**Bookmyshow**

**Objective**

The goal of this assignment is to design a relational database schema for an online movie ticket booking platform similar to BookMyShow. The system is intended to handle core operations such as managing movies, theatres, screens, seat layouts, show schedules, customer registrations, and seat-level ticket bookings. The design ensures high data integrity and adheres to normalization principles to avoid redundancy.

**Schema Overview**

The schema consists of eight well-structured tables. Each table is normalized up to Boyce-Codd Normal Form (BCNF) and is designed to reflect the real-world relationships between entities. Below is a summary of the purpose of each table:

| **Table** | **Description** |
| --- | --- |
| movie | Stores details about movies, including title, language, genre, certificate, and IMDb rating. |
| theatre | Represents physical theatres, including name, location, and address details. |
| screen | Represents individual screens/auditoriums within a theatre, including screen number, audio system, and screen type. |
| seat | Defines the seating layout for each screen, including seat label, row, column, and class. |
| showtime | Represents scheduled screenings of a movie on a specific screen at a specific time, with optional subtitle information. |
| customer | Stores customer registration information, including name, email, mobile number, and registration timestamp. |
| ticket | Represents a booking transaction made by a customer for a specific show, including payment status and total amount. |
| ticket\_seat | Contains line items for each seat booked under a ticket, ensuring seat-level traceability and preventing duplicate bookings. |

**Data Integrity and Constraints**

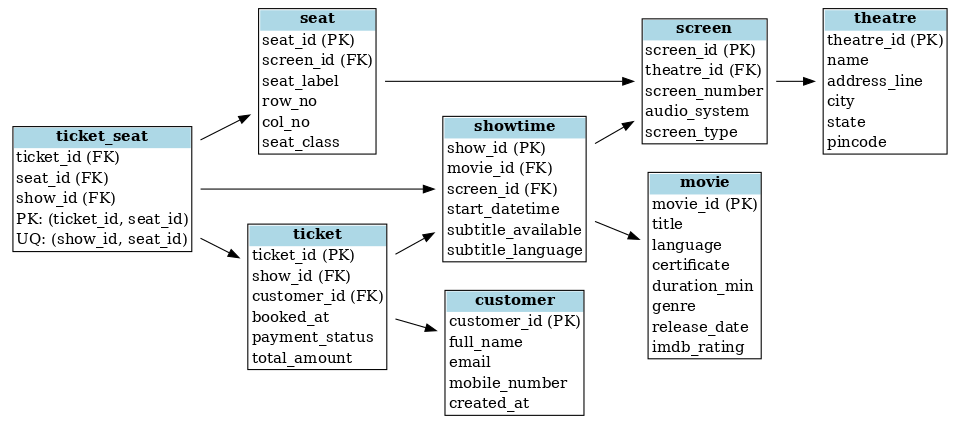
* All tables use primary keys to uniquely identify records.
* Foreign key constraints are implemented to maintain referential integrity between related entities.
* Unique constraints are used where necessary to prevent duplication, such as preventing the same seat from being booked more than once for a show.
* Enum types are used to restrict values for attributes like certificate and payment\_status, improving data consistency.
* The ticket\_seat table includes show\_id to enable a direct uniqueness constraint on (show\_id, seat\_id), which enforces seat-level exclusivity per show.

**Normalization Compliance**

* All tables are in **First Normal Form (1NF)** – atomic values are stored in each column.
* **Second Normal Form (2NF)** is ensured by eliminating partial dependencies.
* **Third Normal Form (3NF)** is achieved by removing transitive dependencies.
* **Boyce-Codd Normal Form (BCNF)** compliance is maintained by ensuring that every determinant is a candidate key.

**ER Diagram**

An entity-relationship diagram has been generated to visually represent the schema, including all relationships, primary keys, and foreign keys. This diagram clearly illustrates how entities such as movies, screens, customers, and tickets are connected.



**Design Highlights**

* Supports multi-seat bookings per ticket while maintaining a single transaction record.
* Prevents double booking of the same seat in the same show through declarative database constraints.
* Modular and extensible structure, suitable for future enhancements such as pricing models, promotional offers, and seat availability tracking.
* Designed using best practices in relational modeling and normalization.

**P1**

**Tables and Attributes**

**1) Movie**

* **movie\_id** (PK)
* **title**
* **language**
* **certificate** (e.g., UA)
* **duration\_min**
* **genre**
* **release\_date**
* **imdb\_rating**
  + **FDs:** movie\_id → all other attrs

**2) Theatre**

* **theatre\_id** (PK)
* **name**
* **address\_line**
* **city**
* **state**
* **pincode**
  + **FDs:** theatre\_id → all other attrs

**3) Screen**

* **screen\_id** (PK)
* **theatre\_id** (FK → Theatre.theatre\_id)
* **screen\_number**
* **audio\_system**
* **screen\_type**
  + **FDs:** screen\_id

**4) Seat (physical seat master for a screen)**

* **seat\_id** (PK)
* **screen\_id** (FK → Screen.screen\_id)
* **seat\_label**
  + **UQ:** (screen\_id, seat\_label)
  + **FDs:** seat\_id → screen\_id, seat\_label

**5) Show Time**

* **show\_id** (PK)
* **movie\_id** (FK → Movie.movie\_id)
* **screen\_id** (FK → Screen.screen\_id)
* **start\_datetime**
  + **UQ:** (screen\_id, start\_datetime)
  + **FDs:** show\_id → movie\_id, screen\_id, start\_datetime

**6) Customer**

* **customer\_id** (PK)
* **full\_name**
* **email**
* **mobile\_number**
* **created\_at**
  + **UQ:** email, mobile\_number
  + **FDs:** customer\_id → full\_name, email, mobile\_number

**7) Ticket**

* **ticket\_id** (PK)
* **show\_id** (FK → Show.show\_id)
* **customer\_id** (FK → Customer.customer\_id)
* **booked\_at**
* **payment\_status**
* **total\_amount**
  + **FDs:** ticket\_id → show\_id, customer\_id

**8) TicketSeat**

* **ticket\_id** (FK → Ticket.ticket\_id)
* **show\_id** (FK → Show.show\_id)
* **seat\_id** (FK → Seat.seat\_id)
  + **PK:** (ticket\_id, seat\_id)
  + **UQ:** (show\_id, seat\_id)
  + **FDs:** (ticket\_id, seat\_id) → show\_id; and (show\_id, seat\_id) determines “booked”

**SQL Commands**

**Create commands**

CREATE TABLE movie (movie\_id BIGINT PRIMARY KEY AUTO\_INCREMENT, title VARCHAR(200) NOT NULL, language VARCHAR(50) NOT NULL, certificate ENUM('U', 'UA', 'A', 'S') DEFAULT NULL, duration\_min SMALLINT NOT NULL CHECK (duration\_min > 0), genre VARCHAR(100), release\_date DATE, imdb\_rating DECIMAL(3,1) CHECK (imdb\_rating BETWEEN 0.0 AND 10.0), UNIQUE KEY uq\_movie\_identity (title, language, release\_date));

CREATE TABLE theatre (theatre\_id BIGINT PRIMARY KEY AUTO\_INCREMENT, name VARCHAR(150) NOT NULL, address\_line VARCHAR(255), city VARCHAR(100) NOT NULL, state VARCHAR(100), pincode VARCHAR(20), UNIQUE KEY uq\_theatre\_name\_city (name, city));

CREATE TABLE screen (screen\_id BIGINT PRIMARY KEY AUTO\_INCREMENT, theatre\_id BIGINT NOT NULL, screen\_number VARCHAR(50) NOT NULL, audio\_system VARCHAR(50) NOT NULL, screen\_type VARCHAR(50) NOT NULL, UNIQUE KEY uq\_theatre\_screen\_number (theatre\_id, screen\_number), CONSTRAINT fk\_screen\_theatre FOREIGN KEY (theatre\_id) REFERENCES theatre(theatre\_id) ON DELETE CASCADE);

CREATE TABLE seat (seat\_id BIGINT PRIMARY KEY AUTO\_INCREMENT, screen\_id BIGINT NOT NULL, seat\_label VARCHAR(10) NOT NULL, UNIQUE KEY uq\_screen\_seat (screen\_id, seat\_label), CONSTRAINT fk\_seat\_screen FOREIGN KEY (screen\_id) REFERENCES screen(screen\_id) ON DELETE CASCADE);

CREATE TABLE showtime (show\_id BIGINT PRIMARY KEY AUTO\_INCREMENT, movie\_id BIGINT NOT NULL, screen\_id BIGINT NOT NULL, start\_datetime DATETIME NOT NULL, subtitle\_available BOOLEAN DEFAULT FALSE, subtitle\_language VARCHAR(50), UNIQUE KEY uq\_show\_screen\_start (screen\_id, start\_datetime), CONSTRAINT fk\_show\_movie FOREIGN KEY (movie\_id) REFERENCES movie(movie\_id) ON DELETE RESTRICT, CONSTRAINT fk\_show\_screen FOREIGN KEY (screen\_id) REFERENCES screen(screen\_id) ON DELETE RESTRICT);

CREATE TABLE customer (customer\_id BIGINT PRIMARY KEY AUTO\_INCREMENT, full\_name VARCHAR(150) NOT NULL, email VARCHAR(200) NOT NULL, mobile\_number VARCHAR(20) NOT NULL, created\_at DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP, UNIQUE KEY uq\_customer\_email (email), UNIQUE KEY uq\_customer\_mobile (mobile\_number));

CREATE TABLE ticket (ticket\_id BIGINT PRIMARY KEY AUTO\_INCREMENT, show\_id BIGINT NOT NULL, customer\_id BIGINT NOT NULL, booked\_at DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP, payment\_status ENUM('PENDING', 'PAID', 'FAILED', 'REFUNDED') NOT NULL DEFAULT 'PENDING', total\_amount DECIMAL(10,2) NOT NULL DEFAULT 0.00, CONSTRAINT fk\_ticket\_show FOREIGN KEY (show\_id) REFERENCES showtime(show\_id) ON DELETE RESTRICT, CONSTRAINT fk\_ticket\_customer FOREIGN KEY (customer\_id) REFERENCES customer(customer\_id) ON DELETE RESTRICT);

CREATE TABLE ticket\_seat (ticket\_id BIGINT NOT NULL, show\_id BIGINT NOT NULL, seat\_id BIGINT NOT NULL, PRIMARY KEY (ticket\_id, seat\_id), CONSTRAINT fk\_tseat\_ticket FOREIGN KEY (ticket\_id) REFERENCES ticket(ticket\_id) ON DELETE CASCADE, CONSTRAINT fk\_tseat\_show FOREIGN KEY (show\_id) REFERENCES showtime(show\_id) ON DELETE RESTRICT, CONSTRAINT fk\_tseat\_seat FOREIGN KEY (seat\_id) REFERENCES seat(seat\_id) ON DELETE RESTRICT, UNIQUE KEY uq\_show\_seat\_once (show\_id, seat\_id));

**Insert**

INSERT INTO movie (title, language, certificate, duration\_min, genre, release\_date, imdb\_rating) VALUES ('Inception', 'English', 'UA', 148, 'Sci-Fi', '2010-07-16', 8.8);  
INSERT INTO movie (title, language, certificate, duration\_min, genre, release\_date, imdb\_rating) VALUES ('3 Idiots', 'Hindi', 'U', 171, 'Drama', '2009-12-25', 8.4);  
INSERT INTO theatre (name, address\_line, city, state, pincode) VALUES ('PVR Cinemas', '123 Main Road', 'Chennai', 'Tamil Nadu', '600001');  
INSERT INTO theatre (name, address\_line, city, state, pincode) VALUES ('INOX', '45 Arcot Road', 'Bengaluru', 'Karnataka', '560001');  
INSERT INTO screen (theatre\_id, screen\_number, audio\_system, screen\_type) VALUES (1, 'Screen 1', 'Dolby Atmos', 'IMAX');  
INSERT INTO screen (theatre\_id, screen\_number, audio\_system, screen\_type) VALUES (2, 'Screen 2', '7.1', '3D');  
INSERT INTO seat (screen\_id, seat\_label) VALUES (1, 'A1');  
INSERT INTO seat (screen\_id, seat\_label) VALUES (1, 'A2');  
INSERT INTO seat (screen\_id, seat\_label) VALUES (2, 'B1');  
INSERT INTO seat (screen\_id, seat\_label) VALUES (2, 'B2');  
INSERT INTO showtime (movie\_id, screen\_id, start\_datetime, subtitle\_available, subtitle\_language) VALUES (1, 1, '2025-09-01 18:00:00', TRUE, 'Hindi');  
INSERT INTO showtime (movie\_id, screen\_id, start\_datetime, subtitle\_available, subtitle\_language) VALUES (2, 2, '2025-09-01 21:00:00', FALSE, NULL);  
INSERT INTO customer (full\_name, email, mobile\_number) VALUES ('Ram A', 'rama@gmail.com', '9876543210');  
INSERT INTO customer (full\_name, email, mobile\_number) VALUES ('Vishnu K', 'vishnu@yahoo.com', '9123456789');  
INSERT INTO ticket (show\_id, customer\_id, payment\_status, total\_amount) VALUES (1, 1, 'PAID', 500.00);  
INSERT INTO ticket (show\_id, customer\_id, payment\_status, total\_amount) VALUES (2, 2, 'PAID', 600.00);  
INSERT INTO ticket\_seat (ticket\_id, show\_id, seat\_id) VALUES (1, 1, 1);  
INSERT INTO ticket\_seat (ticket\_id, show\_id, seat\_id) VALUES (1, 1, 2);  
INSERT INTO ticket\_seat (ticket\_id, show\_id, seat\_id) VALUES (2, 2, 3);  
INSERT INTO ticket\_seat (ticket\_id, show\_id, seat\_id) VALUES (2, 2, 4);

**P2**

Query to list down all the shows on a given date at a given theatre along with their respective show timings.

SELECT m.title AS movie\_name,s.screen\_number,st.start\_datetime FROM showtime st JOIN movie m ON st.movie\_id=m.movie\_id JOIN screen s ON st.screen\_id=s.screen\_id JOIN theatre t ON s.theatre\_id=t.theatre\_id WHERE t.theatre\_id=1 AND DATE(st.start\_datetime)='2025-09-01' ORDER BY st.start\_datetime;

The screenshot below displays the query results based on the sample insert statements above.

