

Capital University of Science and Technology

Department of Software Engineering

|  |
| --- |
| SE2313 – Introduction to Database Systems |

**Member’s Name:** Muhammad Saad Imran, Abdul Waseh, Zohaib Fayyaz.

**Registration Number:** BSE233117, BSE233073, BSE233069.

**Section:** 02.

**Date:** January **13th**,2025.

**Submitted To:** Mam Hina Rashid.

GITHUB LINK

<https://github.com/AbdulWaseh46/MovieRecommendation_DBMS>

**Table of Contents:**

[ **Objective:** 3](#_Toc187617205)

[ **Database Implementation:** 3](#_Toc187617206)

[**1.** **Database Schema:** 3](#_Toc187617207)

[ **Data Population:** 4](#_Toc187617208)

[**1.** **Users Table:** 4](#_Toc187617209)

[**2.** **Movies Table:** 4](#_Toc187617210)

[**3.** **Ratings Table:** 5](#_Toc187617211)

[**4.** **Recommendations Table:** 5](#_Toc187617212)

[**5.** **WatchHistory Table:** 5](#_Toc187617213)

[ **SQL Queries:** 6](#_Toc187617214)

[ **Relational Algebra Queries:** 6](#_Toc187617215)

[ **Conclusion:** 7](#_Toc187617216)

**Database Implementation, Data Population, and Query Capabilities**

# **Objective:**

The goal of this document is to detail the database implementation, data population, and query capabilities for a Movie Recommendation System. This includes an explanation of the schema, sample data, and SQL and relational algebra queries executed to test the functionality of the database.

# **Database Implementation:**

## **Database Schema:**

The schema is designed based on the Movie Recommendation System requirements and includes the following tables:

* **Users:(clients)**
  + **UserID** (Primary Key)
  + Name
  + Email
  + Password
  + JoinDate
* **Movies:**
  + **MovieID** (Primary Key)
  + Title
  + Genre (stores genre IDs as comma-separated values)
  + ReleaseDate
  + Director
  + Rating
* **Ratings:(review list)**
  + **RatingID** (Primary Key)
  + **UserID** (Foreign Key referencing Users.UserID)
  + **MovieID** (Foreign Key referencing Movies.MovieID)
  + Rating
  + Review
* **Recommendations:**
  + **RecommendationID** (Primary Key)
  + **UserID** (Foreign Key referencing Users.UserID)
  + **MovieID** (Foreign Key referencing Movies.MovieID)
  + RecommendationDate
* **WatchHistory:**
  + **WatchID** (Primary Key)
  + **UserID** (Foreign Key referencing Users.UserID)
  + **MovieID** (Foreign Key referencing Movies.MovieID)
  + WatchDate

This schema was implemented using MySQL, ensuring referential integrity with primary and foreign key constraints.

# **Data Population:**

Sample data was inserted into the tables to ensure functionality and demonstrate query capabilities. Below are examples:

## **Users Table:**

| **UserID** | **Name** | **Email** | **Password** | **JoinDate** |
| --- | --- | --- | --- | --- |
| 1 | Alice Johnson | [alice@gmail.com](mailto:alice@gmail.com) | \*\*\*\*\*\*\*\* | 2023-01-15 |
| 2 | Bob Smith | [bob@gmail.com](mailto:bob@gmail.com) | \*\*\*\*\*\*\*\* | 2023-02-20 |
| 3 | Charlie Brown | [charlie@gmail.com](mailto:charlie@gmail.com) | \*\*\*\*\*\*\*\* | 2023-03-10 |
|  |  |  |  |  |

## **Movies Table:**

| **MovieID** | **Title** | **Genre** | **ReleaseDate** | **Director** | **Rating** |
| --- | --- | --- | --- | --- | --- |
| 1 | Inception | 11,13 | 2010-07-16 | Christopher Nolan | 8.8 |
| 2 | The Dark Knight | 2 | 2008-07-18 | Christopher Nolan | 9.0 |
| 3 | Interstellar | 11 | 2014-11-07 | Christopher Nolan | 8.6 |
|  |  |  |  |  |  |

## **Ratings Table:**

| **RatingID** | **UserID** | **MovieID** | **Rating** | **Review** |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 5 | Mind-blowing! |
| 2 | 2 | 2 | 4.5 | Outstanding action |
| 3 | 3 | 3 | 4 | Great visuals |
|  |  |  |  |  |

## **Recommendations Table:**

| **RecommendationID** | **UserID** | **MovieID** | **RecommendationDate** |
| --- | --- | --- | --- |
| 1 | 1 | 2 | 2024-01-05 |
| 2 | 2 | 1 | 2024-01-06 |
| 3 | 3 | 3 | 2024-01-07 |
|  |  |  |  |

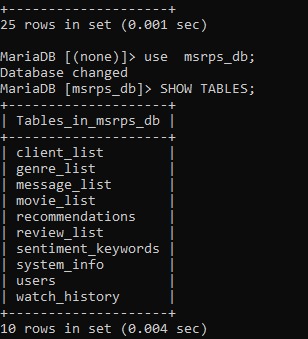
## **WatchHistory Table:**

| **WatchID** | **UserID** | **MovieID** | **WatchDate** |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 2024-01-01 |
| 2 | 2 | 2 | 2024-01-02 |
| 3 | 3 | 3 | 2024-01-03 |

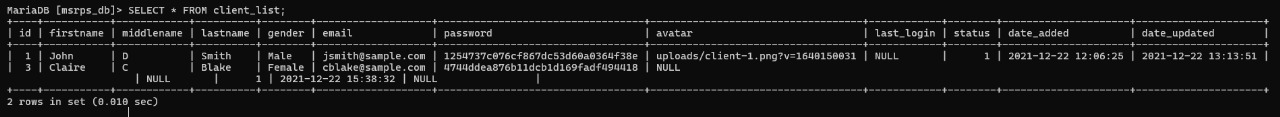
* **SQL QUERIES**

**DATABASE msrps\_db**

**Table**

****

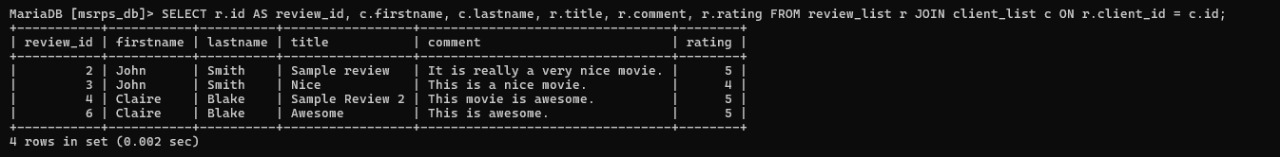
1. **SELECT \* FROM client\_list;**

****

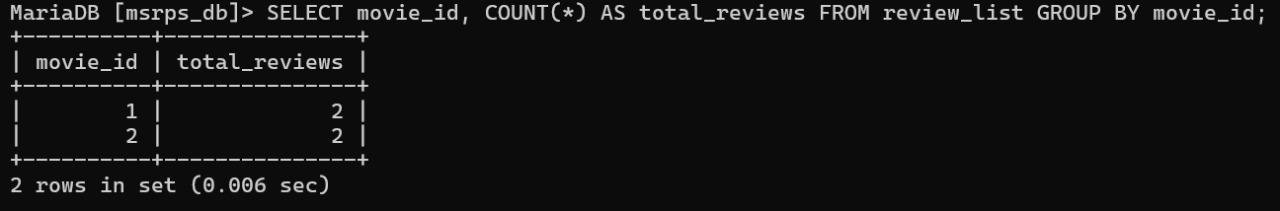
1. **Find all clients who have given reviews but have not rated any movie below 3:**

****

1. **SELECT r.id AS review\_id, c.firstname, c.lastname, r.title, r.comment, r.rating FROM review\_list r JOIN client\_list c ON r.client\_id = c.id;**

****

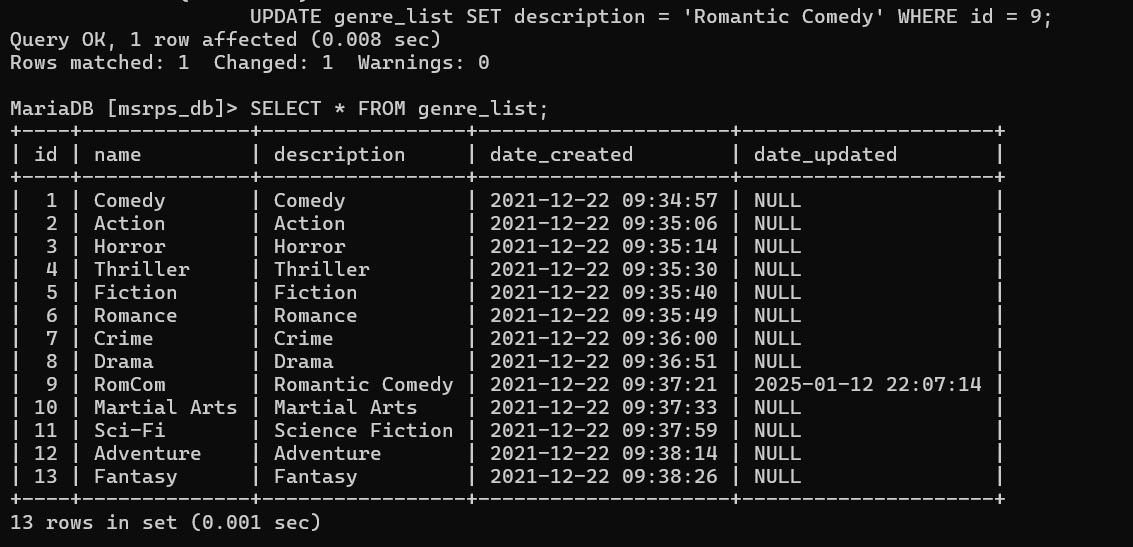
1. **SELECT movie\_id, COUNT(\*) AS total\_reviews FROM review\_list GROUP BY movie\_id;**

****

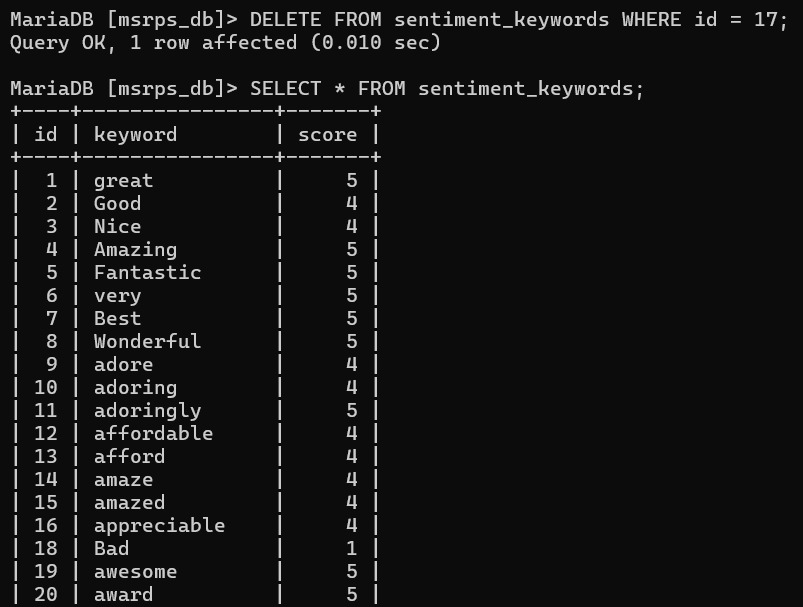
1. **INSERT INTO movie\_list (id, title, genres, director, produced, writter, actors, description, release\_date) VALUES (3, 'Inception', '11,13', 'Christopher Nolan', 'Emma Thomas', 'Jonathan Nolan', 'Leonardo DiCaprio', 'A mind-bending thriller', '2010-07-16');**

**WhatsApp Image 2025-01-12 at 22.06.59_6941e473**

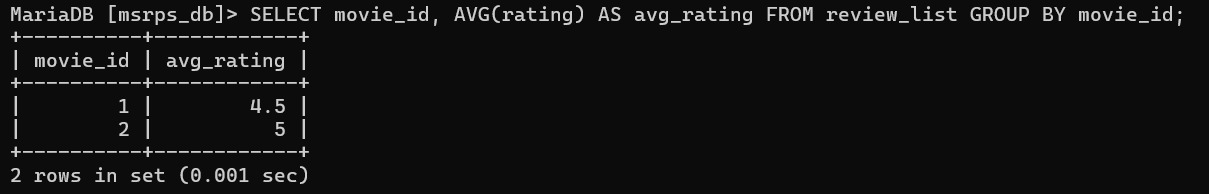
1. **UPDATE genre\_list SET description = 'Romantic Comedy' WHERE id = 9;**

****

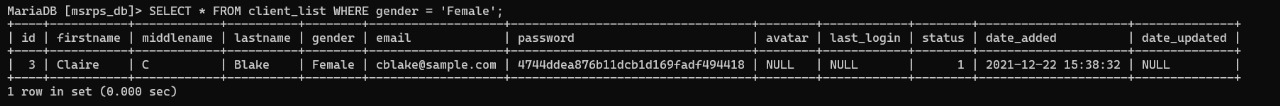
1. **DELETE FROM sentiment\_keywords WHERE id = 17;**

****

1. **SELECT movie\_id, AVG(rating) AS avg\_rating FROM review\_list GROUP BY movie\_id;**

****

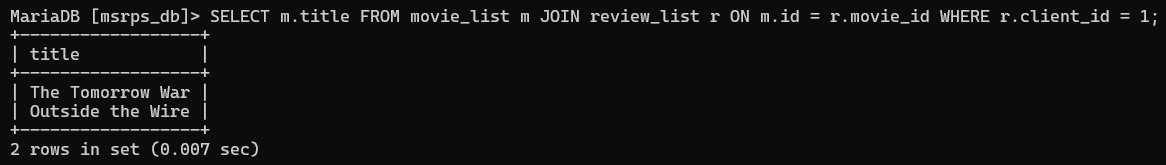
1. **SELECT \* FROM client\_list WHERE gender = 'Female';**

****

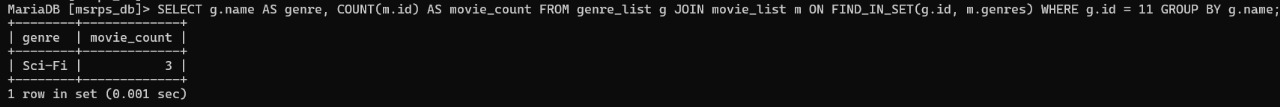
1. **SELECT \* FROM movie\_list WHERE FIND\_IN\_SET('2', genres) OR FIND\_IN\_SET('4', genres);**

**WhatsApp Image 2025-01-12 at 22.12.53_b2334bf5**

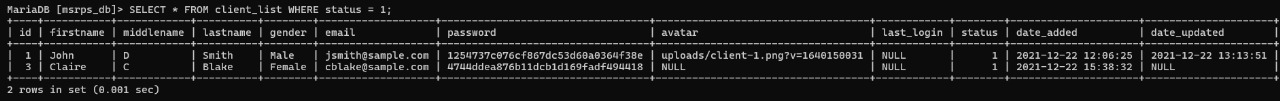
1. **SELECT m.title FROM movie\_list m JOIN review\_list r ON m.id = r.movie\_id WHERE r.client\_id = 1;**

****

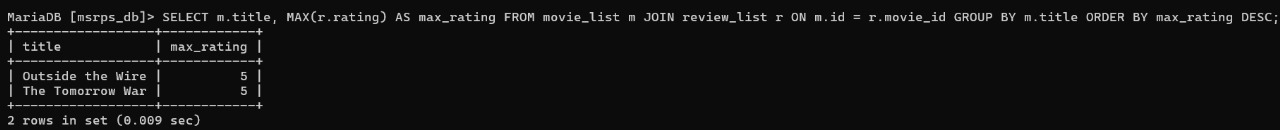
1. **SELECT g.name AS genre, COUNT(m.id) AS movie\_count FROM genre\_list g JOIN movie\_list m ON FIND\_IN\_SET(g.id, m.genres) WHERE g.id = 11 GROUP BY g.name;**

****

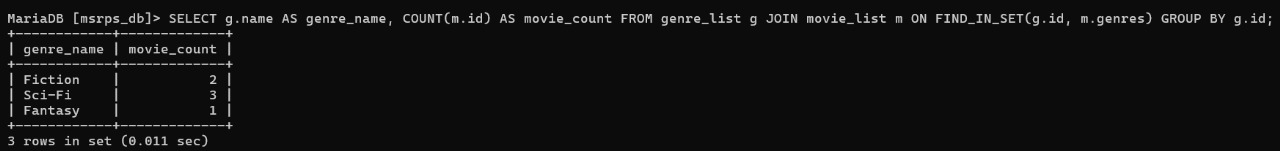
1. **SELECT \* FROM client\_list WHERE status = 1;**

****

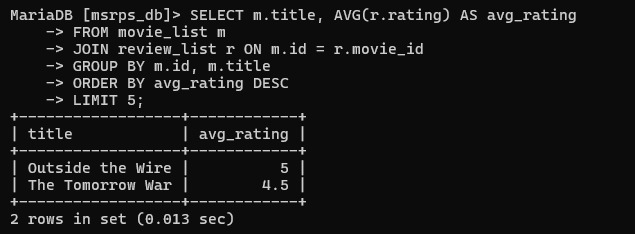
1. **SELECT m.title, MAX(r.rating) AS max\_rating FROM movie\_list m JOIN review\_list r ON m.id = r.movie\_id GROUP BY m.title ORDER BY max\_rating DESC;**

****

1. **SELECT g.name AS genre\_name, COUNT(m.id) AS movie\_count FROM genre\_list g JOIN movie\_list m ON FIND\_IN\_SET(g.id, m.genres) GROUP BY g.id;**

****

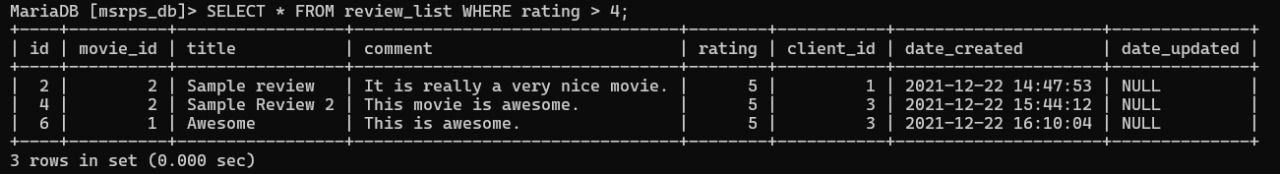
1. **Retrieve the top 5 highest-rated movies along with their average rating:**

****

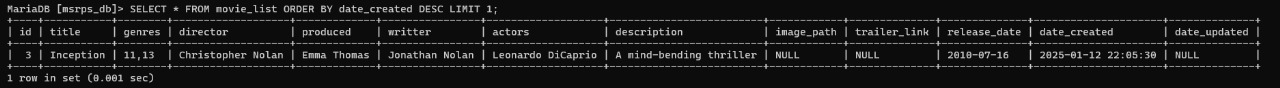
1. **SELECT \* FROM client\_list WHERE id NOT IN (SELECT DISTINCT client\_id FROM review\_list);**

**WhatsApp Image 2025-01-12 at 22.18.28_718d0f78**

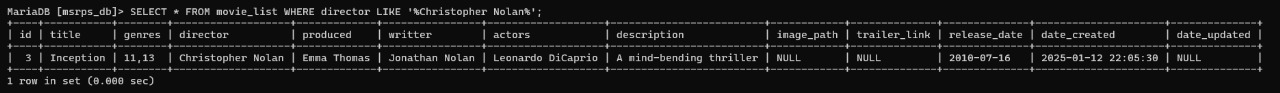
1. **SELECT \* FROM review\_list WHERE rating > 4;**

****

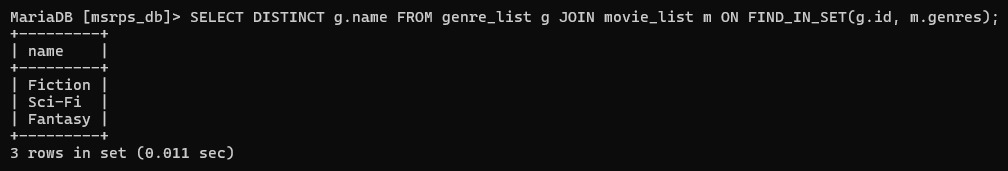
1. **SELECT \* FROM movie\_list ORDER BY date\_created DESC LIMIT 1;**

****

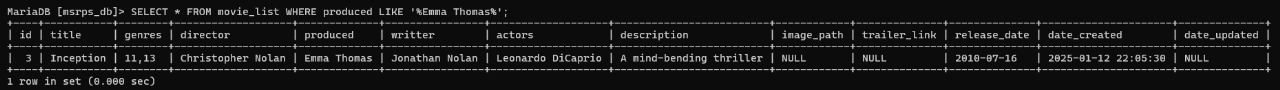
1. **SELECT \* FROM movie\_list WHERE genres LIKE '%,%';**
2. **SELECT \* FROM movie\_list WHERE director LIKE '%Christopher Nolan%';**

****

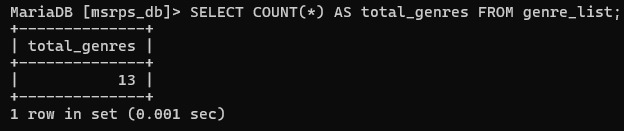
1. **SELECT DISTINCT g.name FROM genre\_list g JOIN movie\_list m ON FIND\_IN\_SET(g.id, m.genres);**

****

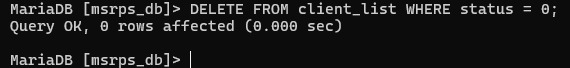
1. **SELECT \* FROM movie\_list WHERE produced LIKE '%Emma Thomas%';**

****

1. **SELECT COUNT(\*) AS total\_genres FROM genre\_list;**

****

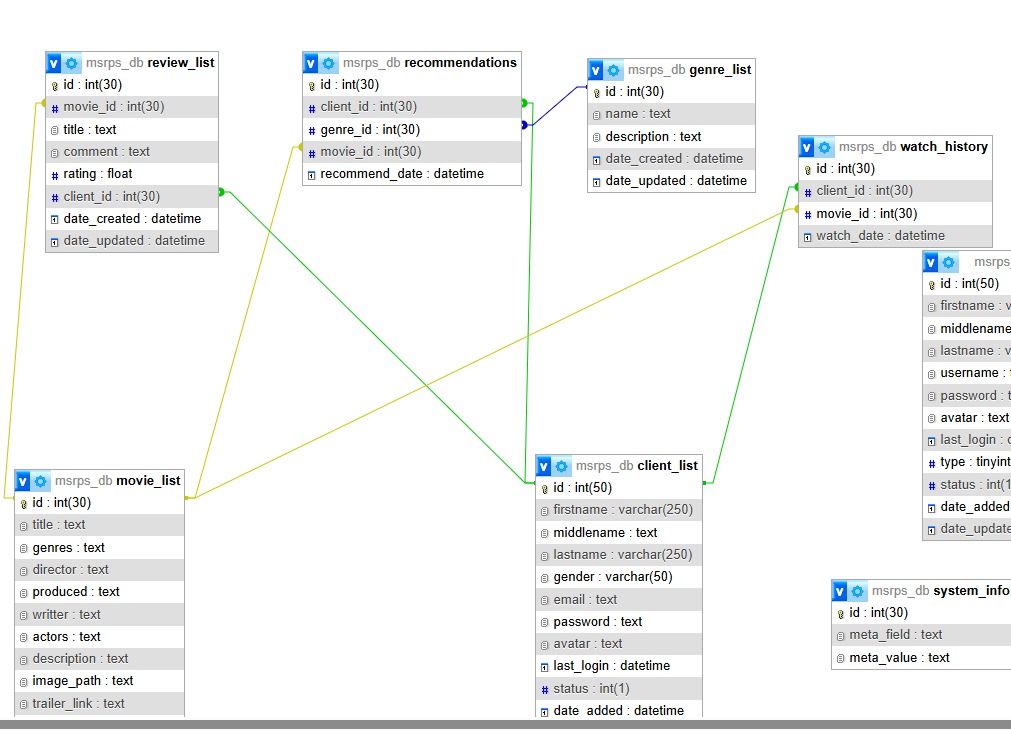
1. **DELETE FROM client\_list WHERE status = 0;**



* **Relational Algebra Queries**

1. σ (client\_list)
2. σ gender = 'Female' (client\_list)
3. π firstname (client\_list)
4. client\_list ⨝ client\_list.id = review\_list.client\_id (review\_list)
5. σ rating > 4 (client\_list ⨝ client\_list.id = review\_list.client\_id)
6. σ release\_date > '2020-12-31' (movie\_list)
7. π title (movie\_list)
8. σ movie\_id = 1 (review\_list)
9. client\_list ∪ client\_archive
10. genre\_list - { 'Action', 'Thriller' }
11. genre\_list ∩ other\_genre\_list
12. movie\_list × review\_list
13. σ status = 1 (client\_list)
14. genre\_list - π genres (movie\_list)
15. movie\_list ⨝ movie\_list.id = review\_list.movie\_id (review\_list)
16. γ MAX(rating) (review\_list)
17. π director (movie\_list)
18. σ genres = '11' (movie\_list)
19. γ COUNT(review\_id) GROUP BY movie\_id (review\_list)
20. σ status = 0 (client\_list)

* **Relational Schema**



# **Conclusion:**

This report outlines the functionality and flexibility of the Movie Recommendation System database, showcasing its ability to handle complex queries and provide meaningful insights for users and administrators.