Tasks 1: Database Design:

1.Create the database named "TicketBookingSystem"

```
mysql> CREATE DATABASE TicketBookingSystem;
Query OK, 1 row affected (0.02 sec)

mysql> USE TicketBookingSystem;
Database changed
```

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

• Venu • Event • Customers • Booking

```
mysql> CREATE TABLE Venue (
    ->     venue_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     venue_name VARCHAR(100) NOT NULL,
    ->     address VARCHAR(255) NOT NULL
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Event (
    ->     event_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     event_name VARCHAR(100) NOT NULL,
    ->     event_date DATE NOT NULL,
    ->     event_time TIME NOT NULL,
    ->     venue_id INT,
    ->     total_seats INT NOT NULL,
    ->     available_seats INT NOT NULL,
    ->     ticket_price DECIMAL(10,2) NOT NULL,
    ->     event_type ENUM('Movie', 'Sports', 'Concert') NOT NULL,
    ->     FOREIGN KEY (venue_id) REFERENCES Venue(venue_id)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE Customer (
    ->     customer_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     customer_name VARCHAR(100) NOT NULL,
    ->     email VARCHAR(100) NOT NULL,
    ->     phone_number VARCHAR(20) NOT NULL
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE Booking (
    ->     booking_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     customer_id INT,
    ->     event_id INT,
    ->     num_tickets INT NOT NULL,
    ->     total_cost DECIMAL(10,2) NOT NULL,
    ->     booking_date DATETIME NOT NULL,
    ->     FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    ->     FOREIGN KEY (event_id) REFERENCES Event(event_id)
    -> );
Query OK, 0 rows affected (0.10 sec)
```

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

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

Venue: Primary Key on `venue_id`

Event: Primary Key on `event_id`, Foreign Key on `venue_id` referencing `Venue(venue_id)`

Customer: Primary Key on `customer_id`

Booking: Primary Key on `booking_id`, Foreign Key on `customer_id` referencing `Customer(customer_id)`, Foreign Key on `event_id` referencing `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

```
mysql> INSERT INTO Venue (venue_name, address)
    -> VALUES
    ->     ('City Center Cinema', '123 Main Street'),
    ->     ('National Stadium', '456 Sports Avenue'),
    ->     ('Grand Concert Hall', '789 Music Lane'),
    ->     ('Theater Royale', '1011 Broadway Street'),
    ->     ('Arts Center', '1213 Art Boulevard'),
    ->     ('Amphitheater', '1415 Park Road'),
    ->     ('Civic Arena', '1617 Stadium Drive'),
    ->     ('The Rex', '1819 Entertainment Drive'),
    ->     ('The Music Box', '2021 Music Lane'),
    ->     ('The Dome', '2223 Sports Way');
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Event (event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type)
    -> VALUES
    ->     ('The Batman', '2024-01-20', '20:00:00', 1, 200, 200, 15.00, 'Movie'),
    ->     ('Avengers: Endgame', '2024-01-21', '19:00:00', 1, 150, 150, 12.50, 'Movie'),
    ->     ('World Cup Final', '2024-07-15', '16:00:00', 2, 50000, 50000, 250.00, 'Sports'),
    ->     ('Justin Bieber Concert', '2024-03-05', '21:00:00', 3, 1000, 1000, 100.00, 'Concert'),
    ->     ('Hamilton', '2024-02-14', '19:30:00', 4, 800, 800, 75.00, 'Movie'),
    ->     ('The Nutcracker', '2024-12-25', '15:00:00', 4, 500, 500, 50.00, 'Movie'),
    ->     ('Local Symphony Orchestra', '2024-05-10', '18:00:00', 3, 250, 250, 40.00, 'Concert'),
    ->     ('International Art Exhibition', '2024-06-01', '10:00:00', 5, 1000, 1000, 20.00, 'Movie'),
    ->     ('Summer Music Festival', '2024-08-15', '17:00:00', 6, 5000, 5000, 35.00, 'Concert'),
    ->     ('Tennis Open', '2024-09-01', '11:00:00', 7, 15000, 15000, 60.00, 'Sports');
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO Customer (customer_name, email, phone_number)VALUES    ('Alice Smith', 'alice@example.com', '1234567890'),    ('Bob
Johnson', 'bob@example.com', '9876543210'),    ('Charlie Brown', 'charlie@example.com', '5551234567'),    ('David Miller', 'david@examp
le.com', '8889991234'),    ('Emily Davis', 'emily@example.com', '7778889990'),    ('Frank Williams', 'frank@example.com', '4445556667')
    ('Grace Taylor', 'grace@example.com', '3334445556'),    ('Henry Jackson', 'henry@example.com', '2223334445'),    ('Isabella Cooper
', 'isabella@example.com', '1112223334'),    ('Jack Moore', 'jack@example.com', '0001112223');
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Booking (customer_id, event_id, num_tickets, total_cost, booking_date)VALUES    (1, 1, 2, 40.00, '2024-01-15 16:43:0
0'),    (2, 2, 3, 60.00, '2024-01-16 17:00:00'),    (3, 3, 1, 25.00, '2024-01-17 18:30:00'),    (4, 1, 4, 80.00, '2024-01-18 19:00:00')
,    (5, 2, 2, 40.00, '2024-01-19 20:00:00'),    (6, 3, 3, 75.00, '2024-01-20 21:00:00'),    (7, 4, 1, 25.00, '2024-01-21 16:00:00'),
 (8, 5, 2, 50.00, '2024-01-22 17:30:00'),    (9, 1, 3, 60.00, '2024-01-23 19:45:00'),    (10, 2, 4, 80.00, '2024-01-24 20:15:00');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

2. Write a SQL query to list all Events

```
mysql> SELECT *
    -> FROM Event;
+----------+-------------------------------+------------+------------+----------+-------------+----------------+--------------+--------
----+
| event_id | event_name                    | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_t
ype |
+----------+-------------------------------+------------+------------+----------+-------------+----------------+--------------+--------
----+
|        1 | The Batman                    | 2024-01-20 | 20:00:00   |        1 |         200 |             200 |        15.00 | Movie
    |
|        2 | Avengers: Endgame             | 2024-01-21 | 19:00:00   |        1 |         150 |             150 |        12.50 | Movie
    |
|        3 | World Cup Final               | 2024-07-15 | 16:00:00   |        2 |       50000 |           50000 |       250.00 | Sports
    |
|        4 | Justin Bieber Concert         | 2024-03-05 | 21:00:00   |        3 |        1000 |            1000 |       100.00 | Concert
    |
|        5 | Hamilton                      | 2024-02-14 | 19:30:00   |        4 |         800 |             800 |        75.00 | Movie
    |
|        6 | The Nutcracker                | 2024-12-25 | 15:00:00   |        4 |         500 |             500 |        50.00 | Movie
    |
|        7 | Local Symphony Orchestra      | 2024-05-10 | 18:00:00   |        3 |         250 |             250 |        40.00 | Concert
    |
|        8 | International Art Exhibition | 2024-06-01 | 10:00:00   |        5 |        1000 |            1000 |        20.00 | Movie
    |
|        9 | Summer Music Festival         | 2024-08-15 | 17:00:00   |        6 |        5000 |            5000 |        35.00 | Concert
    |
|       10 | Tennis Open                   | 2024-09-01 | 11:00:00   |        7 |       15000 |           15000 |        60.00 | Sports
    |
+----------+-------------------------------+------------+------------+----------+-------------+----------------+--------------+--------
----+
10 rows in set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE available_seats > 0;
+----------+-------------------------------+------------+------------+----------+-------------+----------------+--------------+--------
----+
| event_id | event_name                    | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_t
ype |
+----------+-------------------------------+------------+------------+----------+-------------+----------------+--------------+--------
----+
|        1 | The Batman                    | 2024-01-20 | 20:00:00   |        1 |         200 |             200 |        15.00 | Movie
    |
|        2 | Avengers: Endgame             | 2024-01-21 | 19:00:00   |        1 |         150 |             150 |        12.50 | Movie
    |
|        3 | World Cup Final               | 2024-07-15 | 16:00:00   |        2 |       50000 |           50000 |       250.00 | Sports
    |
|        4 | Justin Bieber Concert         | 2024-03-05 | 21:00:00   |        3 |        1000 |            1000 |       100.00 | Concert
    |
|        5 | Hamilton                      | 2024-02-14 | 19:30:00   |        4 |         800 |             800 |        75.00 | Movie
    |
|        6 | The Nutcracker                | 2024-12-25 | 15:00:00   |        4 |         500 |             500 |        50.00 | Movie
    |
|        7 | Local Symphony Orchestra      | 2024-05-10 | 18:00:00   |        3 |         250 |             250 |        40.00 | Concert
    |
|        8 | International Art Exhibition | 2024-06-01 | 10:00:00   |        5 |        1000 |            1000 |        20.00 | Movie
    |
|        9 | Summer Music Festival         | 2024-08-15 | 17:00:00   |        6 |        5000 |            5000 |        35.00 | Concert
    |
|       10 | Tennis Open                   | 2024-09-01 | 11:00:00   |        7 |       15000 |           15000 |        60.00 | Sports
    |
+----------+-------------------------------+------------+------------+----------+-------------+----------------+--------------+--------
----+
10 rows in set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_name LIKE '%cup%';
+----------+----------------+------------+------------+----------+-------------+----------------+--------------+------------+
| event_id | event_name     | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type |
+----------+----------------+------------+------------+----------+-------------+----------------+--------------+------------+
|        3 | World Cup Final | 2024-07-15 | 16:00:00   |        2 |       50000 |           50000 |       250.00 | Sports     |
+----------+----------------+------------+------------+----------+-------------+----------------+--------------+------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE ticket_price BETWEEN 1000 AND 2500;
Empty set (0.01 sec)
```

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

```
mysql> SELECT *
   -> FROM Event
   -> WHERE event_date BETWEEN '2024-01-15' AND '2024-02-15';
+----------+------------------+------------+------------+----------+-------------+-----------------+-------------+------------+
| event_id | event_name       | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type |
+----------+------------------+------------+------------+----------+-------------+-----------------+-------------+------------+
|        1 | The Batman       | 2024-01-20 | 20:00:00   |        1 |         200 |             200 |        15.00 | Movie      |
|        2 | Avengers: Endgame | 2024-01-21 | 19:00:00   |        1 |         150 |             150 |        12.50 | Movie      |
|        5 | Hamilton         | 2024-02-14 | 19:30:00   |        4 |         800 |             800 |        75.00 | Movie      |
+----------+------------------+------------+------------+----------+-------------+-----------------+-------------+------------+
3 rows in set (0.00 sec)
```

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

```
mysql> SELECT *
   -> FROM Event
   -> WHERE available_seats > 0 AND event_name LIKE '%Concert%';
+----------+----------------------+------------+------------+----------+-------------+-----------------+-------------+------------+
| event_id | event_name           | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type |
+----------+----------------------+------------+------------+----------+-------------+-----------------+-------------+------------+
|        4 | Justin Bieber Concert | 2024-03-05 | 21:00:00   |        3 |        1000 |            1000 |       100.00 | Concert    |
+----------+----------------------+------------+------------+----------+-------------+-----------------+-------------+------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Customer
    -> ORDER BY customer_id
    -> LIMIT 5 OFFSET 5;
+-------------+-----------------+----------------------+--------------+
| customer_id | customer_name   | email                | phone_number |
+-------------+-----------------+----------------------+--------------+
|           6 | Frank Williams  | frank@example.com    | 4445556667   |
|           7 | Grace Taylor    | grace@example.com    | 3334445556   |
|           8 | Henry Jackson   | henry@example.com    | 2223334445   |
|           9 | Isabella Cooper | isabella@example.com | 1112223334   |
|          10 | Jack Moore      | jack@example.com     | 0001112223   |
+-------------+-----------------+----------------------+--------------+
5 rows in set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Booking
    -> WHERE num_tickets > 4;
Empty set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Customer
    -> WHERE phone_number LIKE '%000';
Empty set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE available_seats + total_seats > 15000
    -> ORDER BY total_seats DESC;
+----------+---------------+------------+------------+----------+-------------+-----------------+-------------+------------+
| event_id | event_name    | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type |
+----------+---------------+------------+------------+----------+-------------+-----------------+-------------+------------+
|        3 | World Cup Final | 2024-07-15 | 16:00:00   |        2 |       50000 |           50000 |       250.00 | Sports     |
|       10 | Tennis Open   | 2024-09-01 | 11:00:00   |        7 |       15000 |           15000 |        60.00 | Sports     |
+----------+---------------+------------+------------+----------+-------------+-----------------+-------------+------------+
2 rows in set (0.01 sec)
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_name NOT LIKE 'x%' AND event_name NOT LIKE 'y%' AND event_name NOT LIKE 'z%';
+----------+-----------------------------+------------+------------+----------+-------------+-----------------+-------------+--------
----+
| event_id | event_name                  | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_t
ype |
+----------+-----------------------------+------------+------------+----------+-------------+-----------------+-------------+--------
----+
|        1 | The Batman                  | 2024-01-20 | 20:00:00   |        1 |         200 |             200 |        15.00 | Movie
    |
|        2 | Avengers: Endgame           | 2024-01-21 | 19:00:00   |        1 |         150 |             150 |        12.50 | Movie
    |
|        3 | World Cup Final             | 2024-07-15 | 16:00:00   |        2 |       50000 |           50000 |       250.00 | Sports
    |
|        4 | Justin Bieber Concert       | 2024-03-05 | 21:00:00   |        3 |        1000 |            1000 |       100.00 | Concert
    |
|        5 | Hamilton                    | 2024-02-14 | 19:30:00   |        4 |         800 |             800 |        75.00 | Movie
    |
|        6 | The Nutcracker              | 2024-12-25 | 15:00:00   |        4 |         500 |             500 |        50.00 | Movie
    |
|        7 | Local Symphony Orchestra    | 2024-05-10 | 18:00:00   |        3 |         250 |             250 |        40.00 | Concert
    |
|        8 | International Art Exhibition | 2024-06-01 | 10:00:00   |        5 |        1000 |            1000 |        20.00 | Movie
    |
|        9 | Summer Music Festival       | 2024-08-15 | 17:00:00   |        6 |        5000 |            5000 |        35.00 | Concert
    |
|       10 | Tennis Open                 | 2024-09-01 | 11:00:00   |        7 |       15000 |           15000 |        60.00 | Sports
    |
+----------+-----------------------------+------------+------------+----------+-------------+-----------------+-------------+--------
----+
10 rows in set (0.00 sec)
```

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

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

```
mysql> SELECT event_name, AVG(ticket_price) AS average_ticket_price
    -> FROM Event
    -> GROUP BY event_name;
+------------------------------+----------------------+
| event_name                   | average_ticket_price |
+------------------------------+----------------------+
| The Batman                   |            15.000000 |
| Avengers: Endgame            |            12.500000 |
| World Cup Final              |           250.000000 |
| Justin Bieber Concert        |           100.000000 |
| Hamilton                     |            75.000000 |
| The Nutcracker               |            50.000000 |
| Local Symphony Orchestra     |            40.000000 |
| International Art Exhibition |            20.000000 |
| Summer Music Festival        |            35.000000 |
| Tennis Open                  |            60.000000 |
+------------------------------+----------------------+
10 rows in set (0.01 sec)
```

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

```
mysql> SELECT SUM(num_tickets * ticket_price) AS total_revenue
    -> FROM Booking
    -> JOIN Event ON Booking.event_id = Event.event_id;
+---------------+
| total_revenue |
+---------------+
|       1497.50 |
+---------------+
1 row in set (0.01 sec)
```

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

```
mysql> SELECT event_id, SUM(num_tickets) AS total_tickets_sold
    -> FROM Booking
    -> GROUP BY event_id;
+----------+--------------------+
| event_id | total_tickets_sold |
+----------+--------------------+
|        1 |                  9 |
|        2 |                  9 |
|        3 |                  4 |
|        4 |                  1 |
|        5 |                  2 |
+----------+--------------------+
5 rows in set (0.00 sec)
```

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

```
mysql> SELECT e.event_name, b.total_tickets_sold
    -> FROM (
    ->     SELECT event_id, SUM(num_tickets) AS total_tickets_sold
    ->     FROM Booking
    ->     GROUP BY event_id
    -> ) b
    -> JOIN Event e ON b.event_id = e.event_id;
+----------------------+--------------------+
| event_name           | total_tickets_sold |
+----------------------+--------------------+
| The Batman           |                  9 |
| Avengers: Endgame    |                  9 |
| World Cup Final      |                  4 |
| Justin Bieber Concert |                 1 |
| Hamilton             |                  2 |
+----------------------+--------------------+
5 rows in set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM (
    ->     SELECT e.event_name, b.total_tickets_sold
    ->     FROM (
    ->         SELECT event_id, SUM(num_tickets) AS total_tickets_sold
    ->         FROM Booking
    ->         GROUP BY event_id
    ->     ) b
    ->     JOIN Event e ON b.event_id = e.event_id
    -> ) AS subquery
    -> ORDER BY total_tickets_sold DESC
    -> LIMIT 1;
+------------+--------------------+
| event_name | total_tickets_sold |
+------------+--------------------+
| The Batman |                  9 |
+------------+--------------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT event_id, SUM(num_tickets) AS total_tickets_sold
    -> FROM Booking
    -> GROUP BY event_id;
+----------+--------------------+
| event_id | total_tickets_sold |
+----------+--------------------+
|        1 |                  9 |
|        2 |                  9 |
|        3 |                  4 |
|        4 |                  1 |
|        5 |                  2 |
+----------+--------------------+
5 rows in set (0.00 sec)
```

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

```
mysql> SELECT
    ->     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_name, MONTH(b.booking_date)
    -> ORDER BY booking_month;
+----------------------+---------------+---------------------+
| event_name           | booking_month | total_tickets_sold |
+----------------------+---------------+---------------------+
| The Batman           |             1 |                   9 |
| Avengers: Endgame    |             1 |                   9 |
| World Cup Final      |             1 |                   4 |
| Justin Bieber Concert |             1 |                   1 |
| Hamilton             |             1 |                   2 |
+----------------------+---------------+---------------------+
5 rows in set (0.01 sec)
```

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

```
mysql> SELECT
    ->   e.event_type AS category,
    ->   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;
+----------+--------------------+----------------------+
| category | venue_name         | average_ticket_price |
+----------+--------------------+----------------------+
| Movie    | City Center Cinema |            13.750000 |
| Sports   | National Stadium   |           250.000000 |
| Concert  | Grand Concert Hall |            70.000000 |
| Movie    | Theater Royale     |            62.500000 |
| Movie    | Arts Center        |            20.000000 |
| Concert  | Amphitheater       |            35.000000 |
| Sports   | Civic Arena        |            60.000000 |
+----------+--------------------+----------------------+
7 rows in set (0.00 sec)
```

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

```
mysql> SELECT event_type, SUM(num_tickets) AS total_tickets_sold
    -> FROM Booking
    -> JOIN Event ON Booking.event_id = Event.event_id
    -> GROUP BY event_type;
+------------+---------------------+
| event_type | total_tickets_sold |
+------------+---------------------+
| Movie      |                  20 |
| Sports     |                   4 |
| Concert    |                   1 |
+------------+---------------------+
3 rows in set (0.00 sec)
```

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

```
mysql> SELECT YEAR(event_date) AS year, SUM(num_tickets * ticket_p
rice) AS total_revenue
    -> FROM Booking
    -> JOIN Event ON Booking.event_id = Event.event_id
    -> GROUP BY YEAR(event_date);
+------+---------------+
| year | total_revenue |
+------+---------------+
| 2024 |       1497.50 |
+------+---------------+
1 row in set (0.00 sec)
```

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

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

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

```
mysql> SELECT
    ->   c.customer_name,
    ->   SUM(b.num_tickets * e.ticket_price) AS total_revenue
    -> FROM Booking b
    -> JOIN Event e ON b.event_id = e.event_id
    -> JOIN Customer c ON b.customer_id = c.customer_id
    -> GROUP BY c.customer_id, c.customer_name
    -> ORDER BY total_revenue DESC;
+----------------+---------------+
| customer_name  | total_revenue |
+----------------+---------------+
| Frank Williams |        750.00 |
| Charlie Brown  |        250.00 |
| Henry Jackson  |        150.00 |
| Grace Taylor   |        100.00 |
| David Miller   |         60.00 |
| Jack Moore     |         50.00 |
| Isabella Cooper|         45.00 |
| Bob Johnson    |         37.50 |
| Alice Smith    |         30.00 |
| Emily Davis    |         25.00 |
+----------------+---------------+
10 rows in set (0.00 sec)
```

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

```
mysql> SELECT
    ->   e.event_type AS category,
    ->   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;
+----------+--------------------+----------------------+
| category | venue_name         | average_ticket_price |
+----------+--------------------+----------------------+
| Movie    | City Center Cinema |            13.750000 |
| Sports   | National Stadium   |           250.000000 |
| Concert  | Grand Concert Hall |            70.000000 |
| Movie    | Theater Royale     |            62.500000 |
| Movie    | Arts Center        |            20.000000 |
| Concert  | Amphitheater       |            35.000000 |
| Sports   | Civic Arena        |            60.000000 |
+----------+--------------------+----------------------+
7 rows in set (0.00 sec)
```

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

```
mysql> SELECT
    ->   c.customer_name,
    ->   SUM(b.num_tickets) AS total_tickets_purchased
    -> FROM Booking b
    -> JOIN Customer c ON b.customer_id = c.customer_id
    -> WHERE b.booking_date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)
    -> GROUP BY c.customer_id, c.customer_name
    -> ORDER BY total_tickets_purchased DESC;
+-----------------+--------------------------+
| customer_name   | total_tickets_purchased |
+-----------------+--------------------------+
| David Miller    |                        4 |
| Jack Moore      |                        4 |
| Bob Johnson     |                        3 |
| Frank Williams  |                        3 |
| Isabella Cooper |                        3 |
| Alice Smith     |                        2 |
| Emily Davis     |                        2 |
| Henry Jackson   |                        2 |
| Charlie Brown   |                        1 |
| Grace Taylor    |                        1 |
+-----------------+--------------------------+
10 rows in set (0.01 sec)
```

Tasks 4: Subquery and its types

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

```
mysql> SELECT v.venue_name, avg_price.average_ticket_price
    -> FROM Venue v
    -> JOIN (
    ->   SELECT event_id, AVG(ticket_price) AS average_ticket_price
    ->   FROM Event
    ->   GROUP BY event_id
    -> ) AS avg_price ON v.venue_id = avg_price.event_id;
+--------------------+----------------------+
| venue_name         | average_ticket_price |
+--------------------+----------------------+
| City Center Cinema |            15.000000 |
| National Stadium   |            12.500000 |
| Grand Concert Hall |           250.000000 |
| Theater Royale     |           100.000000 |
| Arts Center        |            75.000000 |
| Amphitheater       |            50.000000 |
| Civic Arena        |            40.000000 |
| The Rex            |            20.000000 |
| The Music Box      |            35.000000 |
| The Dome           |            60.000000 |
+--------------------+----------------------+
```

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

```
mysql> SELECT e.event_name, e.total_seats, e.available_seats,
    -> (e.total_seats - e.available_seats) / e.total_seats * 100 AS per
centage_sold
    -> FROM Event e
    -> WHERE EXISTS (
    ->   SELECT *
    ->   FROM Booking b
    ->   WHERE b.event_id = e.event_id
    ->   HAVING COUNT(*) / e.total_seats * 100 > 50
    -> );
```

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

```
mysql> SELECT event_id, SUM(num_tickets) AS total_tickets_sold
    -> FROM Booking
    -> GROUP BY event_id;
+----------+--------------------+
| event_id | total_tickets_sold |
+----------+--------------------+
|        1 |                  9 |
|        2 |                  9 |
|        3 |                  4 |
|        4 |                  1 |
|        5 |                  2 |
+----------+--------------------+
5 rows in set (0.00 sec)
```

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

```
mysql> SELECT customer_name
    -> FROM Customer c
    -> WHERE NOT EXISTS (
    ->   SELECT *
    ->   FROM Booking b
    ->   WHERE b.customer_id = c.customer_id
    -> );
Empty set (0.01 sec)
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_id NOT IN (
    ->     SELECT event_id
    ->     FROM Booking
    -> );
+----------+------------------------------+------------+------------+----------+-------------+-----------------+--------------+--------
---+
| event_id | event_name                   | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_t
ype |
+----------+------------------------------+------------+------------+----------+-------------+-----------------+--------------+--------
---+
|        6 | The Nutcracker               | 2024-12-25 | 15:00:00   |        4 |         500 |             500 |        50.00 | Movie
   |
|        7 | Local Symphony Orchestra     | 2024-05-10 | 18:00:00   |        3 |         250 |             250 |        40.00 | Concert
   |
|        8 | International Art Exhibition | 2024-06-01 | 10:00:00   |        5 |        1000 |            1000 |        20.00 | Movie
   |
|        9 | Summer Music Festival        | 2024-08-15 | 17:00:00   |        6 |        5000 |            5000 |        35.00 | Concert
   |
|       10 | Tennis Open                  | 2024-09-01 | 11:00:00   |        7 |       15000 |           15000 |        60.00 | Sports
   |
+----------+------------------------------+------------+------------+----------+-------------+-----------------+--------------+--------
---+
```

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

```
mysql> SELECT event_type, SUM(total_tickets_sold) AS total_tickets_by_type
    -> FROM (
    ->   SELECT event_type, SUM(num_tickets) AS total_tickets_sold
    ->   FROM Booking
    ->   JOIN Event ON Booking.event_id = Event.event_id
    ->   GROUP BY event_type
    -> ) AS ticket_sales
    -> GROUP BY event_type;
+------------+----------------------+
| event_type | total_tickets_by_type |
+------------+----------------------+
| Movie      |                   20 |
| Sports     |                    4 |
| Concert    |                    1 |
+------------+----------------------+
3 rows in set (0.00 sec)
```

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

```
mysql> SELECT event_id, event_name, ticket_price
    -> FROM Event
    -> WHERE ticket_price > (
    ->   SELECT AVG(ticket_price)
    ->   FROM Event
    -> );
+----------+-----------------------+--------------+
| event_id | event_name            | ticket_price |
+----------+-----------------------+--------------+
|        3 | World Cup Final       |       250.00 |
|        4 | Justin Bieber Concert |       100.00 |
|        5 | Hamilton              |        75.00 |
+----------+-----------------------+--------------+
3 rows in set (0.00 sec)
```

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

```
mysql> SELECT c.customer_id, c.customer_name,
    -> (SELECT SUM(b.num_tickets * e.ticket_price)
    ->  FROM Booking b
    ->  JOIN Event e ON b.event_id = e.event_id
    ->  WHERE b.customer_id = c.customer_id) AS total_revenue
    -> FROM Customer c;
+-------------+-----------------+---------------+
| customer_id | customer_name   | total_revenue |
+-------------+-----------------+---------------+
|           1 | Alice Smith     |         30.00 |
|           2 | Bob Johnson     |         37.50 |
|           3 | Charlie Brown   |        250.00 |
|           4 | David Miller    |         60.00 |
|           5 | Emily Davis     |         25.00 |
|           6 | Frank Williams  |        750.00 |
|           7 | Grace Taylor    |        100.00 |
|           8 | Henry Jackson   |        150.00 |
|           9 | Isabella Cooper |         45.00 |
|          10 | Jack Moore      |         50.00 |
+-------------+-----------------+---------------+
10 rows in set (0.00 sec)
```

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

```
mysql> SELECT c.customer_name
    -> FROM Customer c
    -> WHERE EXISTS (
    ->   SELECT *
    ->   FROM Booking b
    ->   JOIN Event e ON b.event_id = e.event_id
    ->   WHERE b.customer_id = c.customer_id
    ->   AND e.venue_id = (SELECT venue_id FROM Venue WHERE venue_name = 'Target Venue Name')
;
```

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

```
mysql> SELECT category, SUM(total_tickets_sold) AS total_tickets_by_category
    -> FROM (
    ->   SELECT e.event_type AS category, SUM(b.num_tickets) AS total_tickets_sold
    ->   FROM Booking b
    ->   JOIN Event e ON b.event_id = e.event_id
    ->   GROUP BY e.event_type
    -> ) AS ticket_sales
    -> GROUP BY category;
+----------+---------------------------+
| category | total_tickets_by_category |
+----------+---------------------------+
| Movie    |                        20 |
| Sports   |                         4 |
| Concert  |                         1 |
+----------+---------------------------+
```

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

```
mysql> SELECT c.customer_name, month_booked
    -> FROM Customer c
    -> JOIN Booking b ON c.customer_id = b.customer_id
    -> WHERE EXISTS (
    ->   SELECT *
    ->   FROM Booking b2
    ->   JOIN Event e ON b2.event_id = e.event_id
    ->   WHERE b2.customer_id = c.customer_id
    ->   AND MONTH(b2.booking_date) = MONTH(b.booking_date)
    ->   GROUP BY MONTH(b2.booking_date)
    ->   HAVING COUNT(*) >= 1
    -> );
```

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

```
mysql> SELECT v.venue_name, avg_price.average_ticket_price
    -> FROM Venue v
    -> JOIN (
    ->   SELECT event_id, AVG(ticket_price) AS average_ticket_price
    ->   FROM Event
    ->   GROUP BY event_id
    -> ) AS avg_price ON v.venue_id = avg_price.event_id;
+--------------------+-----------------------+
| venue_name         | average_ticket_price |
+--------------------+-----------------------+
| City Center Cinema |             15.000000 |
| National Stadium   |             12.500000 |
| Grand Concert Hall |            250.000000 |
| Theater Royale     |            100.000000 |
| Arts Center        |             75.000000 |
| Amphitheater       |             50.000000 |
| Civic Arena        |             40.000000 |
| The Rex            |             20.000000 |
| The Music Box      |             35.000000 |
| The Dome           |             60.000000 |
+--------------------+-----------------------+
10 rows in set (0.00 sec)
```