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

Aggregations refer to any data transformation that produces scalar values from arrays.

Table 10-1. Optimized groupby methods

Function name	Description
count	Number of non-NA values in the group
sum	Sum of non-NA values
mean	Mean of non-NA values
median	Arithmetic median of non-NA values
std, var	Unbiased (n – 1 denominator) standard deviation and variance
min, max	Minimum and maximum of non-NA values
prod	Product of non-NA values
first, last	First and last non-NA values

quantile - computes sample quantiles of a Series or a DataFrame's columns.

In [2]:
```
df = pd.DataFrame({'key1' : ['a', 'a', 'b', 'b', 'a'],
....: 'key2' : ['one', 'two', 'one', 'two', 'one'],
....: 'data1' : np.random.randn(5),
....: 'data2' : np.random.randn(5)})
df
```

Out[2]:

	key1	key2	data1	data2
0	a	one	0.711414	0.063523
1	a	two	0.401743	0.758611
2	b	one	-0.312033	-1.053392
3	b	two	1.435064	-0.318484
4	a	one	-1.611115	-0.437059

In [3]:
```
grouped = df.groupby('key1')
grouped['data1'].quantile(0.5)
```

Out[3]:
```
key1
a    0.401743
b    0.561516
Name: data1, dtype: float64
```

In [ ]:
```

```

.... see book

In [4]:
```
grouped.describe()
```

Out[4]:

	data1								data2
	count	mean	std	min	25%	50%	75%	max	count
key1									
a	3.0	-0.165986	1.261060	-1.611115	-0.604686	0.401743	0.556579	0.711414	3.0
b	2.0	0.561516	1.235385	-0.312033	0.124741	0.561516	0.998290	1.435064	2.0

QUESTION - compute Interquartile Range

```
Given data in a list [13, 15,16, 20, 23, 21,33,56,67,89]
Write code to find the range and interquartile range.

Given [13, 15,16, 20, 23, 21,33,56,67,89]    (numbers are in order)
The total number of terms n = 10.
So, the data set is divided into two parts: 13, 15,16, 20, 23 and
21,33,56,67,89

Q1 = Median of first part = 16
Q3 = Median of second part = 56
Now, use the formula for interquartile range i.e., IQR = Q3 – Q1
IQR = 56-15 = 40
```

In [5]:
```
import numpy as np

lst = [15,20, 23,13,16, 21,33, 56,67,89]
min = np.min(lst)
max = np.max(lst)

range = max - min
print('Range =',range)

# Q1 - First quartile
Q1 = np.median(lst[:5])
print(Q1)
# Q3 - Third quartile
Q3 = np.median(lst[5:])
print(Q3)
# Interquartile range (IQR)
IQR = Q3 - Q1

print('Interquartile range =',IQR)
```

```
Range = 76
16.0
56.0
Interquartile range = 40.0
```

In [ ]:
```

```

```
Custom aggregation functions are generally much slower than the
optimized functions found in Table 10-1.
This is because there is some extra overhead (function calls, data
rearrangement) in constructing
the intermediate group data chunks
```

Column-Wise and Multiple Function Application

```
.... see textbook
```

EXAMPLE

In [6]:
```
data1 = pd.DataFrame({'State':['Delhi', 'Bihar', 'Delhi', 'Bihar', 'Haryana
                      'Polulation':[30,55,50,35, 45]})
data1
```

Out[6]:

	State	Polulation
0	Delhi	30
1	Bihar	55
2	Delhi	50
3	Bihar	35
4	Haryana	45

In [8]:
```
data2 = data1.set_index('State')
data2
```

Out[8]:

State	Polulation
Delhi	30
Bihar	55
Delhi	50
Bihar	35
Haryana	45

In [9]:
```
groups = data2.groupby('State')
```

In [10]:
```
groups.mean()
```

Out[10]:

State	Polulation
Bihar	45.0
Delhi	40.0
Haryana	45.0

In [11]:
```
functions = ['count', 'mean', 'max', 'min']
groups.agg(functions)
```

Out[11]:

	Polulation			
	count	mean	max	min
State				
Bihar	2	45.0	55	35
Delhi	2	40.0	50	30
Haryana	1	45.0	45	45

Returning Aggregated Data Without Row Indexes ¶

```
.... see textbook
```

In [ ]:
```

```