A1

Part 1

1. Find all the users who have never liked or viewed a post or story of a user that they do not follow. Report their user id and “about” information. Put the information into a relation with attributes “username” and “description”.

Relations(feruid, feduid):= Π follower, followed (Follows)

Post\_relations:= Π liker, uid (Likes ⋈ Post)

Liked\_stranger(uid):=

σ Relations.feruid = Post\_relations.liker and Relations.feduid ≠ Post\_relations.uid (Relations X Post\_relations)

Post\_answer:= Π uid (Users) – Liked\_stranger

Story\_relations:= Π viewerid, uid (Saw ⋈ Story)

Saw\_stranger(uid):=

σ Relations.feruid = Story\_relations.viewerid and Relations.feduid ≠ Story\_relations.uid (Relations X Story\_relations)

Story\_answer:= Π uid (Users) – Saw\_stranger

Never\_stranger:= Post\_answer ∩ Story\_answer

Final\_answer(Username, Description):= Π uid, about (Never\_stranger ⋈ User)

1. Find every hashtag that has been mentioned in at least three post captions on every day of 2017. You may assume that there is at least one post on each day of a year.

Post\_2017:= σwhen.year = 2017(Post)

Hashtag\_2017:= Post\_2017 ⋈ Hashtag

Three\_times\_same\_day:=

σ h1.tag = h2.tag = h3.tag and h1.when.day = h2.when.day = h3.when.day (ρh1(Hashtag\_2017) X ρh2(Hashtag\_2017) X ρh2(Hashtag\_2017))

1. Let’s say that a pair of users are “reciprocal followers” if they follow each other. For each pair of reciprocal followers, find all of their “uncommon followers”: users who follow one of them but not the other. Report one row for each of the pair’s uncommon follower. In it, include the identifiers of the reciprocal followers, and the identifier, name and email of the uncommon follower.

Pair(one, two):= σ f1.follower = f2.followed and f1.followed = f2.follower and f1.follower > f1.followed (ρf1(Follows) X ρf2(Follows))

-- f1.follower > f1.followed to eliminate duplicates

Follow\_one:= Π follower, one, two(σ one = followed (Pair X Follows))

Follow\_two:= Π follower, one, two(σ two = followed (Pair X Follows))

Follow\_one\_but\_not\_other(uid, one, two):= Π follower, one, two (Follow\_one ∪ Follow\_two – (Follow\_one ∩ Follow\_two))

Final\_answer(pair\_one, pair\_two, uid, name, email):=Π one, two, User.uid, name, email (Follow\_one\_but\_not\_other ⋈ User)

1. Find the user who has liked the most posts. Report the user’s id, name and email, and the id of the posts they have liked. If there is a tie, report them all.

Cannot be expressed.

1. Let’s say a pair of users are “backscratchers” if they follow each other and like all of each others’ posts. Report the user id of all users who follow some pair of backscratcher users.

Pair(one, two):= σ f1.follower = f2.followed and f1.followed = f2.follower and f1.follower > f1.followed (ρf1(Follows) X ρf2(Follows))

-- f1.follower > f1.followed to eliminate duplicates

Total\_post\_one:= Πpid (σ uid = one (Pair X Post))

Total\_post\_two:= Πpid (σ uid = two (Pair X Post))

One\_like:= ΠTotal\_post\_two.pid (σliker = one (Total\_post\_two ⋈ Likes X Pair))

Two\_like:= Π Total\_post\_one.pid (σliker = two (Total\_post\_one ⋈ Likes X Pair))

Not\_backscratchers\_two:= Total\_post\_one ∪ Two\_like – (Total\_post\_one ∩ Two\_like)

Not\_backscratchers\_one:= Total\_post\_two ∪ One\_like – (Total\_post\_two ∩ One\_like)

Backscratchers:= Pair – Not\_backscratchers\_two - Not\_backscratchers\_one

Follow\_one:= Π follower, one, two(σ one = followed (Backscratchers X Follows))

-- check is follow one’s backscratcher

Follow\_backscratcher(uid):= Π Follow\_one.follower(σ Follow\_one.follower = Follows.follower and two = followed (Follow\_one X Follows))

1. The “most recent activity” of a user is his or her latest story or post. The “most recently active user” is the user whose most recent activity occurred most recently. Report the name of every user, and for the most recently active user they follow, report their name and email, and the date of their most-recent activity. If there is a tie for the most recently active user that a user follows, report a row for each of them.

Post\_newest:= Π Story.uid, Post.when(σ Story.uid = Post.uid and Story.when < Post.when (Story X Post))

Story\_newest:= Π Story.uid, Story.when(σ Story.uid = Post.uid and Story.when > Post.when (Story X Post))

Recent:= Post\_newest ∪ Story\_newest

Follow\_pair\_with\_recent(follower, followed, recent\_act):=

Π User.uid, Recent.uid, Recent.when (σ User.uid = Follow.follower and Follow.followed = Recent.uid (User X Follow X Recent))

Not\_most\_recent\_followed:=

Π f1.follower, f1.followed, f1.recent\_act (σ f1.follower = f2.follower and f1.recent\_act < f2.recent\_act (ρf1(Follow\_pair\_with\_recent) X ρf2(Follow\_pair\_with\_recent)))

Most\_recent:= (Follow\_pair\_with\_recent – Not\_most\_recent\_followed)

Answer(uid, followed\_name, email, recent\_activity):=

Π Most.recent.follower, User.name, User.email, Most\_recent.when (σMost\_recent.followed = User.uid (Most\_recent X User))

Final\_answer(Name, followed\_name, email, recent\_activity):=

Π User.name, Answer.followed\_name, Answer.email, Answer.recent\_activity (σAnswer.uid = User.uid (Answer X User))

1. Find the users who have always liked posts in the same order as the order in which they were posted. Report the user’s name and email.

Post\_and\_likes(pid, liker, when\_p, when\_l):=

ΠPost.pid, Likes.liker, Post.when, Likes.when (σPost.pid = Likes.pid (Post X Likes))

Not\_same\_order(uid):= Πp1.liker (σp1.liker = p2.liker and p1.pid < p2.pid and ((p1.when\_p < p2.when\_p and p1.when\_l > p2.when\_l) or (p1.when\_p > p2.when\_p and p1.when\_l < p2.when\_l)) (ρp1(Post\_and\_likes) X ρp2(Post\_and\_likes)))

Same\_order:= Π User.name, User.email ((ΠUser.uid (User) – Not\_same\_order) ⋈ User)

1. Report the name and email of the user who has gained the greatest number of new followers in 2017. If there is a tie, report them all.

Cannot be expressed.

1. For each user who has ever viewed any story, report their id and the id of the first and of the last story they have seen. If there is a tie for the first story seen, report both; if there is a tie for the last story seen, report both. This means that a user could have up to 4 rows in the resulting relation.

-- first.when < last.when

Not\_last:= Πs1.viewerid, s1.sid (σs1.viewerid = s2.viewerid and s1.sid ≠ s2.sid and s1.when < s2.when (ρs1(Saw) X ρs2(Saw)))

Not\_first:= Πs1.viewerid, s1.sid (σs1.viewerid = s2.viewerid and s1.sid ≠ s2.sid and s1.when > s2.when (ρs1(Saw) X ρs2(Saw)))

Last:= Πviewerid, sid (Saw) – Not\_last

First:= Πviewerid, sid (Saw) – Not\_first

Final\_answer(uid, first, last):= ΠFirst.viewerid, First.sid, Last.sid (σFirst.viewerid = Last.viewerid (First X Last))

1. Find posts that have at least three comments and for which there has been a sentiment shift over time. For each post, report the user who owns it and, for each comment on the post, the commenter’s id, the date of their comment and its sentiment.

Comment\_at\_least\_3:= Πc1.pid (σc1.pid = c2.pid = c3.pid and (c1.comenter ≠ c2.commenter or c1.when ≠ c2.when) and (c3.comenter ≠ c2.commenter or c3.when ≠ c2.when) and (c1.comenter ≠ c3.commenter or c1.when ≠ c3.when) (ρc1(Comment) X ρc2(Comment) X ρc3(Comment)))

Post\_with\_comment:= ΠComment.pid, Comment.commenter, Comment.when, Comment.text (σComment\_at\_least\_3.pid = Comment.pid (Comment\_at\_least\_3 X Comment))

Post\_with\_no\_shift:= Πp1.pid(σp1.pid = p2.pid= p3.pid and p1.when < p2.when < p3.when and ((sentiment(p1.text) = positive and sentiment(p2.text) = negative and sentiment(p3.text) = positive) or (sentiment(p1.text) = negative and sentiment(p2.text) = positive and sentiment(p3.text) = negative)) (ρp1(Post\_with\_comment) X ρp2(Post\_with\_comment) X ρp3(Post\_with\_comment)))

Post\_with\_shift:= (Πpid (Comment) – Post\_with\_no\_shift)

Final\_answer(post, owner, commenter, date, sentiment):= ΠComment.pid, Post.uid, Comment.commenter, Comment.when, sentiment(Comment.text) (σPost.pid = Comment.pid (Post X (σPost\_with\_shift.pid = Comment.pid (Post\_with\_shift X Comment))))

Part 2

1. Violator:= σPost.pid = Comment.pid and Post.when > Comment.when (Post X Comment)

Violator = ∅

1. Violator:= σs1.when < s2.when and s1.current = “yes” (ρs1(Story) X ρs2(Story))

Violator = ∅

1. Post[pid] <= PInclude[pid]

Story[sid] <= SInclude[sid]