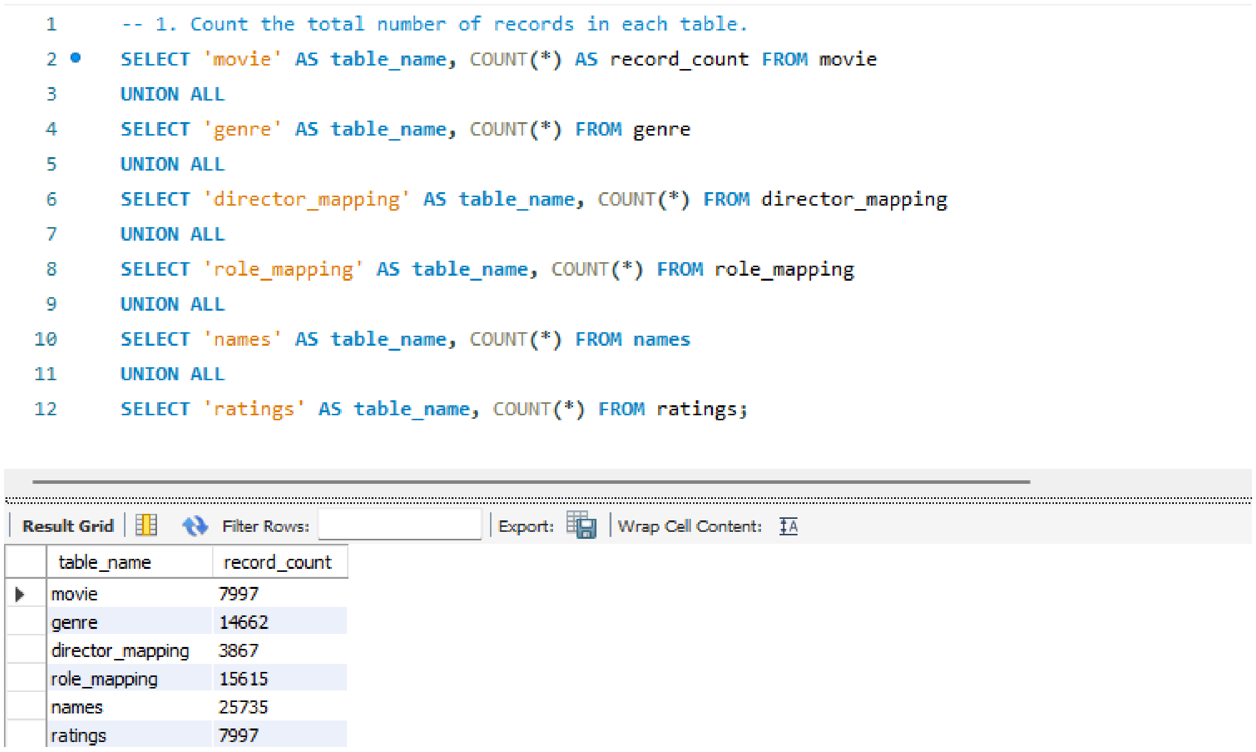
IMDB Dataset: SQL Queries and Solutions

This document provides 25 SQL queries to analyze the IMDB dataset, addressing the questions below.

1. Count the records in each table.

```
SELECT 'movie' AS table_name, COUNT(*) AS record_count FROM movie
UNION ALL
SELECT 'genre', COUNT(*) FROM genre
UNION ALL
SELECT 'director_mapping', COUNT(*) FROM director_mapping
UNION ALL
SELECT 'role_mapping', COUNT(*) FROM role_mapping
UNION ALL
SELECT 'names', COUNT(*) FROM names
UNION ALL
SELECT 'ratings', COUNT(*) FROM ratings;
```

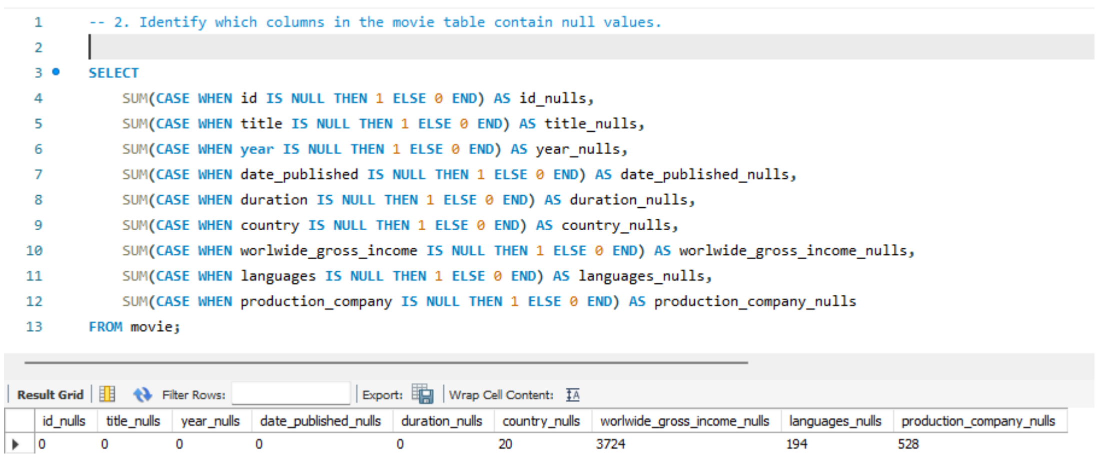


```
-- 1. Count the total number of records in each table.
SELECT 'movie' AS table_name, COUNT(*) AS record_count FROM movie
UNION ALL
SELECT 'genre' AS table_name, COUNT(*) FROM genre
UNION ALL
SELECT 'director_mapping' AS table_name, COUNT(*) FROM director_mapping
UNION ALL
SELECT 'role_mapping' AS table_name, COUNT(*) FROM role_mapping
UNION ALL
SELECT 'names' AS table_name, COUNT(*) FROM names
UNION ALL
SELECT 'ratings' AS table_name, COUNT(*) FROM ratings;
```

table_name	record_count
movie	7997
genre	14662
director_mapping	3867
role_mapping	15615
names	25735
ratings	7997

2. Find columns with null values in the `movie` table.

```
SELECT
   SUM(CASE WHEN id IS NULL THEN 1 ELSE 0 END) AS id_nulls,
   SUM(CASE WHEN title IS NULL THEN 1 ELSE 0 END) AS title_nulls,
   SUM(CASE WHEN year IS NULL THEN 1 ELSE 0 END) AS year_nulls,
   SUM(CASE WHEN date_published IS NULL THEN 1 ELSE 0 END) AS date_published_nulls,
   SUM(CASE WHEN duration IS NULL THEN 1 ELSE 0 END) AS duration_nulls,
   SUM(CASE WHEN country IS NULL THEN 1 ELSE 0 END) AS country_nulls,
   SUM(CASE WHEN worlwide_gross_income IS NULL THEN 1 ELSE 0 END) AS worlwide_gross_income_nulls,
   SUM(CASE WHEN languages IS NULL THEN 1 ELSE 0 END) AS languages_nulls,
   SUM(CASE WHEN production_company IS NULL THEN 1 ELSE 0 END) AS production_company_nulls
FROM movie;
```

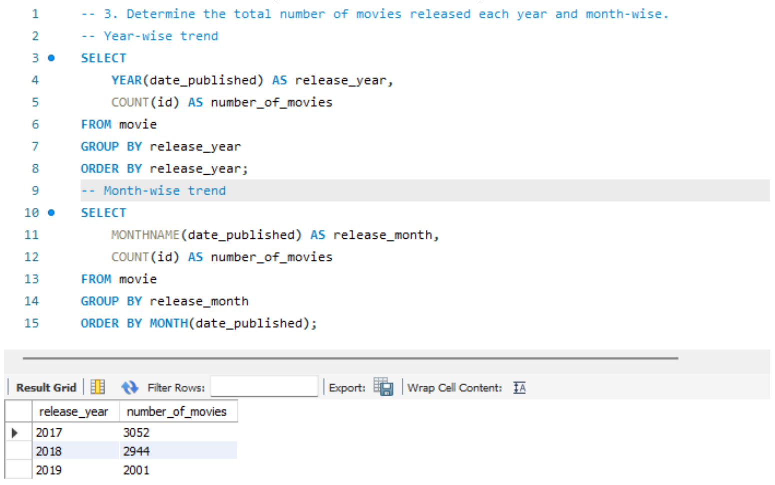


```
-- 2. Identify which columns in the movie table contain null values.

SELECT
    SUM(CASE WHEN id IS NULL THEN 1 ELSE 0 END) AS id_nulls,
    SUM(CASE WHEN title IS NULL THEN 1 ELSE 0 END) AS title_nulls,
    SUM(CASE WHEN year IS NULL THEN 1 ELSE 0 END) AS year_nulls,
    SUM(CASE WHEN date_published IS NULL THEN 1 ELSE 0 END) AS date_published_nulls,
    SUM(CASE WHEN duration IS NULL THEN 1 ELSE 0 END) AS duration_nulls,
    SUM(CASE WHEN country IS NULL THEN 1 ELSE 0 END) AS country_nulls,
    SUM(CASE WHEN worlwide_gross_income IS NULL THEN 1 ELSE 0 END) AS worlwide_gross_income_nulls,
    SUM(CASE WHEN languages IS NULL THEN 1 ELSE 0 END) AS languages_nulls,
    SUM(CASE WHEN production_company IS NULL THEN 1 ELSE 0 END) AS production_company_nulls
FROM movie;
```

```
id_nulls  title_nulls  year_nulls  date_published_nulls  duration_nulls  country_nulls  worlwide_gross_income_nulls  languages_nulls  production_company_nulls
0         0            0           0                     0               20             3724                         194              528
```

3. Analyze movie release trends by year and month.

```
-- Year-wise trend
SELECT
    YEAR(date_published) AS release_year,
    COUNT(id) AS number_of_movies
FROM movie
GROUP BY release_year
ORDER BY release_year;

-- Month-wise trend
SELECT
    MONTHNAME(date_published) AS release_month,
    COUNT(id) AS number_of_movies
FROM movie
GROUP BY release_month
ORDER BY MONTH(date_published);
```

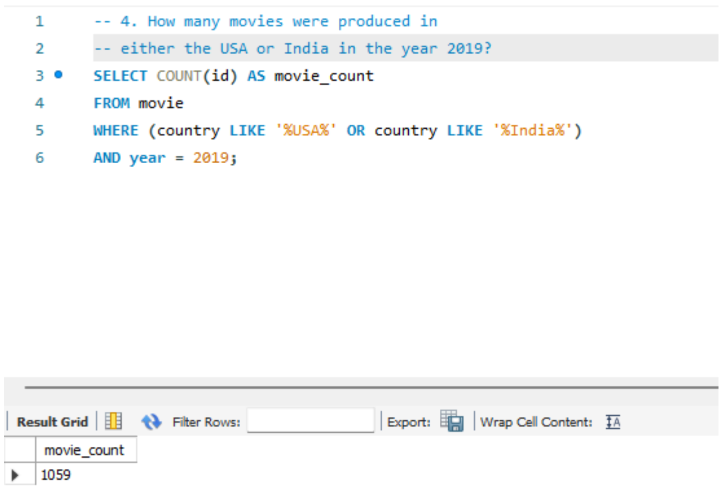


```
-- 3. Determine the total number of movies released each year and month-wise.
-- Year-wise trend
SELECT
    YEAR(date_published) AS release_year,
    COUNT(id) AS number_of_movies
FROM movie
GROUP BY release_year
ORDER BY release_year;
-- Month-wise trend
SELECT
    MONTHNAME(date_published) AS release_month,
    COUNT(id) AS number_of_movies
FROM movie
GROUP BY release_month
ORDER BY MONTH(date_published);
```

```
release_year  number_of_movies
2017          3052
2018          2944
2019          2001
```

4. Count movies from the USA or India in 2019.

```
SELECT COUNT(id) AS movie_count
FROM movie
WHERE (country LIKE '%USA%' OR country LIKE '%India%')
AND year = 2019;
```

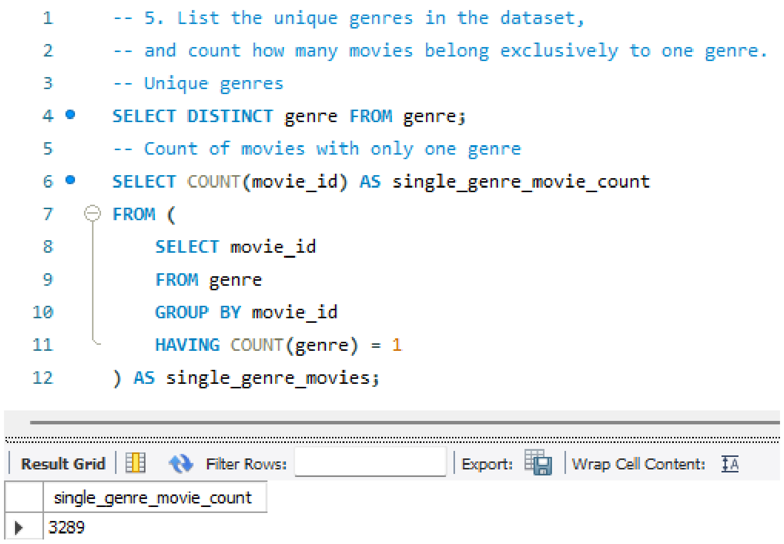


```
-- 4. How many movies were produced in
-- either the USA or India in the year 2019?
SELECT COUNT(id) AS movie_count
FROM movie
WHERE (country LIKE '%USA%' OR country LIKE '%India%')
AND year = 2019;
```

```
movie_count
1059
```

5. List unique genres and count single-genre movies.

```
-- Unique genres
SELECT DISTINCT genre FROM genre;

-- Count of movies with only one genre
SELECT COUNT(movie_id) AS single_genre_movie_count
FROM (
    SELECT movie_id
    FROM genre
    GROUP BY movie_id
    HAVING COUNT(genre) = 1
) AS single_genre_movies;
```

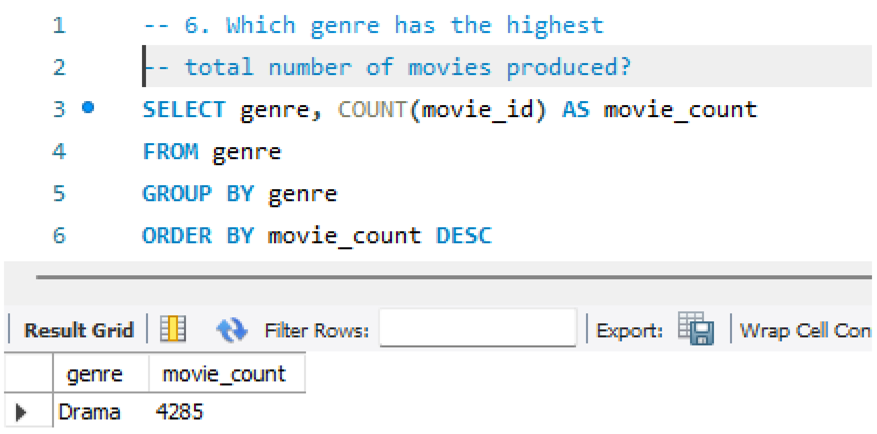


```
-- 5. List the unique genres in the dataset,
-- and count how many movies belong exclusively to one genre.
-- Unique genres
SELECT DISTINCT genre FROM genre;
-- Count of movies with only one genre
SELECT COUNT(movie_id) AS single_genre_movie_count
FROM (
    SELECT movie_id
    FROM genre
    GROUP BY movie_id
    HAVING COUNT(genre) = 1
) AS single_genre_movies;
```

```
single_genre_movie_count
3289
```

6. Find the genre with the most movies.

```
SELECT genre, COUNT(movie_id) AS movie_count
FROM genre
GROUP BY genre
ORDER BY movie_count DESC
LIMIT 1;
```

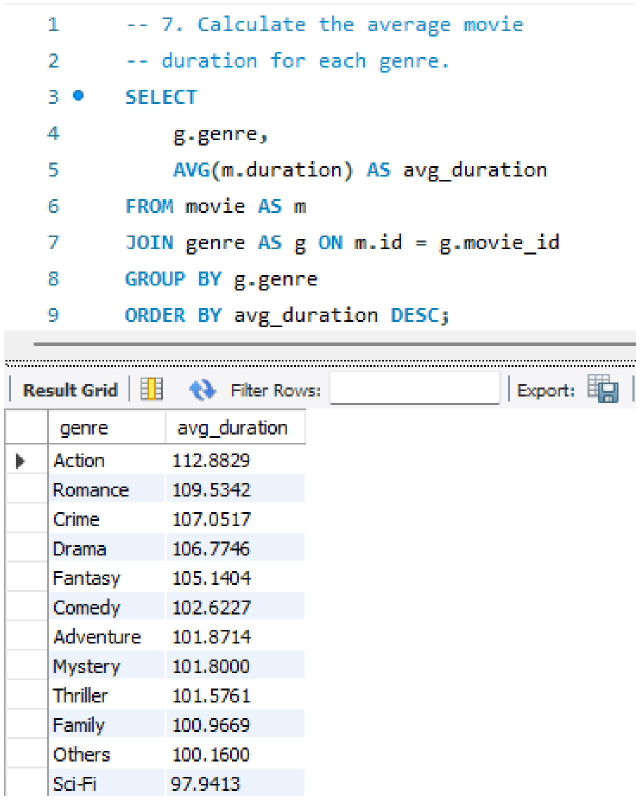


```
-- 6. Which genre has the highest
-- total number of movies produced?
SELECT genre, COUNT(movie_id) AS movie_count
FROM genre
GROUP BY genre
ORDER BY movie_count DESC
```

genre	movie_count
Drama	4285

7. Calculate the average duration for each genre.

```
SELECT
   g.genre,
   AVG(m.duration) AS avg_duration
FROM movie AS m
JOIN genre AS g ON m.id = g.movie_id
GROUP BY g.genre
ORDER BY avg_duration DESC;
```



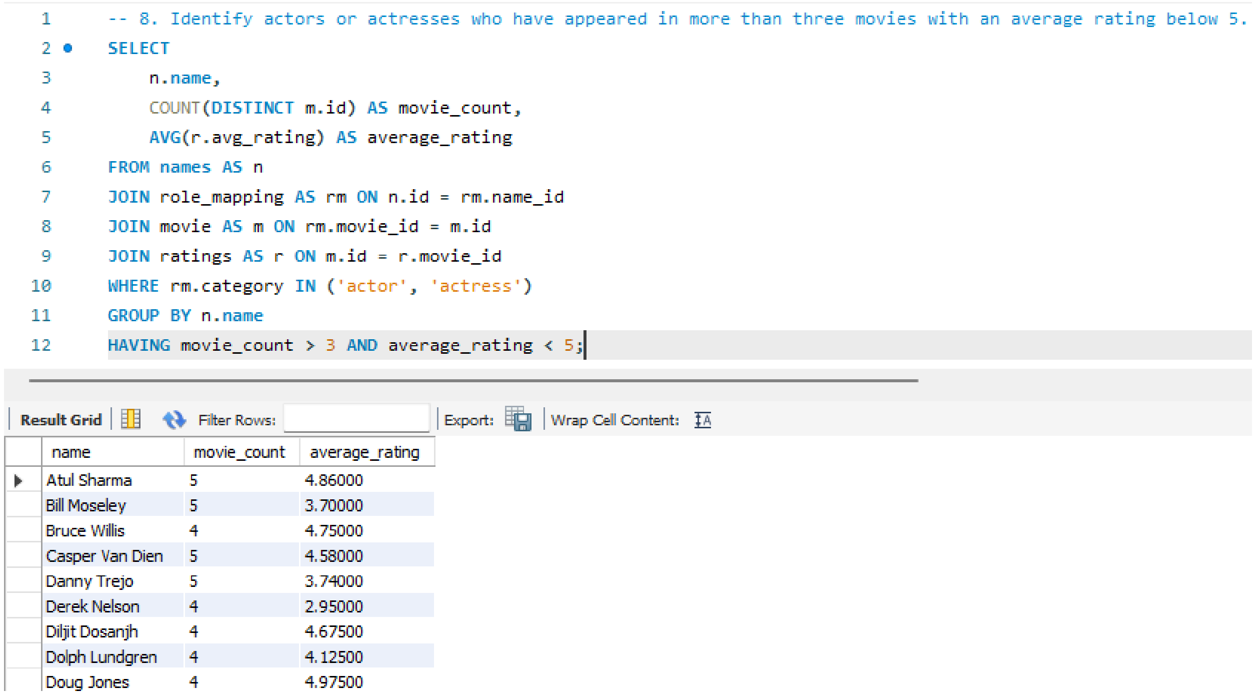
```
-- 7. Calculate the average movie
-- duration for each genre.
SELECT
    g.genre,
    AVG(m.duration) AS avg_duration
FROM movie AS m
JOIN genre AS g ON m.id = g.movie_id
GROUP BY g.genre
ORDER BY avg_duration DESC;
```

genre	avg_duration
Action	112.8829
Romance	109.5342
Crime	107.0517
Drama	106.7746
Fantasy	105.1404
Comedy	102.6227
Adventure	101.8714
Mystery	101.8000
Thriller	101.5761
Family	100.9669
Others	100.1600
Sci-Fi	97.9413

8. Find actors in >3 movies with an average rating < 5.

```
SELECT
   n.name,
   COUNT(DISTINCT m.id) AS movie_count,
   AVG(r.avg_rating) AS average_rating
FROM names AS n
JOIN role_mapping AS rm ON n.id = rm.name_id
JOIN movie AS m ON rm.movie_id = m.id
JOIN ratings AS r ON m.id = r.movie_id
WHERE rm.category IN ('actor', 'actress')
```

GROUP BY n.name
HAVING movie_count > 3 AND average_rating < 5;

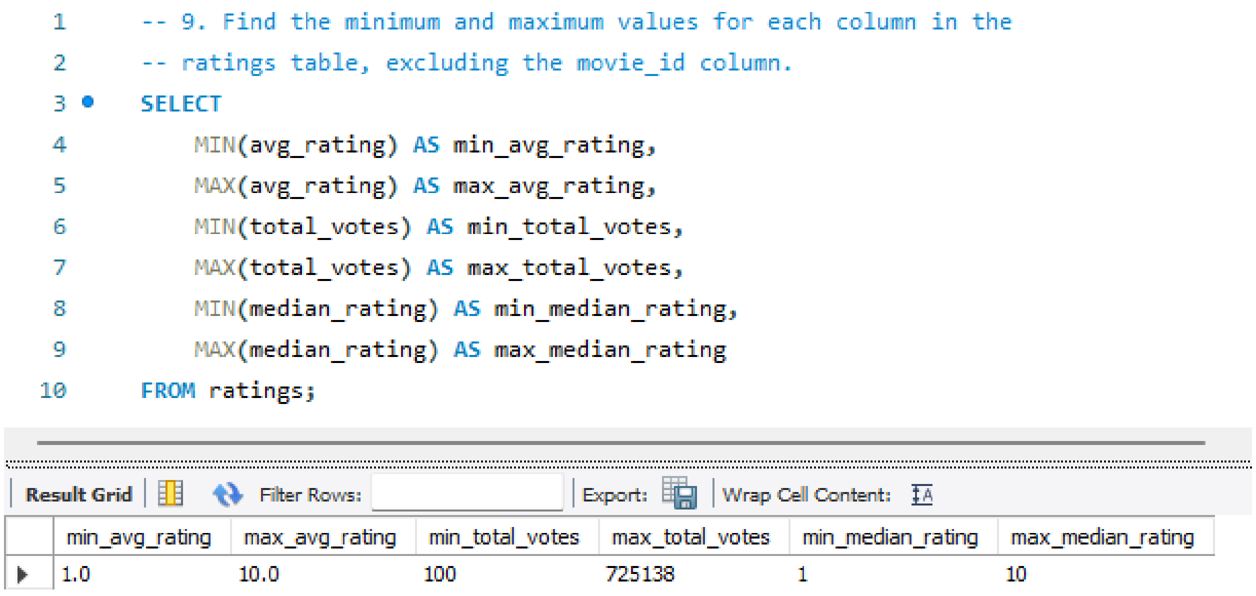


```
-- 8. Identify actors or actresses who have appeared in more than three movies with an average rating below 5.
SELECT
    n.name,
    COUNT(DISTINCT m.id) AS movie_count,
    AVG(r.avg_rating) AS average_rating
FROM names AS n
JOIN role_mapping AS rm ON n.id = rm.name_id
JOIN movie AS m ON rm.movie_id = m.id
JOIN ratings AS r ON m.id = r.movie_id
WHERE rm.category IN ('actor', 'actress')
GROUP BY n.name
HAVING movie_count > 3 AND average_rating < 5;
```

name	movie_count	average_rating
Atul Sharma	5	4.86000
Bill Moseley	5	3.70000
Bruce Willis	4	4.75000
Casper Van Dien	5	4.58000
Danny Trejo	5	3.74000
Derek Nelson	4	2.95000
Diljit Dosanjh	4	4.67500
Dolph Lundgren	4	4.12500
Doug Jones	4	4.97500

9. Find min/max values for each column in the `ratings` table (excluding `movie_id`).

SELECT
    MIN(avg_rating) AS min_avg_rating,
    MAX(avg_rating) AS max_avg_rating,
    MIN(total_votes) AS min_total_votes,
    MAX(total_votes) AS max_total_votes,
    MIN(median_rating) AS min_median_rating,
    MAX(median_rating) AS max_median_rating
FROM ratings;

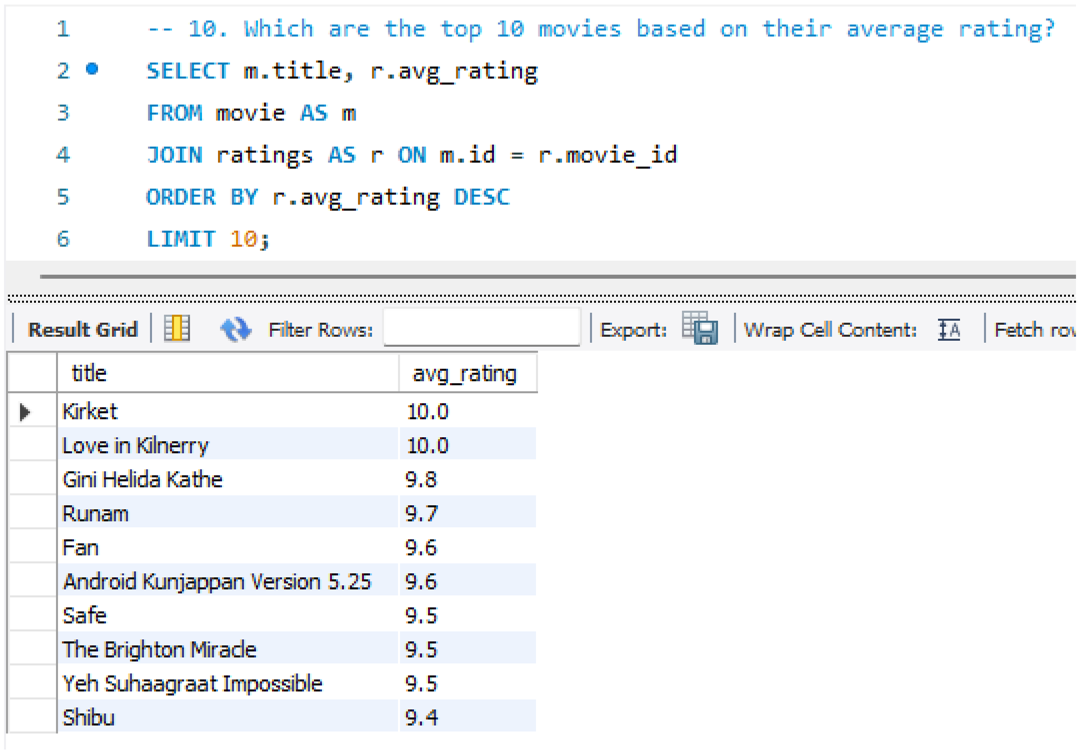


```
-- 9. Find the minimum and maximum values for each column in the
-- ratings table, excluding the movie_id column.
SELECT
    MIN(avg_rating) AS min_avg_rating,
    MAX(avg_rating) AS max_avg_rating,
    MIN(total_votes) AS min_total_votes,
    MAX(total_votes) AS max_total_votes,
    MIN(median_rating) AS min_median_rating,
    MAX(median_rating) AS max_median_rating
FROM ratings;
```

min_avg_rating	max_avg_rating	min_total_votes	max_total_votes	min_median_rating	max_median_rating
1.0	10.0	100	725138	1	10

10. List the top 10 movies by average rating.

```
SELECT m.title, r.avg_rating
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
ORDER BY r.avg_rating DESC
LIMIT 10;
```

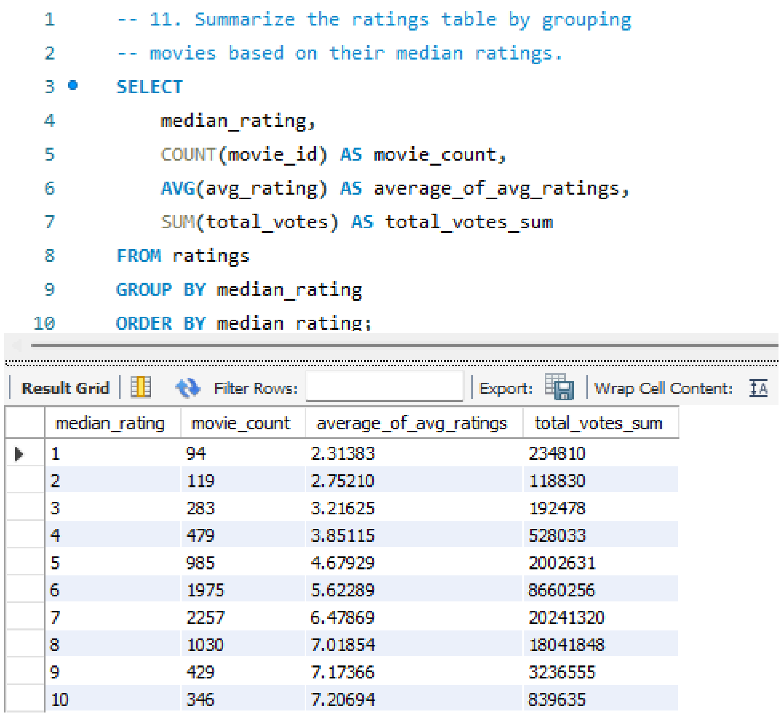


```
-- 10. Which are the top 10 movies based on their average rating?
SELECT m.title, r.avg_rating
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
ORDER BY r.avg_rating DESC
LIMIT 10;
```

title	avg_rating
Kirket	10.0
Love in Kilnerry	10.0
Gini Helida Kathe	9.8
Runam	9.7
Fan	9.6
Android Kunjappan Version 5.25	9.6
Safe	9.5
The Brighton Miracle	9.5
Yeh Suhaagraat Impossible	9.5
Shibu	9.4

11. Summarize the `ratings` table by median rating.

```
SELECT
    median_rating,
    COUNT(movie_id) AS movie_count,
    AVG(avg_rating) AS average_of_avg_ratings,
    SUM(total_votes) AS total_votes_sum
FROM ratings
GROUP BY median_rating
ORDER BY median_rating;
```

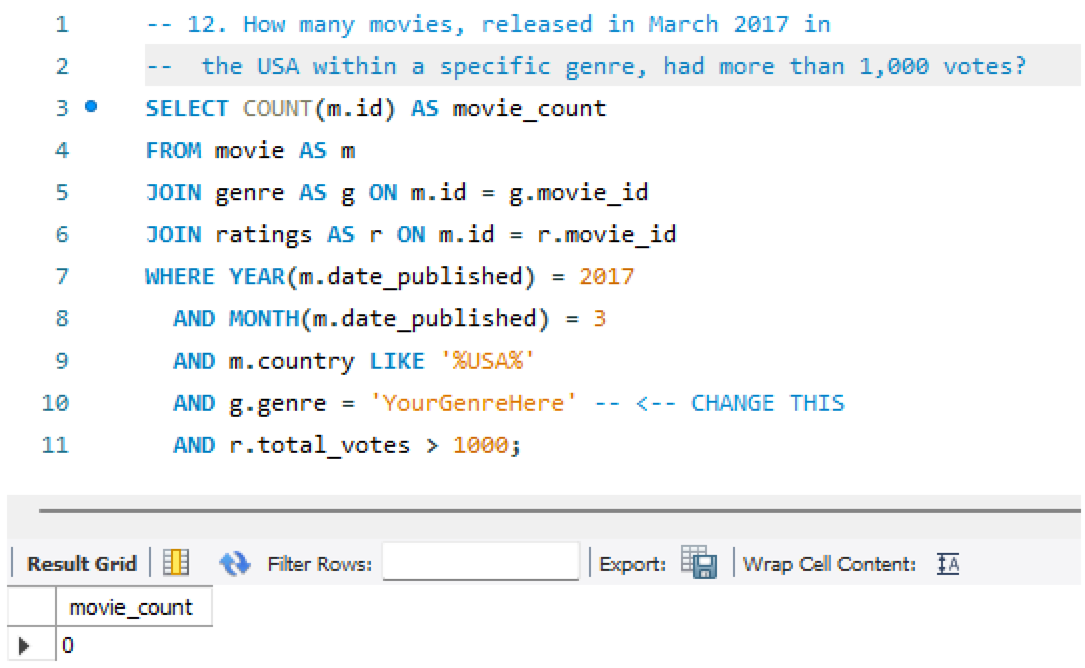


```
-- 11. Summarize the ratings table by grouping
-- movies based on their median ratings.
SELECT
    median_rating,
    COUNT(movie_id) AS movie_count,
    AVG(avg_rating) AS average_of_avg_ratings,
    SUM(total_votes) AS total_votes_sum
FROM ratings
GROUP BY median_rating
ORDER BY median_rating;
```

median_rating	movie_count	average_of_avg_ratings	total_votes_sum
1	94	2.31383	234810
2	119	2.75210	118830
3	283	3.21625	192478
4	479	3.85115	528033
5	985	4.67929	2002631
6	1975	5.62289	8660256
7	2257	6.47869	20241320
8	1030	7.01854	18041848
9	429	7.17366	3236555
10	346	7.20694	839635

12. Count movies from March 2017 in the USA of a specific genre with >1,000 votes.

```
-- Note: Replace 'YourGenreHere' with the actual genre you're interested in.
SELECT COUNT(m.id) AS movie_count
FROM movie AS m
JOIN genre AS g ON m.id = g.movie_id
JOIN ratings AS r ON m.id = r.movie_id
WHERE YEAR(m.date_published) = 2017
  AND MONTH(m.date_published) = 3
  AND m.country LIKE '%USA%'
  AND g.genre = 'YourGenreHere' -- <-- CHANGE THIS
  AND r.total_votes > 1000;
```

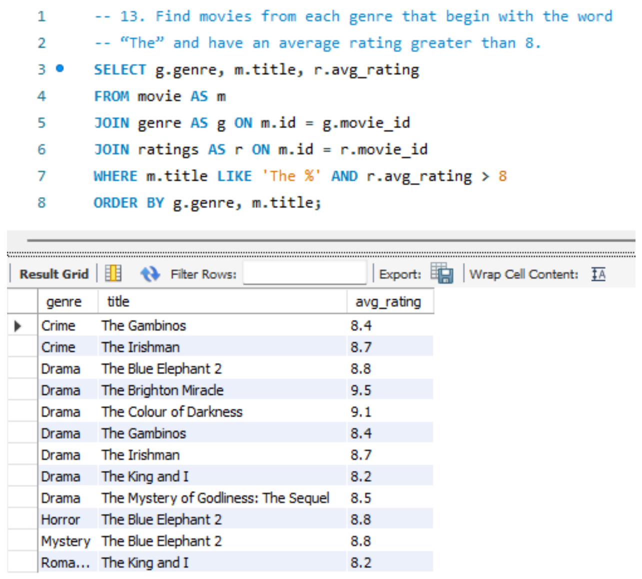


```
-- 12. How many movies, released in March 2017 in
--  the USA within a specific genre, had more than 1,000 votes?
SELECT COUNT(m.id) AS movie_count
FROM movie AS m
JOIN genre AS g ON m.id = g.movie_id
JOIN ratings AS r ON m.id = r.movie_id
WHERE YEAR(m.date_published) = 2017
  AND MONTH(m.date_published) = 3
  AND m.country LIKE '%USA%'
  AND g.genre = 'YourGenreHere' -- <-- CHANGE THIS
  AND r.total_votes > 1000;
```

movie_count
0

13. Find movies starting with "The" with an average rating > 8.

```
SELECT g.genre, m.title, r.avg_rating
FROM movie AS m
JOIN genre AS g ON m.id = g.movie_id
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.title LIKE 'The %' AND r.avg_rating > 8
ORDER BY g.genre, m.title;
```

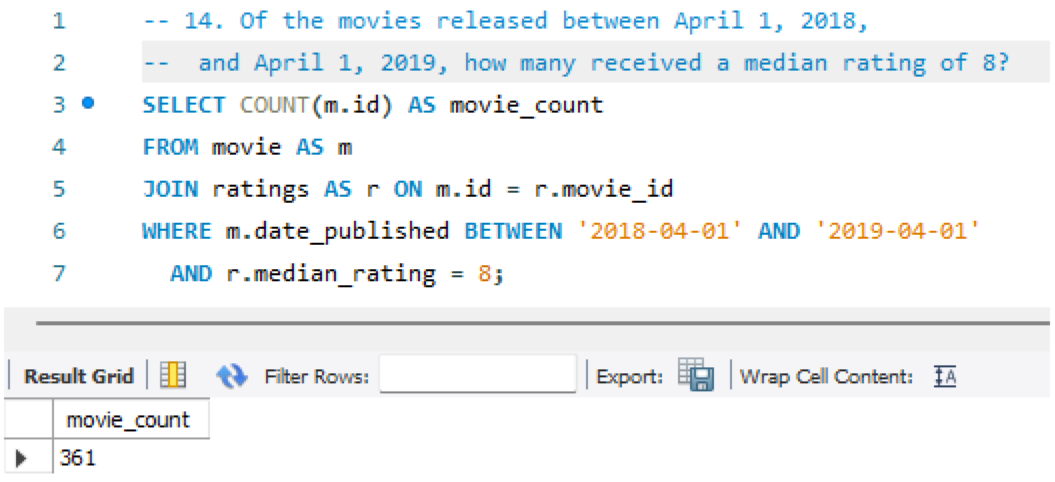


```
-- 13. Find movies from each genre that begin with the word
-- "The" and have an average rating greater than 8.
SELECT g.genre, m.title, r.avg_rating
FROM movie AS m
JOIN genre AS g ON m.id = g.movie_id
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.title LIKE 'The %' AND r.avg_rating > 8
ORDER BY g.genre, m.title;
```

genre	title	avg_rating
Crime	The Gambinos	8.4
Crime	The Irishman	8.7
Drama	The Blue Elephant 2	8.8
Drama	The Brighton Miracle	9.5
Drama	The Colour of Darkness	9.1
Drama	The Gambinos	8.4
Drama	The Irishman	8.7
Drama	The King and I	8.2
Drama	The Mystery of Godliness: The Sequel	8.5
Horror	The Blue Elephant 2	8.8
Mystery	The Blue Elephant 2	8.8
Roma...	The King and I	8.2

14. Count movies from Apr 2018 - Apr 2019 with a median rating of 8.

```
SELECT COUNT(m.id) AS movie_count
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.date_published BETWEEN '2018-04-01' AND '2019-04-01'
  AND r.median_rating = 8;
```

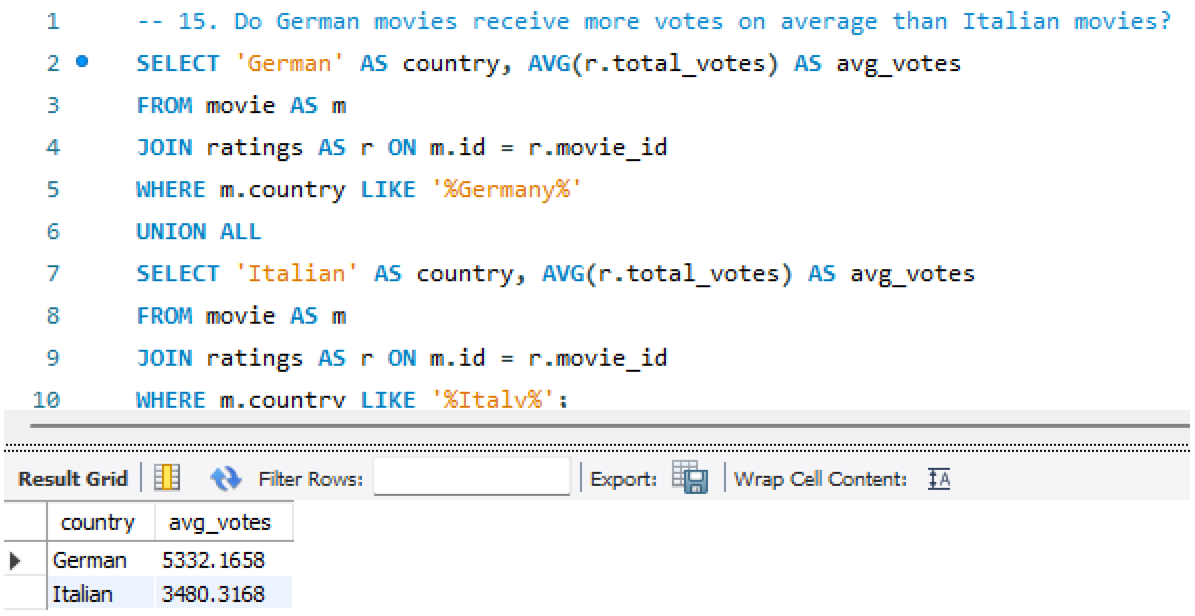


```
-- 14. Of the movies released between April 1, 2018,
--  and April 1, 2019, how many received a median rating of 8?
SELECT COUNT(m.id) AS movie_count
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.date_published BETWEEN '2018-04-01' AND '2019-04-01'
  AND r.median_rating = 8;
```

```
movie_count
361
```

15. Compare the average votes for German vs. Italian movies.

```
SELECT 'German' AS country, AVG(r.total_votes) AS avg_votes
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.country LIKE '%Germany%'
UNION ALL
SELECT 'Italian' AS country, AVG(r.total_votes) AS avg_votes
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.country LIKE '%Italy%';
```

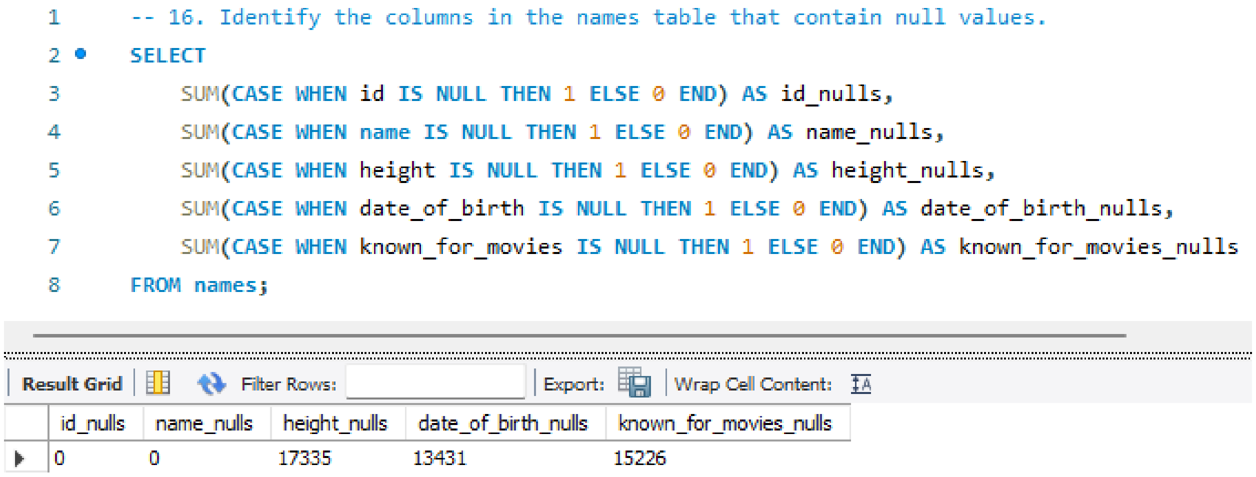


```
-- 15. Do German movies receive more votes on average than Italian movies?
SELECT 'German' AS country, AVG(r.total_votes) AS avg_votes
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.country LIKE '%Germany%'
UNION ALL
SELECT 'Italian' AS country, AVG(r.total_votes) AS avg_votes
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.country LIKE '%Italy%';
```

```
country   avg_votes
German    5332.1658
Italian   3480.3168
```

16. Find columns with null values in the names table.

```
SELECT
   SUM(CASE WHEN id IS NULL THEN 1 ELSE 0 END) AS id_nulls,
   SUM(CASE WHEN name IS NULL THEN 1 ELSE 0 END) AS name_nulls,
   SUM(CASE WHEN height IS NULL THEN 1 ELSE 0 END) AS height_nulls,
   SUM(CASE WHEN date_of_birth IS NULL THEN 1 ELSE 0 END) AS date_of_birth_nulls,
   SUM(CASE WHEN known_for_movies IS NULL THEN 1 ELSE 0 END) AS known_for_movies_nulls
FROM names;
```

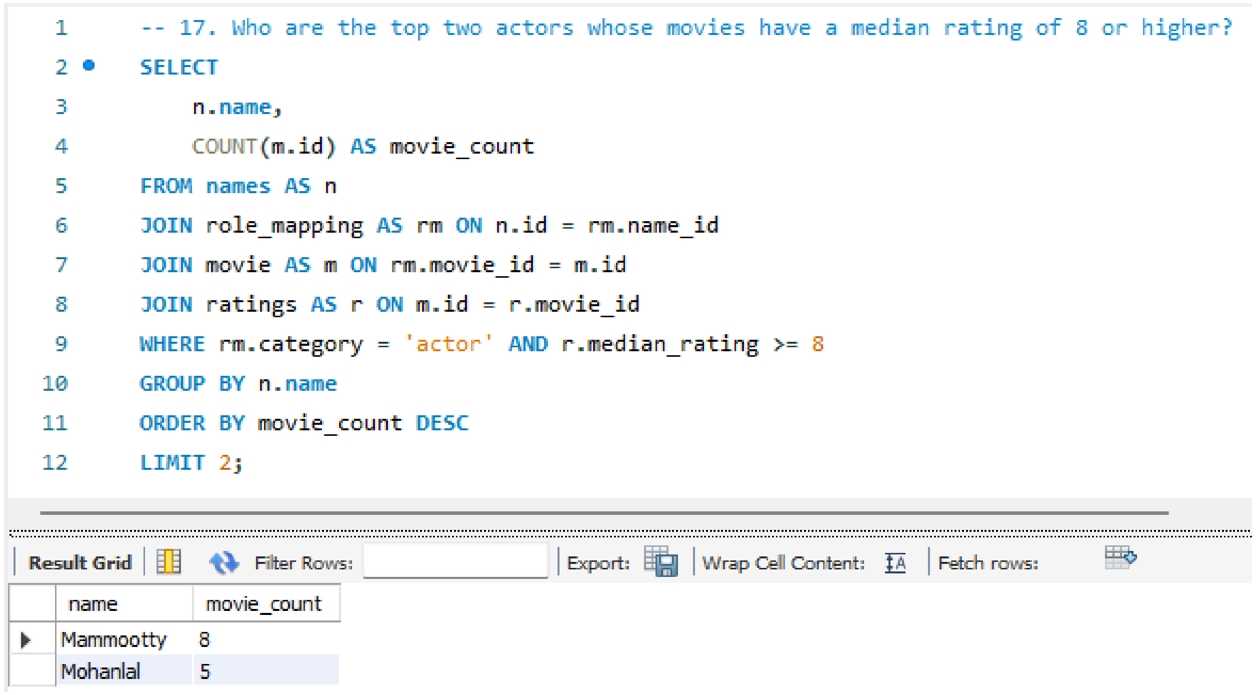


```
-- 16. Identify the columns in the names table that contain null values.
SELECT
    SUM(CASE WHEN id IS NULL THEN 1 ELSE 0 END) AS id_nulls,
    SUM(CASE WHEN name IS NULL THEN 1 ELSE 0 END) AS name_nulls,
    SUM(CASE WHEN height IS NULL THEN 1 ELSE 0 END) AS height_nulls,
    SUM(CASE WHEN date_of_birth IS NULL THEN 1 ELSE 0 END) AS date_of_birth_nulls,
    SUM(CASE WHEN known_for_movies IS NULL THEN 1 ELSE 0 END) AS known_for_movies_nulls
FROM names;
```

id_nulls	name_nulls	height_nulls	date_of_birth_nulls	known_for_movies_nulls
0	0	17335	13431	15226

17. Find the top two actors in movies with a median rating >= 8.

```
SELECT
    n.name,
    COUNT(m.id) AS movie_count
FROM names AS n
JOIN role_mapping AS rm ON n.id = rm.name_id
JOIN movie AS m ON rm.movie_id = m.id
JOIN ratings AS r ON m.id = r.movie_id
WHERE rm.category = 'actor' AND r.median_rating >= 8
GROUP BY n.name
ORDER BY movie_count DESC
LIMIT 2;
```

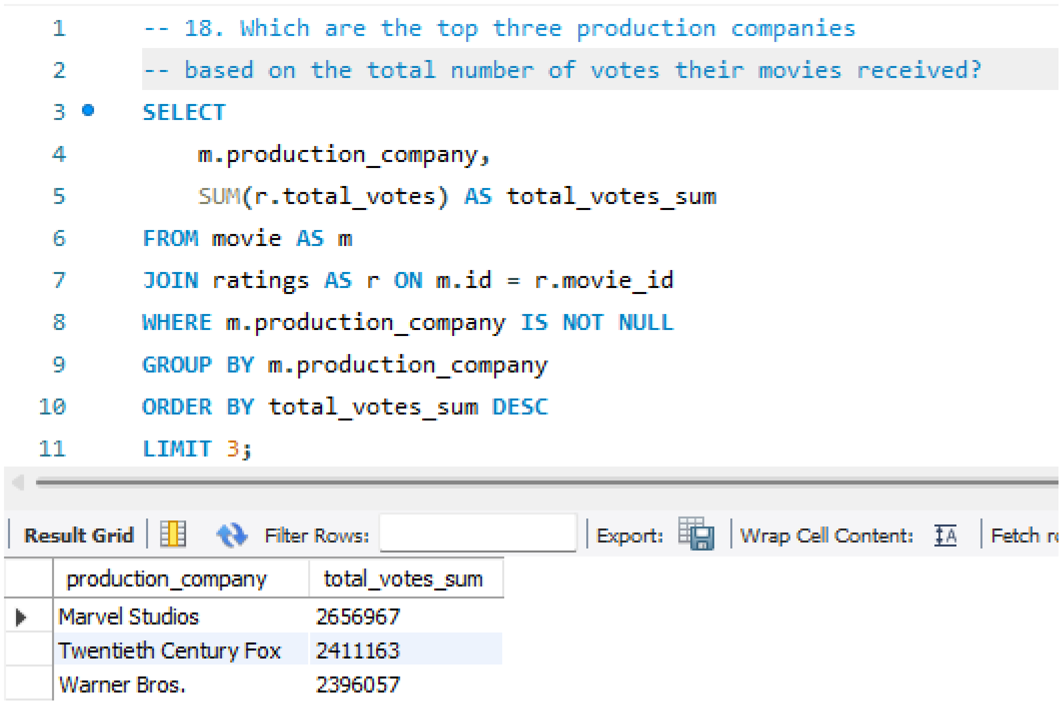


```
-- 17. Who are the top two actors whose movies have a median rating of 8 or higher?
SELECT
    n.name,
    COUNT(m.id) AS movie_count
FROM names AS n
JOIN role_mapping AS rm ON n.id = rm.name_id
JOIN movie AS m ON rm.movie_id = m.id
JOIN ratings AS r ON m.id = r.movie_id
WHERE rm.category = 'actor' AND r.median_rating >= 8
GROUP BY n.name
ORDER BY movie_count DESC
LIMIT 2;
```

name	movie_count
Mammootty	8
Mohanlal	5

18. List the top three production companies by total votes.

```
SELECT
    m.production_company,
    SUM(r.total_votes) AS total_votes_sum
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.production_company IS NOT NULL
GROUP BY m.production_company
ORDER BY total_votes_sum DESC
LIMIT 3;
```

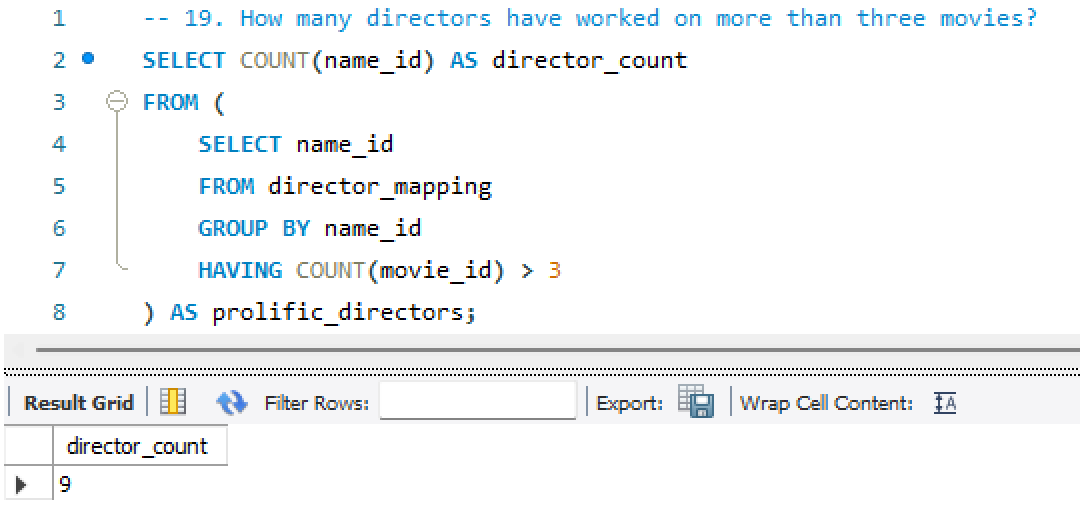


```
-- 18. Which are the top three production companies
-- based on the total number of votes their movies received?
SELECT
    m.production_company,
    SUM(r.total_votes) AS total_votes_sum
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.production_company IS NOT NULL
GROUP BY m.production_company
ORDER BY total_votes_sum DESC
LIMIT 3;
```

production_company	total_votes_sum
Marvel Studios	2656967
Twentieth Century Fox	2411163
Warner Bros.	2396057

19. Count directors who have worked on more than three movies.

```
SELECT COUNT(name_id) AS director_count
FROM (
    SELECT name_id
    FROM director_mapping
    GROUP BY name_id
    HAVING COUNT(movie_id) > 3
) AS prolific_directors;
```



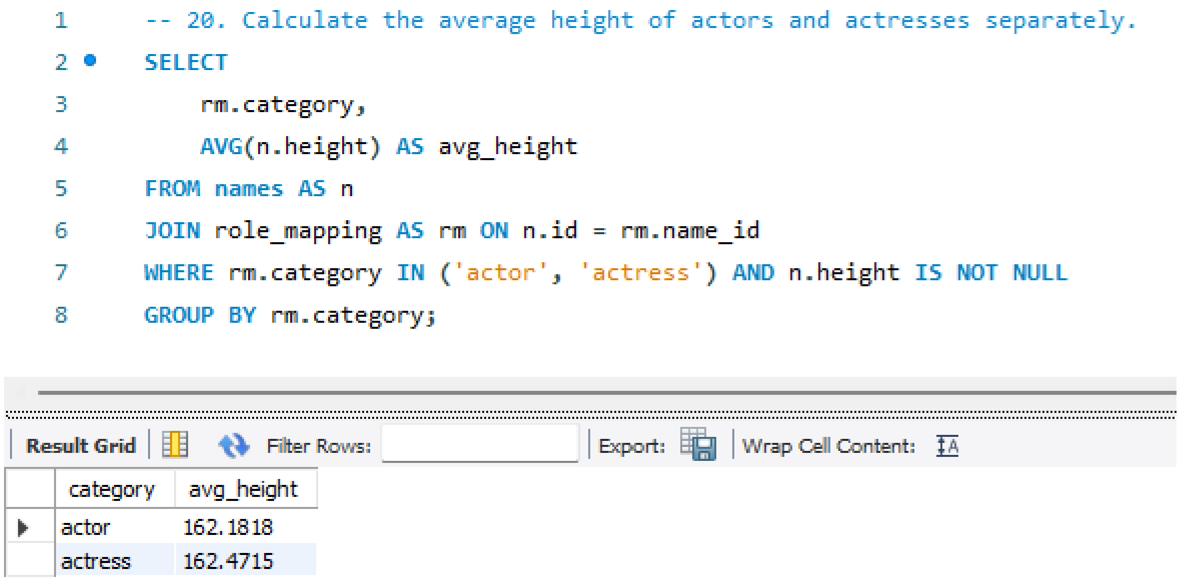
```
-- 19. How many directors have worked on more than three movies?
SELECT COUNT(name_id) AS director_count
FROM (
    SELECT name_id
    FROM director_mapping
    GROUP BY name_id
    HAVING COUNT(movie_id) > 3
) AS prolific_directors;
```

director_count
9

20. Calculate the average height for actors and actresses separately.

```
SELECT
    rm.category,
    AVG(n.height) AS avg_height
```

```
FROM names AS n
JOIN role_mapping AS rm ON n.id = rm.name_id
WHERE rm.category IN ('actor', 'actress') AND n.height IS NOT NULL
GROUP BY rm.category;
```

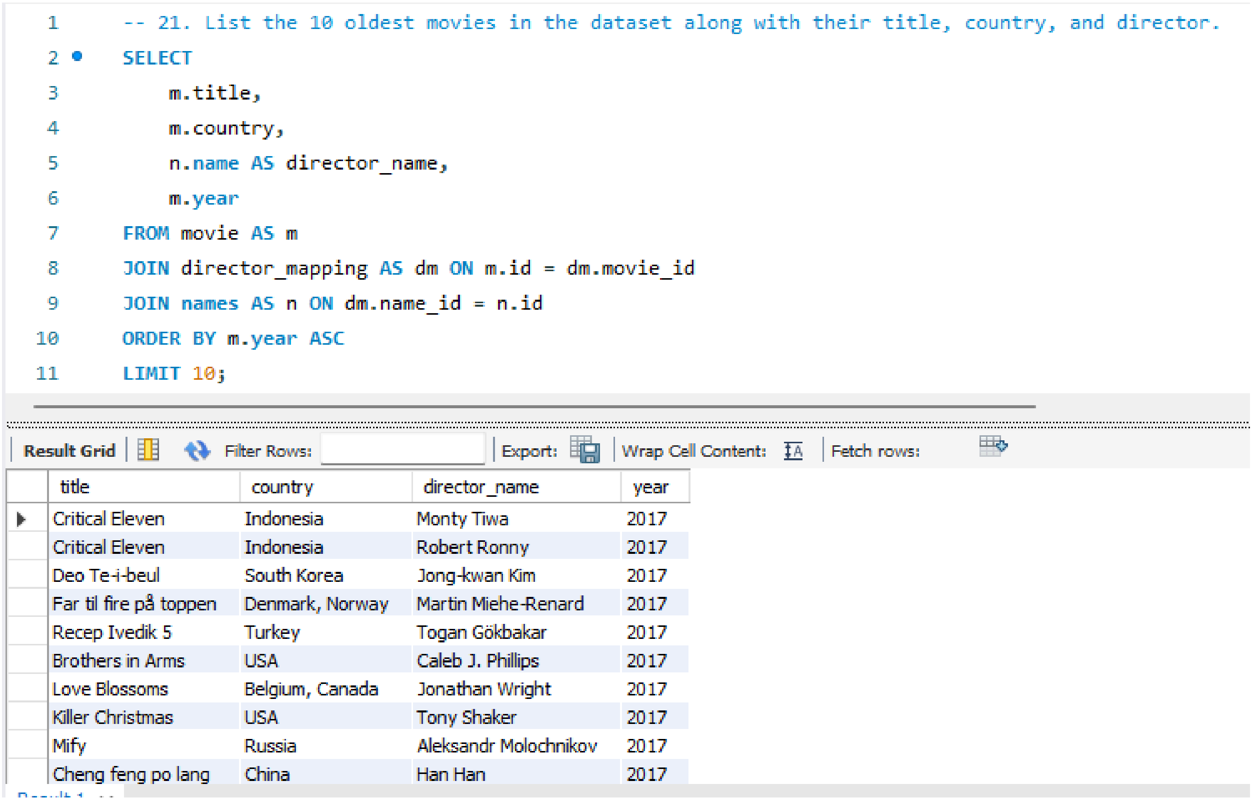


```
-- 20. Calculate the average height of actors and actresses separately.
SELECT
    rm.category,
    AVG(n.height) AS avg_height
FROM names AS n
JOIN role_mapping AS rm ON n.id = rm.name_id
WHERE rm.category IN ('actor', 'actress') AND n.height IS NOT NULL
GROUP BY rm.category;
```

category	avg_height
actor	162.1818
actress	162.4715

21. List the 10 oldest movies with their title, country, and director.

```
SELECT
   m.title,
   m.country,
   n.name AS director_name,
   m.year
FROM movie AS m
JOIN director_mapping AS dm ON m.id = dm.movie_id
JOIN names AS n ON dm.name_id = n.id
ORDER BY m.year ASC
LIMIT 10;
```

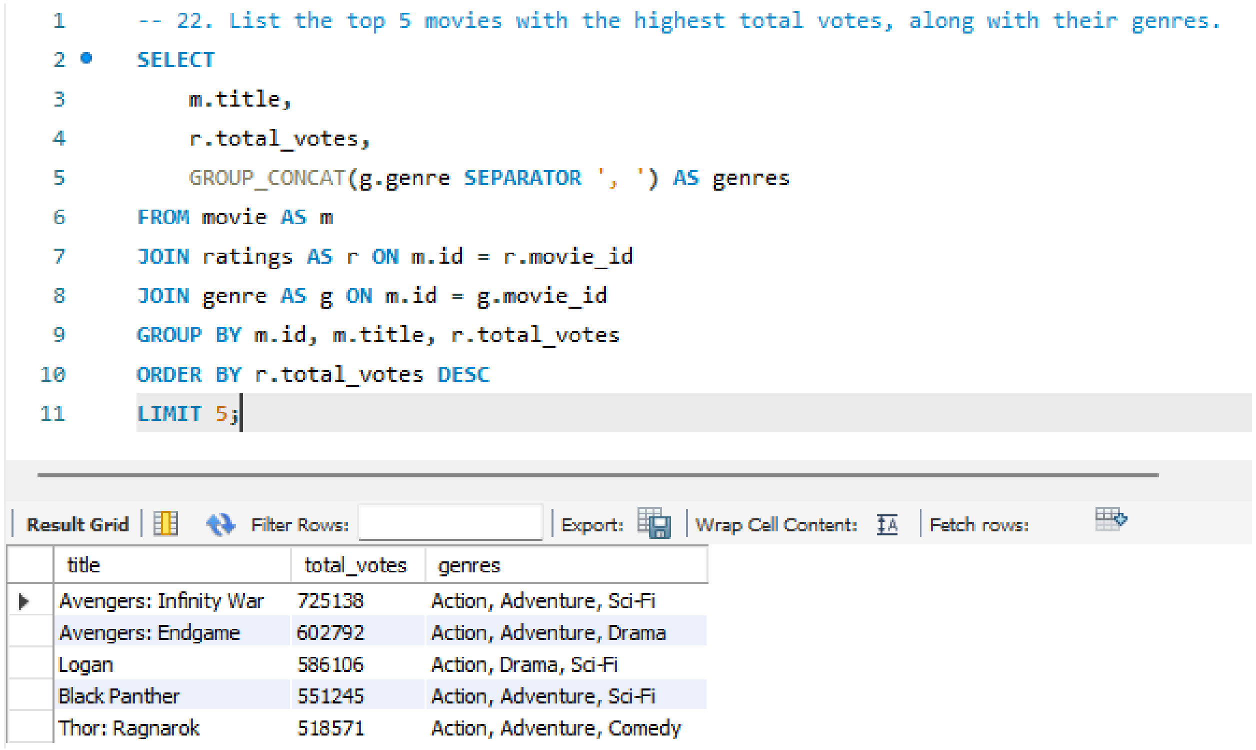


```
-- 21. List the 10 oldest movies in the dataset along with their title, country, and director.
SELECT
    m.title,
    m.country,
    n.name AS director_name,
    m.year
FROM movie AS m
JOIN director_mapping AS dm ON m.id = dm.movie_id
JOIN names AS n ON dm.name_id = n.id
ORDER BY m.year ASC
LIMIT 10;
```

title	country	director_name	year
Critical Eleven	Indonesia	Monty Tiwa	2017
Critical Eleven	Indonesia	Robert Ronny	2017
Deo Te-i-beul	South Korea	Jong-kwan Kim	2017
Far til fire på toppen	Denmark, Norway	Martin Miehe-Renard	2017
Recep Ivedik 5	Turkey	Togan Gökbakar	2017
Brothers in Arms	USA	Caleb J. Phillips	2017
Love Blossoms	Belgium, Canada	Jonathan Wright	2017
Killer Christmas	USA	Tony Shaker	2017
Mify	Russia	Aleksandr Molochnikov	2017
Cheng feng po lang	China	Han Han	2017

22. List the top 5 movies by total votes, including their genres.

```
SELECT
    m.title,
    r.total_votes,
    GROUP_CONCAT(g.genre SEPARATOR ', ') AS genres
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
JOIN genre AS g ON m.id = g.movie_id
GROUP BY m.id, m.title, r.total_votes
ORDER BY r.total_votes DESC
LIMIT 5;
```

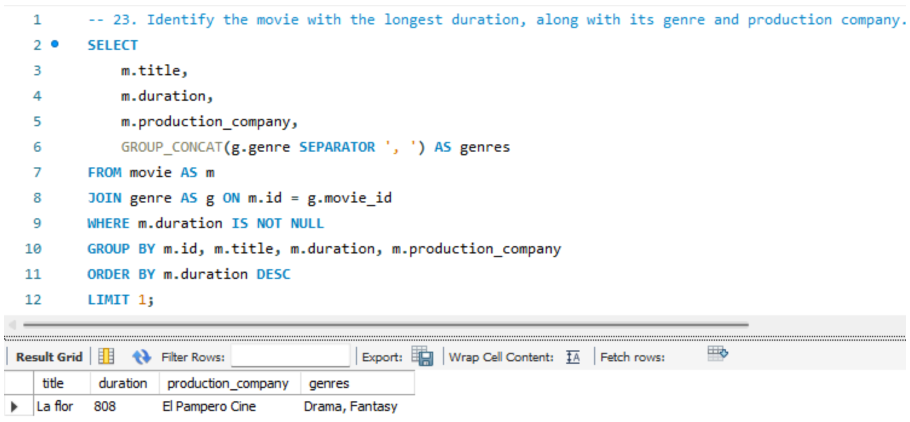


```
-- 22. List the top 5 movies with the highest total votes, along with their genres.
SELECT
    m.title,
    r.total_votes,
    GROUP_CONCAT(g.genre SEPARATOR ', ') AS genres
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
JOIN genre AS g ON m.id = g.movie_id
GROUP BY m.id, m.title, r.total_votes
ORDER BY r.total_votes DESC
LIMIT 5;
```

title	total_votes	genres
Avengers: Infinity War	725138	Action, Adventure, Sci-Fi
Avengers: Endgame	602792	Action, Adventure, Drama
Logan	586106	Action, Drama, Sci-Fi
Black Panther	551245	Action, Adventure, Sci-Fi
Thor: Ragnarok	518571	Action, Adventure, Comedy

23. Find the longest movie with its genre and production company.

```
SELECT
    m.title,
    m.duration,
    m.production_company,
    GROUP_CONCAT(g.genre SEPARATOR ', ') AS genres
FROM movie AS m
JOIN genre AS g ON m.id = g.movie_id
WHERE m.duration IS NOT NULL
GROUP BY m.id, m.title, m.duration, m.production_company
ORDER BY m.duration DESC
LIMIT 1;
```

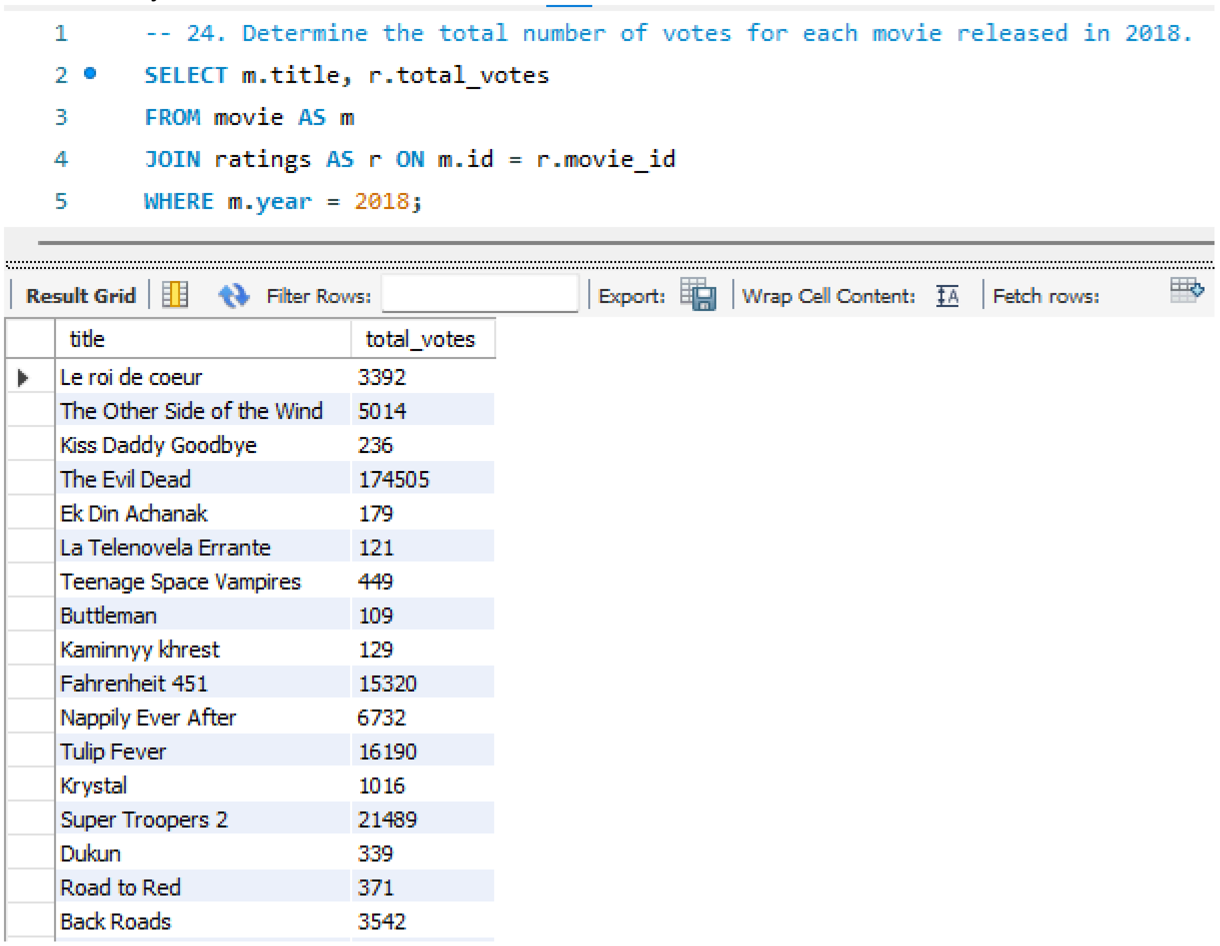


```
-- 23. Identify the movie with the longest duration, along with its genre and production company.
SELECT
    m.title,
    m.duration,
    m.production_company,
    GROUP_CONCAT(g.genre SEPARATOR ', ') AS genres
FROM movie AS m
JOIN genre AS g ON m.id = g.movie_id
WHERE m.duration IS NOT NULL
GROUP BY m.id, m.title, m.duration, m.production_company
ORDER BY m.duration DESC
LIMIT 1;
```

title	duration	production_company	genres
La flor	808	El Pampero Cine	Drama, Fantasy

24. Get the total votes for each movie from 2018.

SELECT m.title, r.total_votes
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.year = 2018;

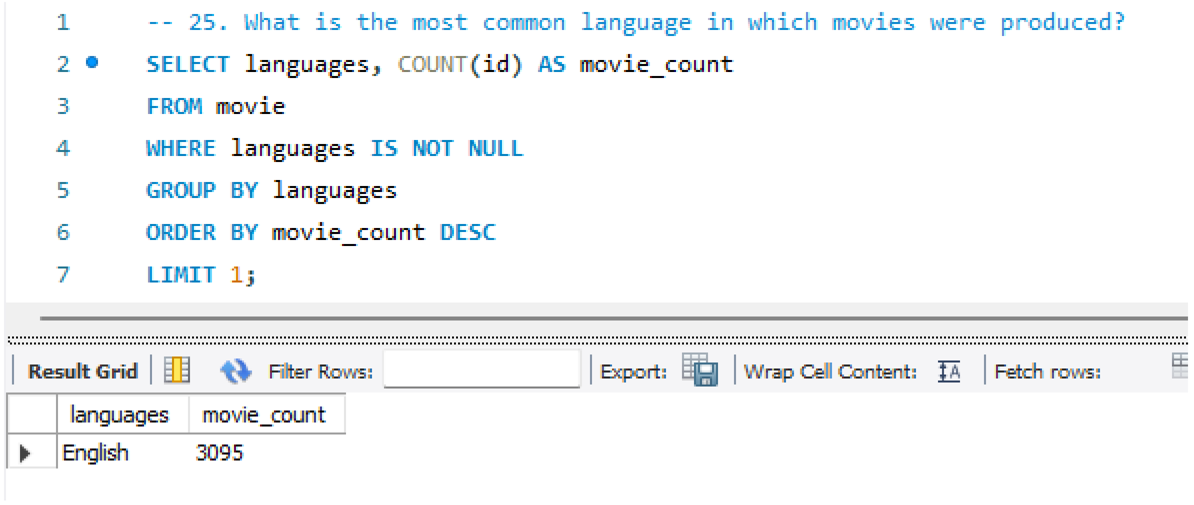


```sql
-- 24. Determine the total number of votes for each movie released in 2018.
SELECT m.title, r.total_votes
FROM movie AS m
JOIN ratings AS r ON m.id = r.movie_id
WHERE m.year = 2018;
```

title	total_votes
Le roi de coeur	3392
The Other Side of the Wind	5014
Kiss Daddy Goodbye	236
The Evil Dead	174505
Ek Din Achanak	179
La Telenovela Errante	121
Teenage Space Vampires	449
Buttleman	109
Kaminnyy khrest	129
Fahrenheit 451	15320
Nappily Ever After	6732
Tulip Fever	16190
Krystal	1016
Super Troopers 2	21489
Dukun	339
Road to Red	371
Back Roads	3542

25. Find the most common movie language.

SELECT languages, COUNT(id) AS movie_count
FROM movie
WHERE languages IS NOT NULL
GROUP BY languages
ORDER BY movie_count DESC
LIMIT 1;



```sql
-- 25. What is the most common language in which movies were produced?
SELECT languages, COUNT(id) AS movie_count
FROM movie
WHERE languages IS NOT NULL
GROUP BY languages
ORDER BY movie_count DESC
LIMIT 1;
```

languages	movie_count
English	3095