# ONLINE OTT PLATFORM DATABASE PROJECT

Unmesh Varade

BT23CSE215

## ***Queries with their output:***

Q1. Write a query to list all users from the Users table in alphabetical order.

```
SELECT user_id, name, email, city, created_on
FROM Users
ORDER BY name;
```

| user_id | name | email | city | created_on |
|---|---|---|---|---|
| 1 | Aman Verma | aman.v@example.com | Pune | 2024-05-01 10:00:00 |
| 3 | David Smith | david.s@example.co.uk | London | 2023-11-11 08:25:00 |
| 2 | Nina Patel | nina.p@example.com | Bangalore | 2024-07-20 09:30:00 |
| 5 | Rohit Kapoor | rohit.k@example.com | Mumbai | 2025-03-10 11:15:00 |
| 4 | Sara Lee | sara.lee@example.com | Dubai | 2025-01-05 15:45:00 |
| NULL | NULL | NULL | NULL | NULL |

Q2. Write a query to display all movies along with their genres.

```
SELECT m.movie_id, m.title, m.release_year,
    GROUP_CONCAT(g.name SEPARATOR ', ') AS genres
FROM Movies m
JOIN Movie_Genres mg ON m.movie_id = mg.movie_id
JOIN Genres g ON mg.genre_id = g.genre_id
GROUP BY m.movie_id;
```

| movie_id | title | release_year | genres |
|---|---|---|---|
| 1 | The Silent Sea | 2021 | Thriller, Sci-Fi |
| 2 | Midnight Chase | 2019 | Action, Thriller |
| 3 | Love & Pixels | 2023 | Comedy, Romance |
| 4 | Quantum Break | 2020 | Action, Sci-Fi |
| 5 | Laugh Out Loud | 2018 | Comedy |
| 6 | Edge of Tomorrow | 2014 | Action, Sci-Fi |

Q3. Write a query to list all active subscriptions along with remaining days.

```
SELECT s.subscription_id, u.name, s.plan_name, s.end_date,
    DATEDIFF(s.end_date, CURDATE()) AS days_left
FROM Subscriptions s
JOIN Users u ON s.user_id = u.user_id
WHERE s.status='Active';
```

| subscription_id | name | plan_name | end_date | days_left |
|---|---|---|---|---|
| 1 | Aman Verma | Premium | 2025-12-31 | 47 |
| 2 | Nina Patel | Standard | 2025-06-14 | -153 |
| 4 | Sara Lee | Premium | 2026-09-30 | 320 |

Q4. Write a query to calculate the total revenue generated from all payments.

```
SELECT SUM(amount) AS total_revenue
FROM Payments;
```

| total_revenue |
|---|
| 1446.00 |

Q5. Write a query to find the most-watched movie based on total watch sessions.

```
SELECT m.movie_id, m.title, COUNT(w.watch_id) AS watch_sessions
FROM Movies m
LEFT JOIN Watch_History w ON m.movie_id = w.movie_id
GROUP BY m.movie_id
ORDER BY watch_sessions DESC
LIMIT 1;
```

| movie_id | title | watch_sessions |
|---|---|---|
| 3 | Love & Pixels | 2 |

Q6. Write a query to find the top 3 most-watched movies based on total minutes watched.

```
SELECT m.movie_id, m.title, SUM(w.watch_duration_min) AS total_minutes
FROM Movies m
LEFT JOIN Watch_History w ON m.movie_id = w.movie_id
```

```
GROUP BY m.movie_id
ORDER BY total_minutes DESC
LIMIT 3;
```

| movie_id | title | total_minutes |
|---|---|---|
| 4 | Quantum Break | 255 |
| 6 | Edge of Tomorrow | 223 |
| 3 | Love & Pixels | 150 |

Q7. Write a query to find the most popular genre based on watch count.

```
SELECT g.genre_id, g.name, COUNT(w.watch_id) AS watch_count
FROM Genres g
JOIN Movie_Genres mg ON g.genre_id = mg.genre_id
JOIN Watch_History w ON mg.movie_id = w.movie_id
GROUP BY g.genre_id
ORDER BY watch_count DESC
LIMIT 1;
```

| genre_id | name | watch_count |
|---|---|---|
| 2 | Action | 5 |

Q8. Find the top 3 users who watched the most variety of different movies.

```
SELECT u.user_id, u.name,
    COUNT(DISTINCT w.movie_id) AS unique_movies_watched
FROM Users u
JOIN Watch_History w ON u.user_id = w.user_id
GROUP BY u.user_id
ORDER BY unique_movies_watched DESC
LIMIT 3;
```

| user_id | name | unique_movies_watched |
|---|---|---|
| 1 | Aman Verma | 3 |
| 2 | Nina Patel | 3 |
| 3 | David Smith | 1 |

Q9. Write a query to calculate the average rating and rating count for each movie.

```
SELECT m.movie_id, m.title,
    COALESCE(AVG(r.rating),0) AS avg_rating,
    COUNT(r.rating_id) AS rating_count
FROM Movies m
LEFT JOIN Ratings r ON m.movie_id = r.movie_id
GROUP BY m.movie_id
ORDER BY avg_rating DESC;
```

| movie_id | title | avg_rating | rating_count |
|---|---|---|---|
| 1 | The Silent Sea | 5.0000 | 1 |
| 6 | Edge of Tomorrow | 5.0000 | 1 |
| 4 | Quantum Break | 4.0000 | 2 |
| 2 | Midnight Chase | 3.0000 | 1 |
| 3 | Love & Pixels | 0.0000 | 0 |
| 5 | Laugh Out Loud | 0.0000 | 0 |

Q10. Write a query to calculate the total watch time for each user.

```
SELECT u.user_id, u.name,
    COALESCE(SUM(w.watch_duration_min),0) AS total_watch_minutes
FROM Users u
LEFT JOIN Watch_History w ON u.user_id = w.user_id
GROUP BY u.user_id
ORDER BY total_watch_minutes DESC;
```

| user_id | name | total_watch_minutes |
|---|---|---|
| 2 | Nina Patel | 338 |
| 1 | Aman Verma | 280 |
| 3 | David Smith | 125 |
| 4 | Sara Lee | 110 |
| 5 | Rohit Kapoor | 80 |

Q11. Write a query to display movies released in the last 5 years.

```
SELECT movie_id, title, release_year
FROM Movies
WHERE release_year >= (YEAR(CURDATE()) - 5);
```

| movie_id | title | release_year |
|---|---|---|
| 1 | The Silent Sea | 2021 |
| 3 | Love & Pixels | 2023 |
| 4 | Quantum Break | 2020 |
| NULL | NULL | NULL |

Q12. Write a query to compute revenue and subscriber count for each subscription plan.

```
SELECT s.plan_name,
    COALESCE(SUM(p.amount),0) AS plan_revenue,
    COUNT(DISTINCT s.user_id) AS subscribers
FROM Subscriptions s
LEFT JOIN Payments p ON s.subscription_id = p.subscription_id
GROUP BY s.plan_name;
```

| plan_name | plan_revenue | subscribers |
|---|---|---|
| Basic | 149.00 | 1 |
| Premium | 998.00 | 2 |
| Standard | 299.00 | 2 |

Q13. Write a query to list all users who have given a 5-star rating.

```
SELECT u.user_id, u.name, m.title, r.rating
FROM Ratings r
JOIN Users u ON r.user_id = u.user_id
JOIN Movies m ON r.movie_id = m.movie_id
WHERE r.rating = 5;
```

| user_id | name | title | rating |
|---|---|---|---|
| 1 | Aman Verma | The Silent Sea | 5 |
| 4 | Sara Lee | Edge of Tomorrow | 5 |

Q14. Write a query to indicate whether user 1 has rated each movie.

```
SELECT m.movie_id, m.title,
  CASE
    WHEN EXISTS (
      SELECT 1 FROM Ratings r
      WHERE r.movie_id = m.movie_id AND r.user_id = 1
```

```
    )
    THEN 'Rated'
    ELSE 'Not Rated'
  END AS user1_rating_status
FROM Movies m;
```

| movie_id | title | user1_rating_status |
|---|---|---|
| 1 | The Silent Sea | Rated |
| 2 | Midnight Chase | Not Rated |
| 3 | Love & Pixels | Not Rated |
| 4 | Quantum Break | Rated |
| 5 | Laugh Out Loud | Not Rated |
| 6 | Edge of Tomorrow | Not Rated |

Q15. Write a query to calculate the average watch duration for each genre.

```
SELECT g.name AS genre,
    AVG(w.watch_duration_min) AS avg_watch_minutes
FROM Genres g
JOIN Movie_Genres mg ON g.genre_id = mg.genre_id
JOIN Watch_History w ON mg.movie_id = w.movie_id
GROUP BY g.genre_id
ORDER BY avg_watch_minutes DESC;
```

| genre | avg_watch_minutes |
|---|---|
| Action | 120.6000 |
| Sci-Fi | 115.6000 |
| Thriller | 112.5000 |
| Comedy | 76.6667 |
| Romance | 75.0000 |

Q16. Get full user profile of any with active subscription details.

```
SELECT u.user_id, u.name, u.email, u.city,
    s.plan_name, s.start_date, s.end_date, s.status
FROM Users u
LEFT JOIN Subscriptions s ON u.user_id = s.user_id
WHERE u.user_id = 1;
```

| user_id | name | email | city | plan_name | start_date | end_date | status |
|---|---|---|---|---|---|---|---|
| 1 | Aman Verma | aman.v@example.com | Pune | Premium | 2025-01-01 | 2025-12-31 | Active |

Q17. Get last 10 recently watched movies of a user given user id as 1.

```
SELECT m.movie_id, m.title, w.watched_on
FROM Watch_History w
JOIN Movies m ON w.movie_id = m.movie_id
WHERE w.user_id = 1
ORDER BY w.watched_on DESC
LIMIT 10;
```

| movie_id | title | watched_on |
|---|---|---|
| 3 | Love & Pixels | 2025-11-05 20:15:00 |
| 4 | Quantum Break | 2025-10-10 21:30:00 |
| 1 | The Silent Sea | 2025-10-05 20:00:00 |

Q18. List movies with no ratings yet.

```
SELECT m.movie_id, m.title
FROM Movies m
LEFT JOIN Ratings r ON m.movie_id = r.movie_id
WHERE r.rating_id IS NULL;
```

| movie_id | title |
|---|---|
| 3 | Love & Pixels |
| 5 | Laugh Out Loud |