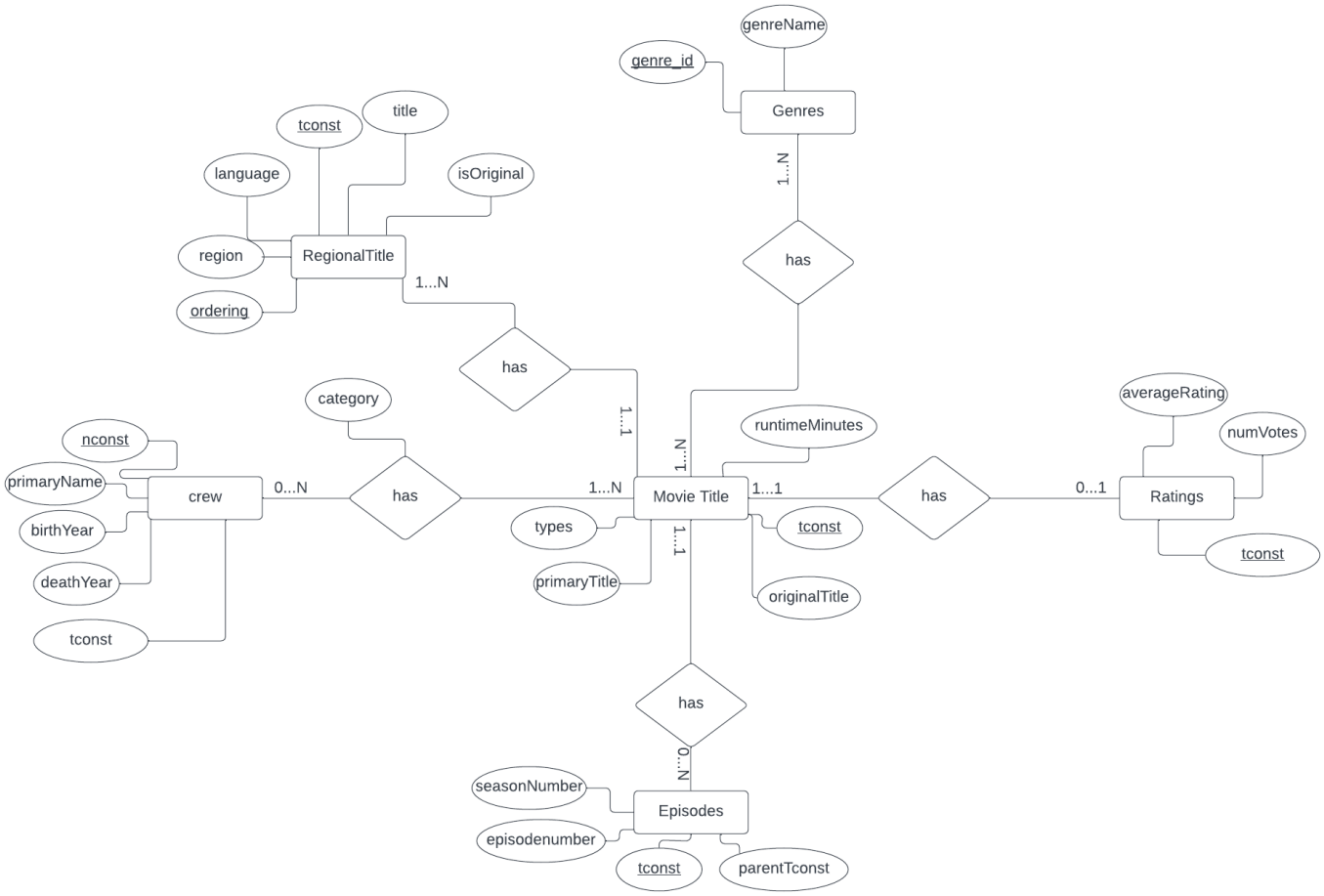
Question 1:


genreName
genre_id
Genres
1...N
has
tconst
title
language
isOriginal
region
RegionalTitle
ordering
1...N
has
1...1
category
nconst
primaryName
crew
0...N
has
1...N
Movie Title
runtimeMinutes
N...1
1...1
types
primaryTitle
tconst
originalTitle
birthYear
deathYear
tconst
has
0...N
seasonNumber
episodenumber
Episodes
tconst
parentTconst
1...1
has
0...1
averageRating
numVotes
Ratings
tconst

Question 2:

Movie_title(tconst, language, types, region , originalTitle, primaryTitle)

Crew(nconst, directors, writers, birthYear, deathYear, primaryName, tconst)

Genre(genreId, genreName)

Episodes(tconst, episodeNumber, seasonNumber, parentTconst)

Ratings(averageRating, numVotes, tconst)

Regional Title(ordering, tconst, language, region, title, isOriginal)

movie_genre(tconst, genre_id)

movie_crew(tconst,nconst, ordering, category)

Movie Title

```
create table hw_schema.movie_title
(
    tconst           integer not null
        constraint movie_title_pk
            primary key,
    types            varchar,
    "originalTitle"  varchar,
    "primaryTitle"   varchar,
    "runtimeMinutes" integer
)
```

Crew

```
create table hw_schema.crew
(
    nconst        integer not null
        constraint crew_pk
            primary key,
    "birthYear"   integer,
    "deathYear"   integer,
    "primaryName" varchar
);
```

Genre Table

```
create table hw_schema.genre
(
    "genreId"   integer not null
        constraint genre_pk
            primary key,
    "genreName" varchar
)
```

Episodes

```
create table hw_schema.episodes
(
    tconst          integer
        constraint episodes_pk
            primary key,
    "episodeNumber" integer,
    "seasonNumber"  integer,
    "parentTconst"  integer
        constraint episodes_fk_movie_title
            references hw_schema.movie_title
);
```

Ratings

```
create table hw_schema.ratings
(
    "averageRating" integer,
    "numVotes"      integer,
    tconst          integer not null
        constraint ratings_pk
            primary key
        constraint ratings_fk_movie_title
            references movie_title
);
```

Regional Title

```
create table hw_schema.regional_title
(
    ordering          integer not null,
    tconst            integer not null
        constraint regional_title_fk_movie_title
            references movie_title,
    language          varchar,
    region            varchar,
    title             varchar,
    "isOriginalTitle" integer,
    constraint regional_title_pk
        primary key (ordering, tconst)
);
```

Crew_movie_title_mapping

```
create table hw_schema.crew_movie_mapping
(
    tconst   integer not null
        constraint crew_movie_mapping_movie_title_tconst_fk
            references movie_title,
    ordering integer,
    nconst   integer not null
        constraint crew_movie_mapping_crew_nconst_fk
            references crew,
    category varchar not null,
    constraint crew_movie_mapping_pk
        primary key (nconst, tconst, category)
);
```

Genre_movie_mapping

```
create table hw_schema.genre_movie_mapping
(
    genre_id integer not null
        constraint genre_movie_mapping_fk_genre_id
            references genre,
    tconst   integer not null
        constraint genre_movie_mapping_fk_movie_title
            references movie_title,
    constraint genre_movie_mapping_pk
        primary key (genre_id, tconst)
);
```

Using integers instead of strings for primary key introduces the uniqueness and causes no name collision. Suppose if we want to have a name as the primary key then two users can have the same name. Primary key is an identifier and should be unique and should serve its purpose to categorize the data. Using integers introduces better indexing of data. Using strings as the primary key in a dataset of more than 1 million records slows the performance. Used integer with no size attached to it. Let the int, varchar be the max size of each of them so that no data gets excluded from the table while adding.

Question 3:

name.basics - This data file gives us details about the crew including its nconst which is a primary key with unique values, primary name is the name of the crew member, birthYear is the year in which the crew member was born, deathYear indicates the year in which they passed away, primaryProfession gives us information about the profession of each crew member and they can have multiple professions and knownForTitles describes the titles for which the respective crew member has worked for. The whole purpose of this file is to give us an idea about each crew member and the title which they are associated with.

title.akas - the whole file gives us a description of the movie title which is also known as. Where the file describes a titleId which is unique for each movie and the name of the title under title, the region/regions to which the title belonged, the language associated with the title, whether the title is a original title or if it was derived of from another title maybe in another language, the type to which it belonged to.

title.basics- the title.basics file describes each movie title with a unique id associated with each title and indicated under tconst, the titleType indicates what kind of a movie was it, whether it was short or a full length movie or a video and so on. And what was its primary title and the original title whether both were the same or if the primary was different from the original title which was decided by someone. Furthermore, if the movie classifies as a adult or not and when did the movie start under the startYear and when did the movie end described under endYear. The runtimeMinutes, how long did

the respective titleType run for and the genres associated with whether short, documentary, animation etc.

title.crew- includes the tconst which is the unique id for the title and the directors and writes along with their unique ids which denote the director and writer associated with the particular title and with this we can know the director/writer has been associated with many titles or not.

title.episode- this file denotes the information about episodes and has a unique key associated with each episode under tconst and the parrenTconst which contains the unique title id and indicates which episode has been linked to which title and the seasonNumber denoting the season to which the episode/title belongs and the episodeNumber associated with it.

title.principals- is a file which contains the mapping of the crew unique id nconst to the title unique id which is tconst. Category indicates to which the title belonged to like self, cinematogapher, director, producer and so on.

title.ratings- this file gives an overview of the rating for each title and tconst gives us the unique id associated with the movie_title and average tells us whats the average rating for the movie_title, numVotes denotes the votes that respective movie title has received.

Question 4:

- Created individual csv(column separated values) files from the tsv(tab separated values).
- First reading the respective files using pd.read_table to read the delimited file and then creating data frames for the files and storing them in a variable.
- Removing tt from tconst and nm from nconst.
- Removed unwanted outliers
- Dropped columns which were not relevant and not adding them to the csv file.
- Mapping data where the columns are dependent on other columns of a file or the same file.
- Pre-processed the data by removing all '\N' with NaN values.
- Created individual csv files for the entities in the ER diagram after pre-processing the files.
- For a few files read only the columns which had to be cleaned.
- For each table certain attributes of the respective tsv files were cleaned before creating the csv file.
- Ignored index before creating the csv files, so that the index of the files were not added.
- Split the strings wherever necessary.
- Where multiple entries for one row under one column used explode to convert each element into a row.
- Used read_table, explode, apply, drop, replace, merge of the pandas library to help with the pre-processing of data.
- Imported the data to postgres tables using the data grip import option.
- Right clicking on the table gives an option to import the data file to the table and click on import file(we can import the data file) and check if each attribute is mapped correctly to load the data.

The time it took to import the data files to the tables was: 3hr 26 minutes

Question 5:

Forcing an error on foreign key constraint using transaction. Because, the foreign key referencing the primary in the movie_title does not exist.

```
BEGIN;

INSERT INTO hw_schema.episodes(tconst, "parentTconst", "seasonNumber", "episodeNumber")
VALUES (1,2,4,5);
INSERT INTO hw_schema.episodes(tconst, "parentTconst", "seasonNumber", "episodeNumber")
VALUES (444444,555555,4.0,5.0);
INSERT INTO hw_schema.episodes(tconst, "parentTconst", "seasonNumber", "episodeNumber")
VALUES (6.6,7.7,6,7);

COMMIT;
```

```
big_data_hw1.public> BEGIN
[2023-02-10 17:08:57] [25P02] ERROR: current transaction is aborted, commands ignored until end of transaction block
[2023-02-10 17:08:57] [23503] ERROR: insert or update on table "episodes" violates foreign key constraint "episodes_fk_m
[2023-02-10 17:08:57] Detail: Key (parentTconst)=(555555) is not present in table "movie_title".
```