MAVERICKS ARENA: BOOKING MANAGEMENT SYSTEM

A project report submitted to Prof. Guillaume Faddoul of
San Francisco State University
In partial fulfillment of
the requirements for
ISYS 864 Data Management for Analytics

Master of Science

In

Business Analytics

by

Kavita Kathaith

Shailesh Krishna

Sushma Srinivas

San Francisco, California

May 2020

Abstract

The project report contains the database design of a fictitious organization known as Mavericks Arena that stages indoor and outdoor shows. The shows include stage plays, dance, musicals, movies, and performances associated with local youth cricket, football, and rugby teams. The report consists of sections describing the various entities within the organization, business rules, EER diagram, and various SQL statements used to create, load, and query the database.

Table of Contents

Introduction

Mavericks Arena is a fictitious organization that falls under the business sector of the amphitheater industry. Mavericks Arena stages both indoor and outdoor shows. The shows include stage plays, dance, musicals, movies, and performances associated with local youth cricket, football, and rugby teams.

The amphitheater is a commercial center that attracts customers day in and day out. Since many simultaneous events happen at the amphitheater, an efficient booking system is required to handle the humongous amount of data related to various show bookings, their seat reservations, including printing of tickets. For the project, we wanted to model the database of such an organization that is capable of handling simultaneous transactions of show listings, theater management, customer bookings, seat reservations, and ticket printing. Hence, we came up with the concept of Mavericks Arena.

Entities

Mavericks Arena's schema consists of multiple entities. Below mentioned are the details for each of the entities.

- **Shows**: It contains the list of all the shows and their attributes such as name, description, owner, start date, and end date. A show can either be indoor or outdoor, which is denoted by its show type.
- **Theater**: It contains the listing of all the theaters in the Maverick Arena and their attributes, such as name, manager, seating capacity, and theater level. Mavericks Arena is a multi-level amphitheater; hence theaters are located either at the lower or upper level.
- **Manager**: In the manager entity, details of all the theater managers such as first and last name, contact details (email and phone), and their residential address are stored.
- **Show Schedule**: This entity captures all the show listings for a particular theater. Details like theater id, show start time, and show end time are stored.
- **Seats**: All the details about the seating capacity of the theater like seat number and row number are present within this entity. The seat type determines the price of the seat.
- **Customers**: It contains information about the customers of Mavericks Arena, customer details such as their first and last name, contact details (email and phone), and their address are stored.
- **Tickets**: An issued ticket can only belong to a single show of Mavericks Arena. In this tickets entity, details about booking id, customer id, booking status, and booking time for every ticket gets stored.
- **Booking**: This entity contains booking id and its associated information such as show id and customer id.

Business Rules

The below mentioned business rules govern Mavericks Arena:

- One and only one manager can manage a theater.
- A manager can manage zero or multiple theaters.
- A theater has at least one seat and may have multiple seats.
- A seat type can be a front, middle, or back but not all of them simultaneously.
- A show can either be indoor or outdoor, but not both of them at the same time.
- For a show booking, an individual seat assignment can be done only once.
- A theater can schedule one or multiple shows, but two shows cannot have the same timings.
- A show can be scheduled in one or more theaters within the Mavericks Arena.
- Each booking results in at least one ticket generation and can have multiple tickets associated with it.
- An issued ticket can only belong to a single customer.
- A customer in the database must be associated with a booking.

EER Diagram

The below mentioned EER diagram represents the business rules of Mavericks Arena.

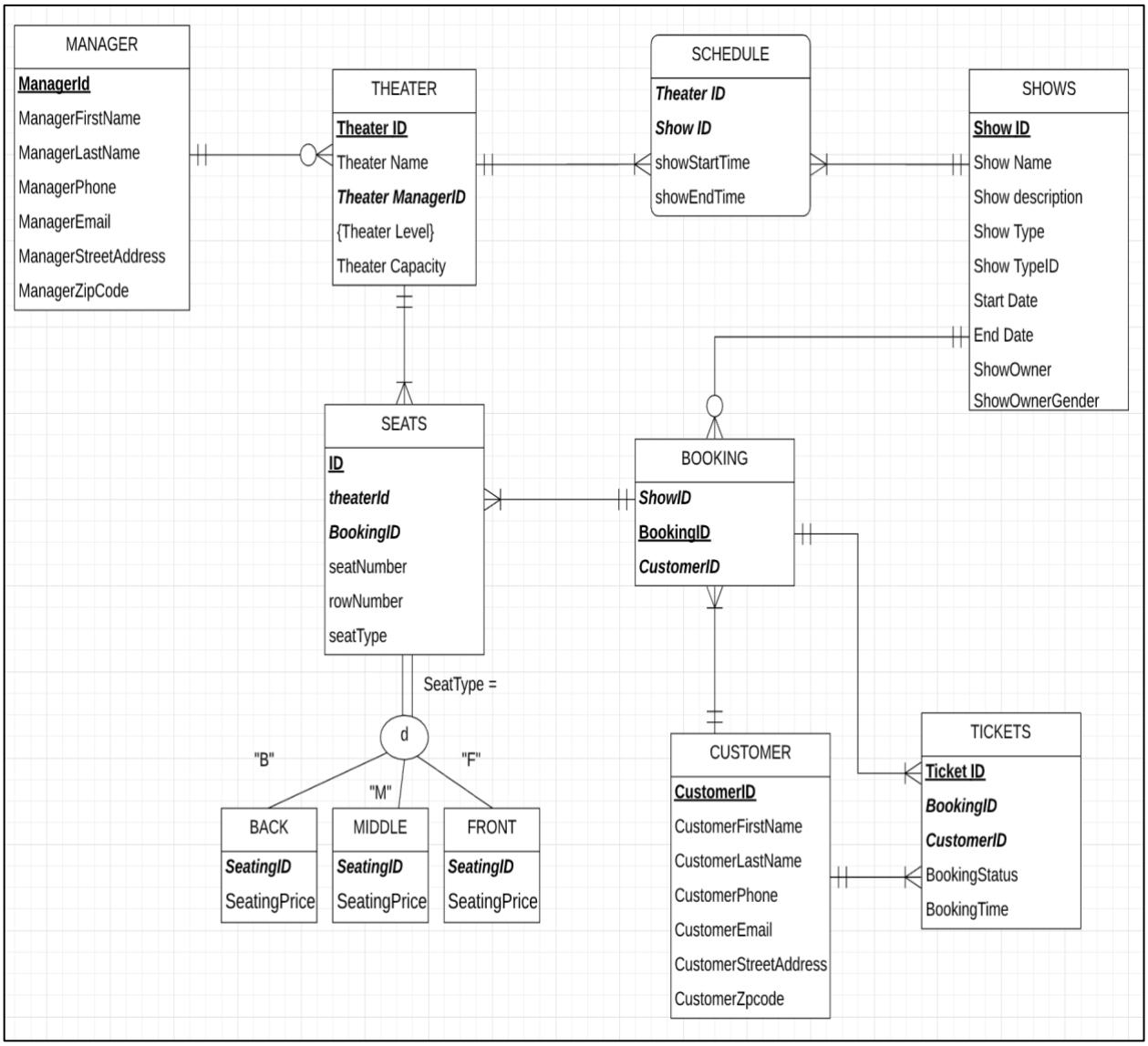

Figure 1. Mavericks Arena EER Diagram

Relational Model (3NF)

The below mentioned diagram depicts the relational model (3NF) of Mavericks Arena.

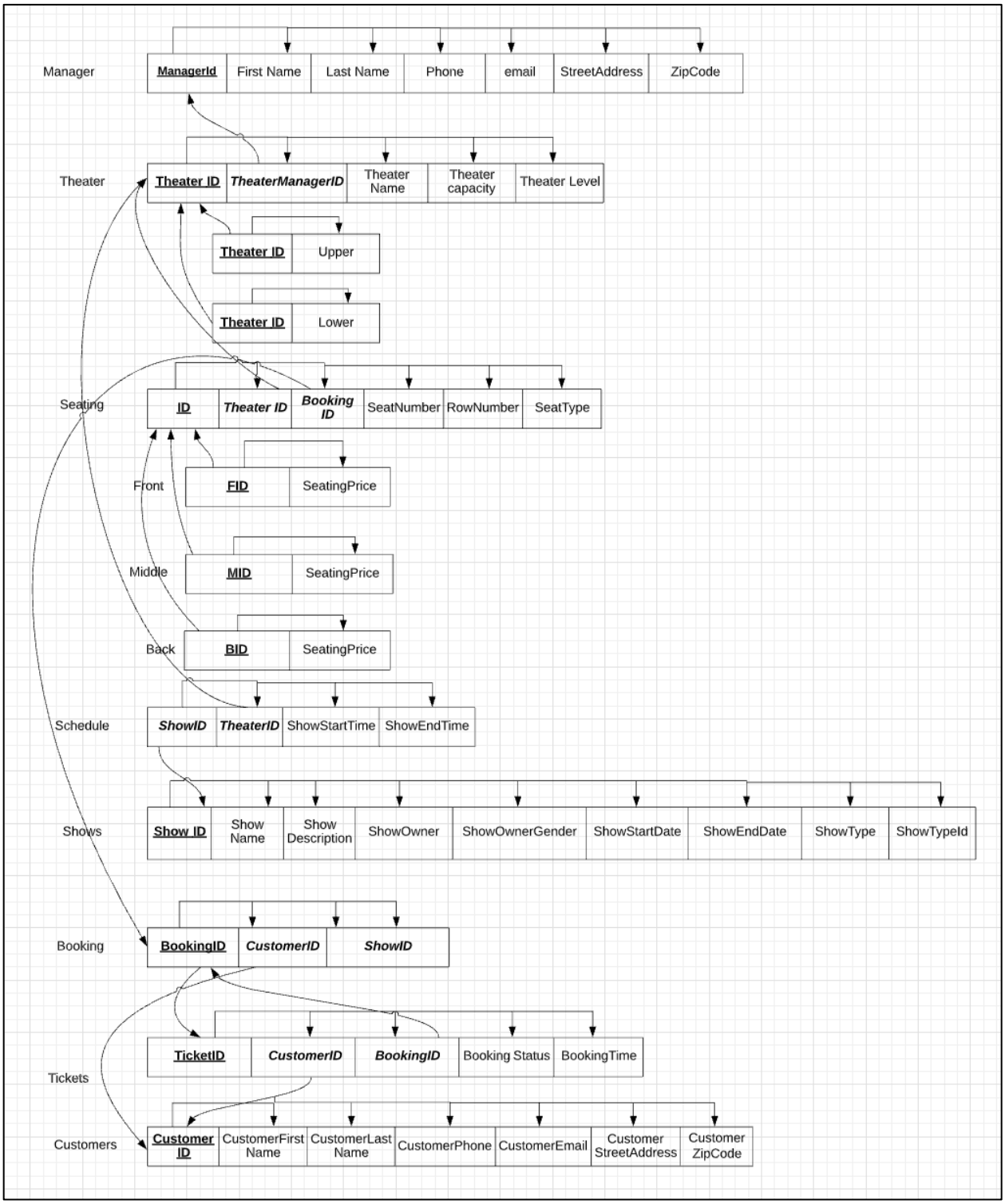

Figure 2. Mavericks Arena Relational Model

SQL Statements

A. DDL Statements

- Mavericks Arena Database

```
/* DATABASE */
-- Drop database
DROP DATABASE IF EXISTS mavericks_db;
-- Creating database mavericks_arena
CREATE DATABASE mavericks_db;
-- Using the schema created above
USE mavericks_db;
```

- Shows Table

```
-- Shows Table, contains details about the specific shows.
CREATE TABLE mavericks_db.shows_T (
showName VARCHAR(50) NOT NULL,
showDescription VARCHAR(250) NOT NULL,
showOwner VARCHAR(50) NOT NULL,
showOwnerGender CHAR(1) NOT NULL CHECK (showOwnerGender IN ('M','F','O')),
showType VARCHAR(10) NOT NULL CHECK(showType IN ('Indoor','Outdoor')),
showId INT NOT NULL,
showTypeId INT NOT NULL,
showStartDate DATE NOT NULL,
showEndDate DATE NOT NULL,
CONSTRAINT SHOWS_PK PRIMARY KEY (showId)
);
```

- Theater Table

```
-- Theater Table, contains information about the theatre.
CREATE TABLE mavericks_db.theater_T (
theaterId INT NOT NULL,
theaterName VARCHAR(25) NOT NULL,
theaterManagerId INT NOT NULL,
theaterCapacity INT NOT NULL,
theaterLevel CHAR(5) NOT NULL CHECK(theaterLevel IN ('UPPER','LOWER')),
CONSTRAINT THEATER_PK PRIMARY KEY (theaterId),
CONSTRAINT THEATER_MANAGER_FK FOREIGN KEY (theaterManagerId) REFERENCES mavericks_db.manager_T(managerId)
);
```

- Theater Manager Table

```
-- Theater manager table
CREATE TABLE mavericks_db.manager_T(
managerId INT NOT NULL AUTO_INCREMENT,
managerFirstName VARCHAR(25) NOT NULL,
managerLastName VARCHAR(25) NOT NULL,
managerPhone VARCHAR(10) NOT NULL,
managerEmail VARCHAR(50) NOT NULL,
managerStreetAddress VARCHAR(100) NOT NULL,
managerZipCode CHAR(5) NOT NULL,
CONSTRAINT manager_PK PRIMARY KEY (managerId)
);
```

- Show Schedule Table

```
-- Show Timings Table, contains information about the various shows and events that happen in amphitheater.
CREATE TABLE mavericks_db.show_schedule_T (
 showId INT NOT NULL,
 theaterId INT NOT NULL,
 showStartTime datetime NOT NULL,
 showEndTime DATETIME NOT NULL,
 CONSTRAINT SHOW_TIMING_PK PRIMARY KEY (showId),
 CONSTRAINT show_schedule_SHOW_ID_FK FOREIGN KEY (showId) REFERENCES mavericks_db.shows_T(showId),
 CONSTRAINT show_schedule_THEATER_ID_FK FOREIGN KEY (theaterId) REFERENCES mavericks_db.theater_T(theaterId),
 UNIQUE INDEX SHOW_TIMING_UNIQUE_IDX (showId,theaterId,showStartTime)
);
```

- Seats Table

```
-- seats Table, contains information about seats in a theater.
CREATE TABLE mavericks_db.seats_T(
id INT NOT NULL AUTO_INCREMENT,
theaterId INT NOT NULL,
bookingId INT NOT NULL,
seatNumber INT NOT NULL,
rowNumber CHAR(1) NOT NULL,
CONSTRAINT seats_PK PRIMARY KEY(id),
CONSTRAINT seats_THEATER_FK FOREIGN KEY (theaterId) REFERENCES mavericks_db.theater_T(theaterId),
CONSTRAINT seats_TYPE_FK FOREIGN KEY (bookingId) REFERENCES mavericks_db.booking_T(bookingId),
UNIQUE INDEX seats_UNIQUE_IDX (theaterId,seatNumber,rowNumber)
);
```

- Customers Table

```
-- Customer Table, contains information about the customer.
CREATE TABLE mavericks_db.customers_T(
customerId INT NOT NULL AUTO_INCREMENT,
customerFirstName VARCHAR(25) NOT NULL,
customerLastName VARCHAR(25) NOT NULL,
customerPhone VARCHAR(10) NOT NULL,
customerEmail VARCHAR(100) NOT NULL,
customerStreetAddress VARCHAR(100) NOT NULL,
customerZipCode CHAR(5) NOT NULL,
CONSTRAINT CUSTOMER_PK PRIMARY KEY (customerId)
);
```

- Tickets Table

```
-- Tickets Table, contains information about ticket bookings.
CREATE TABLE mavericks_db.tickets_T(
ticketId INT NOT NULL AUTO_INCREMENT,
bookingId INT NOT NULL,
CustomerdId INT NOT NULL,
BookingStatus BOOLEAN NOT NULL,
BookingTime DATETIME NOT NULL,
CONSTRAINT tickets_PK PRIMARY KEY (ticketId),
CONSTRAINT BOOKINGS_FK FOREIGN KEY (bookingId) REFERENCES mavericks_db.booking_T(bookingId),
CONSTRAINT BOOKINGS_CUSTOMER_FK FOREIGN KEY (CustomerdId) REFERENCES mavericks_db.customers_T(customerId)
);
```

- Booking Table

```
CREATE TABLE mavericks_db.booking_T(
-- seatBookingId INT NOT NULL AUTO_INCREMENT,
showId INT NOT NULL,
bookingId INT NOT NULL,
CustomerId INT NOT NULL,
-- CONSTRAINT seats_RESERVATION_PK PRIMARY KEY (seatBookingId),
CONSTRAINT booking_PK PRIMARY KEY (bookingId),
CONSTRAINT seats_BOOKING_FK FOREIGN KEY (showId) REFERENCES mavericks_db.shows_T(showId),
CONSTRAINT seats_SHOWTIME_FK FOREIGN KEY (CustomerId) REFERENCES mavericks_db.customers_T(CustomerId),
UNIQUE INDEX SEAT_RESERVATION_IDX (showId,bookingId,CustomerId)
);
```

B. DML Statements

- Theater Manager Table

```
-- manager_T
INSERT INTO `manager_T` (`managerId`,`managerFirstName`,`managerLastName`,`managerPhone`,`managerEmail`,`managerStreetAddress`,`managerZipCode`)
VALUES (1,"Keegan","Eaton","4232734658","tincidunt.nunc.ac@elitpharetra.org","2346 Mauris Street","93862");
INSERT INTO `manager_T` (`managerId`,`managerFirstName`,`managerLastName`,`managerPhone`,`managerEmail`,`managerStreetAddress`,`managerZipCode`)
VALUES (2,"Kerry","Ward","9617251526","pellentesque@nisl.net","Ap #883-3975 Id, Ave","57845");
INSERT INTO `manager_T` (`managerId`,`managerFirstName`,`managerLastName`,`managerPhone`,`managerEmail`,`managerStreetAddress`,`managerZipCode`)
VALUES (3,"Genevieve","Summers","1982726819","vestibulum.lorem@dui.ca","6976 Sed St.","56391");
```

- Shows Table

```
-- shows_T
INSERT INTO `shows_T` (`showId`,`showName`,`showDescription`,`showOwner`,`showOwnerGender`,`showType`,`showTypeId`,`showStartDate`,`showEndDate`)
VALUES (1,"Les Miserables","Integer urna. Vivamus molestie dapibus","Aidan Stark","F","Indoor",1,"2020-05-09","2021-05-08");
INSERT INTO `shows_T` (`showId`,`showName`,`showDescription`,`showOwner`,`showOwnerGender`,`showType`,`showTypeId`,`showStartDate`,`showEndDate`)
VALUES (2,"Les Miserables","commodo at, libero. Morbi accumsan","Nora Gould","F","Indoor",1,"2020-04-04","2021-03-11");
INSERT INTO `shows_T` (`showId`,`showName`,`showDescription`,`showOwner`,`showOwnerGender`,`showType`,`showTypeId`,`showStartDate`,`showEndDate`)
VALUES (3,"Matilda","nec orci. Donec nibh. Quisque","Quon Christensen","F","Indoor",1,"2020-01-16","2021-02-17");
```

- Theater Table

```
-- theater_t
INSERT INTO `theater_T` (`theaterId`,`theaterName`,`theaterManagerId`,`theaterCapacity`,`theaterLevel`) VALUES (1,"A",3,54,"UPPER");
INSERT INTO `theater_T` (`theaterId`,`theaterName`,`theaterManagerId`,`theaterCapacity`,`theaterLevel`) VALUES (2,"B",2,84,"UPPER");
INSERT INTO `theater_T` (`theaterId`,`theaterName`,`theaterManagerId`,`theaterCapacity`,`theaterLevel`) VALUES (3,"C",2,91,"UPPER");
```

- Show Schedule Table

```
-- show_schedule_T
INSERT INTO `show_schedule_T` (`showId`,`theaterId`,`showStartTime`,`showEndTime`) VALUES (1,3,"2020-05-18 16:05:18","2019-09-29 15:09:24");
INSERT INTO `show_schedule_T` (`showId`,`theaterId`,`showStartTime`,`showEndTime`) VALUES (2,7,"2020-12-30 20:12:57","2021-03-03 11:03:08");
INSERT INTO `show_schedule_T` (`showId`,`theaterId`,`showStartTime`,`showEndTime`) VALUES (3,4,"2019-09-08 23:09:26","2020-10-06 04:10:43");
```

- Customers Table

```
-- customers_T
INSERT INTO `customers_T` (`customerId`,`customerFirstName`,`customerLastName`,`customerPhone`,`customerEmail`,`customerStreetAddress`,`customerZipCode`)
VALUES (1,"Cora","Roberts","4231916234","ac@seddictumeleifend.edu","579-1264 Dolor. Road","39353");
INSERT INTO `customers_T` (`customerId`,`customerFirstName`,`customerLastName`,`customerPhone`,`customerEmail`,`customerStreetAddress`,`customerZipCode`)
VALUES (2,"Gray","Terrell","4101453637","sodales.at.velit@Sedauctor.ca","8540 Tempus Road","36950");
INSERT INTO `customers_T` (`customerId`,`customerFirstName`,`customerLastName`,`customerPhone`,`customerEmail`,`customerStreetAddress`,`customerZipCode`)
VALUES (3,"Bruce","Fry","5773395346","Praesent@magnaSed.co.uk","4366 Id, Rd.","63113");
```

- Booking Table

```
INSERT INTO `booking_T` (`showID`,`bookingId`,`CustomerId`) VALUES (1,1,31);
INSERT INTO `booking_T` (`showID`,`bookingId`,`CustomerId`) VALUES (2,2,66);
INSERT INTO `booking_T` (`showID`,`bookingId`,`CustomerId`) VALUES (3,3,26);
```

- Seats Table

```
INSERT INTO `seats_T` (`id`,`theaterId`,`bookingId`,`seatNumber`,`rowNumber`) VALUES (1,6,98,14,"A");
INSERT INTO `seats_T` (`id`,`theaterId`,`bookingId`,`seatNumber`,`rowNumber`) VALUES (2,6,61,20,"A");
INSERT INTO `seats_T` (`id`,`theaterId`,`bookingId`,`seatNumber`,`rowNumber`) VALUES (3,3,10,10,"B");
```

- Tickets Table

```
INSERT INTO `tickets_T` (`ticketId`,`bookingId`,`CustomerdId`,`BookingStatus`,`BookingTime`) VALUES (1,81,67,"1","2020-04-04 07:04:26");
INSERT INTO `tickets_T` (`ticketId`,`bookingId`,`CustomerdId`,`BookingStatus`,`BookingTime`) VALUES (2,56,41,"0","2020-01-03 23:01:37");
INSERT INTO `tickets_T` (`ticketId`,`bookingId`,`CustomerdId`,`BookingStatus`,`BookingTime`) VALUES (3,39,49,"0","2020-01-15 07:01:05");
```

C. Information Retrieval Statements

- **Question**: What is the gender diversity for the content created?

```
SELECT Female_Content_COUNT/(Female_Content_COUNT+Male_Content_COUNT+Others_Content_COUNT) AS FEMALE_CONTENT_RATIO,
       Male_Content_COUNT/(Female_Content_COUNT+Male_Content_COUNT+Others_Content_COUNT) AS MALE_CONTENT_RATIO,
       Others_Content_COUNT/(Female_Content_COUNT+Male_Content_COUNT+Others_Content_COUNT) AS OTHERS_CONTENT_RATIO
FROM
    (SELECT
        SUM(CASE
            WHEN showOwnerGender = "F" THEN 1 ELSE 0
            END) AS Female_Content_COUNT,
        SUM(CASE
            WHEN showOwnerGender = "M" THEN 1 ELSE 0
            END) AS Male_Content_COUNT,
        SUM(CASE
        WHEN showOwnerGender = "O" THEN 1 ELSE 0
        END) AS Others_Content_COUNT
        FROM shows_T) AS Content_Count;
```

Answer:

FEMALE_CONTENT_RATIO	MALE_CONTENT_RATIO	OTHERS_CONTENT_RATIO
0.4000	0.4000	0.2000

- ***Question***: Determine the manager's full name for every theater.

```
SELECT theaterId,
CONCAT(managerFirstName ," ",managerLastName) AS Manager_Name
FROM  manager_T
INNER JOIN theater_T
on manager_T.managerId = theater_T.theaterManagerId
ORDER BY theaterId ASC;
```

Answer:

theaterId	Manager_Name
1	Genevieve Summers
2	Kerry Ward
3	Kerry Ward
4	Keegan Eaton
5	Keegan Eaton
6	Genevieve Summers
7	Robert Kaufman
8	Robert Kaufman
9	Keegan Eaton
10	Robert Kaufman
11	Chaney Fuentes

- ***Question***: What is the top most booked shows of all times based on Total Tickets Booked. If there is a tie, display both the show names.

```
SELECT ShowName,Tickets_Booked
    FROM
    (SELECT distinct shows.ShowName,
                    Count(distinct tickets.ticketId) as Tickets_Booked
                    FROM booking_T booking
                    INNER JOIN shows_T shows
                    ON booking.showId = shows.showId
                    INNER JOIN tickets_T tickets
                    ON booking.bookingId = tickets.bookingId
                    AND tickets.BookingStatus = 1
                    GROUP BY ShowName
                    ORDER BY Count(distinct tickets.ticketId) DESC) AS Top_Shows_T
WHERE Tickets_Booked = (SELECT MAX(Tickets_Booked)
                    FROM
                    (SELECT distinct shows.ShowName,
                    Count(distinct tickets.ticketId) as Tickets_Booked
                    FROM booking_T booking
                    INNER JOIN shows_T shows
                    ON booking.showId = shows.showId
                    INNER JOIN tickets_T tickets
                    ON booking.bookingId = tickets.bookingId
                    AND tickets.BookingStatus = 1
                    GROUP BY ShowName
                    ORDER BY Count(distinct tickets.ticketId) DESC) AS Top_Shows_T);
```

Answer:

ShowName	Tickets_Booked
Fame	9
West Side Story	9

- ***Question***: Which show type is most preferred by people based on tickets booked?

```
SELECT showType AS 'Preferred_Show_Type'
FROM
      (SELECT showType, COUNT(tickets.ticketId) as ticket_count
                        FROM booking_T booking
                        INNER JOIN shows_T shows
                        ON booking.showId = shows.showId
                        INNER JOIN tickets_T tickets
                        ON booking.bookingId = tickets.bookingId
                        AND tickets.BookingStatus = 1
                        GROUP BY showType) AS type_count
ORDER BY ticket_count DESC
LIMIT 1;
```

Answer:

Preferred_Show_Type
Indoor

- ***Question***: What are the top 2 highest grossing show along with box office collection of all times?

```
SELECT DISTINCT shows.showName, ROUND(SUM(seat_types.seatsPrice),2) AS 'Box_Office_Collection'
FROM  booking_T bookings
INNER JOIN tickets_T tickets
ON bookings.bookingId = tickets.bookingId
AND tickets.BookingStatus = 1
INNER JOIN shows_T shows
ON bookings.showId = shows.showId
INNER JOIN seats_T seats
ON bookings.bookingId = seats.bookingId
INNER JOIN seats_type_T seat_types
ON seats.id = seat_types.id
GROUP BY shows.showName
ORDER BY ROUND(SUM(seat_types.seatsPrice),2) DESC
LIMIT 2;
```

Answer:

showName	Box_Office_Collection
Burlesque	2247.70
West Side Story	1170.88

- ***Question***: Determine the customers who are the first to attend the shows.

```
SELECT DISTINCT shows.showName,shows.showStartDate,
        FIRST_VALUE(concat(customers.customerFirstName," ",customers.customerLastName))
        OVER(PARTITION BY shows.showName ORDER BY show_schedule.showStartTime ASC) AS Customer_Name
FROM shows_T shows
INNER JOIN show_schedule_T show_schedule
ON shows.showId = show_schedule.showId
INNER JOIN booking_T bookings
ON shows.showId = bookings.showId
INNER JOIN customers_T customers
ON bookings.CustomerId = customers.CustomerId
INNER JOIN tickets_T tickets
ON customers.CustomerId = tickets.CustomerdId
AND tickets.BookingStatus = 1
ORDER BY shows.showStartDate ASC;
```

Answer:

showName	showStartDate	Customer_Name
Matilda	2020-01-16	Nehru Rocha
Football League Inauguration	2020-02-18	Daniel Peck
West Side Story	2020-02-21	Cora Roberts
Les Miserables	2020-05-09	Jarrod Vincent
Football League Inauguration	2020-05-10	Daniel Peck

Conclusion

In conclusion, the project gave us an overall understanding of how we can model an organization's entities and create relations among them. Since we had to build our model from scratch, we understood each step of the process well. We gained practical experience in building ER Diagrams and 3 NF relational models. On the coding front, we learned how to create our

tables based on our model and also load data into each table. We also learned how to query a database.

Through the process, there were a few hurdles. The first challenging part of the project was to build the EER diagram for an imaginary organization from scratch. The identification of entities and attributes took some time. The next challenge was to generate and add dummy data to each column in the table in the appropriate format. A mistake in one column would result in an error in a linked column of another table. Any changes would also impact our queries. After several updates, we were able to understand the structure better.

Through the process of exploration and constant learning, we successfully achieved the desired results. The project was a great value add to our learning of the course.

References

[1] Hoffer, Ramesh, Topi, “Modern Database Management Systems”