Stage 1

The Dataset

I have chosen a set of four different datasets, based on Movies and TV shows, involving four different hosts, being Netflix, Amazon Prime, Hulu and Disney.

All four of these datasets are of the same format, and consist all the Movies and TV series that are being hosted by these hosts up until the year 2021. The datasets illustrate the names, directors, actors, genres, hosted country, show release date, duration, rating and a short description for each of their Movies / TV Shows.

These datasets have all been retrieved from Kaggle, Netflix [1], Amazon Prime [2], Hulu [3] and Disney [4], all four datasets are released under a public domain license. The public domain license states the dataset and the person / people / organisations associated with the gathering / construction of this dataset waive all of his or her rights to the work under copyright law. With this license we are able to distribute and use this dataset for our project.

Due to the same licensing as well as all four datasets being in the same format and consisting of the same entities, I was able to concatenate these four datasets into one. Which gives us the base dataset we will work from, which will be in the structure of *appendix A*.

Appendix A

Snippet of original concatenated dataset

Appendix **A**
Entity Table 1
Content Data-structure Original

show_id	host	type	title	director	cast	country	date_added	release_year	rating	duration	genre	description
1	Amazon	Movie	The Grand Se	Don McKellar	Brendan Glee	Canada	March 30, 202	2014		113 min	Comedy, Dra	A small fishing
2	Amazon	Movie	Take Care Go	Girish Joshi	Mahesh Manj	India	March 30, 202	2018	13+	110 min	Drama, Intern	A Metro Famil
3	Amazon	TV Show	The American		Keri Russell,			2018	NR	6 Seasons	TV Shows	The American
4	Amazon	TV Show	The American		Robert Redfor			2016	TV-14	1 Season	Arthouse, Arts	From Executiv
5	Disney	Movie	Fuzzbucket	Mick Garris	Chris Hebert,	United States	November 12,	1986	TV-G	46 min	Buddy, Comei	An invisible c

Appendix B

Structure and data-types of original dataset

Appendix **B**
Entity Table 1
Content Data-type

Attribute	Data-type	Data-structure	Empty values
show_id	Integer	atomic	No
host	Text	atomic	No
type	Text	atomic	No
title	Text	atomic	No
director	Text	array	Yes
cast	Text	array	Yes
country	Text	atomic	Yes
date_added	Text	atomic	Yes
release_year	Integer	atomic	No
rating	Text	atomic	Yes
duration	Text	atomic	No
genre	Text	array	No
description	Text	atomic	No

Although these datasets have a public domain license, they still pose some complications that we will discuss in our assessment of the dataset, though, even with these complications, this dataset is well suited for this project and the assessment criteria.

The Dataset - Assessment
The quality and level of detail in our concatenated dataset is decent, although there is some missing information, majority of the missing data could be obtained from some further web-scrapping (scrapping this is beyond the scope of this project) but in-fact wont be necessary as the data is still highly useful as is.

Some examples the dataset could be useful for:

- Identifying what content is available from which host in different countries.
- Analysing Actors / Directors, being relationships between the two or which hosts owns the rights to the most shows involving an Actor / Director.
- Analysing show descriptions and genres to find interesting insights into possible similarities in content.
- And many more questions involving Moves / TV shows, as well as these four hosts.

The documentation of the initial datasets are poorly done, and doesn't give us enough information on the data, as where exactly the data came from, as well as any necessary attributions and licensing involved. The data was obtained via web-scraping and API calls, from Wikipedia, Unofficial DB Search and FlixDatabase. The source of the data stops here and there is no elaborations of where these sources obtained their data and their licensing agreements.

I have not researched further into these sources for possible license and attribution hiccups, so the terms of use of the data as it stands will just be for this project and not for publishing or for propriety use.

The Dataset - Interests
The concatenation of these datasets are interesting as they can give us insights on the types and licenses the hosts have for Movies and TV shows in certain countries, this data is very useful from a customers point of view.

Questions I would like to make answerable:

1. Which host played the most Movies / TV shows from a specific **director** in a specific date range and country.?
2. Which host played the most Movies / TV shows from a specific **actor** in a specific date range and country.?
3. What genres are most common from a specific host in a specific date range and country.?
4. What are the latest Movies / TV shows hosted by a specific host in a specific country.?

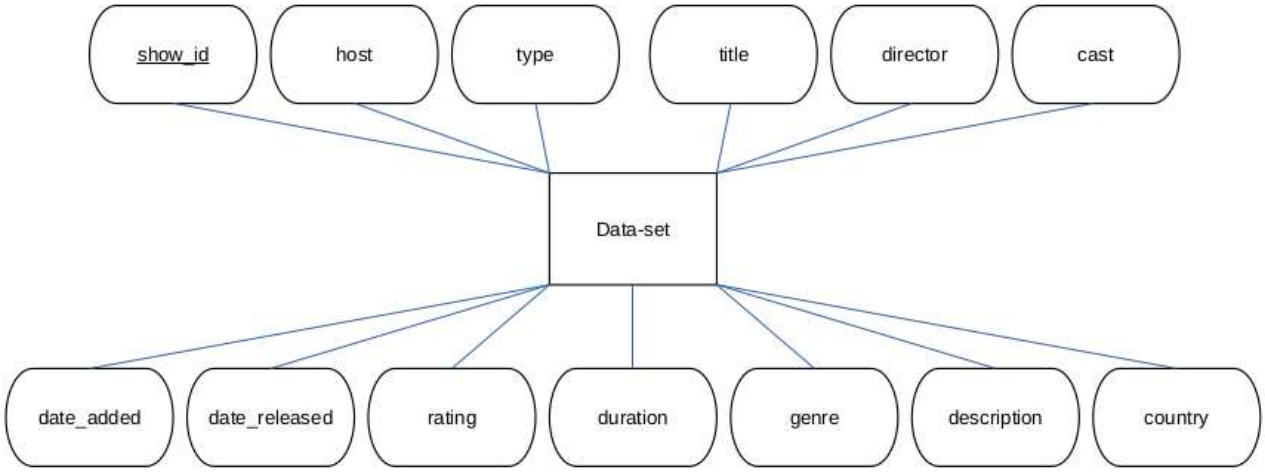
Being able to answer these questions for customers could help customers decide which host they would prefer to subscribe to.

Stage 2

We will be using most entities from the original dataset, which is currently modelled as illustrated in *appendix C*.

Appendix C

ER diagram of original dataset



We will exclude the datasets ‘show_id’ which holds no useful value, we will also be excluding ‘duration’ as this would have no place for the type of analysis’s we targeting for.

Currently the dataset is not in a state we would be able to use, as for example the entity ‘cast’ , is a complex data structure, and we would need scalar values to be able to use these in our analysis.

We will first develop a normalised structure of the dataset.

ER modelling

Currently our data is in no *Normal Form*, and all data is presented in a single entity, indicated in *appendix C*.

Normalising our data to NF1

By removing complex data structures in the attributes ‘cast’, ‘directors’ and ‘genre’. We will also normalise the data-types here for the attributes ‘date_added’, release_year’, ‘rating’ and ‘duration’. We will then create enumerator types for the attributes ‘type’, ‘host’, ‘rating’ and ‘genre’. Lastly we will fill all empty fields or not understood fields to NULL date-type.

After applying these changes we are left with a dataset in NF1, illustrated in *appendix D*.

Appendix D

Data modelled to normal form 1

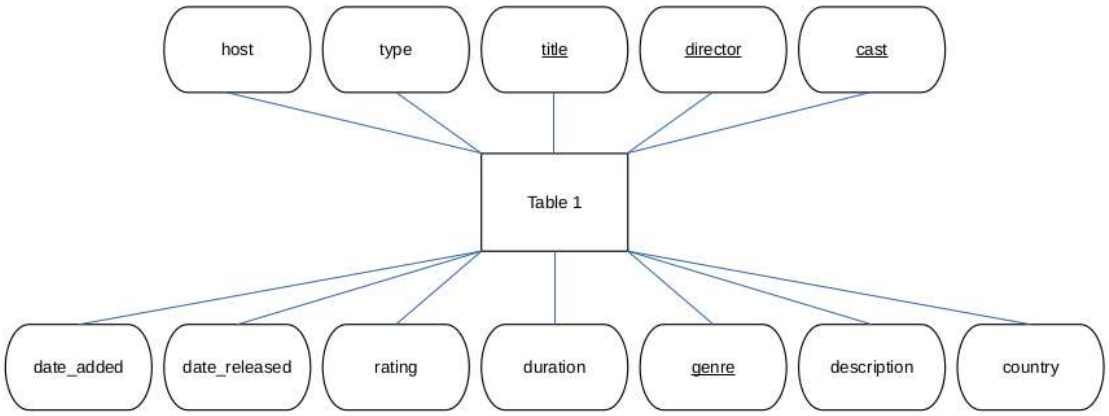
Appendix	**D**
Entity	Table 1
Content	Data-structure NF1
CK's	CK_1 = (title, director, cast, genre)

host	type	title	director	cast	country	date_added	release_year	rating	genre	description
Amazon	Movie	The Grand Se	Don McKellar	Brendan Glee	Canada	2021-03-30	2014	NULL	Comedy	A small fishing
Amazon	Movie	The Grand Se	Don McKellar	Brendan Glee	Canada	2021-03-30	2014	NULL	Drama	A small fishing
Amazon	Movie	The Grand Se	Don McKellar	Taylor Kitsch	Canada	2021-03-30	2014	NULL	Comedy	A small fishing
Amazon	Movie	The Grand Se	Don McKellar	Taylor Kitsch	Canada	2021-03-30	2014	NULL	Drama	A small fishing
Amazon	Movie	The Grand Se	Don McKellar	Gordon Pinse	Canada	2021-03-30	2014	NULL	Comedy	A small fishing

Represented by an ER model, illustrated in *appendix D1*.

Appendix D1

ER model of dataset in normal form 1



Normalising to NF2

If we identify possible candidate keys for the dataset we get the following represented in *appendix D (CK's)*. From here we see we only have one being CK_1 = (title, director, cast, genre).

With the candidate key CK_1, we notice none of the non-primary keys relate to the entire candidate key, and are only relating to part of the candidate key. So lets split all those candidate attributes to form another table, illustrated in *appendix E table 3*. The left over attributes are still not in NF2, we will split this table again to give use two more tables, illustrated in *appendix E table 1 & 2*.

After applying the splits we are left with a model in NF2, illustrated in *appendix E*.

Appendix E

Data modelled to normal form 2

Appendix E

Entity Table 3
Content Data-structure NF2
CK's CK_1 = (title, director, cast, genre)

title	director	cast	genre
The Grand Se	Don McKellar	Brendan Glee	Comedy
The Grand Se	Don McKellar	Brendan Glee	Drama
The Grand Se	Don McKellar	Taylor Kitsch	Comedy
The Grand Se	Don McKellar	Taylor Kitsch	Drama
The Grand Se	Don McKellar	Gordon Pinse	Comedy

Entity Table 2
Content Data-structure NF2
CK's CK_1 = (title, host, country)

title	host	country	date_added
The Grand Se	Amazon	Canada	2021-03-30
Take Care Go	Amazon	India	2021-03-31
The American	Amazon	All	NULL
The American	Amazon	All	NULL
Fuzzbucket	Disney	United States	2021-04-03

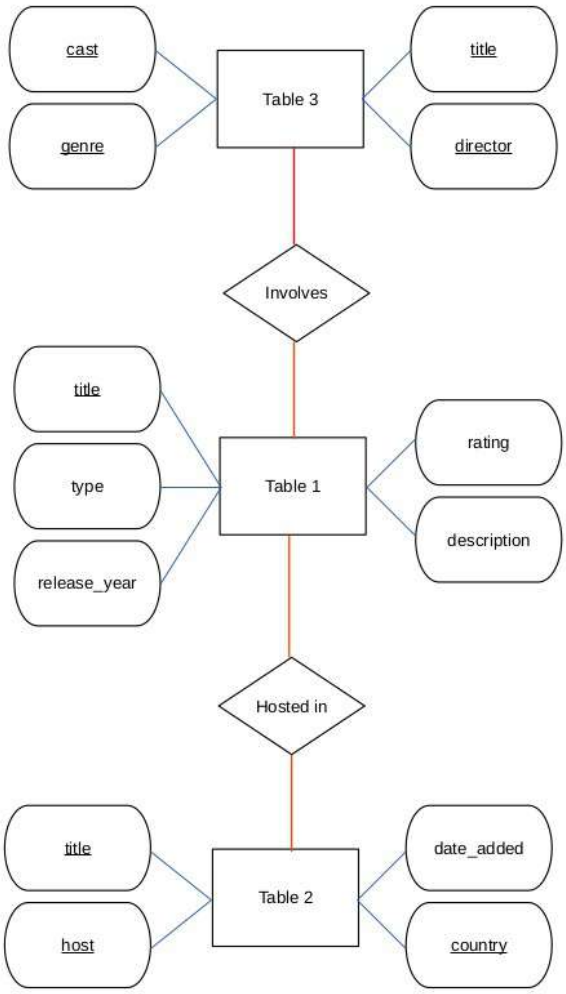
Entity Table 1
Content Data-structure NF2
CK's CK_1 = (title)

title	type	release_year	rating	description
The Grand Se	Movie	2014	NULL	A small fishing
Take Care Go	Movie	2018	13+	A Metro Famil
The American	TV Show	2018	NR	The American
The American	TV Show	2016	TV-14	From Executiv
Fuzzbucket	Movie	1986	TV-G	An invisible c

Represented by an ER model, illustrated in *appendix E1*.

Appendix E1

ER model of dataset in normal form 2



Normalising to NF3 / Boyce-Codd

If we try identify any transitive dependencies, I don't see any, so we currently already in NF3 as well as Boyce-Codd NF.

Normalising to NF4

If we identify any multi-valued dependencies that aren't a candidate key. We notice that from table 3 in *appendix E*, the entities 'director', 'cast' and 'genre' could be derived from sets and are not directly related. So we could split these into individual entities.

After applying these changes we are left with a dataset model in NF4, as illustrated in *appendix F*.

Appendix F

Data modelled to normal form 4

Entity	Table 1
Content	Data-structure NF4
CK's	CK_1 = (title)

title	type	release_year	rating	description
The Grand Se	Movie	2014	NULL	A small fishing
Take Care Go	Movie	2018	13+	A Metro Famil
The American	TV Show	2018	NR	The American
The American	TV Show	2016	TV-14	From Executiv
Fuzzbucket	Movie	1986	TV-G	An invisible c

Appendix	**F**
Entity	Table 2
Content	Data-structure NF4
CK's	CK_1 = (title, host, country)

title	host	country	date_added
The Grand Se	Amazon	Canada	2021-03-30
Take Care Go	Amazon	India	2021-03-31
The American	Amazon	All	NULL
The American	Amazon	All	NULL
Fuzzbucket	Disney	United States	2021-04-03

Entity	Table 5
Content	Data-structure NF4
CK's	CK_1 = (title, director)

title	director
The Grand Se	Don McKellar
Take Care Go	Girish Joshi
Fuzzbucket	Mick Garris
Take Care	Liz Tuccillo
Zoombies	Glenn Miller

Entity	Table 3
Content	Data-structure NF4
CK's	CK_1 = (title, cast)

title	cast
The Grand Se	Brendan Glee
The Grand Se	Taylor Kitsch
The Grand Se	Gordon Pinse
Take Care Go	Mahesh Manj
Take Care Go	Abhay Mahaja

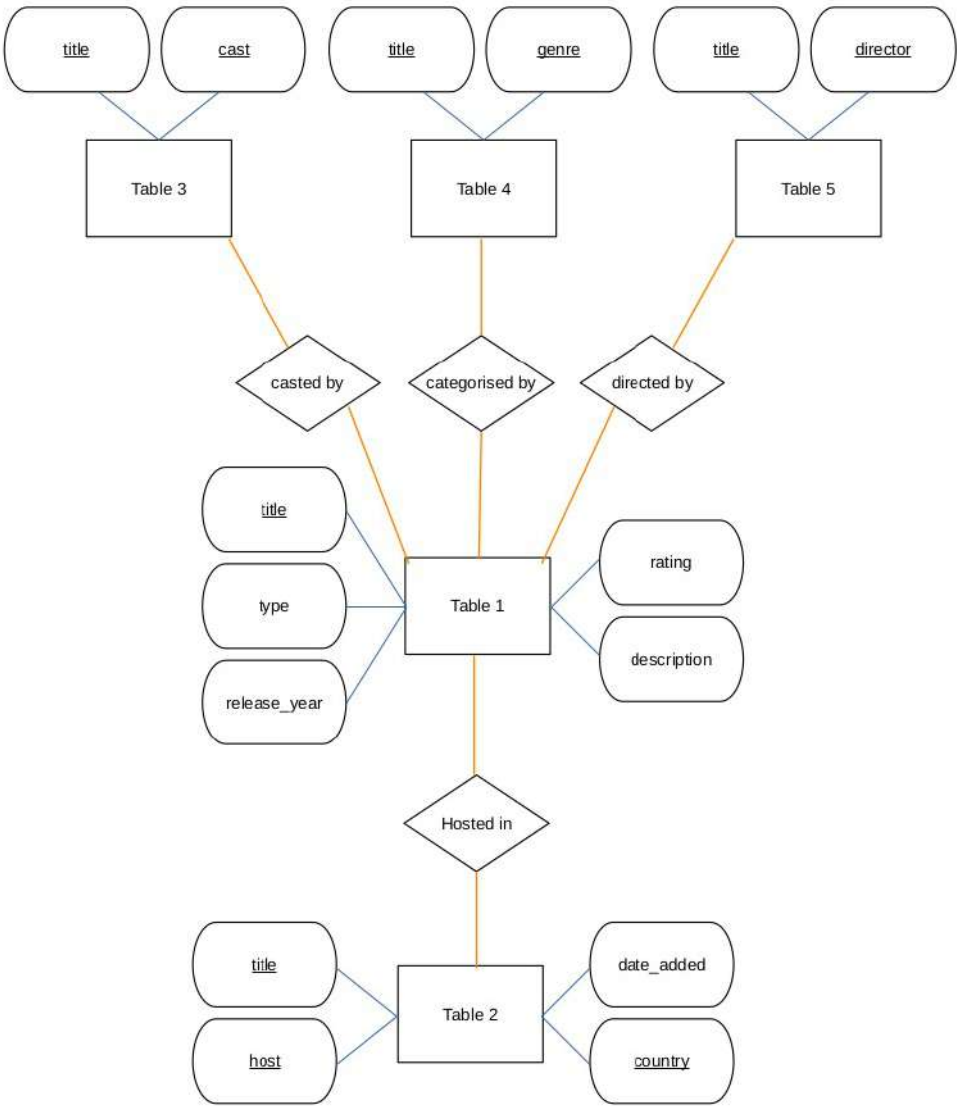
Entity	Table 4
Content	Data-structure NF4
CK's	CK_1 = (title, genre)

title	genre
The Grand Se	Comedy
The Grand Se	Drama
Take Care Go	Comedy
Take Care Go	Drama
The American	Comedy

Represented by an ER model, illustrated in *appendix F1*.

Appendix F1

ER model of dataset in normal form 4



Normalising to NF5

If we try identify a situation where a deleted record would result in losing other data. We don't seem to have this case so we are currently already in NF5.

Referencing possibly 'genre', 'country', 'rating' and 'type'. This is data we would like to keep even after all records are deleted, though these would be enumerators stored as a variable, not in the database as their values would be finite. So we don't need to create separate entities for these mappings.

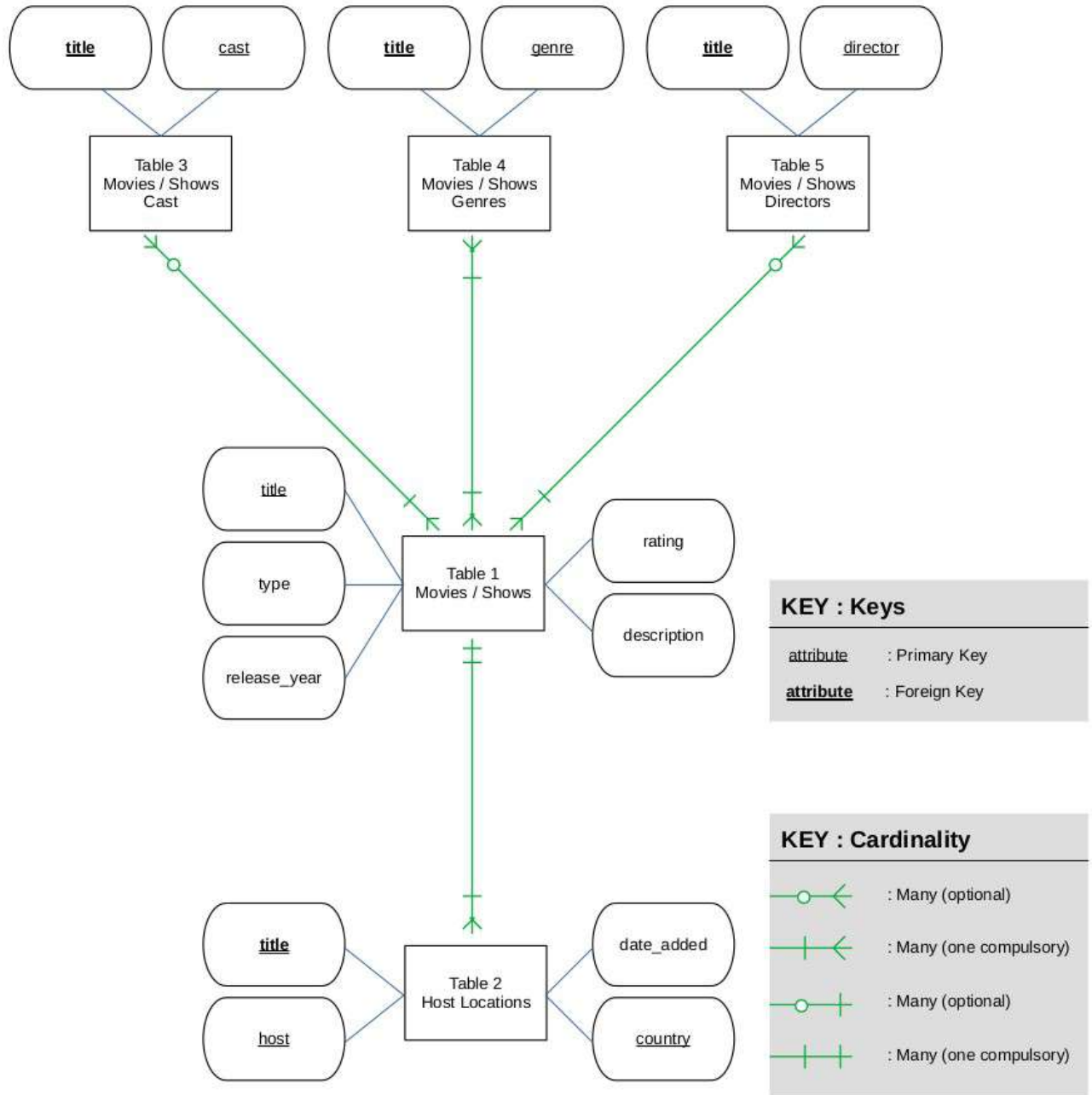
I have opted to normalise the data-structure as far as NF5, to allow the ability of seamlessly query for other questions without having a difference in performance of the database. The records are also of a low amount, being under 30 000, which stepping up the normalisation ladder wont have a noticeable effect on performance.

ER Model with cardinality

Modelling cardinality from our normalised model, as well as identifying what our Primary and Foreign keys would be, illustrated in *appendix F1*. This model would be straight forward to implement into an SQL Database, as illustrated in *appendix G*.

Appendix G

ER model showing cardinality and PK's / FK's



Stage 3

In this stage we will be creating our SQL database, and illustrate how we populate as well as query our database for answering our predefined questions.

Building the MySQL database

Creating all relevant tables:

Creating the database

```
CREATE DATABASE app_database;
```

Creating the movie_shows table

```
CREATE TABLE movie_shows (
    title VARCHAR(255) NOT NULL,
    type ENUM('movie', 'tv show') NOT NULL,
    release_year YEAR NOT NULL,
    rating ENUM('TV-MA', 'PG-13', 'R', 'TV-14', 'PG', 'TV-PG', 'NOT RATED', 'G', 'TV-G', 'TY-Y', 'TV-Y7', 'NR',
    'NC-17', 'TY-Y7-FV', 'UR', '13+', 'ALL', '18+', '16+', '7+', 'TV-NR', 'AGES_16_', 'AGES_18_', 'ALL_AGES'),
    description TEXT,
    PRIMARY KEY (title)
);
```

Creating the movie_host_locations table

```
CREATE TABLE host_locations (
    title VARCHAR(255) NOT NULL,
    host ENUM('netflix', 'amazon', 'hulu', 'disney') NOT NULL,
    country ENUM('all', 'united states', 'japan', 'united kingdom', 'mexico', 'ireland', 'germany', 'sweden',
    'canada', 'france', 'new zealand', 'italy', 'czech republic', 'spain', 'australia', 'denmark', 'israel',
    'russia', 'south africa', 'belgium', 'luxembourg', 'china', 'poland', 'ukraine', 'malta', 'chile', 'netherlands',
    'united arab emirates', 'norway', 'philippines', 'hungary', 'bulgaria', 'south korea', 'thailand', 'switzerland',
    'tunisia', 'romania', 'afghanistan', 'serbia', 'montenegro', 'costa rica', 'hong kong', 'colombia',
    'french polynesia', 'saudi arabia', 'austria', 'argentina', 'uruguay', 'qatar', 'syria', 'venezuela', 'finland',
    'india', 'singapore', 'greece', 'iceland', 'brazil', 'taiwan', 'ghana', 'burkina faso', 'ethiopia', 'turkey',
    'nigeria', '', 'nepal', 'jordan', 'algeria', 'indonesia', 'egypt', 'kuwait', 'malaysia', 'vietnam', 'lebanon',
    'mauritius', 'cameroon', 'palestine', 'kenya', 'cambodia', 'bangladesh', 'portugal', 'cayman islands', 'senegal',
    'namibia', 'angola', 'peru', 'mozambique', 'belarus', 'zimbabwe', 'puerto rico', 'pakistan', 'cyprus', 'guatemala',
    'iraq', 'malawi', 'paraguay', 'croatia', 'iran', 'west germany', 'albania', 'georgia', 'soviet union', 'morocco',
    'slovakia', 'bermuda', 'ecuador', 'armenia', 'mongolia', 'bahamas', 'sri lanka', 'latvia', 'liechtenstein', 'cuba',
    'nicaragua', 'slovenia', 'dominican republic', 'samoa', 'azerbaijan', 'botswana', 'vatican city', 'jamaica',
    'kazakhstan', 'lithuania', 'somalia', 'sudan', 'panama', 'uganda', 'east germany', 'monaco', 'kosovo',
    'tanzania') NOT NULL,
    date_added DATE,
    PRIMARY KEY (title, host, country),
    FOREIGN KEY (title)
        REFERENCES movie_shows(title)
        ON DELETE CASCADE
);
```

Creating the movie_show_cast table

```
CREATE TABLE movie_show_cast (
    title VARCHAR(255) NOT NULL,
    cast VARCHAR(255) NOT NULL,
    PRIMARY KEY (title, cast),
    FOREIGN KEY (title)
        REFERENCES movie_shows(title)
        ON DELETE CASCADE
);
```

Creating the movie_show_genre table

```
CREATE TABLE movie_show_genre (
    title VARCHAR(255) NOT NULL,
    genre ENUM('general', 'comedy', 'stand up', 'crime', 'drama', 'thriller', 'action', 'horror', 'science fiction',
    'family', 'kids', 'cooking & food', 'documentaries', 'lifestyle & culture', 'music', 'reality', 'romance', 'news',
    'mystery', 'sports', 'history', 'teen', 'health & wellness', 'black stories', 'latino', 'adventure', 'anime',
    'late night', 'sketch comedy', 'classics', 'lgbtq+', 'adult animation', 'sitcom', 'international', 'game shows',
    'cartoons', 'science & technology', 'international tv shows', 'tv dramas', 'tv mysteries', 'crime tv shows',
    'tv action & adventure', 'docuseries', 'reality tv', 'romantic tv shows', 'tv comedies', 'tv horror',
    'children & family movies', 'dramas', 'independent movies', 'international movies', 'british tv shows', 'comedies',
    'spanish-language tv shows', 'thrillers', 'romantic movies', 'music & musicals', 'horror movies',
    'sci-fi & fantasy', 'tv thrillers', 'kids tv', 'action & adventure', 'tv sci-fi & fantasy', 'classic movies',
    'anime features', 'sports movies', 'anime series', 'korean tv shows', 'science & nature tv', 'teen tv shows',
    'cult movies', 'tv shows', 'faith & spirituality', 'lgbtq movies', 'stand-up comedy', 'movies',
    'stand-up comedy & talk shows', 'classic & cult tv', 'suspense', 'documentary', 'fantasy', 'special interest',
    'talk show and variety', 'arts', 'entertainment', 'and culture', 'animation', 'music videos and concerts',
    'fitness', 'faith and spirituality', 'military and war', 'western', 'lgbtq', 'unscripted', 'young adult audience',
    'arthouse', 'historical', 'musical', 'biographical', 'action-adventure', 'superhero', 'survival', 'animals & nature',
    'coming of age', 'lifestyle', 'concert film', 'anthology', 'medical', 'variety', 'spy/espionage', 'buddy', 'parody',
    'game show / competition', 'romantic comedy', 'police/cop', 'talk show', 'dance', 'series', 'soap opera / melodrama',
    'disaster', 'travel') NOT NULL,
    PRIMARY KEY (title, genre),
    FOREIGN KEY (title)
        REFERENCES movie_shows(title)
        ON DELETE CASCADE
);
```

Creating the movie_show_director table

```
CREATE TABLE movie_show_director (
    title VARCHAR(255) NOT NULL,
    director VARCHAR(255) NOT NULL,
    PRIMARY KEY (title, director),
    FOREIGN KEY (title)
        REFERENCES movie_shows(title)
        ON DELETE CASCADE
);
```

Creating privileges:

Creating the user and privileges – public user

```
CREATE USER client IDENTIFIED BY 'password';

GRANT SELECT ON movie_shows TO client;
GRANT SELECT ON host_locations TO client;
GRANT SELECT ON movie_show_cast TO client;
GRANT SELECT ON movie_show_genre TO client;
GRANT SELECT ON movie_show_director TO client;
```

Creating the user and privileges – admin user

```
CREATE USER app_admin IDENTIFIED BY 'xxxpass';

-- Granting 'app_admin' permissions
GRANT SELECT, INSERT, UPDATE, DELETE ON movie_shows TO app_admin WITH GRANT OPTION;
GRANT SELECT, INSERT, UPDATE, DELETE ON host_locations TO app_admin WITH GRANT OPTION;
GRANT SELECT, INSERT, UPDATE, DELETE ON movie_show_cast TO app_admin WITH GRANT OPTION;
GRANT SELECT, INSERT, UPDATE, DELETE ON movie_show_genre TO app_admin WITH GRANT OPTION;
GRANT SELECT, INSERT, UPDATE, DELETE ON movie_show_director TO app_admin WITH GRANT OPTION;
```

Here we have just created two accounts, ‘client’ being the public account anyone could access, and ‘admin’ would be the account used for populating and amending the database, though still not having all privileges.

I was able to get users to work in MySQL shell, though not through express and node. So this is why I have used ‘root’ within my application.

Optimising database:

If this was up-to me I would denormalise the data-structure to balance at a point of being able to answer our questions and only those questions at the fastest time possible. Though I have opted to leave the data-structure normalised to allow for future questions to be able to be answered in a decent amount of time.

The optimising we will do will be around indexing tables based on already known queries and possible future queries.

Here I decided to index ‘type’ from table ‘movies_shows’, reason for this is clients would likely want to do isolate their comparisons to the type of content provided from the hosts, and having an index would make this quick.

Creating the index on type

```
CREATE INDEX movie_shows_type_indcx ON movie_shows(type);
```

Populating the Database

Firstly I needed to clean the dataset, by normalising the data-types and enumerator classes. I did this with a python script which is available in the source folder (file name: csv_cleanup.ipynb). This script just cleans the data-types and concatenates the datasets, then creates a clean csv to use for the population of the database.

Below I illustrate the method in which I populated the database. As the dataset wasn’t normalised, I decided to normalise while populating the database with a script in node. Due to running out of time for this project, I decided not to look into implementing transactions with these queries. The full

script I used in populating the database is available in the source folder (file name: populate_database.js). Below is how the queries were made.

Script used to create the population queries – reference source code for entire script

```
for (let i = 0; i < dataset.length; i++) {
    host_locations = []
    movie_shows_cast = []
    movie_shows_genres = []
    movie_shows_directors = []

    // Creating the Movie / Show record
    movie_shows_feilds = [dataset[i].title, dataset[i].type, dataset[i].release_year, dataset[i].rating, dataset[i].description]
    movie_shows_query = 'INSERT INTO movie_shows (title,type,release_year,rating,description) VALUES (?,?,?,?,?);'
    // Creating the Host locations records
    for (let j = 0; j < dataset[i].countries.length; j++) {
        host_locations_feilds = [dataset[i].title, dataset[i].host, dataset[i].date_added, dataset[i].countries[j]]
        host_locations_query = 'INSERT INTO host_locations (title,host,date_added,country) VALUES (?,?,?,?);'
        host_locations.push([host_locations_feilds, host_locations_query])
    }

    // Creating the Cast records
    for (let j = 0; j < dataset[i].cast.length; j++) {
        movie_shows_cast_feilds = [dataset[i].title, dataset[i].cast[j]]
        movie_shows_cast_query = 'INSERT INTO movie_show_cast (title,cast) VALUES (?,?);'
        movie_shows_cast.push([movie_shows_cast_feilds, movie_shows_cast_query])
    }

    // Creating the Genres records
    for (let j = 0; j < dataset[i].genres.length; j++) {
        movie_shows_genres_feilds = [dataset[i].title, dataset[i].genres[j]]
        movie_shows_genres_query = 'INSERT INTO movie_show_genre (title,genre) VALUES (?,?);'
        movie_shows_genres.push([movie_shows_genres_feilds, movie_shows_genres_query])
    }

    // Creating the Directors records
    for (let j = 0; j < dataset[i].directors.length; j++) {
        movie_shows_directors_feilds = [dataset[i].title, dataset[i].directors[j]]
        movie_shows_directors_query = 'INSERT INTO movie_show_director (title,director) VALUES (?,?);'
        movie_shows_directors.push([movie_shows_directors_feilds, movie_shows_directors_query])
    }

    // RUN ALL THE CREATED QUERIES
}
```

Database review

The database we have got is good, though the missing data fields affect our accuracy. So some web-scrapping for these missing fields would be ideal (out of the scope of this project).

Database queries

Here we will state what queries we would use for our node application when gathering answers to our predefined questions.

Question 1:

Which host played the most Movies / TV shows from a specific **director** in a specific date range and country.?

Query for question 1

```
SELECT host_locations.host
FROM host_locations
JOIN movie_shows ON host_locations.title = movie_shows.title
JOIN movie_show_director ON host_locations.title = movie_show_director.title
WHERE host_locations.country = ?
AND host_locations.date_added > ?
AND host_locations.date_added < ?
AND movie_show_director.director LIKE ?
GROUP BY host
ORDER BY COUNT(*) DESC
LIMIT 1;
```

Question 2:

Which host played the most Movies / TV shows from a specific **actor** in a specific date range and country.?

Query for question 2

```
SELECT host_locations.host
FROM host_locations
JOIN movie_show_cast ON host_locations.title = movie_show_cast.title
WHERE host_locations.country = ?
AND host_locations.date_added > ?
AND host_locations.date_added < ?
AND movie_show_cast.cast LIKE ?
GROUP BY host
ORDER BY COUNT(*) DESC
LIMIT 1;
```

Question 3:

What genres are most common from a specific host in a specific date range and country.?

Query for question 3

```
SELECT movie_show_genre.genre
FROM movie_show_genre
JOIN host_locations ON movie_show_genre.title = host_locations.title
WHERE host_locations.country = ?
AND host_locations.host = ?
AND host_locations.date_added > ?
AND host_locations.date_added < ?
GROUP BY genre
ORDER BY COUNT(*) DESC
LIMIT 5;
```

Question 4:

What are the latest Movies / TV shows hosted by a specific host in a specific country.?

Query for question 4

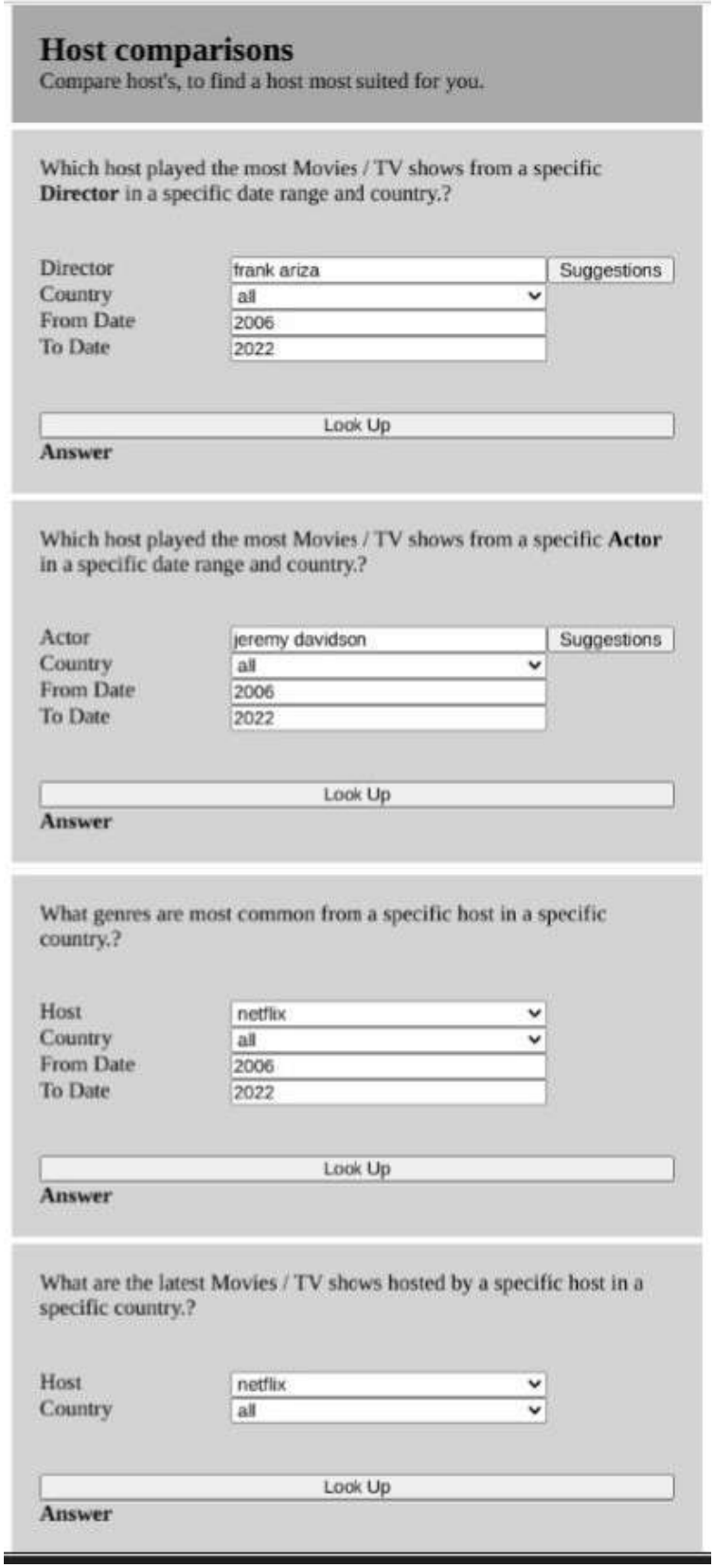
```
SELECT movie_shows.title, movie_shows.type, movie_shows.release_year, movie_shows.rating, host_locations.country,
GROUP_CONCAT(movie_show_genre.genre) genres
FROM movie_shows
JOIN host_locations ON movie_shows.title = host_locations.title
JOIN movie_show_genre ON movie_shows.title = movie_show_genre.title
WHERE host_locations.country = ?
AND host_locations.host = ?
GROUP BY title
ORDER BY movie_shows.release_year DESC
LIMIT 5;
```

Stage 4

The web application

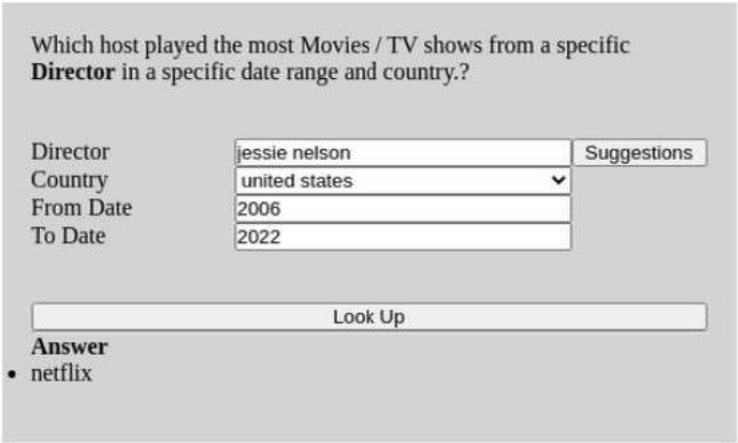
Appendix G.1

Overview of application



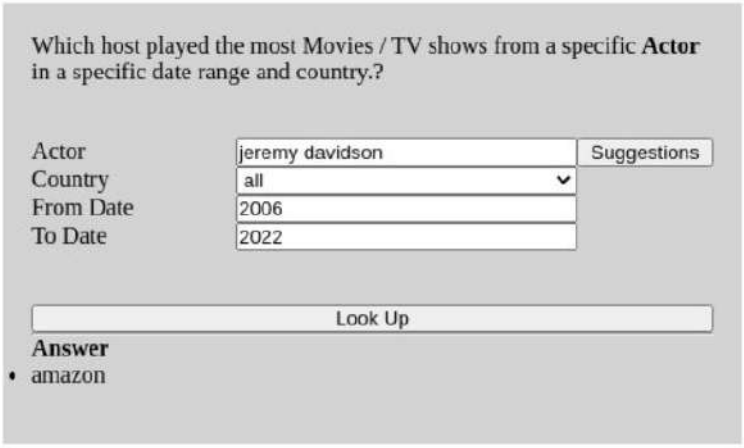
Appendix G.2

Question 1 : Input and Output



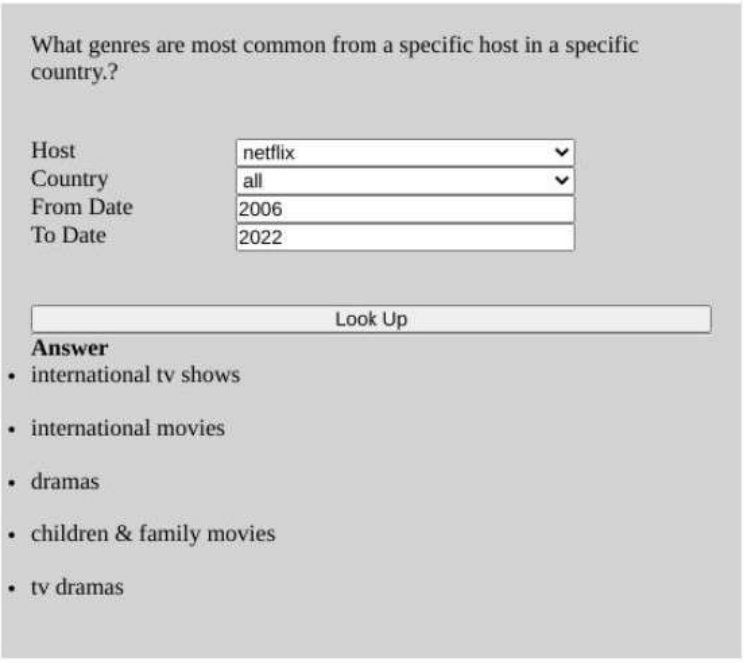
Appendix G.3

Question 2 : Input and Output



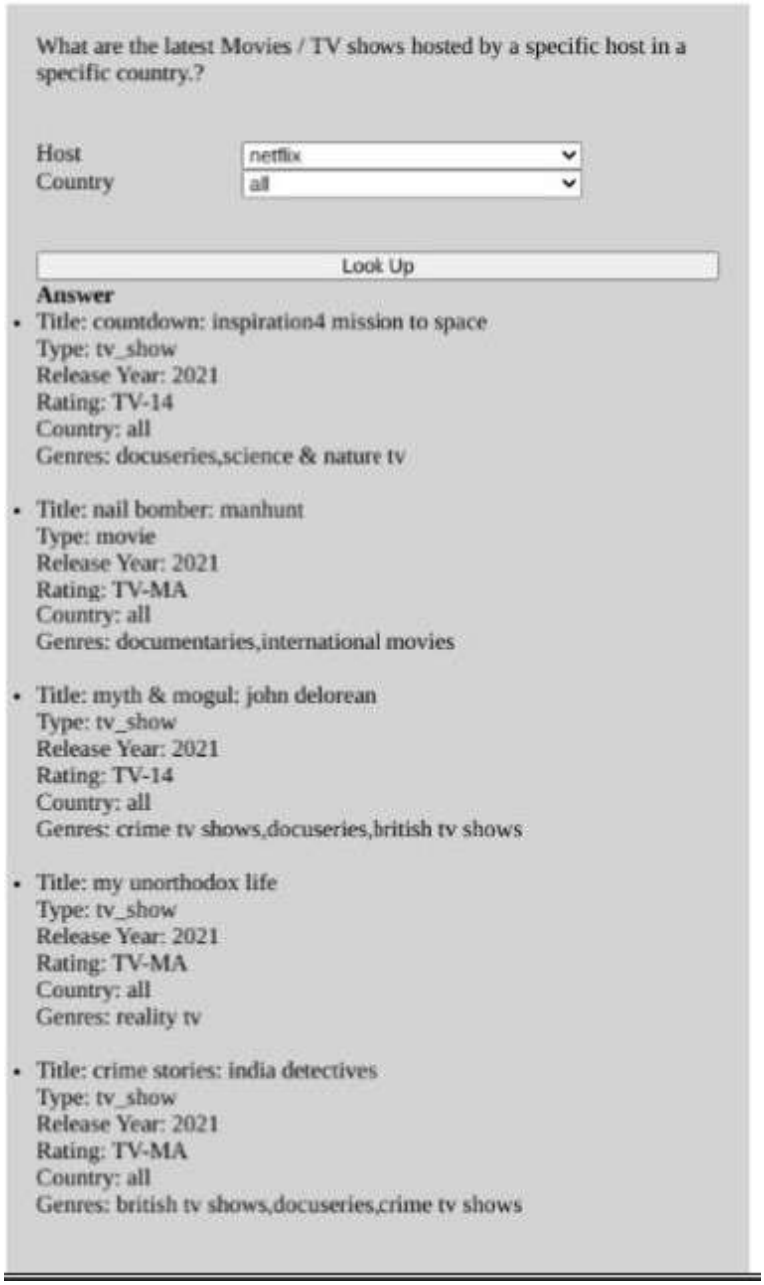
Appendix G.4

Question 3 : Input and Output



Appendix G.5

Question 4 : Input and Output



References:

1. Kaggle, Shivam Bansal, 2021
   https://www.kaggle.com/datasets/shivamb/netflix-shows
2. Kaggle, Shivam Bansal, 2021
   https://www.kaggle.com/shivamb/hulu-movies-and-tv-shows
3. Kaggle, Shivam Bansal, 2021
   https://www.kaggle.com/datasets/shivamb/amazon-prime-movies-and-tv-shows
4. Kaggle, Shivam Bansal, 2021
   https://www.kaggle.com/datasets/shivamb/disney-movies-and-tv-shows