E03 - SQL Review (Part 2)

E03, Business Intelligence

Oemer Furkan Coban

21.11.2025 20:22

Business Intelligence
Winter Term 2025/2026

University of Oldenburg

Accepted: 9 (100%)
Total: 9

1.1 *Find the titles of all movies directed by Steven Spielberg.*

accepted

```sql
SELECT title FROM movie WHERE director = 'Steven Spielberg';
```

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

accepted

```sql
SELECT DISTINCT movie.release_year
FROM movie JOIN rating ON movie.m_id = rating.m_id
WHERE rating.stars = 4 OR rating.stars = 5
ORDER BY movie.release_year ASC;
```

1.3 *Find the titles of all movies that have no ratings.*

accepted

```sql
SELECT movie.title
FROM movie
LEFT JOIN rating ON movie.m_id = rating.m_id
WHERE rating.stars IS NULL;
```

1.4 *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.*

accepted

```sql
SELECT reviewer.reviewer_name FROM reviewer LEFT JOIN rating ON reviewer.r_id = rating.r_id
    WHERE rating.rating_date IS NULL;
```

1.5 *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.*

accepted

```sql
SELECT reviewer.reviewer_name, movie.title, rating.stars, rating.rating_date
FROM reviewer
JOIN rating ON reviewer.r_id = rating.r_id
JOIN movie ON movie.m_id = rating.m_id
ORDER BY reviewer.reviewer_name;
```

1.6 *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.*

accepted

```sql
SELECT reviewer.reviewer_name, movie.title
FROM rating AS r1
JOIN rating AS r2 ON r1.r_id = r2.r_id AND r1.m_id = r2.m_id AND r1.rating_date <
    r2.rating_date AND r1.stars < r2.stars
```

```sql
JOIN reviewer ON r1.r_id = reviewer.r_id
JOIN movie ON r1.m_id = movie.m_id;
```

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

accepted

```sql
SELECT movie.title, MAX(rating.stars) AS max
FROM movie
JOIN rating ON movie.m_id = rating.m_id
GROUP BY movie.title
ORDER BY movie.title;
```

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

accepted

```sql
SELECT movie.title, (MAX(rating.stars) - MIN(rating.stars)) AS result_
FROM movie
JOIN rating ON movie.m_id = rating.m_id
GROUP BY movie.title
ORDER BY result_ DESC, movie.title ASC;
```

1.9 *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).*

accepted

```sql
SELECT
(SELECT AVG(avg_rating)
FROM (
SELECT AVG(rating.stars) AS avg_rating
FROM movie
JOIN rating ON movie.m_id = rating.m_id
WHERE movie.release_year < 1980
GROUP BY movie.m_id)
AS before1980)
-
(
SELECT AVG(avg_rating)
FROM (
SELECT AVG(rating.stars) AS avg_rating
FROM movie
JOIN rating ON movie.m_id = rating.m_id
WHERE movie.release_year > 1980
GROUP BY movie.m_id
) AS after1980)
AS result_;
```