backpackerhh / core-set.sql

Last active 2 days ago • Report abuse

Star

Code Revisions 3 Stars 60 Forks 31

SQL - Movie-Rating Query Exercises

core-set.sql

```sql
-- 1. Find the titles of all movies directed by Steven Spielberg.

SELECT title
FROM Movie
WHERE director = 'Steven Spielberg';


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

SELECT DISTINCT year
FROM Movie, Rating
WHERE Movie.mId = Rating.mId AND stars IN (4, 5)
ORDER BY year;

SELECT DISTINCT year
FROM Movie
INNER JOIN Rating ON Movie.mId = Rating.mId
WHERE stars IN (4, 5)
ORDER BY year;

SELECT DISTINCT year
FROM Movie
INNER JOIN Rating USING(mId)
WHERE stars IN (4, 5)
ORDER BY year;

SELECT DISTINCT year
FROM Movie NATURAL JOIN Rating
WHERE stars IN (4, 5)
ORDER BY year;


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

SELECT title
FROM Movie
WHERE mId NOT IN (SELECT mID FROM Rating);

```

```sql

-- 4. Some reviewers didn't provide a date with their rating. Find the names of all reviewers w

SELECT name
FROM Reviewer
INNER JOIN Rating USING(rId)
WHERE ratingDate IS NULL;


-- 5. Write a query to return the ratings data in a more readable format: reviewer name, movie

SELECT name, title, stars, ratingDate
FROM Movie, Rating, Reviewer
WHERE Movie.mId = Rating.mId AND Reviewer.rId = Rating.rId
ORDER BY name, title, stars;

SELECT name, title, stars, ratingDate
FROM Movie
INNER JOIN Rating ON Movie.mId = Rating.mId
INNER JOIN Reviewer ON Reviewer.rId = Rating.rId
ORDER BY name, title, stars;

SELECT name, title, stars, ratingDate
FROM Movie
INNER JOIN Rating USING(mId)
INNER JOIN Reviewer USING(rId)
ORDER BY name, title, stars;

SELECT name, title, stars, ratingDate
FROM Movie NATURAL JOIN Rating NATURAL JOIN Reviewer
ORDER BY name, title, stars;


-- 6. For all cases where the same reviewer rated the same movie twice and gave it a higher rat

SELECT name, title
FROM Movie
INNER JOIN Rating R1 USING(mId)
INNER JOIN Rating R2 USING(rId)
INNER JOIN Reviewer USING(rId)
WHERE R1.mId = R2.mId AND R1.ratingDate < R2.ratingDate AND R1.stars < R2.stars;

SELECT name, title
FROM Movie
INNER JOIN Rating R1 USING(mId)
INNER JOIN Rating R2 USING(rId, mId)
INNER JOIN Reviewer USING(rId)
WHERE R1.ratingDate < R2.ratingDate AND R1.stars < R2.stars;


-- 7. For each movie that has at least one rating, find the highest number of stars that movie

```

```sql
SELECT title, MAX(stars)
FROM Movie
INNER JOIN Rating USING(mId)
GROUP BY mId
ORDER BY title;


-- 8. For each movie, return the title and the 'rating spread', that is, the difference betweer

SELECT title, (MAX(stars) - MIN(stars)) AS rating_spread
FROM Movie
INNER JOIN Rating USING(mId)
GROUP BY mId
ORDER BY rating_spread DESC, title;


-- 9. Find the difference between the average rating of movies released before 1980 and the ave

SELECT AVG(Before1980.avg) - AVG(After1980.avg)
FROM (
  SELECT AVG(stars) AS avg
  FROM Movie
  INNER JOIN Rating USING(mId)
  WHERE year < 1980
  GROUP BY mId
) AS Before1980, (
  SELECT AVG(stars) AS avg
  FROM Movie
  INNER JOIN Rating USING(mId)
  WHERE year > 1980
  GROUP BY mId
) AS After1980;
```

extras.sql

```sql
-- 1. Find the names of all reviewers who rated Gone with the Wind.

SELECT DISTINCT name
FROM Movie
INNER JOIN Rating USING(mId)
INNER JOIN Reviewer USING(rId)
WHERE title = "Gone with the Wind";


-- 2. For any rating where the reviewer is the same as the director of the movie, return the re

SELECT name, title, stars
FROM Movie
INNER JOIN Rating USING(mId)
INNER JOIN Reviewer USING(rId)
WHERE director = name;
```

```sql


-- 3. Return all reviewer names and movie names together in a single list, alphabetized. (Sorti

SELECT title FROM Movie
UNION
SELECT name FROM Reviewer
ORDER BY name, title;


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

SELECT title
FROM Movie
WHERE mId NOT IN (
  SELECT mId
  FROM Rating
  INNER JOIN Reviewer USING(rId)
  WHERE name = "Chris Jackson"
);


-- 5. For all pairs of reviewers such that both reviewers gave a rating to the same movie, retu

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;


-- 6. For each rating that is the lowest (fewest stars) currently in the database, return the r

SELECT name, title, stars
FROM Movie
INNER JOIN Rating USING(mId)
INNER JOIN Reviewer USING(rId)
WHERE stars = (SELECT MIN(stars) FROM Rating);


-- 7. List movie titles and average ratings, from highest-rated to lowest-rated. If two or more

SELECT title, AVG(stars) AS average
FROM Movie
INNER JOIN Rating USING(mId)
GROUP BY mId
ORDER BY average DESC, title;


-- 8. Find the names of all reviewers who have contributed three or more ratings.
```

```sql

SELECT name
FROM Reviewer
WHERE (SELECT COUNT(*) FROM Rating WHERE Rating.rId = Reviewer.rId) >= 3;

SELECT name
FROM Reviewer
INNER JOIN Rating USING(rId)
GROUP BY rId
HAVING COUNT(*) >= 3;

-- At least 3 ratings to different movies (Remainder to myself)

SELECT name
FROM Reviewer
WHERE (SELECT COUNT(DISTINCT mId) FROM Rating WHERE Rating.rId = Reviewer.rId) >= 3;


-- 9. Some directors directed more than one movie. For all such directors, return the titles o

SELECT title, director
FROM Movie M1
WHERE (SELECT COUNT(*) FROM Movie M2 WHERE M1.director = M2.director) > 1
ORDER BY director, title;

SELECT M1.title, director
FROM Movie M1
INNER JOIN Movie M2 USING(director)
GROUP BY M1.mId
HAVING COUNT(*) > 1
ORDER BY director, M1.title;


-- 10. Find the movie(s) with the highest average rating. Return the movie title(s) and average

SELECT title, AVG(stars) AS average
FROM Movie
INNER JOIN Rating USING(mId)
GROUP BY mId
HAVING average = (
  SELECT MAX(average_stars)
  FROM (
    SELECT title, AVG(stars) AS average_stars
    FROM Movie
    INNER JOIN Rating USING(mId)
    GROUP BY mId
  )
);


-- 11. Find the movie(s) with the lowest average rating. Return the movie title(s) and average

```

```sql
SELECT title, AVG(stars) AS average
FROM Movie
INNER JOIN Rating USING(mId)
GROUP BY mId
HAVING average = (
  SELECT MIN(average_stars)
  FROM (
    SELECT title, AVG(stars) AS average_stars
    FROM Movie
    INNER JOIN Rating USING(mId)
    GROUP BY mId
  )
);


-- 12. For each director, return the director's name together with the title(s) of the movie(s

SELECT director, title, MAX(stars)
FROM Movie
INNER JOIN Rating USING(mId)
WHERE director IS NOT NULL
GROUP BY director;
```

modification.sql

```sql
-- 1. Add the reviewer Roger Ebert to your database, with an rID of 209.

INSERT INTO Reviewer
VALUES (209, "Roger Ebert");


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

INSERT INTO Rating
SELECT (SELECT rId FROM Reviewer WHERE name = "James Cameron"), mId, 5, NULL
FROM Movie;


-- 3. For all movies that have an average rating of 4 stars or higher, add 25 to the release y

UPDATE Movie
SET year = year + 25
WHERE mId IN (
  SELECT mId
  FROM Movie
  INNER JOIN Rating USING(mId)
  GROUP BY mId
  HAVING AVG(stars) >= 4
);


```

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

DELETE FROM Rating
WHERE mId IN (
  SELECT mId
  FROM Movie
  WHERE year < 1970 OR year > 2000
) AND stars < 4;
```

bmwilllee commented on Apr 2, 2017

Thanks, very helpful!

EmiliaDariel commented on Jul 12, 2018

can anyone please explain this:
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;

safwans commented on Jun 6, 2019 • edited

Is there a potential issue in #9 because you may have repeating rows due to the join? I wrote the query below and got slightly different average

select
(select avg(ratings)
from(
select avg(r.stars) ratings
from rating r, movie m
where r.mid = m.mid
and m.year < '1980'
group by r.mid
)) - (select avg(ratings)
from(
select avg(r.stars) ratings
from rating r, movie m
where r.mid = m.mid
and m.year >= '1980'
group by r.mid
))rat

femiaiyeku commented on Jul 27, 2019

very helpful to prepare for sql interview

AshwinAJa commented on Sep 21, 2019

how to download data

macso95 commented on Feb 23, 2020 • edited

How would I do these ones?

1. For each movie, display the number of times it was reviewed and the average of the number of stars it received. List only the movies that were reviewed three or more times.

2. Use a correlated reference to find all reviews that have occurred on the same day by different reviewers. Display the reviewer ID and date of the review. Print out the. Order by rating date. You must use the word EXISTS within query.

bartubozkurt commented on May 2, 2020

How can I do ?
- How many movies have been made each year?
- How many actors are there in each movie?
thank you for the Exercises

bikashghadai3 commented on Jun 3, 2020

how find the rating of 1 and 2 stars for the last 5 days in a week in a table.

wahabmemo commented on Apr 21, 2021

You have to display an actor name who has worked in many films. [Use join, group by, order by]

ghost commented on Jun 9, 2021

> How can I do ?
>
> - How many movies have been made each year?

> - How many actors are there in each movie?
>   thank you for the Exercises

Windsleeper commented on Jun 25, 2021

Thank you, this is very helpful!

arsalh commented on Aug 27, 2021

For the average rating of movies before and after 1980 question (movies #9), can someone help me what I am doing wrong in my query below? Instead of getting result of 0.0555555555555558, mine comes to 0.05555555555555536. Small difference but would like to understand what I am doing wrong. Thank you so much!

SELECT distinct
(SELECT avg(rt_avg)
FROM (SELECT m.mID, avg(rt.stars) as rt_avg, year
FROM Rating rt JOIN Movie m ON rt.mID=m.mID
GROUP BY rt.mID) temp
WHERE year<1980)

(SELECT avg(rt_avg)
FROM (SELECT m.mID, avg(rt.stars) as rt_avg, year
FROM Rating rt JOIN Movie m ON rt.mID=m.mID
GROUP BY rt.mID) temp
WHERE year>1980)

FROM Movie

AlMokgalaka commented 17 days ago

EmiliaDariel, Have a look at this site from Stanford might help you
http://linyishui.top/2019090601.html
Otherwise the SELECT DISTINCT statement is used to return only distinct/different values (avoiding duplicate values present in any specific columns of a table.). An example, inside a table, a column often contains many duplicate values, and sometimes we only want to return or list the different values. The second line FROM clause, third line WHERE clause, 4th-line AND clauses (the two tables having common columns, matching id, mid and ratings id, rid) follows ANSI (American National Standards Institute) table aliases and ANSI old/theta style to reduce those chains of names. Remember ratings, reviewers tables and id are spelt in small letters and when you submit that query, the query handler might not be able to accept and or recognize those big/capital letters as the configuration settings in SQL might have been disenabled/abled although by SQL is by default case insensitive (Query handler checks spelling (=goes to view RATINGS or ratings, like when you are hungry you would ask a lunch not 2 lunches) and recognize only available views or table views and then raises a red flag, saying I don't have such table, RATING or rating in here, that means it only saw ratings/RATINGS/RaTIngs). see an example: SELECT

Orders.OrderID, Orders.CustomerID, Orders.EmployeeID, Orders.OrderDate, Orders.RequiredDate, Orders.ShippedDate, Orders.ShipVia, Orders.Freight, Orders.ShipName, Orders.ShipAddress, Orders.ShipCity, Orders.ShipRegion, Orders.ShipPostalCode, Orders.ShipCountry,

Customers.CompanyName, Customers.Address, Customers.City, Customers.Region, Customers.PostalCode, Customers.Country

FROM Customers

INNER JOIN Orders

```
ON Customers.CustomerID = Orders.CustomerID;
```

Note: The table names need not be repeated unless the same column names exist in both tables. The table names are only required in the FROM, JOIN, and ON clauses, and in the latter, only because the relating column, CustomerID, has the same name in both tables.

The query syntax shown above follows ANSI (American National Standards Institute) rules and should work in the latest versions of all relational databases. Older syntax includes the join condition in the WHERE clause (theta style). Note the number of rows and columns in the result set for the Orders Query and try the same example (with fewer columns), using the older style and table aliases, as follows:

SELECT o.OrderID, o.EmployeeID, o.OrderDate, o.RequiredDate, o.ShippedDate, o.ShipVia, o.Freight, c.CompanyName, c.Address, c.City, c.Region, c.PostalCode, c.Country

FROM Customers c, Orders o
WHERE c.CustomerID = o.CustomerID;

Note for MS Access users: Compare this query in design view with the ANSI style query. MS Access runs the query correctly but cannot represent it in the usual way In the graphical query interface.