## ****Phase III: Logical Model Design****

This phase is about building a well-normalized, relational database structure with appropriate **entities**, **attributes**, **relationships**, and **constraints**, aligned with the business process you modeled in Phase II.

### Objectives:

Create an Entity-Relationship (ER) Model

Define relationships and constraints

Normalize the schema (up to 3NF)

Ensure data integrity and realistic data scenarios

## Step1:Identify Entities and Attributes

| **Entity** | **Attributes (with data types)** |
| --- | --- |
| **Movie** | movie\_id (PK), title, genre, duration, language, rating |
| **Theater** | theater\_id (PK), name, location, total\_seats |
| **Show** | show\_id (PK), movie\_id (FK), theater\_id (FK), show\_time |
| **Customer** | customer\_id (PK), full\_name, email, phone |
| **Ticket** | ticket\_id (PK), show\_id (FK), customer\_id (FK), seat\_number, price |
| **Staff** | staff\_id (PK), name, role, username, password |
| **Payment** | payment\_id (PK), ticket\_id (FK), amount, payment\_method, payment\_date |

## Step2:Define Relationships and Constraints

**Movie ↔ Show**: One-to-Many (A movie can have many shows)

**Theater ↔ Show**: One-to-Many (A theater can host many shows)

**Show ↔ Ticket**: One-to-Many (Each show has multiple tickets)

**Customer ↔ Ticket**: One-to-Many (A customer can buy multiple tickets)

**Ticket ↔ Payment**: One-to-One (Each ticket has a payment)

**Staff** is standalone for admin purposes (used in auditing, Phase VII)

### Constraints:

**Primary Keys:** Defined for all tables

**Foreign Keys:** Enforce referential integrity

**NOT NULL:** On essential fields (e.g., movie title, seat number)

**UNIQUE:** On seat\_number within the same show

**CHECK:** On ticket price (> 0)

**DEFAULT:** payment\_method defaults to 'cash' if not provided

## Step 3: Normalization (up to 3NF)

### Example: Movie and Show

**1NF:** All attributes have atomic values  
**2NF:** No partial dependencies (e.g., show\_time is only in Show, not in Movie)  
**3NF:** No transitive dependencies (e.g., theater location isn't dependent on movie ID)

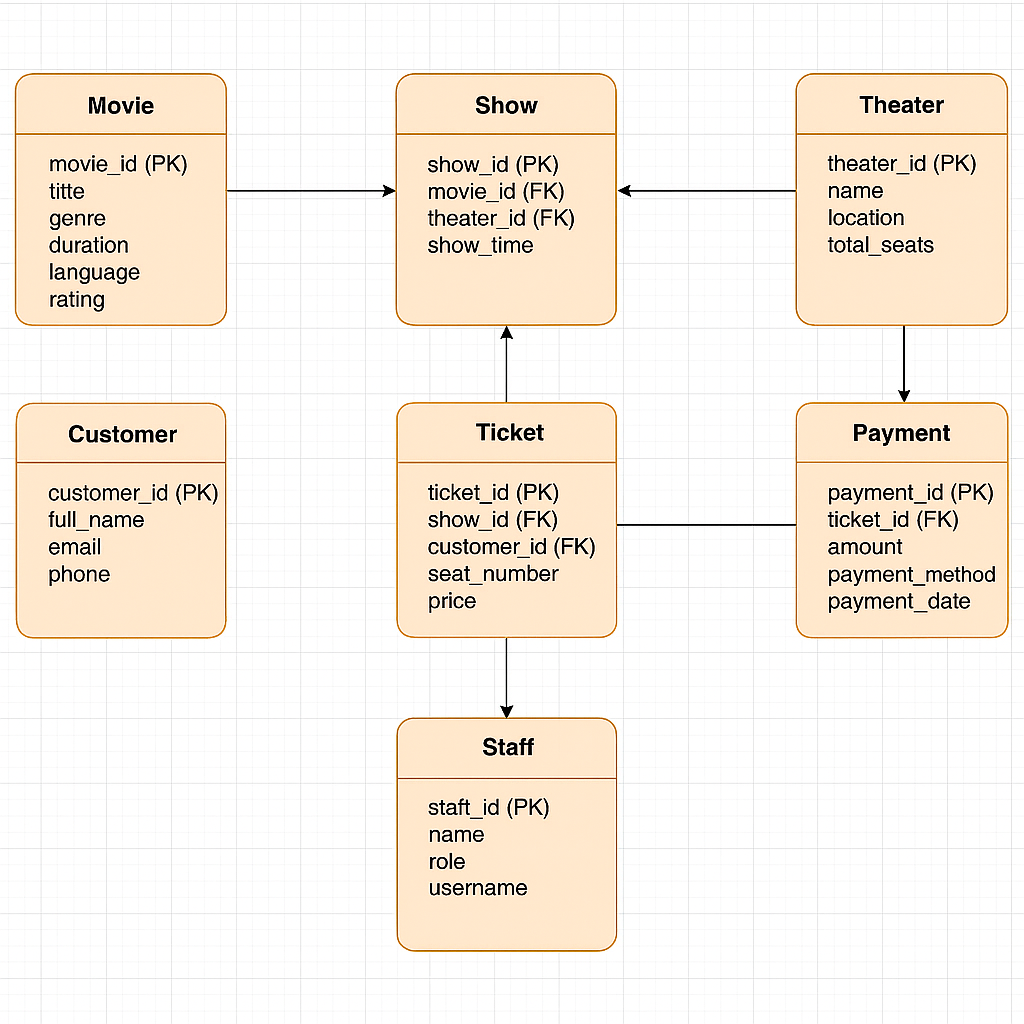
## 🧾 Step 4: Data Scenarios

A customer books 2 tickets for 2 different shows.

A theater hosts 3 shows for different movies in one day.

Payment for each ticket is stored individually with method and timestamp.

Tickets can't be issued for the same seat in a show.

**RELATED IMAGE  
  
**