Project- Ticket Booking System

Code:

```
create database learnings5;
use learnings5;

-- Create Venu Table
CREATE TABLE Venu (
    venue_id INT AUTO_INCREMENT PRIMARY KEY,
    venue_name VARCHAR(255),
    address VARCHAR(255)
);

-- Create Event Table
CREATE TABLE Event (
    event_id INT AUTO_INCREMENT PRIMARY KEY,
    event_name VARCHAR(255),
    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,
    FOREIGN KEY (venue_id) REFERENCES Venu(venue_id),
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
);

-- Create Customer Table
CREATE TABLE Customer (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_name VARCHAR(255),
    email VARCHAR(255),
    phone_number VARCHAR(20),
    booking_id INT,
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
);

-- Create Booking Table
CREATE TABLE Booking (
    booking_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    event_id INT,
    num_tickets INT,
    total_cost DECIMAL(10, 2),
```

```
    booking_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (event_id) REFERENCES Event(event_id)
);


INSERT INTO Venu (venue_name, address) VALUES
    ('Sri Sai Convention Hall', '123 ABC Street, Chennai'),
    ('Lakshmi Auditorium', '456 XYZ Road, Bangalore'),
    ('Ganesh Marriage Hall', '789 PQR Avenue, Hyderabad'),
    ('Krishna Function Hall', '101 JKLM Lane, Kochi'),
    ('Annapurna Kalyana Mandapam', '321 DEF Road, Chennai'),
    ('Chaitanya Event Center', '654 GHI Street, Bangalore'),
    ('Meenakshi Marriage Hall', '987 RST Avenue, Hyderabad'),
    ('Saravana Thirumana Maligai', '147 MNO Lane, Kochi'),
    ('Raghavendra Kalyana Mandapam', '369 UVW Road, Chennai'),
    ('Vasavi Convention Center', '753 IJK Street, Bangalore');

INSERT INTO Event (event_name, event_date, event_time, venue_id, total_seats, available_seats,
ticket_price, event_type) VALUES
    ('South Indian Music Night', '2024-03-10', '19:00:00', 1, 200, 200, 50.00, 'Concert'),
    ('Kerala Cultural Show', '2024-03-15', '18:30:00', 4, 150, 150, 75.00, 'Concert'),
    ('Bangalore IPL Match', '2024-03-20', '20:00:00', 2, 500, 500, 100.00, 'Sports'),
    ('Tamil Movie Premiere', '2024-03-25', '17:00:00', 5, 300, 300, 80.00, 'Movie'),
    ('Hyderabad Comedy Show', '2024-04-01', '21:00:00', 3, 100, 100, 60.00, 'Concert'),
    ('Kochi Food Festival', '2024-04-05', '12:00:00', 8, 200, 200, 30.00, 'Concert'),
    ('Chennai Marathon', '2024-04-10', '06:00:00', 9, 1000, 1000, 20.00, 'Sports'),
    ('Bangalore Book Fair', '2024-04-15', '10:00:00', 6, 300, 300, 40.00, 'Concert'),
    ('Hyderabad Dance Competition', '2024-04-20', '15:30:00', 7, 200, 200, 70.00, 'Concert'),
    ('Kochi Art Exhibition', '2024-04-25', '11:00:00', 10, 150, 150, 25.00, 'Concert');

INSERT INTO Customer (customer_name, email, phone_number) VALUES
    ('Arjun Kumar', 'arjun@example.com', '9876543210'),
    ('Divya Nair', 'divya@example.com', '8765432109'),
    ('Gowri Shankar', 'gowri@example.com', '7654321098'),
    ('Hari Prasad', 'hari@example.com', '6543210987'),
    ('Isha Sharma', 'isha@example.com', '5432109876'),
    ('Kavya Reddy', 'kavya@example.com', '4321098765'),
    ('Lakshmi Menon', 'lakshmi@example.com', '3210987654'),
    ('Manoj Kumar', 'manoj@example.com', '2109876543'),
    ('Nandini Raj', 'nandini@example.com', '1098765432'),
    ('Omkar Pai', 'omkar@example.com', '0987654321');

INSERT INTO Booking (customer_id, event_id, num_tickets, total_cost, booking_date) VALUES
    (1, 1, 2, 100.00, '2024-03-08'),
    (2, 2, 3, 225.00, '2024-03-09'),
    (3, 3, 5, 500.00, '2024-03-10'),
    (4, 4, 4, 320.00, '2024-03-11'),
    (5, 5, 1, 60.00, '2024-03-12'),
    (6, 6, 2, 60.00, '2024-03-13'),
    (7, 7, 3, 60.00, '2024-03-14'),
    (8, 8, 4, 160.00, '2024-03-15'),
    (9, 9, 2, 140.00, '2024-03-16'),
    (10, 10, 3, 75.00, '2024-03-17');
```

```
-- Select events with available tickets
SELECT * FROM Event WHERE available_seats > 0;

-- Select events name partial match with ‘cup’
SELECT * FROM Event WHERE event_name LIKE '%cup%';

-- Select events with ticket price range between 1000 to 2500
SELECT * FROM Event WHERE ticket_price BETWEEN 1000 AND 2500;

-- Retrieve events with dates falling within a specific range
SELECT * FROM Event WHERE event_date BETWEEN 'start_date' AND 'end_date';
-- Replace 'start_date' and 'end_date' with your desired date range.

-- Retrieve events with available tickets that also have "Concert" in their name
SELECT * FROM Event WHERE available_seats > 0 AND event_name LIKE '%Concert%';

-- Retrieve users in batches of 5, starting from the 6th user
SELECT * FROM Customer LIMIT 5 OFFSET 5;

-- Retrieve bookings details containing booked no of tickets more than 4
SELECT * FROM Booking WHERE num_tickets > 4;

-- Retrieve customer information whose phone number ends with ‘000’
SELECT * FROM Customer WHERE phone_number LIKE '%000';

-- Retrieve events in order whose seat capacity is more than 15000
SELECT * FROM Event WHERE total_seats > 15000 ORDER BY total_seats;

-- 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%';



-- List Events and Their Average Ticket Prices
SELECT event_id, event_name, AVG(ticket_price) AS avg_ticket_price
FROM Event
GROUP BY event_id, event_name;

-- Calculate the Total Revenue Generated by Events
SELECT SUM(total_cost) AS total_revenue
FROM Booking;

-- Find the event with the highest ticket sales
SELECT event_id, event_name, SUM(num_tickets) AS total_tickets_sold
FROM Booking
GROUP BY event_id, event_name
ORDER BY total_tickets_sold DESC
LIMIT 1;

-- Calculate the Total Number of Tickets Sold for Each Event
SELECT event_id, event_name, SUM(num_tickets) AS total_tickets_sold
FROM Booking
GROUP BY event_id, event_name;
```

```
-- Find Events with No Ticket Sales
SELECT *
FROM Event
WHERE event_id NOT IN (SELECT DISTINCT event_id FROM Booking);

-- Find the User Who Has Booked the Most Tickets
SELECT customer_id, customer_name, SUM(num_tickets) AS total_tickets_booked
FROM Booking
GROUP BY customer_id, customer_name
ORDER BY total_tickets_booked DESC
LIMIT 1;

-- List Events and the total number of tickets sold for each month
SELECT MONTH(event_date) AS month, SUM(num_tickets) AS total_tickets_sold
FROM Booking
GROUP BY MONTH(event_date);

-- Calculate the Average Ticket Price for Events in Each Venue
SELECT e.venue_id, v.venue_name, AVG(e.ticket_price) AS avg_ticket_price
FROM Event e
JOIN Venu v ON e.venue_id = v.venue_id
GROUP BY e.venue_id, v.venue_name;

-- Calculate the Total Number of Tickets Sold for Each Event Type
SELECT event_type, SUM(num_tickets) AS total_tickets_sold
FROM Event e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY event_type;

-- Calculate the Total Revenue Generated by Events in Each Year
SELECT YEAR(event_date) AS year, SUM(total_cost) AS total_revenue
FROM Booking
GROUP BY YEAR(event_date);

-- List users who have booked tickets for multiple events
SELECT customer_id, customer_name, COUNT(DISTINCT event_id) AS num_events_booked
FROM Booking
GROUP BY customer_id, customer_name
HAVING num_events_booked > 1;

-- Calculate the Total Revenue Generated by Events for Each User
SELECT customer_id, customer_name, SUM(total_cost) AS total_revenue
FROM Booking
GROUP BY customer_id, customer_name;

-- Calculate the Average Ticket Price for Events in Each Category and Venue
SELECT e.venue_id, v.venue_name, e.event_type, AVG(e.ticket_price) AS avg_ticket_price
FROM Event e
JOIN Venu v ON e.venue_id = v.venue_id
GROUP BY e.venue_id, v.venue_name, e.event_type;

-- List Users and the Total Number of Tickets They've Purchased in the Last 30 Days
SELECT customer_id, customer_name, SUM(num_tickets) AS total_tickets_purchased
```

```
FROM Booking
WHERE booking_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 30 DAY)
GROUP BY customer_id, customer_name;



-- Calculate the Average Ticket Price for Events in Each Venue Using a Subquery
SELECT v.venue_id, v.venue_name,
   (SELECT AVG(ticket_price)
    FROM Event e
    WHERE e.venue_id = v.venue_id) AS avg_ticket_price
FROM Venu v;

-- Find Events with More Than 50% of Tickets Sold using subquery
SELECT *
FROM Event
WHERE (SELECT SUM(num_tickets) FROM Booking WHERE Booking.event_id = Event.event_id) >
(total_seats * 0.5);

-- 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;

-- 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 * FROM Booking WHERE Booking.customer_id = c.customer_id);

-- List Events with No Ticket Sales Using a NOT IN Subquery
SELECT *
FROM Event
WHERE event_id NOT IN (SELECT DISTINCT event_id FROM Booking);

-- Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause
SELECT event_type, SUM(total_tickets_sold) AS total_tickets_sold
FROM (SELECT event_type, event_id,
        (SELECT SUM(num_tickets) FROM Booking WHERE Booking.event_id = Event.event_id) AS
total_tickets_sold
     FROM Event) AS subquery
GROUP BY event_type;

-- Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause
SELECT *
FROM Event
WHERE ticket_price > (SELECT AVG(ticket_price) FROM Event);

-- Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery
SELECT customer_id, customer_name,
   (SELECT SUM(total_cost)
    FROM Booking
    WHERE Booking.customer_id = Customer.customer_id) AS total_revenue
FROM Customer;
```

```
-- 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 EXISTS (SELECT * FROM Booking JOIN Event ON Booking.event_id = Event.event_id WHERE Event.venue_id = 'your_venue_id' AND Booking.customer_id = Customer.customer_id);

-- Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY
SELECT event_type, SUM(total_tickets_sold) AS total_tickets_sold
FROM (SELECT event_type, event_id,
        (SELECT SUM(num_tickets) FROM Booking WHERE Booking.event_id = Event.event_id) AS total_tickets_sold
     FROM Event) AS subquery
GROUP BY event_type;

-- Find Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE_FORMAT
SELECT customer_id, customer_name,
   (SELECT SUM(num_tickets)
    FROM Booking
    WHERE Booking.customer_id = Customer.customer_id
    AND DATE_FORMAT(booking_date, '%Y-%m') = 'desired_year_month') AS total_tickets_booked
FROM Customer;

-- Calculate the Average Ticket Price for Events in Each Venue Using a Subquery
SELECT venue_id, venue_name,
   (SELECT AVG(ticket_price)
    FROM Event
    WHERE Event.venue_id = Venu.venue_id) AS avg_ticket_price
FROM Venu;
```