Analyzing Data using groupby

In [2]:

```
import numpy as np
```

In [4]:

```
import pandas as pd
```

In [5]:

```
p = {'items':['apple','apple','orange','orange','mango','mango','mango'], 'days':['mon','tue','wed','thurs','fri'
,'sat','sun'], 'sales':[100,80,200,100,50,70,120]}
```

In [6]:

```
df = pd.DataFrame(p)
```

In [7]:

```
df
```

Out[7]:

	items	days	sales
0	apple	mon	100
1	apple	tue	80
2	orange	wed	200
3	orange	thurs	100
4	mango	fri	50
5	mango	sat	70
6	mango	sun	120

In [8]:

```
x = df.groupby('items')
```

In [9]:

```
x.mean()
```

Out[9]:

	sales
items	
apple	90
mango	80
orange	150

In [10]:

```
x.sum()
```

Out[10]:

	sales
items	
apple	180
mango	240
orange	300

In [11]:

```
x.std()
```

Out[11]:

items	sales
apple	14.142136
mango	36.055513
orange	70.710678

In [12]:

```
x.count()
```

Out[12]:

items	days	sales
apple	2	2
mango	3	3
orange	2	2

In [13]:

```
x.max()
```

Out[13]:

items	days	sales
apple	tue	100
mango	sun	120
orange	wed	200

In [14]:

```
x.min()
```

Out[14]:

items	days	sales
apple	mon	80
mango	fri	50
orange	thurs	100

In [15]:

```
x.describe()
```

Out[15]:

	sales							
	count	mean	std	min	25%	50%	75%	max
items								
apple	2.0	90.0	14.142136	80.0	85.0	90.0	95.0	100.0
mango	3.0	80.0	36.055513	50.0	60.0	70.0	95.0	120.0
orange	2.0	150.0	70.710678	100.0	125.0	150.0	175.0	200.0

In [16]:

```
x.describe().transpose()
```

Out[16]:

	items	apple	mango	orange
sales	count	2.000000	3.000000	2.000000
	mean	90.000000	80.000000	150.000000
	std	14.142136	36.055513	70.710678
	min	80.000000	50.000000	100.000000
	25%	85.000000	60.000000	125.000000
	50%	90.000000	70.000000	150.000000
	75%	95.000000	95.000000	175.000000
	max	100.000000	120.000000	200.000000

Joining

In [2]:

```
import numpy as np
import pandas as pd
```

In [3]:

```
x1 = {'a':[1,2,3], 'b':[5,6,7]}
y1 = {'c':[3,4,5], 'd':[2,3,6]}
x = pd.DataFrame(x1, index=['p1','p2','p3'])
y = pd.DataFrame(y1, index=['p1','p2','p3'])
```

In [4]:

```
x
```

Out[4]:

	a	b
p1	1	5
p2	2	6
p3	3	7

In [5]:

```
y
```

Out[5]:

	c	d
p1	3	2
p2	4	3
p3	5	6

In [6]:

```
x.join(y)
```

Out[6]:

	a	b	c	d
p1	1	5	3	2
p2	2	6	4	3
p3	3	7	5	6

In [7]:

```
x1 = {'a':[1,2,3], 'b':[5,6,7]}
y1 = {'c':[3,4,5], 'd':[2,3,6]}
x = pd.DataFrame(x1, index=['p1','p2','p3'])
y = pd.DataFrame(y1, index=['p1','p5','p6'])
```

In [8]:

```
x.join(y, how='outer')
```

Out[8]:

	a	b	c	d
p1	1.0	5.0	3.0	2.0
p2	2.0	6.0	NaN	NaN
p3	3.0	7.0	NaN	NaN
p5	NaN	NaN	4.0	3.0
p6	NaN	NaN	5.0	6.0

In [9]:

```
x.join(y, how='inner')
```

Out[9]:

	a	b	c	d
p1	1	5	3	2

In [11]:

```
x.join(y, how='left')
```

Out[11]:

	a	b	c	d
p1	1	5	3.0	2.0
p2	2	6	NaN	NaN
p3	3	7	NaN	NaN

In [12]:

```
x.join(y, how='right')
```

Out[12]:

	a	b	c	d
p1	1.0	5.0	3	2
p5	NaN	NaN	4	3
p6	NaN	NaN	5	6

Concatinating

In [1]:

```
import numpy as np
```

In [2]:

```
import pandas as pd
```

In [10]:

```
x1 = {'a':[1,1,1,1,1], 'b':[1,1,1,1,1], 'c':[1,1,1,1,1], 'd':[1,1,1,1,1], 'e':[1,1,1,1,1]}
x2 = {'e':[2,2,2,2,2], 'f':[2,2,2,2,2], 'g':[2,2,2,2,2], 'h':[2,2,2,2,2], 'i':[2,2,2,2,2]}
x3 = {'a':[3,3,3,3,3], 'b':[3,3,3,3,3], 'c':[3,3,3,3,3], 'd':[3,3,3,3,3], 'e':[3,3,3,3,3]}
```

In [11]:

```
df1 = pd.DataFrame(x1, index=[1,2,3,4,5])
df2 = pd.DataFrame(x2, index=[1,2,3,4,5])
df3 = pd.DataFrame(x3, index=[5,6,7,8,9])
```

In [12]:

```
pd.concat([df1,df2])
```

```
C:\Users\A3878355\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:1: Futur
eWarning: Sorting because non-concatenation axis is not aligned. A future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  """Entry point for launching an IPython kernel.
```

Out[12]:

	a	b	c	d	e	f	g	h	i
1	1.0	1.0	1.0	1.0	1	NaN	NaN	NaN	NaN
2	1.0	1.0	1.0	1.0	1	NaN	NaN	NaN	NaN
3	1.0	1.0	1.0	1.0	1	NaN	NaN	NaN	NaN
4	1.0	1.0	1.0	1.0	1	NaN	NaN	NaN	NaN
5	1.0	1.0	1.0	1.0	1	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	2	2.0	2.0	2.0	2.0
2	NaN	NaN	NaN	NaN	2	2.0	2.0	2.0	2.0
3	NaN	NaN	NaN	NaN	2	2.0	2.0	2.0	2.0
4	NaN	NaN	NaN	NaN	2	2.0	2.0	2.0	2.0
5	NaN	NaN	NaN	NaN	2	2.0	2.0	2.0	2.0

In [13]:

```
pd.concat([df1,df2], axis = 1)
```

Out[13]:

	a	b	c	d	e	e	f	g	h	i
1	1	1	1	1	1	2	2	2	2	2
2	1	1	1	1	1	2	2	2	2	2
3	1	1	1	1	1	2	2	2	2	2
4	1	1	1	1	1	2	2	2	2	2
5	1	1	1	1	1	2	2	2	2	2

In [14]:

```
pd.concat([df1,df3], axis=0)
```

Out[14]:

	a	b	c	d	e
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1
5	3	3	3	3	3
6	3	3	3	3	3
7	3	3	3	3	3
8	3	3	3	3	3
9	3	3	3	3	3

In [15]:

```
pd.concat([df1,df2,df3])
```

```
C:\Users\A3878355\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:1: Futur
eWarning: Sorting because non-concatenation axis is not aligned. A future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  """Entry point for launching an IPython kernel.
```

Out[15]:

	a	b	c	d	e	f	g	h	i
1	1.0	1.0	1.0	1.0	1	NaN	NaN	NaN	NaN
2	1.0	1.0	1.0	1.0	1	NaN	NaN	NaN	NaN
3	1.0	1.0	1.0	1.0	1	NaN	NaN	NaN	NaN
4	1.0	1.0	1.0	1.0	1	NaN	NaN	NaN	NaN
5	1.0	1.0	1.0	1.0	1	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	2	2.0	2.0	2.0	2.0
2	NaN	NaN	NaN	NaN	2	2.0	2.0	2.0	2.0
3	NaN	NaN	NaN	NaN	2	2.0	2.0	2.0	2.0
4	NaN	NaN	NaN	NaN	2	2.0	2.0	2.0	2.0
5	NaN	NaN	NaN	NaN	2	2.0	2.0	2.0	2.0
5	3.0	3.0	3.0	3.0	3	NaN	NaN	NaN	NaN
6	3.0	3.0	3.0	3.0	3	NaN	NaN	NaN	NaN
7	3.0	3.0	3.0	3.0	3	NaN	NaN	NaN	NaN
8	3.0	3.0	3.0	3.0	3	NaN	NaN	NaN	NaN
9	3.0	3.0	3.0	3.0	3	NaN	NaN	NaN	NaN

In [16]:

```
pd.concat([df1,df3], axis=1)
```

Out[16]:

	a	b	c	d	e	a	b	c	d	e
1	1.0	1.0	1.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN
2	1.0	1.0	1.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN
3	1.0	1.0	1.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN
4	1.0	1.0	1.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN
5	1.0	1.0	1.0	1.0	1.0	3.0	3.0	3.0	3.0	3.0
6	NaN	NaN	NaN	NaN	NaN	3.0	3.0	3.0	3.0	3.0
7	NaN	NaN	NaN	NaN	NaN	3.0	3.0	3.0	3.0	3.0
8	NaN	NaN	NaN	NaN	NaN	3.0	3.0	3.0	3.0	3.0
9	NaN	NaN	NaN	NaN	NaN	3.0	3.0	3.0	3.0	3.0

Merging

In [1]:

```
import numpy as np
import pandas as pd
```

In [4]:

```
df1 = pd.DataFrame({'key1':[1,2,3], 'a':[5,6,7], 'b': [2,3,4]})
df2 = pd.DataFrame({'key1':[1,2,3], 'c':[5,8,9], 'd': [1,2,9]})
```

In [5]:

```
pd.merge(df1,df2)
```

Out[5]:

	key1	a	b	c	d
0	1	5	2	5	1
1	2	6	3	8	2
2	3	7	4	9	9

In [6]:

```
df1 = pd.DataFrame({'key1':[1,2,3], 'a':[5,6,7], 'b': [2,3,4]})
df2 = pd.DataFrame({'key1':[1,2,4], 'c':[5,8,9], 'd': [1,2,9]})
```

In [7]:

```
pd.merge(df1,df2)
```

Out[7]:

	key1	a	b	c	d
0	1	5	2	5	1
1	2	6	3	8	2

In [8]:

```
pd.merge(df1,df2, how='outer')
```

Out[8]:

	key1	a	b	c	d
0	1	5.0	2.0	5.0	1.0
1	2	6.0	3.0	8.0	2.0
2	3	7.0	4.0	NaN	NaN
3	4	NaN	NaN	9.0	9.0

In [9]:

```
pd.merge(df1,df2, how='left')
```

Out[9]:

	key1	a	b	c	d
0	1	5	2	5.0	1.0
1	2	6	3	8.0	2.0
2	3	7	4	NaN	NaN

In [10]:

```
df1 = pd.DataFrame({'key1':[1,2,3], 'a':[5,6,7], 'b': [2,3,4], 'key2': [5,2,3]})
df2 = pd.DataFrame({'key1':[1,2,4], 'c':[5,8,9], 'd': [1,2,9], 'key2': [5,2,3]})
```

In [11]:

```
pd.merge(df1,df2, how='inner', on=['key1','key2'])
```

Out[11]:

	key1	a	b	key2	c	d
0	1	5	2	5	5	1
1	2	6	3	2	8	2

In [12]:

```
pd.merge(df1,df2, how='inner')
```

Out[12]:

	key1	a	b	key2	c	d
0	1	5	2	5	5	1
1	2	6	3	2	8	2

In [13]:

```
pd.merge(df1,df2, how='inner', on='key1')
```

Out[13]:

	key1	a	b	key2_x	c	d	key2_y
0	1	5	2	5	5	1	5
1	2	6	3	2	8	2	2