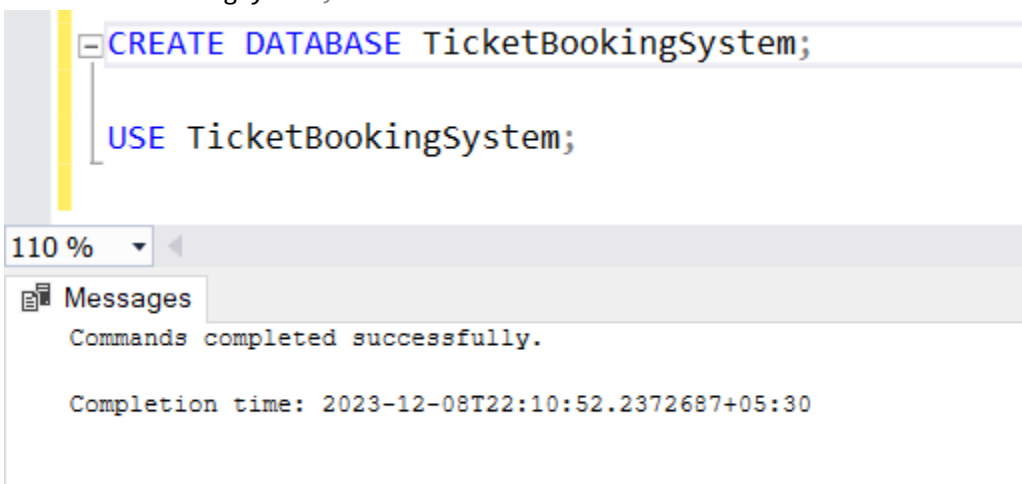
# Ticket Booking System

**Tasks 1: Database Design:**

1. Create the database named "TicketBookingSystem"

Ans)
```sql
CREATE DATABASE TicketBookingSystem;
```

```sql
USE TicketBookingSystem;
```




2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.
   - Venu
   - Event
   - Customers
   - Booking

Ans)

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

-- Create Event Table
CREATE TABLE Event (
    event_id INT PRIMARY KEY,
```

```sql
    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
);

-- Create 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
);

-- Create Customer Table
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    email VARCHAR(255),
    phone_number VARCHAR(15),
    booking_id INT
);
```

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

-- Create 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
);
```

110 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-08T22:26:06.0825897+05:30
```

```sql
-- Create 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
);

-- Create Customer Table
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    email VARCHAR(255),
    phone_number VARCHAR(15),
    booking_id INT
);
```

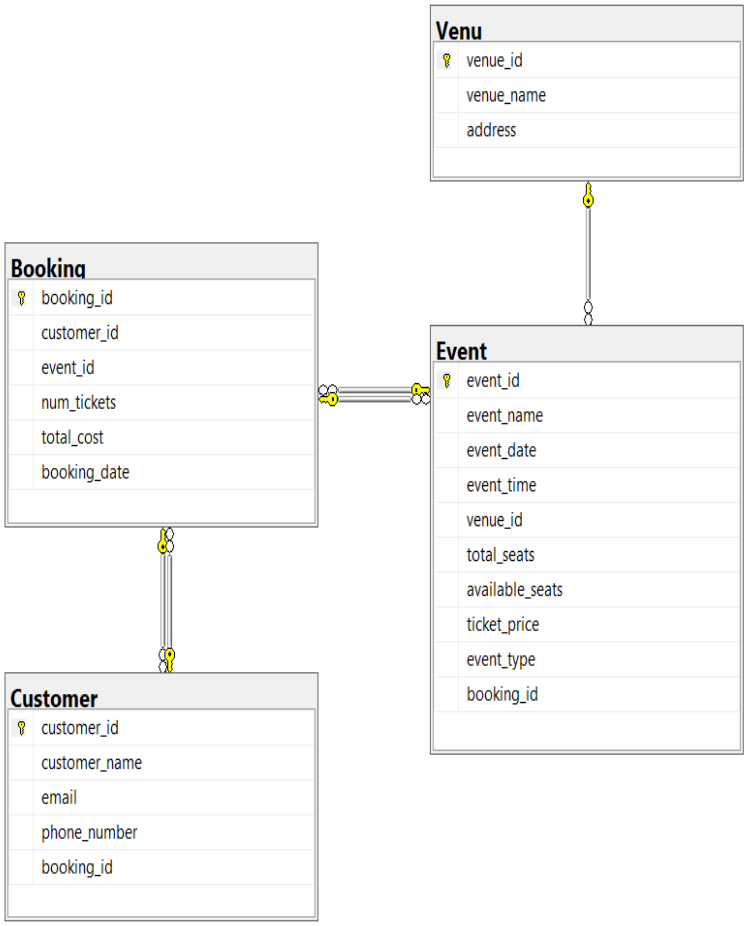
110 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-08T22:26:06.0825897+05:30
```

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




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

```
ALTER TABLE Customer

ADD FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);


ALTER TABLE Booking

ADD FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),

    FOREIGN KEY (event_id) REFERENCES Event(event_id);


ALTER TABLE Event

ADD FOREIGN KEY (venue_id) REFERENCES Venu(venue_id),

    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
```

```sql
-- Add CHECK constraint for event_type


ALTER TABLE Event

ADD CHECK (event_type IN ('Movie', 'Sports', 'Concert'));
```

```sql
-- Add Foreign Key Constraints using ALTER TABLE
ALTER TABLE Customer
ADD FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);

ALTER TABLE Booking
ADD FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (event_id) REFERENCES Event(event_id);

ALTER TABLE Event
ADD FOREIGN KEY (venue_id) REFERENCES Venu(venue_id),
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);

-- Add CHECK constraint for event_type
ALTER TABLE Event
ADD CHECK (event_type IN ('Movie', 'Sports', 'Concert'));
```

10 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-08T22:31:59.6282420+05:30
```

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

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

```sql
-- Insert sample records into Venu Table

INSERT INTO Venu (venue_id, venue_name, address) VALUES

(1, 'Grand Theater', '123 Main Street, Cityville'),
```

```sql
(2, 'City Arena', '456 Center Avenue, Townsville'),

(3, 'Sports Stadium', '789 Stadium Road, Sportstown'),

(4, 'Film Palace', '101 Movie Lane, Cinemaville'),

(5, 'Concert Hall', '202 Melody Street, Harmonytown'),

(6, 'Community Center', '303 Social Square, Gatherburg'),

(7, 'Live Lounge', '404 Entertainment Avenue, Showville'),

(8, 'Cinematic Complex', '505 Film Street, Filmington'),

(9, 'Soccer Park', '606 Goal Street, Kicksville'),

(10, 'Music Dome', '707 Harmony Road, Concertburg');


-- Insert sample records into Event Table

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, 'Movie Night: Inception', '2023-01-15', '18:00:00', 1, 150, 120, 2220.00, 'Movie', NULL),

(2, 'Concert: Acoustic Vibes', '2023-02-20', '20:00:00', 2, 300, 250, 1235.00, 'Concert',
NULL),

(3, 'Soccer Match: City Rivals', '2023-03-25', '19:30:00', 3, 200, 180, 1525.00, 'Sports',
NULL),

(4, 'Movie Night: The Great Gatsby', '2023-04-10', '21:00:00', 4, 120, 80, 1555.00, 'Movie',
NULL),

(5, 'Concert: Pop Explosion', '2023-05-05', '17:45:00', 5, 250, 200, 3330.00, 'Concert', NULL),

(6, 'Live Music: Jazz Evening', '2023-06-12', '19:00:00', 6, 300, 280, 1440.00, 'Concert',
NULL),

(7, 'Basketball Game: Finals', '2023-07-08', '18:30:00', 7, 350, 300, 1330.00, 'Sports', NULL),

(8, 'Movie Night: Casablanca', '2023-08-20', '20:15:00', 8, 150, 120, 2220.00, 'Movie', NULL),

(9, 'Soccer Match: International Clash', '2023-09-18', '19:45:00', 9, 200, 180, 1225.00,
'Sports', NULL),

(10, 'Concert: Rock Revolution', '2023-10-30', '22:00:00', 10, 250, 200, 3310.00, 'Concert',
NULL);


-- Insert sample records into Customer Table

INSERT INTO Customer (customer_id, customer_name, email, phone_number, booking_id) VALUES

(1, 'John Doe', 'john.doe@email.com', '555-1234', NULL),

(2, 'Jane Smith', 'jane.smith@email.com', '555-5678', NULL),
```

```sql
(3, 'Robert Johnson', 'robert.j@email.com', '555-9012', NULL),
(4, 'Samantha Brown', 'samantha.b@email.com', '555-3456', NULL),
(5, 'Chris Miller', 'chris.m@email.com', '555-7890', NULL),
(6, 'Emma White', 'emma.w@email.com', '555-2345', NULL),
(7, 'Michael Davis', 'michael.d@email.com', '555-6789', NULL),
(8, 'Olivia Taylor', 'olivia.t@email.com', '555-1234', NULL),
(9, 'Daniel Wilson', 'daniel.w@email.com', '555-5678', NULL),
(10, 'Sophia Adams', 'sophia.a@email.com', '555-9012', NULL);

-- Insert sample records into Booking Table
INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date)
VALUES
(1, 1, 1, 2, 4440.00, '2023-01-15'),
(2, 2, 2, 3, 3705.00, '2023-02-20'),
(3, 3, 3, 1, 1525.00, '2023-03-25'),
(4, 4, 4, 4, 6220.00, '2023-04-10'),
(5, 5, 5, 2, 6660.00, '2023-05-05'),
(6, 6, 6, 3, 4320.00, '2023-06-12'),
(7, 7, 7, 5, 6650.00, '2023-07-08'),
(8, 8, 8, 1, 2220.00, '2023-08-20'),
(9, 9, 9, 2, 2450.00, '2023-09-18'),
(10, 10, 10, 3, 9930.00, '2023-10-30');

-- Update Booking Table with correct booking_id values
UPDATE Event SET booking_id = 1 WHERE event_id = 1;
UPDATE Event SET booking_id = 2 WHERE event_id = 2;
UPDATE Event SET booking_id = 3 WHERE event_id = 3;
UPDATE Event SET booking_id = 4 WHERE event_id = 4;
UPDATE Event SET booking_id = 5 WHERE event_id = 5;
UPDATE Event SET booking_id = 6 WHERE event_id = 6;
UPDATE Event SET booking_id = 7 WHERE event_id = 7;
```

```sql
UPDATE Event SET booking_id = 8 WHERE event_id = 8;

UPDATE Event SET booking_id = 9 WHERE event_id = 9;

UPDATE Event SET booking_id = 10 WHERE event_id = 10;


-- Update Customer Table with correct booking_id values

UPDATE Customer SET booking_id = 1 WHERE customer_id = 1;

UPDATE Customer SET booking_id = 2 WHERE customer_id = 2;

UPDATE Customer SET booking_id = 3 WHERE customer_id = 3;

UPDATE Customer SET booking_id = 4 WHERE customer_id = 4;

UPDATE Customer SET booking_id = 5 WHERE customer_id = 5;

UPDATE Customer SET booking_id = 6 WHERE customer_id = 6;

UPDATE Customer SET booking_id = 7 WHERE customer_id = 7;

UPDATE Customer SET booking_id = 8 WHERE customer_id = 8;

UPDATE Customer SET booking_id = 9 WHERE customer_id = 9;

UPDATE Customer SET booking_id = 10 WHERE customer_id = 10;
```

```sql
-- Insert sample records into Venu Table
INSERT INTO Venu (venue_id, venue_name, address) VALUES
(1, 'Grand Theater', '123 Main Street, Cityville'),
(2, 'City Arena', '456 Center Avenue, Townsville'),
(3, 'Sports Stadium', '789 Stadium Road, Sportstown'),
(4, 'Film Palace', '101 Movie Lane, Cinemaville'),
(5, 'Concert Hall', '202 Melody Street, Harmonytown'),
(6, 'Community Center', '303 Social Square, Gatherburg'),
(7, 'Live Lounge', '404 Entertainment Avenue, Showville'),
(8, 'Cinematic Complex', '505 Film Street, Filmington'),
(9, 'Soccer Park', '606 Goal Street, Kicksville'),
(10, 'Music Dome', '707 Harmony Road, Concertburg');

-- Insert sample records into Event Table
INSERT INTO Event (event_id, event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type, booking_id)
(1, 'Movie Night: Inception', '2023-01-15', '18:00:00', 1, 150, 120, 2220.00, 'Movie', NULL),
(2, 'Concert: Acoustic Vibes', '2023-02-20', '20:00:00', 2, 300, 250, 1235.00, 'Concert', NULL),
(3, 'Soccer Match: City Rivals', '2023-03-25', '19:30:00', 3, 200, 180, 1525.00, 'Sports', NULL),
(4, 'Movie Night: The Great Gatsby', '2023-04-10', '21:00:00', 4, 120, 80, 1555.00, 'Movie', NULL),
(5, 'Concert: Pop Explosion', '2023-05-05', '17:45:00', 5, 250, 200, 3330.00, 'Concert', NULL),
(6, 'Live Music: Jazz Evening', '2023-06-12', '19:00:00', 6, 300, 280, 1440.00, 'Concert', NULL),
(7, 'Basketball Game: Finals', '2023-07-08', '18:30:00', 7, 350, 300, 1330.00, 'Sports', NULL),
(8, 'Movie Night: Casablanca', '2023-08-20', '20:15:00', 8, 150, 120, 2220.00, 'Movie', NULL),
(9, 'Soccer Match: International Clash', '2023-09-18', '19:45:00', 9, 200, 180, 1225.00, 'Sports', NULL),
(10, 'Concert: Rock Revolution', '2023-10-30', '22:00:00', 10, 250, 200, 3310.00, 'Concert', NULL);
```

.10 %

Messages

```
(10 rows affected)

(10 rows affected)

(10 rows affected)

(10 rows affected)
```

```sql
-- Insert sample records into Customer Table
INSERT INTO Customer (customer_id, customer_name, email, phone_number, booking_id) VALUES
(1, 'John Doe', 'john.doe@email.com', '555-1234', NULL),
(2, 'Jane Smith', 'jane.smith@email.com', '555-5678', NULL),
(3, 'Robert Johnson', 'robert.j@email.com', '555-9012', NULL),
(4, 'Samantha Brown', 'samantha.b@email.com', '555-3456', NULL),
(5, 'Chris Miller', 'chris.m@email.com', '555-7890', NULL),
(6, 'Emma White', 'emma.w@email.com', '555-2345', NULL),
(7, 'Michael Davis', 'michael.d@email.com', '555-6789', NULL),
(8, 'Olivia Taylor', 'olivia.t@email.com', '555-1234', NULL),
(9, 'Daniel Wilson', 'daniel.w@email.com', '555-5678', NULL),
(10, 'Sophia Adams', 'sophia.a@email.com', '555-9012', NULL);

-- Insert sample records into Booking Table
INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date) VALUES
(1, 1, 1, 2, 4440.00, '2023-01-15'),
(2, 2, 2, 3, 3705.00, '2023-02-20'),
(3, 3, 3, 1, 1525.00, '2023-03-25'),
(4, 4, 4, 4, 6220.00, '2023-04-10'),
(5, 5, 5, 2, 6660.00, '2023-05-05'),
(6, 6, 6, 3, 4320.00, '2023-06-12'),
(7, 7, 7, 5, 6650.00, '2023-07-08'),
(8, 8, 8, 1, 2220.00, '2023-08-20'),
(9, 9, 9, 2, 2450.00, '2023-09-18'),
(10, 10, 10, 3, 9930.00, '2023-10-30');
```

10 %

Messages

```
(10 rows affected)

(10 rows affected)

(10 rows affected)

(10 rows affected)
```

```sql
-- Update Booking Table with correct booking_id values
UPDATE Event SET booking_id = 1 WHERE event_id = 1;
UPDATE Event SET booking_id = 2 WHERE event_id = 2;
UPDATE Event SET booking_id = 3 WHERE event_id = 3;
UPDATE Event SET booking_id = 4 WHERE event_id = 4;
UPDATE Event SET booking_id = 5 WHERE event_id = 5;
UPDATE Event SET booking_id = 6 WHERE event_id = 6;
UPDATE Event SET booking_id = 7 WHERE event_id = 7;
UPDATE Event SET booking_id = 8 WHERE event_id = 8;
UPDATE Event SET booking_id = 9 WHERE event_id = 9;
UPDATE Event SET booking_id = 10 WHERE event_id = 10;

-- Update Customer Table with correct booking_id values
UPDATE Customer SET booking_id = 1 WHERE customer_id = 1;
UPDATE Customer SET booking_id = 2 WHERE customer_id = 2;
UPDATE Customer SET booking_id = 3 WHERE customer_id = 3;
UPDATE Customer SET booking_id = 4 WHERE customer_id = 4;
UPDATE Customer SET booking_id = 5 WHERE customer_id = 5;
UPDATE Customer SET booking_id = 6 WHERE customer_id = 6;
UPDATE Customer SET booking_id = 7 WHERE customer_id = 7;
UPDATE Customer SET booking_id = 8 WHERE customer_id = 8;
UPDATE Customer SET booking_id = 9 WHERE customer_id = 9;
UPDATE Customer SET booking_id = 10 WHERE customer_id = 10;
```

0 %

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2023-12-08T23:34:30.1337607+05:30
```

2. Write a SQL query to list all Events.

Ans) `SELECT * FROM Event;`

```sql
SELECT * FROM Event;
```

110 %

Results | Messages

| | event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Movie Night: Inception | 2023-01-15 | 18:00:00.0000000 | 1 | 150 | 120 | 2220.00 | Movie | 1 |
| 2 | 2 | Concert: Acoustic Vibes | 2023-02-20 | 20:00:00.0000000 | 2 | 300 | 250 | 1235.00 | Concert | 2 |
| 3 | 3 | Soccer Match: City Rivals | 2023-03-25 | 19:30:00.0000000 | 3 | 200 | 180 | 1525.00 | Sports | 3 |
| 4 | 4 | Movie Night: The Great Gatsby | 2023-04-10 | 21:00:00.0000000 | 4 | 120 | 80 | 1555.00 | Movie | 4 |
| 5 | 5 | Concert: Pop Explosion | 2023-05-05 | 17:45:00.0000000 | 5 | 250 | 200 | 3330.00 | Concert | 5 |
| 6 | 6 | Live Music: Jazz Evening | 2023-06-12 | 19:00:00.0000000 | 6 | 300 | 280 | 1440.00 | Concert | 6 |
| 7 | 7 | Basketball Game: Finals | 2023-07-08 | 18:30:00.0000000 | 7 | 350 | 300 | 1330.00 | Sports | 7 |
| 8 | 8 | Movie Night: Casablanca | 2023-08-20 | 20:15:00.0000000 | 8 | 150 | 120 | 2220.00 | Movie | 8 |
| 9 | 9 | Soccer Match: International ... | 2023-09-18 | 19:45:00.0000000 | 9 | 200 | 180 | 1225.00 | Sports | 9 |
| 10 | 10 | Concert: Rock Revolution | 2023-10-30 | 22:00:00.0000000 | 10 | 250 | 200 | 3310.00 | Concert | 10 |

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

Ans) `SELECT * FROM Event WHERE available_seats > 0;`

```
SELECT * FROM Event WHERE available_seats > 0;
```

110 %

Results | Messages

| | event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Movie Night: Inception | 2023-01-15 | 18:00:00.0000000 | 1 | 150 | 120 | 2220.00 | Movie | 1 |
| 2 | 2 | Concert: Acoustic Vibes | 2023-02-20 | 20:00:00.0000000 | 2 | 300 | 250 | 1235.00 | Concert | 2 |
| 3 | 3 | Soccer Match: City Rivals | 2023-03-25 | 19:30:00.0000000 | 3 | 200 | 180 | 1525.00 | Sports | 3 |
| 4 | 4 | Movie Night: The Great Gatsby | 2023-04-10 | 21:00:00.0000000 | 4 | 120 | 80 | 1555.00 | Movie | 4 |
| 5 | 5 | Concert: Pop Explosion | 2023-05-05 | 17:45:00.0000000 | 5 | 250 | 200 | 3330.00 | Concert | 5 |
| 6 | 6 | Live Music: Jazz Evening | 2023-06-12 | 19:00:00.0000000 | 6 | 300 | 280 | 1440.00 | Concert | 6 |
| 7 | 7 | Basketball Game: Finals | 2023-07-08 | 18:30:00.0000000 | 7 | 350 | 300 | 1330.00 | Sports | 7 |
| 8 | 8 | Movie Night: Casablanca | 2023-08-20 | 20:15:00.0000000 | 8 | 150 | 120 | 2220.00 | Movie | 8 |
| 9 | 9 | Soccer Match: International Clash | 2023-09-18 | 19:45:00.0000000 | 9 | 200 | 180 | 1225.00 | Sports | 9 |
| 10 | 10 | Concert: Rock Revolution | 2023-10-30 | 22:00:00.0000000 | 10 | 250 | 200 | 3310.00 | Concert | 10 |

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

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

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

110 %

Results | Messages

| event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
|---|---|---|---|---|---|---|---|---|---|

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

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

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

110 %

Results | Messages

| | event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Movie Night: Inception | 2023-01-15 | 18:00:00.0000000 | 1 | 150 | 120 | 2220.00 | Movie | 1 |
| 2 | 2 | Concert: Acoustic Vibes | 2023-02-20 | 20:00:00.0000000 | 2 | 300 | 250 | 1235.00 | Concert | 2 |
| 3 | 3 | Soccer Match: City Rivals | 2023-03-25 | 19:30:00.0000000 | 3 | 200 | 180 | 1525.00 | Sports | 3 |
| 4 | 4 | Movie Night: The Great Gatsby | 2023-04-10 | 21:00:00.0000000 | 4 | 120 | 80 | 1555.00 | Movie | 4 |
| 5 | 6 | Live Music: Jazz Evening | 2023-06-12 | 19:00:00.0000000 | 6 | 300 | 280 | 1440.00 | Concert | 6 |
| 6 | 7 | Basketball Game: Finals | 2023-07-08 | 18:30:00.0000000 | 7 | 350 | 300 | 1330.00 | Sports | 7 |
| 7 | 8 | Movie Night: Casablanca | 2023-08-20 | 20:15:00.0000000 | 8 | 150 | 120 | 2220.00 | Movie | 8 |
| 8 | 9 | Soccer Match: International Clash | 2023-09-18 | 19:45:00.0000000 | 9 | 200 | 180 | 1225.00 | Sports | 9 |

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

Ans) `SELECT * FROM Event WHERE event_date BETWEEN '2023-01-01' AND '2023-05-31';`

```
SELECT * FROM Event WHERE event_date BETWEEN '2023-01-01' AND '2023-05-31';
```

110 %

Results | Messages

| | event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Movie Night: Inception | 2023-01-15 | 18:00:00.0000000 | 1 | 150 | 120 | 2220.00 | Movie | 1 |
| 2 | 2 | Concert: Acoustic Vibes | 2023-02-20 | 20:00:00.0000000 | 2 | 300 | 250 | 1235.00 | Concert | 2 |
| 3 | 3 | Soccer Match: City Rivals | 2023-03-25 | 19:30:00.0000000 | 3 | 200 | 180 | 1525.00 | Sports | 3 |
| 4 | 4 | Movie Night: The Great Gatsby | 2023-04-10 | 21:00:00.0000000 | 4 | 120 | 80 | 1555.00 | Movie | 4 |
| 5 | 5 | Concert: Pop Explosion | 2023-05-05 | 17:45:00.0000000 | 5 | 250 | 200 | 3330.00 | Concert | 5 |

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

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

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

110 %

Results | Messages

| | event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | Concert: Acoustic Vibes | 2023-02-20 | 20:00:00.0000000 | 2 | 300 | 250 | 1235.00 | Concert | 2 |
| 2 | 5 | Concert: Pop Explosion | 2023-05-05 | 17:45:00.0000000 | 5 | 250 | 200 | 3330.00 | Concert | 5 |
| 3 | 10 | Concert: Rock Revolution | 2023-10-30 | 22:00:00.0000000 | 10 | 250 | 200 | 3310.00 | Concert | 10 |

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

Ans) `SELECT * FROM Customer ORDER BY customer_id OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;`

```
SELECT * FROM Customer ORDER BY customer_id OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```

110 %

Results | Messages

| | customer_id | customer_name | email | phone_number | booking_id |
|---|---|---|---|---|---|
| 1 | 6 | Emma White | emma.w@email.com | 555-2345 | 6 |
| 2 | 7 | Michael Davis | michael.d@email.com | 555-6789 | 7 |
| 3 | 8 | Olivia Taylor | olivia.t@email.com | 555-1234 | 8 |
| 4 | 9 | Daniel Wilson | daniel.w@email.com | 555-5678 | 9 |
| 5 | 10 | Sophia Adams | sophia.a@email.com | 555-9012 | 10 |

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

   Ans) `SELECT * FROM Booking WHERE num_tickets > 4;`

```
SELECT * FROM Booking WHERE num_tickets > 4;
```

110 %

Results | Messages

| | booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
|---|---|---|---|---|---|---|
| 1 | 7 | 7 | 7 | 5 | 6650.00 | 2023-07-08 |

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

Ans) `SELECT * FROM Customer WHERE phone_number LIKE '%000';`

```
SELECT * FROM Customer WHERE phone_number LIKE '%000';
```

110 %

Results | Messages

| customer_id | customer_name | email | phone_number | booking_id |
|---|---|---|---|---|

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

Ans) `SELECT * FROM Event WHERE total_seats > 15000 ORDER BY total_seats;`

```
SELECT * FROM Event WHERE total_seats > 15000 ORDER BY total_seats;
```

110 %

Results | Messages

| event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_ |
|---|---|---|---|---|---|---|---|---|---|

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

Ans) `SELECT * FROM Event WHERE NOT event_name LIKE '[x-z]%';`

```
SELECT * FROM Event WHERE NOT event_name LIKE '[x-z]%';
```

110 %

Results | Messages

| | event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Movie Night: Inception | 2023-01-15 | 18:00:00.0000000 | 1 | 150 | 120 | 2220.00 | Movie | 1 |
| 2 | 2 | Concert: Acoustic Vibes | 2023-02-20 | 20:00:00.0000000 | 2 | 300 | 250 | 1235.00 | Concert | 2 |
| 3 | 3 | Soccer Match: City Rivals | 2023-03-25 | 19:30:00.0000000 | 3 | 200 | 180 | 1525.00 | Sports | 3 |
| 4 | 4 | Movie Night: The Great Gatsby | 2023-04-10 | 21:00:00.0000000 | 4 | 120 | 80 | 1555.00 | Movie | 4 |
| 5 | 5 | Concert: Pop Explosion | 2023-05-05 | 17:45:00.0000000 | 5 | 250 | 200 | 3330.00 | Concert | 5 |
| 6 | 6 | Live Music: Jazz Evening | 2023-06-12 | 19:00:00.0000000 | 6 | 300 | 280 | 1440.00 | Concert | 6 |
| 7 | 7 | Basketball Game: Finals | 2023-07-08 | 18:30:00.0000000 | 7 | 350 | 300 | 1330.00 | Sports | 7 |
| 8 | 8 | Movie Night: Casablanca | 2023-08-20 | 20:15:00.0000000 | 8 | 150 | 120 | 2220.00 | Movie | 8 |
| 9 | 9 | Soccer Match: International Clash | 2023-09-18 | 19:45:00.0000000 | 9 | 200 | 180 | 1225.00 | Sports | 9 |
| 10 | 10 | Concert: Rock Revolution | 2023-10-30 | 22:00:00.0000000 | 10 | 250 | 200 | 3310.00 | Concert | 10 |

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

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

Ans)
```
SELECT event_name, AVG(ticket_price) AS average_ticket_price
FROM Event GROUP BY event_name;
```

```sql
SELECT event_name, AVG(ticket_price) AS average_ticket_price
FROM Event GROUP BY event_name;
```

110 %

Results | Messages

| | event_name | average_ticket_price |
|---|---|---|
| 1 | Basketball Game: Finals | 1330.000000 |
| 2 | Concert: Acoustic Vibes | 1235.000000 |
| 3 | Concert: Pop Explosion | 3330.000000 |
| 4 | Concert: Rock Revolution | 3310.000000 |
| 5 | Live Music: Jazz Evening | 1440.000000 |
| 6 | Movie Night: Casablanca | 2220.000000 |
| 7 | Movie Night: Inception | 2220.000000 |
| 8 | Movie Night: The Great Gatsby | 1555.000000 |
| 9 | Soccer Match: City Rivals | 1525.000000 |
| 10 | Soccer Match: International Clash | 1225.000000 |

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

Ans) `SELECT SUM(total_cost) AS total_revenue FROM Booking;`

```sql
SELECT SUM(total_cost) AS total_revenue FROM Booking;
```

.0 %

Results | Messages

| total_revenue |
|---|
| 48120.00 |

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

Ans)
```sql
SELECT TOP 1 event_id, SUM(num_tickets) AS total_tickets_sold FROM Booking
GROUP BY event_id ORDER BY total_tickets_sold DESC;
```

```sql
SELECT TOP 1 event_id, SUM(num_tickets) AS total_tickets_sold FROM Booking
GROUP BY event_id ORDER BY total_tickets_sold DESC;
```

110 %

Results | Messages

| | event_id | total_tickets_sold |
|---|---|---|
| 1 | 7 | 5 |

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

Ans)
```sql
SELECT event_id, SUM(num_tickets) AS total_tickets_sold
FROM Booking GROUP BY event_id;
```

```sql
SELECT event_id, SUM(num_tickets) AS total_tickets_sold
FROM Booking GROUP BY event_id;
```

110 %

Results | Messages

| | event_id | total_tickets_sold |
|---|---|---|
| 1 | 1 | 2 |
| 2 | 2 | 3 |
| 3 | 3 | 1 |
| 4 | 4 | 4 |
| 5 | 5 | 2 |
| 6 | 6 | 3 |
| 7 | 7 | 5 |
| 8 | 8 | 1 |
| 9 | 9 | 2 |
| 10 | 10 | 3 |

5. Write a SQL query to Find Events with No Ticket Sales.
   Ans)
```sql
SELECT event_id, event_name FROM Event

WHERE event_id NOT IN (SELECT DISTINCT event_id FROM Booking);
```

```sql
SELECT event_id, event_name FROM Event
WHERE event_id NOT IN (SELECT DISTINCT event_id FROM Booking);
```

110 %

Results | Messages

| event_id | event_name |
|---|---|

6. Write a SQL query to Find the User Who Has Booked the Most Tickets.
   Ans)
```sql
SELECT TOP 1 c.customer_id, c.customer_name, COUNT(b.booking_id) AS
total_tickets_booked FROM Customer c

JOIN Booking b ON c.customer_id = b.customer_id

GROUP BY c.customer_id, c.customer_name

ORDER BY total_tickets_booked DESC;
```

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

110 %

Results | Messages

| | customer_id | customer_name | total_tickets_booked |
|---|---|---|---|
| 1 | 2 | Jane Smith | 1 |

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

Ans) `SELECT MONTH(booking_date) AS month, event_id, SUM(num_tickets) AS total_tickets_sold FROM Booking GROUP BY MONTH(booking_date), event_id;`

```sql
SELECT MONTH(booking_date) AS month, event_id, SUM(num_tickets) AS total_tickets_sol
FROM Booking GROUP BY MONTH(booking_date), event_id;
```

110 %

Results | Messages

| | month | event_id | total_tickets_sold |
|---|---|---|---|
| 1 | 1 | 1 | 2 |
| 2 | 2 | 2 | 3 |
| 3 | 3 | 3 | 1 |
| 4 | 4 | 4 | 4 |
| 5 | 5 | 5 | 2 |
| 6 | 6 | 6 | 3 |
| 7 | 7 | 7 | 5 |
| 8 | 8 | 8 | 1 |
| 9 | 9 | 9 | 2 |
| 10 | 10 | 10 | 3 |

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

Ans)
```sql
SELECT v.venue_id, v.venue_name, AVG(e.ticket_price) AS
average_ticket_price

FROM Venu v JOIN Event e ON v.venue_id = e.venue_id

GROUP BY v.venue_id, v.venue_name;
```

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

.10 %

Results | Messages

| | venue_id | venue_name | average_ticket_price |
|---|---|---|---|
| 1 | 1 | Grand Theater | 2220.000000 |
| 2 | 2 | City Arena | 1235.000000 |
| 3 | 3 | Sports Stadium | 1525.000000 |
| 4 | 4 | Film Palace | 1555.000000 |
| 5 | 5 | Concert Hall | 3330.000000 |
| 6 | 6 | Community Center | 1440.000000 |
| 7 | 7 | Live Lounge | 1330.000000 |
| 8 | 8 | Cinematic Complex | 2220.000000 |
| 9 | 9 | Soccer Park | 1225.000000 |
| 10 | 10 | Music Dome | 3310.000000 |

9. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.
   Ans)
```sql
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;
```

```sql
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;
```

110 %

Results | Messages

| | event_type | total_tickets_sold |
|---|---|---|
| 1 | Concert | 11 |
| 2 | Movie | 7 |
| 3 | Sports | 8 |

10. Write a SQL query to calculate the total Revenue Generated by Events in Each Year.
    Ans)
```sql
SELECT YEAR(booking_date) AS year, SUM(total_cost) AS total_revenue
```

```
FROM Booking GROUP BY YEAR(booking_date);
```

```
SELECT YEAR(booking_date) AS year, SUM(total_cost) AS total_revenu
FROM Booking
GROUP BY YEAR(booking_date);
```

110 %

Results | Messages

| | year | total_revenue |
|---|---|---|
| 1 | 2023 | 48120.00 |

11. Write a SQL query to list users who have booked tickets for multiple events.
Ans)
```
SELECT c.customer_id, c.customer_name

FROM Customer c

JOIN Booking b ON c.customer_id = b.customer_id

GROUP BY c.customer_id, c.customer_name

HAVING COUNT(DISTINCT b.event_id) > 1;
```

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

110 %

Results | Messages

| customer_id | customer_name |
|---|---|

12. Write a SQL query to calculate the Total Revenue Generated by Events for Each User.
Ans)
```
SELECT c.customer_id, c.customer_name, SUM(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;
```

```sql
SELECT c.customer_id, c.customer_name, SUM(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;
```

110 %

Results | Messages

| | customer_id | customer_name | total_revenue |
|---|---|---|---|
| 1 | 1 | John Doe | 4440.00 |
| 2 | 2 | Jane Smith | 3705.00 |
| 3 | 3 | Robert Johnson | 1525.00 |
| 4 | 4 | Samantha Brown | 6220.00 |
| 5 | 5 | Chris Miller | 6660.00 |
| 6 | 6 | Emma White | 4320.00 |
| 7 | 7 | Michael Davis | 6650.00 |
| 8 | 8 | Olivia Taylor | 2220.00 |
| 9 | 9 | Daniel Wilson | 2450.00 |
| 10 | 10 | Sophia Adams | 9930.00 |

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

Ans)
```sql
SELECT v.venue_id, v.venue_name, e.event_type, AVG(e.ticket_price) AS average_ticket_price FROM Venu v

JOIN Event e ON v.venue_id = e.venue_id

GROUP BY v.venue_id, v.venue_name, e.event_type;
```

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

110 %

Results | Messages

| | venue_id | venue_name | event_type | average_ticket_price |
|---|---|---|---|---|
| 1 | 2 | City Arena | Concert | 1235.000000 |
| 2 | 5 | Concert Hall | Concert | 3330.000000 |
| 3 | 6 | Community Center | Concert | 1440.000000 |
| 4 | 10 | Music Dome | Concert | 3310.000000 |
| 5 | 1 | Grand Theater | Movie | 2220.000000 |
| 6 | 4 | Film Palace | Movie | 1555.000000 |
| 7 | 8 | Cinematic Complex | Movie | 2220.000000 |
| 8 | 3 | Sports Stadium | Sports | 1525.000000 |
| 9 | 7 | Live Lounge | Sports | 1330.000000 |
| 10 | 9 | Soccer Park | Sports | 1225.000000 |

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

Ans)
```sql
SELECT c.customer_id, c.customer_name, COUNT(b.booking_id) AS total_tickets_purchased

FROM Customer c
```

```sql
JOIN Booking b ON c.customer_id = b.customer_id

WHERE b.booking_date >= DATEADD(DAY, -30, GETDATE())

GROUP BY c.customer_id, c.customer_name;
```

```sql
INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date) VALU
(11, 1, 1, 2, 4440.00, '2023-11-15'),
(12, 2, 2, 3, 3705.00, '2023-12-20');

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

0 %

Results | Messages

| customer_id | customer_name | total_tickets_purchased |
|---|---|---|
| 1 | John Doe | 1 |
| 2 | Jane Smith | 1 |

**Tasks 4: Subquery and its types**

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

Ans)
```sql
SELECT venue_id, venue_name, (

SELECT AVG(ticket_price)

FROM Event

WHERE venue_id = v.venue_id

) AS average_ticket_price FROM Venu v;
```

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

110 %

Results | Messages

| | venue_id | venue_name | average_ticket_price |
|---|---|---|---|
| 1 | 1 | Grand Theater | 2220.000000 |
| 2 | 2 | City Arena | 1235.000000 |
| 3 | 3 | Sports Stadium | 1525.000000 |
| 4 | 4 | Film Palace | 1555.000000 |
| 5 | 5 | Concert Hall | 3330.000000 |
| 6 | 6 | Community Center | 1440.000000 |
| 7 | 7 | Live Lounge | 1330.000000 |
| 8 | 8 | Cinematic Complex | 2220.000000 |
| 9 | 9 | Soccer Park | 1225.000000 |
| 10 | 10 | Music Dome | 3310.000000 |

2. Find Events with More Than 50% of Tickets Sold using subquery.
   Ans)
```sql
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;
```

```sql
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;
```

.0 %

Results | Messages

| event_id | event_name |
|---|---|

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

Ans)
```sql
SELECT event_id, event_name, (
  SELECT SUM(num_tickets) FROM Booking
WHERE Booking.event_id = Event.event_id
) AS total_tickets_sold FROM Event;
```

```sql
SELECT event_id, event_name, (
    SELECT SUM(num_tickets)
    FROM Booking
    WHERE Booking.event_id = Event.event_id
) AS total_tickets_sold
FROM Event;
```

110 %

Results | Messages

| | event_id | event_name | total_tickets_sold |
|---|---|---|---|
| 1 | 1 | Movie Night: Inception | 4 |
| 2 | 2 | Concert: Acoustic Vibes | 6 |
| 3 | 3 | Soccer Match: City Rivals | 1 |
| 4 | 4 | Movie Night: The Great Gatsby | 4 |
| 5 | 5 | Concert: Pop Explosion | 2 |
| 6 | 6 | Live Music: Jazz Evening | 3 |
| 7 | 7 | Basketball Game: Finals | 5 |
| 8 | 8 | Movie Night: Casablanca | 1 |
| 9 | 9 | Soccer Match: International Clash | 2 |
| 10 | 10 | Concert: Rock Revolution | 3 |

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

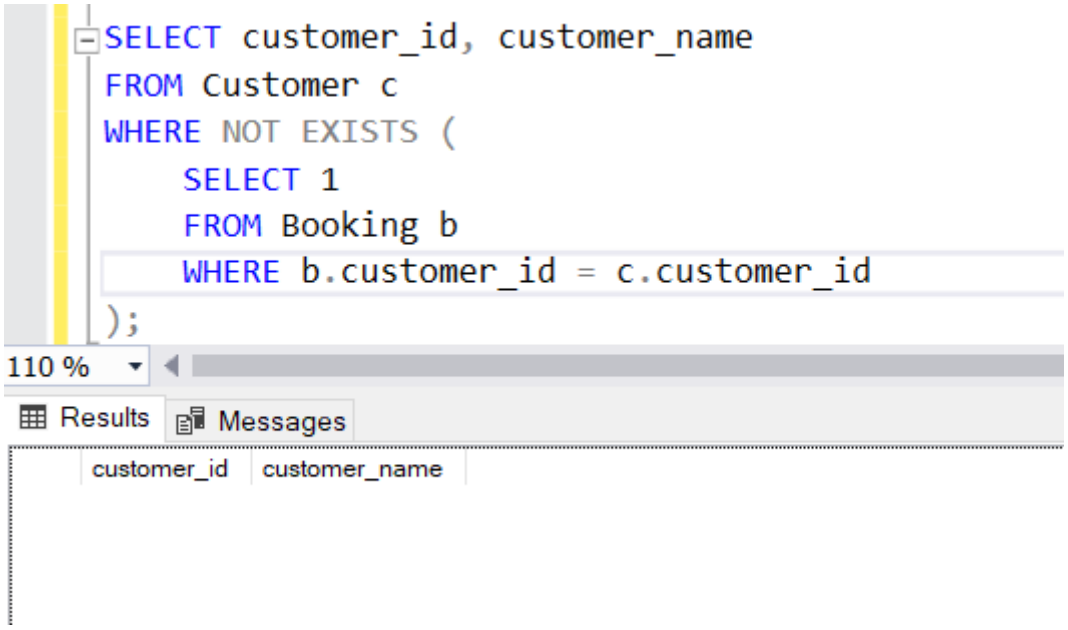
Ans)
```
SELECT customer_id, customer_name FROM Customer c

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.

Ans)
```
SELECT event_id, event_name FROM Event

WHERE event_id NOT IN (

    SELECT DISTINCT event_id

    FROM Booking );
```

```sql
SELECT event_id, event_name
FROM Event
WHERE event_id NOT IN (
    SELECT DISTINCT event_id
    FROM Booking
);
```

110 %

Results | Messages

| event_id | event_name |
|---|---|

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

Ans)
```sql
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;
```

```sql
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;
```

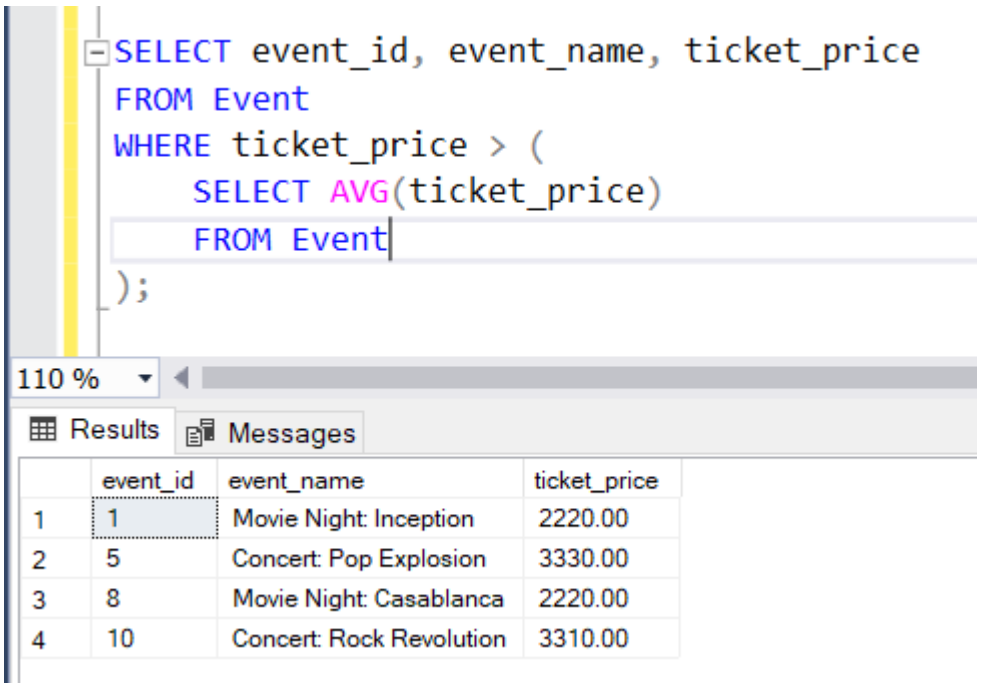
110 %

Results | Messages

| | event_type | total_tickets_sold |
|---|---|---|
| 1 | Concert | 14 |
| 2 | Movie | 9 |
| 3 | Sports | 8 |

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

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




| | event_id | event_name | ticket_price |
|---|---|---|---|
| 1 | 1 | Movie Night: Inception | 2220.00 |
| 2 | 5 | Concert: Pop Explosion | 3330.00 |
| 3 | 8 | Movie Night: Casablanca | 2220.00 |
| 4 | 10 | Concert: Rock Revolution | 3310.00 |

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

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

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

110 %

Results | Messages

| | customer_id | customer_name | total_revenue |
|---|---|---|---|
| 1 | 1 | John Doe | 8880.00 |
| 2 | 2 | Jane Smith | 7410.00 |
| 3 | 3 | Robert Johnson | 1525.00 |
| 4 | 4 | Samantha Brown | 6220.00 |
| 5 | 5 | Chris Miller | 6660.00 |
| 6 | 6 | Emma White | 4320.00 |
| 7 | 7 | Michael Davis | 6650.00 |
| 8 | 8 | Olivia Taylor | 2220.00 |
| 9 | 9 | Daniel Wilson | 2450.00 |
| 10 | 10 | Sophia Adams | 9930.00 |

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

Ans)
```sql
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 = 1

    AND Booking.customer_id = Customer.customer_id );
```

```sql
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 = 1
    AND Booking.customer_id = Customer.customer_id
);
```

Results | Messages

| customer_id | customer_name |
|---|---|
| 1 | John Doe |

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

Ans)
```sql
SELECT event_type, SUM(total_tickets_sold) AS total_tickets_sold
FROM ( SELECT 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;
```

```sql
SELECT event_type, SUM(total_tickets_sold) AS total_tickets_sold
FROM (
    SELECT 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;
```

110 % | Results | Messages

| | event_type | total_tickets_sold |
|---|---|---|
| 1 | Concert | 14 |
| 2 | Movie | 9 |
| 3 | Sports | 8 |

11. Find Users Who Have Booked Tickets for Events in each Month Using a Subquery withDATE_FORMAT.

Ans)
```
SELECT customer_id, customer_name FROM Customer c

WHERE EXISTS ( SELECT * FROM Booking b

  WHERE b.customer_id = c.customer_id

  AND FORMAT(b.booking_date, 'yyyy-MM') = '2023-01'  -- Replace with the
desired month );
```

```
SELECT customer_id, customer_name
FROM Customer c
WHERE EXISTS (
    SELECT *
    FROM Booking b
    WHERE b.customer_id = c.customer_id
    AND FORMAT(b.booking_date, 'yyyy-MM') = '2023-01'  -- Replace with the desired mont
);
```

110 %

Results | Messages

| | customer_id | customer_name |
|---|---|---|
| 1 | 1 | John Doe |

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

Ans)
```
SELECT venue_id, venue_name, (

  SELECT AVG(ticket_price)

  FROM Event

  WHERE venue_id = v.venue_id

) AS average_ticket_price FROM Venu v;
```

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

110 %

Results | Messages

| | venue_id | venue_name | average_ticket_price |
|---|---|---|---|
| 1 | 1 | Grand Theater | 2220.000000 |
| 2 | 2 | City Arena | 1235.000000 |
| 3 | 3 | Sports Stadium | 1525.000000 |
| 4 | 4 | Film Palace | 1555.000000 |
| 5 | 5 | Concert Hall | 3330.000000 |
| 6 | 6 | Community Center | 1440.000000 |
| 7 | 7 | Live Lounge | 1330.000000 |
| 8 | 8 | Cinematic Complex | 2220.000000 |
| 9 | 9 | Soccer Park | 1225.000000 |
| 10 | 10 | Music Dome | 3310.000000 |