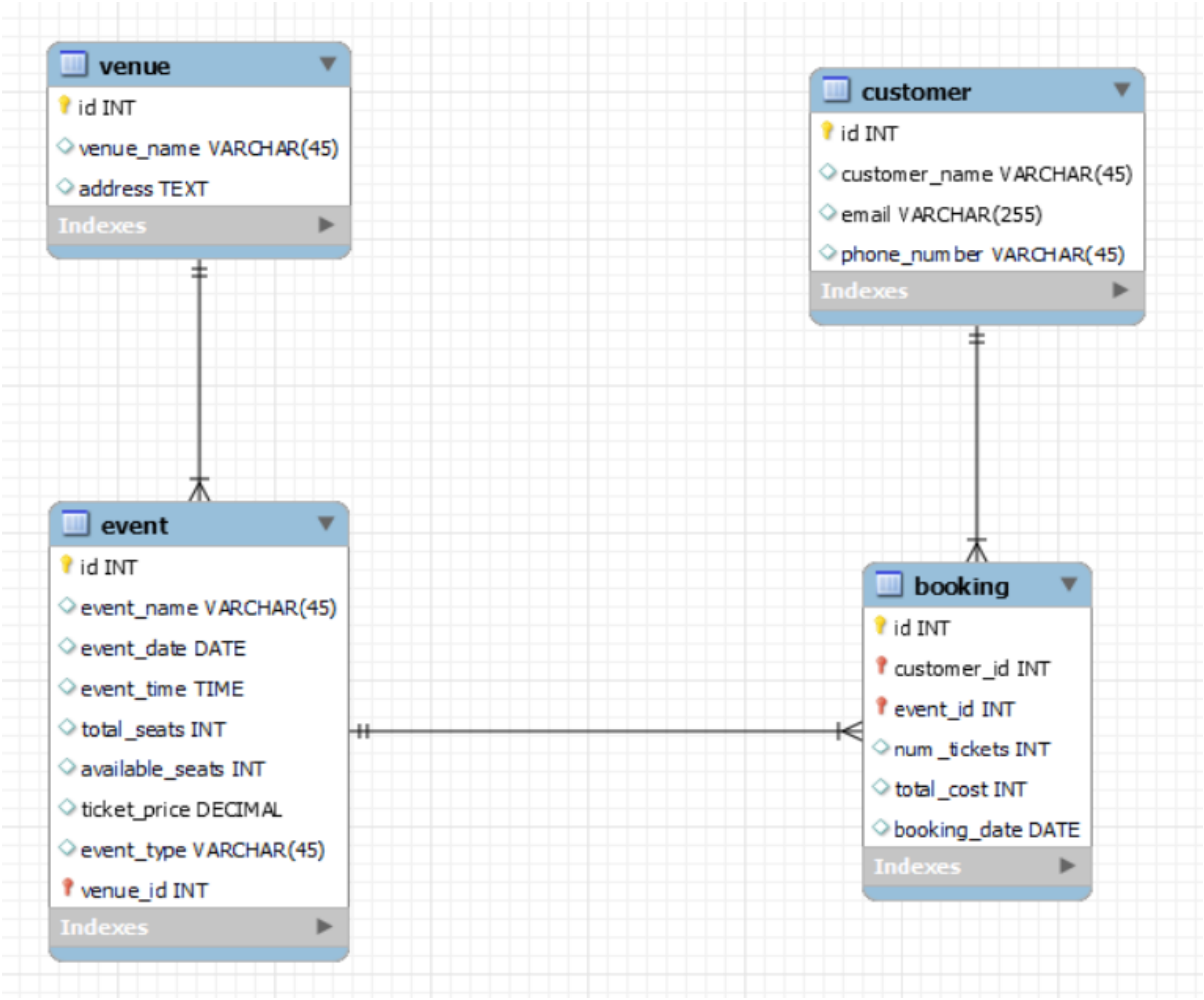
**ASSIGNMENT NO : 5**

**Ticket Booking System**

**ER Diagram:**




**Task:1. Database Design:**

-- MySQL Workbench Forward Engineering

-- -----------------------------------------------------

-- Schema ticketbookingsystem

-- -----------------------------------------------------

```
-- -----------------------------------------------------
-- Schema ticketbookingsystem
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `ticketbookingsystem` DEFAULT CHARACTER SET utf8 ;
USE `ticketbookingsystem` ;

-- -----------------------------------------------------
-- Table `ticketbookingsystem`.`venue`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ticketbookingsystem`.`venue` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `venue_name` VARCHAR(45) NULL,
  `address` TEXT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `ticketbookingsystem`.`event`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ticketbookingsystem`.`event` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `event_name` VARCHAR(45) NULL,
  `event_date` DATE NULL,
  `event_time` TIME NULL,
  `total_seats` INT NULL,
  `available_seats` INT NULL,
  `ticket_price` DECIMAL NULL,
  `event_type` VARCHAR(45) NULL,
  `venue_id` INT NOT NULL,
```

```
  PRIMARY KEY (`id`, `venue_id`),
  INDEX `fk_event_venue1_idx` (`venue_id` ASC) ,
  CONSTRAINT `fk_event_venue1`
    FOREIGN KEY (`venue_id`)
    REFERENCES `ticketbookingsystem`.`venue` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `ticketbookingsystem`.`customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ticketbookingsystem`.`customer` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `customer_name` VARCHAR(45) NULL,
  `email` VARCHAR(255) NULL,
  `phone_number` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `ticketbookingsystem`.`booking`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ticketbookingsystem`.`booking` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `customer_id` INT NOT NULL,
  `event_id` INT NOT NULL,
  `num_tickets` INT NULL,
  `total_cost` INT NULL,
  `booking_date` DATE NULL,
```

```
  PRIMARY KEY (`id`, `customer_id`, `event_id`),
  INDEX `fk_customer_has_event_event1_idx` (`event_id` ASC) ,
  INDEX `fk_customer_has_event_customer_idx` (`customer_id` ASC) ,
  CONSTRAINT `fk_customer_has_event_customer`
    FOREIGN KEY (`customer_id`)
    REFERENCES `ticketbookingsystem`.`customer` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_customer_has_event_event1`
    FOREIGN KEY (`event_id`)
    REFERENCES `ticketbookingsystem`.`event` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

use ticketbookingsystem;
```

**INSERTION:**

**-- venue table**

```
insert into venue(venue_name,address)values
('mumbai', 'marol andheri(w)'),
('chennai', 'IT Park'),
('pondicherry ', 'rock beach'),
('delhi','badharpur'),
('pondicherry','white town'),
('chennai','siruseri'),
('mumbai','andheri'),
('chennai','Tech park'),
(' chennai','warehouse'),
('pondicherry','mudayanpet');
```

```
mysql> select*from venue;
+----+--------------+------------------+
| id | venue_name   | address          |
+----+--------------+------------------+
|  1 | mumbai       | marol andheri(w) |
|  2 | chennai      | IT Park          |
|  3 | pondicherry  | rock beach       |
|  4 | delhi        | badharpur        |
|  5 | pondicherry  | white town       |
|  6 | chennai      | siruseri         |
|  7 | mumbai       | andheri          |
|  8 | chennai      | Tech park        |
|  9 |  chennai     | warehouse        |
| 10 | pondicherry  | mudayanpet       |
+----+--------------+------------------+
```

**-- event table**

insert into event(event_name,event_date,event_time,total_seats,available_seats,

ticket_price,event_type,venue_id)values

('technokings2k24 concert','2024-05-07','09.00',800,390,450,'concert',2),

('despotix23', '2023-11-09','10:00',920,270,900,'sports',4),

('mystifest2k23', '2023-10-19','14:00',500,100,200,'sports',9),

('tool tribute', '2024-04-18','18:00',1000,600,1600,'concert',7),

('pen show', '2021-03-03','11:00',600,370,450,'movie',6),

('conference cup', '2023-12-12','19:00',16000,100,200,'movie',8),

('Late Ms. Lata Mangeshkar concert', '2021-09-12','20:00',320,270,600,'concert',8),

('CSK vs RCB', '2024-04-11','19:30',23000,3,3600,'sports',1),

('CSK vs RR', '2024-05-19','19:30',20000,10,3400,'sports',10),

('MI vs KKR', '2024-05-01','15:30',28000,100,8000,'sports',1);

```
mysql> select*from event;
+----+---------------------------------+------------+------------+-------------+-----------------+--------------+------------+----------+
| id | event_name                      | event_date | event_time | total_seats | available_seats | ticket_price | event_type | venue_id |
+----+---------------------------------+------------+------------+-------------+-----------------+--------------+------------+----------+
|  1 | technokings2k24 concert         | 2024-05-07 | 00:00:09   |         800 |             390 |          450 | concert    |        2 |
|  2 | despotix23                      | 2023-11-09 | 10:00:00   |         920 |             270 |          900 | sports     |        4 |
|  3 | mystifest2k23                   | 2023-10-19 | 14:00:00   |         500 |             100 |          200 | sports     |        9 |
|  4 | tool tribute                    | 2024-04-18 | 18:00:00   |        1000 |             600 |         1600 | concert    |        7 |
|  5 | pen show                        | 2021-03-03 | 11:00:00   |         600 |             370 |          450 | movie      |        6 |
|  6 | conference cup                  | 2023-12-12 | 19:00:00   |       16000 |             100 |          200 | movie      |        8 |
|  7 | Late Ms. Lata Mangeshkar concert | 2021-09-12 | 20:00:00   |         320 |             270 |          600 | concert    |        8 |
|  8 | CSK vs RCB                      | 2024-04-11 | 19:30:00   |       23000 |               3 |         3600 | sports     |        1 |
|  9 | CSK vs RR                       | 2024-05-19 | 19:30:00   |       20000 |              10 |         3400 | sports     |       10 |
| 10 | MI vs KKR                       | 2024-05-01 | 15:30:00   |       28000 |             100 |         8000 | sports     |        1 |
+----+---------------------------------+------------+------------+-------------+-----------------+--------------+------------+----------+
```

**-- customer table**

insert into customer(customer_name,email,phone_number)values

('Ram Prasad','ram@gmail.com','9024554745'),

('Sandiya Vishwanath','sandiya@gmail.com','9174543526'),

('Jayanthi Selvam','selvam@gmail.com','9082707000'),

('Swetha Seetharaman','swetha@gmail.com','7098645321'),

('Divya Dharshini','divya@gmail.com','9123765480'),

('Nisha Vaithiyanathan','nisha@gmail.com','9865432000'),

('Darshini Balamurali','darshnini@gmail.com','709834521'),

('Agalya Shanmugam','agalya@gmail.com','8143256790'),

('Harini Murugavel','harini@gmail.com','9024554000'),

('Selva Ramaiah','selva@gmail.com','9156473420');

```
mysql> select* from customer;
+----+---------------------+---------------------+--------------+
| id | customer_name       | email               | phone_number |
+----+---------------------+---------------------+--------------+
|  1 | Ram Prasad          | ram@gmail.com       | 9024554745   |
|  2 | Sandiya Vishwanath  | sandiya@gmail.com   | 9174543526   |
|  3 | Jayanthi Selvam     | selvam@gmail.com    | 9082707000   |
|  4 | Swetha Seetharaman  | swetha@gmail.com    | 7098645321   |
|  5 | Divya Dharshini     | divya@gmail.com     | 9123765480   |
|  6 | Nisha Vaithiyanathan| nisha@gmail.com     | 9865432000   |
|  7 | Darshini Balamurali | darshnini@gmail.com | 709834521    |
|  8 | Agalya Shanmugam    | agalya@gmail.com    | 8143256790   |
|  9 | Harini Murugavel    | harini@gmail.com    | 9024554000   |
| 10 | Selva Ramaiah       | selva@gmail.com     | 9156473420   |
+----+---------------------+---------------------+--------------+
```

**-- booking table**

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

(1,1,2,900,'2024-05-02'),

(3,7,6,4200,'2021-09-05'),

(8,8,5,18000,'2024-04-09'),

(9,7,4,2400,'2021-09-03'),

(2,9,3,10200,'2024-05-15'),

(4,4,5,8000,'2024-04-17'),

(2,5,2,2900,'2021-03-01'),

(5,1,5,4000,'2024-03-06'),

(10,10,2,16000,'2024-04-20'),

(8,4,3,4800,'2024-04-15');

```
mysql> select*from booking;
+----+-------------+----------+-------------+------------+--------------+
| id | customer_id | event_id | num_tickets | total_cost | booking_date |
+----+-------------+----------+-------------+------------+--------------+
|  1 |           1 |        1 |           2 |        900 | 2024-05-02   |
|  2 |           3 |        7 |           6 |       4200 | 2021-09-05   |
|  3 |           8 |        8 |           5 |      18000 | 2024-04-09   |
|  4 |           9 |        7 |           4 |       2400 | 2021-09-03   |
|  5 |           2 |        9 |           3 |      10200 | 2024-05-15   |
|  6 |           4 |        4 |           5 |       8000 | 2024-04-17   |
|  7 |           2 |        5 |           2 |       2900 | 2021-03-01   |
|  8 |           5 |        1 |           5 |       4000 | 2024-03-06   |
|  9 |          10 |       10 |           2 |      16000 | 2024-04-20   |
| 10 |           8 |        4 |           3 |       4800 | 2024-04-15   |
+----+-------------+----------+-------------+------------+--------------+
```

**-- Tasks 2: Select, Where, Between, AND, LIKE:**

**-- 1. Write a SQL query to insert at least 10 sample records into each table.**

-- => inserted

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

```
select id,event_name from event;
```

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

```
select id,event_name
from event
where available_seats>0;
```

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

```
select event_name
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 event_name,ticket_price
from event
where ticket_price between 1000 and 2500;
```

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

```
select * from event
where event_date between '2024-05-01' and '2024-05-31';
```

**/* 7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their name. */**

```
select * from event
where event_type="concert" and event_name LIKE '%concert%';
```

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

```
select * from customer limit 5,6;
```

**/* 9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4. */**

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

**/* 10. Write a SQL query to retrieve customer information whose phone number end with '000' */**

```
select * from customer
where phone_number like '%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;
```

**-- 12. 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%';
```

**-- Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:**

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

```
select event_name,event_date,avg(ticket_price) as avg_price
from event
group by id;
```

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

```
select SUM((total_seats  - available_seats) *  ticket_price) as total_revene
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. Write a SQL query to Find Events with No Ticket Sales.**

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

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

```
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 List Events and the total number of tickets sold for each month. */**

```
select event_name,month(b.booking_date)as months,sum(b.num_tickets)as ticket_sold
from event e join booking b on e.id=b.event_id
group by event_name,months;
```

**-- 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 avg_ticket
from event e,venue v
where v.id=e.venue_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(b.num_tickets)as tickets_sold
from event e,booking b
where e.id=b.event_id
group by event_type;
```

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

```
select year(event_date)as years,sum((total_seats-available_seats)*ticket_price)as
total_revenue
```

```
from event
group by years;
```

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

```
select c.customer_name , count(c.id) as events_booked
from customer c, booking b
where b.customer_id = c.id
group by c.customer_name
having events_booked>1;
```

**/* 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 revenue
from event e,booking b,customer c
where e.id=b.event_id
and c.id=b.customer_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 e.event_type,v.venue_name,avg(e.ticket_price) as avg_price
from event e,venue v
where v.id=e.venue_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 total_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;
```

**-- Tasks 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)
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,total_seats,available_seats
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 e.event_name,sum(b.num_tickets)as total_number
from booking b join event e on e.id=b.event_id
group by e.event_name;
```

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

```
select id,customer_name
from customer
where not exists(select customer_id from booking b
        where b.customer_id=customer.id);
```

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

```
select event_name
from event
```

```
where id not in(select 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(b.num_tickets)as total_sold
from event join booking b on event.id=b.event_id
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(b.total_cost)
        from booking b
        where c.id=b.customer_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
where id IN (
    select customer_id
    from booking
    where event_id IN (
        select id from event
```

```
                where venue_id=1));
```

**/* 10. Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY. */**

```
select event_type,(
                select sum(b.num_tickets)
                from booking b
                where b.event_id=e.id)as total_sold
from event e
group by event_type;
```

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

```
select c.customer_name,month(booking_date) as booking_month
from customer c JOIN booking b ON c.id = b.customer_id;
```

**-- 12. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery**

```
select v.venue_name,avg(e.ticket_price) as avg_price
from venue v,event e
where v.id=e.venue_id
group by v.venue_name;
```