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';

3. 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:

a. 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.

b. 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.

c. 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.