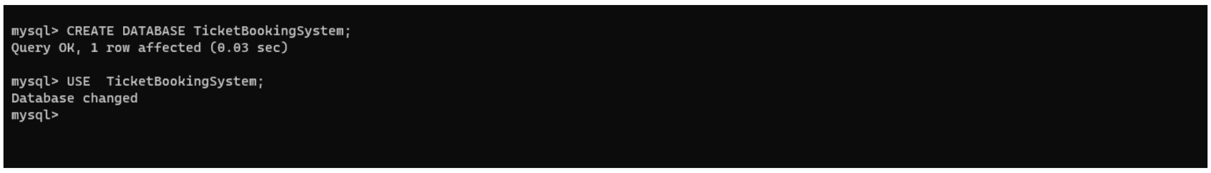
Assignment 5

BY SHREYASI REJA

Ticket Booking System

Tasks 1: Database Design:

1. Create the database named "TicketBookingSystem"
   QUERY:-
   CREATE DATABASE TicketBookingSystem;

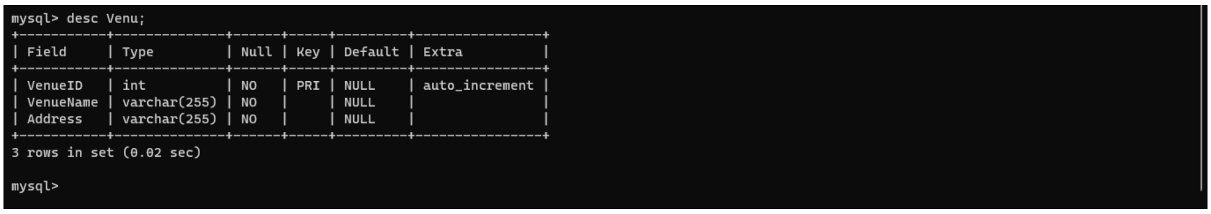


```
mysql> CREATE DATABASE TicketBookingSystem;
Query OK, 1 row affected (0.03 sec)

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

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

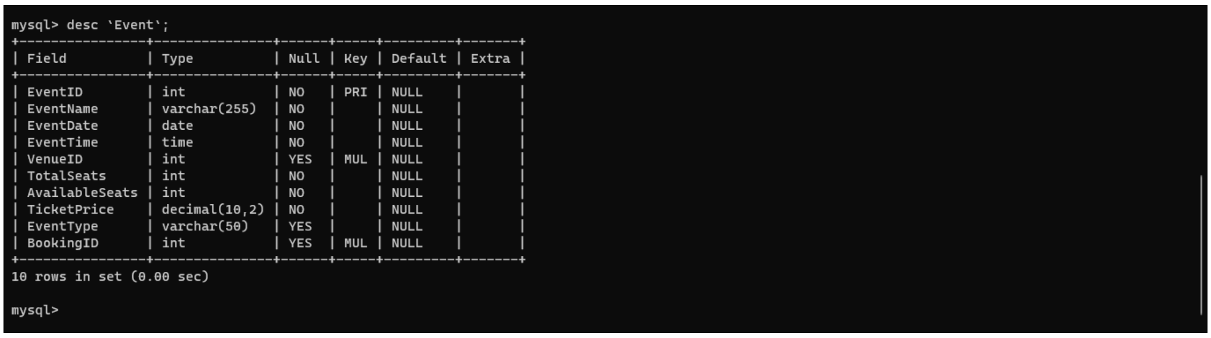
• Venu

```
CREATE TABLE Venu (
    VenueID INT AUTO_INCREMENT PRIMARY KEY,
    VenueName VARCHAR(255) NOT NULL,
   Address VARCHAR(255) NOT NULL
    );
```



```
mysql> desc Venu;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| VenueID   | int          | NO   | PRI | NULL    | auto_increment |
| VenueName | varchar(255) | NO   |     | NULL    |                |
| Address   | varchar(255) | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
3 rows in set (0.02 sec)

mysql>
```

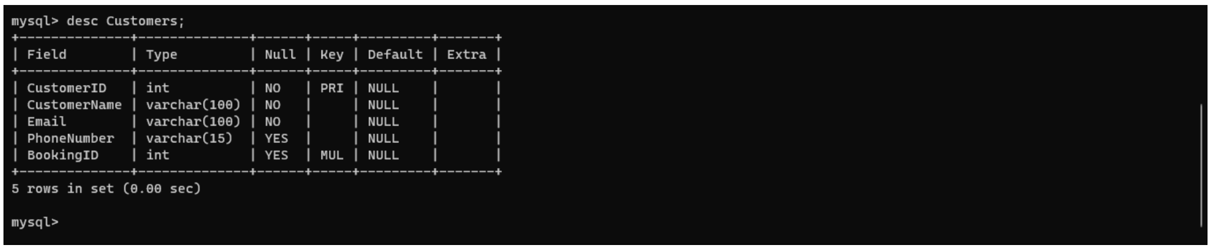
Event :- (AS Event IS A KEYWORD SO I AM NAMING THE TABLE AS `Event`)

```
CREATE TABLE `Event` (
   `EventID` INT PRIMARY KEY,
    `EventName` VARCHAR(255) NOT NULL,
   `EventDate` DATE NOT NULL,
   `EventTime` TIME NOT NULL,
    `VenueID` INT,
   `TotalSeats` INT NOT NULL,
   `AvailableSeats` INT NOT NULL,
   `TicketPrice` DECIMAL(10, 2) NOT NULL,
   `EventType` VARCHAR(50) CHECK (`EventType` IN
('Movie', 'Sports', 'Concert')),
  `BookingID` INT,
   FOREIGN KEY (`VenueID`)  REFERENCES
`Venu`(`VenueID`)
    ),
FOREIGN KEY (`BookingID`) REFERENCES
`Booking`(`BookingID`)
    );
```


```
mysql> desc `Event`;
+----------------+---------------+------+-----+---------+-------+
| Field          | Type          | Null | Key | Default | Extra |
+----------------+---------------+------+-----+---------+-------+
| EventID        | int           | NO   | PRI | NULL    |       |
| EventName      | varchar(255)  | NO   |     | NULL    |       |
| EventDate      | date          | NO   |     | NULL    |       |
| EventTime      | time          | NO   |     | NULL    |       |
| VenueID        | int           | YES  | MUL | NULL    |       |
| TotalSeats     | int           | NO   |     | NULL    |       |
| AvailableSeats | int           | NO   |     | NULL    |       |
| TicketPrice    | decimal(10,2) | NO   |     | NULL    |       |
| EventType      | varchar(50)   | YES  |     | NULL    |       |
| BookingID      | int           | YES  | MUL | NULL    |       |
+----------------+---------------+------+-----+---------+-------+
10 rows in set (0.00 sec)

mysql>
```

```
CREATE TABLE Customers (
    CustomerID INT  PRIMARY KEY,
   CustomerName VARCHAR(100) NOT NULL,
     Email VARCHAR(100) NOT NULL,
   PhoneNumber VARCHAR(15)
    BookingID INT,
   FOREIGN KEY (BookingID) REFERENCES
Booking(BookingID)
```

    );

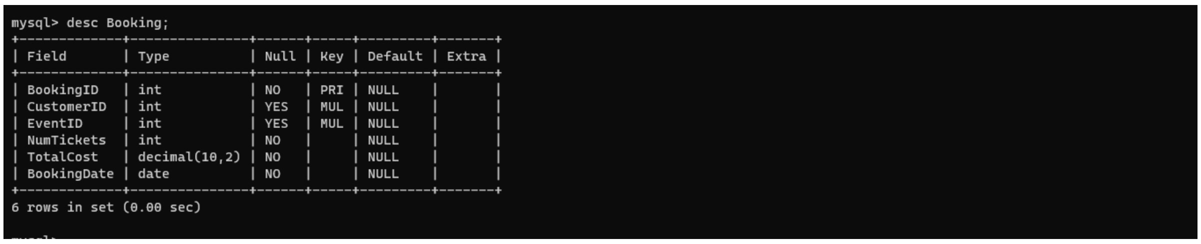


```
mysql> desc Customers;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| CustomerID   | int          | NO   | PRI | NULL    |       |
| CustomerName | varchar(100) | NO   |     | NULL    |       |
| Email        | varchar(100) | NO   |     | NULL    |       |
| PhoneNumber  | varchar(15)  | YES  |     | NULL    |       |
| BookingID    | int          | YES  | MUL | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql>
```

Booking :-

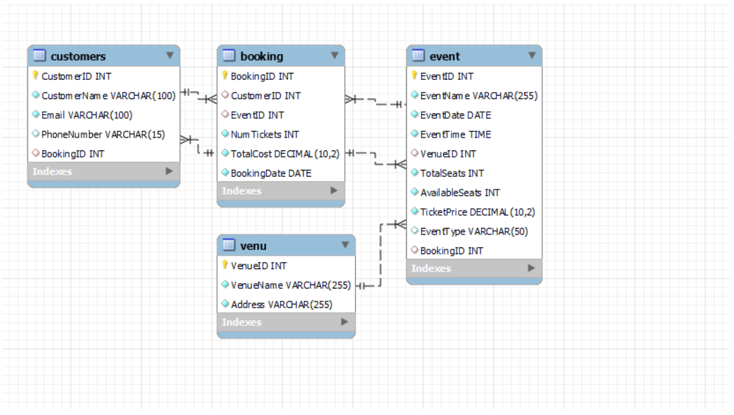
CREATE TABLE Booking (

    BookingID INT PRIMARY KEY,
    CustomerID INT,
    EventID INT,
    NumTickets INT NOT NULL,
    TotalCost DECIMAL(10, 2) NOT NULL,
    BookingDate DATE NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES
Customers(CustomerID),
    FOREIGN KEY (`EventID`) REFERENCES `Event`(`EventID`)
    );



```
mysql> desc Booking;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| BookingID   | int           | NO   | PRI | NULL    |       |
| CustomerID  | int           | YES  | MUL | NULL    |       |
| EventID     | int           | YES  | MUL | NULL    |       |
| NumTickets  | int           | NO   |     | NULL    |       |
| TotalCost   | decimal(10,2) | NO   |     | NULL    |       |
| BookingDate | date          | NO   |     | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

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

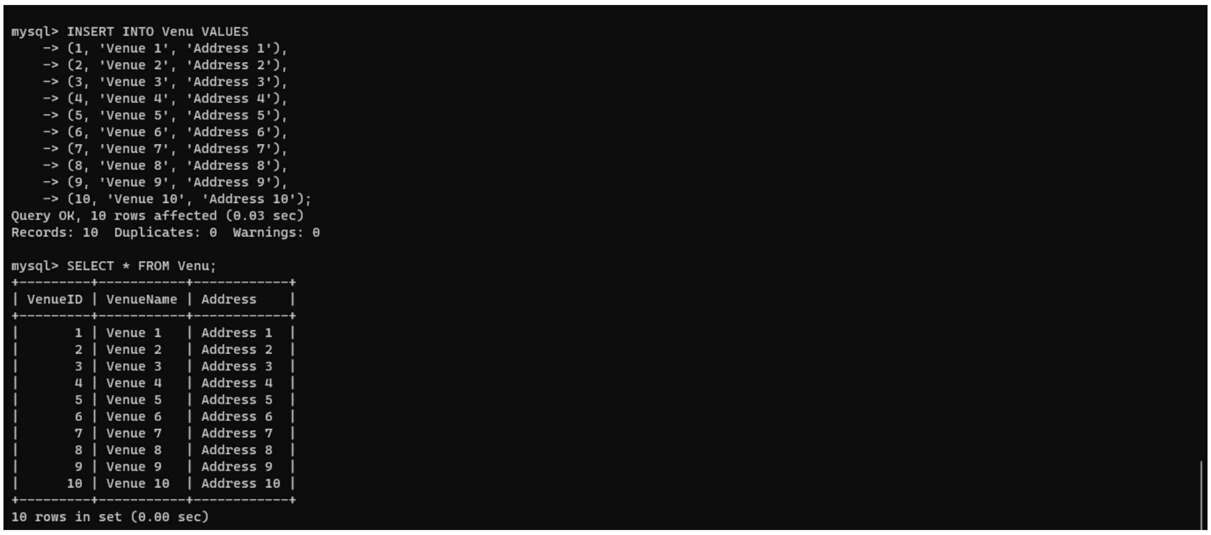


Tasks 2: Select, Where, Between, AND, LIKE:

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

VENU DATA :-

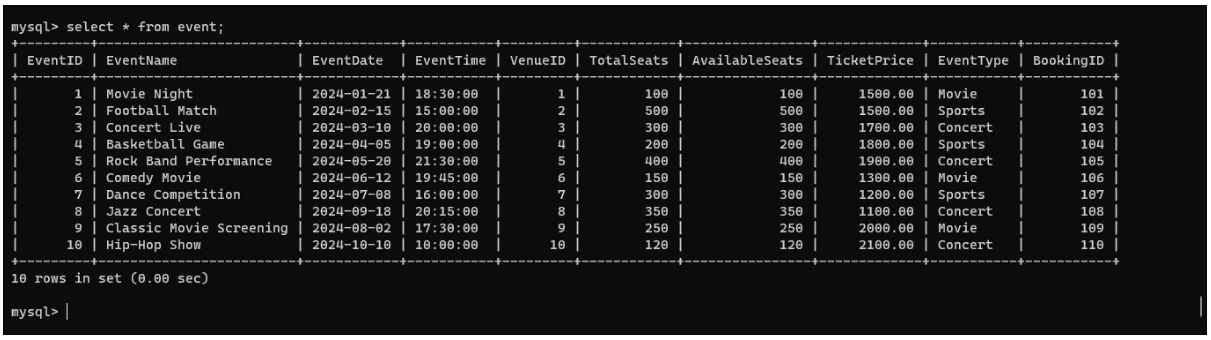
```
INSERT INTO Venu VALUES
    (1, 'Venue 1', 'Address 1'),
   (2, 'Venue 2', 'Address 2'),
    (3, 'Venue 3', 'Address 3'),
   (4, 'Venue 4', 'Address 4'),
    (5, 'Venue 5', 'Address 5'),
    (6, 'Venue 6', 'Address 6'),
     (7, 'Venue 7', 'Address 7'),
    (8, 'Venue 8', 'Address 8'),
    (9, 'Venue 9', 'Address 9'),
    (10, 'Venue 10', 'Address 10');
```



```
mysql> INSERT INTO Venu VALUES
    -> (1, 'Venue 1', 'Address 1'),
    -> (2, 'Venue 2', 'Address 2'),
    -> (3, 'Venue 3', 'Address 3'),
    -> (4, 'Venue 4', 'Address 4'),
    -> (5, 'Venue 5', 'Address 5'),
    -> (6, 'Venue 6', 'Address 6'),
    -> (7, 'Venue 7', 'Address 7'),
    -> (8, 'Venue 8', 'Address 8'),
    -> (9, 'Venue 9', 'Address 9'),
    -> (10, 'Venue 10', 'Address 10');
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Venu;
+---------+-----------+------------+
| VenueID | VenueName | Address    |
+---------+-----------+------------+
|       1 | Venue 1   | Address 1  |
|       2 | Venue 2   | Address 2  |
|       3 | Venue 3   | Address 3  |
|       4 | Venue 4   | Address 4  |
|       5 | Venue 5   | Address 5  |
|       6 | Venue 6   | Address 6  |
|       7 | Venue 7   | Address 7  |
|       8 | Venue 8   | Address 8  |
|       9 | Venue 9   | Address 9  |
|      10 | Venue 10  | Address 10 |
+---------+-----------+------------+
10 rows in set (0.00 sec)
```

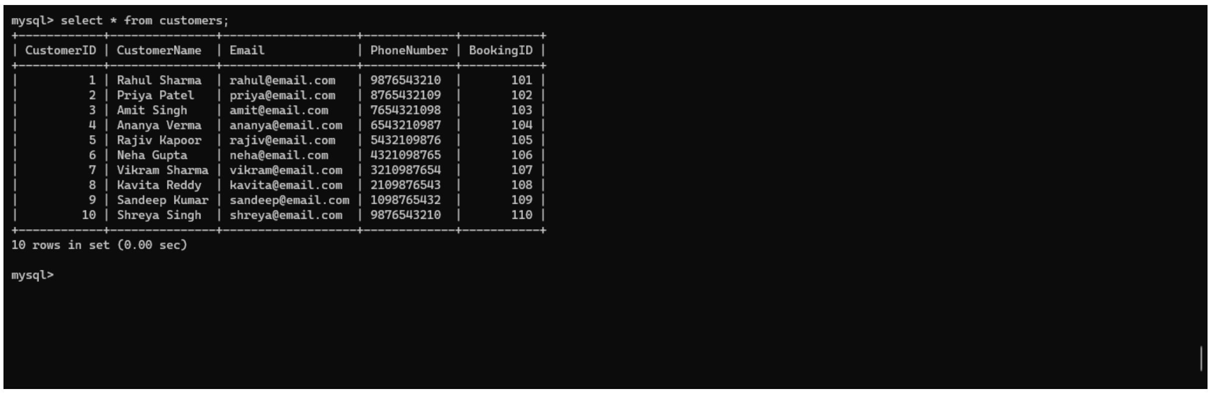
`Event` Data:-

```
INSERT INTO `Event` (`EventID`,`EventName`, `EventDate`,
`EventTime`,`TotalSeats`, `AvailableSeats`, `TicketPrice`,
`EventType`) VALUES
   ->  (1,'Movie Night', '2024-01-21', '18:30:00',100, 100, 1500.00,
'Movie'),
    ->  (2,'Football Match', '2024-02-15', '15:00:00',500, 500,
1500.00, 'Sports'),
    ->  (3,'Concert Live', '2024-03-10', '20:00:00', 300, 300, 1700.00,
'Concert'),
    -> (4,'Basketball Game', '2024-04-05', '19:00:00',200, 200,
1800.00, 'Sports'),
    -> (5,'Rock Band Performance', '2024-05-20', '21:30:00', 400,
400, 1900.00, 'Concert'),
    -> (6,'Comedy Movie', '2024-06-12', '19:45:00',150, 150,
1300.00, 'Movie'),
    ->  (7,'Dance Competition', '2024-07-08', '16:00:00',300, 300,
1200.00, 'Sports'),
    ->  (8,'Jazz Concert', '2024-09-18', '20:15:00',350, 350, 1100.00,
'Concert'),
    ->  (9,'Classic Movie Screening', '2024-08-02', '17:30:00',250,
250, 2000.00, 'Movie'),
    -> (10,'Hip-Hop Show', '2024-10-10', '10:00:00',120, 120,
2100.00, 'Concert'); update `Event` set  BookingID = 101 where
EventID = 1;
update `Event` set  BookingID = 102 where  EventID = 2;
update `Event` set  BookingID = 103 where  EventID = 3;
update `Event` set  BookingID = 104 where  EventID = 4;
update `Event` set  BookingID = 105 where  EventID = 5;
update `Event` set  BookingID = 106 where  EventID = 6;
update `Event` set  BookingID = 107 where  EventID = 7;
update `Event` set  BookingID = 108 where  EventID = 8;
update `Event` set  BookingID = 109 where  EventID = 9;
update `Event` set  BookingID = 110 where  EventID = 10;
update `Event` set  VenueID = 1 where  EventID = 1;
update `Event` set  VenueID = 2 where  EventID = 2;
update `Event` set  VenueID = 3 where  EventID = 3;
update `Event` set  VenueID = 4 where  EventID = 4;
update `Event` set  VenueID = 5 where  EventID = 5;
update `Event` set  VenueID = 6 where  EventID = 6;
update `Event` set  VenueID = 7 where  EventID = 7;
update `Event` set  VenueID = 8 where  EventID = 8;
update `Event` set  VenueID = 9 where  EventID = 9;
update `Event` set  VenueID = 10 where  EventID = 10;
```



```
mysql> select * from event;
+---------+------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
| EventID | EventName              | EventDate  | EventTime | VenueID | TotalSeats | AvailableSeats | TicketPrice | EventType | BookingID |
+---------+------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
|       1 | Movie Night            | 2024-01-21 | 18:30:00  |       1 |        100 |            100 |     1500.00 | Movie     |       101 |
|       2 | Football Match         | 2024-02-15 | 15:00:00  |       2 |        500 |            500 |     1500.00 | Sports    |       102 |
|       3 | Concert Live           | 2024-03-10 | 20:00:00  |       3 |        300 |            300 |     1700.00 | Concert   |       103 |
|       4 | Basketball Game        | 2024-04-05 | 19:00:00  |       4 |        200 |            200 |     1800.00 | Sports    |       104 |
|       5 | Rock Band Performance  | 2024-05-20 | 21:30:00  |       5 |        400 |            400 |     1900.00 | Concert   |       105 |
|       6 | Comedy Movie           | 2024-06-12 | 19:45:00  |       6 |        150 |            150 |     1300.00 | Movie     |       106 |
|       7 | Dance Competition      | 2024-07-08 | 16:00:00  |       7 |        300 |            300 |     1200.00 | Sports    |       107 |
|       8 | Jazz Concert           | 2024-09-18 | 20:15:00  |       8 |        350 |            350 |     1100.00 | Concert   |       108 |
|       9 | Classic Movie Screening | 2024-08-02 | 17:30:00  |       9 |        250 |            250 |     2000.00 | Movie     |       109 |
|      10 | Hip-Hop Show           | 2024-10-10 | 10:00:00  |      10 |        120 |            120 |     2100.00 | Concert   |       110 |
+---------+------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
10 rows in set (0.00 sec)

mysql> |
```

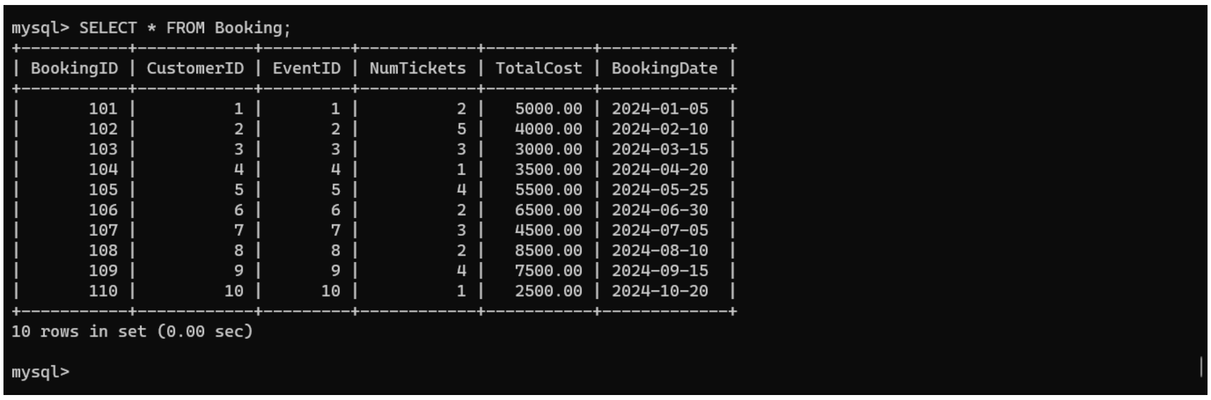
```
INSERT INTO Customers (CustomerID, CustomerName, Email, PhoneNumber) VALUES
    -> (1, 'Rahul Sharma', 'rahul@email.com', '9876543210'),
    -> (2, 'Priya Patel', 'priya@email.com', '8765432109'),
    -> (3, 'Amit Singh', 'amit@email.com', '7654321098'),
    -> (4, 'Ananya Verma', 'ananya@email.com', '6543210987'),
    -> (5, 'Rajiv Kapoor', 'rajiv@email.com', '5432109876'),
    -> (6, 'Neha Gupta', 'neha@email.com', '4321098765'),
    -> (7, 'Vikram Sharma', 'vikram@email.com', '3210987654'),
    -> (8, 'Kavita Reddy', 'kavita@email.com', '2109876543'),
    -> (9, 'Sandeep Kumar', 'sandeep@email.com', '1098765432'),
    -> (10, 'Shreya Singh', 'shreya@email.com', '9876543210');
update customers set  BookingID=101 where CustomerID=1;
 update customers set  BookingID=102 where CustomerID=2;
  update customers set  BookingID=103 where CustomerID=3;
update customers set  BookingID=104 where CustomerID=4;
update customers set  BookingID=105 where CustomerID=5;
update customers set  BookingID=106 where CustomerID=6;
update customers set  BookingID=107 where CustomerID=7;
update customers set  BookingID=108 where CustomerID=8;
update customers set  BookingID=109 where CustomerID=9;
update customers set  BookingID=110 where CustomerID=10;
```



```
mysql> select * from customers;
+------------+--------------+-------------------+-------------+-----------+
| CustomerID | CustomerName | Email             | PhoneNumber | BookingID |
+------------+--------------+-------------------+-------------+-----------+
|          1 | Rahul Sharma | rahul@email.com   | 9876543210  |       101 |
|          2 | Priya Patel  | priya@email.com   | 8765432109  |       102 |
|          3 | Amit Singh   | amit@email.com    | 7654321098  |       103 |
|          4 | Ananya Verma | ananya@email.com  | 6543210987  |       104 |
|          5 | Rajiv Kapoor | rajiv@email.com   | 5432109876  |       105 |
|          6 | Neha Gupta   | neha@email.com    | 4321098765  |       106 |
|          7 | Vikram Sharma | vikram@email.com  | 3210987654  |       107 |
|          8 | Kavita Reddy | kavita@email.com  | 2109876543  |       108 |
|          9 | Sandeep Kumar | sandeep@email.com | 1098765432  |       109 |
|         10 | Shreya Singh | shreya@email.com  | 9876543210  |       110 |
+------------+--------------+-------------------+-------------+-----------+
10 rows in set (0.00 sec)

mysql>
```

Booking DATA:-

```
INSERT INTO Booking (BookingID, CustomerID, EventID,
NumTickets, TotalCost, BookingDate) VALUES
    -> (101, 1, 1, 2, 5000.00, '2024-01-05'),
    -> (102, 2, 2, 5, 4000.00, '2024-02-10'),
    -> (103, 3, 3, 3, 3000.00, '2024-03-15'),
    -> (104, 4, 4, 1, 3500.00, '2024-04-20'),
    -> (105, 5, 5, 4, 5500.00, '2024-05-25'),
    -> (106, 6, 6, 2, 6500.00, '2024-06-30'),
    -> (107, 7, 7, 3, 4500.00, '2024-07-05'),
    -> (108, 8, 8, 2, 8500.00, '2024-08-10'),
    -> (109, 9, 9, 4, 7500.00, '2024-09-15'),
    -> (110, 10, 10, 1, 2500.00, '2024-10-20');
```

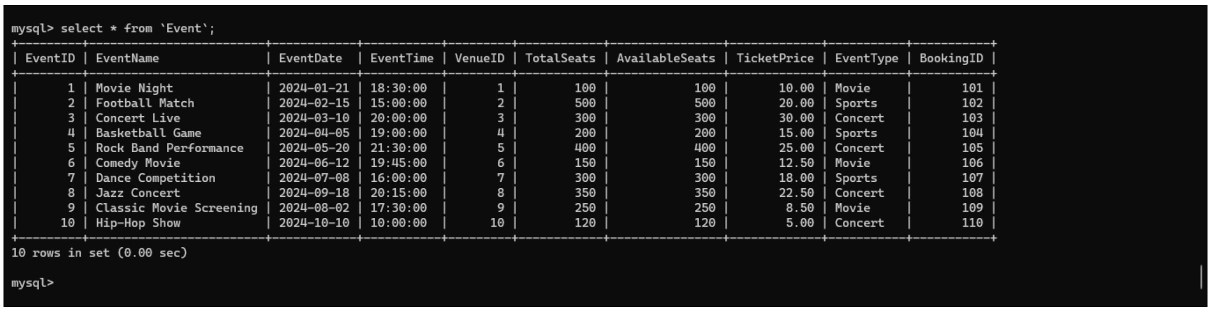


```
mysql> SELECT * FROM Booking;
+-----------+------------+---------+------------+-----------+-------------+
| BookingID | CustomerID | EventID | NumTickets | TotalCost | BookingDate |
+-----------+------------+---------+------------+-----------+-------------+
|       101 |          1 |       1 |          2 |   5000.00 | 2024-01-05  |
|       102 |          2 |       2 |          5 |   4000.00 | 2024-02-10  |
|       103 |          3 |       3 |          3 |   3000.00 | 2024-03-15  |
|       104 |          4 |       4 |          1 |   3500.00 | 2024-04-20  |
|       105 |          5 |       5 |          4 |   5500.00 | 2024-05-25  |
|       106 |          6 |       6 |          2 |   6500.00 | 2024-06-30  |
|       107 |          7 |       7 |          3 |   4500.00 | 2024-07-05  |
|       108 |          8 |       8 |          2 |   8500.00 | 2024-08-10  |
|       109 |          9 |       9 |          4 |   7500.00 | 2024-09-15  |
|       110 |         10 |      10 |          1 |   2500.00 | 2024-10-20  |
+-----------+------------+---------+------------+-----------+-------------+
10 rows in set (0.00 sec)

mysql>
```

2. Write a SQL query to list all Events.

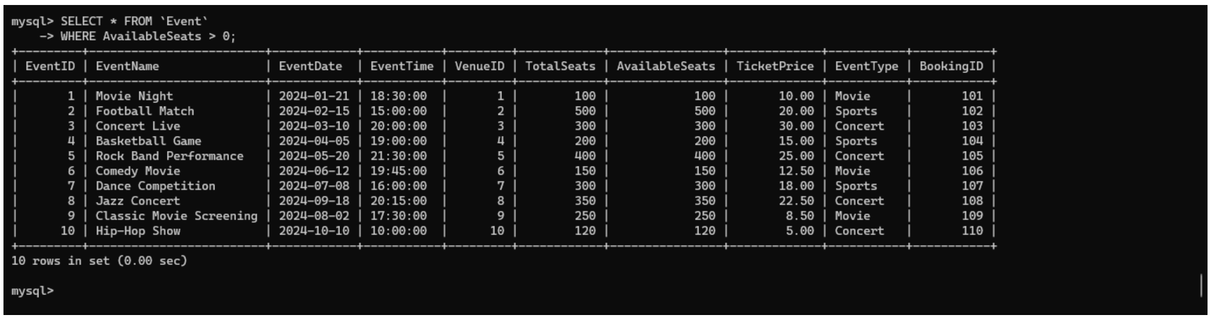
   QUERY:-

   select * from `Event`;



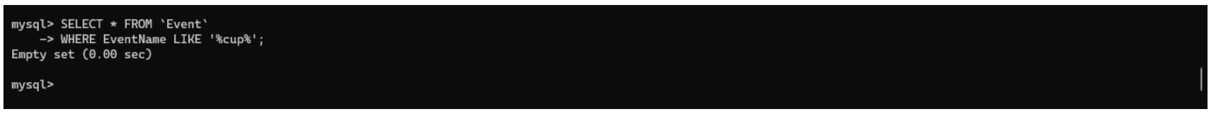
```
mysql> select * from `Event`;
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
| EventID | EventName               | EventDate  | EventTime | VenueID | TotalSeats | AvailableSeats | TicketPrice | EventType | BookingID |
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
|       1 | Movie Night             | 2024-01-21 | 18:30:00  |       1 |        100 |            100 |       10.00 | Movie     |       101 |
|       2 | Football Match          | 2024-02-15 | 15:00:00  |       2 |        500 |            500 |       20.00 | Sports    |       102 |
|       3 | Concert Live            | 2024-03-10 | 20:00:00  |       3 |        300 |            300 |       30.00 | Concert   |       103 |
|       4 | Basketball Game         | 2024-04-05 | 19:00:00  |       4 |        200 |            200 |       15.00 | Sports    |       104 |
|       5 | Rock Band Performance   | 2024-05-20 | 21:30:00  |       5 |        400 |            400 |       25.00 | Concert   |       105 |
|       6 | Comedy Movie            | 2024-06-12 | 19:45:00  |       6 |        150 |            150 |       12.50 | Movie     |       106 |
|       7 | Dance Competition       | 2024-07-08 | 16:00:00  |       7 |        300 |            300 |       18.00 | Sports    |       107 |
|       8 | Jazz Concert            | 2024-09-18 | 20:15:00  |       8 |        350 |            350 |       22.50 | Concert   |       108 |
|       9 | Classic Movie Screening | 2024-08-02 | 17:30:00  |       9 |        250 |            250 |        8.50 | Movie     |       109 |
|      10 | Hip-Hop Show            | 2024-10-10 | 10:00:00  |      10 |        120 |            120 |        5.00 | Concert   |       110 |
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
10 rows in set (0.00 sec)

mysql>
```

3. Write a SQL query to select events with available tickets.
   QUERY:-
   SELECT * FROM `Event`
   -> WHERE AvailableSeats > 0;



```
mysql> SELECT * FROM `Event`
    -> WHERE AvailableSeats > 0;
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
| EventID | EventName               | EventDate  | EventTime | VenueID | TotalSeats | AvailableSeats | TicketPrice | EventType | BookingID |
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
|       1 | Movie Night             | 2024-01-21 | 18:30:00  |       1 |        100 |            100 |       10.00 | Movie     |       101 |
|       2 | Football Match          | 2024-02-15 | 15:00:00  |       2 |        500 |            500 |       20.00 | Sports    |       102 |
|       3 | Concert Live            | 2024-03-10 | 20:00:00  |       3 |        300 |            300 |       30.00 | Concert   |       103 |
|       4 | Basketball Game         | 2024-04-05 | 19:00:00  |       4 |        200 |            200 |       15.00 | Sports    |       104 |
|       5 | Rock Band Performance   | 2024-05-20 | 21:30:00  |       5 |        400 |            400 |       25.00 | Concert   |       105 |
|       6 | Comedy Movie            | 2024-06-12 | 19:45:00  |       6 |        150 |            150 |       12.50 | Movie     |       106 |
|       7 | Dance Competition       | 2024-07-08 | 16:00:00  |       7 |        300 |            300 |       18.00 | Sports    |       107 |
|       8 | Jazz Concert            | 2024-09-18 | 20:15:00  |       8 |        350 |            350 |       22.50 | Concert   |       108 |
|       9 | Classic Movie Screening | 2024-08-02 | 17:30:00  |       9 |        250 |            250 |        8.50 | Movie     |       109 |
|      10 | Hip-Hop Show            | 2024-10-10 | 10:00:00  |      10 |        120 |            120 |        5.00 | Concert   |       110 |
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
10 rows in set (0.00 sec)

mysql>
```

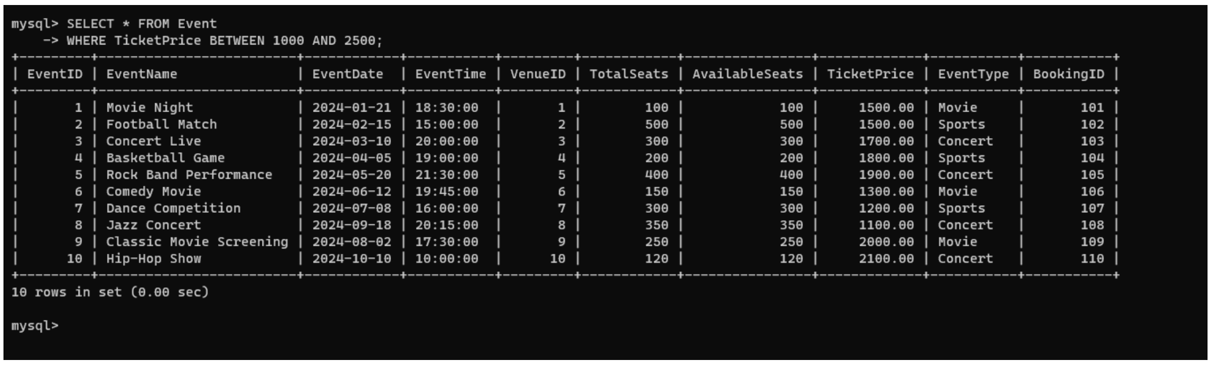
4. Write a SQL query to select events name partial match with ‘cup’.
   QUERY:-
   SELECT * FROM `Event`
   -> WHERE EventName LIKE '%cup%';



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

mysql>
```

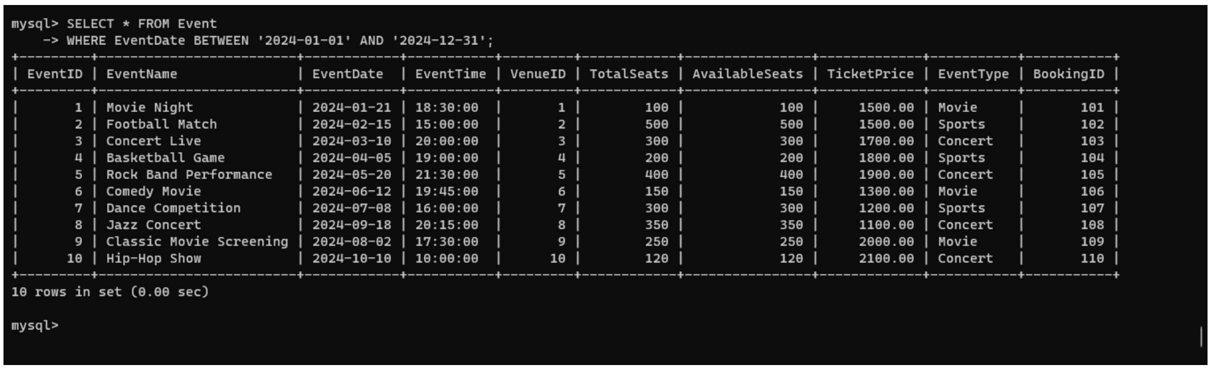
5. Write a SQL query to select events with ticket price range is between 1000 to 2500.
   QUERY:-

   SELECT * FROM Event
   -> WHERE TicketPrice BETWEEN 1000 AND 2500;



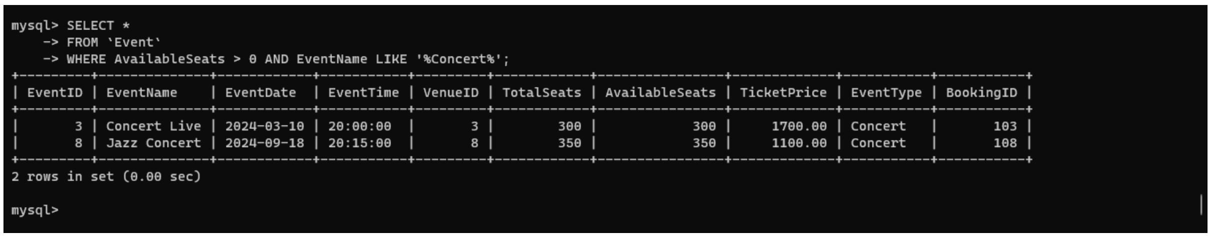
```
mysql> SELECT * FROM Event
    -> WHERE TicketPrice BETWEEN 1000 AND 2500;
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
| EventID | EventName               | EventDate  | EventTime | VenueID | TotalSeats | AvailableSeats | TicketPrice | EventType | BookingID |
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
|       1 | Movie Night             | 2024-01-21 | 18:30:00  |       1 |        100 |            100 |     1500.00 | Movie     |       101 |
|       2 | Football Match          | 2024-02-15 | 15:00:00  |       2 |        500 |            500 |     1500.00 | Sports    |       102 |
|       3 | Concert Live            | 2024-03-10 | 20:00:00  |       3 |        300 |            300 |     1700.00 | Concert   |       103 |
|       4 | Basketball Game         | 2024-04-05 | 19:00:00  |       4 |        200 |            200 |     1800.00 | Sports    |       104 |
|       5 | Rock Band Performance   | 2024-05-20 | 21:30:00  |       5 |        400 |            400 |     1900.00 | Concert   |       105 |
|       6 | Comedy Movie            | 2024-06-12 | 19:45:00  |       6 |        150 |            150 |     1300.00 | Movie     |       106 |
|       7 | Dance Competition       | 2024-07-08 | 16:00:00  |       7 |        300 |            300 |     1200.00 | Sports    |       107 |
|       8 | Jazz Concert            | 2024-09-18 | 20:15:00  |       8 |        350 |            350 |     1100.00 | Concert   |       108 |
|       9 | Classic Movie Screening | 2024-08-02 | 17:30:00  |       9 |        250 |            250 |     2000.00 | Movie     |       109 |
|      10 | Hip-Hop Show            | 2024-10-10 | 10:00:00  |      10 |        120 |            120 |     2100.00 | Concert   |       110 |
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
10 rows in set (0.00 sec)

mysql>
```

6. Write a SQL query to retrieve events with dates falling within a specific range.
   QUERY:-
   SELECT * FROM Event
      -> WHERE EventDate BETWEEN '2024-01-01' AND '2024-12-31';



```
mysql> SELECT * FROM Event
    -> WHERE EventDate BETWEEN '2024-01-01' AND '2024-12-31';
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
| EventID | EventName               | EventDate  | EventTime | VenueID | TotalSeats | AvailableSeats | TicketPrice | EventType | BookingID |
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
|       1 | Movie Night             | 2024-01-21 | 18:30:00  |       1 |        100 |            100 |     1500.00 | Movie     |       101 |
|       2 | Football Match          | 2024-02-15 | 15:00:00  |       2 |        500 |            500 |     1500.00 | Sports    |       102 |
|       3 | Concert Live            | 2024-03-10 | 20:00:00  |       3 |        300 |            300 |     1700.00 | Concert   |       103 |
|       4 | Basketball Game         | 2024-04-05 | 19:00:00  |       4 |        200 |            200 |     1800.00 | Sports    |       104 |
|       5 | Rock Band Performance   | 2024-05-20 | 21:30:00  |       5 |        400 |            400 |     1900.00 | Concert   |       105 |
|       6 | Comedy Movie            | 2024-06-12 | 19:45:00  |       6 |        150 |            150 |     1300.00 | Movie     |       106 |
|       7 | Dance Competition       | 2024-07-08 | 16:00:00  |       7 |        300 |            300 |     1200.00 | Sports    |       107 |
|       8 | Jazz Concert            | 2024-09-18 | 20:15:00  |       8 |        350 |            350 |     1100.00 | Concert   |       108 |
|       9 | Classic Movie Screening | 2024-08-02 | 17:30:00  |       9 |        250 |            250 |     2000.00 | Movie     |       109 |
|      10 | Hip-Hop Show            | 2024-10-10 | 10:00:00  |      10 |        120 |            120 |     2100.00 | Concert   |       110 |
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
10 rows in set (0.00 sec)

mysql>
```

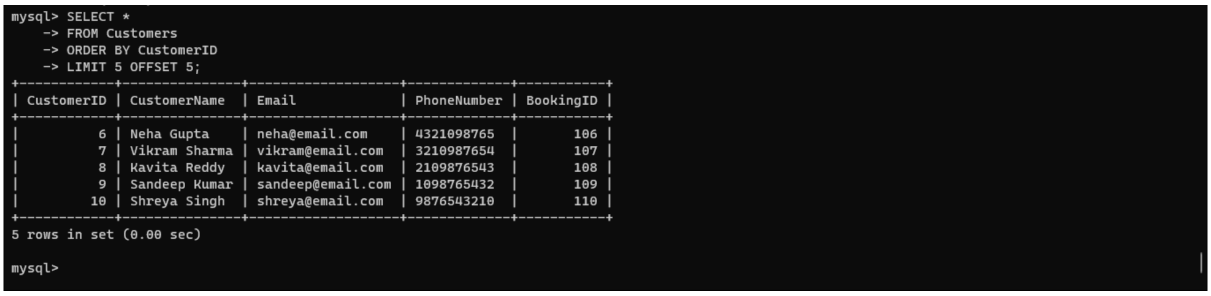
7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.
   QUERY:-
   SELECT *
      -> FROM `Event`
      -> WHERE AvailableSeats > 0 AND EventName LIKE
   '%Concert%';



```
mysql> SELECT *
    -> FROM `Event`
    -> WHERE AvailableSeats > 0 AND EventName LIKE '%Concert%';
+---------+--------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
| EventID | EventName    | EventDate  | EventTime | VenueID | TotalSeats | AvailableSeats | TicketPrice | EventType | BookingID |
+---------+--------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
|       3 | Concert Live | 2024-03-10 | 20:00:00  |       3 |        300 |            300 |     1700.00 | Concert   |       103 |
|       8 | Jazz Concert | 2024-09-18 | 20:15:00  |       8 |        350 |            350 |     1100.00 | Concert   |       108 |
+---------+--------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
2 rows in set (0.00 sec)

mysql>
```

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

-> FROM Customers
-> ORDER BY CustomerID
-> LIMIT 5 OFFSET 5;

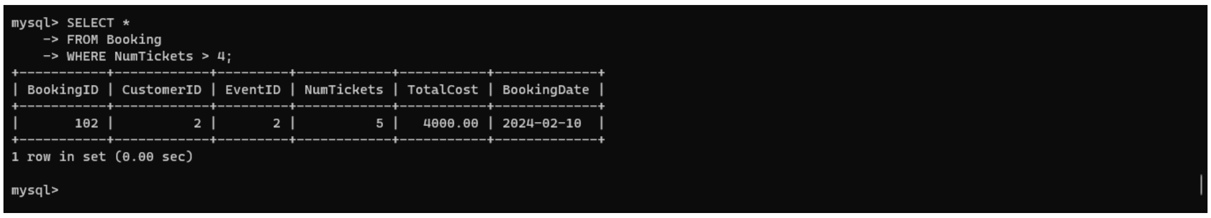


```
mysql> SELECT *
    -> FROM Customers
    -> ORDER BY CustomerID
    -> LIMIT 5 OFFSET 5;
+------------+---------------+-------------------+-------------+-----------+
| CustomerID | CustomerName  | Email             | PhoneNumber | BookingID |
+------------+---------------+-------------------+-------------+-----------+
|          6 | Neha Gupta    | neha@email.com    | 4321098765  |       106 |
|          7 | Vikram Sharma | vikram@email.com  | 3210987654  |       107 |
|          8 | Kavita Reddy  | kavita@email.com  | 2109876543  |       108 |
|          9 | Sandeep Kumar | sandeep@email.com | 1098765432  |       109 |
|         10 | Shreya Singh  | shreya@email.com  | 9876543210  |       110 |
+------------+---------------+-------------------+-------------+-----------+
5 rows in set (0.00 sec)

mysql>
```

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

QUERY:-

SELECT *
-> FROM Booking
-> WHERE NumTickets > 4;

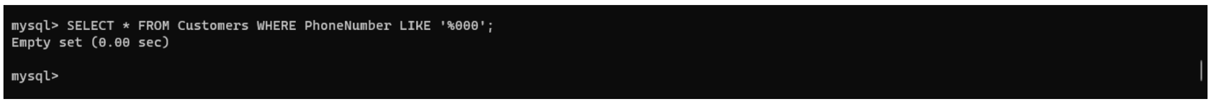


```
mysql> SELECT *
    -> FROM Booking
    -> WHERE NumTickets > 4;
+-----------+------------+---------+------------+-----------+-------------+
| BookingID | CustomerID | EventID | NumTickets | TotalCost | BookingDate |
+-----------+------------+---------+------------+-----------+-------------+
|       102 |          2 |       2 |          5 |   4000.00 | 2024-02-10  |
+-----------+------------+---------+------------+-----------+-------------+
1 row in set (0.00 sec)

mysql>
```

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

QUERY:-

SELECT * FROM Customers WHERE PhoneNumber LIKE '%000';

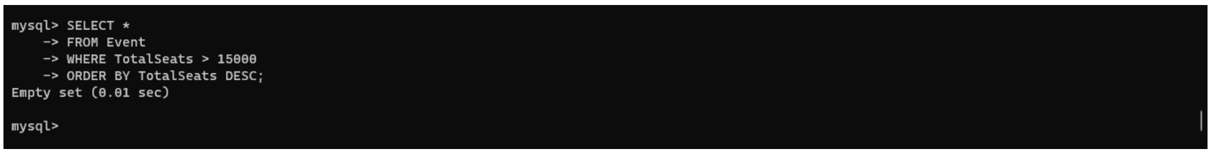


```
mysql> SELECT * FROM Customers WHERE PhoneNumber LIKE '%000';
Empty set (0.00 sec)

mysql>
```

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

QUERY:-

SELECT *
-> FROM Event
-> WHERE TotalSeats > 15000
-> ORDER BY TotalSeats DESC;



```
mysql> SELECT *
    -> FROM Event
    -> WHERE TotalSeats > 15000
    -> ORDER BY TotalSeats DESC;
Empty set (0.01 sec)

mysql>
```

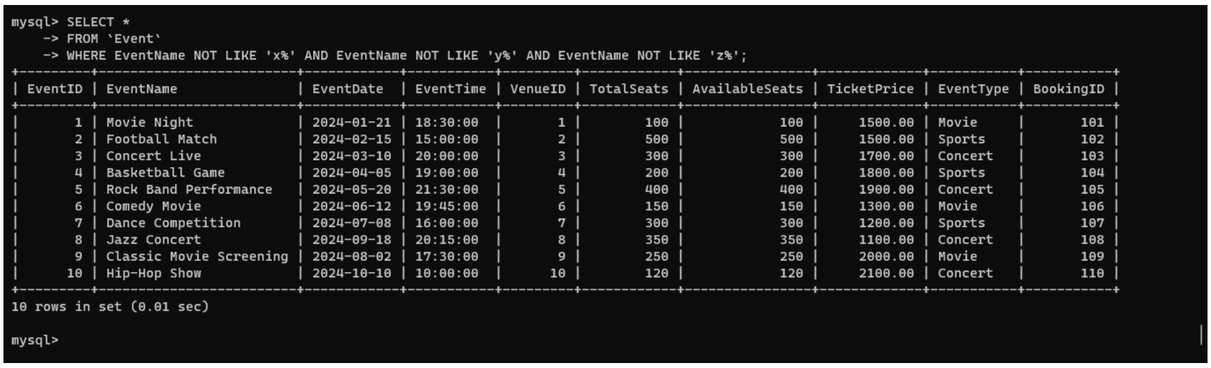
12. Write a SQL query to select events name not start with ‘x’, ‘y’, ‘z’

QUERY:-

SELECT *

   -> FROM `Event`

   -> WHERE EventName NOT LIKE 'x%' AND EventName NOT LIKE 'y%' AND EventName NOT LIKE 'z%';



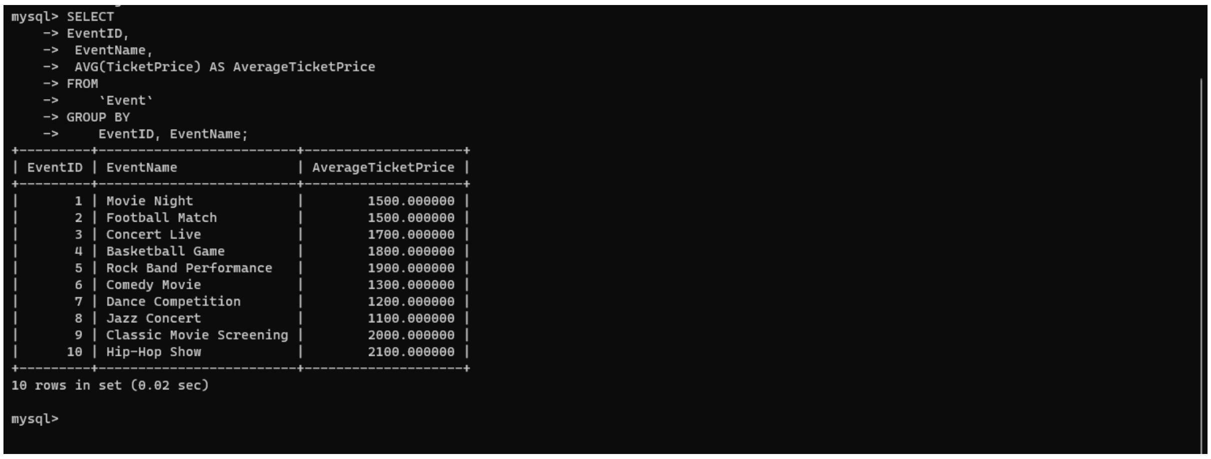
```
mysql> SELECT *
    -> FROM `Event`
    -> WHERE EventName NOT LIKE 'x%' AND EventName NOT LIKE 'y%' AND EventName NOT LIKE 'z%';
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
| EventID | EventName               | EventDate  | EventTime | VenueID | TotalSeats | AvailableSeats | TicketPrice | EventType | BookingID |
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
|       1 | Movie Night             | 2024-01-21 | 18:30:00  |       1 |        100 |            100 |     1500.00 | Movie     |       101 |
|       2 | Football Match          | 2024-02-15 | 15:00:00  |       2 |        500 |            500 |     1500.00 | Sports    |       102 |
|       3 | Concert Live            | 2024-03-10 | 20:00:00  |       3 |        300 |            300 |     1700.00 | Concert   |       103 |
|       4 | Basketball Game         | 2024-04-05 | 19:00:00  |       4 |        200 |            200 |     1800.00 | Sports    |       104 |
|       5 | Rock Band Performance   | 2024-05-20 | 21:30:00  |       5 |        400 |            400 |     1900.00 | Concert   |       105 |
|       6 | Comedy Movie            | 2024-06-12 | 19:45:00  |       6 |        150 |            150 |     1300.00 | Movie     |       106 |
|       7 | Dance Competition       | 2024-07-08 | 16:00:00  |       7 |        300 |            300 |     1200.00 | Sports    |       107 |
|       8 | Jazz Concert            | 2024-09-18 | 20:15:00  |       8 |        350 |            350 |     1100.00 | Concert   |       108 |
|       9 | Classic Movie Screening | 2024-08-02 | 17:30:00  |       9 |        250 |            250 |     2000.00 | Movie     |       109 |
|      10 | Hip-Hop Show            | 2024-10-10 | 10:00:00  |      10 |        120 |            120 |     2100.00 | Concert   |       110 |
+---------+-------------------------+------------+-----------+---------+------------+----------------+-------------+-----------+-----------+
10 rows in set (0.01 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.

QUERY:-

```
SELECT
   -> EventID,
   ->  EventName,
   ->  AVG(TicketPrice) AS AverageTicketPrice
   -> FROM
   ->     `Event`
   -> GROUP BY
   ->     EventID, EventName;
```

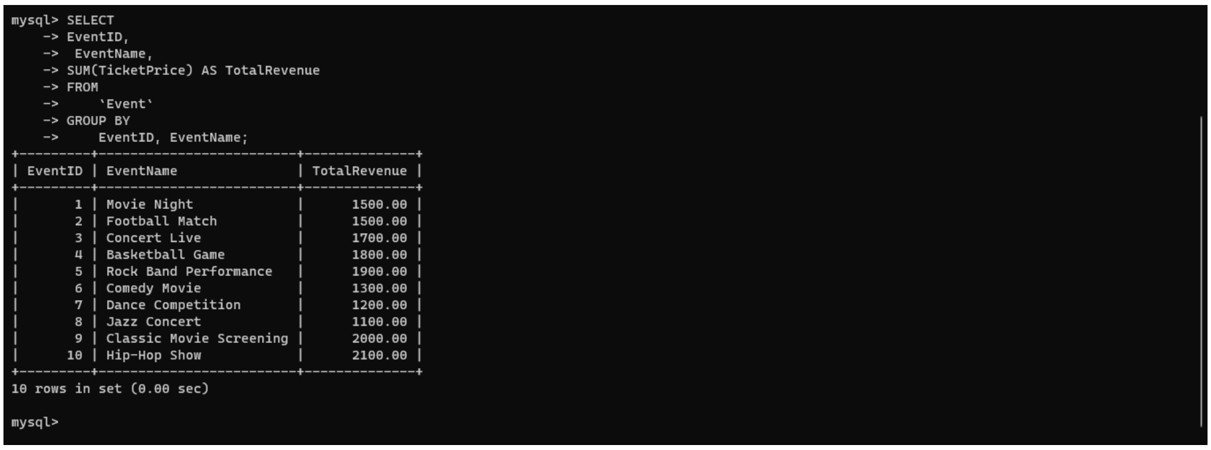


```
mysql> SELECT
    -> EventID,
    ->  EventName,
    ->  AVG(TicketPrice) AS AverageTicketPrice
    -> FROM
    ->     `Event`
    -> GROUP BY
    ->     EventID, EventName;
+---------+-------------------------+--------------------+
| EventID | EventName               | AverageTicketPrice |
+---------+-------------------------+--------------------+
|       1 | Movie Night             |        1500.000000 |
|       2 | Football Match          |        1500.000000 |
|       3 | Concert Live            |        1700.000000 |
|       4 | Basketball Game         |        1800.000000 |
|       5 | Rock Band Performance   |        1900.000000 |
|       6 | Comedy Movie            |        1300.000000 |
|       7 | Dance Competition       |        1200.000000 |
|       8 | Jazz Concert            |        1100.000000 |
|       9 | Classic Movie Screening |        2000.000000 |
|      10 | Hip-Hop Show            |        2100.000000 |
+---------+-------------------------+--------------------+
10 rows in set (0.02 sec)

mysql>
```

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

QUERY:-

```
SELECT
    -> EventID,
    ->  EventName,
    -> SUM(TicketPrice) AS TotalRevenue
    -> FROM
    ->     `Event`
    -> GROUP BY
    ->     EventID, EventName;
```

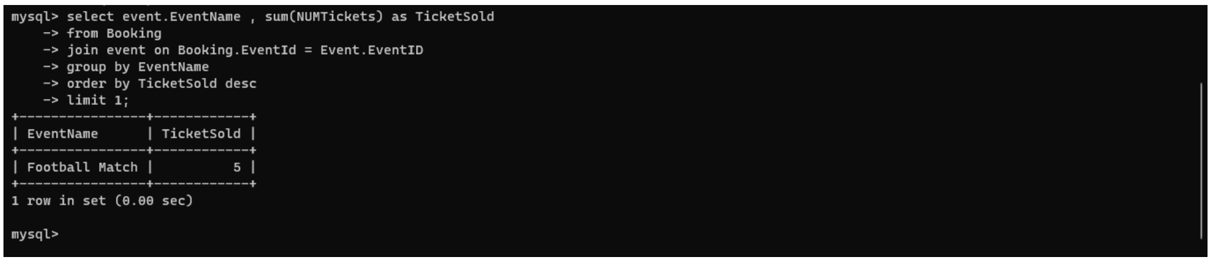


```
mysql> SELECT
    -> EventID,
    ->  EventName,
    -> SUM(TicketPrice) AS TotalRevenue
    -> FROM
    ->     `Event`
    -> GROUP BY
    ->     EventID, EventName;
+---------+-------------------------+--------------+
| EventID | EventName               | TotalRevenue |
+---------+-------------------------+--------------+
|       1 | Movie Night             |      1500.00 |
|       2 | Football Match          |      1500.00 |
|       3 | Concert Live            |      1700.00 |
|       4 | Basketball Game         |      1800.00 |
|       5 | Rock Band Performance   |      1900.00 |
|       6 | Comedy Movie            |      1300.00 |
|       7 | Dance Competition       |      1200.00 |
|       8 | Jazz Concert            |      1100.00 |
|       9 | Classic Movie Screening |      2000.00 |
|      10 | Hip-Hop Show            |      2100.00 |
+---------+-------------------------+--------------+
10 rows in set (0.00 sec)

mysql>
```

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

```
select event.EventName , sum(NUMTickets) as TicketSold
    -> from Booking
    -> join event on Booking.EventId = Event.EventID
    -> group by EventName
```

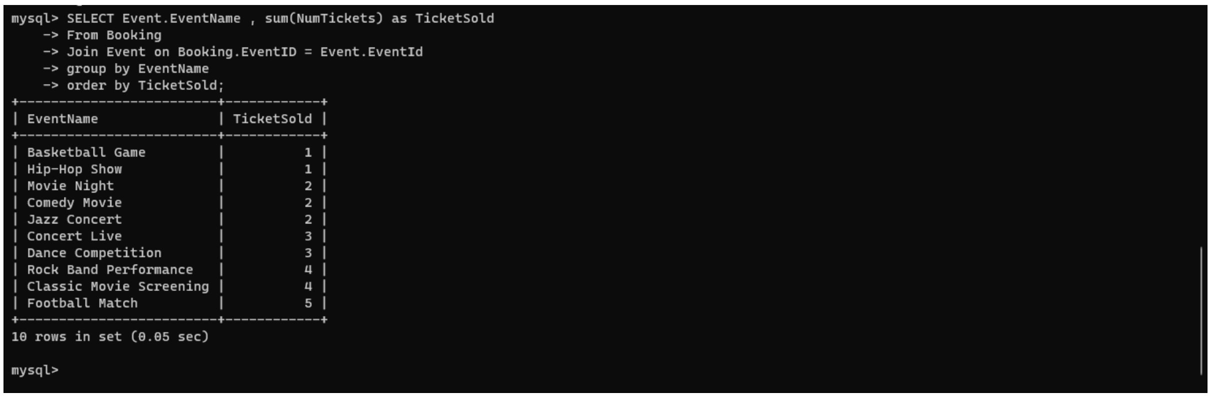
-> order by TicketSold desc
-> limit 1;



```
mysql> select event.EventName , sum(NUMTickets) as TicketSold
    -> from Booking
    -> join event on Booking.EventId = Event.EventID
    -> group by EventName
    -> order by TicketSold desc
    -> limit 1;
+----------------+------------+
| EventName      | TicketSold |
+----------------+------------+
| Football Match |          5 |
+----------------+------------+
1 row in set (0.00 sec)

mysql>
```

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

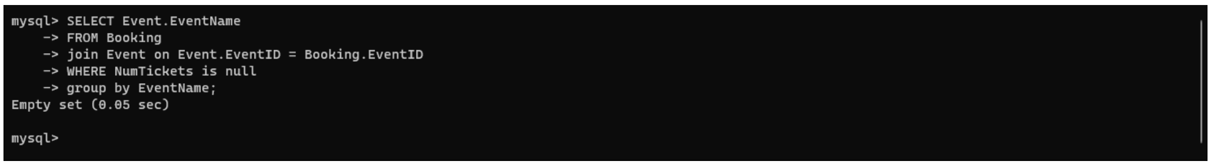
SELECT Event.EventName , sum(NumTickets) as TicketSold
-> From Booking
-> Join Event on Booking.EventID = Event.EventId
-> group by EventName
-> order by TicketSold;



```
mysql> SELECT Event.EventName , sum(NumTickets) as TicketSold
    -> From Booking
    -> Join Event on Booking.EventID = Event.EventId
    -> group by EventName
    -> order by TicketSold;
+------------------------+------------+
| EventName              | TicketSold |
+------------------------+------------+
| Basketball Game        |          1 |
| Hip-Hop Show           |          1 |
| Movie Night            |          2 |
| Comedy Movie           |          2 |
| Jazz Concert           |          2 |
| Concert Live           |          3 |
| Dance Competition      |          3 |
| Rock Band Performance  |          4 |
| Classic Movie Screening |         4 |
| Football Match         |          5 |
+------------------------+------------+
10 rows in set (0.05 sec)

mysql>
```

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

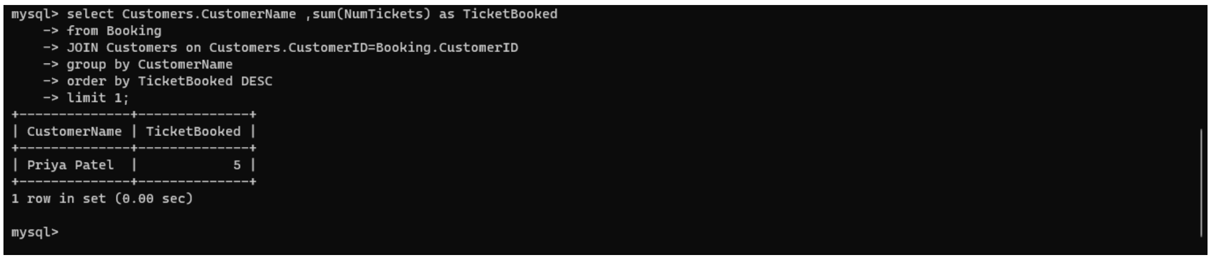
SELECT Event.EventName
-> FROM Booking
-> join Event on Event.EventID = Booking.EventID
-> WHERE NumTickets is null
-> group by EventName;



```
mysql> SELECT Event.EventName
    -> FROM Booking
    -> join Event on Event.EventID = Booking.EventID
    -> WHERE NumTickets is null
    -> group by EventName;
Empty set (0.05 sec)

mysql>
```

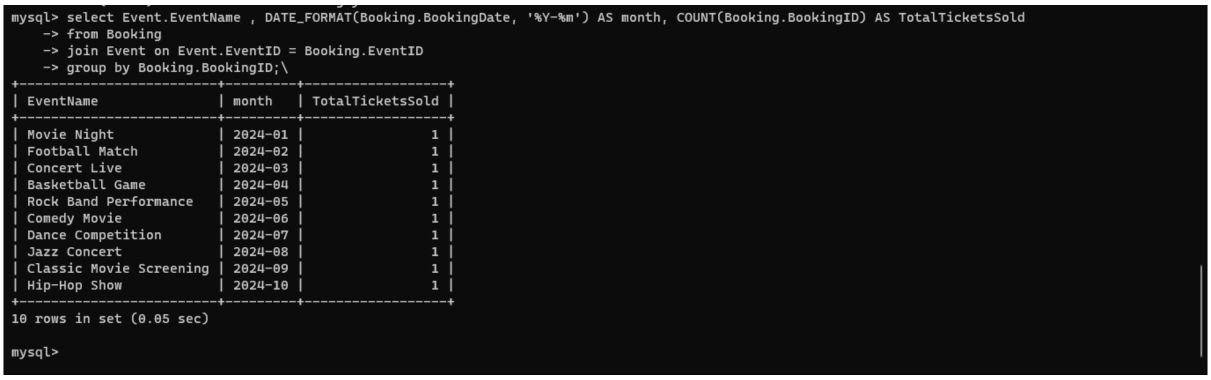
6. Write a SQL query to Find the User Who Has Booked the Most Tickets.

select Customers.CustomerName ,sum(NumTickets) as TicketBooked
-> from Booking
-> JOIN Customers on Customers.CustomerID=Booking.CustomerID
-> group by CustomerName
-> order by TicketBooked DESC
-> limit 1;



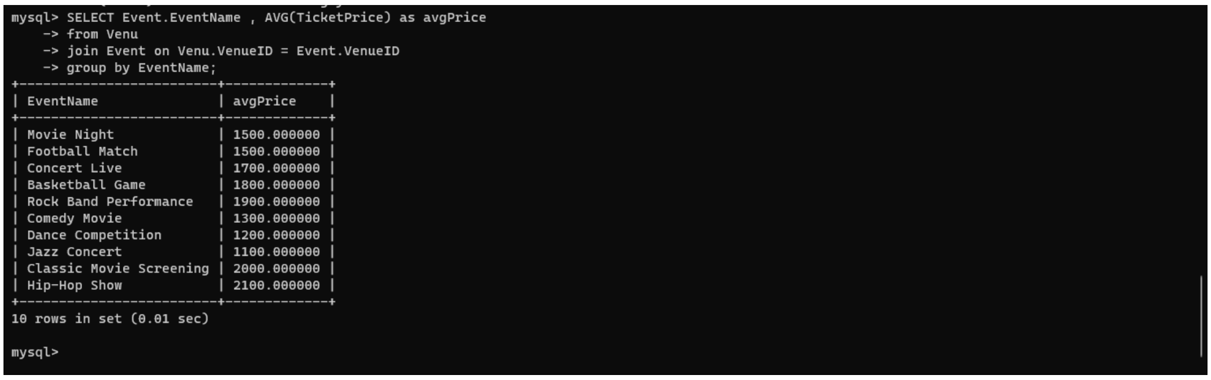
```
mysql> select Customers.CustomerName ,sum(NumTickets) as TicketBooked
    -> from Booking
    -> JOIN Customers on Customers.CustomerID=Booking.CustomerID
    -> group by CustomerName
    -> order by TicketBooked DESC
    -> limit 1;
+--------------+--------------+
| CustomerName | TicketBooked |
+--------------+--------------+
| Priya Patel  |            5 |
+--------------+--------------+
1 row in set (0.00 sec)

mysql>
```

7. Write a SQL query to List Events and the total number of tickets sold for each month.
select Event.EventName , DATE_FORMAT(Booking.BookingDate, '%Y-%m') AS month, COUNT(Booking.BookingID) AS TotalTicketsSold
-> from Booking
-> join Event on Event.EventID = Booking.EventID
-> group by Booking.BookingID;



```
mysql> select Event.EventName , DATE_FORMAT(Booking.BookingDate, '%Y-%m') AS month, COUNT(Booking.BookingID) AS TotalTicketsSold
    -> from Booking
    -> join Event on Event.EventID = Booking.EventID
    -> group by Booking.BookingID;\
+------------------------+---------+------------------+
| EventName              | month   | TotalTicketsSold |
+------------------------+---------+------------------+
| Movie Night            | 2024-01 |                1 |
| Football Match         | 2024-02 |                1 |
| Concert Live           | 2024-03 |                1 |
| Basketball Game        | 2024-04 |                1 |
| Rock Band Performance  | 2024-05 |                1 |
| Comedy Movie           | 2024-06 |                1 |
| Dance Competition      | 2024-07 |                1 |
| Jazz Concert           | 2024-08 |                1 |
| Classic Movie Screening | 2024-09 |               1 |
| Hip-Hop Show           | 2024-10 |                1 |
+------------------------+---------+------------------+
10 rows in set (0.05 sec)

mysql>
```

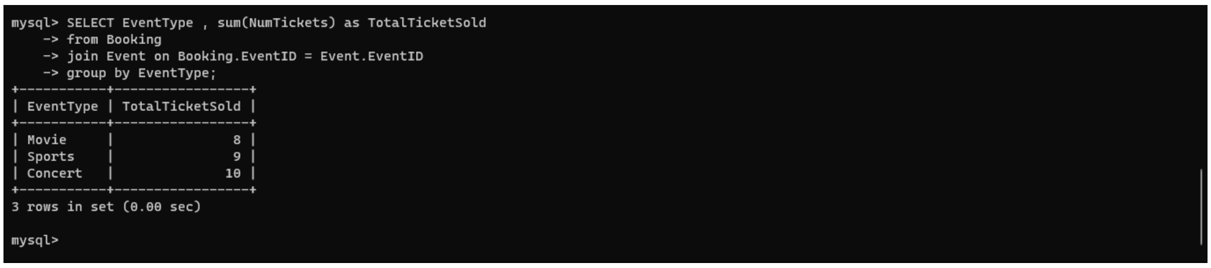
8. Write a SQL query to calculate the average Ticket Price for Events in Each Venue.
SELECT Event.EventName , AVG(TicketPrice) as avgPrice
-> from Venu
-> join Event on Venu.VenueID = Event.VenueID
-> group by EventName;



```
mysql> SELECT Event.EventName , AVG(TicketPrice) as avgPrice
    -> from Venu
    -> join Event on Venu.VenueID = Event.VenueID
    -> group by EventName;
+-------------------------+-------------+
| EventName               | avgPrice    |
+-------------------------+-------------+
| Movie Night             | 1500.000000 |
| Football Match          | 1500.000000 |
| Concert Live            | 1700.000000 |
| Basketball Game         | 1800.000000 |
| Rock Band Performance   | 1900.000000 |
| Comedy Movie            | 1300.000000 |
| Dance Competition       | 1200.000000 |
| Jazz Concert            | 1100.000000 |
| Classic Movie Screening | 2000.000000 |
| Hip-Hop Show            | 2100.000000 |
+-------------------------+-------------+
10 rows in set (0.01 sec)

mysql>
```

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

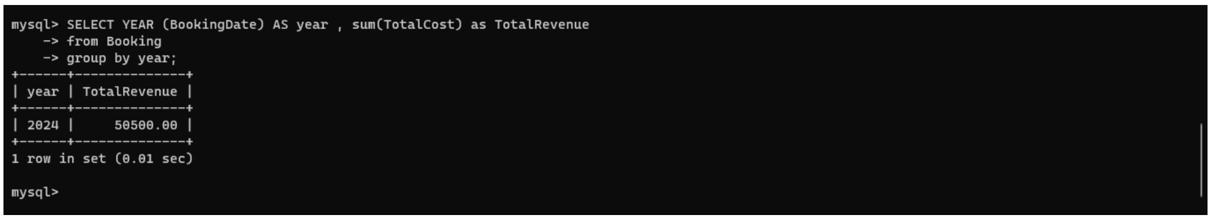
```
SELECT EventType , sum(NumTickets) as TotalTicketSold
    -> from Booking
    -> join Event on Booking.EventID = Event.EventID
    -> group by EventType;
```



```
mysql> SELECT EventType , sum(NumTickets) as TotalTicketSold
    -> from Booking
    -> join Event on Booking.EventID = Event.EventID
    -> group by EventType;
+-----------+-----------------+
| EventType | TotalTicketSold |
+-----------+-----------------+
| Movie     |               8 |
| Sports    |               9 |
| Concert   |              10 |
+-----------+-----------------+
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 (BookingDate) AS year , sum(TotalCost) as TotalRevenue
    -> from Booking
    -> group by year;
```

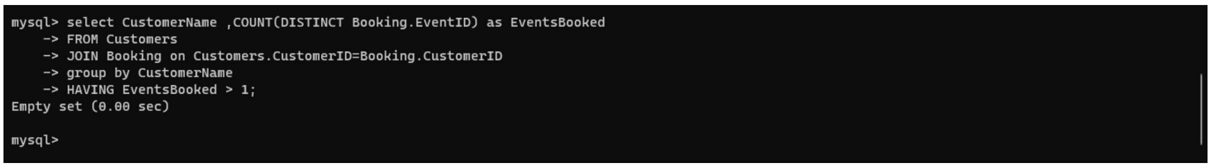


```
mysql> SELECT YEAR (BookingDate) AS year , sum(TotalCost) as TotalRevenue
    -> from Booking
    -> group by year;
+------+--------------+
| year | TotalRevenue |
+------+--------------+
| 2024 |     50500.00 |
+------+--------------+
1 row in set (0.01 sec)

mysql>
```

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

```
select CustomerName ,COUNT(DISTINCT Booking.EventID) as EventsBooked
    -> FROM Customers
```

    -> JOIN Booking on
Customers.CustomerID=Booking.CustomerID
    -> group by CustomerName
    -> HAVING EventsBooked > 1;



```
mysql> select CustomerName ,COUNT(DISTINCT Booking.EventID) as EventsBooked
    -> FROM Customers
    -> JOIN Booking on Customers.CustomerID=Booking.CustomerID
    -> group by CustomerName
    -> HAVING EventsBooked > 1;
Empty set (0.00 sec)

mysql>
```

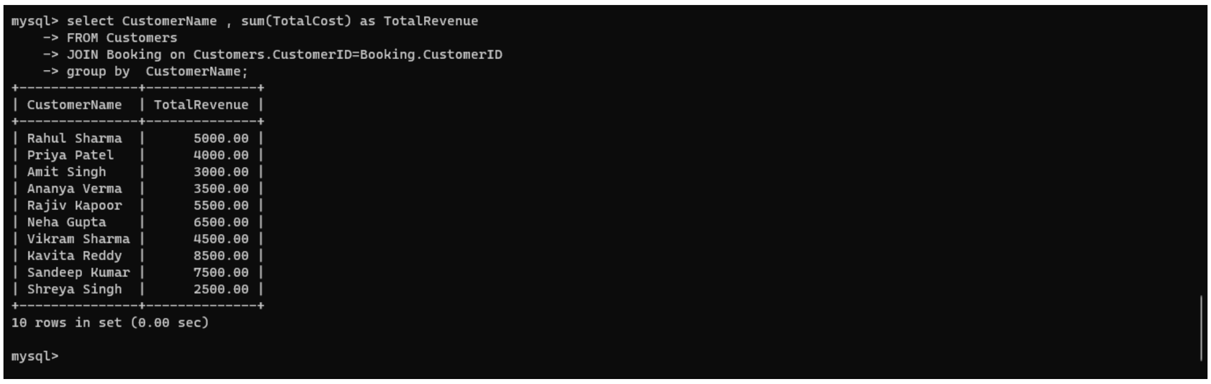
12.Write a SQL query to calculate the Total Revenue Generated by Events for Each User

select CustomerName , sum(TotalCost) as TotalRevenue

   -> FROM Customers

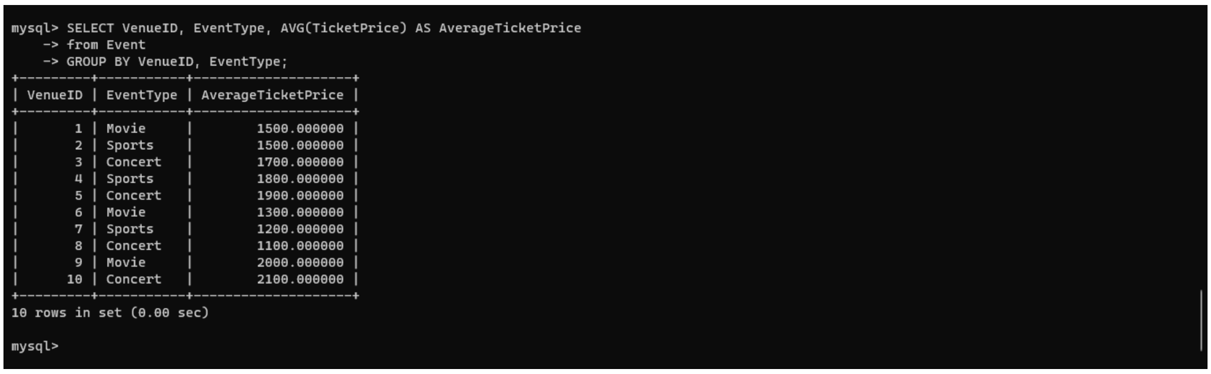
   -> JOIN Booking on Customers.CustomerID=Booking.CustomerID

   -> group by CustomerName;



```
mysql> select CustomerName , sum(TotalCost) as TotalRevenue
    -> FROM Customers
    -> JOIN Booking on Customers.CustomerID=Booking.CustomerID
    -> group by  CustomerName;
+---------------+--------------+
| CustomerName  | TotalRevenue |
+---------------+--------------+
| Rahul Sharma  |      5000.00 |
| Priya Patel   |      4000.00 |
| Amit Singh    |      3000.00 |
| Ananya Verma  |      3500.00 |
| Rajiv Kapoor  |      5500.00 |
| Neha Gupta    |      6500.00 |
| Vikram Sharma |      4500.00 |
| Kavita Reddy  |      8500.00 |
| Sandeep Kumar |      7500.00 |
| Shreya Singh  |      2500.00 |
+---------------+--------------+
10 rows in set (0.00 sec)

mysql>
```

13. Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.
SELECT VenueID, EventType, AVG(TicketPrice) AS AverageTicketPrice
    -> from Event
    -> GROUP BY VenueID, EventType;



```
mysql> SELECT VenueID, EventType, AVG(TicketPrice) AS AverageTicketPrice
    -> from Event
    -> GROUP BY VenueID, EventType;
+---------+-----------+--------------------+
| VenueID | EventType | AverageTicketPrice |
+---------+-----------+--------------------+
|       1 | Movie     |        1500.000000 |
|       2 | Sports    |        1500.000000 |
|       3 | Concert   |        1700.000000 |
|       4 | Sports    |        1800.000000 |
|       5 | Concert   |        1900.000000 |
|       6 | Movie     |        1300.000000 |
|       7 | Sports    |        1200.000000 |
|       8 | Concert   |        1100.000000 |
|       9 | Movie     |        2000.000000 |
|      10 | Concert   |        2100.000000 |
+---------+-----------+--------------------+
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.

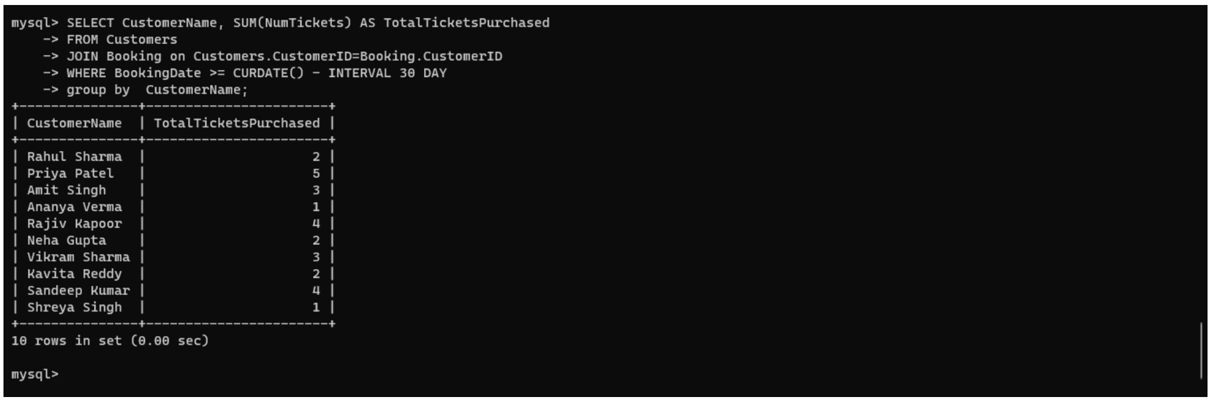
SELECT CustomerName, SUM(NumTickets) AS TotalTicketsPurchased

-> FROM Customers

-> JOIN Booking on Customers.CustomerID=Booking.CustomerID

-> WHERE BookingDate >= CURDATE() - INTERVAL 30 DAY

-> group by CustomerName;



```
mysql> SELECT CustomerName, SUM(NumTickets) AS TotalTicketsPurchased
    -> FROM Customers
    -> JOIN Booking on Customers.CustomerID=Booking.CustomerID
    -> WHERE BookingDate >= CURDATE() - INTERVAL 30 DAY
    -> group by  CustomerName;
+---------------+-----------------------+
| CustomerName  | TotalTicketsPurchased |
+---------------+-----------------------+
| Rahul Sharma  |                     2 |
| Priya Patel   |                     5 |
| Amit Singh    |                     3 |
| Ananya Verma  |                     1 |
| Rajiv Kapoor  |                     4 |
| Neha Gupta    |                     2 |
| Vikram Sharma |                     3 |
| Kavita Reddy  |                     2 |
| Sandeep Kumar |                     4 |
| Shreya Singh  |                     1 |
+---------------+-----------------------+
10 rows in set (0.00 sec)

mysql>
```

Tasks 4: Subquery and its types

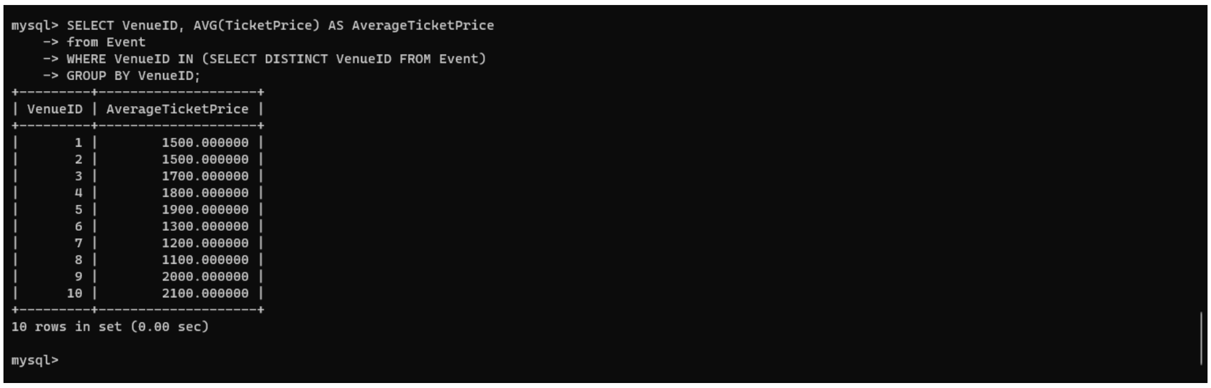
1. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery.

   SELECT VenueID, AVG(TicketPrice) AS AverageTicketPrice

   -> from Event

   -> WHERE VenueID IN (SELECT DISTINCT VenueID FROM Event)

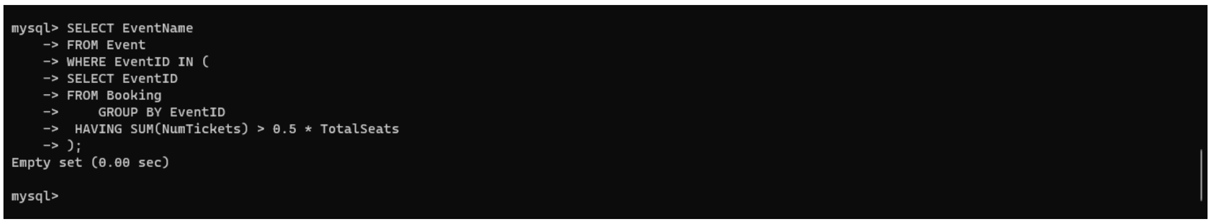
   -> GROUP BY VenueID;



```
mysql> SELECT VenueID, AVG(TicketPrice) AS AverageTicketPrice
    -> from Event
    -> WHERE VenueID IN (SELECT DISTINCT VenueID FROM Event)
    -> GROUP BY VenueID;
+---------+--------------------+
| VenueID | AverageTicketPrice |
+---------+--------------------+
|       1 |        1500.000000 |
|       2 |        1500.000000 |
|       3 |        1700.000000 |
|       4 |        1800.000000 |
|       5 |        1900.000000 |
|       6 |        1300.000000 |
|       7 |        1200.000000 |
|       8 |        1100.000000 |
|       9 |        2000.000000 |
|      10 |        2100.000000 |
+---------+--------------------+
10 rows in set (0.00 sec)

mysql>
```

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

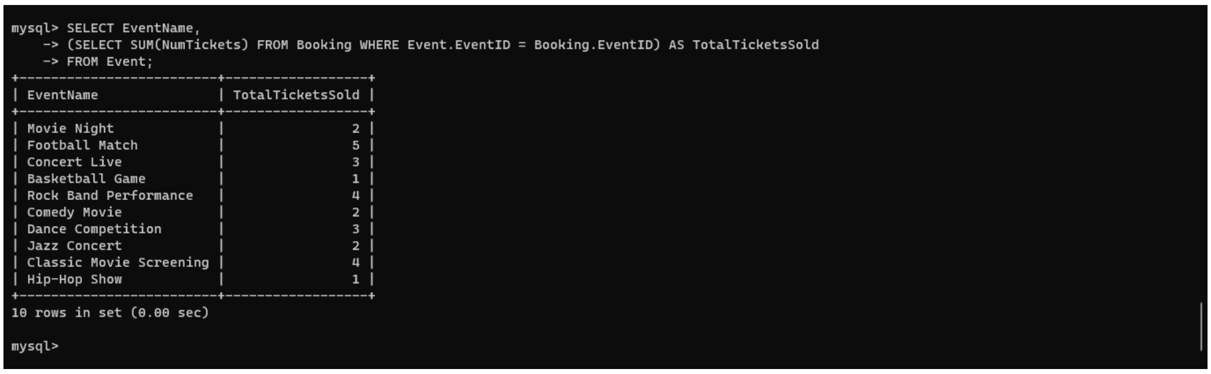
```
SELECT EventName
   -> FROM Event
   -> WHERE EventID IN (
   -> SELECT EventID
   -> FROM Booking
   ->     GROUP BY EventID
   ->  HAVING SUM(NumTickets) > 0.5 * TotalSeats
   -> );
```



```
mysql> SELECT EventName
    -> FROM Event
    -> WHERE EventID IN (
    -> SELECT EventID
    -> FROM Booking
    ->     GROUP BY EventID
    ->  HAVING SUM(NumTickets) > 0.5 * TotalSeats
    -> );
Empty set (0.00 sec)

mysql>
```

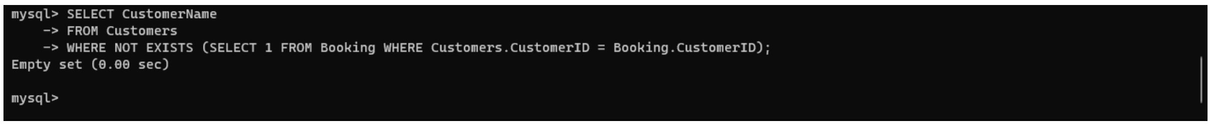
3. Calculate the Total Number of Tickets Sold for Each Event.

```
SELECT EventName,
   -> (SELECT SUM(NumTickets) FROM Booking WHERE
Event.EventID = Booking.EventID) AS TotalTicketsSold
   -> FROM Event;
```



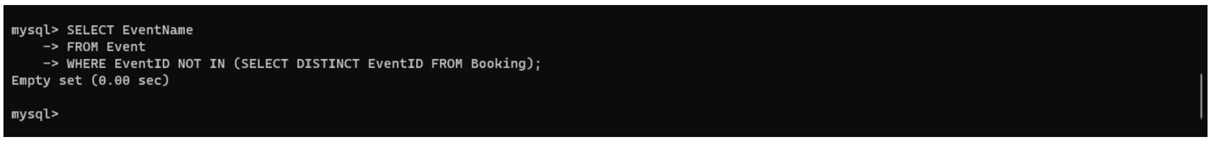
```
mysql> SELECT EventName,
    -> (SELECT SUM(NumTickets) FROM Booking WHERE Event.EventID = Booking.EventID) AS TotalTicketsSold
    -> FROM Event;
+------------------------+------------------+
| EventName              | TotalTicketsSold |
+------------------------+------------------+
| Movie Night            |                2 |
| Football Match         |                5 |
| Concert Live           |                3 |
| Basketball Game        |                1 |
| Rock Band Performance  |                4 |
| Comedy Movie           |                2 |
| Dance Competition      |                3 |
| Jazz Concert           |                2 |
| Classic Movie Screening |               4 |
| Hip-Hop Show           |                1 |
+------------------------+------------------+
10 rows in set (0.00 sec)

mysql>
```

4. Find Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.
   SELECT CustomerName
   -> FROM Customers
   -> WHERE NOT EXISTS (SELECT 1 FROM Booking WHERE Customers.CustomerID = Booking.CustomerID);



```
mysql> SELECT CustomerName
    -> FROM Customers
    -> WHERE NOT EXISTS (SELECT 1 FROM Booking WHERE Customers.CustomerID = Booking.CustomerID);
Empty set (0.00 sec)

mysql>
```

5. List Events with No Ticket Sales Using a NOT IN Subquery.
   SELECT EventName
   -> FROM Event
   -> WHERE EventID NOT IN (SELECT DISTINCT EventID FROM Booking);



```
mysql> SELECT EventName
    -> FROM Event
    -> WHERE EventID NOT IN (SELECT DISTINCT EventID FROM Booking);
Empty set (0.00 sec)

mysql>
```

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

SELECT EventType, SUM(TotalTicketsSold) AS TotalTicketsSold

-> FROM (

->   SELECT EventType, COUNT(Event.BookingID) AS TotalTicketsSold

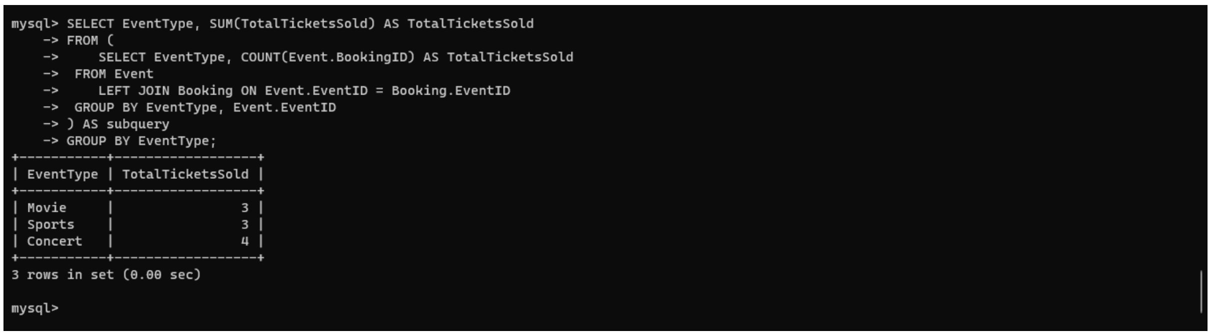
-> FROM Event

->   LEFT JOIN Booking ON Event.EventID = Booking.EventID

-> GROUP BY EventType, Event.EventID

->) AS subquery

-> GROUP BY EventType;



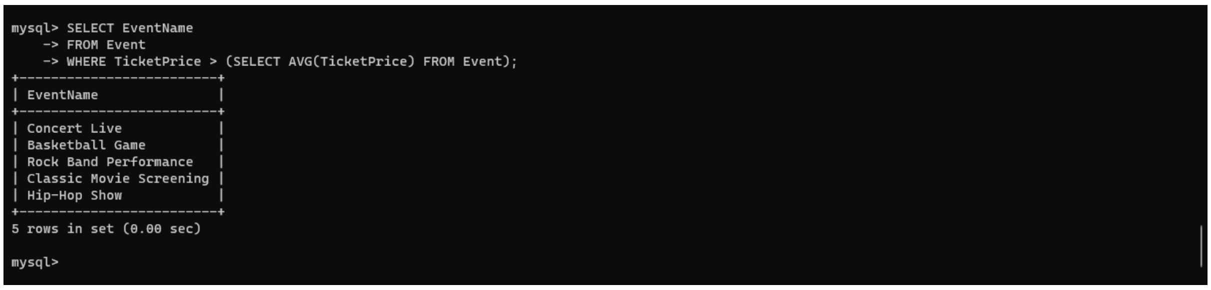
```
mysql> SELECT EventType, SUM(TotalTicketsSold) AS TotalTicketsSold
    -> FROM (
    ->     SELECT EventType, COUNT(Event.BookingID) AS TotalTicketsSold
    ->  FROM Event
    ->     LEFT JOIN Booking ON Event.EventID = Booking.EventID
    ->  GROUP BY EventType, Event.EventID
    -> ) AS subquery
    -> GROUP BY EventType;
+-----------+------------------+
| EventType | TotalTicketsSold |
+-----------+------------------+
| Movie     |                3 |
| Sports    |                3 |
| Concert   |                4 |
+-----------+------------------+
3 rows in set (0.00 sec)

mysql>
```

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

SELECT EventName

-> FROM Event

-> WHERE TicketPrice > (SELECT AVG(TicketPrice) FROM Event);



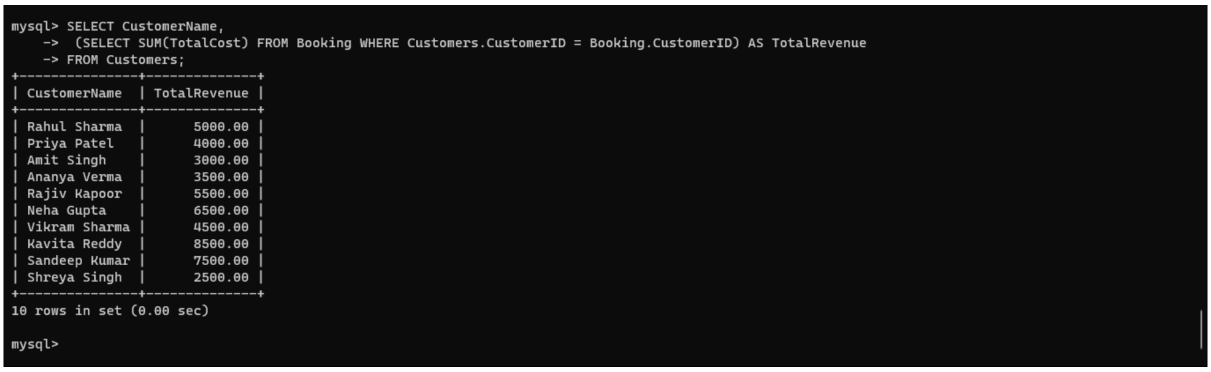
```
mysql> SELECT EventName
    -> FROM Event
    -> WHERE TicketPrice > (SELECT AVG(TicketPrice) FROM Event);
+-------------------------+
| EventName               |
+-------------------------+
| Concert Live            |
| Basketball Game         |
| Rock Band Performance   |
| Classic Movie Screening |
| Hip-Hop Show            |
+-------------------------+
5 rows in set (0.00 sec)

mysql>
```

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

SELECT CustomerName,

-> (SELECT SUM(TotalCost) FROM Booking WHERE Customers.CustomerID = Booking.CustomerID) AS TotalRevenue

-> FROM Customers;



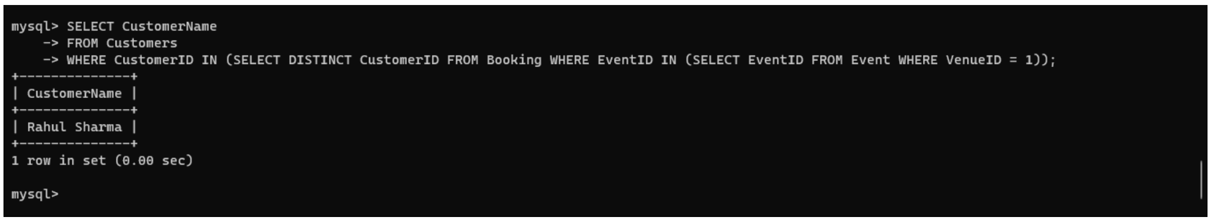
```
mysql> SELECT CustomerName,
    ->  (SELECT SUM(TotalCost) FROM Booking WHERE Customers.CustomerID = Booking.CustomerID) AS TotalRevenue
    -> FROM Customers;
+---------------+--------------+
| CustomerName  | TotalRevenue |
+---------------+--------------+
| Rahul Sharma  |      5000.00 |
| Priya Patel   |      4000.00 |
| Amit Singh    |      3000.00 |
| Ananya Verma  |      3500.00 |
| Rajiv Kapoor  |      5500.00 |
| Neha Gupta    |      6500.00 |
| Vikram Sharma |      4500.00 |
| Kavita Reddy  |      8500.00 |
| Sandeep Kumar |      7500.00 |
| Shreya Singh  |      2500.00 |
+---------------+--------------+
10 rows in set (0.00 sec)

mysql>
```

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

SELECT CustomerName

-> FROM Customers

-> WHERE CustomerID IN (SELECT DISTINCT CustomerID FROM Booking WHERE EventID IN (SELECT EventID FROM Event WHERE VenueID = 1));



```
mysql> SELECT CustomerName
    -> FROM Customers
    -> WHERE CustomerID IN (SELECT DISTINCT CustomerID FROM Booking WHERE EventID IN (SELECT EventID FROM Event WHERE VenueID = 1));
+--------------+
| CustomerName |
+--------------+
| Rahul Sharma |
+--------------+
1 row in set (0.00 sec)

mysql>
```

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

SELECT EventType, SUM(TotalTicketsSold) AS TotalTicketsSold

-> FROM (

->   SELECT EventType, COUNT(Event.BookingID) AS TotalTicketsSold

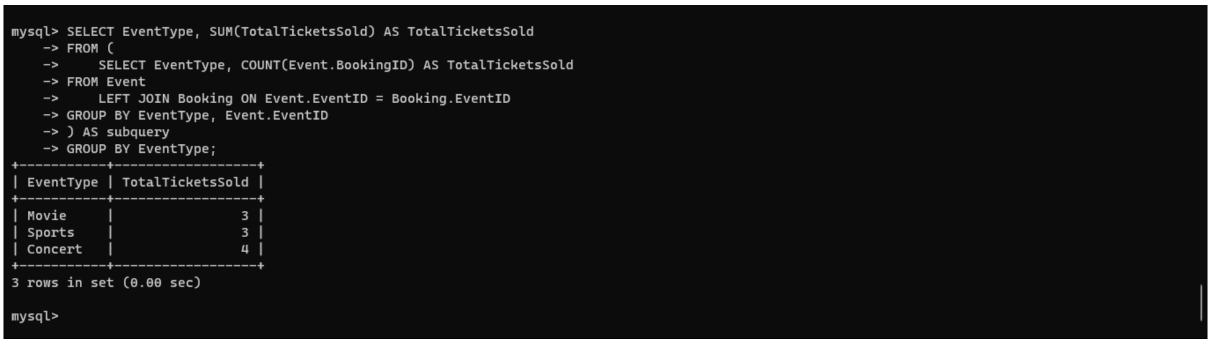
-> FROM Event

->   LEFT JOIN Booking ON Event.EventID = Booking.EventID

-> GROUP BY EventType, Event.EventID

->) AS subquery

-> GROUP BY EventType;



```
mysql> SELECT EventType, SUM(TotalTicketsSold) AS TotalTicketsSold
    -> FROM (
    ->     SELECT EventType, COUNT(Event.BookingID) AS TotalTicketsSold
    -> FROM Event
    ->     LEFT JOIN Booking ON Event.EventID = Booking.EventID
    -> GROUP BY EventType, Event.EventID
    -> ) AS subquery
    -> GROUP BY EventType;
+-----------+------------------+
| EventType | TotalTicketsSold |
+-----------+------------------+
| Movie     |                3 |
| Sports    |                3 |
| Concert   |                4 |
+-----------+------------------+
3 rows in set (0.00 sec)

mysql>
```

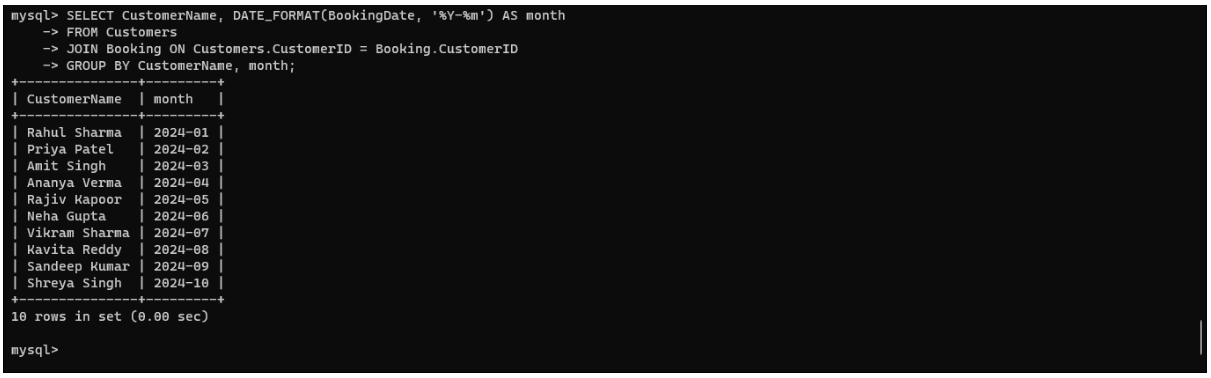
11. Find Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE_FORMAT

SELECT CustomerName, DATE_FORMAT(BookingDate, '%Y-%m') AS month

-> FROM Customers

-> JOIN Booking ON Customers.CustomerID = Booking.CustomerID

-> GROUP BY CustomerName, month;



```
mysql> SELECT CustomerName, DATE_FORMAT(BookingDate, '%Y-%m') AS month
    -> FROM Customers
    -> JOIN Booking ON Customers.CustomerID = Booking.CustomerID
    -> GROUP BY CustomerName, month;
+---------------+---------+
| CustomerName  | month   |
+---------------+---------+
| Rahul Sharma  | 2024-01 |
| Priya Patel   | 2024-02 |
| Amit Singh    | 2024-03 |
| Ananya Verma  | 2024-04 |
| Rajiv Kapoor  | 2024-05 |
| Neha Gupta    | 2024-06 |
| Vikram Sharma | 2024-07 |
| Kavita Reddy  | 2024-08 |
| Sandeep Kumar | 2024-09 |
| Shreya Singh  | 2024-10 |
+---------------+---------+
10 rows in set (0.00 sec)

mysql>
```

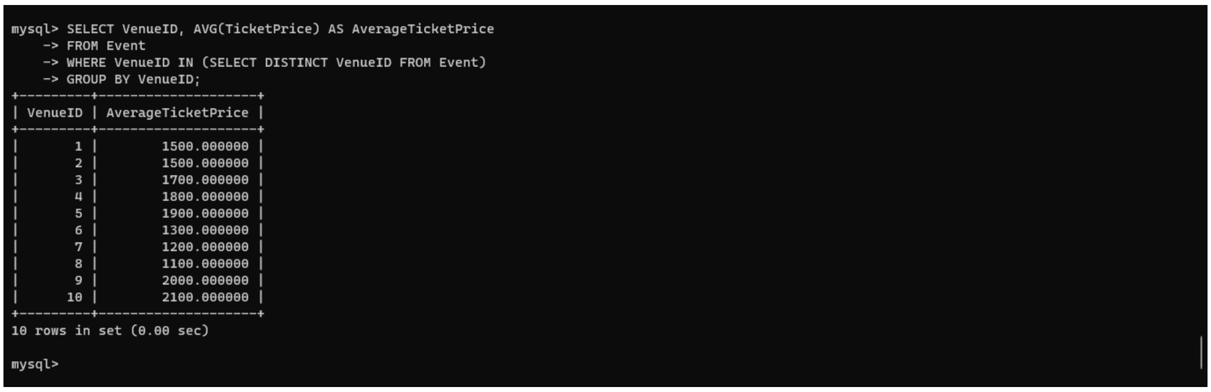
12. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery

SELECT VenueID, AVG(TicketPrice) AS AverageTicketPrice

-> FROM Event

-> WHERE VenueID IN (SELECT DISTINCT VenueID FROM Event)

-> GROUP BY VenueID;



```
mysql> SELECT VenueID, AVG(TicketPrice) AS AverageTicketPrice
    -> FROM Event
    -> WHERE VenueID IN (SELECT DISTINCT VenueID FROM Event)
    -> GROUP BY VenueID;
+---------+--------------------+
| VenueID | AverageTicketPrice |
+---------+--------------------+
|       1 |        1500.000000 |
|       2 |        1500.000000 |
|       3 |        1700.000000 |
|       4 |        1800.000000 |
|       5 |        1900.000000 |
|       6 |        1300.000000 |
|       7 |        1200.000000 |
|       8 |        1100.000000 |
|       9 |        2000.000000 |
|      10 |        2100.000000 |
+---------+--------------------+
10 rows in set (0.00 sec)

mysql>
```