Ticket Booking System

Tasks 1: Database Design:

1. Create the database named "TicketBookingSystem"
   CREATE DATABASE TicketBookingSystem;

```
mysql> CREATE DATABASE TicketBookingSystem;
Query OK, 1 row affected (0.036 sec)

mysql>
```

   USE TicketBookingSystem;

```
mysql> USE TicketBookingSystem;
Database changed
mysql>
```

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

   - **Venue:**
     CREATE TABLE Venue (
     venue_id INT AUTO_INCREMENT PRIMARY KEY,
     venue_name VARCHAR(100) NOT NULL,
     address TEXT
     );

```
mysql> CREATE TABLE Venue ( venue_id INT AUTO_INCREMENT PRIMARY KEY, venue_name VARCHAR(100) NOT NULL, address TEXT);
Query OK, 0 rows affected (0.137 sec)

mysql>
```

   - **Event**
     CREATE TABLE Event (
     event_id INT AUTO_INCREMENT PRIMARY KEY,
     event_name VARCHAR(100) NOT NULL,
     event_date DATE,
     event_time TIME,
     venue_id INT,
     total_seats INT,
     available_seats INT,
     ticket_price DECIMAL(10,2),
     event_type ENUM('Movie', 'Sports', 'Concert'),

FOREIGN KEY (venue_id) REFERENCES Venue(venue_id),
);

```
mysql> CREATE TABLE Event ( event_id INT AUTO_INCREMENT PRIMARY KEY, event_name VARCHAR(100) NOT NULL, event_date DATE,
event_time TIME, venue_id INT, total_seats INT, available_seats INT, ticket_price DECIMAL(10,2), event_type ENUM('Movie'
, 'Sports', 'Concert'), FOREIGN KEY (venue_id) REFERENCES Venue(venue_id));
Query OK, 0 rows affected (0.267 sec)

mysql>
```

ALTER TABLE Event ADD COLUMN booking_id INT;

```
mysql> ALTER TABLE Event ADD COLUMN booking_id INT;
Query OK, 0 rows affected (0.600 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

ALTER TABLE Event
ADD CONSTRAINT fk_event_booking
FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);

```
mysql> ALTER TABLE Event ADD CONSTRAINT fk_event_booking FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
Query OK, 0 rows affected (0.465 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

- **Customers**

CREATE TABLE Customer (
customer_id INT AUTO_INCREMENT PRIMARY KEY,
customer_name VARCHAR(100) NOT NULL,
email VARCHAR(100) UNIQUE,
phone_number VARCHAR(15),
);

```
mysql> CREATE TABLE Customer ( customer_id INT AUTO_INCREMENT PRIMARY KEY, customer_name VARCHAR(100) NOT NULL, email VA
RCHAR(100) UNIQUE, phone_number VARCHAR(15));
Query OK, 0 rows affected (0.254 sec)
```

ALTER TABLE Customer ADD COLUMN booking_id INT;

```
mysql> ALTER TABLE Customer ADD COLUMN booking_id INT;
Query OK, 0 rows affected (1.038 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

ALTER TABLE Customer
ADD CONSTRAINT fk_customer_booking
FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);

```
mysql> ALTER TABLE Customer ADD CONSTRAINT fk_customer_booking FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
Query OK, 0 rows affected (0.702 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

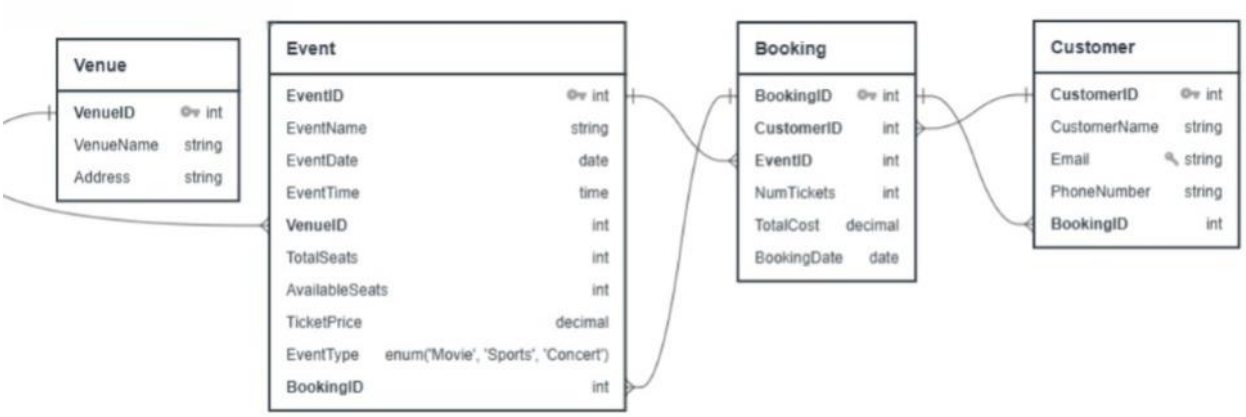
- **Booking**

CREATE TABLE Booking (
booking_id INT AUTO_INCREMENT PRIMARY KEY,

```
    customer_id INT,
    event_id INT,
    num_tickets INT NOT NULL,
    total_cost DECIMAL(10, 2),
    booking_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (event_id) REFERENCES Event(event_id)
);
```

```
mysql> CREATE TABLE Booking ( booking_id INT AUTO_INCREMENT PRIMARY KEY, customer_id INT, event_id INT, num_tickets INT
NOT NULL, total_cost DECIMAL(10, 2), booking_date DATE, FOREIGN KEY (customer_id) REFERENCES Customer(customer_id), FORE
IGN KEY (event_id) REFERENCES Event(event_id));
Query OK, 0 rows affected (0.251 sec)
```

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



4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.
   - Primary Keys**:** venue_id, event_id, customer_id, booking_id
   - Foreign Keys**:**
     - Event.venue_id → Venue.venue_id
     - Event.booking_id → Booking.booking_id
     - Booking.customer_id → Customer.customer_id
     - Booking.event_id → Event.event_id
     - Customer.booking_id → Booking.booking_id

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

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

   Venue:
   INSERT INTO Venue (venue_name, address) VALUES
   ('Arena Max', 'New York'),
   ('Grand Hall', 'Los Angeles'),

('Sky Dome', 'Chicago'),
('Open Grounds', 'Houston'),
('City Auditorium', 'Phoenix'),
('Stadium One', 'San Diego'),
('Night Arena', 'Dallas'),
('Riverfront Grounds', 'Austin'),
('Event Plaza', 'San Jose'),
('Rock Dome', 'Philadelphia');

```
mysql> INSERT INTO Venue (venue_name, address) VALUES ('Arena Max', 'New York'), ('Grand Hall', 'Los Angeles'), ('Sky Do
me', 'Chicago'), ('Open Grounds', 'Houston'), ('City Auditorium', 'Phoenix'), ('Stadium One', 'San Diego'), ('Night Aren
a', 'Dallas'), ('Riverfront Grounds', 'Austin'), ('Event Plaza', 'San Jose'), ('Rock Dome', 'Philadelphia');
Query OK, 10 rows affected (0.054 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
```

Event:

INSERT INTO Event (event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type) VALUES
('Rock Concert', '2025-07-01', '18:00:00', 1, 20000, 5000, 1500.00, 'Concert'),
('World Cup Final', '2025-08-10', '16:00:00', 2, 50000, 10000, 3000.00, 'Sports'),
('Jazz Night', '2025-07-15', '20:00:00', 3, 10000, 3000, 1200.00, 'Concert'),
('Movie Premiere', '2025-07-20', '19:00:00', 4, 8000, 1500, 1000.00, 'Movie'),
('Stand-up Comedy', '2025-06-30', '21:00:00', 5, 3000, 800, 800.00, 'Concert'),
('Basketball Cup', '2025-08-05', '17:00:00', 6, 25000, 10000, 2500.00, 'Sports'),
('Indie Film Fest', '2025-07-10', '17:30:00', 7, 2000, 300, 900.00, 'Movie'),
('Football Night', '2025-07-08', '20:30:00', 8, 30000, 15000, 1800.00, 'Sports'),
('Mega Rock Show', '2025-09-01', '22:00:00', 9, 15000, 7000, 2200.00, 'Concert'),
('HipHop Beats', '2025-09-05', '21:00:00', 10, 10000, 1000, 2000.00, 'Concert');

```
mysql> INSERT INTO Event (event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, even
t_type) VALUES ('Rock Concert', '2025-07-01', '18:00:00', 1, 20000, 5000, 1500.00, 'Concert'), ('World Cup Final', '2025
-08-10', '16:00:00', 2, 50000, 10000, 3000.00, 'Sports'), ('Jazz Night', '2025-07-15', '20:00:00', 3, 10000, 3000, 1200.
00, 'Concert'), ('Movie Premiere', '2025-07-20', '19:00:00', 4, 8000, 1500, 1000.00, 'Movie'), ('Stand-up Comedy', '2025
-06-30', '21:00:00', 5, 3000, 800, 800.00, 'Concert'), ('Basketball Cup', '2025-08-05', '17:00:00', 6, 25000, 10000, 250
0.00, 'Sports'), ('Indie Film Fest', '2025-07-10', '17:30:00', 7, 2000, 300, 900.00, 'Movie'), ('Football Night', '2025-
07-08', '20:30:00', 8, 30000, 15000, 1800.00, 'Sports'), ('Mega Rock Show', '2025-09-01', '22:00:00', 9, 15000, 7000, 22
00.00, 'Concert'), ('HipHop Beats', '2025-09-05', '21:00:00', 10, 10000, 1000, 2000.00, 'Concert');
Query OK, 10 rows affected (0.030 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
```

Customer:

INSERT INTO Customer (customer_name, email, phone_number) VALUES
('Alice', 'alice@gmail.com', '9998800001'),
('Bob', 'bob@gmail.com', '9876543000'),
('Charlie', 'charlie@gmail.com', '9988776655'),
('David', 'david@gmail.com', '9999911112'),
('Emma', 'emma@gmail.com', '9123456000'),
('Fiona', 'fiona@gmail.com', '9012345678'),
('George', 'george@gmail.com', '8888888888'),
('Helen', 'helen@gmail.com', '9090909090'),

('Ivan', 'ivan@gmail.com', '7000000000'),
('Judy', 'judy@gmail.com', '8223456789');

```
mysql> INSERT INTO Customer (customer_name, email, phone_number) VALUES ('Alice', 'alice@gmail.com', '9998800001'), ('Bo
b', 'bob@gmail.com', '9876543000'), ('Charlie', 'charlie@gmail.com', '9988776655'), ('David', 'david@gmail.com', '999991
1112'), ('Emma', 'emma@gmail.com', '9123456000'), ('Fiona', 'fiona@gmail.com', '9012345678'), ('George', 'george@gmail.c
om', '8888888888'), ('Helen', 'helen@gmail.com', '9090909090'), ('Ivan', 'ivan@gmail.com', '7000000000'), ('Judy', 'judy
@gmail.com', '8223456789');
Query OK, 10 rows affected (0.034 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
```

Booking:

INSERT INTO Booking (customer_id, event_id, num_tickets, total_cost, booking_date)
VALUES
(11, 1, 3, 4500.00, '2025-06-01'),
(12, 2, 5, 15000.00, '2025-06-02'),
(13, 3, 2, 2400.00, '2025-06-03'),
(14, 4, 1, 1000.00, '2025-06-04'),
(15, 5, 4, 3200.00, '2025-06-05'),
(16, 6, 6, 15000.00, '2025-06-06'),
(17, 7, 2, 1800.00, '2025-06-07'),
(18, 8, 8, 14400.00, '2025-06-08'),
(19, 9, 1, 2200.00, '2025-06-09'),
(20, 10, 3, 6000.00, '2025-06-10');

```
mysql> INSERT INTO Booking (customer_id, event_id, num_tickets, total_cost, booking_date) VALUES (11, 1, 3, 4500.00, '20
25-06-01'), (12, 2, 5, 15000.00, '2025-06-02'), (13, 3, 2, 2400.00, '2025-06-03'), (14, 4, 1, 1000.00, '2025-06-04'), (1
5, 5, 4, 3200.00, '2025-06-05'), (16, 6, 6, 15000.00, '2025-06-06'), (17, 7, 2, 1800.00, '2025-06-07'), (18, 8, 8, 14400
.00, '2025-06-08'), (19, 9, 1, 2200.00, '2025-06-09'), (20, 10, 3, 6000.00, '2025-06-10');
Query OK, 10 rows affected (0.028 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
```

2. Write a SQL query to list all Events.
   SELECT * FROM Event;

```
mysql> SELECT * FROM Event;
+----------+------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name       | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Rock Concert     | 2025-07-01 | 18:00:00   |        1 |       20000 |            5000 |      1500.00 | Concert    |       NULL |
|        2 | World Cup Final  | 2025-08-10 | 16:00:00   |        2 |       50000 |           10000 |      3000.00 | Sports     |       NULL |
|        3 | Jazz Night       | 2025-07-15 | 20:00:00   |        3 |       10000 |            3000 |      1200.00 | Concert    |       NULL |
|        4 | Movie Premiere   | 2025-07-20 | 19:00:00   |        4 |        8000 |            1500 |      1000.00 | Movie      |       NULL |
|        5 | Stand-up Comedy  | 2025-06-30 | 21:00:00   |        5 |        3000 |             800 |       800.00 | Concert    |       NULL |
|        6 | Basketball Cup   | 2025-08-05 | 17:00:00   |        6 |       25000 |           10000 |      2500.00 | Sports     |       NULL |
|        7 | Indie Film Fest  | 2025-07-10 | 17:30:00   |        7 |        2000 |             300 |       900.00 | Movie      |       NULL |
|        8 | Football Night   | 2025-07-08 | 20:30:00   |        8 |       30000 |           15000 |      1800.00 | Sports     |       NULL |
|        9 | Mega Rock Show   | 2025-09-01 | 22:00:00   |        9 |       15000 |            7000 |      2200.00 | Concert    |       NULL |
|       10 | HipHop Beats     | 2025-09-05 | 21:00:00   |       10 |       10000 |            1000 |      2000.00 | Concert    |       NULL |
|       11 | Free Yoga Workshop | 2025-07-20 | 08:00:00   |        1 |         100 |             100 |         0.00 | Concert    |       NULL |
|       12 | Art & Craft Expo | 2025-08-01 | 11:00:00   |        1 |         300 |             300 |       100.00 | Concert    |       NULL |
+----------+------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
12 rows in set (0.010 sec)

mysql>
```

3. Write a SQL query to select events with available tickets.
   SELECT * FROM Event WHERE available_seats > 0;

```
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_type | booking_id |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Rock Concert      | 2025-07-01 | 18:00:00   |        1 |       20000 |            5000 |      1500.00 | Concert    |       NULL |
|        2 | World Cup Final   | 2025-08-10 | 16:00:00   |        2 |       50000 |           10000 |      3000.00 | Sports     |       NULL |
|        3 | Jazz Night        | 2025-07-15 | 20:00:00   |        3 |       10000 |            3000 |      1200.00 | Concert    |       NULL |
|        4 | Movie Premiere    | 2025-07-20 | 19:00:00   |        4 |        8000 |            1500 |      1000.00 | Movie      |       NULL |
|        5 | Stand-up Comedy   | 2025-06-30 | 21:00:00   |        5 |        3000 |             800 |       800.00 | Concert    |       NULL |
|        6 | Basketball Cup    | 2025-08-05 | 17:00:00   |        6 |       25000 |           10000 |      2500.00 | Sports     |       NULL |
|        7 | Indie Film Fest   | 2025-07-10 | 17:30:00   |        7 |        2000 |             300 |       900.00 | Movie      |       NULL |
|        8 | Football Night    | 2025-07-08 | 20:30:00   |        8 |       30000 |           15000 |      1800.00 | Sports     |       NULL |
|        9 | Mega Rock Show    | 2025-09-01 | 22:00:00   |        9 |       15000 |            7000 |      2200.00 | Concert    |       NULL |
|       10 | HipHop Beats      | 2025-09-05 | 21:00:00   |       10 |       10000 |            1000 |      2000.00 | Concert    |       NULL |
|       11 | Free Yoga Workshop | 2025-07-20 | 08:00:00   |        1 |         100 |             100 |         0.00 | Concert    |       NULL |
|       12 | Art & Craft Expo  | 2025-08-01 | 11:00:00   |        1 |         300 |             300 |       100.00 | Concert    |       NULL |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
12 rows in set (0.009 sec)

mysql>
```

4. Write a SQL query to select events name partial match with ‘cup’.
   SELECT * FROM Event WHERE event_name LIKE '%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 | booking_id |
+----------+-----------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        2 | World Cup Final | 2025-08-10 | 16:00:00   |        2 |       50000 |           10000 |      3000.00 | Sports     |       NULL |
|        6 | Basketball Cup  | 2025-08-05 | 17:00:00   |        6 |       25000 |           10000 |      2500.00 | Sports     |       NULL |
+----------+-----------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
2 rows in set (0.017 sec)

mysql>
```

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;

```
mysql> SELECT * FROM Event WHERE ticket_price BETWEEN 1000 AND 2500;
+----------+-----------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name      | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+-----------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Rock Concert    | 2025-07-01 | 18:00:00   |        1 |       20000 |            5000 |      1500.00 | Concert    |       NULL |
|        3 | Jazz Night      | 2025-07-15 | 20:00:00   |        3 |       10000 |            3000 |      1200.00 | Concert    |       NULL |
|        4 | Movie Premiere  | 2025-07-20 | 19:00:00   |        4 |        8000 |            1500 |      1000.00 | Movie      |       NULL |
|        6 | Basketball Cup  | 2025-08-05 | 17:00:00   |        6 |       25000 |           10000 |      2500.00 | Sports     |       NULL |
|        8 | Football Night  | 2025-07-08 | 20:30:00   |        8 |       30000 |           15000 |      1800.00 | Sports     |       NULL |
|        9 | Mega Rock Show  | 2025-09-01 | 22:00:00   |        9 |       15000 |            7000 |      2200.00 | Concert    |       NULL |
|       10 | HipHop Beats    | 2025-09-05 | 21:00:00   |       10 |       10000 |            1000 |      2000.00 | Concert    |       NULL |
+----------+-----------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
7 rows in set (0.011 sec)

mysql>
```

6. Write a SQL query to retrieve events with dates falling within a specific range.
   SELECT * FROM Event WHERE event_date BETWEEN '2025-07-01' AND '2025-08-01';

```
mysql> SELECT * FROM Event WHERE event_date BETWEEN '2025-07-01' AND '2025-08-01';
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name        | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Rock Concert      | 2025-07-01 | 18:00:00   |        1 |       20000 |            5000 |      1500.00 | Concert    |       NULL |
|        3 | Jazz Night        | 2025-07-15 | 20:00:00   |        3 |       10000 |            3000 |      1200.00 | Concert    |       NULL |
|        4 | Movie Premiere    | 2025-07-20 | 19:00:00   |        4 |        8000 |            1500 |      1000.00 | Movie      |       NULL |
|        7 | Indie Film Fest   | 2025-07-10 | 17:30:00   |        7 |        2000 |             300 |       900.00 | Movie      |       NULL |
|        8 | Football Night    | 2025-07-08 | 20:30:00   |        8 |       30000 |           15000 |      1800.00 | Sports     |       NULL |
|       11 | Free Yoga Workshop | 2025-07-20 | 08:00:00   |        1 |         100 |             100 |         0.00 | Concert    |       NULL |
|       12 | Art & Craft Expo  | 2025-08-01 | 11:00:00   |        1 |         300 |             300 |       100.00 | Concert    |       NULL |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
7 rows in set (0.016 sec)

mysql>
```

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_type = 'Concert'

AND event_name LIKE '%Concert%';

```
mysql> SELECT * FROM Event WHERE available_seats > 0 AND event_type = 'Concert' AND event_name LIKE '%Concert%';
+----------+--------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
| event_id | event_name   | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+--------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
|        1 | Rock Concert | 2025-07-01 | 18:00:00   |        1 |       20000 |           5000 |      1500.00 | Concert    |       NULL |
+----------+--------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
1 row in set (0.008 sec)

mysql>
```

8. Write a SQL query to retrieve users in batches of 5, starting from the 6th user.
   SELECT * FROM Customer LIMIT 5 OFFSET 5;

```
mysql> SELECT * FROM Customer LIMIT 5 OFFSET 5;
+-------------+---------------+------------------+--------------+-------------+
| customer_id | customer_name | email            | phone_number | booking_id |
+-------------+---------------+------------------+--------------+-------------+
|          16 | Fiona         | fiona@gmail.com  | 9012345678   |        NULL |
|          17 | George        | george@gmail.com | 8888888888   |        NULL |
|          18 | Helen         | helen@gmail.com  | 9090909090   |        NULL |
|          19 | Ivan          | ivan@gmail.com   | 7000000000   |        NULL |
|          20 | Judy          | judy@gmail.com   | 8223456789   |        NULL |
+-------------+---------------+------------------+--------------+-------------+
5 rows in set (0.006 sec)

mysql>
```

9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.
   SELECT * FROM Booking WHERE num_tickets > 4;

```
mysql> SELECT * FROM Booking WHERE num_tickets > 4;
+------------+-------------+----------+-------------+------------+--------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
+------------+-------------+----------+-------------+------------+--------------+
|         22 |          12 |        2 |           5 |   15000.00 | 2025-06-02   |
|         26 |          16 |        6 |           6 |   15000.00 | 2025-06-06   |
|         28 |          18 |        8 |           8 |   14400.00 | 2025-06-08   |
+------------+-------------+----------+-------------+------------+--------------+
3 rows in set (0.009 sec)

mysql>
```

10. Write a SQL query to retrieve customer information whose phone number end with ‘000’
    SELECT * FROM Customer WHERE phone_number LIKE '%000';

```
mysql> SELECT * FROM Customer WHERE phone_number LIKE '%000';
+-------------+---------------+----------------+--------------+-------------+
| customer_id | customer_name | email          | phone_number | booking_id |
+-------------+---------------+----------------+--------------+-------------+
|          12 | Bob           | bob@gmail.com  | 9876543000   |        NULL |
|          15 | Emma          | emma@gmail.com | 9123456000   |        NULL |
|          19 | Ivan          | ivan@gmail.com | 7000000000   |        NULL |
+-------------+---------------+----------------+--------------+-------------+
3 rows in set (0.008 sec)

mysql>
```

11. Write a SQL query to retrieve the events in order whose seat capacity more than 15000.
    SELECT * FROM Event WHERE total_seats > 15000 ORDER BY total_seats DESC;

```
mysql> SELECT * FROM Event WHERE 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 | booking_id |
+----------+-----------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        2 | World Cup Final | 2025-08-10 | 16:00:00   |        2 |       50000 |           10000 |      3000.00 | Sports     |       NULL |
|        8 | Football Night  | 2025-07-08 | 20:30:00   |        8 |       30000 |           15000 |      1800.00 | Sports     |       NULL |
|        6 | Basketball Cup  | 2025-08-05 | 17:00:00   |        6 |       25000 |           10000 |      2500.00 | Sports     |       NULL |
|        1 | Rock Concert    | 2025-07-01 | 18:00:00   |        1 |       20000 |            5000 |      1500.00 | Concert    |       NULL |
+----------+-----------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
4 rows in set (0.012 sec)

mysql>
```

12. Write a SQL query to select events name not start with ‘x’, ‘y’, ‘z’
    SELECT * FROM Event WHERE event_name NOT REGEXP '^[xyzXYZ]';

```
mysql> SELECT * FROM Event WHERE event_name NOT REGEXP '^[xyzXYZ]';
+----------+-----------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name      | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+-----------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Rock Concert    | 2025-07-01 | 18:00:00   |        1 |       20000 |            5000 |      1500.00 | Concert    |       NULL |
|        2 | World Cup Final | 2025-08-10 | 16:00:00   |        2 |       50000 |           10000 |      3000.00 | Sports     |       NULL |
|        3 | Jazz Night      | 2025-07-15 | 20:00:00   |        3 |       10000 |            3000 |      1200.00 | Concert    |       NULL |
|        4 | Movie Premiere  | 2025-07-20 | 19:00:00   |        4 |        8000 |            1500 |      1000.00 | Movie      |       NULL |
|        5 | Stand-up Comedy | 2025-06-30 | 21:00:00   |        5 |        3000 |             800 |       800.00 | Concert    |       NULL |
|        6 | Basketball Cup  | 2025-08-05 | 17:00:00   |        6 |       25000 |           10000 |      2500.00 | Sports     |       NULL |
|        7 | Indie Film Fest | 2025-07-10 | 17:30:00   |        7 |        2000 |             300 |       900.00 | Movie      |       NULL |
|        8 | Football Night  | 2025-07-08 | 20:30:00   |        8 |       30000 |           15000 |      1800.00 | Sports     |       NULL |
|        9 | Mega Rock Show  | 2025-09-01 | 22:00:00   |        9 |       15000 |            7000 |      2200.00 | Concert    |       NULL |
|       10 | HipHop Beats    | 2025-09-05 | 21:00:00   |       10 |       10000 |            1000 |      2000.00 | Concert    |       NULL |
+----------+-----------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
10 rows in set (0.070 sec)

mysql>
```

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

1. Write a SQL query to List Events and Their Average Ticket Prices.
   SELECT event_name, AVG(ticket_price) AS avg_price
   FROM Event
   GROUP BY event_name;

```
mysql> SELECT event_name, AVG(ticket_price) AS avg_price FROM Event GROUP BY event_name;
+-----------------+-------------+
| event_name      | avg_price   |
+-----------------+-------------+
| Rock Concert    | 1500.000000 |
| World Cup Final | 3000.000000 |
| Jazz Night      | 1200.000000 |
| Movie Premiere  | 1000.000000 |
| Stand-up Comedy |  800.000000 |
| Basketball Cup  | 2500.000000 |
| Indie Film Fest |  900.000000 |
| Football Night  | 1800.000000 |
| Mega Rock Show  | 2200.000000 |
| HipHop Beats    | 2000.000000 |
+-----------------+-------------+
10 rows in set (0.029 sec)

mysql>
```

2. Write a SQL query to Calculate the Total Revenue Generated by Events.
   SELECT e.event_name, SUM(b.total_cost) AS total_revenue
   FROM Booking b
   JOIN Event e ON b.event_id = e.event_id

GROUP BY e.event_name;

```
mysql> SELECT e.event_name, SUM(b.total_cost) AS total_revenue FROM Booking b JOIN Event e ON b.event_id = e.event_id GROUP BY e.event_name;
+-----------------+---------------+
| event_name      | total_revenue |
+-----------------+---------------+
| Rock Concert    |       4500.00 |
| World Cup Final |      15000.00 |
| Jazz Night      |       2400.00 |
| Movie Premiere  |       1000.00 |
| Stand-up Comedy |       3200.00 |
| Basketball Cup  |      15000.00 |
| Indie Film Fest |       1800.00 |
| Football Night  |      14400.00 |
| Mega Rock Show  |       2200.00 |
| HipHop Beats    |       6000.00 |
+-----------------+---------------+
10 rows in set (0.009 sec)

mysql>
```

3. Write a SQL query to find the event with the highest ticket sales.
   SELECT e.event_name, SUM(b.num_tickets) AS total_tickets
   FROM Booking b
   JOIN Event e ON b.event_id = e.event_id
   GROUP BY e.event_name
   ORDER BY total_tickets DESC
   LIMIT 1;

```
mysql> SELECT e.event_name, SUM(b.num_tickets) AS total_tickets FROM Booking b JOIN Event e ON b.event_id = e.event_id GROUP BY e.event_name ORDER BY total_
tickets DESC LIMIT 1;
+---------------+---------------+
| event_name    | total_tickets |
+---------------+---------------+
| Footbal Night |             8 |
+---------------+---------------+
1 row in set (0.008 sec)

mysql>
```

4. Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.
   SELECT e.event_name, SUM(b.num_tickets) AS tickets_sold
   FROM Booking b
   JOIN Event e ON b.event_id = e.event_id
   GROUP BY e.event_name;

```
mysql> SELECT e.event_name, SUM(b.num_tickets) AS tickets_sold FROM Booking b JOIN Event e ON b.event_id = e.event_id GROUP BY e.event_name;
+-----------------+--------------+
| event_name      | tickets_sold |
+-----------------+--------------+
| Rock Concert    |            3 |
| World Cup Final |            5 |
| Jazz Night      |            2 |
| Movie Premiere  |            1 |
| Stand-up Comedy |            4 |
| Basketball Cup  |            6 |
| Indie Film Fest |            2 |
| Football Night  |            8 |
| Mega Rock Show  |            1 |
| HipHop Beats    |            3 |
+-----------------+--------------+
10 rows in set (0.006 sec)

mysql>
```

5. Write a SQL query to Find Events with No Ticket Sales.
   SELECT e.event_name
   FROM Event e
   LEFT JOIN Booking b ON e.event_id = b.event_id
   WHERE b.booking_id IS NULL;

```
mysql> SELECT e.event_name FROM Event e LEFT JOIN Booking b ON e.event_id = b.event_id WHERE b.booking_id IS NULL;
+---------------------+
| event_name          |
+---------------------+
| Free Yoga Workshop |
+---------------------+
1 row in set (0.163 sec)

mysql>
```

6. Write a SQL query to Find the User Who Has Booked the Most Tickets.
   SELECT c.customer_name, SUM(b.num_tickets) AS total_tickets
   FROM Booking b
   JOIN Customer c ON b.customer_id = c.customer_id
   GROUP BY c.customer_name
   ORDER BY total_tickets DESC
   LIMIT 1;

```
mysql> SELECT c.customer_name, SUM(b.num_tickets) AS total_tickets FROM Booking b JOIN Customer c ON b.customer_id = c.customer_id GROUP BY c.customer_name
ORDER BY total_tickets DESC LIMIT 1;
+---------------+---------------+
| customer_name | total_tickets |
+---------------+---------------+
| Helen         |             8 |
+---------------+---------------+
1 row in set (0.007 sec)

mysql>
```

7. Write a SQL query to List Events and the total number of tickets sold for each month.
   SELECT e.event_name, MONTH(b.booking_date) AS booking_month,
   SUM(b.num_tickets) AS tickets_sold
   FROM Booking b
   JOIN Event e ON b.event_id = e.event_id
   GROUP BY e.event_name, booking_month
   ORDER BY booking_month;

```
mysql> SELECT e.event_name, MONTH(b.booking_date) AS booking_month, SUM(b.num_tickets) AS tickets_sold FROM Booking b JOIN Event e ON b.event_id = e.event_i
d GROUP BY e.event_name, booking_month ORDER BY booking_month;
+-----------------+---------------+--------------+
| event_name      | booking_month | tickets_sold |
+-----------------+---------------+--------------+
| Rock Concert    |             6 |            3 |
| World Cup Final |             6 |            5 |
| Jazz Night      |             6 |            2 |
| Movie Premiere  |             6 |            1 |
| Stand-up Comedy |             6 |            4 |
| Basketball Cup  |             6 |            6 |
| Indie Film Fest |             6 |            2 |
| Football Night  |             6 |            8 |
| Mega Rock Show  |             6 |            1 |
| HipHop Beats    |             6 |            3 |
+-----------------+---------------+--------------+
10 rows in set (0.027 sec)

mysql>
```

8. Write a SQL query to calculate the average Ticket Price for Events in Each Venue.
   SELECT v.venue_name, AVG(e.ticket_price) AS avg_ticket_price
   FROM Event e
   JOIN Venue v ON e.venue_id = v.venue_id
   GROUP BY v.venue_name;

```
mysql> SELECT v.venue_name, AVG(e.ticket_price) AS avg_ticket_price FROM Event e JOIN Venue v ON e.venue_id = v.venue_id GROUP BY v.venue_name;
+--------------------+------------------+
| venue_name         | avg_ticket_price |
+--------------------+------------------+
| Arena Max          |      1500.000000 |
| Grand Hall         |      3000.000000 |
| Sky Dome           |      1200.000000 |
| Open Grounds       |      1000.000000 |
| City Auditorium    |       800.000000 |
| Stadium One        |      2500.000000 |
| Night Arena        |       900.000000 |
| Riverfront Grounds |      1800.000000 |
| Event Plaza        |      2200.000000 |
| Rock Dome          |      2000.000000 |
+--------------------+------------------+
10 rows in set (0.012 sec)

mysql>
```

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

```
mysql> SELECT e.event_type, SUM(b.num_tickets) AS tickets_sold FROM Booking b JOIN Event e ON b.event_id = e.event_id GROUP BY e.event_type;
+------------+--------------+
| event_type | tickets_sold |
+------------+--------------+
| Concert    |           13 |
| Sports     |           19 |
| Movie      |            3 |
+------------+--------------+
3 rows in set (0.010 sec)

mysql>
```

10. Write a SQL query to calculate the total Revenue Generated by Events in Each Year.
    SELECT YEAR(b.booking_date) AS year, SUM(b.total_cost) AS total_revenue
    FROM Booking b
    GROUP BY year;

```
mysql> SELECT YEAR(b.booking_date) AS year, SUM(b.total_cost) AS total_revenue FROM Booking b GROUP BY year;
+------+----------------+
| year | total_revenue  |
+------+----------------+
| 2025 |       65500.00 |
+------+----------------+
1 row in set (0.006 sec)

mysql>
```

11. Write a SQL query to list users who have booked tickets for multiple events.
    SELECT c.customer_name, COUNT(DISTINCT b.event_id) AS event_count
    FROM Booking b
    JOIN Customer c ON b.customer_id = c.customer_id
    GROUP BY c.customer_name
    HAVING event_count > 1;

```
mysql> SELECT c.customer_name, COUNT(DISTINCT b.event_id) AS event_count FROM Booking b JOIN Customer c ON b.customer_id = c.customer_id GROUP BY c.customer
_name HAVING event_count > 1;
+---------------+-------------+
| customer_name | event_count |
+---------------+-------------+
| Alice         |           2 |
+---------------+-------------+
1 row in set (0.057 sec)

mysql>
```

12. Write a SQL query to calculate the Total Revenue Generated by Events for Each User.
    SELECT c.customer_name, SUM(b.total_cost) AS user_revenue
    FROM Booking b
    JOIN Customer c ON b.customer_id = c.customer_id
    GROUP BY c.customer_name;

```
mysql> SELECT c.customer_name, SUM(b.total_cost) AS user_revenue FROM Booking b JOIN Customer c ON b.customer_id = c.customer_id GROUP BY c.customer_name;
+---------------+--------------+
| customer_name | user_revenue |
+---------------+--------------+
| Alice         |      4500.00 |
| Bob           |     15000.00 |
| Charlie       |      2400.00 |
| David         |      1000.00 |
| Emma          |      3200.00 |
| Fiona         |     15000.00 |
| George        |      1800.00 |
| Helen         |     14400.00 |
| Ivan          |      2200.00 |
| Judy          |      6000.00 |
+---------------+--------------+
10 rows in set (0.011 sec)

mysql>
```

13. Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.
    SELECT e.event_type, v.venue_name, AVG(e.ticket_price) AS avg_price
    FROM Event e
    JOIN Venue v ON e.venue_id = v.venue_id
    GROUP BY e.event_type, v.venue_name;

```
mysql> SELECT e.event_type, v.venue_name, AVG(e.ticket_price) AS avg_price FROM Event e JOIN Venue v ON e.venue_id = v.venue_id GROUP BY e.event_type, v.ven
ue_name;
+------------+--------------------+-------------+
| event_type | venue_name         | avg_price   |
+------------+--------------------+-------------+
| Concert    | Arena Max          | 1500.000000 |
| Sports     | Grand Hall         | 3000.000000 |
| Concert    | Sky Dome           | 1200.000000 |
| Movie      | Open Grounds       | 1000.000000 |
| Concert    | City Auditorium    |  800.000000 |
| Sports     | Stadium One        | 2500.000000 |
| Movie      | Night Arena        |  900.000000 |
| Sports     | Riverfront Grounds | 1800.000000 |
| Concert    | Event Plaza        | 2200.000000 |
| Concert    | Rock Dome          | 2000.000000 |
+------------+--------------------+-------------+
10 rows in set (0.010 sec)

mysql>
```

14. Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days.
    SELECT c.customer_name, SUM(b.num_tickets) AS tickets_last_30_days
    FROM Booking b
    JOIN Customer c ON b.customer_id = c.customer_id
    WHERE b.booking_date >= CURDATE() - INTERVAL 30 DAY
    GROUP BY c.customer_name;

```
mysql> SELECT c.customer_name, SUM(b.num_tickets) AS tickets_last_30_days FROM Booking b JOIN Customer c ON b.customer_id = c.customer_id WHERE b.booking_da
te >= CURDATE() - INTERVAL 30 DAY GROUP BY c.customer_name;
+---------------+----------------------+
| customer_name | tickets_last_30_days |
+---------------+----------------------+
| Alice         |                    3 |
| Bob           |                    5 |
| Charlie       |                    2 |
| David         |                    1 |
| Emma          |                    4 |
| Fiona         |                    6 |
| George        |                    2 |
| Helen         |                    8 |
| Ivan          |                    1 |
| Judy          |                    3 |
+---------------+----------------------+
10 rows in set (0.032 sec)

mysql>
```

Tasks 4: Subquery and its types:

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

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

```
mysql> SELECT venue_name, (SELECT AVG(ticket_price) FROM Event e WHERE e.venue_id = v.venue_id) AS avg_ticket_price FROM Venue v;
+--------------------+-------------------+
| venue_name         | avg_ticket_price |
+--------------------+-------------------+
| Arena Max          |      1500.000000 |
| Grand Hall         |      3000.000000 |
| Sky Dome           |      1200.000000 |
| Open Grounds       |      1000.000000 |
| City Auditorium    |       800.000000 |
| Stadium One        |      2500.000000 |
| Night Arena        |       900.000000 |
| Riverfront Grounds |      1800.000000 |
| Event Plaza        |      2200.000000 |
| Rock Dome          |      2000.000000 |
+--------------------+-------------------+
10 rows in set (0.023 sec)

mysql>
```

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

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

```
mysql> SELECT event_name FROM Event WHERE (total_seats - available_seats) > (total_seats / 2);
+-----------------+
| event_name      |
+-----------------+
| Rock Concert    |
| World Cup Final |
| Jazz Night      |
| Movie Premiere  |
| Stand-up Comedy |
| Basketball Cup  |
| Indie Film Fest |
| Mega Rock Show  |
| HipHop Beats    |
+-----------------+
9 rows in set (0.016 sec)

mysql>
```

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

```
SELECT 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;
```

```
mysql> SELECT 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;
+----------------+--------------------+
| event_name     | total_tickets_sold |
+----------------+--------------------+
| Rock Concert   |                  3 |
| World Cup Final |                 5 |
| Jazz Night     |                  2 |
| Movie Premiere |                  1 |
| Stand-up Comedy |                 4 |
| Basketball Cup |                  6 |
| Indie Film Fest |                 2 |
| Football Night |                  8 |
| Mega Rock Show |                  1 |
| HipHop Beats   |                  3 |
+----------------+--------------------+
10 rows in set (0.008 sec)

mysql>
```

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

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

```
mysql> SELECT c.customer_name FROM Customer c WHERE NOT EXISTS (SELECT 1 FROM Booking b WHERE b.customer_id = c.customer_id );
+---------------+
| customer_name |
+---------------+
| Kevin         |
+---------------+
1 row in set (0.015 sec)

mysql>
```

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

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

```
mysql> SELECT event_name FROM Event WHERE event_id NOT IN (SELECT DISTINCT event_id FROM Booking);
+--------------------+
| event_name         |
+--------------------+
| Free Yoga Workshop |
| Art & Craft Expo   |
+--------------------+
2 rows in set (0.041 sec)

mysql>
```

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

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

```
mysql> SELECT event_type, SUM(total_tickets) AS total_tickets_sold FROM (SELECT e.event_type, b.num_tickets AS total_tickets  FROM Booking b JOIN Event e ON
 b.event_id = e.event_id ) AS sub GROUP BY event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Concert    |                 13 |
| Sports     |                 19 |
| Movie      |                  3 |
+------------+--------------------+
3 rows in set (0.011 sec)

mysql>
```

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

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

```
mysql> SELECT event_name, ticket_price FROM Event WHERE ticket_price > (SELECT AVG(ticket_price) FROM Event);
+-----------------+--------------+
| event_name      | ticket_price |
+-----------------+--------------+
| World Cup Final |      3000.00 |
| Basketball Cup  |      2500.00 |
| Football Night  |      1800.00 |
| Mega Rock Show  |      2200.00 |
| HipHop Beats    |      2000.00 |
+-----------------+--------------+
5 rows in set (0.013 sec)

mysql>
```

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

```
SELECT c.customer_name,
     (SELECT SUM(b.total_cost)
```

```
    FROM Booking b
    WHERE b.customer_id = c.customer_id) AS total_revenue
FROM Customer c;
```

```
mysql> SELECT c.customer_name, (SELECT SUM(b.total_cost) FROM Booking b WHERE b.customer_id = c.customer_id) AS total_revenue FROM Customer c;
+---------------+---------------+
| customer_name | total_revenue |
+---------------+---------------+
| Alice         |       4500.00 |
| Bob           |      15000.00 |
| Charlie       |       2400.00 |
| David         |       1000.00 |
| Emma          |       3200.00 |
| Fiona         |      15000.00 |
| George        |       1800.00 |
| Helen         |      14400.00 |
| Ivan          |       2200.00 |
| Judy          |       6000.00 |
+---------------+---------------+
10 rows in set (0.007 sec)

mysql>
```

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

```
SELECT customer_name
FROM Customer
WHERE customer_id IN (
   SELECT b.customer_id
   FROM Booking b
   JOIN Event e ON b.event_id = e.event_id
   WHERE e.venue_id = 1
);
```

```
mysql> SELECT customer_name FROM Customer WHERE customer_id IN ( SELECT b.customer_id FROM Booking b JOIN Event e ON b.event_id = e.event_id WHERE e.venue_i
d = 1 );
+---------------+
| customer_name |
+---------------+
| Alice         |
+---------------+
1 row in set (0.010 sec)

mysql>
```

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

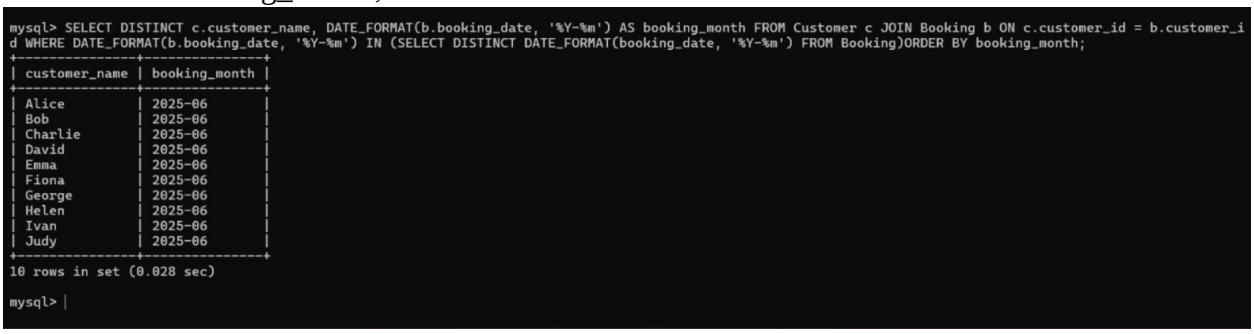
```
SELECT event_type, SUM(total_tickets) AS total_tickets_sold
FROM (
   SELECT e.event_type, b.num_tickets AS total_tickets
   FROM Event e
   JOIN Booking b ON e.event_id = b.event_id
) AS sub
GROUP BY event_type;
```

```
mysql> SELECT event_type, SUM(total_tickets) AS total_tickets_sold FROM ( SELECT e.event_type, b.num_tickets AS total_tickets FROM Event e JOIN Booking b ON
 e.event_id = b.event_id) AS sub GROUP BY event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Concert    |                 13 |
| Sports     |                 19 |
| Movie      |                  3 |
+------------+--------------------+
3 rows in set (0.008 sec)

mysql>
```

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

```
SELECT DISTINCT c.customer_name,
      DATE_FORMAT(b.booking_date, '%Y-%m') AS booking_month
FROM Customer c
JOIN Booking b ON c.customer_id = b.customer_id
WHERE DATE_FORMAT(b.booking_date, '%Y-%m') IN (
   SELECT DISTINCT DATE_FORMAT(booking_date, '%Y-%m')
   FROM Booking
)
ORDER BY booking_month;
```

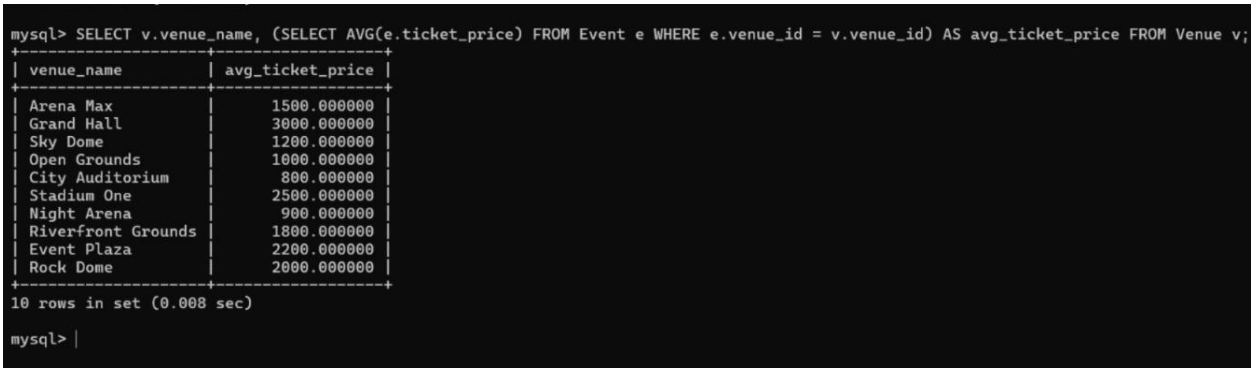


```
mysql> SELECT DISTINCT c.customer_name, DATE_FORMAT(b.booking_date, '%Y-%m') AS booking_month FROM Customer c JOIN Booking b ON c.customer_id = b.customer_i
d WHERE DATE_FORMAT(b.booking_date, '%Y-%m') IN (SELECT DISTINCT DATE_FORMAT(booking_date, '%Y-%m') FROM Booking)ORDER BY booking_month;
+---------------+---------------+
| customer_name | booking_month |
+---------------+---------------+
| Alice         | 2025-06       |
| Bob           | 2025-06       |
| Charlie       | 2025-06       |
| David         | 2025-06       |
| Emma          | 2025-06       |
| Fiona         | 2025-06       |
| George        | 2025-06       |
| Helen         | 2025-06       |
| Ivan          | 2025-06       |
| Judy          | 2025-06       |
+---------------+---------------+
10 rows in set (0.028 sec)

mysql>
```

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

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



```
mysql> SELECT v.venue_name, (SELECT AVG(e.ticket_price) FROM Event e WHERE e.venue_id = v.venue_id) AS avg_ticket_price FROM Venue v;
+--------------------+------------------+
| venue_name         | avg_ticket_price |
+--------------------+------------------+
| Arena Max          |      1500.000000 |
| Grand Hall         |      3000.000000 |
| Sky Dome           |      1200.000000 |
| Open Grounds       |      1000.000000 |
| City Auditorium    |       800.000000 |
| Stadium One        |      2500.000000 |
| Night Arena        |       900.000000 |
| Riverfront Grounds |      1800.000000 |
| Event Plaza        |      2200.000000 |
| Rock Dome          |      2000.000000 |
+--------------------+------------------+
10 rows in set (0.008 sec)

mysql>
```