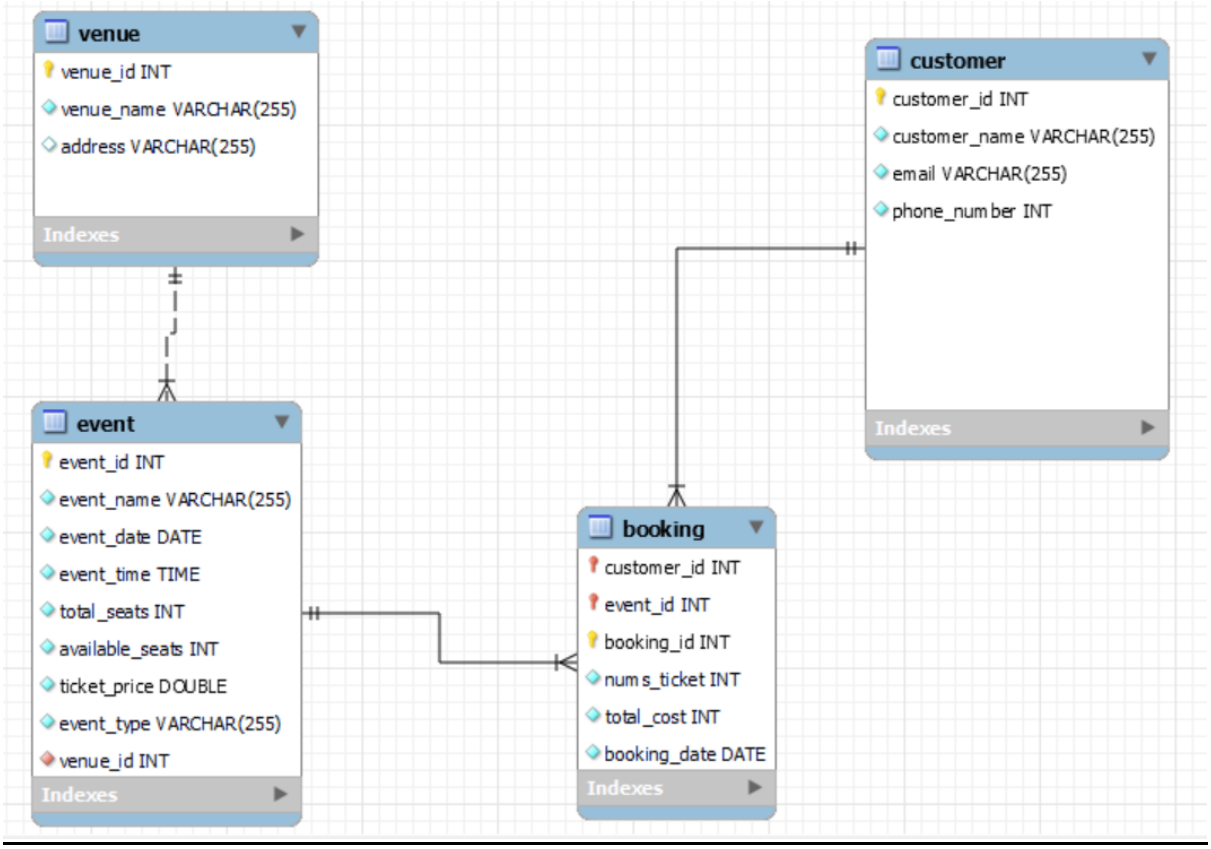
ASSIGNMENT – 5 Ticket Booking System

ER DIAGRAM:



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
```

(4,1,2,640,'2021-09-12'),

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

(5,1,3,10800,'2024-04-11'),

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

(6,5,10,34000,'2024-04-15'),

(7,2,4,32000,'2024-05-01');

#SQL Queries - Task 2

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

select * from event;

update event SET event_name='Conferece CUP' where id=7;

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

select *

from event

where event_name LIKE '%cup%';

-- 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';

-- 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
*/

-- 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

-- 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%';

#Level 2: Multi Table Queries using Manual Mapping Technique

-- display list of events hosted by venue 'chennai'.

select e.id,e.event_name,e.event_date,e.event_time,e.total_seats
from event e,venue v
where v.id = e.venue_id AND v.venue_name='chennai';

-- select customers that have booked tickes for event 'csk v rcb'  game with id=5;

select c.customer_name,email,phone_number
from customer c, booking b
where c.id = b.customer_id AND b.event_id=5;
```

#Display the names of venues visited by customer with email 'harry@gmail.com'

```
select v.venue_name,v.address,c.customer_name
from venue v,booking b,event e,customer c
where v.id=e.venue_id AND
e.id = b.event_id AND
b.customer_id = c.id AND
c.email='harry@gmail.com';
```

```
/*
. 1.Write a SQL query to List Venues and Their Average Ticket Prices.
*/
```

```
SELECT e.event_name, AVG(e.ticket_price) AS average_ticket_price
FROM event e
GROUP BY e.event_name;
```

/*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;
```

#note: We can join multiple tables like venue and fetch extra info from there like venue_name.

#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;
```

#5.5. Write a SQL query to Find Events with No Ticket Sales.

```
SELECT e.event_name
FROM event e
WHERE (e.total_seats - e.available_seats) = 0;
```

#6.Write a SQL query to Find the Customer Who Has Booked the Most Tickets.

#plan: first, find the tickets booked by each customer. then find the most

```
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 Typ

#plan- first display all customer_name and event_name with seats booked and then

#step 2: I will find those customers who have booked 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;
```

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

```
SELECT v.venue_name, AVG(e.ticket_price) AS average_ticket_price
FROM event e
JOIN venue v ON e.venue_id = v.id
GROUP BY v.venue_name;
```

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

```
SELECT e.event_type, SUM(e.total_seats - e.available_seats) AS
total_tickets_sold
FROM event e
GROUP BY e.event_type;
```

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

```
SELECT YEAR(e.event_date) AS event_year, SUM((e.total_seats -
e.available_seats) * e.ticket_price) AS total_revenue
FROM event e
GROUP BY event_year;
```

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

```
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 ;
```

#now I vl display the records that have events_booked>1

```
 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;
```

-- having events_booked>1;

```
use ticketbooking_feb_hex_24;


-- step 1: Join and bring the tables togather.
select *
from  event e  JOIN booking b  ON e.id = b.event_id JOIN  customer c ON c.id = b.customer_id;

-- step 2: group by customer name as we need to compute revenue for each customer which will
-- give customer_name and number of bookings

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;

-- Step 3: We need to calculate sum of total couse for each customer, so updating above query
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;
```

#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 booking b
JOIN customer c ON b.customer_id = c.id
GROUP BY c.customer_name;
```

#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(e.ticket_price) AS
average_ticket_price
FROM event e
JOIN venue v ON e.venue_id = v.id
GROUP BY v.venue_name, 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.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;
```

-- now() gives todays date

#task 4

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

```
SELECT v.venue_name, AVG(e.ticket_price) AS average_ticket_price
FROM venue v
JOIN event e ON v.id = e.venue_id
GROUP BY v.venue_name;
```

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

```
SELECT event_name
FROM event
WHERE (total_seats - available_seats) > (0.5 * total_seats);
```

#3. Calculate the Total Number of Tickets Sold for Each Event:

```
SELECT event_name, (total_seats - available_seats) AS total_tickets_sold
FROM event;
```

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

```
SELECT customer_name
```

```
FROM customer c
WHERE NOT EXISTS (
   SELECT 1
   FROM booking b
   WHERE 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 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(total_tickets_sold) AS total_tickets_sold
FROM (
   SELECT event_type, (total_seats - available_seats) AS total_tickets_sold
   FROM event
) AS ticket_counts
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 c.customer_name, (
   SELECT SUM(total_cost)
   FROM booking b
   WHERE b.customer_id = c.id
) AS total_revenue
FROM customer c;
```

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

```
SELECT customer_name
FROM customer c
WHERE EXISTS (
   SELECT 1
```

```
    FROM booking b
    JOIN event e ON b.event_id = e.id
    WHERE b.customer_id = c.id AND e.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(total_tickets_sold) AS total_tickets_sold
FROM (
    SELECT event_type, (total_seats - available_seats) AS total_tickets_sold
    FROM event
) AS ticket_counts
GROUP BY event_type;
```

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

```
SELECT customer_name, MONTH(booking_date) AS booking_month
FROM booking b
JOIN customer c ON b.customer_id = c.id;
```

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

```
SELECT v.venue_name, (
    SELECT AVG(ticket_price)
```

```
    FROM event e
    WHERE e.venue_id = v.id
) AS average_ticket_price
FROM venue v;
```