TICKET BOOKING SYSTEM

Task-1

Creation of database

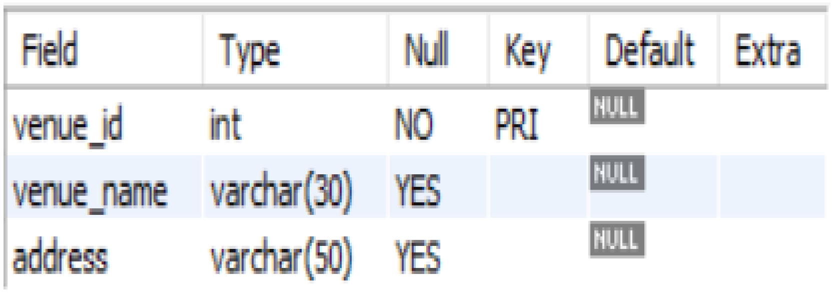
```
create database TicketBookingSystem;
```

To use the database

```
use TicketBookingSystem;
```

Creation of table venu

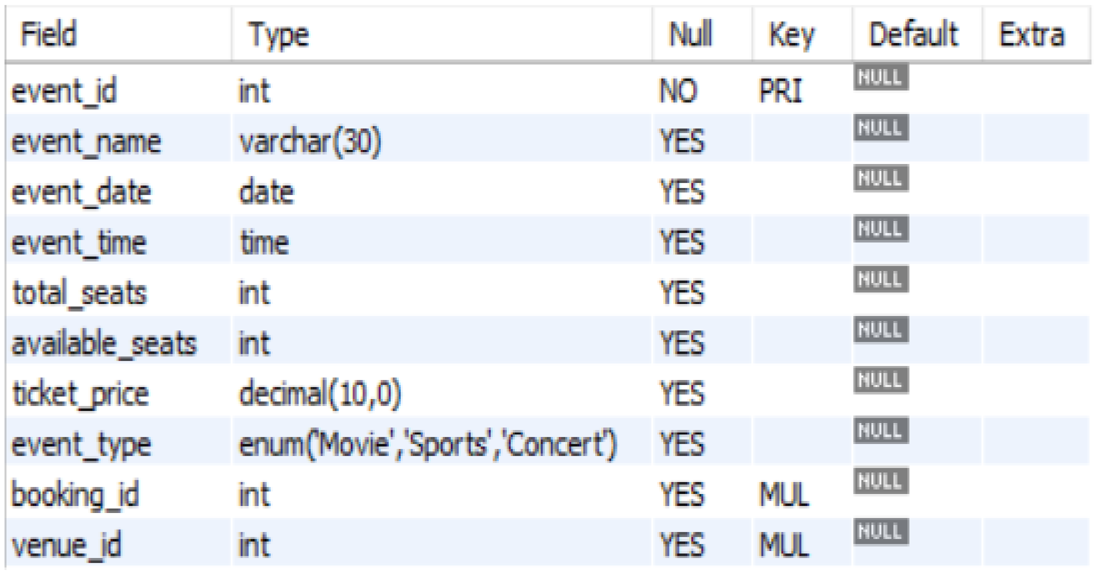
```
create table Venu
(
venue_id int primary key,
venue_name varchar(30),
address varchar(50)
);
```



Field	Type	Null	Key	Default	Extra
venue_id	int	NO	PRI	NULL	
venue_name	varchar(30)	YES		NULL	
address	varchar(50)	YES		NULL	

Creation of table event

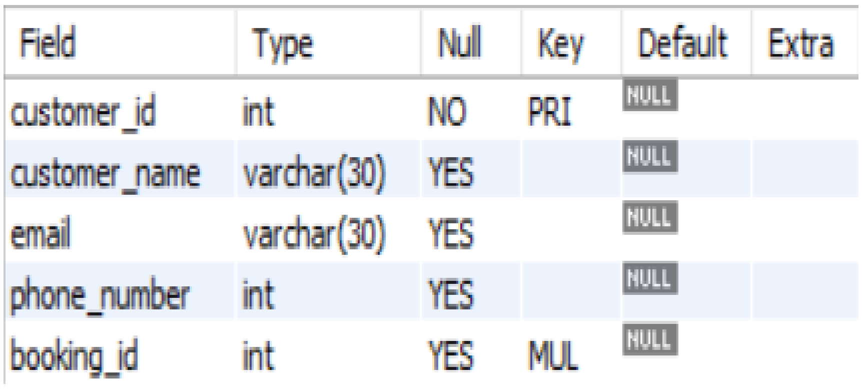
```
create table Event
(
event_id int primary key,
event_name varchar(30),
event_date date,
event_time time,
total_seats int,

available_seats int,

ticket_price decimal,

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

booking_id int,

venue_id int,

foreign key (venue_id) references Venu(venue_id)

);
```



Field	Type	Null	Key	Default	Extra
event_id	int	NO	PRI	NULL	
event_name	varchar(30)	YES		NULL	
event_date	date	YES		NULL	
event_time	time	YES		NULL	
total_seats	int	YES		NULL	
available_seats	int	YES		NULL	
ticket_price	decimal(10,0)	YES		NULL	
event_type	enum('Movie','Sports','Concert')	YES		NULL	
booking_id	int	YES	MUL	NULL	
venue_id	int	YES	MUL	NULL	

Creation of table customer

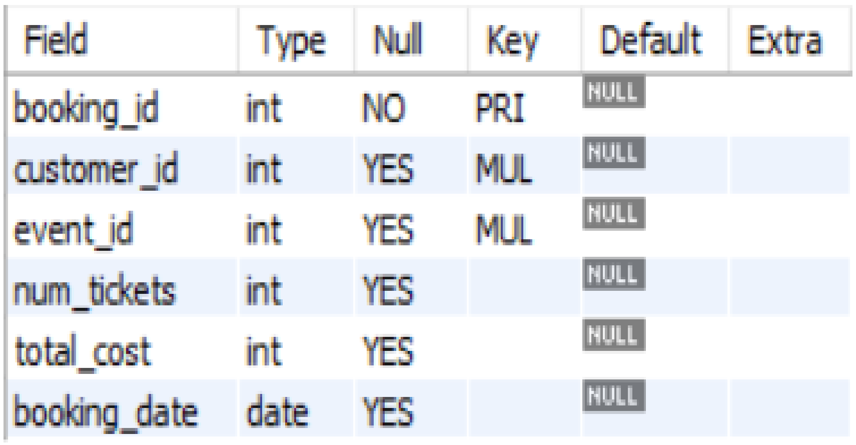
```
create table Customer

(

customer_id int primary key,

customer_name varchar(30),

email varchar(30),

phone_number varchar(15),

booking_id int);
```



Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	
customer_name	varchar(30)	YES		NULL	
email	varchar(30)	YES		NULL	
phone_number	int	YES		NULL	
booking_id	int	YES	MUL	NULL	

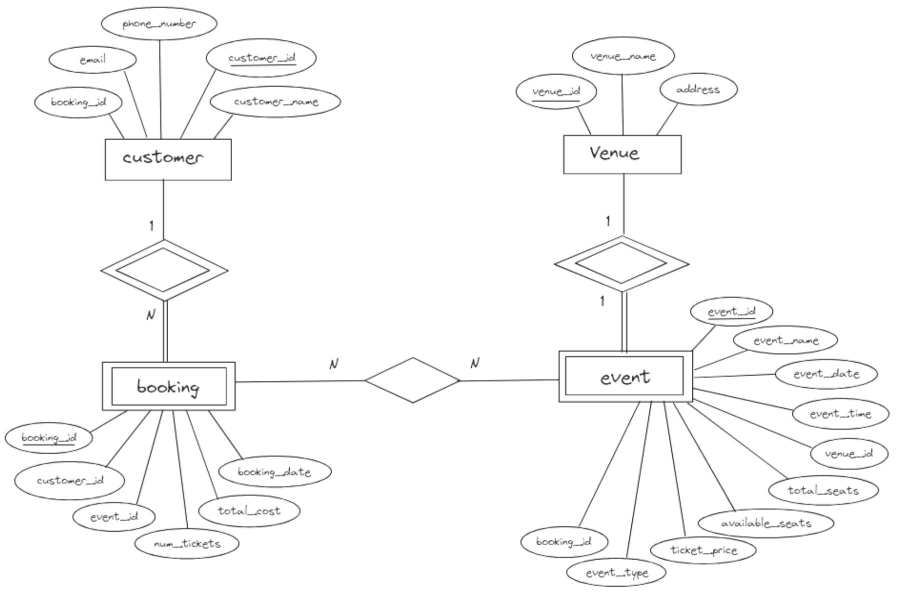
Creation of table booking

```
create table Booking
(
booking_id int primary key,
customer_id int,
foreign key(customer_id) references Customer(customer_id),
event_id int,
foreign key(event_id) references Event(event_id),
num_tickets int,
total_cost decimal,
booking_date date
);
```



Field	Type	Null	Key	Default	Extra
booking_id	int	NO	PRI	NULL	
customer_id	int	YES	MUL	NULL	
event_id	int	YES	MUL	NULL	
num_tickets	int	YES		NULL	
total_cost	int	YES		NULL	
booking_date	date	YES		NULL	

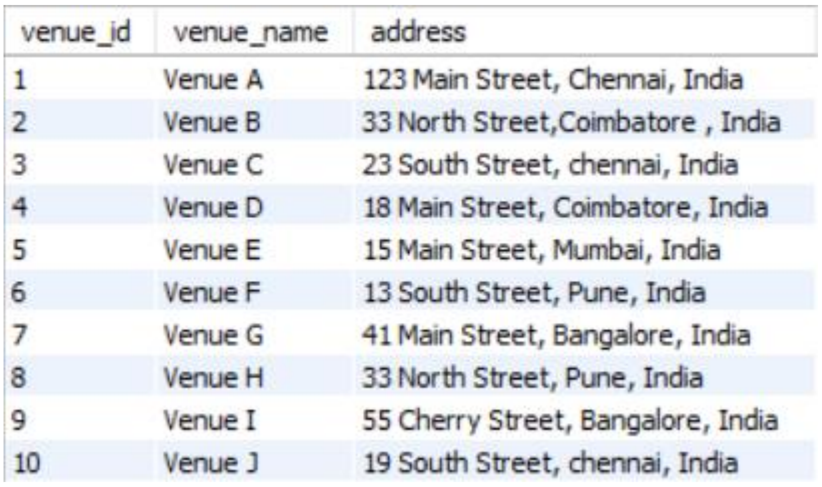
ER DIAGRAM



TASK 2

Inserting values in Venu table

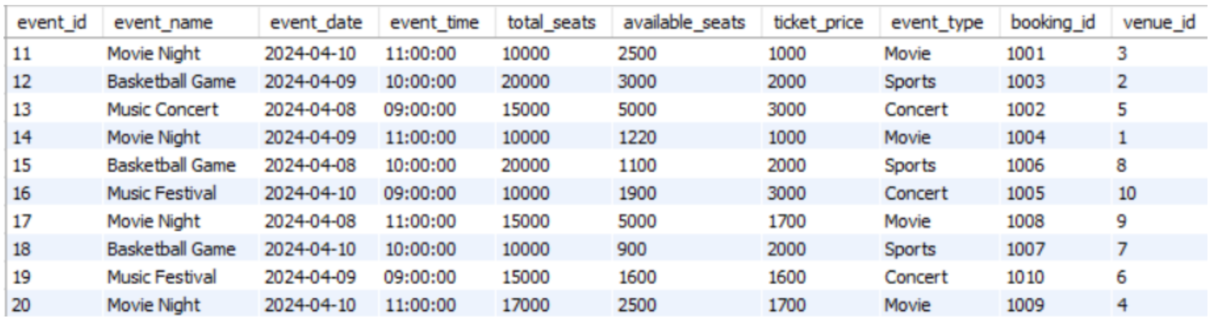
```
insert into Venu(venue_id, venue_name, address) values
(1, 'Venue A', '123 Main Street, Chennai, India'),
(2, 'Venue B', '33 North Street,Coimbatore , India'),
(3, 'Venue C', '23 South Street, chennai, India'),
(4, 'Venue D', '18 Main Street, Coimbatore, India'),
(5, 'Venue E', '15 Main Street, Mumbai, India'),
(6, 'Venue F', '13 South Street, Pune, India'),
(7, 'Venue G', '41 Main Street, Bangalore, India'),
(8, 'Venue H', '33 North Street, Pune, India'),
(9, 'Venue I', '55 Cherry Street, Bangalore, India'),
(10, 'Venue J', '19 South Street, chennai, India');
```



venue_id	venue_name	address
1	Venue A	123 Main Street, Chennai, India
2	Venue B	33 North Street,Coimbatore , India
3	Venue C	23 South Street, chennai, India
4	Venue D	18 Main Street, Coimbatore, India
5	Venue E	15 Main Street, Mumbai, India
6	Venue F	13 South Street, Pune, India
7	Venue G	41 Main Street, Bangalore, India
8	Venue H	33 North Street, Pune, India
9	Venue I	55 Cherry Street, Bangalore, India
10	Venue J	19 South Street, chennai, India

Inserting values in Event table

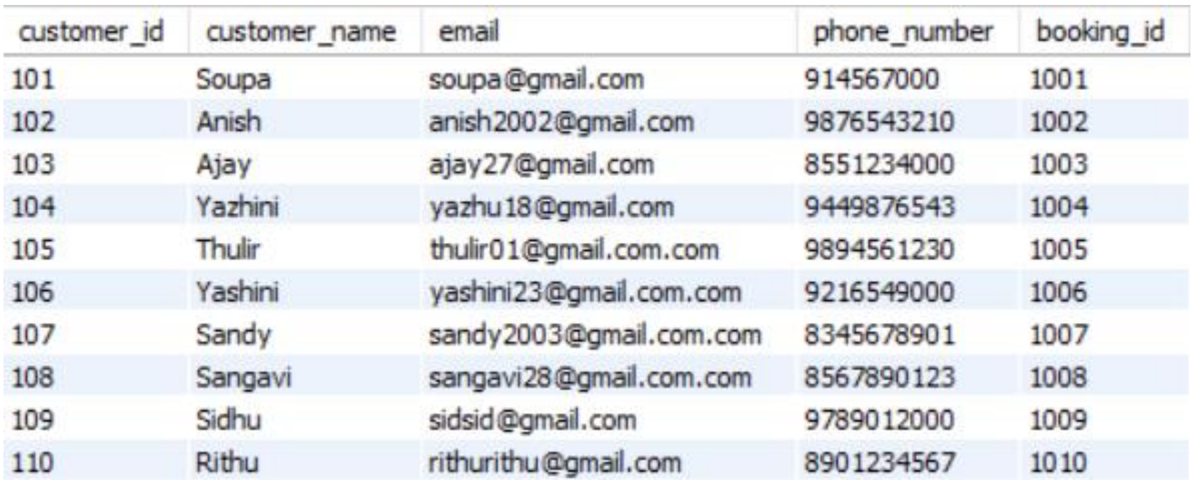
```
insert into event (event_id, event_name, event_date, event_time, total_seats,
available_seats, ticket_price, event_type, venue_id,booking_id) values

(11, 'Movie Night', '2024-04-10', '11:00:00', 10000, 2500, 1000.00, 'Movie', 3,1001),

(12, 'Basketball Game', '2024-04-09', '10:00:00', 20000,3000, 2000.00, 'Sports',
2,1003),

(13, 'Music Concert', '2024-04-08', '09:00:00', 15000,5000, 3000.00, 'Concert',
5,1002),

(14, 'Movie Night', '2024-04-09', '11:00:00', 10000, 1220, 1000.00, 'Movie', 1,1004),

(15, 'Basketball Game', '2024-04-08', '10:00:00', 20000, 1100, 2000.00, 'Sports',
8,1006),

(16, 'Music Festival', '2024-04-10', '09:00:00', 10000, 1900, 3000.00, 'Concert',
10,1005),

(17, 'Movie Night', '2024-04-08', '11:00:00',15000, 5000, 1700.00, 'Movie', 9,1008),

(18, 'Basketball Game', '2024-04-10', '10:00:00', 10000, 900, 2000.00, 'Sports',
7,1007),

(19, 'Music Festival', '2024-04-09', '09:00:00', 15000, 1600, 1600.00, 'Concert',
6,1010),

(20, 'Movie Night', '2024-04-10', '11:00:00', 17000, 2500, 1700.00, 'Movie', 4,1009);
```



event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	booking_id	venue_id
11	Movie Night	2024-04-10	11:00:00	10000	2500	1000	Movie	1001	3
12	Basketball Game	2024-04-09	10:00:00	20000	3000	2000	Sports	1003	2
13	Music Concert	2024-04-08	09:00:00	15000	5000	3000	Concert	1002	5
14	Movie Night	2024-04-09	11:00:00	10000	1220	1000	Movie	1004	1
15	Basketball Game	2024-04-08	10:00:00	20000	1100	2000	Sports	1006	8
16	Music Festival	2024-04-10	09:00:00	10000	1900	3000	Concert	1005	10
17	Movie Night	2024-04-08	11:00:00	15000	5000	1700	Movie	1008	9
18	Basketball Game	2024-04-10	10:00:00	10000	900	2000	Sports	1007	7
19	Music Festival	2024-04-09	09:00:00	15000	1600	1600	Concert	1010	6
20	Movie Night	2024-04-10	11:00:00	17000	2500	1700	Movie	1009	4

Inserting values into customer table

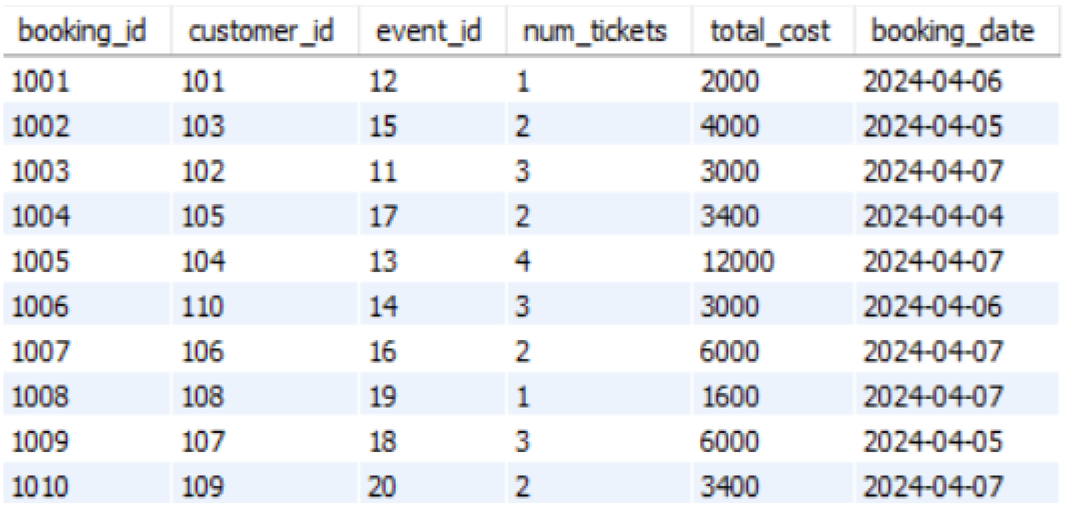
```
insert into Customer (customer_id, customer_name, email,
phone_number,booking_id) values

(101, 'Soupa', 'soupa@gmail.com', '914567000',1001),

(102, 'Anish', 'anish2002@gmail.com', '9876543210',1002),

(103, 'Ajay', 'ajay27@gmail.com', '8551234000',1003),

(104, 'Yazhini', 'yazhu18@gmail.com', '9449876543',1004),

(105, 'Thulir', 'thulir01@gmail.com.com', '9894561230',1005),

(106, 'Yashini', 'yashini23@gmail.com.com', '9216549000',1006),

(107, 'Sandy', 'sandy2003@gmail.com.com', '8345678901',1007),

(108, 'Sangavi', 'sangavi28@gmail.com.com', '8567890123',1008),

(109, 'Sidhu', 'sidsid@gmail.com', '9789012000',1009),

(110, 'Rithu', 'rithurithu@gmail.com', '8901234567',1010);
```



customer_id	customer_name	email	phone_number	booking_id
101	Soupa	soupa@gmail.com	914567000	1001
102	Anish	anish2002@gmail.com	9876543210	1002
103	Ajay	ajay27@gmail.com	8551234000	1003
104	Yazhini	yazhu18@gmail.com	9449876543	1004
105	Thulir	thulir01@gmail.com.com	9894561230	1005
106	Yashini	yashini23@gmail.com.com	9216549000	1006
107	Sandy	sandy2003@gmail.com.com	8345678901	1007
108	Sangavi	sangavi28@gmail.com.com	8567890123	1008
109	Sidhu	sidsid@gmail.com	9789012000	1009
110	Rithu	rithurithu@gmail.com	8901234567	1010

Inserting values in booking table

```
insert into Booking(booking_id, customer_id, event_id, num_tickets, total_cost,
booking_date) values

(1001,101,12,5,10000.00,'2024-04-06'),

(1002,103,15,2,4000.00,'2024-04-05'),

(1003,102,11,3,3000.00,'2024-04-07'),

(1004,105,17,2,3400.00,'2024-04-04'),

(1005,104,13,4,12000.00,'2024-04-07'),

(1006,110,14,3,3000.00,'2024-04-06'),

(1007,106,16,2,6000.00,'2024-04-07'),

(1008,108,19,1,1600.00,'2024-04-07'),

(1009,107,18,3,6000.00,'2024-04-05'),

(1010,109,20,2,3400.00,'2024-04-07');
```



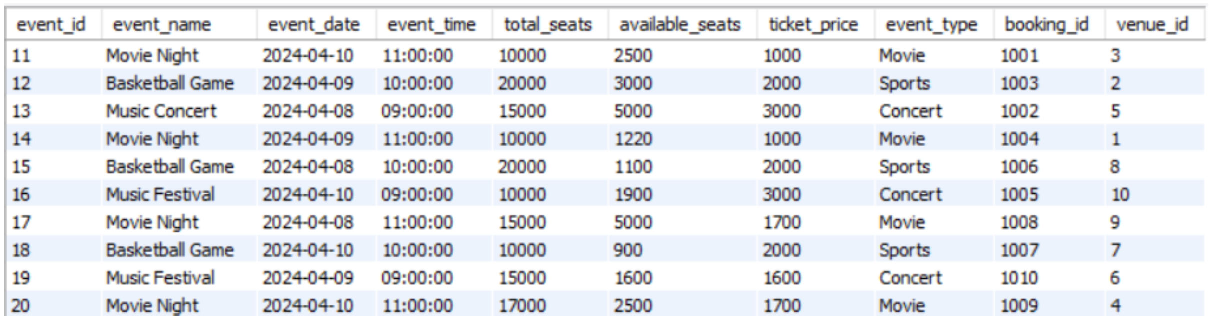
booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
1001	101	12	1	2000	2024-04-06
1002	103	15	2	4000	2024-04-05
1003	102	11	3	3000	2024-04-07
1004	105	17	2	3400	2024-04-04
1005	104	13	4	12000	2024-04-07
1006	110	14	3	3000	2024-04-06
1007	106	16	2	6000	2024-04-07
1008	108	19	1	1600	2024-04-07
1009	107	18	3	6000	2024-04-05
1010	109	20	2	3400	2024-04-07

Making column as foreign key

```
alter table event add constraint fk_event_booking_id FOREIGN KEY (booking_id)
REFERENCES Booking(booking_id);

alter table customer add constraint fk_customer_booking_id FOREIGN KEY
(booking_id) REFERENCES Booking(booking_id);
```

2)Write a SQL query to list all events.

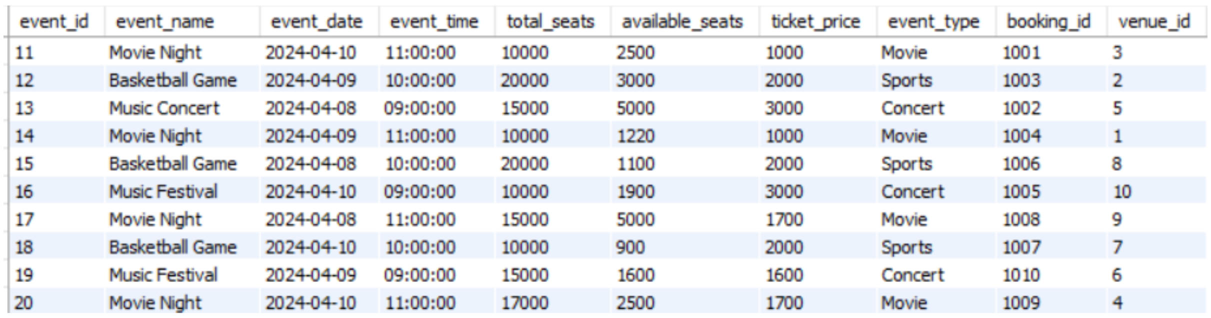
Select * from event;



event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	booking_id	venue_id
11	Movie Night	2024-04-10	11:00:00	10000	2500	1000	Movie	1001	3
12	Basketball Game	2024-04-09	10:00:00	20000	3000	2000	Sports	1003	2
13	Music Concert	2024-04-08	09:00:00	15000	5000	3000	Concert	1002	5
14	Movie Night	2024-04-09	11:00:00	10000	1220	1000	Movie	1004	1
15	Basketball Game	2024-04-08	10:00:00	20000	1100	2000	Sports	1006	8
16	Music Festival	2024-04-10	09:00:00	10000	1900	3000	Concert	1005	10
17	Movie Night	2024-04-08	11:00:00	15000	5000	1700	Movie	1008	9
18	Basketball Game	2024-04-10	10:00:00	10000	900	2000	Sports	1007	7
19	Music Festival	2024-04-09	09:00:00	15000	1600	1600	Concert	1010	6
20	Movie Night	2024-04-10	11:00:00	17000	2500	1700	Movie	1009	4

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

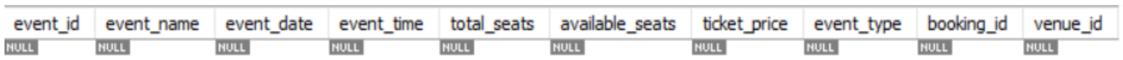
Select * from event where available_seats>0;



event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	booking_id	venue_id
11	Movie Night	2024-04-10	11:00:00	10000	2500	1000	Movie	1001	3
12	Basketball Game	2024-04-09	10:00:00	20000	3000	2000	Sports	1003	2
13	Music Concert	2024-04-08	09:00:00	15000	5000	3000	Concert	1002	5
14	Movie Night	2024-04-09	11:00:00	10000	1220	1000	Movie	1004	1
15	Basketball Game	2024-04-08	10:00:00	20000	1100	2000	Sports	1006	8
16	Music Festival	2024-04-10	09:00:00	10000	1900	3000	Concert	1005	10
17	Movie Night	2024-04-08	11:00:00	15000	5000	1700	Movie	1008	9
18	Basketball Game	2024-04-10	10:00:00	10000	900	2000	Sports	1007	7
19	Music Festival	2024-04-09	09:00:00	15000	1600	1600	Concert	1010	6
20	Movie Night	2024-04-10	11:00:00	17000	2500	1700	Movie	1009	4

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

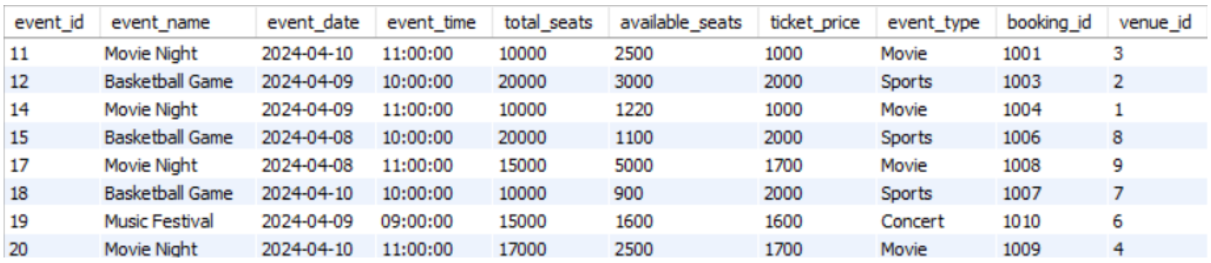
Select * from event where event_name like '%cup%';



event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	booking_id	venue_id
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

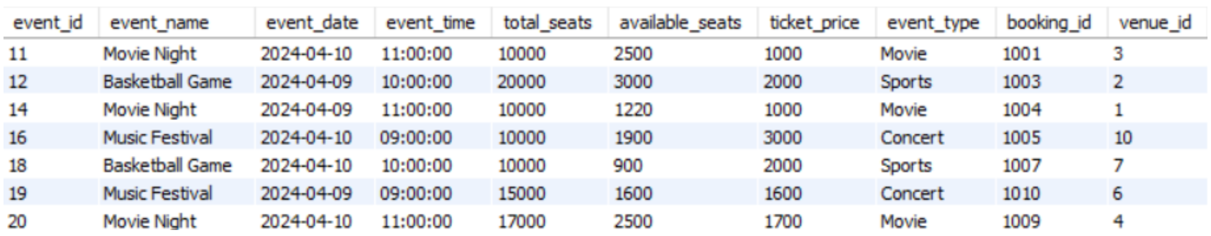
Select * from event where ticket_price between 1000 and 2500;



event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	booking_id	venue_id
11	Movie Night	2024-04-10	11:00:00	10000	2500	1000	Movie	1001	3
12	Basketball Game	2024-04-09	10:00:00	20000	3000	2000	Sports	1003	2
14	Movie Night	2024-04-09	11:00:00	10000	1220	1000	Movie	1004	1
15	Basketball Game	2024-04-08	10:00:00	20000	1100	2000	Sports	1006	8
17	Movie Night	2024-04-08	11:00:00	15000	5000	1700	Movie	1008	9
18	Basketball Game	2024-04-10	10:00:00	10000	900	2000	Sports	1007	7
19	Music Festival	2024-04-09	09:00:00	15000	1600	1600	Concert	1010	6
20	Movie Night	2024-04-10	11:00:00	17000	2500	1700	Movie	1009	4

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

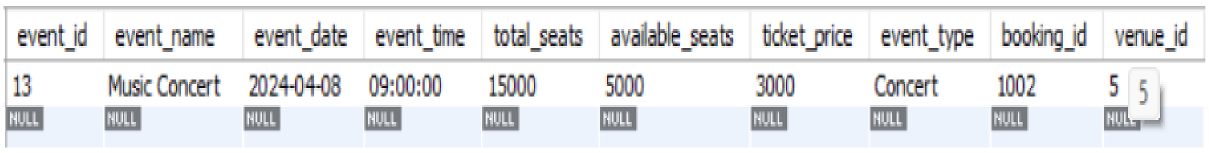
Select * from event where event_date between '2024-04-09' and '2024-04-10';



event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	booking_id	venue_id
11	Movie Night	2024-04-10	11:00:00	10000	2500	1000	Movie	1001	3
12	Basketball Game	2024-04-09	10:00:00	20000	3000	2000	Sports	1003	2
14	Movie Night	2024-04-09	11:00:00	10000	1220	1000	Movie	1004	1
16	Music Festival	2024-04-10	09:00:00	10000	1900	3000	Concert	1005	10
18	Basketball Game	2024-04-10	10:00:00	10000	900	2000	Sports	1007	7
19	Music Festival	2024-04-09	09:00:00	15000	1600	1600	Concert	1010	6
20	Movie Night	2024-04-10	11:00:00	17000	2500	1700	Movie	1009	4

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

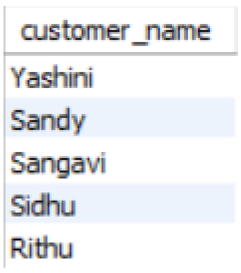
select* from event where available_seats > 0 and event_name like '%Concert%';



event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	booking_id	venue_id
13	Music Concert	2024-04-08	09:00:00	15000	5000	3000	Concert	1002	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

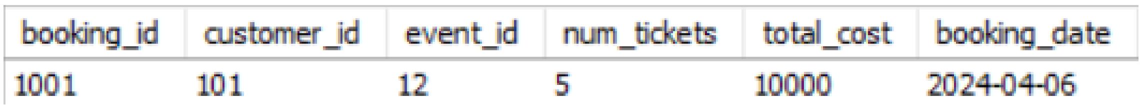
select customer_name from customer order by customer_id limit 5 offset 5;



customer_name
Yashini
Sandy
Sangavi
Sidhu
Rithu

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

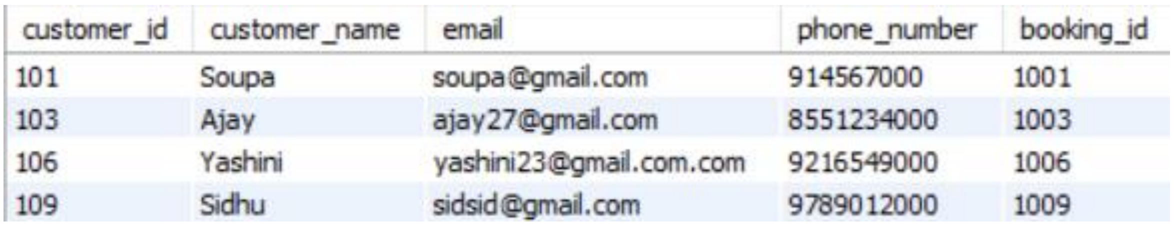
Select * from booking where num_tickets > 4;



booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
1001	101	12	5	10000	2024-04-06

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

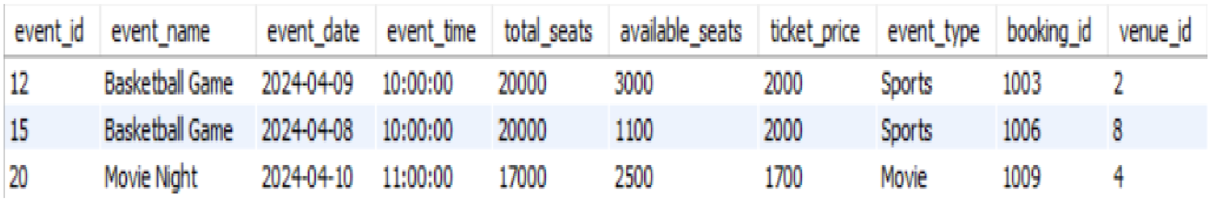
Select * from customer where phone_number like '%000';



customer_id	customer_name	email	phone_number	booking_id
101	Soupa	soupa@gmail.com	914567000	1001
103	Ajay	ajay27@gmail.com	8551234000	1003
106	Yashini	yashini23@gmail.com.com	9216549000	1006
109	Sidhu	sidsid@gmail.com	9789012000	1009

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

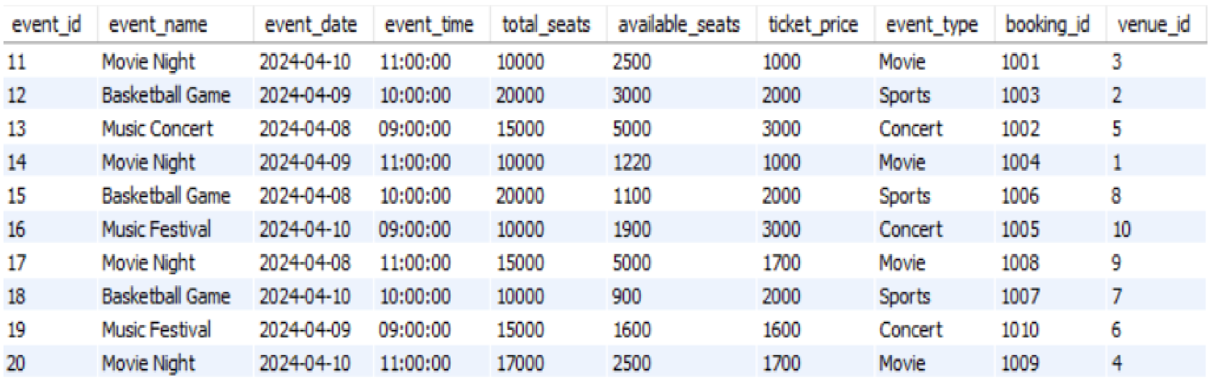
Select * from event where total_seats > 15000;



event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	booking_id	venue_id
12	Basketball Game	2024-04-09	10:00:00	20000	3000	2000	Sports	1003	2
15	Basketball Game	2024-04-08	10:00:00	20000	1100	2000	Sports	1006	8
20	Movie Night	2024-04-10	11:00:00	17000	2500	1700	Movie	1009	4

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%';

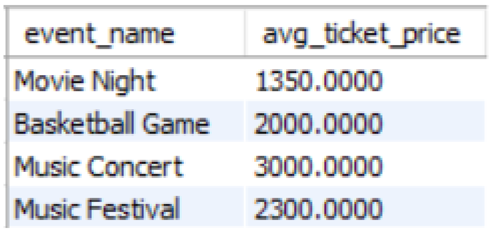


event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	booking_id	venue_id
11	Movie Night	2024-04-10	11:00:00	10000	2500	1000	Movie	1001	3
12	Basketball Game	2024-04-09	10:00:00	20000	3000	2000	Sports	1003	2
13	Music Concert	2024-04-08	09:00:00	15000	5000	3000	Concert	1002	5
14	Movie Night	2024-04-09	11:00:00	10000	1220	1000	Movie	1004	1
15	Basketball Game	2024-04-08	10:00:00	20000	1100	2000	Sports	1006	8
16	Music Festival	2024-04-10	09:00:00	10000	1900	3000	Concert	1005	10
17	Movie Night	2024-04-08	11:00:00	15000	5000	1700	Movie	1008	9
18	Basketball Game	2024-04-10	10:00:00	10000	900	2000	Sports	1007	7
19	Music Festival	2024-04-09	09:00:00	15000	1600	1600	Concert	1010	6
20	Movie Night	2024-04-10	11:00:00	17000	2500	1700	Movie	1009	4

TASK 3

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

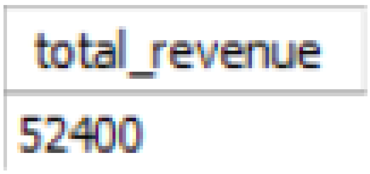
select event_name, avg(ticket_price) as avg_ticket_price from event group by event_name;



event_name	avg_ticket_price
Movie Night	1350.0000
Basketball Game	2000.0000
Music Concert	3000.0000
Music Festival	2300.0000

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

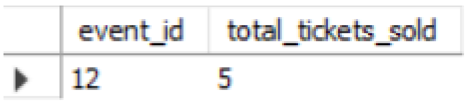
select sum(total_cost) as total_revenue from Booking;



total_revenue
52400

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

select event_id, sum(num_tickets) as total_tickets_sold from booking group by event_id order by total_tickets_sold desc limit 1;

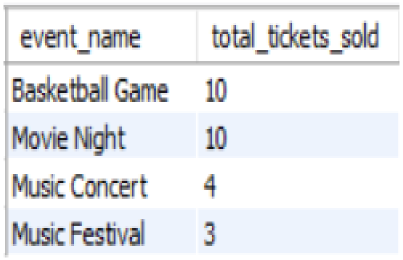


	event_id	total_tickets_sold
▶	12	5

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

select event_name, sum(num_tickets) as total_tickets_sold from Booking b

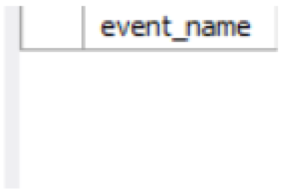
join event e on b.event_id = e.event_id group by event_name;



event_name	total_tickets_sold
Basketball Game	10
Movie Night	10
Music Concert	4
Music Festival	3

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

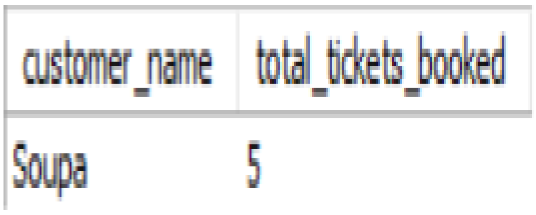
select event_name from event where event_id not in (select event_id from booking where num_tickets>0);



event_name

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

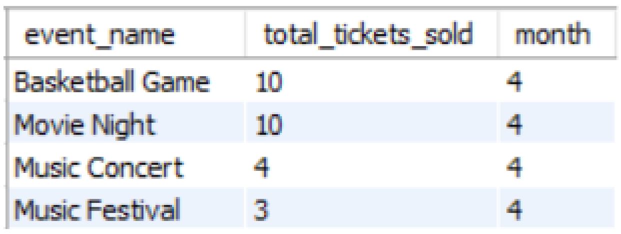
select customer_name, sum(num_tickets) as total_tickets_booked from Customer c join Booking b on c.customer_id = b.customer_id group by customer_name order by total_tickets_booked desc limit 1;



customer_name	total_tickets_booked
Soupa	5

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

select event_name, SUM(num_tickets) as total_tickets_sold, month(booking_date) as month from Booking b join event e on b.event_id = e.event_id group by event_name, month(booking_date);

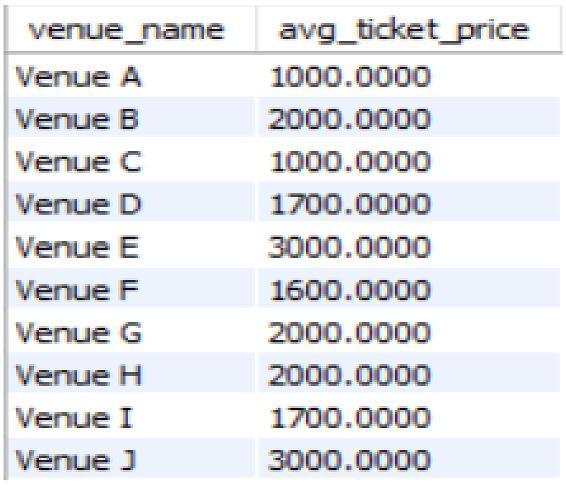


event_name	total_tickets_sold	month
Basketball Game	10	4
Movie Night	10	4
Music Concert	4	4
Music Festival	3	4

8. 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

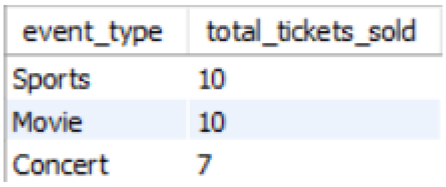
from event e join Venu v on e.venue_id = v.venue_id group by v.venue_name;



venue_name	avg_ticket_price
Venue A	1000.0000
Venue B	2000.0000
Venue C	1000.0000
Venue D	1700.0000
Venue E	3000.0000
Venue F	1600.0000
Venue G	2000.0000
Venue H	2000.0000
Venue I	1700.0000
Venue J	3000.0000

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

select event_type, SUM(num_tickets) as total_tickets_sold from Booking b join event e on b.event_id = e.event_id group by event_type;

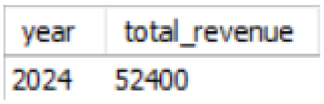


event_type	total_tickets_sold
Sports	10
Movie	10
Concert	7

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

select year(booking_date) as year, sum(total_cost) as total_revenue

from Booking group by year(booking_date);



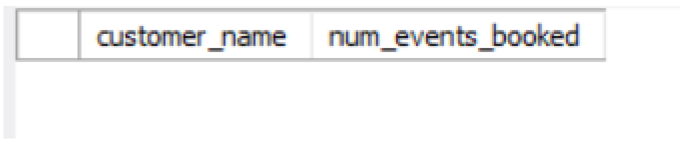
year	total_revenue
2024	52400

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

select customer_name, COUNT(distinct event_id) as num_events_booked

from Customer c join Booking b on c.customer_id = b.customer_id

group by customer_name having num_events_booked > 1;



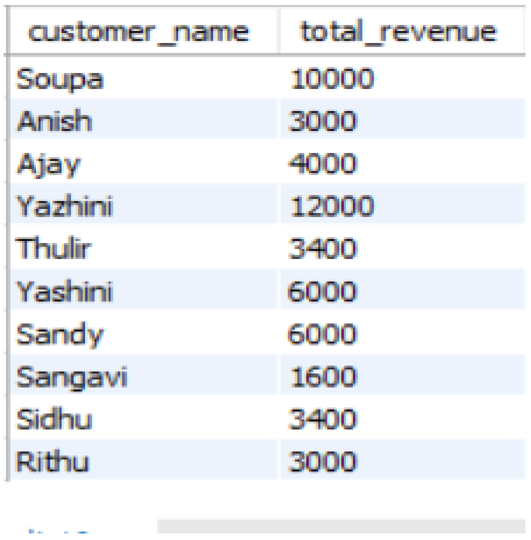
	customer_name	num_events_booked

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

select customer_name,sum(total_cost) as total_revenue

from Customer c join Booking b on c.customer_id = b.customer_id group by

customer_name;

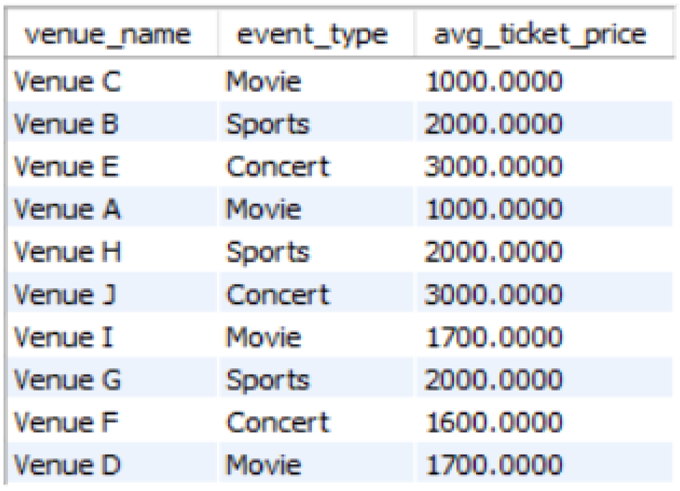


customer_name	total_revenue
Soupa	10000
Anish	3000
Ajay	4000
Yazhini	12000
Thulir	3400
Yashini	6000
Sandy	6000
Sangavi	1600
Sidhu	3400
Rithu	3000

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

select v.venue_name, event_type, avg(ticket_price) as avg_ticket_price

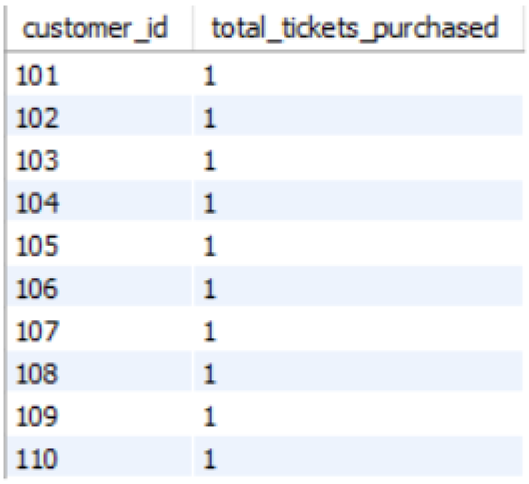
from event e join Venu v on e.venue_id = v.venue_id group by v.venue_name, event_type;



venue_name	event_type	avg_ticket_price
Venue C	Movie	1000.0000
Venue B	Sports	2000.0000
Venue E	Concert	3000.0000
Venue A	Movie	1000.0000
Venue H	Sports	2000.0000
Venue J	Concert	3000.0000
Venue I	Movie	1700.0000
Venue G	Sports	2000.0000
Venue F	Concert	1600.0000
Venue D	Movie	1700.0000

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

select customer_id, count(*) as total_tickets_purchased from booking where booking_date >= date_sub(current_date(), interval 30 day) group by customer_id;

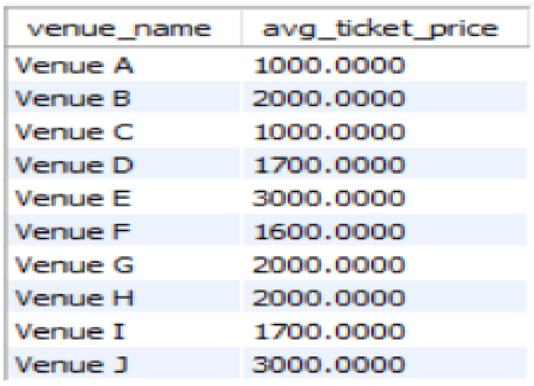


customer_id	total_tickets_purchased
101	1
102	1
103	1
104	1
105	1
106	1
107	1
108	1
109	1
110	1

TASK 4

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

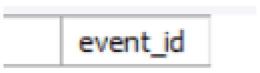
select v.venue_name,(select avg(ticket_price) from event where venue_id = v.venue_id) as avg_ticket_price from Venu v;



venue_name	avg_ticket_price
Venue A	1000.0000
Venue B	2000.0000
Venue C	1000.0000
Venue D	1700.0000
Venue E	3000.0000
Venue F	1600.0000
Venue G	2000.0000
Venue H	2000.0000
Venue I	1700.0000
Venue J	3000.0000

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

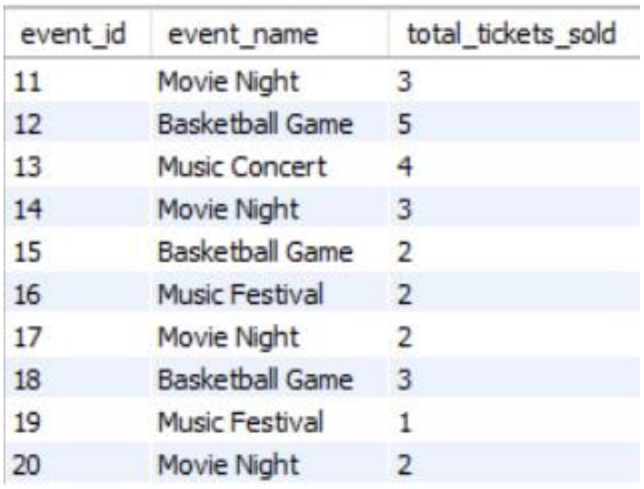
select event_id from booking group by event_id having SUM(num_tickets) > (select 0.5 * total_seats from event where event.event_id = booking.event_id);



	event_id

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

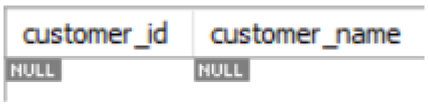
select event_id, event_name, (select sum(num_tickets) from Booking where Booking.event_id = Event.event_id) as total_tickets_sold from Event;



event_id	event_name	total_tickets_sold
11	Movie Night	3
12	Basketball Game	5
13	Music Concert	4
14	Movie Night	3
15	Basketball Game	2
16	Music Festival	2
17	Movie Night	2
18	Basketball Game	3
19	Music Festival	1
20	Movie Night	2

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

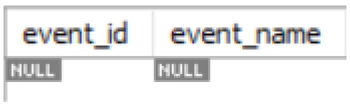
select customer_id, customer_name from Customer c where not exists (select * from Booking where Booking.customer_id = c.customer_id);



customer_id	customer_name
NULL	NULL

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

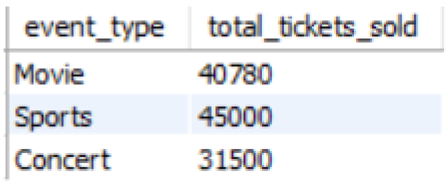
select event_id, event_name from event where event_id not in (select distinct event_id from Booking);



event_id	event_name
NULL	NULL

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

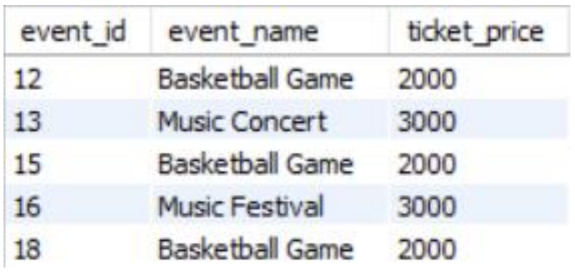
select e1.event_type, e1.total_seats - e1.available_seats as total_tickets_sold from (select event_type, sum(total_seats) as total_seats, sum(available_seats) as available_seats from event group by event_type) as e1;



event_type	total_tickets_sold
Movie	40780
Sports	45000
Concert	31500

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

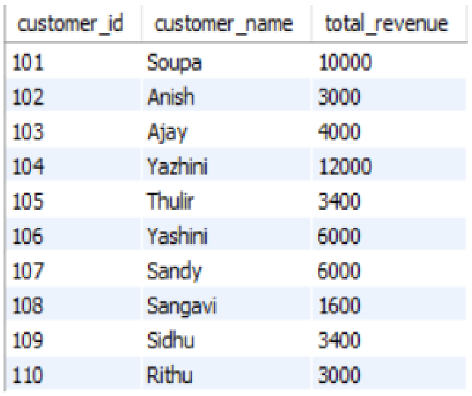
select event_id, event_name, ticket_price from event where twhicket_price > (select avg(ticket_price) from event);



event_id	event_name	ticket_price
12	Basketball Game	2000
13	Music Concert	3000
15	Basketball Game	2000
16	Music Festival	3000
18	Basketball Game	2000

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

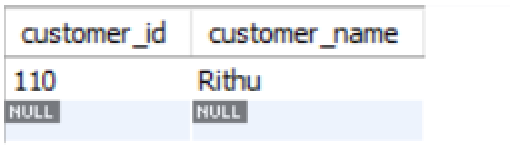
select customer_id, customer_name,(select SUM(total_cost) from Booking where Booking.customer_id = Customer.customer_id) as total_revenue from Customer;



customer_id	customer_name	total_revenue
101	Soupa	10000
102	Anish	3000
103	Ajay	4000
104	Yazhini	12000
105	Thulir	3400
106	Yashini	6000
107	Sandy	6000
108	Sangavi	1600
109	Sidhu	3400
110	Rithu	3000

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

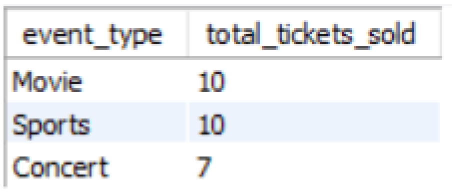
```
select customer_id, customer_name from Customer where customer_id in (select
distinct customer_id from Booking where event_id in (select event_id from Event
where venue_id = (select venue_id from Venu where venue_name = 'Venue A')));
```



customer_id	customer_name
110	Rithu
NULL	NULL

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

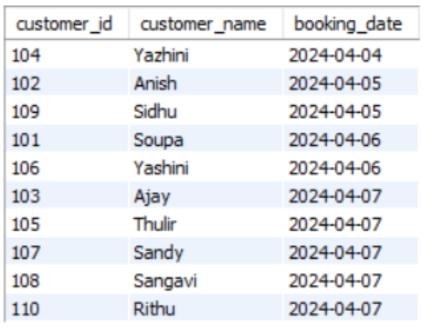
```
select event_type, sum(total_tickets_sold) as total_tickets_soldfrom (select
event_type, event_id, (Select (num_tickets) from Booking where Booking.event_id =
Event.event_id) as total_tickets_sold from event) as EventsByCategory group by
event_type;
```



event_type	total_tickets_sold
Movie	10
Sports	10
Concert	7

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

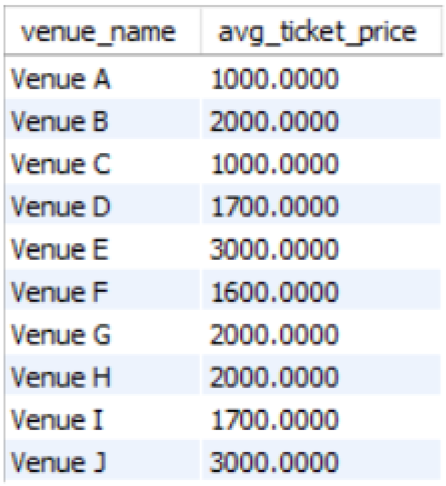
```
select c.customer_id,  c.customer_name,  (select b.booking_date from booking b
where c.booking_id = b.booking_id) as booking_date from customer c order
by booking_date;
```



customer_id	customer_name	booking_date
104	Yazhini	2024-04-04
102	Anish	2024-04-05
109	Sidhu	2024-04-05
101	Soupa	2024-04-06
106	Yashini	2024-04-06
103	Ajay	2024-04-07
105	Thulir	2024-04-07
107	Sandy	2024-04-07
108	Sangavi	2024-04-07
110	Rithu	2024-04-07

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

```
select v.venue_name,(select avg(ticket_price) from event where Event.venue_id =
v.venue_id) AS avg_ticket_price from Venu v;
```



venue_name	avg_ticket_price
Venue A	1000.0000
Venue B	2000.0000
Venue C	1000.0000
Venue D	1700.0000
Venue E	3000.0000
Venue F	1600.0000
Venue G	2000.0000
Venue H	2000.0000
Venue I	1700.0000
Venue J	3000.0000