MOVIE RESERVATION SYSTEM.

Project Participants:

1. Ataish Nehra, NUID: 002919899
2. Pritish Arora, NUID: 002781425

- **Labour Division:**

Ataish Nehra: Data mining, cleaning and pre-processing, SQL queries(equally divided), MongoDB, plots creation (data analytics).

Pritish Arora: SQL queries, python widgets, SQL queries execution in python, Data loader script from mongo to SQL.

Note: SQL queries have been divided equally amongst the team members.

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:

categories	17-11-2022 19:13	JSON Source File	4 KB
comments	17-11-2022 19:13	JSON Source File	3,663 KB
movie_category	17-11-2022 19:13	JSON Source File	172 KB
movies	17-11-2022 19:13	JSON Source File	430 KB
notifications	17-11-2022 19:13	JSON Source File	13 KB
people	17-11-2022 19:13	JSON Source File	1,446 KB
products	17-11-2022 19:13	JSON Source File	3 KB
promotions	17-11-2022 19:13	JSON Source File	3,684 KB
reservations	17-11-2022 19:13	JSON Source File	40 KB
seats	17-11-2022 19:13	JSON Source File	641 KB
show_times	17-11-2022 19:13	JSON Source File	782 KB
theatres	17-11-2022 19:13	JSON Source File	6 KB
tickets	17-11-2022 19:13	JSON Source File	46,028 KB
users	17-11-2022 19:13	JSON Source File	189 KB

Entity relation Diagram:
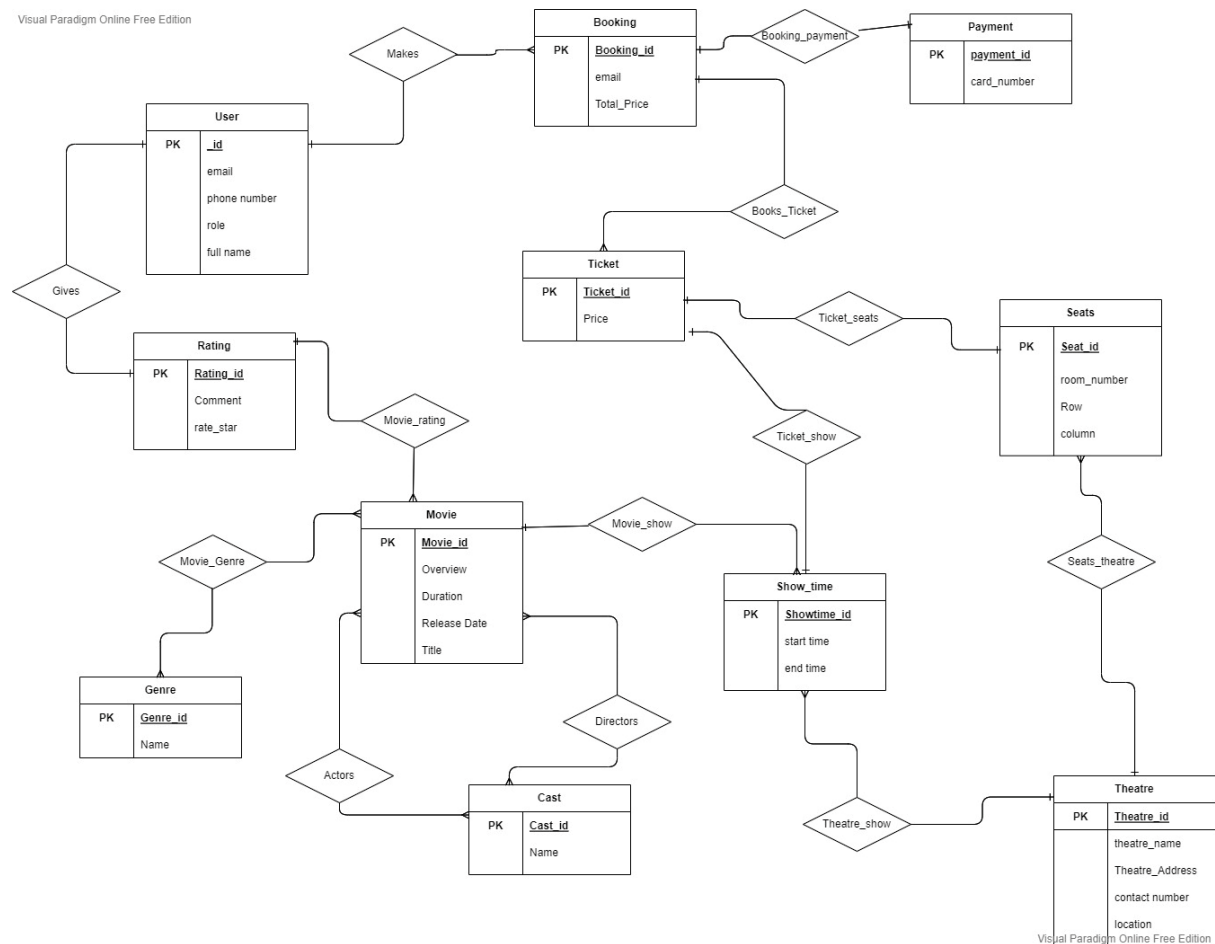
MongoDB:

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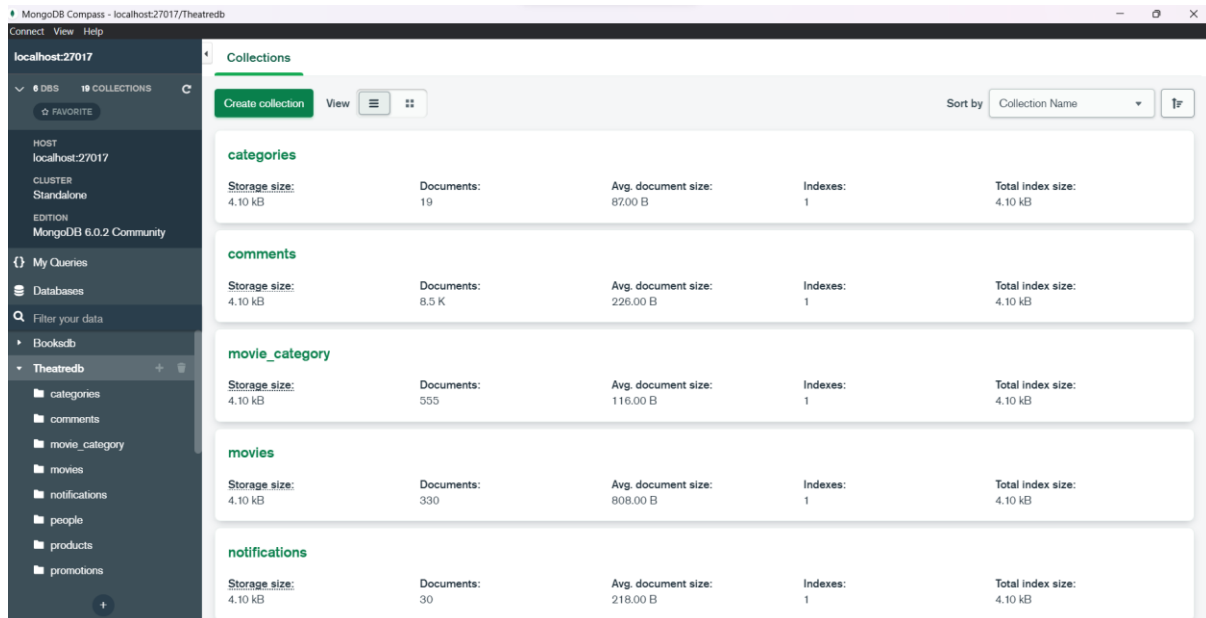
The data collected in MongoDB is connected to python using mysqlconnector and the engines are created to parse the data into SQL workbench.

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.

```
13 •   create table Movie
14     (Movie_id varchar(50) primary key,
15     Title varchar(100),
16     Overview varchar(5000),
17     Duration varchar(100),
18     Release_date date,
19     original_language varchar(32),
20     age_type varchar(10));
21     -- done
22
23
24 •   create table Theatre
25     (Theatre_id varchar(30) primary key,
26     Theatre_Name varchar(100),
27     Theatre_Address varchar(500),
28     contact_number varchar(100),
29     email varchar(150),
30     opening_hours varchar(150));
31     -- done
```

```sql
33 •    create table Showtime
34  ⊖  (Showtime_id varchar(50) primary key,
35      Movie_id varchar(50),
36      Theatre_id varchar(50),
37      Start_time timestamp,
38      End_time timestamp,
39      room varchar(50),
40      foreign key (Movie_id) references Movie(Movie_id) on delete cascade,
41      foreign key (Theatre_id) references Theatre(Theatre_id) on delete cascade);
42      -- done
43
44 •    create table Payment
45  ⊖  (Payment_id varchar(50) primary key,
46      Booking_id varchar(50),
47      card_number varchar(100));
48      -- done
```

Data Loader Script:

Install pymongo[srv]

pip install pymongo[srv] pip install PyMySQL

Fetch data from MongoDb

```python
from pymongo import MongoClient
import pandas as pd
from sqlalchemy import create_engine
from random import randint
import pymysql
```
[153]   Python

```python
client = MongoClient('localhost',27017)
```
[154]   Python

```python
client
```
[155]   Python

··· MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True)

```python
db = client['TheatreDb']
```
[156]   Python

```python
engine = create_engine("mysql+pymysql://{user}:{pw}@localhost/{db}".format(user="root",pw="password",db="theatre"))
```
[157]   Python

Getting data form indivisual collections in TheatreDb to populate respective tables.

```python
users = db['users']

users_cursor = users.find()
users_df = pd.DataFrame(list(users_cursor))

users_df.columns
users_df = users_df[['_id', 'full_name', 'email', 'phone_number', 'address', 'gender', 'role']]

users_df = users_df.rename(columns = {'_id':'user_id'})

#display(users_df)
users_df.to_sql('user', con = engine, if_exists = 'append', chunksize = 1000, index= False)
```
[158]   Python

··· 297

```
movies = db['movies']

movies_cursor = movies.find()
movies_df = pd.DataFrame(list(movies_cursor))

movies_df.columns
movies_df = movies_df[['_id', 'title', 'overview', 'duration','released_date', 'original_language','age_type']]

movies_df = movies_df.rename(columns = {'_id':'Movie_id','title':'Title','overview':'Overview','duration':'Duration',
                                        'released_date':'Release_date'})

movies_df.to_sql('movie', con = engine, if_exists = 'append', chunksize = 1000,index= False)
```
[159]
```
... 330
```

```
theatres = db['theatres']

theatres_cursor = theatres.find()
theatres_df = pd.DataFrame(list(theatres_cursor))

theatres_df.columns
theatres_df = theatres_df[['_id', 'name', 'address','phone_number', 'email', 'opening_hours']]

theatres_df = theatres_df.rename(columns = {'_id':'Theatre_id','name':'Theatre_name','address':'Theatre_Address'
                                            ,'phone_number':'contact_number'})

theatres_df.to_sql('theatre', con = engine, if_exists = 'append', chunksize = 1000, index= False)
```
[160]
```
... 5
```

```
tickets = db['tickets']

tickets_cursor = tickets.find()
tickets_df = pd.DataFrame(list(tickets_cursor))

tickets_df.columns
tickets_df = tickets_df[['_id','seat', 'show_time', 'reservation', 'price']]

# ab = tickets_df.groupby(['reservation'])
# print(ab.first())

tickets_df = tickets_df.rename(columns = {'_id':'Ticket_id','seat':'Seat_id','show_time':'Showtime_id',
                                          'reservation':'Booking_id','price':'Price'
                                          })

tickets_df = seats_df.merge(tickets_df, on='Seat_id', indicator=True)
tickets_df = tickets_df[['Ticket_id','Seat_id','Showtime_id','Booking_id','Price']]

tickets_df = shows_df.merge(tickets_df, on='Showtime_id', indicator=True)
tickets_df = tickets_df[['Ticket_id','Seat_id','Showtime_id','Booking_id','Price']]

#display(tickets_df)

tickets_df.to_sql('ticket', con = engine, if_exists = 'append', chunksize = 1000, index= False)
```
[173]
```
... 120462
```

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 $\alpha \rightarrow \beta$ in F^+

➢ at least one of the following holds:

1. $\alpha \rightarrow \beta$ is trivial.

2. α is a superkey of R.

- Each attribute A in $\beta - \alpha$ 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:
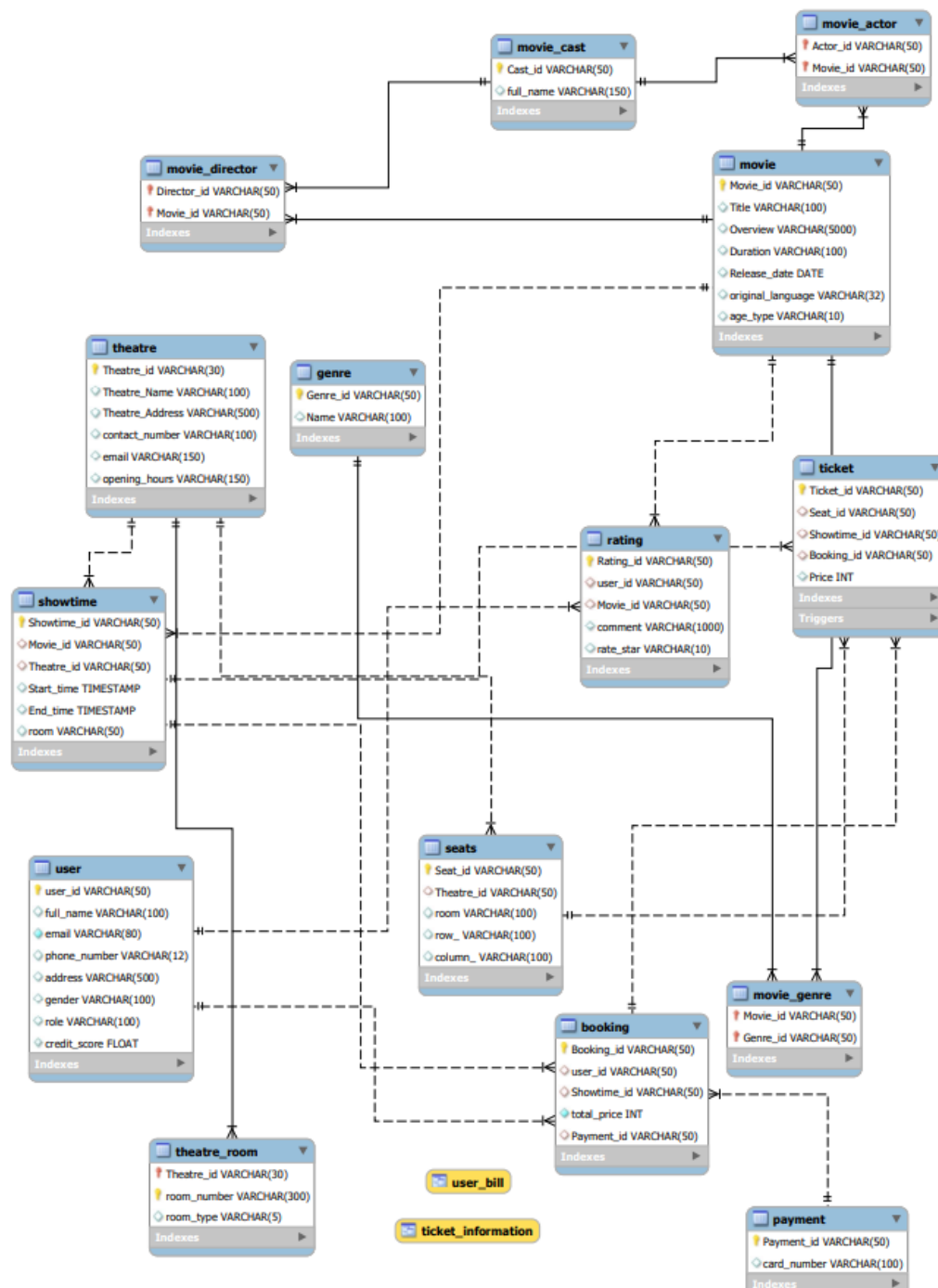
In the above relational schema, the main(key) tables are user, theatre, movie, ticket and payment. All the other tables are consisting of complimentary information for these key tables. Each sub table is connected to the main table through various types of relations like one to many, many to many, etc. 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:
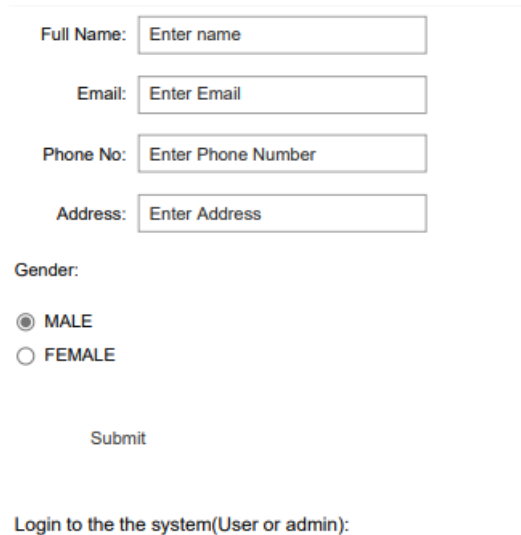
1. 2 Views.

    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.

<div align="center">

Full Name: | Enter name

Email: | Enter Email

Phone No: | Enter Phone Number

Address: | Enter Address

Gender:

⦿ MALE

◯ FEMALE

Submit

Login to the the system(User or admin):

</div>

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

<div align="center">

Username: | hoc081098@gmail.com

Password: | •••

Connect!
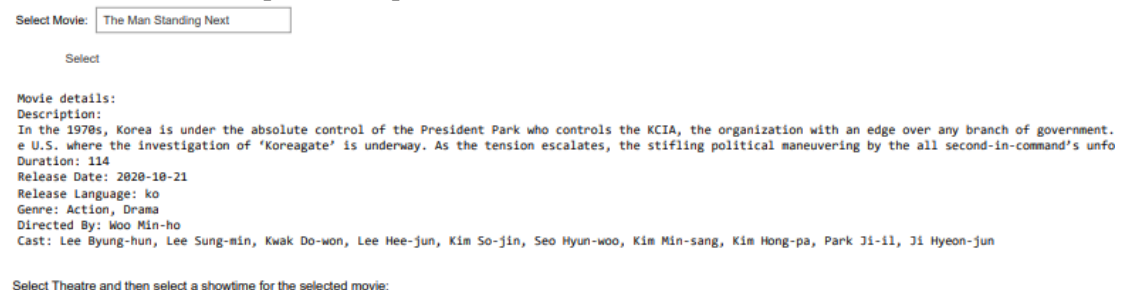
</div>

```
('5f6a22e3075f5b523f6085a4', 'Petrus Hoc', 'hoc081098@gmail.com', '0363438135', '123', 'MALE', 'USER', 30.0)
Welcome User: Petrus Hoc
Status: Logged In
<mysql.connector.connection_cext.CMySQLConnection object at 0x00000205C2CE2790>

initial
<mysql.connector.connection_cext.CMySQLConnection object at 0x00000205C2CE2C70>
```

User selects a movie:

3. 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.

```
Select Movie:  The Man Standing Next

        Select

Movie details:
Description:
In the 1970s, Korea is under the absolute control of the President Park who controls the KCIA, the organization with an edge over any branch of government.
e U.S. where the investigation of 'Koreagate' is underway. As the tension escalates, the stifling political maneuvering by the all second-in-command's unfo
Duration: 114
Release Date: 2020-10-21
Release Language: ko
Genre: Action, Drama
Directed By: Woo Min-ho
Cast: Lee Byung-hun, Lee Sung-min, Kwak Do-won, Lee Hee-jun, Kim So-jin, Seo Hyun-woo, Kim Min-sang, Kim Hong-pa, Park Ji-il, Ji Hyeon-jun
```

Select Theatre and then select a showtime for the selected movie:

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

Theatres:

○ Lotte Cinema Đà Nẵng
◉ Galaxy - Đà Nẵng
○ Starlight Đà Nẵng

Select

Shows:

◉ 2020-10-18 02:00:00 - 2020-10-18 03:54:00
○ 2020-09-23 02:00:00 - 2020-09-23 03:54:00
○ 2022-12-16 12:26:00 - 2022-12-21 06:26:00
○ 2020-10-18 04:00:00 - 2020-10-18 05:54:00
○ 2020-10-20 04:00:00 - 2020-10-20 05:54:00
○ 2020-10-20 02:00:00 - 2020-10-20 03:54:00
○ 2020-09-23 04:00:00 - 2020-09-23 05:54:00
○ 2022-12-12 21:36:00 - 2022-12-17 15:36:00

Select Seats: (shows only unbooked seats)

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

Select Seat:
```
A - 6
A - 7
A - 8
A - 9
A - 10
B - 1
B - 2
B - 3
B - 4
B - 5
B - 6
B - 7
```

Applying discount on basis of credit points:

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

```
Charges:  140
Payable after applying credit score:  131
```

Generate new Booking and payment IDs:

Booking ID: 1IUGmY7tP1EhslUt
Payment ID: wwWoA4TVkomuB7MM

Make payment by adding card number:

7. 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.

Enter Card: 5764636879

Book tickets

Booking Successfull

Print tickets for customer:

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

Tickets:

	Movie	Audi	Row	Seat	Theatre	Location	Contact Number(Theatre)	Booking_Confirmation_No	Form (time)	To
0	The Man Standing Next	2D 4	A	3	Galaxy - Đà Nẵng	Tầng 3, TTTM Coop Mart, 478 Điện Biên Phủ, Quả...		02363739888	1IU0mY7tP1EhslUt	2020-10-18 02:00:00	2020-10-18 03:54:00
1	The Man Standing Next	2D 4	A	4	Galaxy - Đà Nẵng	Tầng 3, TTTM Coop Mart, 478 Điện Biên Phủ, Quả...		02363739888	1IU0mY7tP1EhslUt	2020-10-18 02:00:00	2020-10-18 03:54:00

Print bill for customer:

9. 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.

Bill:

Bill_Number	Movie	Theatre	Theatre Contact Number	Payee	User Email	Amount Paid
0 wwWoA4TVkomu87MM	The Man Standing Next	Galaxy - Đà Nẵng	02363739888	Petrus Hoc	hoc081098@gmail.com	131

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

Select Movie: The Man Standing Next

Comments: Great Movie

Rating:

○ 1
○ 2
○ 3
◉ 4
○ 5

Submit

Review Recorded with ID JZigULClQkYSuBlk

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.

Title: Enter Title

Overview: Enter Overview

Duration: Enter Duration in Hrs

Release_d... Enter Release Date

Original La... Enter Original Language

Age type: Enter Age Type

Select Gen... Documentary
Drama
Family
Fantasy
History
Horror
Music
Mystery
Romance
Science Fiction
TV Movie
Thriller

Director: Enter Director

Actor: Enter Actor

Submit

2. 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.

Select Movie: The Man Standing Next

Delete

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

Select Movie:	The Man Standing Next
Select The...	Galaxy - Đà Nẵng
Ticket Price:	80
Ticket ID:	73856add

Submit

| Select Room: | 2D 4 |
| Pick a Time | 12/16/2022 11:10 PM |

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

Select Movie: The Man Standing Next

Select

Movie details:
Description:
In the 1970s, Korea is under the absolute control of the President Park who controls the KCIA, the organization with an edge over any branch of
e U.S. where the investigation of 'Koreagate' is underway. As the tension escalates, the stifling political maneuvering by the all second-in-co
Duration: 114
Release Date: 2020-10-21
Release Language: ko
Genre: Action, Drama
Directed By: Woo Min-ho
Cast: Lee Byung-hun, Lee Sung-min, Kwak Do-won, Lee Hee-jun, Kim So-jin, Seo Hyun-woo, Kim Min-sang, Kim Hong-pa, Park Ji-il, Ji Hyeon-jun

Theatres:

○ Lotte Cinema Đà Nẵng
◉ Galaxy - Đà Nẵng
○ Starlight Đà Nẵng

Select

Shows:

○ 2020-10-18 02:00:00 - 2020-10-18 03:54:00
○ 2020-09-23 02:00:00 - 2020-09-23 03:54:00
○ 2022-12-16 23:10:00 - 2022-12-17 01:04:00
○ 2022-12-16 12:26:00 - 2022-12-21 06:26:00
◉ 2020-10-18 04:00:00 - 2020-10-18 05:54:00
○ 2020-10-20 04:00:00 - 2020-10-20 05:54:00
○ 2020-10-20 02:00:00 - 2020-10-20 03:54:00
○ 2020-09-23 04:00:00 - 2020-09-23 05:54:00
○ 2022-12-12 21:36:00 - 2022-12-17 15:36:00

Deleting showtime_id: 5f760e14b4e1ec097c158169

Delete

change the prices for a show (add numeric value with +/- sign ex: +10 or -5):

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

Updating ticket Proces for showtime_id: 5f760e14b4e1ec097c158169

| Update by: | +10 |

Update

update ticket set price = price+10 where showtime_id = "5f760e14b4e1ec097c158169";
Show Updated.

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

Select Movie: [The Man Standing Next]

Select

Total revenue generated from movie: 8470000$

7. 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.

Select Movie: [The Man Standing Next]

Select The···: [Galaxy - Đà Nẵng]

Select

Occupancy Ratio: 0.00509338

8. 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.
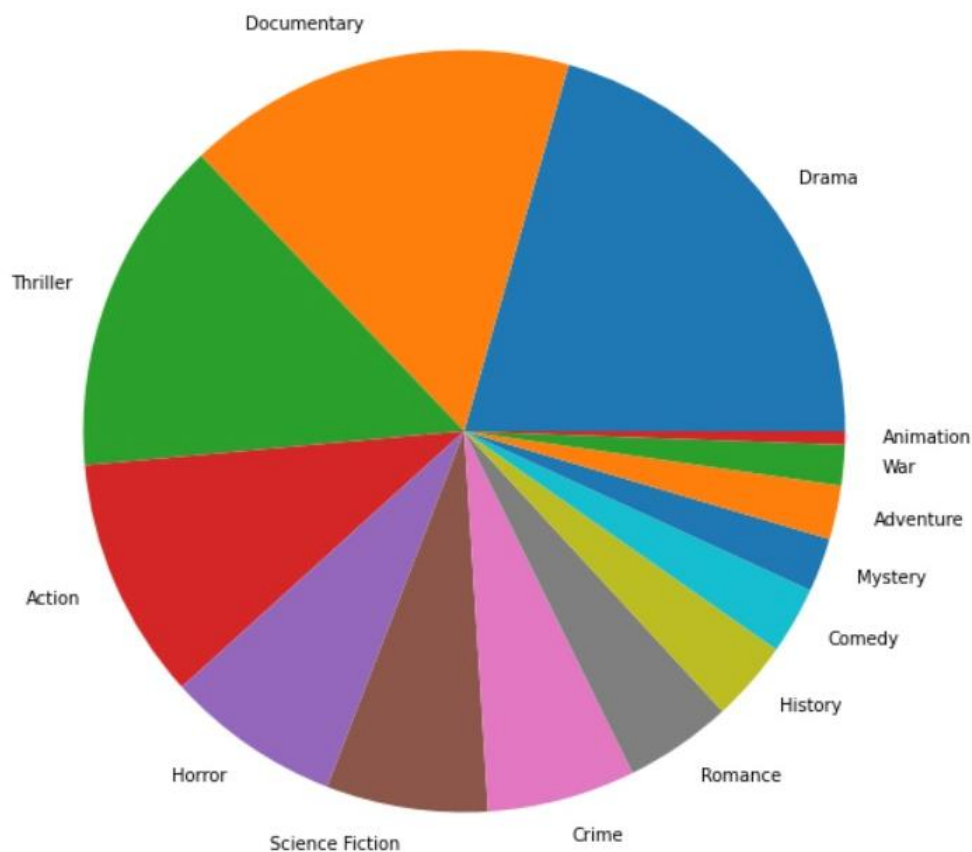

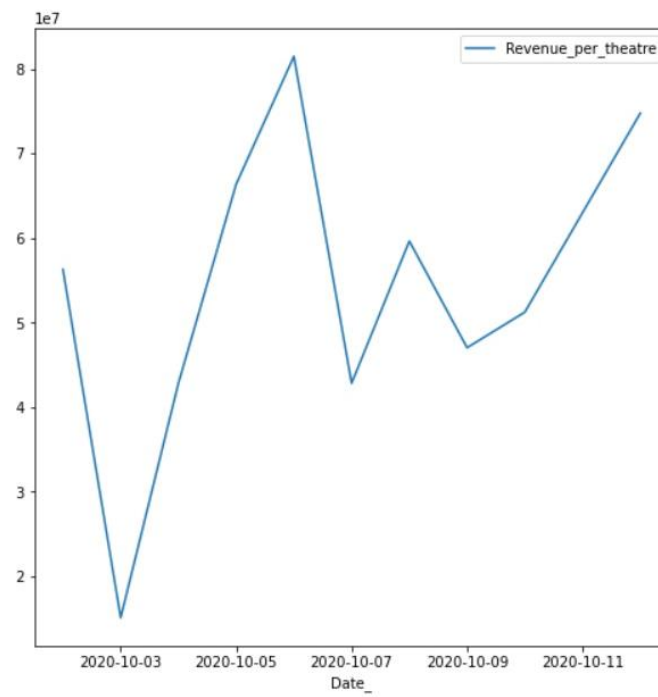*Figure 1: Most popular genre as per number of bookings.*

<AxesSubplot:xlabel='Date_'>
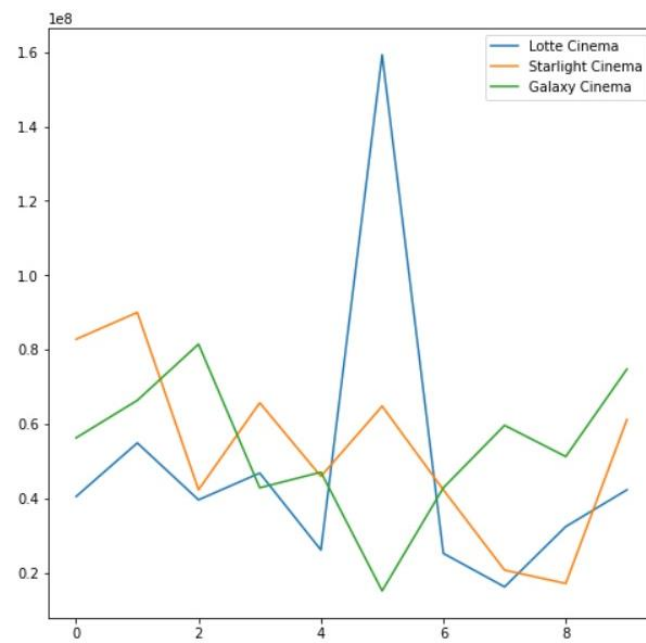

Figure 2: Revenue for 10 days for Galaxy Cinema.

<Figure size 432x288 with 0 Axes>

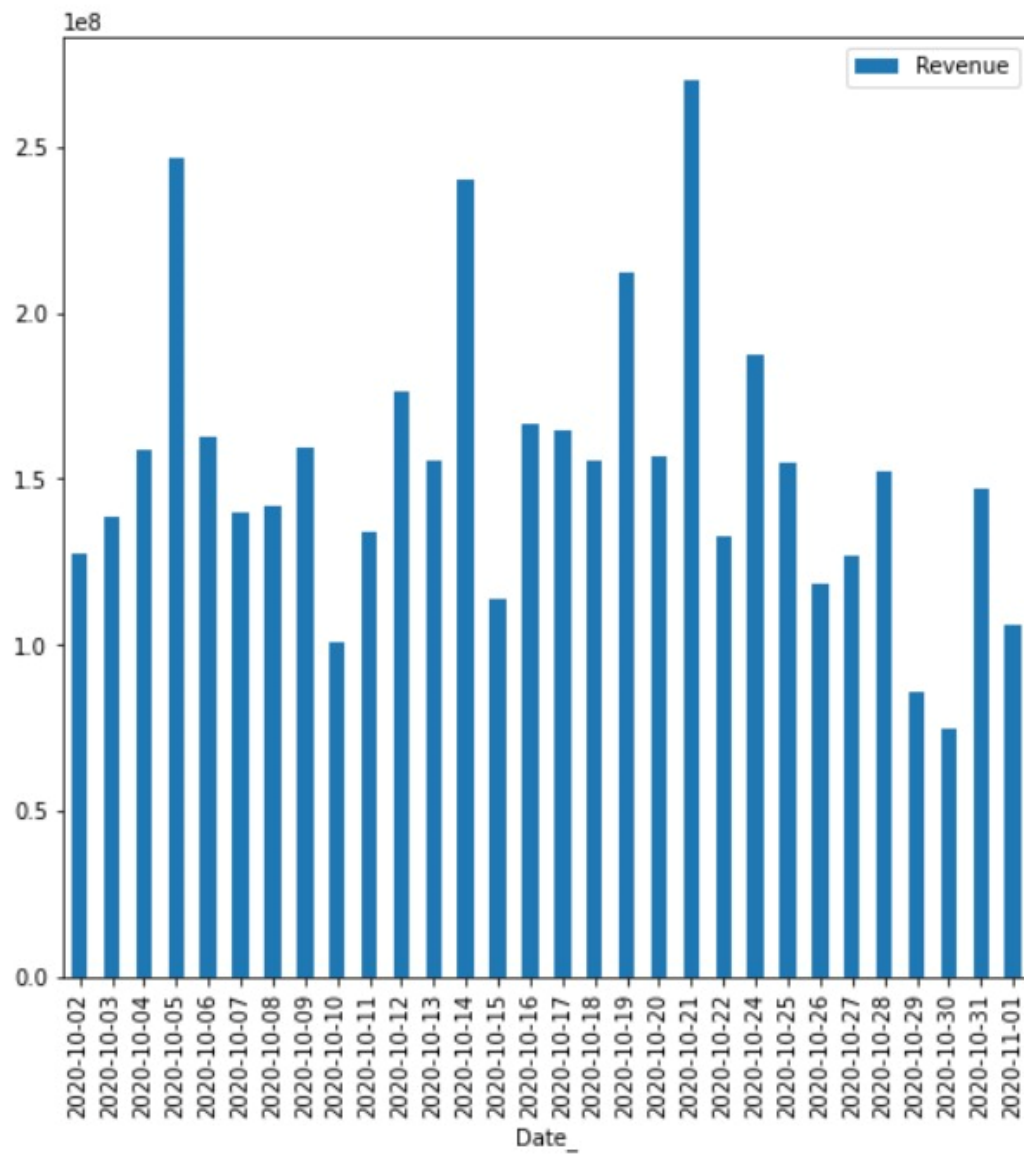

Figure 3: Revenue comparison for 10 days for 3 theatres.

`<AxesSubplot:xlabel='Date_'>`

Figure 4: Total revenue from ticket sales of 1 month.

	title	Number_of_5_star_reviews
0	Confessions of a Time Traveler - The Man from ...	40
1	El Camino: A Breaking Bad Movie	40
2	A Score to Settle	39
3	No Ordinary Man	37
4	A Beautiful Day in the Neighborhood	35
5	De patitas a la calle	33
6	Patients of a Saint	30
7	Blind	27
8	A Quiet Place Part II	26
9	All Day and a Night	25

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

	Revenue_per_theatre	Date_	Theatre
0	56280000	2020-10-02	Galaxy - Đà Nẵng
1	66360000	2020-10-05	Galaxy - Đà Nẵng
2	81480000	2020-10-06	Galaxy - Đà Nẵng
3	42840000	2020-10-07	Galaxy - Đà Nẵng
4	47040000	2020-10-09	Galaxy - Đà Nẵng
...	...	...	...
175	52200000	2020-12-18	Starlight Đà Nẵng
176	47700000	2020-12-14	Lotte Cinema Đà Nẵng
177	4500000	2020-12-31	Galaxy - Đà Nẵng
178	7650	2022-12-16	Galaxy - Đà Nẵng
179	6720	2022-12-12	Galaxy - Đà Nẵng

180 rows × 3 columns

Figure 6: Revenue of given theatre for each date.

	Date_	Revenue
0	2020-10-02	127380000
1	2020-10-03	138420000
2	2020-10-04	158940000
3	2020-10-05	246360000
4	2020-10-06	162480000
...	...	...
61	2020-12-17	265620000
62	2020-12-18	267300000
63	2020-12-31	4500000
64	2022-12-12	6720
65	2022-12-16	7650

66 rows × 2 columns

Figure 7: 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.