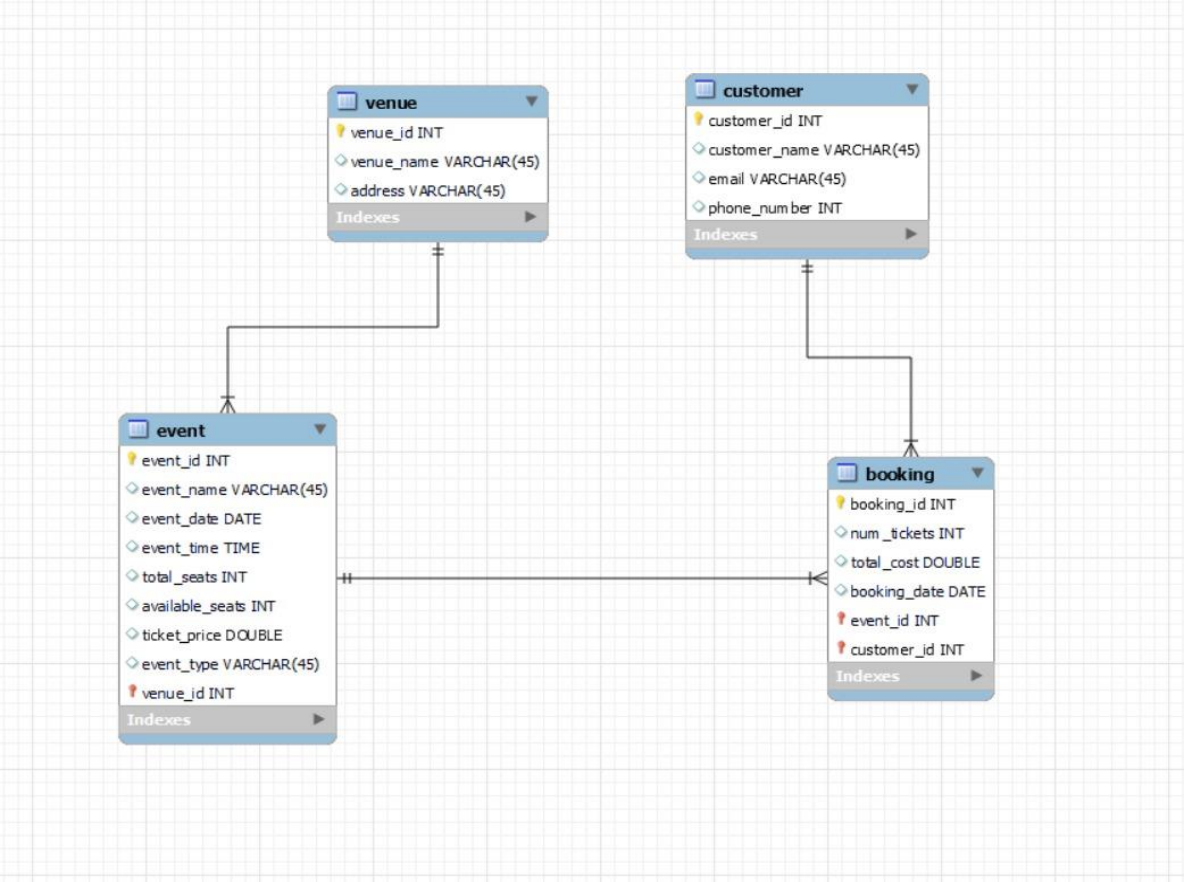
TICKET BOOKING SYSTEM

ER DIAGRAM



use TicketBooking;

show databases;

show tables;

insert into venue(venue_name,address) values

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

('chennai', 'IT Park'),

('pondicherry', 'state beach'),

('bangalore','KR Park'),

('kerala','Vargala-beach'),

('chennai','Anna Museum'),

('Agra','MS University'),

('Bangalore','Line Cross(A)'),

('Kerala','Vagamon'),

('Pondicherry','Gandhi Park');

select * from venue;

insert into customer(customer_name,email,phone_number) values

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

('Ronald weasley','ron@gmail.com','45454546'),

('Hermione granger','her@gmail.com','45454547'),

('Draco malfoy','drac@gmail.com','45454548'),

('Ginni weasley','ginni@gmail.com','45454549'),

('Longbottom','bottom@gmail.com','45454553'),

('Lunalovegood','lovegood@gmail.com','45454554'),

('Prince','pri@gmail.com','45454555'),

('Azkaban','ban@gmail.com','45454556'),

('John','john@gmail.com','45454557');

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','20:00:00',320,270,600,'concert',8);

insert into event(event_name,event_date,event_time,total_seats,available_seats,ticket_price,event_type,venue_id) values

('AR Rahman Musical Fest', '2021-10-24','20:00:00',430,275,900,'concert',9),

('Pro Kabady', '2021-10-12','20:00:00',340,280,680,'sports',4),

('CSK vs MI', '2021-10-12','30:00:00',320,270,600,'sports',9),

('CSK vs RCB', '2021-09-25','20:40:00',320,290,690,'sports',5);

select * from event;

insert into booking(num_tickets,total_cost,booking_date,customer_id,event_id) values

(1650,900,'2023-12-24',5,4);

insert into booking(num_tickets,total_cost,booking_date,customer_id,event_id) values

(1750,800,'2023-10-24',6,5),

(1760,809,'2023-10-12',7,3),

```
(1750,890,'2023-08-24',8,2),
(1750,800,'2023-10-24',8,2);
select * from booking;

#T2
-- Q3.Write a SQL query to select events with available tickets.
select * from Event where available_seats > 0;
-- Q4.Write a SQL query to select events name partial match with ‘cup’.
SELECT * from event WHERE event_name LIKE '%cup%';
-- Q5.Write a SQL query to select events with ticket price range is between 1000 to 2500.
SELECT ticket_price from event where ticket_price>=1000 and ticket_price<=2500;
-- Q6.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';
-- 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 ORDER BY customer_id 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="%000";
-- Q11.Write a SQL query to retrieve the events in order whose seat capacity more than 15000.
select * from event where total_seats>15000;
-- Q12. Write a SQL query to select events name not start with ‘x’, ‘y’, ‘z’ .
select * from event where event_name not like 'x%' and event_name not like 'y%' and event_name not like 'z%';

#T3
-- 1. Write a SQL query to List Events and Their Average Ticket Prices.
SELECT event_id,event_name,event_type,AVG(ticket_price) AS AverageTicketPrice FROM event GROUP BY event_id;
-- 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,event_name,b.num_tickets from event e,booking b where e.event_id=b.event_id order by num_tickets desc limit 0,1;
```

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

```
select event_id,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,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_id,c.customer_name,sum(b.num_tickets) as MostBookedtickets from customer c,booking b where c.customer_id=b.customer_id group by customer_name

 order by MostBookedtickets desc limit 0,1;
```

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

```
select  venue_id,avg(ticket_price) as avgprice 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 ticketssold 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;
```

-- 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 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,c.customer_id 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 v.venue_name,e.event_type,avg(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;
```

#T4

-- 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 venue_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 event_id IN ( select event_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, SUM(total_seats-available_seats) AS total_tickets_sold FROM event GROUP BY event_name;
```

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

```
SELECT customer_id, customer_name FROM customer WHERE customer_id NOT IN (SELECT DISTINCT customer_id FROM booking);
```

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

```
select event_id,event_name from event where event_id NOT IN ( select event_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_id,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 event_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.

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

```
SELECT DISTINCT c.customer_id, c.customer_name from customer c JOIN booking b ON c.customer_id = b.customer_id WHERE b.event_id IN ( SELECT event_id FROM event

WHERE venue_id = 4);
```

-- 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;
```

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