CINETECH PRACTICAL ASSIGNMENT 5

23535246 : Lebogang Masenya

22747363 : Aundrea Ncube

21824241 : Mpho Siminya

22526171 : Nobuhle Mtshali

23539764 : Amantle Temu

23618583 : Malaika Kamangu

TASK 1

Research
What makes a good streaming service:

•Content Library:
A comprehensive content library is paramount to the success of a streaming service. It is imperative to seek out a platform that offers a diverse selection of movies, TV shows, documentaries, and original programming to cater to a wide range of viewer preferences and interests.

•Original Content:
Exclusive original shows and movies play a pivotal role in distinguishing a streaming service from its competitors. A prime example of this is

Netflix, renowned for its extensive catalogue of original programming, encompassing both acclaimed series and blockbuster films.

•Pricing:
When evaluating streaming services, it is crucial to consider the subscription cost. While some platforms offer competitive pricing, others may feature premium tiers with additional perks. Striking a balance between affordability and content quality is key to ensuring a satisfying user experience.
•User Interface:
A user-friendly interface greatly enhances the navigation and content discovery process for viewers. Services like Hulu and Disney+ are lauded for their intuitive interfaces, facilitating seamless interaction with the platform.

•Streaming Quality:
The capability to support high-definition (HD) and 4K streaming is imperative for delivering an immersive viewing experience. It is imperative to select a service that ensures smooth playback even under varying internet connection speeds.

•Device Compatibility:
Compatibility across a diverse array of devices, including smartphones, tablets, smart TVs, and gaming consoles, is essential to accommodate the diverse viewing habits of users. Most popular streaming services are readily accessible across multiple platforms.

•Ad-Free Options:
For users seeking an uninterrupted viewing experience, the availability of ad-free options is crucial. Platforms such as Amazon Prime Video offer competitive ad-free pricing, catering to this preference.

•Live TV and Sports:

For enthusiasts of live TV channels or sports content, platforms like YouTube TV or Fubo are noteworthy considerations, offering an extensive array of live programming options.

•Family-Friendly Content:
Streaming services that curate family-friendly content are highly valued by households. Disney+, for instance, stands out as an exemplary choice due to its extensive library of family-friendly entertainment options.
•Content Discovery and Personalization:
Effective content discovery features, such as personalised recommendations and curated playlists, help users discover new titles tailored to their interests. Implementing algorithms that analyse user viewing history, preferences, and ratings can enhance the accuracy of content recommendations over time.

•Offline Viewing:
Offline viewing functionality allows users to download content for later viewing without requiring an internet connection. This feature is particularly valuable for users with limited or unreliable internet access.

•Accessibility and Inclusivity:
Providing accessibility features such as closed captions, audio descriptions, and customizable subtitles ensures that the streaming service is inclusive and caters to users with diverse needs.

•Community Engagement:
Foster a sense of community among users through features such as discussion forums, user-generated content, and social media integration. Encourage user interaction and feedback to create a more engaging and immersive experience.

•Brand Identity and Differentiation:
Cultivate a distinct brand identity and positioning that sets the streaming service apart from competitors such as exclusive content partnerships,

niche genre offerings, or innovative features, to attract and retain subscribers

Movie/Film trends in genres

1.Hardcore Sci-Fi
2.Female Superheroes
3.Film to TV adaptations.
4.Video On Demand
5.Classic Horror
6.Embrace of Niceness

Streaming sites trend

1.Week-to-week releases
2.Platform exclusives
3.Docuseries
4.Traditional TV channels joining streaming services.
5.On-Demand convenience
6.Social media platforms hosting live-streamed video.
7.Spending on indie content

Streaming sites strengths:

•Netflix: Best for Original shows
•Hulu: Best for Live and On-Demand content
•Peacock: Affordable
•Amazon Prime Video: Best for On-Demand shows and Movies.
•HBO Max: Best for HBO shows and Movies.
•Disney+: Best for Family friendly content
•Crunchyroll: Best for Anime

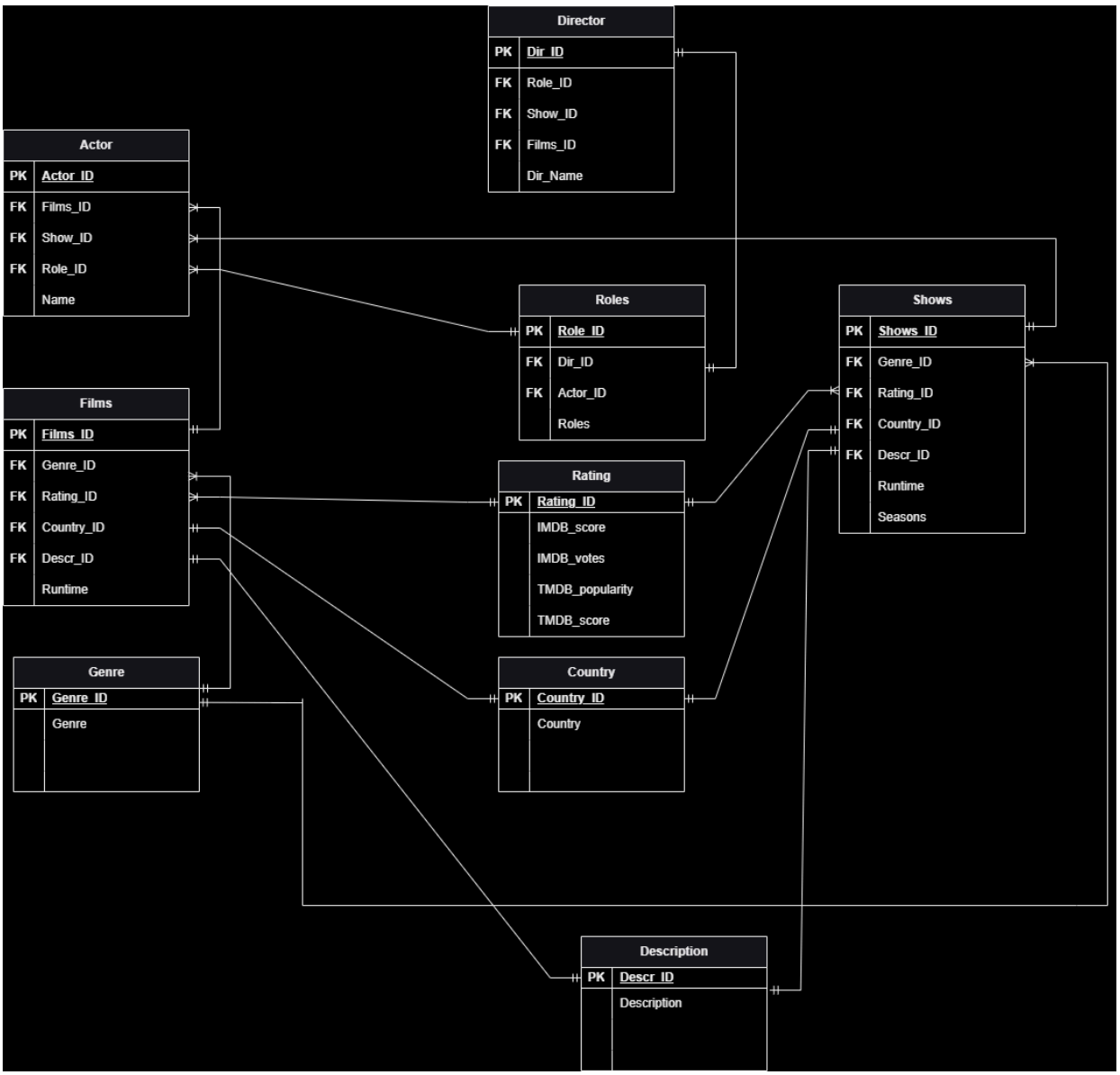
References

The Best Video Streaming Services for 2024 | PCMag
https://www.reviews.org/tv-service/best-streaming-service
https://mybroadband.co.za/news/broadcasting/525088-best-rated-streaming-apps-in-south-africa-with-prices.html
11 Huge Streaming Trends 2024-2027 (explodingtopics.com)
10 Movie And TV Trends We Love (slashfilm.com)

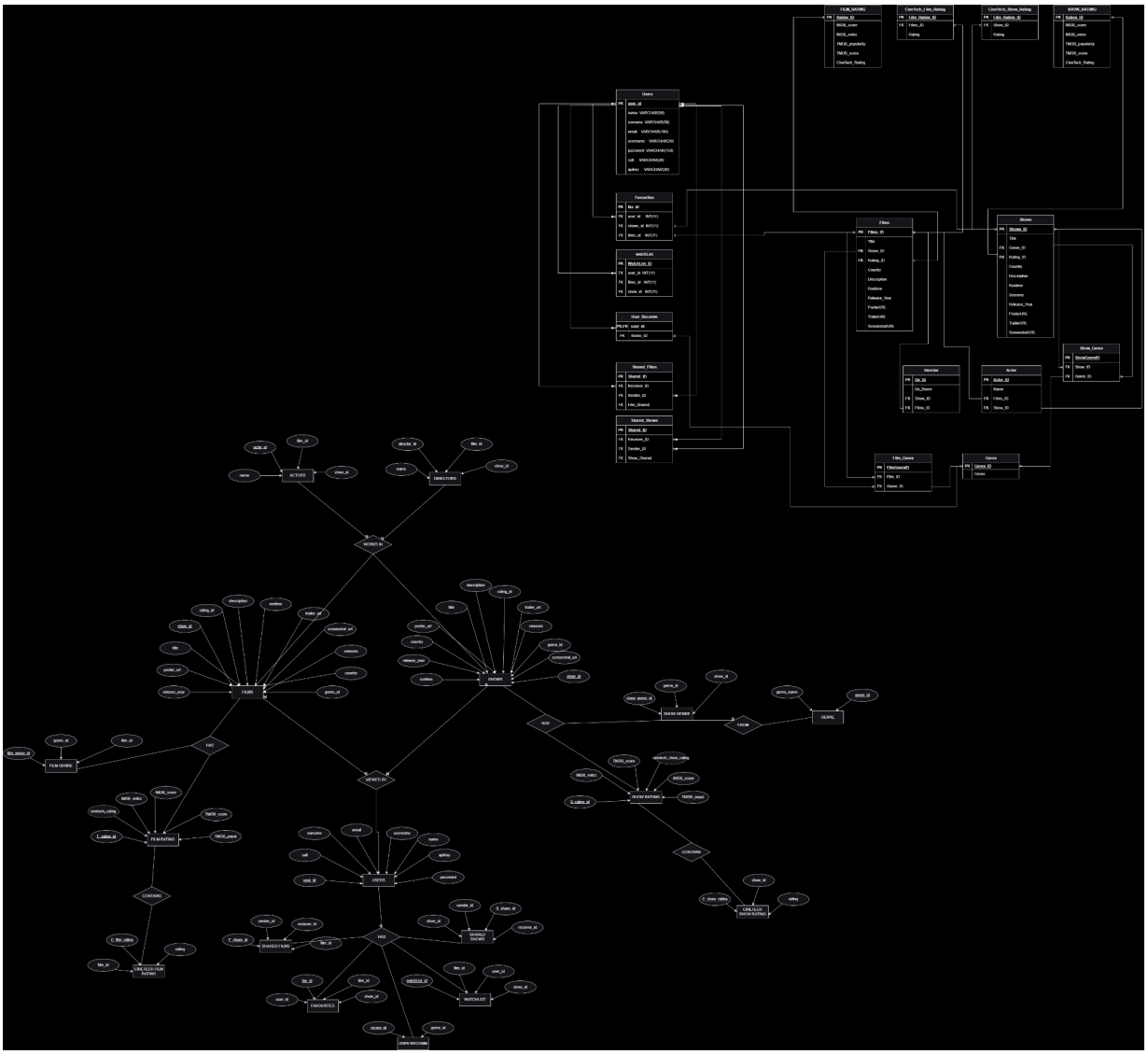
An Industry Transformed: Four Emerging Trends in Film & TV (visualcapitalist.com)
Genre Trend Analysis of 2023 Film and Television | Parrot Analytics
EER Diagram
EER Diagram to Relational Mapping
Relational Schema
Dataset: Netflix

TASK 2 : (E)ER-DIAGRAM

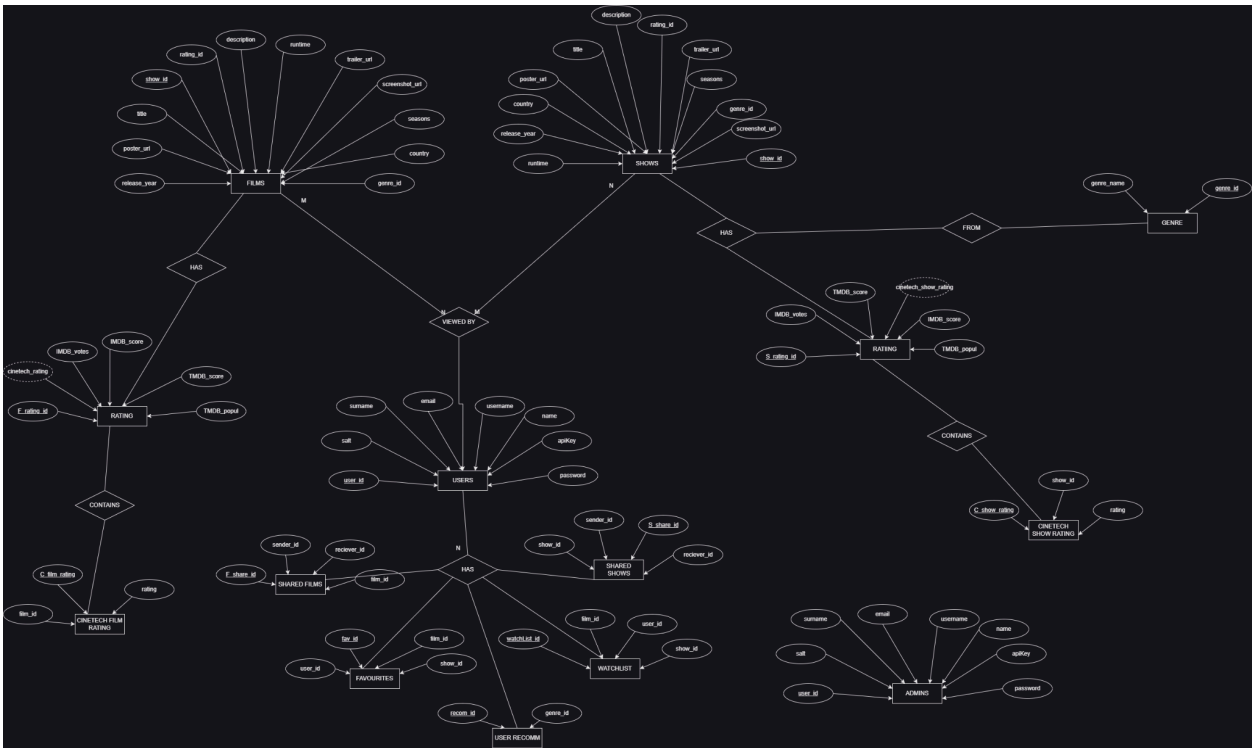
- First Draft:

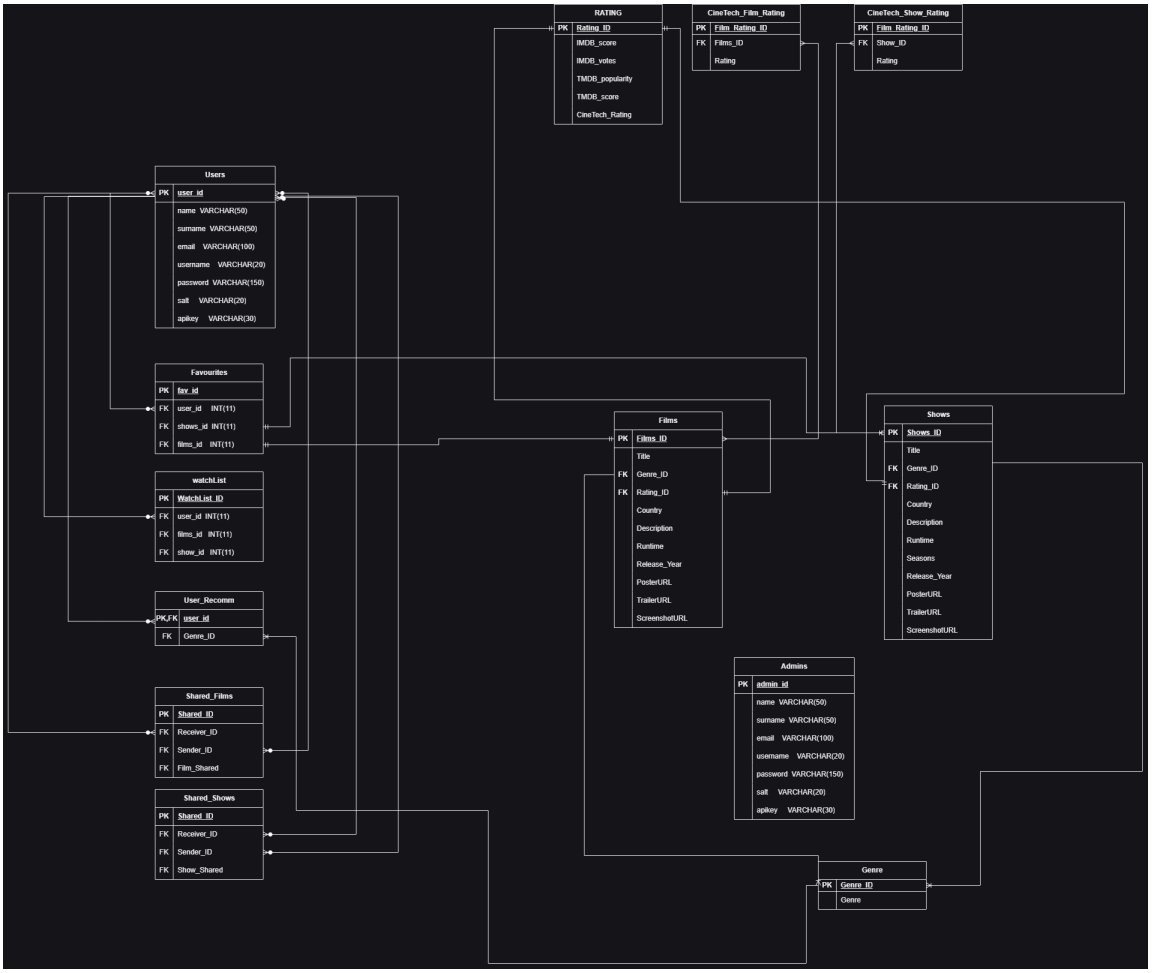


- Second Draft:



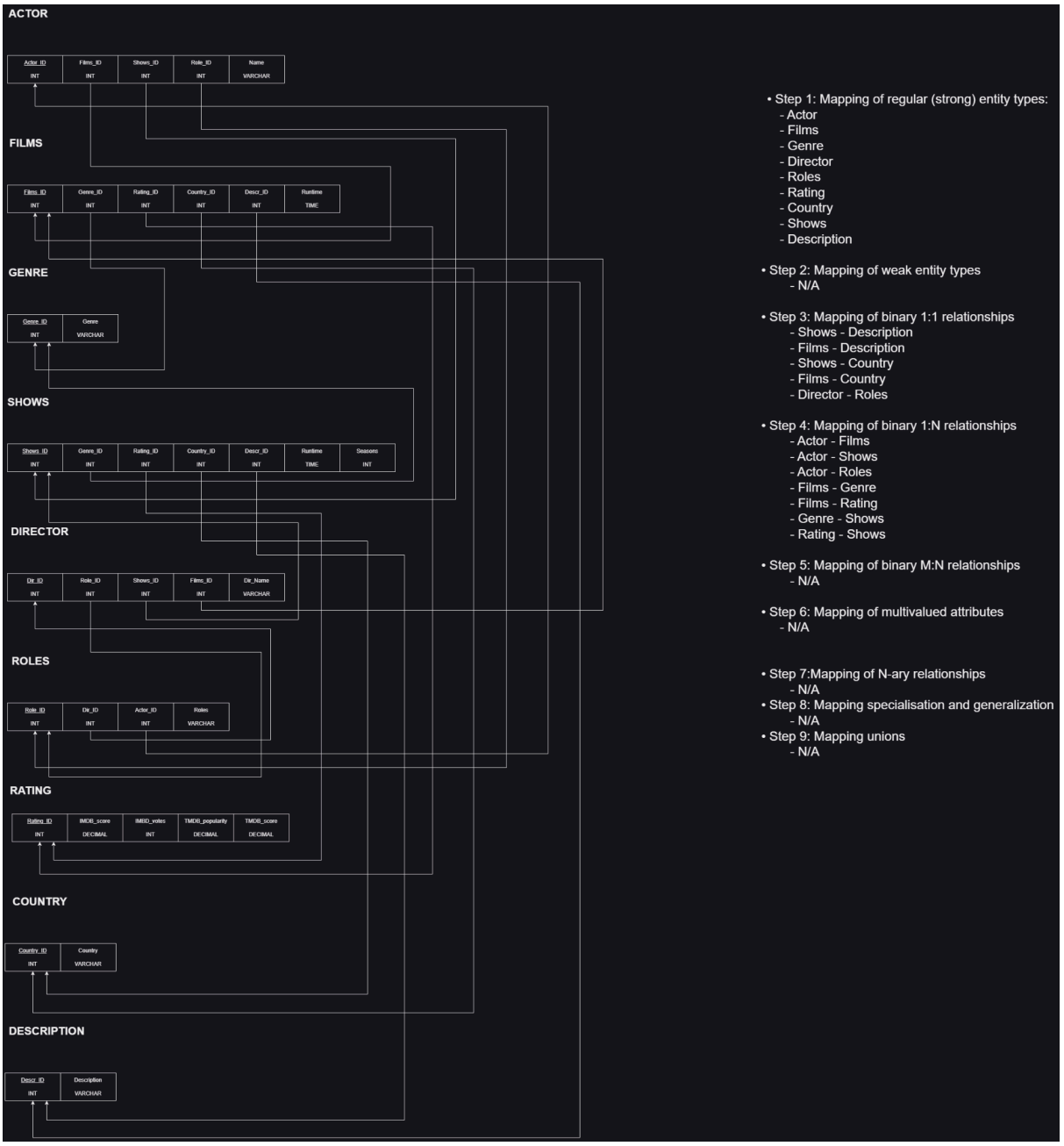
- Final Draft:


description
rating_id
runtime
trailer_url
screenshot_url
seasons
country
genre_id
show_id
title
poster_url
release_year
FILMS
SHOWS
genre_name
GENRE
FROM
HAS
M
N
VIEWED BY
IMDB_votes
IMDB_score
cinetech_rating
TMDB_score
F_rating_id
TMDB_popul
RATING
S_rating_id
cinetech_show_rating
CONTAINS
surname
email
username
name
call
apiKey
user_id
password
USERS
C_film_rating
rating
film_id
CINETECH FILM RATING
C_show_rating
CINETECH SHOW RATING
sender_id
reciever_id
F_share_id
SHARED FILMS
S_share_id
SHARED SHOWS
fav_id
FAVOURITES
watchList_id
WATCHLIST
recomt_id
USER RECOMM
ADMINS

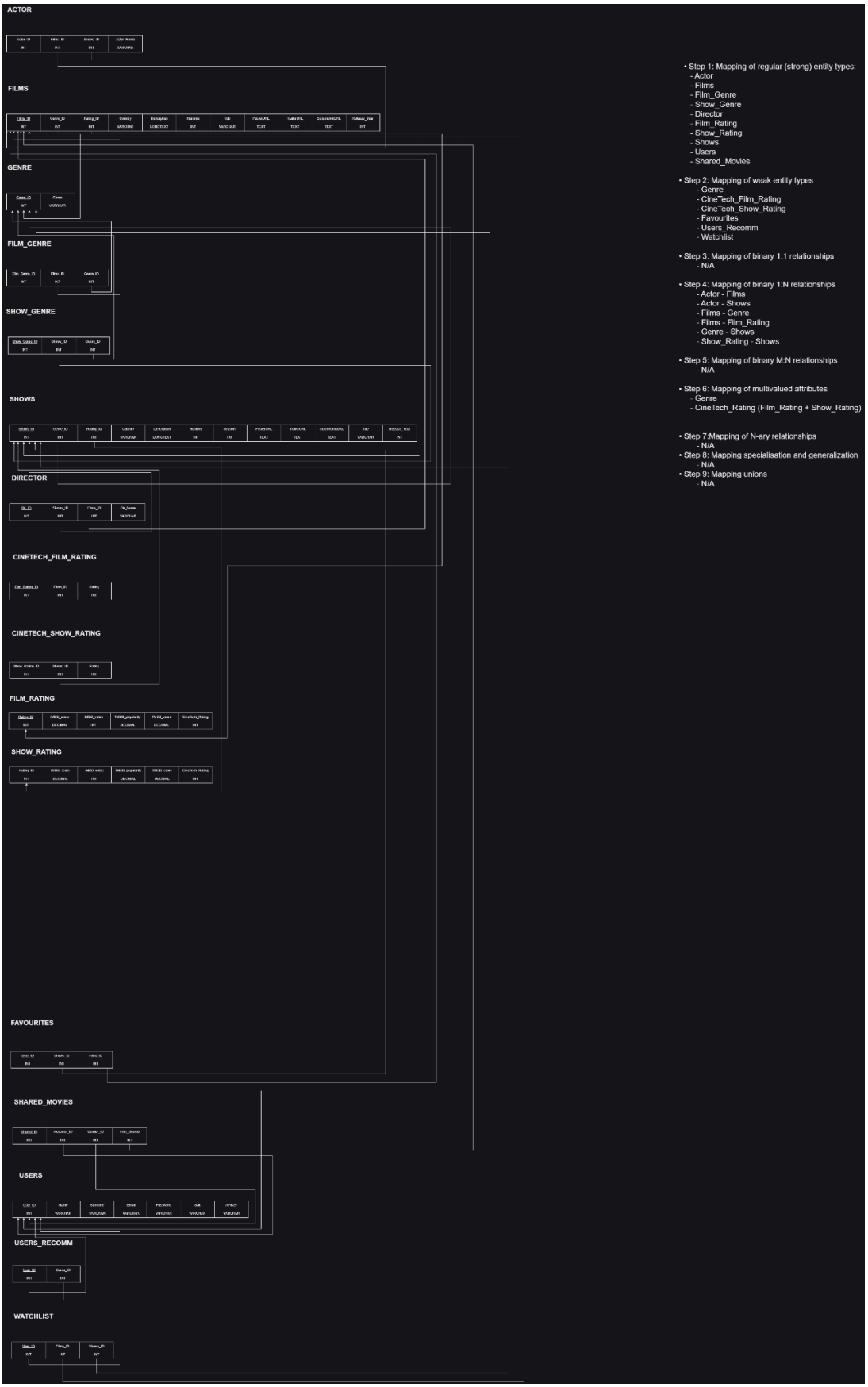

RATING
PK Rating_ID
IMDB_score
IMDB_votes
TMDB_popularity
TMDB_score
CineTech_Rating
CineTech_Film_Rating
PK Film_Rating_ID
FK Films_ID
Rating
CineTech_Show_Rating
PK Film_Rating_ID
FK Show_ID
Rating
Users
PK user_id
name VARCHAR(50)
surname VARCHAR(50)
email VARCHAR(100)
username VARCHAR(20)
password VARCHAR(150)
salt VARCHAR(20)
apikey VARCHAR(30)
Favourites
PK fav_id
FK user_id INT(11)
FK shows_id INT(11)
FK films_id INT(11)
watchList
PK WatchList_ID
FK user_id INT(11)
FK films_id INT(11)
FK show_id INT(11)
User_Recomm
PK,FK user_id
FK Genre_ID
Shared_Films
PK Shared_ID
FK Receiver_ID
FK Sender_ID
FK Film_Shared
Shared_Shows
PK Shared_ID
FK Receiver_ID
FK Sender_ID
FK Show_Shared
Films
PK Films_ID
Title
FK Genre_ID
FK Rating_ID
Country
Description
Runtime
Release_Year
PosterURL
TrailerURL
ScreenshotURL
Shows
PK Shows_ID
Title
FK Genre_ID
FK Rating_ID
Country
Description
Runtime
Seasons
Release_Year
PosterURL
TrailerURL
ScreenshotURL
Admins
PK admin_id
name VARCHAR(50)
surname VARCHAR(50)
email VARCHAR(100)
username VARCHAR(20)
password VARCHAR(150)
salt VARCHAR(20)
apikey VARCHAR(30)
Genre
PK Genre_ID
Genre

TASK 3 : (E)ER-DIAGRAM TO RELATIONAL MAPPING

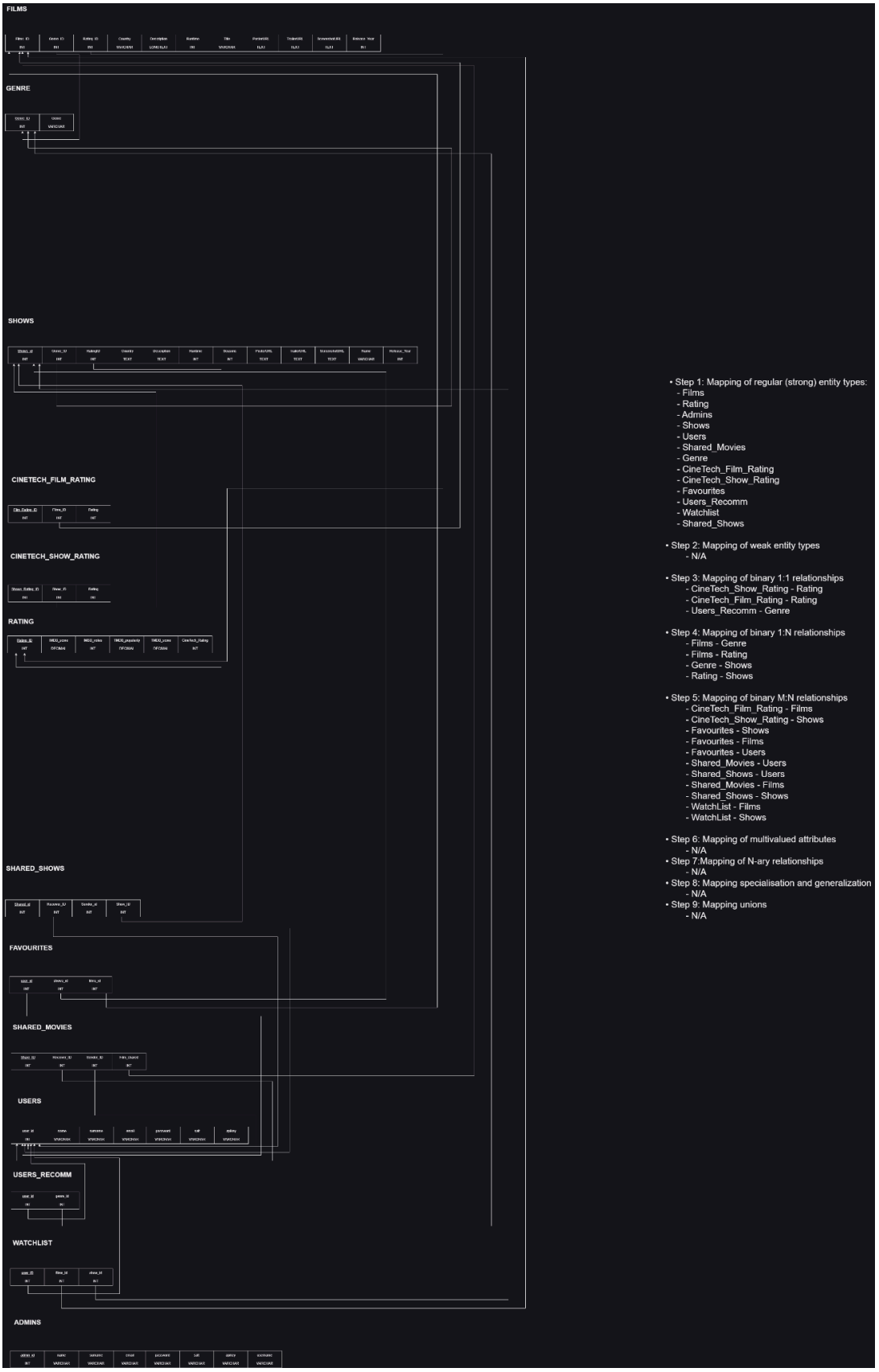
- First Draft :



- Second Draft :



- Final Draft :


FILMS
GENRE
SHOWS
CINETECH_FILM_RATING
CINETECH_SHOW_RATING
RATING
SHARED_SHOWS
FAVOURITES
SHARED_MOVIES
USERS
USERS_RECOMM
WATCHLIST
ADMINS
• Step 1: Mapping of regular (strong) entity types:
- Films
- Rating
- Admins
- Shows
- Users
- Shared_Movies
- Genre
- CineTech_Film_Rating
- CineTech_Show_Rating
- Favourites
- Users_Recomm
- Watchlist
- Shared_Shows
• Step 2: Mapping of weak entity types
- N/A
• Step 3: Mapping of binary 1:1 relationships
- CineTech_Show_Rating - Rating
- CineTech_Film_Rating - Rating
- Users_Recomm - Genre
• Step 4: Mapping of binary 1:N relationships
- Films - Genre
- Films - Rating
- Genre - Shows
- Rating - Shows
• Step 5: Mapping of binary M:N relationships
- CineTech_Film_Rating - Films
- CineTech_Show_Rating - Shows
- Favourites - Shows
- Favourites - Films
- Favourites - Users
- Shared_Movies - Users
- Shared_Shows - Users
- Shared_Movies - Films
- Shared_Shows - Shows
- WatchList - Films
- WatchList - Shows
• Step 6: Mapping of multivalued attributes
- N/A
• Step 7:Mapping of N-ary relationships
- N/A
• Step 8: Mapping specialisation and generalization
- N/A
• Step 9: Mapping unions
- N/A

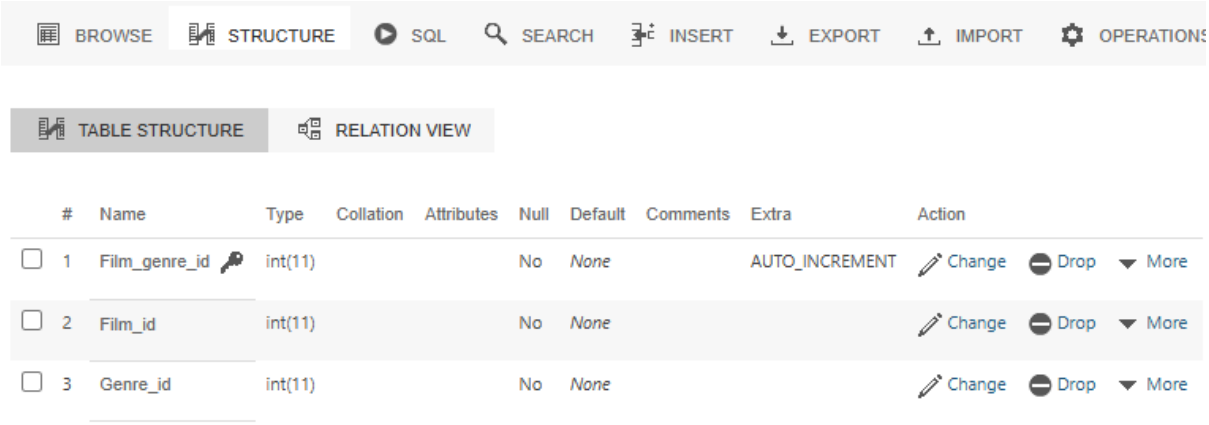
TASK 4 : RELATIONAL SCHEMA

TABLE STRUCTURE | RELATION VIEW

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Films_ID	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	Title	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	Genre_ID	int(11)			Yes	NULL			Change Drop More
4	Rating_ID	int(11)			Yes	NULL			Change Drop More
5	Country	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
6	Description	longtext	utf8mb4_general_ci		Yes	NULL			Change Drop More
7	PosterURL	text	utf8mb4_general_ci		Yes	NULL			Change Drop More
8	TrailerURL	text	utf8mb4_general_ci		Yes	NULL			Change Drop More
9	ScreenshotURL	text	utf8mb4_general_ci		Yes	NULL			Change Drop More
10	Runtime	int(11)			Yes	NULL			Change Drop More
11	Release_Year	int(11)			Yes	NULL			Change Drop More

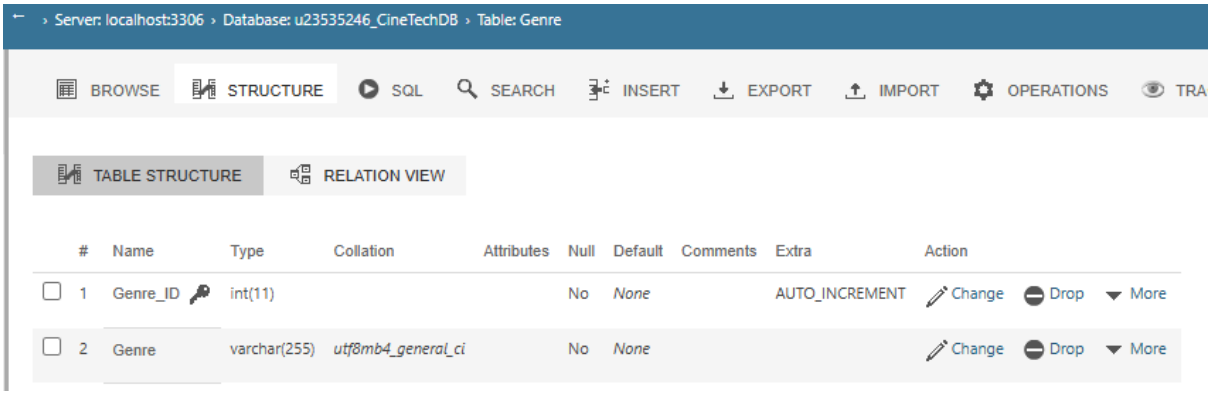
The above table is the films table with the following information:

- Primary & Foreign Keys : The Films_ID is the primary key of the table to uniquely identify each film in the table. The foreign keys in this table are Genre_ID which links to the genre table which consists of all existing genres, moreso the second foreign key is Rating_ID which links to the Rating table.
- Constraints and Checks :
- Data Types and Length Constraints :



The above table is the Film_genre table with the following information:

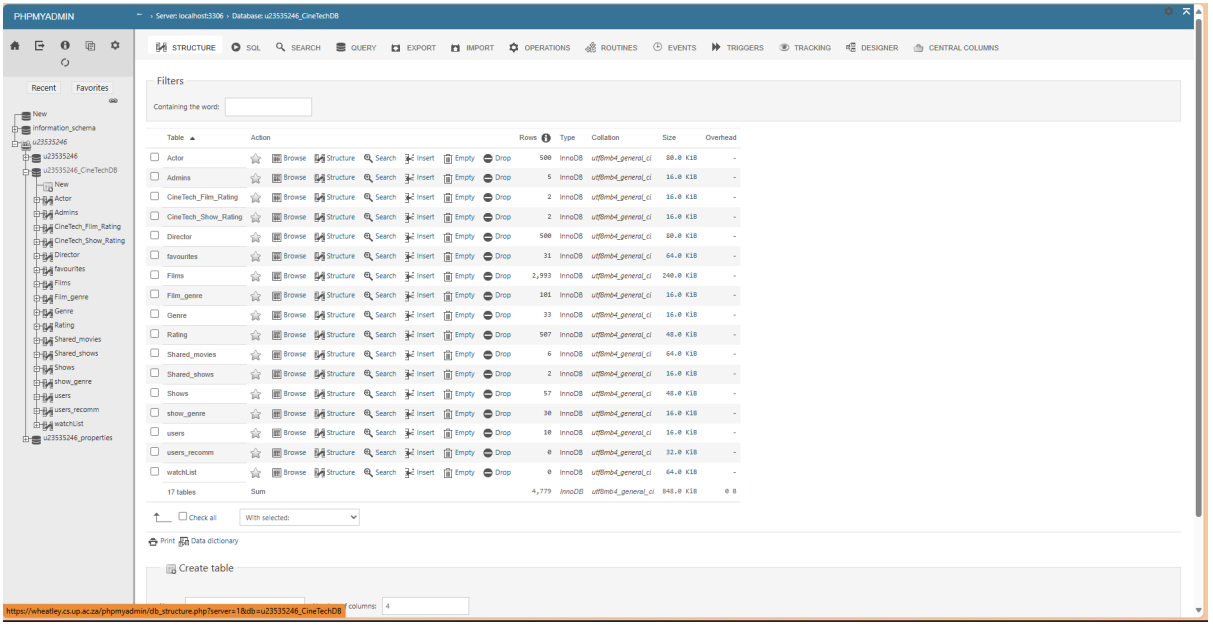
- Primary & Foreign Keys : The Films_genre_id is the primary key of the table to uniquely identify each film linked to each genre in the table. The foreign keys in this table are Film_ID which links to the Films table which consists of all existing genres, moreso the second foreign key is Genre_ID which links to the Genre table. This is done so that we can facilitate a single Film with multiple genres
- Constraints and Checks : All the data types cannot be null
- Data Types and Length Constraints : Each foreign key has a data type of int with length 11 which is the same as the keys on their relevant table to make sure there is consistency across all tables.



The above table is the genre table with the following information:

- Primary & Foreign Keys : The Genre_id is the primary key of the table to uniquely identify each genre in the table. There are no foreign keys in this table.
- Constraints and Checks : All the data types cannot be null and no default values
- Data Types and Length Constraints : Primary key Genre_id has a data type of int with length 11 and Genre is varchar 255.

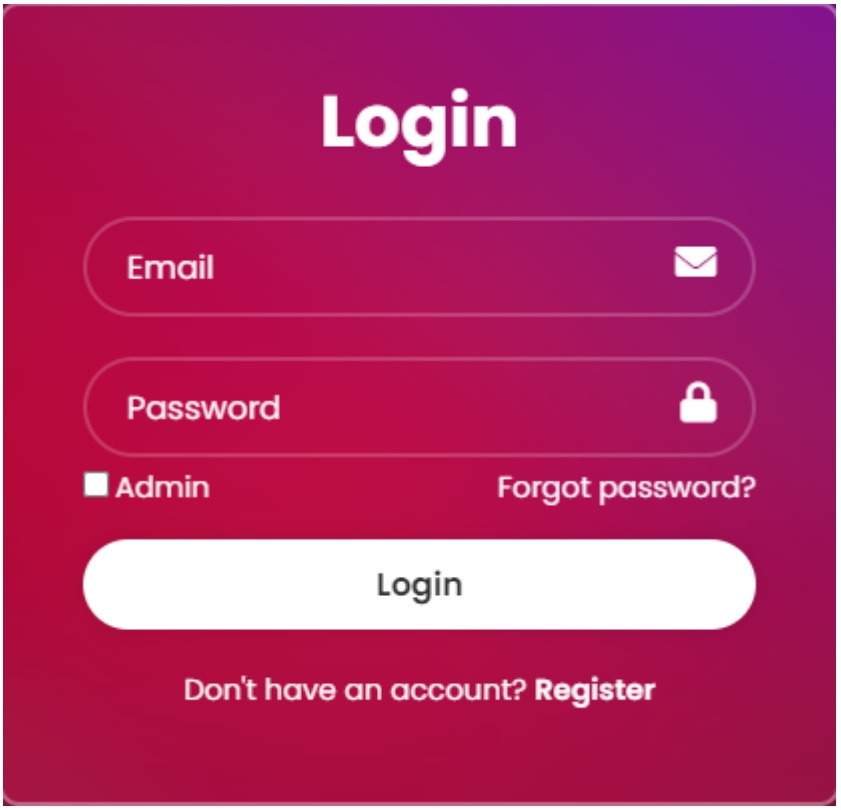
And the full database with the respective tables we used:



TASK 5 : WEB-BASED APPLICATION

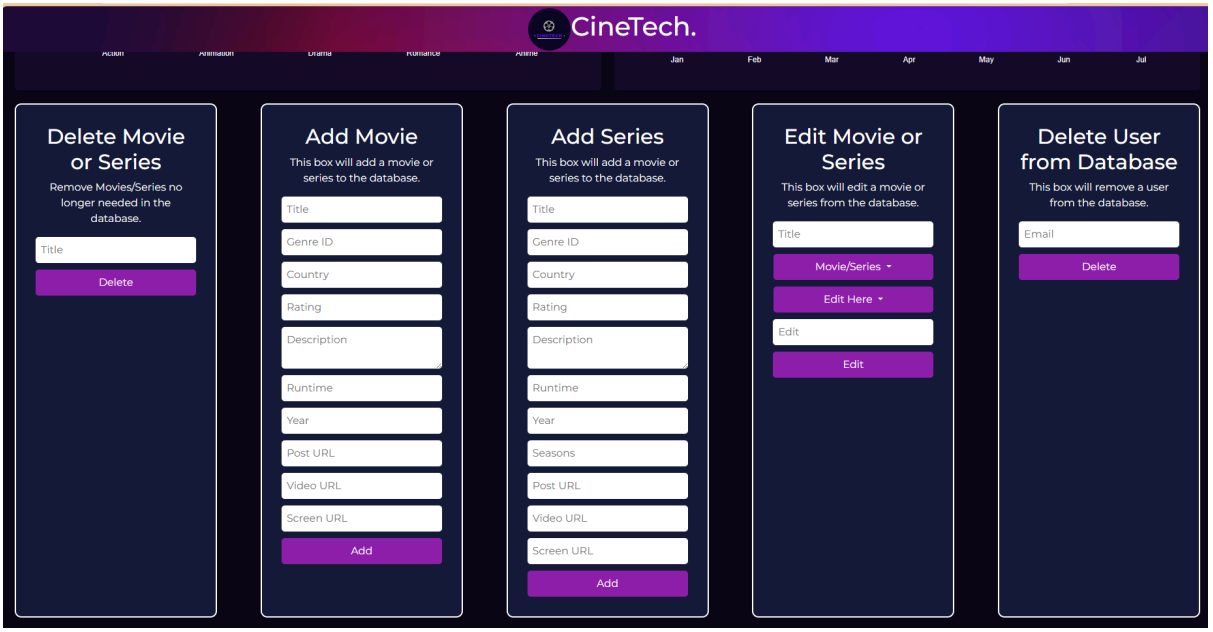
Login Page

Added a feature to login as either a normal user or as an admin.



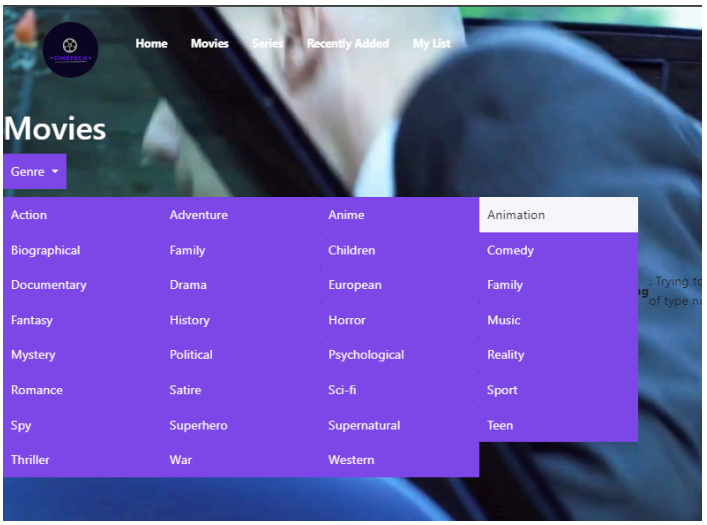
Add, edit, and delete Movies and TV series

This gets directly done on the database itself.



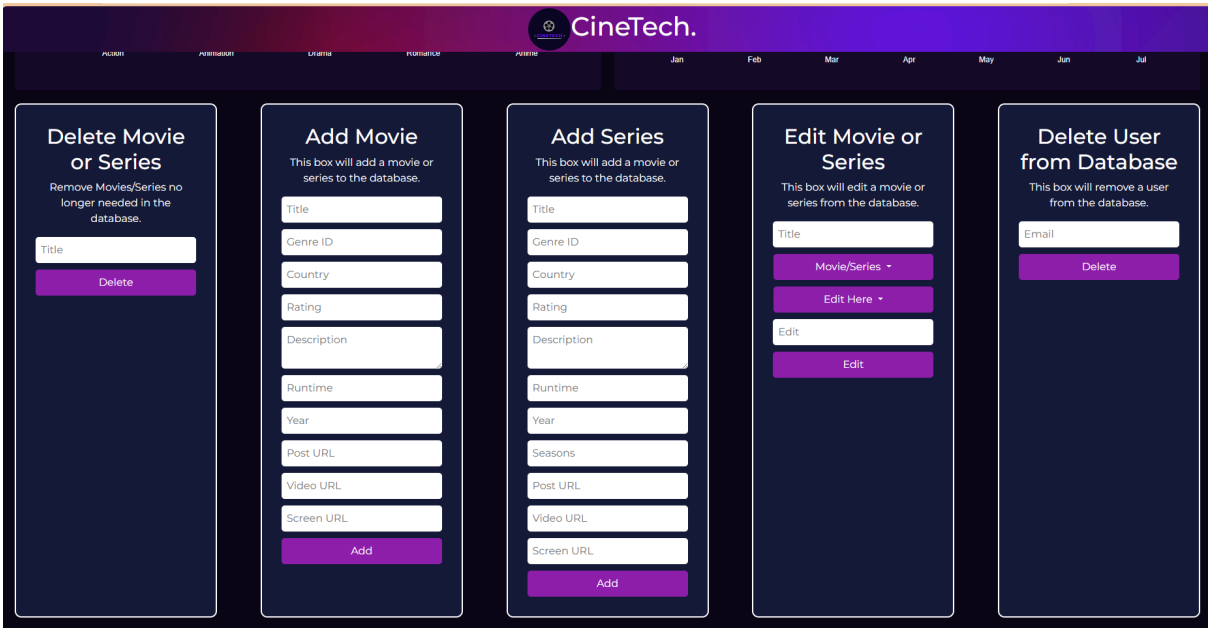
Sort and filter content based on various criteria

Added a sort by genre function on the movies and series pages. The movies and series pages were a way of filtering the data by type, either movie or series.
And a page that filters only the recently added movies by their release year.



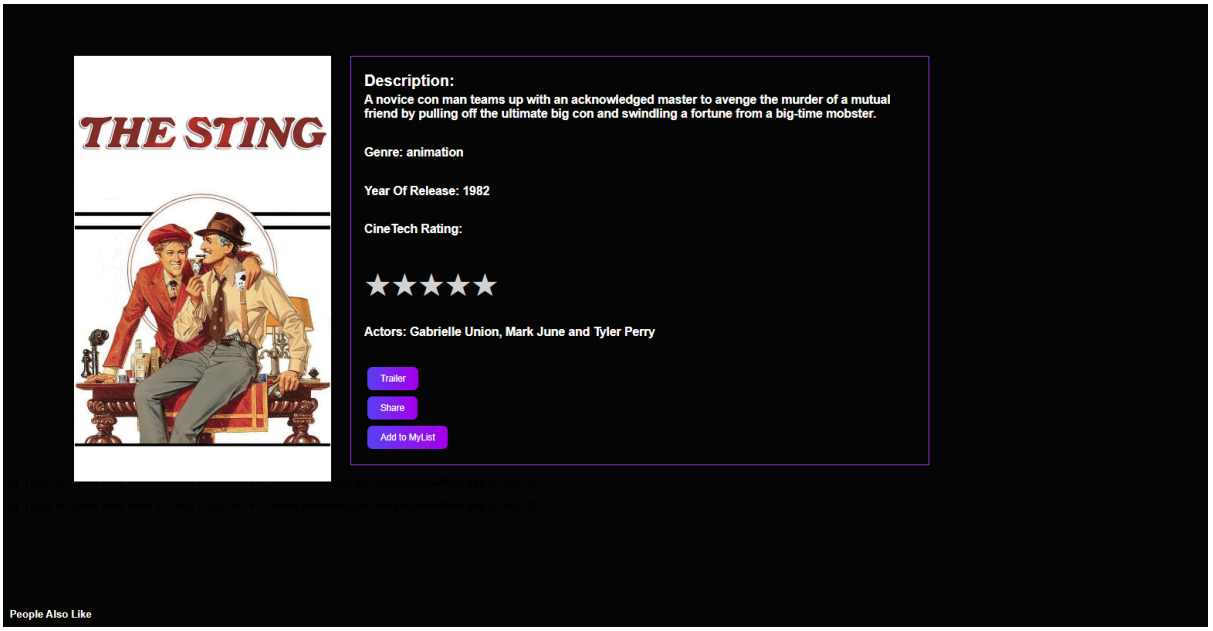
Update the database and/or extend, delete

Update to the database is done as an edit to either a movie or series, this is done, with no fuzzy search and section by section.



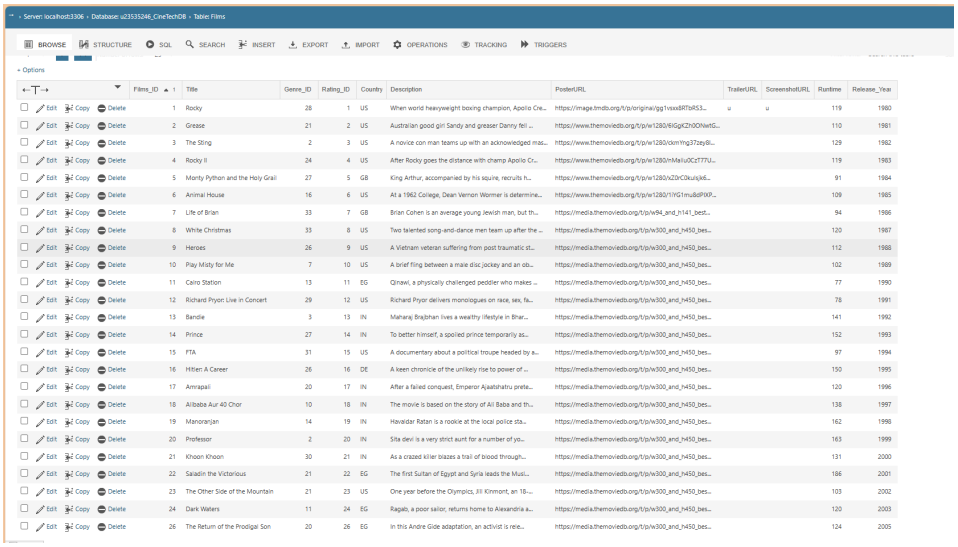
Recommend Movies and TV series based on a criteria

This is the recommended movies that appear in the view more, this is decided by the users frequently visited genres.If the user selects a action movie, underneath it will have people also like which is our recommended in this case,however this user hasn't watched a movie hence why there are no movies appearing underneath.

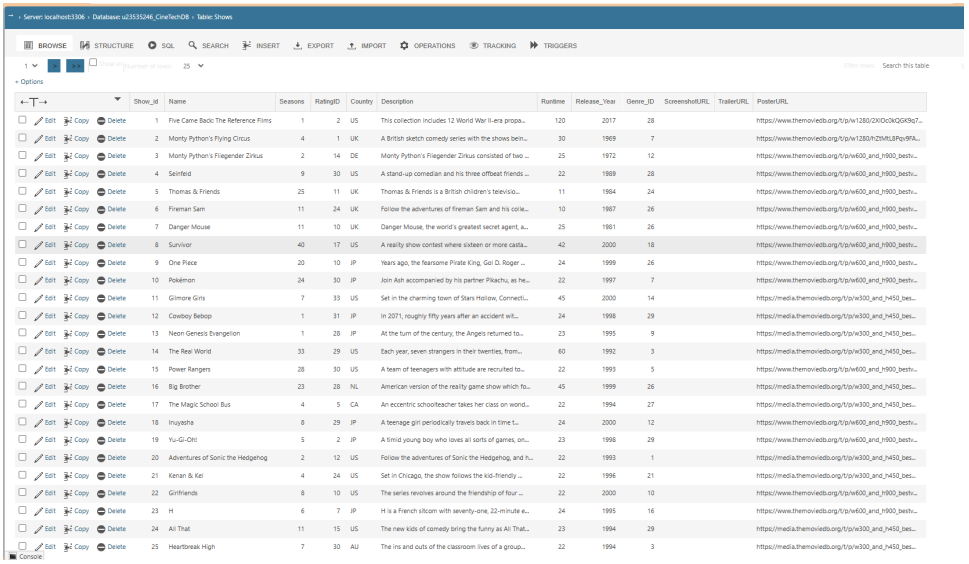


TASK 6 : DATA

Screenshot of the films (movies) table :



Screenshot of the shows table :



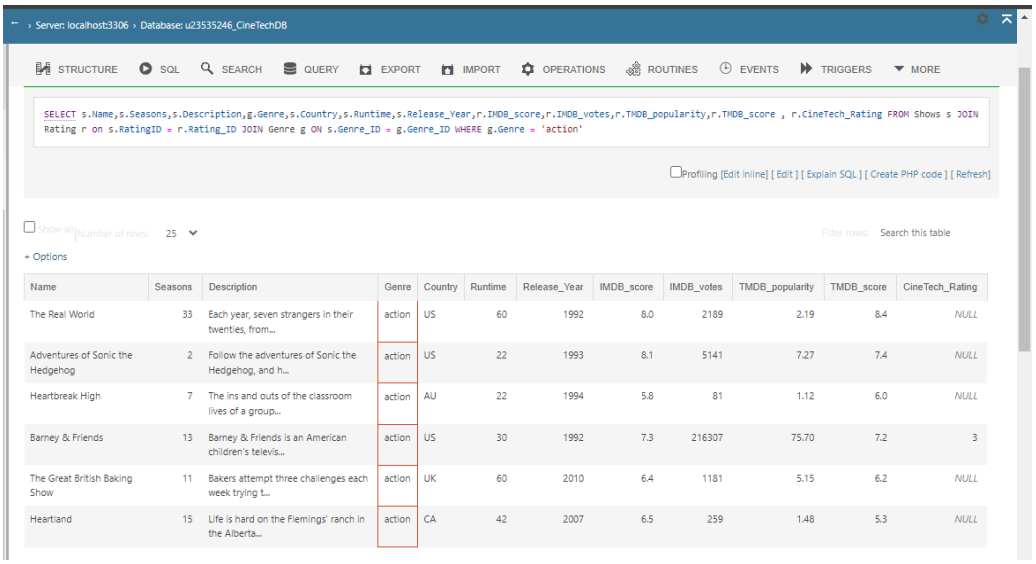
We used to populate each table with the CSV file from kaggle and used mySQL workbench to populate the dataset into our database using a script. An additional database cleanup where by we added we split the initial table which has all the movies in one table into sub tables for optimising SQL queries.

TASK 7 : ANALYSE

We made use of phpMyAdmin as our chosen phpMyAdmin.
Execution Plan Output:

Chose query: SELECT

```
s.Name,s.Seasons,s.Description,g.Genre,s.Country,s.Runtime,s.Release_Year,r.IMDB_score,r.IMDB_votes,r.TMDB_popularity,r.TMDB_score , r.CineTech_Rating FROM Shows s JOIN Rating r on s.RatingID = r.Rating_ID JOIN Genre g ON s.Genre_ID = g.Genre_ID WHERE g.Genre = 'action'
```



Name	Seasons	Description	Genre	Country	Runtime	Release_Year	IMDB_score	IMDB_votes	TMDB_popularity	TMDB_score	CineTech_Rating
The Real World	33	Each year, seven strangers in their twenties, from...	action	US	60	1992	8.0	2189	2.19	8.4	NULL
Adventures of Sonic the Hedgehog	2	Follow the adventures of Sonic the Hedgehog, and h...	action	US	22	1993	8.1	5141	7.27	7.4	NULL
Heartbreak High	7	The ins and outs of the classroom lives of a group...	action	AU	22	1994	5.8	81	1.12	6.0	NULL
Barney & Friends	13	Barney & Friends is an American children's televis...	action	US	30	1992	7.3	216307	75.70	7.2	3
The Great British Baking Show	11	Bakers attempt three challenges each week trying t...	action	UK	60	2010	6.4	1181	5.15	6.2	NULL
Heartland	15	Life is hard on the Flemings' ranch in the Alberta...	action	CA	42	2007	6.5	259	1.48	5.3	NULL

Report on the Initial Performance:

Key Observations

1. **Type: ALL for Table s (Shows):**
    - This indicates a full table scan, which is inefficient especially if the table is large.
2. **Type: eq_ref for Tables g (Genre) and r (Rating):**
    - These are using the primary key to fetch the rows, which is efficient.

Optimization Strategies

1. **Indexing:**
    - Add indexes on columns that are used in joins and the **WHERE** clause to avoid full table scans and improve performance.
        - Index on **s.Genre_ID**.

- Index on **s.RatingID**.
- Index on **g.Genre**.

Implementing Indexes

Here are the SQL commands to add the necessary indexes:

```
CREATE INDEX idx_shows_genre_id ON Shows(Genre_ID); CREATE INDEX idx_shows_rating_id ON Shows(RatingID); CREATE INDEX idx_genre_genre ON Genre(Genre);
```

Re-run EXPLAIN Command

After adding indexes, re-run the **EXPLAIN** command to see the new execution plan:

```
EXPLAIN SELECT s.Name, s.Seasons, s.Description, g.Genre, s.Country, s.Runtime, s.Release_Year, r.IMDB_score, r.IMDB_votes, r.TMDB_popularity, r.TMDB_score, r.CineTech_Rating FROM Shows s JOIN Rating r ON s.RatingID = r.Rating_ID JOIN Genre g ON s.Genre_ID = g.Genre_ID WHERE g.Genre = 'action';
```

Expected Results

With the new indexes, the execution plan should ideally show **ref** or **index** types instead of **ALL** for the **Shows** table, indicating that indexes are being used.

Optimised Execution Plan:

- **Optimised Query:**

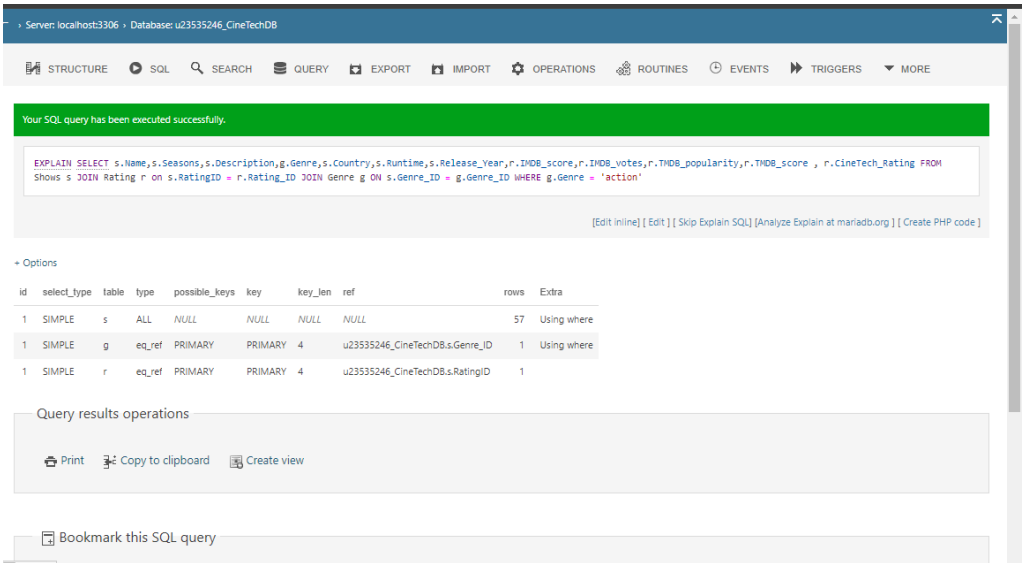
```
CREATE INDEX idx_shows_genre_id ON Shows(Genre_ID); CREATE INDEX idx_shows_rating_id ON Shows(RatingID); CREATE INDEX idx_genre_genre ON Genre(Genre); EXPLAIN SELECT s.Name, s.Seasons, s.Description, g.Genre, s.Country, s.Runtime, s.Release_Year, r.IMDB_score, r.IMDB_votes, r.TMDB_popularity, r.TMDB_score, r.CineTech_Rating FROM Shows s JOIN Rating r ON s.RatingID = r.Rating_ID JOIN Genre g ON s.Genre_ID = g.Genre_ID WHERE g.Genre = 'action';
```

- **Expected Execution Plan:**

Explanation of Performance Gain:

- **Improved Index Usage:**
  - The addition of indexes allows MySQL to use **ref** types instead of **ALL**, significantly reducing the number of rows scanned and improving query performance.

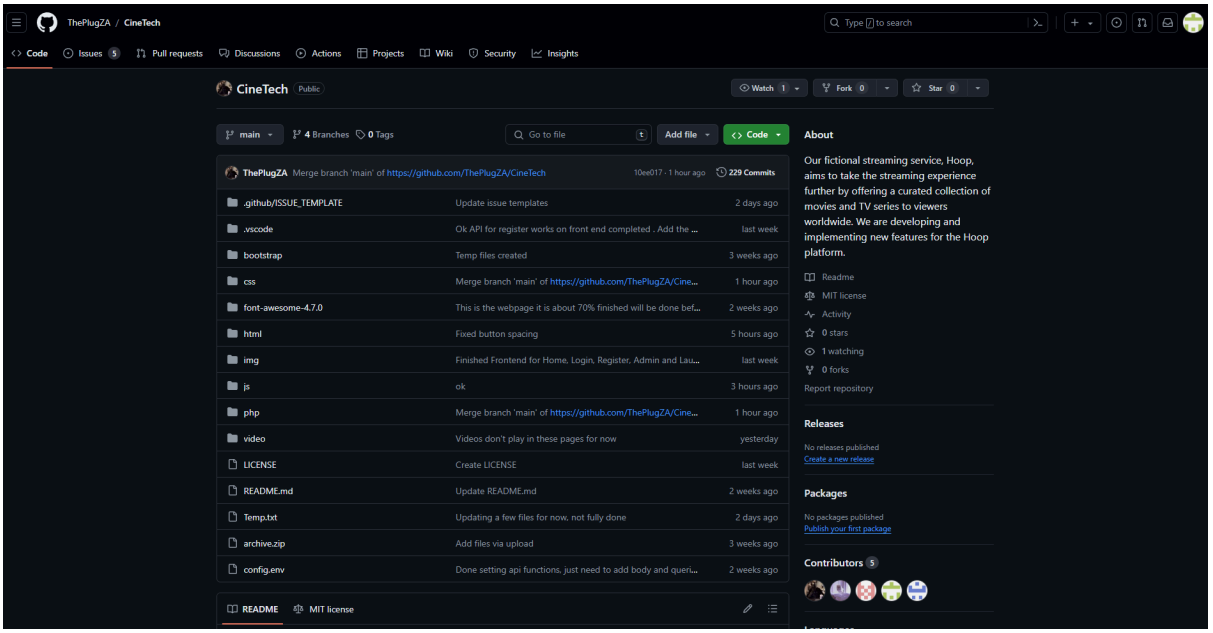
- **Reduced I/O Operations:**
    - Indexes help in reducing the number of disk I/O operations, leading to faster query execution times.

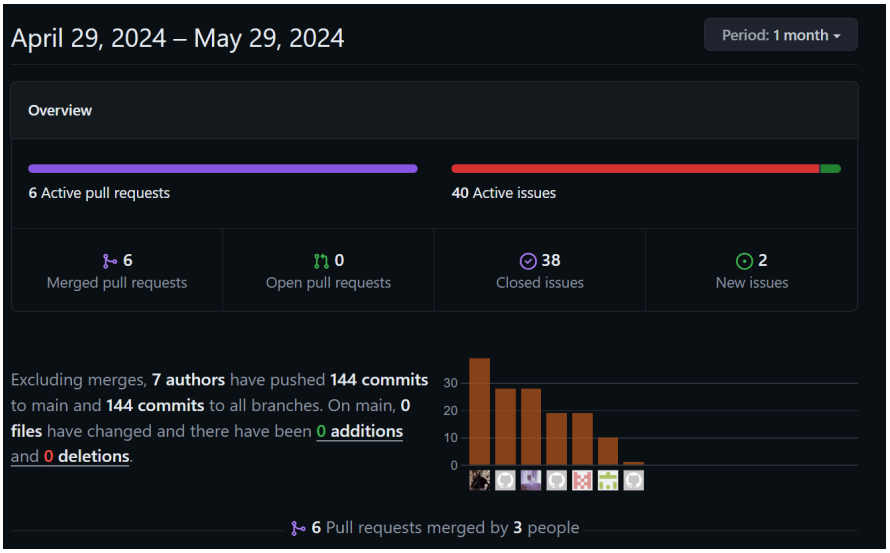


TASK 8 : DEVELOPMENT

Usage of Git

In our project we used git (below) to effectively update the files and make sure each team member has recent developments of what each individual has contributed to the project. With every commit, each team member provided comments consisting of brief explanations to allow for the next person to understand and build upon the previous implementation. Repository: https://github.com/ThePlugZA/CineTech





Data Validation Techniques

```php
public function registerUser($name, $surname, $email, $password, $username, $admin) {
    if (empty($name) || empty($surname) || empty($email) || empty($password)) {
        return json_encode(array("message" => "All fields are required"));
    }

    // Validate email format
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        return $this->errorResponse(time(), "Invalid email format");
    }

    // Validate password strength (e.g., minimum length, contain uppercase, lowercase, digit, symbol)
    if (strlen($password) < 8) {
        return json_encode(array('message' => 'Password must have at least 8 characters'));
    } else if (!preg_match('/[A-Z]/', $password)) {
        return json_encode(array('message' => 'Password should include at least one uppercase letter'));
    } else if (!preg_match('/[a-z]/', $password)) {
        return json_encode(array('message' => 'Password should include at least one lowercase letter'));
    } else if (!preg_match('/[0-9]/', $password)) {
        return json_encode(array('message' => 'Password should include at least one number'));
    }

    // Check if user already exists
    if ($admin === "true") {
        $stmt = $GLOBALS['connection']->prepare("SELECT admin_id FROM Admins WHERE email = ?");
    } else {
        $stmt = $GLOBALS['connection']->prepare("SELECT user_id FROM users WHERE email = ?");
    }

    $stmt->bind_param("s", $email);
    $stmt->execute();
    $result = $stmt->get_result();
    if ($result->num_rows > 0) {
        return $this->errorResponse(time(), "Person already exists");
    }
```

The above snippet code illustrates the use of regular expressions such as the ‘preg_match’ function to test the user input, if it is valid before inserting it into the database, we included error messages to alert the user in the event that their input does not meet the criteria and prompt them to enter the correct information.

Package Manager

For our website we decided that it would be best if we do not use the package manager.These are the reasons we had for not using it:

- **Minimal Dependencies:** Since our project is a small project with minimal external dependencies, managing these dependencies manually was straightforward and did not warrant the overhead of a package manager.
- **Quick Prototyping:** For quick prototyping or proof-of-concept projects, manually adding a few libraries was faster than setting up a package manager in our case.
- **Security Concerns:** Directly managing dependencies provided tighter control over which libraries and versions we could use, potentially reducing the risk of introducing vulnerabilities from third-party packages.