Hive 基本操作

```
（1）启动 hive

bin/hive

（2）查看数据库

hive> show databases;

（3）打开默认数据库

hive> use default;

（4）显示 default 数据库中的表

hive> show tables;

（5）创建一张表

hive> create table student(id int, name string);

（6）显示数据库中有几张表

hive> show tables;

（7）查看表的结构

hive> desc student;

（8）向表中插入数据

hive> insert into student values(1000,"ss");

（9）查询表中数据

hive> select * from student;

（10）退出 hive

hive> quit;
```

Hive 实际操作

```
（1）启动 hive

bin/hive

（2）显示数据库

hive> show databases;

（3）使用 default 数据库
```

```
hive> use default;

（4）显示 default 数据库中的表

hive> show tables;

（5）删除已创建的 student 表

hive> drop table student;

（6）创建 student 表，并声明文件分隔符'\t'

hive> create table student(id int, name string) ROW FORMAT

DELIMITED FIELDS TERMINATED

BY '\t';

（7）加载/opt/module/data/student.txt 文件到 student 数据库表中。

hive> load data local inpath '/opt/module/data/student.txt' into  table student;
```

建表语句

```
create table test(
name string, friends array<string>, children map<string, int>, address
struct<street:string, city:string>
) row format delimited fields terminated by ',' collection items terminated by
'_' map keys terminated by ':'
lines terminated by '\n';
字段解释:
row format delimited fields terminated by ',' -- 列分隔符
collection items terminated by '_'
--MAP STRUCT 和 ARRAY 的分隔符(数据分割 符号)
map keys terminated by ':' -- MAP 中的 key 与 value 的分隔符
lines terminated by '\n'; -- 行分隔符
```

显示数据库信息

```
hive> desc database db_hive;
```

显示数据库详细信息，extended

```
hive> desc database extended db_hive;
```

删除空数据库

```
hive>drop database db_hive2;
```

如果删除的数据库不存在，最好采用 if exists 判断数据库是否存在

```
hive> drop database db_hive;
hive> drop database if exists db_hive2;
```

如果数据库不为空，可以采用 cascade 命令，强制删除

```
hive> drop database db_hive cascade;
```

查看表格式化数据

```
hive > desc formatted dept;
```

管理表与外部表的互相转换

```
（1）查询表的类型

hive (default)> desc formatted student2;

（2）修改内部表 student2 为外部表

alter table student2 set tblproperties('EXTERNAL'='TRUE');

（3）查询表的类型

hive (default)> desc formatted student2;

（4）修改外部表 student2 为内部表

alter table student2 set tblproperties('EXTERNAL'='FALSE'); ( 括号内部要大写)

（5）查询表的类型

hive (default)> desc formatted student2;
```

分区表

分区表基本操作

```
1.创建分区表语法
hive (default)> create table dept_partition(
deptno int, dname string, loc string
)
partitioned by (month string)
row format delimited fields terminated by '\t';
2. 加载数据到分区表中
hive (default)> load data local inpath '/opt/module/datas/dept.txt' into table
default.dept_partition partition(month='201709');
hive (default)> load data local inpath '/opt/module/datas/dept.txt' into table
default.dept_partition partition(month='201708');
hive (default)> load data local inpath '/opt/module/datas/dept.txt' into table
default.dept_partition partition(month='201707');

3. 查询分区表中数据
(1).单分区查询
```

```
hive (default)> select * from dept_partition where month='201709';
(2)多分区联合查询
hive (default)> select * from dept_partition where month='201709'
 union select * from dept_partition where month='201708'
 union select * from dept_partition where month='201707';
4. 增加分区
(1).创建单个分区
hive (default)> alter table dept_partition add partition(month='201706') ;
(2).同时创建多个分区
hive (default)> alter table dept_partition add partition(month='201705')
           partition(month='201704');
5. 删除分区
(1).删除单个分区
hive (default)> alter table dept_partition drop partition (month='201704');
(2).同时删除多个分区
hive (default)> alter table dept_partition drop partition (month='201705'),
partition (month='201706');
6. 查看分区表有多少分区
hive> show partitions dept_partition;
7. 查看分区表结构
hive> desc formatted dept_partition;
```

分区表注意事项

```
1.创建二级分区表
hive (default)> create table dept_partition2(
 deptno int, dname string, loc string)
 partitioned by (month string, day string)
 row format delimited fields terminated by '\t';
2.正常的加载数据
（1）加载数据到二级分区表中 hive (default)> load data local inpath
    '/opt/module/datas/dept.txt' into table
    default.dept_partition2 partition(month='201709', day='13');
（2）查询分区数据
hive (default)> select * from dept_partition2 where month='201709' and day='13';
3.查询数据
select * from dept_partition2 where month='201709' and day='10';
```

修改表

重命名表

```
1. 语法
alter table table_name rename to new_table_name
2. 实操案例
hive (default)> alter table dept_partition2 rename to
dept_partition3;
```

增加、修改和删除表分区

```
详见 4.6.1 分区表基本操作。
```

增加**/修改/替换列信息**

```
1. 语法
更新列
ALTER TABLE table_name CHANGE [COLUMN] col_old_name col_new_name
column_type [COMMENT col_comment] [FIRST|AFTER column_name]
增加和替换列
ALTER TABLE table_name ADD|REPLACE COLUMNS (col_name data_type
[COMMENT col_comment], ...)
注：ADD 是代表新增一字段，字段位置在所有列后面(partition 列前)，REPLACE 则是
表示替换表中所有字段。
```

```
2. 实操案例
（1）查询表结构
hive> desc dept_partition;
（2）添加列
hive (default)> alter table dept_partition add columns(deptdesc string);
（3）查询表结构
hive> desc dept_partition;
（4）更新列
hive (default)> alter table dept_partition change column deptdesc desc int;
（5）查询表结构
hive> desc dept_partition;
（6）替换列
hive (default)> alter table dept_partition replace columns(deptno string, dname
string, loc string);
（7）查询表结构
hive> desc dept_partition;
```

删除表

```
hive (default)> drop table dept_partition;
```

数据导入

向表中装载数据

```
创建一张表
hive (default)> create table student(id string, name string) row
format delimited fields terminated by '\t';
```

```
加载本地文件到 hive
hive (default)> load data local inpath
'/opt/module/datas/student.txt' into table default.student;
```

```
加载 HDFS 文件到 hive 中上传文件到 HDFS
hive (default)> dfs -put /opt/module/datas/student.txt /user/atguigu/hive;
加载 HDFS 上数据
hive (default)> load data inpath '/user/atguigu/hive/student.txt' into table
default.student;
```

```
加载数据覆盖表中已有的数据上传文件到 HDFS
hive (default)> dfs -put /opt/module/datas/student.txt /user/atguigu/hive;
加载数据覆盖表中已有的数据
hive (default)> load data inpath '/user/atguigu/hive/student.txt' overwrite into
table default.student;
```

通过查询语句向表中插入数据（Insert）

1．创建一张分区表

```
hive (default)> create table student(id int, name string) partitioned by (month
string) row format delimited fields terminated by '\t';
```

2．基本插入数据

```
hive (default)> insert into table student partition(month='201709')
values(1,'wangwu');
```

3．基本模式插入（根据单张表查询结果）

```
hive (default)> insert overwrite table student partition(month='201708') select
id, name from student where month='201709';
```

4．多插入模式（根据多张表查询结果）

```
hive (default)> from student insert overwrite table student
partition(month='201707')
        select id, name where month='201709'  insert overwrite table student
partition(month='201706')
        select id, name where month='201709';
```

创建表时通过 Location 指定加载数据路径

```
1. 创建表，并指定在 hdfs 上的位置
 hive (default)> create table if not exists student5( id int, name string)
 row format delimited fields terminated by '\t'
 location '/user/hive/warehouse/student5';
2. 上传数据到 hdfs 上 hive (default)> dfs -put /opt/module/datas/student.txt
/user/hive/warehouse/student5;
3. 查询数据
   hive (default)> select * from student5;
```

数据导出

Insert 导出

```
1. 将查询的结果导出到本地
hive (default)> insert overwrite local directory
'/opt/module/datas/export/student'
               select * from student;
2. 将查询的结果格式化导出到本地
hive(default)>insert overwrite local directory
'/opt/module/datas/export/student1'
             row format delimited fileds terminated by '\t'  select * from
student;
3. 将查询的结果导出到 HDFS 上(没有 local)
hive (default)> insert overwrite directory '/user/atguigu/student2'    row format
delimited fileds                    terminated by '\t'  select * from student;
```

Hadoop 命令导出到本地

```
hive (default)> dfs -get  /user/hive/warehouse/student/month=201709/000000_0
/opt/module/datas/export/student3.txt;
```

Hive Shell 命令导出

```
基本语法：（hive -f/-e 执行语句或者脚本 > file）
[atguigu@hadoop102 hive]$ bin/hive -e 'select * from default.student;' >
                        /opt/module/datas/export/student4.txt;
```

Export 导出到HDFS 上

```
(defahiveult)> export table default.student to
 '/user/hive/warehouse/export/student';
```

清除表中数据(Truncate)

```
注意：Truncate 只能删除管理表，不能删除外部表中数据

hive (default)> truncate table student;
```

查询

基本查询（**Select…From）**

全表和特定列查询

```
1. 全表查询
hive (default)> select * from emp;
```

```
2. 选择特定列查询
hive (default)> select empno, ename from emp;
```

注意：
（1）SQL 语言大小写不敏感。
（2）SQL 可以写在一行或者多行
（3）关键字不能被缩写也不能分行
（4）各子句一般要分行写。
（5）使用缩进提高语句的可读性。

列别名

1. 重命名一个列
2. 便于计算
3. 紧跟列名，也可以在列名和别名之间加入关键字‘AS’

案例实操

查询名称和部门

```
hive (default)> select ename AS name, deptno dn from emp;
```

算术运算符

运算符	描述
A+B	A 和 B 相加
A-B	A 减去 B
A*B	A 和 B 相乘
A/B	A 除以 B
A%B	A 对 B 取余
A&B	A 和 B 按位取与
A|B	A 和 B 按位取或
A^B	A 和 B 按位取异或
~A	A 按位取反

案例实操
查询出所有员工的薪水后加 1 显示。

```
hive (default)> select sal +1 from emp;
```

常用函数

```
1. 求总行数（count）
hive (default)> select count(*) cnt from emp;
2. 求工资的最大值（max）
hive (default)> select max(sal) max_sal from emp;
3. 求工资的最小值（min）
hive (default)> select min(sal) min_sal from emp;
4. 求工资的总和（sum）
hive (default)> select sum(sal) sum_sal from emp;
5. 求工资的平均值（avg）
hive (default)> select avg(sal) avg_sal from emp;
```

Limit 语句

```
典型的查询会返回多行数据。LIMIT 子句用于限制返回的行数。
hive (default)> select * from emp limit 5;
```

Where 语句

```
1. 使用 WHERE 子句，将不满足条件的行过滤掉
2. WHERE 子句紧随 FROM 子句
3. 案例实操
查询出薪水大于 1000 的所有员工
hive (default)> select * from emp where sal >1000;
```

比较运算符（**Between/In/ Is Null）**

操作符	支持的数据类型	描述
A=B	基本数据类型	如果 A 等于 B 则返回 TRUE，反之返回 FALSE
A<=>B	基本数据类型	如果 A 和 B 都为 NULL，则返回 TRUE，其他的和等号（=）操作符的结果一致，如果任一为 NULL 则结果为 NULL
A<>B, A!=B	基本数据类型	A 或者 B 为 NULL 则返回 NULL；如果 A 不等于 B，则返回 TRUE，反之返回 FALSE
A<B	基本数据类型	A 或者 B 为 NULL，则返回 NULL；如果 A 小于 B，则返回 TRUE，反之返回 FALSE
A<=B	基本数据类型	A 或者 B 为 NULL，则返回 NULL；如果 A 小于等于 B，则返回 TRUE，反之返回 FALSE
A>B	基本数据类型	A 或者 B 为 NULL，则返回 NULL；如果 A 大于 B，则返回 TRUE，反之返回 FALSE
A>=B	基本数据类型	A 或者 B 为 NULL，则返回 NULL；如果 A 大于等于 B，则返回 TRUE，反之返回 FALSE
A [NOT] BETWEEN B AND C	基本数据类型	如果 A，B 或者 C 任一为 NULL，则结果为 NULL。如果 A 的值大于等于 B 而且小于或等于 C，则结果为 TRUE，反之为 FALSE。如果使用 NOT 关键字则可达到相反的效果。
A IS NULL	所有数据类型	如果 A 等于 NULL，则返回 TRUE，反之返回 FALSE
A IS NOT NULL	所有数据类型	如果 A 不等于 NULL，则返回 TRUE，反之返回 FALSE
IN(数值 1, 数值 2)	所有数据类型	使用 IN 运算显示列表中的值
A [NOT] LIKE B	STRING 类型	B 是一个 SQL 下的简单正则表达式，如果 A 与其匹配的话，则返回 TRUE；反之返回 FALSE。B 的表达式说明如下：‘x%’表示 A必须以字母‘x’开头，‘%x’表示 A 必须以字母‘x’结尾，而‘%x%’表示 A 包含有字母‘x’,可以位于开头，结尾或者字符串中间。如果使用 NOT 关键字则可达到相反的效果。
A RLIKE B, A REGEXP B	STRING 类型	B 是一个正则表达式，如果 A 与其匹配，则返回 TRUE；反之返回 FALSE。匹配使用的是 JDK 中的正则表达式接口实现的，因为正也依据其中的规则。例如，正则表达式必须和整个字符串 A 相匹配，而不是只需与其字符串匹配。

案例实操

```
（1）查询出薪水等于 5000 的所有员工
hive (default)> select * from emp where sal =5000;
（2）查询工资在 500 到 1000 的员工信息
hive (default)> select * from emp where sal between 500 and 1000;
（3）查询 comm 为空的所有员工信息
hive (default)> select * from emp where comm is null;
（4）查询工资是 1500 或 5000 的员工信息
hive (default)> select * from emp where sal IN (1500, 5000);
```

Like 和 RLike

```
1）使用 LIKE 运算选择类似的值
2）选择条件可以包含字符或数字:
% 代表零个或多个字符(任意个字符)。
_ 代表一个字符。
3）RLIKE 子句是 Hive 中这个功能的一个扩展，其可以通过 Java 的正则表达式这个更
    强大的语言来指定匹配条件。
```

```
案例实操
（1）查找以 2 开头薪水的员工信息
hive (default)> select * from emp where sal LIKE '2%';
（2）查找第二个数值为 2 的薪水的员工信息
hive (default)> select * from emp where sal LIKE '_2%';
（3）查找薪水中含有 2 的员工信息
hive (default)> select * from emp where sal RLIKE '[2]';
```

逻辑运算符（**And/Or/Not）

操作符	含义
AND	逻辑并
OR	逻辑或
NOT	逻辑否

```
案例实操
（1）查询薪水大于 1000，部门是 30
hive (default)> select * from emp where sal>1000 and deptno=30;
（2）查询薪水大于 1000，或者部门是 30
hive (default)> select * from emp where sal>1000 or deptno=30;
（3）查询除了 20 部门和 30 部门以外的员工信息
hive (default)> select * from emp where deptno not IN(30, 20);
```

分组

Group By 语句

```
GROUP BY 语句通常会和聚合函数一起使用，按照一个或者多个列队结果进行分组，
然后对每个组执行聚合操作。
案例实操：
（1）计算 emp 表每个部门的平均工资
hive (default)> select t.deptno, avg(t.sal) avg_sal from emp t group by t.deptno;
（2）计算 emp 每个部门中每个岗位的最高薪水
hive (default)> select t.deptno, t.job, max(t.sal) max_sal from emp t group by
t.deptno, t.job;
```

Having 语句

1. having 与 where 不同点

```
（1）where 针对表中的列发挥作用，查询数据；having 针对查询结果中的列发挥作用，筛选数据。
（2）where 后面不能写聚合函数，而 having 后面可以使用聚合函数。
（3）having 只用于 group by 分组统计语句。
```

2. 案例实操

```
（1）求每个部门的平均薪水大于 2000 的部门求每个部门的平均工资
hive (default)> select deptno, avg(sal) from emp group by deptno;
（2） 求每个部门的平均薪水大于 2000 的部门
hive (default)> select deptno, avg(sal) avg_sal from emp group by deptno having
avg_sal > 2000;
```

Join 语句

等值 Join

```
Hive 支持通常的 SQL JOIN 语句，但是只支持等值连接，不支持非等值连接。
```

案例实操

```
表和部门表中的部门编号相等，查询员工编号、员工名称和部门名称；
根据员工表和部门表中的部门编号相等，查询员工编号、员工名称和部门名称；
hive (default)> select e.empno, e.ename, d.deptno, d.dname from emp e join dept d
on e.deptno = d.deptno;
```

表的别名

```
1. 好处
（1）使用别名可以简化查询。
（2）使用表名前缀可以提高执行效率.
```

案例实操

```
合并员工表和部门表
hive (default)> select e.empno, e.ename, d.deptno from emp e join dept d on
e.deptno = d.deptno;
```

内连接

```
内连接：只有进行连接的两个表中都存在与连接条件相匹配的数据才会被保留下来。
hive (default)> select e.empno, e.ename, d.deptno from emp e join dept d on
e.deptno = d.deptno;
```

左外连接

```
左外连接：JOIN 操作符左边表中符合 WHERE 子句的所有记录将会被返回。
hive (default)> select e.empno, e.ename, d.deptno from emp e left join dept d on
e.deptno = d.deptno;
```

右外连接

```
右外连接：JOIN 操作符右边表中符合 WHERE 子句的所有记录将会被返回。
hive (default)> select e.empno, e.ename, d.deptno from emp e right join dept d on
e.deptno = d.deptno;
```

满外连接

```
满外连接：将会返回所有表中符合 WHERE 语句条件的所有记录。如果任一表的指定
字段没有符合条件的值的话，那么就使用 NULL 值替代。location.txt
hive (default)> select e.empno, e.ename, d.deptno from emp e full join dept d on
e.deptno = d.deptno;
```

笛卡尔积

笛卡尔集会在下面条件下产生

```
（1）省略连接条件
（2）连接条件无效
（3）所有表中的所有行互相连
```

案例实操

```
hive (default)> select empno, dname from emp, dept;
```

排序

全局排序（**Order By）**

Order By：全局排序，一个 Reducer

```
1. 使用 ORDER BY 子句排序 ASC（ascend）: 升序（默认）DESC（descend）: 降序
2. ORDER BY 子句在 SELECT 语句的结尾
3. 案例实操
（1）查询员工信息按工资升序排列
  hive (default)> select * from emp order by sal;
（2）查询员工信息按工资降序排列
  hive (default)> select * from emp order by sal desc;
```

按照别名排序

```
按照员工薪水的 2 倍排序
hive (default)> select ename, sal*2 twosal from emp order by twosal;
```

多个列排序

```
按照部门和工资升序排序
hive (default)> select ename, deptno, sal from emp order by deptno, sal ;
```

每个 MapReduce 内部排序（**Sort By)

Sort By：每个 Reducer 内部进行排序，对全局结果集来说不是排序。

```
1. 设置 reduce 个数
hive (default)> set mapreduce.job.reduces=3;
2. 查看设置 reduce 个数
hive (default)> set mapreduce.job.reduces;
3. 根据部门编号降序查看员工信息
hive (default)> select * from emp sort by empno desc;
4. 将查询结果导入到文件中（按照部门编号降序排序）
hive (default)> insert overwrite local directory '/opt/module/datas/sortby-
result'
select * from emp sort by deptno desc;
```

分区排序（**Distribute By)

```
Distribute By：类似 MR 中 partition，进行分区，结合 sort by 使用。
注意，Hive 要求 DISTRIBUTE BY 语句要写在 SORT BY 语句之前。
对于 distribute by 进行测试，一定要分配多 reduce 进行处理，否则无法看到 distribute by
的效果。
```

```
案例实操：
（1）先按照部门编号分区，再按照员工编号降序排序。
hive (default)> set mapreduce.job.reduces=3;
hive (default)> insert overwrite local directory '/opt/module/datas/distribute-
result' select * from emp
distribute by deptno sort by empno desc;
```

Cluster By

```
当 distribute by 和 sorts by 字段相同时，可以使用 cluster by 方式。
cluster by 除了具有 distribute by 的功能外还兼具 sort by 的功能。但是排序只能是升序排
序，不能指定排序规则为 ASC 或者 DESC。 1）以下两种写法等价
hive (default)> select * from emp cluster by deptno;
hive (default)> select * from emp distribute by deptno sort by
deptno;
注意：按照部门编号分区，不一定就是固定死的数值，可以是 20 号和 30 号部门分到一
个分区里面去
```

分桶及抽样查询

分桶表数据存储

```
分区针对的是数据的存储路径；分桶针对的是数据文件。
分区提供一个隔离数据和优化查询的便利方式。不过，并非所有的数据集都可形成合理
的分区，特别是之前所提到过的要确定合适的划分大小这个疑虑。
分桶是将数据集分解成更容易管理的若干部分的另一个技术.
```

```
创建分桶表
create table stu_buck(id int, name string) clustered by(id) into 4 buckets row
format delimited fields terminated by '\t';
查看表结构
hive (default)> desc formatted stu_buck;
Num Buckets: 4
导入数据到分桶表中
hive (default)> load data local inpath '/opt/module/datas/student.txt' into table
stu_buck;
```

创建分桶表时，数据通过子查询的方式导入

```
（1）先建一个普通的 stu 表
create table stu(id int, name string) row format delimited fields terminated by
'\t';
（2）向普通的 stu 表中导入数据
load data local inpath '/opt/module/datas/student.txt' into table stu;
（3）清空 stu_buck 表中数据
truncate table stu_buck;
select * from stu_buck;
（4）导入数据到分桶表，通过子查询的方式
insert into table stu_buck select id, name from stu;
（5）需要设置一个属性
hive (default)> set hive.enforce.bucketing=true;
hive (default)> set mapreduce.job.reduces=-1;
hive (default)> insert into table stu_buck select id, name from stu;
```

分桶抽样查询

```
对于非常大的数据集，有时用户需要使用的是一个具有代表性的查询结果而不是全部结
果。Hive 可以通过对表进行抽样来满足这个需求。
查询表 stu_buck 中的数据。
hive (default)> select * from stu_buck tablesample(bucket 1 out of 4 on id);
```

注：tablesample 是抽样语句，语法：TABLESAMPLE(BUCKET x OUT OF y) 。 y 必须是 table 总 bucket 数的倍数或者因子。hive 根据 y 的大小，决定抽样的比例。例
如，table 总共分了 4 份，当 y=2 时，抽取(4/2=)2 个 bucket 的数据，当 y=8 时，抽取(4/8=)1/2 个 bucket 的数据。
x 表示从哪个 bucket 开始抽取，如果需要取多个分区，以后的分区号为当前分区号加上 y。例如，table 总 bucket 数为 4，tablesample(bucket 1 out of 2)，表示总共抽取（4/2=）2 个 bucket 的数据，抽取第 1(x)个和第 3(x+y)个 bucket 的数据。
注意：x 的值必须小于等于 y 的值，否则
FAILED: SemanticException [Error 10061]: Numerator should not be bigger than denominator in sample clause for table stu_buck

其他常用查询函数

空字段赋值

NVL：给值为 NULL 的数据赋值，它的格式是 NVL(string1, replace_with)。它的功能是如果 string1 为 NULL，则 NVL 函数返回 replace_with 的值，否则返回 string1 的值，如果两个参数都为 NULL ，则返回 NULL。

```
查询：如果员工的 comm 为 NULL，则用-1 代替
hive (default)> select nvl(comm,-1) from emp;
查询：如果员工的 comm 为 NULL，则用领导 id 代替
hive (default)> select nvl(comm,mgr) from emp;
```

时间类

1）date_format:格式化时间

```
hive (default)> select date_format('2019-06-29','yyyy-MM-dd');
OK
_c0
2019-06-29
```

2）date_add:时间跟天数相加

```
- hive (default)> select date_add('2019-06-29',5);
OK
  _c0
2019-07-04
- hive (default)> select date_add('2019-06-29',-5);
  OK
 _c0
 2019-06-24
```

3）date_sub:时间跟天数相减

```
- hive (default)> select date_sub('2019-06-29',5);
OK
_c0
2019-06-24
- hive (default)> select date_sub('2019-06-29 12:12:12',5);
OK
_c0
2019-06-24
- hive (default)> select date_sub('2019-06-29',-5);
OK
_c0
2019-07-04
```

4）datediff:两个时间相减

```
- hive (default)> select datediff('2019-06-29','2019-06-24');
OK
_c0
5
- hive (default)> select datediff('2019-06-24','2019-06-29');
OK
_c0
```

```
-5
- hive (default)> select datediff('2019-06-24 12:12:12','2019-06-29');
OK
_c0
-5
- hive (default)> select datediff('2019-06-24 12:12:12','2019-06-29 13:13:13');
OK
_c0
-5
```