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):= $\Pi_{\text{follower, followed}}$ (Follows)
   Post_relations:= $\Pi_{\text{liker, uid}}$ (Likes $\bowtie$ Post)
   Liked_stranger(uid):=

   $\sigma_{\text{Relations.feruid = Post\_relations.liker and Relations.feduid} \neq \text{Post\_relations.uid}}$ (Relations X Post_relations)

   Post_answer:= Π_{uid} (Users) – Liked_stranger

   Story_relations:= $\Pi_{\text{viewerid, uid}}$ (Saw $\bowtie$ Story)

   Saw_stranger(uid):=

   $\sigma_{\text{Relations.feruid = Story\_relations.viewerid and Relations.feduid} \neq \text{Story\_relations.uid}}$ (Relations X Story_relations)

   Story_answer:= Π_{uid} (Users) – Saw_stranger

   Never_stranger:= Post_answer $\cap$ Story_answer

   Final_answer(Username, Description):= $\Pi_{\text{uid, about}}$ (Never_stranger $\bowtie$ User)

2. 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:= $\sigma_{\text{when.year = 2017}}$(Post)
   Hashtag_2017:= Post_2017 $\bowtie$ Hashtag
   Three_times_same_day:=

   $\sigma_{\text{h1.tag = h2.tag = h3.tag and h1.when.day = h2.when.day = h3.when.day}}$ (ρ_{h1}(Hashtag_2017) X ρ_{h2}(Hashtag_2017) X ρ_{h2}(Hashtag_2017))

3. 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):= $\sigma_{f1.follower = f2.followed \text{ and } f1.followed = f2.follower \text{ and } f1.follower > f1.followed}$ (ρ_{f1}(Follows) X ρ_{f2}(Follows))

   -- $_{f1.follower > f1.followed}$ to eliminate duplicates

   Follow_one:= $\Pi_{follower, one, two}$($\sigma_{one = followed}$ (Pair X Follows))
   Follow_two:= $\Pi_{follower, one, two}$($\sigma_{two = followed}$ (Pair X Follows))
   Follow_one_but_not_other(uid, one, two):= $\Pi_{follower, one, two}$ (Follow_one $\cup$ Follow_two – (Follow_one $\cap$ Follow_two))
   Final_answer(pair_one, pair_two, uid, name, email):=$\Pi_{one, two, User.uid, name, email}$ (Follow_one_but_not_other $\bowtie$ User)

4. 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.

5. 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):= $\sigma_{f1.follower = f2.followed \text{ and } f1.followed = f2.follower \text{ and } f1.follower > f1.followed}$ (ρ_{f1}(Follows) X ρ_{f2}(Follows))

   -- $_{f1.follower > f1.followed}$ to eliminate duplicates

Total_post_one:= Π_{pid} (σ uid = one (Pair X Post))
Total_post_two:= Π_{pid} (σ uid = two (Pair X Post))
One_like:= $\Pi_{Total\_post\_two.pid}$ ($\sigma_{liker = one}$ (Total_post_two ⋈ Likes X Pair))
Two_like:= $\Pi_{Total\_post\_one.pid}$ ($\sigma_{liker = two}$ (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:= $\Pi_{follower,\ one,\ two}$($\sigma_{one\ =\ followed}$ (Backscratchers X Follows))
-- check is follow one's backscratcher
Follow_backscratcher(uid):= $\Pi_{Follow\_one.follower}$($\sigma_{Follow\_one.follower\ =\ Follows.follower\ and\ two\ =\ followed}$ (Follow_one X Follows))

6. 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:= $\Pi_{Story.uid,\ Post.when}$($\sigma_{Story.uid\ =\ Post.uid\ and\ Story.when\ <\ Post.when}$ (Story X Post))
Story_newest:= $\Pi_{Story.uid,\ Story.when}$($\sigma_{Story.uid\ =\ Post.uid\ and\ Story.when\ >\ Post.when}$ (Story X Post))
Recent:= Post_newest ∪ Story_newest
Follow_pair_with_recent(follower, followed, recent_act):=
$\Pi_{User.uid,\ Recent.uid,\ Recent.when}$ ($\sigma_{User.uid\ =\ Follow.follower\ and\ Follow.followed\ =\ Recent.uid}$ (User X Follow X Recent))
Not_most_recent_followed:=
$\Pi_{f1.follower,\ f1.followed,\ f1.recent\_act}$ ($\sigma_{f1.follower\ =\ f2.follower\ and\ f1.recent\_act\ <\ f2.recent\_act}$ (ρ_{f1}(Follow_pair_with_recent) X ρ_{f2}(Follow_pair_with_recent)))

Most_recent:= (Follow_pair_with_recent – Not_most_recent_followed)

Answer(uid, followed_name, email, recent_activity):=

$\Pi_{\text{Most.recent.follower, User.name, User.email, Most\_recent.when}}$ ($\sigma_{\text{Most\_recent.followed = User.uid}}$ (Most_recent X User))

Final_answer(Name, followed_name, email, recent_activity):=

$\Pi_{\text{User.name, Answer.followed\_name, Answer.email, Answer.recent\_activity}}$ ($\sigma_{\text{Answer.uid = User.uid}}$ (Answer X User))

7. 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):=

   $\Pi_{\text{Post.pid, Likes.liker, Post.when, Likes.when}}$ ($\sigma_{\text{Post.pid = Likes.pid}}$ (Post X Likes))

   Not_same_order(uid):= $\Pi_{\text{p1.liker}}$ ($\sigma_{\text{p1.liker = p2.liker and p1.pid < p2.pid and ((p1.when\_p < p2.when\_p and p1.when\_l > p2.when\_l) or (p1.when\_p > p2.when\_p and p1.when\_l < p2.when\_l))}}$ (ρ_{p1}(Post_and_likes) X ρ_{p2}(Post_and_likes)))

   Same_order:= $\Pi_{\text{User.name, User.email}}$ (($\Pi_{\text{User.uid}}$ (User) – Not_same_order) ⋈ User)

8. 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.

9. 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:= $\Pi_{\text{s1.viewerid, s1.sid}}$ ($\sigma_{\text{s1.viewerid = s2.viewerid and s1.sid} \neq \text{s2.sid and s1.when < s2.when}}$ (ρ_{s1}(Saw) X ρ_{s2}(Saw)))

Not_first:= $\Pi_{\text{s1.viewerid, s1.sid}}$ ($\sigma_{\text{s1.viewerid = s2.viewerid and s1.sid} \neq \text{s2.sid and s1.when > s2.when}}$ (ρ_{s1}(Saw) X ρ_{s2}(Saw)))

Last:= $\Pi_{\text{viewerid, sid}}$ (Saw) – Not_last

First:= $\Pi_{\text{viewerid, sid}}$ (Saw) – Not_first

Final_answer(uid, first, last):= $\Pi_{\text{First.viewerid, First.sid, Last.sid}}$ ($\sigma_{\text{First.viewerid = Last.viewerid}}$ (First X Last))

10. 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:= $\Pi_{\text{c1.pid}}$ ($\sigma_{\text{c1.pid = c2.pid = c3.pid and (c1.comenter} \neq \text{c2.commenter or c1.when} \neq \text{c2.when) and (c3.comenter} \neq \text{c2.commenter or c3.when} \neq \text{c2.when) and (c1.comenter} \neq \text{c3.commenter or c1.when} \neq \text{c3.when)}}$ (ρ_{c1}(Comment) X ρ_{c2}(Comment) X ρ_{c3}(Comment)))

Post_with_comment:= $\Pi_{\text{Comment.pid, Comment.commenter, Comment.when, Comment.text}}$ ($\sigma_{\text{Comment\_at\_least\_3.pid = Comment.pid}}$ (Comment_at_least_3 X Comment))

Post_with_no_shift:= $\Pi_{\text{p1.pid}}$($\sigma_{\text{p1.pid = p2.pid= p3.pid and p1.when < p2.when < p3.when and ((sentiment(p1.text) = positive and sentiment(p2.text) = negative and sentiment(p3.text) = positive) or (sentiment(p1.text) = negative and sentiment(p2.text) = positive and sentiment(p3.text) = negative))}}$ (ρ_{p1}(Post_with_comment) X ρ_{p2}(Post_with_comment) X ρ_{p3}(Post_with_comment)))

Post_with_shift:= (Π_{pid} (Comment) – Post_with_no_shift)

Final_answer(post, owner, commenter, date, sentiment):= $\Pi_{\text{Comment.pid, Post.uid, Comment.commenter, Comment.when, sentiment(Comment.text)}}$ ($\sigma_{\text{Post.pid = Comment.pid}}$ (Post X ($\sigma_{\text{Post\_with\_shift.pid = Comment.pid}}$ (Post_with_shift X Comment))))

Part 2

1. Violator:= $\sigma_{\text{Post.pid = Comment.pid and Post.when > Comment.when}}$ (Post X Comment)
   Violator = Ø
2. Violator:= $\sigma_{\text{s1.when < s2.when and s1.current = “yes”}}$ (ρ_{s1}(Story) X ρ_{s2}(Story))
   Violator = Ø
3. Post[pid] <= PInclude[pid]
   Story[sid] <= SInclude[sid]