Pandas 和 SQL 比较

导入 pandas 和 Numpy

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

把数据读出，并写入 csv 中，同时导入 Mysql 数据库

```
def pandas_sql_url():
    url = ('https://raw.githubusercontent.com/pandas-dev/pandas/master/pandas/tests/io/data/csv/tips.csv')
    tip = pd.read_csv(url)
    print(tip.head())
    tip.to_csv(r'E:\python_data\20200916.csv')
```

控制台输出如下：

```
F:\2020Python\python_code\Scripts\python.exe C:/Users/Administrator/PycharmProjects/pythonProject/test1.py
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4

Process finished with exit code 0
```

Csv 文件如下：

	A	B	C	D	E	F	G	H	I
		total_bil	tip	sex	smoker	day	time	size	
	0	16.99	1.01	Female	No	Sun	Dinner	2	
	1	10.34	1.66	Male	No	Sun	Dinner	3	
	2	21.01	3.5	Male	No	Sun	Dinner	3	
	3	23.68	3.31	Male	No	Sun	Dinner	2	
	4	24.59	3.61	Female	No	Sun	Dinner	4	
	5	25.29	4.71	Male	No	Sun	Dinner	4	
	6	8.77	2	Male	No	Sun	Dinner	2	
	7	26.88	3.12	Male	No	Sun	Dinner	4	
	8	15.04	1.96	Male	No	Sun	Dinner	2	
	9	14.78	3.23	Male	No	Sun	Dinner	2	
	10	10.27	1.71	Male	No	Sun	Dinner	2	

数据库如下：

```sql
SELECT total_bill, tip, smoker, time
FROM tips
LIMIT 5;
```

信息 | 结果 1 | 剖析 | 状态

total_bill	tip	smoker	time
16.99	1.01	No	Dinner
10.34	1.66	No	Dinner
21.01	3.5	No	Dinner
23.68	3.31	No	Dinner
24.59	3.61	No	Dinner

SELECT

(1) 显示指定列的前 5 行

SQL

```sql
SELECT total_bill, tip, smoker, time
FROM tips
LIMIT 5;
```

信息 | 结果 1 | 剖析 | 状态

total_bill	tip	smoker	time
16.99	1.01	No	Dinner
10.34	1.66	No	Dinner
21.01	3.5	No	Dinner
23.68	3.31	No	Dinner
24.59	3.61	No	Dinner

Pandas

```python
def pandas_sql1(table1):
    """
    SELECT total_bill, tip, smoker, time
    FROM tips
    LIMIT 5;
    """
    print(table1[['total_bill', 'tip', 'smoker', 'time']].head(5))
    # print(table1.head(5))
```

```
   total_bill   tip smoker    time
0       16.99  1.01     No  Dinner
1       10.34  1.66     No  Dinner
2       21.01  3.50     No  Dinner
3       23.68  3.31     No  Dinner
4       24.59  3.61     No  Dinner

Process finished with exit code 0
```

(2) 在 SQL 中增加一个计算列，显示小费的费率

```sql
SELECT *, tip/total_bill as tip_rate
FROM tips
LIMIT 5;
```

信息 | 结果 1 | 剖析 | 状态

total_bill	tip	sex	smoker	day	time	size	tip_rate
16.99	1.01	Female	No	Sun	Dinner	2	0.05944673337257211
10.34	1.66	Male	No	Sun	Dinner	3	0.16054158607350097
21.01	3.5	Male	No	Sun	Dinner	3	0.16658733936220846
23.68	3.31	Male	No	Sun	Dinner	2	0.1397804054054054
24.59	3.61	Female	No	Sun	Dinner	4	0.14680764538430255

Pandas 代码如下：

```python
def pandas_sql2(table1):
    """
    SELECT *, tip/total_bill as tip_rate
    FROM tips
    LIMIT 5;
    :param table1:
    :return:
    """
    print(table1.assign(tip_rate=table1['tip'] / table1['total_bill']).head(5))
```

```
def pandas_sql():
    # tips = pd.read_table(r'E:\python_data\tips.csv', sep=',')
    tips = pd.read_table(r'E:\python_data\20200916.csv', sep=',')
    # pandas_sql_url()
    pandas_sql2(tips)
```

运行结果

```
   ID  total_bill   tip     sex smoker  day    time  size  tip_rate
0   0       16.99  1.01  Female     No  Sun  Dinner     2  0.059447
1   1       10.34  1.66    Male     No  Sun  Dinner     3  0.160542
2   2       21.01  3.50    Male     No  Sun  Dinner     3  0.166587
3   3       23.68  3.31    Male     No  Sun  Dinner     2  0.139780
4   4       24.59  3.61  Female     No  Sun  Dinner     4  0.146808

Process finished with exit code 0
```

显示指定的列：

```
def pandas_sql2(table1):
    """
    SELECT *, tip/total_bill as tip_rate
    FROM tips
    LIMIT 5;
    :param table1:
    :return:
    """
    print(table1.assign(tip_rate=table1['tip'] / table1['total_bill'])[['ID','tip','total_bill']].head(5))
```

WHERE

(1) SQL 过滤采用 Where 关键字

```sql
SELECT *
FROM tips
WHERE time = 'Dinner'
LIMIT 5;
```

信息 | 结果 1 | 剖析 | 状态

total_bill	tip	sex	smoker	day	time	size
16.99	1.01	Female	No	Sun	Dinner	2
10.34	1.66	Male	No	Sun	Dinner	3
21.01	3.5	Male	No	Sun	Dinner	3
23.68	3.31	Male	No	Sun	Dinner	2
24.59	3.61	Female	No	Sun	Dinner	4

Python 代码如下：

```python
def pandas_sql3(table1):
    """
    SELECT *
    FROM tips
    WHERE time = 'Dinner'
    LIMIT 5;
    :param table1:
    :return:
    """
    return table1[table1['time']=='Dinner'].head(5)
```

```python
tips = pd.read_table(r'E:\python_data\20200916.csv', sep=',')
# pandas_sql_url()
print(pandas_sql3(tips))
```

运行结果

```
   ID  total_bill   tip     sex smoker  day    time  size
0   0       16.99  1.01  Female     No  Sun  Dinner     2
1   1       10.34  1.66    Male     No  Sun  Dinner     3
2   2       21.01  3.50    Male     No  Sun  Dinner     3
3   3       23.68  3.31    Male     No  Sun  Dinner     2
4   4       24.59  3.61  Female     No  Sun  Dinner     4

Process finished with exit code 0
```

(2) SQL 中用到 OR 或者 AND ，多条件查询，在 pandas 中采用 | (OR) ，&(And)

```sql
SELECT *
FROM tips
WHERE time = 'Dinner' AND tip > 5.00;
```

信息 | 结果 1 | 剖析 | 状态

total_bill	tip	sex	smoker	day	time	size
39.42	7.58	Male	No	Sat	Dinner	4
30.4	5.6	Male	No	Sun	Dinner	4
32.4	6.0	Male	No	Sun	Dinner	4
34.81	5.2	Female	No	Sun	Dinner	4
48.27	6.73	Male	No	Sat	Dinner	4
29.93	5.07	Male	No	Sun	Dinner	4
29.85	5.14	Female	No	Sun	Dinner	5
50.81	10.0	Male	Yes	Sat	Dinner	3
7.25	5.15	Male	Yes	Sun	Dinner	2
23.33	5.65	Male	Yes	Sun	Dinner	2
23.17	6.5	Male	Yes	Sun	Dinner	4
25.89	5.16	Male	Yes	Sat	Dinner	4
48.33	9.0	Male	No	Sat	Dinner	4
28.17	6.5	Female	Yes	Sat	Dinner	3
29.03	5.92	Male	No	Sat	Dinner	3

Pandas 代码如下：

```python
def pandas_sql4(table1):
    """
    SELECT *
    FROM tips
    WHERE time = 'Dinner' AND tip > 5.00;
    :param table1:
    :return:
    """
    return table1[(table1['time']=='Dinner')&(table1['tip']>5.00)]
```

```python
tips = pd.read_table(r'E:\python_data\20200916.csv', sep=',')
# pandas_sql_url()
print(pandas_sql4(tips))
```

运行结果如下:

	ID	total_bill	tip	sex	smoker	day	time	size
23	23	39.42	7.58	Male	No	Sat	Dinner	4
44	44	30.40	5.60	Male	No	Sun	Dinner	4
47	47	32.40	6.00	Male	No	Sun	Dinner	4
52	52	34.81	5.20	Female	No	Sun	Dinner	4
59	59	48.27	6.73	Male	No	Sat	Dinner	4
116	116	29.93	5.07	Male	No	Sun	Dinner	4
155	155	29.85	5.14	Female	No	Sun	Dinner	5
170	170	50.81	10.00	Male	Yes	Sat	Dinner	3
172	172	7.25	5.15	Male	Yes	Sun	Dinner	2
181	181	23.33	5.65	Male	Yes	Sun	Dinner	2
183	183	23.17	6.50	Male	Yes	Sun	Dinner	4
211	211	25.89	5.16	Male	Yes	Sat	Dinner	4

(3) SQL 中使用了'OR'

```sql
SELECT *
FROM tips
WHERE size >= 5 OR total_bill > 45;
```

信息 | 结果 1 | 剖析 | 状态

total_bill	tip	sex	smoker	day	time	size
48.27	6.73	Male	No	Sat	Dinner	4
29.8	4.2	Female	No	Thur	Lunch	6
34.3	6.7	Male	No	Thur	Lunch	6
41.19	5.0	Male	No	Thur	Lunch	5
27.05	5.0	Female	No	Thur	Lunch	6
29.85	5.14	Female	No	Sun	Dinner	5
48.17	5.0	Male	No	Sun	Dinner	6
50.81	10.0	Male	Yes	Sat	Dinner	3
45.35	3.5	Male	Yes	Sun	Dinner	3
20.69	5.0	Male	No	Sun	Dinner	5
30.46	2.0	Male	Yes	Sun	Dinner	5
48.33	9.0	Male	No	Sat	Dinner	4
28.15	3.0	Male	Yes	Sat	Dinner	5

```python
def pandas_sql5(table1):
    """
    SELECT *
    FROM tips
    WHERE size >= 5 OR total_bill > 45;
    :param table1:
    :return:
    """
    return table1[(table1['size']>=5) | (table1['total_bill']>45)]
```

运行结果

	ID	total_bill	tip	sex	smoker	day	time	size
59	59	48.27	6.73	Male	No	Sat	Dinner	4
125	125	29.80	4.20	Female	No	Thur	Lunch	6
141	141	34.30	6.70	Male	No	Thur	Lunch	6
142	142	41.19	5.00	Male	No	Thur	Lunch	5
143	143	27.05	5.00	Female	No	Thur	Lunch	6
155	155	29.85	5.14	Female	No	Sun	Dinner	5
156	156	48.17	5.00	Male	No	Sun	Dinner	6
170	170	50.81	10.00	Male	Yes	Sat	Dinner	3
182	182	45.35	3.50	Male	Yes	Sun	Dinner	3
185	185	20.69	5.00	Male	No	Sun	Dinner	5
187	187	30.46	2.00	Male	Yes	Sun	Dinner	5
212	212	48.33	8.00	Male	No	Sat	Dinner	4

(4) NULL 检测用 notna() and isna()方法

首先构造一个 dataframe，用于测试

```
def pandas_sql6():
    """
    :return:
    """
    fream1=pd.DataFrame({'Col1':['A','B',np.nan,'D','E'],'Col2':['E',np.nan,'F','G','H']})
    return fream1
```

构造的数据集如下：

	Col1	Col2
0	A	E
1	B	NaN
2	NaN	F
3	D	G
4	E	H

假设有个表和数据集是相同的结构，SQL 查询 Col2 里面为 NULL 的记录

```
SELECT *
FROM frame
WHERE col2 IS NULL;
```

Pandas 代码如下

```
def pandas_sql61(table1):
    """
    SELECT *
    FROM frame
    WHERE col2 IS NULL;
    :param table1:
    :return:
    """
    print(table1[table1['Col2'].isna()])
```

```
def pandas_sql6():
    """
    :return:
    """
    fream1=pd.DataFrame({'Col1':['A','B',np.nan,'D','E'],'Col2':['E',np.nan,'F','G','H']})
    return fream1
```

```
print(pandas_sql61(pandas_sql6()))
```

运行结果

```
  Col1 Col2
1    B  NaN
None
```

(5) SQL：col1 IS NOT NULL ;Pandas：

```
def pandas_sql61(table1):
    """
    SELECT *
    FROM frame
    WHERE col2 IS NULL;
    :param table1:
    :return:
    """
    print(table1[table1['Col2'].isna()])
    """
    SELECT *
    FROM frame
    WHERE col1 IS NOT NULL;
    """
    print(table1[table1['Col1'].notna()])
```

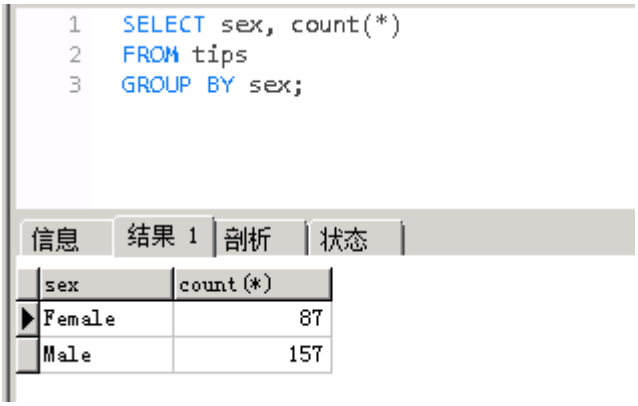
运行结果如下

```
  Col1 Col2
1    B  NaN
  Col1 Col2
0    A    E
1    B  NaN
3    D    G
4    E    H
None
```

GROPY BY

在 pandas 里面，SQL's GROUP BY 操作是执行命令是 groupby()方法。groupby() 典型的应用涉及到一个过程可以分开一个数据集到分组里面，应用一些方法(典型的聚集体),然后把这些分组聚集在一起。

(1)SQL ，按照性别进行分组求和



Pandas 如下

```python
def pandas_sql7(table1):
    """
    SELECT sex, count(*)
    FROM tips
    GROUP BY sex;
    :param table1:
    :return:
    """
    print(table1.groupby('sex').size())
    print(table1.groupby('sex').count())
    print(table1.groupby('sex')['total_bill'].count())
```

运行结果

```
sex
Female     87
Male      157
dtype: int64
        ID  total_bill  tip  smoker  day  time  size
sex
Female   87          87   87      87   87    87    87
Male    157         157  157     157  157   157   157
sex
Female     87
Male      157
Name: total_bill, dtype: int64

Process finished with exit code 0
```

注意：在 pandas 里面我们采用 size() 不用 count().这是因为 count()应用方法是针对每一列的,返回非空列的数量。

同样，可以用 count()方法到一个单独的列。

(2)多个函数也可以应用，例如，想看到一周中每天小费的金额和数量的变化，agg()方法允许通过字段传递到分组数据集里面去，标志函数应用到指定的列中去。

```sql
SELECT day, AVG(tip), COUNT(*)
FROM tips
GROUP BY day;
```

信息 | 结果 1 | 剖析 | 状态

day	AVG(tip)	COUNT(*)
Fri	2.734736842105263	19
Sat	2.993103448275862	87
Sun	3.255131578947369	76
Thur	2.771451612903226	62

Pandas 代码

```python
def pandas_sql8(table1):
    """
    SELECT day, AVG(tip), COUNT(*)
    FROM tips
    GROUP BY day;
    :param table1:
    :return:
    """
    print(table1.groupby('day').agg({'tip': np.mean, 'day': np.size}))
```

```
            tip  day
day
Fri    2.734737   19
Sat    2.993103   87
Sun    3.255132   76
Thur   2.771452   62
```

(3) SQL 分组

```sql
SELECT smoker, day, COUNT(tip), AVG(tip)
FROM tips
GROUP BY smoker, day;
```

信息 | 结果 1 | 剖析 | 状态

smoker	day	COUNT(tip)	AVG(tip)
No	Fri	4	2.8125
No	Sat	45	3.102888888888889
No	Sun	57	3.1678947368421055
No	Thur	45	2.673777777777778
Yes	Fri	15	2.714
Yes	Sat	42	2.8754761904761903
Yes	Sun	19	3.5168421052631573
Yes	Thur	17	3.0299999999999994

对应 Pandas 代码

```python
def pandas_sql9(table1):
    """
    SELECT smoker, day, COUNT(*), AVG(tip)
    FROM tips
    GROUP BY smoker, day;
    :param table1:
    :return:
    """
    table2=table1.groupby(['smoker','day']).agg({'tip':[np.size,np.mean]})
    print(table2)
```

```
              tip
             size      mean
smoker day
No     Fri    4.0  2.812500
       Sat   45.0  3.102889
       Sun   57.0  3.167895
       Thur  45.0  2.673778
Yes    Fri   15.0  2.714000
       Sat   42.0  2.875476
       Sun   19.0  3.516842
       Thur  17.0  3.030000

Process finished with exit code 0
```

mean()函数的功能是求取平均值

size()函数主要是用来统计矩阵元素个数

JOIN

JOIN 能被 join()或者 merge()执行。默认情况下，join()也将加入到索引数据集中。每一个方法都有参数允许指定连接的类型(LEFT,RIGHT,INNER,FULL)或者要连接的列(列名或者索引)

首先构造 2 个数据集 df1,df2

```
def pandas_createDataFrame():
    """
    :return:
    """
    df1=pd.DataFrame({'key':['A','B','C','D'],'value':np.random.randn(4)})
    print(df1)
    df2=pd.DataFrame({'key':['B','D','D','E'],'value':np.random.randn(4)})
    print(df2)
```

```
  key     value
0   A  0.351292
1   B -0.896812
2   C  0.348098
3   D -0.222054
  key     value
0   B  0.663671
1   D -1.299508
2   D -0.363177
3   E -0.711688
```

假设有 2 个相同名称和结构的数据库表在数据集中。让我们回顾一下各种类型的连接。

(1) INNER JOIN

```
SELECT *
FROM df1
INNER JOIN df2
  ON df1.key = df2.key;
```

```python
def pandas_sql10(table1,table2):
    """
    SELECT *
    FROM df1
    INNER JOIN df2
    ON df1.key = df2.key;
    :param table1:
    :param table2:
    :return:
    """
    print(pd.merge(table1,table2,on='key'))
```

```python
def pandas_createDataFrame():
    """
    :return:
    """
    df1=pd.DataFrame({'key':['A','B','C','D'],'value':np.random.randn(4)})
    print(df1)
    df2=pd.DataFrame({'key':['B','D','D','E'],'value':np.random.randn(4)})
    print(df2)
    pandas_sql10(df1,df2)
```

运行结果

```
  key   value_x   value_y
0   B  0.097090 -0.302145
1   D  0.806863  0.969167
2   D  0.806863 -0.179546
```

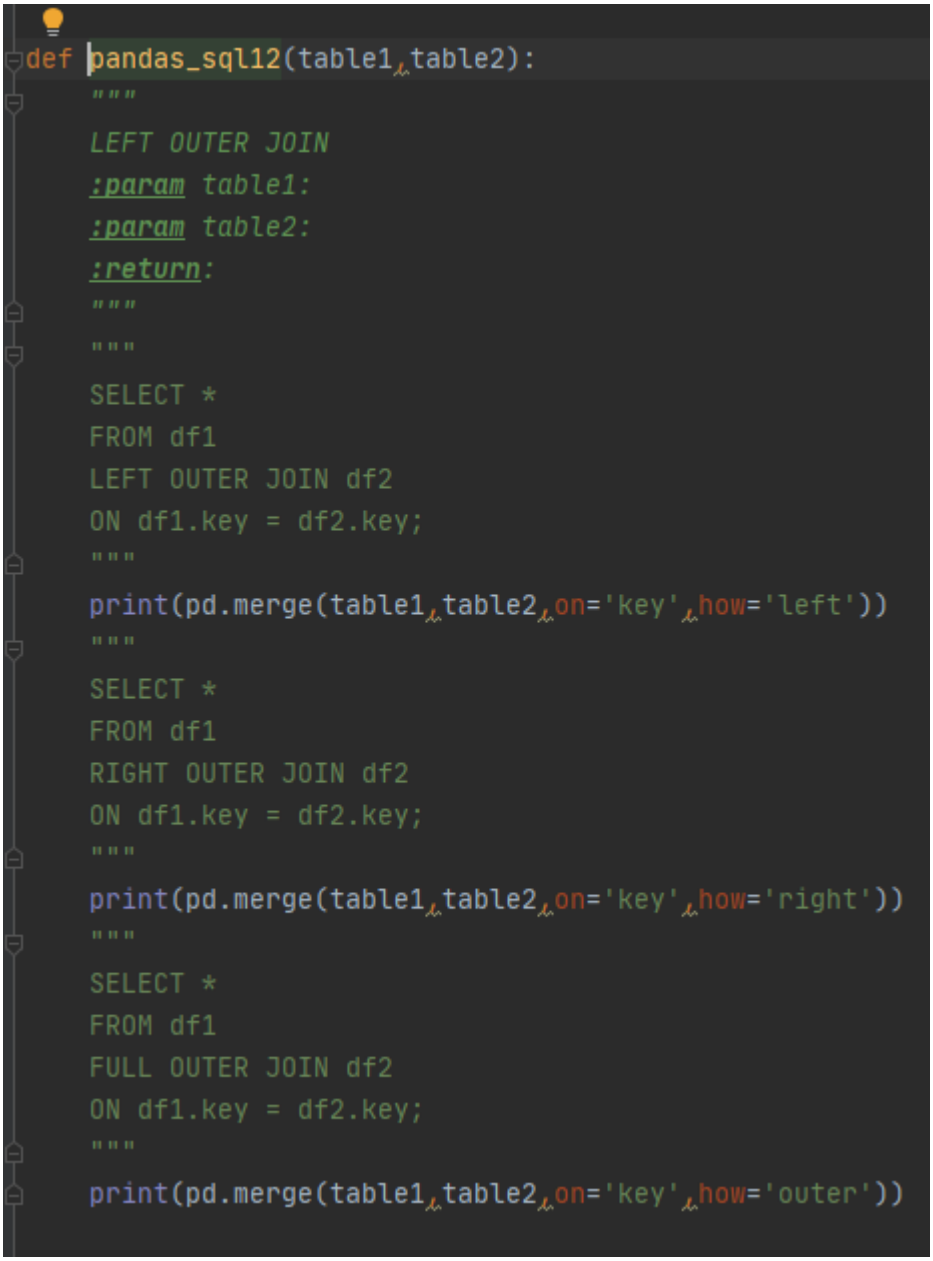
当你想一个数据集的列 与另一个数据集的索引连接起来的情况， Merge() 也允许提供参数。

```python
def pandas_sql11(table1,table2):
    indexed_df2=table2.set_index('key')
    print(pd.merge(table1,indexed_df2,left_on='key',right_index=True))
```

运行结果

```
  key   value_x   value_y
0   B  0.097090 -0.302145
1   D  0.806863  0.969167
2   D  0.806863 -0.179546
```

(2) LEFT OUTER JOIN(左连接); RIGHT JOIN(右连接); FULL JOIN(全连接)



```
def pandas_sql12(table1,table2):
    """
    LEFT OUTER JOIN
    :param table1:
    :param table2:
    :return:
    """
    """
    SELECT *
    FROM df1
    LEFT OUTER JOIN df2
    ON df1.key = df2.key;
    """
    print(pd.merge(table1,table2,on='key',how='left'))
    """
    SELECT *
    FROM df1
    RIGHT OUTER JOIN df2
    ON df1.key = df2.key;
    """
    print(pd.merge(table1,table2,on='key',how='right'))
    """
    SELECT *
    FROM df1
    FULL OUTER JOIN df2
    ON df1.key = df2.key;
    """
    print(pd.merge(table1,table2,on='key',how='outer'))
```

```
  key     value
0   A  0.535097
1   B  0.580922
2   C  0.180833
3   D  0.007489
  key     value
0   B -1.393951
1   D -0.113544
2   D  1.168533
3   E  0.829729
  key   value_x   value_y
0   A  0.535097       NaN
1   B  0.580922 -1.393951
2   C  0.180833       NaN
3   D  0.007489 -0.113544
4   D  0.007489  1.168533
  key   value_x   value_y
0   B  0.580922 -1.393951
1   D  0.007489 -0.113544
2   D  0.007489  1.168533
3   E       NaN  0.829729
  key   value_x   value_y
0   A  0.535097       NaN
1   B  0.580922 -1.393951
2   C  0.180833       NaN
3   D  0.007489 -0.113544
4   D  0.007489  1.168533
5   E       NaN  0.829729
```

UNION

UNION ALL 可以用 concat()执行

```python
def pandas_createDataFrame11():
    """
    UNION
    SELECT city, rank
    FROM df1
    UNION ALL
    SELECT city, rank
    FROM df2;
    :return:
    """
    df1=pd.DataFrame({'city':['Shanghai','BeiJing','Suzhou','Nanjing'],'rank':range(1,5)})
    print(df1)
    df2=pd.DataFrame({'city':['Shanghai','Zhengzhou','Wuxi'],'rank':[1,4,5]})
    print(df2)
    print(pd.concat([df1,df2]))
    """
```

```
      city  rank
0  Shanghai     1
1   BeiJing     2
2    Suzhou     3
3   Nanjing     4
      city  rank
0  Shanghai     1
1  Zhengzhou     4
2       Wuxi     5
      city  rank
0  Shanghai     1
1   BeiJing     2
2    Suzhou     3
3   Nanjing     4
0  Shanghai     1
1  Zhengzhou     4
2       Wuxi     5
```

SQLUNION 类似与 UNION ALL，然而 UNION 也将删除重复的行。

```python
    """
    SELECT city, rank
    FROM df1
    UNION
    SELECT city, rank
    FROM df2;
    """
    print(pd.concat([df1,df2]).drop_duplicates())
```

```
         city  rank
0    Shanghai     1
1     BeiJing     2
2      Suzhou     3
3     Nanjing     4
1   Zhengzhou     4
2        Wuxi     5
```

在 pandas 中，可以用 concat()和 drop_duplicates()连接在一起使用。

Pandas 与一些 SQL 分析聚合功能

(1) 偏移行显示

```
1  SELECT * FROM tips
2  ORDER BY tip DESC
3  LIMIT 10 OFFSET 5;
```

信息 | 结果 1 | 剖析 | 状态

total_bill	tip	sex	smoker	day	time	size
28.17	6.5	Female	Yes	Sat	Dinner	3
32.4	6.0	Male	No	Sun	Dinner	4
29.03	5.92	Male	No	Sat	Dinner	3
24.71	5.85	Male	No	Thur	Lunch	2
23.33	5.65	Male	Yes	Sun	Dinner	2
30.4	5.6	Male	No	Sun	Dinner	4
34.81	5.2	Female	No	Sun	Dinner	4
34.83	5.17	Female	No	Thur	Lunch	4
25.89	5.16	Male	Yes	Sat	Dinner	4
7.25	5.15	Male	Yes	Sun	Dinner	2

```python
def pandas_sql13(table1):
    """
    SELECT * FROM tips
    ORDER BY tip DESC
    LIMIT 10 OFFSET 5;
    :return:
    """
    # print(table1.nlargest(15, columns='tip').tail(10))
    print(table1.nlargest(11 + 5, columns='tip').tail(10))
```

	ID	total_bill	tip	sex	smoker	day	time	size
214	214	28.17	6.50	Female	Yes	Sat	Dinner	3
47	47	32.40	6.00	Male	No	Sun	Dinner	4
239	239	29.03	5.92	Male	No	Sat	Dinner	3
88	88	24.71	5.85	Male	No	Thur	Lunch	2
181	181	23.33	5.65	Male	Yes	Sun	Dinner	2
44	44	30.40	5.60	Male	No	Sun	Dinner	4
52	52	34.81	5.20	Female	No	Sun	Dinner	4
85	85	34.83	5.17	Female	No	Thur	Lunch	4
211	211	25.89	5.16	Male	Yes	Sat	Dinner	4
172	172	7.25	5.15	Male	Yes	Sun	Dinner	2

UPDATE

```
UPDATE tips
SET tip = tip*2
WHERE tip < 2;
```

```python
def pandas_sql14(table1):
    """
    UPDATE tips
    SET tip = tip*2
    WHERE tip < 2;
    :param table1:
    :return:
    """
    # tips[tips['time'] == 'Dinner'].head(5)
    # print(table1[table1['tip']==20])
    # table2=table1[table1['tip'] < 2].size
    # print(table1[table1['tip'] < 2].count())
    print(table1[table1['tip'] < 2])
    print(table1[table1['tip'] < 2]['tip'].count())
    # table1.loc[table1['tip'] < 2, 'tip'] *= 2
    table1.loc[table1['tip'] < 2, 'tip'] = table1.loc[table1['tip'] < 2, 'tip'] * 2
    print(table1[table1['tip'] < 2]['tip'].count())
    # print(table1[table1['tip'] > 2])
    # table1.loc[table1['tip'] < 2, 'tip'] *= 2
    #tips[(tips['time'] == 'Dinner') & (tips['tip'] > 5.00)]
```

运行结果

```
45
0
```

DELETE

```
DELETE FROM tips
WHERE tip > 9;
```

在 pandas 里面应该选择保留的行，而不是删除他们

```python
def pandas_sql15(table1):
    """
    DELETE FROM tips
    WHERE tip > 9;
    :param table1:
    :return:
    """
    print(table1['tip'].count())
    print(table1[table1['tip']>9])
    print(table1[table1['tip']>9]['tip'].count())
    # print(table1)
    # tips = tips.loc[tips['tip'] <= 9]
    table1=table1.loc[table1['tip']<=9]
    print(table1['tip'].count())
```

```
244
      ID  total_bill   tip   sex smoker  day    time  size
170  170       50.81  10.0  Male    Yes  Sat  Dinner     3
1
243
```

From

https://pandas.pydata.org/pandas-docs/stable/getting_started/comparison/comparison_with_sql.html

Comparison with SQL