

# 大数据数据库系统

## 6.8 Hive窗口函数

# 分析函数与窗口函数

## ◆ 主要内容

6.8.1 窗口函数的介绍

6.8.2 窗口函数的使用

## 6.8.1窗口函数的介绍

### ◆ 窗口函数

窗口函数可以对分组后的数据进行组内每行的处理，用户可以指定窗口，窗口大小也可随着行的变化而变化

例：select name,orderdate,cost, sum(cost) over(partition by name order by orderdate rows between 1 PRECEDING and 1 FOLLOWING) from business;

### ◆窗口函数格式

Function (arg1,..., argn) OVER ([PARTITION BY <...>] [ORDER BY <....>][ROWS BETWEEN <start\_expr> AND <end\_expr> ])

Function (arg1,..., argn)表示函数及传入的参数

PARTITION BY <...> 对哪一个字段进行分组，和Group By类似

ORDER BY <....>对哪一个字段数据进行排序，默认为升序

ROWS BETWEEN <start\_expr> AND <end\_expr>:

指定窗口的范围，有的函数不支持ROW BETWEEN

例: select name,orderdate,cost, sum(cost) over(partition by name order by orderdate rows between 1 PRECEDING and 1 FOLLOWING) from business;

Function (arg1,..., argn) OVER ([PARTITION BY <...>] [ORDER BY <...>][ROWS BETWEEN <start\_expr> AND <end\_expr> ])

- Function函数包括:
- COUNT计数、SUM 求和、AVG 求平均、MIN/MAX 最小/最大 (支持ROW BETWEEN)
- ROW\_NUMBER 从1开始, 按照顺序, 生成分组内每条记录的序列编号
- RANK 生成数据项在分组中的排名, 排名相等会在名次中留下空位
- DENSE\_RANK 生成数据项在分组中的排名, 排名相等会在名次中不会留下空位
- FIRST\_VALUE取分组内排序后, 截止到当前行, 第一个值
- LAST\_VALUE取分组内排序后, 截止到当前行, 最后一个值
- LEAD用于统计窗口内往后第n行值
- LAG 用于统计窗口内往前第n行值
- NTILE n切分的片数

Function (arg1,..., argn) OVER ([PARTITION BY <...>] [ORDER BY <...>])[ROWS BETWEEN <start\_expr> AND <end\_expr> ])

ROWS BETWEEN表示定义窗口大小，之后可以跟窗口范围包括：

- CURRENT ROW：当前行
- n PRECEDING：当前行往前 n 行数据
- n FOLLOWING：当前行往后 n 行数据
- UNBOUNDED：
  - UNBOUNDED PRECEDING 表示起点
  - UNBOUNDED FOLLOWING 表示终点

如果不指定ROWS BETWEEN，默认为从起点到当前行；

不指定ORDER BY,表示起点到终点

NTILE,ROW\_NUMBER,RANK,DENSE\_RANK不支持ROWS BETWEEN

例子：

rows between unbounded preceding and current row 从起点到当前行,默认

rows between 3 preceding and current row 从当前行的往前3行到当前行

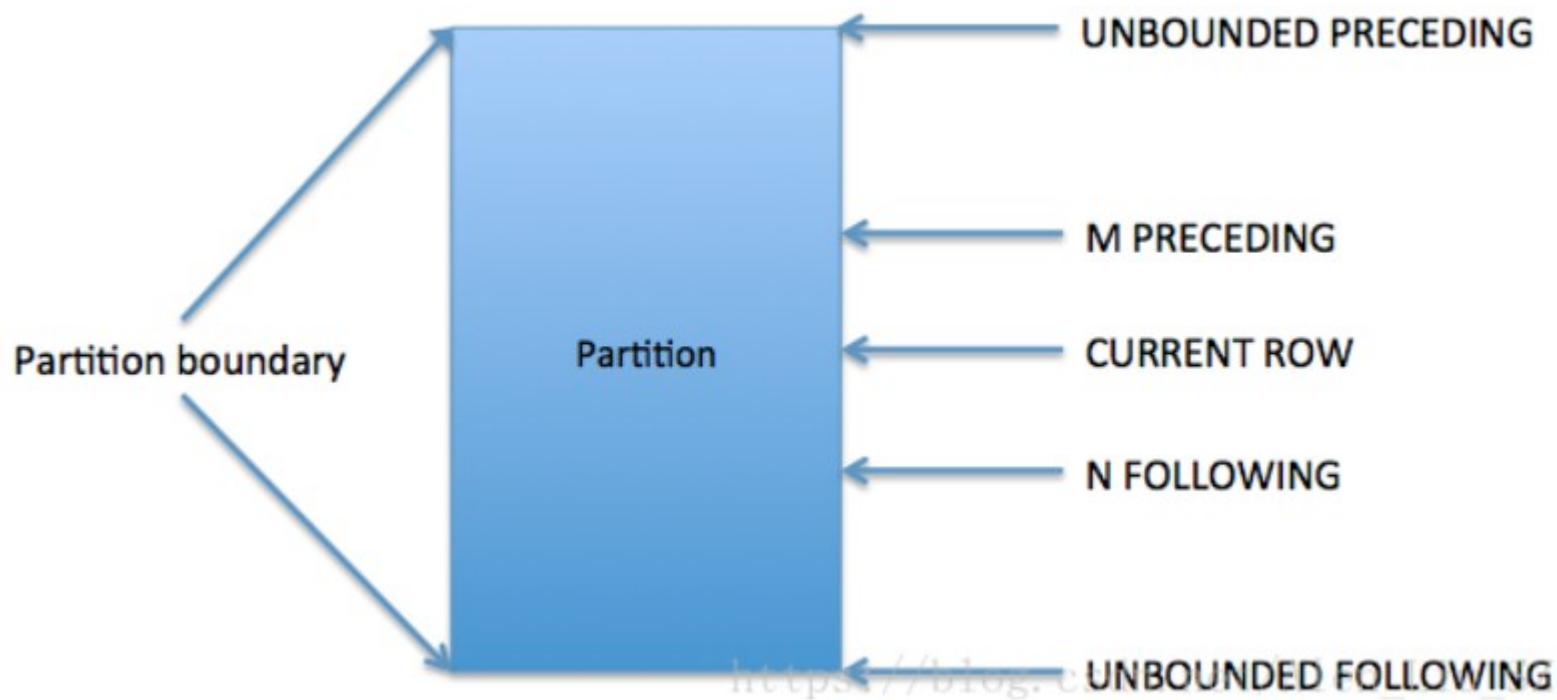
rows between 3 preceding and 1 following 从当前行往前3行到当前行往后1行

rows between current row and unbounded following 从当前行到终点

rows between unbounded preceding and unbounded following 从起点到终点

## 6.8.1 窗口函数的介绍

### ◆ ROWS BETWEEN 范围图示



## 6.8.2 窗口函数的使用

### ◆ 需求

#### 1、对emp表进行薪资降序排列

```
select empno,ename,deptno,sal from emp order by sal desc;
```

empno	ename	deptno	sal
7839	KING	10	5000.0
7902	FORD	20	3000.0
7788	SCOTT	20	3000.0
7566	JONES	20	2975.0
7698	BLAKE	30	2850.0
7782	CLARK	10	2450.0
7499	ALLEN	30	1600.0
7844	TURNER	30	1500.0
7934	MILLER	10	1300.0
7654	MARTIN	30	1250.0
7521	WARD	30	1250.0
7876	ADAMS	20	1100.0
7900	JAMES	30	950.0
7369	SMITH	20	800.0

Time taken: 23.635 seconds, Fetched: 14 row(s)  
hive (emp\_test)>

- 从结果可以看到所有雇员的薪资降序排列，但雇员们来自各个部门，如果我想看到按照进行每个部门的薪资降序排列，并显示序号该怎么做？



## 6.8.2 窗口函数的使用

2、对emp表进行每个部门的薪资降序排列,并显示每个雇员的序号

- `select empno,ename,deptno,sal,ROW_NUMBER() over (partition by deptno order by sal desc) as number from emp;`
  - `ROW_NUMBER()` 从1开始, 按照顺序, 生成分组内记录的序列编号
  - `partition by deptno` 按照部门编号分组
  - `order by sal desc` 按照薪资降序排列
  - `Over` 固定格式, 前面跟函数后面跟括号内容
  - `as number` 对之前的窗口函数字段起个别名



2、对emp表进行每个部门的薪资降序排列,并显示每个雇员的序号

```
select empno,ename,deptno,sal,ROW_NUMBER() over (partition by deptno order by sal desc)
as number from emp;
```

empno	ename	deptno	sal	number	
7839	KING	10	5000.0	1	} 10号部门的
7782	CLARK	10	2450.0	2	
7934	MILLER	10	1300.0	3	
7788	SCOTT	20	3000.0	1	} 20号部门的
7902	FORD	20	3000.0	2	
7566	JONES	20	2975.0	3	
7876	ADAMS	20	1100.0	4	
7369	SMITH	20	800.0	5	
7698	BLAKE	30	2850.0	1	} 30号部门的
7499	ALLEN	30	1600.0	2	
7844	TURNER	30	1500.0	3	
7654	MARTIN	30	1250.0	4	
7521	WARD	30	1250.0	5	
7900	JAMES	30	950.0	6	
Time taken: 30.918 seconds, Fetched: 14 row(s)					
hive (emp_test)> █					

多了一列number, 按照部门分组, 并且部门内按薪资降序排列并按顺序进行编号

执行过程: 先分组, 再排序, 每组内按顺序, 每行执行一次ROW\_NUMBER

问: 如果想获得每个部门薪资前两名的员工该怎么做?

答: 可以将该表保存为临时表, 然后对这个临时表使用where语句, 寻找number小于等于2的数据

### 3、显示每个部门最高的薪资

```
select empno,ename,deptno,sal,max(sal) over (partition by deptno order by sal desc) as  
max_sal from emp;
```

OK					
empno	ename	deptno	sal	max_sal	
7839	KING	10	5000.0	5000.0	10号部门的
7782	CLARK	10	2450.0	5000.0	
7934	MILLER	10	1300.0	5000.0	
7788	SCOTT	20	3000.0	3000.0	20号部门的
7902	FORD	20	3000.0	3000.0	
7566	JONES	20	2975.0	3000.0	
7876	ADAMS	20	1100.0	3000.0	
7369	SMITH	20	800.0	3000.0	
7698	BLAKE	30	2850.0	2850.0	30号部门的
7499	ALLEN	30	1600.0	2850.0	
7844	TURNER	30	1500.0	2850.0	
7654	MARTIN	30	1250.0	2850.0	
7521	WARD	30	1250.0	2850.0	
7900	JAMES	30	950.0	2850.0	
Time taken: 30.795 seconds, Fetched: 14 row(s)					
hive (emp_test)>					

执行过程：先分组，再按薪资排序，没加row between，默认按照每组的开头到当前行，每行都执行从组的开头到当前行取max，例：员工7782行，则取7839，7782最大薪资；员工7844，则取7689、7499、7844行中的最大薪资填入max\_sal

## 6.8.2 窗口函数的使用

### ◆ LEAD函数

`lead(col,value2,value3)`

col: 所要选取的目标字段

value2: 偏移量

value3: 超出窗口的默认值

功能: 显示当前行的后value2行的目标列col的值

例: `select empno,ename,deptno,sal,lead(sal,1,0) over (partition by deptno  
order by sal desc) from emp;`

```
select empno,ename,deptno,sal,lead(sal,1,0) over (partition by deptno order by sal desc)
from emp;
```

```
OK
empno  ename    deptno  sal      _wcol0
7839   KING     10      5000.0   2450.0
7782   CLARK    10      2450.0   1300.0
7934   MILLER   10      1300.0   0.0
-----
7788   SCOTT    20      3000.0   3000.0
7902   FORD     20      3000.0   2975.0
7566   JONES    20      2975.0   1100.0
7876   ADAMS    20      1100.0   800.0
7369   SMITH    20      800.0    0.0
-----
7698   BLAKE    30      2850.0   1600.0
7499   ALLEN    30      1600.0   1500.0
7844   TURNER   30      1500.0   1250.0
7654   MARTIN   30      1250.0   1250.0
7521   WARD     30      1250.0   950.0
7900   JAMES    30      950.0    0.0
-----
Time taken: 30.838 seconds, Fetched: 14 row(s)
hive (emp_test)>
```

lead(sal,1,0) : \_wcol0列显示的是当前行的后1行的sal值，如果后一行超出分组范围（deptno分组），则填入0，如，7934号员工的下一行不在10号部门里了，因此该行填入0

## 6.8.2 窗口函数的使用

### ◆ Lag

`lag(col,value2,value3)`

col: 所要选取的目标列

value2: 偏移量

value3: 超出窗口的默认值

功能: 显示当前行的前value2行的目标列col的值

例: `select empno,ename,deptno,sal,lag(sal,1,0) over (partition by deptno  
order by sal desc) from emp;`

例：select empno,ename,deptno,sal,lag(sal,1,0) over (partition by deptno order by sal desc)  
from emp;

OK					
empno	ename	deptno	sal	_wco10	
7839	KING	10	5000.0	0.0	10号部门的
7782	CLARK	10	2450.0	5000.0	
7934	MILLER	10	1300.0	2450.0	
7788	SCOTT	20	3000.0	0.0	20号部门的
7902	FORD	20	3000.0	3000.0	
7566	JONES	20	2975.0	3000.0	
7876	ADAMS	20	1100.0	2975.0	
7369	SMITH	20	800.0	1100.0	
7698	BLAKE	30	2850.0	0.0	30号部门的
7499	ALLEN	30	1600.0	2850.0	
7844	TURNER	30	1500.0	1600.0	
7654	MARTIN	30	1250.0	1500.0	
7521	WARD	30	1250.0	1250.0	
7900	JAMES	30	950.0	1250.0	
Time taken: 30.357 seconds, Fetched: 14 row(s)					
hive (emp_test)> █					

lag(sal,1,0)：与lead函数方向相反，\_wco10列显示的是当前行的前1行的sal值，如果前一行超出分组范围（deptno分组），则填入0，如员工7839前一行没有值，则填入0；员工7698前一行不在30号部门里，因此也填入0



## ◆ 深入理解

1、准备数据：三个字段分别为name（姓名，类型string）、orderdate（购买时间，类型string）、cost（价格，类型int），存储在opt/module/data/business.txt

jack,2017-01-01,10

tony,2017-01-02,15

jack,2017-02-03,23

tony,2017-01-04,29

jack,2017-01-05,46

jack,2017-04-06,42

tony,2017-01-07,50

jack,2017-01-08,55

mart,2017-04-08,62

mart,2017-04-09,68

neil,2017-05-10,12

mart,2017-04-11,75

neil,2017-06-12,80

mart,2017-04-13,94



## 6.8.2 窗口函数的使用

### 2、构造表并导入数据

```
create table business(  
    name string,  
    orderdate string,  
    cost int  
)  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';  
load data local inpath "/opt/module/data/business.txt" into table  
business;
```

## 6.8.2 窗口函数的使用

### ◆ 需求

- (1) 查询每个顾客的购买明细及购买总额
- (2) 查询每个顾客的每个月的购买总额
- (3) 查询每个月的购买总额
- (4) 上述的场景, 将每个顾客的 cost 按照日期进行累加
- (5) 查询每个顾客上次的购买时间

jack,2017-01-01,10  
tony,2017-01-02,15  
jack,2017-02-03,23  
tony,2017-01-04,29  
jack,2017-01-05,46  
jack,2017-04-06,42  
tony,2017-01-07,50  
jack,2017-01-08,55  
mart,2017-04-08,62  
mart,2017-04-09,68  
neil,2017-05-10,12  
mart,2017-04-11,75  
neil,2017-06-12,80  
mart,2017-04-13,94

## 1、查询每个顾客的购买总额

```
select name,orderdate,cost,sum(cost) over(partition by name) from business;
```

```
OK
name      orderdate      cost      sum_window_0
jack      2017-01-05      46        176
jack      2017-01-08      55        176
jack      2017-01-01      10        176
jack      2017-04-06      42        176
jack      2017-02-03      23        176
mart      2017-04-13      94        299
mart      2017-04-11      75        299
mart      2017-04-09      68        299
mart      2017-04-08      62        299
neil      2017-05-10      12        92
neil      2017-06-12      80        92
tony      2017-01-04      29        94
tony      2017-01-02      15        94
tony      2017-01-07      50        94
Time taken: 20.115 seconds, Fetched: 14 row(s)
hive (default)>
```

$46+55+10+42+23=176$

$94+75+68+62=299$

按照姓名分组，由于sum支持row between，但不指定order by，表示窗口范围是分组的起点到终点，所以每行最后一个字段的值，是分组的cost之和

## 2、查询每个顾客的每个月的购买总额

```
select name,orderdate,cost,sum(cost) over (partition by  
name,month(orderdate)) from business;
```

OK

name	orderdate	cost	sum_window_0
jack	2017-01-05	46	111
jack	2017-01-08	55	111
jack	2017-01-01	10	111
jack	2017-02-03	23	23
jack	2017-04-06	42	42
mart	2017-04-13	94	299
mart	2017-04-11	75	299
mart	2017-04-09	68	299
mart	2017-04-08	62	299
neil	2017-05-10	12	12
neil	2017-06-12	80	80
tony	2017-01-04	29	94
tony	2017-01-02	15	94
tony	2017-01-07	50	94

Time taken: 21.012 seconds, Fetched: 14 row(s)  
hive (default)> █

Annotations in the original image:  
- A red bracket groups the first three rows (jack, 2017-01-\*) with the calculation  $46+55+10=111$ .  
- A red bracket groups the last three rows (tony, 2017-01-\*) with the calculation  $29+15+50=94$ .

partition by之后跟了两个字段，一个name，一个月份，分组变为  
(jack, 01月)、(jack, 02月)、(jack, 04月)、(mart, 04月)、  
(neil, 05月)、(neil, 06月)、(tony, 07月)，然后每组内求cost和（窗口范围依然是是分组的起点到终点）

### 3、查询每个月的购买总额

```
select name,orderdate,cost,sum(cost) over (partition by month(orderdate))  
from business;
```

```
OK  
name      orderdate      cost      sum_window_0  
jack      2017-01-01     10        205  
jack      2017-01-08     55        205  
tony      2017-01-07     50        205  
jack      2017-01-05     46        205  
tony      2017-01-04     29        205  
tony      2017-01-02     15        205  
jack      2017-02-03     23        23  
mart      2017-04-13     94        341  
jack      2017-04-06     42        341  
mart      2017-04-11     75        341  
mart      2017-04-09     68        341  
mart      2017-04-08     62        341  
neil      2017-05-10     12        12  
neil      2017-06-12     80        80  
Time taken: 20.708 seconds, Fetched: 14 row(s)  
hive (default)>
```

#### 4、 将每个顾客的当日 cost 与之前的cost进行累加

```
select name,orderdate,cost, sum(cost) over(partition by name order by orderdate)
from business;
```

```
OK
name      orderdate      cost      sum_window_0
jack      2017-01-01      10        10
jack      2017-01-05      46        56 ← 56=10+46
jack      2017-01-08      55        111 ← 111=10+46+55
jack      2017-02-03      23        134
jack      2017-04-06      42        176
mart      2017-04-08      62        62
mart      2017-04-09      68        130
mart      2017-04-11      75        205
mart      2017-04-13      94        299 ← 299=62+68+75+94
neil      2017-05-10      12        12
neil      2017-06-12      80        92
tony      2017-01-02      15        15
tony      2017-01-04      29        44
tony      2017-01-07      50        94
Time taken: 22.327 seconds, Fetched: 14 row(s)
hive (default)>
```

首先按照名字分组，窗口范围就是每个分组的起点到当前行（省略了row between 语句），因此每行的最后一个字段就是起点到当前行的cost累加

上述语句等同于： select name,orderdate,cost, sum(cost) over(partition by name order by orderdate **rows between UNBOUNDED PRECEDING and current row**) from business;

## 6.8.2 窗口函数的使用

ROWS BETWEEN表示定义窗口大小，之后可以跟窗口范围包括：

- CURRENT ROW：当前行
- **n PRECEDING**：当前行**往前 n 行数据**
- **n FOLLOWING**：当前行**往后 n 行数据**
- UNBOUNDED：起点，
  - UNBOUNDED PRECEDING 表示从前面的起点，
  - UNBOUNDED FOLLOWING 表示到后面的终点
- **NTILE,ROW\_NUMBER,RANK,DENSE\_RANK不支持ROWS BETWEEN**



例1: select name,orderdate,cost, sum(cost) over(partition by name order by  
orderdate rows between 1 PRECEDING and 1 FOLLOWING) from  
business;

这段语句执行后会有什么效果呢?

name	orderdate	cost	sum_window_0
jack	2017-01-01	10	56 ← $56=0+10+46$
jack	2017-01-05	46	111 ← $111=10+46+55$
jack	2017-01-08	55	124 ← $124=46+55+23$
jack	2017-02-03	23	120 ← $120=55+23+42$
jack	2017-04-06	42	65 ← $65=23+42+0$
mart	2017-04-08	62	130
mart	2017-04-09	68	205
mart	2017-04-11	75	237
mart	2017-04-13	94	169
neil	2017-05-10	12	92
neil	2017-06-12	80	92
tony	2017-01-02	15	44
tony	2017-01-04	29	94
tony	2017-01-07	50	79

Time taken: 21.86 seconds, Fetched: 14 row(s)  
hive (default)>

rows between 1 PRECEDING and 1 FOLLOWING表示窗口大小是分组内,当前行往上一行到当前行的往下1行,每行的最后一个字段就为窗口中3行数据的cost之和,超过分组边界的就记cost=0

例2: `select name,orderdate,cost, sum(cost) over(order by orderdate) from business;`

这段语句执行后会有什么效果呢?

```
OK
name      orderdate      cost      sum_window_0
jack      2017-01-01      10        10
tony      2017-01-02      15        25
tony      2017-01-04      29        54
jack      2017-01-05      46        100
tony      2017-01-07      50        150
jack      2017-01-08      55        205
jack      2017-02-03      23        228
jack      2017-04-06      42        270
mart      2017-04-08      62        332
mart      2017-04-09      68        400
mart      2017-04-11      75        475
mart      2017-04-13      94        569
neil      2017-05-10      12        581
neil      2017-06-12      80        661
Time taken: 21.514 seconds, Fetched: 14 row(s)
hive (default)>
```

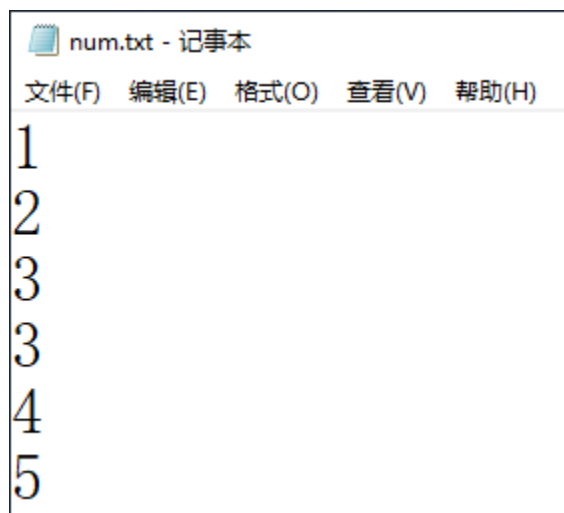
没有partition by 语句, 即不进行分组, 按orderdate升序排列, 窗口为起点到当前行, 因此每行的最后一个字段就为起点到当前行cost的累加

## 6.8.2 窗口函数的使用

➤ 需要注意一点，如果有这么一段数据

建表：create table num(id int);

数据如下



```
num.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
1
2
3
3
4
5
```

此时，如果使用 `select id ,sum(id) over(order by id) from num;` 会出什么结果？

## 6.8.2 窗口函数的使用

```
select id ,sum(id) over(order by id) from num;
```

```
id      sum_window_0
1        1
2        3
3        9
3        9
4       13
5       18
Time taken: 20.583 seconds, Fetched: 6 row(s)
hive (default)>
```

因为hive认为第3行的id=3与第4行的id=3是同一个数据，所以窗口就包含了这两行，把这两行作为“当前行”与之前的数据累加（可以看作以最后一行代表前面的行），所以“当前行”并不只代表一行，可以是多行

如果要区分这两个id=3，则考虑给这个表再加一个字段，比如：行号等，再使用partition by语句按行号分组

例：查看每个顾客上次的购买时间

```
select name,orderdate,cost,lag(orderdate,1) over(partition by name order  
by orderdate ) from business;
```

```
OK
name      orderdate      lag_window_0
jack      2017-01-01      NULL
jack      2017-01-05      2017-01-01
jack      2017-01-08      2017-01-05
jack      2017-02-03      2017-01-08
jack      2017-04-06      2017-02-03
mart      2017-04-08      NULL
mart      2017-04-09      2017-04-08
mart      2017-04-11      2017-04-09
mart      2017-04-13      2017-04-11
neil      2017-05-10      NULL
neil      2017-06-12      2017-05-10
tony      2017-01-02      NULL
tony      2017-01-04      2017-01-02
tony      2017-01-07      2017-01-04
Time taken: 21.72 seconds, Fetched: 14 row(s)
hive (default)>
```

考虑之前学的LAG函数，获得当前行上一行的指定字段，又由于是每个顾客，因此按照顾客分组

## 6.8.2 窗口函数的使用

### ◆ RANK、DENSE\_RANK、ROW\_NUMBER()函数

RANK() 生成数据项在分组中的排名，排名相等会在名次中留下空位

DENSE\_RANK() 生成数据项在分组中的排名，排名相等在名次中不会留下空位

ROW\_NUMBER,RANK,DENSE\_RANK不支持ROWS BETWEEN

数据准备:

name	subject	score
孙悟空	语文	87
孙悟空	数学	95
孙悟空	英语	68
大海	语文	94
大海	数学	56
大海	英语	84
宋宋	语文	64
宋宋	数学	86
宋宋	英语	84
婷婷	语文	65
婷婷	数学	85
婷婷	英语	78

create table score(

name string,

subject string,

score int)

row format delimited fields terminated by "\t";

load data local inpath '/opt/module/data/score.txt' into table score;



## 例：计算成绩排名，分别用RANK、DENSE\_RANK对比结果

```
select *, rank() over(order by score) from score;
```

score.name	score.subject	score.score	rank_window_0
大海	数学	56	1
宋宋	语文	64	2
婷婷	语文	65	3
孙悟空	英语	68	4
婷婷	英语	78	5
宋宋	英语	84	6
大海	英语	84	6
婷婷	数学	85	8
宋宋	数学	86	9
孙悟空	语文	87	10
大海	语文	94	11
孙悟空	数学	95	12

Time taken: 23.898 seconds, Fetched: 12 row(s)

hive (default)> █

```
select *, dense_rank() over(order by score) from score;
```

score.name	score.subject	score.score	dense_rank_window_0
大海	数学	56	1
宋宋	语文	64	2
婷婷	语文	65	3
孙悟空	英语	68	4
婷婷	英语	78	5
宋宋	英语	84	6
大海	英语	84	6
婷婷	数学	85	7
宋宋	数学	86	8
孙悟空	语文	87	9
大海	语文	94	10
孙悟空	数学	95	11

Time taken: 22.195 seconds, Fetched: 12 row(s)

hive (default)> █

## 例：计算各学科成绩排名

```
select *, rank() over(partition by subject order by score asc) from score;
```

OK

score.name	score.subject	score.score	rank_window_0
大海	数学	56	1
婷婷	数学	85	2
宋宋	数学	86	3
孙悟空	数学	95	4
孙悟空	英语	68	1
婷婷	英语	78	2
宋宋	英语	84	3
大海	英语	84	3
宋宋	语文	64	1
婷婷	语文	65	2
孙悟空	语文	87	3
大海	语文	94	4

Time taken: 20.681 seconds, Fetched: 12 row(s)

hive (default)> █

例：求出每门学科前三名的学生？

利用上一个例子求出的各学科成绩排名表

```
select *, rank() over(partition by subject order by score desc) rk from score;
```

```
select name, subject, score from ( ) t1 where rk <=3;
```



```
select name, subject, score
```

```
from
```

```
(select *, rank() over(partition by subject order by score desc) rk from score )
```

```
t1
```

```
where rk <=3;
```

注意：在HIVE中，子查询必须有别名！

select name, subject, score

from

(select \*, rank() over(partition by subject order by score desc) rk from

score ) t1

where rk <=3;

```
OK
name      subject  score
孙悟空    数学      95
宋宋      数学      86
婷婷      数学      85
大海      英语      84
宋宋      英语      84
婷婷      英语      78
大海      语文      94
孙悟空    语文      87
婷婷      语文      65
Time taken: 26.566 seconds, Fetched: 9 row(s)
hive (default)>
```

# 窗口函数与Group by区别

◆ 窗口函数可以视为group by的升级版：

- 会保留所有进入分区的数据，不去重（没有数据损失）
- 窗口函数可以进行更为复杂个性化的操作

# 总结

- ◆ `Function (arg1,..., argn) OVER ([PARTITION BY <...>] [ORDER BY <...>][ROWS BETWEEN <start_expr> AND <end_expr> ])`

`Function (arg1,..., argn)`表示函数及传入的参数

`PARTITION BY <...>` 对哪一个字段进行分组，和Group By类似

`ORDER BY <...>`对哪一个字段数据进行排序，默认为升序

`ROWS BETWEEN <start_expr> AND <end_expr>`:

指定窗口的范围，有的函数不支持ROW BETWEEN

# 总结

ROWS BETWEEN表示定义窗口大小，之后可以跟窗口范围包括：

- CURRENT ROW：当前行
- n PRECEDING：当前行往前 n 行数据
- n FOLLOWING：当前行往后 n 行数据
- UNBOUNDED：起点，
  - UNBOUNDED PRECEDING 表示从前面的起点，
  - UNBOUNDED FOLLOWING 表示到后面的终点

如果不指定ROWS BETWEEN,默认为从起点到当前行；

不指定ORDER BY,表示起点到终点



# 总结

