Tasks 1: Database Design:

1.Create the database named "TicketBookingSystem"

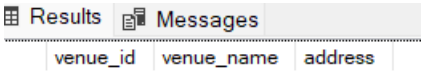
```
CREATE DATABASE TicketBookingSystem;

USE TicketBookingSystem;
```

2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

- **Venue**
- **Event**
- **Customers**
- **Booking**

Venue Table:

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

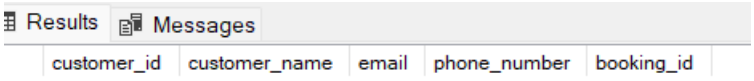


Event Table:

```
CREATE TABLE Event (
    event_id INT 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 VARCHAR(50),
    booking_id INT
);
```

Customer Table:

```
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    email VARCHAR(255),
    phone_number VARCHAR(20),
    booking_id INT
);
```



Booking Table:

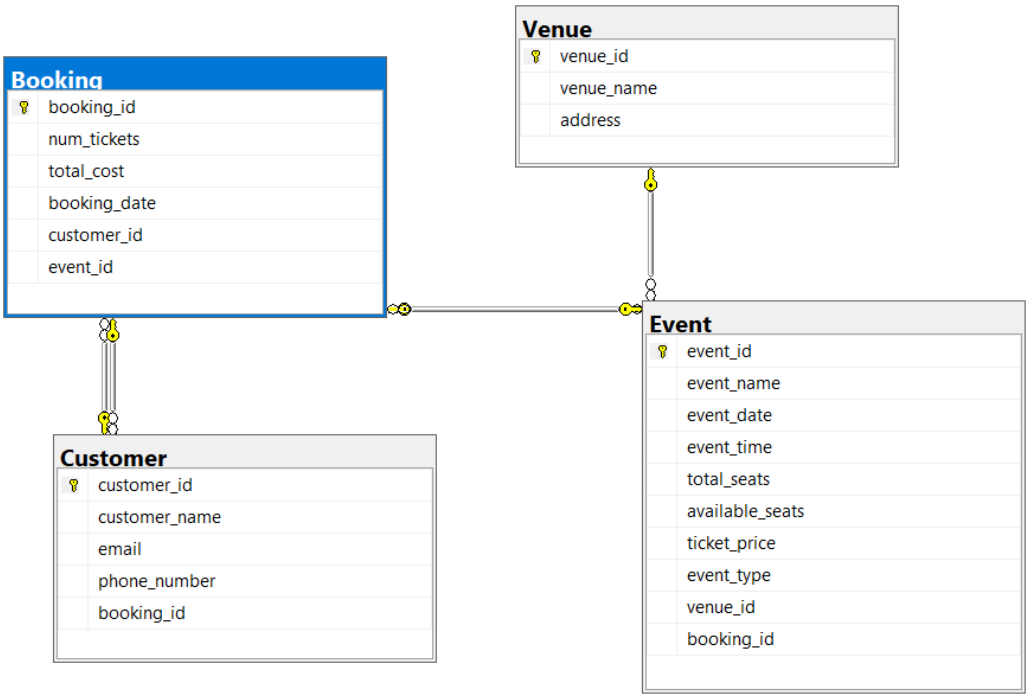
```
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY,
    customer_id INT,
    event_id INT,
```

```
    num_tickets INT,
    total_cost DECIMAL(10, 2),
    booking_date DATE
);
```

Results | Messages

booking_id	customer_id	event_id	num_tickets	total_cost	booking_date

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



4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

```
Alter table Event
add venue_id INT

Alter table Event
add booking_id INT

Alter table Customer
add booking_id INT
```

```
Alter table booking
add customer_id INT

Alter table booking
add event_id INT
ALTER TABLE Event
add constraint FK_venue_id FOREIGN KEY (venue_id) REFERENCES Venue(venue_id);

ALTER TABLE Event
add constraint FK_booking_id FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);

ALTER TABLE Customer
add constraint FKey_booking_id FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);

ALTER TABLE booking
add constraint FK_customer_id  FOREIGN KEY(customer_id) references Customer(customer_id);


ALTER TABLE booking
add constraint FK_event_id FOREIGN KEY(event_id) references Event(event_id);
```

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

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

```
INSERT INTO Venue (venue_id, venue_name, address) VALUES
(1, 'Delhi', 'Suite 640 1856 Turner Camp, East Harriett, NJ 34953'),
(2, 'Mumbai', 'F-1/10, Sector 10, Vashi'),
(3, 'Hyderabad', '5-1-459/7, Jam Bagh'),
(4, 'Bangalore', '92, 2nd Flr Mosque Road, Frazer Town'),
(5, 'Ahmedabad', 'Opp Spss Hall Nr Aditya Complex, Navrangpura'),
(6, 'Vadodara', '30, Narendra Park Society, Near'),
(7, 'Pune', 'Opp Green Roadways,near Hotel, Orient, Kasba Peth'),
(8, 'Madurai', 'D/g 29-30-31, Sardar Patel Cp, Station Road, Gidc, Ankleshwar'),
(9, 'Trichy', 'ABC Street,Navinagar'),
(10, 'Chennai', '12/13 agenda complex,AnnaNagar');
```

Results | Messages

	venue_id	venue_name	address
1	1	Delhi	Suite 640 1856 Turner Camp, East Harriett, NJ 349...
2	2	Mumbai	F-1/10, Sector 10, Vashi
3	3	Hyderabad	5-1-459/7, Jam Bagh
4	4	Bangalore	92, 2nd Flr Mosque Road, Frazer Town
5	5	Ahmedabad	Opp Spss Hall Nr Aditya Complex, Navrangpura
6	6	Vadodara	30, Narendra Park Society, Near
7	7	Pune	Opp Green Roadways,near Hotel, Orient, Kasba Peth
8	8	Madurai	D/g 29-30-31, Sardar Patel Cp, Station Road, Gidc, ...
9	9	Trichy	ABC Street,Navinagar
10	10	Chennai	12/13 agenda complex,AnnaNagar

INSERT INTO Event (event_id, event_name, event_date, event_time, total_seats, available_seats, ticket_price, event_type, venue_id,booking_id)
VALUES
(1, 'Fictional', '2023-01-01', '12:00:00', 100, 50, 250.00, 'Movie',1,1),
(2, 'Kabbadi Cup', '2023-02-02', '15:30:00', 1500, 500, 380.00, 'Sports',2,2),
(3, 'Dance Concert', '2023-02-03', '15:00:00', 155, 100, 2000.00, 'Concert',3,3),
(4, 'Horror', '2023-02-04', '11:30:00', 150, 100, 300.00, 'Movie',4,4),
(5, 'singing', '2023-02-05', '12:30:00', 150, 100, 2000.00, 'Concert',5,5),
(6, 'Volleyball', '2023-02-06', '13:30:00', 150, 100, 300.00, 'Sports',6,6),
(7, 'Comics', '2023-02-07', '14:30:00', 150, 100, 200.00, 'Movie',7,7),
(8, 'Music', '2023-02-08', '15:30:00', 1000, 600, 3000.00, 'Concert',8,8),
(9, 'Thriller', '2023-02-09', '16:30:00', 350, 250, 200.00, 'Movie',9,9),
(10, 'Football', '2023-02-10', '17:30:00', 150, 100, 350.00, 'Sports',10,10);

Results | Messages

	event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	venue_id	booking_id
1	1	Fictional	2023-01-01	12:00:00.0000000	100	50	250.00	Movie	1	1
2	2	Kabbadi Cup	2023-02-02	15:30:00.0000000	1500	500	380.00	Sports	2	2
3	3	Dance Concert	2023-02-03	15:00:00.0000000	155	100	2000.00	Concert	3	3
4	4	Horror	2023-02-04	11:30:00.0000000	150	100	300.00	Movie	4	4
5	5	singing	2023-02-05	12:30:00.0000000	150	100	2000.00	Concert	5	5
6	6	Volleyball	2023-02-06	13:30:00.0000000	150	100	300.00	Sports	6	6
7	7	Comics	2023-02-07	14:30:00.0000000	150	100	200.00	Movie	7	7
8	8	Music	2023-02-08	15:30:00.0000000	1000	600	3000.00	Concert	8	8
9	9	Thriller	2023-02-09	16:30:00.0000000	350	250	200.00	Movie	9	9
10	10	Football	2023-02-10	17:30:00.0000000	150	100	350.00	Sports	10	10

INSERT INTO Customer (customer_id, customer_name, email, phone_number, booking_id) VALUES
(1, 'John', 'john@example.com', '123-456-7890',1),
(2, 'Jane', 'jane@example.com', '987-654-3210',2),
(3, 'Bob', 'bob@example.com', '423-456-7890',3),
(4, 'Alice', 'alice@example.com', '523-456-7890',4),
(5, 'Charlie', 'charlie@example.com', '623-456-7890',5),
(6, 'Eva', 'eva@example.com', '723-456-7890',6),
(7, 'Frank', 'frank@example.com', '823-456-7000',7),
(8, 'Grace', 'grace@example.com', '923-456-7000',8),
(9, 'Henry', 'henry@example.com', '223-456-7000',9),
(10, 'Ivy', 'ivy.@example.com', '323-456-7890',10);

Results Messages

	customer_id	customer_name	email	phone_number	booking_id
1	1	John	john@example.com	123-456-7890	1
2	2	Jane	jane@example.com	987-654-3210	2
3	3	Bob	bob@example.com	423-456-7890	3
4	4	Alice	alice@example.com	523-456-7890	4
5	5	Charlie	charlie@example.com	623-456-7890	5
6	6	Eva	eva@example.com	723-456-7890	6
7	7	Frank	frank@example.com	823-456-7000	7
8	8	Grace	grace@example.com	923-456-7000	8
9	9	Henry	henry@example.com	223-456-7000	9
10	10	Ivy	ivy.@example.com	323-456-7890	10

INSERT INTO Booking (booking_id, num_tickets, total_cost, booking_date, customer_id, event_id) VALUES
(1, 2, 4000.00, '2023-01-01 10:00:00',1,1),
(2, 3, 900.00, '2023-02-02 12:30:00',2, 2),
(3, 2, 5550.00, '2023-01-03 09:00:00',3, 3),
(4, 2, 4444.00, '2023-01-04 09:00:00', 4, 4),
(5, 1, 4300.00, '2023-01-05 10:00:00', 5, 5),
(6, 2, 3540.00, '2023-01-06 11:00:00', 6, 6),
(7, 4, 3670.00, '2023-01-07 11:00:00',7, 7),
(8, 2, 2240.00, '2023-01-08 10:00:00', 8, 8),
(9, 3, 4540.00, '2023-01-09 09:00:00',9,9),
(10, 2, 5540.00, '2023-01-10 10:00:00',10, 10);

Results Messages

	booking_id	num_tickets	total_cost	booking_date	customer_id	event_id
1	1	2	4000.00	2023-01-01	1	1
2	2	3	900.00	2023-02-02	2	2
3	3	2	5550.00	2023-01-03	3	3
4	4	2	4444.00	2023-01-04	4	4
5	5	1	4300.00	2023-01-05	5	5
6	6	2	3540.00	2023-01-06	6	6
7	7	4	3670.00	2023-01-07	7	7
8	8	2	2240.00	2023-01-08	8	8
9	9	3	4540.00	2023-01-09	9	9
10	10	2	5540.00	2023-01-10	10	10

2. Write a SQL query to list all Events.

SELECT * FROM Event;

Results Messages

	event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	venue_id	booking_id
1	1	Fictional	2023-01-01	12:00:00.0000000	100	50	250.00	Movie	1	1
2	2	Kabbadi Cup	2023-02-02	15:30:00.0000000	1500	500	380.00	Sports	2	2
3	3	Dance Concert	2023-02-03	15:00:00.0000000	155	100	2000.00	Concert	3	3
4	4	Horror	2023-02-04	11:30:00.0000000	150	100	300.00	Movie	4	4
5	5	singing	2023-02-05	12:30:00.0000000	150	100	2000.00	Concert	5	5
6	6	Volleyball	2023-02-06	13:30:00.0000000	150	100	300.00	Sports	6	6
7	7	Comics	2023-02-07	14:30:00.0000000	150	100	200.00	Movie	7	7
8	8	Music	2023-02-08	15:30:00.0000000	1000	600	3000.00	Concert	8	8
9	9	Thriller	2023-02-09	16:30:00.0000000	350	250	200.00	Movie	9	9
10	10	Football	2023-02-10	17:30:00.0000000	150	100	350.00	Sports	10	10

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

SELECT event_id,event_name,available_seats FROM Event ;

Results Messages

	event_id	event_name	available_seats
1	1	Fictional	50
2	2	Kabbadi Cup	500
3	3	Dance Concert	100
4	4	Horror	100
5	5	singing	100
6	6	Volleyball	100
7	7	Comics	100
8	8	Music	600
9	9	Thriller	250
10	10	Football	100

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

```
SELECT * FROM Event
WHERE event_name LIKE '%cup%';
```

Results Messages

	event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	venue_id	booking_id
1	2	Kabbadi Cup	2023-02-02	15:30:00.0000000	1500	500	380.00	Sports	2	2

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

```
SELECT   *  FROM Event
WHERE ticket_price BETWEEN 1000 AND 2500;
```

Results Messages

	event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	venue_id	booking_id
1	3	Dance Concert	2023-02-03	15:00:00.0000000	155	100	2000.00	Concert	3	3
2	5	singing	2023-02-05	12:30:00.0000000	150	100	2000.00	Concert	5	5

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

```
SELECT * FROM Event
WHERE event_date BETWEEN '2023-01-01' AND '2023-12-31';
```

Results Messages

	event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	venue_id	booking_id
1	1	Fictional	2023-01-01	12:00:00.0000000	100	50	250.00	Movie	1	1
2	2	Kabbadi Cup	2023-02-02	15:30:00.0000000	1500	500	380.00	Sports	2	2
3	3	Dance Concert	2023-02-03	15:00:00.0000000	155	100	2000.00	Concert	3	3
4	4	Horror	2023-02-04	11:30:00.0000000	150	100	300.00	Movie	4	4
5	5	singing	2023-02-05	12:30:00.0000000	150	100	2000.00	Concert	5	5
6	6	Volleyball	2023-02-06	13:30:00.0000000	150	100	300.00	Sports	6	6
7	7	Comics	2023-02-07	14:30:00.0000000	150	100	200.00	Movie	7	7
8	8	Music	2023-02-08	15:30:00.0000000	1000	600	3000.00	Concert	8	8
9	9	Thriller	2023-02-09	16:30:00.0000000	350	250	200.00	Movie	9	9
10	10	Football	2023-02-10	17:30:00.0000000	150	100	350.00	Sports	10	10

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

```
SELECT * FROM Event

WHERE available_seats > 0 AND event_name LIKE '%Concert%';
```

Results Messages

	event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	venue_id	booking_id
1	3	Dance Concert	2023-02-03	15:00:00.0000000	155	100	2000.00	Concert	3	3

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

```
SELECT customer_id, customer_name, email, phone_number
FROM Customer
ORDER BY customer_id
OFFSET 5 ROWS
FETCH NEXT 5 ROWS ONLY;
```

Results Messages

	customer_id	customer_name	email	phone_number
1	6	Eva	eva@example.com	723-456-7890
2	7	Frank	frank@example.com	823-456-7000
3	8	Grace	grace@example.com	923-456-7000
4	9	Henry	henry@example.com	223-456-7000
5	10	Ivy	ivy.@example.com	323-456-7890

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

```
SELECT * FROM Booking

WHERE num_tickets > =4;
```

Results Messages

	booking_id	num_tickets	total_cost	booking_date	customer_id	event_id
1	7	4	3670.00	2023-01-07	7	7

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

```
SELECT * FROM Customer

WHERE phone_number LIKE '%000';
```

Results Messages

	customer_id	customer_name	email	phone_number	booking_id
1	7	Frank	frank@example.com	823-456-7000	7
2	8	Grace	grace@example.com	923-456-7000	8
3	9	Henry	henry@example.com	223-456-7000	9

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

```
SELECT * FROM Event

WHERE total_seats > 150

ORDER BY total_seats;
```

Results Messages

	event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	venue_id	booking_id
1	3	Dance Concert	2023-02-03	15:00:00.0000000	155	100	2000.00	Concert	3	3
2	9	Thriller	2023-02-09	16:30:00.0000000	350	250	200.00	Movie	9	9
3	8	Music	2023-02-08	15:30:00.0000000	1000	600	3000.00	Concert	8	8
4	2	Kabbadi Cup	2023-02-02	15:30:00.0000000	1500	500	380.00	Sports	2	2

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

```
SELECT * FROM Event

WHERE event_name NOT LIKE 'x%' AND event_name NOT LIKE 'y%' AND
event_name NOT LIKE 'v%';
```

Results Messages

	event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	venue_id	booking_id
1	1	Fictional	2023-01-01	12:00:00.0000000	100	50	250.00	Movie	1	1
2	2	Kabbadi Cup	2023-02-02	15:30:00.0000000	1500	500	380.00	Sports	2	2
3	3	Dance Concert	2023-02-03	15:00:00.0000000	155	100	2000.00	Concert	3	3
4	4	Horror	2023-02-04	11:30:00.0000000	150	100	300.00	Movie	4	4
5	5	singing	2023-02-05	12:30:00.0000000	150	100	2000.00	Concert	5	5
6	7	Comics	2023-02-07	14:30:00.0000000	150	100	200.00	Movie	7	7
7	8	Music	2023-02-08	15:30:00.0000000	1000	600	3000.00	Concert	8	8
8	9	Thriller	2023-02-09	16:30:00.0000000	350	250	200.00	Movie	9	9
9	10	Football	2023-02-10	17:30:00.0000000	150	100	350.00	Sports	10	10

TASK 3: Aggregate functions, Group By and Joins:

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

```
SELECT e.event_id, e.event_name, AVG(e.ticket_price) AS average_ticket_price
FROM Event e
GROUP BY e.event_id, e.event_name;
```

Results Messages

	event_id	event_name	average_ticket_price
1	1	Fictional	250.000000
2	2	Kabbadi Cup	380.000000
3	3	Dance Concert	2000.000000
4	4	Horror	300.000000
5	5	singing	2000.000000
6	6	Volleyball	300.000000
7	7	Comics	200.000000
8	8	Music	3000.000000
9	9	Thriller	200.000000
10	10	Football	350.000000

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

```
SELECT SUM(B.total_cost) AS total_revenue FROM Booking B;
```

Results Messages

	total_revenue
1	38724.00

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

```
SELECT TOP 1 E.event_id,E.event_name,SUM(B.num_tickets) AS total_tickets_sold
FROM Event E
JOIN Booking B ON E.event_id = B.event_id
GROUP BY E.event_id, E.event_name
ORDER BY total_tickets_sold DESC;
```

Results Messages

	event_id	event_name	total_tickets_sold
1	7	Comics	4

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

```
SELECT E.event_id, E.event_name, SUM(B.num_tickets) AS total_tickets_sold
FROM Event E
JOIN Booking B ON E.event_id = B.event_id
GROUP BY E.event_id, E.event_name;
```

Results Messages

	event_id	event_name	total_tickets_sold
1	1	Fictional	2
2	2	Kabbadi Cup	3
3	3	Dance Concert	2
4	4	Horror	2
5	5	singing	1
6	6	Volleyball	2
7	7	Comics	4
8	8	Music	2
9	9	Thriller	3
10	10	Football	2

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

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

Results Messages

event_id	event_name

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

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

Results Messages

	customer_id	customer_name	total_tickets_booked
1	7	Frank	4

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

```
SELECT E.event_id,E.event_name,MONTH(B.booking_date) AS booking_month,SUM(B.num_tickets) AS total_tickets_sold
FROM Event E JOIN Booking B ON E.event_id = B.event_id
GROUP BY E.event_id, E.event_name, MONTH(booking_date)
ORDER BY booking_month, E.event_id;
```

Results Messages

	event_id	event_name	booking_month	total_tickets_sold
1	1	Fictional	1	2
2	3	Dance Concert	1	2
3	4	Horror	1	2
4	5	singing	1	1
5	6	Volleyball	1	2
6	7	Comics	1	4
7	8	Music	1	2
8	9	Thriller	1	3
9	10	Football	1	2
10	2	Kabbadi Cup	2	3

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

```
SELECT V.venue_id, V.venue_name, AVG(E.ticket_price) AS average_ticket_price FROM Venue V
JOIN Event E ON V.venue_id = E.venue_id
GROUP BY V.venue_id, V.venue_name;
```

Results Messages

	venue_id	venue_name	average_ticket_price
1	1	Delhi	250.000000
2	2	Mumbai	380.000000
3	3	Hyderabad	2000.000000
4	4	Bangalore	300.000000
5	5	Ahmedabad	2000.000000
6	6	Vadodara	300.000000
7	7	Pune	200.000000
8	8	Madurai	3000.000000
9	9	Trichy	200.000000
10	10	Chennai	350.000000

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 total_tickets_sold FROM Event e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY e.event_type;
```

Results Messages

	event_type	total_tickets_sold
1	Concert	5
2	Movie	11
3	Sports	7

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

```
SELECT YEAR(B.booking_date) AS booking_year, SUM(B.total_cost) AS total_revenue
FROM Booking B
GROUP BY YEAR(B.booking_date)
ORDER BY booking_year;
```

Results Messages

	booking_year	total_revenue
1	2023	38724.00

11. List users who have booked tickets for multiple events:

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

Results Messages

customer_id	customer_name	num_events_booked

12. Calculate the Total Revenue Generated by Events for Each User:

```
select c.customer_id,c.customer_name, SUM(b.total_cost) as total_revenue from Customer c join Booking b on c.customer_id = b.customer_id group by c.customer_id, c.customer_name;
```

Results Messages

	customer_id	customer_name	total_revenue
1	1	John	4000.00
2	2	Jane	900.00
3	3	Bob	5550.00
4	4	Alice	4444.00
5	5	Charlie	4300.00
6	6	Eva	3540.00
7	7	Frank	3670.00
8	8	Grace	2240.00
9	9	Henry	4540.00
10	10	Ivy	5540.00

13. Calculate the Average Ticket Price for Events in Each Category and Venue:

```
SELECT e.event_type, v.venue_name, AVG(e.ticket_price) AS average_ticket_price
FROM Event e
JOIN Venue v ON e.venue_id = v.venue_id
GROUP BY e.event_type, v.venue_name;
```

Results Messages

	event_type	venue_name	average_ticket_price
1	Concert	Ahmedabad	2000.000000
2	Movie	Bangalore	300.000000
3	Sports	Chennai	350.000000
4	Movie	Delhi	250.000000
5	Concert	Hyderabad	2000.000000
6	Concert	Madurai	3000.000000
7	Sports	Mumbai	380.000000
8	Movie	Pune	200.000000
9	Movie	Trichy	200.000000
10	Sports	Vadodara	300.000000

14. List Users and the Total Number of Tickets They've Purchased in the Last 30 Days:

```
SELECT c.customer_id, c.customer_name, COUNT(b.booking_id) AS total_tickets_purchased
FROM Customer c
JOIN Booking b ON c.booking_id = b.booking_id
WHERE b.booking_date >= DATEADD(DAY,-30,GETDATE())
GROUP BY c.customer_id, c.customer_name;
```

Results Messages

	customer_id	customer_name	total_tickets_purchased

TASK 4 – Subquery and its Types

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

```
SELECT v.venue_id, v.venue_name,
(SELECT AVG(e.ticket_price) FROM Event e WHERE e.venue_id = v.venue_id) AS average_ticket_price
FROM Venue v;
```

Results Messages

	venue_id	venue_name	average_ticket_price
1	1	Delhi	250.000000
2	2	Mumbai	380.000000
3	3	Hyderabad	2000.000000
4	4	Bangalore	300.000000
5	5	Ahmedabad	2000.000000
6	6	Vadodara	300.000000
7	7	Pune	200.000000
8	8	Madurai	3000.000000
9	9	Trichy	200.000000
10	10	Chennai	350.000000

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

```
SELECT
e.event_id,e.event_name,e.total_seats,available_seats,ticket_price,event_type
FROM Event e
WHERE (SELECT SUM(num_tickets) FROM Booking b WHERE b.event_id = e.event_id) > 0.5 * e.total_seats;
```

Results Messages

	event_id	event_name

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

```
SELECT e.event_id, e.event_name,
```

(SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id = e.event_id) AS total_tickets_sold
FROM Event e;

Results Messages

	event_id	event_name	total_tickets_sold
1	1	Fictional	2
2	2	Kabbadi Cup	3
3	3	Dance Concert	2
4	4	Horror	2
5	5	singing	1
6	6	Volleyball	2
7	7	Comics	4
8	8	Music	2
9	9	Thriller	3
10	10	Football	2

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

SELECT c.customer_id, c.customer_name FROM Customer c

WHERE NOT EXISTS (SELECT 1 FROM Booking b WHERE c.booking_id = b.booking_id);

Results Messages

customer_id	customer_name

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

SELECT e.event_id, e.event_name
FROM Event e
WHERE e.event_id NOT IN (SELECT DISTINCT event_id FROM Booking b);

Results Messages

event_id	event_name

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

SELECT

```
    e.event_type,
    SUM(b.num_tickets) AS total_tickets_sold
FROM
    (SELECT
        event_id,
        event_type
    FROM
        Event) e
JOIN
    Booking b ON e.event_id = b.event_id
GROUP BY
    e.event_type;
```

Results Messages

	event_type	total_tickets_sold
1	Concert	5
2	Movie	11
3	Sports	7

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

```
SELECT event_id, event_name, ticket_price FROM Event
WHERE ticket_price > ( SELECT AVG(ticket_price) FROM Event );
```

Results Messages

	event_id	event_name	ticket_price
1	3	Dance	2000.00
2	5	singing	2000.00
3	8	Music	3000.00

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

```
SELECT
    c.customer_id,
    c.customer_name,
    (
        SELECT
            SUM(b.total_cost)
        FROM
            Booking b
        WHERE
            b.customer_id = c.customer_id
    ) AS total_revenue_generated
    FROM
    Customer c;
```

Results Messages

	customer_id	customer_name	total_revenue
1	1	John	4000.00
2	2	Jane	900.00
3	3	Bob	5550.00
4	4	Alice	4444.00
5	5	Charlie	4300.00
6	6	Eva	3540.00
7	7	Frank	3670.00
8	8	Grace	2240.00
9	9	Henry	4540.00
10	10	Ivy	5540.00

9. 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 customer_id IN (SELECT DISTINCT customer_id FROM Booking
WHERE event_id IN (SELECT event_id FROM Event WHERE venue_id = 1));
```

Results Messages

	customer_id	customer_name
1	1	John

10. 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_id, event_type,
      (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;
```

Results Messages

	event_type	total_tickets_sold
1	Concert	5
2	Movie	11
3	Sports	7

11. Find Users Who Have Booked Tickets for Events in a Given Month Using a Subquery with DATE_FORMAT.

```
SELECT
    c.customer_id,
    c.customer_name,
    FORMAT(booking_date, 'MM-yyyy') AS booking_month
FROM
    Customer c
JOIN
    Booking b ON c.customer_id = b.customer_id
GROUP BY
    c.customer_id, c.customer_name, FORMAT(booking_date, 'MM-yyyy');
```

Results | Messages

	customer_id	customer_name	booking_month
1	1	John	01-2023
2	2	Jane	02-2023
3	3	Bob	01-2023
4	4	Alice	01-2023
5	5	Charlie	01-2023
6	6	Eva	01-2023
7	7	Frank	01-2023
8	8	Grace	01-2023
9	9	Henry	01-2023
10	10	Ivy	01-2023

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

```
SELECT v.venue_id, v.venue_name,
(SELECT AVG(e.ticket_price) FROM Event e WHERE e.venue_id = v.venue_id) AS average_ticket_price
FROM Venue v;
```

Results | Messages

	venue_id	venue_name	average_ticket_price
1	1	Delhi	250.000000
2	2	Mumbai	380.000000
3	3	Hyderabad	2000.000000
4	4	Bangalore	300.000000
5	5	Ahmedabad	2000.000000
6	6	Vadodara	300.000000
7	7	Pune	200.000000
8	8	Madurai	3000.000000
9	9	Trichy	200.000000
10	10	Chennai	350.000000