学生（**学号**，姓名，年龄，性别，班级）

课程（**课程号**，课程名，先修课程号，学分）

教师（**教师号**，教师名称）

选课（**学号**，**课程号**，教师号，成绩，**选课时间**）

关系代数

1. 查找选修了物理课的学生姓名

$$\prod_{姓名}\left(\sigma_{课程名='物理'}(课程)\bowtie 选课\bowtie 学生\right)$$

2. 查找教的学生的成绩都大于60分的教师（给出教师号即可）

$$\prod_{教师号}(选课)-\prod_{教师号}\left(\sigma_{成绩\leq 60}(选课)\right)$$

3. 查找没有选修张三老师教的所有课的学生

$$\prod_{学号}(学生)-\prod_{学号,课程号}(选课)\div\prod_{课程号}\left(\sigma_{教师名称='张三'}(教师)\bowtie 选课\right)$$

SQL

以下语法为 MySQL，与 SQL Server 大部分类似。

```
create database midterm;

use midterm

create table 学生 (
    学号 char(20) primary key,
    姓名 varchar(30),
    年龄 tinyint,
    性别 tinyint,
    班级 int
);

create table 课程 (
    课程号 char(20) primary key,
    课程名 varchar(30),
    先修课程号 char(20),
    学分 tinyint,
    constraint fk_course_prior foreign key (先修课程号)
        references 课程 (课程号)
        on update cascade
);

create table 教师 (
```

```
    教师号 char(20) primary key,
    教师名称 varchar(30)
);

create table 选课 (
    学号 char(20),
    课程号 char(20),
    教师号 char(20),
    成绩 tinyint,
    选课时间 datetime,
    primary key (学号, 课程号, 选课时间),
    constraint fk_sc_sid foreign key (学号)
        references 学生 (学号)
        on update cascade,
    constraint fk_sc_cid foreign key (课程号)
        references 课程 (课程号)
        on update cascade,
    constraint fk_sc_tid foreign key (教师号)
        references 教师 (教师号)
        on update cascade
);
```

1. 创建一个表，字段为：学生ID（主键） 字符型 长度20，学生姓名（非空）字符型 长度20，课程数量 数字型。查询每个学生选修的课程数量，将结果插入表中。（两条SQL语句，一条为create语句，一条为insert语句）

```
create table 选课数量 (
    学生ID char(20) primary key,
    学生姓名 varchar(20) not null,
    课程数量 int
);

insert into 选课数量
    select 学生.学号, 姓名, count(distinct 课程号)
    from 学生 left join 选课
        on 学生.学号 = 选课.学号
    group by 学生.学号;
```

2. 找出所有姓诸的学生姓名（排除姓‘诸葛’的学生）

```
select 姓名
from 学生
where 姓名 like '诸%' and 姓名 not like '诸葛%';
```

3. 检索至少得过一次课程最高分的学生学号姓名（不考虑重修的情况，也就是比较分数时不考虑重修的记录）

```
select distinct sc1.学号, 姓名
from (
    select *
    from 选课 sc1
    where 选课时间 <= (
        select min(选课时间)
        from 选课 sc2
        where sc1.学号 = sc2.学号 and sc1.课程号 = sc2.课程号
        group by sc2.学号, sc2.课程号
    )
) sc1, 学生
where sc1.学号 = 学生.学号 and 成绩 >= (
    select max(成绩)
    from (
        select *
        from 选课 sc2
        where 选课时间 <= (
            select min(选课时间)
            from 选课 sc3
            where sc2.学号 = sc3.学号 and sc2.课程号 = sc3.课程号
            group by sc3.学号, sc3.课程号
        )
    ) sc2
    where sc1.课程号 = sc2.课程号
);
```

4. 查询如下内容（学生ID，课程ID，时间），列出**每个**学生第一次选某课程的时间（即非重修的选课时间）。

如果某个学生没有选任何课，就把“课程ID”和“时间”设为 NULL。

```
select 学生.学号, 课程号, 选课时间
from 学生 left join 选课 sc1
    on 学生.学号 = sc1.学号
where 选课时间 is null or 选课时间 <= (
    select min(选课时间)
    from 选课 sc2
    where sc1.学号 = sc2.学号 and sc1.课程号 = sc2.课程号
    group by sc2.学号, sc2.课程号
);
```

5. 将学生的重修课程成绩都改成60分（只要这门课重修过，这门课每次的成绩都改成 60 分）

由于 MySQL 不支持对同一个表先查询再更新，所以需要在 WHERE 子句里使用临时表。

```
update 选课 sc1
set 成绩 = 60
where (sc1.学号, sc1.课程号) in (
```

```
    select temp.学号, temp.课程号
    from (
        select distinct sc2.学号, sc2.课程号
        from 选课 sc2
        where exists (
            select sc3.学号, sc3.课程号
            from 选课 sc3
            where sc2.学号 = sc3.学号 and sc2.课程号 = sc3.课程号
            group by sc3.学号, sc3.课程号
            having count(*) >= 2
        )
    ) temp
);
```

简化写法，SQL Server 应该支持：

```
update 选课 sc1
set 成绩 = 60
where exists (
    select sc2.学号, sc2.课程号
    from 选课 sc2
    where sc1.学号 = sc2.学号 and sc1.课程号 = sc2.课程号
    group by sc2.学号, sc2.课程号
    having count(*) >= 2
);
```

6. 查找每个学生当前可选修的课程列表（即该学生没有选该课程，且该学生已经**修过**了该课程的先修课）

只需要修过先修课，不需要及格。

```
select 学生.学号, 课程.课程号
from 学生, 课程
where not exists (
    select *
    from 选课
    where 选课.学号 = 学生.学号 and 选课.课程号 = 课程.课程号
) and exists (
    select *
    from 选课
    where 选课.学号 = 学生.学号 and 选课.课程号 = (
        select 先修课程号
        from 课程 c2
        where c2.课程号 = 课程.课程号
    )
);
```