In [17]:
```
import pandas as pd

earthquake = pd.read_csv('temporary/earthquakes.csv')
faang = pd.read_csv('temporary/faang.csv')
```

In [20]:
```
#clean first
earthquake.fillna(0)
```

Out[20]:

	mag	magType	time	place	tsunami	parsed_place
0	1.35	ml	1539475168010	9km NE of Aguanga, CA	0	California
1	1.29	ml	1539475129610	9km NE of Aguanga, CA	0	California
2	3.42	ml	1539475062610	8km NE of Aguanga, CA	0	California
3	0.44	ml	1539474978070	9km NE of Aguanga, CA	0	California
4	2.16	md	1539474716050	10km NW of Avenal, CA	0	California
...	...	...	...	...	...	...
9327	0.62	md	1537230228060	9km ENE of Mammoth Lakes, CA	0	California
9328	1.00	ml	1537230135130	3km W of Julian, CA	0	California
9329	2.40	md	1537229908180	35km NNE of Hatillo, Puerto Rico	0	Puerto Rico
9330	1.10	ml	1537229545350	9km NE of Aguanga, CA	0	California
9331	0.66	ml	1537228864470	9km NE of Aguanga, CA	0	California

9332 rows × 6 columns

In [21]:
```
#clean
faang.fillna(0)
```

Out[21]:

	ticker	date	open	high	low	close	volume
0	FB	2018-01-02	177.68	181.58	177.5500	181.42	18151903
1	FB	2018-01-03	181.88	184.78	181.3300	184.67	16886563
2	FB	2018-01-04	184.90	186.21	184.0996	184.33	13880896
3	FB	2018-01-05	185.59	186.90	184.9300	186.85	13574535
4	FB	2018-01-08	187.20	188.90	186.3300	188.28	17994726
...	...	...	...	...	...	...	...
1250	GOOG	2018-12-24	973.90	1003.54	970.1100	976.22	1590328
1251	GOOG	2018-12-26	989.01	1040.00	983.0000	1039.46	2373270
1252	GOOG	2018-12-27	1017.15	1043.89	997.0000	1043.88	2109777
1253	GOOG	2018-12-28	1049.62	1055.56	1033.1000	1037.08	1413772
1254	GOOG	2018-12-31	1050.96	1052.70	1023.5900	1035.61	1493722

1255 rows × 7 columns

1. With the earthquakes.csv file, select all the earthquakes in Japan with a magType of mb and a magnitude of 4.9 or greater.

In [16]:
```
#earthquake.query('`magType` == 'mb' and `parsed_place` == 'Japan'`')
#1st method did not work and I did not bother fixing it. So I moved onto the altern

earthquake[(earthquake.magType == 'mb') & (earthquake.parsed_place == 'Japan')]
#finally, this worked.
```

Out[16]:

	mag	magType	time	place	tsunami	parsed_place
67	4.6	mb	1539448501800	160km NNW of Nago, Japan	0	Japan
713	4.7	mb	1539238726290	139km WSW of Naze, Japan	0	Japan
1124	4.6	mb	1539115120470	53km ESE of Kamaishi, Japan	0	Japan
1309	4.5	mb	1539058606580	80km ENE of Misawa, Japan	0	Japan
1398	4.4	mb	1539034912990	65km NNE of Hachijo-jima, Japan	0	Japan
1422	4.7	mb	1539028322670	41km E of Namie, Japan	0	Japan
1435	4.4	mb	1539020712670	21km E of Tomakomai, Japan	0	Japan
1492	4.6	mb	1539003238480	29km E of Tomakomai, Japan	0	Japan
1563	4.9	mb	1538977532250	293km ESE of Iwo Jima, Japan	0	Japan
1732	4.1	mb	1538923896600	89km NE of Miyako, Japan	0	Japan
1875	4.7	mb	1538881270700	177km WSW of Chichi-shima, Japan	0	Japan
1898	4.8	mb	1538874859870	12km N of Shinshiro, Japan	0	Japan
2053	4.8	mb	1538839123780	147km E of Nago, Japan	0	Japan
2191	4.6	mb	1538799265130	27km ESE of Chitose, Japan	0	Japan
2564	4.6	mb	1538701532340	162km E of Nago, Japan	0	Japan
2576	5.4	mb	1538697528010	37km E of Tomakomai, Japan	0	Japan
2688	4.0	mb	1538673414250	51km WNW of Mikuni, Japan	0	Japan
2824	4.4	mb	1538641253520	51km SSE of Kushima, Japan	0	Japan
3072	4.9	mb	1538579732490	15km ENE of Hasaki, Japan	0	Japan
3632	4.9	mb	1538450871260	53km ESE of Hitachi, Japan	0	Japan
3771	4.5	mb	1538422163330	87km ESE of Kamaishi, Japan	0	Japan
3851	4.7	mb	1538406431250	173km E of Nago, Japan	0	Japan

	mag	magType	time	place	tsunami	parsed_place
4082	4.8	mb	1538360525340	31km E of Chitose, Japan	0	Japan
4192	4.3	mb	1538336412660	50km NNE of Shizunai, Japan	0	Japan
4257	4.6	mb	1538323620710	213km SE of Hachijo-jima, Japan	0	Japan
4846	4.5	mb	1538213154900	25km ESE of Chitose, Japan	0	Japan
5655	4.6	mb	1538048754220	156km ENE of Nago, Japan	0	Japan
5915	4.4	mb	1538003928830	78km ESE of Yamada, Japan	0	Japan
6769	4.8	mb	1537846261100	58km SE of Ofunato, Japan	0	Japan
6961	4.7	mb	1537799121060	46km ENE of Hirara, Japan	0	Japan
7001	4.3	mb	1537792438760	25km ESE of Muroran, Japan	0	Japan
7066	4.4	mb	1537780626560	35km E of Tomakomai, Japan	0	Japan
7145	4.1	mb	1537764836900	97km N of Mombetsu, Japan	0	Japan
7261	4.1	mb	1537738421220	64km ESE of Owase, Japan	0	Japan
7381	4.6	mb	1537715134540	9km ENE of Funaishikawa, Japan	0	Japan
7408	4.8	mb	1537709390500	65km SSE of Hasaki, Japan	0	Japan
7652	4.2	mb	1537654508260	164km E of Nago, Japan	0	Japan
7696	4.0	mb	1537640754840	65km E of Shizunai, Japan	0	Japan
7807	4.4	mb	1537619327220	67km NNE of Hachijo-jima, Japan	0	Japan
8071	4.4	mb	1537550307820	67km E of Nemuro, Japan	0	Japan
8311	4.4	mb	1537487811890	166km E of Nago, Japan	0	Japan
8324	4.6	mb	1537484762840	167km E of Nago, Japan	0	Japan
8327	4.2	mb	1537484169100	34km ESE of Chitose, Japan	0	Japan
8376	4.4	mb	1537470382450	171km E of Nago, Japan	0	Japan
8431	4.6	mb	1537455447970	171km E of Nago, Japan	0	Japan
8550	4.4	mb	1537430889800	105km ENE of Miyako, Japan	0	Japan

	mag	magType	time	place	tsunami	parsed_place
8555	4.6	mb	1537429466000	105km ENE of Miyako, Japan	0	Japan
8687	4.2	mb	1537392267190	39km NE of Misawa, Japan	0	Japan
9070	4.7	mb	1537289668660	96km E of Hasaki, Japan	0	Japan
9198	4.6	mb	1537258271290	3km ESE of Sugito, Japan	0	Japan

2. Create bins for each full number of magnitude (for example, the first bin is 0-1, the second is 1-2, and so on) with a magType of ml and count how many are in each bin.

3. Using the faang.csv file, group by the ticker and resample to monthly frequency. Make the following aggregations:

- Mean of the opening price
- Maximum of the high price
- Minimum of the low price
- Mean of the closing price
- Sum of the volume traded

In [29]:
```
faang.head(3)
```

Out[29]:

	ticker	date	open	high	low	close	volume
0	FB	2018-01-02	177.68	181.58	177.5500	181.42	18151903
1	FB	2018-01-03	181.88	184.78	181.3300	184.67	16886563
2	FB	2018-01-04	184.90	186.21	184.0996	184.33	13880896

In [27]:
```
#check the ticker's unique values
faang.ticker.unique()
```

Out[27]:
```
array(['FB', 'AAPL', 'AMZN', 'NFLX', 'GOOG'], dtype=object)
```

In [32]:
```
#mean
grouped_fb = faang[faang['ticker'] == 'FB']
fb_mean = grouped_fb[open].mean()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[32], line 3
      1 #mean
      2 grouped_fb = faang[faang['ticker'] == 'FB']
----> 3 fb_mean = grouped_fb[open].mean()

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:4065, in DataFrame.__getitem
__(self, key)
   4063 check_dict_or_set_indexers(key)
   4064 key = lib.item_from_zerodim(key)
-> 4065 key = com.apply_if_callable(key, self)
   4067 if is_hashable(key) and not is_iterator(key):
   4068     # is_iterator to exclude generator e.g. test_getitem_listlike
   4069     # shortcut if the key is in columns
   4070     is_mi = isinstance(self.columns, MultiIndex)

File ~\anaconda3\Lib\site-packages\pandas\core\common.py:384, in apply_if_callable(m
aybe_callable, obj, **kwargs)
    373 """
    374 Evaluate possibly callable input using obj and kwargs if it is callable,
    375 otherwise return as it is.
   (...)
    381 **kwargs
    382 """
    383 if callable(maybe_callable):
--> 384     return maybe_callable(obj, **kwargs)
    386 return maybe_callable

File ~\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py:317, in _modifie
d_open(file, *args, **kwargs)
    315 @functools.wraps(io_open)
    316 def _modified_open(file, *args, **kwargs):
--> 317     if file in {0, 1, 2}:
    318         raise ValueError(
    319             f"IPython won't let you open fd={file} by default "
    320             "as it is likely to crash IPython. If you know what you are doin
g, "
    321             "you can use builtins' open."
    322         )
    324     return io_open(file, *args, **kwargs)

TypeError: unhashable type: 'DataFrame'
```

In [ ]: