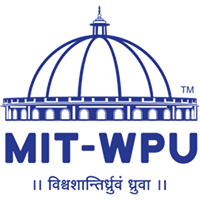
**-**

**MiniProject Report**

on

**Online Movie Ticket Booking Management System**

Submitted by

Group -08

**Project Members**

Dhairya Hindoriy1032210405

Rohan Nair 1032210342

Sudhanshu Kale 10332210485

**Under the Guidance of**

**Prof. Varsha Powar**

**School of Computer Engineering and Technology**

**MIT World Peace University, Kothrud,**

**Pune 411 038, Maharashtra - India**

**2022-2023**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR NO.** | **TABLE OF CONTENTS** | **PAGE NO.** |
| 1. | ABSTRACT | 3 |
| 2. | 2.1 INTRODUCTION  2.2 OBJECTIVES | 3-4 |
| 3. | PROBLEM DEFINITION | 4 |
| 4. | TOOLS AND TECHNOLOGIES ARE | 5 |
| 5. | DATABASE DESIGN (ER DIAGRAM) | 5 |
| 6. | DATABASE SCHEMA | 6 |
| 7. | DDL | 7-12 |
| 8. | DML ALONG WITH QUERIES | 12-19 |
| 9. | DCL | 20 |
| 10. | TRIGGERS | 20-25 |
| 11. | PL SQL PROCEDURES / FUNCTIONS | 25-27 |
| 12. | FRONT AND GUI SCREENSHOTS | 27-31 |
| 13. | CONCLUSION | 31 |
| 14. | REFERENCES FROM IEEE FORMAT | 32 |
|  |  |  |

Abstract:

An online movie ticket booking management system is a web-based application that provides users with the ability to search for and purchase movie tickets from the convenience of their own homes. The system typically includes features such as a database of movies, theatres, and show timings, as well as an interface for users to search for and select tickets, make payments, and receive confirmation of their bookings.

The system may also include features for managing the inventory of tickets, monitoring sales, generating reports, and administering user accounts. The system is typically designed to be user-friendly and easy to navigate, with a simple and intuitive interface that allows users to quickly find the movies they want to watch and book their tickets with ease.

Overall, an online movie ticket booking management system is an essential tool for movie theatres and cinema chains looking to streamline their ticketing processes and improve the customer experience. By providing users with a convenient and efficient way to book tickets online, the system helps to increase ticket sales, reduce waiting times, and enhance overall customer satisfaction.

Title:

Online Movie Ticket Booking Management System

Introduction:

An online movie ticket booking management system is a web-based application that enables users to purchase movie tickets online. The system provides a convenient way for moviegoers to browse through movie listings, select show timings and theatres, and purchase tickets from the comfort of their own homes. It eliminates the need for physical box office sales, making the process of buying movie tickets simpler and faster.

The online movie ticket booking management system offers several advantages over traditional ticket booking methods. It allows users to avoid long queues at the box office and provides them with the flexibility to book tickets anytime and from anywhere. The system also enables users to easily compare movie schedules, select preferred seats, and make secure online payments.

The online movie ticket booking management system is a critical tool for cinema chains and movie theatres to manage their ticketing processes efficiently. It helps to reduce costs associated with physical ticket sales, improves the accuracy of ticket inventory, and provides valuable data analytics to optimize sales and marketing efforts.

This paper aims to explore the various components and features of an online movie ticket booking management system. It will cover the architecture of the system, the different modules involved in its functioning, and the benefits it offers to cinema chains and moviegoers alike. We will also discuss the challenges involved in implementing and maintaining an effective online ticketing system and provide some best practices for overcoming them.

Problem Definition:

Tools & Technologies:

Front end: HTML, CSS, JavaScript

1. HTML: HTML is used to create and save web document. E.g. Notepad/Notepad++

2. CSS: (Cascading Style Sheets) Create attractive Layout

3. Bootstrap: responsive design mobile friendly site

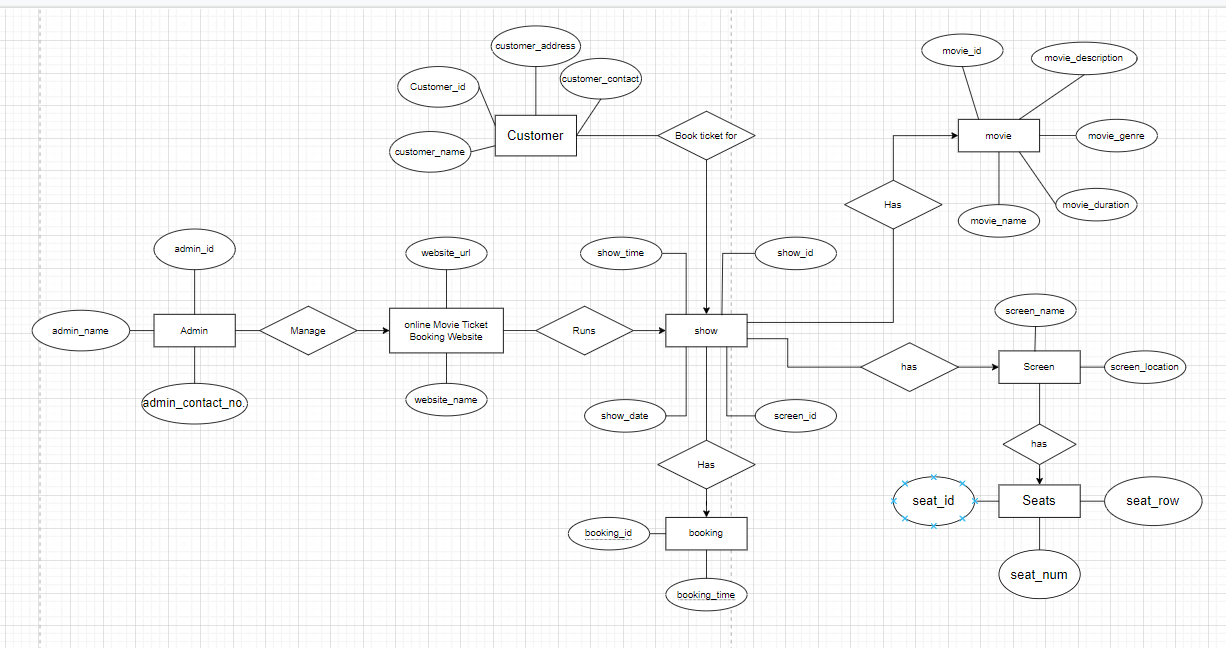
4. JavaScript: it is a programming language, commonly use with web browsers.

Back end: PHP, MySQL

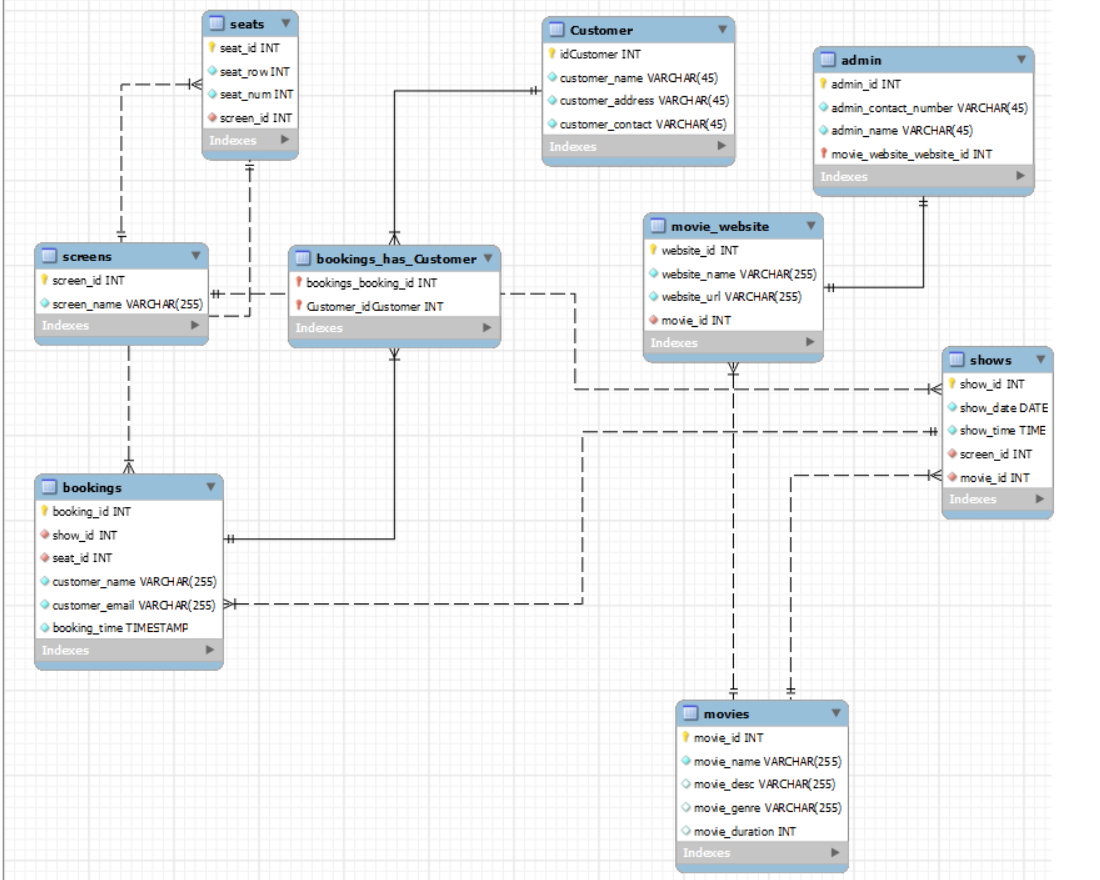
1. PHP: Hypertext Pre-processor (PHP) is a technology that allows software developers to create dynamically generated web pages, in HTML, XML, or other document types, as per client request. PHP is open-source software.

2. MySQL: MySQL is a database, widely used for accessing querying, updating, and managing data in databases

Database Design (ER Diagram):



Database Schema:



DDL (Create table and constraints commands) script:

drop DATABASE movies99990;

CREATE DATABASE movies99990;

USE movies99990;

CREATE TABLE Movies (

movie\_id INT AUTO\_INCREMENT PRIMARY KEY,

movie\_name VARCHAR(255) NOT NULL,

movie\_desc VARCHAR(255),

movie\_genre VARCHAR(255),

movie\_duration INT

);

CREATE TABLE Screens (

screen\_id INT AUTO\_INCREMENT PRIMARY KEY,

screen\_name VARCHAR(255) NOT NULL

);

CREATE TABLE Shows (

show\_id INT AUTO\_INCREMENT PRIMARY KEY,

show\_date DATE NOT NULL,

show\_time TIME NOT NULL,

screen\_id INT NOT NULL,

movie\_id INT NOT NULL,

FOREIGN KEY (screen\_id) REFERENCES Screens(screen\_id),

FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id)

);

CREATE TABLE Seats (

seat\_id INT AUTO\_INCREMENT PRIMARY KEY,

seat\_row INT NOT NULL,

seat\_num INT NOT NULL,

screen\_id INT NOT NULL,

FOREIGN KEY (screen\_id) REFERENCES Screens(screen\_id)

);

CREATE TABLE Bookings (

booking\_id INT AUTO\_INCREMENT PRIMARY KEY,

show\_id INT NOT NULL,

seat\_id INT NOT NULL,

customer\_name VARCHAR(255) NOT NULL,

customer\_email VARCHAR(255) NOT NULL,

booking\_time TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (show\_id) REFERENCES Shows(show\_id),

FOREIGN KEY (seat\_id) REFERENCES Seats(seat\_id)

);

CREATE TABLE Movie\_Website (

website\_id INT AUTO\_INCREMENT PRIMARY KEY,

website\_name VARCHAR(255) NOT NULL,

website\_url VARCHAR(255) NOT NULL,

movie\_id INT NOT NULL,

FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id)

);

-- Sample data insertion

INSERT INTO Movies (movie\_name, movie\_desc, movie\_genre, movie\_duration)

VALUES ('The Matrix', 'A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers.', 'Action, Sci-Fi', 136);

INSERT INTO Screens (screen\_name)

VALUES ('Screen 1'), ('Screen 2'), ('Screen 3');

INSERT INTO Shows (show\_date, show\_time, screen\_id, movie\_id)

VALUES ('2023-03-16', '20:00:00', 1, 1), ('2023-03-17', '18:00:00', 2, 1), ('2023-03-17', '20:00:00', 1, 1);

INSERT INTO Seats (seat\_row, seat\_num, screen\_id)

VALUES (1, 1, 1), (1, 2, 1), (1, 3, 1), (2, 1, 1), (2, 2, 1), (2, 3, 1), (3, 1, 1), (3, 2, 1), (3, 3, 1);

-- Query to get all movies

SELECT \* FROM Movies;

-- Query to get all screens

SELECT \* FROM Screens;

-- Query to get all shows

SELECT Shows.show\_id, Shows.show\_date, Shows.show\_time, Movies.movie\_name, Screens.screen\_name

FROM Shows

INNER JOIN Movies ON Shows.movie\_id = Movies.movie\_id

INNER JOIN Screens ON Shows.screen\_id = Screens.screen\_id;

-- Query to get all seats available for a particular show

SELECT \* FROM Seats WHERE Seats.screen\_id = 1;

-- Query to book a seat for a customer

INSERT INTO Bookings (show\_id, seat\_id, customer\_name, customer\_email)

VALUES (1, 1, 'Aditya Patil', 'adi\*\*\*\*\*\*@gmail.com');

-- Query to get all bookings for a particular show

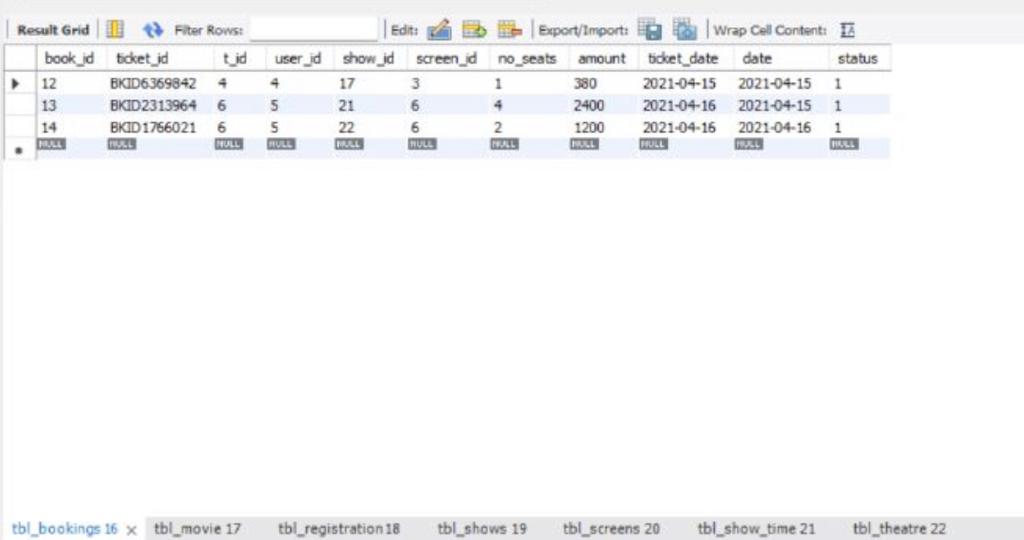
SELECT \* FROM Bookings WHERE Bookings.show\_id = 1;

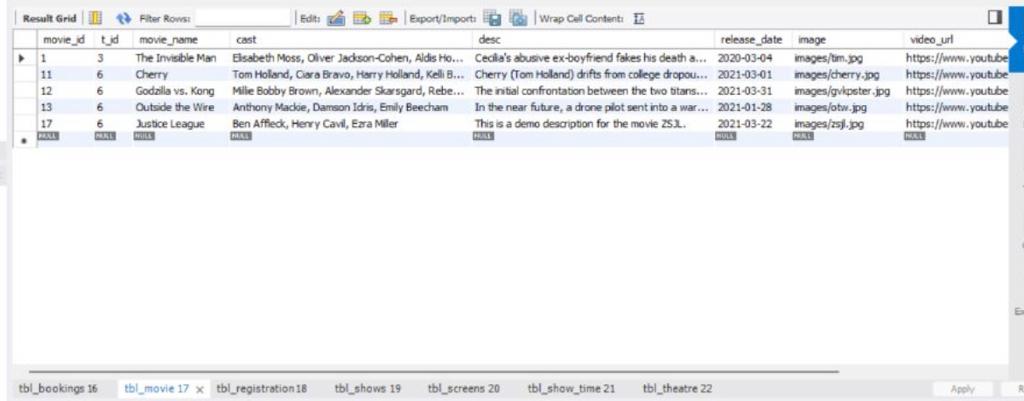
-- Query to get all bookings for a particular customer

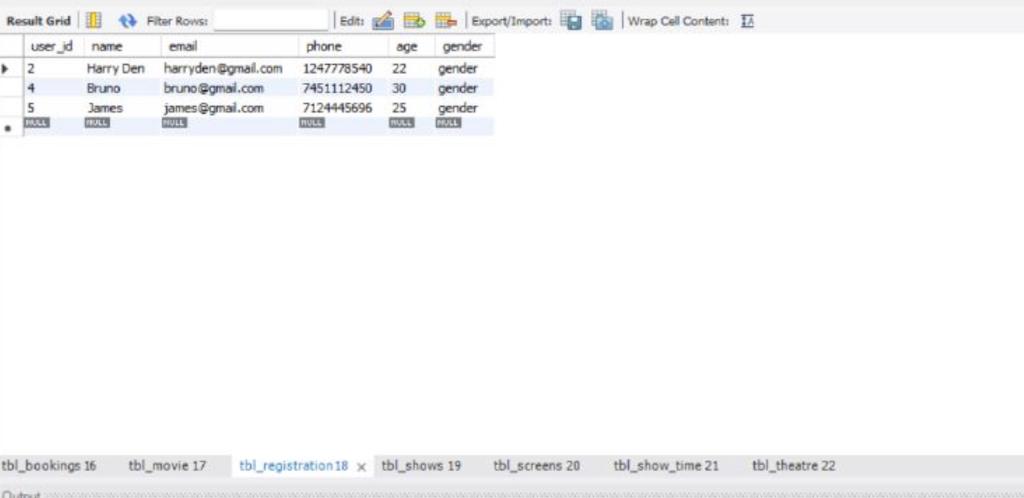
SELECT \* FROM Bookings WHERE Bookings.customer\_name = 'Aditya Patil';

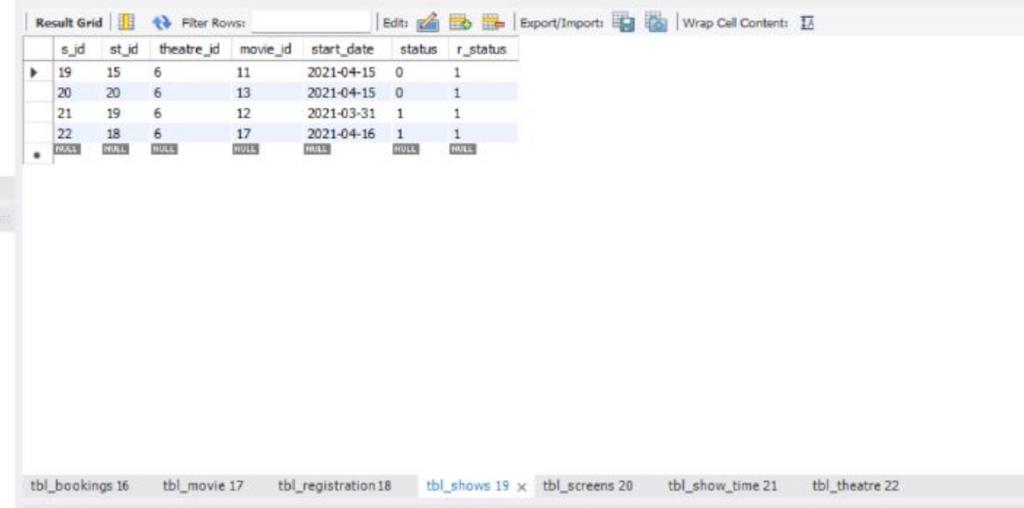
-- Query to get all bookings for a particular show and seat

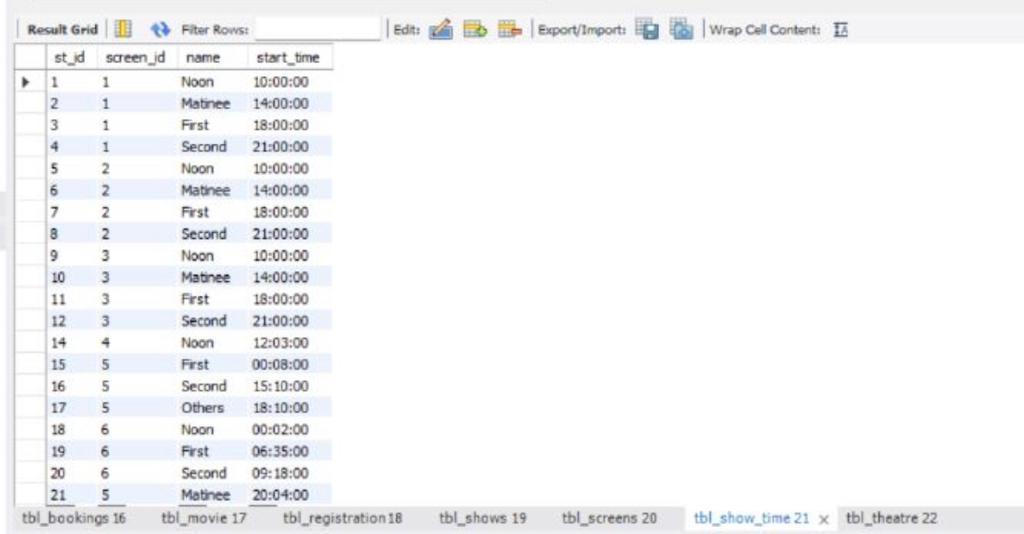
SELECT \* FROM Bookings;

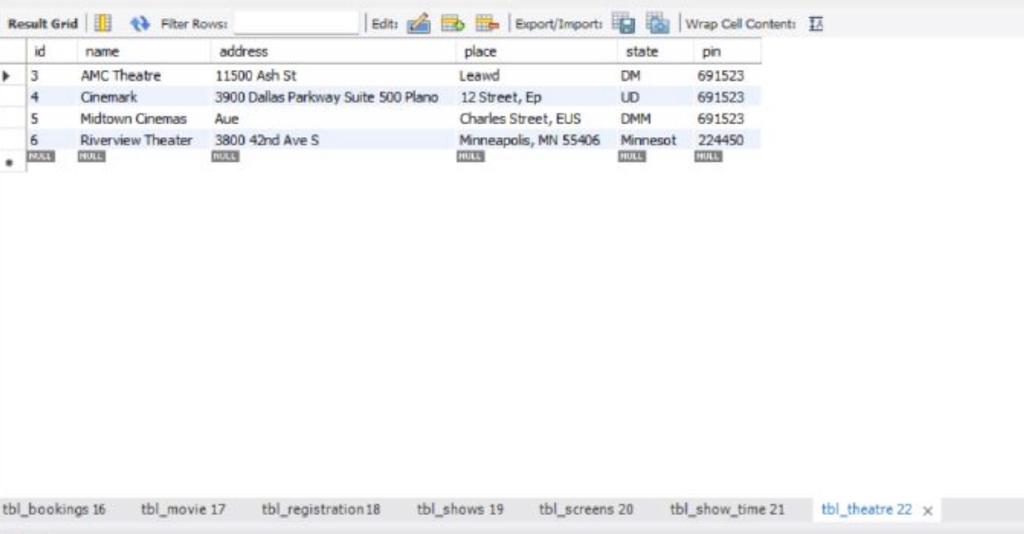


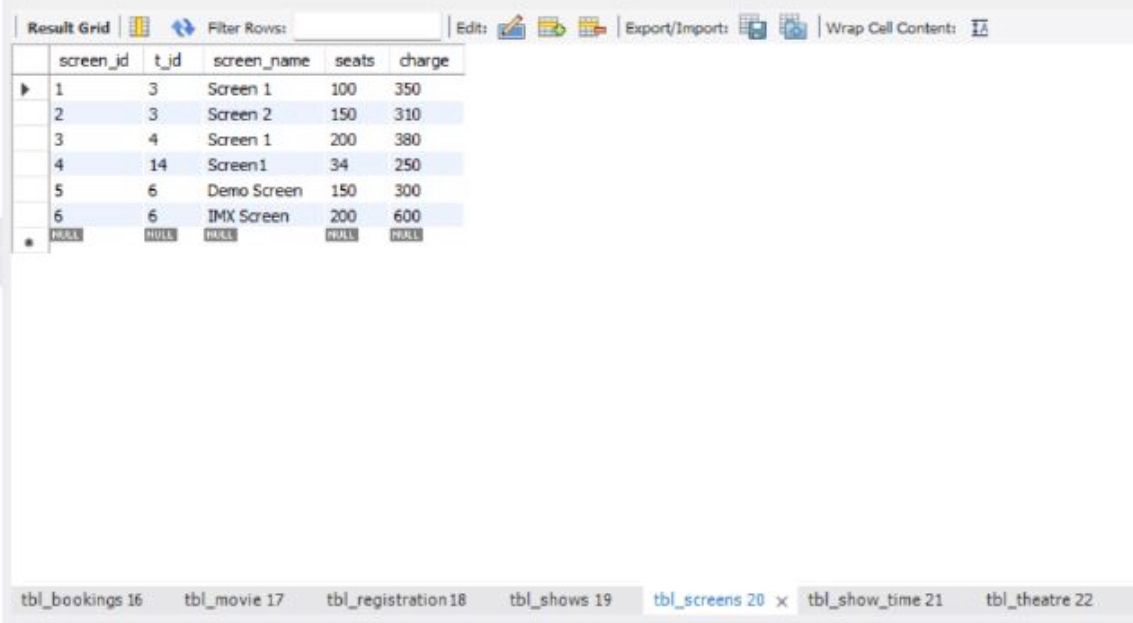


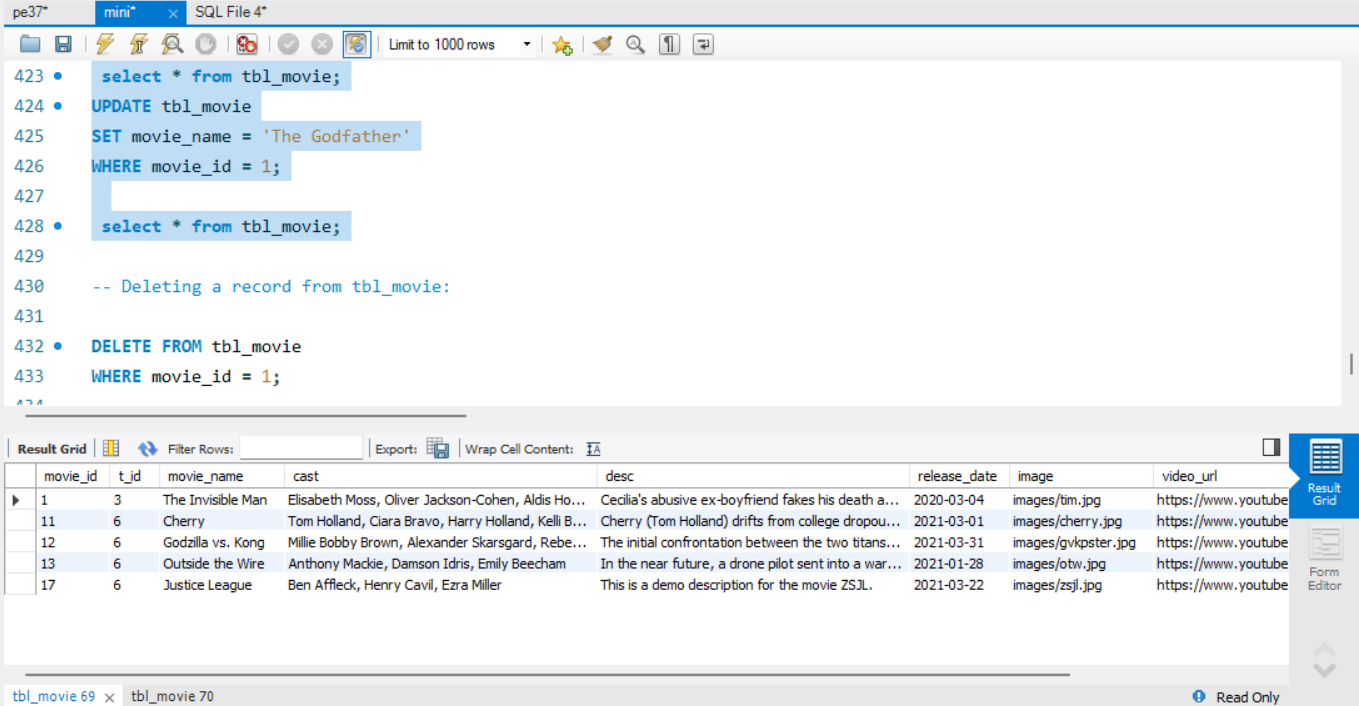


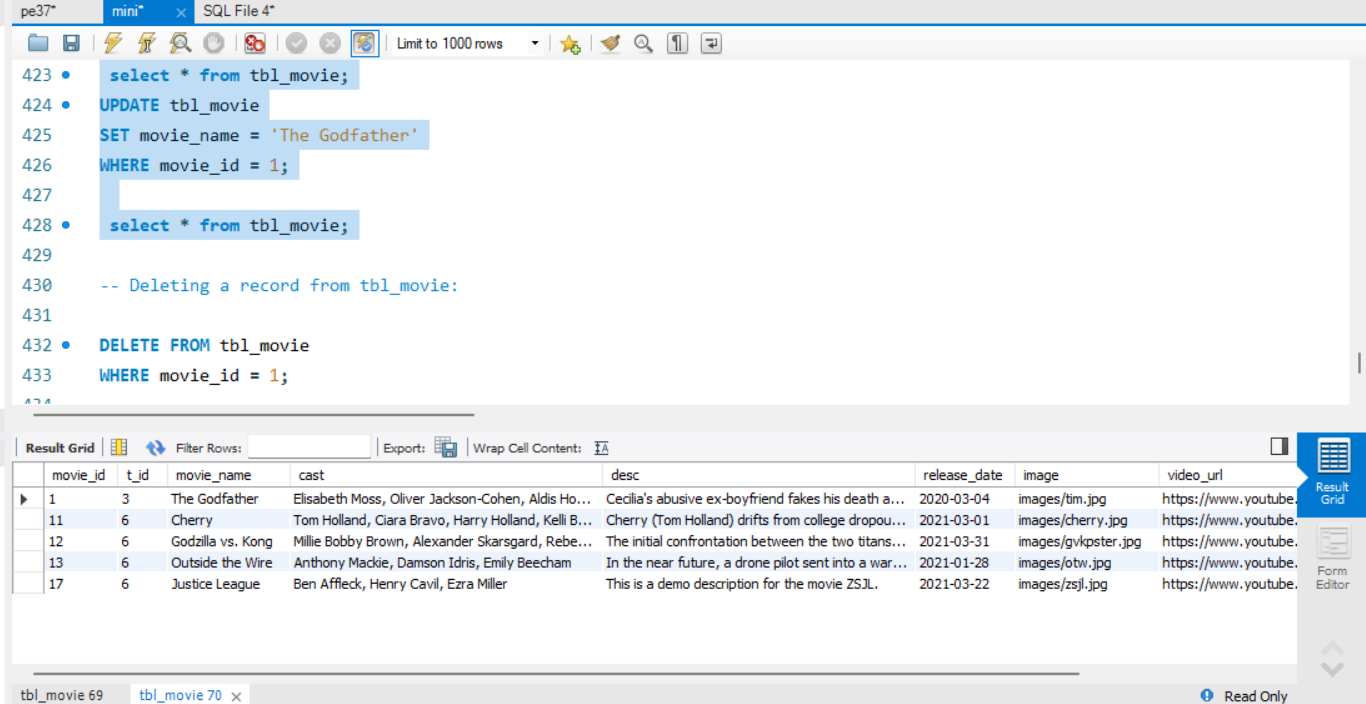








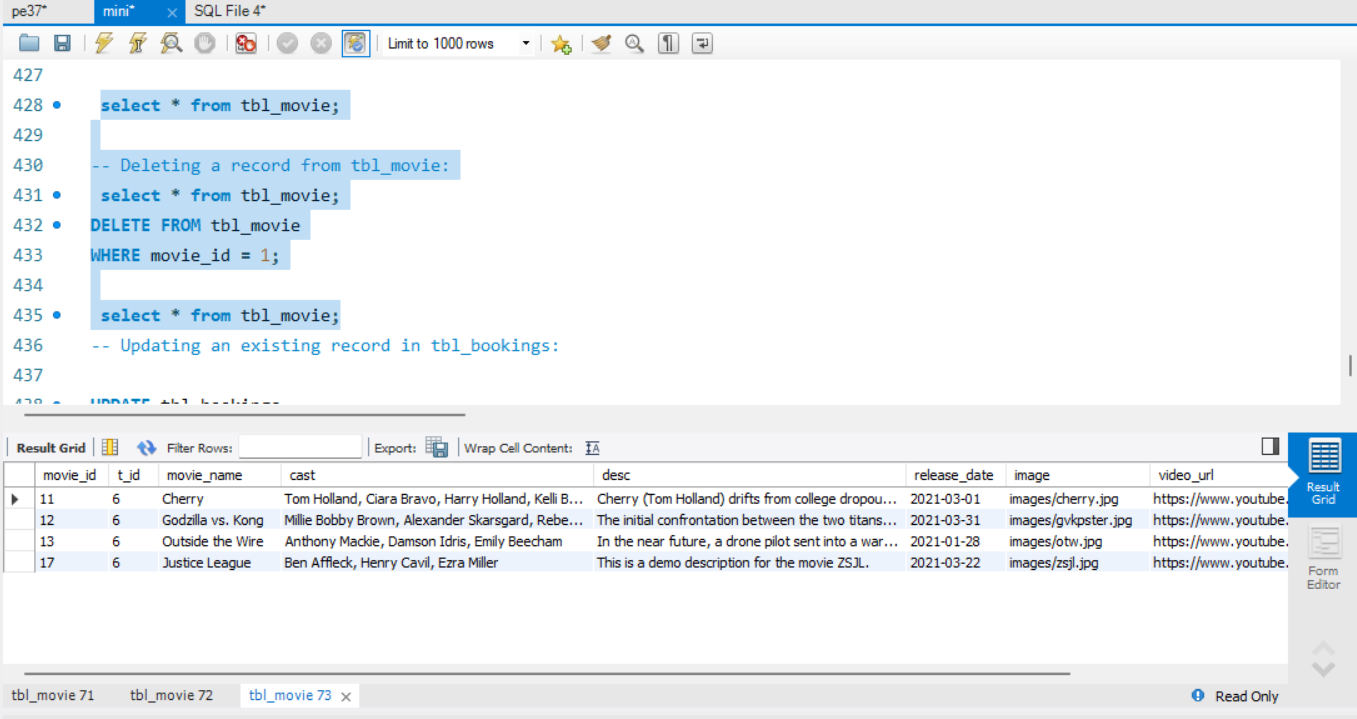
DML queries:



—Deleting a record from tbl\_movie:

DELETE FROM tbl\_movie

WHERE movie\_id = 1;

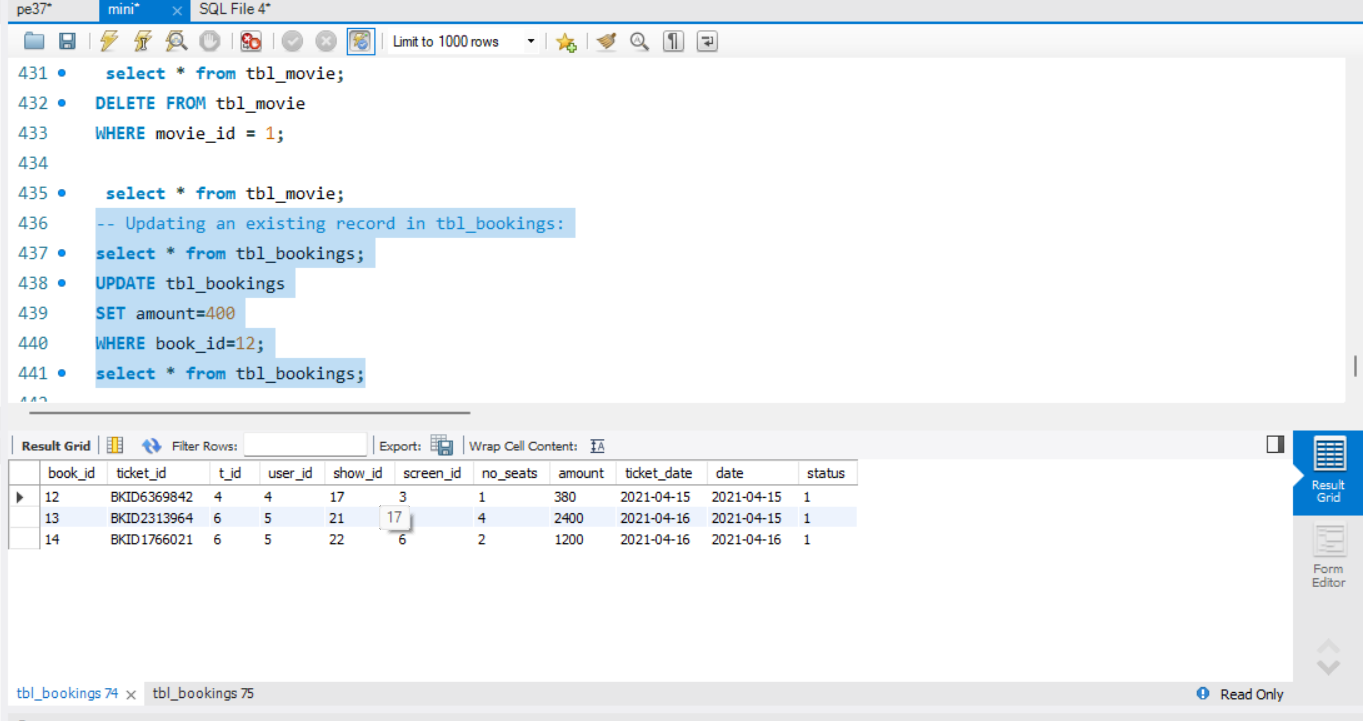


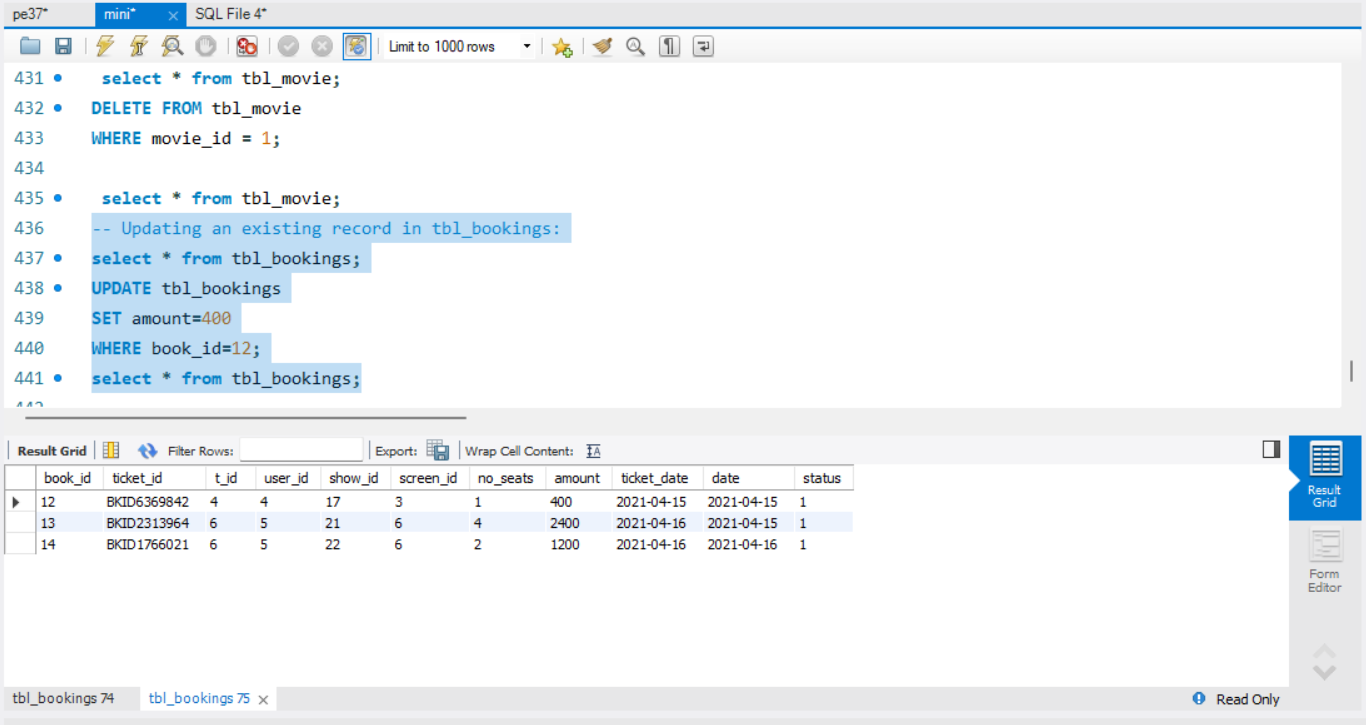
--Updating an existing record in tbl\_bookings:

UPDATE tbl\_bookings

SET amount=400

WHERE book\_id=12;



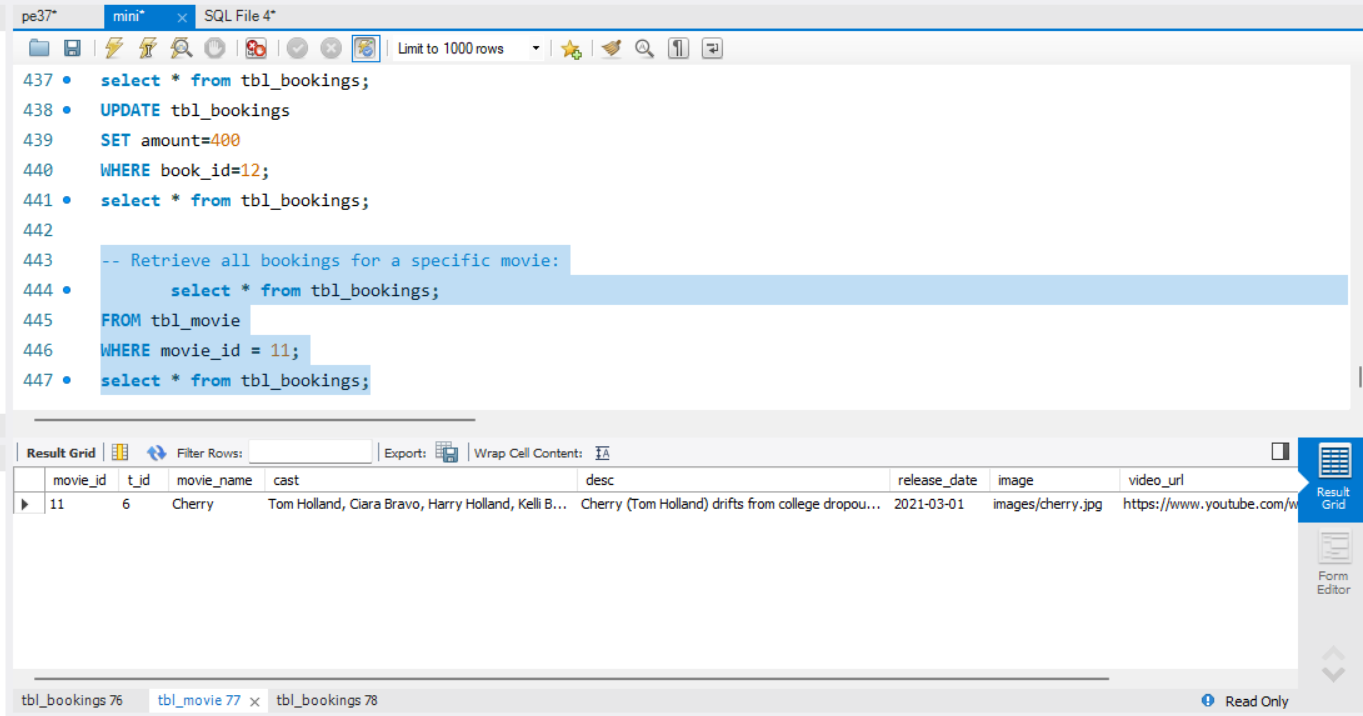


--Retrieve all bookings for a specific movie:

SELECT \*

FROM tbl\_movie

WHERE movie\_id = 11



Triggers:

DELIMITER //

CREATE TRIGGER after\_booking\_insert

AFTER INSERT ON Bookings

FOR EACH ROW

BEGIN

-- Update the number of available seats for the corresponding show and screen

UPDATE Shows

SET available\_seats = available\_seats - 1

WHERE show\_id = NEW.show\_id;

-- Check if all seats are booked for the corresponding show

IF (SELECT available\_seats FROM Shows WHERE show\_id = NEW.show\_id) = 0 THEN

-- Perform some action when all seats are booked (e.g., send a notification)

-- REPLACE WITH YOUR CUSTOM ACTION

INSERT INTO Notifications (message)

VALUES ('All seats booked for show ' || NEW.show\_id);

END IF;

DELIMITER ;

PLSQL Procedure/Function:

Procedure to insert a new movie into the database:

mysql> DELIMITER //

mysql>

mysql> CREATE PROCEDURE InsertNewMovie(

IN p\_movie\_name VARCHAR(255),

IN p\_movie\_desc VARCHAR(255),

IN p\_movie\_genre VARCHAR(255),

IN p\_movie\_duration INT,

IN p\_website\_name VARCHAR(255),

IN p\_website\_url VARCHAR(255),

IN p\_screen\_name VARCHAR(255),

IN p\_show\_date DATE,

IN p\_show\_time TIME,

IN p\_seat\_rows INT,

IN p\_seat\_nums INT

)

BEGIN

-- Insert into Movies table

INSERT INTO Movies (movie\_name, movie\_desc, movie\_genre, movie\_duration)

VALUES (p\_movie\_name, p\_movie\_desc, p\_movie\_genre, p\_movie\_duration);

SET @v\_movie\_id := LAST\_INSERT\_ID();

-- Insert into Movie\_Website table if website information is provided

IF p\_website\_name IS NOT NULL AND p\_website\_url IS NOT NULL THEN

INSERT INTO Movie\_Website (website\_name, website\_url, movie\_id)

VALUES (p\_website\_name, p\_website\_url, @v\_movie\_id);

END IF;

-- Insert into Screens table if screen information is provided

IF p\_screen\_name IS NOT NULL THEN

INSERT INTO Screens (screen\_name)

VALUES (p\_screen\_name);

SET @v\_screen\_id := LAST\_INSERT\_ID();

-- Insert into Shows table if show information is provided

IF p\_show\_date IS NOT NULL AND p\_show\_time IS NOT NULL THEN

INSERT INTO Shows (show\_date, show\_time, screen\_id, movie\_id)

VALUES (p\_show\_date, p\_show\_time, @v\_screen\_id, @v\_movie\_id);

END IF;

END IF;

-- Insert into Seats table if seat information is provided

IF p\_seat\_rows IS NOT NULL AND p\_seat\_nums IS NOT NULL THEN

IF @v\_screen\_id IS NULL THEN

INSERT INTO Screens (screen\_name)

VALUES (NULL);

SET @v\_screen\_id := LAST\_INSERT\_ID();

END IF;

INSERT INTO Seats (seat\_row, seat\_num, screen\_id)

VALUES (p\_seat\_rows, p\_seat\_nums, @v\_screen\_id);

END IF;

END //

Query OK, 0 rows affected (0.01 sec)

mysql>

mysql> DELIMITER ;

mysql> CALL InsertNewMovie(

'The New Movie',

'A description of the new movie',

'Action, Drama',

120,

'Movie Website',

'https://www.example.com/movie',

'Screen 4',

'2023-05-20',

'18:00:00',

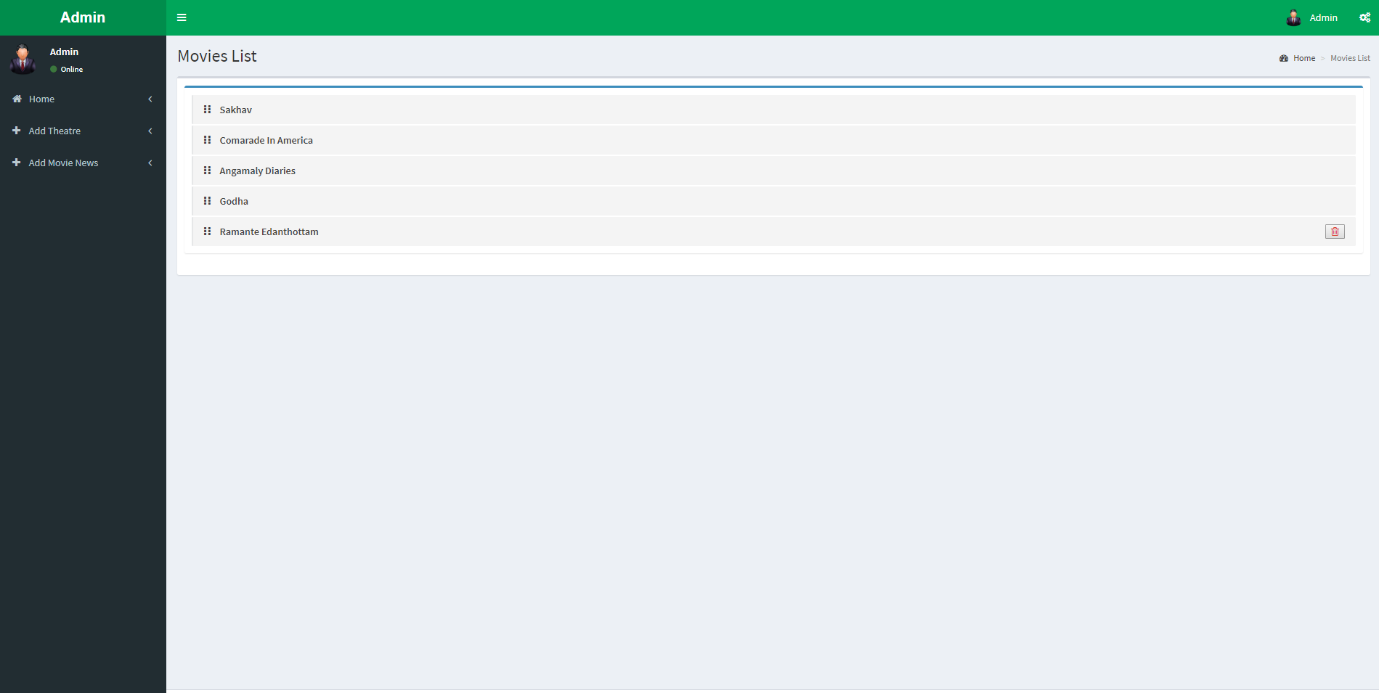
1,

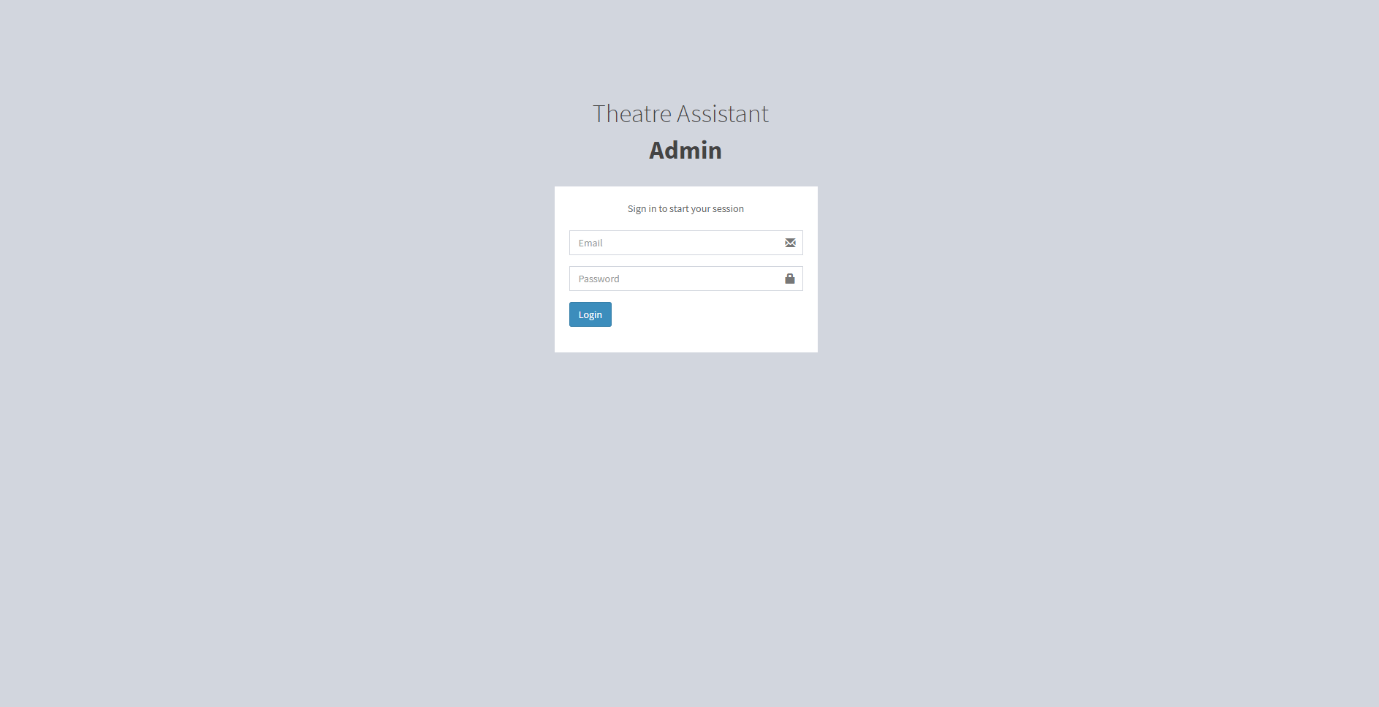
1

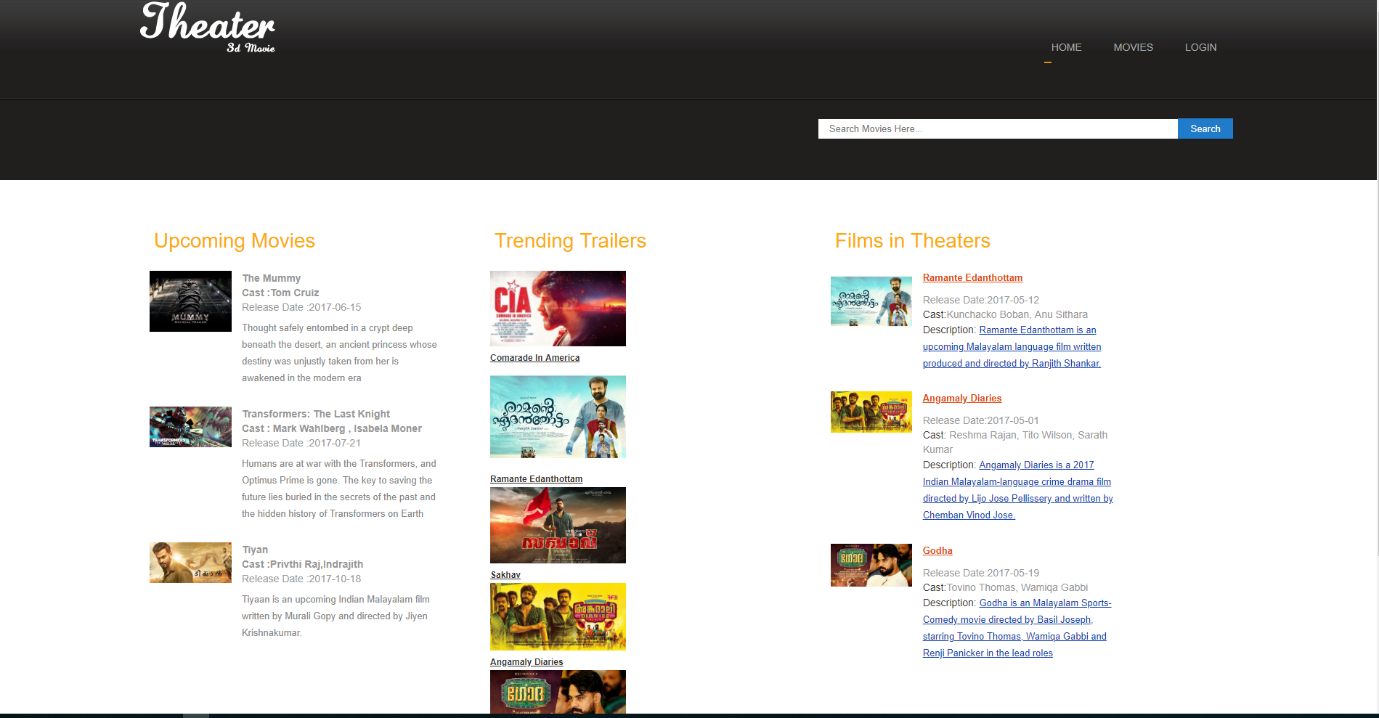
);

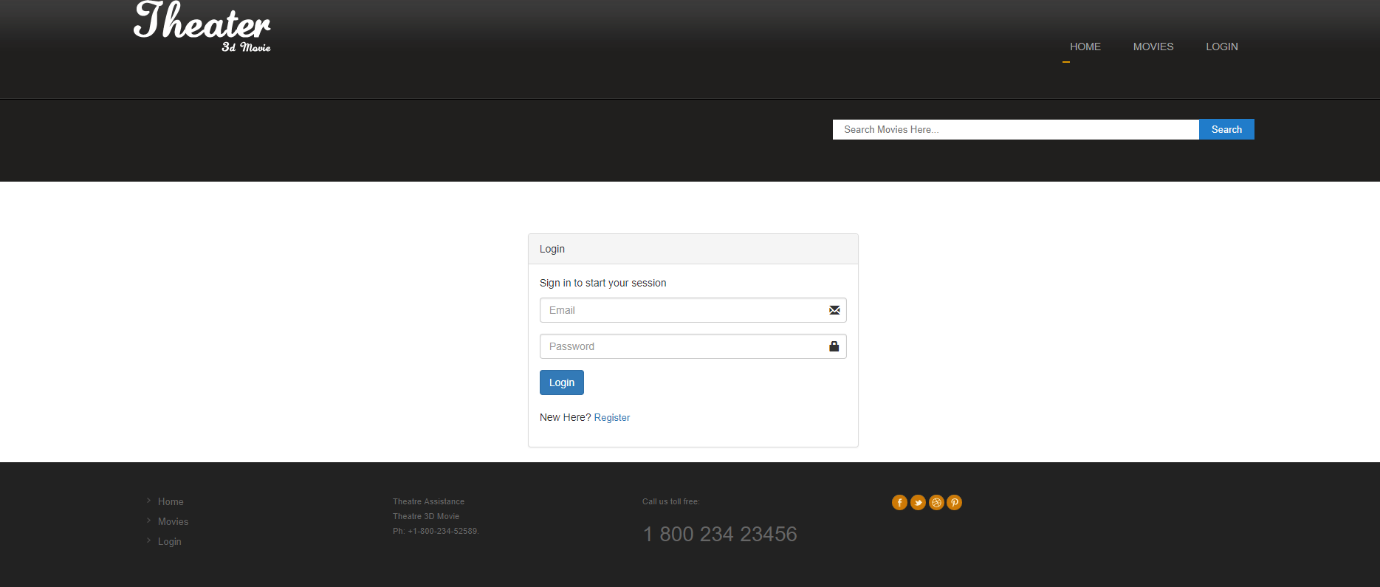
Query OK, 1 row affected (0.04 sec)

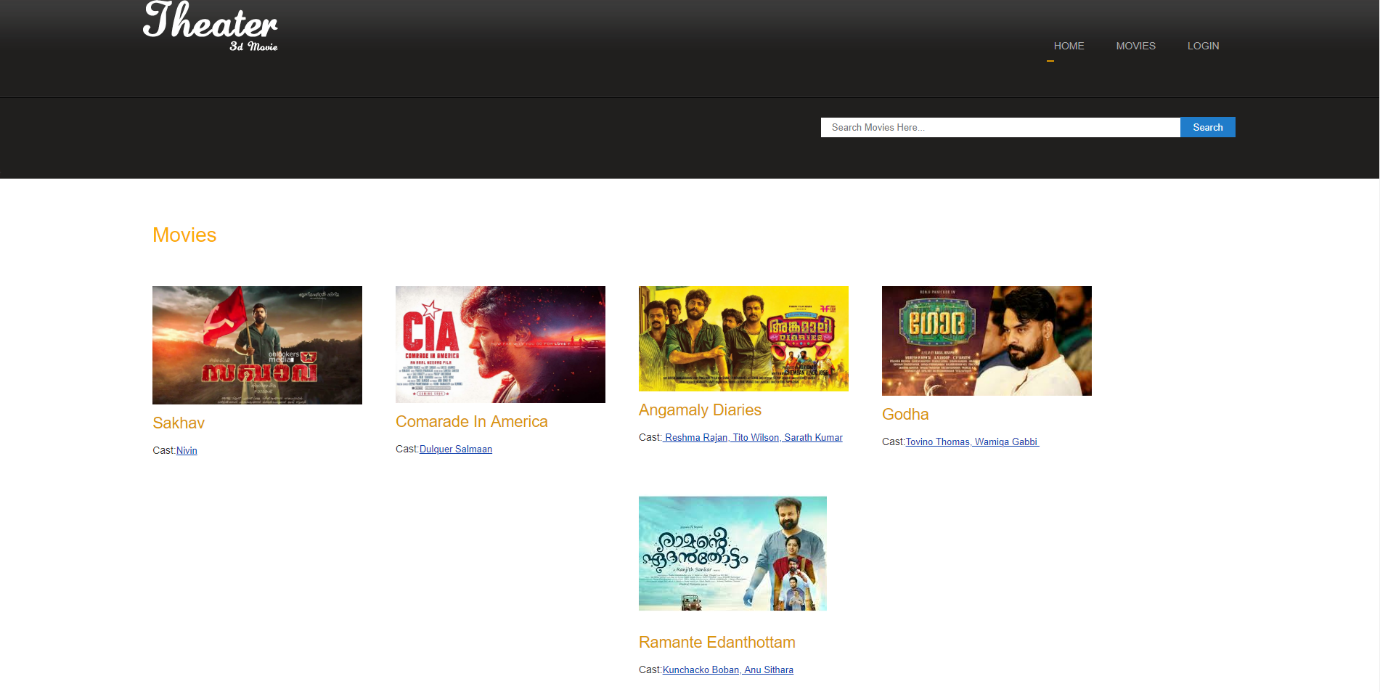
Frontend GUI screenshot:

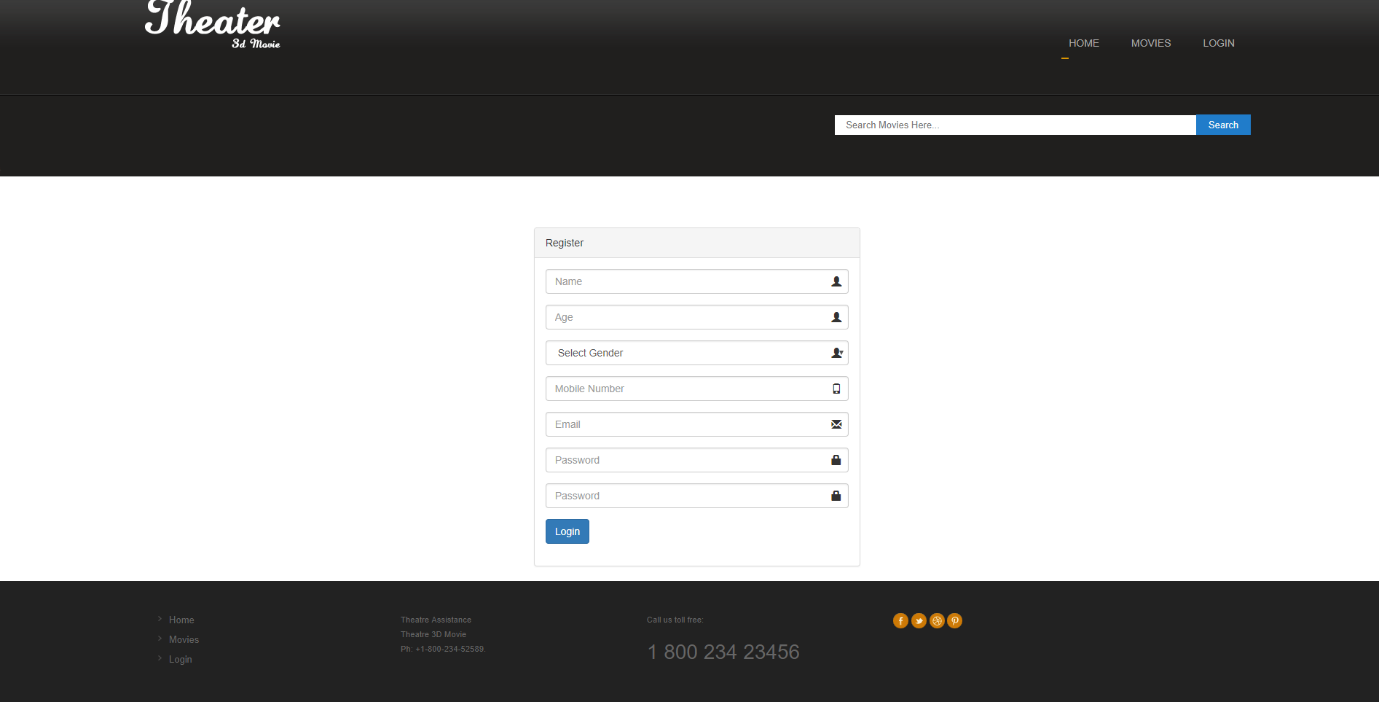


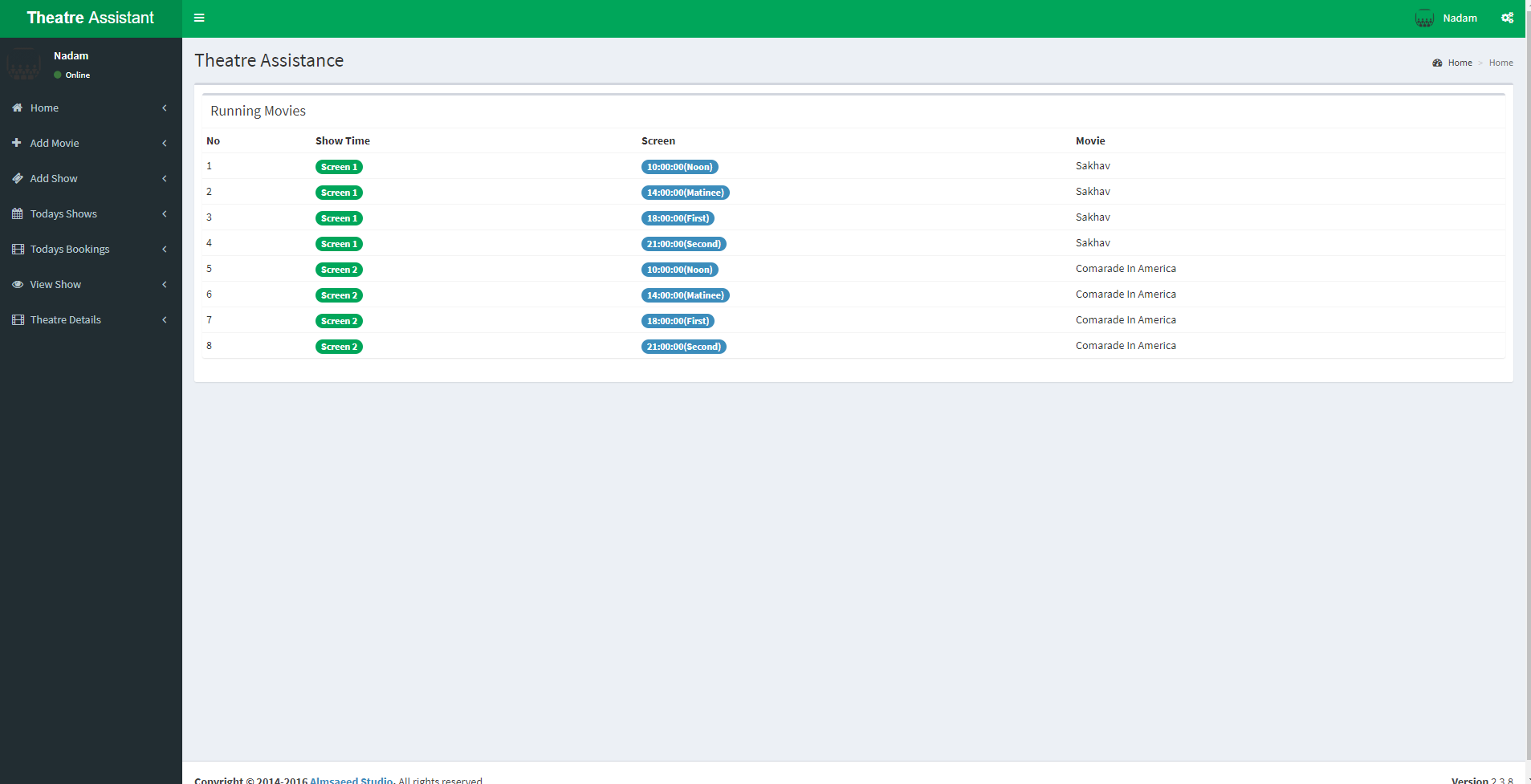












Conclusion: In conclusion, an online movie ticket management system revolutionizes the industry by offering convenient booking, personalized experiences, improved operational efficiency, and increased revenue opportunities for theatres. It enhances the moviegoing experience for customers and contributes to the overall growth and success of the industry.

References in IEEE format:

Kumar, A., Sharma, R., & Gupta, A. (2018). Design and Implementation of Online Movie Ticket Booking System. In 2018 3rd International Conference on Communication and Electronics Systems (ICCES) (pp. 930-935). IEEE.