DMA USE CASE STUDY REPORT

Group -18

Student Names: Gangadhar Kakarlathotta, Vipul Kumar Akula

I. Introduction

Problem Statement:

The traditional way of buying tickets for a movie is that the customer needs to go to the theatre, stand in the queue for hours and buy tickets. This is a more time consuming process both for the customer and the theatre owners.

Goal:

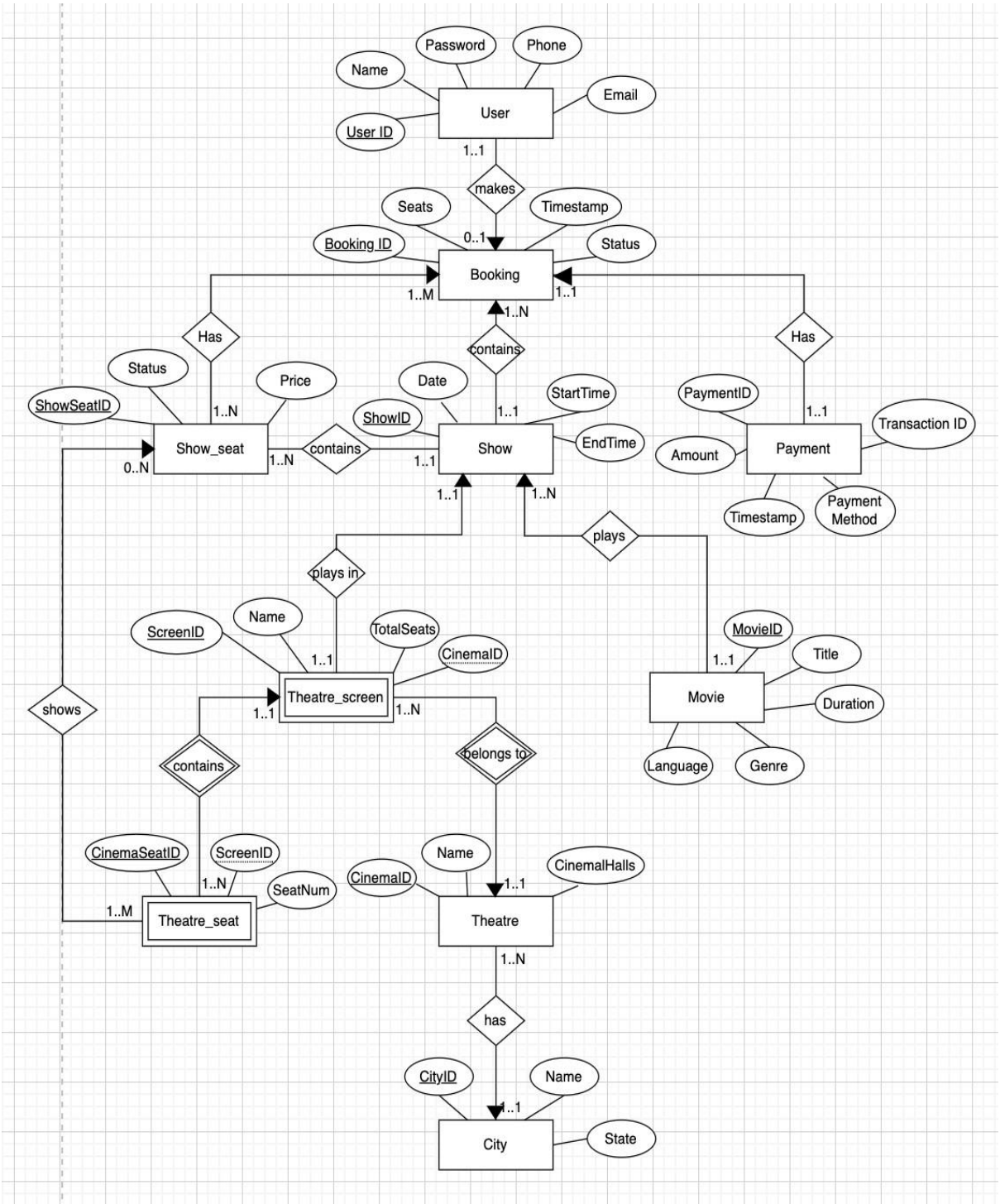
Our goal is to design a database to streamline the process of booking tickets online, which helps the users to select shows and seats of their choice and also makes it easier for the theatre owners to control the crowd and play movies on time.

Requirements:

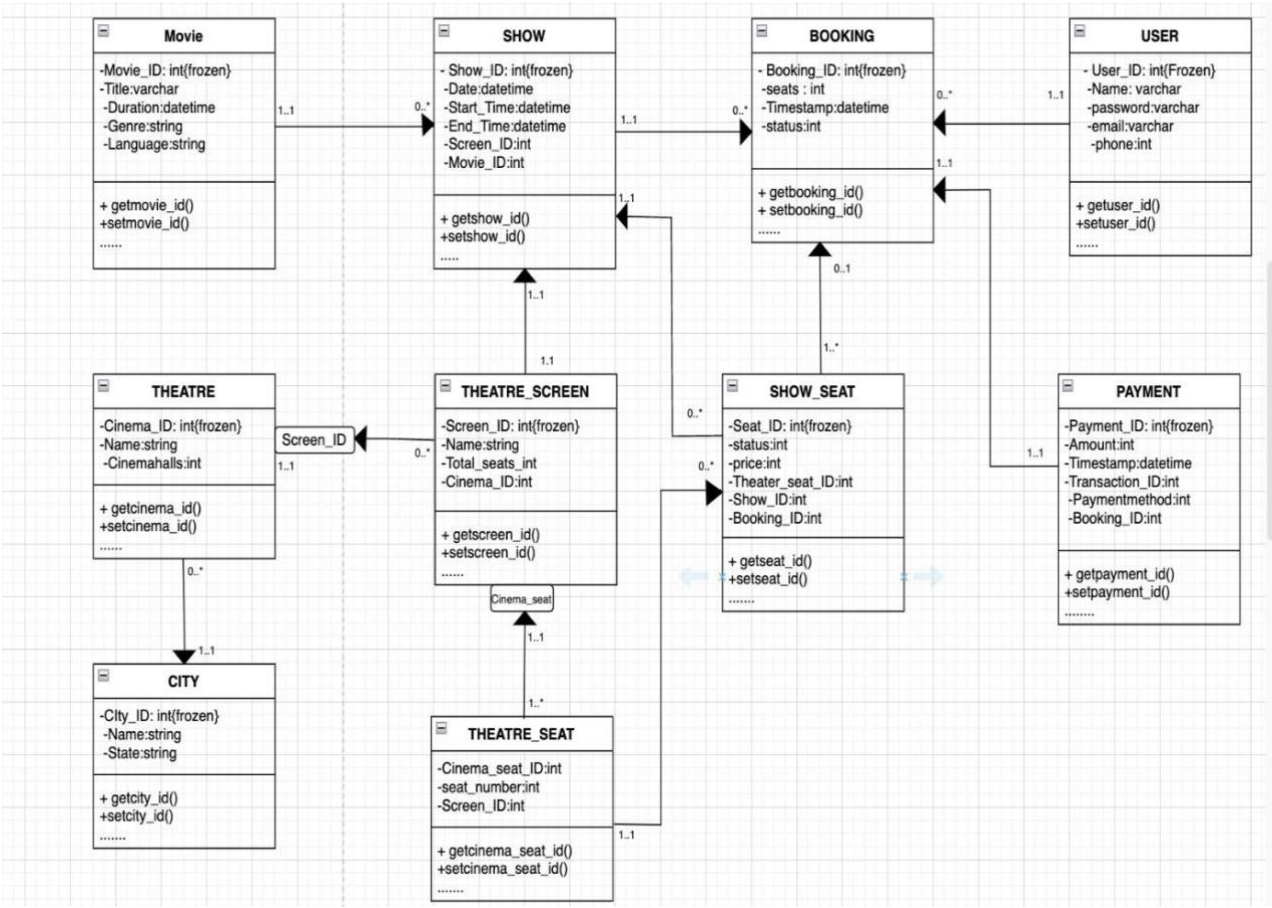
The system must be able to provide a list of the cities where affiliate theaters are found. Each movie theater may have several halls, and each hall may host one film show at a time. There will be several showings of each film. Movies should be searchable by title, language, genre, release date, and city name for the benefit of customers. When a consumer chooses a movie, the service needs to provide the theaters showing it and the shows it has scheduled. The customer ought to be able to choose a show at a specific theater and purchase tickets. The customer should be shown the theater's seating arrangement by the service. The customer needs to be allowed to pick from a variety of seats in accordance with their preferences. The user should be able to tell the difference between booked and available seats. When a new movie is released, as well as when a booking is made or canceled, the system should notify users. Customers using our system should be able to pay with debit or with credit card. No two consumers should be able to reserve the same seat, according to the system.

II. Conceptual Data Modelling

a. EER Model:



b. UML Diagram



III. Mapping Conceptual Model to Relational Model

(Primary Keys are Underlined and Foreign Keys are indicated in *Italics*.)

USER(<u>UserID</u>, Name, Password, Phone, Email)

Assumptions: Phone and Email should be single valued attributes and unique to each user.

BOOKING(<u>BookingID</u>, Seats, Timestamp, Status, *UserID*, *ShowID*)
FOREIGN KEY *UserID* refers to UserID in USER; NOT NULL
FOREIGN KEY *ShowID* refers to ShowID in SHOW; NOT NULL

PAYMENT(<u>PaymentID</u>, Amount, Timestamp, PaymentMethod, TransactionID, *BookingID*)
FOREIGN KEY *BookingID* refers to BookingID in BOOKING; NOT NULL

SHOW(<u>ShowID</u>, Date, StartTime, EndTime, *ScreenID*, *MovieID*)

FOREIGN KEY *ScreenID* refers to ScreenID in THEATRE_SCREEN; NOT NULL
FOREIGN KEY *MovieID* refers to MovieID in MOVIE; NOT NULL

SHOW_SEAT(ShowSeatID, Status, Price, *TheatreSeatID, ShowID, BookingID*)
FOREIGN KEY *TheatreSeatID* refers to TheatreSeatID in THEATRE_SEAT; NULL ALLOWED
FOREIGN KEY *ShowID* refers to ShowID in SHOW; NOT NULL
FOREIGN KEY *BookingID* refers to BookingID in BOOKING; NULL ALLOWED

THEATRE_SCREEN(ScreenID, Name, TotalSeats, *Cinema_ID*)
FOREIGN KEY *Cinema_ID* refers to Cinema_ID in THEATRE; NOT NULL

THEATRE_SEAT(TheatreSeatID, SeatNum, *ScreenID*)
FOREIGN KEY *ScreenID* refers to ScreenID in THEATRE_SCREEN; NOT NULL

THEATRE(Cinema_ID, Name, NumberOfScreens, *CityID*)
FOREIGN KEY *CityID* refers to CityID in CITY; NOT NULL

MOVIE(MovieID, Title, Language, Genre, Duration)

CITY(CityID, Name, State, Zipcode)

IV. Implementation of Relation Model via MySQL and NoSQL

a. SQL Implementation using MySQL workbench

Query 1 :

```
# 1 Top 5 States with most no of screens
with cte1 as (select c.state, sum(t.numberofscreens) as Total_no_of_screens
from city c join theatre t on c.idcity = t.cityid
group by c.state)

select * from cte1 c1
where 5 > (select count(c1.state) from cte1 c2 where c1.Total_no_of_screens < c2.Total_no_of_screens)
order by Total_no_of_screens desc;
```

1:70

Result Grid Filter Rows: Search Export:

state	Total_no_of_scree...
Wyoming	27
NorthDakota	27
Texas	24
Missouri	23
Maine	19
Florida	19

From this query we can infer that these are the top 5 states with most number of screens available in the US.

Query 2 :

```
#5 how does rating affect movie
select m.Rating, sum(b.seats) as Tickets_Booked
from booking b join shows s on b.showid = s.idshow
join movie m on s.movieid = m.idmovie
where b.status = 'booked'
group by m.rating
order by sum(b.seats) desc;
```

100% 26:34

Result Grid Filter Rows: Search Export:

Rating	Tickets_Book...
9	386
8	284
5	231
7	144
2	52

The purpose of this query is to understand how rating affects the movie sales and it allows the people to decide on which movie to watch.

Query 3 :

```
#6 City with most no of bookings
with cte1 as (select c.name, sum(b.seats) as Bookings
from city c join theatre t on c.idcity = t.cityid join theaterscreen ts on t.idcinema = ts.cinemaid
join shows s on ts.idscreen = s.screenid join booking b on s.idshow = b.showid
group by c.idcity)

select name as city from cte1
where Bookings = (select max(Bookings) from cte1);
```

100% 1:39

Result Grid Filter Rows: Search Export:

city
New Kavontown

The purpose of this query is to get the city which has most number of booking and it allows the theatre owners to construct more theatres in the city to get more revenue.

Query 4 :

```
#7 All time hit - Most No of Tickets sold
with blockbuster_cte as (
select m.title, sum(b.seats) as total_tickets
from movie m join shows s on m.idmovie = s.movieid join booking b on s.idshow = b.showid
group by m.title)

select title as All_time_hit, Total_tickets
from blockbuster_cte
where total_tickets = (select max(total_tickets) from blockbuster_cte);
```

100% 21:54

Result Grid Filter Rows: Search Export:

All_time_hit	Total_tick...
so-alice-began-telling-them-her-adventures-fr	79

The purpose of this query is to get the all-time hit with most tickets sold.

Query 5 :

```sql
#10 Top 5 Theatres with most occupancy in the year 2022
with cte1 as (select t.idcinema, t.name as Theatre, sum(b.seats) as Booked_Seats
from theatre t join theaterscreen ts on t.idcinema = ts.cinemaid
join shows s on ts.idscreen = s.screenid join booking b on s.idshow = b.showid
where year(b.timestamp) = 2022 and b.status = 'Booked'
group by t.idcinema),

cte2 as (select t.idcinema, t.name, sum(ts.totalseats) as Total_seats
from theatre t join theaterscreen ts on t.idcinema = ts.cinemaid
group by t.idcinema),

cte3 as (select a.Theatre, (a.Booked_seats/b.Total_Seats)*100 as Occupancy
from cte1 a join cte2 b on a.idcinema = b.idcinema)

select * from cte3 c1
where 5 > (select count(c1.Theatre) from cte3 c2 where c1.Occupancy < c2.Occupancy)
order by Occupancy desc;
```

100% 25:85

Result Grid Filter Rows: Search Export:

Theatre	Occupancy
Klein, Grant and Considine	85.18518518518519
Baumbach Ltd	73.25581395348837
Breitenberg Ltd	72.11538461538461
Macejkovic PLC	67.64705882352942
Larson, Mills and Ryan	60

The purpose of this query is to get the top five theatres with most occupancy(in %) which will help the theatre owners to setup more business near the theatres and improve the facilities in the theatres.

b. NoSQL Implementation using Mongo dB

Some tables from SQL have been mapped to collections in mongo db and some queries were written and executed based on these collections in online playground.

Query 1: Usage of Credit and Debit Card :

Mongo Playground run share $group docs

Configuration bson

```bson
    "amount": 15,
    "timestamp": "2012-12-04 06:40:28",
    "paymentmethod": "debit card",
    "transactionid": 484,
    "bookingid": 162
  },
  {
    "idpayment": 40,
    "amount": 13,
    "timestamp": "1998-05-13 23:53:47",
    "paymentmethod": "credit card",
    "transactionid": 410,
    "bookingid": 218
  },
  {
    "idpayment": 41,
    "amount": 15,
    "timestamp": "1974-10-03 04:23:09",
    "paymentmethod": "credit card",
    "transactionid": 433,
    "bookingid": 113
  },
  {
    "idpayment": 42,
    "amount": 12,
    "timestamp": "2011-02-24 02:41:58",
    "paymentmethod": "debit card",
    "transactionid": 570,
    "bookingid": 70
  },
```

Query

```mongo
db.payment.aggregate([
  {
    $group: {
      _id: "$paymentmethod",
      Usage: {
        $count: {}
      }
    }
  }
])
```

Result

```mongo
[
  {
    "Usage": 41,
    "_id": "credit card"
  },
  {
    "Usage": 59,
    "_id": "debit card"
  }
]
```

Query 2: Average Rating of films in each language :

Mongo Playground run share $sort docs

Configuration bson

```
db={
  "movie": [
    {
      "idmovie": 0,
      "title": "alice-led-the-way-and-the-ma
      "language": "hindi",
      "genre": "Drama",
      "duration": 121,
      "Certificate": "U/A",
      "Rating": 8
    },
    {
      "idmovie": 1,
      "title": "mock-turtle-would-be-worth-t
      "language": "hindi",
      "genre": "Drama",
      "duration": 131,
      "Certificate": "U/A",
      "Rating": 5
    },
    {
      "idmovie": 2,
      "title": "on-which-seven-looked-up-and
      "language": "hindi",
      "genre": "Fantasy",
      "duration": 116,
      "Certificate": "A",
      "Rating": 7
```

Query

```
db.movie.aggregate([
  {
    $group: {
      _id: "$language",
      AvgRating: {
        $avg: "$Rating"
      }
    }
  },
  {
    $sort: {
      "AvgRating": -1
    }
  },

])
```

Result

```
[
  {
    "AvgRating": 8,
    "_id": "telugu"
  },
  {
    "AvgRating": 7.358974358974359,
    "_id": "hindi"
  },
  {
    "AvgRating": 7.138888888888889,
    "_id": "english"
  }
]
```

Query 3: Average duration of each Genre :

Mongo Playground run share $sort docs

Configuration bson

```
db={
  "movie": [
    {
      "idmovie": 0,
      "title": "alice-led-the-way-and-the-m
      "language": "hindi",
      "genre": "Drama",
      "duration": 121,
      "Certificate": "U/A",
      "Rating": 8
    },
    {
      "idmovie": 1,
      "title": "mock-turtle-would-be-worth-
      "language": "hindi",
      "genre": "Drama",
      "duration": 131,
      "Certificate": "U/A",
      "Rating": 5
    },
    {
      "idmovie": 2,
      "title": "on-which-seven-looked-up-an
      "language": "hindi",
      "genre": "Fantasy",
      "duration": 116,
      "Certificate": "A",
      "Rating": 7
    },
    {
```

Query

```
db.movie.aggregate([
  {
    $group: {
      _id: "$genre",
      AvgDuration: {
        $avg: "$duration"
      }
    }
  },
  {
    $sort: {
      "AvgDuration": -1
    }
  },
])
```

Result

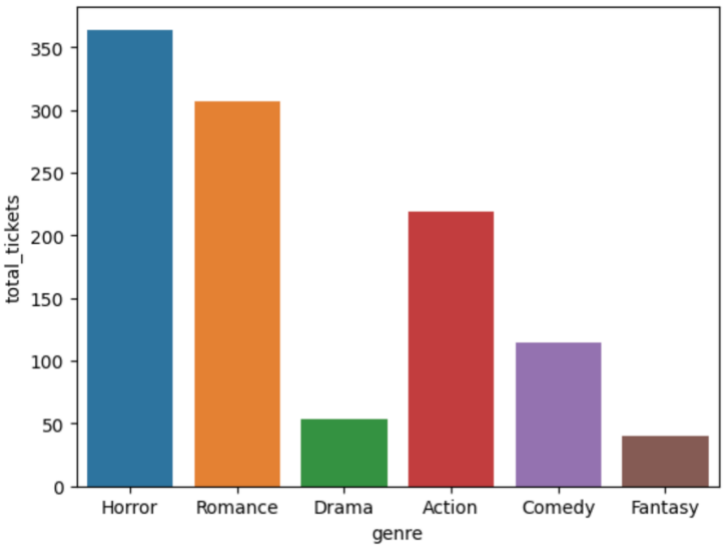
```
[
  {
    "AvgDuration": 123.2,
    "_id": "Action"
  },
  {
    "AvgDuration": 122.2,
    "_id": "Fantasy"
  },
  {
    "AvgDuration": 121.81818181818181,
    "_id": "Horror"
  },
  {
    "AvgDuration": 120.1304347826087,
    "_id": "Drama"
  },
  {
    "AvgDuration": 118.4,
    "_id": "Comedy"
  },
  {
    "AvgDuration": 118.2,
    "_id": "Romance"
  }
]
```

V. Database Access via Python

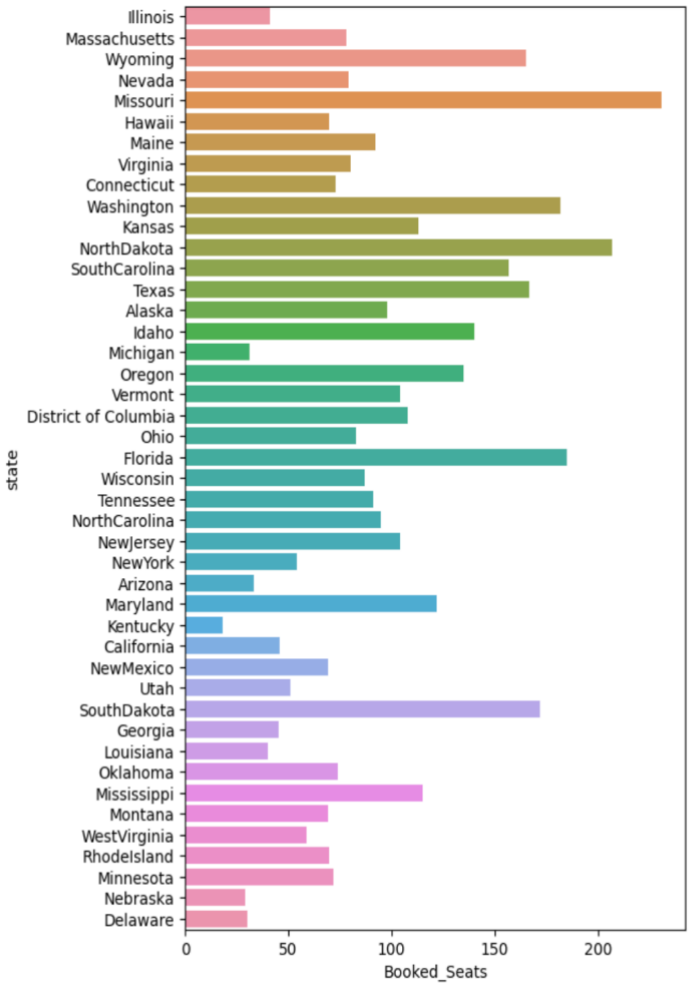
The database is accessed via Python to produce visualizations and perform data analysis. The connection to MySQL database is made using
conn=pymysql.connect(host='localhost',port=int(3306),user='root',passwd='#####',db='ticketly')
Reading Sql query and storing it in DataFrame :
df = a.read_sql_query('MySQL Query',conn)

Few Noteworthy visualizations are shown below:

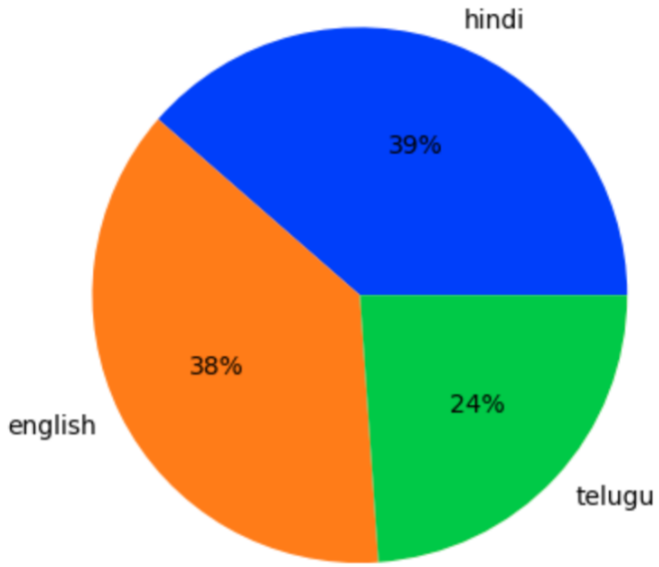
Most popular movie genres



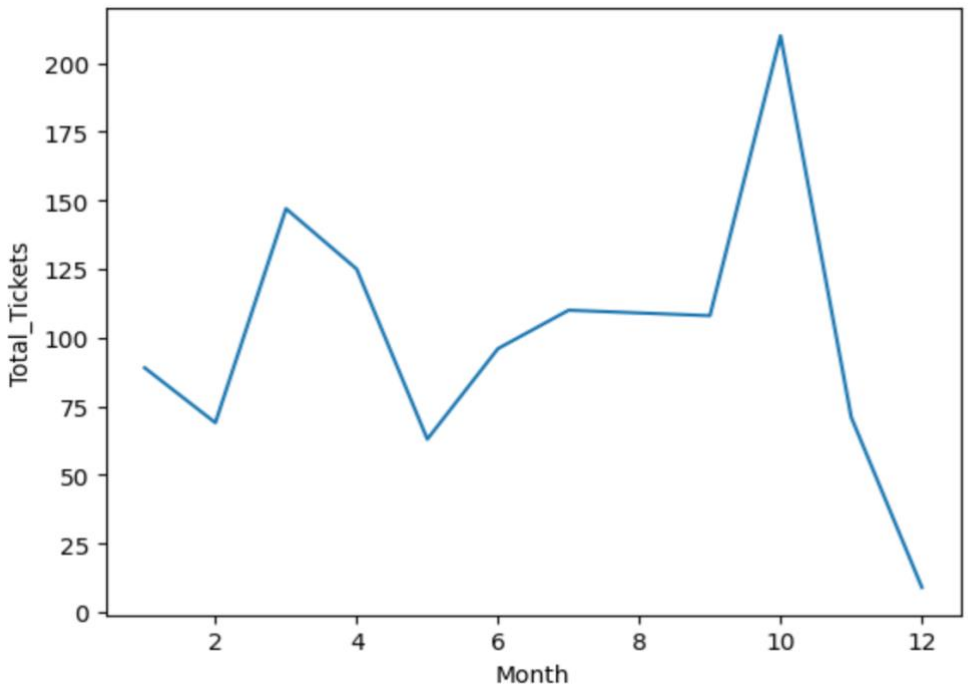
Ticket sales in each state



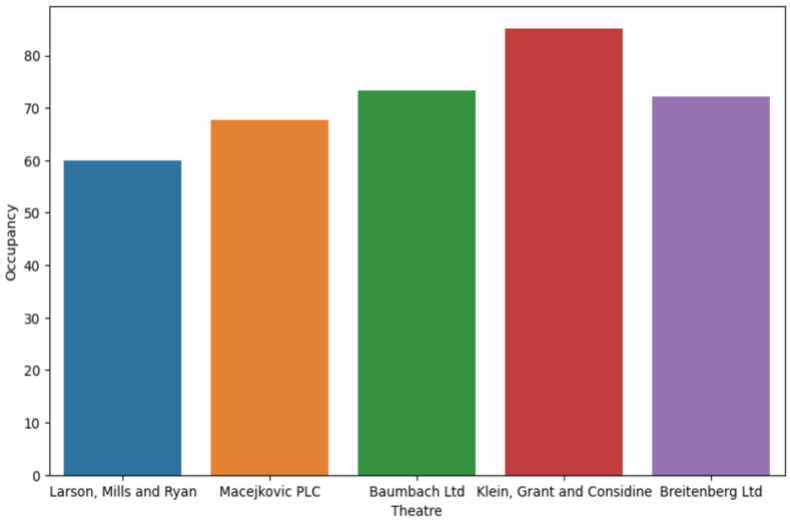
Percentage of Tickets sold in each language



Trend of Bookings in a year



Top 5 Theatres with most occupancy in the year 2022



VI. Summary and Recommendation

This use case demonstrated the implementation of a database in MySQL and NoSQL including querying to show the functionality of the database. The database was also accessed via python and some data visualizations were produced using SQL querying.

The Database can be further improved by enforcing it in salesforce. We can add validation rules and permission sets which can restrict access to specific users and by using dashboards and reports we can generate insights which can be later used by data scientists to make recommendations for the people.