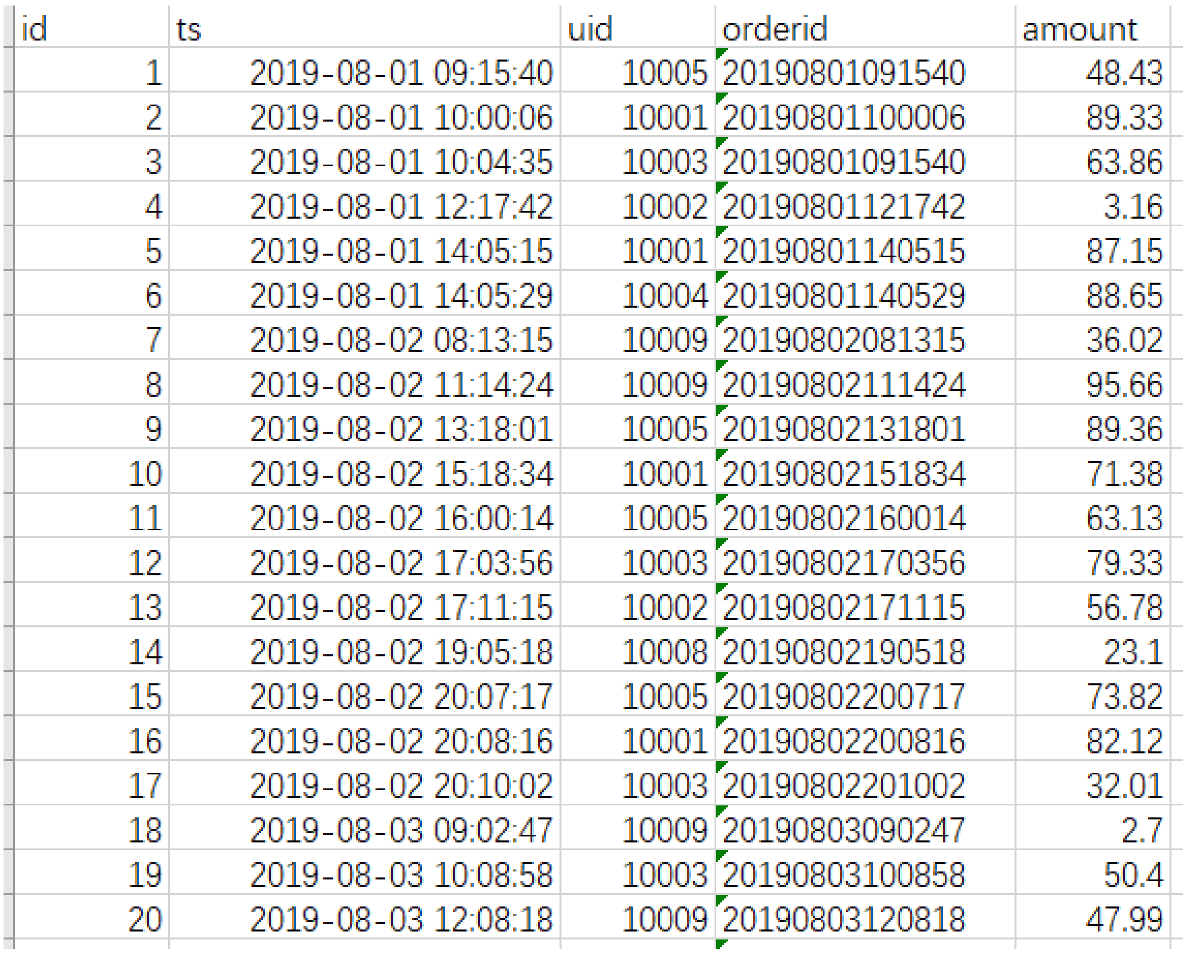
一场Pandas与SQL的巅峰大战三

在前两篇文章中，我们从多个角度，由浅入深，对比了pandas和SQL在数据处理方面常见的一些操作。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。您可以点击往期链接进行阅读回顾。在日常工作中，我们经常会与日期类型打交道，会在不同的日期格式之间转来转去。本文依然沿着前两篇文章的思路，对pandas和SQL中的日期操作进行总结，其中SQL采用Hive SQL+MySQL两种方式，内容与前两篇相对独立又彼此互为补充。一起开始学习吧！

数据概况

数据方面，我们依然采用前面文章的订单数据，样例如下。在正式开始学习之前，我们需要把数据加载到dataframe和数据表中。本文的数据和代码可以在公众号后台回复"**对比三**"获取哦~



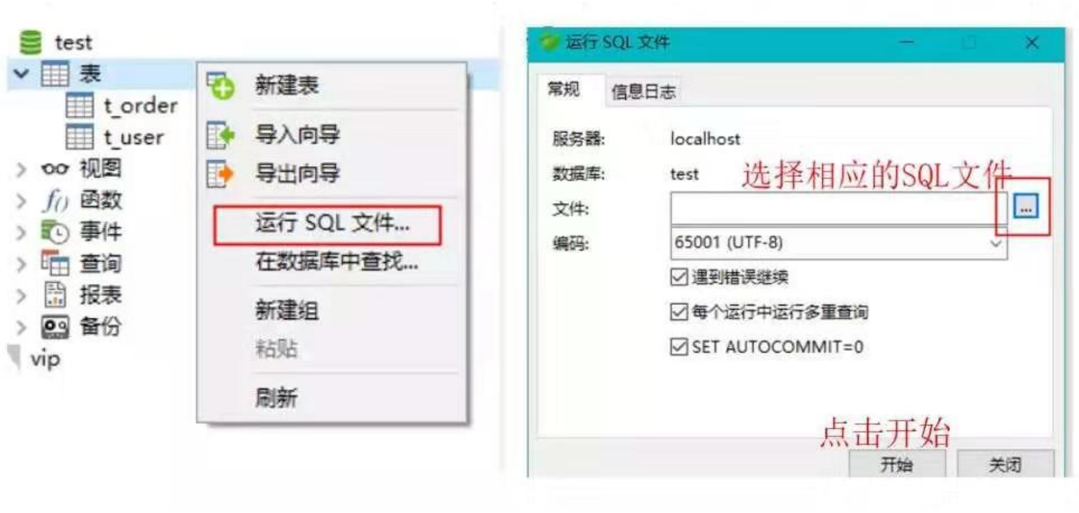
id	ts	uid	orderid	amount
1	2019-08-01 09:15:40	10005	20190801091540	48.43
2	2019-08-01 10:00:06	10001	20190801100006	89.33
3	2019-08-01 10:04:35	10003	20190801091540	63.86
4	2019-08-01 12:17:42	10002	20190801121742	3.16
5	2019-08-01 14:05:15	10001	20190801140515	87.15
6	2019-08-01 14:05:29	10004	20190801140529	88.65
7	2019-08-02 08:13:15	10009	20190802081315	36.02
8	2019-08-02 11:14:24	10009	20190802111424	95.66
9	2019-08-02 13:18:01	10005	20190802131801	89.36
10	2019-08-02 15:18:34	10001	20190802151834	71.38
11	2019-08-02 16:00:14	10005	20190802160014	63.13
12	2019-08-02 17:03:56	10003	20190802170356	79.33
13	2019-08-02 17:11:15	10002	20190802171115	56.78
14	2019-08-02 19:05:18	10008	20190802190518	23.1
15	2019-08-02 20:07:17	10005	20190802200717	73.82
16	2019-08-02 20:08:16	10001	20190802200816	82.12
17	2019-08-02 20:10:02	10003	20190802201002	32.01
18	2019-08-03 09:02:47	10009	20190803090247	2.7
19	2019-08-03 10:08:58	10003	20190803100858	50.4
20	2019-08-03 12:08:18	10009	20190803120818	47.99

pandas加载数据

```
import pandas as pd
data = pd.read_excel('order.xlsx')
#data2 = pd.read_excel('order.xlsx', parse_dates=['ts'])
data.head()
data.dtypes
```

需要指出，pandas读取数据对于日期类型有特殊的支持。无论是在read_csv中还是在read_excel中，都有parse_dates参数，可以把数据集中的一列或多列转成pandas中的日期格式。上面代码中的data是使用默认的参数读取的，在data.dtypes的结果中ts列是`datetime64[ns]`格式，而data2是显示指定了ts为日期列，因此data2的ts类型也是`datetime[ns]`。如果在使用默认方法读取时，日期列没有成功转换，就可以使用类似data2这样显示指定的方式。

MySQL 加载数据

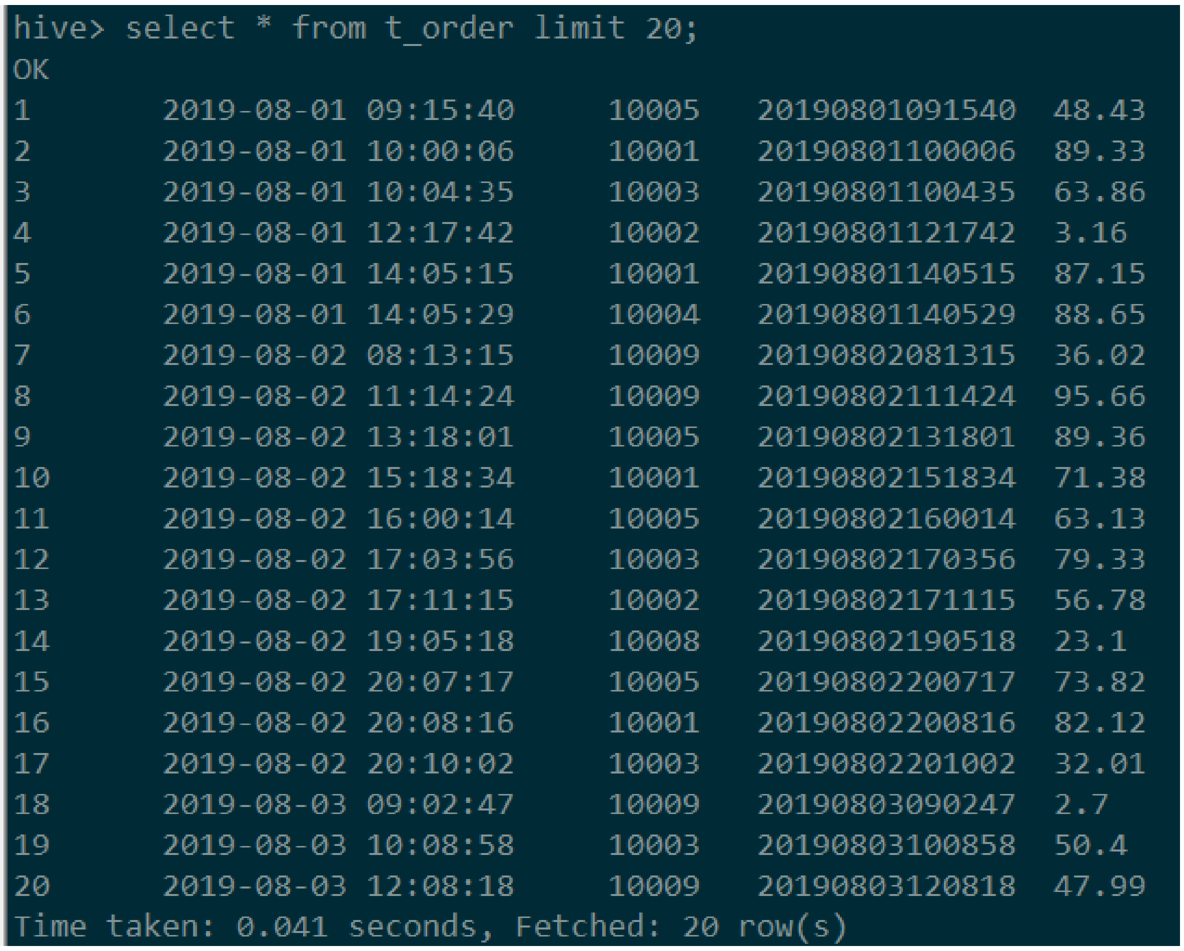


我准备了一个sql文件，推荐使用navicate客户端，按照途中所示方式，直接导入即可。

Hive加载数据

```
create table `t_order`(
`id` int,
`ts` string,
`uid` string,
`orderid` string,
`amount` float
)
row format delimited fields terminated by ','
stored as textfile;

load data local inpath 't_order.csv' overwrite into table t_order;
select * from t_order limit 20;
```

在hive中加载数据我们需要先建立表，然后把文本文件中的数据load到表中，结果如下图所示。



```
hive> select * from t_order limit 20;
OK
1       2019-08-01 09:15:40     10005   20190801091540  48.43
2       2019-08-01 10:00:06     10001   20190801100006  89.33
3       2019-08-01 10:04:35     10003   20190801100435  63.86
4       2019-08-01 12:17:42     10002   20190801121742  3.16
5       2019-08-01 14:05:15     10001   20190801140515  87.15
6       2019-08-01 14:05:29     10004   20190801140529  88.65
7       2019-08-02 08:13:15     10009   20190802081315  36.02
8       2019-08-02 11:14:24     10009   20190802111424  95.66
9       2019-08-02 13:18:01     10005   20190802131801  89.36
10      2019-08-02 15:18:34     10001   20190802151834  71.38
11      2019-08-02 16:00:14     10005   20190802160014  63.13
12      2019-08-02 17:03:56     10003   20190802170356  79.33
13      2019-08-02 17:11:15     10002   20190802171115  56.78
14      2019-08-02 19:05:18     10008   20190802190518  23.1
15      2019-08-02 20:07:17     10005   20190802200717  73.82
16      2019-08-02 20:08:16     10001   20190802200816  82.12
17      2019-08-02 20:10:02     10003   20190802201002  32.01
18      2019-08-03 09:02:47     10009   20190803090247  2.7
19      2019-08-03 10:08:58     10003   20190803100858  50.4
20      2019-08-03 12:08:18     10009   20190803120818  47.99
Time taken: 0.041 seconds, Fetched: 20 row(s)
```

我们在MySQL和Hive中都把时间存储成字符串，这在工作中比较常见，使用起来也比较灵活和习惯，因此我们没有使用专门的日期类型。

开始学习

我们把日期相关的操作分为**日期获取**，**日期转换**，**日期计算**三类。下面开始逐一学习。

日期获取

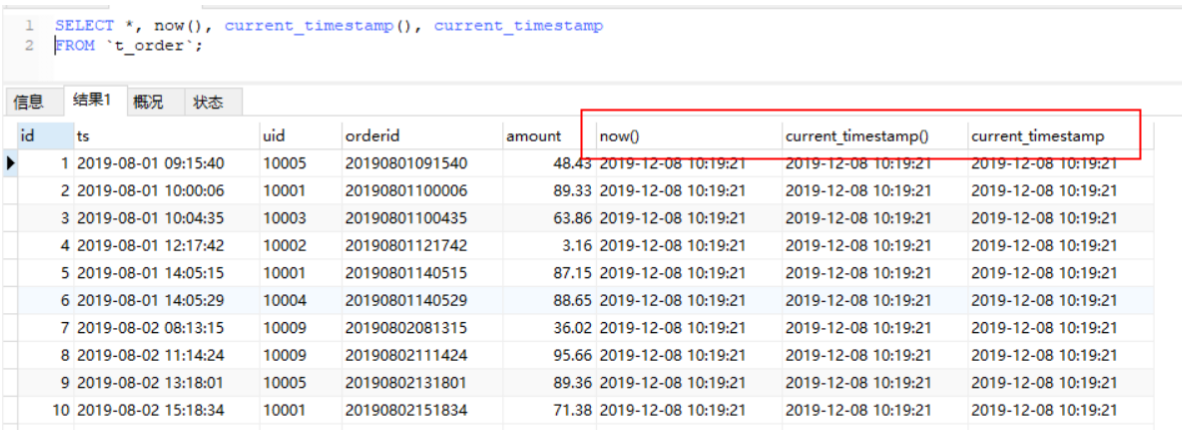
1.获取当前时间，年月日时分秒

pandas中可以使用now函数获取当前时间，但需要再进行一次格式化操作来调整显示的格式。我们在数据集上新加一列当前时间的操作如下：

```
data['current_dt'] = pd.datetime.now()
data['current_dt'] = data['current_dt'].apply(lambda x : x.strftime('%Y-%m-%d %H:%M:%S'))
data.head()
```

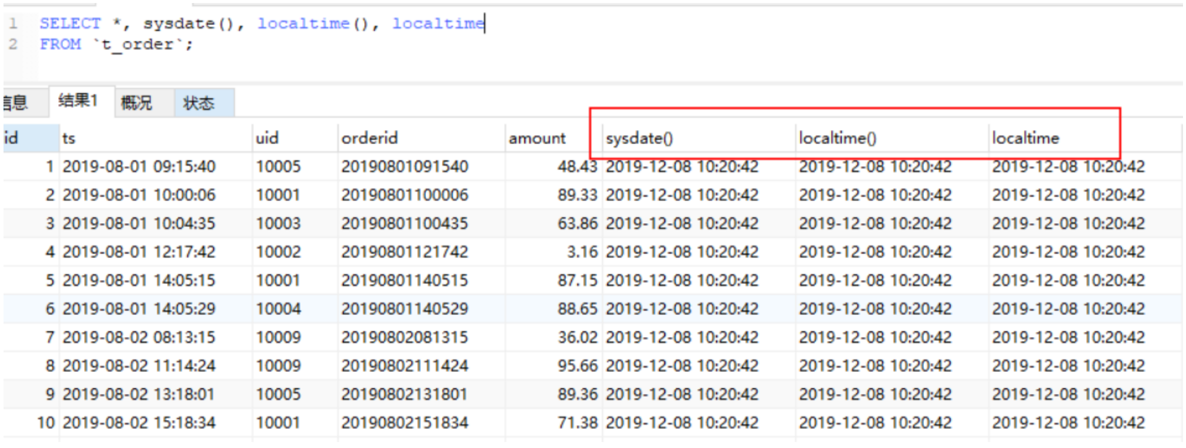
	id	ts	uid	orderid	amount	current_dt
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-12-08 10:12:49
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-12-08 10:12:49
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-12-08 10:12:49
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-12-08 10:12:49
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-12-08 10:12:49

MySQL有多个函数可以获取当前时间：now()，current_timestamp，current_timestamp()，sysdate()，localtime()，localtime，localtimestamp，localtimestamp()等。



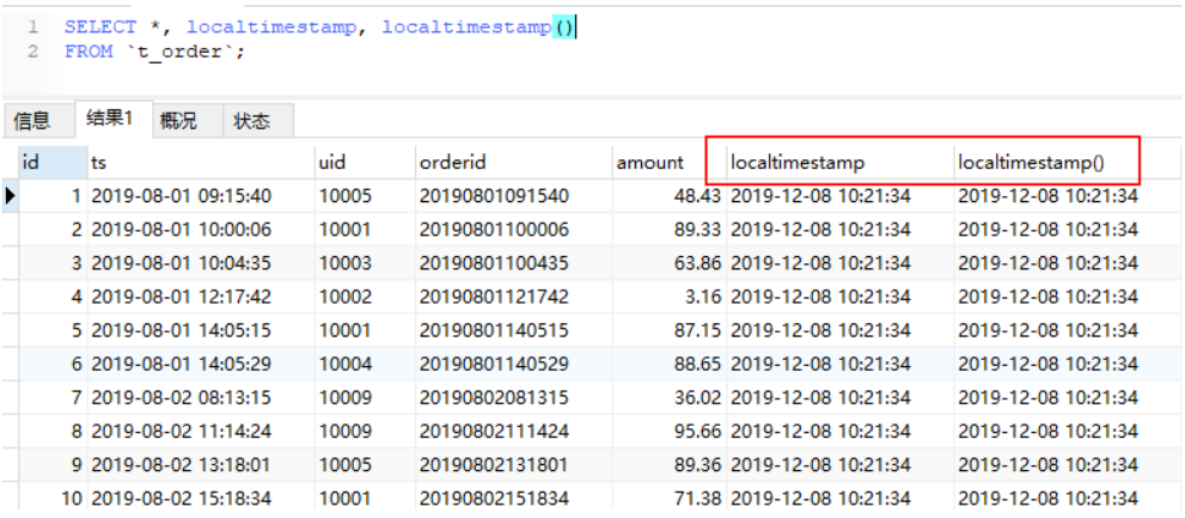
```sql
SELECT *, now(), current_timestamp(), current_timestamp
FROM `t_order`;
```

id	ts	uid	orderid	amount	now()	current_timestamp()	current_timestamp
1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-12-08 10:19:21	2019-12-08 10:19:21	2019-12-08 10:19:21
2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-12-08 10:19:21	2019-12-08 10:19:21	2019-12-08 10:19:21
3	2019-08-01 10:04:35	10003	20190801100435	63.86	2019-12-08 10:19:21	2019-12-08 10:19:21	2019-12-08 10:19:21
4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-12-08 10:19:21	2019-12-08 10:19:21	2019-12-08 10:19:21
5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-12-08 10:19:21	2019-12-08 10:19:21	2019-12-08 10:19:21
6	2019-08-01 14:05:29	10004	20190801140529	88.65	2019-12-08 10:19:21	2019-12-08 10:19:21	2019-12-08 10:19:21
7	2019-08-02 08:13:15	10009	20190802081315	36.02	2019-12-08 10:19:21	2019-12-08 10:19:21	2019-12-08 10:19:21
8	2019-08-02 11:14:24	10009	20190802111424	95.66	2019-12-08 10:19:21	2019-12-08 10:19:21	2019-12-08 10:19:21
9	2019-08-02 13:18:01	10005	20190802131801	89.36	2019-12-08 10:19:21	2019-12-08 10:19:21	2019-12-08 10:19:21
10	2019-08-02 15:18:34	10001	20190802151834	71.38	2019-12-08 10:19:21	2019-12-08 10:19:21	2019-12-08 10:19:21



```sql
SELECT *, sysdate(), localtime(), localtime
FROM `t_order`;
```

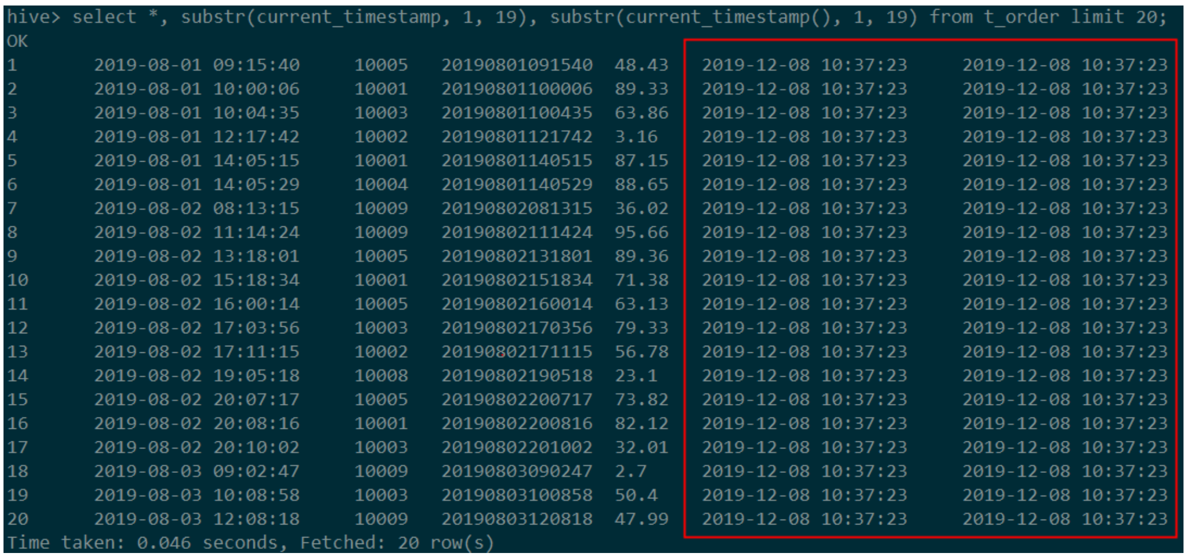
id	ts	uid	orderid	amount	sysdate()	localtime()	localtime
1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-12-08 10:20:42	2019-12-08 10:20:42	2019-12-08 10:20:42
2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-12-08 10:20:42	2019-12-08 10:20:42	2019-12-08 10:20:42
3	2019-08-01 10:04:35	10003	20190801100435	63.86	2019-12-08 10:20:42	2019-12-08 10:20:42	2019-12-08 10:20:42
4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-12-08 10:20:42	2019-12-08 10:20:42	2019-12-08 10:20:42
5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-12-08 10:20:42	2019-12-08 10:20:42	2019-12-08 10:20:42
6	2019-08-01 14:05:29	10004	20190801140529	88.65	2019-12-08 10:20:42	2019-12-08 10:20:42	2019-12-08 10:20:42
7	2019-08-02 08:13:15	10009	20190802081315	36.02	2019-12-08 10:20:42	2019-12-08 10:20:42	2019-12-08 10:20:42
8	2019-08-02 11:14:24	10009	20190802111424	95.66	2019-12-08 10:20:42	2019-12-08 10:20:42	2019-12-08 10:20:42
9	2019-08-02 13:18:01	10005	20190802131801	89.36	2019-12-08 10:20:42	2019-12-08 10:20:42	2019-12-08 10:20:42
10	2019-08-02 15:18:34	10001	20190802151834	71.38	2019-12-08 10:20:42	2019-12-08 10:20:42	2019-12-08 10:20:42



```sql
SELECT *, localtimestamp, localtimestamp()
FROM `t_order`;
```

id	ts	uid	orderid	amount	localtimestamp	localtimestamp()
1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-12-08 10:21:34	2019-12-08 10:21:34
2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-12-08 10:21:34	2019-12-08 10:21:34
3	2019-08-01 10:04:35	10003	20190801100435	63.86	2019-12-08 10:21:34	2019-12-08 10:21:34
4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-12-08 10:21:34	2019-12-08 10:21:34
5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-12-08 10:21:34	2019-12-08 10:21:34
6	2019-08-01 14:05:29	10004	20190801140529	88.65	2019-12-08 10:21:34	2019-12-08 10:21:34
7	2019-08-02 08:13:15	10009	20190802081315	36.02	2019-12-08 10:21:34	2019-12-08 10:21:34
8	2019-08-02 11:14:24	10009	20190802111424	95.66	2019-12-08 10:21:34	2019-12-08 10:21:34
9	2019-08-02 13:18:01	10005	20190802131801	89.36	2019-12-08 10:21:34	2019-12-08 10:21:34
10	2019-08-02 15:18:34	10001	20190802151834	71.38	2019-12-08 10:21:34	2019-12-08 10:21:34

hive中获取当前时间，可以使用 current_timestamp()， current_timestamp，得到的是带有毫秒的，如果想保持和上面同样的格式，需要使用字符串截取一下。如下图所示：



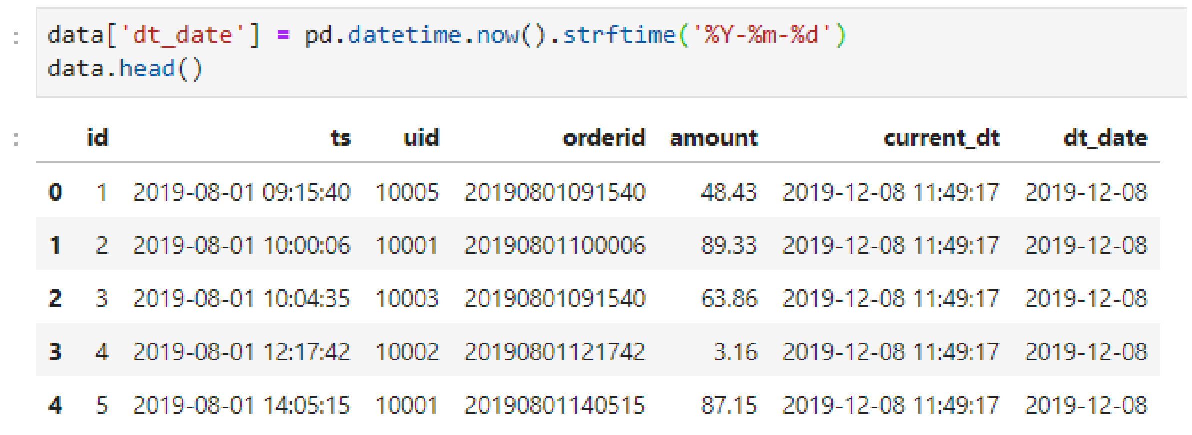
```
hive> select *, substr(current_timestamp, 1, 19), substr(current_timestamp(), 1, 19) from t_order limit 20;
OK
1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-12-08 10:37:23	2019-12-08 10:37:23
2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-12-08 10:37:23	2019-12-08 10:37:23
3	2019-08-01 10:04:35	10003	20190801100435	63.86	2019-12-08 10:37:23	2019-12-08 10:37:23
4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-12-08 10:37:23	2019-12-08 10:37:23
5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-12-08 10:37:23	2019-12-08 10:37:23
6	2019-08-01 14:05:29	10004	20190801140529	88.65	2019-12-08 10:37:23	2019-12-08 10:37:23
7	2019-08-02 08:13:15	10009	20190802081315	36.02	2019-12-08 10:37:23	2019-12-08 10:37:23
8	2019-08-02 11:14:24	10009	20190802111424	95.66	2019-12-08 10:37:23	2019-12-08 10:37:23
9	2019-08-02 13:18:01	10005	20190802131801	89.36	2019-12-08 10:37:23	2019-12-08 10:37:23
10	2019-08-02 15:18:34	10001	20190802151834	71.38	2019-12-08 10:37:23	2019-12-08 10:37:23
11	2019-08-02 16:00:14	10005	20190802160014	63.13	2019-12-08 10:37:23	2019-12-08 10:37:23
12	2019-08-02 17:03:56	10003	20190802170356	79.33	2019-12-08 10:37:23	2019-12-08 10:37:23
13	2019-08-02 17:11:15	10002	20190802171115	56.78	2019-12-08 10:37:23	2019-12-08 10:37:23
14	2019-08-02 19:05:18	10008	20190802190518	23.1	2019-12-08 10:37:23	2019-12-08 10:37:23
15	2019-08-02 20:07:17	10005	20190802200717	73.82	2019-12-08 10:37:23	2019-12-08 10:37:23
16	2019-08-02 20:08:16	10001	20190802200816	82.12	2019-12-08 10:37:23	2019-12-08 10:37:23
17	2019-08-02 20:10:02	10003	20190802201002	32.01	2019-12-08 10:37:23	2019-12-08 10:37:23
18	2019-08-03 09:02:47	10009	20190803090247	2.7	2019-12-08 10:37:23	2019-12-08 10:37:23
19	2019-08-03 10:08:58	10003	20190803100858	50.4	2019-12-08 10:37:23	2019-12-08 10:37:23
20	2019-08-03 12:08:18	10009	20190803120818	47.99	2019-12-08 10:37:23	2019-12-08 10:37:23
Time taken: 0.046 seconds, Fetched: 20 row(s)
```

图中代码：

```
#pandas
data['current_dt'] = pd.datetime.now()
data['current_dt'] = data['current_dt'].apply(lambda x : x.strftime('%Y-%m-%d
%H:%M:%S'))
data.head()
#也可以data['current_dt'] = pd.datetime.now().strftime('%Y-%m-%d %H:%M:%S')一步到
位

#MySQL
SELECT *, now(),current_timestamp(),current_timestamp
FROM `t_order`;
SELECT *, sysdate(),ocaltime(),localtime
FROM `t_order`;
SELECT *, localtimestamp, localtimestamp()
FROM `t_order`;

#HiveQL
select *, substr(current_timestamp, 1, 19), substr(current_timestamp(), 1, 19)
from t_order limit 20;
```

2.获取当前时间，年月日

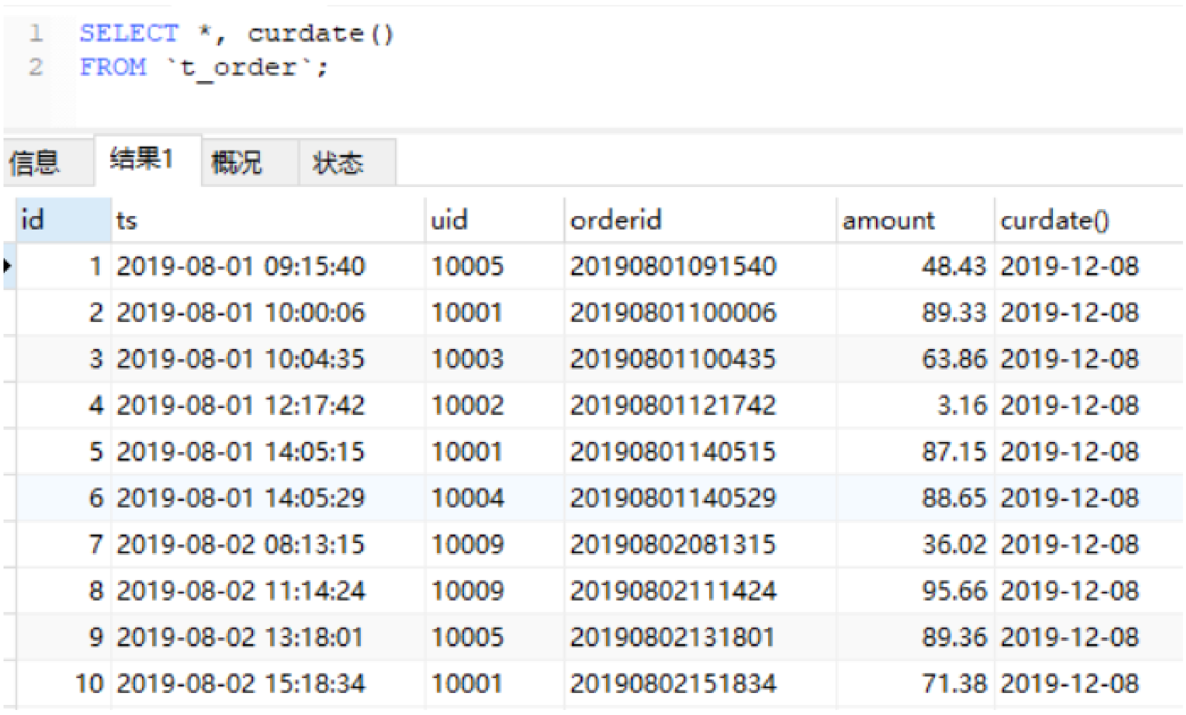
pandas中似乎没有直接获取当前日期的方法，我们沿用上一小节中思路，进行格式转换得到当前日期。当然这不代表python中的其他模块不能实现，有兴趣的朋友可以自己查阅相关文档。



```
data['dt_date'] = pd.datetime.now().strftime('%Y-%m-%d')
data.head()
```

	id	ts	uid	orderid	amount	current_dt	dt_date
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-12-08 11:49:17	2019-12-08
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-12-08 11:49:17	2019-12-08
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-12-08 11:49:17	2019-12-08
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-12-08 11:49:17	2019-12-08
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-12-08 11:49:17	2019-12-08

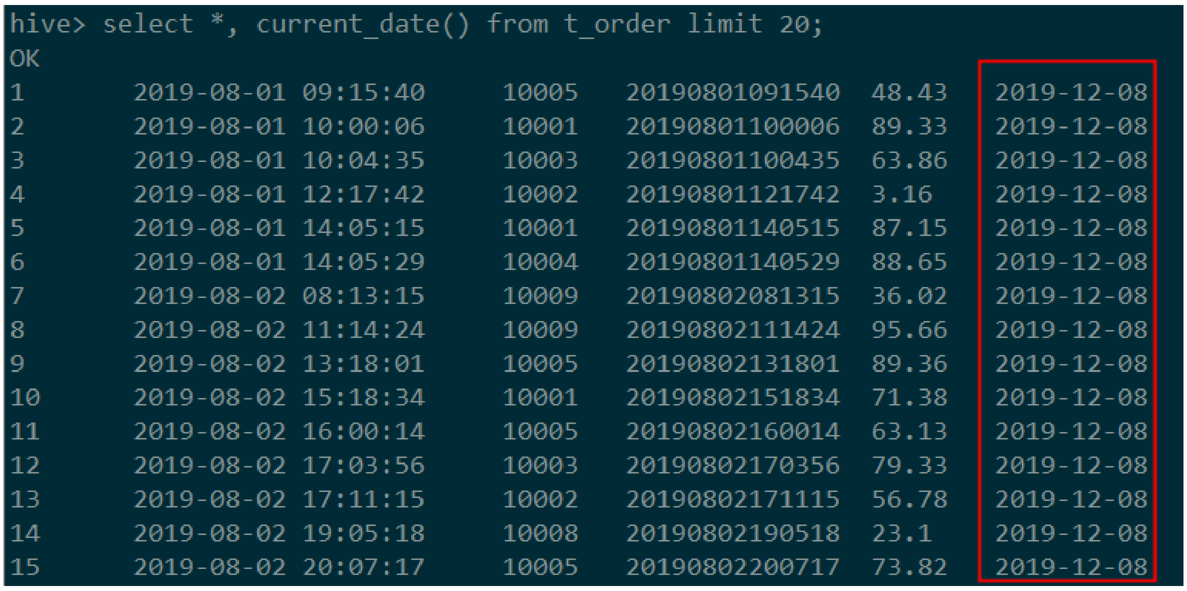
MySQL中可以直接获取当前日期，使用curdate()即可，hive中也有相对应的函数：current_date()。



```
SELECT *, curdate()
FROM `t_order`;
```

信息 结果1 概况 状态

id	ts	uid	orderid	amount	curdate()
1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-12-08
2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-12-08
3	2019-08-01 10:04:35	10003	20190801100435	63.86	2019-12-08
4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-12-08
5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-12-08
6	2019-08-01 14:05:29	10004	20190801140529	88.65	2019-12-08
7	2019-08-02 08:13:15	10009	20190802081315	36.02	2019-12-08
8	2019-08-02 11:14:24	10009	20190802111424	95.66	2019-12-08
9	2019-08-02 13:18:01	10005	20190802131801	89.36	2019-12-08
10	2019-08-02 15:18:34	10001	20190802151834	71.38	2019-12-08



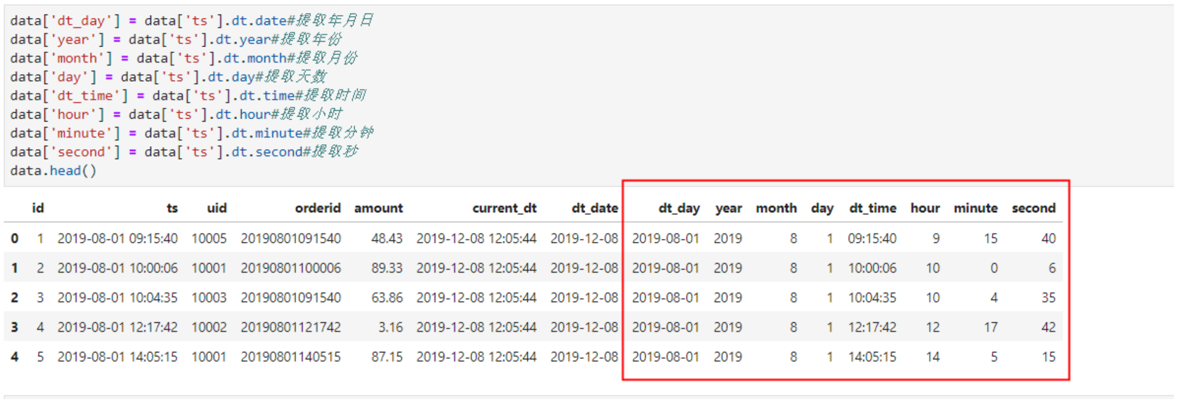
```
hive> select *, current_date() from t_order limit 20;
OK
1       2019-08-01 09:15:40     10005   20190801091540  48.43   2019-12-08
2       2019-08-01 10:00:06     10001   20190801100006  89.33   2019-12-08
3       2019-08-01 10:04:35     10003   20190801100435  63.86   2019-12-08
4       2019-08-01 12:17:42     10002   20190801121742  3.16    2019-12-08
5       2019-08-01 14:05:15     10001   20190801140515  87.15   2019-12-08
6       2019-08-01 14:05:29     10004   20190801140529  88.65   2019-12-08
7       2019-08-02 08:13:15     10009   20190802081315  36.02   2019-12-08
8       2019-08-02 11:14:24     10009   20190802111424  95.66   2019-12-08
9       2019-08-02 13:18:01     10005   20190802131801  89.36   2019-12-08
10      2019-08-02 15:18:34     10001   20190802151834  71.38   2019-12-08
11      2019-08-02 16:00:14     10005   20190802160014  63.13   2019-12-08
12      2019-08-02 17:03:56     10003   20190802170356  79.33   2019-12-08
13      2019-08-02 17:11:15     10002   20190802171115  56.78   2019-12-08
14      2019-08-02 19:05:18     10008   20190802190518  23.1    2019-12-08
15      2019-08-02 20:07:17     10005   20190802200717  73.82   2019-12-08
```

图片中的代码：

```
#pandas
data['dt_date'] = pd.datetime.now().strftime('%Y-%m-%d')
data.head()

#MySQL
SELECT *, curdate() FROM `t_order`;

#HiveQL
select *, current_date() from t_order limit 20;
```

3.提取日期中的相关信息

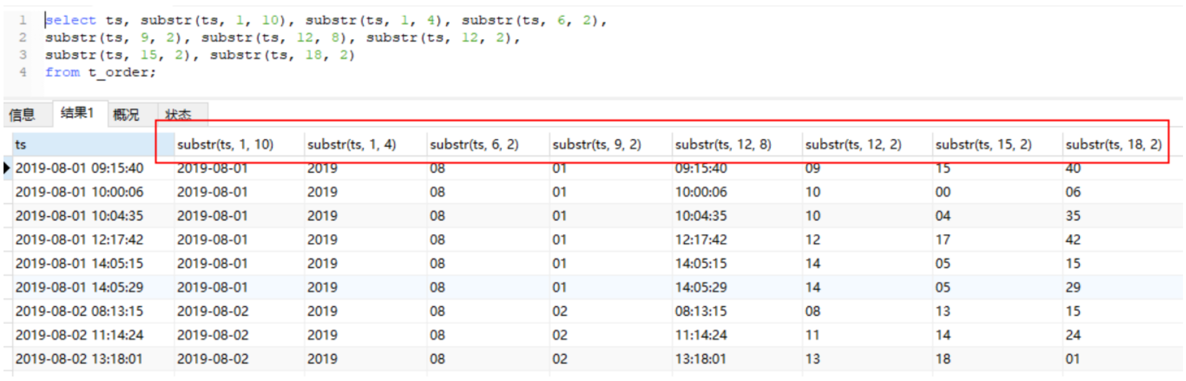
日期中包含有年月日时分秒，我们可以用相应的函数进行分别提取。下面我们提取一下ts字段中的天，时间年月日时分秒信息。



```
data['dt_day'] = data['ts'].dt.date#提取年月日
data['year'] = data['ts'].dt.year#提取年份
data['month'] = data['ts'].dt.month#提取月份
data['day'] = data['ts'].dt.day#提取天数
data['dt_time'] = data['ts'].dt.time#提取时间
data['hour'] = data['ts'].dt.hour#提取小时
data['minute'] = data['ts'].dt.minute#提取分钟
data['second'] = data['ts'].dt.second#提取秒
data.head()
```

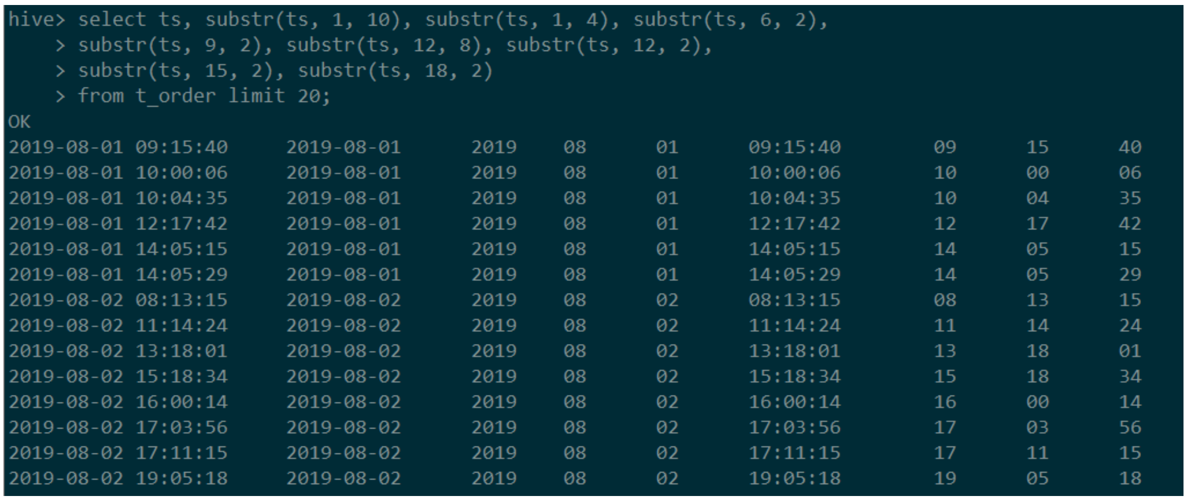
	id	ts	uid	orderid	amount	current_dt	dt_date	dt_day	year	month	day	dt_time	hour	minute	second
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-12-08 12:05:44	2019-12-08	2019-08-01	2019	8	1	09:15:40	9	15	40
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-12-08 12:05:44	2019-12-08	2019-08-01	2019	8	1	10:00:06	10	0	6
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-12-08 12:05:44	2019-12-08	2019-08-01	2019	8	1	10:04:35	10	4	35
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-12-08 12:05:44	2019-12-08	2019-08-01	2019	8	1	12:17:42	12	17	42
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-12-08 12:05:44	2019-12-08	2019-08-01	2019	8	1	14:05:15	14	5	15

在MySQL和Hive中，由于ts字段是字符串格式存储的，我们只需使用字符串截取函数即可。两者的代码是一样的，只需要注意截取的位置和长度即可，效果如下：



```
select ts, substr(ts, 1, 10), substr(ts, 1, 4), substr(ts, 6, 2),
substr(ts, 9, 2), substr(ts, 12, 8), substr(ts, 12, 2),
substr(ts, 15, 2), substr(ts, 18, 2)
from t_order;
```

ts	substr(ts, 1, 10)	substr(ts, 1, 4)	substr(ts, 6, 2)	substr(ts, 9, 2)	substr(ts, 12, 8)	substr(ts, 12, 2)	substr(ts, 15, 2)	substr(ts, 18, 2)
2019-08-01 09:15:40	2019-08-01	2019	08	01	09:15:40	09	15	40
2019-08-01 10:00:06	2019-08-01	2019	08	01	10:00:06	10	00	06
2019-08-01 10:04:35	2019-08-01	2019	08	01	10:04:35	10	04	35
2019-08-01 12:17:42	2019-08-01	2019	08	01	12:17:42	12	17	42
2019-08-01 14:05:15	2019-08-01	2019	08	01	14:05:15	14	05	15
2019-08-01 14:05:29	2019-08-01	2019	08	01	14:05:29	14	05	29
2019-08-02 08:13:15	2019-08-02	2019	08	02	08:13:15	08	13	15
2019-08-02 11:14:24	2019-08-02	2019	08	02	11:14:24	11	14	24
2019-08-02 13:18:01	2019-08-02	2019	08	02	13:18:01	13	18	01



```
hive> select ts, substr(ts, 1, 10), substr(ts, 1, 4), substr(ts, 6, 2),
    > substr(ts, 9, 2), substr(ts, 12, 8), substr(ts, 12, 2),
    > substr(ts, 15, 2), substr(ts, 18, 2)
    > from t_order limit 20;
OK
2019-08-01 09:15:40     2019-08-01      2019    08      01      09:15:40        09      15      40
2019-08-01 10:00:06     2019-08-01      2019    08      01      10:00:06        10      00      06
2019-08-01 10:04:35     2019-08-01      2019    08      01      10:04:35        10      04      35
2019-08-01 12:17:42     2019-08-01      2019    08      01      12:17:42        12      17      42
2019-08-01 14:05:15     2019-08-01      2019    08      01      14:05:15        14      05      15
2019-08-01 14:05:29     2019-08-01      2019    08      01      14:05:29        14      05      29
2019-08-02 08:13:15     2019-08-02      2019    08      02      08:13:15        08      13      15
2019-08-02 11:14:24     2019-08-02      2019    08      02      11:14:24        11      14      24
2019-08-02 13:18:01     2019-08-02      2019    08      02      13:18:01        13      18      01
2019-08-02 15:18:34     2019-08-02      2019    08      02      15:18:34        15      18      34
2019-08-02 16:00:14     2019-08-02      2019    08      02      16:00:14        16      00      14
2019-08-02 17:03:56     2019-08-02      2019    08      02      17:03:56        17      03      56
2019-08-02 17:11:15     2019-08-02      2019    08      02      17:11:15        17      11      15
2019-08-02 19:05:18     2019-08-02      2019    08      02      19:05:18        19      05      18
```

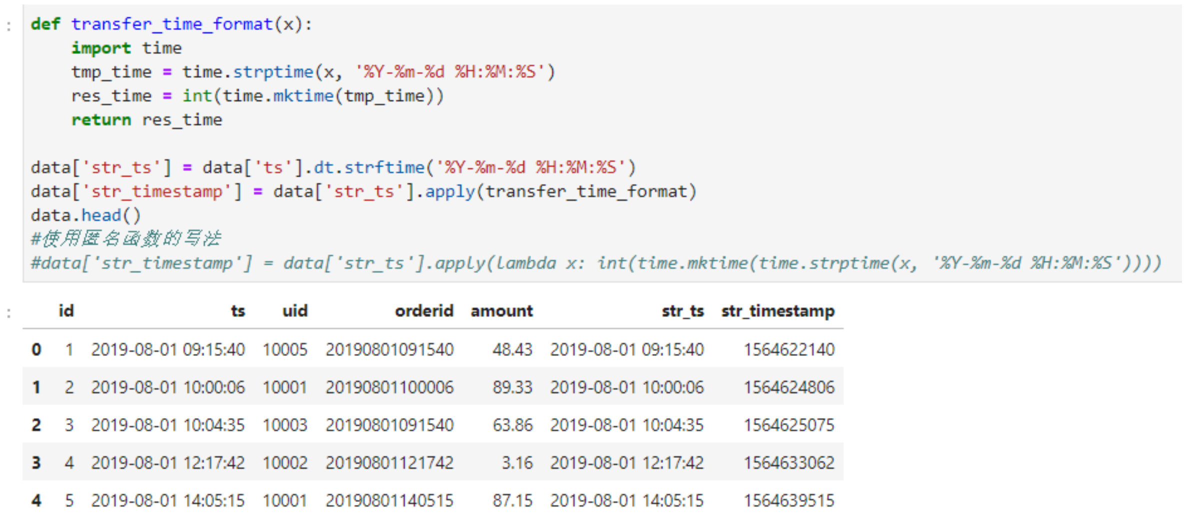
图片中代码：

```
#pandas
data['dt_day'] = data['ts'].dt.date#提取年月日
data['year'] = data['ts'].dt.year#提取年份
data['month'] = data['ts'].dt.month#提取月份
data['day'] = data['ts'].dt.day#提取天数
data['dt_time'] = data['ts'].dt.time#提取时间
data['hour'] = data['ts'].dt.hour#提取小时
data['minute'] = data['ts'].dt.minute#提取分钟
data['second'] = data['ts'].dt.second#提取秒
data.head()

#MySQL
select ts, substr(ts, 1, 10), substr(ts, 1, 4), substr(ts, 6, 2),
substr(ts, 9, 2), substr(ts, 12, 8), substr(ts, 12, 2),
substr(ts, 15, 2), substr(ts, 18, 2)
from t_order;

#HiveQL
select ts, substr(ts, 1, 10), substr(ts, 1, 4), substr(ts, 6, 2),
substr(ts, 9, 2), substr(ts, 12, 8), substr(ts, 12, 2),
substr(ts, 15, 2), substr(ts, 18, 2)
from t_order limit 20;
```

日期转换

1.可读日期转换为unix时间戳

在pandas中，我找到的方法是先`datetime64[ns]`转换位字符串，再调用time模块来实现，代码如下：



```python
def transfer_time_format(x):
    import time
    tmp_time = time.strptime(x, '%Y-%m-%d %H:%M:%S')
    res_time = int(time.mktime(tmp_time))
    return res_time

data['str_ts'] = data['ts'].dt.strftime('%Y-%m-%d %H:%M:%S')
data['str_timestamp'] = data['str_ts'].apply(transfer_time_format)
data.head()
#使用匿名函数的写法
#data['str_timestamp'] = data['str_ts'].apply(lambda x: int(time.mktime(time.strptime(x, '%Y-%m-%d %H:%M:%S'))))
```

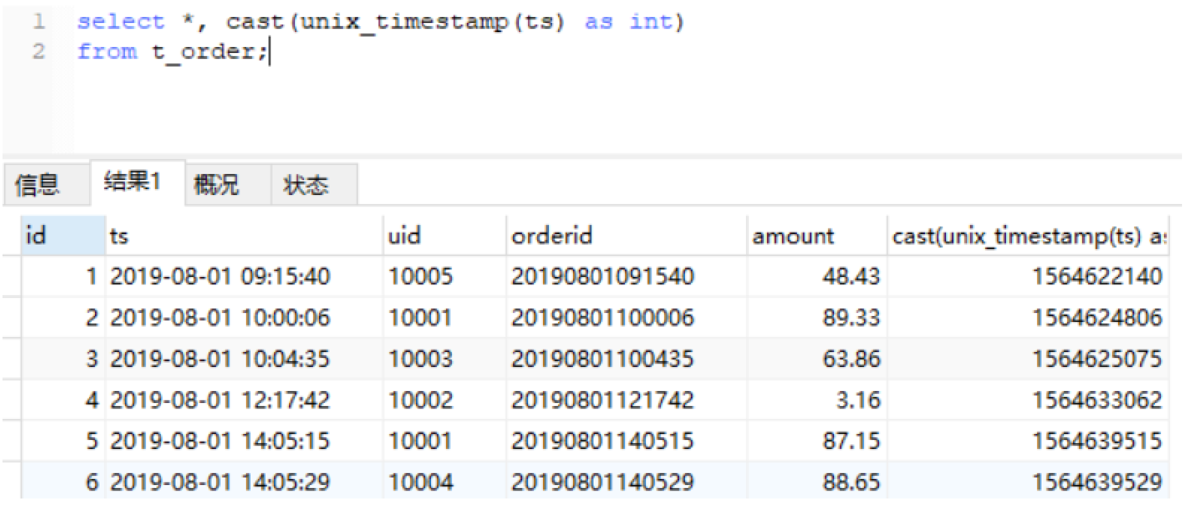
	id	ts	uid	orderid	amount	str_ts	str_timestamp
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-08-01 09:15:40	1564622140
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-08-01 10:00:06	1564624806
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-08-01 10:04:35	1564625075
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-08-01 12:17:42	1564633062
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-08-01 14:05:15	1564639515

可以验证最后一列的十位数字就是ts的时间戳形式。

ps.在此之前，我尝试了另外一种借助numpy的方式，进行类型的转换，但转出来结果不正确，我写在这里，欢迎有经验的读者指正。

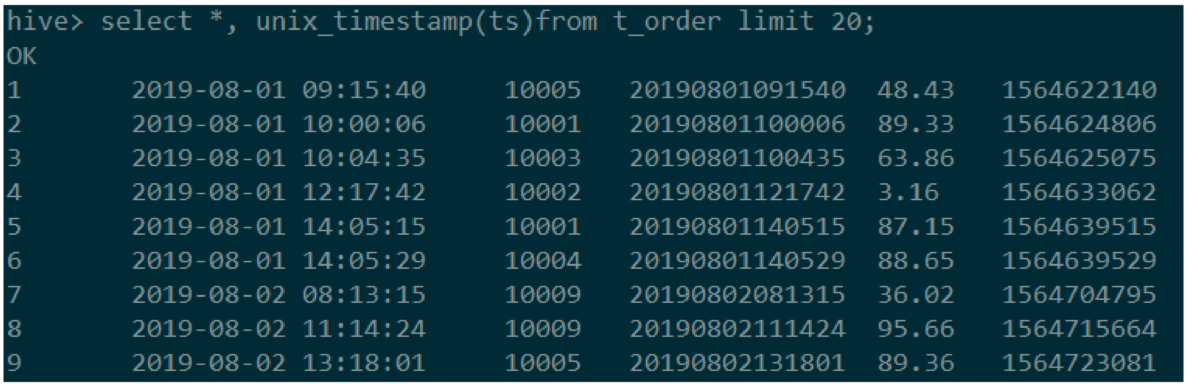
```
import numpy as np
data['ts_timestamp'] = (data.ts.astype(np.int64)/1e9).astype(np.int64)
data.head()
#得到的ts_timestamp结果
#1564650940  1564653606  1564653875等刚好比正确的结果多8个小时
```

MySQL和Hive中可以使用时间戳转换函数进行这项操作，其中MySQL需要进行一下类型转换，Hive不需要。



```
select *, cast(unix_timestamp(ts) as int)
from t_order;
```

id	ts	uid	orderid	amount	cast(unix_timestamp(ts) a
1	2019-08-01 09:15:40	10005	20190801091540	48.43	1564622140
2	2019-08-01 10:00:06	10001	20190801100006	89.33	1564624806
3	2019-08-01 10:04:35	10003	20190801100435	63.86	1564625075
4	2019-08-01 12:17:42	10002	20190801121742	3.16	1564633062
5	2019-08-01 14:05:15	10001	20190801140515	87.15	1564639515
6	2019-08-01 14:05:29	10004	20190801140529	88.65	1564639529



```
hive> select *, unix_timestamp(ts)from t_order limit 20;
OK
1       2019-08-01 09:15:40     10005   20190801091540  48.43   1564622140
2       2019-08-01 10:00:06     10001   20190801100006  89.33   1564624806
3       2019-08-01 10:04:35     10003   20190801100435  63.86   1564625075
4       2019-08-01 12:17:42     10002   20190801121742  3.16    1564633062
5       2019-08-01 14:05:15     10001   20190801140515  87.15   1564639515
6       2019-08-01 14:05:29     10004   20190801140529  88.65   1564639529
7       2019-08-02 08:13:15     10009   20190802081315  36.02   1564704795
8       2019-08-02 11:14:24     10009   20190802111424  95.66   1564715664
9       2019-08-02 13:18:01     10005   20190802131801  89.36   1564723081
```

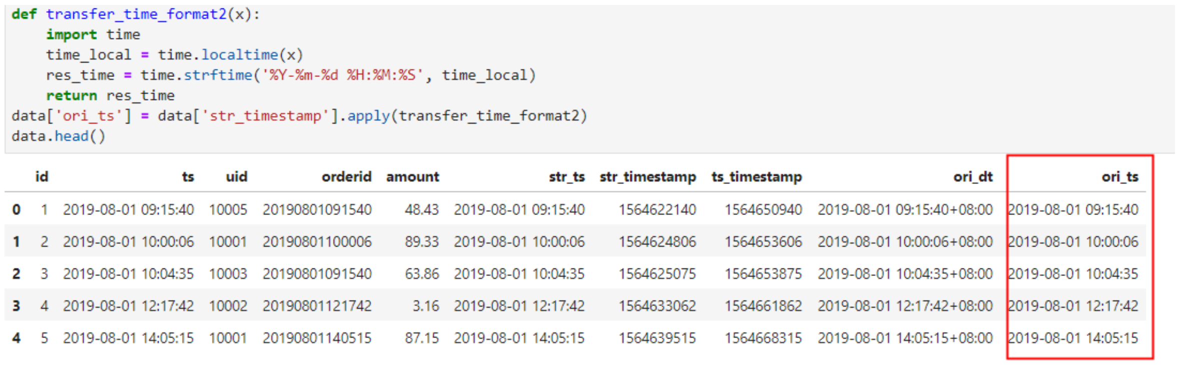
图中代码：

```
#python
def transfer_time_format(x):
    import time
    tmp_time = time.strptime(x, '%Y-%m-%d %H:%M:%S')
    res_time = int(time.mktime(tmp_time))
    return res_time

data['str_ts'] = data['ts'].dt.strftime('%Y-%m-%d %H:%M:%S')
data['str_timestamp'] = data['str_ts'].apply(transfer_time_format)
data.head()
#使用匿名函数的写法
#data['str_timestamp'] = data['str_ts'].apply(lambda x:
int(time.mktime(time.strptime(x, '%Y-%m-%d %H:%M:%S'))))

#MySQL
select *, cast(unix_timestamp(ts) as int)
from t_order;

#Hive
select *, unix_timestamp(ts) from t_order limit 20;
```

2.unix时间戳转为可读日期

这一操作为上一小节的逆向操作。

在pandas中，我们看一下如何将str_timestamp列转换为原来的ts列。这里依然采用time模块中的方法来实现。



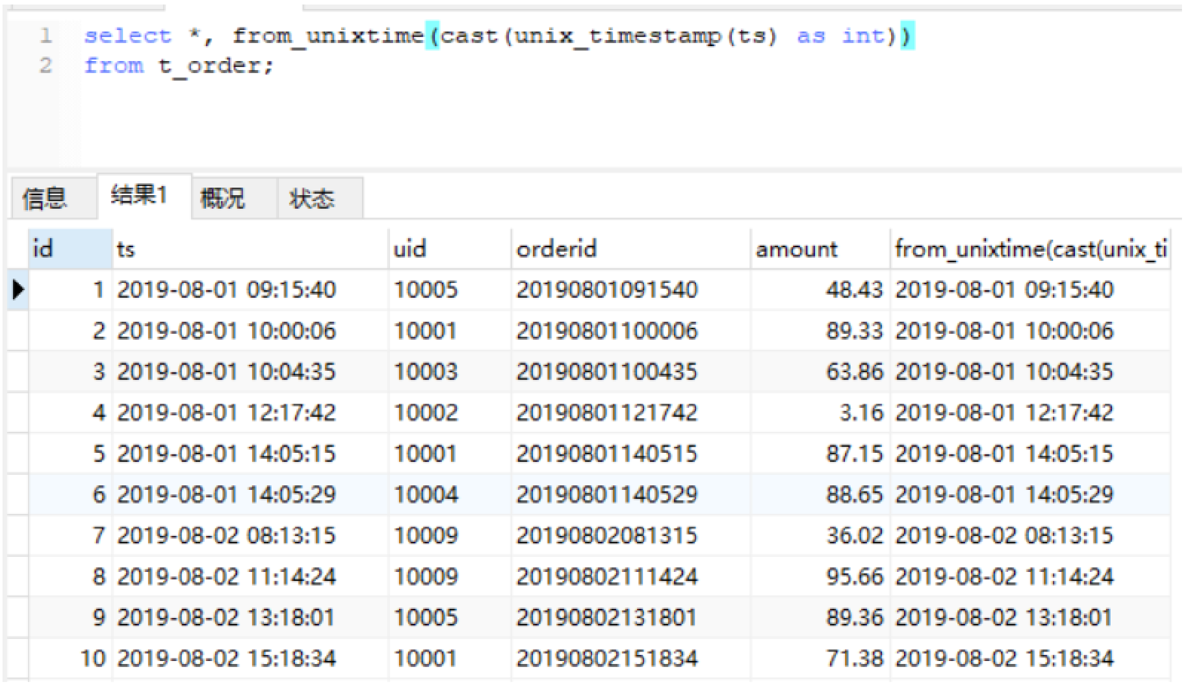
```
def transfer_time_format2(x):
    import time
    time_local = time.localtime(x)
    res_time = time.strftime('%Y-%m-%d %H:%M:%S', time_local)
    return res_time
data['ori_ts'] = data['str_timestamp'].apply(transfer_time_format2)
data.head()
```

	id	ts	uid	orderid	amount	str_ts	str_timestamp	ts_timestamp	ori_dt	ori_ts
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-08-01 09:15:40	1564622140	1564650940	2019-08-01 09:15:40+08:00	2019-08-01 09:15:40
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-08-01 10:00:06	1564624806	1564653606	2019-08-01 10:00:06+08:00	2019-08-01 10:00:06
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-08-01 10:04:35	1564625075	1564653875	2019-08-01 10:04:35+08:00	2019-08-01 10:04:35
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-08-01 12:17:42	1564633062	1564661862	2019-08-01 12:17:42+08:00	2019-08-01 12:17:42
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-08-01 14:05:15	1564639515	1564668315	2019-08-01 14:05:15+08:00	2019-08-01 14:05:15

ps.你可能发现了上面代码中有一列是ori_dt，虽然看上去是正确的，但格式多少有那么点奇怪，这也是我在学习过程中看到的一个不那么正确的写法，贴出来供大家思考。

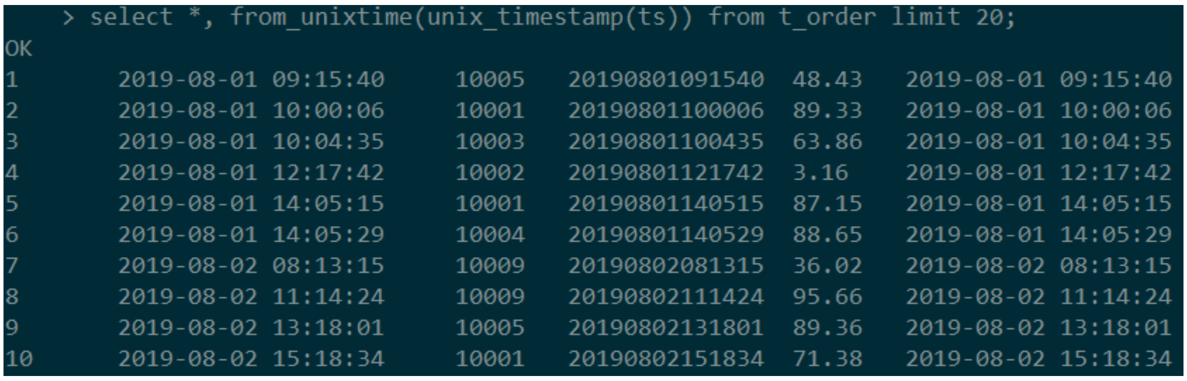
```
data['ori_dt'] = pd.to_datetime(data['str_timestamp'].values, unit='s',
utc=True).tz_convert('Asia/Shanghai')
data.head()
#使用默认的pd.to_datetime并不能转会正确的时间，比实际时间小8个小时
#在网上看到了这种写法能把8个小时加回来，但显示的很奇怪。
```

回到MySQL和Hive，依然只是用一个函数就解决了。



```
select *, from_unixtime(cast(unix_timestamp(ts) as int))
from t_order;
```

id	ts	uid	orderid	amount	from_unixtime(cast(unix_ti
1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-08-01 09:15:40
2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-08-01 10:00:06
3	2019-08-01 10:04:35	10003	20190801100435	63.86	2019-08-01 10:04:35
4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-08-01 12:17:42
5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-08-01 14:05:15
6	2019-08-01 14:05:29	10004	20190801140529	88.65	2019-08-01 14:05:29
7	2019-08-02 08:13:15	10009	20190802081315	36.02	2019-08-02 08:13:15
8	2019-08-02 11:14:24	10009	20190802111424	95.66	2019-08-02 11:14:24
9	2019-08-02 13:18:01	10005	20190802131801	89.36	2019-08-02 13:18:01
10	2019-08-02 15:18:34	10001	20190802151834	71.38	2019-08-02 15:18:34



```
    > select *, from_unixtime(unix_timestamp(ts)) from t_order limit 20;
OK
1       2019-08-01 09:15:40     10005   20190801091540  48.43   2019-08-01 09:15:40
2       2019-08-01 10:00:06     10001   20190801100006  89.33   2019-08-01 10:00:06
3       2019-08-01 10:04:35     10003   20190801100435  63.86   2019-08-01 10:04:35
4       2019-08-01 12:17:42     10002   20190801121742  3.16    2019-08-01 12:17:42
5       2019-08-01 14:05:15     10001   20190801140515  87.15   2019-08-01 14:05:15
6       2019-08-01 14:05:29     10004   20190801140529  88.65   2019-08-01 14:05:29
7       2019-08-02 08:13:15     10009   20190802081315  36.02   2019-08-02 08:13:15
8       2019-08-02 11:14:24     10009   20190802111424  95.66   2019-08-02 11:14:24
9       2019-08-02 13:18:01     10005   20190802131801  89.36   2019-08-02 13:18:01
10      2019-08-02 15:18:34     10001   20190802151834  71.38   2019-08-02 15:18:34
```

图中代码如下：

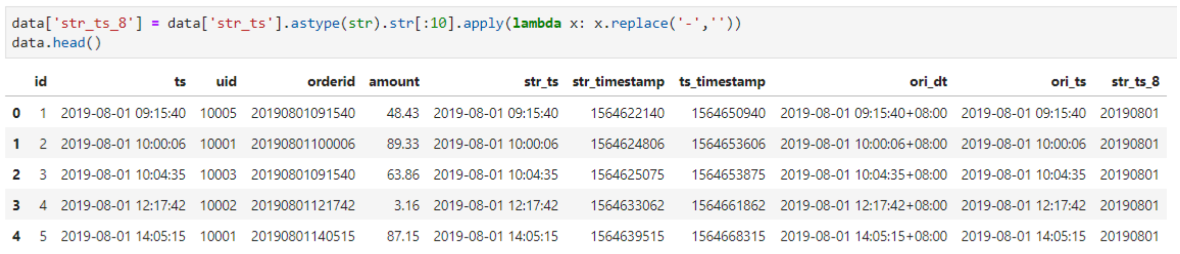
```
#pandas:
```

```
def transfer_time_format2(x):
    import time
    time_local = time.localtime(x)
    res_time = time.strftime('%Y-%m-%d %H:%M:%S', time_local)
    return res_time
data['ori_ts'] = data['str_timestamp'].apply(transfer_time_format2)
data.head()

#MySQL
select *, from_unixtime(cast(unix_timestamp(ts) as int))
from t_order;

#Hive
select *, from_unixtime(unix_timestamp(ts)) from t_order limit 20;
```

3.10位日期转8位

对于初始是ts列这样年月日时分秒的形式，我们通常需要先转换为10位年月日的格式，再把中间的横杠替换掉，就可以得到8位的日期了。

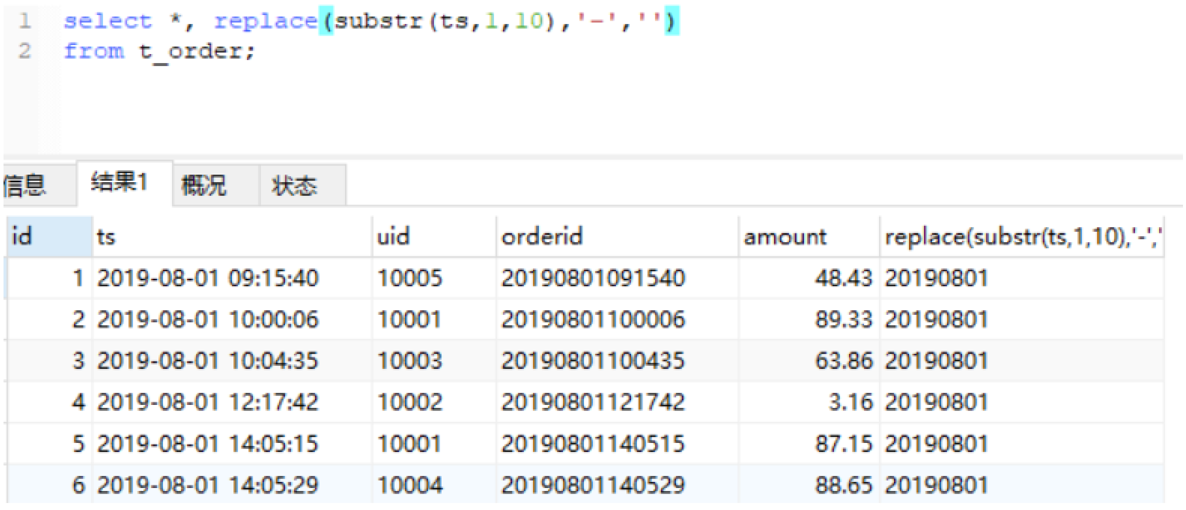
由于打算使用字符串替换，我们先要将ts转换为字符串的形式，在前面的转换中，我们生成了一列str_ts，该列的数据类型是object，相当于字符串，可以在此基础上进行这里的转换。



```
data['str_ts_8'] = data['str_ts'].astype(str).str[:10].apply(lambda x: x.replace('-',''))
data.head()
```

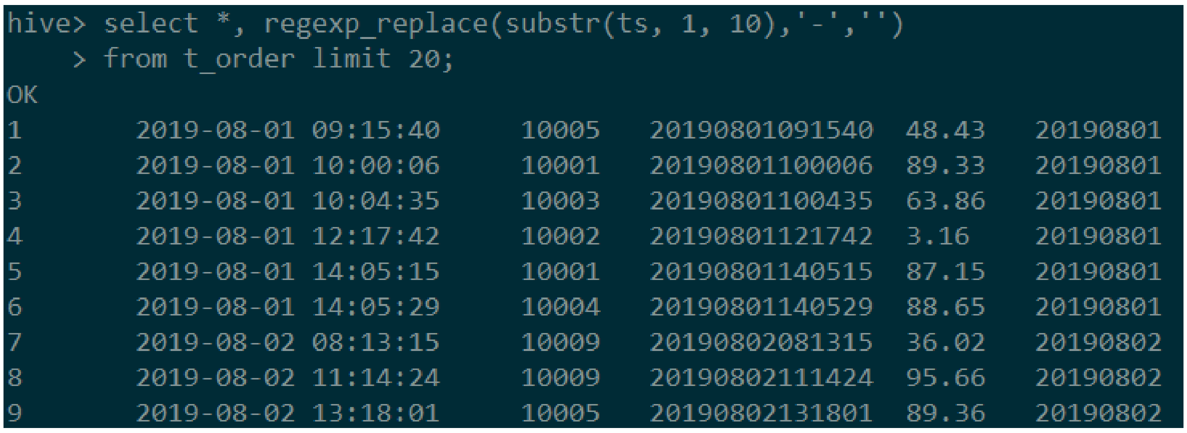
	id	ts	uid	orderid	amount	str_ts	str_timestamp	ts_timestamp	ori_dt	ori_ts	str_ts_8
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-08-01 09:15:40	1564622140	1564650940	2019-08-01 09:15:40+08:00	2019-08-01 09:15:40	20190801
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-08-01 10:00:06	1564624806	1564653606	2019-08-01 10:00:06+08:00	2019-08-01 10:00:06	20190801
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-08-01 10:04:35	1564625075	1564653875	2019-08-01 10:04:35+08:00	2019-08-01 10:04:35	20190801
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-08-01 12:17:42	1564633062	1564661862	2019-08-01 12:17:42+08:00	2019-08-01 12:17:42	20190801
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-08-01 14:05:15	1564639515	1564668315	2019-08-01 14:05:15+08:00	2019-08-01 14:05:15	20190801

MySQL和Hive中也是同样的套路，截取和替换几乎是最简便的方法了。



```
select *, replace(substr(ts,1,10),'-','')
from t_order;
```

id	ts	uid	orderid	amount	replace(substr(ts,1,10),'-','
1	2019-08-01 09:15:40	10005	20190801091540	48.43	20190801
2	2019-08-01 10:00:06	10001	20190801100006	89.33	20190801
3	2019-08-01 10:04:35	10003	20190801100435	63.86	20190801
4	2019-08-01 12:17:42	10002	20190801121742	3.16	20190801
5	2019-08-01 14:05:15	10001	20190801140515	87.15	20190801
6	2019-08-01 14:05:29	10004	20190801140529	88.65	20190801

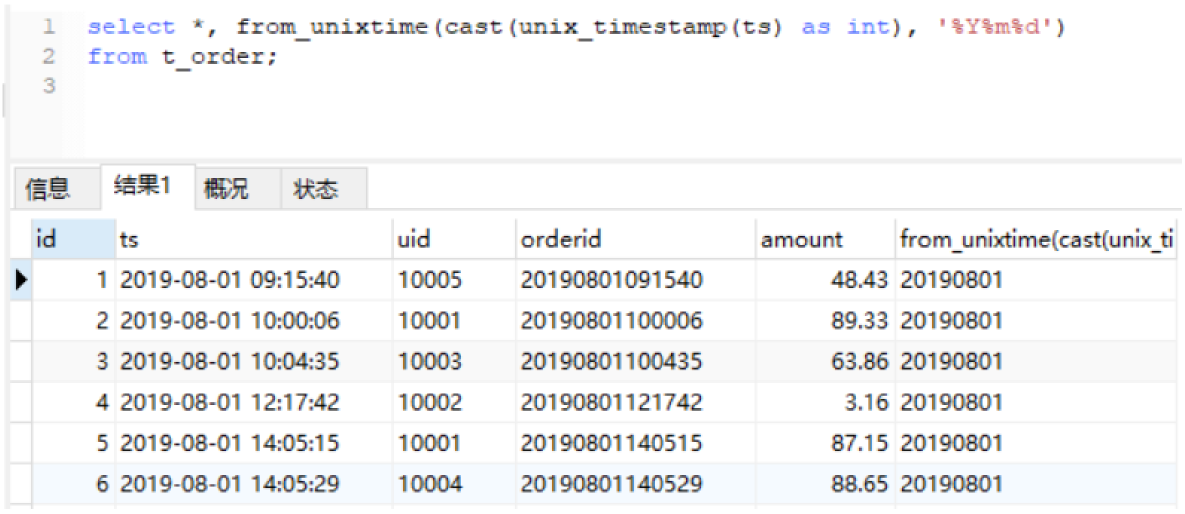


```
hive> select *, regexp_replace(substr(ts, 1, 10),'-','')
    > from t_order limit 20;
OK
1       2019-08-01 09:15:40     10005   20190801091540  48.43   20190801
2       2019-08-01 10:00:06     10001   20190801100006  89.33   20190801
3       2019-08-01 10:04:35     10003   20190801100435  63.86   20190801
4       2019-08-01 12:17:42     10002   20190801121742  3.16    20190801
5       2019-08-01 14:05:15     10001   20190801140515  87.15   20190801
6       2019-08-01 14:05:29     10004   20190801140529  88.65   20190801
7       2019-08-02 08:13:15     10009   20190802081315  36.02   20190802
8       2019-08-02 11:14:24     10009   20190802111424  95.66   20190802
9       2019-08-02 13:18:01     10005   20190802131801  89.36   20190802
```

图中代码：

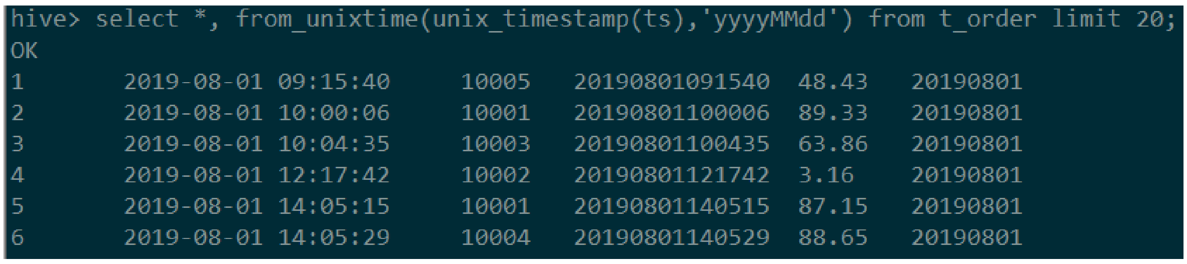
```
#pandas
data['str_ts_8'] = data['str_ts'].astype(str).str[:10].apply(lambda x:
x.replace('-',''))
data.head()

#MySQL
select replace(substr(ts, 1, 10), '-', '')
from t_order;

#Hive
select *, regexp_replace(substr(ts, 1, 10),'-','')
from t_order limit 20;
```

当然，我们也有另外的解法：使用先将字符串转为unix时间戳的形式，再格式化为8位的日期。



```
select *, from_unixtime(cast(unix_timestamp(ts) as int), '%Y%m%d')
from t_order;
```

id	ts	uid	orderid	amount	from_unixtime(cast(unix_ti
1	2019-08-01 09:15:40	10005	20190801091540	48.43	20190801
2	2019-08-01 10:00:06	10001	20190801100006	89.33	20190801
3	2019-08-01 10:04:35	10003	20190801100435	63.86	20190801
4	2019-08-01 12:17:42	10002	20190801121742	3.16	20190801
5	2019-08-01 14:05:15	10001	20190801140515	87.15	20190801
6	2019-08-01 14:05:29	10004	20190801140529	88.65	20190801

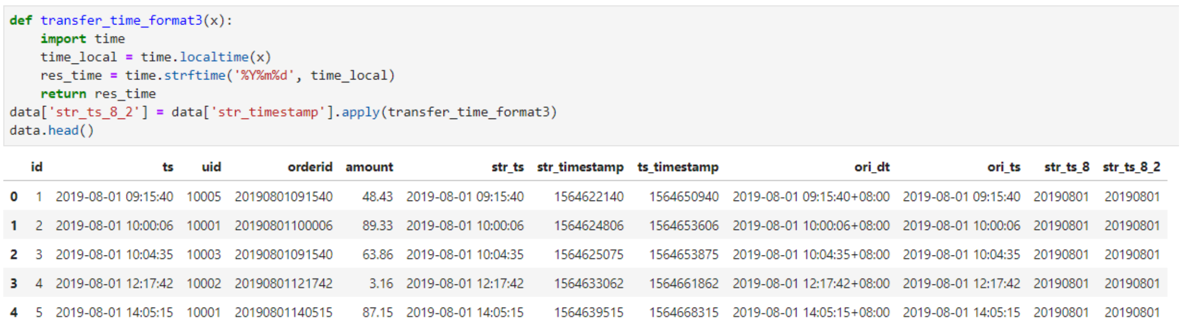


```
hive> select *, from_unixtime(unix_timestamp(ts),'yyyyMMdd') from t_order limit 20;
OK
1       2019-08-01 09:15:40     10005   20190801091540  48.43   20190801
2       2019-08-01 10:00:06     10001   20190801100006  89.33   20190801
3       2019-08-01 10:04:35     10003   20190801100435  63.86   20190801
4       2019-08-01 12:17:42     10002   20190801121742  3.16    20190801
5       2019-08-01 14:05:15     10001   20190801140515  87.15   20190801
6       2019-08-01 14:05:29     10004   20190801140529  88.65   20190801
```

图中代码：

```
#MySQL
select *, from_unixtime(cast(unix_timestamp(ts) as int), '%Y%M%d')
from t_order;

#Hive
select *, from_unixtime(unix_timestamp(ts),'yyyyMMdd') from t_order limit 20;
```

pandas中我们也可以直接在unix时间戳的基础上进行操作，转为8位的日期。具体做法只要上面的transfer_time_format2函数即可，效果如下图所示。



```
def transfer_time_format3(x):
    import time
    time_local = time.localtime(x)
    res_time = time.strftime('%Y%m%d', time_local)
    return res_time
data['str_ts_8_2'] = data['str_timestamp'].apply(transfer_time_format3)
data.head()
```

	id	ts	uid	orderid	amount	str_ts	str_timestamp	ts_timestamp	ori_dt	ori_ts	str_ts_8	str_ts_8_2
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-08-01 09:15:40	1564622140	1564650940	2019-08-01 09:15:40+08:00	2019-08-01 09:15:40	20190801	20190801
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-08-01 10:00:06	1564624806	1564653606	2019-08-01 10:00:06+08:00	2019-08-01 10:00:06	20190801	20190801
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-08-01 10:04:35	1564625075	1564653875	2019-08-01 10:04:35+08:00	2019-08-01 10:04:35	20190801	20190801
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-08-01 12:17:42	1564633062	1564661862	2019-08-01 12:17:42+08:00	2019-08-01 12:17:42	20190801	20190801
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-08-01 14:05:15	1564639515	1564668315	2019-08-01 14:05:15+08:00	2019-08-01 14:05:15	20190801	20190801

```
def transfer_time_format3(x):
    import time
    time_local = time.localtime(x)
    res_time = time.strftime('%Y%m%d', time_local)#改这里的格式就好
    return res_time
data['str_ts_8_2'] = data['str_timestamp'].apply(transfer_time_format3)
data.head()
```

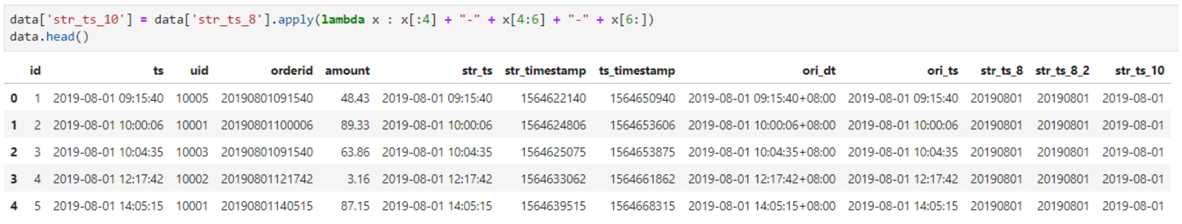
4.8位日期转10位

这一操作同样为上一小节的逆向操作。

结合上一小节，实现10位转8位，我们至少有两种思路。可以进行先截取后拼接，把横线`-`拼接再日期之间即可。二是借助于unix时间戳进行中转。SQL中两种方法都很容易实现，在pandas我们还有另外的方式。

方法一：

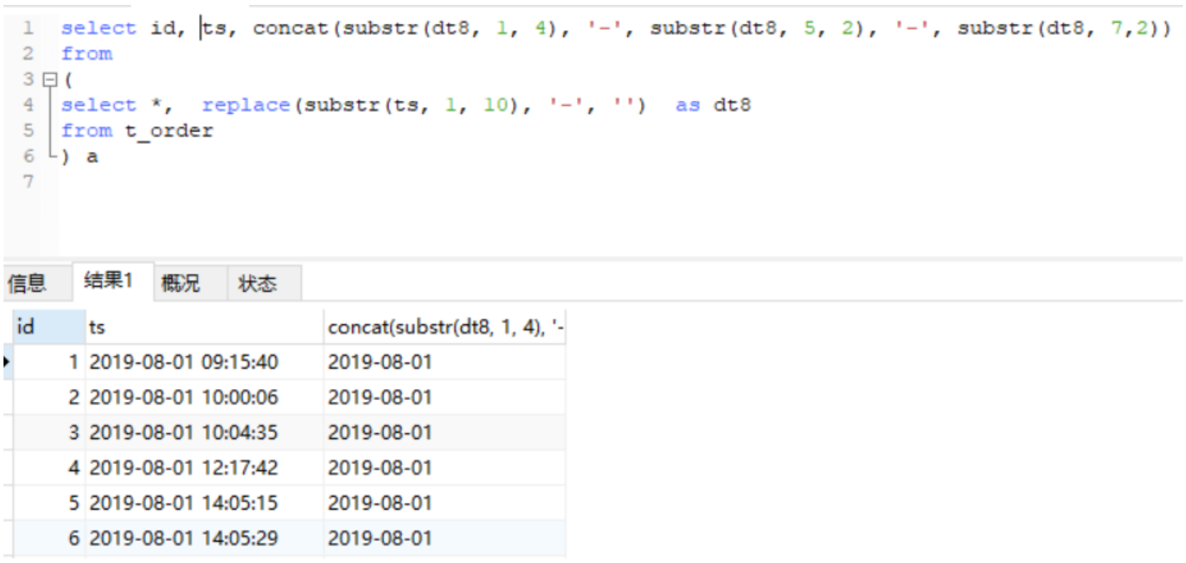
pandas中的拼接也是需要转化为字符串进行。如下：



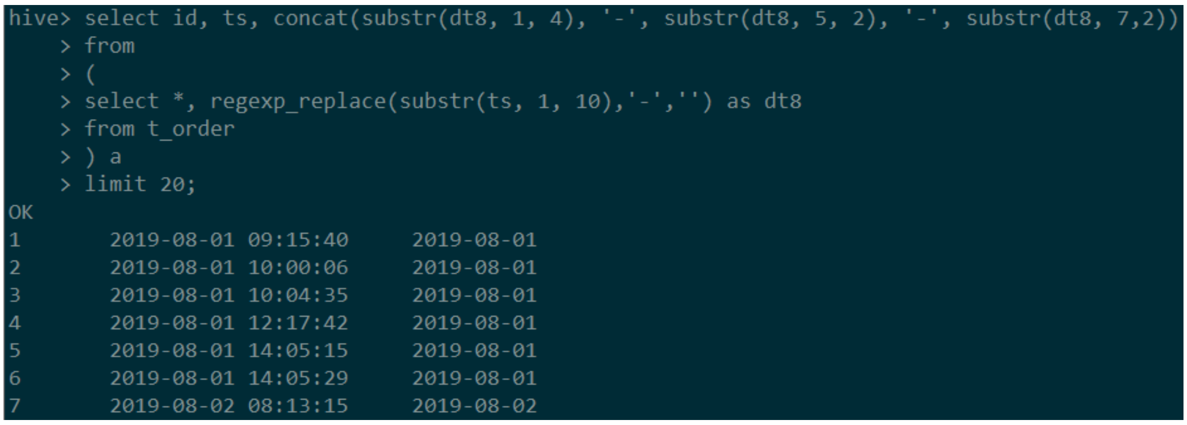
```
data['str_ts_10'] = data['str_ts_8'].apply(lambda x : x[:4] + "-" + x[4:6] + "-" + x[6:])
data.head()
```

	id	ts	uid	orderid	amount	str_ts	str_timestamp	ts_timestamp	ori_dt	ori_ts	str_ts_8	str_ts_8_2	str_ts_10
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-08-01 09:15:40	1564622140	1564650940	2019-08-01 09:15:40+08:00	2019-08-01 09:15:40	20190801	20190801	2019-08-01
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-08-01 10:00:06	1564624806	1564653606	2019-08-01 10:00:06+08:00	2019-08-01 10:00:06	20190801	20190801	2019-08-01
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-08-01 10:04:35	1564625075	1564653875	2019-08-01 10:04:35+08:00	2019-08-01 10:04:35	20190801	20190801	2019-08-01
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-08-01 12:17:42	1564633062	1564661862	2019-08-01 12:17:42+08:00	2019-08-01 12:17:42	20190801	20190801	2019-08-01
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-08-01 14:05:15	1564639515	1564668315	2019-08-01 14:05:15+08:00	2019-08-01 14:05:15	20190801	20190801	2019-08-01

MySQL和Hive中，可以使用concat函数进行拼接：


```
select id, ts, concat(substr(dt8, 1, 4), '-', substr(dt8, 5, 2), '-', substr(dt8, 7,2))
from
(
select *,  replace(substr(ts, 1, 10), '-', '')  as dt8
from t_order
) a
```

id	ts	concat(substr(dt8, 1, 4), '-
1	2019-08-01 09:15:40	2019-08-01
2	2019-08-01 10:00:06	2019-08-01
3	2019-08-01 10:04:35	2019-08-01
4	2019-08-01 12:17:42	2019-08-01
5	2019-08-01 14:05:15	2019-08-01
6	2019-08-01 14:05:29	2019-08-01

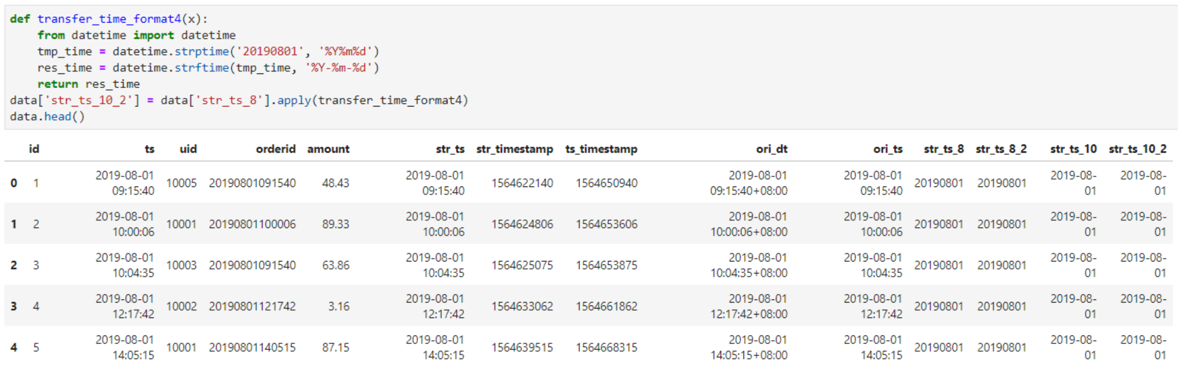

```
hive> select id, ts, concat(substr(dt8, 1, 4), '-', substr(dt8, 5, 2), '-', substr(dt8, 7,2))
    > from
    > (
    > select *, regexp_replace(substr(ts, 1, 10),'-','') as dt8
    > from t_order
    > ) a
    > limit 20;
OK
1       2019-08-01 09:15:40     2019-08-01
2       2019-08-01 10:00:06     2019-08-01
3       2019-08-01 10:04:35     2019-08-01
4       2019-08-01 12:17:42     2019-08-01
5       2019-08-01 14:05:15     2019-08-01
6       2019-08-01 14:05:29     2019-08-01
7       2019-08-02 08:13:15     2019-08-02
```

图中代码如下：

```
#python
data['str_ts_10'] = data['str_ts_8'].apply(lambda x : x[:4] + "-" + x[4:6] + "-"
+ x[6:])
data.head()

#MySQL
select id, ts, concat(substr(dt8, 1, 4), '-', substr(dt8, 5, 2), '-',
substr(dt8, 7,2))
from
(
select *,  replace(substr(ts, 1, 10), '-', '')  as dt8
from t_order
) a

#Hive
select id, ts, concat(substr(dt8, 1, 4), '-', substr(dt8, 5, 2), '-',
substr(dt8, 7,2))
from
(
select *, regexp_replace(substr(ts, 1, 10),'-','') as dt8
from t_order
) a
limit 20;
```

方法二，通过unix时间戳转换：

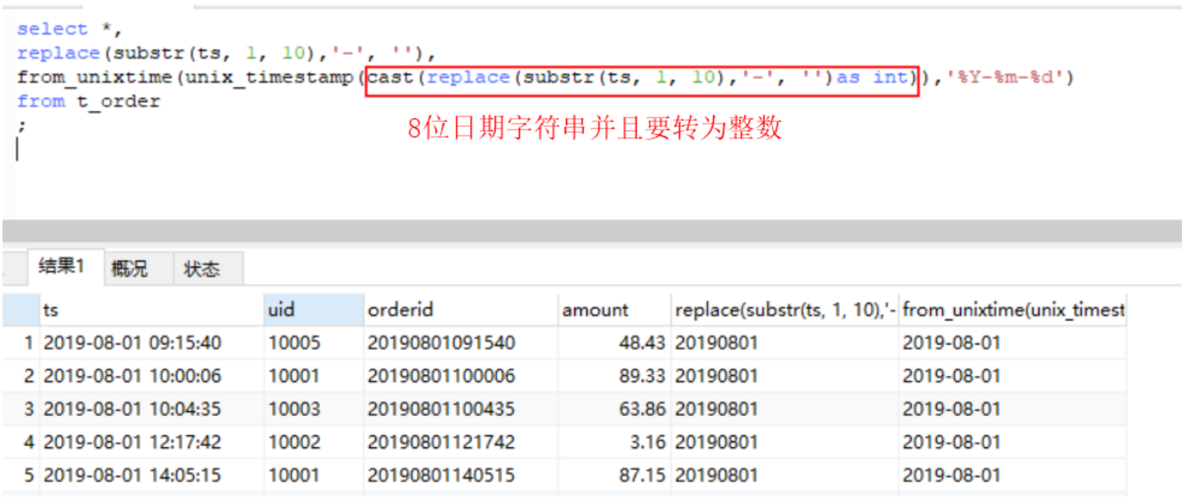
在pandas中，借助unix时间戳转换并不方便，我们可以使用datetime模块的格式化函数来实现，如下所示。



```python
def transfer_time_format4(x):
    from datetime import datetime
    tmp_time = datetime.strptime('20190801', '%Y%m%d')
    res_time = datetime.strftime(tmp_time, '%Y-%m-%d')
    return res_time
data['str_ts_10_2'] = data['str_ts_8'].apply(transfer_time_format4)
data.head()
```

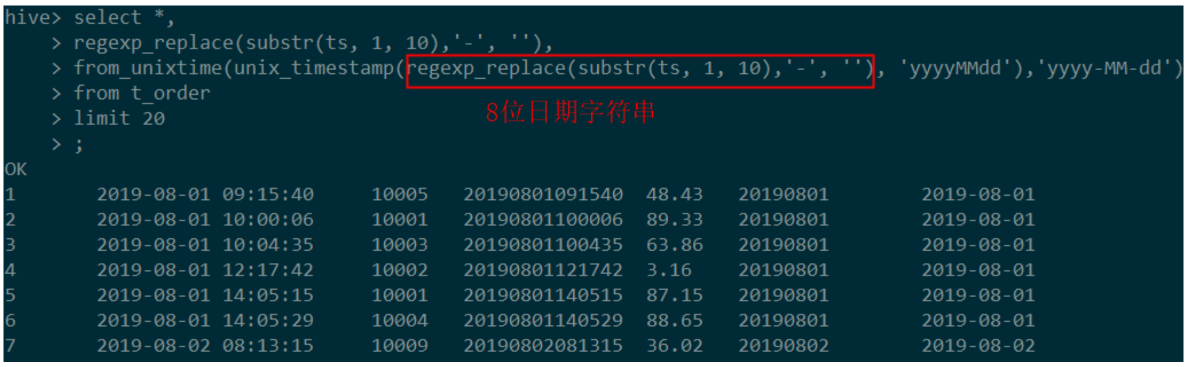
	id	ts	uid	orderid	amount	str_ts	str_timestamp	ts_timestamp	ori_dt	ori_ts	str_ts_8	str_ts_8_2	str_ts_10	str_ts_10_2
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-08-01 09:15:40	1564622140	1564650940	2019-08-01 09:15:40+08:00	2019-08-01 09:15:40	20190801	20190801	2019-08-01	2019-08-01
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-08-01 10:00:06	1564624806	1564653606	2019-08-01 10:00:06+08:00	2019-08-01 10:00:06	20190801	20190801	2019-08-01	2019-08-01
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-08-01 10:04:35	1564625075	1564653875	2019-08-01 10:04:35+08:00	2019-08-01 10:04:35	20190801	20190801	2019-08-01	2019-08-01
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-08-01 12:17:42	1564633062	1564661862	2019-08-01 12:17:42+08:00	2019-08-01 12:17:42	20190801	20190801	2019-08-01	2019-08-01
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-08-01 14:05:15	1564639515	1564668315	2019-08-01 14:05:15+08:00	2019-08-01 14:05:15	20190801	20190801	2019-08-01	2019-08-01

Mysql和Hive中unix_timestamp接收的参数不一样，前者必须输入为整数，后者可以为字符串。我们的目标是输入一个8位的时间字符串，输出一个10位的时间字符串。由于原始数据集中没有8位时间，我们临时构造了一个。代码如下：



```sql
select *,
replace(substr(ts, 1, 10),'-', ''),
from_unixtime(unix_timestamp(cast(replace(substr(ts, 1, 10),'-', '')as int)),'%Y-%m-%d')
from t_order
;
```

	ts	uid	orderid	amount	replace(substr(ts, 1, 10),'-	from_unixtime(unix_timest
1	2019-08-01 09:15:40	10005	20190801091540	48.43	20190801	2019-08-01
2	2019-08-01 10:00:06	10001	20190801100006	89.33	20190801	2019-08-01
3	2019-08-01 10:04:35	10003	20190801100435	63.86	20190801	2019-08-01
4	2019-08-01 12:17:42	10002	20190801121742	3.16	20190801	2019-08-01
5	2019-08-01 14:05:15	10001	20190801140515	87.15	20190801	2019-08-01



```
hive> select *,
    > regexp_replace(substr(ts, 1, 10),'-', ''),
    > from_unixtime(unix_timestamp(regexp_replace(substr(ts, 1, 10),'-', ''), 'yyyyMMdd'),'yyyy-MM-dd')
    > from t_order
    > limit 20
    > ;
OK
1       2019-08-01 09:15:40     10005   20190801091540  48.43   20190801        2019-08-01
2       2019-08-01 10:00:06     10001   20190801100006  89.33   20190801        2019-08-01
3       2019-08-01 10:04:35     10003   20190801100435  63.86   20190801        2019-08-01
4       2019-08-01 12:17:42     10002   20190801121742  3.16    20190801        2019-08-01
5       2019-08-01 14:05:15     10001   20190801140515  87.15   20190801        2019-08-01
6       2019-08-01 14:05:29     10004   20190801140529  88.65   20190801        2019-08-01
7       2019-08-02 08:13:15     10009   20190802081315  36.02   20190802        2019-08-02
```

```
#pandas
def transfer_time_format4(x):
    from datetime import datetime
    tmp_time = datetime.strptime('20190801', '%Y%m%d')
    res_time = datetime.strftime(tmp_time, '%Y-%m-%d')
    return res_time
data['str_ts_10_2'] = data['str_ts_8'].apply(transfer_time_format4)
data.head()

#MySQL
select *,
replace(substr(ts, 1, 10),'-', ''),
from_unixtime(unix_timestamp(cast(replace(substr(ts, 1, 10),'-', '')as
int)),'%Y-%m-%d')
from t_order
;

#Hive
select *,
regexp_replace(substr(ts, 1, 10),'-', ''),
from_unixtime(unix_timestamp(regexp_replace(substr(ts, 1, 10),'-', ''),
'yyyyMMdd'),'yyyy-MM-dd')
from t_order
limit 20
;
```

ps.关于时间Hive中的时间转换，我在之前总结Hive函数的文章的最后一部分中已经有过梳理，例子比此处更加具体，欢迎翻阅。

日期计算

日期计算主要包括日期间隔(加减一个数变为另一个日期)和计算两个日期之间的差值。

1.日期间隔

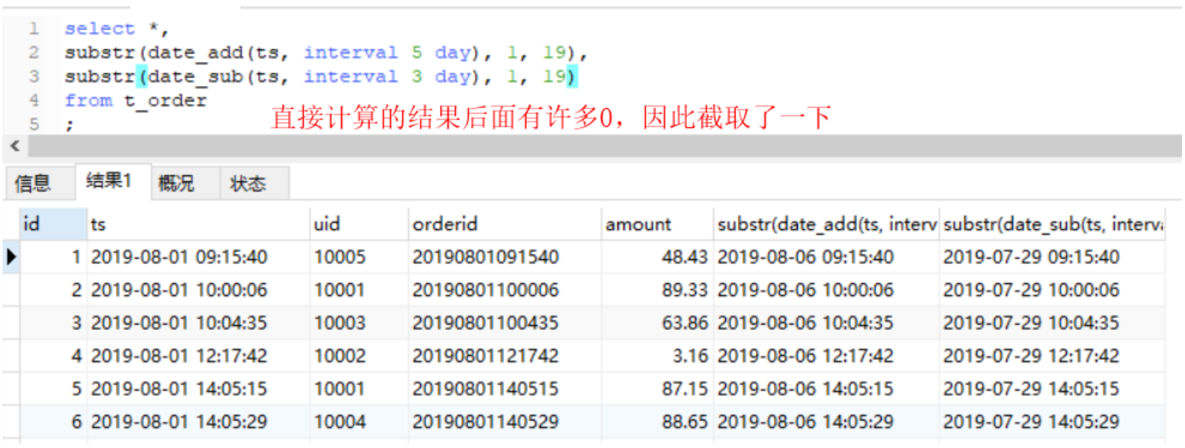
pandas中对于日期间隔的计算需要借助datetime 模块。我们来看一下如何计算ts之后5天和之前3天。

```python
import datetime
from datetime import timedelta
data['ts_plus_5'] = data['ts'] + timedelta(days=5)
data['ts_minus_3'] = data['ts'] + timedelta(days=-3)
data.head()
```

	id	ts	uid	orderid	amount	ts_plus_5	ts_minus_3
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-08-06 09:15:40	2019-07-29 09:15:40
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-08-06 10:00:06	2019-07-29 10:00:06
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-08-06 10:04:35	2019-07-29 10:04:35
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-08-06 12:17:42	2019-07-29 12:17:42
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-08-06 14:05:15	2019-07-29 14:05:15

使用timedelta函数既可以实现天为单位的日期间隔，也可以按周，分钟，秒等进行计算。

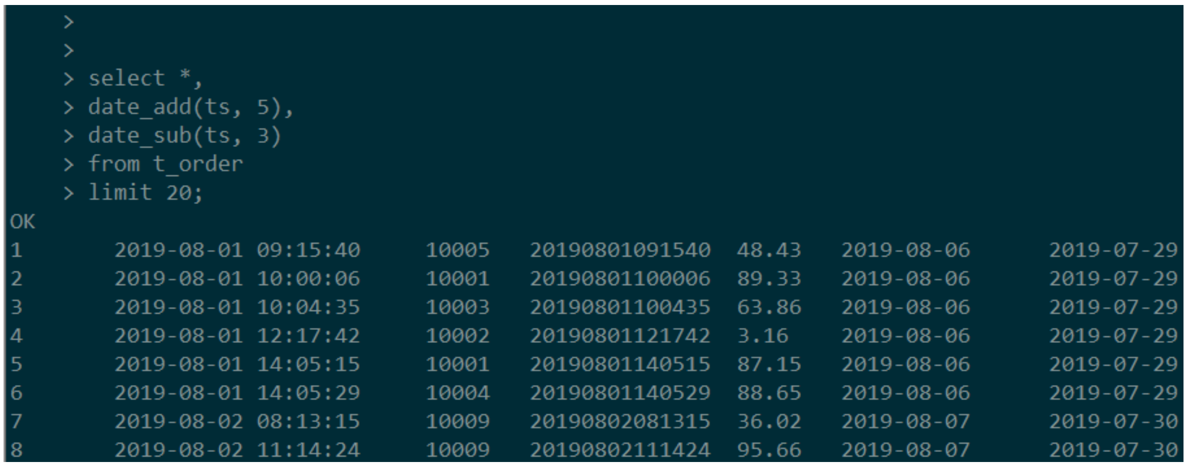
在MySQL和Hive中有相应的日期间隔函数date_add，date_sub函数，但使用的格式略有差异。



```sql
select *,
substr(date_add(ts, interval 5 day), 1, 19),
substr(date_sub(ts, interval 3 day), 1, 19)
from t_order
;
```

直接计算的结果后面有许多0，因此截取了一下

id	ts	uid	orderid	amount	substr(date_add(ts, interv	substr(date_sub(ts, interv
1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-08-06 09:15:40	2019-07-29 09:15:40
2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-08-06 10:00:06	2019-07-29 10:00:06
3	2019-08-01 10:04:35	10003	20190801100435	63.86	2019-08-06 10:04:35	2019-07-29 10:04:35
4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-08-06 12:17:42	2019-07-29 12:17:42
5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-08-06 14:05:15	2019-07-29 14:05:15
6	2019-08-01 14:05:29	10004	20190801140529	88.65	2019-08-06 14:05:29	2019-07-29 14:05:29



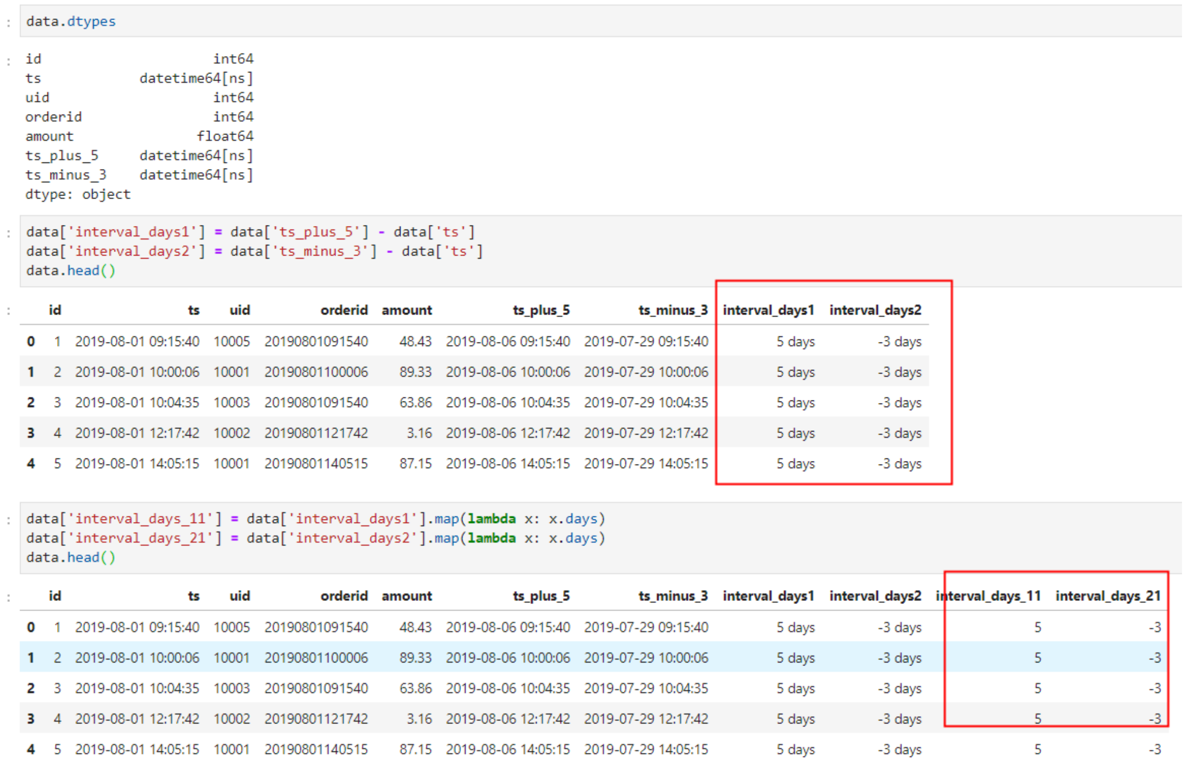
```
    >
    >
    > select *,
    > date_add(ts, 5),
    > date_sub(ts, 3)
    > from t_order
    > limit 20;
OK
1       2019-08-01 09:15:40     10005   20190801091540  48.43   2019-08-06      2019-07-29
2       2019-08-01 10:00:06     10001   20190801100006  89.33   2019-08-06      2019-07-29
3       2019-08-01 10:04:35     10003   20190801100435  63.86   2019-08-06      2019-07-29
4       2019-08-01 12:17:42     10002   20190801121742  3.16    2019-08-06      2019-07-29
5       2019-08-01 14:05:15     10001   20190801140515  87.15   2019-08-06      2019-07-29
6       2019-08-01 14:05:29     10004   20190801140529  88.65   2019-08-06      2019-07-29
7       2019-08-02 08:13:15     10009   20190802081315  36.02   2019-08-07      2019-07-30
8       2019-08-02 11:14:24     10009   20190802111424  95.66   2019-08-07      2019-07-30
```

H需要注意的是Hive计算的结果没有时分秒，如果需要，依然可以使用拼接的方式获得，此处略。

2.日期差

这一小节仍然是上一小节的逆操作。(怎么这么多逆操作，累不累啊......)我们来看一下如何计算两个时间的日期差。

在pandas中，如果事件类型是datetime64[ns]类型，直接作差就可以得出日期差，但是得到的数据后面还有一个"days"的单位，这其实就是上一小节提到的timedelta类型。为了便于使用，我们使用map函数获取其days属性，得到我们想要的数值的差。如下所示：



```
data.dtypes
```

```
id                    int64
ts           datetime64[ns]
uid                   int64
orderid               int64
amount              float64
ts_plus_5    datetime64[ns]
ts_minus_3   datetime64[ns]
dtype: object
```

```
data['interval_days1'] = data['ts_plus_5'] - data['ts']
data['interval_days2'] = data['ts_minus_3'] - data['ts']
data.head()
```

	id	ts	uid	orderid	amount	ts_plus_5	ts_minus_3	interval_days1	interval_days2
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-08-06 09:15:40	2019-07-29 09:15:40	5 days	-3 days
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-08-06 10:00:06	2019-07-29 10:00:06	5 days	-3 days
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-08-06 10:04:35	2019-07-29 10:04:35	5 days	-3 days
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-08-06 12:17:42	2019-07-29 12:17:42	5 days	-3 days
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-08-06 14:05:15	2019-07-29 14:05:15	5 days	-3 days

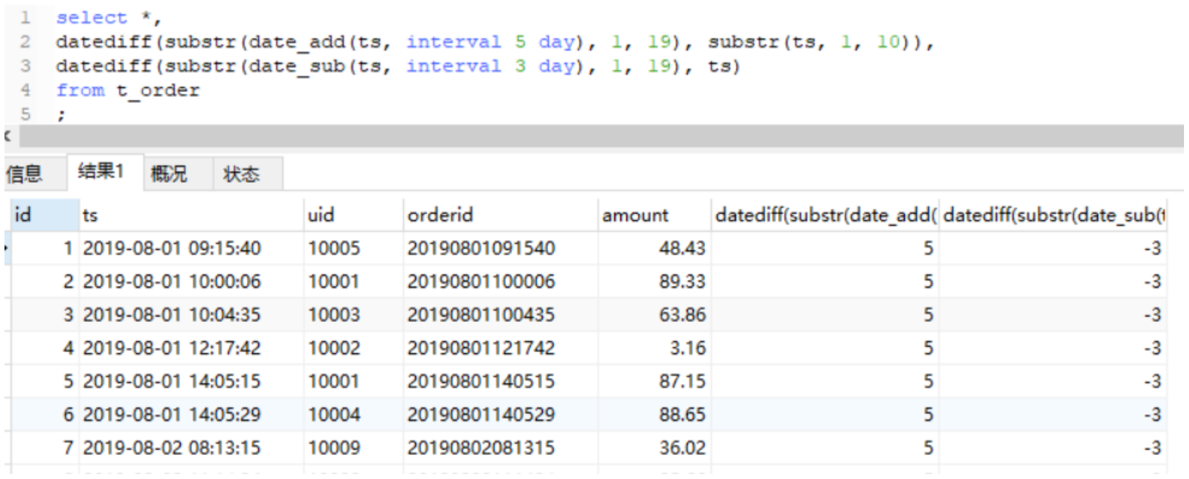
```
data['interval_days_11'] = data['interval_days1'].map(lambda x: x.days)
data['interval_days_21'] = data['interval_days2'].map(lambda x: x.days)
data.head()
```

	id	ts	uid	orderid	amount	ts_plus_5	ts_minus_3	interval_days1	interval_days2	interval_days_11	interval_days_21
0	1	2019-08-01 09:15:40	10005	20190801091540	48.43	2019-08-06 09:15:40	2019-07-29 09:15:40	5 days	-3 days	5	-3
1	2	2019-08-01 10:00:06	10001	20190801100006	89.33	2019-08-06 10:00:06	2019-07-29 10:00:06	5 days	-3 days	5	-3
2	3	2019-08-01 10:04:35	10003	20190801091540	63.86	2019-08-06 10:04:35	2019-07-29 10:04:35	5 days	-3 days	5	-3
3	4	2019-08-01 12:17:42	10002	20190801121742	3.16	2019-08-06 12:17:42	2019-07-29 12:17:42	5 days	-3 days	5	-3
4	5	2019-08-01 14:05:15	10001	20190801140515	87.15	2019-08-06 14:05:15	2019-07-29 14:05:15	5 days	-3 days	5	-3

如果步是datetime格式，可以先用下面的代码进行一次转换。

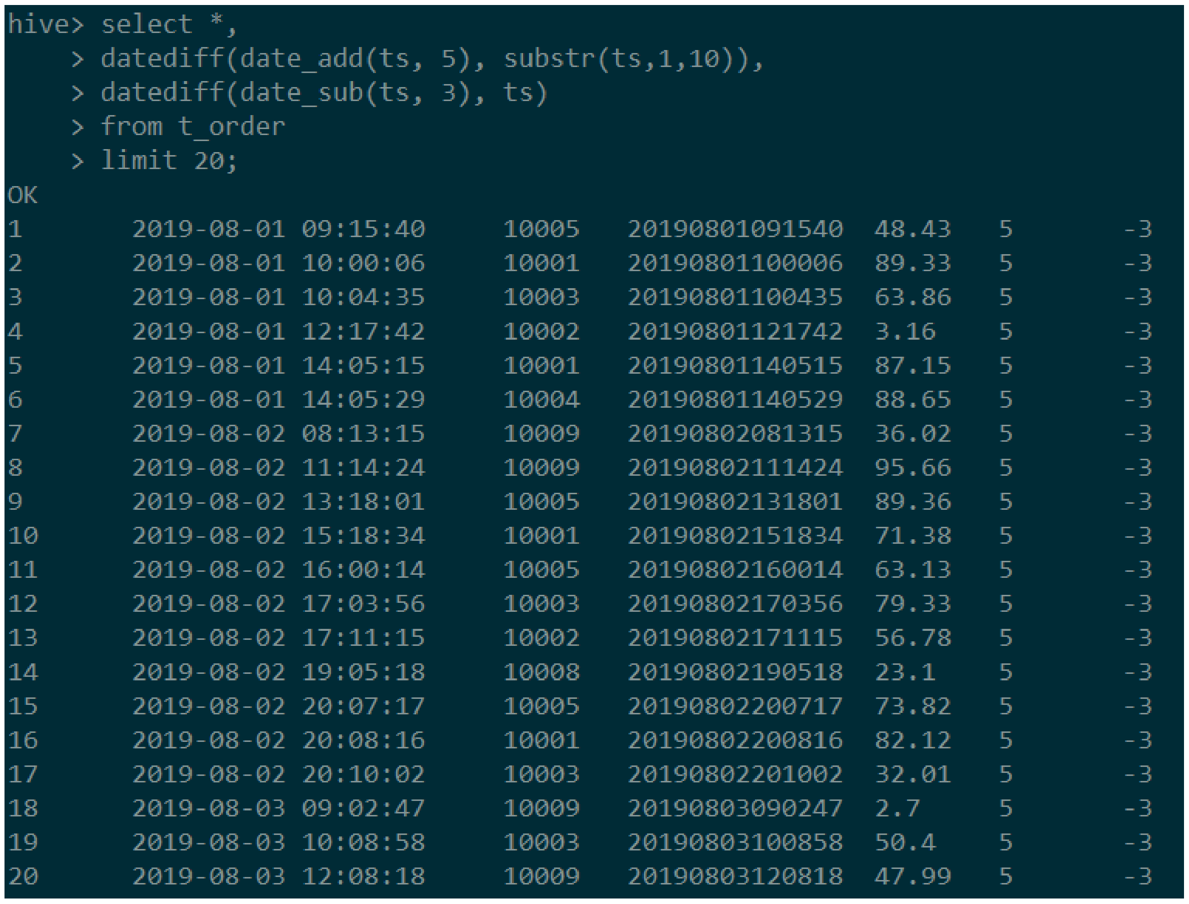
```
#str_ts是字符串格式，转换出的dt_ts是datetime64[ns]格式
data['dt_ts'] = pd.to_datetime(data['str_ts'], format='%Y-%m-%d %H:%M:%S')
```

Hive和MySQL中的日期差有相应的函数datediff。但需要注意它的输入格式。



```sql
select *,
datediff(substr(date_add(ts, interval 5 day), 1, 19), substr(ts, 1, 10)),
datediff(substr(date_sub(ts, interval 3 day), 1, 19), ts)
from t_order
;
```

id	ts	uid	orderid	amount	datediff(substr(date_add(	datediff(substr(date_sub(
1	2019-08-01 09:15:40	10005	20190801091540	48.43	5	-3
2	2019-08-01 10:00:06	10001	20190801100006	89.33	5	-3
3	2019-08-01 10:04:35	10003	20190801100435	63.86	5	-3
4	2019-08-01 12:17:42	10002	20190801121742	3.16	5	-3
5	2019-08-01 14:05:15	10001	20190801140515	87.15	5	-3
6	2019-08-01 14:05:29	10004	20190801140529	88.65	5	-3
7	2019-08-02 08:13:15	10009	20190802081315	36.02	5	-3



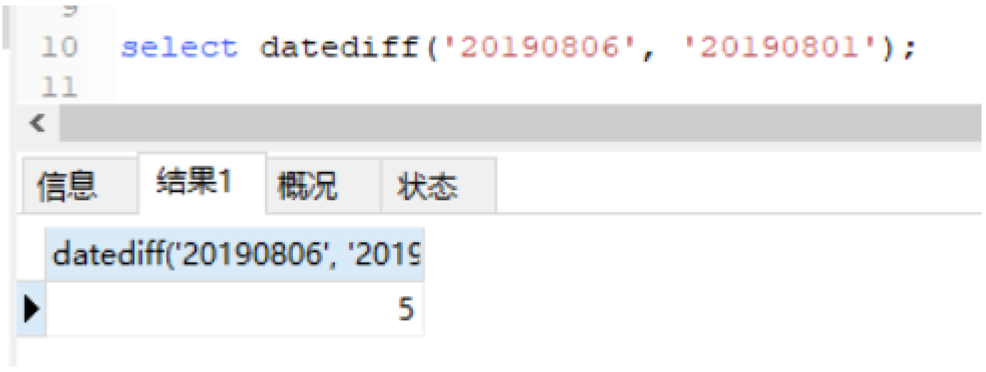
```
hive> select *,
    > datediff(date_add(ts, 5), substr(ts,1,10)),
    > datediff(date_sub(ts, 3), ts)
    > from t_order
    > limit 20;
OK
1       2019-08-01 09:15:40     10005   20190801091540  48.43   5       -3
2       2019-08-01 10:00:06     10001   20190801100006  89.33   5       -3
3       2019-08-01 10:04:35     10003   20190801100435  63.86   5       -3
4       2019-08-01 12:17:42     10002   20190801121742  3.16    5       -3
5       2019-08-01 14:05:15     10001   20190801140515  87.15   5       -3
6       2019-08-01 14:05:29     10004   20190801140529  88.65   5       -3
7       2019-08-02 08:13:15     10009   20190802081315  36.02   5       -3
8       2019-08-02 11:14:24     10009   20190802111424  95.66   5       -3
9       2019-08-02 13:18:01     10005   20190802131801  89.36   5       -3
10      2019-08-02 15:18:34     10001   20190802151834  71.38   5       -3
11      2019-08-02 16:00:14     10005   20190802160014  63.13   5       -3
12      2019-08-02 17:03:56     10003   20190802170356  79.33   5       -3
13      2019-08-02 17:11:15     10002   20190802171115  56.78   5       -3
14      2019-08-02 19:05:18     10008   20190802190518  23.1    5       -3
15      2019-08-02 20:07:17     10005   20190802200717  73.82   5       -3
16      2019-08-02 20:08:16     10001   20190802200816  82.12   5       -3
17      2019-08-02 20:10:02     10003   20190802201002  32.01   5       -3
18      2019-08-03 09:02:47     10009   20190803090247  2.7     5       -3
19      2019-08-03 10:08:58     10003   20190803100858  50.4    5       -3
20      2019-08-03 12:08:18     10009   20190803120818  47.99   5       -3
```

代码如下:

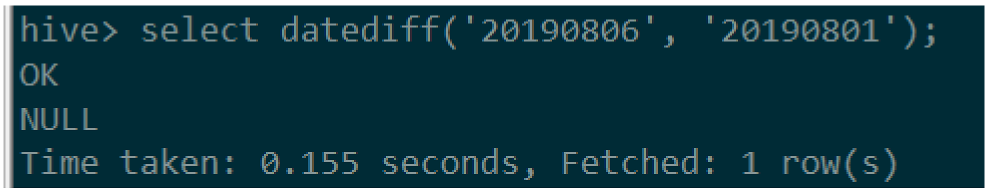
```
#pandas
data['interval_days1'] = data['ts_plus_5'] - data['ts']
data['interval_days2'] = data['ts_minus_3'] - data['ts']
data['interval_days_11'] = data['interval_days1'].map(lambda x: x.days)
data['interval_days_21'] = data['interval_days2'].map(lambda x: x.days)
data.head()

#MySQL
select *,
datediff(substr(date_add(ts, interval 5 day), 1, 19), substr(ts, 1, 10)),
datediff(substr(date_sub(ts, interval 3 day), 1, 19), ts)
from t_order
;
```

```
#Hive
select *,
datediff(date_add(ts, 5), substr(ts,1,10)),
datediff(date_sub(ts, 3), ts)
from t_order
limit 20;
```

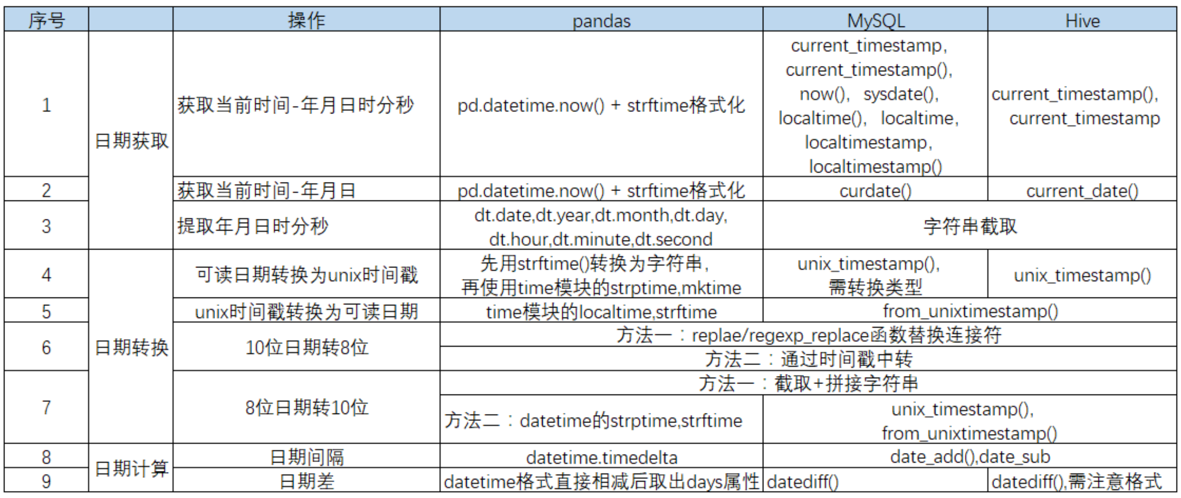
可以看到输入的形式既可以是具体到时分秒的格式，也可以是年月日格式。但是要注意Hive中输入的日期必须是10位的格式，否则得不到正确的结果，比如输入8位的，结果会是NULL，而MySQL则可以进行8位日期的计算。


```
select datediff('20190806', '20190801');
```

datediff('20190806', '2019
5


```
hive> select datediff('20190806', '20190801');
OK
NULL
Time taken: 0.155 seconds, Fetched: 1 row(s)
```

小结



序号		操作	pandas	MySQL	Hive
1	日期获取	获取当前时间-年月日时分秒	pd.datetime.now() + strftime格式化	current_timestamp, current_timestamp(), now(), sysdate(), localtime(), localtime, localtimestamp, localtimestamp()	current_timestamp(), current_timestamp
2		获取当前时间-年月日	pd.datetime.now() + strftime格式化	curdate()	current_date()
3		提取年月日时分秒	dt.date,dt.year,dt.month,dt.day, dt.hour,dt.minute,dt.second	字符串截取	
4	日期转换	可读日期转换为unix时间戳	先用strftime()转换为字符串，再使用time模块的strptime,mktime	unix_timestamp(), 需转换类型	unix_timestamp()
5		unix时间戳转换为可读日期	time模块的localtime,strftime	from_unixtimestamp()	
6		10位日期转8位	方法一：replae/regexp_replace函数替换连接符		
			方法二：通过时间戳中转		
7		8位日期转10位	方法一：截取+拼接字符串		
			方法二：datetime的strptime,strftime	unix_timestamp(), from_unixtimestamp()	
8	日期计算	日期间隔	datetime.timedelta	date_add(),date_sub	
9		日期差	datetime格式直接相减后取出days属性	datediff()	datediff(),需注意格式

本文涉及到的对比操作和相应的解法如上图所示。整体看起来比之前的要“乱”一些，但仔细看看并没有多少内容。需要指出，关于日期操作，本文只是总结了一些pandas和SQL都有的部分操作，也都是比较常见的。python中和SQL本身还有很多其他用法，限于时间关系就省略了。由于时间匆忙，行文不当之处还请多多包含。如果你有好的想法，欢迎一起交流学习。本文的代码和数据可以在公众号后台回复“**对比三**”获取，祝学习愉快！