TASK 1:

Creating databases

1. create database ticketBookingSystem;

2.use ticketBookingSystem;

creating tables

create table Venue (

venue\_id int primary key auto\_increment,

venue\_name varchar(255),

address text

);

create table Event (

event\_id int primary key auto\_increment,

event\_name varchar(255),

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

);

create table Customer (

customer\_id int primary key auto\_increment,

customer\_name varchar(255),

email varchar(255) unique,

phone\_number varchar(15) unique,

booking\_id int

);

create table Booking (

booking\_id int primary key auto\_increment,

customer\_id int,

event\_id int,

num\_tickets int,

total\_cost decimal(10,2),

booking\_date datetime default current\_timestamp

);

Adding references

alter table Event

add foreign key (venue\_id) references Venue(venue\_id) on delete cascade,

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

alter table Customer

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

alter table Booking

add foreign key (customer\_id) references Customer(customer\_id) on delete cascade,

add foreign key (event\_id) references Event(event\_id) on delete cascade;

TASK 2:SELECT WHERE BETWEEN LIKE AND:

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

**Insert into table venue**

insert into Venue (venue\_name, address) values

('Grand Arena', '123 Main Street, NY'),

('Skyline Stadium', '45 Park Avenue, LA'),

('Ocean View Theater', '88 Beach Road, FL'),

('Sunset Concert Hall', '56 Sunset Blvd, TX'),

('City Sports Complex', '77 Downtown St, IL'),

('The Empire Hall', '90 King Street, NJ'),

('Greenwood Arena', '101 Oakwood Lane, CO'),

('Blue Lagoon Center', '203 River Road, WA'),

('Silver Dome', '305 Mountain Ave, AZ'),

('Golden Plaza', '409 Gold Street, NV');

**Insert Data into Event Table**

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

('Alice Johnson', 'alice@gmail.com', '1234567890'),

('Bob Smith', 'bob@gmail.com', '2345678901'),

('Charlie Brown', 'charlie@gmail.com', '3456789012'),

('David Lee', 'david@gmail.com', '4567890123'),

('Emma Wilson', 'emma@gmail.com', '5678901234'),

('Frank Harris', 'frank@gmail.com', '6789012345'),

('Grace Moore', 'grace@gmail.com', '7890123456'),

('Henry Clark', 'henry@gmail.com', '8901234567'),

('Isabella Martinez', 'isabella@gmail.com', '9012345678'),

('Jack White', 'jack@gmail.com', '0123456789');

**Insert Data into Booking Table**

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

(1, 1, 2, 199.98, '2025-04-01 12:00:00'),

(2, 2, 4, 600.00, '2025-04-02 14:30:00'),

(3, 3, 1, 12.50, '2025-04-03 10:45:00'),

(4, 4, 3, 150.00, '2025-04-04 16:20:00'),

(5, 5, 2, 160.00, '2025-04-05 11:15:00'),

(6, 6, 1, 120.00, '2025-04-06 17:50:00'),

(7, 7, 5, 375.00, '2025-04-07 09:30:00'),

(8, 8, 2, 80.00, '2025-04-08 13:10:00'),

(9, 9, 3, 75.00, '2025-04-09 15:40:00'),

(10, 10, 4, 380.00, '2025-04-10 19:00:00');

**Update Event and Customer Tables to Link Bookings**

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;

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

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

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

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

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

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

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

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

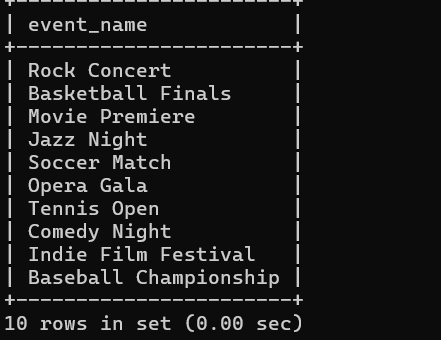
update Customer set booking\_id = 9 where customer\_id = 9;

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

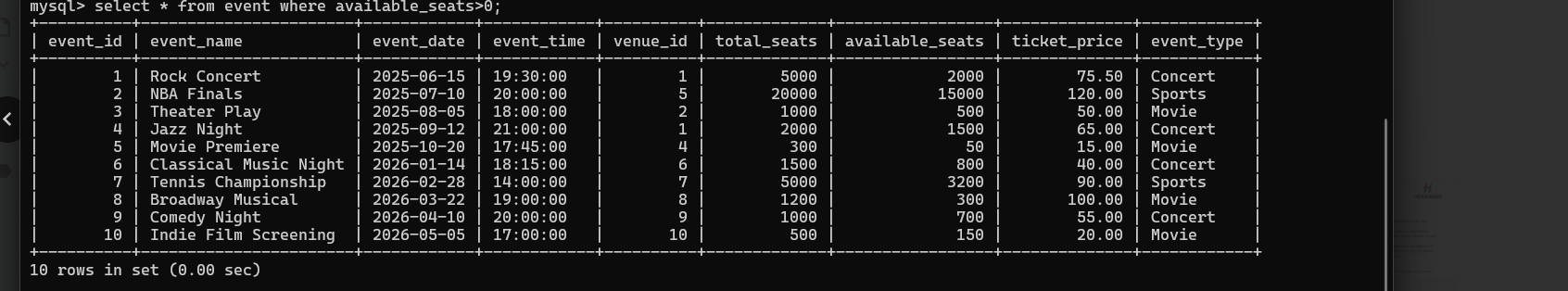
2.Write a sql query to list all events

Select \* from event;

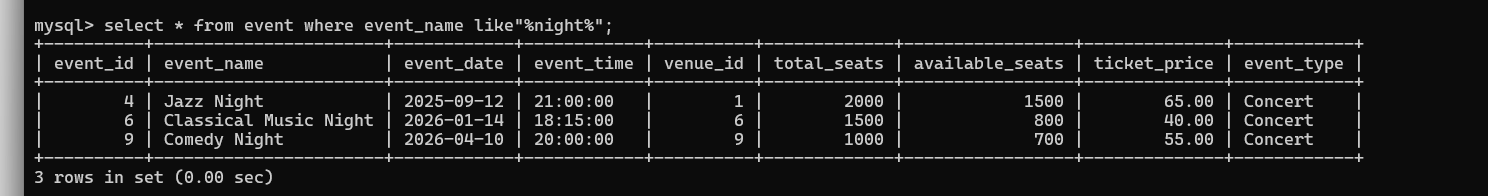
Output



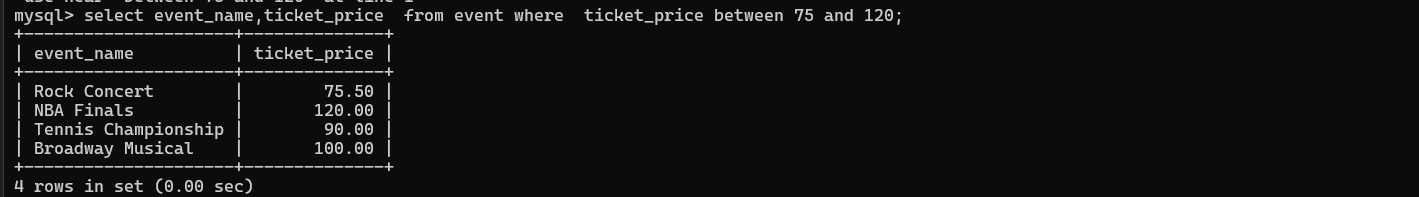
**3.Write a SQL query to select events with available tickets.**



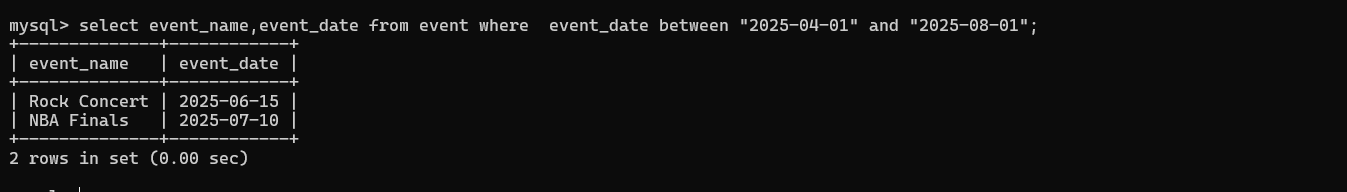
**4. Write a SQL query to select events name partial match with ‘cup’.**

****

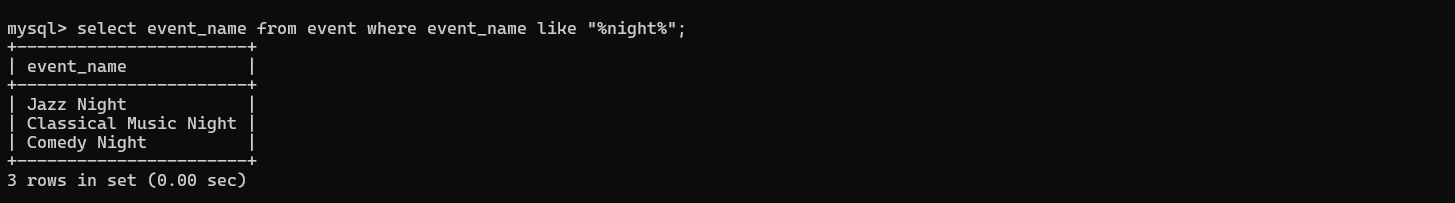
**5. Write a SQL query to select events with ticket price range is between 1000 to 2500.**

****

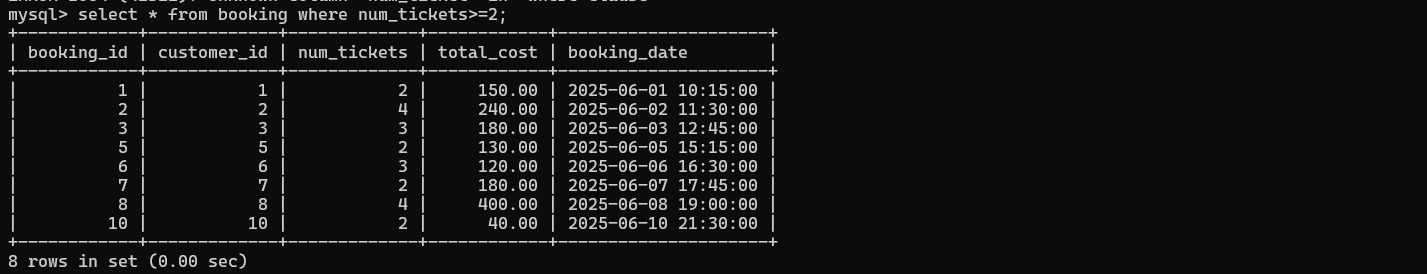
**6. Write a SQL query to retrieve events with dates falling within a specific range.**



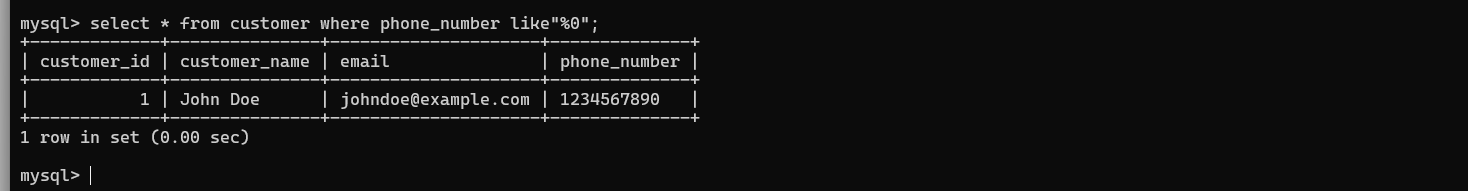
**7.**  **Write a SQL query to retrieve events with available tickets that also have "Concert" in their name**

****

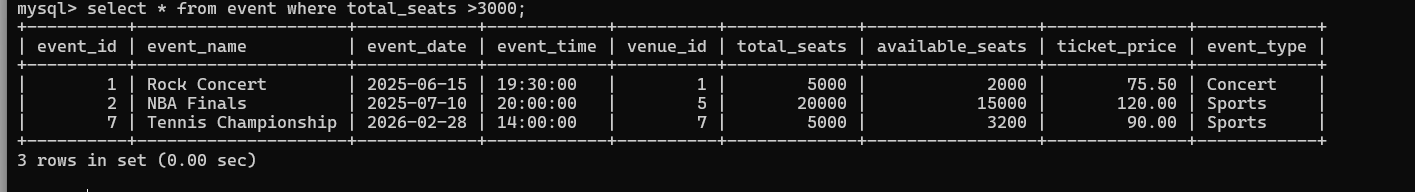
**9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.**

****

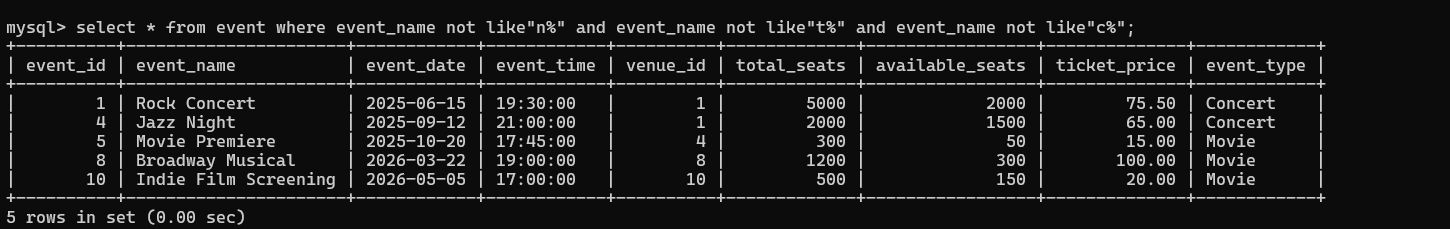
**10. Write a SQL query to retrieve customer information whose phone number end with ‘000’**

****

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

****

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

****

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

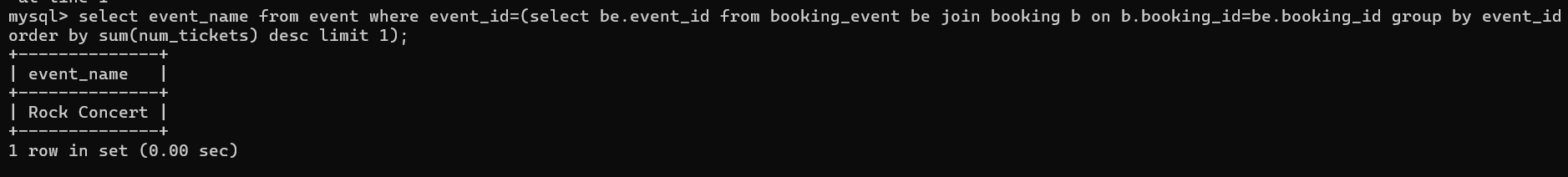
1. **Write a SQL query to List Events and Their Average Ticket Prices.**



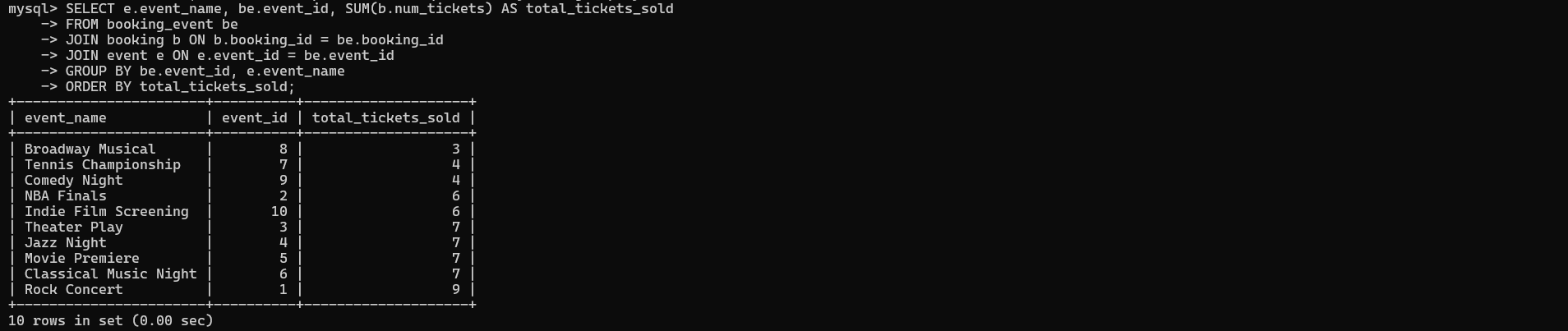
**2. Write a SQL query to Calculate the Total Revenue Generated by Events.**

****

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



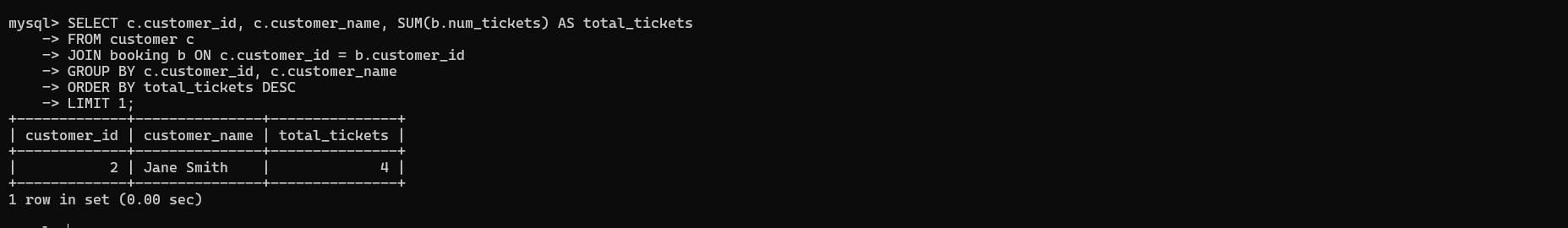
**4. Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.**



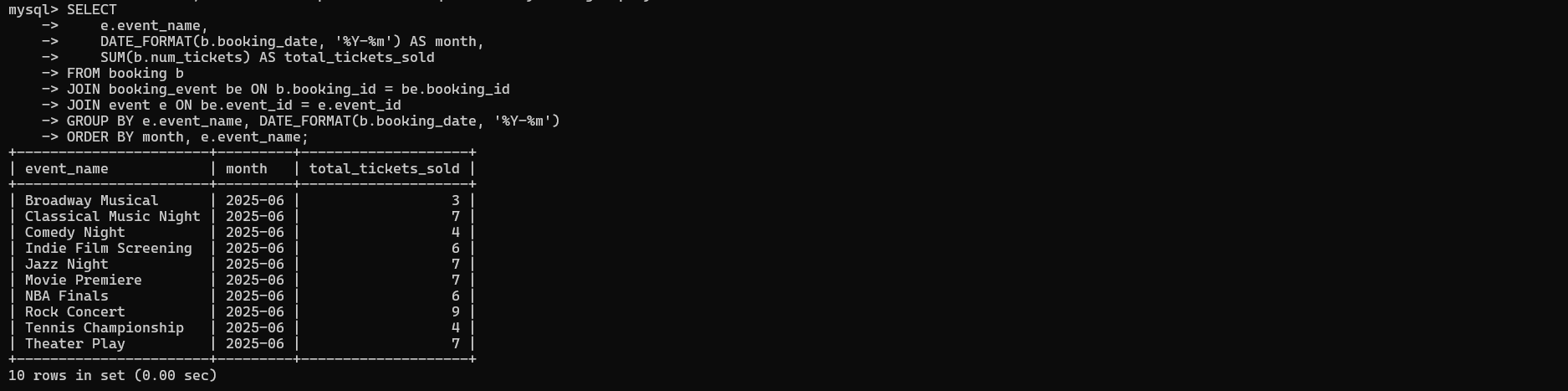
**5. Write a SQL query to Find Events with No Ticket Sales.**



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

****

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

****

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

****

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

****

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



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



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



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

****

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

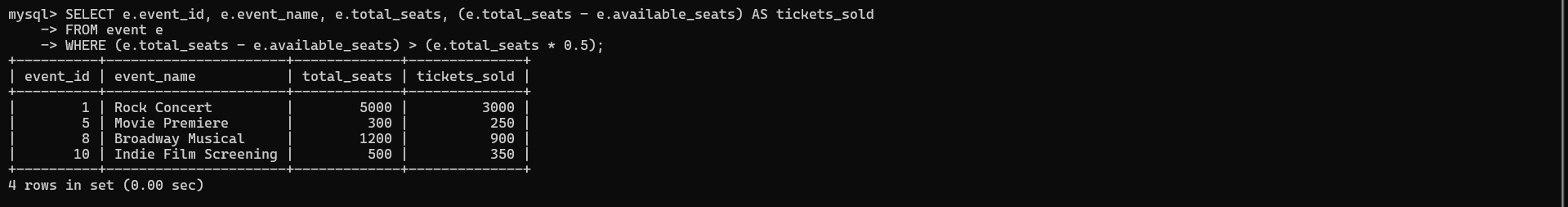
****

**Tasks 4: Subquery and its types**

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

****

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

****

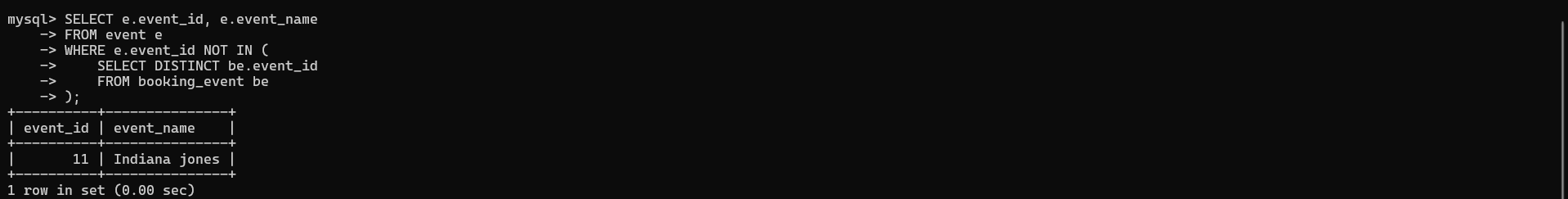
**3.Calculate the Total Number of Tickets Sold for Each Event.**



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

****

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

****

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

****

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

**WHERE Clause.**

****

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

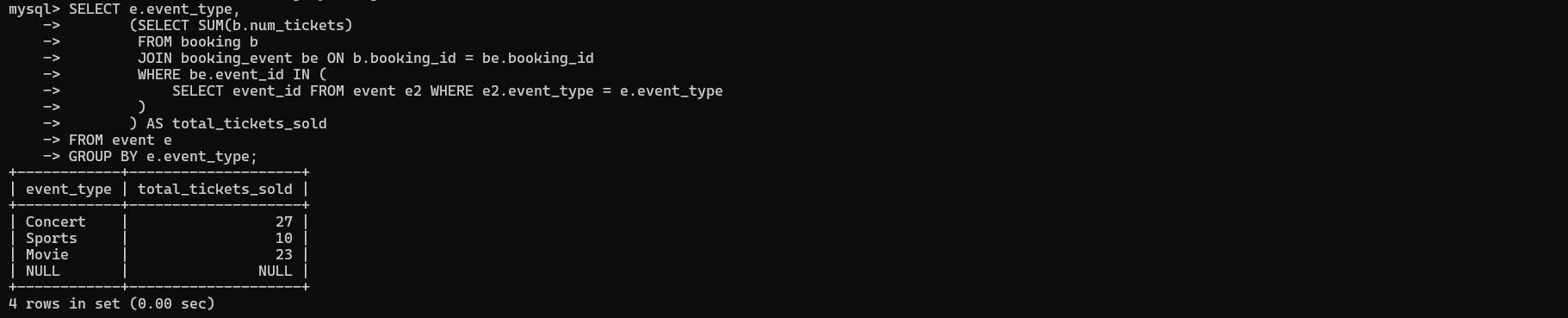


**9. List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause.**

****

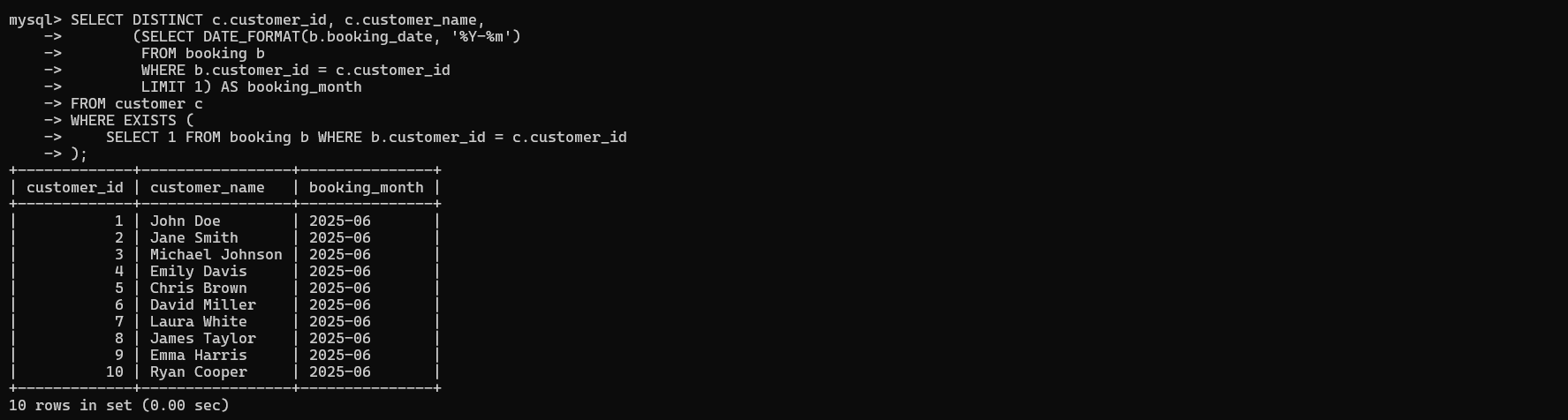
**10. Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with**

**GROUP BY.**

****

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

**DATE\_FORMAT.**



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

