2_Understanding Financial Data

September 13, 2021

[3]:
```
# source: https://www.kaggle.com/camnugent/sandp500
!wget -nc https://lazyprogrammer.me/course_files/all_stocks_5yr.csv
```

```
zsh:1: command not found: wget
```

[2]:
```
import pandas as pd
df = pd.read_csv('all_stocks_5yr.csv')
```

[3]:
```
df.head()
```

[3]:

	date	open	high	low	close	volume	Name
0	2013-02-08	15.07	15.12	14.63	14.75	8407500	AAL
1	2013-02-11	14.89	15.01	14.26	14.46	8882000	AAL
2	2013-02-12	14.45	14.51	14.10	14.27	8126000	AAL
3	2013-02-13	14.30	14.94	14.25	14.66	10259500	AAL
4	2013-02-14	14.94	14.96	13.16	13.99	31879900	AAL

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 619040 entries, 0 to 619039
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   date    619040 non-null  object 
 1   open    619029 non-null  float64
 2   high    619032 non-null  float64
 3   low     619032 non-null  float64
 4   close   619040 non-null  float64
 5   volume  619040 non-null  int64  
 6   Name    619040 non-null  object 
dtypes: float64(4), int64(1), object(2)
memory usage: 33.1+ MB
```

[5]:
```
#using this we get symbols array we used in Get Data file
df['Name'].unique()
```

[5]:
```
array(['AAL', 'AAPL', 'AAP', 'ABBV', 'ABC', 'ABT', 'ACN', 'ADBE', 'ADI',
       'ADM', 'ADP', 'ADSK', 'ADS', 'AEE', 'AEP', 'AES', 'AET', 'AFL',
       'AGN', 'AIG', 'AIV', 'AIZ', 'AJG', 'AKAM', 'ALB', 'ALGN', 'ALK',
       'ALLE', 'ALL', 'ALXN', 'AMAT', 'AMD', 'AME', 'AMGN', 'AMG', 'AMP',
       'AMT', 'AMZN', 'ANDV', 'ANSS', 'ANTM', 'AON', 'AOS', 'APA', 'APC',
       'APD', 'APH', 'APTV', 'ARE', 'ARNC', 'ATVI', 'AVB', 'AVGO', 'AVY',
       'AWK', 'AXP', 'AYI', 'AZO', 'A', 'BAC', 'BAX', 'BA', 'BBT', 'BBY',
       'BDX', 'BEN', 'BF.B', 'BHF', 'BHGE', 'BIIB', 'BK', 'BLK', 'BLL',
       'BMY', 'BRK.B', 'BSX', 'BWA', 'BXP', 'CAG', 'CAH', 'CAT', 'CA',
       'CBG', 'CBOE', 'CBS', 'CB', 'CCI', 'CCL', 'CDNS', 'CELG', 'CERN',
       'CFG', 'CF', 'CHD', 'CHK', 'CHRW', 'CHTR', 'CINF', 'CI', 'CLX',
       'CL', 'CMA', 'CMCSA', 'CME', 'CMG', 'CMI', 'CMS', 'CNC', 'CNP',
       'COF', 'COG', 'COL', 'COO', 'COP', 'COST', 'COTY', 'CPB', 'CRM',
       'CSCO', 'CSRA', 'CSX', 'CTAS', 'CTL', 'CTSH', 'CTXS', 'CVS', 'CVX',
       'CXO', 'C', 'DAL', 'DE', 'DFS', 'DGX', 'DG', 'DHI', 'DHR', 'DISCA',
       'DISCK', 'DISH', 'DIS', 'DLR', 'DLTR', 'DOV', 'DPS', 'DRE', 'DRI',
       'DTE', 'DUK', 'DVA', 'DVN', 'DWDP', 'DXC', 'D', 'EA', 'EBAY',
       'ECL', 'ED', 'EFX', 'EIX', 'EL', 'EMN', 'EMR', 'EOG', 'EQIX',
       'EQR', 'EQT', 'ESRX', 'ESS', 'ES', 'ETFC', 'ETN', 'ETR', 'EVHC',
       'EW', 'EXC', 'EXPD', 'EXPE', 'EXR', 'FAST', 'FBHS', 'FB', 'FCX',
       'FDX', 'FE', 'FFIV', 'FISV', 'FIS', 'FITB', 'FLIR', 'FLR', 'FLS',
       'FL', 'FMC', 'FOXA', 'FOX', 'FRT', 'FTI', 'FTV', 'F', 'GD', 'GE',
       'GGP', 'GILD', 'GIS', 'GLW', 'GM', 'GOOGL', 'GOOG', 'GPC', 'GPN',
       'GPS', 'GRMN', 'GS', 'GT', 'GWW', 'HAL', 'HAS', 'HBAN', 'HBI',
       'HCA', 'HCN', 'HCP', 'HD', 'HES', 'HIG', 'HII', 'HLT', 'HOG',
       'HOLX', 'HON', 'HPE', 'HPQ', 'HP', 'HRB', 'HRL', 'HRS', 'HSIC',
       'HST', 'HSY', 'HUM', 'IBM', 'ICE', 'IDXX', 'IFF', 'ILMN', 'INCY',
       'INFO', 'INTC', 'INTU', 'IPG', 'IP', 'IQV', 'IRM', 'IR', 'ISRG',
       'ITW', 'IT', 'IVZ', 'JBHT', 'JCI', 'JEC', 'JNJ', 'JNPR', 'JPM',
       'JWN', 'KEY', 'KHC', 'KIM', 'KLAC', 'KMB', 'KMI', 'KMX', 'KORS',
       'KO', 'KR', 'KSS', 'KSU', 'K', 'LB', 'LEG', 'LEN', 'LH', 'LKQ',
       'LLL', 'LLY', 'LMT', 'LNC', 'LNT', 'LOW', 'LRCX', 'LUK', 'LUV',
       'LYB', 'L', 'MAA', 'MAC', 'MAR', 'MAS', 'MAT', 'MA', 'MCD', 'MCHP',
       'MCK', 'MCO', 'MDLZ', 'MDT', 'MET', 'MGM', 'MHK', 'MKC', 'MLM',
       'MMC', 'MMM', 'MNST', 'MON', 'MOS', 'MO', 'MPC', 'MRK', 'MRO',
       'MSFT', 'MSI', 'MS', 'MTB', 'MTD', 'MU', 'MYL', 'M', 'NAVI', 'NBL',
       'NCLH', 'NDAQ', 'NEE', 'NEM', 'NFLX', 'NFX', 'NI', 'NKE', 'NLSN',
       'NOC', 'NOV', 'NRG', 'NSC', 'NTAP', 'NTRS', 'NUE', 'NVDA', 'NWL',
       'NWSA', 'NWS', 'OKE', 'OMC', 'ORCL', 'ORLY', 'OXY', 'O', 'PAYX',
       'PBCT', 'PCAR', 'PCG', 'PCLN', 'PDCO', 'PEG', 'PEP', 'PFE', 'PFG',
       'PGR', 'PG', 'PHM', 'PH', 'PKG', 'PKI', 'PLD', 'PM', 'PNC', 'PNR',
       'PNW', 'PPG', 'PPL', 'PRGO', 'PRU', 'PSA', 'PSX', 'PVH', 'PWR',
       'PXD', 'PX', 'PYPL', 'QCOM', 'QRVO', 'RCL', 'REGN', 'REG', 'RE',
       'RF', 'RHI', 'RHT', 'RJF', 'RL', 'RMD', 'ROK', 'ROP', 'ROST',
       'RRC', 'RSG', 'RTN', 'SBAC', 'SBUX', 'SCG', 'SCHW', 'SEE', 'SHW',
       'SIG', 'SJM', 'SLB', 'SLG', 'SNA', 'SNI', 'SNPS', 'SO', 'SPGI',
       'SPG', 'SRCL', 'SRE', 'STI', 'STT', 'STX', 'STZ', 'SWKS', 'SWK',
```

```
       'SYF', 'SYK', 'SYMC', 'SYY', 'TAP', 'TDG', 'TEL', 'TGT', 'TIF',
       'TJX', 'TMK', 'TMO', 'TPR', 'TRIP', 'TROW', 'TRV', 'TSCO', 'TSN',
       'TSS', 'TWX', 'TXN', 'TXT', 'T', 'UAA', 'UAL', 'UA', 'UDR', 'UHS',
       'ULTA', 'UNH', 'UNM', 'UNP', 'UPS', 'URI', 'USB', 'UTX', 'VAR',
       'VFC', 'VIAB', 'VLO', 'VMC', 'VNO', 'VRSK', 'VRSN', 'VRTX', 'VTR',
       'VZ', 'V', 'WAT', 'WBA', 'WDC', 'WEC', 'WFC', 'WHR', 'WLTW', 'WMB',
       'WMT', 'WM', 'WRK', 'WU', 'WYNN', 'WYN', 'WY', 'XEC', 'XEL',
       'XLNX', 'XL', 'XOM', 'XRAY', 'XRX', 'XYL', 'YUM', 'ZBH', 'ZION',
       'ZTS'], dtype=object)
```

[6]:
```
# we have 505 stocks in S&P5000
df['Name'].unique().shape
```

[6]:
```
(505,)
```

[7]:
```
#Extracting  IBM Stock data from dataframe
ibm = df[df['Name'] == 'IBM']
```

[9]:
```
ibm.head()
```

[9]:

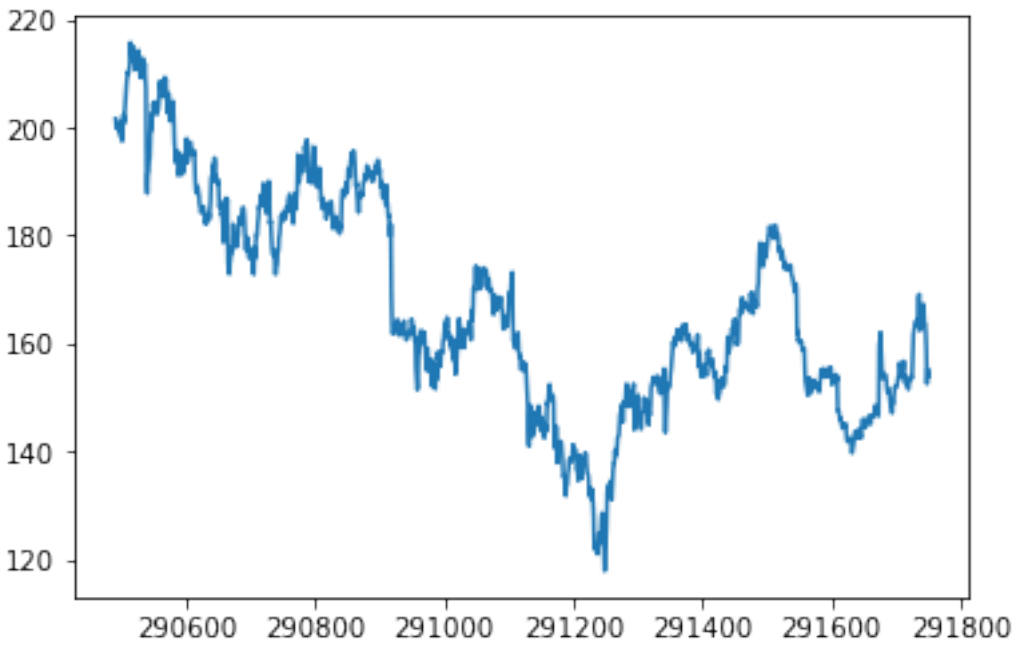
	date	open	high	low	close	volume	Name
290491	2013-02-08	199.97	202.090	199.68	201.68	2893254	IBM
290492	2013-02-11	200.98	201.950	199.75	200.16	2944651	IBM
290493	2013-02-12	200.01	200.735	199.02	200.04	2461779	IBM
290494	2013-02-13	200.65	200.950	199.57	200.09	2169757	IBM
290495	2013-02-14	199.73	200.320	199.26	199.65	3294126	IBM

[10]:
```
#IBM close Price
ibm['close']
```

[10]:
```
290491    201.68
290492    200.16
290493    200.04
290494    200.09
290495    199.65
           …
291745    162.40
291746    159.03
291747    152.53
291748    155.34
291749    153.85
Name: close, Length: 1259, dtype: float64
```

[11]:
```
#Plotting IBM close Price
ibm['close'].plot()
```

[11]:
```
<AxesSubplot:>
```



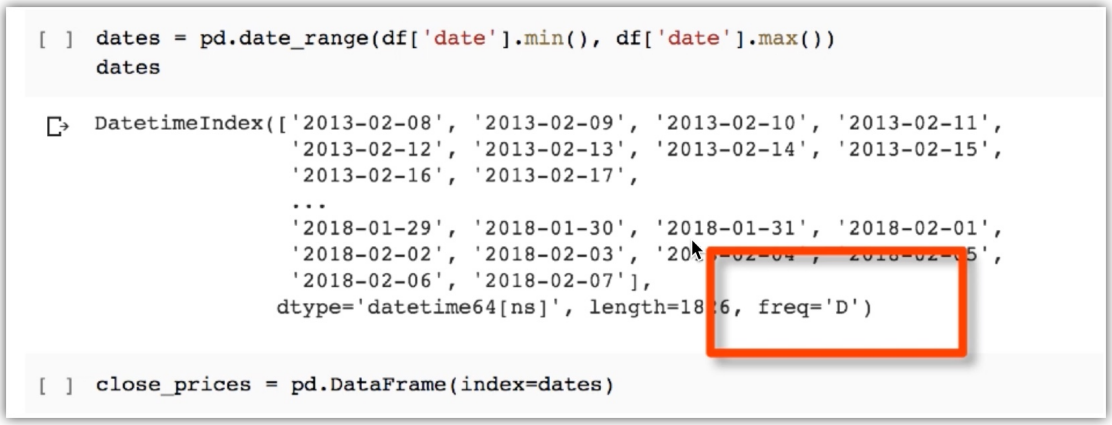
[12]:
```
# convert to timestamp object
df['date'] = pd.to_datetime(df['date'])
```

[13]:
```
#getting Max and min date for which IBM data available
df['date'].min(), df['date'].max()
```

[13]:
```
(Timestamp('2013-02-08 00:00:00'), Timestamp('2018-02-07 00:00:00'))
```

[22]:
```
'''
I'm going to call derange passing in the min and max dates that I found earlier.
This will create a date time index object.
There is an attribute called freq=D Show It for Frequency, which tells us that␣
 ↪each value in the index
is a different day.
These stands for Daily.
'''
from IPython.display import Image
Image(filename='/Users/subhasish/GIT/Interstellar/SB-AI-DEV/ML/SB/TimeSeries/
 ↪Lazy Programmers/Image/2021-09-13_19-23-14.jpg')
```

[22]:

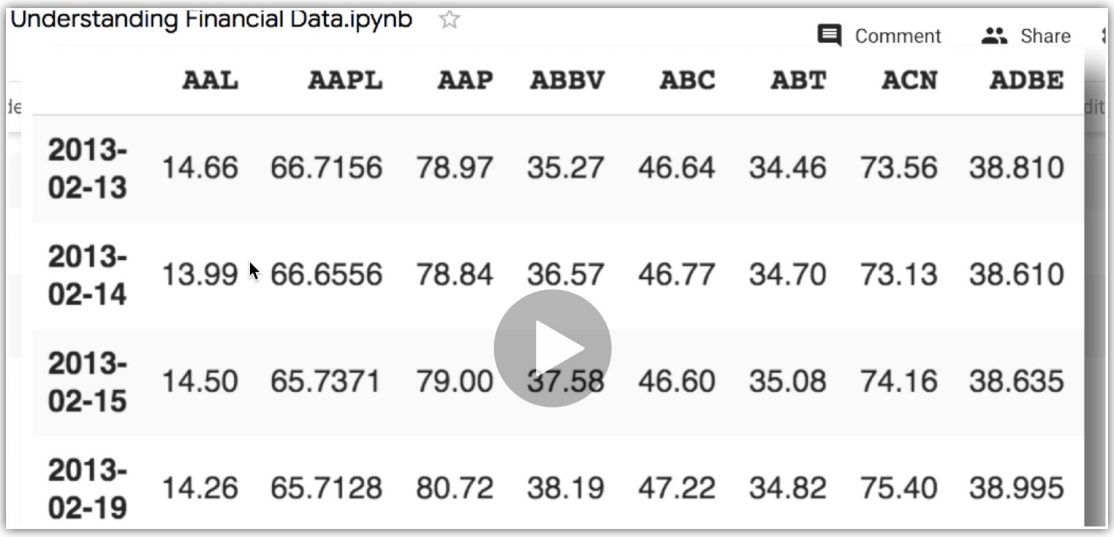


[14]:
```
#Getiing all the dates from data frame
dates = pd.date_range(df['date'].min(), df['date'].max())
dates
```

[14]:
```
DatetimeIndex(['2013-02-08', '2013-02-09', '2013-02-10', '2013-02-11',
               '2013-02-12', '2013-02-13', '2013-02-14', '2013-02-15',
               '2013-02-16', '2013-02-17',
               …
               '2018-01-29', '2018-01-30', '2018-01-31', '2018-02-01',
               '2018-02-02', '2018-02-03', '2018-02-04', '2018-02-05',
               '2018-02-06', '2018-02-07'],
              dtype='datetime64[ns]', length=1826, freq='D')
```

[21]:
```
#setting close_price dataframe index as date
'''
So what we would like to have is a single data free organized by date, where␣
 ↪each column is the close
price for a different stock.
'''
from IPython.display import Image
Image(filename='/Users/subhasish/GIT/Interstellar/SB-AI-DEV/ML/SB/TimeSeries/
 ↪Lazy Programmers/Image/2021-09-13_19-17-34.jpg')
```

[21]:



	AAL	AAPL	AAP	ABBV	ABC	ABT	ACN	ADBE
2013-02-13	14.66	66.7156	78.97	35.27	46.64	34.46	73.56	38.810
2013-02-14	13.99	66.6556	78.84	36.57	46.77	34.70	73.13	38.610
2013-02-15	14.50	65.7371	79.00	37.58	46.60	35.08	74.16	38.635
2013-02-19	14.26	65.7128	80.72	38.19	47.22	34.82	75.40	38.995

[16]:
```
'''
Next, I'm going to create a data frame called Close Prices by calling the␣
 ↪constructor pd.DataFrame(index=dates).

The only argument I'm going to pass into the constructor is the index argument␣
 ↪for which I'm going to
pass in the daytime index I just created.
'''
close_prices = pd.DataFrame(index=dates)
close_prices.head()
```

[16]:
```
Empty DataFrame
Columns: []
Index: [2013-02-08 00:00:00, 2013-02-09 00:00:00, 2013-02-10 00:00:00,
2013-02-11 00:00:00, 2013-02-12 00:00:00]
```

[ ]:
```
'''
Of course, the next step is going to be to fill in close price for different␣
 ↪stocks columns with the data from our original data
There is a slight problem with this, which is that in our original data frame,␣
 ↪the dates are just a regular column.
They are not part of the index.

This actually makes sense for that data frame, since the dates are not unique.
Multiple rows have the same date simply because they contain prices for␣
 ↪different stocks.

We know that we're going to want to do something like a database joint where we␣
 ↪join at two tables on
```

```
some index, specifically the date.

In order for this to work, what we would like to do is create a temporary data␣
 ↪frame for each stock,
one at a time.

This temporary data frame should contain only a single column, the close price␣
 ↪for a specific stock,
and as its index it should have the corresponding dates.
'''
```

[23]:
```
'''
So the next line demonstrates how to do that.

Specifically, we're going to call the pd.DataFrame constructor as arguments␣
 ↪we're going to pass in data
equal to the close price from the IBM subset dataframe, for the index argument␣
 ↪will pass in IBM date
For the columns, argument will pass in the string IBM
'''
df2 = pd.DataFrame(data=ibm['close'].to_numpy(), index=ibm['date'],
                   columns=['IBM'])
```

[24]:
```
'''
The date is being used as the index and we have a single column containing␣
 ↪close prices.
And the name of that column is IBM
'''
df2.head()
```

[24]:

date	IBM
2013-02-08	201.68
2013-02-11	200.16
2013-02-12	200.04
2013-02-13	200.09
2013-02-14	199.65

[25]:
```
'''
OK, so the next thing we want to do is to do this for all of the symbols in our␣
 ↪original data frame.

So next we're going to loop through every symbol and our symbols list.
Inside the loop, we grab the sub data frame containing only the symbol in the␣
 ↪name column.
```

```
Next, we create a new temporary data frame called df_tmp using the same code we␣
 ↪discussed earlier.
Finally, we call the joint function on the closed prices data frame passing in␣
 ↪the df_tmp.
We assign this to close prices.
So on every round of the loop, closed prices will accumulate each new symbol.


'''
symbols = df['Name'].unique()
for symbol in symbols:
    df_sym = df[df['Name'] == symbol]
    df_tmp = pd.DataFrame(data=df_sym['close'].to_numpy(),
                          index=df_sym['date'], columns=[symbol])
    close_prices = close_prices.join(df_tmp) # left-join by default
```

[26]:
```
close_prices.head()
```

[26]:

	AAL	AAPL	AAP	ABBV	ABC	ABT	ACN	ADBE	ADI	\
2013-02-08	14.75	67.8542	78.90	36.25	46.89	34.41	73.31	39.12	45.70	
2013-02-09	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2013-02-10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2013-02-11	14.46	68.5614	78.39	35.85	46.76	34.26	73.07	38.64	46.08	
2013-02-12	14.27	66.8428	78.60	35.42	46.96	34.30	73.37	38.89	46.27	

	ADM	…	XLNX	XL	XOM	XRAY	XRX	XYL	YUM	\
2013-02-08	30.22	…	37.51	28.24	88.61	42.87	31.84	27.09	65.30	
2013-02-09	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2013-02-10	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2013-02-11	30.28	…	37.46	28.31	88.28	42.84	31.96	27.46	64.55	
2013-02-12	30.81	…	37.58	28.41	88.46	42.87	31.84	27.95	64.75	

	ZBH	ZION	ZTS
2013-02-08	75.85	24.14	33.05
2013-02-09	NaN	NaN	NaN
2013-02-10	NaN	NaN	NaN
2013-02-11	75.65	24.21	33.26
2013-02-12	75.44	24.49	33.74

[5 rows x 505 columns]

[32]:
```
'''
It has a date time index with 1826 entries with a frequency D, which
means daily.

It has 505 columns from AAL to ZTS.
```

```
The type is Flow 64, which makes sense.
'''
close_prices.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1826 entries, 2013-02-08 to 2018-02-07
Freq: D
Columns: 505 entries, AAL to ZTS
dtypes: float64(505)
memory usage: 7.1 MB
```

[27]:
```
close_prices.to_csv('sp500_close.csv')
```

[28]:
```
!head sp500_close.csv
```

```
,AAL,AAPL,AAP,ABBV,ABC,ABT,ACN,ADBE,ADI,ADM,ADP,ADSK,ADS,AEE,AEP,AES,AET,AFL,AGN
,AIG,AIV,AIZ,AJG,AKAM,ALB,ALGN,ALK,ALLE,ALL,ALXN,AMAT,AMD,AME,AMGN,AMG,AMP,AMT,A
MZN,ANDV,ANSS,ANTM,AON,AOS,APA,APC,APD,APH,APTV,ARE,ARNC,ATVI,AVB,AVGO,AVY,AWK,A
XP,AYI,AZO,A,BAC,BAX,BA,BBT,BBY,BDX,BEN,BF.B,BHF,BHGE,BIIB,BK,BLK,BLL,BMY,BRK.B,
BSX,BWA,BXP,CAG,CAH,CAT,CA,CBG,CBOE,CBS,CB,CCI,CCL,CDNS,CELG,CERN,CFG,CF,CHD,CHK
,CHRW,CHTR,CINF,CI,CLX,CL,CMA,CMCSA,CME,CMG,CMI,CMS,CNC,CNP,COF,COG,COL,COO,COP,
COST,COTY,CPB,CRM,CSCO,CSRA,CSX,CTAS,CTL,CTSH,CTXS,CVS,CVX,CXO,C,DAL,DE,DFS,DGX,
DG,DHI,DHR,DISCA,DISCK,DISH,DIS,DLR,DLTR,DOV,DPS,DRE,DRI,DTE,DUK,DVA,DVN,DWDP,DX
C,D,EA,EBAY,ECL,ED,EFX,EIX,EL,EMN,EMR,EOG,EQIX,EQR,EQT,ESRX,ESS,ES,ETFC,ETN,ETR,
EVHC,EW,EXC,EXPD,EXPE,EXR,FAST,FBHS,FB,FCX,FDX,FE,FFIV,FISV,FIS,FITB,FLIR,FLR,FL
S,FL,FMC,FOXA,FOX,FRT,FTI,FTV,F,GD,GE,GGP,GILD,GIS,GLW,GM,GOOGL,GOOG,GPC,GPN,GPS
,GRMN,GS,GT,GWW,HAL,HAS,HBAN,HBI,HCA,HCN,HCP,HD,HES,HIG,HII,HLT,HOG,HOLX,HON,HPE
,HPQ,HP,HRB,HRL,HRS,HSIC,HST,HSY,HUM,IBM,ICE,IDXX,IFF,ILMN,INCY,INFO,INTC,INTU,I
PG,IP,IQV,IRM,IR,ISRG,ITW,IT,IVZ,JBHT,JCI,JEC,JNJ,JNPR,JPM,JWN,KEY,KHC,KIM,KLAC,
KMB,KMI,KMX,KORS,KO,KR,KSS,KSU,K,LB,LEG,LEN,LH,LKQ,LLL,LLY,LMT,LNC,LNT,LOW,LRCX,
LUK,LUV,LYB,L,MAA,MAC,MAR,MAS,MAT,MA,MCD,MCHP,MCK,MCO,MDLZ,MDT,MET,MGM,MHK,MKC,M
LM,MMC,MMM,MNST,MON,MOS,MO,MPC,MRK,MRO,MSFT,MSI,MS,MTB,MTD,MU,MYL,M,NAVI,NBL,NCL
H,NDAQ,NEE,NEM,NFLX,NFX,NI,NKE,NLSN,NOC,NOV,NRG,NSC,NTAP,NTRS,NUE,NVDA,NWL,NWSA,
NWS,OKE,OMC,ORCL,ORLY,OXY,O,PAYX,PBCT,PCAR,PCG,PCLN,PDCO,PEG,PEP,PFE,PFG,PGR,PG,
PHM,PH,PKG,PKI,PLD,PM,PNC,PNR,PNW,PPG,PPL,PRGO,PRU,PSA,PSX,PVH,PWR,PXD,PX,PYPL,Q
COM,QRVO,RCL,REGN,REG,RE,RF,RHI,RHT,RJF,RL,RMD,ROK,ROP,ROST,RRC,RSG,RTN,SBAC,SBU
X,SCG,SCHW,SEE,SHW,SIG,SJM,SLB,SLG,SNA,SNI,SNPS,SO,SPGI,SPG,SRCL,SRE,STI,STT,STX
,STZ,SWKS,SWK,SYF,SYK,SYMC,SYY,TAP,TDG,TEL,TGT,TIF,TJX,TMK,TMO,TPR,TRIP,TROW,TRV
,TSCO,TSN,TSS,TWX,TXN,TXT,T,UAA,UAL,UA,UDR,UHS,ULTA,UNH,UNM,UNP,UPS,URI,USB,UTX,
VAR,VFC,VIAB,VLO,VMC,VNO,VRSK,VRSN,VRTX,VTR,VZ,V,WAT,WBA,WDC,WEC,WFC,WHR,WLTW,WM
B,WMT,WM,WRK,WU,WYNN,WYN,WY,XEC,XEL,XLNX,XL,XOM,XRAY,XRX,XYL,YUM,ZBH,ZION,ZTS
2013-02-08,14.75,67.8542,78.9,36.25,46.89,34.41,73.31,39.12,45.7,30.22,60.925,38
.89,154.08,32.61,44.57,11.07,50.6,50.35,87.45,38.79,28.57,38.56,37.97,35.42,62.6
6,32.73,24.335,,45.14,93.66,13.61,2.59,41.46,86.77,146.45,66.49,77.06,261.95,53.
86,74.69,66.28,56.53,17.4875,84.65,84.45,88.35,35.22,,72.62,26.82,13.41,128.84,3
5.32,39.25,39.18,61.8,70.09,385.89,45.08,11.76,69.0,76.56,30.65,15.29,88.25,46.7
333,32.54,,,164.44,27.9,238.16,22.68,37.03,97.25,7.7,37.47,105.14,33.38,45.18,96
```

.85,25.02,24.19,34.46,42.83,86.2,70.79,39.01,14.16,50.065,44.05,,45.232,30.065,2
0.23,59.72,80.89,44.61,61.93,80.73,54.245,35.37,19.375,57.27,320.72,119.47,25.66
,23.02,20.91,57.27,27.45,60.49,104.26,57.87,102.79,,37.93,42.4875,21.16,,21.97,4
2.87,41.36,39.365,72.91,51.2,115.64,96.15,42.68,14.62,92.81,39.88,58.5,46.0,23.0
8,60.81,71.47,64.6,37.64,54.66,65.4,41.06,71.13,45.73,16.07,47.36,64.28,69.0,59.
46,60.42,,,54.31,17.37,56.62,74.4,57.15,55.56,47.84,62.75,73.31,57.5,66.64,219.7
1,55.44,61.83,55.74,151.86,41.11,11.02,58.92,64.47,,43.61,31.08,43.34,65.65,40.2
8,50.69,33.61,28.545,35.64,106.41,40.22,105.25,40.24,37.62,16.61,26.5,63.71,52.5
799,34.85,59.82,,,107.76,48.62,,13.1,66.82,22.5,20.14,40.9,42.6,12.28,28.57,393.
0777,,69.92,24.97,32.23,38.08,151.6,13.59,216.71,41.26,39.84,7.2,10.1275,37.22,6
2.3,46.7,67.01,67.37,24.17,45.39,,52.86,22.32,70.53,,,64.39,24.4,17.915,46.62,44
.555,16.81,80.21,81.35,201.68,29.8,46.785,73.6,50.88,18.32,,21.0,61.51,12.16,42.
6,,34.25,53.43,192.9165,62.72,49.75,27.31,69.07,32.2094,48.08,75.48,21.84,48.63,
55.82,9.57,,21.55,56.57,90.9,37.6,39.77,57.1,38.77,14.1,46.01,95.09,58.91,45.81,
29.74,39.1765,90.24,23.73,77.67,53.72,88.01,29.43,23.19,39.03,42.61,26.73,11.64,
62.11,43.85,66.48,61.64,40.97,17.79,40.32,52.447,94.87,36.39,103.8,43.37,27.76,4
7.1,36.9,13.51,103.11,63.63,98.27,36.55,102.66,15.6133,101.15,61.4,34.69,40.715,
41.18,34.53,27.55,60.3,23.32,104.39,221.15,7.75,28.86,39.92,,57.68,27.91,29.47,7
2.1,44.93,25.8528,29.69,26.92,27.295,32.81,66.09,69.08,24.13,69.0,35.75,52.7,46.
74,12.37,24.14,,,47.82,55.22,34.9,101.57,88.36,43.2,33.32,12.55,47.5,42.77,700.8
3,37.31,31.43,72.6,26.88,30.9,23.44,75.75,19.47,94.32,40.15,34.75,39.47,90.45,63
.35,51.05,53.95,69.44,30.31,108.3,57.6,157.4,64.02,122.47,28.42,128.97,111.61,,6
6.95,,36.13,165.88,51.41,119.24,7.93,35.46,54.96,45.31,177.56,44.09,90.98,119.34
,30.53,72.15,31.12,54.16,69.49,28.185,47.23,16.89,19.25,163.77,62.91,89.16,79.07
,81.89,79.14,60.14,34.4,43.85,42.67,162.76,96.8,76.14,29.25,56.55,35.48,31.85,23
.93,76.24,,63.78,22.59,31.61,44.41,143.05,40.52,62.49,62.89,45.68,37.0334,74.59,
48.93,47.02,73.27,78.86,52.02,23.78,23.2,52.57,34.28,29.31,35.27,12.55,26.31,,23
.6,58.11,99.32,57.74,24.06,66.575,82.38,53.09,33.65,90.09,71.1,36.4357,59.28,46.
13,56.08,84.51,54.9,45.39,46.0,67.42,44.35,39.45,92.53,41.4,48.49,40.06,34.88,10
7.32,,36.0,71.48,36.3,,14.66,126.52,59.89,30.21,66.6,27.84,37.51,28.24,88.61,42.
87,31.84,27.09,65.3,75.85,24.14,33.05
2013-02-09,,
,,
,,
,,
,,
,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
2013-02-10,,
,,
,,
,,
,,
,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
2013-02-11,14.46,68.5614,78.39,35.85,46.76,34.26,73.07,38.64,46.08,30.28,60.34,3
8.89,153.42,32.84,44.73,11.2,50.38,50.26,86.66,39.45,28.92,38.48,37.88,35.31,62.
76,33.03,24.4,,45.25,92.3,13.66,2.67,41.51,84.92,145.57,66.49,76.38,257.21,54.55

,74.71,66.01,56.66,17.4775,83.81,83.47,88.42,35.205,,72.73,26.61,13.57,129.78,35
.4,39.21,39.04,61.98,70.25,385.23,44.6,11.86,68.73,75.87,30.63,15.71,88.31,46.72
,32.05,,,163.96,27.93,239.45,22.57,36.93,97.13,7.63,37.465,105.29,33.37,45.22,96
.6,24.86,24.13,34.78,42.28,86.41,70.03,38.72,14.22,49.97,43.75,,45.124,29.815,20
.05,59.34,79.31,44.6,61.83,80.5,54.145,35.06,19.32,56.93,323.56,119.26,25.67,22.
25,20.92,56.55,27.33,60.22,104.05,57.59,101.99,,38.12,42.6575,21.27,,22.65,43.23
,41.51,38.56,72.4,51.24,115.64,96.52,43.15,14.69,93.23,40.03,57.93,45.75,22.92,6
0.68,70.95,64.31,37.11,54.75,65.21,40.96,71.06,45.42,15.86,47.08,64.4,69.14,59.4
,59.7,,,54.33,17.41,56.41,74.06,56.59,54.98,47.98,61.78,74.13,57.73,66.67,218.03
,56.46,61.37,55.05,151.86,41.12,11.06,59.67,64.62,,43.525,31.42,43.0,64.8,40.41,
50.74,33.55,28.26,35.33,106.01,40.25,103.52,40.42,37.56,16.5,26.48,63.43,52.5533
,34.84,59.2,,,107.82,48.33,,13.11,66.66,22.45,20.18,40.4,42.67,12.44,28.53,391.6
012,,69.89,24.965,32.58,37.87,152.24,13.91,217.61,40.87,39.69,7.12,10.0975,36.66
,62.36,46.91,66.38,66.74,24.41,45.23,,53.49,22.48,70.09,,,64.86,24.45,17.815,46.
54,44.725,16.9,80.38,80.18,200.16,29.966,46.78,73.33,50.25,18.36,,21.03,61.42,12
.15,42.17,,34.03,52.95,190.5065,62.8,49.49,27.43,68.32,32.178,47.84,75.41,21.71,
48.66,55.37,9.48,,21.43,56.38,90.4,37.65,39.61,57.0,38.61,14.195,46.11,95.36,59.
12,45.26,29.58,38.5,90.06,23.31,78.12,53.9,87.53,29.53,23.255,39.06,42.71,26.58,
11.65,60.54,43.51,67.26,61.52,41.22,17.79,40.27,52.085,95.17,36.44,103.53,45.49,
27.75,47.22,36.9,13.05,104.84,63.94,98.07,36.35,102.62,15.8167,102.12,61.14,34.7
5,40.845,41.38,34.31,27.86,60.9,23.28,104.24,219.37,7.91,28.85,40.24,,56.375,27.
68,30.38,72.18,44.49,25.4128,29.45,26.94,27.61,32.67,66.12,68.26,24.12,70.57,35.
36,52.69,46.87,12.51,23.94,,,47.8,55.36,34.96,102.4,87.21,43.63,33.32,12.59,47.7
9,42.69,696.11,37.07,31.54,72.36,27.14,31.06,23.56,75.81,19.3,94.39,40.22,34.62,
39.33,90.35,64.0,51.22,54.13,69.675,30.39,111.4,57.21,156.85,64.18,119.5,28.19,1
28.55,110.99,,67.18,,36.23,170.35,51.2,120.31,7.94,35.36,54.51,45.32,177.01,43.6
,90.93,119.12,30.43,70.68,31.27,54.55,69.28,28.07,47.47,16.73,19.11,164.5,62.74,
88.75,77.93,81.79,78.39,61.01,34.15,44.06,44.28,163.5,96.25,75.76,29.35,56.6,35.
05,31.32,24.25,75.93,,63.9,22.64,31.56,44.05,145.18,40.44,62.67,63.1,45.4,37.213
4,74.75,48.93,46.23,73.13,79.21,51.6,23.64,23.01,52.13,33.69,29.56,35.23,12.455,
26.51,,23.81,57.38,99.19,57.12,24.17,66.71,82.69,53.32,34.09,89.61,71.11,36.2119
,59.31,46.14,55.65,84.8,54.62,45.66,45.7,67.7,44.32,39.0825,92.47,41.48,48.6,40.
14,35.26,106.47,,35.74,71.4,36.31,,14.25,124.1,59.62,30.14,66.75,27.94,37.46,28.
31,88.28,42.84,31.96,27.46,64.55,75.65,24.21,33.26
2013-02-12,14.27,66.8428,78.6,35.42,46.96,34.3,73.37,38.89,46.27,30.81,60.36,38.
91,152.32,33.15,44.9,11.31,50.15,49.5,85.78,38.63,29.16,41.05,37.82,36.74,62.7,3
3.02,24.43,,45.61,90.21,13.66,2.77,41.82,84.54,145.75,67.49,75.59,258.7,53.49,75
.74,66.01,56.7,17.6325,84.18,83.45,88.83,35.2,,72.79,27.09,13.51,130.83,35.14,39
.43,39.22,62.2,70.01,382.09,44.62,12.245,68.68,75.99,30.6,15.43,88.55,47.4166,32
.1,,,165.45,27.9,238.86,22.445,36.52,97.7,7.59,37.705,107.03,33.3,45.54,97.22,24
.87,24.26,34.96,43.21,86.82,68.31,39.02,14.36,49.48,43.775,,44.068,29.99,20.01,5
9.13,80.0,44.62,61.98,80.59,54.025,35.0,19.485,57.2,322.51,119.19,25.84,22.39,21
.05,55.78,27.35,59.49,104.61,58.25,101.76,,38.26,43.09,20.97,,22.64,43.49,41.76,
38.3,73.59,51.17,116.5,96.16,44.35,14.5,93.97,39.48,56.88,45.04,24.05,60.97,71.0
3,64.22,37.66,54.95,65.69,40.23,71.46,45.32,15.92,46.81,64.51,69.5,59.215,59.53,
,,54.63,17.54,56.78,74.46,56.56,54.32,48.0,62.67,73.93,58.08,67.05,214.53,57.49,
60.68,54.34,152.65,41.15,11.31,60.21,63.93,,43.71,31.95,42.75,65.55,40.35,51.0,3
4.6,27.37,35.67,106.12,40.5,102.15,40.27,36.5,16.56,26.69,63.57,53.0299,34.72,58

.13,,,108.95,48.06,,13.08,66.68,22.58,20.15,40.83,42.97,12.57,28.55,390.7403,,70
.19,24.785,32.11,37.68,154.11,13.86,223.09,40.71,39.89,7.22,10.035,36.45,62.7,47
.08,67.32,66.98,24.4,45.25,,53.28,22.33,70.42,,,65.42,24.74,17.815,46.75,45.185,
17.18,80.77,79.2,200.04,30.142,46.465,73.44,50.27,18.42,,21.19,61.44,12.24,42.01
,,34.47,53.05,190.4618,63.02,49.37,27.71,67.13,32.4712,48.14,75.8,21.61,49.14,54
.98,9.53,,21.51,56.71,89.9,37.74,39.62,62.0,37.56,14.27,46.06,95.87,59.25,44.84,
29.9,40.0686,89.41,23.2,78.36,53.8,87.26,29.89,23.32,39.79,42.84,26.87,11.55,60.
51,43.65,67.4,61.87,41.39,20.01,40.67,51.964,95.1,36.62,104.83,46.09,27.68,47.11
,37.13,13.32,106.38,64.09,98.43,36.02,103.46,15.5867,101.77,61.49,34.66,40.445,4
1.45,34.61,27.88,60.83,23.63,105.45,217.99,7.92,28.8,40.11,,56.11,28.32,30.55,72
.49,45.28,25.4214,29.43,27.03,27.545,33.26,66.34,68.92,24.24,71.64,35.41,52.93,4
7.24,12.43,24.19,,,48.17,56.02,35.11,101.56,86.93,44.5,33.44,12.78,47.93,42.72,7
03.38,37.34,31.51,72.17,26.99,31.18,23.75,75.98,20.32,95.18,40.08,34.52,39.6,90.
14,64.77,52.03,54.49,68.225,30.52,111.55,57.0,157.13,64.43,120.72,28.61,128.5,11
0.98,,65.885,,36.44,168.18,51.55,121.29,7.93,35.31,55.33,45.41,176.79,43.65,91.3
4,119.57,30.33,71.03,31.19,54.36,68.271,28.13,47.86,16.86,18.95,164.67,62.49,90.
92,78.53,82.79,78.79,61.04,34.44,44.16,44.03,162.97,96.34,75.83,29.42,56.93,34.6
8,31.49,24.2,77.49,,64.01,22.7,31.56,44.22,144.91,40.49,62.73,63.6,45.2,37.3334,
74.58,49.01,46.66,73.75,79.9,51.48,23.51,23.0,52.5,33.56,29.14,35.6,12.43,25.89,
,24.17,56.3,98.71,57.01,24.17,67.335,82.31,54.3,34.31,89.98,71.36,36.3524,59.12,
45.65,56.12,85.57,54.23,45.53,45.89,68.16,44.44,39.2,91.99,41.95,48.18,39.93,35.
51,110.81,,35.84,71.4,36.46,,14.34,122.67,59.62,30.31,67.0,28.0,37.58,28.41,88.4
6,42.87,31.84,27.95,64.75,75.44,24.49,33.74
2013-02-13,14.66,66.7156,78.97,35.27,46.64,34.46,73.56,38.81,46.26,31.16,60.45,3
8.66,152.32,33.28,44.93,11.34,49.51,48.65,86.73,38.87,28.9,41.39,38.54,37.31,66.
15,32.75,24.76,,45.88,92.01,13.77,2.75,42.0,84.94,146.47,67.89,74.0,269.47,54.07
,75.89,63.0,57.28,17.7625,84.33,84.13,88.78,35.055,,72.36,27.3,13.73,130.23,35.1
7,39.61,39.41,62.1,71.14,379.73,44.75,12.17,68.15,74.78,30.52,15.12,88.47,47.516
6,32.055,,,165.91,28.17,242.52,22.6,36.12,97.97,7.45,37.585,107.18,33.48,45.44,9
6.38,25.0,24.5,34.99,42.67,86.66,68.04,37.46,14.25,49.535,43.81,,43.578,29.81,20
.57,58.81,80.03,44.71,61.89,81.04,53.95,35.1,20.065,57.26,317.01,119.04,25.86,22
.58,20.78,55.07,27.73,59.14,104.25,58.61,101.63,,38.18,43.075,21.14,,22.87,43.38
,41.69,38.19,72.39,51.06,115.53,93.96,44.0,14.78,90.68,39.22,56.97,44.39,23.83,6
1.02,71.52,64.31,37.28,54.96,65.5,39.72,71.61,42.69,15.96,45.86,64.4,68.82,59.37
,61.48,,,54.69,17.17,57.05,74.67,56.62,54.01,47.64,63.3,74.78,58.1,66.8,213.02,5
7.65,60.71,55.43,152.85,41.09,11.3,61.04,63.66,,43.53,31.59,42.67,65.47,40.06,52
.005,34.62,27.9075,35.64,106.16,40.68,102.28,40.275,37.2,16.11,27.0,64.05,53.339
9,33.44,58.81,,,109.18,50.46,,13.04,66.11,23.39,19.91,41.4,42.98,12.78,28.67,391
.8214,,70.54,24.9,31.84,38.56,154.52,14.08,225.62,40.84,40.42,7.13,10.1025,36.89
,62.74,47.26,67.45,67.96,24.45,45.55,,53.63,22.05,70.18,,,65.68,24.84,17.92,47.1
,44.625,17.15,81.35,78.09,200.09,30.362,46.04,74.17,49.76,18.65,,21.25,61.44,12.
4,41.75,,34.51,53.19,191.2331,63.55,48.9,27.63,66.92,32.9843,48.69,75.66,21.67,4
8.68,55.22,9.45,,21.54,56.61,89.89,37.8,39.4,61.35,37.21,14.23,46.6,97.36,59.29,
44.72,30.24,40.0294,89.7,23.26,78.07,53.44,87.06,30.16,23.305,39.77,42.93,27.05,
11.61,62.3,44.08,67.41,61.88,41.18,20.36,40.99,51.921,94.0,36.77,104.91,46.79,27
.75,46.99,37.5,13.34,107.57,64.09,99.53,36.3,102.86,15.8967,102.81,62.2,34.78,40
.835,41.15,34.96,28.03,60.81,23.63,104.67,218.94,7.95,28.86,39.96,,56.135,28.92,
31.65,72.55,44.85,26.6098,29.79,27.18,27.28,33.72,65.76,68.96,24.12,71.88,35.82,

53.22,47.47,12.37,24.03,,,47.95,56.7,34.99,102.0,86.94,44.55,33.53,12.86,48.04,4
2.61,704.17,37.15,31.5,71.5,27.0,31.46,24.49,76.56,20.23,96.33,40.28,34.57,39.47
,90.99,64.17,52.98,54.56,69.555,30.47,111.72,57.33,156.08,64.11,120.49,28.99,126
.76,111.77,,65.45,,35.61,170.33,51.57,120.78,7.85,35.2,54.5,45.58,175.63,43.54,9
1.45,119.78,30.22,71.31,31.22,53.97,67.45,27.915,47.89,16.92,18.98,165.45,62.5,9
0.33,78.63,82.76,79.04,62.04,34.24,44.17,44.33,162.05,96.13,76.33,28.94,57.07,35
.01,31.88,24.3,77.48,,63.88,22.62,31.78,44.62,144.75,40.92,62.69,63.45,45.01,37.
42,75.1,48.77,46.9,74.21,79.99,51.55,23.86,23.01,52.85,33.72,29.4,35.42,12.175,2
6.19,,24.3,57.28,99.49,57.23,24.49,67.645,82.5,55.49,33.93,89.5,71.5,36.2667,59.
55,46.24,55.86,85.59,54.57,45.58,46.48,68.11,44.52,38.6925,93.22,41.48,48.63,39.
84,35.13,110.98,,35.64,71.39,36.64,,14.17,122.4,59.05,30.48,67.75,27.92,37.8,28.
42,88.67,43.08,32.0,28.26,64.41,76.0,24.74,33.55
2013-02-14,13.99,66.6556,78.84,36.57,46.77,34.7,73.13,38.61,46.54,31.4,60.74,38.
67,152.37,33.14,44.77,11.21,49.56,48.83,86.49,39.21,28.97,42.01,38.82,37.87,65.4
1,32.56,24.355,,45.88,87.63,13.89,2.75,41.91,84.72,147.34,69.04,73.48,269.24,55.
18,75.51,63.29,57.3,17.8525,80.33,84.0,88.69,35.05,,71.56,27.87,14.0,130.05,35.6
4,39.69,39.15,62.34,71.42,373.61,44.58,12.13,68.17,74.93,30.36,15.71,88.39,48.14
33,32.165,,,164.59,28.1,245.57,22.605,36.58,99.21,7.53,37.57,107.35,33.78,46.0,9
6.07,25.19,24.7,35.69,42.94,86.44,66.66,37.35,14.2,49.755,43.95,,43.572,29.64,20
.4,58.08,79.48,44.69,61.45,81.39,54.165,35.32,20.17,58.44,317.72,118.73,25.75,22
.755,20.43,55.59,27.48,59.13,103.79,57.92,102.02,,38.72,43.615,20.99,,22.93,43.6
7,32.27,38.09,72.01,51.89,115.71,96.45,44.32,14.24,90.57,39.52,56.53,45.03,23.78
,61.16,70.84,63.76,36.84,54.88,65.37,39.93,71.86,42.8,15.91,45.52,64.36,68.0,59.
605,60.66,,,54.47,17.15,56.83,74.82,56.36,54.25,46.18,63.54,74.08,58.41,66.665,2
26.0,57.56,60.5,55.1,152.49,40.76,11.42,61.24,61.8,,43.175,30.71,42.4,65.17,39.5
4,52.49,34.22,28.5,35.53,106.56,40.49,100.89,40.08,36.98,16.13,26.7,64.57,53.323
3,33.29,59.01,,,108.53,51.52,,13.11,66.22,23.41,19.8,41.6,44.31,12.95,27.75,394.
3039,,70.79,24.96,31.36,38.97,155.93,14.17,226.29,43.32,40.88,7.17,10.105,36.83,
62.45,47.15,67.34,68.35,24.49,45.2,,53.45,22.1,70.09,,,67.86,24.86,17.955,47.74,
44.2,17.15,80.89,77.83,199.65,31.104,45.84,73.8,48.79,20.56,,21.23,60.76,12.49,4
1.8,,34.56,53.03,190.4198,63.7,48.53,27.78,67.41,33.3717,48.9,75.81,21.68,49.22,
55.25,9.48,,21.54,56.1,90.19,37.57,39.72,61.67,36.84,14.12,46.98,97.61,59.58,44.
05,30.6,39.8431,89.19,23.08,77.74,53.5,87.08,30.45,23.5,39.57,42.9,27.57,11.47,6
1.17,44.08,67.56,61.62,41.16,20.11,41.15,52.216,93.56,37.32,104.21,46.68,26.57,4
7.17,36.69,13.29,107.05,64.39,100.35,36.07,102.78,15.8967,102.0,61.98,34.11,41.5
8,41.19,35.34,28.04,61.12,23.83,104.6,217.75,8.08,29.16,39.54,,56.075,29.25,30.6
,72.2,44.55,26.7714,28.11,26.91,27.31,32.55,65.8,70.29,23.89,71.97,35.86,53.09,4
8.23,12.73,24.28,,,47.61,57.35,34.9,101.68,86.91,44.29,33.72,12.91,48.17,42.43,7
05.62,37.08,31.12,72.28,27.06,31.62,24.0,76.78,20.27,96.31,40.36,34.53,39.21,90.
01,63.87,52.83,54.33,69.5,30.53,113.16,57.06,154.2,64.83,120.3,29.11,130.05,111.
21,,65.53,,35.66,169.24,51.59,120.45,7.91,35.2,54.07,45.6,174.86,43.18,90.85,119
.86,30.19,71.73,31.3,53.76,68.11,27.775,47.86,16.92,18.97,163.67,62.66,92.16,81.
56,82.15,79.17,62.37,34.33,43.91,44.86,160.57,96.44,75.35,28.79,57.2,34.95,43.75
,24.97,77.81,,64.02,22.67,32.03,43.97,143.44,41.08,63.09,63.7,45.21,37.32,74.88,
48.2,43.55,73.84,80.34,51.875,24.22,23.03,53.63,33.79,29.48,35.29,12.225,25.87,,
24.32,57.91,87.8,57.05,24.4,68.085,82.69,55.9,33.99,89.69,70.96,36.3833,59.95,47
.23,54.87,85.04,54.27,45.67,45.66,67.78,44.33,38.995,93.47,41.86,48.81,39.45,35.
21,111.0,,35.46,70.82,37.14,,14.14,123.2,58.83,30.44,67.25,27.89,38.44,28.22,88.

52,42.91,32.12,28.47,63.89,76.34,24.63,33.27
2013-02-15,14.5,65.7371,79.0,37.58,46.6,35.08,74.16,38.635,46.175,32.57,61.22,39
.0,155.41,33.13,45.13,11.17,49.2,49.34,85.24,38.35,29.12,41.84,38.87,38.49,65.42
,32.73,24.49,,45.85,86.01,13.74,2.71,41.91,83.61,146.51,68.77,75.18,265.09,54.64
,75.62,62.7,57.0,17.9925,76.85,82.08,88.23,35.21,,71.62,27.96,14.37,130.26,35.14
,39.73,39.17,61.69,71.61,373.7,42.25,12.03,68.17,75.03,30.27,16.87,88.49,48.1866
,32.62,,,168.74,27.99,242.95,22.77,36.58,99.77,7.54,38.055,107.21,33.73,46.7,95.
61,25.05,24.28,35.96,44.64,86.5,67.9,36.92,14.27,49.385,43.965,,43.334,29.94,20.
01,57.78,79.54,44.84,61.08,81.75,55.01,34.83,20.62,58.52,314.19,119.21,25.77,22.
98,20.41,54.03,27.155,59.1,104.01,57.02,102.17,,39.4,43.7575,20.99,,22.79,44.66,
33.02,38.425,72.62,51.12,114.96,96.44,43.84,14.45,89.75,39.15,57.32,44.37,23.66,
61.68,70.3,63.01,36.93,55.61,63.65,41.12,72.56,42.91,15.83,45.15,64.42,68.06,58.
73,59.25,,,54.92,16.97,56.7,75.21,56.58,55.17,46.68,63.11,73.46,58.29,64.595,224
.53,57.51,59.37,55.57,151.69,40.94,11.22,60.87,61.45,,43.18,30.62,42.26,64.58,38
.7,52.76,34.03,28.32,35.04,106.42,40.57,101.72,40.435,36.49,16.045,26.74,64.51,5
3.3166,33.51,59.47,,,108.53,51.8,,13.02,66.4,23.29,19.96,41.6,44.59,12.79,27.76,
396.8414,,71.44,25.065,32.88,39.14,154.99,14.1,229.48,42.7,40.53,7.1,9.99,36.28,
62.96,47.8,67.52,67.42,24.2,45.35,,52.66,21.935,70.11,,,67.83,25.1,18.04,47.61,4
4.34,17.17,80.93,77.99,200.98,31.292,45.6,73.88,48.73,22.0,,21.115,62.07,12.57,4
2.17,,34.71,53.03,189.7365,63.25,48.91,27.36,66.8,33.5602,49.35,76.16,21.86,48.8
8,55.4,9.48,,21.71,55.76,91.2,37.52,39.36,63.27,37.42,13.945,46.34,97.77,59.77,4
4.06,30.7,39.4118,89.84,23.25,77.67,53.59,87.87,30.33,23.47,39.14,42.5,27.06,11.
6,60.4,43.91,67.85,61.53,41.23,20.0,40.849,52.313,93.9,37.075,103.42,46.8,26.72,
47.12,36.44,12.88,107.8,64.69,101.09,36.12,103.23,17.0266,103.51,62.18,34.38,41.
28,41.42,34.66,28.01,61.43,23.87,104.83,215.14,7.91,29.22,39.12,,56.01,30.27,31.
07,72.5,43.27,27.0731,27.15,26.78,27.475,32.4,65.63,69.33,23.94,71.5,36.55,53.47
,48.12,12.731,24.18,,,47.75,57.71,34.81,101.31,84.71,44.46,33.97,12.95,48.09,42.
72,702.26,37.22,31.09,73.68,27.29,31.69,24.19,76.54,20.31,96.04,40.64,34.13,38.8
4,89.99,63.87,52.91,54.34,69.695,30.41,113.58,57.28,151.26,64.17,120.1,28.74,130
.56,110.15,,65.43,,35.67,164.93,51.7,121.05,7.85,35.52,53.93,45.4,176.07,42.66,9
0.76,120.04,29.955,70.36,31.31,53.98,68.69,27.17,47.92,16.9,19.4,166.7,62.09,92.
4,80.03,82.7,79.43,62.16,34.54,44.11,44.95,159.9,97.12,75.62,28.12,57.24,34.445,
43.39,24.5,78.46,,63.85,22.53,32.07,43.53,143.77,41.03,61.71,64.78,45.01,37.4734
,74.43,48.4,44.415,73.13,80.39,51.885,24.01,23.27,53.52,33.585,29.39,35.36,12.38
75,26.37,,24.45,57.42,87.96,57.32,24.46,68.415,83.48,54.87,33.91,90.78,70.05,37.
5905,59.46,47.29,55.03,84.89,54.88,46.14,45.61,69.47,44.4,39.4975,92.61,41.15,48
.59,39.78,35.16,111.33,,35.11,69.3,36.69,,14.4,122.96,59.41,30.65,64.95,28.11,38
.12,28.67,88.36,42.8,31.88,28.28,63.99,75.9,24.34,33.98
2013-02-16,,,
,,
,,
,,
,,
,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

[29]:
```
'''
So for this data frame, we want to make sure that the first column in CSV gets␣
 ↪assigned the date index.

Otherwise what would happen is the index would just be an integer like 0,1,2␣
 ↪and so forth.
So the date would become a regular data column. We don't want that.

So we specify index_col=0.

We also want these dates to be converted to dates and not to be treated like␣
 ↪strings, so we pass in
the argument parse_dates=True.
'''
close2 = pd.read_csv('sp500_close.csv', index_col=0, parse_dates=True)
close2.head()
```

[29]:

	AAL	AAPL	AAP	ABBV	ABC	ABT	ACN	ADBE	ADI	\
2013-02-08	14.75	67.8542	78.90	36.25	46.89	34.41	73.31	39.12	45.70	
2013-02-09	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2013-02-10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2013-02-11	14.46	68.5614	78.39	35.85	46.76	34.26	73.07	38.64	46.08	
2013-02-12	14.27	66.8428	78.60	35.42	46.96	34.30	73.37	38.89	46.27	

	ADM	…	XLNX	XL	XOM	XRAY	XRX	XYL	YUM	\
2013-02-08	30.22	…	37.51	28.24	88.61	42.87	31.84	27.09	65.30	
2013-02-09	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2013-02-10	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2013-02-11	30.28	…	37.46	28.31	88.28	42.84	31.96	27.46	64.55	
2013-02-12	30.81	…	37.58	28.41	88.46	42.87	31.84	27.95	64.75	

	ZBH	ZION	ZTS
2013-02-08	75.85	24.14	33.05
2013-02-09	NaN	NaN	NaN
2013-02-10	NaN	NaN	NaN
2013-02-11	75.65	24.21	33.26
2013-02-12	75.44	24.49	33.74

[5 rows x 505 columns]

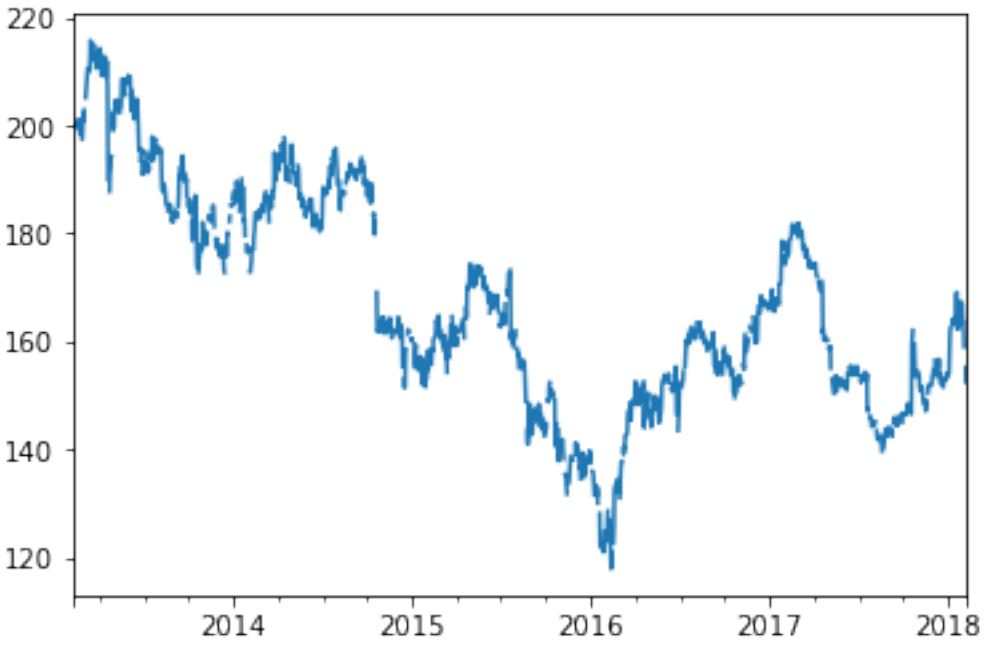
[30]:
```
'''
It has a date time index with 1826 entries with a frequency D, which
means daily.

It has 505 columns from AAL to ZTS.

The type is Flow 64, which makes sense.
```

```
'''
close2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1826 entries, 2013-02-08 to 2018-02-07
Columns: 505 entries, AAL to ZTS
dtypes: float64(505)
memory usage: 7.0 MB
```

[31]:
```
# Now it has dates
# Also has missing values
'''
So first, we're going to start by using our clothes Price's data set and␣
 ↪calling the plot function
directly on the series given by the IBM column.

Note that there is some missing data here, as you can see from the gaps in the␣
 ↪line chart.

However, recall that this may be because of how we created our data set.
We created the data set by creating a date range, which included all the days␣
 ↪between a specified start
date and end date.

Of course, this necessarily includes weekends and holidays.
'''
close_prices['IBM'].plot()
```

[31]:
```
<AxesSubplot:>
```

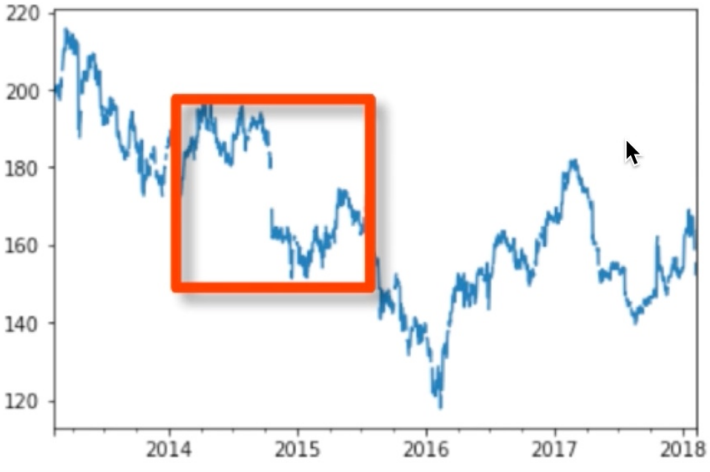


[37]:
```
Image(filename='/Users/subhasish/GIT/Interstellar/SB-AI-DEV/ML/SB/TimeSeries/
↪Lazy Programmers/Image/2021-09-13_20-11-16.jpg')
```

[37]:

```
[ ]  # Now it has dates
     # Also has missing values
     close_prices['IBM'].plot()
```

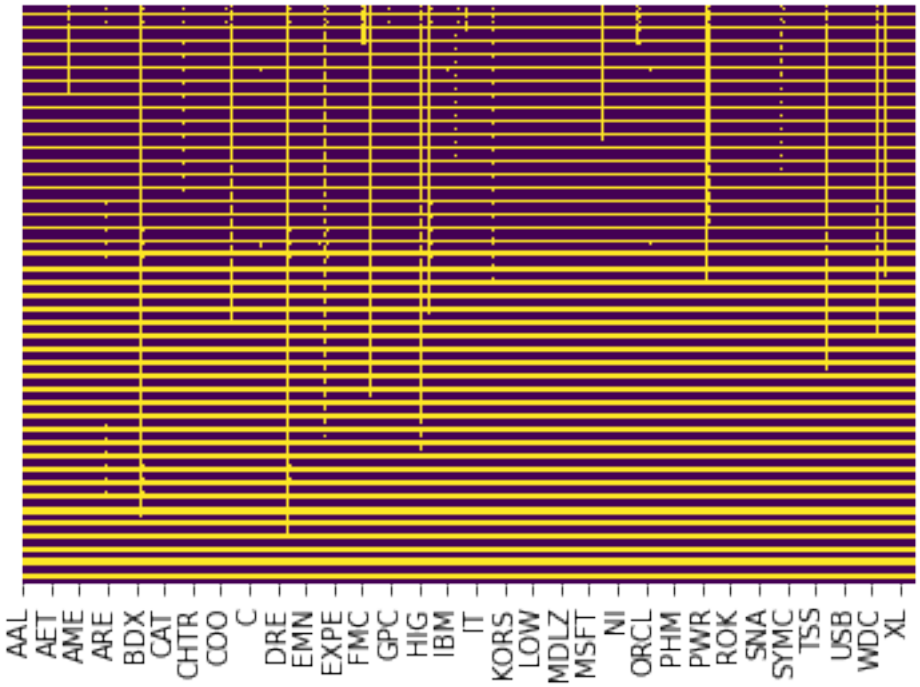
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb536261588>
```



[ ]:

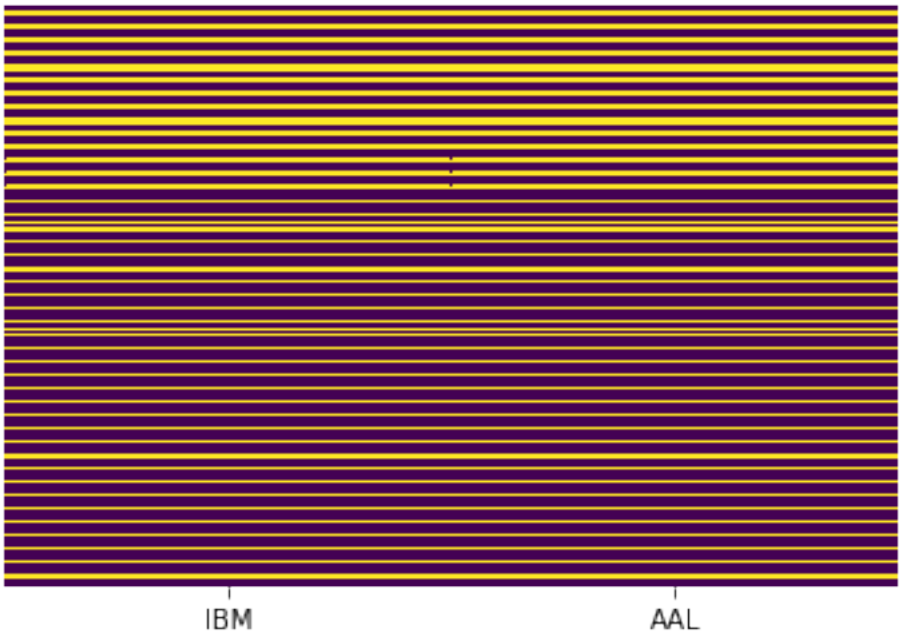
[33]:
```
'''
#I can actually make a heat map so I'm going to say this and I say sns.
 ↪heatMap()and then I'm

#yticklabels=False't get a bunch of tick labels.

#cbar =False because we're not doing an actual color bar.

#cmap='viridis' to chnage the color.


#I've created a heat map of those boolean values those true and false␣
 ↪statements and due to this.
#map does color mapping every yellow dash here basically stands for a true␣
 ↪point where true it was null
#So we can just glimps now order data from a very far bird's eye view and check␣
 ↪out that yes we're missing
'''
import seaborn as sns
sns.heatmap(close_prices.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

[33]: <AxesSubplot:>



[36]:
```
import seaborn as sns
sns.heatmap(close_prices[["IBM","AAL"]].
 ↪isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

[36]: <AxesSubplot:>

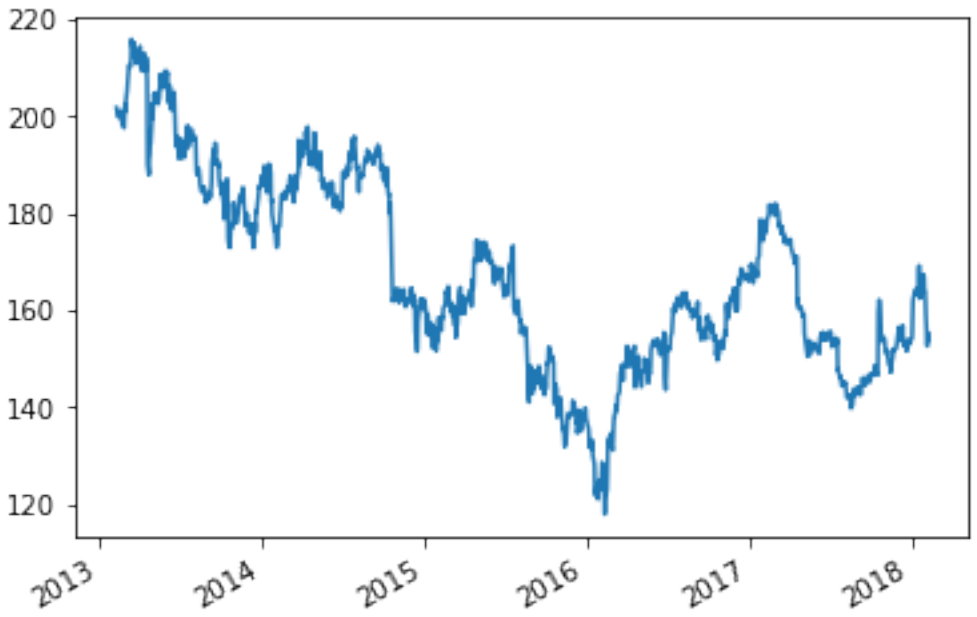


[38]:
```
'''
The next thing I want to show you how to do is how to get rid of all the rows␣
 ↪in which the entire row
is missing.

These are days which likely correspond to non-trading days.
And even if they are not non-trading days, we can't do anything with these days␣
 ↪anyway.

As you may know, this can be accomplished by using the dropna(axis=0,␣
 ↪how='all', inplace=True) function.
But if you just call drop in a without any arguments, it will not do what you␣
 ↪want.

Drop in a will drop any row with any missing data by default.
We only want to drop the rows where all the data is missing.

So first we specify axis=0, which means drop rows and not columns.
We say how='all' so that only rows in which all data is missing are dropped.
```

```
Finally, we say inplace=True so that we do these operations on the existing␣
 ↪data frame.
'''
# drop rows with all nan
# most likely correspond to weekends, holidays (non-trading days)
close_prices.dropna(axis=0, how='all', inplace=True)
```

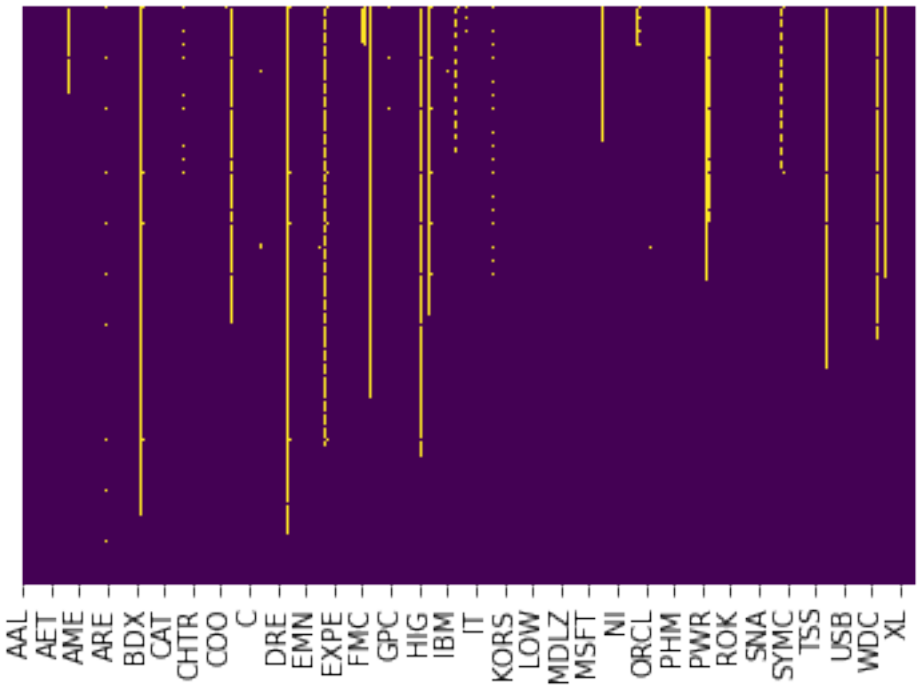
[39]:
```
'''
Next, we plot the IBM close prices again, and at least the visible gaps which␣
 ↪were pretty evident,
are now gone.
'''
close_prices['IBM'].plot()
```

[39]: <AxesSubplot:>



[40]:
```
sns.heatmap(close_prices.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

[40]: <AxesSubplot:>



[41]:
```
'''
We can check how many missing values are still in the data set by using isna()␣
 ↪function, this returns
a boolean in every location.

True, if the cell is isna() and false, otherwise

recall that in Python a true equals one and false
equals zero.
So if we simply sum all the values we should get how many missing values there␣
 ↪are.
'''
close_prices.isna().sum()
```

[41]:
```
AAL     0
AAPL    0
AAP     0
ABBV    0
ABC     0
       ..
XYL     0
YUM     0
ZBH     0
ZION    0
```

```
ZTS     0
Length: 505, dtype: int64
```

[42]:
```
'''
If we call sum() once, you will notice that it only comes along one axis at a␣
 ↪time.
So now we have the number of missing values for each individual stock.

Since there are too many stocks to look at all at once, the next thing we can␣
 ↪do is simply sum() the
result.

So in effect, we call some twice and that tells us how many missing values␣
 ↪there are in total.
As you can see, there are about 16755 missing values.

'''
close_prices.isna().sum().sum()
```

[42]:
```
16755
```

[43]:
```
'''
Now, we know that any stock for which there is no initial value must be␣
 ↪backwards filled.
So let's try to figure out how many stocks fit that scenario.
'''

'''
We can do so by checking how many stocks have a missing value in the first row␣
 ↪of the data frame that
can be accomplished by using the iloc[] function.

We call iloc[0, :] to get the first row of the data frame.
We call isna() to get a Boolean for whether the data is missing or not.
And finally, we call some to get the total number of missing values in this row.

As you can see, there are 29 stocks for which there is no initial value and␣
 ↪these are the
ones which must be backward filled.
'''
close_prices.iloc[0, :].isna().sum()
```

[43]:
```
29
```

[44]:
```
'''
As mentioned before, we do any backwards filling, we must forward fill, we can␣
 ↪do that by calling
```

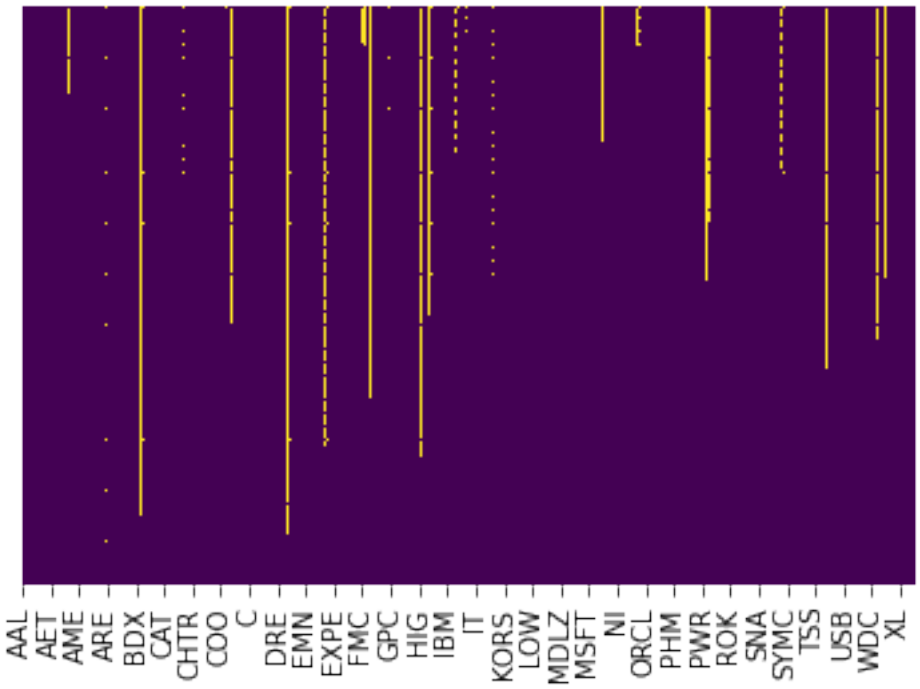
```
the fill in a function saying method equals ffill-> fillna(method='ffill',␣
↪inplace=True).

direct pandas to use forward filling and inplace=True means the operation is␣
↪committed on our
existing data frame.


'''
close_prices.fillna(method='ffill', inplace=True)
```

[45]:
```
'''
After we do this, we should check again to see how many missing values there␣
↪are now.

Interestingly, we see that a majority of the missing values from before are␣
↪still missing.

This indicates that many of the companies in our list are for some reason, are␣
↪missing a lot of data
from the start of this data said.
'''
close_prices.isna().sum().sum()
```

[45]: 16704

[46]:
```
sns.heatmap(close_prices.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

[46]: <AxesSubplot:>



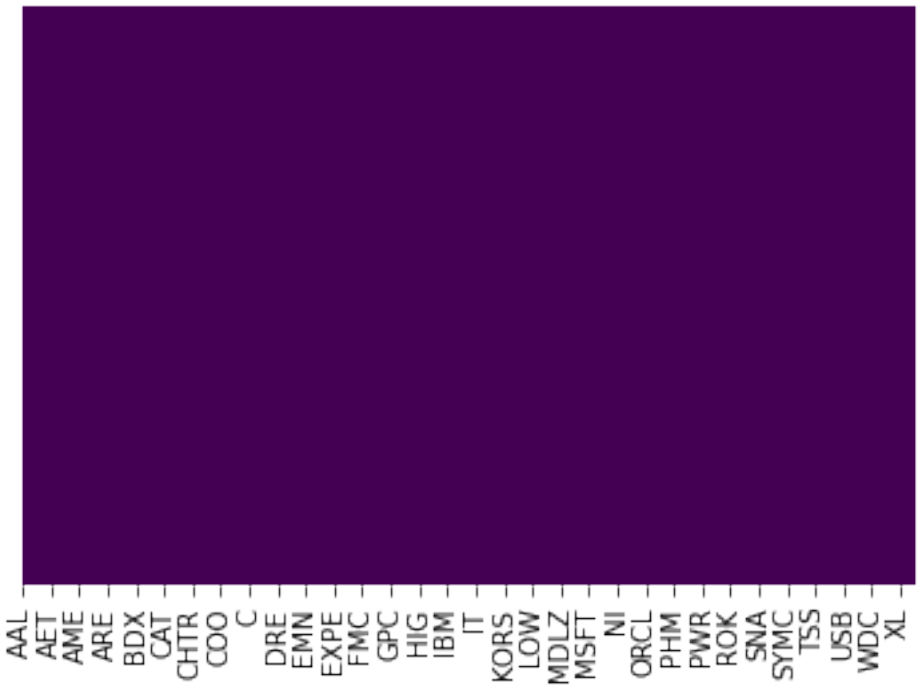
[47]:
```
'''
So how do we backfill?

Well, we call the exact same function fillna(method='bfill', inplace=True),␣
 ↪except now we pass in method=bfill.

'''

close_prices.fillna(method='bfill', inplace=True)
```

[48]:
```
'''
After this, when we check the number of missing values, we see that it is now␣
 ↪zero as expected.
'''
close_prices.isna().sum().sum()
```

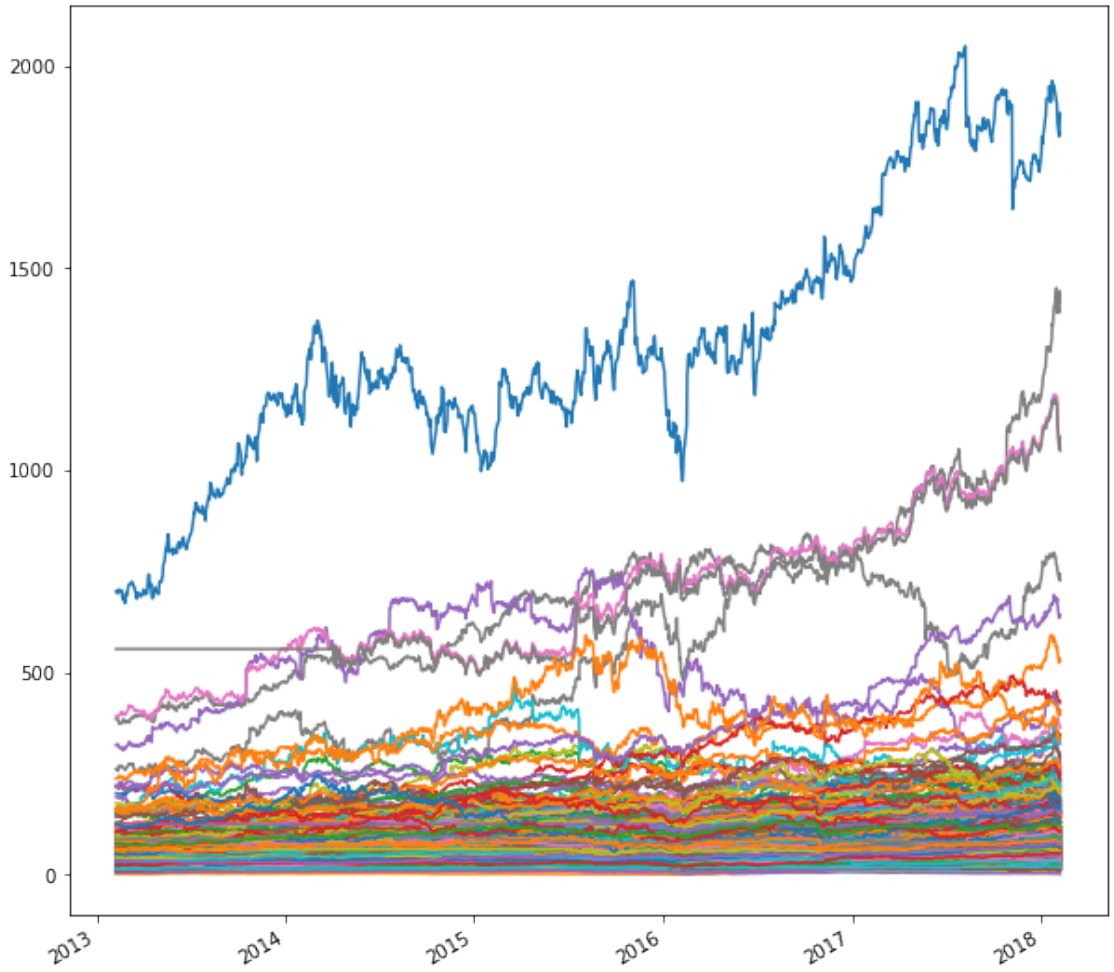
[48]: 0

[49]:
```
sns.heatmap(close_prices.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

[49]: <AxesSubplot:>



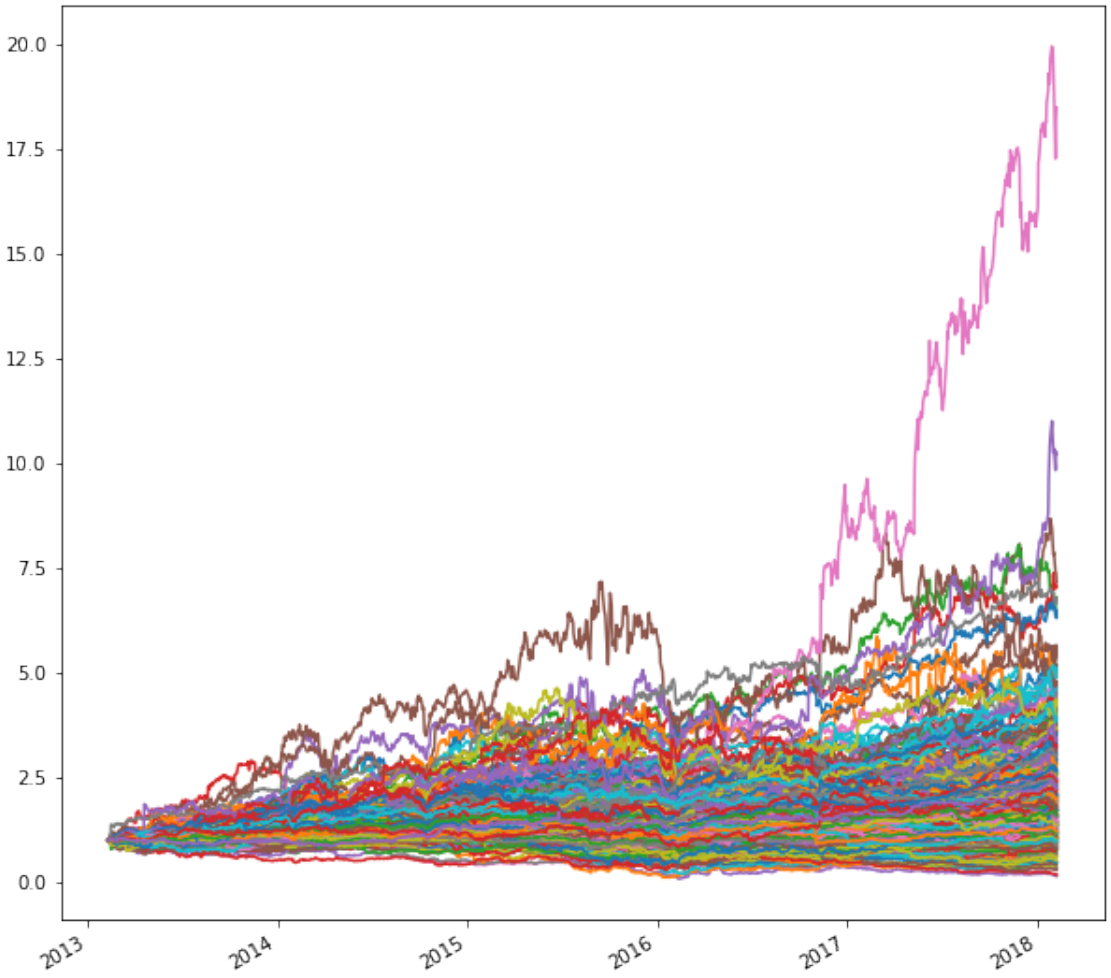
[50]:
```
'''
The next thing I'd like to do is plot all of our stock prices on the same chart␣
↪for this plot.

I'm going to say legend equals false, since showing 500 items in the legend is␣
↪a bit too much.
I'm also going to set fixes equals 10 10 so that we can see the plot more␣
↪easily.
'''
close_prices.plot(legend=False, figsize=(10, 10));

'''
Well, they look like a bunch of stocks.

There is at least one stock where we can see this backwards Phil behavior.

So it's this gray one over here.

As you can see, there is over a year's worth of missing data for this stock.

'''
```



[51]:
```
'''
One common operation and finance is to plot the relative stock price over time.

This allows us to more easily compare growth or lack thereof of different␣
 ↪stocks.

We can accomplish this by dividing each column of stocks by its initial value.

Of course, the initial value divided by the initial value will become 1, and␣
 ↪then each subsequent
value will be the ratio between the current value and the initial value.

This allows us to easily see the cumulative return on each stock, which we will␣
 ↪define more rigorously
in another lecture.
```

```
'''
close_prices_normalized = close_prices / close_prices.iloc[0]
```

[52]:
```
close_prices_normalized.plot(legend=False, figsize=(10, 10));
```



[ ]:
```

```