Second :

1. select name from student where dept\_name='ECE';

2. select course.title,course.credits from teaches join course on teaches.course\_id=course.course\_id join instructor on teaches.ID=instructor.ID where instructor.name='bell';

1. select distinct sec\_id from section where year = 2016 and (semester = 'fall' and building = 'sharp\_lab');

4. select distinct start\_time,end\_time from time\_slot join section on time\_slot.time\_slot\_id = section.time\_slot\_id where time\_slot.day like 'Mon%' and section.building ='smith';

5. select course.title,course.dept\_name from course join prereq on prereq.prereq\_id = course.course\_id where prereq.course\_id like 'CISC4%';

6. select round(sum((case takes.grade when 'A' then 4 when 'B' then 3 when 'C' then 2 when 'D' then 1 else 0 end)\*credits)/sum(credits),2) as GPA from takes,course,student where takes.course\_id = course.course\_id and student.ID = '1' and takes.ID = '1';

7. select distinct course.title from section,course where section.course\_id = course.course\_id and ((section.semester='spring' and section.year=2016) or (section.semester='fall' and section.year=2016));

8. select distinct course.title from course,section where course.course\_id=section.course\_id and section.year=2016 and section.semester in ('fall','spring') group by course.course\_id having count(section.course\_id) =2;

9. select a.dept\_name as department, avg(b.salary) as avg\_salary from instructor a left join instructor b on a.dept\_name = b.dept\_name group by a.dept\_name;

10. select (classroom.capacity-count(distinct takes.ID)) as remain\_cisc220 from section,classroom,takes where takes.course\_id='CISC220' and takes.sec\_id='011' and takes.semester='fall' and takes.year=2016 and section.course\_id='CISC220' and section.sec\_id='011' and section.building = classroom.building and section.room\_no =classroom.room\_no and section.semester = 'fall' and section.year=2016 group by classroom.capacity;

11. select instructor.name as math\_instr,count(distinct advisor.s\_id) as tot\_student from advisor join instructor on advisor.i\_id=instructor.ID join student on advisor.s\_id=student.ID where instructor.dept\_name='math' group by instructor.name;

12. select dept\_name as Dept,round(avg(GPA),2) as avg\_GPA\_dept from(select student.dept\_name,sum((case grade when 'A' then 4 when 'B' then 3 when 'C' then 2 when 'D' then 1 else 0 end)\*credits)/sum(credits) as GPA from takes,course,student where takes.course\_id=course.course\_id and student.ID=takes.ID group by takes.ID) as all\_GPA group by dept\_name;

Fourth:

1. First, semester and year(as primary keys) are keeping the section-takes-teaches tables away from inconsistency, and they work like supplemental and unique identifications for each different data, which are inserted into the tables. So, if there are not primary key of these tables, some invalid data will be added into the database, like a student called Bob takes class, when semester and year are not primary keys in these tables, wrong information, like Bob takes a certain course during a certain semester and a certain year, but a instructor may use the same section id and course id last year, so Bob would be likely to take wrong class. This scenario will cause problems, which will be a class full of students, but no instructor at all. As a result, keeping these primary keys and holding the takes and teaches for a certain course as a one to one relation are critical for the database’s integrity.
2. It will be like this:

Table (building\_info)

building varchar(25) This is the abbreviated name for buildings and primary key,

buildingfull varchar(50) full names,

address varchar(50),

year\_built smallint,

First, we should add the table to the database like showing above, and add some building data into it.

Second, the ‘department’ and ‘section’ will refer to the ‘building\_info’ table’s primary key building, so there will be additional foreign key for ‘department’ and ‘section’ tables in “building” column.

1. 1. Exceeding people in a course, which means there will be much more students than the classroom’s max capacity (Even with a waiting list, a class for 20 students can not receive 200 students).

2.self pre-required course, for example, course:MATH106 can be the prereq of its self. So, “insert into prereq(course\_id,prereq\_id) values ('MATH106','MATH106');” will be OK and executed.

3.In table ‘time\_slot’, ‘start\_time’ could be same as ‘end\_time’, which is impossible in real world(or the end\_time will be ealier than start\_time, which is also impossible in real life).

4.Some classroom conflict, they may take the same time frame at the same semester.