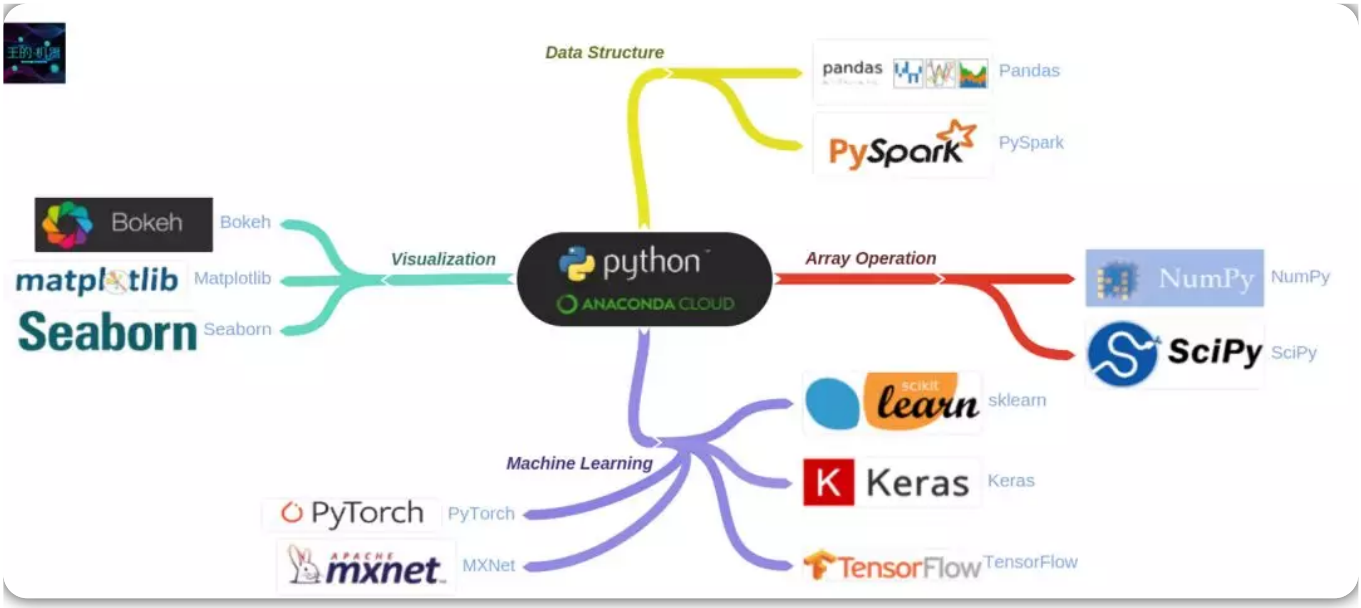
盘一盘 Python 特别篇 15 - Pivot Table

原创 王圣元 王的机器 6月12日

来自专辑

Python



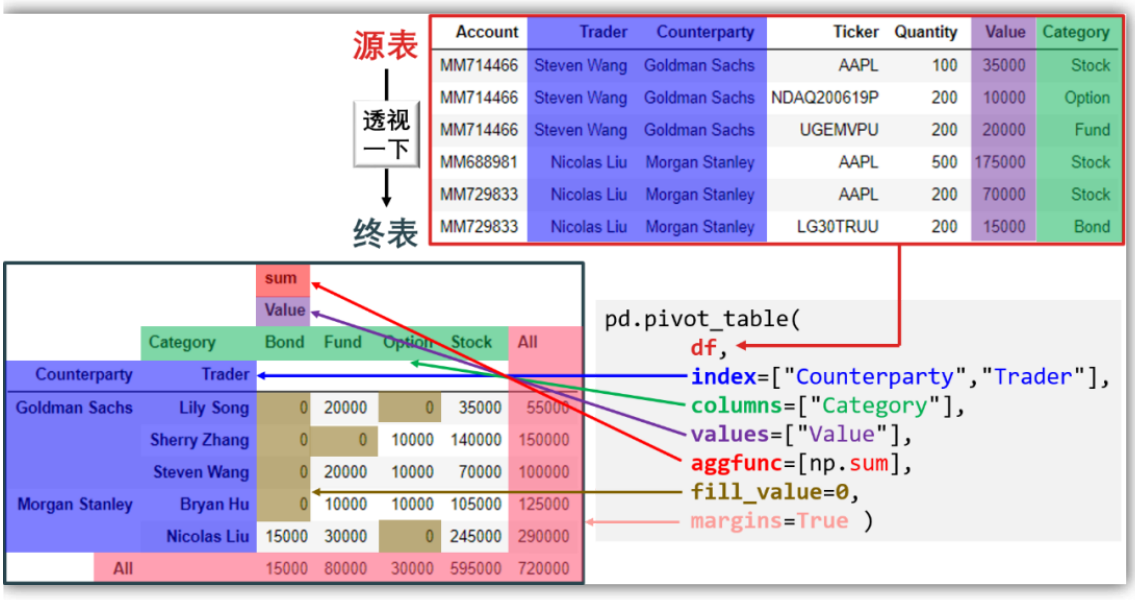
本文含 3796 字，22 图表截屏
建议阅读 20 分钟

本文是 Python 系列的特别篇的第十五篇

- 特别篇 1 - PyEcharts TreeMap
- 特别篇 2 - 面向对象编程
- 特别篇 3 - 两大利「器」
- 特别篇 4 - 装饰器
- 特别篇 5 - Sklearn 0.22
- 特别篇 6 - Jupyter Notebook
- 特别篇 7 - 格式化字符串
- 特别篇 8 - 正则表达式
- 特别篇 9 - 正则表达式实战
- 特别篇 10 - 错误类型
- 特别篇 11 - 异常处理
- 特别篇 12 - Collection
- 特别篇 13 - Matplotlib Animation
- 特别篇 14 - All 和 Any
- 特别篇 15 - 透视表 Pivot Table

从功能上讲，Pandas 中用透视表 (pivot table) 和 Excel 里面的透视表是一样的。透视表是一种做多维数据分析的工具，还记得 **Pandas** 的 split-apply-combine 三部曲吗？首先用 groupby 分组，再平行将某个函数应用到各组上，最后自动连接成一个总表。今天介绍的 pivot_table() 函数可以将上面“**拆分-应用-结合**”三个步骤用一行来完成。

先看一张图：



Account	Trader	Counterparty	Ticker	Quantity	Value	Category
MM714466	Steven Wang	Goldman Sachs	AAPL	100	35000	Stock
MM714466	Steven Wang	Goldman Sachs	NDAQ200619P	200	10000	Option
MM714466	Steven Wang	Goldman Sachs	UGEMVPU	200	20000	Fund
MM688981	Nicolas Liu	Morgan Stanley	AAPL	500	175000	Stock
MM729833	Nicolas Liu	Morgan Stanley	AAPL	200	70000	Stock
MM729833	Nicolas Liu	Morgan Stanley	LG30TRUU	200	15000	Bond

		sum				
		Value				
	Category	Bond	Fund	Option	Stock	All
Counterparty	Trader					
Goldman Sachs	Lily Song	0	20000	0	35000	55000
	Sherry Zhang	0	0	10000	140000	150000
	Steven Wang	0	20000	10000	70000	100000
Morgan Stanley	Bryan Hu	0	10000	10000	105000	125000
	Nicolas Liu	15000	30000	0	245000	290000
All		15000	80000	30000	595000	720000

Pivot 字面意思是支点，即上图中的 index 和 columns 指定的行和列标签，支点可想理解成数据 (values) 在哪个维度上做整合 (aggfunc)，再吧 NaN 值用 fill_value 替代，按行按列加总 (margin=True)。

理解还有些难度？没问题，看完本帖就懂了。

数据

首先从 csv 读数据。

	A	B	C	D	E	F	G
1	Account	Trader	Counterparty	Ticker	Quantity	Value	Category
2	MM714466	Steven Wang	Goldman Sachs	AAPL	100	35000	Stock
3	MM714466	Steven Wang	Goldman Sachs	NDAQ200619P	200	10000	Option
4	MM714466	Steven Wang	Goldman Sachs	UGEMVPU	200	20000	Fund
5	MM737550	Steven Wang	Goldman Sachs	AAPL	100	35000	Stock
6	MM146832	Sherry Zhang	Goldman Sachs	AAPL	200	70000	Stock
7	MM218895	Sherry Zhang	Goldman Sachs	AAPL	200	70000	Stock
8	MM218895	Sherry Zhang	Goldman Sachs	NDAQ200619P	200	10000	Option
9	MM412290	Lily Song	Goldman Sachs	UGEMVPU	200	20000	Fund
10	MM740150	Lily Song	Goldman Sachs	AAPL	100	35000	Stock
11	MM141962	Bryan Hu	Morgan Stanley	AAPL	200	70000	Stock
12	MM163416	Bryan Hu	Morgan Stanley	AAPL	100	35000	Stock
13	MM239344	Bryan Hu	Morgan Stanley	UGEMVPU	100	10000	Fund
14	MM239344	Bryan Hu	Morgan Stanley	NDAQ200619P	200	10000	Option
15	MM307599	Nicolas Liu	Morgan Stanley	UGEMVPU	300	30000	Fund
16	MM688981	Nicolas Liu	Morgan Stanley	AAPL	500	175000	Stock
17	MM729833	Nicolas Liu	Morgan Stanley	AAPL	200	70000	Stock
18	MM729833	Nicolas Liu	Morgan Stanley	LG30TRUU	200	15000	Bond

```
1 df = pd.read_csv('PB Sales.csv')
2 df
```

	Account	Trader	Counterparty	Ticker	Quantity	Value	Category
0	MM714466	Steven Wang	Goldman Sachs	AAPL	100	35000	Stock
1	MM714466	Steven Wang	Goldman Sachs	NDAQ200619P	200	10000	Option
2	MM714466	Steven Wang	Goldman Sachs	UGEMVPU	200	20000	Fund
3	MM737550	Steven Wang	Goldman Sachs	AAPL	100	35000	Stock
4	MM146832	Sherry Zhang	Goldman Sachs	AAPL	200	70000	Stock
5	MM218895	Sherry Zhang	Goldman Sachs	AAPL	200	70000	Stock
6	MM218895	Sherry Zhang	Goldman Sachs	NDAQ200619P	200	10000	Option
7	MM412290	Lily Song	Goldman Sachs	UGEMVPU	200	20000	Fund
8	MM740150	Lily Song	Goldman Sachs	AAPL	100	35000	Stock
9	MM141962	Bryan Hu	Morgan Stanley	AAPL	200	70000	Stock
10	MM163416	Bryan Hu	Morgan Stanley	AAPL	100	35000	Stock
11	MM239344	Bryan Hu	Morgan Stanley	UGEMVPU	100	10000	Fund
12	MM239344	Bryan Hu	Morgan Stanley	NDAQ200619P	200	10000	Option
13	MM307599	Nicolas Liu	Morgan Stanley	UGEMVPU	300	30000	Fund
14	MM688981	Nicolas Liu	Morgan Stanley	AAPL	500	175000	Stock
15	MM729833	Nicolas Liu	Morgan Stanley	AAPL	200	70000	Stock
16	MM729833	Nicolas Liu	Morgan Stanley	LG30TRUU	200	15000	Bond

设置"单行"为 Pivot

创建透视表的 pivot_table() 函数里面的参数设置很多，学习它最有效的方式是每一步设置一个参数，检查结果是否符合预期。

先从最简单的语法开始，只设置 index='Account'，通用语法如下：

```
pd.pivot_table(df, index=label_str)
```

```
1 pd.pivot_table( df, index="Account" )
```

Account	Quantity	Value
MM141962	200.000000	70000.000000
MM146832	200.000000	70000.000000
MM163416	100.000000	35000.000000
MM218895	200.000000	40000.000000
MM239344	150.000000	10000.000000
MM307599	300.000000	30000.000000
MM412290	200.000000	20000.000000
MM688981	500.000000	175000.000000
MM714466	166.666667	21666.666667
MM729833	200.000000	42500.000000
MM737550	100.000000	35000.000000
MM740150	100.000000	35000.000000

从上表结果看，Price 和 Quantity 两列按照 Account 以**某种**方式合并了。看原 df 最后两行，账户 MM729833 合并成一行，对应的 Price 和 Quantity 是 42500 和 200，看来**某种**方式是求平均。现在大概可以猜出 pivot_table() 函数中有个参数用来设置**整合**方式，而默认值为平均。

设置"多行"为 Pivot

上例设置单个 index，接下来看看设置多个 index 的结果是什么样的。这时用列表来存储多个 index。通用语法如下：

```
pd.pivot_table(df, index=label_list)
```

```
pd.pivot_table( df, index=["Account","Trader","Counterparty"] )
```

			Quantity	Value
Account	Trader	Counterparty		
MM141962	Bryan Hu	Morgan Stanley	200.000000	70000.000000
MM146832	Sherry Zhang	Goldman Sachs	200.000000	70000.000000
MM163416	Bryan Hu	Morgan Stanley	100.000000	35000.000000
MM218895	Sherry Zhang	Goldman Sachs	200.000000	40000.000000
MM239344	Bryan Hu	Morgan Stanley	150.000000	10000.000000
MM307599	Nicolas Liu	Morgan Stanley	300.000000	30000.000000
MM412290	Lily Song	Goldman Sachs	200.000000	20000.000000
MM688981	Nicolas Liu	Morgan Stanley	500.000000	175000.000000
MM714466	Steven Wang	Goldman Sachs	166.666667	21666.666667
MM729833	Nicolas Liu	Morgan Stanley	200.000000	42500.000000
MM737550	Steven Wang	Goldman Sachs	100.000000	35000.000000
MM740150	Lily Song	Goldman Sachs	100.000000	35000.000000

上表显示着每个 Account 对应的交易员和交易对手的信息，但有点乱。

一个交易员管理一个或多个账户，多个交易员可以和一个交易对手交易，改变 index 里面的标签顺序，先按 Counterparty 合并，再按 Trader 合并。

```
pd.pivot_table( df, index=["Counterparty","Trader"] )
```

		Quantity	Value
Counterparty	Trader		
Goldman Sachs	Lily Song	150	27500
	Sherry Zhang	200	50000
	Steven Wang	150	25000
Morgan Stanley	Bryan Hu	150	31250
	Nicolas Liu	300	72500

到目前为止，我们只设置了 index，那为什么只在 Price 和 Quantity 两列上做整合呢？因为这两列的值是数值型（`int`，`float`），而其他例的值是非数值型（`object`），用 df.dtypes 就可看出。

```
df.dtypes
```

```
Account object
Trader object
Counterparty object
Ticker object
Quantity int64
Value int64
Category object
dtype: object
```

设定被整合的数据

如果只看 Price 列下的整合结果，只需设置 values='Price' 或者 values=['Price']，通用语法如下：

```
pd.pivot_table(df, index=label_list, values=label_list)
```

```
pd.pivot_table( df, index=["Counterparty","Trader"],
values="Value" )
```

		Value
Counterparty	Trader	
Goldman Sachs	Lily Song	27500
	Sherry Zhang	50000
	Steven Wang	25000
Morgan Stanley	Bryan Hu	31250
	Nicolas Liu	72500

设置整合函数

默认整合函数是求平均，如果要用求和的函数需要设置 aggfunc=np.sum，通用语法为

```
pd.pivot_table(df, index=label_list, values=label_list, aggfunc=func)
```

```
pd.pivot_table( df, index=["Counterparty","Trader"],
                    values=["Value"],
                    aggfunc=np.sum )
```

Counterparty	Trader	Value
Goldman Sachs	Lily Song	55000
	Sherry Zhang	150000
	Steven Wang	100000
Morgan Stanley	Bryan Hu	125000
	Nicolas Liu	290000

aggfunc 参数可以被设置为多个函数，用列表储存，通用语法为

```
pd.pivot_table(df, index=label_list, values=label_list, aggfunc=func_list)
```

```
pd.pivot_table( df, index=["Counterparty","Trader"],
                    values=["Value"],
                    aggfunc=[len, np.sum] )
```

		len	sum
		Value	Value
Counterparty	Trader		
Goldman Sachs	Lily Song	2	55000
	Sherry Zhang	3	150000
	Steven Wang	4	100000
Morgan Stanley	Bryan Hu	4	125000
	Nicolas Liu	4	290000

设定"列" 为 Pivot

如果进一步想看按产品类别划分后的整合结果，可以设置 columns=['Category]，通用语法为，

```
pd.pivot_table(df, index=label_list, values=label_list, columns=label_list, aggfunc=func_list)
```

```
pd.pivot_table( df, index=["Counterparty","Trader"],
                    values=["Value"],
                    columns=["Category"],
                    aggfunc=[len, np.sum] )
```

		len				sum			
		Value				Value			
	Category	Bond	Fund	Option	Stock	Bond	Fund	Option	Stock
Counterparty	Trader								
Goldman Sachs	Lily Song	NaN	1.0	NaN	1.0	NaN	20000.0	NaN	35000.0
	Sherry Zhang	NaN	NaN	1.0	2.0	NaN	NaN	10000.0	140000.0
	Steven Wang	NaN	1.0	1.0	2.0	NaN	20000.0	10000.0	70000.0
Morgan Stanley	Bryan Hu	NaN	1.0	1.0	2.0	NaN	10000.0	10000.0	105000.0
	Nicolas Liu	1.0	1.0	NaN	2.0	15000.0	30000.0	NaN	245000.0

上表结果中的 NaN 不好看，可设置 fill_value=0 用零替代。

```
pd.pivot_table( df, index=["Counterparty","Trader"],
                    values=["Value"],
                    columns=["Category"],
                    aggfunc=[len, np.sum],
                    fill_value=0 )
```

		len				sum			
		Value				Value			
	Category	Bond	Fund	Option	Stock	Bond	Fund	Option	Stock
Counterparty	Trader								
Goldman Sachs	Lily Song	0	1	0	1	0	20000	0	35000
	Sherry Zhang	0	0	1	2	0	0	10000	140000
	Steven Wang	0	1	1	2	0	20000	10000	70000
Morgan Stanley	Bryan Hu	0	1	1	2	0	10000	10000	105000
	Nicolas Liu	1	1	0	2	15000	30000	0	245000

除此之外还可以把产品类别放在 index 中，改变结果的展示方式而已 (那些结果为零都没显示了，看起来更舒服点)。

```
pd.pivot_table( df, index=["Counterparty","Trader","Category"],
                    values=["Value"],
                    aggfunc=[len, np.sum],
                    fill_value=0 )
```

			len	sum
			Value	Value
Counterparty	Trader	Category		
Goldman Sachs	Lily Song	Fund	1	20000
		Stock	1	35000
	Sherry Zhang	Option	1	10000
		Stock	2	140000
	Steven Wang	Fund	1	20000
		Option	1	10000
		Stock	2	70000
Morgan Stanley	Bryan Hu	Fund	1	10000
		Option	1	10000
		Stock	2	105000
	Nicolas Liu	Bond	1	15000
		Fund	1	30000
		Stock	2	245000

设置"显示"总和

要看总计怎么办？设置 `margins=True` 就可以了。

```
pd.pivot_table( df, index=["Counterparty","Trader","Category"],
                    values=["Value","Quantity"],
                    aggfunc=[len, np.sum],
                    fill_value=0,
                    margins=True )
```

			len		sum	
			Quantity	Value	Quantity	Value
Counterparty	Trader	Category				
Goldman Sachs	Lily Song	Fund	1	1	200	20000
		Stock	1	1	100	35000
	Sherry Zhang	Option	1	1	200	10000
		Stock	2	2	400	140000
	Steven Wang	Fund	1	1	200	20000
		Option	1	1	200	10000
		Stock	2	2	200	70000
Morgan Stanley	Bryan Hu	Fund	1	1	100	10000
		Option	1	1	200	10000
		Stock	2	2	300	105000
	Nicolas Liu	Bond	1	1	200	15000
		Fund	1	1	300	30000
		Stock	2	2	700	245000
All			17	17	3300	720000

整合函数的不同设置方式

`aggfunc` 参数还可以传进一个字典来实现不同列下应用不同的整合函数，语法如下：

```
aggfunc = {col_1:func_1, col_2:func_2, ... col_n:func_n}
```

```
pd.pivot_table( df, index=["Counterparty","Trader","Category"],
                    values=["Value","Quantity"],
                    aggfunc={"Value":np.sum, "Quantity":len},
                    fill_value=0,
                    margins=True )
```

			Quantity	Value
Counterparty	Trader	Category		
Goldman Sachs	Lily Song	Fund	1	20000
		Stock	1	35000
	Sherry Zhang	Option	1	10000
		Stock	2	140000
	Steven Wang	Fund	1	20000
		Option	1	10000
		Stock	2	70000
Morgan Stanley	Bryan Hu	Fund	1	10000
		Option	1	10000
		Stock	2	105000
	Nicolas Liu	Bond	1	15000
		Fund	1	30000
		Stock	2	245000
All			17	720000

再进一步，不同列还可以应用多个函数，只需把函数名称变成函数列表就可以了。语法如下：

```
aggfunc = {col_1:func_1, col_2:func_list, ... col_n:func_n}
```

假设第二列传入一个函数列表。

```
pd.pivot_table( df, index=["Counterparty","Trader","Category"],
                    values=["Value","Quantity"],
                    aggfunc={"Value":[np.sum, min, max],
                             "Quantity":len},
```

```
                    fill_value=0 )
```

			Quantity	Value		
			len	max	min	sum
Counterparty	Trader	Category				
Goldman Sachs	Lily Song	Fund	1	20000	20000	20000
		Stock	1	35000	35000	35000
	Sherry Zhang	Option	1	10000	10000	10000
		Stock	2	70000	70000	140000
	Steven Wang	Fund	1	20000	20000	20000
		Option	1	10000	10000	10000
		Stock	2	35000	35000	70000
Morgan Stanley	Bryan Hu	Fund	1	10000	10000	10000
		Option	1	10000	10000	10000
		Stock	2	70000	35000	105000
	Nicolas Liu	Bond	1	15000	15000	15000
		Fund	1	30000	30000	30000
		Stock	2	175000	70000	245000

查询终表

一次性做出这样的表不容易，但按步摸索 (一次设置一次参数) 的方式却很简单。

一旦得到最终结果，它本质还是个数据帧，因此可以使用所有标配函数。下例用 `query()` 函数来查询名叫 Steven Wang 和 Sherry Zhang 的交易员。

```
table = pd.pivot_table( df, index=["Counterparty","Trader","Category"],
                            values=["Value","Quantity"],
                            aggfunc={"Value":np.sum, "Quantity":len},
                            fill_value=0 )
table.query('Trader == ["Steven Wang", "Sherry Zhang"]')
```

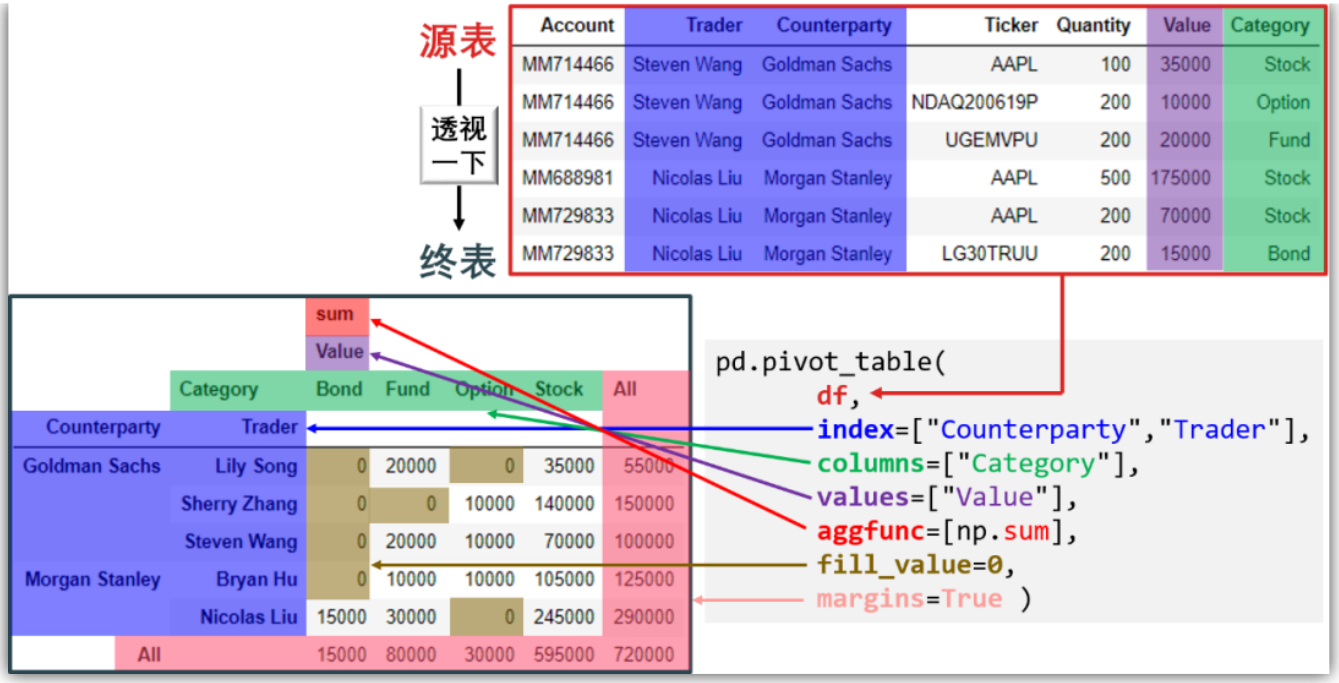
Counterparty	Trader	Category	Quantity	Value
Goldman Sachs	Sherry Zhang	Option	1	10000
		Stock	2	140000
	Steven Wang	Fund	1	20000
		Option	1	10000
		Stock	2	70000

查询所有期权和基金产品相关的信息。

```
table.query('Category == ["Fund", "Option"]')
```

Counterparty	Trader	Category	Quantity	Value
Goldman Sachs	Lily Song	Fund	1	20000
	Sherry Zhang	Option	1	10000
	Steven Wang	Fund	1	20000
		Option	1	10000
Morgan Stanley	Bryan Hu	Fund	1	10000
		Option	1	10000
	Nicolas Liu	Fund	1	30000

总结，一图胜千言！现在再看下图可视化 `pivot_table()` 函数的用法，是不是都能理解了？



Stay Tuned!

END

现在京东上有活动，扫以下二维码去买书**满 100 返 50**。