Milestone 2 - Group 6

Movie and TV Series Recommendation System

Devin Lim
Oliver Xing
Tejaswi Neelapu
Maelice Yamdjieu
Shiva Skandhan

Queries:

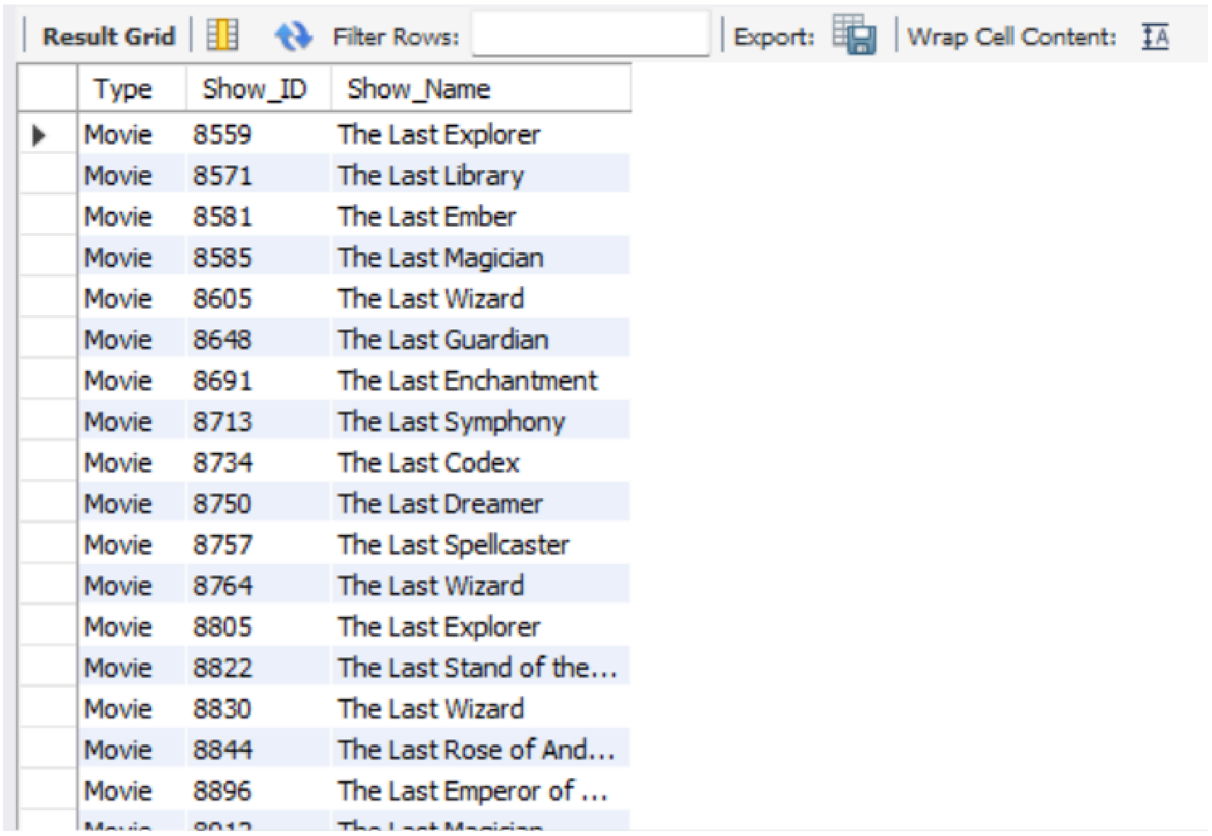
1. Query to find a Movie and a TV Series containing substring “last”

This query retrieves information on both movies and TV series containing the substring “last” in their names. Using union all, it combines the results of the two select statements for movies and TV series.

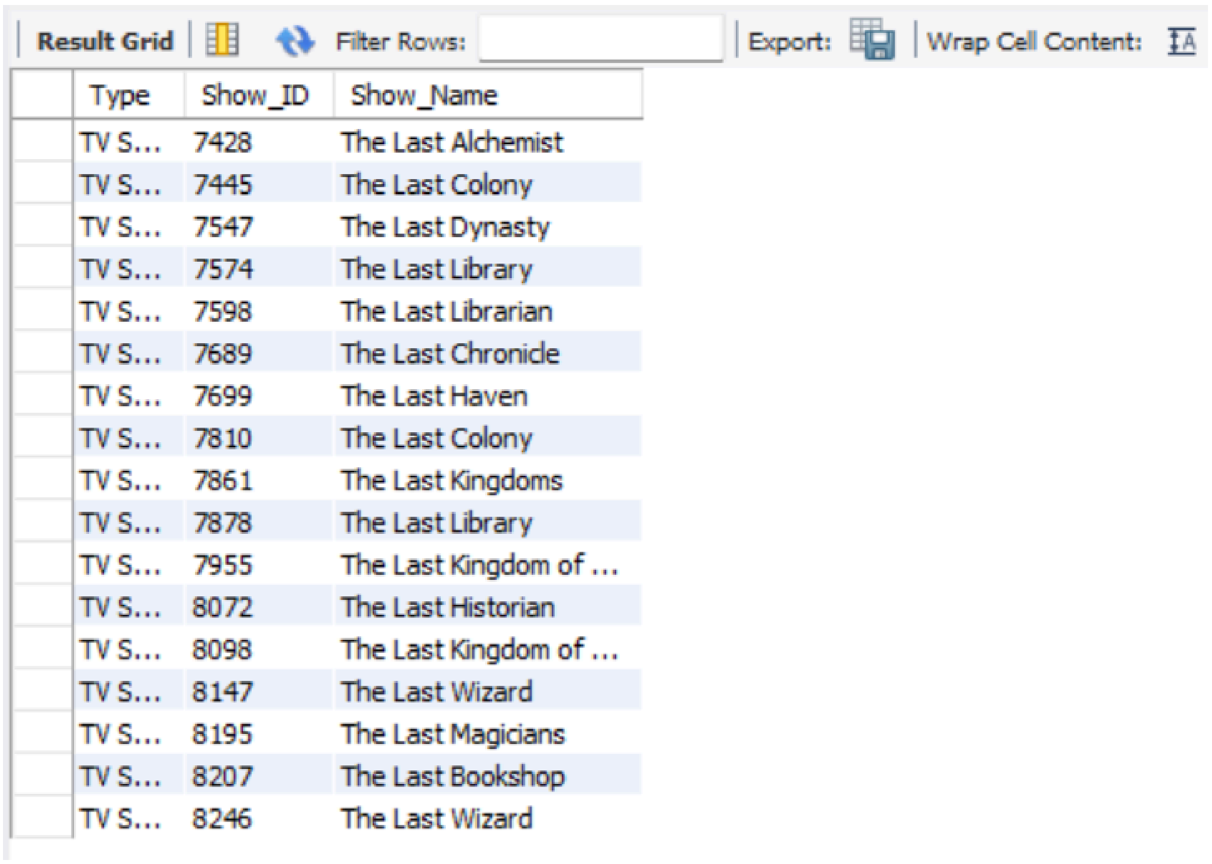
```
SELECT 'Movie' AS Type,
       M.Movie_ID AS Show_ID,
       M.Name AS Show_Name
FROM Movie M
WHERE M.Name LIKE '%last%'

UNION ALL

SELECT 'TV Series' AS Type,
       TV.TV_Series_ID AS Show_ID,
       TV.Name AS Show_Name
FROM TV_Series TV
WHERE TV.Name LIKE '%last%';
```



Type	Show_ID	Show_Name
Movie	8559	The Last Explorer
Movie	8571	The Last Library
Movie	8581	The Last Ember
Movie	8585	The Last Magician
Movie	8605	The Last Wizard
Movie	8648	The Last Guardian
Movie	8691	The Last Enchantment
Movie	8713	The Last Symphony
Movie	8734	The Last Codex
Movie	8750	The Last Dreamer
Movie	8757	The Last Spellcaster
Movie	8764	The Last Wizard
Movie	8805	The Last Explorer
Movie	8822	The Last Stand of the...
Movie	8830	The Last Wizard
Movie	8844	The Last Rose of And...
Movie	8896	The Last Emperor of ...



Type	Show_ID	Show_Name
TV S...	7428	The Last Alchemist
TV S...	7445	The Last Colony
TV S...	7547	The Last Dynasty
TV S...	7574	The Last Library
TV S...	7598	The Last Librarian
TV S...	7689	The Last Chronicle
TV S...	7699	The Last Haven
TV S...	7810	The Last Colony
TV S...	7861	The Last Kingdoms
TV S...	7878	The Last Library
TV S...	7955	The Last Kingdom of ...
TV S...	8072	The Last Historian
TV S...	8098	The Last Kingdom of ...
TV S...	8147	The Last Wizard
TV S...	8195	The Last Magicians
TV S...	8207	The Last Bookshop
TV S...	8246	The Last Wizard

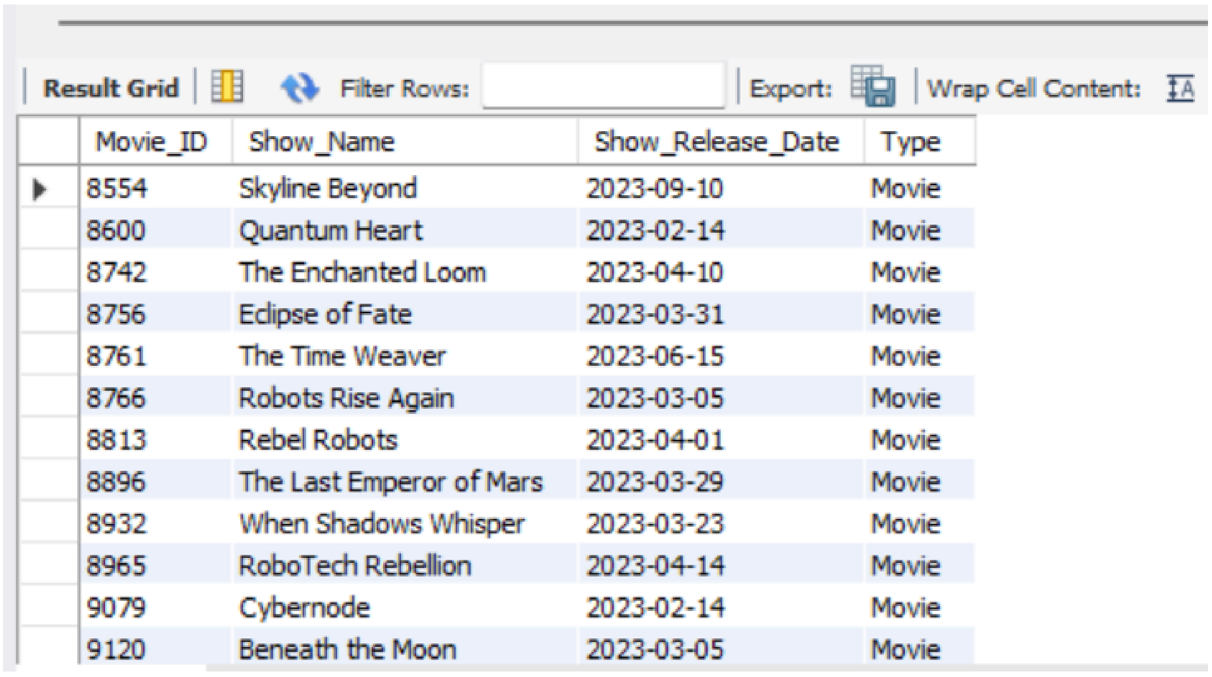
2. Shows released within the past year (-365 days)

This query gives shows, both movies and TV series, released within the past year, specified as the last 365 days. Using union all, it combines the results of the two select statements for movies and TV series.

```sql
SELECT
    Movie_ID,
    Name AS Show_Name,
    Release_Date AS Show_Release_Date,
    'Movie' AS Type
FROM
    Movie
WHERE
    Release_Date BETWEEN DATE_SUB(CURDATE(), INTERVAL 365 DAY) AND
CURDATE()

UNION ALL

SELECT
    TV_Series_ID AS Movie_ID,
    Name AS Show_Name,
    Release_Date AS Show_Release_Date,
    'TV Series' AS Type
FROM
    TV_Series
WHERE
    Release_Date BETWEEN DATE_SUB(CURDATE(), INTERVAL 365 DAY) AND
CURDATE();
```

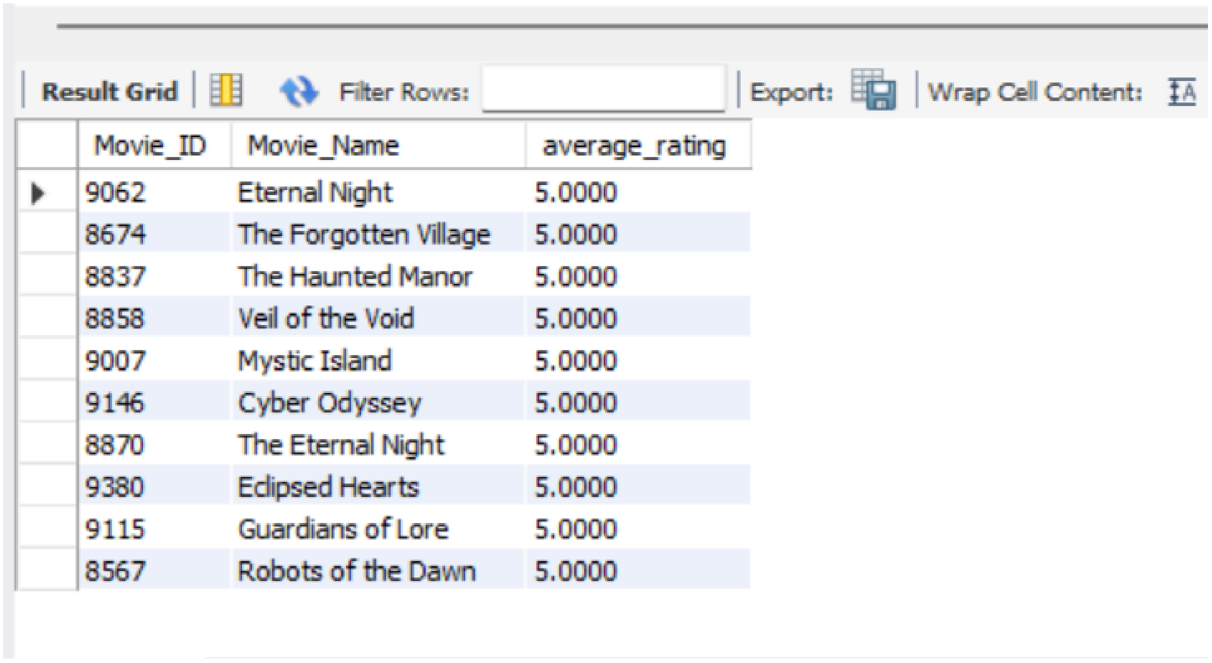

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Movie_ID	Show_Name	Show_Release_Date	Type
8554	Skyline Beyond	2023-09-10	Movie
8600	Quantum Heart	2023-02-14	Movie
8742	The Enchanted Loom	2023-04-10	Movie
8756	Eclipse of Fate	2023-03-31	Movie
8761	The Time Weaver	2023-06-15	Movie
8766	Robots Rise Again	2023-03-05	Movie
8813	Rebel Robots	2023-04-01	Movie
8896	The Last Emperor of Mars	2023-03-29	Movie
8932	When Shadows Whisper	2023-03-23	Movie
8965	RoboTech Rebellion	2023-04-14	Movie
9079	Cybernode	2023-02-14	Movie
9120	Beneath the Moon	2023-03-05	Movie

3. Top rated movies for users between the age of 18 and 30

The query retrieves the top-rated movies among users aged between 18 and 30. It joins various tables including ‘user’, ‘History’, ‘show_table’, and ‘movie’ linking users, their history, and the associated movie details. The query calculates the average ratings by filtering users based on their age range.

```
SELECT
    M.Movie_ID,
    M.Name AS Movie_Name,
    AVG(H.rating) AS average_rating
FROM
    User U
JOIN
    History H ON U.User_ID = H.User_ID
JOIN
    Show_Table ST ON H.Show_ID = ST.Show_ID
JOIN
    Movie M ON ST.Movie_ID = M.Movie_ID
WHERE
    YEAR(CURDATE()) - YEAR(U.birthday) BETWEEN 18 AND 30
GROUP BY
    M.Movie_ID, M.Name
ORDER BY
    average_rating DESC
LIMIT 10;
```

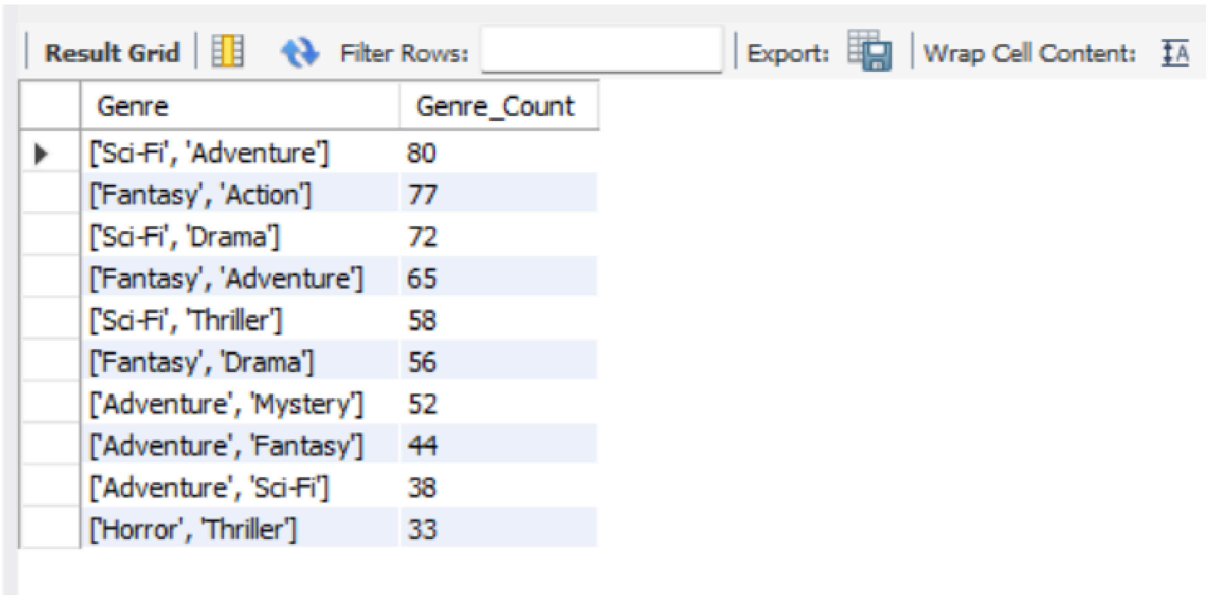

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Movie_ID	Movie_Name	average_rating
9062	Eternal Night	5.0000
8674	The Forgotten Village	5.0000
8837	The Haunted Manor	5.0000
8858	Veil of the Void	5.0000
9007	Mystic Island	5.0000
9146	Cyber Odyssey	5.0000
8870	The Eternal Night	5.0000
9380	Eclipsed Hearts	5.0000
9115	Guardians of Lore	5.0000
8567	Robots of the Dawn	5.0000

4. Top 10 Genres

The query calculates the top 10 genres based on the number of views across all movies. It joins the ‘movie’, ‘show_table’, and ‘history’ tables to associate movies with their genres and user viewing history. By grouping the data by genre and counting the occurrences, the query orders the genre in descending order of their view counts ensuring only the top 10 genres are displayed.

```
SELECT
    M.Genre,
    COUNT(*) AS Genre_Count
FROM
    Movie M
JOIN
    Show_Table S ON M.Movie_ID = S.Movie_ID
JOIN
    History H ON S.Show_ID = H.Show_ID
GROUP BY
    M.Genre
ORDER BY
    Genre_Count DESC
LIMIT 10;
```

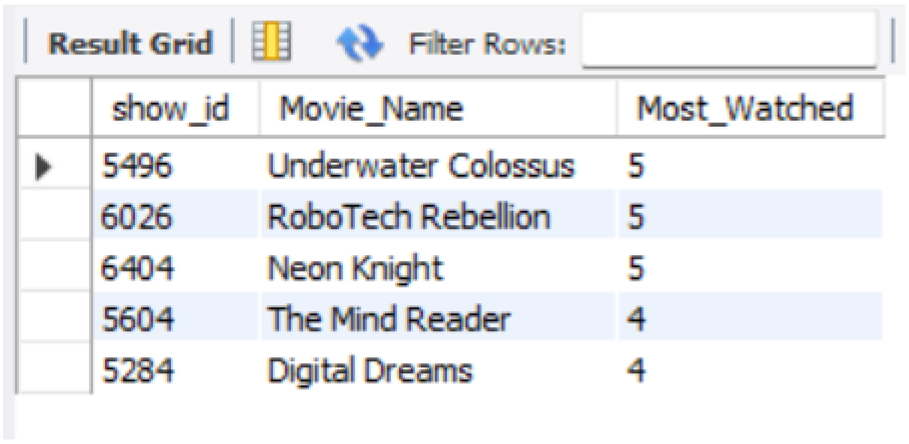


Genre	Genre_Count
['Sci-Fi', 'Adventure']	80
['Fantasy', 'Action']	77
['Sci-Fi', 'Drama']	72
['Fantasy', 'Adventure']	65
['Sci-Fi', 'Thriller']	58
['Fantasy', 'Drama']	56
['Adventure', 'Mystery']	52
['Adventure', 'Fantasy']	44
['Adventure', 'Sci-Fi']	38
['Horror', 'Thriller']	33

5. Most Watched shows between 2020-2024

The query gives the 5 most-watched shows released between 2020 and 2024. It gives this by joining the 'history', 'show_table', and 'movie' tables to associate viewing history with show details and filter shows based on their release dates within the specific year. Grouping the data by 'show id' enables the calculation of the count of views for each show, facilitating the identification of the most watched shows.

```
SELECT S.show_id, M.name AS Movie_Name, COUNT(*) AS Most_Watched
FROM History H
JOIN  Show_Table S ON H.show_id = S.show_id
JOIN Movie M ON S.movie_id = M.movie_id
WHERE YEAR(M.Release_Date) BETWEEN 2020 AND 2024
GROUP BY S.show_id
ORDER BY Most_Watched DESC
LIMIT 5;
```

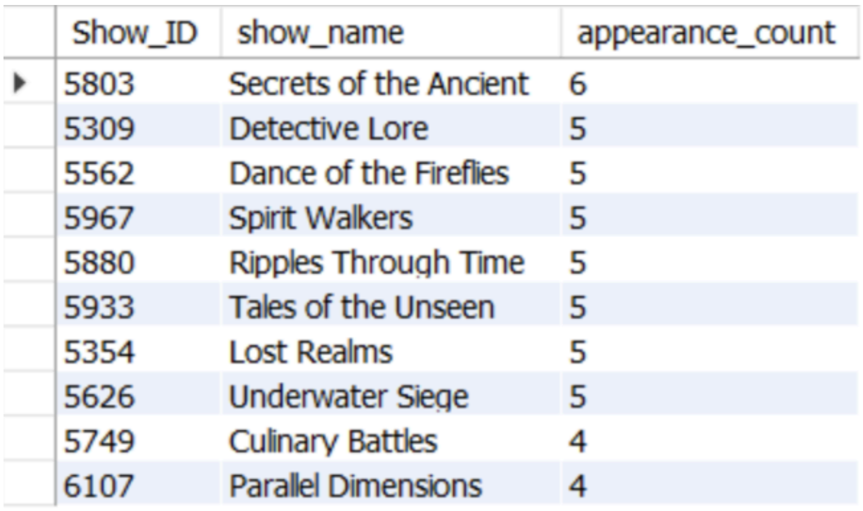


show_id	Movie_Name	Most_Watched
5496	Underwater Colossus	5
6026	RoboTech Rebellion	5
6404	Neon Knight	5
5604	The Mind Reader	4
5284	Digital Dreams	4

6. Most Popular Shows in User's watchlist

This query checks the top 10 shows present in users' watchlists across the platform (i.e. most desired movies currently).

```sql
SELECT
    s.Show_ID,
    IFNULL(m.Name, tv.Name) AS show_name,
    COUNT(*) AS appearance_count
FROM
    WatchlistShow wli
JOIN
    Show_Table s ON wli.Show_ID = s.Show_ID
LEFT JOIN
    Movie m ON s.Movie_ID = m.Movie_ID
LEFT JOIN
    TV_Series tv ON s.TV_Series_ID = tv.TV_Series_ID
GROUP BY
    s.show_ID
ORDER BY
    appearance_count DESC
LIMIT 10;
```

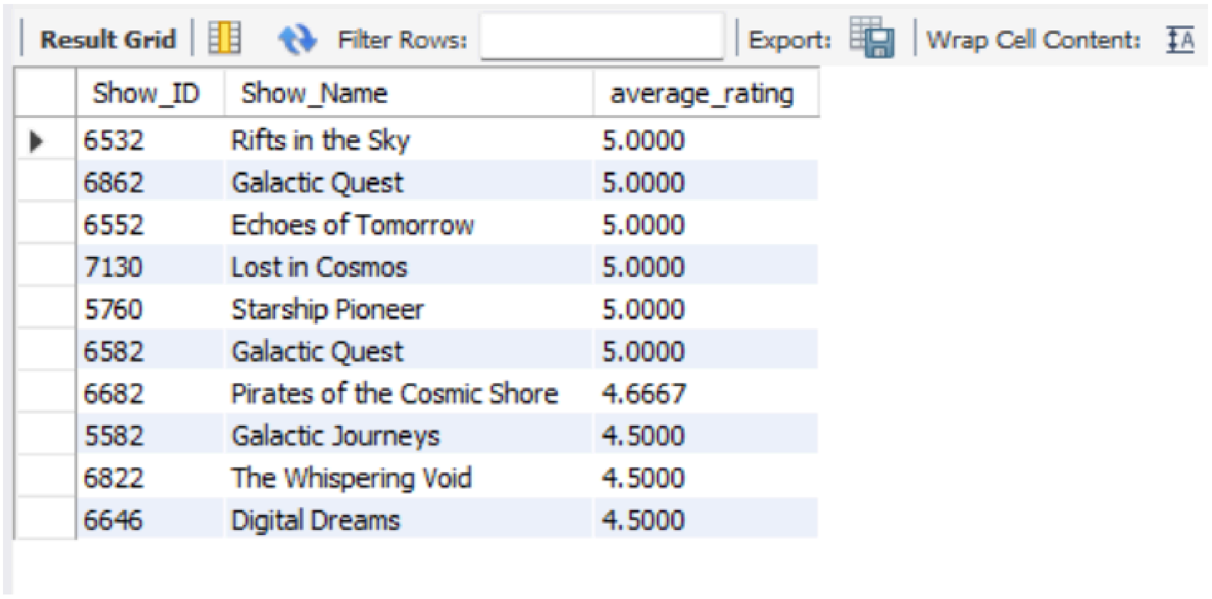


Show_ID	show_name	appearance_count
5803	Secrets of the Ancient	6
5309	Detective Lore	5
5562	Dance of the Fireflies	5
5967	Spirit Walkers	5
5880	Ripples Through Time	5
5933	Tales of the Unseen	5
5354	Lost Realms	5
5626	Underwater Siege	5
5749	Culinary Battles	4
6107	Parallel Dimensions	4

7. Recommendation based on Watch History of the User

This query gives a list of distinct shows that the particular user has not watched yet. It calculates the average rating for each show based on the user's watch history. It recommends shows that belong to the genres that the user watched by ranking those with higher average ratings and doing it randomly within the top-rated shows for variety recommendations.

```
SELECT DISTINCT ST.Show_ID,
       CASE
           WHEN M.Name IS NOT NULL THEN M.Name
           WHEN TS.Name IS NOT NULL THEN TS.Name
           ELSE 'Unknown'
       END AS Show_Name,
       AVG(H.rating) AS average_rating
FROM Show_Table ST
LEFT JOIN Movie M ON ST.Movie_ID = M.Movie_ID
LEFT JOIN TV_Series TS ON ST.TV_Series_ID = TS.TV_Series_ID
LEFT JOIN History H ON ST.Show_ID = H.Show_ID
WHERE ST.Show_ID NOT IN (
    SELECT Show_ID
    FROM History
    WHERE User_ID = 10
)
AND (
    M.Genre IN (
        SELECT M.Genre
        FROM History H
        JOIN Show_Table ST ON H.Show_ID = ST.Show_ID
        JOIN Movie M ON ST.Movie_ID = M.Movie_ID
        WHERE H.User_ID = 10
    )
    OR
    TS.Genre IN (
        SELECT TS.Genre
        FROM History H
        JOIN Show_Table ST ON H.Show_ID = ST.Show_ID
        JOIN TV_Series TS ON ST.TV_Series_ID = TS.TV_Series_ID
        WHERE H.User_ID = 10
    )
)
GROUP BY ST.Show_ID, Show_Name
ORDER BY AVG(H.rating) DESC, RAND()
LIMIT 10;
```



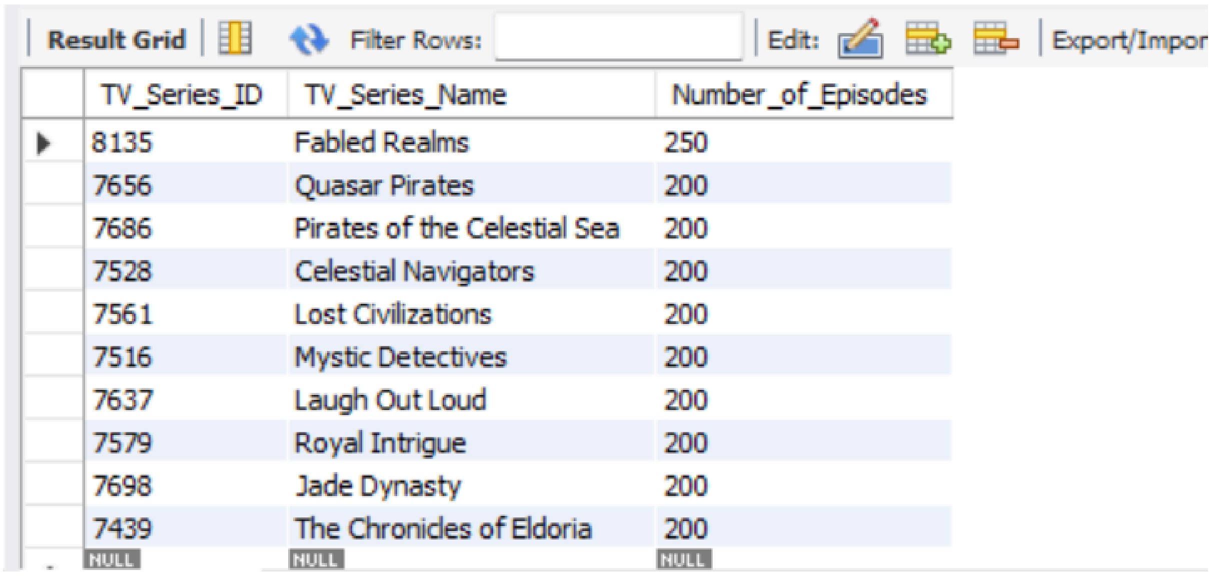
Show_ID	Show_Name	average_rating
6532	Rifts in the Sky	5.0000
6862	Galactic Quest	5.0000
6552	Echoes of Tomorrow	5.0000
7130	Lost in Cosmos	5.0000
5760	Starship Pioneer	5.0000
6582	Galactic Quest	5.0000
6682	Pirates of the Cosmic Shore	4.6667
5582	Galactic Journeys	4.5000
6822	The Whispering Void	4.5000
6646	Digital Dreams	4.5000

8. TV Series with the most number of Episodes

The query gives the top 10 TV series with the highest number of episodes. By selecting the ‘th series id’, ‘name’, and ‘number of episodes’ columns from the ‘tv series’ tabe and order them in descending order based on the number of episodes.

```
SELECT
    TV_Series_ID,
    Name AS TV_Series_Name,
    Number_of_Episodes
FROM
    TV_Series
ORDER BY
    Number_of_Episodes DESC
```

```
LIMIT 10;
```

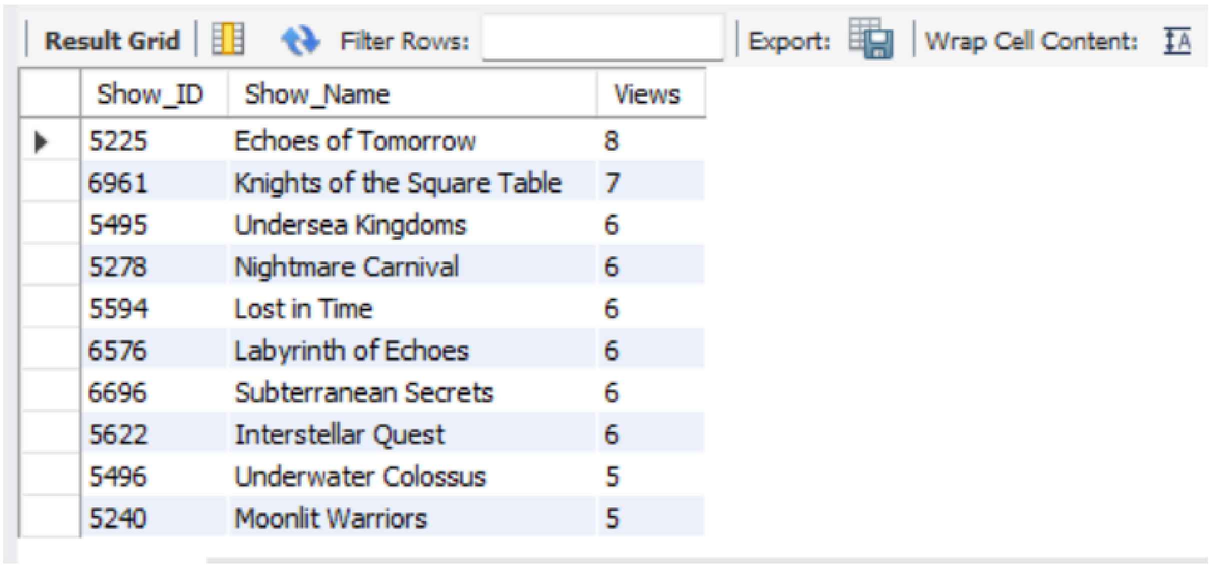


TV_Series_ID	TV_Series_Name	Number_of_Episodes
8135	Fabled Realms	250
7656	Quasar Pirates	200
7686	Pirates of the Celestial Sea	200
7528	Celestial Navigators	200
7561	Lost Civilizations	200
7516	Mystic Detectives	200
7637	Laugh Out Loud	200
7579	Royal Intrigue	200
7698	Jade Dynasty	200
7439	The Chronicles of Eldoria	200
NULL	NULL	NULL

9. Shows with the most Views

The query gives the top 10 shows with the most views. By joining the 'history', 'show table', 'movie', and tv series' tables, it associates viewing history with shows and includes both movies and tv shows.

```
SELECT
    ST.Show_ID,
    CASE
        WHEN M.Name IS NOT NULL THEN M.Name
        WHEN TS.Name IS NOT NULL THEN TS.Name
    END AS Show_Name,
    COUNT(*) AS Views
FROM
    History H
JOIN
    Show_Table ST ON H.Show_ID = ST.Show_ID
LEFT JOIN
    Movie M ON ST.Movie_ID = M.Movie_ID
LEFT JOIN
    TV_Series TS ON ST.TV_Series_ID = TS.TV_Series_ID
GROUP BY
    ST.Show_ID, Show_Name
ORDER BY
    Views DESC
LIMIT 10;
```

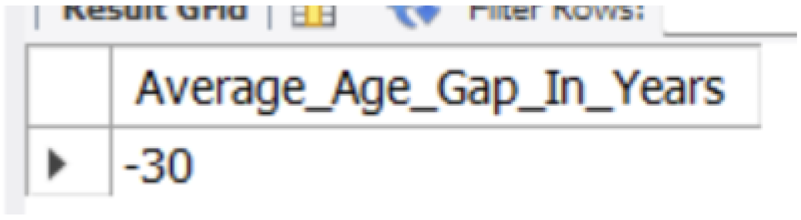

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Show_ID	Show_Name	Views
5225	Echoes of Tomorrow	8
6961	Knights of the Square Table	7
5495	Undersea Kingdoms	6
5278	Nightmare Carnival	6
5594	Lost in Time	6
6576	Labyrinth of Echoes	6
6696	Subterranean Secrets	6
5622	Interstellar Quest	6
5496	Underwater Colossus	5
5240	Moonlit Warriors	5

10.Find the average age gap between users' ages and show release dates

The query calculates the average age gap between users' ages and show release data. It Can be useful to analyze our platform's audience and their preferences

```
SELECT ROUND(AVG(DATEDIFF(u.Birthday, m.Release_Date)/365.25), 0) as
Average_Age_Gap_In_Years
FROM User u, Movie m;
```


Result Grid | Filter Rows:

Average_Age_Gap_In_Years
-30

Stored Procedure

1. Get top 10 rated shows of all time

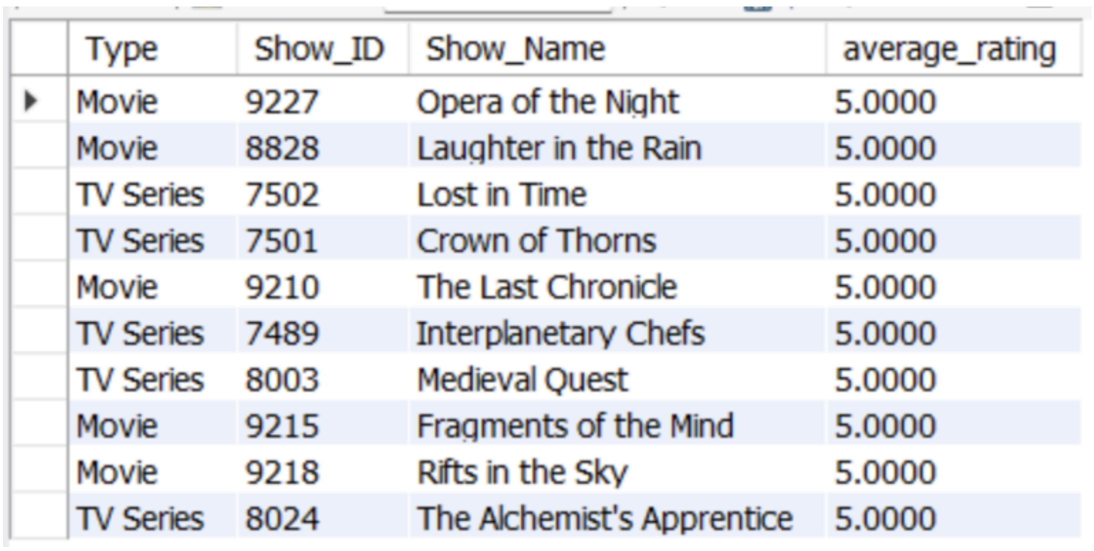
This stored procedure gives the top 10 highest rated shows which includes both movies and tv shows, based on the ratings from the "History" table.

```
CREATE DEFINER=`mm_team06_01`@`%` PROCEDURE `GetTopRatedShows`()
BEGIN
    SELECT
        'Movie' AS Type,
        M.Movie_ID AS Show_ID,
        M.Name AS Show_Name,
        AVG(HM.Rating) AS average_rating
    FROM
        Movie M
    JOIN
        Show_Table SM ON M.Movie_ID = SM.Movie_ID
    LEFT JOIN
        History HM ON SM.Show_ID = HM.Show_ID
    GROUP BY
        M.Movie_ID, M.Name

    UNION ALL

    SELECT
        'TV Series' AS Type,
        TV.TV_Series_ID AS Show_ID,
        TV.Name AS Show_Name,
        AVG(HT.Rating) AS average_rating
    FROM
        TV_Series TV
    JOIN
        Show_Table ST ON TV.TV_Series_ID = ST.TV_Series_ID
    LEFT JOIN
        History HT ON ST.Show_ID = HT.Show_ID
    GROUP BY
        TV.TV_Series_ID, TV.Name

    ORDER BY
        average_rating DESC
    LIMIT 10;
END
```

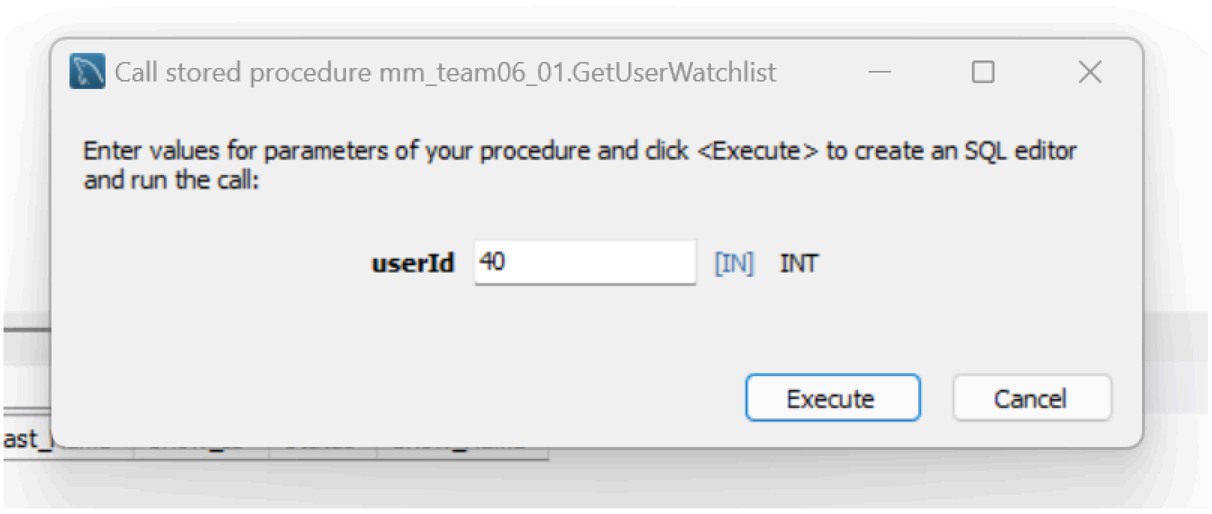


Type	Show_ID	Show_Name	average_rating
Movie	9227	Opera of the Night	5.0000
Movie	8828	Laughter in the Rain	5.0000
TV Series	7502	Lost in Time	5.0000
TV Series	7501	Crown of Thorns	5.0000
Movie	9210	The Last Chronicle	5.0000
TV Series	7489	Interplanetary Chefs	5.0000
TV Series	8003	Medieval Quest	5.0000
Movie	9215	Fragments of the Mind	5.0000
Movie	9218	Rifts in the Sky	5.0000
TV Series	8024	The Alchemist's Apprentice	5.0000

2. Get a user's watchlist by UserId

The store procedure retrieves a user's watchlist, including details about the user, the show in their watchlist, and the status of each show. It joins the 'user', 'watchlist', 'watchlist show', and 'show table' tables to link users with their watchlists and the shows they have added. It also uses the let joins with the 'movie' and 'tv series' tables to give details about the shows in the watchlist, including their names.

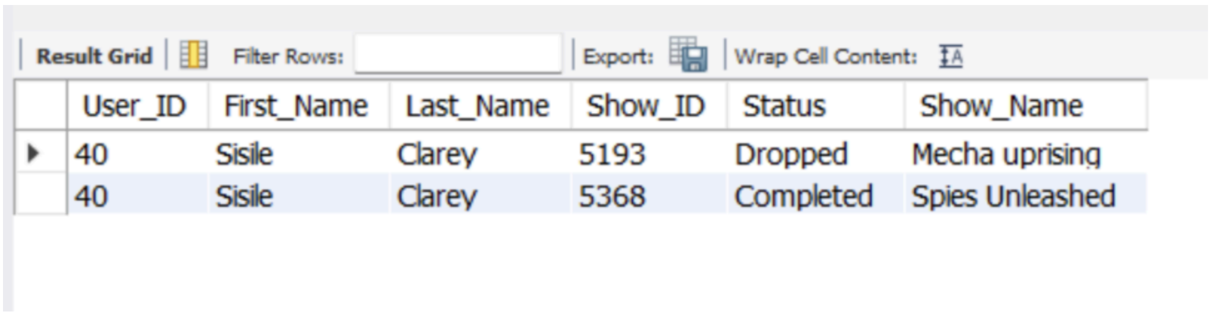
```
DELIMITER $$
CREATE DEFINER=`mm_team06_01`@`%` PROCEDURE `GetUserWatchlist`(IN userId
INT)
BEGIN
      SELECT U.User_ID, U.First_Name, U.Last_Name, S.Show_ID, WS.Status,
            CASE
                  WHEN S.Movie_ID IS NOT NULL THEN M.Name
                  WHEN S.TV_Series_ID IS NOT NULL THEN TS.Name
            END AS Show_Name
      FROM User U
      JOIN Watchlist WL ON U.User_ID = WL.User_ID
      JOIN WatchlistShow WS ON WL.Watchlist_ID = WS.Watchlist_ID
      JOIN Show_Table S ON WS.Show_ID = S.Show_ID
      LEFT JOIN Movie M ON S.Movie_ID = M.Movie_ID
      LEFT JOIN TV_Series TS ON S.TV_Series_ID = TS.TV_Series_ID
      WHERE U.User_ID=userId;
END$$
DELIMITER ;
```



Call stored procedure mm_team06_01.GetUserWatchlist

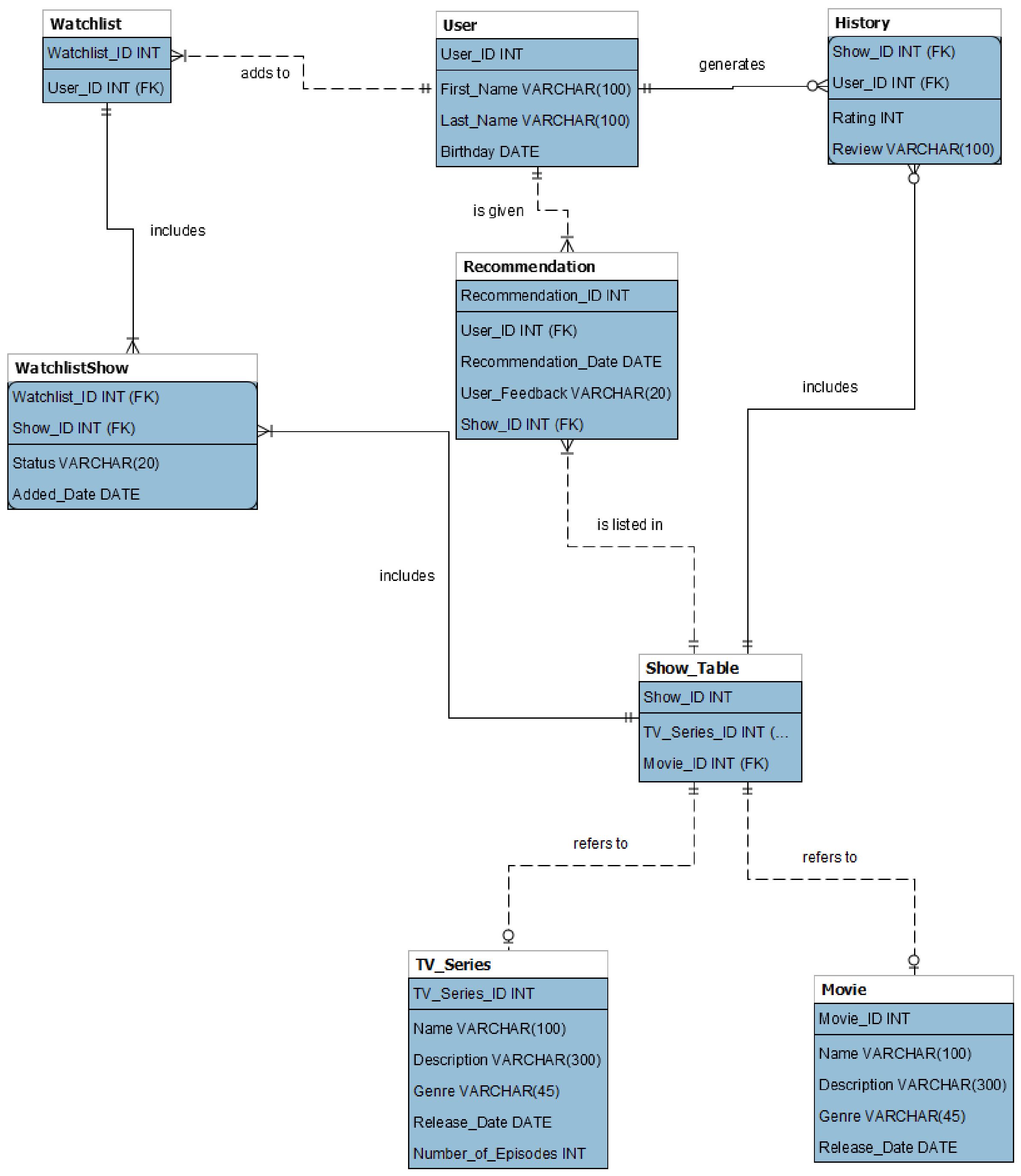
Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

userId 40 [IN] INT



User_ID	First_Name	Last_Name	Show_ID	Status	Show_Name
40	Sisile	Clarey	5193	Dropped	Mecha uprising
40	Sisile	Clarey	5368	Completed	Spies Unleashed

ER Diagram



Updates since Milestone 1:

Per feedback:

- Name in User table has been split into First Name and Last Name
- All ENUMs have been removed and replaced with VARCHAR
- No more blank spaces between table names, they are connected with underscores now