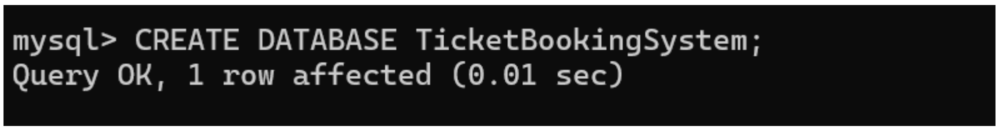
Ticket booking system

Dhamini Machireddy

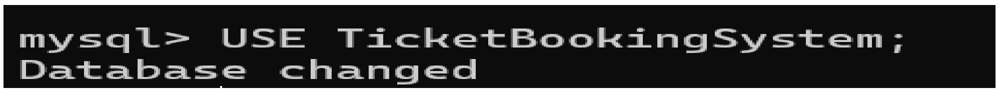
Task 1:

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



```
mysql> CREATE DATABASE TicketBookingSystem;
Query OK, 1 row affected (0.01 sec)
```

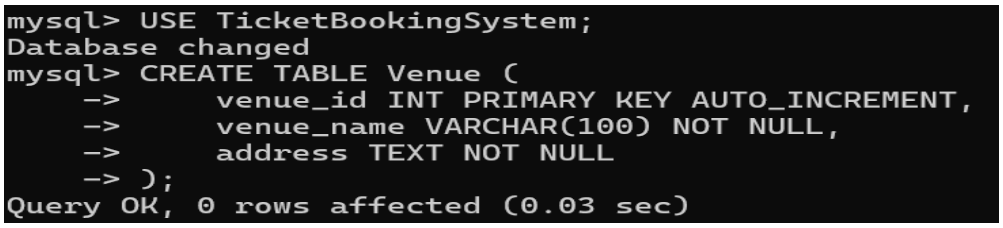
2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.



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

   1. Venu :

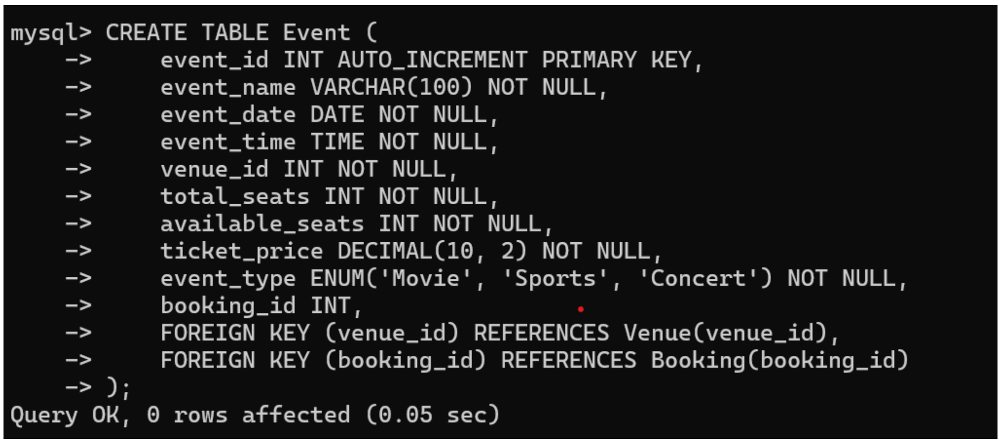
```
CREATE TABLE Venue (
   venue_id INT AUTO_INCREMENT PRIMARY KEY,
   venue_name VARCHAR(100) NOT NULL,
   address VARCHAR(255) NOT NULL
);
```



```
mysql> USE TicketBookingSystem;
Database changed
mysql> CREATE TABLE Venue (
    ->     venue_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     venue_name VARCHAR(100) NOT NULL,
    ->     address TEXT NOT NULL
    -> );
Query OK, 0 rows affected (0.03 sec)
```

   2. Event :

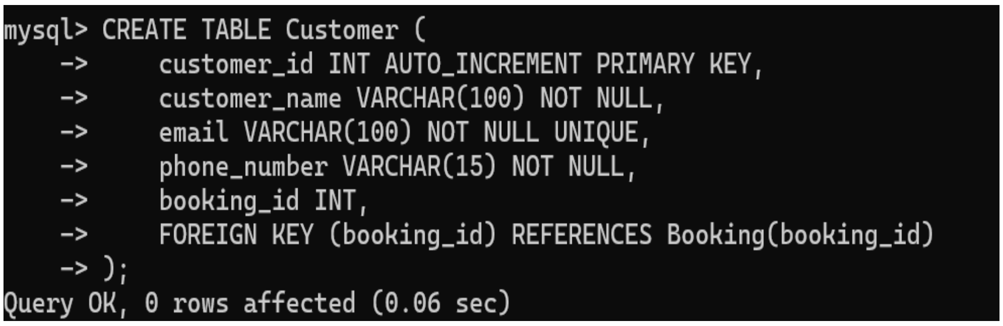
```
CREATE TABLE Event (
   event_id INT AUTO_INCREMENT PRIMARY KEY,
   event_name VARCHAR(100) NOT NULL,
   event_date DATE NOT NULL,
   event_time TIME NOT NULL,
   venue_id INT NOT NULL,
   total_seats INT NOT NULL,
   available_seats INT NOT NULL,
   ticket_price DECIMAL(10, 2) NOT NULL,
   event_type ENUM('Movie', 'Sports', 'Concert') NOT NULL,
   booking_id INT,
   FOREIGN KEY (venue_id) REFERENCES Venue(venue_id),
   FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
);
```



```
mysql> CREATE TABLE Event (
    ->     event_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     event_name VARCHAR(100) NOT NULL,
    ->     event_date DATE NOT NULL,
    ->     event_time TIME NOT NULL,
    ->     venue_id INT NOT NULL,
    ->     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,
    ->     booking_id INT,
    ->     FOREIGN KEY (venue_id) REFERENCES Venue(venue_id),
    ->     FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

3. Customers:

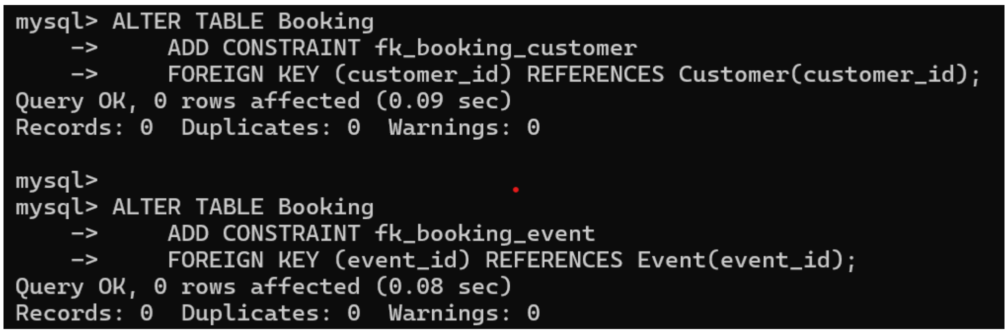
```
CREATE TABLE Customer (
  customer_id INT AUTO_INCREMENT PRIMARY KEY,
  customer_name VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE,
  phone_number VARCHAR(15) NOT NULL,
  booking_id INT,
  FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
);
```



```
mysql> CREATE TABLE Customer (
    ->     customer_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     customer_name VARCHAR(100) NOT NULL,
    ->     email VARCHAR(100) NOT NULL UNIQUE,
    ->     phone_number VARCHAR(15) NOT NULL,
    ->     booking_id INT,
    ->     FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

4. Booking **:**

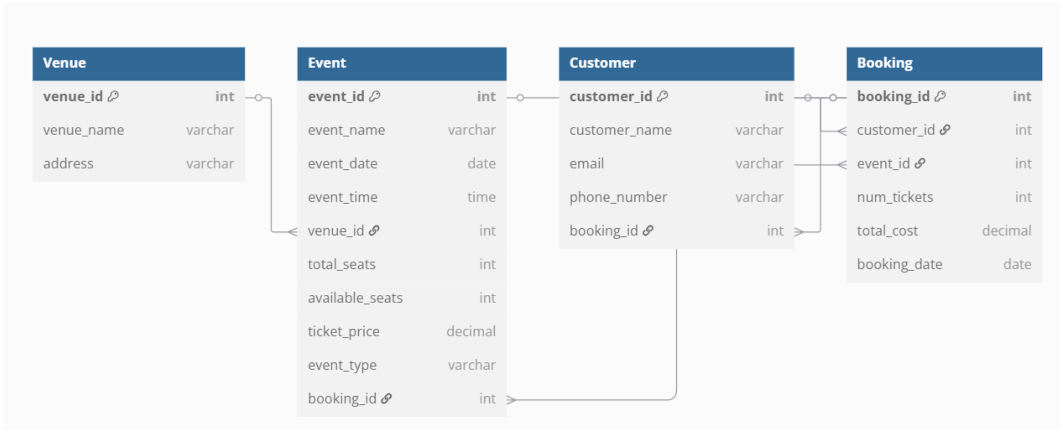
```
CREATE TABLE Booking (
  booking_id INT AUTO_INCREMENT PRIMARY KEY,
  customer_id INT,
  event_id INT,
  num_tickets INT NOT NULL,
  total_cost DECIMAL(10, 2) NOT NULL,
  booking_date DATE NOT NULL
);
```



```
mysql> ALTER TABLE Booking
    ->     ADD CONSTRAINT fk_booking_customer
    ->     FOREIGN KEY (customer_id) REFERENCES Customer(customer_id);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Booking
    ->     ADD CONSTRAINT fk_booking_event
    ->     FOREIGN KEY (event_id) REFERENCES Event(event_id);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

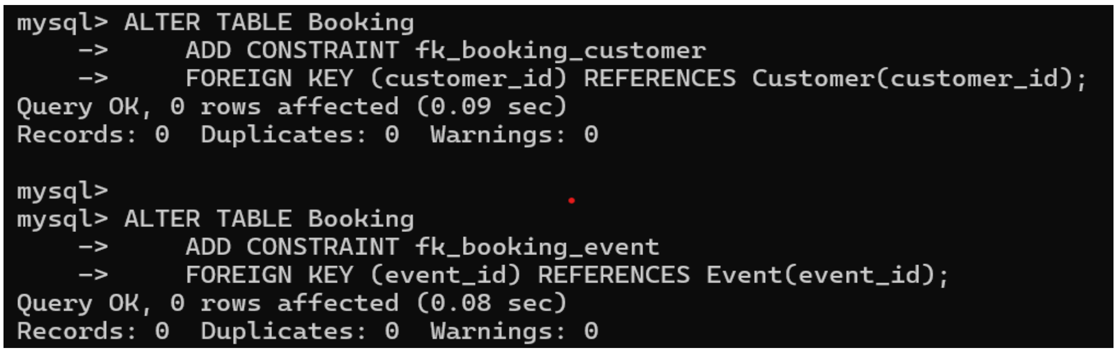
3. Create an ERD (Entity Relationship Diagram) for the database



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

ALTER TABLE Booking
    ADD CONSTRAINT fk_booking_customer
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id);

ALTER TABLE Booking
    ADD CONSTRAINT fk_booking_event
    FOREIGN KEY (event_id) REFERENCES Event(event_id);



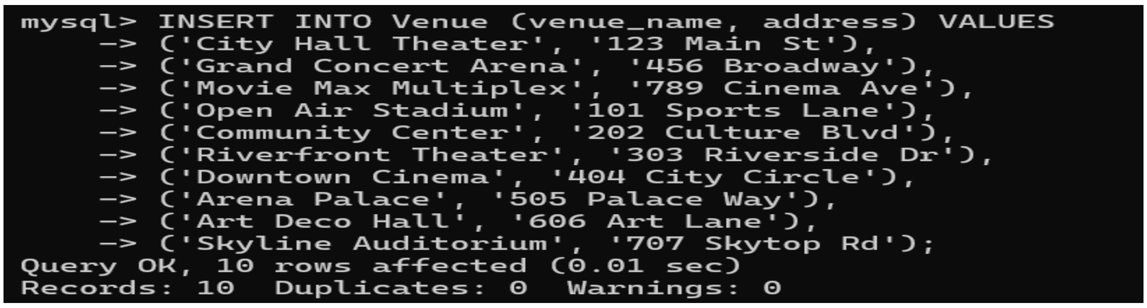
```
mysql> ALTER TABLE Booking
    ->     ADD CONSTRAINT fk_booking_customer
    ->     FOREIGN KEY (customer_id) REFERENCES Customer(customer_id);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Booking
    ->     ADD CONSTRAINT fk_booking_event
    ->     FOREIGN KEY (event_id) REFERENCES Event(event_id);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

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

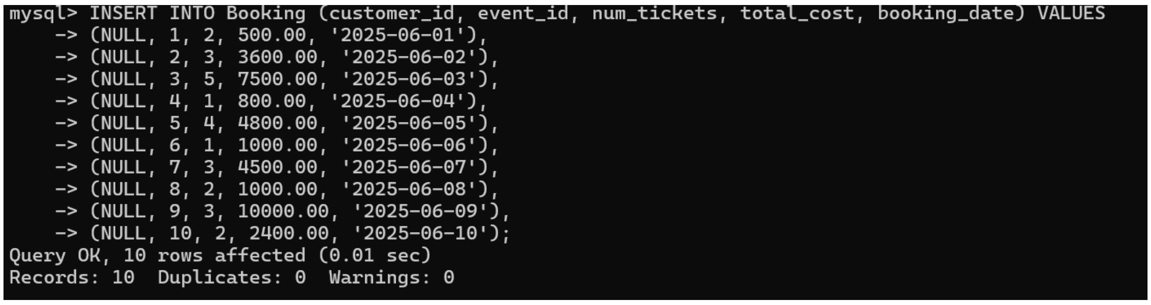
1. Write a SQL query to insert at least 10 sample records into each table.

```
INSERT INTO Venue (venue_name, address) VALUES
('City Hall Theater', '123 Main St'),
('Grand Concert Arena', '456 Broadway'),
('Movie Max Multiplex', '789 Cinema Ave'),
('Open Air Stadium', '101 Sports Lane'),
('Community Center', '202 Culture Blvd'),
('Riverfront Theater', '303 Riverside Dr'),
('Downtown Cinema', '404 City Circle'),
('Arena Palace', '505 Palace Way'),
('Art Deco Hall', '606 Art Lane'),
('Skyline Auditorium', '707 Skytop Rd');
```



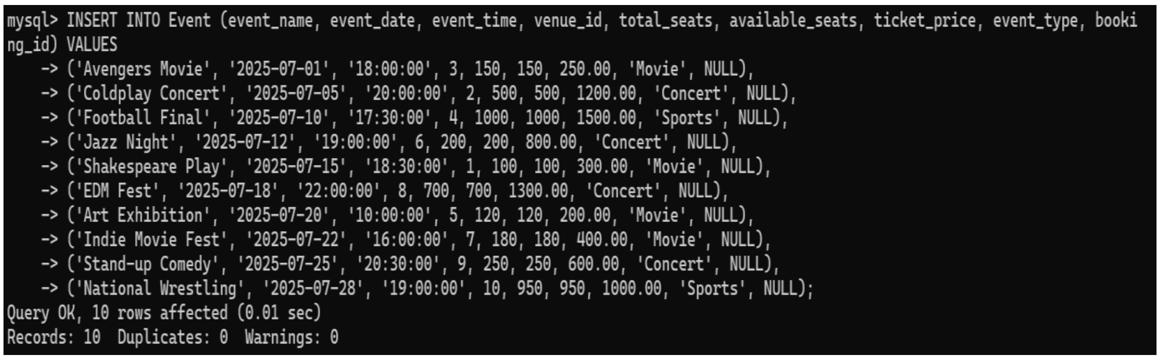
```
mysql> INSERT INTO Venue (venue_name, address) VALUES
    -> ('City Hall Theater', '123 Main St'),
    -> ('Grand Concert Arena', '456 Broadway'),
    -> ('Movie Max Multiplex', '789 Cinema Ave'),
    -> ('Open Air Stadium', '101 Sports Lane'),
    -> ('Community Center', '202 Culture Blvd'),
    -> ('Riverfront Theater', '303 Riverside Dr'),
    -> ('Downtown Cinema', '404 City Circle'),
    -> ('Arena Palace', '505 Palace Way'),
    -> ('Art Deco Hall', '606 Art Lane'),
    -> ('Skyline Auditorium', '707 Skytop Rd');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
INSERT INTO Booking (customer_id, event_id, num_tickets, total_cost, booking_date)
VALUES
(NULL, 1, 2, 500.00, '2025-06-01'),
(NULL, 2, 3, 3600.00, '2025-06-02'),
(NULL, 3, 5, 7500.00, '2025-06-03'),
(NULL, 4, 1, 800.00, '2025-06-04'),
(NULL, 5, 4,  4800.00, '2025-06-05'),
(NULL, 6, 1, 1000.00, '2025-06-06'),
(NULL, 7, 3, 4500.00, '2025-06-07'),
(NULL, 8, 2, 1000.00, '2025-06-08'),
(NULL, 9, 3, 10000.00, '2025-06-09'),
(NULL, 10, 2, 2400.00, '2025-06-10');
```



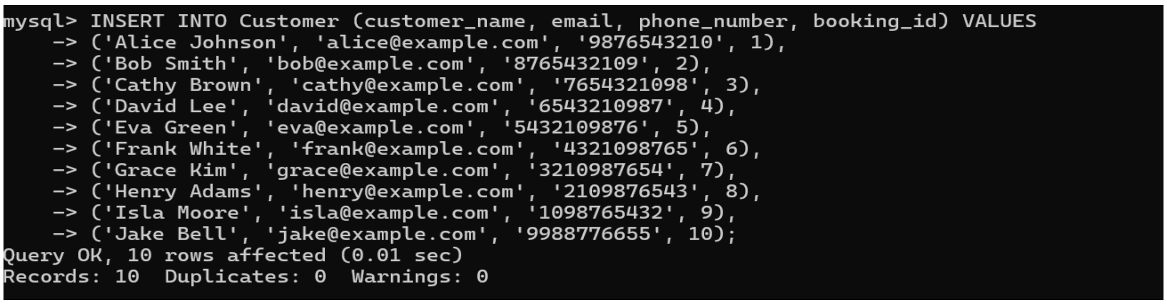
```
mysql> INSERT INTO Booking (customer_id, event_id, num_tickets, total_cost, booking_date) VALUES
    -> (NULL, 1, 2, 500.00, '2025-06-01'),
    -> (NULL, 2, 3, 3600.00, '2025-06-02'),
    -> (NULL, 3, 5, 7500.00, '2025-06-03'),
    -> (NULL, 4, 1, 800.00, '2025-06-04'),
    -> (NULL, 5, 4, 4800.00, '2025-06-05'),
    -> (NULL, 6, 1, 1000.00, '2025-06-06'),
    -> (NULL, 7, 3, 4500.00, '2025-06-07'),
    -> (NULL, 8, 2, 1000.00, '2025-06-08'),
    -> (NULL, 9, 3, 10000.00, '2025-06-09'),
    -> (NULL, 10, 2, 2400.00, '2025-06-10');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
INSERT INTO Event (event_name, event_date, event_time, venue_id, total_seats,
available_seats, ticket_price, event_type, booking_id) VALUES
('Avengers Movie', '2025-07-01', '18:00:00', 3, 150, 150, 250.00, 'Movie', NULL),
('Coldplay Concert', '2025-07-05', '20:00:00', 2, 500, 500, 1200.00, 'Concert', NULL),
('Football Final', '2025-07-10', '17:30:00', 4, 1000, 1000, 1500.00, 'Sports', NULL),
('Jazz Night', '2025-07-12', '19:00:00', 6, 200, 200, 800.00, 'Concert', NULL),
('Shakespeare Play', '2025-07-15', '18:30:00', 1, 100, 100, 300.00, 'Movie', NULL),
('EDM Fest', '2025-07-18', '22:00:00', 8, 700, 700, 1300.00, 'Concert', NULL),
('Art Exhibition', '2025-07-20', '10:00:00', 5, 120, 120, 200.00, 'Movie', NULL),
('Indie Movie Fest', '2025-07-22', '16:00:00', 7, 180, 180, 400.00, 'Movie', NULL),
('Stand-up Comedy', '2025-07-25', '20:30:00', 9, 250, 250, 600.00, 'Concert', NULL),
('National Wrestling', '2025-07-28', '19:00:00', 10, 950, 950, 1000.00, 'Sports', NULL);
```



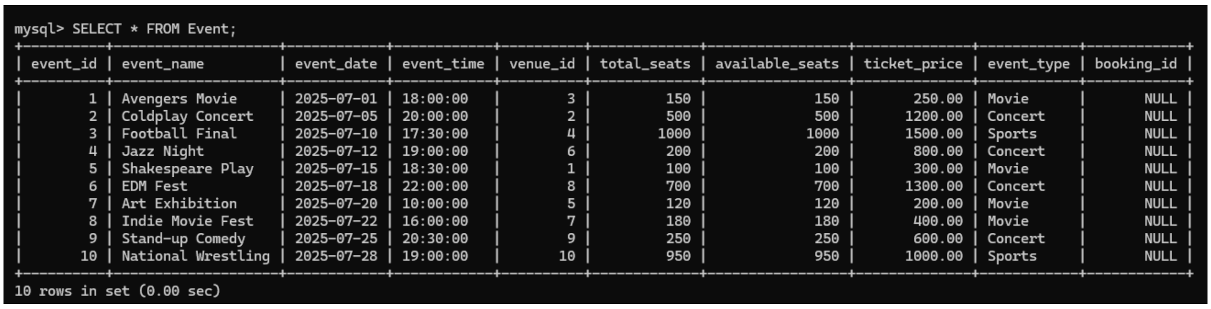
```
mysql> INSERT INTO Event (event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type, booki
ng_id) VALUES
    -> ('Avengers Movie', '2025-07-01', '18:00:00', 3, 150, 150, 250.00, 'Movie', NULL),
    -> ('Coldplay Concert', '2025-07-05', '20:00:00', 2, 500, 500, 1200.00, 'Concert', NULL),
    -> ('Football Final', '2025-07-10', '17:30:00', 4, 1000, 1000, 1500.00, 'Sports', NULL),
    -> ('Jazz Night', '2025-07-12', '19:00:00', 6, 200, 200, 800.00, 'Concert', NULL),
    -> ('Shakespeare Play', '2025-07-15', '18:30:00', 1, 100, 100, 300.00, 'Movie', NULL),
    -> ('EDM Fest', '2025-07-18', '22:00:00', 8, 700, 700, 1300.00, 'Concert', NULL),
    -> ('Art Exhibition', '2025-07-20', '10:00:00', 5, 120, 120, 200.00, 'Movie', NULL),
    -> ('Indie Movie Fest', '2025-07-22', '16:00:00', 7, 180, 180, 400.00, 'Movie', NULL),
    -> ('Stand-up Comedy', '2025-07-25', '20:30:00', 9, 250, 250, 600.00, 'Concert', NULL),
    -> ('National Wrestling', '2025-07-28', '19:00:00', 10, 950, 950, 1000.00, 'Sports', NULL);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
INSERT INTO Customer (customer_name, email, phone_number, booking_id) VALUES
('Alice Johnson', 'alice@example.com', '9876543210', 1),
('Bob Smith', 'bob@example.com', '8765432109', 2),
('Cathy Brown', 'cathy@example.com', '7654321098', 3),
('David Lee', 'david@example.com', '6543210987', 4),
('Eva Green', 'eva@example.com', '5432109876', 5),
('Frank White', 'frank@example.com', '4321098765', 6),
('Grace Kim', 'grace@example.com', '3210987654', 7),
('Henry Adams', 'henry@example.com', '2109876543', 8),
('Isla Moore', 'isla@example.com', '1098765432', 9),
('Jake Bell', 'jake@example.com', '9988776655', 10);
```



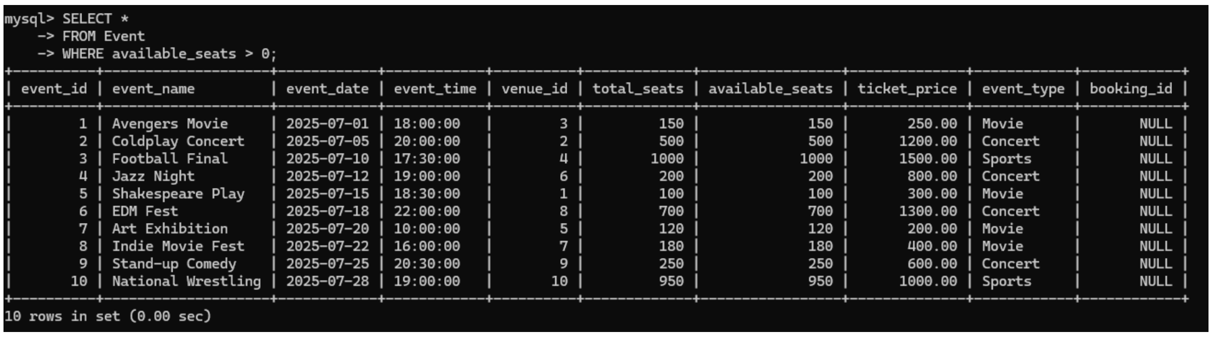
```
mysql> INSERT INTO Customer (customer_name, email, phone_number, booking_id) VALUES
    -> ('Alice Johnson', 'alice@example.com', '9876543210', 1),
    -> ('Bob Smith', 'bob@example.com', '8765432109', 2),
    -> ('Cathy Brown', 'cathy@example.com', '7654321098', 3),
    -> ('David Lee', 'david@example.com', '6543210987', 4),
    -> ('Eva Green', 'eva@example.com', '5432109876', 5),
    -> ('Frank White', 'frank@example.com', '4321098765', 6),
    -> ('Grace Kim', 'grace@example.com', '3210987654', 7),
    -> ('Henry Adams', 'henry@example.com', '2109876543', 8),
    -> ('Isla Moore', 'isla@example.com', '1098765432', 9),
    -> ('Jake Bell', 'jake@example.com', '9988776655', 10);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

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



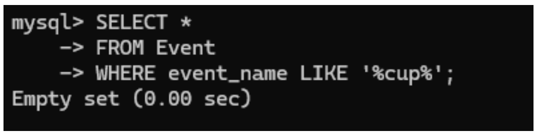
```
mysql> SELECT * FROM Event;
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name         | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Avengers Movie     | 2025-07-01 | 18:00:00   |        3 |         150 |             150 |       250.00 | Movie      |       NULL |
|        2 | Coldplay Concert   | 2025-07-05 | 20:00:00   |        2 |         500 |             500 |      1200.00 | Concert    |       NULL |
|        3 | Football Final     | 2025-07-10 | 17:30:00   |        4 |        1000 |            1000 |      1500.00 | Sports     |       NULL |
|        4 | Jazz Night         | 2025-07-12 | 19:00:00   |        6 |         200 |             200 |       800.00 | Concert    |       NULL |
|        5 | Shakespeare Play   | 2025-07-15 | 18:30:00   |        1 |         100 |             100 |       300.00 | Movie      |       NULL |
|        6 | EDM Fest           | 2025-07-18 | 22:00:00   |        8 |         700 |             700 |      1300.00 | Concert    |       NULL |
|        7 | Art Exhibition     | 2025-07-20 | 10:00:00   |        5 |         120 |             120 |       200.00 | Movie      |       NULL |
|        8 | Indie Movie Fest   | 2025-07-22 | 16:00:00   |        7 |         180 |             180 |       400.00 | Movie      |       NULL |
|        9 | Stand-up Comedy    | 2025-07-25 | 20:30:00   |        9 |         250 |             250 |       600.00 | Concert    |       NULL |
|       10 | National Wrestling | 2025-07-28 | 19:00:00   |       10 |         950 |             950 |      1000.00 | Sports     |       NULL |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
10 rows in set (0.00 sec)
```

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



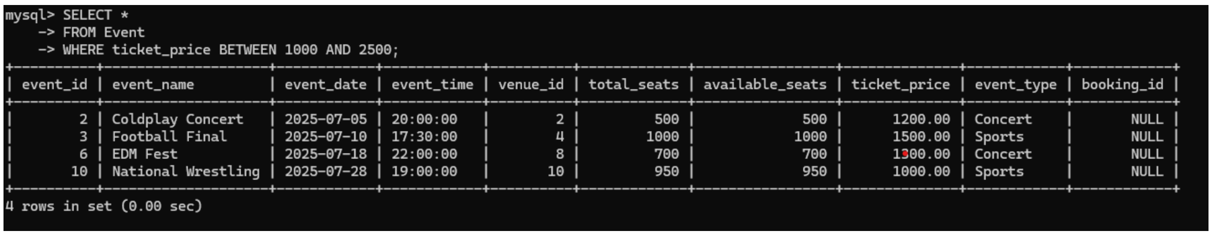
```
mysql> SELECT *
    -> FROM Event
    -> WHERE available_seats > 0;
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name         | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Avengers Movie     | 2025-07-01 | 18:00:00   |        3 |         150 |             150 |       250.00 | Movie      |       NULL |
|        2 | Coldplay Concert   | 2025-07-05 | 20:00:00   |        2 |         500 |             500 |      1200.00 | Concert    |       NULL |
|        3 | Football Final     | 2025-07-10 | 17:30:00   |        4 |        1000 |            1000 |      1500.00 | Sports     |       NULL |
|        4 | Jazz Night         | 2025-07-12 | 19:00:00   |        6 |         200 |             200 |       800.00 | Concert    |       NULL |
|        5 | Shakespeare Play   | 2025-07-15 | 18:30:00   |        1 |         100 |             100 |       300.00 | Movie      |       NULL |
|        6 | EDM Fest           | 2025-07-18 | 22:00:00   |        8 |         700 |             700 |      1300.00 | Concert    |       NULL |
|        7 | Art Exhibition     | 2025-07-20 | 10:00:00   |        5 |         120 |             120 |       200.00 | Movie      |       NULL |
|        8 | Indie Movie Fest   | 2025-07-22 | 16:00:00   |        7 |         180 |             180 |       400.00 | Movie      |       NULL |
|        9 | Stand-up Comedy    | 2025-07-25 | 20:30:00   |        9 |         250 |             250 |       600.00 | Concert    |       NULL |
|       10 | National Wrestling | 2025-07-28 | 19:00:00   |       10 |         950 |             950 |      1000.00 | Sports     |       NULL |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
10 rows in set (0.00 sec)
```

4. Write a SQL query to select events name partial match with 'cup'
   SELECT *
   FROM Event
   WHERE event_name LIKE '%cup%';



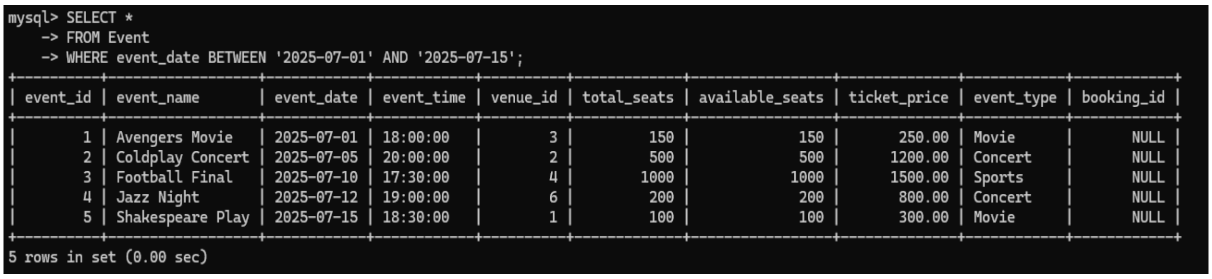
```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_name LIKE '%cup%';
Empty set (0.00 sec)
```

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;



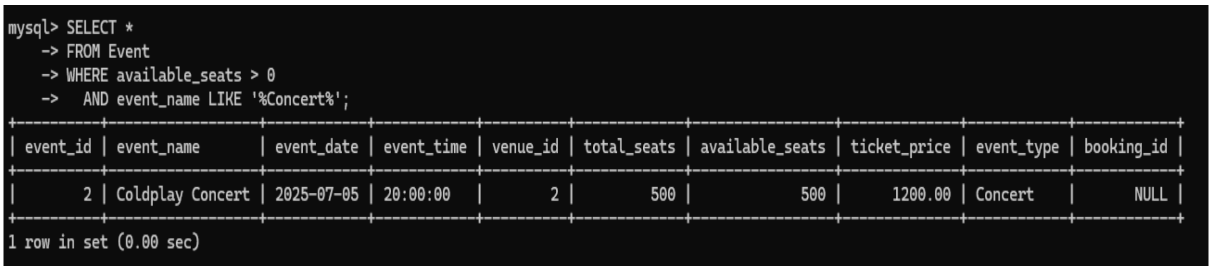
```
mysql> SELECT *
    -> FROM Event
    -> WHERE ticket_price BETWEEN 1000 AND 2500;
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name         | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        2 | Coldplay Concert   | 2025-07-05 | 20:00:00   |        2 |         500 |             500 |      1200.00 | Concert    |       NULL |
|        3 | Football Final     | 2025-07-10 | 17:30:00   |        4 |        1000 |            1000 |      1500.00 | Sports     |       NULL |
|        6 | EDM Fest           | 2025-07-18 | 22:00:00   |        8 |         700 |             700 |      1300.00 | Concert    |       NULL |
|       10 | National Wrestling | 2025-07-28 | 19:00:00   |       10 |         950 |             950 |      1000.00 | Sports     |       NULL |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
4 rows in set (0.00 sec)
```

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 '2025-07-15';



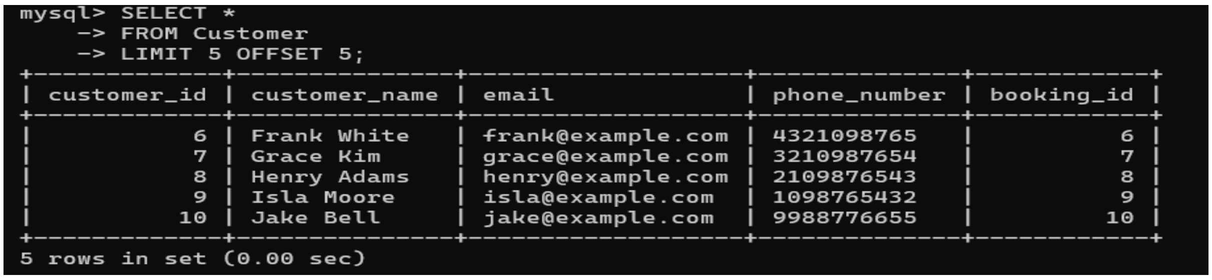
```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_date BETWEEN '2025-07-01' AND '2025-07-15';
+----------+------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name       | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Avengers Movie   | 2025-07-01 | 18:00:00   |        3 |         150 |             150 |       250.00 | Movie      |       NULL |
|        2 | Coldplay Concert | 2025-07-05 | 20:00:00   |        2 |         500 |             500 |      1200.00 | Concert    |       NULL |
|        3 | Football Final   | 2025-07-10 | 17:30:00   |        4 |        1000 |            1000 |      1500.00 | Sports     |       NULL |
|        4 | Jazz Night       | 2025-07-12 | 19:00:00   |        6 |         200 |             200 |       800.00 | Concert    |       NULL |
|        5 | Shakespeare Play | 2025-07-15 | 18:30:00   |        1 |         100 |             100 |       300.00 | Movie      |       NULL |
+----------+------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
5 rows in set (0.00 sec)
```

7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.
   SELECT *
   FROM Event
   WHERE available_seats > 0
     AND event_name LIKE '%Concert%';



```
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 | booking_id |
+----------+------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        2 | Coldplay Concert | 2025-07-05 | 20:00:00   |        2 |         500 |             500 |      1200.00 | Concert    |       NULL |
+----------+------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
1 row in set (0.00 sec)
```

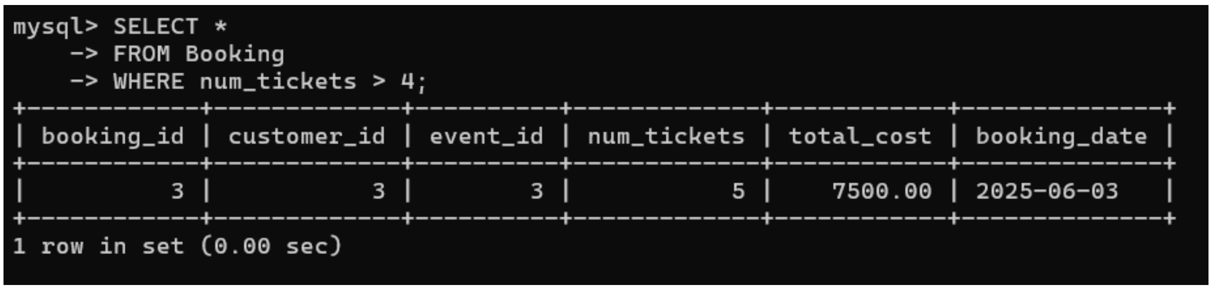
8. Write a SQL query to retrieve users in batches of 5, starting from the 6th user.
   SELECT *
   FROM Customer
   LIMIT 5 OFFSET 5;



```
mysql> SELECT *
    -> FROM Customer
    -> LIMIT 5 OFFSET 5;
+-------------+---------------+--------------------+--------------+------------+
| customer_id | customer_name | email              | phone_number | booking_id |
+-------------+---------------+--------------------+--------------+------------+
|           6 | Frank White   | frank@example.com  | 4321098765   |          6 |
|           7 | Grace Kim     | grace@example.com  | 3210987654   |          7 |
|           8 | Henry Adams   | henry@example.com  | 2109876543   |          8 |
|           9 | Isla Moore    | isla@example.com   | 1098765432   |          9 |
|          10 | Jake Bell     | jake@example.com   | 9988776655   |         10 |
+-------------+---------------+--------------------+--------------+------------+
5 rows in set (0.00 sec)
```

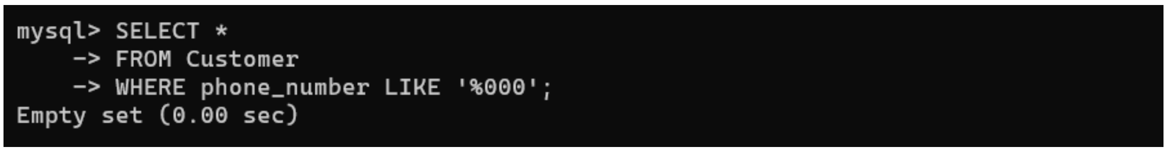
9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.

   SELECT *
   FROM Booking
   WHERE num_tickets > 4;



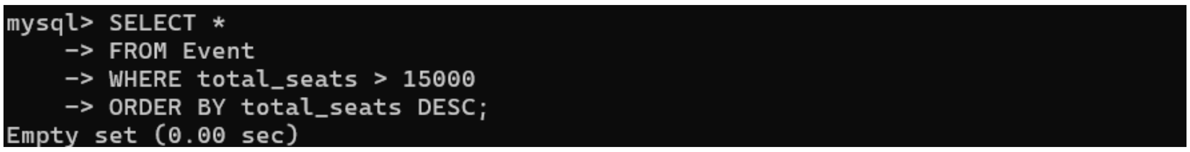
```
mysql> SELECT *
    -> FROM Booking
    -> WHERE num_tickets > 4;
+------------+-------------+----------+-------------+------------+--------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
+------------+-------------+----------+-------------+------------+--------------+
|          3 |           3 |        3 |           5 |    7500.00 | 2025-06-03   |
+------------+-------------+----------+-------------+------------+--------------+
1 row in set (0.00 sec)
```

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';
Empty set (0.00 sec)
```

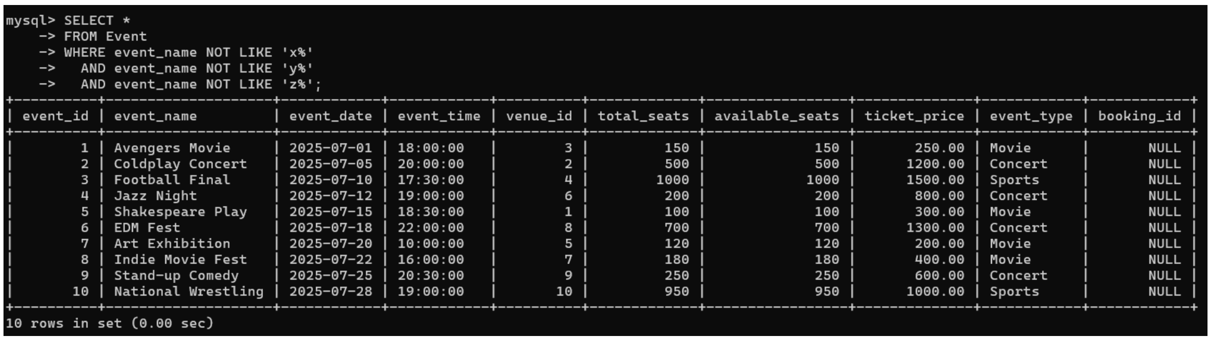
11. Write a SQL query to retrieve the events in order whose seat capacity more than 15000.
    SELECT *
    FROM Event
    WHERE total_seats > 15000
    ORDER BY total_seats DESC;



```
mysql> SELECT *
    -> FROM Event
    -> WHERE total_seats > 15000
    -> ORDER BY total_seats DESC;
Empty set (0.00 sec)
```

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

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

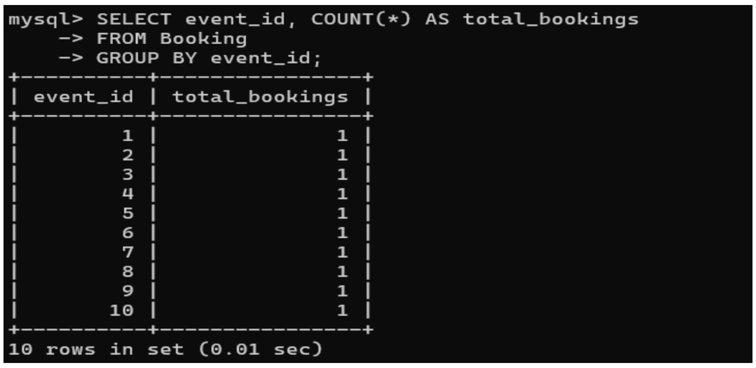


```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_name NOT LIKE 'x%'
    ->   AND event_name NOT LIKE 'y%'
    ->   AND event_name NOT LIKE 'z%';
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name         | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Avengers Movie     | 2025-07-01 | 18:00:00   |        3 |         150 |             150 |       250.00 | Movie      |       NULL |
|        2 | Coldplay Concert   | 2025-07-05 | 20:00:00   |        2 |         500 |             500 |      1200.00 | Concert    |       NULL |
|        3 | Football Final     | 2025-07-10 | 17:30:00   |        4 |        1000 |            1000 |      1500.00 | Sports     |       NULL |
|        4 | Jazz Night         | 2025-07-12 | 19:00:00   |        6 |         200 |             200 |       800.00 | Concert    |       NULL |
|        5 | Shakespeare Play   | 2025-07-15 | 18:30:00   |        1 |         100 |             100 |       300.00 | Movie      |       NULL |
|        6 | EDM Fest           | 2025-07-18 | 22:00:00   |        8 |         700 |             700 |      1300.00 | Concert    |       NULL |
|        7 | Art Exhibition     | 2025-07-20 | 10:00:00   |        5 |         120 |             120 |       200.00 | Movie      |       NULL |
|        8 | Indie Movie Fest   | 2025-07-22 | 16:00:00   |        7 |         180 |             180 |       400.00 | Movie      |       NULL |
|        9 | Stand-up Comedy    | 2025-07-25 | 20:30:00   |        9 |         250 |             250 |       600.00 | Concert    |       NULL |
|       10 | National Wrestling | 2025-07-28 | 19:00:00   |       10 |         950 |             950 |      1000.00 | Sports     |       NULL |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
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.

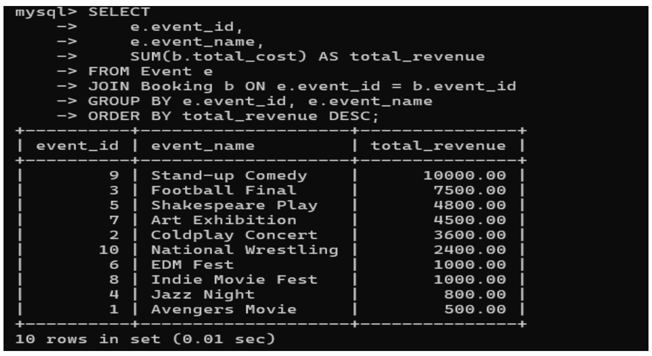
```
SELECT event_id, COUNT(*) AS total_bookings
FROM Booking
GROUP BY event_id;
```



```
mysql> SELECT event_id, COUNT(*) AS total_bookings
    -> FROM Booking
    -> GROUP BY event_id;
+----------+----------------+
| event_id | total_bookings |
+----------+----------------+
|        1 |              1 |
|        2 |              1 |
|        3 |              1 |
|        4 |              1 |
|        5 |              1 |
|        6 |              1 |
|        7 |              1 |
|        8 |              1 |
|        9 |              1 |
|       10 |              1 |
+----------+----------------+
10 rows in set (0.01 sec)
```

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

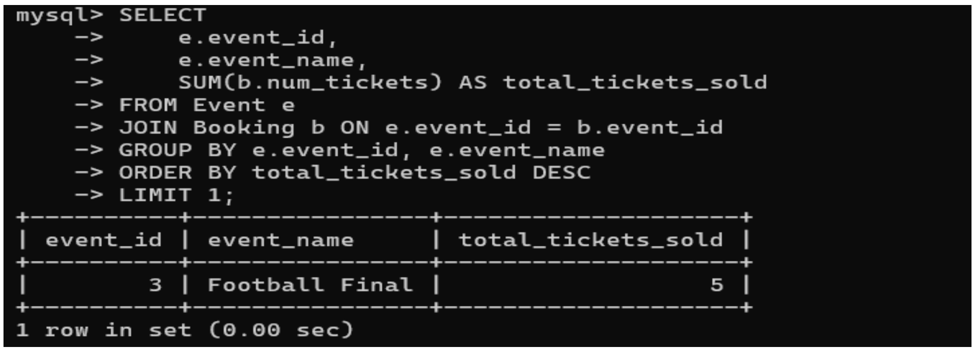
```
SELECT
    e.event_id,
    e.event_name,
    SUM(b.total_cost) AS total_revenue
FROM Event e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY e.event_id, e.event_name
            ORDER BY total_revenue DESC;
```



```
mysql> SELECT
    ->     e.event_id,
    ->     e.event_name,
    ->     SUM(b.total_cost) AS total_revenue
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY e.event_id, e.event_name
    -> ORDER BY total_revenue DESC;
+----------+--------------------+---------------+
| event_id | event_name         | total_revenue |
+----------+--------------------+---------------+
|        9 | Stand-up Comedy    |      10000.00 |
|        3 | Football Final     |       7500.00 |
|        5 | Shakespeare Play   |       4800.00 |
|        7 | Art Exhibition     |       4500.00 |
|        2 | Coldplay Concert   |       3600.00 |
|       10 | National Wrestling |       2400.00 |
|        6 | EDM Fest           |       1000.00 |
|        8 | Indie Movie Fest   |       1000.00 |
|        4 | Jazz Night         |        800.00 |
|        1 | Avengers Movie     |        500.00 |
+----------+--------------------+---------------+
10 rows in set (0.01 sec)
```

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

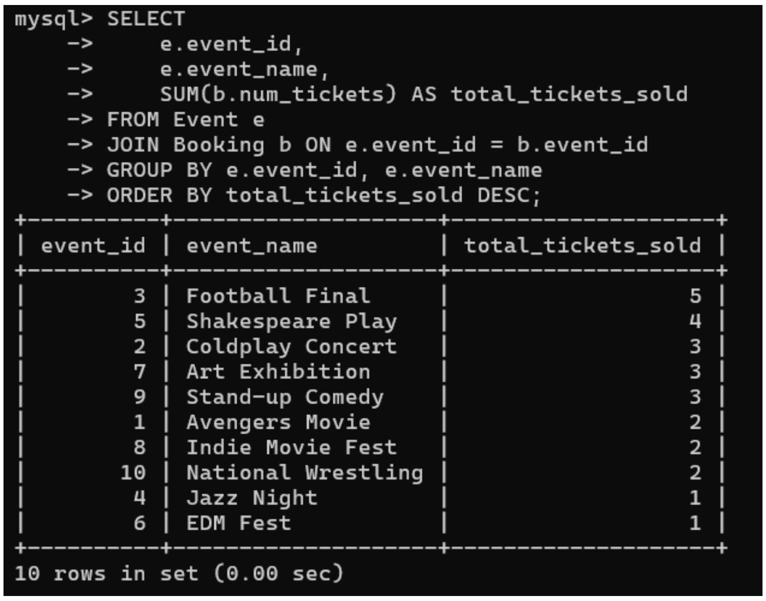
```
SELECT
    e.event_id,
    e.event_name,
    SUM(b.num_tickets) AS total_tickets_sold
FROM Event e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY e.event_id, e.event_name
ORDER BY total_tickets_sold DESC
LIMIT 1;
```



```
mysql> SELECT
    ->     e.event_id,
    ->     e.event_name,
    ->     SUM(b.num_tickets) AS total_tickets_sold
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY e.event_id, e.event_name
    -> ORDER BY total_tickets_sold DESC
    -> LIMIT 1;
+----------+----------------+--------------------+
| event_id | event_name     | total_tickets_sold |
+----------+----------------+--------------------+
|        3 | Football Final |                  5 |
+----------+----------------+--------------------+
1 row in set (0.00 sec)
```

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

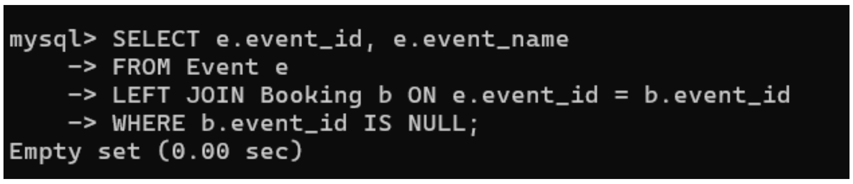
```
SELECT
    e.event_id,
    e.event_name,
    SUM(b.num_tickets) AS total_tickets_sold
FROM Event e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY e.event_id, e.event_name
ORDER BY total_tickets_sold DESC;
```



```
mysql> SELECT
    ->     e.event_id,
    ->     e.event_name,
    ->     SUM(b.num_tickets) AS total_tickets_sold
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY e.event_id, e.event_name
    -> ORDER BY total_tickets_sold DESC;
+----------+-------------------+--------------------+
| event_id | event_name        | total_tickets_sold |
+----------+-------------------+--------------------+
|        3 | Football Final    |                  5 |
|        5 | Shakespeare Play  |                  4 |
|        2 | Coldplay Concert  |                  3 |
|        7 | Art Exhibition    |                  3 |
|        9 | Stand-up Comedy   |                  3 |
|        1 | Avengers Movie    |                  2 |
|        8 | Indie Movie Fest  |                  2 |
|       10 | National Wrestling |                 2 |
|        4 | Jazz Night        |                  1 |
|        6 | EDM Fest          |                  1 |
+----------+-------------------+--------------------+
10 rows in set (0.00 sec)
```

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

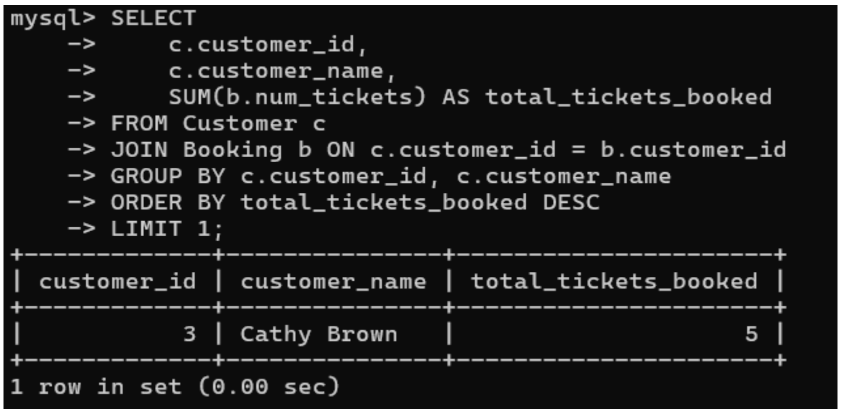
```
SELECT e.event_id, e.event_name
FROM Event e
LEFT JOIN Booking b ON e.event_id = b.event_id
WHERE b.event_id IS NULL;
```



```
mysql> SELECT e.event_id, e.event_name
    -> FROM Event e
    -> LEFT JOIN Booking b ON e.event_id = b.event_id
    -> WHERE b.event_id IS NULL;
Empty set (0.00 sec)
```

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

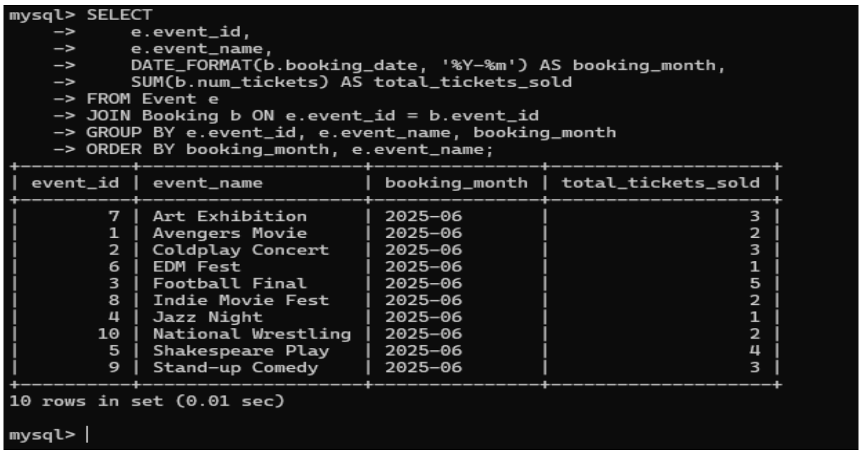
```
SELECT
  c.customer_id,
  c.customer_name,
  SUM(b.num_tickets) AS total_tickets_booked
FROM Customer c
JOIN Booking b ON c.customer_id = b.customer_id
GROUP BY c.customer_id, c.customer_name
ORDER BY total_tickets_booked DESC
LIMIT 1;
```



```
mysql> SELECT
    ->     c.customer_id,
    ->     c.customer_name,
    ->     SUM(b.num_tickets) 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
    -> LIMIT 1;
+-------------+---------------+----------------------+
| customer_id | customer_name | total_tickets_booked |
+-------------+---------------+----------------------+
|           3 | Cathy Brown   |                    5 |
+-------------+---------------+----------------------+
1 row in set (0.00 sec)
```

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

```
SELECT
  e.event_id,
  e.event_name,
  DATE_FORMAT(b.booking_date, '%Y-%m') AS booking_month,
  SUM(b.num_tickets) AS total_tickets_sold
FROM Event e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY e.event_id, e.event_name, booking_month
ORDER BY booking_month, e.event_name;
```

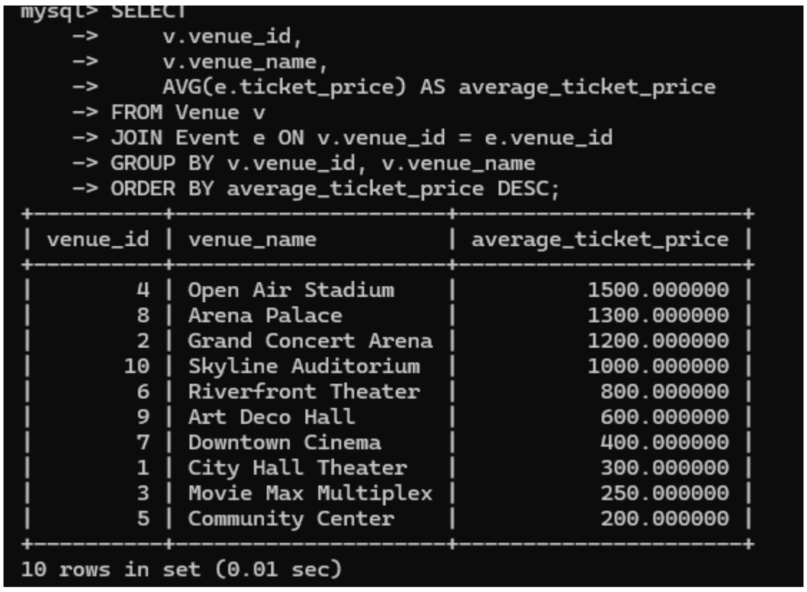


```
mysql> SELECT
    ->     e.event_id,
    ->     e.event_name,
    ->     DATE_FORMAT(b.booking_date, '%Y-%m') AS booking_month,
    ->     SUM(b.num_tickets) AS total_tickets_sold
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY e.event_id, e.event_name, booking_month
    -> ORDER BY booking_month, e.event_name;
+----------+-------------------+---------------+--------------------+
| event_id | event_name        | booking_month | total_tickets_sold |
+----------+-------------------+---------------+--------------------+
|        7 | Art Exhibition    | 2025-06       |                  3 |
|        1 | Avengers Movie    | 2025-06       |                  2 |
|        2 | Coldplay Concert  | 2025-06       |                  3 |
|        6 | EDM Fest          | 2025-06       |                  1 |
|        3 | Football Final    | 2025-06       |                  5 |
|        8 | Indie Movie Fest  | 2025-06       |                  2 |
|        4 | Jazz Night        | 2025-06       |                  1 |
|       10 | National Wrestling| 2025-06       |                  2 |
|        5 | Shakespeare Play  | 2025-06       |                  4 |
|        9 | Stand-up Comedy   | 2025-06       |                  3 |
+----------+-------------------+---------------+--------------------+
10 rows in set (0.01 sec)

mysql> |
```

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

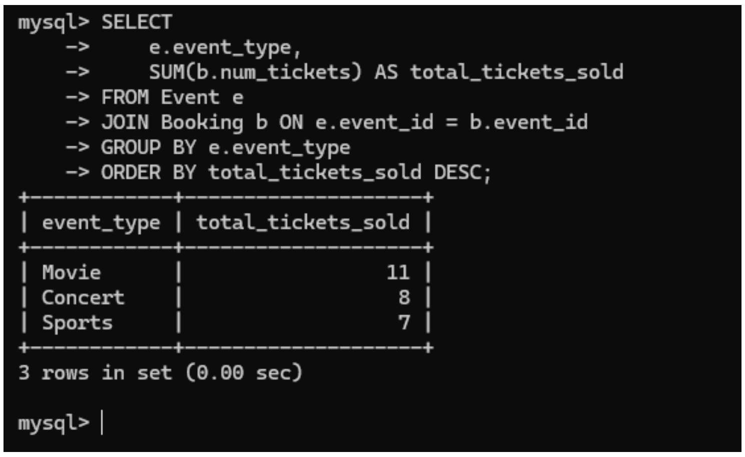
```
SELECT
   v.venue_id,
   v.venue_name,
   AVG(e.ticket_price) AS average_ticket_price
FROM Venu v
JOIN Event e ON v.venue_id = e.venue_id
GROUP BY v.venue_id, v.venue_name
ORDER BY average_ticket_price DESC;
```



```
mysql> SELECT
    ->     v.venue_id,
    ->     v.venue_name,
    ->     AVG(e.ticket_price) AS average_ticket_price
    -> FROM Venue v
    -> JOIN Event e ON v.venue_id = e.venue_id
    -> GROUP BY v.venue_id, v.venue_name
    -> ORDER BY average_ticket_price DESC;
+----------+--------------------+----------------------+
| venue_id | venue_name         | average_ticket_price |
+----------+--------------------+----------------------+
|        4 | Open Air Stadium   |          1500.000000 |
|        8 | Arena Palace       |          1300.000000 |
|        2 | Grand Concert Arena|          1200.000000 |
|       10 | Skyline Auditorium |          1000.000000 |
|        6 | Riverfront Theater |           800.000000 |
|        9 | Art Deco Hall      |           600.000000 |
|        7 | Downtown Cinema    |           400.000000 |
|        1 | City Hall Theater  |           300.000000 |
|        3 | Movie Max Multiplex|           250.000000 |
|        5 | Community Center   |           200.000000 |
+----------+--------------------+----------------------+
10 rows in set (0.01 sec)
```

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

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

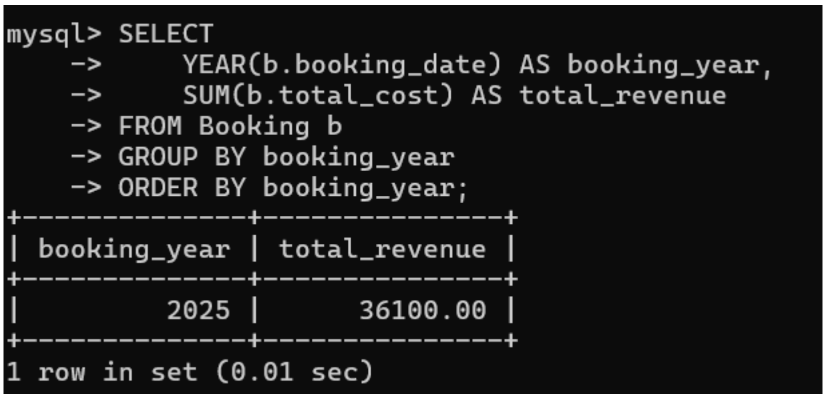


```
mysql> SELECT
    ->     e.event_type,
    ->     SUM(b.num_tickets) AS total_tickets_sold
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY e.event_type
    -> ORDER BY total_tickets_sold DESC;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Movie      |                 11 |
| Concert    |                  8 |
| Sports     |                  7 |
+------------+--------------------+
3 rows in set (0.00 sec)

mysql> |
```

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

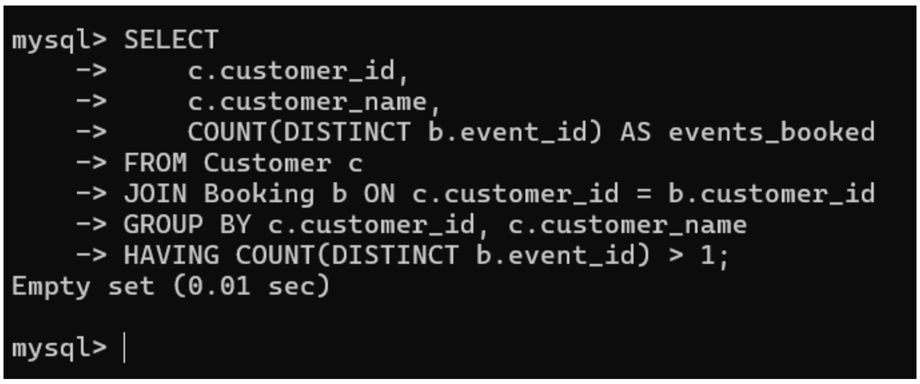
```
SELECT
   YEAR(b.booking_date) AS booking_year,
   SUM(b.total_cost) AS total_revenue
FROM Booking b
GROUP BY booking_year
ORDER BY booking_year;
```



```
mysql> SELECT
    ->     YEAR(b.booking_date) AS booking_year,
    ->     SUM(b.total_cost) AS total_revenue
    -> FROM Booking b
    -> GROUP BY booking_year
    -> ORDER BY booking_year;
+--------------+---------------+
| booking_year | total_revenue |
+--------------+---------------+
|         2025 |      36100.00 |
+--------------+---------------+
1 row in set (0.01 sec)
```

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

```
SELECT
   c.customer_id,
   c.customer_name,
   COUNT(DISTINCT b.event_id) AS events_booked
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;
```

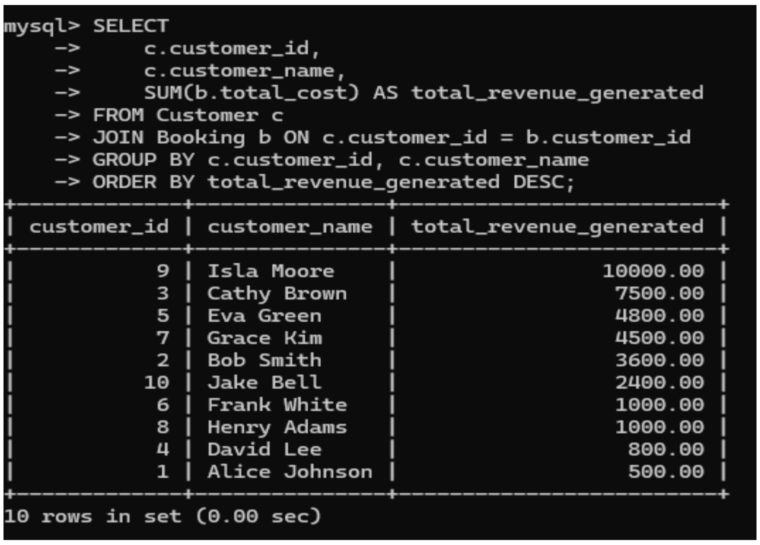


```
mysql> SELECT
    ->     c.customer_id,
    ->     c.customer_name,
    ->     COUNT(DISTINCT b.event_id) AS events_booked
    -> FROM Customer c
    -> JOIN Booking b ON c.customer_id = b.customer_id
    -> GROUP BY c.customer_id, c.customer_name
    -> HAVING COUNT(DISTINCT b.event_id) > 1;
Empty set (0.01 sec)

mysql> |
```

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

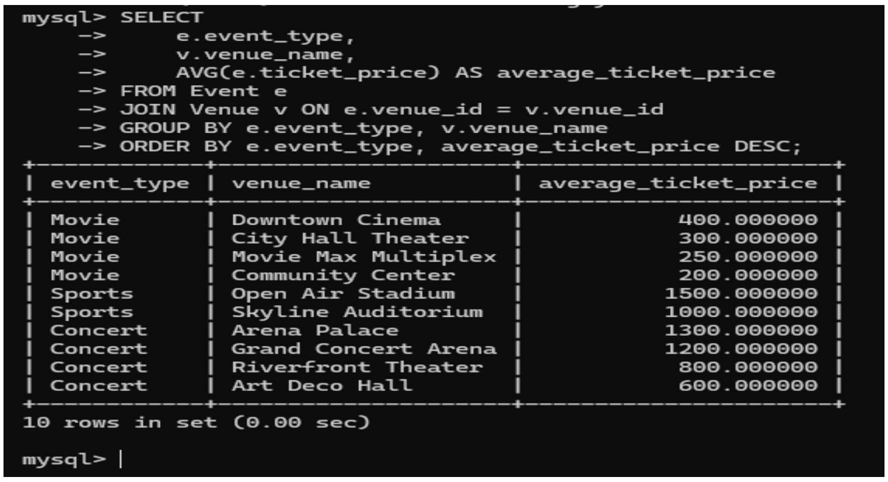
```
SELECT
   c.customer_id,
   c.customer_name,
   SUM(b.total_cost) AS total_revenue_generated
FROM Customer c
JOIN Booking b ON c.customer_id = b.customer_id
GROUP BY c.customer_id, c.customer_name
ORDER BY total_revenue_generated DESC;
```



```
mysql> SELECT
    ->     c.customer_id,
    ->     c.customer_name,
    ->     SUM(b.total_cost) AS total_revenue_generated
    -> FROM Customer c
    -> JOIN Booking b ON c.customer_id = b.customer_id
    -> GROUP BY c.customer_id, c.customer_name
    -> ORDER BY total_revenue_generated DESC;
+-------------+---------------+-------------------------+
| customer_id | customer_name | total_revenue_generated |
+-------------+---------------+-------------------------+
|           9 | Isla Moore    |                10000.00 |
|           3 | Cathy Brown   |                 7500.00 |
|           5 | Eva Green     |                 4800.00 |
|           7 | Grace Kim     |                 4500.00 |
|           2 | Bob Smith     |                 3600.00 |
|          10 | Jake Bell     |                 2400.00 |
|           6 | Frank White   |                 1000.00 |
|           8 | Henry Adams   |                 1000.00 |
|           4 | David Lee     |                  800.00 |
|           1 | Alice Johnson |                  500.00 |
+-------------+---------------+-------------------------+
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.

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

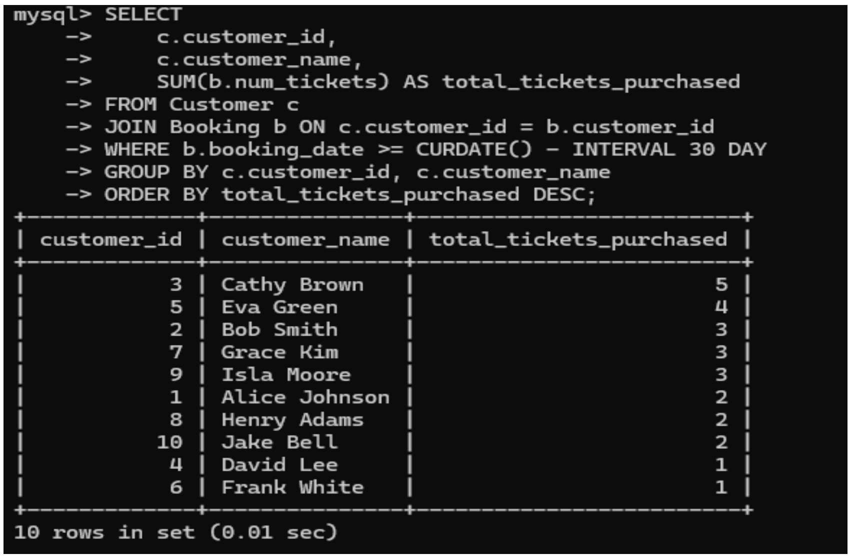


```
mysql> SELECT
    ->     e.event_type,
    ->     v.venue_name,
    ->     AVG(e.ticket_price) AS average_ticket_price
    -> FROM Event e
    -> JOIN Venue v ON e.venue_id = v.venue_id
    -> GROUP BY e.event_type, v.venue_name
    -> ORDER BY e.event_type, average_ticket_price DESC;
+------------+---------------------+----------------------+
| event_type | venue_name          | average_ticket_price |
+------------+---------------------+----------------------+
| Movie      | Downtown Cinema     |           400.000000 |
| Movie      | City Hall Theater   |           300.000000 |
| Movie      | Movie Max Multiplex |           250.000000 |
| Movie      | Community Center    |           200.000000 |
| Sports     | Open Air Stadium    |          1500.000000 |
| Sports     | Skyline Auditorium  |          1000.000000 |
| Concert    | Arena Palace        |          1300.000000 |
| Concert    | Grand Concert Arena |          1200.000000 |
| Concert    | Riverfront Theater  |           800.000000 |
| Concert    | Art Deco Hall       |           600.000000 |
+------------+---------------------+----------------------+
10 rows in set (0.00 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_id,
  c.customer_name,
  SUM(b.num_tickets) AS total_tickets_purchased
FROM Customer c
JOIN Booking b ON c.customer_id = b.customer_id
WHERE b.booking_date >= CURDATE() - INTERVAL 30 DAY
GROUP BY c.customer_id, c.customer_name
ORDER BY total_tickets_purchased DESC;
```

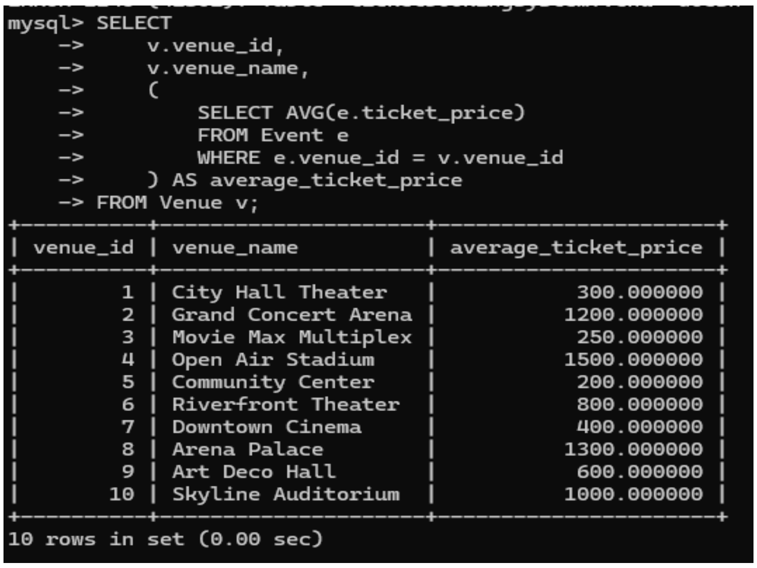


```
mysql> SELECT
    ->     c.customer_id,
    ->     c.customer_name,
    ->     SUM(b.num_tickets) AS total_tickets_purchased
    -> FROM Customer c
    -> JOIN Booking b ON c.customer_id = b.customer_id
    -> WHERE b.booking_date >= CURDATE() - INTERVAL 30 DAY
    -> GROUP BY c.customer_id, c.customer_name
    -> ORDER BY total_tickets_purchased DESC;
+-------------+---------------+-------------------------+
| customer_id | customer_name | total_tickets_purchased |
+-------------+---------------+-------------------------+
|           3 | Cathy Brown   |                       5 |
|           5 | Eva Green     |                       4 |
|           2 | Bob Smith     |                       3 |
|           7 | Grace Kim     |                       3 |
|           9 | Isla Moore    |                       3 |
|           1 | Alice Johnson |                       2 |
|           8 | Henry Adams   |                       2 |
|          10 | Jake Bell     |                       2 |
|           4 | David Lee     |                       1 |
|           6 | Frank White   |                       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.

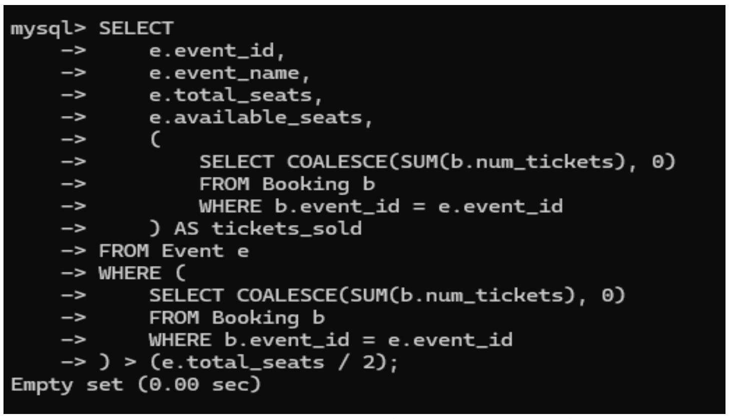
```
SELECT
  v.venue_id,
  v.venue_name,
  (
    SELECT AVG(e.ticket_price)
    FROM Event e
    WHERE e.venue_id = v.venue_id
  ) AS average_ticket_price
FROM Venu v;
```



```
mysql> SELECT
    ->     v.venue_id,
    ->     v.venue_name,
    ->     (
    ->         SELECT AVG(e.ticket_price)
    ->         FROM Event e
    ->         WHERE e.venue_id = v.venue_id
    ->     ) AS average_ticket_price
    -> FROM Venue v;
+----------+---------------------+----------------------+
| venue_id | venue_name          | average_ticket_price |
+----------+---------------------+----------------------+
|        1 | City Hall Theater   |           300.000000 |
|        2 | Grand Concert Arena |          1200.000000 |
|        3 | Movie Max Multiplex |           250.000000 |
|        4 | Open Air Stadium    |          1500.000000 |
|        5 | Community Center    |           200.000000 |
|        6 | Riverfront Theater  |           800.000000 |
|        7 | Downtown Cinema     |           400.000000 |
|        8 | Arena Palace        |          1300.000000 |
|        9 | Art Deco Hall       |           600.000000 |
|       10 | Skyline Auditorium  |          1000.000000 |
+----------+---------------------+----------------------+
10 rows in set (0.00 sec)
```

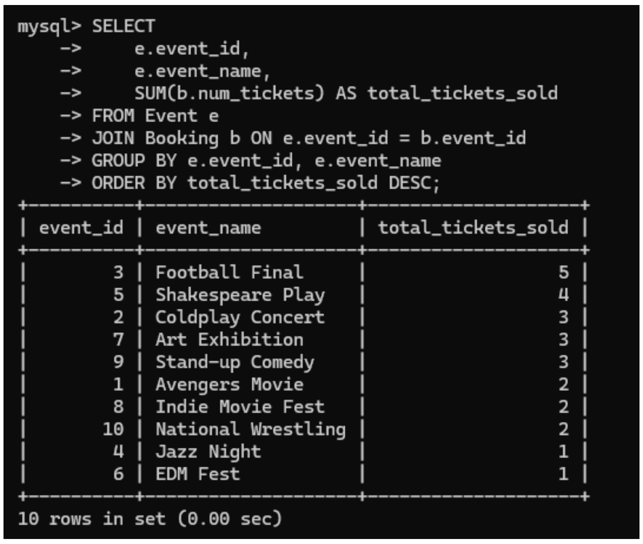
2. Find Events with More Than 50% of Tickets Sold using subquery.

```
SELECT
  e.event_id,
  e.event_name,
  e.total_seats,
  e.available_seats,
  (
    SELECT COALESCE(SUM(b.num_tickets), 0)
    FROM Booking b
    WHERE b.event_id = e.event_id
  ) AS tickets_sold
FROM Event e
WHERE (
  SELECT COALESCE(SUM(b.num_tickets), 0)
  FROM Booking b
  WHERE b.event_id = e.event_id
) > (e.total_seats / 2);
```


```
mysql> SELECT
    ->     e.event_id,
    ->     e.event_name,
    ->     e.total_seats,
    ->     e.available_seats,
    ->     (
    ->         SELECT COALESCE(SUM(b.num_tickets), 0)
    ->         FROM Booking b
    ->         WHERE b.event_id = e.event_id
    ->     ) AS tickets_sold
    -> FROM Event e
    -> WHERE (
    ->     SELECT COALESCE(SUM(b.num_tickets), 0)
    ->     FROM Booking b
    ->     WHERE b.event_id = e.event_id
    -> ) > (e.total_seats / 2);
Empty set (0.00 sec)
```

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

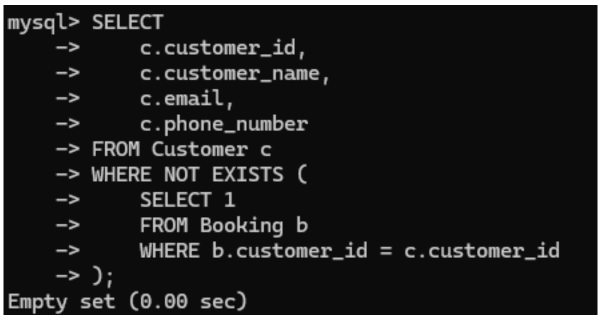
```
SELECT
   e.event_id,
   e.event_name,
   SUM(b.num_tickets) AS total_tickets_sold
FROM Event e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY e.event_id, e.event_name
ORDER BY total_tickets_sold DESC;
```


```
mysql> SELECT
    ->     e.event_id,
    ->     e.event_name,
    ->     SUM(b.num_tickets) AS total_tickets_sold
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY e.event_id, e.event_name
    -> ORDER BY total_tickets_sold DESC;
+----------+-------------------+--------------------+
| event_id | event_name        | total_tickets_sold |
+----------+-------------------+--------------------+
|        3 | Football Final    |                  5 |
|        5 | Shakespeare Play  |                  4 |
|        2 | Coldplay Concert  |                  3 |
|        7 | Art Exhibition    |                  3 |
|        9 | Stand-up Comedy   |                  3 |
|        1 | Avengers Movie    |                  2 |
|        8 | Indie Movie Fest  |                  2 |
|       10 | National Wrestling |                 2 |
|        4 | Jazz Night        |                  1 |
|        6 | EDM Fest          |                  1 |
+----------+-------------------+--------------------+
10 rows in set (0.00 sec)
```

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

```
SELECT
   c.customer_id,
   c.customer_name,
   c.email,
   c.phone_number
FROM Customer c
WHERE NOT EXISTS (
   SELECT 1
   FROM Booking b
```

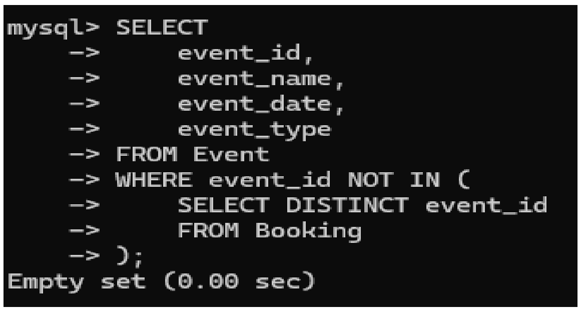
```
    WHERE b.customer_id = c.customer_id
);
```



```
mysql> SELECT
    ->     c.customer_id,
    ->     c.customer_name,
    ->     c.email,
    ->     c.phone_number
    -> FROM Customer c
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM Booking b
    ->     WHERE b.customer_id = c.customer_id
    -> );
Empty set (0.00 sec)
```

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

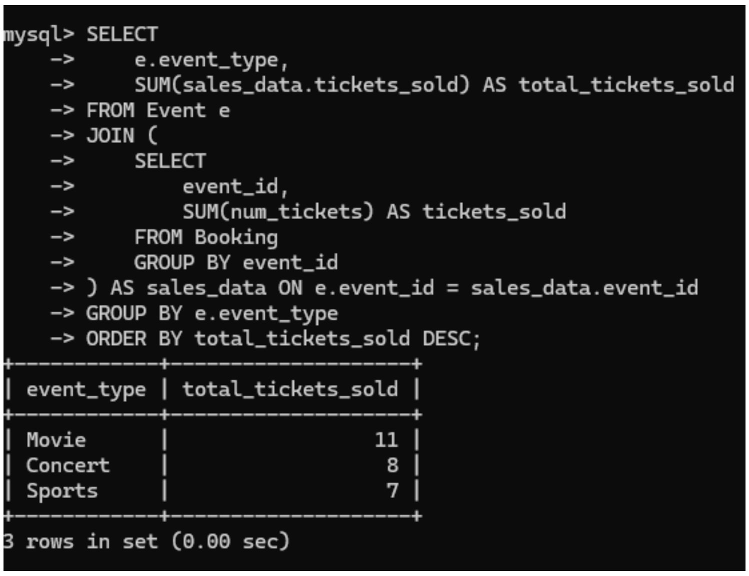
```
SELECT
    event_id,
    event_name,
    event_date,
    event_type
FROM Event
WHERE event_id NOT IN (
    SELECT DISTINCT event_id
    FROM Booking
);
```



```
mysql> SELECT
    ->     event_id,
    ->     event_name,
    ->     event_date,
    ->     event_type
    -> FROM Event
    -> WHERE event_id NOT IN (
    ->     SELECT DISTINCT event_id
    ->     FROM Booking
    -> );
Empty set (0.00 sec)
```

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

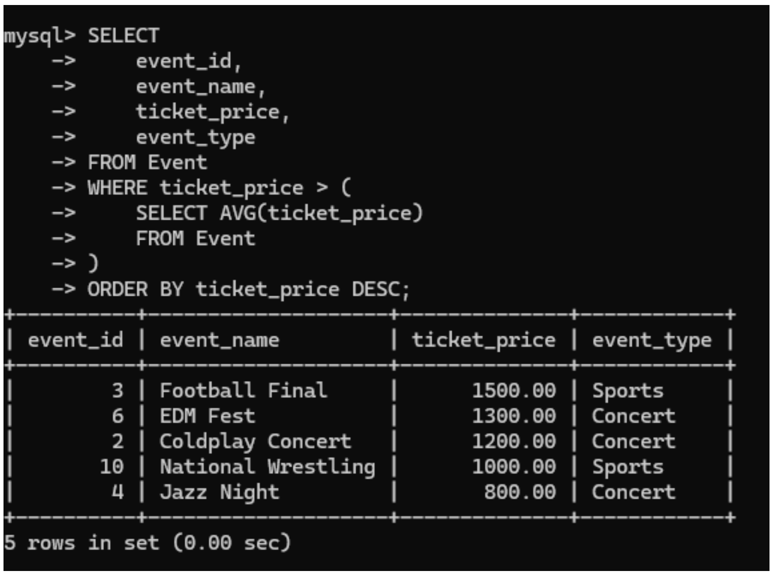
```
SELECT
    e.event_type,
    SUM(sales_data.tickets_sold) AS total_tickets_sold
FROM Event e
JOIN (
    SELECT
        event_id,
        SUM(num_tickets) AS tickets_sold
    FROM Booking
    GROUP BY event_id
) AS sales_data ON e.event_id = sales_data.event_id
GROUP BY e.event_type
ORDER BY total_tickets_sold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



```
mysql> SELECT
    ->     e.event_type,
    ->     SUM(sales_data.tickets_sold) AS total_tickets_sold
    -> FROM Event e
    -> JOIN (
    ->     SELECT
    ->         event_id,
    ->         SUM(num_tickets) AS tickets_sold
    ->     FROM Booking
    ->     GROUP BY event_id
    -> ) AS sales_data ON e.event_id = sales_data.event_id
    -> GROUP BY e.event_type
    -> ORDER BY total_tickets_sold DESC;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Movie      |                 11 |
| Concert    |                  8 |
| Sports     |                  7 |
+------------+--------------------+
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.

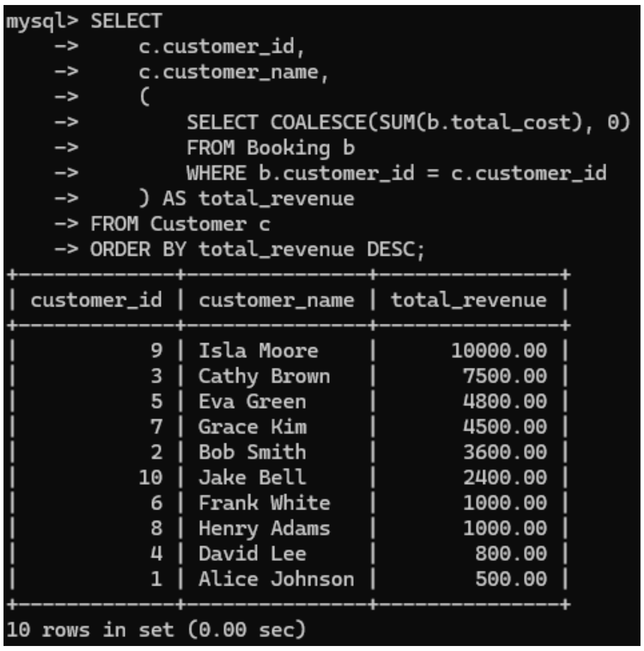
```
SELECT
   event_id,
   event_name,
   ticket_price,
   event_type
FROM Event
WHERE ticket_price > (
   SELECT AVG(ticket_price)
   FROM Event
)
ORDER BY ticket_price DESC;
```



```
mysql> SELECT
    ->     event_id,
    ->     event_name,
    ->     ticket_price,
    ->     event_type
    -> FROM Event
    -> WHERE ticket_price > (
    ->     SELECT AVG(ticket_price)
    ->     FROM Event
    -> )
    -> ORDER BY ticket_price DESC;
+----------+-------------------+--------------+------------+
| event_id | event_name        | ticket_price | event_type |
+----------+-------------------+--------------+------------+
|        3 | Football Final    |      1500.00 | Sports     |
|        6 | EDM Fest          |      1300.00 | Concert    |
|        2 | Coldplay Concert  |      1200.00 | Concert    |
|       10 | National Wrestling |     1000.00 | Sports     |
|        4 | Jazz Night        |       800.00 | Concert    |
+----------+-------------------+--------------+------------+
5 rows in set (0.00 sec)
```

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

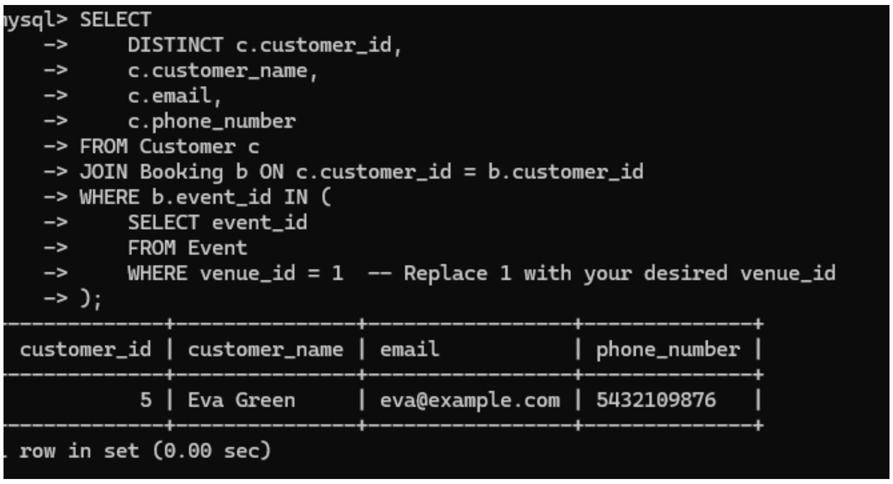
```
SELECT
   c.customer_id,
   c.customer_name,
   (
      SELECT COALESCE(SUM(b.total_cost), 0)
      FROM Booking b
      WHERE b.customer_id = c.customer_id
   ) AS total_revenue
FROM Customer c
ORDER BY total_revenue DESC;
```



```
mysql> SELECT
    ->     c.customer_id,
    ->     c.customer_name,
    ->     (
    ->         SELECT COALESCE(SUM(b.total_cost), 0)
    ->         FROM Booking b
    ->         WHERE b.customer_id = c.customer_id
    ->     ) AS total_revenue
    -> FROM Customer c
    -> ORDER BY total_revenue DESC;
+-------------+---------------+---------------+
| customer_id | customer_name | total_revenue |
+-------------+---------------+---------------+
|           9 | Isla Moore    |      10000.00 |
|           3 | Cathy Brown   |       7500.00 |
|           5 | Eva Green     |       4800.00 |
|           7 | Grace Kim     |       4500.00 |
|           2 | Bob Smith     |       3600.00 |
|          10 | Jake Bell     |       2400.00 |
|           6 | Frank White   |       1000.00 |
|           8 | Henry Adams   |       1000.00 |
|           4 | David Lee     |        800.00 |
|           1 | Alice Johnson |        500.00 |
+-------------+---------------+---------------+
10 rows in set (0.00 sec)
```

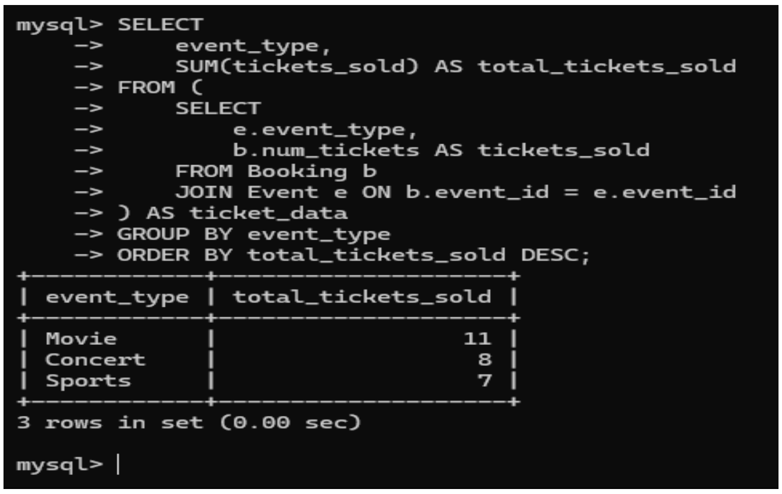
9. List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause.

```
SELECT
   DISTINCT c.customer_id,
   c.customer_name,
   c.email,
   c.phone_number
FROM Customer c
JOIN Booking b ON c.customer_id = b.customer_id
WHERE b.event_id IN (
   SELECT event_id
   FROM Event
   WHERE venue_id = 1  -- Replace 1 with your desired venue_id
);
```


```
ysql> SELECT
   ->     DISTINCT c.customer_id,
   ->     c.customer_name,
   ->     c.email,
   ->     c.phone_number
   -> FROM Customer c
   -> JOIN Booking b ON c.customer_id = b.customer_id
   -> WHERE b.event_id IN (
   ->     SELECT event_id
   ->     FROM Event
   ->     WHERE venue_id = 1  -- Replace 1 with your desired venue_id
   -> );
-------------+---------------+-----------------+--------------+
 customer_id | customer_name | email           | phone_number |
-------------+---------------+-----------------+--------------+
           5 | Eva Green     | eva@example.com | 5432109876   |
-------------+---------------+-----------------+--------------+
 row in set (0.00 sec)
```

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

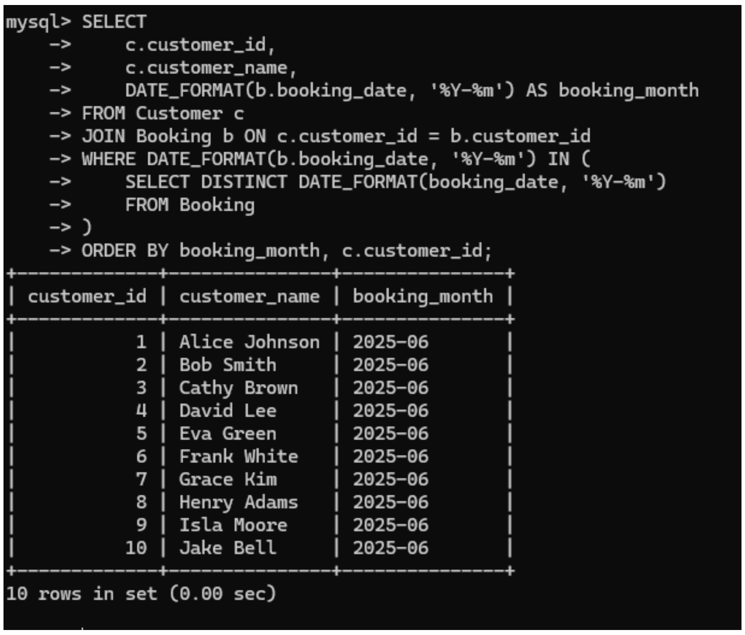
```
SELECT
   event_type,
   SUM(tickets_sold) AS total_tickets_sold
FROM (
   SELECT
      e.event_type,
      b.num_tickets AS tickets_sold
   FROM Booking b
   JOIN Event e ON b.event_id = e.event_id
) AS ticket_data
GROUP BY event_type
ORDER BY total_tickets_sold DESC;
```


```
mysql> SELECT
    ->     event_type,
    ->     SUM(tickets_sold) AS total_tickets_sold
    -> FROM (
    ->     SELECT
    ->         e.event_type,
    ->         b.num_tickets AS tickets_sold
    ->     FROM Booking b
    ->     JOIN Event e ON b.event_id = e.event_id
    -> ) AS ticket_data
    -> GROUP BY event_type
    -> ORDER BY total_tickets_sold DESC;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Movie      |                 11 |
| Concert    |                  8 |
| Sports     |                  7 |
+------------+--------------------+
3 rows in set (0.00 sec)

mysql> |
```

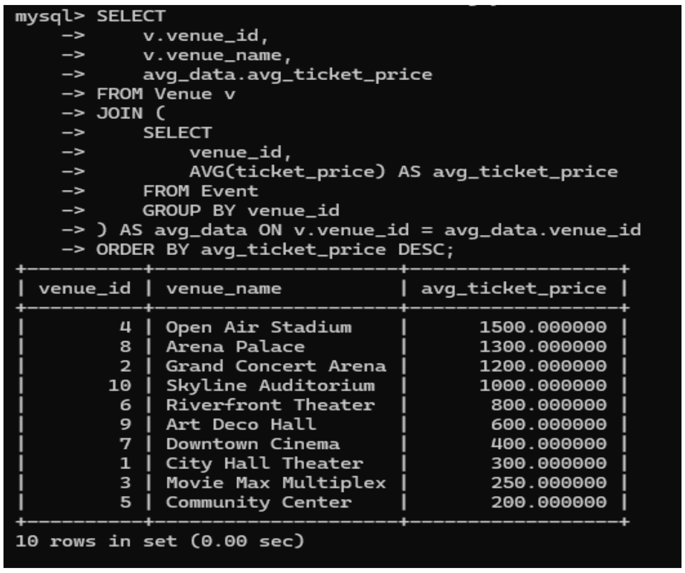
11. Find Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE_FORMAT.

SELECT
c.customer_id,
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, c.customer_id;



```
mysql> SELECT
    ->     c.customer_id,
    ->     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, c.customer_id;
+-------------+---------------+---------------+
| customer_id | customer_name | booking_month |
+-------------+---------------+---------------+
|           1 | Alice Johnson | 2025-06       |
|           2 | Bob Smith     | 2025-06       |
|           3 | Cathy Brown   | 2025-06       |
|           4 | David Lee     | 2025-06       |
|           5 | Eva Green     | 2025-06       |
|           6 | Frank White   | 2025-06       |
|           7 | Grace Kim     | 2025-06       |
|           8 | Henry Adams   | 2025-06       |
|           9 | Isla Moore    | 2025-06       |
|          10 | Jake Bell     | 2025-06       |
+-------------+---------------+---------------+
10 rows in set (0.00 sec)
```

12. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery
SELECT
v.venue_id,
v.venue_name,
avg_data.avg_ticket_price
FROM Venu v
JOIN (
SELECT
venue_id,
AVG(ticket_price) AS avg_ticket_price
FROM Event
GROUP BY venue_id
) AS avg_data ON v.venue_id = avg_data.venue_id
ORDER BY avg_ticket_price DESC;



```
mysql> SELECT
    ->     v.venue_id,
    ->     v.venue_name,
    ->     avg_data.avg_ticket_price
    -> FROM Venue v
    -> JOIN (
    ->     SELECT
    ->         venue_id,
    ->         AVG(ticket_price) AS avg_ticket_price
    ->     FROM Event
    ->     GROUP BY venue_id
    -> ) AS avg_data ON v.venue_id = avg_data.venue_id
    -> ORDER BY avg_ticket_price DESC;
+----------+---------------------+------------------+
| venue_id | venue_name          | avg_ticket_price |
+----------+---------------------+------------------+
|        4 | Open Air Stadium    |      1500.000000 |
|        8 | Arena Palace        |      1300.000000 |
|        2 | Grand Concert Arena |      1200.000000 |
|       10 | Skyline Auditorium  |      1000.000000 |
|        6 | Riverfront Theater  |       800.000000 |
|        9 | Art Deco Hall       |       600.000000 |
|        7 | Downtown Cinema     |       400.000000 |
|        1 | City Hall Theater   |       300.000000 |
|        3 | Movie Max Multiplex |       250.000000 |
|        5 | Community Center    |       200.000000 |
+----------+---------------------+------------------+
10 rows in set (0.00 sec)
```