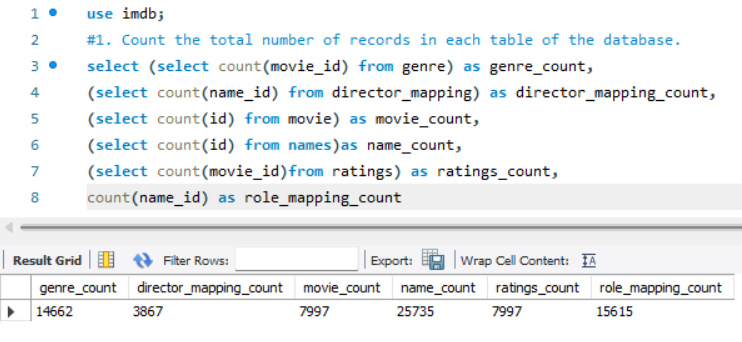
**IMDB movies database**

This database contains detailed information about movies, including their titles, release dates, genres, languages, and production companies. It also includes movie ratings, total votes, budgets, and revenues to analyze their popularity and financial success. The database has details about actors, actresses, and directors, helping to understand their contributions to the film industry. It tracks the most common languages and countries producing movies. Additionally, it includes data on movie durations and award-winning films. By analyzing votes and ratings, we can identify the most popular movies over time. The database helps in understanding trends in movie production and audience preferences. It is useful for researchers, analysts, and movie enthusiasts to explore insights about the film industry.

1.



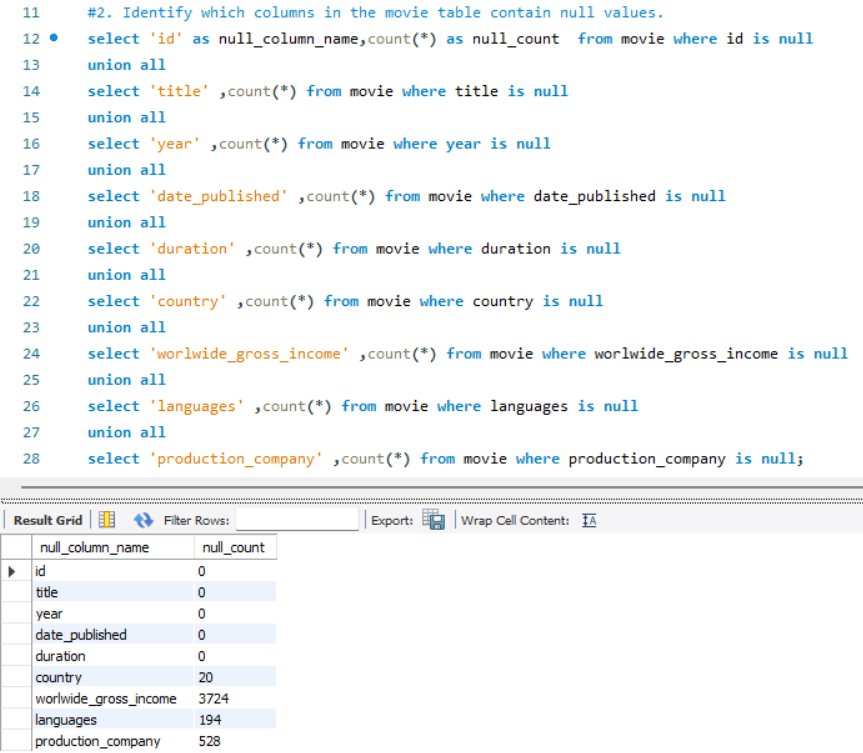
Query explanation:

This SQL query counts the total number of records in different tables of a database. It gets the number of entries in the genre, director mapping, movie, names, and ratings tables using subqueries. It also counts the records in the role mapping table directly and shows all counts in one row.

Answer:

The query counts the total number of records in different tables of the IMDb database and returns the results as column values. The output shows that the genre table has **14,662** records, director\_mapping has **3,867**, movie has **7,997**, names has **25,735**, ratings has **7,997**, and role\_mapping has **15,615** records.

2.



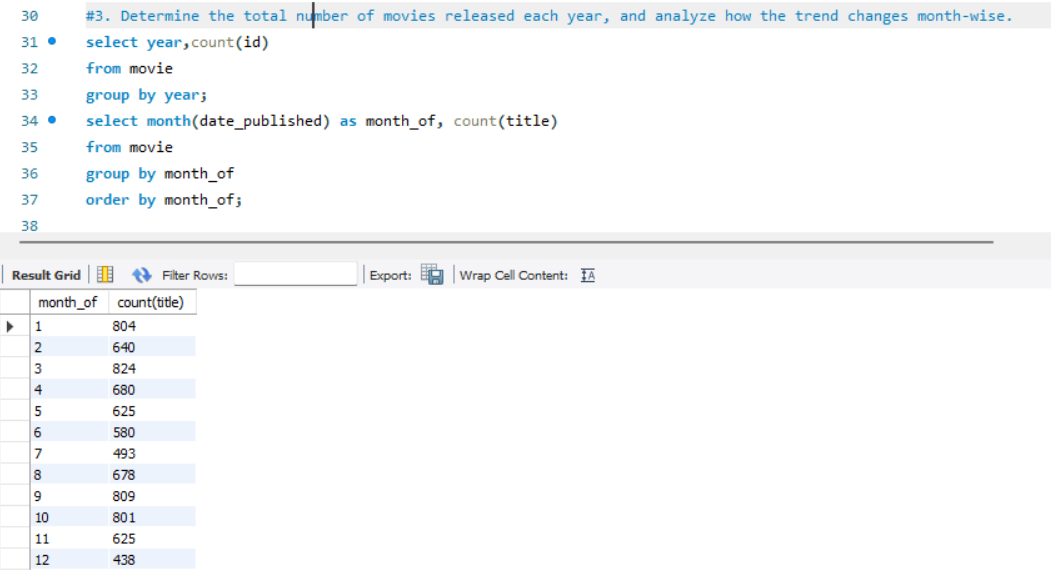
Query explanation:

This SQL query checks which columns in the movie table have NULL values and counts how many NULL values each column has. It does this by selecting each column one by one, counting the NULL values, and combining the results using UNION ALL. The final output shows a list of columns with their respective NULL counts.

Answer:

The query checks which columns in the movie table contain NULL values and counts them. The output shows that **worldwide\_gross\_income** has the most missing values (**3,724**), followed by **production\_company** (**528**), **languages** (**194**), and **country** (**20**), while other columns have no missing values.

3.



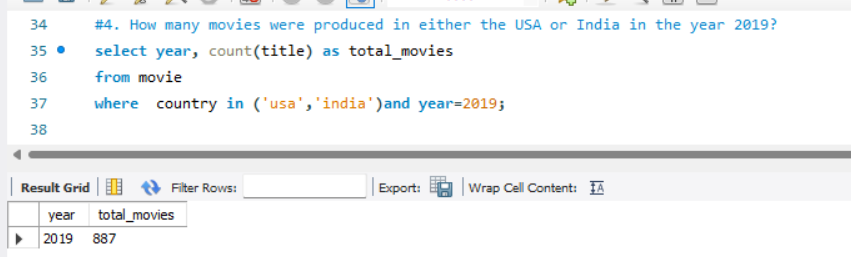
Query explanation

This query finds how many movies were released each year and how the trend changes month by month. The **COUNT(id)** function counts total movies per year, while **MONTH(date\_published)** extracts the month from the release date. The results are grouped using **GROUP BY** to analyze yearly and monthly movie releases.

Answer:

The query shows the number of movies released each year and the distribution of releases across months. The result reveals that **January (804), March (824), September (809), and October (801)** had the highest movie releases, while **December (438) had the lowest**.

4.



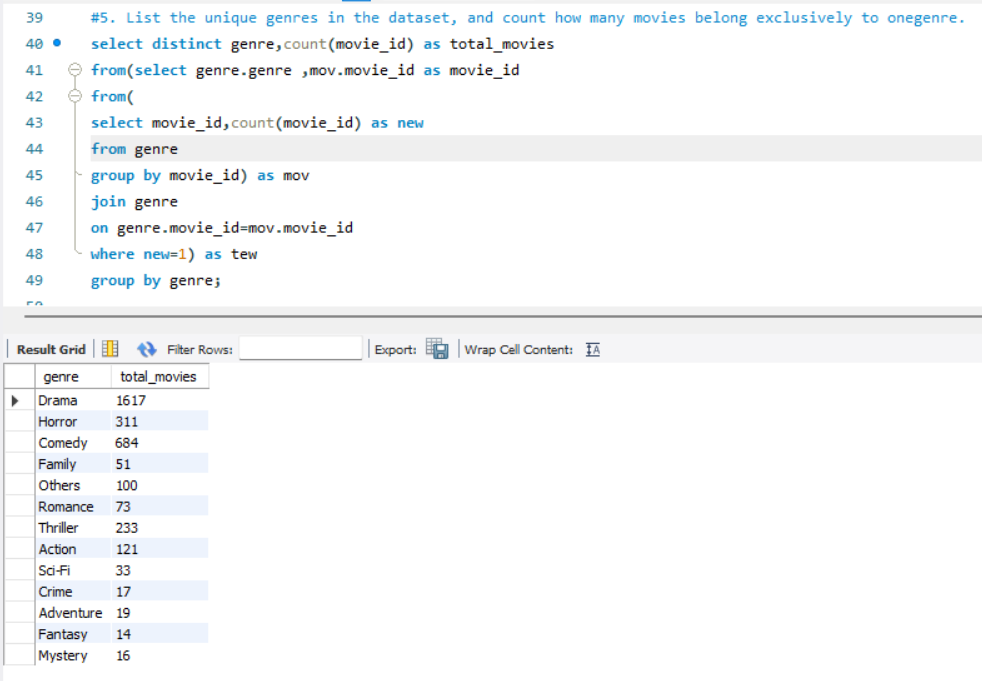
Query explanation:

This SQL query finds how many movies were made in the USA or India in 2019. The WHERE clause filters the movie table to include only records where the country is 'USA' or 'India' and the year is 2019. The COUNT(title) function counts the number of movies, and the result shows the year along with the total movie count.

Answer:

The query counts the number of movies produced in **USA or India** in the year **2019**. The result shows that a total of **887 movies** were produced in these two countries during that year.

5.



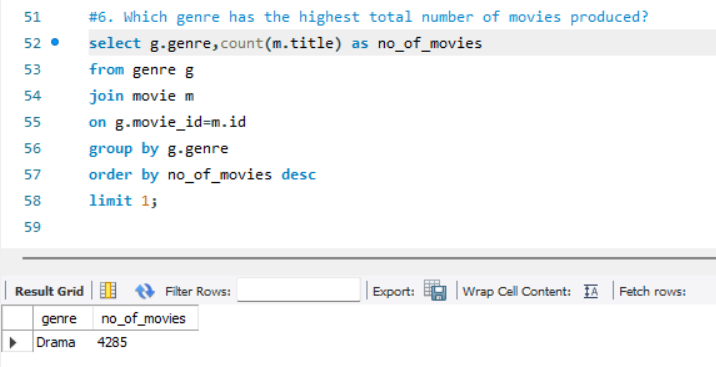
Query explanation:

This SQL query finds all unique movie genres and counts how many movies belong to only one genre. First, it groups movies by movie id in the genre table and counts how many genres each movie has. Then, it filters out movies that belong to more than one genre and keeps only those with a single genre. Finally, it counts the number of such movies for each genre using COUNT(movie id).

Answer:

The query lists unique movie genres and counts how many movies belong **exclusively** to each genre. The result shows that **Drama** has the highest number of such movies (**1617**), followed by **Comedy (684)** and **Horror (311)**.

6.



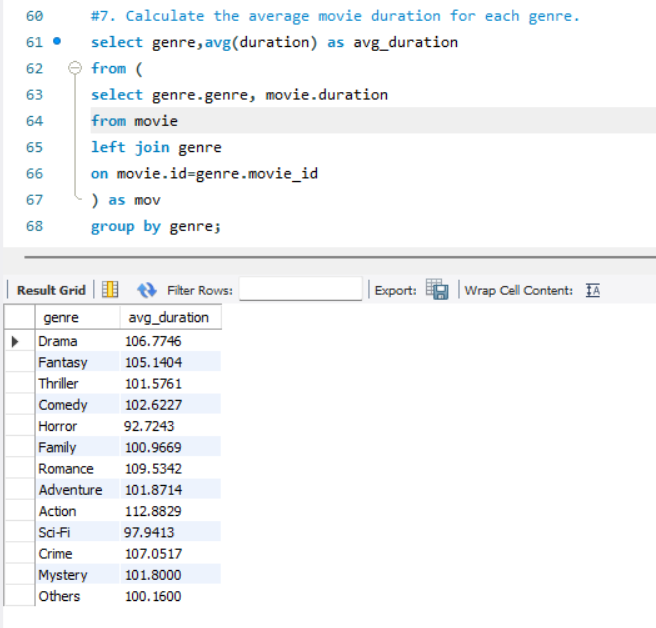
Query explanation:

This SQL query finds the genre with the most movies. It joins the genre table with the movie table using movie id, then groups the data by genre and counts the number of movies in each genre using COUNT(m.title). Finally, it sorts the results in descending order by movie count and uses LIMIT 1 to display only the top genre.

Answer:

The query finds the genre with the highest number of movies produced. The result shows that **Drama** is the most produced genre, with **4,285 movies**.

7.



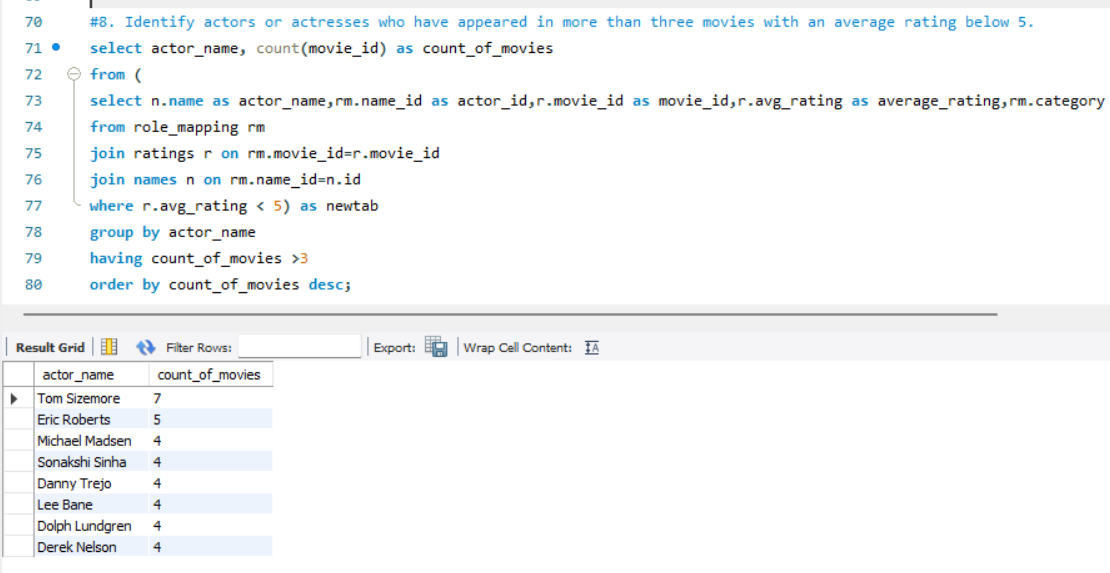
Query explanation:

This SQL query calculates the average duration of movies for each genre. It first joins the movie and genre tables using LEFT JOIN to get the duration of each movie along with its genre. Then, it groups the results by genre and uses the AVG(duration) function to find the average movie length for each genre.

Answer:

The query calculates the average duration of movies for each genre. The result shows that **Action movies have the highest average duration of 112.88 minutes**, while **Horror movies have the shortest average duration of 92.72 minutes**.

8.



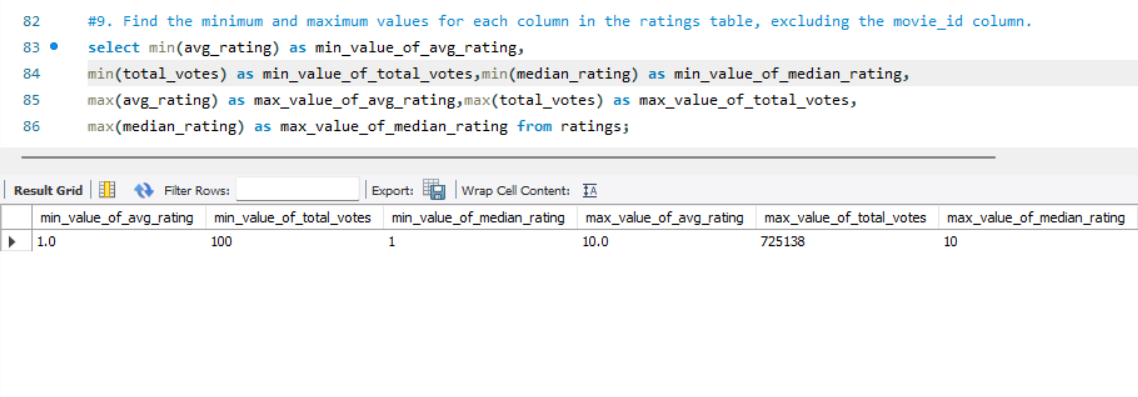
Query explanation:

This SQL query finds actors or actresses who have appeared in more than three movies with an average rating below 5. It first joins the role mapping, ratings, and names tables to get the actor's name, movie ID, and average rating, filtering only movies with a rating below 5. Then, it groups the results by actor name, counts the number of such movies using COUNT(movie id), and filters actors who have appeared in more than three such movies using HAVING count of movies > 3, sorting the list in descending order.

Answer:

The query identifies actors who have appeared in more than three movies with an average rating below 5. **Tom Sizemore has the most such movies (7), followed by Eric Roberts (5) and several actors with 4 movies.**

9.



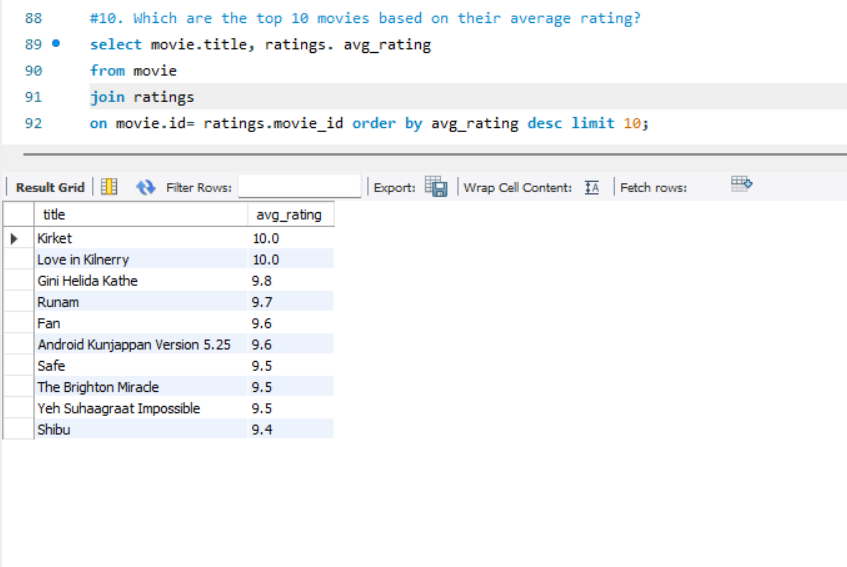
Query explanation:

This SQL query finds the minimum and maximum values for each column in the ratings table, except movie id. It uses the MIN() function to get the smallest values and the MAX() function to get the largest values for avg rating, total votes, and median rating. The result shows the lowest and highest values for each of these columns in a single row.

Answer:

The query finds the minimum and maximum values for average rating, total votes, and median rating from the ratings table. **The lowest average rating is 1.0, the highest is 10.0, total votes range from 100 to 725,138, and median ratings range from 1 to 10.**

10.



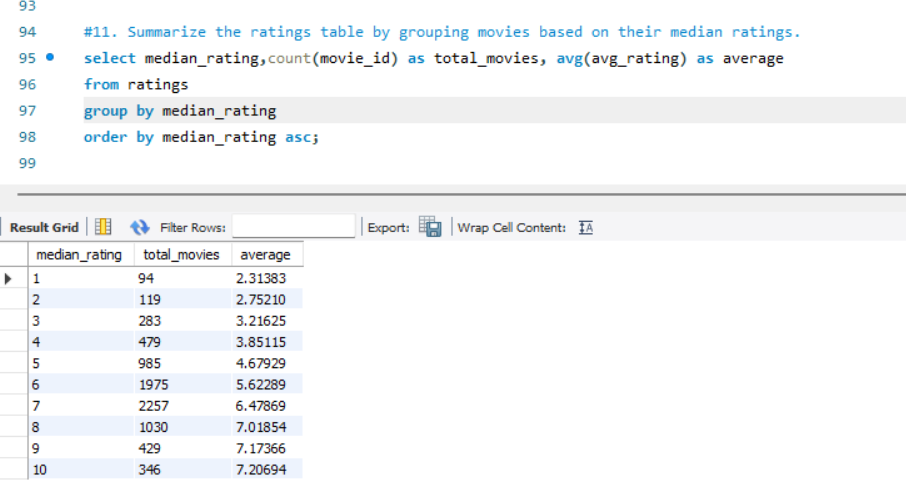
Query explanation:

This SQL query finds the top 10 highest-rated movies based on their average rating. It joins the movie and ratings tables using movie\_id, then sorts the results in descending order using ORDER BY avg\_rating DESC. Finally, it uses LIMIT 10 to display only the top 10 movies with the highest ratings.

Answer:

The query finds the top 10 movies based on their average rating. **The highest-rated movies are "Kirket" and "Love in Kilnerry," both with a perfect rating of 10.0, followed by "Gini Helida Kathe" (9.8) and "Runam" (9.7).**

11.



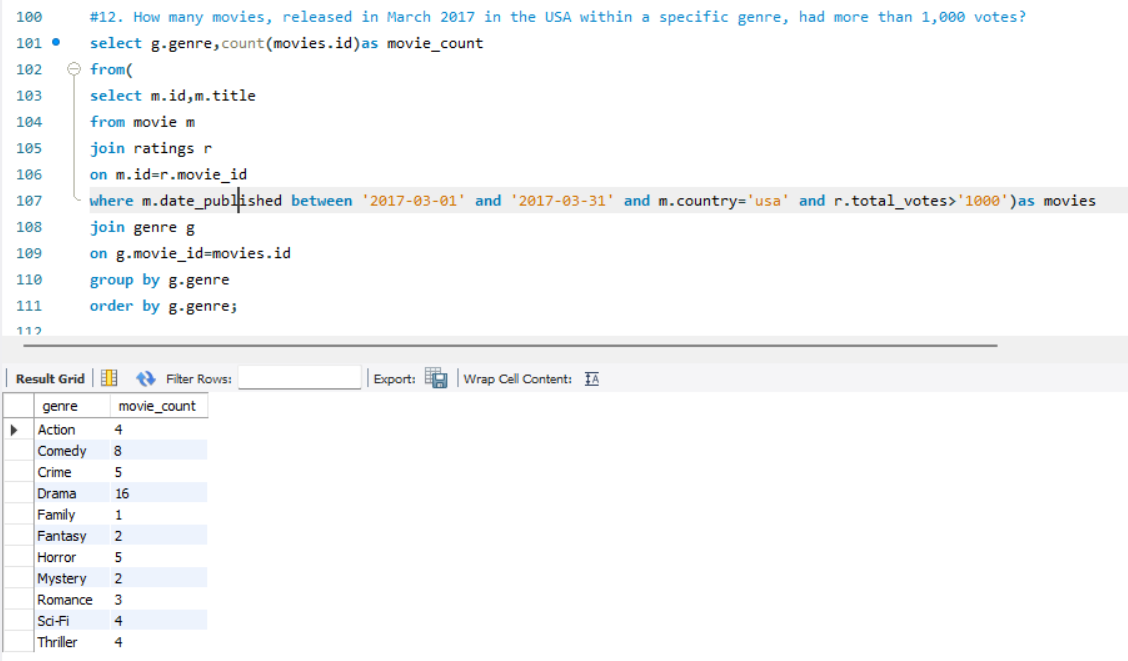
Query explanation:

This SQL query groups movies based on their median rating and provides a summary of the ratings table. It counts how many movies fall into each median rating category using COUNT(movie id) and calculates the average avg rating for each group using AVG(avg rating). The results are sorted in ascending order of median rating using ORDER BY median rating ASC.

Answer:

The query groups movies based on their median rating and calculates the total movies and average rating for each group. **Movies with a median rating of 10 have the highest average rating (7.2064), while those with a median rating of 1 have the lowest (2.3138).**

12.



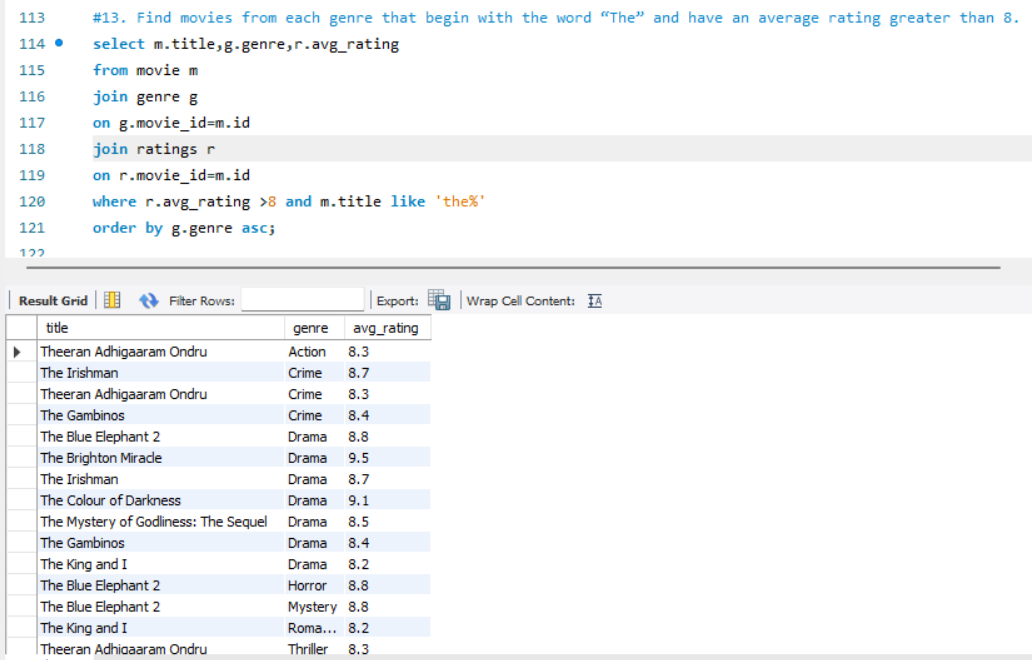
Query explanation:

This SQL query counts how many movies, released in March 2017 in the USA, belong to each genre and have more than 1,000 votes. It first filters the movie table to select films released in March 2017 in the USA and joins it with the ratings table to ensure only movies with more than 1,000 votes are included. Then, it joins this filtered data with the genre table, groups by genre using GROUP BY g.genre, and counts the number of such movies per genre using COUNT(movies.id), sorting the results alphabetically by genre.

Answer:

The query finds the number of movies released in the USA in March 2017, with more than 1,000 votes, grouped by genre. **Drama had the most movies (16), while Family had the least (1).**

13.



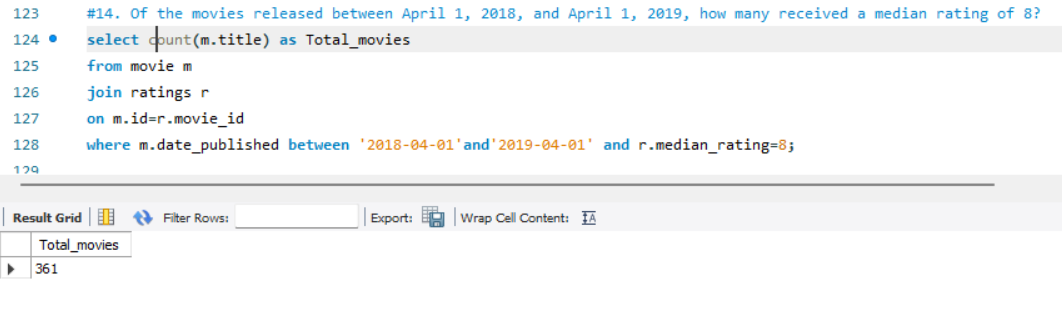
Query explanation:

This SQL query finds movies from each genre that start with "The" and have an average rating greater than 8. It joins the movie, genre, and ratings tables using movie\_id to get the movie title, genre, and rating. The WHERE clause filters movies with an avg\_rating greater than 8 and titles starting with "The" using LIKE 'The%', and the results are sorted by genre in ascending order.

Answer:

The query finds movies from each genre that start with "The" and have an average rating above 8. **Examples include "The Irishman" (Crime, 8.7) and "The Colour of Darkness" (Drama, 9.1).**

14.



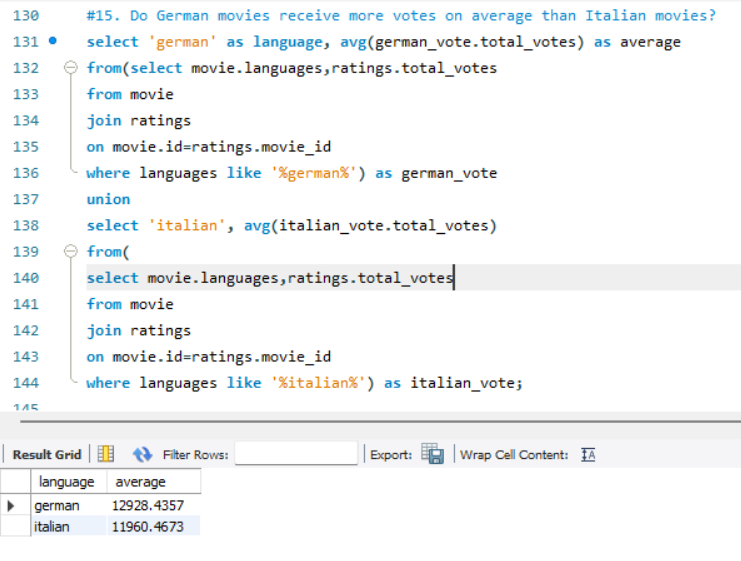
Query explanation:

This SQL query counts the number of movies released between April 1, 2018, and April 1, 2019, that received a median rating of 8. It joins the movie and ratings tables using movie\_id, then filters the results using WHERE to include only movies within the specified date range and with a median\_rating of 8. The COUNT(m.title) function is used to count the total number of such movies.

Answer:

The query counts how many movies, released between April 1, 2018, and April 1, 2019, received a median rating of 8. **The result is 361 movies.**

15.



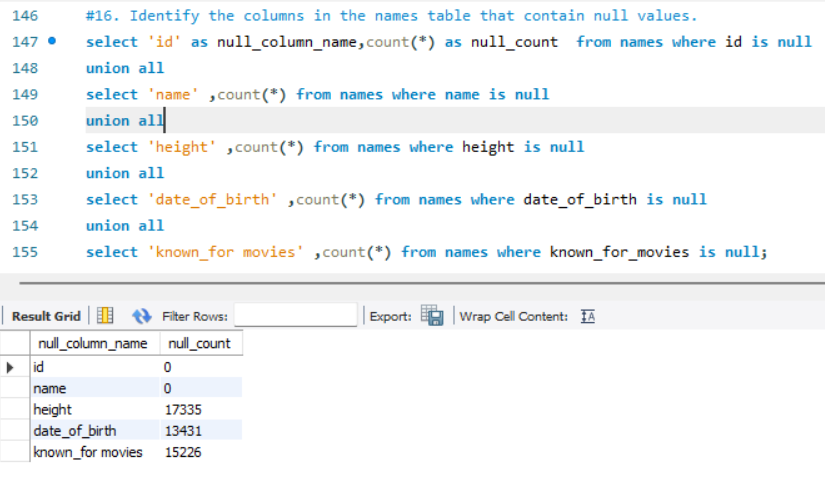
Query explanation:

This SQL query compares the average number of votes received by German and Italian movies. It first filters movies that include "German" in the languages column and calculates their average votes, then does the same for Italian movies. The UNION combines both results into a single output, showing whether German or Italian movies receive more votes on average.

Answer:

The query compares the average number of votes received by German and Italian movies. **German movies have a higher average number of votes (12,928.44) compared to Italian movies (11,960.47).**

16.



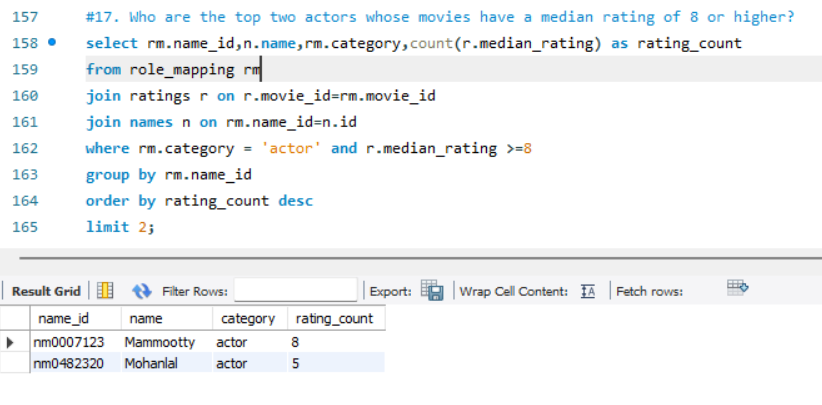
Query explanation:

This SQL query checks which columns in the names table contain NULL values and counts how many NULL values each column has. It does this by selecting each column individually, counting the NULL values using COUNT(\*), and combining the results using UNION ALL. The final output lists the column names along with their respective NULL counts.

Answer:

The query checks which columns in the "names" table contain NULL values. **The columns "height" (17,335 NULLs), "date\_of\_birth" (13,431 NULLs), and "known\_for\_movies" (15,226 NULLs) have missing values, while "id" and "name" have no NULLs.**

17.



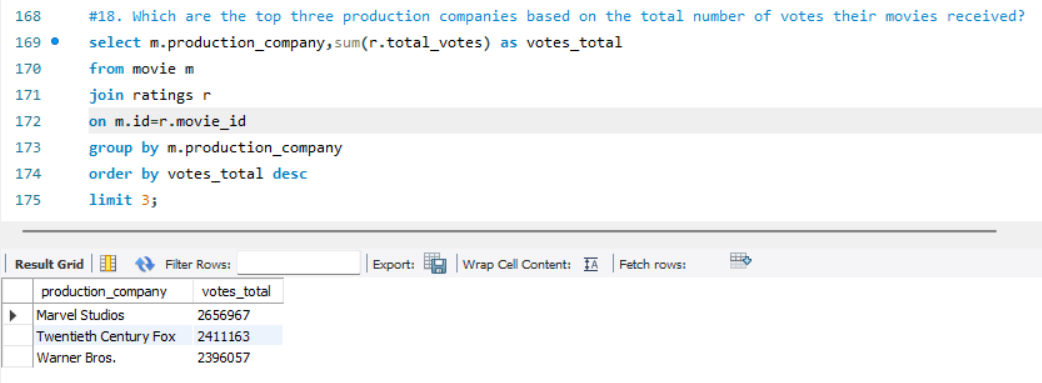
Query explanation:

This SQL query finds the top two actors whose movies have a median rating of 8 or higher. It joins the role\_mapping, ratings, and names tables using movie\_id and name\_id, then filters for actors (category = 'actor') and movies with a median\_rating of at least 8. The results are grouped by actor, counted using COUNT(r.median\_rating), sorted in descending order, and limited to the top two actors.

Answer:

The query finds the top two actors whose movies have a median rating of 8 or higher. **The top two actors are Mammootty (8 movies) and Mohanlal (5 movies).**

18.



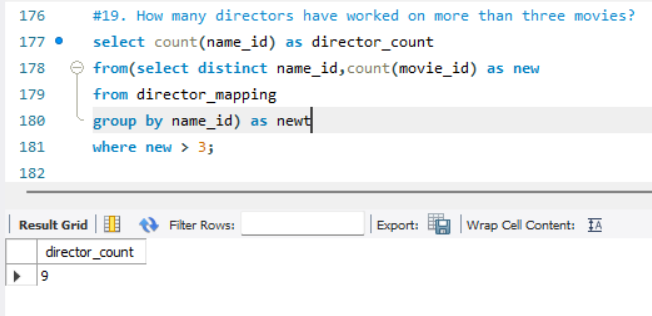
Query explanation:

This SQL query finds the top three production companies whose movies have received the most total votes. It joins the movie and ratings tables using movie\_id, then groups the results by production\_company and sums up the total\_votes for each company using SUM(r.total\_votes). Finally, it sorts the results in descending order and uses LIMIT 3 to show only the top three companies.

Answer:

The query finds the top three production companies based on the total number of votes their movies received. **The top three are Marvel Studios (26,596,67 votes), Twentieth Century Fox (24,111,63 votes), and Warner Bros. (23,960,57 votes).**

19.



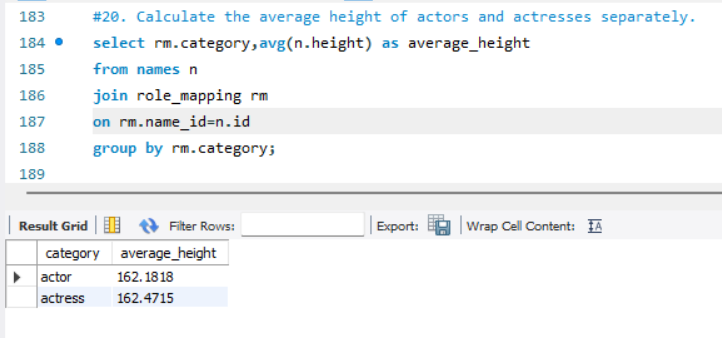
Query explanation:

This SQL query counts how many directors have worked on more than three movies. It first groups the director\_mapping table by name\_id (director ID) and counts the number of movies each director has worked on. Then, it filters out directors with three or fewer movies using WHERE new > 3 and counts the remaining directors using COUNT(name\_id).

Answer:

The query finds the number of directors who have worked on more than three movies. **The answer is 9 directors.**

20.



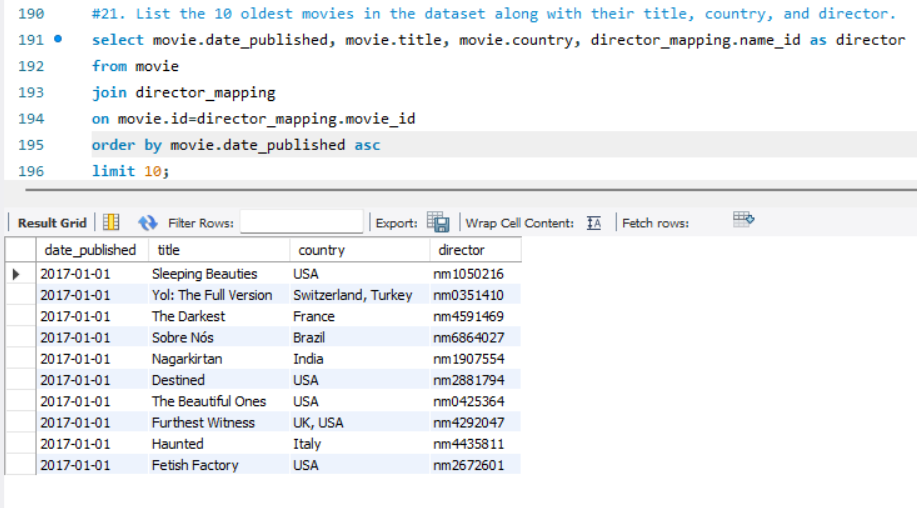
Query explanation:

This SQL query calculates the average height of actors and actresses separately. It joins the names and role\_mapping tables using name\_id, then groups the results by category (actor or actress) and calculates the average height using AVG(n.height). The final output shows the average height for both categories.

Answer:

The query calculates the average height of actors and actresses separately. **Actors have an average height of 162.1818 cm, and actresses have an average height of 162.4715 cm.**

21.



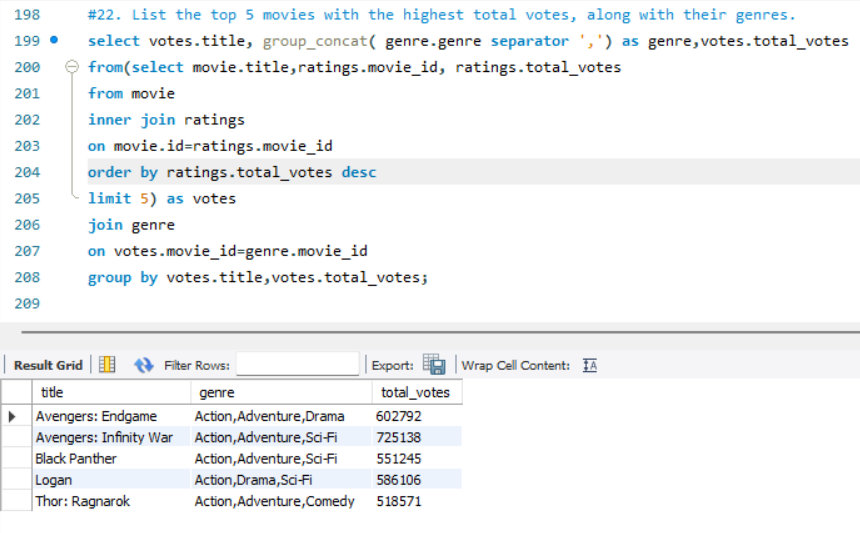
Query explanation:

This SQL query retrieves the 10 oldest movies in the dataset, displaying their release date, title, country, and director. It joins the movie and director\_mapping tables using movie\_id to link movies with their directors. The results are sorted in ascending order by date\_published using ORDER BY movie.date\_published ASC and limited to 10 using LIMIT 10.

Answer:

The query retrieves the **10 oldest movies** in the dataset, listing their **title, country, and director ID**. However, all retrieved movies have the same **release date of 2017-01-01**

22.



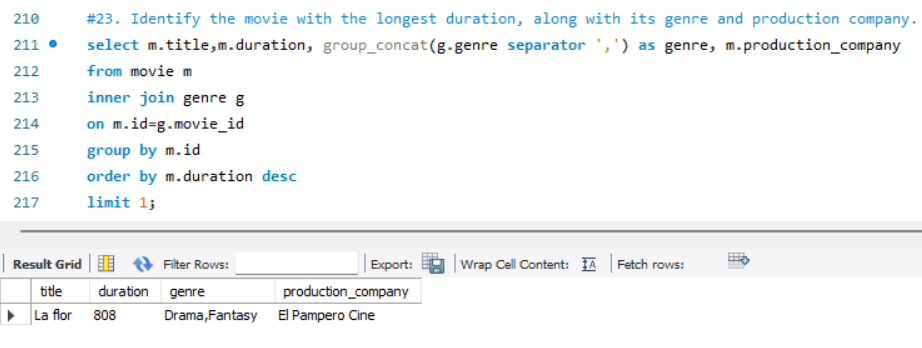
Query explanation:

This SQL query finds the top 5 movies with the highest total votes and their respective genres. It first selects the top 5 movies by joining the movie and ratings tables, sorting by total\_votes in descending order. Then, it joins this result with the genre table and uses GROUP\_CONCAT() to list multiple genres for each movie in a single row, grouping by movie title and total votes.

Answer;

The query finds the **top 5 movies** with the **highest total votes** and their respective **genres**. The top movie is **"Avengers: Infinity War"** with **725,138 votes**, followed by **"Logan"**, **"Black Panther"**, **"Avengers: Endgame"**, and **"Thor: Ragnarok**

23.



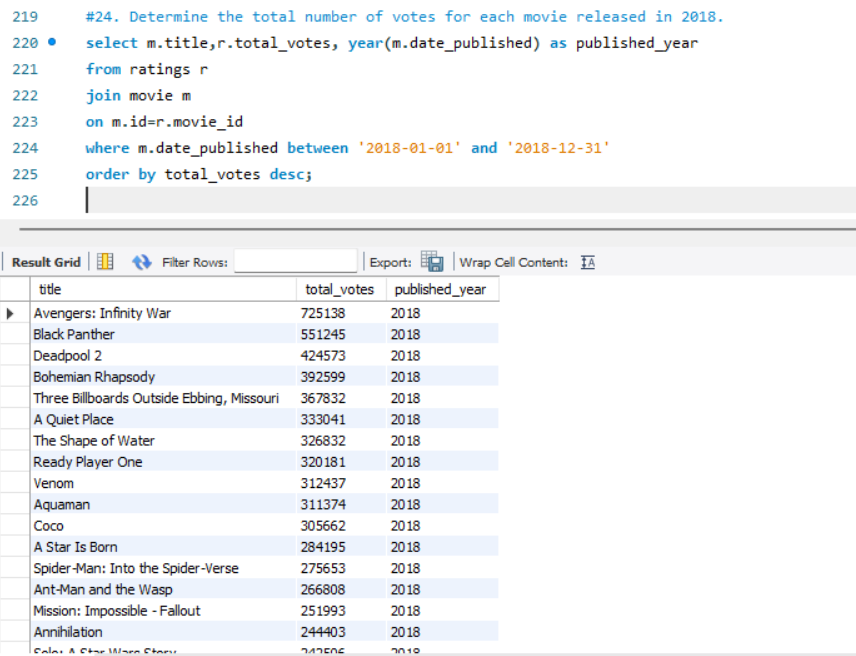
Query explanation:

This SQL query finds the longest movie in the dataset along with its genre and production company. It joins the movie and genre tables using movie\_id, then groups by movie.id and uses GROUP\_CONCAT() to list multiple genres in a single row. The results are sorted in descending order by duration, and LIMIT 1 ensures only the longest movie is displayed.

Answer:

The query finds the **movie with the longest duration**. The result shows that **"La Flor"** is the longest movie, with a **duration of 808 minutes**, categorized as **Drama, Fantasy**, and produced by **El Pampero Cine**.

24.



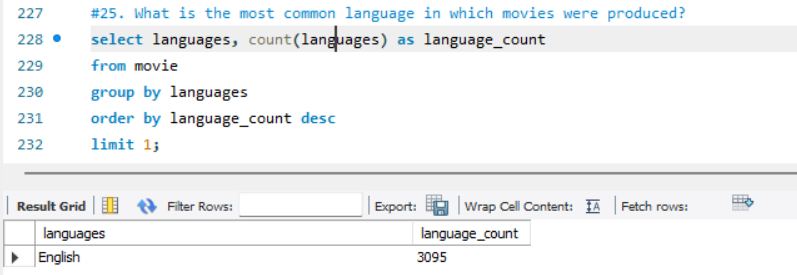
Query explanation:

This SQL query retrieves the total number of votes for each movie released in 2018. It joins the ratings and movie tables using movie\_id, filters movies with a date\_published between January 1 and December 31, 2018, and extracts the year using YEAR(m.date\_published). Finally, the results are sorted in descending order by total\_votes to show the most voted movies first.

Answer:

The query finds the total number of votes for movies released in **2018**. The movie with the most votes is **"Avengers: Infinity War"** with **725,138 votes**, followed by **"Black Panther"** with **551,245 votes**.

25.



Query explanation:

This SQL query identifies the most common language in which movies were produced. It groups the movie table by the languages column, counts the occurrences of each language using COUNT(languages), and sorts the results in descending order by language\_count. The LIMIT 1 ensures that only the most frequently used language is displayed.

Answer:

The query finds the most common language used in movies. The result shows that **English** is the most common language, appearing in **3,095 movies**.