Pandas

Monday, September 24, 2018 12:31 PM

```
In [66]: df['Temperature'].max()
Out[66]: 50

In [67]: df['EST'][df['Events']=='Rain']
Out[67]: 8     1/9/2016
         9    1/18/2016
         15   1/16/2016
         26   1/27/2016
         Name: EST, dtype: object

In [69]: df.fillna(0, inplace=True)
         df['WindSpeedMPH'].mean()
Out[69]: 6.225806451612903
```

###convert dict to dataframe

```
In [4]: import pandas as pd
        weather_data = {
            'day': ['1/1/2017','1/2/2017','1/3/2017','1/4/2017','1/5/2017','1/6/2017'],
            'temperature': [32,35,28,24,32,31],
            'windspeed': [6,7,2,7,4,2],
            'event': ['Rain', 'Sunny', 'Snow','Snow','Rain', 'Sunny']
        }
        df = pd.DataFrame(weather_data)
        df
```

Out[4]:

	day	event	temperature	windspeed
0	1/1/2017	Rain	32	6
1	1/2/2017	Sunny	35	7
2	1/3/2017	Snow	28	2
3	1/4/2017	Snow	24	7
4	1/5/2017	Rain	32	4
5	1/6/2017	Sunny	31	2

Df.columns ==> gives column names

Selective columns

```
In [23]: df[['event','day','temperature']]
```

Out[23]:

	event	day	temperature
0	Rain	1/1/2017	32
1	Sunny	1/2/2017	35
2	Snow	1/3/2017	28

Drop rows if entry is Na for some columns

```
In [20]: new_df = df.dropna()
         new_df
```

Out[20]:

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

Dropna(how=ALL) only when all columns of the row are misisng

Replace functions:-->
Df.replace("9999", np.NaN)
Df.replace({dictionary details},np.NaN)

REgexreplace

```
In [12]: new_df = df.replace('[A-Za-z]','',regex=True)
         new_df
```

Out[12]:

	day	temperature	windspeed	event
0	1/1/2017	32	6	
1	1/2/2017	-99999	7	
2	1/3/2017	28	-99999	
3	1/4/2017	-99999	7	
4	1/5/2017	32	-99999	
5	1/6/2017	31	2	
6	1/6/2017	34	6	

```
In [12]: new_df = df.replace({
             'temperature': '[A-Za-z]',
             'windspeed': '[A-Za-z]'
         },'',regex=True)
         new_df
```

Out[12]:

	day	temperature	windspeed	event
0	1/1/2017	32	6	
1	1/2/2017	-99999	7	

Selective querying

```
In [32]: df[df.temperature==df.temperature.max()]
```

Out[32]:

	day	event	temperature	windspeed
1	1/2/2017	Sunny	35	7

```
In [35]: df[['day','temperature']][df.temperature==df['temperature'].max()]
```

Out[35]:

	day	temperature
1	1/2/2017	35

Set_index ==> sets as index

U can use loc[] based on the index value

HANDLE MISSING DATA

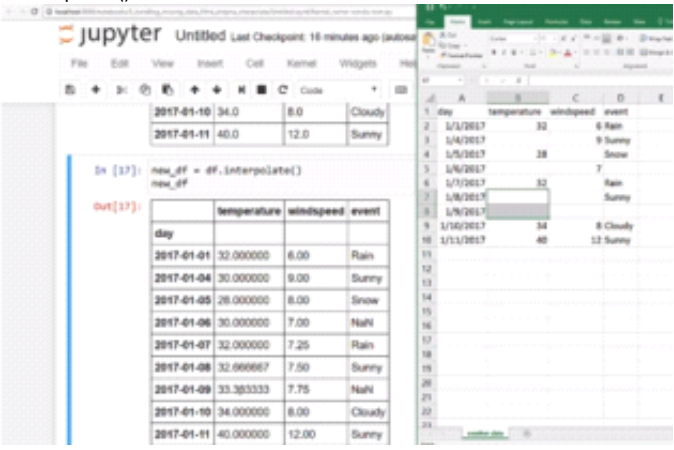
Df.fillna(<value to be replaced>)

Like fillna(0)

```
In [ ]: new_df = df.fillna({
            'temperature': 0,
            'windspeed': 0,
            'event': 'no event'
        })
```

Method =ffill==>carryforward prev value
=bfill ==>next value

Df.interpolate()



Group by:-->
Df_group = Df.groupby("city")

```
In [3]: g = df.groupby('city')
        g
Out[3]: <pandas.core.groupby.DataFrameGroupBy object at 0x000002A411979E8>

In [4]: for city, city_df in g:
            print(city)
            print(city_df)

        mumbai
           day    city  temperature  windspeed  event
        4 1/1/2017 mumbai          90          5  Sunny
```

r

```
In [5]: g.get_group('mumbai')
```

Out[5]:

	day	city	temperature	windspeed	event
4	1/1/2017	mumbai	90	5	Sunny
5	1/2/2017	mumbai	85	12	Fog

```
new_df
```

Out[12]:

	day	temperature	windspeed	event
0	1/1/2017	32	6	
1	1/2/2017	-99999	7	
2	1/3/2017	28	-99999	
3	1/4/2017	-99999	7	
4	1/5/2017	32	-99999	
5	1/6/2017	31	2	
6	1/6/2017	34	5	

```
In [5]: g.get_group('mumbai')
```

Out[5]:

	day	city	temperature	windspeed	event
4	1/1/2017	mumbai	90	5	Sunny
5	1/2/2017	mumbai	85	12	Fog
6	1/3/2017	mumbai	87	15	Fog
7	1/4/2017	mumbai	92	5	Rain

```
In [6]: g.max()
```

Out[6]:

	day	temperature	windspeed	event
city				
mumbai	1/4/2017	92	15	Sunny
new york	1/4/2017	36	12	Sunny
paris	1/4/2017	54	20	Sunny

CONCATINATION:-
Pd.concat(DF1,DF2)

Axis =0==>append asrows
Axis =1 ==>append as cols

Pivot table

Index = <>,cols = <> ,Aggfunc ="count"

MERGE
Pd.merge(df1,df2,on =<column name>)

Pd.merge(df1,df2, how ="outer")

Indicator =True tells whether the value was present in both

Pandas melt function:->

Pd.melt(df1,id_vars=[list of columns to keep

```
In [7]: g.mean()
```

Out[7]:

	temperature	windspeed
city		
mumbai	88.50	9.25
new york	32.25	8.00
paris	47.75	12.75

Crosstab

```
In [2]: pd.crosstab(df.Nationality,df.Handedness)
```

Out[2]:

Handedness	Left	Right
Nationality		
Bangadesh	2	0
China	2	1
India	2	1
USA	1	3

```
In [4]: pd.crosstab(df.Sex,df.Handedness,margins=True)
```

Out[4]:

Handedness	Left	Right	All
Sex			
Female	2	3	5
Male	5	2	7
All	7	5	12

```
In [5]: pd.crosstab(df.Sex,[df.Handedness],margins=True)
```

Out[5]:

Handedness	Left				Right			All
Nationality	Bangadesh	China	India	USA	China	India	USA	
Sex								
Female	1	1	0	0	1	0	2	5
Male	1	1	2	1	0	1	1	7
All	2	2	2	1	1	1	3	12

Normalize=True gives the pecentage distribution