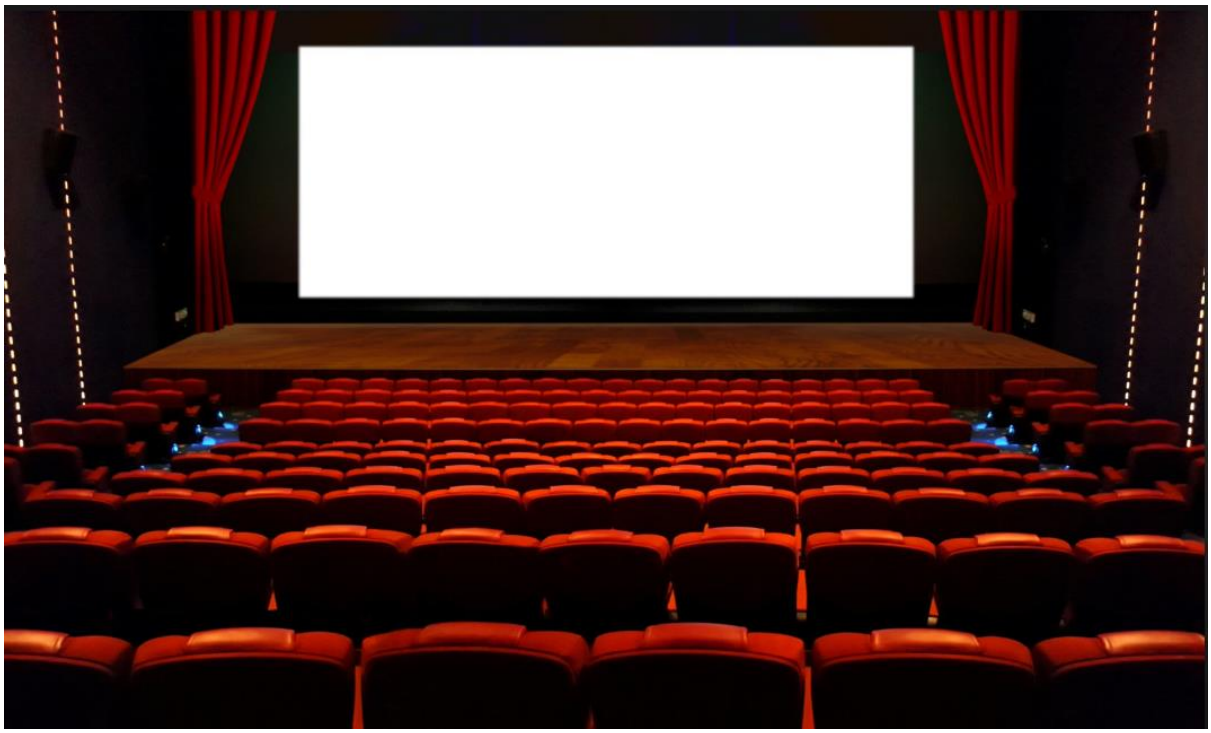
MOVIE TICKET BOOKING MANAGEMENT SYSTEM



SUBMITED BY : ARTHI.K

COURSE : MYSQL

TABLE OF CONTENT

1.INTRODUCTION:

- **Project Overview:** The Movie Ticket Booking Management System is a project designed to help users book movie tickets easily. It allows users to view available movies, check showtimes, and reserve seats at different Theaters. The system is built using MySQL to manage movie details, show schedules, ticket availability, and bookings efficiently.

- **Technology used:** MySQL was chosen as the database platform.

2.OBJECTIVES:

1. To design a database that stores movie details, theater information, show schedules, and seat availability.
2. To allow users to book tickets for movies by selecting available shows and seats.
3. To implement stored procedures for managing bookings and updating available seats in real time.
4. To ensure data integrity by handling cancellations and modifying bookings through database triggers.
5. To create an efficient and user-friendly system for managing movie ticket bookings.

3.ENTITIES:

There are 5 entities in this database, listed as follows:

- **Movies:**

  MovieId(Primary Key), Title, Genre, Duration, ReleaseDate, Director.

- **Theatres:**

  TheaterID(Primary Key), T_name, Location, TotalSeats.

- **Shows:**

  ShowID(Primary Key), MovieID(Foreign Key), TheaterID(Foreign Key), ShowDate, ShowTime, AvailableSeats.

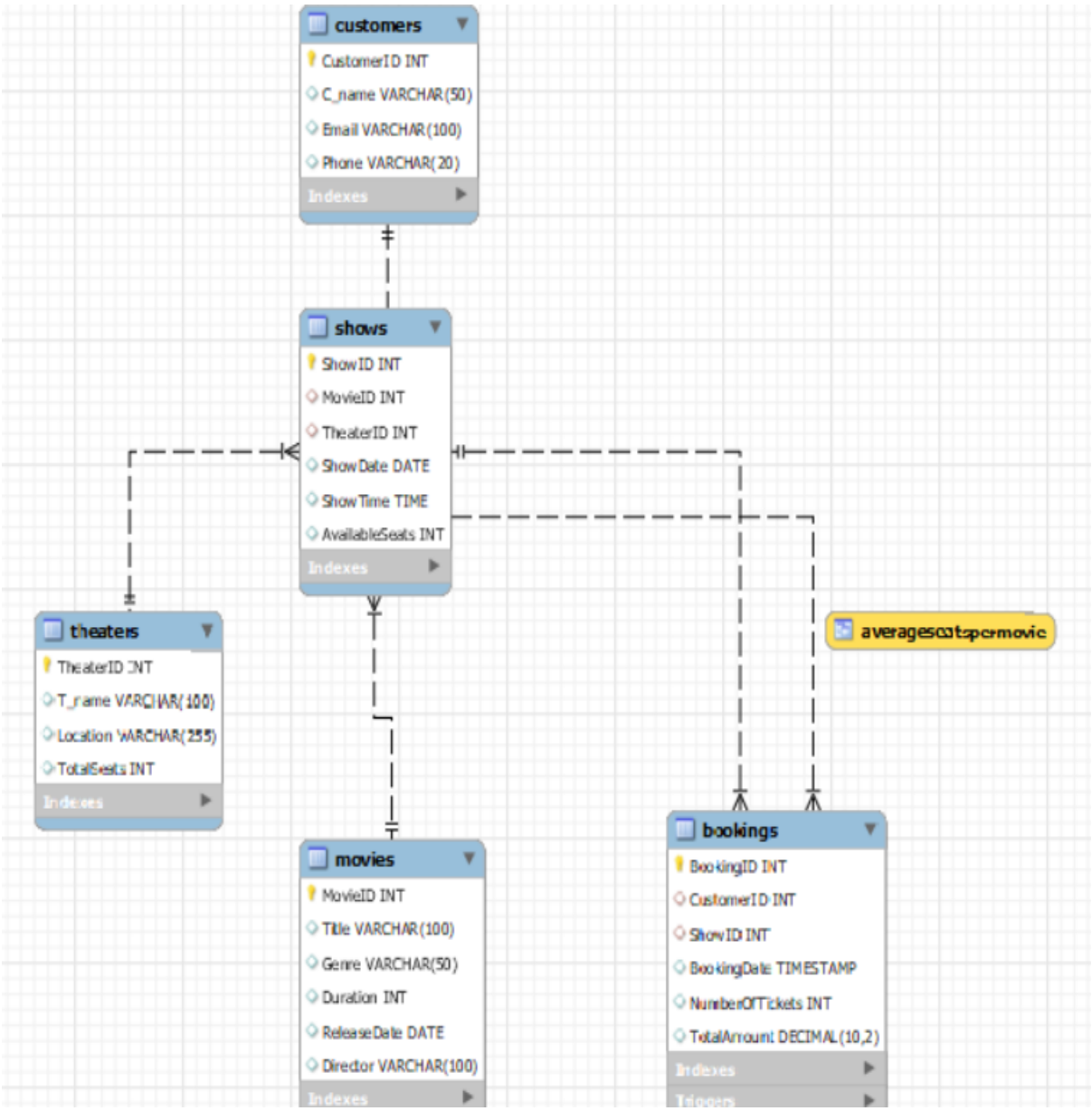
- **Customers:**

  CustomerID(Primary Key), C_name, Email(Unique), Phone.

- **Bookings:**

  BookingID(Primary Key), CustomerID(Foreign Key), ShowID(Foreign Key), BookingDate(Default), NumberOfTickets, TotalAmount.

4.EER DIAGRAM:



5.DATABASE CREATION:

First, we need to create a database to stores a tables.

- create database project1;
- use project1;

6.TABLE CREATION:

Table 1: Movies

- create table Movies (

  MovieID INT AUTO_INCREMENT PRIMARY KEY,

  Title VARCHAR(100),

  Genre VARCHAR(50),

  Duration INT,

  ReleaseDate DATE,

  Director VARCHAR(100));

Table 2: Theaters

- create table Theaters (

  TheaterID INT AUTO_INCREMENT PRIMARY KEY,

  T_name VARCHAR(100),

  Location VARCHAR(255),

  TotalSeats INT);

Table 3: Shows

- create table Shows (

  ShowID INT AUTO_INCREMENT PRIMARY KEY,

  MovieID INT,

  TheaterID INT,

  ShowDate DATE,

  ShowTime TIME,

AvailableSeats INT,

FOREIGN KEY (MovieID) REFERENCES Movies(MoviesID),

FOREIGN KEY (TheaterID) REFERENCES Theaters(TheaterID));

Table 4: Customers

- create table Customers (

  CustomerID INT AUTO_INCREMENT PRIMARY KEY,

  C_name VARCHAR(50),

  Email VARCHAR(100) UNIQUE,

  Phone VARCHAR(20));

Table 5: Bookings

- create table Bookings (

  BookingID INT AUTO_INCREMENT PRIMARY KEY,

  CustomerID INT,

  ShowID INT,

  BookingDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  NumberOfTickets INT,

  TotalAmount DECIMAL(10, 2),

  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

  FOREIGN KEY (ShowID) REFERENCES Shows(ShowID));

7.TRIGGER CREATION:

A trigger named amount is created to automatically calculate the total booking amount before inserting a new booking. The total amount is determined by multiplying the number of tickets by a fixed price (200.00).

- delimiter &&

  create trigger amount

  before insert on Bookings

  for each row

  begin

  set new.TotalAmount=new.NumberOfTickets*200.00;

  end &&

  delimiter ;

8.DATA INSERTION:

Table 1:

- insert into Movies (Title, Genre, Duration, ReleaseDate, Director) values

  ('Ghilli', 'Action', 165, '2004-04-09', 'Dharani'),

  ('Thuppakki', 'Action', 153, '2012-11-13', 'A.R. Murugadoss'),

  ('Mersal', 'Action', 174, '2017-10-18', 'Atlee'),

  ('Master', 'Action', 179, '2021-01-13', 'Lokesh Kanagaraj'),

  ('Sarkar', 'Action', 164, '2018-11-06', 'A.R. Murugadoss'),

  ('Kavalan', 'Romantic Comedy', 139, '2011-01-15', 'R. Kannan'),

  ('Bigil', 'Sports Drama', 173, '2019-11-09', 'Atlee');

Table 2:

- insert into Theaters (T_name, Location, TotalSeats) values

  ('PVR Cinemas', 'Chennai', 250),

  ('INOX', 'Chennai', 200),

  ('Sathyam Cinemas', 'Chennai', 300),

  ('Cinepolis', 'Chennai', 180),

  ('Escape Cinemas', 'Chennai', 220),

  ('Carnival Cinemas', 'Chennai', 150),

  ('Eros International', 'Chennai', 170);

Table 3:

- insert into Shows (MovieID, TheaterID, ShowDate, ShowTime, AvailableSeats) values

  (1, 1, '2024-09-10', '18:00:00', 250),

  (2, 2, '2024-09-10', '20:00:00', 200),

  (3, 3, '2024-09-11', '21:00:00', 300),

  (4, 4, '2024-09-12', '19:00:00', 180),

  (5, 5, '2024-09-13', '20:30:00', 220),

  (6, 6, '2024-09-14', '18:30:00', 150),

  (7, 7, '2024-09-15', '21:15:00', 170);

Table 4:

- insert into Customers (C_name,Email, Phone) values

  ('Kumar', 'kumar04@gmail.com', '9876543210'),

  ('Anjali', 'anjali13@gmail.com', '8765432109'),

  ('Arthi', 'arthi2004@gmail.com', '7654321098'),

  ('Naren', 'naren20@gmail.com', '6543210987'),

  ('Dharani', 'dharani15@.gmail.com','5432109876'),

  ('Divya', 'divya26@gmail.com', '4321098765'),

  ('Sri','sri132005@gmail.com','3210987654');

Table 5:

- insert into Bookings (CustomerID, ShowID, NumberOfTickets) values

  (1, 1, 2),

  (2, 2, 1),

  (3, 3, 3),

  (4, 4, 2),

  (5, 5, 4),

  (6, 6, 1),

  (7, 7, 1);

9.TABLES:

Table 1:

MovieID	Title	Genre	Duration	ReleaseDate	Director
1	Ghilli	Action	165	2004-04-09	Dharani
2	Thuppakki	Action	153	2012-11-13	A.R. Murugadoss
3	Mersal	Action	174	2017-10-18	Atlee
4	Master	Action	179	2021-01-13	Lokesh Kanagaraj
5	Sarkar	Action	164	2018-11-06	A.R. Murugadoss
6	Kavalan	Romantic Comedy	139	2011-01-15	R. Kannan
7	Bigil	Sports Drama	173	2019-11-09	Atlee

Table 2:

TheaterID	T_name	Location	TotalSeats
1	PVR Cinemas	Chennai	250
2	INOX	Chennai	200
3	Sathyam Cinemas	Chennai	300
4	Cinepolis	Chennai	180
5	Escape Cinemas	Chennai	220
6	Carnival Cinemas	Chennai	150
7	Eros International	Chennai	170

Table 3:

ShowID	MovieID	TheaterID	ShowDate	ShowTime	AvailableSeats
1	1	1	2024-09-10	18:00:00	250
2	2	2	2024-09-10	20:00:00	200
3	3	3	2024-09-11	21:00:00	300
4	4	4	2024-09-12	19:00:00	180
5	5	5	2024-09-13	20:30:00	220
6	6	6	2024-09-14	18:30:00	150
7	7	7	2024-09-15	21:15:00	170

Table 4:

CustomerID	C_name	Email	Phone
1	Kumar	kumar04@gmail.com	9876543210
2	Anjali	anjali13@gmail.com	8765432109
3	Arthi	arthi2004@gmail.com	7654321098
4	Naren	naren20@gmail.com	6543210987
5	Dharani	dharani15@.gmail.com	5432109876
6	Divya	divya26@gmail.com	4321098765
7	Sri	sri132005@gmail.com	3210987654

Table 5:

BookingID	CustomerID	ShowID	BookingDate	NumberOfTickets	TotalAmount
1	1	1	2024-09-07 18:19:48	2	400.00
2	2	2	2024-09-07 18:19:48	1	200.00
3	3	3	2024-09-07 18:19:48	3	600.00
4	4	4	2024-09-07 18:19:48	2	400.00
5	5	5	2024-09-07 18:19:48	4	800.00
6	6	6	2024-09-07 18:19:48	1	200.00
7	7	7	2024-09-07 18:19:48	1	200.00

10.QUERIES:

1. **FIND MOVIES BY A SPECIFIC DIRECTOR:**

```
SELECT * FROM Movies WHERE Director='Atlee';
```

- **Explanation:** This query retrieves all the columns (*) from the Movies table where the Director column matches the name 'Atlee'.

➢ **OUTPUT:**

MovieID	Title	Genre	Duration	ReleaseDate	Director
3	Mersal	Action	174	2017-10-18	Atlee
7	Bigil	Sports Drama	173	2019-11-09	Atlee

2. **GET THEATERS IN A SPECIFIC LOCATION WITH GREATER THAN 220 SEATS:**

```
SELECT * FROM Theaters WHERE Location='Chennai' AND
TotalSeats>220;
```

- **Explanation:** This query selects all columns from the Theaters table where the Location is 'Chennai' and the TotalSeats are greater than 220.

➢ **OUTPUT:**

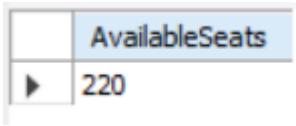
TheaterID	T_name	Location	TotalSeats
1	PVR Cinemas	Chennai	250
3	Sathyam Cinemas	Chennai	300

3. **GET AVAILABLE SEATS FOR A SPECIFIC SHOW:**

```
SELECT AvailableSeats FROM Shows WHERE ShowID = 5;
```

- **Explanation:** This query fetches the AvailableSeats for the show with ShowID 5 from the Shows table. It provides the number of available seats for that particular show.

➢ **OUTPUT:**



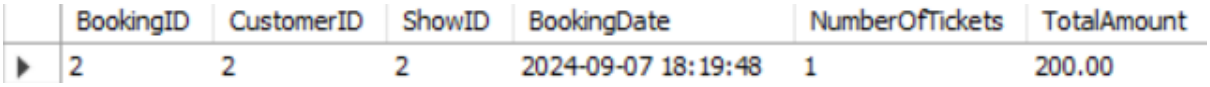
	AvailableSeats
▶	220

4. GET BOOKING DETAILS FOR A SPECIFIC CUSTOMER:

```
SELECT * FROM Bookings WHERE CustomerID = 2;
```

- **Explanation:** This query retrieves all booking details from the Bookings table where the CustomerID is 2. It returns the booking records for the specified customer.

➢ **OUTPUT:**



	BookingID	CustomerID	ShowID	BookingDate	NumberOfTickets	TotalAmount
▶	2	2	2	2024-09-07 18:19:48	1	200.00

5. GET MOVIES SORTED ASCENDING BY THE "RELEASEDATE" COLUMN:

```
SELECT Title, Director, ReleaseDate FROM Movies ORDER BY
ReleaseDate;
```

- **Explanation:** This query selects the Title, Director, and ReleaseDate columns from the Movies table and sorts the results in ascending order based on the ReleaseDate column. It organizes movies from the earliest to the most recent release.

➢ **OUTPUT:**

Title	Director	ReleaseDate
Ghilli	Dharani	2004-04-09
Kavalan	R. Kannan	2011-01-15
Thuppakki	A.R. Murugadoss	2012-11-13
Mersal	Atlee	2017-10-18
Sarkar	A.R. Murugadoss	2018-11-06
Bigil	Atlee	2019-11-09
Master	Lokesh Kanagaraj	2021-01-13

6. GET THEATERS SORTED DESCENDING BY THE "TOTALSEATS" COLUMN:

```
SELECT T_name, TotalSeats FROM Theaters ORDER BY
TotalSeats DESC;
```

- **Explanation:** This query selects the T_name (theater name) and TotalSeats from the Theaters table, sorting the results in descending order of TotalSeats. It displays theaters starting from the one with the most seats.

➢ **OUTPUT:**

T_name	TotalSeats
Sathyam Cinemas	300
PVR Cinemas	250
Escape Cinemas	220
INOX	200
Cinepolis	180
Eros International	170
Carnival Cinemas	150

7. **GET CUSTOMERS, RETURNING ONLY 3 RECORDS STARTING ON RECORD 2:**

```
SELECT * FROM Customers ORDER BY C_name LIMIT 3
OFFSET 1;
```

- **Explanation:** This query selects all columns from the Customers table and orders the results by C_name (customer name). It limits the result set to 3 records but skips the first record using the OFFSET 1 clause. It starts from the second record and fetches three results.

➢ **OUTPUT:**

CustomerID	C_name	Email	Phone
3	Arthi	arthi2004@gmail.com	7654321098
5	Dharani	dharani15@.gmail.com	5432109876
6	Divya	divya26@gmail.com	4321098765

8. **GET MAXIMUM AND MINIMUM MOVIE DURATION:**

```
SELECT MAX(Duration), MIN(Duration) FROM Movies;
```

- **Explanation:** This query calculates the maximum and minimum values of the Duration column in the Movies table. It returns the longest and shortest movie durations.

➢ **OUTPUT:**

max(duration)	min(duration)
179	139

9. FIND THE HIGHEST AND LOWEST BOOKING AMOUNT:

```
SELECT MAX(TotalAmount) AS 'Highest amount',
MIN(TotalAmount) AS 'Lowest amount' FROM Bookings;
```

- **Explanation:** This query calculates the highest and lowest values from the TotalAmount column in the Bookings table. The results are returned with the alias 'Highest amount' and 'Lowest amount'.

➢ **OUTPUT:**

	Highest amount	lowest amount
▶	800.00	200.00

10. LIST THE NUMBER OF MOVIES IN EACH GENRE:

```
SELECT Genre, COUNT(Genre) AS 'Number of Movies' FROM
Movies GROUP BY Genre;
```

- **Explanation:** This query groups the movies by their Genre and counts the number of movies for each genre using the COUNT() function. The result displays the genre and the total number of movies per genre.

➢ **OUTPUT:**

	Genre	Number of Movies
▶	Action	5
	Romantic Comedy	1
	Sports Drama	1

11. LIST THE NUMBER OF MOVIES FOR EACH DIRECTOR:

```
SELECT Director, COUNT(Title) AS 'Number of Movies' FROM
Movies GROUP BY Director;
```

- **Explanation:** This query groups the movies by the Director column and counts the number of movies directed by each director. It shows the director's name and the total number of movies they directed.

➢ **OUTPUT:**

Director	Number of Movies
Dharani	1
A.R. Murugadoss	2
Atlee	2
Lokesh Kanagaraj	1
R. Kannan	1

12. GET TOTAL REVENUE PER SHOW:

```
SELECT ShowID, SUM(TotalAmount) AS TotalRevenue FROM
Bookings GROUP BY ShowID;
```

- **Explanation:** This query calculates the total revenue generated for each show by summing the TotalAmount column in the Bookings table. The results are grouped by ShowID, so each show's total revenue is displayed.

➢ **OUTPUT:**

	ShowID	TotalRevenue
▶	1	400.00
	2	200.00
	3	600.00
	4	400.00
	5	800.00
	6	200.00
	7	200.00

13.GET CUSTOMERS SORTED DESCENDING BY THE "NUMBER OF TICKETS" COLUMN:

```
SELECT CustomerID, SUM(NumberOfTickets) AS TotalTickets
FROM Bookings
GROUP BY CustomerID
ORDER BY TotalTickets DESC;
```

- **Explanation:** This query calculates the total number of tickets booked by each customer by summing the NumberOfTickets column in the Bookings table. It groups the results by CustomerID and sorts the customers in descending order based on the total tickets booked.

➢ **OUTPUT:**

	CustomerID	TotalTickets
▶	5	4
	3	3
	1	2
	4	2
	2	1
	6	1
	7	1

14. GET SHOWS WITH LOW AVAILABILITY (SEATS < 160):

```
SELECT ShowID, AvailableSeats FROM Shows WHERE AvailableSeats < 160;
```

- **Explanation:** This query selects the ShowID and AvailableSeats from the Shows table where the number of available seats is less than 160. It identifies shows with low seat availability.

➢ **OUTPUT:**

ShowID	AvailableSeats
6	150

15. SELECT ALL MOVIES WITH A TITLE STARTING WITH LETTER "M":

```
SELECT * FROM Movies WHERE Title LIKE 'M%';
```

- **Explanation:** This query selects all columns from the Movies table where the Title starts with the letter 'M'. The LIKE 'M%' clause filters movie titles that begin with 'M'.

➢ **OUTPUT:**

MovieID	Title	Genre	Duration	ReleaseDate	Director
3	Mersal	Action	174	2017-10-18	Atlee
4	Master	Action	179	2021-01-13	Lokesh Kanagaraj

16. SELECT ALL MOVIES WITH A DIRECTOR STARTING WITH ANY LETTER FOLLOWED BY "TLEE":

```
SELECT * FROM Movies WHERE Director LIKE '_tlee';
```

- **Explanation:** This query selects all columns from the Movies table where the Director name follows the pattern of any first character followed by "tlee". The _ in LIKE '_tlee' represents a single character wildcard.

➢ **OUTPUT:**

MovieID	Title	Genre	Duration	ReleaseDate	Director
3	Mersal	Action	174	2017-10-18	Atlee
7	Bigil	Sports Drama	173	2019-11-09	Atlee

17. UPDATE AVAILABLE SEATS AFTER BOOKING:

```
UPDATE Shows SET AvailableSeats = AvailableSeats - 2 WHERE ShowID = 3;
```

- **Explanation:** This query updates the AvailableSeats in the Shows table by subtracting 2 from the current available seats for the show with ShowID = 3. It adjusts the seat availability after booking tickets.

➢ **OUTPUT:**

BEFORE:

ShowID	MovieID	TheaterID	ShowDate	ShowTime	AvailableSeats
1	1	1	2024-09-10	18:00:00	250
2	2	2	2024-09-10	20:00:00	200
3	3	3	2024-09-11	21:00:00	300
4	4	4	2024-09-12	19:00:00	180
5	5	5	2024-09-13	20:30:00	220
6	6	6	2024-09-14	18:30:00	150
7	7	7	2024-09-15	21:15:00	170

AFTER:

ShowID	MovieID	TheaterID	ShowDate	ShowTime	AvailableSeats
1	1	1	2024-09-10	18:00:00	250
2	2	2	2024-09-10	20:00:00	200
3	3	3	2024-09-11	21:00:00	298
4	4	4	2024-09-12	19:00:00	180
5	5	5	2024-09-13	20:30:00	220
6	6	6	2024-09-14	18:30:00	150
7	7	7	2024-09-15	21:15:00	170

18. UPDATE CUSTOMER CONTACT INFORMATION:

```
UPDATE Customers SET Email = 'sri202004@gmail.com' WHERE
CustomerID = 7;
```

- **Explanation:** This query updates the Email address for the customer with CustomerID = 7. It changes the existing email to 'sri202004@gmail.com'.

➢ **OUTPUT:**

BEFORE:

CustomerID	C_name	Email	Phone
1	Kumar	kumar04@gmail.com	9876543210
2	Anjali	anjali13@gmail.com	8765432109
3	Arthi	arthi2004@gmail.com	7654321098
4	Naren	naren20@gmail.com	6543210987
5	Dharani	dharani15@.gmail.com	5432109876
6	Divya	divya26@gmail.com	4321098765
7	Sri	sri132005@gmail.com	3210987654

AFTER:

CustomerID	C_name	Email	Phone
1	Kumar	kumar04@gmail.com	9876543210
2	Anjali	anjali13@gmail.com	8765432109
3	Arthi	arthi2004@gmail.com	7654321098
4	Naren	naren20@gmail.com	6543210987
5	Dharani	dharani15@.gmail.com	5432109876
6	Divya	divya26@gmail.com	4321098765
7	Sri	sri202004@gmail.com	3210987654

19. FIND THE SHOW DATE, SHOW TIME, AND AVAILABLE SEATS FOR A PARTICULAR MOVIE:

```
SELECT ShowDate, ShowTime, AvailableSeats
FROM Shows
WHERE MovieID = (SELECT MovieID FROM Movies WHERE
Title = 'Master');
```

- **Explanation:** This query retrieves the ShowDate, ShowTime, and AvailableSeats from the Shows table for the movie titled 'Master'. The subquery first fetches the MovieID from the Movies table where the title is 'Master'.

➢ **OUTPUT:**

	ShowDate	ShowTime	AvailableSeats
▶	2024-09-12	19:00:00	180

20. DISPLAY MOVIE TITLE, THEATER NAME, SHOW DATE, AND SHOW TIME:

```
SELECT Movies.Title, Theaters.T_name, Shows.ShowDate,
Shows.ShowTime
FROM Shows
INNER JOIN Movies ON Movies.MovieID = Shows.MovieID
INNER JOIN Theaters ON Theaters.TheaterID = Shows.TheaterID;
```

- **Explanation:** This query uses inner joins to retrieve the Title from the Movies table, T_name (theater name) from the Theaters table, and ShowDate and ShowTime from the Shows table. It combines the data from all three tables to display show information for each movie and theater.

➢ **OUTPUT:**

Title	T_name	ShowDate	ShowTime
Ghilli	PVR Cinemas	2024-09-10	18:00:00
Thuppakki	INOX	2024-09-10	20:00:00
Mersal	Sathyam Cinemas	2024-09-11	21:00:00
Master	Cinepolis	2024-09-12	19:00:00
Sarkar	Escape Cinemas	2024-09-13	20:30:00
Kavalan	Carnival Cinemas	2024-09-14	18:30:00
Bigil	Eros International	2024-09-15	21:15:00

21. DISPLAY AVAILABLE SEATS FOR A SPECIFIC MOVIE:

```
SELECT Movies.Title, Theaters.T_Name, Shows.ShowDate,
Shows.ShowTime, Shows.AvailableSeats
FROM Shows
INNER JOIN Movies ON Shows.MovieID = Movies.MovieID
INNER JOIN Theaters ON Shows.TheaterID = Theaters.TheaterID
WHERE Movies.Title = 'Master';
```

- **Explanation:** This query selects the Title from the Movies table, T_Name (theater name) from the Theaters table, and ShowDate, ShowTime, and AvailableSeats from the Shows table. It filters the results to only show information for the movie titled 'Master'.

➢ **OUTPUT:**

	Title	T_Name	ShowDate	ShowTime	AvailableSeats
▶	Master	Cinepolis	2024-09-12	19:00:00	180

22. DISPLAY MOVIES AND THEIR AVERAGE AVAILABLE SEATS ACROSS ALL THEATERS:

```
CREATE VIEW AverageSeatsPerMovie AS
SELECT Movies.Title, AVG(Shows.AvailableSeats) AS AverageSeats
FROM Shows
INNER JOIN Movies ON Shows.MovieID = Movies.MovieID
GROUP BY Movies.Title;
```

- **Explanation:** This query creates a view AverageSeatsPerMovie that calculates the average available seats for each movie. It joins the Movies and Shows tables and groups the results by the movie title, displaying the average.

```
SELECT * FROM AverageSeatsPerMovie;
```

➢ **OUTPUT:**

	Title	AverageSeats
▶	Ghilli	250.0000
	Thuppakki	200.0000
	Mersal	298.0000
	Master	180.0000
	Sarkar	220.0000
	Kavalan	150.0000
	Bigil	170.0000

23. DISPLAYS THE TOTAL NUMBER OF BOOKINGS FOR EACH MOVIE.

```
CREATE VIEW MovieBookingCounts AS
SELECT Movies.Title, COUNT(Bookings.BookingID) AS
TotalBookings
FROM Movies
LEFT JOIN Shows ON Movies.MovieID = Shows.MovieID
LEFT JOIN Bookings ON Shows.ShowID = Bookings.ShowID
GROUP BY Movies.Title;
```

- **Explanation:**This view aggregates the number of bookings for each movie by joining the Movies, Shows, and Bookings tables. It groups the results by movie title and shows the total number of bookings.

```
SELECT * FROM MovieBookingCounts;
```

➢ **OUTPUT:**

Title	TotalBookings
Ghilli	1
Thuppakki	1
Mersal	1
Master	1
Sarkar	0
Kavalan	1
Bigil	1

24. DISPLAY MOVIE TITLE AND A MINIMUM NUMBER OF AVAILABLE SEATS USING PROCEDURE:

```
DELIMITER //

CREATE PROCEDURE mov(
IN movieTitle VARCHAR(100),
IN minSeats INT
)
BEGIN
SELECT ShowDate, ShowTime, AvailableSeats
FROM Shows
WHERE MovieID = (SELECT MovieID FROM Movies WHERE
Title = movieTitle)
AND AvailableSeats > minSeats;
END //

DELIMITER ;
```

- **Explanation:** This stored procedure allows users to retrieve show details (date, time, and available seats) for a specific movie based on two input parameters: the movie title and a minimum number of available seats.

    - **Input Parameters:**
        - movieTitle (VARCHAR): The title of the movie.
        - minSeats (INT): Minimum available seats required.

- **Logic:** The procedure selects the show date, time, and available seats from the Shows table where the MovieID matches the provided movie title and the available seats exceed the specified minimum.

```
CALL mov('Ghilli', 50);
```

➢ **OUTPUT:**

	ShowDate	ShowTime	AvailableSeats
▶	2024-09-10	18:00:00	250

25. TO DELETE A SPECIFIC SHOW FROM THE SHOWS TABLE:

```
DELETE FROM Shows WHERE ShowID = 5;
```

- **Explanation:** This command deletes the record from the Shows table where ShowID is 5. The WHERE clause ensures that only the specific show with ShowID 5 is removed, preventing the deletion of all records in the table.

- **OUTPUT:**

BEFORE:

ShowID	MovieID	TheaterID	ShowDate	ShowTime	AvailableSeats
1	1	1	2024-09-10	18:00:00	250
2	2	2	2024-09-10	20:00:00	200
3	3	3	2024-09-11	21:00:00	300
4	4	4	2024-09-12	19:00:00	180
5	5	5	2024-09-13	20:30:00	220
6	6	6	2024-09-14	18:30:00	150
7	7	7	2024-09-15	21:15:00	170

AFTER:

ShowID	MovieID	TheaterID	ShowDate	ShowTime	AvailableSeats
1	1	1	2024-09-10	18:00:00	250
2	2	2	2024-09-10	20:00:00	200
3	3	3	2024-09-11	21:00:00	298
4	4	4	2024-09-12	19:00:00	180
6	6	6	2024-09-14	18:30:00	150
7	7	7	2024-09-15	21:15:00	170