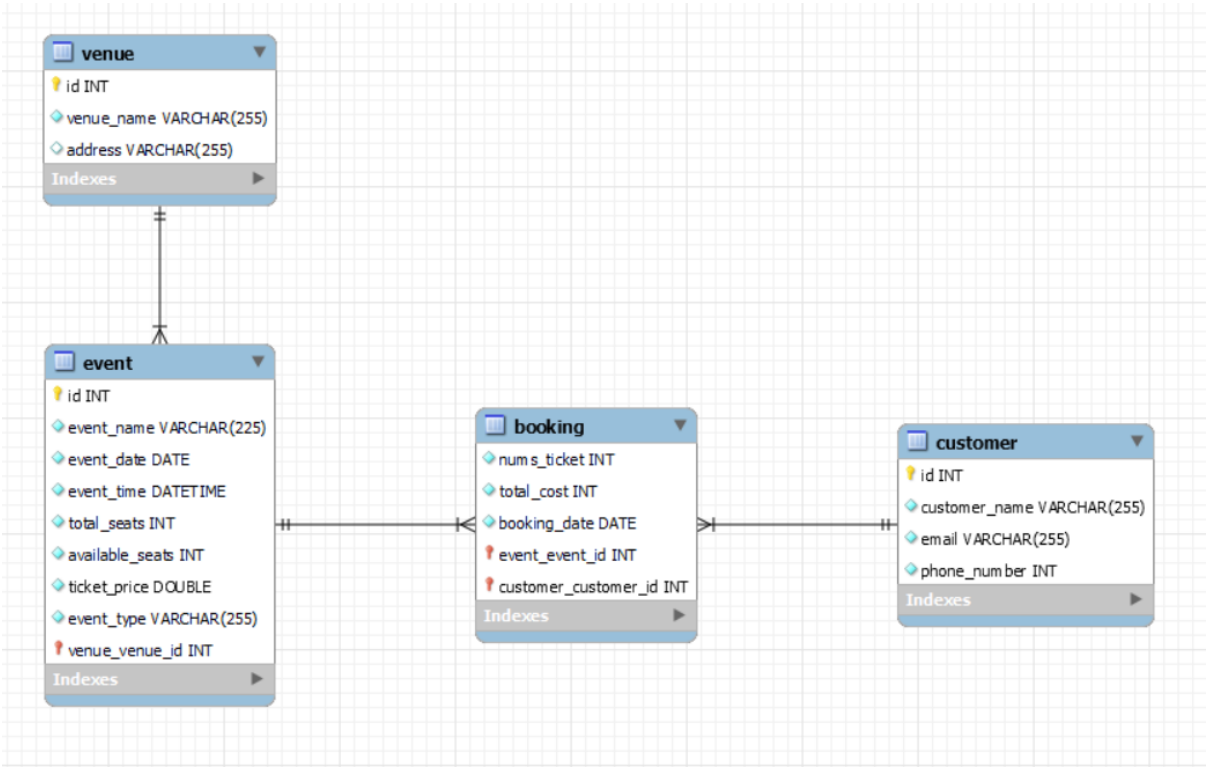
TICKET BOOKING SYSTEM



CODE:

```
#ticket booking Case study
use ticketbooking_feb_hex_24;

#insertions
insert into venue(venue_name,address) values
('mumbai', 'marol andheri(w)'),
('chennai', 'IT Park'),
('pondicherry ', 'state beach');
select * from venue;

insert into customer(customer_name,email,phone_number)
values
('harry potter','harry@gmail.com','45454545'),
('ronald weasley','ron@gmail.com','45454545'),
```

```
('hermione granger','her@gmail.com','45454545'),
('draco malfoy','drac@gmail.com','45454545'),
('ginni weasley','ginni@gmail.com','45454545');
select * from customer;

insert into
event(event_name,event_date,event_time,total_seats,available_seats,ticket_price,event_type,venue_id)
values
('Late Ms. Lata Mangeshkar Musical', '2021-09-12','20:00',320,270,600,'concert',3),
('CSK vs RCB', '2024-04-11','19:30',23000,3,3600,'sports',2),
('CSK vs RR', '2024-04-19','19:30',23000,10,3400,'sports',2),
('MI vs KKR', '2024-05-01','15:30',28000,100,8000,'sports',1);
select * from event;

insert into booking values
(1,1,2,640,'2021-09-12'),
(1,4,3,960,'2021-09-12'),
(2,1,3,10800,'2024-04-11'),
(2,3,5,18000,'2024-04-10'),
(3,5,10,34000,'2024-04-15'),
(4,2,4,32000,'2024-05-01');



#SQL Queries - Task 2

-- 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;
```

```
update event set event_name='Conference Cup' where id= 5;
-- 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 '2024-04-11' AND '2024-05-01';

-- 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 3,2;

select *
from customer
limit 5,5; #records 6-10

-- 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 ASC ;
```

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

```
select *
from event
where event_name NOT LIKE 'c%' AND event_name NOT LIKE 'x%' and event_name NOT LIKE 'y%';
```

#SQL Queries - Task 3

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

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

```
select e.venue_id,v.venue_name,AVG(e.ticket_price )
from event e, venue v
where v.id = e.venue_id
group by e.venue_id;
```

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

```
select SUM((total_seats  - available_seats) *  ticket_price) #We can perform arithmetic ops in select statement
from event;
```

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

```
select event_name,MAX((total_seats  - available_seats) *  ticket_price) as total_sales
from event
group by event_name
```

```
order by total_sales DESC
limit 0,1;

-- 4. Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.
select event_name, total_seats  - available_seats as total_tickets_sold
from event
group by event_name;

-- 5. Write a SQL query to Find Events with No Ticket Sales.
select event_name
from event
where available_seats = total_seats;

-- 6. Write a SQL query to Find the Customer Who Has Booked the Most Tickets.
select customer_name, SUM(b.num_tickets) as tickets_booked
from booking b, customer c
where b.customer_id = c.id
group by customer_name
order by tickets_booked DESC
limit 0,1;

-- 7. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type
select e.event_name , sum(b.num_tickets) as total_tickets  , Month(b.booking_date) as Month
from booking b , event e
where e.id=b.event_id
group by  MONTH(b.booking_date);

-- 9. Write a SQL query to list customer who have booked tickets for multiple events.
-- 11.Write a SQL query to list users who have booked tickets for multiple events.
select  c.customer_name ,count(customer_id ) as event_booked
from booking b, customer c
```

```
where c.id=b.customer_id
group by customer_id;

-- 10. Write a SQL query to calculate the Total Revenue Generated by Events for Each Customer
select c.customer_name as Customer_Name, sum(b.total_cost) as Revenue
from booking b JOIN  customer c ON c.id = b.customer_id
group by c.customer_name
order by Revenue DESC;

-- 11.Write a SQL query to list users who have booked tickets for multiple events.
select  c.customer_name ,count(customer_id ) as event_booked
from booking b, customer c
where c.id=b.customer_id
group by customer_id
having event_booked>1;

-- 12. 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 event e , booking b , customer c
where e.id=b.event_id AND
b.customer_id = c.id
group by c.customer_name
order by total_revenue DESC;

-- 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(ticket_price) as Average_price
from event e , venue v
where v.id =e.venue_id
group by 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_name, SUM(b.num_tickets) as Number_Of_tickets
from booking b JOIN  customer c ON c.id = b.customer_id
where b.booking_date between DATE_SUB('2024-04-30',INTERVAL 30 DAY) and  '2024-04-30'
group by c.customer_name;
```

Task 4: Subquery and its types

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

```
select venue_id,AVG(ticket_price) as Avg_Price
from event
where venue_id IN (select id from venue)
group by venue_id;
```

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

```
select event_name
from event
where id IN ( select id
                        from event
        where (total_seats - available_seats) > (total_seats/2));
```

-- 3. Find Events having ticket price more than average ticket price of all events

```
select event_name
from event
where ticket_price >  (select avg(ticket_price) from event);
```

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

```
select customer_name
from customer
```

```
 where NOT EXISTS (select distinct c.customer_name
                                    from customer c join booking b ON b.customer_id = c.id);

-- 5. List Events with No Ticket Sales Using a NOT IN Subquery.
select *
from event
where id NOT IN (select distinct event_id
                                    from booking);

-- 6. Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause.
select event_type, SUM(num_tickets) as tickets_sold
from event e
JOIN booking b ON e.id = b.event_id
group by event_type;

-- 7. Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause.
select event_name, ticket_price
from event
where ticket_price > (select avg(ticket_price)
                                        from event);

-- 8. Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery.
select id, customer_name, sum(total_cost) as revenue
from  customer c
JOIN booking b ON c.id = b.customer_id
group by id, customer_name;

-- 9. List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause.
select id, customer_name
```

```
from customer
where id in (select distinct customer_id
                    from booking b
        JOIN event e ON b.event_id = e.id where venue_id = 1);

-- 10. Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY.
select event_type, SUM(num_tickets) as number_of_ticket
from event e
JOIN booking b ON e.id = b.event_id
group by event_type;

-- 12. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery
select venue_id, venue_name, avg(ticket_price) as ticket_price
from venue v
JOIN event e on v.id = e.venue_id
group by venue_id, venue_name;
```