Given the following tables:

- students(<u>sid</u>, name, age, gpa)
- courses(<u>cid</u>, deptid, name)
- professors(<u>ssn</u>, name, address, phone, deptd)
- enrollment(<u>sid</u>, <u>cid</u>, section, grade,
  foreign key (sid) references students,
  foreign key (cid) references courses,
  foreign key (cid, section) references teaches)
- teaches(<u>cid</u>, <u>section</u>, ssn,
  foreign key (cid) refernces courses,
  foreign key (ssn) references professors)

Domain

- *cid* is in {‘198:11’, ‘640:151’, ‘198:112’, …}
- *deptid* is in {‘cs’, ‘math’, ‘music’, …}
- *grade* is in {‘A’, ‘B’, ‘C’, …}
- *section, age, ssn* are an integers
- *address, phone, name* are strings
- *gpa* is float

Provide SQL instructions for each of the following questions

1. Create the database schema.

```
create table students (sid INTEGER, name VARCHAR(50), age, INTEGER, gpa REAL,
       PRIMARY KEY (sid))

create table courses (cid VARCHAR(7), deptid VARCHAR(5), name VARCHAR(50),
       PRIMARY KEY(cid))

create table professors (ssn INTEGER, name VARCHAR(50), address VARCHAR(200),
       phone VARCHAR(10), deptid VARCHAR(5), PRIMARY KEY (ssn))

create table enrollment (sid INTEGER, cid VARCHAR(7), section INTEGER,
       grade VARCHAR(2), PRIMARY KEY (sid,cid),
       FOREIGN KEY (sid) REFERENCES students,
       FOREIGN KEY (cid) REFERNCES courses,
       FOREIGN KEY (cid, section) REFERNCES teaches)

create table teaches(cid VARCHAR(7), section INTEGER, ssn INTEGER,
       PRIMARY KEY (cid, section)
       FOREIGN KEY (cid) REFERENCES courses,
       FOREIGN KEY (ssn) REFERENCES professors)
```

2. Find the name of professors that work for the cs department.

```
SELECT p.name
FROM   professors p
WHERE  deptid='cs';
```

3. Find those students (sid) enrolled in courses in the cs department.

```
SELECT DISTINCT s.sid
FROM   students s, enrollment e, courses c
WHERE  s.sid=e.sid AND e.cid=c.cid AND c.deptid='cs';
```

4. List ssn and name of professors that work for the cs department, but are not teaching any cs courses.

```
SELECT p.ssn, p.name
```

```
FROM   professors p
WHERE  p.deptid='cs' AND p.ssn NOT IN (SELECT t.ssn FROM teaches t);
```

5. List the number of courses offered by each department. Just the number of courses (not sections).

```
SELECT       c.deptid, COUNT(c.cid)
FROM         courses c
GROUPED BY   c.deptid;
```

6. List of those departments that offer more than 10 courses.

```
SELECT t.deptid
FROM   (      SELECT       c.deptid, COUNT(c.cid) AS total
              FROM         courses c
              GROUPED BY   c.deptid) t
WHERE  t.total > 10;
```

7. Produce a list of the name of those students whose professor's name starts with an M. Your result must have no duplicates.

```
SELECT DISTINCT s.name
FROM   students s, enrollment e, teaches t, professors p
WHERE  s.sid=e.sid AND e.cid=t.cid AND t.ssn=p.ssn AND p.name LIKE 'M%';
```

8. Assume that **small** sections have less than 30 students, **medium** sections have at least 30 students but less than 80, and **large** sections have at least 80 students.
Your result table should have the following rows and columns:

deptid	small	medium	large
cs			
math			
…	…	…	…

Each table entry must have the number of sections of a given size offered by each department.

```
CREATE TABLE section (deptid VARCHAR(5), section INTEGER, sid INTEGER, grade
                      varchar(2)
             PRIMARY KEY (deptid, section)
             FOREIGN KEY  deptid REFERENCES courses
             FOREIGN KEY  section REFERENCES enrollment
             FOREIGN KEY  sid REFERENCES enrollment)

SELECT temp.deptid,  sum(if(temp.size = 'small', 1, 0)) small,
                     sum(if(temp.size = 'medium', 1, 0)) medium,
                     sum(if(temp.size = 'large', 1, 0)) large,
FROM   (SELECT DISTINCT    c.deptid, s.sid,
      if(COUNT(sect.sid) < 30, 'small', if(COUNT(sect.sid) < 80, 'medium', large))
       size
       FROM   courses c, enrollment e, section sect
       GROUP BY c.deptid, sect.section) temp;
```

9. List of professors that work for departments with more than 20 faculty members and that offer more large sections than small and medium sections combined.

```
SELECT p.name, p.deptid
FROM   professors p, temp
WHERE 20 <= ( SELECT       COUNT(*)
             FROM         professors p
             GROUP BY     p.deptid)
AND    temp.large > temp.medium + temp.small;
```

10. Assume grades are A, B, C, D, F where D and F are failing grades. For each course (section) find the percentage of students that failed the course.

```
SELECT temp2.section, temp2.pass / (temp2.pass + temp2.fail) pass_pct
FROM   (SELECT             sect.section, sum(if(sect.grade = 'A', 1, 0) +
                           sum(if(sect.grade = 'B', 1, 0) +
                           sum(if(sect.grade = 'C', 1, 0) pass,
                           sum(if(sect.grade = 'D', 1, 0) +
                           sum(if(sect.grade = 'F', 1, 0) fail
       FROM         Section sect
       GROUP BY     sect.section )
GROUP BY     temp2.section;
```

11. Find the name of the professor with the maximum percentage of students that failed his course.

```
SELECT p.name
FROM   professors p, teaches t, temp2
WHERE  p.ssn = t.ssn AND temp2.pass_pct = min(temp2.pass_pct);
```

12. On average what percentage of students fail a course? (total number of students that failed a course / total number of enrolled students).

```
SELECT COUNT(temp2.fail) / COUNT(s.sid)
FROM   temp2, students s
GROUP BY     s.sid;
```

13. Find a list of courses (sections) where the percentage of students with D or F is greater than average.

```
SELECT       sect.section
FROM         section sect
GROUP BY     sect.section
HAVING       (SELECT      COUNT(temp2.fail) / COUNT(s.sid)
             FROM   temp2, students s
             GROUP BY     s.sid) < (sect.fail / (sect.pass + sect.fail);
```

14. Write a query that produces the following table:

deptid	**SPS**	**% A**	**% B**	**% C**	**% D**	**% F**
cs						
math						
…	…	…	…	…	…	…

Where SPS is the average number of students in each section and column %A has the percentage of students that got an A, and so on, over all the courses offered by each department.

```
SELECT        sect.deptid, (COUNT(sect.sid) / COUNT(sect.section) SPS,
              sum(if(sect.grade = 'A', 1, 0) / COUNT(*) %A,
              sum(if(sect.grade = 'B', 1, 0) / COUNT(*) %B,
              sum(if(sect.grade = 'C', 1, 0) / COUNT(*) %C,
              sum(if(sect.grade = 'D', 1, 0) / COUNT(*) %D,
              sum(if(sect.grade = 'F', 1, 0) / COUNT(*) %F,
FROM          section sect
GROUP BY      sect.deptid, sect.section;
```