CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

UNIVERSITY INSTITUTE OF COMPUTING

CASE STUDY REPORT
ON
CINEMA MANAGEMENT SYSTEM

Program Name: BCA

Subject Name/Code: Database Management System (23CAT-251)

Submitted By:

Name: Chetan Sharma
UID: 23BCA10433
Section: 4-‘B’

Submitted To:

Name: Mr.Arvinder Singh
Designation: Assistant Professor

INTRODUCTION

The Cinema Database Management System is a relational database solution developed to streamline and automate the processes involved in managing a cinema hall. In modern cinema operations, efficiency, speed, and accuracy in handling information such as movie schedules, customer bookings, theater infrastructure, and show timings are crucial. Manual handling of these tasks often leads to errors, delays, and inefficiencies.

This system addresses these issues by storing and organising data in structured tables with well-defined relationships, allowing for fast and reliable data access. The system is built to support daily operations such as adding new movies, assigning screens, scheduling shows, registering customers, and managing bookings. It also provides query capabilities to generate insights like revenue reports, seat availability, and booking history.

By adopting a database-driven approach, cinema administrators can ensure a smooth workflow, better customer satisfaction, and improved decision-making based on real-time data.

TECHNIQUES

The primary technology used in this project is MySQL, an open-source relational database management system. The following techniques have been implemented:

- **Entity-Relationship Modeling** for data structure visualisation.
- **Normalisation** to organise data efficiently and remove redundancy.
- **SQL Queries** for data manipulation and retrieval.
- **Use of Constraints** like PRIMARY KEY, FOREIGN KEY to enforce relationships.
- **Join operations** to combine data from multiple tables.
- **Aggregate Functions** to summarize and analyze data.
- **Filtering and Sorting** to extract meaningful insights from the dataset.
- **Stored Procedures and Views** (optional enhancements) for automation.

The goal is to simulate a real-time cinema database with multiple users accessing the system concurrently. Though our current system is simplified, it lays the foundation for large-scale enterprise software.

SYSTEM CONFIGURATION

- **Operating System:**
- Windows 10 or higher / Linux / macOS
- **Database Software:**
- MySQL or PostgreSQL
- **RAM:**
- Minimum 4GB
- **Processor:**
- Intel i3 or equivalent and above
- **Other Tools:**
- MySQL Workbench, DBeaver, or phpMyAdmin

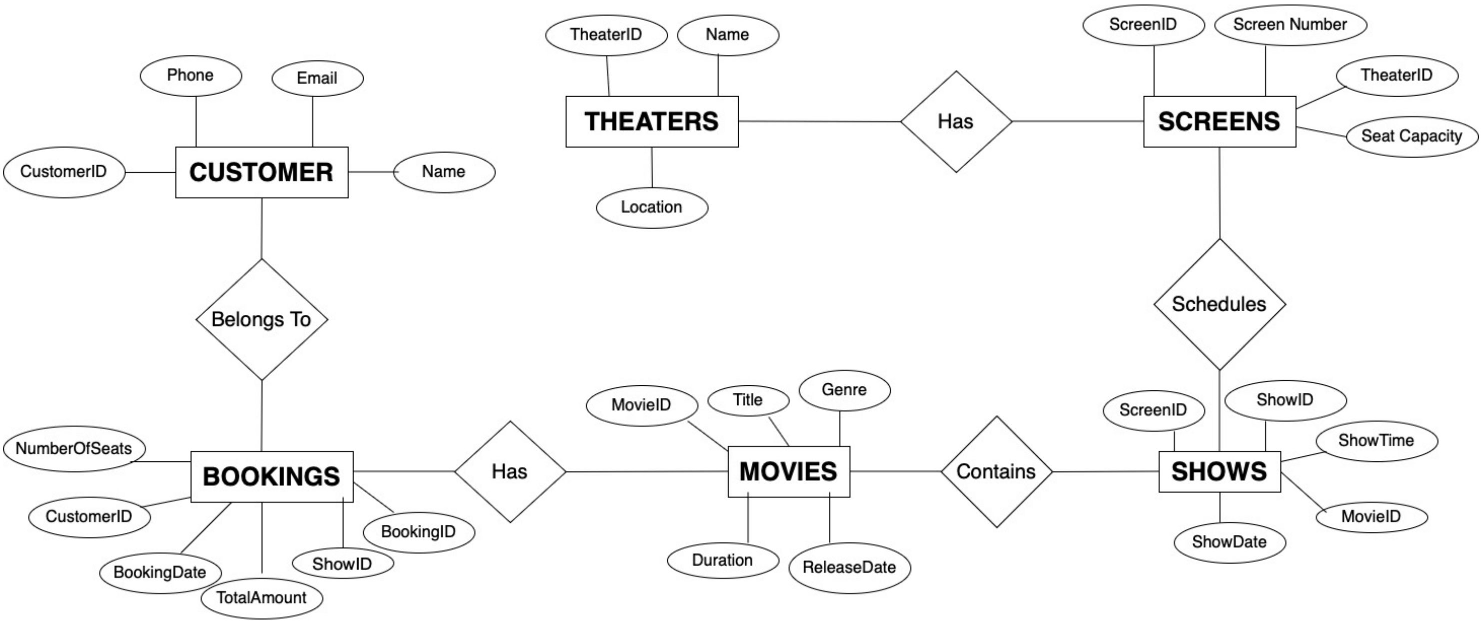
INPUT

The system accepts a variety of inputs to populate the database and simulate cinema functionality. These inputs include:

- **Movie details** (title, genre, release date)
- **Theater details** (name, location)
- **Screen details** (screen number, seat capacity)
- **Show schedule** (date, time, movie, screen)
- **Customer details** (name, email, phone)
- **Booking details** (number of seats, total amount)

Each input plays a vital role in maintaining an up-to-date and accurate database.

ENTITY-RELATIONSHIP DIAGRAM



The Entity-Relationship (ER) diagram outlines the structure and relationships among different entities of the hospital. It forms the blueprint for the actual database schema.

Each entity has clearly defined attributes and is connected using appropriate relationships like one-to-many and many-to-one, ensuring normalization and avoiding data redundancy.

RELATIONSHIP BETWEEN TABLES

These relationships ensure that the relational database mirrors real-world interactions within a cinema.

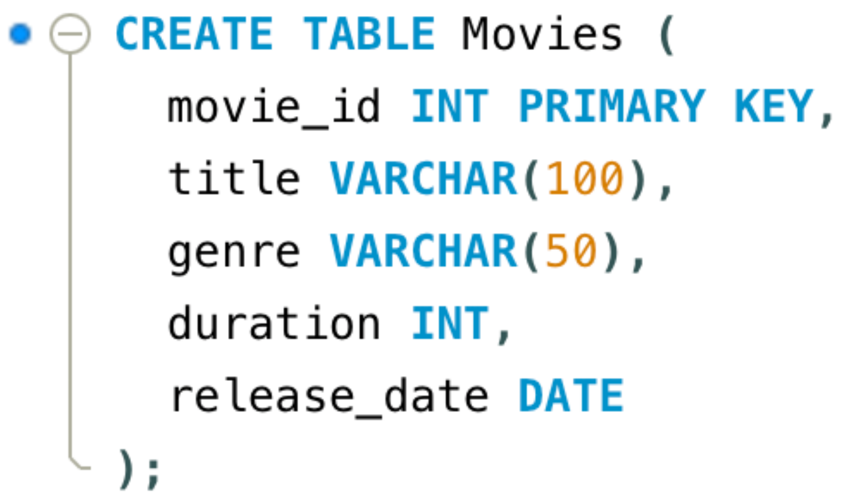
No.	Relationship Type	Parent Table	Child Table	Foreign key	Description
1	One-to-many	Theaters	Screens	TheaterID	One theater can have multiple screens
2	One-to-many	Screens	Shows	ScreenID	One screen can host multiple shows
3	One-to-many	Movies	Shows	MovieID	One movie can be shown in multiple shows
4	One-to-many	Customers	Bookings	CustomerID	One customer make multiple booking
5	One-to-many	Shows	Bookings	ShowID	One show have multiple booking
6	One-to-one	Bookings	Customers	BookingID (unique)	One booking with one customer

TABULAR FORMAT (SCHEMA)

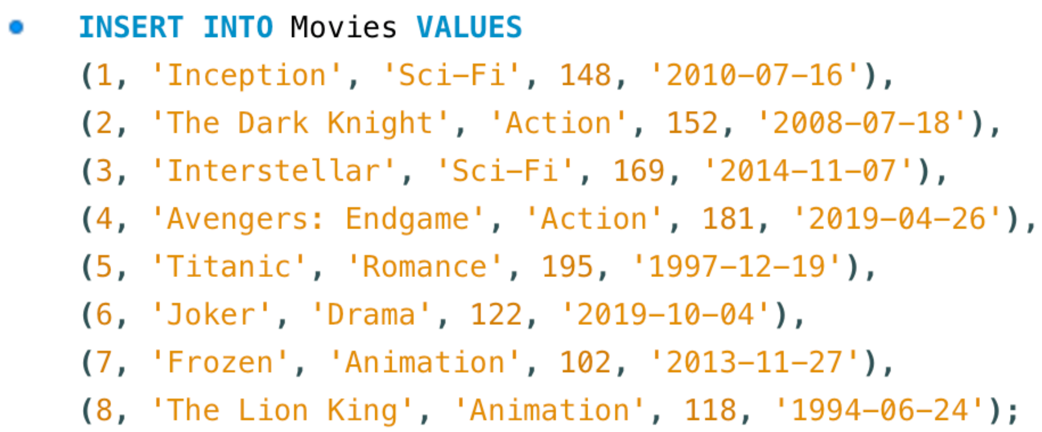
Table Name	Primary Key	Foreign Key	Description
Movies	MovieID	—	Stores movie details
Theaters	TheaterID	—	Stores theater detail
Screens	ScreenID	TheaterID	Screens in theater
Shows	ShowID	MovieID	Schedule for movie
Customers	CustomerID	—	Store customer info
Bookings	BoookingID	CustomerID	Records for shows

TABLE CREATION

1. Movie Table:

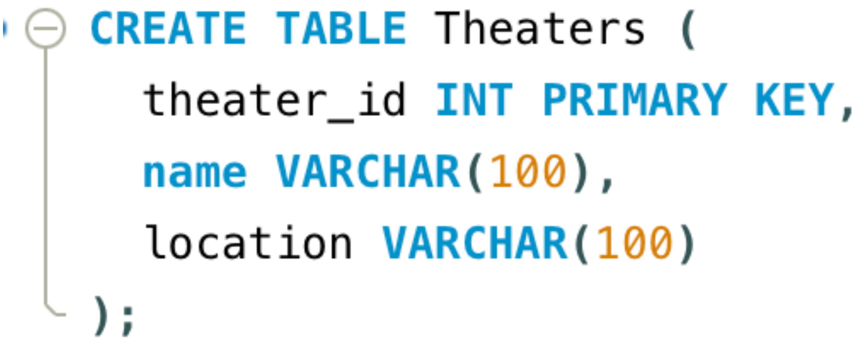


```
CREATE TABLE Movies (
  movie_id INT PRIMARY KEY,
  title VARCHAR(100),
  genre VARCHAR(50),
  duration INT,
  release_date DATE
);
```

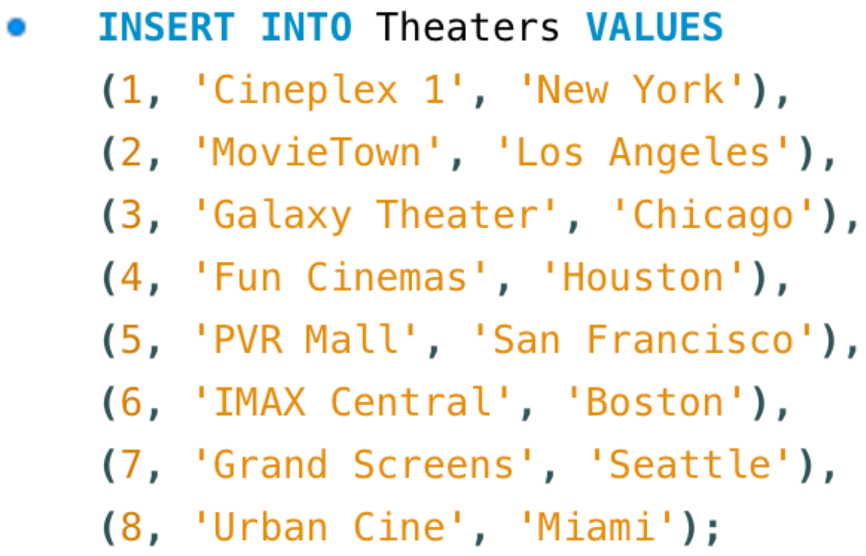


```
INSERT INTO Movies VALUES
(1, 'Inception', 'Sci-Fi', 148, '2010-07-16'),
(2, 'The Dark Knight', 'Action', 152, '2008-07-18'),
(3, 'Interstellar', 'Sci-Fi', 169, '2014-11-07'),
(4, 'Avengers: Endgame', 'Action', 181, '2019-04-26'),
(5, 'Titanic', 'Romance', 195, '1997-12-19'),
(6, 'Joker', 'Drama', 122, '2019-10-04'),
(7, 'Frozen', 'Animation', 102, '2013-11-27'),
(8, 'The Lion King', 'Animation', 118, '1994-06-24');
```

2. Theater Table:

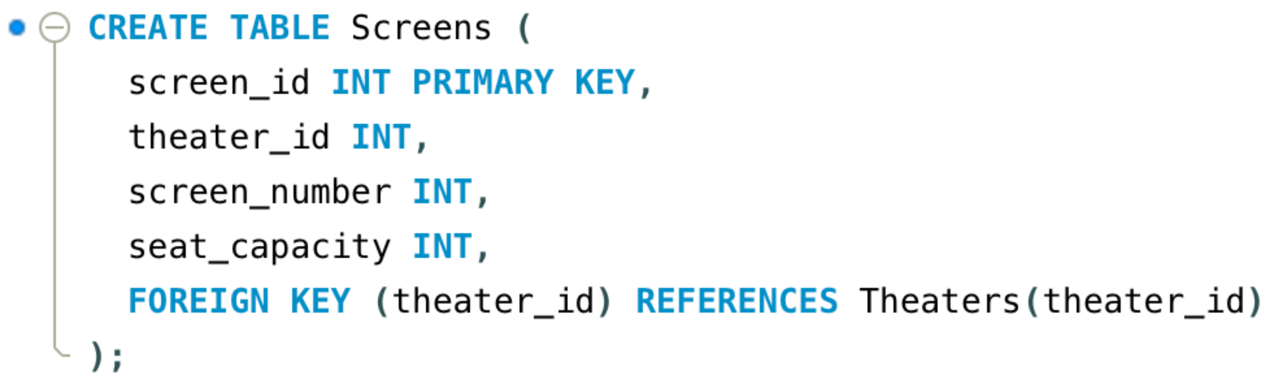


```
CREATE TABLE Theaters (
  theater_id INT PRIMARY KEY,
  name VARCHAR(100),
  location VARCHAR(100)
);
```

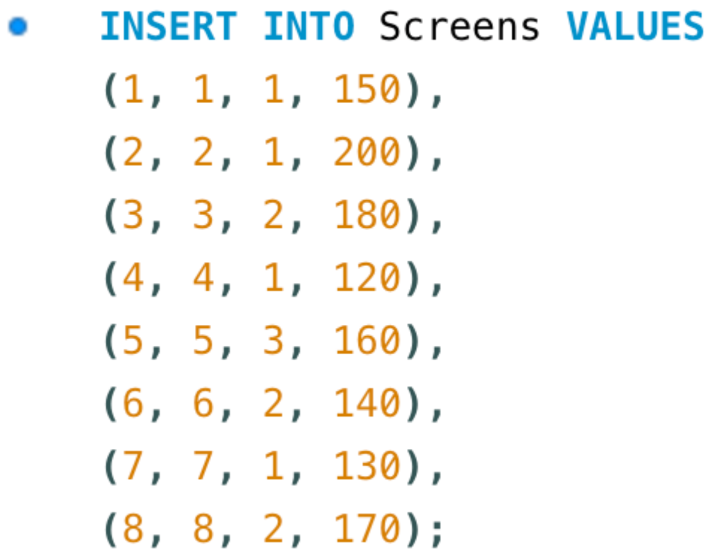


```
INSERT INTO Theaters VALUES
(1, 'Cineplex 1', 'New York'),
(2, 'MovieTown', 'Los Angeles'),
(3, 'Galaxy Theater', 'Chicago'),
(4, 'Fun Cinemas', 'Houston'),
(5, 'PVR Mall', 'San Francisco'),
(6, 'IMAX Central', 'Boston'),
(7, 'Grand Screens', 'Seattle'),
(8, 'Urban Cine', 'Miami');
```

3. Screens Table

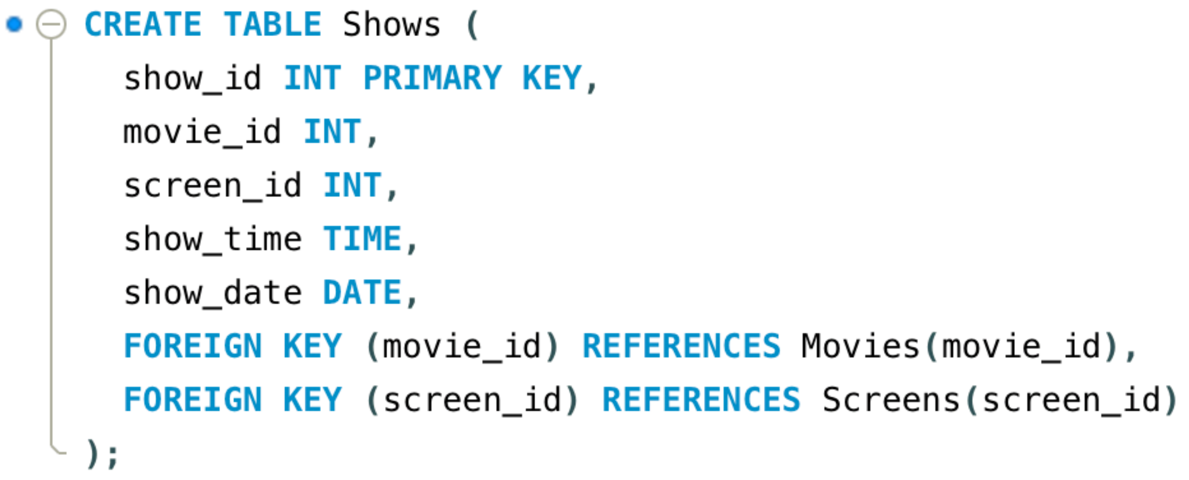


```
CREATE TABLE Screens (
  screen_id INT PRIMARY KEY,
  theater_id INT,
  screen_number INT,
  seat_capacity INT,
  FOREIGN KEY (theater_id) REFERENCES Theaters(theater_id)
);
```

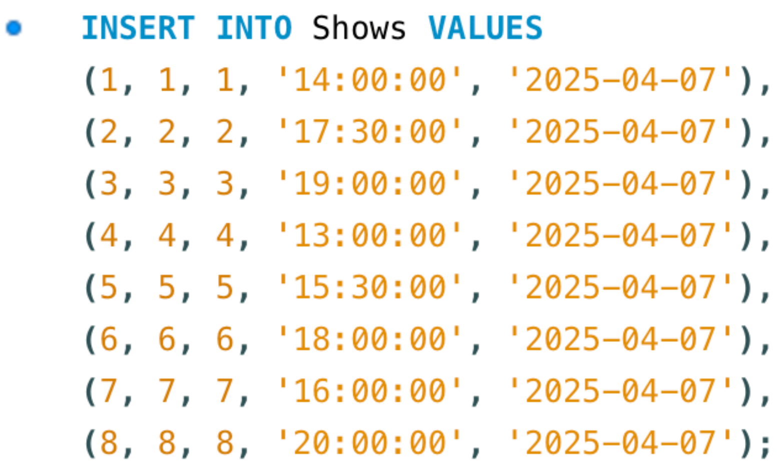


```
INSERT INTO Screens VALUES
(1, 1, 1, 150),
(2, 2, 1, 200),
(3, 3, 2, 180),
(4, 4, 1, 120),
(5, 5, 3, 160),
(6, 6, 2, 140),
(7, 7, 1, 130),
(8, 8, 2, 170);
```

4. Shows Table

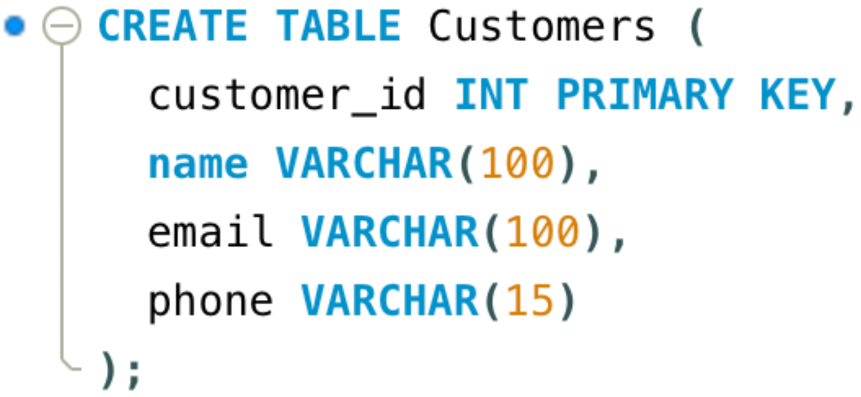


```
CREATE TABLE Shows (
  show_id INT PRIMARY KEY,
  movie_id INT,
  screen_id INT,
  show_time TIME,
  show_date DATE,
  FOREIGN KEY (movie_id) REFERENCES Movies(movie_id),
  FOREIGN KEY (screen_id) REFERENCES Screens(screen_id)
);
```

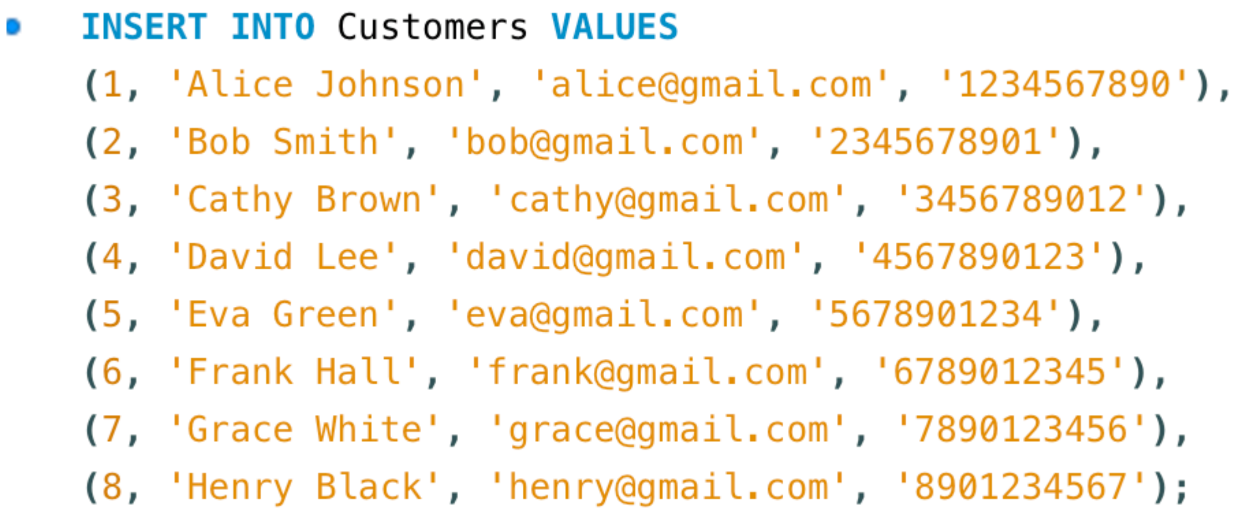


```
INSERT INTO Shows VALUES
(1, 1, 1, '14:00:00', '2025-04-07'),
(2, 2, 2, '17:30:00', '2025-04-07'),
(3, 3, 3, '19:00:00', '2025-04-07'),
(4, 4, 4, '13:00:00', '2025-04-07'),
(5, 5, 5, '15:30:00', '2025-04-07'),
(6, 6, 6, '18:00:00', '2025-04-07'),
(7, 7, 7, '16:00:00', '2025-04-07'),
(8, 8, 8, '20:00:00', '2025-04-07');
```

5. Customers Table

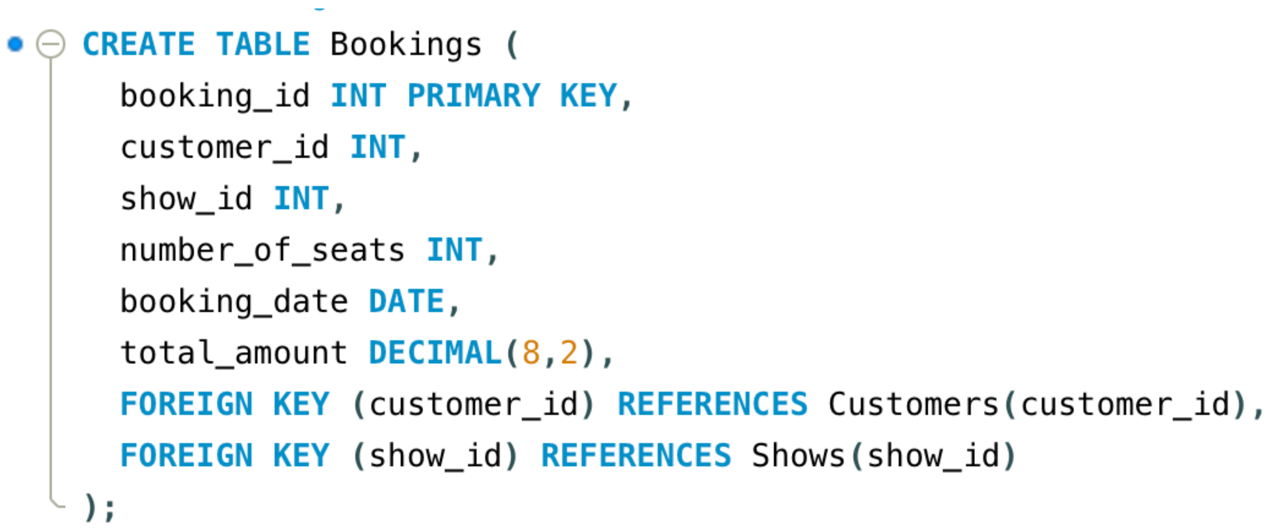


```
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(15)
);
```

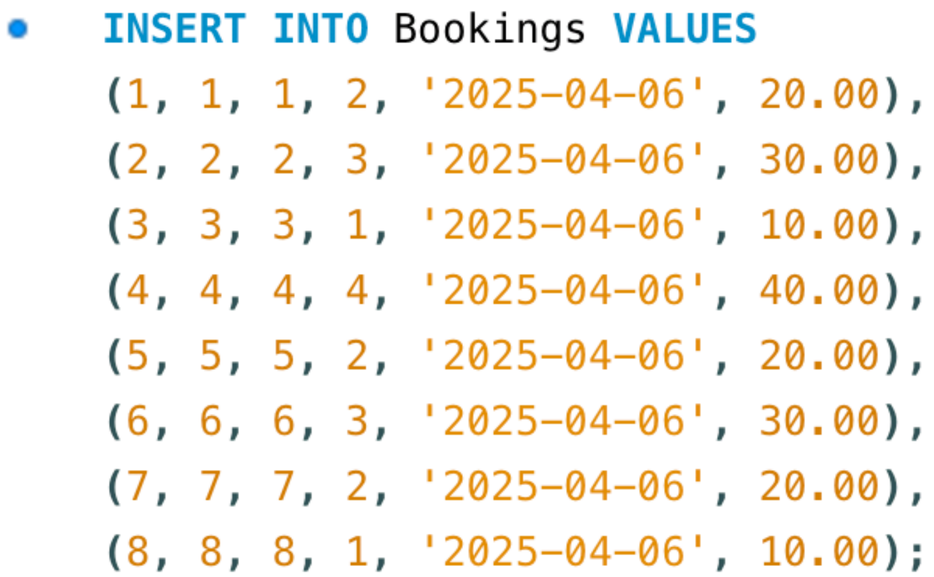


```
INSERT INTO Customers VALUES
(1, 'Alice Johnson', 'alice@gmail.com', '1234567890'),
(2, 'Bob Smith', 'bob@gmail.com', '2345678901'),
(3, 'Cathy Brown', 'cathy@gmail.com', '3456789012'),
(4, 'David Lee', 'david@gmail.com', '4567890123'),
(5, 'Eva Green', 'eva@gmail.com', '5678901234'),
(6, 'Frank Hall', 'frank@gmail.com', '6789012345'),
(7, 'Grace White', 'grace@gmail.com', '7890123456'),
(8, 'Henry Black', 'henry@gmail.com', '8901234567');
```

6. Bookings Table

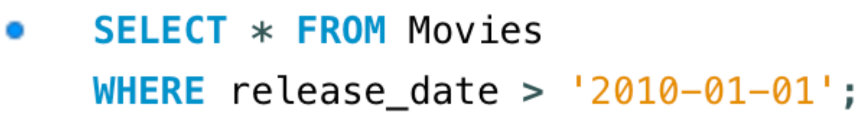


```
CREATE TABLE Bookings (
    booking_id INT PRIMARY KEY,
    customer_id INT,
    show_id INT,
    number_of_seats INT,
    booking_date DATE,
    total_amount DECIMAL(8,2),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
    FOREIGN KEY (show_id) REFERENCES Shows(show_id)
);
```

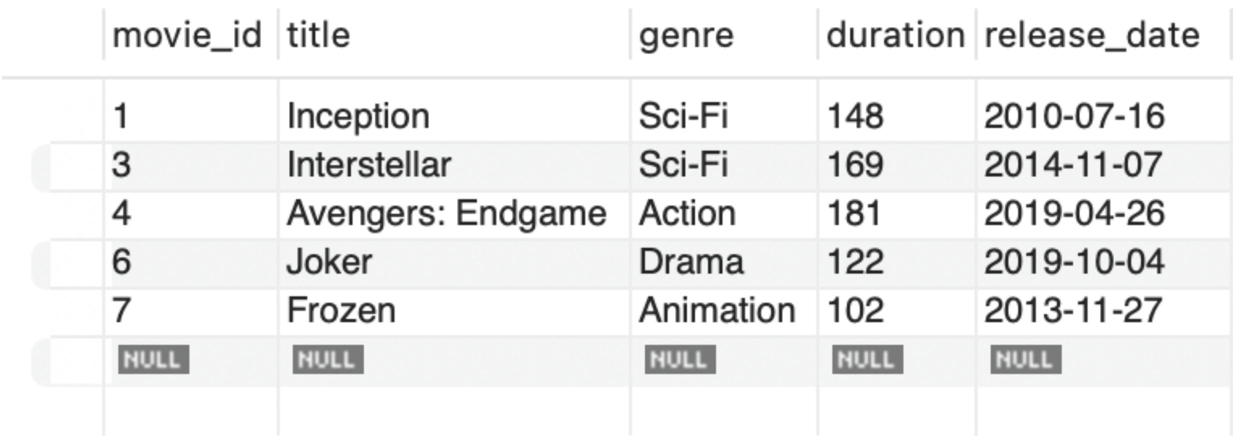


```
INSERT INTO Bookings VALUES
(1, 1, 1, 2, '2025-04-06', 20.00),
(2, 2, 2, 3, '2025-04-06', 30.00),
(3, 3, 3, 1, '2025-04-06', 10.00),
(4, 4, 4, 4, '2025-04-06', 40.00),
(5, 5, 5, 2, '2025-04-06', 20.00),
(6, 6, 6, 3, '2025-04-06', 30.00),
(7, 7, 7, 2, '2025-04-06', 20.00),
(8, 8, 8, 1, '2025-04-06', 10.00);
```

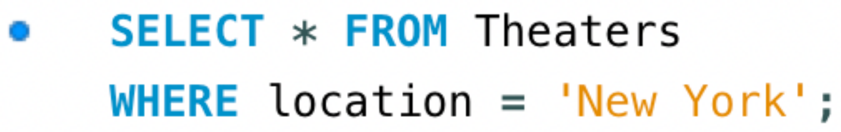
SQL QUERIES (13 Queries)



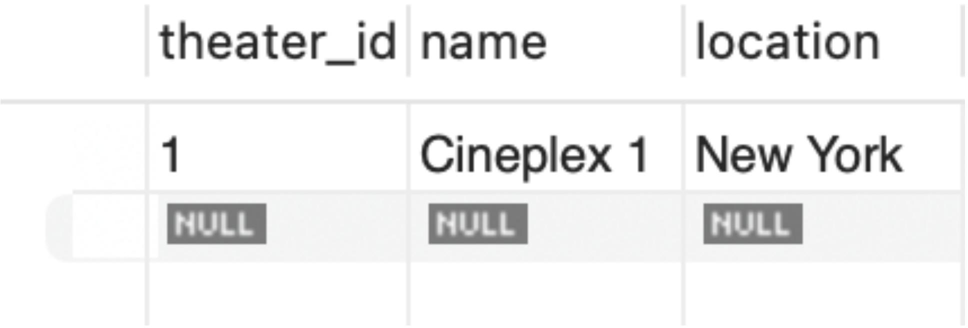
```
SELECT * FROM Movies
WHERE release_date > '2010-01-01';
```



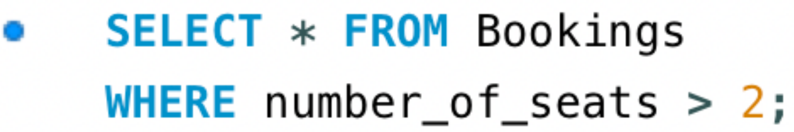
movie_id	title	genre	duration	release_date
1	Inception	Sci-Fi	148	2010-07-16
3	Interstellar	Sci-Fi	169	2014-11-07
4	Avengers: Endgame	Action	181	2019-04-26
6	Joker	Drama	122	2019-10-04
7	Frozen	Animation	102	2013-11-27
NULL	NULL	NULL	NULL	NULL



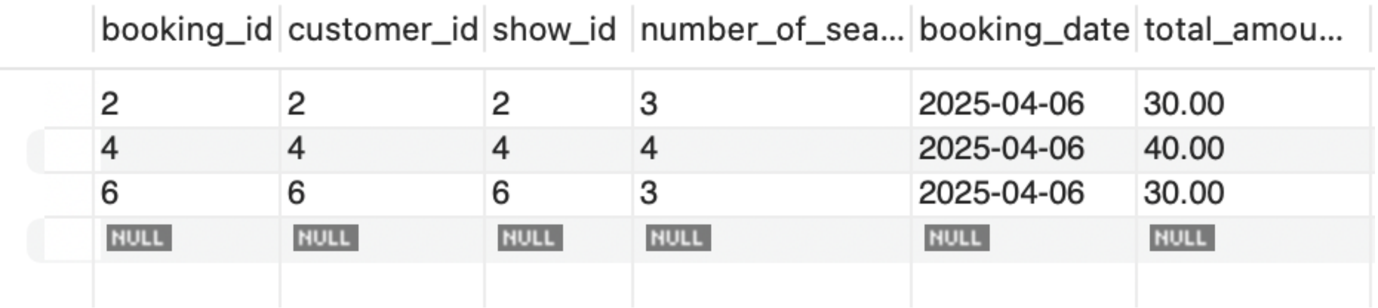
```
SELECT * FROM Theaters
WHERE location = 'New York';
```



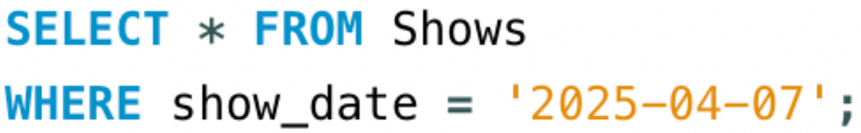
theater_id	name	location
1	Cineplex 1	New York
NULL	NULL	NULL



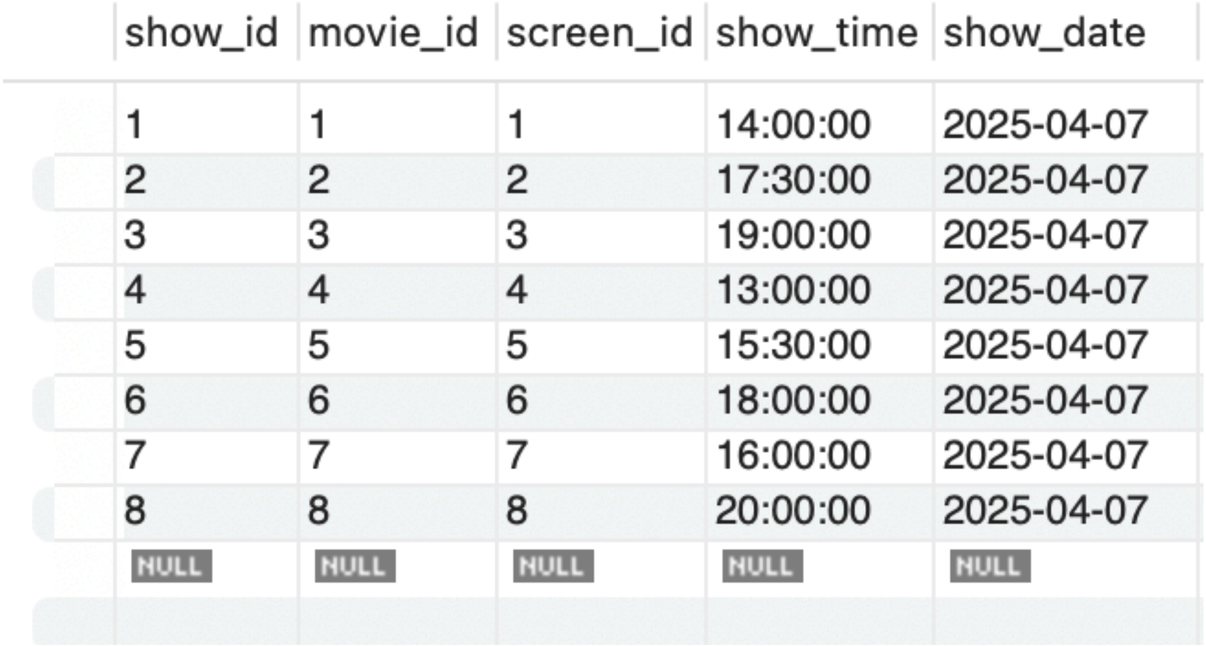
```
SELECT * FROM Bookings
WHERE number_of_seats > 2;
```



booking_id	customer_id	show_id	number_of_sea...	booking_date	total_amou...
2	2	2	3	2025-04-06	30.00
4	4	4	4	2025-04-06	40.00
6	6	6	3	2025-04-06	30.00
NULL	NULL	NULL	NULL	NULL	NULL

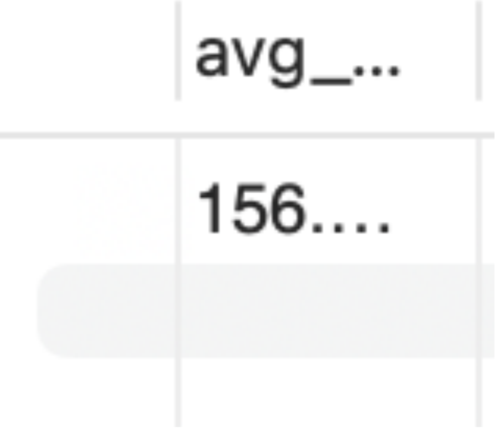


```
SELECT * FROM Shows
WHERE show_date = '2025-04-07';
```



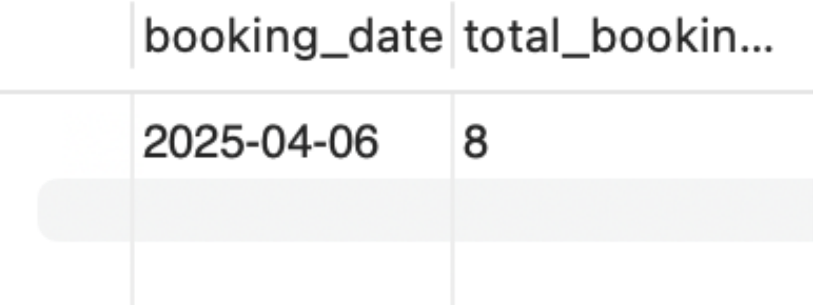
show_id	movie_id	screen_id	show_time	show_date
1	1	1	14:00:00	2025-04-07
2	2	2	17:30:00	2025-04-07
3	3	3	19:00:00	2025-04-07
4	4	4	13:00:00	2025-04-07
5	5	5	15:30:00	2025-04-07
6	6	6	18:00:00	2025-04-07
7	7	7	16:00:00	2025-04-07
8	8	8	20:00:00	2025-04-07
NULL	NULL	NULL	NULL	NULL

-
  ```sql
  SELECT AVG(seat_capacity) AS avg_capacity
  FROM Screens;
  ```



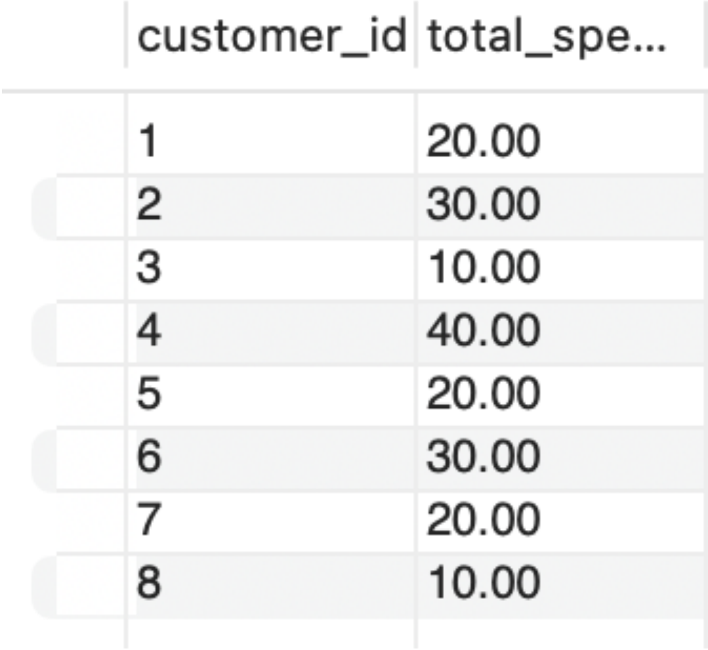
	avg_...
	156....

-
  ```sql
  SELECT booking_date, COUNT(*) AS total_bookings
  FROM Bookings
  GROUP BY booking_date;
  ```



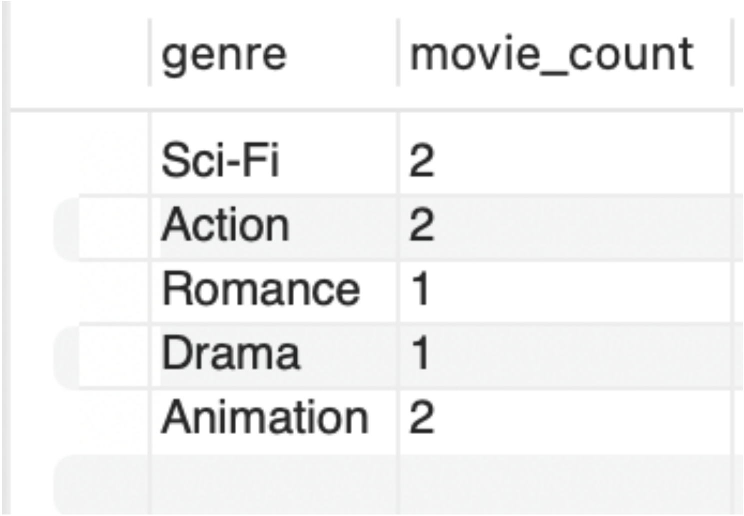
	booking_date	total_bookin...
	2025-04-06	8

-
  ```
  SELECT customer_id, SUM(total_amount) AS total_spent
  FROM Bookings
  GROUP BY customer_id;
  ```



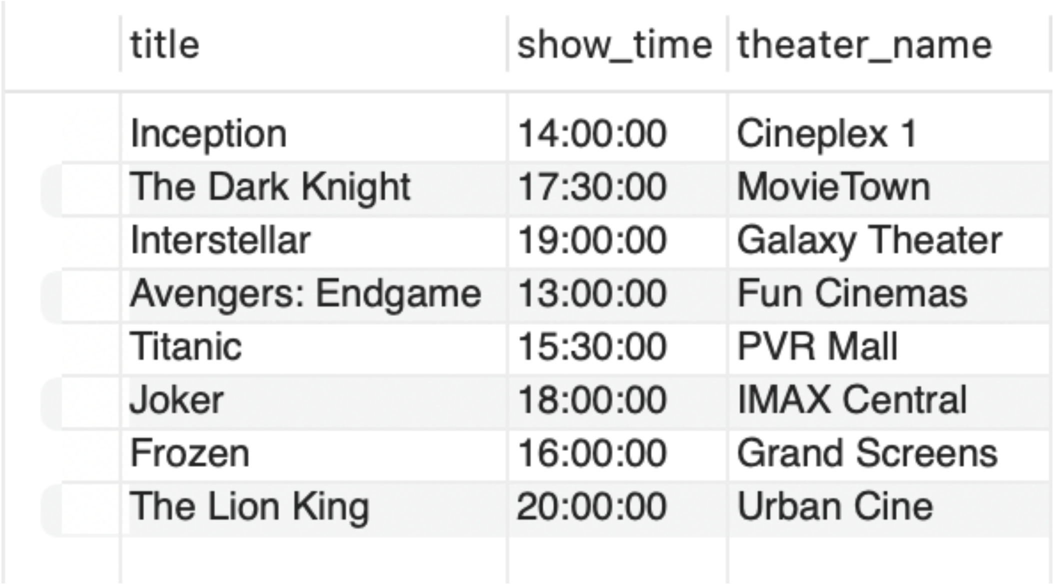
customer_id	total_spe...
1	20.00
2	30.00
3	10.00
4	40.00
5	20.00
6	30.00
7	20.00
8	10.00

-
  ```
  SELECT genre, COUNT(*) AS movie_count
  FROM Movies
  GROUP BY genre;
  ```



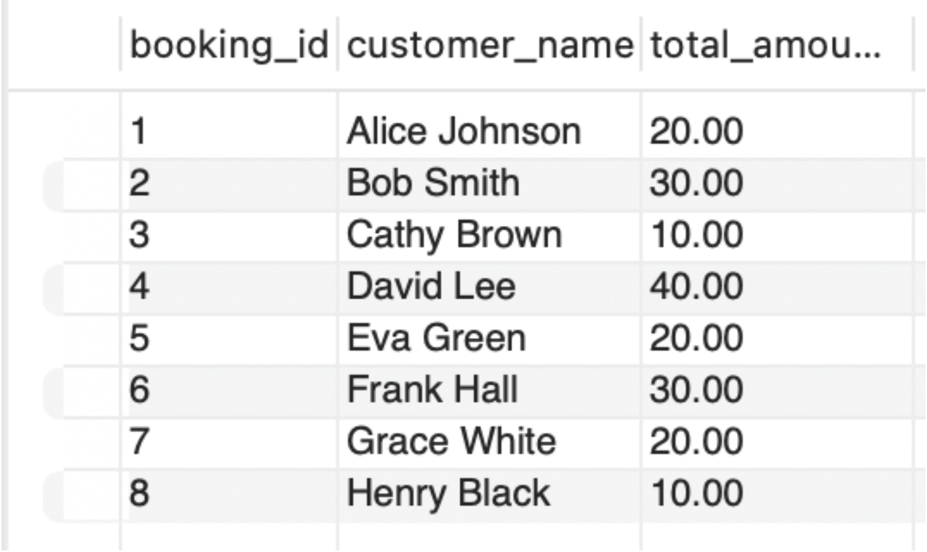
genre	movie_count
Sci-Fi	2
Action	2
Romance	1
Drama	1
Animation	2

-
  ```
  SELECT m.title, s.show_time, t.name AS theater_name
  FROM Shows s
  JOIN Movies m ON s.movie_id = m.movie_id
  JOIN Screens sc ON s.screen_id = sc.screen_id
  JOIN Theaters t ON sc.theater_id = t.theater_id;
  ```

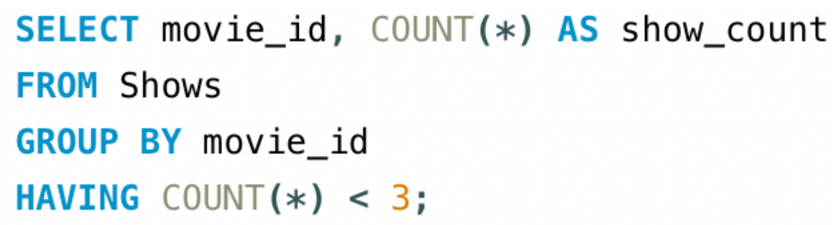


title	show_time	theater_name
Inception	14:00:00	Cineplex 1
The Dark Knight	17:30:00	MovieTown
Interstellar	19:00:00	Galaxy Theater
Avengers: Endgame	13:00:00	Fun Cinemas
Titanic	15:30:00	PVR Mall
Joker	18:00:00	IMAX Central
Frozen	16:00:00	Grand Screens
The Lion King	20:00:00	Urban Cine

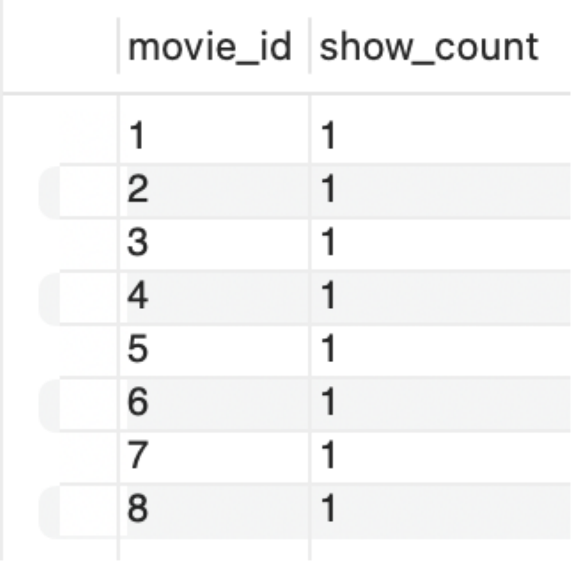
-
  ```
  SELECT b.booking_id, c.name AS customer_name, b.total_amount
  FROM Bookings b
  JOIN Customers c ON b.customer_id = c.customer_id;
  ```



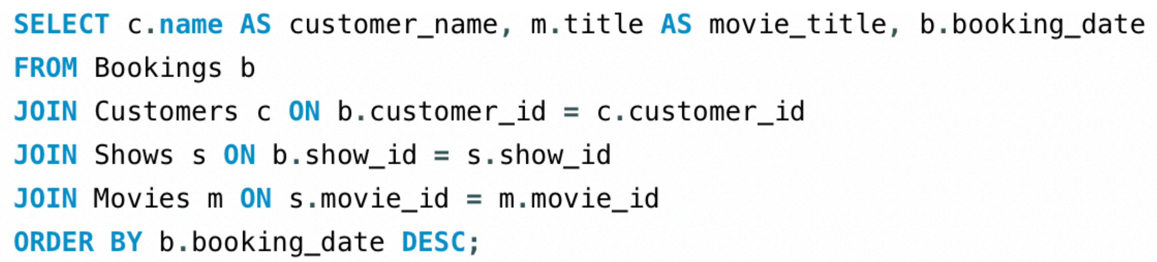
booking_id	customer_name	total_amou...
1	Alice Johnson	20.00
2	Bob Smith	30.00
3	Cathy Brown	10.00
4	David Lee	40.00
5	Eva Green	20.00
6	Frank Hall	30.00
7	Grace White	20.00
8	Henry Black	10.00

- 

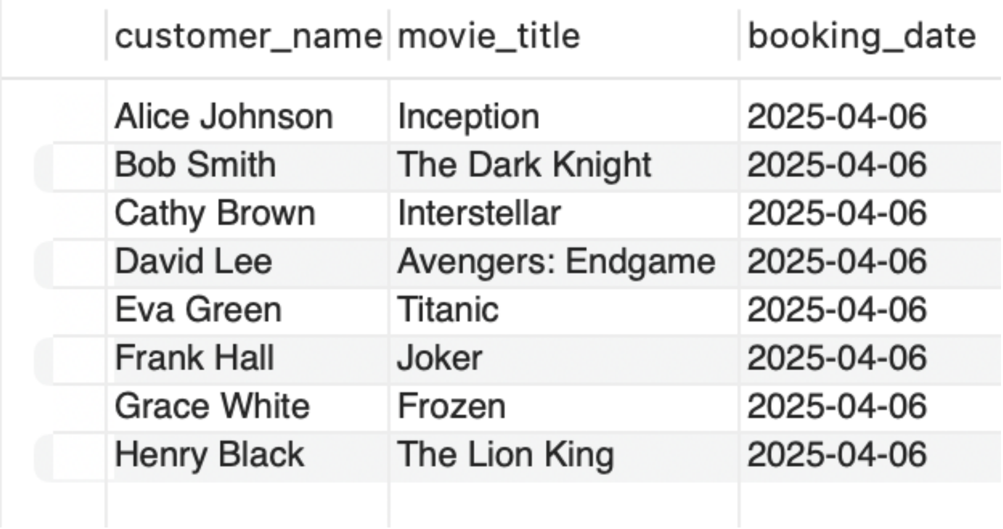
```
SELECT movie_id, COUNT(*) AS show_count
FROM Shows
GROUP BY movie_id
HAVING COUNT(*) < 3;
```



movie_id	show_count
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1

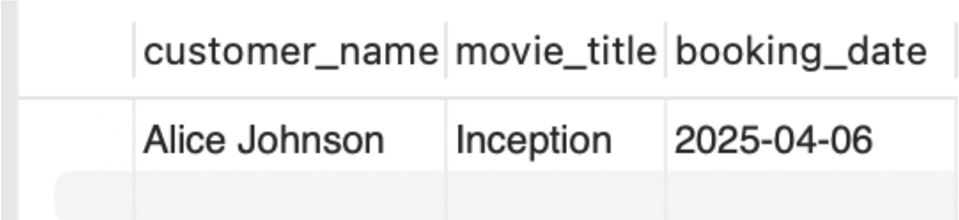
- 

```
SELECT c.name AS customer_name, m.title AS movie_title, b.booking_date
FROM Bookings b
JOIN Customers c ON b.customer_id = c.customer_id
JOIN Shows s ON b.show_id = s.show_id
JOIN Movies m ON s.movie_id = m.movie_id
ORDER BY b.booking_date DESC;
```



customer_name	movie_title	booking_date
Alice Johnson	Inception	2025-04-06
Bob Smith	The Dark Knight	2025-04-06
Cathy Brown	Interstellar	2025-04-06
David Lee	Avengers: Endgame	2025-04-06
Eva Green	Titanic	2025-04-06
Frank Hall	Joker	2025-04-06
Grace White	Frozen	2025-04-06
Henry Black	The Lion King	2025-04-06

-
  ```
  SELECT c.name AS customer_name, m.title AS movie_title, b.booking_date
  FROM Bookings b
  JOIN Customers c ON b.customer_id = c.customer_id
  JOIN Shows s ON b.show_id = s.show_id
  JOIN Movies m ON s.movie_id = m.movie_id
  ORDER BY b.booking_date DESC
  LIMIT 1;
  ```



customer_name	movie_title	booking_date
Alice Johnson	Inception	2025-04-06

SUMMARY

The Cinema Database Management System serves as a powerful tool for automating and organising the key functions of cinema operations. This system ensures a structured and logical way to manage data related to movies, theaters, screens, customers, shows, and bookings. It emphasises the importance of relational database design, including normalisation to eliminate redundancy and improve data consistency.

By implementing SQL queries, this system allows users to easily retrieve and manipulate data, offering practical insights such as customer activity, ticket sales, and seat allocation. The design and development of this project also demonstrate how database concepts such as entity-relationship modeling and foreign key constraints support data integrity and meaningful data analysis.

Overall, the project integrates technical knowledge with real-world application, showcasing the importance of a database system in enhancing efficiency, reducing errors, and delivering a smooth user experience for both customers and cinema staff.

CONCLUSION

The Cinema Database Management System offers a robust and organized approach to managing cinema operations. Here is a point-wise conclusion summarising key aspects:

Key Observations:

- The system simplifies booking, show scheduling, and customer management.
- Normalised table structure helps avoid data redundancy and ensures consistency.
- SQL queries allow efficient data retrieval for various management purposes.

Limitations:

- The current system does not handle real-time seat availability or payment gateway integration.
- User interface and front-end components are not part of this version.
- Security measures like user authentication and role-based access are not implemented.

Future Scope:

- Integration with online ticket booking portals and payment systems.
- Addition of admin panel and user dashboard for better interactivity.
- Implementation of data analytics tools for tracking customer behaviour and revenue trends.
- Real-time notification systems for show updates and promotional messages.