Pandas Group By

In this tutorial we are going to look at weather data from various cities and see how group by can be used to run some analytics.

```
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')

df = pd.read_csv("/content/drive/My Drive/pandas/pandas/7_group_by/weather_by_cities.cs

df
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491h

Enter your authorization code:
··········
Mounted at /content/drive
```

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

For this dataset, get following answers,

1. What was the maximum temperature in each of these 3 cities?
2. What was the average windspeed in each of these 3 cities?

```
g = df.groupby("city")
g
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f5e24d72390>
```

```
g.first()
```

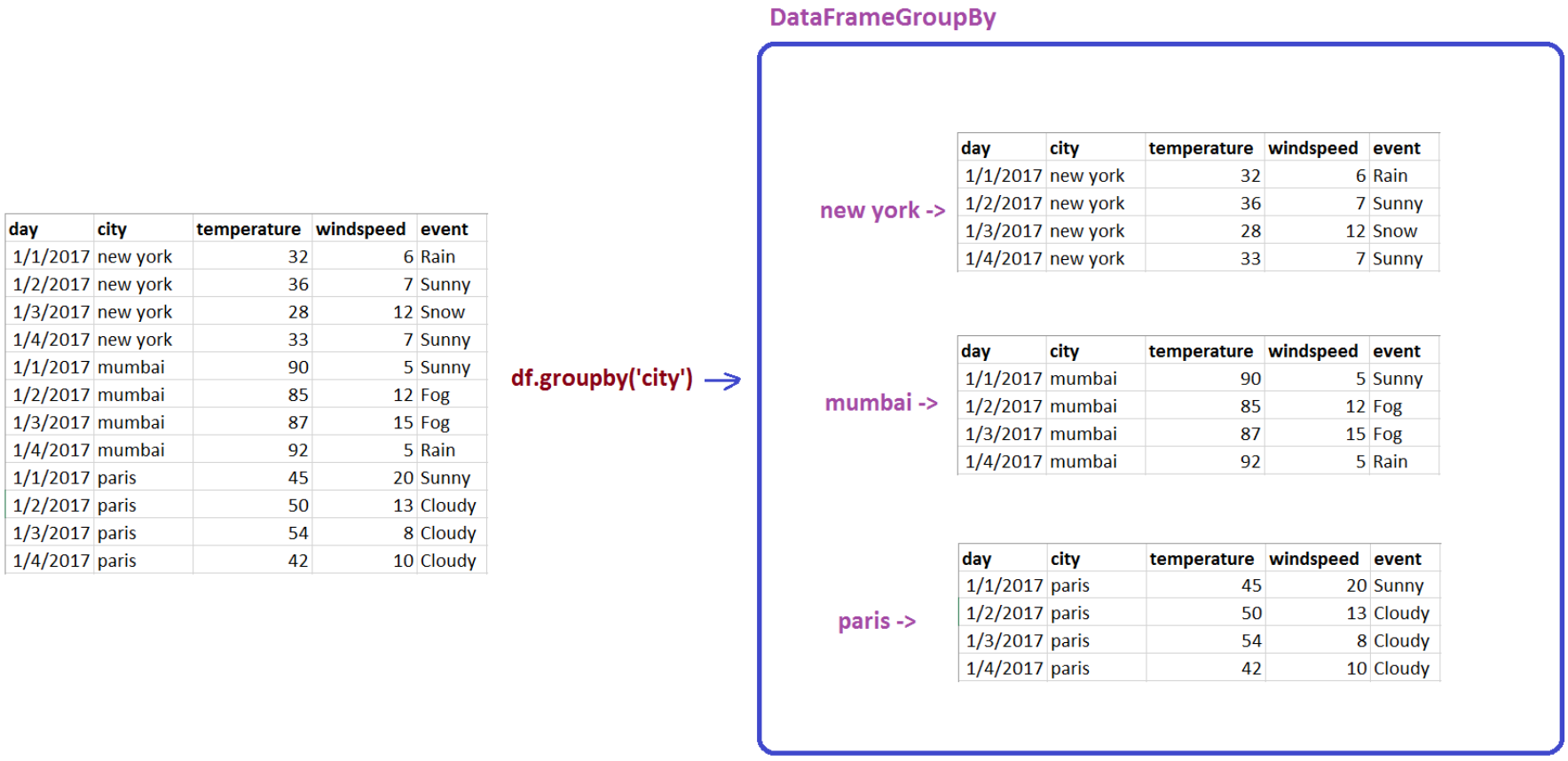
city	day	temperature	windspeed	event
mumbai	1/1/2017	90	5	Sunny
new york	1/1/2017	32	6	Rain
paris	1/1/2017	45	20	Sunny

```
g.get_group('mumbai') # return a dataframe
```

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

DataFrameGroupBy object looks something like below,

7	1/4/2017	92	5	Rain



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

```
for city, data in g:
    print("city:",city)
    print("\n")
```

```
print("data:",data)
```

```
city: mumbai


data:         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
city: new york


data:         day      city  temperature  windspeed  event
0  1/1/2017  new york           32          6   Rain
1  1/2/2017  new york           36          7  Sunny
2  1/3/2017  new york           28         12   Snow
3  1/4/2017  new york           33          7  Sunny
city: paris


data:          day   city  temperature  windspeed   event
8   1/1/2017  paris           45         20   Sunny
9   1/2/2017  paris           50         13  Cloudy
10  1/3/2017  paris           54          8  Cloudy
11  1/4/2017  paris           42         10  Cloudy
```

This is similar to SQL,

SELECT * from weather_data GROUP BY city

```
g.get_group('mumbai')
```

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

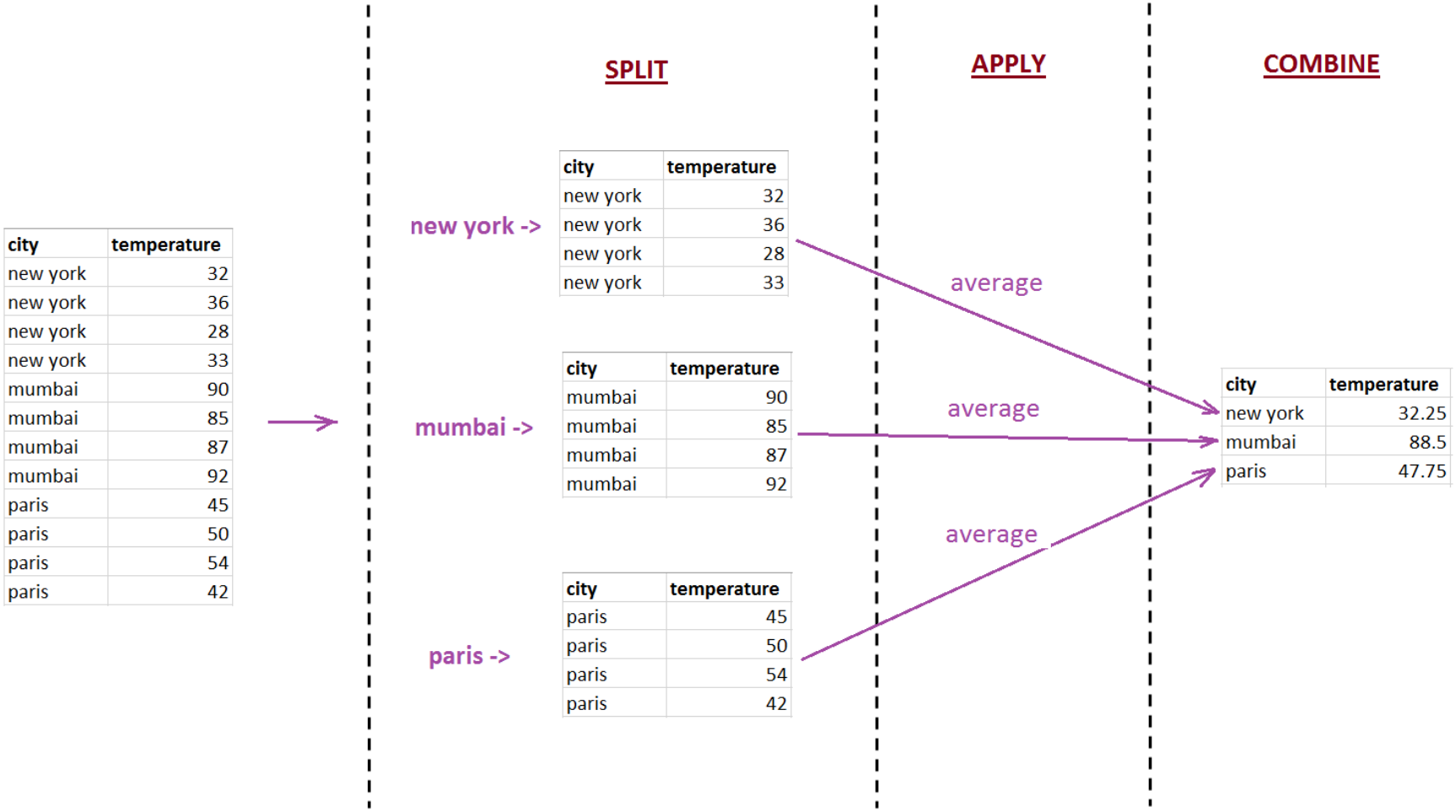
```
g.max()
```

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

```
g.mean()
```

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

This method of splitting your dataset in smaller groups and then applying an operation (such as min or max) to get aggregate result is called Split-Apply-Combine. It is illustrated in a diagram below


SPLIT
APPLY
COMBINE
city temperature
new york 32
new york 36
new york 28
new york 33
mumbai 90
mumbai 85
mumbai 87
mumbai 92
paris 45
paris 50
paris 54
paris 42
new york ->
mumbai ->
paris ->
average
average
average
city temperature
new york 32.25
mumbai 88.5
paris 47.75

```
g.min()
```

```
g.plot()
```

	day	temperature	windspeed	event

g.describe()

	temperature								windspeed							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
city																
mumbai	4.0	88.50	3.109126	85.0	86.50	88.5	90.50	92.0	4.0	9.25	5.057997	5.0	5.00	8.5	12.75	15.0
new york	4.0	32.25	3.304038	28.0	31.00	32.5	33.75	36.0	4.0	8.00	2.708013	6.0	6.75	7.0	8.25	12.0
paris	4.0	47.75	5.315073	42.0	44.25	47.5	51.00	54.0	4.0	12.75	5.251984	8.0	9.50	11.5	14.75	20.0

g.size()

```
city
mumbai      4
new york    4
paris       4
dtype: int64
```

g.count()

	day	temperature	windspeed	event
city				
mumbai	4	4	4	4
new york	4	4	4	4
paris	4	4	4	4

%matplotlib inline

g.plot()

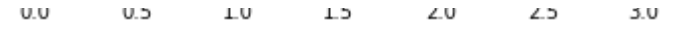
```
city
mumbai      Axes(0.125,0.125;0.775x0.755)
new york    Axes(0.125,0.125;0.775x0.755)
paris       Axes(0.125,0.125;0.775x0.755)
dtype: object
```

Group data using custom function: Let's say you want to group your data using custom function. Here the requirement is to create three groups

1. Days when temperature was between 80 and 90
2. Days when it was between 50 and 60
3. Days when it was anything else

For this you need to write custom grouping function and pass that to groupby

```
def grouper(df, idx, col):
    if 80 <= df[col].loc[idx] <= 90:
        return '80-90'
    elif 50 <= df[col].loc[idx] <= 60:
        return '50-60'
    else:
        return 'others'
```



```
g = df.groupby(lambda x: grouper(df, x, 'temperature'))
g
```

```
<pandas.core.groupby.groupby.DataFrameGroupBy object at 0x0000018EE31DCDA0>
```


```
for key, d in g:
```

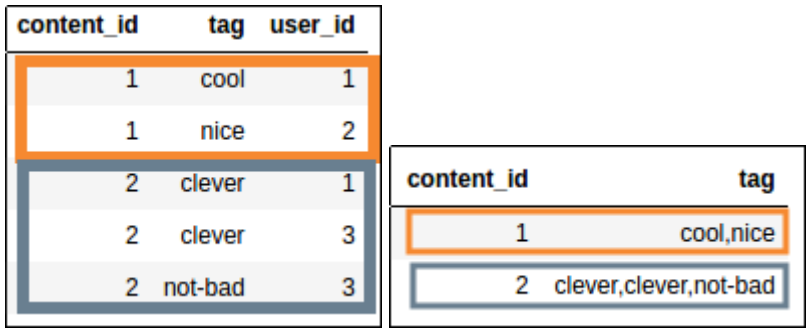
```
    print("Group by Key: {}\n".format(key))
    print(d)
```

```
Group by Key: 50-60

         day   city  temperature  windspeed   event
9   1/2/2017  paris           50         13  Cloudy
10  1/3/2017  paris           54          8  Cloudy
Group by Key: 80-90

        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
Group by Key: others

         day      city  temperature  windspeed   event
0   1/1/2017  new york           32          6    Rain
1   1/2/2017  new york           36          7   Sunny
2   1/3/2017  new york           28         12    Snow
3   1/4/2017  new york           33          7   Sunny
7   1/4/2017    mumbai           92          5    Rain
8   1/1/2017     paris           45         20   Sunny
11  1/4/2017     paris           42         10  Cloudy
```

```
import pandas as pd

df = pd.DataFrame({
    'user_id':[1,2,1,3,3,],
    'content_id':[1,1,2,2,2],
    'tag':['cool','nice','clever','clever','not-bad']
})
```

```
df.groupby("content_id")['tag'].apply(lambda tags: ','.join(tags))
```

```
content_id
1                cool,nice
2    clever,clever,not-bad
Name: tag, dtype: object
```



content_id	tag	user_id
1	cool	1
1	nice	2
2	clever	1
2	clever	3
2	not-bad	3

content_id	tag
1	cool,nice
2	clever,clever,not-bad

```
df.groupby("content_id")["user_id"].nunique()
```

```
content_id
1    2
2    2
Name: user_id, dtype: int64
```

```
df.groupby("content_id")["user_id"].nunique().to_frame()
```

content_id	user_id
1	2
2	2

```
df.groupby("content_id")["user_id"].nunique().to_frame().reset_index()
```

	content_id	user_id
0	1	2
1	2	2

```
df = pd.DataFrame({
    'value':[20.45,22.89,32.12,111.22,33.22,100.00,99.99],
    'product':['table','chair','chair','mobile phone','table','mobile phone','table']
})

df1 = df.groupby('product')['value'].sum().to_frame().reset_index()

df1
```

	product	value
0	chair	55.01
1	mobile phone	211.22
2	table	153.66

```
df2 = df.groupby('product')['value'].sum().to_frame().reset_index().sort_values(by='val
df2
```

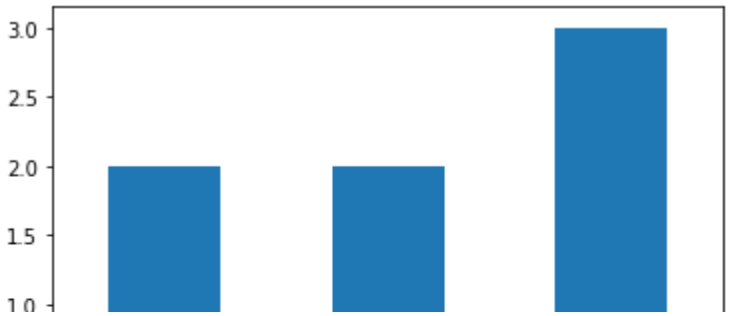
	product	value
0	chair	55.01
2	table	153.66
1	mobile phone	211.22

	product	value
0	table	20.45
1	chair	22.89
2	chair	32.12
3	mobile phone	111.22
4	table	33.22
5	mobile phone	100.00
6	table	99.99

	product	value
0	chair	55.01
1	mobile phone	211.22
2	table	153.66

	product	value
0	chair	55.01
2	table	153.66
1	mobile phone	211.22

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.DataFrame({
    'value':[20.45,22.89,32.12,111.22,33.22,100.00,99.99],
    'product':['table','chair','chair','mobile phone','table','mobile phone','table']
})

#plt.clf()
df.groupby('product').size().plot(kind='bar')
plt.show()
```




```
df = pd.DataFrame({
    'value':[20.45,22.89,32.12,111.22,33.22,100.00,99.99],
    'product':['table','chair','chair','mobile phone','table','mobile phone','table'],
    'val':[10,20,30,40,50,60,70],
    'str' : ['a','b','c','d','e','f','a']
})

#plt.clf()
df.groupby('product').sum().plot(kind='bar')
plt.show(),
```

```
<Figure size 432x288 with 0 Axes>
```


```
df = pd.DataFrame({
    'value':[20.45,22.89,32.12,111.22,33.22,100.00,99.99],
    'product':['table','chair','chair','mobile phone','table','mobile phone','table'],
    'price':[100,200,300,400,500,600,700]
})

# call
grouped_df = df.groupby('product').agg({'value':['min','max','mean']})

grouped_df
```

	value		
	min	max	mean
product			
chair	22.89	32.12	27.505
mobile phone	100.00	111.22	105.610
table	20.45	99.99	51.220

```
grouped_df.columns.values
```

```
array([('value', 'min'), ('value', 'max'), ('value', 'mean')],
      dtype=object)
```

```
grouped_df.columns = ['_'.join(col) for col in grouped_df.columns.values]
grouped_df = grouped_df.reset_index()
```

```
grouped_df
```

	product	value_min	value_max	value_mean
0	chair	22.89	32.12	27.505
1	mobile phone	100.00	111.22	105.610
2	table	20.45	99.99	51.220

```
grouped_df = df.groupby('product')['value'].sum().reset_index(name='value_sum')
```

```
grouped_df
```

	product	value_sum
0	chair	55.01
1	mobile phone	211.22
2	table	153.66

```
df.groupby('product')['value'].apply(lambda group_series: group_series.tolist()).reset_
```

	product	value
0	chair	[22.89, 32.12]
1	mobile phone	[111.22, 100.0]
2	table	[20.45, 33.22, 99.99]

```
# you can define a function like this or use a lambda function
def count_even_numbers(series):
```

```
def count_even_numbers(series):
  return len([elem for elem in series if elem % 2 == 0 ])

df.groupby('product')['value'].apply(count_even_numbers).reset_index(name='num_even_num
```

	product	num_even_numbers
0	chair	0
1	mobile phone	1
2	table	0