Schema 1

Queries: (5 points each)

1. Find the clinic name and department name for all departments whose head is also head of another department.

We join relation Department with itself on attribute dHead and select tuples with different departmentIDs.

Answer:

$$\pi_{hName,dName}((\sigma_{departmentID \neq dID}(Department \bowtie \rho_{dID \leftarrow departmentID, hID \leftarrow clinicID, dN \leftarrow dName}(Department))) \bowtie Clinic)$$

2. Find the first name and last name of all Physicians whose patients are never in the clinic for less than 10 days.

(They must have had patients admitted in the clinic at some time.)

Answer:

First we find physicians who had some patients in clinic for less than 10 days:

$$PhywithSomePatients :=$$

$$\pi_{prPhysician}(\sigma_{prPhysician = prInDate < 10}(PatientRoomAssignment))$$

Then we subtract the above physicians from the set of all physicians to get the ones who never had a patient in clinic for less than 10 days:

$$PhywithnosuchPatients: = \pi_{prPhysician}(PatientRoomAssignment) - PhywithSomePatients$$

Now we join the above relation with Employee to get the names:

$$\pi_{eFirstName,eLastName}(\rho_{prPhysician \leftarrow employeeID}(\text{Employee}) \bowtie PhywithnosuchPatients)$$

3. Find the first and last name of all patients who have exactly two physicians in the clinic's system.

(Note that physicians are specified in both appointments and room assignment.)

Answer:

First we create a temporary relation containing all the patients and their Physicians.

This information can be found in both $PatientRoomAssignment$ and $PatientAppointment$;

relations:

$$PatientPhysician :=$$
$$\pi_{patientID,prPhysician}(PatientRoomAssignment) \cup$$
$$\rho_{prPhysician \leftarrow employeeID}(\pi patientID, employeeID$$
$$(\sigma_{dRole='Physician'}(Employee \bowtie PatientAppointment)))$$

We next find patients who have at least two Physicians:

$$PatientwithtwoPhy := \pi_{patientID}(\sigma_{prPhysician \neq pD}$$
$$(PatientPhysician \bowtie \rho_{pD \leftarrow prPhysician}(PatientPhysician)))$$

Now we find patients with at least three Physicians:

$$PatientsatleastthreePhy := \pi_{patientID}(\sigma_{prPhysician \neq pD1 \wedge prPhysician \neq pD2 \wedge pD2 \neq pD1}$$
$$(PatientPhysician \bowtie \rho_{pD1 \leftarrow prPhysician}(PatientPhysician)$$
$$\bowtie \rho_{pD2 \leftarrow prDoctor}(PatientPhysician)))$$

Patients with *exactly* two physicians are those who are in $PatientwithtwoPhy$ but not in $PatientsatleastthreePhy$:

$$Final := \pi_{pFirstName,pLastName}$$
$$(P \bowtie (PatientwithtwoPhy - PatientsatleastthreePhy))$$

4. Find the clinic and department name for the department at which there were no appointments from April 3, 2019 to April 8, 2019.

We first find departments which had some appointments between April 3, 2019 to April 8, 2019.

$$SomeAppointments :=$$
$$\pi_{departmentID}(\sigma_{paDate > '2019-04-02' \wedge paDate < '2019-04-09'}(PA))$$

Departments with no appointment in that period can be found by subtracting $SomeAppointments$ from the set of all departments:

$$Final := \pi_{hName,dName}(Clinic \bowtie Department \bowtie$$
$$(\pi_{departmentID}(Department) - SomeAppointments))$$

5. Find the clinic, room number, and patient first and last names of all semi-private (size = 2) rooms which were occupied by female patients who were over 50 on May 13, 2019.

We first select rooms of size 2:

$$RoomSize2 := \pi_{roomID}\ (\sigma_{rSize\ =\ 2}\ (\ Room\ ))$$

Then we find rooms (and patients) in $RoomSize2$ in which there was a female patient on May 13, 2019 with over 50 years of age:

$$RoomFemalePatient50 := \pi_{\ roomID,patientID}\ (\ \sigma_{prInDate < 2019-05-13\ \wedge\ prOutDate > 2019-05-13\ \wedge\ pSex='F'\ \wedge\ pDOB < 1969-08-13}\ (Patient\ \bowtie\ PatientRoomAssignment\ \bowtie\ RoomSize2))$$

Now we find names of the patients who occupied a room in $RoomFemalePatient50$:

$$Patients := \pi_{\ hName,rNumber,pFirstName,pLastName}\ (H\ \bowtie\ R\ \bowtie\ P\ \bowtie\ RoomFemalePatient50))$$

$$Patients := \pi hName, rNumber, pFirstName, pLastName\ (H\ \bowtie\ R\ \bowtie\ P\ \bowtie\ RoomFemalePatient50))$$

Schema 2

Queries (5 points each)

1. For each pair of users that are subscribed to each other, find the videos that they both likeand were uploaded by neither of them. Report only one row for each video the pair likes. Report back user1id, user2id, vid.

$$SubscribedToEachOther := \pi_{U1.subscriber,U2.subscriber}(\sigma_{U1.subscriber=U2.subscribed \wedge U2.subscriber=U1.subscribed \wedge U1.subscriber<U2.subscriber}(\rho_{U1}Subscription \times \rho_{U2}Subscription)$$

$$FirstUserLikedVideos := \pi_{U1.subscriber,U2.subscriber,vid}(\sigma_{uid=U1.subscriber \wedge value=1}(SubscribedToEachOther \times LikeDislikeVideo))$$

$$SecondUserLikedVideos := \pi_{U1.subscriber,U2.subscriber,vid}(\sigma_{uid=U2.subscriber \wedge value=1}(SubscribedToEachOther \times LikeDislikeVideo))$$

$$BothUsersLikedVideos := FirstUserLikedVideos \cap SecondUserLikedVideos$$

$$Answer(user1id, user2id, vid) := \pi_{U1.subscriber,U2.subscriber,Video.vid}(\sigma_{LikeDislikeVideo.vid=Video.vid \wedge uploader \neq U1.subscriber \wedge uploader \neq U2.subscriber}(BothUsersLikedVideos \times Video))$$

2. For each user, report their playlists that do not contain any videos they dislike. Report backuid, uname, pid, pname.

$$Disliked := \sigma_{value=-1}(LikeDislikeVideo)$$

$$PlaylistsWithVidsDislikedByCreator := \pi_{Playlist.uid,Playlist.pid,Playlist.pname}(Disliked \bowtie Playlist \bowtie PlaylistItem)$$

$$AllPlaylists = := \pi_{Playlist.uid,Playlist.pid,Playlist.pname}(Playlist)$$

$$PlaylistsWithoutVidsDislikedByCreator := AllPlaylists - PlaylistsWithVidsDislikedByCreator$$

$$Answer(uid, uname, pid, pname) := \pi_{Playlist.uid,User.uname,Playlist.pid,Playlist.pname}(PlaylistsWithoutVidsDislikedByCreator \bowtie User)$$

3. For each user, report their public playlists that contain only videos they've liked and theuploaders are users they are subscribed to. These playlists should also contain at least one video. Report back uid, pid.

$$PublicPlaylists := \pi_{Playlist.pid,Playlist.uid}(\sigma_{Playlist.privacy="Public"} Playlist)$$

$$PublicPlaylistItems := \pi_{Playlist.pid,Playlist.uid,PlaylistItem.vid}(PlaylistItem \bowtie PublicPlaylists)$$

$$PlaylistCreatorLiked := \sigma_{LikeDislikeVideo.value=1}(LikeDislikeVideo \bowtie_{LikeDislikeVideo.uid=Playlist.uid \wedge LikeDislikeVideo.vid=PlaylistItem.vid} PublicPlaylistItems))$$

$$DislikedOrNotLiked := PublicPlaylistItems - \pi_{Playlist.pid,Playlist.uid,LikeDislikeVideo.vid} PlaylistCreatorLiked$$

$$LikedVideosPlaylists := \pi_{Playlist.pid,Playlist.uid} PublicPlaylistItems - \pi_{Playlist.pid,Playlist.uid} DislikedOrNotLiked$$

$$PublicPlaylistItemsInfo := \pi_{Playlist.pid,Playlist.uid,PlaylistItem.vid,Video.uploader}(Video \bowtie_{Video.vid=PlaylistItem.vid} PublicPlaylistItems)$$

$$VidsCreatorSubUploader := \pi_{Playlist.pid,Playlist.uid,PlaylistItem.vid}(Subscription \bowtie_{Subscription.subscriber=Playlist.uid \wedge Subscription.subscribed=Video.uploader} PublicPlaylistItemsInfo)$$

$$VidCreatorNotSubUploader := PublicPlaylistItems - VidsCreatorSubUploader$$

$$SubbedToUploaderPlaylists := \pi_{Playlist.pid,Playlist.uid} PublicPlaylistItems - \pi_{Playlist.pid,Playlist.uid} VidCreatorNotSubUploader$$

$$Answer(Playlist.pid, Playlist.uid) := LikedVideosPlaylists \cap SubbedToUploaderPlaylists$$

4. For each user, find the videos that they have watched multiple times and at least one of thesetimes was on the day the video was released. These videos must also be at least 8 minutes long. Report back uid, vid, title.

$$VideosWatchedMultipleTimes := \sigma_{(W1.uid=W2.uid \wedge W1.vid=W2.vid \wedge W1.date\_watched \neq W2.date\_watched)} (\rho_{W1}(WatchVideo) \times \rho_{W2}(WatchVideo))$$

$$MultipleWatchesAnd8Minutes := \sigma_{Video.duration \geq 480}(VideosWatchedMultipleTimes \bowtie Video)$$

$$Answer(uid, vid, title) := \pi_{W1.uid,W1.vid,Video.title} (\sigma_{(W1.date\_watched.date=Video.date\_uploaded.date \vee W2.date\_watched.date=Video.date\_uploaded.date)} MultipleWatchesAnd8Minutes)$$

5. Find the playlists such that all videos in the playlist were uploaded before the creation of theplaylist. These playlists should also contain at least one video. Report back uid, pname, date_created.

$$AllVidsInPlaylist := \pi_{Playlist.uid,Playlist.pid,Playlist.pname,Playlist.date\_created}(Playlist \bowtie PlaylistItem \bowtie Video)$$

$$PlaylistsWithVidUploadedAfterCreation := \pi_{Playlist.uid,Playlist.pid,Playlist.pname,Playlist.date\_created} (\sigma_{Playlist.date\_created \leq Video.date\_uploaded}(AllVidsInPlaylist))$$

$$Answer(uid, pname, date\_created) := \pi_{Playlist.uid,Playlist.pname,Playlist.date\_created} (AllVidsInPlaylist - PlaylistsWithVidUploadedAfterCreation)$$

6. For each user, report the users they are subscribed to that they've watched every video of.These users should also have at least one video uploaded. Report back uid, subscribed.

$$SubscriberPairs := \pi_{subscriber,subscribed}(Subscription)$$

$$SubscribedUsersVideos := \pi_{Subscription.subscriber,Subscription.subscribed,Video.vid}(SubscriberPairs \bowtie_{Subscription.subscribed=Video.uploader} (Video))$$

$$VideosSubscribersWatched := WatchVideo \bowtie_{WatchVideo.uid=Subscription.subscriber \wedge WatchVideo.vid=Video.vid} SubscribedUsersVideos$$

$$VideosSubscribersNotWatched := SubscribedUsersVideos - \pi_{Subscription.subscriber,Subscription.subscribed,Video.vid} VideosSubscribersWatched$$

$$Answer(subscriber, subscribed) := \pi_{Subscription.subscriber,Subscription.subscribed} SubscribedUsersVideos - \pi_{Subscription.subscriber,Subscription.subscribed} VideosSubscribersNotWatched$$

7. Find the movies that have been tagged with at least 3 different genres. One of which must be'Action' and we do not want the ones that have been tagged with 'Horror'. Report back mid, title, description.

$$Atleast3Genres := \pi_{M1.mid}(\sigma_{M1.mid=M2.mid \wedge M2.mid=M3.mid \wedge M1.gid \neq M2.gid \wedge M2.gid \neq M3.gid \wedge M1.gid \neq M3.gid} (\rho_{M1}(MovieGenre) \times \rho_{M2}(MovieGenre) \times \rho_{M3}(MovieGenre))$$

$$HorrorMovies := \pi_{mid}(\sigma_{tag='Horror'}(MovieGenre \bowtie Genre))$$

$$ActionMovies := \pi_{mid}(\sigma_{tag='Action'}(MovieGenre \bowtie Genre))$$

$$FinalMovies := (ActionMovies \cap Atleast3Genres) - HorrorMovies$$

$$Answer(mid, title, description) := \pi_{mid,title,description}(FinalMovies \bowtie Movie)$$

8. Find the free movies that are rated 'G', 'PG', or 'PG13' and were never reviewed 1 star by auser after they watched the movie. Report back mid, title, rating.

$$AppropriateFreeMovies := \sigma_{Movie.price=0 \wedge (Movie.rating="G" \vee Movie.rating="PG" \vee Movie.rating="PG13")}(Movie)$$

$$OneStarMovies := ReviewMovie \bowtie_{(ReviewMovie.uid=WatchMovie.uid \wedge ReviewMovie.mid=WatchMovie.mid \wedge ReviewMovie.stars=1 \wedge ReviewMovie.date\_reviewed>WatchMovie.date\_watched}(WatchMovie)$$

$$NoOneStarRatingMovies := (\pi_{mid}Movie) - (\pi_{mid}OneStarMovies)$$

$$Answer(mid, title, rating) := \pi_{Movie.mid,Movie.title,Movie.rating}(AppropriateFreeMovies \bowtie NoOneStarRatingMovies)$$

9. For all paid movies provided by Disney in 2016, find the users that purchased the moviewithin a year of the movie's release date and watched the movie within a year of when they purchased the movie. Assume no movies were released on February 29 2016. Report only one row per user-movie pair. Report back mid, uid, date_released, date_purchased, date_watched. The date_watched reported per user-movie pair should be the first time the user watched that movie.

$$PaidDisney2016Movies := \sigma_{Movie.price>0 \wedge Movie.provider="Disney" \wedge Movie.date\_released.date \leq "2016:12:31" \wedge Movie.date\_released.date \geq "2016:01:01"} Movie$$

$$UsersPurchasedSpecificMovies := \pi_{PurchaseMovie.uid, Movie.date\_released, Movie.mid, PurchaseMovie.date\_purchased}(PurchaseMovie \bowtie_{PurchaseMovie.mid = Movie.mid} PaidDisney2016Movies)$$

$$PurchasedInYrOfRelease := \sigma_{PurchaseMovie.date\_purchased \geq Movie.date\_released \wedge ((Movie.date\_released.date < 2016:02:29 \wedge (Movie.date\_released.date + 366 \geq PurchaseMovie.date\_purchased.date)) \vee (Movie.date\_released.date > 2016:02:29 \wedge (Movie.date\_released.date + 365 \geq PurchaseMovie.date\_purchased.date)))} UsersPurchasedSpecificMovies$$

$$WhenWatchedMovie := WatchMovie \bowtie_{WatchMovie.uid = PurchaseMovie.uid \wedge WatchMovie.mid = Movie.mid} PurchasedInYrOfRelease$$

$$WatchInYrOfPurchase := \sigma_{WatchMovie.date\_watched \geq PurchaseMovie.date\_purchased \wedge ((PurchaseMovie.date\_purchased.date < 2016:02:29 \wedge (PurchaseMovie.date\_purchased.date + 366 \geq WatchMovie.date\_watched.date)) \vee (PurchaseMovie.date\_purchased.date > 2016:02:29 \wedge (PurchaseMovie.date\_purchased.date + 365 \geq WatchMovie.date\_watched.date)))} PurchasedInYrOfRelease$$

$$AllButFirstTimeWatched := (\rho_{WatchInYrOfPurchase1} WatchInYrOfPurchase) \bowtie_{WatchInYrOfPurchase1.date\_watched > WatchMovie.date\_watched \wedge PurchaseMovie.uid = WatchInYrOfPurchase1.uid \wedge Movie.mid = WatchInYrOfPurchase1.mid} WatchInYrOfPurchase$$

$$Answer(mid, uid, date\_watched, date\_purchased, date\_released) := \pi_{Movie.mid, PurchaseMovie.uid, WatchMovie.date\_watched, PurchaseMovie.date\_purchased, Movie.date\_released} WatchInYrOfPurchase - \pi_{Movie.mid, PurchaseMovie.uid, WatchInYrOfPurchase1.date\_watched PurchaseMovie.date\_purchased, Movie.date\_released} AllButFirstTimeWatched$$

10. Find the movies that were released before 2020, are at least 2 hours long, and are taggedwith 'Fantasy' and at least one of 'Action' or 'Adventure'. These movies must also have been given a review for each possible star rating with a non-empty description at least once. Report back mid, title, date_released, duration.

$$TwoHourMoviesBefore2020 := \pi_{mid}(\sigma_{date\_released.date<2020:01:01 \wedge duration \geq 7200} Movie)$$

$$FantasyMovies := \pi_{mid}(\sigma_{tag='Fantasy'}(MovieGenre \bowtie Genre))$$

$$ActionOrAdventureMovies := \pi_{mid}(\sigma_{tag='Action' \vee tag='Adventure'}(MovieGenre \bowtie Genre))$$

$$FantasyAndActionOrAdventureMovies := ActionOrAdventureMovies \cap FantasyMovies$$

$$\begin{aligned} OneStarReviews &:= \pi_{mid}(\sigma_{stars=1 \wedge description \neq ""} ReviewMovie) \\ TwoStarReviews &:= \pi_{mid}(\sigma_{stars=2 \wedge description \neq ""} ReviewMovie) \\ ThreeStarReviews &:= \pi_{mid}(\sigma_{stars=3 \wedge description \neq ""} ReviewMovie) \\ FourStarReviews &:= \pi_{mid}(\sigma_{stars=4 \wedge description \neq ""} ReviewMovie) \\ FiveStarReviews &:= \pi_{mid}(\sigma_{stars=5 \wedge description \neq ""} ReviewMovie) \end{aligned}$$

$$\begin{aligned} AllStarReviews := {} & OneStarReviews \cap TwoStarReviews \\ & \cap ThreeStarReviews \cap FourStarReviews \\ & \cap FiveStarReviews \end{aligned}$$

$$\begin{aligned} FinalMovies := {} & TwoHourMoviesBefore2020 \\ & \cap FantasyAndActionOrAdventureMovies \\ & \cap AllStarReviews \end{aligned}$$

$$Answer(mid, title, date\_released, duration) := \pi_{mid,title,date\_released,duration}(FinalMovies \bowtie Movie)$$

Additional Integrity Constraints

1. All videos in a playlist must be distinct.

$$\begin{aligned} SameVidInPlaylistMultipleTimes := {} & \sigma_{P1.pid=P2.pid \wedge P1.vid=P2.vid \wedge P1.date\_added} \\ & \neq P2.date\_added(\rho_{P1} PlaylistItem \times \rho_{P2} PlaylistItem) = \emptyset \end{aligned}$$

2. If a movie is not free, it can only be purchased by a user after the movie's released date andcan only be watched after they have purchased it. Movies that are free can only be watched by a user after the movie's release date. Movies can only be reviewed by a user after they have watched the movie.

$$PaidMovies := \sigma_{Movie.price>0}(Movie)$$

$$PaidMoviesPurchasedBeforeRelease := PurchaseMovie \bowtie_{PurchaseMovie.date\_purchased < Movie.date\_released \wedge PurchaseMovie.mid=Movie.mid} (PaidMovies)$$

$$PaidMoviesWatchedBeforePurchase = WatchMovie \bowtie_{WatchMovie.date\_watched < PurchaseMovie.date\_purchased \wedge WatchMovie.mid=PurchaseMovie.mid} (PurchaseMovie \bowtie_{PurchaseMovie.mid = Movie.mid} PaidMovies)$$

$$FreeMoviesWatchedBeforeRelease = WatchMovie \bowtie_{WatchMovie.date\_watched < Movie.date\_released \wedge WatchMovie.mid=Movie.mid} (\sigma_{Movies.price=0}(Movie))$$

$$MoviesReviewedBeforeRelease = WatchMovie \bowtie_{WatchMovie.date\_watched > ReviewMovie.date\_reviewed \wedge WatchMovie.mid=ReviewMovie.mid \wedge WatchMovie.uid=ReviewMovie.uid} ReviewMovie$$

$$PaidMoviesPurchasedBeforeRelease \cup PaidMoviesWatchedBeforePurchase \cup FreeMoviesWatchedBeforeRelease \cup MoviesReviewedBeforeRelease = \emptyset$$