PANDAS PART 4

select, where, groupby, aggregate functions

Selecting rows or columns in pandas

NOTE: Use the 'titanic.csv' dataset.

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

[3]:
```
df = pd.read_csv("D:/aatish/datasets/titanic.csv")
# to select the first 5 rows
df.head()  # df.head(10) for 10 rows
```

[4]:
```
# to select last 5 rows
df.tail()  # df.tail(10) for last 10 rows
```

[5]:
```
# to select one column only
df['Pclass']  # this is same as df.Pclass
```

[7]:
```
# to see all column names
df.columns
```

[7]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

[8]:
```
# to select multiple columns
df[['Name', 'Sex']]
```

13]:
```
# to select only one row using iloc[] or loc[] functions
df.iloc[5]  # stars counting from 0 onwards - same as loc[5]
```

16]:
```
# select a range of rows.
# we can use head(), tail(), iloc[] and loc[]
df.iloc[400:403]  # also use loc[400:403]
```

19]:
```
# select a cell
df.iloc[2, 3]  # 2nd row and 3rd col value - cannot use loc[] here
```

19]:
```
'Heikkinen, Miss. Laina'
```

22]:
```
# select 5th to 8th rows in 3rd column
df.iloc[5:9, 3]
```

22]:
```
5                                  Moran, Mr. James
6                           McCarthy, Mr. Timothy J
7                    Palsson, Master. Gosta Leonard
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
Name: Name, dtype: object
```

Selection based on condition

[27]:
```
# display rows with Age>45
df[df['Age']>45]    # same as df[df.Age>45]
```

[34]:
```
# display rows with Age>45 and Sex='male'
df[(df['Age']>45) & (df['Sex']== 'male')]
```

[36]:
```
# display PassengerId and Name where Age>45 and Sex='male'
df[['PassengerId', 'Name']][(df['Age']>45) & (df['Sex']== 'male')]
```

Data Frame with where()

Syn: pd.DataFrame.where(cond=condition_to_check, other="Value To Fill")

```
import pandas as pd
```

```
import numpy as np
```

```
# create data frame with 4 rows and 3 cols
df = pd.DataFrame(data=np.random.randint(0, 100, (4,3)),
                  columns= ("Test1", "Test2", "Test3"),
                  index= ['Vinay', 'Anil', 'Gopal', 'Rao'])

df
```

	Test1	Test2	Test3
Vinay	15	32	75
Anil	45	36	87
Gopal	66	31	13
Rao	73	68	96

```
df<80  # less than 80 is True. Others will be False
```

	Test1	Test2	Test3
Vinay	True	True	True
Anil	True	True	False
Gopal	True	True	True
Rao	True	True	False

```
# if df>=80, then fill with 'A+'
df.where(df<80, 'A+')
```

	Test1	Test2	Test3
Vinay	15	32	75
Anil	45	36	A+
Gopal	66	31	13
Rao	73	68	A+

```
# check the 'Test3' col values
df['Test3']<80
```

```
Vinay     True
Anil     False
Gopal     True
Rao      False
Name: Test3, dtype: bool
```

```
# apply where() to df based on 'Test3' column
# when 'Test3' becomes 'Excellent', the entire row gets that value
df.where(df['Test3']<80, 'Excellent')
```

	Test1	Test2	Test3
Vinay	15	32	75
Anil	Excellent	Excellent	Excellent
Gopal	66	31	13
Rao	Excellent	Excellent	Excellent

```
# where() with lambda. If x>=80, replace with 'Good'
df.where(lambda x: x<80, 'Good')
```

	Test1	Test2	Test3
Vinay	15	32	75
Anil	45	36	Good
Gopal	66	31	13
Rao	73	68	Good

```
# create another data frame with 4 rows and 3 cols
df2 = pd.DataFrame(data=np.random.randint(0, 100, (4,3)),
                   columns= ("Test1", "Test2", "Test3"),
                   index= ['Vinay', 'Anil', 'Gopal', 'Rao'])

df2
```

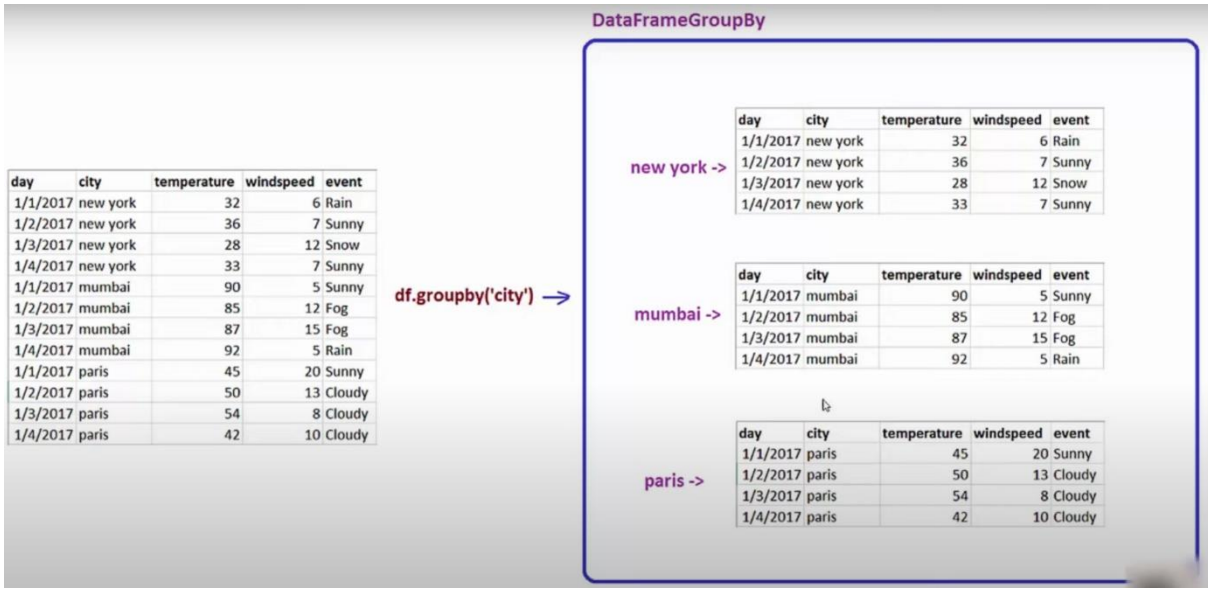
	Test1	Test2	Test3
Vinay	79	27	87
Anil	32	25	84
Gopal	15	41	50
Rao	40	15	59

```
# replace the values in df with those of df2 when the condition is
df.where(df<80, df2)                                          is False
```

	Test1	Test2	Test3
Vinay	15	32	75
Anil	45	36	84
Gopal	66	31	13
Rao	73	68	59

groupby() method

is useful to split the data into groups and then apply any operation on the groups. When groupby() is applied on a Data Frame, it creates GroupByDataFrame object splitting the data into groups. See the Figure:



day	city	temperature	windspeed	event
1/1/2017	new york	32	6	Rain
1/2/2017	new york	36	7	Sunny
1/3/2017	new york	28	12	Snow
1/4/2017	new york	33	7	Sunny
1/1/2017	mumbai	90	5	Sunny
1/2/2017	mumbai	85	12	Fog
1/3/2017	mumbai	87	15	Fog
1/4/2017	mumbai	92	5	Rain
1/1/2017	paris	45	20	Sunny
1/2/2017	paris	50	13	Cloudy
1/3/2017	paris	54	8	Cloudy
1/4/2017	paris	42	10	Cloudy

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

[2]:
```
df = pd.read_csv("D:/aatish/datasets/cities_weather.csv")
df
```

[2]:

	day	city	temperature	windspeed	event
0	01-01-2017	new york	32	6	Rain
1	01-02-2017	new york	36	7	Sunny
2	01-03-2017	new york	28	12	Snow
3	01-04-2017	new york	33	7	Sunny
4	01-01-2017	mumbai	90	5	Sunny
5	01-02-2017	mumbai	85	12	Fog
6	01-03-2017	mumbai	87	15	Fog
7	01-04-2017	mumbai	92	5	Rain
8	01-01-2017	paris	45	20	Sunny
9	01-02-2017	paris	50	13	Cloudy
10	01-03-2017	paris	54	8	Cloudy
11	01-04-2017	paris	42	10	Cloudy

[3]:
```
# separate data into groups based on 'city'
df1 = df.groupby('city')
# The above is same as: SELECT * FROM CITY_DATA GROUPY BY CITY
df1  # DataFrameGroupBy object
```

[3]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000210381029A0>
```

[4]:
```
# use a for loop to retrieve city and city data frame
for city, city_df in df1:
    print(city)
    print(city_df)
```

```
mumbai
          day    city  temperature  windspeed  event
4  01-01-2017  mumbai           90          5  Sunny
5  01-02-2017  mumbai           85         12    Fog
6  01-03-2017  mumbai           87         15    Fog
7  01-04-2017  mumbai           92          5   Rain
new york
          day      city  temperature  windspeed  event
0  01-01-2017  new york           32          6   Rain
1  01-02-2017  new york           36          7  Sunny
2  01-03-2017  new york           28         12   Snow
3  01-04-2017  new york           33          7  Sunny
paris
           day   city  temperature  windspeed   event
8   01-01-2017  paris           45         20   Sunny
9   01-02-2017  paris           50         13  Cloudy
10  01-03-2017  paris           54          8  Cloudy
11  01-04-2017  paris           42         10  Cloudy
```

```
[5]: # get only mumbai city data frame
     df1.get_group('mumbai')
```

[5]:

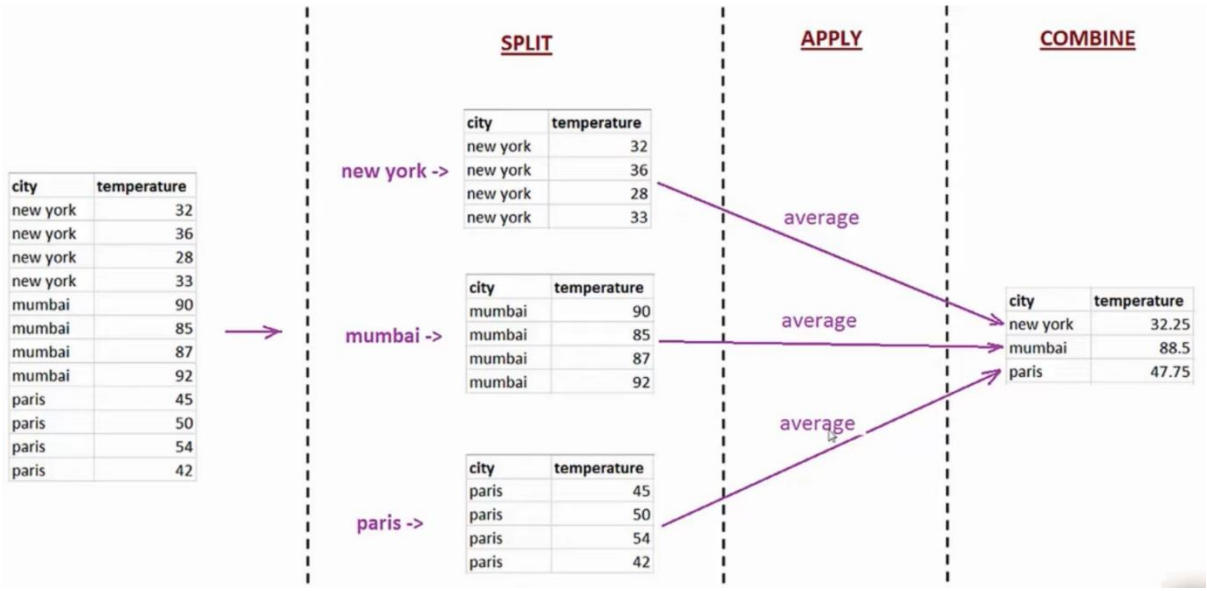
	day	city	temperature	windspeed	event
4	01-01-2017	mumbai	90	5	Sunny
5	01-02-2017	mumbai	85	12	Fog
6	01-03-2017	mumbai	87	15	Fog
7	01-04-2017	mumbai	92	5	Rain

```
[6]: # we can apply any methods like max(), min(), mean(), describe()
     # find city-wise max temperature and max windspeed
     df1.max()
```

[6]:

city	day	temperature	windspeed	event
mumbai	01-04-2017	92	15	Sunny
new york	01-04-2017	36	12	Sunny
paris	01-04-2017	54	20	Sunny

This is the reason, groupby() is called split-apply-combine. First it splits the data into groups, then we can apply any method on it. Finally the results are combined and displayed as shown above.



Aggregate functions in groupby()

agg() or aggregate() can be used on a data frame to apply aggregate function.

[14]:
```
# agg() is useful to apply aggregate function
df1.agg('max')
```

[14]:

city	day	temperature	windspeed	event
mumbai	01-04-2017	92	15	Sunny
new york	01-04-2017	36	12	Sunny
paris	01-04-2017	54	20	Sunny

[15]:
```
df1.agg(['max', 'min', 'mean'])
```

[15]:

	temperature			windspeed		
	max	min	mean	max	min	mean
city						
mumbai	92	85	88.50	15	5	9.25
new york	36	28	32.25	12	6	8.00
paris	54	42	47.75	20	8	12.75

Aggregate functions on Data Frames

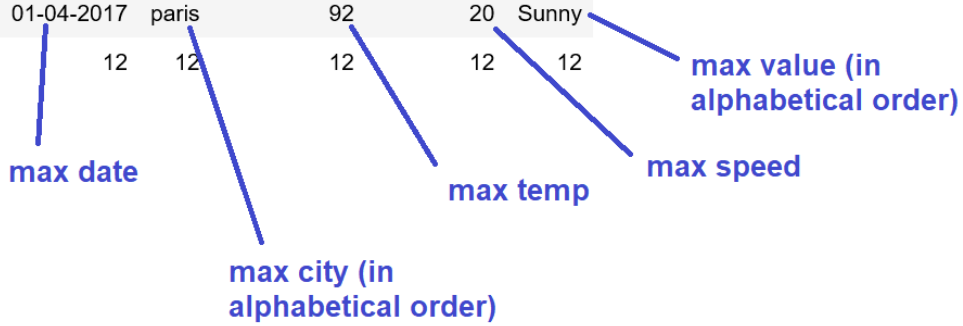
min() , max(), mean(), median(), sum(), prod(), mode(), std(), count() etc. are called aggregate functions since they work on group.

Example

[32]:
```
# find max and no. of rows in the data frame
df.agg(['max', 'count'])
```

[32]:

	day	city	temperature	windspeed	event
max	01-04-2017	paris	92	20	Sunny
count	12	12	12	12	12



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

[2]:
```
df = pd.read_csv("D:/aatish/datasets/cities_weather.csv")
df
```

[2]:

	day	city	temperature	windspeed	event
0	01-01-2017	new york	32	6	Rain
1	01-02-2017	new york	36	7	Sunny
2	01-03-2017	new york	28	12	Snow
3	01-04-2017	new york	33	7	Sunny
4	01-01-2017	mumbai	90	5	Sunny
5	01-02-2017	mumbai	85	12	Fog
6	01-03-2017	mumbai	87	15	Fog
7	01-04-2017	mumbai	92	5	Rain
8	01-01-2017	paris	45	20	Sunny
9	01-02-2017	paris	50	13	Cloudy
10	01-03-2017	paris	54	8	Cloudy
11	01-04-2017	paris	42	10	Cloudy

[17]:
```
# find min and max of temperature
df['temperature'].agg(['min', 'max'])
```

[17]:
```
min    28
max    92
Name: temperature, dtype: int64
```

[18]:
```
# find min and max of temperature and windspeed cols
df[['temperature', 'windspeed']].agg(['min', 'max'])
```

[18]:

	temperature	windspeed
min	28	5
max	92	20

[22]:
```
# find min and max of temperature and
# mean and median on windspeed
df.agg({'temperature': ['min', 'max'],
        'windspeed': ['mean', 'median']})
```

[22]:

	temperature	windspeed
min	28.0	NaN
max	92.0	NaN
mean	NaN	10.0
median	NaN	9.0