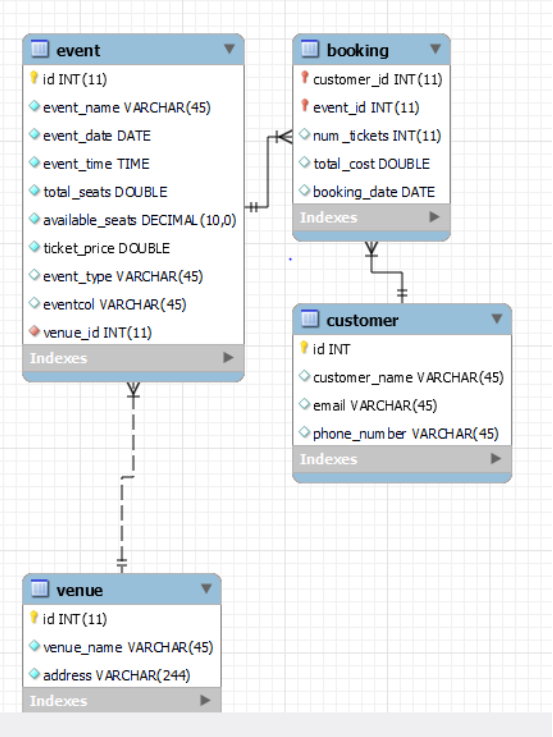
TICKET BOOKING ASSIGNMENT

ER Diagram:



Task 2:

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

insert into venue(venue_name,address) values

('mumbai', 'marol andheri(w)'),

('chennai', 'IT Park'),

('pondicherry ', 'state beach');

---

insert into customer(id,customer_name,email,phone_number) values

('1','harry potter','harry@gmail.com','45454545'),

('2','ronald weasley','ron@gmail.com','45454545'),

('3','hermione granger','her@gmail.com','45454545'),

('4','draco malfoy','drac@gmail.com','45454545'),

('5','ginni weasley','ginni@gmail.com','45454545');

```
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);
```

```
insert into booking values

(1,1,2,640,'2021-09-12'),

(4,4,3,960,'2021-09-12'),

(2,3,2,10800,'2024-04-11'),

(5,3,5,18000,'2024-04-10'),

(3,2,4,32000,'2024-05-01');
```

2.Write a SQL query to list all Events.

```
select * from venue;

select * from customer;

select * from event;

select * from booking;
```

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

```
select * from event

where available_seats > 0;

update event SET event_name='Conferece CUP' where id=4;
```

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_name LIKE '%concert%';
```

8. Write a SQL query to retrieve customers 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'; # ends number with 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 'y%' AND event_name NOT LIKE 'x%'  AND event_name NOT LIKE 'z%';
```

#TASK 3

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

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

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

```
select SUM((total_seats  - available_seats) *  ticket_price)
   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,sum(total_seats - available_seats) as bal_seats
                        from event
                    group by event_name;
```

```
select event_name
from event
where bal_seats = 0;
```

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 List Events and the total number of tickets sold for each month.

```
select
   year(e.event_date) as event_year,
   month(e.event_date) as event_month,
   count(b.event_id) as total_tickets_sold
from event e
join     booking b on e.id = b.event_id
group by year(e.event_date), month(e.event_date);
```

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;
```

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

```
select event_type,sum(total_seats-available_seats) as num_of_tickets
from event
group by event_type;
```

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

```
select year(event_date) as event_year, sum(ticket_price) as total_revenue
                    from event
                    group by year(event_date);
```

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

```
select e.event_name, c.customer_name, b.num_tickets
                    from event e,customer c, booking b
                    where e.id = b.event_id AND
                    b.customer_id = c.id;
                     select c.customer_name , count(c.id) as events_booked
                    from event e,customer c, booking b
                    where e.id = b.event_id AND
                    b.customer_id = c.id
                    group by c.customer_name ;
                     select c.customer_name , count(c.id) as events_booked
                    from event e,customer c, booking b
                    where e.id = b.event_id AND
                    b.customer_id = c.id
                    group by c.customer_name
                    having events_booked>1;
```

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

```
 select * from  event e  JOIN booking b  ON e.id = b.event_id JOIN  customer c ON c.id = b.customer_id;

select c.customer_name, count(c.id) as Number_Of_bookings from  event e  JOIN booking b ON e.id = b.event_id JOIN  customer c ON c.id = b.customer_id group by c.customer_name;

select c.customer_name as Customer_Name, sum(b.total_cost) as Revenue from  event e JOIN booking b  ON e.id = b.event_id JOIN  customer c ON c.id = b.customer_id group by c.customer_name order by 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.id, avg(e.ticket_price) as average_ticket_price
```

```
from event e
join venue v on e.venue_id = v.id
group by e.event_type, v.id;
```

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  event e  JOIN booking b  ON e.id = b.event_id 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;
```

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

```
select id,event_name ,(total_seats - available_seats) as total_seats
from event;
```

4. Find Users 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);

 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 event_name

from event

where id NOT IN ( select id

                        from event

        where (total_seats - available_seats) = total_seats);
```

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

```
SELECT event_type, SUM(total_seats - available_seats) AS total_tickets_sold

   FROM event

   group by event_type;
```

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

```
SELECT id, 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 c.id, c.customer_name,

     SUM((b.num_tickets) * e.ticket_price) AS total_revenue

FROM customer c

JOIN booking b ON c.id = b.customer_id

JOIN event e ON e.id = b.event_id
```

GROUP BY c.id,c.customer_name;

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

```
SELECT DISTINCT  c.id, c.customer_name

from customer c

JOIN booking  b ON c.id = b.customer_id

WHERE b.event_id IN (

   SELECT event_id

   FROM event

   WHERE 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(e.total_seats - e.available_seats) AS total_tickets_sold

FROM event e

GROUP BY e.event_type;
```

NOTE: For me its hard to use subquery to get answer for this.

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

```
SELECT DISTINCT c.id, c.customer_name, DATE_FORMAT(b.booking_date, '%Y-%m') AS booking_month

FROM customer c

JOIN booking b ON c.id = b.customer_id

JOIN event e ON b.event_id = e.id

WHERE DATE_FORMAT(b.booking_date, '%Y-%m') IN (

   SELECT DISTINCT DATE_FORMAT(booking_date, '%Y-%m')

   FROM booking

);
```

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

```
SELECT venue_id, AVG(ticket_price) AS average_ticket_price
```

```
FROM (
    SELECT venue_id, ticket_price
    FROM event
) AS event_ticket_prices
GROUP BY venue_id;
```