[SQL Server 分析函数和排名函数](https://www.cnblogs.com/ljhdo/p/10528448.html)

分析函数基于分组，计算分组内数据的聚合值，经常会和窗口函数OVER()一起使用，使用分析函数可以很方便地计算同比和环比，获得中位数，获得分组的最大值和最小值。分析函数和聚合函数不同，不需要GROUP BY子句，对SELECT子句的结果集，通过OVER()子句分组。

使用以下脚本插入示例数据：

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif View Code

**一，分析函数**

分析函数通常和OVER()函数搭配使用，SQL Server中共有4类分析函数。

注意：distinct子句的执行顺序是在分析函数之后。

**1，CUME\_DIST 和PERCENT\_RANK**

CUME\_DIST 计算的逻辑是：小于等于当前值的行数/分组内总行数

PERCENT\_RANK 计算的逻辑是：（分组内当前行的RANK值-1）/ （分组内总行数-1），排名值是RANK()函数排序的结果值。

以下代码，用于计算累积分布和排名百分比：

[复制代码](javascript:void(0);)

select Department

,LastName

,Rate

,cume\_dist() over(partition by Department order by Rate) as CumeDist

,percent\_rank() over(partition by Department order by Rate) as PtcRank

,rank() over(partition by Department order by Rate asc) as rank\_number

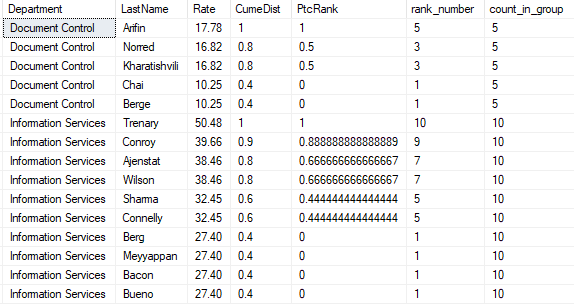
,count(**0**) over(partition by Department) as count\_in\_group

from #data

order by DepartMent

,Rate desc

[复制代码](javascript:void(0);)



**2，PERCENTILE\_CONT和PERCENTILE\_DISC**

PERCENTILE\_CONT和PERCENTILE\_DISC都是为了计算百分位的数值，比如计算在某个百分位时某个栏位的数值是多少。

PERCENTILE\_CONT ( numeric\_literal ) WITHIN GROUP ( ORDER BY order\_by\_expression [ ASC | DESC ] ) OVER ( [ <partition\_by\_clause> ] )

PERCENTILE\_DISC ( numeric\_literal ) WITHIN GROUP ( ORDER BY order\_by\_expression [ ASC | DESC ] ) OVER ( [ <partition\_by\_clause> ] )

这两个函数的区别是前者是连续型，后者是离散型。CONT代表continuous，连续值，DISC代表discrete，离散值。PERCENTILE\_CONT是连续型，意味它考虑的是区间，所以值是绝对的中间值；而PERCENTILE\_DISC是离散型，所以它更多考虑向上或者向下取舍，而不会考虑区间。

以下脚本用于获得分位数：

[复制代码](javascript:void(0);)

select Department

,LastName

,Rate

,PERCENTILE\_CONT(**0.5**) WITHIN GROUP (ORDER BY Rate) OVER (PARTITION BY Department) AS MedianCont

,PERCENTILE\_DISC(**0.5**) WITHIN GROUP (ORDER BY Rate) OVER (PARTITION BY Department) AS MedianDisc

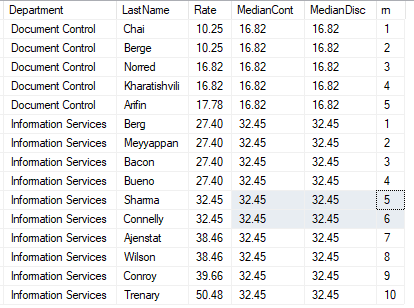
,row\_number() over(partition by Department order by Rate) as rn

from #data

order by DepartMent

,Rate asc

[复制代码](javascript:void(0);)



**3，LAG和LEAD**

在一次查询中，对于同一个字段进行排序，Lag 函数用于获取同一分组内的前N行，Lead函数用于获取同一分组内的后N行，

LAG (scalar\_expression [,offset] [,default])

OVER ( [ partition\_by\_clause ] order\_by\_clause )

LEAD ( scalar\_expression [ ,offset ] , [ default ] )

OVER ( [ partition\_by\_clause ] order\_by\_clause )

结果日期，这两个函数特别适合用于计算同比和环比。

[复制代码](javascript:void(0);)

select DepartMent

,LastName

,Rate

,lag(Rate,**1**,**0**) over(partition by Department order by LastName) as LastRate

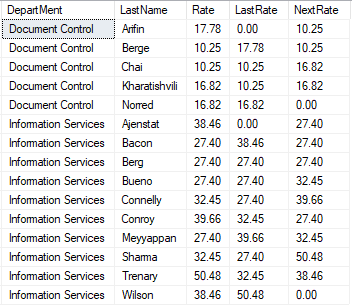
,lead(Rate,**1**,**0**) over(partition by Department order by LastName) as NextRate

from #data

order by Department

,LastName

[复制代码](javascript:void(0);)



**4，FIRST\_VALUE和LAST\_VALUE**

 获取分组内的最大值和最小值，分组内的最大值和最小值是唯一的。

LAST\_VALUE ( [scalar\_expression ) OVER ( [ partition\_by\_clause ] order\_by\_clause rows\_range\_clause )

FIRST\_VALUE ( [scalar\_expression ] ) OVER ( [ partition\_by\_clause ] order\_by\_clause [ rows\_range\_clause ] )

**二，排名函数**

SQL Server的排名函数是对查询的结果进行排名和分组，TSQL共有4个排名函数，分别是：RANK、NTILE、DENSE\_RANK和ROW\_NUMBER，和OVER()函数搭配使用，按照特定的顺序排名。

**1，ROW\_NUMBER函数**

ROW\_NUMBER函数实际上是一个序列，每个分组内都会创建一个序列，序列从1开始，按照顺序依次 +1 递增。

ROW\_NUMBER ( )

OVER ( [ PARTITION\_BY\_clause ] order\_by\_clause )

分组内序列的最大值就是该分组内的行的数目。

**2，RANK函数**

RANK函数用于排名时，不会返回连续的整数。RANK函数的语法是：在分组内，按照特定的顺序排名，序号从1依次递增，排名函数以tie为单位，每个tie中的所有行的排名是相同的，排名可能是不连续的。

RANK ( ) OVER ( [ partition\_by\_clause ] order\_by\_clause )

排名的算法是：

* step1：按照指定的分区字段分组，在每个分组内按照指定的字段排序。
* step2：在每个分组内，如果相邻的两行或多行相同在排序字段上的值相同，那么这些行称作一个tie，每个tie中的所有行都会获得相同的排名。
* step3：后面的排名会计算每个tie中的行数，RANK函数不总是返回连续的整数，例如，班级中，A，B分数都是100分，C的分数是90分，那么A和B的排名是1，**C的排名是3**。

**3，DENSE\_RANK**

DENSE\_RANK函数用于排名时，会返回连续的整数。每个tie占用一个排名，每个tie中的所有行的排名是相同的。排名值是连续的

DENSE\_RANK ( ) OVER ( [ <partition\_by\_clause> ] < order\_by\_clause > )

排名的算法是：

* step1：按照指定的分区字段分组，在每个分组内按照指定的字段排序。
* step2：在每个分组内，如果相邻的两行或多行相同在排序字段上的值相同，那么这些行称作一个tie，每个tie中的所有行都会获得相同的排名。
* step3：后面的排名会计算每个tie中的行数，RANK函数总是返回连续的整数，例如，班级中，A，B分数都是100分，C的分数是90分，那么A和B的排名是1，**C的排名是2**。

**4，NTILE**

在每个分组中，NTILE按照指定的顺序，把数据行分为N个小组（tile），NTILE返回小组编号。在每个分组内，具有相同的小组编号的数据行，位于同一个小组。注意：**小组的编号是按照行数，而不是按照列值。在同一分组内，存在两行的列值相同，而小组编号不同。**

NTILE (integer\_expression) OVER ( [ <partition\_by\_clause> ] < order\_by\_clause > )

如果分区中的行数不能被integer\_expression整除，那么会导致小组相差一个成员：较大的小组按OVER子句指定的顺序位于较小的小组之前。 例如，如果把8行分为3个小组，前2个小组有3行，后一个小组有2行。

如果分区中的中行数能被integer\_expression整除，那么每个小组具有相同的行数。

特别地，NTILE(4) 把一个分组分成4份，叫做Quartile。例如，以下脚本显示各个排名函数的执行结果：

[复制代码](javascript:void(0);)

select Department

,LastName

,Rate

,row\_number() over(order by Rate) as [row number]

,rank() over(order by rate) as rate\_rank

,dense\_rank() over(order by rate) as rate\_dense\_rank

,ntile(**4**) over(order by rate) as quartile\_by\_rate

from #data

[复制代码](javascript:void(0);)

