# TicketBookingSystem

Task-1

**--creating TicketBookingSystem database**

CREATE DATABASE TicketBookingSystem

**--switching to TicketBookingSystem**

USE TicketBookingSystem

**--creating Venue Table**

CREATE TABLE Venu(

venue\_id INT PRIMARY KEY NOT NULL,

venue\_name VARCHAR(50),

address VARCHAR(200)

)

**--creating Event Table**

CREATE TABLE Event(

event\_id INT PRIMARY KEY NOT NULL,

event\_name VARCHAR(50),

event\_date DATE,

event\_time TIME,

venue\_id INT,

total\_seats INT,

available\_seats INT,

ticket\_price DECIMAL,

event\_type VARCHAR(50) CHECK(event\_type IN ('Movie', 'Sports', 'Concert')),

booking\_id INT,

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

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

)

**--creating Customer Table**

CREATE TABLE Customers(

customer\_id INT PRIMARY KEY NOT NULL,

customer\_name VARCHAR(50),

email VARCHAR(50),

phone\_number VARCHAR(20)

booking\_id INT,

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

)

**--creating Booking Table**

CREATE TABLE Booking(

booking\_id INT PRIMARY KEY NOT NULL,

customer\_id INT,

event\_id INT,

num\_tickets INT,

total\_cost DECIMAL,

booking\_date DATETIME,

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

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

)

Task-2

USE TicketBookingSystem

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

**--Inserting values into venu table**

INSERT INTO Venu

VALUES

(1, 'Hotel Gagan Plaza', 'Kanpur city'),

(2, 'Mandakini Heaven Huts', 'South of Kanpur'),

(3, 'Rainforest Retreats Pvt Ltd', 'Saleshpur'),

(4, 'Viveda Wellness Village', 'Nashik'),

(5, 'Ahinsa Residency', 'Gurugram'),

(6, 'Serieti', 'Soul Society'),

(7, 'Leaf village', 'Ohio'),

(8, 'Shiganshina', 'Paradi island'),

(9, 'J-World Tokyo', 'Tokyo'),

(10, 'Chennai Trade Centre', 'Chennai')

**--inserting values into booking table**

INSERT INTO Booking

VALUES

(1, 1, 1, 2, 90.00, '2025-03-25'),

(2, 2, 3, 4, 300.00, '2025-03-26'),

(3, 3, 2, 5, 150.00, '2025-03-27'),

(4, 4, 4, 3, 45.00, '2025-03-28'),

(5, 5, 5, 2, 20.00, '2025-03-29'),

(6, 6, 6, 6, 150.00, '2025-03-30'),

(7, 7, 7, 4, 48.00, '2025-04-01'),

(8, 8, 8, 1, 100.00, '2025-04-02'),

(9, 9, 9, 10, 350.00, '2025-04-03'),

(10, 10, 10, 5, 100.00, '2025-04-04');

**--Inserting values into customer table**

INSERT INTO Customers

VALUES

(1, 'Ajith Kumar', 'thala@example.com', '9865551234', 5),

(2, 'Joesph Vijay', 'tvkvijay@example.com', '7845552345', 4),

(3, 'Mohan Ravi', 'jayamravi@example.com', '9665553456', 3),

(4, 'Dhanush', 'dkraja@example.com', '9995554567', 2),

(5, 'Suriya', 'suriya@example.com', '9475555678', 1),

(6, 'Karthick', 'dili@example.com', '9265556789', 10),

(7, 'Vimal', 'supremestar@example.com', '9365557890', 9),

(8, 'Prithiv Raj', 'goatlife@example.com', '8305558901', 8),

(9, 'Pream', 'pream@example.com', '9875559012', 7),

(10, 'RajiniKanth', 'eagle@example.com', '6875550123',6);

**--Inserting values into event table**

INSERT INTO Event

VALUES

(1, 'Audio Launch', '2025-04-15', '19:00', 1, 500, 500, 45.00, 'Movie', 1),

(2, 'Kabbadi Championship', '2025-04-20', '14:00', 2, 10000, 10000, 30.00, 'Sports', 9),

(3, 'Music Orchestra', '2025-05-01', '20:00', 3, 300, 300, 75.00, 'Concert', 10),

(4, 'Avengers Reloaded', '2025-04-10', '18:00', 4, 200, 200, 15.00, 'Movie', 7),

(5, 'Art Exhibition', '2025-05-05', '10:00', 9, 150, 150, 10.00, 'Concert', 8),

(6, 'Tennis Tournament', '2025-05-10', '13:00', 6, 5000, 5000, 25.00, 'Sports', 6),

(7, 'University Play', '2025-04-25', '19:30', 7, 250, 250, 12.00, 'Movie', 5),

(8, 'Tech Conference', '2025-06-01', '09:00', 8, 1000, 1000, 100.00, 'Concert', 4),

(9, 'Cricket Game', '2025-05-15', '15:00', 6, 12000, 12000, 35.00, 'Sports', 3),

(10, 'Pongal Festival', '2025-06-15', '12:00', 10, 2000, 2000, 20.00, 'Concert', 2);

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

SELECT DISTINCT(event\_name) AS Events FROM Event

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

SELECT event\_name AS available\_tickets FROM Event

WHERE available\_seats >= 1

**--4. Write a SQL query to select events name partial match with 'cup'**

SELECT event\_name AS Events 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 event\_name AS Events 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 event\_name, event\_date FROM Event

WHERE event\_date BETWEEN '2025-04-01' AND '2025-04-30'

**--7. Write a SQL Query to retrieve events with available tickets that also have 'Concert' in their name.**

SELECT event\_name AS Concert\_Events, available\_seats FROM Event

WHERE event\_type = 'Concert' AND available\_seats >= 1

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

SELECT customer\_name AS Batch\_Users FROM Customers

ORDER BY customer\_id ASC

OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY

**--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 Customers

WHERE phone\_number LIKE '%000'

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

SELECT event\_name AS Events FROM Event

WHERE total\_seats > 15000

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

SELECT event\_name AS Events FROM Event

WHERE event\_name NOT LIKE 'x%'

AND event\_name NOT LIKE 'y%'

AND event\_name NOT LIKE 'z%'

Task-3

**--1. Write a SQL query to list events and their Average ticket prices**

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

GROUP BY event\_name

**--2. Write a SQL query to calculate the total revenue generated by events.**

SELECT SUM(total\_cost) AS total\_revenue FROM booking

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

SELECT TOP 1 e.event\_name, SUM(b.num\_tickets) AS tickets\_sold FROM Booking b

INNER JOIN Event e

ON b.booking\_id = e.booking\_id

GROUP BY e.event\_name

ORDER BY tickets\_sold DESC

--4. Write a SQL query to calculate the total number of tickets sold for each event

**SELECT e.event\_name, SUM(b.num\_tickets) AS total\_tickets\_sold FROM Event e**

INNER JOIN Booking b ON e.booking\_id = b.booking\_id

GROUP BY e.event\_name

ORDER BY total\_tickets\_sold DESC

**--5. Write a SQL query to find events with no ticket sales**

SELECT e.event\_name AS no\_sales FROM Event e

LEFT JOIN Booking b

ON e.booking\_id = b.booking\_id

WHERE b.booking\_id is NULL

**--6. Write a SQL query to find the user who has booked the most tickets**

SELECT TOP 1 c.customer\_name, SUM(b.num\_tickets) AS tickets\_sold FROM Customers c

INNER JOIN Booking b ON

c.customer\_id = b.customer\_id

GROUP BY c.customer\_name

ORDER BY tickets\_sold DESC

**--7. Write a SQL query to list events and total number of tickets sold for each month**

SELECT MONTH(booking\_date) AS month, SUM(num\_tickets) AS tickets\_sold FROM Booking

GROUP BY MONTH(booking\_date)

ORDER BY tickets\_sold DESC

**--8. Write a SQL query to calculate the average ticket price for events in each venve**

SELECT v.venue\_name, e.event\_name, AVG(e.ticket\_price) AS average\_ticket\_price FROM Event e

INNER JOIN Venu v

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

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

**--9. Write a SQL query to calculate the total number of tickets sold for each event type**

SELECT e.event\_type, SUM(b.num\_tickets) AS tickets\_sold FROM Event e

INNER JOIN Booking b

ON e.booking\_id = b.booking\_id

GROUP BY e.event\_type

ORDER BY tickets\_sold DESC

**--10. Write a SQL query to calculate the total revenue generated by events in each year**

SELECT YEAR(b.booking\_date) AS year, SUM(b.total\_cost) AS revenue FROM Booking b

GROUP BY YEAR(b.booking\_date)

ORDER BY revenue

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

SELECT c.customer\_name, COUNT(b.customer\_id) AS events\_booked FROM Booking b

INNER JOIN Customers c ON

b.customer\_id = c.customer\_id

GROUP BY c.customer\_name

HAVING COUNT(b.customer\_id) >= 2

**--12. Write a SQL query to calculate the total revenue generated by events for each user.**

SELECT c.customer\_name, e.event\_name, SUM(b.total\_cost) AS revenue FROM Customers c

INNER JOIN Booking b

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

INNER JOIN Event e

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

GROUP BY c.customer\_name, e.event\_name

ORDER BY revenue DESC

**--13. Write a SQL query to calculate the average ticket price for events in each category and venue**

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

INNER JOIN Venu v

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

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

ORDER BY average DESC

**--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\_name, SUM(b.num\_tickets) AS tickets\_booked FROM Booking b

INNER JOIN Customers c

ON b.booking\_id = c.booking\_id

WHERE DATEDIFF(DAY, b.booking\_date, GETDATE()) <= 30

GROUP BY c.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 e WHERE e.venue\_id = v.venue\_id) AS average

FROM Venu v

WHERE EXISTS (SELECT 1 FROM Event e WHERE e.venue\_id = v.venue\_id)

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

SELECT event\_id, SUM(num\_tickets) AS tickets\_sold,

(SELECT e.total\_seats FROM Event e WHERE e.event\_id = b.event\_id)

AS total FROM booking b

GROUP BY event\_id

HAVING SUM(num\_tickets) > (SELECT total\_seats/2 AS num\_tickets FROM Event e

WHERE e.event\_id = b.event\_id)

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

SELECT b.event\_id,

(SELECT event\_name FROM Event e WHERE e.event\_id = b.event\_id) AS event\_name ,

SUM(b.num\_tickets) AS tickets\_sold FROM Booking b

GROUP BY b.event\_id

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

SELECT \* FROM Customers c

WHERE NOT EXISTS (SELECT 1 FROM Booking b

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

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

SELECT event\_name FROM Event e

WHERE e.event\_id NOT IN (SELECT b.event\_id FROM Booking b)

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

SELECT (SELECT e.event\_name FROM Event e WHERE e.event\_id = t.event\_id) AS event\_name,

t.tickets\_sold FROM (SELECT b.event\_id, SUM(b.num\_tickets) AS tickets\_sold FROM Booking b

GROUP BY b.event\_id) AS t

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

SELECT e.event\_name, e.ticket\_price FROM Event e

WHERE e.ticket\_price >= (SELECT AVG(ticket\_price) FROM Event)

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

SELECT (SELECT customer\_name FROM Customers c WHERE b.customer\_id = c.customer\_id) AS customer\_name,

(SELECT e.event\_name FROM Event e WHERE e.event\_id = b.event\_id) AS event\_name ,

SUM(b.total\_cost) AS revenue FROM Booking b

GROUP BY b.customer\_id, b.event\_id

ORDER BY revenue DESC

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

SELECT (SELECT v.venue\_name FROM Venu v WHERE e.venue\_id = v.venue\_id) AS venue, e.event\_name,

(SELECT c.customer\_name FROM Customers c WHERE c.booking\_id = e.booking\_id) AS customer\_name

FROM Event e

WHERE e.venue\_id = 3

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

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

(SELECT b.num\_tickets FROM Booking b WHERE e.booking\_id = b.booking\_id) AS total\_tickets

FROM Event e) temp\_table

GROUP BY event\_type

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

SELECT MONTH(b.booking\_date) AS month,

(SELECT c.customer\_name FROM Customers c WHERE c.customer\_id = b.customer\_id) AS customer\_name

FROM Booking b

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

SELECT venue\_name,

(SELECT AVG(ticket\_price) FROM Event e WHERE e.venue\_id = v.venue\_id) AS average

FROM Venu v

WHERE EXISTS (SELECT 1 FROM Event e WHERE e.venue\_id = v.venue\_id)