【SQL】17.数据分析利器---窗口函数（二）

原创 数据码头 [数据码头](javascript:void(0);)

 2025年06月10日 08:00 江苏

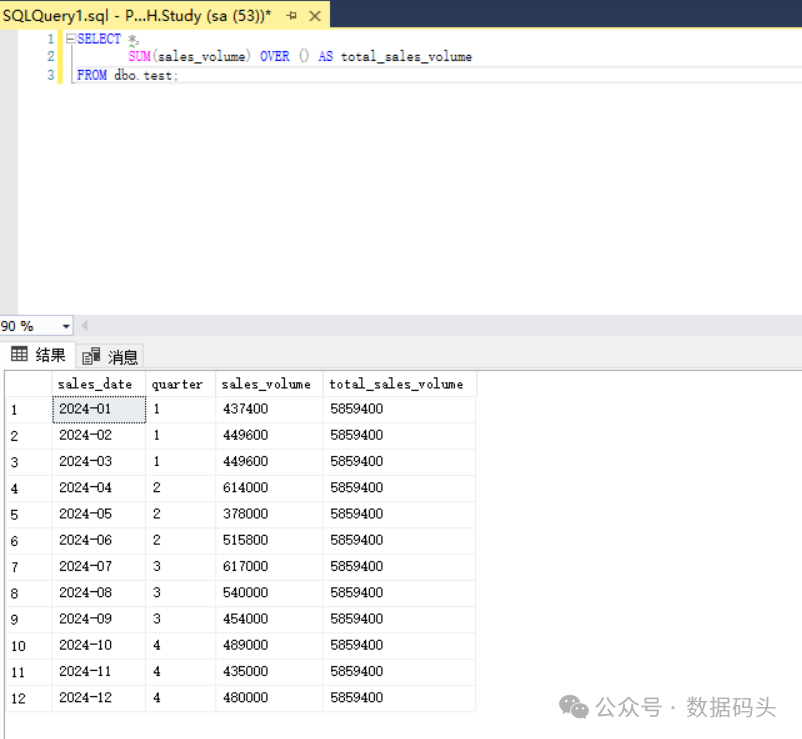
上篇介绍了窗口函数的语法以及三种排序相关的函数，还没看的记得先看下，今天继续讲窗口函数。

我们常用的聚合函数，像sum、max、avg这些都可以用作窗口函数，下面就用这张test表中的数据详细详细演示一下不同的情形：

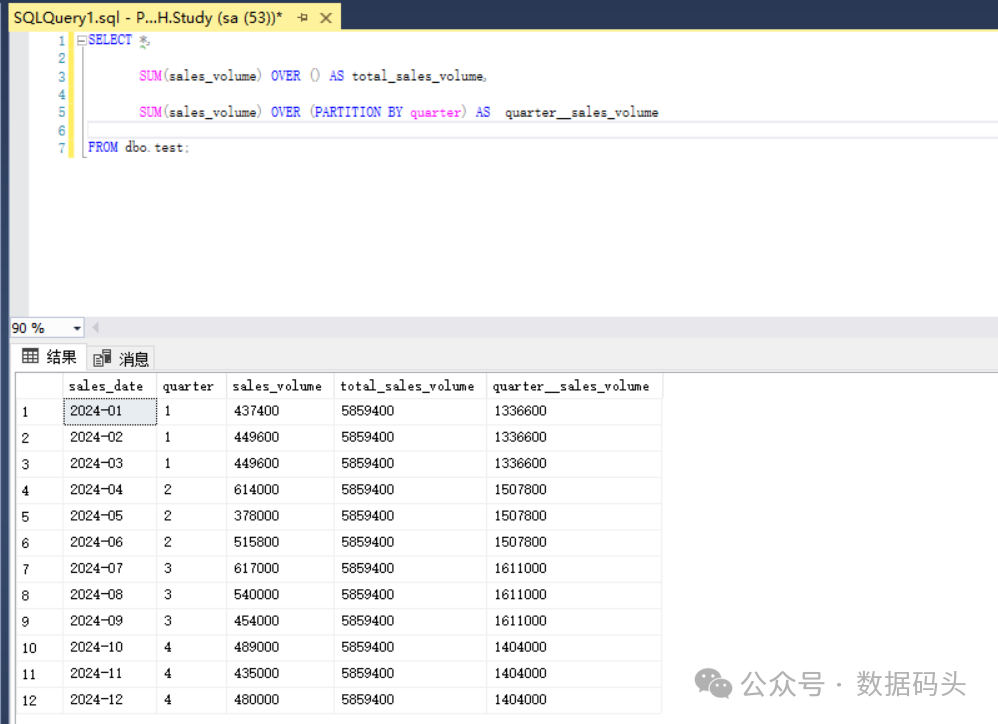


**1.计算总销售额**如果要在每一行后面加上所有月份的总销售额total\_sales\_volume可以使用如下的窗口函数：

SELECT \*,  SUM(sales\_volume) OVER () AS total\_sales\_volumeFROM dbo.test;

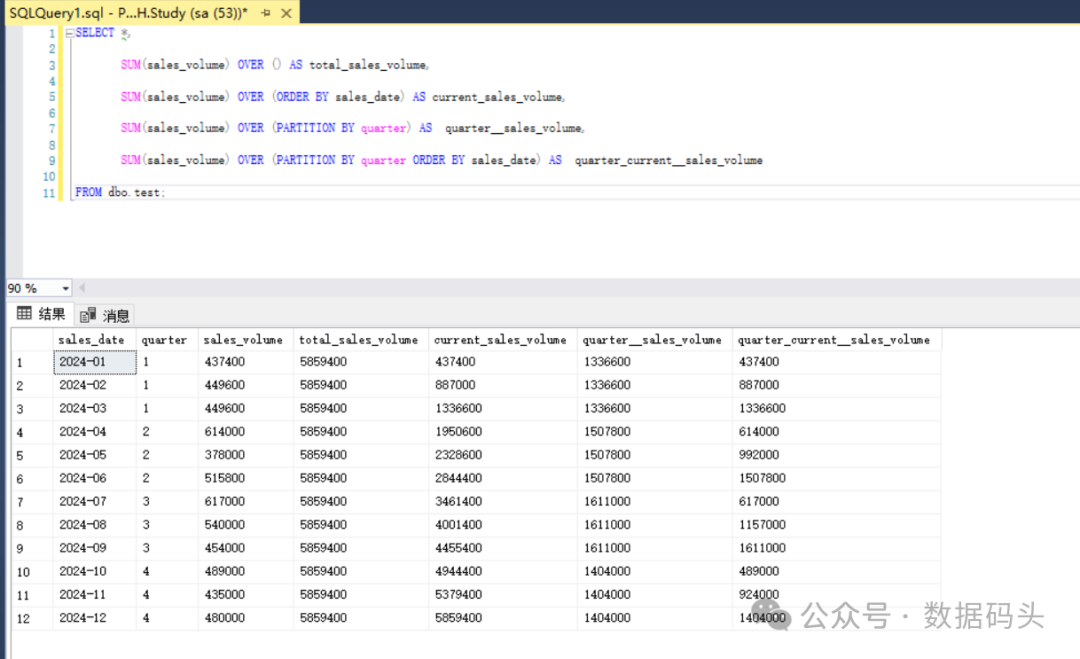
其中sum是求和，over()中不添加参数，则默认对所有数据进行求和，可以看到新加的一列total\_sales\_volume，每一行的结果都是5859400。**2.每个季度的总销售额**年度总销售额知道了，但如果想求出每个季度的销售总额应该怎么办呢？回忆一下之前的聚合函数，是不是得根据季度来进行分组？在窗口函数中，PARTITION BY就可以对数据进行分组，将数据拆分成一个个窗口，sql如下：

SELECT \*,  SUM(sales\_volume) OVER () AS total\_sales\_volume,  SUM(sales\_volume) OVER (PARTITION BY quarter) AS  quarter\_sales\_volumeFROM dbo.test;

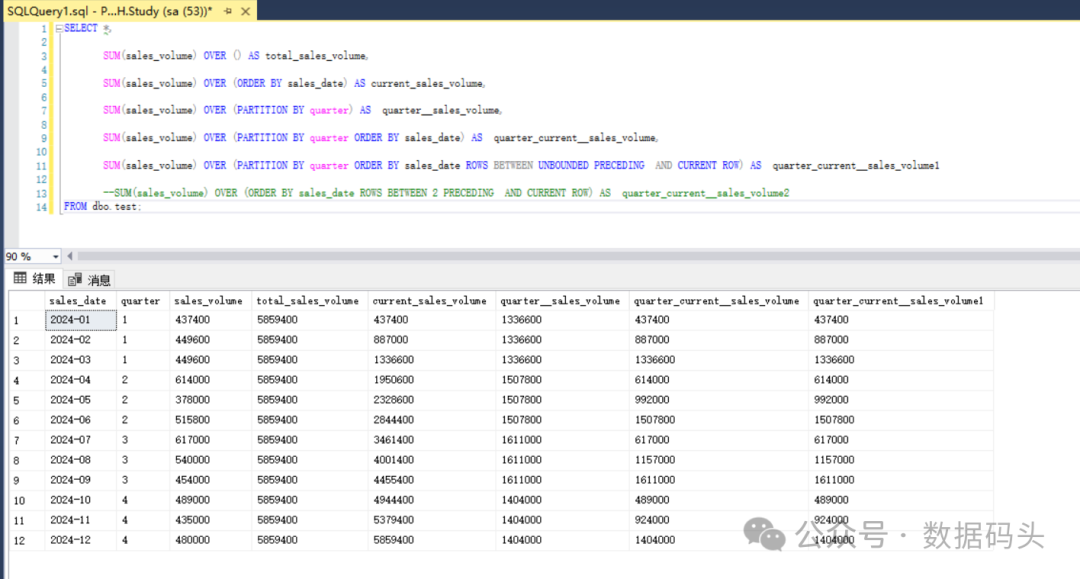
PARTITION BY的功能与GROUP BY的类似，指定按照哪一列进行分组，只是PARTITION BY没有对原数据进行压缩，而是在原表的基础上增加了一列。上面的sql用 quarter分组，则将原来的12行数据拆分成了4个窗口，在小窗口里面对每个季度的值进行计算。

**3.累计销售额**年度和季度的总销售额我们求出来了，但如果想求出累计至当月的销售额，比如说从1月到3，到8月累计销量分别是多少，类似从开始到指定日期的总进度，应该怎么求呢？即对窗口里的数据来进行移动统计，这时候就得加上另外一个关键字：ORDER BY了。sql如下：

SELECT \*, SUM(sales\_volume) OVER () AS total\_sales\_volume, SUM(sales\_volume) OVER (ORDER BY sales\_date) AS current\_sales\_volume, SUM(sales\_volume) OVER (PARTITION BY quarter) AS  quarter\_\_sales\_volume, SUM(sales\_volume) OVER (PARTITION BY quarter ORDER BY sales\_date) AS  quarter\_current\_\_sales\_volumeFROM dbo.test;

上面的current\_sales\_volume是截止到当月，这一年的销量，可以看到每个月的值都是逐行累加的。quarter\_current\_\_sales\_volume是在每个季度这个小组中，对销量进行累加，比如2月的季度销量就是1月的加上2月的，但是到4月份了，季度累计销量会重新计算，因为是第二季度了。通过上面4个SQL语句，大家应该能看出来：加不加PARTITION BY、加不加ORDER BY结果分别有什么区别了。如果把上面这4种情况都理解了，那么常见的求和、分组求和应该没什么问题了。如果再拓展一点的话，就是开窗函数其实除了上面那种常见的写法外，后面还可以指定分行，只是通常被省略了，如果补充完整应该是：

OVER (PARTITION BY quarter ORDER BY sales\_date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

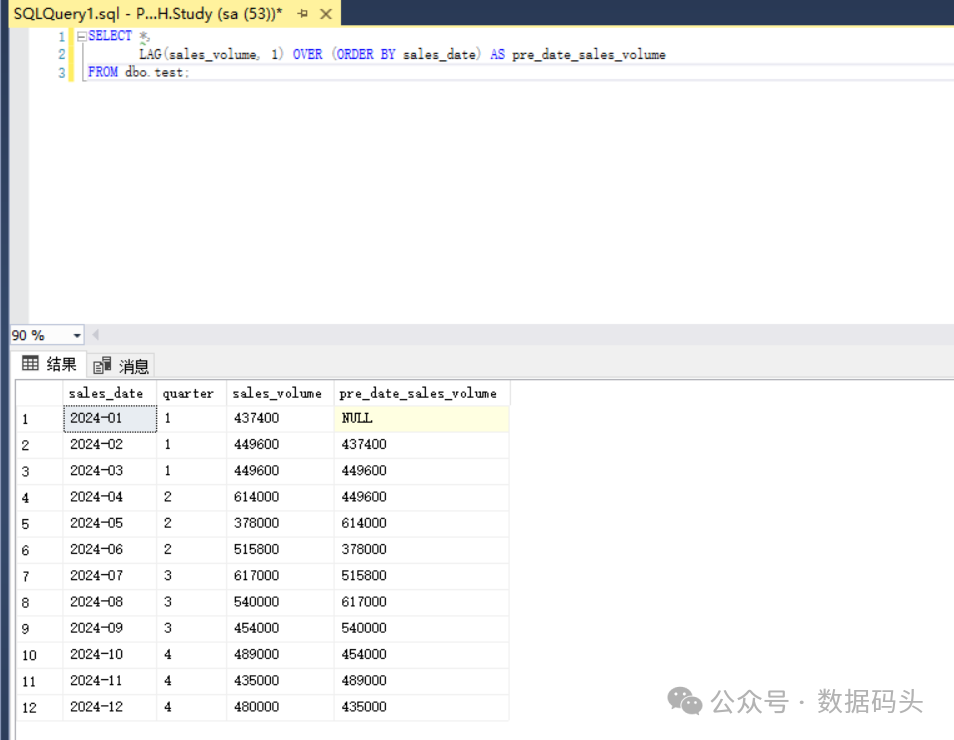


可以看出上面补充完整后计算的结果和我们刚求出来的是一样。只是我们可以指定求多少行到多少行的累计数据，比如说我上图中最后注释的那行代码，相对于提供了移动的窗口，如果感兴趣的话，可以自己尝试一下。只是这种写法很少用到，只要了解即可。

上面通过sum给大家演示了开窗函数的具体用法，只要求和会了，剩下的avg求平均，max、min求最大最小值都是一样的道理，相信大家没什么问题。

**4.偏移开窗函数**还有一种偏移类型的开窗函数，如果我们要取当前记录的前/后几位的数据，需要用到偏移函数lag/lead。lag是在当前行之前查找，lead是在当前行之后查找。比如求环比增长率，就需要知道当前月的销售额和前一个月的销售额，那么就可以用偏移函数,sql如下：

SELECT \*,  LAG(sales\_volume, 1) OVER (ORDER BY sales\_date) AS pre\_date\_sales\_volumeFROM dbo.test;

函数的第一个参数是我们要处理的列sales\_volume，第二个参数是偏移量，当然如果不指定的话默认也是1，还有第三个参数是在处理的偏移量没有返回值时所给的默认值，比如第一行没有前一行，返回NULL，我们可以设定如果没有前一个值，返回0，那么sql就可以改成：

SELECT \*,   LAG(sales\_volume,1,0) OVER (ORDER BY sales\_date) AS pre\_date\_sales\_volumeFROM dbo.test;

通过上面的偏移函数，我们就很容易解决同比增长、环比增长这种情形。还有lead是取向后偏移的值，用法也是同理。  
以上就是窗口函数的内容，熟悉窗口函数相信定能帮助你处理数据分析相关的问题。如果你在学习或工作中遇到相关问题，欢迎随时交流探讨！如果觉得不错，也欢迎分享、点赞和推荐哈~

|  |  |  |
| --- | --- | --- |
| 分类 | 函数示例 | 说明 |
| 前后值函数 | LAG(col, n, default) | 取当前行前第n行的值 |
|  | LEAD(col, n, default) | 取当前行后第n行的值 |
| 首尾值函数 | FIRST\_VALUE(col) | 窗口第一行的值 |
|  | LAST\_VALUE(col) | 窗口最后一行的值 |

**📈 一、LAG(col, n, default)**

💡 核心价值：不用自连接！轻松获取历史数据

🌰 经典场景：计算环比增长、用户行为路径分析

🚀 案例：电商GMV周环比暴增分析

SELECT  
  统计周,  
  GMV,  
  LAG(GMV, 1, 0) OVER (ORDER BY 统计周) AS 上周GMV, -- 默认0避免NULL  
  ROUND((GMV - LAG(GMV) OVER (ORDER BY 统计周))/LAG(GMV) OVER (ORDER BY 统计周)\*100,2) AS 环比增长率  
FROM 周销售报表

⚠️ 避坑指南：

1️⃣ 必加ORDER BY：无排序的LAG就像乱序时空穿越！

2️⃣ 处理NULL值：用default参数给历史数据兜底

3️⃣ 性能优化：相同OVER()子句可复用，减少重复计算

**🔮 二、LEAD(col, n, default)**

💡 核心价值：提前获取未来数据，轻松计算差值

🌰 经典场景：用户下一步行为预测、库存消耗预警

🚀 案例：用户行为路径分析

SELECT  
  user\_id,  
  event\_time,  
  current\_action,  
  LEAD(current\_action, 1, '退出') OVER (PARTITION BY user\_id ORDER BY event\_time) AS 下一步动作  
FROM 用户行为日志  
WHERE event\_date = '2023-06-18'

⚠️ 避坑指南：

1️⃣ 分区陷阱：一定要用PARTITION BY隔离不同用户/业务

2️⃣ 数据边界：末尾数据默认返回NULL，用default参数给明确状态

3️⃣ 性能杀器：避免在十亿级日志表全量使用，先过滤时间分区！

**📊 三、FIRST\_VALUE(col)**

💡 核心价值：快速定位窗口起点，计算累计值神器

🌰 经典场景：用户首单金额分析、累计销售额计算

🚀 案例：用户生命周期价值分析

SELECT  
  user\_id,  
  order\_date,  
  amount,  
  FIRST\_VALUE(order\_date) OVER (PARTITION BY user\_id ORDER BY order\_date) AS 首单日期,  
  SUM(amount) OVER (PARTITION BY user\_id ORDER BY order\_date) AS 累计消费  
FROM 订单表

⚠️ 避坑指南：

1️⃣ 排序陷阱：不加ORDER BY会导致取值为窗口随机第一行！

2️⃣ 窗口帧控制：默认RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

3️⃣ 去重技巧：结合DISTINCT快速获取用户首单信息

**📉 四、LAST\_VALUE(col)**

💡 核心价值：捕捉窗口终点数据，动态跟踪最新值

🌰 经典场景：库存最新状态跟踪、用户最近一次登录时间

🚀 案例：动态库存监控

SELECT  
  product\_id,  
  stock\_time,  
  current\_stock,  
  LAST\_VALUE(current\_stock) OVER (  
    PARTITION BY product\_id   
    ORDER BY stock\_time   
    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING *-- 关键！*  
  ) AS 最新库存  
FROM 库存流水表

⚠️ 避坑指南：

1️⃣ 窗口帧必改：默认只看到当前行！必须扩展窗口范围

2️⃣ 性能警报：UNBOUNDED FOLLOWING在大数据量时慎用

3️⃣ 替代方案：用LEAD(0)或反向排序的FIRST\_VALUE更高效

**💣 四大函数通用避坑指南：**

1️⃣ 索引优化：窗口函数依赖排序，提前在ORDER BY字段建索引

2️⃣ 内存控制：一次处理10万+数据时，调大数据库的sort\_buffer\_size

3️⃣ 版本陷阱：MySQL 8.0+才能用，MariaDB 10.2+语法略有不同

4️⃣ 执行顺序：窗口函数在WHERE之后执行，不能作为过滤条件

Eg.

WITH 用户订单 AS (  
SELECT  
    user\_id,  
    order\_date,  
    LAG(order\_date) OVER (PARTITIONBY user\_id ORDERBY order\_date) AS 上次购买日,  
    FIRST\_VALUE(order\_date) OVER (PARTITIONBY user\_id ORDERBY order\_date DESC) AS 最近购买日  
FROM 订单表  
)

SELECT  
  user\_id,  
  DATEDIFF(order\_date, 上次购买日) AS 购买间隔,  
  DATEDIFF(CURRENT\_DATE(), 最近购买日) AS 沉默天数   
FROM 用户订单