SQL ASSIGNMENTS

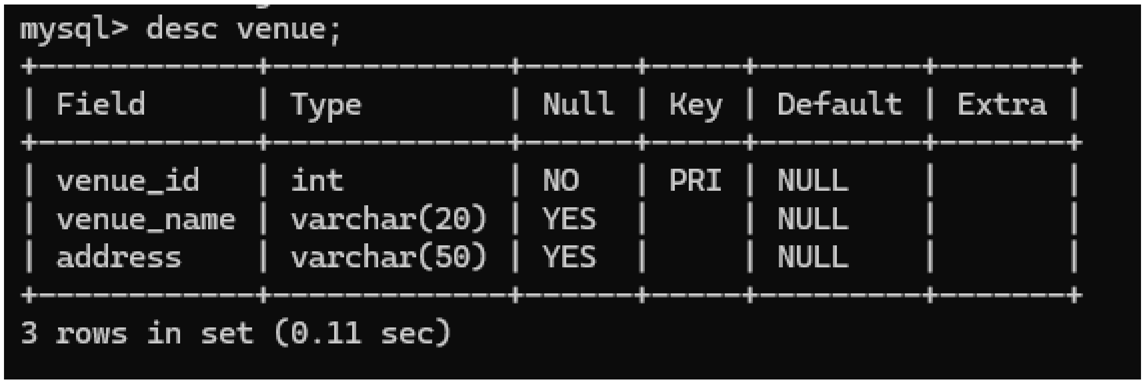
TASK 1:

1. Create the database named "TicketBookingSystem"

Create database TicketBookingSystem;

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

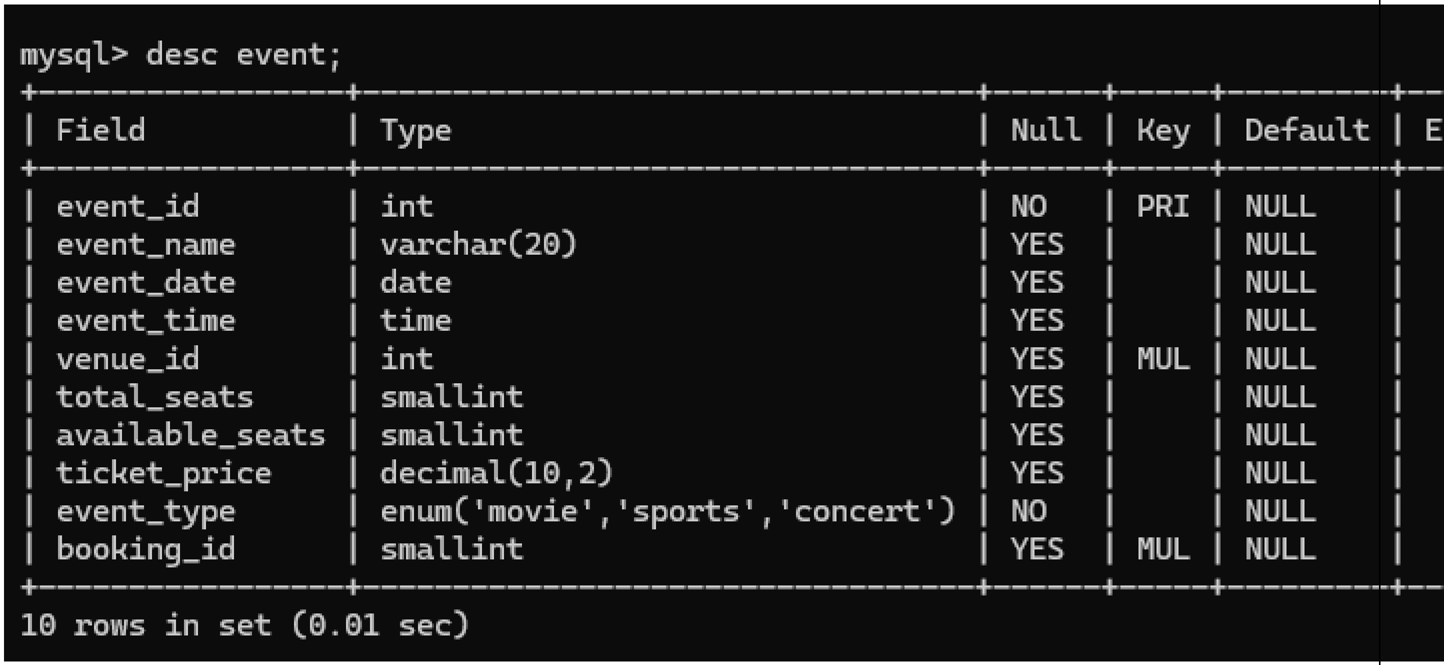
```
CREATE TABLE venue (
    venue_id INT PRIMARY KEY,
    venue_name VARCHAR(20),
    address VARCHAR(50));
```



```
mysql> desc venue;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| venue_id   | int         | NO   | PRI | NULL    |       |
| venue_name | varchar(20) | YES  |     | NULL    |       |
| address    | varchar(50) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.11 sec)
```

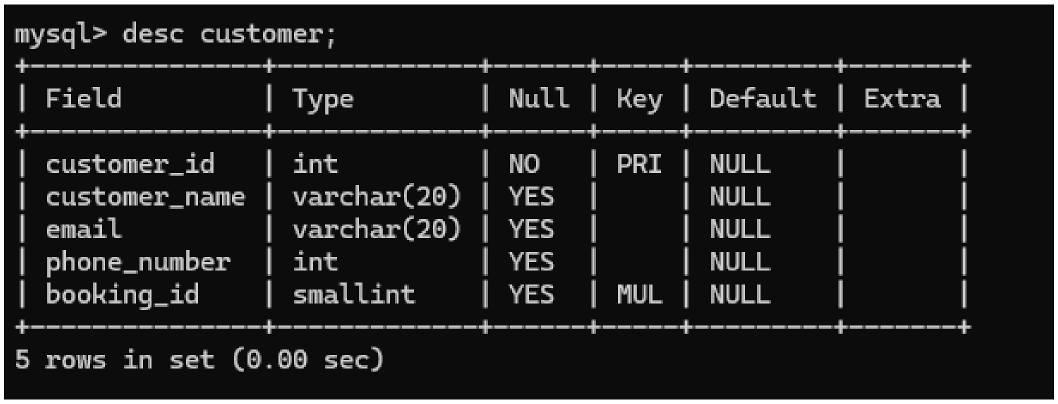
;

```
CREATE TABLE Event (
    event_id INT PRIMARY KEY,
    event_name VARCHAR(100),
    event_date DATE,
    event_time TIME,
    venue_id INT,
    total_seats INT,
    available_seats INT,
    ticket_price DECIMAL(10, 2),

    event_type ENUM('Movie', 'Sports', 'Concert'),

    booking_id INT,

    FOREIGN KEY (venue_id) REFERENCES Venue(venue_id),

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



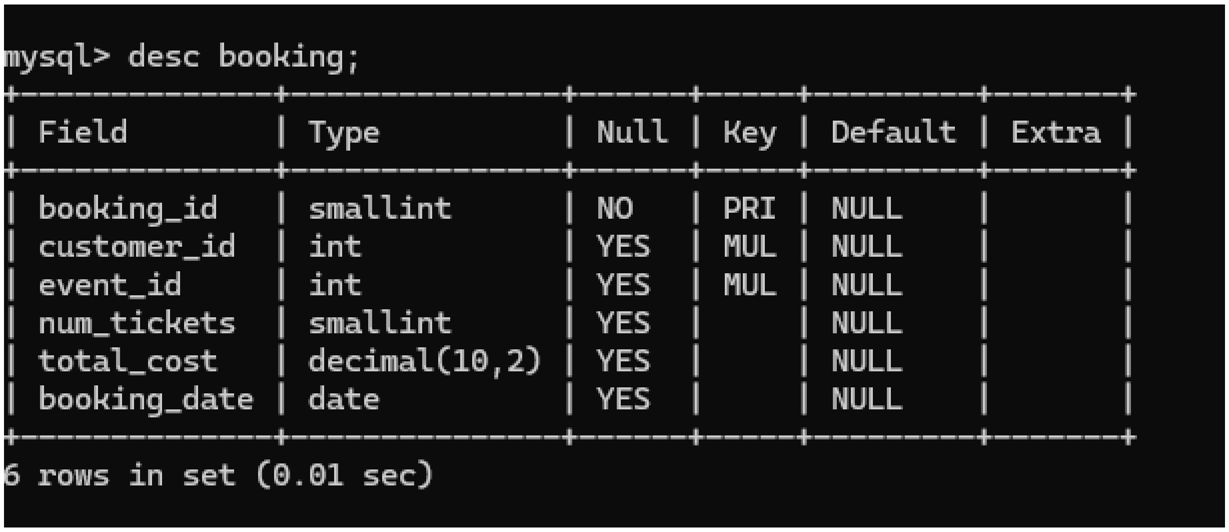
```
mysql> desc event;
+-----------------+-----------------------------------+------+-----+---------+--
| Field           | Type                              | Null | Key | Default | E
+-----------------+-----------------------------------+------+-----+---------+--
| event_id        | int                               | NO   | PRI | NULL    |
| event_name      | varchar(20)                       | YES  |     | NULL    |
| event_date      | date                              | YES  |     | NULL    |
| event_time      | time                              | YES  |     | NULL    |
| venue_id        | int                               | YES  | MUL | NULL    |
| total_seats     | smallint                          | YES  |     | NULL    |
| available_seats | smallint                          | YES  |     | NULL    |
| ticket_price    | decimal(10,2)                     | YES  |     | NULL    |
| event_type      | enum('movie','sports','concert')  | NO   |     | NULL    |
| booking_id      | smallint                          | YES  | MUL | NULL    |
+-----------------+-----------------------------------+------+-----+---------+--
10 rows in set (0.01 sec)
```

```
;

CREATE TABLE Customer (

    customer_id INT PRIMARY KEY,

    customer_name VARCHAR(100),

    email VARCHAR(100),

    phone_number VARCHAR(15) NOT,

    booking_id INT,

    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)

);
```



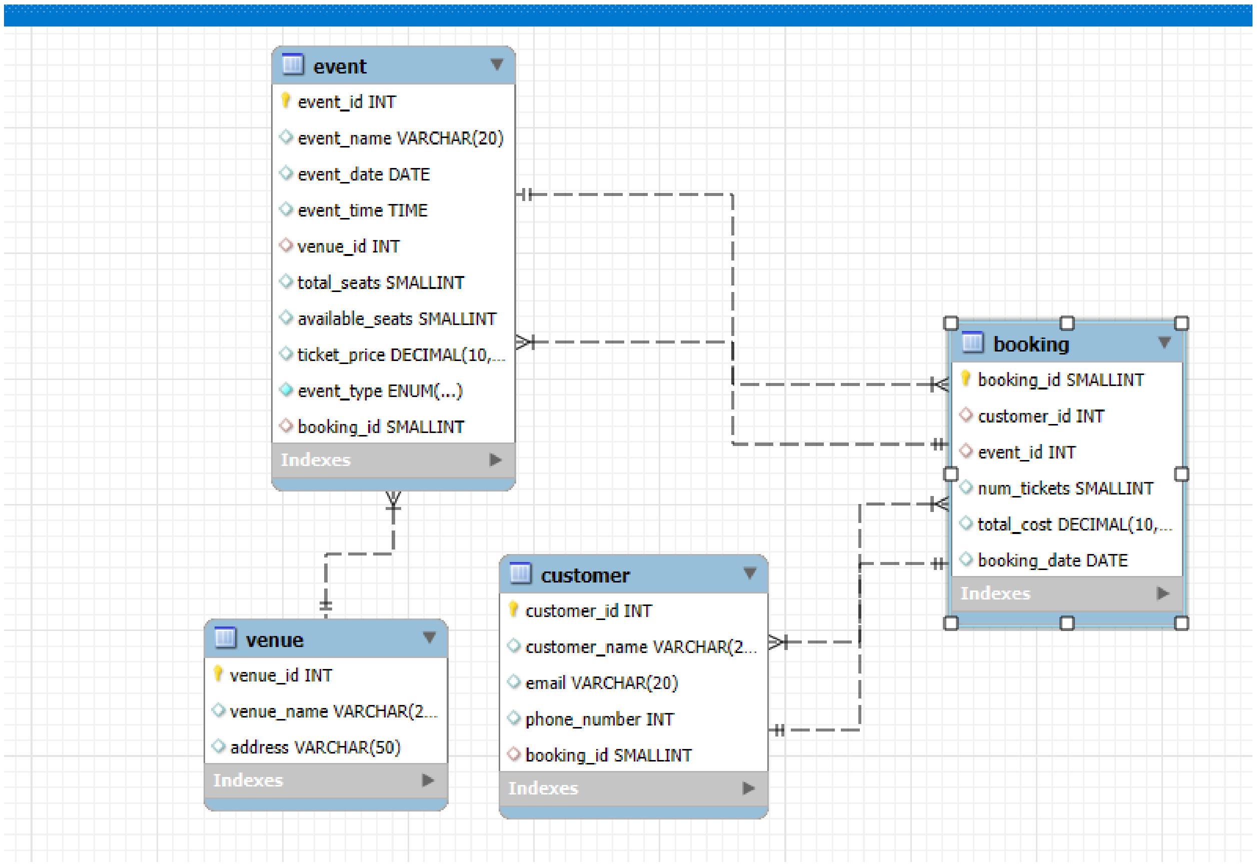
```
mysql> desc customer;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| customer_id   | int         | NO   | PRI | NULL    |       |
| customer_name | varchar(20) | YES  |     | NULL    |       |
| email         | varchar(20) | YES  |     | NULL    |       |
| phone_number  | int         | YES  |     | NULL    |       |
| booking_id    | smallint    | YES  | MUL | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY,
    customer_id INT,
    event_id INT,
    num_tickets INT,
    total_cost DECIMAL(10, 2),
    booking_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (event_id) REFERENCES Event(event_id)
);
```



```
mysql> desc booking;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| booking_id   | smallint      | NO   | PRI | NULL    |       |
| customer_id  | int           | YES  | MUL | NULL    |       |
| event_id     | int           | YES  | MUL | NULL    |       |
| num_tickets  | smallint      | YES  |     | NULL    |       |
| total_cost   | decimal(10,2) | YES  |     | NULL    |       |
| booking_date | date          | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

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



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

Hence referential integrity is maintained.

TASK 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_id, venue_name, address) VALUES

(1, 'casagrand', 'chennai'),

(2, 'pears', 'mumbai'),

(3, 'laxi', 'kolkata'),

(4, 'royal', 'chennai'),

(5, 'popi', 'salem'),

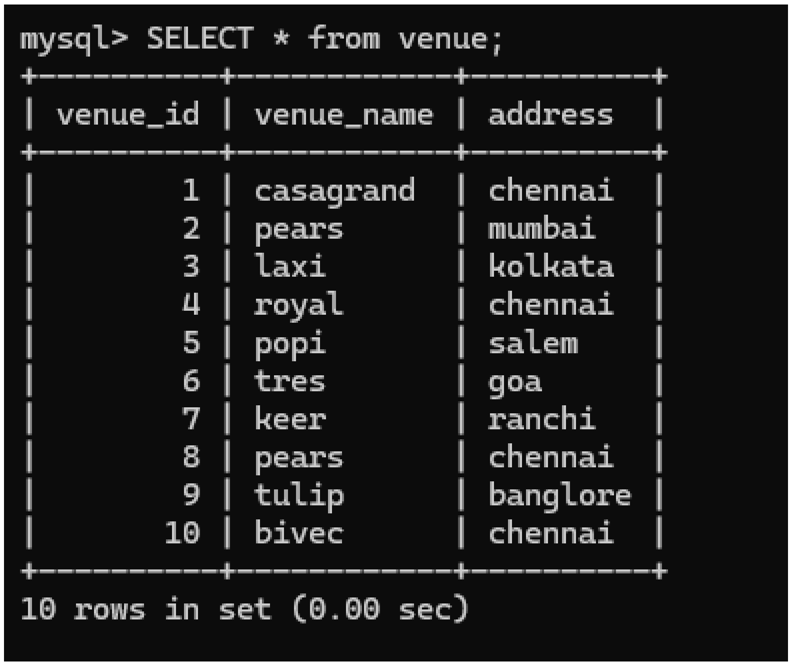
(6, 'tres', 'goa'),

(7, 'keer', 'ranchi'),

(8, 'pears', 'chennai'),

(9, 'tulip', 'banglore'),

(10, 'bivec', 'chennai');



```
mysql> SELECT * from venue;
+----------+------------+----------+
| venue_id | venue_name | address  |
+----------+------------+----------+
|        1 | casagrand  | chennai  |
|        2 | pears      | mumbai   |
|        3 | laxi       | kolkata  |
|        4 | royal      | chennai  |
|        5 | popi       | salem    |
|        6 | tres       | goa      |
|        7 | keer       | ranchi   |
|        8 | pears      | chennai  |
|        9 | tulip      | banglore |
|       10 | bivec      | chennai  |
+----------+------------+----------+
10 rows in set (0.00 sec)
```

INSERT INTO Event (event_id, event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type, booking_id) VALUES

(1, 'rock cup', '2025-01-01', '09:00:00', 1, 10050, 50, 500.05, 'concert', 110),

(2, 'worldcup', '2025-01-02', '09:00:00', 2, 10100, 100, 500.05, 'sports', 120),

(3, 'moonlight', '2025-01-03', '09:30:00', 3, 10050, 0, 1500.09, 'concert', 130),

(4, 'twilt', '2025-01-04', '10:00:00', 4, 15500, 200, 200.05, 'movie', 140),

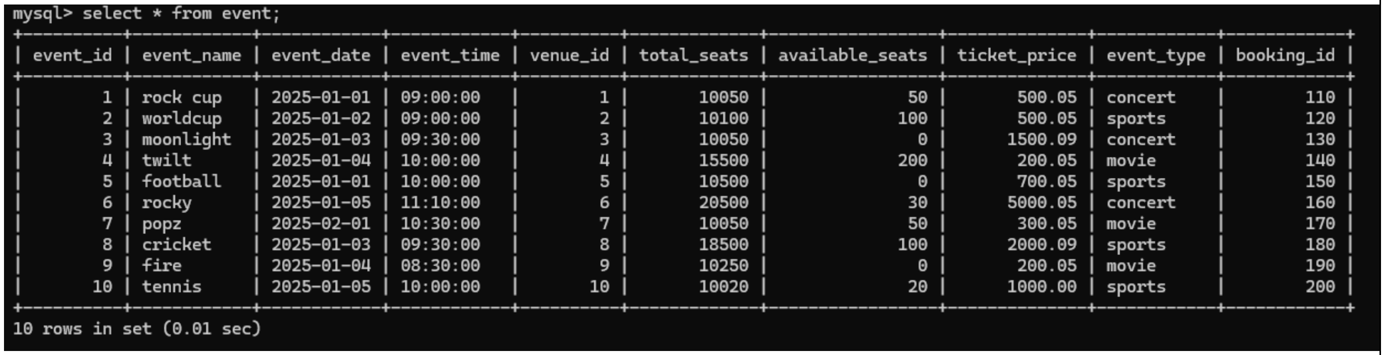
(5, 'football', '2025-01-01', '10:00:00', 5, 10500, 0, 700.05, 'sports', 150),

(6, 'rocky', '2025-01-05', '11:10:00', 6, 20500, 30, 5000.05, 'concert', 160), (7, 'popz', '2025-02-01', '10:30:00', 7, 10050, 50, 300.05, 'movie', 170),

(8, 'cricket', '2025-01-03', '09:30:00', 8, 18500, 100, 2000.09, 'sports', 180),

(9, 'fire', '2025-01-04', '08:30:00', 9, 10250, 0, 200.05, 'movie', 190),

(10, 'tennis', '2025-01-05', '10:00:00', 10, 10020, 20, 1000.00, 'sports', 200);



```
mysql> select * from event;
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | rock cup   | 2025-01-01 | 09:00:00   |        1 |       10050 |              50 |       500.05 | concert    |        110 |
|        2 | worldcup   | 2025-01-02 | 09:00:00   |        2 |       10100 |             100 |       500.05 | sports     |        120 |
|        3 | moonlight  | 2025-01-03 | 09:30:00   |        3 |       10050 |               0 |      1500.09 | concert    |        130 |
|        4 | twilt      | 2025-01-04 | 10:00:00   |        4 |       15500 |             200 |       200.05 | movie      |        140 |
|        5 | football   | 2025-01-01 | 10:00:00   |        5 |       10500 |               0 |       700.05 | sports     |        150 |
|        6 | rocky      | 2025-01-05 | 11:10:00   |        6 |       20500 |              30 |      5000.05 | concert    |        160 |
|        7 | popz       | 2025-02-01 | 10:30:00   |        7 |       10050 |              50 |       300.05 | movie      |        170 |
|        8 | cricket    | 2025-01-03 | 09:30:00   |        8 |       18500 |             100 |      2000.09 | sports     |        180 |
|        9 | fire       | 2025-01-04 | 08:30:00   |        9 |       10250 |               0 |       200.05 | movie      |        190 |
|       10 | tennis     | 2025-01-05 | 10:00:00   |       10 |       10020 |              20 |      1000.00 | sports     |        200 |
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
10 rows in set (0.01 sec)
```

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

(101, 'ramya', 'gmail', '123444000', 110),

(102, 'saru', 'gmail', '908767', 120),

(103, 'kaviya', 'yahoo', '896457', 130),

(104, 'priya', 'outlook', '892456', 140),

(105, 'sam', 'yahoo', '123097', 150),

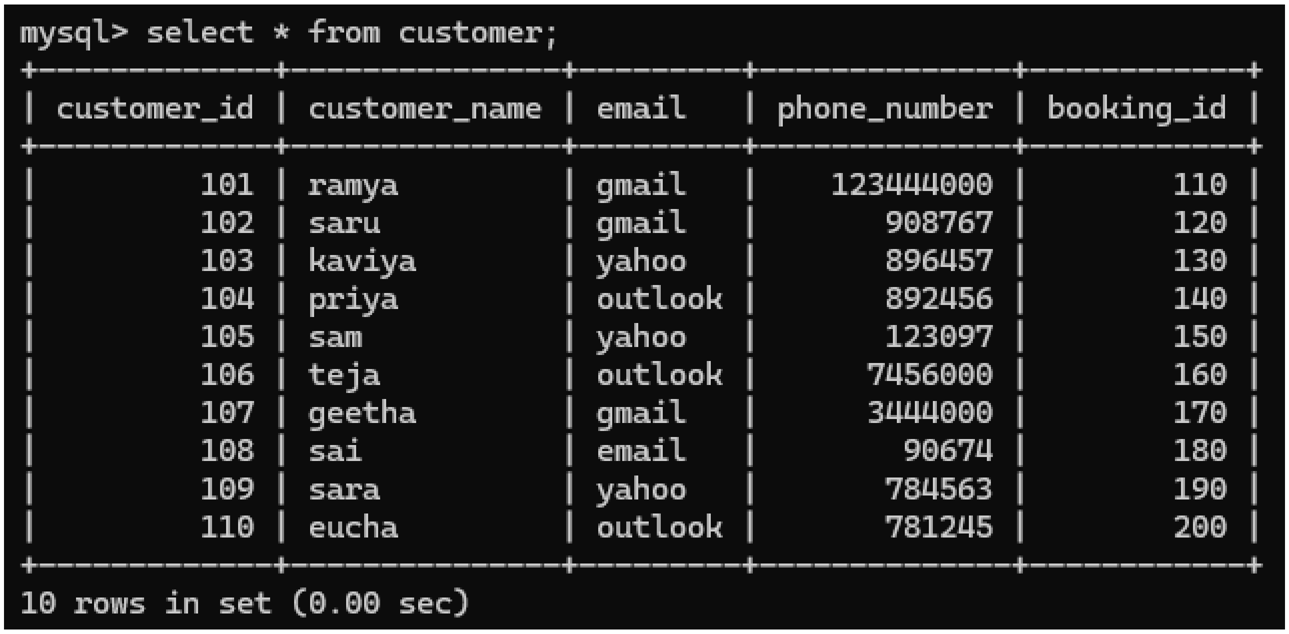
(106, 'teja', 'outlook', '7456000', 160),

(107, 'geetha', 'gmail', '3444000', 170),

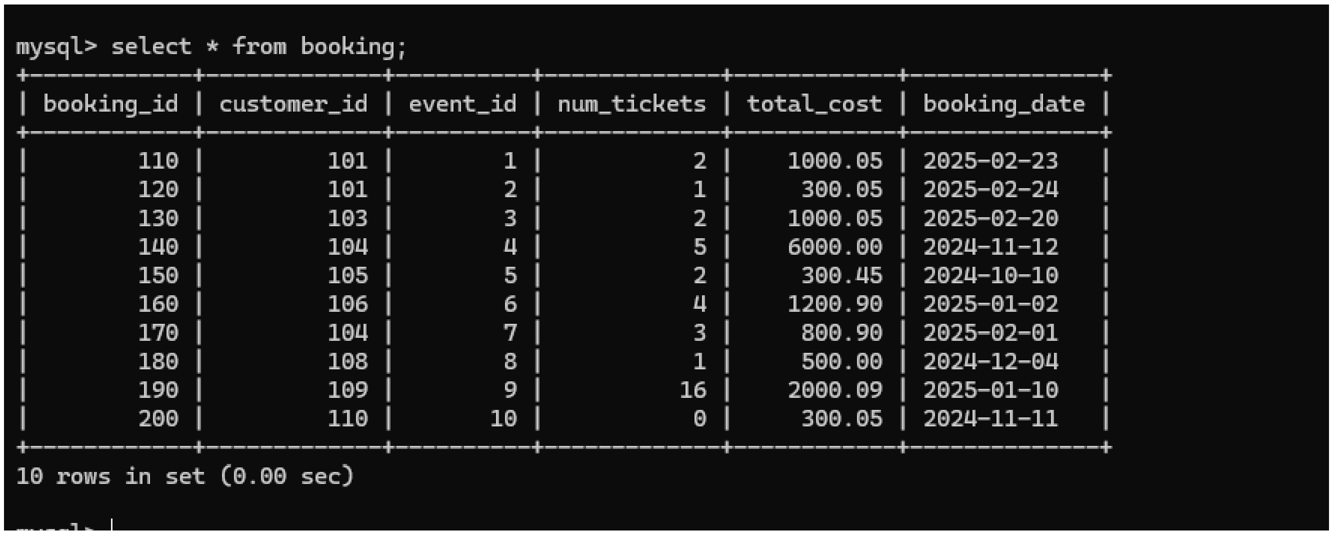
(108, 'sai', 'email', '90674', 180),

(109, 'sara', 'yahoo', '784563', 190),

(110, 'eucha', 'outlook', '781245', 200);


```
mysql> select * from customer;
+-------------+---------------+---------+--------------+------------+
| customer_id | customer_name | email   | phone_number | booking_id |
+-------------+---------------+---------+--------------+------------+
|         101 | ramya         | gmail   |    123444000 |        110 |
|         102 | saru          | gmail   |       908767 |        120 |
|         103 | kaviya        | yahoo   |       896457 |        130 |
|         104 | priya         | outlook |       892456 |        140 |
|         105 | sam           | yahoo   |       123097 |        150 |
|         106 | teja          | outlook |      7456000 |        160 |
|         107 | geetha        | gmail   |      3444000 |        170 |
|         108 | sai           | email   |        90674 |        180 |
|         109 | sara          | yahoo   |       784563 |        190 |
|         110 | eucha         | outlook |       781245 |        200 |
+-------------+---------------+---------+--------------+------------+
10 rows in set (0.00 sec)
```

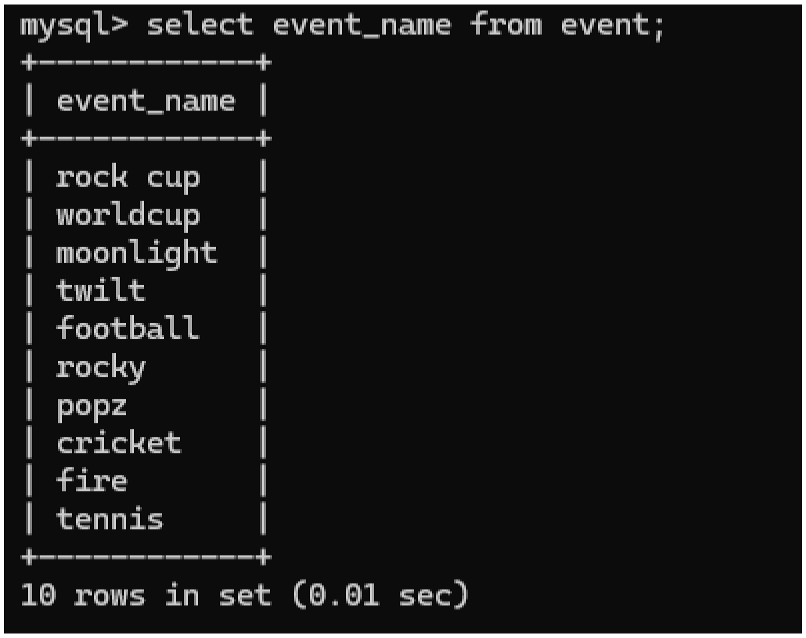
```
INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost,
booking_date) VALUES

(110, 101, 1, 2, 1000.05, '2025-02-23'),

(120, 101, 2, 1, 300.05, '2025-02-24'),

(130, 103, 3, 2, 1000.05, '2025-02-20'),

(140, 104, 4, 5, 6000.00, '2024-11-12'),

(150, 105, 5, 2, 300.45, '2024-10-10'),

(160, 106, 6, 4, 1200.90, '2025-01-02'),

(170, 104, 7, 3, 800.90, '2025-02-01'),

(180, 108, 8, 1, 500.00, '2024-12-04'),

(190, 109, 9, 16, 2000.09, '2025-01-10'),

(200, 110, 10, 0, 300.05, '2024-11-11');
```



```
mysql> select * from booking;
+------------+-------------+----------+-------------+------------+--------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
+------------+-------------+----------+-------------+------------+--------------+
|        110 |         101 |        1 |           2 |    1000.05 | 2025-02-23   |
|        120 |         101 |        2 |           1 |     300.05 | 2025-02-24   |
|        130 |         103 |        3 |           2 |    1000.05 | 2025-02-20   |
|        140 |         104 |        4 |           5 |    6000.00 | 2024-11-12   |
|        150 |         105 |        5 |           2 |     300.45 | 2024-10-10   |
|        160 |         106 |        6 |           4 |    1200.90 | 2025-01-02   |
|        170 |         104 |        7 |           3 |     800.90 | 2025-02-01   |
|        180 |         108 |        8 |           1 |     500.00 | 2024-12-04   |
|        190 |         109 |        9 |          16 |    2000.09 | 2025-01-10   |
|        200 |         110 |       10 |           0 |     300.05 | 2024-11-11   |
+------------+-------------+----------+-------------+------------+--------------+
10 rows in set (0.00 sec)
```

2. Write a SQL query to list all Events.

select event_name from event;



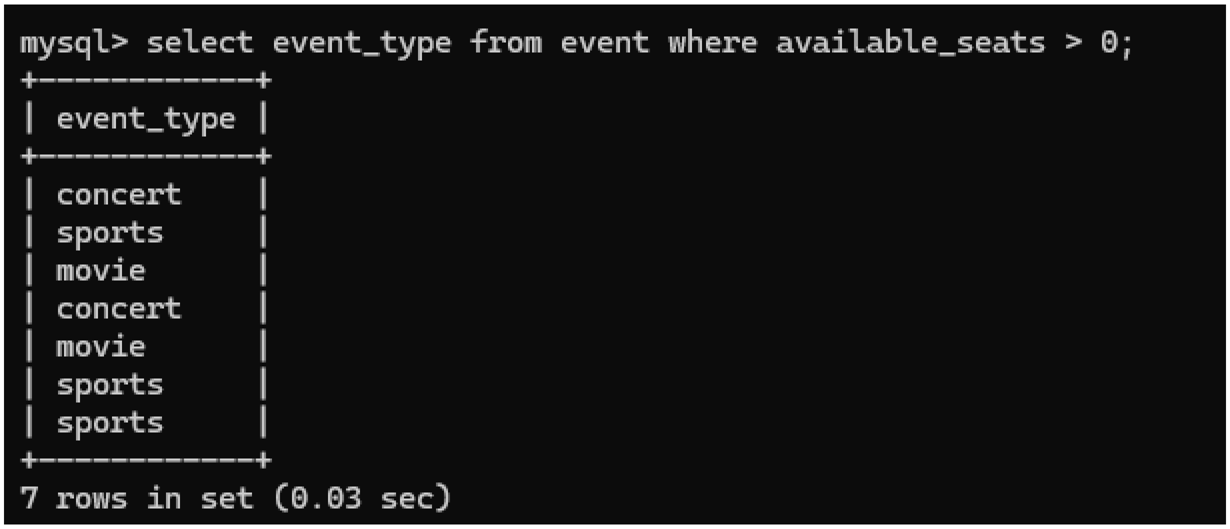
```
mysql> select event_name from event;
+------------+
| event_name |
+------------+
| rock cup   |
| worldcup   |
| moonlight  |
| twilt      |
| football   |
| rocky      |
| popz       |
| cricket    |
| fire       |
| tennis     |
+------------+
10 rows in set (0.01 sec)
```

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

select event_type from event where available_seats > 0;

or

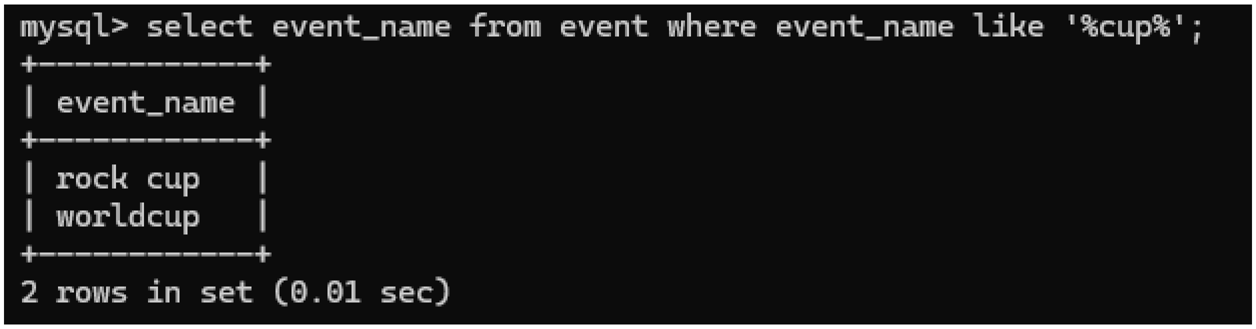
select event_name from event where available_seats > 0;



```
mysql> select event_type from event where available_seats > 0;
+------------+
| event_type |
+------------+
| concert    |
| sports     |
| movie      |
| concert    |
| movie      |
| sports     |
| sports     |
+------------+
7 rows in set (0.03 sec)
```

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

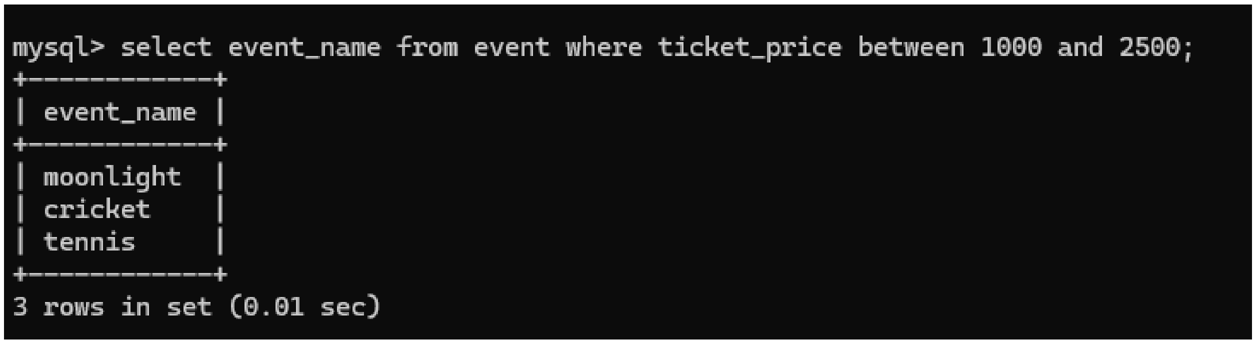
select event_name from event where event_name like '%cup%';



```
mysql> select event_name from event where event_name like '%cup%';
+------------+
| event_name |
+------------+
| rock cup   |
| worldcup   |
+------------+
2 rows in set (0.01 sec)
```

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

select event_name from event where ticket_price between 1000 and 2500;

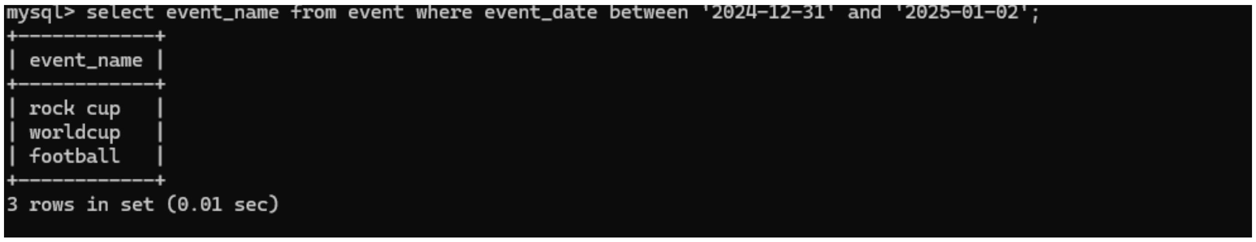


```
mysql> select event_name from event where ticket_price between 1000 and 2500;
+------------+
| event_name |
+------------+
| moonlight  |
| cricket    |
| tennis     |
+------------+
3 rows in set (0.01 sec)
```

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

select event_name from event where event_date > '2024-12-31' and event_date < '2025-01-02';

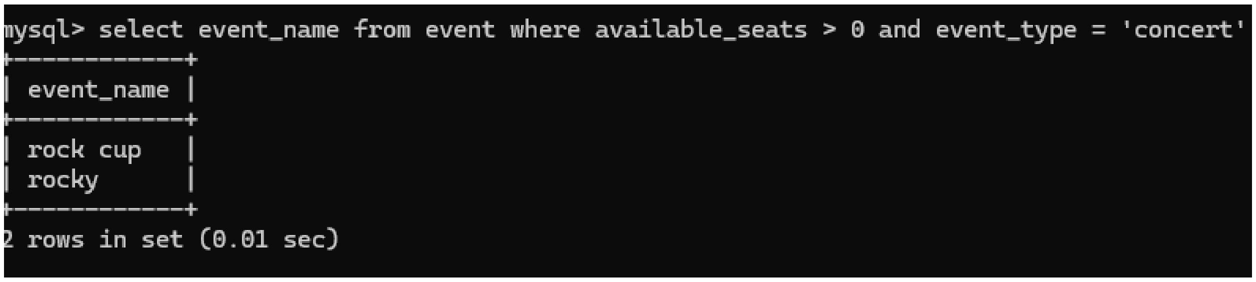
select event_name from event where event_date between '2024-12-31' and '2025-01-02';



```
mysql> select event_name from event where event_date between '2024-12-31' and '2025-01-02';
+------------+
| event_name |
+------------+
| rock cup   |
| worldcup   |
| football   |
+------------+
3 rows in set (0.01 sec)
```

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

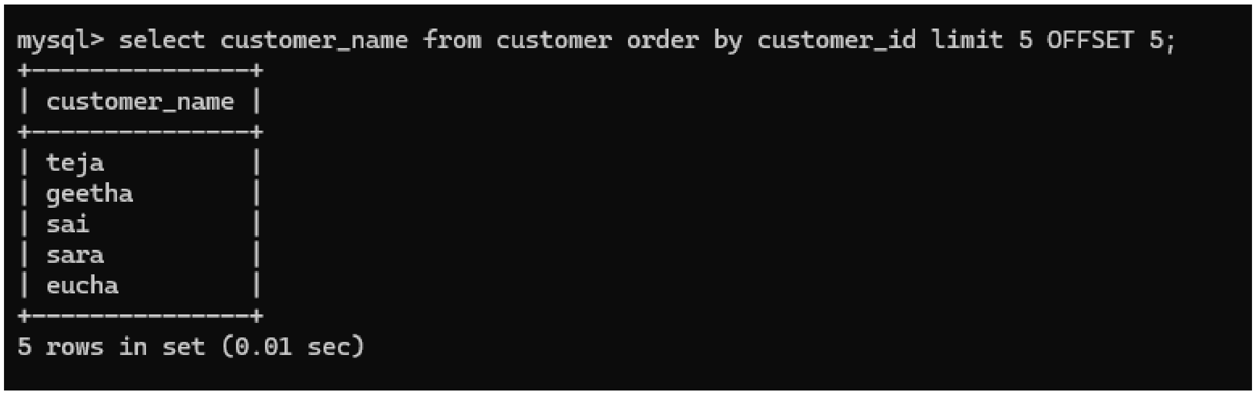
select event_name from event where available_seats > 0 and event_type = 'concert';.



```
ysql> select event_name from event where available_seats > 0 and event_type = 'concert'
+------------+
| event_name |
+------------+
| rock cup   |
| rocky      |
+------------+
 rows in set (0.01 sec)
```

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

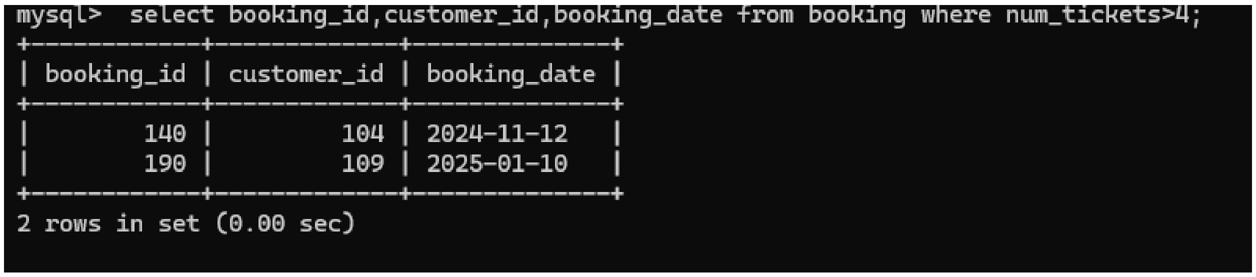
select customer_name from customer order by customer_id limit 5 OFFSET 5;



```
mysql> select customer_name from customer order by customer_id limit 5 OFFSET 5;
+---------------+
| customer_name |
+---------------+
| teja          |
| geetha        |
| sai           |
| sara          |
| eucha         |
+---------------+
5 rows in set (0.01 sec)
```

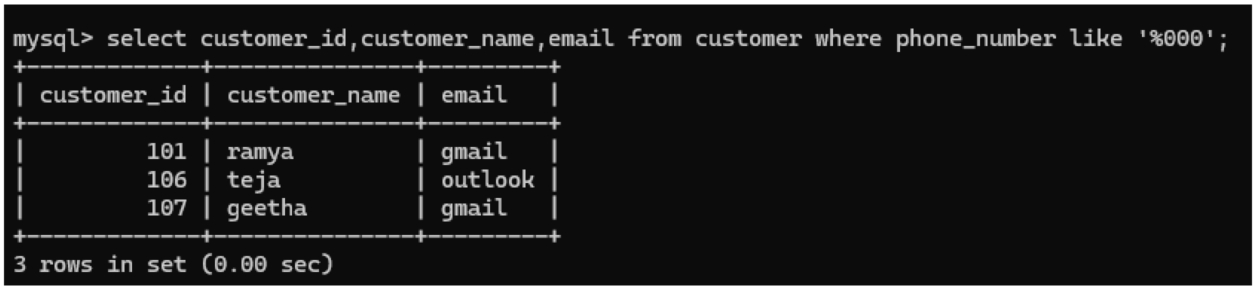
9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.

> select booking_id,customer_id,booking_date from booking where num_tickets>4;


```
mysql>  select booking_id,customer_id,booking_date from booking where num_tickets>4;
+------------+-------------+--------------+
| booking_id | customer_id | booking_date |
+------------+-------------+--------------+
|        140 |         104 | 2024-11-12   |
|        190 |         109 | 2025-01-10   |
+------------+-------------+--------------+
2 rows in set (0.00 sec)
```

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

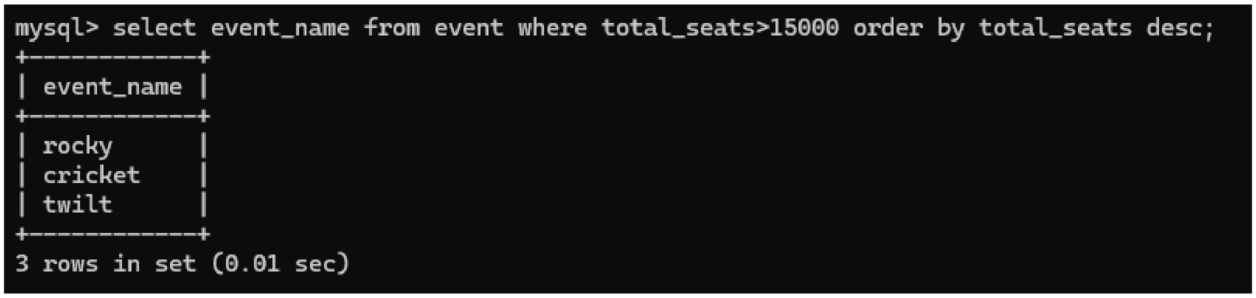
select customer_id,customer_name,email from customer where phone_number like '%000';


```
mysql> select customer_id,customer_name,email from customer where phone_number like '%000';
+-------------+---------------+---------+
| customer_id | customer_name | email   |
+-------------+---------------+---------+
|         101 | ramya         | gmail   |
|         106 | teja          | outlook |
|         107 | geetha        | gmail   |
+-------------+---------------+---------+
3 rows in set (0.00 sec)
```

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

select event_name from event where total_seats>15000 order by total_seats asc;

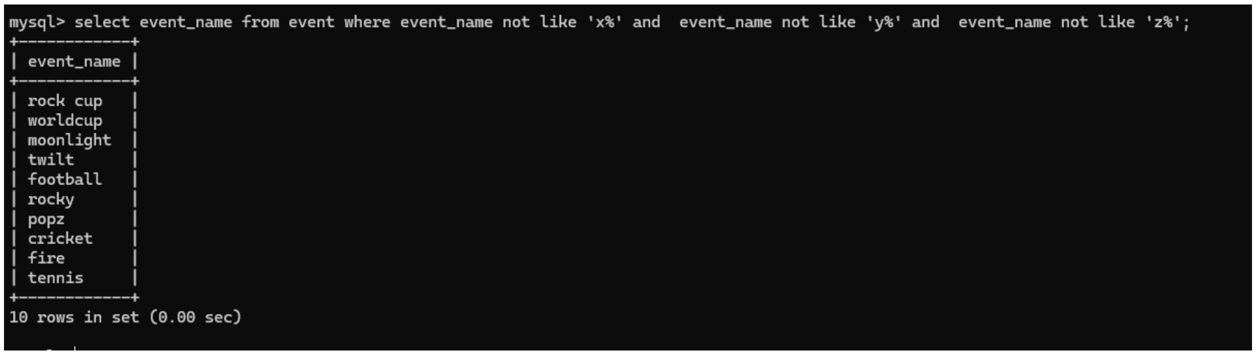
select event_name from event where total_seats>15000 order by total_seats desc;


```
mysql> select event_name from event where total_seats>15000 order by total_seats desc;
+------------+
| event_name |
+------------+
| rocky      |
| cricket    |
| twilt      |
+------------+
3 rows in set (0.01 sec)
```

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

select event_name from event where event_name not like 'x%' or event_name not like 'y%' or event_name not like 'z%';

select event_name from event where event_name not like 'x%' and event_name not like 'y%' and event_name not like 'z%';

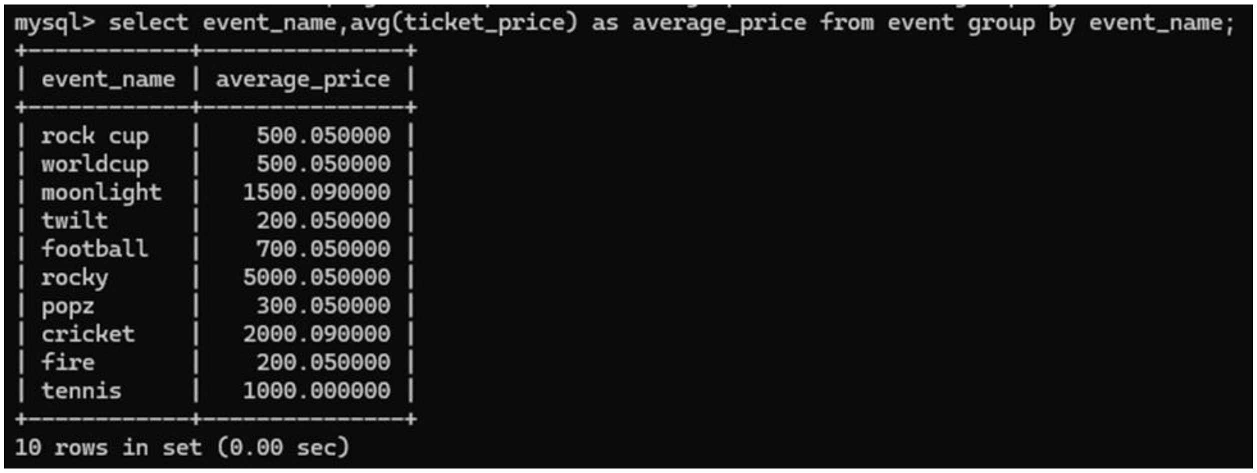


```
mysql> select event_name from event where event_name not like 'x%' and  event_name not like 'y%' and  event_name not like 'z%';
+------------+
| event_name |
+------------+
| rock cup   |
| worldcup   |
| moonlight  |
| twilt      |
| football   |
| rocky      |
| popz       |
| cricket    |
| fire       |
| tennis     |
+------------+
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.

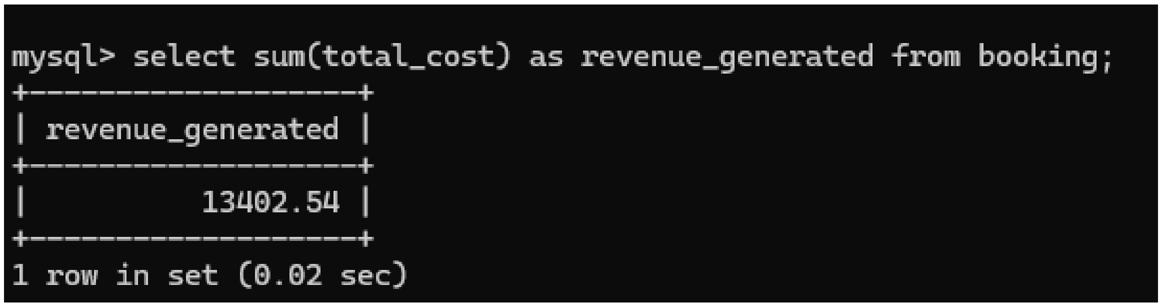
select event_name,avg(ticket_price) as average_price from event group by event_name;



```
mysql> select event_name,avg(ticket_price) as average_price from event group by event_name;
+------------+---------------+
| event_name | average_price |
+------------+---------------+
| rock cup   |    500.050000 |
| worldcup   |    500.050000 |
| moonlight  |   1500.090000 |
| twilt      |    200.050000 |
| football   |    700.050000 |
| rocky      |   5000.050000 |
| popz       |    300.050000 |
| cricket    |   2000.090000 |
| fire       |    200.050000 |
| tennis     |   1000.000000 |
+------------+---------------+
10 rows in set (0.00 sec)
```

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

select sum(total_cost) as revenue_generated from booking;

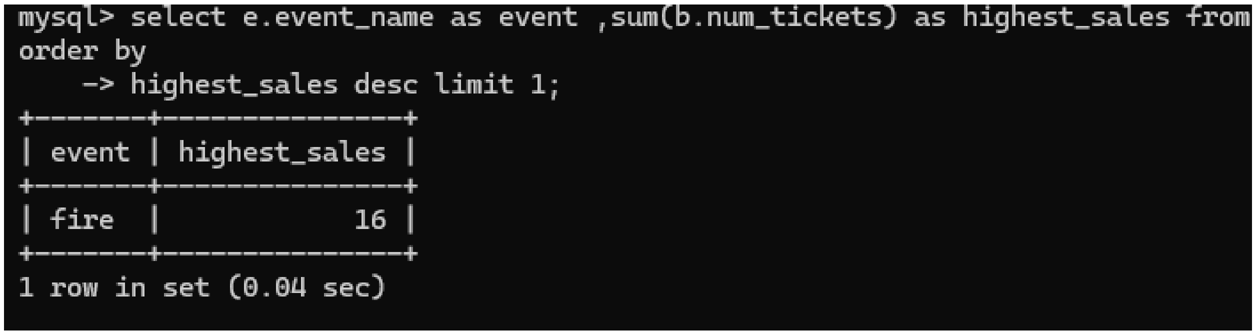


```
mysql> select sum(total_cost) as revenue_generated from booking;
+-------------------+
| revenue_generated |
+-------------------+
|          13402.54 |
+-------------------+
1 row in set (0.02 sec)
```

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

select e.event_name as event ,sum(b.num_tickets) as highest_sales from event e join booking b on e.event_id=b.event_id group by e.event_name order by

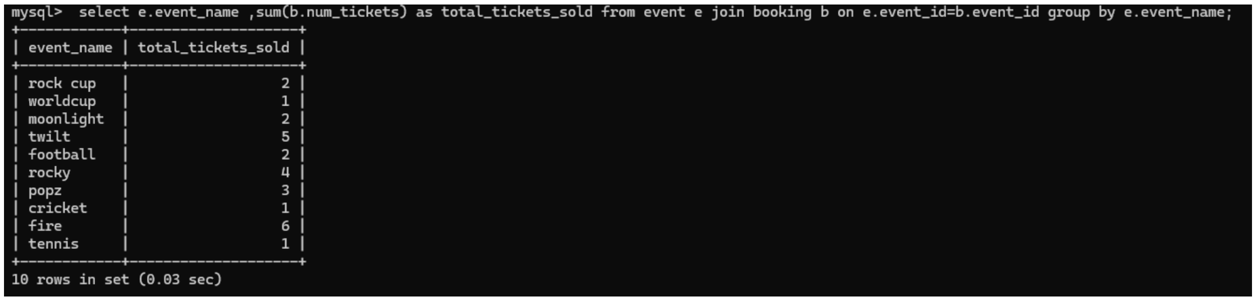
highest_sales desc limit 1;



```
mysql> select e.event_name as event ,sum(b.num_tickets) as highest_sales from
order by
    -> highest_sales desc limit 1;
+-------+---------------+
| event | highest_sales |
+-------+---------------+
| fire  |            16 |
+-------+---------------+
1 row in set (0.04 sec)
```

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

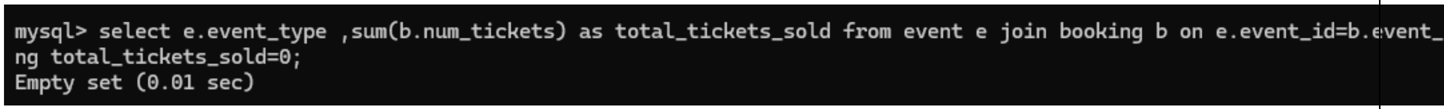
select e.event_name ,sum(b.num_tickets) as total_tickets_sold from event e join booking b on e.event_id=b.event_id group by e.event_name;



```
mysql>  select 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_name;
+------------+--------------------+
| event_name | total_tickets_sold |
+------------+--------------------+
| rock cup   |                  2 |
| worldcup   |                  1 |
| moonlight  |                  2 |
| twilt      |                  5 |
| football   |                  2 |
| rocky      |                  4 |
| popz       |                  3 |
| cricket    |                  1 |
| fire       |                  6 |
| tennis     |                  1 |
+------------+--------------------+
10 rows in set (0.03 sec)
```

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

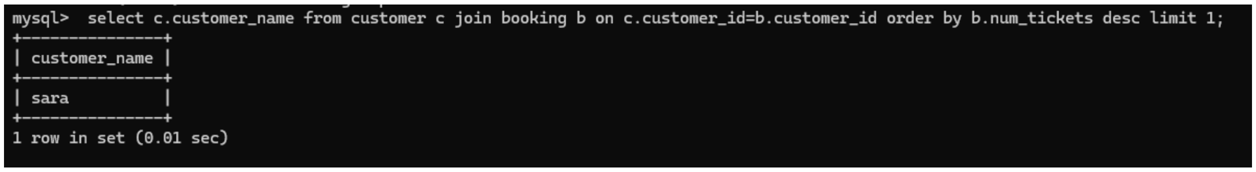
select e.event_type ,sum(b.num_tickets) as total_tickets_sold from event e join booking b on e.event_id=b.event_id group by e.event_type having total_tickets_sold=0;



```
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_
ng total_tickets_sold=0;
Empty set (0.01 sec)
```

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

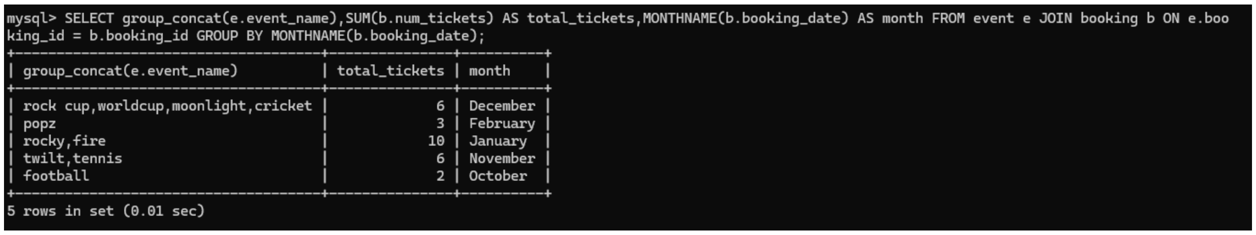
select c.customer_name from customer c join booking b on c.customer_id=b.customer_id order by b.num_tickets desc limit 1;



```
mysql>  select c.customer_name from customer c join booking b on c.customer_id=b.customer_id order by b.num_tickets desc limit 1;
+---------------+
| customer_name |
+---------------+
| sara          |
+---------------+
1 row in set (0.01 sec)
```

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

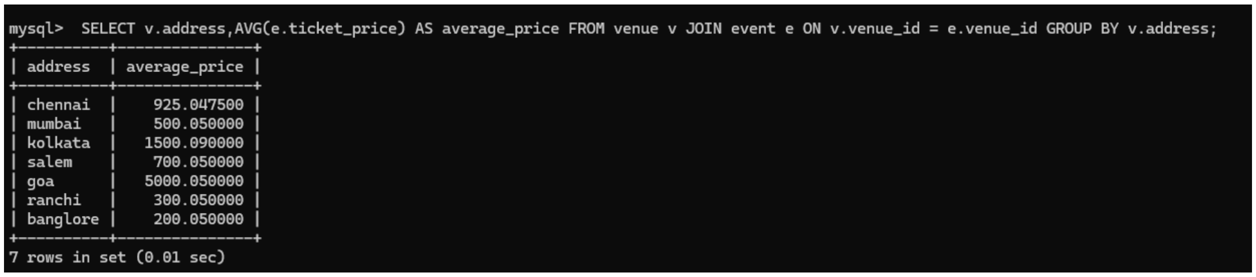
SELECT group_concat(e.event_name),SUM(b.num_tickets) AS total_tickets,MONTHNAME(b.booking_date) AS month FROM event e JOIN booking b ON e.booking_id = b.booking_id GROUP BY MONTHNAME(b.booking_date);



```
mysql> SELECT group_concat(e.event_name),SUM(b.num_tickets) AS total_tickets,MONTHNAME(b.booking_date) AS month FROM event e JOIN booking b ON e.boo
king_id = b.booking_id GROUP BY MONTHNAME(b.booking_date);
+-----------------------------------+---------------+----------+
| group_concat(e.event_name)        | total_tickets | month    |
+-----------------------------------+---------------+----------+
| rock cup,worldcup,moonlight,cricket |             6 | December |
| popz                              |             3 | February |
| rocky,fire                        |            10 | January  |
| twilt,tennis                      |             6 | November |
| football                          |             2 | October  |
+-----------------------------------+---------------+----------+
5 rows in set (0.01 sec)
```

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

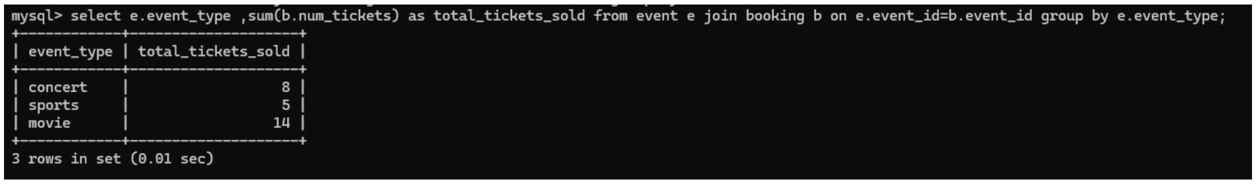
SELECT v.address,AVG(e.ticket_price) AS average_price FROM venue v JOIN event e ON v.venue_id = e.venue_id GROUP BY v.address;



```
mysql>  SELECT v.address,AVG(e.ticket_price) AS average_price FROM venue v JOIN event e ON v.venue_id = e.venue_id GROUP BY v.address;
+----------+---------------+
| address  | average_price |
+----------+---------------+
| chennai  |    925.047500 |
| mumbai   |    500.050000 |
| kolkata  |   1500.090000 |
| salem    |    700.050000 |
| goa      |   5000.050000 |
| ranchi   |    300.050000 |
| banglore |    200.050000 |
+----------+---------------+
7 rows in set (0.01 sec)
```

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

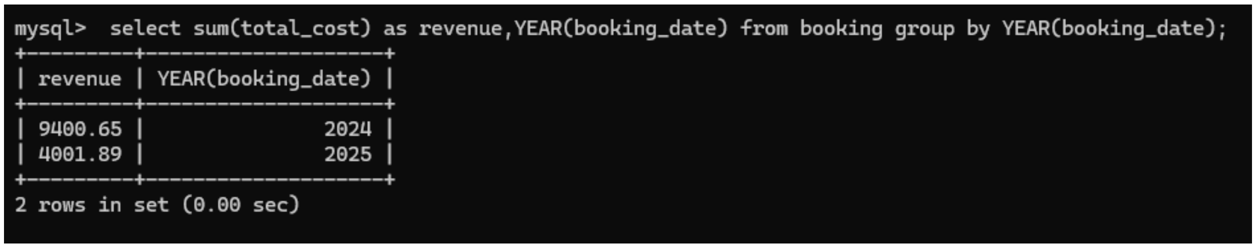
Select e.event_type ,sum(b.num_tickets) as total_tickets_sold from event e join booking b on e.event_id=b.event_id group by e.event_type;



```
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;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| concert    |                  8 |
| sports     |                  5 |
| movie      |                 14 |
+------------+--------------------+
3 rows in set (0.01 sec)
```

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

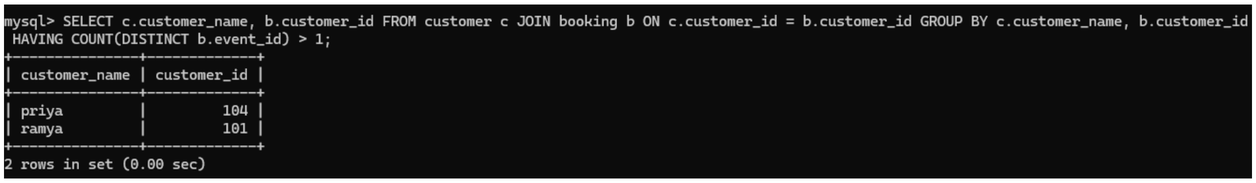
select sum(total_cost) as revenue,YEAR(booking_date) from booking group by YEAR(booking_date);



```
mysql>  select sum(total_cost) as revenue,YEAR(booking_date) from booking group by YEAR(booking_date);
+---------+--------------------+
| revenue | YEAR(booking_date) |
+---------+--------------------+
| 9400.65 |               2024 |
| 4001.89 |               2025 |
+---------+--------------------+
2 rows in set (0.00 sec)
```

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

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

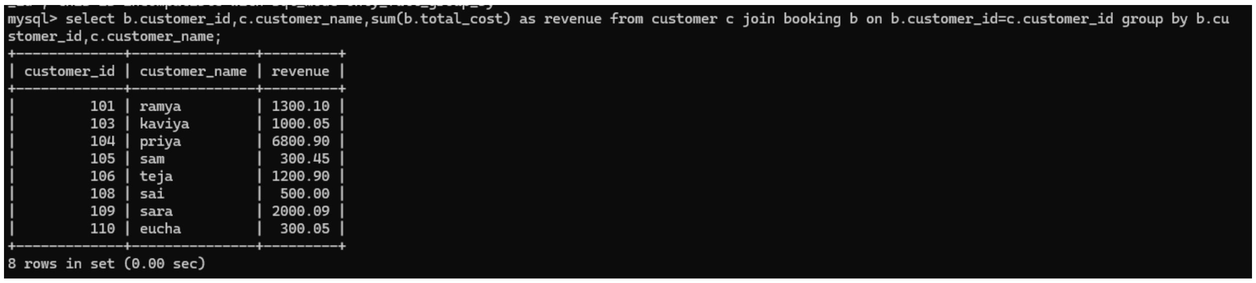


```
mysql> SELECT c.customer_name, b.customer_id FROM customer c JOIN booking b ON c.customer_id = b.customer_id GROUP BY c.customer_name, b.customer_id
 HAVING COUNT(DISTINCT b.event_id) > 1;
+---------------+-------------+
| customer_name | customer_id |
+---------------+-------------+
| priya         |         104 |
| ramya         |         101 |
+---------------+-------------+
2 rows in set (0.00 sec)
```

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

select b.customer_id,c.customer_name,sum(b.total_cost) as revenue from customer c join booking b on b.customer_id=c.customer_id group by b.cu

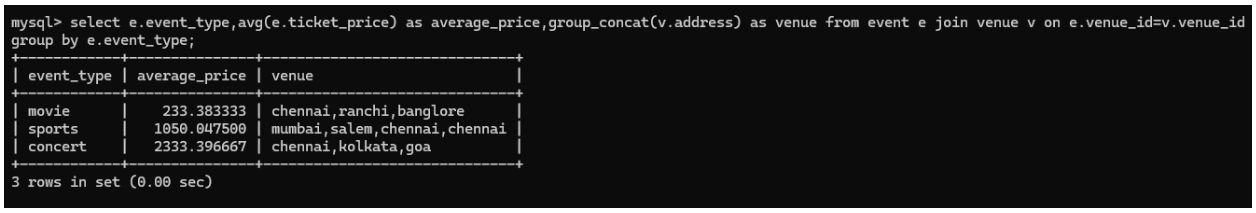
stomer_id,c.customer_name;



```
mysql> select b.customer_id,c.customer_name,sum(b.total_cost) as revenue from customer c join booking b on b.customer_id=c.customer_id group by b.cu
stomer_id,c.customer_name;
+-------------+---------------+---------+
| customer_id | customer_name | revenue |
+-------------+---------------+---------+
|         101 | ramya         | 1300.10 |
|         103 | kaviya        | 1000.05 |
|         104 | priya         | 6800.90 |
|         105 | sam           |  300.45 |
|         106 | teja          | 1200.90 |
|         108 | sai           |  500.00 |
|         109 | sara          | 2000.09 |
|         110 | eucha         |  300.05 |
+-------------+---------------+---------+
8 rows in set (0.00 sec)
```

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

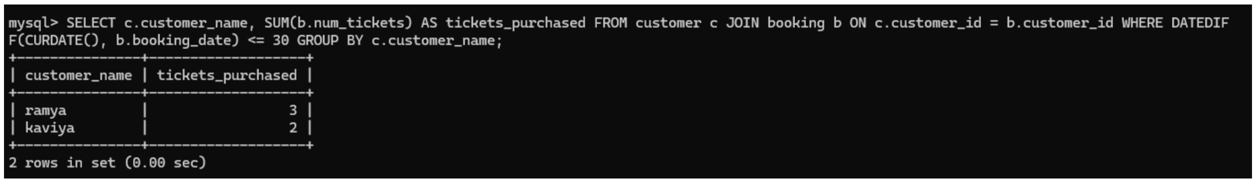
select e.event_type,avg(e.ticket_price) as average_price,group_concat(v.address) as venue from event e join venue v on e.venue_id=v.venue_id group by e.event_type;



```
mysql> select e.event_type,avg(e.ticket_price) as average_price,group_concat(v.address) as venue from event e join venue v on e.venue_id=v.venue_id
group by e.event_type;
+------------+---------------+-------------------------------+
| event_type | average_price | venue                         |
+------------+---------------+-------------------------------+
| movie      |    233.383333 | chennai,ranchi,banglore       |
| sports     |   1050.047500 | mumbai,salem,chennai,chennai  |
| concert    |   2333.396667 | chennai,kolkata,goa           |
+------------+---------------+-------------------------------+
3 rows in set (0.00 sec)
```

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

SELECT c.customer_name, SUM(b.num_tickets) AS tickets_purchased FROM customer c JOIN booking b ON c.customer_id = b.customer_id WHERE DATEDIFF(CURDATE(), b.booking_date) <= 30 GROUP BY c.customer_name;



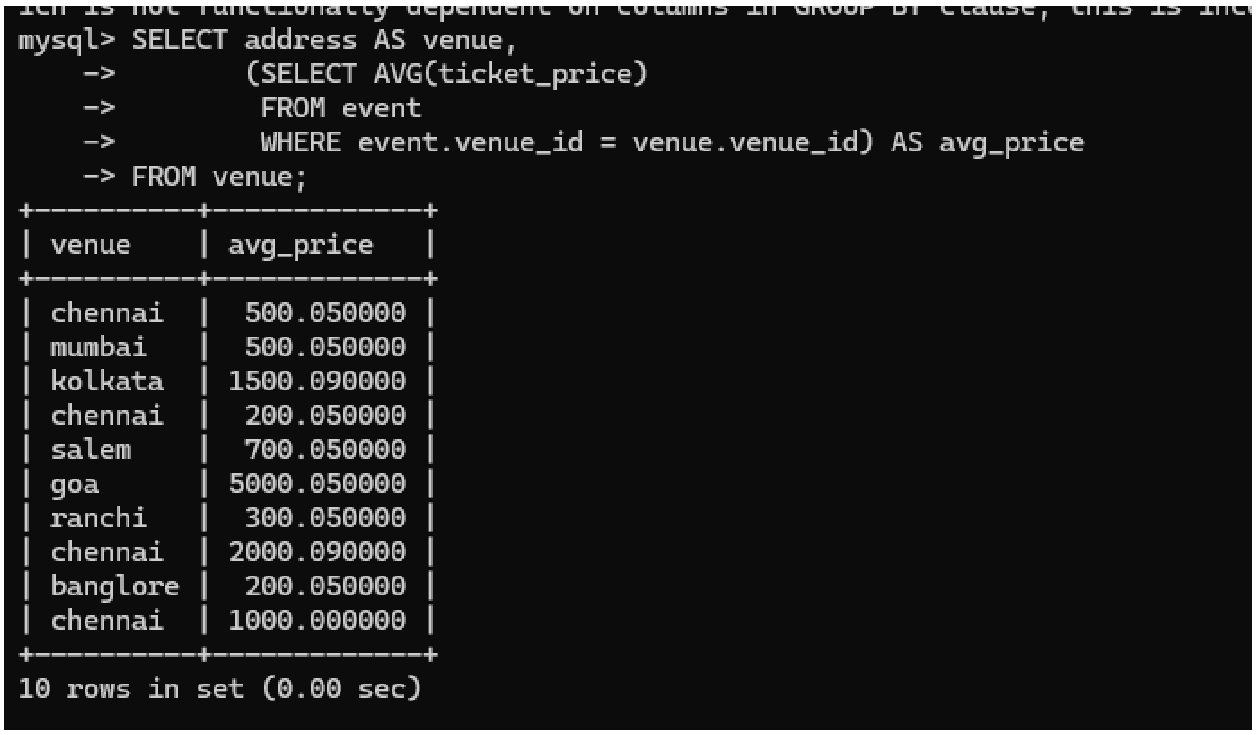
```
mysql> SELECT c.customer_name, SUM(b.num_tickets) AS tickets_purchased FROM customer c JOIN booking b ON c.customer_id = b.customer_id WHERE DATEDIF
F(CURDATE(), b.booking_date) <= 30 GROUP BY c.customer_name;
+---------------+-------------------+
| customer_name | tickets_purchased |
+---------------+-------------------+
| ramya         |                 3 |
| kaviya        |                 2 |
+---------------+-------------------+
2 rows in set (0.00 sec)
```

Tasks 4: Subquery and its types

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

```
SELECT address AS venue,
    ->       (SELECT AVG(ticket_price)
    ->        FROM event
    ->        WHERE event.venue_id = venue.venue_id) AS avg_price
```

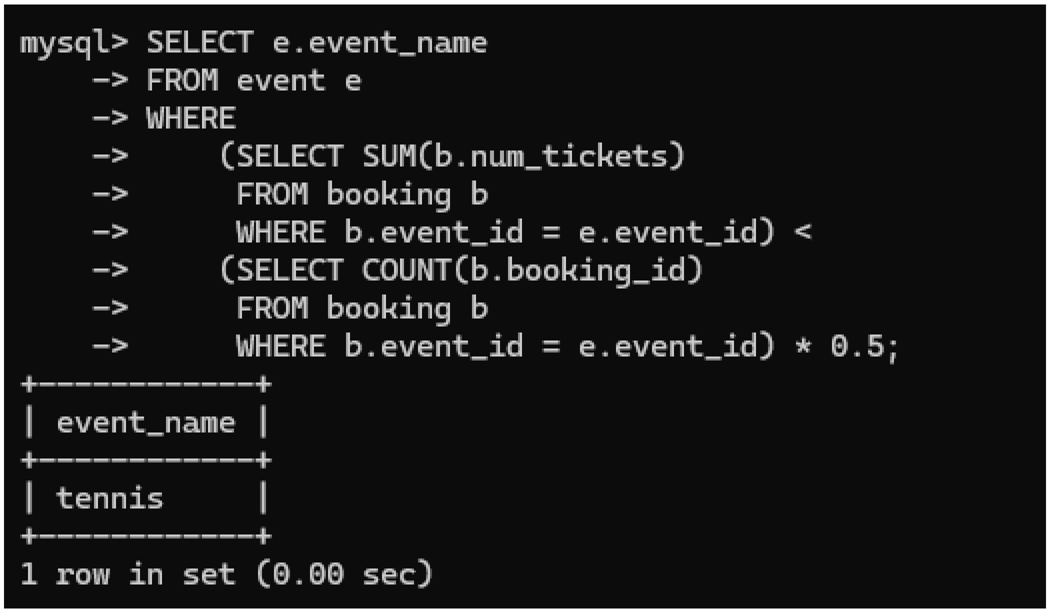
-> FROM venue;



```
ich is not functionally dependent on columns in GROUP BY clause; this is inc
mysql> SELECT address AS venue,
    ->        (SELECT AVG(ticket_price)
    ->         FROM event
    ->         WHERE event.venue_id = venue.venue_id) AS avg_price
    -> FROM venue;
+----------+-------------+
| venue    | avg_price   |
+----------+-------------+
| chennai  |  500.050000 |
| mumbai   |  500.050000 |
| kolkata  | 1500.090000 |
| chennai  |  200.050000 |
| salem    |  700.050000 |
| goa      | 5000.050000 |
| ranchi   |  300.050000 |
| chennai  | 2000.090000 |
| banglore |  200.050000 |
| chennai  | 1000.000000 |
+----------+-------------+
10 rows in set (0.00 sec)
```

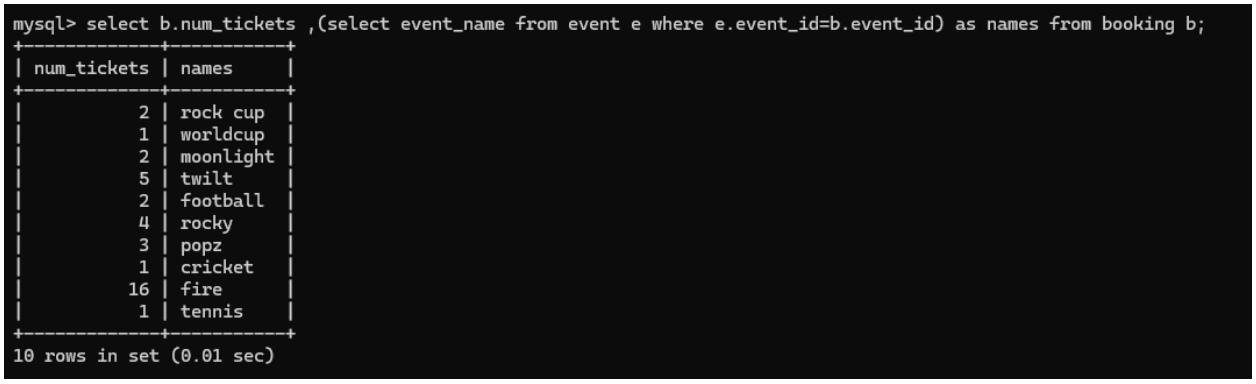
2.Find Events with More Than 50% of Tickets Sold using subquery.

```
SELECT e.event_name

FROM event e

WHERE

    (SELECT SUM(b.num_tickets)

     FROM booking b

     WHERE b.event_id = e.event_id) <

    (SELECT COUNT(b.booking_id)

     FROM booking b

     WHERE b.event_id = e.event_id) * 0.5;
```


```
mysql> SELECT e.event_name
    -> FROM event e
    -> WHERE
    ->     (SELECT SUM(b.num_tickets)
    ->      FROM booking b
    ->      WHERE b.event_id = e.event_id) <
    ->     (SELECT COUNT(b.booking_id)
    ->      FROM booking b
    ->      WHERE b.event_id = e.event_id) * 0.5;
+------------+
| event_name |
+------------+
| tennis     |
+------------+
1 row in set (0.00 sec)
```

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

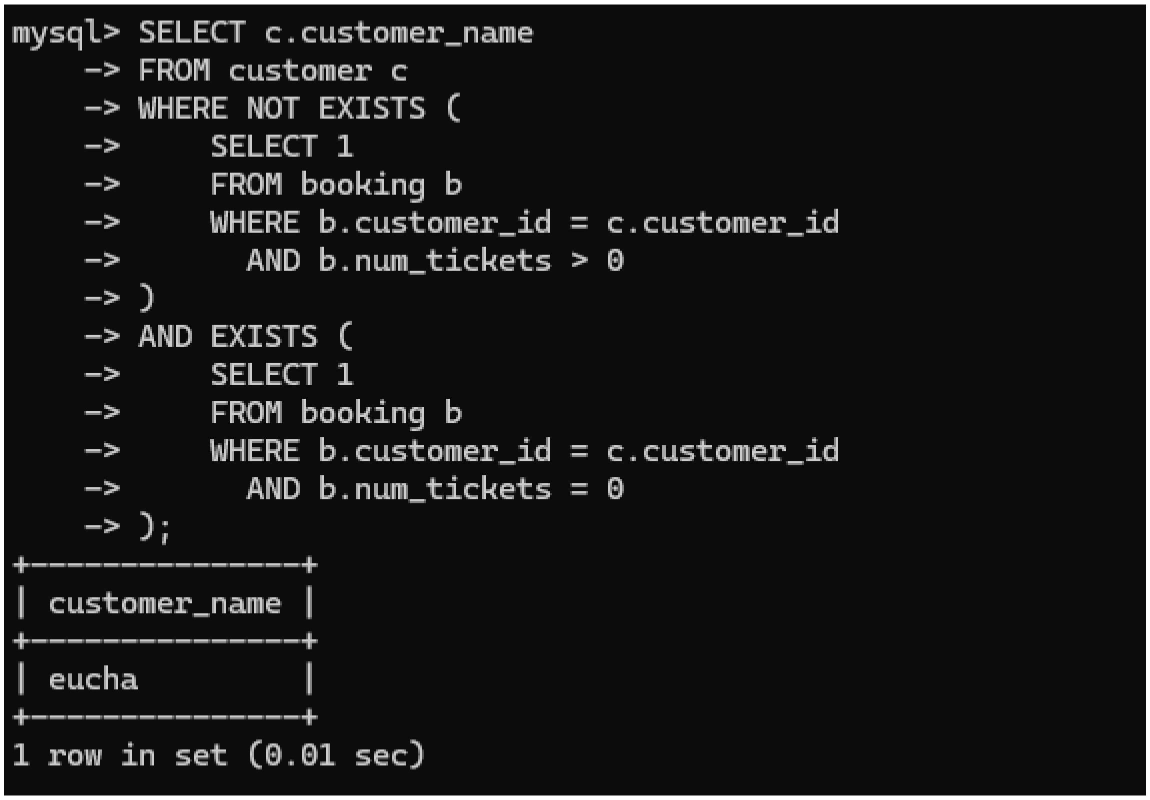
select b.num_tickets ,(select event_name from event e where e.event_id=b.event_id) as names from booking b


```
mysql> select b.num_tickets ,(select event_name from event e where e.event_id=b.event_id) as names from booking b;
+-------------+-----------+
| num_tickets | names     |
+-------------+-----------+
|           2 | rock cup  |
|           1 | worldcup  |
|           2 | moonlight |
|           5 | twilt     |
|           2 | football  |
|           4 | rocky     |
|           3 | popz      |
|           1 | cricket   |
|          16 | fire      |
|           1 | tennis    |
+-------------+-----------+
10 rows in set (0.01 sec)
```

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

```
SELECT c.customer_name

FROM customer c

WHERE NOT EXISTS (

   SELECT 1
```

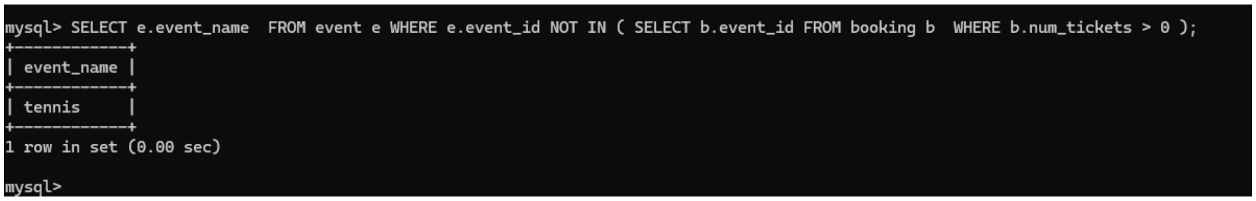
```
    FROM booking b
    WHERE b.customer_id = c.customer_id
      AND b.num_tickets > 0
)
AND EXISTS (
    SELECT 1
    FROM booking b
    WHERE b.customer_id = c.customer_id
      AND b.num_tickets = 0
```



```
mysql> SELECT c.customer_name
    -> FROM customer c
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM booking b
    ->     WHERE b.customer_id = c.customer_id
    ->       AND b.num_tickets > 0
    -> )
    -> AND EXISTS (
    ->     SELECT 1
    ->     FROM booking b
    ->     WHERE b.customer_id = c.customer_id
    ->       AND b.num_tickets = 0
    -> );
+---------------+
| customer_name |
+---------------+
| eucha         |
+---------------+
1 row in set (0.01 sec)
```

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

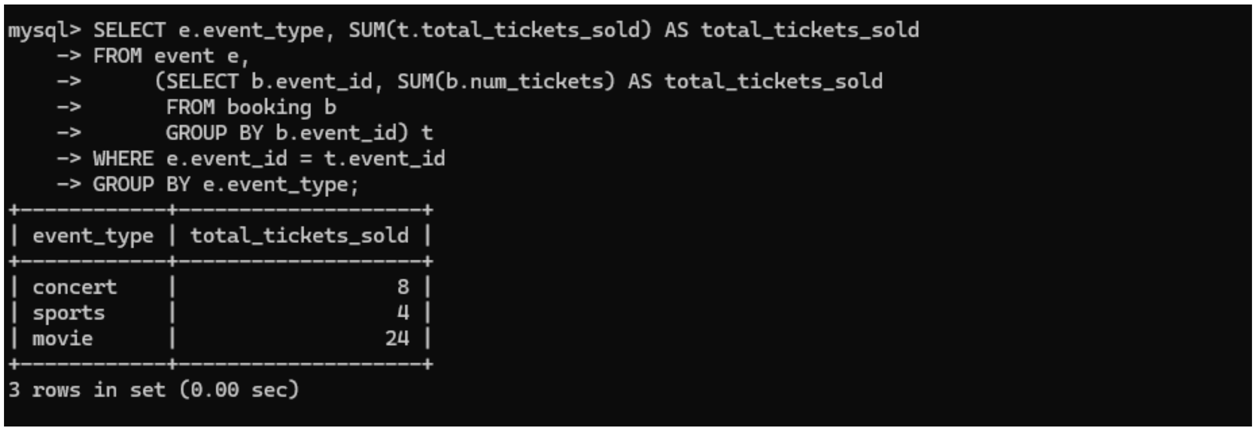
SELECT e.event_name FROM event e WHERE e.event_id NOT IN (SELECT b.event_id FROM booking b WHERE b.num_tickets > 0);



```
mysql> SELECT e.event_name  FROM event e WHERE e.event_id NOT IN ( SELECT b.event_id FROM booking b  WHERE b.num_tickets > 0 );
+------------+
| event_name |
+------------+
| tennis     |
+------------+
1 row in set (0.00 sec)

mysql>
```

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

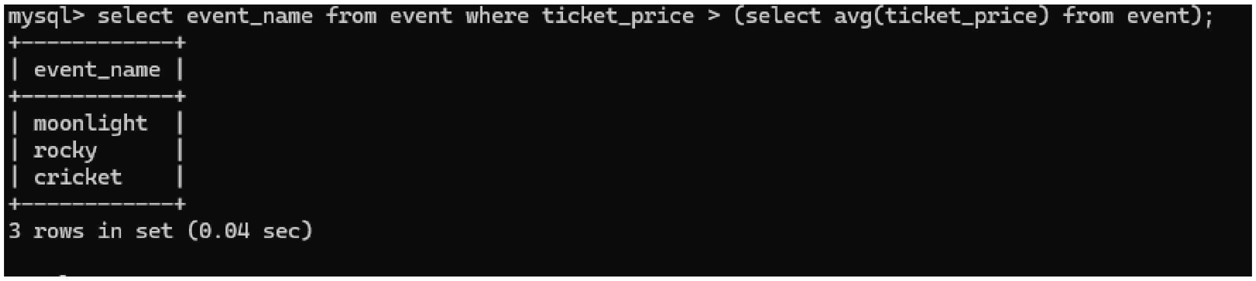
```
ELECT e.event_type, SUM(t.total_tickets_sold) AS total_tickets_sold

    -> FROM event e,

    ->      (SELECT b.event_id, SUM(b.num_tickets) AS total_tickets_sold

    ->       FROM booking b

    ->       GROUP BY b.event_id) t

    -> WHERE e.event_id = t.event_id

    -> GROUP BY e.event_type;
```



```
mysql> SELECT e.event_type, SUM(t.total_tickets_sold) AS total_tickets_sold
    -> FROM event e,
    ->      (SELECT b.event_id, SUM(b.num_tickets) AS total_tickets_sold
    ->       FROM booking b
    ->       GROUP BY b.event_id) t
    -> WHERE e.event_id = t.event_id
    -> GROUP BY e.event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| concert    |                  8 |
| sports     |                  4 |
| movie      |                 24 |
+------------+--------------------+
3 rows in set (0.00 sec)
```

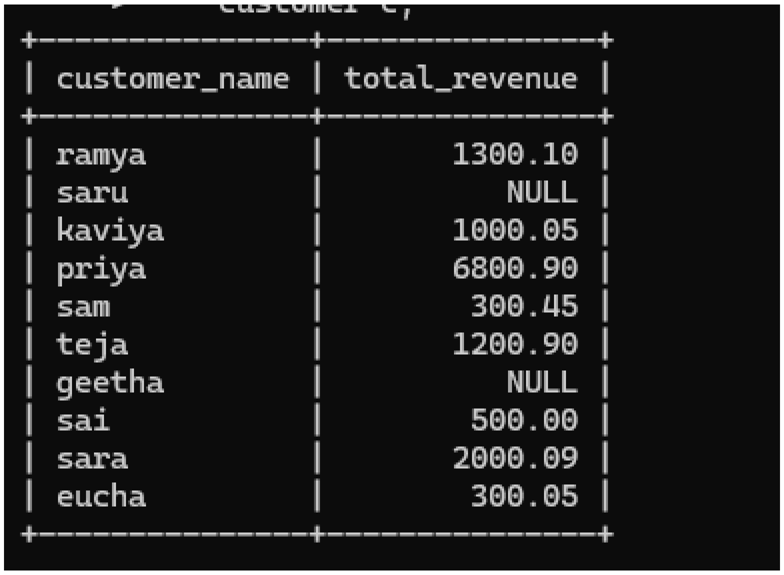
7. Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause.

```
select event_name from event where ticket_price > (select avg(ticket_price) from event);
```


```
mysql> select event_name from event where ticket_price > (select avg(ticket_price) from event);
+------------+
| event_name |
+------------+
| moonlight  |
| rocky      |
| cricket    |
+------------+
3 rows in set (0.04 sec)
```

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

```
SELECT
    c.customer_name,
    (
        SELECT
            SUM(b.total_cost) -- Use SUM to aggregate
        FROM
            booking b
        WHERE
            b.customer_id = c.customer_id
    ) AS total_revenue
FROM
    customer c;
```

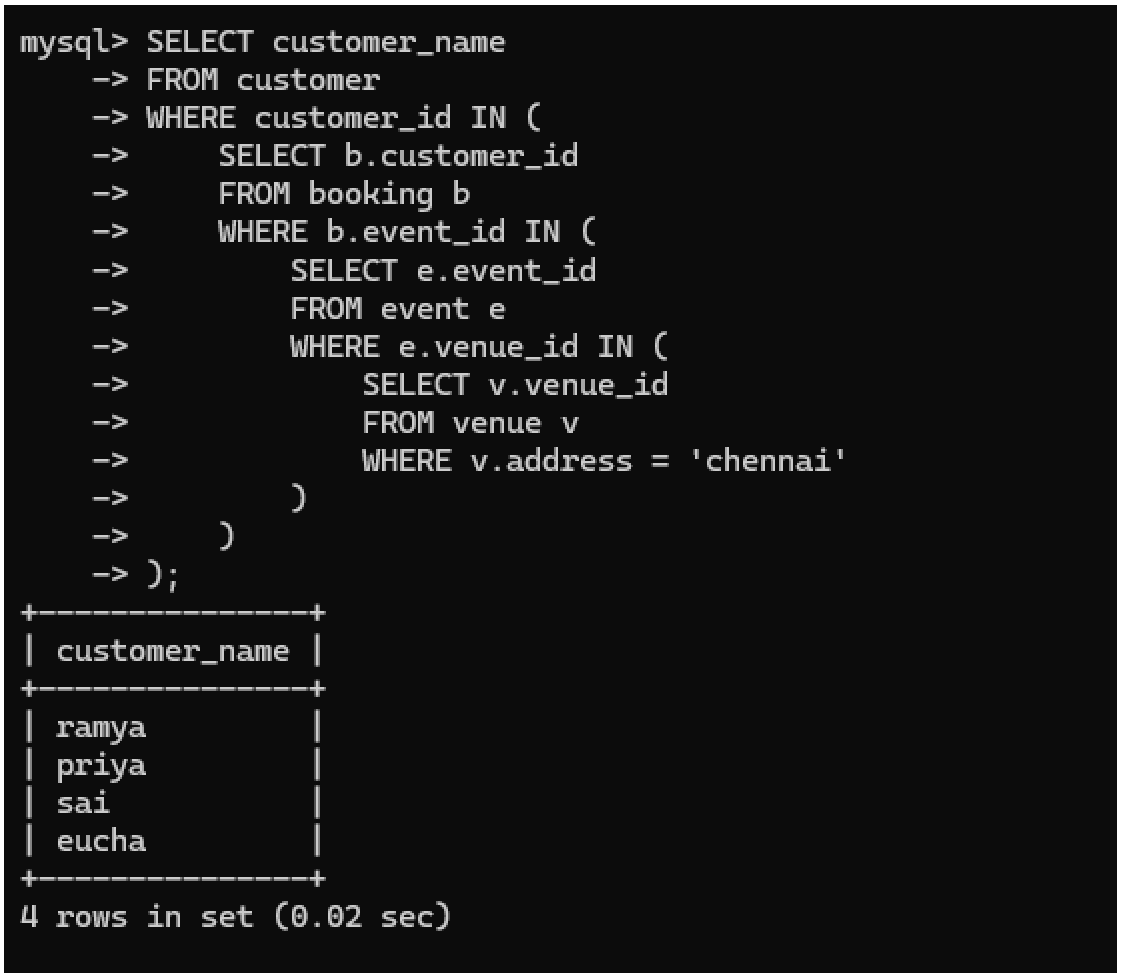


```
    ->     customer c;
+---------------+---------------+
| customer_name | total_revenue |
+---------------+---------------+
| ramya         |       1300.10 |
| saru          |          NULL |
| kaviya        |       1000.05 |
| priya         |       6800.90 |
| sam           |        300.45 |
| teja          |       1200.90 |
| geetha        |          NULL |
| sai           |        500.00 |
| sara          |       2000.09 |
| eucha         |        300.05 |
+---------------+---------------+
```

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

```
SELECT customer_name
   -> FROM customer
   -> WHERE customer_id IN (
   ->     SELECT b.customer_id
   ->     FROM booking b
   ->     WHERE b.event_id IN (
   ->         SELECT e.event_id
   ->         FROM event e
   ->         WHERE e.venue_id IN (
   ->             SELECT v.venue_id
   ->             FROM venue v
   ->             WHERE v.address = 'chennai'
   ->         )
   ->     )
```

->);



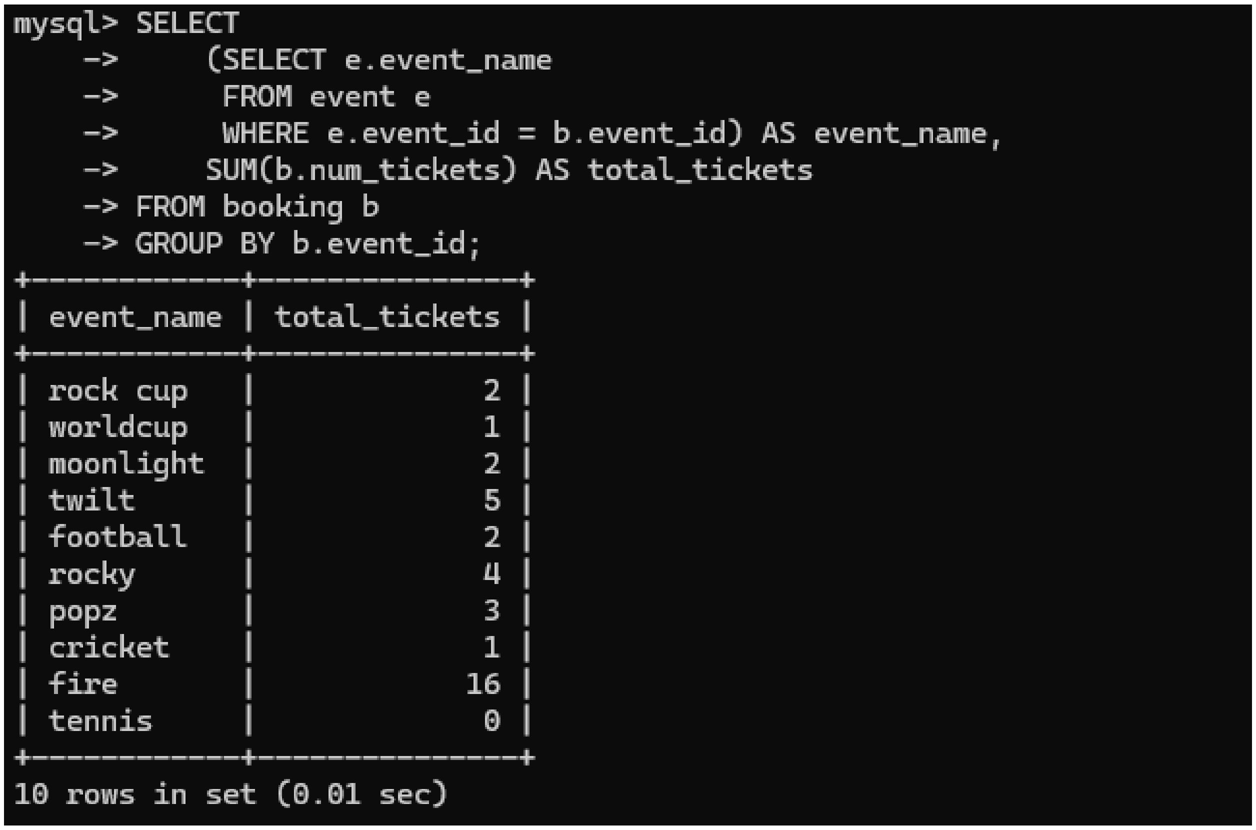
```
mysql> SELECT customer_name
    -> FROM customer
    -> WHERE customer_id IN (
    ->     SELECT b.customer_id
    ->     FROM booking b
    ->     WHERE b.event_id IN (
    ->         SELECT e.event_id
    ->         FROM event e
    ->         WHERE e.venue_id IN (
    ->             SELECT v.venue_id
    ->             FROM venue v
    ->             WHERE v.address = 'chennai'
    ->         )
    ->     )
    -> );
+---------------+
| customer_name |
+---------------+
| ramya         |
| priya         |
| sai           |
| eucha         |
+---------------+
4 rows in set (0.02 sec)
```

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

GROUP BY.

```
SELECT
   ->     (SELECT e.event_name
   ->      FROM event e
   ->      WHERE e.event_id = b.event_id) AS event_name,
   ->     SUM(b.num_tickets) AS total_tickets
   -> FROM booking b
```

-> GROUP BY b.event_id;

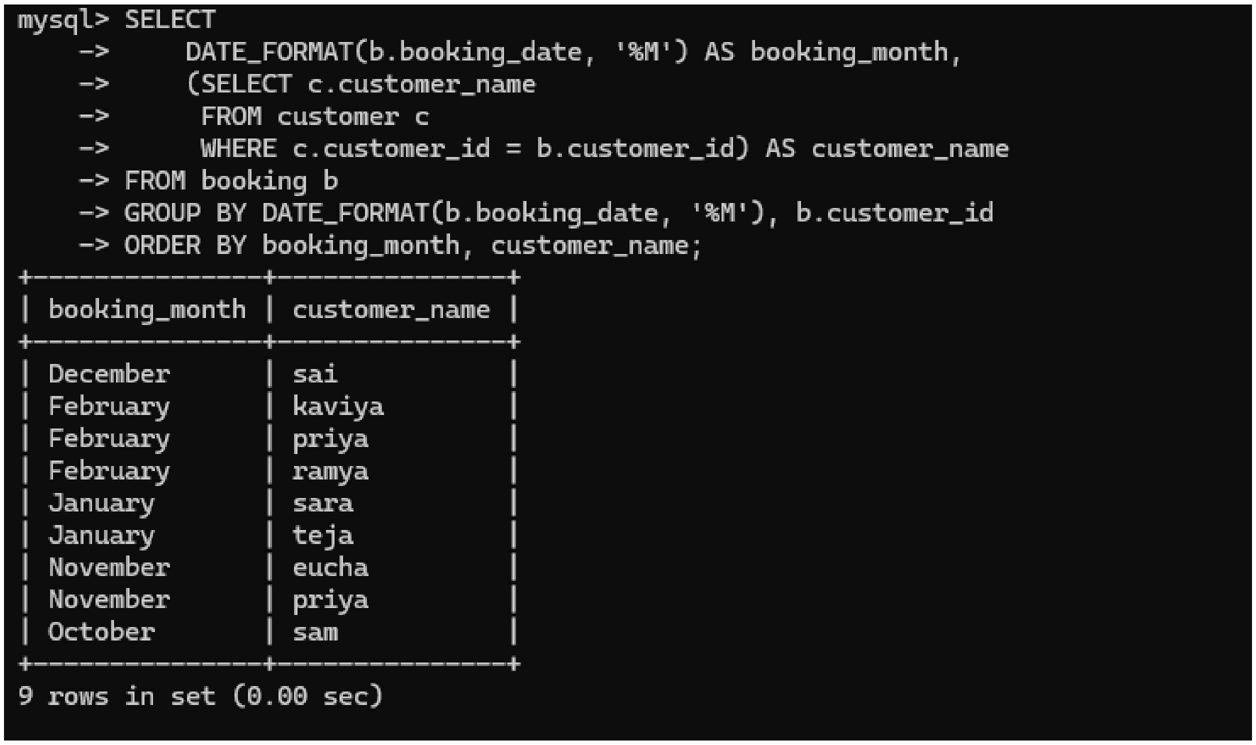


```
mysql> SELECT
    ->     (SELECT e.event_name
    ->      FROM event e
    ->      WHERE e.event_id = b.event_id) AS event_name,
    ->     SUM(b.num_tickets) AS total_tickets
    -> FROM booking b
    -> GROUP BY b.event_id;
+------------+---------------+
| event_name | total_tickets |
+------------+---------------+
| rock cup   |             2 |
| worldcup   |             1 |
| moonlight  |             2 |
| twilt      |             5 |
| football   |             2 |
| rocky      |             4 |
| popz       |             3 |
| cricket    |             1 |
| fire       |            16 |
| tennis     |             0 |
+------------+---------------+
10 rows in set (0.01 sec)
```

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

DATE_FORMAT.

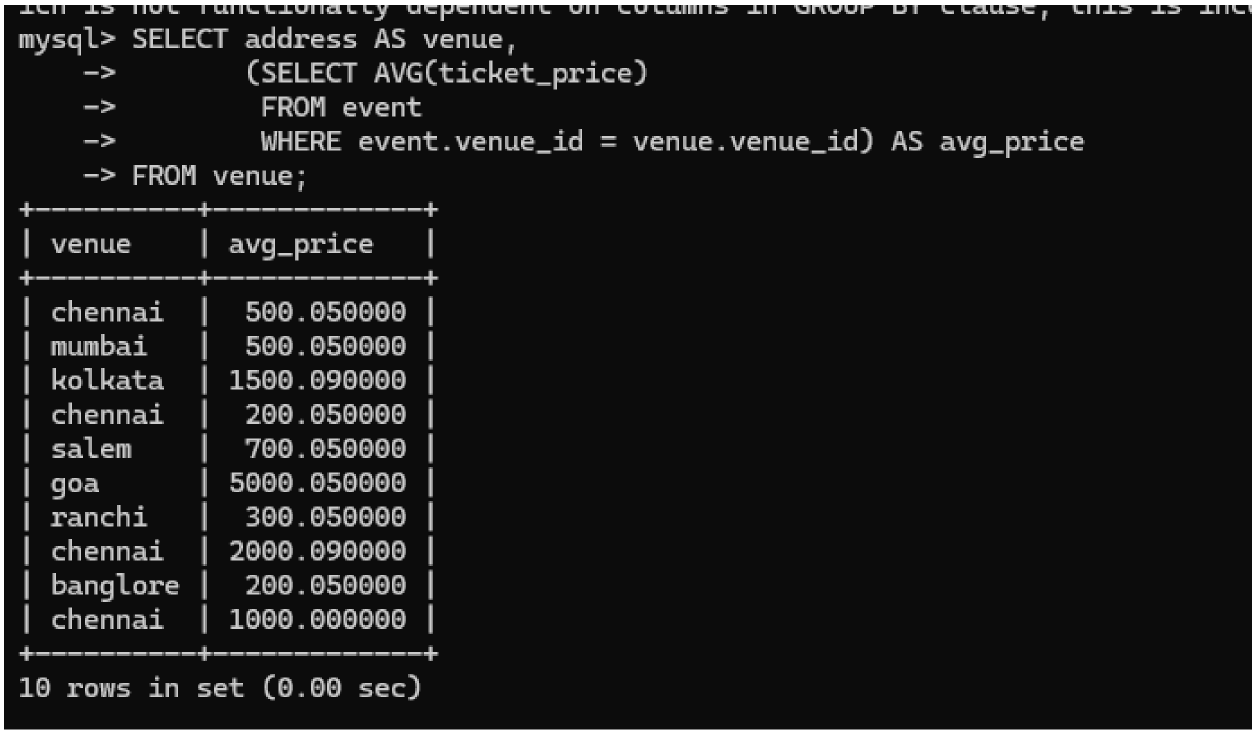
```
SELECT
    DATE_FORMAT(b.booking_date, '%M') AS booking_month,
    (SELECT c.customer_name
     FROM customer c
     WHERE c.customer_id = b.customer_id) AS customer_name
FROM booking b
GROUP BY DATE_FORMAT(b.booking_date, '%M'), b.customer_id
ORDER BY booking_month, customer_name;
```



```
mysql> SELECT
    ->     DATE_FORMAT(b.booking_date, '%M') AS booking_month,
    ->     (SELECT c.customer_name
    ->      FROM customer c
    ->      WHERE c.customer_id = b.customer_id) AS customer_name
    -> FROM booking b
    -> GROUP BY DATE_FORMAT(b.booking_date, '%M'), b.customer_id
    -> ORDER BY booking_month, customer_name;
+---------------+---------------+
| booking_month | customer_name |
+---------------+---------------+
| December      | sai           |
| February      | kaviya        |
| February      | priya         |
| February      | ramya         |
| January       | sara          |
| January       | teja          |
| November      | eucha         |
| November      | priya         |
| October       | sam           |
+---------------+---------------+
9 rows in set (0.00 sec)
```

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

```
SELECT address AS venue,
   -> (SELECT AVG(ticket_price)
   ->   FROM event
   ->   WHERE event.venue_id = venue.venue_id) AS avg_price
-> FROM venue;
```



```
ich is not functionally dependent on columns in GROUP BY clause; this is inc
mysql> SELECT address AS venue,
    ->        (SELECT AVG(ticket_price)
    ->         FROM event
    ->         WHERE event.venue_id = venue.venue_id) AS avg_price
    -> FROM venue;
+----------+-------------+
| venue    | avg_price   |
+----------+-------------+
| chennai  |  500.050000 |
| mumbai   |  500.050000 |
| kolkata  | 1500.090000 |
| chennai  |  200.050000 |
| salem    |  700.050000 |
| goa      | 5000.050000 |
| ranchi   |  300.050000 |
| chennai  | 2000.090000 |
| banglore |  200.050000 |
| chennai  | 1000.000000 |
+----------+-------------+
10 rows in set (0.00 sec)
```