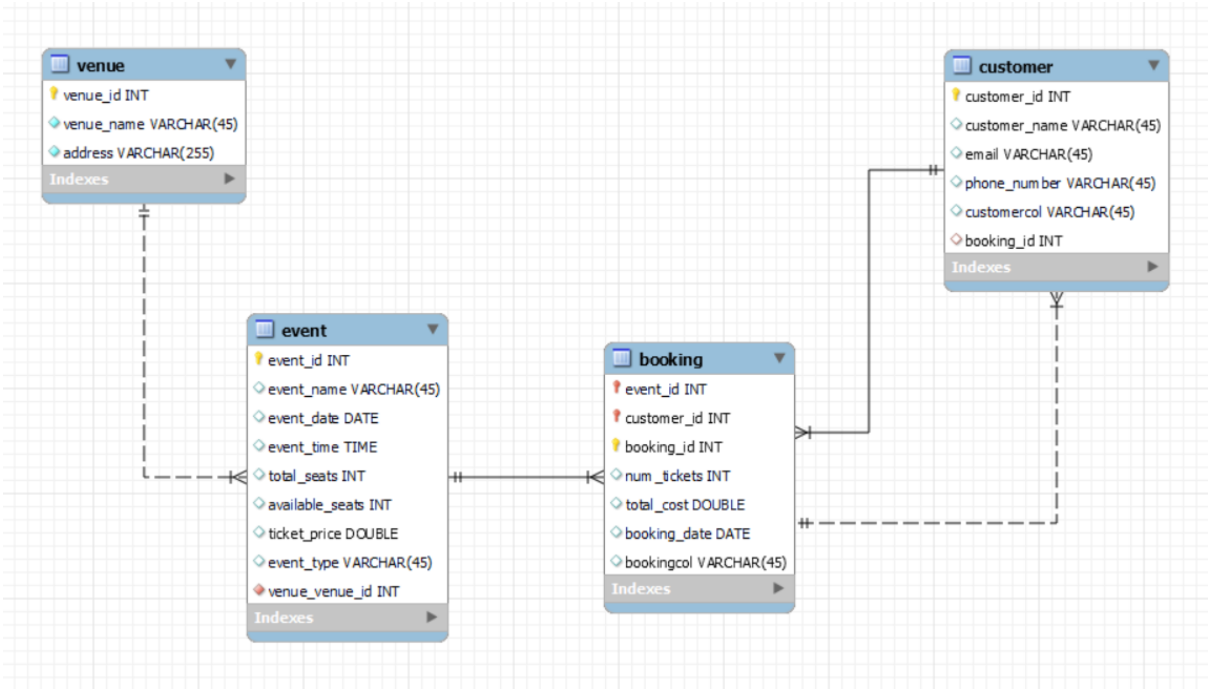
ASSIGNMENT 1 - TICKET BOOKING SYSTEM

ER DIAGRAM:



Queries:

create ticketbookingdb;
use ticketbookingdb;

-- creating tables
create table venue(venue_id int primary key auto_increment not null,
venue_name varchar(45) not null, address varchar(255) not null);

create table booking(booking_id int primary key auto_increment not null, num_tickets int not null,
total_cost double not null, booking_date date not null, customer_id int, event_id int,
foreign key(customer_id) references customer(customer_id), foreign key(event_id) references event(event_id));

create table event(event_id int primary key auto_increment not null,
event_name varchar(45) not null, event_date date not null,
event_time time not null, total_seats int not null,
available_seats int not null, ticket_price double not null,
event_type varchar(45) not null, venue_id int , booking_id int, foreign key(venue_id) references venue(venue_id),
foreign key (booking_id) references booking(booking_id));

create table customer(customer_id int primary key auto_increment not null,
customer_name varchar(45) not null, email varchar(255) not null, phone_number varchar(10) not null,
booking_id int, foreign key(booking_id) references booking(booking_id));

```sql
-- Task 2 :

-- Q1. Write a SQL query to insert at least 10 sample records into each table.
-- insertion of values

-- insertion into venue table
  insert into venue(venue_name, address) values('Mumbai','Navi'),
('Kolkata','Stadium'),
('Pondicherry','Pondy Marina'),
('Chennai','Phoenix mall'),
('Coimbatore','Eesha'),
('Kanyakumari','Statue'),
('Kerala','Kochi'),
('Kerala','Guruvayoor'),
('Goa','Beach'),
('Dubai','Desert');

-- insertion into customer table
insert into customer(customer_name, email, phone_number,booking_id) values
('Pavithra','pavi@gmail.com','7418511592',4),
('Merushiga','meru@gmail.com','9747047296',3),
('Sandeep','kmc@gmail.com','6381701721',7),
('Agas','agas@gmail.com','9876543210',6),
('Rashmi','rashmi@gmail.com','8759463210',1),
('Patrick','patrick@gmail.com','7895463210',9),
('Thara','thara@gmail.com','6359874123',8),
('Mufi','mufi@gmail.com','8597463214',5),
('Swethaa','sg@gmail.com','9747074962',2),
('Anitha','ani@gmail.com','9994733289',10);

-- insertion into booking table
insert into booking(num_tickets, total_cost, booking_date, customer_id, event_id) values
(1,50,'2024-01-01',1,1),
(3,150,'2024-02-01',7,7),
(2,100,'2024-01-15',2,8),
(4,200,'2024-02-14',4,2),
(1,50,'2024-01-03',8,6),
(2,100,'2024-01-20',3,9),
(5,250,'2024-01-25',6,5),
(6,300,'2024-02-10',9,3),
(1,50,'2024-01-05',5,4),
(1,50,'2024-01-14',10,10);

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

```
('Dance','2024-05-05','21:00',5,5,10,'concert',1,3),
('Song','2024-05-05','17:00',5,5,10,'concert',5,4),
('Rally song','2024-05-15','17:00',5,3,10,'concert',9,2),
('T20','2024-05-24','18:00',50,40,10,'Sports',6,7),
('World cup','2024-04-05','11:00',55,40,100,'Sports',2,1),
('Belly dance','2024-04-15','18:00',25,5,10,'concert',7,6),
('World war II','2024-04-04','14:00',500,412,25,'movie',10,8),
('Angry birds','2024-05-06','21:00',50,5,10,'movie',8,9),
('Dancing lords','2024-05-06','09:00',50,5,10,'movie',4,10),
('Solo','2024-04-25','18:00',25,5,10,'concert',3,5);

-- Q2. Write a SQL query to list all Events.

select event_name from event;

-- Q3. Write a SQL query to select events with available tickets

select event_id, event_name as 'events with available tickets' from event
where available_seats> 0;

-- Q4. Write a SQL query to select events name partial match with ‘cup’.

select event_name from event
where event_name like '%cup%';

-- Q5. Write a SQL query to select events with ticket price range is between 10 to 25.

select event_id, event_name, ticket_price from event where ticket_price between 10 and 25;

-- Q6. Write a SQL query to retrieve events with dates falling within a specific range.

select * from event
where event_date between '2024-04-01' and '2024-04-15';

-- Q7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.

select event_id, event_name, available_seats, event_type from event
where available_seats >0 and event_type = 'concert';

-- Q8. Write a SQL query to retrieve users in batches of 5, starting from the 6th user.

select * from customer
limit 5,5;
```

-- Q9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.

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

-- Q10. Write a SQL query to retrieve customer information whose phone number end with ‘000’

```
select * from customer where phone_number like '%000';
update event set total_seats=16000 where event_id=4;
update event set total_seats=15700 where event_id=7;
```

-- Q11. Write a SQL query to retrieve the events in order whose seat capacity more than 15000.

```
select event_id, event_name, total_seats from event
where total_seats>15000
order by total_seats asc;
update event set event_name='xylo choreo' where event_id=5;
```

-- Q12. Write a SQL query to select events name not start with ‘x’, ‘y’, ‘z’

```
select event_name from event
where event_name not like 'x%' and event_name not like 'y%' and event_name not like 'z%';
```

-- Task 3---------

-- Q1. Write a SQL query to List Events and Their Average Ticket Prices

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

-- Q2. Write a SQL query to Calculate the Total Revenue Generated by Events.

```
select sum(total_cost) as total_revenue from booking;
```

-- Q3. Write a SQL query to find the event with the highest ticket sales.

```
select e.event_name from event e, booking b
where e.event_id=b.event_id and b.num_tickets=(select max(num_tickets) from booking);
```

-- Q4. Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.

```
select e.event_id, e.event_name, b.num_tickets
from event e, booking b
where e.event_id=b.event_id
group by event_name
order by num_tickets asc;
```

```
-- Q5. Write a SQL query to Find Events with No Ticket Sales.

select event_name as event_with_no_ticket_sales, total_seats, available_seats from event
where total_seats = available_seats;

-- Q6. Write a SQL query to Find the User Who Has Booked the Most Tickets.

select c.customer_name, b.num_tickets
from customer c, booking b
where c.customer_id=b.customer_id and num_tickets=(select max(num_tickets) from booking);

-- Q7. Write a SQL query to List Events and the total number of tickets sold for each month.

select e.event_id, e.event_name, e.event_date,
  sum(b.num_tickets) as total_tickets_sold
from event e
JOIN booking b on e.event_id = b.event_id
group by e.event_id, e.event_name, e.event_date
order by e.event_date;

-- Q8. 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) as avg_price
    from event e join venue v
    on e.venue_id=v.venue_id
    group by v.venue_name;

-- Q9. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.

select e.event_type, sum(b.num_tickets) as tickets_sold
    from booking b join event e
    on b.event_id=e.event_id
    group by event_type;

-- Q10. 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 revenue
    from booking
    group by year(booking_date);

-- Q11. Write a SQL query to list users who have booked tickets for multiple events.

select c.customer_id, c.customer_name,
    count(distinct b.event_id) as num_of_events
    from customer c
    join booking b
    on c.customer_id=b.customer_id
    group by c.customer_id, c.customer_name
    having count(distinct b.event_id) >1;

-- Q12. Write a SQL query to calculate the Total Revenue Generated by Events for Each User.

 select b.customer_id, c.customer_name, sum(b.total_cost) as total_revenue
    from booking b
    join customer c on c.customer_id=b.customer_id
    group by b.customer_id;

-- Q13. 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(e.ticket_price) as avg_price
    from event e join venue v
    on e.venue_id=v.venue_id
    group by v.venue_name, e.event_type
    order by e.event_type;

-- Q14. Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days.

Select c.customer_id, c.customer_name, sum(b.num_tickets) as total_tickets_purchased, b.booking_date
    from customer c join booking b
    on c.customer_id=b.customer_id
    where b.booking_date>=current_date - interval 30 day
    group by c.customer_id;

-- Task 4 :
-- Q1. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery.

select e.venue_id, avg(e.ticket_price) as avg_ticket_price
    from event e, venue v
    where e.venue_id in (select venue_id from venue)
    group by e.venue_id;

-- Q2. Find Events with More Than 50% of Tickets Sold using subquery.

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

```
-- Q3. Calculate the Total Number of Tickets Sold for Each Event.

select event_id, sum(num_tickets) as total_tickets_sold
    from booking
    where event_id in (select event_id from event)
    group by event_id;

-- Q4. 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.customer_id);

-- Q5. List Events with No Ticket Sales Using a NOT IN Subquery.

select e.event_name
    from event e
    where e.event_id = (select event_id from booking where num_tickets=0);

-- Q6. Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM
Clause.

select event_id, num_tickets
    from booking
    group by event_id;

-- Q7. Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the
WHERE Clause.

select e.event_name, e.ticket_price
    from event e
    where e.ticket_price > (select avg(ticket_price) from event);

-- Q8. Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery.

select c.customer_name, sum(b.total_cost) as total_revenue
    from customer c, booking b
    where c.customer_id=b.customer_id
    group by b.customer_id;

-- Q9. List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE
Clause

select c.customer_name
    from customer c
    join booking b on c.customer_id=b.customer_id
    join event e on b.event_id=e.event_id
    join venue v on v.venue_id=e.venue_id
    where v.venue_name!='Kerala';
```

-- Q10. Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY.

```
select e.event_type, sum(b.num_tickets) as total_tickets
    from event e
    join booking b on e.event_id=b.event_id
    group by e.event_type;
```