
2023-07-05	191.071976
2023-07-06	191.551331
2023-07-07	190.422852
2023-07-10	188.355652

In [12]:

```
1
```

Out[12]:

Adj Close	
Date	
2023-07-03	192.200470
2023-07-05	191.071976
2023-07-06	191.551331
2023-07-07	190.422852
2023-07-10	188.355652

In [16]:

```
1 apple_08.head()
```

Out[16]:

Adj Close	
Date	
2023-08-01	195.346207
2023-08-02	192.320297
2023-08-03	190.912201
2023-08-04	181.744583
2023-08-07	178.608810

In [13]:

```
1
```

Out[13]:

Adj Close	
Date	
2023-08-01	195.346207
2023-08-02	192.320297
2023-08-03	190.912201
2023-08-04	181.744583
2023-08-07	178.608810

Question 1-1. Please combine the first three rows of each of apple_07 and apple_08

In [17]:

```
1 apple_78_h3 = pd.concat([apple_07.head(3), apple_08.head(3)])
2 apple_78_h3
```

Out[17]:

Adj Close	
Date	
2023-07-03	192.200470
2023-07-05	191.071976
2023-07-06	191.551331
2023-08-01	195.346207
2023-08-02	192.320297
2023-08-03	190.912201

In [14]:

```
1
```

Out[14]:

Adj Close	
Date	
2023-07-03	192.200470
2023-07-05	191.071976
2023-07-06	191.551331
2023-08-01	195.346207
2023-08-02	192.320297
2023-08-03	190.912201

Question 1-2. How would you extract only the 2023 August GOOGL values?

In [18]:

```
1 # Method 1
2
3 google\
4     .loc[start_08:end_08]\
5     ['Adj Close']\
6     .values
7
```

Out[18]:

```
array([131.55000305, 128.38000488, 128.44999695, 128.11000061,
       131.52999878, 131.3999939 , 129.66000366, 129.69000244,
       129.55999756, 131.33000183, 129.77999878, 128.69999695,
       129.91999817, 127.45999908, 128.36999512, 129.08000183,
       132.36999512, 129.77999878, 129.88000488, 131.00999451,
       134.57000732, 135.88000488])
```


In [19]:

```
1 # Method 2
2
3 google_08 = google.loc[start_08:end_08][['Adj Close']]
4 google_08
```

Out[19]:

Adj Close	
Date	
2023-08-01	131.550003
2023-08-02	128.380005
2023-08-03	128.449997
2023-08-04	128.110001
2023-08-07	131.529999
2023-08-08	131.399994
2023-08-09	129.660004
2023-08-10	129.690002
2023-08-11	129.559998
2023-08-14	131.330002
2023-08-15	129.779999
2023-08-16	128.699997
2023-08-17	129.919998
2023-08-18	127.459999
2023-08-21	128.369995
2023-08-22	129.080002
2023-08-23	132.369995
2023-08-24	129.779999
2023-08-25	129.880005
2023-08-28	131.009995
2023-08-29	134.570007
2023-08-30	135.880005

Question 1-3. Please concatenate the AAPL and GOOGL 2023 August stock data--there will be duplicate index labels

In [20]:

```
1 AAGG_08 =\  
2     pd.concat\  
3     (  
4         [apple_08,  
5           google_08]  
6     )  
7 AAGG_08
```

Out[20]:

Adj Close	
Date	
2023-08-01	195.346207
2023-08-02	192.320297
2023-08-03	190.912201
2023-08-04	181.744583
2023-08-07	178.608810
2023-08-08	179.557526
2023-08-09	177.949707
2023-08-10	177.729996
2023-08-11	177.789993
2023-08-14	179.460007
2023-08-15	177.449997
2023-08-16	176.570007
2023-08-17	174.000000
2023-08-18	174.490005
2023-08-21	175.839996
2023-08-22	177.229996
2023-08-23	181.119995
2023-08-24	176.380005
2023-08-25	178.610001
2023-08-28	180.190002
2023-08-29	184.119995
2023-08-30	187.649994
2023-08-01	131.550003
2023-08-02	128.380005
2023-08-03	128.449997
2023-08-04	128.110001
2023-08-07	131.529999
2023-08-08	131.399994
2023-08-09	129.660004
2023-08-10	129.690002
2023-08-11	129.559998
2023-08-14	131.330002
2023-08-15	129.779999
2023-08-16	128.699997
2023-08-17	129.919998
2023-08-18	127.459999

Adj Close	
Date	
2023-08-21	128.369995
2023-08-22	129.080002
2023-08-23	132.369995
2023-08-24	129.779999
2023-08-25	129.880005
2023-08-28	131.009995
2023-08-29	134.570007
2023-08-30	135.880005

Out[16]:

Adj Close	
Date	
2023-08-01	195.346207
2023-08-02	192.320297
2023-08-03	190.912201
2023-08-01	131.550003
2023-08-02	128.380005
2023-08-03	128.449997

Let's see the two records for data of 2023-08-03

```
In [21]:  
1 AAGG_08.loc[datetime.datetime(2023, 8, 3)]
```

Out[21]:

Adj Close	
Date	
2023-08-03	190.912201
2023-08-03	128.449997

In [17]:

1	
---	--

Out[17]:

	Adj Close
Date	
2023-08-03	190.912201
2023-08-03	128.449997

Question 1-4. Please concatenate to show a specification of the stock tickets being part of the index. This will help disambiguate the duplicate dates using a hierarchal index.

In [22]:

1	# Method 1
2	mid = len(AAGG_08) // 2
3	mid

Out[22]:

22

In [23]:

```
1 AAGG_08['ticker'] = ['AAPL'] * mid + ['GOOGL'] * mid
2 AAGG_08
```

Out[23]:

	Adj Close	ticker
Date		
2023-08-01	195.346207	AAPL
2023-08-02	192.320297	AAPL
2023-08-03	190.912201	AAPL
2023-08-04	181.744583	AAPL
2023-08-07	178.608810	AAPL
2023-08-08	179.557526	AAPL
2023-08-09	177.949707	AAPL
2023-08-10	177.729996	AAPL
2023-08-11	177.789993	AAPL
2023-08-14	179.460007	AAPL
2023-08-15	177.449997	AAPL
2023-08-16	176.570007	AAPL
2023-08-17	174.000000	AAPL
2023-08-18	174.490005	AAPL
2023-08-21	175.839996	AAPL
2023-08-22	177.229996	AAPL
2023-08-23	181.119995	AAPL
2023-08-24	176.380005	AAPL
2023-08-25	178.610001	AAPL
2023-08-28	180.190002	AAPL
2023-08-29	184.119995	AAPL
2023-08-30	187.649994	AAPL
2023-08-01	131.550003	GOOGL
2023-08-02	128.380005	GOOGL
2023-08-03	128.449997	GOOGL
2023-08-04	128.110001	GOOGL
2023-08-07	131.529999	GOOGL
2023-08-08	131.399994	GOOGL
2023-08-09	129.660004	GOOGL
2023-08-10	129.690002	GOOGL
2023-08-11	129.559998	GOOGL
2023-08-14	131.330002	GOOGL
2023-08-15	129.779999	GOOGL
2023-08-16	128.699997	GOOGL
2023-08-17	129.919998	GOOGL
2023-08-18	127.459999	GOOGL

	Adj Close	ticker
Date		
2023-08-21	128.369995	GOOGL
2023-08-22	129.080002	GOOGL
2023-08-23	132.369995	GOOGL
2023-08-24	129.779999	GOOGL
2023-08-25	129.880005	GOOGL
2023-08-28	131.009995	GOOGL
2023-08-29	134.570007	GOOGL
2023-08-30	135.880005	GOOGL

In [24]:

```
1 AAGG_08.set_index(['ticker', AAGG_08.index], inplace = True)  
2 AAGG_08
```

Out[24]:

Adj Close		
ticker	Date	
AAPL	2023-08-01	195.346207
	2023-08-02	192.320297
	2023-08-03	190.912201
	2023-08-04	181.744583
	2023-08-07	178.608810
	2023-08-08	179.557526
	2023-08-09	177.949707
	2023-08-10	177.729996
	2023-08-11	177.789993
	2023-08-14	179.460007
	2023-08-15	177.449997
	2023-08-16	176.570007
	2023-08-17	174.000000
	2023-08-18	174.490005
	2023-08-21	175.839996
	2023-08-22	177.229996
	2023-08-23	181.119995
	2023-08-24	176.380005
	2023-08-25	178.610001
	2023-08-28	180.190002
	2023-08-29	184.119995
	2023-08-30	187.649994

Adj Close		
ticker	Date	
GOOGL	2023-08-01	131.550003
	2023-08-02	128.380005
	2023-08-03	128.449997
	2023-08-04	128.110001
	2023-08-07	131.529999
	2023-08-08	131.399994
	2023-08-09	129.660004
	2023-08-10	129.690002
	2023-08-11	129.559998

```
1 # Method 2
2 AA_08 = AA_08.set_index([date, ticker], inplace = True)
3 AA_08['ticker'] = 'AAPL'
4 AA_08.set_index([date, ticker], AA_08.index, inplace = True)
5 AA_08
6
```

Out[30]:

Adj Close		
ticker	Date	
AAPL	2023-08-01	195.346207
	2023-08-02	192.320297
	2023-08-03	190.912201
	2023-08-04	181.744583
	2023-08-07	178.608810
	2023-08-08	179.557526
	2023-08-09	177.949707
	2023-08-10	177.729996
	2023-08-11	177.789993
	2023-08-14	179.460007
	2023-08-15	177.449997
	2023-08-16	176.570007
	2023-08-17	174.000000
	2023-08-18	174.490005
	2023-08-21	175.839996
	2023-08-22	177.229996
	2023-08-23	181.119995
	2023-08-24	176.380005
	2023-08-25	178.610001
	2023-08-28	180.190002
	2023-08-29	184.119995
	2023-08-30	187.649994
	2023-08-31	187.649994
	2023-09-01	187.649994
	2023-09-04	187.649994
	2023-09-05	187.649994
	2023-09-06	187.649994
	2023-09-07	187.649994
	2023-09-11	187.649994
	2023-09-12	187.649994
	2023-09-13	187.649994
	2023-09-14	187.649994
	2023-09-15	187.649994
	2023-09-18	187.649994
	2023-09-19	187.649994
	2023-09-20	187.649994
	2023-09-21	187.649994
	2023-09-22	187.649994
	2023-09-25	187.649994
	2023-09-26	187.649994
	2023-09-27	187.649994
	2023-09-28	187.649994
	2023-09-29	187.649994
	2023-09-30	187.649994

In [31]:

```
1 GG_08 = google_08.copy()
2 GG_08['ticker'] = 'GOOGL'
3 GG_08.set_index(['ticker', GG_08.index], inplace = True)
4 GG_08
```

Out[31]:

		Adj Close
ticker	Date	
GOOGL	2023-08-01	131.550003
	2023-08-02	128.380005
	2023-08-03	128.449997
	2023-08-04	128.110001
	2023-08-07	131.529999
	2023-08-08	131.399994
	2023-08-09	129.660004
	2023-08-10	129.690002
	2023-08-11	129.559998
	2023-08-14	131.330002
	2023-08-15	129.779999
	2023-08-16	128.699997
	2023-08-17	129.919998
	2023-08-18	127.459999
	2023-08-21	128.369995
	2023-08-22	129.080002
	2023-08-23	132.369995
	2023-08-24	129.779999
	2023-08-25	129.880005
	2023-08-28	131.009995
	2023-08-29	134.570007
	2023-08-30	135.880005

In [32]:

```
1 # Method 1 for slicing
2 AAGG_08.groupby('ticker').head(3)
3
```

Out[32]:

		Adj Close
ticker	Date	
AAPL	2023-08-01	195.346207
	2023-08-02	192.320297
	2023-08-03	190.912201
GOOGL	2023-08-01	131.550003
	2023-08-02	128.380005
	2023-08-03	128.449997

In [33]:

```
1 AAGG_08_m2 =\  
2     pd.concat\  
3     (  
4     [AA_08,  
5     GG_08]  
6     )  
7 AAGG_08_m2
```

Out[33]:

Adj Close		
ticker	Date	
AAPL	2023-08-01	195.346207
	2023-08-02	192.320297
	2023-08-03	190.912201
	2023-08-04	181.744583
	2023-08-07	178.608810
	2023-08-08	179.557526
	2023-08-09	177.949707
	2023-08-10	177.729996
	2023-08-11	177.789993
	2023-08-14	179.460007
	2023-08-15	177.449997
	2023-08-16	176.570007
	2023-08-17	174.000000
	2023-08-18	174.490005
	2023-08-21	175.839996
	2023-08-22	177.229996
	2023-08-23	181.119995
	2023-08-24	176.380005
	2023-08-25	178.610001
	2023-08-28	180.190002
	2023-08-29	184.119995
	2023-08-30	187.649994

Adj Close		
ticker	Date	
GOOGL	2023-08-01	131.550003
	2023-08-02	128.380005
	2023-08-03	128.449997
	2023-08-04	128.110001
	2023-08-07	131.529999
	2023-08-08	131.399994
	2023-08-09	129.660004
	2023-08-10	129.690002
	2023-08-11	129.559998

In [34]:

```
1 # Method 2 for slicing
2 AAGG_2023_08_11.loc[AAGG_2023_08_11.index.slice[:, start_08:datetime.datetime(2023, 8, 3)], ['Adj Close']]
```

GOOGL	2023-08-14	131.330002
	2023-08-15	129.779999
	2023-08-16	129.699997
	2023-08-17	129.919998
	2023-08-18	129.349999
	2023-08-21	128.380005
	2023-08-22	129.080002
	2023-08-23	131.559998
	2023-08-24	129.380005
GOOGL	2023-08-25	129.649997
	2023-08-28	131.009995
	2023-08-29	134.570007
	2023-08-30	135.880005

In [18]:

```
1
```

Out[18]:

Adj Close		
Date		
AAPL	2023-08-01	195.346207
	2023-08-02	192.320297
	2023-08-03	190.912201
GOOGL	2023-08-01	131.550003
	2023-08-02	128.380005
	2023-08-03	128.449997

Let's extract just GOOGL values using `.loc`

In [35]:

```
1 AAGG_08.loc['GOOGL'].head()
```

Out[35]:

	Adj Close
Date	
2023-08-01	131.550003
2023-08-02	128.380005
2023-08-03	128.449997
2023-08-04	128.110001
2023-08-07	131.529999

In [19]:

```
1
```

Out[19]:

	Adj Close
Date	
2023-08-01	131.550003
2023-08-02	128.380005
2023-08-03	128.449997

Question 1-5. Let’s show concatenation using two DataFrame's that each have two columns. pandas will align the data in columns by the column names (labels)

In [36]:

```
1 pd.concat([apple.head(3), google.tail(3)])
```

Out[36]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2023-01-03	130.279999	130.899994	124.169998	125.070000	124.538658	112117500
2023-01-04	126.889999	128.660004	125.080002	126.360001	125.823189	89113600
2023-01-05	127.129997	127.769997	124.760002	125.019997	124.488876	80962700
2023-08-28	131.309998	132.539993	130.139999	131.009995	131.009995	20543300
2023-08-29	132.240005	136.570007	132.240005	134.570007	134.570007	43075600
2023-08-30	134.779999	136.279999	134.070007	135.880005	135.880005	28315800

In [41]:

```
1 AAGG1_5_top = apple[['Adj Close', 'Volume']]
2 AAGG1_5_below = google[['Adj Close', 'Volume']]
3
```

In [42]:

```
1 AAGG1_5 = pd.concat([AAGG1_5_top, AAGG1_5_below])
2 AAGG1_5
```

Out[42]:

	Adj Close	Volume
Date		
2023-01-03	124.538658	112117500
2023-01-04	125.823189	89113600
2023-01-05	124.488876	80962700
2023-01-06	129.069321	87754700
2023-01-09	129.597061	70790800
...
2023-08-24	129.779999	28500700
2023-08-25	129.880005	26762900
2023-08-28	131.009995	20543300
2023-08-29	134.570007	43075600
2023-08-30	135.880005	28315800

332 rows × 2 columns

Let's see concatenation with DataFrame objects that do not have the same set of columns.

This will show pandas filling in NaN values.

In [40]:

```
1 google_a = google[['Adj Close']]
2 AAGG1_5_na = pd.concat([AAGG1_5_top, google_a])
3 AAGG1_5_na
```

Out[40]:

	Adj Close	Volume
Date		
2023-01-03	124.538658	112117500.0
2023-01-04	125.823189	89113600.0
2023-01-05	124.488876	80962700.0
2023-01-06	129.069321	87754700.0
2023-01-09	129.597061	70790800.0
...
2023-08-24	129.779999	NaN
2023-08-25	129.880005	NaN
2023-08-28	131.009995	NaN
2023-08-29	134.570007	NaN
2023-08-30	135.880005	NaN

332 rows × 2 columns

In [21]:

```
1
```

Out[21]:

	Adj Close	Volume
Date		
2023-01-03	124.538666	112117500.0
2023-01-04	125.823189	89113600.0
2023-01-05	124.488876	80962700.0
2023-01-06	129.069321	87754700.0
2023-01-09	129.597061	70790800.0
...
2023-08-24	129.779999	NaN
2023-08-25	129.880005	NaN
2023-08-28	131.009995	NaN
2023-08-29	134.570007	NaN
2023-08-30	135.880005	NaN

332 rows × 2 columns

Question 2-1. Please perform an inner join on the DataFrame's since google_a does not have a Volume column, pandas will not include that column in the result.

In [43]:

```
1 AAGG2_1 = pd.concat([AAGG1_5_top, google_a], axis=0, join='inner')
2 AAGG2_1
```

Out[43]:

Adj Close	
Date	
2023-01-03	124.538658
2023-01-04	125.823189
2023-01-05	124.488876
2023-01-06	129.069321
2023-01-09	129.597061
...	...
2023-08-24	129.779999
2023-08-25	129.880005
2023-08-28	131.009995
2023-08-29	134.570007
2023-08-30	135.880005

332 rows × 1 columns

In [44]:

```
1 # wrong method
2 AAGG2_1_ = pd.concat([AAGG1_5_top, google_a], axis=0).dropna(axis=1)
3 AAGG2_1_
```

Out[44]:

Adj Close	
Date	
2023-01-03	124.538658
2023-01-04	125.823189
2023-01-05	124.488876
2023-01-06	129.069321
2023-01-09	129.597061
...	...
2023-08-24	129.779999
2023-08-25	129.880005
2023-08-28	131.009995
2023-08-29	134.570007
2023-08-30	135.880005

332 rows × 1 columns

In [22]:

```
1
```

Out[22]:

Adj Close	
Date	
2023-01-03	124.538666
2023-01-04	125.823189
2023-01-05	124.488876
2023-01-06	129.069321
2023-01-09	129.597061
...	...
2023-08-24	129.779999
2023-08-25	129.880005
2023-08-28	131.009995
2023-08-29	134.570007
2023-08-30	135.880005

332 rows × 1 columns

Question 2-2. Then, please concatenate along the rows, causing duplicate columns to be created in the result.

In [46]:

```
1 # don't know why cannot get result.
2 # referring to previous vertical concat, there should not be a Volume Column.
3 # 水平合并时，inner join是看共同行；竖直合并时，inner join是看共同列。这里的volume应该是要提
4 AAGG2_2 = pd.concat([AAGG1_5_top['Adj Close'], google_a['Adj Close']],axis=1)
5 AAGG2_2
```

Out[46]:

	Adj Close	Adj Close
Date		
2023-01-03	124.538658	89.120003
2023-01-04	125.823189	88.080002
2023-01-05	124.488876	86.199997
2023-01-06	129.069321	87.339996
2023-01-09	129.597061	88.019997
...
2023-08-24	176.380005	129.779999
2023-08-25	178.610001	129.880005
2023-08-28	180.190002	131.009995
2023-08-29	184.119995	134.570007
2023-08-30	187.649994	135.880005

166 rows × 2 columns

In [47]:

```
1 # not so good method
2 common_columns = AAGG1_5_top.columns.intersection(google_a.columns)
3 AAGG2_2_ = pd.concat([AAGG1_5_top[common_columns], google_a[common_columns]], axis=1)
4 AAGG2_2_
```

Out[47]:

	Adj Close	Adj Close
Date		
2023-01-03	124.538658	89.120003
2023-01-04	125.823189	88.080002
2023-01-05	124.488876	86.199997
2023-01-06	129.069321	87.339996
2023-01-09	129.597061	88.019997
...
2023-08-24	176.380005	129.779999
2023-08-25	178.610001	129.880005
2023-08-28	180.190002	131.009995
2023-08-29	184.119995	134.570007
2023-08-30	187.649994	135.880005

166 rows × 2 columns

In [23]:

```
1
```

Out[23]:

	Adj Close	Adj Close
Date		
2023-01-03	124.538666	89.120003
2023-01-04	125.823189	88.080002
2023-01-05	124.488876	86.199997
2023-01-06	129.069321	87.339996
2023-01-09	129.597061	88.019997
...
2023-08-24	176.380005	129.779999
2023-08-25	178.610001	129.880005
2023-08-28	180.190002	131.009995
2023-08-29	184.119995	134.570007
2023-08-30	187.649994	135.880005

166 rows × 2 columns

Question 2-3. Let's concat along rows using two DataFrame objects with different number of rows.

This demonstrates how NaN values will be filled in those rows for `apple` which only has three rows as compared to 5 for `google`

In [52]:

```
1 apple5x2=AAGG1_5_top.iloc[:5]
2 google3x2=AAGG1_5_below.iloc[:3]
```

In [53]:

```
1 apple5x2
```

Out[53]:

	Adj Close	Volume
Date		
2023-01-03	124.538658	112117500
2023-01-04	125.823189	89113600
2023-01-05	124.488876	80962700
2023-01-06	129.069321	87754700
2023-01-09	129.597061	70790800

In [54]:

```
1 google3x2
```

Out[54]:

	Adj Close	Volume
Date		
2023-01-03	89.120003	28131200
2023-01-04	88.080002	34854800
2023-01-05	86.199997	27194400

In [55]:

```
1 AAGG2_3 = pd.concat([apple5x2, google3x2],axis=1, keys=('APPL', 'GOOGL'))
2 AAGG2_3
```

Out[55]:

Date	APPL		GOOGL	
	Adj Close	Volume	Adj Close	Volume
2023-01-03	124.538658	112117500	89.120003	28131200.0
2023-01-04	125.823189	89113600	88.080002	34854800.0
2023-01-05	124.488876	80962700	86.199997	27194400.0
2023-01-06	129.069321	87754700	NaN	NaN
2023-01-09	129.597061	70790800	NaN	NaN

In [56]:

```
1 # not so good method
2 apple2_3 = AAGG1_5_top.iloc[:5]
3 google2_3 = AAGG1_5_below.iloc[:3]
4
5 apple2_3_index = pd.MultiIndex.from_tuples([('APPL', 'Adj Close'), ('APPL', 'Volume')])
6 apple2_3.columns = apple2_3_index
7
8 google2_3_index = pd.MultiIndex.from_tuples([('GOOGL', 'Adj Close'), ('GOOGL', 'Volume')])
9 google2_3.columns = google2_3_index
```

In [57]:

```
1 AAGG2_3_ = pd.concat([apple2_3, google2_3],axis=1)
2 AAGG2_3_
```

Out[57]:

Date	AAPL		GOOGL	
	Adj Close	Volume	Adj Close	Volume
2023-01-03	124.538658	112117500	89.120003	28131200.0
2023-01-04	125.823189	89113600	88.080002	34854800.0
2023-01-05	124.488876	80962700	86.199997	27194400.0
2023-01-06	129.069321	87754700	NaN	NaN
2023-01-09	129.597061	70790800	NaN	NaN

In [24]:

1	
---	--

Out[24]:

	AAPL		GOOGL	
	Adj Close	Volume	Adj Close	Volume
Date				
2023-01-03	124.538666	112117500	89.120003	28131200.0
2023-01-04	125.823189	89113600	88.080002	34854800.0
2023-01-05	124.488876	80962700	86.199997	27194400.0
2023-01-06	129.069321	87754700	NaN	NaN
2023-01-09	129.597061	70790800	NaN	NaN

Question 2-4. You know that `inner join` can also be used along this axis.

Please include rows with index labels that do not exist in both DataFrame objects

In [58]:

1	AAGG2_4 = pd.concat([apple5x2, google3x2],axis=1, join='inner', keys=('AAPL', 'GOOGL'))
2	AAGG2_4

Out[58]:

	AAPL		GOOGL	
	Adj Close	Volume	Adj Close	Volume
Date				
2023-01-03	124.538658	112117500	89.120003	28131200
2023-01-04	125.823189	89113600	88.080002	34854800
2023-01-05	124.488876	80962700	86.199997	27194400

In [25]:

1	
---	--

Out[25]:

	AAPL		GOOGL	
	Adj Close	Volume	Adj Close	Volume
Date				
2023-01-03	124.538666	112117500	89.120003	28131200
2023-01-04	125.823189	89113600	88.080002	34854800
2023-01-05	124.488876	80962700	86.199997	27194400

Question 2-5. Please ignore indexes and just concatenate the data and have the result have a default integer index

In [59]:

```
1 apple5x2.iloc[:3, :1]
```

Out[59]:

	Adj Close
Date	
2023-01-03	124.538658
2023-01-04	125.823189
2023-01-05	124.488876

In [60]:

```
1 AAGG2_5 = pd.concat([apple5x2.iloc[:3, :1], google3x2.iloc[:, :1]], axis = 0, join='inner', ignore_index=True)
2
3 AAGG2_5
```

Out[60]:

	Adj Close
0	124.538658
1	125.823189
2	124.488876
3	89.120003
4	88.080002
5	86.199997

In [26]:

```
1
```

Out[26]:

	Adj Close
0	124.538666
1	125.823189
2	124.488876
3	89.120003
4	88.080002
5	86.199997

Now let's merging DataFrame objects

In [106]:

```
1 AAGG2_5_m = apple[['Adj Close']].reset_index()  
2 AAGG2_5_m
```

Out[106]:

	Date	Adj Close
0	2023-01-03	124.538658
1	2023-01-04	125.823189
2	2023-01-05	124.488876
3	2023-01-06	129.069321
4	2023-01-09	129.597061
...
161	2023-08-24	176.380005
162	2023-08-25	178.610001
163	2023-08-28	180.190002
164	2023-08-29	184.119995
165	2023-08-30	187.649994

166 rows × 2 columns

In []:

```
1
```

In [27]:

1	
---	--

Out[27]:

	Date	Adj Close
0	2023-01-03	124.538666
1	2023-01-04	125.823189
2	2023-01-05	124.488876
3	2023-01-06	129.069321
4	2023-01-09	129.597061
...
161	2023-08-24	176.380005
162	2023-08-25	178.610001
163	2023-08-28	180.190002
164	2023-08-29	184.119995
165	2023-08-30	187.649994

166 rows × 2 columns

Question 3-1. Please merge the two DataFrame objects, so lets peek at the data to remind ourselves of what they contain.

pandas finds the columns in common, in this case Date, and merges on that column and adds a column for all the other columns in both DataFrames.

In [114]:

```
1 AAGG_3 = apple.reset_index()
2 AAGG_3
```

Out[114]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2023-01-03	130.279999	130.899994	124.169998	125.070000	124.538658	112117500
1	2023-01-04	126.889999	128.660004	125.080002	126.360001	125.823189	89113600
2	2023-01-05	127.129997	127.769997	124.760002	125.019997	124.488876	80962700
3	2023-01-06	126.010002	130.289993	124.889999	129.619995	129.069321	87754700
4	2023-01-09	130.470001	133.410004	129.889999	130.149994	129.597061	70790800
...
161	2023-08-24	180.669998	181.100006	176.009995	176.380005	176.380005	54945800
162	2023-08-25	177.380005	179.149994	175.820007	178.610001	178.610001	51449600
163	2023-08-28	180.089996	180.589996	178.550003	180.190002	180.190002	43820700
164	2023-08-29	179.699997	184.899994	179.500000	184.119995	184.119995	53003900
165	2023-08-30	184.940002	187.850006	184.740005	187.649994	187.649994	60813900

166 rows × 7 columns

In [117]:

```
1 AAGG_l = AAGG_3[['Date', 'Adj Close']]
2 AAGG_r = AAGG_3[['Date', 'Volume']]
3 AAGG_r
```

Out[117]:

	Date	Volume
0	2023-01-03	112117500
1	2023-01-04	89113600
2	2023-01-05	80962700
3	2023-01-06	87754700
4	2023-01-09	70790800
...
161	2023-08-24	54945800
162	2023-08-25	51449600
163	2023-08-28	43820700
164	2023-08-29	53003900
165	2023-08-30	60813900

166 rows × 2 columns

In [125]:

```
1 AAGG3_l = pd.merge(AAGG_l.loc[:,4], AAGG_r)
2 AAGG3_l
```

Out[125]:

	Date	Adj Close	Volume
0	2023-01-03	124.538658	112117500
1	2023-01-04	125.823189	89113600
2	2023-01-05	124.488876	80962700
3	2023-01-06	129.069321	87754700
4	2023-01-09	129.597061	70790800

In [28]:

```
1
```

Out[28]:

	Date	Adj Close	Volume
0	2023-01-03	124.538666	112117500
1	2023-01-04	125.823189	89113600
2	2023-01-05	124.488876	80962700
3	2023-01-06	129.069321	87754700
4	2023-01-09	129.597061	70790800

Question 3-2. Please use the merge function to create the below.

In [120]:

```
1 AAGG3_1.loc[2:3]
```

Out[120]:

	Date	Adj Close	Volume
2	2023-01-05	124.488876	80962700
3	2023-01-06	129.069321	87754700

In [121]:

```
1 AAGG3_2 = AAGG_1.loc[2:3].merge(AAGG_r)
2 AAGG3_2
```

Out[121]:

	Date	Adj Close	Volume
0	2023-01-05	124.488876	80962700
1	2023-01-06	129.069321	87754700

In [31]:

```
1
```

Out[31]:

	Date	Adj Close	Volume
0	2023-01-05	124.488876	80962700
1	2023-01-06	129.069321	87754700

Question 3-3. How would you create the below then?

In [124]:

```
1 AAGG3_2 = AAGG_l.loc[:,4].merge(AAGG_r.loc[2:3], how = 'outer')
2 AAGG3_2
```

Out[124]:

	Date	Adj Close	Volume
0	2023-01-03	124.538658	NaN
1	2023-01-04	125.823189	NaN
2	2023-01-05	124.488876	80962700.0
3	2023-01-06	129.069321	87754700.0
4	2023-01-09	129.597061	NaN

In [32]:

```
1
```

Out[32]:

	Date	Adj Close	Volume
0	2023-01-03	124.538666	NaN
1	2023-01-04	125.823189	NaN
2	2023-01-05	124.488876	80962700.0
3	2023-01-06	129.069321	87754700.0
4	2023-01-09	129.597061	NaN

Let's do some pivoting

Question 4-1. Please insert Symbol column before combining

In [150]:

```
1 apple4 = apple
2 #.reset_index()
3 apple4.insert(0, 'Symbol', ['AAPL'] * len(apple))
4 apple4
```

Out[150]:

	Symbol	Open	High	Low	Close	Adj Close	Volume
Date							
2023-01-03	AAPL	130.279999	130.899994	124.169998	125.070000	124.538658	112117500
2023-01-04	AAPL	126.889999	128.660004	125.080002	126.360001	125.823189	89113600
2023-01-05	AAPL	127.129997	127.769997	124.760002	125.019997	124.488876	80962700
2023-01-06	AAPL	126.010002	130.289993	124.889999	129.619995	129.069321	87754700
2023-01-09	AAPL	130.470001	133.410004	129.889999	130.149994	129.597061	70790800
...
2023-08-24	AAPL	180.669998	181.100006	176.009995	176.380005	176.380005	54945800
2023-08-25	AAPL	177.380005	179.149994	175.820007	178.610001	178.610001	51449600
2023-08-28	AAPL	180.089996	180.589996	178.550003	180.190002	180.190002	43820700
2023-08-29	AAPL	179.699997	184.899994	179.500000	184.119995	184.119995	53003900
2023-08-30	AAPL	184.940002	187.850006	184.740005	187.649994	187.649994	60813900

166 rows × 7 columns

In [154]:

```
1 google4 = google
2 #.reset_index()
3 google4.insert(0, 'Symbol', ['GOOGL'] * len(google))
4 google4
```

Out[154]:

	Symbol	Open	High	Low	Close	Adj Close	Volume
Date							
2023-01-03	GOOGL	89.589996	91.050003	88.519997	89.120003	89.120003	28131200
2023-01-04	GOOGL	90.349998	90.650002	87.269997	88.080002	88.080002	34854800
2023-01-05	GOOGL	87.470001	87.570000	85.900002	86.199997	86.199997	27194400
2023-01-06	GOOGL	86.790001	87.690002	84.860001	87.339996	87.339996	41381500
2023-01-09	GOOGL	88.360001	90.050003	87.860001	88.019997	88.019997	29003900
...
2023-08-24	GOOGL	133.949997	134.250000	129.570007	129.779999	129.779999	28500700
2023-08-25	GOOGL	129.539993	130.759995	127.250000	129.880005	129.880005	26762900
2023-08-28	GOOGL	131.309998	132.539993	130.139999	131.009995	131.009995	20543300
2023-08-29	GOOGL	132.240005	136.570007	132.240005	134.570007	134.570007	43075600
2023-08-30	GOOGL	134.779999	136.279999	134.070007	135.880005	135.880005	28315800

166 rows × 7 columns

In []:

```
1
```

In []:

```
1
```

In [33]:

```
1
```

Question 4-2. Please concatenate the Apple and Google data index will consist of the Date column, which we will sort.

In [160]:

```
1 google4
```

Out[160]:

	Symbol	Open	High	Low	Close	Adj Close	Volume
Date							
2023-01-03	GOOGL	89.589996	91.050003	88.519997	89.120003	89.120003	28131200
2023-01-04	GOOGL	90.349998	90.650002	87.269997	88.080002	88.080002	34854800
2023-01-05	GOOGL	87.470001	87.570000	85.900002	86.199997	86.199997	27194400
2023-01-06	GOOGL	86.790001	87.690002	84.860001	87.339996	87.339996	41381500
2023-01-09	GOOGL	88.360001	90.050003	87.860001	88.019997	88.019997	29003900
...
2023-08-24	GOOGL	133.949997	134.250000	129.570007	129.779999	129.779999	28500700
2023-08-25	GOOGL	129.539993	130.759995	127.250000	129.880005	129.880005	26762900
2023-08-28	GOOGL	131.309998	132.539993	130.139999	131.009995	131.009995	20543300
2023-08-29	GOOGL	132.240005	136.570007	132.240005	134.570007	134.570007	43075600
2023-08-30	GOOGL	134.779999	136.279999	134.070007	135.880005	135.880005	28315800

166 rows × 7 columns

In [263]:

```
1 AAGG4_2 = pd.concat([apple4, google4], axis = 0).sort_index()
2 AAGG4_2
```

Out[263]:

	Symbol	Open	High	Low	Close	Adj Close	Volume
Date							
2023-01-03	AAPL	130.279999	130.899994	124.169998	125.070000	124.538658	112117500
2023-01-03	GOOGL	89.589996	91.050003	88.519997	89.120003	89.120003	28131200
2023-01-04	GOOGL	90.349998	90.650002	87.269997	88.080002	88.080002	34854800
2023-01-04	AAPL	126.889999	128.660004	125.080002	126.360001	125.823189	89113600
2023-01-05	AAPL	127.129997	127.769997	124.760002	125.019997	124.488876	80962700
...
2023-08-28	AAPL	180.089996	180.589996	178.550003	180.190002	180.190002	43820700
2023-08-29	GOOGL	132.240005	136.570007	132.240005	134.570007	134.570007	43075600
2023-08-29	AAPL	179.699997	184.899994	179.500000	184.119995	184.119995	53003900
2023-08-30	AAPL	184.940002	187.850006	184.740005	187.649994	187.649994	60813900
2023-08-30	GOOGL	134.779999	136.279999	134.070007	135.880005	135.880005	28315800

332 rows × 7 columns

In [35]:

```
1
```

Out[35]:

	Symbol	Open	High	Low	Close	Adj Close	Volume
Date							
2023-01-03	AAPL	130.279999	130.899994	124.169998	125.070000	124.538666	112117500
2023-01-03	GOOGL	89.589996	91.050003	88.519997	89.120003	89.120003	28131200
2023-01-04	GOOGL	90.349998	90.650002	87.269997	88.080002	88.080002	34854800
2023-01-04	AAPL	126.889999	128.660004	125.080002	126.360001	125.823189	89113600
2023-01-05	AAPL	127.129997	127.769997	124.760002	125.019997	124.488876	80962700
...
2023-08-28	AAPL	180.089996	180.589996	178.550003	180.190002	180.190002	43820700
2023-08-29	GOOGL	132.240005	136.570007	132.240005	134.570007	134.570007	43075600
2023-08-29	AAPL	179.699997	184.899994	179.500000	184.119995	184.119995	53003900
2023-08-30	AAPL	184.940002	187.850006	184.740005	187.649994	187.649994	60813900
2023-08-30	GOOGL	134.779999	136.279999	134.070007	135.880005	135.880005	28315800

332 rows × 7 columns

Question 4-3. Please pushes the index into a column and resets to a default integer index.

In [264]:

```
1 AAGG4_3 = AAGG4_2.reset_index()
2 AAGG4_3.head()
```

Out[264]:

	Date	Symbol	Open	High	Low	Close	Adj Close	Volume
0	2023-01-03	AAPL	130.279999	130.899994	124.169998	125.070000	124.538658	112117500
1	2023-01-03	GOOGL	89.589996	91.050003	88.519997	89.120003	89.120003	28131200
2	2023-01-04	GOOGL	90.349998	90.650002	87.269997	88.080002	88.080002	34854800
3	2023-01-04	AAPL	126.889999	128.660004	125.080002	126.360001	125.823189	89113600
4	2023-01-05	AAPL	127.129997	127.769997	124.760002	125.019997	124.488876	80962700

In [36]:

1	
---	--

Out[36]:

	Date	Symbol	Open	High	Low	Close	Adj Close	Volume
0	2023-01-03	AAPL	130.279999	130.899994	124.169998	125.070000	124.538666	112117500
1	2023-01-03	GOOGL	89.589996	91.050003	88.519997	89.120003	89.120003	28131200
2	2023-01-04	GOOGL	90.349998	90.650002	87.269997	88.080002	88.080002	34854800
3	2023-01-04	AAPL	126.889999	128.660004	125.080002	126.360001	125.823189	89113600
4	2023-01-05	AAPL	127.129997	127.769997	124.760002	125.019997	124.488876	80962700

Question 4-4. Please pivot Date into the Index, make the columns match the unique values in the Symbol column, and the values will be the AdjClose values

In [266]:

```
1 AAGG4_4 = \
2     (
3         AAGG4_3
4     #     .reset_index()
5         .pivot(index = "Date",
6                 columns = "Symbol",
7                 values = "Adj Close")
8     )
9 AAGG4_4
```

Out[266]:

Symbol	AAPL	GOOGL
Date		
2023-01-03	124.538658	89.120003
2023-01-04	125.823189	88.080002
2023-01-05	124.488876	86.199997
2023-01-06	129.069321	87.339996
2023-01-09	129.597061	88.019997
...
2023-08-24	176.380005	129.779999
2023-08-25	178.610001	129.880005
2023-08-28	180.190002	131.009995
2023-08-29	184.119995	134.570007
2023-08-30	187.649994	135.880005

166 rows × 2 columns

In [38]:

1	
---	--

Out[38]:

Symbol	AAPL	GOOGL
Date		
2023-01-03	124.538666	89.120003
2023-01-04	125.823189	88.080002
2023-01-05	124.488876	86.199997
2023-01-06	129.069321	87.339996
2023-01-09	129.597061	88.019997
...
2023-08-24	176.380005	129.779999
2023-08-25	178.610001	129.880005
2023-08-28	180.190002	131.009995
2023-08-29	184.119995	134.570007
2023-08-30	187.649994	135.880005

166 rows × 2 columns

Let's execute `stacking` and `unstacking`

Question 5-1. Please stack the first level of columns into the index.

Let's move **AAPL** and **GOOGL** into the index leaving a single colum which is the **AdjClose** values

In [268]:

1	AAGG5_1 = AAGG4_4.stack(level=0)
2	AAGG5_1

Out[268]:

Date	Symbol	
2023-01-03	AAPL	124.538658
	GOOGL	89.120003
2023-01-04	AAPL	125.823189
	GOOGL	88.080002
2023-01-05	AAPL	124.488876
	...	
2023-08-28	GOOGL	131.009995
2023-08-29	AAPL	184.119995
	GOOGL	134.570007
2023-08-30	AAPL	187.649994
	GOOGL	135.880005

Length: 332, dtype: float64

In [39]:

1

Out[39]:

Date	Symbol	
2023-01-03	AAPL	124.538666
	GOOGL	89.120003
2023-01-04	AAPL	125.823189
	GOOGL	88.080002
2023-01-05	AAPL	124.488876
	...	
2023-08-28	GOOGL	131.009995
2023-08-29	AAPL	184.119995
	GOOGL	134.570007
2023-08-30	AAPL	187.649994
	GOOGL	135.880005

Length: 332, dtype: float64

Using `.loc` we can retrieve close values by specifying both the date and ticker

In [269]:

1 AAGG5_1.loc[datetime.datetime(2023, 1, 4), 'AAPL']

Out[269]:

125.82318878173828

In [40]:

1

Out[40]:

125.82318878173828

Question 5-2. Please lookup on just the date, which will give us two values one each for AAPL and GOOGL.

In [270]:

1 AAGG5_1.loc[datetime.datetime(2023, 1, 4)]

Out[270]:

Symbol	
AAPL	125.823189
GOOGL	88.080002

dtype: float64

In [41]:

```
1
```

Out[41]:

Symbol
AAPL 125.823189
GOOGL 88.080002
dtype: float64

Let's have a look at all values for the GOOGL symbol

In [271]:

```
1 AAGG5_1.loc[:, 'GOOGL']
```

Out[271]:

Date
2023-01-03 89.120003
2023-01-04 88.080002
2023-01-05 86.199997
2023-01-06 87.339996
2023-01-09 88.019997
...
2023-08-24 129.779999
2023-08-25 129.880005
2023-08-28 131.009995
2023-08-29 134.570007
2023-08-30 135.880005
Length: 166, dtype: float64

In [42]:

```
1
```

Out[42]:

Date
2023-01-03 89.120003
2023-01-04 88.080002
2023-01-05 86.199997
2023-01-06 87.339996
2023-01-09 88.019997
...
2023-08-24 129.779999
2023-08-25 129.880005
2023-08-28 131.009995
2023-08-29 134.570007
2023-08-30 135.880005
Length: 166, dtype: float64

Question 5-3. Please pivot the last level of the index back into a column

In [272]:

```
1 AAGG5_1
```

Out[272]:

Date	Symbol	
2023-01-03	AAPL	124.538658
	GOOGL	89.120003
2023-01-04	AAPL	125.823189
	GOOGL	88.080002
2023-01-05	AAPL	124.488876
...		
2023-08-28	GOOGL	131.009995
2023-08-29	AAPL	184.119995
	GOOGL	134.570007
2023-08-30	AAPL	187.649994
	GOOGL	135.880005
Length: 332, dtype: float64		

In [273]:

```
1 AAGG5_3 =\  
2 (  
3     AAGG5_1  
4     .reset_index(level = -1)# 改回Date做行index level = -1是选 Date,-2是选Symbol  
5     .pivot(columns='Symbol', values=0) #Symbol back to column name  
6 )  
7 AAGG5_3
```

Out[273]:

Symbol	AAPL	GOOGL
Date		
2023-01-03	124.538658	89.120003
2023-01-04	125.823189	88.080002
2023-01-05	124.488876	86.199997
2023-01-06	129.069321	87.339996
2023-01-09	129.597061	88.019997
...
2023-08-24	176.380005	129.779999
2023-08-25	178.610001	129.880005
2023-08-28	180.190002	131.009995
2023-08-29	184.119995	134.570007
2023-08-30	187.649994	135.880005
166 rows × 2 columns		

In [44]:

```
1
```

Out[44]:

Symbol	AAPL	GOOGL
Date		
2023-01-03	124.538666	89.120003
2023-01-04	125.823189	88.080002
2023-01-05	124.488876	86.199997

Now, let's do some `melt`

Question 6-1. Please use `melt` to make `id_vars` of `Date` and `Symbol`, making the column names the variable and the for each the value

In [299]:

```
1 AAGG4_2
```

Out[299]:

	Symbol	Open	High	Low	Close	Adj Close	Volume
Date							
2023-01-03	AAPL	130.279999	130.899994	124.169998	125.070000	124.538658	112117500
2023-01-03	GOOGL	89.589996	91.050003	88.519997	89.120003	89.120003	28131200
2023-01-04	GOOGL	90.349998	90.650002	87.269997	88.080002	88.080002	34854800
2023-01-04	AAPL	126.889999	128.660004	125.080002	126.360001	125.823189	89113600
2023-01-05	AAPL	127.129997	127.769997	124.760002	125.019997	124.488876	80962700
...
2023-08-28	AAPL	180.089996	180.589996	178.550003	180.190002	180.190002	43820700
2023-08-29	GOOGL	132.240005	136.570007	132.240005	134.570007	134.570007	43075600
2023-08-29	AAPL	179.699997	184.899994	179.500000	184.119995	184.119995	53003900
2023-08-30	AAPL	184.940002	187.850006	184.740005	187.649994	187.649994	60813900
2023-08-30	GOOGL	134.779999	136.279999	134.070007	135.880005	135.880005	28315800

332 rows × 7 columns

In [301]:

```
1 AAGG6_1 =\  
2     AAGG4_2\  
3     .reset_index()\  
4     .melt(id_vars=['Date', 'Symbol'], var_name='variable', value_name='value')  
5  
6 AAGG6_1
```

Out[301]:

	Date	Symbol	variable	value
0	2023-01-03	AAPL	Open	1.302800e+02
1	2023-01-03	GOOGL	Open	8.959000e+01
2	2023-01-04	GOOGL	Open	9.035000e+01
3	2023-01-04	AAPL	Open	1.268900e+02
4	2023-01-05	AAPL	Open	1.271300e+02
...
1987	2023-08-28	AAPL	Volume	4.382070e+07
1988	2023-08-29	GOOGL	Volume	4.307560e+07
1989	2023-08-29	AAPL	Volume	5.300390e+07
1990	2023-08-30	AAPL	Volume	6.081390e+07
1991	2023-08-30	GOOGL	Volume	2.831580e+07

1992 rows × 4 columns

In [46]:

```
1
```

Out[46]:

	Date	Symbol	variable	value
0	2023-01-03	AAPL	Open	130.279999
1	2023-01-03	GOOGL	Open	89.589996
2	2023-01-04	GOOGL	Open	90.349998
3	2023-01-04	AAPL	Open	126.889999
4	2023-01-05	AAPL	Open	127.129997

Question 6-2. Please extract the values for the data for Google on 2023-08-24

In [302]:

```
1 AAGG6_2 =\
2     AAGG6_1\
3     .loc[(AAGG6_1['Date'] == '2023-08-24') &\
4           (AAGG6_1['Symbol'] == 'GOOGL')]
5 AAGG6_2
```

Out[302]:

	Date	Symbol	variable	value
322	2023-08-24	GOOGL	Open	1.339500e+02
654	2023-08-24	GOOGL	High	1.342500e+02
986	2023-08-24	GOOGL	Low	1.295700e+02
1318	2023-08-24	GOOGL	Close	1.297800e+02
1650	2023-08-24	GOOGL	Adj Close	1.297800e+02
1982	2023-08-24	GOOGL	Volume	2.850070e+07

In [47]:

```
1
```

Out[47]:

	Date	Symbol	variable	value
322	2023-08-24	GOOGL	Open	1.339500e+02
654	2023-08-24	GOOGL	High	1.342500e+02
986	2023-08-24	GOOGL	Low	1.295700e+02
1318	2023-08-24	GOOGL	Close	1.297800e+02
1650	2023-08-24	GOOGL	Adj Close	1.297800e+02
1982	2023-08-24	GOOGL	Volume	2.850070e+07

Let's do some grouping and aggregation

To do so, we will do some splitting first.

Let's construct a DataFrame to demonstrate splitting

Question 7-1. Please extract from combined the Symbol and AdjClose, and reset the index

In [309]:

```
1 AAGG7_1 = AAGG4_2[['Symbol', 'Adj Close']].reset_index()
2 AAGG7_1
```

Out[309]:

	Date	Symbol	Adj Close
0	2023-01-03	AAPL	124.538658
1	2023-01-03	GOOGL	89.120003
2	2023-01-04	GOOGL	88.080002
3	2023-01-04	AAPL	125.823189
4	2023-01-05	AAPL	124.488876
...
327	2023-08-28	AAPL	180.190002
328	2023-08-29	GOOGL	134.570007
329	2023-08-29	AAPL	184.119995
330	2023-08-30	AAPL	187.649994
331	2023-08-30	GOOGL	135.880005

332 rows × 3 columns

In [48]:

1	
---	--

Out[48]:

	Date	Symbol	Adj Close
0	2023-01-03	AAPL	124.538666
1	2023-01-03	GOOGL	89.120003
2	2023-01-04	GOOGL	88.080002
3	2023-01-04	AAPL	125.823189
4	2023-01-05	AAPL	124.488876
...
327	2023-08-28	AAPL	180.190002
328	2023-08-29	GOOGL	134.570007
329	2023-08-29	AAPL	184.119995
330	2023-08-30	AAPL	187.649994
331	2023-08-30	GOOGL	135.880005

332 rows × 3 columns

Question 7-2. Please add two columns, year and month, using the year and month portions of the data as integers

In [314]:

```
1 AAGG7_2 = AAGG7_1.copy()
2 # .dt 是 Pandas Series 的一个属性，用于访问 Series 中日期时间相关的方法和属性。
3 # 在这种情况下，.dt.year 用于提取 Series 中每个日期时间值的年份部分。
4 AAGG7_2.insert(1, "Year", AAGG7_2['Date'].dt.year)
5 AAGG7_2
```

Out[314]:

	Date	Year	Symbol	Adj Close
0	2023-01-03	2023	AAPL	124.538658
1	2023-01-03	2023	GOOGL	89.120003
2	2023-01-04	2023	GOOGL	88.080002
3	2023-01-04	2023	AAPL	125.823189
4	2023-01-05	2023	AAPL	124.488876
...
327	2023-08-28	2023	AAPL	180.190002
328	2023-08-29	2023	GOOGL	134.570007
329	2023-08-29	2023	AAPL	184.119995
330	2023-08-30	2023	AAPL	187.649994
331	2023-08-30	2023	GOOGL	135.880005

332 rows × 4 columns

In [50]:

```
1
```

Out[50]:

	Date	Year	Symbol	Adj Close
0	2023-01-03	2023	AAPL	124.538666
1	2023-01-03	2023	GOOGL	89.120003
2	2023-01-04	2023	GOOGL	88.080002
3	2023-01-04	2023	AAPL	125.823189
4	2023-01-05	2023	AAPL	124.488876
...
327	2023-08-28	2023	AAPL	180.190002
328	2023-08-29	2023	GOOGL	134.570007
329	2023-08-29	2023	AAPL	184.119995
330	2023-08-30	2023	AAPL	187.649994
331	2023-08-30	2023	GOOGL	135.880005

332 rows × 4 columns

In [315]:

```
1 AAGG7_2.insert(2, "Month", AAGG7_2['Date'].dt.month)
2 AAGG7_2
```

Out[315]:

	Date	Year	Month	Symbol	Adj Close
0	2023-01-03	2023	1	AAPL	124.538658
1	2023-01-03	2023	1	GOOGL	89.120003
2	2023-01-04	2023	1	GOOGL	88.080002
3	2023-01-04	2023	1	AAPL	125.823189
4	2023-01-05	2023	1	AAPL	124.488876
...
327	2023-08-28	2023	8	AAPL	180.190002
328	2023-08-29	2023	8	GOOGL	134.570007
329	2023-08-29	2023	8	AAPL	184.119995
330	2023-08-30	2023	8	AAPL	187.649994
331	2023-08-30	2023	8	GOOGL	135.880005

332 rows × 5 columns

In [52]:

```
1
```

Out[52]:

	Date	Year	Month	Symbol	Adj Close
0	2023-01-03	2023	1	AAPL	124.538666
1	2023-01-03	2023	1	GOOGL	89.120003
2	2023-01-04	2023	1	GOOGL	88.080002
3	2023-01-04	2023	1	AAPL	125.823189
4	2023-01-05	2023	1	AAPL	124.488876
...
327	2023-08-28	2023	8	AAPL	180.190002
328	2023-08-29	2023	8	GOOGL	134.570007
329	2023-08-29	2023	8	AAPL	184.119995
330	2023-08-30	2023	8	AAPL	187.649994
331	2023-08-30	2023	8	GOOGL	135.880005

332 rows × 5 columns

Let's group by the Symbol column

In [316]:

```
1 AAGG7_2.groupby('Symbol')
```

Out[316]:

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000023F8E8BAA10>

In [53]:

```
1
```

Out[53]:

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fea789f4070>

Here, please save the result!

In [317]:

```
1 AAGG7_2_grouped = AAGG7_2.groupby('Symbol')
2 AAGG7_2_grouped
```

Out[317]:

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000023F96BAE810>

The groupby object has a property groups, which shows how all rows will in mapped into the groups.

The type of this object is a python dict

In [319]:

```
1 type(AAGG7_2_grouped.groups)
```

Out[319]:

pandas.io.formats.printing.PrettyDict

In [55]:

```
1
```

Out[55]:

pandas.io.formats.printing.PrettyDict

Let's show the mappings of rows to groups

In [321]:

```
1 # mappings of a dict object is just to call the dict itself
2 AAGG7_2_grouped.groups
```

Out[321]:

```
{'AAPL': [0, 3, 4, 6, 8, 11, 13, 15, 17, 19, 21, 23, 24, 26, 28, 31, 32, 35, 37, 3
9, 40, 42, 45, 46, 48, 50, 52, 54, 56, 59, 61, 62, 65, 66, 69, 71, 72, 75, 76, 79,
80, 82, 84, 86, 88, 91, 93, 95, 97, 98, 101, 102, 104, 107, 108, 111, 112, 114, 11
6, 119, 121, 123, 125, 127, 129, 130, 132, 135, 137, 139, 141, 143, 145, 147, 148,
151, 153, 154, 157, 158, 161, 163, 164, 167, 169, 171, 172, 174, 176, 178, 180, 18
3, 185, 186, 188, 190, 193, 195, 197, 199, ...], 'GOOGL': [1, 2, 5, 7, 9, 10, 12,
14, 16, 18, 20, 22, 25, 27, 29, 30, 33, 34, 36, 38, 41, 43, 44, 47, 49, 51, 53, 5
5, 57, 58, 60, 63, 64, 67, 68, 70, 73, 74, 77, 78, 81, 83, 85, 87, 89, 90, 92, 94,
96, 99, 100, 103, 105, 106, 109, 110, 113, 115, 117, 118, 120, 122, 124, 126, 128,
131, 133, 134, 136, 138, 140, 142, 144, 146, 149, 150, 152, 155, 156, 159, 160, 16
2, 165, 166, 168, 170, 173, 175, 177, 179, 181, 182, 184, 187, 189, 191, 192, 194,
196, 198, ...]}
```

In [56]:

```
1
```

Out[56]:

```
{'AAPL': [0, 3, 4, 6, 8, 11, 13, 15, 17, 19, 21, 23, 24, 26, 28, 31, 32, 35, 37, 3
9, 40, 42, 45, 46, 48, 50, 52, 54, 56, 59, 61, 62, 65, 66, 69, 71, 72, 75, 76, 79,
80, 82, 84, 86, 88, 91, 93, 95, 97, 98, 101, 102, 104, 107, 108, 111, 112, 114, 11
6, 119, 121, 123, 125, 127, 129, 130, 132, 135, 137, 139, 141, 143, 145, 147, 148,
151, 153, 154, 157, 158, 161, 163, 164, 167, 169, 171, 172, 174, 176, 178, 180, 18
3, 185, 186, 188, 190, 193, 195, 197, 199, ...], 'GOOGL': [1, 2, 5, 7, 9, 10, 12,
14, 16, 18, 20, 22, 25, 27, 29, 30, 33, 34, 36, 38, 41, 43, 44, 47, 49, 51, 53, 5
5, 57, 58, 60, 63, 64, 67, 68, 70, 73, 74, 77, 78, 81, 83, 85, 87, 89, 90, 92, 94,
96, 99, 100, 103, 105, 106, 109, 110, 113, 115, 117, 118, 120, 122, 124, 126, 128,
131, 133, 134, 136, 138, 140, 142, 144, 146, 149, 150, 152, 155, 156, 159, 160, 16
2, 165, 166, 168, 170, 173, 175, 177, 179, 181, 182, 184, 187, 189, 191, 192, 194,
196, 198, ...]}
```

Yes, this reports the number of groups that resulted from the grouping

In [330]:

```
1 AAGG7_2_grouped.count()
```

Out[330]:

	Date	Year	Month	Adj Close
Symbol				
AAPL	166	166	166	166
GOOGL	166	166	166	166

In [331]:

```
1 AAGG7_2_grouped.ngroups
```

Out[331]:

2

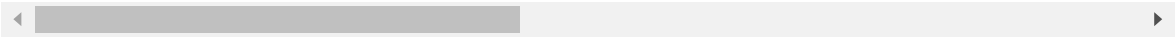
In [338]:

```
1 AAGG7_2_grouped.describe()
```

Out[338]:

Date										Yea
	count	mean	min	25%	50%	75%	max	std	cou	
Symbol										
AAPL	166	2023-05-02 16:46:15.903614464	2023-01-03 00:00:00	2023-03-03 18:00:00	2023-05-02 12:00:00	2023-07-02 06:00:00	2023-08-30 00:00:00	NaN	166	
GOOGL	166	2023-05-02 16:46:15.903614464	2023-01-03 00:00:00	2023-03-03 18:00:00	2023-05-02 12:00:00	2023-07-02 06:00:00	2023-08-30 00:00:00	NaN	166	

2 rows × 32 columns



In [340]:

```
1 (len(AAGG7_2_grouped), AAGG7_2_grouped.ngroups)
```

Out[340]:

(2, 2)

In [57]:

```
1
```

Out[57]:

(2, 2)

Question 8-1. Please create a function that prints the contents of a group.

When you execute your function you should see the following results.

In [353]:

```
1 def print_grouped_df(grouped_df):
2     # grouped object是个dict, 所以循环要包括keys和values
3     for group_name, group_data in grouped_df:
4         print(group_name)
5         print(group_data)
6
```

In [354]:

```
1 print_grouped_df(AAGG7_2_grouped)
```

AAPL

	Date	Year	Month	Symbol	Adj Close
0	2023-01-03	2023	1	AAPL	124.538658
3	2023-01-04	2023	1	AAPL	125.823189
4	2023-01-05	2023	1	AAPL	124.488876
6	2023-01-06	2023	1	AAPL	129.069321
8	2023-01-09	2023	1	AAPL	129.597061
..
323	2023-08-24	2023	8	AAPL	176.380005
325	2023-08-25	2023	8	AAPL	178.610001
327	2023-08-28	2023	8	AAPL	180.190002
329	2023-08-29	2023	8	AAPL	184.119995
330	2023-08-30	2023	8	AAPL	187.649994

[166 rows x 5 columns]

GOOGL

	Date	Year	Month	Symbol	Adj Close
1	2023-01-03	2023	1	GOOGL	89.120003
2	2023-01-04	2023	1	GOOGL	88.080002
5	2023-01-05	2023	1	GOOGL	86.199997
7	2023-01-06	2023	1	GOOGL	87.339996
9	2023-01-09	2023	1	GOOGL	88.019997
..
322	2023-08-24	2023	8	GOOGL	129.779999
324	2023-08-25	2023	8	GOOGL	129.880005
326	2023-08-28	2023	8	GOOGL	131.009995
328	2023-08-29	2023	8	GOOGL	134.570007
331	2023-08-30	2023	8	GOOGL	135.880005

[166 rows x 5 columns]

In [357]:

```
1 def print_grouped_df2(grouped_df2):
2     # grouped object是个dict, 所以循环要包括keys和values
3     for group_name, group_data in grouped_df2:
4         print(group_name)
5         print(group_data.to_string(index=False)) #为了避免打印出最后的 [166 rows x 5 columns]
6         print()
```

In [358]:

```
1 print_grouped_df2(AAGG7_2_grouped)
```


AAPL					
	Date	Year	Month	Symbol	Adj Close
	2023-01-03	2023	1	AAPL	124.538658
	2023-01-04	2023	1	AAPL	125.823189
	2023-01-05	2023	1	AAPL	124.488876
	2023-01-06	2023	1	AAPL	129.069321
	2023-01-09	2023	1	AAPL	129.597061
	2023-01-10	2023	1	AAPL	130.174606
	2023-01-11	2023	1	AAPL	132.922897
	2023-01-12	2023	1	AAPL	132.843231
	2023-01-13	2023	1	AAPL	134.187500
	2023-01-17	2023	1	AAPL	135.362473
	2023-01-18	2023	1	AAPL	134.635590
	2023-01-19	2023	1	AAPL	134.695328
	2023-01-20	2023	1	AAPL	137.284286
	2023-01-23	2023	1	AAPL	140.510513
	2023-01-24	2023	1	AAPL	141.924500
	2023-01-25	2023	1	AAPL	141.257324
	2023-01-26	2023	1	AAPL	143.348419
	2023-01-27	2023	1	AAPL	145.310043
	2023-01-30	2023	1	AAPL	142.392487
	2023-01-31	2023	1	AAPL	143.676987
	2023-02-01	2023	2	AAPL	144.812164
	2023-02-02	2023	2	AAPL	150.179276
	2023-02-03	2023	2	AAPL	153.843628
	2023-02-06	2023	2	AAPL	151.085403
	2023-02-07	2023	2	AAPL	153.992996
	2023-02-08	2023	2	AAPL	151.274597
	2023-02-09	2023	2	AAPL	150.229065
	2023-02-10	2023	2	AAPL	150.598038
	2023-02-13	2023	2	AAPL	153.430313
	2023-02-14	2023	2	AAPL	152.782074
	2023-02-15	2023	2	AAPL	154.906265
	2023-02-16	2023	2	AAPL	153.290695
	2023-02-17	2023	2	AAPL	152.133850
	2023-02-21	2023	2	AAPL	148.074951
	2023-02-22	2023	2	AAPL	148.503784
	2023-02-23	2023	2	AAPL	148.992432
	2023-02-24	2023	2	AAPL	146.309784
	2023-02-27	2023	2	AAPL	147.516479
	2023-02-28	2023	2	AAPL	147.007874
	2023-03-01	2023	3	AAPL	144.913589
	2023-03-02	2023	3	AAPL	145.511963
	2023-03-03	2023	3	AAPL	150.617996
	2023-03-06	2023	3	AAPL	153.410355
	2023-03-07	2023	3	AAPL	151.186447
	2023-03-08	2023	3	AAPL	152.452972
	2023-03-09	2023	3	AAPL	150.179184
	2023-03-10	2023	3	AAPL	148.094894
	2023-03-13	2023	3	AAPL	150.059525
	2023-03-14	2023	3	AAPL	152.173737
	2023-03-15	2023	3	AAPL	152.572662
	2023-03-16	2023	3	AAPL	155.424850
	2023-03-17	2023	3	AAPL	154.577164
	2023-03-20	2023	3	AAPL	156.970612
	2023-03-21	2023	3	AAPL	158.845474
	2023-03-22	2023	3	AAPL	157.399445
	2023-03-23	2023	3	AAPL	158.496429
	2023-03-24	2023	3	AAPL	159.812851
	2023-03-27	2023	3	AAPL	157.848221
	2023-03-28	2023	3	AAPL	157.219925

2023-03-29	2023	3	AAPL	160.331436
2023-03-30	2023	3	AAPL	161.917099
2023-03-31	2023	3	AAPL	164.450150
2023-04-03	2023	4	AAPL	165.716690
2023-04-04	2023	4	AAPL	165.178177
2023-04-05	2023	4	AAPL	163.313263
2023-04-06	2023	4	AAPL	164.210815
2023-04-10	2023	4	AAPL	161.587982
2023-04-11	2023	4	AAPL	160.361343
2023-04-12	2023	4	AAPL	159.663254
2023-04-13	2023	4	AAPL	165.108353
2023-04-14	2023	4	AAPL	164.759323
2023-04-17	2023	4	AAPL	164.779251
2023-04-18	2023	4	AAPL	166.015884
2023-04-19	2023	4	AAPL	167.172729
2023-04-20	2023	4	AAPL	166.195374
2023-04-21	2023	4	AAPL	164.569839
2023-04-24	2023	4	AAPL	164.878983
2023-04-25	2023	4	AAPL	163.323242
2023-04-26	2023	4	AAPL	163.313263
2023-04-27	2023	4	AAPL	167.950592
2023-04-28	2023	4	AAPL	169.217117
2023-05-01	2023	5	AAPL	169.127365
2023-05-02	2023	5	AAPL	168.080215
2023-05-03	2023	5	AAPL	166.993195
2023-05-04	2023	5	AAPL	165.337723
2023-05-05	2023	5	AAPL	173.096512
2023-05-08	2023	5	AAPL	173.026688
2023-05-09	2023	5	AAPL	171.301422
2023-05-10	2023	5	AAPL	173.086533
2023-05-11	2023	5	AAPL	173.276016
2023-05-12	2023	5	AAPL	172.337280
2023-05-15	2023	5	AAPL	171.837967
2023-05-16	2023	5	AAPL	171.837967
2023-05-17	2023	5	AAPL	172.457123
2023-05-18	2023	5	AAPL	174.813934
2023-05-19	2023	5	AAPL	174.923798
2023-05-22	2023	5	AAPL	173.965073
2023-05-23	2023	5	AAPL	171.328644
2023-05-24	2023	5	AAPL	171.608261
2023-05-25	2023	5	AAPL	172.756714
2023-05-26	2023	5	AAPL	175.193420
2023-05-30	2023	5	AAPL	177.060898
2023-05-31	2023	5	AAPL	177.010971
2023-06-01	2023	6	AAPL	179.847137
2023-06-02	2023	6	AAPL	180.705978
2023-06-05	2023	6	AAPL	179.337830
2023-06-06	2023	6	AAPL	178.968338
2023-06-07	2023	6	AAPL	177.580200
2023-06-08	2023	6	AAPL	180.326492
2023-06-09	2023	6	AAPL	180.715973
2023-06-12	2023	6	AAPL	183.542145
2023-06-13	2023	6	AAPL	183.062790
2023-06-14	2023	6	AAPL	183.701935
2023-06-15	2023	6	AAPL	185.759155
2023-06-16	2023	6	AAPL	184.670624
2023-06-20	2023	6	AAPL	184.760498
2023-06-21	2023	6	AAPL	183.711929
2023-06-22	2023	6	AAPL	186.747818
2023-06-23	2023	6	AAPL	186.428238
2023-06-26	2023	6	AAPL	185.020157

2023-06-27	2023	6	AAPL	187.806381
2023-06-28	2023	6	AAPL	188.994781
2023-06-29	2023	6	AAPL	189.334320
2023-06-30	2023	6	AAPL	193.708420
2023-07-03	2023	7	AAPL	192.200470
2023-07-05	2023	7	AAPL	191.071976
2023-07-06	2023	7	AAPL	191.551331
2023-07-07	2023	7	AAPL	190.422852
2023-07-10	2023	7	AAPL	188.355652
2023-07-11	2023	7	AAPL	187.826370
2023-07-12	2023	7	AAPL	189.514084
2023-07-13	2023	7	AAPL	190.283035
2023-07-14	2023	7	AAPL	190.432846
2023-07-17	2023	7	AAPL	193.728394
2023-07-18	2023	7	AAPL	193.468735
2023-07-19	2023	7	AAPL	194.836899
2023-07-20	2023	7	AAPL	192.869553
2023-07-21	2023	7	AAPL	191.681168
2023-07-24	2023	7	AAPL	192.490067
2023-07-25	2023	7	AAPL	193.358887
2023-07-26	2023	7	AAPL	194.237701
2023-07-27	2023	7	AAPL	192.959427
2023-07-28	2023	7	AAPL	195.565918
2023-07-31	2023	7	AAPL	196.185074
2023-08-01	2023	8	AAPL	195.346207
2023-08-02	2023	8	AAPL	192.320297
2023-08-03	2023	8	AAPL	190.912201
2023-08-04	2023	8	AAPL	181.744583
2023-08-07	2023	8	AAPL	178.608810
2023-08-08	2023	8	AAPL	179.557526
2023-08-09	2023	8	AAPL	177.949707
2023-08-10	2023	8	AAPL	177.729996
2023-08-11	2023	8	AAPL	177.789993
2023-08-14	2023	8	AAPL	179.460007
2023-08-15	2023	8	AAPL	177.449997
2023-08-16	2023	8	AAPL	176.570007
2023-08-17	2023	8	AAPL	174.000000
2023-08-18	2023	8	AAPL	174.490005
2023-08-21	2023	8	AAPL	175.839996
2023-08-22	2023	8	AAPL	177.229996
2023-08-23	2023	8	AAPL	181.119995
2023-08-24	2023	8	AAPL	176.380005
2023-08-25	2023	8	AAPL	178.610001
2023-08-28	2023	8	AAPL	180.190002
2023-08-29	2023	8	AAPL	184.119995
2023-08-30	2023	8	AAPL	187.649994

GOOGL

Date	Year	Month	Symbol	Adj Close
2023-01-03	2023	1	GOOGL	89.120003
2023-01-04	2023	1	GOOGL	88.080002
2023-01-05	2023	1	GOOGL	86.199997
2023-01-06	2023	1	GOOGL	87.339996
2023-01-09	2023	1	GOOGL	88.019997
2023-01-10	2023	1	GOOGL	88.419998
2023-01-11	2023	1	GOOGL	91.519997
2023-01-12	2023	1	GOOGL	91.129997
2023-01-13	2023	1	GOOGL	92.120003
2023-01-17	2023	1	GOOGL	91.290001
2023-01-18	2023	1	GOOGL	91.120003
2023-01-19	2023	1	GOOGL	93.050003

2023-01-20	2023	1	GOOGL	98.019997
2023-01-23	2023	1	GOOGL	99.790001
2023-01-24	2023	1	GOOGL	97.699997
2023-01-25	2023	1	GOOGL	95.220001
2023-01-26	2023	1	GOOGL	97.519997
2023-01-27	2023	1	GOOGL	99.370003
2023-01-30	2023	1	GOOGL	96.940002
2023-01-31	2023	1	GOOGL	98.839996
2023-02-01	2023	2	GOOGL	100.430000
2023-02-02	2023	2	GOOGL	107.739998
2023-02-03	2023	2	GOOGL	104.779999
2023-02-06	2023	2	GOOGL	102.900002
2023-02-07	2023	2	GOOGL	107.639999
2023-02-08	2023	2	GOOGL	99.370003
2023-02-09	2023	2	GOOGL	95.010002
2023-02-10	2023	2	GOOGL	94.570000
2023-02-13	2023	2	GOOGL	94.610001
2023-02-14	2023	2	GOOGL	94.680000
2023-02-15	2023	2	GOOGL	96.940002
2023-02-16	2023	2	GOOGL	95.510002
2023-02-17	2023	2	GOOGL	94.349998
2023-02-21	2023	2	GOOGL	91.790001
2023-02-22	2023	2	GOOGL	91.650002
2023-02-23	2023	2	GOOGL	90.889999
2023-02-24	2023	2	GOOGL	89.129997
2023-02-27	2023	2	GOOGL	89.870003
2023-02-28	2023	2	GOOGL	90.059998
2023-03-01	2023	3	GOOGL	90.360001
2023-03-02	2023	3	GOOGL	92.000000
2023-03-03	2023	3	GOOGL	93.650002
2023-03-06	2023	3	GOOGL	95.129997
2023-03-07	2023	3	GOOGL	93.860001
2023-03-08	2023	3	GOOGL	94.250000
2023-03-09	2023	3	GOOGL	92.320000
2023-03-10	2023	3	GOOGL	90.629997
2023-03-13	2023	3	GOOGL	91.110001
2023-03-14	2023	3	GOOGL	93.970001
2023-03-15	2023	3	GOOGL	96.110001
2023-03-16	2023	3	GOOGL	100.320000
2023-03-17	2023	3	GOOGL	101.620003
2023-03-20	2023	3	GOOGL	101.220001
2023-03-21	2023	3	GOOGL	104.919998
2023-03-22	2023	3	GOOGL	103.370003
2023-03-23	2023	3	GOOGL	105.599998
2023-03-24	2023	3	GOOGL	105.440002
2023-03-27	2023	3	GOOGL	102.459999
2023-03-28	2023	3	GOOGL	101.029999
2023-03-29	2023	3	GOOGL	101.389999
2023-03-30	2023	3	GOOGL	100.889999
2023-03-31	2023	3	GOOGL	103.730003
2023-04-03	2023	4	GOOGL	104.360001
2023-04-04	2023	4	GOOGL	104.720001
2023-04-05	2023	4	GOOGL	104.470001
2023-04-06	2023	4	GOOGL	108.419998
2023-04-10	2023	4	GOOGL	106.440002
2023-04-11	2023	4	GOOGL	105.349998
2023-04-12	2023	4	GOOGL	104.639999
2023-04-13	2023	4	GOOGL	107.430000
2023-04-14	2023	4	GOOGL	108.870003
2023-04-17	2023	4	GOOGL	105.970001
2023-04-18	2023	4	GOOGL	104.500000

2023-04-19	2023	4	GOOGL	104.180000
2023-04-20	2023	4	GOOGL	105.290001
2023-04-21	2023	4	GOOGL	105.410004
2023-04-24	2023	4	GOOGL	105.970001
2023-04-25	2023	4	GOOGL	103.849998
2023-04-26	2023	4	GOOGL	103.709999
2023-04-27	2023	4	GOOGL	107.589996
2023-04-28	2023	4	GOOGL	107.339996
2023-05-01	2023	5	GOOGL	107.199997
2023-05-02	2023	5	GOOGL	105.320000
2023-05-03	2023	5	GOOGL	105.410004
2023-05-04	2023	5	GOOGL	104.690002
2023-05-05	2023	5	GOOGL	105.570000
2023-05-08	2023	5	GOOGL	107.769997
2023-05-09	2023	5	GOOGL	107.349998
2023-05-10	2023	5	GOOGL	111.750000
2023-05-11	2023	5	GOOGL	116.570000
2023-05-12	2023	5	GOOGL	117.510002
2023-05-15	2023	5	GOOGL	116.510002
2023-05-16	2023	5	GOOGL	119.510002
2023-05-17	2023	5	GOOGL	120.839996
2023-05-18	2023	5	GOOGL	122.830002
2023-05-19	2023	5	GOOGL	122.760002
2023-05-22	2023	5	GOOGL	125.050003
2023-05-23	2023	5	GOOGL	122.559998
2023-05-24	2023	5	GOOGL	120.900002
2023-05-25	2023	5	GOOGL	123.480003
2023-05-26	2023	5	GOOGL	124.610001
2023-05-30	2023	5	GOOGL	123.669998
2023-05-31	2023	5	GOOGL	122.870003
2023-06-01	2023	6	GOOGL	123.720001
2023-06-02	2023	6	GOOGL	124.669998
2023-06-05	2023	6	GOOGL	126.010002
2023-06-06	2023	6	GOOGL	127.309998
2023-06-07	2023	6	GOOGL	122.500000
2023-06-08	2023	6	GOOGL	122.139999
2023-06-09	2023	6	GOOGL	122.230003
2023-06-12	2023	6	GOOGL	123.639999
2023-06-13	2023	6	GOOGL	123.830002
2023-06-14	2023	6	GOOGL	123.669998
2023-06-15	2023	6	GOOGL	125.089996
2023-06-16	2023	6	GOOGL	123.529999
2023-06-20	2023	6	GOOGL	123.099998
2023-06-21	2023	6	GOOGL	120.550003
2023-06-22	2023	6	GOOGL	123.150002
2023-06-23	2023	6	GOOGL	122.339996
2023-06-26	2023	6	GOOGL	118.339996
2023-06-27	2023	6	GOOGL	118.330002
2023-06-28	2023	6	GOOGL	120.180000
2023-06-29	2023	6	GOOGL	119.099998
2023-06-30	2023	6	GOOGL	119.699997
2023-07-03	2023	7	GOOGL	119.900002
2023-07-05	2023	7	GOOGL	121.750000
2023-07-06	2023	7	GOOGL	120.110001
2023-07-07	2023	7	GOOGL	119.480003
2023-07-10	2023	7	GOOGL	116.449997
2023-07-11	2023	7	GOOGL	117.139999
2023-07-12	2023	7	GOOGL	118.930000
2023-07-13	2023	7	GOOGL	124.540001
2023-07-14	2023	7	GOOGL	125.419998
2023-07-17	2023	7	GOOGL	124.650002

```
2023-07-18 2023      7  GOOGL 123.760002
2023-07-19 2023      7  GOOGL 122.029999
2023-07-20 2023      7  GOOGL 119.199997
2023-07-21 2023      7  GOOGL 120.019997
2023-07-24 2023      7  GOOGL 121.529999
2023-07-25 2023      7  GOOGL 122.209999
2023-07-26 2023      7  GOOGL 129.270004
2023-07-27 2023      7  GOOGL 129.399994
2023-07-28 2023      7  GOOGL 132.580002
2023-07-31 2023      7  GOOGL 132.720001
2023-08-01 2023      8  GOOGL 131.550003
2023-08-02 2023      8  GOOGL 128.380005
2023-08-03 2023      8  GOOGL 128.449997
2023-08-04 2023      8  GOOGL 128.110001
2023-08-07 2023      8  GOOGL 131.529999
2023-08-08 2023      8  GOOGL 131.399994
2023-08-09 2023      8  GOOGL 129.660004
2023-08-10 2023      8  GOOGL 129.690002
2023-08-11 2023      8  GOOGL 129.559998
2023-08-14 2023      8  GOOGL 131.330002
2023-08-15 2023      8  GOOGL 129.779999
2023-08-16 2023      8  GOOGL 128.699997
2023-08-17 2023      8  GOOGL 129.919998
2023-08-18 2023      8  GOOGL 127.459999
2023-08-21 2023      8  GOOGL 128.369995
2023-08-22 2023      8  GOOGL 129.080002
2023-08-23 2023      8  GOOGL 132.369995
2023-08-24 2023      8  GOOGL 129.779999
2023-08-25 2023      8  GOOGL 129.880005
2023-08-28 2023      8  GOOGL 131.009995
2023-08-29 2023      8  GOOGL 134.570007
2023-08-30 2023      8  GOOGL 135.880005
```

In []:

1

In [62]:

1

```
AAPL
      Date Year Month Symbol  Adj Close
0 2023-01-03 2023     1  AAPL  124.538666
3 2023-01-04 2023     1  AAPL  125.823189
4 2023-01-05 2023     1  AAPL  124.488876
6 2023-01-06 2023     1  AAPL  129.069321
8 2023-01-09 2023     1  AAPL  129.597061
GOOGL
      Date Year Month Symbol  Adj Close
1 2023-01-03 2023     1  GOOGL   89.120003
2 2023-01-04 2023     1  GOOGL   88.080002
5 2023-01-05 2023     1  GOOGL   86.199997
7 2023-01-06 2023     1  GOOGL   87.339996
9 2023-01-09 2023     1  GOOGL   88.019997
```

.size will tell us the count of items in each group

In [341]:

```
1 AAGG7_2_grouped.size()
```

Out[341]:

Symbol
AAPL 166
GOOGL 166
dtype: int64

In [63]:

```
1
```

Out[63]:

Symbol
AAPL 166
GOOGL 166
dtype: int64

Question 8-2. As you know, a specific group can be retrieved using .get_group() which returns a DataFrame representing the specified group. Please execute so that you can have the output below.

In [343]:

```
1 apple8_2 = AAGG7_2_grouped.get_group('AAPL')  
2 apple8_2
```

Out[343]:

	Date	Year	Month	Symbol	Adj Close
0	2023-01-03	2023	1	AAPL	124.538658
3	2023-01-04	2023	1	AAPL	125.823189
4	2023-01-05	2023	1	AAPL	124.488876
6	2023-01-06	2023	1	AAPL	129.069321
8	2023-01-09	2023	1	AAPL	129.597061
...
323	2023-08-24	2023	8	AAPL	176.380005
325	2023-08-25	2023	8	AAPL	178.610001
327	2023-08-28	2023	8	AAPL	180.190002
329	2023-08-29	2023	8	AAPL	184.119995
330	2023-08-30	2023	8	AAPL	187.649994

166 rows × 5 columns

In [64]:

```
1
```

Out[64]:

	Date	Year	Month	Symbol	Adj Close
0	2023-01-03	2023	1	AAPL	124.538666
3	2023-01-04	2023	1	AAPL	125.823189
4	2023-01-05	2023	1	AAPL	124.488876
6	2023-01-06	2023	1	AAPL	129.069321
8	2023-01-09	2023	1	AAPL	129.597061
...
323	2023-08-24	2023	8	AAPL	176.380005
325	2023-08-25	2023	8	AAPL	178.610001
327	2023-08-28	2023	8	AAPL	180.190002
329	2023-08-29	2023	8	AAPL	184.119995
330	2023-08-30	2023	8	AAPL	187.649994

166 rows × 5 columns

Question 8-3. Please group by three different fields (Symbol , Year , Month) and print the result



In [359]:

```
1 AAGG7_2
```

Out[359]:

	Date	Year	Month	Symbol	Adj Close
0	2023-01-03	2023	1	AAPL	124.538658
1	2023-01-03	2023	1	GOOGL	89.120003
2	2023-01-04	2023	1	GOOGL	88.080002
3	2023-01-04	2023	1	AAPL	125.823189
4	2023-01-05	2023	1	AAPL	124.488876
...
327	2023-08-28	2023	8	AAPL	180.190002
328	2023-08-29	2023	8	GOOGL	134.570007
329	2023-08-29	2023	8	AAPL	184.119995
330	2023-08-30	2023	8	AAPL	187.649994
331	2023-08-30	2023	8	GOOGL	135.880005

332 rows × 5 columns

In [360]:

```
1 AAGG8_3_grouped = AAGG7_2.groupby(['Symbol', 'Year', 'Month'])
```

In [361]:

```
1 print_grouped_df(AAGG8_3_grouped)
```

(' AAPL', 2023, 1)

	Date	Year	Month	Symbol	Adj Close
0	2023-01-03	2023	1	AAPL	124.538658
3	2023-01-04	2023	1	AAPL	125.823189
4	2023-01-05	2023	1	AAPL	124.488876
6	2023-01-06	2023	1	AAPL	129.069321
8	2023-01-09	2023	1	AAPL	129.597061
11	2023-01-10	2023	1	AAPL	130.174606
13	2023-01-11	2023	1	AAPL	132.922897
15	2023-01-12	2023	1	AAPL	132.843231
17	2023-01-13	2023	1	AAPL	134.187500
19	2023-01-17	2023	1	AAPL	135.362473
21	2023-01-18	2023	1	AAPL	134.635590
23	2023-01-19	2023	1	AAPL	134.695328
24	2023-01-20	2023	1	AAPL	137.284286
26	2023-01-23	2023	1	AAPL	140.510513
28	2023-01-24	2023	1	AAPL	141.924500
31	2023-01-25	2023	1	AAPL	141.257324
32	2023-01-26	2023	1	AAPL	143.348419
35	2023-01-27	2023	1	AAPL	145.310043
37	2023-01-30	2023	1	AAPL	142.392487
39	2023-01-31	2023	1	AAPL	143.676987

(' AAPL', 2023, 2)

	Date	Year	Month	Symbol	Adj Close
40	2023-02-01	2023	2	AAPL	144.812164
42	2023-02-02	2023	2	AAPL	150.179276
45	2023-02-03	2023	2	AAPL	153.843628
46	2023-02-06	2023	2	AAPL	151.085403
48	2023-02-07	2023	2	AAPL	153.992996
50	2023-02-08	2023	2	AAPL	151.274597
52	2023-02-09	2023	2	AAPL	150.229065
54	2023-02-10	2023	2	AAPL	150.598038
56	2023-02-13	2023	2	AAPL	153.430313
59	2023-02-14	2023	2	AAPL	152.782074
61	2023-02-15	2023	2	AAPL	154.906265
62	2023-02-16	2023	2	AAPL	153.290695
65	2023-02-17	2023	2	AAPL	152.133850
66	2023-02-21	2023	2	AAPL	148.074951
69	2023-02-22	2023	2	AAPL	148.503784
71	2023-02-23	2023	2	AAPL	148.992432
72	2023-02-24	2023	2	AAPL	146.309784
75	2023-02-27	2023	2	AAPL	147.516479
76	2023-02-28	2023	2	AAPL	147.007874

(' AAPL', 2023, 3)

	Date	Year	Month	Symbol	Adj Close
79	2023-03-01	2023	3	AAPL	144.913589
80	2023-03-02	2023	3	AAPL	145.511963
82	2023-03-03	2023	3	AAPL	150.617996
84	2023-03-06	2023	3	AAPL	153.410355
86	2023-03-07	2023	3	AAPL	151.186447
88	2023-03-08	2023	3	AAPL	152.452972
91	2023-03-09	2023	3	AAPL	150.179184
93	2023-03-10	2023	3	AAPL	148.094894
95	2023-03-13	2023	3	AAPL	150.059525
97	2023-03-14	2023	3	AAPL	152.173737
98	2023-03-15	2023	3	AAPL	152.572662
101	2023-03-16	2023	3	AAPL	155.424850
102	2023-03-17	2023	3	AAPL	154.577164
104	2023-03-20	2023	3	AAPL	156.970612
107	2023-03-21	2023	3	AAPL	158.845474
108	2023-03-22	2023	3	AAPL	157.399445

```

111 2023-03-23 2023      3  AAPL  158.496429
112 2023-03-24 2023      3  AAPL  159.812851
114 2023-03-27 2023      3  AAPL  157.848221
116 2023-03-28 2023      3  AAPL  157.219925
119 2023-03-29 2023      3  AAPL  160.331436
121 2023-03-30 2023      3  AAPL  161.917099
123 2023-03-31 2023      3  AAPL  164.450150
('AAPL', 2023, 4)

```

	Date	Year	Month	Symbol	Adj Close
125	2023-04-03	2023	4	AAPL	165.716690
127	2023-04-04	2023	4	AAPL	165.178177
129	2023-04-05	2023	4	AAPL	163.313263
130	2023-04-06	2023	4	AAPL	164.210815
132	2023-04-10	2023	4	AAPL	161.587982
135	2023-04-11	2023	4	AAPL	160.361343
137	2023-04-12	2023	4	AAPL	159.663254
139	2023-04-13	2023	4	AAPL	165.108353
141	2023-04-14	2023	4	AAPL	164.759323
143	2023-04-17	2023	4	AAPL	164.779251
145	2023-04-18	2023	4	AAPL	166.015884
147	2023-04-19	2023	4	AAPL	167.172729
148	2023-04-20	2023	4	AAPL	166.195374
151	2023-04-21	2023	4	AAPL	164.569839
153	2023-04-24	2023	4	AAPL	164.878983
154	2023-04-25	2023	4	AAPL	163.323242
157	2023-04-26	2023	4	AAPL	163.313263
158	2023-04-27	2023	4	AAPL	167.950592
161	2023-04-28	2023	4	AAPL	169.217117

```

('AAPL', 2023, 5)

```

	Date	Year	Month	Symbol	Adj Close
163	2023-05-01	2023	5	AAPL	169.127365
164	2023-05-02	2023	5	AAPL	168.080215
167	2023-05-03	2023	5	AAPL	166.993195
169	2023-05-04	2023	5	AAPL	165.337723
171	2023-05-05	2023	5	AAPL	173.096512
172	2023-05-08	2023	5	AAPL	173.026688
174	2023-05-09	2023	5	AAPL	171.301422
176	2023-05-10	2023	5	AAPL	173.086533
178	2023-05-11	2023	5	AAPL	173.276016
180	2023-05-12	2023	5	AAPL	172.337280
183	2023-05-15	2023	5	AAPL	171.837967
185	2023-05-16	2023	5	AAPL	171.837967
186	2023-05-17	2023	5	AAPL	172.457123
188	2023-05-18	2023	5	AAPL	174.813934
190	2023-05-19	2023	5	AAPL	174.923798
193	2023-05-22	2023	5	AAPL	173.965073
195	2023-05-23	2023	5	AAPL	171.328644
197	2023-05-24	2023	5	AAPL	171.608261
199	2023-05-25	2023	5	AAPL	172.756714
200	2023-05-26	2023	5	AAPL	175.193420
202	2023-05-30	2023	5	AAPL	177.060898
205	2023-05-31	2023	5	AAPL	177.010971

```

('AAPL', 2023, 6)

```

	Date	Year	Month	Symbol	Adj Close
207	2023-06-01	2023	6	AAPL	179.847137
208	2023-06-02	2023	6	AAPL	180.705978
210	2023-06-05	2023	6	AAPL	179.337830
212	2023-06-06	2023	6	AAPL	178.968338
215	2023-06-07	2023	6	AAPL	177.580200
217	2023-06-08	2023	6	AAPL	180.326492
218	2023-06-09	2023	6	AAPL	180.715973

```

221 2023-06-12 2023      6 AAPL 183.542145
222 2023-06-13 2023      6 AAPL 183.062790
224 2023-06-14 2023      6 AAPL 183.701935
226 2023-06-15 2023      6 AAPL 185.759155
228 2023-06-16 2023      6 AAPL 184.670624
231 2023-06-20 2023      6 AAPL 184.760498
233 2023-06-21 2023      6 AAPL 183.711929
234 2023-06-22 2023      6 AAPL 186.747818
237 2023-06-23 2023      6 AAPL 186.428238
238 2023-06-26 2023      6 AAPL 185.020157
240 2023-06-27 2023      6 AAPL 187.806381
242 2023-06-28 2023      6 AAPL 188.994781
245 2023-06-29 2023      6 AAPL 189.334320
247 2023-06-30 2023      6 AAPL 193.708420

```

```
('AAPL', 2023, 7)
```

	Date	Year	Month	Symbol	Adj Close
249	2023-07-03	2023	7	AAPL	192.200470
250	2023-07-05	2023	7	AAPL	191.071976
252	2023-07-06	2023	7	AAPL	191.551331
255	2023-07-07	2023	7	AAPL	190.422852
256	2023-07-10	2023	7	AAPL	188.355652
258	2023-07-11	2023	7	AAPL	187.826370
260	2023-07-12	2023	7	AAPL	189.514084
263	2023-07-13	2023	7	AAPL	190.283035
264	2023-07-14	2023	7	AAPL	190.432846
267	2023-07-17	2023	7	AAPL	193.728394
269	2023-07-18	2023	7	AAPL	193.468735
271	2023-07-19	2023	7	AAPL	194.836899
272	2023-07-20	2023	7	AAPL	192.869553
274	2023-07-21	2023	7	AAPL	191.681168
276	2023-07-24	2023	7	AAPL	192.490067
279	2023-07-25	2023	7	AAPL	193.358887
280	2023-07-26	2023	7	AAPL	194.237701
283	2023-07-27	2023	7	AAPL	192.959427
284	2023-07-28	2023	7	AAPL	195.565918
287	2023-07-31	2023	7	AAPL	196.185074

```
('AAPL', 2023, 8)
```

	Date	Year	Month	Symbol	Adj Close
288	2023-08-01	2023	8	AAPL	195.346207
291	2023-08-02	2023	8	AAPL	192.320297
293	2023-08-03	2023	8	AAPL	190.912201
295	2023-08-04	2023	8	AAPL	181.744583
296	2023-08-07	2023	8	AAPL	178.608810
299	2023-08-08	2023	8	AAPL	179.557526
301	2023-08-09	2023	8	AAPL	177.949707
303	2023-08-10	2023	8	AAPL	177.729996
304	2023-08-11	2023	8	AAPL	177.789993
307	2023-08-14	2023	8	AAPL	179.460007
309	2023-08-15	2023	8	AAPL	177.449997
311	2023-08-16	2023	8	AAPL	176.570007
313	2023-08-17	2023	8	AAPL	174.000000
315	2023-08-18	2023	8	AAPL	174.490005
317	2023-08-21	2023	8	AAPL	175.839996
319	2023-08-22	2023	8	AAPL	177.229996
320	2023-08-23	2023	8	AAPL	181.119995
323	2023-08-24	2023	8	AAPL	176.380005
325	2023-08-25	2023	8	AAPL	178.610001
327	2023-08-28	2023	8	AAPL	180.190002
329	2023-08-29	2023	8	AAPL	184.119995
330	2023-08-30	2023	8	AAPL	187.649994

```
('GOOGL', 2023, 1)
```

	Date	Year	Month	Symbol	Adj Close
1	2023-01-03	2023	1	GOOGL	89.120003
2	2023-01-04	2023	1	GOOGL	88.080002
5	2023-01-05	2023	1	GOOGL	86.199997
7	2023-01-06	2023	1	GOOGL	87.339996
9	2023-01-09	2023	1	GOOGL	88.019997
10	2023-01-10	2023	1	GOOGL	88.419998
12	2023-01-11	2023	1	GOOGL	91.519997
14	2023-01-12	2023	1	GOOGL	91.129997
16	2023-01-13	2023	1	GOOGL	92.120003
18	2023-01-17	2023	1	GOOGL	91.290001
20	2023-01-18	2023	1	GOOGL	91.120003
22	2023-01-19	2023	1	GOOGL	93.050003
25	2023-01-20	2023	1	GOOGL	98.019997
27	2023-01-23	2023	1	GOOGL	99.790001
29	2023-01-24	2023	1	GOOGL	97.699997
30	2023-01-25	2023	1	GOOGL	95.220001
33	2023-01-26	2023	1	GOOGL	97.519997
34	2023-01-27	2023	1	GOOGL	99.370003
36	2023-01-30	2023	1	GOOGL	96.940002
38	2023-01-31	2023	1	GOOGL	98.839996

('GOOGL', 2023, 2)

	Date	Year	Month	Symbol	Adj Close
41	2023-02-01	2023	2	GOOGL	100.430000
43	2023-02-02	2023	2	GOOGL	107.739998
44	2023-02-03	2023	2	GOOGL	104.779999
47	2023-02-06	2023	2	GOOGL	102.900002
49	2023-02-07	2023	2	GOOGL	107.639999
51	2023-02-08	2023	2	GOOGL	99.370003
53	2023-02-09	2023	2	GOOGL	95.010002
55	2023-02-10	2023	2	GOOGL	94.570000
57	2023-02-13	2023	2	GOOGL	94.610001
58	2023-02-14	2023	2	GOOGL	94.680000
60	2023-02-15	2023	2	GOOGL	96.940002
63	2023-02-16	2023	2	GOOGL	95.510002
64	2023-02-17	2023	2	GOOGL	94.349998
67	2023-02-21	2023	2	GOOGL	91.790001
68	2023-02-22	2023	2	GOOGL	91.650002
70	2023-02-23	2023	2	GOOGL	90.889999
73	2023-02-24	2023	2	GOOGL	89.129997
74	2023-02-27	2023	2	GOOGL	89.870003
77	2023-02-28	2023	2	GOOGL	90.059998

('GOOGL', 2023, 3)

	Date	Year	Month	Symbol	Adj Close
78	2023-03-01	2023	3	GOOGL	90.360001
81	2023-03-02	2023	3	GOOGL	92.000000
83	2023-03-03	2023	3	GOOGL	93.650002
85	2023-03-06	2023	3	GOOGL	95.129997
87	2023-03-07	2023	3	GOOGL	93.860001
89	2023-03-08	2023	3	GOOGL	94.250000
90	2023-03-09	2023	3	GOOGL	92.320000
92	2023-03-10	2023	3	GOOGL	90.629997
94	2023-03-13	2023	3	GOOGL	91.110001
96	2023-03-14	2023	3	GOOGL	93.970001
99	2023-03-15	2023	3	GOOGL	96.110001
100	2023-03-16	2023	3	GOOGL	100.320000
103	2023-03-17	2023	3	GOOGL	101.620003
105	2023-03-20	2023	3	GOOGL	101.220001
106	2023-03-21	2023	3	GOOGL	104.919998
109	2023-03-22	2023	3	GOOGL	103.370003
110	2023-03-23	2023	3	GOOGL	105.599998

```

113 2023-03-24 2023      3  GOOGL 105.440002
115 2023-03-27 2023      3  GOOGL 102.459999
117 2023-03-28 2023      3  GOOGL 101.029999
118 2023-03-29 2023      3  GOOGL 101.389999
120 2023-03-30 2023      3  GOOGL 100.889999
122 2023-03-31 2023      3  GOOGL 103.730003
('GOOGL', 2023, 4)

```

```

      Date Year Month Symbol  Adj Close
124 2023-04-03 2023      4  GOOGL 104.360001
126 2023-04-04 2023      4  GOOGL 104.720001
128 2023-04-05 2023      4  GOOGL 104.470001
131 2023-04-06 2023      4  GOOGL 108.419998
133 2023-04-10 2023      4  GOOGL 106.440002
134 2023-04-11 2023      4  GOOGL 105.349998
136 2023-04-12 2023      4  GOOGL 104.639999
138 2023-04-13 2023      4  GOOGL 107.430000
140 2023-04-14 2023      4  GOOGL 108.870003
142 2023-04-17 2023      4  GOOGL 105.970001
144 2023-04-18 2023      4  GOOGL 104.500000
146 2023-04-19 2023      4  GOOGL 104.180000
149 2023-04-20 2023      4  GOOGL 105.290001
150 2023-04-21 2023      4  GOOGL 105.410004
152 2023-04-24 2023      4  GOOGL 105.970001
155 2023-04-25 2023      4  GOOGL 103.849998
156 2023-04-26 2023      4  GOOGL 103.709999
159 2023-04-27 2023      4  GOOGL 107.589996
160 2023-04-28 2023      4  GOOGL 107.339996
('GOOGL', 2023, 5)

```

```

      Date Year Month Symbol  Adj Close
162 2023-05-01 2023      5  GOOGL 107.199997
165 2023-05-02 2023      5  GOOGL 105.320000
166 2023-05-03 2023      5  GOOGL 105.410004
168 2023-05-04 2023      5  GOOGL 104.690002
170 2023-05-05 2023      5  GOOGL 105.570000
173 2023-05-08 2023      5  GOOGL 107.769997
175 2023-05-09 2023      5  GOOGL 107.349998
177 2023-05-10 2023      5  GOOGL 111.750000
179 2023-05-11 2023      5  GOOGL 116.570000
181 2023-05-12 2023      5  GOOGL 117.510002
182 2023-05-15 2023      5  GOOGL 116.510002
184 2023-05-16 2023      5  GOOGL 119.510002
187 2023-05-17 2023      5  GOOGL 120.839996
189 2023-05-18 2023      5  GOOGL 122.830002
191 2023-05-19 2023      5  GOOGL 122.760002
192 2023-05-22 2023      5  GOOGL 125.050003
194 2023-05-23 2023      5  GOOGL 122.559998
196 2023-05-24 2023      5  GOOGL 120.900002
198 2023-05-25 2023      5  GOOGL 123.480003
201 2023-05-26 2023      5  GOOGL 124.610001
203 2023-05-30 2023      5  GOOGL 123.669998
204 2023-05-31 2023      5  GOOGL 122.870003
('GOOGL', 2023, 6)

```

```

      Date Year Month Symbol  Adj Close
206 2023-06-01 2023      6  GOOGL 123.720001
209 2023-06-02 2023      6  GOOGL 124.669998
211 2023-06-05 2023      6  GOOGL 126.010002
213 2023-06-06 2023      6  GOOGL 127.309998
214 2023-06-07 2023      6  GOOGL 122.500000
216 2023-06-08 2023      6  GOOGL 122.139999
219 2023-06-09 2023      6  GOOGL 122.230003
220 2023-06-12 2023      6  GOOGL 123.639999

```

```

223 2023-06-13 2023      6  GOOGL 123.830002
225 2023-06-14 2023      6  GOOGL 123.669998
227 2023-06-15 2023      6  GOOGL 125.089996
229 2023-06-16 2023      6  GOOGL 123.529999
230 2023-06-20 2023      6  GOOGL 123.099998
232 2023-06-21 2023      6  GOOGL 120.550003
235 2023-06-22 2023      6  GOOGL 123.150002
236 2023-06-23 2023      6  GOOGL 122.339996
239 2023-06-26 2023      6  GOOGL 118.339996
241 2023-06-27 2023      6  GOOGL 118.330002
243 2023-06-28 2023      6  GOOGL 120.180000
244 2023-06-29 2023      6  GOOGL 119.099998
246 2023-06-30 2023      6  GOOGL 119.699997
('GOOGL', 2023, 7)

```

```

      Date Year Month Symbol  Adj Close
248 2023-07-03 2023      7  GOOGL 119.900002
251 2023-07-05 2023      7  GOOGL 121.750000
253 2023-07-06 2023      7  GOOGL 120.110001
254 2023-07-07 2023      7  GOOGL 119.480003
257 2023-07-10 2023      7  GOOGL 116.449997
259 2023-07-11 2023      7  GOOGL 117.139999
261 2023-07-12 2023      7  GOOGL 118.930000
262 2023-07-13 2023      7  GOOGL 124.540001
265 2023-07-14 2023      7  GOOGL 125.419998
266 2023-07-17 2023      7  GOOGL 124.650002
268 2023-07-18 2023      7  GOOGL 123.760002
270 2023-07-19 2023      7  GOOGL 122.029999
273 2023-07-20 2023      7  GOOGL 119.199997
275 2023-07-21 2023      7  GOOGL 120.019997
277 2023-07-24 2023      7  GOOGL 121.529999
278 2023-07-25 2023      7  GOOGL 122.209999
281 2023-07-26 2023      7  GOOGL 129.270004
282 2023-07-27 2023      7  GOOGL 129.399994
285 2023-07-28 2023      7  GOOGL 132.580002
286 2023-07-31 2023      7  GOOGL 132.720001
('GOOGL', 2023, 8)

```

```

      Date Year Month Symbol  Adj Close
289 2023-08-01 2023      8  GOOGL 131.550003
290 2023-08-02 2023      8  GOOGL 128.380005
292 2023-08-03 2023      8  GOOGL 128.449997
294 2023-08-04 2023      8  GOOGL 128.110001
297 2023-08-07 2023      8  GOOGL 131.529999
298 2023-08-08 2023      8  GOOGL 131.399994
300 2023-08-09 2023      8  GOOGL 129.660004
302 2023-08-10 2023      8  GOOGL 129.690002
305 2023-08-11 2023      8  GOOGL 129.559998
306 2023-08-14 2023      8  GOOGL 131.330002
308 2023-08-15 2023      8  GOOGL 129.779999
310 2023-08-16 2023      8  GOOGL 128.699997
312 2023-08-17 2023      8  GOOGL 129.919998
314 2023-08-18 2023      8  GOOGL 127.459999
316 2023-08-21 2023      8  GOOGL 128.369995
318 2023-08-22 2023      8  GOOGL 129.080002
321 2023-08-23 2023      8  GOOGL 132.369995
322 2023-08-24 2023      8  GOOGL 129.779999
324 2023-08-25 2023      8  GOOGL 129.880005
326 2023-08-28 2023      8  GOOGL 131.009995
328 2023-08-29 2023      8  GOOGL 134.570007
331 2023-08-30 2023      8  GOOGL 135.880005

```


In []:

1	
---	--

In [65]:

1	
---	--

('AAPL', 2023, 1)

	Date	Year	Month	Symbol	Adj Close
0	2023-01-03	2023	1	AAPL	124.538666
3	2023-01-04	2023	1	AAPL	125.823189
4	2023-01-05	2023	1	AAPL	124.488876
6	2023-01-06	2023	1	AAPL	129.069321
8	2023-01-09	2023	1	AAPL	129.597061

('AAPL', 2023, 2)

	Date	Year	Month	Symbol	Adj Close
40	2023-02-01	2023	2	AAPL	144.812164
42	2023-02-02	2023	2	AAPL	150.179291
45	2023-02-03	2023	2	AAPL	153.843628
46	2023-02-06	2023	2	AAPL	151.085388
48	2023-02-07	2023	2	AAPL	153.992981

('AAPL', 2023, 3)

	Date	Year	Month	Symbol	Adj Close
79	2023-03-01	2023	3	AAPL	144.913605
80	2023-03-02	2023	3	AAPL	145.511978
82	2023-03-03	2023	3	AAPL	150.617996
84	2023-03-06	2023	3	AAPL	153.410370
86	2023-03-07	2023	3	AAPL	151.186447

('AAPL', 2023, 4)

	Date	Year	Month	Symbol	Adj Close
125	2023-04-03	2023	4	AAPL	165.716690
127	2023-04-04	2023	4	AAPL	165.178177
129	2023-04-05	2023	4	AAPL	163.313263
130	2023-04-06	2023	4	AAPL	164.210831
132	2023-04-10	2023	4	AAPL	161.587982

('AAPL', 2023, 5)

	Date	Year	Month	Symbol	Adj Close
163	2023-05-01	2023	5	AAPL	169.127365
164	2023-05-02	2023	5	AAPL	168.080231
167	2023-05-03	2023	5	AAPL	166.993195
169	2023-05-04	2023	5	AAPL	165.337723
171	2023-05-05	2023	5	AAPL	173.096512

('AAPL', 2023, 6)

	Date	Year	Month	Symbol	Adj Close
207	2023-06-01	2023	6	AAPL	179.847137
208	2023-06-02	2023	6	AAPL	180.705978
210	2023-06-05	2023	6	AAPL	179.337830
212	2023-06-06	2023	6	AAPL	178.968338
215	2023-06-07	2023	6	AAPL	177.580200

('AAPL', 2023, 7)

	Date	Year	Month	Symbol	Adj Close
249	2023-07-03	2023	7	AAPL	192.200470
250	2023-07-05	2023	7	AAPL	191.071976
252	2023-07-06	2023	7	AAPL	191.551331
255	2023-07-07	2023	7	AAPL	190.422852
256	2023-07-10	2023	7	AAPL	188.355652

('AAPL', 2023, 8)

	Date	Year	Month	Symbol	Adj Close
288	2023-08-01	2023	8	AAPL	195.346207
291	2023-08-02	2023	8	AAPL	192.320297
293	2023-08-03	2023	8	AAPL	190.912201
295	2023-08-04	2023	8	AAPL	181.744583
296	2023-08-07	2023	8	AAPL	178.608810

('GOOGL', 2023, 1)

	Date	Year	Month	Symbol	Adj Close
1	2023-01-03	2023	1	GOOGL	89.120003
2	2023-01-04	2023	1	GOOGL	88.080002
5	2023-01-05	2023	1	GOOGL	86.199997

```

7 2023-01-06 2023      1  GOOGL  87.339996
9 2023-01-09 2023      1  GOOGL  88.019997
('GOOGL', 2023, 2)

```

```

      Date  Year  Month  Symbol  Adj Close
41 2023-02-01 2023      2  GOOGL  100.430000
43 2023-02-02 2023      2  GOOGL  107.739998
44 2023-02-03 2023      2  GOOGL  104.779999
47 2023-02-06 2023      2  GOOGL  102.900002
49 2023-02-07 2023      2  GOOGL  107.639999
('GOOGL', 2023, 3)

```

```

      Date  Year  Month  Symbol  Adj Close
78 2023-03-01 2023      3  GOOGL  90.360001
81 2023-03-02 2023      3  GOOGL  92.000000
83 2023-03-03 2023      3  GOOGL  93.650002
85 2023-03-06 2023      3  GOOGL  95.129997
87 2023-03-07 2023      3  GOOGL  93.860001
('GOOGL', 2023, 4)

```

```

      Date  Year  Month  Symbol  Adj Close
124 2023-04-03 2023      4  GOOGL  104.360001
126 2023-04-04 2023      4  GOOGL  104.720001
128 2023-04-05 2023      4  GOOGL  104.470001
131 2023-04-06 2023      4  GOOGL  108.419998
133 2023-04-10 2023      4  GOOGL  106.440002
('GOOGL', 2023, 5)

```

```

      Date  Year  Month  Symbol  Adj Close
162 2023-05-01 2023      5  GOOGL  107.199997
165 2023-05-02 2023      5  GOOGL  105.320000
166 2023-05-03 2023      5  GOOGL  105.410004
168 2023-05-04 2023      5  GOOGL  104.690002
170 2023-05-05 2023      5  GOOGL  105.570000
('GOOGL', 2023, 6)

```

```

      Date  Year  Month  Symbol  Adj Close
206 2023-06-01 2023      6  GOOGL  123.720001
209 2023-06-02 2023      6  GOOGL  124.669998
211 2023-06-05 2023      6  GOOGL  126.010002
213 2023-06-06 2023      6  GOOGL  127.309998
214 2023-06-07 2023      6  GOOGL  122.500000
('GOOGL', 2023, 7)

```

```

      Date  Year  Month  Symbol  Adj Close
248 2023-07-03 2023      7  GOOGL  119.900002
251 2023-07-05 2023      7  GOOGL  121.750000
253 2023-07-06 2023      7  GOOGL  120.110001
254 2023-07-07 2023      7  GOOGL  119.480003
257 2023-07-10 2023      7  GOOGL  116.449997
('GOOGL', 2023, 8)

```

```

      Date  Year  Month  Symbol  Adj Close
289 2023-08-01 2023      8  GOOGL  131.550003
290 2023-08-02 2023      8  GOOGL  128.380005
292 2023-08-03 2023      8  GOOGL  128.449997
294 2023-08-04 2023      8  GOOGL  128.110001
297 2023-08-07 2023      8  GOOGL  131.529999

```

Question 9-1. Let's set the index of the data to be the following three fields we are creating a multiindex

In [362]:

```
1 AAGG9_1 = AAGG7_2.set_index(['Symbol', 'Year', 'Month'])
2 AAGG9_1
```

Out[362]:

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	1	2023-01-03	124.538658
GOOGL	2023	1	2023-01-03	89.120003
		1	2023-01-04	88.080002
AAPL	2023	1	2023-01-04	125.823189
		1	2023-01-05	124.488876
...		
8		2023-08-28	180.190002	
GOOGL	2023	8	2023-08-29	134.570007
AAPL	2023	8	2023-08-29	184.119995
		8	2023-08-30	187.649994
GOOGL	2023	8	2023-08-30	135.880005

332 rows × 2 columns

In [66]:

```
1
```

Out[66]:

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	1	2023-01-03	124.538666
GOOGL	2023	1	2023-01-03	89.120003
		1	2023-01-04	88.080002
AAPL	2023	1	2023-01-04	125.823189
		1	2023-01-05	124.488876
	
		8	2023-08-28	180.190002
GOOGL	2023	8	2023-08-29	134.570007
AAPL	2023	8	2023-08-29	184.119995
		8	2023-08-30	187.649994
GOOGL	2023	8	2023-08-30	135.880005

332 rows × 2 columns

Question 9-2. Then, please group based upon values in the actual index the following groups by level 0 of the index (Month)

In [364]:

```
1 print(AAGG9_1.index.levels)
```

[['AAPL', 'GOOGL'], [2023], [1, 2, 3, 4, 5, 6, 7, 8]]

In [373]:

```
1 AAGG9_1.index.get_level_values(2)
2 # there might be some typo errors in the description
3 # Lv 0 is symbol; lv 1 is year; lv 2 is month
4 # but as set to Lv = 0, the grouped object is just as the supposed output shows
```

Out[373]:

Index([1, 1, 1, 1, 1, 1, 1, 1, 1,
...
8, 8, 8, 8, 8, 8, 8, 8, 8],
dtype='int32', name='Month', length=332)

In [378]:

```
1 AAGG9_2_grouped = AAGG9_1.groupby(level = 0)
```

In [379]:

```
1 print_grouped_df(AAGG9_2_grouped)
```

AAPL			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	1	2023-01-03	124.538658
		1	2023-01-04	125.823189
		1	2023-01-05	124.488876
		1	2023-01-06	129.069321
		1	2023-01-09	129.597061
...		
		8	2023-08-24	176.380005
		8	2023-08-25	178.610001
		8	2023-08-28	180.190002
		8	2023-08-29	184.119995
		8	2023-08-30	187.649994

[166 rows x 5 columns]
GOOGL

			Date	Adj Close
Symbol	Year	Month		
GOOGL	2023	1	2023-01-03	89.120003
		1	2023-01-04	88.080002
		1	2023-01-05	86.199997
		1	2023-01-06	87.339996
		1	2023-01-09	88.019997
...		
		8	2023-08-24	129.779999
		8	2023-08-25	129.880005
		8	2023-08-28	131.009995
		8	2023-08-29	134.570007
		8	2023-08-30	135.880005

[166 rows x 5 columns]

In []:

```
1
```

In [67]:

```
1
```

AAPL			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	1	2023-01-03	124.538666
		1	2023-01-04	125.823189
		1	2023-01-05	124.488876
		1	2023-01-06	129.069321
		1	2023-01-09	129.597061
GOOGL			Date	Adj Close
Symbol	Year	Month		
GOOGL	2023	1	2023-01-03	89.120003
		1	2023-01-04	88.080002
		1	2023-01-05	86.199997
		1	2023-01-06	87.339996
		1	2023-01-09	88.019997

Question 9-3. Please group by three levels in the index using their names

In [380]:

```
1 AAGG9_3_grouped = AAGG9_1.groupby(level=['Symbol', 'Year', 'Month'])
```


In [381]:

```
1 print_grouped_df(AAGG9_3_grouped)
```

(' AAPL' , 2023, 1)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	1	2023-01-03	124.538658
		1	2023-01-04	125.823189
		1	2023-01-05	124.488876
		1	2023-01-06	129.069321
		1	2023-01-09	129.597061
		1	2023-01-10	130.174606
		1	2023-01-11	132.922897
		1	2023-01-12	132.843231
		1	2023-01-13	134.187500
		1	2023-01-17	135.362473
		1	2023-01-18	134.635590
		1	2023-01-19	134.695328
		1	2023-01-20	137.284286
		1	2023-01-23	140.510513
		1	2023-01-24	141.924500
		1	2023-01-25	141.257324
		1	2023-01-26	143.348419
		1	2023-01-27	145.310043
		1	2023-01-30	142.392487
		1	2023-01-31	143.676987

(' AAPL' , 2023, 2)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	2	2023-02-01	144.812164
		2	2023-02-02	150.179276
		2	2023-02-03	153.843628
		2	2023-02-06	151.085403
		2	2023-02-07	153.992996
		2	2023-02-08	151.274597
		2	2023-02-09	150.229065
		2	2023-02-10	150.598038
		2	2023-02-13	153.430313
		2	2023-02-14	152.782074
		2	2023-02-15	154.906265
		2	2023-02-16	153.290695
		2	2023-02-17	152.133850
		2	2023-02-21	148.074951
		2	2023-02-22	148.503784
		2	2023-02-23	148.992432
		2	2023-02-24	146.309784
		2	2023-02-27	147.516479
		2	2023-02-28	147.007874

(' AAPL' , 2023, 3)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	3	2023-03-01	144.913589
		3	2023-03-02	145.511963
		3	2023-03-03	150.617996
		3	2023-03-06	153.410355
		3	2023-03-07	151.186447
		3	2023-03-08	152.452972
		3	2023-03-09	150.179184
		3	2023-03-10	148.094894
		3	2023-03-13	150.059525
		3	2023-03-14	152.173737
		3	2023-03-15	152.572662
		3	2023-03-16	155.424850
		3	2023-03-17	154.577164

3	2023-03-20	156.970612
3	2023-03-21	158.845474
3	2023-03-22	157.399445
3	2023-03-23	158.496429
3	2023-03-24	159.812851
3	2023-03-27	157.848221
3	2023-03-28	157.219925
3	2023-03-29	160.331436
3	2023-03-30	161.917099
3	2023-03-31	164.450150

(' AAPL' , 2023, 4)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	4	2023-04-03	165.716690
		4	2023-04-04	165.178177
		4	2023-04-05	163.313263
		4	2023-04-06	164.210815
		4	2023-04-10	161.587982
		4	2023-04-11	160.361343
		4	2023-04-12	159.663254
		4	2023-04-13	165.108353
		4	2023-04-14	164.759323
		4	2023-04-17	164.779251
		4	2023-04-18	166.015884
		4	2023-04-19	167.172729
		4	2023-04-20	166.195374
		4	2023-04-21	164.569839
		4	2023-04-24	164.878983
		4	2023-04-25	163.323242
		4	2023-04-26	163.313263
		4	2023-04-27	167.950592
		4	2023-04-28	169.217117

(' AAPL' , 2023, 5)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	5	2023-05-01	169.127365
		5	2023-05-02	168.080215
		5	2023-05-03	166.993195
		5	2023-05-04	165.337723
		5	2023-05-05	173.096512
		5	2023-05-08	173.026688
		5	2023-05-09	171.301422
		5	2023-05-10	173.086533
		5	2023-05-11	173.276016
		5	2023-05-12	172.337280
		5	2023-05-15	171.837967
		5	2023-05-16	171.837967
		5	2023-05-17	172.457123
		5	2023-05-18	174.813934
		5	2023-05-19	174.923798
		5	2023-05-22	173.965073
		5	2023-05-23	171.328644
		5	2023-05-24	171.608261
		5	2023-05-25	172.756714
		5	2023-05-26	175.193420
		5	2023-05-30	177.060898
		5	2023-05-31	177.010971

(' AAPL' , 2023, 6)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	6	2023-06-01	179.847137

6	2023-06-02	180.705978
6	2023-06-05	179.337830
6	2023-06-06	178.968338
6	2023-06-07	177.580200
6	2023-06-08	180.326492
6	2023-06-09	180.715973
6	2023-06-12	183.542145
6	2023-06-13	183.062790
6	2023-06-14	183.701935
6	2023-06-15	185.759155
6	2023-06-16	184.670624
6	2023-06-20	184.760498
6	2023-06-21	183.711929
6	2023-06-22	186.747818
6	2023-06-23	186.428238
6	2023-06-26	185.020157
6	2023-06-27	187.806381
6	2023-06-28	188.994781
6	2023-06-29	189.334320
6	2023-06-30	193.708420

('AAPL', 2023, 7)

Symbol	Year	Month	Date	Adj Close
AAPL	2023	7	2023-07-03	192.200470
		7	2023-07-05	191.071976
		7	2023-07-06	191.551331
		7	2023-07-07	190.422852
		7	2023-07-10	188.355652
		7	2023-07-11	187.826370
		7	2023-07-12	189.514084
		7	2023-07-13	190.283035
		7	2023-07-14	190.432846
		7	2023-07-17	193.728394
		7	2023-07-18	193.468735
		7	2023-07-19	194.836899
		7	2023-07-20	192.869553
		7	2023-07-21	191.681168
		7	2023-07-24	192.490067
		7	2023-07-25	193.358887
		7	2023-07-26	194.237701
		7	2023-07-27	192.959427
		7	2023-07-28	195.565918
		7	2023-07-31	196.185074

('AAPL', 2023, 8)

Symbol	Year	Month	Date	Adj Close
AAPL	2023	8	2023-08-01	195.346207
		8	2023-08-02	192.320297
		8	2023-08-03	190.912201
		8	2023-08-04	181.744583
		8	2023-08-07	178.608810
		8	2023-08-08	179.557526
		8	2023-08-09	177.949707
		8	2023-08-10	177.729996
		8	2023-08-11	177.789993
		8	2023-08-14	179.460007
		8	2023-08-15	177.449997
		8	2023-08-16	176.570007
		8	2023-08-17	174.000000
		8	2023-08-18	174.490005
		8	2023-08-21	175.839996

```
8      2023-08-22  177.229996
8      2023-08-23  181.119995
8      2023-08-24  176.380005
8      2023-08-25  178.610001
8      2023-08-28  180.190002
8      2023-08-29  184.119995
8      2023-08-30  187.649994
('GOOGL', 2023, 1)
```

```
      Date Adj Close
Symbol Year Month
GOOGL  2023 1      2023-01-03  89.120003
      1      2023-01-04  88.080002
      1      2023-01-05  86.199997
      1      2023-01-06  87.339996
      1      2023-01-09  88.019997
      1      2023-01-10  88.419998
      1      2023-01-11  91.519997
      1      2023-01-12  91.129997
      1      2023-01-13  92.120003
      1      2023-01-17  91.290001
      1      2023-01-18  91.120003
      1      2023-01-19  93.050003
      1      2023-01-20  98.019997
      1      2023-01-23  99.790001
      1      2023-01-24  97.699997
      1      2023-01-25  95.220001
      1      2023-01-26  97.519997
      1      2023-01-27  99.370003
      1      2023-01-30  96.940002
      1      2023-01-31  98.839996
('GOOGL', 2023, 2)
```

```
      Date Adj Close
Symbol Year Month
GOOGL  2023 2      2023-02-01  100.430000
      2      2023-02-02  107.739998
      2      2023-02-03  104.779999
      2      2023-02-06  102.900002
      2      2023-02-07  107.639999
      2      2023-02-08   99.370003
      2      2023-02-09  95.010002
      2      2023-02-10  94.570000
      2      2023-02-13  94.610001
      2      2023-02-14  94.680000
      2      2023-02-15  96.940002
      2      2023-02-16  95.510002
      2      2023-02-17  94.349998
      2      2023-02-21  91.790001
      2      2023-02-22  91.650002
      2      2023-02-23  90.889999
      2      2023-02-24  89.129997
      2      2023-02-27  89.870003
      2      2023-02-28  90.059998
('GOOGL', 2023, 3)
```

```
      Date Adj Close
Symbol Year Month
GOOGL  2023 3      2023-03-01  90.360001
      3      2023-03-02  92.000000
      3      2023-03-03  93.650002
      3      2023-03-06  95.129997
      3      2023-03-07  93.860001
      3      2023-03-08  94.250000
```

```
3      2023-03-09    92.320000
3      2023-03-10    90.629997
3      2023-03-13    91.110001
3      2023-03-14    93.970001
3      2023-03-15    96.110001
3      2023-03-16   100.320000
3      2023-03-17   101.620003
3      2023-03-20   101.220001
3      2023-03-21   104.919998
3      2023-03-22   103.370003
3      2023-03-23   105.599998
3      2023-03-24   105.440002
3      2023-03-27   102.459999
3      2023-03-28   101.029999
3      2023-03-29   101.389999
3      2023-03-30   100.889999
3      2023-03-31   103.730003
('GOOGL', 2023, 4)
```

```
          Date    Adj Close
Symbol Year Month
GOOGL  2023  4      2023-04-03   104.360001
          4      2023-04-04   104.720001
          4      2023-04-05   104.470001
          4      2023-04-06   108.419998
          4      2023-04-10   106.440002
          4      2023-04-11   105.349998
          4      2023-04-12   104.639999
          4      2023-04-13   107.430000
          4      2023-04-14   108.870003
          4      2023-04-17   105.970001
          4      2023-04-18   104.500000
          4      2023-04-19   104.180000
          4      2023-04-20   105.290001
          4      2023-04-21   105.410004
          4      2023-04-24   105.970001
          4      2023-04-25   103.849998
          4      2023-04-26   103.709999
          4      2023-04-27   107.589996
          4      2023-04-28   107.339996
```

```
('GOOGL', 2023, 5)
          Date    Adj Close
Symbol Year Month
GOOGL  2023  5      2023-05-01   107.199997
          5      2023-05-02   105.320000
          5      2023-05-03   105.410004
          5      2023-05-04   104.690002
          5      2023-05-05   105.570000
          5      2023-05-08   107.769997
          5      2023-05-09   107.349998
          5      2023-05-10   111.750000
          5      2023-05-11   116.570000
          5      2023-05-12   117.510002
          5      2023-05-15   116.510002
          5      2023-05-16   119.510002
          5      2023-05-17   120.839996
          5      2023-05-18   122.830002
          5      2023-05-19   122.760002
          5      2023-05-22   125.050003
          5      2023-05-23   122.559998
          5      2023-05-24   120.900002
          5      2023-05-25   123.480003
```

```
5      2023-05-26  124.610001
5      2023-05-30  123.669998
5      2023-05-31  122.870003
('GOOGL', 2023, 6)
```

			Date	Adj Close
Symbol	Year	Month		
GOOGL	2023	6	2023-06-01	123.720001
		6	2023-06-02	124.669998
		6	2023-06-05	126.010002
		6	2023-06-06	127.309998
		6	2023-06-07	122.500000
		6	2023-06-08	122.139999
		6	2023-06-09	122.230003
		6	2023-06-12	123.639999
		6	2023-06-13	123.830002
		6	2023-06-14	123.669998
		6	2023-06-15	125.089996
		6	2023-06-16	123.529999
		6	2023-06-20	123.099998
		6	2023-06-21	120.550003
		6	2023-06-22	123.150002
		6	2023-06-23	122.339996
		6	2023-06-26	118.339996
		6	2023-06-27	118.330002
		6	2023-06-28	120.180000
		6	2023-06-29	119.099998
		6	2023-06-30	119.699997

```
('GOOGL', 2023, 7)
```

			Date	Adj Close
Symbol	Year	Month		
GOOGL	2023	7	2023-07-03	119.900002
		7	2023-07-05	121.750000
		7	2023-07-06	120.110001
		7	2023-07-07	119.480003
		7	2023-07-10	116.449997
		7	2023-07-11	117.139999
		7	2023-07-12	118.930000
		7	2023-07-13	124.540001
		7	2023-07-14	125.419998
		7	2023-07-17	124.650002
		7	2023-07-18	123.760002
		7	2023-07-19	122.029999
		7	2023-07-20	119.199997
		7	2023-07-21	120.019997
		7	2023-07-24	121.529999
		7	2023-07-25	122.209999
		7	2023-07-26	129.270004
		7	2023-07-27	129.399994
		7	2023-07-28	132.580002
		7	2023-07-31	132.720001

```
('GOOGL', 2023, 8)
```

			Date	Adj Close
Symbol	Year	Month		
GOOGL	2023	8	2023-08-01	131.550003
		8	2023-08-02	128.380005
		8	2023-08-03	128.449997
		8	2023-08-04	128.110001
		8	2023-08-07	131.529999
		8	2023-08-08	131.399994
		8	2023-08-09	129.660004
		8	2023-08-10	129.690002

```
8      2023-08-11  129.559998
8      2023-08-14  131.330002
8      2023-08-15  129.779999
8      2023-08-16  128.699997
8      2023-08-17  129.919998
8      2023-08-18  127.459999
8      2023-08-21  128.369995
8      2023-08-22  129.080002
8      2023-08-23  132.369995
8      2023-08-24  129.779999
8      2023-08-25  129.880005
8      2023-08-28  131.009995
8      2023-08-29  134.570007
8      2023-08-30  135.880005
```

In []:

1	
---	--

In [69]:

1	
---	--

(' AAPL' , 2023, 1)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	1	2023-01-03	124.538666
		1	2023-01-04	125.823189
		1	2023-01-05	124.488876
		1	2023-01-06	129.069321
		1	2023-01-09	129.597061

(' AAPL' , 2023, 2)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	2	2023-02-01	144.812164
		2	2023-02-02	150.179291
		2	2023-02-03	153.843628
		2	2023-02-06	151.085388
		2	2023-02-07	153.992981

(' AAPL' , 2023, 3)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	3	2023-03-01	144.913605
		3	2023-03-02	145.511978
		3	2023-03-03	150.617996
		3	2023-03-06	153.410370
		3	2023-03-07	151.186447

(' AAPL' , 2023, 4)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	4	2023-04-03	165.716690
		4	2023-04-04	165.178177
		4	2023-04-05	163.313263
		4	2023-04-06	164.210831
		4	2023-04-10	161.587982

(' AAPL' , 2023, 5)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	5	2023-05-01	169.127365
		5	2023-05-02	168.080231
		5	2023-05-03	166.993195
		5	2023-05-04	165.337723
		5	2023-05-05	173.096512

(' AAPL' , 2023, 6)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	6	2023-06-01	179.847137
		6	2023-06-02	180.705978
		6	2023-06-05	179.337830
		6	2023-06-06	178.968338
		6	2023-06-07	177.580200

(' AAPL' , 2023, 7)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	7	2023-07-03	192.200470
		7	2023-07-05	191.071976
		7	2023-07-06	191.551331
		7	2023-07-07	190.422852
		7	2023-07-10	188.355652

(' AAPL' , 2023, 8)

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	8	2023-08-01	195.346207
		8	2023-08-02	192.320297

	8	2023-08-03	190.912201
	8	2023-08-04	181.744583
	8	2023-08-07	178.608810
('GOOGL', 2023, 1)			
		Date	Adj Close
Symbol	Year	Month	
GOOGL	2023	1	2023-01-03 89.120003
		1	2023-01-04 88.080002
		1	2023-01-05 86.199997
		1	2023-01-06 87.339996
		1	2023-01-09 88.019997
('GOOGL', 2023, 2)			
		Date	Adj Close
Symbol	Year	Month	
GOOGL	2023	2	2023-02-01 100.430000
		2	2023-02-02 107.739998
		2	2023-02-03 104.779999
		2	2023-02-06 102.900002
		2	2023-02-07 107.639999
('GOOGL', 2023, 3)			
		Date	Adj Close
Symbol	Year	Month	
GOOGL	2023	3	2023-03-01 90.360001
		3	2023-03-02 92.000000
		3	2023-03-03 93.650002
		3	2023-03-06 95.129997
		3	2023-03-07 93.860001
('GOOGL', 2023, 4)			
		Date	Adj Close
Symbol	Year	Month	
GOOGL	2023	4	2023-04-03 104.360001
		4	2023-04-04 104.720001
		4	2023-04-05 104.470001
		4	2023-04-06 108.419998
		4	2023-04-10 106.440002
('GOOGL', 2023, 5)			
		Date	Adj Close
Symbol	Year	Month	
GOOGL	2023	5	2023-05-01 107.199997
		5	2023-05-02 105.320000
		5	2023-05-03 105.410004
		5	2023-05-04 104.690002
		5	2023-05-05 105.570000
('GOOGL', 2023, 6)			
		Date	Adj Close
Symbol	Year	Month	
GOOGL	2023	6	2023-06-01 123.720001
		6	2023-06-02 124.669998
		6	2023-06-05 126.010002
		6	2023-06-06 127.309998
		6	2023-06-07 122.500000
('GOOGL', 2023, 7)			
		Date	Adj Close
Symbol	Year	Month	
GOOGL	2023	7	2023-07-03 119.900002
		7	2023-07-05 121.750000
		7	2023-07-06 120.110001
		7	2023-07-07 119.480003
		7	2023-07-10 116.449997
('GOOGL', 2023, 8)			
		Date	Adj Close

Symbol	Year	Month		
GOOGL	2023	8	2023-08-01	131.550003
		8	2023-08-02	128.380005
		8	2023-08-03	128.449997
		8	2023-08-04	128.110001
		8	2023-08-07	131.529999

Remember `agg` ?

Question 10-1. Use `numpy` apply the mean function to each group.

In [391]:

```
1 mean10_1 = AAGG9_3_grouped.agg('mean')
2 mean10_1
```

Out[391]:

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	1	2023-01-16 15:36:00.000000000	135.202164
		2	2023-02-14 00:00:00.000000000	150.471772
		3	2023-03-16 08:20:52.173913088	154.542043
		4	2023-04-15 22:44:12.631578880	164.595551
		5	2023-05-15 01:05:27.272727296	172.293533
		6	2023-06-15 21:42:51.428571392	184.034816
		7	2023-07-16 20:24:00.000000000	192.152022
		8	2023-08-15 07:38:10.909090816	180.684969
GOOGL	2023	1	2023-01-16 15:36:00.000000000	93.040499
		2	2023-02-14 00:00:00.000000000	96.416842
		3	2023-03-16 08:20:52.173913088	98.060000
		4	2023-04-15 22:44:12.631578880	105.711053
		5	2023-05-15 01:05:27.272727296	116.124091
		6	2023-06-15 21:42:51.428571392	122.530000
		7	2023-07-16 20:24:00.000000000	123.054500
		8	2023-08-15 07:38:10.909090816	130.293636

In [70]:

1	
---	--

Out[70]:

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	1	2023-01-16 15:36:00.000000000	135.202166
		2	2023-02-14 00:00:00.000000000	150.471768
		3	2023-03-16 08:20:52.173913088	154.542043
		4	2023-04-15 22:44:12.631578880	164.595553
		5	2023-05-15 01:05:27.272727296	172.293533
		6	2023-06-15 21:42:51.428571392	184.034816
		7	2023-07-16 20:24:00.000000000	192.152022
		8	2023-08-15 07:38:10.909090816	180.684969
GOOGL	2023	1	2023-01-16 15:36:00.000000000	93.040499
		2	2023-02-14 00:00:00.000000000	96.416842
		3	2023-03-16 08:20:52.173913088	98.060000
		4	2023-04-15 22:44:12.631578880	105.711053
		5	2023-05-15 01:05:27.272727296	116.124091
		6	2023-06-15 21:42:51.428571392	122.530000
		7	2023-07-16 20:24:00.000000000	123.054500
		8	2023-08-15 07:38:10.909090816	130.293636

Question 10-2. How would you obtain the below then?

In [504]:

```
1 # AAGG9_3_grouped = AAGG9_1.groupby(level=['Symbol', 'Year', 'Month'])
2 AAGG9_1
```

Out[504]:

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	1	2023-01-03	124.538658
GOOGL	2023	1	2023-01-03	89.120003
		1	2023-01-04	88.080002
AAPL	2023	1	2023-01-04	125.823189
		1	2023-01-05	124.488876
	
		8	2023-08-28	180.190002
GOOGL	2023	8	2023-08-29	134.570007
AAPL	2023	8	2023-08-29	184.119995
		8	2023-08-30	187.649994
GOOGL	2023	8	2023-08-30	135.880005

332 rows × 2 columns

In [497]:

```
1 length = AAGG9_3_grouped.size().max()
2 length
```

Out[497]:

23

In [505]:

```
1 AAGG10_2 = AAGG9_3_grouped[['Date', 'Adj Close']].head(length)
2 AAGG10_2
```

Out[505]:

			Date	Adj Close
Symbol	Year	Month		
AAPL	2023	1	2023-01-03	124.538658
GOOGL	2023	1	2023-01-03	89.120003
		1	2023-01-04	88.080002
AAPL	2023	1	2023-01-04	125.823189
		1	2023-01-05	124.488876
	
		8	2023-08-28	180.190002
GOOGL	2023	8	2023-08-29	134.570007
AAPL	2023	8	2023-08-29	184.119995
		8	2023-08-30	187.649994
GOOGL	2023	8	2023-08-30	135.880005

332 rows × 2 columns

In [513]:

```
1 AAGG10_2 = AAGG10_2.reset_index(drop=True)
2 AAGG10_2
```

Out[513]:

	Date	Year	Month	Symbol	Adj Close
0	2023-01-03	2023	1	AAPL	124.538658
1	2023-01-03	2023	1	GOOGL	89.120003
2	2023-01-04	2023	1	GOOGL	88.080002
3	2023-01-04	2023	1	AAPL	125.823189
4	2023-01-05	2023	1	AAPL	124.488876
...
327	2023-08-28	2023	8	AAPL	180.190002
328	2023-08-29	2023	8	GOOGL	134.570007
329	2023-08-29	2023	8	AAPL	184.119995
330	2023-08-30	2023	8	AAPL	187.649994
331	2023-08-30	2023	8	GOOGL	135.880005

332 rows × 5 columns

In [514]:

```
1 AAGG10_2 = AAGG10_2[['Date', 'Year', 'Month', 'Symbol', 'Adj Close']]
2 AAGG10_2
```

Out[514]:

	Date	Year	Month	Symbol	Adj Close
0	2023-01-03	2023	1	AAPL	124.538658
1	2023-01-03	2023	1	GOOGL	89.120003
2	2023-01-04	2023	1	GOOGL	88.080002
3	2023-01-04	2023	1	AAPL	125.823189
4	2023-01-05	2023	1	AAPL	124.488876
...
327	2023-08-28	2023	8	AAPL	180.190002
328	2023-08-29	2023	8	GOOGL	134.570007
329	2023-08-29	2023	8	AAPL	184.119995
330	2023-08-30	2023	8	AAPL	187.649994
331	2023-08-30	2023	8	GOOGL	135.880005

332 rows × 5 columns

In []:

```
1
```

In [71]:

```
1
```

Out[71]:

	Date	Year	Month	Symbol	Adj Close
0	2023-01-03	2023	1	AAPL	124.538666
1	2023-01-03	2023	1	GOOGL	89.120003
2	2023-01-04	2023	1	GOOGL	88.080002
3	2023-01-04	2023	1	AAPL	125.823189
4	2023-01-05	2023	1	AAPL	124.488876
...
327	2023-08-28	2023	8	AAPL	180.190002
328	2023-08-29	2023	8	GOOGL	134.570007
329	2023-08-29	2023	8	AAPL	184.119995
330	2023-08-30	2023	8	AAPL	187.649994
331	2023-08-30	2023	8	GOOGL	135.880005

332 rows × 5 columns

In [425]:

```
1 apple_mean_10_1 = AAGG9_3_grouped.agg({'Date': 'mean', 'Adj Close': 'mean'}).reset_index()
2 apple_mean_10_1 = apple_mean_10_1.loc[apple_mean_10_1['Symbol']=='AAPL']
3 apple_mean_10_1
```

Out[425]:

	Symbol	Year	Month	Date	Adj Close
0	AAPL	2023	1	2023-01-16 15:36:00.000000000	135.202164
1	AAPL	2023	2	2023-02-14 00:00:00.000000000	150.471772
2	AAPL	2023	3	2023-03-16 08:20:52.173913088	154.542043
3	AAPL	2023	4	2023-04-15 22:44:12.631578880	164.595551
4	AAPL	2023	5	2023-05-15 01:05:27.272727296	172.293533
5	AAPL	2023	6	2023-06-15 21:42:51.428571392	184.034816
6	AAPL	2023	7	2023-07-16 20:24:00.000000000	192.152022
7	AAPL	2023	8	2023-08-15 07:38:10.909090816	180.684969

In [72]:

```
1
```

Out[72]:

	Symbol	Year	Month	Date	Adj Close
0	AAPL	2023	1	2023-01-16 15:36:00.000000000	135.202166
1	AAPL	2023	2	2023-02-14 00:00:00.000000000	150.471768
2	AAPL	2023	3	2023-03-16 08:20:52.173913088	154.542043
3	AAPL	2023	4	2023-04-15 22:44:12.631578880	164.595553
4	AAPL	2023	5	2023-05-15 01:05:27.272727296	172.293533

Question 10-3. Please apply multiple functions to each group in one call so that you can obtain the below.

In [427]:

```
1 mean_10_3 = AAGG9_3_grouped.agg({'Date': ('mean', 'std'), 'Adj Close': ('mean', 'std')})
2
3 mean_10_3
```

Out[427]:

			Date		Adj Close	
			mean	std	mean	std
Symbol	Year	Month				
AAPL	2023	1	2023-01-16	8 days	135.202164	6.607694
			15:36:00.000000000	23:07:28.349947786		
		2	2023-02-14	8 days	150.471772	2.899289
			00:00:00.000000000	11:48:13.778084579		
		3	2023-03-16	9 days	154.542043	5.124342
			08:20:52.173913088	11:31:26.207511746		
		4	2023-04-15	8 days	164.595551	2.382711
			22:44:12.631578880	03:03:16.733566575		
GOOGL	2023	5	2023-05-15	9 days	172.293533	2.923869
			01:05:27.272727296	05:38:16.163540207		
		6	2023-06-15	9 days	184.034816	4.016423
			21:42:51.428571392	06:41:01.999313202		
		7	2023-07-16	8 days	192.152022	2.267709
			20:24:00.000000000	09:17:32.086684966		
		8	2023-08-15	9 days	180.684969	5.843950
			07:38:10.909090816	03:15:45.066017849		
		1	2023-01-16	8 days	93.040499	4.502412
			15:36:00.000000000	23:07:28.349947786		
		2	2023-02-14	8 days	96.416842	5.864478
			00:00:00.000000000	11:48:13.778084579		
		3	2023-03-16	9 days	98.060000	5.252508
			08:20:52.173913088	11:31:26.207511746		
		4	2023-04-15	8 days	105.711053	1.568672
			22:44:12.631578880	03:03:16.733566575		
		5	2023-05-15	9 days	116.124091	7.611528
			01:05:27.272727296	05:38:16.163540207		
		6	2023-06-15	9 days	122.530000	2.430339
			21:42:51.428571392	06:41:01.999313202		
		7	2023-07-16	8 days	123.054500	4.761495
			20:24:00.000000000	09:17:32.086684966		
		8	2023-08-15	9 days	130.293636	2.063730
			07:38:10.909090816	03:15:45.066017849		

In []:

```
1
```

In [73]:

1	
---	--

Out[73]:

Symbol	Year	Month	Date		Adj Close	
			mean	std	mean	std
AAPL	2023	1	2023-01-16	8 days	135.202166	6.607695
			15:36:00.000000000	23:07:28.349947786		
		2	2023-02-14	8 days	150.471768	2.899287
			00:00:00.000000000	11:48:13.778084579		
		3	2023-03-16	9 days	154.542043	5.124338
			08:20:52.173913088	11:31:26.207511746		
		4	2023-04-15	8 days	164.595553	2.382709
			22:44:12.631578880	03:03:16.733566575		
GOOGL	2023	5	2023-05-15	9 days	172.293533	2.923868
			01:05:27.272727296	05:38:16.163540207		
		6	2023-06-15	9 days	184.034816	4.016423
			21:42:51.428571392	06:41:01.999313202		
		7	2023-07-16	8 days	192.152022	2.267709
			20:24:00.000000000	09:17:32.086684966		
		8	2023-08-15	9 days	180.684969	5.843950
			07:38:10.909090816	03:15:45.066017849		
		1	2023-01-16	8 days	93.040499	4.502412
			15:36:00.000000000	23:07:28.349947786		
		2	2023-02-14	8 days	96.416842	5.864478
			00:00:00.000000000	11:48:13.778084579		
		3	2023-03-16	9 days	98.060000	5.252508
			08:20:52.173913088	11:31:26.207511746		
		4	2023-04-15	8 days	105.711053	1.568672
			22:44:12.631578880	03:03:16.733566575		
		5	2023-05-15	9 days	116.124091	7.611528
			01:05:27.272727296	05:38:16.163540207		
		6	2023-06-15	9 days	122.530000	2.430339
			21:42:51.428571392	06:41:01.999313202		
		7	2023-07-16	8 days	123.054500	4.761495
			20:24:00.000000000	09:17:32.086684966		
		8	2023-08-15	9 days	130.293636	2.063730
			07:38:10.909090816	03:15:45.066017849		

Now let's do something with a 2-D array of Dow Jones Industrial Average in 2008

The array `data_problem_sets_3.csv` is a 2-D array with each row holding the daily performance of the Dow Jones Industrial Average from the beginning of 2008 (dates have been removed for exercise simplicity). The array has the following structure::

OPEN	HIGH	LOW	CLOSE	VOLUME	ADJ_CLOSE
13261.82	13338.23	12969.42	13043.96	3452650000	13043.96
13044.12	13197.43	12968.44	13056.72	3429500000	13056.72
13046.56	13049.65	12740.51	12800.18	4166000000	12800.18
12801.15	12984.95	12640.44	12827.49	4221260000	12827.49

Below you will find the necessary modules for the task.

In [435]:

```
1 %matplotlib inline
2 from numpy import loadtxt, sum, where
3 import matplotlib.pyplot as plt
```

Below are constants that indicate what data is held in each column of the `dow` array.

- OPEN = 0
- HIGH = 1
- LOW = 2
- CLOSE = 3
- VOLUME = 4
- ADJ_CLOSE = 5

`data_problem_sets_3.csv` is our NumPy array that you will manipulate. Below you will find another way to load your csv file :)

In [436]:

```
1 dow = loadtxt("data_problem_sets_3.csv", delimiter=",")
2 dow
```

Out[436]:

```
array([[1.326182e+04, 1.333823e+04, 1.296942e+04, 1.304396e+04,
        3.452650e+09, 1.304396e+04],
       [1.304412e+04, 1.319743e+04, 1.296844e+04, 1.305672e+04,
        3.429500e+09, 1.305672e+04],
       [1.304656e+04, 1.304965e+04, 1.274051e+04, 1.280018e+04,
        4.166000e+09, 1.280018e+04],
       ...,
       [1.141246e+04, 1.157514e+04, 1.134969e+04, 1.150251e+04,
        3.499610e+09, 1.150251e+04],
       [1.149987e+04, 1.175646e+04, 1.149372e+04, 1.171518e+04,
        3.854280e+09, 1.171518e+04],
       [1.171323e+04, 1.173049e+04, 1.150878e+04, 1.154355e+04,
        3.288120e+09, 1.154355e+04]])
```

Question 11-1. Please create a `mask` array that indicates which rows have a volume greater than 5.5 billion.

Hint: The mask refers to boolean masking. How would you go about doing boolean masking?
How will you apply the `mask` to the dataframe?

In [438]:

```
1 dow.shape
```

Out[438]:

```
(168, 6)
```

In [442]:

1	dow[:, 4]
---	-----------

Out[442]:

```
array([3.4526500e+09, 3.4295000e+09, 4.1660000e+09, 4.2212600e+09,
       4.7053900e+09, 5.3510300e+09, 5.1704900e+09, 4.4958400e+09,
       3.6820900e+09, 4.6016400e+09, 5.4406200e+09, 5.3031300e+09,
       6.0048400e+09, 6.5446900e+09, 3.2416800e+09, 5.7353000e+09,
       4.8822500e+09, 4.1009300e+09, 4.2329600e+09, 4.7427600e+09,
       4.9702900e+09, 4.6507700e+09, 3.4957800e+09, 4.3157400e+09,
       4.0081200e+09, 4.5891600e+09, 3.7684900e+09, 3.5931400e+09,
       4.0446400e+09, 3.8564200e+09, 3.6447600e+09, 3.5833000e+09,
       3.6135500e+09, 3.8705200e+09, 3.6966600e+09, 3.5726600e+09,
       3.8663500e+09, 4.0960600e+09, 3.9047000e+09, 3.9385800e+09,
       4.4267300e+09, 4.1175700e+09, 4.7571800e+09, 4.2777100e+09,
       4.3234600e+09, 4.5654100e+09, 4.2612400e+09, 5.1090800e+09,
       4.4142800e+09, 5.0733600e+09, 5.1537800e+09, 5.6830100e+09,
       5.3356300e+09, 1.2038300e+09, 6.1452200e+09, 4.4990000e+09,
       4.1451200e+09, 4.0556700e+09, 4.0379300e+09, 3.6869800e+09,
       4.1889900e+09, 4.7451200e+09, 4.3204400e+09, 3.9201000e+09,
       3.7031000e+09, 3.7477800e+09, 3.6025000e+09, 3.5566700e+09,
       3.6861500e+09, 3.7237900e+09, 3.5650200e+09, 3.5812300e+09,
       4.2603700e+09, 3.7138800e+09, 4.2223800e+09, 3.4205700e+09,
       3.8219000e+09, 4.1036100e+09, 4.4616600e+09, 3.8911500e+09,
       3.6070000e+09, 3.8153200e+09, 4.5088900e+09, 4.4487800e+09,
       3.9530300e+09, 3.4100900e+09, 3.9241000e+09, 4.0758600e+09,
       3.8275500e+09, 3.5186200e+09, 3.3706300e+09, 4.0185900e+09,
       3.9793700e+09, 3.8364800e+09, 3.8425900e+09, 3.6839700e+09,
       3.8543200e+09, 4.5179900e+09, 3.9559600e+09, 3.5163800e+09,
       3.5888600e+09, 3.9272400e+09, 3.8944400e+09, 3.8456300e+09,
       3.7143200e+09, 4.3963800e+09, 4.3386400e+09, 4.3507900e+09,
       4.7716600e+09, 4.4045700e+09, 4.6350700e+09, 4.7799800e+09,
       4.7342400e+09, 4.0804200e+09, 3.7069400e+09, 3.8019600e+09,
       4.5735700e+09, 4.8116700e+09, 5.3249000e+09, 4.1863700e+09,
       4.7050500e+09, 4.8256400e+09, 5.2312800e+09, 6.2082600e+09,
       5.0323300e+09, 5.8462900e+09, 5.2760900e+09, 3.2475900e+09,
       5.2654200e+09, 6.0341100e+09, 5.1810000e+09, 5.8404300e+09,
       6.7422000e+09, 5.4348600e+09, 7.3636400e+09, 6.7386304e+09,
       7.3652096e+09, 5.6532800e+09, 4.6306400e+09, 6.1802300e+09,
       6.7058300e+09, 6.1279800e+09, 4.6725600e+09, 4.2829600e+09,
       5.4142400e+09, 5.6313300e+09, 5.3460500e+09, 4.6848700e+09,
       4.5622800e+09, 1.2193100e+09, 4.8734200e+09, 5.3193800e+09,
       4.9668100e+09, 5.0673100e+09, 4.7112900e+09, 4.7876000e+09,
       4.0640000e+09, 4.0418200e+09, 3.8292900e+09, 4.1597600e+09,
       4.5550300e+09, 4.0325900e+09, 3.7410700e+09, 3.4206000e+09,
       3.5875700e+09, 3.4996100e+09, 3.8542800e+09, 3.2881200e+09])
```

In [444]:

```
1 v_mask = dow[:, 4] > 5.5e9
2 v_mask
```

Out[444]:

```
array([False, False, False, False, False, False, False, False, False,
       False, False, False,  True,  True, False,  True, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
        True, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False,  True, False,  True,
       False, False, False,  True, False,  True,  True, False,  True,
        True,  True,  True, False,  True,  True,  True, False, False,
       False,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False])
```

In [453]:

```
1 type(v_mask)
```

Out[453]:

```
numpy.ndarray
```

Question 11-2. How many are there? (Please use print function to give your answer).

Hint: Is there a way to incorporate variables within the print statement?

In [458]:

```
1 (v_mask==True).sum()
```

Out[458]:

```
18
```

In [459]:

```
1 print('The dow volume has been above 5.5 billion on ' + str((v_mask==True).sum()) + ' days this year')
```

```
The dow volume has been above 5.5 billion on 18 days this year.
```

In [457]:

1

The dow volume has been above 5.5 billion on 18 days this year.

Question 11-3. Please find the index of every row (or day) where the volume is greater than 5.5.

Hint: This is another case of boolean masking but in this case, applied to the index.

In [475]:

```
1 high_volume_dates = np.where(v_mask)[0]  
2 high_volume_dates
```

Out[475]:

```
array([ 12,  13,  15,  51,  54, 123, 125, 129, 131, 132, 134, 135, 136,  
       137, 139, 140, 141, 145], dtype=int64)
```

In [78]:

1

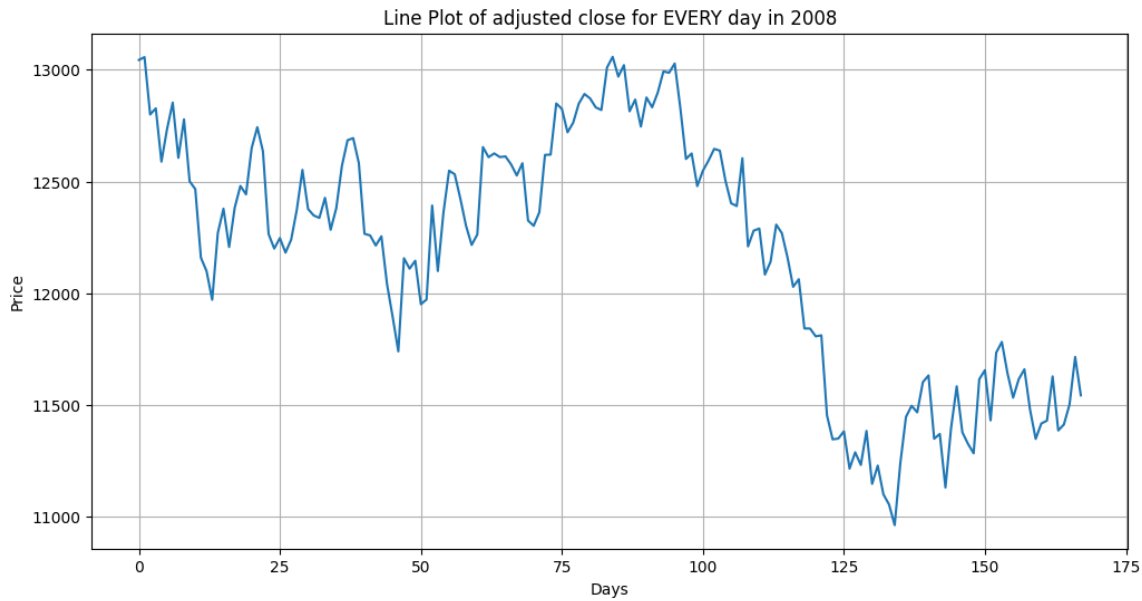
Out[78]:

```
array([ 12,  13,  15,  51,  54, 123, 125, 129, 131, 132, 134, 135, 136,  
       137, 139, 140, 141, 145])
```

Question 11-4. Plot the adjusted close for EVERY day in 2008.

In [474]:

```
1 adj = dow[:, -1]
2 plt.figure(figsize=(12, 6))
3 # Create a line plot
4 plt.plot(adj)
5 plt.xlabel('Days')
6 plt.ylabel('Price')
7 plt.title('Line Plot of adjusted close for EVERY day in 2008')
8 plt.grid(True)
9 plt.show()
10
11
```

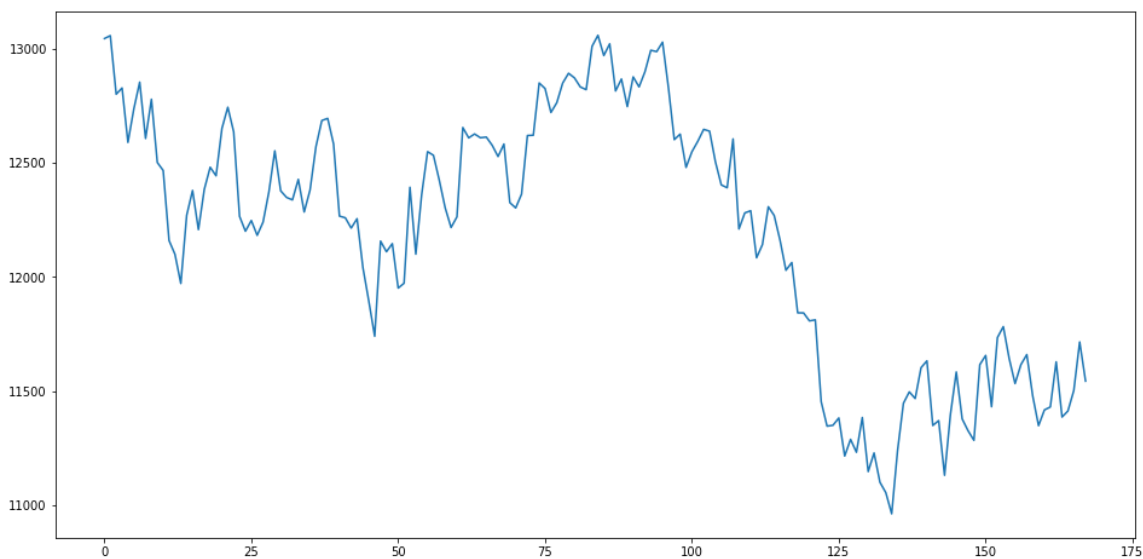


In [79]:

1

Out[79]:

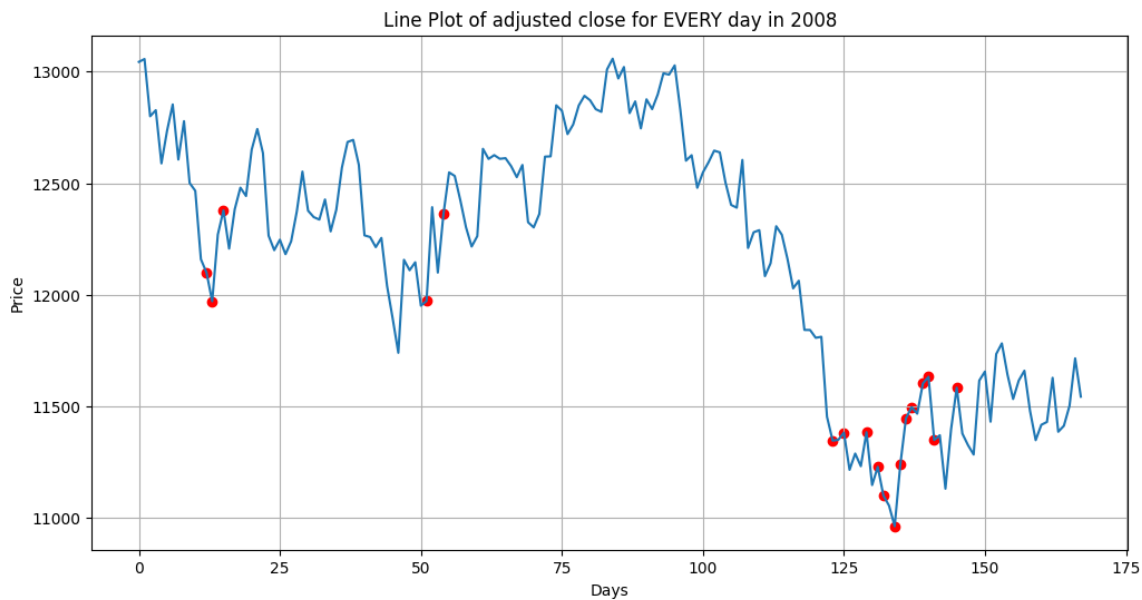
[<matplotlib.lines.Line2D at 0x7fea78a6c250>]



Question 11-5. Now over-plot this plot with a red dot marker for every day where the dow volume was greater than 5.5 billion.

In [478]:

```
1
2
3
4 plt.figure(figsize=(12, 6))
5 # Create a line plot
6 plt.plot(adj)
7 # Overlay red dots for high volume days
8 plt.scatter(high_volume_dates, adj[v_mask], color='red', marker='o', label='High Volume Days')
9
10 plt.xlabel('Days')
11 plt.ylabel('Price')
12 plt.title('Line Plot of adjusted close for EVERY day in 2008')
13 plt.grid(True)
14 plt.show()
15
16
```

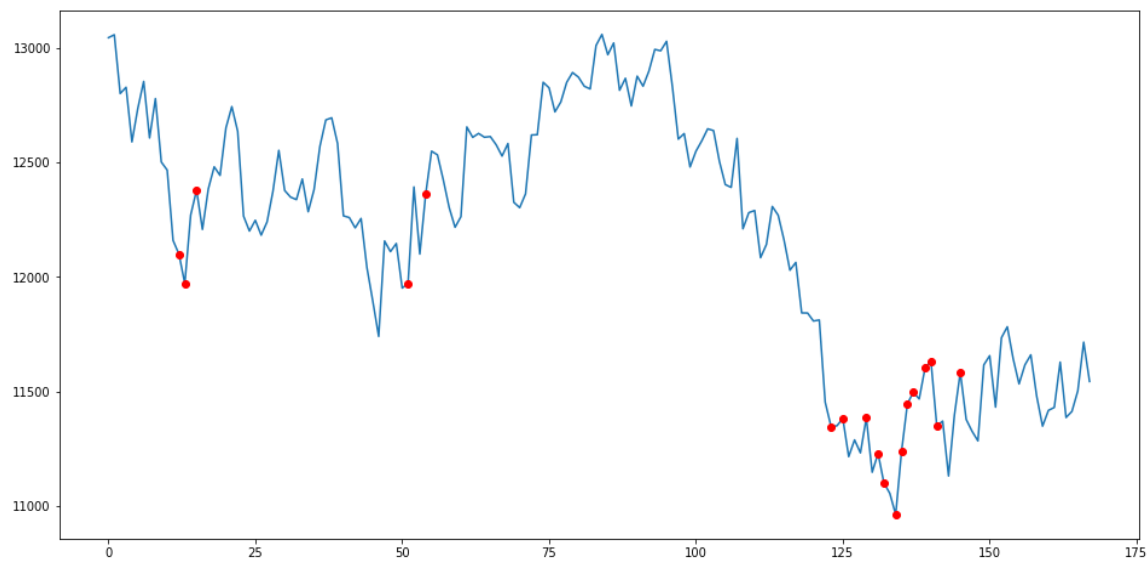


In [80]:

1

Out[80]:

[<matplotlib.lines.Line2D at 0x7fea2829bd60>]



"Thank you for putting your efforts into the exercise problem sets 100"