Introduction

Abstract:

The Movie Ticket Management System is devised to address the intricacies of the entertainment industry, particularly in the context of movie ticket sales and customer experience. This system incorporates various entities such as moviegoers, cinemas, and can include cinema operators, which play a pivotal role in enhancing the cinematic experience through curated movie packages. Movie enthusiasts can conveniently access and purchase tickets, leveraging the diverse packages offered by cinema operators.

Objectives:

Our main objectives for the Movie Ticket Management System are :

- **Efficient Ticket Sales:** Develop a user-friendly interface that enables moviegoers to conveniently browse, select, and purchase tickets for various movies and screening times.

  Market trends: Implement a feature for cinema operators to create and modify movie packages, allowing for flexibility in pricing, offerings, and promotions to cater to diverse customer preferences.

- **Effective Communication:** Facilitate seamless communication between moviegoers and cinema operators, ensuring timely updates on movie schedules, promotions, and any changes to screening venues or times.

- **Logistics Management:** Enable cinema operators to efficiently manage logistics related to screening venues, transportation, and accommodation, ensuring a smooth and well-coordinated movie-watching experience.

- **Data Security and Privacy:** Implement robust security measures to protect customer information, payment details, and other sensitive data, ensuring compliance with privacy regulations.

- **User Analytics**: Incorporate analytics tools to gather insights into customer preferences, popular movie choices, and booking patterns, providing valuable data for marketing strategies and system improvements.
- **Customer Satisfaction:** Prioritize customer satisfaction by offering a responsive and user-friendly platform, addressing customer feedback, and continually improving the overall moviegoing experience.
- **Integration with External Services:** Integrate the system with external services such as online payment gateways, transportation providers, and hotel accommodations to create a comprehensive and integrated movie ticket management ecosystem.

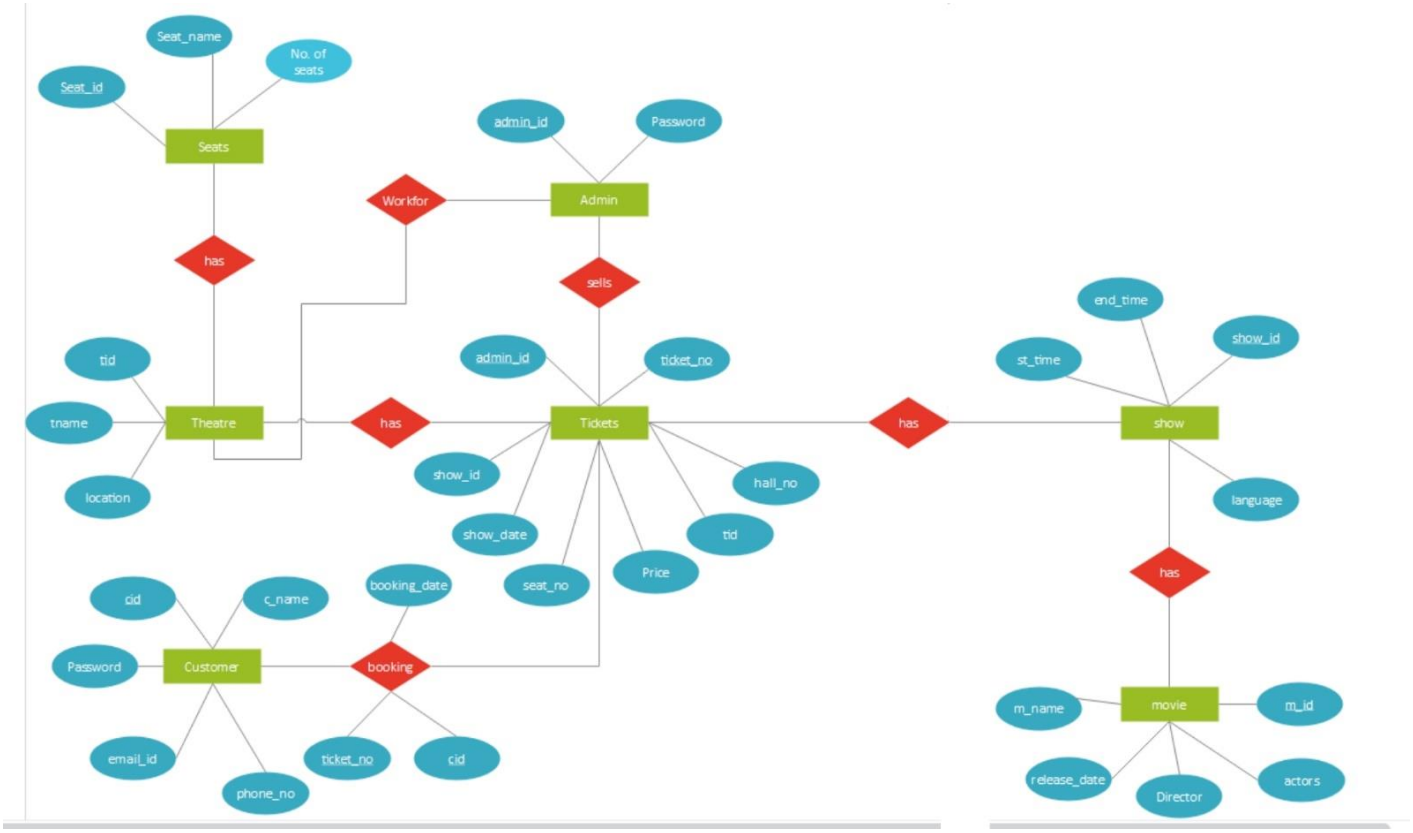
Description:

Movie Ticket Management System' is designed to address and enhance the operational efficiency of software developed for the entertainment industry, providing users with the capability to generate or view pricing offered by cinema operators and update relevant information as needed.

Customers interacting with the system can choose from a range of menus tailored to meet their specific needs, including options such as Customer, Cinema Operator, Movie Packages, Movies, Screening Schedules, Maps, Hotels, and Transportation. Each menu option directs users to dedicated pages where they can seamlessly update, delete, or add information to the server.

The system allows customers to explore movie options, select preferred cinema operators, and choose from diverse movie packages. Cinema operators have the flexibility to manage movie packages, update screening schedules, and customize pricing structures. This dynamic interaction between customers and cinema operators ensures a smooth and enjoyable movie ticket booking experience

ER DIAGRAM


Seat_name
No. of seats
Seat_id
Seats
admin_id
Password
Admin
Workfor
has
sells
end_time
show_id
st_time
tid
admin_id
ticket_no
tname
Theatre
has
Tickets
has
show
show_id
hall_no
location
language
show_date
tid
Price
has
booking_date
seat_no
cid
c_name
Password
Customer
booking
m_name
movie
m_id
email_id
ticket_no
cid
release_date
Director
actors
phone_no

Requirements:

Hardware Requirements:

- A desktop or laptop with a proper internet connection
- 500GB of storage for optimal operation
- Minimum of 4GB ram
- Any operating system is compatible

Software Specifications:

- Programming language: Python
- Database: SQL 8.0.35
- OS: windows7/8/10 or any OS

Database:

- **MySQL:** It is a relational database management system that operates on the foundation of SQL, or Structured Query Language. While its applications are diverse, MySQL is most commonly employed as a web database. It proves invaluable in storing a spectrum of data, ranging from a singular record of information to the comprehensive inventory records of information.

Backend:

- **Python:** For backend connectivity with Python and MySQL, mysql-connector-python library emerges as a pivotal tool. It serves as a bridge between Python and MySQL databases, allowing developers to execute SQL queries, fetch results, and manage database connections with ease.

Code Implementation:

Table creation:

1. Seats Table :

```
CREATE TABLE Seats (
  seat_id INT PRIMARY KEY,
  seat_name VARCHAR(255),
  no_of_seats INT
);
```

- seat_id: Unique identifier for each seat.
- seat_name: Descriptive name for the seat.
- no_of_seats: The total number of seats available.

```
mysql> CREATE TABLE Seats (
    ->     seat_id INT PRIMARY KEY,
    ->     seat_name VARCHAR(255),
    ->     no_of_seats INT
    -> );
Query OK, 0 rows affected (0.18 sec)
```

2. Theatre Table :

```
CREATE TABLE Theatre (
  theatre_id INT PRIMARY KEY,
  theatre_name VARCHAR(255),
  location VARCHAR(255),
  seat_id INT,
  FOREIGN KEY (seat_id) REFERENCES Seats(seat_id)
);
```

- theatre_id: Unique identifier for each theatre.
- theatre_name: The name of the theatre.
- location: Location or address of the theatre.
- seat_id: Foreign key referencing the seat_id in the Seats table. Represents the seating configuration of the theatre

```
mysql> CREATE TABLE Theatre (
    ->     theatre_id INT PRIMARY KEY,
    ->     theatre_name VARCHAR(255),
    ->     location VARCHAR(255),
    ->     seat_id INT,
    ->     FOREIGN KEY (seat_id) REFERENCES Seats(seat_id)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

3. Admin Table :

```
CREATE TABLE Admin (
   admin_id INT PRIMARY KEY,
   admin_password VARCHAR(255)
);
```

- admin_id: Unique identifier for each admin.
- admin_password: Password associated with the admin's account.

```
mysql> CREATE TABLE Admin (
    ->     admin_id INT PRIMARY KEY,
    ->     admin_password VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

4. Customer Table :

```
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    email_id VARCHAR(255),
    phone_num VARCHAR(15)
);
```

- customer_id: Unique identifier for each customer.
- customer_name: Name of the customer.
- email_id: Email address of the customer.
- phone_num: Phone number of the customer.

```
mysql> CREATE TABLE Customer (
    ->     customer_id INT PRIMARY KEY,
    ->     customer_name VARCHAR(255),
    ->     email_id VARCHAR(255),
    ->     phone_num VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

5. CustomerCredentials Table :

```
CREATE TABLE CustomerCredentials (
    customer_id INT PRIMARY KEY,
    customer_password VARCHAR(255),
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);
```

- customer_id: Foreign key referencing the customer_id in the Customer table. Represents the link between customer credentials and customer information.

- customer_password: Password associated with the customer's account.

```
mysql> CREATE TABLE CustomerCredentials (
    ->     customer_id INT PRIMARY KEY,
    ->     customer_password VARCHAR(255),
    ->     FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

6. Show Table :

CREATE TABLE Show (

show_id INT PRIMARY KEY,

movie_name VARCHAR(255),

start_time TIME,

end_time TIME

);

- show_id: Unique identifier for each show.
- movie_name: Movie name associated with the show.
- start_time: Start time of the show.
- end_time: End time of the show.
- am_pm: Indicator for whether the show is in the morning (AM) or afternoon/evening (PM).

```
mysql> CREATE TABLE Shows (
    ->     show_id INT PRIMARY KEY,
    ->     movie_name VARCHAR(255),
    ->     start_time INT,
    ->     end_time INT,
    ->     am_pm INT
    -> );
Query OK, 0 rows affected (0.03 sec)
```

7. Movie Table :

```
CREATE TABLE Movie (
    movie_id INT PRIMARY KEY,
    movie_name VARCHAR(255),
    date DATE,
    director VARCHAR(255),
    actor VARCHAR(255)
);
```

- movie_id: Unique identifier for each movie.
- movie_name: Name of the movie.
- date: Release date of the movie.
- director: Director of the movie.
- actor: Lead actor in the movie.

```
mysql> CREATE TABLE Movie (
    ->     movie_id INT PRIMARY KEY,
    ->     movie_name VARCHAR(255),
    ->     date DATE,
    ->     director VARCHAR(255),
    ->     actor VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

8. Tickets Table :

```
CREATE TABLE Tickets (
    ticket_id INT PRIMARY KEY,
    admin_id INT,
    show_id INT,
    show_date DATE,
    seat_id INT,
```

```
    price DECIMAL(10,2),
    theatre_id INT,
    FOREIGN KEY (admin_id) REFERENCES Admin(admin_id),
    FOREIGN KEY (show_id) REFERENCES Shows(show_id),
    FOREIGN KEY (seat_id) REFERENCES Seats(seat_id),
    FOREIGN KEY (theatre_id) REFERENCES Theatre(theatre_id)
);
```

- ticket_id: Unique identifier for each ticket.
- admin_id: Foreign key referencing the admin_id in the Admin table. Represents the admin associated with the ticket.
- show_id: Foreign key referencing the show_id in the Shows table. Represents the show for which the ticket is booked.
- show_date: Date of the show.
- seat_id: Foreign key referencing the seat_id in the Seats table. Represents the seat booked for the ticket.
- price: Price of the ticket.
- theatre_id: Foreign key referencing the theatre_id in the Theatre table. Represents the theatre associated with the ticket.

```
mysql> CREATE TABLE Tickets (
    ->     ticket_id INT PRIMARY KEY,
    ->     admin_id INT,
    ->     show_id INT,
    ->     show_date DATE,
    ->     seat_id INT,
    ->     price DECIMAL(10,2),
    ->     theatre_id INT,
    ->     FOREIGN KEY (admin_id) REFERENCES Admin(admin_id),
    ->     FOREIGN KEY (show_id) REFERENCES Shows(show_id),
    ->     FOREIGN KEY (seat_id) REFERENCES Seats(seat_id),
    ->     FOREIGN KEY (theatre_id) REFERENCES Theatre(theatre_id)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

Values Insertion:

1. Insertion in Seats Table :

INSERT INTO SEATS VALUES

(1,'LOWER',20),

(2,'UPPER',30),

(3,'PREMIUM',5),

(4,'BALCONY',10),

(5,'VIP',3);

```
mysql> INSERT INTO SEATS VALUES
    -> (1,'LOWER',20),
    -> (2,'UPPER',30),
    -> (3,'PREMIUM',5),
    -> (4,'BALCONY',10),
    -> (5,'VIP',3);
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

2. Insertion in Theatre Table :

INSERT INTO THEATRE VALUES

(1310,'SANGAM SARAT', 'VISAKHAPATNAM', 4),

(1311,'MELODY', 'VISAKHAPATNAM', 3),

(1312,'SVC CINEMAS', 'VIZIANAGARAM', 1),

(1313,'RAMULAMMA', 'TAGARAPUVALASA', 2),

(1314,'GANESH', 'TAGARAPUVALASA', 2);

```
mysql> INSERT INTO THEATRE VALUES
    -> (1310,'SANGAM SARAT', 'VISAKHAPATNAM', 4),
    -> (1311,'MELODY', 'VISAKHAPATNAM', 3),
    -> (1312,'SVC CINEMAS', 'VIZIANAGARAM', 1),
    -> (1313,'RAMULAMMA', 'TAGARAPUVALASA', 2),
    -> (1314,'GANESH', 'TAGARAPUVALASA', 2);
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

3. Insertion in Admin Table :

INSERT INTO ADMIN VALUES

(1310,'sangam@vsp'),

(1311,'melody@vsp'),

(1312,'svc@vzm'),

(1313,'ramulamma@tgp'),

(1314,'ganesh@tgp');

```
mysql> INSERT INTO ADMIN VALUES
    -> (1310,'sangam@vsp'),
    -> (1311,'melody@vsp'),
    -> (1312,'svc@vzm'),
    -> (1313,'ramulamma@tgp'),
    -> (1314,'ganesh@tgp');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

4. Insertion in Shows Table :

INSERT INTO SHOWS VALUES

(101, 'HINANNA', 2, 5, 1),

(100, 'HINANNA', 10, 1, 0),

(102, 'ANIMAL', 10, 1, 0),

(103, 'ANIMAL', 2, 5, 1),

(104, 'SALAAR', 2, 5, 1);

```
mysql> INSERT INTO SHOWS VALUES
    -> (101, 'HINANNA', 2, 5, 1),
    -> (100, 'HINANNA', 10, 1, 0),
    -> (102, 'ANIMAL', 10, 1, 0),
    -> (103, 'ANIMAL', 2, 5, 1),
    -> (104, 'SALAAR', 2, 5, 1);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

5. Insertion in Movie Table :

```
INSERT INTO MOVIE VALUES
(200, 'HINANNA', '2023-12-07', 'SHOURYUV', 'NANI'),
(201, 'ANIMAL', '2023-12-01', 'SANDEEP REDDY VANGA', 'RANBIR KAPOOR'),
(202, 'SALAAR', '2023-12-22', 'PRASHANTH NEEL', 'PRABHAS'),
(203, 'EXTRA ORDINARY MAN', '2023-12-08', 'VAKKANTHAM VAMSI', 'NITHIN');
```

```
mysql> INSERT INTO MOVIE VALUES
    -> (200, 'HINANNA', '2023-12-07', 'SHOURYUV', 'NANI'),
    -> (201, 'ANIMAL', '2023-12-01', 'SANDEEP REDDY VANGA', 'RANBIR KAPOOR'),
    -> (202, 'SALAAR', '2023-12-22', 'PRASHANTH NEEL', 'PRABHAS'),
    -> (203, 'EXTRA ORDINARY MAN', '2023-12-08', 'VAKKANTHAM VAMSI', 'NITHIN');
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

Altering the table :

Modifying the datatype of a column :

```
ALTER TABLE CUSTOMER
MODIFY COLUMN customer_id VARCHAR(255),
DROP PRIMARY KEY,
ADD PRIMARY KEY (customer_id);
```

```
mysql> ALTER TABLE CUSTOMER
    -> MODIFY COLUMN customer_id VARCHAR(255),
    -> DROP PRIMARY KEY,
    -> ADD PRIMARY KEY (customer_id);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Dropping Table :

Dropping the entire table:

DROP TABLE customercredentials;

```
mysql> DROP TABLE customercredentials;
Query OK, 0 rows affected (0.02 sec)
```

Deleting rows in the table :

Deleting specific rows in the table:

DELETE FROM SEATS WHERE SEAT_ID = 1;

```
mysql> DELETE FROM SEATS WHERE SEAT_ID = 1;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM SEATS WHERE SEAT_ID = 2;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM SEATS WHERE SEAT_ID = 3;
Query OK, 1 row affected (0.00 sec)

mysql> DELETE FROM SEATS WHERE SEAT_ID = 4;
Query OK, 1 row affected (0.00 sec)

mysql> DELETE FROM SEATS WHERE SEAT_ID = 5;
Query OK, 1 row affected (0.00 sec)
```

Displaying Tables:

1. Seats Table :

SELECT * FROM SEATS;

```
mysql>  SELECT * FROM SEATS;
+---------+-----------+-------------+
| seat_id | seat_name | no_of_seats |
+---------+-----------+-------------+
|       1 | LOWER     |          20 |
|       2 | UPPER     |          30 |
|       3 | PREMIUM   |           5 |
|       4 | BALCONY   |          10 |
|       5 | VIP       |           3 |
+---------+-----------+-------------+
5 rows in set (0.01 sec)
```

2. Theatre Table :

SELECT * FROM THEATRE;

```
mysql>  SELECT * FROM THEATRE;
+------------+--------------+----------------+---------+
| theatre_id | theatre_name | location       | seat_id |
+------------+--------------+----------------+---------+
|       1310 | SANGAM SARAT | VISAKHAPATNAM  |       4 |
|       1311 | MELODY       | VISAKHAPATNAM  |       3 |
|       1312 | SVC CINEMAS  | VIZIANAGARAM   |       1 |
|       1313 | RAMULAMMA    | TAGARAPUVALASA |       2 |
|       1314 | GANESH       | TAGARAPUVALASA |       2 |
+------------+--------------+----------------+---------+
5 rows in set (0.01 sec)
```

3. Admin Table :

SELECT * FROM ADMIN;

```
mysql>  SELECT * FROM ADMIN;
+----------+----------------+
| admin_id | admin_password |
+----------+----------------+
|     1310 | sangam@vsp     |
|     1311 | melody@vsp     |
|     1312 | svc@vzm        |
|     1313 | ramulamma@tgp  |
|     1314 | ganesh@tgp     |
+----------+----------------+
5 rows in set (0.01 sec)
```

4. Shows Table :

SELECT * FROM SHOWS;

```
mysql>  SELECT * FROM SHOWS;
+---------+------------+------------+----------+-------+
| show_id | movie_name | start_time | end_time | am_pm |
+---------+------------+------------+----------+-------+
|     100 | HINANNA    |         10 |        1 |     0 |
|     101 | HINANNA    |          2 |        5 |     1 |
|     102 | ANIMAL     |         10 |        1 |     0 |
|     103 | ANIMAL     |          2 |        5 |     1 |
|     104 | SALAAR     |          2 |        5 |     1 |
+---------+------------+------------+----------+-------+
5 rows in set (0.01 sec)
```

5. Movie Table :

SELECT * FROM MOVIE;

```
mysql>  SELECT * FROM MOVIE;
+----------+-------------------+------------+--------------------+--------------+
| movie_id | movie_name        | date       | director           | actor        |
+----------+-------------------+------------+--------------------+--------------+
|      200 | HINANNA           | 2023-12-07 | SHOURYUV           | NANI         |
|      201 | ANIMAL            | 2023-12-01 | SANDEEP REDDY VANGA | RANBIR KAPOOR |
|      202 | SALAAR            | 2023-12-22 | PRASHANTH NEEL     | PRABHAS      |
|      203 | EXTRA ORDINARY MAN | 2023-12-08 | VAKKANTHAM VAMSI   | NITHIN       |
+----------+-------------------+------------+--------------------+--------------+
4 rows in set (0.01 sec)
```

Description of Tables:

1. Seats Table :

DESC SEATS;

```
mysql> DESC SEATS;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| seat_id     | int          | NO   | PRI | NULL    |       |
| seat_name   | varchar(255) | YES  |     | NULL    |       |
| no_of_seats | int          | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.06 sec)
```

2. Theatre Table :

DESC TABLE;

```
mysql> DESC THEATRE;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| theatre_id   | int          | NO   | PRI | NULL    |       |
| theatre_name | varchar(255) | YES  |     | NULL    |       |
| location     | varchar(255) | YES  |     | NULL    |       |
| seat_id      | int          | YES  | MUL | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

3. Admin Table :

DESC ADMIN;

```
mysql> DESC ADMIN;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| admin_id       | int          | NO   | PRI | NULL    |       |
| admin_password | varchar(255) | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

4. Shows Table :

DESC SHOWS;

```
mysql> DESC SHOWS;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| show_id    | int         | NO   | PRI | NULL    |       |
| movie_name | varchar(255) | YES  |     | NULL    |       |
| start_time | int         | YES  |     | NULL    |       |
| end_time   | int         | YES  |     | NULL    |       |
| am_pm      | int         | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

5. Movie Table :

DESC MOVIE;

```
mysql> DESC MOVIE;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| movie_id   | int         | NO   | PRI | NULL    |       |
| movie_name | varchar(255) | YES  |     | NULL    |       |
| date       | date        | YES  |     | NULL    |       |
| director   | varchar(255) | YES  |     | NULL    |       |
| actor      | varchar(255) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

6. Customer Table :

DESC CUSTOMER;

```
mysql> DESC CUSTOMER;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| customer_id   | varchar(255) | NO   | PRI | NULL    |       |
| customer_name | varchar(255) | YES  |     | NULL    |       |
| email_id      | varchar(255) | YES  |     | NULL    |       |
| phone_num     | varchar(15)  | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

7. CustomerCredentials Table :

DESC CUSTOMERCREDENTIALS;

```
mysql> DESC CUSTOMERCREDENTIALS;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| customer_id      | varchar(255) | NO   | PRI | NULL    |       |
| customer_password | varchar(255) | YES  |     | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

8. Tickets Table :

DESC TICKETS;

```
mysql> DESC TICKETS;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| ticket_id  | int           | NO   | PRI | NULL    |       |
| admin_id   | int           | YES  | MUL | NULL    |       |
| show_id    | int           | YES  | MUL | NULL    |       |
| show_date  | date          | YES  |     | NULL    |       |
| seat_id    | int           | YES  | MUL | NULL    |       |
| price      | decimal(10,2) | YES  |     | NULL    |       |
| theatre_id | int           | YES  | MUL | NULL    |       |
+------------+---------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Normalization Check:

To check the normalization level of our database, we need to analyze each table and its dependencies on primary keys and foreign keys. Let's go through each table:

Seats Table (1NF and 2NF):

- It has a primary key (seat_id).
- No repeating groups or arrays are present.
- No partial dependencies.
- It is in 1NF and 2NF.

Theatre Table (1NF, 2NF, and 3NF):

- It has a primary key (theatre_id).
- It has foreign key references (seat_id) that are part of the candidate key.
- It is in 1NF, 2NF, and 3NF.

Admin Table (1NF):

- It has a primary key (admin_id).
- It is in 1NF.

Customer Table (1NF):

- It has a primary key (customer_id).
- It is in 1NF.

CustomerCredentials Table (1NF and 2NF):

- It has a primary key (customer_id).
- It has a foreign key reference (customer_id) that is part of the candidate key.
- It is in 1NF and 2NF.

Movie Table (1NF):

- It has a primary key (movie_id).
- It is in 1NF.

Shows Table (1NF, 2NF, and 3NF):

- It has a primary key (show_id).
- It is in 1NF, 2NF, and 3NF.

Tickets Table (1NF, 2NF, and 3NF):

- It has a primary key (ticket_id).
- It has foreign key references (admin_id, show_id, seat_id, theatre_id) that are part of the candidate key.
- It is in 1NF, 2NF, and 3NF.

The database appears to be in at least 1NF, 2NF, and 3NF. This indicates that the tables are free of redundancy and follow normalization principles up to the third normal form. Please note that higher normal forms (4NF, 5NF) are usually considered when dealing with more complex scenarios, and their application depends on specific requirements.

Integrating with Python:

After integrating our database with Python using the SQLConnector library, we utilize the following functions to execute various operations.

1. DisplayTable() :

Operations:

- Establishes a connection to the database.
- Creates a cursor to execute SQL queries.
- Selects all rows from the specified table.
- Fetches and displays all rows or indicates if no data is found.
- Closes the cursor and the database connection.

2. ConnectDatabase():

Operations:

- Attempts to connect to the MySQL database with specified credentials.
- Returns the database connection object if successful, otherwise returns None.

3. CustomerLogin():

Operations:

- Gets customer email and password from the user.
- Connects to the database.
- Creates a cursor to execute SQL queries.
- Selects the customer ID from the 'CustomerCredentials' table based on provided email and password.
- Returns the customer ID if login is successful, otherwise returns None.
- Closes the cursor and the database connection.

4. CustomerRegistration():

Operations:

- Gets customer details from the user.
- Connects to the database.
- Creates a cursor to execute SQL queries.
- Inserts customer details into the 'Customer' and 'CustomerCredentials' tables.
- Commits the changes to the database if successful, otherwise rolls back.
- Closes the cursor and the database connection.

5. BookTicket():

Operations:

- Collects necessary ticket details from the user.
- Connects to the database.
- Creates a cursor to execute SQL queries.
- Gets the current maximum ticket_id from the 'Tickets' table.
- Generates a new ticket ID.
- Inserts ticket details into the 'Tickets' table.
- Commits the changes to the database if successful, otherwise rolls back.
- Displays the booked ticket details.
- Closes the cursor and the database connection.

6. DisplayTicket():

Operations:

- Connects to the database.
- Creates a cursor to execute SQL queries.
- Selects ticket details for the specified ticket_id from the 'Tickets' table.
- Fetches and displays the ticket details or indicates if not found.

Running the application in Teriminal:

1. Printing Menu:

```
    MENU


Available 'Seats' :
(1, 'LOWER', 20)
(2, 'UPPER', 30)
(3, 'PREMIUM', 5)
(4, 'BALCONY', 10)
(5, 'VIP', 3)

====================================

Available 'Theatre' :
(1310, 'SANGAM SARAT', 'VISAKHAPATNAM', 4)
(1311, 'MELODY', 'VISAKHAPATNAM', 3)
(1312, 'SVC CINEMAS', 'VIZIANAGARAM', 1)
(1313, 'RAMULAMMA', 'TAGARAPUVALASA', 2)
(1314, 'GANESH', 'TAGARAPUVALASA', 2)

====================================

Available 'Movie' :
(200, 'HINANNA', datetime.date(2023, 12, 7), 'SHOURYUV', 'NANI')
(201, 'ANIMAL', datetime.date(2023, 12, 1), 'SANDEEP REDDY VANGA', 'RANBIR KAPOOR')
(202, 'SALAAR', datetime.date(2023, 12, 22), 'PRASHANTH NEEL', 'PRABHAS')
(203, 'EXTRA ORDINARY MAN', datetime.date(2023, 12, 8), 'VAKKANTHAM VAMSI', 'NITHIN')

====================================

Available 'Shows' :
(100, 'HINANNA', 10, 1, 0)
(101, 'HINANNA', 2, 5, 1)
(102, 'ANIMAL', 10, 1, 0)
(103, 'ANIMAL', 2, 5, 1)
(104, 'SALAAR', 2, 5, 1)

====================================
```

2. When user attempts to login with no credentials in database :

```
==========================================

Enter your email: nameisjoseph@gmail.com
Enter your password: joseph110
Invalid login credentials. Please try again.
Do you want to create an account? (yes/no): yes
```

Reasons:

There are no credentials in our database with that respective user_id and password. So it shows Invalid Credentials.

```
mysql> SELECT * FROM CUSTOMERCREDENTIALS;
Empty set (0.00 sec)
```

3. User Registeration:

```
========================================

Enter your email: nameisjoseph@gmail.com
Enter your password: joseph110
Invalid login credentials. Please try again.
Do you want to create an account? (yes/no): yes
Enter your name: Joseph Prasad
Enter your email: nameisjoseph@gmail.com
Create a password: joseph110
Enter your phone number: 7337217630
Enter your username: nameisjoseph
Registration successful. You can now log in.
```

After Registration tables are:

```
mysql> SELECT * FROM CUSTOMERCREDENTIALS;
+-----------------------+-------------------+
| customer_id           | customer_password |
+-----------------------+-------------------+
| nameisjoseph@gmail.com | joseph110        |
+-----------------------+-------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM CUSTOMER;
+-------------+--------------+-----------------------+------------+
| customer_id | customer_name | email_id              | phone_num  |
+-------------+--------------+-----------------------+------------+
| nameisjoseph | Joseph Prasad | nameisjoseph@gmail.com | 7337217630 |
+-------------+--------------+-----------------------+------------+
1 row in set (0.00 sec)
```

4. Ticket Booking :

```
========================================

Enter your email: nameisjoseph@gmail.com
Enter your password: joseph110
Enter the show ID: 100
Enter the show date (YYYY-MM-DD): 2023-12-07
Enter the seat ID: 4
Enter the ticket price: 1000
Enter the theatre ID: 1310
Ticket booked successfully.


------------------------------------------------
```

After Booking tables are:

```
mysql> SELECT * FROM TICKETS;
+-----------+----------+---------+------------+---------+---------+------------+
| ticket_id | admin_id | show_id | show_date  | seat_id | price   | theatre_id |
+-----------+----------+---------+------------+---------+---------+------------+
|      2000 |     1310 |     100 | 2023-12-07 |       4 | 1000.00 |       1310 |
+-----------+----------+---------+------------+---------+---------+------------+
1 row in set (0.00 sec)
```

5. Displaying Ticket :

```
------------------------------------------------

Ticket Details:
Ticket ID: 2001
Admin ID: 1310
Show ID: 100
Show Date: 2023-12-07
Seat ID: 4
Price: 1000.00
Theatre ID: 1310
------------------------------------------------
```

CONCLUSION:

The Movie Ticket Booking System is now acknowledged as a burgeoning global industry, mirroring the growth trends observed in various sectors. Our application plays a pivotal role in database management within this industry. Featuring a user-friendly environment, it fosters seamless connectivity among customers, streamlining processes and saving both time and effort. The application proves invaluable for tour managers, empowering them to oversee and manage tour-related activities with utmost efficiency.

As a conclusion, our implemented system not only enhances the operational efficiency of movie ticket booking processes but also lays the groundwork for potential expansions. Further modifications could involve integrating the system with larger organizations, such as movie production companies or cinema chains, to facilitate and streamline their operations. The adaptability of our system positions it as a valuable tool not only for individual users but also for broader industry applications, contributing to the overall growth and efficiency of the movie ticket booking ecosystem.