RSVP-Movies

```
USE imdb;
```

/ Now that you have imported the data sets, let’s explore some of the tables. To begin with, it is beneficial to know the shape of the tables and whether any column has null values. Further in this segment, you will take a look at 'movies' and 'genre' tables. /

```
show tables;
desc imdb.director_mapping;
desc imdb.role_mapping;
desc imdb.ratings;
desc imdb.genre;
desc imdb.movie;
desc imdb.names;
```

-- above lines of code will give us the description of each tables, structure and kind of data in persent in table

```
select * from director_mapping limit 1;
select * from genre limit 1;
select * from movie limit 1;
select * from names limit 1;
select * from ratings limit 1;
select * from role_mapping limit 1;
```

-- -- select queries with limit for a quick overview of the data

/*markdown

Segment 1:

*/

-- Q1. Find the total number of rows in each table of the schema? -- Type your code below:

```
SELECT 'director_mapping' AS 'Table_Name',
        Count(*) AS 'rows_count'
FROM director_mapping
UNION
SELECT 'genre',
```

```
        Count(*) AS 'rows_count'
FROM genre
UNION ALL
SELECT 'movie',
        Count(*) AS 'rows_count'
FROM movie
UNION ALL
SELECT 'names',
        Count(*) AS 'rows_count'
FROM names
UNION ALL
SELECT 'ratings',
        Count(*) AS 'rows_count'
FROM ratings
UNION ALL
SELECT 'role_mapping',
        Count(*) AS 'rows_count'
FROM role_mapping;
```

/* Notes Used union all to dislay total count of rows of all the tables in single query, instead of separate queries as output. Output for Q1:

Table_Name	rows_count
director_mapping	3867
genre	14662
movie	7997
names	25735
ratings	7997
role_mapping	15615

---

*/

-- Q2. Which columns in the movie table have null values? -- Type your code below:

```
SELECT SUM(IF(id IS NULL, 1, 0)) AS id_null_cnt,
    SUM(IF(title IS NULL, 1, 0)) AS title_null_cnt,
```

```
    SUM(IF(`year` IS NULL, 1, 0)) AS year_null_cnt,
    SUM(IF(date_published IS NULL, 1, 0)) AS date_published_null_cnt,
    SUM(IF(duration IS NULL, 1, 0)) AS duration_null_cnt,
    SUM(IF(country IS NULL, 1, 0)) AS country_null_cnt,
    SUM(IF(worlwide_gross_income IS NULL, 1, 0)) AS
worlwide_gross_income_null_cnt,
    SUM(IF(languages IS NULL, 1, 0)) AS languages_null_cnt,
    SUM(IF(production_company IS NULL, 1, 0)) AS production_company_null_cnt
FROM imdb.movie;
```

/* Notes: -Used IF condition to replace null values with 1 and sum function to calculate the total null values of each column in movies table. -We are getting null values in following 4 columns of movies table:- 1.country 2.worlwide_gross_income 3.languages 4.production_company Output for Q2:

---

id_null_cnt	title_null_cnt	year_null_cnt	date_published_null_cnt	duration_
0	0	0	0	0

---

*/

-- Now as you can see four columns of the movie table has null values. Let's look at the at the movies released each year. -- Q3. Find the total number of movies released each year? How does the trend look month wise? (Output expected) / *Output format for the first part: +---------------+-------------------+ | Year | number_of_movies| +-------------------+---------------- | 2017 | 2134 | | 2018 | . | | 2019 | . | +---------------+-------------------+ Output format for the second part of the question: +---------------+-------------------+ | month_num | number_of_movies| +---------------+------------------- | 1 | 134 | | 2 | 231 | | . | . | +---------------+-------------------+ /* -- Type your code below:

```
SELECT `year` AS Year,
    Count(id) AS number_of_movies
FROM imdb.movie
GROUP BY year;
```

/* Note:

- The above query will give us the total number of movies released in each year
- Maxinum number of movies produced is in the year 2017 with 3052 movies Output for Q3:

Year	number_of_movies
2017	3052
2018	2944
2019	2001

---

*/

```
SELECT Month(date_published) AS month_num,
    Count(id) AS number_of_movies
FROM imdb.movie
GROUP BY month_num
ORDER BY number_of_movies DESC;
```

/* Note

- above query will give us the total number of movies released in each month
- maximum number of movies released are in the month of March with 824 movies. Output for Q3:

---

month_num	number_of_movies
3	824
9	809
1	804
10	801
4	680
8	678
2	640
11	625
5	625
6	580

month_num	number_of_movies
7	493
12	438

---

*/

/ The highest number of movies is produced in the month of March.So, now that you have understood the month-wise trend of movies, let's take a look at the other details in the movies table. We know USA and India produces huge number of movies each year. Lets find the number of movies produced by USA or India for the last year. / -- Q4. How many movies were produced in the USA or India in the year 2019?? -- Type your code below:

```
SELECT count(distinct(id)) AS Movies_Produced_USA_India
FROM imdb.movie
WHERE REGEXP_LIKE(country, '.*(USA|India).*')
AND `year` IN ('2019');
```

/* Note

- Used REGEXP_LIKE to filter rows with countries values USA or India,
- Then counted the total distinct movies released in 2019 Output for Q4:

Movies_Produced_USA_India
1059

---

*/

*/ USA and India produced more than a thousand movies(you know the exact number!) in the year 2019. Exploring table Genre would be fun!! Let's find out the different genres in the dataset./ -- Q5. Find the unique list of the genres present in the data set? -- Type your code below:

```
SELECT DISTINCT(genre) FROM imdb.genre;
```

/* Note

- used distinct here for not getting repetitive value of genre and there are 13 genre categories Output for Q5:

---

genre
Drama
Fantasy
Thriller
Comedy
Horror
Family
Romance
Adventure
Action
Sci-Fi
Crime
Mystery
Others

---

*/

/ So, RSVP Movies plans to make a movie of one of these genres. Now, wouldn't you want to know which genre had the highest number of movies produced in the last year? Combining both the movie and genres table can give more interesting insights. / -- since last year is not specified then we are considering 2019 as last year /*

```
with movies_cnt_cte as (select genre, count(distinct(m.id)) as movies_cnt from
imdb.movie m
        inner join imdb.genre g on g.movie_id = m.id
        where `year` = 2019
        group by g.genre
)select genre, movies_cnt FROM movies_cnt_cte where movies_cnt = (select
```

```
Max(movies_cnt) FROM movies_cnt_cte);
-- movies produced in the last year
```

output

genre	movies_cnt
Drama	1078

---

*/

-- Q6.Which genre had the highest number of movies produced overall? -- Type your code below:

```
WITH movies_cnt_cte AS (
    SELECT genre,
        Count(m.id) AS movies_cnt
    FROM imdb.movie m
        INNER JOIN imdb.genre g ON g.movie_id = m.id
    GROUP BY g.genre
)
SELECT genre,
    movies_cnt
FROM movies_cnt_cte
WHERE movies_cnt = (
        SELECT Max(movies_cnt)
        FROM movies_cnt_cte
    );
```

/* Note

- Here we used Common Table Expression (CTE) to get the total number of movies for each genre.
- Then used the max function in where clause to find the topmost genre. Output for Q6:

---

genre	movies_cnt
Drama	4285

---

*/

/ So, based on the insight that you just drew, RSVP Movies should focus on the 'Drama' genre. But wait, it is too early to decide. A movie can belong to two or more genres. So, let's find out the count of movies that belong to only one genre./ -- Q7. How many movies belong to only one genre? -- Type your code below:

```
WITH movie_genre_cnt_cte AS (
    SELECT g.movie_id,
        COUNT(g.genre) AS genre_cnt
    FROM imdb.movie m
        INNER JOIN imdb.genre g ON g.movie_id = m.id
    GROUP BY g.movie_id
    HAVING genre_cnt = 1
)
SELECT COUNT(movie_id) AS "genre_cnt"
FROM movie_genre_cnt_cte;
```

/* Note

- We have used CTE to group the movies that have only one genre and then finally counted the total movies from movie_genre_cnt_cte. Output for Q7:

---

genre_cnt
3289

---

*/

/ There are more than three thousand movies which has only one genre associated with them. So, this figure appears significant. Now, let's find out the possible duration of RSVP Movies' next project./ -- Q8.What is the average duration of movies in each genre? -- (Note: The same movie can belong to multiple genres.) */ Output format: +---------------+-------------------+ | genre | avg_duration | +---------------+------------------- | thriller | 105 | | . | . | | . | . | +---------------+-------------------+ */* -- Type your code below:

```
SELECT g.genre,
    ROUND(Avg(duration), 2) AS avg_duration
```

```
FROM imdb.movie m
    INNER JOIN imdb.genre g ON g.movie_id = m.id
GROUP BY g.genre;
```

/* Note -- used AVG FUNCTION to find avg duration for each genre, rounded to 2 decimal places

- maximum avg_duration is 112.88 which is for Action genre
- minimum avg_duration is 92.72 which is for horror genre Output for Q8:

---

genre	avg_duration
Drama	106.77
Fantasy	105.14
Thriller	101.58
Comedy	102.62
Horror	92.72
Family	100.97
Romance	109.53
Adventure	101.87
Action	112.88
Sci-Fi	97.94
Crime	107.05
Mystery	101.80
Others	100.16

---

*/

/ Now you know, movies of genre 'Drama' (produced highest in number in 2019) has the average duration of 106.77 mins. Lets find where the movies of genre 'thriller' on the basis of number of movies./ -- Q9.What is the rank of the ‘thriller’ genre of movies among all the genres in terms of number of movies produced? -- (Hint: Use the Rank function) */ Output format: +---------------+-------------------+---------------------+ | genre | movie_count | genre_rank | +-------*

--------+------------------+--------------------+ /drama | 2312 | 2 | +--------------+------------------+---------------
-----+/ -- Type your code below:

```
WITH cnt_genre_cte AS (
    SELECT genre,
        Count(movie_id) AS movie_count,
        RANK() OVER(
            ORDER BY Count(movie_id) DESC
        ) AS genre_rank
    FROM genre
    GROUP BY genre
)
SELECT *
FROM cnt_genre_cte
WHERE genre = 'Thriller';
```

/* Note

- We have ranked the genre based on the movie count by using rank function in cnt_genre_cte
- tThen filtered the result for finding the count of 'Thriller' genre. Output for Q9:

---

genre	movie_count	genre_rank
Thriller	1484	3

---

*/

/Thriller movies is in top 3 among all genres in terms of number of movies In the previous segment, you analyzed the movies and genres tables. In this segment, you will analyse the ratings table as well. To start with lets get the min and max values of different columns in the table/

/*markdown

Segment 2:

*/

-- Q10. Find the minimum and maximum values in each column of the ratings table except the movie_id column? / *Output format: +---------------+-----------------+-------------------+-------------------+---------------+---------------+ | min_avg_rating| max_avg_rating | min_total_votes | max_total_votes |min_median_rating|min_median_rating| +---------------+-----------------+-------------------+-------------------+---------------+---------------+ | 0 | 5 | 177 | 2000 | 0 | 8 | +---------------+-----------------+-------------------+-------------------+---------------+---------------+/* --* Type your code below:

```
SELECT Round(Min(avg_rating)) AS min_avg_rating,
    Round(Max(avg_rating)) AS max_avg_rating,
    Round(Min(total_votes)) AS min_total_votes,
    Round(Max(total_votes)) AS max_total_votes,
    Round(Min(median_rating)) AS min_median_rating,
    Round(Max(median_rating)) AS max_median_rating
FROM ratings;
```

/* Note

- Used min and max function for getting the min and max values for all the columns in rating table and round function to round it to 2 decimal places. Output for Q10:

---

min_avg_rating	max_avg_rating	min_total_votes	max_total_votes	min_mec
1	10	100	725138	1

---

*/

/ So, the minimum and maximum values in each column of the ratings table are in the expected range. This implies there are no outliers in the table. Now, let's find out the top 10 movies based on average rating./ -- Q11. Which are the top 10 movies based on average rating? / *Output format: +---------------+-------------------+---------------------+ | title | avg_rating | movie_rank | +---------------+-------------------+---------------------+ | Fan | 9.6 | 5 | | . | . | . | | . | . | . | | . | . | . | | . | . | . | +---------------+-------------------+---------------------+/* -- Type your code below: -- It's ok if RANK() or DENSE_RANK() is used too

```
WITH movie_rank_cte AS (
    SELECT m.title,
        r.avg_rating,
        DENSE_RANK() OVER(w1) AS movie_rank
    FROM movie AS m
```

```
        INNER JOIN ratings AS r ON r.movie_id = m.id
        WINDOW w1 AS (
            ORDER BY r.avg_rating DESC
        )
)
SELECT *
FROM movie_rank_cte
WHERE movie_rank <= 10
LIMIT 10;
```

/* Note: -- Used dense rank to rank the movies by avg rating in desc order, then finally filtered out top 10 movies based on movie_rank Output for Q11:

---

title	avg_rating	movie_rank
Kirket	10.0	1
Love in Kilnerry	10.0	1
Gini Helida Kathe	9.8	2
Runam	9.7	3
Fan	9.6	4
Android Kunjappan Version 5.25	9.6	4
Yeh Suhaagraat Impossible	9.5	5
Safe	9.5	5
The Brighton Miracle	9.5	5
Shibu	9.4	6

---

*/

/ Do you find your favorite movie FAN in the top 10 movies with an average rating of 9.6? If not, please check your code again!! So, now that you know the top 10 movies, do you think character actors and filler actors can be from these movies? Summarizing the ratings table based on the movie counts by median rating can give an excellent insight./ -- Q12. Summarize the ratings table based on the movie counts by median ratings. */ Output format: +---------------+-------------------+ | median_rating | movie_count | +-------------------+---------------- | 1 | 105 | | . | . | | . | . | +---------------+-------------------+ /* -- Type your code below: -- Order by is good to have

```
SELECT median_rating,
    Count(DISTINCT(movie_id)) AS cnt_by_rating
FROM movie m
    INNER JOIN ratings r ON m.id = r.movie_id
GROUP BY median_rating
ORDER BY cnt_by_rating DESC
```

/* Note:

- We have grouped the ratings table based on the movie counts by median ratings
- It turns out median rating 7 has most number of movies. Output for Q12:

---

median_rating	cnt_by_rating
7	2257
6	1975
8	1030
5	985
4	479
9	429
10	346
3	283
2	119
1	94

---

*/

/* Note:

- Movies with a median rating of 7 is highest in number.
- Now, let's find out the production house with which RSVP Movies can partner for its next project. */ -- *Q13. Which production house has produced the most number of hit movies (average rating > 8) ?* / Output format: +----------------+-----------------+-------------------+ |production_company|movie_count | prod_company_rank| +----------------+-----------------

+--------------------+ | The Archers | 1 | 1 | +-----------------+-----------------+--------------------+*/ -- Type your code below:

```
WITH top_production_company_cte AS (
    SELECT m.production_company,
        Count(DISTINCT(m.id)) AS movie_count,
        DENSE_RANK() OVER (
            ORDER BY Count(DISTINCT(m.id)) DESC
        ) AS prod_company_rank
    FROM movie m
        INNER JOIN ratings r ON m.id = r.movie_id
    WHERE avg_rating > 8
        AND m.production_company IS NOT NULL
    GROUP BY m.production_company
)
SELECT *
FROM top_production_company_cte
WHERE prod_company_rank = 1;
```

/* Note:

- With top_production_company_cte we are getting the production company which has avg_rating > 8 and used dense rank on movie count in desc order,
- finally filtered the top production company name. Output for Q13:

---

production_company	movie_count	prod_company_rank
Dream Warrior Pictures	3	1
National Theatre Live	3	1

---

*/

-- It's ok if RANK() or DENSE_RANK() is used too -- Answer can be Dream Warrior Pictures or National Theatre Live or both -- Q14. How many movies released in each genre during March 2017 in the USA had more than 1,000 votes? / *Output format: +---------------+-------------------+ | genre | movie_count | +---------------+------------------- | thriller | 105 | | . | . | | . | . | +---------------+-------------------+* / -- Type your code below:

```
SELECT g.genre,
    COUNT(m.id) AS movie_count
FROM movie m
    INNER JOIN ratings r ON m.id = r.movie_id
    INNER JOIN genre g ON m.id = g.movie_id
WHERE `year` = 2017
    AND Month(date_published) = 3
    AND REGEXP_LIKE(country, '.*USA.*')
    AND r.total_votes > 1000
GROUP BY g.genre
ORDER BY movie_count DESC;
```

/* Note

- used inner join to join movie, rating and genre tables.
- additionally used regex to filter out the rows that has USA in country column
- used month function to filter the month of march, then grouped all movies based on genre.
- Drama has 24 movies, made in USA, during the month of march 2017 Output for Q14:

---

genre	movie_count
Drama	24
Comedy	9
Action	8
Thriller	8
Sci-Fi	7
Crime	6
Horror	6
Mystery	4
Romance	4
Fantasy	3
Adventure	3
Family	1

---

*/

-- Lets try to analyse with a unique problem statement. -- Q15. Find movies of each genre that start with the word ‘The’ and which have an average rating > 8? / *Output format: +---------------+-------------------+---------------------+ | title | avg_rating | genre | +---------------+-------------------+---------------------+ | Theeran | 8.3 | Thriller | | . | . | . | | . | . | . | | . | . | . | +---------------+-------------------+---------------------+*/ -- Type your code below:

```
SELECT DISTINCT(m.title),
    r.avg_rating,
    g.genre
FROM movie m
    INNER JOIN ratings r ON m.id = r.movie_id
    INNER JOIN genre g ON m.id = g.movie_id
WHERE m.title LIKE "The%"
    AND r.avg_rating > 8
ORDER BY avg_rating DESC;
```

/* Note

- We have used like operator to filter out the movies name which starts with 'THE'. OUTPUT FOR Q15:

---

title	avg_rating	genre
The Blue Elephant 2	8.8	Drama
The Blue Elephant 2	8.8	Horror
The Blue Elephant 2	8.8	Mystery
The Brighton Miracle	9.5	Drama
The Irishman	8.7	Crime
The Irishman	8.7	Drama
The Colour of Darkness	9.1	Drama
Theeran Adhigaaram Ondru	8.3	Action
Theeran Adhigaaram Ondru	8.3	Crime
Theeran Adhigaaram Ondru	8.3	Thriller
The Mystery of Godliness: The Sequel	8.5	Drama
The Gambinos	8.4	Crime

title	avg_rating	genre
The Gambinos	8.4	Drama
The King and I	8.2	Drama
The King and I	8.2	Romance

---

*/

-- You should also try your hand at median rating and check whether the ‘median rating’ column gives any significant insights. -- Q16. Of the movies released between 1 April 2018 and 1 April 2019, how many were given a median rating of 8? -- Type your code below:

```
SELECT COUNT(*) AS 'Count of movies >8 rating'
FROM movie m
    INNER 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);
```

/* Note:

- used 'between' 'and' to filter dates within range. Output for Q16:

---

cnt_movie
361

---

*/ -- Once again, try to solve the problem given below. -- Q17. Do German movies get more votes than Italian movies? -- Hint: Here you have to find the total number of votes for both German and Italian movies. -- Type your code below:

```
SELECT m.country,
    SUM(total_votes) AS Total_votes
FROM movie m
    INNER JOIN ratings r ON m.id = r.movie_id
```

```
WHERE m.country IN ('Germany', 'Italy')
GROUP BY m.country;
```

/* Note

- used 'IN' in where clause to find the movies produced in Germany or Italy
- its observed, we have high votes in Germany Output for Q17:

---

country	Total_votes
Germany	106710
Italy	77965

---

*/

/*markdown

Segment 3:

*/

-- Answer is Yes */ Now that you have analyzed the movies, genres and ratings tables, let us now analyse another table, the names table. Let's begin by searching for null values in the tables./* -- Q18. Which columns in the names table have null values?? */Hint: You can find null values for individual columns or follow below output format +---------------+-------------------+---------------------+----------------------+ | name_nulls | height_nulls |date_of_birth_nulls |known_for_movies_nulls| +---------------+-------------------+---------------------+----------------------+ | 0 | 123 | 1234 | 12345 | +---------------+-------------------+---------------------+----------------------+/* -- Type your code below:

```
SELECT SUM(IF(id is NULL, 1, 0)) AS id_null_cnt,
    SUM(IF(name is NULL, 1, 0)) AS name_null_cnt,
    SUM(IF(height is NULL, 1, 0)) AS height_null_cnt,
    SUM(IF(date_of_birth is NULL, 1, 0)) AS date_of_birth_null_cnt,
    SUM(IF(known_for_movies is NULL, 1, 0)) AS known_for_movies_null_cnt
FROM imdb.names;
```

/* Note

- There are null values in 'height_null_cnt', 'date_of_birth_null_cnt', 'known_for_movies_null_cnt' Output for Q18:

---

id_null_cnt	name_null_cnt	height_null_cnt	date_of_birth_null_cnt	known_f
0	0	17335	13431	15226

---

*/

*/ There are no Null value in the column 'name'. The director is the most important person in a movie crew. Let's find out the top three directors in the top three genres who can be hired by RSVP Movies */* -- Q19. Who are the top three directors in the top three genres whose movies have an average rating > 8? -- (Hint: The top three genres would have the most number of movies with an average rating > 8.) */ Output format: +---------------+-------------------+ | director_name | movie_count | +---------------+-------------------| |James Mangold | 4 | | . | . | | . | . | +---------------+-------------------+ /* -- Type your code below:

```
WITH filtered_mov_cte AS (
    SELECT mo.id,
        ra.avg_rating,
        ge.genre,
        nm.name
    FROM imdb.movie mo
        INNER JOIN imdb.ratings ra ON mo.id = ra.movie_id
        INNER JOIN imdb.genre ge ON mo.id = ge.movie_id
        INNER JOIN imdb.director_mapping dm ON mo.id = dm.movie_id
        INNER JOIN imdb.names nm ON dm.name_id = nm.id
    WHERE ra.avg_rating > 8
),
top_genre_cte AS (
    SELECT fmc.genre,
        COUNT(distinct(fmc.id)) AS mov_cnt,
        DENSE_RANK() OVER (
            ORDER BY COUNT(distinct(fmc.id)) DESC
        ) AS mov_cnt_rank
    FROM filtered_mov_cte fmc
    GROUP BY fmc.genre
)
SELECT name AS director_name,
```

```
    COUNT(distinct(id)) AS movie_count
FROM filtered_mov_cte
WHERE genre IN (
        SELECT genre
        FROM top_genre_cte
        WHERE mov_cnt_rank <= 3
    )
GROUP BY name
ORDER BY movie_count DESC
LIMIT 3;
```

/* Note:

- Joined movie, rating, genre, director mapping, names table and filtered out rows that have avg_rating > 8 and saved the result as filtered_mov_cte
- Used Dense rank to find the top ranked genre and saved the result in top_genre_cte
- finally used sub-query to filter out the top 3 genre to find the top ranked directors based on movie count. Output for Q19:

---

director_name	movie_count	mov_row_rank
Anthony Russo	2	1
James Mangold	2	2
Joe Russo	2	3

---

*/

/ James Mangold can be hired as the director for RSVP's next project. Do you remember his movies, 'Logan' and 'The Wolverine'. Now, let's find out the top two actors./ -- Q20. Who are the top two actors whose movies have a median rating >= 8? */ Output format: +---------------+--------------------+ | actor_name | movie_count | +-------------------+---------------- |Christain Bale | 10 | | . | . | +---------------+-------------------+ /* -- Type your code below:

```
WITH actor_movie_cte AS (
    SELECT nm.name,
        COUNT(distinct(mo.id)) AS movie_count,
        DENSE_RANK() OVER (ORDER BY COUNT(distinct(mo.id)) DESC) AS
movie_count_rank
```

```
    FROM imdb.movie mo
        INNER JOIN imdb.ratings ra ON mo.id = ra.movie_id
        INNER JOIN imdb.role_mapping rm ON mo.id = rm.movie_id
        INNER JOIN imdb.names nm ON rm.name_id = nm.id
    WHERE ra.median_rating >= 8
        AND rm.category = 'actor'
    GROUP BY nm.name
    ORDER BY movie_count DESC
)
SELECT name AS actor_name,
    movie_count
FROM actor_movie_cte
WHERE movie_count_rank <= 2;
```

/* Note we have joined movie, ratings, role_mapping, names tables for getting desired output as CTE output and then we are filtering out top 2 actors from actor_movie_cte based on movie_count_rank. Output for Q20:

---

actor_name	movie_count
Mammootty	8
Mohanlal	5

---

*/

/ Have you find your favourite actor 'Mohanlal' in the list. If no, please check your code again. RSVP Movies plans to partner with other global production houses. Let's find out the top three production houses in the world./ -- Q21. Which are the top three production houses based on the number of votes received by their movies? */ Output format: +------------------+------------------+--------------------+ |production_company|vote_count | prod_comp_rank| +------------------+------------------+--------------------+ | The Archers | 830 | 1 | | . | . | . | | . | . | . | +------------------+------------------+--------------------+*/ -- Type your code below:

```
WITH prod_comp_rank_cte AS (
    SELECT mo.production_company,
        SUM(total_votes) AS vote_count,
        RANK() OVER (
            order by SUM(total_votes) DESC
```

```
        ) AS prod_comp_rank
    FROM imdb.movie mo
        INNER JOIN imdb.ratings ra ON mo.id = ra.movie_id
    WHERE mo.production_company is not null
    GROUP BY mo.production_company
)
SELECT *
FROM prod_comp_rank_cte
WHERE prod_comp_rank <= 3;
```

/* Note we have created CTE for getting production_company ,vote_count and prod_comp_rank using 2 tables. then filtered out the top 3 production company name. Output for Q22:

---

production_company	vote_count	prod_comp_rank
Marvel Studios	2656967	1
Twentieth Century Fox	2411163	2
Warner Bros.	2396057	3

---

*/

/Yes Marvel Studios rules the movie world. So, these are the top three production houses based on the number of votes received by the movies they have produced. Since RSVP Movies is based out of Mumbai, India also wants to woo its local audience. RSVP Movies also wants to hire a few Indian actors for its upcoming project to give a regional feel. Let's find who these actors could be./ -- Q22. Rank actors with movies released in India based on their average ratings. Which actor is at the top of the list? -- Note: The actor should have acted in at least five Indian movies. -- (Hint: You should use the weighted average based on votes. If the ratings clash, then the total number of votes should act as the tie breaker.) */ Output format: +---------------+-------------------+---------------------+----------------------+-----------------+ | actor_name | total_votes | movie_count | actor_avg_rating |actor_rank | +---------------+-------------------+---------------------+----------------------+-----------------+ | Yogi Babu | 3455 | 11 | 8.42 | 1 | | . | . | . | . | . | | . | . | . | . | . | . | | . | . | . | . | . | . | . | +---------------+-------------------+---------------------+----------------------+-----------------+/ --* Type your code below:

```
WITH actor_movie_cte AS (
    SELECT mo.id,
```

```
        nm.name,
        ra.total_votes,
        ra.avg_rating,
        (total_votes * (ra.avg_rating)) AS w_score
    FROM imdb.movie mo
        INNER JOIN imdb.ratings ra ON mo.id = ra.movie_id
        INNER JOIN imdb.role_mapping rm ON mo.id = rm.movie_id
        INNER JOIN imdb.names nm ON rm.name_id = nm.id
    WHERE rm.category = 'actor'
        AND mo.country LIKE '%India%'
),
actor_rating_cte AS (
    SELECT NAME AS actor_name,
        SUM(total_votes) AS total_votes,
        COUNT(id) AS movie_count,
        SUM(w_score) / SUM(total_votes) AS actor_avg_rating
    FROM actor_movie_cte
    GROUP BY actor_name
    HAVING COUNT(id) >= 5
)
SELECT *,
    DENSE_RANK() OVER (
        ORDER BY actor_avg_rating DESC,
            total_votes DESC
    ) AS actor_rank
FROM actor_rating_cte
LIMIT 1;
```

/* Note: we have created 2 CTEs, first one is for getting the actor movies based in country 'India'. and another CTE gives us the actor rating. and then we are filtering out the top actor name along with total_votes movie_count actor_avg_rating and actor_rank Output for Q22:

---

actor_name	total_votes	movie_count	actor_avg_rating	actor_rank
Vijay Sethupathi	23114	5	8.41673	1

---

*/

-- Top actor is Vijay Sethupathi -- Q23.Find out the top five actresses in Hindi movies released in India based on their average ratings? -- Note: The actresses should have acted in at least three Indian movies. -- (Hint: You should use the weighted average based on votes. If the ratings clash, then the total number of votes should act as the tie breaker.) / *Output format: +---------------+-------------------+---------------------+----------------------+-----------------+ | actress_name | total_votes | movie_count | actress_avg_rating |actress_rank | +---------------+-------------------+---------------------+----------------------+-----------------+ | Tabu | 3455 | 11 | 8.42 | 1 | | . | . | . | . | . | | . | | . | . | . | . | . | | . | . | . | . | . | +---------------+-------------------+---------------------+----------------------+-----------------+*/ -- Type your code below:

```
WITH actress_movie_cte AS (
    SELECT mo.id,
        nm.name,
        ra.total_votes,
        ra.avg_rating,
        (total_votes * (ra.avg_rating)) AS w_score
    FROM imdb.movie mo
        INNER JOIN imdb.ratings ra ON mo.id = ra.movie_id
        INNER JOIN imdb.role_mapping rm ON mo.id = rm.movie_id
        INNER JOIN imdb.names nm ON rm.name_id = nm.id
    WHERE rm.category = 'actress'
        AND mo.country LIKE '%India%'
        AND languages LIKE '%Hindi%'
),
actress_rating_cte AS (
    SELECT NAME AS actress_name,
        SUM(total_votes) AS total_votes,
        COUNT(id) AS movie_count,
        SUM(w_score) / SUM(total_votes) AS actress_avg_rating
    FROM actress_movie_cte
    GROUP BY NAME
    HAVING COUNT(id) >= 3
)
SELECT *,
    DENSE_RANK() OVER (
        ORDER BY actress_avg_rating DESC,
            total_votes DESC
    ) AS actress_rank
FROM actress_rating_cte
LIMIT 5;
```

/* Note using 1st CTE we are getting Actress Movie, based in India in Hindi language, using 2nd CTE we are getting Actress name, who have worked in atleast 3 movies. then finally we

are selecting top 5 actress on the basis of rank. Output for Q23:

---

actress_name	total_votes	movie_count	actress_avg_rating	actress_rank
Taapsee Pannu	18061	3	7.73692	1
Kriti Sanon	21967	3	7.04911	2
Divya Dutta	8579	3	6.88440	3
Shraddha Kapoor	26779	3	6.63024	4
Kriti Kharbanda	2549	3	4.80314	5

---

*/

/ Taapsee Pannu tops with average rating 7.74. Now let us divide all the thriller movies in the following categories and find out their numbers./ / Q24. Select thriller movies as per avg rating and classify them in the following category: Rating > 8: Superhit movies Rating between 7 and 8: Hit movies Rating between 5 and 7: One-time-watch movies Rating < 5: Flop movies ----------/ -- Type your code below:

```
WITH movie_rating_cat_cte AS (
    SELECT mo.title,
CASE
            WHEN ra.avg_rating > 8 THEN 'Superhit movies'
            WHEN ra.avg_rating BETWEEN 7 AND 8 THEN 'Hit movies'
            WHEN ra.avg_rating BETWEEN 5 AND 7 THEN 'One-time-watch movies'
            ELSE 'Flop movies'
        END AS category
    FROM imdb.movie mo
        INNER JOIN imdb.ratings ra ON mo.id = ra.movie_id
        INNER JOIN imdb.genre ge ON mo.id = ge.movie_id
    WHERE ge.genre = 'Thriller'
)
SELECT *
FROM movie_rating_cat_cte;
```

/* Note: we are conditioning with below conditions for thriller movies 1.Rating > 8: Superhit movies, 2.Rating between 7 and 8: Hit movies, 3.Rating between 5 and 7: One-time-watch movies, 4.Rating < 5: Flop movies in CTE and then we are selecting Title and Category OUTPUT for Q24:

| title | category |

| Der müde Tod | Hit movies | | Fahrenheit 451 | Flop movies | | Pet Sematary | One-time-watch movies| | Dukun | One-time-watch movies| | Back Roads | Hit movies |

---

we have shown only top 5 values of output. */

/*

```
SELECT category, count(category) as category_count from movie_rating_cat_cte
GROUP BY category
ORDER BY category_count desc;
```

//Until now, you have analyzed various tables of the data set. Now, you will perform some tasks that will give you a broader understanding of the data in this segment.*/

/*markdown

Segment 4:

*/

-- Q25. What is the genre-wise running total and moving average of the average movie duration? -- (Note: You need to show the output table in the question.) / *Output format: +----------+------------------+--------------------+----------------------+ | genre | avg_duration |running_total_duration|moving_avg_duration | +-----------+------------------+--------------------+----------------------+ | comedy | 145 | 106.2 | 128.42 | | . | . | . | . | | . | . | . | . | | . | . | . | . | . | +-----------+------------------+--------------------+----------------------+*/ -- Type your code below:

```
WITH genre_duration_cte AS (
    SELECT genre,
        AVG(mo.duration) AS avg_duration
    FROM imdb.movie mo
        INNER JOIN imdb.genre ge ON mo.id = ge.movie_id
    GROUP BY ge.genre
)
```

```
SELECT genre,
    round(avg_duration, 2) AS avg_duration,
    round(SUM(avg_duration) OVER w1, 2) AS running_total_duration,
    round(AVG(avg_duration) OVER w2, 2) AS moving_avg_duration
FROM genre_duration_cte
WINDOW w1 AS (ORDER BY genre ROWS UNBOUNDED PRECEDING),
       w2 AS (ORDER BY genre ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING);
```

/* Note we have created CTE for getting result based on avg_duration. then getting desired output using window function. w1 is used for getting running avg, w2 is used for getting moving avg with 1 preceding and 1 following value.

---

genre	avg_duration	running_total_duration	moving_avg_duration
Action	112.88	112.88	107.38
Adventure	101.87	214.75	105.79
Comedy	102.62	317.38	103.85
Crime	107.05	424.43	105.48
Drama	106.77	531.20	104.93
Family	100.97	632.17	104.29
Fantasy	105.14	737.31	99.61
Horror	92.72	830.03	99.89
Mystery	101.80	931.83	98.23
Others	100.16	1031.99	103.83
Romance	109.53	1141.53	102.55
Sci-Fi	97.94	1239.47	103.02
Thriller	101.58	1341.05	99.76

---

*/

-- Round is good to have and not a must have; Same thing applies to sorting -- Let us find top 5 movies of each year with top 3 genres. -- Q26. Which are the five highest-grossing movies of each year that belong to the top three genres? -- (Note: The top 3 genres would have the most number of movies.) / *Output format: +---------------+-------------------+---------------------*

+---------------------+-----------------+ | genre | year | movie_name |worldwide_gross_income|movie_rank | +---------------+-------------------+---------------------+-------------------- ------+-----------------+ | comedy | 2017 | indian | $103244842 | 1 | | . | . | . | . | . | | . | . | . | . | . | | . | . | . | . | . | +---------------+-------------------+---------------------+---------------------+-----------------+/ -- Type your code below: -- Top 3 Genres based on most number of movies

```
WITH genre_movies_cte AS (
    SELECT *
    FROM imdb.movie mo
        INNER JOIN imdb.genre ge ON mo.id = ge.movie_id
),
genre_group_cte AS (
    SELECT genre,
        COUNT(distinct(id)) movie_count,
        DENSE_RANK() OVER (ORDER BY COUNT(distinct(id)) DESC ) AS genre_rank
    FROM genre_movies_cte
    GROUP BY genre
),
top_genre_cte AS (
    SELECT genre
    FROM genre_group_cte
    WHERE genre_rank <= 3
),
top_movies_cte AS (
    SELECT genre,
        `year`,
        title AS movie_name,
        worlwide_gross_income,
        CAST( REGEXP_SUBSTR(worlwide_gross_income, '\\d+') AS DECIMAL ) AS
worldwide_gross_income,
        DENSE_RANK() OVER (
            PARTITION BY `year`
            ORDER BY CAST(REGEXP_SUBSTR(worldwide_gross_income, '\\d+') AS
DECIMAL) DESC
        ) AS movie_rank
    FROM genre_movies_cte
    WHERE genre IN (
            SELECT genre
            FROM top_genre_cte
        )
        AND worlwide_gross_income is not null
)
SELECT genre,
```

```
    `year`,
    movie_name,
    worlwide_gross_income,
    movie_rank
FROM top_movies_cte
WHERE movie_rank <= 5;
```

/* we are collecting genre in first CTE then in 2nd CTE we are doing grouping those genre then 3rd CTE we are filtering top 3 genre, then in 4th CTE we are doing partitioning and providing rank based on the income of the movie. Output for Q26:

---

genre	year	movie_name	worlwide_gross_income	movie_rank
Thriller	2017	The Fate of the Furious	$ 1236005118	1
Comedy	2017	Despicable Me 3	$ 1034799409	2
Comedy	2017	Jumanji: Welcome to the Jungle	$ 962102237	3
Drama	2017	Zhan lang II	$ 870325439	4
Thriller	2017	Zhan lang II	$ 870325439	4
Comedy	2017	Guardians of the Galaxy Vol. 2	$ 863756051	5
Thriller	2018	The Villain	INR 1300000000	1
Drama	2018	Bohemian Rhapsody	$ 903655259	2
Thriller	2018	Venom	$ 856085151	3
Thriller	2018	Mission: Impossible - Fallout	$ 791115104	4
Comedy	2018	Deadpool 2	$ 785046920	5
Drama	2019	Avengers: Endgame	$ 2797800564	1
Drama	2019	The Lion King	$ 1655156910	2
Comedy	2019	Toy Story 4	$ 1073168585	3
Drama	2019	Joker	$ 995064593	4
Thriller	2019	Joker	$ 995064593	4
Thriller	2019	Ne Zha zhi mo tong jiang shi	$ 700547754	5

*/

-- Finally, let's find out the names of the top two production houses that have produced the highest number of hits among multilingual movies. -- Q27. Which are the top two production houses that have produced the highest number of hits (median rating >= 8) among multilingual movies? / *Output format: +----------------+----------------+-------------------+ |production_company |movie_count | prod_comp_rank| +----------------+----------------+-------------------+ | The Archers | 830 | 1 | | . | . | . | | . | . | . | +----------------+----------------+-------------------+*/
-- Type your code below:

```
WITH production_houses_cte AS (
    SELECT mo.production_company,
        COUNT(distinct(mo.id)) AS movie_count,
        DENSE_RANK() OVER (
            order by COUNT(mo.id) DESC
        ) AS prod_comp_rank
    FROM movie mo
        INNER JOIN ratings ra ON mo.id = ra.movie_id
    WHERE (ra.median_rating >= 8)
        AND (mo.production_company is not null)
        AND (POSITION(',' IN mo.languages) > 0)
    GROUP BY mo.production_company
)
SELECT *
FROM production_houses_cte
WHERE prod_comp_rank <= 2;
```

/* Note

- Used 'POSITION' function to filter out rows, if the language column has multiple languages based on the availability of ','
- Dense Rank to rank production company based on movie count and saved the result as production_houses_cte
- Finally selected the top 2 production company from "production_houses_cte" Output for Q27:

---

production_company	movie_count	prod_comp_rank
Star Cinema	7	1

production_company	movie_count	prod_comp_rank
Twentieth Century Fox	4	2

---

*/

-- Multilingual is the important piece in the above question. It was created using POSITION(',' IN languages)>0 logic -- If there is a comma, that means the movie is of more than one language -- Q28. Who are the top 3 actresses based on number of Super Hit movies (average rating >8) in drama genre? / *Output format: +---------------+-------------------+---------------------+----------------------+-----------------+ | actress_name | total_votes | movie_count |actress_avg_rating |actress_rank | +---------------+-------------------+---------------------+----------------------+-----------------+ | Laura Dern | 1016 | 1 | 9.60 | 1 | | . | . | . | . | . | . | | . | . | . | . | . | . | +--------------+--------------------+---------------------+----------------------+-----------------+/* -- Type your code below:

```
WITH actress_movie_cte AS (
    SELECT mo.id,
        nm.name AS actress_name,
        ra.total_votes,
        ra.avg_rating,
        (total_votes * (ra.avg_rating)) AS w_score
    FROM imdb.movie mo
        INNER JOIN imdb.genre ge ON mo.id = ge.movie_id
        INNER JOIN imdb.ratings ra ON mo.id = ra.movie_id
        INNER JOIN imdb.role_mapping rm ON mo.id = rm.movie_id
        INNER JOIN imdb.names nm ON rm.name_id = nm.id
    WHERE ra.avg_rating > 8
        AND ge.genre = 'Drama'
        AND rm.category = 'actress'
),
actress_rating_cte AS (
    SELECT actress_name,
        SUM(total_votes) AS total_votes,
        COUNT(id) AS movie_count,
        round(SUM(w_score) / SUM(total_votes), 2) AS actress_avg_rating,
        -- ROW_NUMBER() OVER (order by round(sum(w_score)/sum(total_votes), 2)
desc) AS actress_rank,
        -- DENSE_RANK() OVER (
        --     order by round(SUM(w_score) / SUM(total_votes), 2) DESC
        DENSE_RANK() OVER (
            order by COUNT(id) DESC
```

```
        ) AS actress_rank
    FROM actress_movie_cte
    GROUP BY actress_name
)
SELECT actress_name,
    total_votes,
    movie_count,
    actress_avg_rating,
    actress_rank
FROM actress_rating_cte
WHERE actress_rank <= 3
limit 3;
```

/* Note

- In actress_movie_cte, joined movie, genre, rating, role_mappings, names tables, here we are getting the weighted_rating for actress from Drama genre and rating > 8
- and in actress_rating_cte we are getting actress_avg_rating along with other information too, used dense_rank to rank actress by average rating.
- finally selected the top 3 actress from actress_rating_cte Output for Q28:

---

actress_name	total_votes	movie_count	actress_avg_rating	actress_rank
Parvathy Thiruvothu	4974	2	8.25	1
Susan Brown	656	2	8.94	1
Amanda Lawrence	656	2	8.94	1

---

*/

/ Q29. Get the following details for top 9 directors (based on number of movies) Director id,Name,Number of movies,Average inter movie duration in days,Average movie ratings,Total votes,Min rating,Max rating,total movie durations Format: +---------------+-------------------+---------------------+----------------------+-------------+-------------+-----------+-----------+----------------+ | director_id | director_name | number_of_movies | avg_inter_movie_days | avg_rating | total_votes | min_rating | max_rating | total_duration | +---------------+-------------------+---------------------+----------------------+-------------+-------------+-----------+-----------+----------------+ |nm1777967 | A.L. Vijay | 5 | 177

| 5.65 | 1754 | 3.7 | 6.9 | 613 ||. | . | . | . | . | . | . | . ||. | . | . | . | . | . | . | . ||. | . | . | . | . | . | . | . ||. | . | . | . | . | . | . | . ||. | . | . | . | . | . | . | . ||. | . | . | . | . | . | . | . ||. | . | . | . | . | . | . | +--------------+----------------+------------------+-------------------+------------+------------ --+-----------+-----------+--------------+ --
----/ -- Type you code below:

```
WITH director_movie_cte AS (
    SELECT nm.id,
        nm.name,
        mo.id AS movie_id,
        mo.date_published,
        ra.total_votes,
        ra.avg_rating,
        mo.duration
    FROM imdb.movie mo
        INNER JOIN imdb.ratings ra ON mo.id = ra.movie_id
        INNER JOIN imdb.director_mapping dm ON dm.movie_id = mo.id
        INNER JOIN imdb.names nm ON dm.name_id = nm.id
    ORDER BY nm.name,
        mo.date_published
),
mov_date_diff_cte AS (
    SELECT *,
        IFNULL(
            DATEDIFF(date_published, LAG(date_published) OVER (
                PARTITION BY name ORDER BY date_published)
                ),
            0
        ) AS date_diff
    FROM director_movie_cte
),
ranking_cte AS (
    SELECT id AS 'Director id',
        name AS 'Name',
        COUNT(distinct(movie_id)) AS 'Number of movies',
        ROW_NUMBER() OVER(
            order by COUNT(distinct(movie_id)) DESC
        ) AS movie_cnt_rank,
        AVG(date_diff) AS 'avg_inter_movie_days',
        AVG(avg_rating) AS 'Average movie ratings',
        SUM(total_votes) AS 'Total votes',
        MIN(avg_rating) AS 'Min rating',
        MAX(avg_rating) AS 'Max rating',
```

```
        SUM(duration) AS 'Total movie durations'
    FROM mov_date_diff_cte
    GROUP BY id,
        name
    ORDER BY COUNT(distinct(movie_id)) DESC
)
SELECT `Director id`,
    `Name`,
    `Number of movies`,
    avg_inter_movie_days,
    `Average movie ratings`,
    `Total votes`,
    `Min rating`,
    `Max rating`,
    `Total movie durations`
FROM ranking_cte
WHERE movie_cnt_rank <= 9;
```

/* Note:

- Joined movie, ratings, director_mappings, names tables and saved the result as director_movie_cte ,
- Then used Datediff fun to find the difference in days between two releases and saved the result as mov_date_diff_cte
- Then collated all the results and grouped by director_id and performed required aggregations, and used rank fun to order by movie count, saved the result as ranking_cte
- Finally filtered the result to fetch top 9 directors based on movie_cnt_rank. Output for Q29:

Director id	Name	Number of movies	avg_inter_movie_days	Average movie ratings	Total votes
nm1777967	A.L. Vijay	5	141.4000	5.42000	1754
nm2096009	Andrew Jones	5	152.6000	3.02000	1989
nm0001752	Steven Soderbergh	4	190.7500	6.47500	171684

Director id	Name	Number of movies	avg_inter_movie_days	Average movie ratings	Total votes
nm0425364	Jesse V. Johnson	4	224.2500	5.45000	14778
nm0515005	Sam Liu	4	195.2500	6.22500	28557
nm0814469	Sion Sono	4	248.2500	6.02500	2972
nm0831321	Chris Stokes	4	148.7500	4.32500	3664
nm2691863	Justin Price	4	236.2500	4.50000	5343
nm6356309	Özgür Bakar	4	84.0000	3.75000	1092

---

*/