# CSE 344 Section 4 Worksheet
## More Nested Queries

1) Consider a photo sharing Website, where users can post pictures, as well as comment and rate other user's pictures. The schema is:

Users(<u>uid</u>, name)
Comment(uid, pid, score, txt)
Picture(<u>pid</u>, author, img)

The database has the following constraints:

- Comment.uid is a foreign key to Users
- Comment.pid is a foreign key to Picture
- Picture.author is a foreign key to Users
- Comment.score is an integer number between 1 and 10
- All attributes are NOT NULL

A picture is considered highly rated if it received at least one score of 10, from a user other than its author. A cautious user is a user who commented only on highly rated pictures. (A user who did not comment at all is also cautious.) Write a SQL query that finds all cautious users. Your query should return a list of uid, name pairs


*select x.uid, x.name*
*from users x*
*where x.uid not in*
> *(select y.uid -- these are the non-cautious users*
> *from comment y -- check that y.pid is not highly rated*
> *where y.pid not in*
>> *(select u.pid -- the pictures that are highly rated*
>> *from comment u, picture v*
>> *where u.pid = v.pid and u.score = 10 and u.pid != v.author))*

*A negation can be avoided by using an aggregate, e.g. having(max(score) < 10.)*

2) (SAME SCHEMA AS LAST WEEK) A database with the following schema stores a collection of Webpages and the words they contain, and a collection of dictionaries in several languages and the words in those languages:

Occurs(url, word)
Dictionary(language, word)

- url represents a Webpage.
- Every Webpage may contain several words, and every word may occur in several Webpages.
- Every language may contain several words, and every word may occur in several languages
- There are no nulls in the database.


Write a SQL query that computes, for each Webpage, the largest number of words on that page in any language. For example, if a page has 100 words in French and 50 words in English (these two sets of words may be overlapping), then your query will return 100 for that page. If a Webpage has only words that do not occur in any language at all, then you do not need to return that Webpage.


*select url, max(c)*
*from (select x.url, y.language, count(*) as c*
      *from Occur x, Dictionary y*
      *where x.word = y.word*
      *group by x.url, y.language)*
*group by url*

3) Consider the following social network database:

Person(pid, name)
Relationship(pid1, pid2, type)

Where:

- Person.pid is a key.
- Relationship.pid1 and Relationship.pid2 are foreign keys to Person.
- Relationship.type is either 'friend' or 'enemy'.

Keep in mind that Relationship is not symmetric: if p1 is a friend of p2, that does not mean p2 is a friend of p1. It is not transitive either: if p1 is a fried of p2 who is a friend of p3, it doesn't mean p1 is a fried of p3.

(a) (12 points) A second degree friend is the friend of a friend.  Write a SQL query that computes for each person the total number of their second degree friends. Your query should return answers of the form: pid, name, count. Cryptic hint: "not every person has friends, but you have to count everyone's second degree friends".


*select x.pid, x.name, count(distinct z.pid2)*
*from Person x left outer join Relationship y*
*            left outer join Relationship z*
*on x.pid = y.pid1 and y.pid2 = z.pid1*
*and y.type = 'friend' and z.type = 'friend'*
*group by x.pid, x.name*

(b) (12 points) Write a SQL query that returns all persons who have at least 12 common friends with "Mary". Your query should return answers of the form: pid1, pid2, name, where pid1 is Mary's pid and pid2 is that of a person who has 12 or more common friends (meaning there are at least 12 persons p such that pid1 and p are friends, and pid2 and p are friends). If there are multiple people called Mary, then you will report each of them.

*select x.pid, y.pid, y.name*
*from Person x, Person y, Relationship u, Relationship v*
*where x.pid = u.pid1 and y.pid = v.pid1 and u.pid2 = v.pid2*
*and u.type = 'friend' and v.type = 'friend*
*and x.name = 'Mary'*
*group by x.pid, y.pid, y.name*
*having count(*) >= 12*

(c) (12 points) Fred says: "my enemies' enemies are my friends". Prove that Fred is wrong: write a query that returns all Fred's enemies' enemies that are not his friends. Your query should return answers of the form: pid1, pid2, where pid1 is Fred's pid and pid2 represents an enemy's enemy that is not Fred's friend.

*select x.pid, z.pid2*
*from Person x, Relationship y, Relationship z*
*where x.name = 'Fred'*
*and x.pid = y.pid1 and y.pid2 = z.pid1*
*and y.type = 'enemy' and z.type = 'enemy'*
*and not exists (select * from Relationship u*
                *where u.type = 'friend'*
                *and u.pid1 = y.pid1 and u.pid2 = z.pid2)*