CREATE DATABASE TicketBookingSystem;

USE TicketBookingSystem;

CREATE TABLE Venue (

venue\_id INT PRIMARY KEY AUTO\_INCREMENT,

venue\_name VARCHAR(100) NOT NULL,

address VARCHAR(255) NOT NULL

);

CREATE TABLE Customer (

customer\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

phone\_number VARCHAR(15) NOT NULL

);

CREATE TABLE Event (

event\_id INT PRIMARY KEY AUTO\_INCREMENT,

event\_name VARCHAR(100) 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,

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

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

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

);

ALTER TABLE Customer

ADD COLUMN booking\_id INT,

ADD CONSTRAINT fk\_customer\_booking

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

ALTER TABLE Event

ADD COLUMN booking\_id INT,

ADD CONSTRAINT fk\_event\_booking

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

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

(1, 'City Arena', 'New Street 101'),

(2, 'Downtown Theater', 'Main Road 45'),

(3, 'Riverfront Stadium', 'Lake View 23'),

(4, 'Grand Concert Hall', 'Elm Street 55'),

(5, 'Skyline Center', 'Hill Road 78'),

(6, 'Royal Grounds', 'Garden Avenue 12'),

(7, 'Open Air Theatre', 'Park Street 99'),

(8, 'Metro Dome', 'Underground Blvd 7'),

(9, 'Sunset Arena', 'Beach Road 88'),

(10, 'Galaxy Venue', 'Star Lane 10');

INSERT INTO Event (event\_id, event\_name, event\_date, event\_time, venue\_id, total\_seats, available\_seats, ticket\_price, event\_type) VALUES

(1, 'Champions Cup', '2025-05-01', '18:00:00', 1, 5000, 2000, 1500.00, 'Sports' ),

(2, 'Rock Concert', '2025-06-12', '20:00:00', 4, 3000, 500, 2000.00, 'Concert' ),

(3, 'Film Premiere', '2025-04-15', '19:30:00', 2, 1000, 50, 500.00, 'Movie'),

(4, 'Summer Cup Finals', '2025-05-30', '16:00:00', 3, 4000, 3000, 1200.00, 'Sports' ),

(5, 'Jazz Night', '2025-07-01', '21:00:00', 5, 1500, 1000, 1800.00, 'Concert'),

(6, 'Drama Play', '2025-08-20', '17:30:00', 2, 800, 200, 800.00, 'Movie'),

(7, 'Comedy Show', '2025-09-10', '18:30:00', 8, 1000, 700, 1000.00, 'Movie'),

(8, 'Legends Concert', '2025-10-05', '20:00:00', 4, 2000, 0, 2500.00, 'Concert'),

(9, 'Final Match Cup', '2025-04-20', '15:00:00', 3, 6000, 500, 2300.00, 'Sports'),

(10, 'Retro Music Night', '2025-11-22', '19:00:00', 9, 2500, 1500, 1300.00, 'Concert');

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

(1, 'Alice Smith', 'alice@example.com', '1234567890'),

(2, 'Bob Jones', 'bob@example.com', '2345678901'),

(3, 'Carol Lee', 'carol@example.com', '3456789012'),

(4, 'David Kim', 'david@example.com', '4567890123'),

(5, 'Eva Green', 'eva@example.com', '5678901234'),

(6, 'Frank White', 'frank@example.com', '6789012345'),

(7, 'Grace Hall', 'grace@example.com', '7890123456'),

(8, 'Henry Wood', 'henry@example.com', '8901234567'),

(9, 'Ivy Turner', 'ivy@example.com', '9012345678'),

(10, 'Jake Black', 'jake@example.com', '0123456789');

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

(1, 1, 1, 2, 3000.00, '2025-03-25'),

(2, 2, 2, 5, 10000.00, '2025-04-01'),

(3, 3, 4, 1, 1200.00, '2025-03-28'),

(4, 4, 5, 3, 5400.00, '2025-04-10'),

(5, 5, 6, 6, 4800.00, '2025-04-12'),

(6, 6, 7, 2, 2000.00, '2025-04-15'),

(7, 7, 8, 1, 2500.00, '2025-04-18'),

(8, 8, 9, 8, 18400.00, '2025-04-20'),

(9, 9, 10, 4, 5200.00, '2025-04-22'),

(10, 10, 3, 1, 500.00, '2025-04-25');

SELECT \* FROM Customer

SELECT \* FROM Event

SELECT \*

FROM Event

WHERE available\_seats > 0;

SELECT \*

FROM Event

WHERE event\_name LIKE '%cup%';

SELECT \*

FROM Event

WHERE ticket\_price BETWEEN 1000 AND 2500;

SELECT \*

FROM Event

WHERE event\_date BETWEEN '2025-04-20' AND '2025-05-01';

SELECT \*

FROM Event

WHERE available\_seats > 0

AND event\_name LIKE '%Concert%';

SELECT \*

FROM Customer

LIMIT 5 OFFSET 5;

SELECT \*

FROM Booking

WHERE num\_tickets > 4;

SELECT \*

FROM Customer

WHERE phone\_number LIKE '%000';

SELECT \*

FROM Event

WHERE total\_seats > 15000

ORDER BY total\_seats;

SELECT \*

FROM Event

WHERE LOWER(event\_name) NOT LIKE 'x%'

AND LOWER(event\_name) NOT LIKE 'y%'

AND LOWER(event\_name) NOT LIKE 'z%';

SELECT event\_name, AVG(ticket\_price) AS average\_price

FROM Event

GROUP BY event\_name;

SELECT e.event\_name, SUM(b.total\_cost) AS total\_revenue

FROM Booking b

JOIN Event e ON b.event\_id = e.event\_id

GROUP BY e.event\_name;

SELECT e.event\_name, SUM(b.num\_tickets) AS tickets\_sold

FROM Booking b

JOIN Event e ON b.event\_id = e.event\_id

GROUP BY e.event\_name

ORDER BY tickets\_sold DESC

LIMIT 1;

SELECT e.event\_name, SUM(b.num\_tickets) AS total\_tickets

FROM Booking b

JOIN Event e ON b.event\_id = e.event\_id

GROUP BY e.event\_name;

SELECT e.event\_name

FROM Event e

LEFT JOIN Booking b ON e.event\_id = b.event\_id

WHERE b.booking\_id IS NULL;

SELECT

DATE\_FORMAT(e.event\_date, '%Y-%m') AS month,

e.event\_name,

SUM(b.num\_tickets) AS total\_tickets

FROM Booking b

JOIN Event e ON b.event\_id = e.event\_id

GROUP BY month, e.event\_name

ORDER BY month;

SELECT v.venue\_name, AVG(e.ticket\_price) AS avg\_price

FROM Event e

JOIN Venue v ON e.venue\_id = v.venue\_id

GROUP BY v.venue\_name;

SELECT e.event\_type, SUM(b.num\_tickets) AS total\_tickets

FROM Booking b

JOIN Event e ON b.event\_id = e.event\_id

GROUP BY e.event\_type;

SELECT YEAR(e.event\_date) AS year, SUM(b.total\_cost) AS total\_revenue

FROM Booking b

JOIN Event e ON b.event\_id = e.event\_id

GROUP BY year

ORDER BY year;

SELECT c.customer\_name, COUNT(DISTINCT b.event\_id) AS unique\_events

FROM Booking b

JOIN Customer c ON b.customer\_id = c.customer\_id

GROUP BY c.customer\_name

HAVING unique\_events > 1;

SELECT c.customer\_name, SUM(b.total\_cost) AS total\_spent

FROM Booking b

JOIN Customer c ON b.customer\_id = c.customer\_id

GROUP BY c.customer\_name;

SELECT e.event\_type, v.venue\_name, AVG(e.ticket\_price) AS avg\_price

FROM Event e

JOIN Venue v ON e.venue\_id = v.venue\_id

GROUP BY e.event\_type, v.venue\_name;

SELECT c.customer\_name, SUM(b.num\_tickets) AS tickets\_purchased

FROM Booking b

JOIN Customer c ON b.customer\_id = c.customer\_id

WHERE b.booking\_date >= CURDATE() - INTERVAL 30 DAY

GROUP BY c.customer\_name;

SELECT v.venue\_name, avg\_prices.avg\_price

FROM Venue v

JOIN (

SELECT venue\_id, AVG(ticket\_price) AS avg\_price

FROM Event

GROUP BY venue\_id

) AS avg\_prices ON v.venue\_id = avg\_prices.venue\_id;

SELECT event\_name

FROM Event

WHERE (total\_seats - available\_seats) > (total\_seats \* 0.5);

SELECT event\_name,

(SELECT SUM(num\_tickets)

FROM Booking

WHERE Booking.event\_id = Event.event\_id) AS total\_tickets\_sold

FROM Event;

SELECT \*

FROM Customer c

WHERE NOT EXISTS (

SELECT 1

FROM Booking b

WHERE b.customer\_id = c.customer\_id

);

SELECT \*

FROM Event

WHERE event\_id NOT IN (

SELECT DISTINCT event\_id FROM Booking

);

SELECT sub.event\_type, SUM(sub.num\_tickets) AS total\_tickets

FROM (

SELECT e.event\_type, b.num\_tickets

FROM Booking b

JOIN Event e ON b.event\_id = e.event\_id

) AS sub

GROUP BY sub.event\_type;

SELECT \*

FROM Event

WHERE ticket\_price > (

SELECT AVG(ticket\_price) FROM Event

);

SELECT customer\_name,

(SELECT SUM(total\_cost)

FROM Booking b

WHERE b.customer\_id = c.customer\_id) AS total\_revenue

FROM Customer c;

SELECT \*

FROM Customer

WHERE customer\_id IN (

SELECT b.customer\_id

FROM Booking b

JOIN Event e ON b.event\_id = e.event\_id

WHERE e.venue\_id = 1

);

SELECT event\_type, SUM(total\_tickets) AS total\_sold

FROM (

SELECT e.event\_type, b.num\_tickets AS total\_tickets

FROM Booking b

JOIN Event e ON b.event\_id = e.event\_id

) AS sub

GROUP BY event\_type;

SELECT DISTINCT customer\_name

FROM Customer c

WHERE customer\_id IN (

SELECT b.customer\_id

FROM Booking b

WHERE DATE\_FORMAT(b.booking\_date, '%Y-%m') IS NOT NULL

);

SELECT v.venue\_name,

(SELECT AVG(ticket\_price)

FROM Event e

WHERE e.venue\_id = v.venue\_id) AS avg\_ticket\_price

FROM Venue v;