IMDb Database Project – CSE 560 DMQL

Kiran, Prem and Rohith, Department of Computer Science & Engineering, SUNY Buffalo

I have read and understood the course academic integrity policy in the syllabus of the class

Abstract- This IMDb database includes comprehensive information about movies, their cast members, and crew members. The database is made up of 7 datasets that contain movie-related data. Users can find answers to frequent questions about their favorite entertainment programs using the movie database. The actors, vote totals, cast, crew, ratings, budget, revenue, posters, release dates, languages, production firms, nations, and genres are among the data items in this database. For a long time, everyone's main source of entertainment has been movies and television. Even after Covid, when everyone was confined to their houses, movies or television programs remained the only significant sources of entertainment. Searching for information on a movie, its actors or actresses, or even a simple fact is one of the most frequent activities people engage in. fetching data from the IMDb database after analysis in order to give consumers the information they require.

Keywords- IMDb**,** Movies**,** Database, Vote Counts Genres, Ratings, Analyze.

Problem Statement:

For a long time, everyone's main source of amusement has been movies and television. Even after COVID, when everyone was confined to their homes, movies or television programs remained the only significant sources of amusement. Searching for information on a movie, its actors or actresses, or even a simple fact is one of the most frequent activities people engage in fetching data from the IMDb database after analysis in order to give consumers the information they require. This IMDb database includes comprehensive information about movies, their cast members, and staff members. The database consists of 7 datasets with information related to movies.

Target user:

Our target user base would be all the people who are interested in movies. Through this solution, users can find answers to frequently asked questions about their favorite entertainment programs using the movie database. The actors, voting totals, cast, crew, ratings, budget, revenue, posters, release dates, languages, production firms, nations, and genres are among the data elements in thisdatabase.

Introduction

The main objective of this project is to build an IMDb database using the Internet Movie Database dataset. The dataset consists of 7 compressed tab-separated- value (.tsv) files in the UTF-8-character set, which are converted to (.csv) files.

The Importance and Sequence of steps involved in building this project are as following:

Importance:

Gain knowledge of building and using PostgreSQL database.

Learn about ER diagrams and relationship schemas, database normalization.

Retrieve information by implementing various complex database queries using postgres and get familiar with database manageme n t system.

Learn about querying and datavisualizing.

Steps Involved:

Analyze the dataset and represent it using an Entity-Relationship (ER)diagram.

Create a database IMDb and database elements such as schemas andtables.

Load the dataset using queries and denote the primary and foreign keyconstraints.

Create Indexes.

Check the normalization of different relations in the database.

Form few questions about the database and query them to view the results from the database.

Dataset Information

We took the dataset from the IMDb official site. The IMDb dataset consists of 7 files in a TSV format in the UTF-8- character set. We converted them into CSV files for simple loading into database. We created a table for each file and loaded it into the database. In every file the first line (header) consists information about the type of data in that particular column. There are some missing and null values for different columns in the dataset which are represented as "\N". The detailed information about all the 7 tables in the IMDb database are explained :

Title_Info: This table contains the detailsof the titles. Following are the attributes related to title information such as titleId,name, language etc. and their data types.

- titleId (string) → An alphanumeric unique identifier of the title.
- ordering (integer) → A number to uniquely identify rows for a given titleId.
- title (string) → The localized title of the show.
- region (string) → The region for this version of the title
- language (string) → The language of the title
- types (array) → Enumerated set of attributes for this alternative title. One or more of the following: "alternative", "dvd", "festival", "tv", "video", "working", "original", "imdbDisplay". New values may be added in the future without warning.
- attributes (array) → Additional terms to describe this alternative title, not enumerated
- isOriginalTitle (boolean) → 0 for not original title; 1 for original title

Title_Basics: This table gives the detailed information of the titles such as whether the tile is related to a movie or webseries etc.

- tconst (string) → Alphanumeric unique identifier of the title.
- titleType (string) → The type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc.).
- primaryTitle (string) → The more popular title / the title used by the filmmakers on promotional materials at the point of release.
- originalTitle (string) → Original title, in the original language.
- isAdult (boolean) → 0 for non-adult title; 1 for adult title.
- startYear (YYYY) → Represents the release year of a title. In the case of TV Series, it is the series start year.
- endYear (YYYY) → TV Series end year. '\N' for all other title types.
- runtimeMinutes → Primary runtime of the title, in minutes
- genres (string array) → Includes up to three genres associated with the title.

Title_Crew: This table gives information aboutcrew members such as director, writers etc.

- tconst (string) → alphanumeric unique identifier of the title
- directors (array of nconsts) → director(s) of the given title
- writers (array of nconsts) → writer(s) of the given title

Title_Episode:
This table contains details aboutthe TV episode information.

- tconst (string) → alphanumeric identifier of episode.
- parentTconst (string) → alphanumeric identifier of the parent TV Series.
- seasonNumber (integer) → season number the episode belongs to.
- episodeNumber (integer) → episode number of the tconst in the TV series.

Title_Principal_Crew: This table contains detailed information of the cast and crew of titles.

- tconst (string) → alphanumeric unique identifier of the title.
- ordering (integer) → a number to uniquely identify rows for a given titleId.
- nconst (string) → alphanumeric unique identifier of the name/person.
- category (string) → the category of job that person was in.

- job (string) → the specific job title if applicable, else '\N'.
- characters (string) → the name of the character played if applicable, else '\N'.

Title_Ratings: This table contains the IMDB ratings and votes polled by audience for different titles.

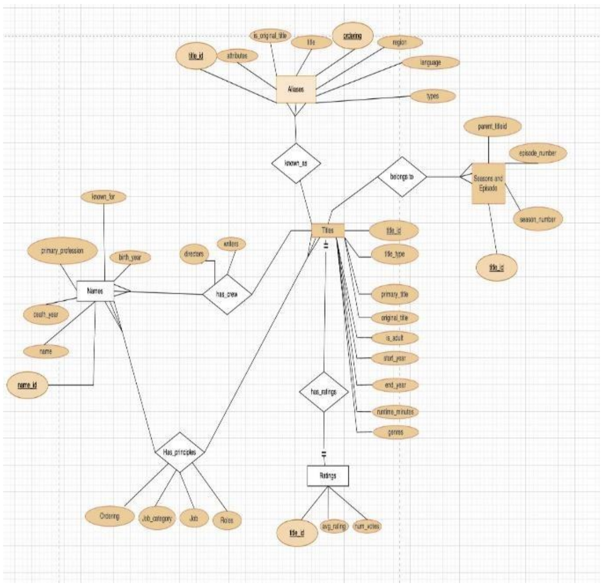
- tconst (string) → alphanumeric unique identifier of the title.
- averageRating → weighted average of all the individual user ratings.
- numVotes → number of votes the title has received.

Crew_Basic_Info: This table contains information related to crew members of the movie or TV show.

- nconst (string) → alphanumeric unique identifier of the name/person.
- primaryName (string) → name by which the person is most often credited.
- birthYear → in YYYY format.
- deathYear → in YYYY format if applicable, else '\N'.
- primaryProfession (array of strings) → the top-3 professions of the person.
- knownForTitles (array of tconsts) → titles the person is known for.

1) Entity-Relationship(ER) Diagram:

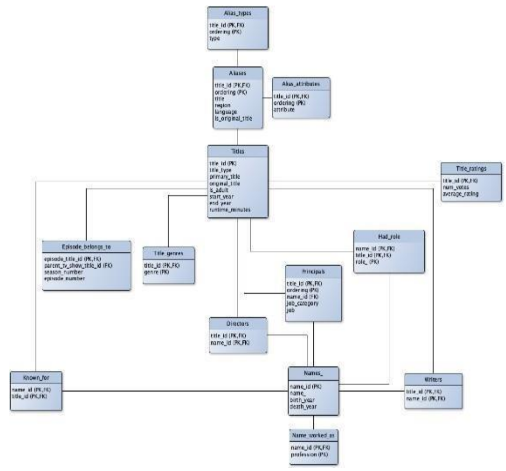
We designed an Entity- Relationship diagram for our IMDb database as our database provided is not normalized. It is shown as below:



2) Logical Schema:

After creating ER diagram, we normalized our database and obtained the logical schema which can be shown and explained below:

- We created few new tables for multi valued attributes.
- As many titles do not have ratings, we removed out the rating information attribute from the Titles relation.
- To avoid multiple NULL values while storing the data in the Titles table we removed it.
- Therefore, we kept the ratings information separately into the Title_Ratings table.
- There are some other sub relations such as directors, principals for better understanding of the data.



3) IMDb Relations – FD's (1NF,2NF,3NF,BCNF)

The functional dependencies and normalization for each relation in our database is explained below:

- Consider one relation that fully describes how our database was normalized.
- Functional Dependencies of the Relation is :
  {title_id → title_name, title_type, primary_title, gender, original_title}.
- Since the above relation only contains atomic values and doesn't have any dependent multi-valued attributes attached to it, we may say that it is in 1NF.
- The title_id is the principal key in the aforementioned functional dependence. Consequently, title_id is the primary attribute. Additionally, non-prime qualities include title_name, title_type, original_title, and gender. Since there are no partial dependencies between non-prime and prime qualities, the connection is in 2NF.
- We may infer that the relation is in 3NF because it is in 2NF and there are no transitive dependencies between the non-prime and prime characteristics.
- As the dependent attribute set uniquely identifies the dataset, it serves as a super key. As a result, the attribute set is a super key in the functional dependency mentioned above. The aforementioned relation is hence in BCNF.

Every relationship in this database complies with normalization requirements and follows BCNF.

Creating and implementing IMDb database

Below are the steps we followed to implement the IMDb database:

a. Create a database server called IMDb.
b. Create 7 different tables using create queries as mentioned in the **Create.sql** file.
c. Load the data files from local into the created tables using the following commands.

All the create table queries are mentioned in the **Create.sql** file.
All the load data queries are mentioned in the **Load.sql** file.
All the SQL queries tested on the database are mentioned in the **Queries.sql** file.
All the csv formatted data files mentioned in the **.dat** file.

Below are the queries we used to load the data into the database.

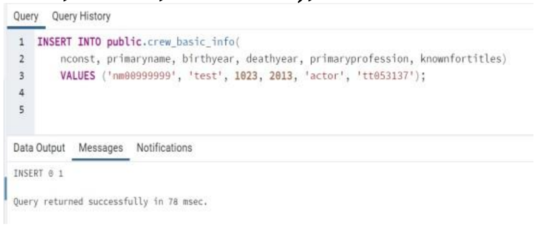
```
copy Title_Information FROM 'C:\Users\Public\title_akas.csv' with csv header;

copy Title_Basic_Info FROM 'C:\Users\Public\title_basics.csv' with csv header;

copy Title_Crew_Info FROM 'C:\Users\Public\title_crew.csv' with csv header;

copy Title_Episode_Info FROM 'C:\Users\Public\title_episode.csv' with csv header;

copy Title_Principal_Cast FROM 'C:\Users\Public\title_principals.csv' with csv header;

copy Title_Ratings FROM 'C:\Users\Public\title_ratings.csv' with csv header;

copy Crew_Basic_Info FROM 'C:\Users\Public\name_basics.csv' with csv header;
```

4) SQL Queries

Our project's primary goal is to create a database where users can access data on movies, so we created a few sample queries that do just that. These are just a few of the searches that may be made utilizing our database. Here are a few examples of queries and the answers returned by our database.

I. Insert a row

**INSERT INTO public.crew_basic_info(nconst, primaryname, birthyear, deathyear, primaryprofession, knownfortitles)
VALUES ('nm00999999', 'test', 1023, 2013, 'actor', 'tt053137');**

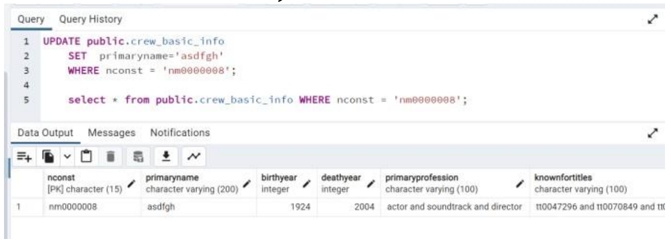


```
INSERT INTO public.crew_basic_info(
    nconst, primaryname, birthyear, deathyear, primaryprofession, knownfortitles)
    VALUES ('nm00999999', 'test', 1023, 2013, 'actor', 'tt053137');
```

```
INSERT 0 1

Query returned successfully in 78 msec.
```

II. Update a row

**UPDATE public.crew_basic_info
SET primaryname='asdfgh'
WHERE nconst = 'nm0000008';
select * from public.crew_basic_info WHERE nconst = 'nm0000008';**

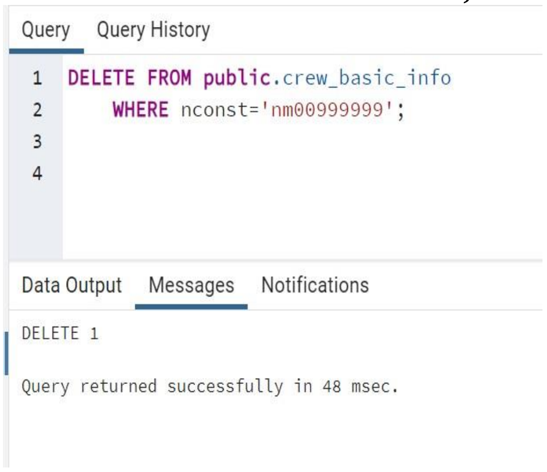


```
UPDATE public.crew_basic_info
    SET  primaryname='asdfgh'
    WHERE nconst = 'nm0000008';

    select * from public.crew_basic_info WHERE nconst = 'nm0000008';
```

	nconst [PK] character (15)	primaryname character varying (200)	birthyear integer	deathyear integer	primaryprofession character varying (100)	knownfortitles character varying (100)
1	nm0000008	asdfgh	1924	2004	actor and soundtrack and director	tt0047296 and tt0070849 and tt

III. Delete a row

**DELETE FROM public.crew_basic_info
WHERE nconst='nm00999999';**

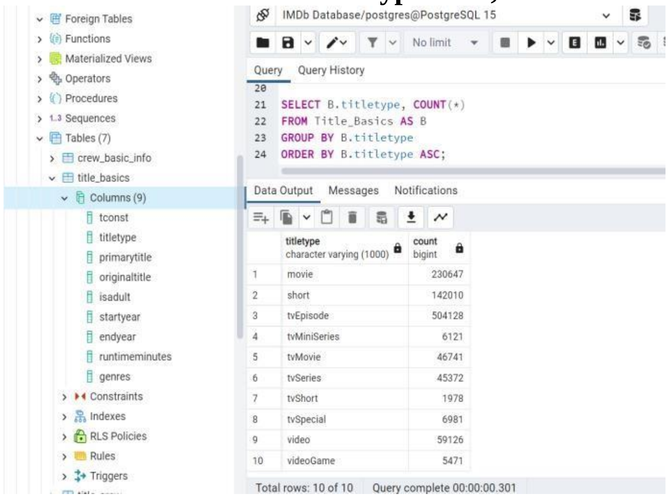


```
DELETE FROM public.crew_basic_info
    WHERE nconst='nm00999999';
```

```
DELETE 1

Query returned successfully in 48 msec.
```

IV. How many different types of titles arethere in the database?

SELECT B.titletype, COUNT(*)FROM Title_Basics AS B GROUP BY B.titletype ORDER BY B.titletype ASC;



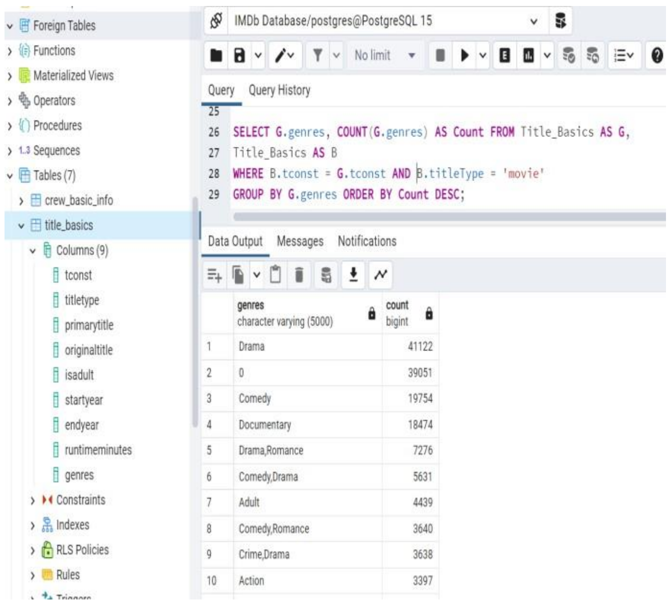
```
SELECT B.titletype, COUNT(*)
FROM Title_Basics AS B
GROUP BY B.titletype
ORDER BY B.titletype ASC;
```

	titletype character varying (1000)	count bigint
1	movie	230647
2	short	142010
3	tvEpisode	504128
4	tvMiniSeries	6121
5	tvMovie	46741
6	tvSeries	45372
7	tvShort	1978
8	tvSpecial	6981
9	video	59126
10	videoGame	5471

Total rows: 10 of 10 Query complete 00:00:00.301

V. How many different types of genres are there and number of movies in each genre?

SELECT G.genres, COUNT(G.genres) ASCount FROM Title_Basic_Info AS G, Title_Basic_Info AS B WHERE B.tconst = G.tconstAND B.titleType = 'movie' GROUP BY G.genres ORDER BY Count DESC;

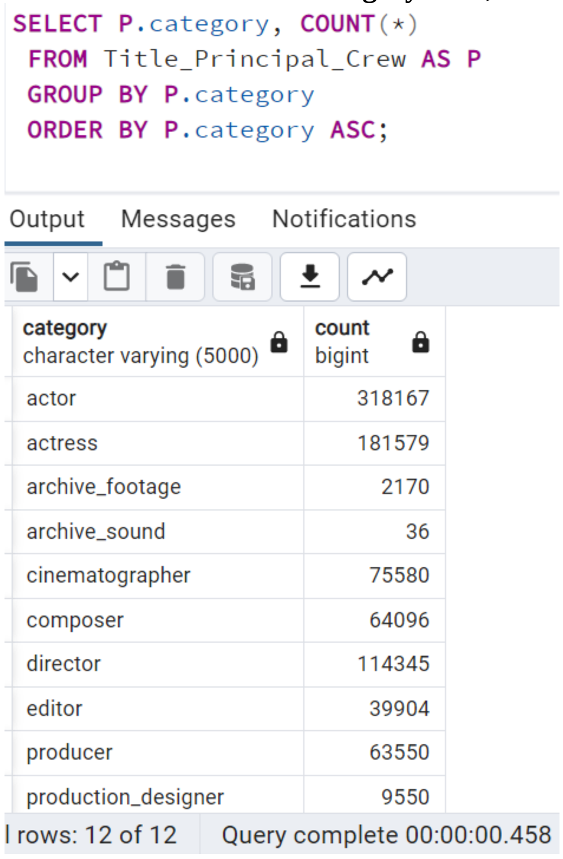


```
SELECT G.genres, COUNT(G.genres) AS Count FROM Title_Basics AS G,
Title_Basics AS B
WHERE B.tconst = G.tconst AND B.titleType = 'movie'
GROUP BY G.genres ORDER BY Count DESC;
```

	genres character varying (5000)	count bigint
1	Drama	41122
2	0	39051
3	Comedy	19754
4	Documentary	18474
5	Drama,Romance	7276
6	Comedy,Drama	5631
7	Adult	4439
8	Comedy,Romance	3640
9	Crime,Drama	3638
10	Action	3397

VI. What are the different types of professions in the database and the count of each type of profession?

SELECT P.category, COUNT(*)
FROM Title_Principal_Crew AS P
GROUP BY P.category
ORDER BY P.category ASC;



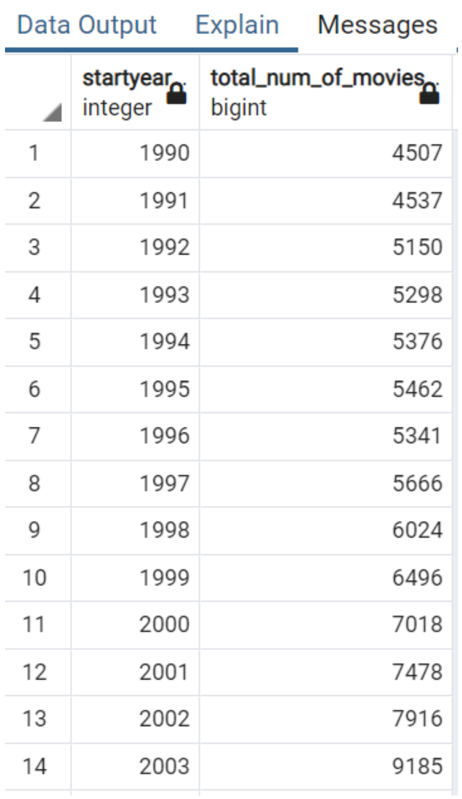
```
SELECT P.category, COUNT(*)
 FROM Title_Principal_Crew AS P
 GROUP BY P.category
 ORDER BY P.category ASC;
```

category character varying (5000)	count bigint
actor	318167
actress	181579
archive_footage	2170
archive_sound	36
cinematographer	75580
composer	64096
director	114345
editor	39904
producer	63550
production_designer	9550

```
rows: 12 of 12    Query complete 00:00:00.458
```

VII. What is the count of movies made in each year from 1990 to 2019 in ascending order?

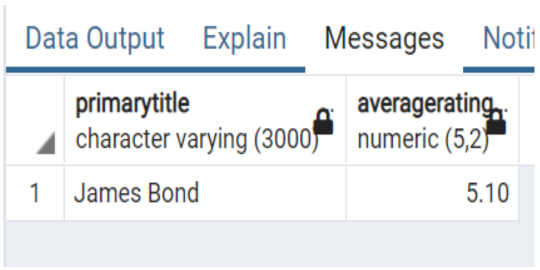
SELECT B.startYear, COUNT(*) AS Total_Num_Of_Movies
FROM Title_Basics AS B
WHERE B.titleType IN ('movie','video')
GROUP BY B.startYear
HAVING B.startYear BETWEEN 1990 AND 2019 ORDER BY B.startYear ASC;


Data Output | Explain | Messages

	startyear integer	total_num_of_movies bigint
1	1990	4507
2	1991	4537
3	1992	5150
4	1993	5298
5	1994	5376
6	1995	5462
7	1996	5341
8	1997	5666
9	1998	6024
10	1999	6496
11	2000	7018
12	2001	7478
13	2002	7916
14	2003	9185

VIII. What is the count of movies made in each year from 1990 to 2019 in ascending order?

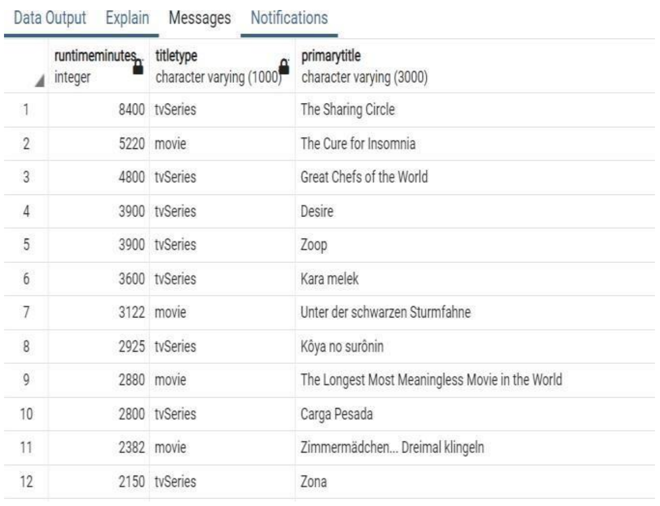
SELECT B.primaryTitle, R.averageRating
FROM Title_Basics AS B, Title_Ratings AS R
WHERE B.tconst = R.tconst
AND B.titleType = 'movie' AND B.primaryTitle like 'James Bond';


Data Output | Explain | Messages | Noti

	primarytitle character varying (3000)	averagerating numeric (5,2)
1	James Bond	5.10

IX. What is the count of movies made in each year from 1990 to 2019 in ascending order?

SELECT runtimeMinutes, titleType, primaryTitle FROM Title_Basics WHERE runtimeMinutes >(10*60) ORDER BY runtimeMinutes DESC, titleType ASC;

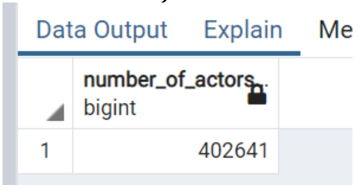

Data Output | Explain | Messages | Notifications

	runtimeminutes integer	titletype character varying (1000)	primarytitle character varying (3000)
1	8400	tvSeries	The Sharing Circle
2	5220	movie	The Cure for Insomnia
3	4800	tvSeries	Great Chefs of the World
4	3900	tvSeries	Desire
5	3900	tvSeries	Zoop
6	3600	tvSeries	Kara melek
7	3122	movie	Unter der schwarzen Sturmfahne
8	2925	tvSeries	Kôya no surônin
9	2880	movie	The Longest Most Meaningless Movie in the World
10	2800	tvSeries	Carga Pesada
11	2382	movie	Zimmermädchen... Dreimal klingeln
12	2150	tvSeries	Zona

X. Total number of actors present in the entire database ?

CREATE OR REPLACE VIEW Num(Number_of_actors)
AS SELECT COUNT(DISTINCT N.nconst) AS Number_of_actors
FROM Crew_Basic_Info AS N
WHERE N.primaryprofession IN ('actor','actress');

select * from Num;

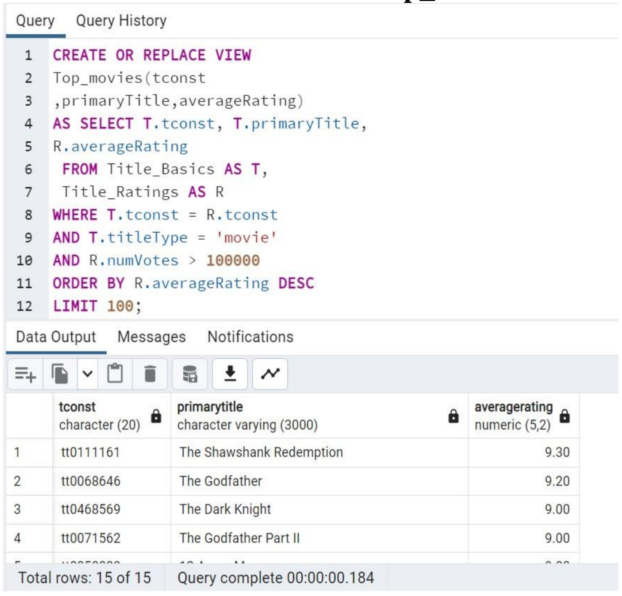

Data Output | Explain | Me

	number_of_actors bigint
1	402641

XI. Top 100 movies based on ratings with votes more than 100,000 ?

CREATE OR REPLACE VIEW Top_movies(tconst ,primaryTitle,averageRating)
AS SELECT T.tconst, T.primaryTitle, R.averageRating
FROM Title_Basics AS T,
Title_Ratings AS R
WHERE T.tconst = R.tconst
AND T.titleType = 'movie'
AND R.numVotes > 100000
ORDER BY R.averageRating DESC
LIMIT 100;

SELECT * FROM Top_movies LIMIT 15;



```
CREATE OR REPLACE VIEW
Top_movies(tconst
,primaryTitle,averageRating)
AS SELECT T.tconst, T.primaryTitle,
R.averageRating
 FROM Title_Basics AS T,
 Title_Ratings AS R
WHERE T.tconst = R.tconst
AND T.titleType = 'movie'
AND R.numVotes > 100000
ORDER BY R.averageRating DESC
LIMIT 100;
```

	tconst character (20)	primarytitle character varying (3000)	averagerating numeric (5,2)
1	tt0111161	The Shawshank Redemption	9.30
2	tt0068646	The Godfather	9.20
3	tt0468569	The Dark Knight	9.00
4	tt0071562	The Godfather Part II	9.00

Total rows: 15 of 15 Query complete 00:00:00.184

5) Individual Contribution

Team: 12

Sai Kiran Mundra (smundra)

Prem Salina (premsali)

Rohith Varma Palukuri (rpalukur)

a) Sai Kiran:

The data files were converted and loaded. two tables, Title_Information and Title_Basic_Info, were created and filled with data. created and put into use logical schema diagram. checked the links between 1NF and 2NF. designed four queries,
The IMDb database was used to test them. prepared a report and a presentation for the section in the presentation that was explained.
video.

b) Prem Salina:

The data files were converted and loaded. 3 tables — Title_Episode_Info, Title_Principal_Cast, and Title_Ratings — were created and loaded with data. Entity-Relationship (ER) diagram created and used. checked the relationships between 3NF and BCNF. 3 queries were created and evaluated on the IMDb database. report and presentation that were prepared for the section covered in the presentation video.

c) Rohith Varma:

The data files were converted and loaded. Title_Crew_Info and Crew_Basic_Info tables were created and loaded with data. Entity-Relationship (ER) diagram created and used. checked the relationships between 3NF and BCNF. 4 queries were created and tested on the IMDb database. report and presentation that were prepared for the section covered in the presentation video.