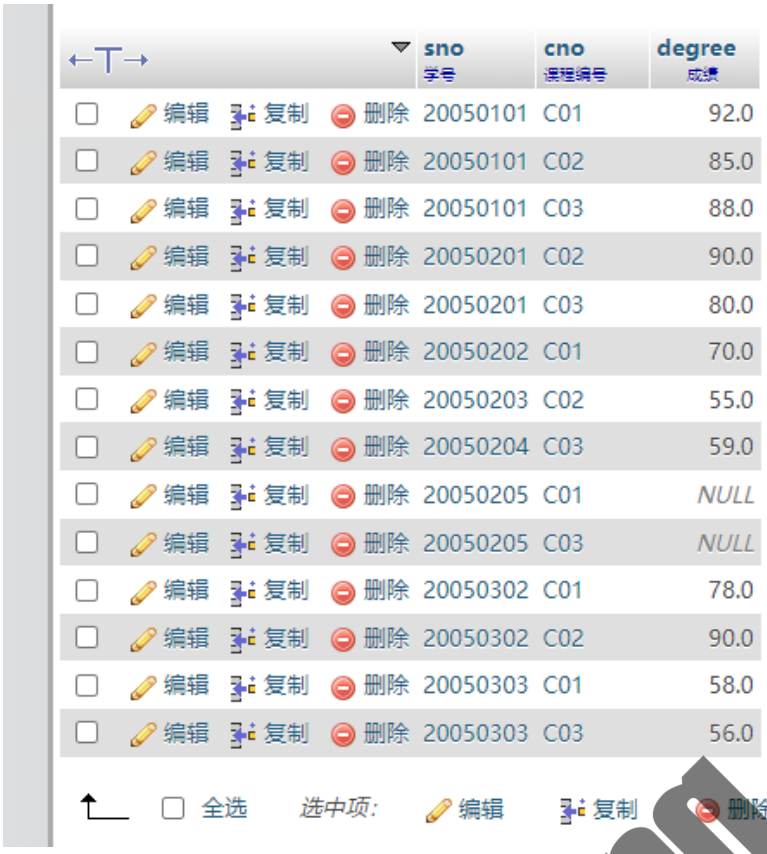
MySQL 笔记 - 来自实训8

基础 - 语法构成

数据库结构：数据库test，数据表sc。字段：sno，cno，degree(学号，课程编号，成绩)

```
-------------------------------------------------------------
CREATE DATABASE test;
-------------------------------------------------------------
DROP TABLE IF EXISTS `sc`;
CREATE TABLE `sc` (
  `sno` char(10) NOT NULL COMMENT '学号',
  `cno` char(5) NOT NULL COMMENT '课程编号',
  `degree` decimal(4,1) DEFAULT NULL COMMENT '成绩',
  PRIMARY KEY (`sno`,`cno`),
  KEY `cno` (`cno`),
  CONSTRAINT `sc_ibfk_1` FOREIGN KEY (`sno`) REFERENCES `student` (`sno`),
  CONSTRAINT `sc_ibfk_2` FOREIGN KEY (`cno`) REFERENCES `course` (`cno`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312;
-------------------------------------------------------------
INSERT INTO `sc` VALUES ('20050101', 'C01', '92.0');
INSERT INTO `sc` VALUES ('20050101', 'C02', '85.0');
INSERT INTO `sc` VALUES ('20050101', 'C03', '88.0');
INSERT INTO `sc` VALUES ('20050201', 'C02', '90.0');
INSERT INTO `sc` VALUES ('20050201', 'C03', '80.0');
INSERT INTO `sc` VALUES ('20050202', 'C01', '70.0');
INSERT INTO `sc` VALUES ('20050203', 'C02', '55.0');
INSERT INTO `sc` VALUES ('20050204', 'C03', '59.0');
INSERT INTO `sc` VALUES ('20050205', 'C01', null);
INSERT INTO `sc` VALUES ('20050205', 'C03', null);
INSERT INTO `sc` VALUES ('20050302', 'C01', '78.0');
INSERT INTO `sc` VALUES ('20050302', 'C02', '90.0');
INSERT INTO `sc` VALUES ('20050303', 'C01', '58.0');
INSERT INTO `sc` VALUES ('20050303', 'C03', '56.0');
-------------------------------------------------------------
```

得到



	sno 学号	cno 课程编号	degree 成绩
编辑 复制 删除	20050101	C01	92.0
编辑 复制 删除	20050101	C02	85.0
编辑 复制 删除	20050101	C03	88.0
编辑 复制 删除	20050201	C02	90.0
编辑 复制 删除	20050201	C03	80.0
编辑 复制 删除	20050202	C01	70.0
编辑 复制 删除	20050203	C02	55.0
编辑 复制 删除	20050204	C03	59.0
编辑 复制 删除	20050205	C01	NULL
编辑 复制 删除	20050205	C03	NULL
编辑 复制 删除	20050302	C01	78.0
编辑 复制 删除	20050302	C02	90.0
编辑 复制 删除	20050303	C01	58.0
编辑 复制 删除	20050303	C03	56.0

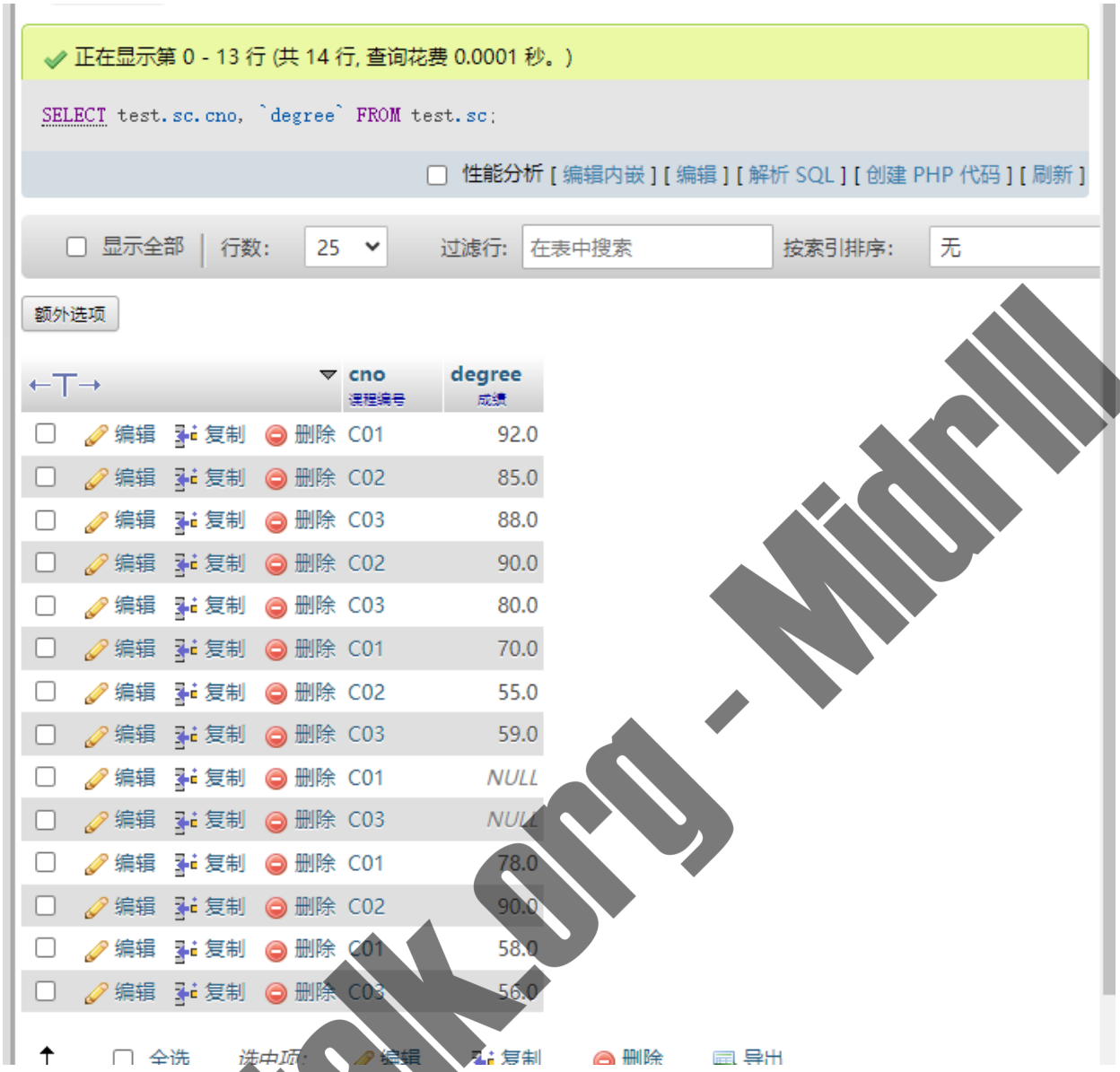
全选 选中项: 编辑 复制 删除 导出

```
SELECT * FROM `sc`
# 执行如上代码即可获得图中效果，上面代码意味着，在数据表sc中查找，显示所有字段(sno，cno，degree)
```


```
SELECT `cno` FROM `sc`
# 执行如上代码即可获得图中效果，上面代码意味着，在数据表sc中查找，但只显示所有字段cno。

# 等价于
SELECT `cno` FROM test.sc
# 而该处test，直接指明了哪个数据库，而sc则是指数据库内某个表。
```

有如下的几种写法



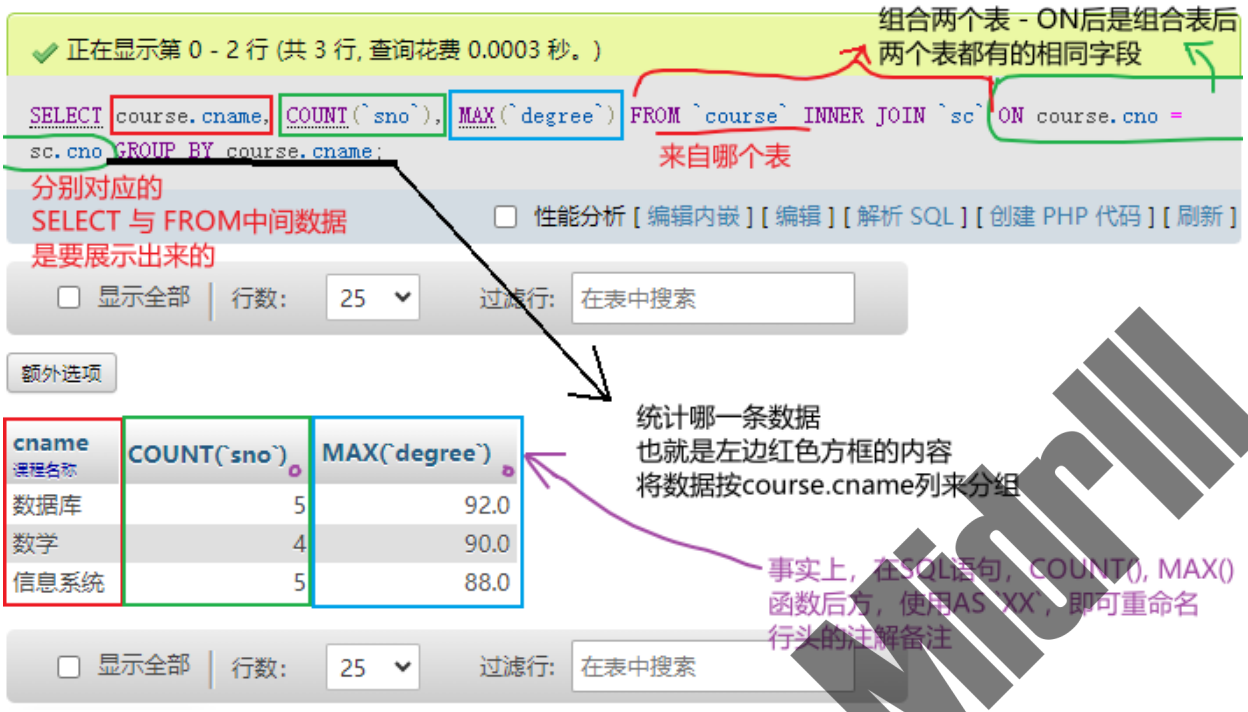
统计 每门课程 的 选课人数 和 最高分 。

来自下方 - 2.单表分组查询与排序查询 - (5)统计每门课程的选课人数和最高分

深入解释

```
SELECT course.cname, COUNT(`sno`), MAX(`degree`)
FROM `course` INNER JOIN `sc` ON course.cno = sc.cno
GROUP BY course.cname;

# 使用cno记得加上【表名.cno】(比如说统计GROUP BY指令)，因为组合数据后，有两个cno，你直接输cno会报错，提示匹配模糊。因为MySQL根本不知道你到底要统计哪个表的cno。
```



单表无条件和有条件查询

1. 查询“C01”课程的开课学期。

```
use test;
SELECT * FROM teaching WHERE con = ‘C01’;
```

```
mysql> SELECT * FROM teaching WHERE cno = ’C01’;
+------+-----+-------+
| cno  | tno | cterm |
+------+-----+-------+
| C01  | 101 |     2 |
+------+-----+-------+
1 row in set (0.00 sec)
```

2. 查询成绩在80～90分之间的学生学号及课号。

```
show tables;
SELECT * FROM sc;
SELECT * FROM sc WHERE degree >=80 AND <= 90;
```

```
mysql> SELECT * FROM sc WHERE degree >= 80 AND degree <= 90;
+----------+-----+--------+
| sno      | cno | degree |
+----------+-----+--------+
| 20050101 | C02 |   85.0 |
| 20050101 | C03 |   88.0 |
| 20050201 | C02 |   90.0 |
| 20050201 | C03 |   80.0 |
| 20050302 | C02 |   90.0 |
+----------+-----+--------+
5 rows in set (0.00 sec)
```

3. 查询在1970年1月1日之前出生的男教师信息。

```
SELECT * FROM teacher WHERE tbirthday > 1970-01-01;
```

```
mysql> SELECT * FROM teacher WHERE tbirthday > 1970-01-01;
+-----+--------+------+------------+--------------+
| tno | tname  | tsex | tbirthday  | tdept        |
+-----+--------+------+------------+--------------+
| 101 | 李新   | 男   | 1977-01-12 | 计算机工程系 |
| 102 | 钱军   | 女   | 1968-06-04 | 计算机工程系 |
| 103 | 王楠   | 女   | 1983-05-04 | 软件系       |
| 105 | 赵凯   | 男   | 1982-05-03 | 体育系       |
| 201 | 王小花 | 女   | 1979-12-23 | 信息工程系   |
| 202 | 张小青 | 男   | 1968-08-25 | 信息工程系   |
| 204 | 刘刚   | 男   | 1990-02-01 | 财经系       |
+-----+--------+------+------------+--------------+
```

4. 输出有成绩的学生学号。

```
SELECT sno FROM sc WHERE degree IS NOT NULL;
```

```
mysql> SELECT sno FROM sc WHERE degree != NULL;
Empty set (0.00 sec)

mysql> SELECT sno FROM sc WHERE degree IS NOT NULL;
+----------+
| sno      |
+----------+
| 20050101 |
| 20050101 |
| 20050101 |
| 20050201 |
| 20050201 |
| 20050202 |
| 20050203 |
| 20050204 |
| 20050302 |
| 20050302 |
| 20050303 |
| 20050303 |
+----------+
12 rows in set (0.00 sec)
```

5. 查询所有姓“刘”的学生信息。

```
SELECT * FROM student WHERE sname LIKE '刘%';
```

```
mysql> SELECT * FROM student WHERE sname LIKE '刘%';
+----------+-------+------+---------------------+----------+-----------+-------
-----+----------+-------+
| sno      | sname | ssex | sbirthday           | saddress | sdept     | specia
lity | sclass   | tutor |
+----------+-------+------+---------------------+----------+-----------+-------
-----+----------+-------+
| 20050201 | 刘晨  | 女   | 1988-06-04 00:00:00 | 山东青岛 | 信息工程系 | 电子商
务   | 20070305 | 孙茂  |
+----------+-------+------+---------------------+----------+-----------+-------
-----+----------+-------+
1 row in set (0.00 sec)
```

6. 查询生源地不是山东省的学生信息。

```
SELECT * FROM student WHERE saddress NOT LIKE '山东%';
```

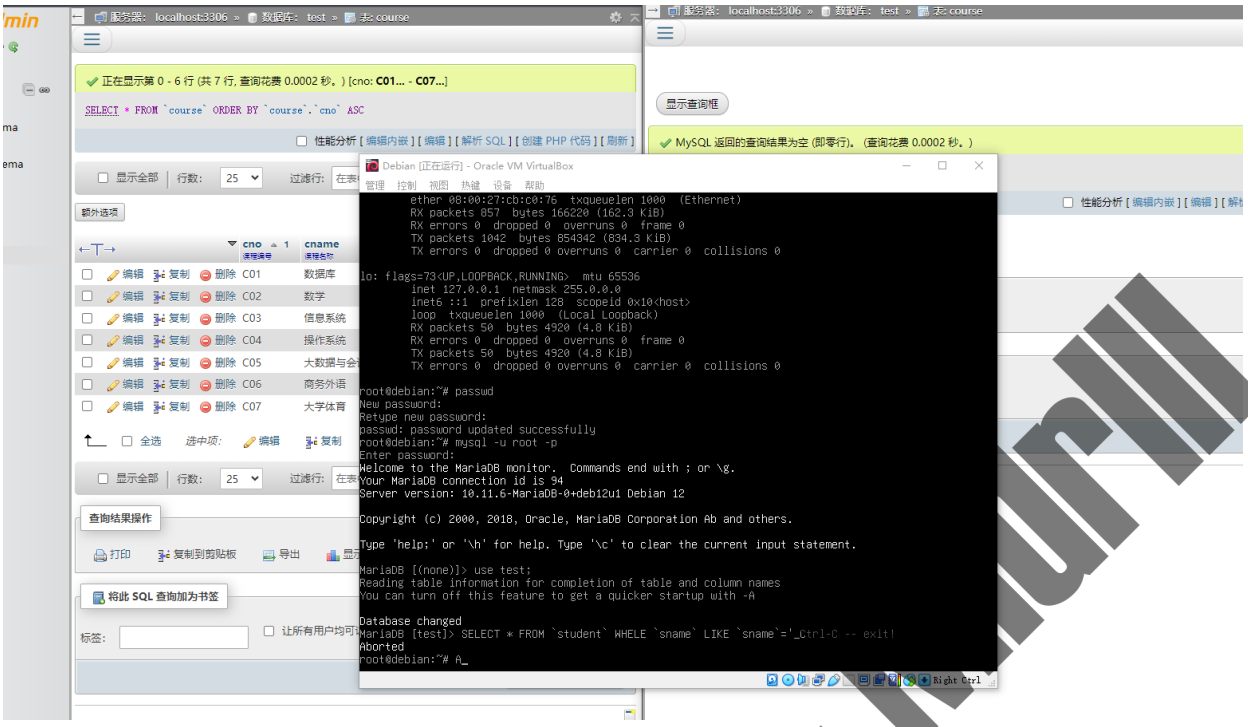
```
mysql> SELECT * FROM student WHERE saddress NOT LIKE '山东%';
+----------+--------+------+---------------------+----------+------------+------
------+----------+---------+
| sno      | sname  | ssex | sbirthday           | saddress | sdept      | speci
ality | sclass   | tutor  |
+----------+--------+------+---------------------+----------+------------+------
------+----------+---------+
| 20050102 | 赵欣欣 | 女   | 1987-08-31 00:00:00 | 浙江宁波 | 体育系     | 篮球
      | 20070302 | 刘小白 |
| 20050103 | 梁健   | 男   | 1988-09-01 00:00:00 | 广西桂林 | 外语系     | 商务
外语   | 20070303 | 邓刚   |
| 20050104 | 董小宛 | 女   | 1989-09-02 00:00:00 | 甘肃宁夏 | 艺术系     | 动漫
制作   | 20070304 | 吴慧   |
| 20050105 | 李志超 | 男   | 1987-09-03 00:00:00 | 四川成都 | 艺术系     | 广告
设计   | 20070304 | 吴慧   |
| 20050202 | 张立   | 男   | 1988-08-25 00:00:00 | 河北唐山 | 信息工程系 | 电子
商务   | 20070305 | 孙茂   |
| 20050203 | 王亮   | 男   | 1981-08-26 00:00:00 | 广东珠海 | 外语系     | 商务
外语   | 20070303 | 邓刚   |
| 20050204 | 孙悦   | 女   | 1986-08-27 00:00:00 | 山西临汾 | 软件系     | 软件
工程   | 20070306 | 张楠   |
| 20050301 | 王敏   | 女   | 1989-12-23 00:00:00 | 江苏苏州 | 数学系     | 数学
      | 20070308 | 杨启动 |
| 20050302 | 李丽丽 | 女   | 1985-08-29 00:00:00 | 河北天津 | 软件系     | 信息
安全   | 20070309 | 吴超   |
| 20050303 | 王力杰 | 男   | 1990-08-30 00:00:00 | 辽宁大连 | 财经系     | 会计
      | 20070310 | 钱坤   |
+----------+--------+------+---------------------+----------+------------+------
------+----------+---------+
10 rows in set (0.00 sec)
```

7. 查询成绩为79分、89分或99分的记录。

```
SELECT * FROM sc WHERE degree = 79 OR degree =80 OR degree = 99;
```

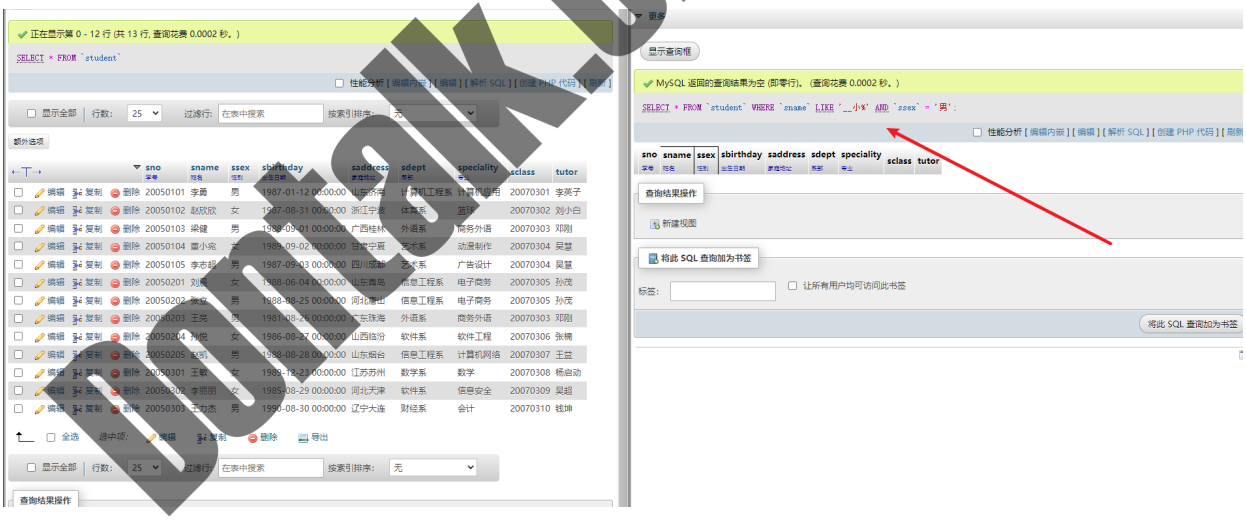
```
mysql> SELECT * FROM sc WHERE degree = 79 OR degree =80 OR degree = 99;
+----------+-----+--------+
| sno      | cno | degree |
+----------+-----+--------+
| 20050201 | C03 |   80.0 |
+----------+-----+--------+
1 row in set (0.00 sec)
```

下面因为环境问题，不得不使用PHPMyAdmin，但命令依旧是查看手册后人工输入的。



8. 查询名字中第二个字是“小”字的男生的学生姓名和地址。

```
SELECT * FROM `student` WHERE `sname` LIKE '__小%' AND `ssex` = '男';
```

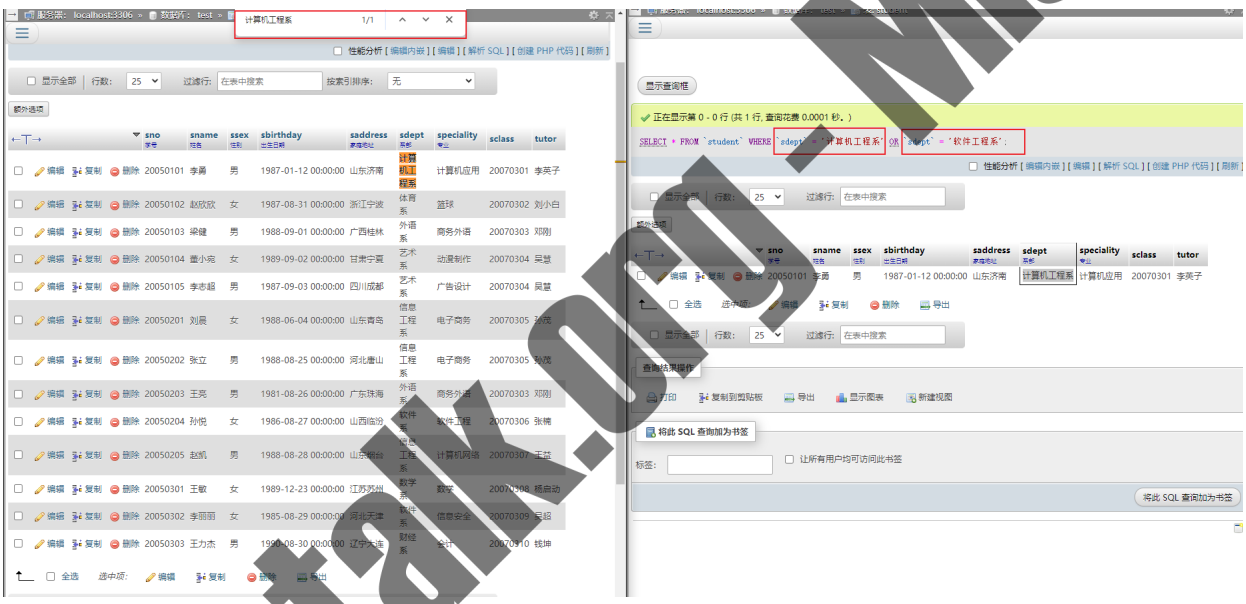


(9) 查询名称以“计算机_”开头的课程名称。

```
SELECT * FROM `course` WHERE `cname` LIKE '计算机_';
```

(10) 查询计算机工程系和软件工程系的学生信息。

```
SELECT * FROM `student` WHERE `sdept` = '计算机工程系' OR `sdept` = '软件工程系'
```



单表分组查询与排序查询

(1) 统计有学生选修的课程的门数。

笔记

当你使用以下的代码进行查询时：

```
SELECT column1, aggregate_function(column2)
FROM table
GROUP BY column1;
```

这个代码用于进行单表的分组查询操作。

- `column1` 是你想要分组的列名。可以是一个或多个列，用逗号分隔。按照这些列的值进行分组。
- `aggregate_function` 是用于计算聚合值的函数。常见的聚合函数有 `COUNT`（计数）、`SUM`（求和）、`AVG`（平均值）、`MAX`（最大值）和 `MIN`（最小值），你可以根据需要选择合适的函数。
- `table` 是你要查询数据的表名。在这个表中进行分组查询。

执行以上代码，将会按照 `column1` 的值对数据进行分组，并针对每个分组计算 `column2` 的聚合值。结果集中将包含分组列以及相应的聚合结果。
例如，假设你有一个 "学生" 表，其中包含学生的姓名和他们所属的班级信息。你希望按照班级对学生进行分组，并计算每个班级的学生人数。你可以使用以下的代码：

```
SELECT class, COUNT(*) as student_count
FROM student
GROUP BY class;
```

这段代码将会返回每个班级的学生人数统计。

INNER JOIN具体用法和用途
当你想要从两个表中检索相关联的数据时，可以使用 INNER JOIN。INNER JOIN 用于根据两个表之间的共同列（通常是外键）将它们连接起来。

INNER JOIN 的基本语法如下：

```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

在这个语法中，`table1` 和 `table2` 是你要连接的两个表，`column_name` 是两个表中的共同列名。通过指定 `ON` 子句后面的条件，你可以指定连接所需的列。

INNER JOIN 返回两个表中满足连接条件的匹配行。如果表1中的列值在表2中没有对应的匹配项，那么这些行不会包含在结果集中。

INNER JOIN 可以帮助你获取两个表之间具有关联的数据，比如学生和选课表、订单和产品表等等。通过将相关的数据合并在一起，你可以轻松地进行复杂的查询和分析。

实现

```
SELECT student.sno
FROM `student`
INNER JOIN `sc`
ON student.sno = sc.sno;
```

我们加个函数来统计学号多少，来知道他们修了多少门课

```
SELECT student.sno, COUNT(*) AS count
FROM `student`
INNER JOIN `sc` ON student.sno = sc.sno
GROUP BY student.sno;
```

顺便列出姓名

```
SELECT student.sno, student.sname, COUNT(*) AS count
FROM `student`
INNER JOIN `sc` ON student.sno = sc.sno
GROUP BY student.sno, student.sname;
```

显示查询框

当前所选内容没有包含唯一字段。单元格编辑、复选框、编辑、复制和删除无法正常使用。

正在显示第 0 - 7 行 (共 8 行, 查询花费 0.0007 秒。)

```
SELECT student.sno, student.sname, COUNT(*) AS count FROM `student` INNER JOIN `sc` ON student.sno = sc.sno GROUP BY
student.sno, student.sname;
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索

额外选项

sno 学号	sname 姓名	count
20050101	李勇	3
20050201	刘晨	2
20050202	张立	1
20050203	王亮	1
20050204	孙悦	1
20050205	赵凯	2
20050302	李丽丽	2
20050303	王力杰	2

(2) 计算“c01”课程的平均成绩。

```
SELECT `cno`, AVG(`degree`) AS average_grade FROM `sc` GROUP BY `cno`;
```

显示查询框

正在显示第 0 - 2 行 (共 3 行, 查询花费 0.0001 秒。)

```
SELECT `cno`, AVG(`degree`) AS average_grade FROM `sc` GROUP BY `cno`;
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索

额外选项

	cno 课程编号	average_grade
编辑 复制 删除	C01	74.50000
编辑 复制 删除	C02	80.00000
编辑 复制 删除	C03	70.75000

全选 选中项: 编辑 复制 删除 导出

笔记

1. `SELECT cno, AVG(degree) AS average_grade`: 这部分指令表示你要从表 `sc` 中选择两列数据。第一列是 `cno`，即课程号，第二列是 `degree`，即学生成绩。然后通过使用 AVG() 函数计算每门课程的平均成绩，并将结果命名为 `average_grade`。
2. `FROM sc`: 这是指定从哪个表中选择数据，即在表 `sc` 中进行选择和计算。
3. `GROUP BY cno`: 通过 GROUP BY 子句，你在查询中指定了按照 `cno` 列进行分组。这意味着每个不同的课程号都会被视为一个单独的组，在每个组内计算其对应学生成绩的平均值。

因此，当你执行这个查询时，它会返回每门课程的平均成绩，以及对应的课程号（`cno`）。

(3) 查询选修了“c03”课程的学生的学号及其成绩，查询结果按分数降序排列。

```
SELECT * FROM `sc` WHERE `cno` = 'C03' ORDER BY `degree` DESC;
```

(4) 查询各个课程号及相应的选课人数。

```
SELECT `cno`, COUNT(`sno`) FROM `sc` GROUP BY `cno`;
```

笔记

当你执行这个查询语句时，它会完成以下操作：

1. `SELECT cno, COUNT(sno)`：这部分指令表示你要从表 `sc` 中选择两列数据。第一列是 `cno`，即课程号，第二列是 `sno`，即学生号。然后使用 COUNT() 函数统计每个课程号对应的学生号数量。
2. `FROM sc`：这是指定从哪个表中选择数据，即在表 `sc` 中进行选择和计算。
3. `GROUP BY cno`：通过 GROUP BY 子句，你在查询中指定了按照 `cno` 列进行分组。这意味着每个不同的课程号都会被视为一个单独的组，在每个组内统计相应的学生号数量。

因此，当你执行这个查询时，它会返回每个课程号以及对应的选课人数。

(5) 统计每门课程的选课人数和最高分。

```
SELECT * FROM `course` INNER JOIN `sc` ON course.cno = sc.cno;
```

报错预览

```
SELECT `cno`, COUNT(`sno`) FROM `course` INNER JOIN `sc` ON course.cno =
sc.cno GROUP BY `cno`;
# 报错
SELECT `cno`, COUNT(`sno`) FROM `course` INNER JOIN `sc` ON course.cno =
sc.cno GROUP BY `cno` LIMIT 0, 25

`#1052 - 列名 'cno' 在 field list 定义模糊`

# 连接多个表时，如果这些表中存在相同的列名，你需要明确地指定每个列的来源
```

通过命令

```
SELECT sc.cno, COUNT(`sno`) FROM `course` INNER JOIN `sc` ON course.cno =
sc.cno GROUP BY sc.cno;
```

我们得到了每个课程编号都有多少人报名了，如上（4）。
我们改一下命令：得到每堂课有多少人

```
SELECT course.cname, COUNT(`sno`) FROM `course` INNER JOIN `sc` ON
course.cno = sc.cno GROUP BY course.cname;
```

最终效果：

```
SELECT course.cname, COUNT(`sno`), MAX(`degree`) FROM `course` INNER JOIN
`sc` ON course.cno = sc.cno GROUP BY course.cname;
```

(6) 统计每个学生的选课门数和考试总成绩，并按选课门数降序排列。

```
SELECT student.sname, COUNT(sc.cno),SUM(sc.degree) FROM student INNER JOIN
sc ON student.sno = sc.sno GROUP BY student.sno;

# 等价于，但一般，student.sname对应它自己的表，也就是student.sno。具体看需求。

SELECT student.sname, COUNT(sc.cno),SUM(sc.degree) FROM student INNER JOIN
sc ON student.sno = sc.sno GROUP BY sc.sno;
```

(7) 查询选修了3门以上课程的学生学号。

错误示范

```
SELECT student.sno FROM student INNER JOIN sc ON student.sno = sc.sno WHERE
COUNT(sc.sno)>3;
```

> SQL 中的 WHERE 子句无法直接使用聚合函数（如 COUNT）来过滤结果。
> 相反，可以使用 HAVING 子句来实现此目的。HAVING 子句用于在对聚合结果进行筛选。

```
SELECT student.sno FROM student INNER JOIN sc ON student.sno = sc.sno GROUP
BY student.sno HAVING COUNT(sc.sno)>3;
```

但是没有一条数据，这是错了吗

MySQL 返回的查询结果为空 (即零行)。 (查询花费 0.0004 秒。)

```
SELECT student.sno FROM student INNER JOIN sc ON student.sno = sc.sno GROUP BY student.sno
HAVING COUNT(sc.sno)>3;
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

sno
学号

当然不

正在显示第 0 - 13 行 (共 14 行, 查询花费 0.0003 秒。)

```
SELECT * FROM `sc`
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索 按索引排序: 无

额外选项

	sno 学号	cno 课程编号	degree 成绩	
编辑 复制 删除	20050101	C01	92.0	3
编辑 复制 删除	20050101	C02	85.0	
编辑 复制 删除	20050101	C03	88.0	
编辑 复制 删除	20050201	C02	90.0	2
编辑 复制 删除	20050201	C03	80.0	
编辑 复制 删除	20050202	C01	70.0	1
编辑 复制 删除	20050203	C02	55.0	1
编辑 复制 删除	20050204	C03	59.0	1
编辑 复制 删除	20050205	C01	NULL	2
编辑 复制 删除	20050205	C03	NULL	
编辑 复制 删除	20050302	C01	78.0	2
编辑 复制 删除	20050302	C02	90.0	
编辑 复制 删除	20050303	C01	58.0	2
编辑 复制 删除	20050303	C03	56.0	

全选 选中项: 编辑 复制 删除 导出

所以，除非`HAVING COUNT(sc.sno)>3`等于3/等于2，才会出现数据。

(8) 查询成绩不及格的学生学号及课号，并按成绩降序排列。

```
SELECT sc.sno, sc.cno, sc.degree FROM sc WHERE sc.degree < 60 ORDER BY
sc.degree DESC;
```

(9) 查询至少选修一门课程的学生学号。

```
SELECT DISTINCT sc.sno FROM sc WHERE sc.degree != 'NULL';
```

`DISTINCT` **关键字会确保结果中不会有重复的学生学号**

(10) 统计输出各系学生的人数

```
SELECT student.sdept, COUNT(student.sdept) FROM student GROUP BY
student.sdept;
```

多表链接查询

(1) 查询计算机工程系女学生的学生学号、姓名及考试成绩。

错误示范，这是会报错的：怎么连起三个表呢？

```
SELECT student.sno, student.sname, sc.degree, course.cname FROM student
INNER JOIN sc ON student.sno = sc.sno WHERE student.ssex = '女';
#1054 - 未知列'course.cname'在'field list'
```

因为缺少了与课程表(course)的连接，修改后：就是那么朴实无华

```
SELECT student.sno, student.sname, sc.degree, course.cname
FROM student
INNER JOIN sc ON student.sno = sc.sno
INNER JOIN course ON sc.cno = course.cno
```

```
WHERE student.ssex = '女';

#也就是
SELECT student.sno, student.sname, sc.degree, course.cname FROM student
INNER JOIN sc ON student.sno = sc.sno INNER JOIN course ON sc.cno =
course.cno WHERE student.ssex = '女';
```

正在显示第 0 - 4 行 (共 5 行, 查询花费 0.0004 秒。)

```
SELECT student.sno, student.sname, sc.degree, course.cname FROM student INNER JOIN sc ON
student.sno = sc.sno INNER JOIN course ON sc.cno = course.cno WHERE student.ssex = '女';
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索 按索引排序: 无

额外选项

sno 学号	sname 姓名	degree	cname
20050302	李丽丽	78.0	数据库
20050201	刘晨	90.0	数学
20050302	李丽丽	90.0	数学
20050201	刘晨	80.0	信息系统
20050204	孙悦	59.0	信息系统

(2) 查询“李勇”同学所选课程的成绩。(不考虑重名)

```
SELECT student.sno, student.sname, sc.degree, course.cname FROM student
INNER JOIN sc ON student.sno = sc.sno INNER JOIN course ON sc.cno =
course.cno WHERE student.sname = '李勇';
```

正在显示第 0 - 2 行 (共 3 行, 查询花费 0.0003 秒。)

```
SELECT student.sno, student.sname, sc.degree, course.cname FROM student INNER JOIN sc ON
student.sno = sc.sno INNER JOIN course ON sc.cno = course.cno WHERE student.sname = '李勇';
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索 按索引排序: 无

额外选项

sno 学号	sname 姓名	degree	cname
20050101	李勇	92.0	数据库
20050101	李勇	85.0	数学
20050101	李勇	88.0	信息系统

显示全部 | 行数: 25 过滤行: 在表中搜索 按索引排序: 无

查询结果操作

(3) 查询“李新”老师所授课程的课程名称。

```
SELECT teacher.tname, course.cname FROM teacher INNER JOIN teaching ON
teacher.tno = teaching.tno INNER JOIN course ON teaching.cno = course.cno
WHERE teacher.tname = '李新';
```

正在显示第 0 - 0 行 (共 1 行, 查询花费 0.0004 秒。)

```
SELECT teacher.tname, course.cname FROM teacher INNER JOIN teaching ON teacher.tno =
teaching.tno INNER JOIN course ON teaching.cno = course.cno WHERE teacher.tname = '李新';
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索

额外选项

tname 姓名	cname
李新	数据库

显示全部 | 行数: 25 过滤行: 在表中搜索

(4) 查询女教师所授课程的课程号及课程名称。

```
SELECT teacher.tname, course.cno, course.cname FROM teacher INNER JOIN
teaching ON teacher.tno = teaching.tno INNER JOIN course ON teaching.cno =
course.cno WHERE teacher.tsex = '女';
```

没有 **|103|王楠|女|1983-05-04|软件系|** 是因为，teaching表根本没有103这个教师编号。

(5) 查询至少选修一门课程的女学生姓名。

```
SELECT student.sname, sc.sno, COUNT(sc.sno) FROM student INNER JOIN sc ON
student.sno = sc.sno WHERE student.ssex = '女' GROUP BY sc.sno HAVING
COUNT(sc.sno)>=1;

#当使用 【SUM函数】 进行 【聚合计算】 时，应该将它放在 【GROUP BY子句】 中。否则将报错。
```

正在显示第 0 - 2 行 (共 3 行, 查询花费 0.0003 秒。)

```
SELECT student.sname, sc.sno, COUNT(sc.sno) FROM student INNER JOIN sc ON student.sno = sc.sno
WHERE student.ssex = '女' GROUP BY sc.sno HAVING COUNT(sc.sno)>=1;
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索

额外选项

sname 姓名	sno	COUNT(sc.sno)
刘晨	20050201	2
孙悦	20050204	1
李丽丽	20050302	2

(6) 查询姓“王”的学生所学的课程名称。

使用数据库表明别名简写
如果你用了别名简写的话，那就得全程用简写，否则会报错

```
SELECT s.sname, c.cname FROM student s JOIN sc ON s.sno = sc.sno JOIN course
c ON sc.cno = c.cno WHERE s.sname LIKE '王%'
```

正在显示第 0 - 2 行 (共 3 行, 查询花费 0.0004 秒。)

```
SELECT s.sname, c.cname FROM student s JOIN sc ON s.sno = sc.sno JOIN course c ON sc.cno = c.cno
WHERE s.sname LIKE '王%'
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索

额外选项

sname	cname
王力杰	数据库
王亮	数学
王力杰	信息系统

(7) 查询选修“数据库”课程且成绩在80～90分之间的学生学号及成绩。□查询姓名?

```
SELECT s.sno, sc.degree, s.sname FROM student s JOIN sc ON s.sno = sc.sno
JOIN course ON sc.cno = course.cno WHERE course.cname = '数据库';
```


正在显示第 0 - 4 行 (共 5 行, 查询花费 0.0004 秒。)

SELECT s.sno, sc.degree, s.sname FROM student s JOIN sc ON s.sno = sc.sno JOIN course ON sc.cno = course.cno WHERE course.cname = '数据库';

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索

额外选项

sno	degree	sname
20050101	92.0	李勇
20050202	70.0	张立
20050205	NULL	赵凯
20050302	78.0	李丽丽
20050303	58.0	王力杰

显示全部 | 行数: 25 过滤行: 在表中搜索

(8) 查询课程成绩及格的男同学的学生信息及课程号与成绩。

```
SELECT s.sno, s.sname, sc.cno, sc.degree FROM student s JOIN sc ON s.sno =
sc.sno JOIN course c ON sc.cno = c.cno WHERE sc.degree >= 60 AND s.ssex =
'男';
```

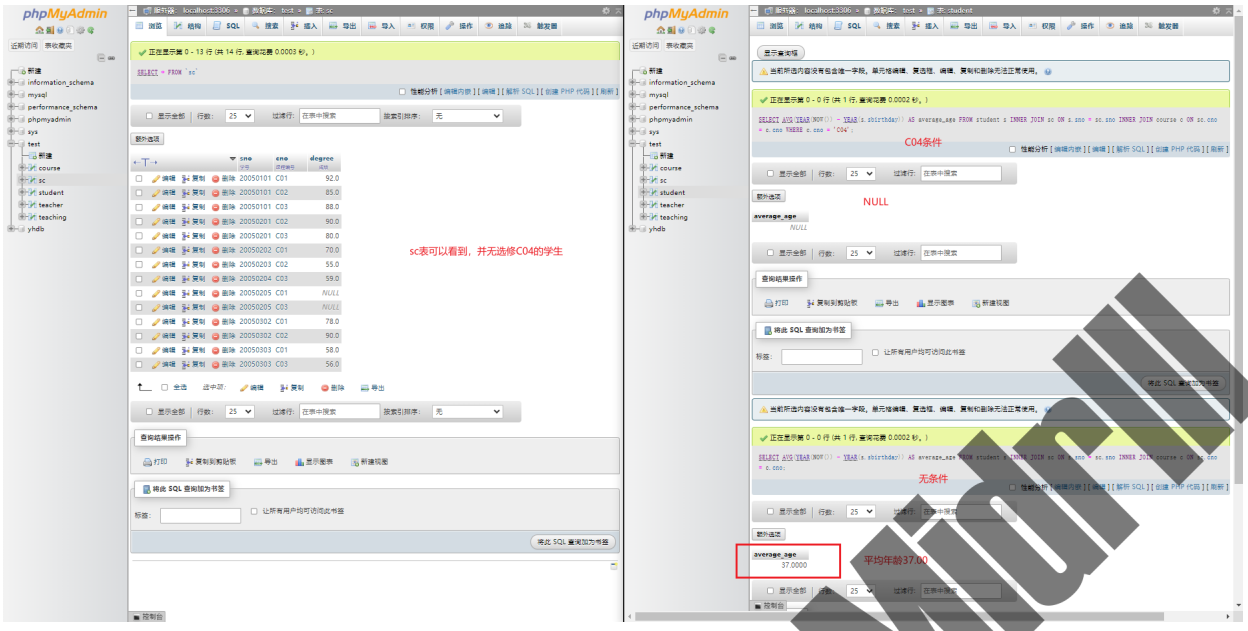
(9) 查询选修“c04”课程的学生的平均年龄。

注意，虽然说 student的sbirthday字段内容是 '1987-01-12 00:00:00'
但是，YEAR(student.sbirthday)，会获得它的年份，也就是1987.

```
SELECT (YEAR(NOW()) - YEAR(student.sbirthday)) FROM student;
```

而上面的语句，会将今年的年份，减去表里的年份，并输出数字来让你知道他几岁了。
所以整条语句就是。

所以。整条语句就是



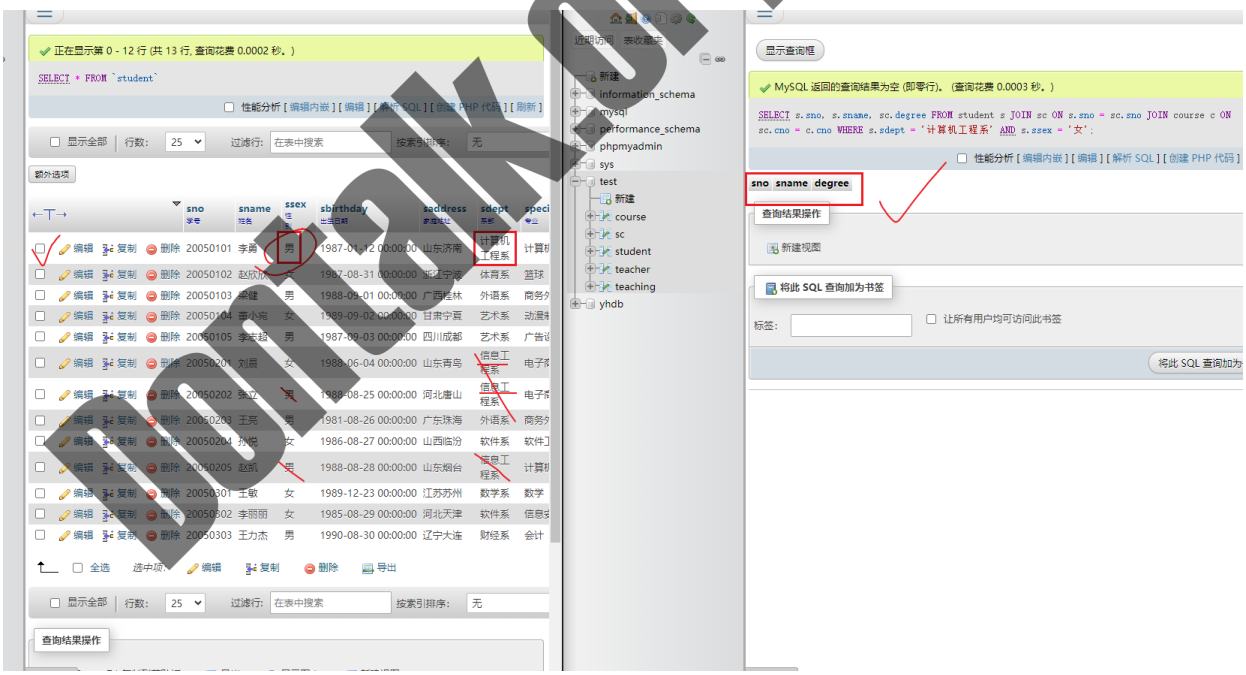

```
SELECT AVG(YEAR(NOW()) - YEAR(s.sbirthday)) AS average_age
FROM student s
INNER JOIN sc ON s.sno = sc.sno
INNER JOIN course c ON sc.cno = c.cno
WHERE c.cno = 'C04';
```

(10) 查询学习课程名为“数学”的学生学号和姓名。

```
SELECT s.sno, s.sname FROM student s JOIN sc ON s.sno = sc.sno JOIN course c
ON sc.cno = c.cno WHERE c.cname = '数学';
```

多表嵌套查询

(1) 查询计算机工程系 女学生 的【学生学号】、【姓名】及【考试成绩】。



别担心，就是空的，但是要是信息工程系，确实有一位女生。但是我们没有过滤相同项

正在显示第 0 - 1 行 (共 2 行, 查询花费 0.0002 秒。)

```
SELECT s.sno, s.sname, sc.degree FROM student s JOIN sc ON s.sno = sc.sno JOIN course c ON sc.cno = c.cno WHERE s.sdept = '信息工程系' AND s.ssex = '女';
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索

额外选项

sno	sname	degree
20050201	刘晨	90.0
20050201	刘晨	80.0

显示全部 | 行数: 25 过滤行: 在表中搜索

查询结果操作

(2) 查询“李勇”同学所选课程的成绩。

```
SELECT c.cname, sc.degree FROM student s JOIN sc ON s.sno = sc.sno JOIN course c ON sc.cno = c.cno WHERE s.sname = '李勇';
```

正在显示第 0 - 2 行 (共 3 行, 查询花费 0.0002 秒。)

```
SELECT c.cname, sc.degree FROM student s JOIN sc ON s.sno = sc.sno JOIN course c ON sc.cno = c.cno WHERE s.sname = '李勇';
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索

额外选项

cname	degree
数据库	92.0
数学	85.0
信息系统	88.0

显示全部 | 行数: 25 过滤行: 在表中搜索

查询结果操作

(3) 查询“李新”老师所授课程的课程名称。

```
SELECT c.cname FROM teacher t JOIN teaching te ON t.tno = te.tno JOIN course
c ON te.cno = c.cno WHERE t.tname = '李新';
```

(4) 查询女教师所授课程的课程号及课程名称。

```
SELECT c.cno, c.cname FROM teacher t JOIN teaching te ON t.tno = te.tno JOIN
course c ON te.cno = c.cno WHERE t.tsex = '女';
```

正在显示第 0 - 2 行 (共 3 行, 查询花费 0.0002 秒。)

```
SELECT c.cno, c.cname FROM teacher t JOIN teaching te ON t.tno = te.tno JOIN course c ON te.cno = c.cno WHERE t.tsex = '女';
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索

额外选项

cno	cname
C02	数学
C03	信息系统
C06	商务外语

显示全部 | 行数: 25 过滤行: 在表中搜索

(5) 查询姓“王”的学生所学的课程名称。

```
SELECT c.cname FROM student s JOIN sc ON s.sno = sc.sno JOIN course c ON sc.cno = c.cno WHERE s.sname LIKE '王%';
```

正在显示第 0 - 2 行 (共 3 行, 查询花费 0.0002 秒。)

```
SELECT c.cname FROM student s JOIN sc ON s.sno = sc.sno JOIN course c ON sc.cno = c.cno WHERE s.sname LIKE '王%';
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索

额外选项

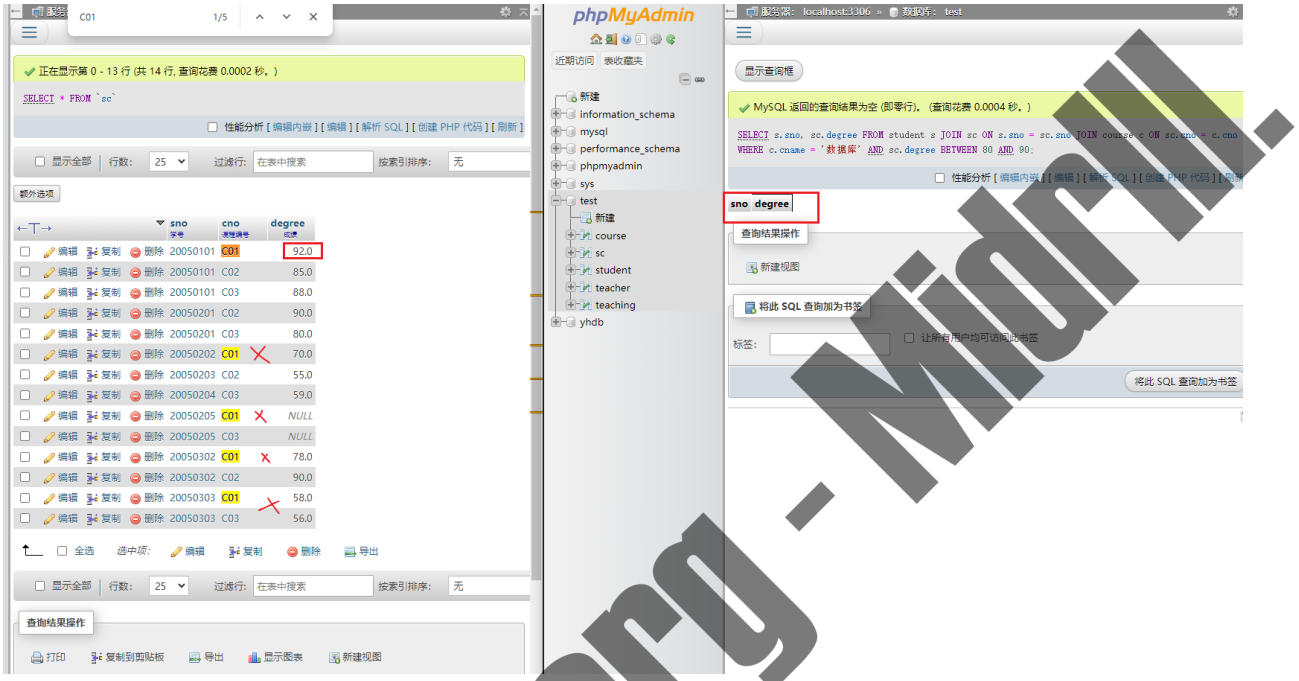
cname
数据库
数学
信息系统

显示全部 | 行数: 25 过滤行: 在表中搜索

查询结果操作

(6) 查询选修“数据库”课程且成绩在80～90分之间的学生学号及成绩。

```
SELECT s.sno, sc.degree FROM student s JOIN sc ON s.sno = sc.sno JOIN course
c ON sc.cno = c.cno WHERE c.cname = '数据库' AND sc.degree BETWEEN 80 AND 90;
```



唯一一个满足条件，欧布，大于90分，不在80-90这个区间，那也不符合条件，所以一条数据没有

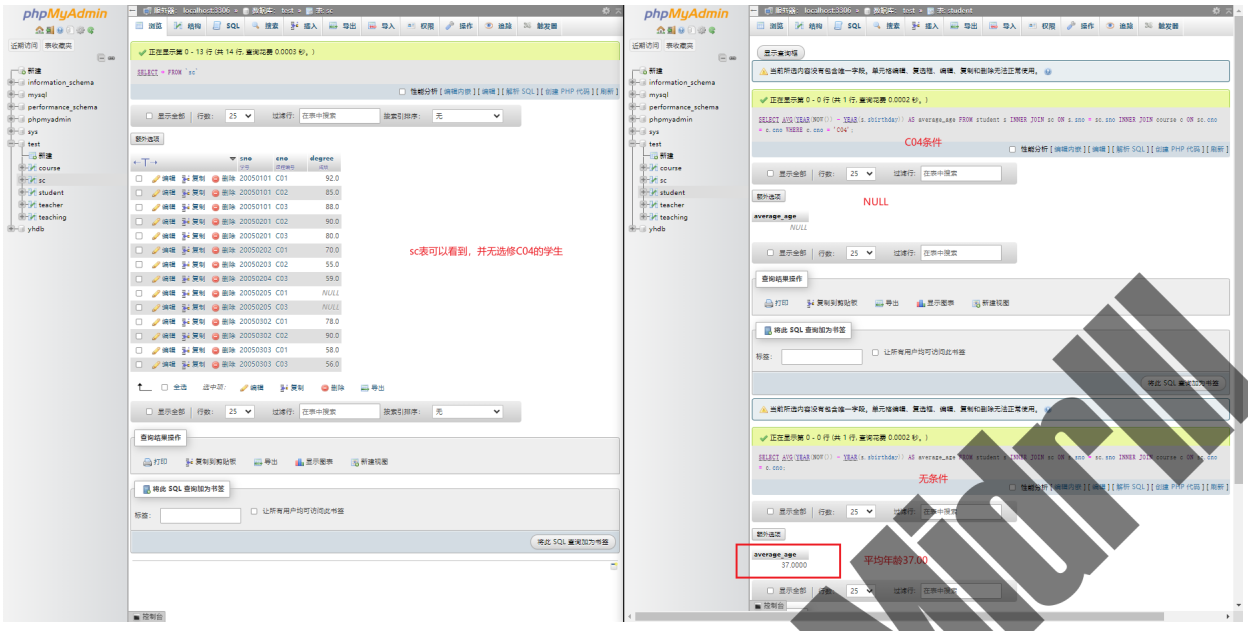
(7) 查询选修“C04”课程的学生的平均年龄。

上面已经讲过了：(如上，同下)
注意，虽然说 student的sbirthday字段内容是 '1987-01-12 00:00:00'
但是，YEAR(student.sbirthday)，会获得它的年份，也就是1987.

```
SELECT (YEAR(NOW()) - YEAR(student.sbirthday)) FROM student;
```

而上面的语句，会将今年的年份，减去表里的年份，并输出数字来让你知道他几岁了。
所以整条语句就是。

所以。整条语句就是




```
SELECT AVG(YEAR(NOW()) - YEAR(s.sbirthday)) AS average_age
FROM student s
INNER JOIN sc ON s.sno = sc.sno
INNER JOIN course c ON sc.cno = c.cno
WHERE c.cno = 'C04';
```

(8) 查询学习课程名为“数学”的学生学号和姓名。

```
SELECT s.sno, s.sname FROM student s JOIN sc ON s.sno = sc.sno JOIN course c
ON sc.cno = c.cno WHERE c.cname = '数学';
```

正在显示第 0 - 3 行 (共 4 行, 查询花费 0.0002 秒。)

```
SELECT s.sno, s.sname FROM student s JOIN sc ON s.sno = sc.sno JOIN course c ON sc.cno = c.cno
WHERE c.cname = '数学';
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索 按索引排序: 无

额外选项

sno	sname
20050101	李勇
20050201	刘晨
20050203	王亮
20050302	李丽丽

显示全部 | 行数: 25 过滤行: 在表中搜索 按索引排序: 无

(9) 查询“钱军”教师任课的课程号，选修其课程的学生的学号和成绩。

```
SELECT c.cno, sc.sno, sc.degree FROM teacher t JOIN teaching te ON t.tno =
te.tno JOIN course c ON te.cno = c.cno JOIN sc ON c.cno = sc.cno WHERE
t.tname = '钱军';
```

正在显示第 0 - 3 行 (共 4 行, 查询花费 0.0003 秒。)

```
SELECT c.cno, sc.sno, sc.degree FROM teacher t JOIN teaching te ON t.tno = te.tno JOIN course c
ON te.cno = c.cno JOIN sc ON c.cno = sc.cno WHERE t.tname = '钱军';
```

性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

显示全部 | 行数: 25 过滤行: 在表中搜索

额外选项

cno	sno	degree
C02	20050101	85.0
C02	20050201	90.0
C02	20050203	55.0
C02	20050302	90.0

显示全部 | 行数: 25 过滤行: 在表中搜索

(10) 查询在第3学期所开课程的课程名称及成绩。

```
SELECT course.cname, student.sname, sc.degree FROM course INNER JOIN sc ON
course.cno = sc.cno INNER JOIN student ON sc.sno = student.sno INNER JOIN
teaching ON course.cno = teaching.cno WHERE teaching.cterm = 3;
```

2. 体会与总结：GG