EDA Time Series

In [2]:
```
# Install Pandas Data Reader
!pip install pandas-datareader
```

```
Collecting pandas-datareader
  Downloading pandas_datareader-0.10.0-py3-none-any.whl (109 kB)
     ------------------------------------ 109.5/109.5 kB 3.2 MB/s eta 0:00:00
Requirement already satisfied: requests>=2.19.0 in c:\users\abhin\anaconda3\lib\site-packages
(from pandas-datareader) (2.28.1)
Requirement already satisfied: pandas>=0.23 in c:\users\abhin\anaconda3\lib\site-packages (from
pandas-datareader) (1.4.4)
Requirement already satisfied: lxml in c:\users\abhin\anaconda3\lib\site-packages (from pandas-
datareader) (4.9.1)
Requirement already satisfied: numpy>=1.18.5 in c:\users\abhin\anaconda3\lib\site-packages (fro
m pandas>=0.23->pandas-datareader) (1.21.5)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\abhin\anaconda3\lib\site-pack
ages (from pandas>=0.23->pandas-datareader) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\abhin\anaconda3\lib\site-packages (from
pandas>=0.23->pandas-datareader) (2022.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\abhin\anaconda3\lib\site-packages
(from requests>=2.19.0->pandas-datareader) (2022.9.14)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\abhin\anaconda3\lib\site-pa
ckages (from requests>=2.19.0->pandas-datareader) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\abhin\anaconda3\lib\site-packages (from
requests>=2.19.0->pandas-datareader) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\abhin\anaconda3\lib\site-packa
ges (from requests>=2.19.0->pandas-datareader) (1.26.11)
Requirement already satisfied: six>=1.5 in c:\users\abhin\anaconda3\lib\site-packages (from pyt
hon-dateutil>=2.8.1->pandas>=0.23->pandas-datareader) (1.16.0)
Installing collected packages: pandas-datareader
Successfully installed pandas-datareader-0.10.0
```

In [22]:
```
!pip install --upgrade pandas-datareader
```

```
Requirement already satisfied: pandas-datareader in c:\users\abhin\anaconda3\lib\site-packages
(0.10.0)
Requirement already satisfied: pandas>=0.23 in c:\users\abhin\anaconda3\lib\site-packages (from
pandas-datareader) (1.4.4)
Requirement already satisfied: requests>=2.19.0 in c:\users\abhin\anaconda3\lib\site-packages
(from pandas-datareader) (2.28.1)
Requirement already satisfied: lxml in c:\users\abhin\anaconda3\lib\site-packages (from pandas-
datareader) (4.9.1)
Requirement already satisfied: numpy>=1.18.5 in c:\users\abhin\anaconda3\lib\site-packages (fro
m pandas>=0.23->pandas-datareader) (1.21.5)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\abhin\anaconda3\lib\site-pack
ages (from pandas>=0.23->pandas-datareader) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\abhin\anaconda3\lib\site-packages (from
pandas>=0.23->pandas-datareader) (2022.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\abhin\anaconda3\lib\site-packages
(from requests>=2.19.0->pandas-datareader) (2022.9.14)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\abhin\anaconda3\lib\site-pa
ckages (from requests>=2.19.0->pandas-datareader) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\abhin\anaconda3\lib\site-packages (from
requests>=2.19.0->pandas-datareader) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\abhin\anaconda3\lib\site-packa
ges (from requests>=2.19.0->pandas-datareader) (1.26.11)
Requirement already satisfied: six>=1.5 in c:\users\abhin\anaconda3\lib\site-packages (from pyt
hon-dateutil>=2.8.1->pandas>=0.23->pandas-datareader) (1.16.0)
```

In [2]:
```
1 pip install yfinance
```

```
Collecting yfinance
  Downloading yfinance-0.2.18-py2.py3-none-any.whl (60 kB)
     ---------------------------------------- 60.3/60.3 kB 1.6 MB/s eta 0:00:00
Collecting frozendict>=2.3.4
  Downloading frozendict-2.3.8-cp39-cp39-win_amd64.whl (35 kB)
Collecting multitasking>=0.0.7
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\abhin\anaconda3\lib\site-pack
ages (from yfinance) (4.11.1)
Collecting pytz>=2022.5
  Using cached pytz-2023.3-py2.py3-none-any.whl (502 kB)
Requirement already satisfied: requests>=2.26 in c:\users\abhin\anaconda3\lib\site-packages (fr
om yfinance) (2.28.1)
Collecting html5lib>=1.1
  Using cached html5lib-1.1-py2.py3-none-any.whl (112 kB)
Requirement already satisfied: appdirs>=1.4.4 in c:\users\abhin\anaconda3\lib\site-packages (fr
om yfinance) (1.4.4)
Requirement already satisfied: lxml>=4.9.1 in c:\users\abhin\anaconda3\lib\site-packages (from
yfinance) (4.9.1)
Requirement already satisfied: cryptography>=3.3.2 in c:\users\abhin\anaconda3\lib\site-package
s (from yfinance) (37.0.1)
Requirement already satisfied: numpy>=1.16.5 in c:\users\abhin\anaconda3\lib\site-packages (fro
m yfinance) (1.21.5)
Requirement already satisfied: pandas>=1.3.0 in c:\users\abhin\anaconda3\lib\site-packages (fro
m yfinance) (1.4.4)
Requirement already satisfied: soupsieve>1.2 in c:\users\abhin\anaconda3\lib\site-packages (fro
m beautifulsoup4>=4.11.1->yfinance) (2.3.1)
Requirement already satisfied: cffi>=1.12 in c:\users\abhin\anaconda3\lib\site-packages (from c
ryptography>=3.3.2->yfinance) (1.15.1)
Requirement already satisfied: six>=1.9 in c:\users\abhin\anaconda3\lib\site-packages (from htm
l5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in c:\users\abhin\anaconda3\lib\site-packages (from
html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\abhin\anaconda3\lib\site-pack
ages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\abhin\anaconda3\lib\site-pa
ckages (from requests>=2.26->yfinance) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\abhin\anaconda3\lib\site-packages
(from requests>=2.26->yfinance) (2022.9.14)
Requirement already satisfied: idna<4,>=2.5 in c:\users\abhin\anaconda3\lib\site-packages (from
requests>=2.26->yfinance) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\abhin\anaconda3\lib\site-packa
ges (from requests>=2.26->yfinance) (1.26.11)
Requirement already satisfied: pycparser in c:\users\abhin\anaconda3\lib\site-packages (from cf
fi>=1.12->cryptography>=3.3.2->yfinance) (2.21)
Installing collected packages: pytz, multitasking, html5lib, frozendict, yfinance
  Attempting uninstall: pytz
    Found existing installation: pytz 2022.1
    Uninstalling pytz-2022.1:
      Successfully uninstalled pytz-2022.1
Successfully installed frozendict-2.3.8 html5lib-1.1 multitasking-0.0.11 pytz-2023.3 yfinance-
0.2.18
Note: you may need to restart the kernel to use updated packages.

ERROR: pip's dependency resolver does not currently take into account all the packages that are
installed. This behaviour is the source of the following dependency conflicts.
conda-repo-cli 1.0.20 requires clyent==1.2.1, but you have clyent 1.2.2 which is incompatible.
conda-repo-cli 1.0.20 requires nbformat==5.4.0, but you have nbformat 5.5.0 which is incompatib
le.
```

In [11]:
```
import pandas_datareader.data as pdr
import yfinance as yf
yf.pdr_override()
from datetime import datetime
data = pdr.get_data_yahoo('TSLA', datetime(2017, 1, 1))
```

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

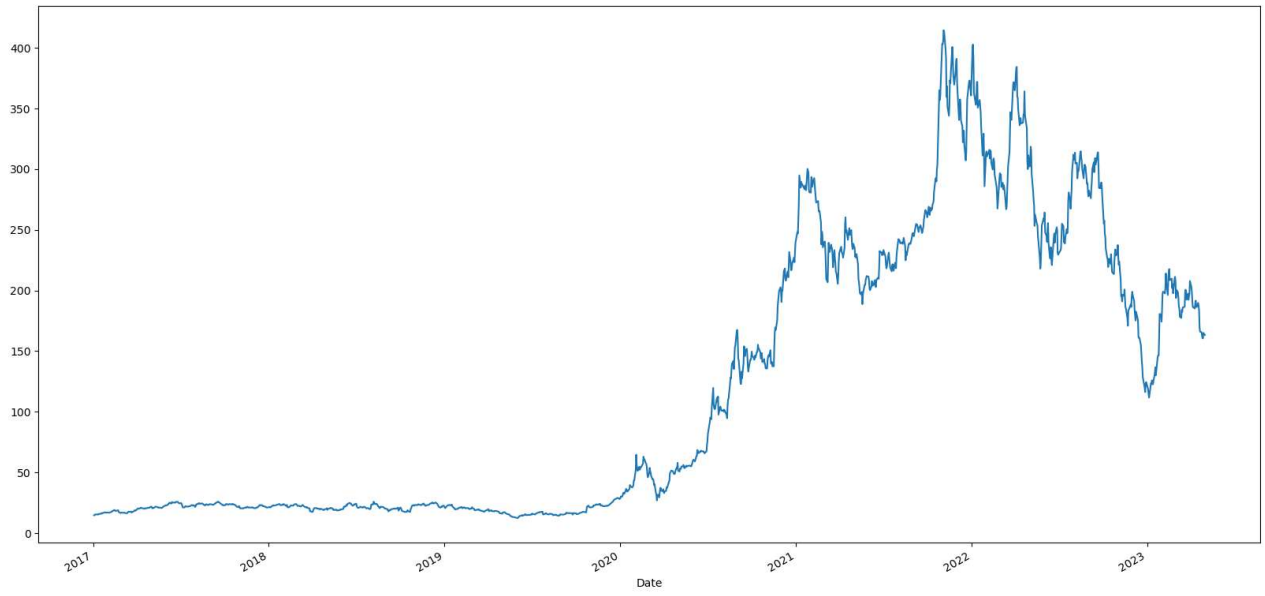
In [12]:
```
data.head()
```

Out[12]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-01-03	14.324000	14.688667	14.064000	14.466000	14.466000	88849500
2017-01-04	14.316667	15.200000	14.287333	15.132667	15.132667	168202500
2017-01-05	15.094667	15.165333	14.796667	15.116667	15.116667	88675500
2017-01-06	15.128667	15.354000	15.030000	15.267333	15.267333	82918500
2017-01-09	15.264667	15.461333	15.200000	15.418667	15.418667	59692500

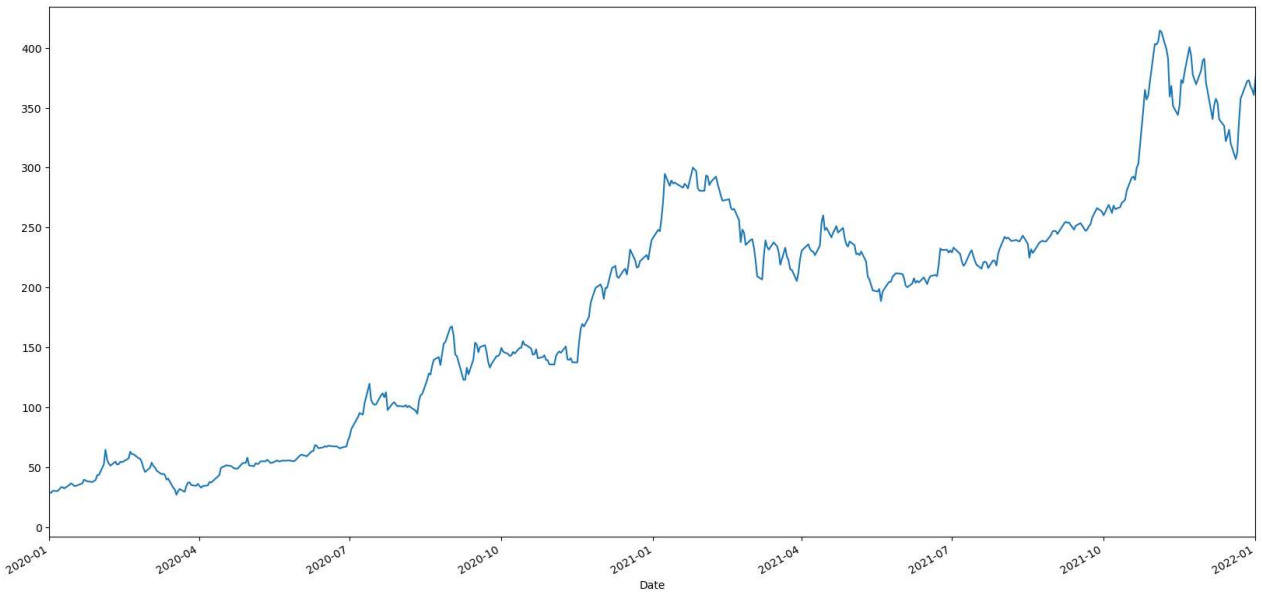
In [17]:
```
data['High'].plot(figsize=(20,10))
```

Out[17]: <AxesSubplot:xlabel='Date'>



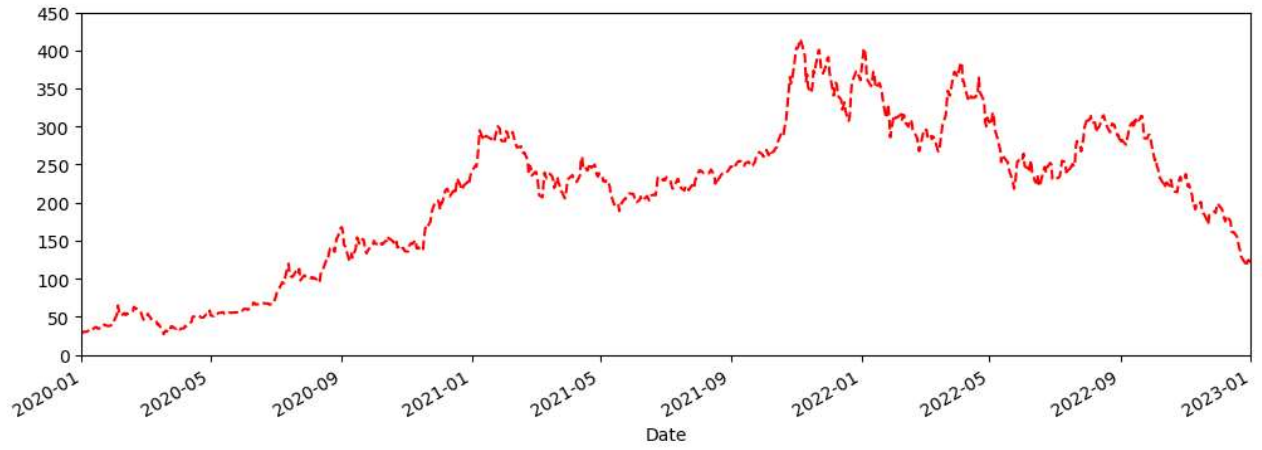
In [18]:
```
# We are really intreseted in the hike in share price from 2020 to 2022
# We define the xlimit and ylimit

data['High'].plot(xlim=['2020-01-01','2022-01-01'],figsize=(20,10))
```

Out[18]: <AxesSubplot:xlabel='Date'>



In [27]:
```
# Applying xlim and ylim along with color
data['High'].plot(xlim=['2020-01-01','2023-01-01'],ylim=[0,450],figsize=(12,4),c='red',ls='-
```

Out[27]: <AxesSubplot:xlabel='Date'>



In [30]:
```
data.iloc[:4]
```

Out[30]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-01-03	14.324000	14.688667	14.064000	14.466000	14.466000	88849500
2017-01-04	14.316667	15.200000	14.287333	15.132667	15.132667	168202500
2017-01-05	15.094667	15.165333	14.796667	15.116667	15.116667	88675500
2017-01-06	15.128667	15.354000	15.030000	15.267333	15.267333	82918500

In [31]:
```
index=data.loc['2020-01-01':'2021-09-01'].index
share_open=data.loc['2020-01-01':'2021-09-01']['Open']
```

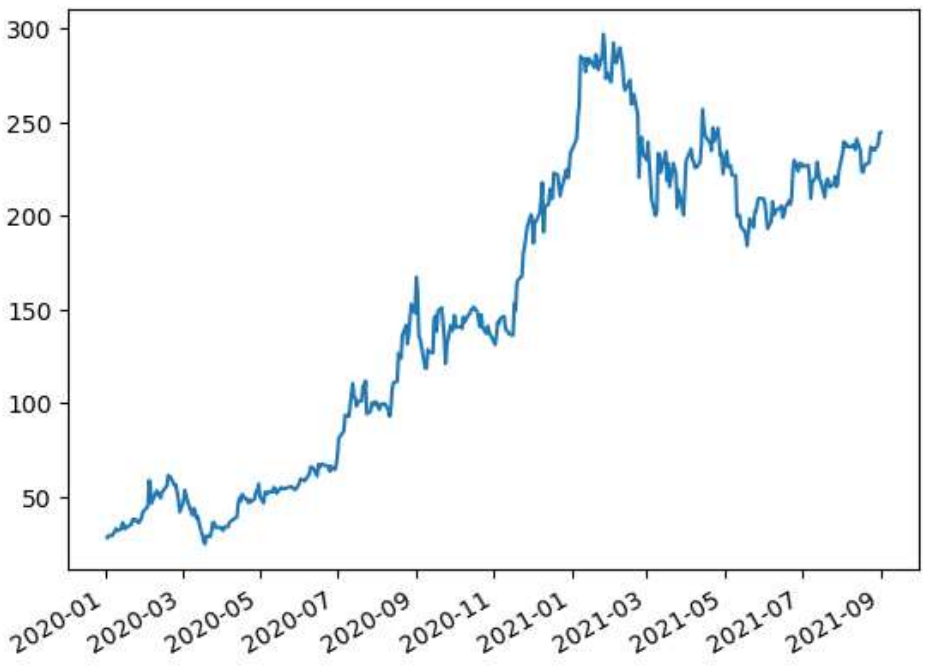
In [32]:
```
share_open
```

Out[32]:
```
Date
2020-01-02     28.299999
2020-01-03     29.366667
2020-01-06     29.364668
2020-01-07     30.760000
2020-01-08     31.580000
                  ...
2021-08-26    236.103333
2021-08-27    235.000000
2021-08-30    238.240005
2021-08-31    244.333328
2021-09-01    244.693329
Name: Open, Length: 421, dtype: float64
```

In [33]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [37]:
```
figure,axis=plt.subplots()
# To prevent overlapping we use autoformating inbuilt function
figure.autofmt_xdate()
axis.plot(index,share_open)
```

Out[37]:
```
[<matplotlib.lines.Line2D at 0x1bad28ab0d0>]
```



In [38]:
```
# Datetime index
```

In [40]:
```
1 data.head()
```

Out[40]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-01-03	14.324000	14.688667	14.064000	14.466000	14.466000	88849500
2017-01-04	14.316667	15.200000	14.287333	15.132667	15.132667	168202500
2017-01-05	15.094667	15.165333	14.796667	15.116667	15.116667	88675500
2017-01-06	15.128667	15.354000	15.030000	15.267333	15.267333	82918500
2017-01-09	15.264667	15.461333	15.200000	15.418667	15.418667	59692500

In [42]:
```
1 data.info() #datetime is not being displayed as in datetime format
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1592 entries, 2017-01-03 to 2023-05-01
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Open       1592 non-null   float64
 1   High       1592 non-null   float64
 2   Low        1592 non-null   float64
 3   Close      1592 non-null   float64
 4   Adj Close  1592 non-null   float64
 5   Volume     1592 non-null   int64  
dtypes: float64(5), int64(1)
memory usage: 151.6 KB
```

In [43]:
```
1 data=data.reset_index()
```

In [44]:
```
1 data.head()
```

Out[44]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-01-03	14.324000	14.688667	14.064000	14.466000	14.466000	88849500
1	2017-01-04	14.316667	15.200000	14.287333	15.132667	15.132667	168202500
2	2017-01-05	15.094667	15.165333	14.796667	15.116667	15.116667	88675500
3	2017-01-06	15.128667	15.354000	15.030000	15.267333	15.267333	82918500
4	2017-01-09	15.264667	15.461333	15.200000	15.418667	15.418667	59692500

In [46]:
```
1 data.info() # datetime is being displayed as a seperate datatype
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1592 entries, 0 to 1591
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   Date       1592 non-null   datetime64[ns]
 1   Open       1592 non-null   float64       
 2   High       1592 non-null   float64       
 3   Low        1592 non-null   float64       
 4   Close      1592 non-null   float64       
 5   Adj Close  1592 non-null   float64       
 6   Volume     1592 non-null   int64         
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 87.2 KB
```

In [57]:
```
data=data.set_index('Date')
```

In [58]:
```
# Datetime
from datetime import datetime
```

In [59]:
```
datetime.now()
```

Out[59]:
```
datetime.datetime(2023, 5, 2, 12, 54, 46, 639789)
```

Time Resampling

In [60]:
```
data.head()
```

Out[60]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-01-03	14.324000	14.688667	14.064000	14.466000	14.466000	88849500
2017-01-04	14.316667	15.200000	14.287333	15.132667	15.132667	168202500
2017-01-05	15.094667	15.165333	14.796667	15.116667	15.116667	88675500
2017-01-06	15.128667	15.354000	15.030000	15.267333	15.267333	82918500
2017-01-09	15.264667	15.461333	15.200000	15.418667	15.418667	59692500

In [61]:
```
# to get the minimum of each column in the particular year that to on the last date of that p
data.resample(rule='A').min()
```

Out[61]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-12-31	14.316667	14.688667	14.064000	14.466000	14.466000	32800500
2018-12-31	16.851999	17.355333	16.306000	16.704000	16.704000	46210500
2019-12-31	12.073333	12.445333	11.799333	11.931333	11.931333	36984000
2020-12-31	24.980000	26.990667	23.367332	24.081333	24.081333	52073100
2021-12-31	184.183334	188.736664	179.830002	187.666672	187.666672	29401800
2022-12-31	110.349998	116.269997	108.239998	109.099998	109.099998	41864700
2023-12-31	103.000000	111.750000	101.809998	108.099998	108.099998	92067000

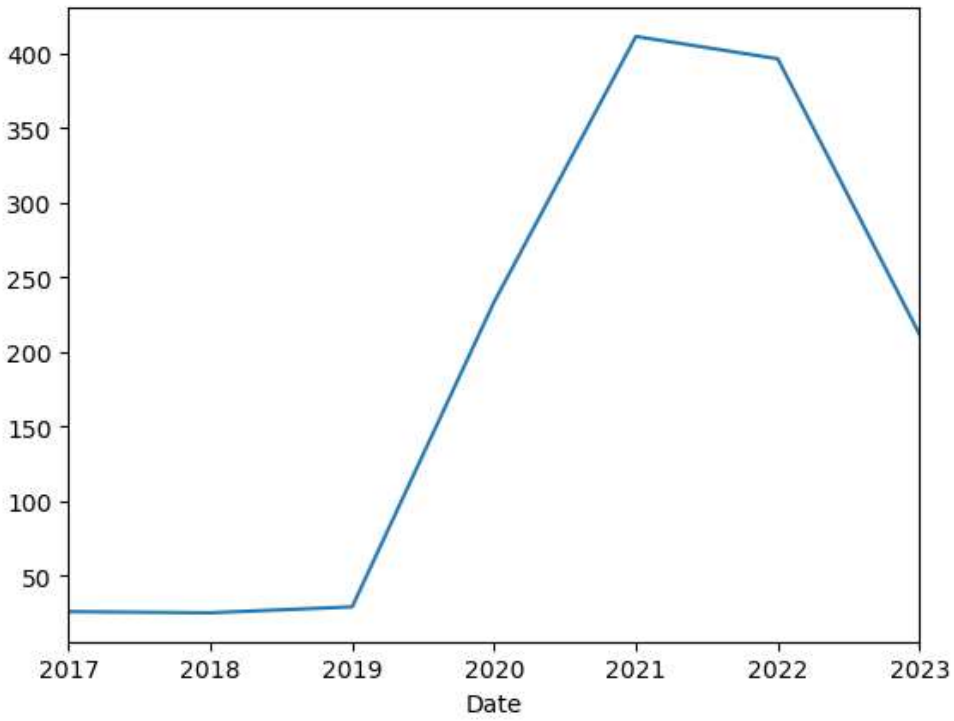
In [62]:
```
# to get the maximum of each column in the particular year that to on the last date of that p

# RULE "A" => Year End Frequency
data.resample(rule='A').max()
```

Out[62]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-12-31	25.779333	25.974001	25.290001	25.666668	25.666668	296871000
2018-12-31	25.000000	25.830667	24.474667	25.304667	25.304667	504745500
2019-12-31	29.000000	29.020666	28.423332	28.729334	28.729334	450091500
2020-12-31	233.330002	239.573334	230.373337	235.223328	235.223328	914082000
2021-12-31	411.470001	414.496674	405.666656	409.970001	409.970001	268189500
2022-12-31	396.516663	402.666656	378.679993	399.926666	399.926666	221923300
2023-12-31	211.759995	217.649994	206.110001	214.240005	214.240005	306590600

In [63]:
```
# Plotting this data
data.resample(rule='A').max()['Open'].plot()
```

Out[63]: <AxesSubplot:xlabel='Date'>



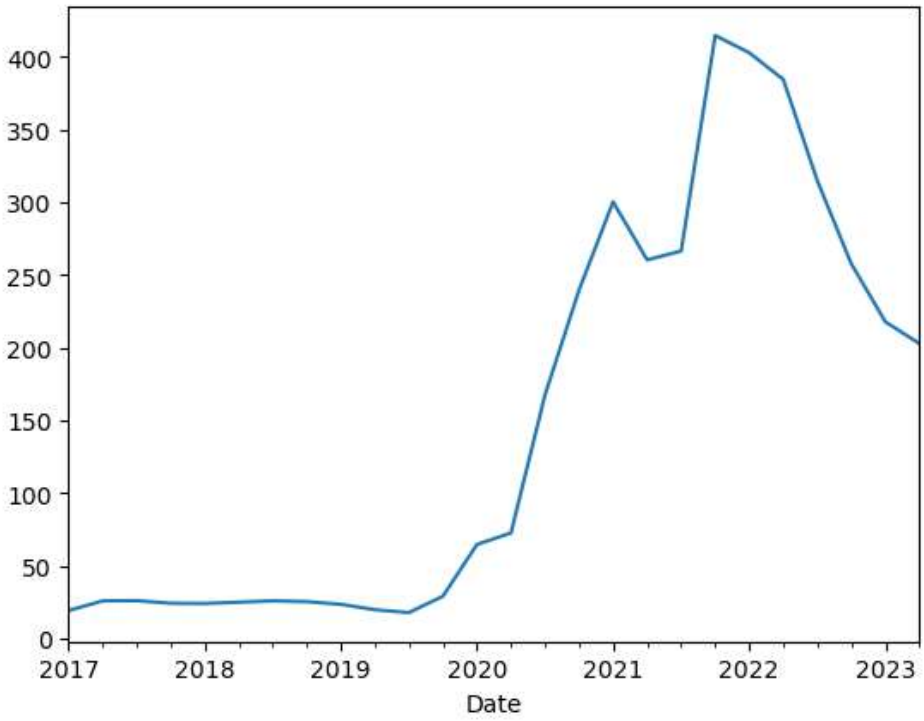
In [64]:
```
# to get the maximum of each column in the particular Quater that to on the last date of that
data.resample(rule='QS').max()
```

Out[64]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-01-01	18.687332	19.159332	18.573999	18.732000	18.732000	223728000
2017-04-01	25.779333	25.799334	25.290001	25.563334	25.563334	258921000
2017-07-01	25.350000	25.974001	25.178667	25.666668	25.666668	289867500
2017-10-01	23.798668	24.200001	23.608667	23.976667	23.976667	296871000
2018-01-01	24.000000	24.033333	23.490667	23.827999	23.827999	315021000
2018-04-01	24.344000	24.915333	23.633333	24.722000	24.722000	335211000
2018-07-01	24.606001	25.830667	24.474667	25.304667	25.304667	504745500
2018-10-01	25.000000	25.299334	24.450001	25.119333	25.119333	411382500
2019-01-01	23.080667	23.466667	22.943333	23.153999	23.153999	362262000
2019-04-01	19.219999	19.744667	19.144667	19.454000	19.454000	398206500
2019-07-01	17.278000	17.738001	17.210667	17.658667	17.658667	336274500
2019-10-01	29.000000	29.020666	28.423332	28.729334	28.729334	450091500
2020-01-01	61.566666	64.599335	60.068001	61.161331	61.161331	914082000
2020-04-01	67.518669	72.512665	66.915337	71.987335	71.987335	487977000
2020-07-01	167.380005	167.496674	156.836670	166.106674	166.106674	584781000
2020-10-01	233.330002	239.573334	230.373337	235.223328	235.223328	666378600
2021-01-01	297.126678	300.133331	290.533325	294.363342	294.363342	268189500
2021-04-01	256.899994	260.263336	244.203339	254.106674	254.106674	147052200
2021-07-01	262.399994	266.333344	258.333344	263.786682	263.786682	100847400
2021-10-01	411.470001	414.496674	405.666656	409.970001	409.970001	188556300
2022-01-01	396.516663	402.666656	378.679993	399.926666	399.926666	151565700
2022-04-01	378.766663	384.290009	362.433319	381.816681	381.816681	144973200
2022-07-01	311.666656	314.666656	305.579987	309.320007	309.320007	142032300
2022-10-01	254.500000	257.500000	242.009995	249.440002	249.440002	221923300
2023-01-01	211.759995	217.649994	206.110001	214.240005	214.240005	306590600
2023-04-01	199.910004	202.690002	192.199997	194.770004	194.770004	210970800

In [65]:
```
1 data.resample(rule='QS').max()['High'].plot()
```

Out[65]: <AxesSubplot:xlabel='Date'>



In [66]:
```
1 # to get the maximum of Bussiness End frequency
2 data.resample(rule='BA').max()
```

Out[66]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-12-29	25.779333	25.974001	25.290001	25.666668	25.666668	296871000
2018-12-31	25.000000	25.830667	24.474667	25.304667	25.304667	504745500
2019-12-31	29.000000	29.020666	28.423332	28.729334	28.729334	450091500
2020-12-31	233.330002	239.573334	230.373337	235.223328	235.223328	914082000
2021-12-31	411.470001	414.496674	405.666656	409.970001	409.970001	268189500
2022-12-30	396.516663	402.666656	378.679993	399.926666	399.926666	221923300
2023-12-29	211.759995	217.649994	206.110001	214.240005	214.240005	306590600

In [69]:
```
# to get the maximum of Bussiness End frequency Quaterly
data.resample(rule='BQS').max()
```

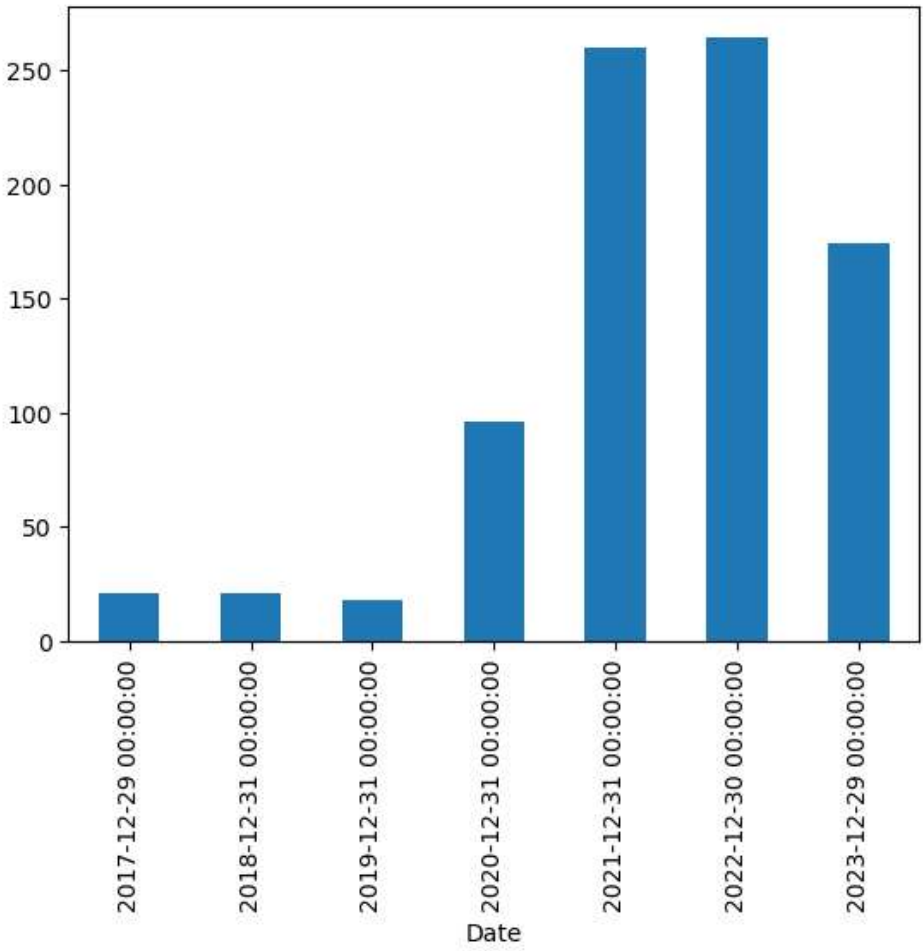
Out[69]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-01-02	18.687332	19.159332	18.573999	18.732000	18.732000	223728000
2017-04-03	25.779333	25.799334	25.290001	25.563334	25.563334	258921000
2017-07-03	25.350000	25.974001	25.178667	25.666668	25.666668	289867500
2017-10-02	23.798668	24.200001	23.608667	23.976667	23.976667	296871000
2018-01-01	24.000000	24.033333	23.490667	23.827999	23.827999	315021000
2018-04-02	24.344000	24.915333	23.633333	24.722000	24.722000	335211000
2018-07-02	24.606001	25.830667	24.474667	25.304667	25.304667	504745500
2018-10-01	25.000000	25.299334	24.450001	25.119333	25.119333	411382500
2019-01-01	23.080667	23.466667	22.943333	23.153999	23.153999	362262000
2019-04-01	19.219999	19.744667	19.144667	19.454000	19.454000	398206500
2019-07-01	17.278000	17.738001	17.210667	17.658667	17.658667	336274500
2019-10-01	29.000000	29.020666	28.423332	28.729334	28.729334	450091500
2020-01-01	61.566666	64.599335	60.068001	61.161331	61.161331	914082000
2020-04-01	67.518669	72.512665	66.915337	71.987335	71.987335	487977000
2020-07-01	167.380005	167.496674	156.836670	166.106674	166.106674	584781000
2020-10-01	233.330002	239.573334	230.373337	235.223328	235.223328	666378600
2021-01-01	297.126678	300.133331	290.533325	294.363342	294.363342	268189500
2021-04-01	256.899994	260.263336	244.203339	254.106674	254.106674	147052200
2021-07-01	262.399994	266.333344	258.333344	263.786682	263.786682	100847400
2021-10-01	411.470001	414.496674	405.666656	409.970001	409.970001	188556300
2022-01-03	396.516663	402.666656	378.679993	399.926666	399.926666	151565700
2022-04-01	378.766663	384.290009	362.433319	381.816681	381.816681	144973200
2022-07-01	311.666656	314.666656	305.579987	309.320007	309.320007	142032300
2022-10-03	254.500000	257.500000	242.009995	249.440002	249.440002	221923300
2023-01-02	211.759995	217.649994	206.110001	214.240005	214.240005	306590600
2023-04-03	199.910004	202.690002	192.199997	194.770004	194.770004	210970800

Visualization

In [72]:
```
1 data['Open'].resample(rule='BA').mean().plot(kind='bar')
```

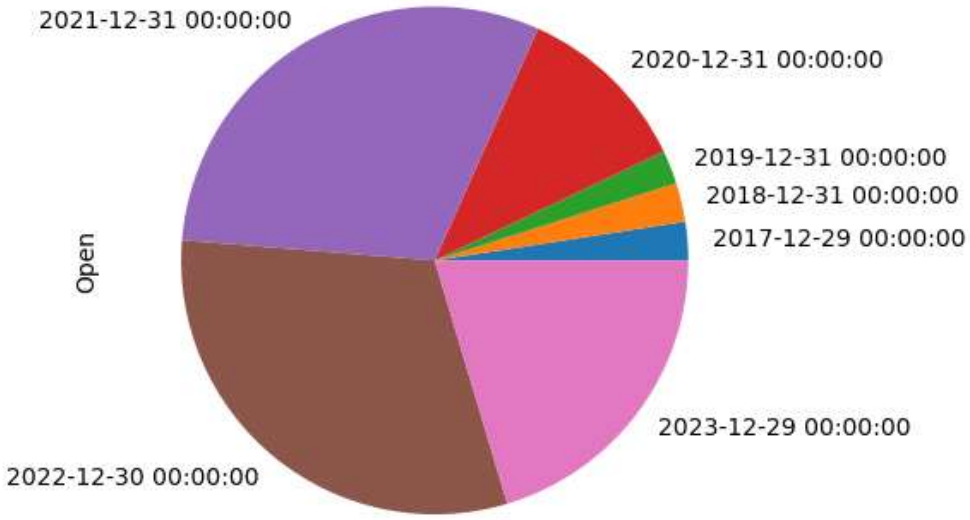
Out[72]: <AxesSubplot:xlabel='Date'>



In [ ]:
```
1
```

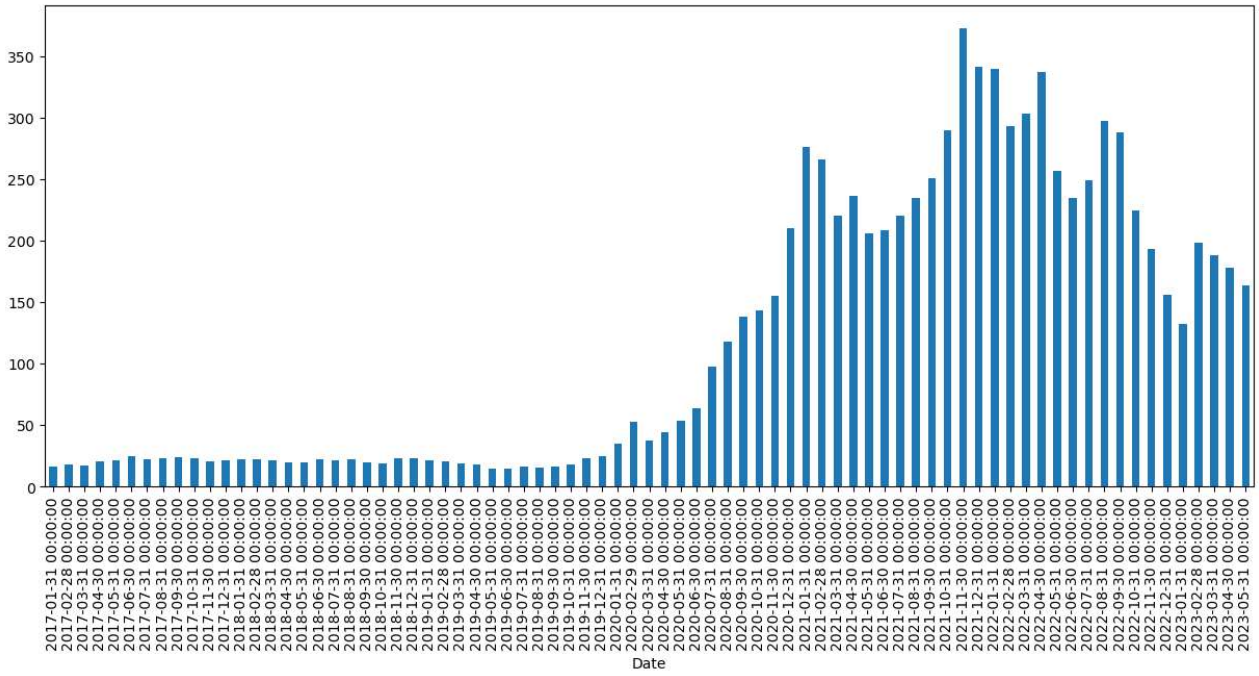
In [77]:
```
1 data['Open'].resample(rule='BA').mean().plot(kind='pie')
```

Out[77]: <AxesSubplot:ylabel='Open'>



In [78]:
```
1 # Rule 'M' is used for the montly data
2 data['Open'].resample(rule='M').mean().plot(kind='bar',figsize=(15,6))
```

Out[78]: <AxesSubplot:xlabel='Date'>



Rolling and Expanding

In [ ]:
```
# Rolling is used to get down the columns and getting the mean for the next 'n' columns
# Rolling window is used for the moving average
```

In [80]:
```
data['Open'].rolling(5).mean().head(10)
```

Out[80]:
```
Date
2017-01-03          NaN
2017-01-04          NaN
2017-01-05          NaN
2017-01-06          NaN
2017-01-09    14.825734
2017-01-10    15.054267
2017-01-11    15.245200
2017-01-12    15.280400
2017-01-13    15.321333
2017-01-17    15.424400
Name: Open, dtype: float64
```

In [81]:
```
data['High'].rolling(10).mean().head(20)
```

Out[81]:
```
Date
2017-01-03          NaN
2017-01-04          NaN
2017-01-05          NaN
2017-01-06          NaN
2017-01-09          NaN
2017-01-10          NaN
2017-01-11          NaN
2017-01-12          NaN
2017-01-13          NaN
2017-01-17    15.390200
2017-01-18    15.519400
2017-01-19    15.657267
2017-01-20    15.780733
2017-01-23    15.917933
2017-01-24    16.070467
2017-01-25    16.246867
2017-01-26    16.418600
2017-01-27    16.567267
2017-01-30    16.683533
2017-01-31    16.789734
Name: High, dtype: float64
```

In [82]:
```
data['Open: 30 days rolling']=data['Open'].rolling(30).mean()
```

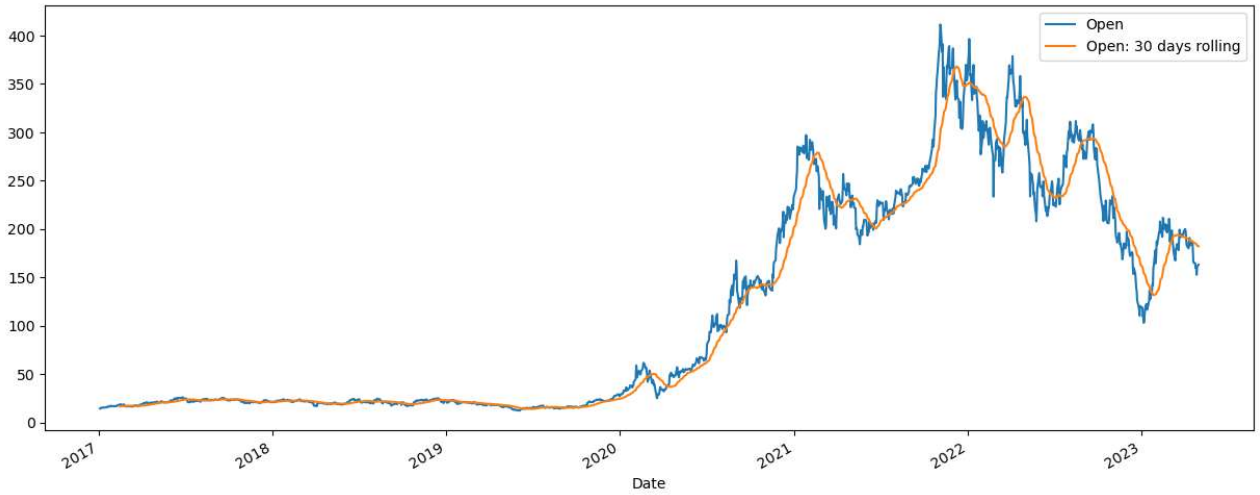
In [83]:
```
data.head()
```

Out[83]:

Date	Open	High	Low	Close	Adj Close	Volume	Open: 30 days rolling
2017-01-03	14.324000	14.688667	14.064000	14.466000	14.466000	88849500	NaN
2017-01-04	14.316667	15.200000	14.287333	15.132667	15.132667	168202500	NaN
2017-01-05	15.094667	15.165333	14.796667	15.116667	15.116667	88675500	NaN
2017-01-06	15.128667	15.354000	15.030000	15.267333	15.267333	82918500	NaN
2017-01-09	15.264667	15.461333	15.200000	15.418667	15.418667	59692500	NaN

In [85]:
```
data[['Open','Open: 30 days rolling']].plot(figsize=(15,6))
# we can see that smoothing has been done
```

Out[85]: <AxesSubplot:xlabel='Date'>



In [ ]:
```

```