目录

SQL 脚本节点使用语法说明

注：本文主要是针对 Smartbi Mining 的 SQL 脚本节点写的使用帮助文档，语法主要以 Spark SQL 为主。

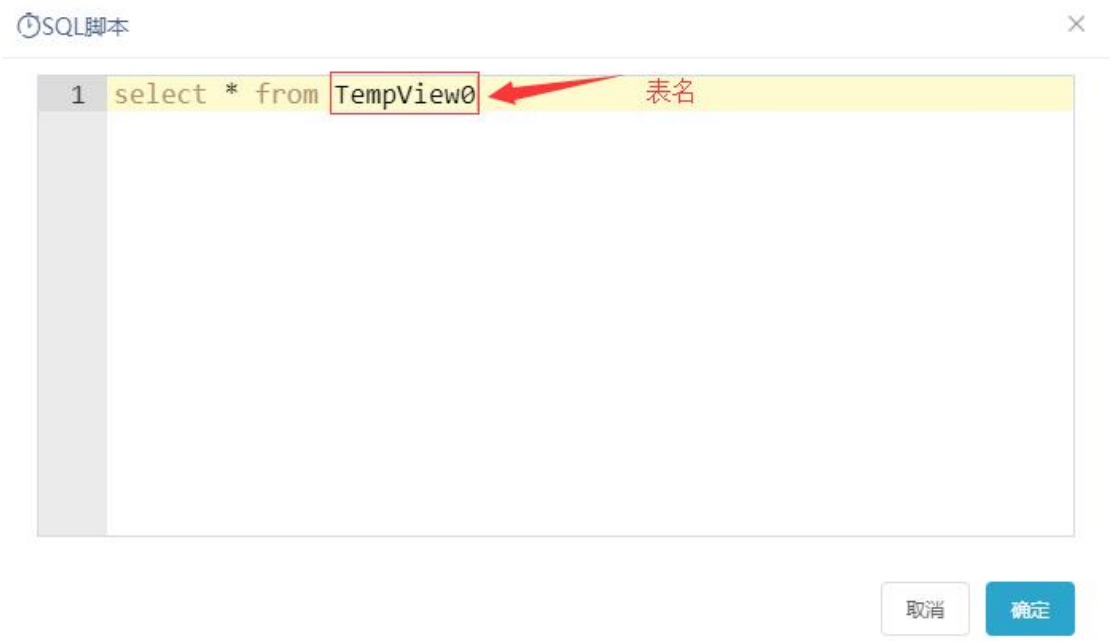
一、常用操作符

=：等值比较
语法：A=B。如果表达式 A 与表达式 B 相等，则为 TRUE；否则为 FALSE
<>：不等值比较
语法: A <> B。如果表达式 A 为 NULL，或者表达式 B 为 NULL，返回 NULL；如果表达式 A 与表达式 B 不相等，则为 TRUE；否则为 FALSE
<：小于比较
语法: A < B。 如果表达式 A 为 NULL，或者表达式 B 为 NULL，返回 NULL；如果表达式 A 小于表达式 B，则为 TRUE；否则为 FALSE
<=：小于等于比较
语法: A <= B。如果表达式 A 为 NULL，或者表达式 B 为 NULL，返回 NULL；如果表达式 A 小于或者等于表达式 B，则为 TRUE；否则为 FALSE
>=：大于等于比较
语法: A >= B。如果表达式 A 为 NULL，或者表达式 B 为 NULL，返回 NULL；如果表达式 A 大于或者等于表达式 B，则为 TRUE；否则为 FALSE
IS NULL：空值判断
语法: A IS NULL。如果表达式 A 的值为 NULL，则为 TRUE；否则为 FALSE
IS NOT NULL：非空判断:
语法: A IS NOT NULL。 如果表达式 A 的值为 NULL，则为 FALSE；否则为 TRUE
LIKE：模糊匹配
语法: A[NOT] LIKE B。如果字符串 A 或者字符串 B 为 NULL，则返回 NULL；如果字符串 A 符合表达式 B 的正则语法，则为 TRUE；否则为 FALSE。B 中字符”_”表示任意单个字符，而字符”%”表示任意数量的字符。
AND：逻辑与
语法：A AND B。如果 A 为 TRUE,B 为 TRUE，则为 TRUE，否则 FALSE。
OR：逻辑或
语法：A OR B。如果 A,B 至少有一个为 TRUE，则为 TRUE，否则 FALSE。

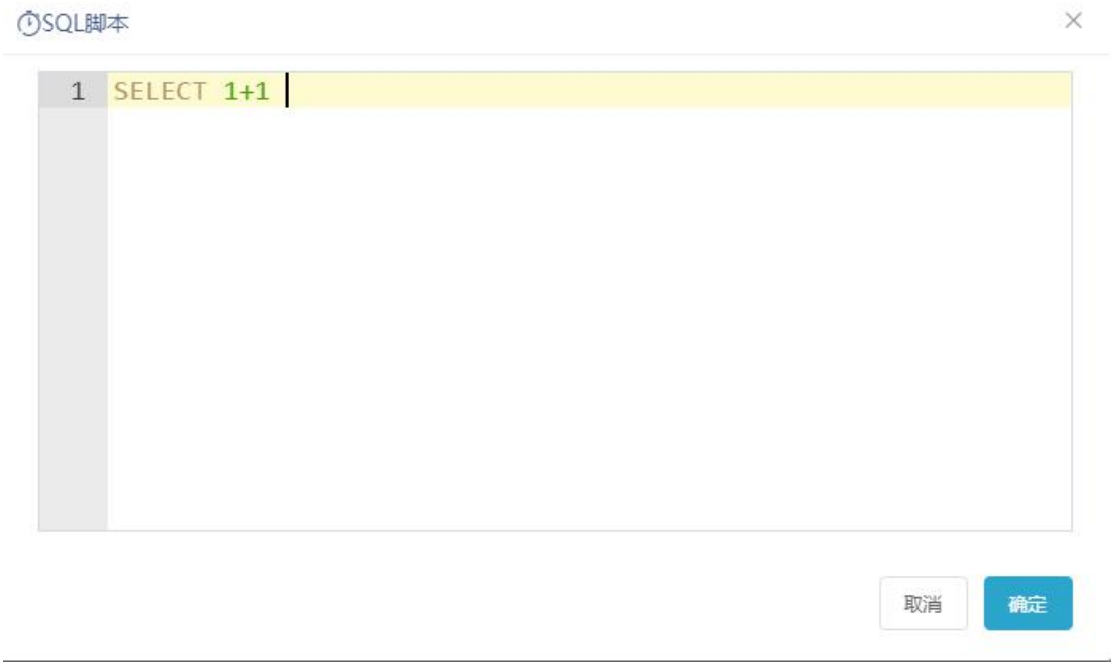
二、常用关键字

2.1 SELECT 查询

示例：select * from TempView0



示例：SELECT 1+1



2.2 AS 更改字段名

示例：select Col0 as id from TempView0

2.3 WHERE 条件语句

操作符：=
<
<=
>
>=
<>
AND
OR
LIKE

示例：select Col0,Col1,Col2 from TV where Col0=1392 and Col1=19503

示例：select Col0 from TV where Col0 like "%39%"

2.4 DISTINC 去重

DISTINC：指定从结果集中删除重复的行
示例：SELECT DISTINCT col0 FROM TV

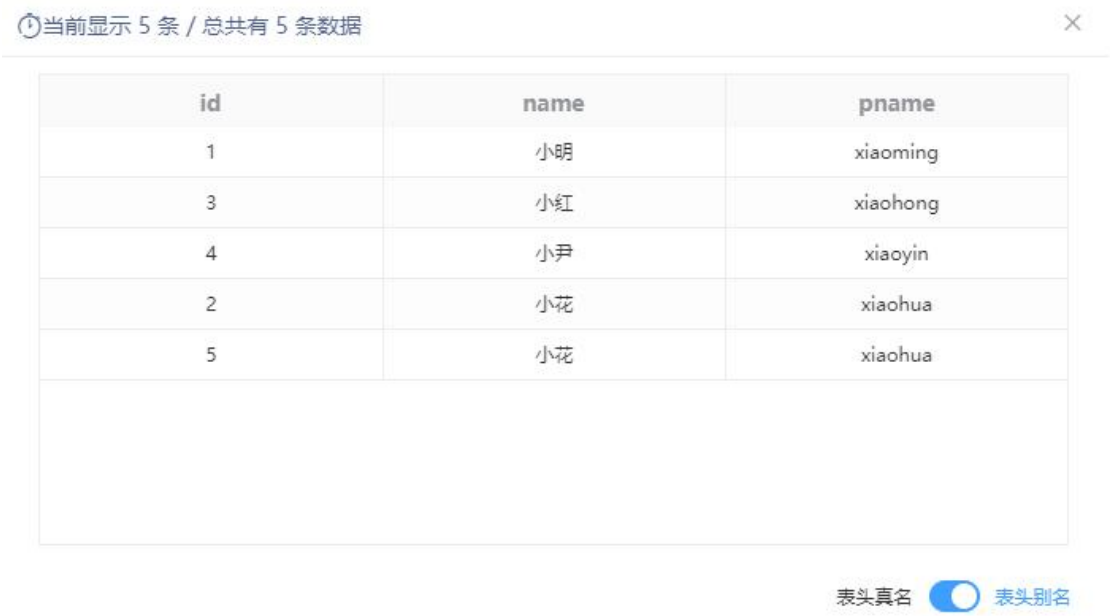
2.5 GROUP BY 分组

注：尽量避免使用 DISTINCT 进行排重，特别是大表操作，用 GROUP BY 代替。
示例：select Col0 from TV group by Col0

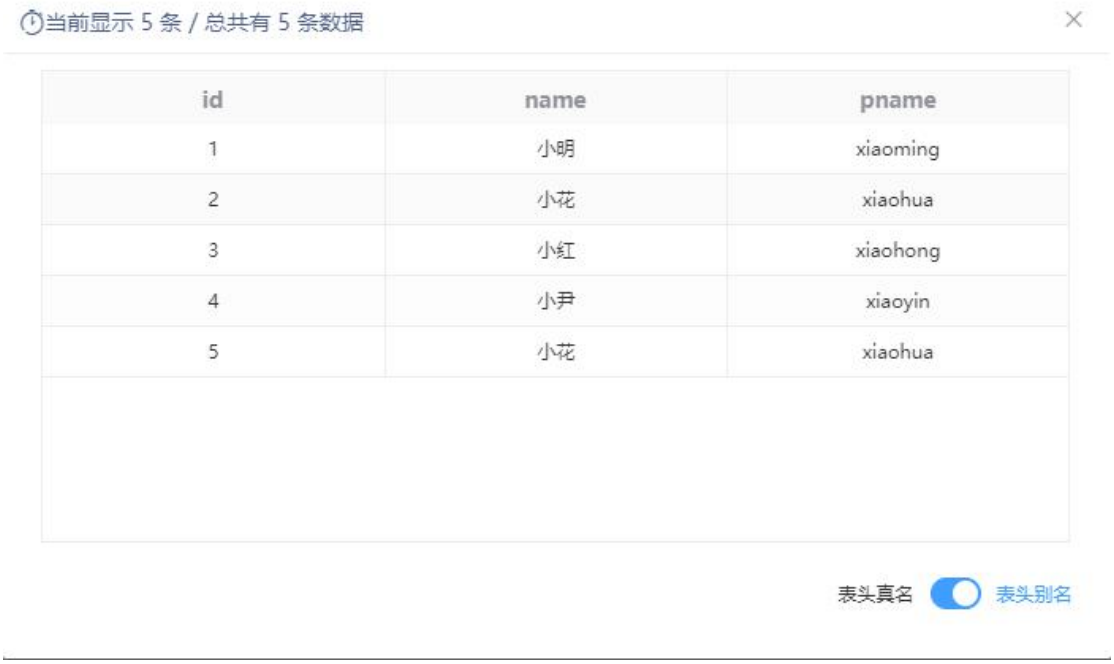
2.6 排序

2.6.1 ORDER BY

Order by 对查询结果集执行一个全局排序，这也就是所有的数据都通过一个 reduce 进行处理的过程，对于大数据量的数据集，这个过程将消耗很大的时间来执行。
desc：降序。
asc：升序。
示例 1：select * from TV order by id desc
示例 2：select * from TV order by id asc



示例 1，示例 2 返回结果集分别如下图所示：



2.6.2 SORT BY 和 DISTRIBUTE BY

sort by 其实会在每个 reduce 中对数据进行排序，也就是执行一个局部排序过程，每个 reducer 出来的数据是有序的，但是不能保证所有的数据是有序的，除非只有一个 reducer），好处是执行了局部排序之后可以为接下去的全局排序提高不少的效率。

ditribute by 是控制 map 的输出在 reducer 是如何划分的。

注：一般 ditribute by 和 sort by 是一起使用的，ditribute by 的功能有点类似 group by。

示例：select * from TV distribute by name sort by name,money asc

原始表 TV 如下图所示：

返回结果集如下图所示：

当前显示 6 条 / 总共有 6 条数据

name	money	shop
B	78	商店1
B	88	商店2
C	58	商店1
A	57	商店3
A	69	商店2
A	80	商店1

表头真名 表头别名

2.6.3 CLUSTER BY

cluster by 的功能就是 distribute by 和 sort by 相结合。

示例 1：select * from TV distribute by name sort by name

示例 2：select * from TV cluster by name

注：示例 1 和示例 2 的 SQL 语句是等价的，它们返回的结果集也是一样的，被 cluster by 指定的列只能是降序。

原始表 TV 如下图所示：

当前显示 6 条 / 总共有 6 条数据

name	money	shop
A	80	商店1
B	78	商店1
B	88	商店2
C	58	商店1
A	69	商店2
A	57	商店3

表头真名 表头别名

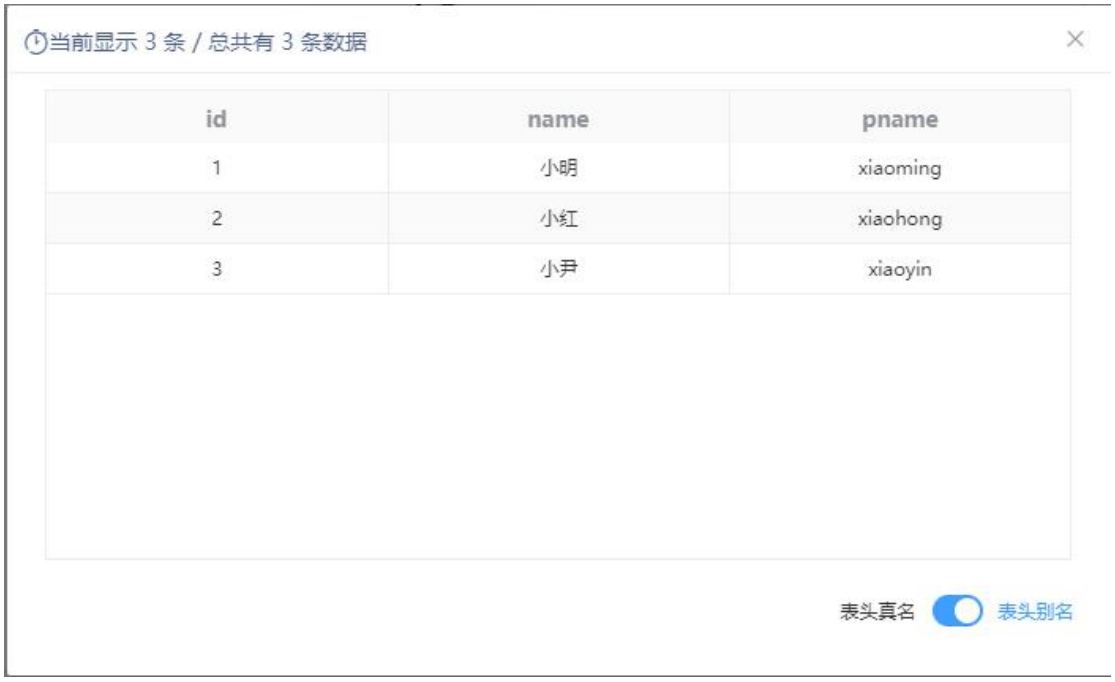
示例 1，实例 2 返回的结果集分别如下图所示：

2.7 约束返回行数

LIMIT：可用于约束 SELECT 语句返回的行数。
BETWEEN ... AND：会选取介于两个值之间的数据范围，可用于对有序列号的数据集进行行区间选择。
注：BETWEEN ... AND 用于行区间选择的前提是有自增序列号的时候。
示例 1：select * from TV limit 3
示例 2：select * from TV where id between 2 and 3

原始表 TV 如下图所示：

示例 1，示例返回结果集分别如下二个图所示：



2.8 表连接

表 t1

表 t2

当前显示 5 条 / 总共有 5 条数据

id_no	score
2	99
2	95
3	88
4	100
6	90

表头真名 表头别名

2.8.1 左连接和左外连接及左半连接

LEFT JION（左连接）和 LEFT OUTER JION（左外连接）：规则根据条件以左表为基准进行匹配，如果匹配则正常输出，没有则以 null 代替，而且右边表有重复值的情况下会输出多条。

示例 1：select * from t1 left join t2 on t1.id = t2.id_no

示例 2：select * from t1 left outer join t2 on t1.id = t2.id_no

LEFT JION 和 LEFT OUTER JION 输出结果如下图所示，它们的输出结果是一样的。

当前显示 6 条 / 总共有 6 条数据

id	name	age	id_no	score
1	小明	6	null	null
2	小红	6	2	95
2	小红	6	2	99
3	小尹	7	3	88
4	小花	7	4	100
5	小飞	7	null	null

表头真名 表头别名

LEFT SEMI JOIN（左半连接）

LEFT SEMI JOIN：只显示左边的表中的列，规则如果条件在左右边表中匹配，则输出，否则过滤掉，而且右边表有重复值的情况下也只输出一条。

示例：select * from t1 left semi join t2 on t1.id = t2.id_no

2.8.2 内连接

INNER JOIN 或 JOIN：规则如果条件在左右边表中都匹配则，左右表中有重复值的时候会输出多条。
示例 1：select * from t1 inner join t2 on t1.id = t2.id_no
实例 2：select * from t1 join t2 on t1.id = t2.id_no

2.8.3 右连接和右外连接

注：左连接、右外连接和右连接、右外连接规则类似。
RIGHT JION（右连接） 和 RIGHT OUTER JION（右外连接）：规则根据条件以右表为基准进

行匹配，如果匹配则正常输出，没有则以 null 代替，而且左边表有重复值的情况下会输出多条。

示例 1：select * from t1 right join t2 on t1.id = t2.id_no

示例 2：select * from t1 right outer join t2 on t1.id = t2.id_no

RIGHT JION 和 RIGHT OUTER JION 输出结果如下图所示，它们的输出结果是一样的。

id	name	age	id_no	score
2	小红	6	2	99
2	小红	6	2	95
3	小尹	7	3	88
4	小花	7	4	100
null	null	null	6	90

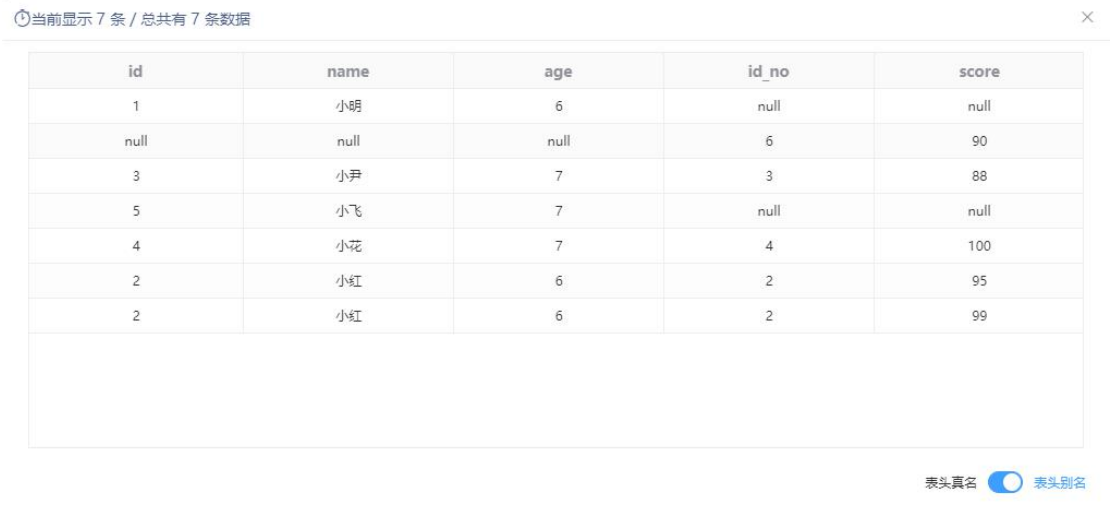
2.8.4 全连接和全外连接

FULL JOIN(全连接)和 FULL OUTER JOIN(全外连接)：规则根据条件以左右表皆为基准进行匹配，如果匹配则正常输出，没有则以 null 代替，左右表有重复值的情况下会输出多条。

示例 1：select * from t1 full join t2 on t1.id = t2.id_no

示例 2：select * from t1 full outer join t2 on t1.id = t2.id_no

FULL JION 和 FULL OUTER JION 输出结果如下图所示，它们的输出结果是一样的。



id	name	age	id_no	score
1	小明	6	null	null
null	null	null	6	90
3	小尹	7	3	88
5	小飞	7	null	null
4	小花	7	4	100
2	小红	6	2	95
2	小红	6	2	99

2.9 UNION 和 UNION ALL 联合

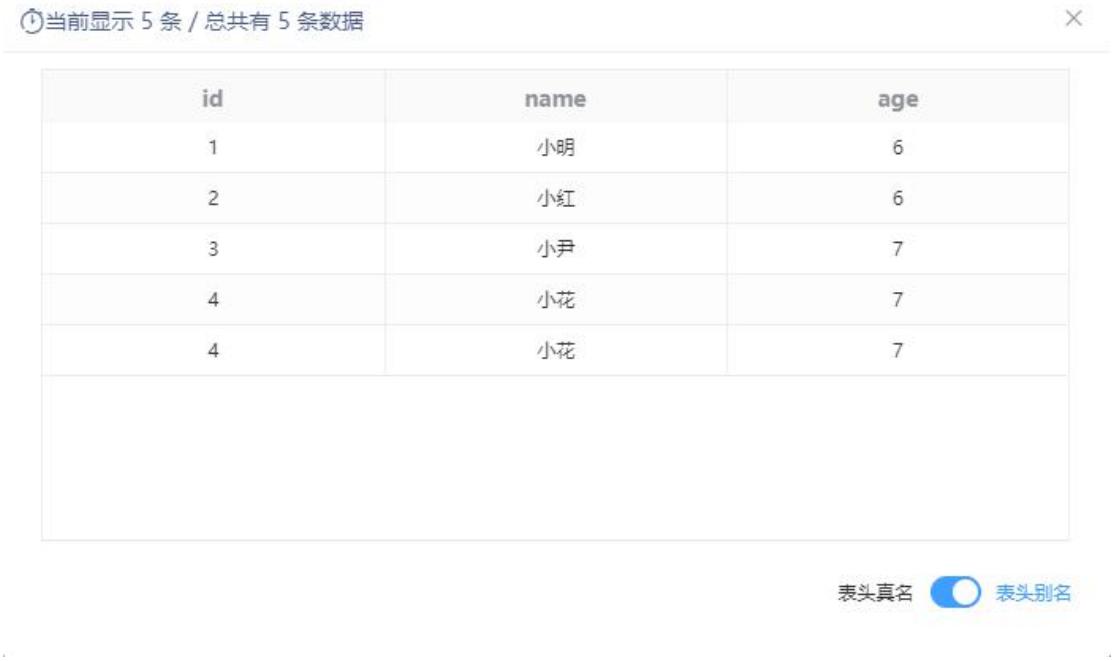
UNION：会去除完全重复的。如果不为了去除重复行，建议使用 UNION ALL，并且效率会更

高。
示例：

```
select* from (
select 2 as id_no,99 as score
union
select 2 ,95
) Student_Score
```

UNION ALL ：对查询的数据集取并集，不会去除重复行。
示例：

```
select* from (
select 1 as id,"小明" as name,6 as age
union all
select 2 ,"小红" ,6
union all
select 3 ,"小尹" ,7
union all
select 4 ,"小花" ,7
union all
select 4 ,"小花" ,7
) Student
```



2.10 HAVING

HAVING：根据 Having 关键字后面指定的筛选条件，将分组后不满足条件的记录筛选掉。
示例：select id from TV group by id having id>2
表 TV 原始数据

查询后数据

三、函数

3.1 字符串函数

3.1.1 字符串的字符数

CHAR_LENGTH(s)：返回字符串 s 的字符数
CHARACTER_LENGTH(s)：返回字符串 s 的字符数
LENGTH(S)：返回字符串 s 的字符数
示例 1：select id,CHAR_LENGTH(name) as name,CHAR_LENGTH(pname) as pname from TV
示例 2：select id,CHARACTER_LENGTH(name) as name,CHAR_LENGTH(pname) as pname from TV
示例 3：select id,length(name) as name,length(pname) as pname from TV
原始表为 TV 和返回的结果集如下图所示，函数返回的结果集是一样的。

3.1.2 字符拼接

CONCAT(s1,s2...sn)：字符串 s1,s2... 等多个字符串合并为一个字符串。
示例：select concat(id,name,pname) as id_name_pname from TV
原始表为 TV 和返回的结果集如下图所示：

CONCAT_WS(split, s1,s2...sn)：同 CONCAT(s1,s2,...) 函数，但是每个字符串直接要加上 split，split 一般是分隔符。

示例：select CONCAT_WS("-",id,name,pname) as id_name_pname from TV

原始表为 TV 和返回的结果集如下图所示：

3.1.3 字符串反转

REVERSE(S)：返回字符串 S 的反转结果。
示例：select reverse(name) as name,reverse(pname) as pname from TV
原始表为 TV 和返回的结果集如下图所示：

3.1.4 字符串截取

SUBSTR(S, START)：返回字符串 S 从 START 位置到结尾的字符串。
SUBSTRING(S, START)：返回字符串 S 从 START 位置到结尾的字符串。
示例 1：select id,substr(name, 2) as name,substr(pname, 5) as pname from TV
示例 2：select id,substring(name, 2) as name,substring(pname, 5) as pname from TV
原始表为 TV 和返回的结果集如下图所示：

当前显示 5 条 / 总共有 5 条数据

id	name	pname
1	小明	xiaoming
2	小红	xiaohong
3	小尹	xiaoyin
4	小花	xiaohua
5	小花	xiaohua

表头真名 表头别名

当前显示 5 条 / 总共有 5 条数据

id	name	pname
1	明	ming
2	红	hong
3	尹	yin
4	花	hua
5	花	hua

表头真名 表头别名

SUBSTR(S, START,LEN)：返回字符串 S 从 START 位置开始，长度为 LEN 的字符串。
SUBSTRING(S, START,LEN)：返回字符串 S 从 START 位置开始，长度为 LEN 的字符串。
示例 1：select id,substr(name,2,1) as name,substr(pname,5,2) as pname from TV
示例 2：select id,substring(name,2,1) as name,substring(pname,5,2) as pname from TV
原始表为 TV 和返回的结果集如下图所示：

3.1.5 字符串大小写转换

UPPER(S)：将字符串 S 转成大写。
UCASE(S)：将字符串 S 转成小写。
示例 1：select upper('XiaoMing') as name
示例 2：select ucase('XiaoMing') as name
返回的结果集如下图所示：

LOWER(S)：将字符串 S 转成小写。
示例 1：select lower('XiaoMing') as name
示例 2：select lcase('XiaoMing') as name
返回的结果集如下图所示：

3.1.6 重复字符串

REPEAT(S,N)：返回重复字符串 S 的次数 N。
示例：select repeat('xiao 小',2)

返回的结果集如下图所示：

3.1.7 补足字符串

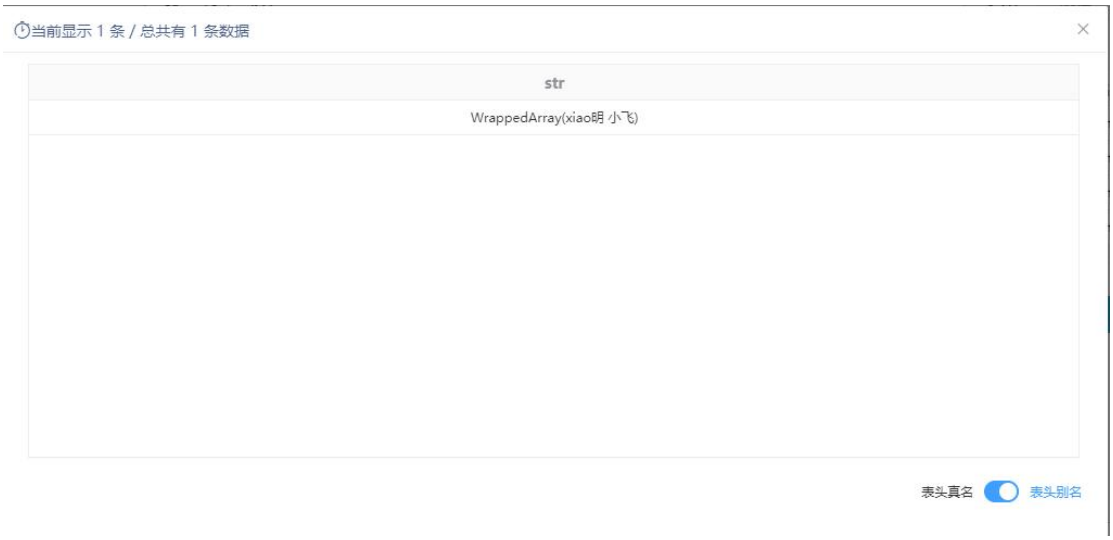
LPAD(S1, LEN, S2)：左补足，将 S1 进行用 S2 进行左补足到 LEN 位。
示例：select lpad('xiao 小', 10, '大 pad') as Str
返回的结果集如下图所示：

RPAD(S1, LEN, S2)：右补足，将 S1 进行用 S2 进行右补足到 LEN 位。
示例：select rpad('xiao 小', 10, '大 pad') as Str

返回的结果集如下图所示：

3.1.8 分割字符串

SPLIT(S, SP)：按照 SP 字符串分割 S，会返回分割后的字符串数组。
示例：select split('xiao 明 xiao 小小飞','xiao 小') as str
返回的结果集如下图所示。



3.1.9 集合字符串查找

FIND_IN_SET(S, SL)：返回字符串 S 在字符串 SL 第一次出现的位置，SL 是用逗号分割的字符串。如果没有找该 S 字符串，则返回 0。
示例：select find_in_set('a 小 b','cd,ef,a 小 b,de') as str

返回的结果集如下图所示：

3.1.10 字符串位置

POSITION(S1 IN S)：返回字符串 S1 在字符串 S 中开始出现的位置，没有则返回 0。
示例：select POSITION('b' IN 'abcb') as str
返回的结果集如下图所示：

3.1.11 字符串替换

REGEXP_REPLACE(S1, S2, S3)：将字符串 S1 中的符合 java 正则表达式 S2 的部分替换为 S3。
示例：select regexp_replace("xiaoming", "ia|min", "") as str

3.2 聚合函数

3.2.1 个数统计

count(*)：统计检索出的行的个数，包括 NULL 值的行。
count(expr)：返回指定字段 expr 的非空值的个数。
count(DISTINCT expr[, expr_.])：返回 expr 是唯一的且非 NULL 的行的数量。
示例 1：select count(*) from TV
示例 2：select count(english) from TV
示例 3：select count(distinct english) from TV
原始数据表 TV 如下图所示：

示例 1，示例 2，示例 3 结果集分别如下三个图所示：

3.2.2 总和统计

sum(col)：对指定列求和（包含重复值）。
sum(DISTINCT col)：对指定列求和（不包含重复值）。
示例 1：select id,sum(math) from TV group by id
示例 2：select id,sum(distinct math) from TV group by id
原始数据表 TV 如下图所示：

当前显示 6 条 / 总共有 6 条数据

id	name	math	english
1	小明	80	90
2	小红	96	89
3	小尹	null	null
4	小花	87	100
4	小花	87	100
4	小花	88	100

表头真名 表头别名

示例 1，示例 2 结果集分别如下二个图所示：

当前显示 4 条 / 总共有 4 条数据

id	sum(math)
1	80
3	null
4	262
2	96

表头真名 表头别名

当前显示 4 条 / 总共有 4 条数据

id	sum(DISTINCT math)
1	80
3	null
4	175
2	96

表头真名 表头别名

3.2.2 平均值统计

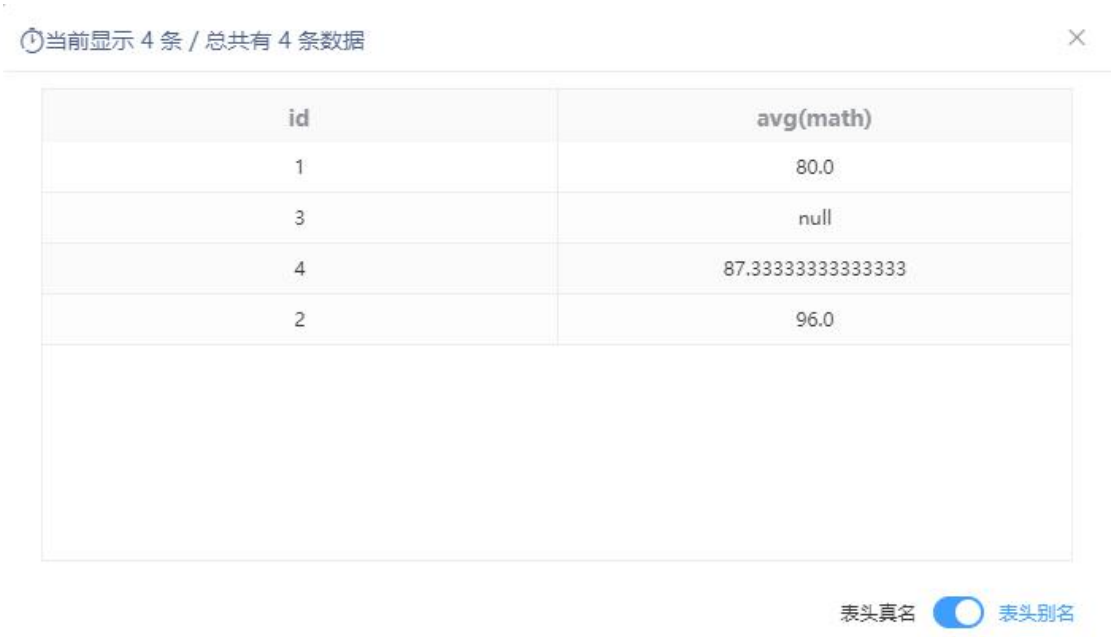
avg(col)：对指定列求平均值者（包含重复值）。
avg(DISTINCT col)：对指定列元素求平均值者（不包含重复值）。
示例 1：select id,avg(math) from TV group by id
示例 2：select id,avg(distinct math) from TV group by id
原始数据表 TV 如下图所示：

当前显示 6 条 / 总共有 6 条数据

id	name	math	english
1	小明	80	90
2	小红	96	89
3	小尹	null	null
4	小花	87	100
4	小花	87	100
4	小花	88	100

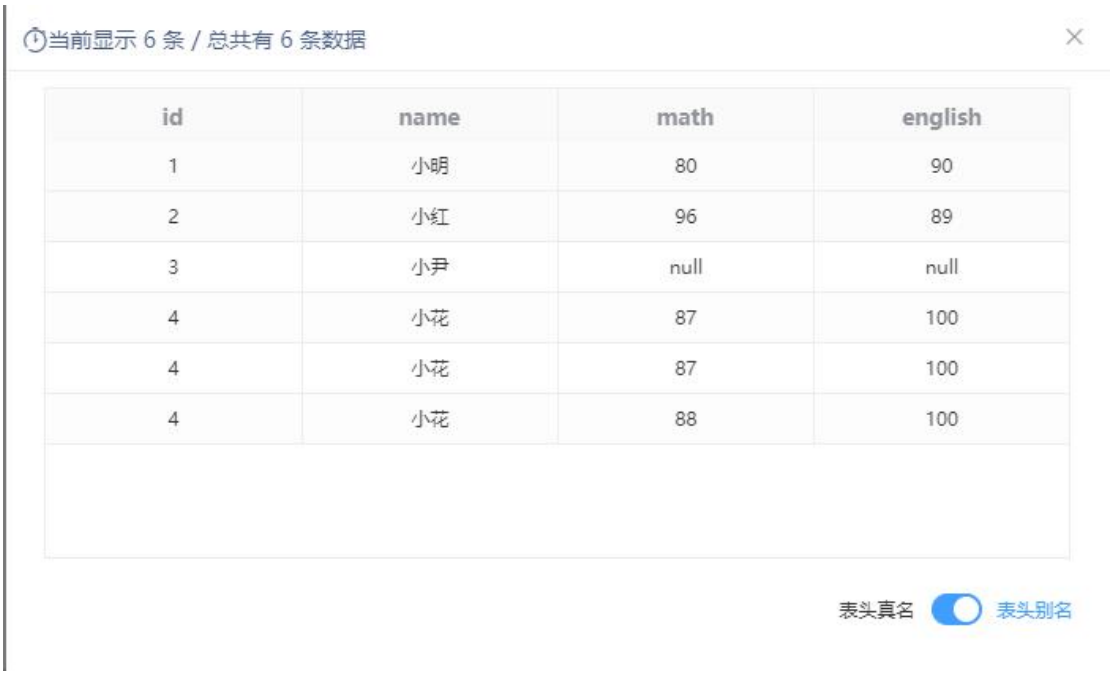
表头真名 表头别名

示例 1，示例 2 结果集分别如下二个图所示：



3.2.3 最小值统计

min(col)：返回指定列的最小值。
示例：select min(math) from TV
原始数据表 TV 如下图所示：



示例结果集如下图所示：

3.2.4 最大值统计

max(col)：返回指定列的最大值。
示例：select max(math) from TV
原始数据表 TV 如下图所示：

示例结果集如下图所示：

3.2.5 方差

variance(col)：返回指定列数值的样本方差，多用于统计学。
var_pop(col)：返回指定列数值的总体方差。
var_samp(col)：返回指定列数值的样本方差。
示例 1：select variance(col1),variance(col2) from TV
示例 2：select var_pop(col1),var_pop(col2) from TV

示例 3：select var_samp(col1),var_samp(col2) from TV
原始数据表 TV 如下图所示：

示例 1，示例 2，示例 3 结果集分别如下三个图所示：

3.2.6 标准差

stddev_pop(col)：求指定列数值的标准偏差。
stddev_samp(col)：求指定列数值的样本标准偏差。
示例 1：select stddev_pop(col1),stddev_pop(col2) from TV
示例 2：select stddev_samp(col1),stddev_samp(col2) from TV
原始数据表 TV 如下图所示：

示例 1，示例 2 结果集分别如下二个图所示：

3.2.7 协方差

covar_pop(col1, col2)：求指定列数值的总体协方差。
covar_samp(col1, col2)：求指定列数值的样本协方差。
示例 1：select covar_pop(col1, col2) from TV
示例 2：select covar_samp(col1, col2) from TV
原始数据表 TV 如下图所示：

示例 1，示例 2 结果集分别如下二个图所示：

3.2.8 相关系数

corr(col1, col2)：返回两列数值的相关系数。
示例：select corr(col1, col2) from TV
原始数据表 TV 如下图所示：

结果集如下图所示：

3.2.9 取 p%分位数

percentile(col, p)：返回 col 的 p%分位数,col 列为 int 型，p 取值 0~1。
percentile(col, 0.5)：取中位数。
示例：select percentile(col1, 0.5),percentile(col2, 0.5) from TV
原始数据表 TV 如下图所示：

结果集如下图所示：

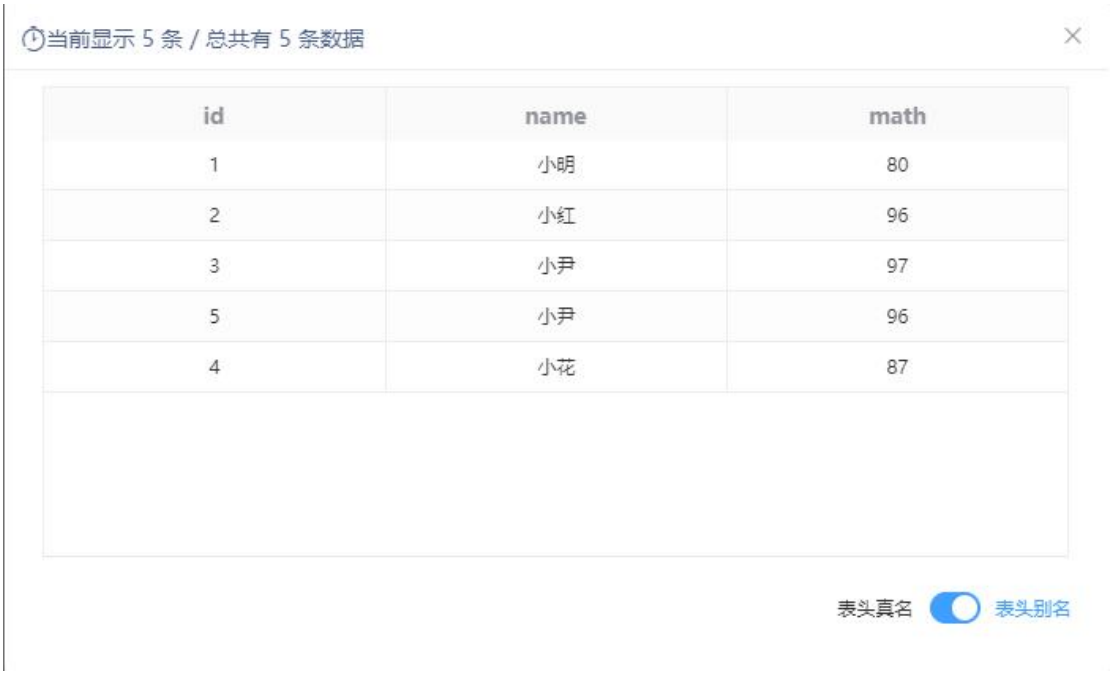
3.3 条件函数

3.3.1 IF 条件

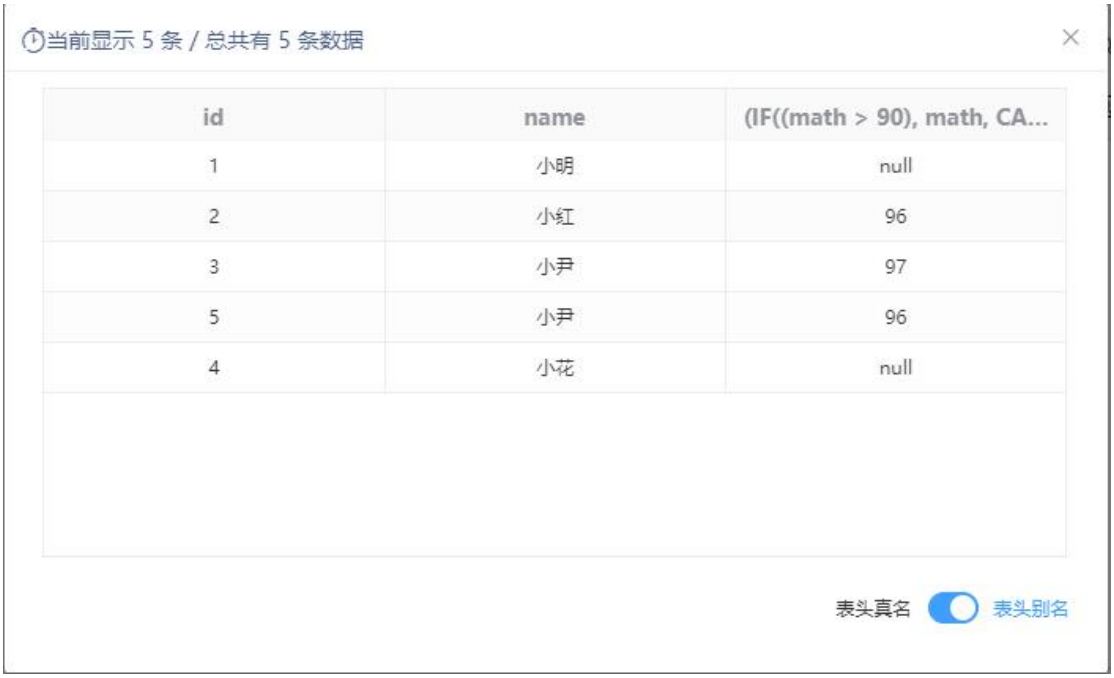
if(boolean testCondition, T valueTrue, T valueFalseOrNull)：如果 testCondition 为 true 就返回 valueTrue,否则返回 valueFalseOrNull 。

示例：select id,name,if(math>90,math,null) from TV

原始数据表 TV 如下图所示：

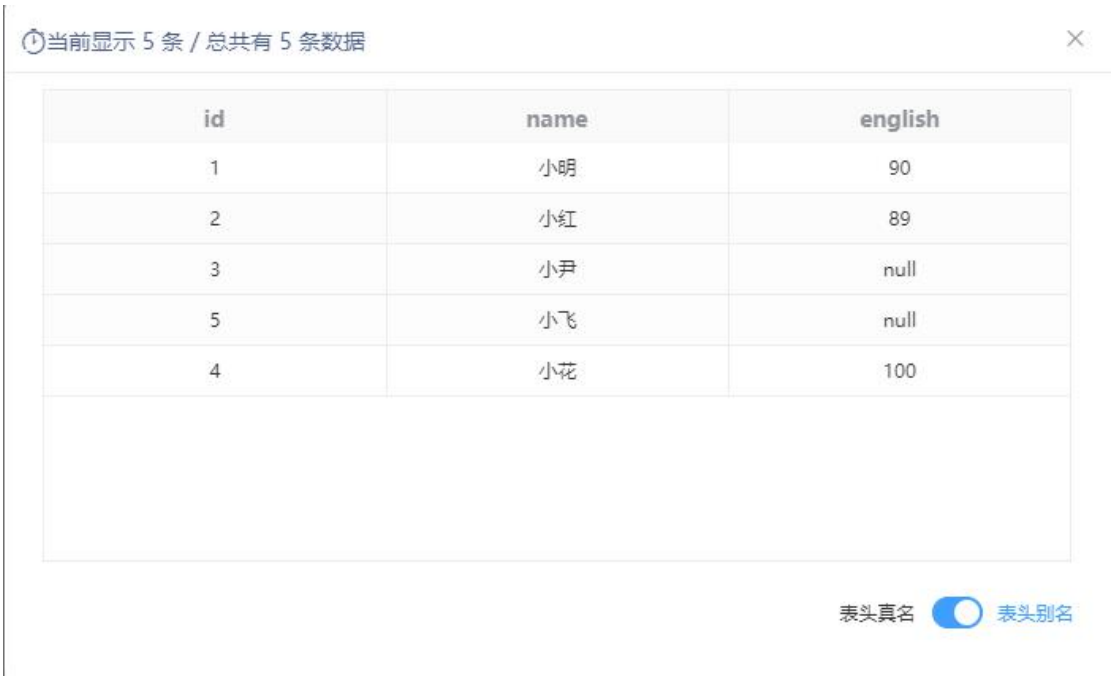


结果集如下图所示：

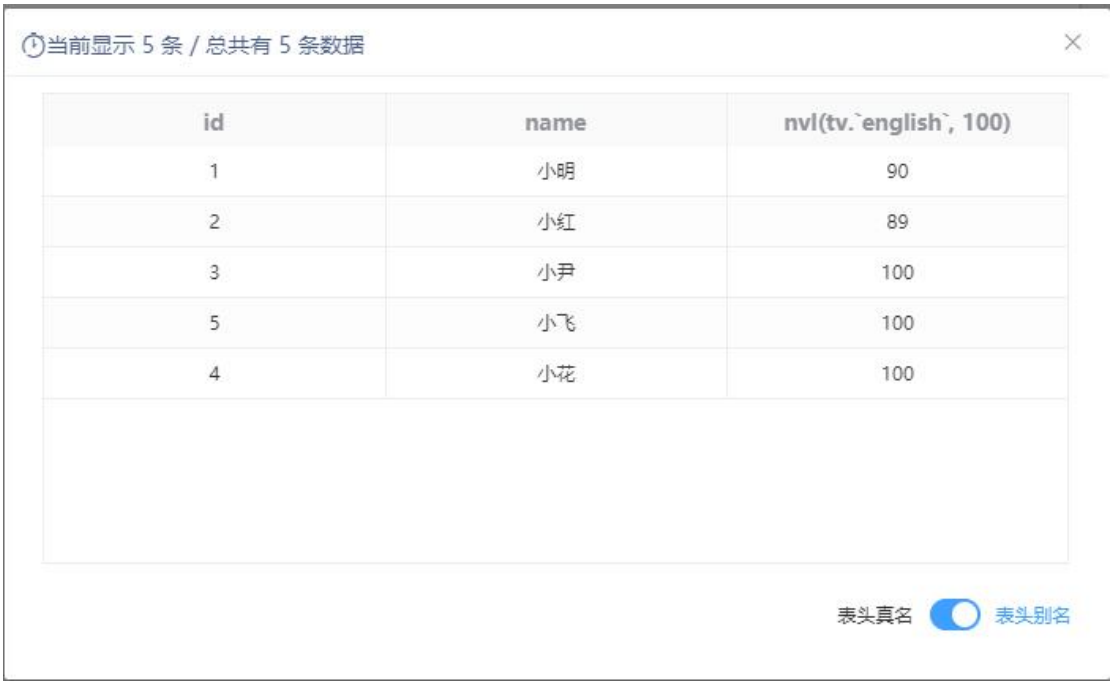


3.3.2 NVL 条件

nvl(T value, T default_value)：如果 value 值为 NULL 就返回 default_value,否则返回 value。
示例：select id,name,nvl(english, 100) from TV
原始数据表 TV 如下图所示：



结果集如下图所示：



3.3.3 COALESCE 条件

COALESCE(T v1, T v2, ...)：返回第一非 null 的值，如果全部都为 NULL 就返回 NUL。
示例：select COALESCE(COL1,COL2) from TV
原始数据表 TV 如下图所示：

结果集如下图所示：

3.3.4 CASE 条件

CASE a WHEN b THEN c [WHEN d THEN e]* [ELSE f] END：如果 a=b 就返回 c,a=d 就返回 e，否则返回 f。

CASE WHEN a THEN b [WHEN c THEN d]* [ELSE e] END：如果 a=ture 就返回 b,c= ture 就返回 d，否则返回 e 。

示例 1：select Col1,(case col1 when 3 then 0 when 4 then '四' else -1 end),COL2 from TV

示例 2：select Col1,COL2 ,(case when col2>5 then '非工作日' else '工作日' end) from TV
原始数据表 TV 如下图所示：

示例 1，示例 2 结果集分别如下二个图所示：

3.3.5 空值处理

isnull(a)：如果 a 为 null 就返回 true，否则返回 false。
isnotnull (a)：如果 a 为非 null 就返回 true，否则返回 false。
示例 1：select * from TV where age is null
示例 2：select * from TV where age is not null
原始数据表 TV 如下图所示：

示例 1，示例 2 结果集分别如下二个图所示：

3.4 数学函数

3.4.1 round 函数

round(DOUBLE a)：返回对 a 四舍五入的 BIGINT 值，取整。
round(DOUBLE a, INT d)：返回 DOUBLE 型 d 的保留 n 位小数的 DOUBLW 型的近似值。
示例 1：select round(3.1415926)
示例 2：select round(3.1415926,2)

示例 1，示例 2 结果集分别如下二个图所示：

3.4.2 bround 函数

bround(DOUBLE a)：银行家舍入法（1~4：舍，6~9：进，5->前位数是偶：舍，5->前位数是奇：进）。

bround(DOUBLE a, INT d)：银行家舍入法,保留 d 位小数。

示例 1：select bround(4.5)

实例 2：select bround(3.145,2)

示例 1，示例 2 结果集分别如下二个图所示：

3.4.3 floor 函数

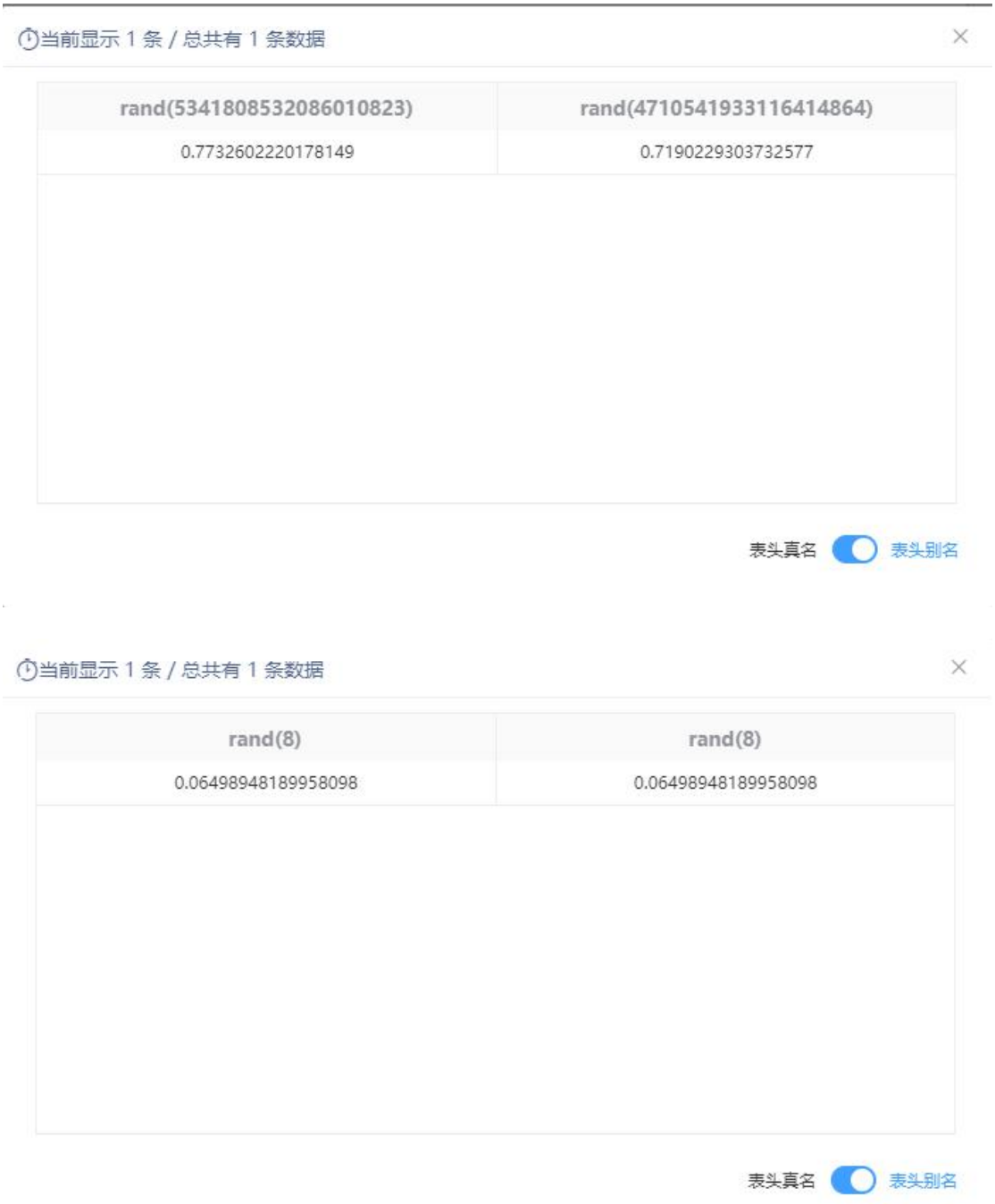
floor(DOUBLE a)：向下取整，最数轴上最接近要求的值的左边的值。
示例：select floor(3.14)
结果集如下图所示：

3.4.4 ceil 函数

ceil(DOUBLE a)：求其不小于给定实数的最小整数。
ceiling(DOUBLE a)：求其不小于给定实数的最小整数。
注：函数返回结果集都是一样。
示例 1：select ceil(-4.1)
示例 2：select ceiling(-4.1)
示例 1，示例 2 结果集如下图所示：

3.4.5 rand 函数

rand()：返回一个 0 到 1 范围内的随机数。
rand(INT seed)：返回一个 0 到 1 范围内的随机数。如果指定种子 seed，则会等到一个稳定的随机数序列。
示例 1：select rand(),rand()
实例 2：select rand(8),rand(8)
示例 1，示例 2 结果集分别如下图所示：



3.4.6 exp 函数

exp(DOUBLE a), exp(DECIMAL a)：返回 e 的 a 幂次方， a 可为小数。
示例：select exp(0),exp(1.11)
结果集如下图所示：

3.4.7 ln 函数

ln(DOUBLE a), ln(DECIMAL a)：以自然常数 e 为底 d 的对数，a 可为小数。
示例：select ln(2),ln(2.1)
结果集如下图所示：

3.4.8 log10 函数

log10(DOUBLE a), log10(DECIMAL a)：以 10 为底 d 的对数，a 可为小数。
示例：select log10(10),log10(10.11)
结果集如下图所示：

3.4.9 log2 函数

log2(DOUBLE a), log2(DECIMAL a)：以 2 为底数 d 的对数，a 可为小数。
示例：select log2(2),log2(2.1)
结果集如下图所示：

3.4.10 log 函数

log(DOUBLE base, DOUBLE a)，log(DECIMAL base, DECIMAL a)：以 base 为底的 a 对数。
示例：select col1,COL2,log(col1,COL2) from TV
原始数据表 TV 如下图所示：

当前显示 4 条 / 总共有 4 条数据

col1	col2
2	4
2	6
3	7
4	7

表头真名 表头别名

结果集如下图所示：

当前显示 4 条 / 总共有 4 条数据

col1	COL2	LOG(CAST(col1 AS DOUB...
2	4	2.0
2	6	2.584962500721156
3	7	1.7712437491614221
4	7	1.403677461028802

表头真名 表头别名

3.4.11 pow 函数

pow(DOUBLE a, DOUBLE p), power(DOUBLE a, DOUBLE p)：计算 a 的 p 次幂。
示例：select col1,COL2,pow(col1,COL2) from TV
原始数据表 TV 如下图所示：

结果集如下图所示：

3.4.12 sqrt 函数

sqrt(DOUBLE a), sqrt(DECIMAL a)：计算 a 的平方根。
示例：select col1,COL2,sqrt(col1),sqrt(col2) from TV
原始数据表 TV 如下图所示：

结果集如下图所示：

3.4.13 abs 函数

abs(DOUBLE a),abs(DECIMAL a)：计算 a 的绝对值。
示例：select abs(-3.1),abs(0)
结果集如下图所示：

3.4.14 pmod 函数

pmod(INT a, INT b), pmod(DOUBLE a, DOUBLE b)： a 除以 b 取余。
示例：select pmod(2,3),pmod(-4,2)
结果集如下图所示：

3.4.15 三角函数

返回值	函数	说明
DOUBLE	sin(DOUBLE a), sin(DECIMAL a)	求 a 的正弦值
DOUBLE	asin(DOUBLE a), asin(DECIMAL a)	求 a 的反正弦值
DOUBLE	cos(DOUBLE a), cos(DECIMAL a)	求 a 余弦值
DOUBLE	acos(DOUBLE a), acos(DECIMAL a)	求 a 的反余弦值
DOUBLE	tan(DOUBLE a), tan(DECIMAL a)	求 a 的正切值
DOUBLE	atan(DOUBLE a), atan(DECIMAL a)	求 a 的反正切值

3.4.16 数学常量

e()：数学常量 e。
pi()：数学常量Π。
示例：select e(),pi()
结果集如下图所示：

3.4.17 negative 函数

negative(INT a), negative(DOUBLE a)：返回 a 的相反数。
示例：select negative(1)
结果集如下图所示：

3.4.18 factorial 函数

factorial(INT a)：求 a 的阶乘。
示例：select factorial(5)
结果集如下图所示：

3.4.19 cbrt 函数

cbrt(DOUBLE a)：求 a 的立方根。
示例 ： select cbrt(512)
结果集如下图所示：

3.4.20 求最值

greatest(T v1, T v2, ...)：求最大值。
least(T v1, T v2, ...)：求最小值。
示例 1：select greatest(col1,col2) from TV
示例 2：select least(col1,col2) from TV
原始数据表 TV 如下图所示：

示例 1，示例 2 结果集分别如下图所示：

3.5 日期函数

3.5.1 Unix 时间戳

unix_timestamp()：获取本地时区下的时间戳。
from_unixtime(bigint unixtime[, string format])：将时间的秒值转换成 format 格式，如 from_unixtime(1250111000,"yyyy-MM-dd") 得到 2009-03-12。
unix_timestamp(string date)：将格式为 yyyy-MM-dd HH:mm:ss 的时间字符串转换成时间戳，如 unix_timestamp('2009-03-20 11:30:01') = 1237573801。
unix_timestamp(string date, string pattern)：将指定时间字符串格式字符串转换成 Unix 时间戳，如果格式不对返回 0，如：unix_timestamp('2009-03-20', 'yyyy-MM-dd') = 1237532400
示例 1：select unix_timestamp()
示例 2：select from_unixtime(1557737000,"yyyy-MM-dd hh:mm:ss")
示例 3：select unix_timestamp('2019-05-13 04:43:20')
示例 4：select unix_timestamp('2019/05/13 04:43:20',"yyyy/MM/dd hh:mm:ss")
示例 1，示例 2，示例 3，示例 4 结果集分别如下图所示：


当前显示 1 条 / 总共有 1 条数据
unix_timestamp(current_timestamp(), yyyy-MM-dd HH:mm:ss)
1557737000
表头真名
表头别名


当前显示 1 条 / 总共有 1 条数据
from_unixtime(CAST(1557737000 AS BIGINT), yyyy-MM-dd hh:mm:ss)
2019-05-1304:43:20
表头真名
表头别名

3.5.2 时间字符串

返回值	函数	说明	示例
string	to_date(string timestamp)	返回日期时间字符串的日期部分。	to_date("1970-01-01 00:00:00") = "1970-01-01"
int	year(string date)	返回日期时间字符串的年份部分。	year("1970-01-01") = 1970

int	quarter(date/timestamp/string)	返回当前日期时间属性哪个季度。	quarter('2015-04-08') = 2
int	month(string date)	返回日期时间字符串的月份部分。	month("1970-11-01") = 11
int	day(string date) dayofmonth(date)	返回日期时间字符串的天。	day("1970-11-01") = 1
int	hour(string date)	返回时间字符串的小时。	hour('2009-07-30 12:58:59') = 12
int	minute(string date)	返回时间字符串的分钟。	hour('2009-07-30 12:58:59') = 58
int	second(string date)	返回时间字符串的秒。	hour('2009-07-30 12:58:59') = 59
int	weekofyear(string date)	返回时间字符串位于一年中的第几个周内。	weekofyear("1970-11-01") = 44
int	datediff(string enddate ,string startdate)	计算开始时间 startdate 到结束时间 enddate 相差的天数。	datediff('2009-03-01', '2009-02-27') = 2
string	date_add(string startdate, int days)	从开始时间 startdate 加上 days。	date_add('2008-12-31', 1) = '2009-01-01'
string	date_sub(string startdate, int days)	从开始时间 startdate 减去 days。	date_sub('2008-12-31', 1) = '2008-12-30'

3.5.3 获取当前日期时间

current_date：返回当前时间日期。
current_timestamp：返回当前时间戳。
示例：select current_date,current_timestamp
结果集如下图所示：

3.5.4 增加月数

add_months(string start_date, int num_months)：返回当前时间下再增加 num_months 个月的日期。

示例：select add_months('2019-05-13',120)

结果集如下图所示：

3.5.5 获取月末日期

last_day(string date)：返回这个月的最后一天的日期，忽略时分秒部分（HH:mm:ss）。
示例：select last_day('2019-02-01')
结果集如下图所示：

3.5.6 获取开始年或月

trunc(string date, string format)：返回时间的最开始年份或月份。
示例：select trunc("2016-06-26","MM"),trunc("2016-06-26","YY")
结果集如下图所示：

3.5.7 日期月份差

months_between(date1, date2)：返回 date1 与 date2 之间相差的月份，如 date1>date2，则返回正，如果 date1<date2,则返回负，否则返回 0.0。
示例：select months_between("2019-04-01", "2019-06-01")
结果集如下图所示：

3.5.8 获取指定格式日期

date_format(date/timestamp/string ts, string fmt)：按指定格式返回时间 date。
示例：select date_format('2019-05-13','MM/dd')
结果集如下图所示：

3.5.9 下周几日期

next_day(string start_date, string day_of_week)：返回当前时间的星期几所对应的日期。
示例：select next_day('2019-05-13', 'TU')
注：2019-05-13 日以后星期二第一次出现的日期。
结果集如下图所示：

3.6 类型转换函数

cast(expr as <type>)：将 expr 转换成 type 类型。
示例：select cast("101" as int),cast(1.01 as int),cast("2019-05-13" as timestamp)
注：将字符串”101”转成 intx 型，浮点型 1.01 转成 int 型，字符串”2019-05-13”转成时间戳。
结果集如下图所示：

3.7 窗口函数

ROWS BETWEEN：指定窗口区间，如果不指定 ROWS BETWEEN,默认为从起点到当前行;
ORDER BY：排序，如果不指定 ORDER BY，则将分组内所有值累加
PRECEDING：往前
FOLLOWING：往后
CURRENT ROW：当前行
UNBOUNDED：起点
UNBOUNDED PRECEDING：从前面的起点
UNBOUNDED FOLLOWING：到后面的终点

3.7.1 求和

示例：

```
select
name,
createtime,
t0,
SUM(t0) OVER(PARTITION BY name ORDER BY createtime) AS t1,
SUM(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS t2,
SUM(t0) OVER(PARTITION BY name) AS t3,
SUM(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS t4,
SUM(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND 1 FOLLOWING) AS t5,
SUM(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS t6
from
TV
```

注：

SUM(t0) OVER(PARTITION BY name ORDER BY createtime) AS t1 为分组内（name）从起点到当前行的 t0 列的值累加。

SUM(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS t2 为分组内（name）从起点到当前行的 t0 列的值累加。t2 等同于 t1。

SUM(t0) OVER(PARTITION BY name) AS t3 为分组内（name）所有 t0 列的值累加。

SUM(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS t4 为分组内（name）当前行加当前行的前三行的 t0 列的值累加。

SUM(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND 1 FOLLOWING) AS t5 为分组内（name）当前行加当前行的前三行和当前行的往后一行的 t0 列的值累加。

SUM(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN CURRENT ROW AND

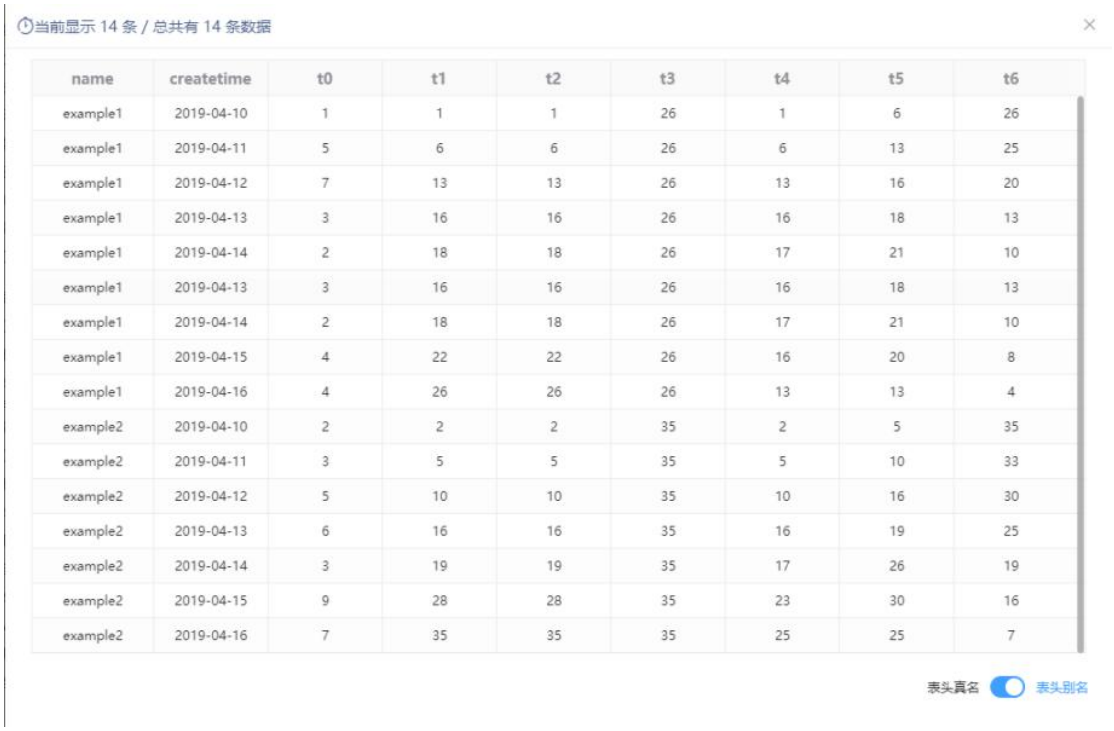
UNBOUNDED FOLLOWING) AS t6 为分组内（name）当前行加往前行往后所有行的 t0 列的值累加。

原始数据表 TV 如下图所示：

当前显示 14 条 / 总共有 14 条数据

name	createtime	t0
example1	2019-04-10	1
example1	2019-04-11	5
example1	2019-04-12	7
example1	2019-04-15	4
example1	2019-04-16	4
example2	2019-04-10	2
example2	2019-04-11	3
example2	2019-04-12	5
example2	2019-04-13	6
example2	2019-04-14	3
example2	2019-04-15	9
example2	2019-04-16	7

结果集如下图所示：



当前显示 14 条 / 总共有 14 条数据

name	createtime	t0	t1	t2	t3	t4	t5	t6
example1	2019-04-10	1	1	1	26	1	6	26
example1	2019-04-11	5	6	6	26	6	13	25
example1	2019-04-12	7	13	13	26	13	16	20
example1	2019-04-13	3	16	16	26	16	18	13
example1	2019-04-14	2	18	18	26	17	21	10
example1	2019-04-13	3	16	16	26	16	18	13
example1	2019-04-14	2	18	18	26	17	21	10
example1	2019-04-15	4	22	22	26	16	20	8
example1	2019-04-16	4	26	26	26	13	13	4
example2	2019-04-10	2	2	2	35	2	5	35
example2	2019-04-11	3	5	5	35	5	10	33
example2	2019-04-12	5	10	10	35	10	16	30
example2	2019-04-13	6	16	16	35	16	19	25
example2	2019-04-14	3	19	19	35	17	26	19
example2	2019-04-15	9	28	28	35	23	30	16
example2	2019-04-16	7	35	35	35	25	25	7

表头真名 表头别名

3.7.2 取平均

注：用法跟求和类似
示例：

```
select
name,
createtime,
t0,
avg(t0) OVER(PARTITION BY name ORDER BY createtime) AS t1,
avg(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS t2,
avg(t0) OVER(PARTITION BY name) AS t3,
avg(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS t4,
avg(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND 1 FOLLOWING) AS t5,
avg(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS t6
from
TV
```

原始数据表 TV 如下图所示：

name	createtime	t0
example1	2019-04-10	1
example1	2019-04-11	5
example1	2019-04-12	7
example1	2019-04-15	4
example1	2019-04-16	4
example2	2019-04-10	2
example2	2019-04-11	3
example2	2019-04-12	5
example2	2019-04-13	6
example2	2019-04-14	3
example2	2019-04-15	9
example2	2019-04-16	7

结果集如下图所示：

name	createtime	t0	t1	t2	t3	t4	t5	t6
example1	2019-04-10	1	1.0	1.0	3.714285714...	1.0	3.0	3.714285714...
example1	2019-04-11	5	3.0	3.0	3.714285714...	3.0	4.333333333...	4.166666666...
example1	2019-04-12	7	4.333333333...	4.333333333...	3.714285714...	4.333333333...	4.0	4.0
example1	2019-04-13	3	4.0	4.0	3.714285714...	4.0	3.6	3.25
example1	2019-04-14	2	3.6	3.6	3.714285714...	4.25	4.2	3.333333333...
example1	2019-04-15	4	3.666666666...	3.666666666...	3.714285714...	4.0	4.0	4.0
example1	2019-04-16	4	3.714285714...	3.714285714...	3.714285714...	3.25	3.25	4.0
example2	2019-04-10	2	2.0	2.0	5.0	2.0	2.5	5.0
example2	2019-04-11	3	2.5	2.5	5.0	2.5	3.333333333...	5.5
example2	2019-04-12	5	3.333333333...	3.333333333...	5.0	3.333333333...	4.0	6.0
example2	2019-04-13	6	4.0	4.0	5.0	4.0	3.8	6.25
example2	2019-04-14	3	3.8	3.8	5.0	4.25	5.2	6.333333333...
example2	2019-04-15	9	4.666666666...	4.666666666...	5.0	5.75	6.0	8.0
example2	2019-04-16	7	5.0	5.0	5.0	6.25	6.25	7.0

表头真名 表头别名

3.7.3 最大值

注：用法跟求和类似。

示例：

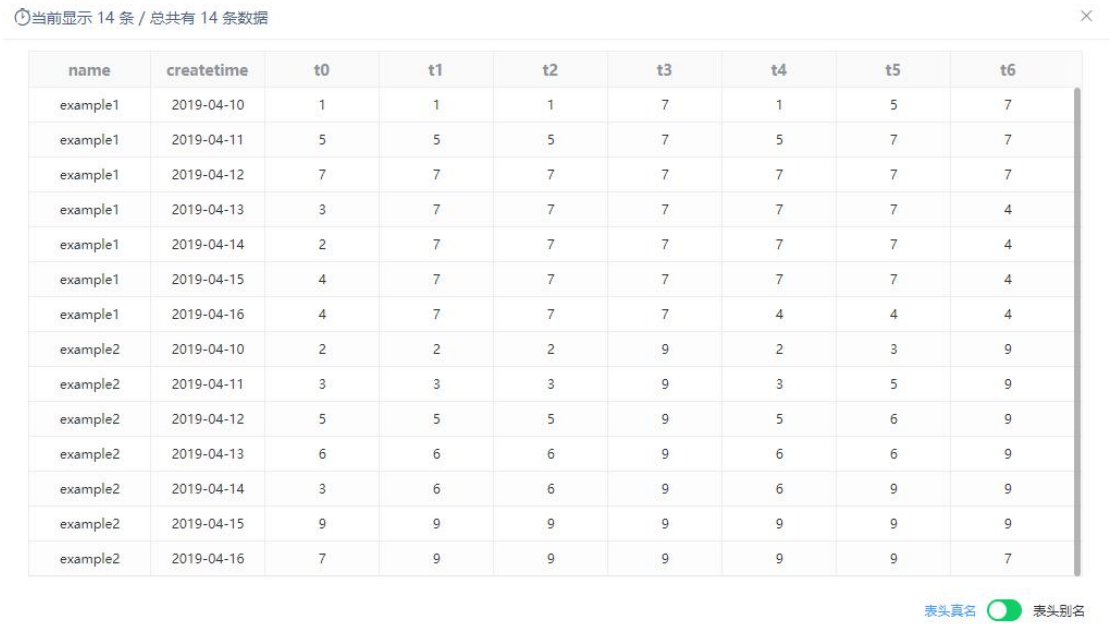
```
select
name,
createtime,
t0,
max(t0) OVER(PARTITION BY name ORDER BY createtime) AS t1,
max(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS t2,
max(t0) OVER(PARTITION BY name) AS t3,
max(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS t4,
max(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND 1 FOLLOWING) AS t5,
max(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS t6
from
TV
```

原始数据表 TV 如下图所示：


当前显示 14 条 / 总共有 14 条数据

name	createtime	t0
example1	2019-04-10	1
example1	2019-04-11	5
example1	2019-04-12	7
example1	2019-04-15	4
example1	2019-04-16	4
example2	2019-04-10	2
example2	2019-04-11	3
example2	2019-04-12	5
example2	2019-04-13	6
example2	2019-04-14	3
example2	2019-04-15	9
example2	2019-04-16	7

结果集如下图所示：


当前显示 14 条 / 总共有 14 条数据

name	createtime	t0	t1	t2	t3	t4	t5	t6
example1	2019-04-10	1	1	1	7	1	5	7
example1	2019-04-11	5	5	5	7	5	7	7
example1	2019-04-12	7	7	7	7	7	7	7
example1	2019-04-13	3	7	7	7	7	7	4
example1	2019-04-14	2	7	7	7	7	7	4
example1	2019-04-15	4	7	7	7	7	7	4
example1	2019-04-16	4	7	7	7	4	4	4
example2	2019-04-10	2	2	2	9	2	3	9
example2	2019-04-11	3	3	3	9	3	5	9
example2	2019-04-12	5	5	5	9	5	6	9
example2	2019-04-13	6	6	6	9	6	6	9
example2	2019-04-14	3	6	6	9	6	9	9
example2	2019-04-15	9	9	9	9	9	9	9
example2	2019-04-16	7	9	9	9	9	9	7

表头真名 表头别名

3.7.4 最小值

注：用法跟求和类似。

示例：

select

```
name,
createtime,
t0,
min(t0) OVER(PARTITION BY name ORDER BY createtime) AS t1,
min(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS t2,
min(t0) OVER(PARTITION BY name) AS t3,
min(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS t4,
min(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN 3 PRECEDING AND 1 FOLLOWING) AS t5,
min(t0) OVER(PARTITION BY name ORDER BY createtime ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS t6
from
TV
```

原始数据表 TV 如下图所示：

name	createtime	t0
example1	2019-04-10	1
example1	2019-04-11	5
example1	2019-04-12	7
example1	2019-04-15	4
example1	2019-04-16	4
example2	2019-04-10	2
example2	2019-04-11	3
example2	2019-04-12	5
example2	2019-04-13	6
example2	2019-04-14	3
example2	2019-04-15	9
example2	2019-04-16	7

结果集如下图所示：

name	createtime	t0	t1	t2	t3	t4	t5	t6
example1	2019-04-10	1	1	1	1	1	1	1
example1	2019-04-11	5	1	1	1	1	1	2
example1	2019-04-12	7	1	1	1	1	1	2
example1	2019-04-13	3	1	1	1	1	1	2
example1	2019-04-14	2	1	1	1	2	2	2
example1	2019-04-15	4	1	1	1	2	2	4
example1	2019-04-16	4	1	1	1	2	2	4
example2	2019-04-10	2	2	2	2	2	2	2
example2	2019-04-11	3	2	2	2	2	2	3
example2	2019-04-12	5	2	2	2	2	2	3
example2	2019-04-13	6	2	2	2	2	2	3
example2	2019-04-14	3	2	2	2	3	3	3
example2	2019-04-15	9	2	2	2	3	3	7
example2	2019-04-16	7	2	2	2	3	3	7

3.7.5 切片

NTILE(n)：用于将分组数据按照顺序切分成 n 片，返回当前切片值。NTILE 不支持 ROWS BETWEEN，如果切片不均匀，默认增加第一个切片的分布。

示例：

```
select
name,
createtime,
t0,
NTILE(2) OVER(PARTITION BY name ORDER BY createtime) AS t1,
NTILE(3) OVER(PARTITION BY name ORDER BY createtime) AS t2,
NTILE(4) OVER(PARTITION BY name ORDER BY createtime) AS t3
from
TV
```

注：

NTILE(2) OVER(PARTITION BY name ORDER BY createtime) AS t1 为分组内将数据分成 2 片。

NTILE(3) OVER(PARTITION BY name ORDER BY createtime) AS t2 为分组内将数据分成 3 片。

NTILE(4) OVER(PARTITION BY name ORDER BY createtime) AS t3 为分组内将数据分成 4 片。

原始数据表 TV 如下图所示：

当前显示 14 条 / 总共有 14 条数据

name	createtime	t0
example1	2019-04-10	1
example1	2019-04-11	5
example1	2019-04-12	7
example1	2019-04-15	4
example1	2019-04-16	4
example2	2019-04-10	2
example2	2019-04-11	3
example2	2019-04-12	5
example2	2019-04-13	6
example2	2019-04-14	3
example2	2019-04-15	9
example2	2019-04-16	7

结果集如下图所示：

name	createtime	t0	t1	t2	t3
example1	2019-04-10	1	1	1	1
example1	2019-04-11	5	1	1	1
example1	2019-04-12	7	1	1	2
example1	2019-04-13	3	1	2	2
example1	2019-04-14	2	2	2	3
example1	2019-04-15	4	2	3	3
example1	2019-04-16	4	2	3	4
example2	2019-04-10	2	1	1	1
example2	2019-04-11	3	1	1	1
example2	2019-04-12	5	1	1	2
example2	2019-04-13	6	1	2	2
example2	2019-04-14	3	2	2	3
example2	2019-04-15	9	2	3	3
example2	2019-04-16	7	2	3	4

3.7.6 添加序列号

ROW_NUMBER()：从 1 开始，按照顺序，生成分组内记录的序列。

示例：

select

```
name,
createtime,
t0,
ROW_NUMBER() OVER(PARTITION BY name ORDER BY t0 asc) AS t1
from
TV
```

注：

ROW_NUMBER() OVER(PARTITION BY name ORDER BY t0 asc) AS t1 为分组内根据 t0 升序排列添加序号 t1 列的值。

原始数据表 TV 如下图所示：

name	createtime	t0
example1	2019-04-10	1
example1	2019-04-11	5
example1	2019-04-12	7
example1	2019-04-15	4
example1	2019-04-16	4
example2	2019-04-10	2
example2	2019-04-11	3
example2	2019-04-12	5
example2	2019-04-13	6
example2	2019-04-14	3
example2	2019-04-15	9
example2	2019-04-16	7

结果集如下图所示：

name	createtime	t0	t1
example1	2019-04-10	1	1
example1	2019-04-14	2	2
example1	2019-04-13	3	3
example1	2019-04-16	4	4
example1	2019-04-15	4	5
example1	2019-04-11	5	6
example1	2019-04-12	7	7
example2	2019-04-10	2	1
example2	2019-04-14	3	2
example2	2019-04-11	3	3
example2	2019-04-12	5	4
example2	2019-04-13	6	5
example2	2019-04-16	7	6
example2	2019-04-15	9	7

3.7.7 排名

RANK()：在分组中的排名，排名相等会在名次中留下空缺位。
DENSE_RANK()：在分组中的排名，排名相等会在名次中不会留下空缺位。
示例：

```
select
name,
createtime,
t0,
ROW_NUMBER() OVER(PARTITION BY name ORDER BY t0 desc) AS ROW_NUMBER,
RANK() OVER(PARTITION BY name ORDER BY t0 desc) AS RANK,
DENSE_RANK() OVER(PARTITION BY name ORDER BY t0 desc) AS DENSE_RANK
from
TV
```

注：
ROW_NUMBER() OVER(PARTITION BY name ORDER BY t0 desc) AS ROW_NUMBER 为分组内 t0 降序添加序号 ROW_NUMBER 列的值。
RANK() OVER(PARTITION BY name ORDER BY t0 desc) AS RANK 为分组内 t0 值排名。

DENSE_RANK() OVER(PARTITION BY name ORDER BY t0 desc) AS DENSE_RANK 为分组内 t0 值的排名。

原始数据表 TV 如下图所示：

当前显示 14 条 / 总共有 14 条数据

name	createtime	t0
example1	2019-04-10	1
example1	2019-04-11	5
example1	2019-04-12	7
example1	2019-04-15	4
example1	2019-04-16	4
example2	2019-04-10	2
example2	2019-04-11	3
example2	2019-04-12	5
example2	2019-04-13	6
example2	2019-04-14	3
example2	2019-04-15	9
example2	2019-04-16	7

结果集如下图所示：

当前显示 14 条 / 总共有 14 条数据

name	createtime	t0	ROW_NUMBER	RANK	DENSE_RANK
example1	2019-04-12	7	1	1	1
example1	2019-04-11	5	2	2	2
example1	2019-04-15	4	3	3	3
example1	2019-04-16	4	4	3	3
example1	2019-04-13	3	5	5	4
example1	2019-04-14	2	6	6	5
example1	2019-04-10	1	7	7	6
example2	2019-04-15	9	1	1	1
example2	2019-04-16	7	2	2	2
example2	2019-04-13	6	3	3	3
example2	2019-04-12	5	4	4	4
example2	2019-04-14	3	5	5	5
example2	2019-04-11	3	6	5	5
example2	2019-04-10	2	7	7	6

表头真名 表头别名

3.7.8 错位

LAG(col,n,DEFAULT)：用于统计窗口内往上第 n 行值。**col** 为列名，**n** 为往上第 n 行（可选，默认为 1），**DEFAULT** 为默认值（当往上第 n 行为 NULL 时候，取默认值，如不指定，则为 NULL）。

LEAD(col,n,DEFAULT)：用于统计窗口内往下第 n 行值。

示例 1：

```
select
name,
createtime,
t0,
ROW_NUMBER() OVER(PARTITION BY name ORDER BY createtime) AS ROW_NUMBER,
LAG(createtime,1,'2019-08-08') OVER(PARTITION BY name ORDER BY createtime) AS last_1_time,
LAG(createtime,2) OVER(PARTITION BY name ORDER BY createtime) AS last_2_time
from TV
```

注：

LAG(createtime,1,'2019-08-08') OVER(PARTITION BY name ORDER BY createtime) AS last_1_time 为分组内 createtime 往上一行,默认值 2019-08-08。

LAG(createtime,2) OVER(PARTITION BY name ORDER BY createtime) AS last_2_time 为分组内 createtime 往上一行，无默认值。

示例 2：

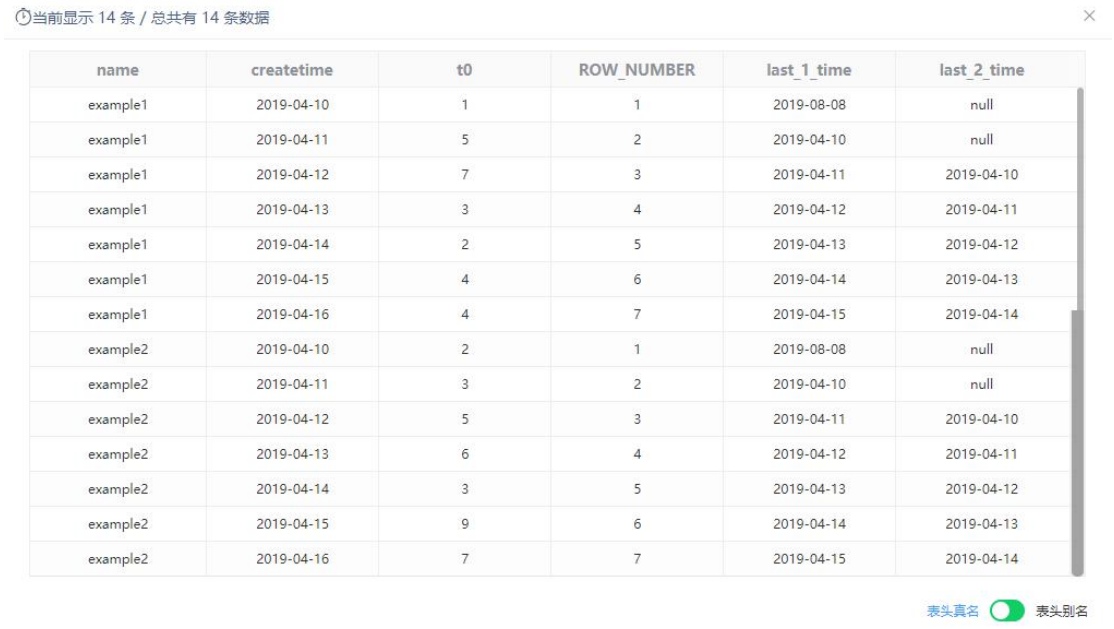
```
select
name,
createtime,
t0,
ROW_NUMBER() OVER(PARTITION BY name ORDER BY createtime) AS ROW_NUMBER,
LEAD(createtime,1,'2019-08-08') OVER(PARTITION BY name ORDER BY createtime) AS last_1_time,
LEAD(createtime,2) OVER(PARTITION BY name ORDER BY createtime) AS last_2_time
from TV
```

原始数据表 TV 如下图所示：

当前显示 14 条 / 总共有 14 条数据

name	createtime	t0
example1	2019-04-10	1
example1	2019-04-11	5
example1	2019-04-12	7
example1	2019-04-15	4
example1	2019-04-16	4
example2	2019-04-10	2
example2	2019-04-11	3
example2	2019-04-12	5
example2	2019-04-13	6
example2	2019-04-14	3
example2	2019-04-15	9
example2	2019-04-16	7

示例 1，示例 2 结果集分别如下图所示：



name	createtime	t0	ROW_NUMBER	last_1_time	last_2_time
example1	2019-04-10	1	1	2019-04-11	2019-04-12
example1	2019-04-11	5	2	2019-04-12	2019-04-13
example1	2019-04-12	7	3	2019-04-13	2019-04-14
example1	2019-04-13	3	4	2019-04-14	2019-04-15
example1	2019-04-14	2	5	2019-04-15	2019-04-16
example1	2019-04-15	4	6	2019-04-16	null
example1	2019-04-16	4	7	2019-08-08	null
example2	2019-04-10	2	1	2019-04-11	2019-04-12
example2	2019-04-11	3	2	2019-04-12	2019-04-13
example2	2019-04-12	5	3	2019-04-13	2019-04-14
example2	2019-04-13	6	4	2019-04-14	2019-04-15
example2	2019-04-14	3	5	2019-04-15	2019-04-16
example2	2019-04-15	9	6	2019-04-16	null
example2	2019-04-16	7	7	2019-08-08	null

3.7.9 取首尾值

FIRST_VALUE(col)：取分组内排序后，截止到当前行，col 列第一个值。
LAST_VALUE(col)：取分组内排序后，截止到当前行，col 列最后一个值
示例 1：

```
select
name,
createtime,
t0,
ROW_NUMBER() OVER(PARTITION BY name ORDER BY createtime) AS ROW_NUMBER,
FIRST_VALUE(t0) OVER(PARTITION BY name ORDER BY createtime) AS FIRST_VALUE
from TV
```

示例 2：

```
select
name,
createtime,
t0,
ROW_NUMBER() OVER(PARTITION BY name ORDER BY createtime) AS ROW_NUMBER,
LAST_VALUE(t0) OVER(PARTITION BY name ORDER BY createtime) AS LAST_VALUE
from TV
```

原始数据表 TV 如下图所示：

当前显示 14 条 / 总共有 14 条数据

name	createtime	t0
example1	2019-04-10	1
example1	2019-04-11	5
example1	2019-04-12	7
example1	2019-04-15	4
example1	2019-04-16	4
example2	2019-04-10	2
example2	2019-04-11	3
example2	2019-04-12	5
example2	2019-04-13	6
example2	2019-04-14	3
example2	2019-04-15	9
example2	2019-04-16	7

示例 1，示例 2 结果集分别如下图所示：


当前显示 14 条 / 总共有 14 条数据

name	createtime	t0	ROW_NUMBER	LAST_VALUE
example1	2019-04-10	1	1	1
example1	2019-04-11	5	2	5
example1	2019-04-12	7	3	7
example1	2019-04-13	3	4	3
example1	2019-04-14	2	5	2
example1	2019-04-15	4	6	4
example1	2019-04-16	4	7	4
example2	2019-04-10	2	1	2
example2	2019-04-11	3	2	3
example2	2019-04-12	5	3	5
example2	2019-04-13	6	4	6
example2	2019-04-14	3	5	3
example2	2019-04-15	9	6	9
example2	2019-04-16	7	7	7

表头真名 表头别名

3.7.10 GROUPING SETS

GROUPING SETS：允许我们在同一数据集中指定多个 GROUP BY。

示例 1：

```
select subject,name,avg(score) from TV group by subject,name grouping sets((subject,name))
```

等价于

```
select subject,name,avg(score) from TV group by subject,name
```

示例 2：

```
select subject,name,avg(score) from TV group by subject,name grouping sets((subject,name),subject)
```

等价于

```
select subject,name,avg(score) from TV group by subject,name
union
select subject,null,avg(score) from TV group by subject
```

示例 3：

```
select subject,name,avg(score) from TV group by subject,name grouping sets(subject,name)
```

等价于

```
select subject,null,avg(score) from TV group by subject
union
select null,name,avg(score) from TV group by name
```

示例 4：

```
select subject,name,avg(score) from TV group by subject,name grouping sets((subject,name),subject,name,())
```

等价于

```
select subject,null,avg(score) from TV group by subject,name
union
```

```
select subject,null,avg(score) from TV group by subject
union
select null,name,avg(score) from TV group by name
union
select null,null,avg(score) from TV
```

原始数据表 TV 如下图所示：

当前显示 8 条 / 总共有 8 条数据

subject	name	score
英语	张三	88
数学	张三	78
英语	李四	86
数学	李四	98
英语	李四	66
数学	李四	78
英语	王五	88
数学	王五	82

表头真名 表头别名

示例 1 结果集如下图所示：

当前显示 6 条 / 总共有 6 条数据

subject	name	avg(score)
数学	王五	82.0
英语	李四	76.0
数学	李四	88.0
英语	王五	88.0
数学	张三	78.0
英语	张三	88.0

表头真名 表头别名

示例 2 结果集如下图所示:

当前显示 8 条 / 总共有 8 条数据

subject	name	avg(score)
数学	王五	82.0
数学	null	84.0
英语	李四	76.0
数学	李四	88.0
英语	王五	88.0
数学	张三	78.0
英语	null	82.0
英语	张三	88.0

表头真名 表头别名

示例 3 结果集如下图所示:

当前显示 5 条 / 总共有 5 条数据

subject	name	avg(score)
null	李四	82.0
数学	null	84.0
null	王五	85.0
null	张三	83.0
英语	null	82.0

表头真名 表头别名

示例 4 结果集如下图所示:

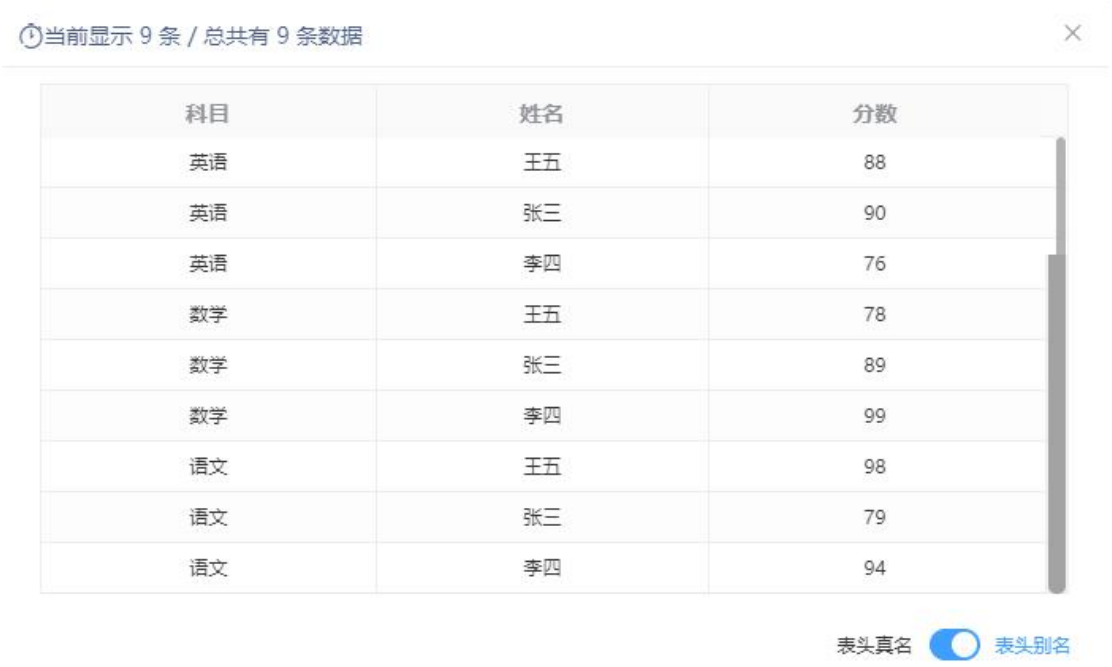
3.8 其它

3.8.1 行转列

使用 PIVOT 函数即可实现行转列。
示例：

```
select * from TV
    pivot
    (
      sum(`分数`) for
      `姓名` in ('王五','张三','李四')
    )
```

原始数据表 TV 如下图所示：



结果集如下图所示：

3.8.2 列转行

Stack()函数可以实现列转行。stack(n, expr1, ..., exprk) -会将 expr1, ..., exprk 分割为 n 行。
示例：

```
select `科目`,
stack(3, '王五', `王五`, '张三', `张三`,'李四', `李四`) as (`姓名`, `分数`)
```

from TV

原始数据表 TV 如下图所示：

结果集如下图所示：

附录 A 主要函数列表

A.1 字符串函数

返回值	函数	描述
int	**char_length(s)**	返回字符串 s 的字符数
int	**character_length(s)**	返回字符串 s 的字符数
int	**length(s)**	返回字符串 s 的字符数
string	**concat(s1,s2...sn)**	字符串 s1,s2... 等多个字符串合并为一个字符串
string	**concat_ws(split, s1,s2...sn)**：	同 CONCAT(s1,s2,...) 函数，但是每个字符串直接要加上 split，split 一般是分隔符。
string	**reverse(s)**	返回字符串 S 的反转结果
string	**substr(s, start)**	返回字符串 S 从 START 位置到结尾的字符串
string	**substring(s, start)**	返回字符串 S 从 START 位置到结尾的字符串
string	**upper(s)**	将字符串 S 转成大写
string	**ucase(s)**	将字符串 S 转成小写
string	**repeat(s,n)**	返回重复字符串 S 的次数 N
string	**lpad(s1, len, s2)**	左补足，将 S1 进行用 S2 进行左补足到 LEN 位
string	**rpad(s1, len, s2)**	右补足，将 S1 进行用 S2 进行右补足到 LEN 位
string	**split(s, sp)**	按照 SP 字符串分割 S，会返回分割后的字符串数组
int	**find_in_set(s, sl)**	返回字符串 S 在字符串 SL 第一次出现的位置，SL 是用逗号分割的字符串，如果没有找该 S 字符串，则返回 0
int	**position(s1 in s)**	返回字符串 S1 在字符串 S 中开始出现的位置，没有则返回 0
string	**regexp_replace(s1, s2, s3)**	将字符串 S1 中的符合 java 正则表达式 S2 的部分替换为 S3

A.2 聚合函数

返回值	函数	描述
int	**count(*)**	统计检索出的行的个数，包括 NULL 值的行
int	**count(expr)**	返回指定字段 expr 的非空值的个数
int	**count(DISTINCT expr[, expr_.])**	返回 expr 是唯一的且非 NULL 的行的数量
int,double	**sum(col)**	对指定列求和（包含重复值）
Int,double	**sum(DISTINCT col)**	对指定列求和（不包含重复值）
double	**avg(col)**	对指定列求平均值者（包含重复值）

double	**avg(DISTINCT col)**	对指定列求平均值者（不包含重复值）
int,double	**min(col)**	返回指定列的最小值
Int,double	**max(col)**	返回指定列的最大值
double	**variance(col)**	返回指定列数值的样本方差，多用于统计学
double	**var_pop(col)**	返回指定列数值的总体方差
double	**var_samp(col)**	返回指定列数值的样本方差
double	**stddev_pop(col)**	求指定列数值的标准偏差
double	**stddev_samp(col)**	求指定列数值的样本标准偏差
double	**covar_pop(col1, col2)**	求指定列数值的总体协方差
double	**covar_samp(col1, col2)**	求指定列数值的样本协方差
double	**corr(col1, col2)**	返回两列数值的相关系数
double	**percentile(col, p)**	返回 col 的 p%分位数,col 列为 int 型，p 取值 0~1

A.3 条件函数

返回值	函数	描述
T	**if(boolean testCondition, T valueTrue, T valueFalseOrNull)**	如果 testCondition 为 true 就返回 valueTrue,否则返回 valueFalseOrNull 。
T	**nvl(T value, T default_value)**	如果 value 值为 NULL 就返回 default_value,否则返回 value
T	**COALESCE(T v1, T v2, ...)**	返回第一非 null 的值，如果全部都为 NULL 就返回 NUL
T	**CASE a WHEN b THEN c [WHEN d THEN e]* [ELSE f] END**	如果 a=b 就返回 c,a=d 就返回 e，否则返回 f
T	**CASE WHEN a THEN b [WHEN c THEN d]* [ELSE e] END**	如果 a=ture 就返回 b,c= ture 就返回 d，否则返回 e
boolean	**isnull(a)**	如果 a 为 null 就返回 true，否则返回 false
boolean	**isnotnull (a)**	如果 a 为非 null 就返回 true，否则返回 false

A.4 数学函数

返回值	函数	描述
int	**round(DOUBLE a)**	返回对 a 四舍五入的 BIGINT 值，取整
double	**round(DOUBLE a, INT d)**	返回 DOUBLE 型 d 的保留 n 位小数的 DOUBLW 型的近似值
int	**floor(DOUBLE a)**	向下取整，最数轴上最接近要求的值的左边的值
int	**ceil(DOUBLE a)**	求其不小于给定实数的最小整数
int	**ceiling(DOUBLE a)**	求其不小于给定实数的最小整数
double	**rand()**	返回一个 0 到 1 范围内的随机数
double	**rand(INT seed)**	返回一个 0 到 1 范围内的随机数。如果指定

		种子 seed，则会等到一个稳定的随机数序列
double	**exp(DOUBLE a)**	返回 e 的 a 幂次方， a 可为小数
double	**ln(DOUBLE a)**	以自然常数 e 为底 d 的对数，a 可为小数
double	**log10(DOUBLE a)**	以 10 为底 d 的对数，a 可为小数
double	**log2(DOUBLE a)**	以 2 为底数 d 的对数，a 可为小数
double	**log(DOUBLE base, DOUBLE a)**	以 base 为底的 a 对数
double	**pow(DOUBLE a, DOUBLE p)**	计算 a 的 p 次幂
double	**sqrt(DOUBLE a)**	计算 a 的平方根
double	**abs(DOUBLE a)**	计算 a 的绝对值
int	**pmod(INT a, INT b)**	a 除以 b 取余
double	**sin(DOUBLE a)**	求 a 的正弦值
double	**asin(DOUBLE a**	求 a 的反正弦值
double	**cos(DOUBLE a**	求 a 余弦值
double	**e()**	数学常量 e
double	**pi()**	数学常量Π
int,double	**negative(a)**	返回 a 的相反数
int	**factorial(INT a)**	求 a 的阶乘
double	**cbrt(DOUBLE a)**	求 a 的立方根
int,double	**greatest(T v1, T v2, ...)**	求最大值
Int,double	**least(T v1, T v2, ...)**	求最小值

A.5 日期函数

返回值	函数	描述
int	**unix_timestamp()**	获取本地时区下的时间戳
string	**from_unixtime(bigint unixtime[, string format])**	将时间的秒值转换成 format 格式
string	**to_date(string timestamp)**	返回日期时间字符串的日期部分。
int	**year(string date)**	返回日期时间字符串的年份部分。
int	**quarter(date/timestamp/string)**	返回当前日期时间属性哪个季度。
int	**month(string date)**	返回日期时间字符串的月份部分。
int	**day(string date) dayofmonth(date)**	返回日期时间字符串的天。
int	**hour(string date)**	返回时间字符串的小时。
int	**minute(string date)**	返回时间字符串的分钟。
int	**second(string date)**	返回时间字符串的秒。
int	**weekofyear(string date)**	返回时间字符串位于一年中的第几个周内。
int	**datediff(string enddate ,string startdate)**	计算开始时间 startdate 到结束时间 enddate 相差的天数。
string	**date_add(string startdate, int days)**	从开始时间 startdate 加上 days。

string	**date_sub(string startdate, int days)**	从开始时间 startdate 减去 days。
string	**current_date**	返回当前时间日期
string	**current_timestamp**	返回当前时间戳
string	**add_months(string start_date, int num_months)**	返回当前时间下再增加 num_months 个月的日期
string	**last_day(string date)**	返回这个月的最后一天的日期，忽略时分秒部分（HH:mm:ss）
string	**trunc(string date, string format)**	返回时间的最开始年份或月份
double	**months_between(date1, date2)**	返回 date1 与 date2 之间相差的月份，如 date1>date2，则返回正，如果 date1<date2，则返回负，否则返回 0.0
string	**date_format(date/timestamp/ string ts, string fmt)**	按指定格式返回时间 date
string	**next_day(string start_date, string day_of_week)**	返回当前时间的星期几所对应的日期

A.6 类型转换和窗口函数

返回值	函数	描述
type	**cast(expr as <type>)**	将 expr 转换成 type 类型
int	**NTILE(n)**	用于将分组数据按照顺序切分成 n 片，返回当前切片值。
int	**ROW_NUMBER()**	从 1 开始，按照顺序，生成分组内记录的序列
int	**RANK()**	在分组中的排名，排名相等会在名次中留下空缺位。
int	**DENSE_RANK()**	在分组中的排名，排名相等会在名次中不会留下空缺位。
type	**LAG(col,n,DEFAULT)**	用于统计窗口内往上第 n 行值。**col** 为列名，**n** 为往上第 n 行（可选，默认为 1），**DEFAULT** 为默认值（当往上第 n 行为 NULL 时候，取默认值，如不指定，则为 NULL）。
type	**LEAD(col,n,DEFAULT)**	用于统计窗口内往下第 n 行值。
type	**FIRST_VALUE(col)**	取分组内排序后，截止到当前行，col 列第一个值。
type	**LAST_VALUE(col)**	取分组内排序后，截止到当前行，col 列最后一个值