[【Intermediate SQL-1 】五道较难的sql练习](http://blog.csdn.net/cryssdut/article/details/49254281)

破题累死我了，网上另外一份答案写的也不错

（链接：[http://blog.csdn.net/cygeek\_dut/article/details/13090743](http://blog.csdn.net/cygeek_dut/article/details/13090743" \t "_blank)）

但是他有的地方显然写麻烦了，而且对于第五个问那份答案好像理解错了题意，我仔细研究了下，写了份原创的，如果改一下缩进和变量名什么的，应该比网上这份仁兄的答案短一点。

我还真头一次知道sql原来可以写的这么跳，写的如此花哨。。简直是秀操作。。。 orz

当然也有可能存在有问题的地方，欢迎指正交流~！

我这里直接贴代码了，题目[**数据库**](http://lib.csdn.net/base/mysql)数据的构建之类的请参考上面提到的那份答案

第一题：

1. Find the sections which have the minimum enrollment among sections registered by students. For each section as such, information displayed should involve:   
l Identifier of section(i.e. the primary key for section) l Course name corresponding to the section l Enrollment of the section('enrollment' is the alias for the number of students who registered for the section) l TOP keyword in SQL Server is denied.

**[sql]** [view plain](http://blog.csdn.net/cryssdut/article/details/49254281) [copy](http://blog.csdn.net/cryssdut/article/details/49254281)

[print?](http://blog.csdn.net/cryssdut/article/details/49254281)

1. **with** enrollment(course\_id, sec\_id, semester, year, enrollment) **as**
2. (
3. **select** takes.course\_id,
4. takes.sec\_id,
5. takes.semester,
6. takes.year,
7. COUNT(takes.course\_id)
8. **from** takes
9. **group** **by**
10. takes.course\_id,
11. takes.sec\_id,
12. takes.semester,
13. takes.year
14. ),
15. min\_enrollment(enrollment) **as**
16. (
17. **select** **min**(enrollment)
18. **from** enrollment
19. )
20. **select** enrollment.course\_id,
21. enrollment.sec\_id,
22. enrollment.semester,
23. enrollment.year,
24. course.title,
25. enrollment.enrollment
26. **from** course, enrollment, min\_enrollment
27. **where** enrollment.enrollment = min\_enrollment.enrollment and
28. course.course\_id = enrollment.course\_id;

第二题：

2. USE aggregation on outer join to construct the following query   
For all students, list the registration information of the students. The students who have never registered for any courses should also be considered. In the case, the aggregative information of such students should be set to 0. For each student, information displayed should involve:   
l Identifier of student(i.e. the primary key for student) l Name of the student l Number of course registrations (Caution: Not the number of section registrations. E.g. , student A registered the course B twice in 2 sections, the number of course registrations is 1 and the number of section registrations is 2). l Number of section registrations l TOP keyword in SQL Server is denied.

**[sql]** [view plain](http://blog.csdn.net/cryssdut/article/details/49254281) [copy](http://blog.csdn.net/cryssdut/article/details/49254281)

[print?](http://blog.csdn.net/cryssdut/article/details/49254281)

1. **select** student.ID, student.**name**, COUNT(**distinct** takes.course\_id) **as** 'course注册数', COUNT(takes.course\_id) **as** 'section注册数'
2. **from** student left join takes **on** student.ID = takes.ID
3. **group** **by** student.ID, student.**name**

第三题：

3. USE scalar subquery to construct the following query   
Find the information for the instructors who have taught more than 1 course (that is, he/she should have taught 2 distinct courses as least). For each instructor as such, information displayed should involve:   
l Identifier of instructor(i.e. the primary key for instructor) l Name of the instructor l Average salary of the department for which the instructor works l Sum of credit points taught by the instructor(for example, if instructor A  has taught course A (2 credit points) twice, course B(3 credit points) once, then the sum of credit points taught by instructor A is 7)

**[sql]** [view plain](http://blog.csdn.net/cryssdut/article/details/49254281) [copy](http://blog.csdn.net/cryssdut/article/details/49254281)

[print?](http://blog.csdn.net/cryssdut/article/details/49254281)

1. --使用标量子查询
2. **with** instructor\_taught\_num(ID, num) **as**
3. (
4. **select** teaches.ID, COUNT(**distinct** course\_id)
5. **from** teaches
6. **group** **by** teaches.ID
7. )
8. **select** **distinct** ins1.ID,
9. ins1.**name**,
10. (
11. **select** AVG(salary)
12. **from** instructor ins2
13. **where** ins1.dept\_name = ins2.dept\_name
14. **group** **by** ins2.dept\_name
15. )**as** '所在学院平均工资',
16. (
17. **select** SUM(course.credits)
18. **from** course, teaches
19. **where** teaches.ID = ins1.ID and
20. course.course\_id = teaches.course\_id
21. )**as** '所授课总学分'
22. **from** instructor ins1, instructor\_taught\_num
23. **where** ins1.ID = instructor\_taught\_num.ID and
24. instructor\_taught\_num.num > 1;

第四题：

4. Find students who have registered for **some but not all** courses taught by instructors of department '拳脚学院'. Do this using the "not exists ... except ..." structure. For each student as such, information displayed should involve:   
l Identifier of student(i.e. the primary key for student) l Name of the student l Number of courses, taught by instructors of department '拳脚学院', registered by the student

**注意这道题说的是选了但又没全选。**

**[sql]** [view plain](http://blog.csdn.net/cryssdut/article/details/49254281) [copy](http://blog.csdn.net/cryssdut/article/details/49254281)

[print?](http://blog.csdn.net/cryssdut/article/details/49254281)

1. **select** **distinct** student.ID, student.**name**
2. **from** student
3. **where** student.ID in (
4. **select** student.ID
5. **from** student, takes, course
6. **where** student.ID = takes.ID and
7. course.course\_id = takes.course\_id and
8. course.dept\_name = '拳脚学院'
9. ) and
10. student.ID not in
11. (
12. **select** S.ID
13. **from** student S
14. **where** not exists ((
15. **select** course\_id
16. **from** course
17. **where** dept\_name = '拳脚学院'
18. ) **except**(
19. **select** takes.course\_id
20. **from** takes
21. **where** takes.course\_id = student.ID
22. )
23. )
24. );

第五题：

5. As query requirement in Q4, Use matching of counts to fulfill the requirement. (don't forget the distinct clause!).

据我理解，这题的意思是用**比较数量**的方法去做第四题，所以应该用 **> < 之类的集合比较**方法来做，也就是说用<来表示比总数小。

另外那份答案说使用having，我觉得有点迷。。。

**[sql]** [view plain](http://blog.csdn.net/cryssdut/article/details/49254281) [copy](http://blog.csdn.net/cryssdut/article/details/49254281)

[print?](http://blog.csdn.net/cryssdut/article/details/49254281)

1. **with** take\_num(ID, **name**, num) **as**(
2. **select** **distinct** S.ID, S.**name**, COUNT(**distinct** T.course\_id)
3. **from** student S, takes T, course C
4. **where** S.ID = T.ID and
5. T.course\_id = C.course\_id and
6. C.dept\_name = '拳脚学院'
7. **group** **by** S.ID, S.**name**
8. )
9. **select** \*
10. **from** take\_num
11. **where** num < (
12. **select** COUNT(\*)
13. **from** course
14. **where** course.dept\_name = '拳脚学院'
15. )