ECE3076 Database Systems

2021/2022 Trimester 2

Scenario: Online Cinema Booking System

Members

Student Name	Student ID	Major
Abdulla Al Mahin Khan	1191302844	CE
Md. Istiaq Ahmed Bhuiyan	1181302904	CE
Haziq Afham Bin Mohd Nasir	1191303285	CE

ECE3076 Database System Group Assignment Project Assessment Rubrics

Phase	**Total Marks**
Phase 1	30
Phase 2	10
Phase 3	10
Assessments Rubriks	
Phase 1	
Criteria	**Total Marks**
Introduction	5
Business Rules	5
Entities and Attributes (Data dictionary)	5
Entity relationship diagram	5
Entity integrity and Referential Integrity	5
Normalization	5
Subtotal	30
Phase 2	
Quantity of records (minimum 20 records for each table)	5
SQL scripts for database implementation	5
Subtotal	10
Phase 3	
SQL Queries Answered. Queries will be given to the students 2 weeks before the submission deadline.	5
Presentation on database design and implementation	5
	10
Grand total	**50**

Phase 1

1. Introduction to scenario

1.1. Scenario background

Over the past centuries there had been changing in the social, economic, and even recreational aspects of the world but when the 21st century or also known as Industrial Age came in, the changes became more common because of the development of technology. Almost every day we are using internet and day by day we are getting used to complete our job in online. However, the impact of technological advancement goes into recreational sector as well and movie theatre is one of source of 21st century's people's amusement. In the prospect of Malaysia there are about 120 movie theatres and with the rapid technological development in Malaysia, the number of movie theatres are increasing year by year.

In today's movie theatre not only provide cinema show but also it provides comfort and convenience in terms of booking tickets, movie show time, availability of tickets. Moreover, it provides quick service of snacks and light cuisine for the people.

Our objective is to design a Movie Ticketing System Database. Movie Ticketing System Database is basically aimed to provide complete information of the movie and schedule to the audience, according to which he/she can easily get a ticket instantly and can book a ticket on his/her favourite movies. Admin can use Movie Ticketing System to insert and delete data such as movie descriptions, movie schedules which will update the related webpage and will be accessible by the audiences. Admin can update the webpage changing according to the data in the database also admin can check the statistics information from the system and operates cinema with better efficiency by automating reservation. Moreover, ticketing process improves profitability and manages cinema better by having access to key data in a centralized and systematic view and increase audience satisfaction by giving your audiences what they want when it comes to the seat preference.

Besides, All the audiences' transactions will be recorded accordingly through our database system. It is to minimize errors in the process of creating, collecting, and storing information into the database system. As a result, it will help us to improve the Movie theatre performance and reliability.

1.2. Business rule

1. In order to book a movie, the desired film is to be selected by the audience from a wide selection of movies. So, from this wide selection of movies only one can be chosen per booking.
2. There are one or many cinemahalls in a branch where the movie is premiered but an audience can choose only one branch to watch the movie per booking.
3. A movie is premiered at one or more cinema halls. But every cinema hall plays one movie at a time.
4. An audience can make one or many bookings for tickets. At least one or more bookings are made by one audience.
5. A movie can have multiple showings, but the showings refer to only one movie.
6. A cinema hall holds a limited seat capacity of 15 seats per hall.
7. An audience can book 1 or more seats. So, 1 seat is booked at a time in the system under the same audience.
8. A seat has a specific seat number and row number as well.
9. Seat status will be displayed if it's available or booked while booking one or many seats.
10. A booking can be made for a particular timetable at a time.
11. A payment is made for only one booking. An audience can make one payment per booking.
12. Audience can choose payment methods between online banking payment or Credi/Debit card payment.
13. An online receipt will be provided to the audience for each booking. One receipt can be made for one payment.

1.3. Database objectives

Our group planned to create such a database system where the admin(cinema_staff) can have all the details of the end user (audience) who in turn will be making one or multiple bookings for different movie shows in a particular cinema hall. Our objectives are stated below -

1. The main purpose of this database is to make movie ticket booking procedures for an audience efficient and smooth.
2. And admin can track orders and receipts and guide the audiences and help them to their respective halls as well as seats.
3. Encapsulates the ordering and payment process.

2. Scenario overview

2.1 Entities and attribute

Branch								
Keys	Field	Description	DataType	Size	Constraints	Format	Range	FK Reference Table
PK	branch_id	Unique ID	varchar	8	Not null	BRxxx	BR000-BR9999	
	branch_name	Name of the branch	varchar	255	Not null	xxxxx	OAA000-OZZ999	
	branch_loc	Name of city where branch is located	varchar	255	Not null	xxxxx	OAA000-OZZ999	
	branch_state	Name of state where branch is located	varchar	255	Not null	xxxxx	OAA000-OZZ999	
	branch_mang	The manager in charge of the branch	varchar	255	Not null	xxxxx	AAA-ZZZ	

Cinema Hall								
Keys	Field	Description	Data Type	Size	Constraints	Format	Range	FK Reference Table
PK	cin_h_id	Unique ID	varchar	8	Not null	CHxxx	CH0000-CH9999	
	cin_h_num	cinema hall number	int	8	Not null	xxxxx	000000-999999	
FK1	branch_id	Unique ID	varchar	8	Not null	BRxxx	BR000-BR9999	
	cin_cap	Capacity limit of cinema hall	int	8	Not null	xxxxx	000000-999999	

Movies								
Keys	Field	Description	Data Type	Size	Constraints	Format	Range	FK Reference Table
PK	movie_id	Unique ID	varchar	6	Not null	MVxxx	MV01 - MV20	
	movie_name	Name of the movie	char	255	Not null	xxxxx	Title of the film	
	IMDb_rat	Ratings by IMDb out of 10	float	(2,2)	Not null	xxxxx	0.0-10.0	
	release_date	Date of the movies' release	date		Not null	YYYY-MM-DD		
	directed_by	Name of the Director	varchar	255	Not null			

Movie Timetable								
Keys	Field	Description	Data Type	Size	Constraints	Format	Range	FK Reference Table
PK	movie_time_id	Unique ID	int	8	Not null	MT-xxx	MT000-MT999	
	movie_time	Timetables for movie	time		Not null	hh:mm:ss.	-838:59:59' to '838:59:59	
	movie_date	Premiere dates for movies	date		Not null	YYYY-MM-DD		
FK1	movie_id	Unique ID	int	8	Not null	MV-xxx	MV0000-MV9999	Movies

Seat details								
Keys	Field	Description	Data Type	Size	Constraints	Format	Range	FK Reference Table
PK	seat_id	Unique ID	char	8	Not null	Axxx	A101-E315	
	seat_num	Number assigned to a seat	char	8	Not null	Ax	A1-E5	
	seat_row	Row where the seat is located	char	8	Not null	A	A-E	
FK1	cin_h_id	Unique ID	varchar	8	Not null	CHxx	CH11-CH99	CinemaHalls
	seat_price	Price of seat to be seated	float	(3,2)	Not null	xx	00000-99999	

Seat Availability Status								
Keys	Field	Description	DataType	Size	Constraints	Format	Range	FK Reference Table
PK	seat_status_id	Unique ID	varchar	8	Not null	xxxx	S000-S999	
FK1	seat_id	Unique ID	varchar	8	Not null	Sxxx		
	seat_status_des	Availability description of the seat	varchar	25	Not null	xxxxx xxxxx - xxxxx xx	Available/Unavailable	

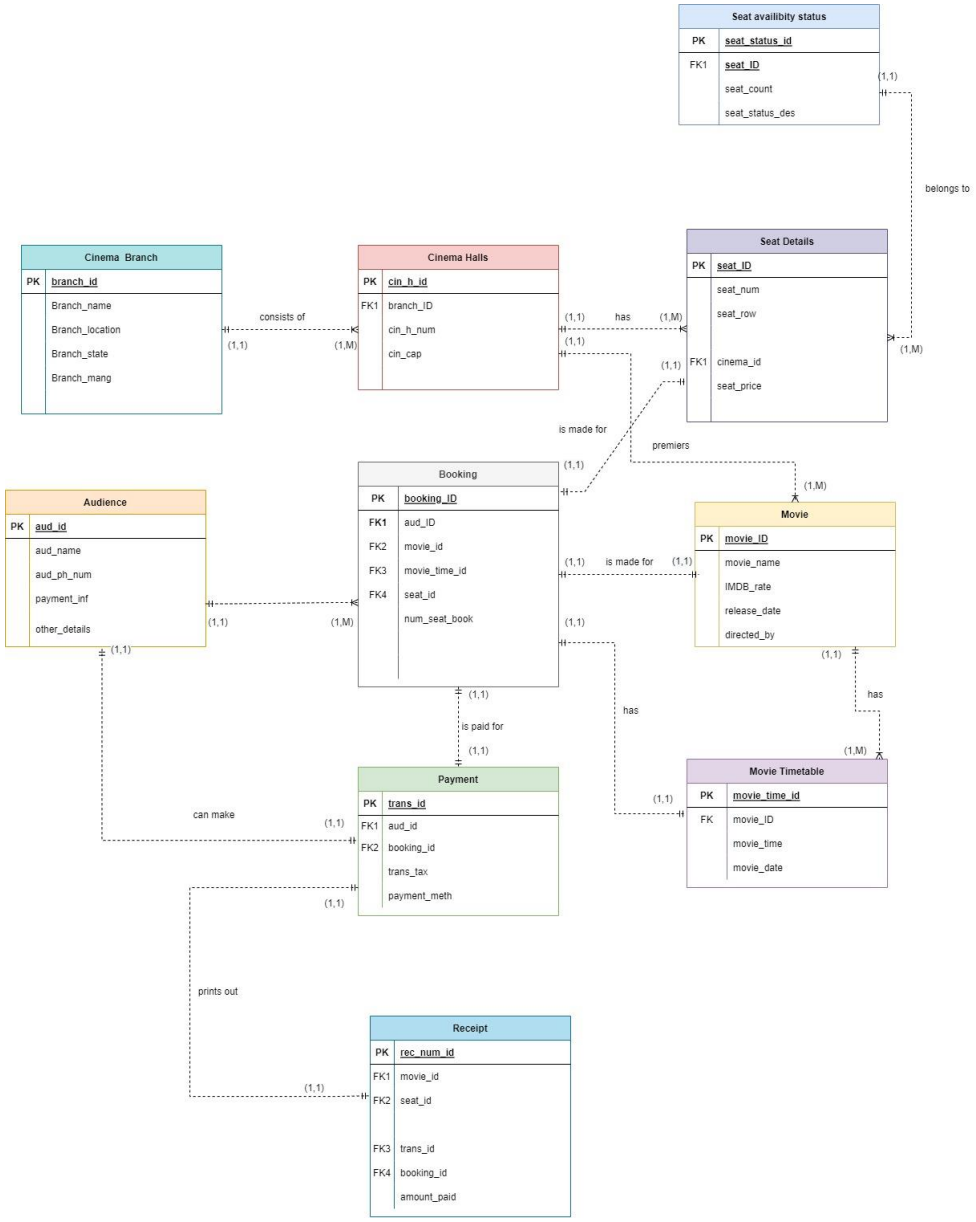
Audience								
Keys	Field	Description	DataType	Size	Constraints	Format	Range	FK Reference Table
PK	aud_id	Unique ID	varchar	8	Not null	AU-xxx	AU000-AU999	
	aud_name	Name of the audience(s)	varchar	255	Not null	xxxxx	A-Z	
	aud_ph_num	Contact details of audience	varchar	255		xxxxxxxxx	0000000000 - 9999999999	
	payment_inf	Card or cash payment	char	255	Not null	xxxx/xxxxxxxxxxx	A-Z	
	other_details		varchar	255		xxxxx		

Bookings								
Keys	Field	Description	Data Type	Size	Constraints	Format	Range	FK Reference Table
PK	booking_id	Unique ID	varchar	8	Not null	BID-xxx	BID000-BID999	
FK1	aud_id	Unique ID	varchar	8	Not null	AU-xxx	AU000-AU999	Audience
FK2	movie_id	Unique ID	varchar	8	Not null	MVxx	MV00-MV99	Movies
FK3	movie_time_id	Unique ID	varchar	8	Not null	MTxx	MT00-MT99	Movie_timetable
FK4	seat_id	Unique ID	int	8	Not null	Si-xxx	Si000-Si9999	Seat_details
	num_seat_book	Seat counts of which bookings are made	int	8	Not null	xxx	00-99	

Payment								
Keys	Field	Description	Data Type	Size	Constraints	Format	Range	FK Reference Table
PK	Trans_id	Unique ID	int	8	Not null	TRX-xxx	TRX0000-TRX9999	
FK1	aud_id	Unique ID	int	8	Not null	AU-xxx	AU000-AU999	
FK2	booking_id	Unique ID	int	8	Not null	Bid-xxxx	Bid0000-Bid9999	
	Trans_tax	Price after Tax	float	(6,2)	Not null	xxxx	0000-9999	
	payment_meth	Audience's payment method	char	15	Not null	xxxxx	A000-Z000	

Receipt Provided to Audiences and Saved in Cashier System								
Keys	Field	Description	Data Type	Size	Constraints	Format	Range	FK Reference Table
PK	rec_num_id	Unique ID	int	8	Not null	Rxxx	R000-R999	
FK1	movie_id	Unique ID	int	8	Not null	MV-xxx	MV000-MV999	
FK2	cinema_id	Unique ID	int	8	Not null	CID-xxx	CID0000-CID9999	
FK3	seat_id	Unique ID	int	8	Not null	Si-xxx	Si000-Si9999	
FK4	booking_id	Unique ID	int	8	Not null	Bid-xxxx	Bid0000-Bid9999	
FK5	trans_id	Unique ID	int	8	Not null	TRX-xxx	TRX0000-TRX9999	
	amount_paid	Total amount paid for booking	float	(8,2)	Not null	xxxx	0000-9999	

2.2. ER Diagram



2.3 Entity and referential integrity

The Primary Key value of every entry contains an “ID” value. ID values cannot be null or be a duplicate of another ID value. This feature is repeated in all tables, ensuring that every entry in every table can be uniquely identified by their Primary Key value. Thus Entity Integrity is achieved. Referential Integrity is maintained by ensuring that the Foreign Key value of every entry matches to a Primary Key value in a related table. So every foreign key value must reference a primary key value in a related table or be null. The following section will explain how Entity and Referential integrity is achieved in each entity:

1. **Movies**

Every Movies in the database will have a “movie_id” value. A “movie_id” cannot be null or be a duplicate of another “movie_id” value. Thus “movie_id is chosen as the Primary Key of the Movies entity and will allow each movie to be uniquely identifiable.

2. **Branch**

Every Branch in the database will have a “branch_id” value. A “branch_id” cannot be null or be a duplicate of another “branch_id” value. Thus “branch_id” is chosen as the Primary Key of the Branch entity and will allow each branch to be uniquely identifiable.

3. **Cinema Hall**

Every Cinema Hall in the database will have a “cin_h_id” value. A “cin_h_id” cannot be null or be a duplicate of another “cin_h_id” value. Thus “cin_h_id” is chosen as the Primary Key of the Cinema Hall entity and will allow each cinema hall to be uniquely identifiable.

Cinema Hall has one Foreign Key, “branch_id”. The value of this field will be unique as it is a Primary Key field in their respective tables. So, this foreign key value in Cinema Hall will match to a Primary Key value in another related table.

4. **Audience**

Every Audience in the database will have a “aud_id” value. A “aud_id” cannot be null or be a duplicate of another “aud_id” value. Thus “aud_id” is chosen as the Primary Key of the Audience entity and will allow each audience to be uniquely identifiable.

5. **Movie Timetable**

Every Movie Showing details in the database will have a “movie_id” value. A “movie_id” cannot be null or be a duplicate of another “movie_id” value. Thus “movie_id” is chosen as the Primary Key of the Movie Showing details entity and will allow each movie showing details to be uniquely identifiable.

Movie Showing details has two Foreign Key, “movie_id” and “cin_h_id”. The values of these fields will be unique as it is a Primary Key field in their respective tables. So, these foreign key values in Movie Showing details will match to a Primary Key value in another related table.

6. **Seat details**

Every Seat details in the database will have a “seat_id” value. A “seat_id” cannot be null or be a duplicate of another “seat_id” value. Thus “seat_id” is chosen as the Primary Key of the Movies entity and will allow each Seat details to be uniquely identifiable.

Seat details has two Foreign Key, “seat_status” and “cinema_id”. The values of these fields will be unique as it is a Primary Key field in their respective tables. So, these foreign key values in Seat details will match to a Primary Key value in another related table.

7. **Seat Availability status**

Every Seat Availability status in the database will have a “seat_status and “seat_id” value. A “seat_status and “seat_id” cannot be null or be a duplicate of another “seat_status and “seat_id” value. Thus “seat_status and “seat_id” is chosen as the Primary Key of the Seat availability status entity and will allow each Seat Availability status to be uniquely identifiable.

Seat Availability status has one Foreign Key, “seat_id”. The value of this field will be unique as it is a Primary Key field in their respective tables. So, this foreign key value in Seat status will match to a Primary Key value in another related table.

8. **Bookings**

Every Bookings in the database will have a “booking_id” value. A “booking_id” cannot be null or be a duplicate of another “booking_id” value. Thus “booking_id” is chosen as the Primary Key of the Bookings entity and will allow each Bookings to be uniquely identifiable.

Bookings has three Foreign Keys, “booking_id”,”aud_id” and ”seat_id”. The value of these fields will be unique as it is a Primary Key field in their respective tables. So, these foreign key values in Bookings will match to a Primary Key value in another related table.

9. **Payment**

Every Payment in the database will have a “trans_id” value. A “trans_id” cannot be null or be a duplicate of another “trans_id” value. Thus “trans_id” is chosen as the Primary Key of the Payment entity and will allow each Payment to be uniquely identifiable.

Payment has four Foreign Keys, “movie_id”,”cinema_id” and ”seat_id” and “booking_id”. The values of these fields will be unique as it is a Primary Key field in their respective tables. So, these foreign key values in Payment will match to a Primary Key value in another related table.

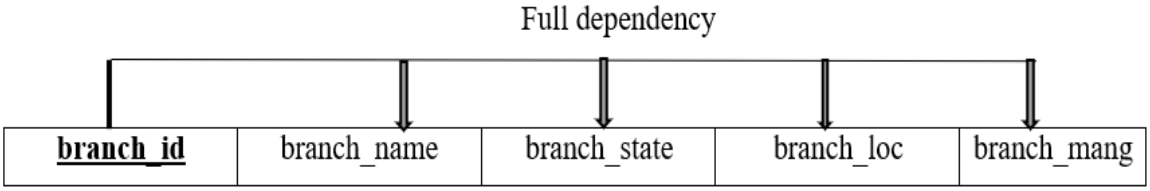
10. **Receipt**

Every Receipt in the database will have a “rec_num” value. A “rec_num” cannot be null or be a duplicate of another “rec_num” value. Thus “rec_num” is chosen as the Primary Key of the Receipt entity and will allow each Receipt to be uniquely identifiable.

Receipt has five Foreign Keys, “movie_id”,”cinema_id”, ”seat_id”, “booking_id” and “trans id”. The values of these fields will be unique as it is a Primary Key field in their respective tables. So, these foreign key values in Payment will match to a Primary Key value in another related table.

3. Normalisation:

A. Normalisation For Branch Table:



[1NF]:

Primary key for Entity ‘Branch’ is defined by ‘branch_id ’ . By ‘branch_id ’attribute each tuple can be uniquely identified and also no repeating records will be found.

[2NF]:

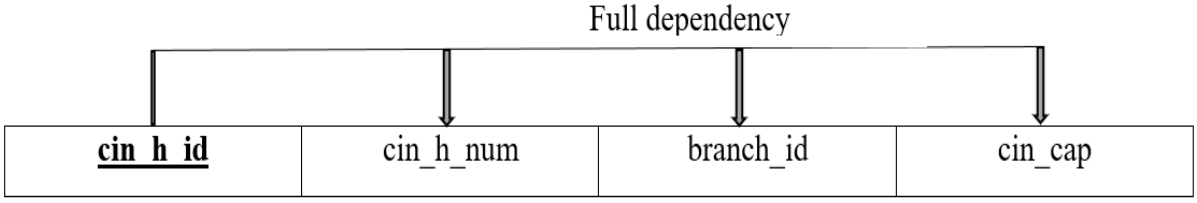
The entity’ Branch’ has no partial dependencies within.

[3NF]:

The entity has no transitive dependencies. All attributes are functionally dependent on the primary key.

Branch (**branch_id** , branch_name , branch_state, branch_loc , branch_mang)

B. Normalisation for Cinema Hall Table:



[1NF]:

Primary key for Entity ‘Cinema Hall’ is defined by ‘cin_h_id ’ . By ‘cin_h_id ’attribute each tuple can be uniquely identified and also no repeating records will be found.

[2NF]:

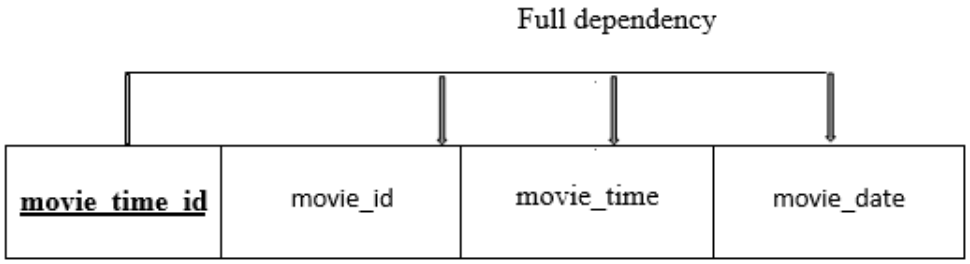
The entity’ Cinema Hall’ has no partial dependencies within.

[3NF]:

The entity has no transitive dependencies. All attributes are functionally dependent on the primary key.

Cinema Hall (**cin_h_id**, cin_h_num, branch_id, cin_cap)

C. Normalisation for Movie-Timetable table:



[1NF]:

Primary key for Entity 'Movie Timetable' is defined by 'movie_time_id'. By 'movie_id' attribute each tuple can be uniquely identified and also no repeating records will be found.

[2NF]:

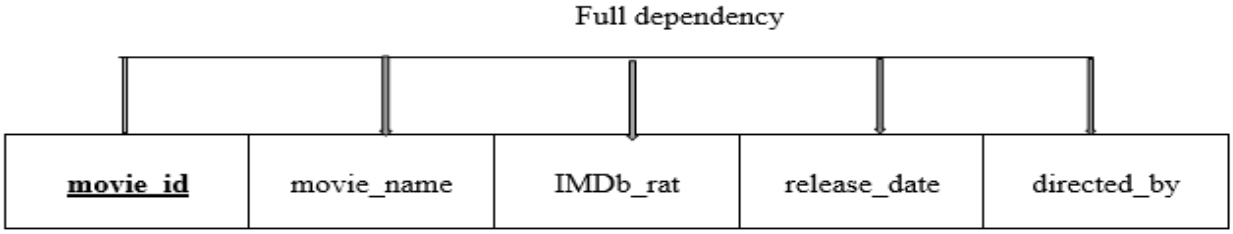
The entity 'Movie Timetable' has no partial dependencies within.

[3NF]:

The entity has no transitive dependencies. All attributes are functionally dependent on the primary key.

Movie Timetable (**movie_time_id** ,movie_id , movie_date , ,movie_time)

D. Normalisation for Movies table:



[1NF]:

Primary key for Entity 'Movie' is defined by 'movie_id'. By 'movie_id' attribute each tuple can be uniquely identified and also no repeating records will be found.

[2NF]:

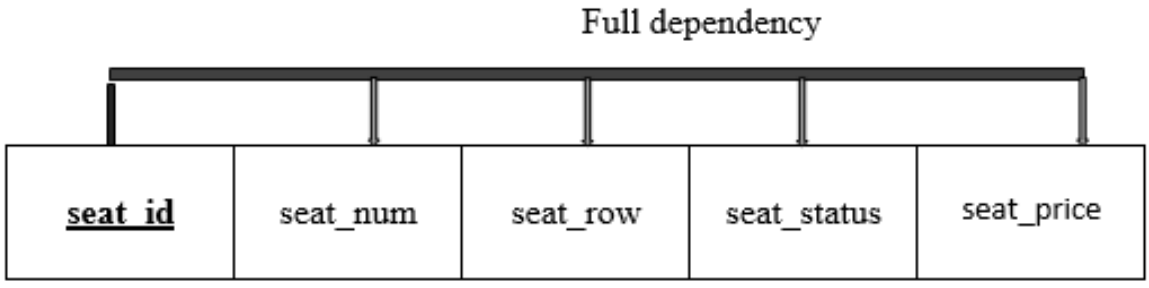
The entity 'Movie' has no partial dependencies within.

[3NF]:

The entity has no transitive dependencies. All attributes are functionally dependent on the primary key.

Movies (**movie_id** , movie_name , IMDb_rat , release_date , directed_by)

E. Normalisation For Seat details Table:



[1NF]:

Primary key for Entity ‘Seat details’ is defined by ‘seat_id’. By ‘seat_id’attribute each tuple can be uniquely identified and also no repeating records will be found.

[2NF]:

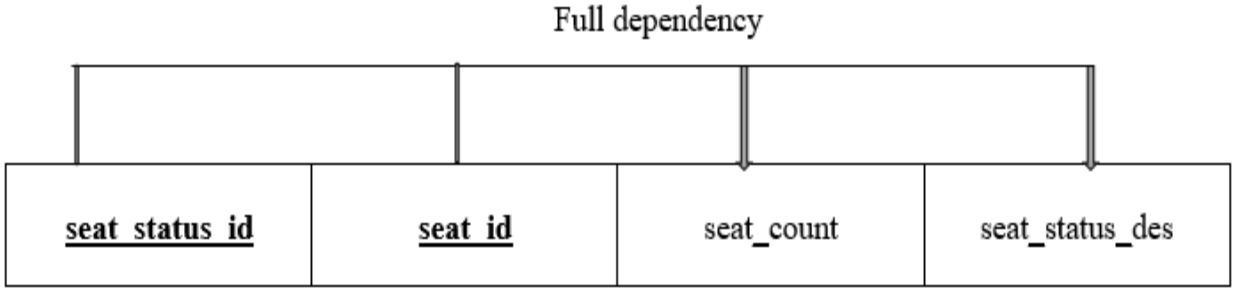
The entity ‘Seat details’ has no partial dependencies within.

[3NF]:

The entity has no transitive dependencies. All attributes are functionally dependent on the primary key

Seat details (**seat_id,** seat_num, seat_row, seat_status, seat_price)

F. Normalisation For Seat Availability Status Table:



[1NF]:

Primary key for Entity ‘Seat Availability Status’ is defined by ‘seat_id’ and ‘seat_status_id’ .Both attribute combinedly uniquely identified each tuple and also no repeating records will be found.

[2NF]:

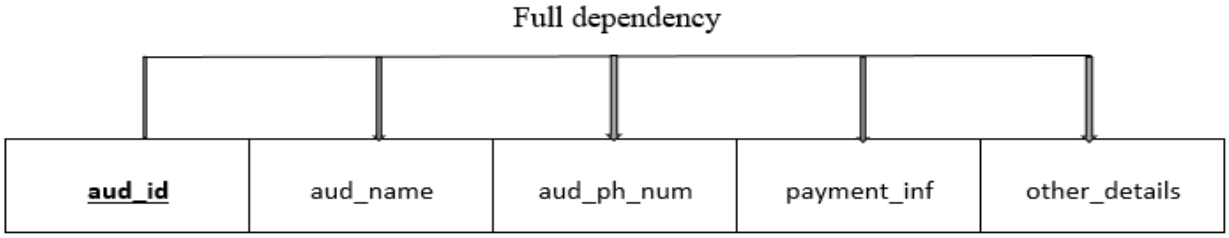
The entity ‘Seat Availability Status’ has no partial dependencies within.

[3NF]:

The entity has no transitive dependencies. All attributes are functionally dependent on the primary key

Seat Availability Status (**seat_status_id** , **seat_id** , seat_count, seat_status_des)

G. For Audience Table:



[1NF]:

Primary key for Entity ‘Audience’ is defined by ‘aud_id . By ‘aud_id ’attribute each tuple can be uniquely identified and also no repeating records will be found.

[2NF]:

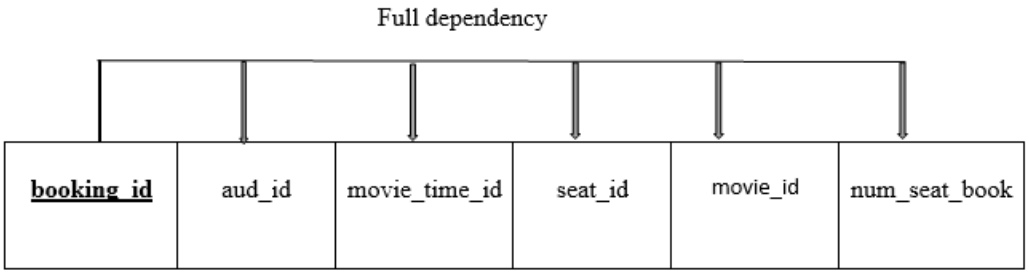
The entity’ Audience’ has no partial dependencies within.

[3NF]:

The entity has no transitive dependencies. All attributes are functionally dependent on the primary key

Audience (**aud_id** , aud_name ,aud_ph_num ,payment_inf, other_details)

H. Normalisation For Table Booking:



[1NF]:

Primary key for Entity ‘Booking’ is defined by ‘booking_id’ .No repeating records will be found.

[2NF]:

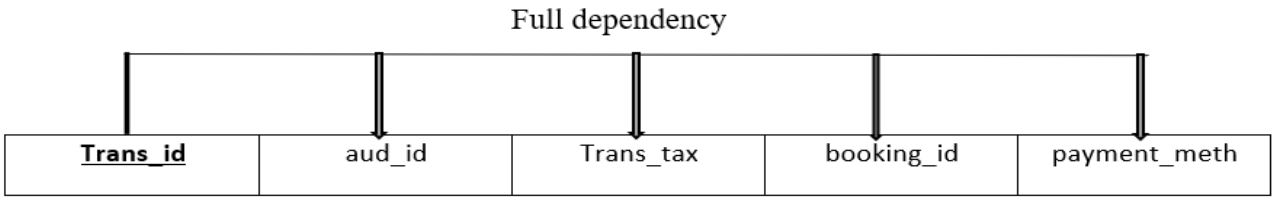
The entity ’Booking’ has no partial dependencies within.

[3NF]:

The entity has no transitive dependencies. All attributes are functionally dependent on the primary key

Booking(**booking_id**, movie_id ,movie_time_id, seat_id , aud_id , num_seat_book)

I. Normalisation For Payment -table:



[1NF]:

Primary key for Entity 'Payment' is defined by 'Trans_id' . By 'Trans_id 'attribute each tuple can be uniquely identified and also no repeating records will be found.

[2NF]:

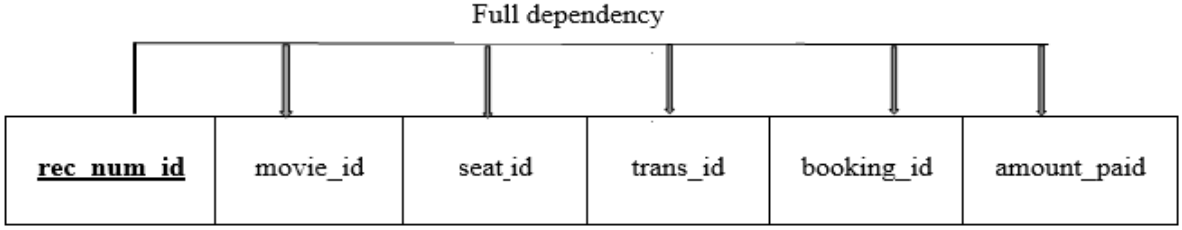
The entity' Payment' has no partial dependencies within.

[3NF]:

The entity has no transitive dependencies. All attributes are functionally dependent on the primary key.

Payment (**Trans_id** , aud_id , booking_id , Trans_tax , payment_meth)

J. Normalisation for Receipt Table:



[1NF]:

Primary key for Entity ‘Receipt’ is defined by ‘rec_num_id ’ . By ‘rec_num_id ’attribute each tuple can be uniquely identified and also no repeating records will be found.

[2NF]:

The entity’ For Receipt Provided to Audiences and Saved in Cashier System’ has no partial dependencies within.

[3NF]:

The entity has no transitive dependencies. All attributes are functionally dependent on the primary key.

Receipt (**rec_num_id** , movie_id , seat_id , booking_id , trans_id , amount_paid)

Phase 2

4. Table Creation

```
CREATE TABLE Branch (
branch_id varchar(8),
branch_name varchar(255),
branch_loc varchar(255),
branch_state varchar(255),
branch_mang varchar(255),
PRIMARY KEY (branch_id)
);

create table CinemaHall (
cin_h_id varchar(8) primary key,
cin_h_num int(8),
branch_id varchar(8),
cin_cap int (8),
Foreign key CinemaHall(branch_id)  references
Branch(branch_id)

);
```

```
CREATE TABLE Movies (
  movie_id varchar(6),
  movie_name char(15),
  IMDb_rat float(2,2),
  release_date date,
  directed_by char(20),
  PRIMARY KEY (movie_id)
);
```

```
CREATE TABLE Seat_Details(
    seat_id varchar(8) PRIMARY KEY,
    seat_num varchar(8),
    seat_row char(8),
    cin_h_id varchar(8),
    seat_price int(8),

   foreign key Seat_Details(cin_h_id) references
CinemaHall(cin_h_id)

);
create table SeatAvailabilityStatus (
  seat_status_id varchar(8) primary key,
  seat_id Varchar(8),
  seat_status_des varchar(25)


);
```

```
CREATE TABLE Audience (
    aud_id varchar(8) PRIMARY KEY NOT NULL,
    aud_name VARCHAR(255) NOT NULL,
    aud_ph_num VARCHAR(255),
    payment_info VARCHAR(255) NOT NULL,
    other_details VARCHAR(255)
);


CREATE TABLE Booking (
booking_id varchar(8) PRIMARY KEY NOT NULL,
    aud_id varchar (8)   NOT NULL,
    movie_id varchar (8) NOT NULL,
    movie_time_id varchar(8) NOT NULL,
    seat_id  varchar(8)  NOT NULL,
    num_seat_book int (8) NOT NULL

);


CREATE TABLE Payment (
trans_id   VARCHAR(8) NOT NULL ,
aud_id   VARCHAR(8) NOT NULL,
booking_id VARCHAR(8) NOT NULL,
trans_tax float(6,2) NOT NULL,
payment_meth char(15) NOT NULL,
PRIMARY KEY (trans_id)

);
```

```
CREATE TABLE Receipt(
    rec_num_id VARCHAR(8) NOT NULL PRIMARY KEY ,
    movie_id VARCHAR(8) NOT NULL,

seat_id VARCHAR(8) NOT NULL,
booking_id VARCHAR(8) NOT NULL,
    trans_id varchar(8) NOT NULL,
    amount_paid FLOAT(8,2) NOT NULL


);



CREATE TABLE Movie_timetable (

movie_time_id  varchar(8) NOT NULL PRIMARY key,
movie_id VARCHAR(8) NOT NULL  ,
movie_time TIME NOT NULL,
movie_date DATE NOT NULL,
cin_h_id VARCHAR(8) NOT NULL
);
```

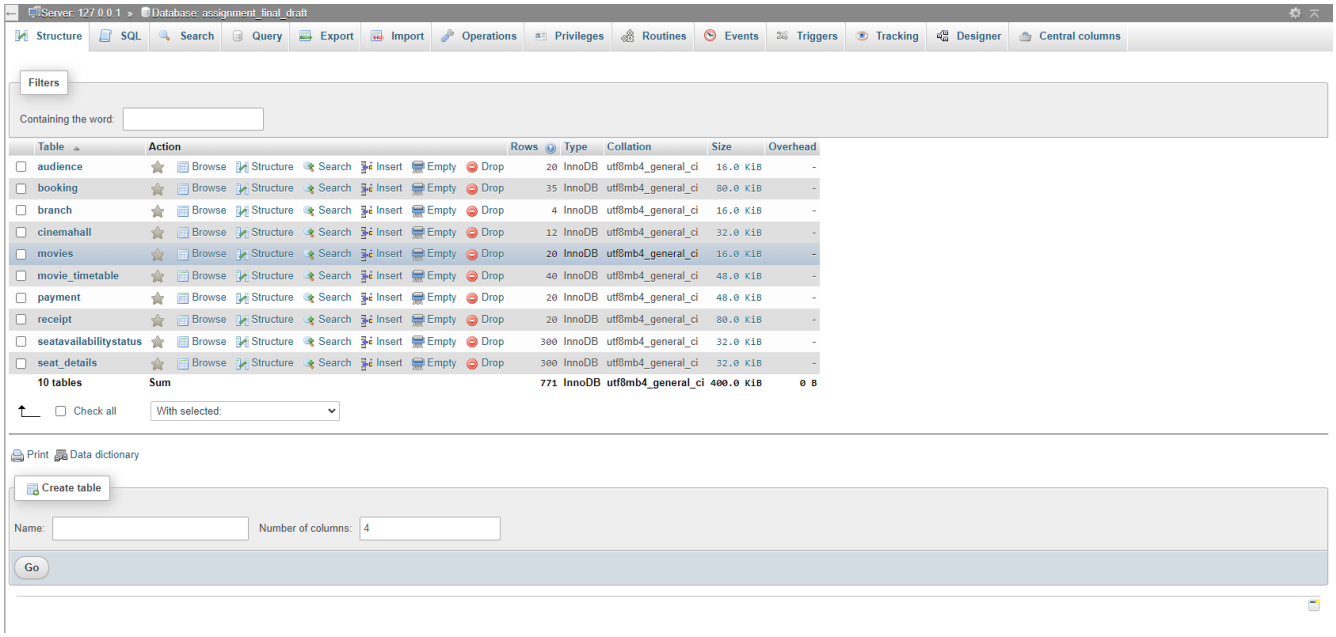


Figure - Creation of all the tables required.

5. Data Insertion

Branch:

```
INSERT INTO Branch (branch_id, branch_name, branch_loc, branch_state, branch_mang)
VALUES      ('BR01', 'USC Cyberjaya', 'DPulze', 'Selangor', 'Jackson'),
('BR02', 'USC IOI City', 'IOICityMall', 'W.P Putrajaya', 'Jamal'),
('BR03', 'USC Petaling Jaya', 'Petaling Jaya', 'W.P Kuala Lumpur', 'John'),
('BR04', 'USC Aurum', 'The Gardens Mall', 'W.P Kuala Lumpur', 'Justine');
```

Cinema Hall:

```
INSERT INTO CinemaHall (cin_h_id, cin_h_num, branch_id, cin_cap)
VALUES      ('CH11', '1', 'BR01', '30'),
        ('CH12', '2', 'BR01', '30'),
        ('CH13', '3', 'BR01', '30'),

        ('CH21', '1', 'BR02', '30'),
        ('CH22', '2', 'BR02', '30'),
        ('CH23', '3', 'BR02', '30'),

        ('CH31', '1', 'BR03', '30'),
        ('CH32', '2', 'BR03', '30'),
        ('CH33', '3', 'BR03', '30'),

        ('CH41', '1', 'BR04', '30'),
        ('CH42', '2', 'BR04', '30'),
        ('CH43', '3', 'BR04', '30');
```

Movies:

```
INSERT INTO Movies (movie_id, movie_name, IMDb_rat, release_date, directed_by)
VALUES      ('MV01', 'The Batman', '8.4','2022-03-04','Matt Reeves'),
('MV02', 'Uncharted', '6.7','2022-02-07','Ruben Fleischer'),
('MV03', 'Scream', '7.3','2022-01-14','Tyler Gillett'),
('MV04', 'Jackass Forever', '7.3','2022-02-01','Jeff Tremaine'),
('MV05', 'Death on the Nile', '6.6','2022-02-09','Kenneth Branagh'),
('MV06', 'Moonfall', '5.3','2022-04-02','Roland Emmerich'),
('MV07', '76 DAYS', '9.8','2021-11-05','Hao Wu'),
                ('MV08', 'QUO VADIS, AIDA?', '9.8','2021-11-01','Jasmila Zbanic'),
                ('MV09', '76 DAYS', '9.8','2021-11-05','Hao Wu'),
                ('MV10', 'SLALOM', '9.7','2021-10-07','Charlène Favier'),
                ('MV11', 'LUZZU', '9.6','2021-07-04','Alex Camilleri'),
                ('MV12', 'SABAYA', '9.6','2021-09-06','Hogir Hirori'),
                ('MV13', 'HIVE', '9.5','2021-06-11','Blerta Basholli'),
                ('MV14', 'AZOR', '9.4','2021-05-13','Andreas Fontana'),
                ('MV15', 'MAYOR', '9.3','2021-04-16','David Osit'),
                ('MV16', 'PAPER SPIDERS', '9.2','2021-03-09','Inon Shampanier'),
                ('MV17', 'CHANGING THE GAME', '9.1','2021-01-13','Michael Barnett'),
                ('MV18', 'THE FATHER', '8.9','2021-05-13','Florian Zeller'),
                ('MV19', 'MLK/FBI', '9.8','2021-07-21','Sam Pollard'),
                ('MV20', 'THERE IS NO EVIL', '9.7','2021-01-09','Mohammad Rasoulof');
```

Seat Details:

Insert into Seat_Details(seat_id, seat_num, seat_row, cin_h_id, seat_price) VALUES

('A101', 'A1', 'A', 'CH11', '12'),

('A102', 'A2', 'A', 'CH11', '12'),

('A103', 'A3', 'A', 'CH11', '12'),

('A104', 'A4', 'A', 'CH11', '12'),

('A105', 'A5', 'A', 'CH11', '12'),

('B101', 'B1', 'B', 'CH11', '12'),

('B102', 'B2', 'B', 'CH11', '12'),

('B103', 'B3', 'B', 'CH11', '12'),

('B104', 'B4', 'B', 'CH11', '12'),

('B105', 'B5', 'B', 'CH11', '12'),

('C101', 'C1', 'C', 'CH11', '16'),

('C102', 'C2', 'C', 'CH11', '16'),

('C103', 'C3', 'C', 'CH11', '16'),

('C104', 'C4', 'C', 'CH11', '16'),

('C105', 'C5', 'C', 'CH11', '16'),

('D101', 'D1', 'D', 'CH11', '16'),

('D102', 'D2', 'D', 'CH11', '16'),

('D103', 'D3', 'D', 'CH11', '16'),

('D104', 'D4', 'D', 'CH11', '16'),

('D105', 'D5', 'D', 'CH11', '16'),

('E101', 'E1', 'E', 'CH11', '12'),

('E102', 'E2', 'E', 'CH11', '12'),

('E103', 'E3', 'E', 'CH11', '12'),

('E104', 'E4', 'E', 'CH11', '12'),

('E105', 'E5', 'E','CH11', '12'), etc.

Note: (The size is too big, kindly refer to the sql for the full insertion)

Seat Availability Status:

INSERT INTO SeatAvailabilityStatus (seat_status_id , seat_id , seat_status_des)

VALUES

('S001', 'A101', 'available'),

('S002', 'A102', 'available'),

('S003', 'A103', 'unavailable'),

('S004', 'A104', 'available'),

('S005', 'A105', 'available'),

('S006', 'B101', 'available'),

('S007', 'B102', 'unavailable'),

('S008', 'B103', 'available'),

('S009', 'B104', 'available'),

('S010', 'B105', 'available'),

('S011', 'C101', 'available'),

('S012', 'C102', 'available'),

('S013', 'C103', 'available'),

('S014', 'C104', 'available'),

('S015', 'C105', 'available'),

('S016', 'D101', 'available'),

('S017', 'D102', 'available'),

('S018', 'D103', 'unavailable'),

('S019', 'D104', 'unavailable'),

('S020', 'D105', 'unavailable'),

('S021', 'E101', 'unavailable'),

('S022', 'E102', 'available'),

('S023', 'E103', 'unavailable'),

('S024', 'E104', 'available'),

('S025', 'E105', 'available'), etc.

Note: (The size is too big, kindly refer to the sql for the full insertion)

Audience:

Insert into Audience(aud_id, aud_name, aud_ph_num, payment_info, other_details) VALUES

('AU-001', 'Mahin Khan', 01128061585, 'online banking', ''),

('AU-002', 'Ishtiaq Ahmed Bhuiyan', 01426067985, 'online banking', ''),

('AU-003', 'Akila Ibnat', 06328061585, 'online banking', 'Disabled'),

('AU-004', 'Anisa Mehreen', 03464361585, 'online banking', 'Assistance-req'),

('AU-005', 'Haziq Afham Bin', 0253256436, 'CASH', ''),

('AU-006', 'Li Min Yu', 0115235155, 'online banking', ''),

('AU-007', 'Xu Yue', 0162624155, 'online banking', 'Assistance-req'),

('AU-008', 'Abdulla Sayeed', 01134563158, 'online banking', ''),

('AU-009', 'Hassan', 04536361585, 'CARD', 'Service-req'),

('AU-010', 'Kareem Iftekhar', 0734561585, 'CASH', 'Disabled'),

('AU-011', 'Noor-uzzaman', 01264661585, 'CASH', 'Assistance-req'),

('AU-012', 'Mahfuzur Rahman', 03128346575, 'CASH', ''),

('AU-013', 'Rahat Hassan', 0112246246, 'CASH', ''),

('AU-014', 'Mahmudur Rahman', 01728764634, 'online banking', ''),

('AU-015', 'Rabbi kul', 0134546255, 'CASH', ''),

('AU-016', 'Zhang', 0364634685, 'online banking', ''),

('AU-017', 'Lio', 0146436125, 'online banking', ''),

('AU-018', 'Wang Cio', 0176582334, 'CASH', ''),

('AU-019', 'Deepti', 0153272544, 'CASH', 'Assistance-req'),

('AU-020', 'Adam', 0146334422, 'CASH', '');

Booking:

Insert into Booking (booking_id, aud_id, movie_id, movie_time_id, seat_id, num_seat_book) values

('BID-001', 'AU-001', 'MV13', 'MT12', 'E505', '3'),

('BID-002', 'AU-001', 'MV13', 'MT12', 'E503', '3'),

('BID-003', 'AU-001', 'MV13', 'MT12', 'E502', '3'),

('BID-004', 'AU-002', 'MV01', 'MT09', 'A101', '2'),

('BID-005', 'AU-002', 'MV01', 'MT09', 'A102', '2'),

('BID-006', 'AU-003', 'MV04', 'MT14', 'C103', '4'),

('BID-007', 'AU-003', 'MV04', 'MT14', 'C102', '4'),

('BID-008', 'AU-003', 'MV04', 'MT14', 'C101', '4'),

('BID-009', 'AU-003', 'MV04', 'MT14', 'C104', '4'),

('BID-010', 'AU-004', 'MV09', 'MT04', 'A304', '1'),

('BID-011', 'AU-005', 'MV07', 'MT03', 'A301', '1'),

('BID-012', 'AU-006', 'MV12', 'MT02', 'E401', '1'),

('BID-013', 'AU-007', 'MV14', 'MT19', 'E202', '3'),

('BID-014', 'AU-007', 'MV14', 'MT19', 'E201', '3'),

('BID-015', 'AU-007', 'MV14', 'MT19', 'E203', '3'),

('BID-016', 'AU-008', 'MV05', 'MT08', 'B101', '2'),

('BID-017', 'AU-008', 'MV05', 'MT08', 'B102', '2'),

('BID-018', 'AU-009', 'MV04', 'MT17', 'B405', '1'),

('BID-019', 'AU-010', 'MV04', 'MT01', 'A205', '1'),

('BID-020', 'AU-011', 'MV18', 'MT20', 'C202', '1'),

Note: (The size is too big, kindly refer to the sql for the full insertion)

Payment:

```
INSERT into Payment(trans_id, aud_id, booking_id ,trans_tax,
        payment_meth) VALUES
('TRX-0001','AU-001','BID-001',2.00,'online banking'),
('TRX-0002','AU-002','BID-002',2.40,'cash'),
('TRX-0003','AU-003','BID-003',1.55,'card'),
('TRX-0004','AU-004','BID-004',1.80,'online banking'),
('TRX-0005','AU-005','BID-005',1.45,'cash'),
('TRX-0006','AU-006','BID-006',3.00,'cash'),
('TRX-0007','AU-007','BID-007',2.10,'online banking'),
('TRX-0008','AU-008','BID-008',4.10,'card'),
('TRX-0009','AU-009','BID-009',3.95,'online banking'),
('TRX-0011','AU-010','BID-010',1.88,'card'),
('TRX-0012','AU-011','BID-011',2.88,'cash'),
('TRX-0013','AU-012','BID-013',3.88,'online banking'),
('TRX-0014','AU-013','BID-012',5.18,'card'),
('TRX-0015','AU-014','BID-014',3.08,'online banking'),
('TRX-0016','AU-015','BID-015',3.18,'cash'),
('TRX-0017','AU-016','BID-016',4.68,'cash'),
('TRX-0018','AU-017','BID-017',1.78,'cash'),
('TRX-0019','AU-018','BID-018',5.12,'online banking'),
('TRX-0010','AU-019','BID-019',2.22,'online banking'),
('TRX-0020','AU-020','BID-020',3.44,'card');
```

Note: (The size is too big, kindly refer to the sql for the full insertion)

Receipt:

```
INSERT into Receipt(rec_num_id,movie_id,seat_id,booking_id,trans_id,amount_paid)
VALUES
('R0000001','MV01','A101','BID-001','TRX-0001',15.00),
('R0000002','MV02','A102','BID-002','TRX-0002',15.00),
('R0000003','MV03','A103','BID-003','TRX-0003',150.00),
('R0000004','MV04','A104','BID-004','TRX-0004',121.00),
('R0000005','MV05','A105','BID-005','TRX-0005',111.00),
('R0000006','MV06','B101','BID-006','TRX-0006',80.00),
('R0000007','MV07','B102','BID-007','TRX-0007',45.10),
('R0000008','MV08','B103','BID-008','TRX-0008',19.90),
('R0000009','MV09','B104','BID-009','TRX-0009',14.58),
('R0000010','MV10','B105','BID-010','TRX-0010',13.08),
('R0000011','MV11','C101','BID-011','TRX-0011',15.06),
('R0000012','MV12','C102','BID-012','TRX-0012',17.50),
('R0000013','MV13','C103','BID-013','TRX-0013',15.73),
('R0000014','MV14','C104','BID-014','TRX-0014',10.50),
('R0000015','MV15','C105','BID-015','TRX-0015',15.00),
('R0000016','MV16','D101','BID-016','TRX-0016',156.00),
('R0000017','MV17','D102','BID-017','TRX-0017',17.30),
('R0000018','MV18','D103','BID-018','TRX-0018',15.40),
('R0000019','MV19','D104','BID-019','TRX-0019',15.80),
('R0000020','MV20','D105','BID-020','TRX-0020',15.90);
```

Movie Timetable:

INSERT INTO Movie_timetable (movie_time_id,movie_id, movie_time,movie_date,cin_h_id) VALUES

('MT01','MV01',"12:00:00",'2022-04-02','CH11'),

('MT02','MV02',"13:00:00",'2022-04-02','CH12'),

('MT03','MV03',"14:00:00",'2022-04-02','CH13'),

('MT04','MV04',"15:00:00",'2022-04-02','CH21'),

('MT05','MV05',"16:00:00",'2022-04-02','CH22'),

('MT06','MV06',"14:00:00",'2022-04-03','CH23'),

('MT07','MV07',"15:00:00",'2022-04-03','CH31'),

('MT08','MV08',"16:00:00",'2022-04-03','CH32'),

('MT09','MV09',"17:00:00",'2022-04-03','CH33'),

('MT10','MV10',"18:00:00",'2022-04-03','CH41'),

('MT11','MV11',"11:00:00",'2022-04-03','CH42'),

('MT12','MV12',"12:00:00",'2022-04-04','CH43'),

('MT13','MV13',"14:30:00",'2022-04-02','CH11'),

('MT14','MV14',"13:00:00",'2022-04-02','CH12'),

('MT15','MV15',"15:00:00",'2022-04-04','CH13'),

('MT16','MV16',"12:00:00",'2022-04-04','CH21'),

('MT17','MV17',"12:00:00",'2022-04-04','CH22'),

('MT18','MV18',"12:00:00",'2022-04-05','CH23'),

('MT19','MV19',"12:00:00",'2022-04-05','CH31'),

('MT20','MV20',"12:00:00",'2022-04-06','CH32'),etc.

Note: (The size is too big, kindly refer to the sql for the full insertion)

6. Alterations

Seat Availability:

```
ALTER TABLE SeatAvailabilityStatus
ADD FOREIGN KEY (seat_id) REFERENCES Seat_Details(seat_id);
```

Booking:

```
ALTER TABLE Booking
ADD FOREIGN KEY (aud_id) REFERENCES Audience(aud_id),
ADD FOREIGN KEY (movie_id) REFERENCES Movies(movie_id),
ADD FOREIGN KEY (movie_time_id) REFERENCES Movie_timetable(movie_time_id),
ADD FOREIGN KEY (seat_id) REFERENCES Seat_Details(seat_id);
```

Payment:

```
ALTER TABLE Payment
  ADD FOREIGN KEY (aud_id) REFERENCES Audience (aud_id),
  ADD FOREIGN KEY (booking_id) REFERENCES Booking (booking_id);
```

Receipt:

```
ALTER TABLE Receipt
  ADD FOREIGN KEY (movie_id) REFERENCES Movies (movie_id),
  ADD FOREIGN KEY (seat_id) REFERENCES Seat_Details (seat_id),
  ADD FOREIGN KEY (booking_id) REFERENCES Booking (booking_id),
  ADD FOREIGN KEY (trans_id) REFERENCES Payment (trans_id);
```

Movie Timetable:

```
ALTER TABLE Movie_timetable
  ADD FOREIGN KEY (movie_id) REFERENCES Movies (movie_id),
  ADD FOREIGN KEY (cin_h_id) REFERENCES CinemaHall(cin_h_id);
```

7. Queries

Group leader: Abdulla Al Mahin Khan

1. Write one SQL command to show the sum of booking made for each movie
2. Write one SQL command to show the total collection for each movie
3. Write one SQL command to show the number of halls for each cinema branch
4. Write one SQL command to show the average seat price

/*Query 1 solution*/

Select b.movie_id, count(booking_ID) from booking as b , Movies as m

Where b.movie_id = m.movie_id

Group by b.movie_id;

```
MariaDB [assignment_final_draft]> Select b.movie_id, count(booking_ID) from booking as b , Movies as m  Where b.mov
ie_id = m.movie_id Group by b.movie_id;
+----------+-------------------+
| movie_id | count(booking_ID) |
+----------+-------------------+
| MV01     |                 4 |
| MV02     |                 2 |
| MV04     |                 6 |
| MV05     |                 2 |
| MV07     |                 1 |
| MV09     |                 3 |
| MV11     |                 1 |
| MV12     |                 2 |
| MV13     |                 3 |
| MV14     |                 3 |
| MV15     |                 1 |
| MV17     |                 3 |
| MV18     |                 4 |
+----------+-------------------+
13 rows in set (0.001 sec)
```

Figure – Solution to Query 1

/*Query 2 solution*/

SELECT movie_id, SUM(amount_paid) as sum_amount

FROM Receipt

GROUP BY movie_id;

```
MariaDB [assignment_draft_2]> SELECT movie_id, SUM(amount_paid) as sum_amountFROM ReceiptGROUP BY movie_id;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server
version for the right syntax to use near 'ReceiptGROUP BY movie_id' at line 1
MariaDB [assignment_draft_2]> SELECT movie_id, SUM(amount_paid) as sum_amount FROM Receipt GROUP BY movie_id;
+----------+------------+
| movie_id | sum_amount |
+----------+------------+
| MV01     |      15.00 |
| MV02     |      15.00 |
| MV03     |     150.00 |
| MV04     |     121.00 |
| MV05     |     111.00 |
| MV06     |      80.00 |
| MV07     |      45.10 |
| MV08     |      19.90 |
| MV09     |      14.58 |
| MV10     |      13.08 |
| MV11     |      15.06 |
| MV12     |      17.50 |
| MV13     |      15.73 |
| MV14     |      10.50 |
| MV15     |      15.00 |
| MV16     |     156.00 |
| MV17     |      17.30 |
| MV18     |      15.40 |
| MV19     |      15.80 |
| MV20     |      15.90 |
+----------+------------+
20 rows in set (0.001 sec)
```

Figure – Solution to Query 2

/*Query 3 solution*/

```
SELECT branch_id,
COUNT(*)
FROM CinemaHall
GROUP BY branch_id;
```

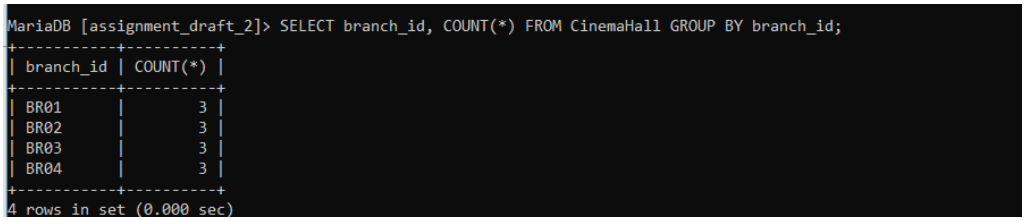


Figure – Solution to Query 3

/*Query 4 solution*/

```
SELECT avg(seat_price)
FROM  Seat_Details;
```

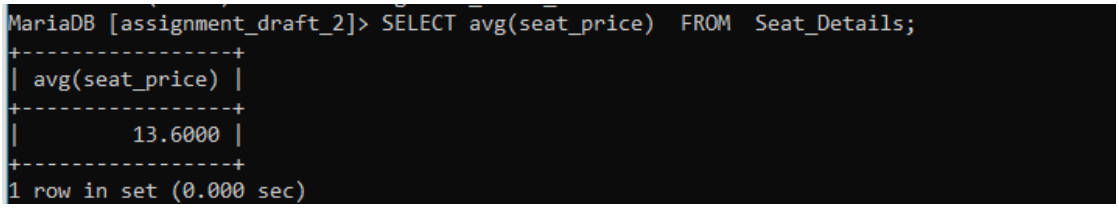


Figure – Solution to Query 4