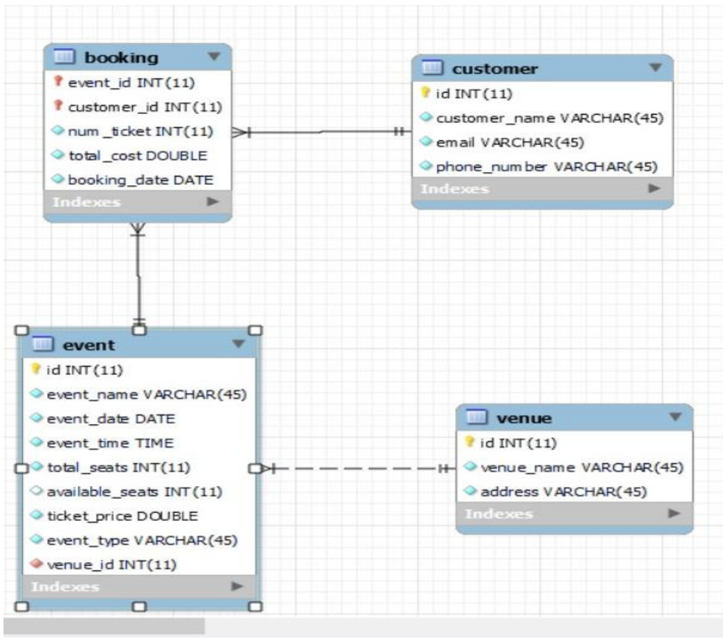
Ticket Booking System



```
#ticket booking Case study

use ticket;

#insertions

show tables;

describe venue;

describe customer;

describe event;

describe booking;

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,even
t_type,venue_id)
values
('Late Ms. Lata Mangeshkar Musical', '2021-09-12','20:00',320,270,600,'concert',6),
('CSK vs RCB', '2024-04-11','19:30',23000,3,3600,'sports',5),
('CSK vs RR', '2024-04-19','19:30',23000,10,3400,'sports',5),
('MI vs KKR', '2024-05-01','15:30',28000,100,8000,'sports',4);

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;

-- 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=7;

-- 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 500 and 2500;

-- 6. Write a SQL query to retrieve events with dates falling within a specific range
select *
```

```sql
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_type like '%concert%';


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

select *

from customer

limit 5,5;


/*

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
e.event_name,e.event_date,event_time,total_seats,available_seats,ticket_price,event_type

from event e,booking b

where e.id=b.event_id and num_ticket>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%';

#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 event details that have booking num_tickets > 1000


select b.event_id,b.num_ticket

from event e , booking b

where e.id = b.event_id AND b.num_ticket > 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';


-- Task 3


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

SELECT v.venue_name, AVG(b.ticketprice) AS AverageTicketPrice

FROM venue v

JOIN booking b ON v.venue_id = b.venue_id
```

```
GROUP BY v.venue_name;


-- 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;


/*

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;


/*

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

*/

select event_name

from event

where total_seats=available_seats;


/*

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_ticket) 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 average Ticket Price for Events in Each Venue.

select venue_name,avg(ticket_price) as average_ticket_price

from venue v,event e

where v.id=e.venue_id

group by v.id;

/*

-- 8. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Typ
```

```sql
*/
select event_type,sum(total_seats-available_seats) as tickets_sold
from event
group by event_type;
/*
-- 9. Write a SQL query to list customer who have booked tickets for multiple events.
*/

#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_ticket
from event e,customer c, booking b
where e.id = b.event_id AND
b.customer_id = c.id;

# step 2: I vl group by customer_name to get info of number_of events booked.

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;

# JOIN Queries

/*

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

*/

use ticket;

-- 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;



-- 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_ticket) 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


/*

Q. Names of Customers who have visited venue 'chennai' using all three
techniques(Nested Query).

*/


select id,customer_name

from customer

where id IN (select customer_id

                        from booking

                        where event_id IN (select id
                                                        from event
              where venue_id IN (select id

from venue
                       where venue_name='chennai')));

/*
+----+------------------+
| id | customer_name    |
+----+------------------+
|  1 | harry potter     |
|  3 | hermione granger |
|  5 | ginni weasley    |
+----+------------------+
*/

-- Task 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 event_name
from event
where ticket_price >  (select avg(ticket_price) from event);


 /*
 4. Find Customers Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.
 */


 insert into customer(customer_name,email,phone_number)
 values ('severus snape', 'sev@gmail.com','56556');


 select * from customer;


 -- SELECT column1 FROM t1 WHERE EXISTS (TABLE t2);


# if there is even 1 row in table t2 then the where clause condition is evaluated to true.
```

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

select * from event

where id NOT IN (select distinct event_id

from booking);


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

select id,event_name

from event where ticket_price > (select avg(ticket_price)

                                                                        from event);


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

select *

from customer

where id in(select customer_id

                          from booking

       where event_id in(select id

                                                                from event

                 where venue_id in (select id


from venue

                             where venue_name='chennai')));


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

select event_type, sum(b.num_tickets)as total_tickets_booked

from event e,booking b

where b.event_id=e.id
```

```
group by event_type;

-- 9. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery
select id,venue_name,(select avg(ticket_price)
from event
where venue.id=event.venue_id) as Avg_ticket_price from venue;




-- Notes & Wxtras


 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;




/*
Display customer details having email 'harry@gmail.com' provided this customer
has attended atleast 1 event.
*/

select *
```

```
from customer
where EXISTS (select distinct c.id
                                from customer c join booking b ON c.id=b.customer_id
        where c.email='harry@gmail.com')
AND email='harry@gmail.com';

select *
from customer
where EXISTS (select distinct c.id
                                from customer c join booking b ON c.id=b.customer_id
        where c.email='sev@gmail.com')
AND email='sev@gmail.com';

-- all customers for each event

select e.event_name,count(e.id)
from event e join booking b on e.id=b.event_id
join customer c on c.id=b.customer_id
group by e.event_name;
```