Ticket Booking System

Tasks 1: Database Design:

1. **Create the database named "TicketBookingSystem**

create database TicketBookingSystem;

use TicketBookingSystem;

1. **Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.**

**• Venue • Event • Customers • Booking**

create table Venue(

venue\_id int not null primary key auto\_increment,

venue\_name varchar(30),

address varchar(50)

);

create table Event(

event\_id int not null primary key auto\_increment,

event\_name varchar(30) not null,

event\_date date,

event\_time time,

venue\_id int not null,

total\_seats int,

available\_seats int,

ticket\_price decimal,

event\_type varchar(30),

booking\_id int,

FOREIGN KEY (venue\_id) REFERENCES Venue(venue\_id)

);

create table Customer(

customer\_id int not null primary key auto\_increment,

customer\_name varchar(30) not null,

email varchar(50),

phone\_number varchar(10),

booking\_id int

);

create table Booking(

booking\_id int not null primary key auto\_increment,

customer\_id int not null,

event\_id int not null,

num\_tickets int,

total\_cost decimal,

booking\_date date,

FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id),

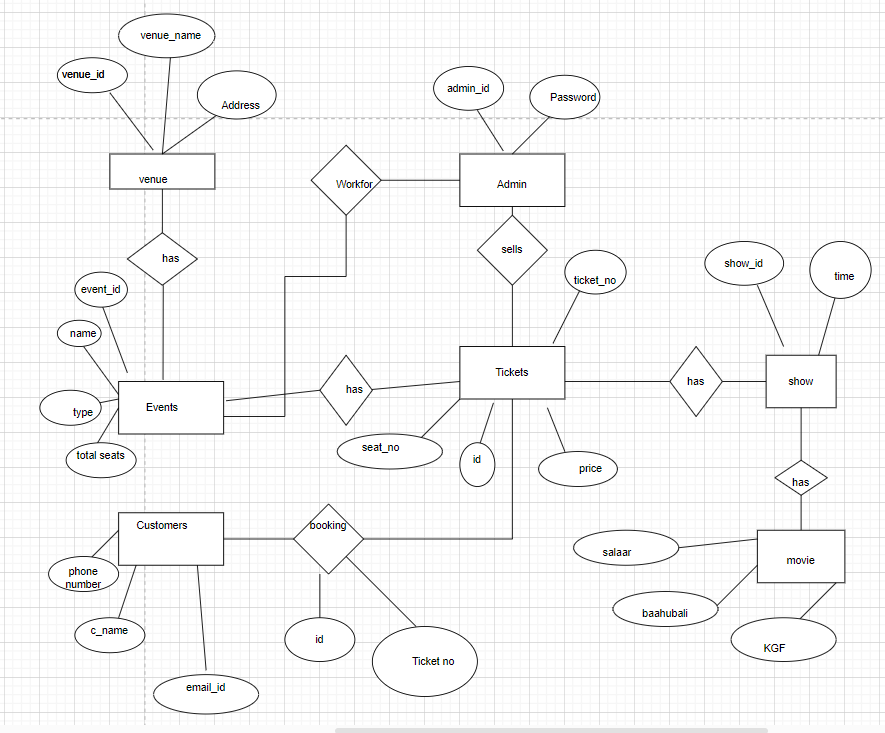
FOREIGN KEY (event\_id) REFERENCES Event(event\_id)

);

alter table Event Add FOREIGN KEY (booking\_id) REFERENCES Booking(booking\_id);

alter table Customer Add FOREIGN KEY (booking\_id) REFERENCES Booking(booking\_id);

1. **Create an ERD (Entity Relationship Diagram) for the database.**

****

1. **Create appropriate Primary Key and Foreign Key constraints for referential integrity.**

* Created Primary Key and Foreign Key constraints while creating tables.

**Tasks 2: Select, Where, Between, AND, LIKE:**

1. **Write a SQL query to insert at least 10 sample records into each table.**

a.Venue

insert into Venue(venue\_name,address)

values('Novotel','Hyderabad'),

('PVR','Chennai'),

('Taj Hotel','Mumbai'),

('Mysore Palace','Bangalore'),

('Park Hyatt','Hyderabad'),

('Inorbit mall','Chennai'),

('Taj Krishna','Hyderabad'),

('PVP','Vijayawada'),

('Novotel','Kolkata'),

('Prost','Bangalore');

b.Event

insert into Event(event\_name,event\_date,event\_time,venue\_id,total\_seats,available\_seats,ticket\_price,event\_type)

values('Hall of Game','2023-12-20','18:00:00',1,150,100,1000,'EDM Night'),

('Hi Nanna','2023-12-15','11:15:00',2,450,300,300,'Movie'),

('Show by Leni','2023-12-17','18:30:00',3,200,120,800,'Stand-Up Comedy'),

('Concert by Karthik','2023-12-15','19:00:00',4,300,150,1500,'Concert'),

('BDN','2023-12-19','18:00:00',5,150,100,500,'EDM Night'),

('Salaar','2023-12-22','11:15:00',6,450,0,300,'Movie'),

('Show by Emma','2023-12-12','18:30:00',7,300,100,800,'Concert'),

('Salaar','2023-12-22','19:00:00',8,300,150,300,'Movie'),

('SPB specials','2023-12-21','18:30:00',9,200,150,400,'Musical Concert'),

('Threeory','2023-12-23','19:00:00',10,300,150,1000,'Concert');

c.Customers

insert into Customer(customer\_name,email,phone\_number)

values('Gopi Reddy','gopireddy@gmail.com','9347717390'),

('Karan Reddy','karanreddy@gmail.com','8467584893'),

('Sathwik Reddy','sathwikreddy@gmail.com','9725364899'),

('Krishna Sai','krishnasai@gmail.com','7563456990'),

('Sai Krishna','saikrishna@gmail.com','6307717390'),

('Pavan Kumar','pavankumar@gmail.com','7463584893'),

('Gopi Krishna','gopikrishna@gmail.com','9438745369'),

('Bhargav','bhargav@gmail.com','7561234990'),

('Vamsi','vamsi@gmail.com','8372634455'),

('Sai teja','socalledsai@gmail.com','9100768621');

d.Booking

insert into Booking(customer\_id,event\_id,num\_tickets,total\_cost,booking\_date)

values(2,5,3,4500,'2023-12-05'),

(3,4,5,4000,'2023-12-09'),

(4,3,10,3000,'2023-12-06'),

(1,2,2,2000,'2023-12-08'),

(10,6,4,2000,'2023-12-06'),

(9,7,8,2400,'2023-12-10'),

(8,8,4,3200,'2023-12-08'),

(7,9,5,1500,'2023-12-03'),

(6,10,7,2800,'2023-12-05'),

(5,11,2,2000,'2023-12-08');

update Customer

SET booking\_id = 1

WHERE customer\_id=2;

update Customer

SET booking\_id = 2

WHERE customer\_id=3;

update Customer

SET booking\_id = 3

WHERE customer\_id=4;

update Customer

SET booking\_id = 4

WHERE customer\_id=1;

update Customer

SET booking\_id = 5

WHERE customer\_id=10;

update Customer

SET booking\_id = 6

WHERE customer\_id=9;

update Customer

SET booking\_id = 7

WHERE customer\_id=8;

update Customer

SET booking\_id = 8

WHERE customer\_id=7;

update Customer

SET booking\_id = 9

WHERE customer\_id=6;

update Customer

SET booking\_id = 10

WHERE customer\_id=5;

update Event

SET booking\_id = 1

WHERE event\_id=5;

update Event

SET booking\_id = 2

WHERE event\_id=4;

update Event

SET booking\_id = 3

WHERE event\_id=3;

update Event

SET booking\_id = 4

WHERE event\_id=2;

update Event

SET booking\_id = 5

WHERE event\_id=6;

update Event

SET booking\_id = 6

WHERE event\_id=7;

update Event

SET booking\_id =7

WHERE event\_id=8;

update Event

SET booking\_id =8

WHERE event\_id=9;

update Event

SET booking\_id =9

WHERE event\_id=10;

update Event

SET booking\_id = 10

WHERE event\_id=11;

1. **Write a SQL query to list all Events.**

SELECT \* FROM Event;

1. **Write a SQL query to select events with available tickets.**

select \* from Event where available\_seats<>0;

1. **Write a SQL query to select events name partial match with ‘cup’.**

select \* from Event where event\_name like '%cup%';

**5. Write a SQL query to select events with ticket price range is between 1000 to 2500**.

select \* from Event where ticket\_price between 1000 and 2500;

**6. Write a SQL query to retrieve events with dates falling within a specific range.**

select \* from Event where event\_date between '2023-12-15' and '2023-12-20';

**7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.**

select \* from Event where available\_seats<>0 and event\_name like '%Concert%';

**8. Write a SQL query to retrieve users in batches of 5, starting from the 6th user.**

select \* from Event limit 5 offset 5;

**9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.**

select \* from Booking where num\_tickets>4;

**10. Write a SQL query to retrieve customer information whose phone number end with ‘000’.**

select \* from Customer where phone\_number like '%000';

**11. Write a SQL query to retrieve the events in order whose seat capacity more than 15000.**

select \* from Event where total\_seats>15000 order by total\_seats;

**12. Write a SQL query to select events name not start with ‘x’, ‘y’, ‘z’.**

select \* from Event

where

event\_name not like 'x%'

and event\_name not like 'y%'

and event\_name not like 'z%';

**Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:**

1. **Write a SQL query to List Events and Their Average Ticket Prices**.

select event\_name,avg(ticket\_price) as average from Event group by event\_name;

1. **Write a SQL query to Calculate the Total Revenue Generated by Events**

select sum(total\_seats\*ticket\_price) as total\_revenue from Event;

1. **Write a SQL query to find the event with the highest ticket sales.**

select event\_id,event\_name, (total\_seats-available\_seats) AS highest\_sales

from Event

group by event\_id

order by highest\_sales desc

limit 1;

1. **Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.**

select event\_id,event\_name, (total\_seats-available\_seats) AS total\_tickets\_sold

from Event

group by event\_id;

1. **Write a SQL query to Find Events with No Ticket Sales.**

select \* from Event

where (total\_seats-available\_seats)=0;

1. **Write a SQL query to Find the User Who Has Booked the Most Tickets.**

select c.customer\_name,b.num\_tickets

from Customer c

join Booking b on c.booking\_id = b.booking\_id

order by b.num\_tickets desc

limit 1;

1. **Write a SQL query to List Events and the total number of tickets sold for each month**

select e.event\_name, year(b.booking\_date) as sales\_year, monthname(b.booking\_date) as sales\_month, sum(b.num\_tickets) as total\_tickets\_sold

from Event e

join Booking b on e.event\_id = b.event\_id

group by e.event\_name, sales\_year, sales\_month

order by sales\_year, sales\_month;

1. **Write a SQL query to calculate the average Ticket Price for Events in Each Venue.**

select v.venue\_id, v.venue\_name, avg(e.ticket\_price) as average\_ticket\_price

from Venue v

join Event e on v.venue\_id = e.venue\_id

group by v.venue\_id, v.venue\_name;

1. **Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.**

select event\_type, sum(total\_seats-available\_seats) as total\_tickets\_sold

from Event

group by event\_type;

**10.Write a SQL query to calculate the total Revenue Generated by Events in Each Year.**

select sum(e.total\_seats\*e.ticket\_price) as total\_revenue,YEAR(b.booking\_date) AS sales\_year

from Event e

join Booking b on e.event\_id = b.event\_id

group by sales\_year;

**11.Write a SQL query to list users who have booked tickets for multiple events.**

select c.customer\_id, c.customer\_name, count(distinct b.event\_id) as num\_events\_booked

from Customer c

join Booking b on c.customer\_id = b.customer\_id

group by c.customer\_id, c.customer\_name

having count(distinct b.event\_id) > 0;

**12. Write a SQL query to calculate the Total Revenue Generated by Events for Each User.**

select c.customer\_id, c.customer\_name, sum(b.total\_cost) as total\_revenue

from Customer c

join Booking b on c.customer\_id = b.customer\_id

group by c.customer\_id, c.customer\_name;

**13. Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.**

select v.venue\_id, v.venue\_name, e.event\_type, avg(e.ticket\_price) as average\_ticket\_price

from Venue v

join Event e on v.venue\_id = e.venue\_id

group by v.venue\_id, v.venue\_name, e.event\_type;

**14. Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days.**

select c.customer\_id, c.customer\_name, sum(b.num\_tickets) AS total\_tickets\_purchased

from Customer c

join Booking b on c.customer\_id = b.customer\_id

where b.booking\_date >= date\_sub(curdate(), interval 30 day)

group by c.customer\_id, c.customer\_name;

**Tasks 4: Subquery and its types:**

1. **Calculate the Average Ticket Price for Events in Each Venue Using a Subquery.**

select v.venue\_id, v.venue\_name, sub.avg\_ticket\_price

from Venue v, (

select e.venue\_id, avg(e.ticket\_price) as avg\_ticket\_price

from Event e

group by e.venue\_id

) as sub where v.venue\_id = sub.venue\_id;

1. **Find Events with More Than 50% of Tickets Sold using subquery.**

select event\_id,event\_name from Event where available\_seats<(select 0.5 \* total\_seats

from Event as e

where e.event\_id = Event.event\_id);

1. **Calculate the Total Number of Tickets Sold for Each Event.**

select event\_id, event\_name,

(select sum(num\_tickets) from Booking where Booking.event\_id = Event.event\_id) as total\_tickets\_sold

from Event;

1. **Find Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.**

select customer\_id, customer\_name

from Customer c

where not exists (

select 1

from Booking b

where b.customer\_id = c.customer\_id

);

1. **List Events with No Ticket Sales Using a NOT IN Subquery.**

select event\_id, event\_name

from Event

where event\_id not in (

select distinct event\_id

from Booking);

1. **Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause.**

select et.event\_type, sum(b.num\_tickets) as total\_tickets\_sold

from Event as et

join Booking b on et.event\_type =(

select event\_type

from Event

where Event.event\_id = b.event\_id

)

group by et.event\_type;

1. **Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause.**

select event\_id, event\_name, ticket\_price

from Event

where ticket\_price > (

select avg(ticket\_price)

from Event

);

1. **Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery.**

select c.customer\_id, c.customer\_name,

(select sum(b.total\_cost)

from Booking b

where b.customer\_id = c.customer\_id

) as total\_revenue

from Customer c;

1. **List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause.**

select customer\_id, customer\_name

from Customer

where customer\_id in (

select distinct b.customer\_id

from Booking b

join Event e on b.event\_id = e.event\_id

where e.venue\_id = 2

);

1. **Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY.**

select et.event\_type, et.total\_tickets\_sold

from (

select e.event\_type, sum(b.num\_tickets) as total\_tickets\_sold

from Event e

join Booking b on e.event\_id = b.event\_id

group by e.event\_type

) as et;

**12.Calculate the Average Ticket Price for Events in Each Venue Using a Subquery**

select v.venue\_id, v.venue\_name,

(select avg(e.ticket\_price)

from Event e

where e.venue\_id = v.venue\_id

) as average\_ticket\_price

from Venue v;