Netflix Content & User Behaviour Analysis Using SQL

By Using MySQL Workbench

Developed by: Rutuja Kamble

Overview

- This report presents a structured SQL-driven analysis of a streaming platform dataset to evaluate user engagement, content performance, and viewing behavior across multiple analytical dimensions.

- By leveraging interconnected tables containing user demographics, content metadata, ratings, and watch-time activity, the study uncovers insights related to content popularity, genre performance, user viewing patterns, high-value users, and rating trends.

- Through twenty well-defined SQL queries involving filtering, aggregation, joins, subqueries, and window functions, the analysis reveals patterns in user behavior, identifies top-performing genres and content types, and highlights factors influencing platform engagement.

- The findings support data-driven decision-making for content acquisition, recommendation improvements, user segmentation, and overall platform optimization.

Objectives

- Analyze the distribution of content types (Movies vs TV Shows) on the platform.
- Identify trends in user ratings to understand content quality and audience preferences.
- Study release year patterns to observe how content production has evolved over time.
- Examine user engagement behavior by analyzing watch time and activity patterns.
- Determine the popularity of content using a combination of watch activity and ratings.

Questions Solved

1 Who are our users and where do they come from?
- Identified total number of users, unique countries, users above/below certain age groups.

2 What types of content (Movies vs TV Shows) are most common on the platform?
- Counted movies vs TV shows.
- Found genre distribution and release year patterns.

3 Which users are the most active on the platform?
- Found users who watched **most movies**, highest watch-time, repeated watchers.

4 Which movies are trending?
- Most watched within a specific month/year.
- Recently released shows with high watch-time.

5 What are the user rating patterns?
- Average rating per movie.
- Movies with ratings above 4.
- Users giving the highest number of ratings.

6 Does user behavior differ by country or age?
- Users grouped by country.
- Users older/younger than 25 and their behavior.

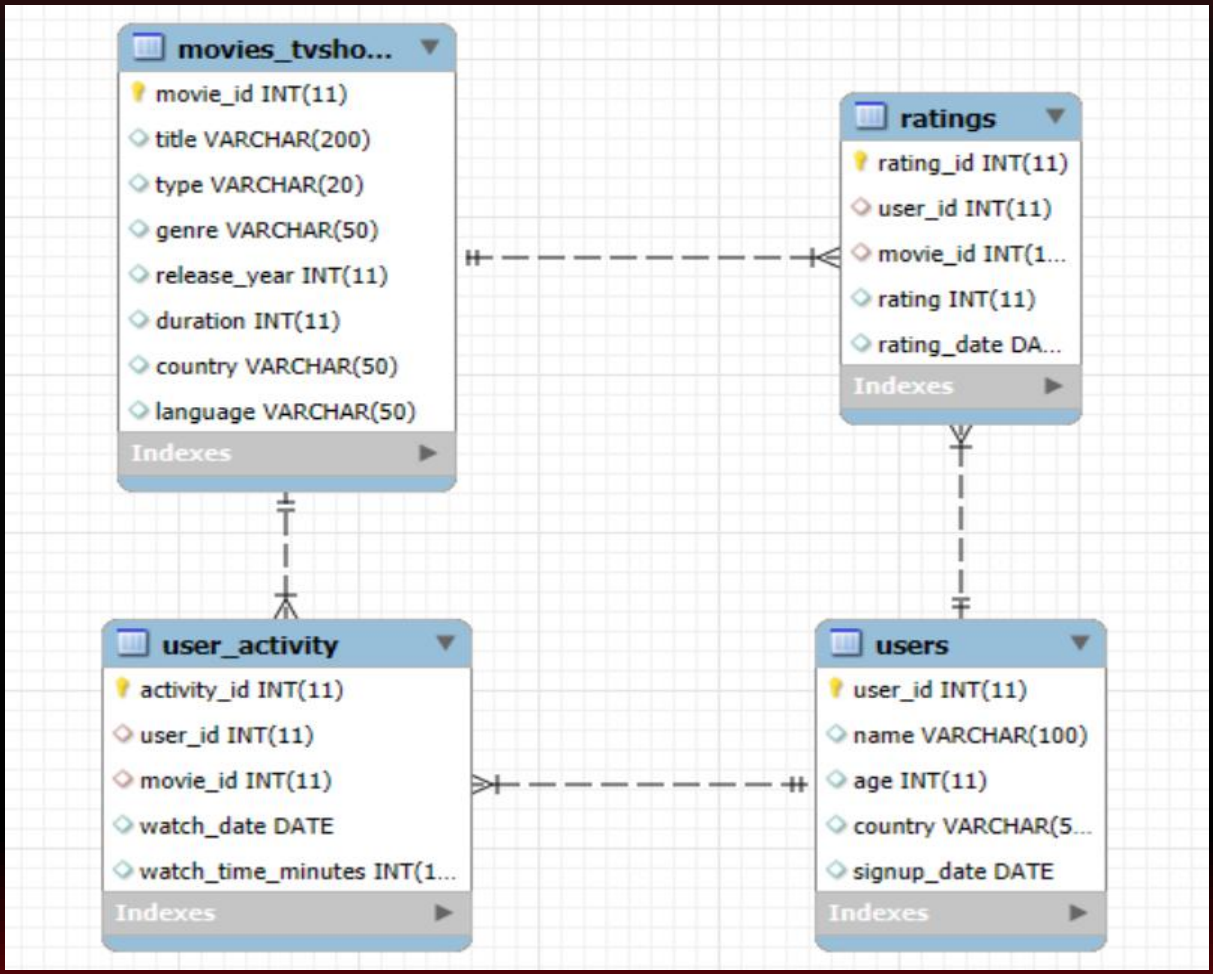
7 Which content is most watched?
- Identified the top 5 most-watched movies/shows.
- Found movies with highest total watch minutes.

8 Which users watched but did not rate any movie?
- Identified users with zero ratings (left join use case).

9 Which movies are being watched but getting poor ratings?
- Movies with average rating below 3 (quality issues or mismatch).

10 Which movies are long vs short, and how do users respond?
- Duration-based filtering.
- Matching long movies with high ratings.

ER Diagram


movies_tvsho...
movie_id INT(11)
title VARCHAR(200)
type VARCHAR(20)
genre VARCHAR(50)
release_year INT(11)
duration INT(11)
country VARCHAR(50)
language VARCHAR(50)
Indexes
ratings
rating_id INT(11)
user_id INT(11)
movie_id INT(1...
rating INT(11)
rating_date DA...
Indexes
user_activity
activity_id INT(11)
user_id INT(11)
movie_id INT(11)
watch_date DATE
watch_time_minutes INT(1...
Indexes
users
user_id INT(11)
name VARCHAR(100)
age INT(11)
country VARCHAR(5...
signup_date DATE
Indexes

Dataset Description

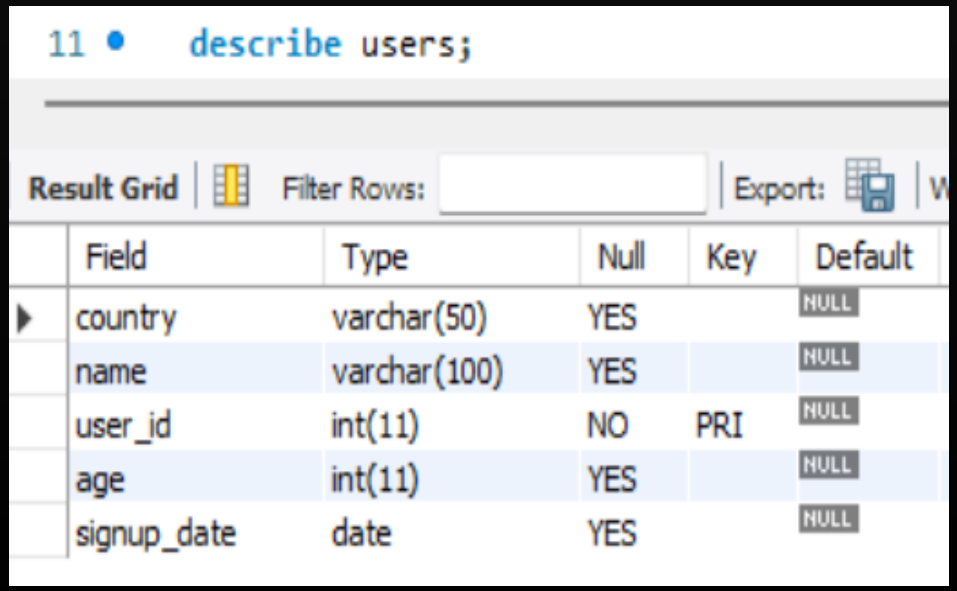
We used **4 related tables:**

- **users** — user_id, name, *age*, country, signup_date
- **movies_tvshows** — movie_id, title, type, genre, release_year, duration, country, language
- **user_activity** — activity_id, user_id, movie_id, watch_date, watch_time_minutes
- **ratings** — rating_id, user_id, movie_id, rating, rating_date

Structure of table

1. Users Table

The **Users** table stores demographic and registration details of each platform user.
It contains fields such as **user_id (PK),** name, age, country and signup_date, helping identify user profiles and analyze demographic-based behavior patterns.



Field	Type	Null	Key	Default
country	varchar(50)	YES		NULL
name	varchar(100)	YES		NULL
user_id	int(11)	NO	PRI	NULL
age	int(11)	YES		NULL
signup_date	date	YES		NULL

Columns:

user_id (Primary Key) — Unique identifier for every user
name — User's name
age — Age of the user
country — Country of residence
signup_date — Date when the user registered on the platform

Structure of table

2. Movies_TVShows Table

The **Movies_TVShows** table contains the catalog of entertainment content available on the platform.

It includes **movie_id (PK)**, title, type, genre, release year, duration, production country and language, enabling content-based analysis such as genre performance, duration analysis, and type-wise trends.

```
23 •  describe user_activity;
```

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default
activity_id	int(11)	NO	PRI	NULL
user_id	int(11)	YES	MUL	NULL
movie_id	int(11)	YES	MUL	NULL
watch_date	date	YES		NULL
watch_time_mi...	int(11)	YES		NULL

Columns:

- **movie_id (Primary Key)** — Unique identifier for each movie or TV show
- **title** — Title of the content
- **type** — Movie or TV Show
- **genre** — Category such as Action, Comedy, Adventure, Drama, etc.
- **release_year** — Year the content was released
- **duration_minutes** — Duration of the content in minutes
- **country** — Country where content was produced
- **language** — Language of the movie/series

Structure of table

3. User_Activity Table

The **User_Activity** table captures all watch events performed by users on the platform. It contains **activity_id (PK)**, user_id, movie_id, watch_date and watch_time, allowing analysis of user engagement, viewing patterns and overall platform consumption.

```
11 •  describe movies_tvshows;
```

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default
movie_id	int(11)	NO	PRI	NULL
title	varchar(200)	YES		NULL
type	varchar(20)	YES		NULL
genre	varchar(50)	YES		NULL
release_year	int(11)	YES		NULL
duration	int(11)	YES		NULL
country	varchar(50)	YES		NULL
language	varchar(50)	YES		NULL

Columns:

- **activity_id (Primary Key)** — Unique identifier for every watch activity
- **user_id (Foreign Key → Users)** — User who watched the content
- **movie_id (Foreign Key → Movies_TVShows)** — Content being watched
- **watch_date** — Date of viewing
- **watch_time_minutes** — Total minutes watched

Structure of table

4. Ratings Table

The **Ratings** table stores feedback given by users on movies and TV shows.
It includes **rating_id (PK),** user_id, movie_id, rating score and rating date, enabling evaluation of content popularity, quality, and user satisfaction.

```
23 •  describe ratings;
```

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default
rating_id	int(11)	NO	PRI	NULL
user_id	int(11)	YES	MUL	NULL
movie_id	int(11)	YES	MUL	NULL
rating	int(11)	YES		NULL
rating_date	date	YES		NULL

Columns:

- **rating_id (Primary Key)** — Unique rating entry
- **user_id (Foreign Key → Users)** — User giving the rating
- **movie_id (Foreign Key → Movies_TVShows)** — Movie/Show being rated
- **rating** — Rating value (1–5 stars)
- **rating_date** — Date when rating was submitted

BASIC QUERIES

"Foundational Understanding of the Data"

1. List of all users from India.

```sql
-- 1. List all users from India.
SELECT *
FROM users
WHERE country = 'India';
```

user_id	name	age	country	signup_date
1	Aarav	23	India	2022-01-05
3	Kabir	34	India	2022-01-12
5	Rohan	27	India	2022-01-20
8	Aisha	25	India	2022-02-05
10	Saanvi	28	India	2022-02-15
13	Arjun	30	India	2022-03-05
15	Reyansh	26	India	2022-03-12
17	Vihaan	32	India	2022-03-25
19	Yash	38	India	2022-04-05
21	Krisha	29	India	2022-04-15
23	Meera	20	India	2022-04-25
25	Sara	23	India	2022-05-05
27	Riya	33	India	2022-05-12
29	Tara	21	India	2022-05-20
NULL	NULL	NULL	NULL	NULL

Output: All users whose country is India.

Conclusion: Identifies the Indian user base, useful for market segmentation.

2. All Movie titles released after 2018.

```sql
-- 2. Show all Movie titles released after 2018.
SELECT title, release_year
FROM movies_tvshows
WHERE release_year > 2018;
```

title	release_year
Midnight Tales	2020
Broken Wings	2019
Fear Night	2021
Soul Ride	2022
Crypto War	2023
Street Beats	2021
Final Laugh	2022
Night Vision	2020
Love Strings	2019
Sky High	2023
Urban Chaos	2022
Dark Woods	2021
Moon Wave	2023
Royal Blood	2020
Desi Diaries	2022
Broken Code	2019
Night Runner	2021
Galaxy Quest	2023
Living Legends	2020

Output: Movies released after 2018.

Conclusion: Helps analyze recent and trending content.

3. Display of the first 5 TV Shows from the dataset.

```sql
-- 3. Display the first 5 TV Shows from the dataset.
SELECT title, type
FROM movies_tvshows
WHERE type = 'TV Show'
LIMIT 5;
```

	title	type
▶	Midnight Tales	TV Show
	Soul Ride	TV Show
	Street Beats	TV Show
	Love Strings	TV Show
	Urban Chaos	TV Show

Output: First 5 TV show records.

Conclusion: Quick snapshot of available TV content.

4. Retrieving unique countries where users are from.

```sql
-- 4. Retrieve unique countries where users are from.
SELECT DISTINCT country
FROM users;
```

	country
▶	India
	USA
	UK
	Canada
	Australia

Output: List of distinct user countries.

Conclusion: Shows global reach and user diversity.

FILTERING & SORTING

"User & Content Segmentation Insights"

5. All Action movies sorted by duration (longest first).

```sql
-- 5. Show all Action movies sorted by duration (longest first).
SELECT title, duration
FROM movies_tvshows
WHERE genre = 'Action'
ORDER BY duration DESC;
```

title	duration
Sky High	140
Storm Hunter	132
Wild Chase	130
Crypto War	125
Silent Storm	120
Night Runner	49
Urban Chaos	48

Output: Longest action movies first.

Conclusion: Useful for understanding long-form action content popularity.

6. List of users aged between 20 and 30.

```sql
-- 6. List users aged between 20 and 30.
SELECT user_id, name, age
FROM users
WHERE age BETWEEN 20 AND 30;
```

user_id	name	age
1	Aarav	23
2	Mia	29
4	Zara	22
5	Rohan	27
8	Aisha	25
10	Saanvi	28
12	Sophia	21
13	Arjun	30

Output: Users between 20–30 years.

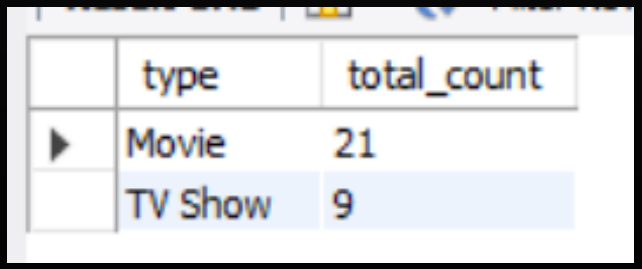
Conclusion: Helps identify young adult audience segment.

AGGREGATION (GROUP BY)

"Trend & Category-Level Insights"

7. Count of how many movies vs TV shows are in the dataset.

```sql
-- 7. Count how many movies vs TV shows are in the dataset.
SELECT type, COUNT(*) AS total_count
FROM movies_tvshows
GROUP BY type;
```



type	total_count
Movie	21
TV Show	9

Output: Users between 20–30 years.

Conclusion: Helps identify young adult audience segment.

8. The average movie duration by genre.

```sql
-- 8. Find the average movie duration by genre.
SELECT genre, AVG(duration) AS avg_duration
FROM movies_tvshows
GROUP BY genre;
```

genre	avg_duration
Action	106.2857
Comedy	78.7500
Documentary	41.5000
Drama	85.4000
Horror	109.0000
Romance	96.0000
Sci-Fi	87.5000
Thriller	109.0000

Output: Average duration for each genre.

Conclusion: Helps identify genre-wise content length trends.

9. Countries having more than 3 registered users

```sql
SELECT country, COUNT(*) AS total_users
FROM users
GROUP BY country
HAVING COUNT(*) > 3;
```

country	total_users
Canada	4
India	14
UK	4
USA	6

Output: Countries with more than 3 users.

Conclusion: Highlights major active user regions.

JOIN QUERIES

"Behavioral & Interaction Insights"

10. All user name, movie title, and watch time for each viewing activity.

```sql
SELECT u.name, m.title, a.watch_time_minutes
FROM user_activity a
INNER JOIN users u ON a.user_id = u.user_id
INNER JOIN movies_tvshows m ON a.movie_id = m.movie_id;
```

name	title	watch_time_minutes
Aarav	Fear Night	45
Mia	Final Laugh	60
Kabir	Soul Ride	40
Zara	Happy Hour	90
Rohan	Laugh Factory	55
Emma	Street Beats	70
Noah	Silent Storm	80
Aisha	Sky High	65
Liam	Wild Chase	100
Saanvi	Crazy Love	75
Oliver	Crypto War	50
Sophia	Midnight Tales	42
Arjun	Night Vision	110
Isabella	Hidden Truth	95
Reya...	Desert Bloom	85
Charl...	Living Legends	60
Vihaan	Open Heart	55
Evelyn	Dark Woods	40
Yash	Storm Hunter	120

Output: Every watch entry with user and content.

Conclusion: Connects users with content they watched for behavior analysis.

11. List of all users and the movies they have rated (include users with no ratings).

```
SELECT u.user_id, u.name, r.movie_id, r.rating
FROM users u
LEFT JOIN ratings r ON u.user_id = r.user_id;
```

user_id	name	movie_id	rating
1	Aarav	5	4
2	Mia	12	5
3	Kabir	7	3
4	Zara	18	4
5	Rohan	3	2
6	Emma	10	5
7	Noah	1	4
8	Aisha	15	3
9	Liam	6	5
10	Saanvi	8	4
11	Oliver	9	2
12	Sophia	2	4
13	Arjun	13	5
14	Isabella	20	4
15	Reya...	11	3
16	Charl...	30	4
17	Vihaan	28	5
18	Evelyn	19	3
19	Yash	25	5

Output: All users, even if they have no rating.

Conclusion: Helps detect inactive raters & engagement gaps.

12. Number of times each movie was watched.

```sql
-- 12. Get number of times each movie was watched.
SELECT m.title, COUNT(a.activity_id) AS total_views
FROM movies_tvshows m
LEFT JOIN user_activity a ON m.movie_id = a.movie_id
GROUP BY m.title;
```

title	total_views
Broken Code	1
Broken Wings	1
Crazy Love	1
Crypto War	1
Dark Woods	1
Desert Bloom	1
Desi Diaries	1
Fear Night	1
Final Laugh	1
Galaxy Quest	1
Happy Hour	1
Hidden Truth	1
Laugh Factory	1
Living Legends	1
Love Strings	1
Melody Lane	1
Midnight Tales	1
Mind Games	1
Moon Wave	1

Output: Total views per movie.

Conclusion: Shows content popularity and demand.

13. The movies with above-average duration.

```sql
-- 13. Find movies with above-average duration.
SELECT title, duration
FROM movies_tvshows
WHERE duration >
    (SELECT AVG(duration) FROM movies_tvshows);
```

title	duration
Silent Storm	120
Laugh Factory	95
Broken Wings	110
Fear Night	100
Wild Chase	130
Crazy Love	105
Crypto War	125
Desert Bloom	115
Night Vision	115
Sky High	140
Mind Games	100
Dark Woods	112
Hidden Truth	107
Melody Lane	102
Royal Blood	118
Storm Hunter	132
Broken Code	110
Open Heart	108
Galaxy Quest	125

Output: Movies longer than overall average.

Conclusion: Identifies long-duration premium content.

14. List of users who have given rating of 5.

```sql
-- 14. List users who have given a rating of 5.
SELECT name
FROM users
WHERE user_id IN (
    SELECT user_id
    FROM ratings
    WHERE rating = 5
);
```

name
Mia
Emma
Liam
Arjun
Vihaan
Yash
Henry
Riya
Aiden

Output: Users who gave perfect ratings.

Conclusion: Identifies highly satisfied or expressive users.

15. The most-watched movie (highest total watch time).

```sql
SELECT title
FROM movies_tvshows
WHERE movie_id = (
    SELECT movie_id
    FROM user_activity
    GROUP BY movie_id
    ORDER BY SUM(watch_time_minutes) DESC
    LIMIT 1
);
```

	title
▶	Storm Hunter

Output: Movie with highest viewing minutes.

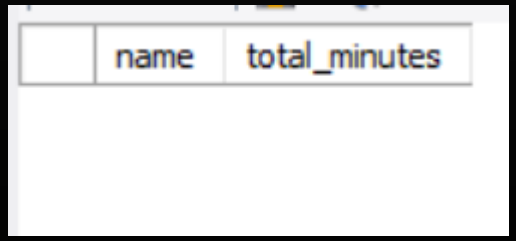
Conclusion: Helps determine the platform's top-performing content.

ADVANCED AGGREGATIONS + HAVING FUNCTIONS

"High-Value User & Performance Insights"

16. Find users who spent more than 200 minutes watching content.

```sql
-- 16. Find users who spent more than 200 minutes watching content.
SELECT u.name, SUM(a.watch_time_minutes) AS total_minutes
FROM users u
JOIN user_activity a ON u.user_id = a.user_id
GROUP BY u.user_id
HAVING SUM(a.watch_time_minutes) > 200;
```



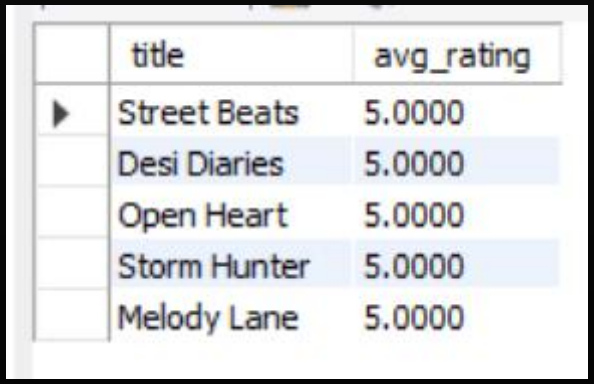
	name	total_minutes

Output: Power users with high watch hours.

Conclusion: Useful for targeted engagement or premium plans.

17. Top 5 highest-rated movies (average rating).

```sql
-- 17. Top 5 highest-rated movies (average rating).
SELECT m.title, AVG(r.rating) AS avg_rating
FROM movies_tvshows m
JOIN ratings r ON m.movie_id = r.movie_id
GROUP BY m.movie_id
ORDER BY avg_rating DESC
LIMIT 5;
```



	title	avg_rating
▶	Street Beats	5.0000
	Desi Diaries	5.0000
	Open Heart	5.0000
	Storm Hunter	5.0000
	Melody Lane	5.0000

Output: Top 5 movies by ratings.

Conclusion: Identifies best-performing content based on quality feedback.

WINDOW FUNCTIONS

"Rankings, Trends & Time-Series Insights"

&

SUBQUERIES

"Comparative & Logical Insights"

18. Rank movies based on average rating.

```sql
-- 18. Rank movies based on average rating.
SELECT
    m.title,
    AVG(r.rating) AS avg_rating,
    RANK() OVER (ORDER BY AVG(r.rating) DESC) AS rating_rank
FROM movies_tvshows m
JOIN ratings r ON m.movie_id = r.movie_id
GROUP BY m.movie_id;
```

title	avg_rating	rating_rank
Silent Storm	4.0000	10
Midnight Tales	4.0000	10
Laugh Factory	2.0000	28
Broken Wings	4.0000	10
Fear Night	4.0000	10
Wild Chase	5.0000	1
Soul Ride	3.0000	21
Crazy Love	4.0000	10
Crypto War	2.0000	28
Street Beats	5.0000	1
Desert Bloom	3.0000	21
Final Laugh	5.0000	1
Night Vision	5.0000	1
Love Strings	5.0000	1
Sky High	3.0000	21
Mind Games	3.0000	21
Urban Chaos	2.0000	28
Happy Hour	4.0000	10
Dark Woods	3.0000	21

Output: Ranked list of movies by rating.

Conclusion: Offers advanced comparison of content quality.

19. Running total of watch time (time-series).

```sql
SELECT
    activity_id,
    watch_date,
    watch_time_minutes,
    SUM(watch_time_minutes) OVER (ORDER BY watch_date) AS running_total
FROM user_activity;
```

activity_id	watch_date	watch_time_minutes	running_total
1	2023-01-05	45	45
2	2023-01-06	60	105
3	2023-01-07	40	145
4	2023-01-09	90	235
5	2023-01-10	55	290
6	2023-01-11	70	360
7	2023-01-12	80	440
8	2023-01-15	65	505
9	2023-01-16	100	605
10	2023-01-18	75	680
11	2023-01-20	50	730
12	2023-01-21	42	772
13	2023-01-22	110	882
14	2023-01-23	95	977
15	2023-01-25	85	1062
16	2023-01-27	60	1122
17	2023-01-29	55	1177
18	2023-02-01	40	1217
19	2023-02-02	120	1337

Output: Cumulative watch time over dates.

Conclusion: Shows platform engagement growth over time.

20. Find the top 3 most popular genres by total watch time.

```sql
-- 20. Find the top 3 most popular genres by total watch time.
SELECT genre, SUM(a.watch_time_minutes) AS total_watch_time
FROM movies_tvshows m
JOIN user_activity a ON m.movie_id = a.movie_id
GROUP BY genre
ORDER BY total_watch_time DESC
LIMIT 3;
```

genre	total_watch_time
Action	532
Drama	337
Romance	323

Output: Top 3 genres by watch time.

Conclusion: Identifies genres that drive maximum viewer engagement.

Conclusion

- **Movies dominate the platform,** and users show higher engagement with long-duration and recently released content, indicating strong demand for fresh and high-quality titles.

- **India and the US are the highest contributors** in both user base and content production, making them the most strategic markets for content investment and targeted campaigns.

- **Action, Drama, and Comedy genres show the highest watch time and ratings,** suggesting that these categories should receive priority in content acquisition and marketing.

- **A portion of users show low engagement or no activity**, highlighting the need for personalized recommendations, re-engagement strategies, and retention-focused campaigns.

Business Recommendations

1. Focus Marketing on High-Engagement Regions

- Users from certain countries (like India) dominate the platform—targeted regional campaigns can significantly boost user acquisition and retention.

2. Expand High-Performing Genres & Categories

- Action, Drama, and Comedy receive the highest watch-time.
  Invest in more content and exclusive releases in these genres to increase user engagement.

3. Strengthen Personalized Recommendations

- User behavior patterns show diverse viewing habits—improving recommendation algorithms can increase watch time and user satisfaction.

4. Improve Retention With Re-Engagement Strategies

- A portion of users remain inactive or show low watch-time.
  Push targeted notifications, new-content alerts, and personalized reminders to reduce churn.

5. Optimize Content Release Timing

- Peak viewing hours identified can guide the ideal timing for releasing new content, promotions, and marketing campaigns for maximum reach.