Ticket Booking System

Tasks 1: Database Design:

1. Create the database named "TicketBookingSystem"

```
create database TicketBookingSystem;
use TicketBookingSystem;
```

2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and
relationships.
• Venu
• Event
• Customers
• Booking

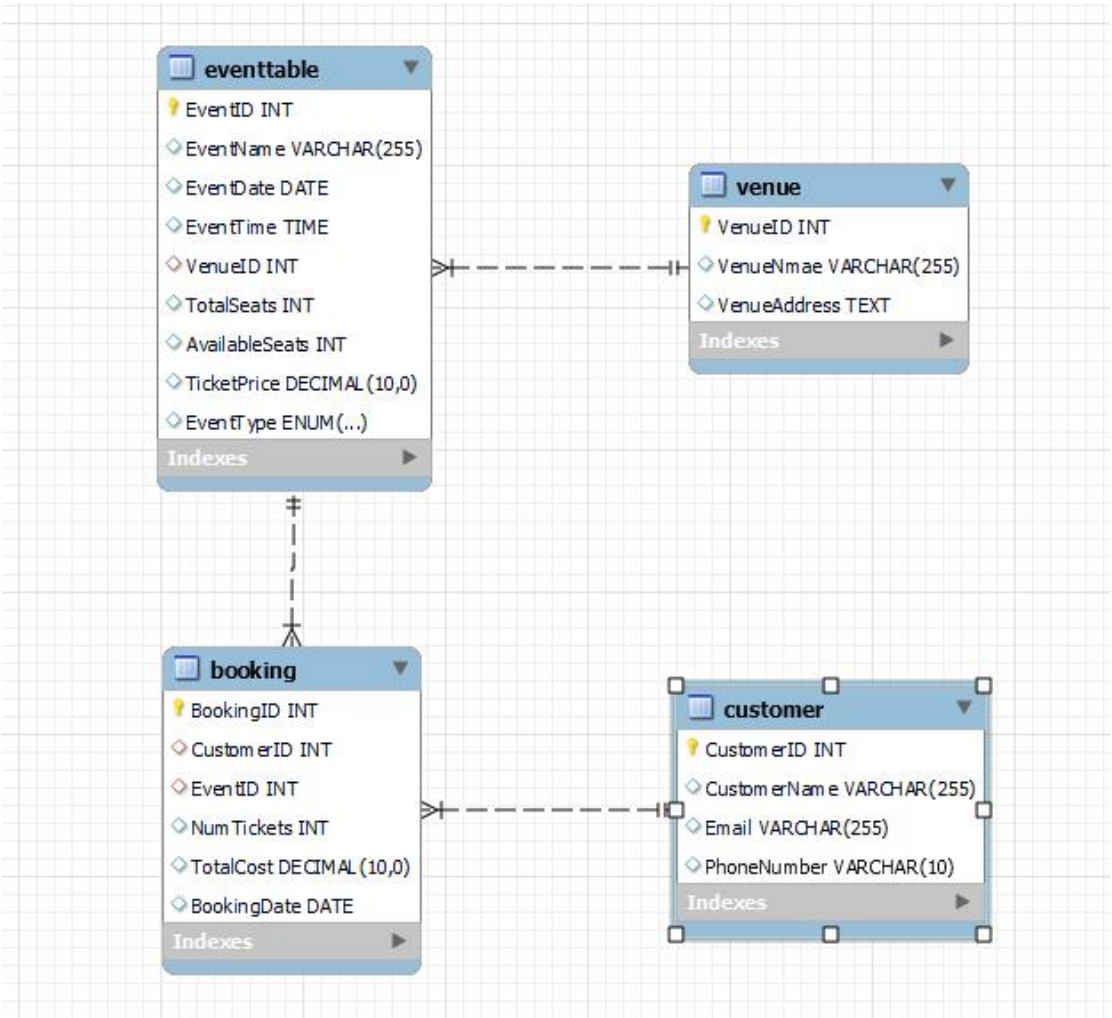
```
create table Venue(
    VenueID int primary key,
    VenueNmae varchar(255),
    VenueAddress text);

create table EventTable(
    EventID int primary key,
    EventName varchar(255),
    EventDate date,
    EventTime time,
    VenueID int,
    TotalSeats int,
    AvailableSeats int,
    TicketPrice decimal,
    EventType enum("movie", "sports","concert"),
    BookingID int);
```

```sql
create table Customer(
    CustomerID int primary key,
    CustomerName varchar(255),
    Email varchar(255),
    PhoneNumber varchar(10),
    BookingID int);

create table Booking(
    BookingID int primary key,
    CustomerID int,
    EventID int,
    NumTickets int,
    TotalCost decimal,
    BookingDate date
    );
```

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



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

```
alter table EventTable add foreign key(VenueID) references Venue(VenueID);
alter table EventTable add foreign key (BookingID) references Booking(BookingID);
alter table Customer add foreign key (BookingID) references Booking(BookingID);
alter table Booking add foreign key (CustomerID) references Customer(CustomerID);
alter table Booking add foreign key (EventID) references EventTable(EventID);
```

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

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

```
insert into venue values
    (1, "ASD stadium", "kolkatta"),
    (2, "xyz stadium", "chennai"),
    (3, "nehru stadium", "madurai"),
    (4, "dadha stadium", "kochi"),
    (5, "leodas stadium", "bangalore"),
    (6, "bone park", "hydrabad"),
    (7, "dodo beach", "delhi"),
    (8, "kj park", "jarkand"),
    (9, "Ascent", "dheradhum"),
    (10, "pearl park", "jammu");

insert into eventtable values
    (1, "ev1", "2024-1-13", "14:00:00", 1, 1000, 500, 1000, "movie"),
    (2, "ev2", "2024-1-14", "14:00:00", 2, 3000, 1500, 2000, "sports"),
    (3, "ev3", "2024-1-15", "14:00:00", 3, 5000, 3000, 3000, "concert"),
    (4, "ev4", "2024-1-16", "14:00:00", 4, 7000, 55000, 4000, "movie"),
    (5, "ev5", "2024-1-17", "14:00:00", 5, 9000, 4000, 5000, "sports"),
    (6, "ev6", "2024-1-18", "14:00:00", 6, 11000, 1000, 6000, "concert"),
    (7, "ev7", "2024-1-19", "14:00:00", 2, 13000, 11000, 7000, "movie"),
    (8, "ev8", "2024-1-20", "14:00:00", 2, 15000, 7000, 8000, "sports"),
    (9, "ev9", "2024-1-21", "14:00:00", 3, 17000, 7000, 9000, "concert"),
    (10, "ev10", "2024-1-22", "14:00:00", 9, 19000, 2000, 10000, "movie");
```

```
insert into Customer values
    (1, "James", "JamesSmith@gmail.com", "1234567890"),
    (2, "Christopher", "ChristopherAnderson@gmail.com", "1234567891"),
    (3, "Ronald", "RonaldClark@gmail.com", "1234567892"),
    (4, "Mary", "MaryWright@gmail.com", "1234567893"),
    (5, "Lisa", "LisaMitchell@gmail.com", "1234567894"),
    (6, "Michelle", "MichelleJohnson@gmail.com", "1234567895"),
    (7, "John", "JohnThomas@gmail.com", "1234567896"),
    (8, "Daniel", "DanielRodriguez@gmail.com","1234567897"),
    (9, "Anthony", "AnthonyLopez@gmail.com", "1234567898"),
    (10, "Patricia", "PatriciaPerez@gmail.com", "1234567000");
```

```
insert into booking values
    (1, 1, 1, 3, 3000, "2024-1-8"),
    (2, 2, 2, 4, 8000, "2024-1-8"),
    (3, 3, 3, 5, 15000, "2024-1-8"),
    (4, 4, 4, 1, 4000, "2024-1-8"),
    (5, 5, 5, 1, 5000, "2024-1-8"),
    (6, 6, 6, 1, 6000, "2024-1-8"),
    (7, 7, 8, 1, 8000, "2024-1-8"),
    (8, 9, 9, 5, 45000, "2024-1-8"),
    (9, 10, 10, 9, 90000, "2024-1-8"),
    (10, 1, 2, 1, 2000, "2024-1-8"),
    (11, 1, 1, 1, 1000, "2024-1-8");
```

VenueID	VenueNmae	VenueAddress
1	ASD stadium	kolkatta
2	xyz stadium	chennai
3	nehru stadium	madurai
4	dadha stadium	kochi
5	leodas stadium	bangalore
6	bone park	hydrabad
7	dodo beach	delhi
8	kj park	jarkand
9	Ascent	dheradhum
10	pearl park	jammu

EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
1	ev1	2024-01-13	14:00:00	1	1000	500	1000	Movie
2	ev2	2024-01-14	14:00:00	2	3000	1500	2000	Sports
3	ev3	2024-01-15	14:00:00	3	5000	3000	3000	Concert
4	xamb	2024-01-16	14:00:00	4	7000	55000	4000	Movie
5	ev5	2024-01-17	14:00:00	5	9000	4000	5000	Sports
6	ev6	2024-01-18	14:00:00	6	11000	1000	6000	Concert
7	ev7	2024-01-19	14:00:00	2	13000	11000	7000	Movie
8	ev8	2024-01-20	14:00:00	2	15000	7000	8000	Sports
9	ev9	2024-01-21	14:00:00	3	17000	7000	9000	Concert
10	ev10	2024-01-22	14:00:00	9	19000	2000	10000	Movie

CustomerID	CustomerName	Email	PhoneNumber
1	James	JamesSmith@gmail.com	1234567890
2	Christopher	ChristopherAnderson@gmail.com	1234567891
3	Ronald	RonaldClark@gmail.com	1234567892
4	Mary	MaryWright@gmail.com	1234567893
5	Lisa	LisaMitchell@gmail.com	1234567894
6	Michelle	MichelleJohnson@gmail.com	1234567895
7	John	JohnThomas@gmail.com	1234567896
8	Daniel	DanielRodriguez@gmail.com	1234567897
9	Anthony	AnthonyLopez@gmail.com	1234567898
10	Patricia	PatriciaPerez@gmail.com	1234567000

BookingID	CustomerID	EventID	NumTickets	TotalCost	BookingDate
1	1	1	3	3000	2024-01-08
2	2	2	4	8000	2024-01-08
3	3	3	5	15000	2024-01-08
4	4	4	1	4000	2024-01-08
5	5	5	1	5000	2024-01-08
6	6	6	1	6000	2024-01-08
7	7	8	1	8000	2024-01-08
8	9	9	5	45000	2024-01-08
9	10	10	9	90000	2024-01-08
10	1	2	1	2000	2024-01-08
11	1	1	1	1000	2024-01-08

2. Write a SQL query to list all Events.

```
select * from eventtable;
```

EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
1	ev1	2024-01-13	14:00:00	1	1000	500	1000	Movie
2	ev2	2024-01-14	14:00:00	2	3000	1500	2000	Sports
3	ev3	2024-01-15	14:00:00	3	5000	3000	3000	Concert
4	xamb	2024-01-16	14:00:00	4	7000	55000	4000	Movie
5	ev5	2024-01-17	14:00:00	5	9000	4000	5000	Sports
6	ev6	2024-01-18	14:00:00	6	11000	1000	6000	Concert
7	ev7	2024-01-19	14:00:00	2	13000	11000	7000	Movie
8	ev8	2024-01-20	14:00:00	2	15000	7000	8000	Sports
9	ev9	2024-01-21	14:00:00	3	17000	7000	9000	Concert
10	ev10	2024-01-22	14:00:00	9	19000	2000	10000	Movie

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

```
select * from eventtable where availableseats >0;
```

EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
1	ev1	2024-01-13	14:00:00	1	1000	500	1000	Movie
2	ev2	2024-01-14	14:00:00	2	3000	1500	2000	Sports
3	ev3	2024-01-15	14:00:00	3	5000	3000	3000	Concert
4	xamb	2024-01-16	14:00:00	4	7000	55000	4000	Movie
5	ev5	2024-01-17	14:00:00	5	9000	4000	5000	Sports
6	ev6	2024-01-18	14:00:00	6	11000	1000	6000	Concert
7	ev7	2024-01-19	14:00:00	2	13000	11000	7000	Movie
8	ev8	2024-01-20	14:00:00	2	15000	7000	8000	Sports
9	ev9	2024-01-21	14:00:00	3	17000	7000	9000	Concert
10	ev10	2024-01-22	14:00:00	9	19000	2000	10000	Movie

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

```
insert into eventtable
values(11, "world cup", "2024-1-12", "14:00:00", 1, 1000, 500, 1000, "movie");
select * from eventtable where eventname like "%cup%";
```

EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
11	world cup	2024-01-12	14:00:00	1	1000	500	1000	Movie

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

```
select * from eventtable where ticketprice between 1000 and 1500;
```

EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
1	ev1	2024-01-13	14:00:00	1	1000	500	1000	Movie
11	world cup	2024-01-12	14:00:00	1	1000	500	1000	Movie

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

```
select * from eventtable where eventdate between "2024-1-13" and "2024-1-16";
```

EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
1	ev1	2024-01-13	14:00:00	1	1000	500	1000	Movie
2	ev2	2024-01-14	14:00:00	2	3000	1500	2000	Sports
3	ev3	2024-01-15	14:00:00	3	5000	3000	3000	Concert
4	xamb	2024-01-16	14:00:00	4	7000	55000	4000	Movie

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

```
select * from eventtable where availableseats >0 and eventtype = "concert";
```

	EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
▶	3	ev3	2024-01-15	14:00:00	3	5000	3000	3000	Concert
	6	ev6	2024-01-18	14:00:00	6	11000	1000	6000	Concert
	9	ev9	2024-01-21	14:00:00	3	17000	7000	9000	Concert

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

```
WITH NumberedRows AS (
  SELECT customerid,
  ((ROW_NUMBER() OVER (ORDER BY customerid))-1) div 5 + 1 AS batch
  FROM customer where customerid>5
)
SELECT customer.*, NumberedRows.batch
FROM customer JOIN NumberedRows
ON customer.customerid = NumberedRows.customerid;
```

	CustomerID	CustomerName	Email	PhoneNumber	batch
▶	6	Michelle	MichelleJohnson@gmail.com	1234567895	1
	7	John	JohnThomas@gmail.com	1234567896	1
	8	Daniel	DanielRodriguez@gmail.com	1234567897	1
	9	Anthony	AnthonyLopez@gmail.com	1234567898	1
	10	Patricia	PatriciaPerez@gmail.com	1234567000	1
	11	dofen	dofen@gmail.com	1231231234	2
	12	dummy	dummy@gmail.com	1231212122	2

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

```
select * from booking where numtickets>4;
```

	BookingID	CustomerID	EventID	NumTickets	TotalCost	BookingDate
▶	3	3	3	5	15000	2024-01-08
	8	9	9	5	45000	2024-01-08
	9	10	10	9	90000	2024-01-08

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

```
select * from customer where phonenumber like "%000";
```

	CustomerID	CustomerName	Email	PhoneNumber
▶	10	Patricia	PatriciaPerez@gmail.com	1234567000

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

```
select * from eventtable where TotalSeats >15000;
```

EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
9	ev9	2024-01-21	14:00:00	3	17000	7000	9000	Concert
10	ev10	2024-01-22	14:00:00	9	19000	2000	10000	Movie

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

```
select * from eventtable
 where eventname not like "x%"
 and eventname not like "y%"
 and eventname not like "z%";
```

EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
1	ev1	2024-01-13	14:00:00	1	1000	500	1000	Movie
2	ev2	2024-01-14	14:00:00	2	3000	1500	2000	Sports
3	ev3	2024-01-15	14:00:00	3	5000	3000	3000	Concert
5	ev5	2024-01-17	14:00:00	5	9000	4000	5000	Sports
6	ev6	2024-01-18	14:00:00	6	11000	1000	6000	Concert
7	ev7	2024-01-19	14:00:00	2	13000	11000	7000	Movie
8	ev8	2024-01-20	14:00:00	2	15000	7000	8000	Sports
9	ev9	2024-01-21	14:00:00	3	17000	7000	9000	Concert
10	ev10	2024-01-22	14:00:00	9	19000	2000	10000	Movie
11	world cup	2024-01-12	14:00:00	1	1000	500	1000	Movie

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

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

```
select eventtype, avg(ticketprice) from eventtable group by eventtype;
```

eventtype	avg(ticketprice)
Movie	4600.0000
Sports	5000.0000
Concert	6000.0000

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

```
select sum(totalcost) as total_rev_gen from booking;
```

total_rev_gen
187000

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

```
select eventid, count(eventid) as no_of_tic
from booking group by eventid
having count(eventid) = (select max(eventcount)
from (select count(eventid) as eventcount
from booking group by eventid) as t1);
```

	eventid	no_of_tic
▶	1	2
	2	2

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

```
select eventid, count(eventid) as no_of_tic
from booking group by eventid;
```

	eventid	no_of_tic
▶	1	2
	2	2
	3	1
	4	1
	5	1
	6	1
	8	1
	9	1
	10	1

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

```
select * from eventtable
where eventid not in(select eventid from booking);
```

	EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
▶	7	ev7	2024-01-19	14:00:00	2	13000	11000	7000	Movie
	11	world cup	2024-01-12	14:00:00	1	1000	500	1000	Movie

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

```
select customerid, sum(numtickets)
from booking group by customerid
having sum(numtickets) = (select max(con)
from (select sum(numtickets) as con
from booking group by customerid) as t1);
```

	customerid	sum(numtickets)
▶	10	9

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

```
select eventtable.eventname,month(booking.bookingdate),
sum(booking.numtickets)
from eventtable join booking
on eventtable.eventid = booking.eventid
group by eventtable.eventname,month(booking.bookingdate);
```

	eventname	month(booking.bookingdate)	sum(booking.numtickets)
▶	ev1	1	4
	ev2	1	5
	ev3	1	5
	xamb	1	1
	ev5	1	1
	ev6	1	1
	ev8	1	1
	ev9	1	5
	ev10	1	9

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

```
select venue.VenueNmae, avg(ticketprice)
from venue join eventtable
on venue.venueid = eventtable.venueid
group by venue.VenueNmae;
```

	VenueNmae	avg(ticketprice)
▶	ASD stadium	1000.0000
	xyz stadium	5666.6667
	nehru stadium	6000.0000
	dadha stadium	4000.0000
	leodas stadium	5000.0000
	bone park	6000.0000
	Ascent	10000.0000

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

```
select eventtable.eventtype, count(numtickets)
from eventtable join booking
on eventtable.eventid = booking.eventid
group by eventtable.eventtype;
```

	eventtype	count(numtickets)
▶	Movie	4
	Sports	4
	Concert	3

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

```
select eventtable.eventname,year(eventtable.eventdate),
sum(totalcost) from eventtable join booking
on eventtable.eventid = booking.eventid
group by eventtable.eventname, year(eventtable.eventdate);
```

eventname	year(eventtable.eventdate)	sum(totalcost)
ev1	2024	4000
ev2	2024	10000
ev3	2024	15000
xamb	2024	4000
ev5	2024	5000
ev6	2024	6000
ev8	2024	8000
ev9	2024	45000
ev10	2024	90000

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

```
select customer.customername, count(distinct eventid) as num_of_eve
from customer join booking
on customer.customerid = booking.customerid
group by customer.customername
having count(distinct eventid)>1;
```

customername	num_of_eve
James	2

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

```
select customer.customername, eventid, sum(totalcost) as tot_amt
from customer join booking
on customer.customerid = booking.customerid
group by customer.customername, booking.eventid
order by customer.customername;
```

	customername	eventid	tot_amt
▶	Anthony	9	45000
	Christopher	2	8000
	James	1	4000
	James	2	2000
	John	8	8000
	Lisa	5	5000
	Mary	Mary	4000
	Michelle	6	6000
	Patricia	10	90000
	Ronald	3	15000

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

```
select eventtable.eventtype, venue.venueid, avg(ticketprice)
from venue join eventtable
on venue.venueid = eventtable.venueid
group by eventtable.eventtype, venue.venueid;
```

	eventtype	venueid	avg(ticketprice)
▶	Movie	1	1000.0000
	Sports	2	5000.0000
	Concert	3	6000.0000
	Movie	4	4000.0000
	Sports	5	5000.0000
	Concert	6	6000.0000
	Movie	2	7000.0000
	Movie	9	10000.0000

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

```
select customer.customername, sum(numtickets)
from customer join booking
on customer.customerid = booking.customerid
where booking.BookingDate
between (select date_sub(now(), interval 30 day))
and now() group by customer.customername;
```

	customername	sum(numtickets)
▶	James	5
	Christopher	4
	Ronald	5
	Mary	1
	Lisa	1
	Michelle	1
	John	1
	Anthony	5
	Patricia	9

Tasks 4: Subquery and its types

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

```
select venue.venuenmae, avg_pri
from venue join (select eventtable.venueid, avg(ticketprice) as avg_pri
from eventtable group by eventtable.venueid) as subq
on venue.venueid = subq.venueid;
```

venuenmae	avg_pri
ASD stadium	1000.0000
xyz stadium	5666.6667
nehru stadium	6000.0000
dadha stadium	4000.0000
leodas stadium	5000.0000
bone park	6000.0000
Ascent	10000.0000

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

```
select * from eventtable
where eventid in (select eventid
from eventtable where (availableseats/totalseats)*100 < 50);
```

EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
5	ev5	2024-01-17	14:00:00	5	9000	4000	5000	Sports
6	ev6	2024-01-18	14:00:00	6	11000	1000	6000	Concert
8	ev8	2024-01-20	14:00:00	2	15000	7000	8000	Sports
9	ev9	2024-01-21	14:00:00	3	17000	7000	9000	Concert
10	ev10	2024-01-22	14:00:00	9	19000	2000	10000	Movie

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

```
select eventtable.eventname, sub.tot_tic_sold
from eventtable join (select booking.eventid, sum(booking.numtickets)
as tot_tic_sold from booking group by booking.eventid) as sub
on eventtable.eventid = sub.eventid;
```

eventname	tot_tic_sold
ev1	4
ev2	5
ev3	5
xamb	1
ev5	1
ev6	1
ev8	1
ev9	5
ev10	9

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

```
INSERT INTO `ticketbookingsystem`.`customer`
(`CustomerID`, `CustomerName`, `Email`, `PhoneNumber`)
VALUES ('12', 'dummy', 'dummy@gmail.com', '1231212122');
select customer.customername
from customer where not exists
(select 1 from booking WHERE booking.customerid = customer.customerid);
```

customername
Daniel
dofen
dummy

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

```
select eventtable.eventid, eventtable.eventname
from eventtable where eventtable.eventid
not in (select distinct booking.eventid from booking);
```

eventid	eventname
7	ev7
11	world cup

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

```
select sub.eventtype, sum(numtickets) as num_of_tic
from (select eventtable.eventtype, booking.numtickets
from eventtable join booking
on eventtable.eventid = booking.eventid) as sub
group by sub.eventtype;
```

eventtype	num_of_tic
Movie	14
Sports	7
Concert	11

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

```
select * from eventtable
where eventtable.ticketprice>(select avg(ticketprice)
from eventtable);
```

EventID	EventName	EventDate	EventTime	VenueID	TotalSeats	AvailableSeats	TicketPrice	EventType
6	ev6	2024-01-18	14:00:00	6	11000	1000	6000	Concert
7	ev7	2024-01-19	14:00:00	2	13000	11000	7000	Movie
8	ev8	2024-01-20	14:00:00	2	15000	7000	8000	Sports
9	ev9	2024-01-21	14:00:00	3	17000	7000	9000	Concert
10	ev10	2024-01-22	14:00:00	9	19000	2000	10000	Movie

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

```
select c.customerid, c.customername,
    (
        SELECT COALESCE(SUM(b.totalcost), 0)
        FROM booking b
        WHERE b.customerid = c.customerid
    ) AS total_revenue
FROM customer c;
```

customerid	customername	total_revenue
1	James	6000
2	Christopher	8000
3	Ronald	15000
4	Mary	4000
5	Lisa	00
6	Michelle	6000
7	John	8000
8	Daniel	0
9	Anthony	45000
10	Patricia	90000
11	dofen	0
12	dummy	0

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

```
SELECT DISTINCT customer.customerid, customer.customername
FROM customer
JOIN booking ON customer.customerid = booking.customerid
WHERE booking.eventid IN (
    SELECT eventid
    FROM eventtable
    WHERE venueid = '3'
);
```

customerid	customername
3	Ronald
9	Anthony

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

```
select sub.eventtype, count(numtickets)
from (select eventtable.eventtype, booking.numtickets
from eventtable join booking
on eventtable.eventid = booking.eventid) as sub
group by sub.eventtype;
```

eventtype	count(numtickets)
Movie	4
Sports	4
Concert	3

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

```
SELECT DISTINCT c.customerid, c.customername FROM customer c
JOIN booking b ON c.customerid = b.customerid
JOIN eventtable e ON b.eventid = e.eventid
WHERE DATE_FORMAT(e.eventdate, '%Y-%m') IN (
    SELECT DISTINCT DATE_FORMAT(eventdate, '%Y-%m')
    FROM eventtable
);
```

customerid	customername
1	James
2	Christopher
3	Ronald
4	Mary
5	Lisa
6	Michelle
7	John
9	Anthony
10	Patricia

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

```
select venue.venuenmae, avg_pri
from venue join (select eventtable.venueid, avg(ticketprice) as avg_pri
from eventtable group by eventtable.venueid) as subq
on venue.venueid = subq.venueid;
```

	venuenmae	avg_pri
▶	ASD stadium	1000.0000
	xyz stadium	5666.6667
	nehru stadium	6000.0000
	dadha stadium	4000.0000
	leodas stadium	5000.0000
	bone park	6000.0000
	Ascent	10000.0000