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

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

```
data = pd.read_csv("/content/cs-1.csv")
```

```
dataAAPL = data.loc[data['Name']  == 'AAPL']
dataGOOG = data.loc[data['Name']  == 'GOOG']
dataMSFT = data.loc[data['Name']  == 'MSFT']
dataAMZN = data.loc[data['Name']  == 'AMZN']
```

```
dataAAPL.describe().T
```

	count	mean	std	min	25%	50%	75%	max
open	1259.0	1.090554e+02	3.054922e+01	5.542420e+01	8.464780e+01	108.97	1.273350e+02	1.793700e+02
high	1259.0	1.099511e+02	3.068619e+01	5.708570e+01	8.533495e+01	110.03	1.281000e+02	1.801000e+02
low	1259.0	1.081416e+02	3.037622e+01	5.501420e+01	8.425065e+01	108.05	1.262900e+02	1.782500e+02
close	1259.0	1.090667e+02	3.055681e+01	5.578990e+01	8.483065e+01	109.01	1.271200e+02	1.792600e+02
volume	1259.0	5.404790e+07	3.346835e+07	1.147592e+07	2.969438e+07	45668931.00	6.870872e+07	2.668336e+08

```
dataGOOG.describe().T
```

	count	mean	std	min	25%	50%	75%	max
open	975.0	7.253642e+02	165.996590	494.650	565.113	722.71	822.035	1177.33
high	975.0	7.308222e+02	166.847404	495.976	570.380	727.00	826.185	1186.89
low	975.0	7.194568e+02	165.526487	487.560	559.055	716.43	818.725	1171.98
close	975.0	7.254034e+02	166.420529	492.550	564.785	720.64	823.330	1175.84
volume	975.0	1.808414e+06	947968.484651	7932.000	1261927.000	1576830.00	2052652.000	11164943.00

```
dataMSFT.describe().T
```

	count	mean	std	min	25%	50%	75%	max
open	1259.0	5.102639e+01	1.485939e+01	27.35	4.030500e+01	4.744000e+01	5.995500e+01	9.514000e+01
high	1259.0	5.143601e+01	1.493014e+01	27.60	4.063750e+01	4.781000e+01	6.043500e+01	9.607000e+01
low	1259.0	5.063040e+01	1.477463e+01	27.23	3.987000e+01	4.700500e+01	5.927500e+01	9.372000e+01
close	1259.0	5.106308e+01	1.485212e+01	27.37	4.031000e+01	4.752000e+01	5.973000e+01	9.501000e+01
volume	1259.0	3.386946e+07	1.958979e+07	7425603.00	2.254879e+07	2.938758e+07	3.842024e+07	2.483542e+08

```
dataAMZN.describe().T
```

	count	mean	std	min	25%	50%	75%	max
open	1259.0	5.768673e+02	2.825000e+02	248.94	325.870	506.00	777.620	1477.39
high	1259.0	5.820172e+02	2.844171e+02	252.93	329.485	512.33	781.845	1498.00
low	1259.0	5.711135e+02	2.802152e+02	245.75	322.185	495.64	770.720	1450.04
close	1259.0	5.768800e+02	2.825004e+02	248.23	325.800	503.82	777.420	1450.89
volume	1259.0	3.730465e+06	2.166506e+06	1092970.00	2511165.000	3144719.00	4220246.500	23856060.00

```
dataAAPL.info()
```

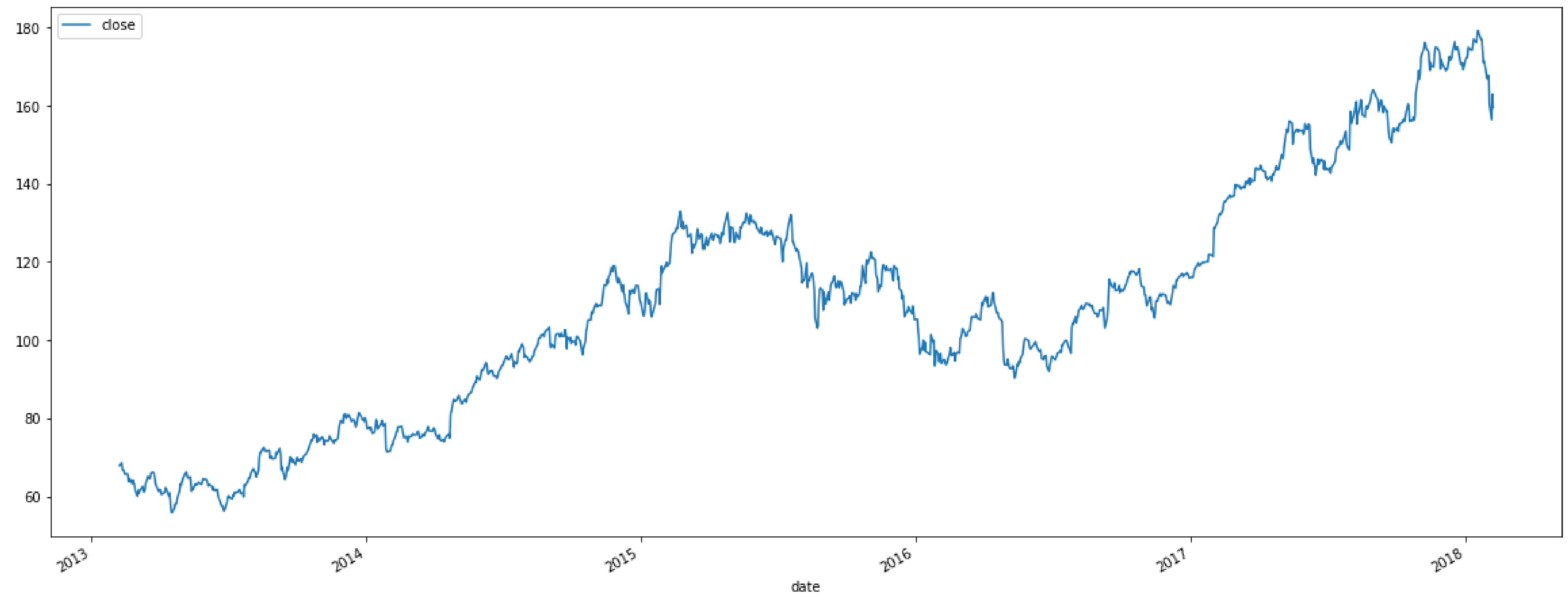
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1259 entries, 1259 to 2517
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   date    1259 non-null   object 
 1   open    1259 non-null   float64
 2   high    1259 non-null   float64
 3   low     1259 non-null   float64
 4   close   1259 non-null   float64
 5   volume  1259 non-null   int64  
 6   Name    1259 non-null   object 
dtypes: float64(4), int64(1), object(2)
memory usage: 78.7+ KB
```

```
dataAAPL['date']=pd.to_datetime(dataAAPL['date'])
dataGOOG['date']=pd.to_datetime(dataGOOG['date'])
dataMSFT['date']=pd.to_datetime(dataMSFT['date'])
dataAMZN['date']=pd.to_datetime(dataAMZN['date'])
dataAAPL.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1259 entries, 1259 to 2517
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   date    1259 non-null   datetime64[ns]
 1   open    1259 non-null   float64       
 2   high    1259 non-null   float64       
 3   low     1259 non-null   float64       
 4   close   1259 non-null   float64       
 5   volume  1259 non-null   int64         
 6   Name    1259 non-null   object        
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 78.7+ KB
```

We can see that the data column values have changed to datetime64 type

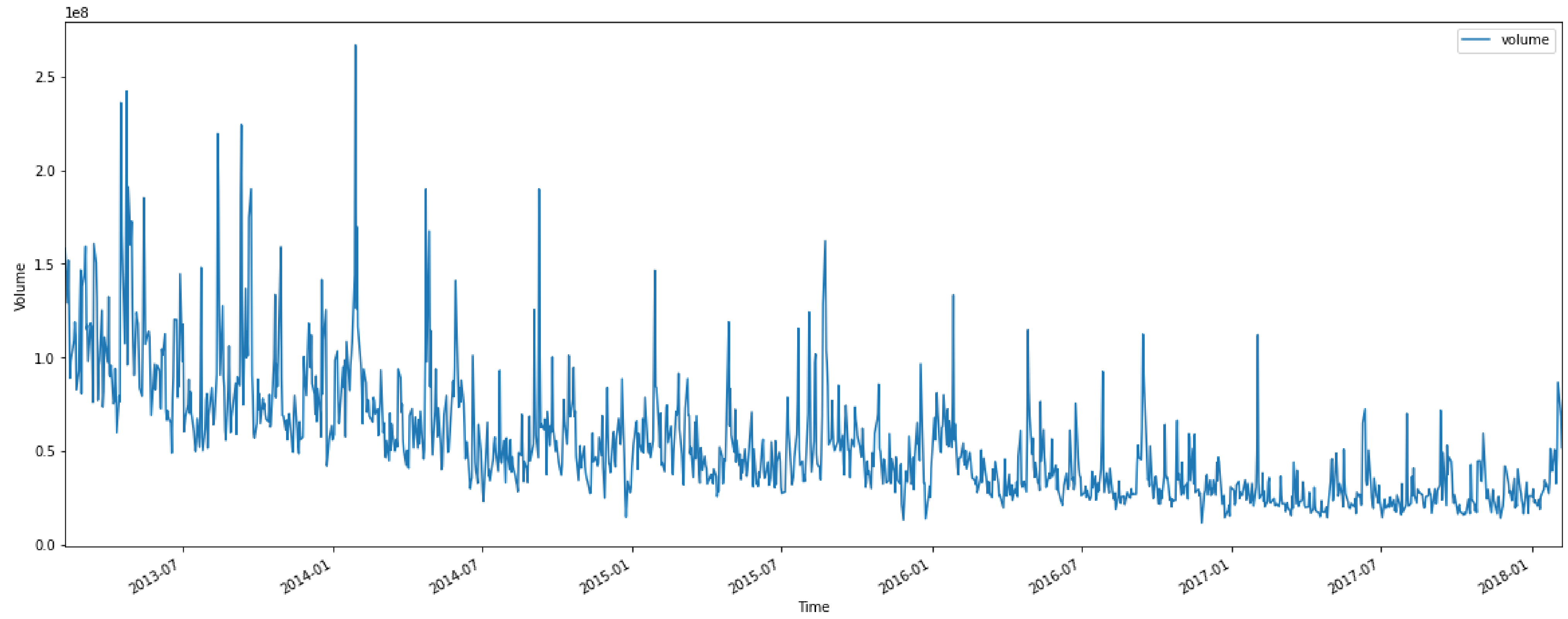
```
dataAAPL.plot(x='date', y='close',legend=True,figsize=(20,8))
plt.ioff()
```



We have stock Price for 5 years starting from 2013 to 2018

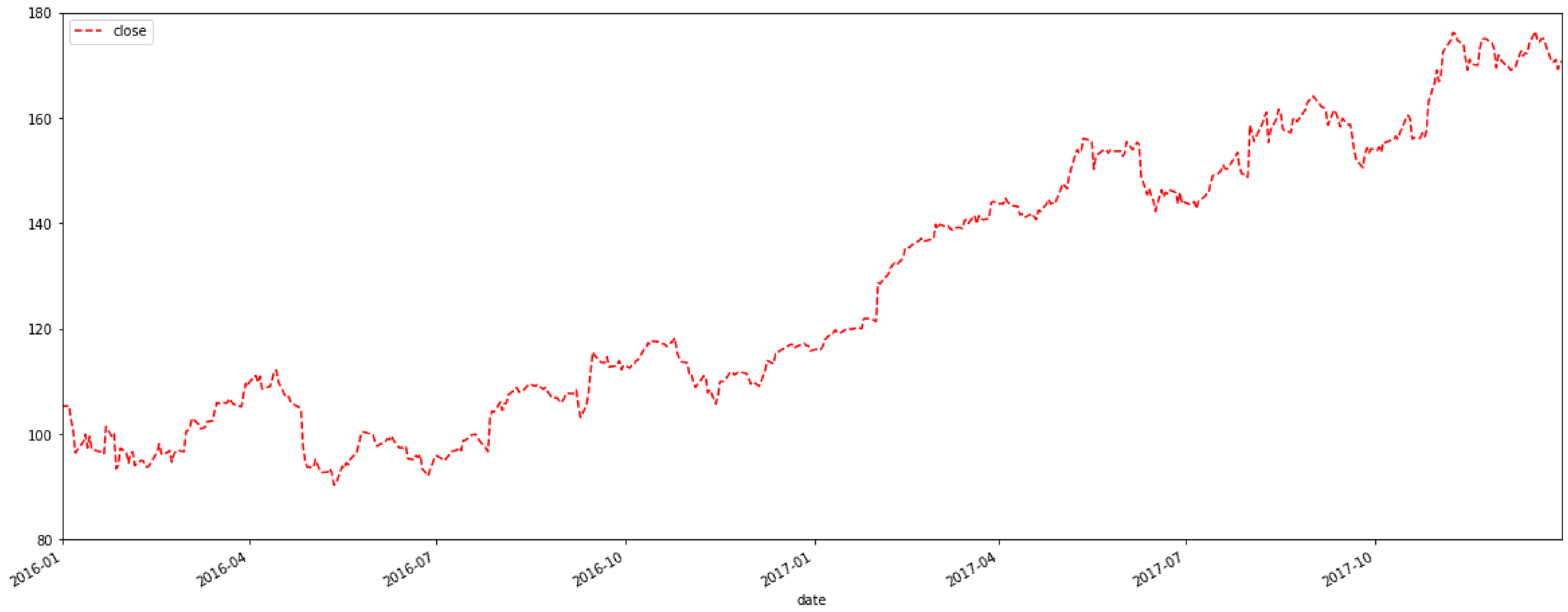
Volume traded for Apple Stock

```
title='VOLUME TRADED'
ylabel='Volume'
xlabel='Time'
ax=dataAAPL.plot(x='date', y='volume',legend=True,figsize=(20,8));
ax.autoscale(axis='x',tight=True)  # use both if want to scale both axis
ax.set(xlabel=xlabel,ylabel=ylabel)
plt.ioff()
```



Plotting between Specified time

```
dataAAPL.plot(x='date', y='close',xlim=['2016-01-01','2017-12-31'],ylim=[80,180],legend=True,figsize=(20,8),ls='--',c='red')
plt.ioff()
```

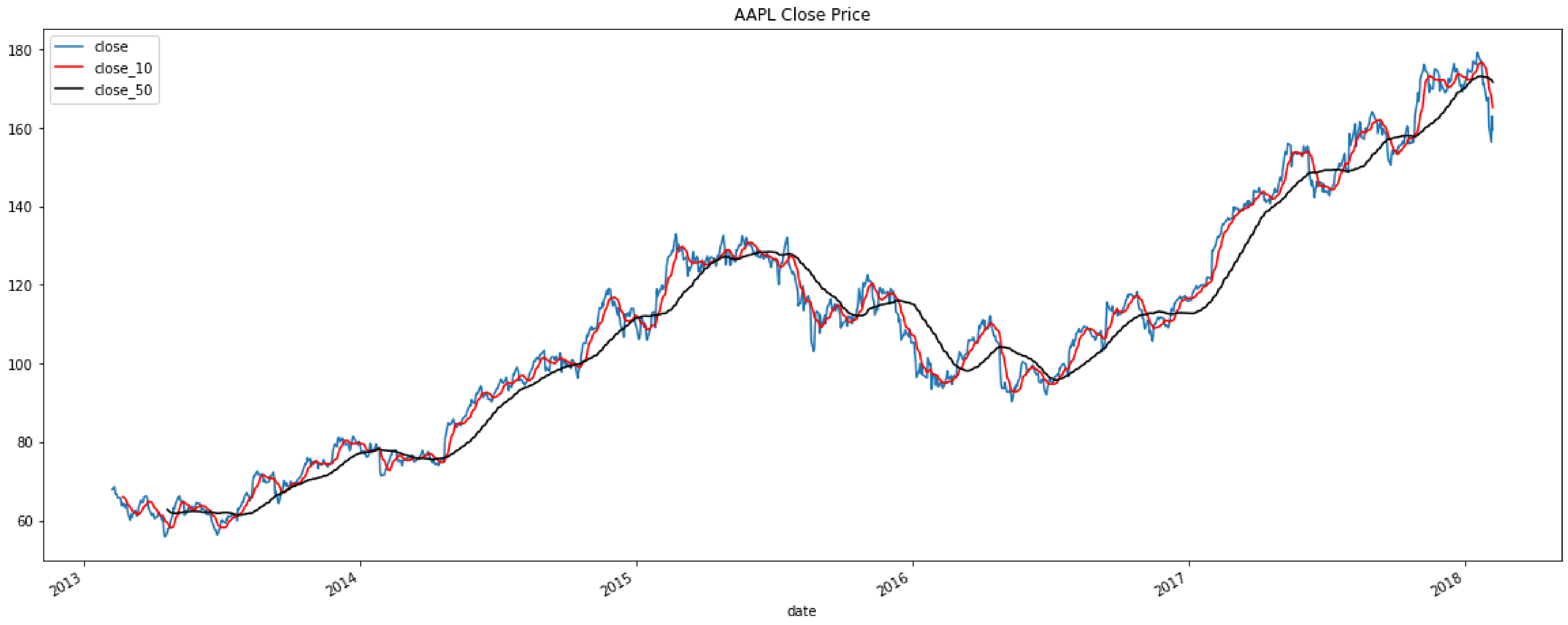


We have ploted the closing Price by specifying the range of dates xlim

Moving Average for Apple Stock

```
dataAAPL['close_10']=dataAAPL['close'].rolling(10).mean()
dataAAPL['close_50']=dataAAPL['close'].rolling(50).mean()
ax=dataAAPL.plot(x='date',y='close',title='AAPL Close Price',figsize=(20,8))
dataAAPL.plot(x='date',y='close_10',color='red',ax=ax)
dataAAPL.plot(x='date',y='close_50',color='k',ax=ax)
dataAAPL.plot(x='date',y='close_50',color='k',ax=ax)
plt.ioff()
```

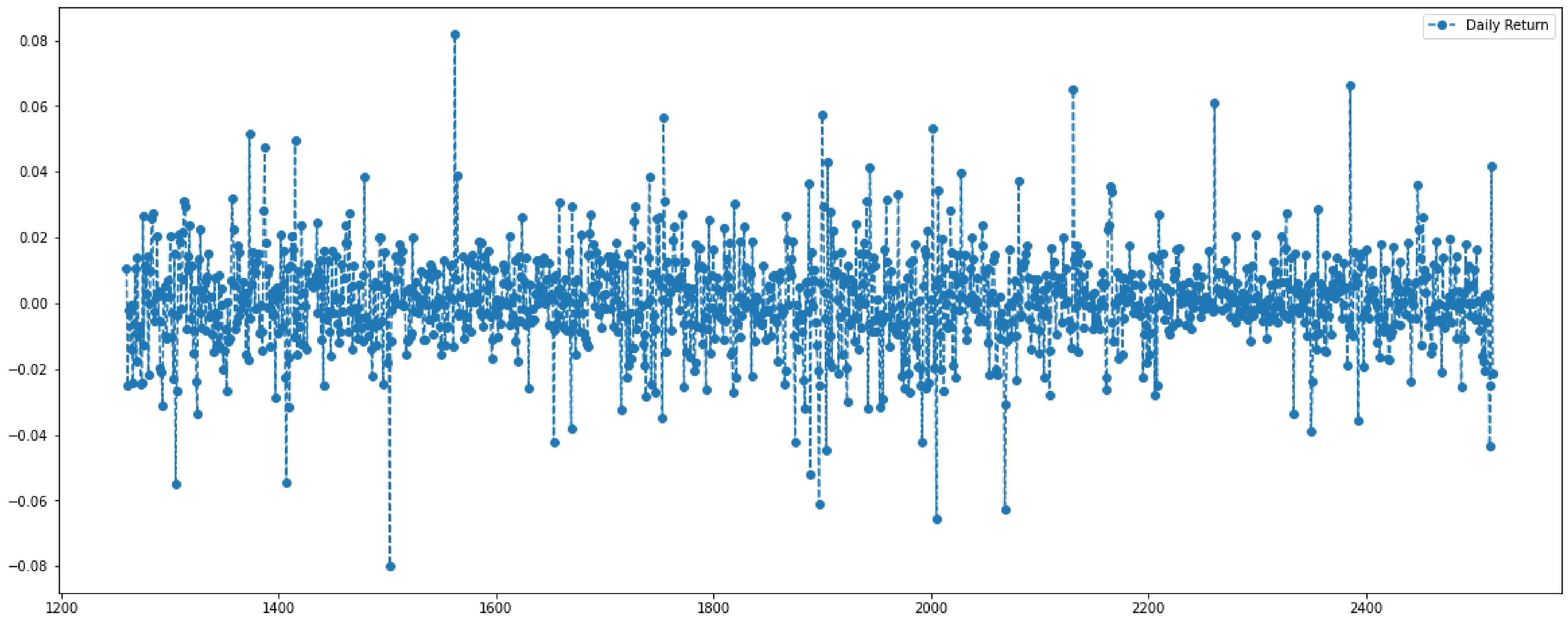
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```



Daily Returns

```
dataAAPL['Daily Return']=dataAAPL['close'].pct_change()
dataAAPL['Daily Return'].plot(figsize=(20,8),legend=True,linestyle='--',marker='o')
plt.ioff()
```

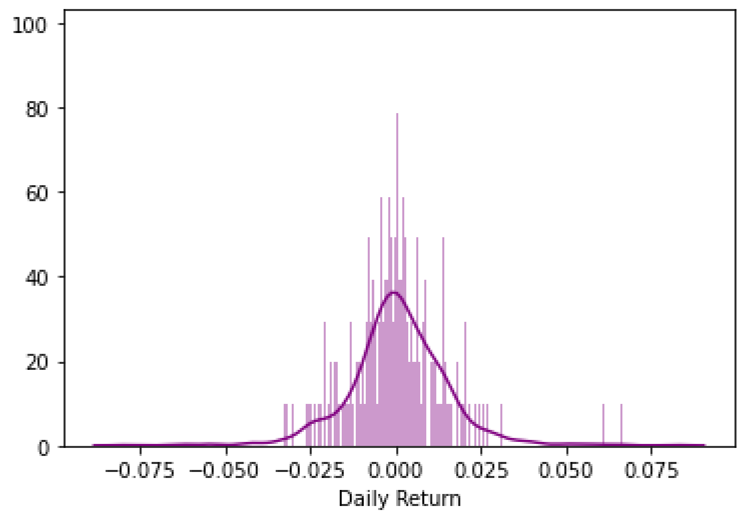
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```



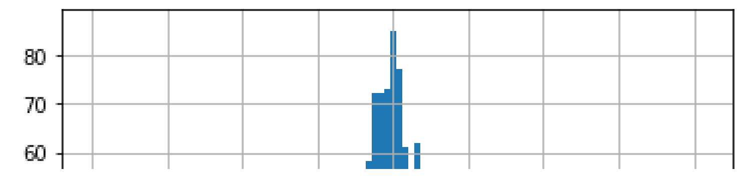
We can See maximum daily fluctuation in ths stock is 8 %

Average Daily return

```
sns.distplot(dataAAPL['Daily Return'].dropna(),bins=2000,color='purple')
plt.ioff()
```



```
dataAAPL['Daily Return'].hist(bins=100)
plt.ioff()
```



The above stock follows a normal distribution betweem +3% and -3%

Forecasting Apple Stock Price

```
df_prophet=dataAAPL[['date','close']]
df_prophet=df_prophet.sort_values('date')
df_prophet
```

	date	close
1259	2013-02-08	67.8542
1260	2013-02-11	68.5614
1261	2013-02-12	66.8428
1262	2013-02-13	66.7156
1263	2013-02-14	66.6556
...	...	...
2513	2018-02-01	167.7800
2514	2018-02-02	160.5000
2515	2018-02-05	156.4900
2516	2018-02-06	163.0300
2517	2018-02-07	159.5400

1259 rows × 2 columns

Renaiming the Column names to Suite Prophet Algorithm

```
df_prophet=df_prophet.rename(columns={'date':'ds','close':'y'})
df_prophet
```

	ds	y
1259	2013-02-08	67.8542
1260	2013-02-11	68.5614
1261	2013-02-12	66.8428
1262	2013-02-13	66.7156
1263	2013-02-14	66.6556
...	...	...
2513	2018-02-01	167.7800
2514	2018-02-02	160.5000
2515	2018-02-05	156.4900
2516	2018-02-06	163.0300
2517	2018-02-07	159.5400

1259 rows × 2 columns

Creating the Prophet Model

```
import random
import seaborn as sns
from fbprophet import Prophet
m=Prophet()
m.fit(df_prophet)
future=m.make_future_dataframe(periods=365)
forecast=m.predict(future)
forecast
```

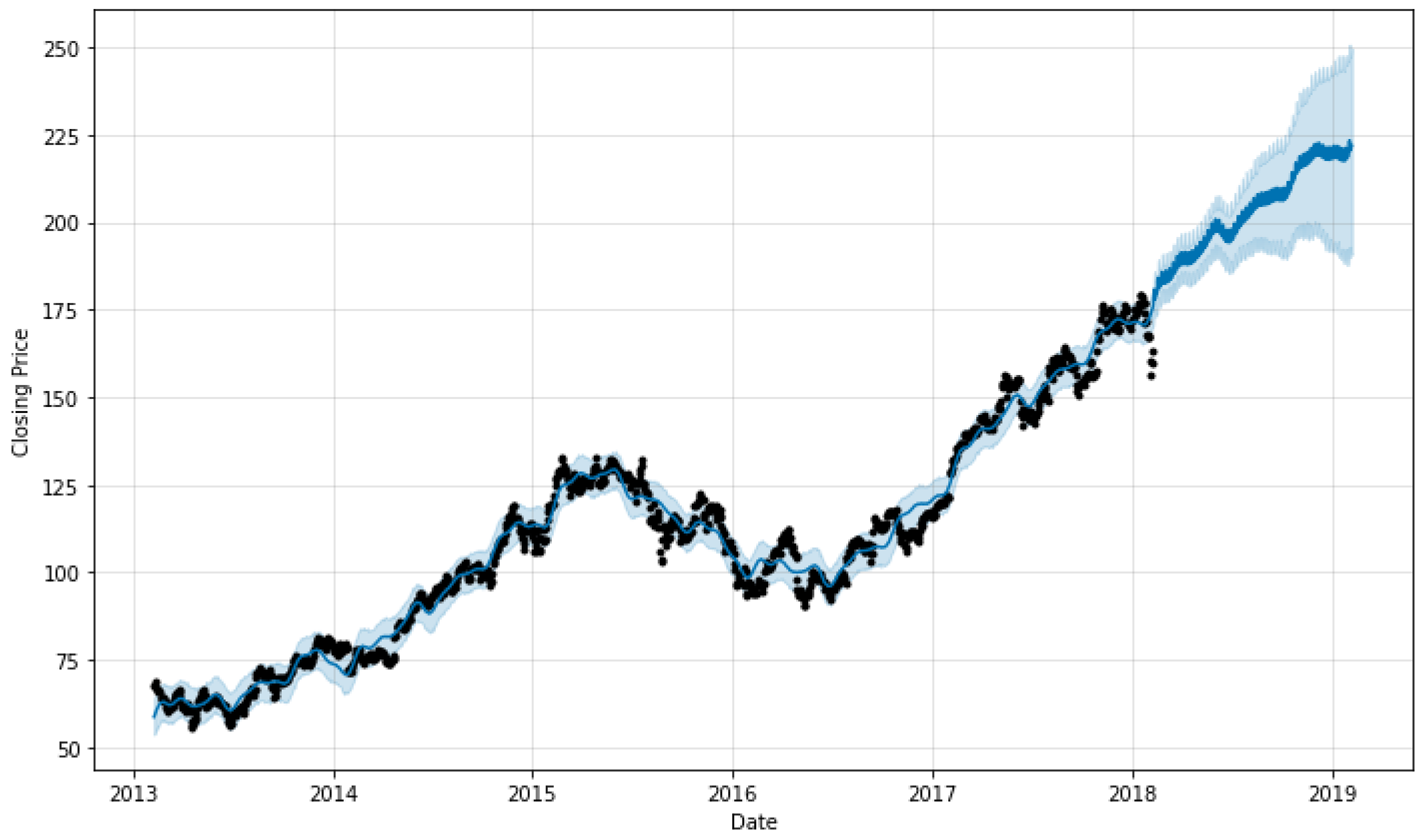
```
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
```

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	weekly	weekly_lower	weekly_upper	yearly	yearly
0	2013-02-08	62.556336	53.607446	63.990965	62.556336	62.556336	-3.874920	-3.874920	-3.874920	-1.064266	-1.064266	-1.064266	-2.810654	-2.
1	2013-02-11	62.540455	54.577254	65.596661	62.540455	62.540455	-2.458165	-2.458165	-2.458165	-0.937652	-0.937652	-0.937652	-1.520513	-1.
2	2013-02-12	62.535162	55.207134	66.024413	62.535162	62.535162	-2.025104	-2.025104	-2.025104	-0.916522	-0.916522	-0.916522	-1.108583	-1.
3	2013-02-13	62.529868	55.169506	65.718347	62.529868	62.529868	-1.704515	-1.704515	-1.704515	-0.989296	-0.989296	-0.989296	-0.715219	-0.
4	2013-02-14	62.524575	55.980263	66.425040	62.524575	62.524575	-1.381101	-1.381101	-1.381101	-1.035629	-1.035629	-1.035629	-0.345472	-0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1619	2019-02-03	225.875070	193.005682	250.912744	196.588915	253.048411	-2.468645	-2.468645	-2.468645	2.471683	2.471683	2.471683	-4.940327	-4.
1620	2019-02-04	226.003512	190.044397	247.321798	196.509165	253.283042	-5.545863	-5.545863	-5.545863	-0.937652	-0.937652	-0.937652	-4.608211	-4.
1621	2019-02-05	226.131954	190.827379	248.143109	196.429416	253.517672	-5.162036	-5.162036	-5.162036	-0.916522	-0.916522	-0.916522	-4.245514	-4.
1622	2019-02-06	226.260396	191.013323	249.909466	196.455640	253.737953	-4.846460	-4.846460	-4.846460	-0.989296	-0.989296	-0.989296	-3.857163	-3.
1623	2019-02-07	226.388838	190.773471	249.416493	196.505908	253.913183	-4.484224	-4.484224	-4.484224	-1.035629	-1.035629	-1.035629	-3.448595	-3.

1624 rows × 19 columns

Plotting the Apple Stock Forecast for Period of One year

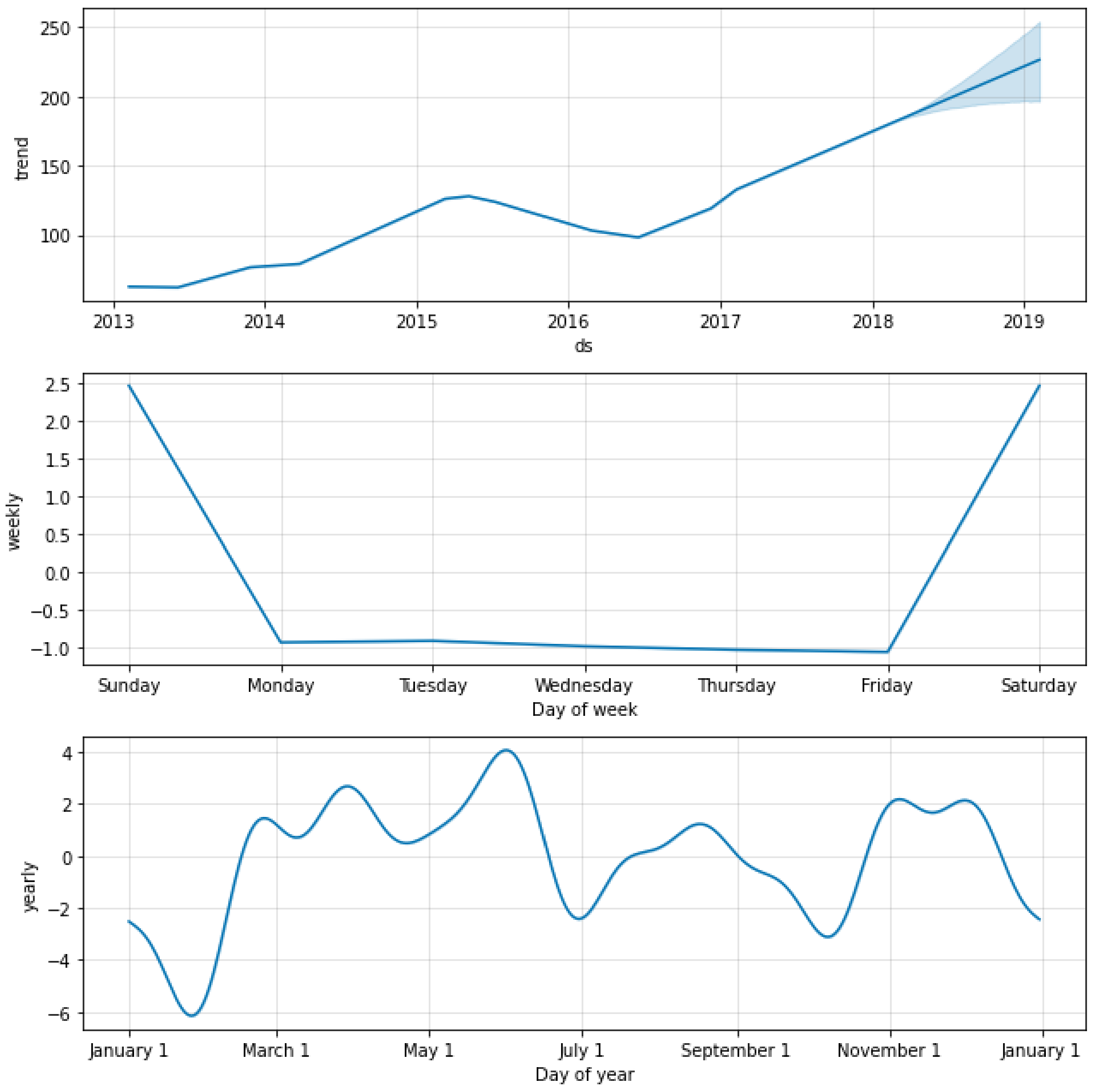
```
figure=m.plot(forecast,xlabel='Date',ylabel='Closing Price')
```



The model predicts that the Apple stock Price would increase from Mar 2018 to Mar 2019.

Plotting component of the Forecast

```
figure=m.plot_components(forecast)
```



1.Historical Trend Show that the Price of Apple stock has been increasing.Ivestors must have made good money on it

2.Weekly trend shows that the Stock price increase is highest on Tuesday then reduces as week proceeds.Please do note that Saturday and Sunday are off for the Stock Exchange.

3.The annual trend shows the seasoniality of the stock.It can be figured out the stock price peaks in month of May.