Hands-on Activity 8.1: Aggregating Data with Pandas

8.1.1 Intended Learning Outcomes

After this activity, the student should be able to:

- Demonstrate querying and merging of dataframes
- Perform advanced calculations on dataframes
- Aggregate dataframes with pandas and numpy
- Work with time series data

8.1.2 Resources

- Computing Environment using Python 3.x
- Attached Datasets (under Instructional Materials)

8.1.3 Procedures

The procedures can be found in the canvas module. Check the following under topics:

- 8.1 Weather Data Collection - GitHub Link: https://github.com/de-fernandez/CPE-311-CPE22S3/tree/1149b835ead333cb7f8654c39e9cd269c62391a1/Don%20Eleazar%20T.%20Fe%20Aggregating%20Pandas%20DataFrames)/8.1%20Weather%20Data%20Collection
- 8.2 Querying and Merging - GitHub Link: https://github.com/de-fernandez/CPE-311-CPE22S3/tree/1149b835ead333cb7f8654c39e9cd269c62391a1/Don%20Eleazar%20T.%20Fe%20Aggregating%20Pandas%20DataFrames)/8.2%20Querying%20and%20Merging
- 8.3 Dataframe Operations - GitHub Link: https://github.com/de-fernandez/CPE-311-CPE22S3/tree/1149b835ead333cb7f8654c39e9cd269c62391a1/Don%20Eleazar%20T.%20Fe%20Aggregating%20Pandas%20DataFrames)/8.3%20Dataframe%20Operations
- 8.4 Aggregations - GitHub Link: https://github.com/de-fernandez/CPE-311-CPE22S3/tree/1149b835ead333cb7f8654c39e9cd269c62391a1/Don%20Eleazar%20T.%20Fe%20Aggregating%20Pandas%20DataFrames)/8.4%20Aggregations
- 8.5 Time Series - GitHub Link: https://github.com/de-fernandez/CPE-311-CPE22S3/tree/1149b835ead333cb7f8654c39e9cd269c62391a1/Don%20Eleazar%20T.%20Fe%20Aggregating%20Pandas%20DataFrames)/8.5%20Time%20Series

8.1.4 Data Analysis

Provide some comments here about the results of the procedures.

- In SQLite, I learned how to save my data in a database with it.
- In handling time based data, I learned how to filter out specific times or ranges, which helped me work with data that's recorded by the hour instead of just by day.
- During the activity, I ran into some problems where the code did not work with the latest version of Pandas. Some functions were outdated.

8.1.5 Supplementary Activity

Using the CSV files provided and what we have learned so far in this module complete the following exercises:

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.
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

4. Build a crosstab with the earthquake data between the tsunami column and the magType column. Rather than showing the frequency count, show the maximum magnitude that was observed for each combination. Put the magType along the columns.
5. Calculate the rolling 60-day aggregations of OHLC data by ticker for the FAANG data. Use the same aggregations as exercise no. 3.
6. Create a pivot table of the FAANG data that compares the stocks. Put the ticker in the rows and show the averages of the OHLC and volume traded data.
7. Calculate the Z-scores for each numeric column of Netflix's data (ticker is NFLX) using apply().
8. Add event descriptions:

- Create a dataframe with the following three columns: ticker, date, and event. The columns should have the following values:
  - ticker: 'FB'
  - date: ['2018-07-25', '2018-03-19', '2018-03-20']
  - event: ['Disappointing user growth announced after close.', 'Cambridge Analytica story', 'FTC investigation']
- Set the index to ['date', 'ticker']
- Merge this data with the FAANG data using an outer join

9. Use the transform() method on the FAANG data to represent all the values in terms of the first date in the data. To do so, divide all the values for each ticker by the values for the first date in the data for that ticker. This is referred to as an index, and the data for the first date is the base (https://ec.europa.eu/eurostat/statistics-explained/index.php/Beginners:Statisticalconcept-Indexandbaseyear). When data is in

this format, we can easily see growth over time. Hint: transform() can take a function name

In [7]:
```
import pandas as pd
```

In [8]:
```
earthquake = pd.read_csv("earthquakes.csv")
earthquake = pd.DataFrame(earthquake)
earthquake.head()
```

Out[8]:

	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

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 [10]:
```
earthquake["parsed_place"].unique()
```

Out[10]:
```
array(['California', 'Dominican Republic', 'Alaska', 'Indonesia',
       'Canada', 'Puerto Rico', 'Montana', 'Nevada', 'Christmas Island',
       'Hawaii', 'Northern Mariana Islands', 'Japan', 'Ecuador',
       'Vanuatu', 'Mexico', 'Russia', 'British Virgin Islands',
       'Washington', 'Papua New Guinea', 'Fiji', 'U.S. Virgin Islands',
       'Chile', 'Peru', 'Yemen', 'Guatemala', 'Kansas', 'Australia',
       'Wyoming', 'Kuril Islands', 'Oklahoma', 'Tennessee',
       'Pacific-Antarctic Ridge', 'Utah', 'Colombia', 'Argentina',
       'Oregon', 'Greece', 'Missouri', 'Tajikistan',
       'Northern Mid-Atlantic Ridge', 'Sumatra', 'Solomon Islands',
       'Burma', 'Taiwan', 'Nicaragua',
       'South Georgia and South Sandwich Islands', 'Idaho', 'Kyrgyzstan',
       'Arizona', 'Tonga', 'Northern East Pacific Rise', 'South Africa',
       'Southern Mid-Atlantic Ridge', 'Costa Rica', 'China',
       'Philippines', 'Haiti', 'Jamaica', 'Kentucky', 'New Zealand',
       'Iran', 'Afghanistan', 'Southwest Indian Ridge', 'Saint Helena',
       'Texas', 'New Caledonia', 'El Salvador',
       'Central Mid-Atlantic Ridge', 'Western Xizang', 'Italy',
       'Mid-Indian Ridge', 'Ascension Island', 'North Carolina',
       'South Sandwich Islands', 'Saint Eustatius and Saba', 'Pakistan',
       'Bolivia', 'Turkey', 'Indian Ocean Triple Junction', 'Mayotte',
       'Queen Charlotte Islands', 'India', 'Arkansas', 'Guam',
       'Uzbekistan', 'Prince Edward Islands', 'Martinique', 'Honduras',
       'Southern East Pacific Rise', 'East Timor', 'Barbuda', 'Mauritius',
       'Carlsberg Ridge', 'Greenland', 'Balleny Islands',
       'Western Indian-Antarctic Ridge', 'South Carolina', 'Vermont',
       'Romania', 'New Hampshire', 'Central East Pacific Rise',
       'Southeast Indian Ridge', 'Kermadec Islands', 'Colorado',
       'Illinois', 'Socotra', 'Azerbaijan', 'Iraq', 'Somalia',
       'New Mexico'], dtype=object)
```

In [11]:
```
# To select all the earthquakes in Japan with a magType of mb and a magnitude of 4.
earthquake_1 = earthquake[(earthquake["magType"] == "mb") & (earthquake["parsed_pla
earthquake_1
```

Out[11]:

	mag	magType	time	place	tsunami	parsed_place
1563	4.9	mb	1538977532250	293km ESE of Iwo Jima, Japan	0	Japan
2576	5.4	mb	1538697528010	37km E of Tomakomai, 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

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.

In [13]:
```
# To create bins with ".cut()"
earthquake_2 = pd.cut(earthquake[(earthquake["magType"] == "ml")]["mag"], bins = [0
                      labels = ["0 to 1", "1 to 2", "2 to 3", "3 to 4", "4 to 5", "
```

```
earthquake_2 = pd.DataFrame(earthquake_2.value_counts().sort_index())
earthquake_2
```

Out[13]:

mag	count
0 to 1	2207
1 to 2	3105
2 to 3	862
3 to 4	122
4 to 5	2
5 to 6	1
6 to 7	0
7 to 8	0

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 [15]:
```
faang = pd.read_csv("faang.csv")
faang = pd.DataFrame(faang)
faang.head()
```

Out[15]:

	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

In [16]:
```
faang["date"] = faang["date"].apply(pd.to_datetime)
faang.dtypes
```

Out[16]:
```
ticker            object
date      datetime64[ns]
open             float64
high             float64
low              float64
close            float64
volume             int64
dtype: object
```

In [17]:
```
faang = faang.set_index("date")
```

In [18]:
```
# Group by the ticker and resample to monthly frequency
faang_3 = faang.groupby("ticker").resample("ME").mean(numeric_only = True)
faang_3
```

Out[18]:

ticker	date	open	high	low	close	volume
AAPL	2018-01-31	170.714690	171.874614	169.534767	170.699271	3.141331e+07
	2018-02-28	164.562753	167.131595	162.797147	164.921884	4.883655e+07
	2018-03-31	172.421381	174.147476	170.465676	171.878919	3.398702e+07
	2018-04-30	167.332895	169.082638	165.767929	167.286924	3.173144e+07
	2018-05-31	182.635582	184.128332	181.555282	183.207418	2.822619e+07
	2018-06-30	186.605843	187.705624	185.319514	186.508652	2.512497e+07
	2018-07-31	188.065786	189.394719	186.897362	188.179724	1.875447e+07
	2018-08-31	210.460287	212.682830	209.455974	211.477743	3.044865e+07
	2018-09-30	220.611742	222.794142	218.376105	220.356353	3.573537e+07
	2018-10-31	219.489426	222.209104	216.189439	219.137822	3.433687e+07
	2018-11-30	190.828681	192.946948	187.725357	190.246652	4.577724e+07
	2018-12-31	164.537405	166.605374	161.199621	163.564732	4.731142e+07
AMZN	2018-01-31	1301.377143	1317.424286	1291.311429	1309.010952	4.589109e+06
	2018-02-28	1447.112632	1469.081053	1417.413158	1442.363158	7.251791e+06
	2018-03-31	1542.160476	1556.584762	1517.223333	1540.367619	6.209531e+06
	2018-04-30	1475.841905	1492.336667	1448.130952	1468.220476	6.187893e+06
	2018-05-31	1590.474545	1601.441818	1582.605000	1594.903636	3.255241e+06
	2018-06-30	1699.088571	1711.796667	1684.512381	1698.823810	4.092453e+06
	2018-07-31	1786.305714	1801.520476	1765.038095	1784.649048	4.649039e+06
	2018-08-31	1891.957826	1908.748696	1880.893043	1897.851304	4.198942e+06
	2018-09-30	1969.239474	1987.199474	1943.608947	1966.077895	4.970826e+06
	2018-10-31	1799.630870	1823.199565	1753.272609	1782.058261	7.966459e+06
	2018-11-30	1622.323810	1653.718095	1593.101905	1625.483810	6.632867e+06
	2018-12-31	1572.922105	1601.891053	1527.985263	1559.443158	8.148016e+06
FB	2018-01-31	184.364762	186.210952	182.924757	184.962857	2.360265e+07
	2018-02-28	180.721579	183.131405	177.756642	180.269474	2.719063e+07
	2018-03-31	173.449524	176.168929	170.508095	173.489524	4.743964e+07
	2018-04-30	164.163557	166.086600	161.546238	163.810476	3.576811e+07
	2018-05-31	181.910509	183.873145	180.950032	182.930000	1.823383e+07

ticker	date	open	high	low	close	volume
	2018-06-30	194.974067	196.797348	193.179524	195.267619	1.844123e+07
	2018-07-31	199.332143	201.598095	197.504767	199.967143	3.108396e+07
	2018-08-31	177.598443	179.433913	175.680935	177.491957	2.387030e+07
	2018-09-30	164.232895	166.399253	162.416726	164.377368	2.634047e+07
	2018-10-31	154.873261	157.124783	152.103117	154.187826	2.706288e+07
	2018-11-30	141.762857	143.657986	139.593095	141.635714	2.467383e+07
	2018-12-31	137.529474	140.493679	134.814037	137.161053	2.940980e+07
GOOG	**2018-01-31**	1127.200952	1135.750000	1120.723333	1130.770476	1.368499e+06
	2018-02-28	1088.629474	1103.917895	1074.523684	1088.206842	2.230742e+06
	2018-03-31	1096.108095	1108.976190	1076.997143	1091.490476	2.163336e+06
	2018-04-30	1038.415238	1049.753333	1024.404762	1035.696190	1.989204e+06
	2018-05-31	1064.021364	1075.978636	1058.456364	1069.275909	1.447691e+06
	2018-06-30	1136.396190	1146.510952	1127.869524	1137.626667	1.528745e+06
	2018-07-31	1183.464286	1196.692857	1173.977143	1187.590476	1.521590e+06
	2018-08-31	1226.156957	1235.015652	1217.231739	1225.671739	1.253060e+06
	2018-09-30	1176.878421	1185.772105	1164.868947	1175.808947	1.519116e+06
	2018-10-31	1116.082174	1132.273043	1095.551304	1110.940435	2.108529e+06
	2018-11-30	1054.971429	1068.019524	1041.509048	1056.162381	1.749313e+06
	2018-12-31	1042.620000	1059.323684	1023.567895	1037.420526	2.118761e+06
NFLX	**2018-01-31**	231.269286	235.313048	228.194838	232.908095	1.135131e+07
	2018-02-28	270.873158	276.718337	266.113947	271.443684	9.715043e+06
	2018-03-31	312.712857	317.488976	305.397095	312.228095	1.254521e+07
	2018-04-30	309.129529	313.808048	302.023662	307.466190	1.247926e+07
	2018-05-31	329.779759	333.969100	327.000664	331.536818	6.456869e+06
	2018-06-30	384.557595	389.790148	378.141595	384.133333	1.162057e+07
	2018-07-31	380.969090	387.178738	374.233038	381.515238	1.454702e+07
	2018-08-31	345.409591	351.161248	340.844835	346.257826	9.267134e+06
	2018-09-30	363.326842	369.201579	356.710111	362.641579	8.991166e+06
	2018-10-31	340.025348	346.519809	328.360230	335.445652	1.580826e+07

		open	high	low	close	volume
ticker	date					
	2018-11-30	290.643333	297.460471	282.389914	290.344762	1.224412e+07
	2018-12-31	266.309474	273.846289	258.559705	265.302368	1.233182e+07

In [19]:
```
faang_3 = faang_3.agg({
    "open": "mean", # Mean of the opening price
    "high": "max", # Maximum of the high price
    "low": "min", # Minimum of the low price
    "close": "mean", # Mean of the closing price
    "volume": "sum" # Sum of the volume traded
})
faang_3
```

Out[19]:
```
open      6.861261e+02
high      1.987199e+03
low       1.348140e+02
close     6.855026e+02
volume    9.713094e+08
dtype: float64
```

4. Build a crosstab with the earthquake data between the tsunami column and the magType column. Rather than showing the frequency count, show the maximum magnitude that was observed for each combination. Put the magType along the columns.

In [21]:
```
earthquake_4 = pd.crosstab(earthquake['tsunami'],earthquake['magType'], values = ea
earthquake_4
```

Out[21]:

magType	mb	mb_lg	md	mh	ml	ms_20	mw	mwb	mwr	mww
tsunami										
0	5.6	3.5	4.11	1.1	4.2	NaN	3.83	5.8	4.8	6.0
1	6.1	NaN	NaN	NaN	5.1	5.7	4.41	NaN	NaN	7.5

5. Calculate the rolling 60-day aggregations of OHLC data by ticker for the FAANG data. Use the same aggregations as exercise no. 3.

In [23]:
```
faang_5 = faang.groupby("ticker").rolling("60D").agg({
    "open": "mean", # Mean of the opening price
    "high": "max", # Maximum of the high price
    "low": "min", # Minimum of the low price
    "close": "mean", # Mean of the closing price
    "volume": "sum" # Sum of the volume traded
})
faang_5
```

Out[23]:

ticker	date	open	high	low	close	volume
AAPL	**2018-01-02**	166.927100	169.0264	166.0442	168.987200	25555934.0
	2018-01-03	168.089600	171.2337	166.0442	168.972500	55073833.0
	2018-01-04	168.480367	171.2337	166.0442	169.229200	77508430.0
	2018-01-05	168.896475	172.0381	166.0442	169.840675	101168448.0
	2018-01-08	169.324680	172.2736	166.0442	170.080040	121736214.0
...	**...**	...	...	...	...	...
NFLX	**2018-12-24**	283.509250	332.0499	233.6800	281.931750	525657894.0
	2018-12-26	281.844500	332.0499	231.2300	280.777750	520444588.0
	2018-12-27	281.070488	332.0499	231.2300	280.162805	532679805.0
	2018-12-28	279.916341	332.0499	231.2300	279.461341	521968250.0
	2018-12-31	278.430769	332.0499	231.2300	277.451410	476309676.0

1255 rows × 5 columns

6. Create a pivot table of the FAANG data that compares the stocks. Put the ticker in the rows and show the averages of the OHLC and volume traded data.

In [25]:
```
faang_6 = faang.pivot_table(index = "ticker", aggfunc = "mean")
faang_6
```

Out[25]:

ticker	close	high	low	open	volume
AAPL	186.986218	188.906858	185.135729	187.038674	3.402145e+07
AMZN	1641.726175	1662.839801	1619.840398	1644.072669	5.649563e+06
FB	171.510936	173.615298	169.303110	171.454424	2.768798e+07
GOOG	1113.225139	1125.777649	1101.001594	1113.554104	1.742645e+06
NFLX	319.290299	325.224583	313.187273	319.620533	1.147030e+07

7. Calculate the Z-scores for each numeric column of Netflix's data (ticker is NFLX) using apply().

In [27]:
```
faang_7 = faang[(faang["ticker"] == "NFLX")]
```

In [28]:
```
# Calculate the z score manually through coding
def z_score(load):
    return (load - load.mean()) / load.std()

faang_7 = faang_7.select_dtypes(include = "number").apply(z_score)
faang_7
```

Out[28]:

date	open	high	low	close	volume
2018-01-02	-2.500753	-2.516023	-2.410226	-2.416644	-0.088760
2018-01-03	-2.380291	-2.423180	-2.285793	-2.335286	-0.507606
2018-01-04	-2.296272	-2.406077	-2.234616	-2.323429	-0.959287
2018-01-05	-2.275014	-2.345607	-2.202087	-2.234303	-0.782331
2018-01-08	-2.218934	-2.295113	-2.143759	-2.192192	-1.038531
...	...	...	...	...	...
2018-12-24	-1.571478	-1.518366	-1.627197	-1.745946	-0.339003
2018-12-26	-1.735063	-1.439978	-1.677339	-1.341402	0.517040
2018-12-27	-1.407286	-1.417785	-1.495805	-1.302664	0.134868
2018-12-28	-1.248762	-1.289018	-1.297285	-1.292137	-0.085164
2018-12-31	-1.203817	-1.122354	-1.088531	-1.055420	0.359444

251 rows × 5 columns

8. Add event descriptions:

- Create a dataframe with the following three columns: ticker, date, and event. The columns should have the following values:
  - ticker: 'FB'
  - date: ['2018-07-25', '2018-03-19', '2018-03-20']
  - event: ['Disappointing user growth announced after close.', 'Cambridge Analytica story', 'FTC investigation']
- Set the index to ['date', 'ticker']
- Merge this data with the FAANG data using an outer join

In [30]:
```
# Create a dataframe with the following three columns: ticker, date, and event. The
# ticker: 'FB'
# date: ['2018-07-25', '2018-03-19', '2018-03-20']
# event: ['Disappointing user growth announced after close.', 'Cambridge Analytica
faang_fb = pd.DataFrame({
    "ticker": ["FB", "FB", "FB"],
    "date": ["2018-07-25", "2018-03-19", "2018-03-20"],
```

```
    "event": ["Disappointing user growth announced after close.",
              "Cambridge Analytica story",
              "FTC investigation"]
})
faang_fb["date"] = pd.to_datetime(faang_fb["date"])
```

In [31]:
```
# Merge with FAANG data using outer join
faang_fb = pd.merge(faang, faang_fb, on = ["date", "ticker"], how = "outer")
faang_fb
```

Out[31]:

	date	ticker	open	high	low	close	volume	event
0	2018-01-02	AAPL	166.9271	169.0264	166.0442	168.9872	25555934	NaN
1	2018-01-02	AMZN	1172.0000	1190.0000	1170.5100	1189.0100	2694494	NaN
2	2018-01-02	FB	177.6800	181.5800	177.5500	181.4200	18151903	NaN
3	2018-01-02	GOOG	1048.3400	1066.9400	1045.2300	1065.0000	1237564	NaN
4	2018-01-02	NFLX	196.1000	201.6500	195.4200	201.0700	10966889	NaN
...	...	...	...	...	...	...	...	...
1250	2018-12-31	AAPL	157.8529	158.6794	155.8117	157.0663	35003466	NaN
1251	2018-12-31	AMZN	1510.8000	1520.7600	1487.0000	1501.9700	6954507	NaN
1252	2018-12-31	FB	134.4500	134.6400	129.9500	131.0900	24625308	NaN
1253	2018-12-31	GOOG	1050.9600	1052.7000	1023.5900	1035.6100	1493722	NaN
1254	2018-12-31	NFLX	260.1600	270.1001	260.0000	267.6600	13508920	NaN

1255 rows × 8 columns

In [32]:
```
faang.tail()
```

Out[32]:

date	ticker	open	high	low	close	volume
2018-12-24	GOOG	973.90	1003.54	970.11	976.22	1590328
2018-12-26	GOOG	989.01	1040.00	983.00	1039.46	2373270
2018-12-27	GOOG	1017.15	1043.89	997.00	1043.88	2109777
2018-12-28	GOOG	1049.62	1055.56	1033.10	1037.08	1413772
2018-12-31	GOOG	1050.96	1052.70	1023.59	1035.61	1493722

9. Use the transform() method on the FAANG data to represent all the values in terms of the first date in the data. To do so, divide all the values for each ticker by the values for the first date in the data for that ticker. This is referred to as an index, and the data for

the first date is the base (https://ec.europa.eu/eurostat/statistics-explained/index.php/Beginners:Statisticalconcept-Indexandbaseyear). When data is in this format, we can easily see growth over time. Hint: transform() can take a function name

In [34]:
```
faang = faang.groupby('ticker').transform(lambda x: x / x.iloc[0])
faang
```

Out[34]:

date	open	high	low	close	volume
2018-01-02	1.000000	1.000000	1.000000	1.000000	1.000000
2018-01-03	1.023638	1.017623	1.021290	1.017914	0.930292
2018-01-04	1.040635	1.025498	1.036889	1.016040	0.764707
2018-01-05	1.044518	1.029298	1.041566	1.029931	0.747830
2018-01-08	1.053579	1.040313	1.049451	1.037813	0.991341
...	...	...	...	...	...
2018-12-24	0.928993	0.940578	0.928131	0.916638	1.285047
2018-12-26	0.943406	0.974750	0.940463	0.976019	1.917695
2018-12-27	0.970248	0.978396	0.953857	0.980169	1.704782
2018-12-28	1.001221	0.989334	0.988395	0.973784	1.142383
2018-12-31	1.002499	0.986653	0.979296	0.972404	1.206986

1255 rows × 5 columns

In [ ]:
```
```