**ASSIGNMENT-1**

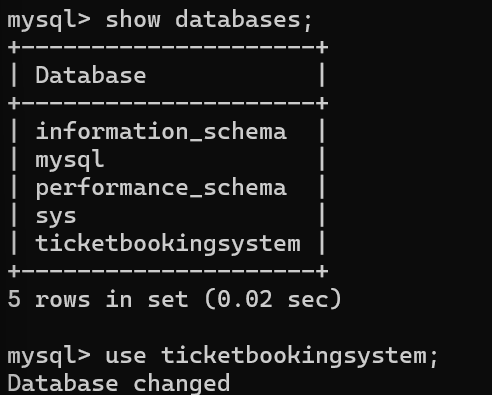
**TICKET BOOKING SYSTEM**

**TASK-1: DATABASE DESIGN**

1. **Create the database named "TicketBookingSystem"**.

Cretae database TicketBookingSystem;

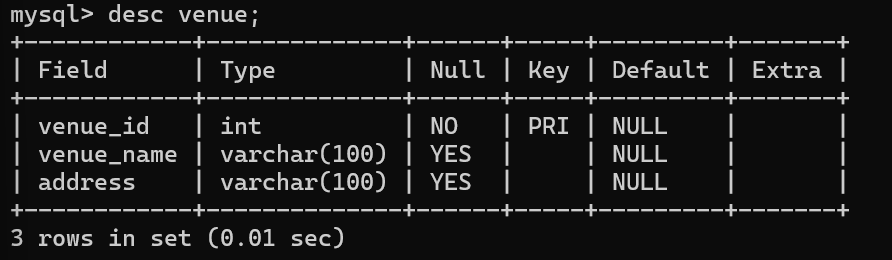
Show databases;



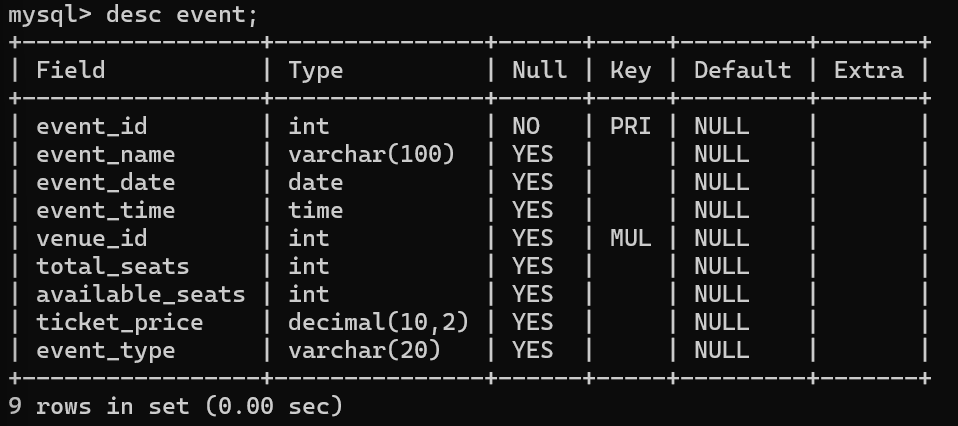
Use TicketBookingSystem;

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

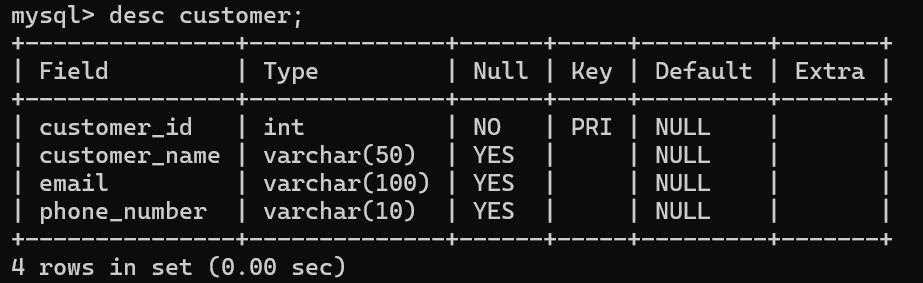
**Venue**: Create table venue(venue\_id integer primary key, venue\_name varchar(50), address varchar(100));



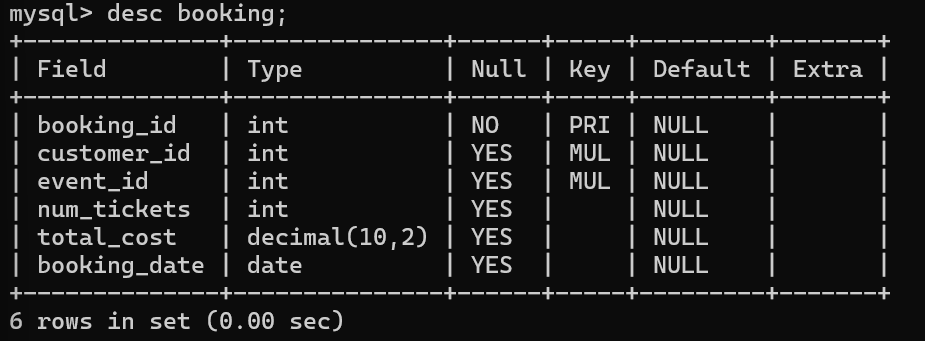
**Event**: Create table event(event\_id integer primary key, event\_name varchar(100), event\_date date, event\_time time, venue\_id integer, total\_seats integer, available\_seats integer, ticket\_price decimal(10,2), event\_type varchar(20), foreign key (venue\_id) references venue(venue\_id));



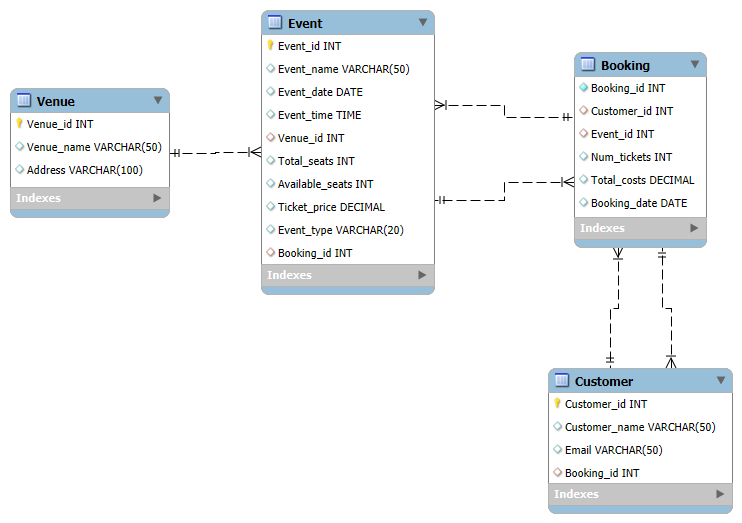
**Customers**: Create table customer(customer\_id integer primary key, customer\_name varchar(50), email varchar(100), phone\_number varchar(10));



**Booking:** Create table Booking (booking\_id integer primary key, customer\_id integer, event\_id integer, num\_tickets integer, total\_cost decimal(10, 2), booking\_date date, foreign key (customer\_id) references Customer(customer\_id), foreign key (event\_id) references Event(event\_id));

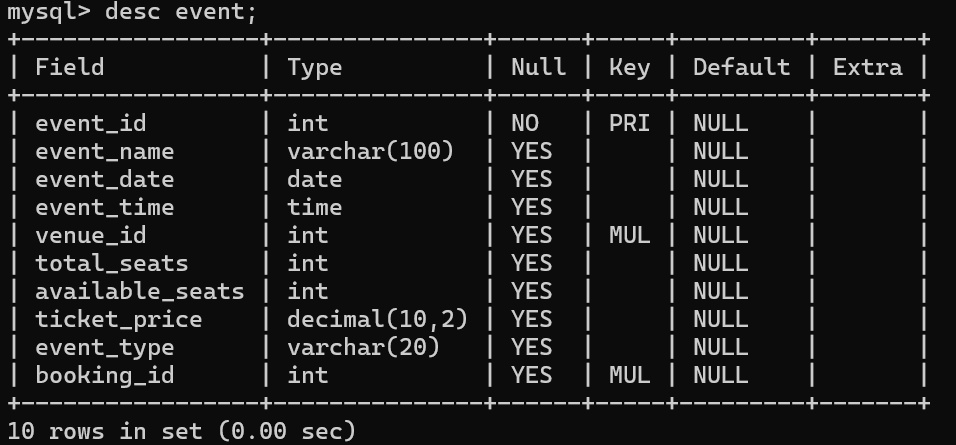


1. **Create an ERD (Entity Relationship Diagram) for the database**.

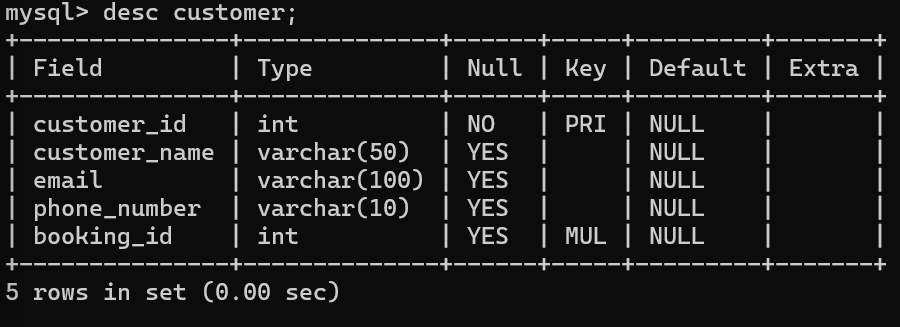


1. **Create appropriate Primary Key and Foreign Key constraints for referential integrity.**

* Alter table event add column booking\_id integer, add foreign key (booking\_id) references booking(booking\_id);



* Alter table customer add column booking\_id integer, add foreign key (booking\_id) references booking(booking\_id);



**TASK-2: SELECT, WHERE, BETWEEN, AND, LIKE**

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

**Venue:**

insert into venue values(101,'Royal cinemas','Pallipalayam');

insert into venue values(102,'PVR cinemas','Palayapalayam');

insert into venue values(103,'Cosmos cinemas','Perundurai');

insert into venue values(104,'ABC Ground','Bhavani');

insert into venue values(105,'DEF Ground','Erode');

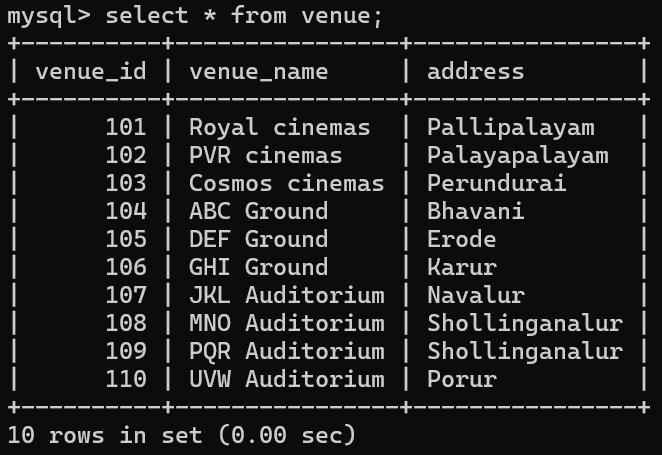
insert into venue values(106,'GHI Ground','Karur');

insert into venue values(107,'JKL Auditorium','Navalur');

insert into venue values(108,'MNO Auditorium','Shollinganalur');

insert into venue values(109,'PQR Auditorium','Shollinganalur');

insert into venue values(110,'UVW Auditorium','Porur');



**Event:**

insert into event(event\_id, event\_name, event\_date, event\_time, venue\_id, total\_seats, available\_seats, ticket\_price, event\_type)

values

(1, 'Summer Concert', '2023-08-15', '19:00:00', 101, 10000, 8000, 1500.00, 'Concert'),

(2, 'Fottball Game', '2023-09-23', '15:00:00', 102, 50000, 45000, 50.00, 'Sports'),

(3, 'Movie Premiere', '2023-10-10', '20:00:00', 101, 2000, 1500, 1200.00, 'Movie'),

(4, 'Winter Concert', '2023-12-25', '21:00:00', 103, 5000, 4000, 1000100, 'Concert'),

(5,'Basketball Game', '2024-01-15', '17:00:00', 102, 15000,12000, 75.00, 'Sports'),

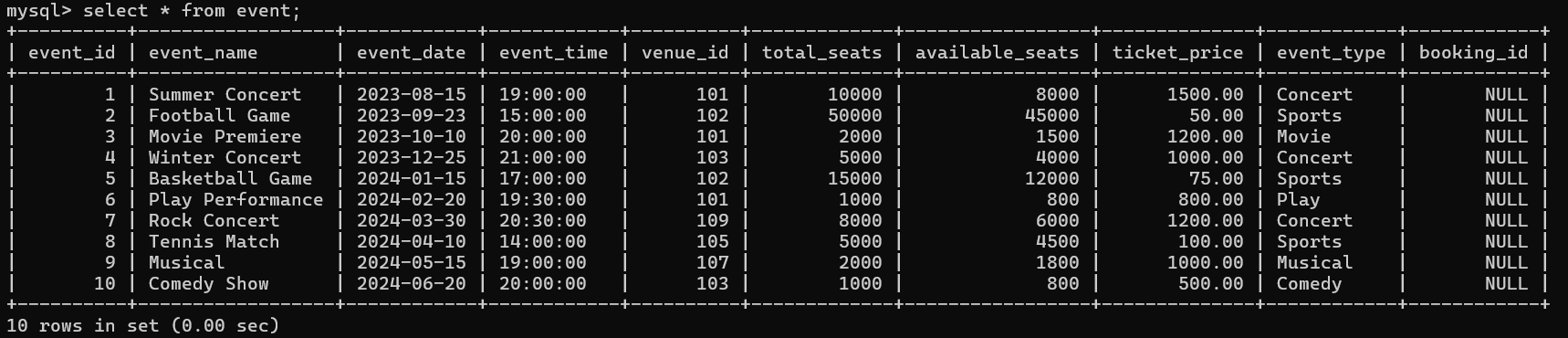
(6, 'Play Performance', '2024-02-20', '19:30:00', 11, 1000, 800, 800.00, 'Play'),

(7, 'Rock Concert', '2024-03-30', '20:30:00', 109, 8000, 6000, 1200.00,'Concert'),

(8, 'Tennis Match', '2024-04-10', '14:00:00', 105, 5000, 4500, 100.00, 'Sports'),

(9, 'Musical', '2024-0515', '19:00:00', 107, 2000, 1800, 1000.00, 'Musical'),

(10, 'Comedy Show', '2024-06-20', '20:00:00',103, 1000, 800,500.00, 'Comedy');



**Customer**:

insert into Customer (customer\_id, customer\_name, email, phone\_number)

values

(1, 'Gayathri', 'gayathri@example.com', '1234567890'),

(2, 'Deepa', 'Deepa@example.com', '9876543210'),

(3, 'Lobi', 'Lobi@example.com', '5555555555'),

(4, 'Moni', 'moni@example.com', '1111111111'),

(5, 'bhusan', 'bhusan@example.com', '2222222222'),

(6, 'harish','harish@example.com', '3333333333'),

(7, 'sandy', 'sandy@example.com', '4444444444'),

(8, 'praveen', 'praveen@example.com', '5555555555'),

(9, 'mei', 'mei@example.com', '6666666666'),

(10, 'deetch', 'deetch@example.com','7777777777');



**Booking**:

insert into Booking (booking\_id, customer\_id, event\_id, num\_tickets, total\_cost, booking\_date)

values

(1, 1, 1, 2, 3000.00, '2023-08-10'),

(2, 2, 2, 5, 250.00, '2023-09-18'),

(3, 3, 3, 3, 3600.00, '2023-10-05'),

(4, 4, 4, 4, 4000.00, '2023-12-20'),

(5, 5, 5, 1, 75.00, '2024-01-10'),

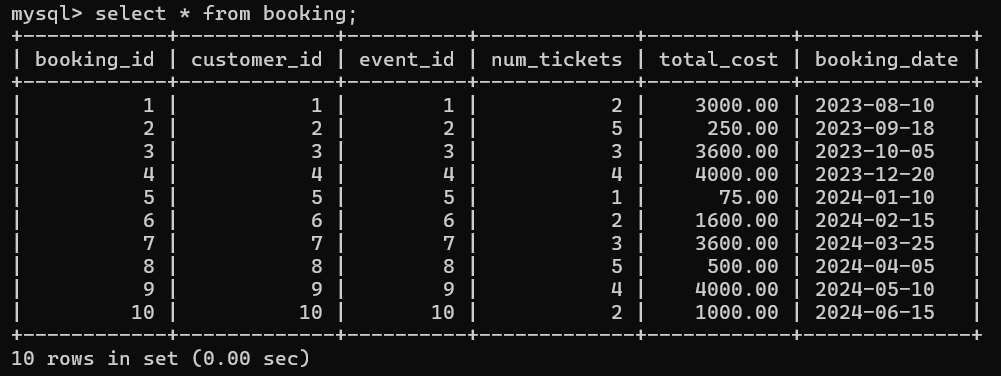
(6, 6, 6, 2, 1600.00, '2024-02-15'),

(7, 7, 7, 3, 3600.00, '2024-03-25'),

(8, 8, 8, 5, 500.00, '2024-04-05'),

(9, 9, 9, 4, 4000.00, '2024-05-10'),

(10, 10, 10, 2, 1000.00, '2024-06-15');



**Updating Booking\_id in Event table**:

update event set booking\_id=1 where event\_id=1;

update event set booking\_id=2 where event\_id=2;

update event set booking\_id=3 where event\_id=3;

update event set booking\_id=4 where event\_id=4;

update event set booking\_id=5 where event\_id=5;

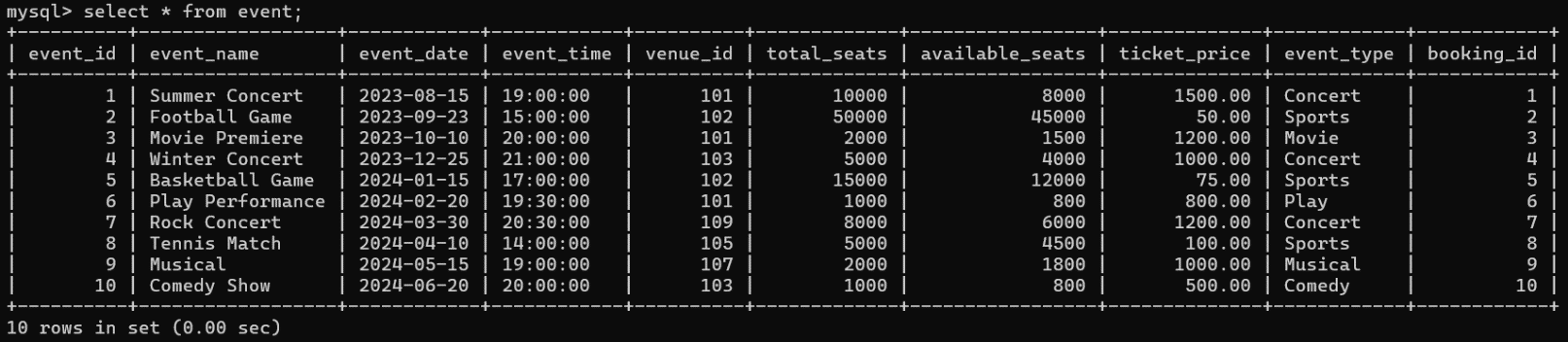
update event set booking\_id=6 where event\_id=6;

update event set booking\_id=7 where event\_id=7;

update event set booking\_id=8 where event\_id=8;

update event set booking\_id=9 where event\_id=9;

update event set booking\_id=10 where event\_id=10;



**Updating Booking\_id in Customer table**:

update customer set booking\_id=10 where customer\_id=10;

update customer set booking\_id=9 where customer\_id=9;

update customer set booking\_id=8 where customer\_id=8;

update customer set booking\_id=7 where customer\_id=7;

update customer set booking\_id=6 where customer\_id=6;

update customer set booking\_id=5 where customer\_id=5;

update customer set booking\_id=4 where customer\_id=4;

update customer set booking\_id=3 where customer\_id=3;

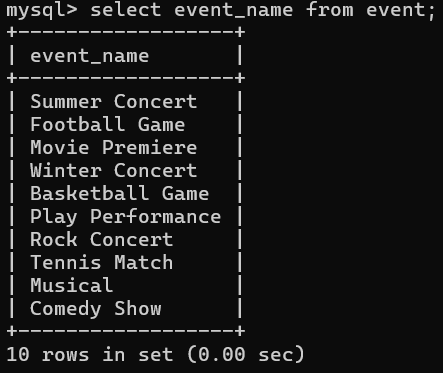
update customer set booking\_id=2 where customer\_id=2;

update customer set booking\_id=1 where customer\_id=1;



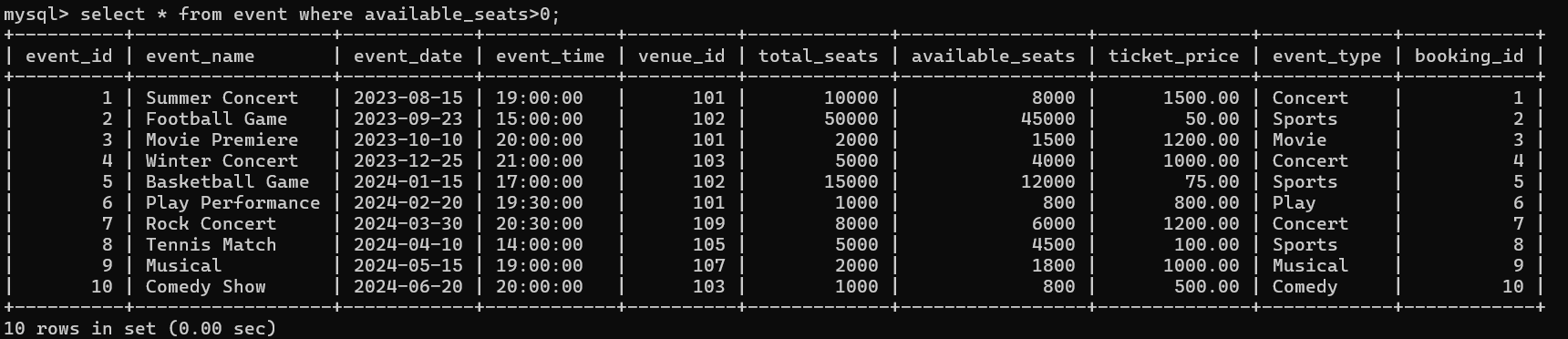
**Write a SQL query to list all Events.**

select event\_name from event;



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

select \* from event where available\_seats>0;



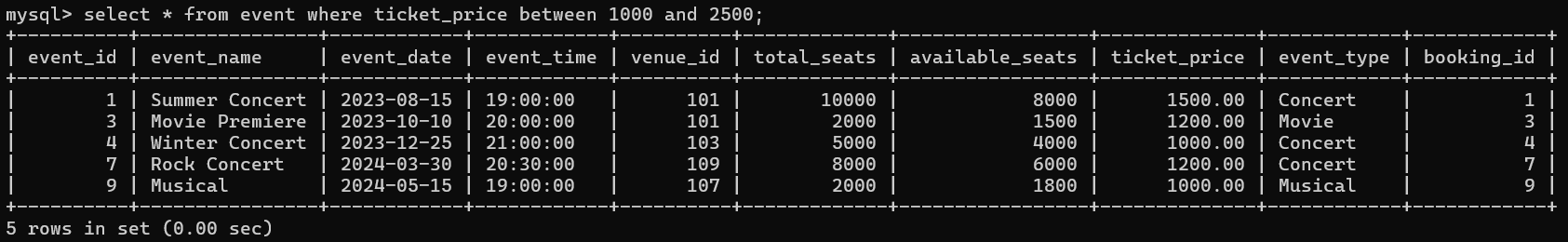
**Write a SQL query to select events name partial match with ‘cup’**.

select \* from event where event\_name like '%cup%';



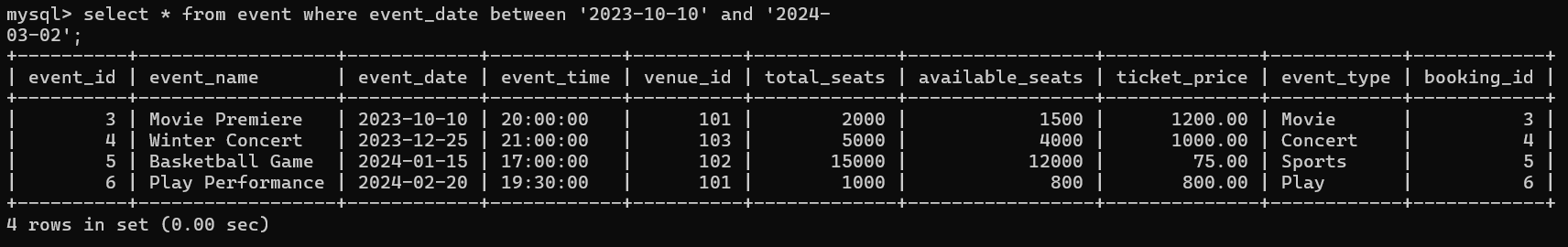
**Write a SQL query to select events with ticket price range is between 1000 to 2500**.

select \* from event where ticket\_price between 1000 and 2500;



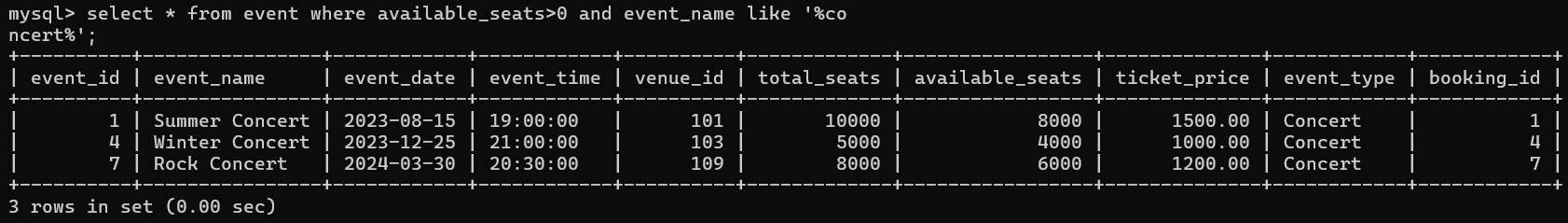
**Write a SQL query to retrieve events with dates falling within a specific range**.

select \* from event where event\_date between '2023-10-10' and '2024-03-02';



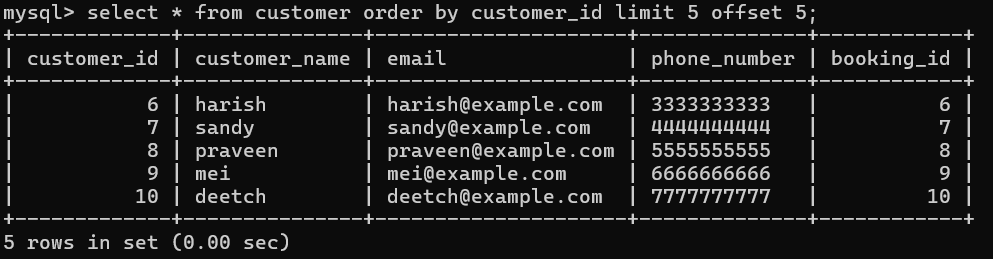
**Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.**

select \* from event where available\_seats>0 and event\_name like '%concert%';



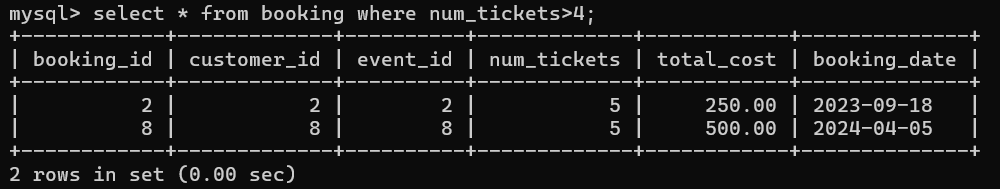
**Write a SQL query to retrieve users in batches of 5, starting from the 6th user.**

select \* from customer order by customer\_id limit 5 offset 5;



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

select \* from booking where num\_tickets>4;



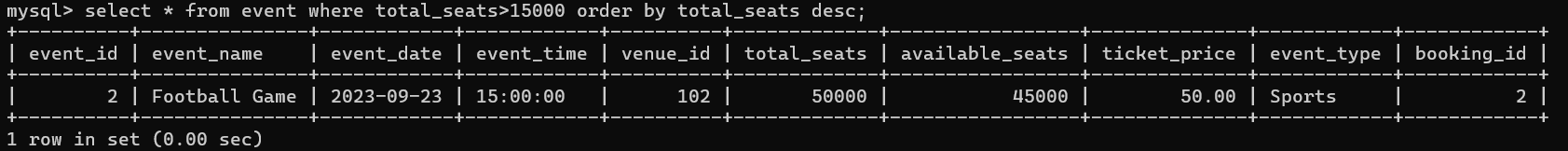
**Write a SQL query to retrieve customer information whose phone number end with ‘000’**

select \* from customer where phone\_number like '%000';



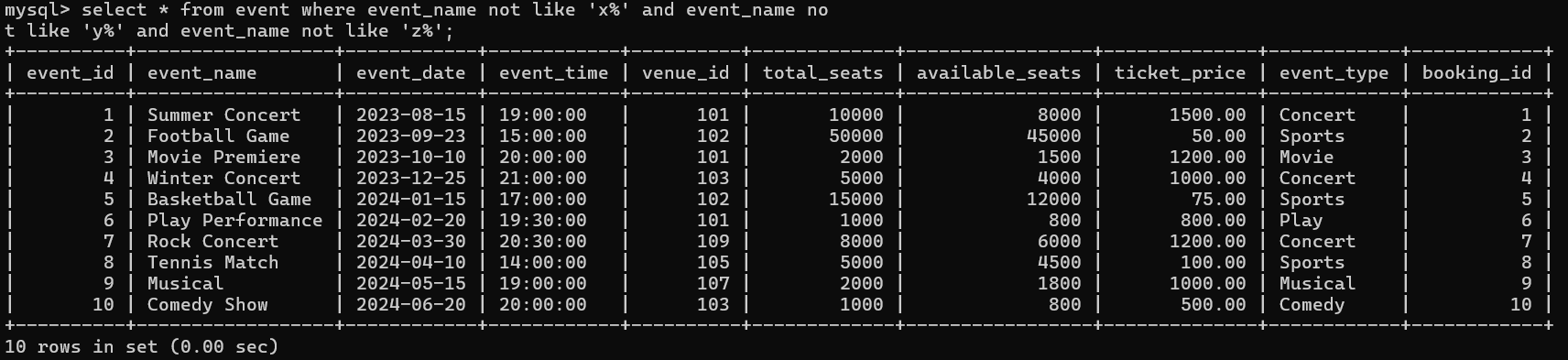
**Write a SQL query to retrieve the events in order whose seat capacity more than 15000.**

select \* from event where total\_seats>15000 order by total\_seats desc;



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

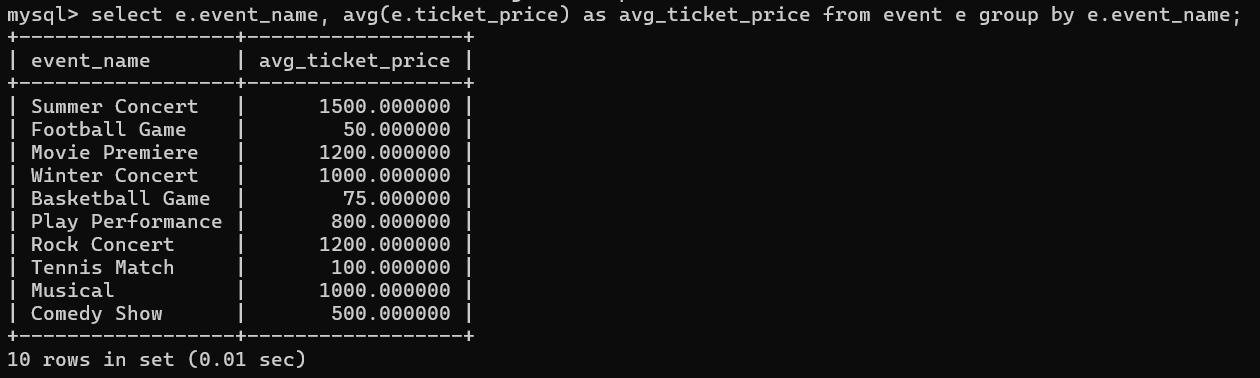
select \* from event where event\_name not like 'x%' and event\_name not like 'y%' and event\_name not like 'z%';



**TASK-3: AGGREGATE FUNCTIONS, HAVING, ORDERBY, GROUPBY AND JOINS**

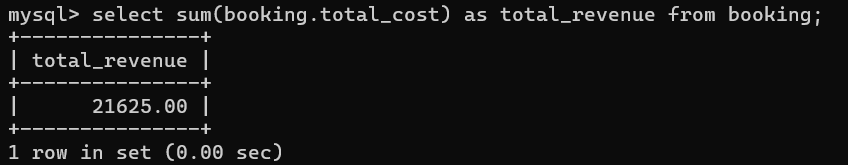
**Write a SQL query to List Events and Their Average Ticket Prices**

select e.event\_name, avg(e.ticket\_price) as avg\_ticket\_price from event e group by e.event\_name;



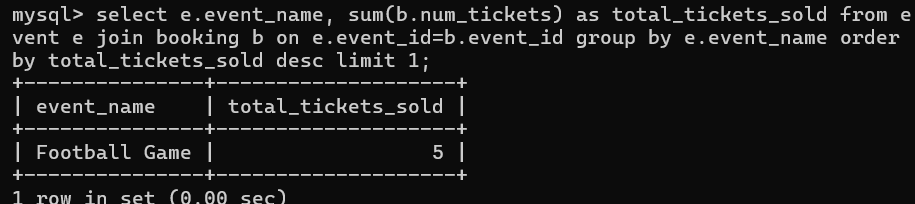
**Write a SQL query to Calculate the Total Revenue Generated by Events.**

select sum(booking.total\_cost) as total\_revenue from booking;



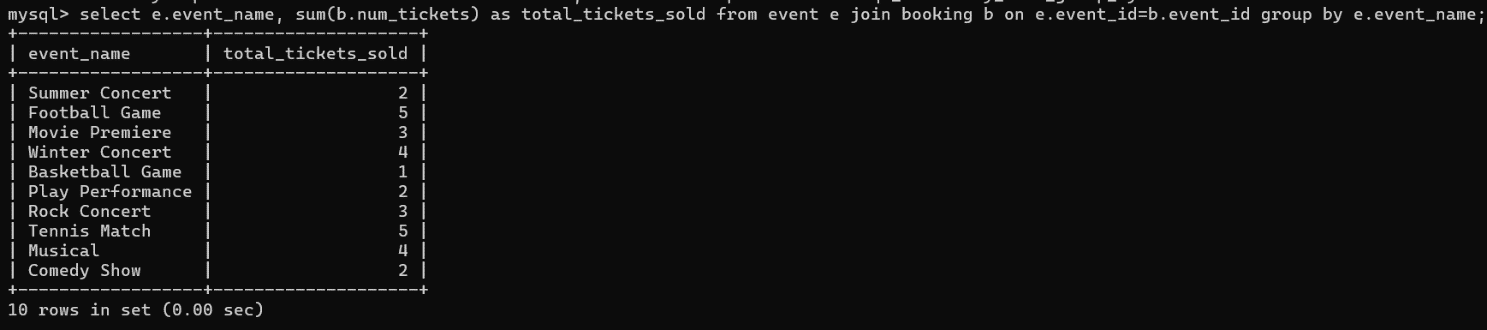
**Write a SQL query to find the event with the highest ticket sales**.

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 order by total\_tickets\_sold desc limit 1;



**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;



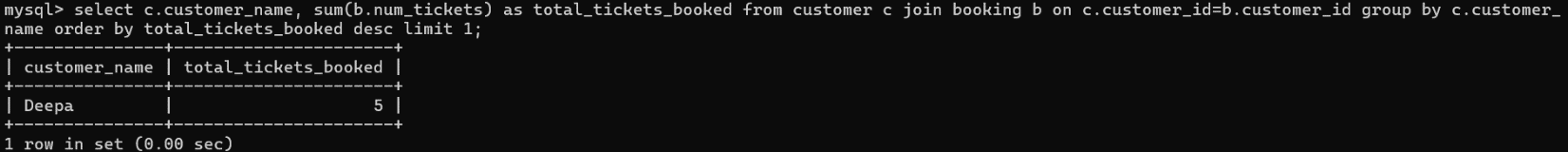
**Write a SQL query to Find Events with No Ticket Sales.**

select e.event\_name from event e left join booking b on e.event\_id=b.event\_id where b.booking\_id is null;



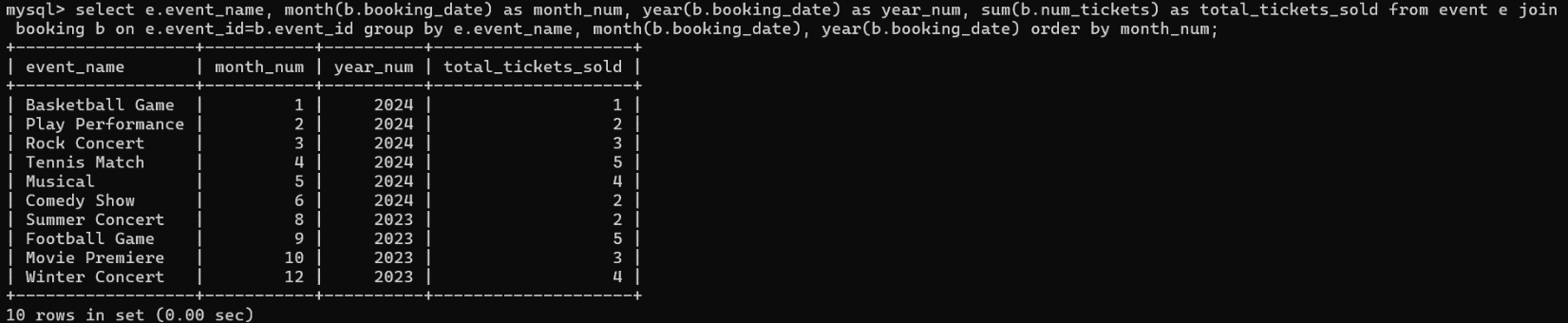
**Write a SQL query to Find the User Who Has Booked the Most Tickets**.

select c.customer\_name, sum(b.num\_tickets) as total\_tickets\_booked from customer c join booking b on c.customer\_id=b.customer\_id group by c.customer\_name order by total\_tickets\_booked desc limit 1;



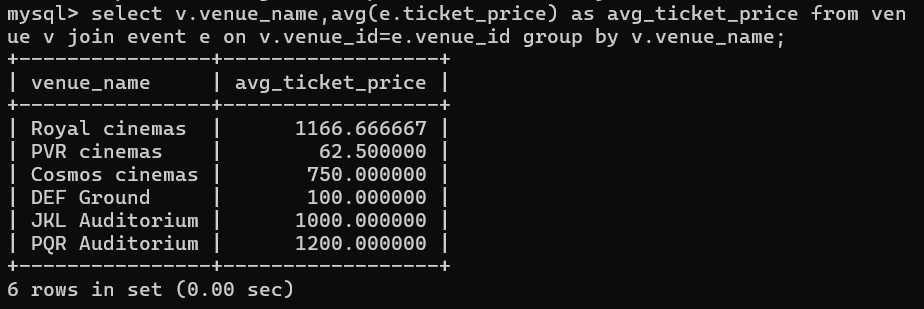
**Write a SQL query to List Events and the total number of tickets sold for each month.**

select e.event\_name, month(b.booking\_date) as month\_num, year(b.booking\_date) as year\_num, sum(b.num\_tickets) as total\_tickets\_sold from event e join booking b on e.event\_id=b.event\_id group by e.event\_name, month(b.booking\_date), year(b.booking\_date) order by month\_num;



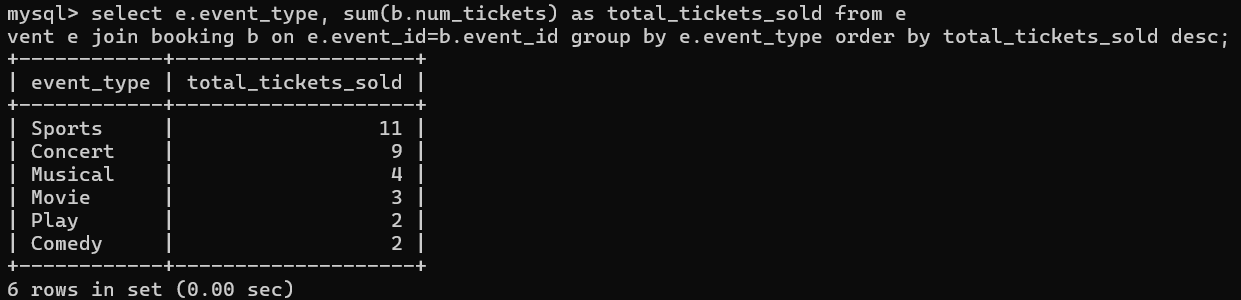
**Write a SQL query to calculate the average Ticket Price for Events in Each Venue**.

select v.venue\_name,avg(e.ticket\_price) as avg\_ticket\_price from venue v join event e on v.venue\_id=e.venue\_id group by v.venue\_name;



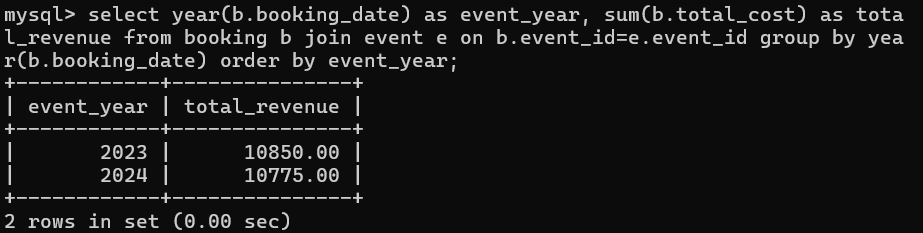
**Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.**

select e.event\_type, sum(b.num\_tickets) as total\_tickets\_sold from event e join booking b on e.event\_id=b.event\_id group by e.event\_type order by total\_tickets\_sold desc;



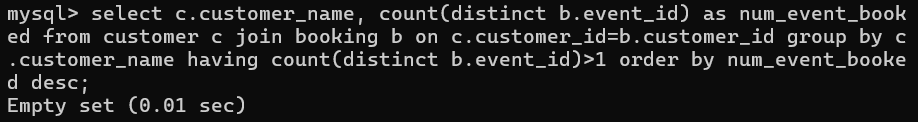
**Write a SQL query to calculate the total Revenue Generated by Events in Each Year**.

select year(b.booking\_date) as event\_year, sum(b.total\_cost) as total\_revenue from booking b join event e on b.event\_id=e.event\_id group by year(b.booking\_date) order by event\_year;



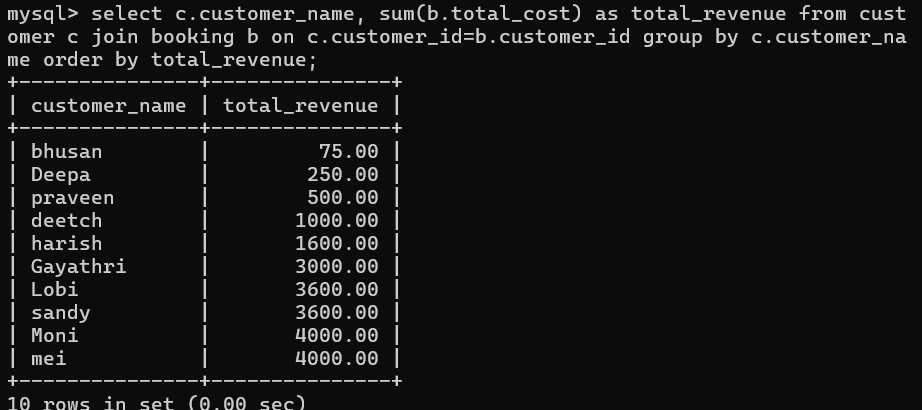
**Write a SQL query to list users who have booked tickets for multiple events**.

select c.customer\_name, count(distinct b.event\_id) as num\_event\_booked from customer c join booking b on c.customer\_id=b.customer\_id group by c.customer\_name having count(distinct b.event\_id)>1 order by num\_event\_booked desc;



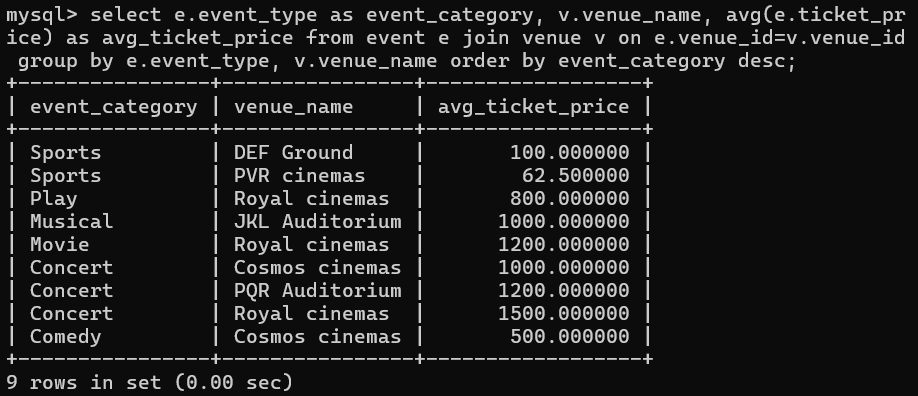
**Write a SQL query to calculate the Total Revenue Generated by Events for Each User.**

select c.customer\_name, sum(b.total\_cost) as total\_revenue from customer c join booking b on c.customer\_id=b.customer\_id group by c.customer\_name order by total\_revenue;



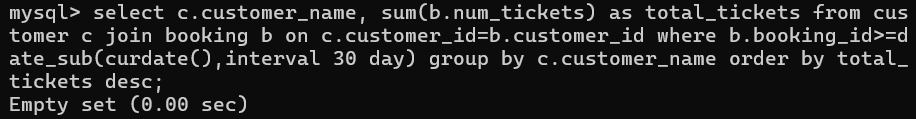
**Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.**

select e.event\_type as event\_category, v.venue\_name, avg(e.ticket\_price) as avg\_ticket\_price from event e join venue v on e.venue\_id=v.venue\_id group by e.event\_type, v.venue\_name order by event\_category desc;



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

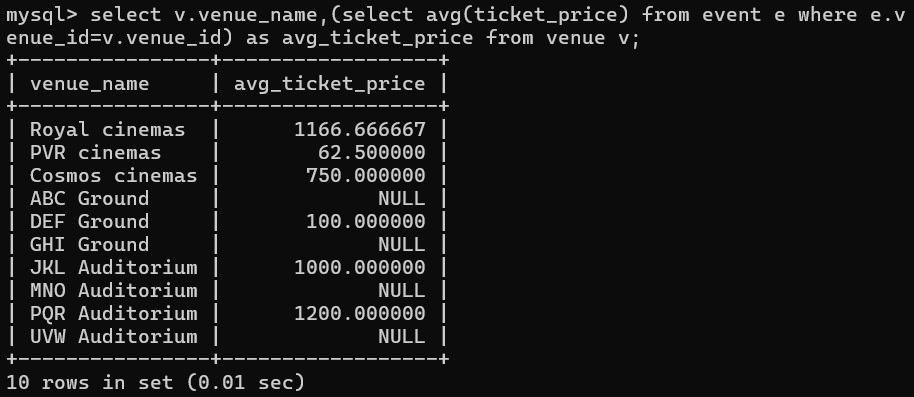
select c.customer\_name, sum(b.num\_tickets) as total\_tickets from customer c join booking b on c.customer\_id=b.customer\_id where b.booking\_id>=date\_sub(curdate(),interval 30 day) group by c.customer\_name order by total\_tickets desc;



**TASK-4: SUBQUERY AND ITS TYPES**

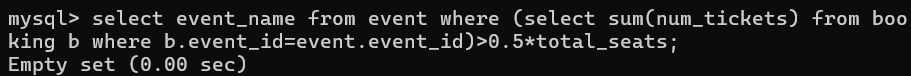
**Calculate the Average Ticket Price for Events in Each Venue Using a Subquery**.

select v.venue\_name,(select avg(ticket\_price) from event e where e.venue\_id=v.venue\_id) as avg\_ticket\_price from venue v;



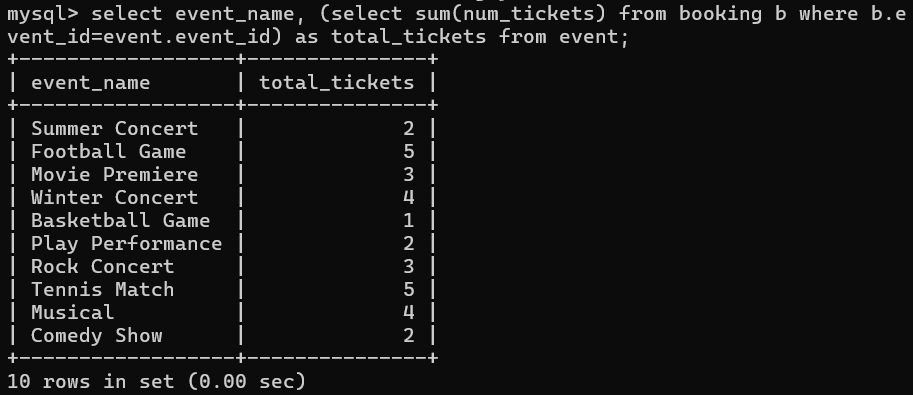
**Find Events with More Than 50% of Tickets Sold using subquery**.

select event\_name from event where (select sum(num\_tickets) from booking b where b.event\_id=event.event\_id)>0.5\*total\_seats;



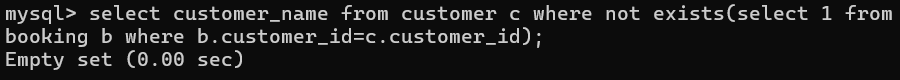
**Calculate the Total Number of Tickets Sold for Each Event.**

select event\_name, (select sum(num\_tickets) from booking b where b.event\_id=event.event\_id) as total\_tickets from event;



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

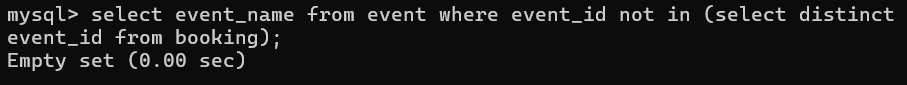
select customer\_name from customer c where not exists(select 1 from booking b where b.customer\_id=c.customer\_id);



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

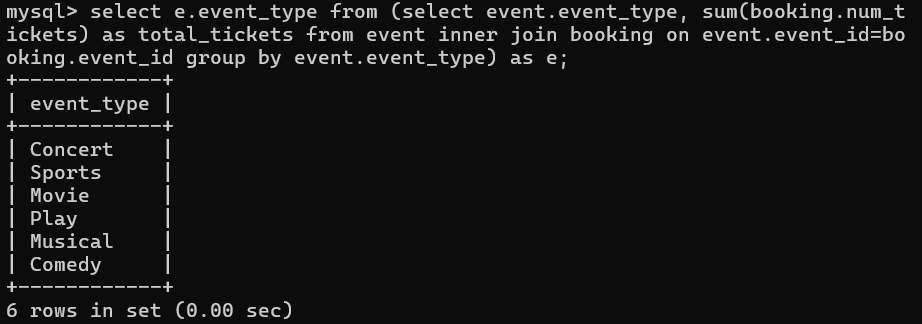
select event\_name from event where event\_id not in (select distinct

event\_id from booking);



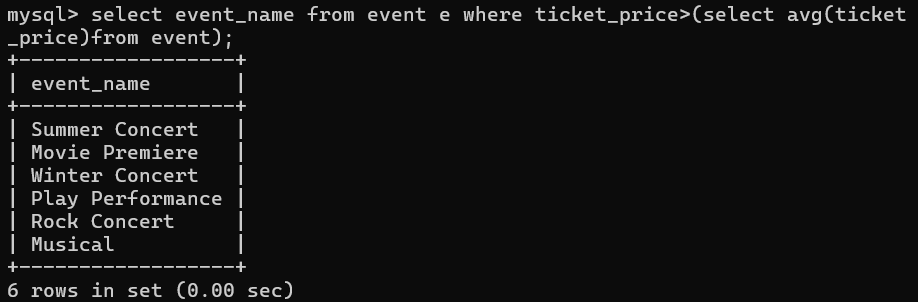
**Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause**.

select e.event\_type from (select event.event\_type, sum(booking.num\_tickets) as total\_tickets from event inner join booking on event.event\_id=booking.event\_id group by event.event\_type) as e;



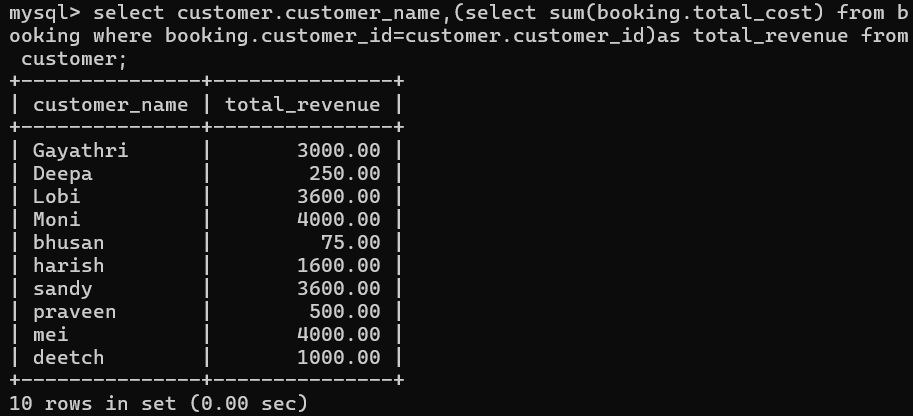
**Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause**.

select event\_name from event e where ticket\_price>(select avg(ticket\_price)from event);



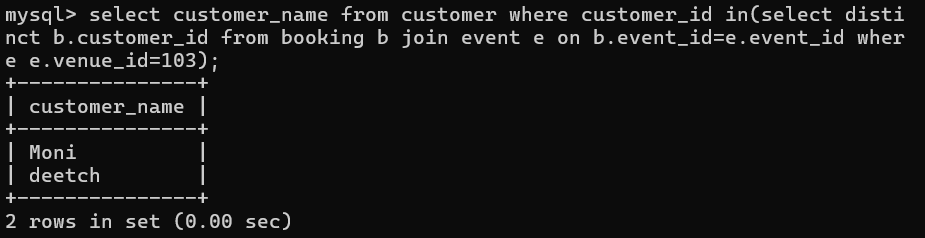
**Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery**.

select customer.customer\_name,(select sum(booking.total\_cost) from booking where booking.customer\_id=customer.customer\_id)as total\_revenue from customer;



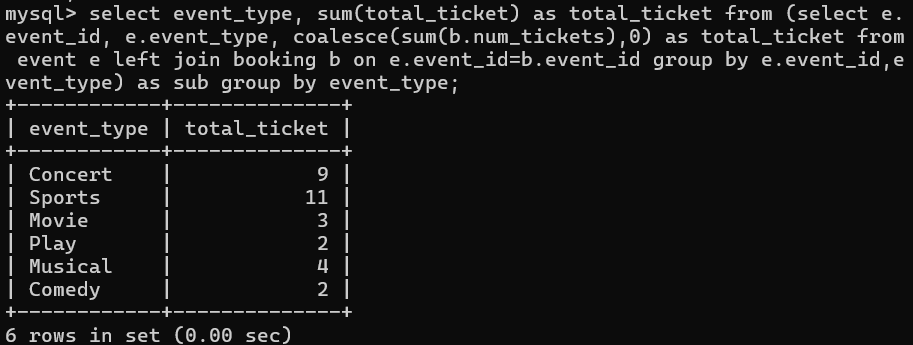
**List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause**.

select customer\_name from customer where customer\_id in(select distinct b.customer\_id from booking b join event e on b.event\_id=e.event\_id where e.venue\_id=103);



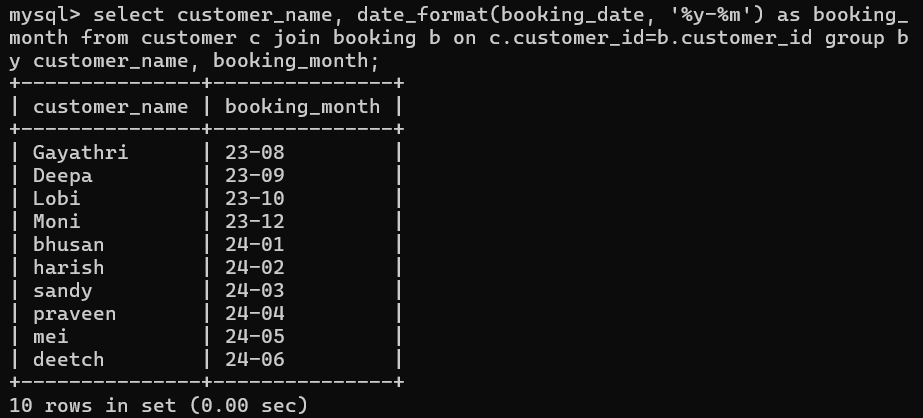
**Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY**.

select event\_type, sum(total\_ticket) as total\_ticket from (select e.event\_id, e.event\_type, coalesce(sum(b.num\_tickets),0) as total\_ticket from event e left join booking b on e.event\_id=b.event\_id group by e.event\_id,event\_type) as sub group by event\_type;



**Find Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE\_FORMAT**.

select customer\_name, date\_format(booking\_date, '%y-%m') as booking\_month from customer c join booking b on c.customer\_id=b.customer\_id group by customer\_name, booking\_month;



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

select v.venue\_name, (select avg(ticket\_price) from event e where e.venue\_id=v.venue\_id) as avg\_ticket\_price from venue v;

