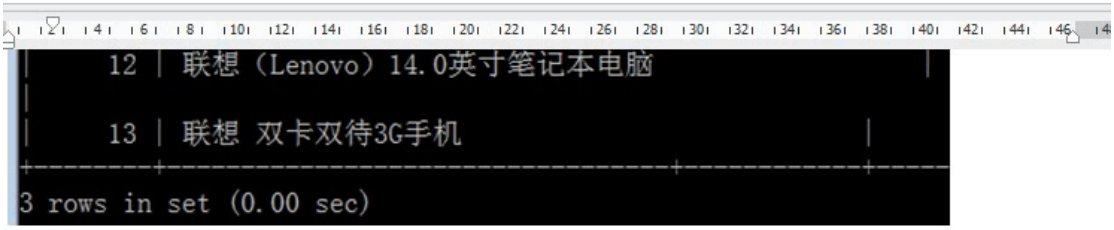
select [all | distinct] 字段或表达式列表 [from 子句] [where 子句] [group by 子句] [having 子句] [order by 子句] [limit 子句];



常见示例及含义：

name like '%罗%': 表示 nam 中“罗”这一个字的所有数据行；
name like '%罗%': 表示 nam 中“罗”这一个字的所有数据行；
name like '%罗': 表示 nam 中以“罗”结尾的所有数据行；
name like '罗_': 表示 nam 中以“罗”开头并只有 2 个字符的所有数据行；

group by 子句：分组

形式：

group by 字段 1 【desc|asc】， 字段 2 【desc|asc】， .....

说明：

1，分组是对“前述”已经找出的数据（即 where 已经筛选过了）进行某种指定标准（依据）的分组。
2，同时，该分组结果，可以同时指定其“排序方式”：desc（倒序），asc（顺序）；
3，通常，分组就一个字段（依据），2 个以上很少。

什么叫做分组？

分组：就是将多行数据，以某种标准（就是指定的字段）来进行“分类”存放。

特别注意：

分组之后的结果，一定要理解为：只有一个一个组了

则，结果是：在 select 语句中的“输出（取出）”部分，只应该出现“组的信息”：

select 组信息 1， 组信息 2， ..... from 数据源 group by 字段；

select [all | distinct] 字段或表达式列表 [from 子句] [where 子句] [group by 子句] [having 子句] [order by 子句] [limit 子句];

继续举例：

```
select pinpai, count(*) as 数量, max(price) as 最高价,min(price) as 最低价,
avg(price) as 平均价, sum(price) as 总价 from product group by pinpai
```

结果为：

pinpai	数量	最高价	最低价	平均价	总价
康佳	1	1999.00	1999.00	1999.000000	1999.00
惠普	1	1169.00	1169.00	1169.000000	1169.00
海信	1	4199.00	4199.00	4199.000000	4199.00
索尼	3	11499.00	3238.00	7245.333333	21736.00
联想	3	5499.00	988.00	3162.000000	9486.00

则这个结果的含义就非常明显：
某个品牌有多少件商品，及该品牌的最高价，最低价，平均价和总价，全都出来了。

having 子句

一句话：

having 的作用跟 where 完全一样，但其只是对“分组的结果数据”进行筛选；

即：

where 对原始数据行进行筛选；

having 对分组之后的数据行进行筛选；

示例 1：找出平均价大于 5000 的品牌信息：

```
select pinpai, count(*) as 数量, max(price) as 最高价,min(price) as 最低价,
avg(price) as 平均价, sum(price) as 总价 from product group by pinpai
having 平均价 > 5000
```

结果为：

pinpai	数量	最高价	最低价	平均价	总价
索尼	3	11499.00	3238.00	7245.333333	21736.00

示例 2：找出商品数超过 2 个的品牌信息：

```
1 select pinpai, count(*) as 数量, max(price) as 最高价,min(price) as 最低价,
2 avg(price) as 平均价, sum(price) as 总价 from product group by pinpai
3 having 数量 > 2
```

结果为：

pinpai	数量	最高价	最低价	平均价	总价
索尼	3	11499.00	3238.00	7245.333333	21736.00
联想	3	5499.00	988.00	3162.000000	9486.00

order by 子句

它用于将前面“取得”的数据以设定的标准（字段）来进行排序以输出结果。

形式：

order by 字段 1 【asc|desc】， 字段 2 【asc|desc】，.....

说明：

1，对前面的结果数据以指定的一个或多个字段排序；
2，排序可以谁定正序（asc，默认值）或倒序（desc）；
3，多个字段的排序，都是在前一个字段排序基础上，如果还有“相等值”，就继续以后续字段排序；

select [all | distinct] 字段或表达式列表 [from 子句] [where 子句] [group by 子句] [having 子句] [order by 子句] [limit 子句]；

它用于将前面“取得”的数据以设定的标准（字段）来进行排序以输出结果。

形式：

order by 字段 1 【asc|desc】， 字段 2 【asc|desc】，.....

说明：

1，对前面的结果数据以指定的一个或多个字段排序；
2，排序可以谁定正序（asc，默认值）或倒序（desc）；
3，多个字段的排序，都是在前一个字段排序基础上，如果还有“相等值”，才继续以后续字段排序；

举例：

```
SELECT * FROM `product` order by protype_id
```

limit 子句

含义：

它用于将“前述取得的数据”，按指定的行取出来：从第几行开始取出多少行；

形式：

limit 起始行号， 要取出的行数；

说明：

1，起始行号都是从 0 开始算起的；
2，起始行号跟数据中的任何一个字段（比如 id）没有关系；
3，要取出的行数也是一个数字，自然应该是大于 0 的；
4，有一个简略形式：limit 行数； 表示直接从第 0 行开始取出指定的行数，它相当于 limit 0, 行数;

对整个 select 语句的一些总结

1，虽然在形式上，select 的很多子句都是可以省略的，但他们的顺序（如果出现），就不能打乱的：必须仍然按照给出的顺序写出；

2，where 子句依赖于 from 子句：即没有 from，就不能有 where；

3，having 子句依赖于 groupby 子句：即没有 groupby，就不能有 having；

4，select 中的“字段”也是依赖于ｆｒｏｍ子句；

5，上述各子句的“内部执行过程”，基本上也都是按照该顺序进行的：

即从 from 的数据源中获得“所有数据”，然后使用 where 对这些数据进行“筛选”，之后再使用 groupby 子句对筛选出来的数据进行“分组”，接下来才可以使用 having 对这些分组的数据进行筛选，然后才可以 orderby 和 limit。