Online Movie Reservation

Course Section: CS605.641.81
Fall, 2021

Prepared by

Dongjun Cho
11/20/2021

Table of Contents

1. INTRODUCTION 3

1.1. SCOPE AND PURPOSE OF DOCUMENT 3
1.2. PROJECT OBJECTIVE 3

2. SYSTEM REQUIREMENTS 3

2.1 HARDWARE REQUIREMENTS 3
2.2 SOFTWARE REQUIREMENTS 3
2.3 FUNCTIONAL REQUIREMENTS 4
2.4 DATABASE REQUIREMENTS 4

3. DATABASE DESIGN DESCRIPTION 4

3.1 DESIGN RATIONALE 4
3.2 E/R MODEL 5
3.2.1 Entities 5
3.2.2 Relationships 6
3.2.3 E/R Diagram 8
3.3 RELATIONAL MODEL 8
3.3.1 Data Dictionary 8
3.3.2 Integrity Rules 12
3.3.3 Operational Rules 12
3.3.4 Operations 12
3.4 SECURITY 12
3.5 DATABASE BACKUP AND RECOVERY 12
3.6 USING DATABASE DESIGN OR CASE TOOL 13
3.7 OTHER POSSIBLE E/R RELATIONSHIPS 13

4. IMPLEMENTATION DESCRIPTION 13

4.1 DATA DICTIONARY 13
4.2 ADVANCED FEATURES 16
4.3 QUERIES 18
4.3.1 Show Ticket information for Customer #1 18
4.3.2 List the customer who lives in San Jose 18
4.3.3 Payment Information of customer's first name Hamilton 18
4.3.4 List all movie by genre type 19
4.3.5 List movie by watching standard level 19
4.3.6 List movies that are screening in the morning 20
4.3.7 List of movies that customers bought the most. 20
4.3.8 List of the movie theaters in the city with the most customers. 21

5. CRUD MATRIX 21

5.1 LIST OF ENTITY TYPES 22
5.2 LIST OF FUNCTIONS 22

6. CONCLUDING REMARKS 23

APPENDICES 24

REFERENCES 32

1. Introduction

Many people like to watch movies. (I also like to watch movies). To watch the most recent movie, we go to Theater. However, every time we watch a new movie, there are so many waiting lines. So, people tend to reserve a movie ticket to save their waiting time. When we usually reserve the movie tickets, we first search for the nearest movie theater. Second, we go to the movie theater website to reserve tickets. Third, we have to make sure whether the movie theater is screening the movie that we want. But some movies only screens in certain movie theaters. I think it is very inconvenient to visit every website to find the movie. Yes, Google has its own system that gathers by theater and time for each movie. I think it would be great if there is a website that handles movie ticket reservations from any theater. Therefore, this online movie reservation project is to handle the database for the website.

1.1. Scope and Purpose of Document

This document provides detailed information on the requirements, design, and implementation of the online movie reservation. It demonstrates the database design phase. It describes the user requirements such as software and hardware. In conceptual and logical design, it describes the database objects of ER model derived by mapping entities to table, attributing the columns and relationship between tables. It provides the functional usage of entities table using the DBMS schema.

1.2. Project Objective

The database design project explains the database design and database implementation using relational DBMS. It demonstrates the database design by showing the ER model/diagram. For ER model, it uses MySQL workbench to show how the entities are mapped to each other. It also introduces the description of the data and the type of the data by providing a data dictionary in the ER diagram. By designing the database, it can describe the functional usage of the table entities by implementing the database schema. So, it can describe the relational DBMS by designing and implementing a database.

2. System Requirements

For this project, I used MySQL Workbench to design the database. So, the system needs the hardware/software system that handles installation requirements.

2.1 Hardware Requirements

CPU: Intel Core
RAM: 4 GB
Minimum Space Required 500 MB

2.2 Software Requirements

Operating Environment Window 7, MySQL Workbench.

2.3 Functional Requirements

- Every customer must provide the first name, last name, email, and password to register. When customer registers, they must provide their personal information such as an address, birth date, and phone number.
- Every customer can find an available local theater near them.
- Customers can find the movie that they want to watch by searching for the genre, screen level, released date, etc.
- When the customer pays for the tickets, the customer must provide card type, card number, expiration date, CSV, and discount price.
- When the customer pays for the tickets, tickets show what movie that they are watching, price, seat number, and theater information.
- Theater information contains theater address, seat number, screen number, and name.

2.4 Database Requirements

For this database project, I used MySQL Server 8.0. I decided to use this database because it is free and has functionality to draw ERD.

3. Database Design Description

To design the database, I used MySQL Workbench. In ER Diagram, it has 14 entities tables. It shows detailed information of the entities, attributes, and relationships between entities.

3.1 Design Rationale

To design the ER diagram, I used the artificial primary key to identify each instance of the relation. By setting up the artificial primary key in sequential order, I was easily able to connect between entities. I choose to use non-identifying relationships for some entities instead of identifying relationships. I know identifying relationship helps us to maintain data integrity. But I use the non-identifying relationship to flexibly accommodate the changing requirements.

Since this database handles online movie reservation systems, it must handle the user. Every time the user register for the website, they must create an account and provide their information. The Customer table stores user-account information such as email, name, and password. The personal information table stores user-personal information such as an address, birth date, and phone number. For the first-time user, the customer's movie info is empty. Every time they purchase the movie ticket, it stores the user's watch count in the customer movie info table.

After the user register, they can search for the movie that they want to watch. The Movie entity table stores the basic information about the movie (name, running time, screen level, etc.) The screening level shows the age rating for each movie. The genre shows what kind of movie genre is. Each movie can have multiple genre types, and one genre type has multiple movies.

After the user chooses what to watch, they must make sure that the movie they choose is currently screening. There are a certain number of movies that theater is screening per day. The showtime entity table shows the current screening movie with start time and end time.

The user must choose where to watch the movie. The theater table explains the address of the theater. The Screen area table manages the theater information. So, it contains screen name, screen size, remains seat, etc. The seating table manages the seats in each theater. So, it shows what type of seat, and whether there is a seat.

The user must purchase the ticket. Since it is an online reservation system, it only accepts credit cards. To purchase the ticket, the user must provide payment information. After the payment, the system stores the payment information. After the payment, the user can get the ticket. The ticket table must contain the number of seats, total price, screen information, seat information, payment information, and movie information.

Using this database, we will be able to answer these questions: "What is the name of the theater that is near customer #1's city?", "What is the most famous purchased movie by the customer?" I think the database with these tables and relationships can handle the online movie reservation system.

3.2 E/R Model

The E/R Model section will introduce the entities tables, the relationship between the entities tables, and the diagram that shows every structure of the database. There are a total of 13 entity tables in my ER diagram. For tables, there are Customer, Personal information, Customer movie information, Payment, Payment detail, Movie, Screen level, movie genre, genre, showtime, theater, screen area, seat, and ticket. To design the ER diagram, I used MySQL Workbench.

3.2.1 Entities

- Customer
  - The purpose of the database design document is to build the database for an Online shopping mall. So, this entity contains the customer login information. It contains attributes of customer id, email address, first name, last name, and password.
- Personal Information
  - This entity contains the personal information of the customer such as a home address, phone number, and birth date. It contains attributes of personal id, street address, city, postal code, state, birth date, phone number, and customer id.
- Customer Movie Information
  - This entity contains the information that the customer has by watching the movie. By watching a movie, customers can earn points. It

contains attributes of points, member grade, watch count, and customer id.

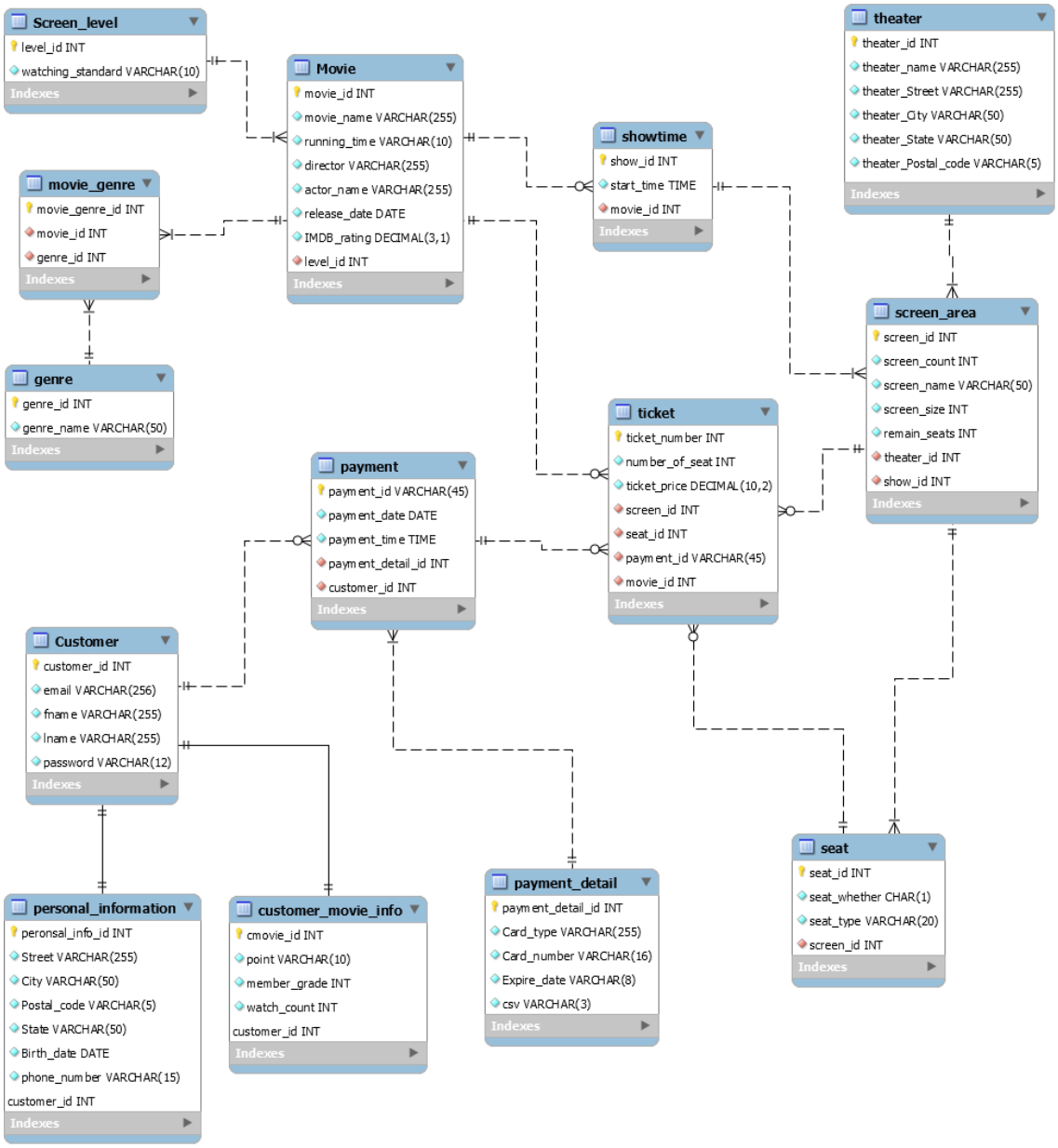
- Payment
  - This entity contains the information of payment. It contains attributes of payment date, payment time, payment detail code, and customer id.
- Payment detail
  - This entity contains major information of the payment such as card information. It contains attributes of card type, card number, expiration date, and CSV.
- Screen Level
  - This entity contains screen levels for movie and film ratings such as PG-13, R, etc. It contains attributes of level id and watching standard.
- Genre
  - This entity shows what kind of genre the movie is. It contains attributes of genre id and genre name.
- Movie
  - This entity provides major information about the movie. It contains attributes of movie id, movie name, running time, directors, actor names, release date, IMDB rating, and level id.
- Movie Genre
  - This entity is to connect between movie entity and genre entity. It contains attributes of movie genre id, movie id, and genre id.
- Show Time
  - This entity shows the information about what kind of movie is showing. It contains the attributes of show id, start time, and movie id.
- Theater
  - This entity contains information about the theater address. It contains attributes of theater id, name, street, city, state, and postal code.
- Screen area
  - This entity manages information about each cinema/theater. It contains attributes of screen id, screen count, name, size, remain seat, theater id, and show id.
- Seat
  - This entity manages the seat information for each theater. It contains the attributes of seat number, seat usage, type of seat, and screen id.
- Ticket
  - This entity contains all the information about the ticket. In Tickets, it contains every information of customer, movie, seat, theater, and billing. It contains the attributes of the ticket number, number of seats, total price, screen id, seat id, customer id, and movie id. The attributes of ticket whether shows whether customers have tickets or not.

3.2.2 Relationships

- Every Customer must have unique personal information and customer movie information (1: 1).

    - If there is no customer, there is no personal information and customer movie information.
- Every purchase must be made by the customer.
    - Payment occurs if and only if the customer tried to pay.
- For every purchase, customer must provide their billing information.
- Every Movie must have information for screen level, and genre type.
    - Each Movie can have multiple genre types. (M: N)
    - One screen level has multiple movies (1: M).
    - One genre type has multiple movies (1: M).
- Theater doesn't always show every movie.
    - It screens the movie, if and only if the movie exists.
- Each theater must have screen information (1: M).
    - Every screen in the theater must have seat information.
- Each theater has multiple screen area information (1: M).
- Each theater has multiple seats (1: M).
- Customers can get a ticket when they finish the payment.
- Ticket should contain number seats, total price, screen information, seat information, payment information, and movie information.

3.2.3 E/R Diagram



3.3 Relational Model

3.3.1 Data Dictionary

Table Name	Column Name	Description	Data Type	Size	Constraint Type	Not Null?	Valid Values

Customer	customer_id	Customer ID	INT	10	Primary Key	Y	10 numeric digits
Customer	email	Customer email	VARCHAR	256	None	Y	256 Character string
Customer	fname	Customer's First Name	VARCHAR	255	None	Y	255 Character string
Customer	lname	Customer's Last Name	VARCHAR	255	None	Y	255 Character string
Customer	password	Customer's password	VARCHAR	12	None	Y	12 numeric digits
Personal information	Personal_info	Personal Information ID	INT	10	Primary Key	Y	10 numeric digits
Personal information	Street	Street Address	VARCHAR	255	None	Y	255 Character string
Personal Information	City	City	VARCHAR	50	None	Y	50 Character string
Personal Information	Postal_ code	Zip code	VARCHAR	5	None	Y	5 numeric digits
Personal Information	State	State	VARCHAR	50	None	Y	50 Character string
Personal Information	Birth_date	Birth date	DATE	10	None	Y	Numerical digits in YYYY-MM-DD format
Personal Information	Phone_number	Phone number	VARCHAR	15	None	Y	15-character strings
Personal Information	Customer_id	Customer ID	INT	10	Primary Key	Y	10 numeric digits
Customer_ movie_info	Cmovie_id	Customer's movie information	INT	10	Primary Key	Y	10 numeric digits
Customer_ movie_info	point	Customer movie watch point	VARCHAR	10	None	Y	10 numeric digits
Customer_ movie_info	Member_grade	Customer's member grade	INT	7	None	Y	7 numeric digits
Customer_ movie_info	Watch_count	Customer watch count	INT	6	None	Y	6 numeric digits
Customer_ movie_info	Customer_id	Customer ID	INT	10	Primary Key	Y	10 numeric digits
Payment	payment_id	Payment ID	INT	10	Primary Key	Y	10 numeric digits
Payment	payment_date	Payment date	DATE	10	None	Y	Numerical digits in YYYY-MM-DD format
Payment	payment _time	Payment Time	TIME	8	None	Y	Numerical digits in HH:MM: SS format
Payment	Payment_detail_i d	Payment detail ID	INT	10	Foreign Key	Y	10 numeric digits
Payment	Customer_id	Customer ID	INT	10	Foreign Key	Y	10 numeric digits

Payment_detail	Payment_detail_id	Payment detail ID	INT	10	Primary Key	Y	10 numeric digits
Payment_detail	Card_type	Credit Card Type	VARCHAR	255	None	Y	255 Character string
Payment_detail	Card_number	Credit Card Number	VARCHAR	16	None	Y	16-character strings
Payement_detail	Expire_date	Credit Card expiration Date	VARCAR	10	None	Y	Numerical digits in YYYY-MM-DD format
Payement_detail	csv	CSV (Card Security Value)	VARCHAR	3	None	Y	3-character strings
Movie	Movie_id	Movie ID	INT	10	Primary Key	Y	10 numeric digits
Movie	Movie_name	Movie Name	VARCHAR	255	None	Y	255 Character string
Movie	Running_time	Running time minutes of the movie	VARCHAR	3	None	Y	3-character strings
Movie	director	Movie's Director	VARCHAR	255	None	Y	255 Character string
Movie	Actor_name	List of movie Actor names	VARCHAR	255	None	Y	255 Character string
Movie	Release_date	Movie released date	DATE	10	None	Y	Numerical digits in YYYY-MM-DD format
Movie	IMDB_rating	Movie IMDB rate	DECIMAL	3,1	None	Y	3 Numeric in 0.00 format
Movie	Level_id	Screen level ID	INT	10	Foreign Key	Y	10 numeric digits
Screen_level	Level_id	Screen Level ID	INT	10	Primary Key	Y	10 numeric digits
Screen_level	Watching_standard	Motion Picture Association film rating	VARCHAR	10	None	Y	10 Character string
Movie_genre	Movie_genre_id	Movie Genre ID	INT	10	Primary Key	Y	10 numeric digits
Movie_genre	Movie_id	Movie ID	INT	10	Foreign Key	Y	10 numeric digits
Movie_genre	Genre_id	Genre ID	INT	10	Foreign Key	Y	10 numeric digits
Genre	Genre ID	Genre ID	INT	10	Primary Key	Y	10 numeric digits
Genre	Genre_name	Movie Genre Type	VARCHAR	50	None	Y	50 Character string
showtime	Show_id	Showtime ID	INT	10	Primary Key	Y	10 numeric digits
showtime	Start_time	Movie start Time	TIME	8	None	Y	Numerical digits in HH:MM:SS format

showtime	Movie_id	Movie ID	INT	10	Foreign Key	Y	10 numeric digits
theater	Therater_id	Theater ID	INT	10	Primary Key	Y	10 numeric digits
theater	Theater_name	Name of the Theater	VARCHAR	255	None	Y	255 Character string
theater	Theater_street	Street Address of the theater	VARCHAR	255	None	Y	255 Character string
theater	Theater_city	City of the theater	VARCHAR	50	None	Y	50 Character string
theater	Theater_state	State of the theater	VARCHAR	50	None	Y	50 Character string
theater	theater_postal_ code	Zip code of the theater	VARCHAR	5	None	Y	5 numeric digits
Screen_area	Screen_id	Screen ID	INT	10	Primary Key	Y	10 numeric digits
Screen_area	Seat_count	Seat count in each screen	INT	3	None	Y	3 numeric digits
Screen_area	Screen_name	Name of the screen in the theater	VARCHAR	50	None	Y	50 Character string
Screen_area	Screen_size	Size of the screen in the theater	INT	3	None	Y	3 numeric digits
Screen_area	Remain_seats	Remain seat in the screen	INT	3	None	Y	3 numeric digits
Screen_area	Theater_id	Theater ID	INT	10	Foreign Key	Y	10 numeric digits
Screen_area	Show_id	Showtime ID	INT	10	Foreign Key	Y	10 numeric digits
Seat	Seat_id	Seat ID	INT	10	Primary Key	Y	10 numeric digits
Seat	Seat_whether	Check whether there is a seat	CHAR	1	None	Y	1 Character (Y or N)
Seat	Seat_type	Type of seat	VARCHAR	20	None	Y	20 Character string
Seat	Screen_id	Screen ID	INT	10	Foreign Key	Y	10 numeric digits
ticket	ticket_number	Ticket Number	INT	10	None	Y	10 numeric digits
ticket	Number_of_seat	Number of ticket that customer ordered	INT	10	None	Y	10 numeric digits
ticket	Ticket_price	Total ticket price	DECIMAL	10	None	Y	3 Numeric in 0.00 format
ticket	Screen_id	Screen ID	INT	10	Foreign Key	Y	10 numeric digits
ticket	Seat_id	Seeat ID	INT	10	Foreign Key	Y	10 numeric digits
ticket	Payment_id	Payment ID	INT	10	Foreign Key	Y	10 numeric digits
ticket	Movie_id	Movie ID	INT	10	Foreign Key	Y	10 numeric digits

3.3.2 Integrity Rules

In the integrity rules, there are 3 types of rules: Entity integrity, Referential integrity, and Domain integrity. To follow entity integrity, I set a unique not null primary key. For domain integrity, it needs to check whether the correct data is inserted. I set the table attributes as not null to handle the domain integrity. To handle referential integrity, I use a foreign key to connect between tables.

- Payment_detail table references to payment table.
- Screen_level table references Movie table.
- Theater table references screen_area table.
- Movie table references showtime table.
- Movie, screen_area, payment, and seat tables reference ticket table.
- Customer table references payment table.

3.3.3 Operational Rules

The database application will allow the user to enter the information if there is an existing user in the database. Even if there is an existing user in the database, the application knows the difference between users by the id. When the user enters the data, the database gives the user an id that is different from others. Unless the id isn't duplicated, the user can enter data that is the same as the existing user.

3.3.4 Operations

When the user register for the application, they can insert the information into the system (insert). When the user reserves for the movie, they can see their information (retrieve). They can also update their information if they need to change their personal information (update). Also, they can delete the information if they cancel the movie reservation (delete). So, when the user makes a registration/reservation, it involves operations of insert/delete/update/retrieve.

3.4 Security

If I have to build the web application using this database, I will block the remote access. I could block the 3306/tcp port that MySQL listens to as a default so that the database is used only by locally installed web applications. Also, I could change the database administrator password to secure the database.

3.5 Database Backup and Recovery

In MySQL Workbench, it has a method to backup the database called "MySQL Enterprise Backup." Using this method, I was able to export the dump database data and save my backup in my storage. MySQL Workbench also supports recovery options called "MySQL Enterprise Backup – Recovery." Using this method, I can import the back the database to recover my database.

3.6 Using Database Design or CASE Tool

For the database design tool, I used MySQL Workbench. It is free and easy to use. I like this database design tool because it has the functionality to draw ERD. To design the database, I draw the ERD. MySQL Workbench has features of "Forward Engineer" to create the database from ERD. Also, it has the feature of "Synchronized model" to update the database based on ERD changes.

3.7 Other Possible E/R Relationships

I think my database don't need any alternatives. However, If I have to consider other alternatives when I designed my database, then I would try to expand the database using normalization.

4. Implementation Description

To implement the query, I used MySQL workbench. Using MySQL workbench, I was able to implement the DDL and DML. I was able to view the data type using DESCRIBE.

4.1 Data Dictionary

```
mysql> DESCRIBE online_movie_reservation_model.customer;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| customer_id | int          | NO   | PRI | NULL    | auto_increment |
| email       | varchar(256) | NO   |     | NULL    |                |
| fname       | varchar(255) | NO   |     | NULL    |                |
| lname       | varchar(255) | NO   |     | NULL    |                |
| password    | varchar(12)  | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```

```
mysql> DESCRIBE online_movie_reservation_model.customer_movie_info;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| cmovie_id    | int         | NO   | PRI | NULL    | auto_increment |
| point        | varchar(10) | NO   |     | NULL    |                |
| member_grade | int         | NO   |     | NULL    |                |
| watch_count  | int         | NO   |     | NULL    |                |
| customer_id  | int         | NO   | PRI | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
```

```
mysql> DESCRIBE online_movie_reservation_model.genre;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| genre_id   | int         | NO   | PRI | NULL    | auto_increment |
| genre_name | varchar(50) | NO   |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
```

```
mysql> DESCRIBE online_movie_reservation_model.movie;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| movie_id     | int          | NO   | PRI | NULL    | auto_increment |
| movie_name   | varchar(255) | NO   |     | NULL    |                |
| running_time | varchar(3)   | NO   |     | NULL    |                |
| director     | varchar(255) | NO   |     | NULL    |                |
| actor_name   | varchar(255) | NO   |     | NULL    |                |
| release_date | date         | NO   |     | NULL    |                |
| IMDB_rating  | decimal(3,1) | NO   |     | NULL    |                |
| level_id     | int          | NO   | MUL | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
```

```
mysql> DESCRIBE online_movie_reservation_model.movie_genre;
+----------------+------+------+-----+---------+-------+
| Field          | Type | Null | Key | Default | Extra |
+----------------+------+------+-----+---------+-------+
| movie_genre_id | int  | NO   | PRI | NULL    |       |
| movie_id       | int  | NO   | MUL | NULL    |       |
| genre_id       | int  | NO   | MUL | NULL    |       |
+----------------+------+------+-----+---------+-------+
```

```
mysql> DESCRIBE online_movie_reservation_model.payment;
+-------------------+-------------+------+-----+---------+-------+
| Field             | Type        | Null | Key | Default | Extra |
+-------------------+-------------+------+-----+---------+-------+
| payment_id        | varchar(45) | NO   | PRI | NULL    |       |
| payment_date      | date        | NO   |     | NULL    |       |
| payment_time      | time        | NO   |     | NULL    |       |
| payment_detail_id | int         | NO   | MUL | NULL    |       |
| customer_id       | int         | NO   | MUL | NULL    |       |
+-------------------+-------------+------+-----+---------+-------+
```

```
mysql> DESCRIBE online_movie_reservation_model.payment_detail;
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| payment_detail_id | int          | NO   | PRI | NULL    | auto_increment |
| Card_type         | varchar(255) | NO   |     | NULL    |                |
| Card_number       | varchar(16)  | NO   |     | NULL    |                |
| Expire_date       | varchar(8)   | NO   |     | NULL    |                |
| csv               | varchar(3)   | NO   |     | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
```

```
mysql> DESCRIBE online_movie_reservation_model.personal_information;
+------------------+--------------+------+-----+---------+----------------+
| Field            | Type         | Null | Key | Default | Extra          |
+------------------+--------------+------+-----+---------+----------------+
| peronsal_info_id | int          | NO   | PRI | NULL    | auto_increment |
| Street           | varchar(255) | NO   |     | NULL    |                |
| City             | varchar(50)  | NO   |     | NULL    |                |
| Postal_code      | varchar(5)   | NO   |     | NULL    |                |
| State            | varchar(50)  | NO   |     | NULL    |                |
| Birth_date       | date         | NO   |     | NULL    |                |
| phone_number     | varchar(15)  | NO   |     | NULL    |                |
| customer_id      | int          | NO   | PRI | NULL    |                |
+------------------+--------------+------+-----+---------+----------------+
```

```
mysql> DESCRIBE online_movie_reservation_model.screen_area;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| screen_id    | int         | NO   | PRI | NULL    | auto_increment |
| seat_count   | int         | NO   |     | NULL    |                |
| screen_name  | varchar(50) | NO   |     | NULL    |                |
| screen_size  | int         | NO   |     | NULL    |                |
| remain_seats | int         | NO   |     | NULL    |                |
| theater_id   | int         | NO   | MUL | NULL    |                |
| show_id      | int         | NO   | MUL | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
```

```
mysql> DESCRIBE online_movie_reservation_model.screen_level;
+-------------------+-------------+------+-----+---------+----------------+
| Field             | Type        | Null | Key | Default | Extra          |
+-------------------+-------------+------+-----+---------+----------------+
| level_id          | int         | NO   | PRI | NULL    | auto_increment |
| watching_standard | varchar(10) | NO   |     | NULL    |                |
+-------------------+-------------+------+-----+---------+----------------+
```

```
mysql> DESCRIBE online_movie_reservation_model.seat;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| seat_id      | int         | NO   | PRI | NULL    | auto_increment |
| seat_whether | char(1)     | NO   |     | NULL    |                |
| seat_type    | varchar(20) | NO   |     | NULL    |                |
| screen_id    | int         | NO   | MUL | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
```

```
mysql> DESCRIBE online_movie_reservation_model.showtime;
+------------+------+------+-----+---------+----------------+
| Field      | Type | Null | Key | Default | Extra          |
+------------+------+------+-----+---------+----------------+
| show_id    | int  | NO   | PRI | NULL    | auto_increment |
| start_time | time | NO   |     | NULL    |                |
| movie_id   | int  | NO   | MUL | NULL    |                |
+------------+------+------+-----+---------+----------------+
```

```
mysql> DESCRIBE online_movie_reservation_model.theater;
+---------------------+--------------+------+-----+---------+----------------+
| Field               | Type         | Null | Key | Default | Extra          |
+---------------------+--------------+------+-----+---------+----------------+
| theater_id          | int          | NO   | PRI | NULL    | auto_increment |
| theater_name        | varchar(255) | NO   |     | NULL    |                |
| theater_Street      | varchar(255) | NO   |     | NULL    |                |
| theater_City        | varchar(50)  | NO   |     | NULL    |                |
| theater_State       | varchar(50)  | NO   |     | NULL    |                |
| theater_Postal_code | varchar(5)   | NO   |     | NULL    |                |
+---------------------+--------------+------+-----+---------+----------------+
```

```
mysql> DESCRIBE online_movie_reservation_model.ticket;
+----------------+---------------+------+-----+---------+----------------+
| Field          | Type          | Null | Key | Default | Extra          |
+----------------+---------------+------+-----+---------+----------------+
| ticket_number  | int           | NO   | PRI | NULL    | auto_increment |
| number_of_seat | int           | NO   |     | NULL    |                |
| ticket_price   | decimal(10,2) | NO   |     | NULL    |                |
| screen_id      | int           | NO   | MUL | NULL    |                |
| seat_id        | int           | NO   | MUL | NULL    |                |
| payment_id     | varchar(45)   | NO   | MUL | NULL    |                |
| movie_id       | int           | NO   | MUL | NULL    |                |
+----------------+---------------+------+-----+---------+----------------+
```

4.2 Advanced Features

For my database project, I tried to use the trigger to test simple checks. So, I decided to check for the number of seats that one person can reserve at once. For my database project, each person shouldn't reserve more than 10. So, I made the trigger to check whether a person reserve more than 10 tickets.

This is the trigger that checks the maximum number of seats that one person can reserve.

```
DELIMITER $$
CREATE TRIGGER check_ticket_seat_number
BEFORE INSERT ON online_movie_reservation_model.ticket
FOR EACH ROW BEGIN
    IF (NEW.number_of_seat > 10) THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning: Number of seat should be less than 10.
    END IF;
END$$
DELIMITER ;

-- DROP TRIGGER check_ticket_seat_number;
```

To test this trigger, I tried to insert the ticket data with 11 tickets.

```
INSERT INTO `online_movie_reservation_model`.`ticket`
(`ticket_number`, `number_of_seat`, `ticket_price`, `screen_id`, `seat_id`, `payment_id`, `movie_id`)
VALUES ('51', '11', '85.00', '4', '9', '50', '3');
```

When I insert the data into the ticket table, the trigger shows a message as "Warning: Number of seats should be less than 10".

```
42  16:14:13  INSERT INTO `online_movie_reservation_model`.`ticket` (`ticket_numb...  Error Code: 1643. Warning: Number of seat should be less than 10.  0.000 sec
```

For the movie reservation database, I think the most common query would be the ticket information for the customer. So, I created the stored procedures to save time to search for the customer's ticket information by customer id.

```
DELIMITER $$
CREATE PROCEDURE GetCustomerTicketInformations(
                IN customerid INT)
BEGIN
    SELECT customer.fname, customer.lname, ticket.ticket_price, movie.movie_name
    FROM online_movie_reservation_model.customer,
         online_movie_reservation_model.payment,
         online_movie_reservation_model.ticket,
         online_movie_reservation_model.movie
    WHERE customer.customer_id = payment.customer_id AND
          payment.payment_id = ticket.payment_id AND
          movie.movie_id = ticket.movie_id AND
          customer.customer_id = customerid;
END$$
DELIMITER ;
```

Using this stored procedure, I can search for customer ticket information by customer id.

```
1 •   CALL GetCustomerTicketInformations(5);
```

Result Grid | Filter Rows: | Export:

fname	lname	ticket_price	movie_name
Ikey	Babonau	7.50	Cry Macho

For the advanced features of the database, I used stored procedures and triggers. Using stored procedures, I was able to call queries to avoid duplicate logic in the query. I think stored procedures can enforce business rules by reducing the amount of traffic between the database and application. Using triggers, I was able to set rules for the database whenever I insert the information into the system. I think triggers can enforce business rules by saving time to insert/update the system.

4.3 Queries

Using these queries, it shows how the database retrieve the information.

4.3.1 Show Ticket information for Customer #1

```
SELECT customer.fname, customer.lname, ticket.ticket_price, Movie.movie_name
FROM online_movie_reservation_model.ticket, online_movie_reservation_model.payment,
     online_movie_reservation_model.customer, online_movie_reservation_model.movie
WHERE ticket.payment_id = payment.payment_id AND payment.customer_id = customer.customer_id AND
      customer.customer_id =1 AND movie.movie_id = ticket.movie_id
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fname	lname	ticket_price	movie_name
Shana	Hassur	7.50	The Addams Family 2

4.3.2 List the customer who lives in San Jose

```
SELECT customer.fname, customer.lname
FROM online_movie_reservation_model.customer, online_movie_reservation_model.personal_information
WHERE customer.customer_id = personal_information.customer_id AND
      personal_information.City = 'San Jose'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fname	lname
Shana	Hassur
Brewster	Glassard
Far	Timblett
Kennett	Sneezem
Ikey	Babonau

4.3.3 Payment Information of customer's first name Hamilton

```
SELECT customer.fname, customer.lname, payment_detail.Card_type,
       payment_detail.Card_number, payment_detail.Expire_date, payment_detail.csv
FROM online_movie_reservation_model.payment_detail,
     online_movie_reservation_model.payment,
     online_movie_reservation_model.customer
WHERE payment_detail.payment_detail_id = payment.payment_detail_id AND
      payment.customer_id = customer.customer_id AND
      customer.fname ='Hamilton'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fname	lname	Card_type	Card_number	Expire_date	csv
Hamilton	Pavic	DISCOVER	6011293988295310	8/2021	111

4.3.4 List all movie by genre type

```
SELECT movie.movie_name, genre.genre_name
FROM online_movie_reservation_model.movie, online_movie_reservation_model.movie_genre,
     online_movie_reservation_model.genre
WHERE movie.movie_id = movie_genre.movie_id AND movie_genre.genre_id = genre.genre_id
ORDER BY genre.genre_id
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

movie_name	genre_name
Copshop	Action
Mission Impossible: Fallout	Action
Deadpool 2	Action
Avengers: Infinity War	Action
F9	Action
No Time To Die	Action
Venom: Let There Be Carnage	Action
Casino Royale	Action
Skyfall	Action
Shang-Chi and the Legend of the Ten Rings	Animation
Jungle Cruise	Animation

4.3.5 List movie by watching standard level

```
SELECT movie.movie_name, screen_level.watching_standard
FROM online_movie_reservation_model.movie, online_movie_reservation_model.screen_level
WHERE movie.level_id = screen_level.level_id
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

movie_name	watching_standard
Paw Patrol: The Movie	G
Show Me The Father	PG-13
The Incredibles 2	PG-13
The Addams Family 2	PG-13
Shang-Chi and the Legend of the Ten Rings	R
Dear Evan Hansen	R
Cry Macho	R
Free Guy	R
Jungle Cruise	R
The Eyes of Tammy Faye	R

4.3.6 List movies that are screening in the morning

```
SELECT movie.movie_name, showtime.start_time as earliest_movie
FROM  online_movie_reservation_model.showtime,
      online_movie_reservation_model.movie
WHERE movie.movie_id = showtime.movie_id AND showtime.start_time < '12:00'
ORDER BY showtime.start_time ASC
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

movie_name	earliest_movie
Dear Evan Hansen	09:20:00
The Addams Family 2	09:30:00
Shang-Chi and the Legend of the Ten Rings	10:00:00
Cry Macho	10:00:00
Malignant	10:30:00
The Eyes of Tammy Faye	11:00:00
The Addams Family 2	11:20:00
No Time To Die	11:30:00
Venom: Let There Be Carnage	11:30:00
Dear Evan Hansen	11:55:00

4.3.7 List of movies that customers bought the most.

```
SELECT movie.movie_name, COUNT(ticket.movie_id)
FROM online_movie_reservation_model.ticket, online_movie_reservation_model.movie
WHERE ticket.movie_id = movie.movie_id
GROUP BY ticket.movie_id
HAVING COUNT(ticket.movie_id) = (SELECT MAX(maxcount)
                                 FROM (SELECT COUNT(ticket.movie_id) as maxcount
                                       FROM online_movie_reservation_model.ticket,
                                            online_movie_reservation_model.movie
                                       WHERE ticket.movie_id = movie.movie_id
                                       GROUP BY ticket.movie_id) as results)
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

movie_name	COUNT(ticket.movie_id)
Cry Macho	5
Malignant	5

4.3.8 List of the movie theaters in the city with the most customers.

```
SELECT theater.theater_name
FROM  online_movie_reservation_model.theater
WHERE theater.theater_City = (SELECT personal_information.city as max_city
                              FROM online_movie_reservation_model.personal_information
                              GROUP BY personal_information.city
                              HAVING COUNT(personal_information.city) = (SELECT MAX(max_count_city)
                                                                         FROM (SELECT COUNT(personal_information.city) as max_count_city
                                                                               FROM online_movie_reservation_model.personal_information,
                                                                                    online_movie_reservation_model.customer
                                                                               WHERE customer.customer_id = personal_information.customer_id
                                                                               GROUP BY personal_information.city) as results))
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

theater_name
AMC Saratoga 14
AMC Eastridge 15
CineLux Almaden Cafe & Lounge
Towne 3 Cinemas
Cin?Arts Santana Row
West Wind Capitol Drive-In
3Below Theaters
Starlight Cinemas
IMAX Dome Theater
Century 20 Oakridge and XD

5. CRUD Matrix

Function / Entity Interaction	E1	E2	E3	E4	E5	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14
F1	CR UD														
F2	RU	CR UD													
F3	R		CR UD												
F4	R			CR UD											
F5				RU	CR UD										
F6						CR UD	CR UD	CR UD	CR UD	CR UD					
F7										R	CR UD				
F8												CR UD			
F9											R	R	CR UD		
F10													R	CR UD	
F11				R						R			R	R	CR UD
F12	R			R					R						R

F13	R	R													
F14	R			R	R										
F15								R	R	R					
F16										R	R				
F17										R					R
F18	R	R										R			

5.1 List of Entity Types

E1: Customer
E2: Personal information
E3: customer movie info
E4: payment
E5: payment detail
E6: screen level
E7: genre
E8: movie genre
E9: Movie
E10: showtime
E11: theater
E12: screen area
E13: seat
E14: ticket

5.2 List of Functions

F1: Insert/update/delete/retrieve a customer
F2: Insert/update/delete/retrieve personal information
F3: Insert/update/delete/retrieve customer movie information
F4: Assign payment information (Insert/update/delete/retrieve)
F5: Insert/update/delete/retrieve detail payment information
F6: Insert/update/delete/retrieve Movie information
F7: Insert/update/delete/retrieve showtime information
F8: Insert/update/delete/retrieve theater information
F9: Insert/update/delete/retrieve screen area information
F10: Insert/update/delete/retrieve seat information
F11: Assign ticket information (Insert/update/delete/retrieve)
--Query—
F12: Seach for ticket information for certain customer
F13: List the customer who lives in certain cities
F14: payment detail information for certain customer
F15: List the movies by genre type

F16: List the movie that is screening in certain time
F17: Famous movie that customer bought most
F18: Movie theaters where is in most customer's city.

6. Concluding Remarks

By writing a database design report, I was able to build the database with ERD. I learned how to draw entity-relational models using MySQL Workbench. I also learned how to handle the relationship between tables using the primary key, foreign key, and constraints. I also learned how to implement queries for the database. By writing a design report, I think my strength is implementing queries. When I was implementing the query, I was able to implement a complex query by using the divide-and-conquer method. I think my weakness would be designing ERD using MySQL Workbench. I was able to handle most of the ERD design. However, I think I still need to learn more about the detailed part of the ERD such as handling constraints in the database system. If I have more time for the database design, I will try to implement/design the advanced features. If I use advanced features such as triggers and stored procedures, then I would be able to enforce my database system.

Appendices

Appendix A - DDL, INSERT, SELECT Statements

To create the database, I used MySQL Workbench function to change from ERD to database. So, I first create the ERD, and then create the ERD using forward engineering.

```sql
-- -----------------------------------------------------
-- Schema online_movie_reservation_model
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `online_movie_reservation_model` DEFAULT CHARACTER SET utf8 ;
USE `online_movie_reservation_model` ;

-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`Screen_level`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`Screen_level` (
  `level_id` INT NOT NULL AUTO_INCREMENT,
  `watching_standard` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`level_id`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`Movie`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`Movie` (
  `movie_id` INT NOT NULL AUTO_INCREMENT,
  `movie_name` VARCHAR(255) NOT NULL,
  `running_time` VARCHAR(3) NOT NULL,
  `director` VARCHAR(255) NOT NULL,
  `actor_name` VARCHAR(255) NOT NULL,
  `release_date` DATE NOT NULL,
  `IMDB_rating` DECIMAL(3,1) NOT NULL,
  `level_id` INT NOT NULL,
  PRIMARY KEY (`movie_id`),
  INDEX `fk_Movie_Screen_level_idx` (`level_id` ASC) VISIBLE,
  CONSTRAINT `fk_Movie_Screen_level`
    FOREIGN KEY (`level_id`)
    REFERENCES `online_movie_reservation_model`.`Screen_level` (`level_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`genre`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`genre` (
  `genre_id` INT NOT NULL AUTO_INCREMENT,
  `genre_name` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`genre_id`))
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`showtime`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`showtime` (
  `show_id` INT NOT NULL AUTO_INCREMENT,
  `start_time` TIME NOT NULL,
  `movie_id` INT NOT NULL,
  PRIMARY KEY (`show_id`),
  INDEX `fk_showtime_Movie1_idx` (`movie_id` ASC) VISIBLE,
  CONSTRAINT `fk_showtime_Movie1`
    FOREIGN KEY (`movie_id`)
    REFERENCES `online_movie_reservation_model`.`Movie` (`movie_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`theater`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`theater` (
  `theater_id` INT NOT NULL AUTO_INCREMENT,
  `theater_name` VARCHAR(255) NOT NULL,
  `theater_Street` VARCHAR(255) NOT NULL,
  `theater_City` VARCHAR(50) NOT NULL,
  `theater_State` VARCHAR(50) NOT NULL,
  `theater_Postal_code` VARCHAR(5) NOT NULL,
  PRIMARY KEY (`theater_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`screen_area`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`screen_area` (
  `screen_id` INT NOT NULL AUTO_INCREMENT,
  `seat_count` INT NOT NULL,
  `screen_name` VARCHAR(50) NOT NULL,
  `screen_size` INT NOT NULL,
  `remain_seats` INT NOT NULL,
  `theater_id` INT NOT NULL,
  `show_id` INT NOT NULL,
  PRIMARY KEY (`screen_id`),
  INDEX `fk_screen_area_cinema1_idx` (`theater_id` ASC) VISIBLE,
  INDEX `fk_screen_area_showtime1_idx` (`show_id` ASC) VISIBLE,
  CONSTRAINT `fk_screen_area_cinema1`
    FOREIGN KEY (`theater_id`)
    REFERENCES `online_movie_reservation_model`.`theater` (`theater_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_screen_area_showtime1`
    FOREIGN KEY (`show_id`)
    REFERENCES `online_movie_reservation_model`.`showtime` (`show_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`Customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`Customer` (
  `customer_id` INT NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(256) NOT NULL,
  `fname` VARCHAR(255) NOT NULL,
  `lname` VARCHAR(255) NOT NULL,
  `password` VARCHAR(12) NOT NULL,
  PRIMARY KEY (`customer_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`customer_movie_info`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`customer_movie_info` (
  `cmovie_id` INT NOT NULL AUTO_INCREMENT,
  `point` VARCHAR(10) NOT NULL,
  `member_grade` INT NOT NULL,
  `watch_count` INT NOT NULL,
  `customer_id` INT NOT NULL,
  PRIMARY KEY (`cmovie_id`, `customer_id`),
  INDEX `fk_customer_movie_info_Customer1_idx` (`customer_id` ASC) VISIBLE,
  CONSTRAINT `fk_customer_movie_info_Customer1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `online_movie_reservation_model`.`Customer` (`customer_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`personal_information`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`personal_information` (
  `peronsal_info_id` INT NOT NULL AUTO_INCREMENT,
  `Street` VARCHAR(255) NOT NULL,
  `City` VARCHAR(50) NOT NULL,
  `Postal_code` VARCHAR(5) NOT NULL,
  `State` VARCHAR(50) NOT NULL,
  `Birth_date` DATE NOT NULL,
  `phone_number` VARCHAR(15) NOT NULL,
  `customer_id` INT NOT NULL,
  PRIMARY KEY (`peronsal_info_id`, `customer_id`),
  INDEX `fk_personal_information_Customer1_idx` (`customer_id` ASC) VISIBLE,
  CONSTRAINT `fk_personal_information_Customer1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `online_movie_reservation_model`.`Customer` (`customer_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`payment_detail`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`payment_detail` (
  `payment_detail_id` INT NOT NULL AUTO_INCREMENT,
  `Card_type` VARCHAR(255) NOT NULL,
  `Card_number` VARCHAR(16) NOT NULL,
  `Expire_date` VARCHAR(8) NOT NULL,
  `csv` VARCHAR(3) NOT NULL,
  PRIMARY KEY (`payment_detail_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`payment`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`payment` (
  `payment_id` VARCHAR(45) NOT NULL,
  `payment_date` DATE NOT NULL,
  `payment_time` TIME NOT NULL,
  `payment_detail_id` INT NOT NULL,
  `customer_id` INT NOT NULL,
  PRIMARY KEY (`payment_id`),
  INDEX `fk_payment_payment_detail1_idx` (`payment_detail_id` ASC) VISIBLE,
  INDEX `fk_payment_Customer1_idx` (`customer_id` ASC) VISIBLE,
  CONSTRAINT `fk_payment_payment_detail1`
    FOREIGN KEY (`payment_detail_id`)
    REFERENCES `online_movie_reservation_model`.`payment_detail` (`payment_detail_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_payment_Customer1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `online_movie_reservation_model`.`Customer` (`customer_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`seat`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`seat` (
  `seat_id` INT NOT NULL AUTO_INCREMENT,
  `seat_whether` CHAR(1) NOT NULL,
  `seat_type` VARCHAR(20) NOT NULL,
  `screen_id` INT NOT NULL,
  PRIMARY KEY (`seat_id`),
  INDEX `fk_seat_screen_area1_idx` (`screen_id` ASC) VISIBLE,
  CONSTRAINT `fk_seat_screen_area1`
    FOREIGN KEY (`screen_id`)
    REFERENCES `online_movie_reservation_model`.`screen_area` (`screen_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`ticket`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`ticket` (
  `ticket_number` INT NOT NULL AUTO_INCREMENT,
  `number_of_seat` INT NOT NULL,
  `ticket_price` DECIMAL(10,2) NOT NULL,
  `screen_id` INT NOT NULL,
  `seat_id` INT NOT NULL,
  `payment_id` VARCHAR(45) GENERATED ALWAYS AS () VIRTUAL,
  `movie_id` INT NOT NULL,
  PRIMARY KEY (`ticket_number`),
  INDEX `fk_ticket_screen_area1_idx` (`screen_id` ASC) VISIBLE,
  INDEX `fk_ticket_seat1_idx` (`seat_id` ASC) VISIBLE,
  INDEX `fk_ticket_payment1_idx` (`payment_id` ASC) VISIBLE,
  INDEX `fk_ticket_Movie1_idx` (`movie_id` ASC) VISIBLE,
  CONSTRAINT `fk_ticket_screen_area1`
    FOREIGN KEY (`screen_id`)
    REFERENCES `online_movie_reservation_model`.`screen_area` (`screen_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_ticket_seat1`
    FOREIGN KEY (`seat_id`)
    REFERENCES `online_movie_reservation_model`.`seat` (`seat_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_ticket_payment1`
    FOREIGN KEY (`payment_id`)
    REFERENCES `online_movie_reservation_model`.`payment` (`payment_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_ticket_Movie1`
    FOREIGN KEY (`movie_id`)
    REFERENCES `online_movie_reservation_model`.`Movie` (`movie_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `online_movie_reservation_model`.`movie_genre`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `online_movie_reservation_model`.`movie_genre` (
  `movie_genre_id` INT NOT NULL,
  `movie_id` INT NOT NULL,
  `genre_id` INT NOT NULL,
  PRIMARY KEY (`movie_genre_id`),
  INDEX `fk_movie_genre_Movie1_idx` (`movie_id` ASC) VISIBLE,
  INDEX `fk_movie_genre_genre1_idx` (`genre_id` ASC) VISIBLE,
  CONSTRAINT `fk_movie_genre_Movie1`
    FOREIGN KEY (`movie_id`)
    REFERENCES `online_movie_reservation_model`.`Movie` (`movie_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_movie_genre_genre1`
    FOREIGN KEY (`genre_id`)
    REFERENCES `online_movie_reservation_model`.`genre` (`genre_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

To populate the data into the database, I didn't use INSERT statements. Instead, I used one of MySQL Workbench function, "Table Data Export Wizard". Using this function, I was able to populate csv data into the database.

```
SELECT customer.fname, customer.lname, personal_information.City
FROM online_movie_reservation_model.customer, online_movie_reservation_model.personal_information
WHERE customer.customer_id = personal_information.customer_id
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fname	lname	City
Shana	Hassur	San Jose
Brewster	Glassard	San Jose
Far	Timblett	San Jose
Kennett	Sneezem	San Jose
Ikey	Babonau	San Jose
Jeanine	Fidele	Fremont
Hendrick	Vinau	Fremont
Lynda	Blasgen	Berkeley
Harmonie	MacKenny	Berkeley
Nevil	Smooth	Oakland
Jessalyn	Fargie	Oakland
Floris	Greneham	Oakland
Olia	McGuane	Hayward

Result Grid | Form Editor | Field Types | Query Stats

Appendix B - Data Dictionary Index

Column Name	Table Name
actor_name	movie
Birth_date	personal_information
Card_number	payment_detail
Card_type	payment_detail
City	personal_information
cmovie_id	customer_movie_info
csv	payment_detail
customer_id	customer_movie_info
customer_id	personal_information
customer_id	payment
customer_id	customer
director	movie
email	customer
Expire_date	payment_detail
fname	customer
genre_id	genre
genre_id	movie_genre
genre_name	genre
IMDB_rating	movie

level_id	movie
level_id	screen_level
lname	customer
member_grade	customer_movie_info
movie_genre_id	movie_genre
movie_id	movie_genre
movie_id	movie
movie_id	showtime
movie_id	ticket
movie_name	movie
number_of_seat	ticket
password	customer
payment_date	payment
payment_detail_id	payment_detail
payment_detail_id	payment
payment_id	ticket
payment_id	payment
payment_time	payment
peronsal_info_id	personal_information
phone_number	personal_information
point	customer_movie_info
Postal_code	personal_information
release_date	movie
remain_seats	screen_area
running_time	movie
screen_id	screen_area
screen_id	ticket
screen_id	seat
screen_name	screen_area
screen_size	screen_area
seat_count	screen_area
seat_id	ticket
seat_id	seat
seat_type	seat
seat_whether	seat
show_id	screen_area
show_id	showtime
start_time	showtime
State	personal_information
Street	personal_information
theater_City	theater
theater_id	theater
theater_id	screen_area

theater_name	theater
theater_Postal_code	theater
theater_State	theater
theater_Street	theater
ticket_number	ticket
ticket_price	ticket
watch_count	customer_movie_info
watching_standard	screen_level

References

"MySQL Workbench Manual :: 6.5.1 Table Data Export and Import Wizard." *MySQL*, https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-table.html.

Soam, Tushar. "Create Er Diagram of a Database in Mysql Workbench." *Medium*, Medium, 3 Feb. 2018, https://medium.com/@tushar0618/how-to-create-er-diagram-of-a-database-in-mysql-workbench-209fbf63fd03.

"Random Data Generator and API Mocking Tool: JSON / CSV / SQL / Excel." *Mockaroo*, https://www.mockaroo.com/.