# User-Oriented Queries:

1. **Retrieve Recommended Content**: Suggests new content based on user's high ratings and excludes already watched content.

⇒

SELECT DISTINCT c.title,

c.age\_rating,

c.origin\_country,

c.release\_date,

user\_id

FROM contents c

JOIN ratings r ON c.content\_id = r.content\_id

WHERE r.ratings >= 4

AND c.content\_id NOT IN (SELECT content\_id FROM watch\_history WHERE user\_id = r.user\_id)

ORDER BY c.release\_date DESC;

1. **Total Time Spent on Movies vs. Series**: Calculates total watch time for movies versus series for a user.

⇒

SELECT content\_type,

SUM(wh.duration) AS total\_watch\_time

FROM watch\_history wh

JOIN contents c ON wh.content\_id = c.content\_id

JOIN (

SELECT content\_id, 'Movie' AS content\_type

FROM movies

UNION

SELECT content\_id, 'Series' AS content\_type

FROM series

) AS content\_types ON c.content\_id = content\_types.content\_id

WHERE wh.user\_id = 101

GROUP BY content\_type;

1. **Retrieve Playlist**: Displays the user's playlist with content titles, sorted by last updated date.

⇒SELECT pl.playlist\_type,

c.title

FROM playlist pl

JOIN playlist\_content pc ON pl.playlist\_id = pc.playlist\_id

JOIN contents c ON pc.content\_id = c.content\_id

WHERE pl.user\_id = 101

ORDER BY pl.last\_updated DESC;

1. **Retrieve Trending Content**: Identifies popular content based on high views and average ratings.

⇒

SELECT c.title,

m.total\_views,

AVG(r.ratings) AS avg\_rating

FROM contents c

JOIN metrics m ON c.content\_id = m.content\_id

JOIN ratings r ON c.content\_id = r.content\_id

GROUP BY c.title, m.total\_views

HAVING AVG(r.ratings) >= 4 AND m.total\_views > 1000

ORDER BY m.total\_views DESC, avg\_rating DESC;

1. **Content in Multiple Languages**: Shows content available in preferred languages for accessibility.

⇒

SELECT c.title,

cl.languages

FROM content\_language cl

JOIN contents c ON cl.content\_id = c.content\_id

WHERE cl.languages IN ('English', 'Spanish')

ORDER BY c.release\_date DESC;

1. **Popular Content in Specific Genre**: Finds popular content within a specific genre with high ratings and views.

⇒

SELECT c.title,

g.genre,

m.total\_views,

AVG(r.ratings) AS avg\_rating

FROM contents c

JOIN content\_genre g ON c.content\_id = g.content\_id

JOIN metrics m ON c.content\_id = m.content\_id

JOIN ratings r ON c.content\_id = r.content\_id

WHERE g.genre = 'Action'

GROUP BY c.title, g.genre, m.total\_views

HAVING AVG(r.ratings) >= 4 AND m.total\_views > 1000

ORDER BY m.total\_views DESC, avg\_rating DESC;

1. **Retrieve Free or Premium Content**: Lists content based on its access type (Free or Premium).

⇒

SELECT c.title,

c.access\_type,

c.age\_rating,

c.origin\_country

FROM contents c

WHERE c.access\_type IN ('Free', 'Premium')

ORDER BY c.release\_date DESC;

1. **Subscription Plan Comparison**: Compares subscription plans based on user count and details.

⇒

SELECT s.subscription\_plan,

s.amount,

s.duration,

COUNT(usp.user\_id) AS total\_users

FROM subscriptions s

LEFT JOIN user\_subscribed\_payment usp ON s.subs\_id = usp.subs\_id

GROUP BY s.subscription\_plan, s.amount, s.duration

ORDER BY total\_users DESC;

1. **View Subscription and Payment Details**: Provides details of a user’s subscription and payment history.

⇒

SELECT u.user\_id,

u.user\_name,

s.subscription\_plan,

s.amount AS subscription\_amount,

s.duration,

p.payment\_method,

p.amount\_paid,

p.date\_of\_pay AS payment\_date,

usp.renewal\_date

FROM users u

JOIN user\_subscribed\_payment usp ON u.user\_id = usp.user\_id

JOIN subscriptions s ON usp.subs\_id = s.subs\_id

JOIN payment p ON usp.pay\_id = p.payment\_id

ORDER BY u.user\_id;

1. **Most Watched Genre**: Identifies the most-watched genre by the user within a specified time frame.

⇒

SELECT genre,

COUNT(\*) AS total\_watch\_count

FROM watch\_history wh

JOIN contents c ON wh.content\_id = c.content\_id

JOIN content\_genre cg ON c.content\_id = cg.content\_id

WHERE wh.user\_id = 101 -- Replace with your user\_id

AND wh.watch\_date BETWEEN '2024-01-01' AND '2024-12-31'

GROUP BY genre

ORDER BY total\_watch\_count DESC;

1. **Unrated Watched Content**: Lists content a user has watched but not rated.

⇒

SELECT wh.content\_id,

c.title,

wh.watch\_date

FROM watch\_history wh

JOIN contents c ON wh.content\_id = c.content\_id

LEFT JOIN ratings r ON wh.user\_id = r.user\_id AND wh.content\_id = r.content\_id

WHERE wh.user\_id = 101

AND r.content\_id IS NULL

ORDER BY wh.watch\_date DESC;

1. **Subscription Plans and Available Content**: Compares plans based on available content in specific genres.

⇒

SELECT s.subscription\_plan,

COUNT(c.content\_id) AS available\_content

FROM subscriptions s

JOIN user\_subscribed\_payment usp ON s.subs\_id = usp.subs\_id

JOIN playlist pl ON usp.user\_id = pl.user\_id

JOIN playlist\_content pc ON pl.playlist\_id = pc.playlist\_id

JOIN contents c ON pc.content\_id = c.content\_id

JOIN content\_genre cg ON c.content\_id = cg.content\_id

WHERE cg.genre IN ('Action', 'Comedy')

GROUP BY s.subscription\_plan

ORDER BY available\_content DESC;

# Administrator-Oriented Queries:

1. **Active Users with Subscription Details**: Lists active users and their subscription plans.

⇒

SELECT u.user\_id, u.user\_name, s.subscription\_plan, s.amount, s.duration

FROM users u

JOIN user\_subscribed\_payment usp ON u.user\_id = usp.user\_id

JOIN subscriptions s ON usp.subs\_id = s.subs\_id

WHERE usp.renewal\_date >= CURRENT\_DATE;

1. **Expiring Subscriptions**: Find users whose subscriptions are expiring soon.

⇒

SELECT u.user\_id, u.user\_name, usp.renewal\_date

FROM users u

JOIN user\_subscribed\_payment usp ON u.user\_id = usp.user\_id

WHERE (usp.renewal\_date - CURRENT\_DATE) BETWEEN 0 AND 7;

1. **Users with Multiple Devices**: Identifies users with more than one linked device.

⇒

SELECT u.user\_id, u.user\_name, COUNT(ud.device\_id) AS device\_count

FROM users u

JOIN user\_device ud ON u.user\_id = ud.user\_id

GROUP BY u.user\_id, u.user\_name

HAVING COUNT(ud.device\_id) > 1;

1. **Users on Specific Subscription Plan**: Retrieves users with a specific subscription plan.

⇒

SELECT u.user\_id, u.user\_name

FROM users u

JOIN user\_subscribed\_payment usp ON u.user\_id = usp.user\_id

JOIN subscriptions s ON usp.subs\_id = s.subs\_id

WHERE s.subscription\_plan = 'Premium';

1. **Top 5 Most-Watched Content**: Lists the most-watched movies or series by view count.

⇒

SELECT c.title, COUNT(wh.user\_id) AS view\_count

FROM contents c

JOIN watch\_history wh ON c.content\_id = wh.content\_id

GROUP BY c.content\_id, c.title

ORDER BY view\_count DESC

LIMIT 5;

1. **Unwatched Content**: Identifies content not watched by any user.

⇒

SELECT c.content\_id, c.title

FROM contents c

LEFT JOIN watch\_history wh ON c.content\_id = wh.content\_id

WHERE wh.content\_id IS NULL;

1. **Hidden Gems**: Finds content with high ratings but low viewership.

⇒

SELECT c.title, m.total\_views, AVG(r.ratings) AS average\_rating

FROM contents c

JOIN metrics m ON c.content\_id = m.content\_id

JOIN ratings r ON c.content\_id = r.content\_id

GROUP BY c.content\_id, c.title, m.total\_views

HAVING m.total\_views < 5000 AND AVG(r.ratings) > 3.5

ORDER BY average\_rating DESC

LIMIT 10;

1. **Monthly Subscription Revenue**: Calculates revenue generated from subscriptions monthly.

⇒

SELECT EXTRACT(YEAR FROM p.date\_of\_pay) AS year,

EXTRACT(MONTH FROM p.date\_of\_pay) AS month,

SUM(p.amount\_paid) AS monthly\_revenue

FROM payment p

GROUP BY year, month

ORDER BY year DESC, month DESC;

1. **Payment Methods and User Count**: Analyzes payment method usage by distinct users.

⇒

SELECT p.payment\_method, COUNT(DISTINCT usp.user\_id) AS user\_count

FROM payment p

JOIN user\_subscribed\_payment usp ON p.payment\_id = usp.pay\_id

GROUP BY p.payment\_method;

1. **Most Watched Genres**: Determines the most popular genres.

⇒

SELECT cg.genre, COUNT(wh.content\_id) AS view\_count

FROM content\_genre cg

JOIN watch\_history wh ON cg.content\_id = wh.content\_id

GROUP BY cg.genre

ORDER BY view\_count DESC;

1. **Content Watched by Age Rating**: Shows content watched by users segmented by age rating.

⇒

SELECT u.user\_name, c.title, c.age\_rating, wh.watch\_date

FROM users u

JOIN watch\_history wh ON u.user\_id = wh.user\_id

JOIN contents c ON wh.content\_id = c.content\_id

WHERE c.age\_rating = 'A';

1. **Profitable Plans by Region**: Identifies the most profitable subscription plans across regions.

⇒

WITH regional\_performance AS (

SELECT u.user\_id,

u.nationality,

s.subscription\_plan,

SUM(p.amount\_paid) AS total\_revenue

FROM users u

JOIN user\_subscribed\_payment usp ON u.user\_id = usp.user\_id

JOIN payment p ON usp.pay\_id = p.payment\_id

JOIN subscriptions s ON usp.subs\_id = s.subs\_id

GROUP BY u.user\_id, u.nationality, s.subscription\_plan

)

SELECT nationality,

subscription\_plan,

SUM(total\_revenue) AS region\_revenue

FROM regional\_performance

GROUP BY nationality, subscription\_plan

ORDER BY region\_revenue DESC;

1. **Movies vs. Series Analysis**: Compares movies and series by revenue, views, and average watch time.

⇒

WITH regional\_performance AS (

SELECT u.user\_id,

u.nationality,

s.subscription\_plan,

SUM(p.amount\_paid) AS total\_revenue

FROM users u

JOIN user\_subscribed\_payment usp ON u.user\_id = usp.user\_id

JOIN payment p ON usp.pay\_id = p.payment\_id

JOIN subscriptions s ON usp.subs\_id = s.subs\_id

GROUP BY u.user\_id, u.nationality, s.subscription\_plan

)

SELECT nationality,

subscription\_plan,

SUM(total\_revenue) AS region\_revenue

FROM regional\_performance

GROUP BY nationality, subscription\_plan

ORDER BY region\_revenue DESC;

# Advertiser-Oriented Queries:

1. **High-Rated Content with Viewer Diversity**: Finds high-rated content with diverse audience nationalities.

⇒

SELECT c.title,

AVG(r.ratings) AS avg\_rating,

COUNT(DISTINCT u.nationality) AS unique\_nationalities

FROM ratings r

JOIN contents c ON r.content\_id = c.content\_id

JOIN watch\_history wh ON r.content\_id = wh.content\_id

JOIN users u ON wh.user\_id = u.user\_id

GROUP BY c.title

HAVING AVG(r.ratings) > 4

AND COUNT(DISTINCT u.nationality) > 3

ORDER BY unique\_nationalities DESC;

1. **Re-Engagement Success**: Tracks content with high re-watch frequency.

⇒

SELECT c.title,

COUNT(DISTINCT wh.user\_id) AS unique\_users,

COUNT(wh.content\_id) AS total\_rewatch\_count

FROM watch\_history wh

JOIN contents c ON wh.content\_id = c.content\_id

GROUP BY c.title

HAVING COUNT(wh.content\_id) > 2

ORDER BY total\_rewatch\_count DESC;

1. **Subscription Plans for Ads**: Analyzes the success of subscription plans for targeted advertising.

⇒

SELECT s.subscription\_plan,

COUNT(usp.user\_id) AS total\_users,

SUM(wh.duration) AS total\_watch\_time

FROM user\_subscribed\_payment usp

JOIN subscriptions s ON usp.subs\_id = s.subs\_id

JOIN watch\_history wh ON usp.user\_id = wh.user\_id

GROUP BY s.subscription\_plan

ORDER BY total\_watch\_time DESC;

1. **Content with Diverse Language Support**: Highlights content available in multiple languages.

⇒

SELECT c.title,

COUNT(DISTINCT cl.languages) AS language\_count

FROM content\_language cl

JOIN contents c ON cl.content\_id = c.content\_id

GROUP BY c.title

HAVING COUNT(DISTINCT cl.languages) > 3 -- Content supporting multiple languages

ORDER BY language\_count DESC;

# Producer-Oriented Queries:

1. **Audience Demographics**: Shows viewership by nationality for each content.

⇒

SELECT

c.title,

u.nationality,

COUNT(\*) AS viewer\_count

FROM

watch\_history wh

JOIN

users u ON wh.user\_id = u.user\_id

JOIN

contents c ON wh.content\_id = c.content\_id

GROUP BY

c.title, u.nationality;

1. **Top-Rated Content**: Lists the highest-rated content.

⇒

SELECT

c.title,

AVG(r.ratings) AS average\_rating

FROM

ratings r

JOIN

contents c ON r.content\_id = c.content\_id

GROUP BY

c.title

ORDER BY

average\_rating DESC

LIMIT 10;

1. **Viewership by Age Rating**: Segments content viewership by age rating.

⇒

SELECT

c.age\_rating,

COUNT(wh.content\_id) AS total\_views

FROM

contents c

JOIN

watch\_history wh ON c.content\_id = wh.content\_id

GROUP BY

c.age\_rating;

# Marketing-Oriented Queries:

1. **Content Popularity by Genre**: Determines popular genres based on view count.

⇒

SELECT

cg.genre,

COUNT(wh.content\_id) AS total\_views

FROM

content\_genre cg

JOIN

watch\_history wh ON cg.content\_id = wh.content\_id

GROUP BY

cg.genre

ORDER BY

total\_views DESC;

1. **Language Preferences**: Tracks viewership by language preferences.

⇒

SELECT

cl.languages,

COUNT(wh.content\_id) AS total\_views

FROM

content\_language cl

JOIN

watch\_history wh ON cl.content\_id = wh.content\_id

GROUP BY

cl.languages

ORDER BY

total\_views DESC;

1. **Total Revenue by Actor:** Identifies the total revenue generated by each actor across all their associated content, ranked in descending order of revenue.

⇒

SELECT a.actor\_name,

SUM(m.total\_revenue) AS total\_revenue

FROM actor a

JOIN content\_artist ca ON a.artist\_id = ca.artist\_id

JOIN metrics m ON ca.content\_id = m.content\_id

GROUP BY

a.actor\_name

ORDER BY

total\_revenue DESC;

# content strategist Oriented-Queries:

1. **Top Earning Content by Genre and Language with Viewership Data:**

Identifies the top content by genre and language based on total revenue and viewership metrics.

**⇒**

SELECT

cg.genre,

cl.languages,

c.title,

SUM(m.total\_revenue) AS total\_revenue,

SUM(m.total\_views) AS total\_views,

COUNT(wh.user\_id) AS total\_watch\_count

FROM contents c

JOIN metrics m ON c.content\_id = m.content\_id

JOIN content\_genre cg ON c.content\_id = cg.content\_id

JOIN content\_language cl ON c.content\_id = cl.content\_id

LEFT JOIN watch\_history wh ON c.content\_id = wh.content\_id

GROUP BY cg.genre, cl.languages, c.title

HAVING SUM(m.total\_revenue) > 10000

ORDER BY total\_revenue DESC, total\_views DESC

LIMIT 3;

1. **User Subscription Revenue by Nationality with Average Watch Time and Age Rating:**Aggregates subscription revenue, average watch time, and preferred age ratings for users grouped by nationality.

⇒

SELECT

u.nationality,

SUM(s.amount) AS total\_revenue,

AVG(EXTRACT(HOUR FROM wh.duration) \* 60 + EXTRACT(MINUTE FROM wh.duration)) AS avg\_watch\_time\_minutes,

STRING\_AGG(DISTINCT c.age\_rating, ', ') AS preferred\_age\_ratings

FROM

users u

JOIN

user\_subscribed\_payment usp ON u.user\_id = usp.user\_id

JOIN

subscriptions s ON usp.subs\_id = s.subs\_id

JOIN

watch\_history wh ON u.user\_id = wh.user\_id

JOIN

contents c ON wh.content\_id = c.content\_id

GROUP BY

u.nationality

ORDER BY

total\_revenue DESC;

1. **Most Popular Subscription Plan by Genre, Age Rating, and Country:**Determines the most popular subscription plans segmented by genre, age rating, and country based on user counts.

⇒

SELECT

s.subscription\_plan,

cg.genre,

c.age\_rating,

u.nationality,

COUNT(DISTINCT usp.user\_id) AS total\_users

FROM

subscriptions s

JOIN

user\_subscribed\_payment usp ON s.subs\_id = usp.subs\_id

JOIN

users u ON usp.user\_id = u.user\_id

JOIN

watch\_history wh ON u.user\_id = wh.user\_id

JOIN

contents c ON wh.content\_id = c.content\_id

JOIN

content\_genre cg ON c.content\_id = cg.content\_id

GROUP BY

s.subscription\_plan, cg.genre, c.age\_rating, u.nationality

ORDER BY

total\_users DESC

LIMIT 10;