NAME: RETHINAGIRI G

ROLL NO: 225229130

COURSE TITLE : DATA AND VISUAL ANALYTICS LAB

Lab8. Pandas Time Series Analysis

In [1]:
```
import pandas as pd
```

In [2]:
```
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
plt.show()
```

In [3]:
```
data=pd.read_csv('amazon_stock.csv')
```

Inspect top 10 rows

In [4]:
```
data.head(10)
```

Out[4]:

	None	ticker	Date	Open	High	Low	Close	Volume	Adj_Close
0	0	AMZN	3/27/2018	1572.40	1575.96	1482.32	1497.05	6793279	1497.05
1	1	AMZN	3/26/2018	1530.00	1556.99	1499.25	1555.86	5547618	1555.86
2	2	AMZN	3/23/2018	1539.01	1549.02	1495.36	1495.56	7843966	1495.56
3	3	AMZN	3/22/2018	1565.47	1573.85	1542.40	1544.10	6177737	1544.10
4	4	AMZN	3/21/2018	1586.45	1590.00	1563.17	1581.86	4667291	1581.86
5	5	AMZN	3/20/2018	1550.34	1587.00	1545.41	1586.51	4507049	1586.51
6	6	AMZN	3/19/2018	1554.53	1561.66	1525.35	1544.93	6376619	1544.93
7	7	AMZN	3/16/2018	1583.45	1589.44	1567.50	1571.68	5145054	1571.68
8	8	AMZN	3/15/2018	1595.00	1596.91	1578.11	1582.32	4026744	1582.32
9	9	AMZN	3/14/2018	1597.00	1606.44	1590.89	1591.00	4164395	1591.00

Remove unwanted columns

Remove first two columns (None and ticker) as they don't add any value to the dataset. Then, print head() to check if removed

In [5]:
```
data=data.drop(['None','ticker'],axis=1)
data
```

Out[5]:

	Date	Open	High	Low	Close	Volume	Adj_Close
0	3/27/2018	1572.40	1575.9600	1482.320	1497.0500	6793279	1497.0500
1	3/26/2018	1530.00	1556.9900	1499.250	1555.8600	5547618	1555.8600
2	3/23/2018	1539.01	1549.0200	1495.360	1495.5600	7843966	1495.5600
3	3/22/2018	1565.47	1573.8500	1542.400	1544.1000	6177737	1544.1000
4	3/21/2018	1586.45	1590.0000	1563.170	1581.8600	4667291	1581.8600
...	...	...	...	...	...	...	...
1311	01-08-2013	267.07	268.9800	263.567	266.3800	3010700	266.3800
1312	01-07-2013	262.97	269.7250	262.670	268.4592	4910000	268.4592
1313	01-04-2013	257.58	259.8000	256.650	259.1500	1874200	259.1500
1314	01-03-2013	257.27	260.8800	256.370	258.4800	2750900	258.4800
1315	01-02-2013	256.08	258.0999	253.260	257.3100	3271000	257.3100

1316 rows × 7 columns

In [6]:
```
data.dtypes
```

Out[6]:
```
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Volume         int64
Adj_Close    float64
dtype: object
```

Inspect the datatypes of columns

Looking at the information, it appears that Date column is being treated as a string rather than as dates. To fix this, we'll use the pandas to_datetime() feature which converts the arguments to dates.

Convert "Date" string column into actual Date object

In [7]:
```
data['Date']=pd.to_datetime(data['Date'])
data.dtypes
```

Out[7]:
```
Date          datetime64[ns]
Open                 float64
High                 float64
Low                  float64
Close                float64
Volume                 int64
Adj_Close            float64
dtype: object
```

Let us check our data once again, with head()

In [8]:
```
data.head()
```

Out[8]:

	Date	Open	High	Low	Close	Volume	Adj_Close
0	2018-03-27	1572.40	1575.96	1482.32	1497.05	6793279	1497.05
1	2018-03-26	1530.00	1556.99	1499.25	1555.86	5547618	1555.86
2	2018-03-23	1539.01	1549.02	1495.36	1495.56	7843966	1495.56
3	2018-03-22	1565.47	1573.85	1542.40	1544.10	6177737	1544.10
4	2018-03-21	1586.45	1590.00	1563.17	1581.86	4667291	1581.86

Set Date object to be index

Here Date is one of the columns. But we want date to be the index. So, set Date as index for the dataframe. Make inplace=True

In [9]:
```
data.set_index(['Date'],inplace=True)
```

In [10]:
```
data.head()
```

Out[10]:

Date	Open	High	Low	Close	Volume	Adj_Close
2018-03-27	1572.40	1575.96	1482.32	1497.05	6793279	1497.05
2018-03-26	1530.00	1556.99	1499.25	1555.86	5547618	1555.86
2018-03-23	1539.01	1549.02	1495.36	1495.56	7843966	1495.56
2018-03-22	1565.47	1573.85	1542.40	1544.10	6177737	1544.10
2018-03-21	1586.45	1590.00	1563.17	1581.86	4667291	1581.86

Understand Stock Data

In [11]:
```
data['Adj_Close'].plot(figsize=(12,6),title="Adjusted Closing Price")
```

Out[11]:
```
<AxesSubplot:title={'center':'Adjusted Closing Price'}, xlabel='Date'>
```

Understand DateTimeIndex

Introduction to datetime module

In [12]:
```
from datetime import datetime

my_year=2020
my_month=5
my_day=1
my_hour=13
my_minute=36
my_second=45

test_date=datetime(my_year,my_month,my_day)
test_date
```

Out[12]:
```
datetime.datetime(2020, 5, 1, 0, 0)
```

test_date = datetime(my_year, my_month, my_day, my_hour, my_minute, my_second) print("The day is : ",test_date.day) print("The hour is : ", test_date.hour) print("The month is : ", test_date.month)

Find minimum and maximum dates from data frame, call info() method

In [13]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1316 entries, 2018-03-27 to 2013-01-02
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Open       1316 non-null   float64
 1   High       1316 non-null   float64
 2   Low        1316 non-null   float64
 3   Close      1316 non-null   float64
 4   Volume     1316 non-null   int64  
 5   Adj_Close  1316 non-null   float64
dtypes: float64(5), int64(1)
memory usage: 72.0 KB
```

Print minimum and maximum index value of dataframe

In [14]:
```
print(data.index.min())
print(data.index.max())
```

```
2013-01-02 00:00:00
2018-03-27 00:00:00
```

Retrieve index of earliest and latest dates using argmin and argmax

We can also calculate the latest date location and the earliest date index location as follows

In [15]:
```
data.index.argmin()
```

Out[15]:
```
1315
```

In [16]:
```
data.index.argmax()
```

Out[16]:
```
0
```

1.Resampling Operation

Resample entire data frame

Resample data with year end frequency ("Y") with average stock price

In [17]:
```
y=data.resample('Y')
y.mean()
```

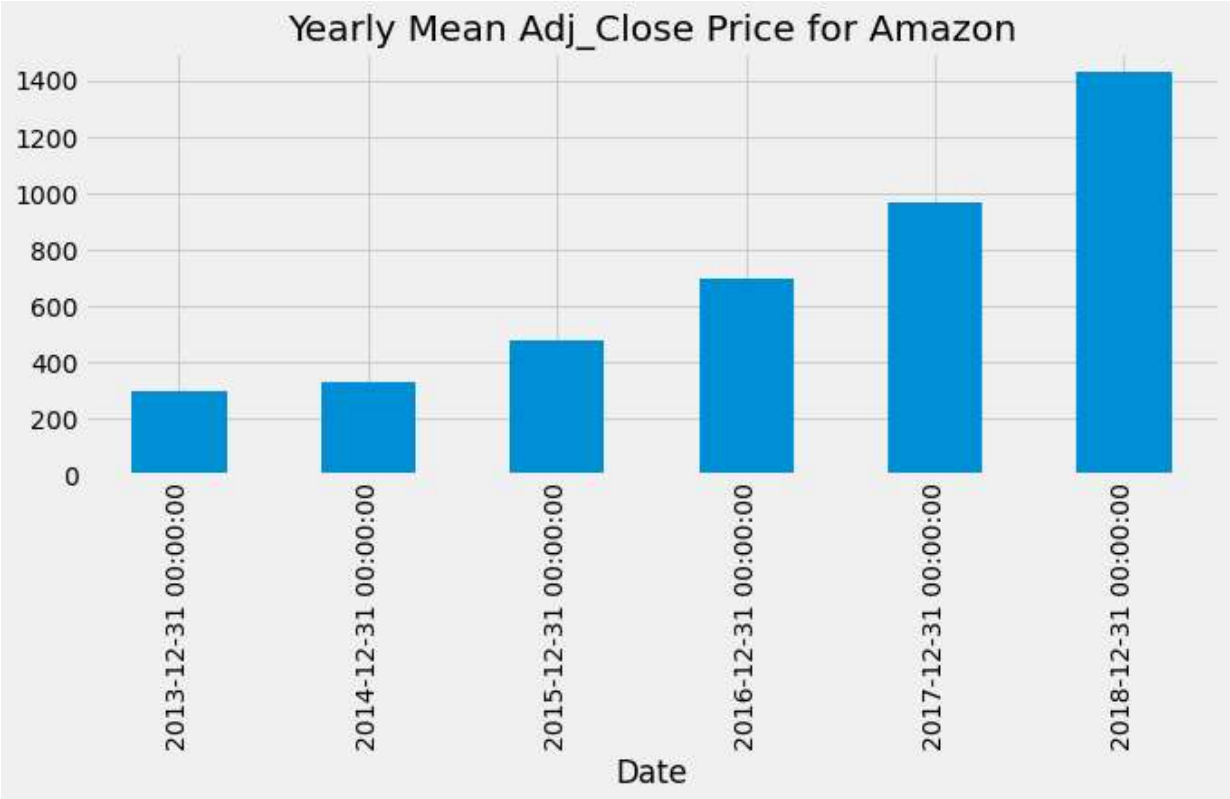
Out[17]:

Date	Open	High	Low	Close	Volume	Adj_Close
2013-12-31	297.877223	300.925966	294.656658	298.032235	2.967880e+06	298.032235
2014-12-31	332.798433	336.317462	328.545440	332.550976	4.083223e+06	332.550976
2015-12-31	478.126230	483.248272	472.875443	478.137321	3.797801e+06	478.137321
2016-12-31	699.669762	705.799103	692.646189	699.523135	4.122043e+06	699.523135
2017-12-31	967.565060	973.789752	959.991826	967.403996	3.466207e+06	967.403996
2018-12-31	1429.770000	1446.701017	1409.469661	1429.991186	5.586829e+06	1429.991186

Resample a specific column

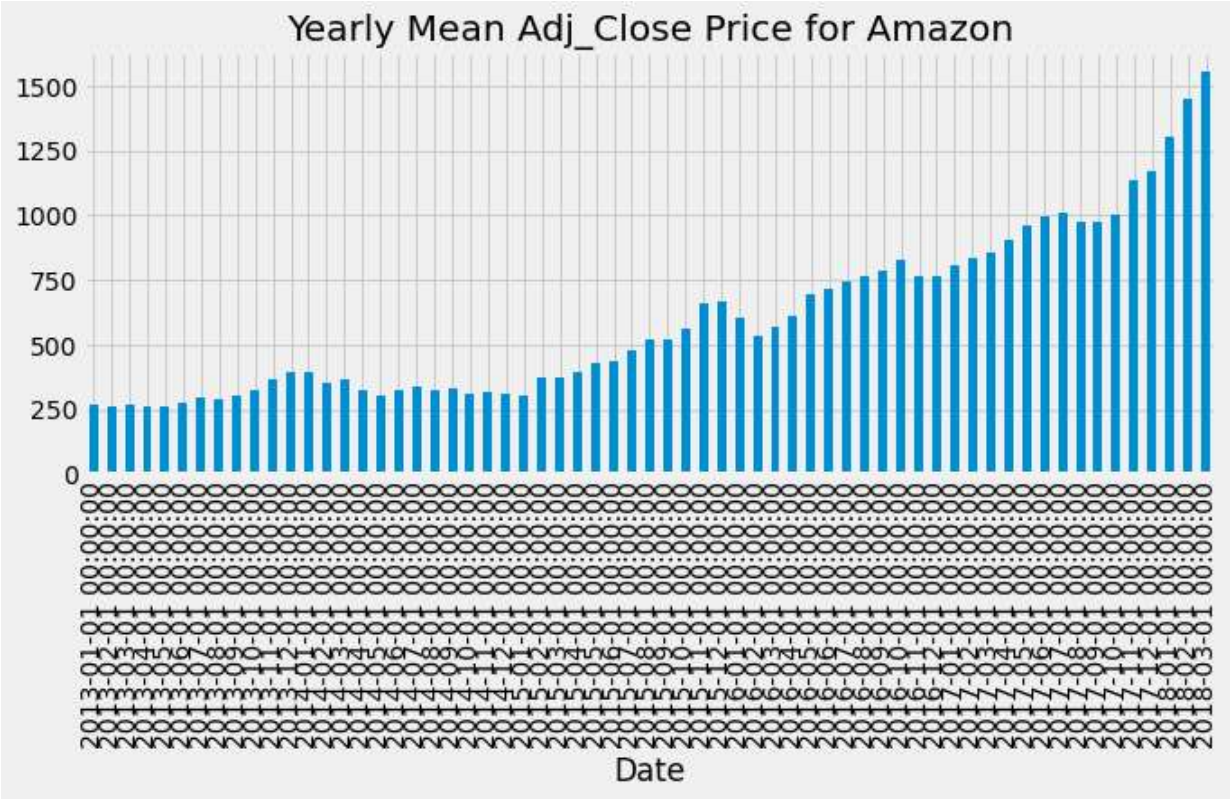
Plot a bar chart to show the yearly (Use "A") mean adjusted close price

In [18]:
```
data["Adj_Close"].resample('A').mean().plot(kind='bar',figsize=(10,4))
plt.title("Yearly Mean Adj_Close Price for Amazon")
plt.show()
```



Plot bar chart to show monthly maximum (Use "MS") opening price for all years

In [19]:
```
data["Open"].resample('MS').mean().plot(kind='bar',figsize=(10,4))
plt.title("Yearly Mean Adj_Close Price for Amazon")
plt.show()
```



Plot bar chart of Quarterly (Use "Q") Average Volume for all years

In [20]:
```
data["Volume"].resample('Q').mean().plot(kind='bar',figsize=(10,4))
plt.title("Yearly Mean Adj_Close Price for Amazon")
plt.show()
```

2.Time Shifting Operations

Shifting data forward and backward

Show head of data

In [21]:
```
data.head()
```

Out[21]:

Date	Open	High	Low	Close	Volume	Adj_Close
2018-03-27	1572.40	1575.96	1482.32	1497.05	6793279	1497.05
2018-03-26	1530.00	1556.99	1499.25	1555.86	5547618	1555.86
2018-03-23	1539.01	1549.02	1495.36	1495.56	7843966	1495.56
2018-03-22	1565.47	1573.85	1542.40	1544.10	6177737	1544.10
2018-03-21	1586.45	1590.00	1563.17	1581.86	4667291	1581.86

Shift data by 1 Day forward

In [22]:
```
data.shift(periods=1).head()
```

Out[22]:

Date	Open	High	Low	Close	Volume	Adj_Close
2018-03-27	NaN	NaN	NaN	NaN	NaN	NaN
2018-03-26	1572.40	1575.96	1482.32	1497.05	6793279.0	1497.05
2018-03-23	1530.00	1556.99	1499.25	1555.86	5547618.0	1555.86
2018-03-22	1539.01	1549.02	1495.36	1495.56	7843966.0	1495.56
2018-03-21	1565.47	1573.85	1542.40	1544.10	6177737.0	1544.10

Shift data by 1 Day Backward

In [23]:
```
data.shift(periods=-1).head()
```

Out[23]:

Date	Open	High	Low	Close	Volume	Adj_Close
2018-03-27	1530.00	1556.99	1499.25	1555.86	5547618.0	1555.86
2018-03-26	1539.01	1549.02	1495.36	1495.56	7843966.0	1495.56
2018-03-23	1565.47	1573.85	1542.40	1544.10	6177737.0	1544.10
2018-03-22	1586.45	1590.00	1563.17	1581.86	4667291.0	1581.86
2018-03-21	1550.34	1587.00	1545.41	1586.51	4507049.0	1586.51

Shifting Time Index

In [24]:
```
data.head(10)
```

Out[24]:

Date	Open	High	Low	Close	Volume	Adj_Close
2018-03-27	1572.40	1575.96	1482.32	1497.05	6793279	1497.05
2018-03-26	1530.00	1556.99	1499.25	1555.86	5547618	1555.86
2018-03-23	1539.01	1549.02	1495.36	1495.56	7843966	1495.56
2018-03-22	1565.47	1573.85	1542.40	1544.10	6177737	1544.10
2018-03-21	1586.45	1590.00	1563.17	1581.86	4667291	1581.86
2018-03-20	1550.34	1587.00	1545.41	1586.51	4507049	1586.51
2018-03-19	1554.53	1561.66	1525.35	1544.93	6376619	1544.93
2018-03-16	1583.45	1589.44	1567.50	1571.68	5145054	1571.68
2018-03-15	1595.00	1596.91	1578.11	1582.32	4026744	1582.32
2018-03-14	1597.00	1606.44	1590.89	1591.00	4164395	1591.00

Shift Time Index by 3 Months

In [25]:
```
data.shift(periods=3,freq="M")
```

Out[25]:

Date	Open	High	Low	Close	Volume	Adj_Close
2018-05-31	1572.40	1575.9600	1482.320	1497.0500	6793279	1497.0500
2018-05-31	1530.00	1556.9900	1499.250	1555.8600	5547618	1555.8600
2018-05-31	1539.01	1549.0200	1495.360	1495.5600	7843966	1495.5600
2018-05-31	1565.47	1573.8500	1542.400	1544.1000	6177737	1544.1000
2018-05-31	1586.45	1590.0000	1563.170	1581.8600	4667291	1581.8600
...	...	...	...	...	...	...
2013-03-31	267.07	268.9800	263.567	266.3800	3010700	266.3800
2013-03-31	262.97	269.7250	262.670	268.4592	4910000	268.4592
2013-03-31	257.58	259.8000	256.650	259.1500	1874200	259.1500
2013-03-31	257.27	260.8800	256.370	258.4800	2750900	258.4800
2013-03-31	256.08	258.0999	253.260	257.3100	3271000	257.3100

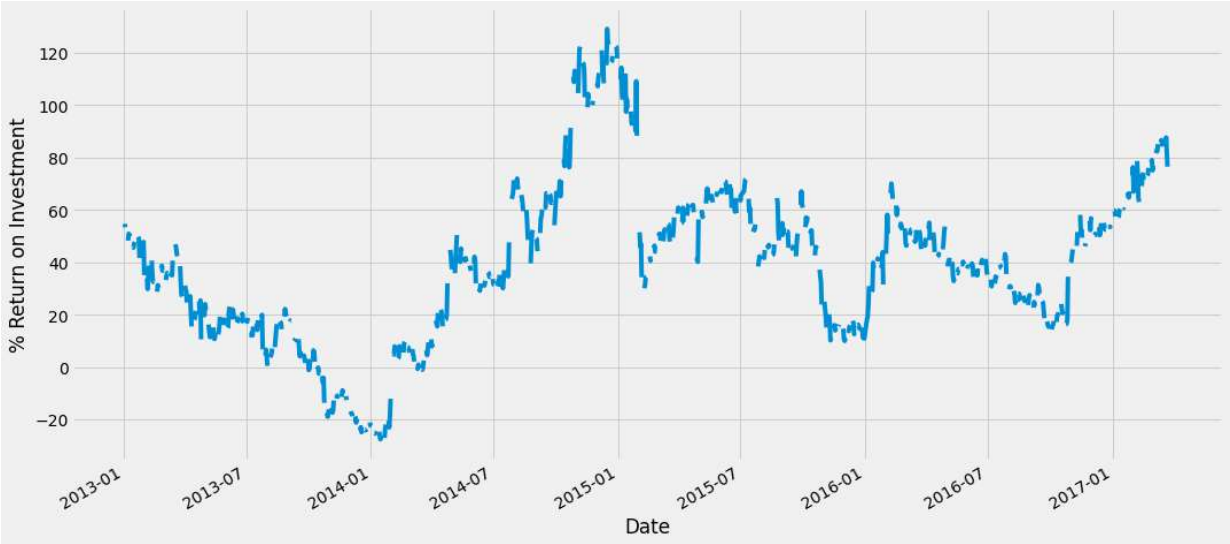
1316 rows × 6 columns

Application: Computing Return on investment

In [26]:
```
ROI=100*(data['Adj_Close'].tshift(periods=-365,freq='D')/data['Adj_Close']-1)
ROI.plot(figsize=(16,8))
plt.ylabel('% Return on Investment')
```

```
C:\Users\1MSCDS~1\AppData\Local\Temp/ipykernel_7128/334012724.py:1: FutureWarni
ng: tshift is deprecated and will be removed in a future version. Please use sh
ift instead.
  ROI=100*(data['Adj_Close'].tshift(periods=-365,freq='D')/data['Adj_Close']-1)
```

Out[26]:
```
Text(0, 0.5, '% Return on Investment')
```



3. Rolling Window or Moving Window Operations

Time series data can be noisy due to high fluctuations in the market. As a result, it becomes difficult to gauge a trend or pattern in the data. Here is a visualization of the Amazon's adjusted close price over the years where we can see such noise (ie, line is not smooth).

In [27]:
```
data['Adj_Close'].plot(figsize=(12,8),color='r')
```

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

Find rolling mean for 7 days and show top-10 rows

In [28]:
```
data.rolling(7).mean().head(10)
```

Out[28]:

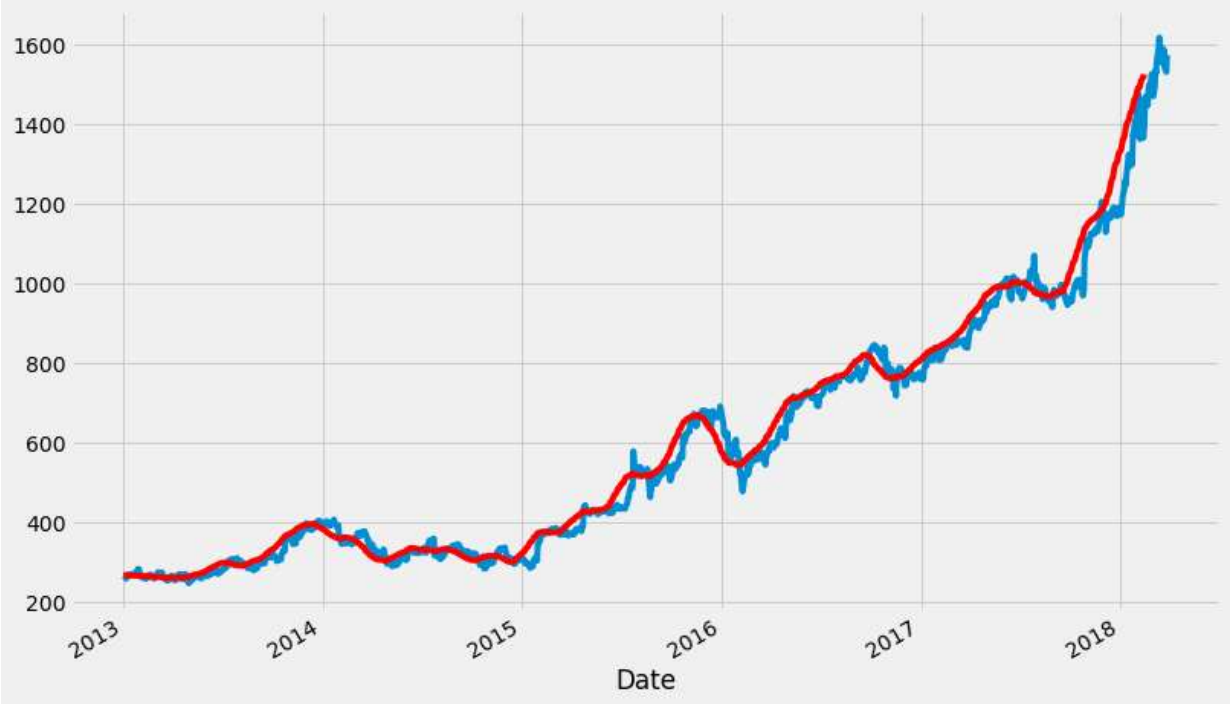
Date	Open	High	Low	Close	Volume	Adj_Close
2018-03-27	NaN	NaN	NaN	NaN	NaN	NaN
2018-03-26	NaN	NaN	NaN	NaN	NaN	NaN
2018-03-23	NaN	NaN	NaN	NaN	NaN	NaN
2018-03-22	NaN	NaN	NaN	NaN	NaN	NaN
2018-03-21	NaN	NaN	NaN	NaN	NaN	NaN
2018-03-20	NaN	NaN	NaN	NaN	NaN	NaN
2018-03-19	1556.885714	1570.640000	1521.894286	1543.695714	5.987651e+06	1543.695714
2018-03-16	1558.464286	1572.565714	1534.062857	1554.357143	5.752191e+06	1554.357143
2018-03-15	1567.750000	1578.268571	1545.328571	1558.137143	5.534923e+06	1558.137143
2018-03-14	1576.034286	1586.471429	1558.975714	1571.771429	5.009270e+06	1571.771429

Plot a line char for "Open" column.

Followed by, average rolling window of 30 days on the same "Open" column

In [29]:
```
data["Open"].plot(figsize=(12,8))
data['Open'].rolling(30).mean().plot(figsize=(12,8),color='r')
```

Out[29]: <AxesSubplot:xlabel='Date'>



Remember, first 29 days aren't going to have the blue line because there wasn't enough data to actually calculate that rolling mean.

In [ ]:
```

```