**QUERIES:**

**Advanced SQL – Reinforcement Project – IMDB Dataset**

**1. Count the total number of records in each table of the database.**

**SELECT 'movie' AS table\_name, COUNT(\*) AS total\_records FROM movie**

**UNION ALL**

**SELECT 'genre', COUNT(\*) FROM genre**

**UNION ALL**

**SELECT 'director\_mapping', COUNT(\*) FROM director\_mapping**

**UNION ALL**

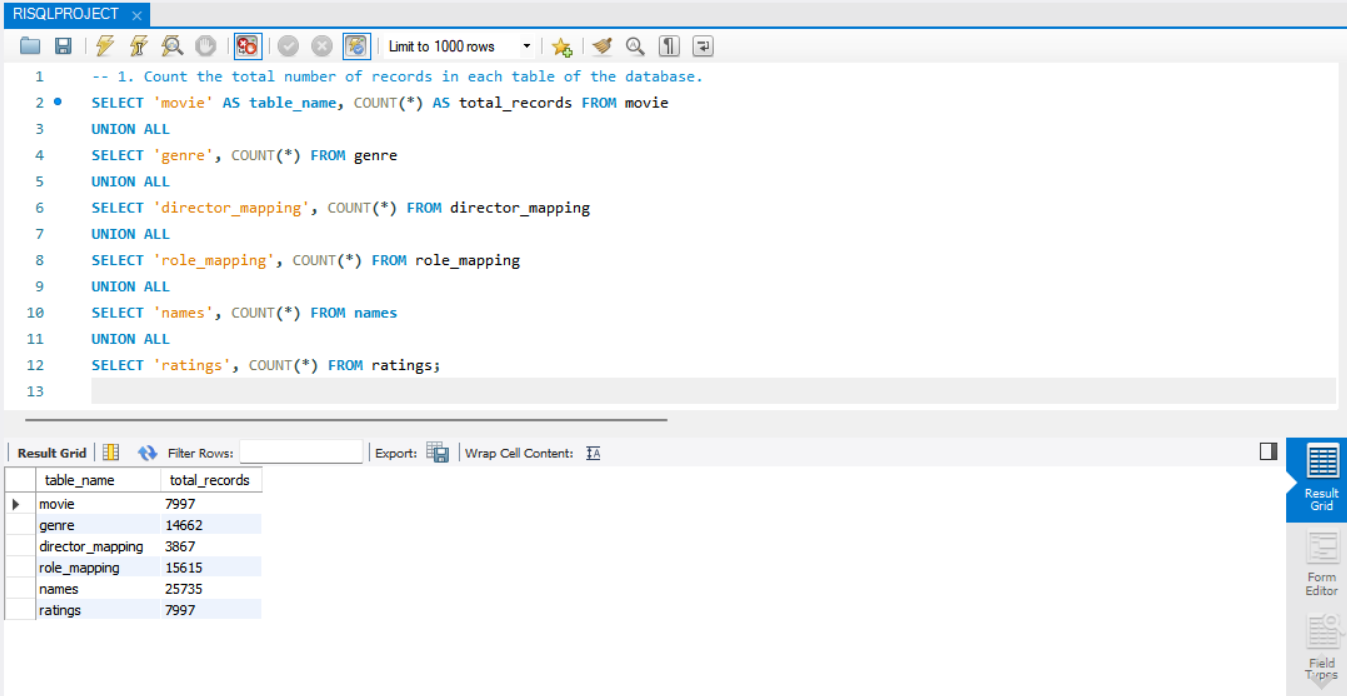
**SELECT 'role\_mapping', COUNT(\*) FROM role\_mapping**

**UNION ALL**

**SELECT 'names', COUNT(\*) FROM names**

**UNION ALL**

**SELECT 'ratings', COUNT(\*) FROM ratings;**

****

OUTPUT:

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

**SELECT**

**'title' AS column\_name, COUNT(\*) AS null\_count**

**FROM movie WHERE title IS NULL**

**UNION ALL**

**SELECT 'year', COUNT(\*) FROM movie WHERE year IS NULL**

**UNION ALL**

**SELECT 'date\_published', COUNT(\*) FROM movie WHERE date\_published IS NULL**

**UNION ALL**

**SELECT 'duration', COUNT(\*) FROM movie WHERE duration IS NULL**

**UNION ALL**

**SELECT 'country', COUNT(\*) FROM movie WHERE country IS NULL**

**UNION ALL**

**SELECT 'worlwide\_gross\_income', COUNT(\*) FROM movie WHERE worlwide\_gross\_income IS NULL**

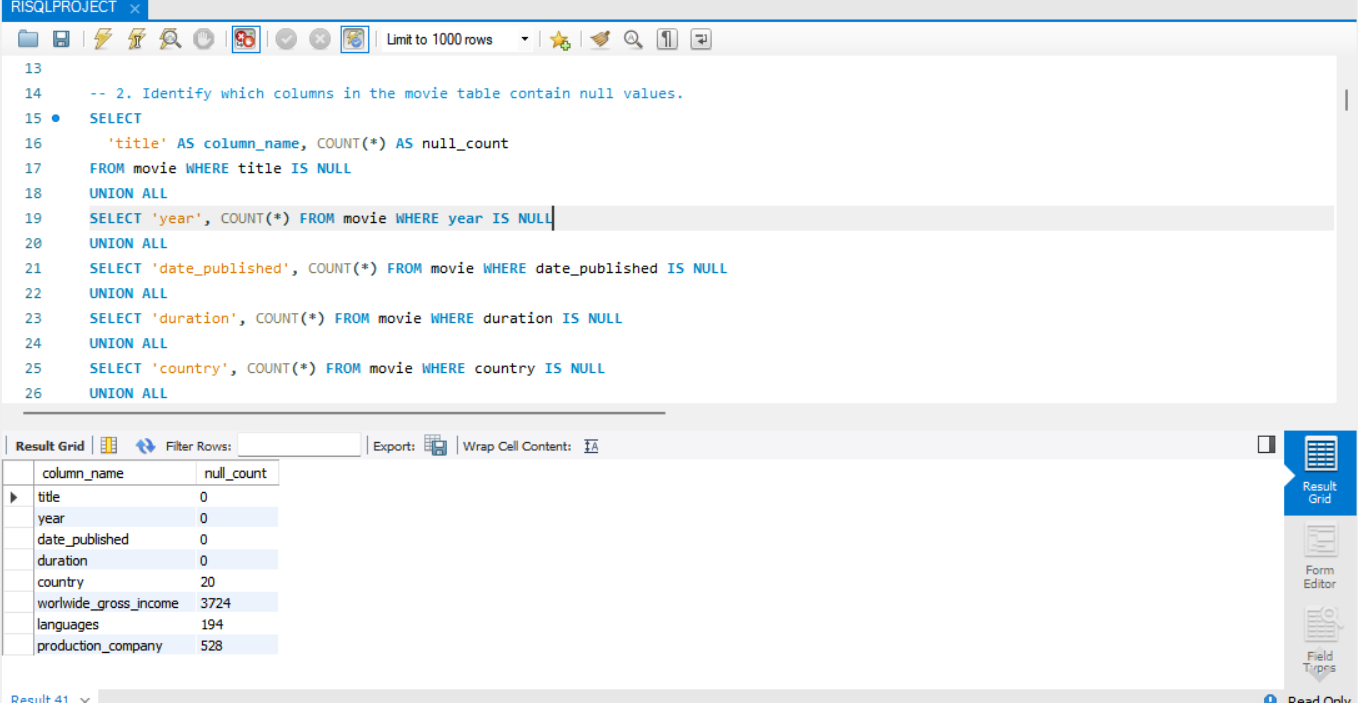
**UNION ALL**

**SELECT 'languages', COUNT(\*) FROM movie WHERE languages IS NULL**

**UNION ALL**

**SELECT 'production\_company', COUNT(\*) FROM movie WHERE production\_company IS NULL;**

OUTPUT:

****

**3. Determine the total number of movies released each year, and analyze how the trend changes month wise.**

**SELECT**

**YEAR(date\_published) AS release\_year,**

**COUNT(\*) AS total\_movies**

**FROM**

**movie**

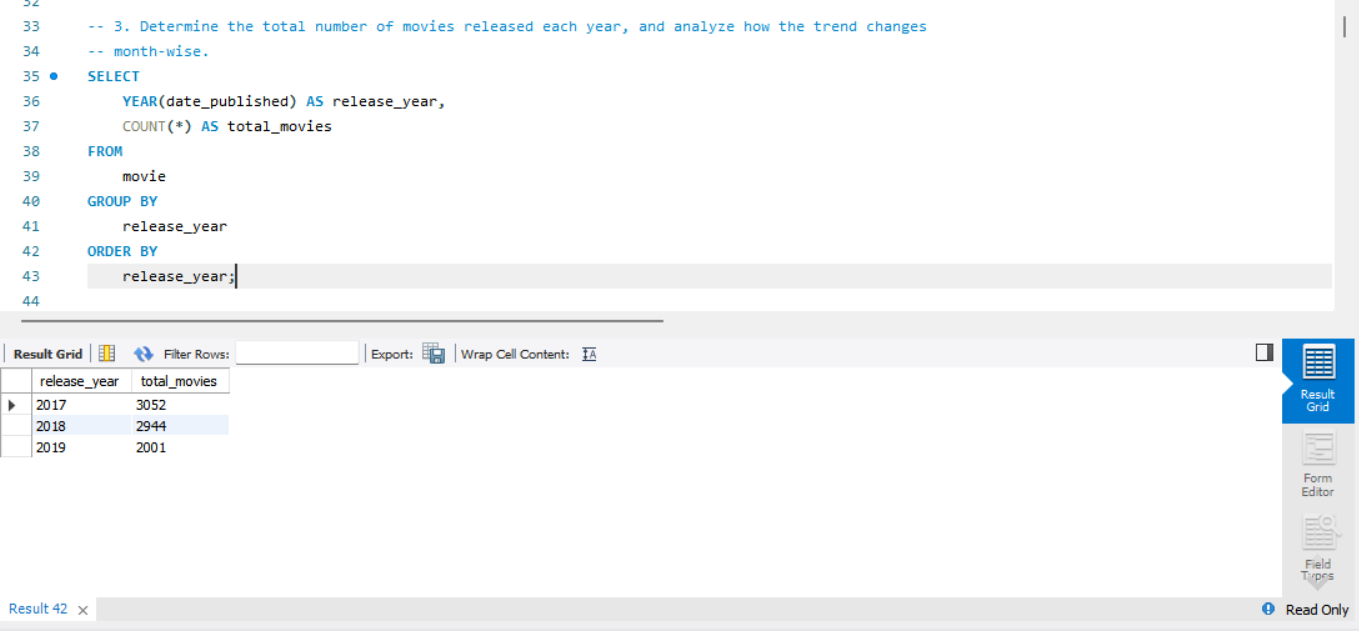
**GROUP BY**

**release\_year**

**ORDER BY**

**release\_year;**

OUTPUT:

****

**4. How many movies were produced in either the USA or India in the year 2019?**

**SELECT**

**COUNT(\*) AS total\_movies**

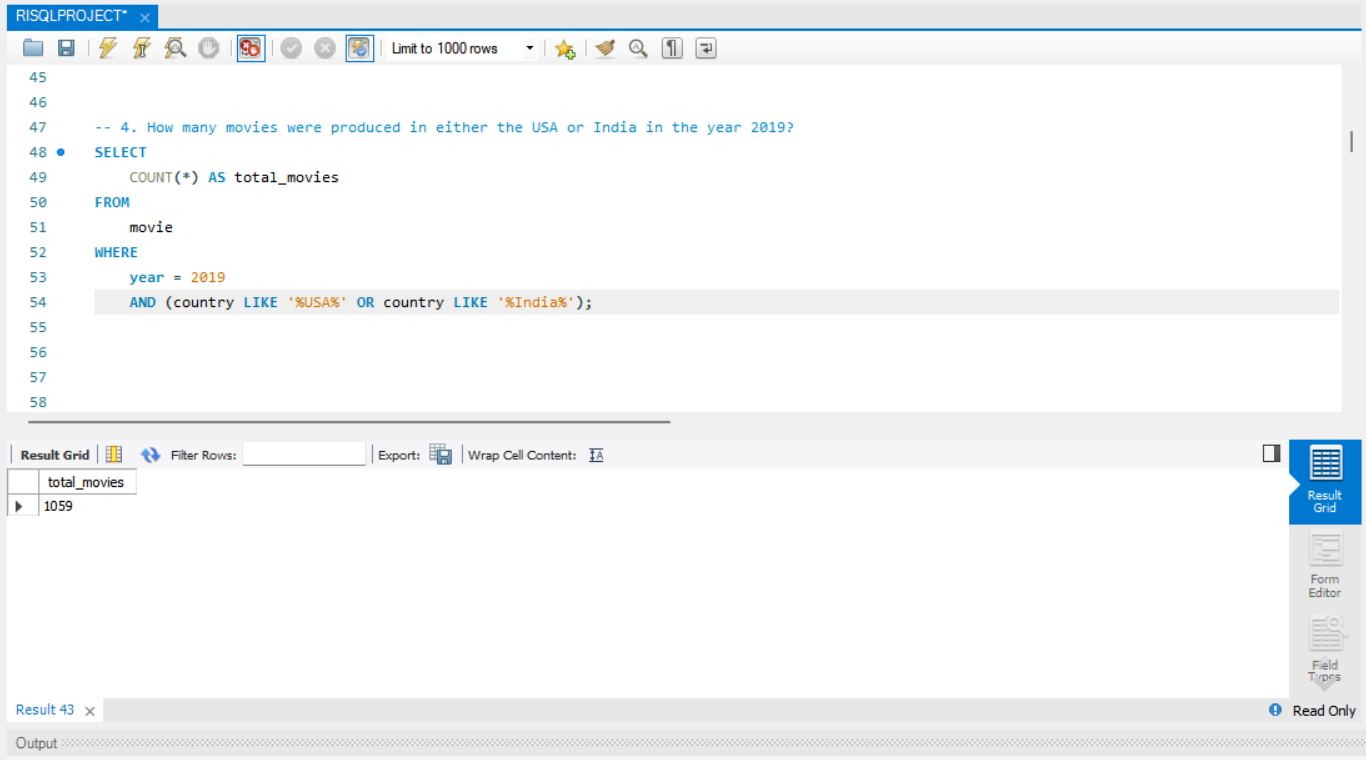
**FROM**

**movie**

**WHERE**

**year = 2019**

**AND (country LIKE '%USA%' OR country LIKE '%India%');**

****

OUTPUT:

**5. List the unique genres in the dataset, and count how many movies belong exclusively to on genre.**

**SELECT DISTINCT genre**

**FROM genre**

**ORDER BY genre;**

**SELECT**

**COUNT(\*) AS single\_genre\_movie\_count**

**FROM (**

**SELECT**

**movie\_id**

**FROM**

**genre**

**GROUP BY**

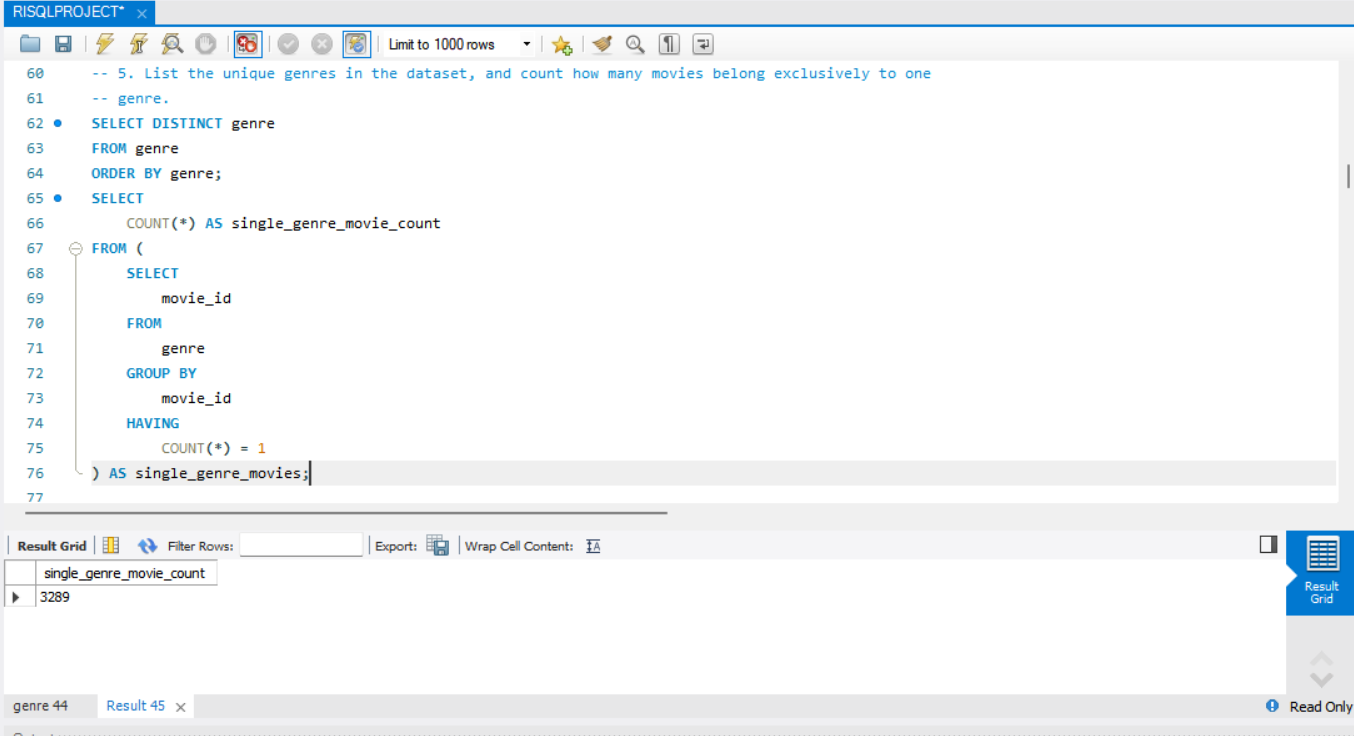
**movie\_id**

**HAVING**

**COUNT(\*) = 1**

**) AS single\_genre\_movies;**

OUTPUT:

****

**6. Which genre has the highest total number of movies produced?**

**SELECT**

**genre,**

**COUNT(\*) AS total\_movies**

**FROM**

**genre**

**GROUP BY**

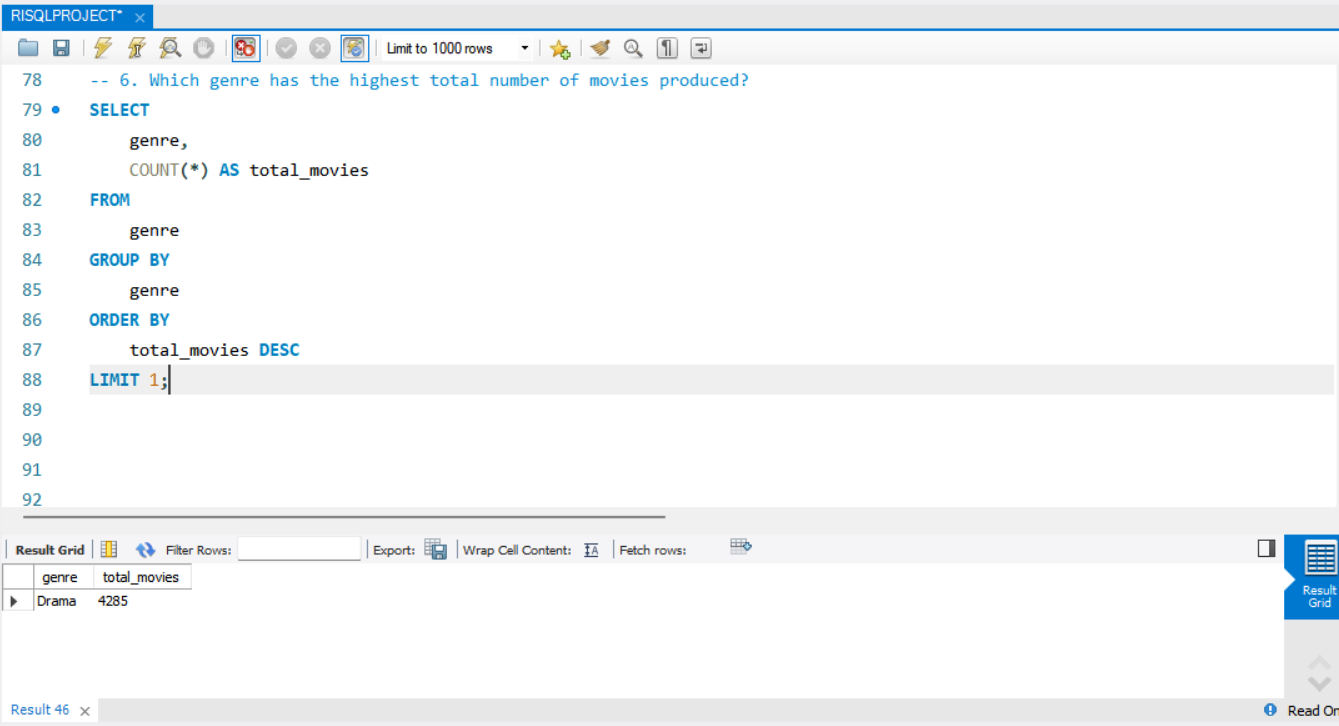
**genre**

**ORDER BY**

**total\_movies DESC**

**LIMIT 1;**

OUTPUT:



**7. Calculate the average movie duration for each genre.**

**SELECT**

**g.genre,**

**ROUND(AVG(m.duration), 2) AS average\_duration**

**FROM**

**genre g**

**JOIN**

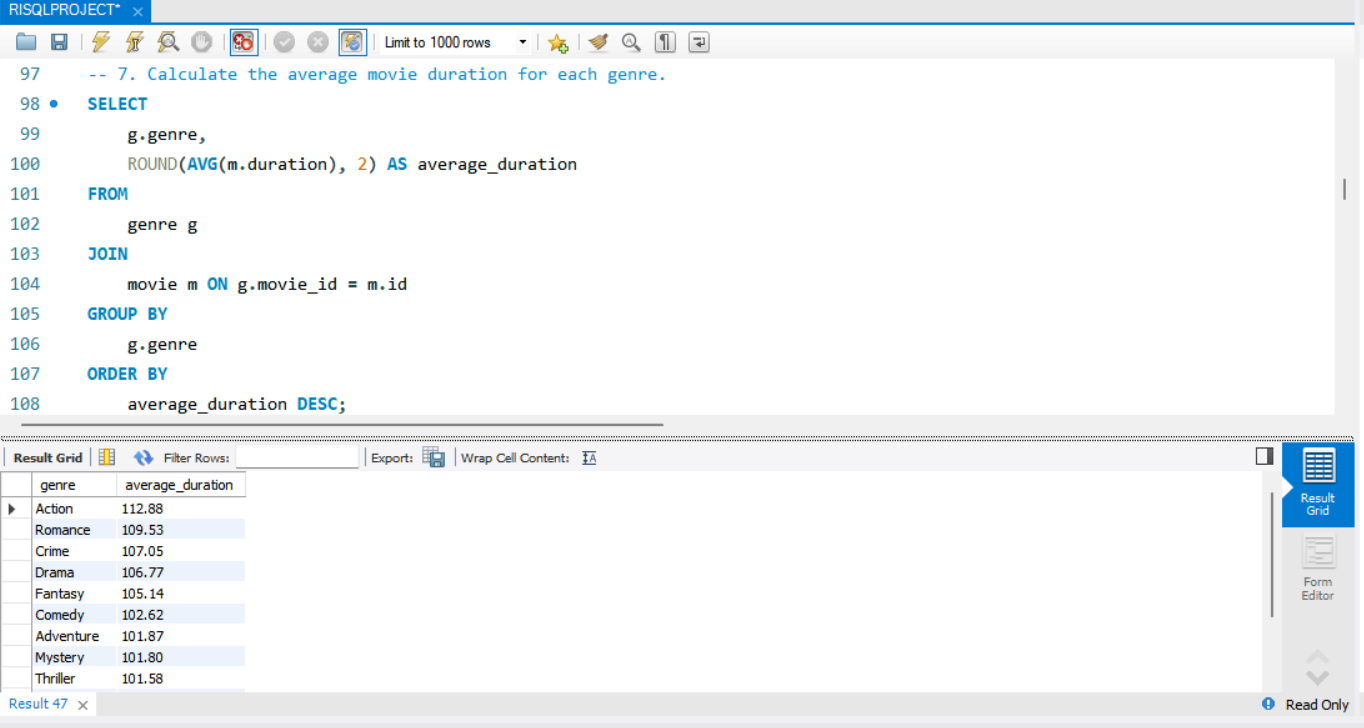
**movie m ON g.movie\_id = m.id**

**GROUP BY**

**g.genre**

**ORDER BY**

**average\_duration DESC;**

****

OUTPUT:

**8. Identify actors or actresses who have appeared in more than three movies with an average rating below 5.**

**SELECT**

**n.name AS actor\_name,**

**COUNT(\*) AS low\_rated\_movies**

**FROM**

**names n**

**JOIN**

**role\_mapping rm ON n.id = rm.name\_id**

**JOIN**

**ratings r ON rm.movie\_id = r.movie\_id**

**WHERE**

**rm.category IN ('actor', 'actress')**

**AND r.avg\_rating < 5**

**GROUP BY**

**n.name**

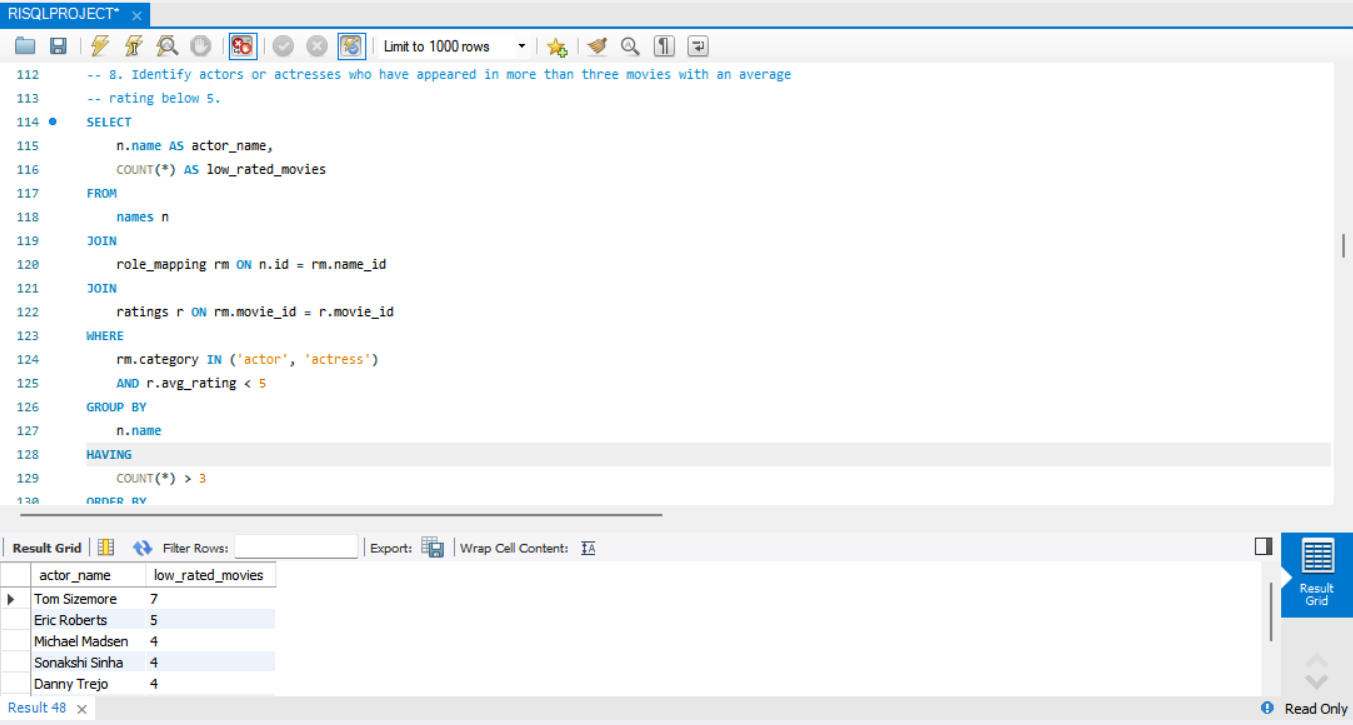
**HAVING**

**COUNT(\*) > 3**

**ORDER BY**

**low\_rated\_movies DESC;**

OUTPUT:

****

**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,**

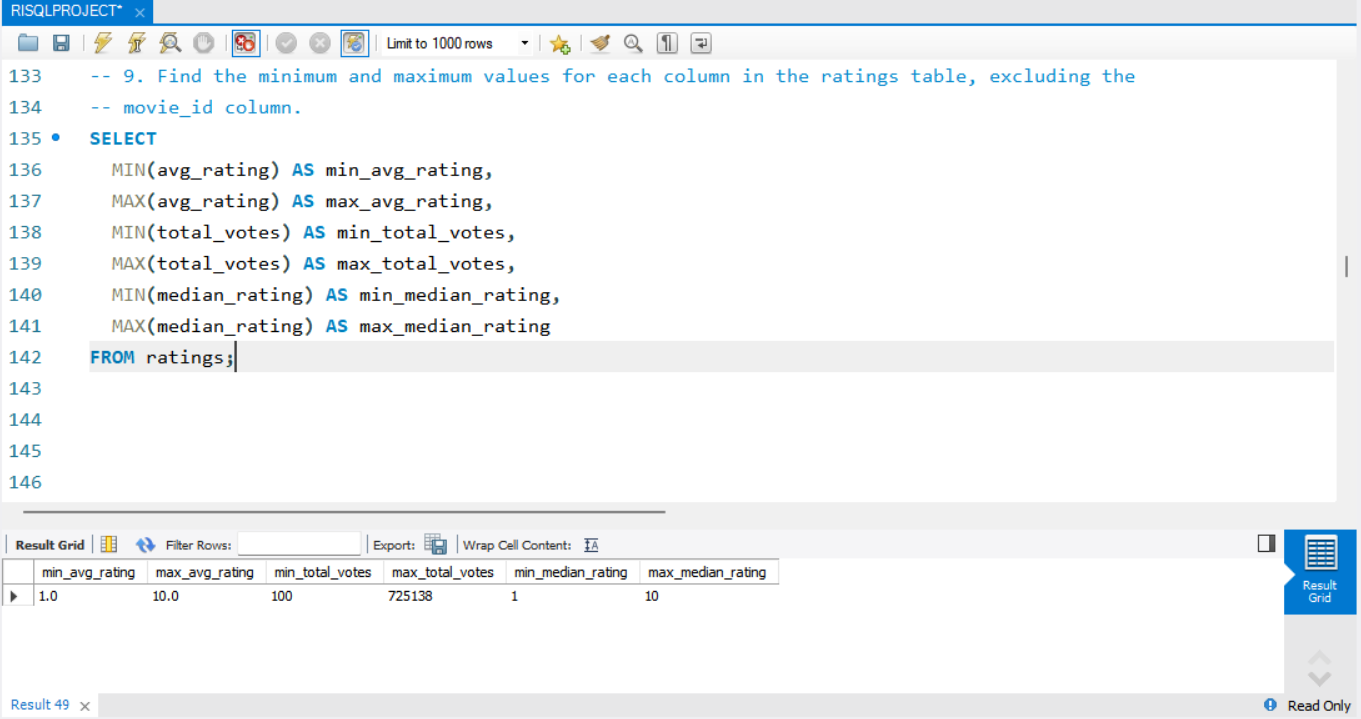
**MAX(total\_votes) AS max\_total\_votes,**

**MIN(median\_rating) AS min\_median\_rating,**

**MAX(median\_rating) AS max\_median\_rating**

**FROM ratings;**

OUTPUT:

****

**10. Which are the top 10 movies based on their average rating?**

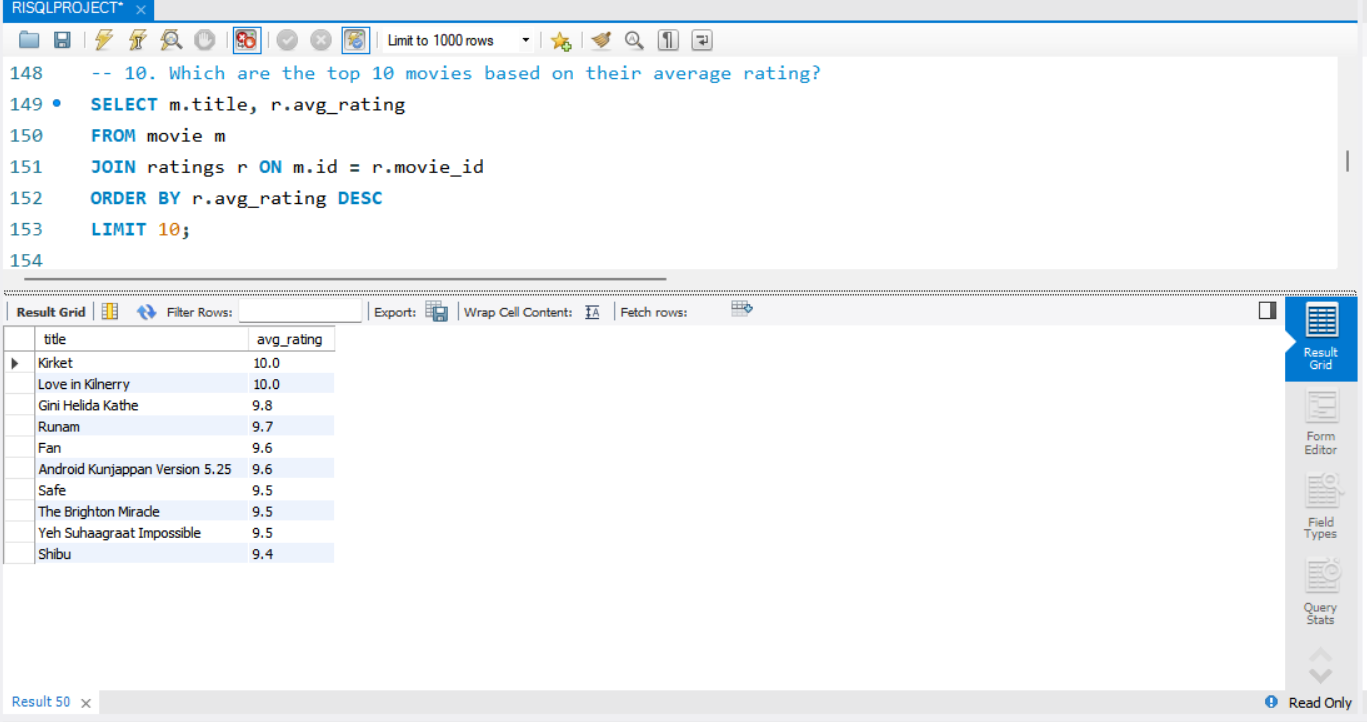
**SELECT m.title, r.avg\_rating**

**FROM movie m**

**JOIN ratings r ON m.id = r.movie\_id**

**ORDER BY r.avg\_rating DESC**

**LIMIT 10;**

****

OUTPUT:

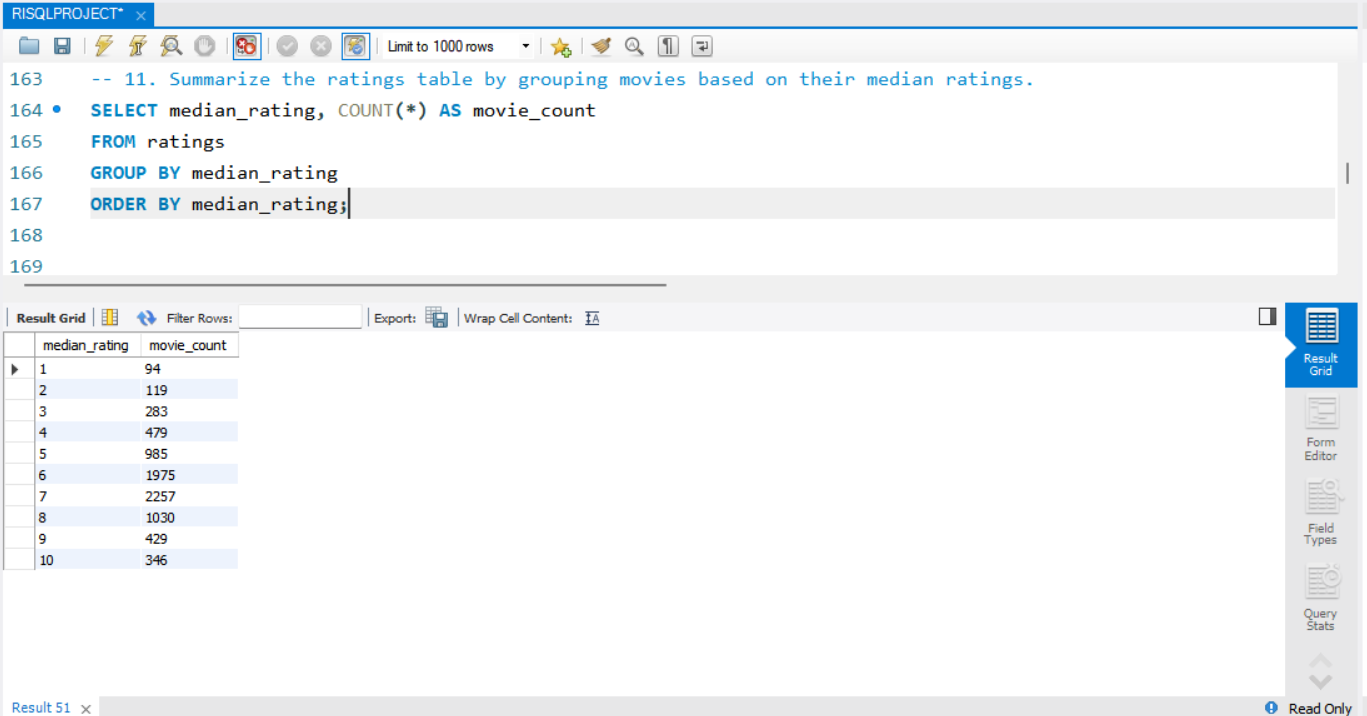
**11. Summarize the ratings table by grouping movies based on their median ratings.**

**SELECT median\_rating, COUNT(\*) AS movie\_count**

**FROM ratings**

**GROUP BY median\_rating**

**ORDER BY median\_rating;**

****

OUTPUT:

**12. How many movies, released in March 2017 in the USA within a specific genre, had more**

**than 1,000 votes?**

**SELECT m.title, m.country, g.genre, r.total\_votes**

**FROM movie m**

**JOIN genre g ON m.id = g.movie\_id**

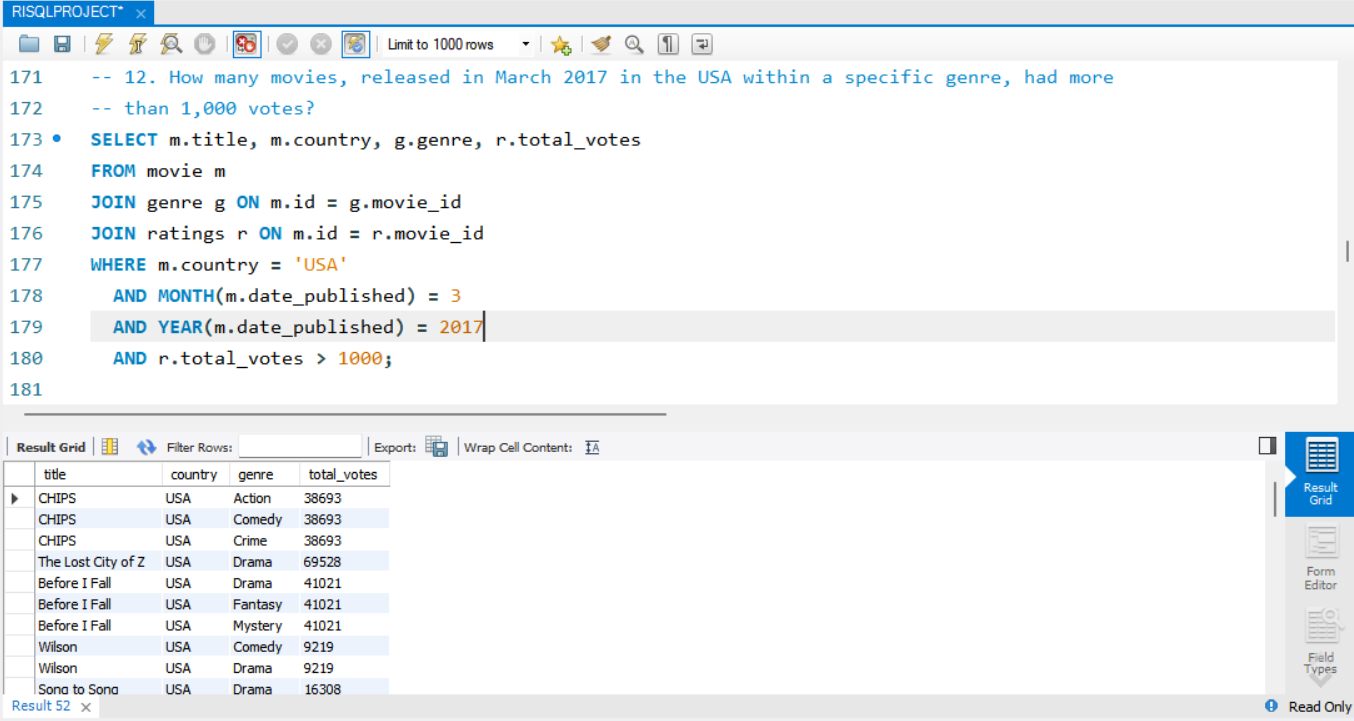
**JOIN ratings r ON m.id = r.movie\_id**

**WHERE m.country = 'USA'**

**AND MONTH(m.date\_published) = 3**

**AND YEAR(m.date\_published) = 2017**

**AND r.total\_votes > 1000;**

****

OUTPUT:

**13. Find movies from each genre that begin with the word “The” and have an average rating**

**greater than 8.**

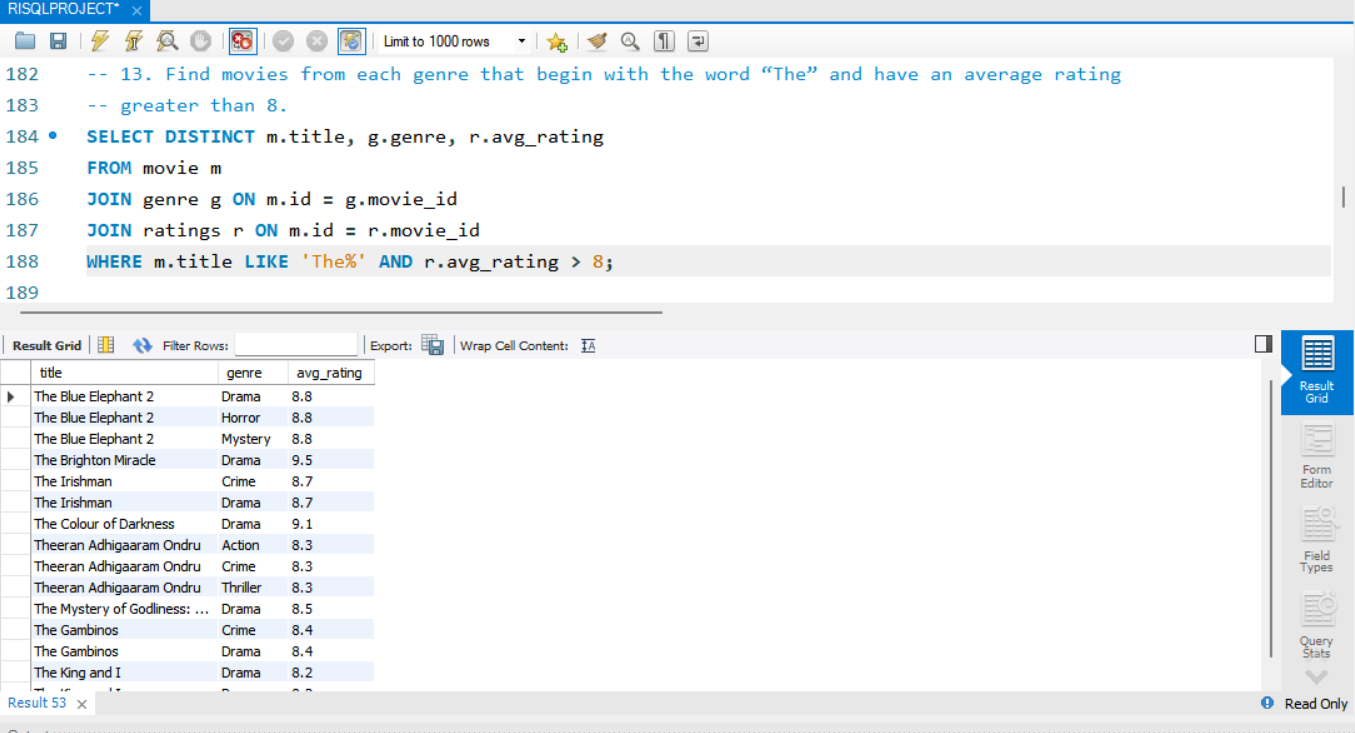
**SELECT DISTINCT m.title, g.genre, r.avg\_rating**

**FROM movie m**

**JOIN genre g ON m.id = g.movie\_id**

**JOIN ratings r ON m.id = r.movie\_id**

**WHERE m.title LIKE 'The%' AND r.avg\_rating > 8;**

****

OUTPUT:

**14. Of the movies released between April 1, 2018, and April 1, 2019, how many received a median rating of 8?**

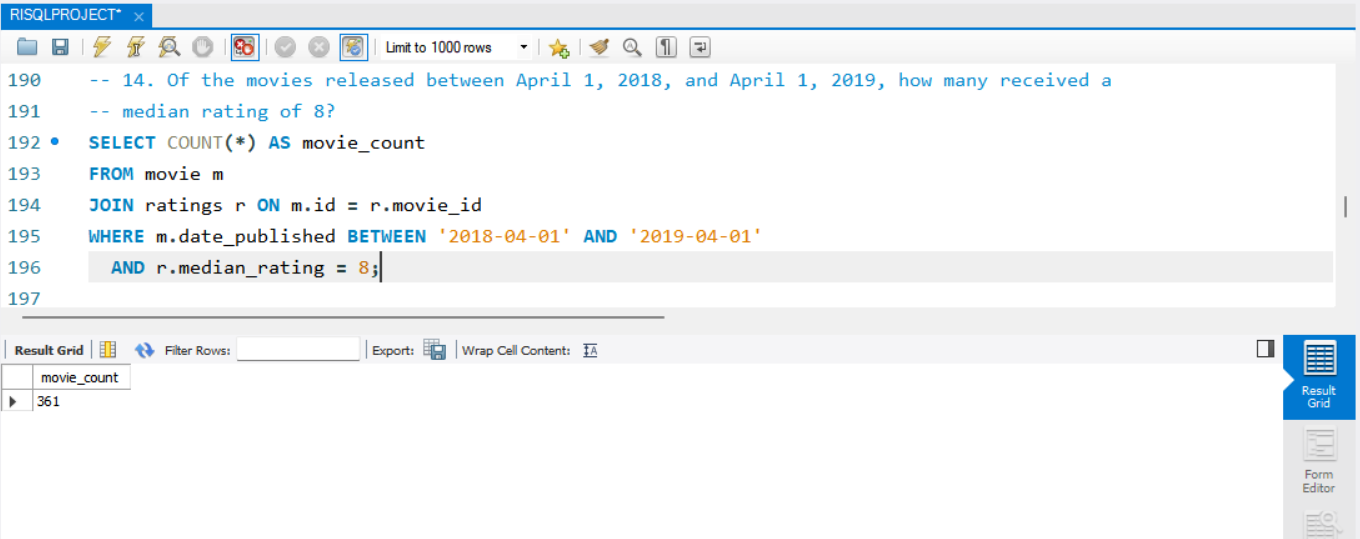
**SELECT COUNT(\*) AS movie\_count**

**FROM movie m**

**JOIN ratings r ON m.id = r.movie\_id**

**WHERE m.date\_published BETWEEN '2018-04-01' AND '2019-04-01'**

**AND r.median\_rating = 8;**

****

OUTPUT:

**15. Do German movies receive more votes on average than Italian movies?**

**SELECT country, AVG(r.total\_votes) AS avg\_votes**

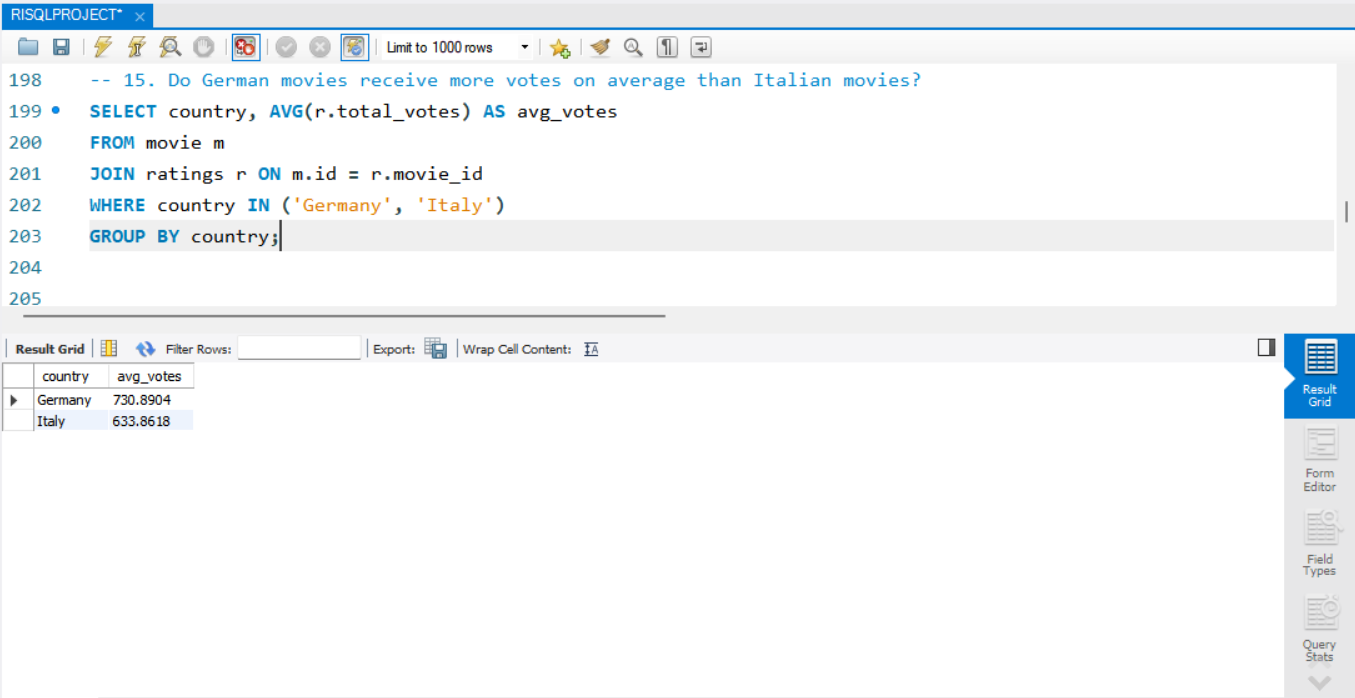
**FROM movie m**

**JOIN ratings r ON m.id = r.movie\_id**

**WHERE country IN ('Germany', 'Italy')**

**GROUP BY country;**

OUTPUT:

****

**16. Identify the columns in the names table that contain null values.**

**SELECT**

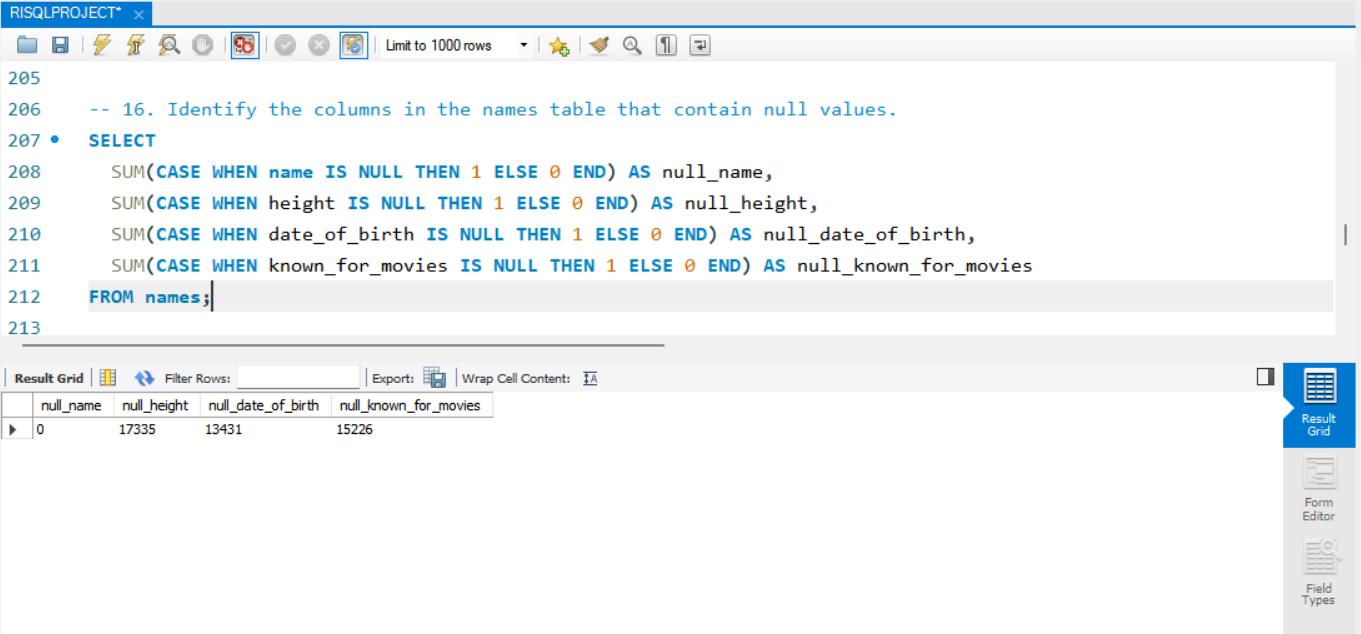
**SUM(CASE WHEN name IS NULL THEN 1 ELSE 0 END) AS null\_name,**

**SUM(CASE WHEN height IS NULL THEN 1 ELSE 0 END) AS null\_height,**

**SUM(CASE WHEN date\_of\_birth IS NULL THEN 1 ELSE 0 END) AS null\_date\_of\_birth,**

**SUM(CASE WHEN known\_for\_movies IS NULL THEN 1 ELSE 0 END) AS null\_known\_for\_movies**

**FROM names;**

****

OUTPUT:

**17. Who are the top two actors whose movies have a median rating of 8 or higher?**

**SELECT n.name, COUNT(\*) AS movie\_count**

**FROM names n**

**JOIN role\_mapping rm ON n.id = rm.name\_id**

**JOIN ratings r ON rm.movie\_id = r.movie\_id**

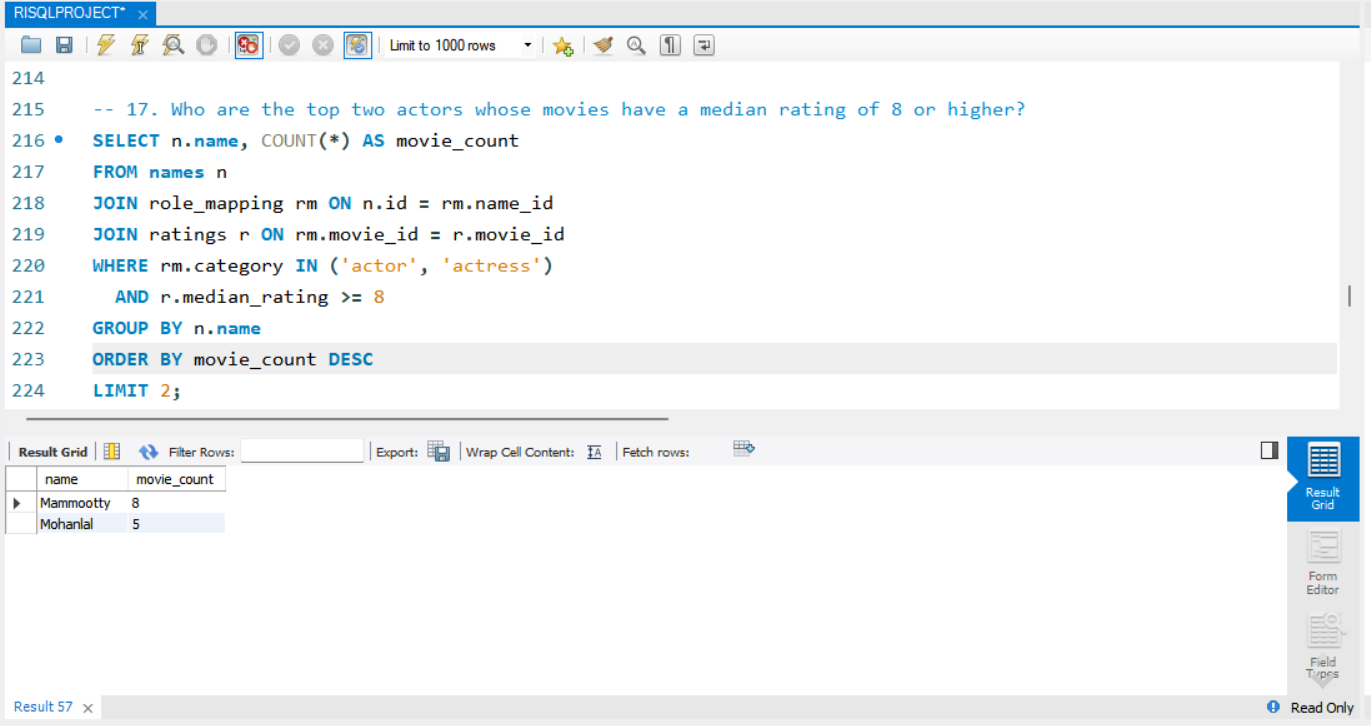
**WHERE rm.category IN ('actor', 'actress')**

**AND r.median\_rating >= 8**

**GROUP BY n.name**

**ORDER BY movie\_count DESC**

**LIMIT 2;**

****

OUTPUT:

**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**

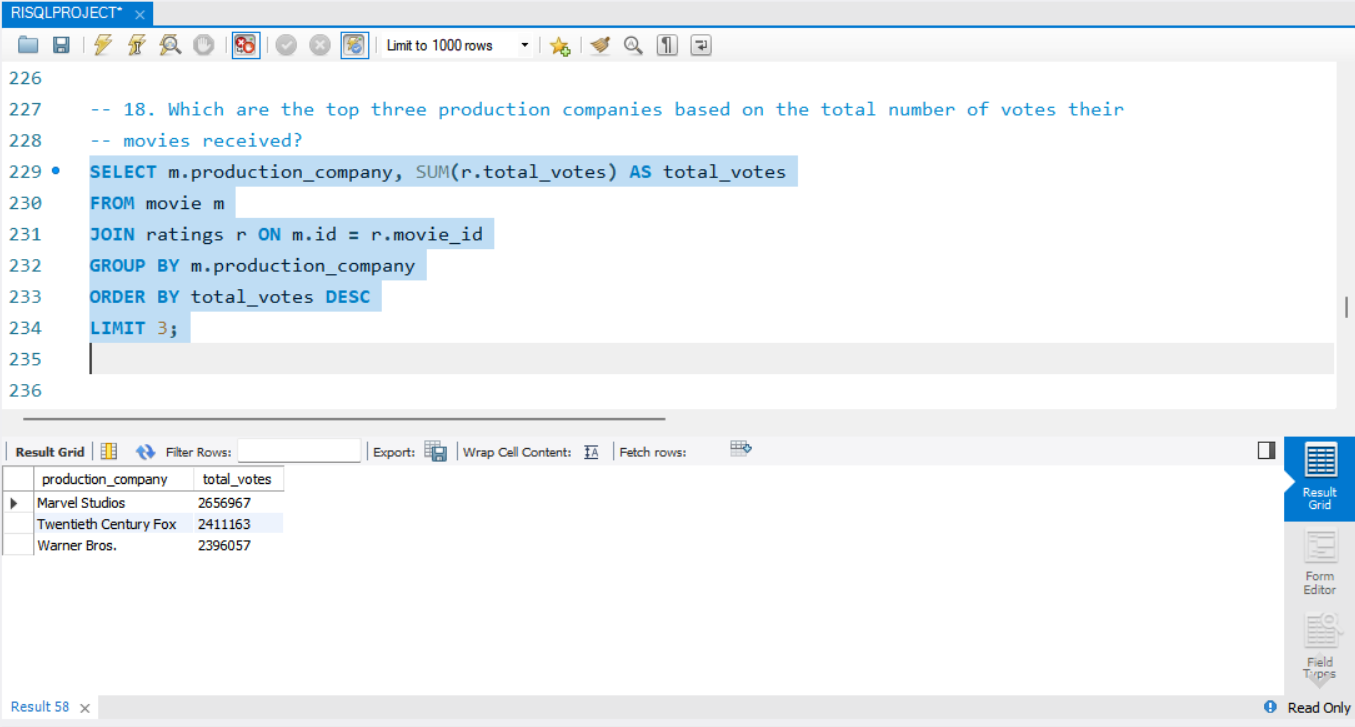
**FROM movie m**

**JOIN ratings r ON m.id = r.movie\_id**

**GROUP BY m.production\_company**

**ORDER BY total\_votes DESC**

**LIMIT 3;**

****

OUTPUT:

**19. How many directors have worked on more than three movies?**

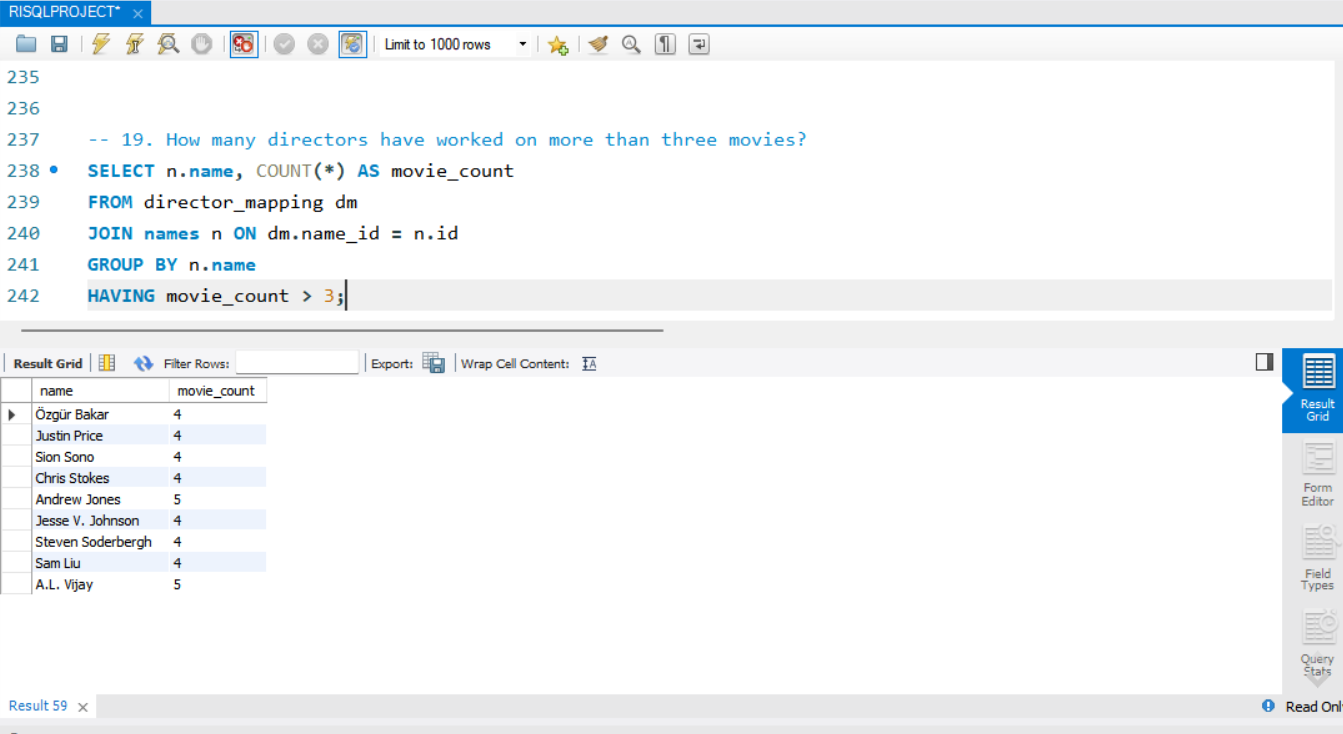
**SELECT n.name, COUNT(\*) AS movie\_count**

**FROM director\_mapping dm**

**JOIN names n ON dm.name\_id = n.id**

**GROUP BY n.name**

**HAVING movie\_count > 3;**

****

OUTPUT:

**20. Calculate the average height of actors and actresses separately.**

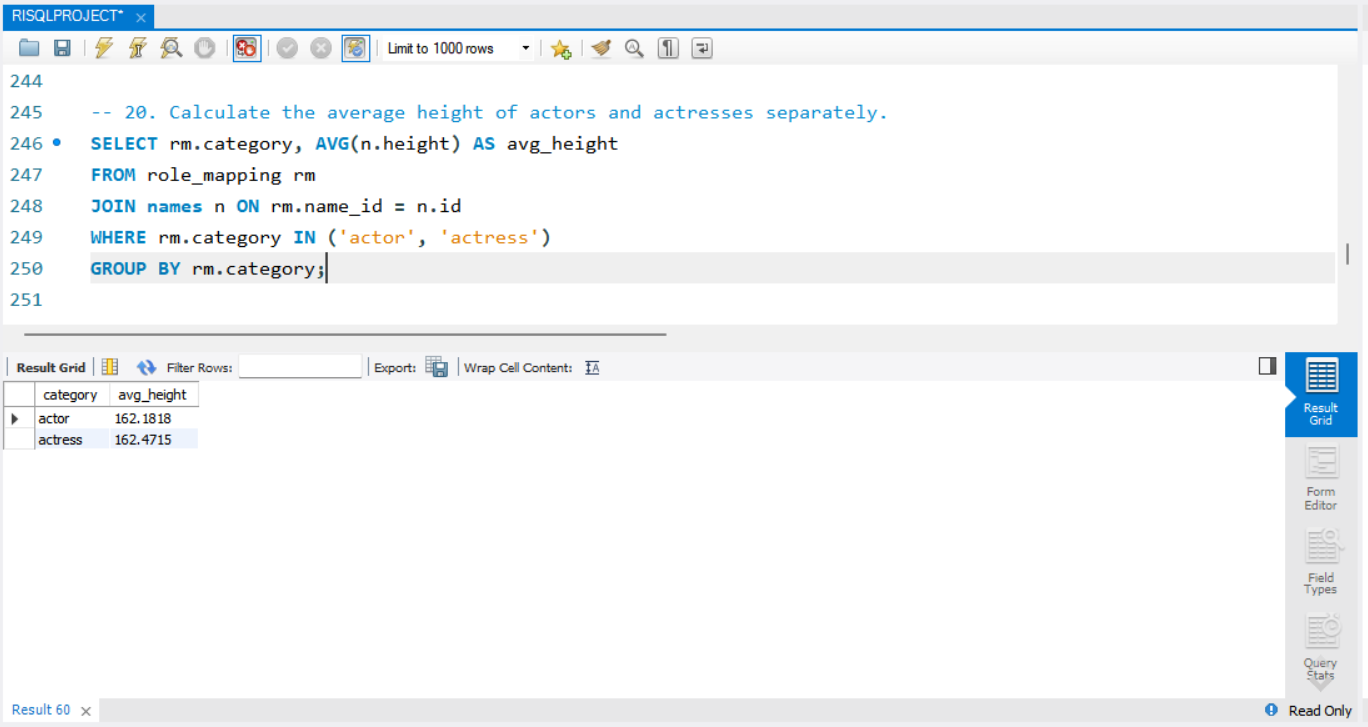
**SELECT rm.category, AVG(n.height) AS avg\_height**

**FROM role\_mapping rm**

**JOIN names n ON rm.name\_id = n.id**

**WHERE rm.category IN ('actor', 'actress')**

**GROUP BY rm.category;**

****

OUTPUT:

**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**

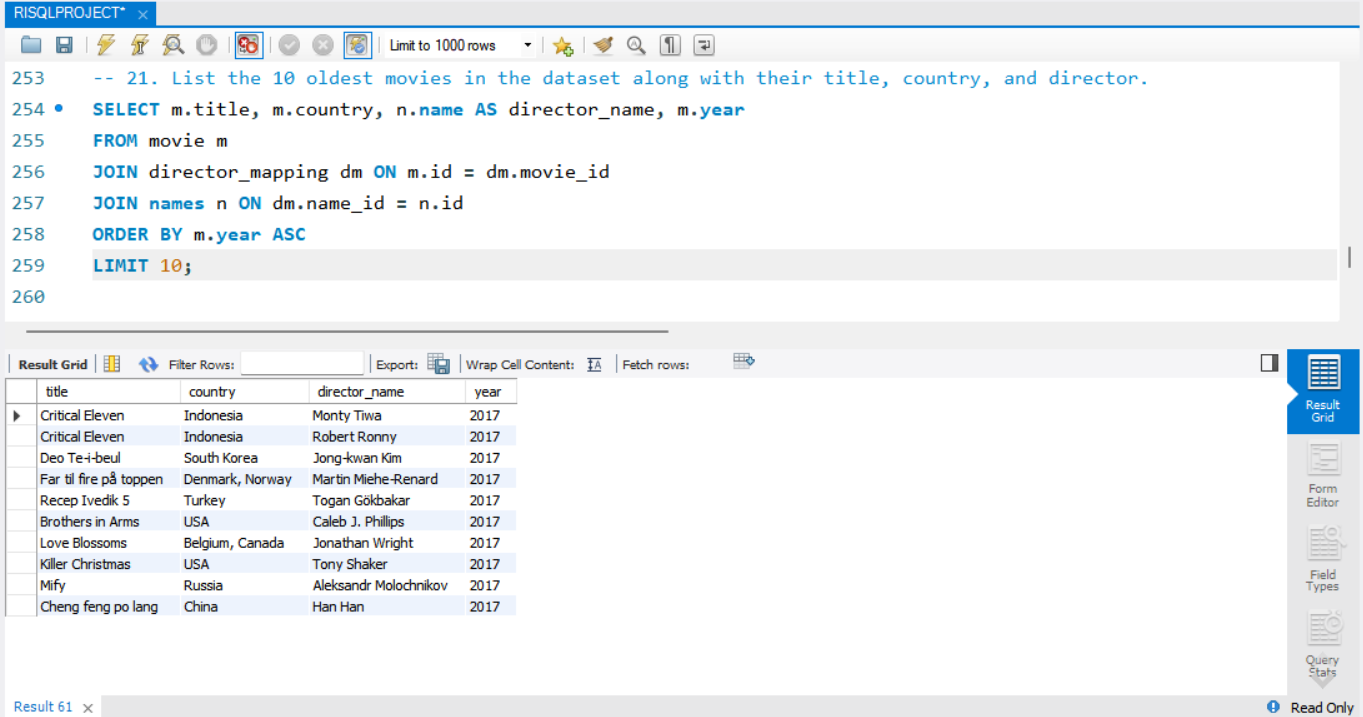
**FROM movie m**

**JOIN director\_mapping dm ON m.id = dm.movie\_id**

**JOIN names n ON dm.name\_id = n.id**

**ORDER BY m.year ASC**

**LIMIT 10;**

****

OUTPUT:

**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) AS genres**

**FROM movie m**

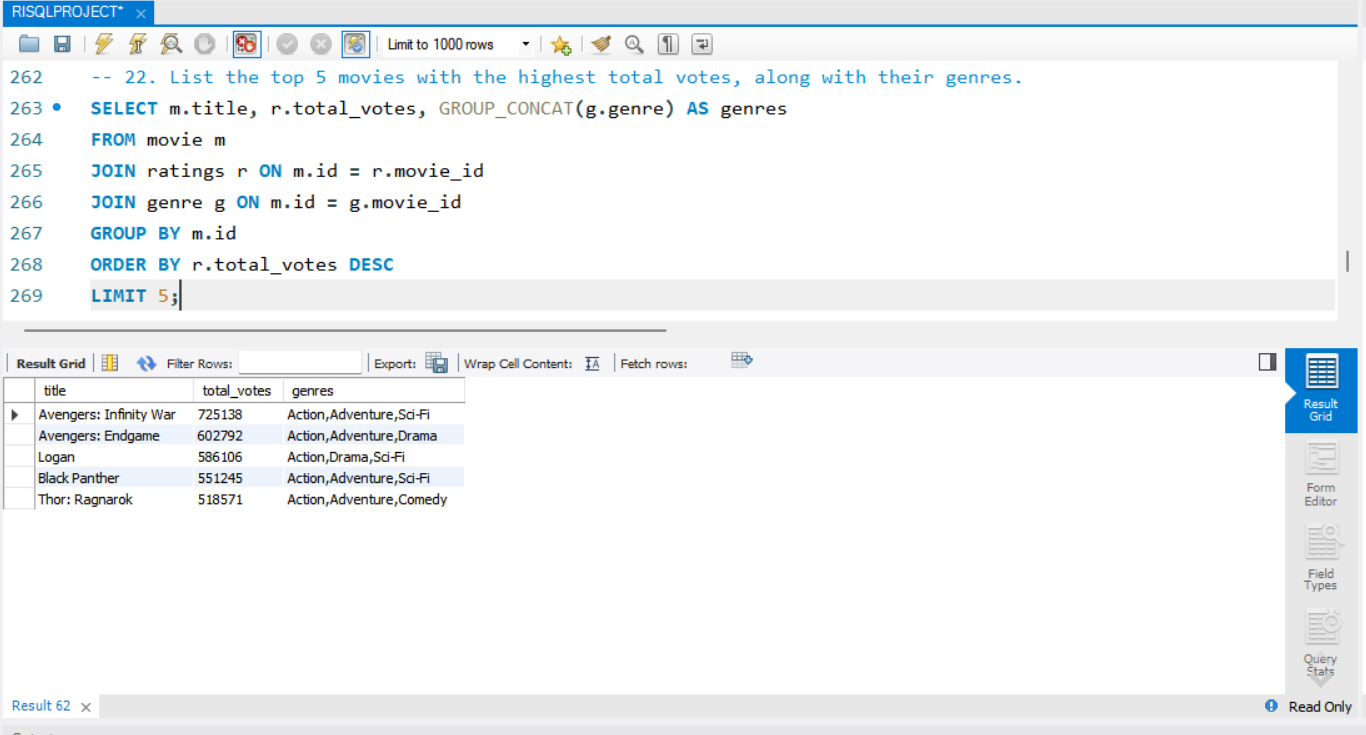
**JOIN ratings r ON m.id = r.movie\_id**

**JOIN genre g ON m.id = g.movie\_id**

**GROUP BY m.id**

**ORDER BY r.total\_votes DESC**

**LIMIT 5;**

****

OUTPUT:

**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) AS genres**

**FROM movie m**

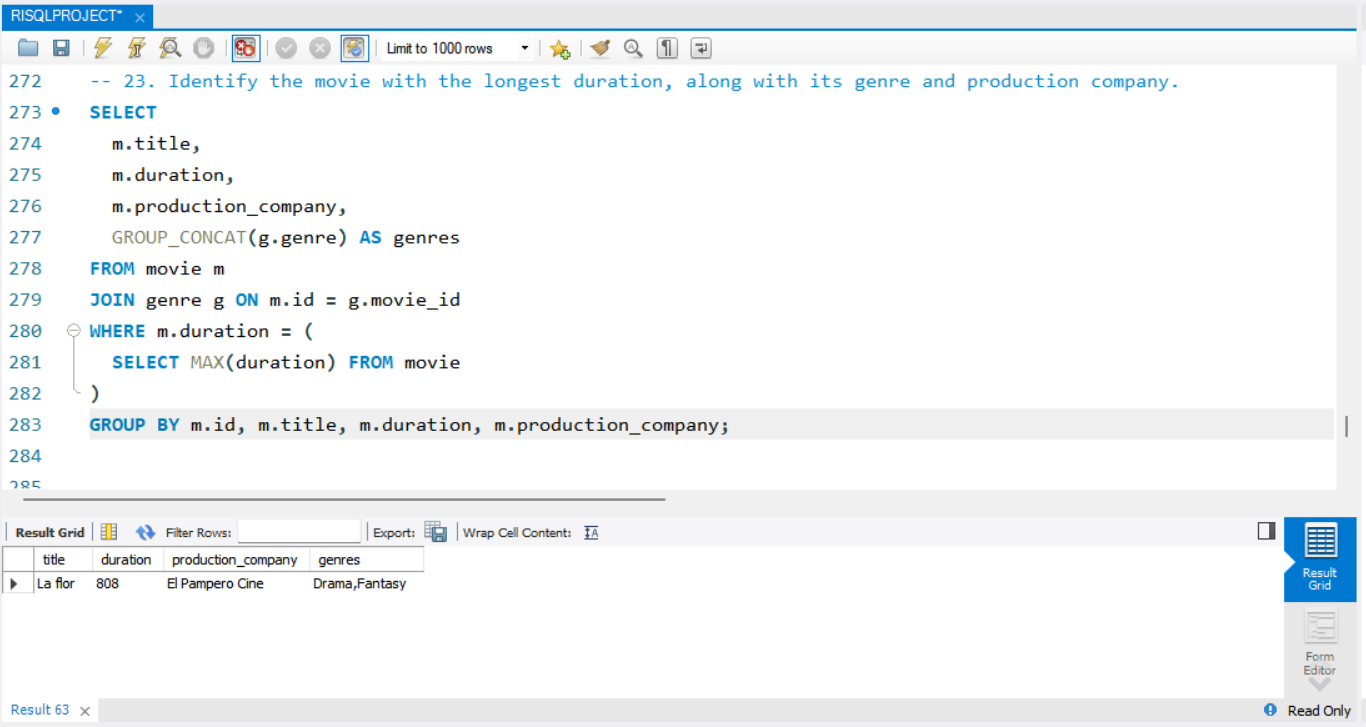
**JOIN genre g ON m.id = g.movie\_id**

**WHERE m.duration = (**

**SELECT MAX(duration) FROM movie**

**)**

**GROUP BY m.id, m.title, m.duration, m.production\_company;**

****

OUTPUT:

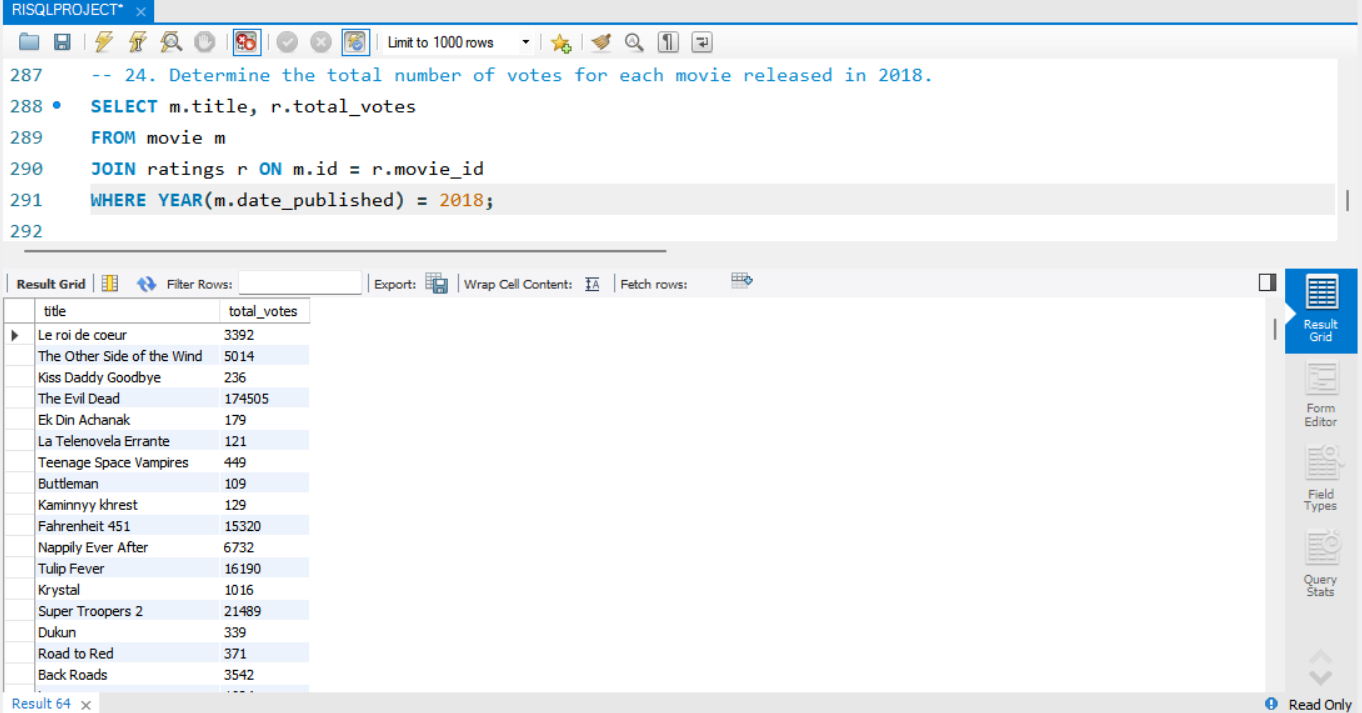
**24. Determine the total number of votes for each movie released in 2018.**

**SELECT m.title, r.total\_votes**

**FROM movie m**

**JOIN ratings r ON m.id = r.movie\_id**

**WHERE YEAR(m.date\_published) = 2018;**

****

OUTPUT:

**25. What is the most common language in which movies were produced?**

**SELECT**

**SUBSTRING\_INDEX(SUBSTRING\_INDEX(m.languages, ',', n.n), ',', -1) AS language**

**FROM movie m**

**JOIN (**

**SELECT a.N + b.N \* 10 + 1 AS n**

**FROM (SELECT 0 AS N UNION SELECT 1 UNION SELECT 2 UNION SELECT 3 UNION SELECT 4**

**UNION SELECT 5 UNION SELECT 6 UNION SELECT 7 UNION SELECT 8 UNION SELECT 9) a,**

**(SELECT 0 AS N UNION SELECT 1 UNION SELECT 2 UNION SELECT 3 UNION SELECT 4**

**UNION SELECT 5 UNION SELECT 6 UNION SELECT 7 UNION SELECT 8 UNION SELECT 9) b**

**) n**

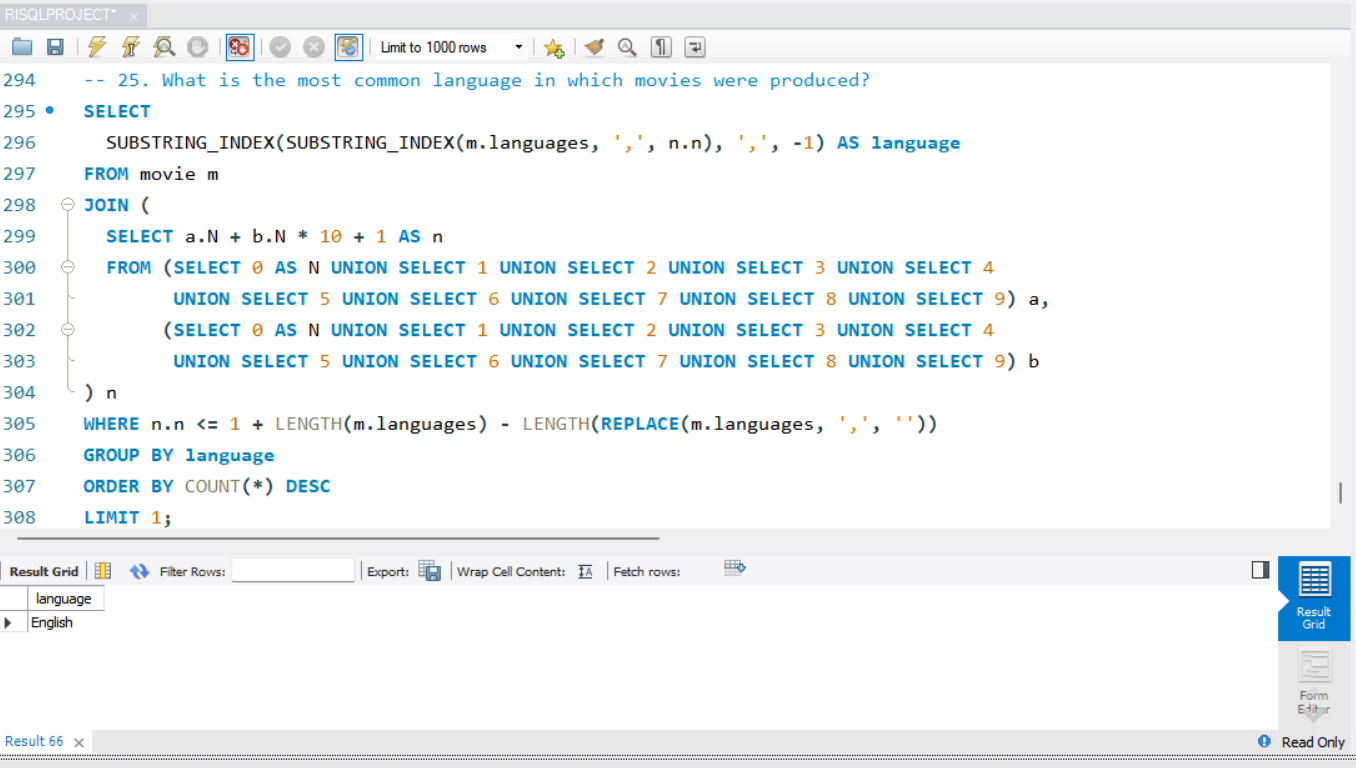
**WHERE n.n <= 1 + LENGTH(m.languages) - LENGTH(REPLACE(m.languages, ',', ''))**

**GROUP BY language**

**ORDER BY COUNT(\*) DESC**

**LIMIT 1;**

OUTPUT:

****