PRACTICAL 5

GROUP: NOTHING BUT HOOPS

Members:

- Z (Zainab) Abdulrasaq
- P (Paballo) Diyase
- C (Christopher) Katranas
- R (Rudolph) Lamprecht
- S (Samkelekile) Ndevu
- TR (Tshireletso) Sebake
- AZ (Arnaud) Strydom

Task 1

Research on the Entertainment Industry:

Current trends, popular preferences, and various genres.

INTRODUCTION:

The movie and television industry has been a cornerstone of entertainment for over a century, continuously evolving with advances in technology and changing audience tastes. This report examines the current trends in the industry, the influence of movies and TV series on these trends, popular genres, and how content is categorized and rated. Additionally, it explores how audience preferences and user reviews shape content recommendations.

INDUSTRY TRENDS:

The entertainment industry has seen significant shifts from traditional viewing methods to digital platforms. Historically, movies were primarily viewed in cinemas, then on DVDs and television. Today, streaming services have become the dominant medium for consuming both movies and TV series, driven by their convenience and on-demand accessibility. From 2015 to 2021, digital mediums have consistently grown, with a notable 15% spike in streaming from 2019 to 2020 due to the COVID-19 pandemic .

This transition has not only increased accessibility but also allowed studios to utilize data analytics for customizing content based on audience preferences. There's a growing trend towards niche content catering to specific interests and marginalized groups, and interactive experiences like choose-your-own-adventure storylines are becoming increasingly popular .

POPULAR GENRES:

A 2023 study highlighted that adventure is the most popular genre overall in movies and TV series. This genre's appeal lies in its exciting challenges and personal growth journeys. However, when focusing on TV series alone, drama emerges as the most popular genre due to its detailed storytelling and character-driven plots that explore various life conflicts .
In movies, action and adventure genres dominate, offering short-term excitement and entertainment through intense physical activities and explorative narratives. Other popular genres include horror, thriller, comedy, and sci-fi. Horror films aim to induce fear, thrillers focus on crime and danger, comedies provide humor and lightheartedness, and sci-fi explores imaginative scientific concepts .
In addition to these, several other genres have consistently drawn large audiences:

- Science fiction and fantasy: These genres captivate viewers with their creative realms and grand narratives.

- Suspenseful crime/thriller: Tales with elaborate storylines and multifaceted characters always maintain the attention of viewers.

- Comedy: Ranging from cheerful sitcoms to black comedies, this genre continues to be popular with audiences.

- Prestigious dramas: There is increasing demand for dramas that delve into social issues, family dynamics, and human relationships.

- Animation: This genre, including content for adults as well as family-friendly options, is making a comeback and is popular among all age groups .

Viewer preferences have also shown a demand for authenticity and inclusivity, with audiences seeking representation of diverse cultures, ethnicities, and sexual orientations. High production quality, compelling characters, and realistic narratives are crucial for capturing viewer interest .

CATEGORISATION AND RATING OF CONTENT:

Content categorization and rating are vital for guiding audience choices. The Motion Picture Association (MPA) provides a standardized film rating system, ranging from G (general audiences) to NC-17 (no one 17 and under admitted), primarily used for American productions . Similarly, the Film and Publication Board (FPB) in South Africa offers ratings to protect children from potentially harmful content, ranging from A (all ages) to XX (prohibited content) .

Ratings are based on factors such as violence, language, nudity, and prejudices, ensuring viewers can make informed decisions about what to watch. Additionally, platforms like IMDb offer ratings generated from user reviews, which significantly influence content recommendations.

AUDIENCE PREFERENCES, USER REVIEWS, AND RECOMMENDATIONS:

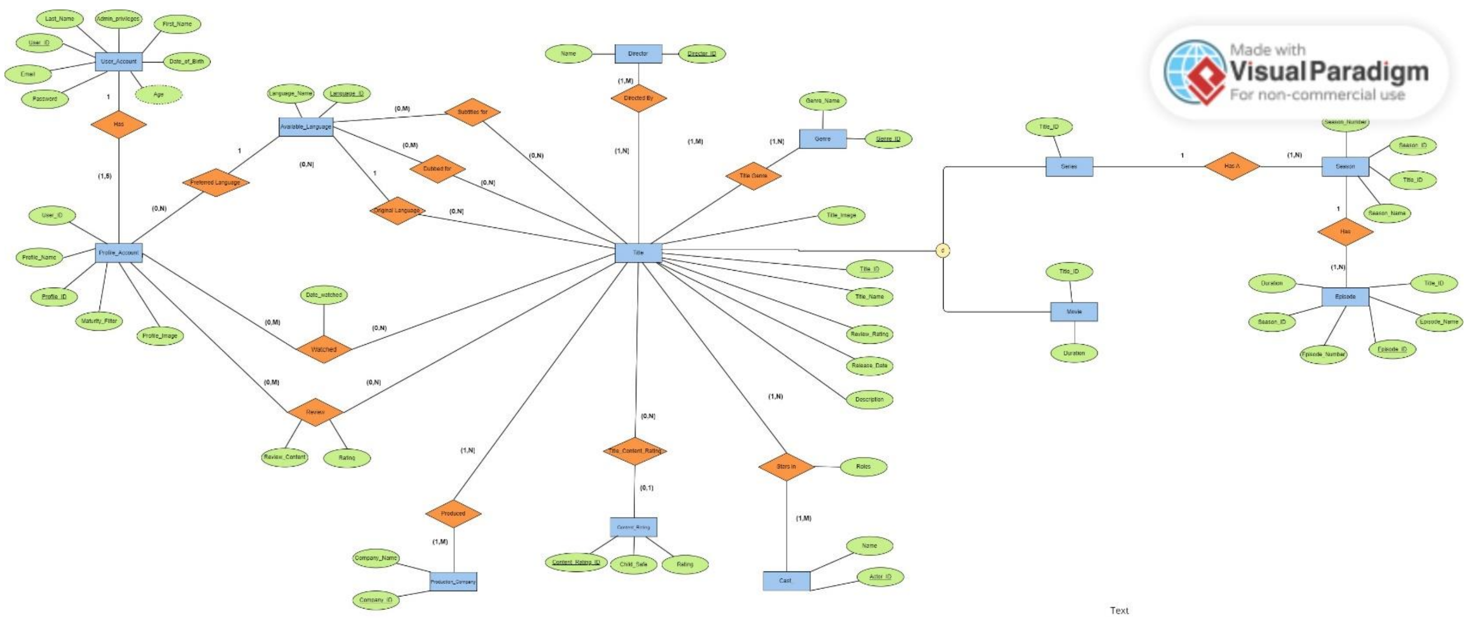
Understanding audience preferences is crucial for the entertainment industry. Viewers tend to favor high-quality content that is engaging and unpredictable. Streaming platforms leverage user data to recommend content based on individual preferences, heavily influenced by user reviews. These reviews, reflecting general audience opinions, often have a more positive impact on popularity compared to professional critic reviews .

In conclusion, the entertainment industry is continually adapting to new technologies and audience preferences. The rise of streaming services, the demand for diverse and high-quality content, and the influence of user reviews are shaping the future of movies and TV series. By staying attuned to these trends, the industry can continue to thrive and meet the evolving tastes of its audience.

REFERENCES:

- Miquido. "Current Trends in Media and Entertainment Industry: Guide for 2024" https://newmiquido.tech/entertainment-app-development/
- Statista: "Most popular movie genres in the U.S. 2021"
- Nielsen: "Streaming in the U.S. – 2021"
- PwC: "Entertainment & Media Outlook 2021-2025”
- https://sites.dwrl.utexas.edu/visualrhetoric/2016/04/04/films-medium-evolution/#:~:tex t=In%20many%20ways%2C%20film%20as,its%20most%20freely%20accessible%20 form
- https://www.statista.com/statistics/1194522/box-office-home-and-mobile-video-entert ainment-revenue-worldwide/
https://www.statista.com/markets/417/topic/476/tv-video-film/#statistic3
- https://www.motionpictures.org/film-ratings/
- https://www.fpb.org.za/classification/https://www.parrotanalytics.com/insights/genre-trend

Task 2

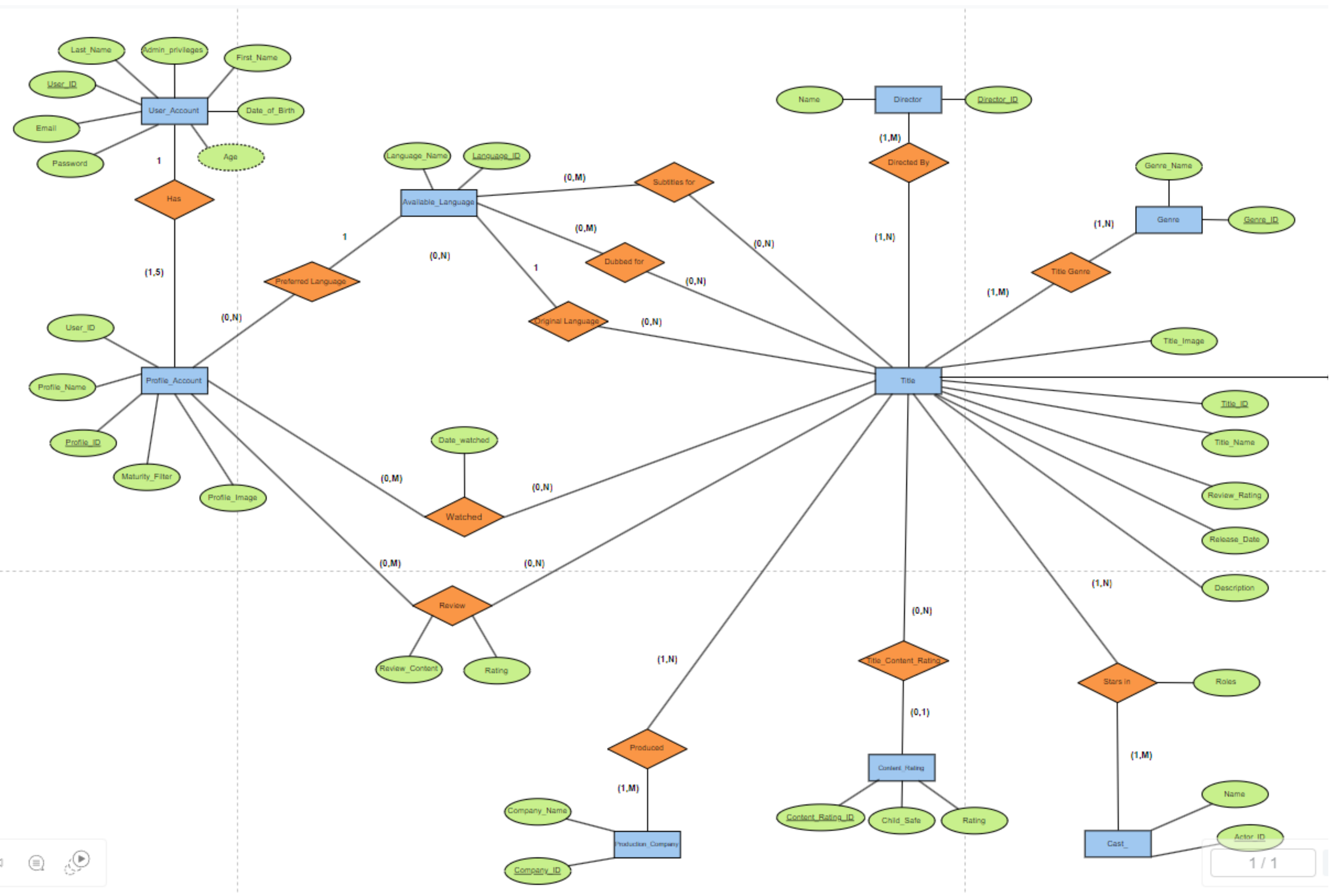


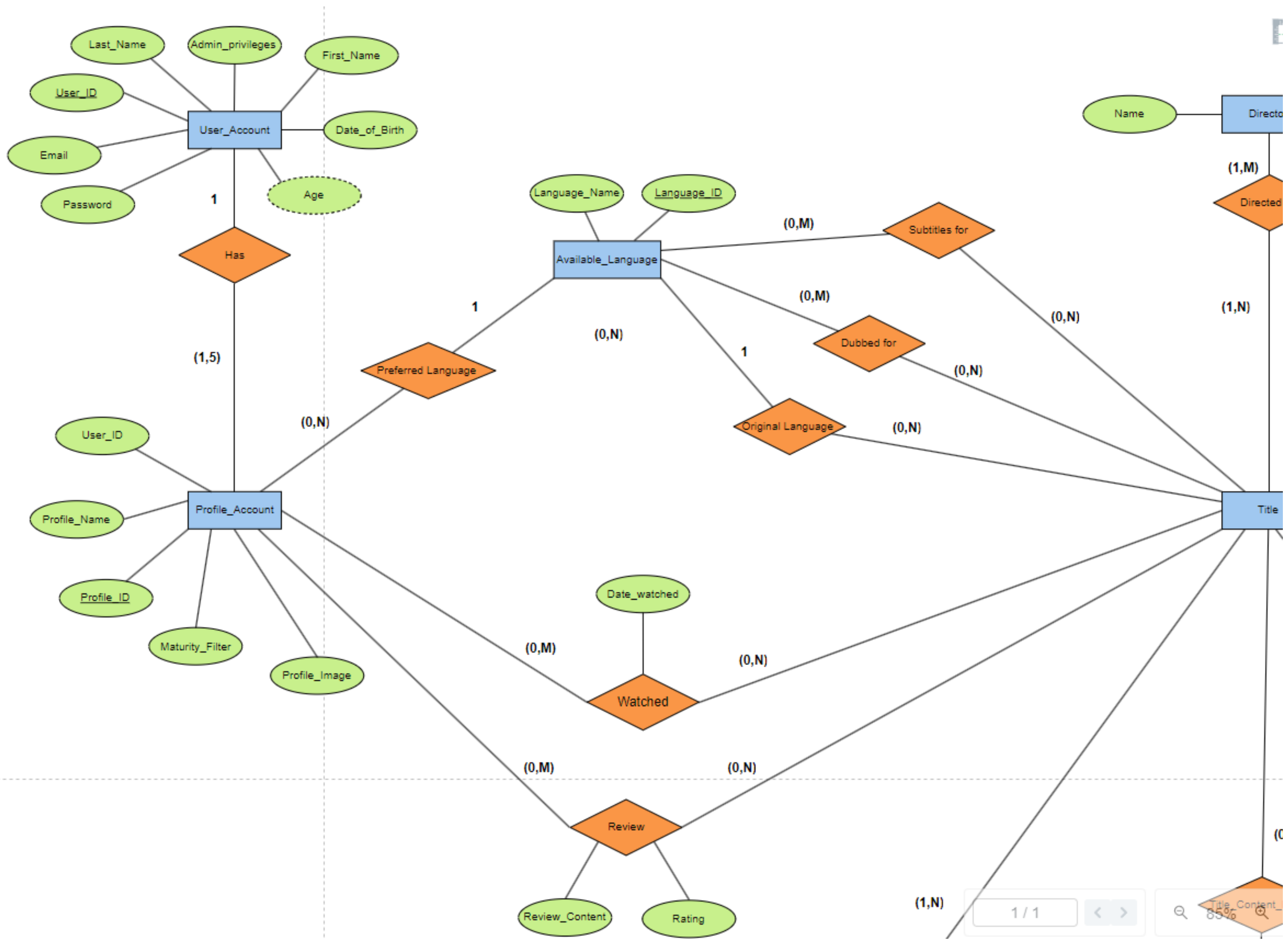
Starting with users, to ensure editing can only be done by admins, user has a bool to ensure only admins can see and use this feature. We made the assumptions that each user will have multiple profiles from one to five. These profiles have a language preference as well as a maturity filter. These are fields that will add filters to the sure to ensure that they are recommended content that is in their language and/ or appropriate for children when being recommended content.

For our ERD, we ensured to design it so that searching though the database would be as easy and efficient as it could be. We did this by having filter topics (such as genre, language, age rating) be in a relational table. This insures that search options will always be known before hand.

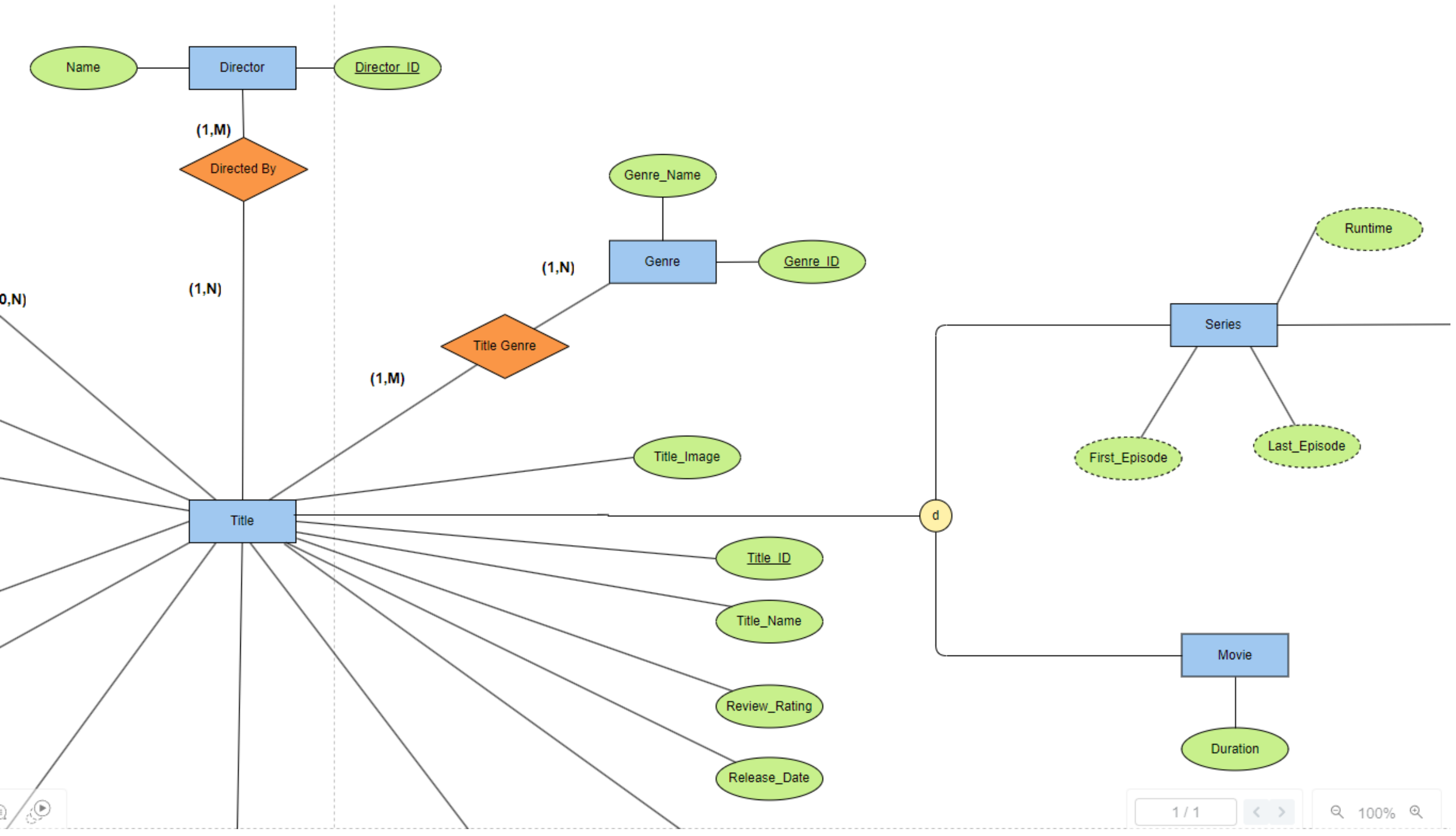
To ensure that editing of details are reliable and efficient, production studios, genres, cast members and the content rating are related to one another and are not attributes in the titles table. This ensures that if editing is required (an actor's name changes for example), they will be consistent with the rest of the entries.

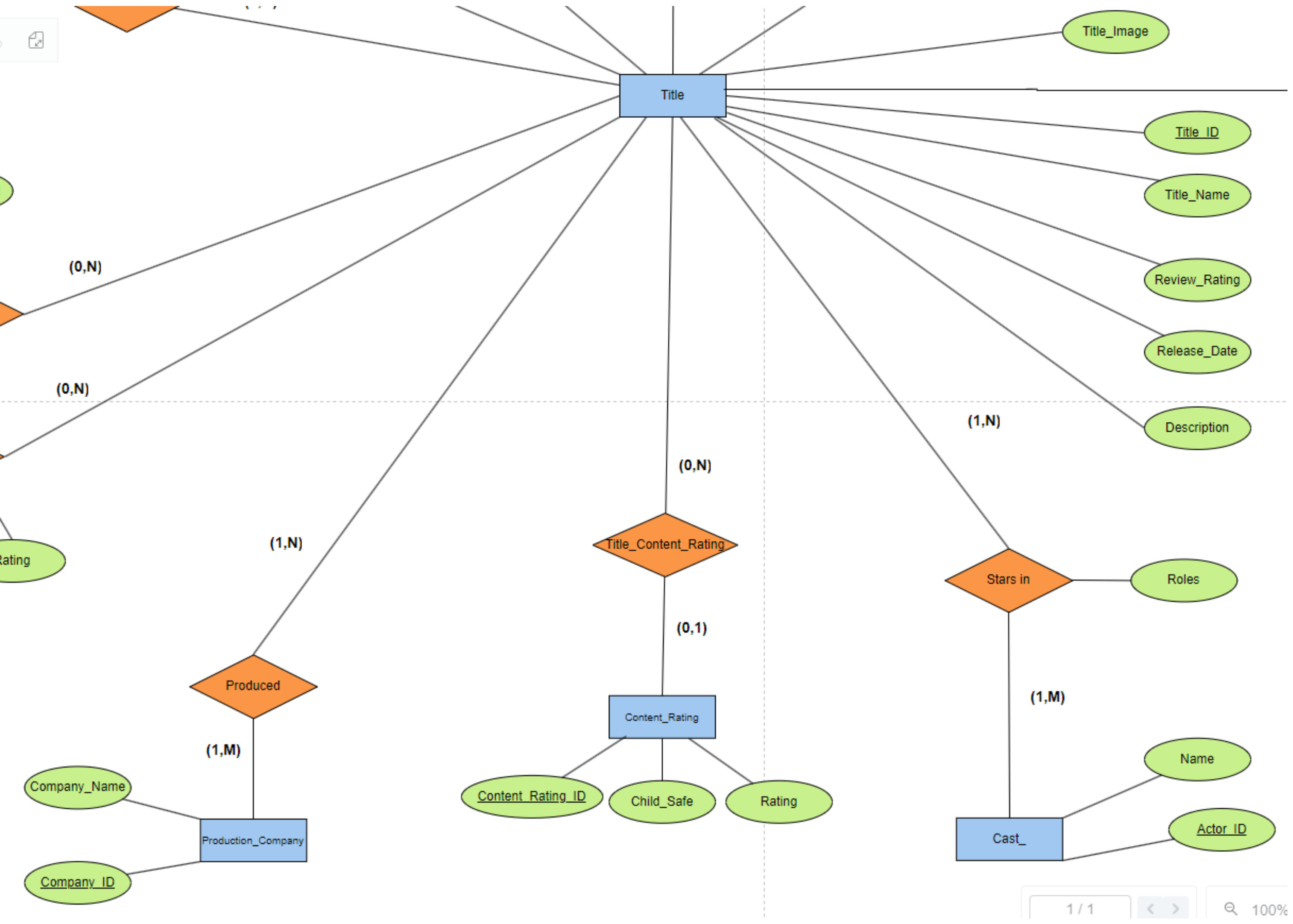
<u>Better view of diagram portions:</u>

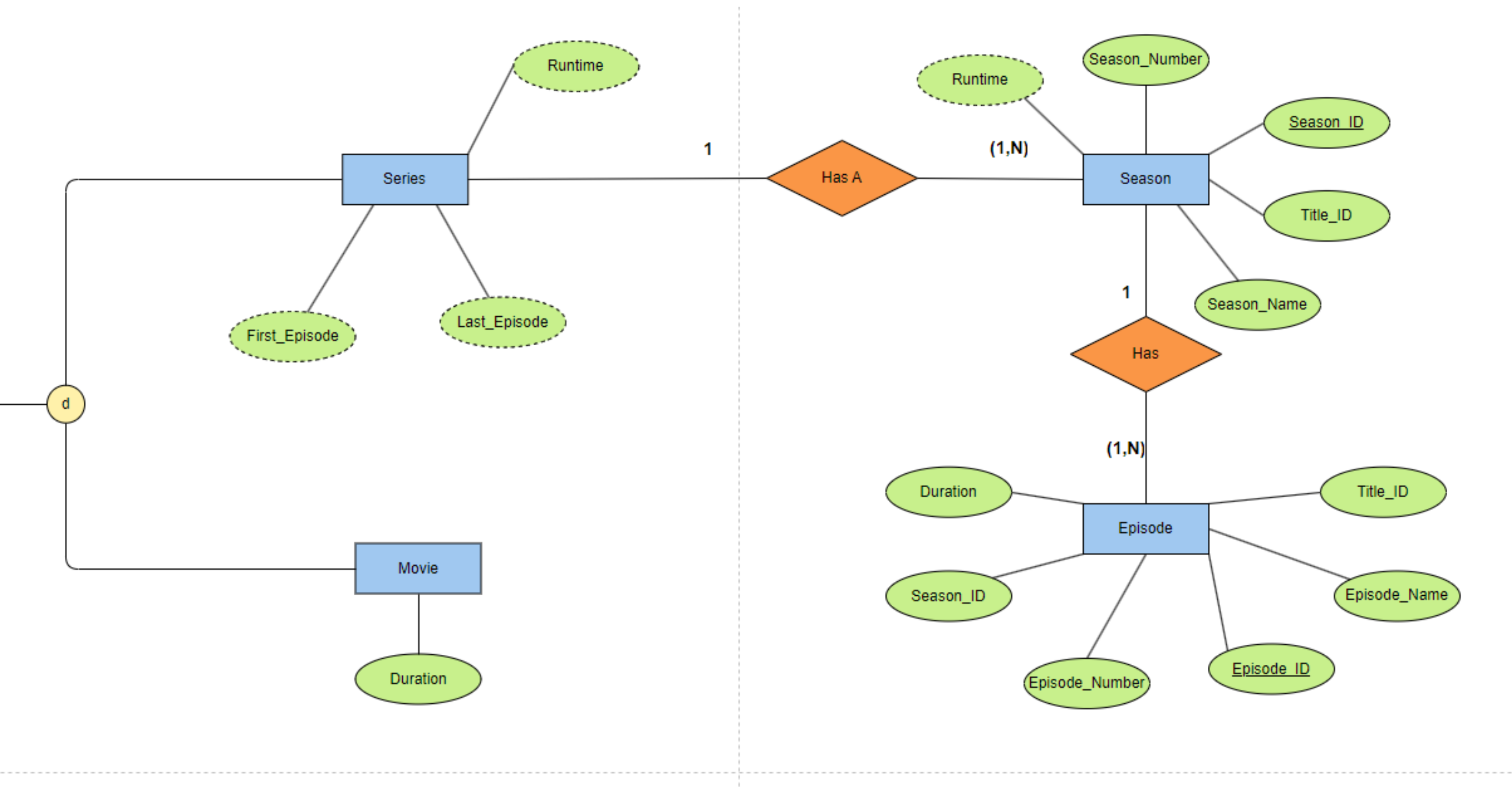




Last_Name
Admin_privileges
First_Name
User_ID
User_Account
Date_of_Birth
Email
Password
Age
1
Has
(1,5)
User_ID
Profile_Account
Profile_Name
Profile_ID
Maturity_Filter
Profile_Image
Language_Name
Language_ID
Available_Language
(0,M)
Subtitles for
1
(0,N)
Preferred Language
(0,N)
(0,M)
Dubbed for
1
(0,N)
(0,N)
Original Language
(0,N)
(0,N)
Name
Directo
(1,M)
Directed
(1,N)
Title
Date_watched
(0,M)
(0,N)
Watched
(0,M)
(0,N)
Review
Review_Content
Rating
(1,N)
Title_Content
1 / 1
85%

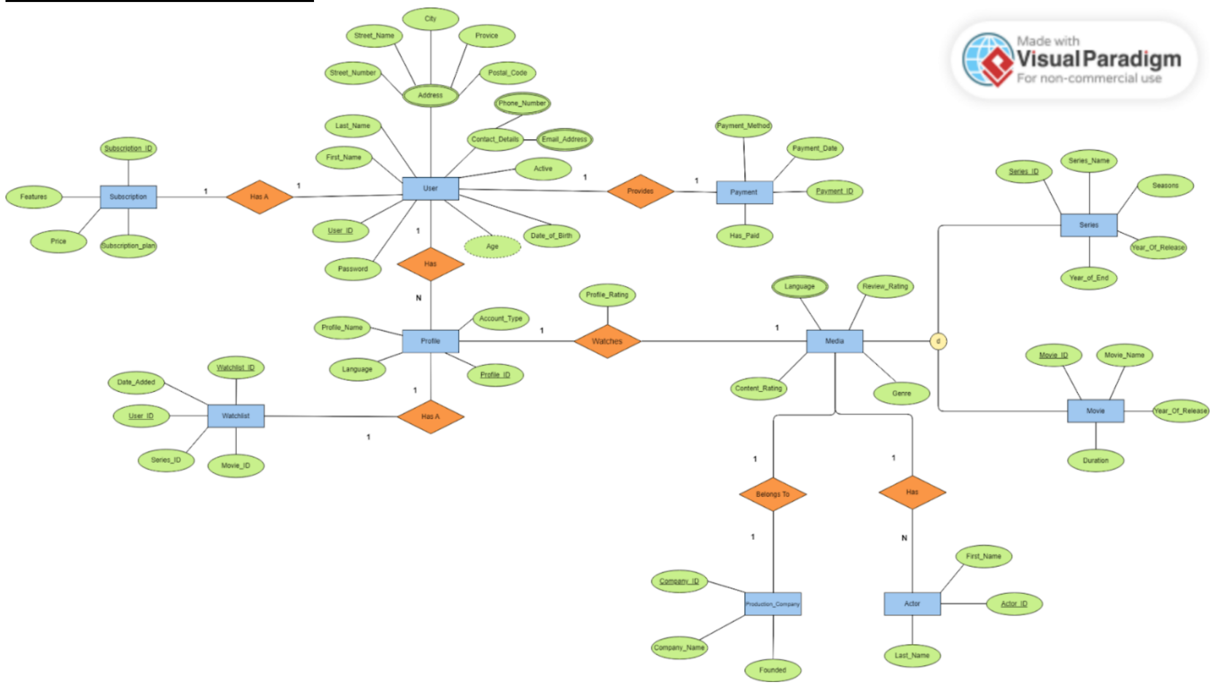

Name
Director
Director_ID
(1,M)
Directed By
(1,N)
0,N)
Genre_Name
Genre
Genre_ID
(1,N)
Title Genre
(1,M)
Runtime
Series
First_Episode
Last_Episode
Title_Image
Title
d
Title_ID
Title_Name
Review_Rating
Release_Date
Movie
Duration
1 / 1
100%


Title
Title_Image
Title_ID
Title_Name
Review_Rating
Release_Date
Description
(0,N)
(0,N)
(1,N)
(0,N)
(1,N)
Rating
Title_Content_Rating
Stars in
Roles
(0,1)
Produced
(1,M)
(1,M)
Content_Rating
Company_Name
Content_Rating_ID
Child_Safe
Rating
Name
Actor_ID
Production_Company
Cast_
Company_ID
1 / 1
100%


Runtime
Series
First_Episode
Last_Episode
d
Movie
Duration
1
Has A
(1,N)
Runtime
Season_Number
Season_ID
Season
Title_ID
Season_Name
1
Has
(1,N)
Duration
Episode
Title_ID
Season_ID
Episode_Name
Episode_Number
Episode_ID

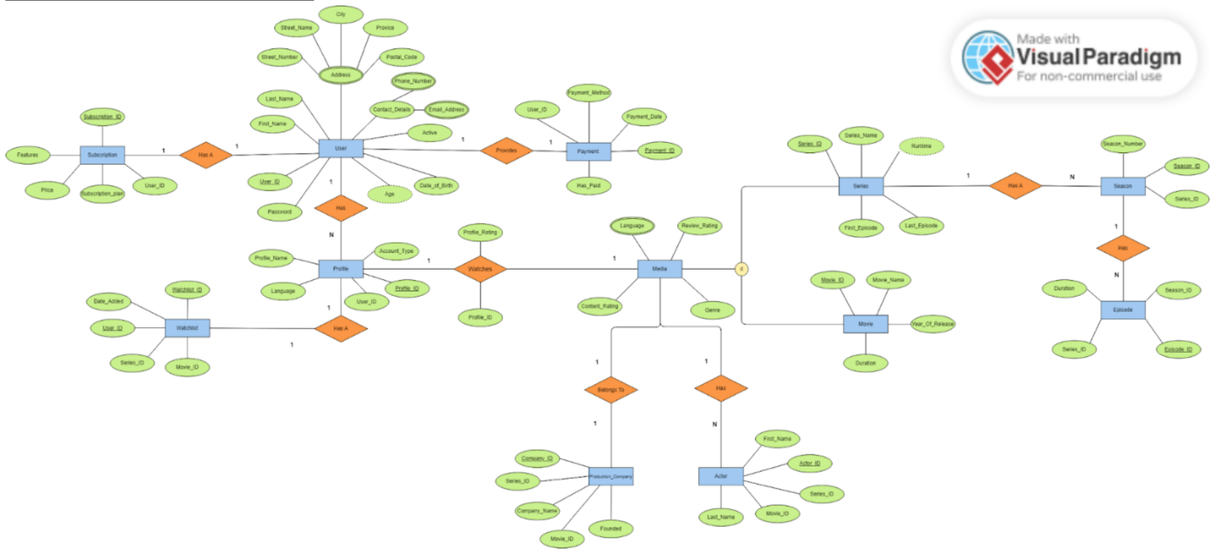
Iterations of the diagram:

ITERATION 1



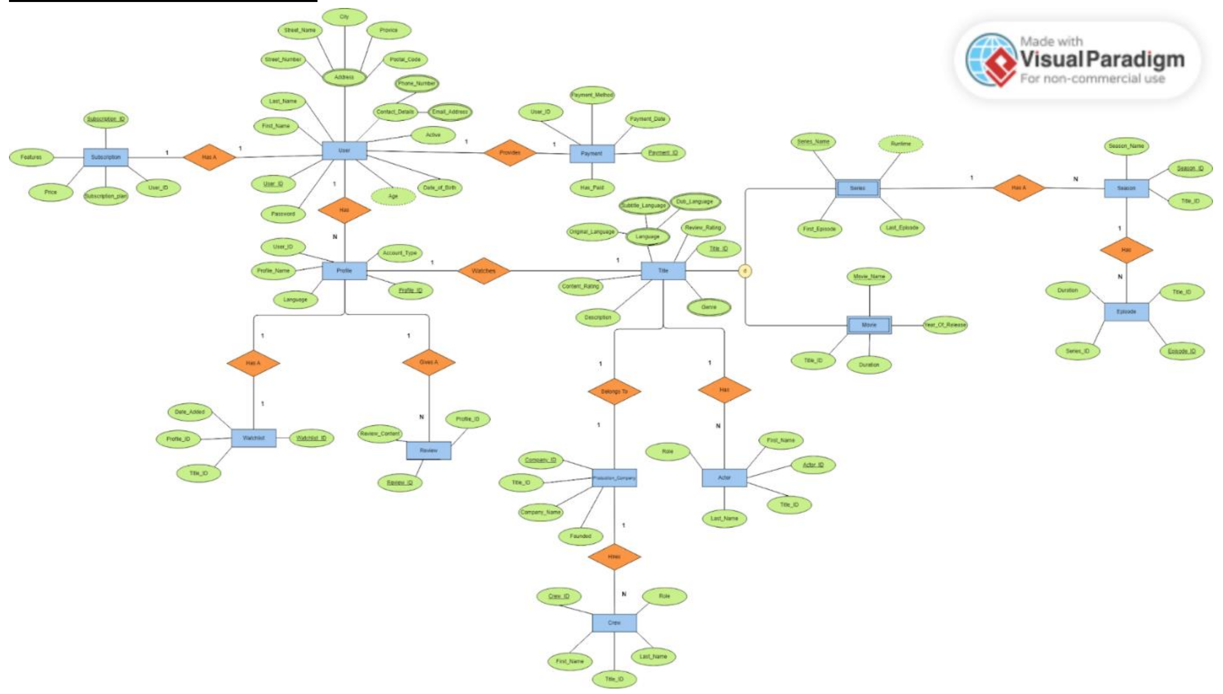
- This is the first iteration of the ERD.

ITERATION 2



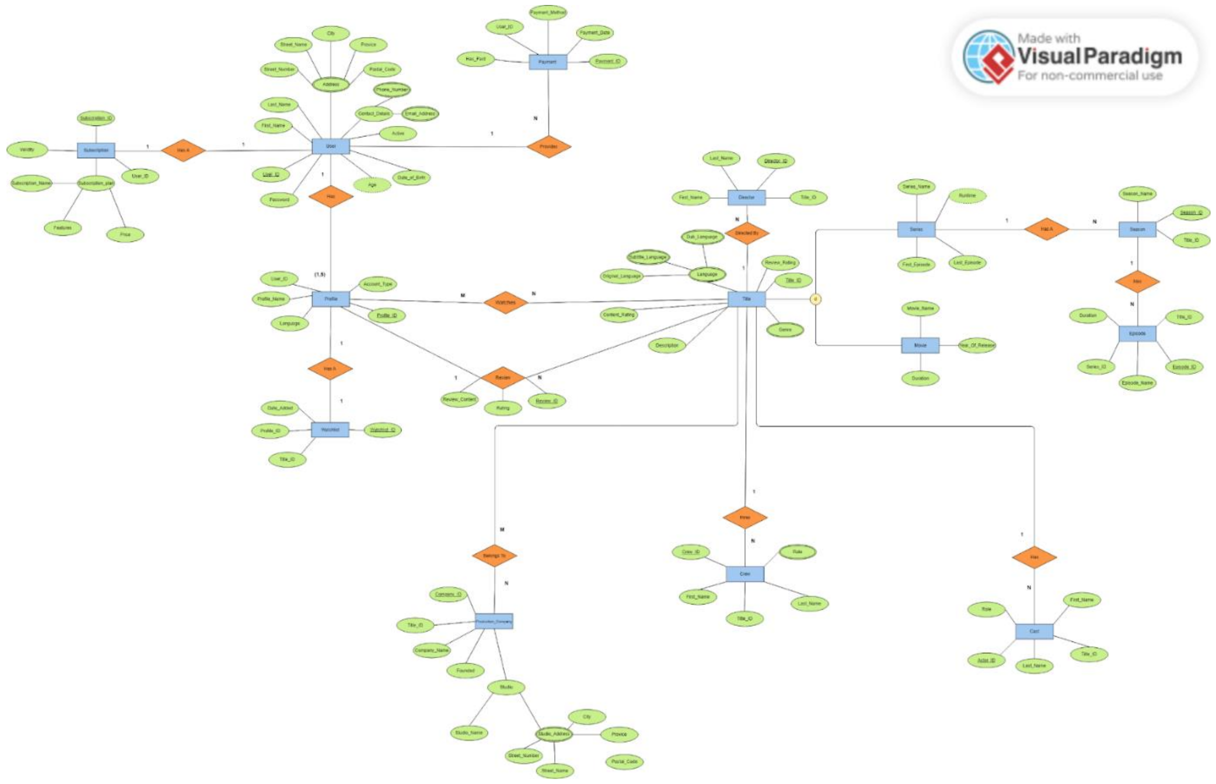
- Foreign keys were added
- Season and Episode entities were added

ITERATION 3



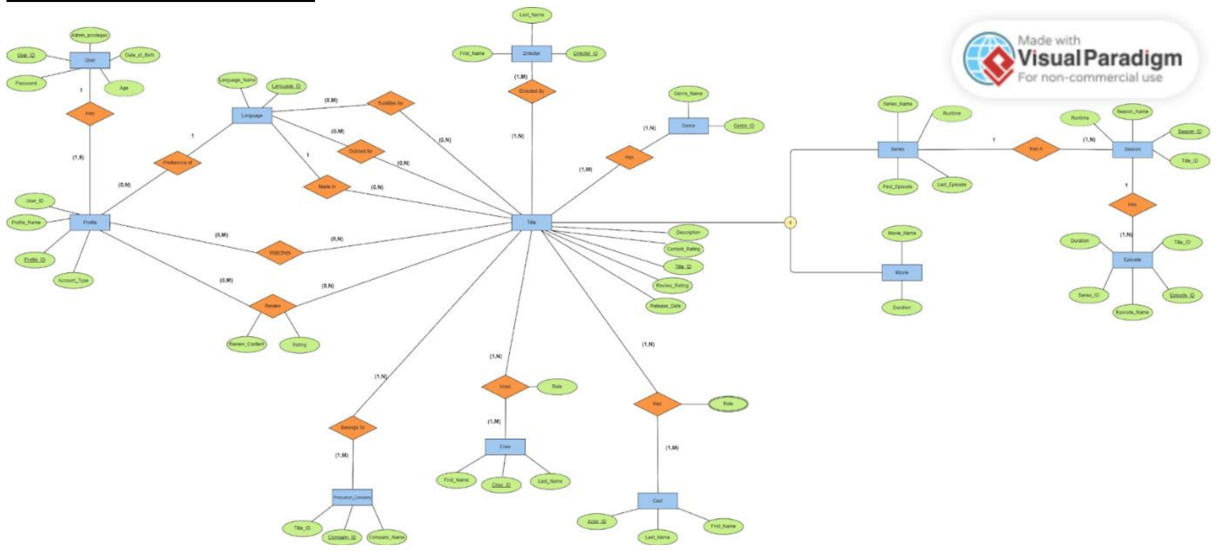
- Media title is changed to Title and an attribute called Title_ID is added.
- Series and Movie are changed to weak attributes with Series_Name and Movie_Name as weak key attributes.
- A crew entity is added and it shows all the information of the crew members that are hired for a title by a production company.
- Review entity is added with attributes that give details about the review a profile makes for a title.
- Languages attribute is made into a composite attribute with other attributes to better describe it.

ITERATION 4



- Crew was changed to have a relationship with Title instead of with Production_Company.
- The composite attribute Studio was added,it contains the address of the studio and the name of the studio.
- Title now has a relationship with the director entity.
- Changes were made to the cardinality of the Has A relationship between User and Profile.
- Subscription Plan is changed to a composite attribute.

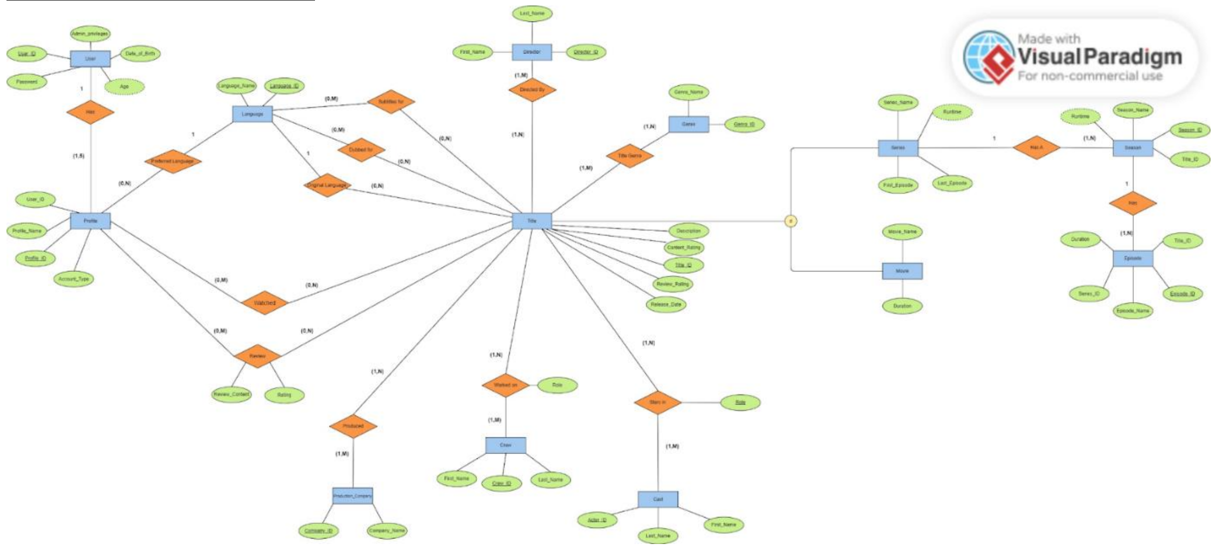
ITERATION 5



- Some attributes have been removed to better focus on functional requirements, with bare bones for functionality in mind. (this includes payments, subscriptions, details of the production studio etcetera.)
- Cardinality has been better defined for all relationships, as well as the addition of structural constraints.
- Languages has been put in its own table since its an option for a profile to choose from existing languages.
- Languages now relates to the titles of movies/ TV shows instead of being a misused composite attribute. The relations are the language it was made in, the subtitle languages it is available in and the dubs it is available in.

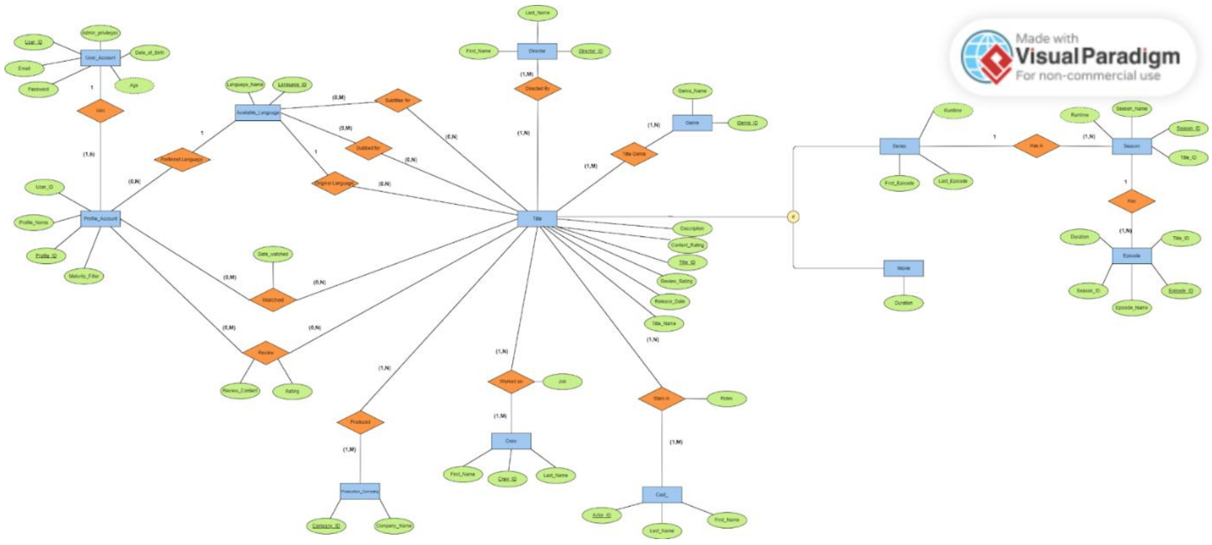
- Roles for cast and crew respectively has been moved to the N,M relation. Before the notation was not functional and added role to the details of the crew/ cast member.
- Genre has been given its own table and a relation has been added between it and the title. This help ensure that users will only be able to search for already existing genres (the same is now available for searches for language options)
- Since searching by release date is a functional requirement for both movies and vs series, it has been added to the title entity and removed from movies.
- Added admin privileges as an attribute to the user entity. This is to separate the normal functionality of the database to the functionality for editing/ manipulating the database.

ITERATION 6



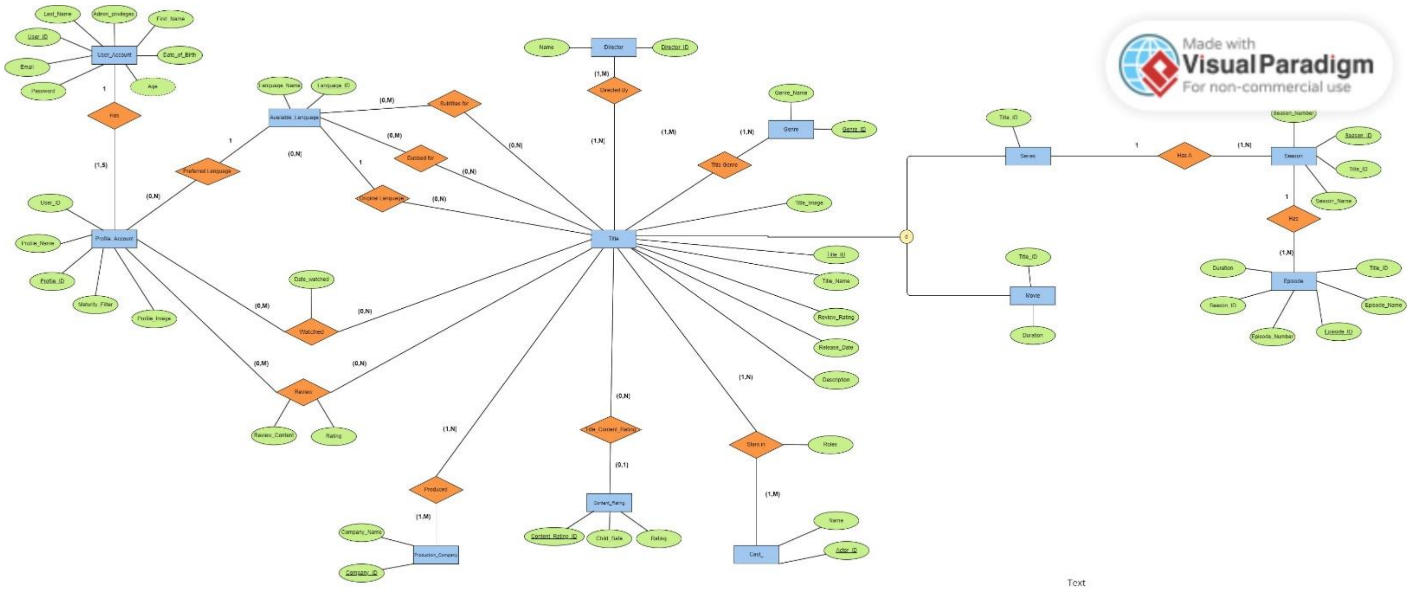
- Name changes of relations to better describe them
- Change of role in the Stars in relation from a multivalued attribute to a key attribute. This is because an actor can play multiple roles for a title, and enables the data to be in one table, while still having tuples uniquely identifiable via their primary key.

ITERATION 7



- Added a date to the watched relation to add the feature for recommending movies and shows that are popular over a time frame
- Added email attribute to the user entity so users have a way to log in
- Changed role to roles in the relation of title and cast. This is because an actor that does multiple roles will be separated by a comma instead of having an individual tuple for each role they played.
- Review rating is now derived.
- Title names have been added to the titles and movie names and series names have been removed.
- Changed some attribute names to ones that don't conflict with the database language.

ITERATION 8 Final



- Added season and episode number.
- Combined first and last name in actors and directors to one attribute
- Removed crew since actor and director fall under these roles
- Removed subbed and dubbed_for
- Removed imposible derived attributes.

Task 3

Step 1: Mapping of regular entity types
We create the relations USER_ ACCOUNT, PROFILE_ ACCOUNT, AVAILABLE_LANGUAGE,TITLE, DIRECTOR, PRODUCTION_COMPANY, CAST_, GENRE,CONTENT_RATING, SEASON, and EPISODE

USER_ACCOUNT

User_ID	Admin_privileges	Date_of_Birth	Age	Email	Password

PROFILE_ACCOUNT

Profile_ID	Profile_Name	Maturity_Filter	Profile_image

AVAILABLE_LANGUAGE

Language_ID	Language_Name

TITLE

Title_ID	Title_Name	Content_ Rating	Review_Rating	Release_Date	Description	Title_image

DIRECTOR

Director_ID	Name

PRODUCTION_COMPANY

Company_ID	Company_Name

CAST_

Actor_ID	Name

GENRE

Genre_ID	Genre_Name

SEASON

Season_ID	Title_ID	Runtime

EPISODE

Episode_ID	Series_ID	Title_ID	Episode_Name	Duration

CONTENT_RATING

Content_Rating_ID	Child_Safe	Rating

Step 2: Mapping of weak entity types:
N/A , no weak entity types in the ER diagram

Step 3:Mapping of binary 1:1 relationships
N/A, no 1:1 relationshipls in the ER diagram

Step 4: Mapping of Binary 1:N Relationship Types

- For the HAS_PROFILE relationship, we include as foreign key the primary key of USER, User_ID in PROFILE .
- For the PREFERRED_LANGUAGE relationship, we include as foreign key the primary key of LANGUAGE, Language_ID in PROFILE.
- For the ORIGINAL_LANGUAGE relationship, we include as foreign key, the primary key of LANGUAGE, Language_ID in TITLE.
- For the HAS_SEASON relationship, we include as foreign key, the primary key of SERIES, Title_ID in SEASON.
- For the HAS_EPISODE relationship, we include as foreign key, the primary key of SEASON, Season_ID in EPISODE.
- For the TITLE_CONTENT_RATING relationship, we include as foreign key, the primary key of CONTENT_RATING, Content_Rating_ID in TITLE.

PROFILE_ACCOUNT

Profile_ID	User_ID	Profile_Name	Maturity_Filter	Profile_Image	Language_ID

TITLE

Title_ID	Content_Rating	Review_Rating	Release_Date	Description	Language_ID

SEASON

Season_ID	Title_ID	Season_Name	Runtime

EPISODE

Episode_ID	Series_ID	Title_ID	Episode_Name	Duration	Season_ID

Step 5: Mapping of Binary M:N Relationship Types

We create the relations:

- WATCHED and include as foreign keys, the primary key of PROFILE, Profile_ID, and the primary key of TITLE, Title_ID.
- REVIEW and include as foreign keys, the primary key of PROFILE, Profile_ID, and the primary key of TITLE, Title_ID.
- DIRECTED_BY and include as foreign keys, the primary key of DIRECTOR, Director_ID, and the primary key of TITLE, Title_ID.
- TITLE_GENRE and include as foreign keys, the primary key of TITLE, Title_ID, and the primary key of GENRE, Genre_ID.
- WORKED_ON and include as foreign keys, the primary key of CREW, Crew_ID, and the primary key of TITLE, Title_ID.
- STARS_IN and include as foreign keys, the primary key of CAST, Actor_ID, and the primary key of TITLE, Title_ID.
- PRODUCED and include as foreign keys, the primary key of PRODUCTION_COMPANY , Company_ID, and the primary key of TITLE, Title_ID.

WATCHED

Profile_ID	Title_ID	Date_Watched

REVIEW

Profile_ID	Title_ID	Review_Content	Rating

DIRECTED_BY

Director_ID	Title_ID

TITLE_GENRE

Title_ID	Genre_ID

STARS_IN

Actor_ID	Title_ID	Roles

PRODUCED

Company_ID	Title_ID

Step 6: Mapping of Multivalued Attributes

N/A, there are no multivalued attributes in the ER diagram.

Step 7: Mapping of N-ary Relationship Types

N/A, there are no n-ary relationship types in the ER diagram.

Step 8: Mapping specialisationand generalisation

- We are using Option 8A: Multiple relations -superclass and subclasses
- We create the relations , SERIES and MOVIE and include the primary key of the superclass TITLE, Title_ID.

SERIES

Title_ID

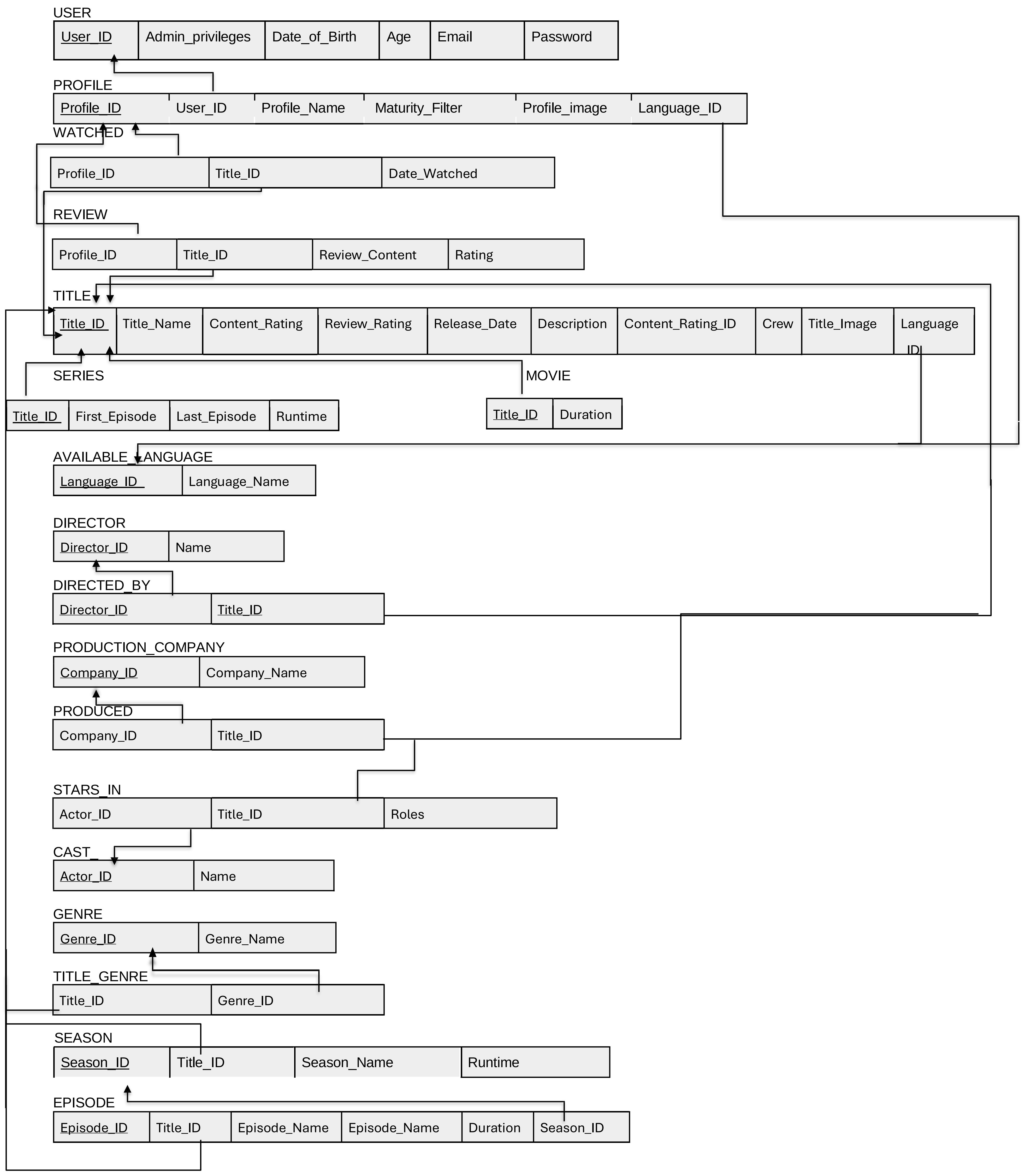
MOVIE

Title_ID	Duration

Step 9: Mapping of Union Types

N/A, there are no union types in the ER diagram.

FINAL RESULT OF MAPPING THE HOOPS SCHEMA INTO A RELATIONAL DATABASE SCHEMA


USER
User_ID | Admin_privileges | Date_of_Birth | Age | Email | Password
PROFILE
Profile_ID | User_ID | Profile_Name | Maturity_Filter | Profile_image | Language_ID
WATCHED
Profile_ID | Title_ID | Date_Watched
REVIEW
Profile_ID | Title_ID | Review_Content | Rating
TITLE
Title_ID | Title_Name | Content_Rating | Review_Rating | Release_Date | Description | Content_Rating_ID | Crew | Title_Image | Language ID
SERIES
Title_ID | First_Episode | Last_Episode | Runtime
MOVIE
Title_ID | Duration
AVAILABLE_LANGUAGE
Language_ID | Language_Name
DIRECTOR
Director_ID | Name
DIRECTED_BY
Director_ID | Title_ID
PRODUCTION_COMPANY
Company_ID | Company_Name
PRODUCED
Company_ID | Title_ID
STARS_IN
Actor_ID | Title_ID | Roles
CAST_
Actor_ID | Name
GENRE
Genre_ID | Genre_Name
TITLE_GENRE
Title_ID | Genre_ID
SEASON
Season_ID | Title_ID | Season_Name | Runtime
EPISODE
Episode_ID | Title_ID | Episode_Name | Episode_Name | Duration | Season_ID

Assumpitons

- In step 4, when mapping 1: N, we used the foreign key method because it reduces the number of tables.
- In step 8, when mapping specialization and generalization, we used option 8A: Superclass – subclass because this option works for any specialization (total or partial, disjoint or overlapping).

Task 4

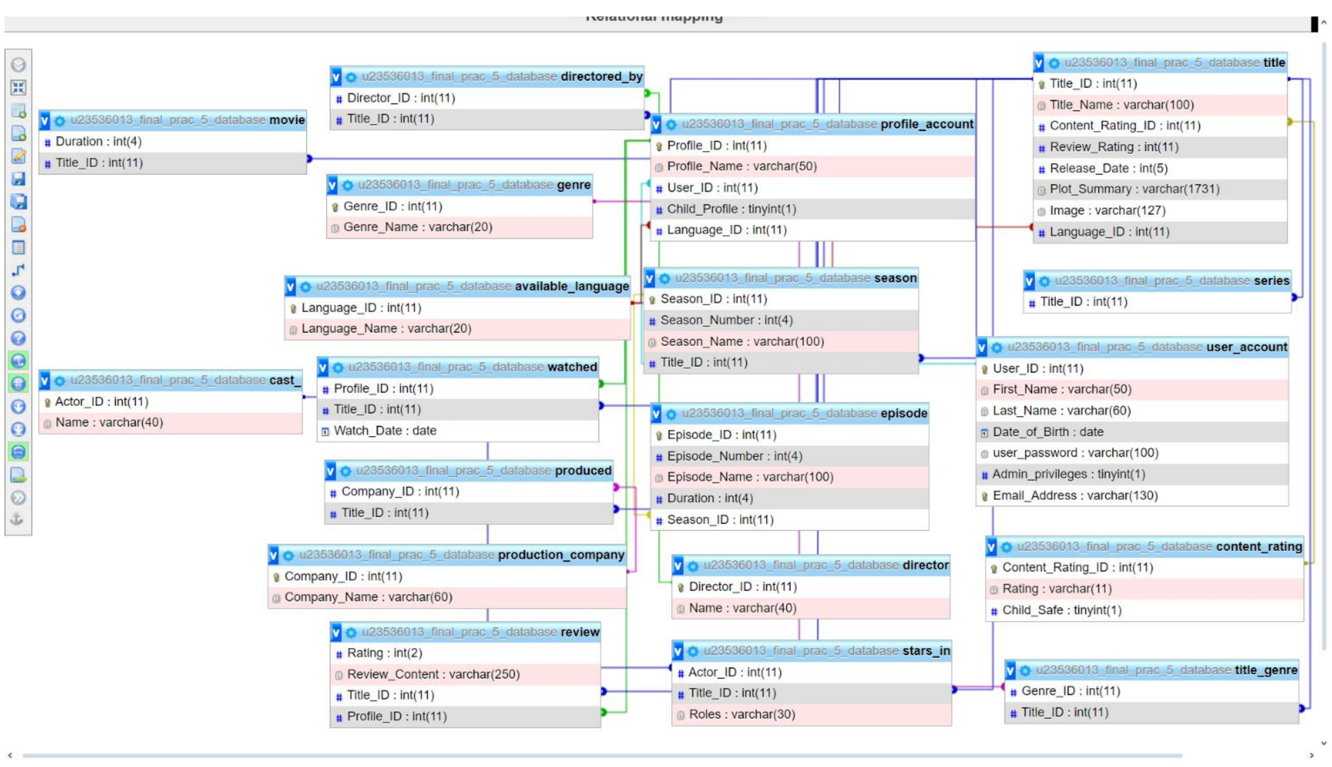
Code for initial database setup

```
-- create database Hoop_DB;
-- use Hoop_DB;
-- drop database Hoop_DB;



-- language table
create table Available_Language
(
    Language_ID int auto_increment primary key,
    Language_Name varchar(20) not null

);


-- table with simple user data,complex and composite get their own tables
create table User_Account
(
    User_ID int auto_increment primary key,
    First_Name varchar(50)not null,
    Last_Name varchar(60) not null,
    Date_of_Birth date not null,
    user_password varchar(100)not null,
    Admin_privileges boolean,
     Email_Address varchar(130),
      constraint user_email unique lower(Email_Address), -- lowercase
    constraint email_format check (Email_Address like '%@%.%') -- format email
someaddress@something.com
);


-- profile table
create table profile_account
(
    Profile_ID int auto_increment primary key,
    Profile_Name varchar(50)NOT NULL,

    User_ID int,
    Child_Profile boolean,
    Language_ID int,

    foreign key (User_ID) references user_account(User_ID),
    foreign key (Language_ID) references Available_Language(Language_ID)
);

-- update from here

create table Content_Rating
```

```sql
(
    Content_Rating_ID int auto_increment primary key,
    Rating varchar(11),
    Child_Safe boolean
);

create table Title
(
    Title_ID int auto_increment primary key,
    Title_Name varchar(100),
    Content_Rating_ID int,
    Review_Rating int(11),
    Release_Date int(5),
    Plot_Summary varchar(1731),
    Crew varchar(100),
    Image varchar(127),
    Language_ID int,

    foreign key (Content_Rating_ID) references
Content_Rating(Content_Rating_ID),
    foreign key (Language_ID) references Available_Language(Language_ID)

);


-- dubbed_for language table
create table Dubbed_For
(
    Language_ID int ,
    Title_ID int,

    foreign key (Language_ID) references Available_Language(Language_ID),
    foreign key (Title_ID) references Title(Title_ID)


);

-- subtitles_for language table
create table Subtitles_For
(
    Language_ID int,
    Title_ID int,

    foreign key (Language_ID) references Available_Language(Language_ID),
    foreign key (Title_ID) references Title(Title_ID)


);

create table Series
(
    Title_ID int,
    foreign key (Title_ID) references Title(Title_ID)
);


create table Season
(
    -- link to series and title
    Season_ID int auto_increment primary key,
    Season_Number int(4),
    Season_Name varchar(100),
    Title_ID int,

    foreign key (Title_ID) references Title(Title_ID)

);

create table Episode
(
    -- link to season and title
    Episode_ID int auto_increment primary key,
    Episode_Number int(4),
    Episode_Name varchar(100),
    Duration int(4),

    Season_ID int,

    foreign key (Season_ID) references Season(Season_ID)
);

create table Movie
(

    Duration int(4),
    Title_ID int,

    foreign key (Title_ID) references Title(Title_ID)
);

create table Review
(
    Rating int(2),
    Review_Content varchar(250),
    Title_ID int,
    Profile_ID int,


    -- foreign keys
     foreign key (Profile_ID) references profile_account(Profile_ID),
     foreign key (Title_ID) references Title(Title_ID)
);
```

```sql
-- update from here

-- watched table
create table Watched
(
    Profile_ID int,
    Title_ID int,
    Watch_Date date,

    foreign key (Profile_ID) references Profile_Account(Profile_ID),
    foreign key (Title_ID) references Title(Title_ID)


);



-- production company table
create table Production_Company
(
    Company_ID int auto_increment primary key,
    Company_Name varchar(20) not null

);

-- produced table
create table Produced
(
    Company_ID int,
    Title_ID int,

    foreign key (Company_ID) references Production_Company(Company_ID),
    foreign key (Title_ID) references Title(Title_ID)


);



-- cast table
create table Cast_
(
    Actor_ID int auto_increment primary key,
    Name varchar(40) not null

    -- check relationship


);

-- stars_in table
create table Stars_In
```

```sql
(
    Actor_ID int,
    Title_ID int,
    Roles varchar(30),

    foreign key (Actor_ID) references Cast_(Actor_ID),
    foreign key (Title_ID) references Title(Title_ID)

    -- check relationship

);


-- director table
create table Director
(
    Director_ID int auto_increment primary key,
    Name varchar(40) not null

    -- check relationship


);

-- directed_by table
create table Directored_By
(
    Director_ID int,
    Title_ID int,

    foreign key (Director_ID) references Director(Director_ID),
    foreign key (Title_ID) references Title(Title_ID)


    -- check relationship


);

-- genre table
create table Genre
(
    Genre_ID int auto_increment primary key,
    Genra_Name varchar(20) not null

    -- check relationship

);

create table Title_Genre
(
    Genre_ID int,
    Title_ID int,

    foreign key (Genre_ID) references Genre(Genre_ID),
    foreign key (Title_ID) references Title(Title_ID)



    -- check relationship

);
```



u23536013_final_prac_5_database **movie**
- Duration : int(4)
- Title_ID : int(11)

u23536013_final_prac_5_database **directored_by**
- Director_ID : int(11)
- Title_ID : int(11)

u23536013_final_prac_5_database **genre**
- Genre_ID : int(11)
- Genre_Name : varchar(20)

u23536013_final_prac_5_database **available_language**
- Language_ID : int(11)
- Language_Name : varchar(20)

u23536013_final_prac_5_database **cast_**
- Actor_ID : int(11)
- Name : varchar(40)

u23536013_final_prac_5_database **watched**
- Profile_ID : int(11)
- Title_ID : int(11)
- Watch_Date : date

u23536013_final_prac_5_database **produced**
- Company_ID : int(11)
- Title_ID : int(11)

u23536013_final_prac_5_database **production_company**
- Company_ID : int(11)
- Company_Name : varchar(60)

u23536013_final_prac_5_database **review**
- Rating : int(2)
- Review_Content : varchar(250)
- Title_ID : int(11)
- Profile_ID : int(11)

u23536013_final_prac_5_database **profile_account**
- Profile_ID : int(11)
- Profile_Name : varchar(50)
- User_ID : int(11)
- Child_Profile : tinyint(1)
- Language_ID : int(11)

u23536013_final_prac_5_database **season**
- Season_ID : int(11)
- Season_Number : int(4)
- Season_Name : varchar(100)
- Title_ID : int(11)

u23536013_final_prac_5_database **episode**
- Episode_ID : int(11)
- Episode_Number : int(4)
- Episode_Name : varchar(100)
- Duration : int(4)
- Season_ID : int(11)

u23536013_final_prac_5_database **director**
- Director_ID : int(11)
- Name : varchar(40)

u23536013_final_prac_5_database **stars_in**
- Actor_ID : int(11)
- Title_ID : int(11)
- Roles : varchar(30)

u23536013_final_prac_5_database **title**
- Title_ID : int(11)
- Title_Name : varchar(100)
- Content_Rating_ID : int(11)
- Review_Rating : int(11)
- Release_Date : int(5)
- Plot_Summary : varchar(1731)
- Image : varchar(127)
- Language_ID : int(11)

u23536013_final_prac_5_database **series**
- Title_ID : int(11)

u23536013_final_prac_5_database **user_account**
- User_ID : int(11)
- First_Name : varchar(50)
- Last_Name : varchar(60)
- Date_of_Birth : date
- user_password : varchar(100)
- Admin_privileges : tinyint(1)
- Email_Address : varchar(130)

u23536013_final_prac_5_database **content_rating**
- Content_Rating_ID : int(11)
- Rating : varchar(11)
- Child_Safe : tinyint(1)

u23536013_final_prac_5_database **title_genre**
- Genre_ID : int(11)
- Title_ID : int(11)

Task 5

Functional requirements

Users should be able to:

1. Log in
2. Create user profiles (and profiles for users)
3. User Account and Profile Management
4. Sort and Filter movies and TV shows by:
   . Genre
   . Ratings
   . Release dates

   . Language

2. View multiple Movies Series and Recommended movies and series.

3. View title details:
   . Cast
   . Plot summaries
   . User reviews
   a. Title poster
   b. Directors
   c. Production studios
   d. Genres
   e. Language
   f. Content Rating
2. Rate and review titles
3. Share favourite content through WhatsApp , Email and SMS.

admins should be able to

0. Manage users
0. Add, edit, and delete Movies and TV series
0. Manage actors, directors, genres and production studios(Update , Edit , Delete)
0. Updated the database
0. Recommend movies and tv series based on criteria.

Profile can be child or normal (changes recommended content).
Child profile can not view or submit ratings or reviews.
Child profile can only view content that is rated for children.
The age for a child profile is any person under the age of 13.

Task 6

Titles, (movies and series), as well as the cast and directors were acquired from https://www.kaggle.com/datasets/dgoenrique/netflix-movies-and-tv-shows/data?select=credits.csv . The images were scraped using the titles name via the “justwatch” website. We strived to have data as accurate as possible so this data was formatted using regex statements so it could be added to a database table. Once in this table, many sql statements were used to usure all sql table were filled appropriately to ensure foreign keys values were valid.

The only tables without existing extrapolated data were the languages, users, profiles, production studios, seasons, episodes, and review data. These were manually entered.

Task 7

Query Optimization Report

Introduction:

The purpose of this report is to optimize the searchMovieTitles query in our application. This query was found to be inefficient during performance testing.

Execution Plan Analysis:

Original execution plan indicated a full table scan:

```
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra       |
|----|-------------|-------|------|---------------|------|---------|------|------|-------------|
|  1 | SIMPLE      | title | ALL  | NULL          | NULL | NULL    | NULL | 6047 | Using where |
```

This was due to the lack of an index on the Title_Name column.

Type: ALL indicates a full table scan, which is inefficient.

Possible Keys: NULL, meaning no indexes are suggested for this query.

Key: NULL, indicating no index is being used.

Rows: 6047, meaning all rows are scanned.

Extra: Using where, which shows a filter is applied after scanning.

Optimization Proposal:

Adding an index on the Title_Name column to improve query performance:
CREATE INDEX idx_title_name ON title (Title_Name);

Implementation and Performance Comparison:

After adding the index, the execution plan showed the use of the index:

```
| id | select_type | table | type  | possible_keys  | key          | key_len | ref  | rows | Extra                    |
|----|-------------|-------|-------|------|------|--------|--------|--------|---------------------------------|
|  1 | SIMPLE      | title | index | null | null | 403    | NULL |  6047  | Using where; Using index   |
```

Execution time before optimization: 0.015s

Execution time after optimization: 0.00577166 s

Query 6: SELECT Title_ID, Title_Name FROM title WHERE Title... - Execution time: 0.00577166 seconds

Query 7: SHOW WARNINGS - Execution time: 0.00010114 seconds

Query 8: SELECT @@lower_case_table_names - Execution time: 0.00023461 seconds

Query 9: SELECT TABLE_NAME FROM information_schema.VIEWS WH... - Execution time: 0.00098324 seconds

Query 10: SELECT *, ... - Execution time: 0.01025259 seconds

Query 11: SELECT COUNT(*) FROM ... - Execution time: 0.00267208 seconds

Query 12: SHOW CREATE TABLE ... - Execution time: 0.00020189 seconds

Query 13: SELECT *, ... - Execution time: 0.00107463 seconds

Query 14: SELECT *, ... - Execution time: 0.00100162 seconds

Query 15: SHOW FULL COLUMNS FROM ... - Execution time: 0.00094084 seconds

Query 16: SHOW INDEXES FROM ... - Execution time: 0.00045352 seconds

Query 17, 18, 19: SHOW SESSION VARIABLES LIKE 'FOREIGN_KEY_CHECKS' - Execution time: 0.00168702, 0.00169033, 0.00166558 seconds respectively

Query 20: SELECT DATABASE() - Execution time: 0.00015345 seconds

Conclusion:

The optimization significantly reduced the execution time by using an index. This confirms that indexing on frequently searched columns can greatly enhance query performance.