姓名：胡瑞康

学号：22336087

1.

外连接表达式可以在SQL中不使用SQL outer join操作来计算。为了说明这一点，展示如何重写以下SQL查询而不使用外连接表达式：

```
select * from student natural left outer join takes
select * from student natural full outer join takes
```

可以通过使用`LEFT JOIN`和明确的`ON`条件来替代`NATURAL LEFT OUTER JOIN`。

```
select *
from student left join takes
on student.ID = takes.ID;
```

`FULL OUTER JOIN`可以使用`UNION`操作结合`LEFT JOIN`和`RIGHT JOIN`来替代。

```
select *
from student left join takes
on student.ID = takes.ID
union
select *
from student right join takes
on student.ID = takes.ID;
```

2.

SQL允许外键依赖引用相同的关系，如以下示例中所示：

```
create table manager
(employee ID char(20),
manager ID char(20),
primary key (employee ID),
foreign key (manager ID) references manager(employee ID)
on delete cascade );
```

在这里，“employee ID”是“manager”表的主键，这意味着每个员工最多只有一个经理。外键子句要求每位经理也必须是一名员工。请解释当“manager”关系中的元组被删除时会发生什么事情。

1. **外键关系**：`manager ID`是一个外键，指向`manager`表中的`employee ID`。这意味着，每个经理也是一名员工。
2. **ON DELETE CASCADE**：如果`manager`表中的一条记录被删除，即某个员工被删除，那么所有引用该员工为经理的记录（下属员工的记录）也将被删除。这是因为`ON DELETE CASCADE`触发了级联删除。

- 如果一名员工是其他员工的经理，并且该员工的记录被删除，那么这些下属员工的`manager ID`也会被删除，意味着他们的经理信息也会被移除。
- 级联删除可以递归发生，即如果被删除的员工本身也是某个员工的经理，整个链条上的相关记录都会被删除。

3.

定义一个名为"tot_credits"的视图（year, num_credits），该视图显示每年所修的总学分数。

```
create view tot_credits as
select year, sum(credits) as num_credits
from takes
group by year;
```

该视图`tot_credits`按每个年份分组，并计算该年所有学生修读的总学分数。

4. 请表达以下查询的SQL，不使用子查询和集合操作（参考fig 1）：

```
select ID
from student
except
select s_id
from advisor
where i_ID is not null;
```

这个查询相当于找出没有导师的学生

使用`LEFT JOIN`和`WHERE`条件来避免子查询和集合操作：

```
select student.ID
from student
left join advisor on student.ID = advisor.s_id
where advisor.s_id is null or advisor.i_ID is null;
```

`advisor.s_id is null` 确保了那些在 `advisor` 表中没有记录的学生也会被包含在结果中

`advisor.i_ID is null` 确保了那些没有导师的学生也会被包含在结果中

5. 重新编写查询，使用内连接和使用"using"条件来代替"natural join" （参考fig1）：

```
select *
from section natural join classroom;
```

在 section 表和 classroom 表中，公共列是 building 和 room_number。
使用 INNER JOIN 和 USING**：将 NATURAL JOIN 替换为 INNER JOIN，并使用 USING 子句来指定连接条件。

```
select *
from section inner join classroom
using (building, room_number);
```

6.

假设用户A对关系r具有所有授权权限，并将关系r的select权限授予public，并附带授权选项。假设用户B随后将r的select权限授予A。这样会在授权图中产生一个循环吗？请解释原因。

授权图是一个有向图，节点代表用户或角色，边代表权限的授予。边的方向从授予权限的用户指向被授予权限的用户，表示某个用户将某项权限授予了另一个用户。

过程分析：

- **用户A**最初拥有对关系r的所有授权权限，并将r的SELECT权限授予了public，并且附带了“授权选项”。这意味着，任何public中的用户（如用户B）可以将r的SELECT权限授予其他用户。
- **用户B**作为public的一员，获得了SELECT权限，并且由于有授权选项，用户B将该权限再次授予了**用户A**。
- **用户A**最初拥有对r的所有权限，所以用户A的权限并不依赖于其他用户授予的权限。
- 即使用户B将SELECT权限授予用户A，这只是一个额外的授权行为，但它不会改变用户A已经拥有的权限。用户A不依赖于用户B的授权行为来获取权限，因为A原本就有完整的权限。

因此，虽然用户B将SELECT权限授予了用户A，但这不会导致授权图中形成循环。**原因是用户A的权限独立于B的授权行为**，因为用户A在最初就拥有了权限，而B的授予只是多余的操作，不会影响图的结构。