04 MySQL之单表查询

MySQL之单表查询

阅读目录

一 单表查询的语法

```
#查询数据的本质：mysql会到你本地的硬盘上找到对应的文件，然后打开文件，按照你的查询条件来找出你需要的数据。下面是完整的
select * from，这个select * 指的是要查询所有字段的数据。
SELECT distinct 字段1,字段2... FROM 库名.表名 #from后面是说从库的某个表中去找数据，mysql会去找到这个库对应的文件夹下去
                WHERE 条件       #从表中找符合条件的数据记录，where后面跟的是你的查询条件
                GROUP BY field（字段）    #分组
                HAVING 筛选      #过滤，过滤之后执行select后面的字段筛选，就是说我要确定一下需要哪个字段的数据，你查询的字段
                ORDER BY field（字段）    #将结果按照后面的字段进行排序
                LIMIT 限制条数    #将最后的结果加一个限制条数，就是说我要过滤或者说限制查询出来的数据记录的条数关于上面这些内
```

二 关键字的执行优先级(重点)

```
重点中的重点：关键字的执行优先级
from
where
group by
having
select
distinct
order by
limit
```

1.**找到表**:from

2.**拿着**where**指定的约束条件，去文件/表中取出一条条记录**

3.**将取出的一条条记录进行分组**group by，**如果没有**group by，**则整体作为一组**

4.**将分组的结果进行**having**过滤**

5.**执行**select

6.**去重**

7.**将结果按条件排序：**order by

8.**限制结果的显示条数**

详细见：https://www.cnblogs.com/clschao/articles/9995517.html

三 简单查询

```
#我们来创建一个员工表，然后对员工表进行一个简单的查询，来看一下效果，下面是员工表的字段company.employee
    员工id      id              int
    姓名        emp_name        varchar
    性别        sex             enum
    年龄        age             int
    入职日期     hire_date       date
    岗位        post            varchar
    职位描述     post_comment    varchar
    薪水        salary          double
    办公室       office          int
    部门编号     depart_id       int



#创建表
create table employee(
    id int not null unique auto_increment,
    name varchar(20) not null,
    sex enum('male','female') not null default 'male', #大部分是男的
    age int(3) unsigned not null default 28,
    hire_date date not null,
    post varchar(50),
    post_comment varchar(100),
    salary double(15,2),
    office int, #一个部门一个屋子
    depart_id int
);


#查看表结构
mysql> desc employee;
+-------------+----------------------+------+-----+---------+----------------+
| Field       | Type                 | Null | Key | Default | Extra          |
+-------------+----------------------+------+-----+---------+----------------+
| id          | int(11)              | NO   | PRI | NULL    | auto_increment |
| name        | varchar(20)          | NO   |     | NULL    |                |
| sex         | enum('male','female') | NO   |     | male    |                |
| age         | int(3) unsigned      | NO   |     | 28      |                |
| hire_date   | date                 | NO   |     | NULL    |                |
| post        | varchar(50)          | YES  |     | NULL    |                |
| post_comment | varchar(100)        | YES  |     | NULL    |                |
| salary      | double(15,2)         | YES  |     | NULL    |                |
| office      | int(11)              | YES  |     | NULL    |                |
| depart_id   | int(11)              | YES  |     | NULL    |                |
+-------------+----------------------+------+-----+---------+----------------+

#插入记录
#三个部门：教学，销售，运营
insert into employee(name,sex,age,hire_date,post,salary,office,depart_id) values
('egon','male',18,'20170301','老男孩驻沙河办事处外交大使',7300.33,401,1), #以下是教学部，全都是老师
('alex','male',78,'20150302','teacher',1000000.31,401,1),
('wupeiqi','male',81,'20130305','teacher',8300,401,1),
```

```
('yuanhao','male',73,'20140701','teacher',3500,401,1),
('liwenzhou','male',28,'20121101','teacher',2100,401,1),
('jingliyang','female',18,'20110211','teacher',9000,401,1),
('jinxin','male',18,'19000301','teacher',30000,401,1),
('成龙','male',48,'20101111','teacher',10000,401,1),

('歪歪','female',48,'20150311','sale',3000.13,402,2),#以下是销售部门
('丫丫','female',38,'20101101','sale',2000.35,402,2),
('丁丁','female',18,'20110312','sale',1000.37,402,2),
('星星','female',18,'20160513','sale',3000.29,402,2),
('格格','female',28,'20170127','sale',4000.33,402,2),

('张野','male',28,'20160311','operation',10000.13,403,3), #以下是运营部门
('程咬金','male',18,'19970312','operation',20000,403,3),
('程咬银','female',18,'20130311','operation',19000,403,3),
('程咬铜','male',18,'20150411','operation',18000,403,3),
('程咬铁','female',18,'20140512','operation',17000,403,3)
;

#ps：如果在windows系统中，插入中文字符，select的结果为空白，可以将所有字符编码统一设置成gbk
```

查询操作：

```
简单查询
    SELECT id,name,sex,age,hire_date,post,post_comment,salary,office,depart_id
    FROM employee;

    SELECT * FROM employee; #不推荐用* ，查询的时候*的效率低，至于为什么低，后面会讲到，先知道一下就行了

    SELECT name,salary FROM employee;

#避免重复DISTINCT   SELECT post FROM employee;#直接这样查询我们会看到很多重复的内容，我只想看一下有哪些职位，那么多
        看一下下面这两句的效果就明白了：注意一点，使用distinct对记录进行去重的时候，distinct必须写在所有查询字段的前面，不
#通过四则运算查询
    SELECT name, salary*12 FROM employee; #查询每个人的年薪，月薪我们有记录，查年薪呢？简单的乘以12就可以了，from 库
    #你会发现，结果是出来了，但是我们的那个薪资的字段名变成了salary*12，是因为我们通过查询语句查询出来的也是一张表，但是
    SELECT name, salary*12 Annual_salary FROM employee;
      #除了乘法以外，加减乘除都是可以的
#自定义显示格式，自己规定查询结果的显示格式
  CONCAT() 函数用于连接字符串
  SELECT CONCAT('姓名: ',name,'  年薪: ', salary*12)  AS Annual_salary  #我想让name这个字段显示的字段名称是中文的姓名，i
  FROM employee;#看结果：通过结果你可以看出，这个concat就是帮我们做字符串拼接的，并且拼接之后的结果，都在一个叫做An
```

```
        +--------------------------------------+
        | Annual_salary |
        +--------------------------------------+
        | 姓名: egon 年薪: 87603.96 |
        | 姓名: alex 年薪: 12000003.72 |
        | 姓名: wupeiqi 年薪: 99600.00 |
        | 姓名: yuanhao 年薪: 42000.00 |

        .....

    +--------------------------------------+
```

```
  SELECT CONCAT('姓名: ',name,'  年薪: ', salary*12)  AS Annual_salary,CONCAT('性别：',sex) from employee;#还可以这样分
  CONCAT_WS() 第一个参数为分隔符来进行字符串拼接
```

```
SELECT CONCAT_WS(':',name,salary*12) AS Annual_salary #通过冒号来将name和salary连接起来
FROM employee;
#上面这个效果我们也可以通过concat来实现：SELECT CONCAT(name,':',salary*12) AS Annual_salary from employee;
结合CASE语句：结合条件来对查询的结果进行一些加工操作
SELECT
    (
        CASE
        WHEN NAME = 'egon' THEN
            NAME
        WHEN NAME = 'alex' THEN
            CONCAT(name,'_BIGSB')
        ELSE
            concat(NAME, 'SB')
        END
    ) as new_name,sex
FROM
    employee;#看结果:
```

```
+--------------+--------+
| new_name | sex |
+--------------+--------+
| egon | male |
| alex_BIGSB | male |
| wupeiqiSB | male |
| yuanhaoSB | male |
| liwenzhouSB | male |
| jingliyangSB | female |
| jinxinSB | male |
| 成龙SB | male |

...

+--------------+
```

简单查询就结束了，我们做一下练习，然后继续学习我们上面列举完整查询语句中的其他内容。

小练习：

```
1 查出所有员工的名字，薪资,格式为
   <名字:egon>    <薪资:3000>
2 查出所有的岗位（去掉重复）
3 查出所有员工名字，以及他们的年薪,年薪的字段名为annual_year
```



```
select concat('<名字:',name,'>    ','<薪资:',salary,'>') from employee;
select distinct depart_id from employee;
select name,salary*12 annual_salary from employee;
```

四 WHERE约束

where语句中可以使用：

之前我们用where 后面跟的语句是不是id=1这种类型的啊，用=号连接的，除了=号外，还能使用其他的，看下面：

1. 比较运算符：> < >= <= <> !=
2. between 80 and 100 值在80到100之间
3. in(80,90,100) 值是80或90或100
4. like 'egon%'
   pattern可以是%或_，
   %表示任意多字符
   _表示一个字符
5. 逻辑运算符：在多个条件直接可以使用逻辑运算符 and or not

```
#1:单条件查询
   SELECT name FROM employee
      WHERE post='sale';  #注意优先级，我们说where的优先级是不是比select要高啊，所以我们的顺序是先找到这个employee表，

#2:多条件查询
   SELECT name,salary FROM employee
      WHERE post='teacher' AND salary>10000;

#3:关键字BETWEEN AND 写的是一个区间
   SELECT name,salary FROM employee
      WHERE salary BETWEEN 10000 AND 20000; #就是salary>=10000 and salary<=20000的数据

   SELECT name,salary FROM employee
      WHERE salary NOT BETWEEN 10000 AND 20000; #加个not，就是不在这个区间内，薪资小于10000的或者薪资大于20000的

#4:关键字IS NULL(判断某个字段是否为NULL不能用等号，需要用IS) 判断null只能用is
   SELECT name,post_comment FROM employee
      WHERE post_comment IS NULL;

   SELECT name,post_comment FROM employee
      WHERE post_comment IS NOT NULL;

   SELECT name,post_comment FROM employee
      WHERE post_comment=''; 注意''是空字符串，不是null，两个是不同的东西，null是啥也没有，''是空的字符串的意思，是一种特
   ps:
      执行
      update employee set post_comment='' where id=2;
      再用上条查看，就会有结果了

#5:关键字IN集合查询
   SELECT name,salary FROM employee
      WHERE salary=3000 OR salary=3500 OR salary=4000 OR salary=9000 ; #这样写是不是太麻烦了，写一大堆的or，下面我

   SELECT name,salary FROM employee
      WHERE salary IN (3000,3500,4000,9000) ;

   SELECT name,salary FROM employee
      WHERE salary NOT IN (3000,3500,4000,9000) ;

#6:关键字LIKE模糊查询，模糊匹配，可以结合通配符来使用
   通配符'%'  #匹配任意所有字符
   SELECT * FROM employee
         WHERE name LIKE 'eg%';

   通配符'_'  #匹配任意一个字符
   SELECT * FROM employee
         WHERE name LIKE 'al__'; #注意我这里写的两个_，用1个的话，匹配不到alex，因为al后面还有两个字符ex。
```

where条件咱们就说完了，这个where条件到底怎么运作的，我们来说一下：我们以select id,name,age from employee where id>7;这个语句来说一下

首先先找到employee表，找到这个表之后，mysql会拿着where后面的约束条件去表里面找符合条件的数据，然后遍历你表中所有的数据，查看一下id是否大于7，逐条的对比，然后只要发现id比7大的，它就会把这一整条记录给select，但是select说我只拿id、name、age这个三个字段里面的数据，然后就打印了这三个字段的数据，然后where继续往下过滤，看看id是不是还有大于7的，然后发现一个符合条件的就给select一个，然后重复这样的事情，直到把数据全部过滤一遍才会结束。这就是where条件的一个工作方式。

然后我们可以来做一些小练习：

```
1. 查看岗位是teacher的员工姓名、年龄
2. 查看岗位是teacher且年龄大于30岁的员工姓名、年龄
3. 查看岗位是teacher且薪资在9000-1000范围内的员工姓名、年龄、薪资
4. 查看岗位描述不为NULL的员工信息
5. 查看岗位是teacher且薪资是10000或9000或30000的员工姓名、年龄、薪资
6. 查看岗位是teacher且薪资不是10000或9000或30000的员工姓名、年龄、薪资
7. 查看岗位是teacher且名字是jin开头的员工姓名、年薪
```

```
select name,age from employee where post = 'teacher';
select name,age from employee where post='teacher' and age > 30;
select name,age,salary from employee where post='teacher' and salary between 9000 and 10000;
select * from employee where post_comment is not null;
select name,age,salary from employee where post='teacher' and salary in (10000,9000,30000);
select name,age,salary from employee where post='teacher' and salary not in (10000,9000,30000);
select name,salary*12 from employee where post='teacher' and name like 'jin%';
```

五 分组查询:GROUP BY

1、 什么是分组？为什么要分组？

```
#1、首先明确一点：分组发生在where之后，即分组是基于where之后得到的记录而进行的

#2、分组指的是：将所有记录按照某个相同字段进行归类，比如针对员工信息表的职位分组，或者按照性别进行分组等

#3、为何要分组呢？是因为我们有时候会需要以组为单位来统计一些数据或者进行一些计算的，对不对，比方说下面的几个例子
   取每个部门的最高工资
   取每个部门的员工数
   取男人数和女人数

   小窍门：‘每’这个字后面的字段，就是我们分组的依据，只是个小窍门，但是不能表示所有的情况，看上面第三个分组，没有'每'字，
   我们能用id进行分组吗，能，但是id是不是重复度很低啊，基本没有重复啊，对不对，这样的字段适合做分组的依据吗？不适合，对

#4、大前提：
   可以按照任意字段分组，但是分组完毕后，比如group by post，只能查看post字段，如果想查看组内信息，需要借助于聚合函数#
```

来，我们来测试一下：

```
mysql> select * from employee group by post;
+----+--------+--------+-----+------------+-----------------------------------------+--------------
| id | name   | sex    | age | hire_date  | post                          | post_comment | salary     | office | depart_id |
+----+--------+--------+-----+------------+-----------------------------------------+--------------
| 14 | 张野   | male   |  28 | 2016-03-11 | operation                       | NULL         |  10000.13 |    403 |         3 |
|  9 | 歪歪   | female |  48 | 2015-03-11 | sale                          | NULL         |   3000.13 |    402 |         2 |
|  2 | alex   | male   |  78 | 2015-03-02 | teacher                         | NULL         | 1000000.31 |    401 |         1 |
|  1 | egon   | male   |  18 | 2017-03-01 | 老男孩驻沙河办事处外交大使          | NULL         |   7300.33 |    401 |         1 |
+----+--------+--------+-----+------------+-----------------------------------------+--------------
4 rows in set (0.06 sec)

通过结果可以看出，如果直接通过post部门字段来进行分组，默认拿到的结果都是每组的第一条数据但是你想，我们分组的意义是什么
```

2、 ONLY_FULL_GROUP_BY

```
#查看MySQL 5.7默认的sql_mode如下：
mysql> select @@global.sql_mode;
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AU

#！！！注意
ONLY_FULL_GROUP_BY的语义就是确定select target list中的所有列的值都是明确语义，简单的说来，在ONLY_FULL_GROUP_BY模


#设置sql_mole如下操作(我们可以去掉ONLY_FULL_GROUP_BY模式)：
mysql> set global sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO
```

看一下测试：

```
mysql> select @@global.sql_mode;
+--------------------+
| @@global.sql_mode  |
+--------------------+
|                    |
+--------------------+
1 row in set (0.00 sec)

mysql> select * from emp group by post;
+----+------+--------+-----+------------+--------------------------+--------------+------------+-
| id | name | sex    | age | hire_date  | post                | post_comment | salary     | office | depart_id |
+----+------+--------+-----+------------+--------------------------+--------------+------------+-
| 14 | 张野 | male   |  28 | 2016-03-11 | operation             | NULL         |  10000.13 |    403 |         3 |
|  9 | 歪歪 | female |  48 | 2015-03-11 | sale                | NULL         |   3000.13 |    402 |         2 |
|  2 | alex | male   |  78 | 2015-03-02 | teacher               | NULL         | 1000000.31 |    401 |         1 |
|  1 | egon | male   |  18 | 2017-03-01 | 老男孩驻沙河办事处外交大使 | NULL         |   7300.33 |    401 |         1 |
+----+------+--------+-----+------------+--------------------------+--------------+------------+-
```

```
4 rows in set (0.00 sec)


#由于没有设置ONLY_FULL_GROUP_BY,于是也可以有结果，默认都是组内的第一条记录，但其实这是没有意义的

mysql> set global sql_mode='ONLY_FULL_GROUP_BY';
Query OK, 0 rows affected (0.00 sec)

mysql> quit #设置成功后，一定要退出，然后重新登录方可生效
Bye

mysql> use db1;
Database changed
mysql> select * from emp group by post; #报错
ERROR 1055 (42000): 'db1.emp.id' isn't in GROUP BY  #意思是告诉你，你select后面取的字段必须在你的group by后面的字段里
mysql> select post,count(id) from emp group by post; #只能查看分组依据和使用聚合函数
+----------------------------+-----------+
| post                       | count(id) |
+----------------------------+-----------+
| operation                  |         5 |
| sale                       |         5 |
| teacher                    |         7 |
| 老男孩驻沙河办事处外交大使 |         1 |
+----------------------------+-----------+
4 rows in set (0.00 sec)因为一般分组之后，我们再考虑其中一条数据就没有什么意义了，所以一般我们都会在这种模式下进行分组，
```

3、 GROUP BY

```
单独使用GROUP BY关键字分组
   SELECT post FROM employee GROUP BY post;
   注意：我们按照post字段分组，那么select查询的字段只能是post，想要获取组内的其他相关信息，需要借助函数

GROUP BY关键字和GROUP_CONCAT()函数一起使用,比如说我想按部门分组，每个组有哪些员工，都显示出来，怎么搞
   SELECT post,GROUP_CONCAT(name) FROM employee GROUP BY post;#按照岗位分组，并查看组内所有成员名，通过逗号拼接
   SELECT post,GROUP_CONCAT(name,':',salary) as emp_members FROM employee GROUP BY post;

GROUP BY一般都会与聚合函数一起使用，聚合是什么意思：聚合就是将分组的数据聚集到一起，合并起来搞事情，拿到一个最后的结果
   select post,count(id) as count from employee group by post;#按照岗位分组，并查看每个组有多少人，每个人都有唯一的id号
关于集合函数，mysql提供了以下几种聚合函数：count、max、min、avg、sum等，上面的group_concat也算是一个聚合函数了，
```

强调：

```
如果我们用设置了unique约束的字段作为分组的依据，则每一条记录自成一组，这种分组没有意义
多条记录之间的某个字段值相同，该字段通常用来作为分组的依据
```

4、 聚合函数

```
#强调：聚合函数聚合的是组的内容，若是没有分组，则默认一组

示例：
```

```
SELECT COUNT(*) FROM employee；#count是统计个数用的
SELECT COUNT(*) FROM employee WHERE depart_id=1；#后面跟where条件的意思是统计一下满足depart_id=1这个的所有记
SELECT MAX(salary) FROM employee；#max（）统计分组后每组的最大值，这里没有写group by，那么就是统计整个表中所有
SELECT MIN(salary) FROM employee;
SELECT AVG(salary) FROM employee;
SELECT SUM(salary) FROM employee;
SELECT SUM(salary) FROM employee WHERE depart_id=3;
```

5、 **小练习：**

```
1. 查询岗位名以及岗位包含的所有员工名字#通过上面的需求来整理逻辑：a、先看一下和哪个表有关系：所有的信息都在employee这
下面的题都按照上面这个逻辑来搞一搞：2. 查询岗位名以及各岗位内包含的员工个数
3. 查询公司内男员工和女员工的个数
4. 查询岗位名以及各岗位的平均薪资
5. 查询岗位名以及各岗位的最高薪资
6. 查询岗位名以及各岗位的最低薪资
7. 查询男员工与男员工的平均薪资，女员工与女员工的平均薪资。 #这道题我们自己提炼一下分组依据，是不是就是性别啊#总结：先
```

mysql> select post,avg(salary) from employee where age>=30 group by post;#因为有的部门里面的员工没有大于30岁的，所以没有显示出所有的部门

```
+---------+---------------+
| post | avg(salary) |
+---------+---------------+
| sale | 2500.240000 |
| teacher | 255450.077500 |
+---------+---------------+
2 rows in set (0.09 sec)
```

到这里我们的group by就讲完了，看一下我们完整查询语句里面还有什么

```
SELECT distinct 字段1,字段2... FROM 库名.表名
            WHERE 条件
            GROUP BY field（字段）
            HAVING 筛选     #过滤，过滤之后执行select后面的字段筛选，就是说我要确定一下需要哪个字段的数据，你查询的字段
            ORDER BY field（字段）    #将结果按照后面的字段进行排序
            LIMIT 限制条数
```

```
注意：虽然语法里面我们先写的select，但是并不是先执行的select，按照mysql自己的规范来执行的下面关键字的优先级
    from
    where
    group by
    having
    select
    distinct
    order by
    limit
```

```
#题1：分组
mysql> select post,group_concat(name) from employee group by post;
```

```
+--------------------------------------+-----------------------------------------------------+
| post                                 | group_concat(name)                                  |
+--------------------------------------+-----------------------------------------------------+
| operation                            | 张野,程咬金,程咬银,程咬铜,程咬铁                   |
| sale                                 | 歪歪,丫丫,丁丁,星星,格格                           |
| teacher                              | alex,wupeiqi,yuanhao,liwenzhou,jingliyang,jinxin,成龙 |
| 老男孩驻沙河办事处外交大使           | egon                                                |
+--------------------------------------+-----------------------------------------------------+


#题目2:
mysql> select post,count(id) from employee group by post;
+--------------------------------------+-----------+
| post                                 | count(id) |
+--------------------------------------+-----------+
| operation                            |         5 |
| sale                                 |         5 |
| teacher                              |         7 |
| 老男孩驻沙河办事处外交大使           |         1 |
+--------------------------------------+-----------+


#题目3:
mysql> select sex,count(id) from employee group by sex;
+--------+-----------+
| sex    | count(id) |
+--------+-----------+
| male   |        10 |
| female |         8 |
+--------+-----------+

#题目4:
mysql> select post,avg(salary) from employee group by post;
+--------------------------------------+---------------+
| post                                 | avg(salary)   |
+--------------------------------------+---------------+
| operation                            |  16800.026000 |
| sale                                 |   2600.294000 |
| teacher                              | 151842.901429 |
| 老男孩驻沙河办事处外交大使           |   7300.330000 |
+--------------------------------------+---------------+

#题目5
mysql> select post,max(salary) from employee group by post;
+--------------------------------------+-------------+
| post                                 | max(salary) |
+--------------------------------------+-------------+
| operation                            |    20000.00 |
| sale                                 |     4000.33 |
| teacher                              |  1000000.31 |
| 老男孩驻沙河办事处外交大使           |     7300.33 |
+--------------------------------------+-------------+

#题目6
mysql> select post,min(salary) from employee group by post;
+--------------------------------------+-------------+
| post                                 | min(salary) |
+--------------------------------------+-------------+
| operation                            |    10000.13 |
| sale                                 |     1000.37 |
| teacher                              |     2100.00 |
| 老男孩驻沙河办事处外交大使           |     7300.33 |
```

```
+-----------------------------------------+-------------+

#题目七
mysql> select sex,avg(salary) from employee group by sex;
+--------+---------------+
| sex    | avg(salary)   |
+--------+---------------+
| male   | 110920.077000 |
| female |   7250.183750 |
+--------+---------------+
```

六 HAVING过滤

讲having之前，我们补充一个点：之前我们写的查询语句是这样的：select id,name from employee;**实际上我们在**select**每个字段的时候，省略了一个表名，有的人可能会这样写，**select employee.id,employee.name from employee;**你会发现查询出来的结果是一样的，但是如果你要将查询出来的结果表，起一个新表名的话，带着表名这样写就错了**

select employee.id,employee.name from employee as tb1;**这样执行会下面的报错：**

```
mysql> select employee.id,employee.name from employee as tb1;
ERROR 1054 (42S22): Unknown column 'employee.id' in 'field list'
```

因为这个语句先执行的是谁啊，是不是我们的from**啊，那么后面的**as**也是比**select**要先执行的，所以你先将表**employee**起了个新名字叫做**tb1**，然后在**tb1**里面取查询数据，那么**tb1**里面找不到**employee.id**这个字段，就会报错，如果我们查询的时候不带表名，你**as**来起一个新的表名也是没问题的，简单提一下这个内容，知道就好了**

HAVING**与**WHERE**不一样的地方在于**!!!!!!

```
having的语法格式和where是一模一样的，只不过having是在分组之后进行的进一步的过滤，where不能使用聚合函数，having是可以
#1. Where 发生在分组group by之前，因而Where中可以有任意字段，但是绝对不能使用聚合函数。

#2. Having发生在分组group by之后，因而Having中可以使用分组的字段，无法直接取到其他字段,having是可以使用聚合函数
```

having简单测试：

```
#来个需求：统计各部门年龄在30岁及以上的员工的平均薪资，并且保留平均工资大于10000的部门答案：select post,avg(salary) as
```

```
+---------+---------------+
| post | new_sa |
+---------+---------------+
| teacher | 255450.077500 |
+---------+---------------+
1 row in set (0.00 sec)
```

```
然后我们看这样一句话：select * from employee having avg(salary) > 10000;只要一运行就会报错：
```

```
mysql> select * from employee having avg(salary) > 10000;
ERROR 1140 (42000): Mixing of GROUP columns (MIN(),MAX(),COUNT(),...) with no GROUP columns is illegal if there is no GROUP BY clause
```

是因为having只能在group by后面运行

小练习：

1. 查询各岗位内包含的员工个数小于2的岗位名、岗位内包含员工名字、个数
3. 查询各岗位平均薪资大于10000的岗位名、平均工资
4. 查询各岗位平均薪资大于10000且小于20000的岗位名、平均工资

```
#题1：
mysql> select post,group_concat(name),count(id) from employee group by post having count(id) < 2;
+----------------------------------------+--------------------+-----------+
| post                                   | group_concat(name) | count(id) |
+----------------------------------------+--------------------+-----------+
| 老男孩驻沙河办事处外交大使             | egon               |         1 |
+----------------------------------------+--------------------+-----------+

#题目2：
mysql> select post,avg(salary) from employee group by post having avg(salary) > 10000;
+-----------+---------------+
| post      | avg(salary)   |
+-----------+---------------+
| operation |  16800.026000 |
| teacher   | 151842.901429 |
+-----------+---------------+

#题目3：
mysql> select post,avg(salary) from employee group by post having avg(salary) > 10000 and avg(salary) <20000;
+-----------+---------------+
| post      | avg(salary)   |
+-----------+---------------+
| operation | 16800.026000 |
+-----------+---------------+
```

说一下去重：distinct

将查询的结果进行去重：select distinct post from employee; 注意distinct去重要写在查询字段的前面，不然会报错，关于distinct使用时的其他问题看下面的总结

```
有时需要查询出某个字段不重复的记录，这时可以使用mysql提供的distinct这个关键字来过滤重复的记录，但是实际中我们往往用dist

mysql> select id,count(distinct post) from employee;
ERROR 1140 (42000): Mixing of GROUP columns (MIN(),MAX(),COUNT(),...) with no GROUP columns is illegal if there is no
报错了：是因为distinct不能返回其他的字段，只能返回目标字段
mysql> select count(distinct post) from employee;
+----------------------+
| count(distinct post) |
```

```
+----------------------+
|                    4 |
+----------------------+
1 row in set (0.00 sec)
```

说完去重，我们看一下order by，排序

七 查询排序:ORDER BY

```
按单列排序
    SELECT * FROM employee ORDER BY salary; #默认是升序排列
    SELECT * FROM employee ORDER BY salary ASC; #升序
    SELECT * FROM employee ORDER BY salary DESC; #降序

但是你看，如果我们按照age来排序，你看看是什么效果：
mysql> SELECT * FROM employee ORDER BY age;
+----+------------+--------+-----+------------+-----------------------------------------+----------
| id | name | sex | age | hire_date | post | post_comment | salary | office | depart_id |
+----+------------+--------+-----+------------+-----------------------------------------+----------
| 1 | egon | male | 18 | 2017-03-01 | 老男孩驻沙河办事处外交大使 | NULL | 7300.33 | 401 | 1 |
| 17 | 程咬铜 | male | 18 | 2015-04-11 | operation | NULL | 18000.00 | 403 | 3 |
| 16 | 程咬银 | female | 18 | 2013-03-11 | operation | NULL | 19000.00 | 403 | 3 |
| 15 | 程咬金 | male | 18 | 1997-03-12 | operation | NULL | 20000.00 | 403 | 3 |
| 12 | 星星 | female | 18 | 2016-05-13 | sale | NULL | 3000.29 | 402 | 2 |
| 11 | 丁丁 | female | 18 | 2011-03-12 | sale | NULL | 1000.37 | 402 | 2 |
| 18 | 程咬铁 | female | 18 | 2014-05-12 | operation | NULL | 17000.00 | 403 | 3 |
| 7 | jinxin | male | 18 | 1900-03-01 | teacher | NULL | 30000.00 | 401 | 1 |
| 6 | jingliyang | female | 18 | 2011-02-11 | teacher | NULL | 9000.00 | 401 | 1 |
| 13 | 格格 | female | 28 | 2017-01-27 | sale | NULL | 4000.33 | 402 | 2 |
| 14 | 张野 | male | 28 | 2016-03-11 | operation | NULL | 10000.13 | 403 | 3 |
| 5 | liwenzhou | male | 28 | 2012-11-01 | teacher | NULL | 2100.00 | 401 | 1 |
| 10 | 丫丫 | female | 38 | 2010-11-01 | sale | NULL | 2000.35 | 402 | 2 |
| 9 | 歪歪 | female | 48 | 2015-03-11 | sale | NULL | 3000.13 | 402 | 2 |
| 8 | 成龙 | male | 48 | 2010-11-11 | teacher | NULL | 10000.00 | 401 | 1 |
| 4 | yuanhao | male | 73 | 2014-07-01 | teacher | NULL | 3500.00 | 401 | 1 |
| 2 | alex | male | 78 | 2015-03-02 | teacher | NULL | 1000000.31 | 401 | 1 |
| 3 | wupeiqi | male | 81 | 2013-03-05 | teacher | NULL | 8300.00 | 401 | 1 |
+----+------------+--------+-----+------------+-----------------------------------------+----------

 发现什么，按照年龄来升序排的，没问题，但是你看年龄相同的那些按什么排的，是不是看着是乱的啊，但是不管它对这种相同数据的

 所以我们可以给相同的这些数据指定一个排序的依据，看下面：
按多列排序:先按照age升序，如果年纪相同，则按照薪资降序
    SELECT * from employee
        ORDER BY age, #注意排序的条件用逗号分隔
        salary DESC;
```

小练习：

```
1. 查询所有员工信息，先按照age升序排序，如果age相同则按照hire_date降序排序
2. 查询各岗位平均薪资大于10000的岗位名、平均工资,结果按平均薪资升序排列
```

3. 查询各岗位平均薪资大于10000的岗位名、平均工资,结果按平均薪资降序排列

```
#题目1
mysql> select * from employee ORDER BY age asc,hire_date desc;

#题目2
mysql> select post,avg(salary) from employee group by post having avg(salary) > 10000 order by avg(salary) asc;
#注意：查询语句的语法是固定上面这样写的，但是运行顺序是这样的：1、from 2、where 3、group by 4、having 5、select 6
+-----------+---------------+
| post      | avg(salary)   |
+-----------+---------------+
| operation |  16800.026000 |
| teacher   | 151842.901429 |
+-----------+---------------+

#题目3
mysql> select post,avg(salary) from employee group by post having avg(salary) > 10000 order by avg(salary) desc;
+-----------+---------------+
| post      | avg(salary)   |
+-----------+---------------+
| teacher   | 151842.901429 |
| operation |  16800.026000 |
+-----------+---------------+
```

八 限制查询的记录数:LIMIT

```
示例:       #取出工资最高的前三位
   SELECT * FROM employee ORDER BY salary DESC
      LIMIT 3;                  #默认初始位置为0,从第一条开始顺序取出三条

   SELECT * FROM employee ORDER BY salary DESC
      LIMIT 0,5; #从第0开始，即先查询出第一条，然后包含这一条在内往后查5条

   SELECT * FROM employee ORDER BY salary DESC
      LIMIT 5,5; #从第5开始，即先查询出第6条，然后包含这一条在内往后查5条
```

小练习：分页显示，每页显示5条。我们工作中经常会涉及到数据分页显示，因为数据量很大的时候，加入上10w条数据，我们是不是要分开给用户显示啊，要不然页面上都显示不过来，即便是显示过来了，用户看着是不是也很不爽啊，要一直往下面滚轮，对不对，用户体验不好，所以你会发现有好多的网站都可以看到一个分页的功能。

```
mysql> select * from  employee limit 0,5;
+----+-----------+------+-----+------------+-----------------------------------------+--------------
| id | name      | sex  | age | hire_date  | post                                    | post_comment | salary     | office | depart_id |
+----+-----------+------+-----+------------+-----------------------------------------+--------------
```

```
| 1 | egon     | male |  18 | 2017-03-01 | 老男孩驻沙河办事处外交大使          | NULL        |   7300.33 |   401 |       1 |
| 2 | alex     | male |  78 | 2015-03-02 | teacher                        |             | 1000000.31 |   401 |       1 |
| 3 | wupeiqi  | male |  81 | 2013-03-05 | teacher                        | NULL        |   8300.00 |   401 |       1 |
| 4 | yuanhao  | male |  73 | 2014-07-01 | teacher                        | NULL        |   3500.00 |   401 |       1 |
| 5 | liwenzhou | male |  28 | 2012-11-01 | teacher                        | NULL        |   2100.00 |   401 |       1 |
+----+-----------+------+-----+------------+--------------------------------------+-------------
5 rows in set (0.00 sec)

mysql> select * from  employee limit 5,5;
+----+------------+--------+-----+------------+---------+--------------+----------+--------+------
| id | name       | sex    | age | hire_date  | post    | post_comment | salary   | office | depart_id |
+----+------------+--------+-----+------------+---------+--------------+----------+--------+------
|  6 | jingliyang | female |  18 | 2011-02-11 | teacher | NULL         |  9000.00 |   401 |       1 |
|  7 | jinxin     | male   |  18 | 1900-03-01 | teacher | NULL         | 30000.00 |   401 |       1 |
|  8 | 成龙       | male   |  48 | 2010-11-11 | teacher | NULL         | 10000.00 |   401 |       1 |
|  9 | 歪歪       | female |  48 | 2015-03-11 | sale    | NULL         |  3000.13 |   402 |       2 |
| 10 | 丫丫       | female |  38 | 2010-11-01 | sale    | NULL         |  2000.35 |   402 |       2 |
+----+------------+--------+-----+------------+---------+--------------+----------+--------+------
5 rows in set (0.00 sec)

mysql> select * from  employee limit 10,5;
+----+-----------+--------+-----+------------+-----------+--------------+----------+--------+------
| id | name      | sex    | age | hire_date  | post      | post_comment | salary   | office | depart_id |
+----+-----------+--------+-----+------------+-----------+--------------+----------+--------+------
| 11 | 丁丁      | female |  18 | 2011-03-12 | sale      | NULL         |  1000.37 |   402 |       2 |
| 12 | 星星      | female |  18 | 2016-05-13 | sale      | NULL         |  3000.29 |   402 |       2 |
| 13 | 格格      | female |  28 | 2017-01-27 | sale      | NULL         |  4000.33 |   402 |       2 |
| 14 | 张野      | male   |  28 | 2016-03-11 | operation | NULL         | 10000.13 |   403 |       3 |
| 15 | 程咬金    | male   |  18 | 1997-03-12 | operation | NULL         | 20000.00 |   403 |       3 |
+----+-----------+--------+-----+------------+-----------+--------------+----------+--------+------
5 rows in set (0.00 sec)到最后不够五条了怎么办，完全不影响，接着写
```

```
mysql> select * from employee limit 15,5;
+----+-----------+--------+-----+------------+-----------+--------------+----------+---
-----+-----------+
| id | name | sex | age | hire_date | post | post_comment | salary | office | depart_id |
+----+-----------+--------+-----+------------+-----------+--------------+----------+---
-----+-----------+
| 16 | 程咬银 | female | 18 | 2013-03-11 | operation | NULL | 19000.00 | 403 | 3 |
| 17 | 程咬铜 | male | 18 | 2015-04-11 | operation | NULL | 18000.00 | 403 | 3 |
| 18 | 程咬铁 | female | 18 | 2014-05-12 | operation | NULL | 17000.00 | 403 | 3 |
+----+-----------+--------+-----+------------+-----------+--------------+----------+---
-----+-----------+
3 rows in set (0.00 sec)
```

#到目前为止，单表查询所有的语法都讲完了，语法就是按照我们博客最上面说的语法顺序来写，但是执行的时候，要按照对应的各个方法的优先级去执行。

九 使用正则表达式查询

```
#之前我们用like做模糊匹配，只有%和_，局限性比较强，所以我们说一个正则，之前我们是不是学过正则匹配，你之前学的正则表达式

SELECT * FROM employee WHERE name REGEXP 'on$';

SELECT * FROM employee WHERE name REGEXP 'm{2}';
```

```
小结：对字符串匹配的方式
WHERE name = 'egon';
WHERE name LIKE 'yua%';
WHERE name REGEXP 'on$';
```

小练习：

```
查看所有员工中名字是jin开头，n或者g结果的员工信息
```

```
select * from employee where name regexp '^jin.*[g|n]$';
```