# **常见 SQL 面试题：经典 50 例**

## **SQL基础知识整理**

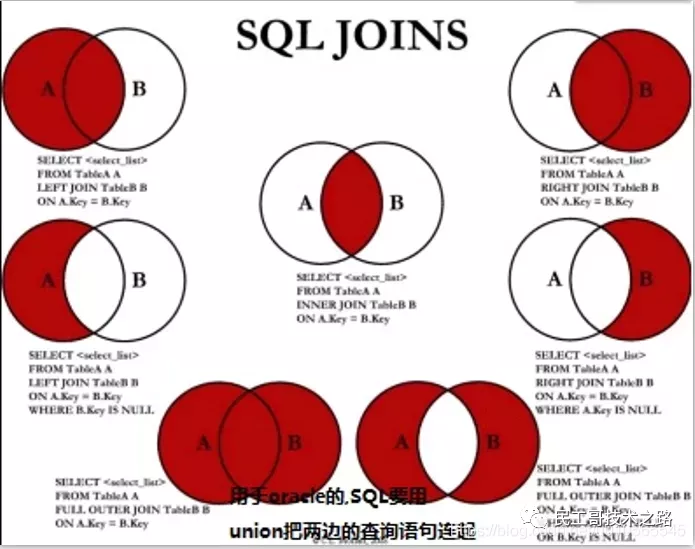
select 查询结果，如: [学号,平均成绩：组函数avg(成绩)]  
  
from 从哪张表中查找数据，如:[涉及到成绩：成绩表score]  
  
where 查询条件，如:[b.课程号='0003' and b.成绩>80]  
  
group by 分组，如:[每个学生的平均：按学号分组](oracle,SQL server中出现在select 子句后的非分组函数，必须出现在group by子句后出现),MySQL中可以不用  
  
having 对分组结果指定条件，如:[大于60分]  
  
order by 对查询结果排序，如:[增序: 成绩  ASC / 降序: 成绩 DESC];  
  
limit 使用limt子句返回topN（对应这个问题返回的成绩前两名），如:[ limit  2 ==>从0索引开始读取2个]limit==>从0索引开始 [0,N-1]

**select** \* **from** **table** **limit** 2,1;                  
-- 含义是跳过2条取出1条数据，limit后面是从第2条开始读，读取1条信息，即读取第3条数据  
   
**select** \* **from** **table** **limit** 2 **offset** 1;       
-- 含义是从第1条（不包括）数据开始取出2条数据，limit后面跟的是2条数据，offset后面是从第1条开始读取，即读取第2,3条

**组函数**: 去重 distinct()  统计总数sum()   计算个数count()  平均数avg()  最大值max() 最小数min()

**多表连接**: 内连接(省略默认inner) join ...on..左连接left join tableName as b on a.key ==b.key右连接right join  连接union(无重复(过滤去重))和union all(有重复[不过滤去重])

* union 并集
* union all(有重复)
* oracle(SQL server)数据库
* intersect 交集
* minus(except) 相减(差集)



## **oracle**

#### **一、数据库对象：表(table)  视图(view)  序列(sequence)  索引(index)  同义词(synonym)**

###### **1.视图: 存储起来的 select 语句**

**create** **view** emp\_vw  
**as**  
**select** employee\_id, last\_name, salary  
**from** employees  
**where** department\_id = 90;  
  
**select** \* **from** emp\_vw;

可以对简单视图进行 DML 操作

**update** emp\_vw  
**set** last\_name = 'HelloKitty'  
**where** employee\_id = 100;  
  
**select** \* **from** employees  
**where** employee\_id = 100;

1). 复杂视图

**create** **view** emp\_vw2  
**as**  
**select** department\_id, **avg**(salary) avg\_sal  
**from** employees  
**group** **by** department\_id;  
  
**select** \* **from** emp\_vw2;

复杂视图不能进行 DML 操作

update emp\_vw2  
set avg\_sal = 10000  
where department\_id = 100;

###### **2.序列：用于生成一组有规律的数值。（通常用于为主键设置值）**

**create** **sequence** emp\_seq1  
**start** **with** 1  
**increment** **by** 1  
maxvalue 10000  
**minvalue** 1  
**cycle**  
nocache;  
  
**select** emp\_seq1.currval **from** dual;  
  
**select** emp\_seq1.nextval **from** dual;

**问题：裂缝，原因**：

* 当多个表共用同一个序列时。
* rollback
* 发生异常

**create** **table** emp1(  
       **id** number(10),  
       **name** varchar2(30)  
);  
  
**insert** **into** emp1  
**values**(emp\_seq1.nextval, '张三');  
  
**select** \* **from** emp1;

###### **3.索引：提高查询效率**

自动创建：Oracle 会为具有唯一约束(唯一约束，主键约束)的列，自动创建索引

**create** **table** emp2(  
       **id** number(10) primary **key**,  
       **name** varchar2(30)  
)

手动创建

**create** **index** emp\_idx  
**on** emp2(**name**);  
  
**create** **index** emp\_idx2  
**on** emp2(**id**, **name**);

###### **4.同义词**

**create** **synonym** d1 **for** departments;  
  
**select** \* **from** d1;

###### **5.表：**

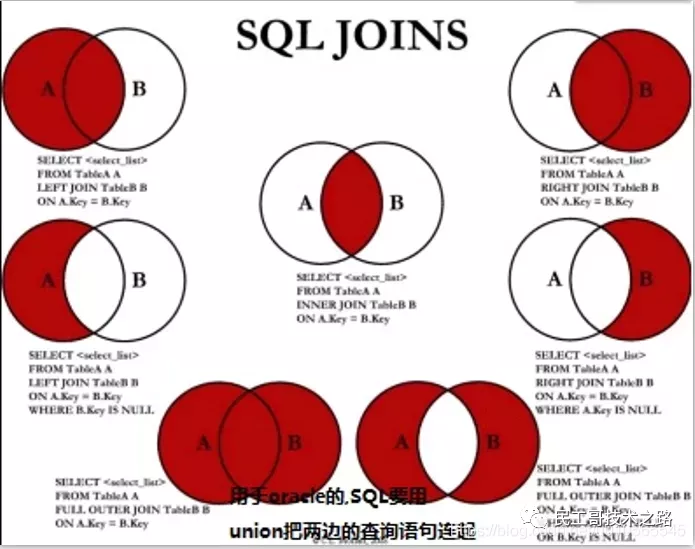
DDL ：数据定义语言 create table .../ drop table ... / rename ... to..../ truncate table.../alter table ...

DML : 数据操纵语言

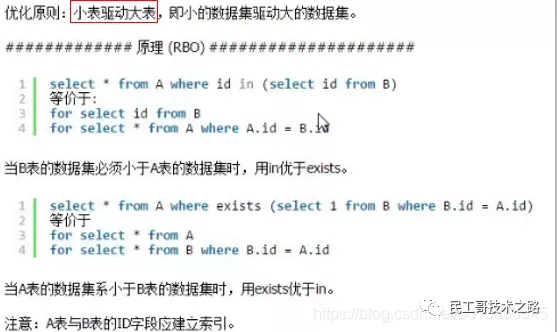
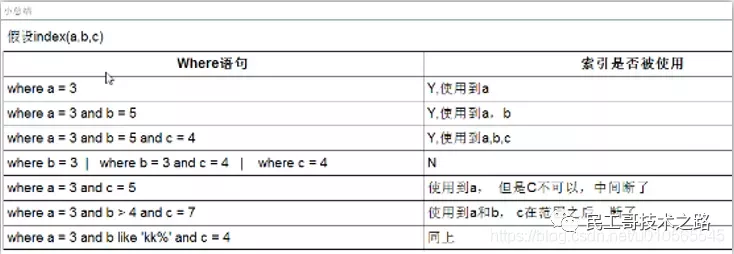
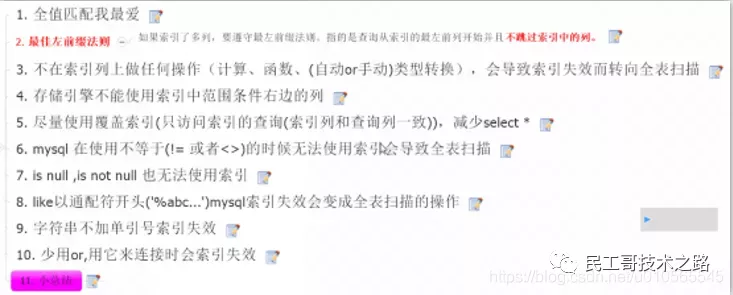
**insert** **into** ... **values** ...  
**update** ... **set** ... **where** ...  
**delete** **from** ... **where** ...

【重要】

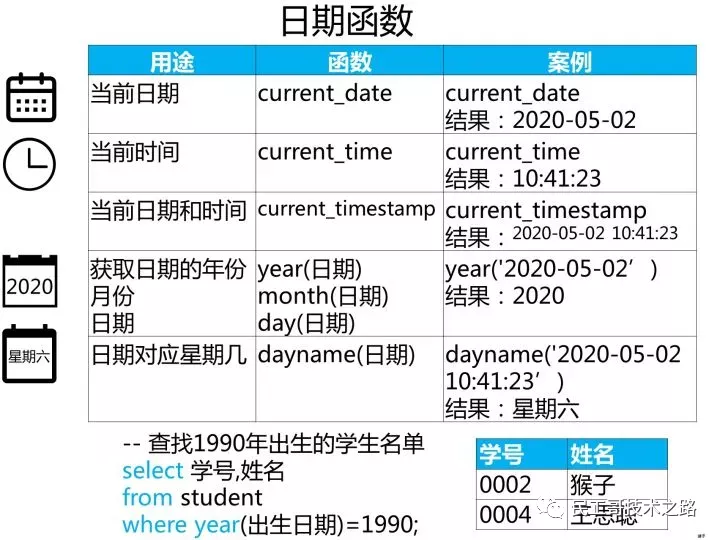
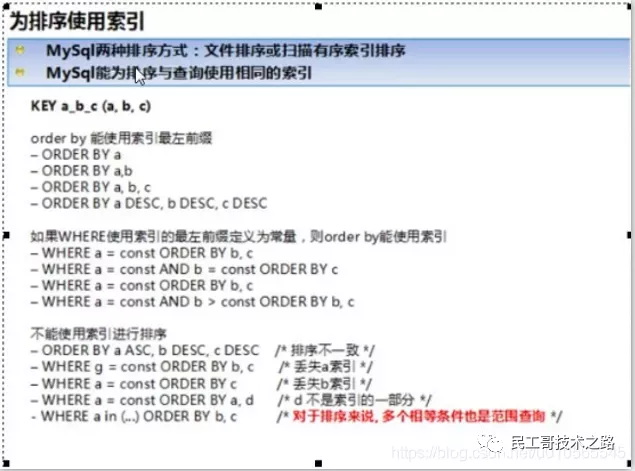
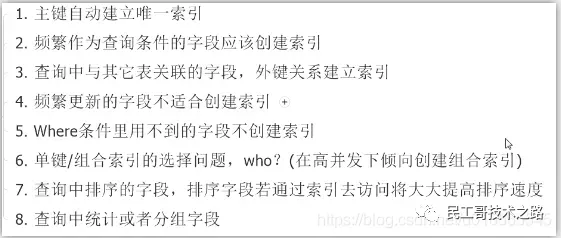
**select** ... 组函数(**MIN**()/**MAX**()/**SUM**()/**AVG**()/**COUNT**())  
**from** ...join ... **on** ... 左外连接：**left** **join** ... **on** ... 右外连接: **right** **join** ... **on** ...  
**where** ...  
**group** **by** ... (**oracle**,**SQL** **server**中出现在**select** 子句后的非分组函数，必须出现在 **group** **by**子句后)  
**having** ... 用于过滤 组函数  
**order** **by** ... **asc** 升序， **desc** 降序  
**limit** (0,4) 限制N条数据 如: topN数据  
- **union** 并集  
- **union** **all**(有重复)  
- **intersect** 交集  
- **minus** 相减  
DCL : 数据控制语言  **commit** : 提交 / **rollback** : 回滚 / 授权grant...to...  /**revoke**



## **索引**



何时创建索引:



**select** employee\_id, last\_name, salary, department\_id  
**from** employees  
**where** department\_id **in** (70, 80) --> 70:1  80:34

* union 并集
* union all(有重复部分)
* intersect 交集
* minus 相减

**select** employee\_id, last\_name, salary, department\_id  
**from** employees  
**where** department\_id **in** (80, 90)  --> 90:4  80:34

问题：查询工资大于149号员工工资的员工的信息

**select** \*   
**from** employees  
**where** salary > (  
      **select** salary  
      **from** employees  
      **where** employee\_id = 149  
)

问题：查询与141号或174号员工的manager\_id和department\_id相同的其他员工的employee\_id, manager\_id, department\_id

**select** employee\_id, manager\_id, department\_id  
**from** employees  
**where** manager\_id **in** (  
      **select** manager\_id  
      **from** employees  
      **where** employee\_id **in**(141, 174)  
) **and** department\_id **in** (  
      **select** department\_id  
      **from** employees  
      **where** employee\_id **in**(141, 174)  
) **and** employee\_id **not** **in** (141, 174);  
  
**select** employee\_id, manager\_id, department\_id  
**from** employees  
**where** (manager\_id, department\_id) **in** (  
      **select** manager\_id, department\_id  
      **from** employees  
      **where** employee\_id **in** (141, 174)  
) **and** employee\_id **not** **in**(141, 174);

from 子句中使用子查询

**select** **max**(**avg**(salary))  
**from** employees  
**group** **by** department\_id;  
  
**select** **max**(avg\_sal)  
**from** (  
      **select** **avg**(salary) avg\_sal  
      **from** employees  
      **group** **by** department\_id  
)

问题：返回比本部门平均工资高的员工的last\_name, department\_id, salary及平均工资

**select** last\_name, department\_id, salary, (**select** **avg**(salary) **from** employees **where** department\_id = e1.department\_id)  
**from** employees e1  
**where** salary > (  
      **select** **avg**(salary)  
      **from** employees e2  
      **where** e1.department\_id = e2.department\_id  
)  
  
**select** last\_name, e1.department\_id, salary, avg\_sal  
**from** employees e1, (  
     **select** department\_id, **avg**(salary) avg\_sal  
     **from** employees  
     **group** **by** department\_id  
) e2  
**where** e1.department\_id = e2.department\_id  
**and** e1.salary > e2.avg\_sal;  
case...when ... then... when ... then ... else ... **end**

查询：若部门为10 查看工资的 1.1 倍，部门号为 20 工资的1.2倍，其余 1.3 倍

**SELECT**  
 employee\_id,  
 last\_name,  
 salary,  
**CASE**  
  department\_id   
  **WHEN** 10 **THEN**  
  salary \* 1.1                                                             
  **WHEN** 20 **THEN**  
  salary \* 1.2  **ELSE** salary \* 1.3                                                             
 **END** "new\_salary"   
**FROM**  
 employees;  
**SELECT**  
 employee\_id,  
 last\_name,  
 salary,  
 **decode**( department\_id, 10, salary \* 1.1, 20, salary \* 1.2,  salary \* 1.3 ) "new\_salary"   
**FROM**  
 employees;

问题：显式员工的employee\_id,last\_name和location。其中，若员工department\_id与location\_id为1800的department\_id相同，则location为’Canada’,其余则为’USA’。

**select** employee\_id, last\_name, **case** department\_id **when** (  
                    **select** department\_id  
                    **from** departments  
                    **where** location\_id = 1800  
) **then** 'Canada' **else** 'USA' **end** "location"  
**from** employees;

问题：查询员工的employee\_id,last\_name,要求按照员工的department\_name排序 select employee\_id, last\_name

from employees e1  
order by (  
      **select** department\_name  
      **from** departments d1  
      **where** e1.department\_id = d1.department\_id  
)

## **SQL 优化：能使用 EXISTS 就不要使用 IN**

问题：查询公司管理者的employee\_id,last\_name,job\_id,department\_id信息

**select** employee\_id, last\_name, job\_id, department\_id  
**from** employees  
**where** employee\_id **in** (  
      **select** manager\_id  
      **from** employees  
)

**select** employee\_id, last\_name, job\_id, department\_id  
**from** employees e1  
**where** **exists** (  
      **select** 'x'  
      **from** employees e2  
      **where** e1.employee\_id = e2.manager\_id  
)

问题：查询departments表中，不存在于employees表中的部门的department\_id和department\_name

**select** department\_id, department\_name  
**from** departments d1  
**where** **not** **exists** (  
      **select** 'x'  
      **from** employees e1  
      **where** e1.department\_id = d1.department\_id  
)

更改 108 员工的信息: 使其工资变为所在部门中的最高工资, job 变为公司中平均工资最低的 job

**update** employees e1  
**set** salary = (  
    **select** **max**(salary)  
    **from** employees e2  
    **where** e1.department\_id = e2.department\_id  
), job\_id = (  
   **select** job\_id  
   **from** employees  
   **group** **by** job\_id  
   **having** **avg**(salary) = (  
         **select** **min**(**avg**(salary))  
         **from** employees  
         **group** **by** job\_id  
   )  
)  
**where** employee\_id = 108;

删除 108 号员工所在部门中工资最低的那个员工.

**delete** **from** employees e1  
**where** salary = (  
      **select** **min**(salary)  
      **from** employees  
      **where** department\_id = (  
            **select** department\_id  
            **from** employees  
            **where** employee\_id = 108  
      )  
)  
  
**select** \* **from** employees **where** employee\_id = 108;  
**select** \* **from** employees **where** department\_id = 100  
**order** **by** salary;  
  
**rollback**;

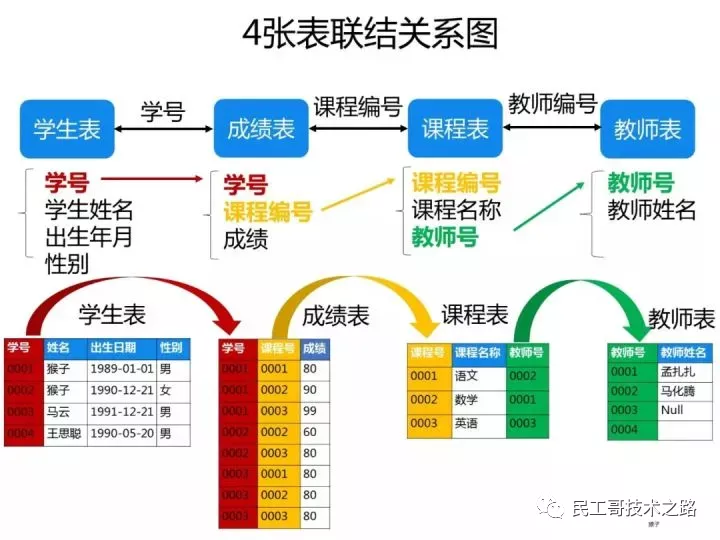
## **常见的SQL面试题：经典50题**

已知有如下4张表：

* 学生表：student(学号,学生姓名,出生年月,性别)
* 成绩表：score(学号,课程号,成绩)
* 课程表：course(课程号,课程名称,教师号)
* 教师表：teacher(教师号,教师姓名)

根据以上信息按照下面要求写出对应的SQL语句。（搜索公众号民工哥技术之路，回复“1024”，送你一份技术资源宝典）

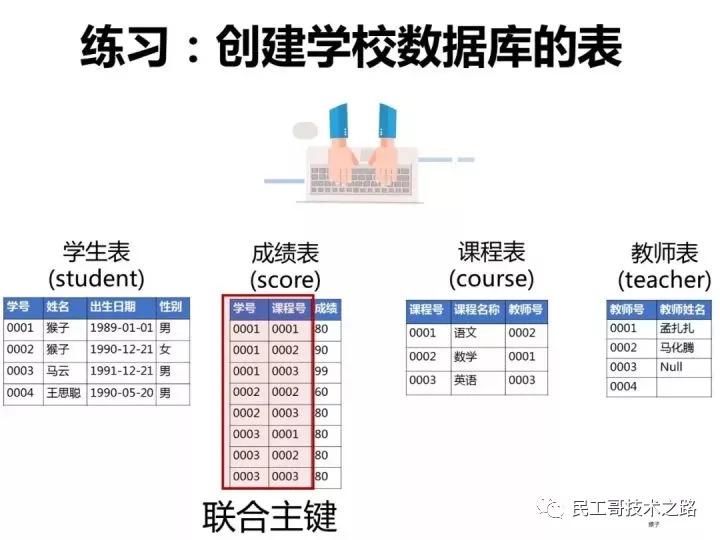
ps：这些题考察SQL的编写能力，对于这类型的题目，需要你先把4张表之间的关联关系搞清楚了，最好的办法是自己在草稿纸上画出关联图，然后再编写对应的SQL语句就比较容易了。下图是我画的这4张表的关系图，可以看出它们之间是通过哪些外键关联起来的：



### **一、创建数据库和表**

为了演示题目的运行过程，我们先按下面语句在客户端navicat中创建数据库和表。

如何你还不懂什么是数据库，什么是客户端navicat，可以先学习这个：



###### **1.创建表**

1）创建学生表（student）

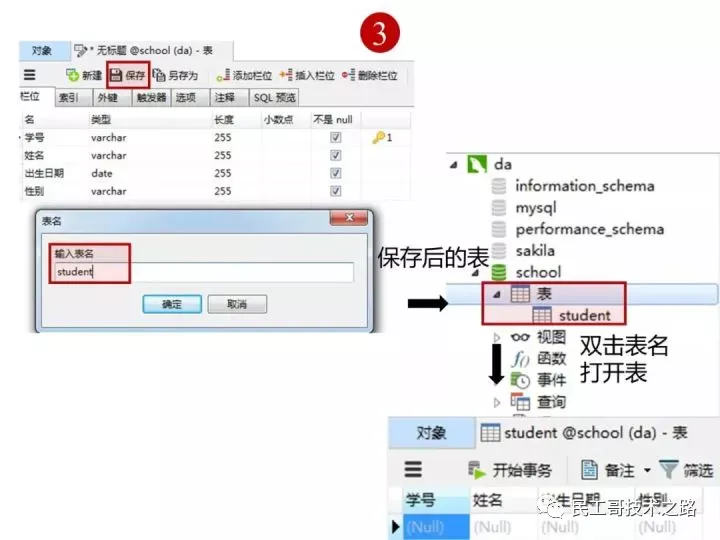
按下图在客户端navicat里创建学生表。推荐：企业面试题汇总



学生表的“学号”列设置为主键约束，下图是每一列设置的数据类型和约束

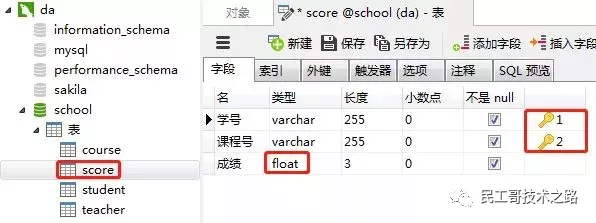


创建完表，点击“保存”



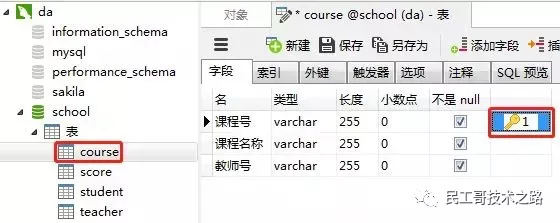
2）创建成绩表（score）

同样的步骤，创建"成绩表“。“课程表的“学号”和“课程号”一起设置为主键约束（联合主键），“成绩”这一列设置为数值类型（float，浮点数值）



3）创建课程表（course）

课程表的“课程号”设置为主键约束

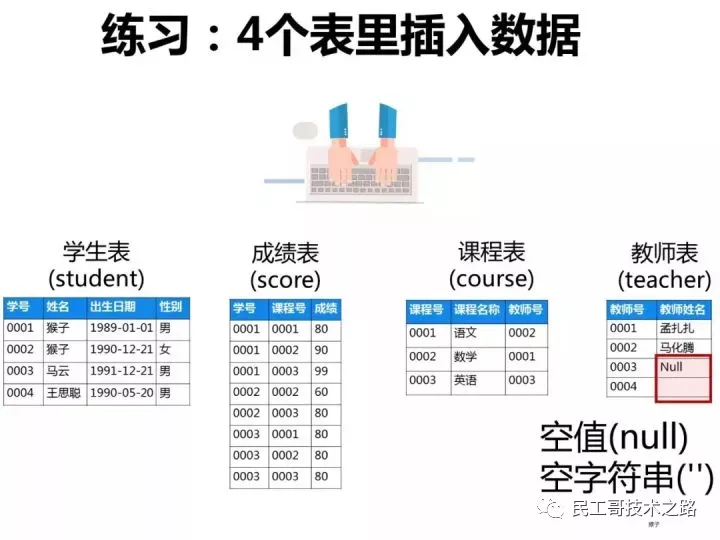


4）教师表（teacher）

教师表的“教师号”列设置为主键约束，教师姓名这一列设置约束为“null”（红框的地方不勾选），表示这一列允许包含空值（null）。推荐：[企业面试题汇总](https://mp.weixin.qq.com/mp/appmsgalbum?__biz=MzI0MDQ4MTM5NQ==&action=getalbum&album_id=1790435592028160001" \l "wechat_redirect" \t "https://mp.weixin.qq.com/_blank)



向表中添加数据

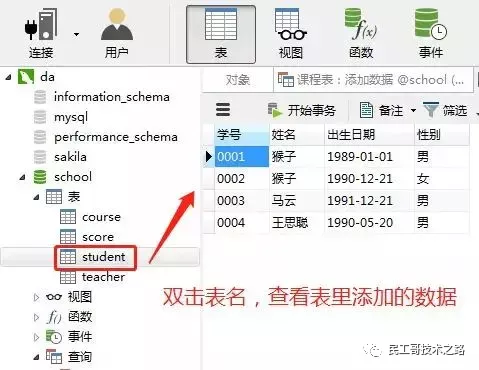
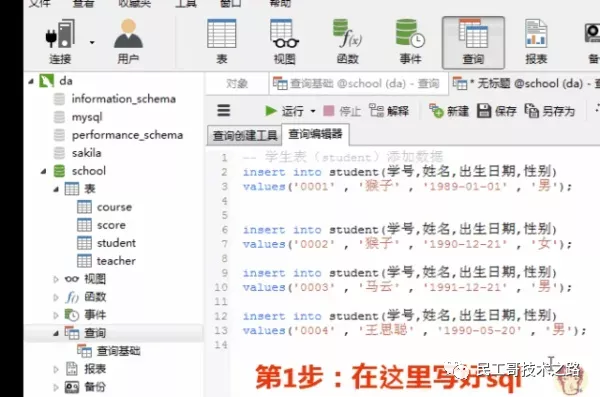
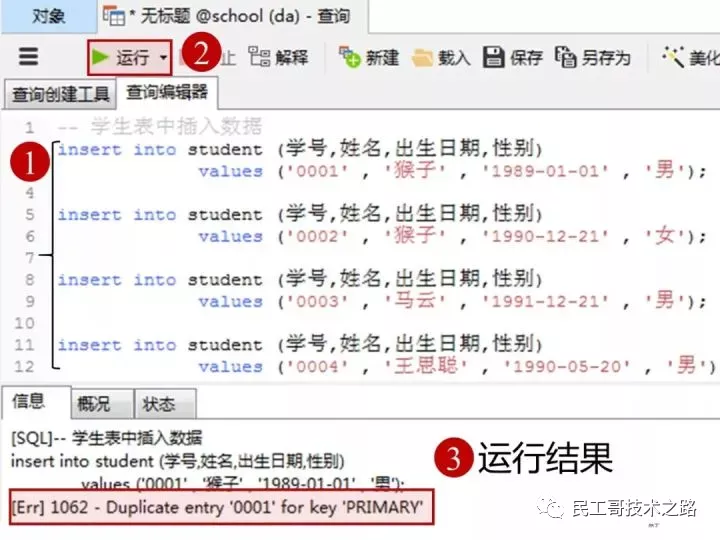


1）向学生表里添加数据

添加数据的sql

**insert** **into** student(学号,姓名,出生日期,性别)   
**values**('0001' , '猴子' , '1989-01-01' , '男');  
   
**insert** **into** student(学号,姓名,出生日期,性别)   
**values**('0002' , '猴子' , '1990-12-21' , '女');  
   
**insert** **into** student(学号,姓名,出生日期,性别)   
**values**('0003' , '马云' , '1991-12-21' , '男');  
   
**insert** **into** student(学号,姓名,出生日期,性别)   
**values**('0004' , '王思聪' , '1990-05-20' , '男');

在客户端navicat里的操作

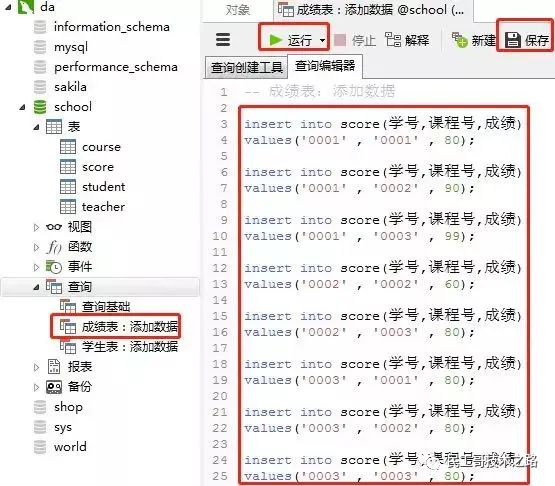


2）成绩表（score）

添加数据的sql

**insert** **into** score(学号,课程号,成绩)   
**values**('0001' , '0001' , 80);  
   
**insert** **into** score(学号,课程号,成绩)   
**values**('0001' , '0002' , 90);  
   
**insert** **into** score(学号,课程号,成绩)   
**values**('0001' , '0003' , 99);  
   
**insert** **into** score(学号,课程号,成绩)   
**values**('0002' , '0002' , 60);  
   
**insert** **into** score(学号,课程号,成绩)   
**values**('0002' , '0003' , 80);  
   
**insert** **into** score(学号,课程号,成绩)   
**values**('0003' , '0001' , 80);  
   
**insert** **into** score(学号,课程号,成绩)   
**values**('0003' , '0002' , 80);  
   
**insert** **into** score(学号,课程号,成绩)   
**values**('0003' , '0003' , 80);

客户端navicat里的操作

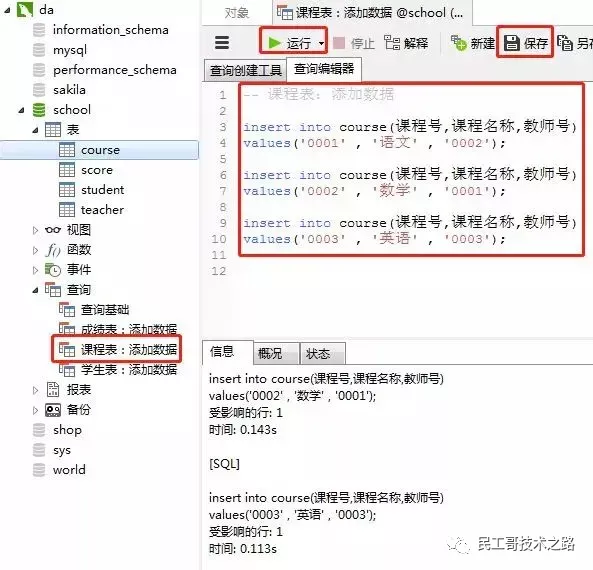


3）课程表

添加数据的sql

**insert** **into** course(课程号,课程名称,教师号)  
**values**('0001' , '语文' , '0002');  
   
**insert** **into** course(课程号,课程名称,教师号)  
**values**('0002' , '数学' , '0001');  
   
**insert** **into** course(课程号,课程名称,教师号)  
**values**('0003' , '英语' , '0003');

客户端navicat里的操作

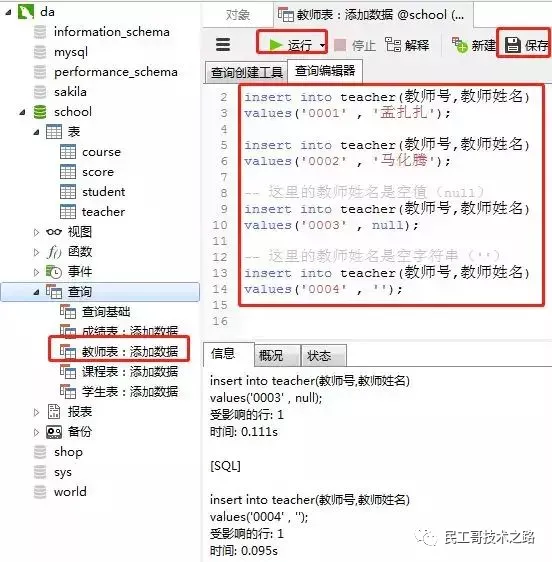


4）教师表里添加数据

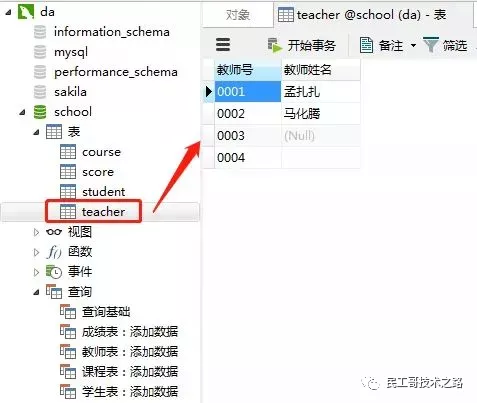
添加数据的sql

-- 教师表：添加数据  
**insert** **into** teacher(教师号,教师姓名)   
**values**('0001' , '孟扎扎');  
   
**insert** **into** teacher(教师号,教师姓名)   
**values**('0002' , '马化腾');  
   
-- 这里的教师姓名是空值（null）  
**insert** **into** teacher(教师号,教师姓名)   
**values**('0003' , **null**);  
   
-- 这里的教师姓名是空字符串（''）  
**insert** **into** teacher(教师号,教师姓名)   
**values**('0004' , '');

客户端navicat里操作



添加结果



## **三、50道面试题**

为了方便学习，我将50道面试题进行了分类



查询姓“猴”的学生名单



查询姓“孟”老师的个数

**select** **count**(教师号)  
**from** teacher  
**where** 教师姓名 **like** '孟%';

###### **2.汇总统计分组分析**

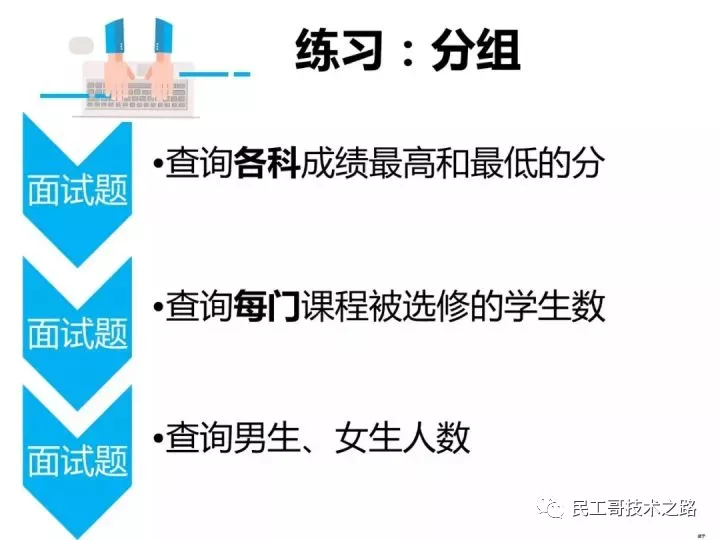


面试题：查询课程编号为“0002”的总成绩

--分析思路  
--select 查询结果 [总成绩:汇总函数sum]  
--from 从哪张表中查找数据[成绩表score]  
--where 查询条件 [课程号是0002]  
**select** **sum**(成绩)  
**from** score  
**where** 课程号 = '0002';

查询选了课程的学生人数

--这个题目翻译成大白话就是：查询有多少人选了课程  
--select 学号，成绩表里学号有重复值需要去掉  
--from 从课程表查找score;  
**select** **count**(**distinct** 学号) **as** 学生人数   
**from** score;



查询各科成绩最高和最低的分， 以如下的形式显示：课程号，最高分，最低分

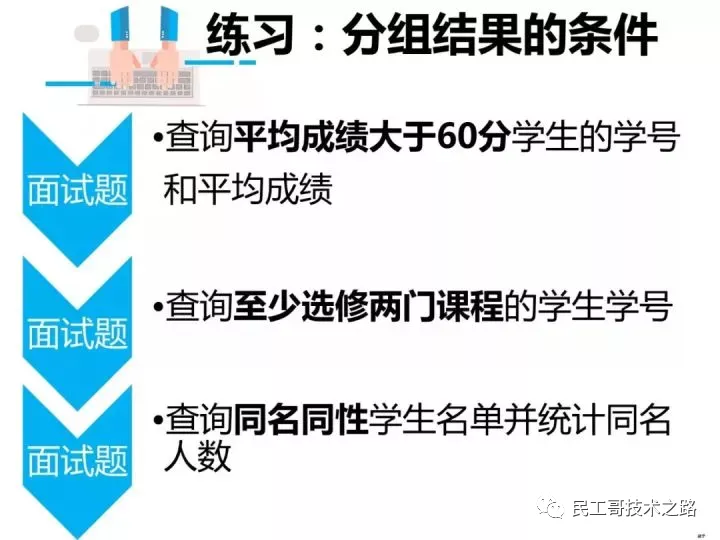
/\*  
分析思路  
select 查询结果 [课程ID：是课程号的别名,最高分：max(成绩) ,最低分：min(成绩)]  
from 从哪张表中查找数据 [成绩表score]  
where 查询条件 [没有]  
group by 分组 [各科成绩：也就是每门课程的成绩，需要按课程号分组];  
\*/  
**select** 课程号,**max**(成绩) **as** 最高分,**min**(成绩) **as** 最低分  
**from** score  
**group** **by** 课程号;

查询每门课程被选修的学生数

/\*  
分析思路  
select 查询结果 [课程号，选修该课程的学生数：汇总函数count]  
from 从哪张表中查找数据 [成绩表score]  
where 查询条件 [没有]  
group by 分组 [每门课程：按课程号分组];  
\*/  
**select** 课程号, **count**(学号)  
**from** score  
**group** **by** 课程号;

查询男生、女生人数

/\*  
分析思路  
select 查询结果 [性别，对应性别的人数：汇总函数count]  
from 从哪张表中查找数据 [性别在学生表中，所以查找的是学生表student]  
where 查询条件 [没有]  
group by 分组 [男生、女生人数：按性别分组]  
having 对分组结果指定条件 [没有]  
order by 对查询结果排序[没有];  
\*/  
**select** 性别,**count**(\*)  
**from** student  
**group** **by** 性别;



查询平均成绩大于60分学生的学号和平均成绩

/\*   
题目翻译成大白话：  
平均成绩：展开来说就是计算每个学生的平均成绩  
这里涉及到“每个”就是要分组了  
平均成绩大于60分，就是对分组结果指定条件  
分析思路  
select 查询结果 [学号，平均成绩：汇总函数avg(成绩)]  
from 从哪张表中查找数据 [成绩在成绩表中，所以查找的是成绩表score]  
where 查询条件 [没有]  
group by 分组 [平均成绩：先按学号分组，再计算平均成绩]  
having 对分组结果指定条件 [平均成绩大于60分]  
\*/  
**select** 学号, **avg**(成绩)  
**from** score  
**group** **by** 学号  
**having** **avg**(成绩)>60;

查询至少选修两门课程的学生学号

/\*   
翻译成大白话：  
第1步，需要先计算出每个学生选修的课程数据，需要按学号分组  
第2步，至少选修两门课程：也就是每个学生选修课程数目>=2，对分组结果指定条件  
分析思路  
select 查询结果 [学号,每个学生选修课程数目：汇总函数count]  
from 从哪张表中查找数据 [课程的学生学号：课程表score]  
where 查询条件 [至少选修两门课程：需要先计算出每个学生选修了多少门课，需要用分组，所以这里没有where子句]  
group by 分组 [每个学生选修课程数目：按课程号分组，然后用汇总函数count计算出选修了多少门课]  
having 对分组结果指定条件 [至少选修两门课程：每个学生选修课程数目>=2]  
\*/  
**select** 学号, **count**(课程号) **as** 选修课程数目  
**from** score  
**group** **by** 学号  
**having** **count**(课程号)>=2;

查询同名同性学生名单并统计同名人数

/\*   
翻译成大白话，问题解析：  
1）查找出姓名相同的学生有谁，每个姓名相同学生的人数  
查询结果：姓名,人数  
条件：怎么算姓名相同？按姓名分组后人数大于等于2，因为同名的人数大于等于2  
  
分析思路  
select 查询结果 [姓名,人数：汇总函数count(\*)]  
from 从哪张表中查找数据 [学生表student]  
where 查询条件 [没有]  
group by 分组 [姓名相同：按姓名分组]  
having 对分组结果指定条件 [姓名相同：count(\*)>=2]  
order by 对查询结果排序[没有];  
\*/  
   
**select** 姓名,**count**(\*) **as** 人数  
**from** student  
**group** **by** 姓名  
**having** **count**(\*)>=2;

查询不及格的课程并按课程号从大到小排列

/\*   
分析思路  
select 查询结果 [课程号]  
from 从哪张表中查找数据 [成绩表score]  
where 查询条件 [不及格：成绩 <60]  
group by 分组 [没有]  
having 对分组结果指定条件 [没有]  
order by 对查询结果排序[课程号从大到小排列：降序desc];  
\*/  
**select** 课程号  
**from** score   
**where** 成绩<60  
**order** **by** 课程号 **desc**;

查询每门课程的平均成绩，结果按平均成绩升序排序，平均成绩相同时，按课程号降序排列

/\*   
分析思路  
select 查询结果 [课程号,平均成绩：汇总函数avg(成绩)]  
from 从哪张表中查找数据 [成绩表score]  
where 查询条件 [没有]  
group by 分组 [每门课程：按课程号分组]  
having 对分组结果指定条件 [没有]  
order by 对查询结果排序[按平均成绩升序排序:asc，平均成绩相同时，按课程号降序排列:desc];  
\*/  
**select** 课程号, **avg**(成绩) **as** 平均成绩  
**from** score  
**group** **by** 课程号  
**order** **by** 平均成绩 **asc**,课程号 **desc**;

检索课程编号为“0004”且分数小于60的学生学号，结果按按分数降序排列

/\*   
分析思路  
select 查询结果 []  
from 从哪张表中查找数据 [成绩表score]  
where 查询条件 [课程编号为“04”且分数小于60]  
group by 分组 [没有]  
having 对分组结果指定条件 []  
order by 对查询结果排序[查询结果按按分数降序排列];  
\*/  
**select** 学号  
**from** score  
**where** 课程号='04' **and** 成绩 <60  
**order** **by** 成绩 **desc**;

统计每门课程的学生选修人数(超过2人的课程才统计)

要求输出课程号和选修人数，查询结果按人数降序排序，若人数相同，按课程号升序排序

/\*   
分析思路  
select 查询结果 [要求输出课程号和选修人数]  
from 从哪张表中查找数据 []  
where 查询条件 []  
group by 分组 [每门课程：按课程号分组]  
having 对分组结果指定条件 [学生选修人数(超过2人的课程才统计)：每门课程学生人数>2]  
order by 对查询结果排序[查询结果按人数降序排序，若人数相同，按课程号升序排序];  
\*/  
**select** 课程号, **count**(学号) **as** '选修人数'  
**from** score  
**group** **by** 课程号  
**having** **count**(学号)>2  
**order** **by** **count**(学号) **desc**,课程号 **asc**;

查询两门以上不及格课程的同学的学号及其平均成绩

/\*  
分析思路  
先分解题目：  
1）[两门以上][不及格课程]限制条件  
2）[同学的学号及其平均成绩]，也就是每个学生的平均成绩，显示学号，平均成绩  
分析过程：  
第1步：得到每个学生的平均成绩，显示学号，平均成绩  
第2步：再加上限制条件：  
1）不及格课程  
2）两门以上[不及格课程]：课程数目>2  
   
   
/\*   
第1步：得到每个学生的平均成绩，显示学号，平均成绩  
select 查询结果 [学号,平均成绩：汇总函数avg(成绩)]  
from 从哪张表中查找数据 [涉及到成绩：成绩表score]  
where 查询条件 [没有]  
group by 分组 [每个学生的平均：按学号分组]  
having 对分组结果指定条件 [没有]  
order by 对查询结果排序[没有];  
\*/  
**select** 学号, **avg**(成绩) **as** 平均成绩  
**from** score  
**group** **by** 学号;  
   
   
/\*   
第2步：再加上限制条件：  
1）不及格课程  
2）两门以上[不及格课程]  
select 查询结果 [学号,平均成绩：汇总函数avg(成绩)]  
from 从哪张表中查找数据 [涉及到成绩：成绩表score]  
where 查询条件 [限制条件：不及格课程，平均成绩<60]  
group by 分组 [每个学生的平均：按学号分组]  
having 对分组结果指定条件 [限制条件：课程数目>2,汇总函数count(课程号)>2]  
order by 对查询结果排序[没有];  
\*/  
**select** 学号, **avg**(成绩) **as** 平均成绩  
**from** score  
**where** 成绩 <60  
**group** **by** 学号  
**having** **count**(课程号)>=2;

如果上面题目不会做，可以复习这部分涉及到的sql知识：

###### **3.复杂查询**

查询所有课程成绩小于60分学生的学号、姓名

【知识点】子查询

1.翻译成大白话

1）查询结果：学生学号，姓名 2）查询条件：所有课程成绩 < 60 的学生，需要从成绩表里查找，用到子查询

第1步，写子查询（所有课程成绩 < 60 的学生）

**select** 查询结果[学号]  
**from** 从哪张表中查找数据[成绩表：score]  
**where** 查询条件[成绩 < 60]  
**group** **by** 分组[没有]  
**having** 对分组结果指定条件[没有]  
**order** **by** 对查询结果排序[没有]  
**limit** 从查询结果中取出指定行[没有];  
**select** 学号   
**from** score  
**where** 成绩 < 60;

第2步，查询结果：学生学号，姓名，条件是前面1步查到的学号

**select** 查询结果[学号,姓名]  
**from** 从哪张表中查找数据[学生表:student]  
**where** 查询条件[用到运算符**in**]  
**group** **by** 分组[没有]  
**having** 对分组结果指定条件[没有]  
**order** **by** 对查询结果排序[没有]  
**limit** 从查询结果中取出指定行[没有];  
**select** 学号,姓名  
**from** student  
**where**  学号 **in** (  
**select** 学号   
**from** score  
**where** 成绩 < 60  
);

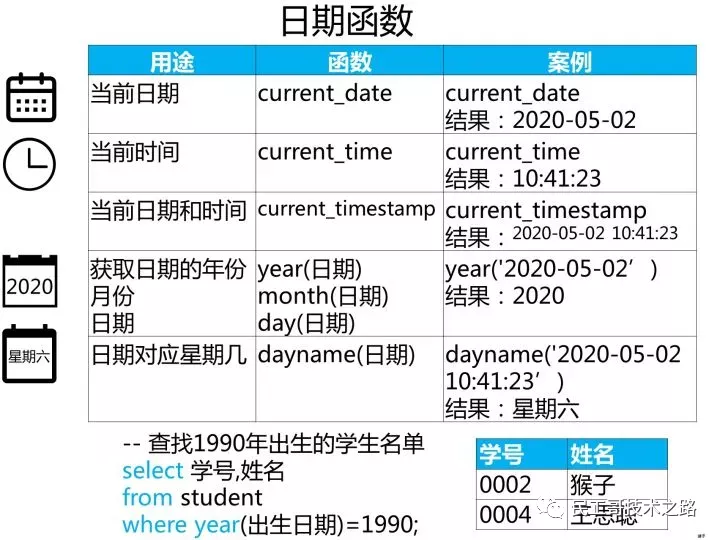
查询没有学全所有课的学生的学号、姓名

/\*  
查找出学号，条件：没有学全所有课，也就是该学生选修的课程数 < 总的课程数  
【考察知识点】in，子查询  
\*/  
**select** 学号,姓名  
**from** student  
**where** 学号 **in**(  
**select** 学号   
**from** score  
**group** **by** 学号  
**having** **count**(课程号) < (**select** **count**(课程号) **from** course)  
);

查询出只选修了两门课程的全部学生的学号和姓名

**select** 学号,姓名  
**from** student  
**where** 学号 **in**(  
**select** 学号  
**from** score  
**group** **by** 学号  
**having** **count**(课程号)=2  
);

1990年出生的学生名单



/\*  
查找1990年出生的学生名单  
学生表中出生日期列的类型是datetime  
\*/  
**select** 学号,姓名   
**from** student   
**where** **year**(出生日期)=1990;

查询各科成绩前两名的记录

这类问题其实就是常见的：分组取每组最大值、最小值，每组最大的N条（top N）记录。

## **sql面试题：topN问题**

工作中会经常遇到这样的业务问题：

* 如何找到每个类别下用户最喜欢的产品是哪个？
* 如果找到每个类别下用户点击最多的5个商品是什么？

这类问题其实就是常见的：分组取每组最大值、最小值，每组最大的N条（top N）记录。

面对该类问题，如何解决呢？

下面我们通过成绩表的例子来给出答案。

成绩表是学生的成绩，里面有学号（学生的学号），课程号（学生选修课程的课程号），成绩（学生选修该课程取得的成绩）



###### **分组取每组最大值**

案例：按课程号分组取成绩最大值所在行的数据

我们可以使用分组（group by）和汇总函数得到每个组里的一个值（最大值，最小值，平均值等）。但是无法得到成绩最大值所在行的数据。

**select** 课程号,**max**(成绩) **as** 最大成绩  
**from** score   
**group** **by** 课程号;



我们可以使用关联子查询来实现：

**select** \*   
**from** score **as** a   
**where** 成绩 = (  
**select** **max**(成绩)   
**from** score **as** b   
**where** b.课程号 = a.课程号);



上面查询结果课程号“0001”有2行数据，是因为最大成绩80有2个

###### **分组取每组最小值**

案例：按课程号分组取成绩最小值所在行的数据

同样的使用关联子查询来实现

**select** \*   
**from** score **as** a   
**where** 成绩 = (  
**select** **min**(成绩)   
**from** score **as** b   
**where** b.课程号 = a.课程号);



###### **每组最大的N条记录**

案例：查询各科成绩前两名的记录

第1步，查出有哪些组

我们可以按课程号分组，查询出有哪些组，对应这个问题里就是有哪些课程号

**select** 课程号,**max**(成绩) **as** 最大成绩  
**from** score   
**group** **by** 课程号;



第2步：先使用order by子句按成绩降序排序（desc），然后使用limt子句返回topN（对应这个问题返回的成绩前两名）

-- 课程号'0001' 这一组里成绩前2名  
**select** \*   
**from** score   
**where** 课程号 = '0001'   
**order** **by** 成绩  **desc**   
**limit** 2;

同样的，可以写出其他组的（其他课程号）取出成绩前2名的sql

第3步，使用union all 将每组选出的数据合并到一起

-- 左右滑动可以可拿到全部sql  
(**select** \* **from** score **where** 课程号 = '0001' **order** **by** 成绩  **desc** **limit** 2)  
**union** **all**  
(**select** \* **from** score **where** 课程号 = '0002' **order** **by** 成绩  **desc** **limit** 2)  
**union** **all**  
(**select** \* **from** score **where** 课程号 = '0003' **order** **by** 成绩  **desc** **limit** 2);

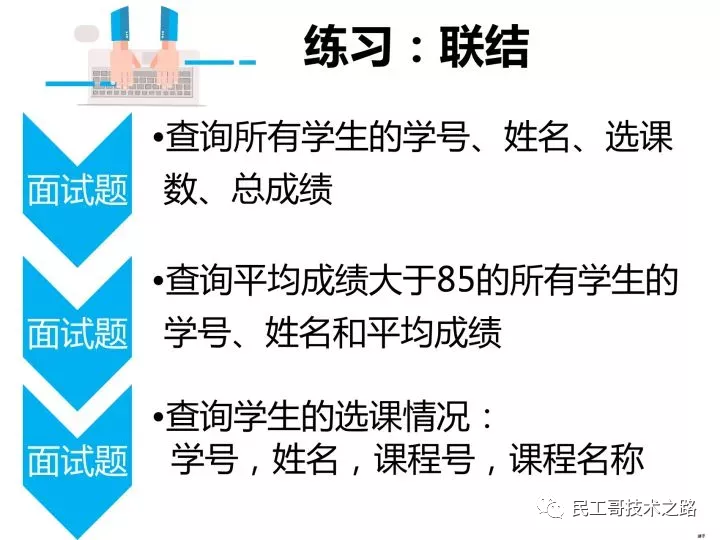


前面我们使用order by子句按某个列降序排序（desc）得到的是每组最大的N个记录。如果想要达到每组最小的N个记录，将order by子句按某个列升序排序（asc）即可。

###### **总结**

常见面试题：分组取每组最大值、最小值，每组最大的N条（top N）记录。

## **多表查询**



查询所有学生的学号、姓名、选课数、总成绩

**select** a.学号,a.姓名,**count**(b.课程号) **as** 选课数,**sum**(b.成绩) **as** 总成绩  
**from** student **as** a **left** **join** score **as** b  
**on** a.学号 = b.学号  
**group** **by** a.学号;

查询平均成绩大于85的所有学生的学号、姓名和平均成绩

**select** a.学号,a.姓名, **avg**(b.成绩) **as** 平均成绩  
**from** student **as** a **left** **join** score **as** b  
**on** a.学号 = b.学号  
**group** **by** a.学号  
**having** **avg**(b.成绩)>85;

查询学生的选课情况：学号，姓名，课程号，课程名称

**select** a.学号, a.姓名, c.课程号,c.课程名称  
**from** student a **inner** **join** score b **on** a.学号=b.学号  
**inner** **join** course c **on** b.课程号=c.课程号;

查询出每门课程的及格人数和不及格人数

-- 考察case表达式  
**select** 课程号,  
**sum**(**case** **when** 成绩>=60 **then** 1   
  **else** 0   
    **end**) **as** 及格人数,  
**sum**(**case** **when** 成绩 <  60 **then** 1   
  **else** 0   
    **end**) **as** 不及格人数  
**from** score  
**group** **by** 课程号;

使用分段[100-85],[85-70],[70-60],[<60]来统计各科成绩，分别统计：各分数段人数，课程号和课程名称

-- 考察case表达式  
**select** a.课程号,b.课程名称,  
**sum**(**case** **when** 成绩 **between** 85 **and** 100   
  **then** 1 **else** 0 **end**) **as** '[100-85]',  
**sum**(**case** **when** 成绩 >=70 **and** 成绩<85   
  **then** 1 **else** 0 **end**) **as** '[85-70]',  
**sum**(**case** **when** 成绩>=60 **and** 成绩<70    
  **then** 1 **else** 0 **end**) **as** '[70-60]',  
**sum**(**case** **when** 成绩<60 **then** 1 **else** 0 **end**) **as** '[<60]'  
**from** score **as** a **right** **join** course **as** b   
**on** a.课程号=b.课程号  
**group** **by** a.课程号,b.课程名称;

查询课程编号为0003且课程成绩在80分以上的学生的学号和姓名|

**select** a.学号,a.姓名  
**from** student  **as** a **inner** **join** score **as** b **on** a.学号=b.学号  
**where** b.课程号='0003' **and** b.成绩>80;

下面是学生的成绩表（表名score，列名：学号、课程号、成绩）



使用sql实现将该表行转列为下面的表结构



【面试题类型总结】这类题目属于行列如何互换，解题思路如下：

【面试题】下面是学生的成绩表（表名score，列名：学号、课程号、成绩）



使用sql实现将该表行转列为下面的表结构



【解答】

第1步，使用常量列输出目标表的结构

可以看到查询结果已经和目标表非常接近了

**select** 学号,'课程号0001','课程号0002','课程号0003'  
**from** score;



第2步，使用case表达式，替换常量列为对应的成绩

select 学号,  
(**case** 课程号 when '0001' **then** 成绩 **else** 0 end) as '课程号0001',  
(**case** 课程号 when '0002' **then** 成绩 **else** 0 end) as  '课程号0002',  
(**case** 课程号 when '0003' **then** 成绩 **else** 0 end) as '课程号0003'  
from score;



在这个查询结果中，每一行表示了某个学生某一门课程的成绩。比如第一行是'学号0001'选修'课程号00001'的成绩，而其他两列的'课程号0002'和'课程号0003'成绩为0。

每个学生选修某门课程的成绩在下图的每个方块内。我们可以通过分组，取出每门课程的成绩。

第3关，分组

分组，并使用最大值函数max取出上图每个方块里的最大值

**select** 学号,  
**max**(**case** 课程号 **when** '0001' **then** 成绩 **else** 0 **end**) **as** '课程号0001',  
**max**(**case** 课程号 **when** '0002' **then** 成绩 **else** 0 **end**) **as** '课程号0002',  
**max**(**case** 课程号 **when** '0003' **then** 成绩 **else** 0 **end**) **as** '课程号0003'  
**from** score  
**group** **by** 学号;

这样我们就得到了目标表（行列互换）

