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

In [2]:
```
df=pd.read_csv(r'C:\Users\234567890-\Desktop\covid_R\weather_data.csv')
```

In [3]:
```
df
```

Out[3]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [3]:
```
df
```

Out[3]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [4]:
```
temp=pd.read_csv(r'C:\Users\234567890-\Desktop\covid_R\weather_data.csv')
```

In [5]:
```
type(temp['day'][0])
```

Out[5]:
```
str
```

In [6]:
```
import pandas as pd
df=pd.read_csv(r'C:\Users\234567890-\Desktop\covid_R\weather_data.csv',parse_dates=['day'
df
```

Out[6]:

	day	temperature	windspeed	event
0	2017-01-01	32.0	6.0	Rain
1	2017-01-04	NaN	9.0	Sunny
2	2017-01-05	28.0	NaN	Snow
3	2017-01-06	NaN	7.0	NaN

Loading [MathJax]/extensions/Safe.js

	day	temperature	windspeed	event
4	2017-01-07	32.0	NaN	Rain
5	2017-01-08	NaN	NaN	Sunny
6	2017-01-09	NaN	NaN	NaN
7	2017-01-10	34.0	8.0	Cloudy
8	2017-01-11	40.0	12.0	Sunny

In [7]:
```
type(df['day'][0])
```

Out[7]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

In [8]:
```
#covert column to index
```

In [9]:
```
df.set_index('day',inplace=True) #changes will be made on original data or dataframr
```

In [10]:
```
df
```

Out[10]:

	temperature	windspeed	event
day			
2017-01-01	32.0	6.0	Rain
2017-01-04	NaN	9.0	Sunny
2017-01-05	28.0	NaN	Snow
2017-01-06	NaN	7.0	NaN
2017-01-07	32.0	NaN	Rain
2017-01-08	NaN	NaN	Sunny
2017-01-09	NaN	NaN	NaN
2017-01-10	34.0	8.0	Cloudy
2017-01-11	40.0	12.0	Sunny

In [18]:
```
df[['temperature','windspeed']]
```

Out[18]:

	temperature	windspeed
day		
2017-01-01	32.0	6.0
2017-01-04	NaN	9.0
2017-01-05	28.0	NaN
2017-01-06	NaN	7.0
2017-01-07	32.0	NaN
2017-01-08	NaN	NaN
2017-01-09	NaN	NaN
2017-01-10	34.0	8.0
2017-01-11	40.0	12.0

In [12]:
```
df.loc['2017-01-08']# what was temperature on 8th of january
```

```
Out[12]: temperature      NaN
         windspeed        NaN
         event          Sunny
         Name: 2017-01-08 00:00:00, dtype: object
```

```
In [13]: df.loc['2017-01-09']
```

```
Out[13]: temperature    NaN
         windspeed      NaN
         event          NaN
         Name: 2017-01-09 00:00:00, dtype: object
```

```
In [14]: df.loc['2017-01-10']['temperature'] #extract only temperature
```

```
Out[14]: 34.0
```

```
In [15]: df['temperature'].loc['2017-01-10'] # we can write this formate also
```

```
Out[15]: 34.0
```

```
In [17]: df.loc['2017-01-10'][0]#this format also
```

```
Out[17]: 34.0
```

fillna

fillna **Fill all NaN with one specific value**

```
In [21]: new_df = df.fillna(0)
         new_df
```

Out[21]:

	temperature	windspeed	event
day			
2017-01-01	32.0	6.0	Rain
2017-01-04	0.0	9.0	Sunny
2017-01-05	28.0	0.0	Snow
2017-01-06	0.0	7.0	0
2017-01-07	32.0	0.0	Rain
2017-01-08	0.0	0.0	Sunny
2017-01-09	0.0	0.0	0
2017-01-10	34.0	8.0	Cloudy
2017-01-11	40.0	12.0	Sunny

```
In [22]: new_df = df.fillna({                        #**Fill na using column names and dict**
                 'temperature':0,
                 'windspeed': 0.0,
                 'event': 'Event not Recorded'
             })
         new_df
```

Out[22]:

	temperature	windspeed	event
day			
2017-01-01	32.0	6.0	Rain

Loading [MathJax]/extensions/Safe.js

day	temperature	windspeed	event
2017-01-04	0.0	9.0	Sunny
2017-01-05	28.0	0.0	Snow
2017-01-06	0.0	7.0	Event not Recorded
2017-01-07	32.0	0.0	Rain
2017-01-08	0.0	0.0	Sunny
2017-01-09	0.0	0.0	Event not Recorded
2017-01-10	34.0	8.0	Cloudy
2017-01-11	40.0	12.0	Sunny

```
In [23]: df['temperature'].fillna(0).mean() # find avg r mean
```

```
Out[23]: 18.444444444444443
```

```
In [24]: df['temperature'].fillna(0).sum()
```

```
Out[24]: 166.0
```

```
In [25]: new_df = df.fillna(method="ffill")  #** forward fillng Use method to determine how to fil
         new_df
```

Out[25]:

day	temperature	windspeed	event
2017-01-01	32.0	6.0	Rain
2017-01-04	32.0	9.0	Sunny
2017-01-05	28.0	9.0	Snow
2017-01-06	28.0	7.0	Snow
2017-01-07	32.0	7.0	Rain
2017-01-08	32.0	7.0	Sunny
2017-01-09	32.0	7.0	Sunny
2017-01-10	34.0	8.0	Cloudy
2017-01-11	40.0	12.0	Sunny

```
In [26]: new_df = df.fillna(method="bfill")  #back word filling
         new_df
```

Out[26]:

day	temperature	windspeed	event
2017-01-01	32.0	6.0	Rain
2017-01-04	28.0	9.0	Sunny
2017-01-05	28.0	7.0	Snow
2017-01-06	32.0	7.0	Rain
2017-01-07	32.0	8.0	Rain
2017-01-08	34.0	8.0	Sunny

day	temperature	windspeed	event
2017-01-09	34.0	8.0	Cloudy
2017-01-10	34.0	8.0	Cloudy
2017-01-11	40.0	12.0	Sunny

In [27]:
```
#**Use of axis**

new_df = df.fillna(method="ffill", axis="columns") # axis is either "index" or "columns"
new_df
```

Out[27]:

day	temperature	windspeed	event
2017-01-01	32	6	Rain
2017-01-04	NaN	9	Sunny
2017-01-05	28	28	Snow
2017-01-06	NaN	7	7
2017-01-07	32	32	Rain
2017-01-08	NaN	NaN	Sunny
2017-01-09	NaN	NaN	NaN
2017-01-10	34	8	Cloudy
2017-01-11	40	12	Sunny

In [28]:
```
#**limit parameter**

new_df = df.fillna(method="ffill",limit=2)
new_df
```

Out[28]:

day	temperature	windspeed	event
2017-01-01	32.0	6.0	Rain
2017-01-04	32.0	9.0	Sunny
2017-01-05	28.0	9.0	Snow
2017-01-06	28.0	7.0	Snow
2017-01-07	32.0	7.0	Rain
2017-01-08	32.0	7.0	Sunny
2017-01-09	32.0	NaN	Sunny
2017-01-10	34.0	8.0	Cloudy
2017-01-11	40.0	12.0	Sunny

In [29]:
```
new_df = df.fillna(method="bfill",limit=1)
new_df
```

Out[29]:

day	temperature	windspeed	event

Loading [MathJax]/extensions/Safe.js

day	temperature	windspeed	event
2017-01-01	32.0	6.0	Rain
2017-01-04	28.0	9.0	Sunny
2017-01-05	28.0	7.0	Snow
2017-01-06	32.0	7.0	Rain
2017-01-07	32.0	NaN	Rain
2017-01-08	NaN	NaN	Sunny
2017-01-09	34.0	8.0	Cloudy
2017-01-10	34.0	8.0	Cloudy
2017-01-11	40.0	12.0	Sunny

interpolate

```
In [11]: new_df = df.interpolate() # fill the missing values only in numeric value not in string
         new_df
```

Out[11]:

day	temperature	windspeed	event
2017-01-01	32.000000	6.00	Rain
2017-01-04	30.000000	9.00	Sunny
2017-01-05	28.000000	8.00	Snow
2017-01-06	30.000000	7.00	NaN
2017-01-07	32.000000	7.25	Rain
2017-01-08	32.666667	7.50	Sunny
2017-01-09	33.333333	7.75	NaN
2017-01-10	34.000000	8.00	Cloudy
2017-01-11	40.000000	12.00	Sunny

```
In [12]: new_df = df.interpolate(method="time") # time method will work om giving date not any ind
         new_df
```

Out[12]:

day	temperature	windspeed	event
2017-01-01	32.000000	6.00	Rain
2017-01-04	29.000000	9.00	Sunny
2017-01-05	28.000000	8.00	Snow
2017-01-06	30.000000	7.00	NaN
2017-01-07	32.000000	7.25	Rain
2017-01-08	32.666667	7.50	Sunny
2017-01-09	33.333333	7.75	NaN
2017-01-10	34.000000	8.00	Cloudy

	temperature	windspeed	event
day			
2017-01-11	40.000000	12.00	Sunny

Notice that in above temperature on 2017-01-04 is 29 instead of 30 (in plain linear interpolate)

There are many other methods for interpolation such as quadratic, piecewise_polynomial, cubic etc. Just google "dataframe interpolate" to see complete documentation

dropna

In [13]:
```
new_df = df.dropna()  # here we drop nan values
new_df
```

Out[13]:

	temperature	windspeed	event
day			
2017-01-01	32.0	6.0	Rain
2017-01-10	34.0	8.0	Cloudy
2017-01-11	40.0	12.0	Sunny

In [14]:
```
new_df = df.dropna(how='all') # drop nan values in row contains all nan values
new_df
```

Out[14]:

	temperature	windspeed	event
day			
2017-01-01	32.0	6.0	Rain
2017-01-04	NaN	9.0	Sunny
2017-01-05	28.0	NaN	Snow
2017-01-06	NaN	7.0	NaN
2017-01-07	32.0	NaN	Rain
2017-01-08	NaN	NaN	Sunny
2017-01-10	34.0	8.0	Cloudy
2017-01-11	40.0	12.0	Sunny

In [15]:
```
new_df = df.dropna(thresh=2)  # it used to drop which row contain 2 nan it will delete
new_df
```

Out[15]:

	temperature	windspeed	event
day			
2017-01-01	32.0	6.0	Rain
2017-01-04	NaN	9.0	Sunny
2017-01-05	28.0	NaN	Snow
2017-01-07	32.0	NaN	Rain
2017-01-10	34.0	8.0	Cloudy
2017-01-11	40.0	12.0	Sunny

Loading [MathJax]/extensions/Safe.js

Inserting Missing Dates

In [17]:
```
dt=pd.date_range("01-01-2017","01-11-2017")
```

In [19]:
```
dt
```

Out[19]:
```
DatetimeIndex(['2017-01-01', '2017-01-02', '2017-01-03', '2017-01-04',
               '2017-01-05', '2017-01-06', '2017-01-07', '2017-01-08',
               '2017-01-09', '2017-01-10', '2017-01-11'],
              dtype='datetime64[ns]', freq='D')
```

In [20]:
```
dt = pd.date_range("01-01-2017","01-11-2017")
idx = pd.DatetimeIndex(dt)
df.reindex(idx)
```

Out[20]:
```
(DatetimeIndex(['2017-01-01', '2017-01-02', '2017-01-03', '2017-01-04',
                '2017-01-05', '2017-01-06', '2017-01-07', '2017-01-08',
                '2017-01-09', '2017-01-10', '2017-01-11'],
               dtype='datetime64[ns]', freq='D'),
 None)
```

Replacing list with another list

In [21]:
```
df = pd.DataFrame({
    'score': ['exceptional','average', 'good', 'poor', 'average', 'exceptional'],
    'student': ['abhi', 'maya', 'parthiv', 'tom', 'julian', 'erica']
})
df
```

Out[21]:

	score	student
0	exceptional	abhi
1	average	maya
2	good	parthiv
3	poor	tom
4	average	julian
5	exceptional	erica

In [22]:
```
li=['poor', 'average', 'good', 'exceptional','erica']
li2=["C","B","A","A+",'ERICA']
df.replace(li,li2)
```

Out[22]:

	score	student
0	A+	abhi
1	B	maya
2	A	parthiv
3	C	tom
4	B	julian
5	A+	ERICA

In [23]:
```
df.replace("abhi","Ravi")
```

Out[23]:

	score	student

Loading [MathJax]/extensions/Safe.js

	score	student
0	exceptional	Ravi
1	average	maya
2	good	parthiv
3	poor	tom
4	average	julian
5	exceptional	erica

```
In [ ]:
```