**J526 Yuvasri Ravikumar**

**SQL Assignment**

Ticket Booking System

**Task 1:**

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

mysql> create database TicketBookingSystem;

mysql> use TicketBookingSystem;

* 1. **2.Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.**
  + **Venu**
  + **Event**
  + **Customers**
  + **Booking**

1. create table Venu(venue\_id int primary key, venue\_name varchar(50), address varchar(80));
2. create table Event( event\_id int primary key, event\_name varchar(80), event\_date date, event\_time time, venue\_id int, total\_seats int, available\_seats int, ticket\_price decimal(10,2), event\_type enum('Movie', 'Sports', 'Concert'), booking\_id int,

constraint fk\_venue\_id FOREIGN KEY(venue\_id) REFERENCES Venu(venue\_id));

1. create table Customer(customer\_id int primary key, customer\_name varchar(50), email varchar(50), phone\_number BigInt);
2. create table Booking(booking\_id int primary key, customer\_id int, event\_id int, num\_tickets int, total\_cost decimal(10, 2), booking\_date date, constraint fk\_customer\_id FOREIGN KEY(customer\_id) REFERENCES Customer(customer\_id), constraint fk\_event\_id FOREIGN KEY(event\_id) REFERENCES Event(event\_id));

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

By using symbols

Oval -> attributes(double oval for multivalued attribute , dashed for derived )

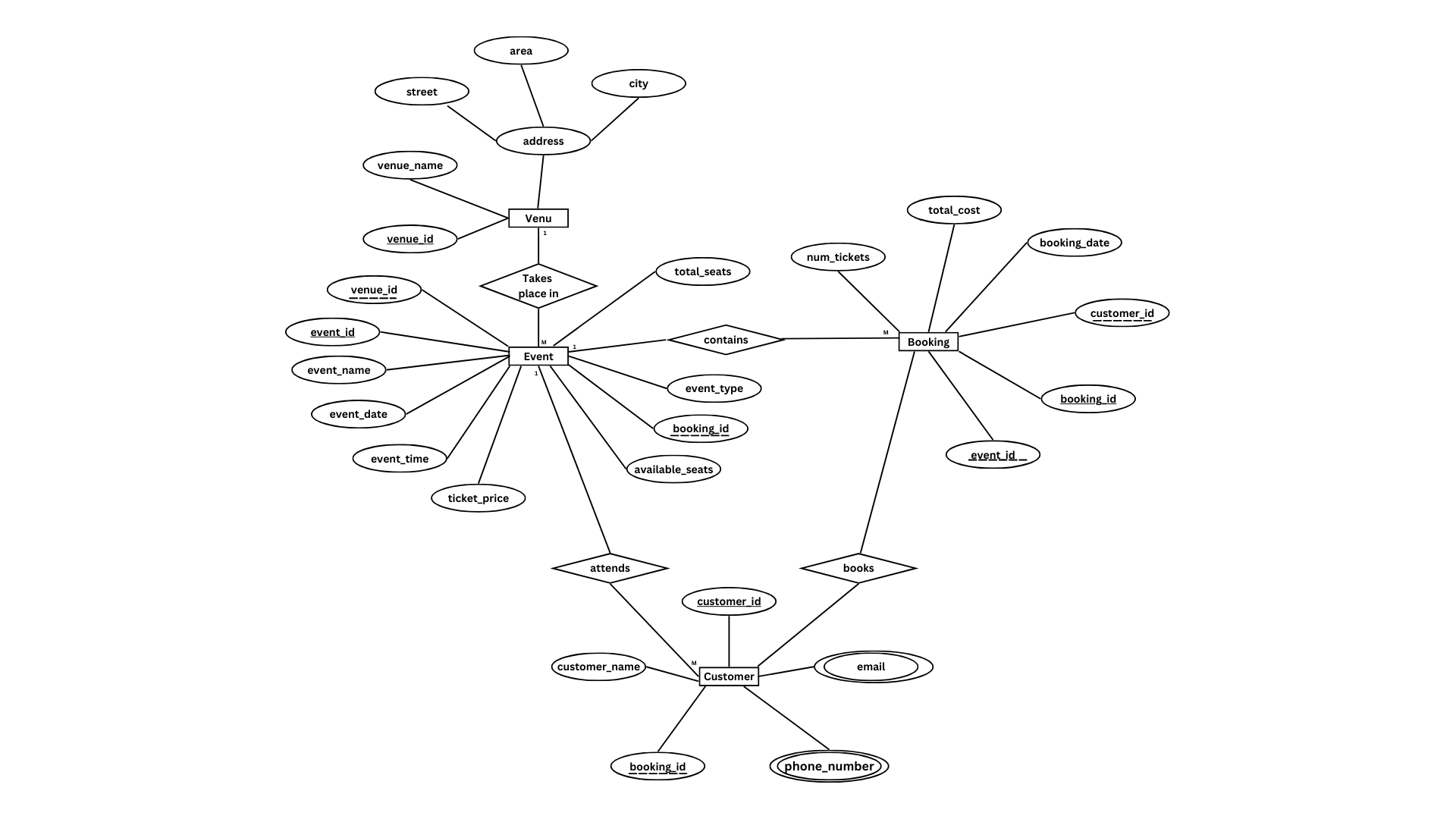
Rectangle -> entity type

Diamond -> relationship

Primary key -> underlined attribute

Foreign key -> dashed underlined attribute

And

Lines to show association between the elements, ERD is drawn to visually view the database.

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

Tables are created using appropriate primary and foreign keys to ensure referential as well as entity integrity.

alter table Customer add booking\_id int;

alter table Customer add constraint fk\_booking\_id FOREIGN KEY (booking\_id) REFERENCES Booking(booking\_id);

alter table Event add constraint fk\_book\_id FOREIGN KEY(booking\_id) REFERENCES Booking (booking\_id);

**Tasks 2:**

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

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

**Venu Table:**

Insert into Venu (venue\_id, venue\_name, address)

Values (11, 'Pine Forest Stadium', '123 Main Street, Ooty, TN'),

(22, 'Oaks And Fun', '4546 , Kalpana St, Kodaikanal, TN'),

(33, 'Girly Garden', '7589, Kamraj, Tirunelveli, TN'),

(44, 'Golden Beach', '1044, Tower Street, Trichy, TN'),

(55, 'Garden Hall', '202, Greenery area, Kanyakumari, TN'),

(66, 'Serene View', '75, Vetri Nagar, Chennai, TN'),

(77, 'Moon and Fun', '44 , TVK Nagar, Chennai, TN'),

(88, 'Candle light', '505,Perambur, Ooty, TN'),

(99, 'Juicy Chemistry', '696, Royal Nagar, Vellore, TN'),

(111, 'Ballet Arena', '77, Pine Glory, Kodaikanal, TN');

**Event Table:**

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

values

(1, 'Avatar', '2025-03-15', '18:30:00', 11, 200, 150, 1500.99, 'Movie'),

(2, 'BTS Concert', '2025-03-20', '22:00:00', 22, 500, 450, 2000.99, 'Concert'),

(3, 'World Cup Cricket', '2025-02-10', '19:30:00', 33, 300, 100, 1000.00, 'Sports'),

(4, 'Vijay Antony Concert', '2025-02-05', '21:00:00', 55, 150, 120, 3000.00, 'Concert'),

(5, 'Soccer World Cup', '2025-01-20', '18:00:00', 44, 800, 750, 1750.00, 'Sports'),

(6, 'Leo', '2025-01-22', '17:00:00', 66, 250, 200, 2440.50, 'Movie'),

(7, 'Anirudh Concert', '2025-03-18', '20:30:00', 77, 600, 550, 1220.00, 'Concert'),

(8, 'Tennis women', '2025-03-12', '16:00:00', 88, 400, 350,1180.00, 'Sports'),

(9, 'Conjuring', '2025-02-04', '22:00:00', 99, 300, 280, 1080.00, 'Movie'),

(10, 'Lana Del Rey', '2025-01-12', '22:30:00', 111, 200, 180, 4500.00, 'Concert');

**Booking Table:**

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

values

(101, 1, 20, 2500.98, '2025-03-15'),

(102, 2, 40, 3990.96, '2025-03-22'),

(103, 3, 30, 4050.00, '2025-02-10'),

(104, 4, 10, 3000.00, '2025-02-05'),

(105, 5, 50, 1500.00, '2025-01-20'),

(106, 6, 20, 3100.00, '2025-01-22'),

(107, 7, 30, 3600.00, '2025-03-18'),

(108, 8, 20, 1600.00, '2025-03-12'),

(109, 9, 40, 4502.00, '2025-02-04'),

(110, 10, 20, 1090.00, '2025-01-12');

**Customer Table:**

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

values

(22, 'Kalpana P', 'kalpana@gmail.com', 1234567890, 101),

(44, 'Stella T', 'stella@gmail.com', 1111111101, 102),

(75, 'Lolita Y', 'lolita@gmail.com', 2222222212, 103),

(46, 'Samantha S', 'Samantha@yahoo.com', 3333333123, 104),

(93, 'Harry P', 'harry@yahoo.com', 5678901234, 105),

(94, 'Vanilla T', 'vanilla@yahoo.com', 7777000000, 106),

(37, 'Geetha A', 'geetha@gmail.com', 1234567890, 107),

(68, 'Mary G', 'maryy@gmail.com', 8888880000, 108),

(47, 'Jaya', 'jaya@gmail.com', 9012345678, 109),

(90, 'Lalitha', 'lalitha@gmail.com', 1234500000, 110);

**To update booking\_id in Event Table;**

update Event set booking\_id=101 where event\_id=1;

update Event set booking\_id=102 where event\_id=2;

update Event set booking\_id=103 where event\_id=3;

update Event set booking\_id=104 where event\_id=4;

update Event set booking\_id=105 where event\_id=5;

update Event set booking\_id=106 where event\_id=6;

update Event set booking\_id=107 where event\_id=7;

update Event set booking\_id=108 where event\_id=8;

update Event set booking\_id=109 where event\_id=9;

update Event set booking\_id=110 where event\_id=10;

**To update booking\_id in Customer Table:**

update Customer set booking\_id=101 where customer\_id=22;

update Customer set booking\_id=102 where customer\_id=44;

update Customer set booking\_id=103 where customer\_id=75;

update Customer set booking\_id=104 where customer\_id=46;

update Customer set booking\_id=105 where customer\_id=93;

update Customer set booking\_id=106 where customer\_id=94;

update Customer set booking\_id=107 where customer\_id=37;

update Customer set booking\_id=108 where customer\_id=68;

update Customer set booking\_id=109 where customer\_id=47;

update Customer set booking\_id=110 where customer\_id=90;

**To update customer\_id in Booking Table:**

update Booking set customer\_id=22 where booking\_id = 101;

update Booking set customer\_id=44 where booking\_id = 102;

update Booking set customer\_id=75 where booking\_id = 103;

update Booking set customer\_id=46 where booking\_id = 104;

update Booking set customer\_id=93 where booking\_id = 105;

update Booking set customer\_id=94 where booking\_id = 106;

update Booking set customer\_id=37 where booking\_id = 107;

update Booking set customer\_id=68 where booking\_id = 108;

update Booking set customer\_id=47 where booking\_id = 109;

update Booking set customer\_id=90 where booking\_id = 110

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

mysql> select \* from Event;

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

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

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

mysql> select event\_id,event\_name from Event where event\_name like '%cup%' or event\_name like

'%Cup%';

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

mysql> select event\_id,event\_name from Event where ticket\_price between 1000.00 and 2500.00;

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

mysql> select event\_id,event\_name,event\_date from Event where event\_date between '2025-01-21' and '2025-03-10';

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

mysql> select event\_id,event\_name 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.**

mysql> select customer\_id,customer\_name from Customer limit 5 offset 5;

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

mysql> 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 cast(phone\_number as char) like '%000';

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

mysql> select event\_id,event\_name 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’**

mysql> select event\_name 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, Group By and Joins:**

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

mysql> select event\_id,event\_name,avg(ticket\_price) from Event group by event\_id,event\_name;

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

mysql> select e.event\_id,e.event\_name, sum(b.total\_cost) as total\_revenue from Event e join Booking b ON e.event\_id = b.event\_id group by event\_id,event\_name;

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

mysql> select e.event\_id,event\_name,sum(num\_tickets) as tickets\_sold from Event e join Booking b where e.event\_id = b.event\_id group by event\_id,event\_name order by tickets\_sold desc limit 1;

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

mysql> select e.event\_id,event\_name,sum(num\_tickets) as tickets\_sold from Event e join Booking b on e.booking\_id = b.booking\_id group by event\_id,event\_name;

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

mysql> select event\_id,event\_name from Event where not event\_id in (select distinct event\_id from Booking);

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

mysql> select c.customer\_id, customer\_name, sum(num\_tickets) as tickets\_booked from Customer c join Booking b on c.booking\_id = b.booking\_id group by c.customer\_id, customer\_name order by tickets\_booked desc limit 1;

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

mysql> select e.event\_id, event\_name,MONTH(event\_date) as month,sum(num\_tickets) as total\_tickets from Event e join Booking b on e.event\_id= b.event\_id group by e.event\_id, event\_name, MONTH(event\_date);

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

mysql> select event\_id,venue\_name,v.venue\_id,avg(ticket\_price) as average\_ticket\_price from Event e join Venu v on v.venue\_id = e.venue\_id group by venue\_name, event\_id;

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

mysql> select event\_type,sum(num\_tickets) as tickets\_sold from Event e join Booking b on b.booking\_id = e.booking\_id group by event\_type;

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

mysql> select year(event\_date) as Year,sum(total\_cost) as total\_revenue from Event e join Booking b on e.booking\_id = b.booking\_id group by year(event\_date);

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

mysql> select c.customer\_id,customer\_name, count(distinct event\_id) as events\_count from Booking b inner join Customer c on c.customer\_id = b.customer\_id group by c.customer\_id,customer\_name having events\_count > 1;

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

mysql> select c.customer\_id,customer\_name,sum(total\_cost)as total\_revenue from Customer c join Booking b on c.customer\_id = b.customer\_id group by c.customer\_id, customer\_name;

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

mysql> select event\_type,v.venue\_id, venue\_name,avg(ticket\_price) as average\_ticket\_price from Venu v join Event e on v.venue\_id=e.venue\_id group by event\_type,v.venue\_id,venue\_name;

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

mysql> select booking\_date, c.customer\_id,customer\_name ,sum(num\_tickets)as total\_tickets from Customer c join Booking b on c.customer\_id = b.customer\_id where booking\_date >= curdate() - INTERVAL 30 day group by booking\_date, c.customer\_id, customer\_name;

**Tasks 4:**

**Subquery and its types**

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

mysql> select v.venue\_id,venue\_name,(select avg(ticket\_price)from Event e where v.venue\_id = e.venue\_id) as average\_ticket\_price from Venu v ;

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

mysql> select event\_id,event\_name from Event where event\_id in(select event\_id from Event where (total\_seats - available\_seats) > (total\_seats\*0.5));

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

mysql> select event\_id,event\_name,(select sum(num\_tickets) from Booking b where b.event\_id = Event.event\_id) as total\_tickets from Event;

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

mysql> select customer\_id,customer\_name from Customer c where not exists(select 1 from Booking b where c.customer\_id=b.customer\_id);

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

mysql> select event\_id,event\_name from Event where event\_id not in(select event\_id from Booking);

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

mysql> select event\_type, sum(tickets\_sold) as tickets\_sold from (select event\_type, sum(num\_tickets) as tickets\_sold from Event e join Booking b on e.event\_id = b.event\_id group by event\_type) as ticket\_sold group by event\_type;

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

mysql> select event\_id,event\_name,event\_type,ticket\_price from Event where ticket\_price > (select avg(ticket\_price) from Event);

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

mysql> select c.customer\_id, c.customer\_name,(select sum(total\_cost) from Booking b where c.customer\_id = b.customer\_id) as tot\_revenue from Customer c;

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

mysql> select c.customer\_id,c.customer\_name from Customer c where customer\_id in (select b.customer\_id from Booking b join Event e on e.event\_id = b.event\_id where e.venue\_id = 77);

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

mysql> select event\_type,(select sum(num\_tickets) from Booking b where b.event\_id = e.event\_id) as tickets\_sold from Event e group by event\_type,e.event\_id;

**11. Find Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE\_FORMAT.**

mysql> select DATE\_FORMAT(b.booking\_date,'%m/%y') as Booked\_month,c.customer\_id,c.customer\_name from Booking b join Customer c where b.customer\_id = c.customer\_id;

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

mysql> select a.venue\_id,v.venue\_name,(select avg(b.ticket\_price) from Event b where a.venue\_id=b.venue\_id) as average\_price\_per\_venue from Event a join Venu v on a.venue\_id=v.venue\_id;