The 201Cats web application provides sophisticated cat video viewing to its users. Each user has a user name and logs in the 201Cats using his Facebook log-in. Consequently, the company regularly obtains information of which ones of the 201Cats users are Facebook followers of other 201Cats users.

When a user logs in, the web application suggests to her 10 cat videos – more on this below. The user may

- Watch one of the suggested videos
- Like a suggested video; may like a video even without watching it. A user may like a video just once. Clicking many times on the like does not result on "liking many times".

The 201Cats database captures the following information, with minimum redundancy:

- The user's name and Facebook login – password not needed.
- The user's "like" activity: store which video were liked and when.
- The user's "watch" activity: store which videos were watched and when.
- The times the user logged in and the videos that were suggested to the user to watch when she logged in.
- Which 201Cats users are friends of each user. You are allowed some redundancy here: It is OK if the database captures both that "X is friend of Y" and "Y is friend of X", despite the fact that this is redundant since Facebook friendships are symmetric.

Due Friday 1/22 midnight: Produce an SQL schema that captures the above information. Optionally (and not graded), submit the corresponding E/R design – if you designed the schema using the E/R technique.

201Cats gives to the logged-in user X many options for controlling how the Top-10 cat videos are selected.

Option "Overall Likes": The Top-10 cat videos are the ones that have collected the highest numbers of likes, overall.

Option "Friend Likes": The Top-10 cat videos are the ones that have collected the highest numbers of likes from the friends of X.

Option "Friends-of-Friends Likes": The Top-10 cat videos are the ones that have collected the highest numbers of likes from friends and friends-of-friends.

Option "My kind of cats": The Top-10 cat videos are the ones that have collected the most likes from users who have liked at least one cat video that was liked by X.

Option "My kind of cats – with preference (to cat aficionados that have the same tastes)": The Top-10 cat videos are the ones that have collected the highest sum of *weighted* likes from every other user Y (i.e., given a cat video, each like on it, is multiplied according to a weight).The weight is the *log cosine lc(X,Y)* defined as follows: Conceptually, there is a vector v_x for each user Y, including the logged-in user

X. The vector has as many elements as the number of cat videos. Element *i* is 1 if Y liked the *i*th cat video; it is 0 otherwise. For example, if 201Cats has five cat videos and user 21 liked only the 1st and the 4th, the v_{21}=<1,0,0,1,0>, i.e., $v_{21}[1]=v_{21}[4]=1$ and $v_{21}[2]= v_{21}[3]=v_{21}[5]=0$. Assuming there are *N* cat videos, the log cosine *lc(X, Y)* is

$$lc(X,Y) = \log\,(1 + \sum_{\{i=1,\ldots,N\}} (v_X[i] \times v_Y[i])\,)$$

Due Friday 2/5 midnight: Each of the options is essentially a query that finds top-10 cat videos. Write and submit the queries.

Your next goal is to deliver top performance for the queries. Expect a significant amount of experimentation until you tune the performance.

In particular, create any indices that will be beneficially used by the queries. Do not create useless indices. Deploy your solution on the provided Amazon instances (and databases) and measure the performance of the Exercise 2 queries on cold runs.

Due Friday 2/26 midnight: Submit your index choices (in the form of the script with the CREATE INDEX statements) and the measured query performance on the Amazon instance. The submission will be followed by a demo and discussion with Monica or Yannis, where you will justify your choices.

Your next goal will be to improve the performance of the "My Kind of cats – with preference" option by appropriate precomputations. It is understood that the maintenance of the precomputed tables will lead to some slowdown while viewers interact. Your precomputation choices must be such that the precomputations that you introduce collectively save more time to this option than they cost to maintain. Precomputed tables will also benefit from creating certain indices on them. Build any beneficial indices. Calibrate your solution against cold runs of the activity script provided with the Amazon instance.

Due Friday 3/11 midnight: Submit the following. The submission will be followed by a demo and discussion with Monica or Yannis, where you will justify your choices.

- Which precomputed tables you created: CREATE TABLE statement and query that initially loaded it.
- Your new "My Kind of cats – with preference" query, which makes the best use of the precomputed tables you chose.
- Which indices you created (CREATE INDEX scripts)