Comprehensive Analysis of Netflix Movies and TV Shows Using SQL

Milestone: Project Documentation

Group 08

Sree Ramya Pasala
Sri Lakshmi Swetha Jalluri

857-339-9096 (Tel of Student 1)

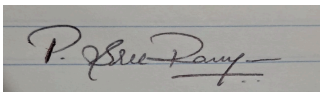
857-379-0666 (Tel of Student 2)

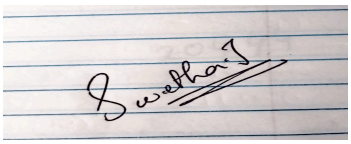
Pasala.sr@northeastern.edu

jalluri.s@northeastern.edu

Percentage of Effort Contributed by Student1: 50

Percentage of Effort Contributed by Student2: 50

Signature of Student 1: 

Signature of Student 2: 

Submission Date: 12-06-2024

USE CASE STUDY REPORT

Group No: Group 08

Students Names: Pasala Sree Ramya, Sri Lakshmi Swetha Jalluri

Executive Summary:
In order to save and analyse information on Netflix-like series and films, we first created a database named Analysis_of_Netflix3. There are five main tables in the database: shows, directors, casts, nations, and genres. Every table is made to provide particular data on the many facets of a program or film. For instance, the names of the directors are stored in the director's database, the names of the cast members are stored in the casts table, and the countries table lists the nations in which the episodes were broadcast. The shows are categorised by genre, such as action, drama, or thriller, under the genres table. The shows table, which is the main table, has comprehensive information about every film or television program, including title, director, cast, country, year of release, rating, length, and genre. Two junction tables, show_casts and show_genres, further improve this structure by enabling many-to-many links between the programs and their corresponding cast members and genres. This makes it possible to associate each program with a number of actors and genres, enabling intricate data linkages and enquiries.

To add data to the database, data is added to each table. For example, the casts table has actors like "John," "Jack," and "Rajiv," while the directors table lists directors like "Kirsten," "Brook," and "Vishal." The nations table lists the various nations where the programs were created or were well-liked, such as the US, India, and Japan. A number of categories, including "Documentaries," "Drama," "Mystery," and "Action," are included in the genres table. Each fictitious film and television program in the shows table is linked to a director, cast member, and genre. Each program is ranked according to its quality and spans a variety of release years, genres (movie or TV show), and nations.

A number of SQL queries are run once the data has been entered into the tables in order to retrieve insightful information. By filtering and classifying the data according to predetermined standards, these queries aid in data analysis. The top-rated programs can be found, for instance, by using queries to locate all shows having a rating of 4.5 or above. Additional searches allow for more detailed research by filtering shows by director and kind (movie or TV show). In order to guarantee that only shows uploaded after a specific date are chosen, the script additionally contains queries that examine the programs' release dates. The database's oldest and newest releases are found using queries, which are also used to establish the average rating for shows in particular categories, including "Action."

I. Introduction

The Analysis_of_Netflix3 database was created to hold and examine data on films and television series that are comparable to what is offered on Netflix. The goal of this system is to offer an organised method of handling information regarding shows, directors, actors, genres, and nations. Making it simpler to examine the connections between many elements of a program, including its cast, genre, place of origin, and release information, is the aim. It is feasible to do in-depth searches and studies on a variety of entertainment industry trends by arranging this data into distinct tables.

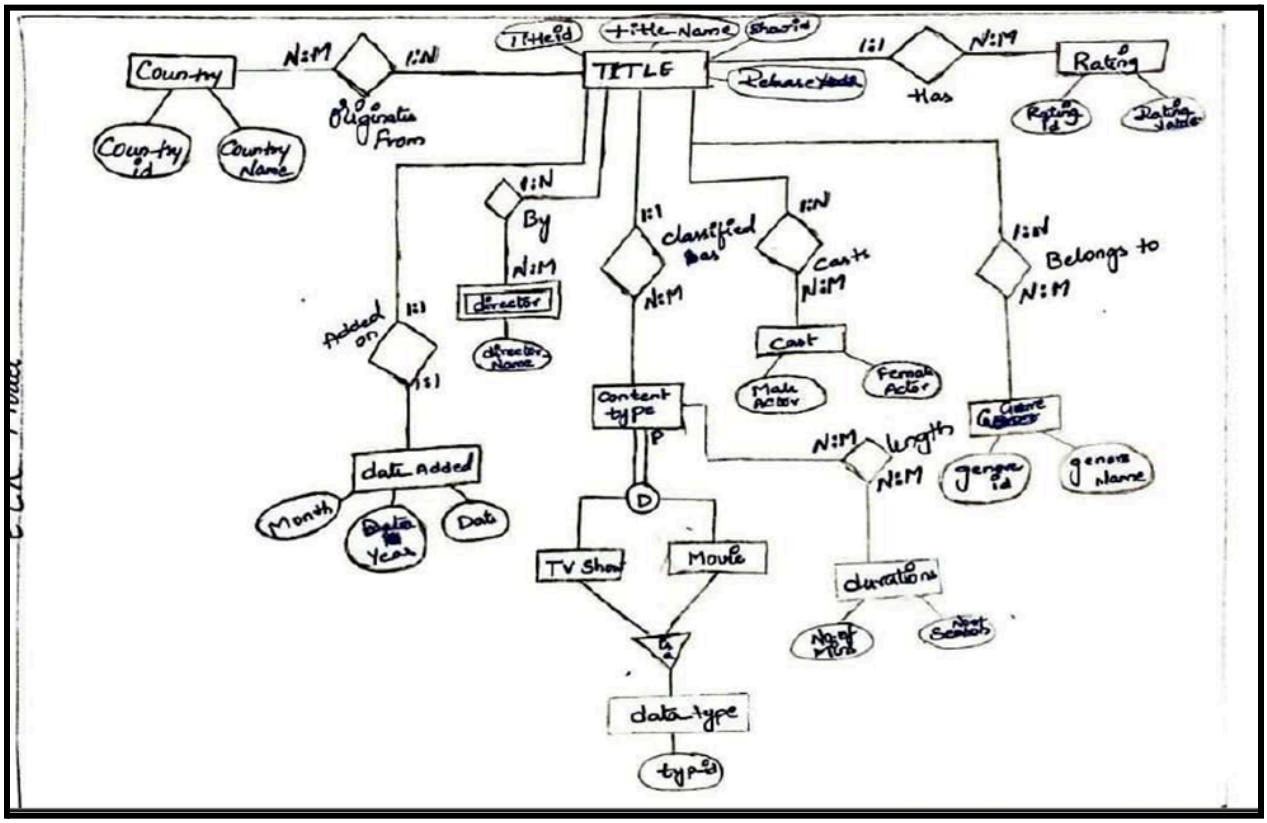
There are multiple core tables in the database, each with a distinct purpose. While the casts table keeps track of the actors and actresses who are part of the productions, the directors table contains information about the show directors. The nations where the shows were produced or broadcast are listed in the countries table. Furthermore, each show is categorised into one or more genres, such as drama, action, or thriller, using the genres table. The title, director, cast, year of release, rating, and length of each film or television program are all listed in detail in the central table. Multiple cast members and genres can be connected to each show using two junction tables, show_casts and show_genres, to increase flexibility.

This database's design is to facilitate the analysis of patterns in films and television series and the retrieval of insightful information. Users can query the database for specific features, such the most popular genres, top-rated shows, or trends in the kinds of shows produced in different nations, by connecting the pertinent data from the many tables. This database can be used to find important information that helps guide choices in content development, marketing, and analysis in the entertainment industry, whether it is determining a certain cast member's engagement in various genres or comprehending the worldwide distribution of shows.

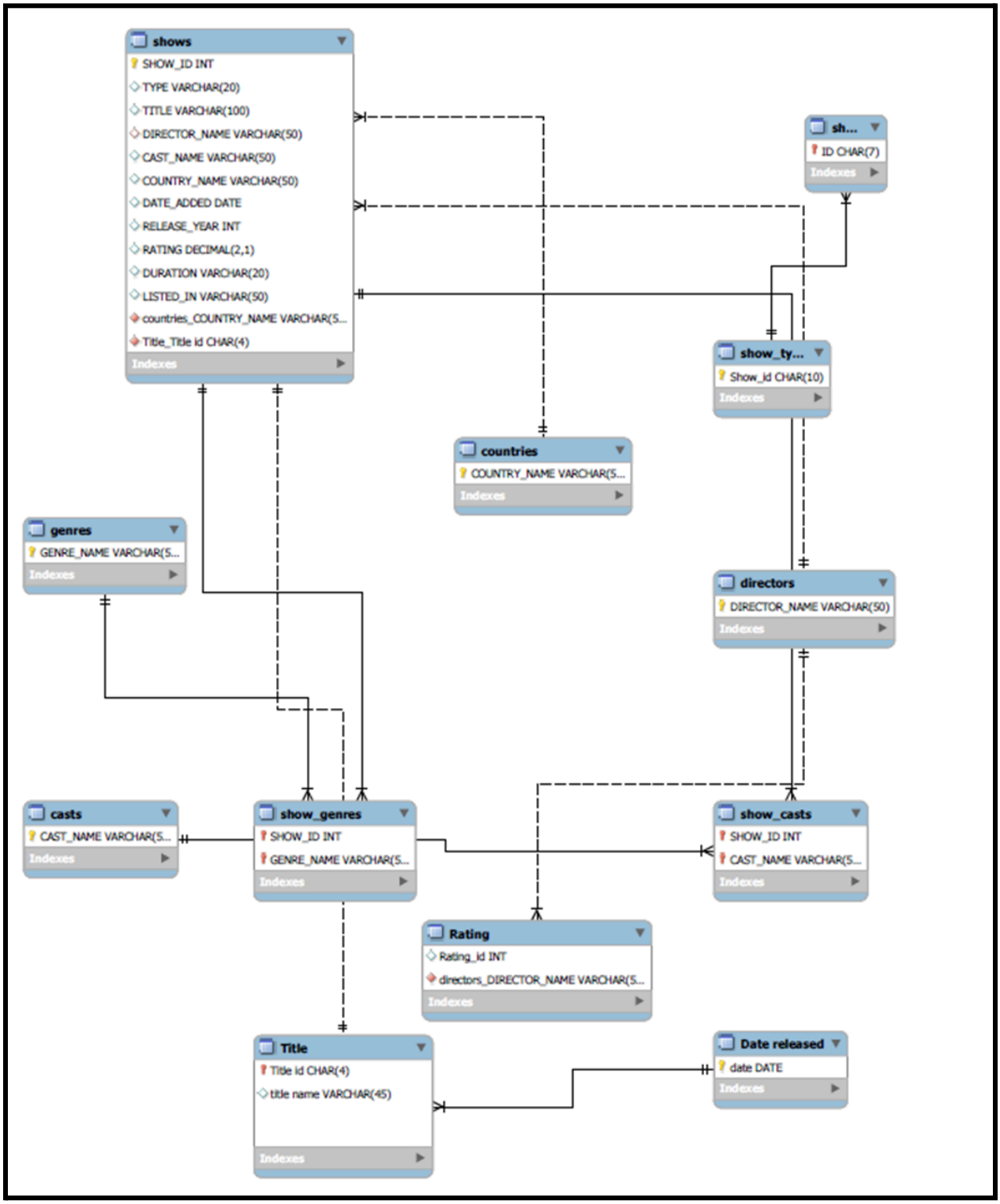
In addition to efficiently organising data, this database gives users the ability to conduct intricate studies and get answers to a variety of queries. For instance, viewers can search for films helmed by particular filmmakers, find highly rated television series, or uncover the average rating of shows in a given category. Additionally, the database facilitates the tracking of trends over time, including determining the earliest and latest years of release or assessing the popularity of genres in various nations. The database's use of relational and junction tables removes redundancy and guarantees data consistency, making it a powerful tool for researching trends and relationships in the entertainment sector. Businesses and researchers can use this method to create data-driven decisions that suit market trends and audience preferences.

II. Conceptual Data Modelling

1. EER Diagram



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

Primary Key- <u>Underlined</u> **Foreign Key- Italicized**

TITLE(TitleID, TitleName, ContentType, DateAdded, Description)

- TitleID: INT, PRIMARY KEY, Auto Increment
- TitleName: VARCHAR, NOT NULL
- ContentType: ENUM('Movie', 'TV Show'), NOT NULL
- DateAdded: DATE, NOT NULL
- Description: TEXT, Nullable

CONTENT_TYPE(ContentTypeID, ContentTypeName)

- ContentTypeID: INT, PRIMARY KEY, Auto Increment
- ContentTypeName: ENUM('Movie', 'TV Show'), NOT NULL

RATING(RatingID, UserID, TitleID, RatingValue, Date)

- RatingID: INT, PRIMARY KEY, Auto Increment
- UserID: INT, FOREIGN KEY refers to User(UserID), NOT NULL
- TitleID: INT, FOREIGN KEY refers to Title(TitleID), NOT NULL
- RatingValue: DECIMAL, NOT NULL
- Date: DATE, NOT NULL

USER(UserID, UserName, UserLocation)

- UserID: INT, PRIMARY KEY, Auto Increment
- UserName: VARCHAR, NOT NULL
- UserLocation: VARCHAR, Nullable

DIRECTOR(DirectorID, DirectorName)

- DirectorID: INT, PRIMARY KEY, Auto Increment
- DirectorName: VARCHAR, NOT NULL

TV_SHOW(ShowID, Seasons)

- ShowID: INT, PRIMARY KEY, FOREIGN KEY refers to Title(TitleID), NOT NULL
- Seasons: INT, NOT NULL, DEFAULT 1

MOVIE(MovieID, Duration)

- MovieID: INT, PRIMARY KEY, FOREIGN KEY refers to Title(TitleID), NOT NULL
- Duration: INT, NOT NULL (In minutes)

CAST(CastID, ActorName, Gender)

- CastID: INT, PRIMARY KEY, Auto Increment
- ActorName: VARCHAR, NOT NULL
- Gender: ENUM('Male', 'Female'), NOT NULL

GENRE(GenreID, GenreName)

- GenreID: INT, PRIMARY KEY, Auto Increment
- GenreName: VARCHAR, NOT NULL

TITLE_GENRE(TitleID, GenreID)

- TitleID: INT, PRIMARY KEY, FOREIGN KEY refers to Title(TitleID), NOT NULL
- GenreID: INT, PRIMARY KEY, FOREIGN KEY refers to Genre(GenreID), NOT NULL

TITLE_CAST(TitleID, CastID)

- TitleID: INT, PRIMARY KEY, FOREIGN KEY refers to Title(TitleID), NOT NULL
- CastID: INT, PRIMARY KEY, FOREIGN KEY refers to Cast(CastID), NOT NULL

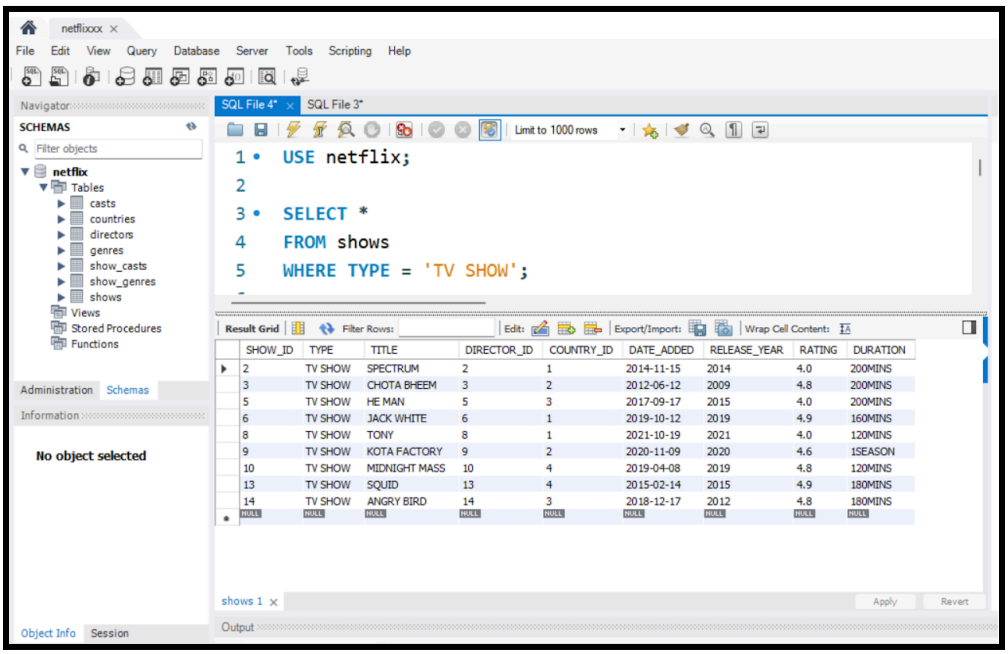
TITLE_DIRECTOR(TitleID, DirectorID)

- TitleID: INT, PRIMARY KEY, FOREIGN KEY refers to Title(TitleID), NOT NULL
- DirectorID: INT, PRIMARY KEY, FOREIGN KEY refers to Director(DirectorID), NOT NULL

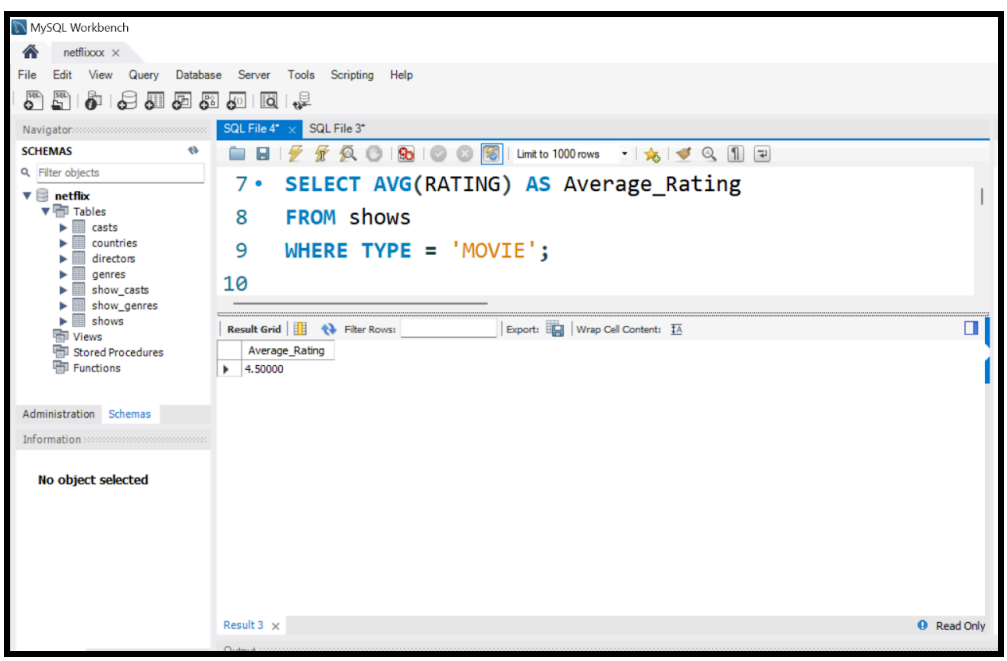
IV. Implementation of Relation Model via MYSQL and NOSQL

MYSQL Implementation:

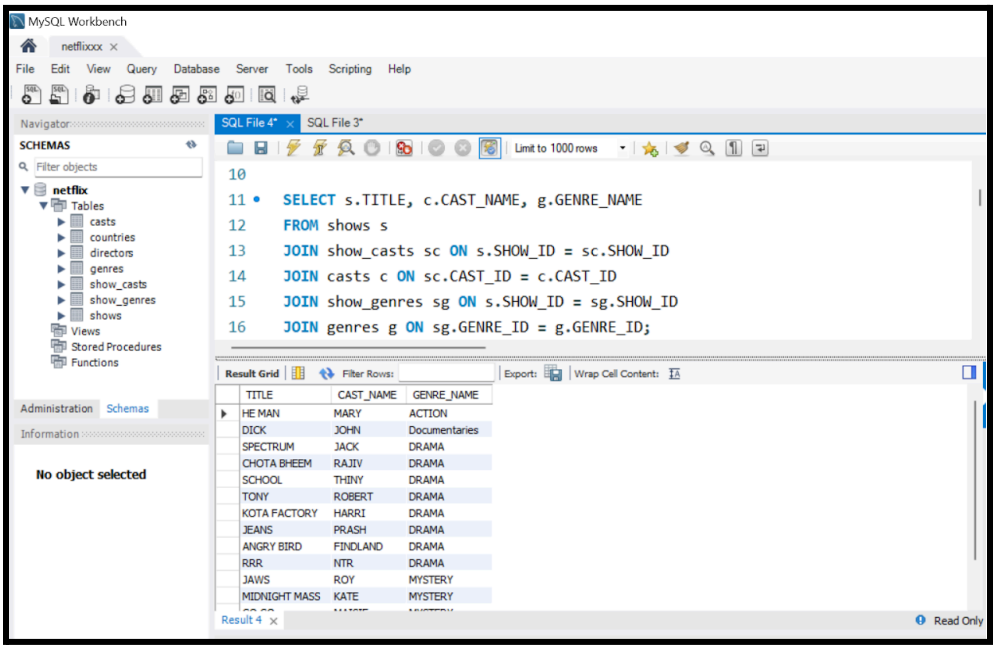
1. **Simple Query: Select all TV shows from the shows table.**

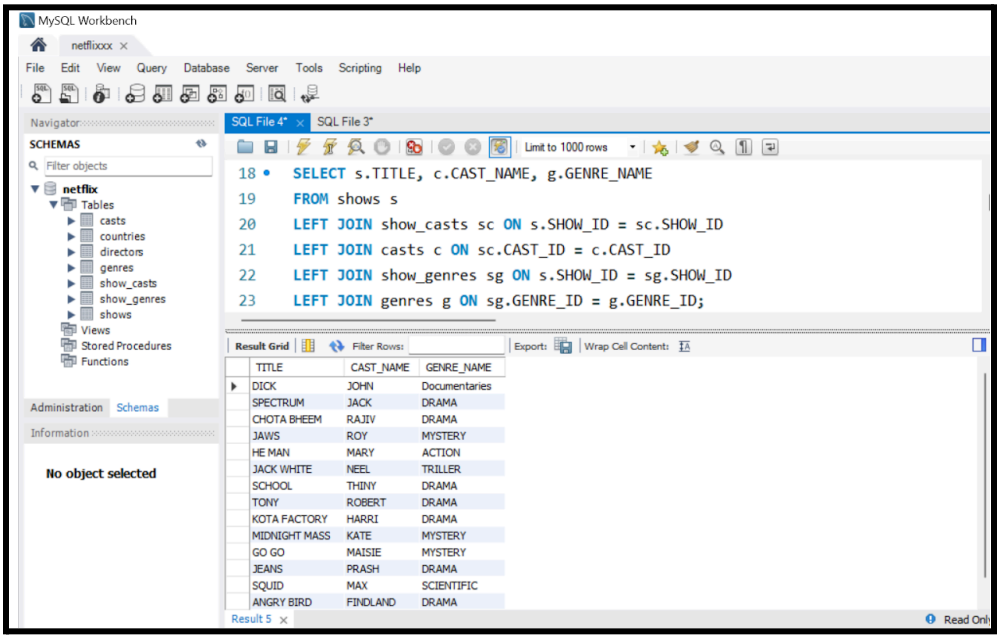


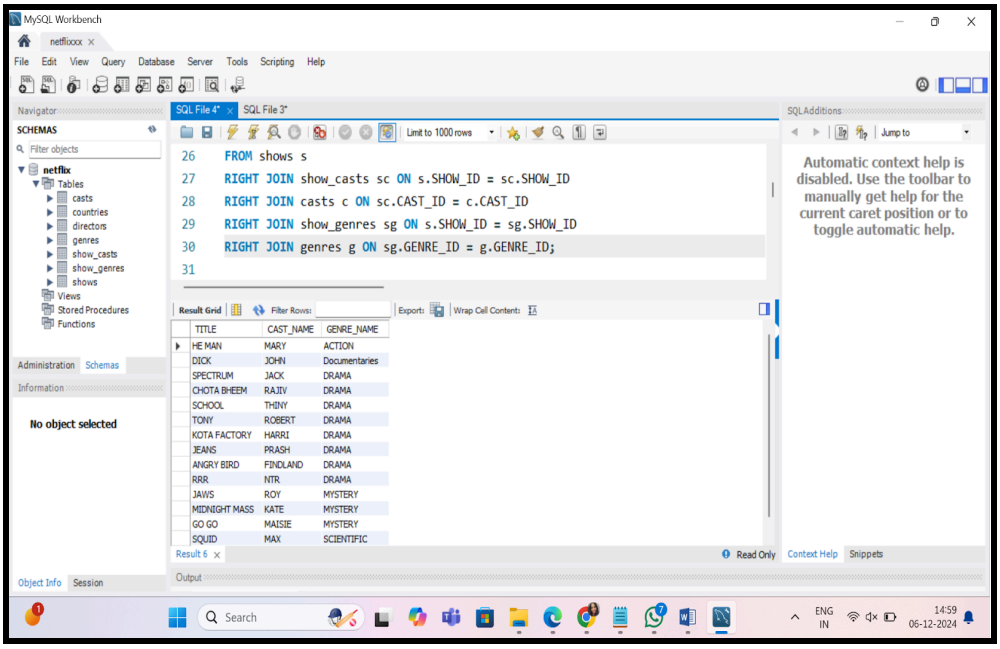
2. **Aggregate Query: Select the average rating of movies from the shows table.**



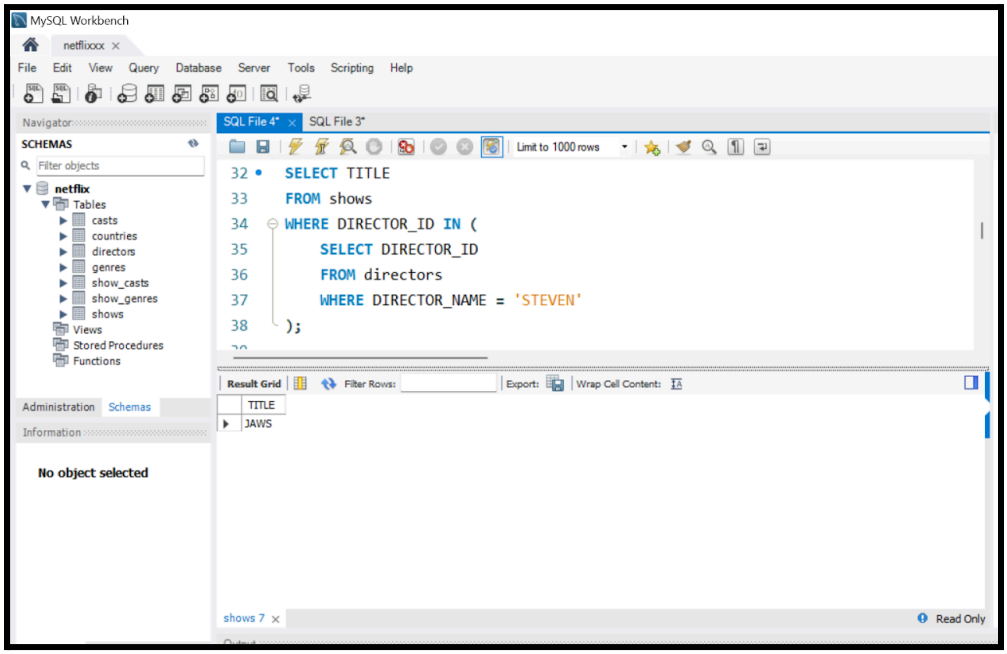
3. **Inner join/ Left Join/ Right Join:**
   A. **Select the title, cast name, and genre name for all shows. (Inner Join)**
   B. **Select the title, cast name, and genre name for all shows, including those without a cast or genre. (Left Join)**
   C. **Select the title, cast name, and genre name for all casts and genres, including those not associated with a show. (Right Join)**



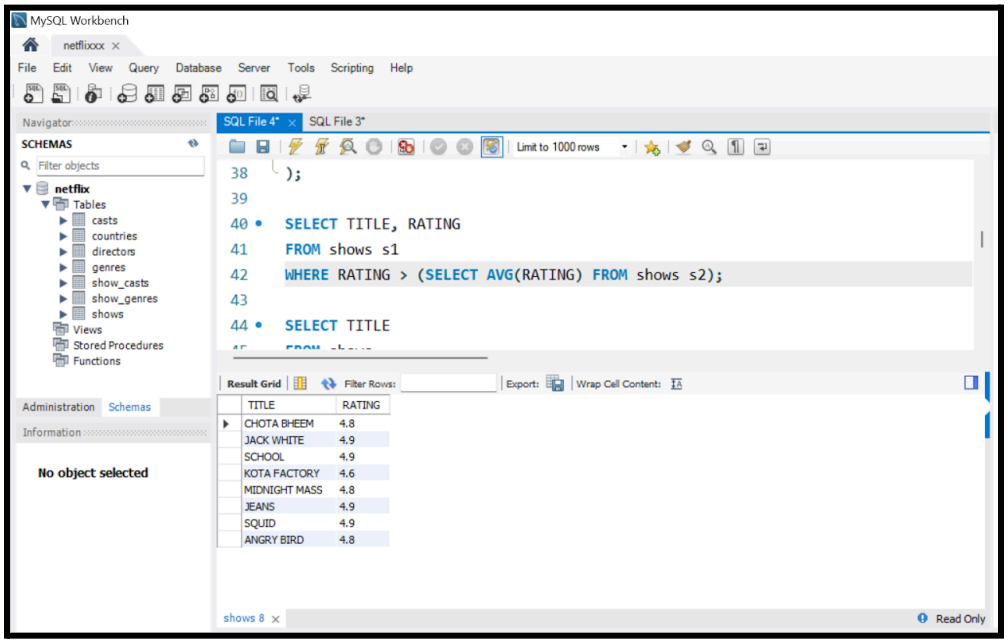




4. **Nested query: Select the titles of shows directed by 'STEVEN'.**



5. **Correlated Query: Select the titles and ratings of shows with a rating higher than the average rating.**

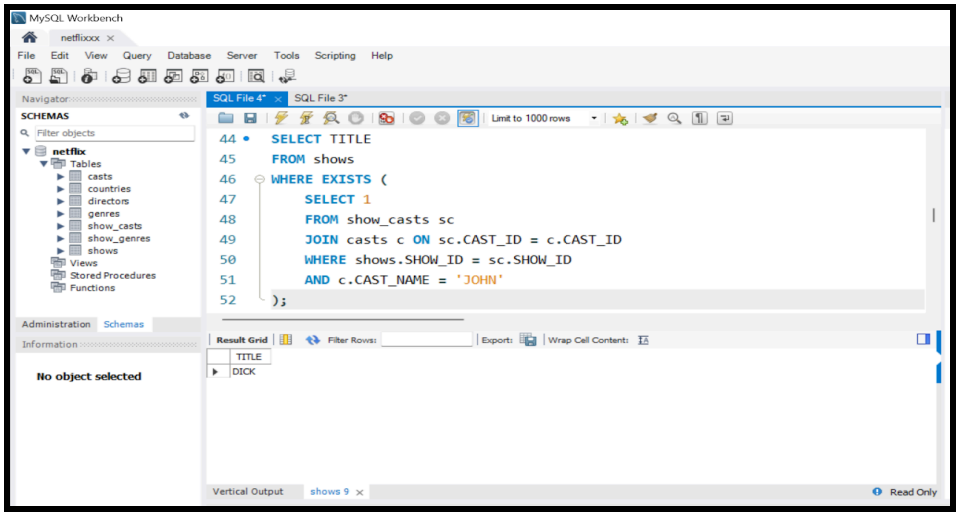


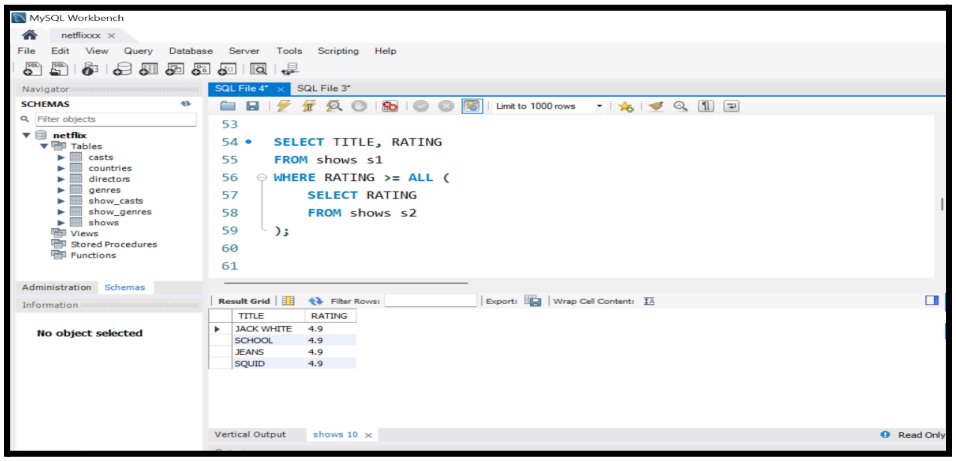
6. **>=ALL/>ANY/Exists/Not Exists:**
   A. **Select the titles of shows that have 'JOHN' as a cast member.**

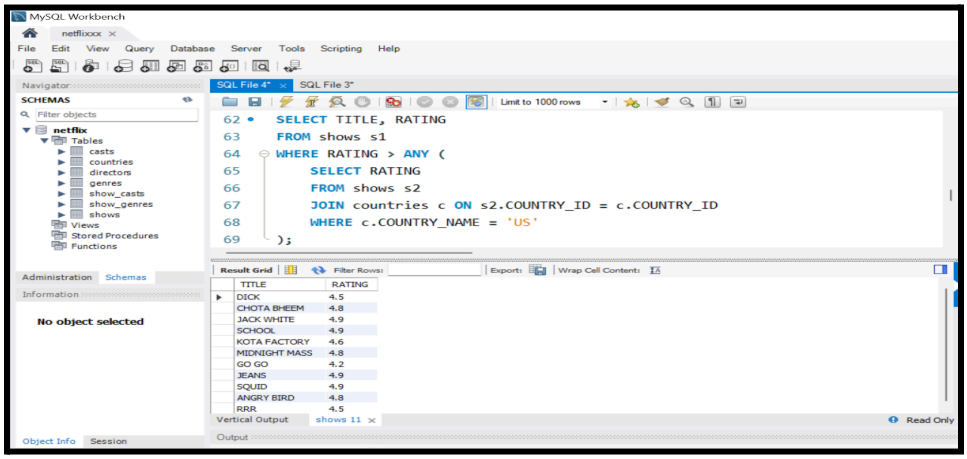
   B. **Select the titles and ratings of shows with a rating higher than or equal to the highest rating.**

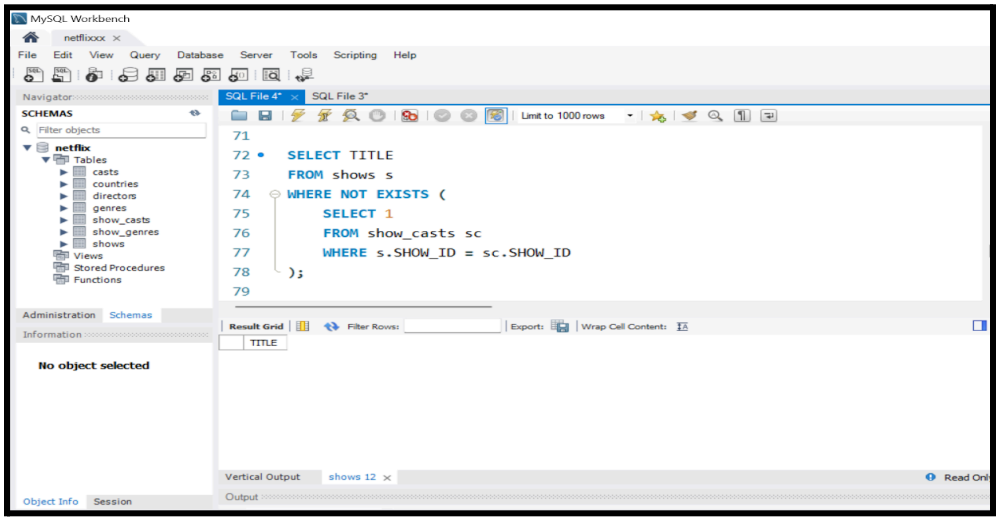
   C. **Select the titles and ratings of shows with a rating higher than any show in the 'US'.**

   D. **Select the titles of shows that do not have any cast members.**

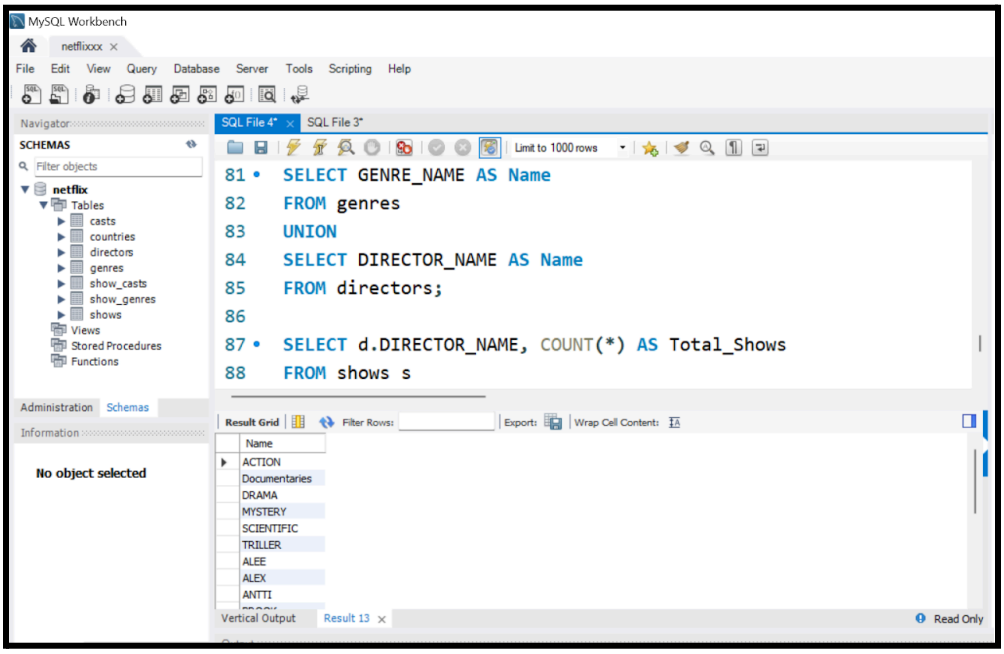

MySQL Workbench
netflixxx
File Edit View Query Database Server Tools Scripting Help
SQL File 4* SQL File 3*
Limit to 1000 rows
SELECT TITLE
FROM shows
WHERE EXISTS (
SELECT 1
FROM show_casts sc
JOIN casts c ON sc.CAST_ID = c.CAST_ID
WHERE shows.SHOW_ID = sc.SHOW_ID
AND c.CAST_NAME = 'JOHN'
);
Result Grid
TITLE
DICK
No object selected
Vertical Output
shows 9
Read Only


MySQL Workbench
netflixxx
File Edit View Query Database Server Tools Scripting Help
SQL File 4* SQL File 3*
Limit to 1000 rows
SELECT TITLE, RATING
FROM shows s1
WHERE RATING >= ALL (
SELECT RATING
FROM shows s2
);
Result Grid
TITLE RATING
JACK WHITE 4.9
SCHOOL 4.9
JEANS 4.9
SQUID 4.9
No object selected
Vertical Output
shows 10
Read Only

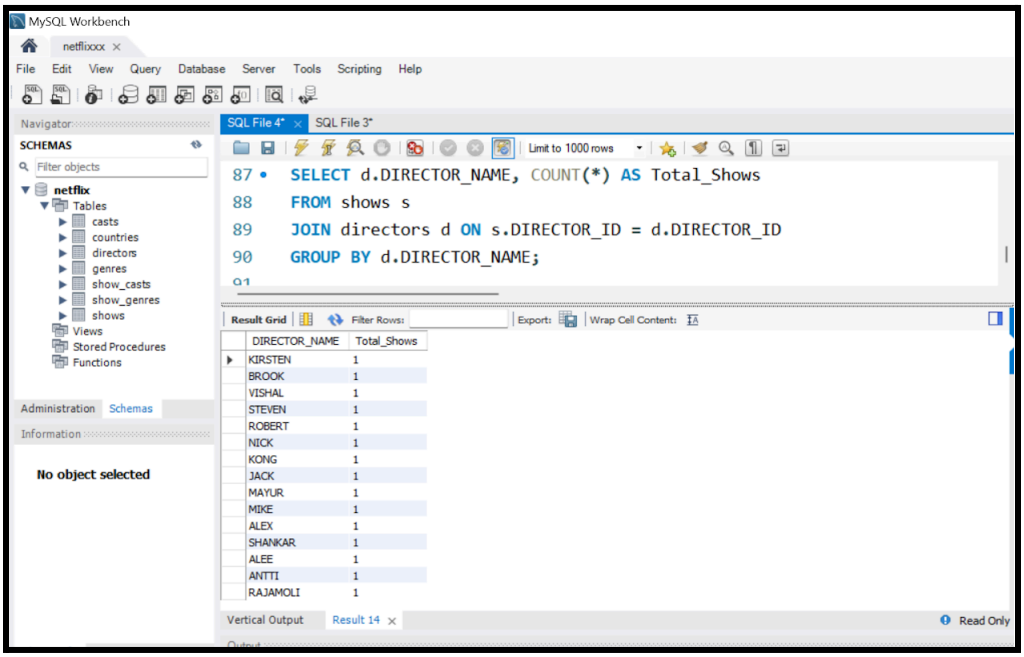

MySQL Workbench
netflixxx
File Edit View Query Database Server Tools Scripting Help
SQL File 4* SQL File 3*
Limit to 1000 rows
SELECT TITLE, RATING
FROM shows s1
WHERE RATING > ANY (
SELECT RATING
FROM shows s2
JOIN countries c ON s2.COUNTRY_ID = c.COUNTRY_ID
WHERE c.COUNTRY_NAME = 'US'
);
Result Grid
TITLE RATING
DICK 4.5
CHOTA BHEEM 4.8
JACK WHITE 4.9
SCHOOL 4.9
KOTA FACTORY 4.6
MIDNIGHT MASS 4.8
GO GO 4.2
JEANS 4.9
SQUID 4.9
ANGRY BIRD 4.8
RRR 4.5
No object selected
Vertical Output
shows 11
Read Only

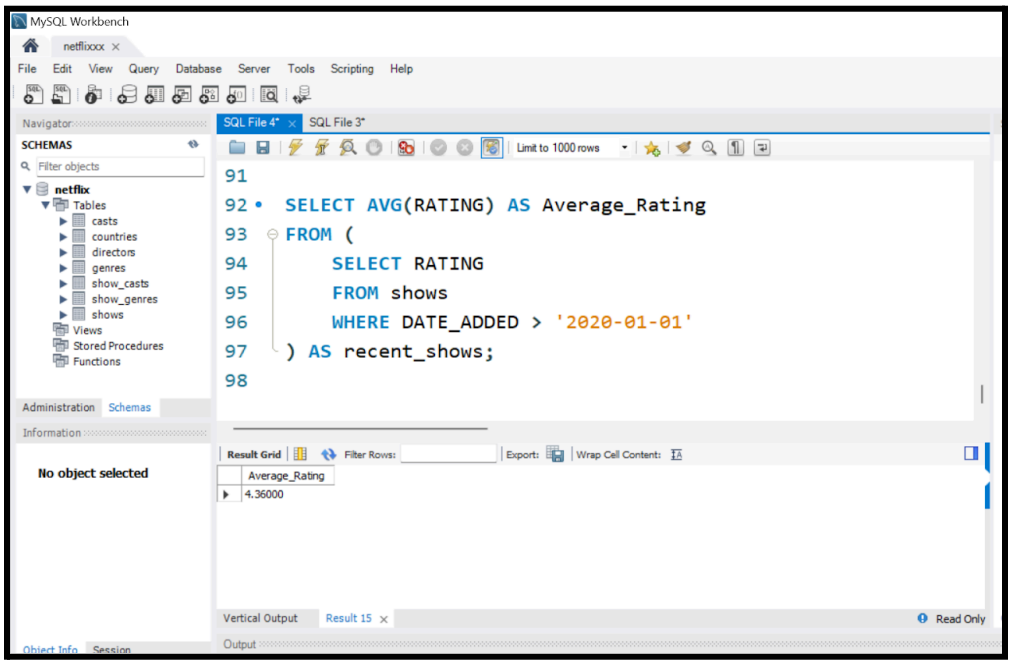


7. **Set operations (Union): Select all genre names and director names as a combined list.**



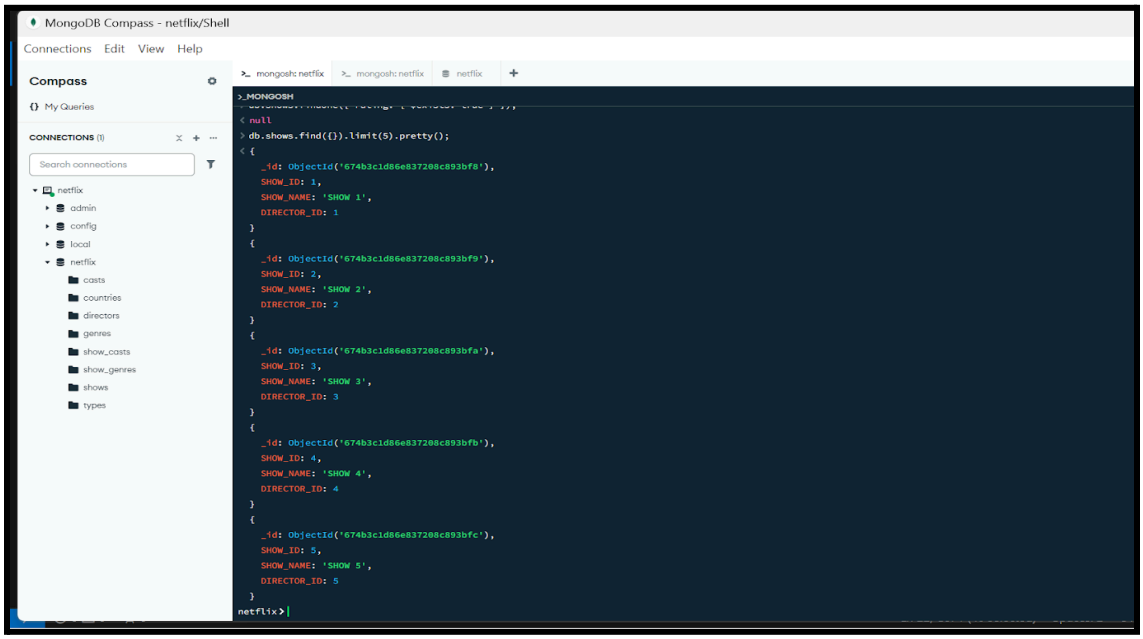
8. **Subqueries in Select and From:**
   A. **Select the director's name and the total number of shows they have directed.**
   B. **Select the average rating of shows added after January 1, 2020.**



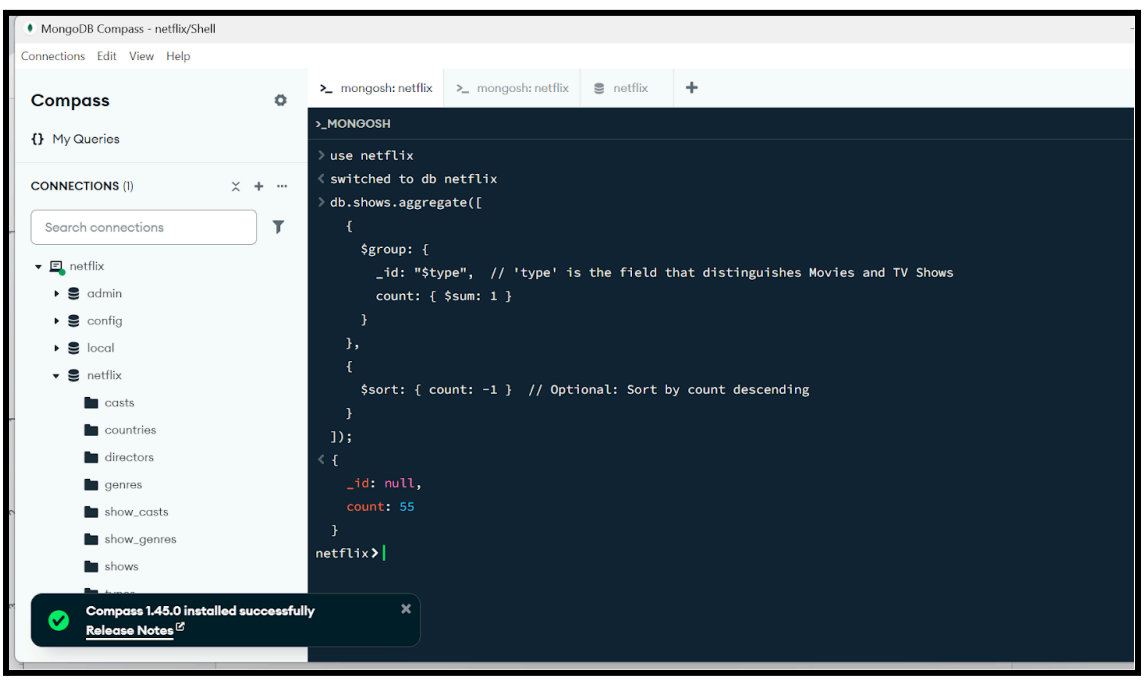


NOSQL Implementation:

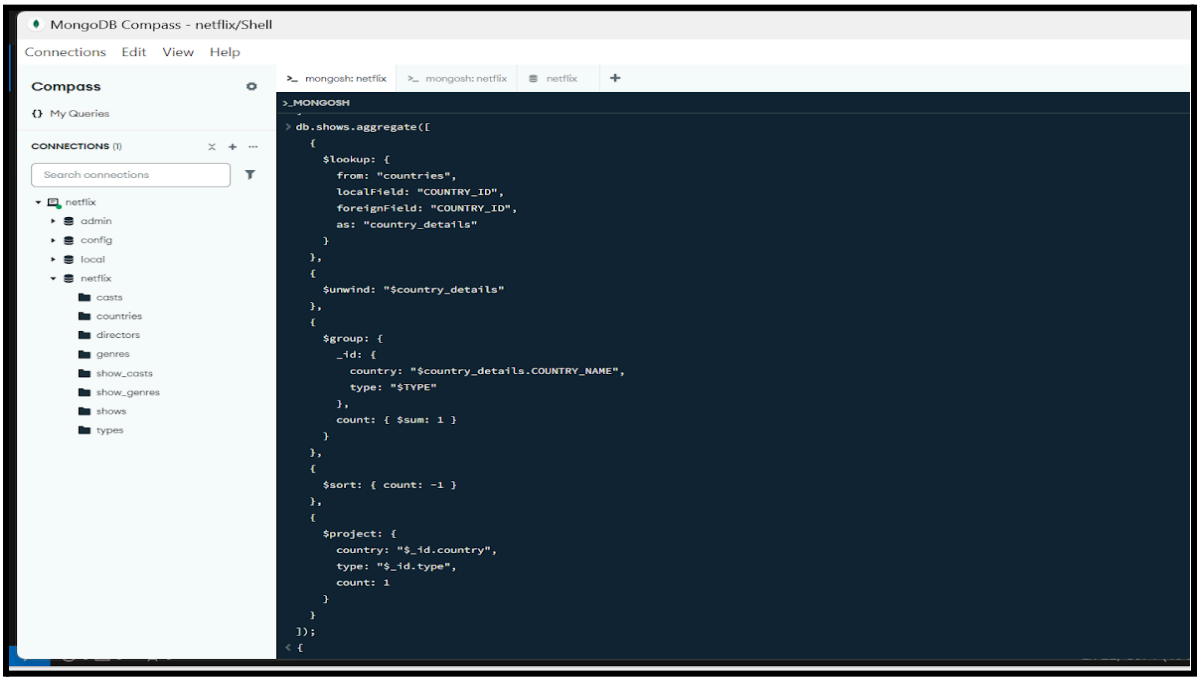
Query 1 (Simple): What are the first 5 shows in the database?

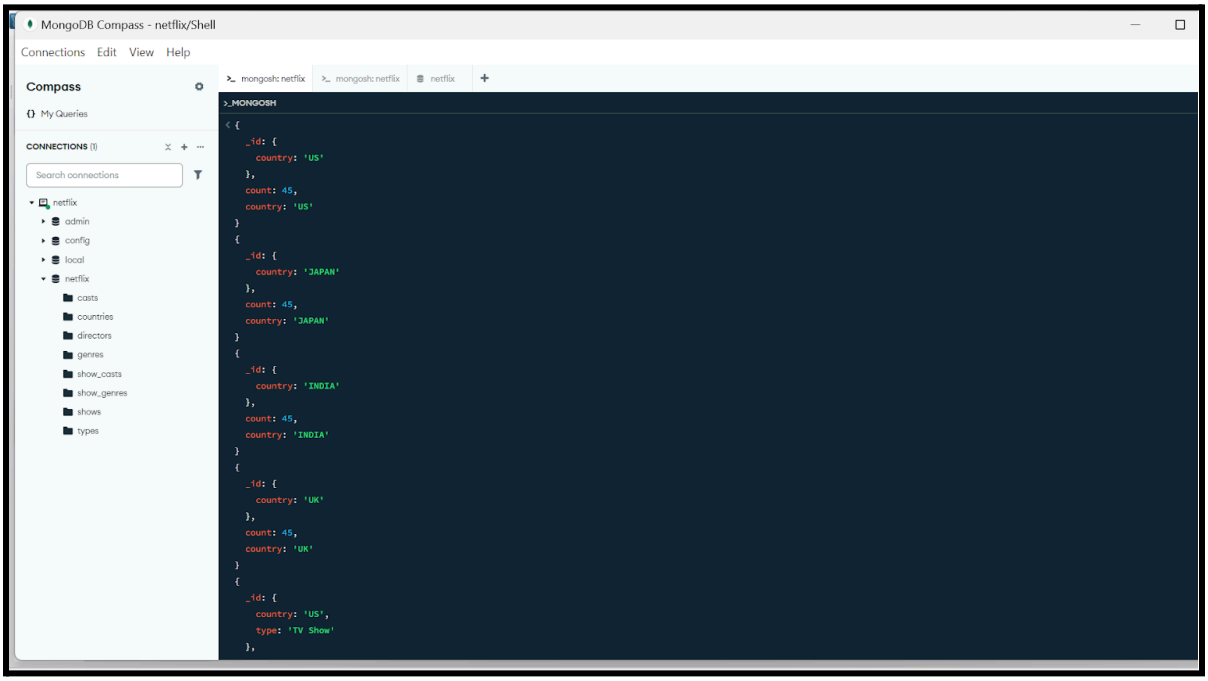


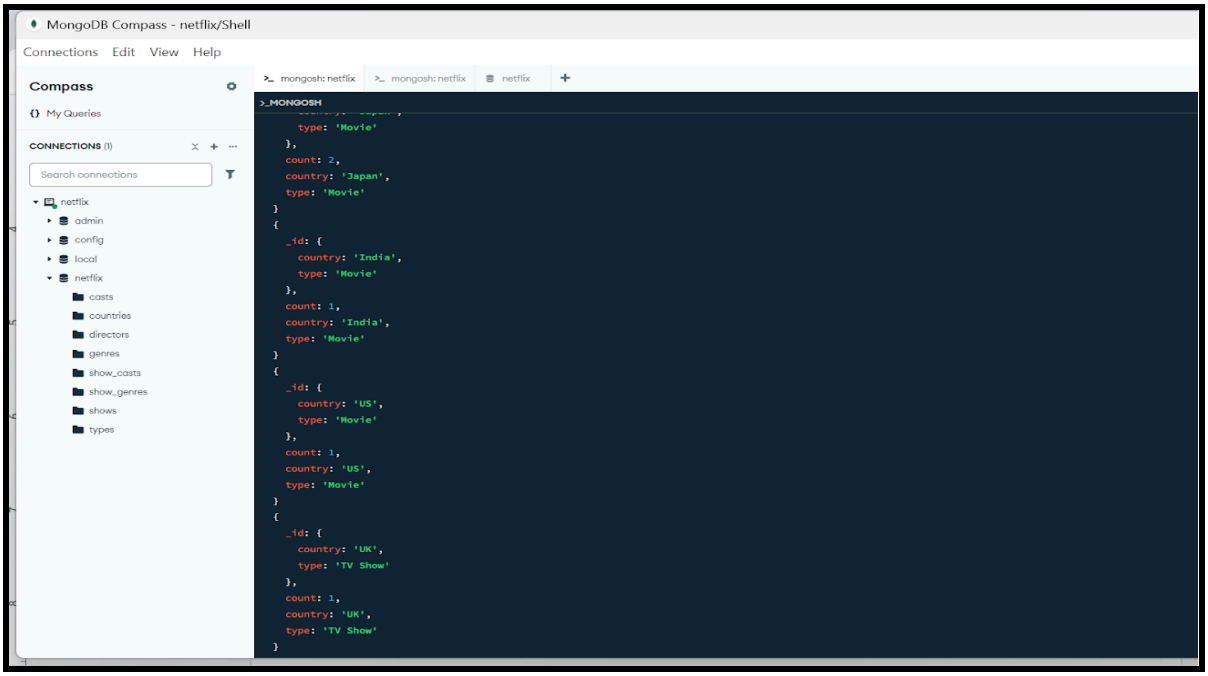
Query 2 (Aggregate): Count the number of Movies vs TV Shows?



Query 3 (Complex): How many movies and TV shows are there for each country, grouped by type (MOVIE or TV SHOW)?



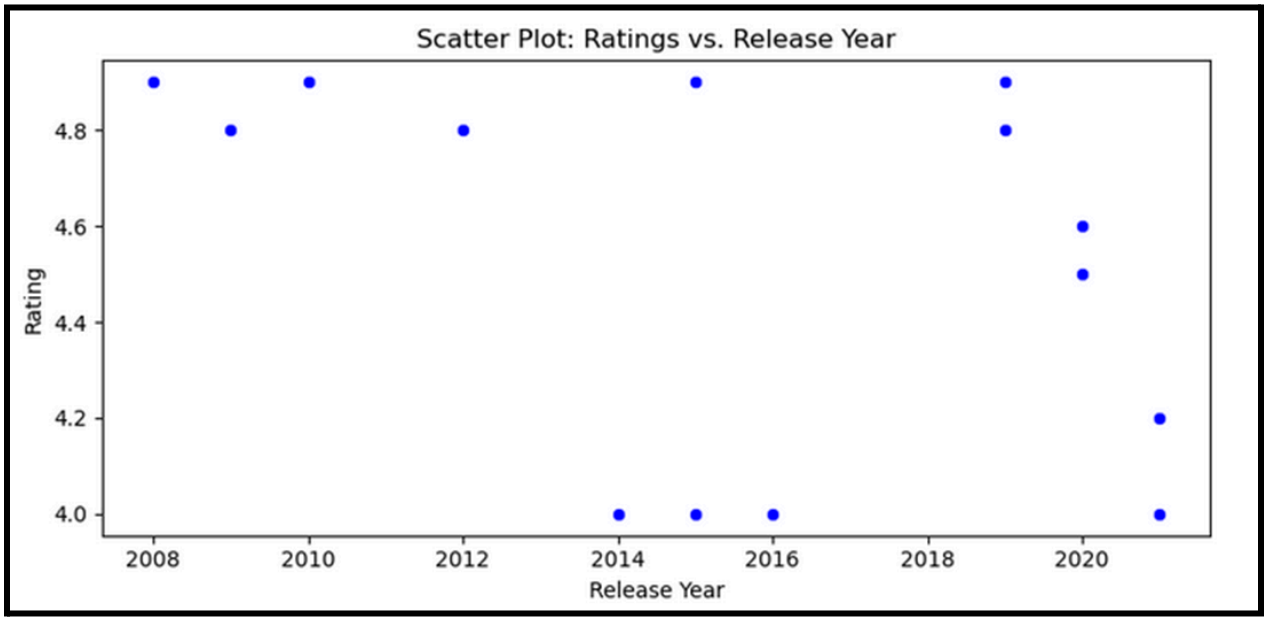




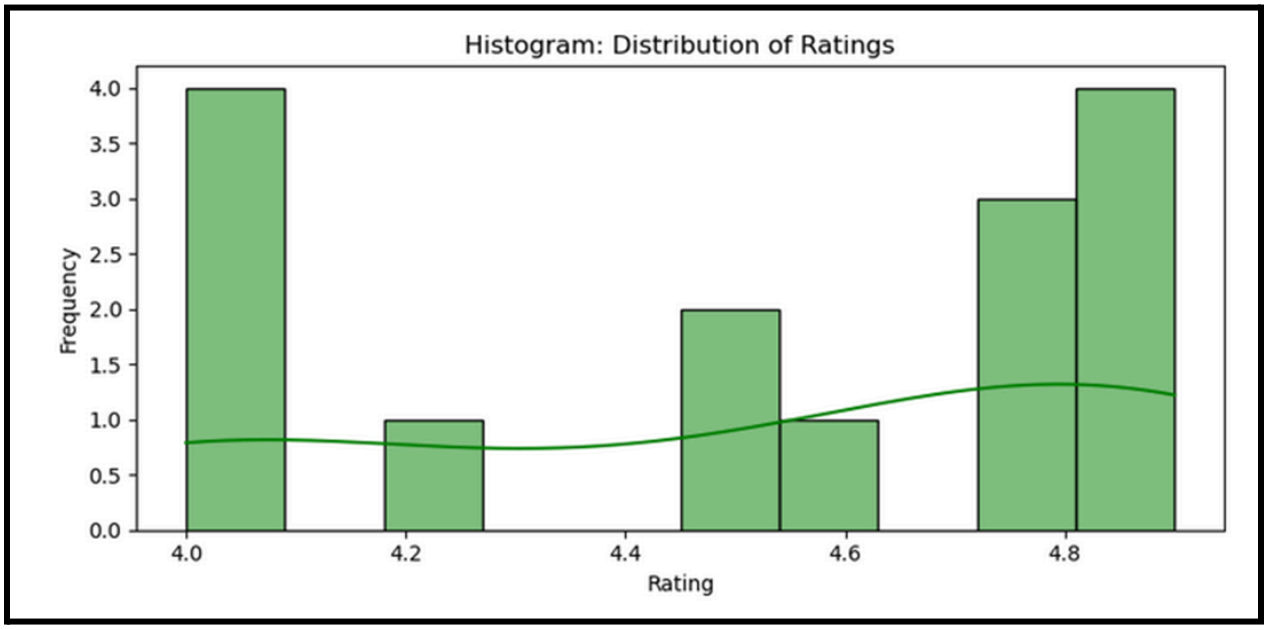
V. Database Access via Python

The database is accessed using Python and visualization of analyzed data is shown below. The connection of MySQL to Python is done using mysql.connector, followed by cursor.execute to run and fetch all from query, followed by converting the list into a dataframe using pandas library and using matplotlib to plot the graphs for the analytics.

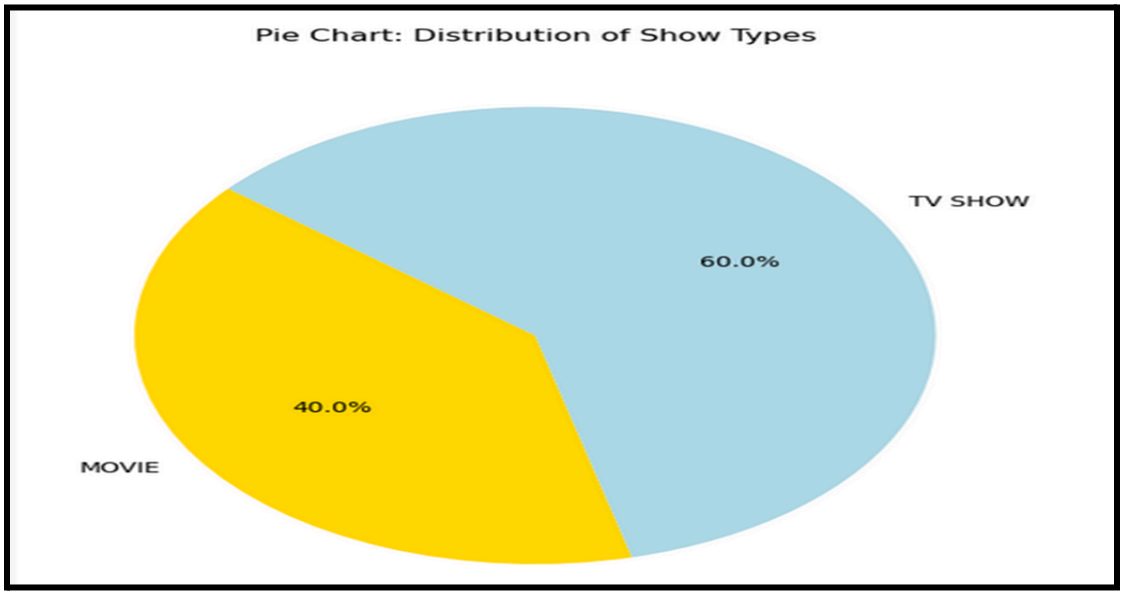
GRAPH 1: Scatter Plot: RELEASE_YEAR vs. RATING



GRAPH 2: Histogram: Distribution of Ratings



GRAPH 3: Pie Chart: Distribution of Type (MOVIE vs. TV SHOW)



VII. Summary and Recommendation

The main goal of this SQL project is to build a strong relational database called Analysis_of_Netflix3 that will be used to analyse Netflix's vast collection of films and TV series. To guarantee effective data storage and retrieval, the database complies with normalisation guidelines. Directors, casts, countries, genres, and series are all included in the basic tables. Foreign keys are used to create associations between tables, improving referential integrity, whereas primary keys are used to guarantee uniqueness for each table. The show's type (movie or TV show), title, director, cast, country of origin, date added, release year, rating, runtime, and genre are among the crucial details that are stored in the shows table. To manage the many-to-many relationships between shows, casts, and genres, two more junction tables are added: show_casts and show_genres. This allows for effective querying across these relationships.

A few enhancements can be made to this database to further improve its usability and maintainability. First, data consistency can be ensured by preventing partial data entries by applying constraints like NOT NULL on crucial fields (such as TITLE, TYPE, and RELEASE_YEAR). Duplicate entries can be further avoided by applying UNIQUE constraints to fields like TITLE and SHOW_ID. Developing indexes on commonly searched columns, such TYPE, RELEASE_YEAR, and RATING, can greatly enhance query efficiency, particularly when working with big datasets. Deeper understanding of user preferences and behaviour would also be possible by adding columns for user interactions, such as ratings, reviews, and viewing history, to the database.