一文搞定数分面试必考题之：窗口函数

数据与智能 今天

以下文章来源于赵小洛洛洛 ， 作者饭小米

赵小洛洛洛

一枚来自北京的互联网行业的数据分析师，主要分享互联网数据分析、产品、运营相关...

窗口函数的主要作用是对数据进行分组排序、求和、求平均值、计数等。对于数据从业者来说， sql窗口函数在实际工作中具备非常广泛的应用场景。可以大大的提高数据查询效率，同时也是数据类相关岗位的面试/笔试的必考点。所以不论是在职的分析师，还是准备找工作的同学，都必须要牢牢掌握窗口函数的概念及用法。感谢群友饭小米的投稿，接下来让我们详细了解一下窗口函数的前世今生吧。

窗口函数基本语法

```
<窗口函数> OVER ([PARTITION BY <列清单>]
                 ORDER BY <排序用列清单>)
```

在窗口函数的基本语法中，最重要的是去理解partition by，partition by划分的范围被称为窗口，这也是窗口函数的由来。其次是order by，它决定着窗口范围内的数据以什么样的方式排序。下面的例子详细的介绍了窗口函数的基本语法和功能。

例一 代码如下

```
SELECT product_name, product_type, sale_price,
       RANK () OVER (PARTITION BY product_type
                     ORDER BY sale_price) AS ranking
  FROM Product;
```

在上面的代码中可以看出，是按照产品的类型去分组，在组内以价格的顺序升序排列，运行的结果如下。（rank的排序下面会单独说）

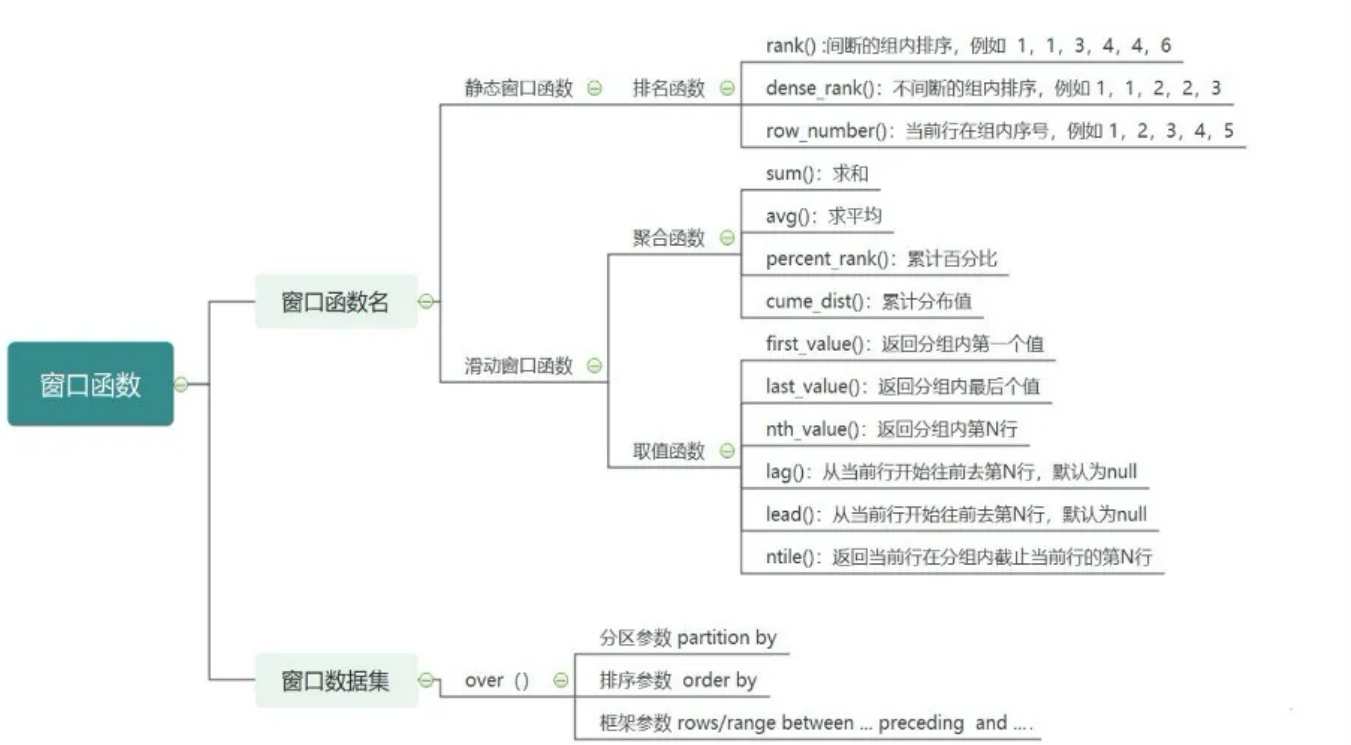
product_name	product_type	sale_price	ranking
叉子	厨房用具	500	1
擦菜板	厨房用具	880	2
菜刀	厨房用具	3000	3
高压锅	厨房用具	6800	4
T恤衫	衣服	1000	1
运动T恤	衣服	4000	2
圆珠笔	办公用品	100	1
打孔器	办公用品	500	2

至于窗口函数与group by的区别：

- **两个order by的区别**，第一个窗口函数中的order by只是决定着窗口里的数据的排序方式，第二个普通的order by决定查询出的数据以什么样的方式整体排序；
- 窗口函数可以在保留原表中的全部数据之后，可以对某些字段做分组排序或者计算，而group by只能保留与分组字段聚合的结果；
- 在加入窗口函数的基础上SQL的执行顺序也会发生变化，具体的执行顺序如下（window就是窗口函数）；

窗口函数类别

专用窗口函数例如rank、row_number、lag和lead等，在窗口函数中有静态函数和动态函数的分类，具体的划分如下。



作为窗口函数的聚合函数，常见的聚合函数有sum、avg、max、min跟count。他们跟窗口函数组合到一起，就会把聚合函数的功能和窗口函数组合在一起。

例二 代码及结果为

```
select*,
sum(成绩)over (order by 学号),
avg(成绩)over (order by 学号),
count(成绩)over (order by 学号),
max(成绩)over (order by 学号),
min(成绩)over (order by 学号),
```

学号	班级	成绩	current_sum	current_avg	current_count	current_max	current_min
0001	1	86	86	86.0000	1	86	86
0002	1	95	181	90.5000	2	95	86
0003	2	89	270	90.0000	3	95	86
0004	1	83	353	88.2500	4	95	83
0005	2	86	439	87.8000	5	95	83
0006	3	92	531	88.5000	6	95	83
0007	3	86	617	88.1429	7	95	83
0008	1	88	705	88.1250	8	95	83

从上面的例子可以看出，在没有partition by 的情况下，是把整个表作为一个大的窗口，SUM（）相当于向下累加，AVG（）相当于求从第一行到当前行的平均值，其他的聚合函数均是如此。

注意点：

1 、在使用专用的窗口函数时，例如rank、lag等，rank（）括号里是不需要指定任何字段的，直接空着就可以；

2 、在使用聚合函数做窗口函数时，SUM（）括号里必须有字段，得指定对哪些字段执行聚合的操作。在学习的初期很容易弄混，不同函数括号里是否需写相应的字段名；

三种分组排序的区别-row_number、rank、dense_rank

- RANK-计算排序时，如果存在相同位次的记录，则会跳过之后的位次。有 3 条记录排在第 1 位时：1 、1 、1 、4；
- DENSE_RANK-同样是计算排序，即使存在相同位次的记录，也不会跳过之后的位次。有 3 条记录排在第 1 位时：1 、1 、1 、2；
- ROW_NUMBER-赋予唯一的连续位次,有 3 条记录排在第 1 位时：1 、2 、3 、4；

示例，在下面的执行结果是以整个表作为窗口，可以清楚的看出三种排序函数的不同之处。（如果想要唯一的排序就直接用row_number）

执行结果

RANK DENSE_RANK ROW_NUMBER

product_name	product_type	sale_price	ranking	dense_ranking	row_num
圆珠笔	办公用品	100	1	1	1
叉子	厨房用具	500	2	2	2
打孔器	办公用品	500	2	2	3
擦菜板	厨房用具	880	4	3	4
T恤衫	衣服	1000	5	4	5
菜刀	厨房用具	3000	6	5	6
运动T恤	衣服	4000	7	6	7
高压锅	厨房用具	6800	8	7	8

窗口函数进阶-滑动窗口函数

在写窗口函数时，order by后面可以有参数，rows/range 和preceding跟following，在组合使用这些参数后，窗口就会变成滑动窗口，因为涉及到动态窗口，所以在理解上比较抽象。

1、 Preceding

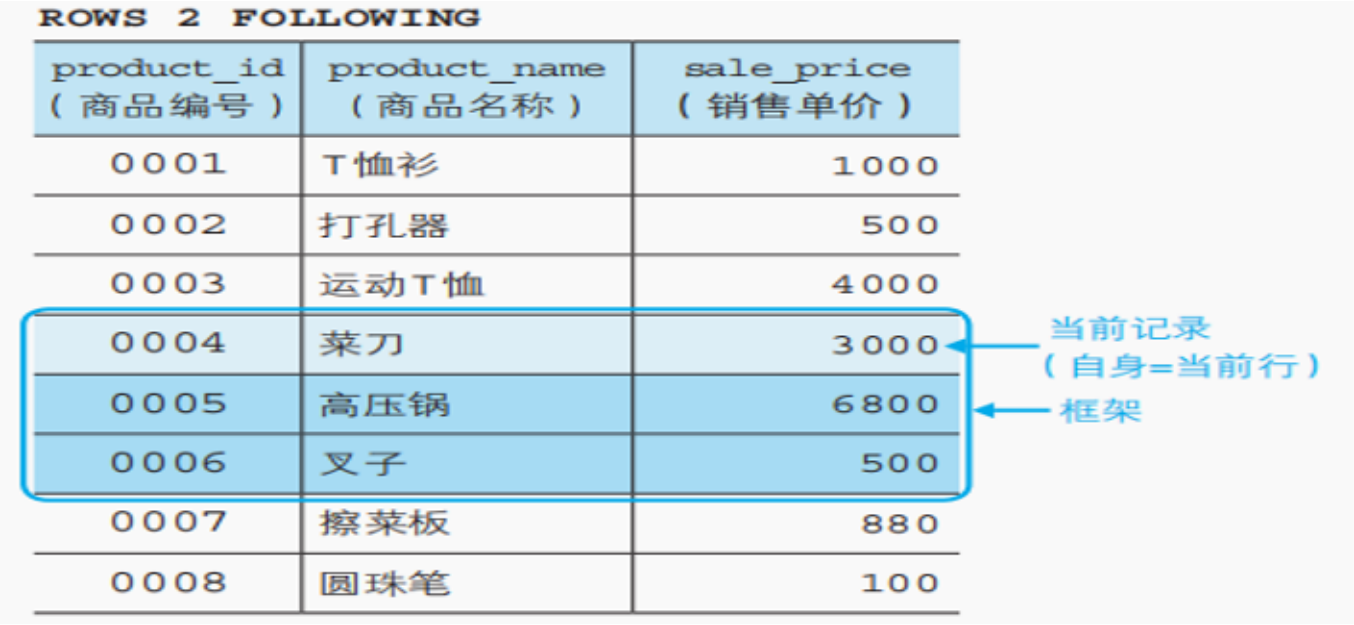
```
SELECT product_id, product_name, sale_price,
       AVG (sale_price) OVER (ORDER BY product_id
                              ROWS 2 PRECEDING) AS moving_avg
  FROM Product;
```

product_id	product_name	sale_price	moving_avg	
0001	T恤衫	1000	1000	←(1000)/1
0002	打孔器	500	750	←(1000+500)/2
0003	运动T恤	4000	1833	←(1000+500+4000)/3
0004	菜刀	3000	2500	←(500+4000+3000)/3
0005	高压锅	6800	4600	←(4000+3000+6800)/3
0006	叉子	500	3433	•
0007	擦菜板	880	2726	•
0008	圆珠笔	100	493	•

Rows 2 preceding 中文的意思是之前的两行，preceding可以把它理解为不含当前行情况下截止到之前几行。根据上图可以看出在每一行，都会求出当前行附近的3行(当前行+附近2行)数据的平均值，这种方法也叫作**移动平均**。

2、Following

Rows 2 following 中文意思是之后的两行，跟preceding正好相反，Preceding是向前，following是向后。



product_id（商品编号）	product_name（商品名称）	sale_price（销售单价）
0001	T恤衫	1000
0002	打孔器	500
0003	运动T恤	4000
0004	菜刀	3000
0005	高压锅	6800
0006	叉子	500
0007	擦菜板	880
0008	圆珠笔	100

3、preceding跟following相结合

代码及运行结果为：

```
SELECT product_id, product_name, sale_price,
       AVG (sale_price) OVER (ORDER BY product_id
                              ROWS BETWEEN 1 PRECEDING AND
                               1 FOLLOWING) AS moving_avg
  FROM Product;
```

product_id	product_name	sale_price	moving_avg	
0001	T恤衫	1000	750	←(1000+500)/2
0002	打孔器	500	1833	←(1000+500+4000)/3
0003	运动T恤	4000	2500	←(500+4000+3000)/3
0004	菜刀	3000	4600	←(4000+3000+6800)/3
0005	高压锅	6800	3433	·
0006	叉子	500	2726	·
0007	擦菜板	880	493	·
0008	圆珠笔	100	490	·

从以上的运行结果可以看出是把每一行（当前行）的前一行和后一行作为汇总的依据。

窗口函数应用真题解析

1、topN问题或者组内排序问题

在实际的场景中，我们会经常会遇到排序或者排名问题，这个时候使用窗口函数会使问题变的简单。

求出每个课程的学生成绩排名：

```
37  -- 查询各科成绩前两名的记录
38  Select *
39  From (Select *,row_number() over (partition by 课程号
40  order by 成绩 DESC) as 排名
41  From score) as a
42  Where 排名<=2;
```

信息 | 结果1 | 概况 | 状态

学号	课程号	成绩	排名
0001	0001	80	1
0003	0001	80	2
0001	0002	90	1
0003	0002	80	2
0001	0003	99	1
0002	0003	80	2

2、连续登录问题

假设有一张含两列(用户id、登陆日期)的表，查询每个用户连续登陆的天数、最早登录时间、最晚登录时间和登录次数。

- 首先要对数据进行去重，防止同一个用户一天之内出现连续登录的情况；
- 假如一个用户是连续登录的话，用login_time-窗口函数的排序后得到的日期应该是一样的，连续登录的用户前后之间的时间差就是一个差值为1的等差数列；

第一步，先用row_number（）函数排序，然后用登录日期减去排名，得到辅助列日期，如果辅助列日期是相同的话，证明用户是连续登录。
运行的代码及结果为：

```
SELECT
  a.user_id,
  a.date,
  a.排序,
  date_sub( a.date, INTERVAL a.排序 DAY ) AS 辅助列
FROM
  ( SELECT user_id, date( login_time ) AS date, ROW_NUMBER() OVER ( PARTITION BY user_id ORDER BY login_time ) 排序 FROM user_login ) AS a
```

user_id	date	排序	辅助列
1	2016-10-31	1	2016-10-30
1	2016-11-01	2	2016-10-30
1	2016-11-09	3	2016-11-06
1	2016-11-10	4	2016-11-06
1	2016-11-23	5	2016-11-18
1	2016-11-24	6	2016-11-18
1	2016-11-24	7	2016-11-17
1	2016-11-24	8	2016-11-16

第二步，用user_id和辅助列作为分组依据，分到一组的就是连续登录的用户。在每一组中最小的日期就是最早的登陆日期，最大的日期就是最近的登陆日期，对每个组内的用户进行计数就是用户连续登录的天数。

运行代码及结果为：

```
SELECT
  b.user_id,
  MAX( b.date ) AS 最近登陆日期,
  MIN( b.date ) AS 最早登陆日期,
  COUNT( b.date ) AS 登陆次数
FROM
  (
  SELECT
    distinct a.user_id,
    a.date,
    a.排序,
    date_sub( a.date, INTERVAL a.排序 DAY ) AS 辅助列
  FROM
    ( SELECT user_id, date( login_time ) AS date, ROW_NUMBER() OVER ( PARTITION BY user_id ORDER BY login_time ) 排序 FROM user_login ) AS a
  ) b
GROUP BY
  b.user_id,
  b.`辅助列`
```

user_id	最近登陆日期	最早登陆日期	登陆次数
1	2016-11-01	2016-10-31	2
1	2016-11-10	2016-11-09	2
1	2016-11-24	2016-11-23	2
1	2016-11-24	2016-11-24	1
1	2016-11-25	2016-11-24	2
2	2016-11-02	2016-10-29	5
2	2016-11-25	2016-11-19	7

若求解每个用户的最大登录天数。其实可以在以上的查询结果为基础，利用聚合函数就可以求出最大的登录天数问题。假如求解连续登录5天的用户，除了可以使用上述的方法，还可以使用lead函数进行窗口偏移来进行求解。

示例：数据还是上题中的数据，求解连续登录五天的用户

第一步，用lead函数进行窗口偏移，查找每个用户5天后的登陆日期是多少，如果是空值，说明他没有登录。运行的代码为

```sql
SELECT
  DISTINCT user_id,
  date( login_time ) AS 日期,
  LEAD( date( login_time ), 4 ) OVER ( PARTITION BY user_id ORDER BY login_time ) AS 第五次登陆日期
FROM
  user_login
```

在lead函数里，为何偏移行数的参数设置为4而不是5呢，这是因为求解的是连续登录5天的用户，包括当前行在内一共是5行，所以应该向下偏移4行。运行的结果如下：

user_id	日期	第五次登陆日期
1	2016-10-31	2016-11-23
1	2016-11-01	2016-11-24
1	2016-11-09	2016-11-24
1	2016-11-10	2016-11-24
1	2016-11-23	2016-11-25
1	2016-11-24	(Null)
1	2016-11-25	(Null)

第二步，用datediff函数计算 （日期-第五次登陆日期）+1是否等于5，等于5证明用户是连续5天登录的，为空值或者大于5都不是5天连续登陆的用户。

第三步，用where设定条件，差值=5筛选连续登录的用户。

二、三步运行的代码为：

```sql
SELECT DISTINCT
  b.user_id
FROM
  (
  SELECT
    user_id,
    a.日期,
    a.第五次登陆日期,
    DATEDIFF( a.第五次登陆日期, a.日期 )+ 1 AS 相差天数
  FROM
    (
    SELECT DISTINCT
      user_id,
      date( login_time ) AS 日期,
      LEAD( date( login_time ), 4 ) OVER ( PARTITION BY user_id ORDER BY login_time ) AS 第五次登陆日期
    FROM
      user_login
    ) AS a
  ) AS b
WHERE
  b.`相差天数` >=5
```

用lead函数求解连续登录的问题还有一个好处就是当表中的数据不在同一个月份时也可以完美的解决，不用再考虑月份带来的影响。