Ticket Booking System

**Tasks 1: Database Design:**

1. Create the database named "TicketBookingSystem"
2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

Venue

Event

Customers

Booking

1.

CREATE DATABASE TicketBookingSystem;

USE TicketBookingSystem;

2.

CREATE TABLE Venue(

venue\_id int NOT NULL PRIMARY KEY,

venue\_name VARCHAR(30),

address VARCHAR(50)

);

CREATE TABLE Event(

event\_id int NOT NULL PRIMARY KEY,

event\_name VARCHAR(255),

event\_date DATE,

event\_time TIME,

venue\_id int,

total\_seats int(10000),

available\_seats int(10000),

ticket\_price DECIMAL(10,2),

event\_type VARCHAR(40) CHECK (event\_type IN('Movie','Sports','COncert')),

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 NOT NULL PRIMARY KEY,

customer\_name VARCHAR(20),

email VARCHAR(30),

phone\_number INT(10),

booking\_id int ,

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

);

CREATE TABLE Booking(

booking\_id int NOT NULL PRIMARY KEY,

customer\_id int,

event\_id int,

num\_tickets int,

total\_cost Decimal(10,2),

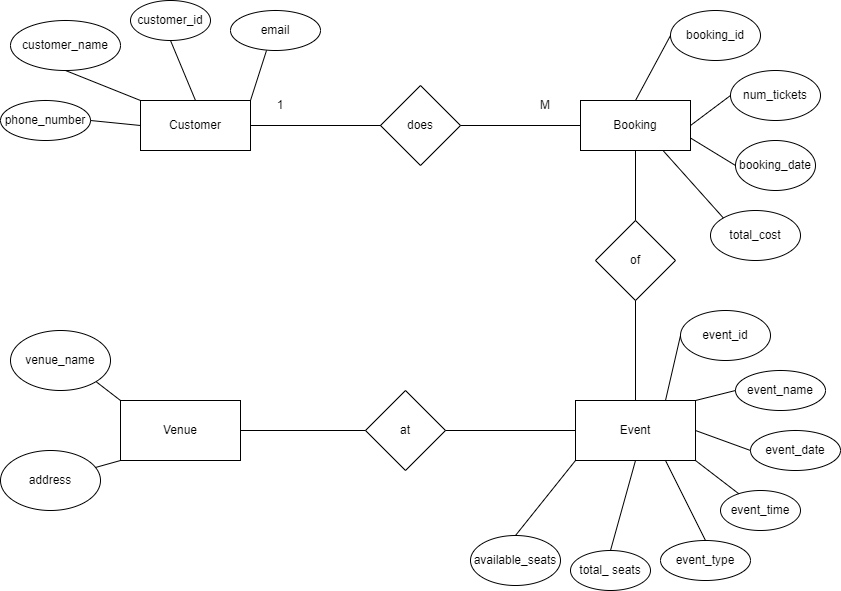
booking\_date DATE,

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

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

);

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



**Tasks 2: Select, Where, Between, AND, LIKE:**

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

2. Write a SQL query to list all Events.

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

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

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

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

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

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

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

10. Write a SQL query to retrieve customer information whose phone number end with ‘000’

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

12. Write a SQL query to select events name not start with ‘x’, ‘y’, ‘z’

**QUERIES**

1.

INSERT INTO Venu(venue\_name,address)

VALUES('PVR Cinemas','The NEXUS Mall, Kormangala Bangalore'),

('Gopalan Cinemas','4th, Gopalan Innovation Mall,Bannerghatta Rd,JP nagar Bangalore'),

('Maheswari Theratre','36th cross road, Near Jal Bhavan, 4th T Block East,Tilak Nagar,

Jayanagar, Bengaluru'),

('M.Chinnaswamy Stadium','Mahatma Gandhi Rd,near cubbon park,Shivaji nagar, Bengaluru'),

('Karnataka State Hockey Stadium','Rhenius St, Langford Town,Langford Gardens,Bengaluru'),

('Shree Kanteerava Indoor Stadium','1st Main Road Sampangi Rama Nagar, Kasturba Rd,

Nunegundlapalli, Raja Ram Mohanroy Extension, Bengaluru'),

('Algorythm','Prestige Shanthinikethan Tower C,Thigalarapalya, Hoodi, Bengaluru'),

('UB City Amphitheatre','24, Vittal Mallaya Rd, KG halli, D'Souza Layout,Sampige Rama Nagar, Bengaluru'),

('Chowdiah Memorial Hall','16th cross Rd, Vyalikaval, Kodandarampura, Malleshwara,

Bengaluru'),

('Palace Ground','JayaMahal, Bengaluru');

INSERT INTO Event(event\_id,event\_name,event\_date,event\_time,venue\_id,total\_seats,available\_seats,

ticket\_price,event\_type,booking\_id)

VALUES(1,’Animal’,’2023-12-13’,’14:00:00’,1,120,60,175.80,’Movie’,1),

(2,’Sapta sagar dache’,’2023-12-14’,’10:00:00’,2,120,60,160.50,’Movie’,2),

(3,’Garadi’,’2023-12-13’,’10:00:00’,3,130,0,155.30,’Movie’,3),

(4,’IND Vs AUS WorldCup’,’2023-12-14’,’14:00:00’,4,9999,250,954.80,’Sports’,4),

(5,’India vs Argentina’,’2023-12-15’,’9:00:00’,5,999,90,1544.20,’Sports’,5),

(6,’Bengaluru Bulls Vs Jaipur Pink Panthers’,’2023-12-14’,’19:00:00’,6,999,150,354.50,’Sports’,6),

(7,’Sonu Nigam Nights’,’2023-12-14’,’18:30:00’,7,1000,400,1754.20,’Concert’,7),

(8,’Armaan Malik Nights’,’2023-12-13’,’18:30:00’,8,1000,0,2354.40,’Concert’,8),

(9,’Arjun Janya Musical Nights’,’2023-12-15’,’18:00:00’,9,1000,300,1650.50,’Concert’,9),

(10,’Indu Nagaraj Concert’,’2023-12-16’,’18:30:00’,10,1000,450,2075.70,’Concert’,10);

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

VALUES(1,Michel,michel@example.com,8453614702,1),

(2,’Zara’,’zara@example.com’,7589641258,2),

(3,’James’,’james@example.com’,7856912432,3),

(4,’Robert’,’robert@example.com’,8475961235,4),

(5,’Suresh’,’suresh@example.com’,7958463000,5),

(6,’Ramya’,’ramya@example.com’,9854672136,6),

(7,’Divya’,’divya@example.com’,9826547135,7),

(8,’Kelin’,’kelin@example.com’,8754692000,8),

(9,’Kavin’,’kavin@example.com’,9784621356,9),

(10,’Purvi’,’purvi@example.com’,9584216782,10);

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

VALUES(1,1,1,4,1234,’2023-12-13’),

(2,2,2,3,658,’2023-12-14’),

(3,3,3,5,3268,’2023-12-15’),

(4,4,4,2,788,’2023-12-16’),

(5,5,5,1,175,’2023-12-13’),

(6,6,6,2,1950,’2023-12-14’),

(7,7,7,4,438,’2023-12-15’),

(8,8,8,6,2318,’2023-12-16’),

(9,9,9,4,764,’2023-12-12’),

(10,10,10,8,2368,’2023-12-15’);

2.

SELECT \* FROM Event;

3.

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

4.

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

5.

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

6.

SELECT \* FROM Event WHERE event\_date BETWEEN 12/12/2023 AND 14/12/2023;

7.

SELECT \* FROM Event WHERE available\_seats > 0 AND event\_type = 'Concert';

8.

SELECT \* FROM Customer ORDER BY customer\_id OFFSET 5 ROWS FETCH next 5 ROWS ONLY;

9.

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

10.

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

11.

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

12.

SELECT \* FROM Event WHERE NOT (event\_name NOT LIKE '%x' OR event\_name LIKE 'y' OR event\_name LIKE 'z');

**Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**QUERIES:**

1.

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

FROM Event

GROUP BY event\_id,event\_name;

2.

SELECT SUM(total\_cost) AS total\_revenue

FROM Booking;

3.

SELECT event\_id, event\_name, SUM(num\_tickets) AS total\_ticket\_sales

FROM Booking

GROUP BY event\_id, event\_name

ORDER BY total\_ticket\_sales DESC

LIMIT 1;

4.

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

FROM Booking

GROUP BY event\_id, event\_name;

5.

SELECT event\_id,event\_name

FROM Event

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

6.

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;

7.

SELECT EXTRACT(MONTH FROM booking\_date) AS month, event\_id, event\_name,SUM(num\_tickets) AS total\_tickets\_sold

FROM Booking

GROUP BY month, event\_id, event\_name;

8.

SELECT venue\_id, AVG(ticket\_price) AS average\_ticket\_price

FROM Event

GROUP BY venue\_id;

9.

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

FROM Event

JOIN Booking ON Event.event\_id = Booking.event\_id

GROUP BY event\_type;

10.

SELECT EXTRACT(YEAR FROM booking\_date) AS year, SUM(total\_cost) AS total\_revenue

FROM Booking

GROUP BY year;

11.

SELECT customer\_id, customer\_name

FROM Customer

WHERE customer\_id IN (SELECT customer\_id FROM Booking GROUP BY customer\_id HAVING COUNT(DISTINCT event\_id) > 1);

12.

SELECT customer\_id, customer\_name, SUM(total\_cost) AS total\_revenue

FROM Booking

JOIN Customer ON Booking.customer\_id = Customer.customer\_id

GROUP BY customer\_id, customer\_name;

13.

SELECT venue\_id, event\_type, AVG(ticket\_price) AS average\_ticket\_price

FROM Event

GROUP BY venue\_id, event\_type;

14.

SELECT customer\_id, customer\_name, SUM(num\_tickets) AS total\_tickets\_purchased

FROM Booking

JOIN Customer ON Booking.customer\_id = Customer.customer\_id

WHERE booking\_date >=CURRENT\_DATE-INTERVAL '30' DAY

GROUP BY customer\_id, customer\_name;

**Tasks 4: Subquery and its types**

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

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

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

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

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

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

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

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

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

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

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

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

**QUERIES**

1.

SELECT venue\_id, venue\_name,

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

FROM Venue;

2.

SELECT event\_id, event\_name

FROM Event

WHERE (SELECT SUM(num\_tickets) FROM Booking WHERE Booking.event\_id = Event.event\_id) > 0.5 \* total\_seats;

3.

SELECT event\_id, event\_name,

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

FROM Event;

4.

SELECT customer\_id, customer\_name

FROM Customer

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

5.

SELECT event\_id, event\_name

FROM Event

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

6.

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

FROM (SELECT Event.event\_type, Booking.num\_tickets

FROM Event

JOIN Booking ON Event.event\_id = Booking.event\_id) AS Subquery

GROUP BY event\_type;

7.

SELECT event\_id, event\_name, ticket\_price

FROM Event

WHERE ticket\_price > (SELECT AVG(ticket\_price) FROM Event);

8.

SELECT customer\_id, customer\_name,

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

FROM Customer;

9.

SELECT customer\_id, customer\_name

FROM Customer

WHERE EXISTS (SELECT 1 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);

10.

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

FROM Event

JOIN Booking ON Event.event\_id = Booking.event\_id

GROUP BY event\_type;

11.

SELECT customer\_id, customer\_name, MONTH(booking\_date) AS booking\_month

FROM Customer

JOIN Booking ON Customer.customer\_id = Booking.customer\_id

GROUP BY customer\_id, customer\_name, booking\_month;

12.

SELECT venue\_id, venue\_name,

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

FROM Venue