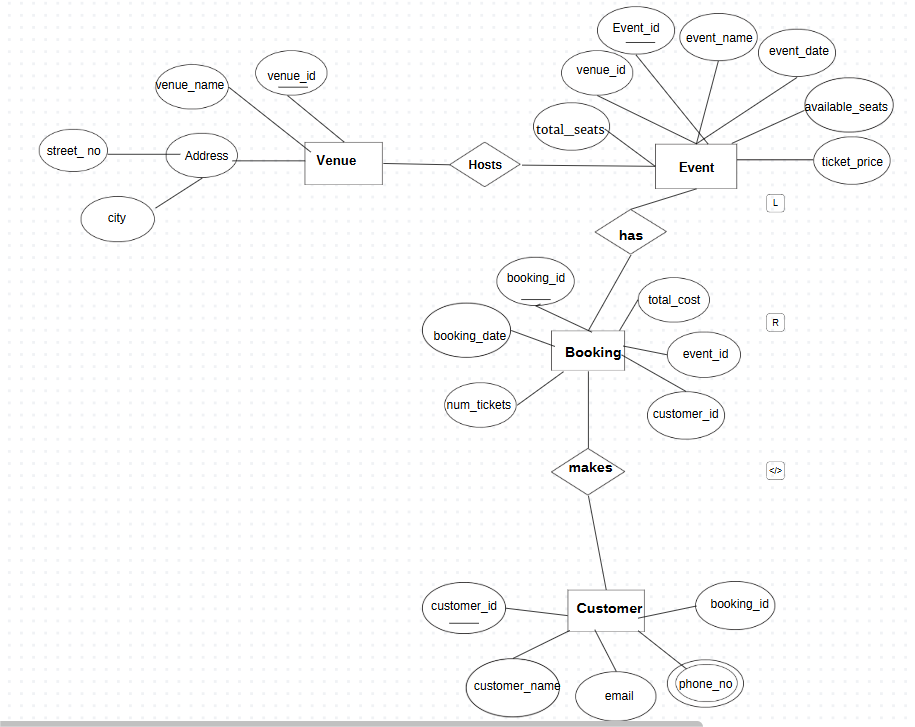
1. R DIAGRAM :



Task 1 :

CREATE TABLE Venue (

venue\_id INT PRIMARY KEY AUTO\_INCREMENT,

venue\_name VARCHAR(255) NOT NULL,

address TEXT NOT NULL

);

CREATE TABLE Event (

event\_id INT PRIMARY KEY AUTO\_INCREMENT,

event\_name VARCHAR(255) NOT NULL,

event\_date DATE NOT NULL,

event\_time TIME NOT NULL,

venue\_id INT NOT NULL,

total\_seats INT NOT NULL,

available\_seats INT NOT NULL,

ticket\_price DECIMAL(10,2) NOT NULL,

event\_type ENUM('Movie', 'Sports', 'Concert') NOT NULL,

booking\_id INT,

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

FOREIGN KEY (booking\_id) REFERENCES Booking(booking\_id)

);

CREATE TABLE Customer (

customer\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

phone\_number VARCHAR(15) UNIQUE NOT NULL,

booking\_id INT,

FOREIGN KEY (booking\_id) REFERENCES Booking(booking\_id)

);

CREATE TABLE Booking (

booking\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT NOT NULL,

event\_id INT NOT NULL,

num\_tickets INT NOT NULL,

total\_cost DECIMAL(10,2) NOT NULL,

booking\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (customer\_id) REFERENCES Customer1(customer\_id),

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

);

Task 2:

1)--Insert 10 sample records into Venue table

INSERT INTO Venue (venue\_id ,venue\_name, address) VALUES

(1,'Stadium A', '123 Main St'),

(2,'Concert Hall B', '456 Broadway'),

(3,'Arena C', '789 Park Ave'),

(4,'Theater D', '101 Elm St'),

(5,'Sports Complex E', '202 Maple St'),

(6,'Cinema F', '303 Oak St'),

(7,'Outdoor Venue G', '404 Pine St'),

(8,'Convention Center H', '505 Birch St'),

(9,'Amphitheater I', '606 Cedar St'),

(10,'Expo Center J', '707 Walnut St');

-- Insert 10 sample records into 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,'Football Cup', '2025-07-10', '18:00:00', 1, 20000, 5000, 1500.00, 'Sports'),

(2,'Rock Concert', '2024-08-15', '20:00:00', 2, 10000, 3000, 2000.00, 'Concert'),

(3,'Basketball Finals', '2025-09-20', '19:30:00', 3, 15000, 7000, 1800.00, 'Sports'),

(4,'Drama Play', '2023-06-05', '17:00:00', 4, 5000, 2000, 1200.00, 'Movie'),

(5,'Cricket Match', '2025-10-10', '15:00:00', 5, 25000, 10000, 2200.00, 'Sports'),

(6,'Jazz Night', '2022-07-25', '21:00:00', 6, 8000, 2000, 2500.00, 'Concert'),

(7,'Movie Premiere', '2023-05-12', '19:00:00', 7, 3000, 1000, 900.00, 'Movie'),

(8,'Tech Conference', '2025-11-18', '09:00:00', 8, 5000, 2500, 500.00, 'Concert'),

(9,'Music Festival', '2025-09-30', '16:00:00', 9, 12000, 6000, 1750.00, 'Concert'),

(10,'Esports Championship', '2024-12-05', '14:00:00', 10, 18000, 8000, 2000.00, 'Sports');

-- Insert 10 sample records into Customer table

INSERT INTO Customer1 (customer\_id,customer\_name, email, phone\_number, booking\_id) VALUES

(1,'John Doe', 'john@example.com', '9876543210', 1),

(2,'Jane Smith', 'jane@example.com', '9876543220', 2),

(3,'Alice Johnson', 'alice@example.com', '9876543230', 3),

(4,'Bob Brown', 'bob@example.com', '9876543240', 4),

(5,'Charlie White', 'charlie@example.com', '9876543250', 5),

(6,'David Black', 'david@example.com', '9876543260', 6),

(7,'Eve Green', 'eve@example.com', '9876543270', 7),

(8,'Frank Blue', 'frank@example.com', '9876543280', 8),

(9,'Grace Yellow', 'grace@example.com', '9876543290', 9),

(10,'Henry Red', 'henry@example.com', '9876543200', 10);

-- Insert 10 sample records into Booking table

INSERT INTO Booking (booking\_id,customer\_id, event\_id, num\_tickets, total\_cost, booking\_date) VALUES

(1,1, 1, 2, 3000.00, NOW()),

(2,2, 2, 3, 6000.00, NOW()),

(3,3, 3, 1, 1800.00, NOW()),

(4,4, 4, 5, 6000.00, NOW()),

(5,5, 5, 4, 8800.00, NOW()),

(6,6, 6, 2, 5000.00, NOW()),

(7,7, 7, 3, 2700.00, NOW()),

(8,8, 8, 1, 500.00, NOW()),

(9,9, 9, 2, 3500.00, NOW()),

(10,10, 10, 6, 12000.00, NOW());

2)-- Query to list all Events

SELECT \* FROM Event;

3)-- Query to select events with available tickets

SELECT \* FROM Event WHERE available\_seats > 0;

4)-- Query to select events with partial match 'cup'

SELECT \* FROM Event WHERE event\_name LIKE '%cup%';

5)-- Query to select events with ticket price between 1000 and 2500

SELECT \* FROM Event WHERE ticket\_price BETWEEN 1000 AND 2500;

6)-- Query to retrieve events within a date range

SELECT \* FROM Event WHERE event\_date BETWEEN '2025-07-01' AND '2025-12-31';

7)-- Query to retrieve events with available tickets and 'Concert' in the name

SELECT \* FROM Event WHERE available\_seats > 0 AND event\_name LIKE '%Concert%';

8)-- Query to retrieve users in batches of 5 starting from 6th user

SELECT \* FROM Customer1 LIMIT 5 OFFSET 5;

9)-- Query to retrieve bookings where num\_tickets > 4

SELECT \* FROM Booking WHERE num\_tickets > 4;

10)-- Query to retrieve customers whose phone number ends with '000'

SELECT \* FROM Customer WHERE phone\_number LIKE '%000';

11)-- Query to retrieve events ordered by seat capacity more than 15000

SELECT \* FROM Event WHERE total\_seats > 15000 ORDER BY total\_seats DESC;

12)-- Query to select events name not starting 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%';

TASK 3:

1)-- List Events and Their Average Ticket Prices

SELECT event\_name, AVG(ticket\_price) AS avg\_price FROM Event GROUP BY event\_name;

2)-- Calculate the Total Revenue Generated by Events

SELECT event\_name, SUM(total\_cost) AS total\_revenue FROM Booking JOIN Event ON Booking.event\_id = Event.event\_id GROUP BY event\_name;

3)-- Find the event with the highest ticket sales

SELECT event\_name, SUM(num\_tickets) AS total\_tickets\_sold FROM Booking JOIN Event ON Booking.event\_id = Event.event\_id GROUP BY event\_name ORDER BY total\_tickets\_sold DESC LIMIT 1;

4)-- Calculate the Total Number of Tickets Sold for Each Event

SELECT event\_name, SUM(num\_tickets) AS total\_tickets FROM Booking JOIN Event ON Booking.event\_id = Event.event\_id GROUP BY event\_name;

5)-- Find Events with No Ticket Sales

SELECT event\_name FROM Event WHERE event\_id NOT IN (SELECT DISTINCT event\_id FROM Booking);

6)-- Find the User Who Has Booked the Most Tickets

SELECT customer\_name, SUM(num\_tickets) AS total\_tickets FROM Booking JOIN Customer ON Booking.customer\_id = Customer.customer\_id GROUP BY customer\_name ORDER BY total\_tickets DESC LIMIT 1;

7)-- List Events and the total number of tickets sold for each month

SELECT MONTH(event\_date) AS month, event\_name, SUM(num\_tickets) AS total\_tickets FROM Booking JOIN Event ON Booking.event\_id = Event.event\_id GROUP BY month, event\_name;

8)-- Calculate the average Ticket Price for Events in Each Venue

SELECT venue\_name, AVG(ticket\_price) AS avg\_ticket\_price FROM Event JOIN Venue ON Event.venue\_id = Venue.venue\_id GROUP BY venue\_name;

9)-- Calculate the total Number of Tickets Sold for Each Event Type

SELECT event\_type, SUM(num\_tickets) AS total\_tickets FROM Booking JOIN Event ON Booking.event\_id = Event.event\_id GROUP BY event\_type;

10)-- Calculate the total Revenue Generated by Events in Each Year

SELECT YEAR(event\_date) AS year, SUM(total\_cost) AS total\_revenue FROM Booking JOIN Event ON Booking.event\_id = Event.event\_id GROUP BY year;

11)-- List users who have booked tickets for multiple events

SELECT customer\_name, COUNT(DISTINCT event\_id) AS event\_count FROM Booking JOIN Customer ON Booking.customer\_id = Customer.customer\_id GROUP BY customer\_name HAVING event\_count > 1;

12)-- Calculate the Total Revenue Generated by Events for Each User

SELECT customer\_name, SUM(total\_cost) AS total\_revenue FROM Booking JOIN Customer ON Booking.customer\_id = Customer.customer\_id GROUP BY customer\_name;

13)-- Calculate the Average Ticket Price for Events in Each Category and Venue

SELECT event\_type, venue\_name, AVG(ticket\_price) AS avg\_price FROM Event JOIN Venue ON Event.venue\_id = Venue.venue\_id GROUP BY event\_type, venue\_name;

14)-- List Users and the Total Number of Tickets They've Purchased in the Last 30 Days

SELECT customer\_name, SUM(num\_tickets) AS total\_tickets FROM Booking JOIN Customer ON Booking.customer\_id = Customer.customer\_id WHERE booking\_date >= NOW() - INTERVAL 30 DAY GROUP BY customer\_name;

TASK 4:

1)-- Calculate the Average Ticket Price for Events in Each Venue Using a Subquery

SELECT venue\_name, (SELECT AVG(ticket\_price) FROM Event WHERE Event.venue\_id = Venue.venue\_id) AS avg\_ticket\_price FROM Venue;

2)-- Find Events with More Than 50% of Tickets Sold using subquery

SELECT event\_name FROM Event WHERE total\_seats - available\_seats > total\_seats \* 0.5;

3)-- Calculate the Total Number of Tickets Sold for Each Event

SELECT event\_name, (SELECT SUM(num\_tickets) FROM Booking WHERE Booking.event\_id = Event.event\_id) AS total\_tickets FROM Event;

4)-- Find Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery

SELECT customer\_name FROM Customer WHERE NOT EXISTS (SELECT 1 FROM Booking WHERE Booking.customer\_id = Customer.customer\_id);

5)-- List Events with No Ticket Sales Using a NOT IN Subquery

SELECT event\_name FROM Event WHERE event\_id NOT IN (SELECT DISTINCT event\_id FROM Booking);

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

SELECT event\_type, SUM(total\_tickets) AS total\_tickets FROM (SELECT event\_type, SUM(num\_tickets) AS total\_tickets FROM Booking JOIN Event ON Booking.event\_id = Event.event\_id GROUP BY event\_type) AS ticket\_counts GROUP BY event\_type;

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

SELECT event\_name 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

SELECT customer\_name, (SELECT SUM(total\_cost) FROM Booking WHERE Booking.customer\_id = Customer.customer\_id) AS total\_revenue FROM Customer;

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

SELECT customer\_name FROM Customer WHERE customer\_id IN (SELECT customer\_id FROM Booking WHERE event\_id IN (SELECT event\_id FROM Event WHERE venue\_id = 1));

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

SELECT event\_type, (SELECT SUM(num\_tickets) FROM Booking WHERE Booking.event\_id = Event.event\_id) AS total\_tickets FROM Event GROUP BY event\_type;

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

SELECT customer\_name, DATE\_FORMAT(booking\_date, '%Y-%m') AS booking\_month FROM Booking JOIN Customer ON Booking.customer\_id = Customer.customer\_id GROUP BY customer\_name, booking\_month;

12)-- Calculate the Average Ticket Price for Events in Each Venue Using a Subquery

SELECT venue\_name, (SELECT AVG(ticket\_price) FROM Event WHERE Event.venue\_id = Venue.venue\_id) AS avg\_ticket\_price FROM Venue;