## 列的各项值

SELECT MAX(article) AS article FROM shop; ///最大值

SELECT MIN(article) AS article FROM shop; ///最小值

SELECT AVG(article) AS article FROM shop; ///平均值

SELECT SUM(article) AS article FROM shop; ///和

如何显示客户表中的第四个最高（工资）记录：

mysql> SELECT sal FROM `emp` order by sal desc limit 3,1

## 持有某一列的最大值的行

子查询：

SELECT article, dealer, price

FROM shop

WHERE price=(SELECT MAX(price) FROM shop);

使用LEFT JOIN：

SELECT s1.article, s1.dealer, s1.price

FROM shop s1

LEFT JOIN shop s2 ON s1.price < s2.price

WHERE s2.article IS NULL;

按照价格降序对所有行进行排序，并使用MySQL特定的LIMIT子句只获取第一行：

SELECT article, dealer, price

FROM shop

ORDER BY price DESC

LIMIT 1;

## 每个组的最大列数

SELECT article, MAX(price) AS price

FROM shop

GROUP BY article;

## 4．行数保持某一列的最大值

SELECT article, dealer, price

FROM shop s1

WHERE price=(SELECT MAX(s2.price)

FROM shop s2

WHERE s1.article = s2.article);

前面的示例使用相关的子查询，效率可能很低。解决问题的其他可能性是在FROM子句中 使用不相关的子查询LEFT JOIN。

不相关的子查询：

SELECT s1.article, dealer, s1.price

FROM shop s1

JOIN (

SELECT article, MAX(price) AS price

FROM shop

GROUP BY article) AS s2

ON s1.article = s2.article AND s1.price = s2.price;

LEFT JOIN：

SELECT s1.article, s1.dealer, s1.price

FROM shop s1

LEFT JOIN shop s2 ON s1.article = s2.article AND s1.price< s2.price

WHERE s2.article IS NULL;

## 使用用户定义的变量

您可以使用MySQL用户变量来记住结果，而不必将它们存储在客户端的临时变量中

mysql> SELECT @min\_price:=MIN(price),@max\_price:=MAX(price) FROM shop;

mysql> SELECT \* FROM shop WHERE price=@min\_price OR price=@max\_price;

@\*\*\* ---用户自定义变量

## 使用外键（REFERENCES *tbl\_name*(*col\_name*)）

外键约束不仅仅需要连接两个表。对于除此之外的存储引擎 InnoDB，当定义一个列来使用一个 没有实际效果的子句时，可能只是作为一个备忘录或注释给你，你当前定义的列是要引用一列另一张桌子。在使用这种语法时，实现这一点非常重要： REFERENCES ***tbl\_name***(***col\_name***)

CREATE TABLE person (

id SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

name CHAR(60) NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE shirt (

id SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

style ENUM('t-shirt', 'polo', 'dress') NOT NULL,

color ENUM('red', 'blue', 'orange', 'white', 'black') NOT NULL,

owner SMALLINT UNSIGNED NOT NULL REFERENCES person(id),

PRIMARY KEY (id)

);

INSERT INTO person VALUES (NULL, 'Antonio Paz');

SELECT @last := LAST\_INSERT\_ID();

INSERT INTO shirt VALUES

(NULL, 'polo', 'blue', @last),

(NULL, 'dress', 'white', @last),

(NULL, 't-shirt', 'blue', @last);

INSERT INTO person VALUES (NULL, 'Lilliana Angelovska');

SELECT @last := LAST\_INSERT\_ID();

INSERT INTO shirt VALUES

(NULL, 'dress', 'orange', @last),

(NULL, 'polo', 'red', @last),

(NULL, 'dress', 'blue', @last),

(NULL, 't-shirt', 'white', @last);

SELECT \* FROM person;

+----+---------------------+

| id | name |

+----+---------------------+

| 1 | Antonio Paz |

| 2 | Lilliana Angelovska |

+----+---------------------+

SELECT \* FROM shirt;

+----+---------+--------+-------+

| id | style | color | owner |

+----+---------+--------+-------+

| 1 | polo | blue | 1 |

| 2 | dress | white | 1 |

| 3 | t-shirt | blue | 1 |

| 4 | dress | orange | 2 |

| 5 | polo | red | 2 |

| 6 | dress | blue | 2 |

| 7 | t-shirt | white | 2 |

+----+---------+--------+-------+

SELECT s.\* FROM person p INNER JOIN shirt s

ON s.owner = p.id

WHERE p.name LIKE 'Lilliana%'

AND s.color <> 'white';

+----+-------+--------+-------+

| id | style | color | owner |

+----+-------+--------+-------+

| 4 | dress | orange | 2 |

| 5 | polo | red | 2 |

| 6 | dress | blue | 2 |

+----+-------+--------+-------+

当以这种方式使用时，该REFERENCES 子句不显示在[SHOW CREATE TABLE](https://dev.mysql.com/doc/refman/5.7/en/show-create-table.html)or 的输出中 [DESCRIBE](https://dev.mysql.com/doc/refman/5.7/en/describe.html)：因为reference只是作为注释或“提醒”

## 计算每天的访问量

计算用户每月访问网页的天数（如果一天内有很多用户访问留下时间记录计算时只能算一天，也就是要去重复）

CREATE TABLE t1 (year YEAR(4), month INT(2) UNSIGNED ZEROFILL,

day INT(2) UNSIGNED ZEROFILL);

INSERT INTO t1 VALUES(2000,1,1),(2000,1,20),(2000,1,30),(2000,2,2),

(2000,2,23),(2000,2,23);

去重方法一：通过多次group by进行去重：

select year, month, count(0)

from (select year, month, day,count(0) from t1 group by year, month, day) as tmp

    group by year, month;

去重方法二：通过位组函数去重：

SELECT year,month,BIT\_COUNT(BIT\_OR(1<<day)) AS days FROM t1

GROUP BY year,month;

去重方法三：通过distinct 去重：

select year,month,count(distinct day) as day from t1 group by  
    -> year,month;

1、BIT\_COUNT( expr )：返回 expr 的二进制表达式中”1“的个数。

     例如：29 = 11101 则：BIT\_COUNT（29）= 4；

2、BIT\_OR( expr )：返回 expr 中所有比特的bitwise OR。计算执行的精确度为64比特(BIGINT) 。  
　  例如：上面例子中，2000年02月中有一条2号的记录两条23号的记录，所以"1<<day"表示出来就是 “1＜＜2”和“1＜＜23”，得到二进制数 100 和 100000000000000000000000 。然后再OR运算。即 100 OR 10000000000000000000000 OR 10000000000000000000000 = 100000000000000000000100；这样再用BIT\_COUNT处理得出的值就是2，自动去除了重复的日期。

“1<<2”意思实在1的后面添加两个0 “1<<23”在1的后面添加23个0

## 9. 使用auto-increment

为该列自动生成序号，还可以显式地将0赋给列以生成序列号 。如果该列已被声明NOT NULL，那么也可以分配 NULL给该列来生成序列号。在AUTO\_INCREMENT列中插入任何其他值时 ，该列将设置为该值，并重置序列，以便下一个自动生成的值从最大列值开始依次跟随。

要以AUTO\_INCREMENT1以外的值开始，请使用[CREATE TABLE](https://dev.mysql.com/doc/refman/5.7/en/create-table.html)or来设置该值[ALTER TABLE](https://dev.mysql.com/doc/refman/5.7/en/alter-table.html)，如下所示：

mysql> ALTER TABLE tbl AUTO\_INCREMENT = 100;

如果该AUTO\_INCREMENT列是多个索引的一部分，则MySQL使用以AUTO\_INCREMENT 列开头的索引（如果有的话）生成序列值。例如，如果该 animals表包含索引PRIMARY KEY (grp, id)和INDEX (id)时，MySQL将忽略PRIMARY KEY用于产生序列的值。结果，表格将包含单个序列，而不是每个grp值的序列。

CREATE TABLE animals (

grp ENUM('fish','mammal','bird') NOT NULL,

id MEDIUMINT NOT NULL AUTO\_INCREMENT,

name CHAR(30) NOT NULL,

PRIMARY KEY (grp,id)

) ENGINE=MyISAM;

INSERT INTO animals (grp,name) VALUES

('mammal','dog'),('mammal','cat'),

('bird','penguin'),('fish','lax'),('mammal','whale'),

('bird','ostrich');

SELECT \* FROM animals ORDER BY grp,id;

+--------+----+---------+

| grp | id | name |

+--------+----+---------+

| fish | 1 | lax |

| mammal | 1 | dog |

| mammal | 2 | cat |

| mammal | 3 | whale |

| bird | 1 | penguin |

| bird | 2 | ostrich |

+--------+----+---------+