```
/*
      Each offering of a course (i.e. a section) can have many Teaching assistants;
      each teaching assistant is a student.  Extend the existing schema(Add/Alter tables)
      to accommodate this requirement.
*/
drop table assistant;
create table assistant
      (
       ID               varchar(5),
       course_id        varchar(8),
       sec_id           varchar(8),
       semester         varchar(6),
       year             numeric(4,0),
       primary key (ID, course_id, sec_id, semester, year),
       foreign key (ID) references student (ID) on delete cascade,
       foreign key (course_id, sec_id, semester, year) references section (course_id, sec_id,
semester, year) on delete cascade
      );


/*
      Adding tuples to assistant relation
*/

delete from assistant;
insert into assistant values ('00128', 'BIO-101', '1', 'Summer', '2009');
insert into assistant values ('12345', 'BIO-101', '1', 'Summer', '2009');
insert into assistant values ('12345', 'CS-347', '1', 'Fall', '2009');
insert into assistant values ('19991', 'FIN-201', '1', 'Spring', '2010');

/*
      According to the existing schema, one student can have only one advisor.
      Alter the schema to allow a student to have multiple advisors and make sure that you
      are able to insert multiple advisors for a student.
*/

alter table advisor drop foreign key advisor_ibfk_1;
alter table advisor drop foreign key advisor_ibfk_2;
alter table advisor drop primary key, add primary key (s_ID, i_ID);
alter table advisor add foreign key (s_ID) references student (ID) on delete cascade;
alter table advisor add foreign key (i_ID) references instructor (ID) on delete cascade;

/*
      Insert in advisor
*/

insert into advisor values ('12345', '45565');
insert into advisor values ('12345', '76543');
insert into advisor values ('12345', '98345');
insert into advisor values ('00128', '10101');
insert into advisor values ('00128', '12121');
insert into advisor values ('55739', '76543');


/*
      Write SQL queries on the modified schema. You will need to insert data to ensure the query
results are not empty.

*/

/*
      Find all students who have more than 3 advisors
*/
select name, ID
from student join (select s_id, count(i_id) as cnt
                  from advisor
                  group by s_id
                  having cnt > 3) as x on student.ID = s_id
;
```

```
/*
      OUTPUT:-
      +---------+-------+
      | name    | ID    |
      +---------+-------+
      | Shankar | 12345 |
      +---------+-------+
      1 row in set (0.01 sec)
*/

/*
      Find all students who are co-advised by Prof. Srinivas and Prof. Ashok.

*/

/*
      inserting Ashok into instructor
*/

insert into instructor values ('54321', 'Ashok', 'Finance', '80000');

/*
      inserting into advisor students under Ashok
*/

insert into advisor values ('00128', '54321');
insert into advisor values ('55739', '54321');

/*
      Query
*/

select name, ID
from student join (select distinct a1.s_id
                   from advisor as a1 join advisor as a2 on a1.s_id = a2.s_id
                   where a1.i_id = (select id
                                    from instructor
                                    where name = 'Srinivasan') and a2.i_id = (select id
                                                                              from instructor
                                                                              where name = 'Ashok')) as
x on student.id = x.s_id
;

/*
      OUTPUT:-
      +-------+-------+
      | name  | ID    |
      +-------+-------+
      | Zhang | 00128 |
      +-------+-------+
      1 row in set (0.00 sec)
*/

/*
      Find students advised by instructors from different departments. etc.
*/

drop view v;
create view v as (select distinct ID, s_ID, dept_name
                  from advisor join instructor
                  on ID = i_ID);
select distinct student.ID, name
from student join v on student.ID = s_ID
where student.dept_name != v.dept_name
;
```

```
/*
      OUTPUT:-
      +-------+---------+
      | ID    | name    |
      +-------+---------+
      | 00128 | Zhang   |
      | 55739 | Sanchez |
      | 12345 | Shankar |
      +-------+---------+
      3 rows in set (0.00 sec)
*/

/*
      Delete all information in the database which is more than 10 years old.
      Add data as necessary to verify your query.
*/

/*
      Adding neccesary entries
*/

insert into section values ('BIO-301', '1', 'Summer', '2005', 'Painter', '514', 'A');
insert into section values ('CS-190', '1', 'Fall', '2004', 'Packard', '101', 'H');
insert into teaches values ('83821', 'CS-190', '1', 'Fall', '2004');
insert into teaches values ('98345', 'BIO-301', '1', 'Summer', '2005');
insert into takes values ('00128', 'CS-190', '1', 'Fall', '2004', 'A');
insert into takes values ('12345', 'BIO-301', '1', 'Summer', '2005', 'A-');

delete
from section
where year < YEAR(CURDATE()) - 10;

/*
      Delete the course CS 101.  Any course which has CS 101 as a prereq should
      remove CS 101 from its prereq set.  Create a cascade constraint to enforce the above rule,
      and verify that it is   working.
      Cascade constraint for above rule is created in "a3DDL.sql"
*/
alter table prereq drop foreign key prereq_ibfk_2;
alter table prereq add foreign key (prereq_id) references course (course_id) on delete cascade;

delete
from course
where course_id = 'CS-101';
```