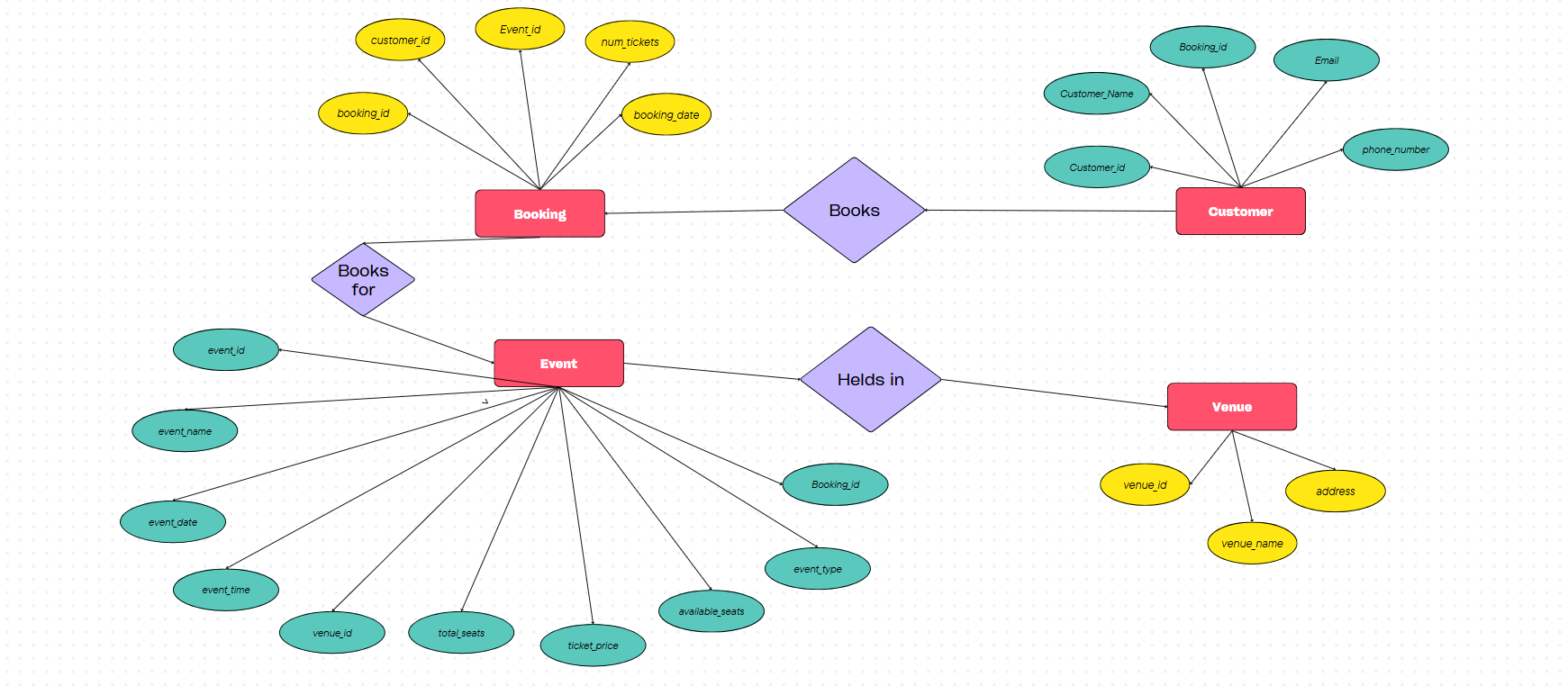
**Ticket Booking System**

**Tasks 1: Database Design:**

****

1Create the database named "TicketBookingSystem"

create database ticketbookingsystem;

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

relationships.

• Venu

create table venu (

venue\_id int primary key,

venue\_name varchar(255) not null,

address text not null

);

• Event

create table event (

event\_id int primary key,

event\_name varchar(255) not null,

event\_date date not null,

event\_time time not null,

venue\_id int,

total\_seats int not null,

available\_seats int not null,

ticket\_price decimal(8, 2) not null,

event\_type enum('movie', 'sports', 'concert') not null,

booking\_id int,

foreign key (venue\_id) references venu(venue\_id),

foreign key (booking\_id) references booking(booking\_id) on delete set null

);

• Customers

create table customer (

customer\_id int primary key,

customer\_name varchar(255) not null,

email varchar(255) unique not null,

phone\_number varchar(20) not null,

booking\_id int,

foreign key (booking\_id) references booking(booking\_id) on delete set null

);

• Booking

create table booking (

booking\_id int primary key,

customer\_id int,

event\_id int,

num\_tickets int not null,

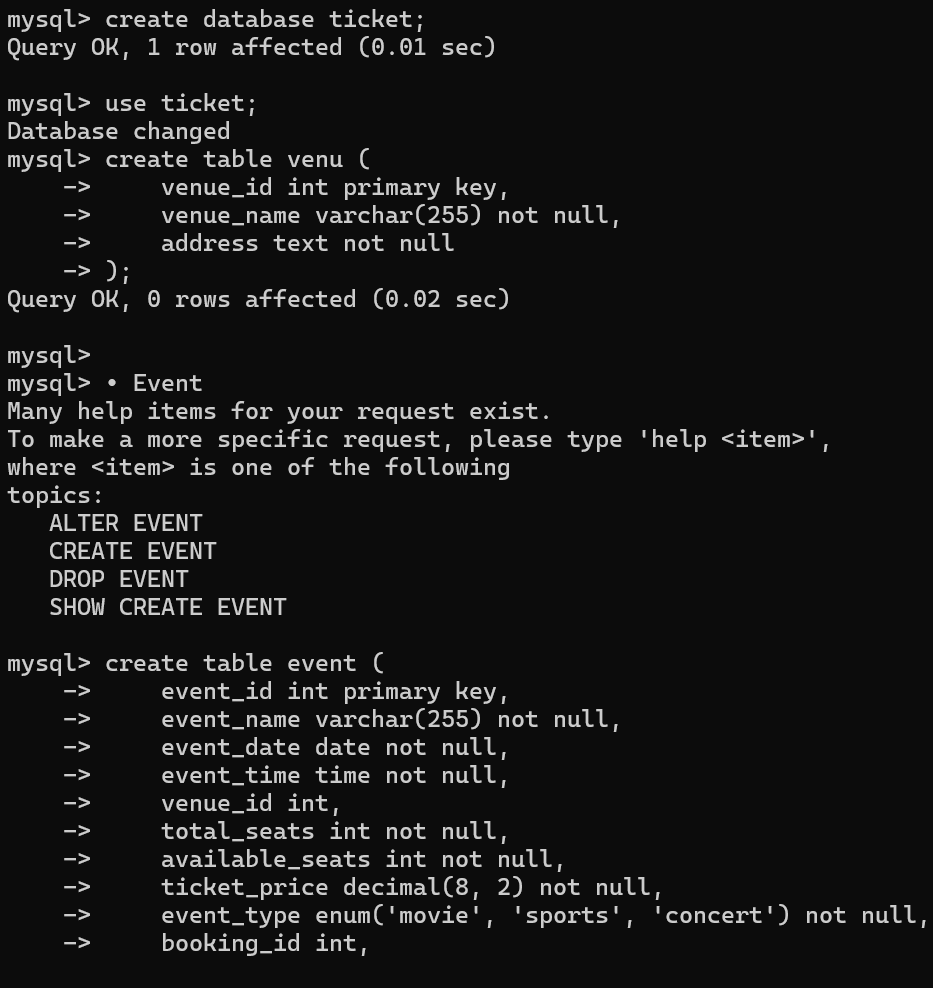
total\_cost decimal(10, 2) not null,

booking\_date datetime default current\_timestamp,

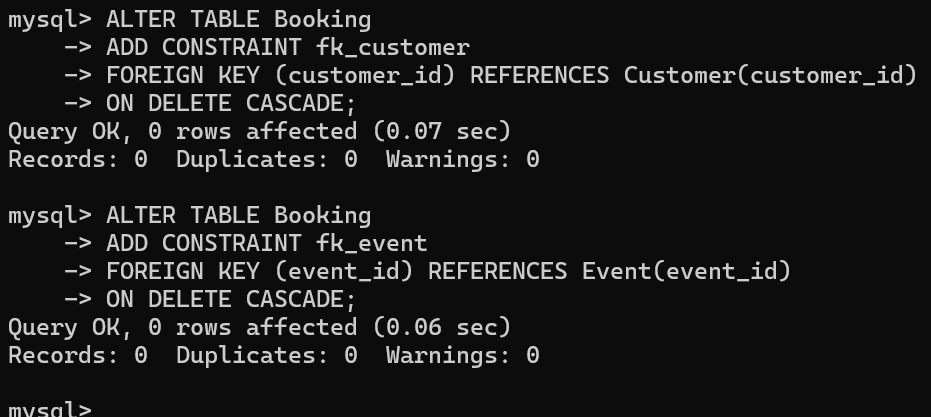
foreign key (customer\_id) references customer(customer\_id),

foreign key (event\_id) references event(event\_id)

);







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

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

Venue:

insert into venu (venue\_id, venue\_name, address) values

(1, 'madison square garden', 'new york, ny'),

(2, 'wembley stadium', 'london, uk'),

(3, 'sydney opera house', 'sydney, au'),

(4, 'staples center', 'los angeles, ca'),

(5, 'national stadium', 'beijing, cn'),

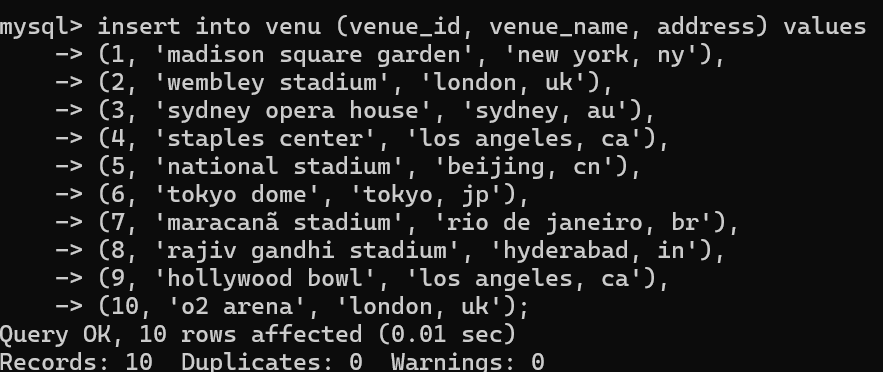
(6, 'tokyo dome', 'tokyo, jp'),

(7, 'maracanã stadium', 'rio de janeiro, br'),

(8, 'rajiv gandhi stadium', 'hyderabad, in'),

(9, 'hollywood bowl', 'los angeles, ca'),

(10, 'o2 arena', 'london, uk');



Event

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, 'world cup finals', '2025-06-15', '18:00:00', 1, 50000, 15000, 2000.00, 'sports', null),

(2, 'concert: coldplay', '2025-08-10', '20:00:00', 3, 10000, 5000, 1500.00, 'concert', null),

(3, 'nba playoffs', '2025-05-20', '19:30:00', 4, 30000, 10000, 1800.00, 'sports', null),

(4, 'opera night', '2025-07-25', '21:00:00', 3, 2000, 300, 2500.00, 'movie', null),

(5, 'tokyo music festival', '2025-09-05', '17:00:00', 6, 20000, 8000, 1200.00, 'concert', null),

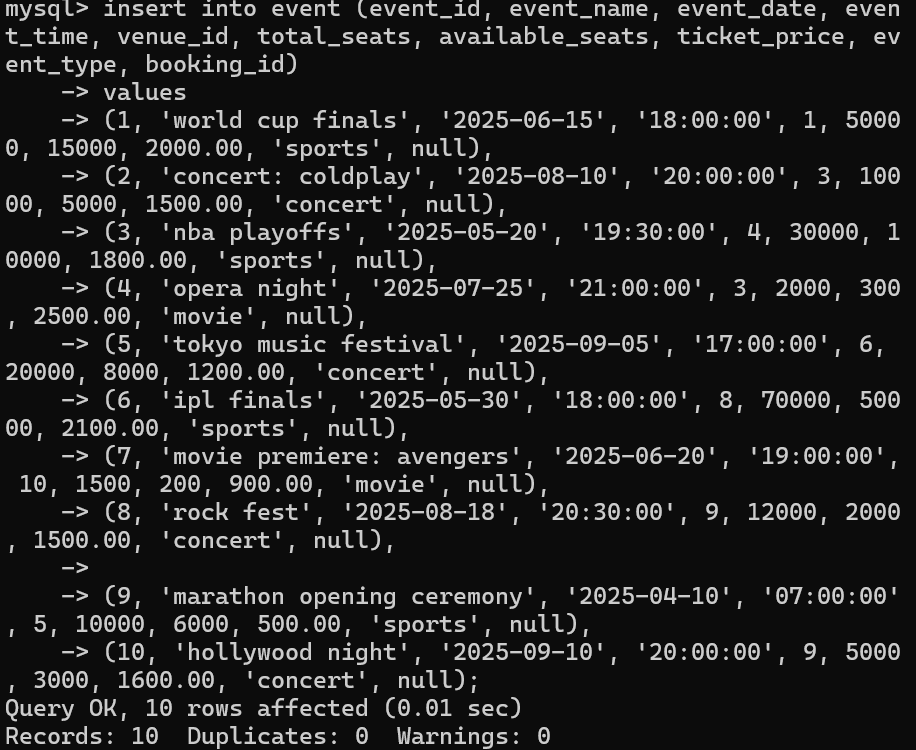
(6, 'ipl finals', '2025-05-30', '18:00:00', 8, 70000, 50000, 2100.00, 'sports', null),

(7, 'movie premiere: avengers', '2025-06-20', '19:00:00', 10, 1500, 200, 900.00, 'movie', null),

(8, 'rock fest', '2025-08-18', '20:30:00', 9, 12000, 2000, 1500.00, 'concert', null),

(9, 'marathon opening ceremony', '2025-04-10', '07:00:00', 5, 10000, 6000, 500.00, 'sports', null),

(10, 'hollywood night', '2025-09-10', '20:00:00', 9, 5000, 3000, 1600.00, 'concert', null);



Customer

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

values

(1, 'john doe', 'john.doe@example.com', '1234567890', null),

(2, 'alice smith', 'alice.smith@example.com', '9876543210', null),

(3, 'bob brown', 'bob.brown@example.com', '1122334455', null),

(4, 'michael scott', 'michael.scott@example.com', '9988776655', null),

(5, 'pam beesly', 'pam.beesly@example.com', '8899776655', null),

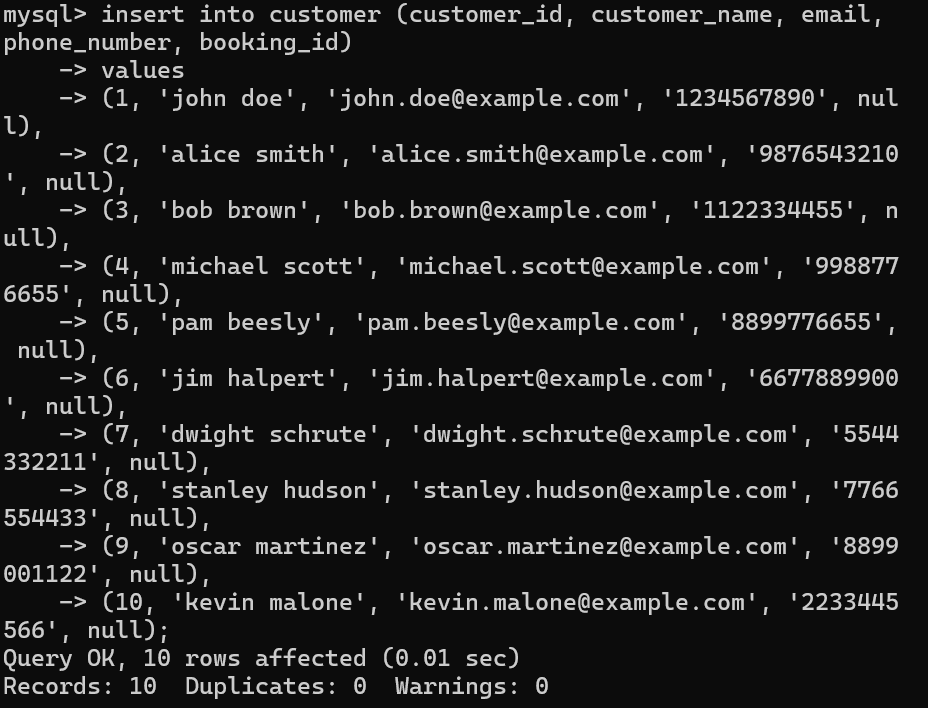
(6, 'jim halpert', 'jim.halpert@example.com', '6677889900', null),

(7, 'dwight schrute', 'dwight.schrute@example.com', '5544332211', null),

(8, 'stanley hudson', 'stanley.hudson@example.com', '7766554433', null),

(9, 'oscar martinez', 'oscar.martinez@example.com', '8899001122', null),

(10, 'kevin malone', 'kevin.malone@example.com', '2233445566', null);



Booking

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

values

(1, 1, 1, 5, 10000.00, '2025-05-01 12:00:00'),

(2, 2, 2, 2, 3000.00, '2025-05-02 14:00:00'),

(3, 3, 3, 8, 14400.00, '2025-05-03 16:00:00'),

(4, 4, 4, 4, 10000.00, '2025-05-04 10:00:00'),

(5, 5, 5, 6, 7200.00, '2025-05-05 18:00:00'),

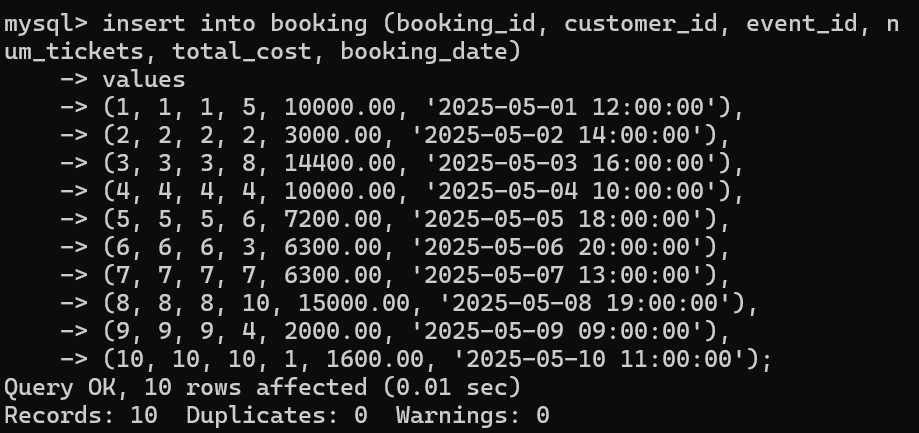
(6, 6, 6, 3, 6300.00, '2025-05-06 20:00:00'),

(7, 7, 7, 7, 6300.00, '2025-05-07 13:00:00'),

(8, 8, 8, 10, 15000.00, '2025-05-08 19:00:00'),

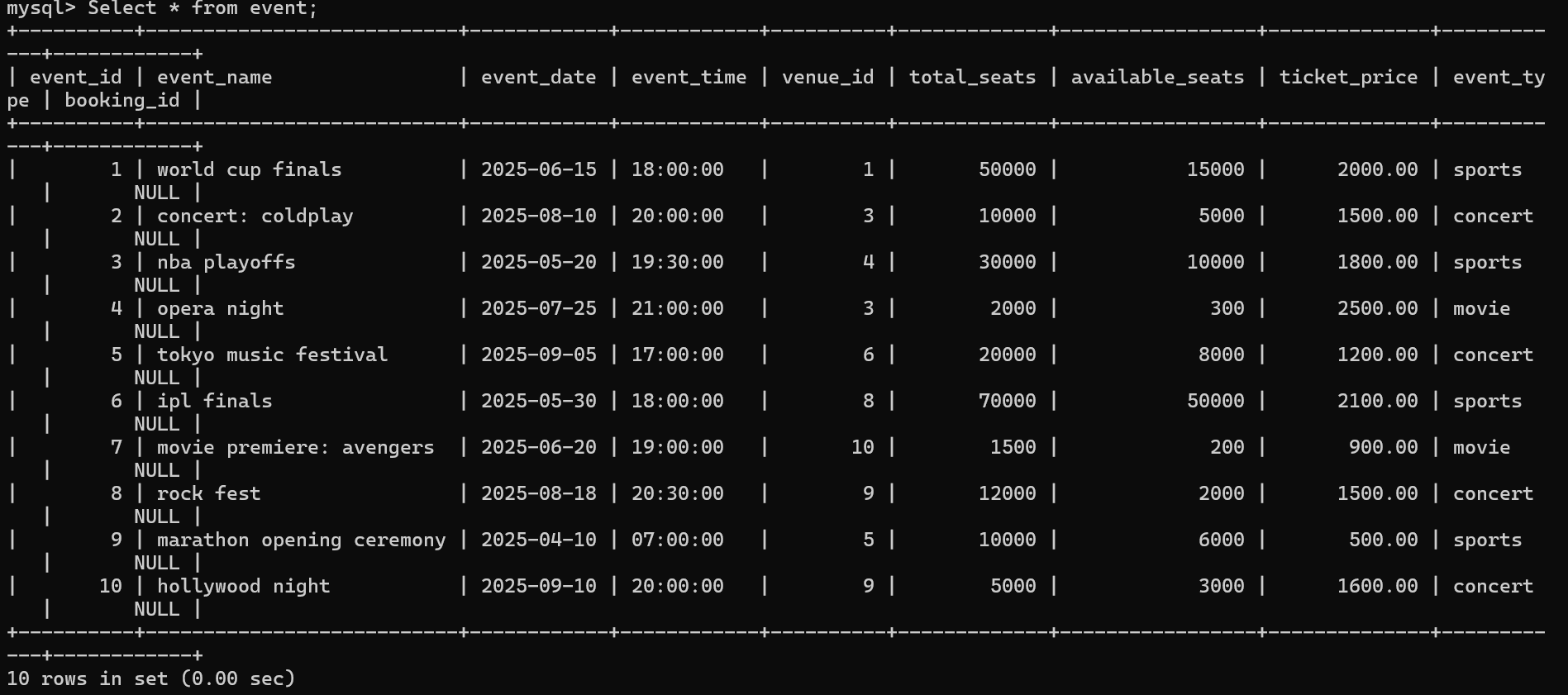
(9, 9, 9, 4, 2000.00, '2025-05-09 09:00:00'),

(10, 10, 10, 1, 1600.00, '2025-05-10 11:00:00');



2. Write a SQL query to list all Events.

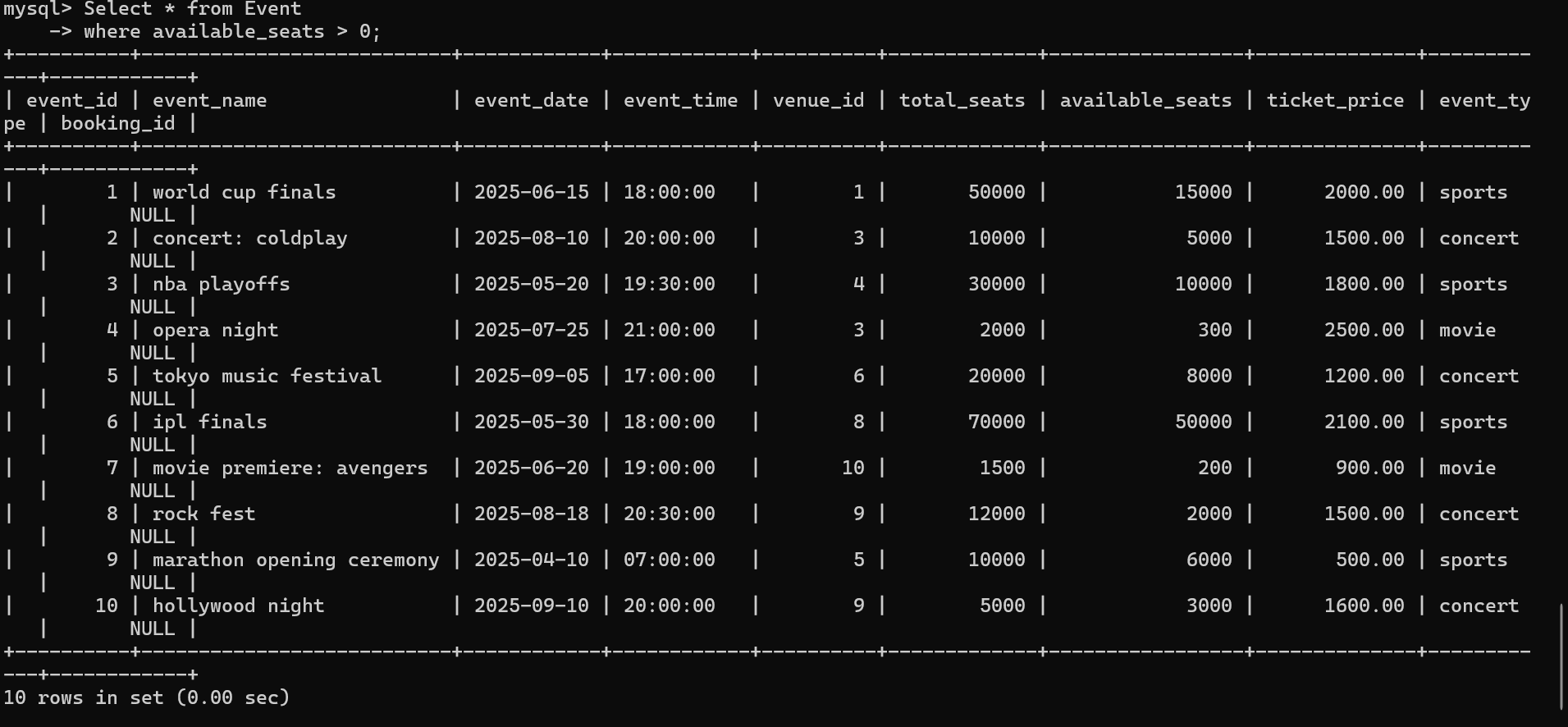
Select \* from event;



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

Select \* from Event

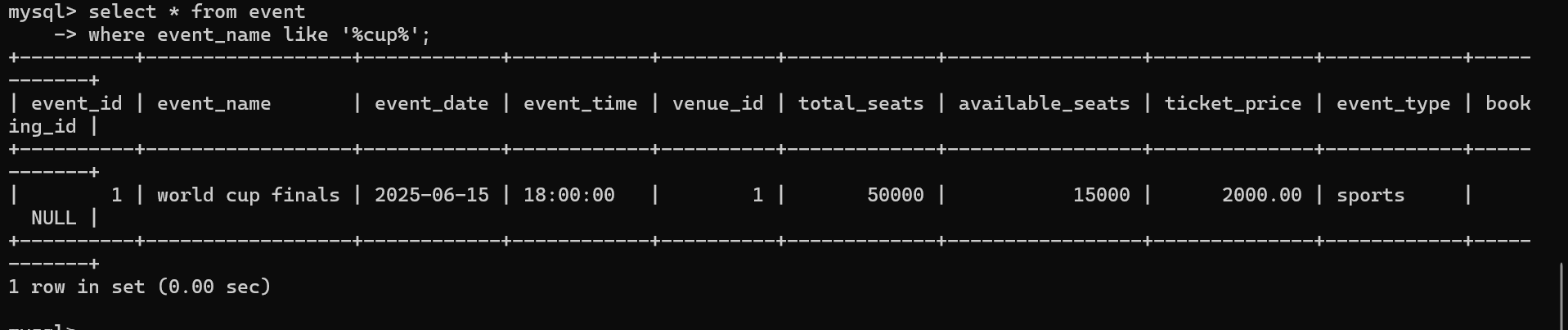
where available\_seats > 0;



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

select \* from event

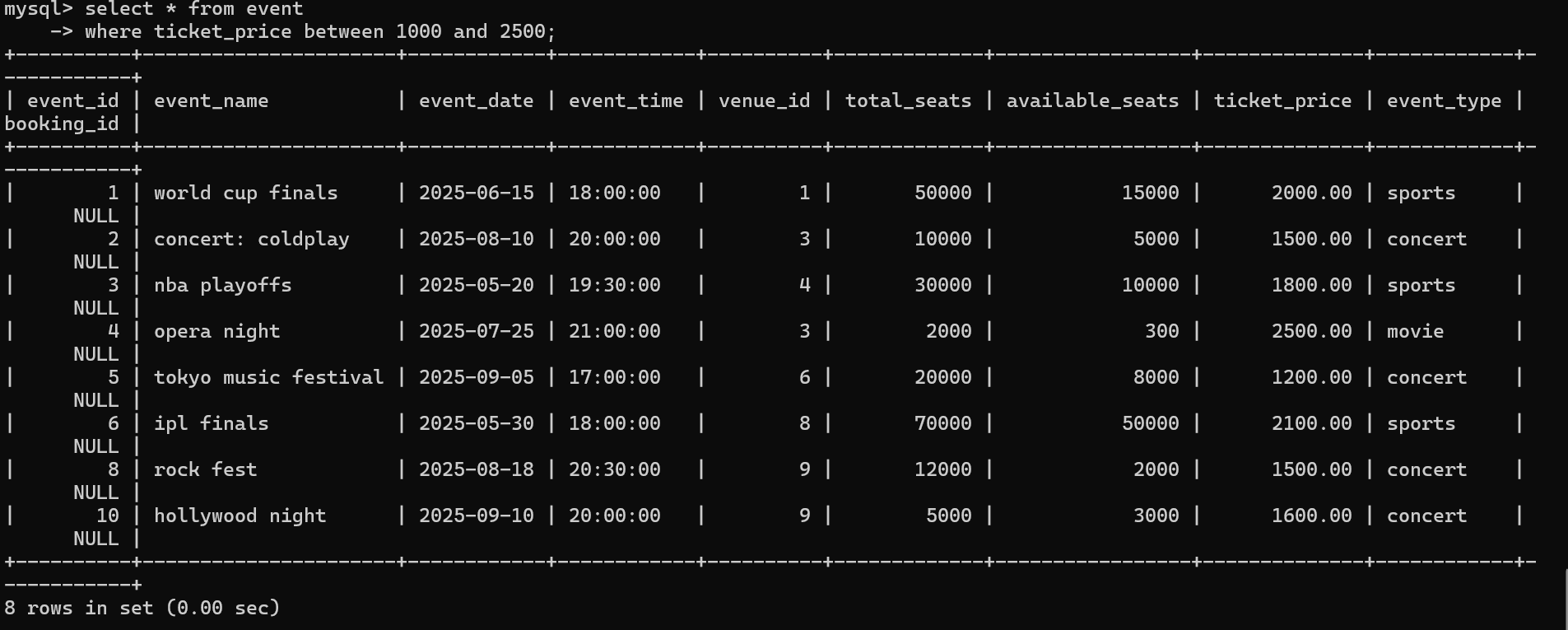
where event\_name like '%cup%';



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

select \* from event

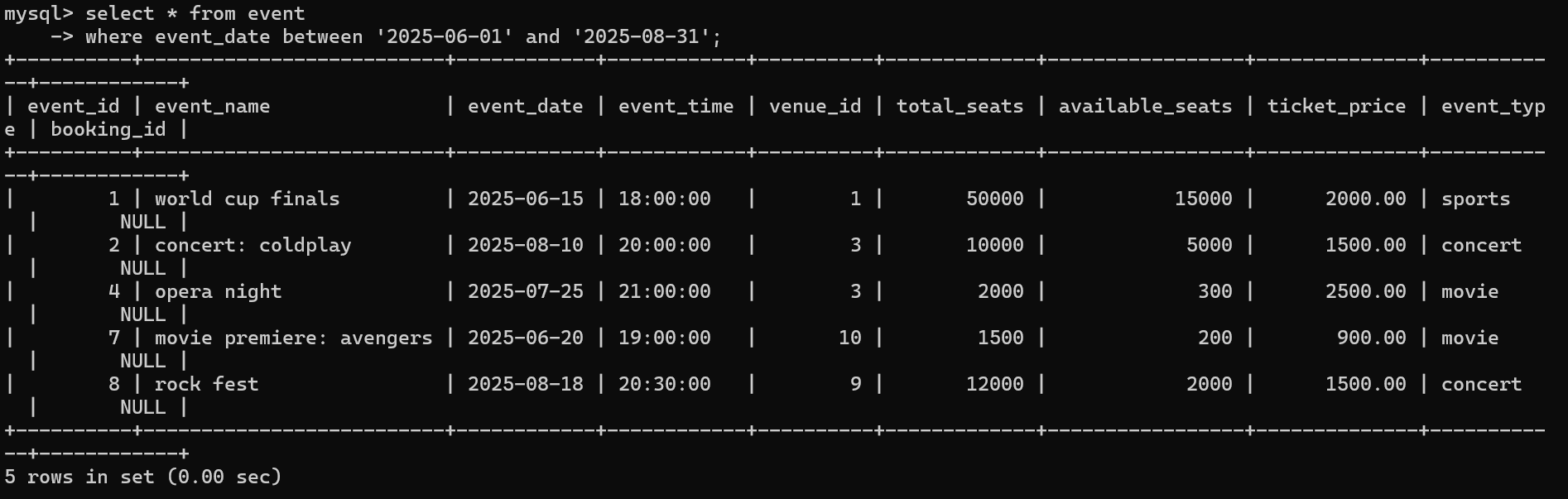
where ticket\_price between 1000 and 2500;



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

select \* from event

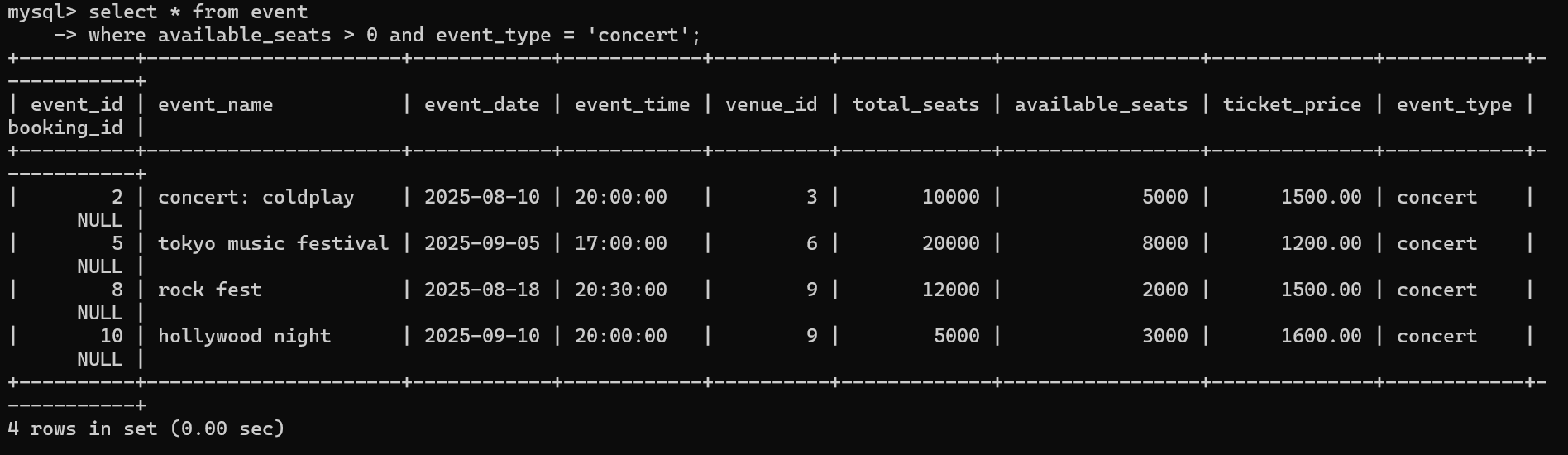
where event\_date between '2025-06-01' and '2025-08-31';



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

select \* from event

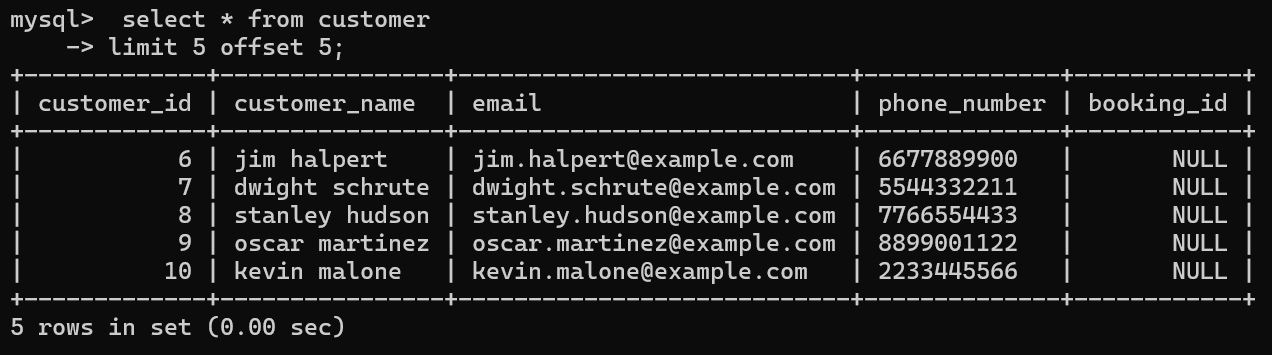
where available\_seats > 0 and event\_type = 'concert';



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

select \* from customer

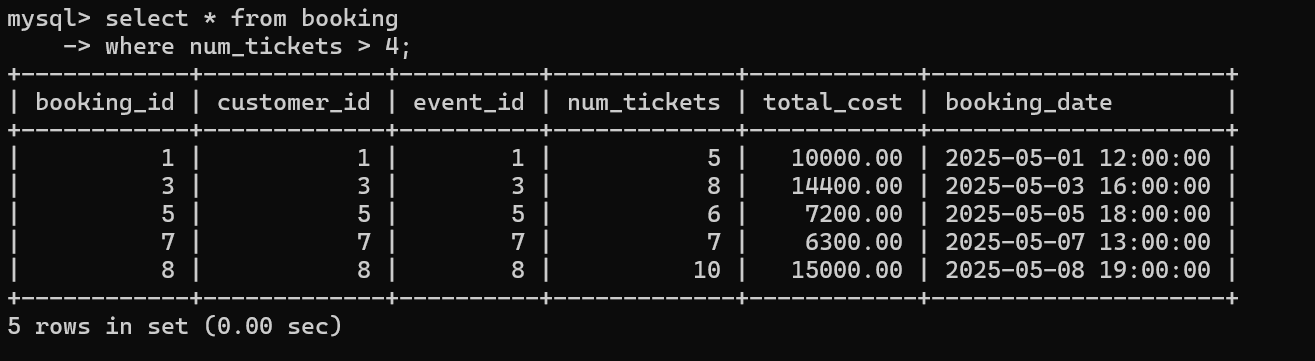
limit 5 offset 5;



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

select \* from booking

where num\_tickets > 4;

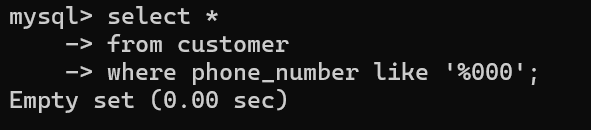


10. Write a SQL query to retrieve customer information whose phone number end with ‘000’

select \*

from customer

where phone\_number like '%000';



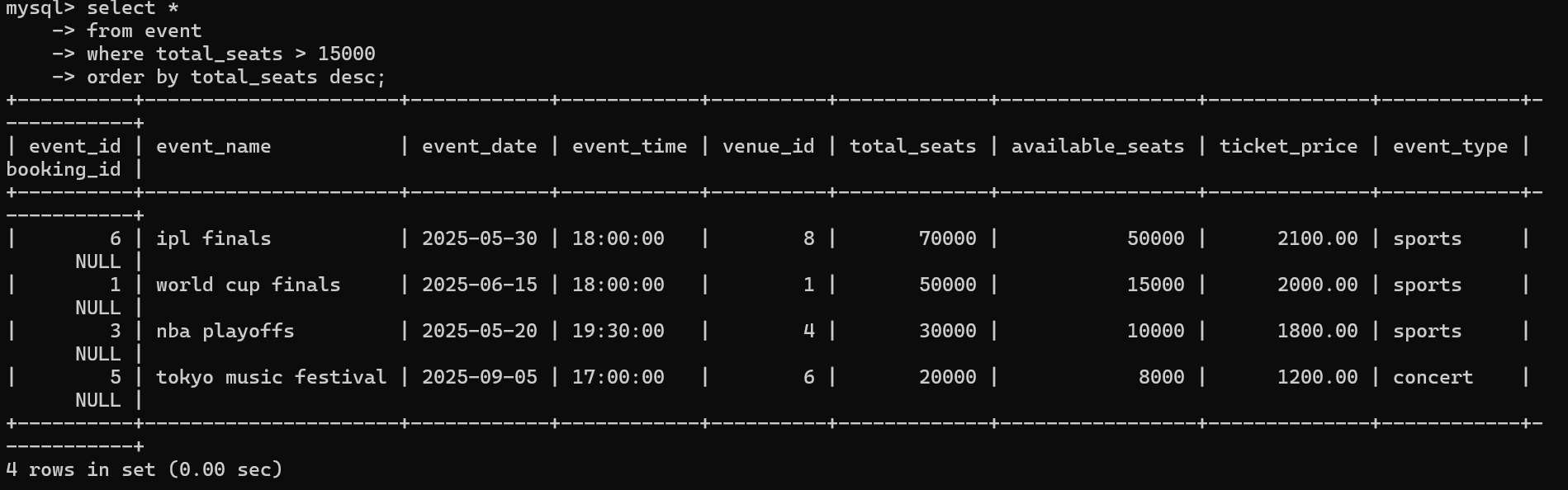
11. Write a SQL query to retrieve the events in order whose seat capacity more than 15000.

select \*

from event

where total\_seats > 15000

order by total\_seats desc;



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

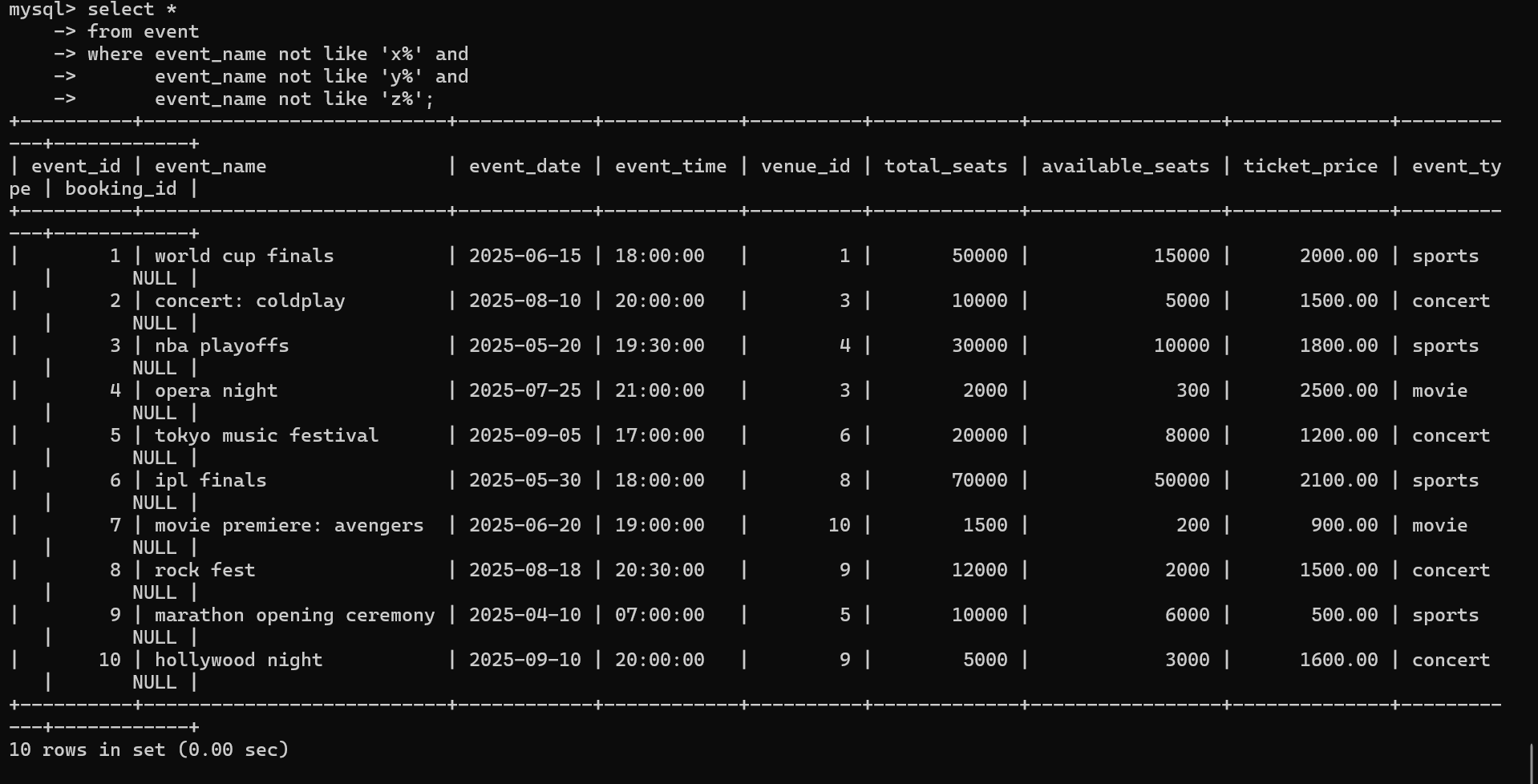
select \*

from event

where event\_name not like 'x%' and

event\_name not like 'y%' and

event\_name not like 'z%';



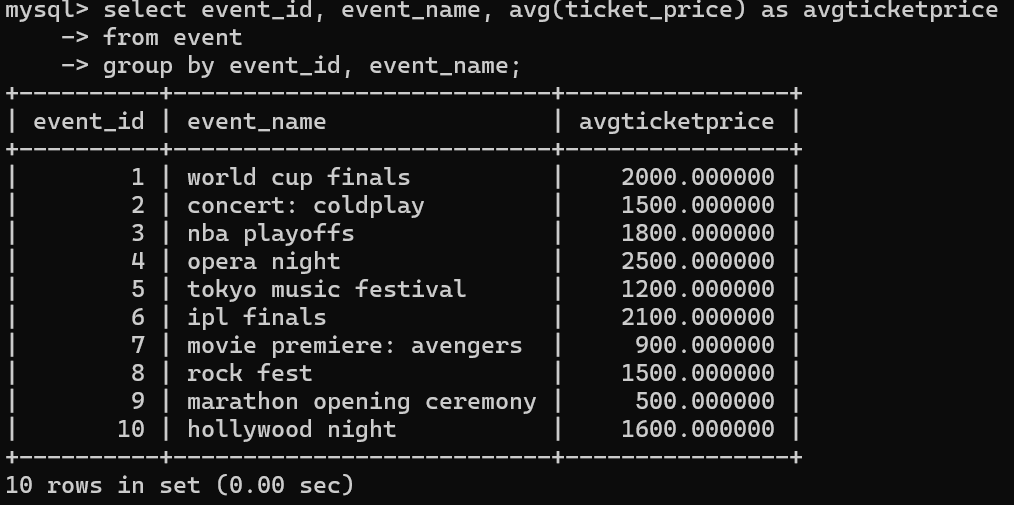
**Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:**

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

select event\_id, event\_name, avg(ticket\_price) as avgticketprice

from event

group by event\_id, event\_name;



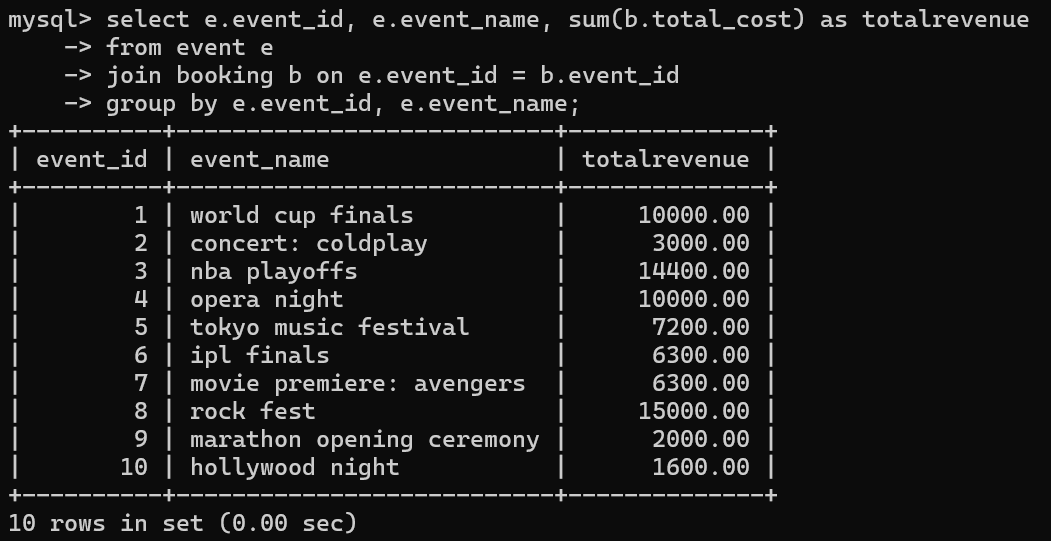
2. Write a SQL query to Calculate the Total Revenue Generated by Events.

select e.event\_id, e.event\_name, sum(b.total\_cost) as totalrevenue

from event e

join booking b on e.event\_id = b.event\_id

group by e.event\_id, e.event\_name;



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

select e.event\_id, e.event\_name, sum(b.num\_tickets) as totaltickets

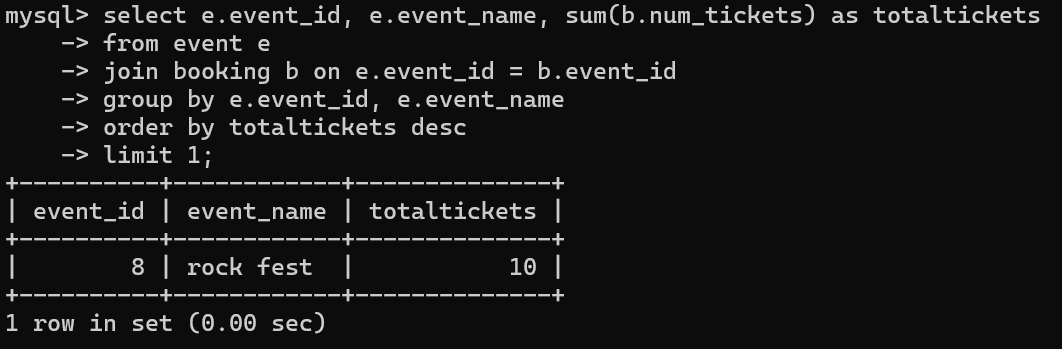
from event e

join booking b on e.event\_id = b.event\_id

group by e.event\_id, e.event\_name

order by totaltickets desc

limit 1;



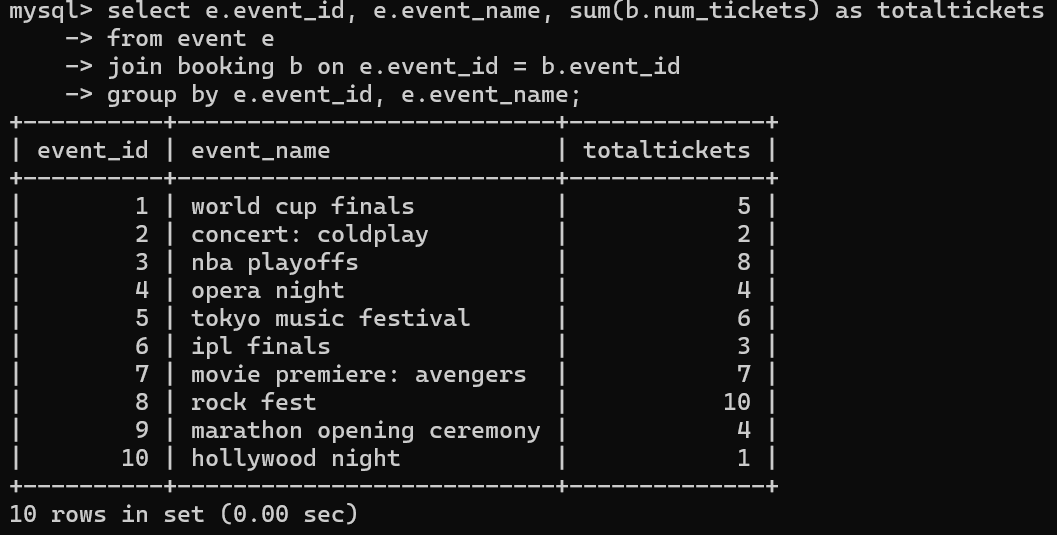
4. Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.

select e.event\_id, e.event\_name, sum(b.num\_tickets) as totaltickets

from event e

join booking b on e.event\_id = b.event\_id

group by e.event\_id, e.event\_name;



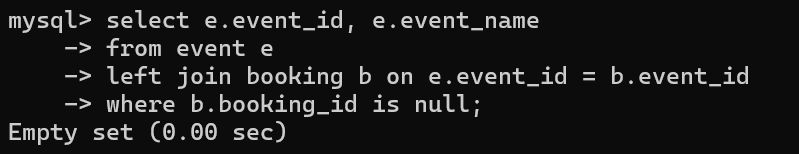
5. Write a SQL query to Find Events with No Ticket Sales.

select e.event\_id, e.event\_name

from event e

left join booking b on e.event\_id = b.event\_id

where b.booking\_id is null;



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

select c.customer\_id, c.customer\_name, sum(b.num\_tickets) as totaltickets

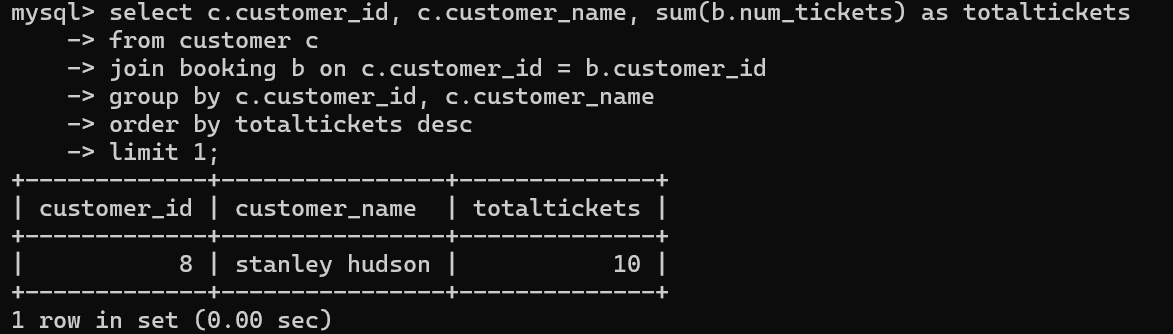
from customer c

join booking b on c.customer\_id = b.customer\_id

group by c.customer\_id, c.customer\_name

order by totaltickets desc

limit 1;



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

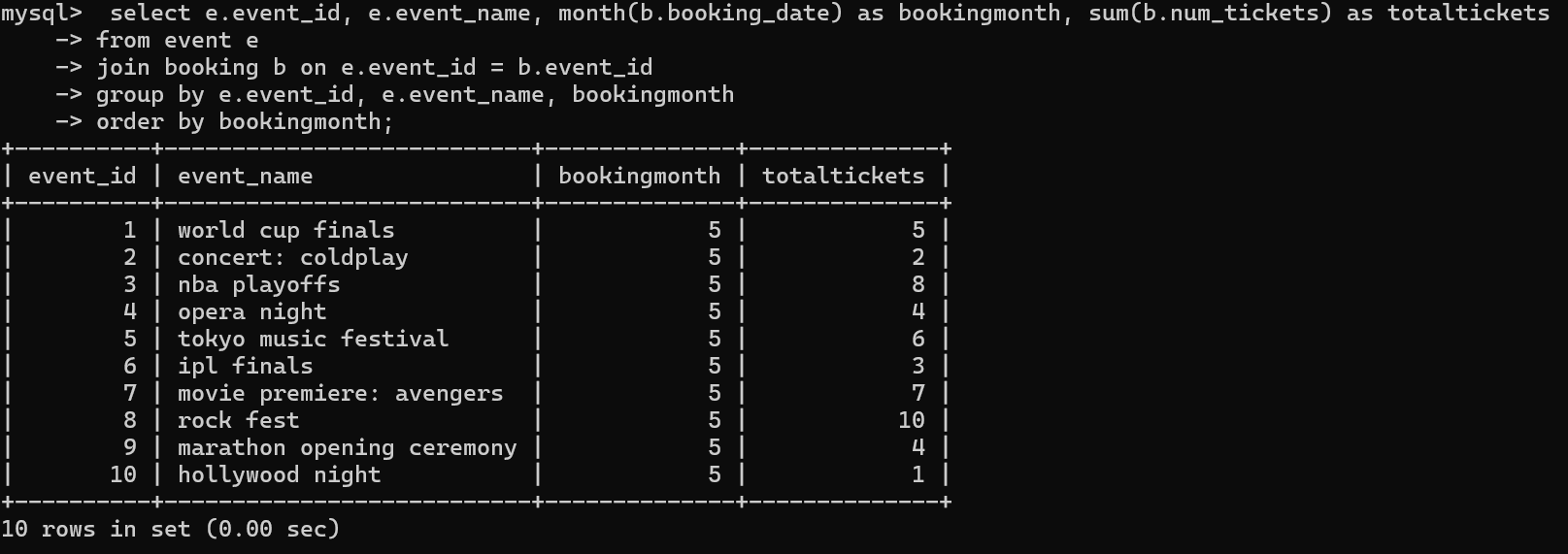
select e.event\_id, e.event\_name, month(b.booking\_date) as bookingmonth, sum(b.num\_tickets) as totaltickets

from event e

join booking b on e.event\_id = b.event\_id

group by e.event\_id, e.event\_name, bookingmonth

order by bookingmonth;



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

select v.venue\_id, v.venue\_name, avg(e.ticket\_price) as avgticketprice

from venue v

join event e on v.venue\_id = e.venue\_id

group by v.venue\_id, v.venue\_name;

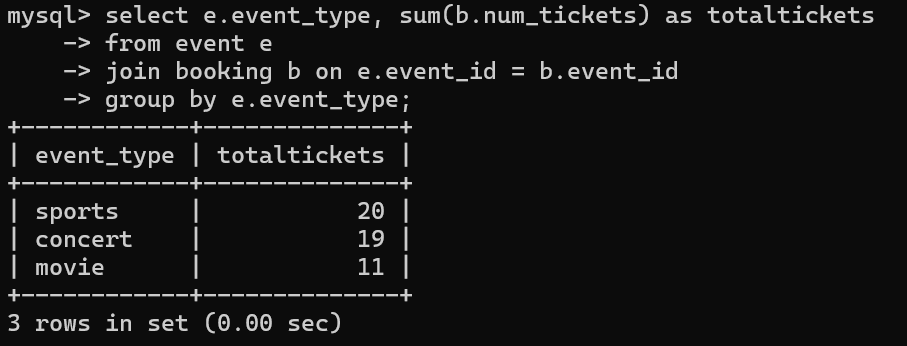
9. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.

select e.event\_type, sum(b.num\_tickets) as totaltickets

from event e

join booking b on e.event\_id = b.event\_id

group by e.event\_type;



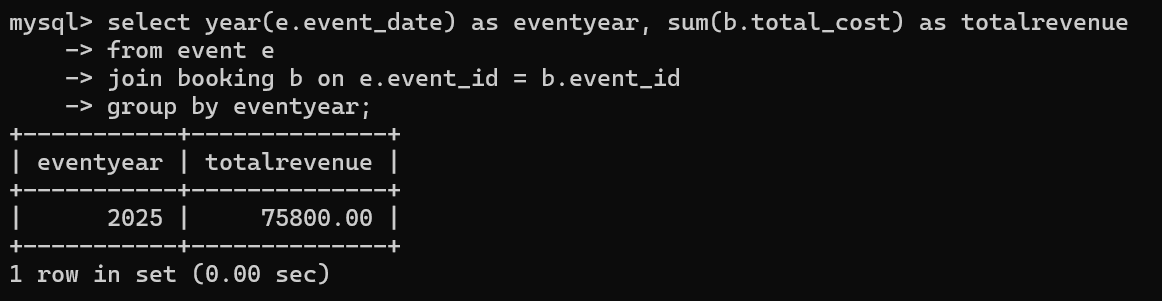
10. Write a SQL query to calculate the total Revenue Generated by Events in Each Year.

select year(e.event\_date) as eventyear, sum(b.total\_cost) as totalrevenue

from event e

join booking b on e.event\_id = b.event\_id

group by eventyear;



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

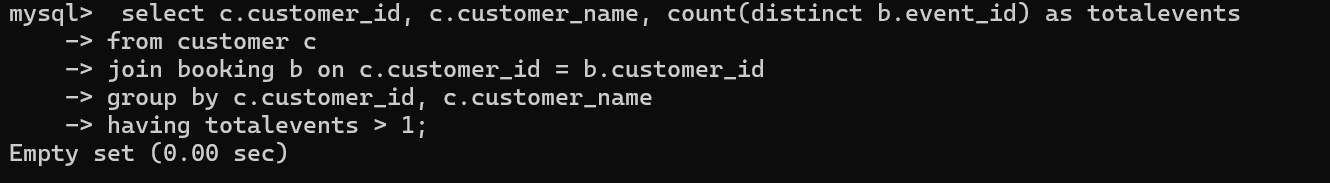
select c.customer\_id, c.customer\_name, count(distinct b.event\_id) as totalevents

from customer c

join booking b on c.customer\_id = b.customer\_id

group by c.customer\_id, c.customer\_name

having totalevents > 1;



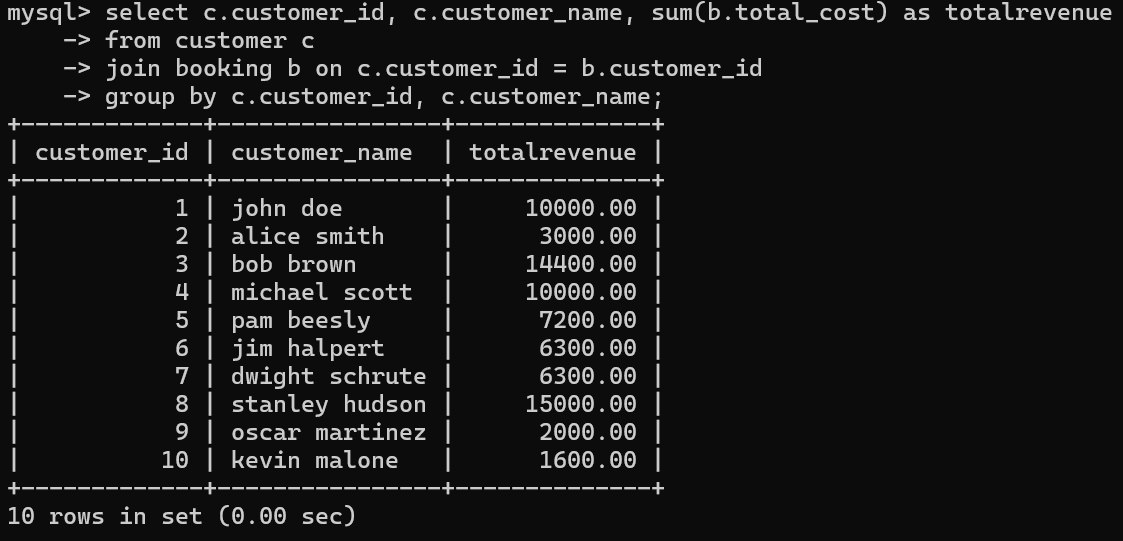
12. Write a SQL query to calculate the Total Revenue Generated by Events for Each User.

select c.customer\_id, c.customer\_name, sum(b.total\_cost) as totalrevenue

from customer c

join booking b on c.customer\_id = b.customer\_id

group by c.customer\_id, c.customer\_name;



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

select e.event\_type, v.venue\_name, avg(e.ticket\_price) as avgticketprice

from event e

join venue v on e.venue\_id = v.venue\_id

group by e.event\_type, v.venue\_name;

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

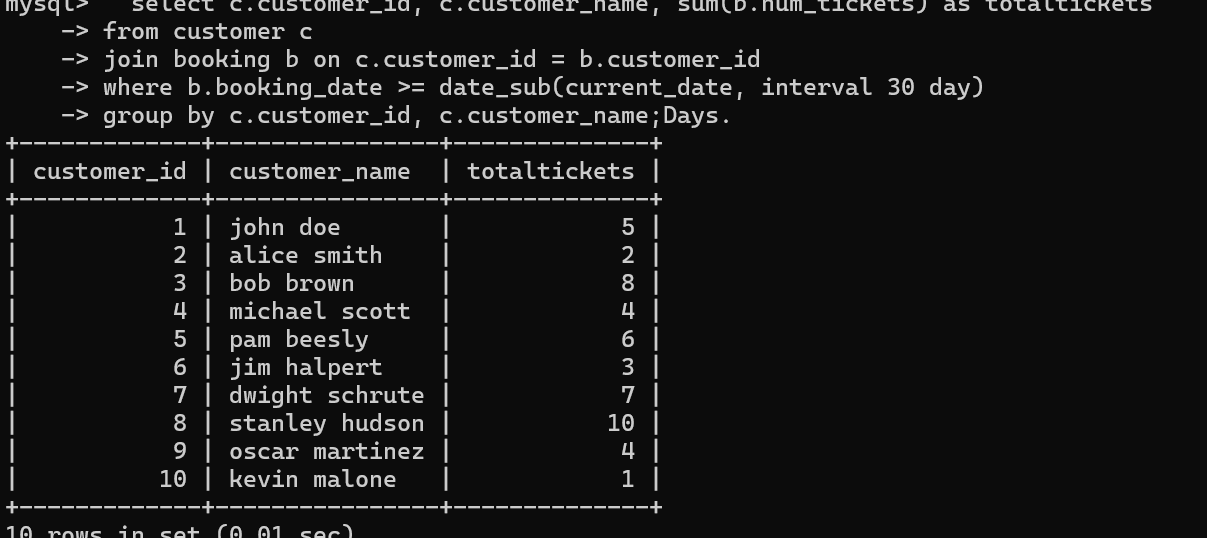
select c.customer\_id, c.customer\_name, sum(b.num\_tickets) as totaltickets

from customer c

join booking b on c.customer\_id = b.customer\_id

where b.booking\_date >= date\_sub(current\_date, interval 30 day)

group by c.customer\_id, c.customer\_name;



**Tasks 4:Subquery and its types**

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

select v.venue\_id, v.venue\_name,

(select avg(e.ticket\_price)

from event e

where e.venue\_id = v.venue\_id) as avgticketprice

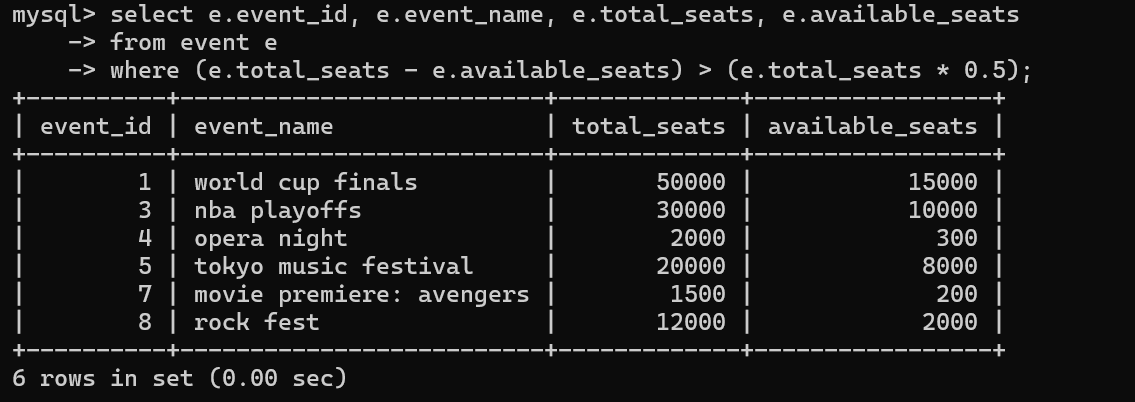
from venu v;

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

select e.event\_id, e.event\_name, e.total\_seats, e.available\_seats

from event e

where (e.total\_seats - e.available\_seats) > (e.total\_seats \* 0.5);



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

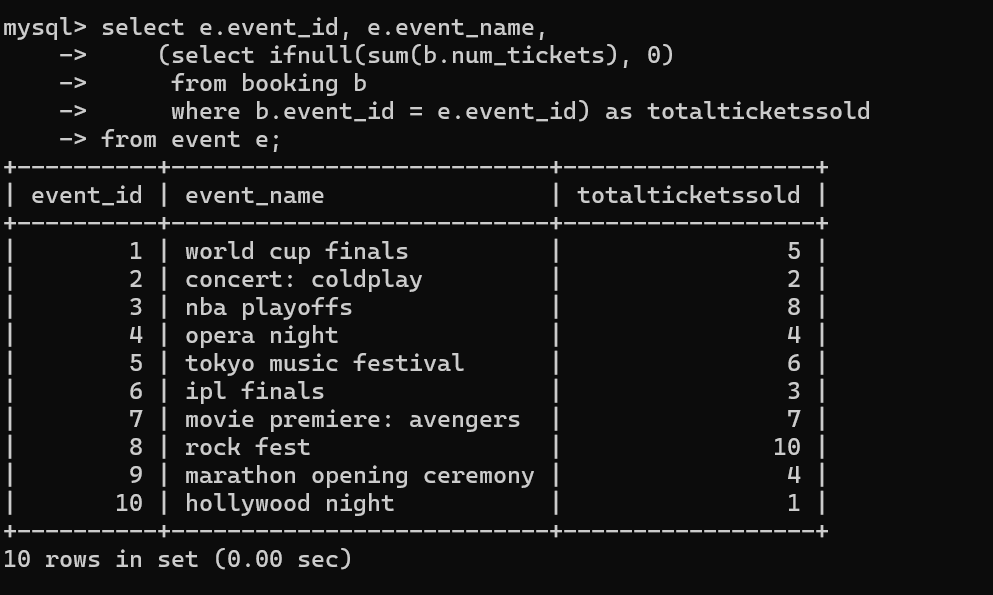
select e.event\_id, e.event\_name,

(select ifnull(sum(b.num\_tickets), 0)

from booking b

where b.event\_id = e.event\_id) as totalticketssold

from event e;



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

select c.customer\_id, c.customer\_name

from customer c

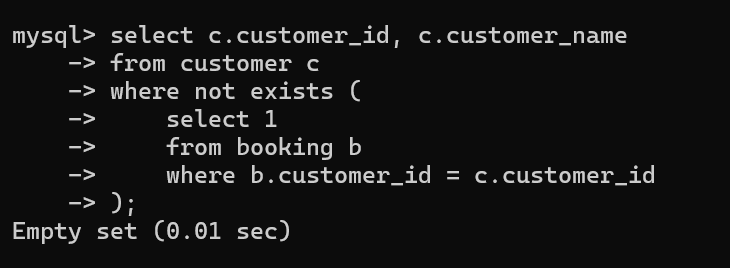
where not exists (

select 1

from booking b

where b.customer\_id = c.customer\_id

);



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

select e.event\_id, e.event\_name

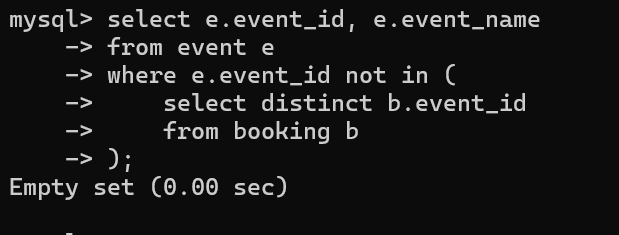
from event e

where e.event\_id not in (

select distinct b.event\_id

from booking b

);



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

select subquery.event\_type, sum(subquery.totaltickets) as totalticketssold

from (

select e.event\_type, ifnull(sum(b.num\_tickets), 0) as totaltickets

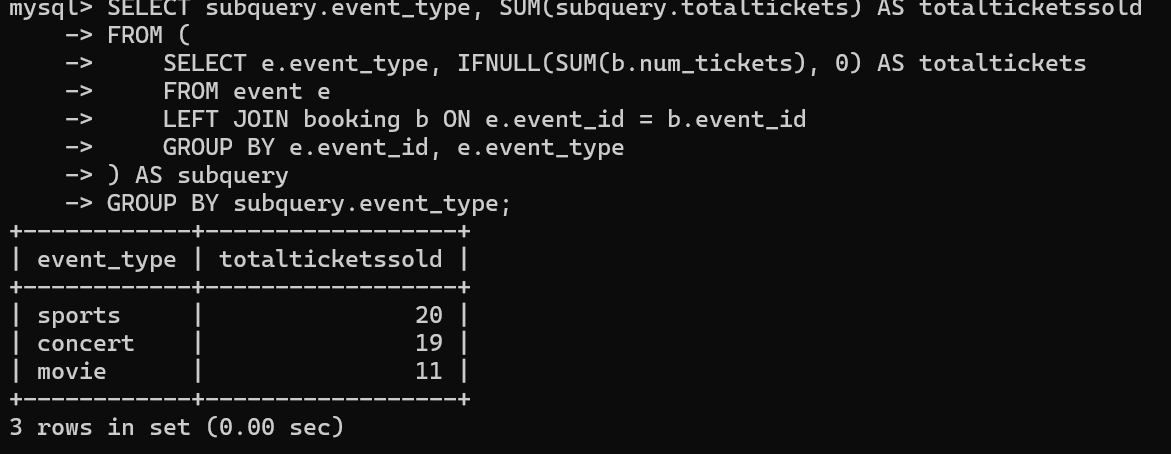
from event e

left join booking b on e.event\_id = b.event\_id

group by e.event\_id, e.event\_type

) as subquery

group by subquery.event\_type;



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

select e.event\_id, e.event\_name, e.ticket\_price

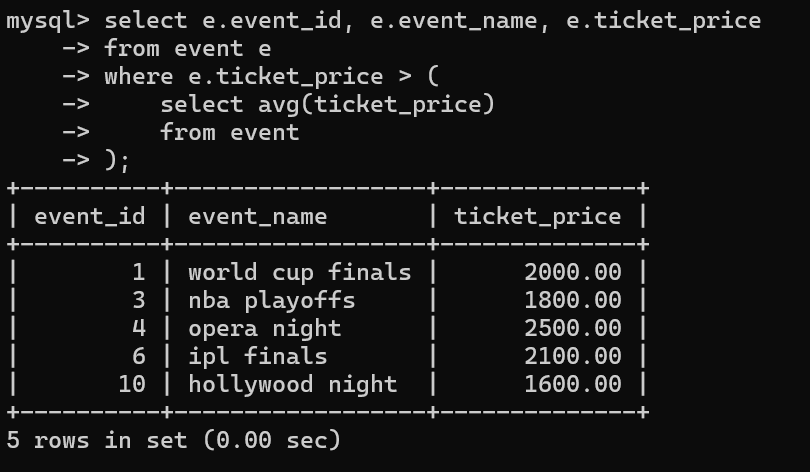
from event e

where e.ticket\_price > (

select avg(ticket\_price)

from event

);



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

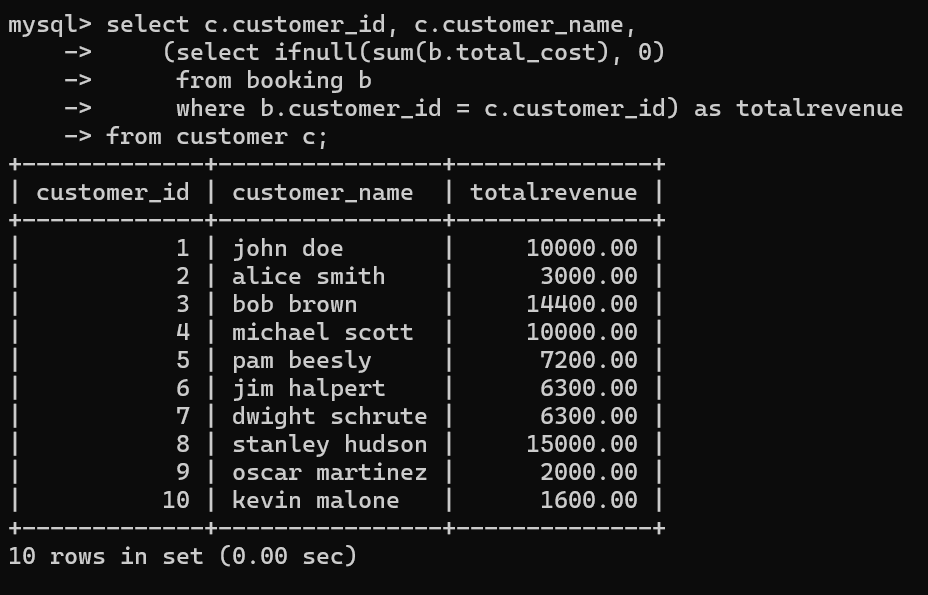
select c.customer\_id, c.customer\_name,

(select ifnull(sum(b.total\_cost), 0)

from booking b

where b.customer\_id = c.customer\_id) as totalrevenue

from customer c;



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

Clause.

select c.customer\_id, c.customer\_name

from customer c

where c.customer\_id in (

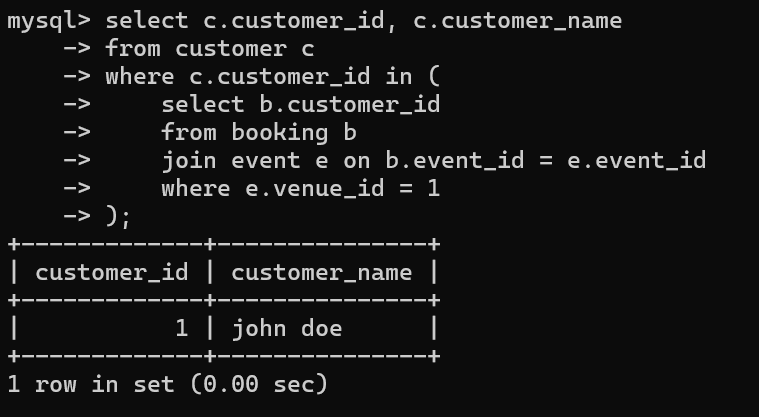
select b.customer\_id

from booking b

join event e on b.event\_id = e.event\_id

where e.venue\_id = 1

);



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

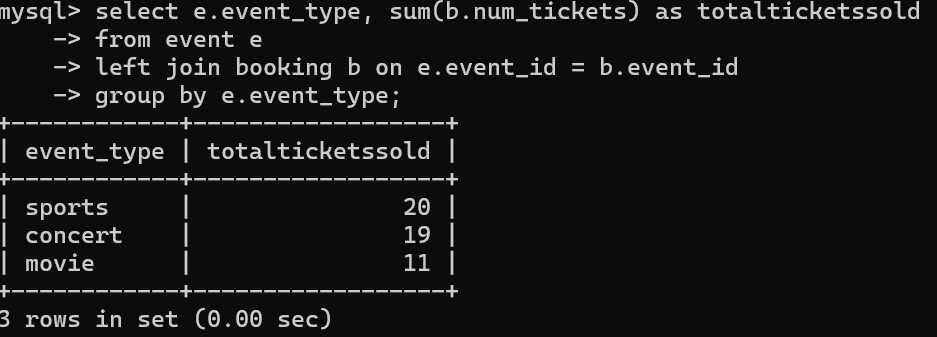
GROUP BY.

select e.event\_type, sum(b.num\_tickets) as totalticketssold

from event e

left join booking b on e.event\_id = b.event\_id

group by e.event\_type;



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

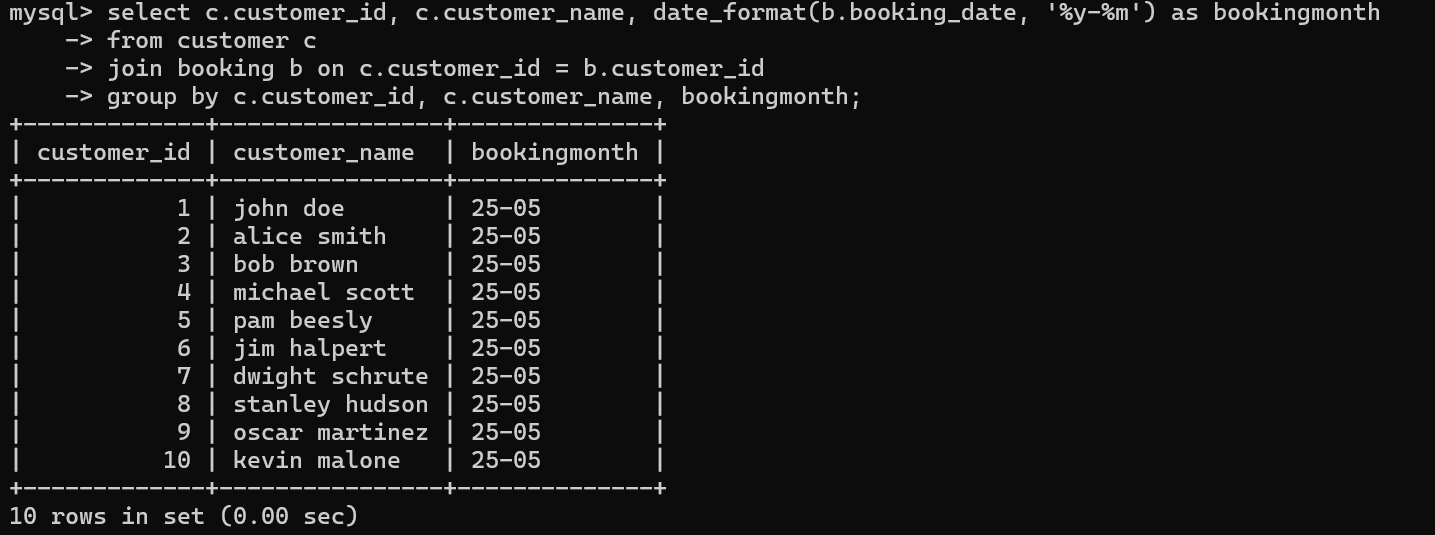
DATE\_FORMAT.

select c.customer\_id, c.customer\_name, date\_format(b.booking\_date, '%y-%m') as bookingmonth

from customer c

join booking b on c.customer\_id = b.customer\_id

group by c.customer\_id, c.customer\_name, bookingmonth;



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

select v.venue\_id, v.venue\_name,

(select avg(e.ticket\_price)

from event e

where e.venue\_id = v.venue\_id) as avgticketprice

from venu v;

