一场Pandas与SQL的巅峰大战四

在之前的三篇系列文章中，我们对比了pandas和SQL在数据方面的多项操作。具体来讲，第一篇文章涉及到数据查看，去重计数，条件选择，合并连接，分组排序等操作，第二篇文章涉及字符串处理，窗口函数，行列转换，类型转换等操作。第三篇文章围绕日期操作展开，主要讨论了日期获取，日期转换，日期计算等内容。本篇文章一起来学习常见的应用实例：如何在SQL和pandas中计算同环比。将分别在MySQL，Hive SQL和pandas中庸多种方案来实现样例数据日环比，周同比计算。

数据准备

同比和环比本身都是相对的概念，同比是指和上个周期内同期数据的对比，可以是年同比，月同比，周同比等。环比是指连续两个统计周期内数据的对比，可以是日环比，周环比，月环比等。工作中常见的是周同比和日环比。周同比即当天和上周同一天数据的变化百分比，日环比即当天和昨天数据的变化百分比。本文也主要计算周同比和日环比。数据概况如下，是随机生成的两个月的销售额数据。

数据样例如下所示，从左到右依次表示，id，日期，当日销售额，数据周期从2019-11-01到2019-12-31。公众号后台回复“**对比四**”，即可获取本文全部代码和数据。

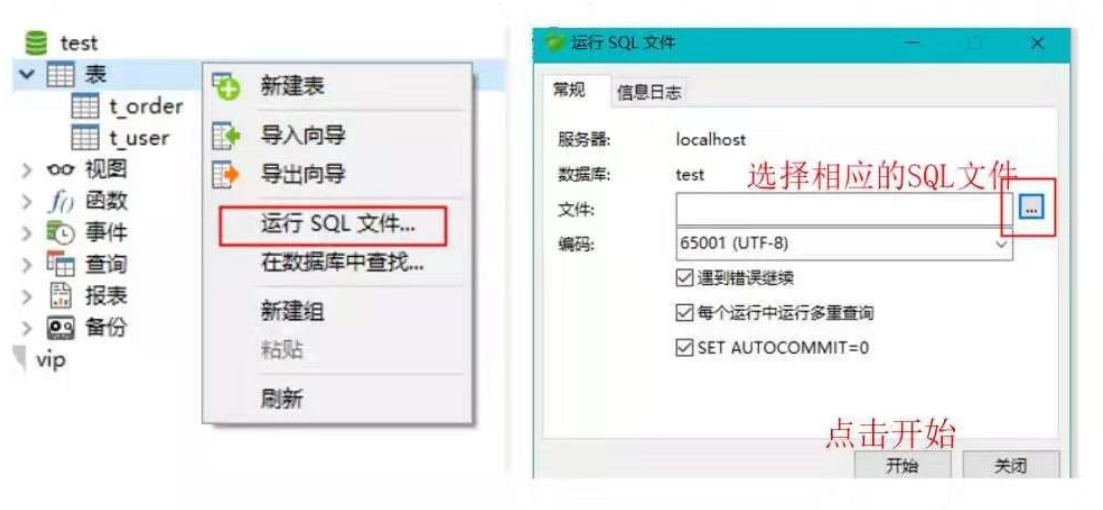
id	dt	amt
1	2019-11-01	1,043
2	2019-11-02	1,119
3	2019-11-03	878
4	2019-11-04	1,197
5	2019-11-05	1,156
6	2019-11-06	1,142
7	2019-11-07	826
8	2019-11-08	948
9	2019-11-09	1,153
10	2019-11-10	826
11	2019-11-11	841
12	2019-11-12	1,017
13	2019-11-13	1,022
14	2019-11-14	945
15	2019-11-15	979
16	2019-11-16	925
17	2019-11-17	885
18	2019-11-18	1,041

pandas加载数据

```
import pandas as pd
import datetime

orderamt = pd.read_excel('orderamt.xlsx')
orderamt.head()
```

MySQL加载数据



和前面的文章类似，使用navicate把我准备的`orderamt.sql`导入数据库中即可。

Hive 加载数据

```
CREATE TABLE `t_orderamt`(
  `id` int,
  `dt` string,
  `orderamt` float)
row format delimited fields terminated by ','
stored as textfile;

load data local inpath 'orderamt.txt' overwrite into table t_orderamt;
select * from t_orderamt limit 20;
```

```
hive> select * from t_orderamt limit 10;
OK
1       2019-11-01      1043.0
2       2019-11-02      1119.0
3       2019-11-03      878.0
4       2019-11-04      1197.0
5       2019-11-05      1156.0
6       2019-11-06      1142.0
7       2019-11-07      826.0
8       2019-11-08      948.0
9       2019-11-09      1153.0
10      2019-11-10      826.0
```

按照上面的代码建表，然后把`orderamt.txt`的内容加载到表中即可，最终数据如上图所示。

SQL计算周同比和日环比

我们关注的是周同比和日环比，其实就是关注当天，昨天，7天前的数据，然后相应的算一下变化的百分比即可。思路一：自关联，关联条件是日期差分别是1和7，分别求出当天，昨天，7天前的数据，用三列形式展示，之后就可以进行作差和相除求得百分比。思路二：不进行关联，直接查询当前日期前一天和前七天的数据，同样以3列的形式展示。

来看一下SQL代码：

```sql
select a.*, b.orderamt ld_amt, c.orderamt lw_amt
from t_orderamt a
left join t_orderamt b
on DATEDIFF(a.dt, b.dt) = 1
left join t_orderamt c
on DATEDIFF(a.dt, c.dt) = 7
order by dt
;
```

信息 | 结果1 | 概况 | 状态

id	dt	orderamt	ld_amt	lw_amt
1	2019-11-01	1043	(Null)	(Null)
2	2019-11-02	1119	1043	(Null)
3	2019-11-03	878	1119	(Null)
4	2019-11-04	1197	878	(Null)
5	2019-11-05	1156	1197	(Null)
6	2019-11-06	1142	1156	(Null)
7	2019-11-07	826	1142	(Null)
8	2019-11-08	948	826	1043
9	2019-11-09	1153	948	1119
10	2019-11-10	826	1153	878
11	2019-11-11	841	826	1197
12	2019-11-12	1017	841	1156
13	2019-11-13	1022	1017	1142

上面代码中我们关联了两次，条件分别是日期相差1天和日期相差7天。关联不上的则留空。

再来看另一种写法：

```
select *,
(select orderamt from t_orderamt where dt = date_add(a.dt, interval -1 day)) ld_amt,
(select orderamt from t_orderamt where dt = date_add(a.dt, interval -7 day)) lw_amt
from t_orderamt a
```

信息 | 结果1 | 概况 | 状态

id	dt	orderamt	ld_amt	lw_amt
1	2019-11-01	1043	(Null)	(Null)
2	2019-11-02	1119	1043	(Null)
3	2019-11-03	878	1119	(Null)
4	2019-11-04	1197	878	(Null)
5	2019-11-05	1156	1197	(Null)
6	2019-11-06	1142	1156	(Null)
7	2019-11-07	826	1142	(Null)
8	2019-11-08	948	826	1043
9	2019-11-09	1153	948	1119
10	2019-11-10	826	1153	878
11	2019-11-11	841	826	1197
12	2019-11-12	1017	841	1156
13	2019-11-13	1022	1017	1142

这种写法巧妙地使用表的别名查询出了前1天和前7天的金额，效果和第一种写法一样，不过这种写法可能小众一点。

回到上面的思路2，我们在前面的学习中知道，Hive中有窗口函数支持查询当前行前n行的数据，可以实现同样的效果。代码如下：

```
select *, lag(orderamt, 1) over(order by dt) ld_amt,
lag(orderamt, 7) over(order by dt) lw_amt
from t_orderamt;
```

```
1       2019-11-01      1043.0  NULL    NULL
2       2019-11-02      1119.0  1043.0  NULL
3       2019-11-03      878.0   1119.0  NULL
4       2019-11-04      1197.0  878.0   NULL
5       2019-11-05      1156.0  1197.0  NULL
6       2019-11-06      1142.0  1156.0  NULL
7       2019-11-07      826.0   1142.0  NULL
8       2019-11-08      948.0   826.0   1043.0
9       2019-11-09      1153.0  948.0   1119.0
10      2019-11-10      826.0   1153.0  878.0
11      2019-11-11      841.0   826.0   1197.0
12      2019-11-12      1017.0  841.0   1156.0
13      2019-11-13      1022.0  1017.0  1142.0
14      2019-11-14      945.0   1022.0  826.0
15      2019-11-15      979.0   945.0   948.0
```

以上面的代码为基础，稍加修改，增加计算百分比的代码，就可以分别得到周同比和日环比。

```
select a.*, concat(round(((a.orderamt - b.orderamt) / b.orderamt) * 100,2), '%')
as ld_pct,
concat(round(((a.orderamt - c.orderamt) / c.orderamt) * 100,2), '%') as lw_pct
```

```
from t_orderamt a
left join t_orderamt b
on DATEDIFF(a.dt, b.dt) = 1
left join t_orderamt c
on DATEDIFF(a.dt, c.dt) = 7
order by dt
;

select
b.id, b.dt, b.orderamt,
concat(round(((b.orderamt - ld_amt) / ld_amt) * 100,2), '%') as ld_pct,
concat(round(((b.orderamt - lw_amt) / lw_amt) * 100,2), '%') as lw_pct
from
(
select *,
(select orderamt from t_orderamt where dt = date_add(a.dt, interval -1 day))
ld_amt,
(select orderamt from t_orderamt where dt = date_add(a.dt, interval -7 day))
lw_amt
from t_orderamt a
) b
;

select
b.id, b.dt, b.orderamt,
concat(round(((b.orderamt - ld_amt) / ld_amt) * 100,2), '%') as ld_pct,
concat(round(((b.orderamt - lw_amt) / lw_amt) * 100,2), '%') as lw_pct
from
(
select *, lag(orderamt, 1) over(order by dt) ld_amt,
lag(orderamt, 7) over(order by dt) lw_amt
from t_orderamt
) b
```

id	dt	orderamt	ld_pct	lw_pct
1	2019-11-01	1043	(Null)	(Null)
2	2019-11-02	1119	7.29%	(Null)
3	2019-11-03	878	-21.54%	(Null)
4	2019-11-04	1197	36.33%	(Null)
5	2019-11-05	1156	-3.43%	(Null)
6	2019-11-06	1142	-1.21%	(Null)
7	2019-11-07	826	-27.67%	(Null)
8	2019-11-08	948	14.77%	-9.11%
9	2019-11-09	1153	21.62%	3.04%
10	2019-11-10	826	-28.36%	-5.92%
11	2019-11-11	841	1.82%	-29.74%
12	2019-11-12	1017	20.93%	-12.02%

1	2019-11-01	1043.0	NULL	NULL
2	2019-11-02	1119.0	7.29%	NULL
3	2019-11-03	878.0	-21.54%	NULL
4	2019-11-04	1197.0	36.33%	NULL
5	2019-11-05	1156.0	-3.43%	NULL
6	2019-11-06	1142.0	-1.21%	NULL
7	2019-11-07	826.0	-27.67%	NULL
8	2019-11-08	948.0	14.77%	-9.11%
9	2019-11-09	1153.0	21.62%	3.04%
10	2019-11-10	826.0	-28.36%	-5.92%
11	2019-11-11	841.0	1.82%	-29.74%
12	2019-11-12	1017.0	20.93%	-12.02%
13	2019-11-13	1022.0	0.49%	-10.51%
14	2019-11-14	945.0	-7.53%	14.41%
15	2019-11-15	979.0	3.6%	3.27%

pandas计算周同比和日环比

在pandas中，我们同样首先按照上面的两种思路进行计算。

方法一：日期关联的方法

```
import pandas as pd
import datetime
orderamt = pd.read_excel('orderamt.xlsx')
#orderamt['dt'] = orderamt['dt'].apply(lambda x: datetime.datetime.strptime(x,
'%Y-%m-%d'))#为了便于日期加减，将dt转换为datetime64[ns]的格式，视情况运行该句

#分别构造两个dateframe用于关联
orderamt_plus_1 = orderamt.copy()
orderamt_plus_7 = orderamt.copy()

orderamt_plus_1['dt'] = orderamt_plus_1['dt'] + datetime.timedelta(days=1)
orderamt_plus_7['dt'] = orderamt_plus_7['dt'] + datetime.timedelta(days=7)
orderamt_1 = pd.merge(orderamt, orderamt_plus_1, on=['dt'],how='left')
orderamt_1_7 = pd.merge(orderamt_1, orderamt_plus_7, on=['dt'],how='left')
orderamt_all = orderamt_1_7[['id_x', 'dt', 'amt_x', 'amt_y', 'amt']]
```

```
orderamt_all
```

	id_x	dt	amt_x	amt_y	amt
0	1	2019-11-01	1043	NaN	NaN
1	2	2019-11-02	1119	1043.0	NaN
2	3	2019-11-03	878	1119.0	NaN
3	4	2019-11-04	1197	878.0	NaN
4	5	2019-11-05	1156	1197.0	NaN
5	6	2019-11-06	1142	1156.0	NaN
6	7	2019-11-07	826	1142.0	NaN
7	8	2019-11-08	948	826.0	1043.0
8	9	2019-11-09	1153	948.0	1119.0
9	10	2019-11-10	826	1153.0	878.0
10	11	2019-11-11	841	826.0	1197.0
11	12	2019-11-12	1017	841.0	1156.0
12	13	2019-11-13	1022	1017.0	1142.0
13	14	2019-11-14	945	1022.0	826.0

方法二：直接选取前面n行的方法：

```
orderamt = pd.read_excel('orderamt.xlsx')
orderamt['ld_amt'] = orderamt['amt'].shift(1)
orderamt['lw_amt'] = orderamt['amt'].shift(7)
orderamt
```

```
orderamt = pd.read_excel('orderamt.xlsx')
orderamt['ld_amt'] = orderamt['amt'].shift(1)
orderamt['lw_amt'] = orderamt['amt'].shift(7)
orderamt
```

	id	dt	amt	ld_amt	lw_amt
0	1	2019-11-01	1043	NaN	NaN
1	2	2019-11-02	1119	1043.0	NaN
2	3	2019-11-03	878	1119.0	NaN
3	4	2019-11-04	1197	878.0	NaN
4	5	2019-11-05	1156	1197.0	NaN
5	6	2019-11-06	1142	1156.0	NaN
6	7	2019-11-07	826	1142.0	NaN
7	8	2019-11-08	948	826.0	1043.0
8	9	2019-11-09	1153	948.0	1119.0
9	10	2019-11-10	826	1153.0	878.0
10	11	2019-11-11	841	826.0	1197.0
11	12	2019-11-12	1017	841.0	1156.0

这样得到的效果和SQL方式是一致的。如果要计算百分比，同样是稍微加工即可：

```
#接方法一代码
orderamt_all['ld_pct'] = (orderamt_all['amt_x'] - orderamt_all['amt_y']) /
orderamt_all['amt_y']
orderamt_all['lw_pct'] = (orderamt_all['amt_x'] - orderamt_all['amt']) /
orderamt_all['amt']
orderamt_all
```

	id_x	dt	amt_x	amt_y	amt	ld_pct	lw_pct
0	1	2019-11-01	1043	NaN	NaN	NaN	NaN
1	2	2019-11-02	1119	1043.0	NaN	0.072867	NaN
2	3	2019-11-03	878	1119.0	NaN	-0.215371	NaN
3	4	2019-11-04	1197	878.0	NaN	0.363326	NaN
4	5	2019-11-05	1156	1197.0	NaN	-0.034252	NaN
5	6	2019-11-06	1142	1156.0	NaN	-0.012111	NaN
6	7	2019-11-07	826	1142.0	NaN	-0.276708	NaN
7	8	2019-11-08	948	826.0	1043.0	0.147700	-0.091083
8	9	2019-11-09	1153	948.0	1119.0	0.216245	0.030384
9	10	2019-11-10	826	1153.0	878.0	-0.283608	-0.059226
10	11	2019-11-11	841	826.0	1197.0	0.018160	-0.297410
11	12	2019-11-12	1017	841.0	1156.0	0.209275	-0.120242
12	13	2019-11-13	1022	1017.0	1142.0	0.004916	-0.105079

```
#接方法二代码
orderamt['ld_pct'] = (orderamt['amt'] - orderamt['ld_amt']) / orderamt['ld_amt']
orderamt['lw_pct'] = (orderamt['amt'] - orderamt['lw_amt']) / orderamt['lw_amt']
orderamt
```

	id	dt	amt	ld_amt	lw_amt	ld_pct	lw_pct
0	1	2019-11-01	1043	NaN	NaN	NaN	NaN
1	2	2019-11-02	1119	1043.0	NaN	0.072867	NaN
2	3	2019-11-03	878	1119.0	NaN	-0.215371	NaN
3	4	2019-11-04	1197	878.0	NaN	0.363326	NaN
4	5	2019-11-05	1156	1197.0	NaN	-0.034252	NaN
5	6	2019-11-06	1142	1156.0	NaN	-0.012111	NaN
6	7	2019-11-07	826	1142.0	NaN	-0.276708	NaN
7	8	2019-11-08	948	826.0	1043.0	0.147700	-0.091083
8	9	2019-11-09	1153	948.0	1119.0	0.216245	0.030384
9	10	2019-11-10	826	1153.0	878.0	-0.283608	-0.059226
10	11	2019-11-11	841	826.0	1197.0	0.018160	-0.297410
11	12	2019-11-12	1017	841.0	1156.0	0.209275	-0.120242

在pandas中，还有专门的计算同环比的函数pct_change。

方法三：使用pandas的pct_change()函数计算

```
orderamt = pd.read_excel('orderamt.xlsx')
orderamt['ld_pct'] = orderamt['amt'].pct_change()
orderamt['lw_pct'] = orderamt['amt'].pct_change(7)
orderamt
```

```
orderamt = pd.read_excel('orderamt.xlsx')
orderamt['ld_pct'] = orderamt['amt'].pct_change()
orderamt['lw_pct'] = orderamt['amt'].pct_change(7)
orderamt
```

	id	dt	amt	ld_pct	lw_pct
0	1	2019-11-01	1043	NaN	NaN
1	2	2019-11-02	1119	0.072867	NaN
2	3	2019-11-03	878	-0.215371	NaN
3	4	2019-11-04	1197	0.363326	NaN
4	5	2019-11-05	1156	-0.034252	NaN
5	6	2019-11-06	1142	-0.012111	NaN
6	7	2019-11-07	826	-0.276708	NaN
7	8	2019-11-08	948	0.147700	-0.091083
8	9	2019-11-09	1153	0.216245	0.030384
9	10	2019-11-10	826	-0.283608	-0.059226
10	11	2019-11-11	841	0.018160	-0.297410
11	12	2019-11-12	1017	0.209275	-0.120242

上面的代码中，我们都没有用百分比的形式保留结果，这里提供一种方式。

```
#接方法三，方法一二类似
orderamt['ld_pct'] = orderamt['ld_pct'].apply(lambda x: format(x, '.2%'))
orderamt['lw_pct'] = orderamt['lw_pct'].apply(lambda x: format(x, '.2%'))
orderamt
```

	id	dt	amt	ld_pct	lw_pct
0	1	2019-11-01	1043	nan%	nan%
1	2	2019-11-02	1119	7.29%	nan%
2	3	2019-11-03	878	-21.54%	nan%
3	4	2019-11-04	1197	36.33%	nan%
4	5	2019-11-05	1156	-3.43%	nan%
5	6	2019-11-06	1142	-1.21%	nan%
6	7	2019-11-07	826	-27.67%	nan%
7	8	2019-11-08	948	14.77%	-9.11%
8	9	2019-11-09	1153	21.62%	3.04%
9	10	2019-11-10	826	-28.36%	-5.92%
10	11	2019-11-11	841	1.82%	-29.74%
11	12	2019-11-12	1017	20.93%	-12.02%
12	13	2019-11-13	1022	0.49%	-10.51%

至此，我们完成了SQL和pandas中对于周同比和日环比计算的过程。

小结：

本篇文章中，我们使用SQL和pandas的多种方法对常见的周同比和日环比进行计算。在同样的思路指导下，SQL和pandas实现的方式各有特色，代码并不复杂，但值得细细品味。公众号后台回复“**对比四**”可以获取本文pdf版本，代码，数据等进行实战，希望对你有所帮助。