1. 查询每个学生选课情况（包括没有选修课程的学生）。

```
select student.student_id, student.student_name, sc.term, sc.class_id,
c.class_name as num
from student
    left join select_class sc
        on student.student_id = sc.student_id
    left join class c
        on sc.class_id = c.class_id;
```

执行结果：

	student_id	student_name	term	class_id	num
1	1101	李明	201201	08305001	离散数学
2	1102	刘晓明	201201	08305001	离散数学
3	1102	刘晓明	201202	08305002	数据库原理
4	1102	刘晓明	201301	08305004	系统结构
5	1103	张颖	201201	08305001	离散数学
6	1103	张颖	201202	08305002	数据库原理
7	1103	张颖	201202	08305003	数据结构
8	1103	张颖	201301	08305001	离散数学
9	1103	张颖	201301	08305004	系统结构
10	1104	刘晶晶	201201	08305001	离散数学
11	1104	刘晶晶	201302	08302001	通信学
12	1105	刘成刚	<null>	<null>	<null>

2. 检索所有课程都选修的的学生的学号与姓名。

```
-- 以下查询语句目的是为了检索所有已经开设课程的学生与姓名
-- 试题2答案1:
select student.student_id, student.student_name, count(distinct
sc.class_id) select_num
from student inner join select_class sc on student.student_id =
sc.student_id
group by student_id
having select_num >= 4;
```

1 行

	student_id	student_name	select_num
1	1103	张颖	4

```
-- 试题2答案2:
-- 换一种思考角度就是没有一门课程是我不选修的
select *
from student s
```

```
where not exists(-- 该条语句注意，当下面的返回结果为空的时候才为真
    -- 下面的语句的目的是为了寻找一门该名同学没有选择过的课程
    select * from open_class c
    where not exists(
            -- 如果说该名同学选择了所有的选修课程，那么每一次返回都不为空
            -- 那么我们就找不到一门课程是该同学没有选修过的,那么就会返回空
            select * from select_class sc
            where s.student_id=sc.student_id and
sc.class_id=c.class_id
        )
    );
```

3. 检索选修课程包含 1106同学所学全部课程 的学生学号和姓名.

```
-- 试题3答案:
-- 参考老师PPT
-- 关键在于不存在这样的课程，学生1106选择了，而检索的学生没有选择
select distinct scx.student_id, s.student_name
from select_class scx
join student s on scx.student_id = s.student_id
where not exists(
    select *
    from select_class scy
    where scy.student_id=1106 and not exists(
        select *
        from select_class scz
        where scz.student_id=scx.student_id and
scz.class_id=scy.class_id
        )
    );
```

3 行

	student_id	student_name
1	1102	刘晓明
2	1103	张颖
3	1106	李二丽

4. 查询每门课程中分数最高的学生学号和学生姓名。

```sql
-- 试题4答案：
-- 内联结学生表来查询最后高分数对应的姓名
select sc.class_id, s.student_id, s.student_name, sc.score
from select_class sc
inner join student s on sc.student_id = s.student_id
where sc.score in (select max( sc.score) as max_score
                   from select_class sc
                   group by sc.class_id
                   having max_score >= 0);
```

3 行

	class_id	student_id	student_name	score
1	08305002	1102	刘晓明	82
2	08305003	1103	张颖	84
3	08305001	1107	张晓峰	90

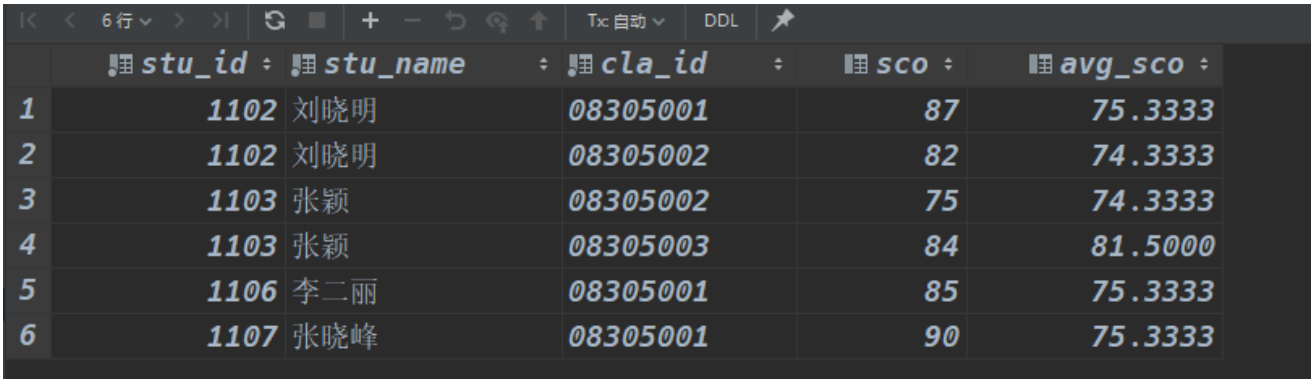
5. 查询年龄小于本学院平均年龄，所有课程总评成绩都高于所选课程平均总评成绩的学生学号、姓名和平均总评成绩，按年龄排序。

暂且不会。

```sql
-- 创建视图，给选课表添加一列属性为平均成绩的一列
create view advance_select as
select class_id, avg(score) avg_score
from select_class
group by class_id;


-- 创建视图2
create view useful_select as
select s.student_id stu_id, s.student_name stu_name, sc.class_id
cla_id, sc.score sco, `as`.avg_score avg_sco
from student s
inner join select_class sc on s.student_id = sc.student_id
inner join advance_select `as` on sc.class_id = `as`.class_id;

select *
from useful_select
where sco > avg_sco;
```



6 行 | Tx: 自动 | DDL

	stu_id	stu_name	cla_id	sco	avg_sco
1	1102	刘晓明	08305001	87	75.3333
2	1102	刘晓明	08305002	82	74.3333
3	1103	张颖	08305002	75	74.3333
4	1103	张颖	08305003	84	81.5000
5	1106	李二丽	08305001	85	75.3333
6	1107	张晓峰	08305001	90	75.3333