Part 1

Question 1.

- Get each post's pid and the creator's uid.

$PostIds(pid, uid) := \Pi_{pid,uid}\ Post$

- Get each story's pid and the creator's uid.

$StoryIds(pid, uid) := \Pi_{sid,uid} Story$

- Get the liker's uid and the post creator's uid.

$Postliker(liker, uid) := \Pi_{liker,uid}\ (Likes \bowtie PostIds)$

- Get the viewer's uid and the story creator's uid.

$StoryViewer(viewer, uid) := \Pi_{viewerid,uid}\ (Saw \bowtie StoryIds)$

- Get all likers and viewers' uid and the corresponding creators' uid

$allProducer(follower, followed) := \Pi_{p.liker,p.uid}(\sigma_{p.liker=s.viewerid}\ (\rho_p\ Postliker \times \rho_s\ StoryViewer))$

- Get the uid of the likers and viewers who have liked or viewed a person they do not follow.

$notMatch(uid) := \Pi_{follower}\ (allProducer - (\Pi_{follower,followed}\ (allProducer \bowtie Follows)))$

- Get the uid that matches Q1's requirement.

$Match(uid) := \Pi_{uid}\ User - notMatch$

- Get the description information of the uids in Match.

Then the requested set can be expressed as follow:

$Result(username, description) := \pi_{uid,about}\ (Match \bowtie User)$

Question 2.

- Get all the tags with their pids and dates.

$PostTag(pid, tag, when) := \Pi_{pid,tag,when}(Post \bowtie Hashtag)$

- Get tags with same tag names, same dates but different pids.

Then the requested set can be expressed as follow:

$Result(tag) := \Pi_{p1.tag}(\sigma_{p1.tag=p2.tag \wedge p1.when=p2.when \wedge p1.pid \neq p2.pid}\ (\rho_{p1} PostTag \bowtie \rho_{p2}\ PostTag))$

Question 3.

- Get the uid of all reciprocal users.

$Reciprocal(user1, user2) := \Pi_{F1.follower,F1.followed} (\sigma_{F1.follower=F2.followed \land F1.followed=F2.follower \land F1.follower<F1.followed}(\rho_{F1}\ Follows \bowtie \rho_{F2}\ Follows))$

- Get the first reciprocal user’s followers.

$Follow1\ (user1, user2, follower) := \Pi_{user1,user2,follower}(\sigma_{user1=followed}\ (reciprocal \times Follows))$

- Get the second reciprocal user’s followers.

$Follow2\ (user1, user2, follower) := \Pi_{user1,user2,follower}(\sigma_{user2=followed}\ (reciprocal \times Follows))$

- Get the uncommon followers (only follow user1 or only follow user2).

$Uncommon\ (user1, user2, follower) := (folllow1 \cup follow2) - (folllow1 \cap follow2)$

- Get the names and emails of these uncommon followers.

Then the requested set can be expressed as follow:

$Result\ (user1, user2, follower, name, email) := \Pi_{user1,user2,follower,name,email}(\sigma_{follower=uid}\ (uncommon \times user))$

Question 4.

Cannot be expressed. This is possible by SQL but not the mere use of the operators listed above.

Question 5.

$Recip(user1, user2) := \Pi_{F1.follower,F1.followed}(\sigma_{F1.follower=F2.followed \land F1.followed=F2.follower}(\rho_{F1}\ Follows\ \times \rho_{F2}\ Follows))$

$AllLikeRel(liker, liked, pid) := \Pi_{Likes.liker,Post.uid,Post.pid}(Post\ \bowtie Likes)$

$RecLikeRel(liker, liked, pid) := \Pi_{AllLikeRel.liker,Recip.user2,pid}\sigma_{AllLikeRel.liker=Recip.user1}(AllLikeRel\ \times Recip)$

$NotRecLikedPost(poster, pid) := \Pi_{uid,pid}Post - \Pi_{user2,pid}RecLikeRel$

$NotBack(user1, user2) := \Pi_{user1, user2} \sigma_{NotRecLikedPost.poster=Recip.user1 \vee NotRecLikedPost.poster=user2} (NotRecLikedPost \bowtie Recip)$

$BackScratcher\ (user1, user2) := Recip - NotBack$

$FollowPair(follower, followed1, followed2) := \sigma_{F1.follower=F2.follower} (\rho_{F1} Follow \times \rho_{F2} Follow)$

Then the requested set can be expressed as follow:
$Result\ (user)$:
$= \Pi_{F1.follower} \sigma_{FollowPair.followed1=BackScratcher.user1 \wedge FollowPair.followed2=BackScratcher.user2} (FollowPair \times BackScratcher)$

Question 6.
$FollowedPost \coloneqq \sigma_{followed=uid} (Follows \times Post)$

$NotMostRecent \coloneqq \sigma_{T_1.when > T_2.when} \left(\rho_{T_1} FollowedPost \times \rho_{T2} FollowedPost \right)$

$MostRecent \coloneqq \Pi_{T_1.follower, T_1.followed, T_1.when} \left(\rho_{T_1} FollowedPost \times \rho_{T_2} FollowedPost - NotMostRecent \right)$

Then the request set can be expressed as follow:
$Result(user, followed, email, time) := \Pi_{follower, followed, email, when} \sigma_{MostRecent.followed=User.uid} (MostRecent \times User)$

Question 7.
Cannot be expressed.

Question 8.

$NotLatest\ (Commenter, pid) := \Pi_{C2.commenter, C2.pid} \sigma_{C1.commenter=C2.commenter \wedge C1.when > C2.when} (\rho_{C1} Comment \times \rho_{C2} Comment)$

$NotEarliest\ (Commenter, pid) := \Pi_{C2.commenter, C2.pid} \sigma_{C1.commenter=C2.commenter \wedge C1.when < C2.when} (\rho_{C1} Comment \times \rho_{C2} Comment)$

$Latest(uid, pid) := \Pi_{commenter, pid} Comment - NotLatest$

$Earliest(uid, pid) := \Pi_{commenter,pid} Comment - NotEarliest$

Then the requested set can be expressed as follow:
$Result(uid, earliest, latest) := \Pi_{uid,Earliest.pid,Latest.pid}(Earliest \ \bowtie Latest)$

Part 2

1. $\sigma_{Story.when<Saw.when}(Story \ \bowtie Saw) = \emptyset$
2. $\sigma_{T_1.uid=T_2.uid \ \wedge T_1.current=\text{"yes"} \wedge T_2.current="yes"}\left(\rho_{T_1} Story \ \bowtie \ \rho_{T_2} Story\right) = \emptyset$