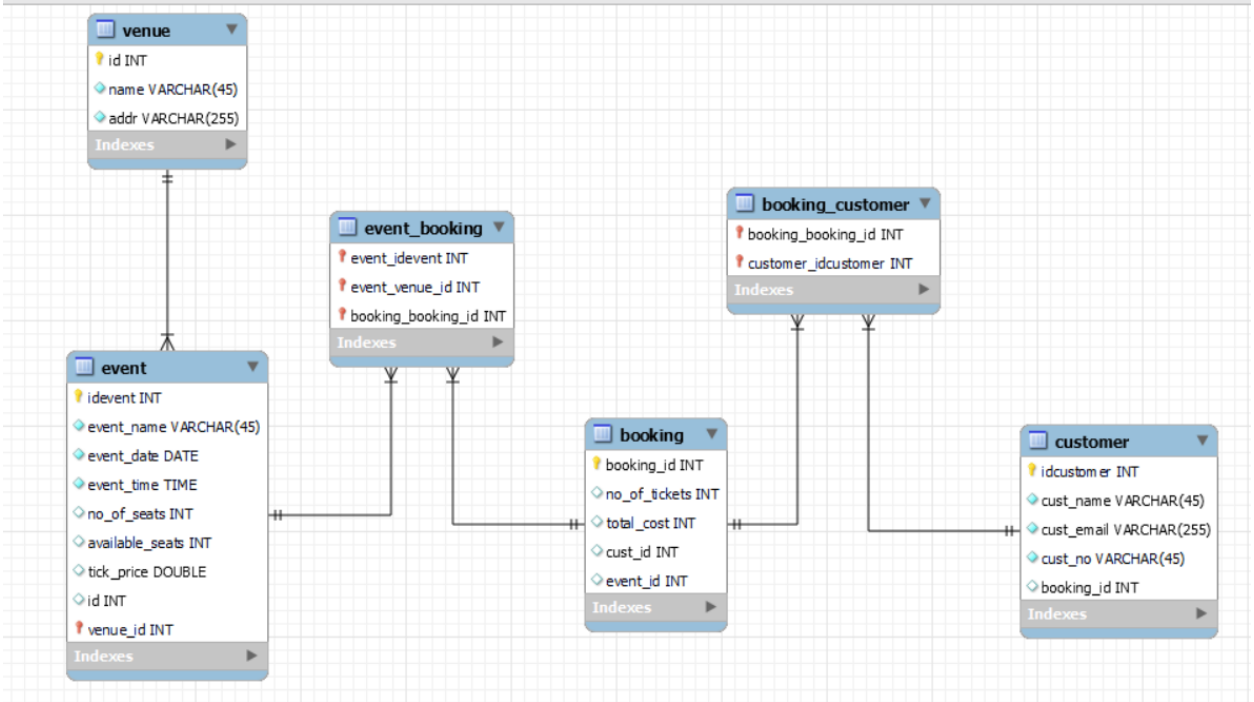
ASSIGNMENT 2

TICKET BOOKING

ER DIAGRAM:



QUERIES:

```
-- 28.02.2024
#ASSIGNMENT-1
#ticket booking Case study (TASK 1)
create database hexfeb;
use hexfeb;
show databases;


#insertions
create table venue(id int primary key auto_increment,
venue_name varchar(50),address varchar(50));
insert into venue(venue_name,address) values
('mumbai', 'marol andheri(w)'),
('chennai', 'IT Park'),
('new delhi', 'pragathi maidan'),
('ooty', 'race course'),
('pondicherry ', 'state beach');
```

```
select * from venue;
drop table customer;
create table customer(customer_id int primary key auto_increment,
customer_name varchar(50),email varchar(50),
phone_number varchar(10));
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'),
('aadhya','aadhya@gmail.com','45454545'),
('zoya','zoya@gmail.com','45454545'),
('louis','louis@gmail.com','45454545'),
('rachel','rachel@gmail.com','45454545'),
('bob','bob@gmail.com','45454545');

select * from customer;
create table event(event_id int primary key auto_increment,event_name varchar(50),
event_date date,event_time time,total_seats int,
available_seats int,ticket_price double,
event_type varchar(50),venue_id int);
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),
('JOYAL DANCE', '2020-05-02','15:30',30000,500,8000,'dance',4),
('V SING','2022-08-25','11:30:00',5000,2000,950,'KAROKE',3),
 ('THE EPIC SHOW','2023-08-20','11:00:00',10000,4000,3000,'CIRCUS',3),
 ('MR Neil on set','2024-08-25','10:30:00',13000,6000,280,'BAND',5),
 ('PARK','2024-08-25','08:30:00',null,null,2000,'AMUSEMENT PARK',1),
 ('VR','2024-08-25','11:45:00',null,null,300,'VIRTUAL REALITY',4);

select * from event;
create table booking(event_id int,customer_id int,num_tickets int,total_cost double,
booking_date date);
```

```
insert into booking values
(4,1,2,640,'2021-09-12'),
(4,4,3,960,'2021-09-12'),
(5,2,3,10800,'2020-01-11'),
(5,3,5,18000,'2020-02-10'),
(6,5,10,3000,'2022-07-15'),
(8,7,15,50000,'2024-01-15'),
(7,9,30,7000,'2023-02-02'),
(9,6,11,34000,'2024-08-25'),
(10,8,4,32000,'2024-08-21');
select * from booking;
#SQL Queries - Tasks 2: Select, Where, Between, AND, LIKE:
-- 1. Write a SQL query to insert at least 10 sample records into each table.
-- 2. Write a SQL query to list all Events.
select event_name,event_type
from event;

-- 3. Write a SQL query to select events with available tickets.
select event_type,available_seats
from event;
-- 4. Write a SQL query to select events name partial match with ‘cup’.
select event_name
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 2000 and 4000;
-- 6. Write a SQL query to retrieve events with dates falling within a specific range.
select event_name,event_type
from event
where event_date between '2021-01-01' and '2023-12-30';
-- 7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their
-- name.
select event_name,event_type,available_seats
from event
where event_type 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

/*
LIMIT <offset>,<number_of_records>
- offest is the record after which we start counting - so if offset is 3 we start from 4
- number_of_records given will be displayed
*/

-- 9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.
select *
from booking
group by num_tickets
having 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 'c%' AND event_name NOT LIKE 'x%';

/*Task  3: Aggregate functions, Having, Order By, GroupBy and Joins:*/

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

select event_id,event_name,avg(ticket_price) as ticketp
```

```
from event
group by event_id;/*if aggregate function used with other columns use group by*/
-- 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 e.event_id,e.event_name,b.num_tickets
from event e,booking b
where e.event_id=b.event_id
order by num_tickets desc/*instead use normal mth*/
limit 0,1;

select event_name,max((total_seats-available_seats)*ticket_price) as sales
from event
group by event_name
order by 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 sold_tickets
from event;

-- 5. Write a SQL query to Find Events with No Ticket Sales.
select e.event_id,e.event_name,b.num_tickets
from event e,booking b
where e.event_id=b.event_id and total_seats=available_seats;

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

-- 7. Write a SQL query to List Events and the total number of tickets sold for
-- each month.
select month(event_date) as mon,count(*) as sold
```

```
from event
group by  month(event_date);

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

select  venue_id,avg(ticket_price) as price
from event
group by 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 sold
from event
group by event_type;

-- 10. Write a SQL query to calculate the total Revenue Generated by Events in Each Year.
select event_id,event_date,sum((total_seats-available_seats)*ticket_price) as revenue
from event
group by event_id
order by event_date desc;
-- 11. Write a SQL query to list users who have booked tickets for multiple events.
select count(c.customer_id) as booked,c.customer_name/*read question properly along with the
relation made*/
from event e,booking b,customer c
where e.event_id=b.event_id and b.customer_id=c.customer_id
group by c.customer_name
having booked>1;

-- 12. Write a SQL query to calculate the Total Revenue Generated by Events for Each User.
select event_id,event_date,sum((total_seats-available_seats)*ticket_price) as revenue
from event
group by event_id
order by event_date desc;
-- 13. Write a SQL query to calculate the Average Ticket Price
-- for Events in Each Category and Venue.
select e.event_type,avg((total_seats-available_seats)*ticket_price) as price
from event e,venue v
where v.id=e.venue_id
GROUP BY e.event_type
;
```

```
-- 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.event_id = b.event_id JOIN  customer c ON c.customer_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:
/*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 tp
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 event_name
from event
where ticket_price >  (select avg(ticket_price) 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);
-- 5. List Events with No Ticket Sales Using a NOT IN Subquery.
select event_name
from event
where event_id not in
(select 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,(select sum(total_seats-available_seats) from event)
 as tickets_sold from event;
/*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) as price from event);
-- 8. Calculate the Total Revenue Generated by Events for Each User Using a Correlated
Subquery.
select customer_name,(select sum(ticket_price) from e.event
where e.customer_id=c.customer_id)as revenue from c.customer;
/*9. List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the
WHERE
Clause.*/
select distinct customer_id from booking
where event_id in( select event_id from event where venue_id=3);
/*10. Calculate the Total Number of Tickets Sold for Each Event Category
 Using a Subquery with GROUP BY.*/
 select event_type,count(*) as sold
 from event
 group by(event_type);

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