PANDAS PART 8

handling time series data.

DateTime indexing, Resampling, date range

Datasets used: AAPL Data.csv, AAPL Data1.csv

NOTE: First load the data as:

```
df = pd.read_csv("D:/aatish/datasets/AAPL Data.csv")
```

And test the 'Date' column type. It shows 'str' type.

```
type(df.Date[0])  # gives str
```

So, change it into Date and time by using parse_dates attribute. Also do indexing on 'Date' column. This has many advantages. We can retrieve data based on the date and time.

[3]:
```
# time series analysis on Apple company shares date
import pandas as pd
df = pd.read_csv("D:/aatish/datasets/AAPL Data.csv", parse_dates=["Date"],
                 index_col= "Date")
df
```

[3]:

Date	Open	High	Low	Close	Volume
2019-10-24 16:00:00	61.13	61.20	60.45	60.90	17916255
2019-10-25 16:00:00	60.79	61.68	60.72	61.65	18369296
2019-10-28 16:00:00	61.86	62.31	61.68	62.26	24143241
2019-10-29 16:00:00	62.24	62.44	60.64	60.82	35709867
2019-10-30 16:00:00	61.19	61.33	60.30	60.82	31130522
...	...	...	...	...	...
2021-10-18 16:00:00	143.45	146.84	143.16	146.55	85589175

[5]:
```
# retrieve only Jan 2020 data rows
df.loc["2020-01"]
```

[5]:

Date	Open	High	Low	Close	Volume
2020-01-13 16:00:00	77.91	79.27	77.79	79.24	30521722
2020-01-14 16:00:00	79.18	79.39	78.04	78.17	40653457
2020-01-15 16:00:00	77.96	78.88	77.39	77.83	30480882
2020-01-16 16:00:00	78.40	78.93	78.02	78.81	27207254
2020-01-17 16:00:00	79.07	79.69	78.75	79.68	34454117
2020-01-21 16:00:00	79.30	79.76	79.00	79.14	27710814

[6]:
```
# find average stock price (closing price) in Jan 2020
df.loc["2020-01"].Close.mean()
```

[6]:
```
85.02190476190475
```

[9]:
```
# Retrieve stock price on any date
df.loc["2020-01-05"]
```

[9]:

Date	Open	High	Low	Close	Volume
2020-01-05 16:00:00	71.56	74.75	71.46	72.27	60154175

[11]:
```
# Retrieve stock prices in given range of dates
df.loc["2020-01-05":"2020-01-10"]
```

```
<ipython-input-11-133c8a3c7233>:2: FutureWarning: Value based partial slic
g keys is deprecated and will raise a KeyError in a future Version.
  df.loc["2020-01-05":"2020-01-10"]
```

[11]:

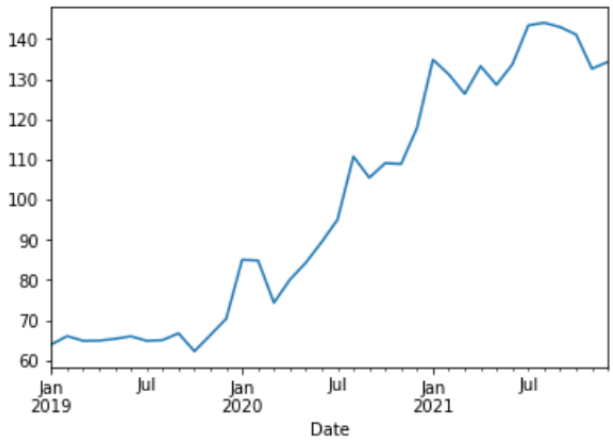
Date	Open	High	Low	Close	Volume
2020-01-05 16:00:00	71.56	74.75	71.46	72.27	60154175
2020-01-06 16:00:00	79.44	80.59	79.30	80.46	20254653
2020-01-07 16:00:00	91.28	91.84	90.98	91.03	27684309
2020-01-09 16:00:00	132.76	134.80	130.53	134.18	152470142
2020-01-10 16:00:00	117.64	117.72	115.83	116.79	116120440

[19]:
```
# Resampling the data - to know monthly average of stock price
df.Close.resample('M').mean()  # Frequencies -> 'M', 'D', 'W', 'Q', 'B', 'H'
```

[19]:
```
Date
2019-01-31     63.960000
2019-02-28     66.040000
2019-03-31     64.860000
2019-04-30     64.910000
2019-05-31     65.340000
2019-06-30     65.995000
2019-07-31     64.860000
```

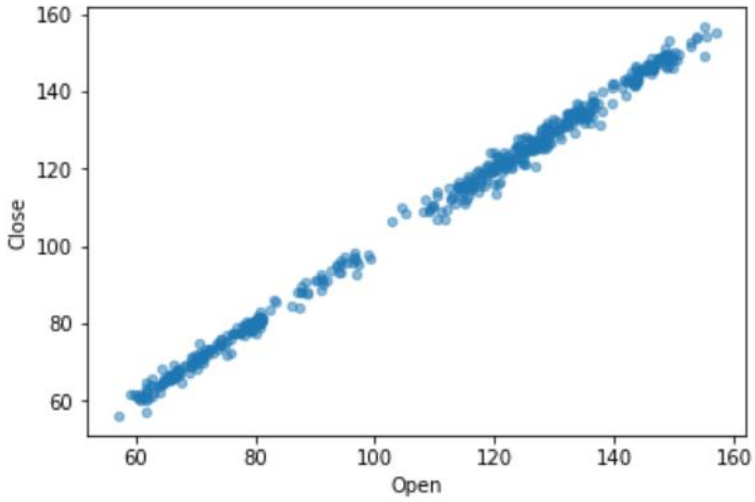
[29]:
```
# show the line plot
%matplotlib inline
df.Close.resample('M').mean().plot(kind='line')    # plot(kind="bar" /'hist' / 'area' )
```

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



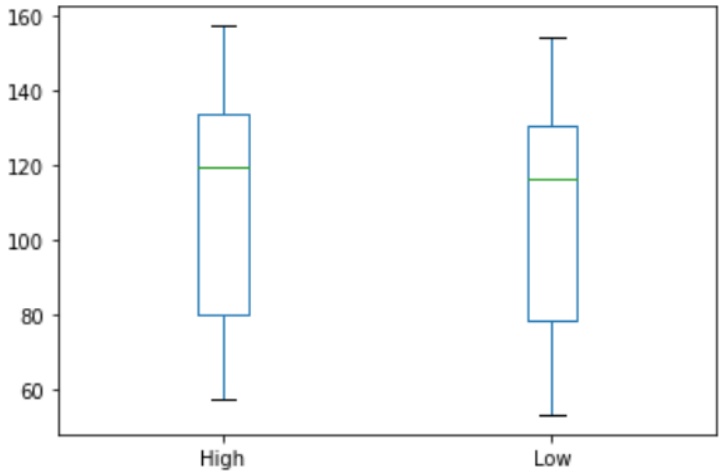
```
[35]: # show the scatter plot
      %matplotlib inline
      df.plot.scatter(x='Open', y='Close', alpha=0.5)
```

```
[35]: <AxesSubplot:xlabel='Open', ylabel='Close'>
```




```
[43]: # show the box plot
      %matplotlib inline
      df[['High', 'Low']].plot.box()
```

```
[43]: <AxesSubplot:>
```



How to add dates to a data frame

Count the no. of rows in the data frame and generate those many dates first. Then add them to the data frame.

[32]:
```
# working with range of dates
import pandas as pd
# display all rows at a time
pd.set_option('display.max_rows', None)
```

[33]:
```
# read data
df = pd.read_csv("D:/aatish/datasets/AAPL Data1.csv")
df
```

18	66.97	67.00	66.35	66.57	19069597
19	66.39	66.52	65.10	65.80	26609919
20	65.92	66.00	65.30	65.50	30348778
21	65.65	65.80	65.21	65.44	16331263
22	65.68	66.61	65.63	66.59	21029517
23	66.74	66.79	65.63	66.07	26334882

[34]:
```
# generate a range of dates - we want 100 dates
# freq='B' -> business days, 'D' -> all days
rng = pd.date_range(start="2020/1/20", end="2020/6/7", freq='B')
rng
```

[34]:
```
DatetimeIndex(['2020-01-20', '2020-01-21', '2020-01-22', '2020-01-23',
               '2020-01-24', '2020-01-27', '2020-01-28', '2020-01-29',
               '2020-01-30', '2020-01-31', '2020-02-03', '2020-02-04',
               '2020-02-05', '2020-02-06', '2020-02-07', '2020-02-10',
               '2020-02-11', '2020-02-12', '2020-02-13', '2020-02-14',
               '2020-02-17', '2020-02-18', '2020-02-19', '2020-02-20',
               '2020-02-21', '2020-02-24', '2020-02-25', '2020-02-26',
               '2020-02-27', '2020-02-28', '2020-03-02', '2020-03-03',
               '2020-03-04', '2020-03-05', '2020-03-06', '2020-03-09',
               '2020-03-10', '2020-03-11', '2020-03-12', '2020-03-13',
               '2020-03-16', '2020-03-17', '2020-03-18', '2020-03-19',
               '2020-03-20', '2020-03-23', '2020-03-24', '2020-03-25',
               '2020-03-26', '2020-03-27', '2020-03-30', '2020-03-31',
               '2020-04-01', '2020-04-02', '2020-04-03', '2020-04-06',
```

[35]:
```
len(rng)  # are there 100 dates?
```

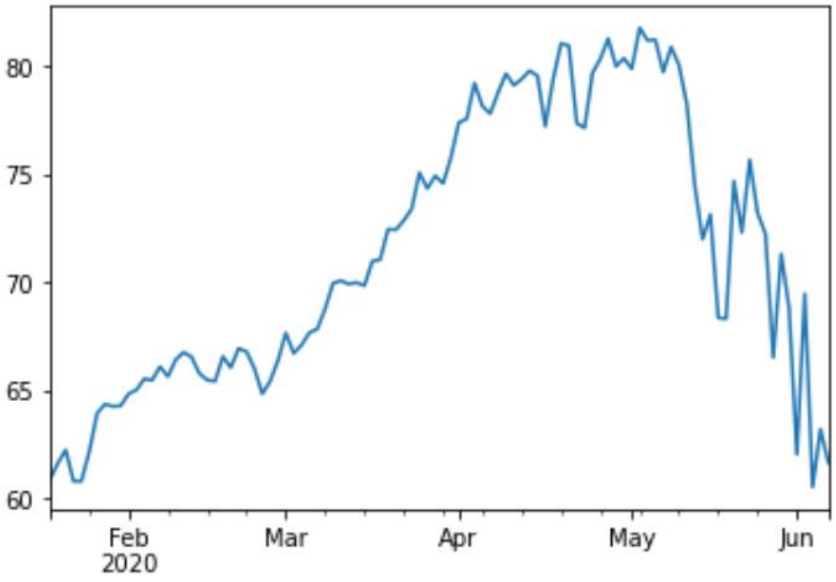
[35]:
```
100
```

[36]:
```
# set index to the range of dates and add into data frame
df.set_index(rng, inplace=True)
df
```

2020-01-22	61.86	62.31	61.68	62.26	24143241
2020-01-23	62.24	62.44	60.64	60.82	35709867
2020-01-24	61.19	61.33	60.30	60.82	31130522
2020-01-27	61.81	62.29	59.32	62.19	34790520
2020-01-28	62.39	63.98	62.29	63.96	37781334
2020-01-29	64.33	64.46	63.85	64.38	25817952
2020-01-30	64.26	64.55	64.08	64.28	19974427
2020-01-31	64.19	64.37	63.84	64.31	18966124

[37]:
```
# display plot for closing values
%matplotlib inline
df.Close.plot()
```

[37]:
```
<AxesSubplot:>
```



[38]:
```
# display average of closing value for first 10 days
df.head(10).Close.mean()
```

[38]:
```
62.556999999999995
```

[39]:
```
# display average of closing value for first 10 days - another way
df['2020-1-20' : '2020-1-31'].Close.mean()
```

[39]:
```
62.556999999999995
```

[40]:
```
# regenerate data frame using new frequency
# 'padding' of previous day values is done to Sat and Sundays.
df.asfreq('D', method='pad')
```

2020-02-26	64.58	64.88	64.07	64.86	29377268
2020-02-27	65.27	65.83	65.17	65.44	16810388
2020-02-28	65.95	66.47	65.68	66.40	18661343
2020-02-29	65.95	66.47	65.68	66.40	18661343
2020-03-01	65.95	66.47	65.68	66.40	18661343

42]:
```
# To show weekly values, use 'W' and for Hourly values, use 'H'
df.asfreq('H', method='pad')
```

2020-01-22 21:00:00	61.86	62.31	61.68	62.26	24143241
2020-01-22 22:00:00	61.86	62.31	61.68	62.26	24143241
2020-01-22 23:00:00	61.86	62.31	61.68	62.26	24143241
2020-01-23 00:00:00	62.24	62.44	60.64	60.82	35709867
2020-01-23 01:00:00	62.24	62.44	60.64	60.82	35709867
2020-01-23 02:00:00	62.24	62.44	60.64	60.82	35709867
2020-01-23 03:00:00	62.24	62.44	60.64	60.82	35709867
2020-01-23 04:00:00	62.24	62.44	60.64	60.82	35709867
2020-01-23 05:00:00	62.24	62.44	60.64	60.82	35709867
2020-01-23 06:00:00	62.24	62.44	60.64	60.82	35709867

[49]:
```
# When we know only starting date, then how to generate range of dates?
rng = pd.date_range(start="2020/1/20", periods=100, freq='B')
rng
```

[49]:
```
DatetimeIndex(['2020-01-20', '2020-01-21', '2020-01-22', '2020-01-23',
               '2020-01-24', '2020-01-27', '2020-01-28', '2020-01-29',
               '2020-01-30', '2020-01-31', '2020-02-03', '2020-02-04',
               '2020-02-05', '2020-02-06', '2020-02-07', '2020-02-10',
               '2020-02-11', '2020-02-12', '2020-02-13', '2020-02-14',
               '2020-02-17', '2020-02-18', '2020-02-19', '2020-02-20',
               '2020-02-21', '2020-02-24', '2020-02-25', '2020-02-26',
               '2020-02-27', '2020-02-28', '2020-03-02', '2020-03-03',
               '2020-03-04', '2020-03-05', '2020-03-06', '2020-03-09',
               '2020-03-10', '2020-03-11', '2020-03-12', '2020-03-13',
               '2020-03-16', '2020-03-17', '2020-03-18', '2020-03-19',
               '2020-03-20', '2020-03-23', '2020-03-24', '2020-03-25',
```