3. Pandas单表处理

本notebook介绍常用的Pandas单表操作

In [1]:

```
#import packages and set display format
import pandas as pd
from pathlib import Path
pd.set_option('display.float_format',  '{:,.2f}'.format)
pd.set_option('display.max_columns', None)
pd.set_option('display.min_rows', 30)
```

准备、读取Excel文件中的第一张工作表（sheet），存为DataFrame

In [2]:

```
folder = Path(r'D:\zs\data')
comp = pd.read_excel(folder/'comp_fin.xlsx')
```

快速预览数据，查看相关信息

In [3]:

```
comp.head()
comp.tail(10)
```

Out[3]:

	FirmID	Ind	Asset	Liability	Equity	Profit
90	C10091	Retail	28835	17721	11114	-2,671.00
91	C10092	Agri	7865	6569	1296	2,950.00
92	C10093	Serv	13335	7241	6094	4,132.00
93	C10094	Agri	27065	486	26579	2,269.00
94	C10095	Serv	43562	21048	22514	10,462.00
95	C10096	Agri	89623	1627	87996	-7,505.00
96	C10097	Retail	106597	91349	15248	29,978.00
97	C10098	Manu	80736	25277	55459	46,081.00
98	C10099	Serv	92617	45564	47053	-1,491.00
99	C10100	Retail	75962	50069	25893	-9,343.00

In [4]:

```
comp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   FirmID     100 non-null    object 
 1   Ind        100 non-null    object 
 2   Asset      100 non-null    int64  
 3   Liability  96 non-null     object 
 4   Equity     99 non-null     object 
 5   Profit     99 non-null     float64
dtypes: float64(1), int64(1), object(4)
memory usage: 4.8+ KB
```

一、选取数据

简单选取一列

与访问字典或Series中的单个元素类似。从此意义上说，可以将DataFrame看作是有一个个Series元素构成的一个字典。

In [10]:

```
# 从comp数据集中，选取Asset列数据
comp['Asset']
```

Out[10]:

```
0       32578
1       61712
2       73379
3       98646
4       85622
5       69988
6       81961
7       62616
8       53617
9       83890
10      75813
11      59817
12       5968
13       9847
14      49500
        ...
85       8574
86      50864
87      50591
88      79141
89      41596
90      28835
91       7865
92      13335
93      27065
94      43562
95      89623
96     106597
97      80736
98      92617
99      75962
Name: Asset, Length: 100, dtype: int64
```

In [ ]:

```
```

多行和多列的选取

使用.loc[]可以按照需要选取指定标签的某几行及某几列的数据。如df.loc[行维度表达式,列维度表达式]

In [6]:

```
# 使用单个标签
comp.loc[3,'Asset']
```

Out[6]:

```
98646
```

In [7]:

```
# 使用切片访问连续的行或列
comp.loc[1:4,'Asset':'Profit']
```

Out[7]:

	Asset	Liability	Equity	Profit
1	61712	54600	7652	12,342.00
2	73379	53702	19677	-14,676.00
3	98646	38581	60065	3,945.00
4	85622	46621	39001	-2,569.00

In [8]:

```
comp.loc[:,'Asset':'Profit']
```

Out[8]:

	Asset	Liability	Equity	Profit
0	32578	23483	9095	-2,607.00
1	61712	54600	7652	12,342.00
2	73379	53702	19677	-14,676.00
3	98646	38581	60065	3,945.00
4	85622	46621	39001	-2,569.00
5	69988	35518	34470	27,995.00
6	81961	28004	53957	11,474.00
7	62616	13522	48993	-6,262.00
8	53617	NaN	$53617	NaN
9	83890	25393	58497	41,106.00
10	75813	24772	51041	34,873.00
11	59817	NaN	59717	11,963.00
12	5968	$4834	1134	2,924.00
13	9847	4234	5613	1,575.00
14	49500	18810	30690	22,770.00
...	...	...	...	...
85	8574	NaN	8574	-770.00
86	50864	31455	19409	23,755.00
87	50591	35081	15510	12,383.00
88	79141	17963	61178	28,548.00
89	41596	21803	19793	5,513.00
90	28835	17721	11114	-2,671.00
91	7865	6569	1296	2,950.00
92	13335	7241	6094	4,132.00
93	27065	486	26579	2,269.00
94	43562	21048	22514	10,462.00
95	89623	1627	87996	-7,505.00
96	106597	91349	15248	29,978.00
97	80736	25277	55459	46,081.00
98	92617	45564	47053	-1,491.00
99	75962	50069	25893	-9,343.00

100 rows × 4 columns

In [9]:

```
# 使用花式索引（fancy indexing）访问不连续的行或列。如[1,3,10]、['Ind','Profit']之类的离散数字或字符串构成的列表
comp.loc[[1,3,5],['Ind','Asset','Profit']]
```

Out[9]:

	Ind	Asset	Profit
1	Retail	61712	12,342.00
3	Agri	98646	3,945.00
5	Manu	69988	27,995.00

【练习4-1】 从sales这个DataFrame中，选取行标签从5到10的记录，并选取'list_price'和'cost'这两列。

In [11]:

```
sales = pd.read_excel(folder/'sales.xlsx')
sales.loc[5:10,['list_price','cost']]
```

Out[11]:

	list_price	cost
5	17.15	10.77
6	8.05	5.13
7	7.35	4.90
8	19.50	11.24
9	13.83	8.65
10	5.09	3.22

筛选

可使用布尔式选取，来筛选出满足条件的记录（行）

In [12]:

```
1 == 2
```

Out[12]:

```
False
```

In [15]:

```
# 什么是布尔式?
comp['Asset'] > 50000
```

Out[15]:

```
0     False
1      True
2      True
3      True
4      True
5      True
6      True
7      True
8      True
9      True
10     True
11     True
12    False
13    False
14    False
      ...
85    False
86     True
87     True
88     True
89    False
90    False
91    False
92    False
93    False
94    False
95     True
96     True
97     True
98     True
99     True
Name: Asset, Length: 100, dtype: bool
```

In [18]:

```
# 选取comp数据中，总资产大于5万元的公司
comp.loc[comp['Asset']>50000,['Liability','Profit']]
```

Out[18]:

	Liability	Profit
1	54600	12,342.00
2	53702	-14,676.00
3	38581	3,945.00
4	46621	-2,569.00
5	35518	27,995.00
6	28004	11,474.00
7	13522	-6,262.00
8	NaN	NaN
9	25393	41,106.00

10	24772	34,873.00
11	NaN	11,963.00
17	69242	9,485.00
22	80619	-16,447.00
23	38239	2,401.00
24	74130	5,285.00
26	55613	10,392.00
27	26271	7,998.00
30	46723	5,913.00
31	11268	9,683.00
32	83598	23,713.00
34	16326	33,660.00
35	32637	-2,167.00
36	49808	-7,069.00
38	14080	5,845.00
40	7922	22,254.00
41	15079	21,024.00
42	25548	-19,771.00
43	27695	-3,447.00
45	23081	-7,190.00
46	66845	41,417.00
47	63154	53,213.00
49	59204	56,384.00
51	60627	-8,721.00
53	35665	26,124.00
54	22186	14,377.00
55	33414	-9,198.00
59	55934	-4,730.00
61	38117	-1,967.00
63	77239	26,507.00
64	28448	7,764.00
66	20799	22,555.00
67	30202	-7,448.00
70	5266	5,522.00
71	34863	15,710.00
74	3632	34,110.00

75	77904	99,401.00
76	82287	-31,174.00
78	46241	14,208.00
79	54703	23,719.00
82	18804	10,079.00
84	39449	59,179.00
86	31455	23,755.00
87	35081	12,383.00
88	17963	28,548.00
95	1627	-7,505.00
96	91349	29,978.00
97	25277	46,081.00
98	45564	-1,491.00
99	50069	-9,343.00

【练习4-2】 从sales中，分别选取（1）客户编号为175749的行；（2）所有region为“Midwest”的记录。

In [3]:

```
sales = pd.read_excel(folder/'sales.xlsx')
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49839 entries, 0 to 49838
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype         
---  ------           --------------  -----         
 0   customer_number  49839 non-null  int64         
 1   region           49839 non-null  object        
 2   date_of_sale     49839 non-null  datetime64[ns]
 3   item             49839 non-null  object        
 4   brand            49839 non-null  object        
 5   collection       49839 non-null  object        
 6   description      49839 non-null  object        
 7   list_price       49839 non-null  float64       
 8   cost             49839 non-null  float64       
 9   quantity_sold    49839 non-null  int64         
dtypes: datetime64[ns](1), float64(2), int64(2), object(5)
memory usage: 3.8+ MB
```

In [21]:

```
sales.loc[sales['customer_number']==175749]
```

Out[21]:

	customer_number	region	date_of_sale	item	brand	collection	description
3	175749	West	2015-01-01	351-128PC	Elements	Calloway	128" CC pull
3637	175749	West	2015-05-11	399SN	Elements	Naples	319 mm CC pull
12929	175749	West	2016-04-05	874-160BNBDL	Jeffrey Alexander	Breighton	160 mm CC pull
15246	175749	West	2016-06-19	818BNBDL	Jeffrey Alexander	Bella	Knob
17192	175749	West	2016-08-21	935-12DBAC	Jeffrey Alexander	Montclair	12" CC app. Pull
22528	175749	West	2017-02-04	4128DBAC	Jeffrey Alexander	Sonoma	128" mm CC pull
26729	175749	West	2017-06-01	G120L-BNBDL	Jeffrey Alexander	Harlow	Knob
28106	175749	West	2017-07-09	239-128SN	Elements	Brenton	128" CC pull
28375	175749	West	2017-07-16	Z115-96DBAC	Elements	Lindos	96" CC pull
38148	175749	West	2018-03-31	718DACM	Jeffrey Alexander	Glenmore	128" mm CC pull
41673	175749	West	2018-06-21	867L-SN	Jeffrey Alexander	Anwick	Knob
43472	175749	West	2018-08-03	519-18DBAC	Jeffrey Alexander	Delgado	18" CC app. Pull
44518	175749	West	2018-08-28	818-12NI	Jeffrey Alexander	Bella	12" CC app. Pull
49096	175749	West	2018-12-14	382-160BNBDL	Elements	Cosgrove	160" CC pull

In [24]:

```
sales.loc[sales['region']=='Midwest']
sales.loc[sales['region']=='Midwest','item':'collection'].head(20)
```

Out[24]:

	item	brand	collection
0	918DP	Jeffrey Alexander	Prestige
23	110SN	Elements	Vienna
29	436-128BNBDL	Jeffrey Alexander	Annadale
32	918L-NI	Jeffrey Alexander	Prestige
42	G130L-DBAC	Jeffrey Alexander	Harlow
62	527-160NI	Jeffrey Alexander	Bremen
87	749DACM	Jeffrey Alexander	Tuscany
116	MO6373-12SN	Jeffrey Alexander	Belcastel
118	988-3PC	Elements	Zachary
135	382-128DBAC	Elements	Cosgrove
153	G140L-PC	Jeffrey Alexander	Harlow
155	993BNBDL	Elements	Aiden
156	183-3BNBDL	Elements	Park
192	80509SN	Jeffrey Alexander	Valencia
208	Z259-3BC	Elements	Madison
220	S6060BNBDL	Jeffrey Alexander	Rhodes
247	3970-DACM	Elements	Gatsby
248	972-96BNBDL	Jeffrey Alexander	Marlo
255	737-128BNBDL	Jeffrey Alexander	Chesapeake
259	737-96SIM	Jeffrey Alexander	Chesapeake

如有多个条件，可使用&、|和~来连接。需要注意运算优先级。

In [25]:

```
cond1 = (comp['Asset']>50000) & (comp['Profit']>0)
comp.loc[cond1]
```

Out[25]:

	FirmID	Ind	Asset	Liability	Equity	Profit
1	C10002	Retail	61712	54600	7652	12,342.00
3	C10004	Agri	98646	38581	60065	3,945.00
5	C10006	Manu	69988	35518	34470	27,995.00
6	C10007	Agri	81961	28004	53957	11,474.00

9	C10010	Serv	83890	25393	58497	41,106.00
10	C10011	Serv	75813	24772	51041	34,873.00
11	C10012	Serv	59817	NaN	59717	11,963.00
17	C10018	Retail	94853	69242	25611	9,485.00
23	C10024	Manu	74972	38239	36733	2,401.00
24	C10025	Manu	84943	74130	10813	5,285.00
26	C10027	Serv	70721	55613	15108	10,392.00
27	C10028	Agri	66488	26271	40217	7,998.00
30	C10031	Serv	61563	46723	14840	5,913.00
31	C10032	Agri	80025	11268	68757	9,683.00
32	C10033	Manu	101371	83598	17773	23,713.00
34	C10035	Agri	81025	16326	64699	33,660.00
38	C10039	Serv	59389	14080	45309	5,845.00
40	C10041	Manu	76345	7922	68423	22,254.00
41	C10042	Agri	58968	15079	44000	21,024.00
46	C10047	Retail	79586	66845	12741	41,417.00
47	C10048	Manu	89226	63154	26072	53,213.00
49	C10050	Serv	100325	59204	41121	56,384.00
53	C10054	Agri	50102	35665	14437	26,124.00
54	C10055	Agri	56611	22186	34425	14,377.00
63	C10064	Agri	102293	77239	25054	26,507.00
64	C10065	Manu	85147	28448	56699	7,764.00
66	C10067	Serv	86837	20799	66038	22,555.00
70	C10071	Retail	59301	5266	54035	5,522.00
71	C10072	Agri	53670	34863	18807	15,710.00
74	C10075	Serv	89072	3632	85440	34,110.00
75	C10076	Agri	93483	77904	15579	99,401.00
78	C10079	Serv	52948	46241	6707	14,208.00
79	C10080	Manu	83947	54703	29244	23,719.00
82	C10083	Manu	54319	18804	35515	10,079.00
84	C10085	Serv	98212	39449	58763	59,179.00
86	C10087	Retail	50864	31455	19409	23,755.00
87	C10088	Retail	50591	35081	15510	12,383.00
88	C10089	Agri	79141	17963	61178	28,548.00
96	C10097	Retail	106597	91349	15248	29,978.00
97	C10098	Manu	80736	25277	55459	46,081.00

二、更改DataFrame中的数据

创建新变量 (列)

在Pandas中，我们可以把一列当成一个变量一样来对待。因此，创建一个新列可用赋值语句实现，无需使用循环逐个实现。

In [3]:

```
comp['Listed'] = 'No'
# comp['new_profit'] = comp['Profit'] / 10
comp['ROA'] = comp['Profit'] / comp['Asset']
comp.head(10)
```

Out[3]:

	FirmID	Ind	Asset	Liability	Equity	Profit	Listed	ROA
0	C10001	Agri	32578	23483	9095	-2,607.00	No	-0.08
1	C10002	Retail	61712	54600	7652	12,342.00	No	0.20
2	C10003	Manu	73379	53702	19677	-14,676.00	No	-0.20
3	C10004	Agri	98646	38581	60065	3,945.00	No	0.04
4	C10005	Agri	85622	46621	39001	-2,569.00	No	-0.03
5	C10006	Manu	69988	35518	34470	27,995.00	No	0.40
6	C10007	Agri	81961	28004	53957	11,474.00	No	0.14
7	C10008	Serv	62616	13522	48993	-6,262.00	No	-0.10
8	C10009	Manu	53617	NaN	$53617	NaN	No	NaN
9	C10010	Serv	83890	25393	58497	41,106.00	No	0.49

【练习4-3】在sales中新建如下列:revenue=list_price X quantity_sold；cogs=cost X quantity_sold；gross_profit。

In [4]:

```
sales = pd.read_excel(folder/'sales.xlsx')

sales['revenue'] = sales['list_price'] * sales['quantity_sold']
sales['cogs'] = sales['cost'] * sales['quantity_sold']
sales['gross_profit'] = sales['revenue'] - sales['cogs']
sales.head(20)
```

Out[4]:

	customer_number	region	date_of_sale	item	brand	collection	description
0	20943	Midwest	2015-01-01	918DP	Jeffrey Alexander	Prestige	Knob
1	126101	Northwest	2015-01-01	2981AB	Elements	Florence	3" pul
2	161675	West	2015-01-01	910-128PC	Jeffrey Alexander	Modena	128 mm CC pul
3	175749	West	2015-01-01	351-128PC	Elements	Calloway	128" CC pul
4	216582	West	2015-01-01	S271-3PB	Elements	Torino	3" CC pul
5	272896	Northeast	2015-01-01	293-160PC	Jeffrey Alexander	Zane	160 mm CC pul
6	350910	East coast	2015-01-01	972DBAC	Jeffrey Alexander	Marlo	Knob
7	394659	East coast	2015-01-01	976-128PC	Elements	Belfast	128" CC pul
8	405112	Central	2015-01-01	286-192DBAC	Jeffrey Alexander	Leyton	192 mm CC pul
9	419198	West	2015-01-01	264-192BNBDL	Jeffrey Alexander	Alvar	192 mm CC pul
10	436082	South	2015-01-01	B812-SN	Jeffrey Alexander	Backplates	Knob Backplate
11	461385	Central	2015-01-01	193-128BC	Elements	Asher	128" CC pul
12	469194	Northeast	2015-01-01	918-12AEM	Jeffrey Alexander	Prestige	12" CC app. Pul
13	505041	East coast	2015-01-01	Z280-ABSB	Jeffrey Alexander	Cordova	96 mm CC pul
14	527133	Northeast	2015-01-01	286-160DBAC	Jeffrey Alexander	Leyton	160 mm CC pul
15	536445	South	2015-01-01	737-128BNBDL	Jeffrey Alexander	Chesapeake	128 mm CC pul
16	550847	Northeast	2015-01-01	749-96B-DBAC	Jeffrey Alexander	Tuscany	96 mm CC pul
17	585268	West	2015-01-01	527-160SIM	Jeffrey Alexander	Bremen	1 160 mm CC pul
18	604530	South	2015-01-01	531-96ABSB	Jeffrey Alexander	Kensington	96 mm CC pul
19	620461	East coast	2015-01-01	527-128DACM	Jeffrey Alexander	Bremen	1 128 mm CC pul

更改已有数据

直接使用赋值语句，即可使新的值覆盖掉原来的值，实现数据更新。

In [34]:

```
comp['Profit'] = comp['Profit'] / 10
comp.head(10)
```

Out[34]:

	FirmID	Ind	Asset	Liability	Equity	Profit	Listed	new_profit	ROA
0	C10001	Agri	32578	23483	9095	-2,607.00	No	-260.70	-0.08
1	C10002	Retail	61712	54600	7652	12,342.00	No	1,234.20	0.20
2	C10003	Manu	73379	53702	19677	-14,676.00	No	-1,467.60	-0.20
3	C10004	Agri	98646	38581	60065	3,945.00	No	394.50	0.04
4	C10005	Agri	85622	46621	39001	-2,569.00	No	-256.90	-0.03
5	C10006	Manu	69988	35518	34470	27,995.00	No	2,799.50	0.40
6	C10007	Agri	81961	28004	53957	11,474.00	No	1,147.40	0.14
7	C10008	Serv	62616	13522	48993	-6,262.00	No	-626.20	-0.10
8	C10009	Manu	53617	NaN	$53617	NaN	No	NaN	NaN
9	C10010	Serv	83890	25393	58497	41,106.00	No	4,110.60	0.49

使用.loc[]选取或筛选后再更改

In [7]:

```
comp.loc[comp['Asset']>50000,'Listed'] = 'Yes'
comp.head(20)
```

Out[7]:

	FirmID	Ind	Asset	Liability	Equity	Profit	Listed	ROA
0	C10001	Agri	32578	23483	9095	-2,607.00	No	-0.08
1	C10002	Retail	61712	54600	7652	12,342.00	Yes	0.20
2	C10003	Manu	73379	53702	19677	-14,676.00	Yes	-0.20
3	C10004	Agri	98646	38581	60065	3,945.00	Yes	0.04
4	C10005	Agri	85622	46621	39001	-2,569.00	Yes	-0.03
5	C10006	Manu	69988	35518	34470	27,995.00	Yes	0.40
6	C10007	Agri	81961	28004	53957	11,474.00	Yes	0.14
7	C10008	Serv	62616	13522	48993	-6,262.00	Yes	-0.10
8	C10009	Manu	53617	0	$53617	0.00	Yes	0.00
9	C10010	Serv	83890	25393	58497	41,106.00	Yes	0.49
10	C10011	Serv	75813	24772	51041	34,873.00	Yes	0.46
11	C10012	Serv	59817	0	59717	11,963.00	Yes	0.20
12	C10013	Manu	5968	$4834	1134	2,924.00	No	0.49
13	C10014	Retail	9847	4234	5613	1,575.00	No	0.16
14	C10015	Manu	49500	18810	30690	22,770.00	No	0.46
15	C10016	Retail	33981	18009	15081	7,475.00	No	0.22
16	C10017	Manu	47501	13775	33726	475.00	No	0.01
17	C10018	Retail	94853	69242	25611	9,485.00	Yes	0.10
18	C10019	Manu	7094	993	6101	2,979.00	No	0.42
19	C10020	Retail	49582	16857	32725	22,311.00	No	0.45

【练习4-4】查看下sales中有几个地区（region列），其中有没有错误。若有，将其修改为正确值。

In [40]:

```
sales['region'].unique()
```

Out[40]:

```
array(['Midwest', 'Northwest', 'West', 'Northeast', 'East coast',
       'Central', 'South', 'International', 'Centrall', 'Soouth'],
      dtype=object)
```

In [5]:

```
#方法一、筛选满足条件的记录后，赋值更改
sales.loc[sales['region'] == 'Centrall','region'] = 'Central'
sales.loc[sales['region'] == 'Soouth','region'] = 'South'
sales['region'].unique()
```

Out[5]:

```
array(['Midwest', 'Northwest', 'West', 'Northeast', 'East coast',
       'Central', 'South', 'International'], dtype=object)
```

In [6]:

```
#方法二、选取整列，进行replace替换后赋值到相应的列。
sales['region'] = sales['region'].replace({'Centrall':'Central','Soouth':'South'})
sales['region'].unique()
```

Out[6]:

```
array(['Midwest', 'Northwest', 'West', 'Northeast', 'East coast',
       'Central', 'South', 'International'], dtype=object)
```

排序

In [16]:

```
comp.head(20)
```

Out[16]:

	FirmID	Ind	Asset	Liability	Equity	Profit
96	C10097	Retail	106597	91349	15248	29,978.00
76	C10077	Manu	105229	82287	22942	-31,174.00
36	C10037	Serv	104225	49808	54417	-7,069.00
63	C10064	Agri	102293	77239	25054	26,507.00
32	C10033	Manu	101371	83598	17773	23,713.00
49	C10050	Serv	100325	59204	41121	56,384.00
3	C10004	Agri	98646	38581	60065	3,945.00
84	C10085	Serv	98212	39449	58763	59,179.00
22	C10023	Manu	97443	80619	16824	-16,447.00
17	C10018	Retail	94853	69242	25611	9,485.00
75	C10076	Agri	93483	77904	15579	99,401.00
98	C10099	Serv	92617	45564	47053	-1,491.00
45	C10046	Manu	90713	23081	67632	-7,190.00
95	C10096	Agri	89623	1627	87996	-7,505.00
47	C10048	Manu	89226	63154	26072	53,213.00
74	C10075	Serv	89072	3632	85440	34,110.00
66	C10067	Serv	86837	20799	66038	22,555.00
4	C10005	Agri	85622	46621	39001	-2,569.00
64	C10065	Manu	85147	28448	56699	7,764.00
24	C10025	Manu	84943	74130	10813	5,285.00

In [12]:

```
# 单变量排序
comp = comp.sort_values('Asset')
```

In [15]:

```
# 降序排序
comp.sort_values('Asset',ascending=False)
```

In [ ]:

```
# 在原变量中保存排序结果
comp.sort_values('Asset',ascending=False,inplace=True)
```

In [17]:

```
#多变量排序
comp.sort_values(['Ind','Profit'])
```

Out[17]:

	FirmID	Ind	Asset	Liability	Equity	Profit
29	C10030	Agri	26029	2065	23964	-12,924.00
51	C10052	Agri	69698	60627	9071	-8,721.00
95	C10096	Agri	89623	1627	87996	-7,505.00
0	C10001	Agri	32578	23483	9095	-2,607.00
4	C10005	Agri	85622	46621	39001	-2,569.00
56	C10057	Agri	13603	12219	1384	-2,226.00
61	C10062	Agri	69299	38117	31182	-1,967.00
93	C10094	Agri	27065	486	26579	2,269.00
28	C10029	Agri	37143	31337	5806	2,543.00
91	C10092	Agri	7865	6569	1296	2,950.00
20	C10021	Agri	10944	1693	9251	3,130.00
3	C10004	Agri	98646	38581	60065	3,945.00
27	C10028	Agri	66488	26271	40217	7,998.00
31	C10032	Agri	80025	11268	68757	9,683.00
6	C10007	Agri	81961	28004	53957	11,474.00
...	...	...	...	...	...	...
62	C10063	Serv	9194	3504	5427	4,710.00
38	C10039	Serv	59389	14080	45309	5,845.00
30	C10031	Serv	61563	46723	14840	5,913.00
58	C10059	Serv	49214	3367	45847	6,289.00
26	C10027	Serv	70721	55613	15108	10,392.00
94	C10095	Serv	43562	21048	22514	10,462.00
11	C10012	Serv	59817	NaN	59717	11,963.00
60	C10061	Serv	29565	17793	11772	13,291.00
78	C10079	Serv	52948	46241	6707	14,208.00
66	C10067	Serv	86837	20799	66038	22,555.00
74	C10075	Serv	89072	3632	85440	34,110.00
10	C10011	Serv	75813	24772	51041	34,873.00
9	C10010	Serv	83890	25393	58497	41,106.00
49	C10050	Serv	100325	59204	41121	56,384.00
84	C10085	Serv	98212	39449	58763	59,179.00

100 rows × 6 columns

In [19]:

```
comp.sort_values(['Ind','Profit'],ascending=[True,False])
```

Out[19]:

	FirmID	Ind	Asset	Liability	Equity	Profit
75	C10076	Agri	93483	77904	15579	99,401.00
34	C10035	Agri	81025	16326	64699	33,660.00
88	C10089	Agri	79141	17963	61178	28,548.00
63	C10064	Agri	102293	77239	25054	26,507.00
53	C10054	Agri	50102	35665	14437	26,124.00
41	C10042	Agri	58968	15079	44000	21,024.00
65	C10066	Agri	44478	17394	27084	17,133.00
71	C10072	Agri	53670	34863	18807	15,710.00
54	C10055	Agri	56611	22186	34425	14,377.00
83	C10084	Agri	22876	12866	10010	13,102.00
6	C10007	Agri	81961	28004	53957	11,474.00
31	C10032	Agri	80025	11268	68757	9,683.00
27	C10028	Agri	66488	26271	40217	7,998.00
3	C10004	Agri	98646	38581	60065	3,945.00
20	C10021	Agri	10944	1693	9251	3,130.00
...	...	...	...	...	...	...
58	C10059	Serv	49214	3367	45847	6,289.00
30	C10031	Serv	61563	46723	14840	5,913.00
38	C10039	Serv	59389	14080	45309	5,845.00
62	C10063	Serv	9194	3504	5427	4,710.00
33	C10034	Serv	9007	7351	1656	4,354.00
92	C10093	Serv	13335	7241	6094	4,132.00
48	C10049	Serv	11116	NaN	8483	1,781.00
72	C10073	Serv	14293	5079	NaN	1,659.00
85	C10086	Serv	8574	NaN	8574	-770.00
98	C10099	Serv	92617	45564	47053	-1,491.00
59	C10060	Serv	70755	55934	14821	-4,730.00
7	C10008	Serv	62616	13522	48993	-6,262.00
36	C10037	Serv	104225	49808	54417	-7,069.00
67	C10068	Serv	61951	30202	31749	-7,448.00
80	C10081	Serv	49640	611	49029	-7,604.00

100 rows × 6 columns

缺失值的处理

在现实世界中，数据往往不是完美的，常常会出现各种异常情况。例如，某些数据可能由于疏忽遗漏、处理方法不一致或无法计算等原因而缺失。此时，我们需要对这些缺失值进行相应的处理，以保证数据的质量。

Pandas有多种表示缺失值的方式，最常见的是以NaN来表示。我们可以用.isna()方法来判断缺失值的存在，并结合loc来筛选出这些记录（行）。

In [21]:

```
comp = pd.read_excel(folder/'comp_fin.xlsx')
comp.head(20)
```

Out[21]:

	FirmID	Ind	Asset	Liability	Equity	Profit
0	C10001	Agri	32578	23483	9095	-2,607.00
1	C10002	Retail	61712	54600	7652	12,342.00
2	C10003	Manu	73379	53702	19677	-14,676.00
3	C10004	Agri	98646	38581	60065	3,945.00
4	C10005	Agri	85622	46621	39001	-2,569.00
5	C10006	Manu	69988	35518	34470	27,995.00
6	C10007	Agri	81961	28004	53957	11,474.00
7	C10008	Serv	62616	13522	48993	-6,262.00
8	C10009	Manu	53617	NaN	$53617	NaN
9	C10010	Serv	83890	25393	58497	41,106.00
10	C10011	Serv	75813	24772	51041	34,873.00
11	C10012	Serv	59817	NaN	59717	11,963.00
12	C10013	Manu	5968	$4834	1134	2,924.00
13	C10014	Retail	9847	4234	5613	1,575.00
14	C10015	Manu	49500	18810	30690	22,770.00
15	C10016	Retail	33981	18009	15081	7,475.00
16	C10017	Manu	47501	13775	33726	475.00
17	C10018	Retail	94853	69242	25611	9,485.00
18	C10019	Manu	7094	993	6101	2,979.00
19	C10020	Retail	49582	16857	32725	22,311.00

In [27]:

```
comp['Liability'].notna()
```

Out[27]:

```
0      True
1      True
2      True
3      True
4      True
5      True
6      True
7      True
8     False
9      True
10     True
11    False
12     True
13     True
14     True
      ...
85    False
86     True
87     True
88     True
89     True
90     True
91     True
92     True
93     True
94     True
95     True
96     True
97     True
98     True
99     True
Name: Liability, Length: 100, dtype: bool
```

In [24]:

```
comp.loc[comp['Liability'].isna()]
```

Out[24]:

	FirmID	Ind	Asset	Liability	Equity	Profit
8	C10009	Manu	53617	NaN	$53617	NaN
11	C10012	Serv	59817	NaN	59717	11,963.00
48	C10049	Serv	11116	NaN	8483	1,781.00
85	C10086	Serv	8574	NaN	8574	-770.00

缺失值的处理需要根据其原因对症下药。最常用的处理有两种，丢弃存在缺失值的记录，或用特定的值取代缺失值。

In [33]:

```
comp
```

Out[33]:

	FirmID	Ind	Asset	Liability	Equity	Profit
0	C10001	Agri	32578	23483	9095	-2,607.00
1	C10002	Retail	61712	54600	7652	12,342.00
2	C10003	Manu	73379	53702	19677	-14,676.00
3	C10004	Agri	98646	38581	60065	3,945.00
4	C10005	Agri	85622	46621	39001	-2,569.00
5	C10006	Manu	69988	35518	34470	27,995.00
6	C10007	Agri	81961	28004	53957	11,474.00
7	C10008	Serv	62616	13522	48993	-6,262.00
8	C10009	Manu	53617	NaN	$53617	NaN
9	C10010	Serv	83890	25393	58497	41,106.00
10	C10011	Serv	75813	24772	51041	34,873.00
11	C10012	Serv	59817	NaN	59717	11,963.00
12	C10013	Manu	5968	$4834	1134	2,924.00
13	C10014	Retail	9847	4234	5613	1,575.00
14	C10015	Manu	49500	18810	30690	22,770.00
...	...	...	...	...	...	...
85	C10086	Serv	8574	NaN	8574	-770.00
86	C10087	Retail	50864	31455	19409	23,755.00
87	C10088	Retail	50591	35081	15510	12,383.00
88	C10089	Agri	79141	17963	61178	28,548.00
89	C10090	Manu	41596	21803	19793	5,513.00
90	C10091	Retail	28835	17721	11114	-2,671.00
91	C10092	Agri	7865	6569	1296	2,950.00
92	C10093	Serv	13335	7241	6094	4,132.00
93	C10094	Agri	27065	486	26579	2,269.00
94	C10095	Serv	43562	21048	22514	10,462.00
95	C10096	Agri	89623	1627	87996	-7,505.00
96	C10097	Retail	106597	91349	15248	29,978.00
97	C10098	Manu	80736	25277	55459	46,081.00
98	C10099	Serv	92617	45564	47053	-1,491.00
99	C10100	Retail	75962	50069	25893	-9,343.00

100 rows × 6 columns

In [30]:

```
# 丢弃某个变量存在缺失值的记录
comp.loc[~comp['Liability'].isna()]
comp.loc[comp['Liability'].notna()]

# 丢弃所有存在任一缺失值的记录
comp.dropna()
comp.dropna(inplace=True)
```

Out[30]:

	FirmID	Ind	Asset	Liability	Equity	Profit
0	C10001	Agri	32578	23483	9095	-2,607.00
1	C10002	Retail	61712	54600	7652	12,342.00
2	C10003	Manu	73379	53702	19677	-14,676.00
3	C10004	Agri	98646	38581	60065	3,945.00
4	C10005	Agri	85622	46621	39001	-2,569.00
5	C10006	Manu	69988	35518	34470	27,995.00
6	C10007	Agri	81961	28004	53957	11,474.00
7	C10008	Serv	62616	13522	48993	-6,262.00
9	C10010	Serv	83890	25393	58497	41,106.00
10	C10011	Serv	75813	24772	51041	34,873.00
12	C10013	Manu	5968	$4834	1134	2,924.00
13	C10014	Retail	9847	4234	5613	1,575.00
14	C10015	Manu	49500	18810	30690	22,770.00
15	C10016	Retail	33981	18009	15081	7,475.00
16	C10017	Manu	47501	13775	33726	475.00
...	...	...	...	...	...	...
84	C10085	Serv	98212	39449	58763	59,179.00
86	C10087	Retail	50864	31455	19409	23,755.00
87	C10088	Retail	50591	35081	15510	12,383.00
88	C10089	Agri	79141	17963	61178	28,548.00
89	C10090	Manu	41596	21803	19793	5,513.00
90	C10091	Retail	28835	17721	11114	-2,671.00
91	C10092	Agri	7865	6569	1296	2,950.00
92	C10093	Serv	13335	7241	6094	4,132.00
93	C10094	Agri	27065	486	26579	2,269.00
94	C10095	Serv	43562	21048	22514	10,462.00
95	C10096	Agri	89623	1627	87996	-7,505.00
96	C10097	Retail	106597	91349	15248	29,978.00
97	C10098	Manu	80736	25277	55459	46,081.00
98	C10099	Serv	92617	45564	47053	-1,491.00
99	C10100	Retail	75962	50069	25893	-9,343.00

95 rows × 6 columns

In [6]:

```
# 使用特定值填充入缺失值
comp.fillna(0,inplace=True)
# comp.fillna(method='bfill')
```

三、操作其他数据类型

处理DataFrame中的字符串数据

如何将python中处理字符串的方法，应用到DataFrame中的整列字符串数据中?

In [40]:

```
comp['Ind'].str.replace('i','x')

# 例如, upper()方法还能用吗?
comp['Ind'].str.upper()
```

Out[40]:

```
0       AGRI
1     RETAIL
2       MANU
3       AGRI
4       AGRI
5       MANU
6       AGRI
7       SERV
8       MANU
9       SERV
10      SERV
11      SERV
12      MANU
13    RETAIL
14      MANU
       ...
85      SERV
86    RETAIL
87    RETAIL
88      AGRI
89      MANU
90    RETAIL
91      AGRI
92      SERV
93      AGRI
94      SERV
95      AGRI
96    RETAIL
97      MANU
98      SERV
99    RETAIL
Name: Ind, Length: 100, dtype: object
```

处理DataFrame中的时间数据

In [3]:

```
sales = pd.read_excel(folder/'sales.xlsx')
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49839 entries, 0 to 49838
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype         
---  ------           --------------  -----         
 0   customer_number  49839 non-null  int64         
 1   region           49839 non-null  object        
 2   date_of_sale     49839 non-null  datetime64[ns]
 3   item             49839 non-null  object        
 4   brand            49839 non-null  object        
 5   collection       49839 non-null  object        
 6   description      49839 non-null  object        
 7   list_price       49839 non-null  float64       
 8   cost             49839 non-null  float64       
 9   quantity_sold    49839 non-null  int64         
dtypes: datetime64[ns](1), float64(2), int64(2), object(5)
memory usage: 3.8+ MB
```

In [44]:

```
sales['date_of_sale'].head()
```

Out[44]:

```
0   2015-01-01
1   2015-01-01
2   2015-01-01
3   2015-01-01
4   2015-01-01
Name: date_of_sale, dtype: datetime64[ns]
```

In [5]:

```
# sales['date_of_sale'].dt.month   #属性
# sales['date_of_sale'].dt.dayofweek
sales['date_of_sale'].dt.day_name('zh_CN')
```

Out[5]:

```
0         星期四
1         星期四
2         星期四
3         星期四
4         星期四
5         星期四
6         星期四
7         星期四
8         星期四
9         星期四
10        星期四
11        星期四
12        星期四
13        星期四
14        星期四
         ...
49824     星期一
49825     星期一
49826     星期一
49827     星期一
49828     星期一
49829     星期一
49830     星期一
49831     星期一
49832     星期一
49833     星期一
49834     星期一
49835     星期一
49836     星期一
49837     星期一
49838     星期一
Name: date_of_sale, Length: 49839, dtype: object
```

【练习4-5】在sales中创建两个新列，分别用于存放每笔销售的发售日期对应的年和季度。

In [8]:

```
sales['year'] = sales['date_of_sale'].dt.year
sales['quarter'] = sales['date_of_sale'].dt.quarter
```

更改列的数据类型

In [22]:

```
comp['Equity']>50000
```

Out[22]:

```
0     False
1     False
2     False
3      True
4     False
5     False
6      True
7     False
8      True
9      True
10     True
11     True
12    False
13    False
14    False
      ...
85    False
86    False
87    False
88     True
89    False
90    False
91    False
92    False
93    False
94    False
95     True
96    False
97     True
98    False
99    False
Name: Equity, Length: 100, dtype: bool
```

In [9]:

```
comp['Equity'] = comp['Equity'].astype('string').str.strip('$').astype('int')
comp['Liability'] = comp['Liability'].astype('string').str.strip('$').astype('int')
```

In [10]:

```
comp.loc[comp['Asset']!=comp['Liability']+comp['Equity']]
```

Out[10]:

	FirmID	Ind	Asset	Liability	Equity	Profit	Listed	ROA
1	C10002	Retail	61712	54600	7652	12,342.00	No	0.20
7	C10008	Serv	62616	13522	48993	-6,262.00	No	-0.10
11	C10012	Serv	59817	0	59717	11,963.00	No	0.20
15	C10016	Retail	33981	18009	15081	7,475.00	No	0.22
41	C10042	Agri	58968	15079	44000	21,024.00	No	0.36
48	C10049	Serv	11116	0	8483	1,781.00	No	0.16
62	C10063	Serv	9194	3504	5427	4,710.00	No	0.51
72	C10073	Serv	14293	5079	0	1,659.00	No	0.12

4. 分析数据

简单的统计量计算

In [24]:

```
comp['Profit'].mean()
```

Out[24]:

```
9031.09
```

In [25]:

```
comp['Ind'].value_counts(normalize=True)
```

Out[25]:

```
Manu      0.30
Serv      0.26
Agri      0.25
Retail    0.19
Name: Ind, dtype: float64
```

分组与重构

使用groupby()方法，可以对数据按某个类别进行分组统计

In [26]:

```
comp.groupby('Ind').sum()
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_1196\97387525.py:1: FutureW
arning: The default value of numeric_only in DataFrameGroupBy.sum is deprecate
d. In a future version, numeric_only will default to False. Either specify nume
ric_only or select only columns which should be valid for the function.
  comp.groupby('Ind').sum()
```

Out[26]:

Ind	Asset	Liability	Equity	Profit
Agri	1439236	656453	782894	301,059.00
Manu	1678656	873419	805237	208,414.00
Retail	976805	611559	364895	85,804.00
Serv	1468251	596930	859010	307,832.00

In [27]:

```
# comp.groupby('Ind').sum()['Asset']
comp.groupby('Ind')['Asset'].sum()
```

Out[27]:

```
Ind
Agri      1439236
Manu      1678656
Retail     976805
Serv      1468251
Name: Asset, dtype: int64
```

In [28]:

```
comp.groupby('Ind').agg({'Asset':'sum','Profit':'mean'})
```

Out[28]:

Ind	Asset	Profit
Agri	1439236	12,042.36
Manu	1678656	6,947.13
Retail	976805	4,516.00
Serv	1468251	11,839.69

数据透视表

In [10]:

```
long_comp = comp.melt(id_vars=['FirmID','Ind','Listed'], var_name='Item', value_name='Amoun
t')
long_comp.sort_values(['FirmID','Item'])
```

Out[10]:

	FirmID	Ind	Listed	Item	Amount
0	C10001	Agri	No	Asset	32,578.00
200	C10001	Agri	No	Equity	9,095.00
100	C10001	Agri	No	Liability	23,483.00
300	C10001	Agri	No	Profit	-2,607.00
400	C10001	Agri	No	ROA	-0.08
1	C10002	Retail	Yes	Asset	61,712.00
201	C10002	Retail	Yes	Equity	7,652.00
101	C10002	Retail	Yes	Liability	54,600.00
301	C10002	Retail	Yes	Profit	12,342.00
401	C10002	Retail	Yes	ROA	0.20
2	C10003	Manu	Yes	Asset	73,379.00
202	C10003	Manu	Yes	Equity	19,677.00
102	C10003	Manu	Yes	Liability	53,702.00
302	C10003	Manu	Yes	Profit	-14,676.00
402	C10003	Manu	Yes	ROA	-0.20
...	...	...	...	...	...
97	C10098	Manu	Yes	Asset	80,736.00
297	C10098	Manu	Yes	Equity	55,459.00
197	C10098	Manu	Yes	Liability	25,277.00
397	C10098	Manu	Yes	Profit	46,081.00
497	C10098	Manu	Yes	ROA	0.57
98	C10099	Serv	Yes	Asset	92,617.00
298	C10099	Serv	Yes	Equity	47,053.00
198	C10099	Serv	Yes	Liability	45,564.00
398	C10099	Serv	Yes	Profit	-1,491.00
498	C10099	Serv	Yes	ROA	-0.02
99	C10100	Retail	Yes	Asset	75,962.00
299	C10100	Retail	Yes	Equity	25,893.00
199	C10100	Retail	Yes	Liability	50,069.00
399	C10100	Retail	Yes	Profit	-9,343.00
499	C10100	Retail	Yes	ROA	-0.12

500 rows × 5 columns

In [13]:

```
long_comp.pivot_table(index=['Ind','Listed'], columns='Item', values='Amount').reset_inde
x()
```

Out[13]:

Item	Ind	Listed	Asset	Equity	Liability	Profit	ROA
0	Agri	No	24,731.22	12,718.78	12,012.44	2,596.67	0.11
1	Agri	Yes	76,040.94	41,776.56	34,271.31	17,355.56	0.23
2	Manu	No	28,409.43	16,173.71	12,235.71	4,111.71	0.15
3	Manu	Yes	80,057.75	36,175.31	43,882.44	9,428.12	0.12
4	Retail	No	28,081.38	12,845.62	15,124.38	-914.88	-0.05
5	Retail	Yes	68,377.64	23,830.00	44,596.73	8,465.73	0.11
6	Serv	No	23,750.00	15,939.60	6,599.40	3,830.40	0.22
7	Serv	Yes	76,921.94	43,725.88	33,183.50	16,845.50	0.20

In [14]:

```
long_comp.pivot_table(index=['Ind','Listed'], columns='Item', values='Amount',aggfunc='ma
x').reset_index()
```

Out[14]:

Item	Ind	Listed	Asset	Equity	Liability	Profit	ROA
0	Agri	No	44,478.00	27,084.00	31,337.00	17,133.00	0.57
1	Agri	Yes	102,293.00	87,996.00	77,904.00	99,401.00	1.06
2	Manu	No	49,500.00	41,936.00	24,879.00	22,770.00	0.65
3	Manu	Yes	105,229.00	68,423.00	83,598.00	53,213.00	0.60
4	Retail	No	49,582.00	32,725.00	25,579.00	22,311.00	0.45
5	Retail	Yes	106,597.00	54,035.00	91,349.00	41,417.00	0.52
6	Serv	No	49,640.00	49,029.00	21,048.00	13,291.00	0.51
7	Serv	Yes	104,225.00	85,440.00	59,204.00	59,179.00	0.60

5. 案例 - 毛利率分析

按“年-季度-地区”进行分组，分别计算销售收入的总额，然后按“年-季度-地区”对数据进行排序。

提示：可将计算出来的汇总数据赋值给一个变量，以便进行后续的操作。

In [15]:

```
sales.head()
```

Out[15]:

	customer_number	region	date_of_sale	item	brand	collection	description	list_
0	20943	Midwest	2015-01-01	918DP	Jeffrey Alexander	Prestige	Knob	
1	126101	Northwest	2015-01-01	2981AB	Elements	Florence	3" pull	
2	161675	West	2015-01-01	910-128PC	Jeffrey Alexander	Modena	128 mm CC pull	
3	175749	West	2015-01-01	351-128PC	Elements	Calloway	128" CC pull	
4	216582	West	2015-01-01	S271-3PB	Elements	Torino	3" CC pull	

In [18]:

```
sales.groupby(['year','quarter','region'])['revenue'].sum().reset_index().sort_values(['yea
r','quarter','revenue'])
```

Out[18]:

	year	quarter	region	revenue
2	2015	1	International	324,699.81
4	2015	1	Northeast	2,085,487.74
5	2015	1	Northwest	2,735,072.80
0	2015	1	Central	2,798,230.82
6	2015	1	South	2,993,980.51
3	2015	1	Midwest	3,420,119.16
7	2015	1	West	3,633,451.50
1	2015	1	East coast	4,690,450.27
10	2015	2	International	435,968.85
12	2015	2	Northeast	2,373,429.15
13	2015	2	Northwest	2,453,924.01
8	2015	2	Central	2,681,816.27
14	2015	2	South	3,176,539.59
11	2015	2	Midwest	3,393,051.18
15	2015	2	West	4,217,226.16
...	...	...	...	...
116	2018	3	Northeast	3,303,945.76
118	2018	3	South	3,843,769.75
112	2018	3	Central	4,513,586.76
117	2018	3	Northwest	4,744,627.22
119	2018	3	West	5,456,515.82
113	2018	3	East coast	6,276,347.34
115	2018	3	Midwest	8,822,018.68
122	2018	4	International	982,306.63
124	2018	4	Northeast	3,596,518.58
126	2018	4	South	3,695,549.07
125	2018	4	Northwest	4,645,175.19
120	2018	4	Central	5,556,415.25
127	2018	4	West	7,018,461.42
121	2018	4	East coast	7,493,010.82
123	2018	4	Midwest	8,608,804.43

128 rows × 4 columns

按“年-季度”为行，“地区”为列的方式，统计销售收入总额

In [20]:

```
sales.pivot_table(index=['year','quarter'], columns='region', values='revenue', aggfunc='sum')
```

Out[20]:

	region	Central	East coast	International	Midwest	Northeast	Northwest
year	quarter						
2015	1	2,798,230.82	4,690,450.27	324,699.81	3,420,119.16	2,085,487.74	2,735,072.80
	2	2,681,816.27	5,227,417.05	435,968.85	3,393,051.18	2,373,429.15	2,453,924.01
	3	2,975,716.28	3,289,198.96	384,504.73	3,251,758.45	1,836,868.32	2,727,339.17
	4	3,223,796.39	3,742,539.10	566,451.14	3,292,546.61	2,053,945.80	2,680,108.17
2016	1	3,736,551.69	4,776,722.97	315,343.04	4,660,718.52	2,076,105.00	2,937,894.12
	2	3,615,980.18	4,436,181.52	681,789.10	4,685,471.29	2,784,704.72	2,839,150.20
	3	3,730,122.97	4,798,090.13	525,265.71	5,580,893.78	2,108,440.21	3,153,580.22
	4	3,523,404.41	4,697,543.11	419,298.12	4,244,249.58	1,875,404.29	3,579,267.21
2017	1	4,248,671.46	5,776,663.62	502,555.95	4,702,470.08	2,526,349.49	3,605,542.28
	2	4,277,325.74	5,915,181.05	494,920.74	6,663,721.05	2,621,056.94	3,662,668.84
	3	3,799,040.87	6,663,087.80	290,288.31	6,603,027.46	2,571,784.28	2,910,208.39
	4	4,373,792.16	5,550,543.35	608,266.26	5,724,446.06	3,386,849.79	3,044,412.36
2018	1	4,376,163.82	5,850,251.16	918,821.09	6,999,015.14	2,835,201.39	3,923,003.92
	2	5,282,296.22	6,100,470.12	904,448.77	8,218,205.99	3,446,716.74	4,081,896.08
	3	4,513,586.76	6,276,347.34	709,733.81	8,822,018.68	3,303,945.76	4,744,627.22
	4	5,556,415.25	7,493,010.82	982,306.63	8,608,804.43	3,596,518.58	4,645,175.19

按照“品牌-系列-地区”（brand-collection-region）进行分组，按分组统计销售产品成本的平均值

In [23]:

```
sales.groupby(['brand','collection','region'])['cogs'].mean().reset_index()
```

Out[23]:

	brand	collection	region	cogs
0	Elements	Aiden	Central	1,839.64
1	Elements	Aiden	East coast	1,848.64
2	Elements	Aiden	International	2,548.93
3	Elements	Aiden	Midwest	3,151.35
4	Elements	Aiden	Northeast	1,851.06
5	Elements	Aiden	Northwest	2,169.58
6	Elements	Aiden	South	1,849.29
7	Elements	Aiden	West	1,965.99
8	Elements	Arcadia	Central	1,619.11
9	Elements	Arcadia	East coast	1,558.44
10	Elements	Arcadia	International	1,278.65
11	Elements	Arcadia	Midwest	2,777.13
12	Elements	Arcadia	Northeast	1,551.15
13	Elements	Arcadia	Northwest	1,626.05
14	Elements	Arcadia	South	1,406.02
...	...	...	...	...
839	Jeffrey Alexander	Zane	East coast	6,944.31
840	Jeffrey Alexander	Zane	International	5,593.34
841	Jeffrey Alexander	Zane	Midwest	13,260.44
842	Jeffrey Alexander	Zane	Northeast	4,766.82
843	Jeffrey Alexander	Zane	Northwest	6,312.33
844	Jeffrey Alexander	Zane	South	5,578.39
845	Jeffrey Alexander	Zane	West	6,890.30
846	Jeffrey Alexander	Zurich	Central	3,517.35
847	Jeffrey Alexander	Zurich	East coast	3,439.05
848	Jeffrey Alexander	Zurich	International	3,536.48
849	Jeffrey Alexander	Zurich	Midwest	6,600.87
850	Jeffrey Alexander	Zurich	Northeast	3,739.22
851	Jeffrey Alexander	Zurich	Northwest	3,346.59
852	Jeffrey Alexander	Zurich	South	2,495.18
853	Jeffrey Alexander	Zurich	West	3,215.90

854 rows × 4 columns

按“品牌-系列”为行，“地区”为列的方式，统计每笔销售成本的平均值

In [25]:

```
sales.pivot_table(index=['brand','collection'], columns='region', values='cogs', aggfunc='mean').reset_index()
```

Out[25]:

region	brand	collection	Central	East coast	International	Midwest	Northeast	Northw
0	Elements	Aiden	1,839.64	1,848.64	2,548.93	3,151.35	1,851.06	2,169
1	Elements	Arcadia	1,619.11	1,558.44	1,278.65	2,777.13	1,551.15	1,626
2	Elements	Asher	2,544.51	2,484.00	2,760.31	4,058.85	2,600.12	2,381
3	Elements	Belfast	2,377.06	2,394.24	3,400.61	4,158.21	2,648.91	2,726
4	Elements	Brenton	2,554.96	2,318.18	2,935.00	4,197.37	2,847.05	2,583
5	Elements	Calloway	1,871.22	1,667.44	2,217.93	3,117.17	1,892.45	1,805
6	Elements	Capri	1,574.08	1,761.22	1,434.28	2,510.71	1,595.16	1,265
7	Elements	Cosgrove	2,948.13	2,648.18	3,579.34	3,913.66	2,308.64	2,404
8	Elements	Cypress	998.23	1,024.96	828.39	1,838.95	980.15	760
9	Elements	Drake	1,760.98	1,677.86	2,217.03	3,074.58	2,010.11	1,368
10	Elements	Edgefield	3,286.51	3,222.42	3,071.71	3,889.97	3,396.13	3,965
11	Elements	Florence	1,938.45	2,197.19	2,064.65	3,473.29	2,014.30	1,999
12	Elements	Gatsby	951.22	1,006.56	1,200.06	1,605.79	923.47	979
13	Elements	Geneva	1,397.05	1,165.29	762.49	2,492.79	1,389.17	1,113
14	Elements	Glendale	1,703.87	1,863.53	1,735.45	2,485.18	1,698.76	1,957
...	...	...	...	...	...	...	...	
92	Jeffrey Alexander	Rhodes	14,218.84	9,263.79	10,242.47	7,296.03	16,034.12	12,528
93	Jeffrey Alexander	Rochester	4,116.18	3,938.90	3,530.84	6,825.05	3,032.07	4,987
94	Jeffrey Alexander	Royce	3,403.38	3,233.04	4,691.44	5,108.02	3,582.52	3,611
95	Jeffrey Alexander	Solana	5,297.31	5,048.24	6,469.83	9,215.87	4,394.54	5,800
96	Jeffrey Alexander	Sonoma	4,702.53	9,242.02	6,771.21	9,960.45	8,571.79	9,591
97	Jeffrey Alexander	Sutton	5,342.06	4,791.73	3,700.95	8,906.43	5,300.97	4,219
98	Jeffrey Alexander	Symphony	2,720.05	3,146.13	1,854.03	5,470.02	2,866.67	2,956
99	Jeffrey Alexander	Tahoe	14,575.58	14,119.67	6,252.30	27,232.54	9,843.50	18,960

100	Jeffrey Alexander	Tiffany	10,878.49	15,916.22	5,768.78	16,159.51	11,973.35	12,831
101	Jeffrey Alexander	Tuscany	3,296.43	3,294.90	4,538.26	5,557.00	4,053.82	3,948
102	Jeffrey Alexander	Valencia	2,840.27	2,528.52	1,304.61	4,414.68	2,672.45	2,616
103	Jeffrey Alexander	Venezia	2,935.36	4,381.74	4,535.32	6,171.50	3,424.71	3,808
104	Jeffrey Alexander	West	5,947.29	5,943.01	3,222.80	10,623.41	8,860.82	7,240
105	Jeffrey Alexander	Zane	9,327.40	6,944.31	5,593.34	13,260.44	4,766.82	6,312
106	Jeffrey Alexander	Zurich	3,517.35	3,439.05	3,536.48	6,600.87	3,739.22	3,346

107 rows × 10 columns

统计每一年各品牌的毛利润总额，并按照其值降序排序。

In [26]:

```
sales.groupby(['year','brand'])['gross_profit'].sum().reset_index().sort_values(['year','gross_profit'],ascending=[True,False])
```

Out[26]:

	year	brand	gross_profit
1	2015	Jeffrey Alexander	30,436,333.51
0	2015	Elements	5,035,848.50
3	2016	Jeffrey Alexander	37,065,394.25
2	2016	Elements	6,515,517.46
5	2017	Jeffrey Alexander	42,388,994.88
4	2017	Elements	7,178,812.53
7	2018	Jeffrey Alexander	50,927,391.87
6	2018	Elements	9,438,793.06

2018年的销售数据中，每个品牌的哪个系列毛利率最大，哪个系列又最小?

In [28]:

```
sales.head(20)
```

Out[28]:

	customer_number	region	date_of_sale	item	brand	collection	description
0	20943	Midwest	2015-01-01	918DP	Jeffrey Alexander	Prestige	Knob
1	126101	Northwest	2015-01-01	2981AB	Elements	Florence	3" pul
2	161675	West	2015-01-01	910-128PC	Jeffrey Alexander	Modena	128 mm CC pul
3	175749	West	2015-01-01	351-128PC	Elements	Calloway	128" CC pul
4	216582	West	2015-01-01	S271-3PB	Elements	Torino	3" CC pul
5	272896	Northeast	2015-01-01	293-160PC	Jeffrey Alexander	Zane	160 mm CC pul
6	350910	East coast	2015-01-01	972DBAC	Jeffrey Alexander	Marlo	Knob
7	394659	East coast	2015-01-01	976-128PC	Elements	Belfast	128" CC pul
8	405112	Central	2015-01-01	286-192DBAC	Jeffrey Alexander	Leyton	192 mm CC pul
9	419198	West	2015-01-01	264-192BNBDL	Jeffrey Alexander	Alvar	192 mm CC pul
10	436082	South	2015-01-01	B812-SN	Jeffrey Alexander	Backplates	Knob Backplate
11	461385	Central	2015-01-01	193-128BC	Elements	Asher	128" CC pul
12	469194	Northeast	2015-01-01	918-12AEM	Jeffrey Alexander	Prestige	12" CC app. Pul
13	505041	East coast	2015-01-01	Z280-ABSB	Jeffrey Alexander	Cordova	96 mm CC pul
14	527133	Northeast	2015-01-01	286-160DBAC	Jeffrey Alexander	Leyton	160 mm CC pul
15	536445	South	2015-01-01	737-128BNBDL	Jeffrey Alexander	Chesapeake	128 mm CC pul
16	550847	Northeast	2015-01-01	749-96B-DBAC	Jeffrey Alexander	Tuscany	96 mm CC pul
17	585268	West	2015-01-01	527-160SIM	Jeffrey Alexander	Bremen	1 160 mm CC pul
18	604530	South	2015-01-01	531-96ABSB	Jeffrey Alexander	Kensington	96 mm CC pul
19	620461	East coast	2015-01-01	527-128DACM	Jeffrey Alexander	Bremen	1 128 mm CC pul

In [36]:

```
sales_collection = sales.loc[sales['year']==2018].groupby(['brand','collection'])[['revenu
e','gross_profit']].sum().reset_index()
sales_collection['gpratio'] = sales_collection['gross_profit']/sales_collection['revenue']
sales_collection.sort_values(['brand','gpratio'], inplace=True)
sales_collection.groupby('brand').tail(2)
```

Out[36]:

	brand	collection	revenue	gross_profit	gpratio
31	Elements	Strickland	621,411.66	229,127.81	0.37
29	Elements	Somerset	390,249.85	144,019.59	0.37
72	Jeffrey Alexander	Key West	21,306.67	9,052.37	0.42
68	Jeffrey Alexander	Hudson	1,634,714.17	699,920.18	0.43