NAME : SWETHIKAA

BATCH : C# BATCH -5

ASSIGNMENT : 5

GITHUB LINK : https://github.com/Swethikakannan/HEXAWARE

ASSIGNMENT 5

--TASK 1--

Database Design

1. CREATE DATABASE TicketBookingSystem

2 AND 4

```
CREATE TABLE Venue (
venue_id INT PRIMARY KEY,
venue_name VARCHAR(100),
address VARCHAR(255)
)

CREATE TABLE Event (
event_id INT PRIMARY KEY,
event_name VARCHAR(100),
event_date DATE,
event_time TIME,
venue_id INT,
total_seats INT,
available_seats INT,
ticket_price DECIMAL(10, 2),
event_type VARCHAR(20) CHECK (event_type IN ('Movie', 'Sports', 'Concert')),
FOREIGN KEY (venue_id) REFERENCES Venue(venue_id)
```

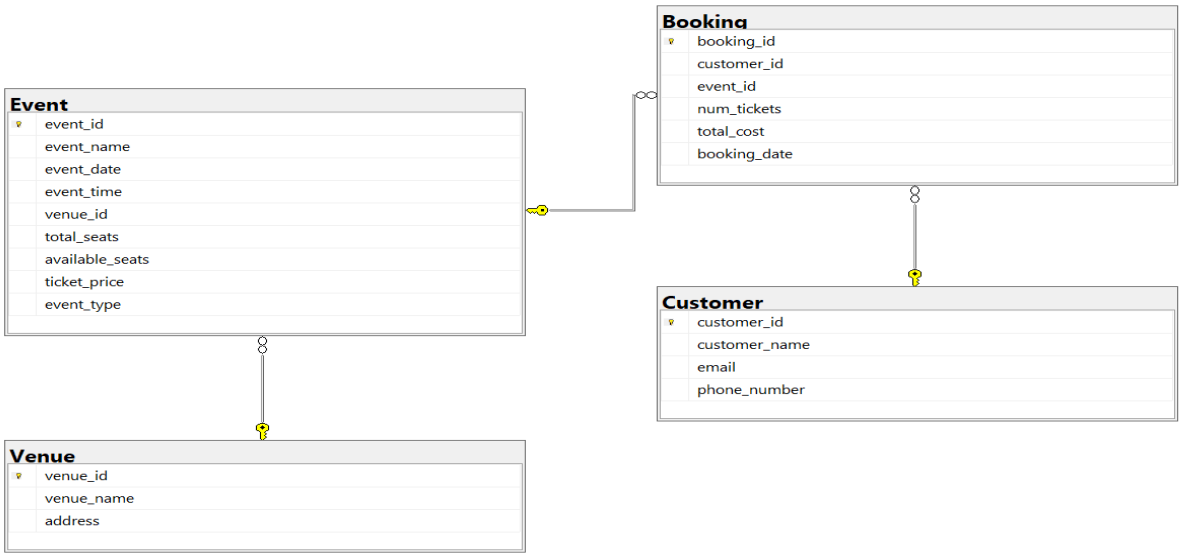
--as mentioning the fk for booking it shows error because it creates a loop,so it will in the booking table respectively

```
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(100),
    email VARCHAR(100),
    phone_number VARCHAR(20)
)
CREATE TABLE Booking (
    booking_id INT 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

ER DIAGRAM:



--TASK 2--

Select, Where, Between, AND, LIKE:

--1

INSERT INTO Venue VALUES
(1, 'Vivanta', 'Coimbatore'),
(2, 'Radisson Blu', 'Chennai'),
(3, 'Taj Skyline', 'Bangalore'),
(4, 'Oberoi Grand', 'Mumbai'),
(5, 'The Leela', 'Delhi'),
(6, 'Trident', 'Hyderabad'),
(7, 'JW Marriott', 'Pune'),
(8, 'Hyatt Regency', 'Jaipur'),
(9, 'The Park', 'Kolkata'),
(10, 'ITC Gardenia', 'Ahmedabad')

INSERT INTO Event VALUES
(101, 'Music Night', '2025-07-10', '18:00:00', 1, 200, 0, 500.00, 'Concert'),
(102, 'Cricket WorldCup Match', '2025-07-15', '19:30:00', 2, 1000, 400, 2000.00, 'Sports'),
(103, 'Movie Show', '2025-07-05', '17:00:00', 3, 300, 100, 250.00, 'Movie'),
(104, 'Live Drama', '2025-08-01', '20:00:00', 4, 350, 150, 350.00, 'Movie'),
(105, 'Football League cup', '2025-08-15', '18:30:00', 5, 800, 300, 900.00, 'Sports'),
(106, 'Comedy Night', '2025-07-20', '16:00:00', 6, 250, 90, 1500.00, 'Movie'),
(107, 'DJ Party', '2025-08-10', '21:00:00', 7, 500, 180, 1000.00, 'Concert'),
(108, 'Kabaddi Match', '2025-07-30', '19:00:00', 8, 600, 200, 700.00, 'Sports'),
(109, 'Short Film Fest', '2025-08-05', '15:30:00', 9, 150, 60, 200.00, 'Movie'),
(110, 'Folk Dance', '2025-08-20', '17:30:00', 10, 400, 250, 550.00, 'Concert')

INSERT INTO Customer VALUES
(1, 'Swetha Raj', 'swetha@gmail.com', '9876543210'),
(2, 'Arjun Menon', 'arjun@gmail.com', '9812345678'),
(3, 'Priya Das', 'priya@gmail.com', '9123456780'),
(4, 'Rohit Sen', 'rohit@gmail.com', '9000012345'),
(5, 'Meera Iyer', 'meera@gmail.com', '9888877777'),

```
(6, 'Karan Patel', 'karan@gmail.com', '9444444444'),
(7, 'Divya Nair', 'divya@gmail.com', '9333333333'),
(8, 'Anil Kumar', 'anil@gmail.com', '9555555555'),
(9, 'Sneha Rao', 'sneha@gmail.com', '9666666666'),
(10, 'Vikram Jha', 'vikram@gmail.com', '9777777777')


INSERT INTO Booking VALUES
(1001, 1, 101, 2, 1598.00, '2025-06-10'),
(1002, 2, 102, 3, 2997.00, '2025-06-11'),
(1003, 3, 103, 1, 299.00, '2025-06-12'),
(1004, 4, 104, 2, 1198.00, '2025-06-13'),
(1005, 5, 105, 4, 4396.00, '2025-06-14'),
(1006, 6, 106, 1, 399.00, '2025-06-15'),
(1007, 7, 107, 2, 1798.00, '2025-06-16'),
(1008, 8, 108, 3, 2397.00, '2025-06-17'),
(1009, 9, 109, 2, 398.00, '2025-06-18'),
(1010, 10, 110, 5, 2495.00, '2025-06-19')

--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 '2025-07-01' AND '2025-08-31'
```

--7

```
SELECT * FROM Event WHERE available_seats > 0 AND event_name LIKE '%Concert%'
```

--8

- SELECT * FROM Customer WHERE customer_id BETWEEN 6 AND 10
- 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
```

--12

```
SELECT * FROM Event WHERE event_name NOT LIKE 'x%'
  AND event_name NOT LIKE 'y%'
  AND event_name NOT LIKE 'z%'
```

--TASK 3--

Aggregate functions, Having, Order By, GroupBy and Joins:

--1

```
SELECT event_name, AVG(ticket_price) AS avg_price FROM Event
GROUP BY event_name
```

--2

```
SELECT e.event_name, (e.total_seats - e.available_seats) * e.ticket_price AS total_revenue
FROM Event e
```

--3

```
SELECT TOP 1 event_name, (total_seats - available_seats) AS tickets_sold
FROM Event
ORDER BY tickets_sold DESC
```

--4

```
SELECT  * FROM Event

SELECT event_name,  SUM(total_seats - available_seats) AS total_sold_seats
FROM Event
GROUP BY event_name
```

--5

```
SELECT e.event_name
FROM Event e
JOIN Booking b ON e.event_id = b.event_id
WHERE b.booking_id IS NULL
```

--6

```
SELECT TOP 1 c.customer_name, SUM(b.num_tickets) AS total_tickets
FROM Booking b
JOIN Customer c ON b.customer_id = c.customer_id
GROUP BY c.customer_name
ORDER BY total_tickets DESC

--7
SELECT e.event_name, MONTH(b.booking_date) AS booking_month, SUM(b.num_tickets) AS tickets_sold
FROM Booking b
JOIN Event e ON b.event_id = e.event_id
GROUP BY MONTH(b.booking_date), e.event_name
ORDER BY booking_month

--8
SELECT v.venue_name, AVG(e.ticket_price) AS avg_ticket_price
FROM Event e
JOIN Venue v ON e.venue_id = v.venue_id
GROUP BY v.venue_name

--9
SELECT e.event_type, SUM(b.num_tickets) AS total_tickets
FROM Event e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY e.event_type

--10
SELECT YEAR(e.event_date) AS event_year,
    SUM(b.total_cost) AS total_revenue
FROM Event e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY YEAR(e.event_date)

--11
```

```
SELECT c.customer_name, COUNT(DISTINCT b.event_id) AS events_booked
FROM Booking b
JOIN Customer c ON b.customer_id = c.customer_id
GROUP BY c.customer_name
HAVING COUNT(DISTINCT b.event_id) > 1

--12
SELECT c.customer_name, SUM(b.total_cost) AS total_revenue
FROM Booking b
JOIN Customer c ON b.customer_id = c.customer_id
GROUP BY c.customer_name

--13
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

--14
SELECT c.customer_name, SUM(b.num_tickets) AS tickets_last_30_days
FROM Booking b
JOIN Customer c ON b.customer_id = c.customer_id
WHERE b.booking_date >= DATEADD(DAY, -30, GETDATE())
GROUP BY c.customer_name
```

--TASK 4--

Subquery and its types

```
--1
SELECT venue_name,
(SELECT AVG(ticket_price) FROM Event e  WHERE e.venue_id = v.venue_id) AS avg_price
FROM Venue v

--2
SELECT event_name
FROM Event
WHERE (total_seats - available_seats) > (total_seats /2 )

--3
SELECT event_name,
      (SELECT SUM(b.num_tickets)
       FROM Booking b
       WHERE b.event_id = e.event_id) AS total_tickets
FROM Event e;

--4
SELECT customer_name
FROM Customer c
WHERE NOT EXISTS (
   SELECT 1
   FROM Booking b
   WHERE b.customer_id = c.customer_id )

--5
SELECT event_name
FROM Event
WHERE event_id NOT IN (
   SELECT event_id
   FROM Booking )
```

```
--6
 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

--7
SELECT event_name, ticket_price
FROM Event
WHERE ticket_price > (
   SELECT AVG(ticket_price) FROM Event)

--8
SELECT customer_name, (SELECT sum(b.total_cost)
       FROM Booking b
       WHERE b.customer_id = c.customer_id) AS total_revenue
FROM Customer c

--9
SELECT customer_name
FROM Customer
WHERE customer_id IN (
   SELECT DISTINCT b.customer_id
   FROM Booking b
   JOIN Event e ON b.event_id = e.event_id
   WHERE e.venue_id =4)

--10
SELECT event_type, SUM(tickets_sold) AS total_tickets
FROM ( SELECT e.event_type, b.num_tickets AS tickets_sold
   FROM Booking b
   JOIN Event e ON b.event_id = e.event_id
) AS ticket_summary GROUP BY event_type
```

```
--11
SELECT DISTINCT c.customer_name
FROM Customer c
WHERE c.customer_id IN (
   SELECT b.customer_id
   FROM Booking b
   WHERE FORMAT(b.booking_date, 'yyyy-MM') IS NOT NULL )


--12
SELECT venue_name,
      (SELECT AVG(ticket_price)
       FROM Event e
       WHERE e.venue_id = v.venue_id) AS avg_price
FROM Venue v
```