**1. Introduction**

This project is a cinema ticket booking management system where users can browse available movies, select seats, and make bookings through an organized database. The goal is to design and create a database that supports the system’s core operations while ensuring data integrity and efficiency.

**2. Requirements Analysis**

**Users**

* **Customers:** Users who book tickets.
* **Cinema Managers:** Responsible for managing cinema and movie data.

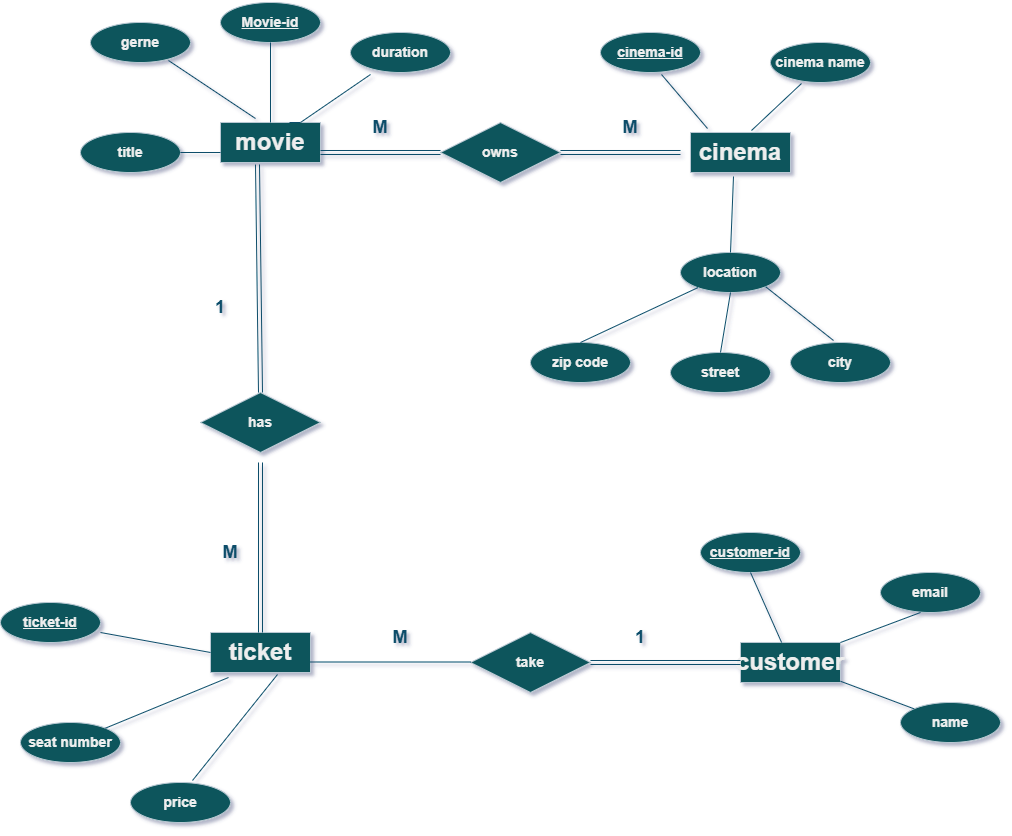
**Entities**

* **Movie:** Contains movie details such as title, genre, and duration.
* **Cinema:** Contains cinema data such as name and location.
* **Customer:** Customer data such as name and email.
* **Ticket:** Details of booked tickets, linked to movie and customer.

**Relationships**

* Each ticket is linked to a movie and a customer.
* A cinema can contain multiple movies.
* **3. Database Design**
* **ER Diagram, Tables, and Fields**  
  The **movie** table contains: movie\_id (primary key), title, genre, duration.
* The **cinema** table contains: cinema\_id, cinema\_name, zipcode, street, city.
* The **customer** table contains: customer\_id, name, email.
* The **ticket** table contains: ticket\_id, seat\_number, price, movie\_id (foreign key), customer\_id (foreign key).

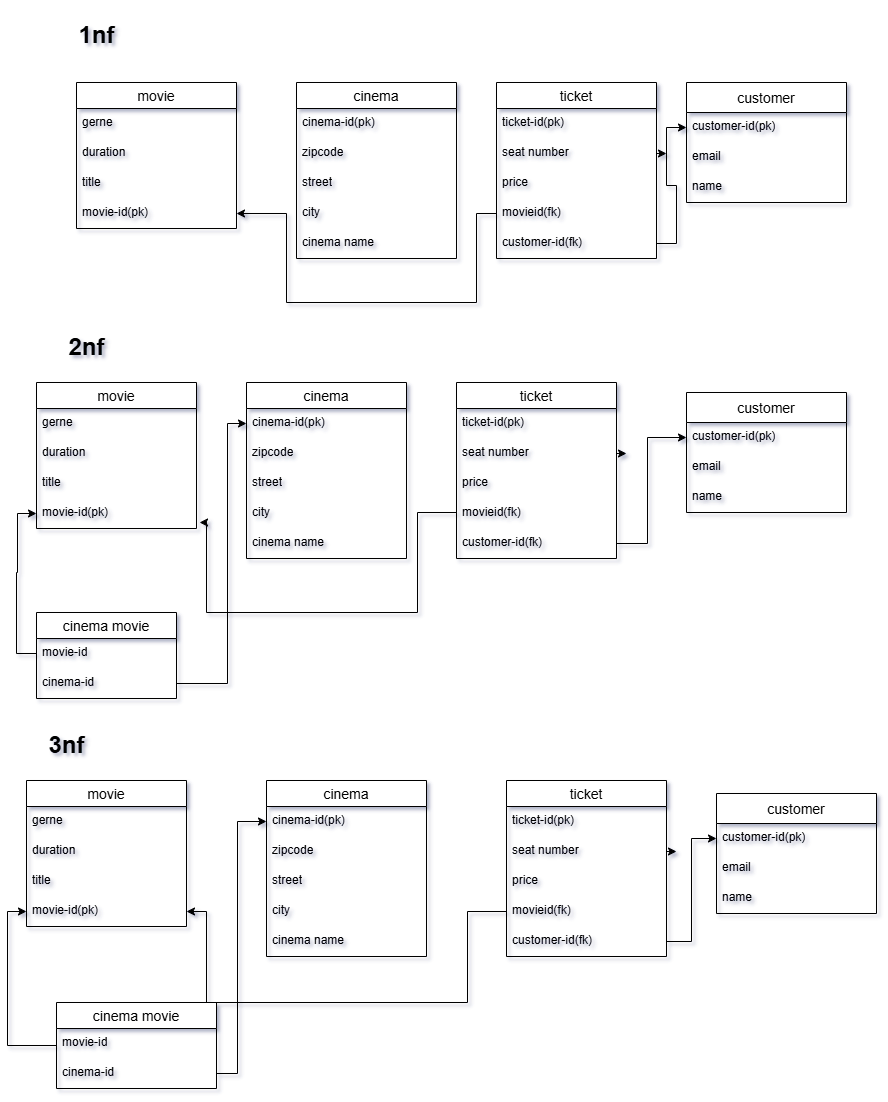
.



1. Normalization  
   **Normalization** rules were applied to ensure data organization and reduce redundancy.

The tables are designed according to the Third Normal Form (3NF), where each table contains data related to a single entity only.

Primary keys and foreign keys were used to maintain integrity between the tables.



5.Data Entry  
At least 10 records were entered in each table to cover data for cinemas, movies, customers, and tickets. Data accuracy and relationships between tables were verified.

6.Queries  
Several queries were executed to meet the system requirements:

Text search: Find movies that contain a specific word in the title.

SELECT \* FROM movie WHERE title LIKE '%Dark%';

Calculate average price:

SELECT AVG(price) AS average\_price FROM ticket;

Order movies by duration:

SELECT \* FROM movie ORDER BY duration ASC;

Join between tables:

SELECT c.name, m.title, t.seat\_number, t.price

FROM ticket t

JOIN customer c ON t.customer\_id = c.customer\_id

JOIN movie m ON t.movie\_id = m.movie\_id;

Query with a price condition:

SELECT \* FROM ticket WHERE price > 55;

Conclusion:

In this project, an efficient database was designed and implemented for a cinema ticket booking system, focusing on data organization and ensuring integrity through the application of normalization rules and the use of primary and foreign keys. Multiple queries were developed to meet the system’s core needs such as searching, calculation, and sorting, which enhance the ease of data management and analysis. This project reflects the importance of good database design in supporting information systems and providing a smooth and reliable user experience.