08 | 聚合函数：怎么高效地进行分组统计？

2021-03-25 朱晓峰

MySQL 必知必会 进入课程 >

讲述：朱晓峰
时长 09:55 大小 9.09M

你好，我是朱晓峰。今天，我来和你聊一聊聚合函数。

MySQL 中有 5 种聚合函数较为常用，分别是求和函数 SUM()、求平均函数 AVG()、最大值函数 MAX()、最小值函数 MIN() 和计数函数 COUNT()。接下来，我就结合超市项目的真实需求，来带你掌握聚合函数的用法，帮你实现高效的分组统计。

咱们的项目需求是这样的：超市经营者提出，他们需要统计某个门店，每天、每个单品的销售情况，包括销售数量和销售金额等。这里涉及 3 个数据表，具体信息如下所示：

销售明细表（demo.transactiondetails）：

transactionid（流水编号，联合主键）	itemnumber（商品编号，联合主键）	quantity（销售数量）	price（销售金额）	salesvalue（销售金额）
1	1	2	89	178
1	2	5	5	25
2	1	3	89	267
2	2	6	5	30
3	1	1	89	89
3	2	10	5	50

销售单头表（demo.transactionhead）：

transactionid（流水编号，主键）	transactionno（流水单号）	cashierid（收款机编号）	memberid（会员编号）	operatorid（操作员编号）	transdate（交易时间）
1	0120201201000001	1	1	1	2020-12-01 14:25:56
2	0120201202000001	1	NULL	2	2020-12-02 10:50:50
3	0120201202000002	1	NULL	1	2020-12-02 12:10:05

商品信息表（demo.goodsmaster）：

Itemnumber（商品编号）	Barcode（条码）	Goodsname（商品名称）	Specification（规格）	Unit（单位）	Salesprice（售价）
1	1	书	16开	本	89
2	2	笔	10支装	包	5

要统计销售，就要用到数据求和，那么我们就先来学习下求和函数 SUM()。

SUM（）

SUM（）函数可以返回指定字段值的和。我们可以用它来获得用户某个门店，每天，每种商品的销售总计数据：

复制代码

```
mysql>  SELECT
    ->      LEFT(b.transdate, 10), -- 从关联表获取交易时间，并且通过LEFT函数，获取交易
    ->      c.goodsname,           -- 从关联表获取商品名称
    ->      SUM(a.quantity),       -- 数量求和
    ->      SUM(a.salesvalue)      -- 金额求和
    -> FROM
    ->      demo.transactiondetails a
    ->          JOIN
    ->      demo.transactionhead b ON (a.transactionid = b.transactionid)
    ->          JOIN
    ->      demo.goodsmaster c ON (a.itemnumber = c.itemnumber)
    -> GROUP BY LEFT(b.transdate, 10) , c.goodsname      -- 分组
    -> ORDER BY LEFT(b.transdate, 10) , c.goodsname;     -- 排序
+-----------------------+-----------+-----------------+-------------------+
| LEFT(b.transdate, 10) | goodsname | SUM(a.quantity) | SUM(a.salesvalue) |
+-----------------------+-----------+-----------------+-------------------+
| 2020-12-01            | 书        |           2.000 |            178.00 |
| 2020-12-01            | 笔        |           5.000 |             25.00 |
| 2020-12-02            | 书        |           4.000 |            356.00 |
| 2020-12-02            | 笔        |          16.000 |             80.00 |
+-----------------------+-----------+-----------------+-------------------+
4 rows in set (0.01 sec)
```

可以看到，我们引入了 2 个关键字：LEFT 和 ORDER BY，你可能对它们不熟悉，我来具体解释下。

LEFT(str，n)：表示返回字符串 str 最左边的 n 个字符。我们这里的 LEFT（a.transdate,10），表示返回交易时间字符串最左边的 10 个字符。在 MySQL 中，DATETIME 类型的默认格式是：YYYY-MM-DD，也就是说，年份 4 个字符，之后是“-”，然后是月份 2 个字符，之后又是“-”，然后是日 2 个字符，所以完整的年月日是 10 个字符。用户要求按照日期统计，所以，我们需要从日期时间数据中，把年月日的部分截取出来。

ORDER BY：表示按照指定的字段排序。超市经营者指定按照日期和单品统计，那么，统计的结果按照交易日期和商品名称的顺序排序，会更加清晰。

知道了 2 个关键字之后，刚刚的查询就容易理解了。接下来我们就再拆解一下，看看这个查询是如何执行的。我用图表来直观地演示一下各个步骤。

第一步，完成 3 个表的连接（由于字段比较多，为了你理解，我省略了一些在这一步不重要的字段）：

a(demo.transactiondetails)			b(demo.transactionhead)			c(demo.goodsmaster)		
itemnumber	quantity	salesvalue	transactionid	transactionno	transdate	goodsname	unit	salesprice
1	2	178	1	0120201201000001	2020-12-01 14:25:56	书	本	89
2	5	25	1	0120201201000001	2020-12-01 14:25:56	笔	包	5
1	3	267	2	0120201202000001	2020-12-02 10:50:50	书	本	89
2	6	30	2	0120201202000001	2020-12-02 10:50:50	笔	包	5
1	1	89	3	0120201202000002	2020-12-02 12:10:05	书	本	89
2	10	50	3	0120201202000002	2020-12-02 12:10:05	笔	包	5

第二步，对结果集按照交易时间和商品名称进行分组，我们可以分成下面 4 组。

第一组：

a(demo.transactiondetails)			b(demo.transactionhead)			c(demo.goodsmaster)		
itemnumber	quantity	salesvalue	transactionid	transactionno	transdate	goodsname	unit	salesprice
1	2	178	1	0120201201000001	2020-12-01 14:25:56	书	本	89

第二组

a(demo.transactiondetails)			b(demo.transactionhead)			c(demo.goodsmaster)		
itemnumber	quantity	salesvalue	transactionid	transactionno	transdate	goodsname	unit	salesprice
2	5	25	1	0120201201000001	2020-12-01 14:25:56	笔	包	5

第三组

a(demo.transactiondetails)			b(demo.transactionhead)			c(demo.goodsmaster)		
itemnumber	quantity	salesvalue	transactionid	transactionno	transdate	goodsname	unit	salesprice
1	3	267	2	0120201202000001	2020-12-02 10:50:50	书	本	89
1	1	89	3	0120201202000002	2020-12-02 12:10:05	书	本	89

第四组

a(demo.transactiondetails)			b(demo.transactionhead)			c(demo.goodsmaster)		
itemnumber	quantity	salesvalue	transactionid	transactionno	transdate	goodsname	unit	salesprice
2	6	30	2	0120201202000001	2020-12-02 10:50:50	笔	包	5
2	10	50	3	0120201202000002	2020-12-02 12:10:05	笔	包	5

第三步，对各组的销售数量和销售金额进行统计，并且按照交易日期和商品名称排序。这样就得到了我们需要的结果，如下所示：

复制代码

```
+----------------------+-----------+-----------------+-------------------+
| LEFT(b.transdate, 10) | goodsname | SUM(a.quantity) | SUM(a.salesvalue) |
+----------------------+-----------+-----------------+-------------------+
| 2020-12-01           | 书        |           2.000 |            178.00 |
| 2020-12-01           | 笔        |           5.000 |             25.00 |
| 2020-12-02           | 书        |           4.000 |            356.00 |
| 2020-12-02           | 笔        |          16.000 |             80.00 |
+----------------------+-----------+-----------------+-------------------+
4 rows in set (0.01 sec)
```

如果用户需要知道全部商品销售的总计数量和总计金额，我们也可以把数据集的整体看作一个分组，进行计算。这样就不需要分组关键字 GROUP BY，以及排序关键字 ORDER BY 了。你甚至不需要从关联表中获取数据，也就不需要连接了。就像下面这样：

复制代码

```
mysql> SELECT
    ->  SUM(quantity), -- 总计数量
    ->  SUM(salesvalue)-- 总计金额
    -> FROM
    ->  demo.transactiondetails;
+---------------+-----------------+
| SUM(quantity) | SUM(salesvalue) |
+---------------+-----------------+
|        27.000 |          639.00 |
+---------------+-----------------+
1 row in set (0.05 sec)
```

到这里呢，求和函数 SUM() 的使用方法我就讲完了。需要提醒你的是，求和函数获取的是分组中的合计数据，所以你要对分组的结果有准确的把握，否则就很容易搞错。这也就是说，你要知道是按什么字段进行分组的。如果是按多个字段分组，你要知道字段之间有什么样的层次关系；如果是按照以字段作为变量的某个函数进行分组的，你要知道这个函数的返回值是什么，返回值又是如何影响分组的等。

AVG（）、MAX（）和 MIN（）

接下来，我们来计算一下分组中数据的平均值、最大值和最小值。这个时候，就要用到 AVG()、MAX() 和 MIN() 了。

1.AVG（）

首先，我们来学习下计算平均值的函数 AVG（）。它的作用是，通过计算分组内指定字段值的和，以及分组内的记录数，算出分组内指定字段的平均值。

举个例子，如果用户需要计算每天、每种商品，平均一次卖出多少个、多少钱，这个时候，我们就可以用到 AVG（）函数了，如下所示：

复制代码

```
mysql> SELECT
-> LEFT(a.transdate, 10),
-> c.goodsname,
-> AVG(b.quantity),    -- 平均数量
-> AVG(b.salesvalue)   -- 平均金额
-> FROM
-> demo.transactionhead a
```

```
-> JOIN
-> demo.transactiondetails b ON (a.transactionid = b.transactionid)
-> JOIN
-> demo.goodsmaster c ON (b.itemnumber = c.itemnumber)
-> GROUP BY LEFT(a.transdate,10),c.goodsname
-> ORDER BY LEFT(a.transdate,10),c.goodsname;
+----------------------+-----------+-----------------+-------------------+
| LEFT(a.transdate, 10) | goodsname | AVG(b.quantity) | AVG(b.salesvalue) |
+----------------------+-----------+-----------------+-------------------+
| 2020-12-01 | 书 | 2.0000000 | 178.000000 |
| 2020-12-01 | 笔 | 5.0000000 | 25.000000 |
| 2020-12-02 | 书 | 2.0000000 | 178.000000 |
| 2020-12-02 | 笔 | 8.0000000 | 40.000000 |
+----------------------+-----------+-----------------+-------------------+
4 rows in set (0.00 sec)
```

2.MAX（）和 MIN（）

MAX() 表示获取指定字段在分组中的最大值，MIN() 表示获取指定字段在分组中的最小值。它们的实现原理差不多，下面我就重点讲一下 MAX()，知道了它的用法，MIN() 也就很好理解了。

我们还是来看具体的例子。假如用户要求计算每天里的一次销售的最大数量和最大金额，就可以用下面的代码，得到我们需要的结果：

复制代码

```
mysql> SELECT
-> LEFT(a.transdate, 10),
-> MAX(b.quantity),     -- 数量最大值
-> MAX(b.salesvalue)    -- 金额最大值
-> FROM
-> demo.transactionhead a
-> JOIN
-> demo.transactiondetails b ON (a.transactionid = b.transactionid)
-> JOIN
-> demo.goodsmaster c ON (b.itemnumber = c.itemnumber)
-> GROUP BY LEFT(a.transdate,10)
-> ORDER BY LEFT(a.transdate,10);
+----------------------+-----------------+-------------------+
| LEFT(a.transdate, 10) | MAX(b.quantity) | MAX(b.salesvalue) |
+----------------------+-----------------+-------------------+
| 2020-12-01 | 5.000 | 178.00 |
| 2020-12-02 | 10.000 | 267.00 |
+----------------------+-----------------+-------------------+
2 rows in set (0.00 sec)
```

代码很简单，你一看就明白了。但是，这里有个问题你要注意：千万不要以为MAX（b.quantity）和 MAX（b.salesvalue）算出的结果一定是同一条记录的数据。实际上，MySQL 是分别计算的。下面我们就来分析一下刚刚的查询。

查询中用到 3 个相互关联的表：销售流水明细表、销售流水单头表和商品信息表。这 3 个表连接完成之后，MySQL 进行了分组。我用图示的办法给你展示出来：

第一组

a(demo.transactiondetails)			b(demo.transactionhead)			c(demo.goodsmaster)		
itemnumber	quantity	salesvalue	transactionid	transactionno	transdate	goodsname	unit	salesprice
1	2	178	1	0120201201000001	2020-12-01 14:25:56	书	本	89
2	5	25	1	0120201201000001	2020-12-01 14:25:56	笔	包	5

第二组

a(demo.transactiondetails)			b(demo.transactionhead)			c(demo.goodsmaster)		
itemnumber	quantity	salesvalue	transactionid	transactionno	transdate	goodsname	unit	salesprice
1	3	267	2	0120201202000001	2020-12-02 10:50:50	书	本	89
2	6	30	2	0120201202000001	2020-12-02 10:50:50	笔	包	5
1	1	89	3	0120201202000002	2020-12-02 12:10:05	书	本	89
2	10	50	3	0120201202000002	2020-12-02 12:10:05	笔	包	5

在第一组中，最大数量出现在第 2 条记录，是 5；最大金额出现在第 1 条记录，是 178。同样道理，在第二组中，最大数量出现在第 4 条记录，是 10；最大金额则出现在第 1 条记录，是 267。

所以，MAX（字段）这个函数返回分组集中最大的那个值。如果你要查询 MAX（字段 1）和 MAX（字段 2），而它们是相互独立、分别计算的，你千万不要想当然地认为结果在同一条记录上。那样的话，你就掉坑里了。

COUNT（）

通过 **COUNT（）**，我们可以了解数据集的大小，这对系统优化十分重要。

举个小例子，在项目实施的过程中，我们遇到了这么一个问题：由于用户的销售数据很多，而且每天都在增长，因此，在做销售查询的时候，经常会遇到卡顿的问题。这是因为，查询的数据量太大了，导致系统不得不花很多时间来处理数据，并给数据集分配资源，比如内存什么的。

怎么解决卡顿的问题呢？我们想到了一个分页的策略。

所谓的分页策略，其实就是，不把查询的结果一次性全部返回给客户端，而是根据用户电脑屏幕的大小，计算一屏可以显示的记录数，每次只返回用户电脑屏幕可以显示的数据集。接着，再通过翻页、跳转等功能按钮，实现查询目标的精准锁定。这样一来，每次查询的数据量较少，也就大大提高了系统响应速度。

这个策略能够实现的一个关键，就是要**计算出符合条件的记录一共有多少条**，之后才能计算出一共有几页、能不能翻页或跳转。

要计算记录数，就要用到 COUNT() 函数了。这个函数有两种情况。

COUNT（*）：统计一共有多少条记录；

COUNT（字段）：统计有多少个不为空的字段值。

1.COUNT(*)

如果 COUNT（*）与 GROUP BY 一起使用，就表示统计分组内有多少条数据。它也可以单独使用，这就相当于数据集全体是一个分组，统计全部数据集的记录数。

我举个小例子，假设我有个销售流水明细表如下：

复制代码

```
mysql> SELECT *
    -> FROM demo.transactiondetails;
+---------------+------------+----------+-------+------------+
| transactionid | itemnumber | quantity | price | salesvalue |
```

```
+---------------+------------+----------+-------+------------+
|             1 |          1 |    2.000 | 89.00 |     178.00 |
|             1 |          2 |    5.000 |  5.00 |      25.00 |
|             2 |          1 |    3.000 | 89.00 |     267.00 |
|             2 |          2 |    6.000 |  5.00 |      30.00 |
|             3 |          1 |    1.000 | 89.00 |      89.00 |
|             3 |          2 |   10.000 |  5.00 |      50.00 |
+---------------+------------+----------+-------+------------+
6 rows in set (0.00 sec)
```

如果我们一屏可以显示 30 行，需要多少页才能显示完这个表的全部数据呢?

复制代码

```
mysql> SELECT COUNT(*)
-> FROM demo.transactiondetails;
+----------+
| COUNT(*) |
+----------+
| 6 |
+----------+
1 row in set (0.03 sec)
```

我们这里只有 6 条数据，一屏就可以显示了，所以一共 1 页。

那么，如果超市经营者想知道，每天、每种商品都有几次销售，我们就需要按天、按商品名称，进行分组查询:

复制代码

```
mysql> SELECT
-> LEFT(a.transdate, 10), c.goodsname, COUNT(*) -- 统计销售次数
-> FROM
-> demo.transactionhead a
-> JOIN
-> demo.transactiondetails b ON (a.transactionid = b.transactionid)
-> JOIN
-> demo.goodsmaster c ON (b.itemnumber = c.itemnumber)
-> GROUP BY LEFT(a.transdate, 10) , c.goodsname
-> ORDER BY LEFT(a.transdate, 10) , c.goodsname;
+----------------------+-----------+----------+
| LEFT(a.transdate, 10) | goodsname | COUNT(*) |
+----------------------+-----------+----------+
| 2020-12-01 | 书 | 1 |
| 2020-12-01 | 笔 | 1 |
| 2020-12-02 | 书 | 2 |
```

```
| 2020-12-02 | 笔 | 2 |
+----------------------+----------+----------+
4 rows in set (0.00 sec)
```

运行这段代码，我们就得到了每天、每种商品有几次销售的全部结果。

2.COUNT（字段）

COUNT（字段）用来统计分组内这个字段的值出现了多少次。如果字段值是空，就不统计。

为了说明它们的区别，我举个小例子。假设我们有这样的一个商品信息表，里面包括了商品编号、条码、名称、规格、单位和售价的信息。

复制代码

```
mysql> SELECT *
-> FROM demo.goodsmaster;
+------------+---------+-----------+---------------+------+------------+
| itemnumber | barcode | goodsname | specification | unit | salesprice |
+------------+---------+-----------+---------------+------+------------+
| 1 | 0001 | 书 | 16开 | 本 | 89.00 |
| 2 | 0002 | 笔 | NULL | 支 | 5.00 |
| 3 | 0002 | 笔 | NULL | 支 | 10.00 |
+------------+---------+-----------+---------------+------+------------+
3 rows in set (0.01 sec)
```

如果我们要统计字段“goodsname”出现了多少次，就要用到函数COUNT（goodsname），结果是 3 次：

复制代码

```
mysql> SELECT COUNT(goodsname) -- 统计商品名称字段
-> FROM demo.goodsmaster;
+------------------+
| COUNT(goodsname) |
+------------------+
| 3 |
+------------------+
1 row in set (0.00 sec)
```

如果我们统计字段“specification”，用 COUNT(specification)，结果是 1 次：

复制代码

```
mysql> SELECT COUNT(specification) -- 统计规格字段
-> FROM demo.goodsmaster;
+-----------------------+
| COUNT(specification) |
+-----------------------+
| 1 |
+-----------------------+
1 row in set (0.00 sec)
```

你可能会问，为啥计数字段“goodsname”的结果是 3，计数字段“specification”却只有 1 呢？其实，这里的原因就是，3 条记录里面的字段“goodsname”没有空值，因此被统计了 3 次；而字段“specification”有 2 个空值，因此只统计了 1 次。

理解了这一点，你就可以利用计数函数对某个字段计数时，不统计空值的特点，对表中字段的非空值进行计数了。

总结

今天，我们学习了聚合函数 SUM（）、AVG（）、MAX（）、MIN（）和 COUNT（）。我们在对分组数据进行统计的时候，可以用这些函数来对分组数据求和、求平均值、最大值、最小值，以及统计分组内的记录数，或者分组内字段的值不为空的次数。

这些函数，为我们对数据库中的数据进行统计和计算提供了方便。因为计算直接在数据库中执行，比在应用层面完成相同的工作，效率高很多。

最后，我还想多说一句，不知道你注意到没有，这节课我还提到了 LEFT 和 ORDER BY。其实，聚合函数可以和其他关键字、函数一起使用，这样会拓展它的使用场景，让原本复杂的计算变简单。所以，我建议你不仅要认真学习这节课的聚合函数，还要掌握 MySQL 的各种关键字的功能和用法，并且根据实际工作的需要，尝试把它们组合在一起使用，这样就能利用好数据库的强大功能，更好地满足用户的需求。

思考题

如果用户想要查询一下，在商品信息表中，到底是哪种商品的商品名称有重复，分别重复了几次，该如何查询呢？

欢迎在留言区写下你的思考和答案，我们一起交流讨论。如果你觉得今天的内容对你有所帮助，也欢迎你分享你的朋友或同事，我们下节课见。

提建议

上一篇 07 | 条件语句：WHERE 与 HAVING有什么不同?

下一篇 09 | 时间函数：时间类数据，MySQL是怎么处理的?

精选留言 (8)

写留言

右耳朵猫咪
2021-03-25

像sum()、avg()这样的函数都是统计非空的记录；文章中对group by后的字段做了函数处理，会降低性能，实际开发中不会这么搞。

展开

3 2

Michael
2021-03-25

我一直有个疑惑就是当我需要做对某些字段做group by来分组 我可能还需要其他不作为group by的字段 但是根据SQL标准 出现在select后面的字段要么是group by里的字段 要么是聚合函数 那么这种情况应该怎么处理呢？

展开

2 1

上邪忘川
2021-03-31

老师你好，每次有新表的时候，可否贴一下建表语句和插入的数据

Harry
2021-03-29

如何像老师一样分析出这些个聚合函数分组统计的步骤？

展开

Harry
2021-03-29

“如果是按照以字段作为变量的某个函数进行分组的，你要知道这个函数的返回值是什么，返回值又是如何影响分组的等”。

——这一段怎么理解呢？

展开

Harry
2021-03-29

聚合函数 SUM 的例子中，需求是统计某个门店，每天、每个单品的销售情况，包括销售数量和销售金额等，那么分组条件是否缺少了门店信息？


Kansei
2021-03-26

老师能说说count(*)，count(1)，count(列名)，这3者的区别吗？


1


lesserror
2021-03-25

之前使用MySQL函数用来截取字符串使用的是SUBSTR函数，今天这节课新学了一个LEFT函数。

不过就像右耳朵猫咪同学评论说的那样，对查询字段使用函数不恰当的话，有可能导致查询效率变慢。我的理解是索引失效。...

展开