SQL Movie-Rating Query Exercises Extras

Bookmark this page

You've started a new movie-rating website, and you've been collecting data on reviewers' ratings of various movies. There's not much data yet, but you can still try out some interesting queries. Here's the schema:

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*.

Your queries will run over a small data set conforming to the schema. View the database. (You can also download the schema and data.)

Instructions: Each problem asks you to write a query in SQL. To run your query against our back-end sample database using SQLite, click the "Submit" button. You will see a display of your query result and the expected result. If the results match, your query will be marked "correct". You may run as many queries as you like for each question.

Important Notes:

- Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.
- Unless a specific result ordering is asked for, you can return the result rows in any order.
- *You are to translate the English into a SQL query that computes the desired result over all possible databases.* All we actually check is that your query gets the right answer on the small sample database. Thus, even if your solution is marked as correct, it is possible that your query does not correctly reflect the problem at hand. (For example, if we ask for a complex condition that requires accessing all of the tables, but over our small data set in the end the condition is satisfied only by Star Wars, then the query "select title from Movie where title = 'Star Wars'" will be marked correct even though it doesn't reflect the actual question.) Circumventing the system in this fashion will get you a high score on the exercises, but it won't help you learn SQL. On the other hand, an incorrect attempt at a general solution is unlikely to produce the right answer, so you shouldn't be led astray by our checking system.

You may perform these exercises as many times as you like, so we strongly encourage you to keep working with them until you complete the exercises with full credit.

Q1

0 points (ungraded)
Find the names of all reviewers who rated Gone with the Wind.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

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

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Q2

0 points (ungraded)
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.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
select Reviewer.name, Movie.title, Rating.stars
from Reviewer
    join Rating on Rating.rID = Reviewer.rID
    join Movie on Movie.mID = Rating.mID
where Reviewer.name = Movie.director;
```

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Q3

0 points (ungraded)
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".)

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
select name from Reviewer
union
select title from Movie
order by Reviewer.name, Movie.title;
```

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Q4

0 points (ungraded)
Find the titles of all movies not reviewed by Chris Jackson.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

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

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Q5

0 points (ungraded)
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.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
select distinct rv1.name, rv2.name
from Reviewer rv1, Reviewer rv2, Rating rt1, Rating rt2
where rv1.rID = rt1.rID
    and rv2.rID = rt2.rID
    and rt1.mID = rt2.mID
    and rv1.name < rv2.name
order by rv1.name
```

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Q6

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

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
select rv.name, m.title, rt.stars
from Reviewer rv
    join Rating rt on rv.rID = rt.rID
    join Movie m on m.mID = rt.mID
where rt.stars = (select min(stars) from Rating);
```

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Q7

0 points (ungraded)
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.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
select m.title, avg(rt.stars)
from Movie m join Rating rt using(mID)
group by rt.mID
order by avg(rt.stars) desc, m.title;
```

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Q8

0 points (ungraded)
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.)

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
select distinct rv.name
from Reviewer rv join Rating rt using(rID)
where (select count(*)
        from Rating rt2
        where rt2.rID = rt.rID) >= 3;
```

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Q9

0 points (ungraded)
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.)

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

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

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Q10

0 points (ungraded)
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.)

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
select m.title, rt1.avgstars
from Movie m
    join (select mID, avg(stars) as avgstars
            from Rating group by mID) rt1 using(mID)
where rt1.avgstars = (select max(avgstars)
                        from (select mID, avg(stars) as avgstars
                                from Rating group by mID));
```

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Q11

0 points (ungraded)
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.)

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
select m.title, rt1.avgstars
from Movie m
    join (select mID, avg(stars) as avgstars
            from Rating group by mID) rt1 using(mID)
where rt1.avgstars = (select min(avgstars)
                        from (select mID, avg(stars) as avgstars
                                from Rating group by mID));
```

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Q12

0 points (ungraded)
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.

Note: Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
select m.director, m.title, max(rt.stars)
from Movie m join Rating rt using(mID)
where m.director is not null
group by m.director
```

Press ESC then TAB or click outside of the code editor to exit
Submit
Correct

Previous
Next Up: SQL Social-Network Query Exercises
2 min + 1 activity