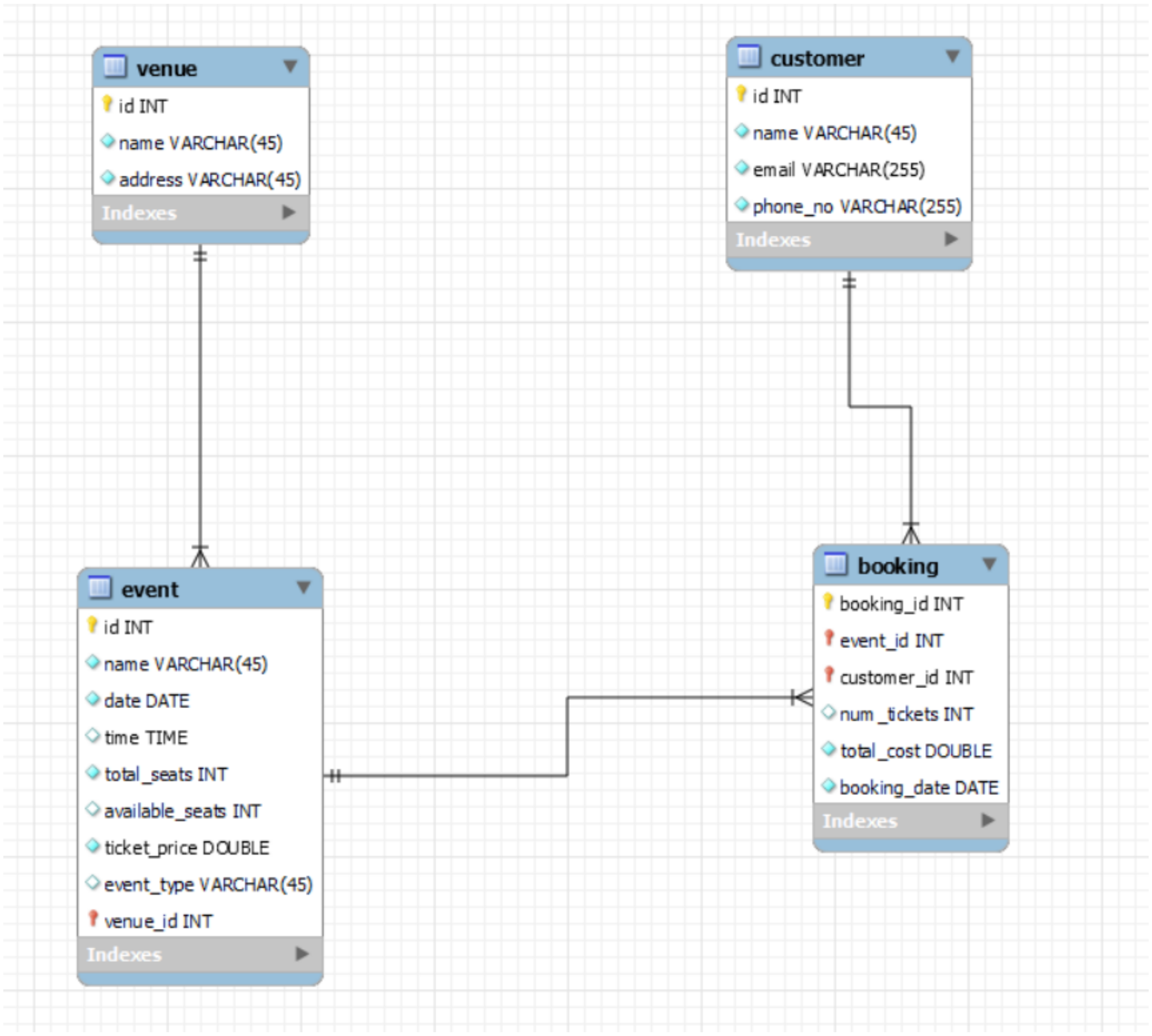
BOOKING TICKETS

ER DIGRAM:



<u>TASK - 1</u>

```
CREATE SCHEMA IF NOT EXISTS `booking` DEFAULT CHARACTER SET utf8 ;

USE `booking` ;

-- Table `booking`.`venue`

CREATE TABLE IF NOT EXISTS `booking`.`venue` (

  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `address` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
 ENGINE = InnoDB;
-- Table `booking`.`event`
CREATE TABLE IF NOT EXISTS `booking`.`event` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `date` DATE NOT NULL,
  `time` TIME NULL,
  `total_seats` INT NOT NULL,
  `available_seats` INT NULL,
  `ticket_price` DOUBLE NOT NULL,
  `event_type` VARCHAR(45) NULL,
  `venue_id` INT NOT NULL,
  PRIMARY KEY (`id`, `venue_id`),
  INDEX `fk_event_venue_idx` (`venue_id` ASC) ,
  CONSTRAINT `fk_event_venue`
    FOREIGN KEY (`venue_id`)
    REFERENCES `booking`.`venue` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- Table `booking`.`customer`
CREATE TABLE IF NOT EXISTS `booking`.`customer` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `email` VARCHAR(255) NOT NULL,
  `phone_no` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `booking`.`booking`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `booking`.`booking` (
  `booking_id` INT NOT NULL AUTO_INCREMENT,
  `event_id` INT NOT NULL,
  `customer_id` INT NOT NULL,
  `num_tickets` INT NULL,
  `total_cost` DOUBLE NOT NULL,
  `booking_date` DATE NOT NULL,
  PRIMARY KEY (`booking_id`, `event_id`, `customer_id`),
  INDEX `fk_event_has_customer_customer1_idx` (`customer_id` ASC) ,
  INDEX `fk_event_has_customer_event1_idx` (`event_id` ASC),
  CONSTRAINT `fk_event_has_customer_event1`
    FOREIGN KEY (`event_id`)
    REFERENCES `booking`.`event` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_event_has_customer_customer1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `booking`.`customer` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

use mydb;
show databases;
show tables;

desc booking;

insert into venue(name,address) values

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

('chennai', 'IT Park'),

('pondicherry ', 'state beach');

select * from venue;

insert into customer(name,email,phone_no)

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(name,date,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(event_id,customer_id,num_tickets,total_cost,booking_date) values

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

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

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

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

(2,5,10,34000,'2024-04-15'),

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

select * from booking;

TASK - 2

#1.TO SELECT EVENTS WITH AVAILABLE TICKETS

select name from event where available_seats>0;

#2.SELECT NAMES WITH TICKET PRICE RANGE FROM 3000 TO 3500

select name from event where ticket_price between 3000 and 3500;

#3.RETRIEVE EVENTS WITH DATE FALLING FROM SPECIFIC RANGE

select name as event_name from event where date between '2024-04-11' and '2024-05-01';-- ALIAS NAME

#4.RETRIEVE USERS IN BATCH OF 5,STARTING FROM 6TH USED

/*

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

*/

select * from customer limit 3,1;-- ONLY 1 RECORD WILL BE DISPLAYED

#5.SEAT CAPACITY MORE THAN 15000

select event_name from event where total_seats>15000;

#6.EVENTS EVENT NAMES THAT DOESN'T START WITH C,S

select name from event where name not like '[cs]%'; -- WHEN MULTIPLE VALUE NEED TO BE GIVEN INSIDE LIKE THEN USE '[--]%'

LEVEL 2: MULTI TABLE QUERY USING MANUAL MAPPING TECHNIQUE

#1. DISPLAY LIST OF EVENTS THAT ARE HOSTED IN VENUE CHENNAI

```
select e.name

from event e,venue v

where v.id=e.venue_id and v.name='chennai';
```

#2.SELECT CUSTOMERS THAT HAVE BOOKED TICKETS FOR EVENT 'CSK VS RCB' GAME WITH ID=2

```
select c.name

from booking b,customer c

where b.customer_id=c.id and b.event_id=2;
```

#3. DISPLAY EVENT DETAILS THAT ARE BOOKED NO_OF_TICKETS MORE THAN 100

```
select e.name

from event e,booking b

where e.id=b.event_id and num_tickets>5 ;
```

<u>TASK - 3</u>

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

```
select v.name

from venue v,event e,booking b,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';
```

#2,9 Write a SQL query to calculate the average Ticket Price for Events in Each Venue.

```
select e.name,avg(ticket_price)

from venue v,event e
```

```
where v.id=e.venue_id
group by e.id;
```

#3.Write a SQL query to Calculate the Total Revenue Generated by Events.

```
select name,(total_seats-available_seats)*ticket_price
from event
group by id;
```

#4.Write a SQL query to find the event with the highest ticket sales

```
select name,((total_seats-available_seats)*ticket_price) as highest_ticket_sales
from event
group by id
order by highest_ticket_sales desc
limit 0,1;
```

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

```
select name,(total_seats-available_seats) as ticket_sold
from event;
```

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

```
select name
from event
where available_seats=total_seats;
```

#7.Write a SQL query to Find the User Who Has Booked the Most Tickets.

```
select c.name,c.id,sum(num_tickets) as ticket_booked
from booking b,customer c
where b.customer_id=c.id
```

```
group by c.id
order by ticket_booked desc
limit 0,1;
```

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

```
select sum(total_seats-available_seats)
from event
group by event_type;
```

#9.Write a SQL query to list users who have booked tickets for multiple events.

```
select c.name as customer_name,count(c.name) as booked_tickets
from event e,booking b,customer c
where e.id=b.event_id and c.id=b.customer_id
group by customer_name
having booked_tickets>1
;



/*show databases;
use mydb;
show tables;
select * from booking;
select * from event;
select * from venue;
select * from customer; */
```

JOINS in TASK 3

/*

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

Sample O/P:

```
Customer          Revenue
Harry Potter   23000
Ronald Weasley 4500
```

*/

```
select c.name,sum(total_cost) as revenue
from booking b join customer c
on b.customer_id=c.id
group by c.name
order by revenue desc;
```

#12. Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.

```
select v.name as venue_name,e.name as event_name,ticket_price
from venue v join event e
on v.id=e.venue_id;
```

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

```
 select c.name,sum(b.num_tickets) as no_of_tickets
 from customer c join booking b
 on c.id=b.customer_id
 where b.booking_date>=date_sub('2024-05-15',interval 30 day)
 group by c.name
```

```
order by no_of_tickets;


-- alternative

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;
```

TASK - 4

#1.display all events hosted by venue 'chennai'

```
-- Manual Mapping

select e.id,e.name,v.name

from event e,venue v

where v.id=e.venue_id and v.name='chennai';


-- joins

select e.id,e.name,v.name

from event e join venue v

on v.id=e.venue_id

where v.name='chennai';


-- Nested Query

select id,name

from event where venue_id in(select id from venue where name='chennai');
```

#2.names of customers who have visted venue 'chennai'

```
-- Manual Mapping

select c.name,v.name
```

```
from venue v,customer c,event e,booking b

where v.id=e.venue_id and e.id=b.event_id and c.id=b.customer_id and v.name='chennai';


-- joins

select c.name,v.name

from venue v join event e on v.id=e.venue_id join

booking b on e.id=b.event_id join customer c on c.id=b.customer_id

where v.name='chennai';


-- Nested Query

select id,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 name='chennai')));
```

#3.display list of events that has sold num_tickets>500 and event_type='sports'

```
select name from event where event_type='sports' and id in(select event_id from booking
where num_tickets>2);
```

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

```
select venue_id,avg(ticket_price)

from event

where venue_id in(select id from venue)

group by venue_id;
```

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

```
select name

from event

where id in(select id from event where (total_seats-available_seats) > (total_seats/2) );
```

#6.Find Events having ticket price more than average ticket price of all events

```
select name from event

where ticket_price>(select avg(ticket_price) from event);
```

#7.Find Customers Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.

```
select name from customer where not exists(

select distinct c.name from customer c join booking b on b.customer_id=c.id);
```

#8.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

) and email='harry@gmail.com';

select * from event;

select * from venue;

select * from customer;

select * from booking;
```