Bitcoin Data Analysis

Analysis of bitcoin dataset throughout the year 2013-2017. We will be determining the change in stock price, the open, high, low and close of the share price

Importing required libraries and reading the data

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
bitcoin_df = pd.read_csv(r'C:\Users\shukumar25\Desktop\My Projects\Bitcoin Data Analysis\Dataset/bitcoin_price.
```

In [3]:

```
bitcoin_df.head(3)
```

Out[3]:

	Date	Open	High	Low	Close	Volume	Market Cap
0	Jul 31, 2017	2763.24	2889.62	2720.61	2875.34	860,575,000	45,535,800,000
1	Jul 30, 2017	2724.39	2758.53	2644.85	2757.18	705,943,000	44,890,700,000
2	Jul 29, 2017	2807.02	2808.76	2692.80	2726.45	803,746,000	46,246,700,000

In [4]:

```
bitcoin_df.columns
```

Out[4]:

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Market Cap'], dtype='object')
```

In [5]:

```
bitcoin_df.shape
```

Out[5]:

```
(1556, 7)
```

In [6]:

```
bitcoin_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1556 entries, 0 to 1555
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Date        1556 non-null   object 
 1   Open        1556 non-null   float64
 2   High        1556 non-null   float64
 3   Low         1556 non-null   float64
 4   Close       1556 non-null   float64
 5   Volume      1556 non-null   object 
 6   Market Cap  1556 non-null   object 
dtypes: float64(4), object(3)
memory usage: 85.2+ KB
```

In [7]:

```
bitcoin_df.describe()
```

Out[7]:

	Open	High	Low	Close
count	1556.000000	1556.000000	1556.000000	1556.000000
mean	582.625328	597.992847	567.851446	584.239396
std	523.137312	542.992855	505.877401	525.904442
min	68.500000	74.560000	65.530000	68.430000
25%	254.287500	260.327500	248.835000	254.320000
50%	438.600000	447.560000	430.570000	438.855000
75%	662.437500	674.525000	646.735000	663.402500
max	2953.220000	2999.910000	2840.530000	2958.110000

In [8]:

```
bitcoin_df.describe().T   # Transpose of the bitcoin_df table
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
Open	1556.0	582.625328	523.137312	68.50	254.2875	438.600	662.4375	2953.22
High	1556.0	597.992847	542.992855	74.56	260.3275	447.560	674.5250	2999.91
Low	1556.0	567.851446	505.877401	65.53	248.8350	430.570	646.7350	2840.53
Close	1556.0	584.239396	525.904442	68.43	254.3200	438.855	663.4025	2958.11

Data Pre-processing

Checking if there is any duplicate or null values in the dataset and checking for the appropriate data type

In [9]:

```
bitcoin_df.dtypes
```

Out[9]:

```
Date           object
Open          float64
High          float64
Low           float64
Close         float64
Volume         object
Market Cap     object
dtype: object
```

In [10]:

```
bitcoin_df['Date'] = bitcoin_df['Date'].astype('datetime64[ns]')

## pd.to_datetime()
```

In [11]:

```
bitcoin_df['Date'].min()
```

Out[11]:

```
Timestamp('2013-04-28 00:00:00')
```

In [12]:

```
bitcoin_df['Date'].max()
```

Out[12]:

```
Timestamp('2017-07-31 00:00:00')
```

In [13]:

```
bitcoin_df['Date']
```

Out[13]:

```
0      2017-07-31
1      2017-07-30
2      2017-07-29
3      2017-07-28
4      2017-07-27
          ...
1551   2013-05-02
1552   2013-05-01
1553   2013-04-30
1554   2013-04-29
1555   2013-04-28
Name: Date, Length: 1556, dtype: datetime64[ns]
```

In [14]:

```
type(bitcoin_df['Date'][0])
```

Out[14]:

```
pandas._libs.tslibs.timestamps.Timestamp
```

In [15]:

```
bitcoin_df.isnull().sum()
```

Out[15]:

```
Date          0
Open          0
High          0
Low           0
Close         0
Volume        0
Market Cap    0
dtype: int64
```

In [16]:

```
bitcoin_df.duplicated().sum()
```

Out[16]:

```
0
```

In [17]:

```
bitcoin_df.head(5)
```

Out[17]:

	Date	Open	High	Low	Close	Volume	Market Cap
0	2017-07-31	2763.24	2889.62	2720.61	2875.34	860,575,000	45,535,800,000
1	2017-07-30	2724.39	2758.53	2644.85	2757.18	705,943,000	44,890,700,000
2	2017-07-29	2807.02	2808.76	2692.80	2726.45	803,746,000	46,246,700,000
3	2017-07-28	2679.73	2897.45	2679.73	2809.01	1,380,100,000	44,144,400,000
4	2017-07-27	2538.71	2693.32	2529.34	2671.78	789,104,000	41,816,500,000

In [18]:

```
bitcoin_df.tail(5)
```

Out[18]:

	Date	Open	High	Low	Close	Volume	Market Cap
1551	2013-05-02	116.38	125.60	92.28	105.21	-	1,292,190,000
1552	2013-05-01	139.00	139.89	107.72	116.99	-	1,542,820,000
1553	2013-04-30	144.00	146.93	134.05	139.00	-	1,597,780,000
1554	2013-04-29	134.44	147.49	134.00	144.54	-	1,491,160,000
1555	2013-04-28	135.30	135.98	132.10	134.21	-	1,500,520,000

In [19]:

```
bitcoin_df.sort_index(ascending = False)  # sorting the data from oldest to newest
```

Out[19]:

	Date	Open	High	Low	Close	Volume	Market Cap
1555	2013-04-28	135.30	135.98	132.10	134.21	-	1,500,520,000
1554	2013-04-29	134.44	147.49	134.00	144.54	-	1,491,160,000
1553	2013-04-30	144.00	146.93	134.05	139.00	-	1,597,780,000
1552	2013-05-01	139.00	139.89	107.72	116.99	-	1,542,820,000
1551	2013-05-02	116.38	125.60	92.28	105.21	-	1,292,190,000
...	...	...	...	...	...	...	...
4	2017-07-27	2538.71	2693.32	2529.34	2671.78	789,104,000	41,816,500,000
3	2017-07-28	2679.73	2897.45	2679.73	2809.01	1,380,100,000	44,144,400,000
2	2017-07-29	2807.02	2808.76	2692.80	2726.45	803,746,000	46,246,700,000
1	2017-07-30	2724.39	2758.53	2644.85	2757.18	705,943,000	44,890,700,000
0	2017-07-31	2763.24	2889.62	2720.61	2875.34	860,575,000	45,535,800,000

1556 rows × 7 columns

Analyzing and Visualizing the Data

1. Change in price of the stock overtime

In [20]:

```
data = bitcoin_df.sort_index(ascending = False).reset_index()
```

In [21]:

```
data.drop('index', axis = 1, inplace = True)
```

In [22]:

```
data
```

Out[22]:

	Date	Open	High	Low	Close	Volume	Market Cap
0	2013-04-28	135.30	135.98	132.10	134.21	-	1,500,520,000
1	2013-04-29	134.44	147.49	134.00	144.54	-	1,491,160,000
2	2013-04-30	144.00	146.93	134.05	139.00	-	1,597,780,000
3	2013-05-01	139.00	139.89	107.72	116.99	-	1,542,820,000
4	2013-05-02	116.38	125.60	92.28	105.21	-	1,292,190,000
...	...	...	...	...	...	...	...
1551	2017-07-27	2538.71	2693.32	2529.34	2671.78	789,104,000	41,816,500,000
1552	2017-07-28	2679.73	2897.45	2679.73	2809.01	1,380,100,000	44,144,400,000
1553	2017-07-29	2807.02	2808.76	2692.80	2726.45	803,746,000	46,246,700,000
1554	2017-07-30	2724.39	2758.53	2644.85	2757.18	705,943,000	44,890,700,000
1555	2017-07-31	2763.24	2889.62	2720.61	2875.34	860,575,000	45,535,800,000

1556 rows × 7 columns

In [23]:

```
data.columns
```

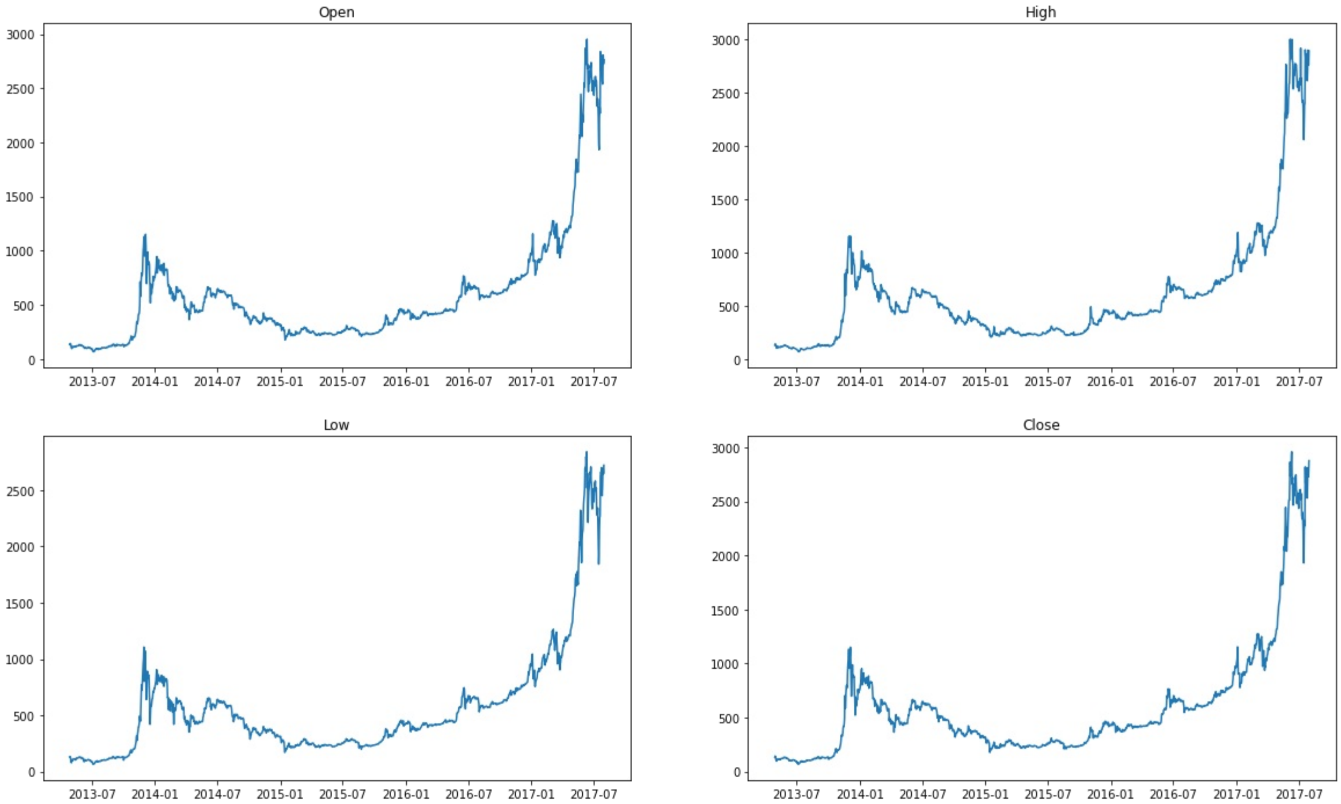
Out[23]:

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Market Cap'], dtype='object')
```

In [24]:

```
plt.figure(figsize = (20, 12))

for index, col in enumerate(['Open', 'High', 'Low', 'Close'], 1):
    plt.subplot(2, 2, index)
    plt.plot(bitcoin_df['Date'], bitcoin_df[col])
    plt.title(col)
```



Conclusion : From the above visualization, we can observe that the price of the bitcoin has increased throughtout year 2013-2017. The price however took a downward trend during the year 2015-2016 but it again showed an upward trend after 2016

2. Analyzing open, high, low, close value of Bitcoin

In [25]:
```
data.shape
```

Out[25]:
```
(1556, 7)
```

In [26]:
```
bitcoin_sample = data[0 : 50]
```

In [ ]:
```
!pip install chart_studio
!pip install plotly
```

In [27]:
```
# importing libraries for viisualization
import chart_studio.plotly as py

import plotly.graph_objs as go

import plotly.express as px

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
```

In [28]:
```
init_notebook_mode(connected = True)
```

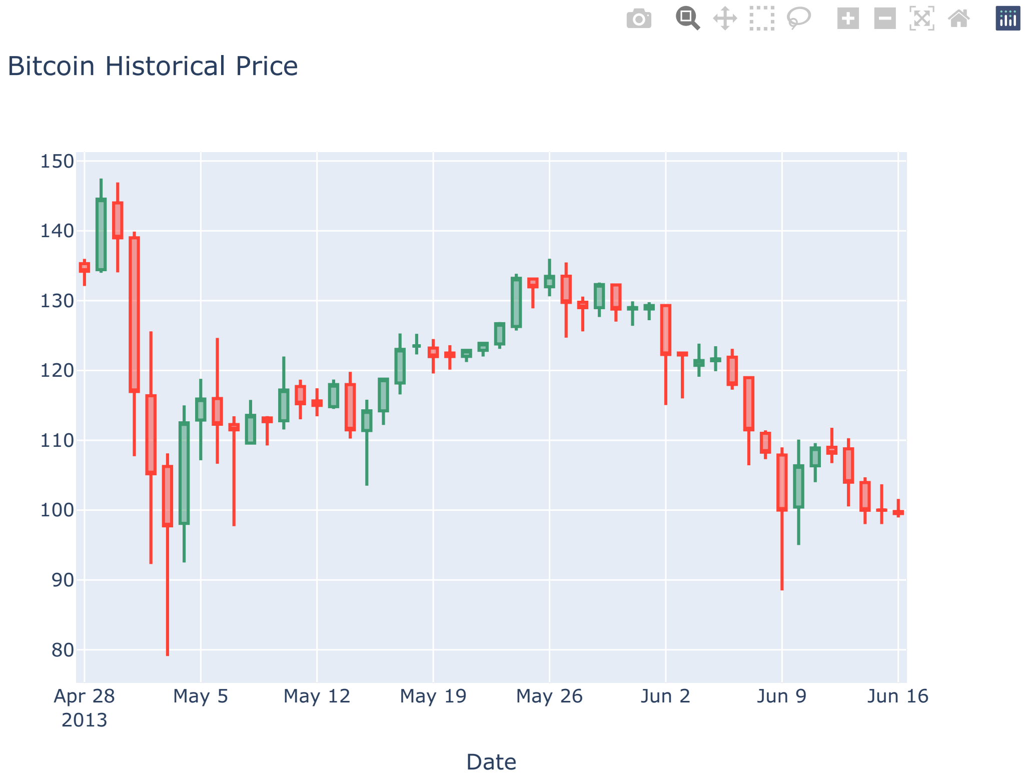
In [29]:
```
trace = go.Candlestick(x = bitcoin_sample['Date'],
              high = bitcoin_sample['High'],
              open = bitcoin_sample['Open'],
              close = bitcoin_sample['Close'],
              low = bitcoin_sample['Low'])
```

In [30]:
```
candle_data = [trace]

layout = {
    'title' : 'Bitcoin Historical Price',
    'xaxis' : {'title' : 'Date'}
}
```

In [31]:
```
fig = go.Figure(data = candle_data, layout = layout)

fig.update_layout(xaxis_rangeslider_visible = False)
fig.show()
```

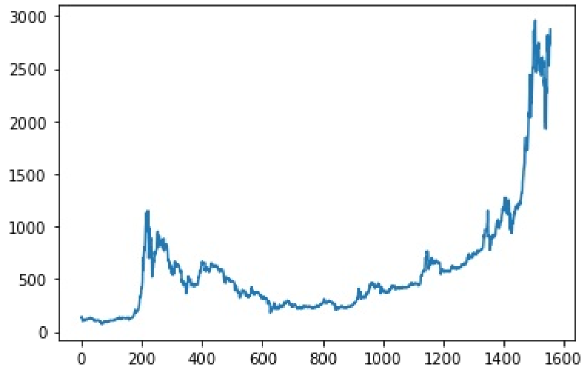


Conclusion : The above visualization gives us the open, low, high, close of the share for a particular date.

3. Analyzing Closing Price on normal scale & log-scale¶

In [32]:
```
data['Close'].plot()
```

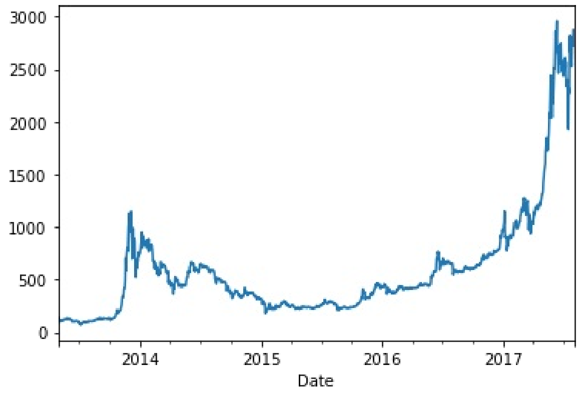
Out[32]:
```
<AxesSubplot:>
```



In [33]:
```
data.set_index('Date', inplace = True)
```

In [34]:
```
data['Close'].plot()
```

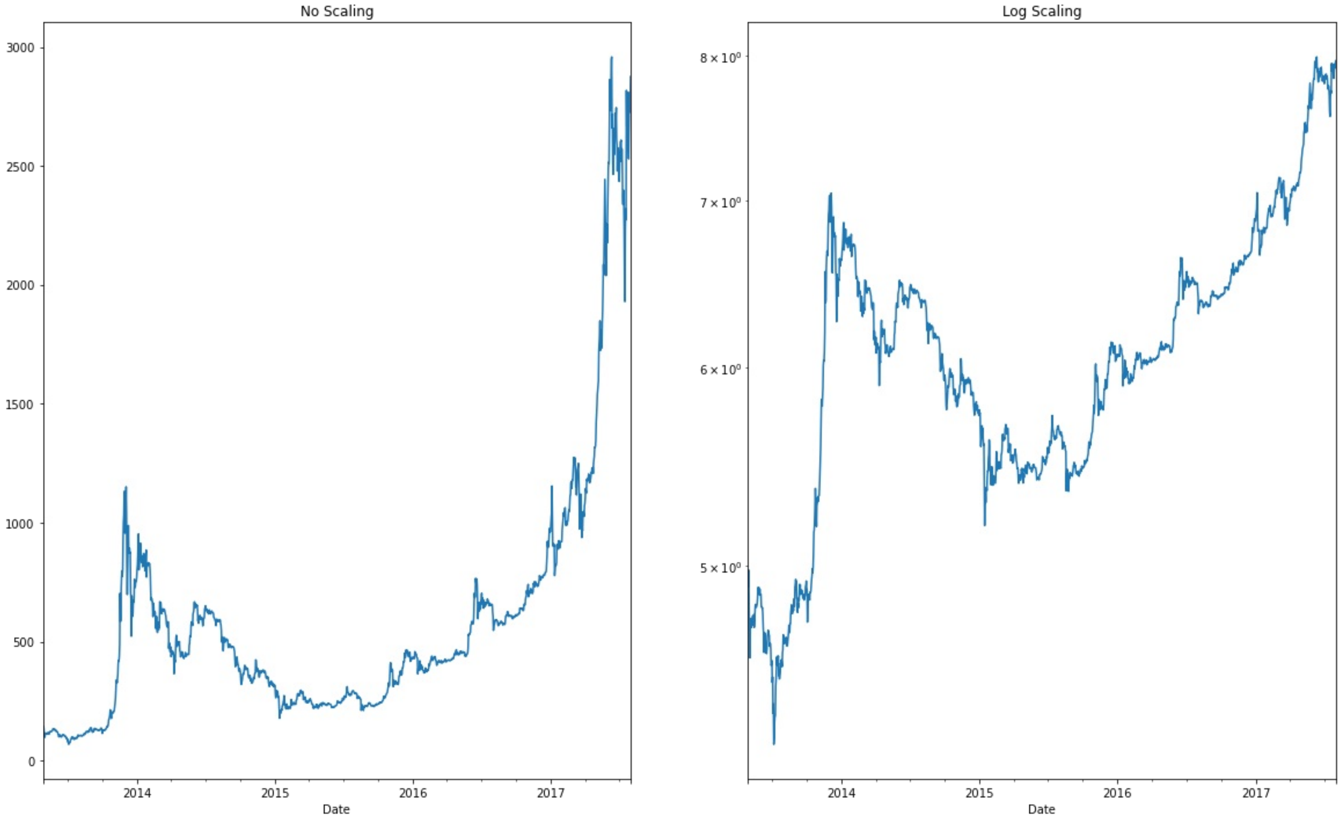
Out[34]:
```
<AxesSubplot:xlabel='Date'>
```



In [35]:
```
plt.figure(figsize = (20, 12))
```

```python
plt.subplot(1, 2, 1)
data['Close'].plot()
plt.title('No Scaling')

plt.subplot(1, 2, 2)
np.log1p(data['Close']).plot()
plt.title('Log Scaling')
plt.yscale('log')
```



4. Analyzing Closing Price on Yearly, Quarterly, monthly basis

In [36]:

```python
data.head(4)
```

Out[36]:

Date	Open	High	Low	Close	Volume	Market Cap
2013-04-28	135.30	135.98	132.10	134.21	-	1,500,520,000
2013-04-29	134.44	147.49	134.00	144.54	-	1,491,160,000
2013-04-30	144.00	146.93	134.05	139.00	-	1,597,780,000
2013-05-01	139.00	139.89	107.72	116.99	-	1,542,820,000

In [37]:

```python
data['Close'].resample('Y').mean() # resampling date feature and finding the avg price of bitcoin on yearly bas
```

Out[37]:

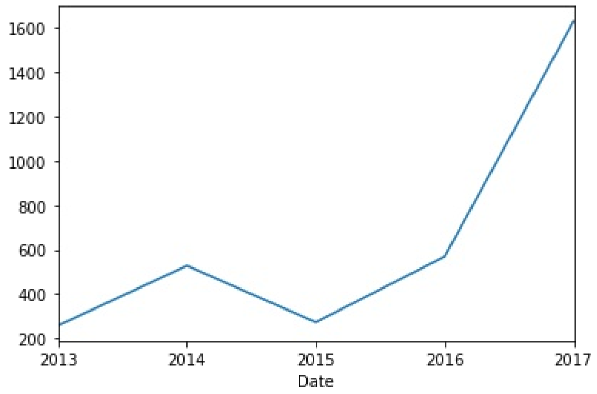
```
Date
2013-12-31     257.474476
2014-12-31     527.236658
2015-12-31     272.453260
2016-12-31     568.492131
2017-12-31    1628.622123
Freq: A-DEC, Name: Close, dtype: float64
```

In [38]:

```python
data['Close'].resample('Y').mean().plot() # Closing price yearly
```

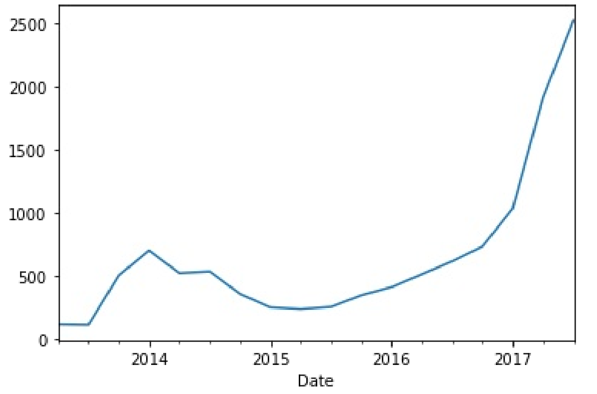
Out[38]:

```
<AxesSubplot:xlabel='Date'>
```



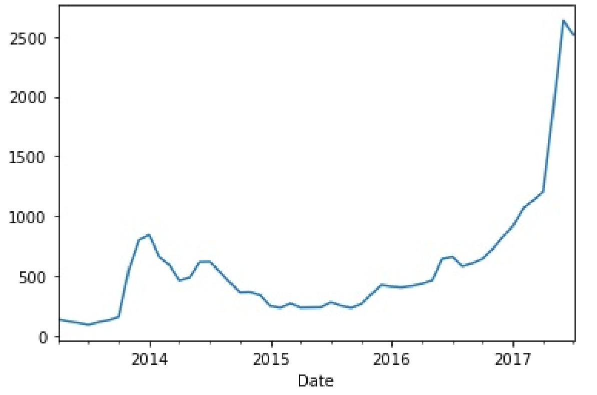
In [39]:
```
data['Close'].resample('Q').mean().plot() # Closing price Quarterly
```

Out[39]:
```
<AxesSubplot:xlabel='Date'>
```



In [40]:
```
data['Close'].resample('M').mean().plot() # Closing price Monthly
```

Out[40]:
```
<AxesSubplot:xlabel='Date'>
```



5. Daily change in Closing price of stocks

To calculate the gain or lost per day for a stock, subtract the opening price from the closing price. Then, multiply the result by the number of shares that we own in the company.

In [41]:
```
data['Close']
```

Out[41]:

```
Date
2013-04-28     134.21
2013-04-29     144.54
2013-04-30     139.00
2013-05-01     116.99
2013-05-02     105.21
               ...
2017-07-27    2671.78
2017-07-28    2809.01
2017-07-29    2726.45
2017-07-30    2757.18
2017-07-31    2875.34
Name: Close, Length: 1556, dtype: float64
```

In [42]:

```
data['Close_price_pct_change'] = data['Close'].pct_change()*100 # calculating the percentage change
```

In [43]:

```
data['Close_price_pct_change']
```

Out[43]:

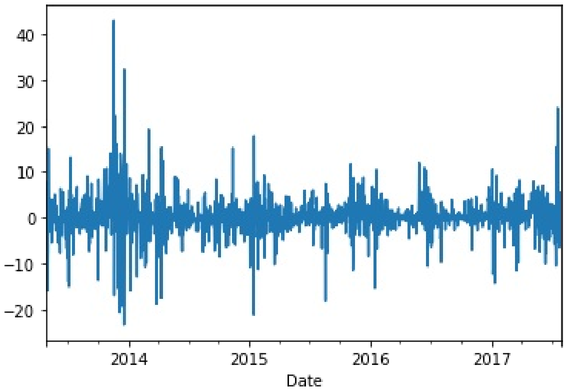
```
Date
2013-04-28          NaN
2013-04-29     7.696893
2013-04-30    -3.832849
2013-05-01   -15.834532
2013-05-02   -10.069237
                ...
2017-07-27     5.626915
2017-07-28     5.136276
2017-07-29    -2.939114
2017-07-30     1.127107
2017-07-31     4.285538
Name: Close_price_pct_change, Length: 1556, dtype: float64
```

In [44]:

```
data['Close_price_pct_change'].plot()
```

Out[44]:

```
<AxesSubplot:xlabel='Date'>
```



In [45]:

```
import chart_studio.plotly as py  # chart_studio provides a web-service for hosting graphs!
import plotly.graph_objs as go
import plotly.express as px
from plotly.offline import download_plotlyjs , init_notebook_mode , plot , iplot
init_notebook_mode(connected=True)
```

In [46]:

```
conda install -c conda-forge python-cufflinks
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.

Retrieving notices: ...working... done

Note: you may need to restart the kernel to use updated packages.
```

```

==> WARNING: A newer version of conda exists. <==
  current version: 4.14.0
  latest version: 23.5.0

Please update conda by running

    $ conda update -n base -c conda-forge conda

```

In [47]:

```
import cufflinks as cf
```

In [48]:

```
cf.go_offline()
```

In [49]:

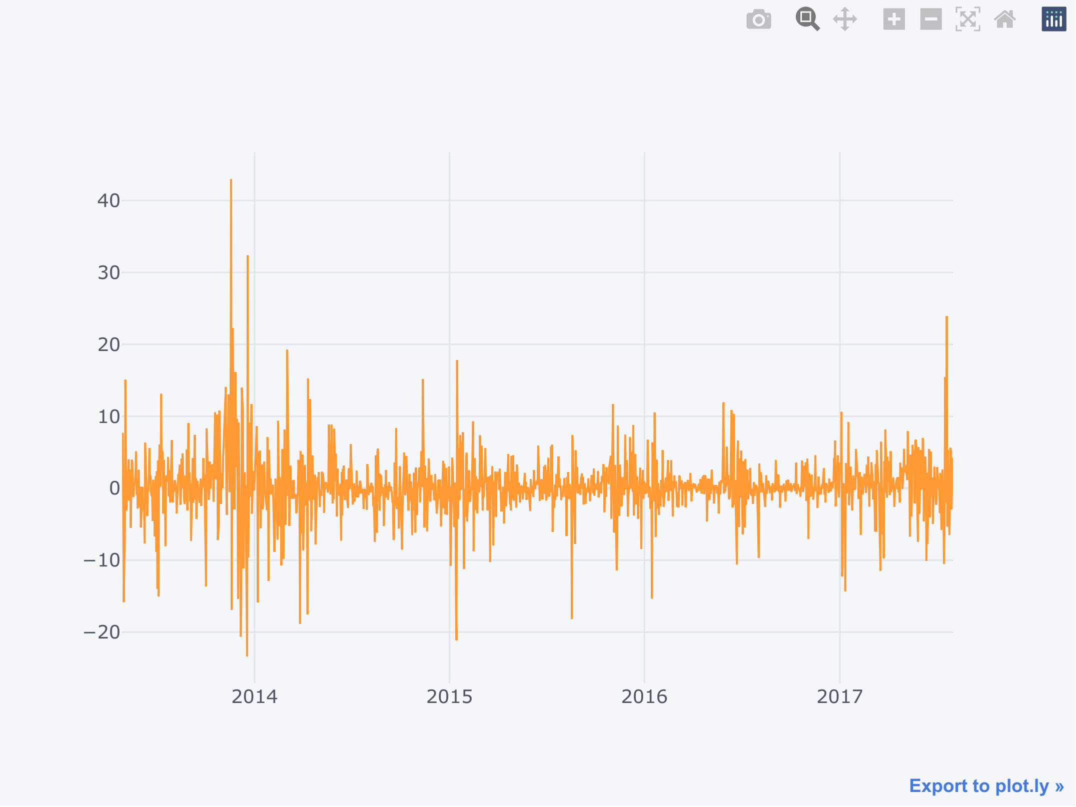
```
type(data['Close_price_pct_change'])
```

Out[49]:

```
pandas.core.series.Series
```

In [50]:

```
data['Close_price_pct_change'].iplot()
```



In [ ]:

Loading [MathJax]/extensions/Safe.js