**MOVIE RESERVATION SYSTEM.**

**INTRODUCTION:**

The aim of this model is to create a movie reservation system which has features for two types of users: admin and the customer. The customer would be able to book tickets for a movie in a step-by-step manner. For the admin, features like being able to create slots for new movies, update price, delete movies, shows and tickets (CRUD operations). The model would interact with a database and query on the same using a high-level language (python). The main aim is to cover topics like; data collection, relational model usage, SQL querying and python script to make the interface for the users. At the end, a report consisting of statistics related to the theatre database for the admin using plots and table as data analytics are shown.

**System description:**

Topic name: Movie Theatre Reservation System.

Objective: To design and implement a system for reserving tickets to a movie theatre that would help users for movie researching/booking and admin for handling the theatre’s specifications.

System requirements:

User accessibility for the following tasks:

1. Searching movies.
2. Viewing details.
3. Receiving Tickets and booking.

Admin accessibility for the following tasks:

1. Insert data.
2. Update data.
3. Delete data.
4. Access statistics of their theatre.
5. Update web pages.

**System functioning:**

Numerous tasks can be performed by the user that includes searching for a movie, viewing movie show details and schedules, a movie show, card registration and receiving tickets. Admins can use the system to insert, update and delete data such as movie descriptions and movie schedules, which will update the related web pages and will be accessible by the customers. Admins can also access statistics about the movie theatre, such as what are the most popular movies and the monthly revenue.

**Assumption:** We have assumed that ‘card registration’ means that the user’s payment card details are being stored separately.

**DATABASE DESIGN:**

**Data Source:**

The database has been collected from an open-source GitHub data set (JSON type) for movie reservation system. The data set will be pre-processed and then be used to develop the project as per the aforementioned system requirements.

GitHub link: <https://github.com/pritisharora55/ticket_reservation_mgmt>

**Data collection:**

Collected data from github, the data is related to movie reservation system and consist of 16 tables. In MySQL, Data Definition Language was used to create a database called “movie\_reservation” system and insert the 16 tables.

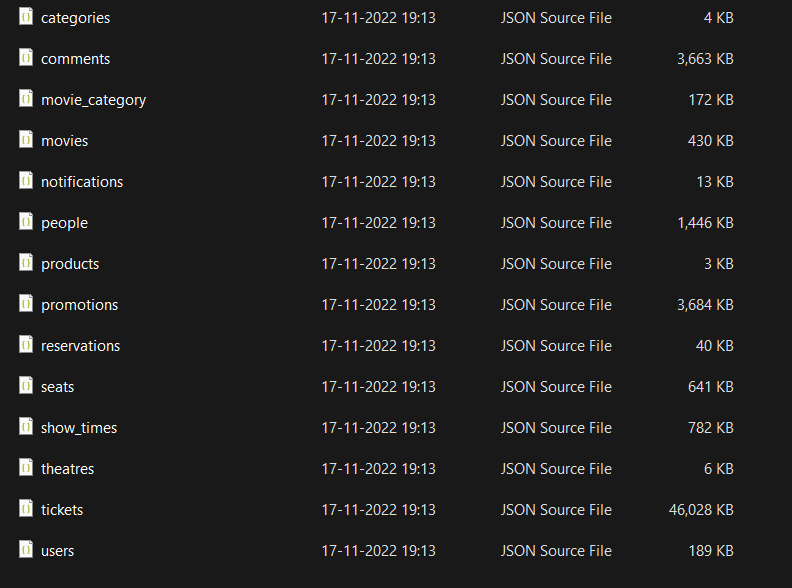
**Data pre-processing:**

1. Removal of duplicate values from table like theatre room.
2. Removal of null primary key values from “Movie\_Actor” and “Movie\_Director” tables.
3. Insertion of rows in tables where data is less than 10 rows.
4. Generation of a new column “card number” in the payments table.
5. Addition of random card number values in the newly created column of payments table.
6. For the tables, “Movie” and “movie\_genre”, “movie\_genre” had a greater number of movies as compared to “Movie” table, to solve this we joined the tables based on “movie\_id” then selected the common movies present in both the tables to insert. Same was done in ticket and seats table.

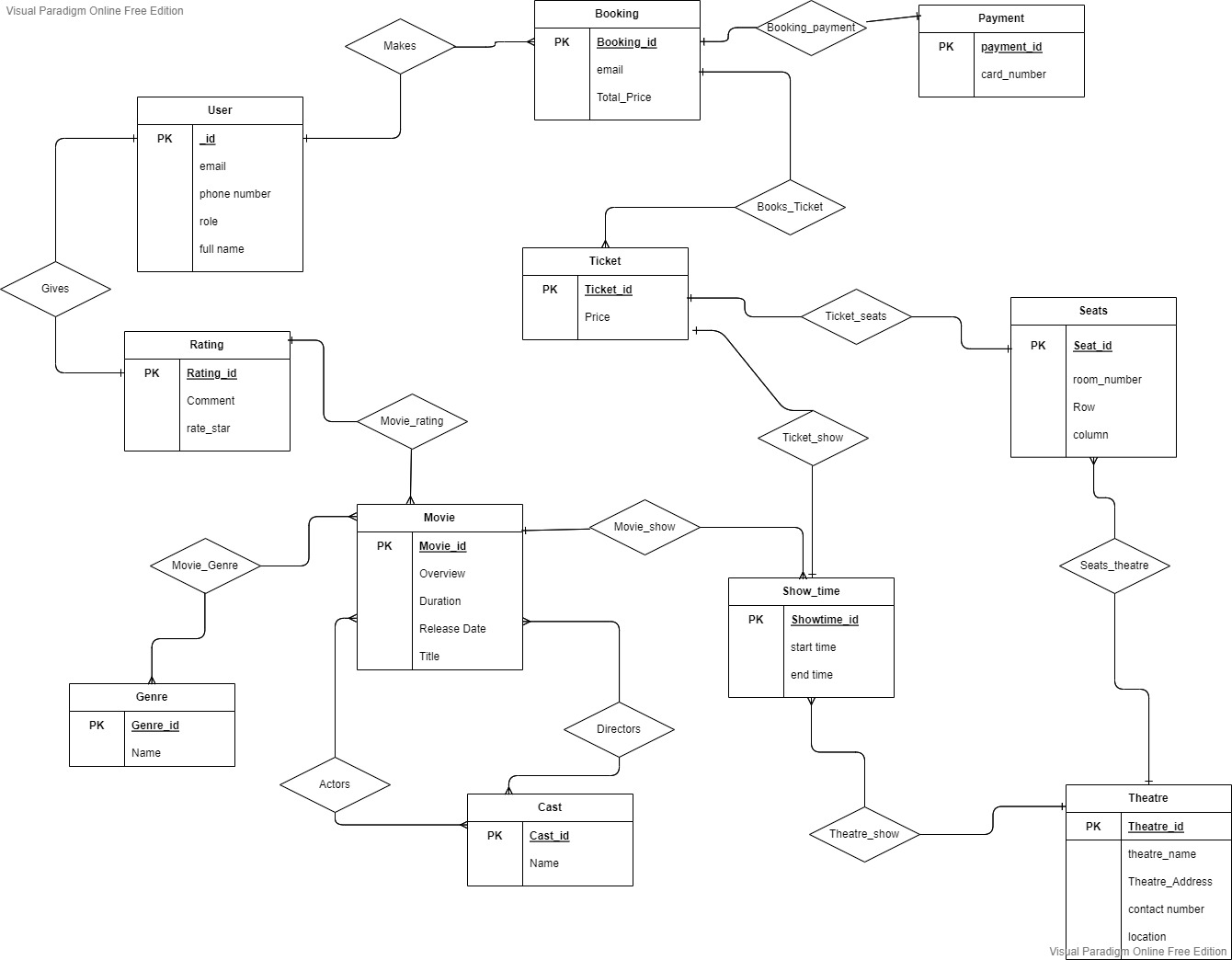
Parsing data through python into SQL tables.

Python was used to create connections with MongoDB and parse the data through MongoDB into SQL’s “movie\_reservation” schema.

The following JSON files were taken:

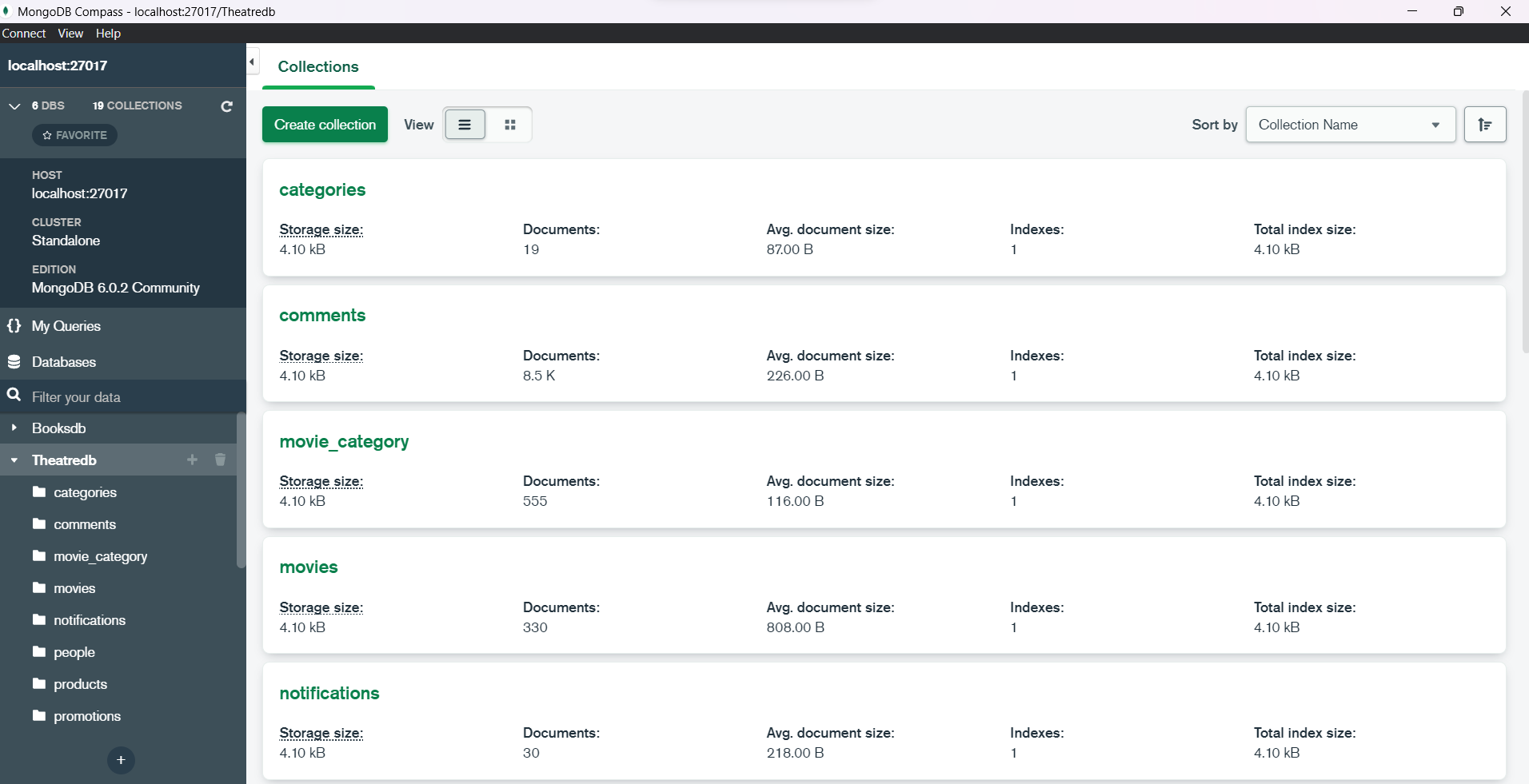


**Entity relation Diagram:**

****

**MongoDB:**

Database “Theatredb” was created with all the data set tables as independent collections. The files were in JSON format which were then uploaded to MongoDB in order to convert them to relational data using python and parsing the data into MySQL.

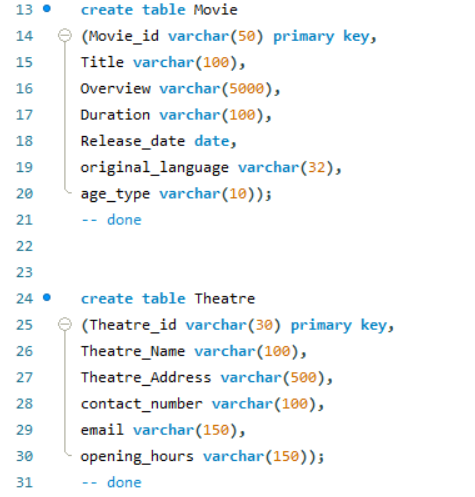


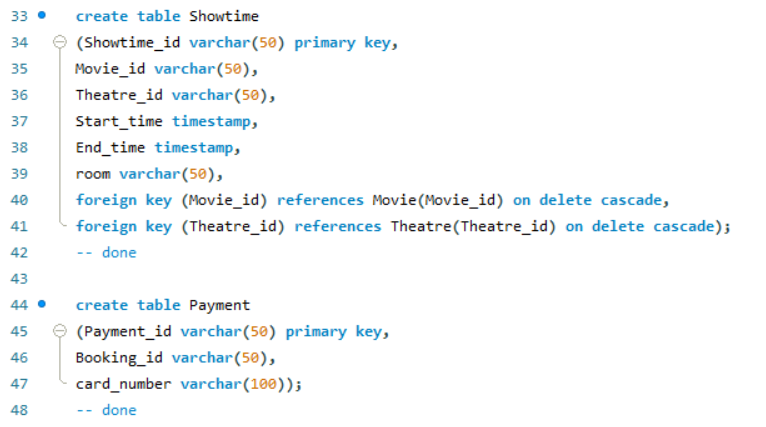
The data collected in MongoDB is connected to python using mysqlconnector and the engines are created to parse the data into SQL workbench.

**Data Definition Language (DDL):**

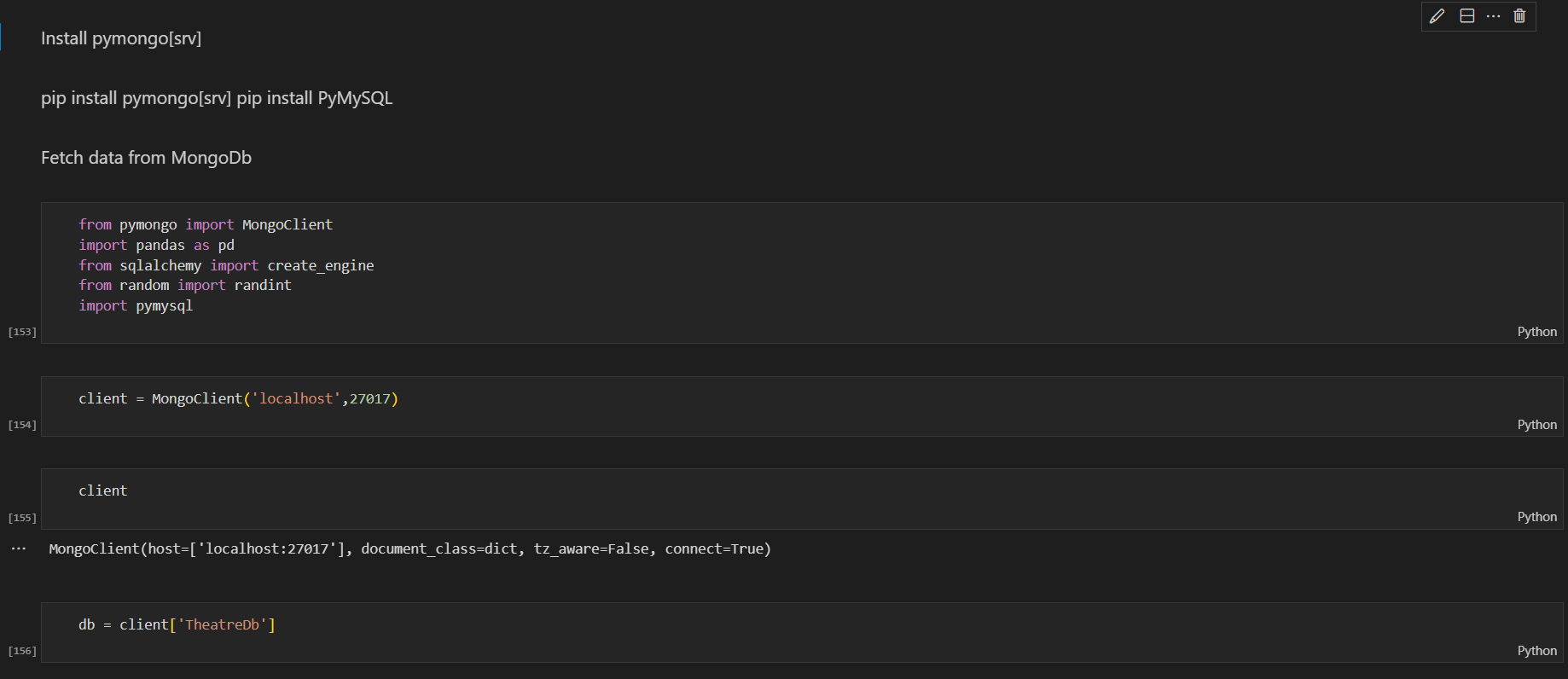
DDL queries were written to convert each collection in MongoDB to create individual tables, this is the step for conversion of JSON files (unstructured data) to relational data. A data loader python script was used to connect to MongoDB as well as MySQL Workbench, and through this script, each collection in MongoDB was mapped to their respective tables in workbench along with uploading data in the tables.

**Note:** SQL file for DDL has been uploaded separately. A sample is shown below.





**Data Loader Script:**









**Note:** PDF of data loader has been uploaded separately.

**RELATIONAL SCHEMA:**

|  |  |
| --- | --- |
| TABLE | FIELDS |
| Booking | Booking\_id, user\_id, Showtime\_id, total\_price, Payment\_id |
| Genre | Genre\_id, Name |
| Movie | Movie\_id, Title, Overview, Duration, Release\_date, original\_language, age\_type |
| Movie\_actor | Actor\_id, Movie\_id |
| Movie\_cast | Cast\_id, full\_name |
| Movie\_director | Director\_id, Movie\_id |
| Movie\_genre | Movie\_id, Genre\_id |
| Payment | Payment\_id, Booking\_id, card\_number |
| Rating | Rating\_id, user\_id, Movie\_id, comment, rate\_star |
| Seats | Seat\_id, Theatre\_id, room, row\_, column\_, seats\_ |
| Showtime | Showtime\_id, Movie\_id, Theatre\_id, Start\_time, End\_time, room |
| Theatre | Theatre\_id, Theatre\_Name, Theatre\_Address, contact\_number, email, opening\_hours |
| Theatre\_room | Theatre\_id, room\_number, room\_type |
| Ticket | Ticket\_id, Seat\_id, Showtime\_id, Booking\_id, Price |
| User | user\_id, full\_name, email, phone\_number, address, gender, role |

Primary key.

Foreign Key.

Primary key as well as Foreign key.

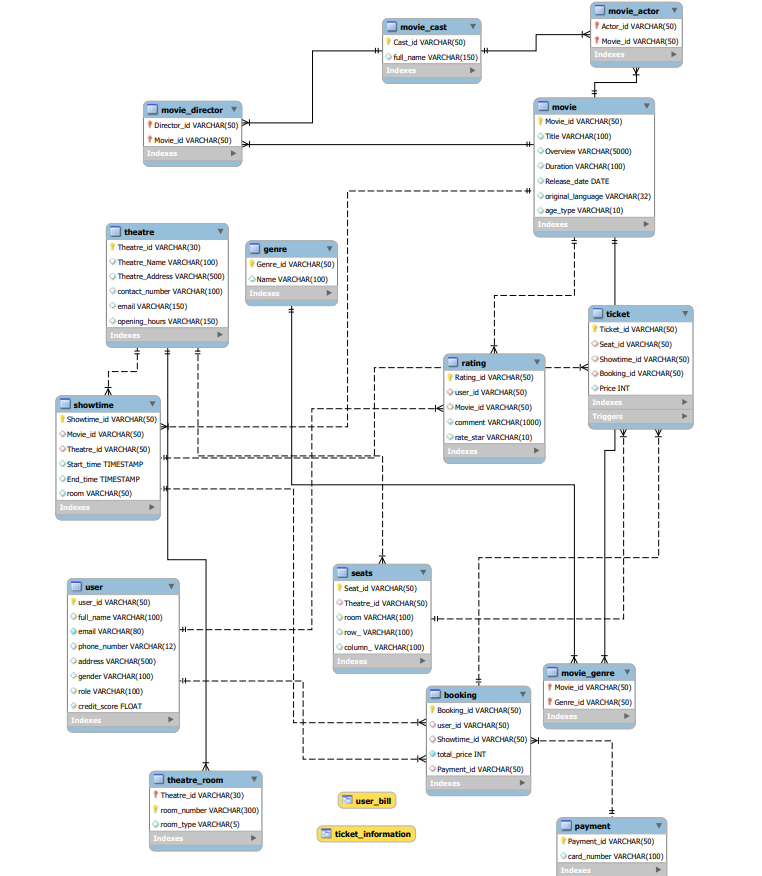
**Data Normalisation:**

* A schema R is in third normal form (3NF) if for all   
  α → β in F+
* at least one of the following holds:

1. α → β is trivial.
2. α is a superkey of R.

* Each attribute A in β – α is contained in a candidate key of R.
* The database of the “movie reservation system” is normalized to 3NF. This ensures that there is no transitive dependency between the non-prime attributes in the relations of the database.

**Relational Model:**



In the above relational schema, the main(key) tables are user, theatre, movie, ticket and payment. All the other tables are consisting of complimentary information for these key tables. Each sub table is connected to the main table through various types of relations like one to many, many to many, etc. The connection is done using foreign keys which references to the parent table as mentioned in the relational schema.

**Work Flow:**

Using the workflow mentioned in the diagram above, we wrote queries in Workbench and implemented these in python jupyter notebook. Our main interface to run the model is via jupyter notebook.

**Basic SQL query coverage:**

1. 2 Views.
2. 3 Stored Procedures.
3. 1 Trigger.
4. 3 Functions.

**Description of SQL coverage:**

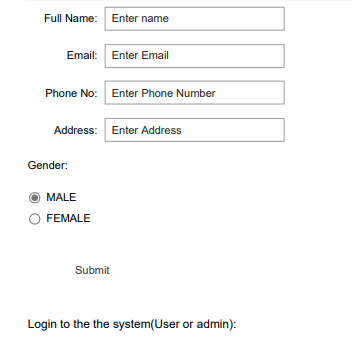
1. 2 Views.
   1. User\_bill (contains billing information)
   2. Ticket\_info (contains ticket information)
2. 3 Stored Procedures.
   1. Makebooking (book tickets for user)
   2. New\_movie(create a new movie with information)
   3. New\_showtime(set up a show at a given theatre and selected time)
3. 1 Trigger
   1. Update\_creditscore(update credit score of user after every booking)
4. 3 Functions.
   1. Calc\_discount(calculate the discounted ticket price based on user credit)
   2. Ratio\_booked\_seats(occupance ratio of the movie show)
   3. Revenue\_generated(calculate totals revenue earned from ticket sales of a movie)

**APPLICATION INTERFACE:**

**Note:** The source code pdf covers all the features in a well-documented manner, a brief description is given below.

Features for the user interface:

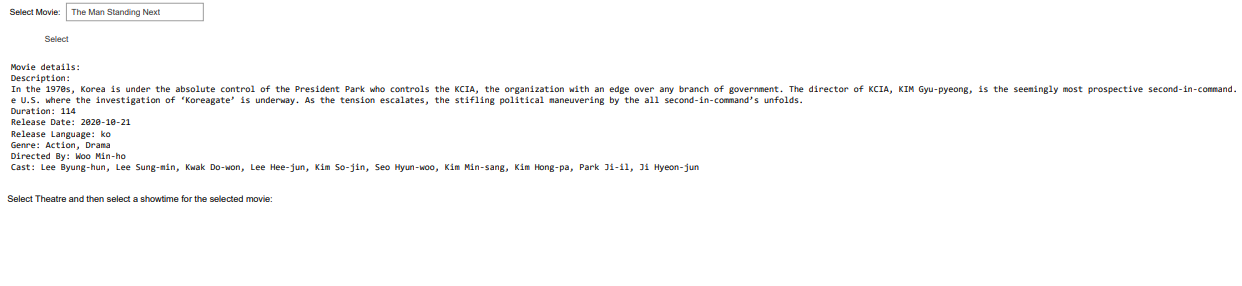
1. Registering a new user by giving out a form for the following information that is to be filled by the user.



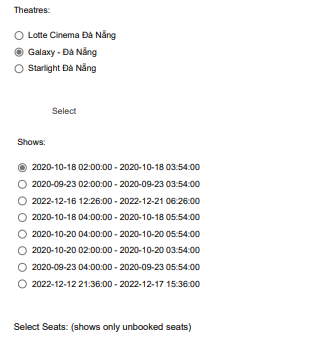
1. Logging in with the email and password to access features.



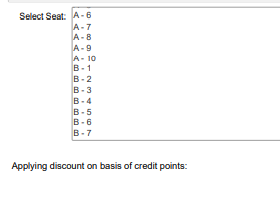
1. The user would be able to view movies through a scrollbar and select the one they want to know details about for example: Description of the movie, Duration, Genre etc.



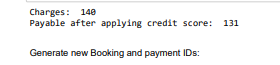
1. Next feature is selecting the theatre and showtime for the movie the user selected in the previous step.



1. Seats can be selected; multiple seat selection is also possible (although we have set a limit of maximum of 4 seat selections).

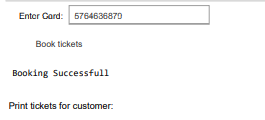


1. Total charges would be visible to the user along with auto generation of unique booking and payment ids.

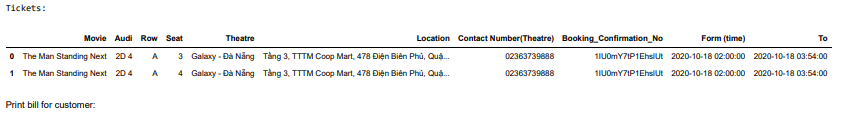




1. The user would have to fill out their card details which would also give them some credit points with every transaction. These points give out a discount automatically which is reflected in the total price.



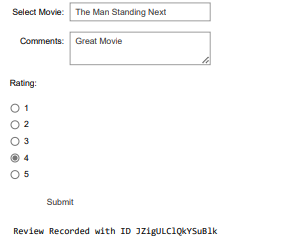
1. A view is used to show the details of the ticket to the user for the movie they have booked.



1. The user will also get a bill(using a view), which is different from a ticket and has the important ids that are helpful in tracking any kind of transaction failure with the theatre and payment issues with their banks.

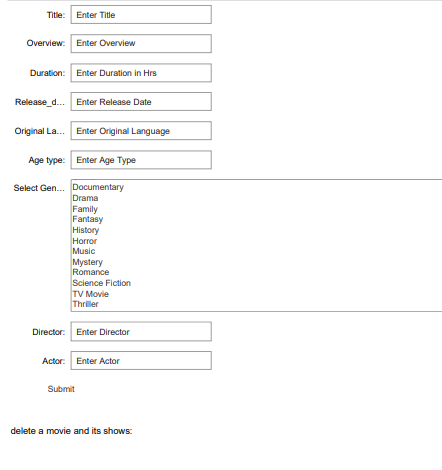


1. Reviews can be taken from the user along with rate star for a particular movie experience.

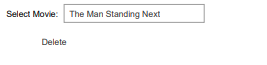


Features for the admin interface:

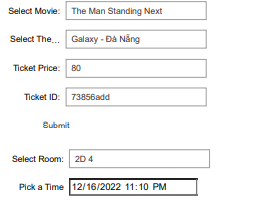
1. The admin would be able to insert details of a new user. These were done by calling a stored procedure.



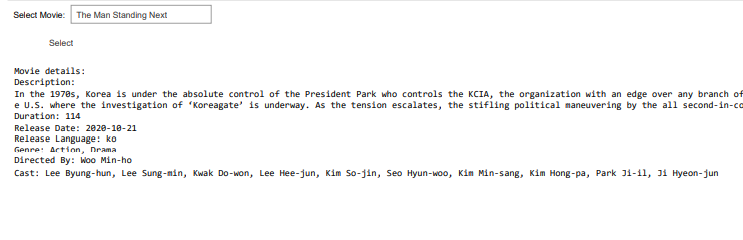
1. With the help of “delete on cascade”, the admin can delete all the records of a movie with just the click of a button. Admin can select the title of the movie and delete all records of that movie.

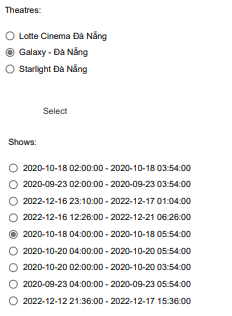


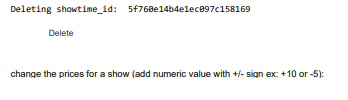
1. Admin can also create a new show for a show time at a selected theatre. This is also done by calling a stored procedure.



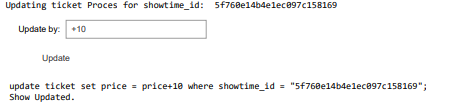
1. A showtime can be also deleted for a particular movie by selecting the movie, then the showtime and theatre.



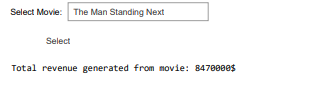




1. The prices for a movie can also be updated using the operators +/-.



1. Total revenue generated by a movie can be seen by selecting its title.



1. The occupancy ratio i.e., the number of tickets sold by the total number of seats can be viewed for a particular theatre and a movie.



1. Further features that would help the admin analyse the business are covered below in the data reporting section.

**Data Reporting:**

* We have created graphs and reports to provide in-depth analytics that are essential in better understanding our model and helps admin of the system.

The following reports show insights about various aspects of our database that will help the “admin” of the movie reservation system.

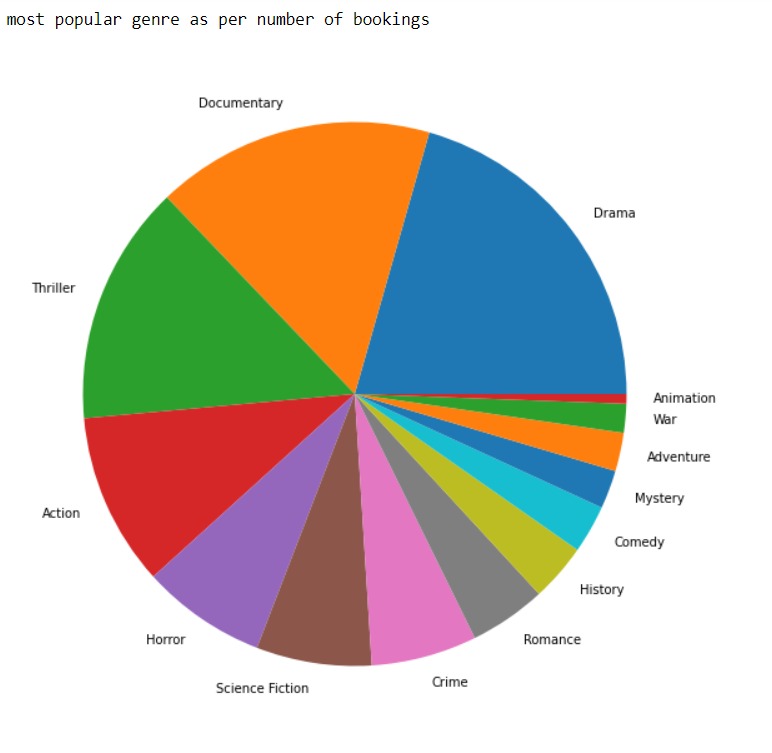


Figure : Most popular genre as per number of bookings.

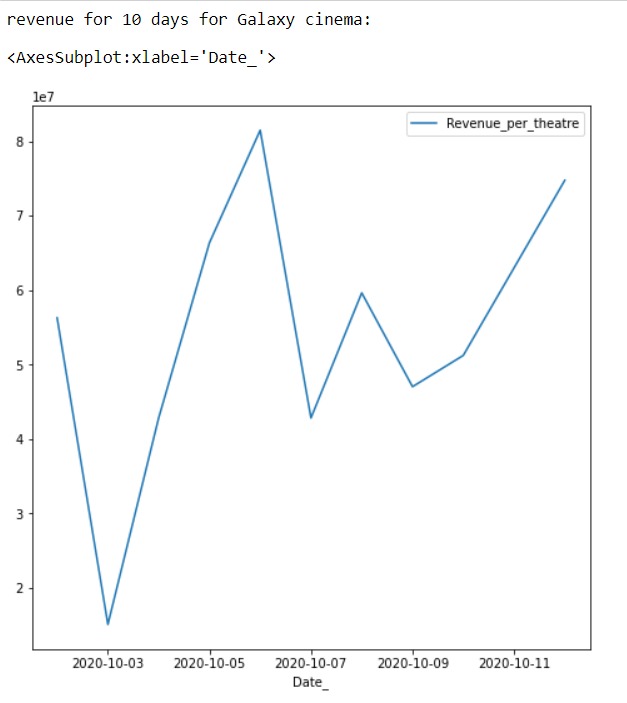


Figure : Revenue for 10 days for Galaxy Cinema.

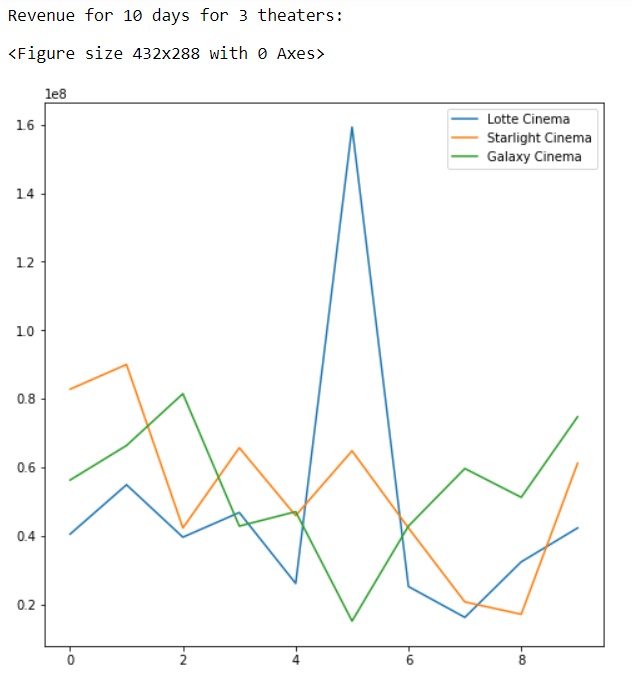


Figure : Revenue comparison for 10 days for 3 theatres.

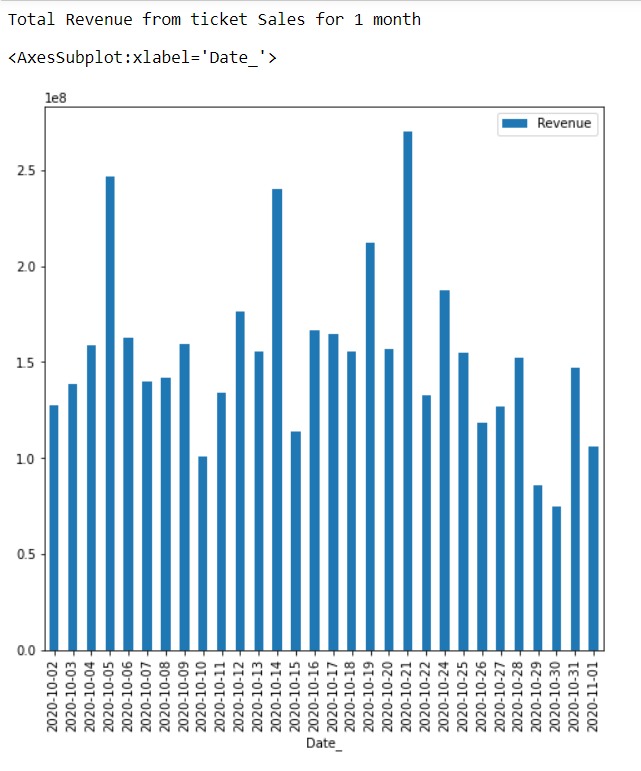


Figure : Total revenue from ticket sales of 1 month.

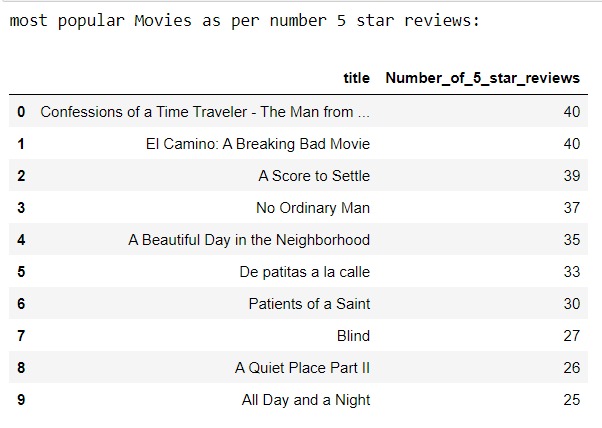


Figure : Most popular movie as per number of 5 star reviews.

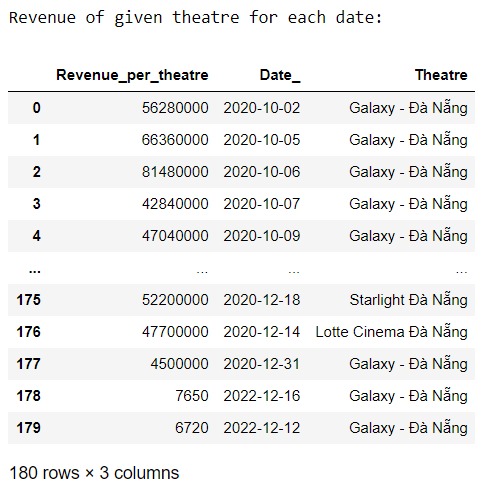


Figure : Revenue of given theatre for each date.

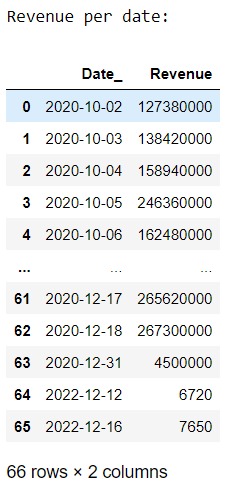


Figure : Revenue per Date/Day.

**Future Enhancements:**

1. Membership clubs can be created for users with different levels of perks for them.
2. Encryption can be done to protect user’s card numbers.
3. Additional payment options can be added and where coupons could be redeemed to get a free ticket or a discount.
4. A menu system can be created for customers to order food and drinks during the show.

**CONCLUSION:**

The “Movie Reservation System” model has been developed to show the successful interaction of database with a high-level language to use that data for accessing many features and developing an environment in itself. The project helped in understanding the core concepts of data extraction, SQL querying, Database connection, Data pre-processing and conversion/usage off multiple types of data format. It also provided a hands-on experience to develop a working model using a high-level language (python in our case), and interacting with multiple interfaces of data and languages. Data normalisation and creating relational schema/model has helped in understanding creating a database and its manipulations for effective usage of raw data.

More features can include a proper Graphical User Interface and converting this model into a website or an application which would query straight from the database’s connection. Additional features which could come in handy are covered in the aforementioned future enhancements.

The advice for future students would be to start with unstructured data having different formats (Json,xml) and convert it to relational data instead of inserting data manually. Moreover, attempting to find more insights about the data would contribute to creating queries of greater complexity. It would help in getting a better understanding of a lot of concepts for this subject, which are essential and hands-on experience of the same would benefit them in future. The main objective should be to implement most concepts taught in the class, in their projects.