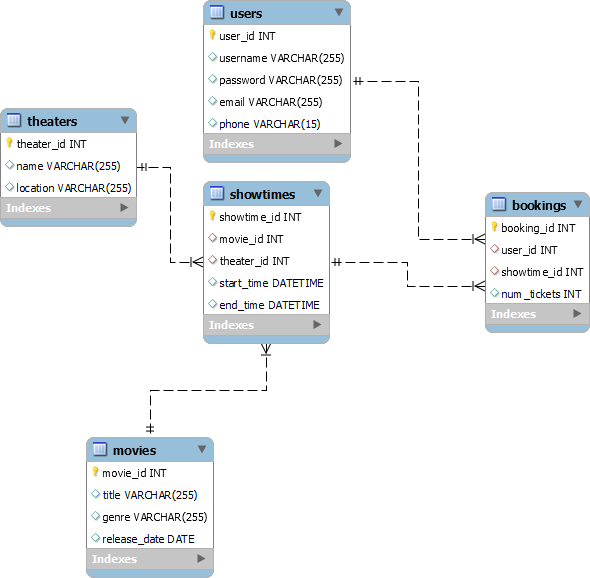
# Assignment 1

**Problem Statement: -** Creating a MySQL project for an Online Movie Ticket Booking System (BookMyShow). BookMyShow-like application involves designing a database to manage information related to movies, theaters, bookings, users, and other relevant entities.

## ER-Diagram: -



**Database Schema:**

### Users Table: -

user\_id (Primary Key), username, password, email, phone

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Users** | | | | |
| **user\_id** | **username** | **password** | **email** | **phone** |
| **1** | Abhishek | abhishek1 | [abhishek@email.com](mailto:abhishek@email.com) | 123-456-7890 |
| **2** | Sarthak | sarthak2 | [sarthak@email.com](mailto:sarthak@email.com) | 234-567-8901 |
| **3** | Prachi | prachi3 | [prachi@email.com](mailto:prachi@email.com) | 345-678-9012 |
| **4** | Priya | priya4 | [priya@email.com](mailto:priya@email.com) | 456-789-0123 |
| **5** | Swapnil | swapnil5 | [swapnil@email.com](mailto:swapnil@email.com) | 567-890-1234 |
| **6** | Sanjana | sanjana6 | [sanjana@email.com](mailto:sanjana@email.com) | 678-901-2345 |
| **7** | Apurva | apurva7 | [apurva@email.com](mailto:apurva@email.com) | 789-012-3456 |
| **8** | Tejashri | tejashri8 | [tejashri@email.com](mailto:tejashri@email.com) | 890-123-4567 |
| **9** | Hasti | hasti9 | [hasti@email.com](mailto:hasti@email.com) | 901-234-5678 |
| **10** | Amit | amit10 | [amit@email.com](mailto:amit@email.com) | 012-345-6789 |
| **11** | Ajit | ajit11 | [ajit@email.com](mailto:ajit@email.com) | 123-456-7890 |
| **12** | Manasi | manasi12 | [manasi@email.com](mailto:manasi@email.com) | 234-567-8901 |
| **13** | Karan | karan13 | [karan@email.com](mailto:karan@email.com) | 345-678-9012 |
| **14** | Shravani | shravani14 | [shravani@email.com](mailto:shravani@email.com) | 456-789-0123 |
| **15** | Mrudul | mrudul15 | [mrudul@email.com](mailto:mrudul@email.com) | 567-890-1234 |
| **16** | Vedant | vedant16 | [vedant@email.com](mailto:vedant@email.com) | 678-901-2345 |
| **17** | Shruti | shruti17 | [shruti@email.com](mailto:shruti@email.com) | 789-012-3456 |
| **18** | Nikhil | nikhil18 | [nikhil@email.com](mailto:nikhil@email.com) | 890-123-4567 |
| **19** | Shivam | shivam19 | [shivam@email.com](mailto:shivam@email.com) | 901-234-5678 |
| **20** | Shashank | shashank20 | [shashank@email.com](mailto:shashank@email.com) | 012-345-6789 |
| **21** | Omkar | omkar18 | [omkar@email.com](mailto:omkar@email.com) | 890-123-4567 |
| **22** | Manas | manas19 | [manas@email.com](mailto:manas@email.com) | 901-234-5678 |
| **23** | Seanna | seanna20 | [seanna@email.com](mailto:seanna@email.com) | 012-345-6789 |
| **24** | Harsh | harsh18 | [harsh@email.com](mailto:harsh@email.com) | 890-123-4567 |
| **25** | Niketan | niketan20 | [niketan@email.com](mailto:niketan@email.com) | 012-345-6789 |

### Movies Table: -

movie\_id (Primary Key), title, genre, release\_date

|  |  |  |  |
| --- | --- | --- | --- |
| **Movies** | | | |
| **movie\_id** | **Title** | **genre** | **release\_date** |
| **1** | The Shawshank Redemption | Drama | 23-09-1994 |
| **2** | The Godfather | Crime, Drama | 24-03-1972 |

|  |  |  |  |
| --- | --- | --- | --- |
| **3** | The Dark Knight | Action, Crime, Drama | 18-07-2008 |
| **4** | Pulp Fiction | Crime, Drama | 14-10-1994 |
| **5** | Schindler's List | Biography, Drama,  History | 15-12-1993 |
| **6** | The Lord of the Rings: The Return of the King | Adventure, Drama,  Fantasy | 17-12-2003 |
| **7** | Fight Club | Drama | 15-10-1999 |
| **8** | Forrest Gump | Drama, Romance | 06-07-1994 |
| **9** | Inception | Action, Adventure, Sci-Fi | 16-07-2010 |
| **10** | The Matrix | Action, Sci-Fi | 31-03-1999 |
| **11** | Goodfellas | Biography, Crime,  Drama | 21-09-1990 |
| **12** | The Lord of the Rings: The Fellowship of the Ring | Adventure, Drama, Fantasy | 19-12-2001 |
| **13** | The Silence of the Lambs | Crime, Drama,  Thriller | 14-02-1991 |
| **14** | The Usual Suspects | Crime, Mystery, Thriller | 15-09-1995 |
| **15** | Se7en | Crime, Drama,  Mystery | 22-09-1995 |
| **16** | The Lord of the Rings: The Two Towers | Adventure, Drama, Fantasy | 18-12-2002 |
| **17** | The Godfather: Part II | Crime, Drama | 20-12-1974 |
| **18** | The Green Mile | Crime, Drama,  Fantasy | 10-12-1999 |
| **19** | Gladiator | Action, Adventure, Drama | 05-05-2000 |
| **20** | Saving Private Ryan | Drama, War | 24-07-1998 |
| **21** | The Departed | Crime, Drama,  Thriller | 06-10-2006 |
| **22** | The Prestige | Drama, Mystery, Sci-Fi | 20-10-2006 |
| **23** | The Lion King | Animation,  Adventure, Drama | 24-06-1994 |
| **24** | The Avengers | Action, Adventure, Sci-Fi | 04-05-2012 |
| **25** | Inglourious Basterds | Adventure, Drama,  War | 21-08-2009 |

### Theaters Table: -

theater\_id (Primary Key), name, location

|  |  |  |
| --- | --- | --- |
| **Theaters** | | |
| **theater\_id** | **name** | **location** |

|  |  |  |
| --- | --- | --- |
| **1** | Phoneix mall | Pune |
| **2** | Cinema Paradise | New York |
| **3** | Theatre Royale | Austin |
| **4** | Pacific Theatres | Kansas City |
| **5** | Harborfront Movies | Indianapolis |
| **6** | Sunrise Cinemas | Charlotte |
| **7** | Palm Tree Theatres | Tampa |
| **8** | Cineplex Central | Orlando |
| **9** | Valley View Cinemas | Cleveland |
| **10** | Midtown Movies | St. Louis |
| **11** | Harborfront Movies | San Diego |
| **12** | Magnolia Cinemas | Minneapolis |
| **13** | Downtown Cinemas | Detroit |
| **14** | Grandview Theatres | Portland |
| **15** | Lakeside Theatres | Dallas |
| **16** | Central Cinema | Miami |
| **17** | Capitol Cinemas | Atlanta |
| **18** | Riverside Cinemas | Denver |
| **19** | Liberty Theatres | Boston |
| **20** | Hollywood Theaters | Seattle |
| **21** | Golden Gate Cinemas | San Francisco |
| **22** | Midtown Movies | Philadelphia |
| **23** | Sunset Theatres | Phoenix |
| **24** | Metroplex Movies | Houston |
| **25** | Downtown Cinemas | Chicago |

### Showtimes Table:

showtime\_id (Primary Key), movie\_id (Foreign Key referencing movies. movie\_id), theater\_id (Foreign Key referencing theaters. theater\_id), start\_time, end\_time

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Showtimes** | | | | |
| **showtime\_id** | **movie\_id** | **theater\_id** | **start\_time** | **end\_time** |
| **1** | 1 | 1 | 14-04-2024 13:00 | 14-04-2024 15:30 |
| 2 | 2 | 2 | 14-04-2024 15:00 | 14-04-2024 17:30 |
| 3 | 3 | 3 | 14-04-2024 18:00 | 14-04-2024 21:00 |
| 4 | 4 | 4 | 14-04-2024 14:30 | 14-04-2024 17:00 |
| 5 | 5 | 5 | 14-04-2024 16:30 | 14-04-2024 19:00 |
| 6 | 6 | 6 | 14-04-2024 19:30 | 14-04-2024 22:30 |
| 7 | 7 | 7 | 14-04-2024 13:30 | 14-04-2024 16:00 |
| 8 | 8 | 8 | 14-04-2024 17:30 | 14-04-2024 20:00 |
| 9 | 9 | 9 | 14-04-2024 14:00 | 14-04-2024 17:00 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 10 | 10 | 14-04-2024 16:00 | 14-04-2024 19:00 |
| 11 | 11 | 11 | 14-04-2024 18:30 | 14-04-2024 21:30 |
| 12 | 12 | 12 | 14-04-2024 20:00 | 14-04-2024 23:00 |
| 13 | 13 | 13 | 14-04-2024 14:30 | 14-04-2024 17:00 |
| 14 | 14 | 14 | 14-04-2024 16:30 | 14-04-2024 19:00 |
| 15 | 15 | 15 | 14-04-2024 19:30 | 14-04-2024 22:00 |
| 16 | 16 | 16 | 14-04-2024 12:00 | 14-04-2024 15:00 |
| 17 | 17 | 17 | 14-04-2024 15:30 | 14-04-2024 18:30 |
| 18 | 18 | 18 | 14-04-2024 18:00 | 14-04-2024 21:00 |
| 19 | 19 | 19 | 14-04-2024 13:00 | 14-04-2024 16:00 |
| 20 | 20 | 20 | 14-04-2024 16:30 | 14-04-2024 19:30 |
| 21 | 21 | 21 | 14-04-2024 19:00 | 14-04-2024 22:00 |
| 22 | 22 | 22 | 14-04-2024 12:30 | 14-04-2024 15:30 |
| 23 | 23 | 23 | 14-04-2024 15:00 | 14-04-2024 17:30 |
| 24 | 24 | 24 | 14-04-2024 17:30 | 14-04-2024 20:30 |
| 25 | 25 | 25 | 14-04-2024 14:00 | 14-04-2024 16:30 |

### Bookings Table: -

booking\_id (Primary Key), user\_id (Foreign Key referencing users.user\_id), showtime\_id (Foreign Key referencing showtimes. showtime\_id), num\_tickets

|  |  |  |  |
| --- | --- | --- | --- |
| **Bookings** | | | |
| **booking\_id** | **user\_id** | **showtime\_id** | **num\_tickets** |
| **1** | 1 | 1 | 2 |
| 2 | 2 | 2 | 3 |
| 3 | 3 | 3 | 1 |
| 4 | 4 | 4 | 2 |
| 5 | 5 | 5 | 2 |
| 6 | 6 | 6 | 3 |
| 7 | 7 | 7 | 1 |
| 8 | 8 | 8 | 2 |
| 9 | 9 | 9 | 3 |
| 10 | 10 | 10 | 2 |
| 11 | 11 | 11 | 2 |
| 12 | 12 | 12 | 1 |
| 13 | 13 | 13 | 2 |
| 14 | 14 | 14 | 3 |
| 15 | 15 | 15 | 2 |
| 16 | 16 | 16 | 2 |
| 17 | 17 | 17 | 1 |
| 18 | 18 | 18 | 3 |
| 19 | 19 | 19 | 2 |
| 20 | 20 | 20 | 3 |
| 21 | 21 | 21 | 2 |
| 22 | 22 | 22 | 2 |
| 23 | 23 | 23 | 1 |
| 24 | 24 | 24 | 3 |
| 25 | 25 | 25 | 2 |

# Assignment 2

## Questions: -

### Let's consider the `movies` table for generating 10 questions: -

Question 1: Retrieve the titles of all movies in the database.

Question 2: Find the genres of movies released on or after a specific date. Question 3: Identify the number of movies in each genre.

Question 4: List the movies released before a certain year. Question 5: Find the most recent movie in the database.

Question 6: Count the total number of movies in the database. Question 7: Display the titles and release dates of movies.

Question 8: Retrieve the titles of movies that belong to a specific genre. Question 9: Identify the oldest movie in the database.

Question 10: List the titles of movies along with their respective genres.

These questions cover various aspects of querying the `movies` table, such as selecting specific columns, filtering based on conditions, and aggregating data.

### Let's consider the relationship between the `movies` and `showtimes` tables for generating 10 questions: -

Question 1: Retrieve the titles and genres of all movies that have showtimes. Question 2: Find the theaters where a specific movie is currently being shown. Question 3: Identify the number of showtimes for each movie.

Question 4: List the movies playing in a particular theater.

Question 5: Find the start times of all showtimes for a specific movie.

Question 6: What are the showtimes for the movie " The Matrix" at the theater named " Cinema Paradise"

Question 7: What is the username of the user who made the booking with ID is 2

Question 8: Identify the theaters with the most showtimes.

Question 9: List the movies along with the corresponding theater locations.

Question 10: Find the total number of tickets booked for each movie.

These questions involve queries that require joining the `movies` and `showtimes` tables to gather information from both tables.

### Let's consider a relationship between the `movies`, `showtimes`, and `bookings` tables for generating 10 questions:

Question 1: Retrieve the titles and genres of all movies for which bookings have been made.

Question 2: Find the usernames and email addresses of users who booked tickets for a specific movie.

Question 3: Identify the total number of tickets booked for each movie.

Question 4: List the movies, showtimes, and corresponding theater names for a specific date. Question 5: Find the total number of tickets booked by a specific user.

Question 6: Display the movie titles, showtimes, and booking details for a specific user. Question 7: Retrieve the total revenue generated by each movie.

Question 8: Find the users who booked tickets for a movie in a specific genre.

Question 9: List the movies, showtimes, and booking details for a specific theater.

Question 10: Identify the most popular showtime (maximum bookings) for each movie.

These questions involve querying information from all three tables by performing joins and aggregations.

# Assignment 3

#### Question 1: Retrieve the titles of all movies in the database.

Ans: - To retrieve the titles of all movies in the database, you can use a simple SELECT query on the

`movies` table.



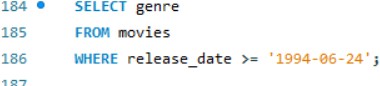
This query selects the `title` column from the `movies` table, returning a list of all movie titles in your database.

### Output: -



#### Question 2: Find the genres of movies released on or after a specific date.

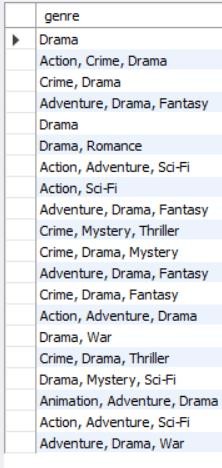
Ans:- To find the genres of movies released on or after a specific date, you can use a SELECT query with a WHERE clause to filter based on the release date.



This query retrieves the genres of movies that were released on or after the specified date from the

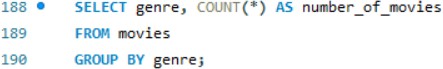
`movies` table.

### Output: -



#### Question 3: Identify the number of movies in each genre.

Ans: - To identify the number of movies in each genre, you can use the COUNT() function along with GROUP BY.



This query counts the number of movies in each genre and presents the result with two columns:

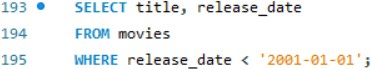
`genre` and `number\_of\_movies`.

### Output: -



#### Question 4: List the movies released before a certain year.

Ans:- To list the movies released before a certain year, you can use a simple SELECT query with a WHERE clause.



This query retrieves the title and release date of movies that were released before the specified year from the `movies` table.

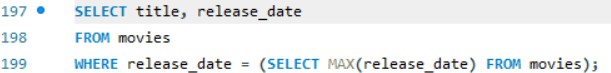
### Output: -



#### Question 5: Find the most recent movie in the database.

Ans:- To find the most recent movie in the database, you can use the MAX() function on the

`release\_date` column.



This query selects the title and release date of the movie(s) with the latest release date from the

`movies` table.

### Output: -



#### Question 6: Count the total number of movies in the database.

Ans:- To count the total number of movies in the database, you can use the COUNT() function.



This query counts the total number of rows in the `movies` table, effectively giving you the total number of movies in the database. The result will be a single column named `total\_movies`.

### Output: -



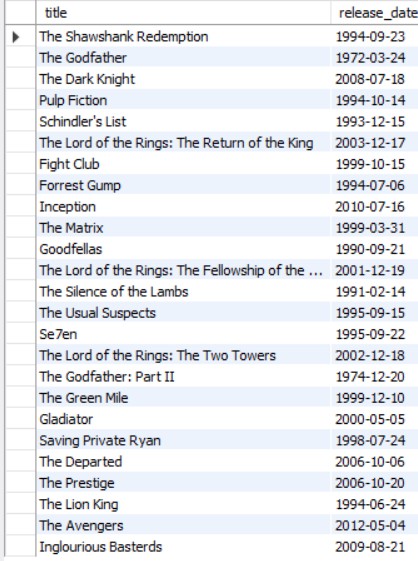
#### Question 7: Display the titles and release dates of movies.

Ans:- To display the titles and release dates of movies.



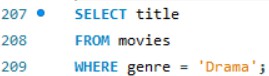
This query retrieves the `title` and `release\_date` columns from the `movies` table.

### Output: -



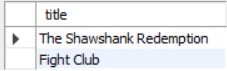
#### Question 8: Retrieve the titles of movies that belong to a specific genre.

Ans:- To retrieve the titles of movies that belong to a specific genre, you can use a SELECT query with a WHERE clause to filter based on the genre.



This query retrieves the titles of movies from the `movies` table that belong to the specified genre.

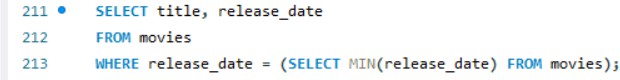
### Output: -



#### Question 9: Identify the oldest movie in the database.

Ans:- To identify the oldest movie in the database, you can use the MIN() function on the

`release\_date` column.



This query selects the title and release date of the movie(s) with the earliest release date from the

`movies` table.

### Output: -



#### Question 10: List the titles of movies along with their respective genres.

Ans:- To list the titles of movies along with their respective genres.



This query retrieves the `title` and `genre` columns from the `movies` table.

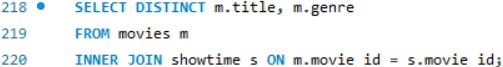
**Output: -**



# Assignment 4

#### Question 1: Retrieve the titles and genres of all movies that have showtimes.

Ans:- To retrieve the titles and genres of all movies that have showtimes, you can use a SELECT query with a JOIN clause to combine information from the `movies` and `showtimes` tables.



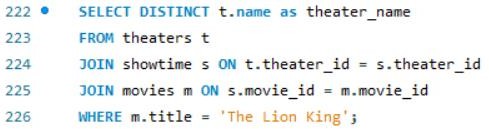
This query selects distinct movie titles and genres from the `movies` table for movies that have associated showtimes in the `showtimes` table.

### Output: -



#### Question 2: Find the theaters where a specific movie is currently being shown.

Ans:- To find the theaters where a specific movie is currently being shown, you can use a SELECT query with a JOIN clause involving the `movies`, `showtimes`, and `theaters` tables.



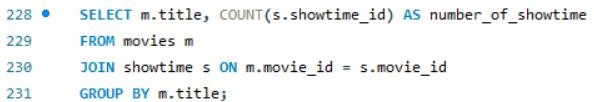
Replace `'specific\_movie\_title'` with the actual title of the movie you're interested in. This query retrieves the distinct names of theaters where the specified movie is currently being shown.

### Output: -



#### Question 3: Identify the number of showtimes for each movie.

Ans:- To identify the number of showtimes for each movie, you can use a SELECT query with a JOIN and GROUP BY clause.



This query retrieves the movie title and counts the number of showtimes for each movie using the

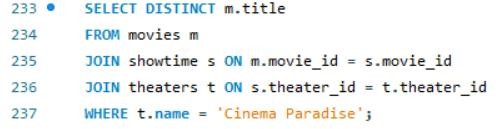
`COUNT` function. The result will show the title of each movie along with the corresponding number of showtimes. Adjust the column and table names based on your actual database schema if needed.

### Output: -



#### Question 4: List the movies playing in a particular theater.

Ans:- To list the movies playing in a particular theater, you can use a SELECT query with a JOIN clause involving the `movies`, `showtimes`, and `theaters` tables.



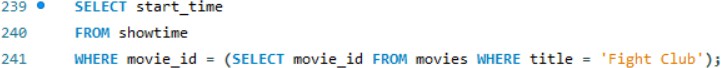
Replace `'specific\_theater\_name'` with the actual name of the theater you're interested in. This query retrieves the distinct titles of movies playing in the specified theater.

### Output: -



#### Question 5: Find the start times of all showtimes for a specific movie.

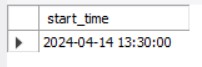
Ans:- To find the start times of all showtimes for a specific movie, you can use a SELECT query with a JOIN clause involving the `movies` and `showtimes` tables.



Replace `'specific\_movie\_title'` with the actual title of the movie you're interested in. This query

retrieves the start times of all showtimes for the specified movie. Adjust the column and table names based on your actual database schema if needed.

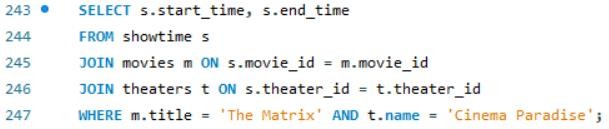
### Output: -



#### Question 6: What are the showtimes for the movie " The Matrix" at the theater named " Cinema Paradise"

Ans: - To retrieve the showtimes for the movie " The Matrix " at the theater named " Cinema Paradise," you would need to perform a JOIN operation between the Movies table and the

Showtimes table, and another JOIN operation between the Theaters table and the Showtimes table, filtering the results based on the movie title and theater name.



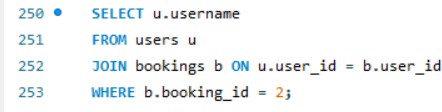
This query retrieves the start time and end time from the Showtimes table, where the movie title is " The Matrix " and the theater name is " Cinema Paradise ".

### Output: -



#### Question 7: What is the username of the user who made the booking with ID is 2

Ans: - To retrieve the username of the user who made the booking with ID 2, you can perform a JOIN operation between the Users table and the Bookings table, filtering the results based on the booking ID.



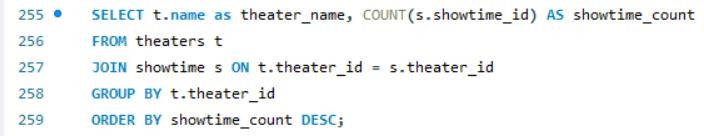
This query selects the username from the Users table where the user ID in the Users table matches the user ID associated with the booking ID 2 in the Bookings table.

### Output: -



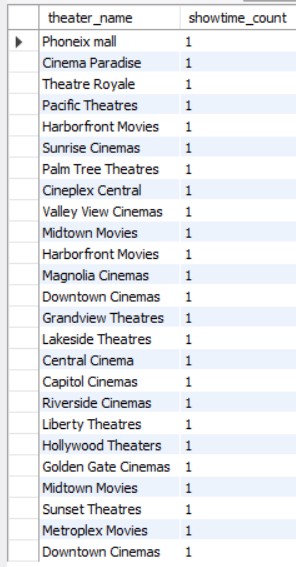
#### Question 8: Identify the theaters with the most showtimes.

Ans:- To identify the theaters with the most showtimes, you can use a SELECT query with a JOIN, GROUP BY, and ORDER BY clauses.



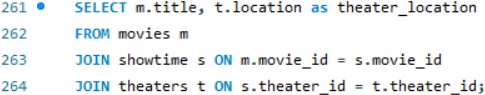
This query counts the number of showtimes for each theater, groups the results by theater, and then orders the result in descending order based on the showtime count. The theater with the most showtimes will appear at the top of the result set.

### Output: -



#### Question 9: List the movies along with the corresponding theater locations.

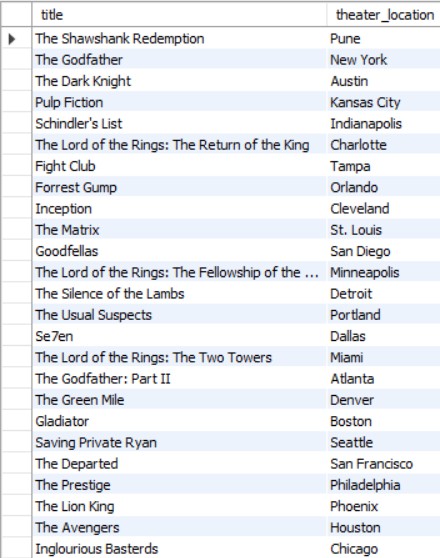
Ans:- To list the movies along with the corresponding theater locations, you can use a SELECT query with a JOIN clause involving the `movies`, `showtimes`, and `theaters` tables.



This query retrieves the movie titles and theater locations by joining the `movies`, `showtimes`, and

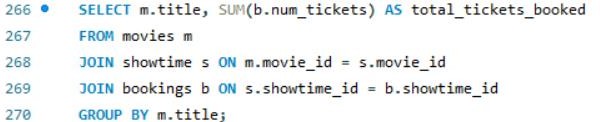
`theaters` tables.

### Output: -



#### Question 10: Find the total number of tickets booked for each movie.

Ans:- To find the total number of tickets booked for each movie, you can use a SELECT query with a JOIN, GROUP BY, and SUM clause. Assuming there's a `bookings` table that stores information about each booking.



This query joins the `movies`, `showtimes`, and `bookings` tables, groups the result by movie title, and calculates the total number of tickets booked for each movie.

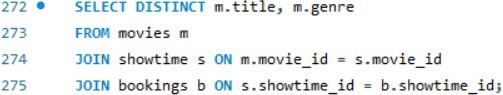
**Output: -**



# Assignment 5

#### Question 1: Retrieve the titles and genres of all movies for which bookings have been made.

Ans: - To retrieve the titles and genres of all movies for which bookings have been made, you can use a SELECT query with a JOIN clause involving the `movies`, `showtimes`, and `bookings` tables.



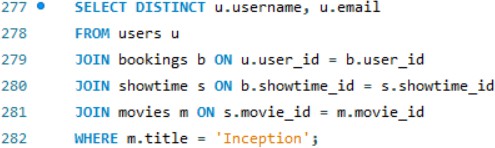
This query retrieves the distinct titles and genres of movies for which bookings have been made by joining the `movies`, `showtimes`, and `bookings` tables.

#### Output: -



**Question 2: Find the usernames and email addresses of users who booked tickets for a specific movie.**

Ans:- To find the usernames and email addresses of users who booked tickets for a specific movie, you can use a SELECT query with a JOIN clause involving the `users`, `bookings`, `showtimes`, and `movies` tables.



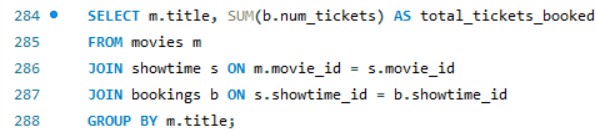
Replace `'specific\_movie\_title'` with the actual title of the movie you're interested in. This query retrieves the distinct usernames and email addresses of users who booked tickets for the specified movie.

#### Output: -



**Question 3: Identify the total number of tickets booked for each movie.**

Ans:- To identify the total number of tickets booked for each movie, you can use a SELECT query with a JOIN, GROUP BY, and SUM clause. Assuming there's a `bookings` table that stores information about each booking.



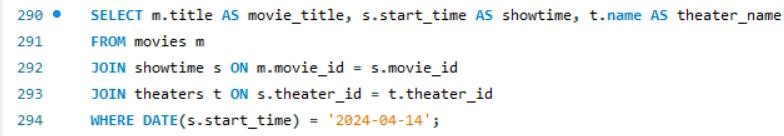
This query joins the `movies`, `showtimes`, and `bookings` tables, groups the result by movie title, and calculates the total number of tickets booked for each movie. Adjust the column and table names based on your actual database schema if needed.

#### Output: -



**Question 4: List the movies, showtimes, and corresponding theater names for a specific date.**

Ans:- To list the movies, showtimes, and corresponding theater names for a specific date, you can use a SELECT query with a JOIN clause involving the `movies`, `showtimes`, and `theaters` tables.



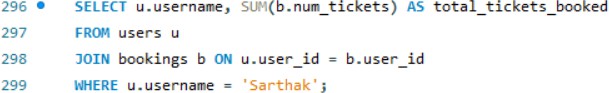
Replace `'specific\_date'` with the actual date you're interested in. This query retrieves the movie titles, showtimes, and corresponding theater names for the specified date.

#### Output: -



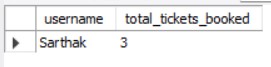
**Question 5: Find the total number of tickets booked by a specific user.**

Ans:- To find the total number of tickets booked by a specific user, you can use a SELECT query with a WHERE clause to filter based on the user's information and then use the SUM function to calculate the total number of tickets. Assuming there's a `users` table storing user information and a `bookings` table for booking details.



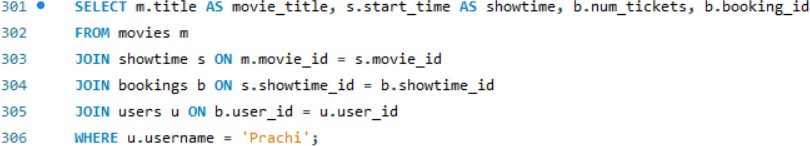
Replace `'specific\_username'` with the actual username of the user you're interested in. This query retrieves the total number of tickets booked by the specified user.

#### Output: -



**Question 6: Display the movie titles, showtimes, and booking details for a specific user.**

Ans:- To display the movie titles, showtimes, and booking details for a specific user, you can use a SELECT query with a JOIN clause involving the `movies`, `showtimes`, `bookings`, and `users` tables.



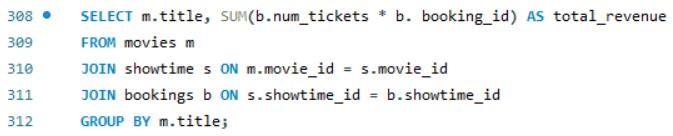
Replace `'specific\_username'` with the actual username of the user you're interested in. This query retrieves the movie titles, showtimes, and booking details for the specified user.

#### Output: -



**Question 7: Retrieve the total revenue generated by each movie.**

Ans:- To retrieve the total revenue generated by each movie, you can use a SELECT query with a JOIN, GROUP BY, and SUM clause. Assuming there's a `bookings` table that stores information about each booking and includes the price per ticket.



This query joins the `movies`, `showtimes`, and `bookings` tables, groups the result by movie title, and calculates the total revenue generated by each movie by multiplying the number of tickets with the

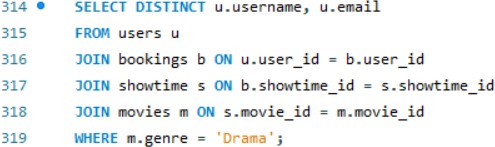
ticket price and then summing them up.

#### Output: -



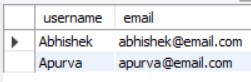
**Question 8: Find the users who booked tickets for a movie in a specific genre.**

Ans:- To find the users who booked tickets for a movie in a specific genre, you can use a SELECT query with a JOIN clause involving the `users`, `bookings`, `showtimes`, and `movies` tables.



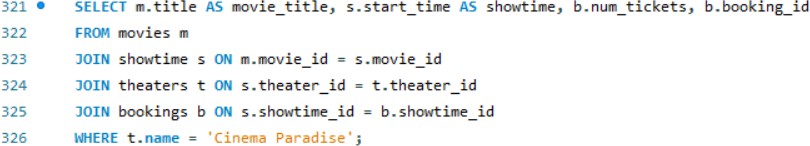
Replace `'specific\_genre'` with the actual genre you're interested in. This query retrieves the distinct usernames and email addresses of users who booked tickets for movies in the specified genre.

#### Output: -



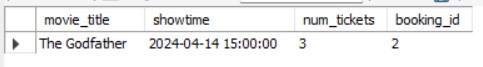
**Question 9: List the movies, showtimes, and booking details for a specific theater.**

Ans:- To list the movies, showtimes, and booking details for a specific theater, you can use a SELECT query with a JOIN clause involving the `movies`, `showtimes`, `bookings`, and `theaters` tables.



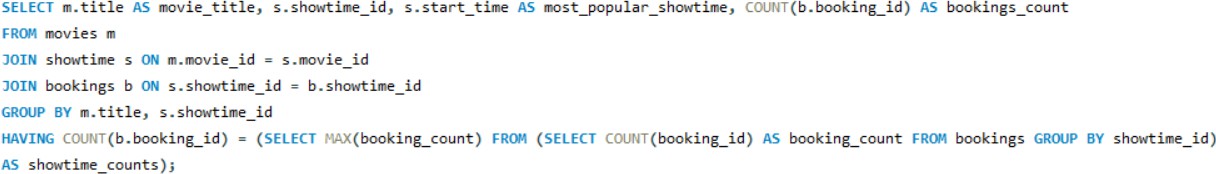
Replace `'specific\_theater\_name'` with the actual name of the theater you're interested in. This query retrieves the movie titles, showtimes, and booking details for the specified theater.

#### Output: -



**Question 10: Identify the most popular showtime (maximum bookings) for each movie.**

Ans:- To identify the most popular showtime (maximum bookings) for each movie, you can use a SELECT query with a JOIN, GROUP BY, and MAX clause.



This query joins the `movies`, `showtimes`, and `bookings` tables, groups the result by movie title and showtime\_id, calculates the number of bookings for each showtime, and selects only those showtimes with the maximum bookings count for each movie.

**Output: -**

