LAB 4: SQL exercise

Movie-Rating Query Exercise

Movie (mID, title, year, director) English: There is a movie with ID number mID, a title, a release year, and a director.

Reviewer (rID, name) English: The reviewer with ID number rID has a certain name.

Rating (rID, mID, stars, ratingDate) English: The reviewer rID gave the movie mID a number of stars rating (1-5) on a certain ratingDate.

Queries:

1/ Find the titles of all movies directed by Steven Spielberg.

2/ Find all years that have a movie that received a rating of 4 or 5, and sort them in increasing order.

3/ Find the titles of all movies that have no ratings.

4/ Some reviewers didn't provide a date with their rating. Find the names of all reviewers who have ratings with a NULL value for the date.

5/ Write a query to return the ratings data in a more readable format: reviewer name, movie title, stars, and ratingDate. Also, sort the data, first by reviewer name, then by movie title, and lastly by number of stars.

6/ For all cases where the same reviewer rated the same movie twice and gave it a higher rating the second time,return the reviewer's name and the title of the movie.

7/ For each movie that has at least one rating, find the highest number of stars that movie received. Return themovie title and number of stars. Sort by movie title.

8/ For each movie, return the title and the 'rating spread', that is, the difference between highest and lowestratings given to that movie. Sort by rating spread from highest to lowest, then by movie title.

9/ Find the difference between the average rating of movies released before 1980 and the average rating of movies released after 1980. (Make sure to calculate the average rating for each movie, then the average of those averages for movies before 1980 and movies after. Don't just calculate the overall average rating before and after 1980.)

10/ Find the names of all reviewers who rated Gone with the Wind.

11/ For any rating where the reviewer is the same as the director of the movie, return the reviewer name, movietitle, and number of stars.

12/ Return all reviewer names and movie names together in a single list, alphabetized. (Sorting by the first name of the reviewer and first word in the title is fine; no need for special processing on last names or removing "The".)

13/ Find the titles of all movies not reviewed by Chris Jackson.

14/ For all pairs of reviewers such that both reviewers gave a rating to the same movie, return the names of bothreviewers. Eliminate duplicates, don't pair reviewers with themselves, and include each pair only once. Foreach pair, return the names in the pair in alphabetical order.

15/ For each rating that is the lowest (fewest stars) currently in the database, return the reviewer name, movietitle, and number of stars.

16/ List movie titles and average ratings, from highest-rated to lowest-rated. If two or more movies have thesame average rating, list them in alphabetical order.

17/ Find the names of all reviewers who have contributed three or more ratings. (As an extra challenge, trywriting the query without HAVING or without COUNT.)

18/ Some directors directed more than one movie. For all such directors, return the titles of all movies directed by them, along with the director name. Sort by director name, then movie title. (As an extra challenge, try writing the query both with and without COUNT.)

19/ Find the movie(s) with the highest average rating. Return the movie title(s) and average rating. (Hint: Thisquery is more difficult to write in SQLite than other systems; you might think of it as finding the highest average rating and then choosing the movie(s) with that average rating.)

20/ Find the movie(s) with the lowest average rating. Return the movie title(s) and average rating. (Hint: This query may be more difficult to write in SQLite than other systems; you might think of it as finding the lowest average rating and then choosing the movie(s) with that average rating.)

21/ For each director, return the director's name together with the title(s) of the movie(s) they directed that received the highest rating among all of their movies, and the value of that rating. Ignore movies whose director is NULL.

22/ Add the reviewer Roger Ebert to your database, with an rID of 209.

23/ Insert 5-star ratings by James Cameron for all movies in the database. Leave the review date as NULL.

24/ For all movies that have an average rating of 4 stars or higher, add 25 to the release year. (Update the existing tuples; don't insert new tuples.)

25/ Remove all ratings where the movie's year is before 1970 or after 2000, and the rating is fewer than 4 stars.

Find the names of all reviewers who rated Gone with the Wind.

```
select distinct name
from Rating
join Movie using(mID)
join Reviewer using(rID)
where title = "Gone with the Wind";
```

For any rating where the reviewer is the same as the director of the movie, return the reviewer name, movie title, and number of stars.

```
select distinct name, title, stars
from Rating
join  Movie using(mID)
join Reviewer using(rID)
where name = director;
```

Return all reviewer names and movie names together in a single list, alphabetized. (Sorting by the first name of the reviewer and first word in the title is fine; no need for special processing on last names or removing "The".)

```
select name from
    (select distinct name
    from Reviewer
    union
    select distinct title
    from Movie)
order by name;
```

Find the titles of all movies not reviewed by Chris Jackson.

```
select title from Movie
where mID not in
(select mID from Rating where rID in
      (select rID from Reviewer
       where name = "Chris Jackson"))
```

For all pairs of reviewers such that both reviewers gave a rating to the same movie, return the names of both reviewers. Eliminate duplicates, don't pair reviewers with themselves, and include each pair only once. For each pair, return the names in the pair in alphabetical order.

```
select distinct Re1.name, Re2.name
from Rating R1, Rating R2, Reviewer Re1, Reviewer Re2
where R1.mID = R2.mID
and R1.rID = Re1.rID
and R2.rID = Re2.rID
and Re1.name < Re2.name
order by Re1.name, Re2.name;
```

For each rating that is the lowest (fewest stars) currently in the database, return the reviewer name, movie title, and number of stars.

```
select name, title, stars
from Rating
join Movie using(mID)
join Reviewer using(rID)
where stars <= (select min(stars) from Rating);
```

List movie titles and average ratings, from highest-rated to lowest-rated. If two or more movies have the same average rating, list them in alphabetical order.

```
select title, avg(stars) as avg
from Rating
join  Movie using(mID)
join Reviewer using(rID)
group by title
order by avg desc, title;
```

Find the names of all reviewers who have contributed three or more ratings. (As an extra challenge, try writing the query without HAVING or without COUNT.)

```
select name
from Rating
join  Movie using(mID)
join Reviewer using(rID)
group by name
having count(mID) >= 3;
```

Some directors directed more than one movie. For all such directors, return the titles of all movies directed by them, along with the director name. Sort by director name, then movie title. (As an extra challenge, try writing the query both with and without COUNT.)

```
select title, director from Movie
where director in
  ( select director
    from movie
    group by director
    having count(*) > 1)
order by director, title;
```

Find the movie(s) with the highest average rating. Return the movie title(s) and average rating. (Hint: This query is more difficult to write in SQLite than other systems; you might think of it as finding the highest average rating and then choosing the movie(s) with that average rating.)

```
select title, avg from
       (select title,
       avg(stars) as avg
       from Rating
       join  Movie using(mID)
       join Reviewer using(rID)
       group by title)
where avg >=
       (select max(avg) from
        (select title,
        avg(stars) as avg
        from Rating
         join  Movie using(mID)
         join Reviewer using(rID)
         group by title));
```

Find the movie(s) with the lowest average rating. Return the movie title(s) and average rating. (Hint: This query may be more difficult to write in SQLite than other systems; you might think of it as finding the lowest average rating and then choosing the movie(s) with that average rating.)

```
select title, avg
       from
       (select title, avg(stars) as avg
       from Rating
```

```
        join  Movie  using(mID)
        join Reviewer  using(rID)
        group  by  title)
where avg  <=
        (select min(avg)  from
        (select title,
        avg(stars)  as avg
        from  Rating
        join  Movie  using(mID)
        join Reviewer  using(rID)
        group  by  title));
```

For each director, return the director's name together with the title(s) of the movie(s) they directed that received the highest rating among all of their movies, and the value of that rating. Ignore movies whose director is NULL.

```
select director,  title,  max(stars)  from Rating
join Movie  using(mID)
join Reviewer  using(rID)
where director  is  not  null
group  by  director;
```

Add the reviewer Roger Ebert to your database, with an rID of 209.

```
insert  into Reviewer  (rID,  name)
values  ("209",  "Roger  Ebert");
```

Insert 5-star ratings by James Cameron for all movies in the database. Leave the review date as NULL.

```
insert into Rating  (rID,  mID,  stars,  ratingDate )
select Reviewer.rID ,  Movie.mID,  5,  null from Movieleft
outer join Reviewer
where  Reviewer.name="James  Cameron";
```

For all movies that have an average rating of 4 stars or higher, add 25 to the release year. (Update the existing tuples; don't insert new tuples.)

```
update Movie
set year  =  year  +  25
where mID  in
(select mID  from
     (select mID,  avg(stars)  as avg
        from Movie  join Rating  using(mID)
        group by mID)
where avg  >=  4);
```

Remove all ratings where the movie's year is before 1970 or after 2000, and the rating is fewer than 4 stars.

```
delete from  rating
where mID  in  (select mID  from  movie  where  year  <  1970  or  year  >  2000)
and stars  <  4;
```