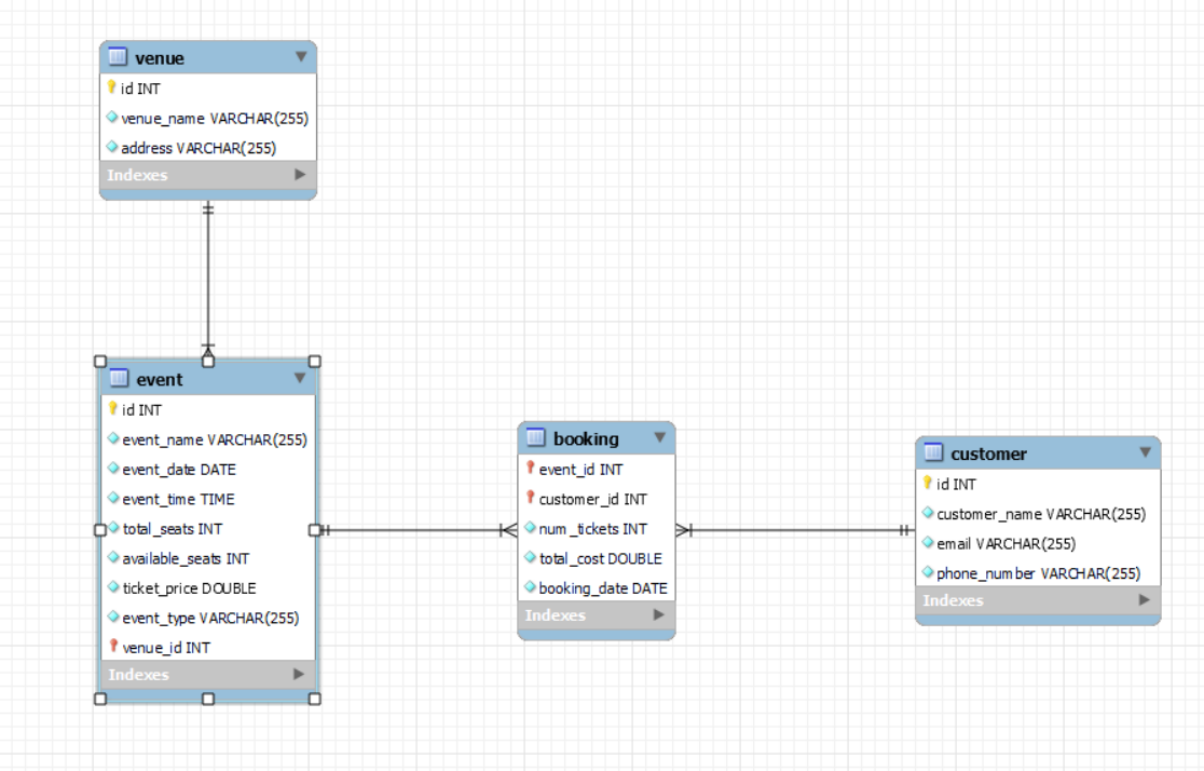
# ASSIGNMENT NO 5

# TICKET BOOKING

## ER DIAGRAM:




## Task 1: Database Design

-- MySQL Workbench Forward Engineering

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

-- Schema assignment_ticket_booking

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

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

-- Schema assignment_ticket_booking

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

```
CREATE SCHEMA IF NOT EXISTS `assignment_ticket_booking` DEFAULT
CHARACTER SET utf8 ;
USE `assignment_ticket_booking` ;


-- -----------------------------------------------------
-- Table `assignment_ticket_booking`.`venue`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_ticket_booking`.`venue` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `venue_name` VARCHAR(255) NOT NULL,
  `address` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `assignment_ticket_booking`.`event`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_ticket_booking`.`event` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `event_name` VARCHAR(255) NOT NULL,
  `event_date` DATE NOT NULL,
  `event_time` TIME NOT NULL,
  `total_seats` INT NOT NULL,
  `available_seats` INT NOT NULL,
  `ticket_price` DOUBLE NOT NULL,
  `event_type` VARCHAR(255) NOT 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 `assignment_ticket_booking`.`venue` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `assignment_ticket_booking`.`customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_ticket_booking`.`customer` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `customer_name` VARCHAR(255) NOT NULL,
  `email` VARCHAR(255) NOT NULL,
  `phone_number` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
COMMENT = '                                    ';


-- -----------------------------------------------------
-- Table `assignment_ticket_booking`.`booking`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_ticket_booking`.`booking` (
  `event_id` INT NOT NULL AUTO_INCREMENT,
  `customer_id` INT NOT NULL,
  `num_tickets` INT NOT NULL,
  `total_cost` DOUBLE NOT NULL,
  `booking_date` DATE NOT NULL,
  PRIMARY KEY (`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 `assignment_ticket_booking`.`event` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_event_has_customer_customer1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `assignment_ticket_booking`.`customer` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

## INSERTION

```
insert into customer(customer_name,email,phone_number)
values
('harry potter','harry@gmail.com','46654545'),
('ronald weasley','ron@gmail.com','43354545'),
('hermione granger','her@gmail.com','42254545'),
('draco malfoy','drac@gmail.com','49954545'),
('ginni weasley','ginni@gmail.com','47754545'),
('dhoni','d@gmail.com','45554545'),
('virat','v@gmail.com','45654545'),
('bumrah','b@gmail.com','45477545'),
('shami','s@gmail.com','45490545'),
('rohit','r@gmail.com','45422545');
```

```
mysql> select * from customer;
+----+------------------+-----------------+--------------+
| id | customer_name    | email           | phone_number |
+----+------------------+-----------------+--------------+
|  1 | harry potter     | harry@gmail.com | 46654545     |
|  2 | ronald weasley   | ron@gmail.com   | 43354545     |
|  3 | hermione granger | her@gmail.com   | 42254545     |
|  4 | draco malfoy     | drac@gmail.com  | 49954545     |
|  5 | ginni weasley    | ginni@gmail.com | 47754545     |
|  6 | dhoni            | d@gmail.com     | 45554545     |
|  7 | virat            | v@gmail.com     | 45654545     |
|  8 | bumrah           | b@gmail.com     | 45477545     |
|  9 | shami            | s@gmail.com     | 45490545     |
| 10 | rohit            | r@gmail.com     | 45422545     |
+----+------------------+-----------------+--------------+
10 rows in set (0.02 sec)
```

insert into venue(venue_name,address) values

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

('chennai', 'IT Park'),

('pondicherry ', 'state beach');

```
mysql> select * from venue;
+----+--------------+------------------+
| id | venue_name   | address          |
+----+--------------+------------------+
|  1 | mumbai       | marol andheri(w) |
|  2 | chennai      | IT Park          |
|  3 | pondicherry  | state beach      |
+----+--------------+------------------+
3 rows in set (0.00 sec)
```

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),

('Dettol Cup', '2024-05-01','15:30',5000,100,2400,'sports',1),

('Conference cup', '2024-05-03','15:30',16000,100,1200,'sports',1);

```
mysql> select * from event;
+----+---------------------------------+------------+------------+-------------+----------------+--------------+------------+----------+
| id | event_name                      | event_date | event_time | total_seats | available_seats | ticket_price | event_type | venue_id |
+----+---------------------------------+------------+------------+-------------+----------------+--------------+------------+----------+
|  1 | Late Ms. Lata Mangeshkar Musical | 2021-09-12 | 20:00:00   |         320 |            270 |          600 | concert    |        3 |
|  2 | CSK vs RCB                      | 2024-04-11 | 19:30:00   |       23000 |              3 |         3600 | sports     |        2 |
|  3 | CSK vs RR                       | 2024-04-19 | 19:30:00   |       23000 |             10 |         3400 | sports     |        2 |
|  4 | MI vs KKR                       | 2024-05-01 | 15:30:00   |       28000 |            100 |         8000 | sports     |        1 |
|  5 | Dettol Cup                      | 2024-05-01 | 15:30:00   |        5000 |            100 |         2400 | sports     |        1 |
|  6 | Conference cup                  | 2024-05-01 | 15:30:00   |       16000 |            100 |         1200 | concert    |        1 |
+----+---------------------------------+------------+------------+-------------+----------------+--------------+------------+----------+
6 rows in set (0.00 sec)
```

insert into booking values

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

(5,2,3,960,'2021-09-12'),

(4,3,3,10800,'2024-04-11'),

(2,3,5000,18000,'2024-04-11'),

(4,4,5,18000,'2024-05-03'),

(3,5,1000,34000,'2024-04-19'),

(3,6,4000,32000,'2024-05-01'),

(6,7,4,32000,'2024-05-01'),

(1,8,2,640,'2021-09-12'),

(2,9,3,960,'2021-09-12'),

(3,10,3,10800,'2024-05-01'),

(5,10,10,34000,'2024-05-03');

```
mysql> select * from booking;
+----------+-------------+-------------+------------+--------------+
| event_id | customer_id | num_tickets | total_cost | booking_date |
+----------+-------------+-------------+------------+--------------+
|        1 |           1 |           2 |        640 | 2021-09-12   |
|        1 |           8 |           2 |        640 | 2021-09-12   |
|        2 |           3 |        5000 |      18000 | 2024-04-11   |
|        2 |           9 |           3 |        960 | 2021-09-12   |
|        3 |           5 |        1000 |      34000 | 2024-04-19   |
|        3 |           6 |        4000 |      32000 | 2024-05-01   |
|        3 |          10 |           3 |      10800 | 2024-05-01   |
|        4 |           3 |           3 |      10800 | 2024-04-11   |
|        4 |           4 |           5 |      18000 | 2024-05-03   |
|        5 |           2 |           3 |        960 | 2021-09-12   |
|        5 |          10 |          10 |      34000 | 2024-05-03   |
|        6 |           7 |           4 |      32000 | 2024-05-01   |
+----------+-------------+-------------+------------+--------------+
12 rows in set (0.00 sec)
```

## Task 2 : Query

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

2. Write a SQL query to list all Events.

```
select event_name from event;
```

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

```
select 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 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 '2024-02-01' 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";
```

8. Write a SQL query to retrieve users in batches of 4, starting from the 6th user.

```
select * from customer limit 5,4;
```

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 ‘l’, ‘m’, ‘w’

select * from event where event_name not like 'l%' and event_name not like 'm%' and event_name

not like 'w%';

## Task 3

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

select v.venue_name,avg(e.ticket_price)

from venue v,event e

where e.venue_id=v.id

group by v.venue_name;

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

select event_name,sum((total_seats-available_seats)*ticket_price) as revenue

from event

group by event_name;

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

select event_name, total_seats-available_seats as total_tickets

from event

group by event_name

order by total_tickets 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

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 c.customer_name,sum(num_tickets) as ticket_count
from customer c, booking b
where c.id=b.customer_id
group by c.customer_name
order by ticket_count desc limit 0,1;
```

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, venue v
where v.id=e.venue_id
group by v.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)
from event
group by event_type;
```

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

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

12. Write a SQL query to calculate the Total Revenue Generated by Events from Each User.

```
select c.customer_name,e.event_name,b.total_cost
from event e,booking b,customer c
where e.id=b.event_id
and c.id=b.customer_id
group by e.event_name ,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)
from event e,venue v
where v.id=e.venue_id
group by event_type,venue_name;
-- joining the tables
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_booking
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 cost 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_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('2024-04-01',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;
```