CSE – 560, Data Model and Query Language
IMDB Dataset

Milestone: 2

Project Details:

Team Crew - X

Name: Mahalakshmi Chintala
UBIT: mchintal
UBID: 50442308

Name: Harinee Purushothaman
UBIT: harineep
UBID: 50442192

Name: Anudeep Balagam
UBIT: abalagam
UBID: 50442091

I. Introduction

Not only does a popular movie amuse viewers, but it also makes huge profits for the film industry. A variety of elements, including talented directors and seasoned actors, are important for making good movies. Famous actors and filmmakers can, however, always bring in the anticipated box office revenue but cannot ensure a high imdb rating. So, we use this imdb rating system to tackle this problem.

II. Problem Statement

The IMDB dataset consists of the details of several movies, tv series and helps a user decide whether to watch the show or movie based on their ratings.

A. *Usage of Database over Excel*

- The use of spreadsheets for numeric and text values is effective in relatively small volumes. Also capable of incorporating other types of information, such as images and documents, databases can handle numeric and text values expertly.
- Since the IMDb Databases consists of more than a million records, it can also accommodate high volume and large file size data downloads. Generally, this includes those from data loggers, GPS devices, cameras, drones, and other collection devices.
- However, in the given scenario, it is easier to deal with data having nulls with a dataset over an excel sheet.

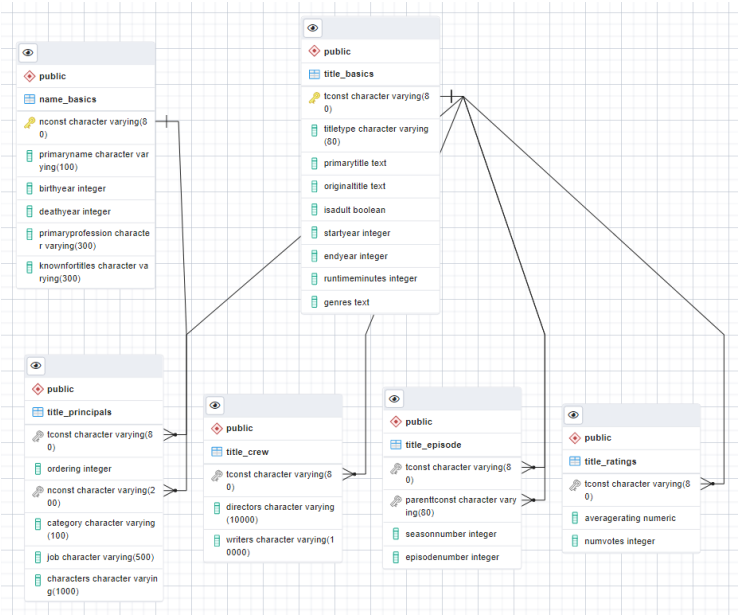
III. Target User

General public or any enthusiast to find out information and ratings of different movies or series, short films, documentaries deciding to view it.

Also, another end user for imdb database system are users who want to rate or review movies or contribute data must sign up for an IMDb account.

The import and upkeep of the database platform fall under the purview of administrators.

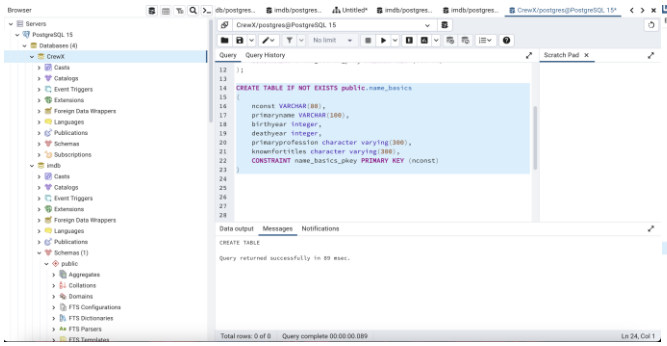
IV. ER Diagram

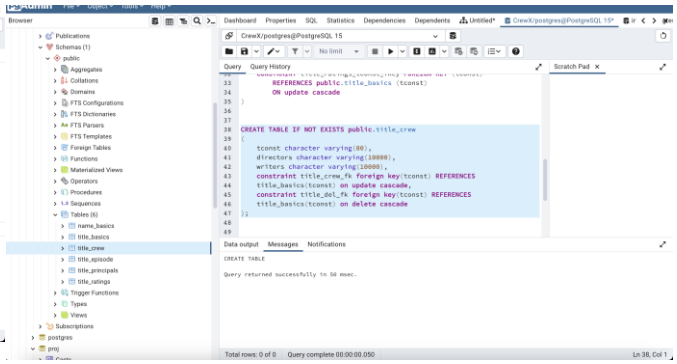


A. *Relationship between relations and attributes:*

The **name_basics** table consists of the below attributes:

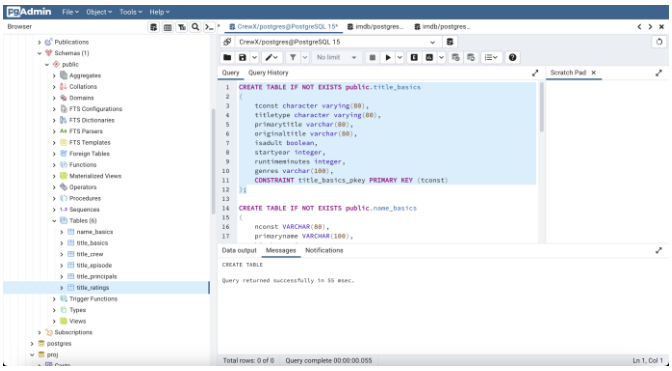
- nconst (varchar) – Primary key, alphanumeric identifier of name of the artist.
- primaryName (varchar)– Most commonly credited name of the individual.
- birthyear (int) – Birth year of an individual artist.
- deathyear (int)– Death year of an individual, if applicable, else "\N".
- primaryProfession (text) – This attribute is an array of text consists of the top-3 professions of the person.
- knownForTitles (text) – Titles the person is known for.





The **title_basics** relation consists of the below:

- tconst (varchar) – Primary key of the table, which is an alphanumeric unique identifier of the title, equivalent to titleID.
- titletype (varchar) – format of the title
- primarytitle (varchar) – the title used by the filmmakers for promotions
- originaltitle (varchar) – Original title of the movie/show
- isadult (Boolean) – true i.e. 1 for adult movies and false i.e., 0 for the rest
- startyear (integer) – release year of the movie/ year of start of the series
- endyear (integer) – end year of tv series and '\N' for all other types
- runtimeminutes (integer) – screentime of the title in minutes
- genres (text) – consists of up to 3 genres associated with the title

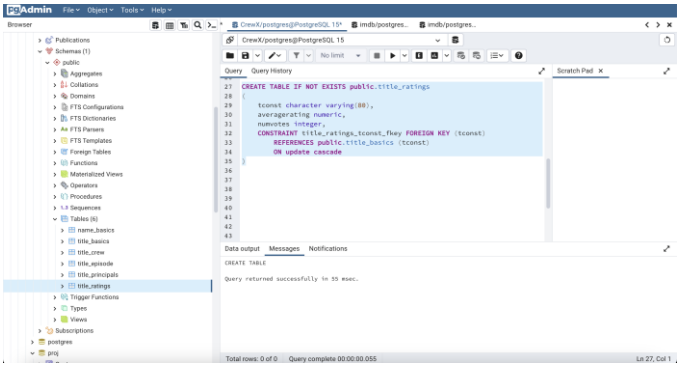


The **title_crew** holds the director and writer data for all the titles in IMDb. Fields include:

- tconst (varchar) – title identifier which is a foreign key being referenced from tconst of title_basics
- directors (varchar) - director(s) of the given title
- writers (varchar) – writer(s) of the given title

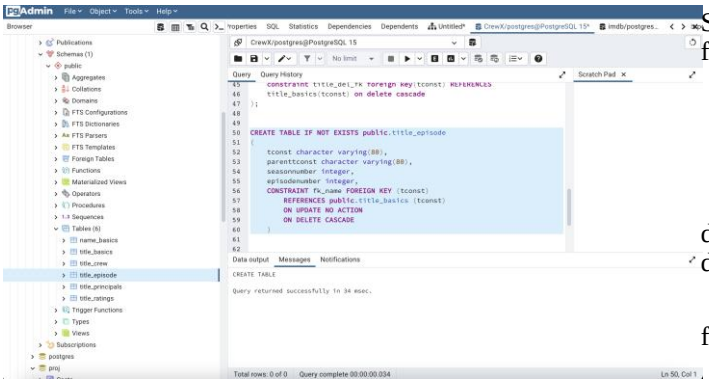
The **title_ratings**: Listed below are the IMDb ratings and votes for the following titles.

- tconst (varchar) – title identifiers being references from tconst of title_basics
- averageRating (numeric)– average of the user ratings
- numVotes (integer) - number of votes received by the title



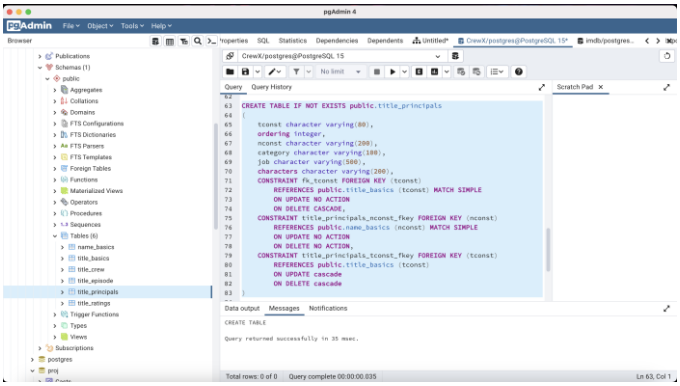
The **title_episode** relation consists of the TV Series' epsiodes information. Fields include:

- tconst (varchar) – episode identifiers being referenced from the tconst of the title_basics
- parenttconst (varchar) –parent TV Series' ID referred from tconst of title_basics as foreign key
- seasonumber (integer) – season number of the episode
- episodenumber (integer) – episode number of the TV series



The **title_principals** table has information about the titles' cast and/or crew, with the following attributes:

- tconst (varchar) – title identifiers referenced from tconst of the title_basics relation
- ordering (integer) – a number to uniquely identify rows for a given titleId
- nconst (varchar) – foreign key of name identifier being referenced from the name_basics table
- category (varchar) - the category of job that artist



B. *Constraints:*

The following attributes have the primary key constraint: nconst and tconst

The tconst and nconst attributes in the tables title_crew, title_ratings, title_episode, title_principals has been referenced from title_basics and name_basics as the foreign key.

The attributes primarytitle and isadult from the schema title_basics is set to NOT NULL as a constraint.

C. *Delete Cascade:*

The cascade delete feature of a foreign key ensures that when a record in the parent table is deleted, the corresponding records in the child table will also be deleted. SQL Server calls this a cascade delete.

Hence, we have applied the delete cascade constraint over foreign keys present in title_crew, title_ratings, title_episode, title_principals.

On deletion of tconst tuple from the title_basics table, tuples from the rest of the tables will also get deleted. Similarly, if nconst o name_basics is deleted, the records from title_principals will automatically get deleted.

V. BCNF

For a relation to be in BCNF, all of its functional dependents must be non-trivial, and the functional dependency's left side must be the relation's super key.

The initial IMDb schema is in BCNF already due to the following functional dependencies:

- Title.basics: Tconst-> title_type, originaltitle, is_adult, start_year, runtime_minutes, genre
- Names: Nconst-> primaryName, birth_year, death_year, primary_profession
- Principals: Tconst, nconst -> ordering, category, job, characters
- Ratings: tconst-> average_ratings, numvotes
- Crew: tconst-> directors, writers
- Episode: tconst-> parentTconst, seasonNumber, episodeNumber

There are additional possible functional dependencies but as they violate BCNF, we have considered only the above.

VI. Problems Faced while using the large dataset:

- Recognizing nulls
- Size of tuples and attributes being larger
- Excess execution time for queries with more than one retrieving detail and from multiple tables

A. *Solutions:*

- Altering query attributes expanding their size
- Indexing

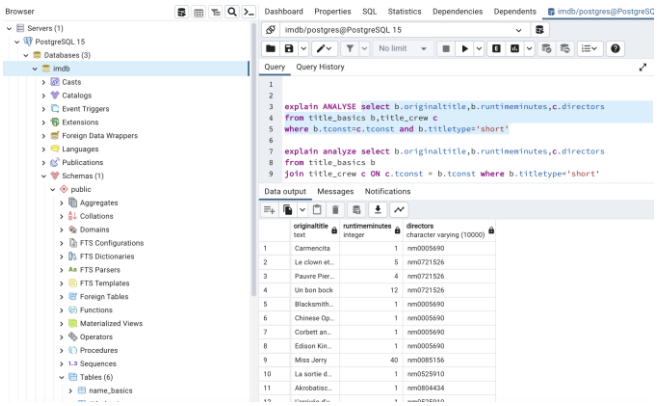
Given that Postgres already produces an index for primary keys by default, it can be shown that when several attributes from the query are returned and indexed individually, the execution time is drastically reduced.

The usage of explain analyze helps understanding the query plan, process executed in running a query. EXPLAIN only supports SELECT, INSERT, UPDATE, DELETE, EXECUTE (of a prepared statement), CREATE TABLE... AS, and DECLARE. It does not handle other types of statements (of a cursor).

VII. Queries

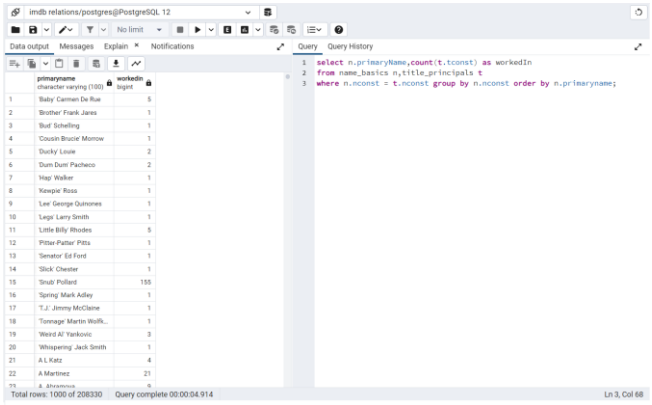
A. SELECT Query 1

A query, to fetch the titles, their screen-time and directors for the title type is short.



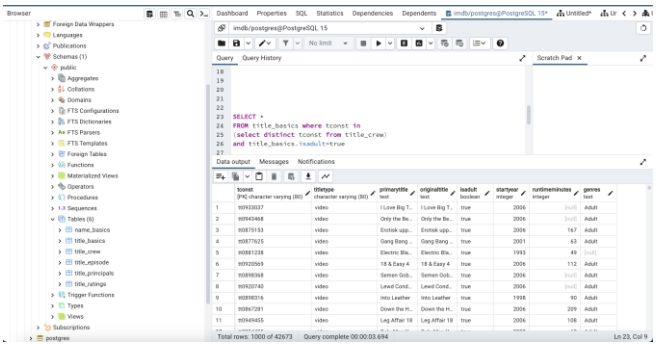
B. SELECT Query 2

Finding name of people who have played more than one part in films along with their count



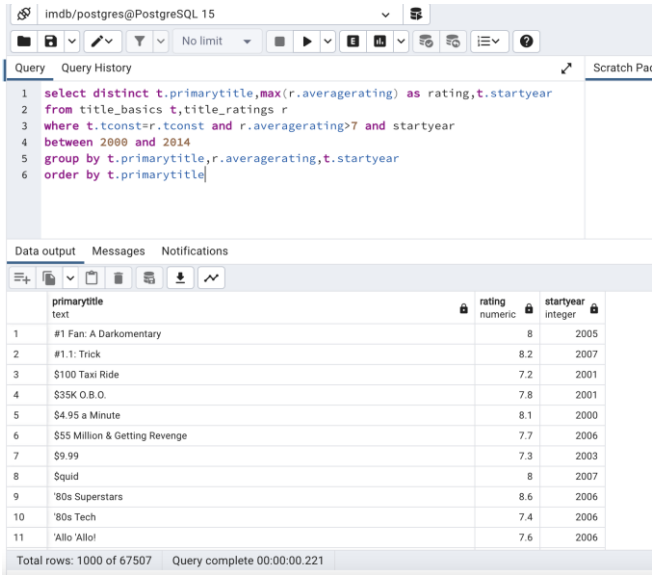
C. SELECT Query 3

A query to find all the titles that are adult rated.



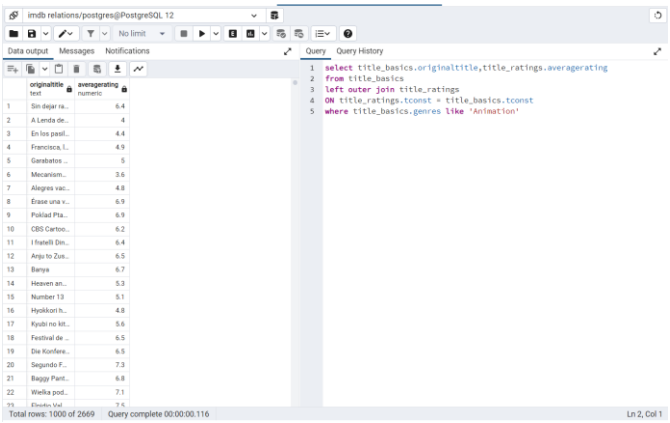
D. SELECT Query 4

Query to find movie titles that have an average rating of 7 and started in between the years 2000 to 2014.



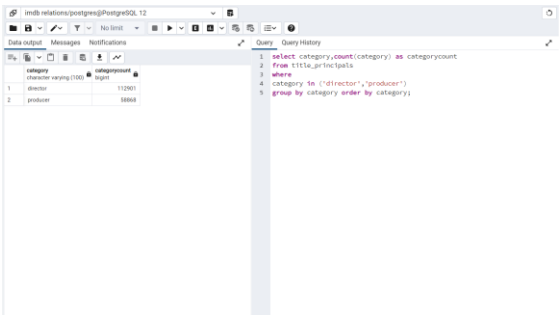
E. SELECT Query 5

Query to find names of every animated movie along with their rating.



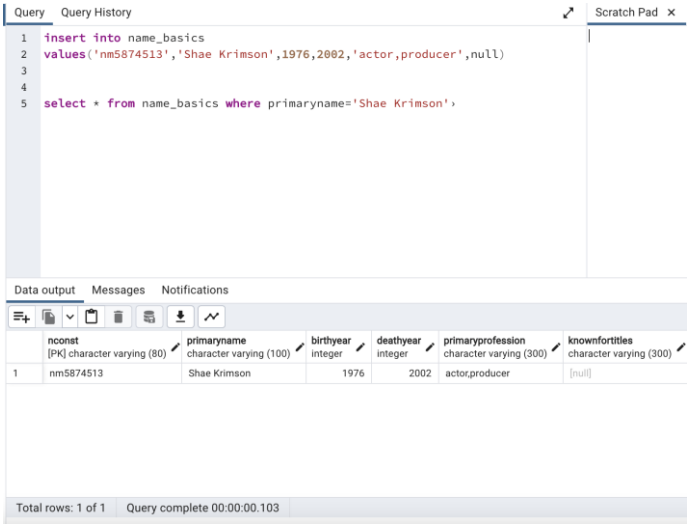
F. SELECT Query 6

Query to find number of titles done by the director being same as writer.



G. *INSERT Query 7*

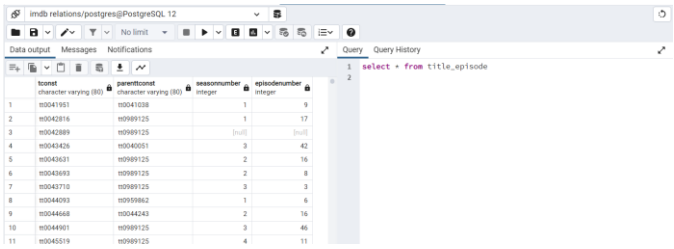
Query to insert values into **name_basics** table.



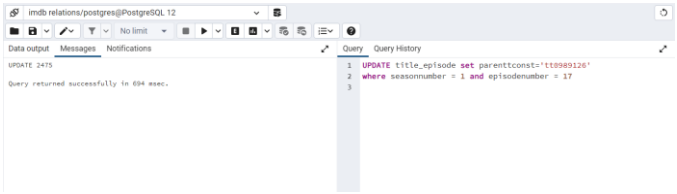
H. *Update Query 8*

Update the parentconst value to 'tt0989126' where seasonnumber = 1 and episodenumber = 17 from **title_episode**.

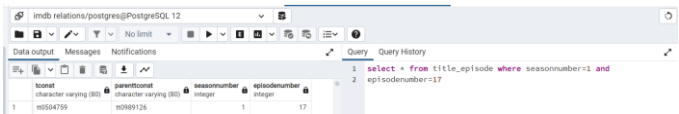
Before updating:



Update:

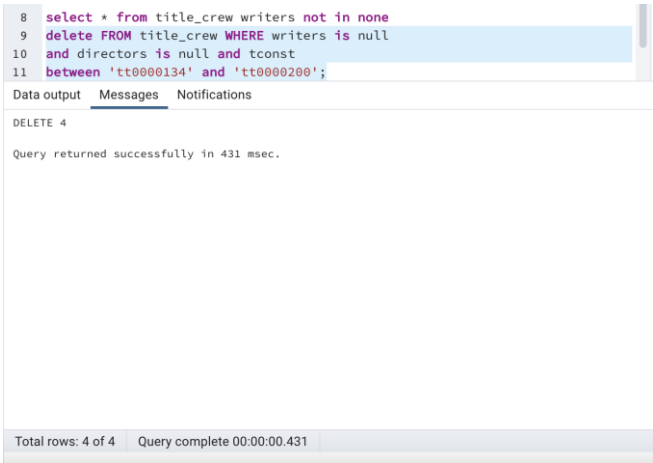


After updating:



I. *Delete Query 9*

Deleting titles with no directors or writers within the title ids between 0134 to 0200.

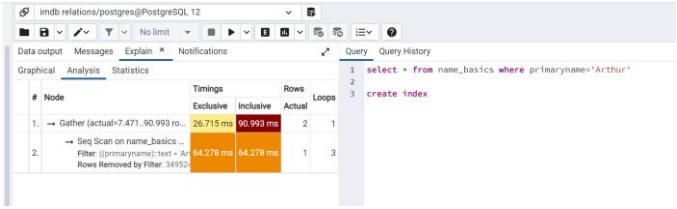


VIII. Query Execution Analysis – Using Explain Tool

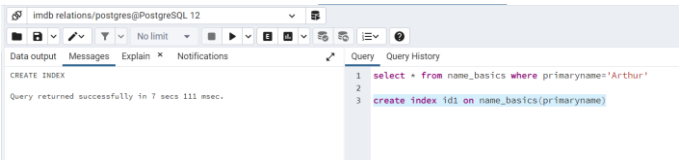
Performed indexing on three different quires and observed the change in execution time.

A. *Performed indexing on primaryname column of **name_basics** table this reduced the execution time:*

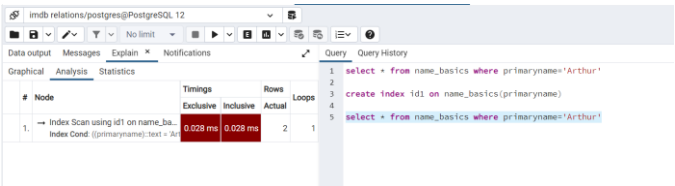
Before Indexing:



Indexing:

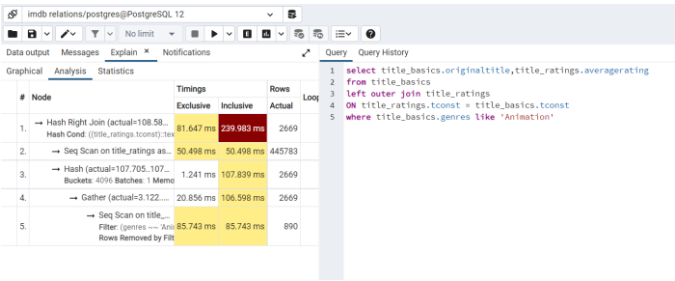


After Indexing:

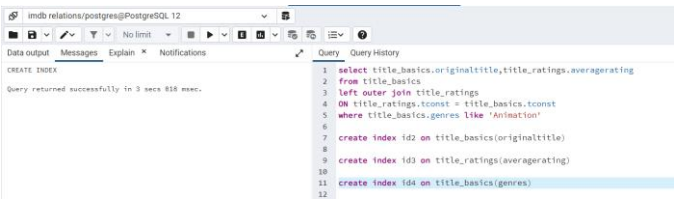


B. Performed indexing on originaltitle, averagerating and genre columns of title_basics this greatly reduced the exceution time of the SELECT Query 5:

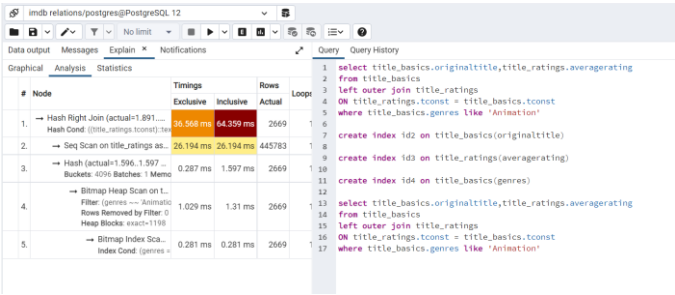
Before Indexing:



Indexing:

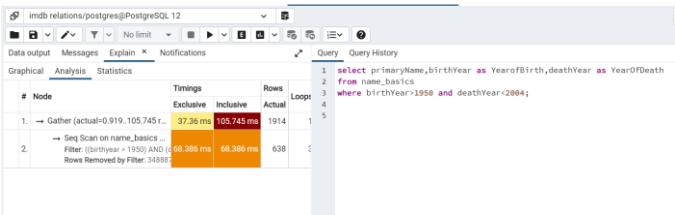


After Indexing:

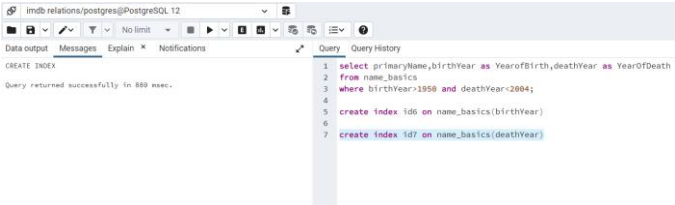


C. Performing indexing on birthYear and deathYear column of name_basics table this also reduces the runtime of the query:

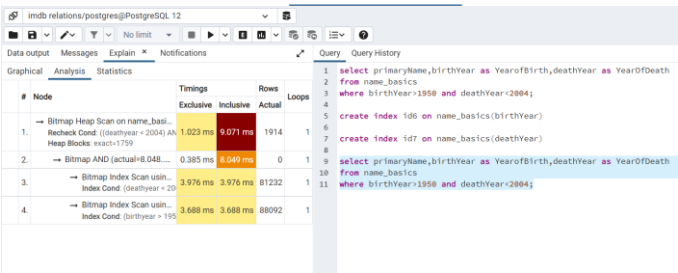
Before Indexing:



Indexing:



After Indexing:



IX. References

https://www.postgresql.org/docs/

https://www.w3schools.blog/postgresql-tutorial

X. Contribution

Task	Task assigned to
Creation of tables	Anudeep
Insertion of data observing nulls	Mahalakshmi
Insertion of data observing nulls	Harinee
select queries(3)	Anudeep
select queries(3)	Mahalakshmi
select queries(3)	Harinee
ER Diagram	Mahalakshmi
Indexing	Harinee
Indexing	Mahalakshmi
Indexing	Anudeep
BCNF	Harinee
Update & delete	Anudeep