Database Management System

Case Study: Movie Theatre Database

Statement:

Multiplex is a movie theatre complex with multiple screens within a single complex. They are usually housed in a specially designed building. Movie Theatre database system is an application of database management system which provides the user to book movie tickets and manager to keep their records. Movie Theatre database system revolutionized the trivial system of distribution of paper tickets and made it easy for users to book tickets from home.

This system provides secure, organized and efficient storage of huge data. One does not have to be physically present at the theatre to get their job done. With the advent of various online platform like BookMyShow and Paytm use type of movie theatre database system, the process became very simple, hustle free and user-friendly. This project aims to provide an insight of such Movie Theatre database system.

Description:

A movie can have multiple screenings at different times, or it can be screened simultaneously in adifferent auditorium. Movies to be shown on the screen in the theatre. Each movie is described by its meta-data movie id, movie title, movie director, movie length and movie genre. One movie can be shown at different screen but a screen cannot show more than one movie at a given time.

We want to track who entered each reservation/sale into the system. Customer who books the tickets. Each customer will be described by Id, name, age, phone number. Customer can book one or more tickets. Using Customer\_Id attributes of booking we can find details of the person who has book the tickets. One screening can have multiple booking by people. As we mentioned customer can book one or more tickets but one booking can confirm only one seat. For which seat reserved entity contains booking\_id to get details of booking. Also, for booking it will store the data of booked tickets containing details of price and bookid.

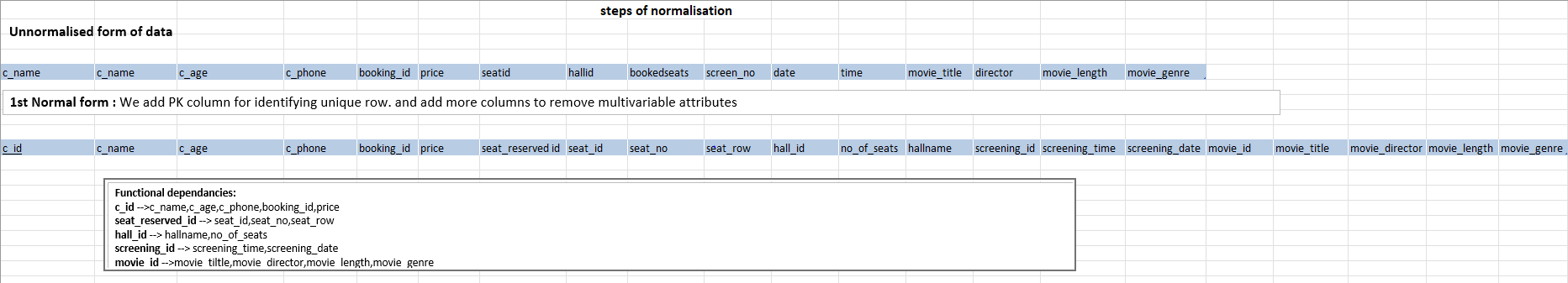
Contemporary multiplex movie theatres can have one or more halls within a larger complex. This auditorium stores the data of hall and seats combined describe by Hall\_Id, Number of seats and Hall name. Each auditorium can have a different number of seats.

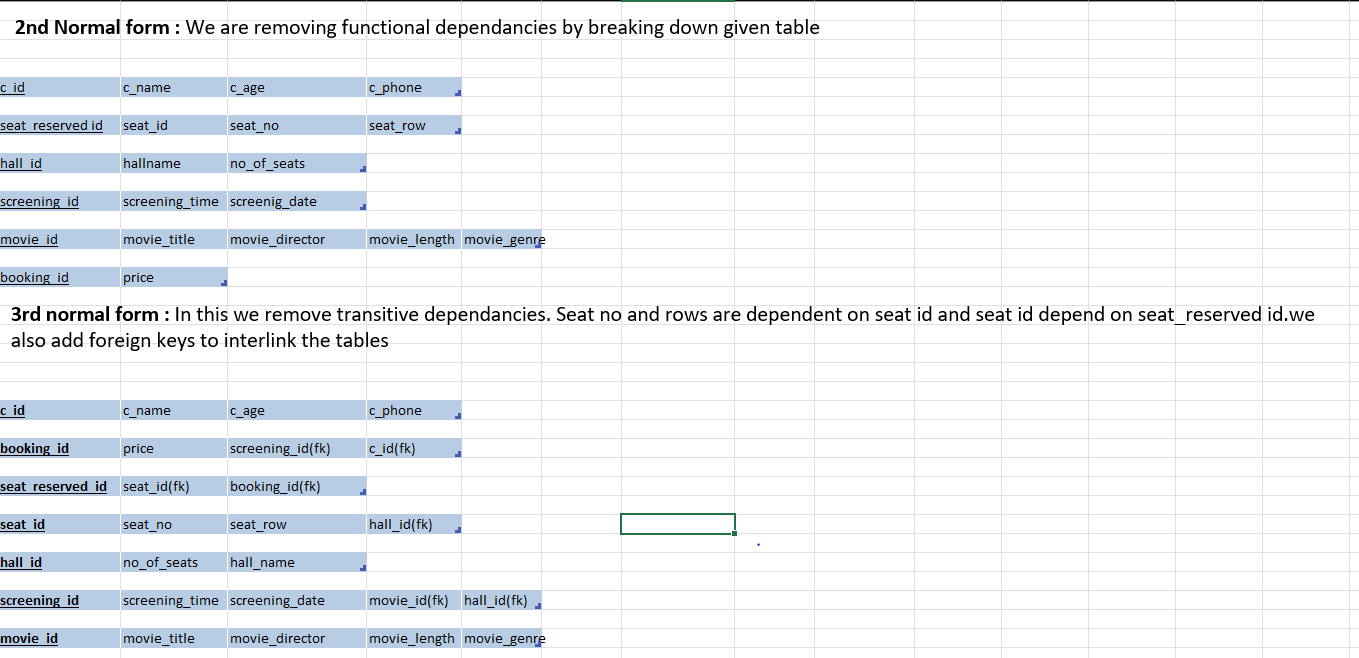
Seats are numerated with row number and seat position within a row. For which it will contain the information of seat in a hall and it will be described by its seat\_ID, seat row, seat number. Based on the booking we need the data of seat reserved. One hall can have many seats. There are multiple seats reserved for screening.

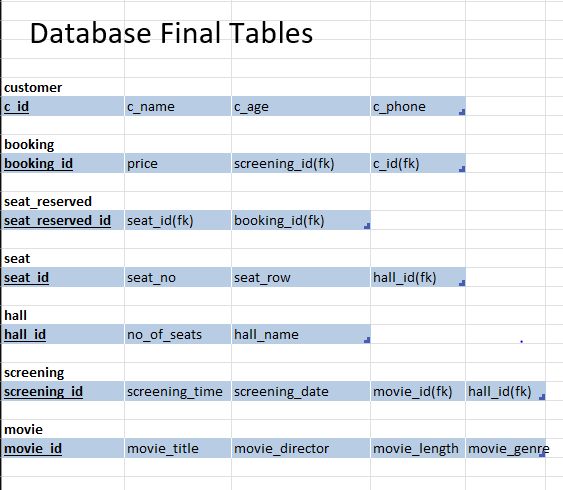
For each screening a seat can be reserved/sold only once and it will combine the data of hall and movie which it is shown. For screening we need the data of screen time and date. Many screenings can be presented in one hall at different times and date. There can be multiple reservation on different time and date of a particular seat.

Normalization:

**Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalization is to eliminate redundant or repetitive data and ensure data is stored logically. (Please check attached excel sheet for the details)





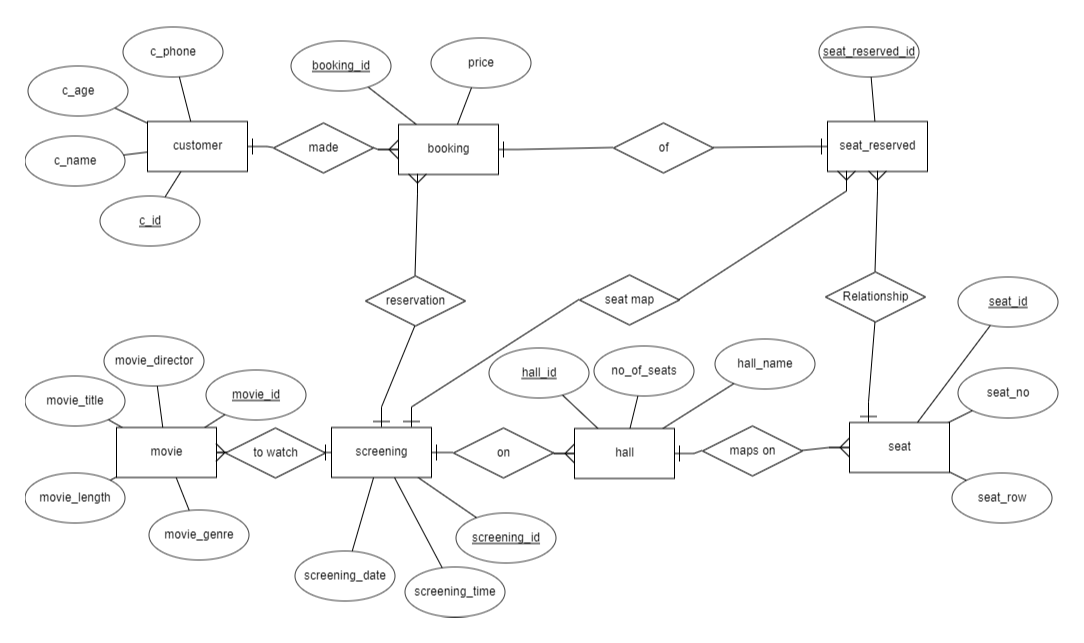


Entity Relationship Diagram:

Entity Relationship Diagram or ERD is a type of structural diagram for use in database design. It contains different symbols and connectors that visualize two important information:

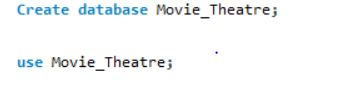
The major entities within the system scope, and the inter-relationships among these entities.

Below is the ERD for Car dealer management system, according to the results of normalization.

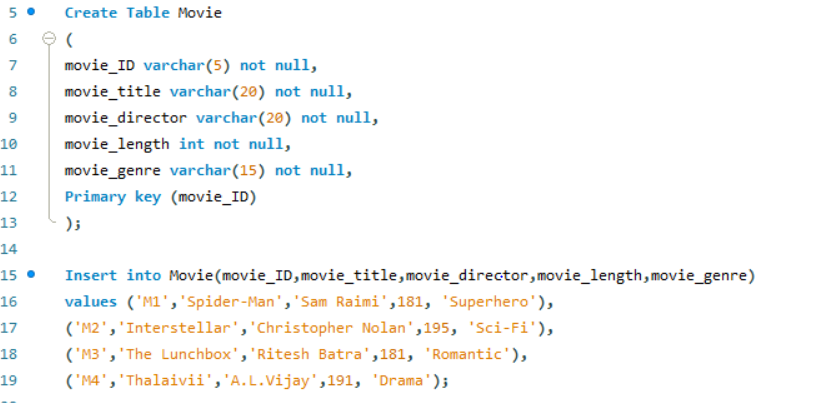


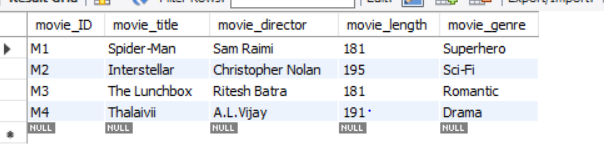
Creation of all Tables:

Creating Movie\_Theatre database:

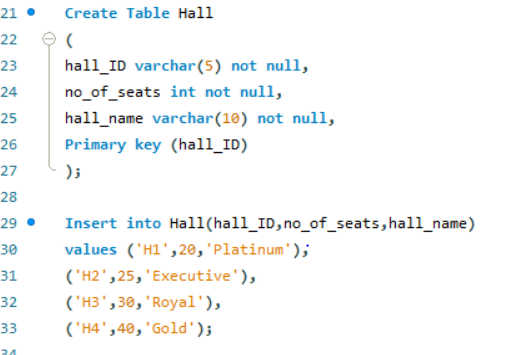


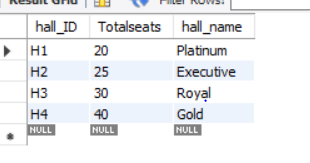
Movie Table:



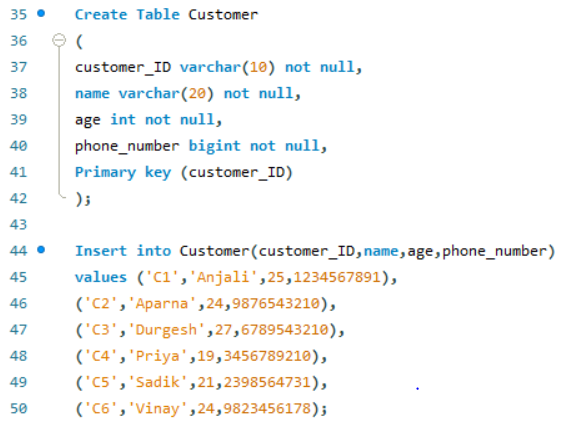


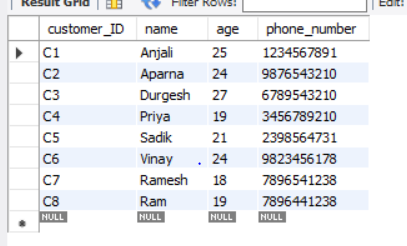
Hall table:



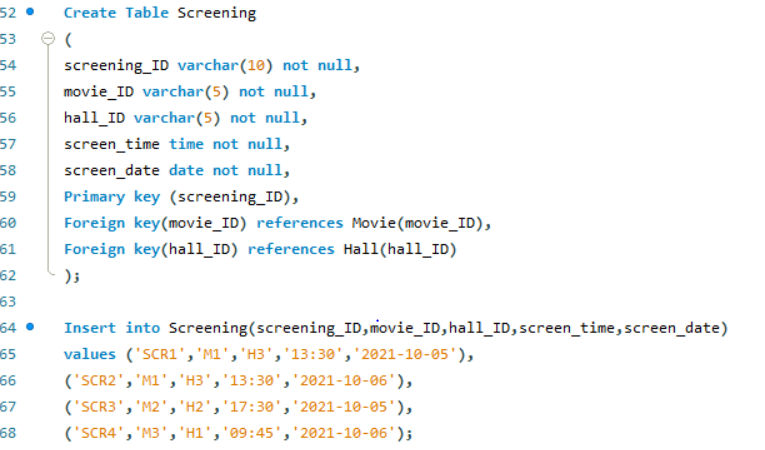


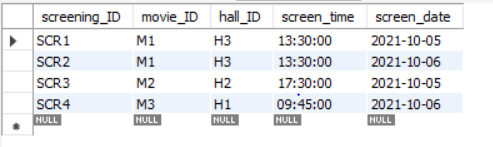
Customer table:



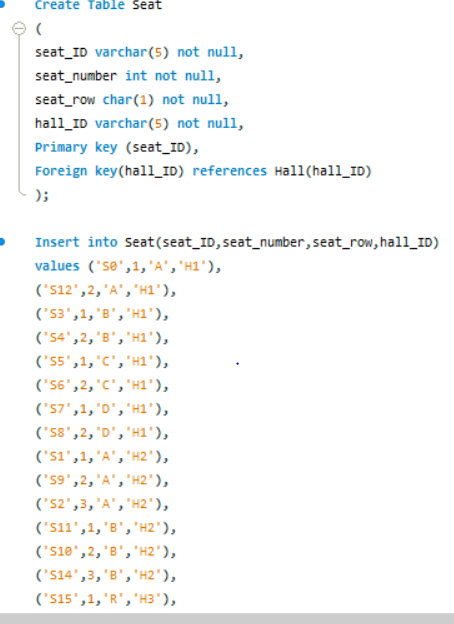


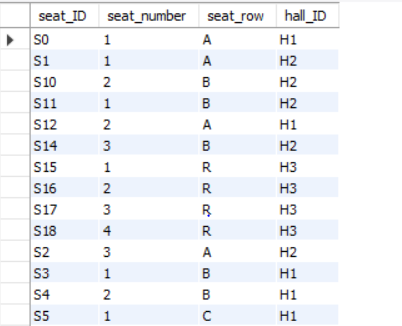
Screening table:



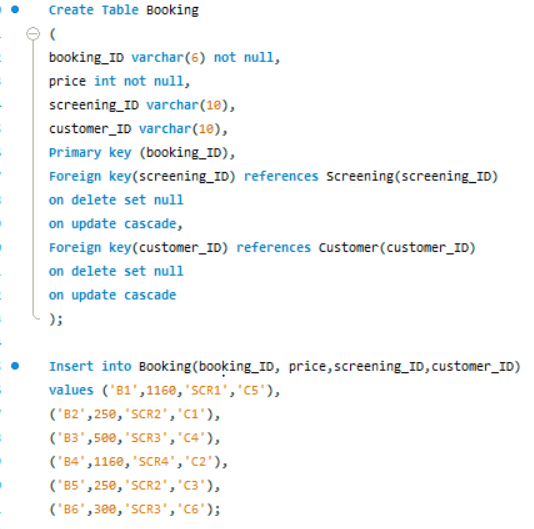


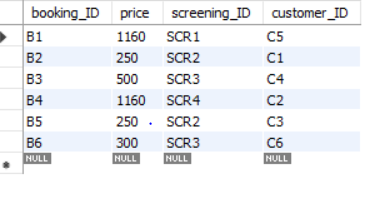
Seat table:



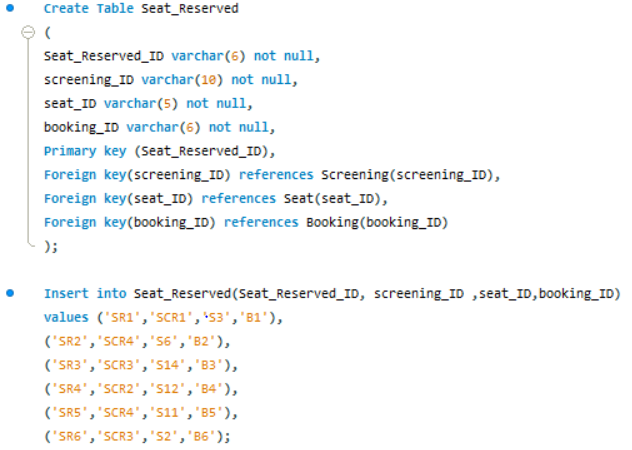


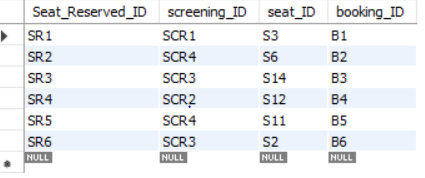
Booking Table





Seat\_Reserved Table:



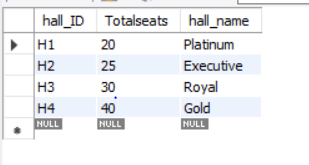


Queries:

DDL queries:

1. Change the name of column name no\_of\_seats to Totalseats

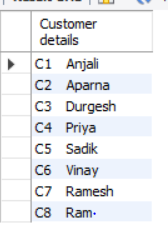
->Alter table Hall change column no\_of\_seats Totalseats int not null;



# DQL Queries

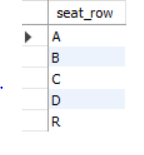
2. Display customer\_ID and name in one column as Customer details

->Select concat(customer\_ID,' ',name) as "Customer details" from customer;

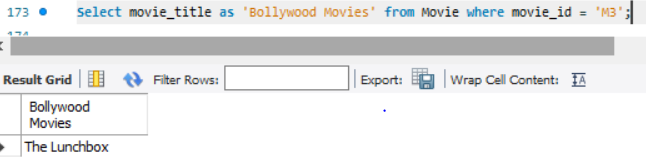


3. Display unique rows from table seat

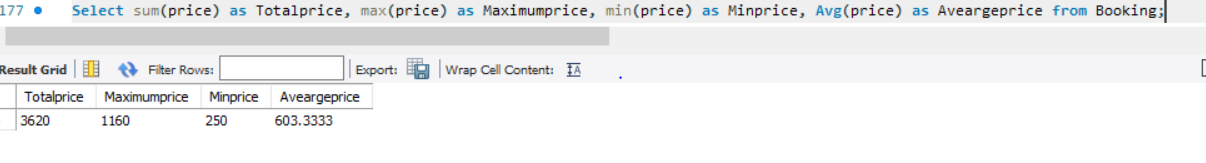
->Select distinct seat\_row from seat order by seat\_row ;

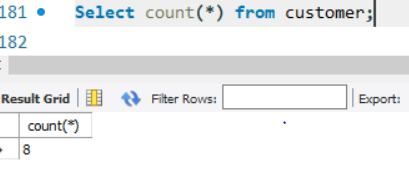


4. Display Bollywood movie from movie table

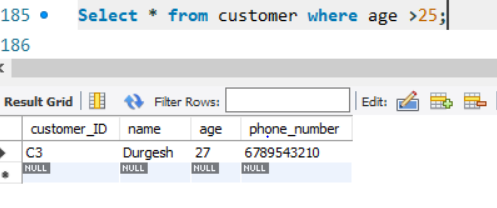


5. Display total price, max, min, average of price from booking table

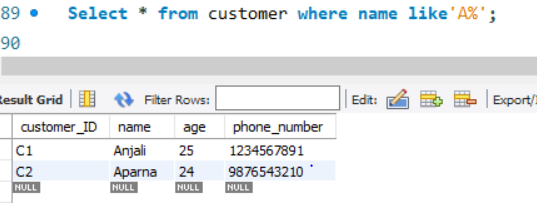
6. Display total number of customers

`

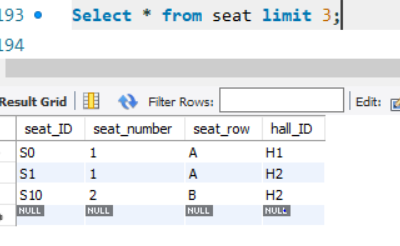
7. Display customer who’s age is more than 25 years



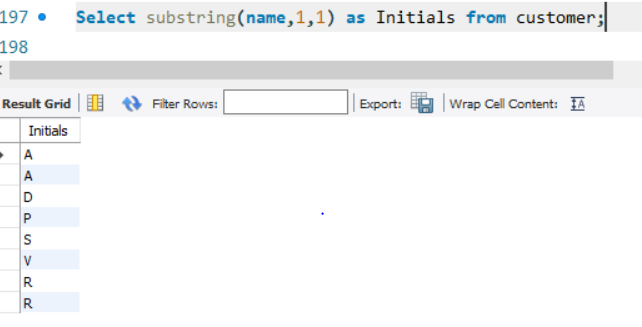
8. Display customers who’s name starting with a.



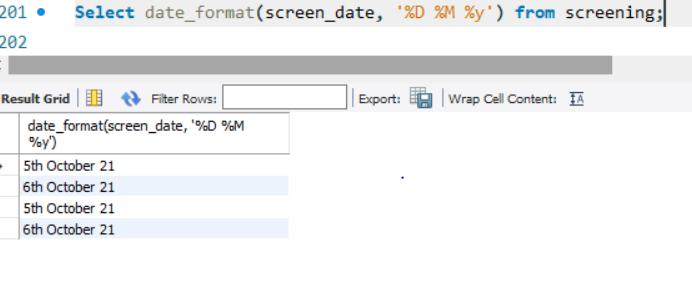
9. Display top3 details from seat table



10. Display the initial letter from customer

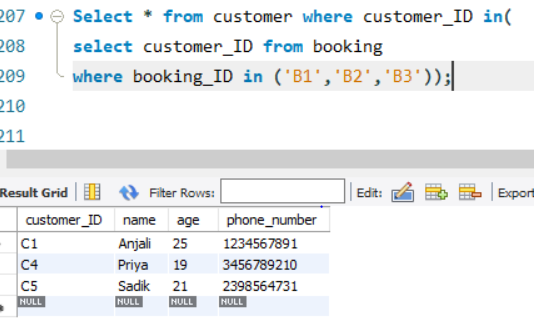


11. Display date in format of 5th October 21



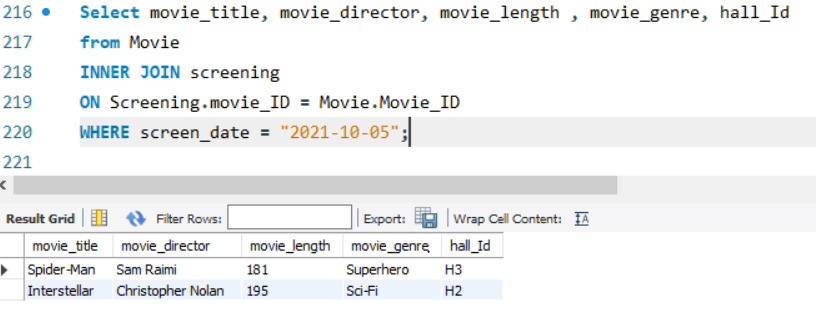
# Subquery:

1.Display customer details having booking id b1,b2,b3



# Joins:

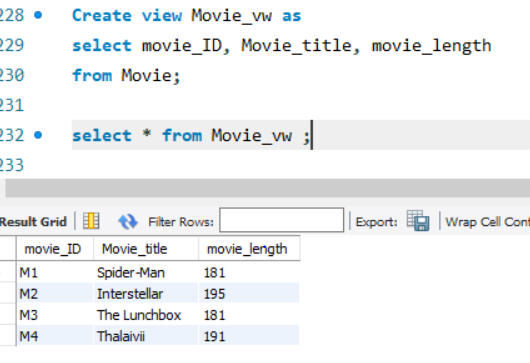
1.Which movie are screened on 2021-10-05?



# Views:

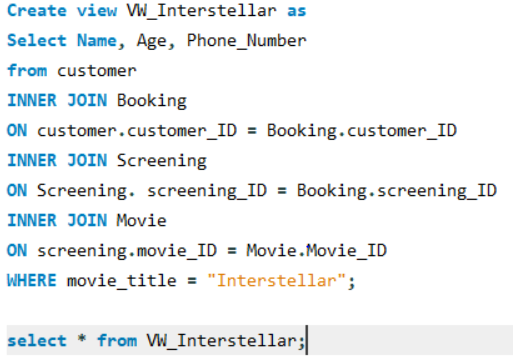
# Simple View:

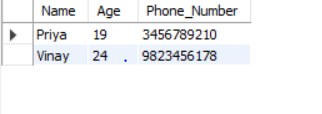
1.Create view for movie table and show movie details



# Complex view:

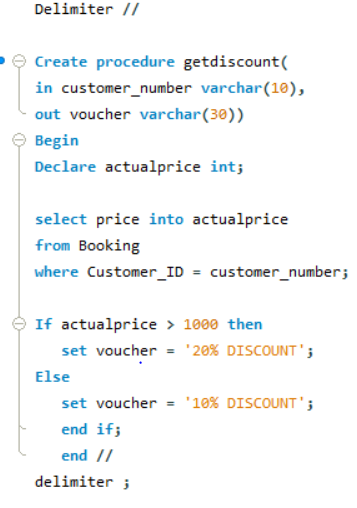
2.Show all the details of customer who booked for the film "Interstellar".

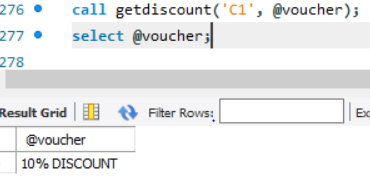




# Stored procedure:

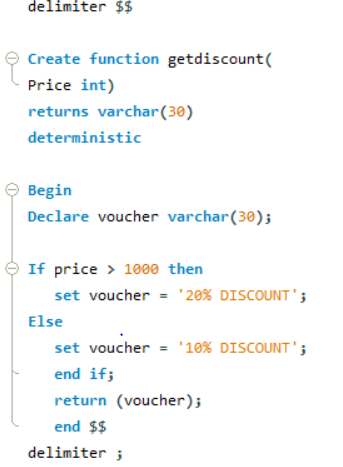
1.Giving voucher based on price.

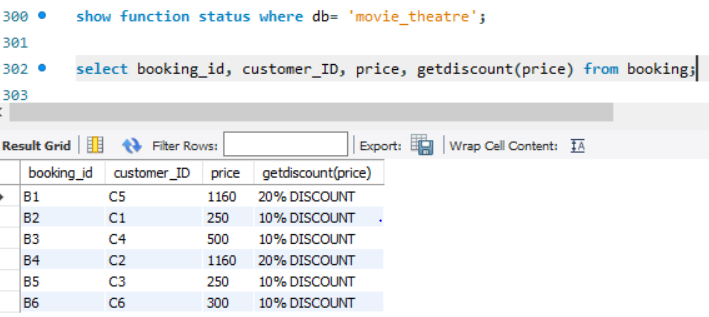




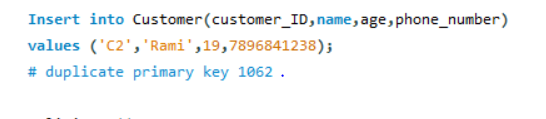
# Stored Function:

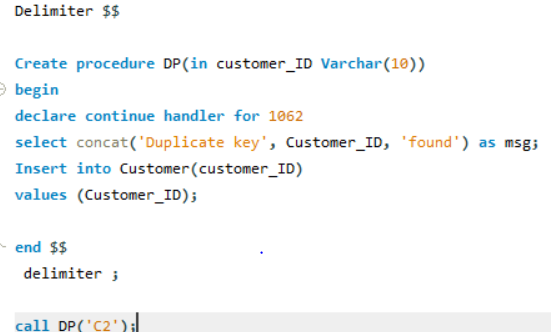
1.Show booking number, customer number, price and voucher details for each customer.

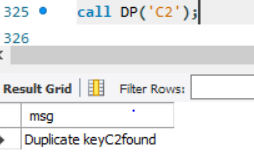




# Exception Handling

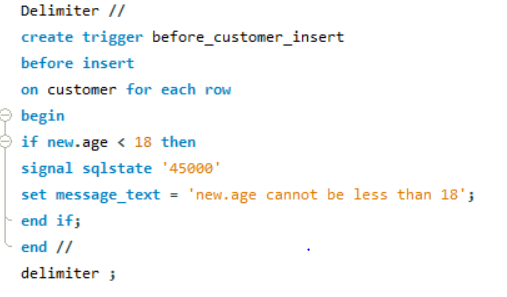


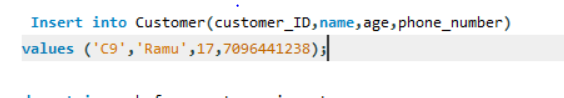




# Triggers:

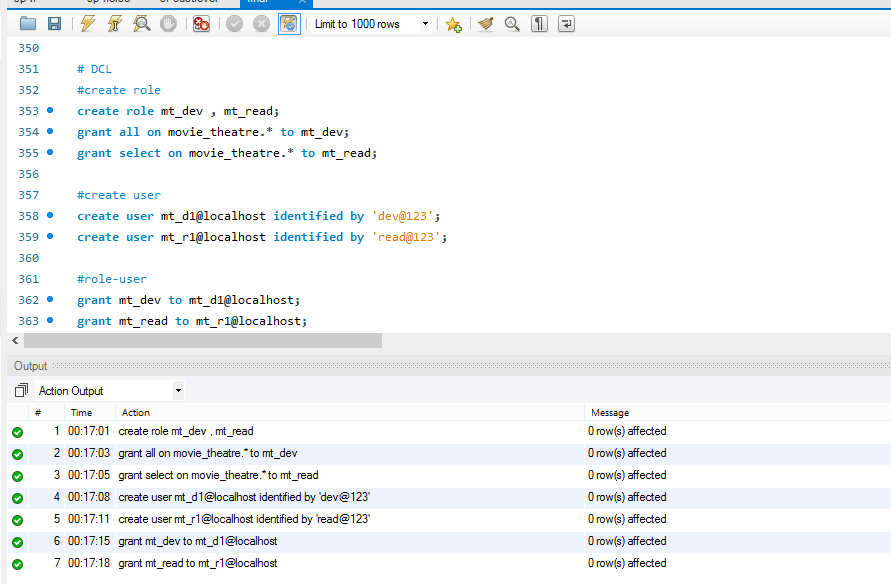
# Before Insert





Error message:



# DCL- creation of role, user. Grant role to the user

