In [1]:
```
#주제
# 구글과 아마존의 15~17년도 주가 분석
```

In [2]:
```
# 1. 구글과 아마존 주가 데이터 불러오기
# 2. Date 열을 추후 시계열 분석을 위해 성질을 Datetime으로 변경하기
# 3. 각 데이터 pd.concat()로 합치기
# 4. 엑셀로 파일 전환
```

In [3]:
```
import pandas as pd
```

In [4]:
```
Google = pd.read_csv("Downloads/Google.csv")
```

In [5]:
```
Google
```

Out[5]:

	Date	Open	High	Low	Close	Volume	Name
0	2006-01-03	211.47	218.05	209.32	217.83	13137450	GOOGL
1	2006-01-04	222.17	224.70	220.09	222.84	15292353	GOOGL
2	2006-01-05	223.22	226.00	220.97	225.85	10815661	GOOGL
3	2006-01-06	228.66	235.49	226.85	233.06	17759521	GOOGL
4	2006-01-09	233.44	236.94	230.70	233.68	12795837	GOOGL
...	...	...	...	...	...	...	...
3014	2017-12-22	1070.00	1071.72	1067.64	1068.86	889446	GOOGL
3015	2017-12-26	1068.64	1068.86	1058.64	1065.85	918767	GOOGL
3016	2017-12-27	1066.60	1068.27	1058.38	1060.20	1116203	GOOGL
3017	2017-12-28	1062.25	1064.84	1053.38	1055.95	994249	GOOGL
3018	2017-12-29	1055.49	1058.05	1052.70	1053.40	1180340	GOOGL

3019 rows × 7 columns

In [6]:
```
Amazon = pd.read_csv("Downloads/Amazon.csv")
```

In [7]:
```
Amazon
```

Out[7]:

	Date	Open	High	Low	Close	Volume	Name
0	2006-01-03	47.47	47.85	46.25	47.58	7582127	AMZN
1	2006-01-04	47.48	47.73	46.69	47.25	7440914	AMZN
2	2006-01-05	47.16	48.20	47.11	47.65	5417258	AMZN
3	2006-01-06	47.97	48.58	47.32	47.87	6154285	AMZN
4	2006-01-09	46.55	47.10	46.40	47.08	8945056	AMZN
...	...	...	...	...	...	...	...
3014	2017-12-22	1172.08	1174.62	1167.83	1168.36	1585054	AMZN
3015	2017-12-26	1168.36	1178.32	1160.55	1176.76	2005187	AMZN
3016	2017-12-27	1179.91	1187.29	1175.61	1182.26	1867208	AMZN
3017	2017-12-28	1189.00	1190.10	1184.38	1186.10	1841676	AMZN
3018	2017-12-29	1182.35	1184.00	1167.50	1169.47	2688391	AMZN

3019 rows × 7 columns

In [8]:
```
Google["Date"] = pd.to_datetime(Google["Date"])
```

In [9]:
```
Amazon["Date"] = pd.to_datetime(Amazon["Date"])
```

In [10]:
```
stocks = pd.concat([Google,Amazon],axis=0)
```

In [11]:
```
stocks
```

Out[11]:

	Date	Open	High	Low	Close	Volume	Name
0	2006-01-03	211.47	218.05	209.32	217.83	13137450	GOOGL
1	2006-01-04	222.17	224.70	220.09	222.84	15292353	GOOGL
2	2006-01-05	223.22	226.00	220.97	225.85	10815661	GOOGL
3	2006-01-06	228.66	235.49	226.85	233.06	17759521	GOOGL
4	2006-01-09	233.44	236.94	230.70	233.68	12795837	GOOGL
...	...	...	...	...	...	...	...
3014	2017-12-22	1172.08	1174.62	1167.83	1168.36	1585054	AMZN
3015	2017-12-26	1168.36	1178.32	1160.55	1176.76	2005187	AMZN
3016	2017-12-27	1179.91	1187.29	1175.61	1182.26	1867208	AMZN
3017	2017-12-28	1189.00	1190.10	1184.38	1186.10	1841676	AMZN
3018	2017-12-29	1182.35	1184.00	1167.50	1169.47	2688391	AMZN

6038 rows × 7 columns

In [12]:
```
stocks["Date"] = stocks.rename(columns={"Date":"Datetime"})
```

In [13]:
```
stocks
```

Out[13]:

	Date	Open	High	Low	Close	Volume	Name
0	2006-01-03 00:00:00	211.47	218.05	209.32	217.83	13137450	GOOGL
1	2006-01-04 00:00:00	222.17	224.70	220.09	222.84	15292353	GOOGL
2	2006-01-05 00:00:00	223.22	226.00	220.97	225.85	10815661	GOOGL
3	2006-01-06 00:00:00	228.66	235.49	226.85	233.06	17759521	GOOGL
4	2006-01-09 00:00:00	233.44	236.94	230.70	233.68	12795837	GOOGL
...	...	...	...	...	...	...	...
3014	2017-12-22 00:00:00	1172.08	1174.62	1167.83	1168.36	1585054	AMZN
3015	2017-12-26 00:00:00	1168.36	1178.32	1160.55	1176.76	2005187	AMZN
3016	2017-12-27 00:00:00	1179.91	1187.29	1175.61	1182.26	1867208	AMZN
3017	2017-12-28 00:00:00	1189.00	1190.10	1184.38	1186.10	1841676	AMZN
3018	2017-12-29 00:00:00	1182.35	1184.00	1167.50	1169.47	2688391	AMZN

6038 rows × 7 columns

In [14]:
```
stocks.to_excel("stocks.xlsx",sheet_name="double stock comparison",index=False)
```

In [15]:
```
stocks.shape
```

Out[15]:
```
(6038, 7)
```

In [16]:
```
stocks.describe()
```

Out[16]:

	Open	High	Low	Close	Volume
count	6038.000000	6038.00000	6038.000000	6038.000000	6.038000e+03
mean	363.768056	367.10339	360.083985	363.710116	4.741608e+06
std	267.017752	268.51303	265.156497	266.930561	4.375820e+06
min	26.090000	26.30000	25.760000	26.070000	5.211410e+05
25%	184.212500	186.50750	181.812500	184.435000	2.251445e+06
50%	281.420000	284.51500	278.680000	281.515000	3.605167e+06
75%	538.260000	542.72750	532.587500	537.940000	5.944314e+06
max	1204.880000	1213.41000	1191.150000	1195.830000	1.044046e+08

In [17]:
```
stocks.head(10)
```

Out[17]:

	Date	Open	High	Low	Close	Volume	Name
0	2006-01-03 00:00:00	211.47	218.05	209.32	217.83	13137450	GOOGL
1	2006-01-04 00:00:00	222.17	224.70	220.09	222.84	15292353	GOOGL
2	2006-01-05 00:00:00	223.22	226.00	220.97	225.85	10815661	GOOGL
3	2006-01-06 00:00:00	228.66	235.49	226.85	233.06	17759521	GOOGL
4	2006-01-09 00:00:00	233.44	236.94	230.70	233.68	12795837	GOOGL
5	2006-01-10 00:00:00	232.44	235.36	231.25	235.11	9104719	GOOGL
6	2006-01-11 00:00:00	235.87	237.79	234.82	236.05	9008664	GOOGL
7	2006-01-12 00:00:00	237.10	237.73	230.98	232.05	10125212	GOOGL
8	2006-01-13 00:00:00	232.39	233.68	231.04	233.36	7660220	GOOGL
9	2006-01-17 00:00:00	231.76	235.18	231.50	233.79	8335300	GOOGL

In [18]:
```
stocks.tail()
```

Out[18]:

	Date	Open	High	Low	Close	Volume	Name
3014	2017-12-22 00:00:00	1172.08	1174.62	1167.83	1168.36	1585054	AMZN
3015	2017-12-26 00:00:00	1168.36	1178.32	1160.55	1176.76	2005187	AMZN
3016	2017-12-27 00:00:00	1179.91	1187.29	1175.61	1182.26	1867208	AMZN
3017	2017-12-28 00:00:00	1189.00	1190.10	1184.38	1186.10	1841676	AMZN
3018	2017-12-29 00:00:00	1182.35	1184.00	1167.50	1169.47	2688391	AMZN

In [19]:
```
stocks.dtypes
```

Out[19]:
```
Date       object
Open      float64
High      float64
Low       float64
Close     float64
Volume      int64
Name       object
dtype: object
```

In [20]:
```
type(stocks["Date"])
```

Out[20]:
```
pandas.core.series.Series
```

In [21]:
```
type(stocks[["High","Volume"]])
```

Out[21]:
```
pandas.core.frame.DataFrame
```

In [22]:
```
#두 주가를 합친 stocks.csv파일의 전반적인 특징을 살피기 위해 전반적인 성격을 나타내 보았습니다.
```

In [23]:
```
stocks[["High","Low"]].head(10)
```

Out[23]:

	High	Low
0	218.05	209.32
1	224.70	220.09
2	226.00	220.97
3	235.49	226.85
4	236.94	230.70
5	235.36	231.25
6	237.79	234.82
7	237.73	230.98
8	233.68	231.04
9	235.18	231.50

In [24]:
```
stocks[["Volume"]]>1000000
```

Out[24]:

	Volume
0	True
1	True
2	True
3	True
4	True
...	...
3014	True
3015	True
3016	True
3017	True
3018	True

6038 rows × 1 columns

In [25]:
```
stocks["Name"].value_counts()
```

Out[25]:
```
GOOGL    3019
AMZN     3019
Name: Name, dtype: int64
```

In [26]:
```
stocks["Name"].str.lower()
```

Out[26]:
```
0       googl
1       googl
2       googl
3       googl
4       googl
        ...
3014     amzn
3015     amzn
3016     amzn
3017     amzn
3018     amzn
Name: Name, Length: 6038, dtype: object
```

In [27]:
```
open_close = stocks[(stocks["Open"]>1180) & (stocks["Close"]>1180)]
```

In [28]:
```
open_close
```

Out[28]:

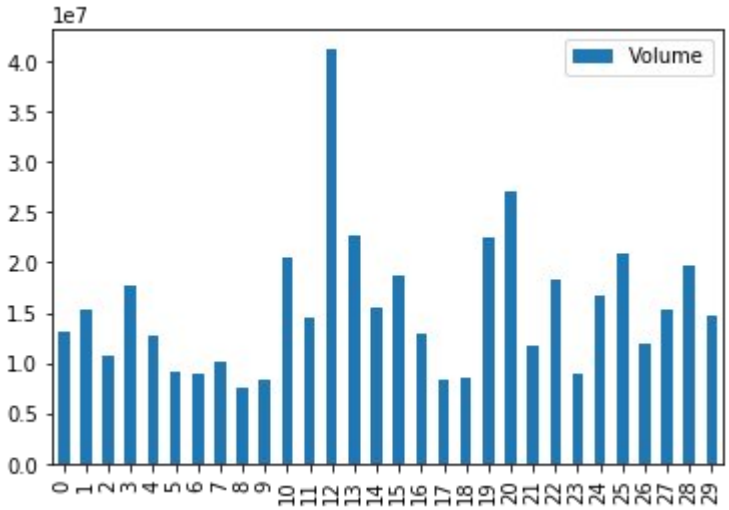
	Date	Open	High	Low	Close	Volume	Name
2995	2017-11-27 00:00:00	1202.66	1213.41	1191.15	1195.83	6744045	AMZN
2996	2017-11-28 00:00:00	1204.88	1205.34	1188.52	1193.60	4559449	AMZN
3010	2017-12-18 00:00:00	1187.37	1194.78	1180.91	1190.58	2947625	AMZN
3011	2017-12-19 00:00:00	1189.15	1192.97	1179.14	1187.38	2587792	AMZN
3017	2017-12-28 00:00:00	1189.00	1190.10	1184.38	1186.10	1841676	AMZN

In [29]:
```
# 각 열의 요소를 묶어보고 큰 데이터 프레임 형태에서 특정 부분만 보고 싶은 경우에 해당하는 몇가지 명령어를 실행해봤습니다.
```

In [30]:
```
import matplotlib.pyplot as plt
```

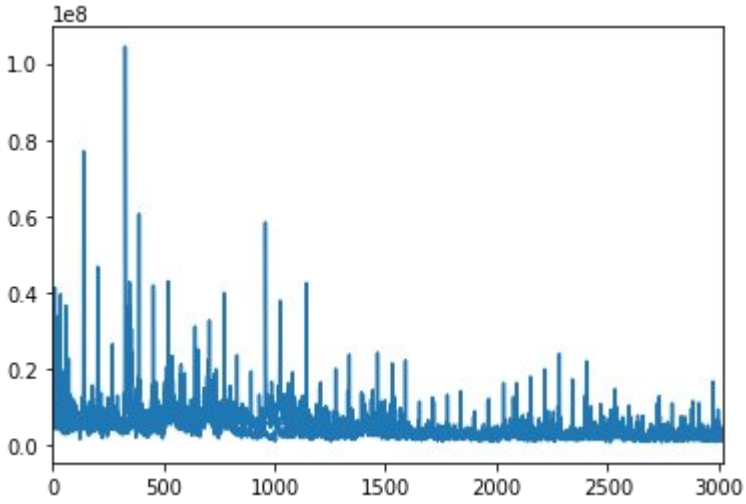
In [31]:
```
stocks[["Volume"]].head(30).plot.bar(stacked=True)
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8cb70ae08>



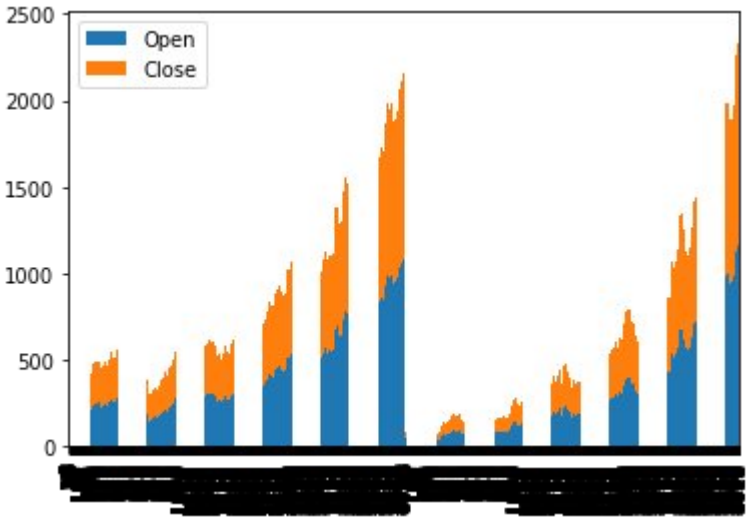
In [32]:
```
stocks["Volume"].plot()
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8cba2e288>



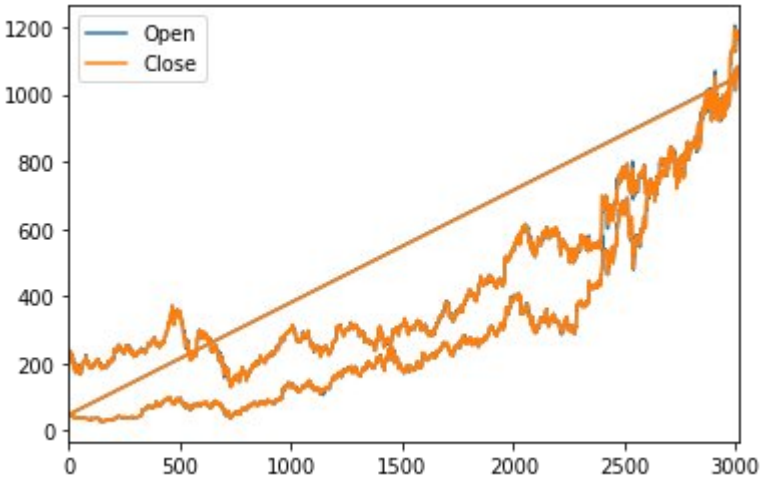
In [33]:
```
stocks[["Open","Close"]].plot.bar(stacked=True)
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8cba9bf48>



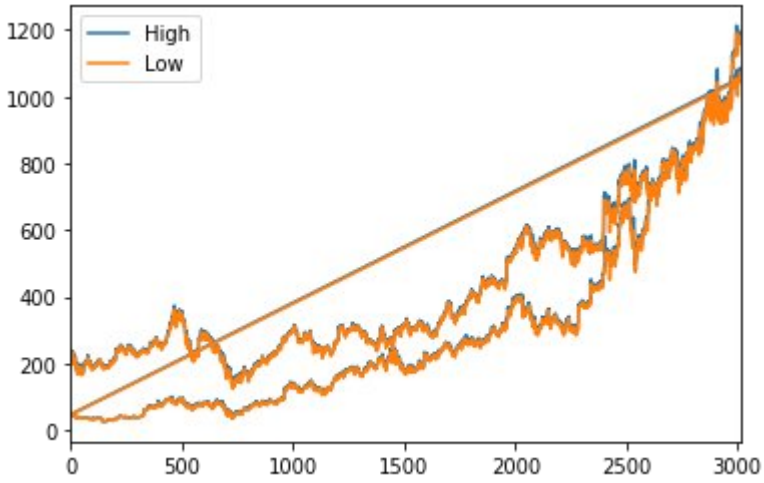
In [34]:
```
stocks[["Open","Close"]].plot()
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8d41707c8>



In [35]:
```
stocks[["High","Low"]].plot()
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f8d463a888>
```



In [36]:
```
#  volume,open(시가),close(종가)의 흐름을 시각화로 표현해봤습니다. 하지만 여기서 중요한 것은 단지 두 데이터를 나열한 상태에서 실행한 값이라는 점입니다.

# 만약 구글 혹은 아마존 중 하나의 객체에 대한 시각화였으면 현금흐름을 크게 보는 의미가 있지만 이것의 경우는 아직 의미가 없습니다.
```

In [37]:
```
stocks["Date"].max()
```

Out[37]:
```
Timestamp('2017-12-29 00:00:00')
```

In [38]:
```
stocks["Date"].min()
```

Out[38]:
```
Timestamp('2006-01-03 00:00:00')
```

In [39]:
```
stocks["Date"].max()-stocks["Date"].min()
```

Out[39]:
```
Timedelta('4378 days 00:00:00')
```

In [40]:
```
stocks["Date"].head(10)
```

Out[40]:
```
0    2006-01-03 00:00:00
1    2006-01-04 00:00:00
2    2006-01-05 00:00:00
3    2006-01-06 00:00:00
4    2006-01-09 00:00:00
5    2006-01-10 00:00:00
6    2006-01-11 00:00:00
7    2006-01-12 00:00:00
8    2006-01-13 00:00:00
9    2006-01-17 00:00:00
Name: Date, dtype: object
```

In [ ]:
```
# 두 객체의 흐름을 이해하기 위해서 우선 시작과 끝 시간을 출력했습니다. 2006~2017년까지의 흐름이므로 부분적 시간의 흐름을 알기 위한 큰 그림이 완성되었습니다.
```

In [42]:
```
stocks[["Name","Volume"]].groupby("Name").describe()
```

Out[42]:

	Volume							
	count	mean	std	min	25%	50%	75%	max
Name								
AMZN	3019.0	5.931712e+06	5.122034e+06	986435.0	3137037.0	4724100.0	7135245.5	1044
GOOGL	3019.0	3.551504e+06	3.038599e+06	521141.0	1760854.0	2517630.0	4242182.5	411

In [43]:
```
stocks[["Name","Open"]].groupby("Name").describe()
```

Out[43]:

	Open							
	count	mean	std	min	25%	50%	75%	max
Name								
AMZN	3019.0	299.335310	280.120547	26.09	81.175	205.33	375.57	1204.88
GOOGL	3019.0	428.200802	236.320026	131.39	247.775	310.48	572.14	1083.02

In [44]:
```
stocks[["Name","Close"]].groupby("Name").describe()
```

Out[44]:

	Close							
	count	mean	std	min	25%	50%	75%	max
Name								
AMZN	3019.0	299.376231	279.980161	26.07	81.090	205.44	375.14	1195.83
GOOGL	3019.0	428.044001	236.343238	128.85	247.605	310.08	570.77	1085.09

In [45]:
```
stocks[["Name","High"]].groupby("Name").describe()
```

Out[45]:

	High							
	count	mean	std	min	25%	50%	75%	max
Name								
AMZN	3019.0	302.371163	281.826442	26.30	82.58	208.00	379.155	1213.41
GOOGL	3019.0	431.835618	237.514087	134.82	250.19	312.81	575.975	1086.49

In [46]:
```
stocks[["Name","Low"]].groupby("Name").describe()
```

Out[46]:

	Low							
	count	mean	std	min	25%	50%	75%	max
Name								
AMZN	3019.0	296.037695	277.927134	25.76	79.725	202.10	373.0	1191.15
GOOGL	3019.0	424.130275	234.923747	123.77	244.035	307.79	565.9	1072.27

In [ ]:
```
#최종 시계열 분석을 하기 전에 아마존과 구글 주식의 최고가,최저가,시가,종가,거래량의 통계적 분석을 비교하는 표를 나타냈습니다.
```

In [47]:
```
stocks_pivot= stocks.pivot(index="Date", columns="Name", values="Volume")
```

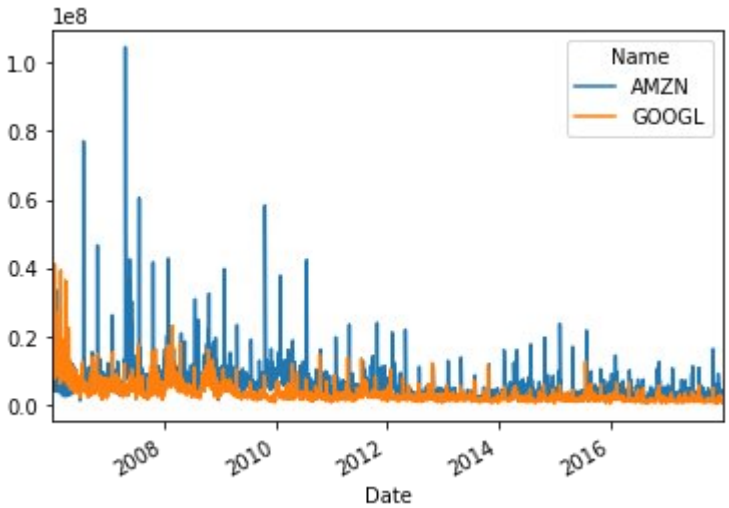
In [48]:
```
stocks_pivot.head()
```

Out[48]:

Name	AMZN	GOOGL
Date		
2006-01-03	7582127	13137450
2006-01-04	7440914	15292353
2006-01-05	5417258	10815661
2006-01-06	6154285	17759521
2006-01-09	8945056	12795837

In [49]:
```
stocks_pivot.plot()
```

Out[49]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f8d5111bc8>
```



In [51]:
```
stocks_pivot.index.year
```

Out[51]:
```
Int64Index([2006, 2006, 2006, 2006, 2006, 2006, 2006, 2006, 2006, 2006,
            ...
            2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017],
           dtype='int64', name='Date', length=3019)
```

In [52]:
```
stocks_pivot.index.month
```

Out[52]:
```
Int64Index([ 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
            ...
            12, 12, 12, 12, 12, 12, 12, 12, 12, 12],
           dtype='int64', name='Date', length=3019)
```

In [53]:
```
stocks_pivot.index.date
```

Out[53]:
```
array([datetime.date(2006, 1, 3), datetime.date(2006, 1, 4),
       datetime.date(2006, 1, 5), ..., datetime.date(2017, 12, 27),
       datetime.date(2017, 12, 28), datetime.date(2017, 12, 29)],
      dtype=object)
```

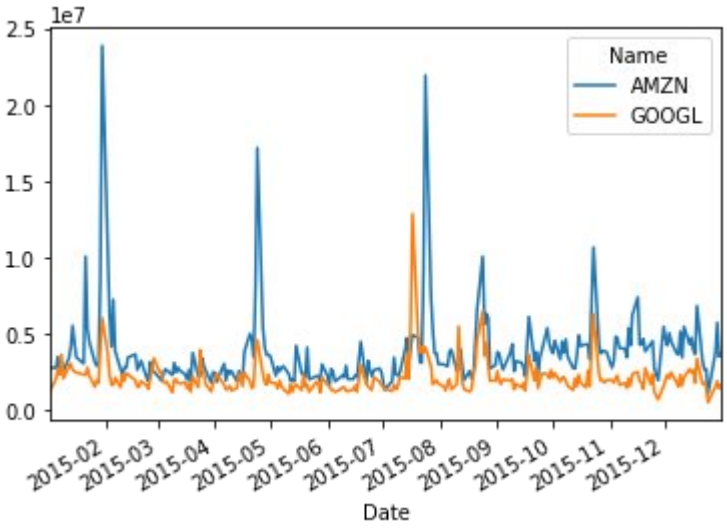
In [ ]:
```
#피벗을 이용해서 데이터 프레임에서 필요한 정보를 추출해서 재 나열했습니다. 이러거 나면 이제 그때그때 원하는 정보를 시각화하기 알맞는 양의 정보를 확보한 것입니다.

#피벗,분류를 하기 전의 시각화와 다르게 이번에는 구글과 아마존이 확실히 구별된 상태로 그래프의 선이 움직이는 것을 볼 수 있습니다.

#plot시각화 이후 전체적인 시간의 차이를 숫자로 나타내기 위해 index()함수를 썼습니다.
```

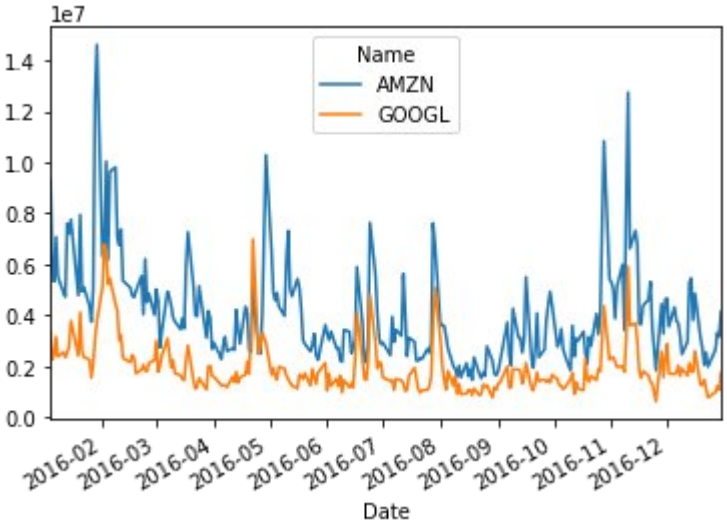
In [54]:
```
stocks_pivot["2015-01-01":"2015-12-31"].plot()
```

Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8d546f808>



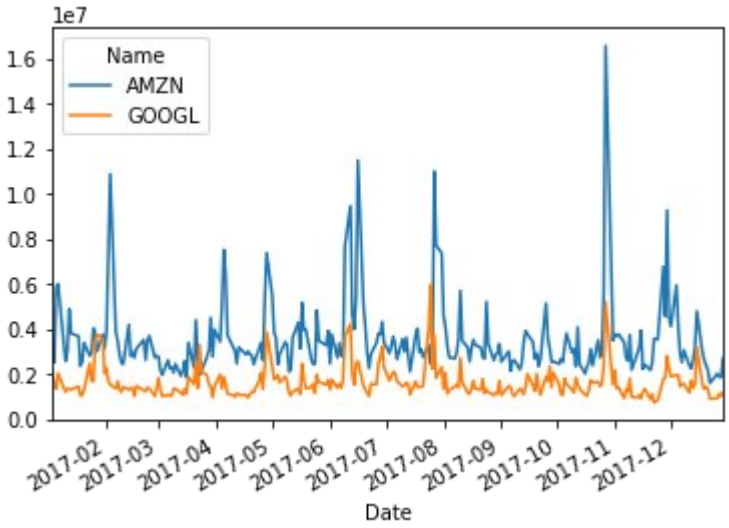
In [55]:
```
stocks_pivot["2016-01-01":"2016-12-31"].plot()
```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8d587aa48>



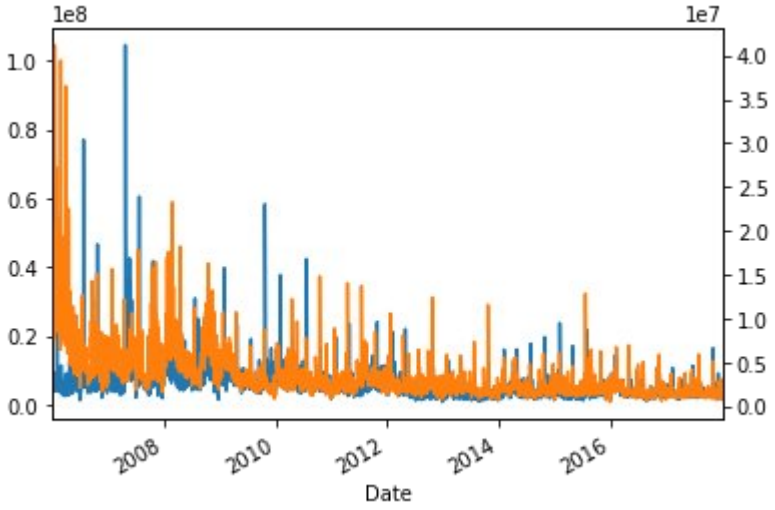
In [56]:
```
stocks_pivot["2017-01-01":"2017-12-29"].plot()
```

Out[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f8d5cb21c8>
```



In [57]:
```
stocks_pivot["AMZN"].plot()
stocks_pivot["GOOGL"].plot(secondary_y=True)
```

Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f8dd159a88>
```

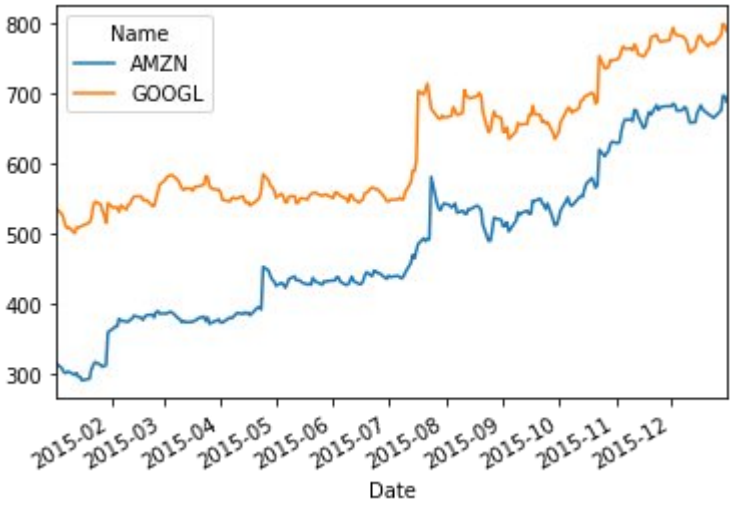


In [ ]:
```
# 11년간의 흐름을 모두 나타나기에는 비교적 너무 옛날인 자료도 있습니다.
# 만약 사용자가 2018년 시점에서 18년 이후의 주가흐름을 비교분석하기 위해서는 최근 3년치의 각 현금흐름의 분할된 정보가 최적일 것입니다.
# 그래서 위에서부터 values()에 차례대로 거래량,시가,종가,최고가,최저가를 차례대로 입력하여 최곤 3년을 각 1년씩의 흐름으로 분할했습니다.
```

In [58]:
```
stocks_pivot= stocks.pivot(index="Date", columns="Name", values="High")
```

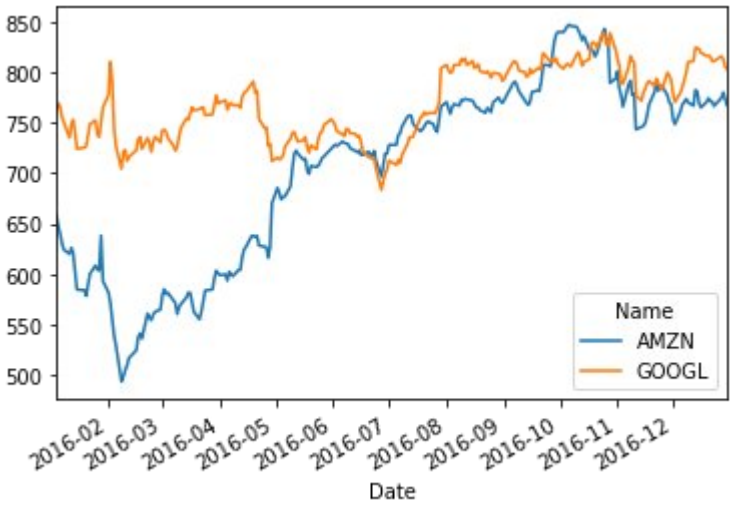
In [59]:
```
stocks_pivot["2015-01-01":"2015-12-31"].plot()
```

Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8ded8ed88>



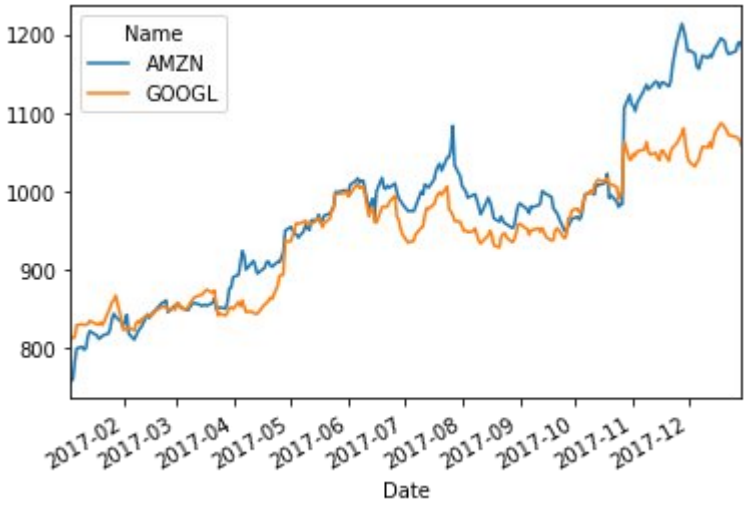
In [60]:
```
stocks_pivot["2016-01-01":"2016-12-31"].plot()
```

Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8df316f48>



In [61]:
```
stocks_pivot["2017-01-01":"2017-12-29"].plot()
```

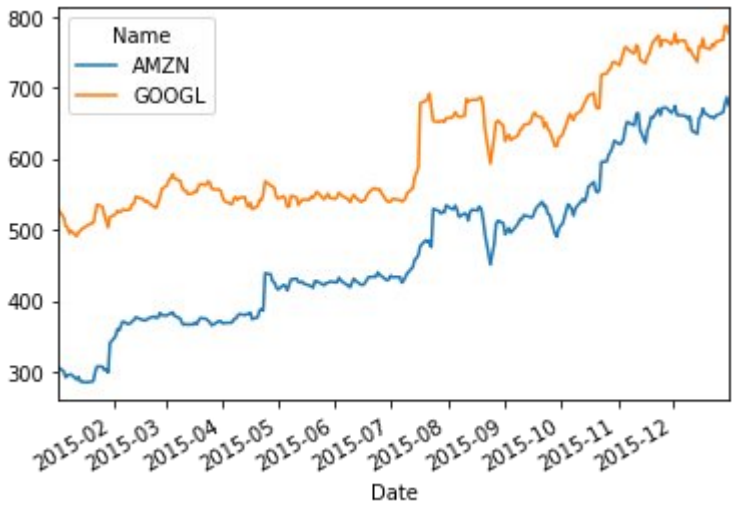
Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8dedd6288>



In [62]:
```
stocks_pivot= stocks.pivot(index="Date", columns="Name", values="Low")
```

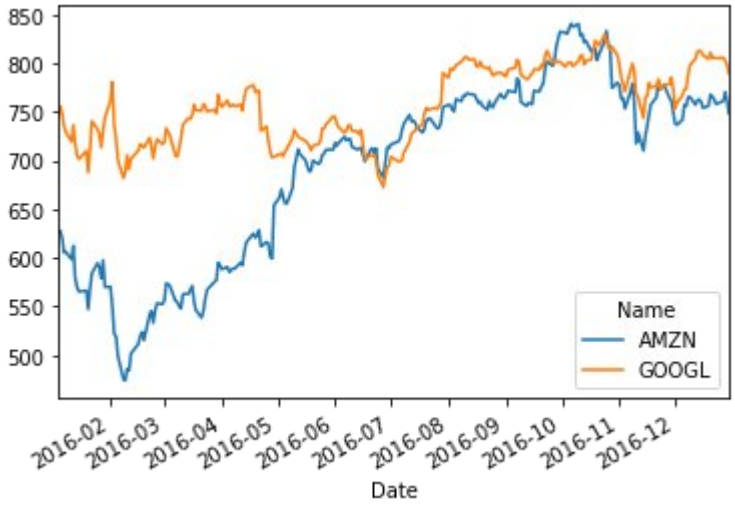
In [63]:
```
stocks_pivot["2015-01-01":"2015-12-31"].plot()
```

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8dedfe448>



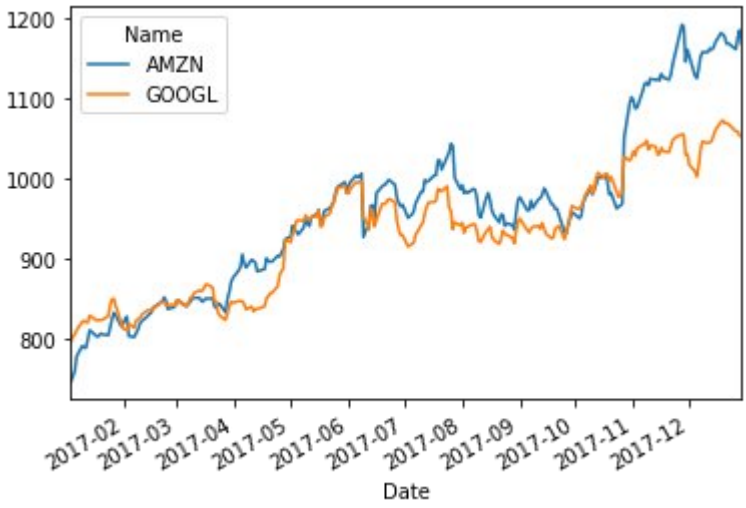
In [64]:
```
stocks_pivot["2016-01-01":"2016-12-31"].plot()
```

Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8deeb8188>



In [65]:
```
stocks_pivot["2017-01-01":"2017-12-29"].plot()
```

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8dedfff48>



In [66]:
```
stocks_pivot= stocks.pivot(index="Date", columns="Name", values="Open")
```

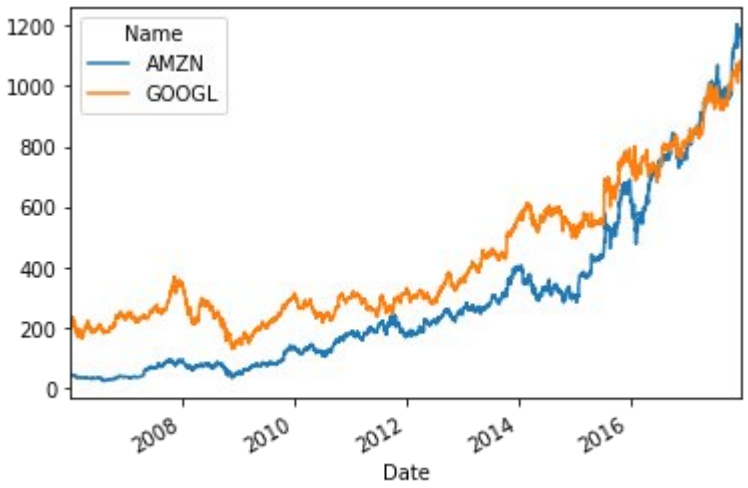
In [67]:
```
stocks_pivot.head()
```

Out[67]:

Name	AMZN	GOOGL
Date		
2006-01-03	47.47	211.47
2006-01-04	47.48	222.17
2006-01-05	47.16	223.22
2006-01-06	47.97	228.66
2006-01-09	46.55	233.44

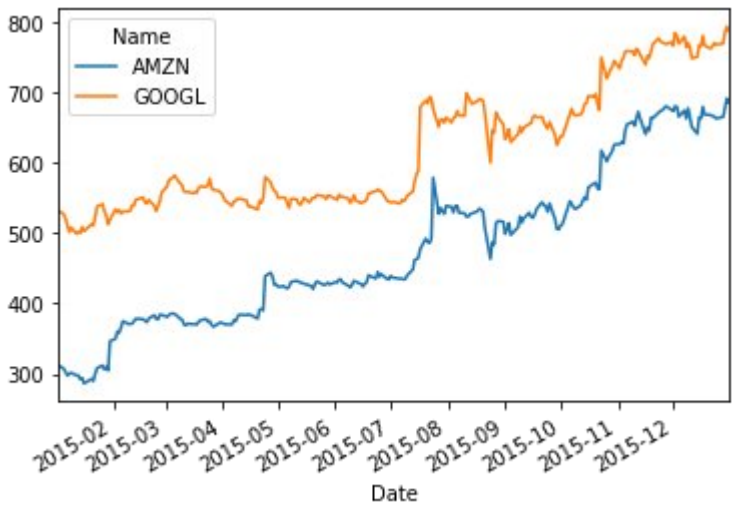
In [68]:
```
stocks_pivot.plot()
```

Out[68]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8de8a45c8>



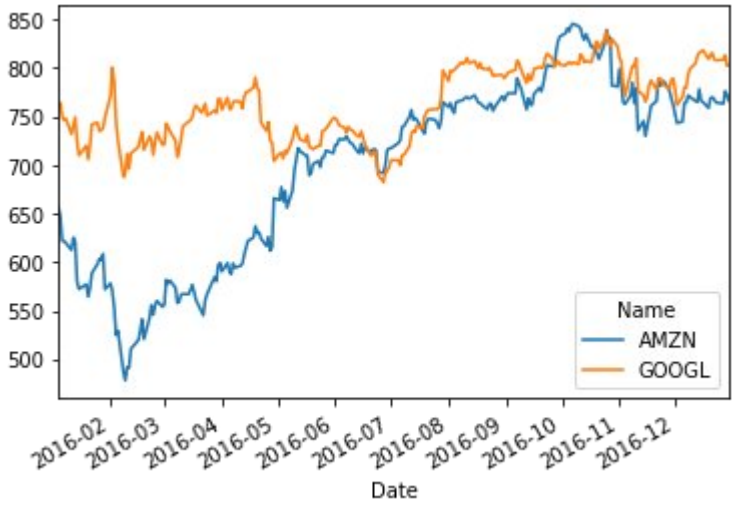
In [69]:
```
stocks_pivot["2015-01-01":"2015-12-31"].plot()
```

Out[69]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8deaf6d88>



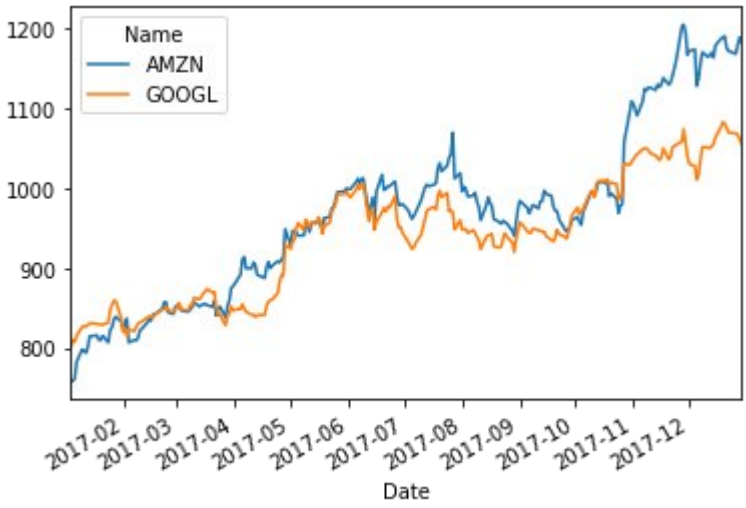
In [70]:
```
stocks_pivot["2016-01-01":"2016-12-31"].plot()
```

Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8deb2eb88>



In [71]:
```
stocks_pivot["2017-01-01":"2017-12-29"].plot()
```

Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8deba2c88>



In [72]:
```
stocks_pivot= stocks.pivot(index="Date", columns="Name", values="Close")
```

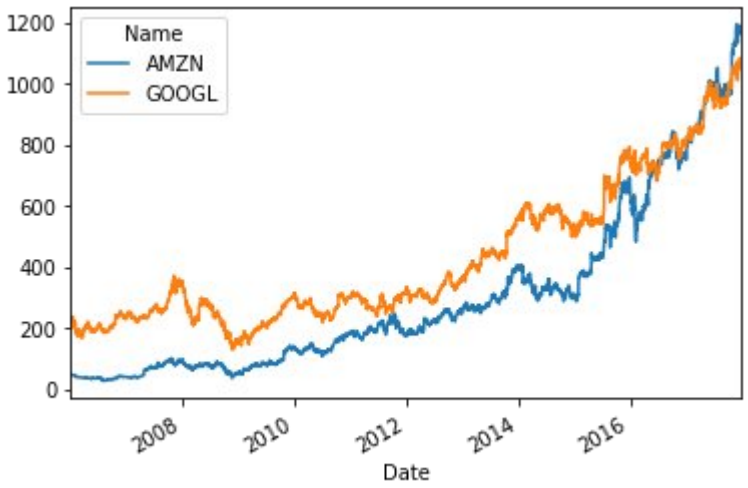
In [73]:
```
stocks_pivot.head()
```

Out[73]:

Name	AMZN	GOOGL
Date		
2006-01-03	47.58	217.83
2006-01-04	47.25	222.84
2006-01-05	47.65	225.85
2006-01-06	47.87	233.06
2006-01-09	47.08	233.68

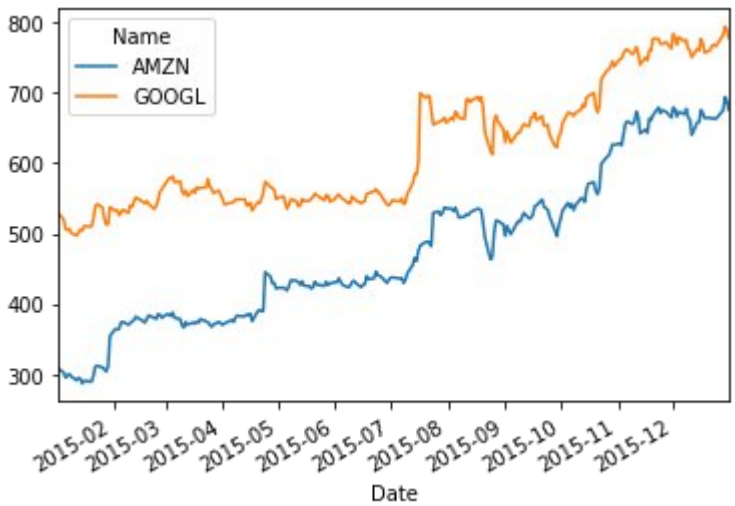
In [74]:
```
stocks_pivot.plot()
```

Out[74]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8debd8648>



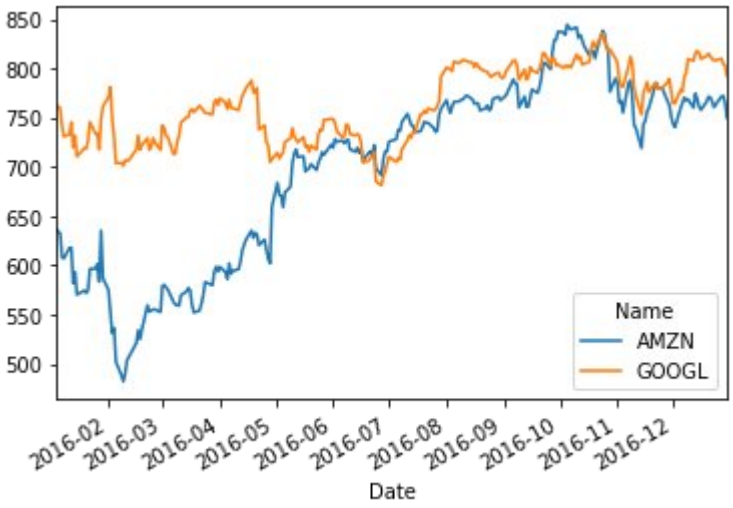
In [75]:
```
stocks_pivot["2015-01-01":"2015-12-31"].plot()
```

Out[75]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8dec316c8>



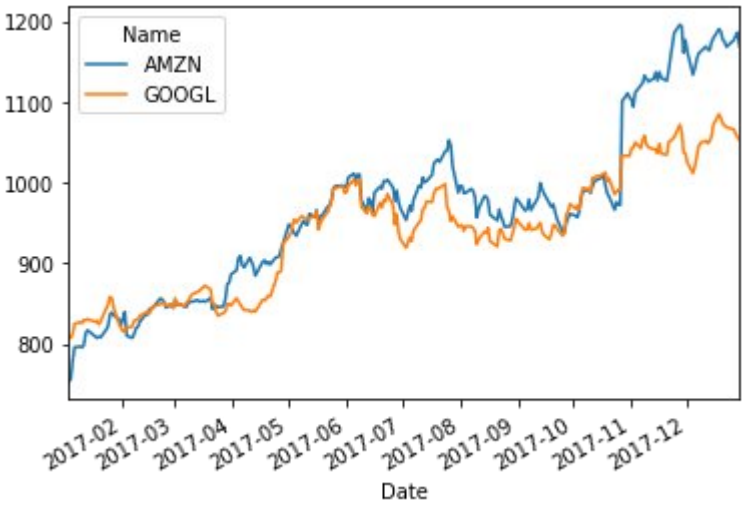
In [76]:
```
stocks_pivot["2016-01-01":"2016-12-31"].plot()
```

Out[76]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8dec6bd08>



In [77]:
```
stocks_pivot["2017-01-01":"2017-12-29"].plot()
```

Out[77]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8dec94408>



In [ ]:
```
# 각 항목의 현금흐름이 크게 요동치지 않아서 시계열 흐름 또한 서로 거의 비슷하게 나타났습니다.
# pivot을 이용해서 정보를 분류,추출하여 원하는 구간을 선택해서 분석할 수 있으면 이것을 기초로 더 많은 기법을 추가하면 보다 정교한 분석을 할 수 있을 것 같습니다.
```