E - TICKET BOOKING DATABASE

ID	NAME	SECTION
2303256	Mennatallah Eslam Ahmed	SQL
2301870	Ashraqat Mahmoud Attia	Mapping
2402724	Taghreed Amr Esmail	ERD
2401171	Nourhan Shrief Abodaif	Report

Introduction

With the rapid development of technology and the widespread use of the internet, electronic ticket booking systems have become an essential part of modern life. These systems allow users to book tickets for events such as concerts, movies, sports matches, and conferences easily and securely without the need to visit physical ticket offices.

The E-Ticket Booking Database System is designed to manage all operations related to online ticket reservations efficiently. The system stores and organizes data about users, events, bookings, tickets, and payments in a structured database. This helps ensure data accuracy, reduces redundancy, and improves overall system performance.

By using a well-designed database and Entity Relationship Diagram (ERD), the system provides clear relationships between entities and supports smooth booking, payment processing, and ticket validation. The main goal of this project is to create a reliable, secure, and scalable database system that meets the needs of both users and event organizers.

ERD

E-Ticket Booking Database System

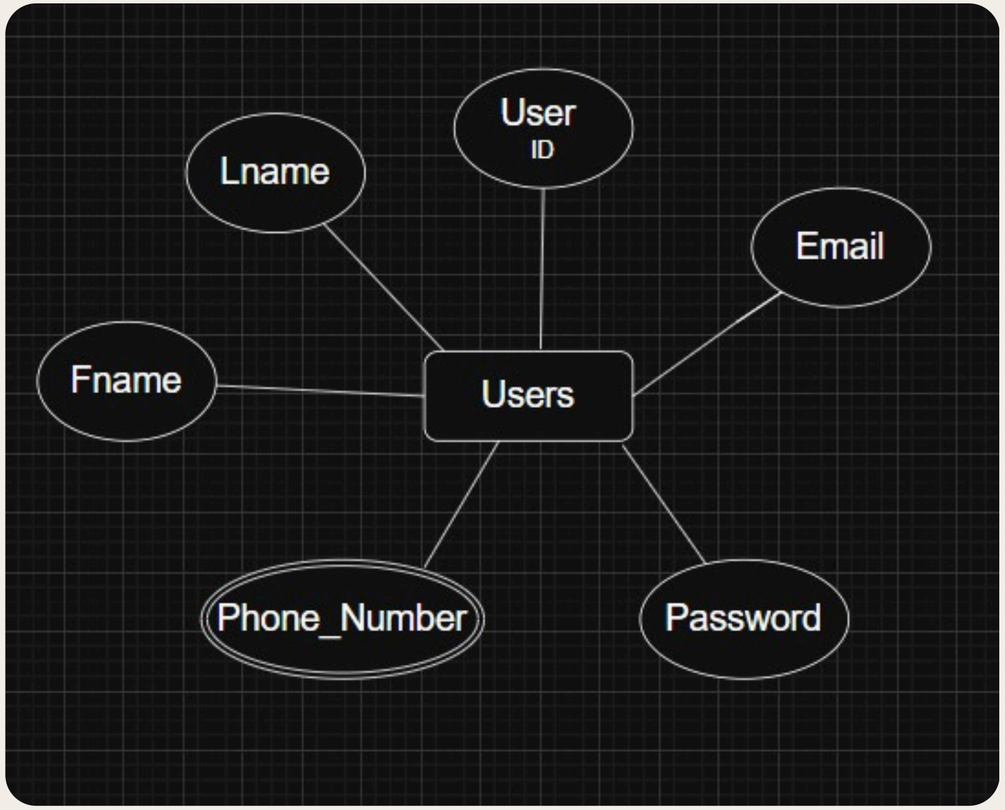
E-Ticket Booking Database System

This system manages online ticket booking

, users, events, payments, and e-tickets efficiently.

Explanation of the Users Entity

The Users entity represents all registered users in the e-ticket booking system.
This part of the ERD shows the attributes related to a user.



Name (Composite Attribute)
The name is divided into:
-Fname → First name
-Lname → Last name
-User ID
-Primary key of the user
Uniquely identifies each user in the system
-Email
User’s email address (unique)
-Phone Number
User’s phone number
Multi-valued attribute (a user may have more than one phone number)
-Password
Used for user authentication

Explanation of the Event Entity

The Event entity represents all events available for booking.

-Event ID

Primary key of the event

-Event Name

Name of the event (Concert, Movie, Match, etc.)

-Event Date

Date when the event takes place

-Event Time

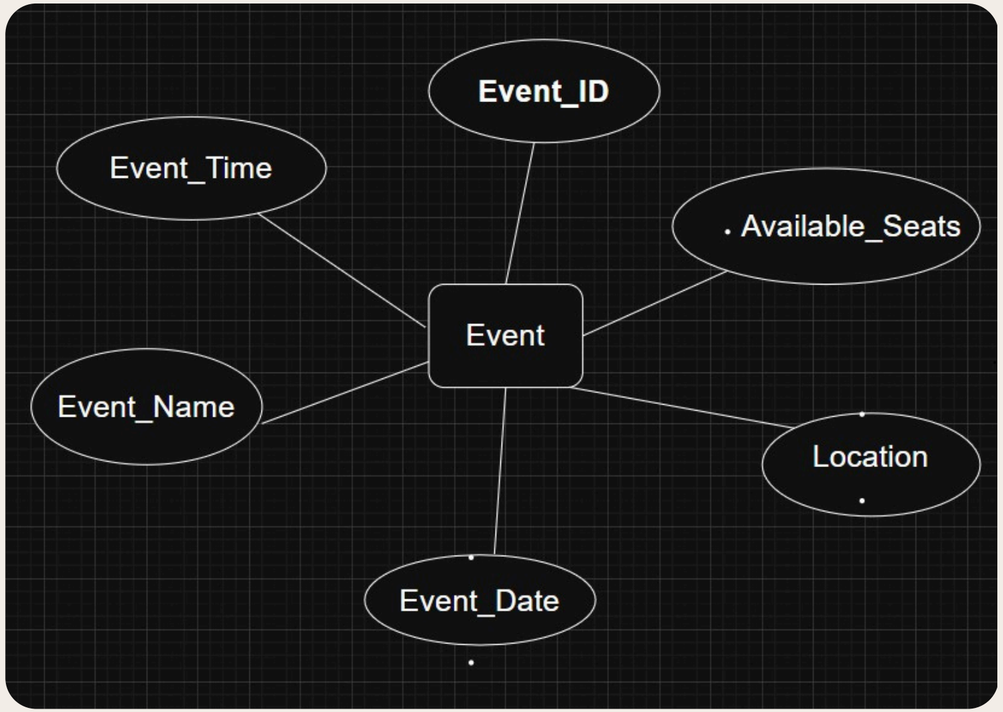
Time of the event

-Location

Place where the event is held

-Available Seats

Number of seats available for booking



Explanation of the Ticket Entity

The Ticket entity represents the tickets booked by users.

-Ticket ID

Primary key of the ticket

-Seat Number

Seat assigned to the ticket

-Price

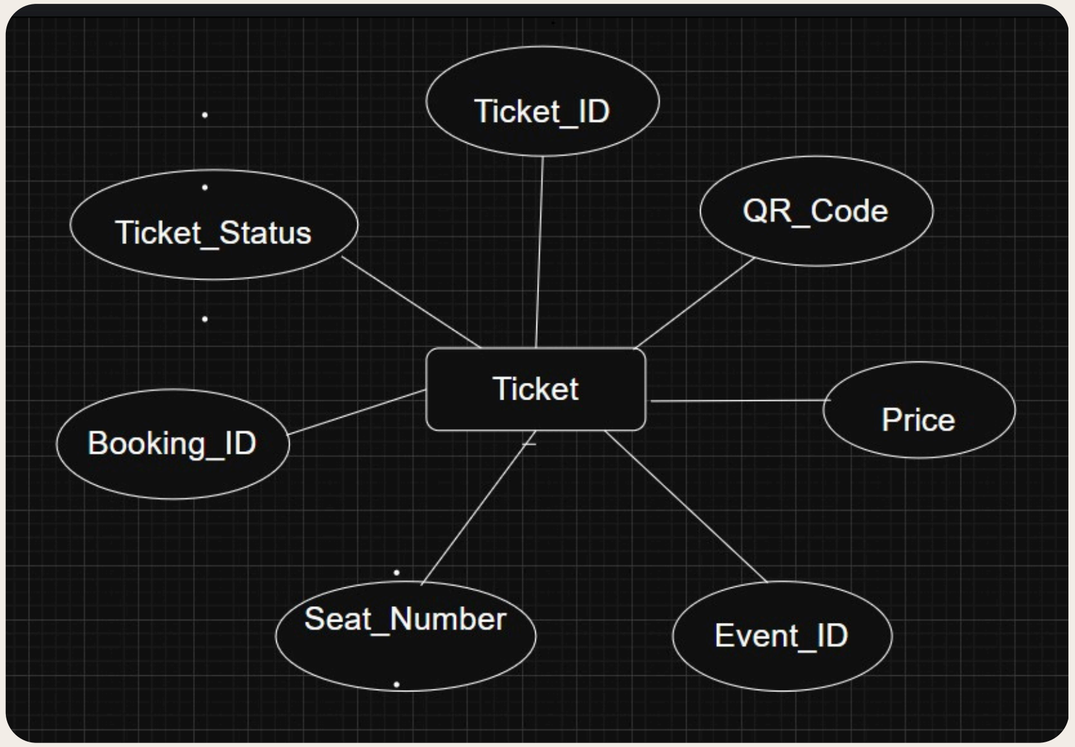
Ticket price

-Ticket Status

Booked / Cancelled / Used

-QR Code

Used for ticket validation at entry



Explanation of the Booking Entity

The Booking entity represents the booking process between users and tickets.

-Booking ID

Primary key of the booking

-Booking Date

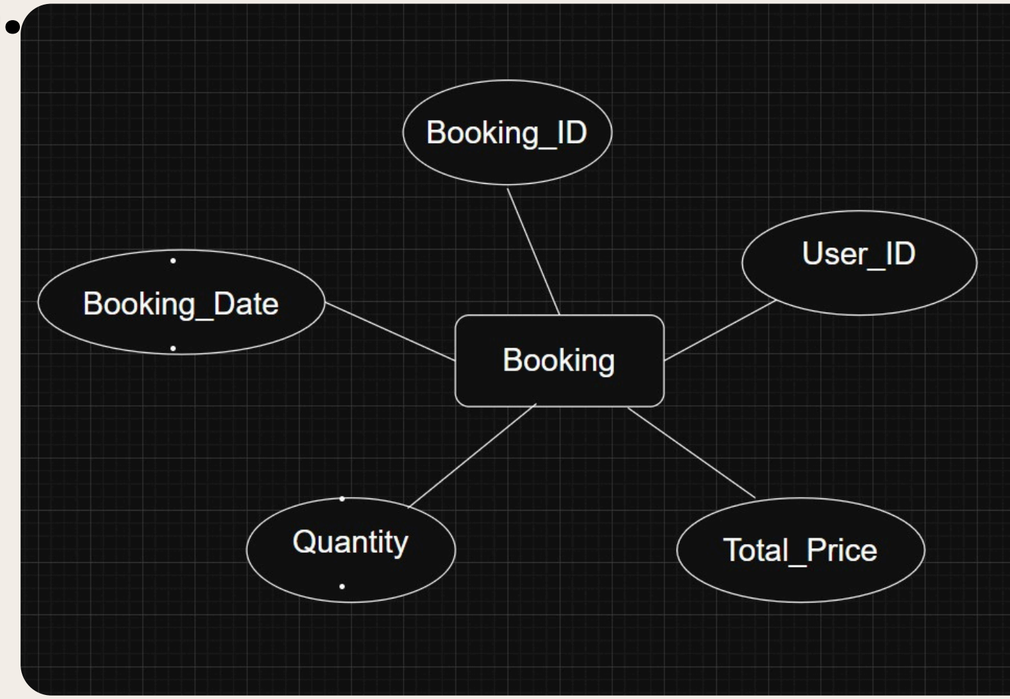
Date when the booking was made

-Quantity

Number of tickets booked

-Total Price

Total cost of the booking



Explanation of the Payment Entity

The Payment entity stores payment transaction details.

-Payment ID

Primary key of the payment

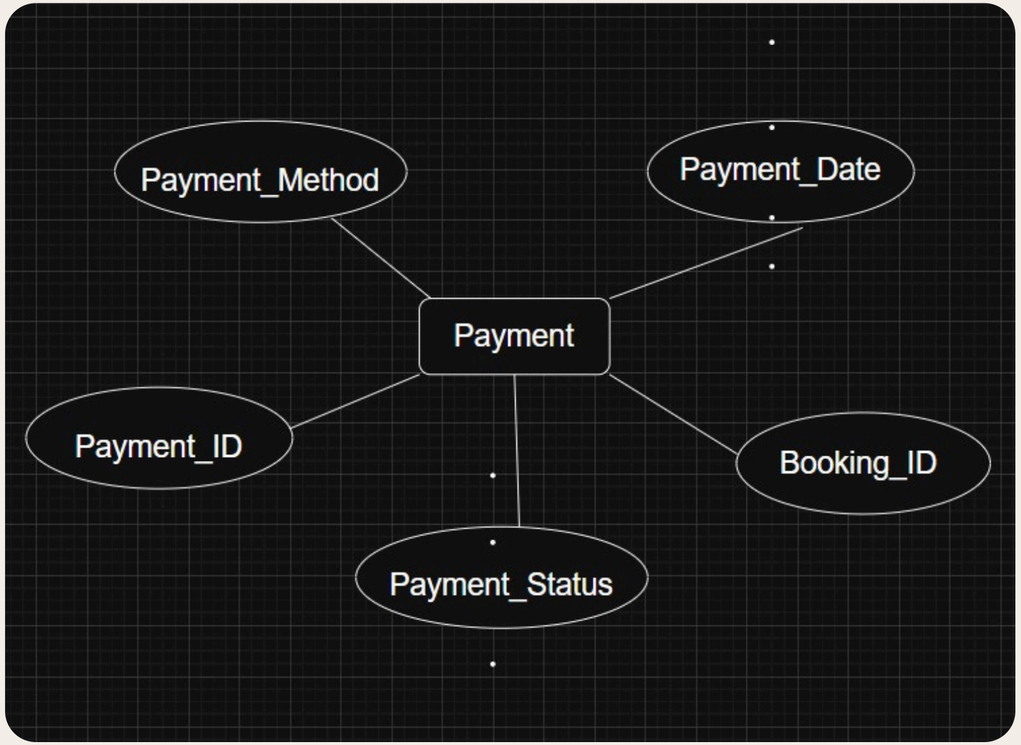
-Payment Method

Credit Card / Wallet / Cash

-Payment Date

Date of payment

-Payment Status



Relationships

Book Relationship

Connects Users – Booking

-A user can make multiple bookings

-Each booking belongs to one user

-Contains Relationship

-Connects Booking – Ticket

-A booking can contain multiple tickets

-Each ticket belongs to one booking

-Has Relationship

-Connects Event – Ticket

-An event has many tickets

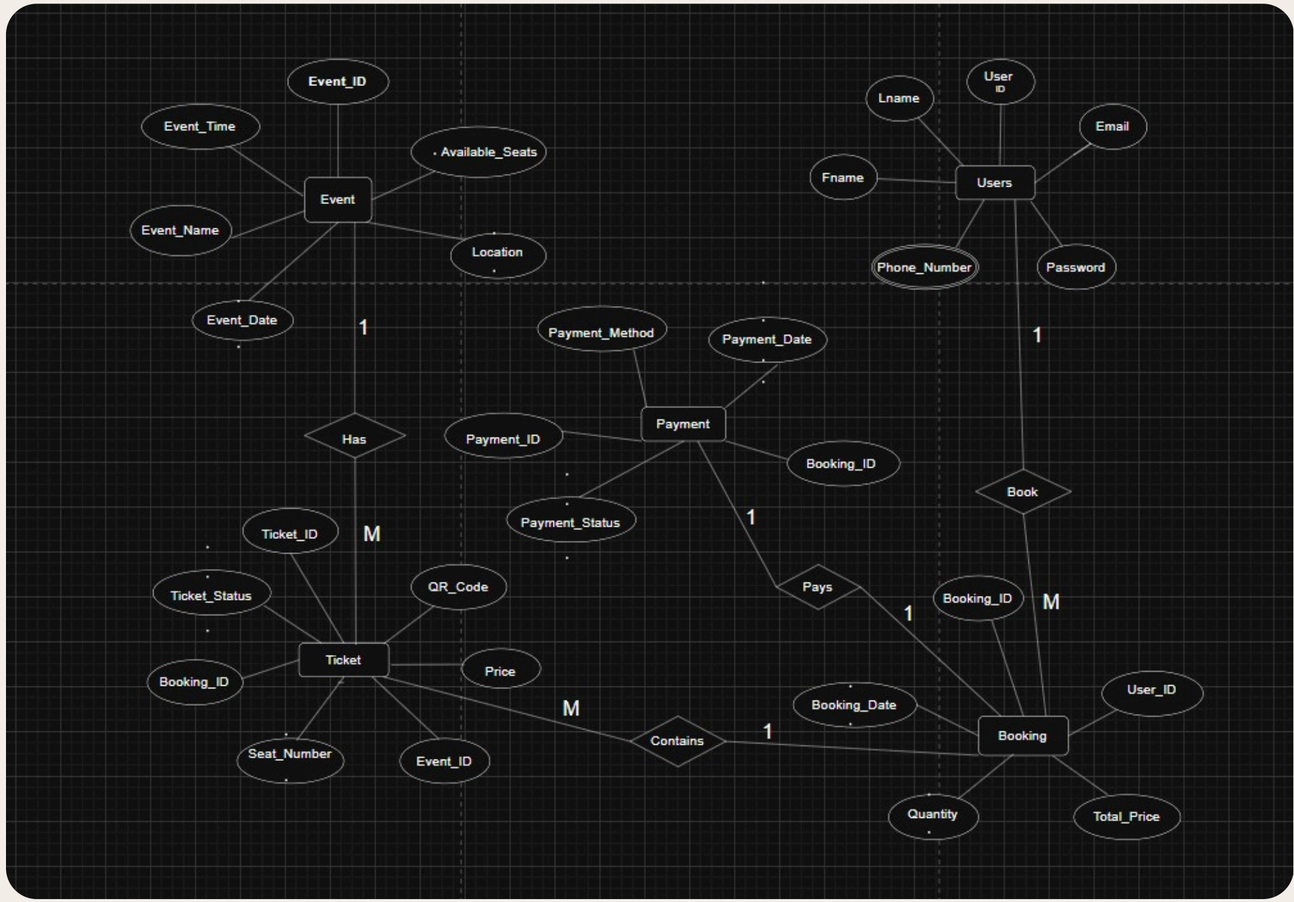
-Each ticket is related to one event

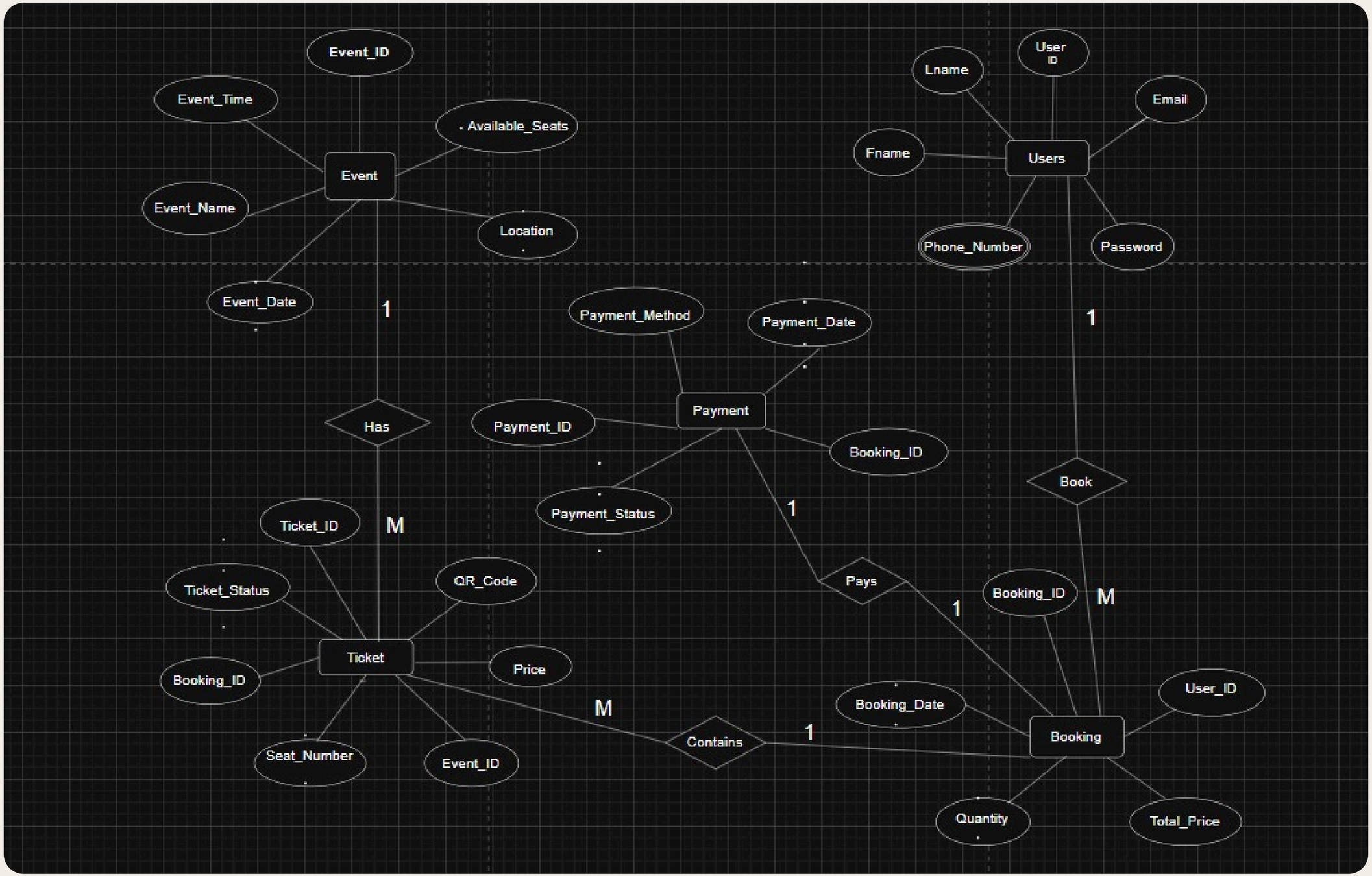
-Pays Relationship

-Connects Booking – Payment

-Each booking has one payment

-Each payment is for one booking




Event_ID
Event_Time
Available_Seats
Event
Event_Name
Location
Event_Date
1
Has
M
Ticket_ID
Ticket_Status
QR_Code
Ticket
Booking_ID
Price
Seat_Number
Event_ID
M
Contains
1
Payment_Method
Payment_Date
Payment
Payment_ID
Booking_ID
Payment_Status
1
Pays
1
User ID
Lname
Email
Fname
Users
Phone_Number
Password
1
Book
Booking_ID
M
Booking_Date
User_ID
Booking
Quantity
Total_Price

MAPPING

ERD to Relational Schema Mapping

E-Ticket Booking System

-Introduction

This report explains the mapping of the ERD of an E-Ticket Booking System into a relational database schema using standard mapping rules.

-USERS

The Users table stores user information. Each user is identified by User_ID (PK) and can make multiple bookings.

-EVENT

The Event table stores event details such as name, date, time, and location. Each event is identified by Event ID (PK).

-BOOKING

The Booking table represents reservations made by users. It contains Booking_ID (PK) and User_ID (FK).

-TICKET

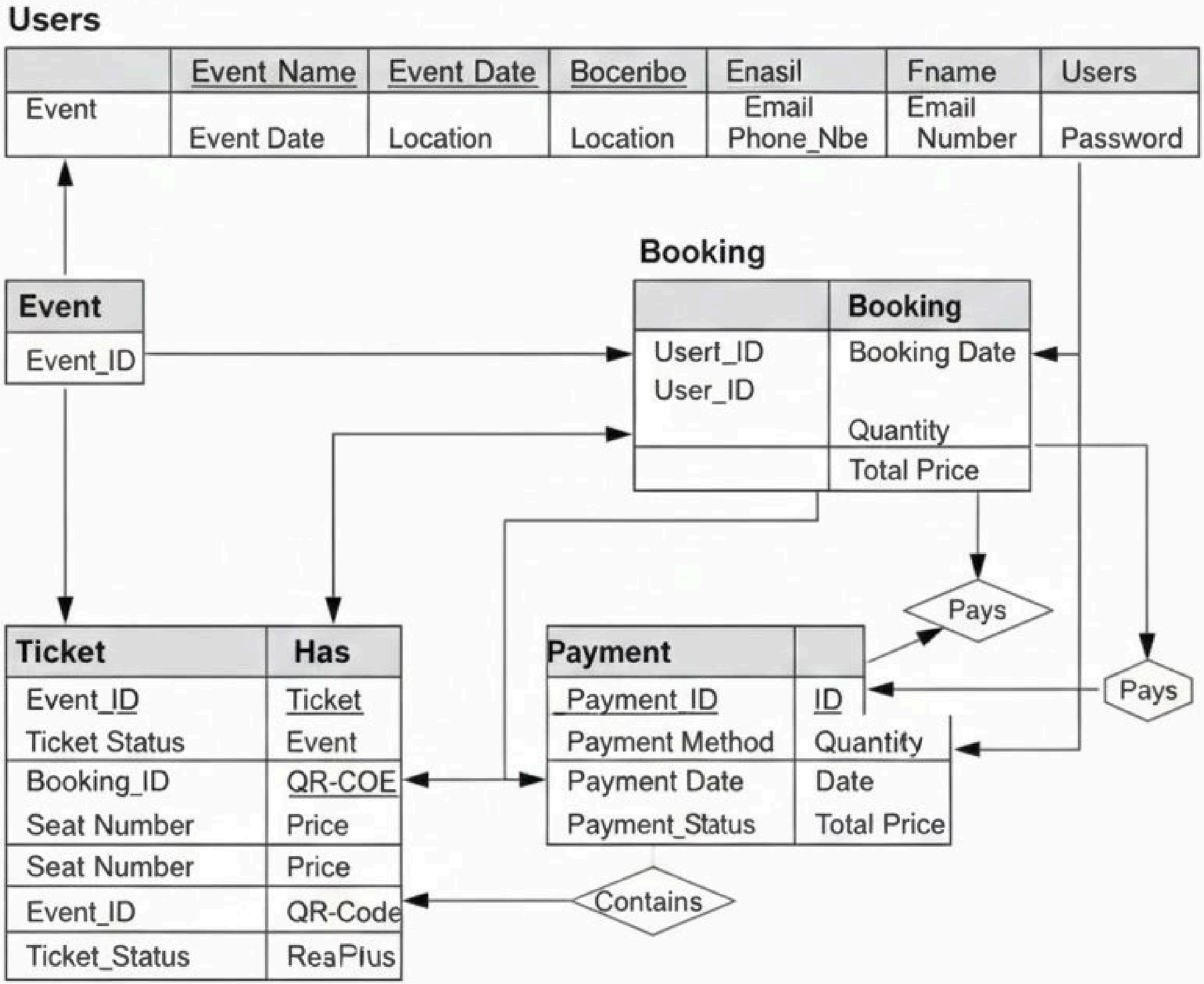
The Ticket table stores ticket information including seat number, price, and QR code. Each ticket is linked to one booking and one event.

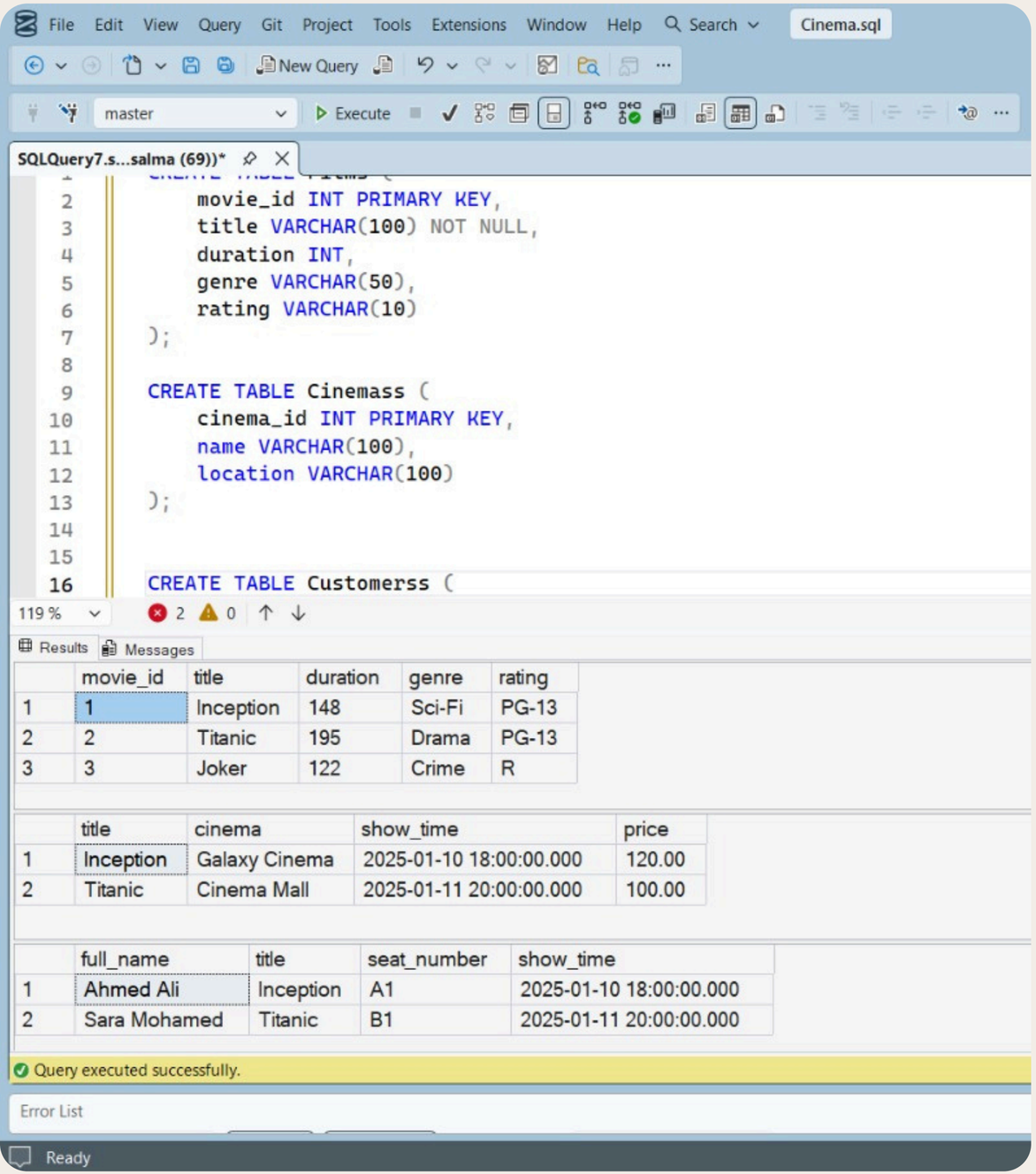
-PAYMENT

The Payment table stores payment details. Each booking has one payment, forming a one-to-one relationship

Conclusion

The ERD was successfully converted into a relational schema with clear primary and foreign keys, ensuring data integrity.


Users
Event Name
Event Date
Boceribo
Enasil
Fname
Users
Event
Event Date
Location
Location
Email
Phone_Nbe
Email
Number
Password
Booking
Event
Event_ID
Booking
Usert_ID
User_ID
Booking Date
Quantity
Total Price
Pays
Pays
Ticket
Has
Event_ID
Ticket Status
Booking_ID
Seat Number
Seat Number
Event_ID
Ticket_Status
Ticket
Event
QR-COE
Price
Price
QR-Code
ReaPlus
Payment
Payment_ID
Payment Method
Payment Date
Payment_Status
ID
Quantity
Date
Total Price
Contains


File Edit View Query Git Project Tools Extensions Window Help Search Cinema.sql
New Query
master
Execute
SQLQuery7.s...salma (69))*
movie_id INT PRIMARY KEY,
title VARCHAR(100) NOT NULL,
duration INT,
genre VARCHAR(50),
rating VARCHAR(10)
);
CREATE TABLE Cinemass (
cinema_id INT PRIMARY KEY,
name VARCHAR(100),
location VARCHAR(100)
);
CREATE TABLE Customerss (
119 %
2
0
Results
Messages
movie_id title duration genre rating
1 1 Inception 148 Sci-Fi PG-13
2 2 Titanic 195 Drama PG-13
3 3 Joker 122 Crime R
title cinema show_time price
1 Inception Galaxy Cinema 2025-01-10 18:00:00.000 120.00
2 Titanic Cinema Mall 2025-01-11 20:00:00.000 100.00
full_name title seat_number show_time
1 Ahmed Ali Inception A1 2025-01-10 18:00:00.000
2 Sara Mohamed Titanic B1 2025-01-11 20:00:00.000
Query executed successfully.
Error List
Ready

SQL

```
CREATE TABLE Films (
movie_id INT PRIMARY KEY,
title VARCHAR(100) NOT NULL,
duration INT,
genre VARCHAR(50),
rating VARCHAR(10)
);
CREATE TABLE Cinemass (
cinema_id INT PRIMARY KEY,
name VARCHAR(100),
location VARCHAR(100)
);
CREATE TABLE Customerss (
customer_id INT PRIMARY KEY,
full_name VARCHAR(100),
phone VARCHAR(20)
);
CREATE TABLE Showss (
show_id INT PRIMARY KEY,
movie_id INT,
cinema_id INT,
show_time DATETIME,
price DECIMAL(6,2),
FOREIGN KEY (movie_id) REFERENCES
Films(movie_id),
FOREIGN KEY (cinema_id) REFERENCES
Cinemass(cinema_id)
);
```

```
CREATE TABLE Seats (
seat_id INT PRIMARY KEY,
cinema_id INT,
seat_number VARCHAR(10),
FOREIGN KEY (cinema_id)
REFERENCES
Cinemass(cinema_id)
);
/* ========================
BOOKINGS TABLE
======================== */
CREATE TABLE Bookings (
booking_id INT PRIMARY KEY,
customer_id INT,
show_id INT,
seat_id INT,
booking_date DATETIME,
FOREIGN KEY (customer_id)
REFERENCES
Customerss(customer_id),
FOREIGN KEY (show_id)
REFERENCES Showss(show_id),
FOREIGN KEY (seat_id) REFERENCES
Seats(seat_id)
);
INSERT INTO Films VALUES
(1, 'Inception', 148, 'Sci-Fi', 'PG-13'),
(2, 'Titanic', 195, 'Drama', 'PG-13'),
(3, 'Joker', 122, 'Crime', 'R');
```

```
/* ========================
INSERT CINEMAS
======================== */
INSERT INTO Cinemass VALUES
(1, 'Galaxy Cinema', 'Nasr City'),
(2, 'Cinema Mall', '6 October');
INSERT INTO Customerss VALUES
(1, 'Ahmed Ali', '01012345678'),
(2, 'Sara Mohamed', '01198765432');
INSERT INTO Showss VALUES
(1, 1, 1, '2025-01-10 18:00:00',
120.00),
(2, 2, 2, '2025-01-11 20:00:00',
100.00);
INSERT INTO Seats VALUES
(1, 1, 'A1'),
(2, 1, 'A2'),
(3, 2, 'B1');
INSERT INTO Bookings VALUES
(1, 1, 1, 1, GETDATE()),
(2, 2, 2, 3, GETDATE());
```

```
SELECT * FROM Films;

SELECT
Films.title,
Cinemass.name AS cinema,
Showss.show_time,
Showss.price
FROM Showss
JOIN Films ON Showss.movie_id =
Films.movie_id
JOIN Cinemass ON
Showss.cinema_id =
Cinemass.cinema_id;

SELECT
Customerss.full_name,
Films.title,
Seats.seat_number,
Showss.show_time
FROM Bookings
JOIN Customerss ON
Bookings.customer_id =
Customerss.customer_id
JOIN Showss ON Bookings.show_id
= Showss.show_id
JOIN Films ON Showss.movie_id =
Films.movie_id
JOIN Seats ON Bookings.seat_id =
Seats.seat_id;
```

Introduction

This project aims to design a database system for cinema ticket booking, allowing customers to book movie tickets, select cinemas, show timings, and available seats, while also enabling management to view shows, customers, and bookings efficiently.

Project Objectives

- Create a flexible and scalable database.
- Manage data for movies, cinemas, and customers.
- Manage show schedules, seats, and ticket bookings.
- Generate queries for easy data display and booking management.

Main Tables

Table name	Description	Key Columns
Movies	Stores movie information	movie_id, title, duration, genre, rating
Cinemas	Stores cinema information	cinema_id, name, location
Costomers	Stores customer information	customer_id, full_name, phone
Shows	Stores show schedules for each movie in each cinema	show_id, movie_id, cinema_id, show_time, price
Seats	Stores seat information for each cinema	seat_id, cinema_id, seat_number
Brookings	Stores ticket bookings for customers	booking_id, customer_id, show_id, seat_id, booking_date

Table Relationships

- **Movies → Shows: One-to-Many (One movie can have multiple shows).**
- **Cinemas → Shows: One-to-Many (One cinema can have multiple shows).**
- **Customers → Bookings: One-to-Many (A customer can book multiple tickets).**
- **Shows → Bookings: One-to-Many (One show can have multiple bookings).**
- **Seats → Bookings: One-to-One per booking (Each seat can only be booked once).ERD (Entity-Relationship Diagram):**

▶ ERD (Entity-Relationship Diagram):

Movies --< Shows >-- Cinemas
Customers --< Bookings >-- Shows
Bookings -- Seats

SQL Queries Used

Display all movies:

```
SELECT * FROM Movies;
```

Display shows with movie name, cinema, and price:

```
SELECT Movies.title, Cinemas.name AS cinema, Shows.show_time, Shows.price
FROM Shows
JOIN Movies ON Shows.movie_id = Movies.movie_id
JOIN Cinemas ON Shows.cinema_id = Cinemas.cinema_id;
```

Display customer bookings:

```
SELECT Customers.full_name, Movies.title, Seats.seat_number, Shows.show_time
FROM Bookings
JOIN Customers ON Bookings.customer_id = Customers.customer_id
JOIN Shows ON Bookings.show_id = Shows.show_id
JOIN Movies ON Shows.movie_id = Movies.movie_id
JOIN Seats ON Bookings.seat_id = Seats.seat_id;
```

Important Notes

- **PRIMARY KEY and AUTO_INCREMENT were used to avoid duplicate records.**
- **FOREIGN KEY constraints were applied to maintain referential integrity.**
- **Queries are ready for demonstration purposes.**
- **The system is scalable for future features, such as:**
  - Payment management
  - Revenue reports
  - Special discounts and offers

Conclusion

This project provides a complete Cinema Ticket Booking System using SQL, with well-structured tables, clear relationships, and ready-to-use queries. It successfully achieves the objectives of managing movies, cinemas, customers, shows, and bookings in an organized and efficient manner.