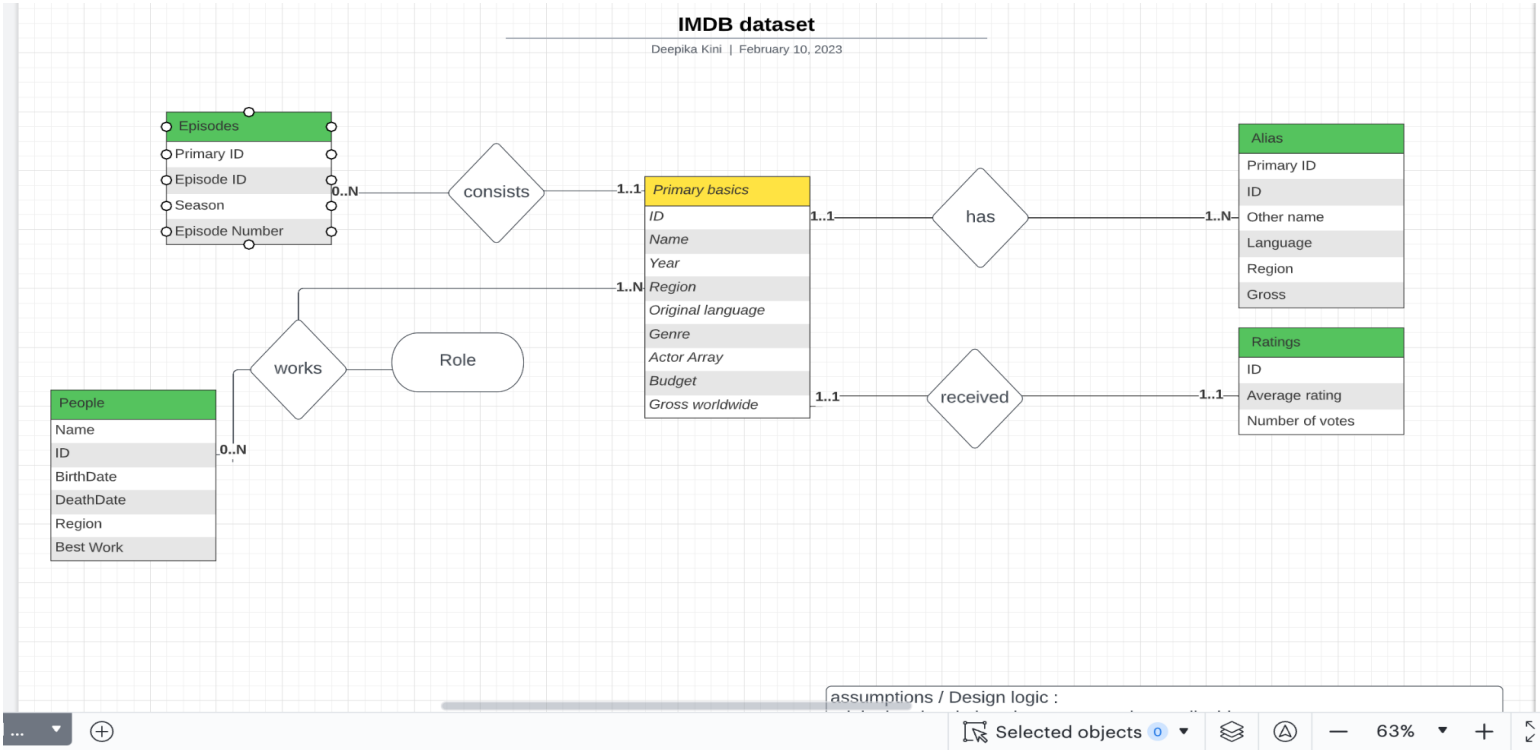
Assignment Number 1 Relational Database
-Deepika Kini

Q.1 ER diagram
Diagram snippet:



Note:
The ER diagram is kept simple and clean for better understanding of component and their dynamic (less attributes). This does not conform to the original IMDB data to avoid complexity.

Q.2
Below are the tables and their fields:

Underlined: primary key
Italicised: foreign key

title = (tconst, titleType,primaryTitle,originalTitle,isAdult, startYear, endYear, runTimeMinutes)
ratings = (*tconst*, averagerating, numvotes)
akas = (*titleId*, ordering, title, region, language, types, attributes, isOriginal)
episode =(tconst, *parenttconst*, seasonnumber, episodenumber)

genre = (genre, genre_tconst) - unique list of genres with generated ID
genre_title = (*tconst, genre_tconst*)
The above two tables help to map M:N genre details for the titles

principals= (*tconst*, ordering, *nconst*, category, job, characters)

people(or name) =(nconst, primaryname, birthyear, deathyear, primaryprofession, knownfortitles)
director_title_mapping = (*tconst, directorId*)
writer_title_mapping = (*tconst, writerId*)
The above two tables are created using crew table

SQL scripts for creating tables: please refer to file **imdb_DDL.sql**

For removing the two character prefix(slice) and converting to int(astype(int)) in foreign and primary key columns, we use python:
Eg: `df_basics['tconst'] = df_basics['tconst'].str.slice(2, ).fillna(0).astype('int')`

Q.3 Description of files in IMDB dataset:
Title basics: The primary file of the star schema which contains the details such as where it is a show/movie, start year/ release year, end year if it is a series, genre (multi-valued which is treated as an array) etc. along with a unique identifier tconst
(titletype has 11 values. This can be understood using query : select distinct(titletype) from title;)

Name: Cast details such as name, profession, best known for

Crew: writer and director identifiers for the particular movie/show identifier

Episode: contains episode details such as show number (parentTconst: referenced from title file) and the season, episode number along with its own primary key

Principals:cast and crew for movie is present. The ordering field helps to work as primary key along with tconst.

Ratings: contains average ratings and count of ratings for the particular movie/show

Akas: The movie “also known as “ details which captures local name and other details like language it was dubbed in, region of release

The link between table is as follows:
Name(people in my database) is connected to principals(1:1) and crew(split into two mapping tables)
Title is connected to Akas(1:1), ratings(1:1), episode(1:N) and principals, crew

Q.4

Python preprocessing:

I write python script to take tsv files and convert them into csv after preprocessing.
Then I use psql load commands to load data into the already created tables in postgres

1. isAdult is filtered out in python using pandas dataframe
2. The foreign key values that tend to be invalid (throws foreign key constraint error if not handled) are either removed from the file by inner joining with the primary file (one that contains the column as primary key) or in case of principals where we don't want to lose rows since they aren't present in people file, we append these extra nconst (using outer join)

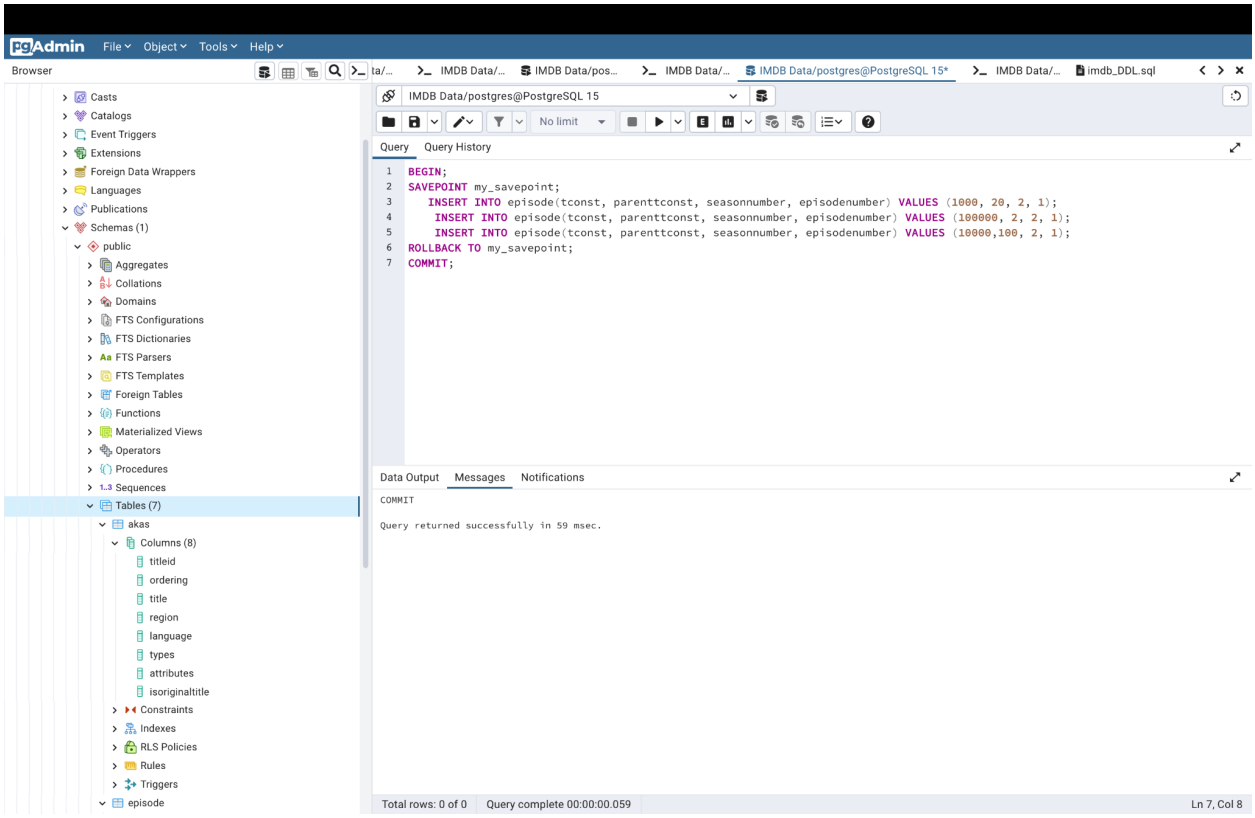
Time taken for the database to load: approx 45 mins (not done in one go so can't tell in exact time)

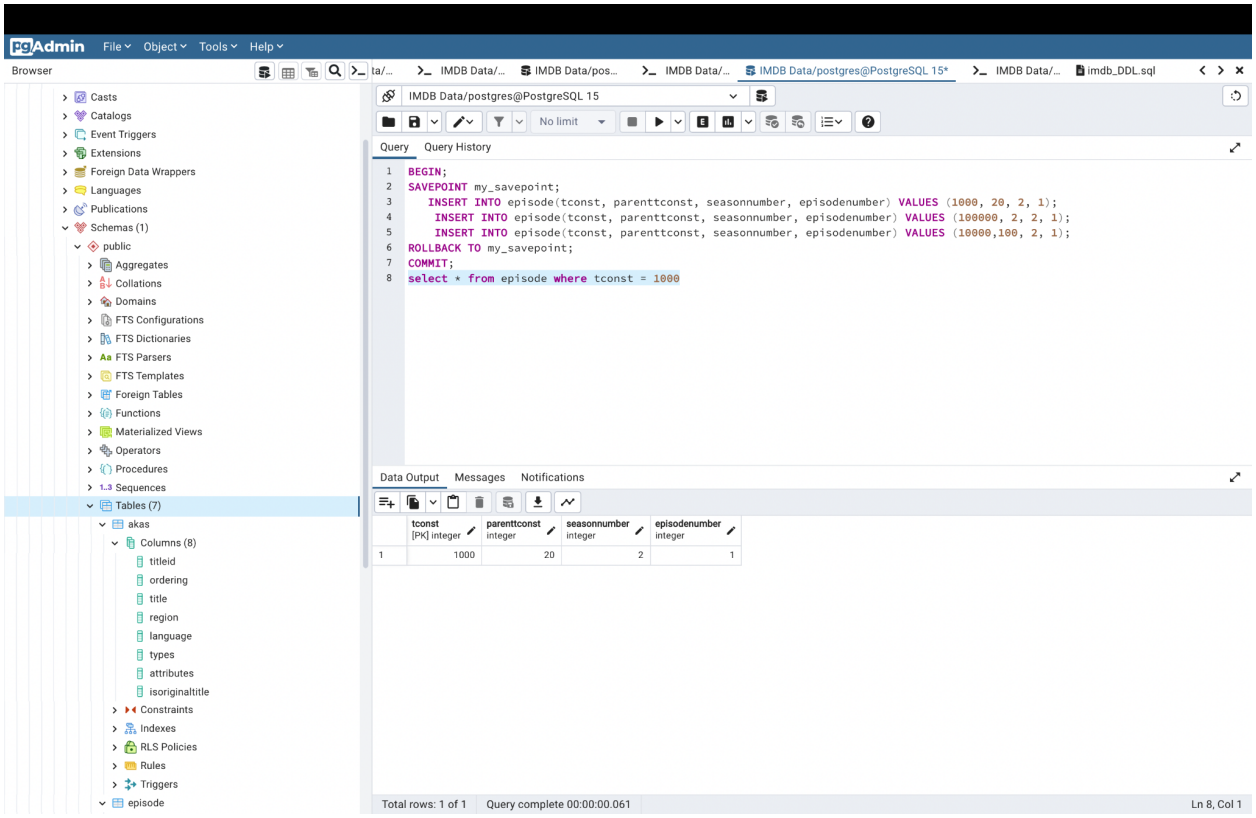
Python file that preprocesses data: **imdb.py**

Psql commands to load data: **load_data.txt**

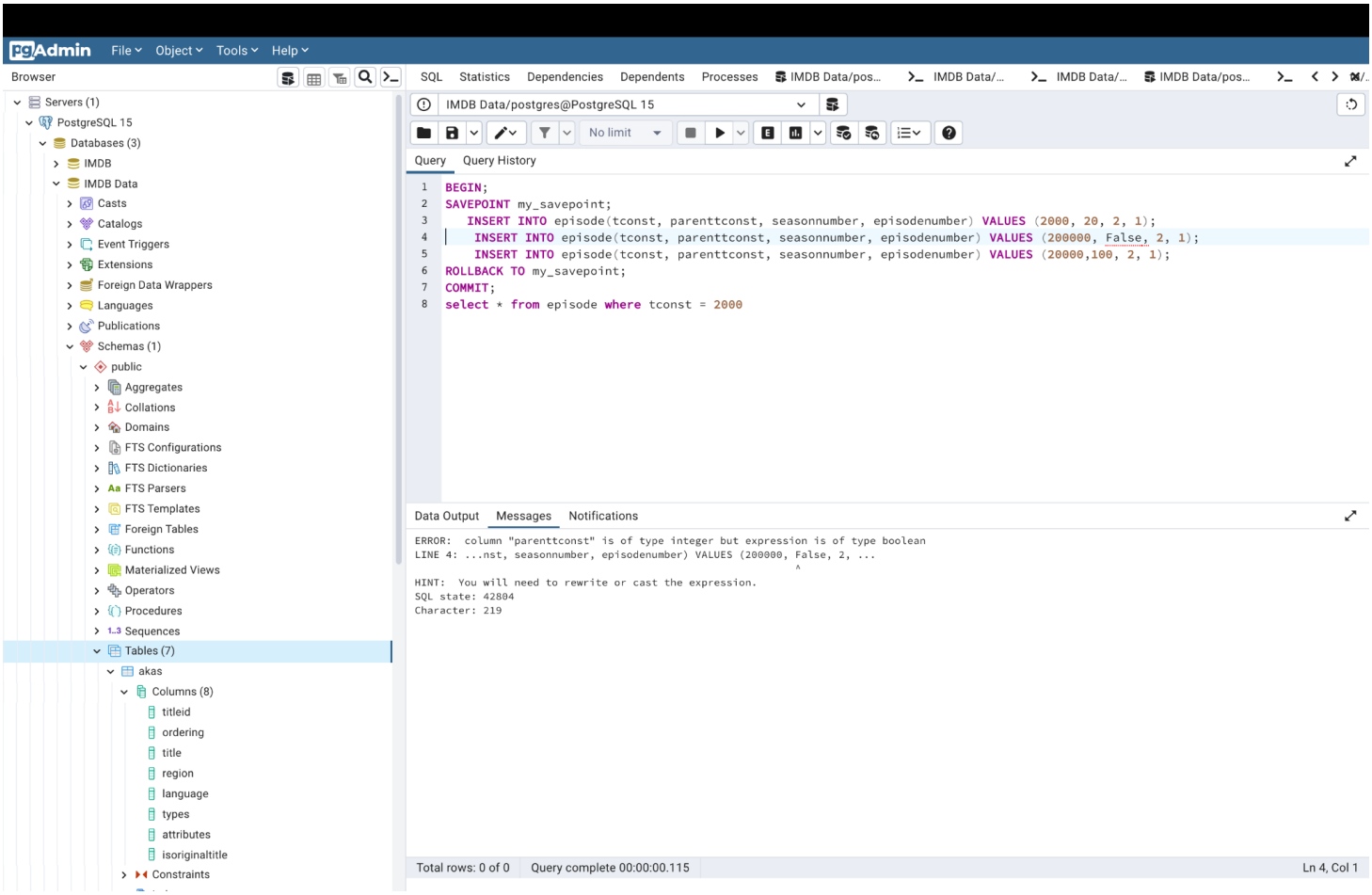
Q.5

Transaction:
The sql code is present in transaction.sql
The begin and end statements help to keep transactions atomic
Below are screenshots for correct and wrong values
Correct values added:


pgAdmin
File
Object
Tools
Help
Browser
IMDB Data/postgres@PostgreSQL 15*
imdb_DDL.sql
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (7)
akas
Columns (8)
titleid
ordering
title
region
language
types
attributes
isoriginaltitle
Constraints
Indexes
RLS Policies
Rules
Triggers
episode
IMDB Data/postgres@PostgreSQL 15
No limit
Query
Query History
1 BEGIN;
2 SAVEPOINT my_savepoint;
3 INSERT INTO episode(tconst, parenttconst, seasonnumber, episodenumber) VALUES (1000, 20, 2, 1);
4 INSERT INTO episode(tconst, parenttconst, seasonnumber, episodenumber) VALUES (100000, 2, 2, 1);
5 INSERT INTO episode(tconst, parenttconst, seasonnumber, episodenumber) VALUES (10000,100, 2, 1);
6 ROLLBACK TO my_savepoint;
7 COMMIT;
Data Output
Messages
Notifications
COMMIT
Query returned successfully in 59 msec.
Total rows: 0 of 0
Query complete 00:00:00.059
Ln 7, Col 8



Wrong data inserted in line 1 throws error:



==========THE END==========

Resources referred to:

https://www.postgresqltutorial.com/postgresql-getting-started/postgresql-sample-database/

https://northeastern-datalab.github.io/cs3200/fa18s2/download/Setup-PostgreSQL.pdf

https://blog.devart.com/create-table-in-postgresql.html

https://stackoverflow.com/questions/19463074/postgres-error-could-not-open-file-for-reading-permission-denied