08丨什么是SQL的聚集函数，如何利用它们汇总表的数据?

我们上节课讲到了 SQL 函数，包括算术函数、字符串函数、日期函数和转换函数。实际上 SQL 函数还有一种，叫做聚集函数，它是对一组数据进行汇总的函数，输入的是一组数据的集合，输出的是单个值。通常我们可以利用聚集函数汇总表的数据，如果稍微复杂一些，我们还需要先对数据做筛选，然后再进行聚集，比如先按照某个条件进行分组，对分组条件进行筛选，然后得到筛选后的分组的汇总信息。

有关今天的内容，你重点需要掌握以下几个方面：

1. 聚集函数都有哪些，能否在一条 SELECT 语句中使用多个聚集函数；
2. 如何对数据进行分组，并进行聚集统计；
3. 如何使用 HAVING 过滤分组，HAVING 和 WHERE 的区别是什么。

聚集函数都有哪些

SQL 中的聚集函数一共包括 5 个，可以帮我们求某列的最大值、最小值和平均值等，它们分别是：

函数	说明
COUNT()	总行数
MAX()	最大值
MIN()	最小值
SUM()	求和
AVG()	平均值

这些函数你可能已经接触过，我们再来简单复习一遍。我们继续使用 heros 数据表，对王者荣耀的英雄数据进行聚合。

如果我们想要查询最大生命值大于 6000 的英雄数量。

```
SQL：SELECT COUNT(*) FROM heros WHERE hp_max > 6000
```

运行结果为 41。

如果想要查询最大生命值大于 6000，且有次要定位的英雄数量，需要使用 COUNT 函数。

复制代码

```
SQL：SELECT COUNT(role_assist) FROM heros WHERE hp_max > 6000
```

运行结果是 23。

需要说明的是，有些英雄没有次要定位，即 role_assist 为 NULL，这时`COUNT(role_assist)`会忽略值为 NULL 的数据行，而 COUNT(*) 只是统计数据行数，不管某个字段是否为 NULL。

如果我们想要查询射手（主要定位或者次要定位是射手）的最大生命值的最大值是多少，需要使用 MAX 函数。

复制代码

```
SQL：SELECT MAX(hp_max) FROM heros WHERE role_main = '射手' or role_assist = '射手'
```

运行结果为 6014。

你能看到，上面的例子里，都是在一条 SELECT 语句中使用了一次聚集函数，实际上我们也可以在一条 SELECT 语句中进行多项聚集函数的查询，比如我们想知道射手（主要定位或者次要定位是射手）的英雄数、平均最大生命值、法力最大值的最大值、攻击最大值的最小值，以及这些英雄总的防御最大值等汇总数据。

如果想要知道英雄的数量，我们使用的是 COUNT(*) 函数，求平均值、最大值、最小值，以及总的防御最大值，我们分别使用的是 AVG、MAX、MIN 和 SUM 函数。另外我们还需要对英雄的主要定位和次要定位进行筛选，使用的是`WHERE role_main = '射手' or role_assist = '射手'`。

复制代码

```
SQL: SELECT COUNT(*), AVG(hp_max), MAX(mp_max), MIN(attack_max), SUM(defense_max) FROM h
```

运行结果：

COUNT(*)	AVG(hp_max)	MAX(mp_max)	MIN(attack_max)	SUM(defense_max)
10	5798.5	1784	362	3333

需要说明的是 AVG、MAX、MIN 等聚集函数会自动忽略值为 NULL 的数据行，MAX 和 MIN 函数也可以用于字符串类型数据的统计，如果是英文字母，则按照 A—Z 的顺序排列，越往后，数值越大。如果是汉字则按照全拼拼音进行排列。比如：

复制代码

```
SQL：SELECT MIN(CONVERT(name USING gbk)), MAX(CONVERT(name USING gbk)) FROM heros
```

运行结果：

MIN(CONVERT(name USING gbk))	MAX(CONVERT(name USING gbk))
阿轲	庄周

需要说明的是，我们需要先把 name 字段统一转化为 gbk 类型，使用`CONVERT(name USING gbk)`，然后再使用 MIN 和 MAX 取最小值和最大值。

我们也可以对数据行中不同的取值进行聚集，先用 DISTINCT 函数取不同的数据，然后再使用聚集函数。比如我们想要查询不同的生命最大值的英雄数量是多少。

复制代码

```
SQL: SELECT COUNT(DISTINCT hp_max) FROM heros
```

运行结果为 61。

实际上在 heros 这个数据表中，一共有 69 个英雄数量，生命最大值不一样的英雄数量是 61 个。

假如我们想要统计不同生命最大值英雄的平均生命最大值，保留小数点后两位。首先需要取不同生命最大值，即`DISTINCT hp_max`，然后针对它们取平均值，即`AVG(DISTINCT hp_max)`，最后再针对这个值保留小数点两位，也就是`ROUND(AVG(DISTINCT hp_max), 2)`。

复制代码

```
SQL: SELECT ROUND(AVG(DISTINCT hp_max), 2) FROM heros
```

运行结果为 6653.84。

你能看到，如果我们不使用 DISTINCT 函数，就是对全部数据进行聚集统计。如果使用了 DISTINCT 函数，就可以对数值不同的数据进行聚集。一般我们使用 MAX 和 MIN 函数统计数据行的时候，不需要再额外使用 DISTINCT，因为使用 DISTINCT 和全部数据行进行最大值、最小值的统计结果是相等的。

如何对数据进行分组，并进行聚集统计

我们在做统计的时候，可能需要先对数据按照不同的数值进行分组，然后对这些分好的组进行聚集统计。对数据进行分组，需要使用 GROUP BY 子句。

比如我们想按照英雄的主要定位进行分组，并统计每组的英雄数量。

复制代码

```
SQL: SELECT COUNT(*), role_main FROM heros GROUP BY role_main
```

运行结果（6 条记录）：

COUNT(*)	role_main
10	坦克
18	战士
19	法师
6	辅助
10	射手
6	刺客

如果我们想要对英雄按照次要定位进行分组，并统计每组英雄的数量。

复制代码

```
1 SELECT COUNT(*), role_assist FROM heros GROUP BY role_assist
```

运行结果：（6 条记录）

COUNT(*)	role_assist
6	战士
10	坦克
5	辅助
40	NULL
2	法师
6	刺客

你能看出如果字段为 NULL，也会被列为一个分组。在这个查询统计中，次要定位为 NULL，即只有一个主要定位的英雄是 40 个。

我们也可以使用多个字段进行分组，这就相当于把这些字段可能出现的所有的取值情况都进行分组。比如，我们想要按照英雄的主要定位、次要定位进行分组，查看这些英雄的数量，并按照这些分组的英雄数量从高到低进行排序。

复制代码

```
SELECT COUNT(*) as num, role_main, role_assist FROM heros GROUP BY role_main, role_assis
```

运行结果：（19 条记录）

num	role_main	role_assist
12	法师	
9	射手	
8	战士	
6	战士	坦克
5	坦克	
3	刺客	
3	坦克	辅助
3	辅助	
3	战士	刺客
3	刺客	战士
2	法师	辅助
2	法师	刺客
2	法师	坦克
2	辅助	坦克
2	坦克	战士
1	射手	刺客
1	辅助	法师
1	法师	战士
1	战士	法师

如何使用 HAVING 过滤分组，它与 WHERE 的区别是什么?

当我们创建出很多分组的时候，有时候就需要对分组进行过滤。你可能首先会想到 WHERE 子句，实际上过滤分组我们使用的是 HAVING。HAVING 的作用和 WHERE 一样，都是起到过滤的作用，只不过 WHERE 是用于数据行，而 HAVING 则作用于分组。

比如我们想要按照英雄的主要定位、次要定位进行分组，并且筛选分组中英雄数量大于 5 的组，最后按照分组中的英雄数量从高到低进行排序。

首先我们需要获取的是英雄的数量、主要定位和次要定位，即`SELECT COUNT(*) as num, role_main, role_assist`。然后按照英雄的主要定位和次要定位进行分组，即`GROUP BY role_main, role_assist`，同时我们要对分组中的英雄数量进行筛选，选择大于 5 的分组，即`HAVING num > 5`，然后按照英雄数量从高到低进行排序，即`ORDER BY num DESC`。

复制代码

```
SQL: SELECT COUNT(*) as num, role_main, role_assist FROM heros GROUP BY role_main, role_
```

运行结果：（4 条记录）

num	role_main	role_assist
12	法师	
9	射手	
8	战士	
6	战士	坦克

你能看到还是上面这个分组，只不过我们按照数量进行了过滤，筛选了数量大于 5 的分组进行输出。如果把 HAVING 替换成了 WHERE，SQL 则会报错。对于分组的筛选，我们一定要用 HAVING，而不是 WHERE。另外你需要知道的是，HAVING 支持所有 WHERE 的操作，因此所有需要 WHERE 子句实现的功能，你都可以使用 HAVING 对分组进行筛选。

我们再来看个例子，通过这个例子查看一下 WHERE 和 HAVING 进行条件过滤的区别。筛选最大生命值大于 6000 的英雄，按照主要定位、次要定位进行分组，并且显示分组中英雄数量大于 5 的分组，按照数量从高到低进行排序。

复制代码

```
SQL: SELECT COUNT(*) as num, role_main, role_assist FROM heros WHERE hp_max > 6000 GROUI
```

运行结果：（2 条记录）

num	role_main	role_assist
8	战士	
6	战士	坦克

你能看到，还是针对上一个例子的查询，只是我们先增加了一个过滤条件，即筛选最大生命值大于 6000 的英雄。这里我们就需要先使用 WHERE 子句对最大生命值大于 6000 的英雄进行条件过滤，然后再使用 GROUP BY 进行分组，使用 HAVING 进行分组的条件判断，然后使用 ORDER BY 进行排序。

总结

今天我对 SQL 的聚集函数进行了讲解。通常我们还会对数据先进行分组，然后再使用聚集函数统计不同组的数据概况，比如数据行数、平均值、最大值、最小值以及求和等。我们也可以使用 HAVING 对分组进行过滤，然后通过 ORDER BY 按照某个字段的顺序进行排序输出。有时候你能看到在一条 SELECT 语句中，可能会包括多个子句，用 WHERE 进行数据量的过滤，用 GROUP BY 进行分组，用 HAVING 进行分组过滤，用 ORDER BY 进行排序……

你要记住，在 SELECT 查询中，关键字的顺序是不能颠倒的，它们的顺序是：

复制代码

```
SELECT ... FROM ... WHERE ... GROUP BY ... HAVING ... ORDER BY ...
```

另外需要注意的是，使用 GROUP BY 进行分组，如果想让输出的结果有序，可以在 GROUP BY 后使用 ORDER BY。因为 GROUP BY 只起到了分组的作用，排序还是需要通过 ORDER BY 来完成。

我今天对 SQL 的聚集函数以及 SQL 查询中的关键字顺序进行了讲解，但你还是需要通过训练加深理解，基于 heros 数据表，请你写出下面 2 个 SQL 查询语句：

1. 筛选最大生命值大于 6000 的英雄，按照主要定位进行分组，选择分组英雄数量大于 5 的分组，按照分组英雄数从高到低进行排序，并显示每个分组的英雄数量、主要定位和平均最大生命值。
2. 筛选最大生命值与最大法力值之和大于 7000 的英雄，按照攻击范围来进行分组，显示分组的英雄数量，以及分组英雄的最大生命值与法力值之和的平均值、最大值和最小值，并按照分组英雄数从高到低进行排序，其中聚集函数的结果包括小数点后两位。