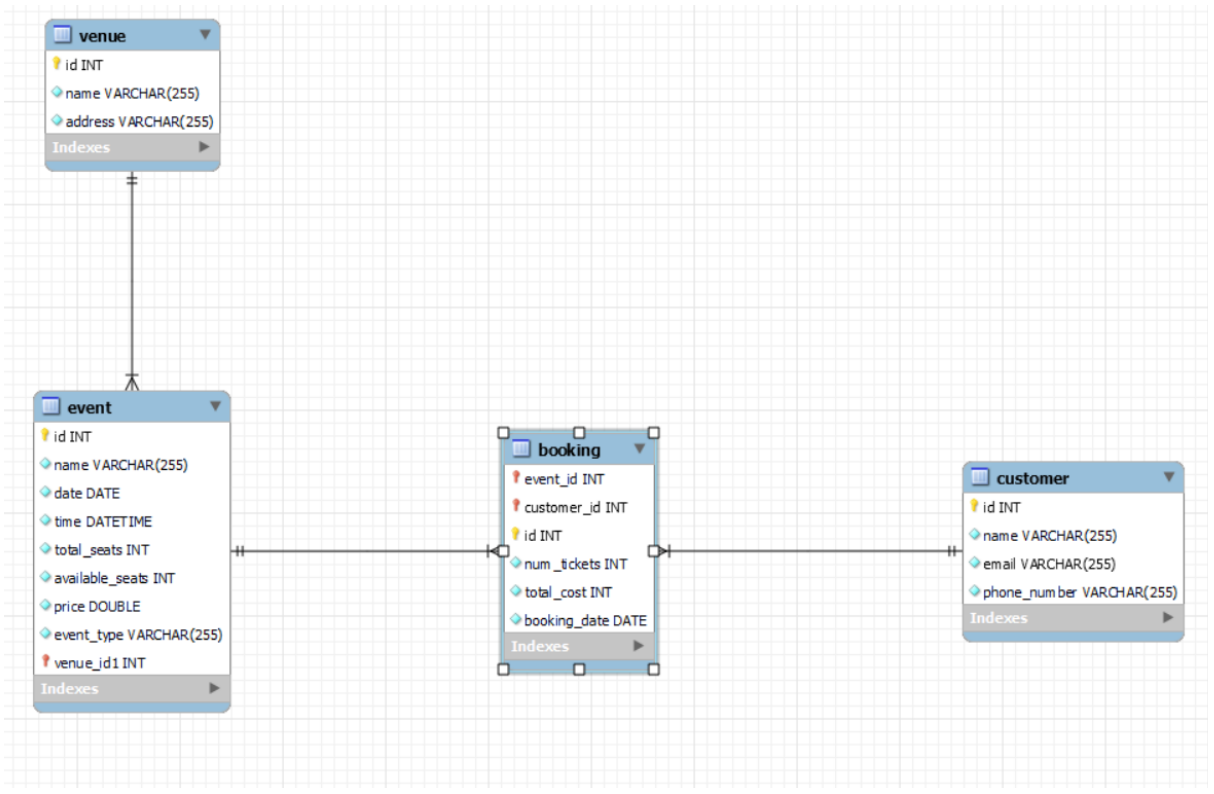
ASSIGNMENT NO : 1

Ticket Booking System

ER DIAGRAM:



Task:1. Database Design:

-- MySQL Workbench Forward Engineering

-- ---

-- Schema mydb

-- ---

-- ---

-- Schema mydb

-- ---

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

USE `mydb` ;

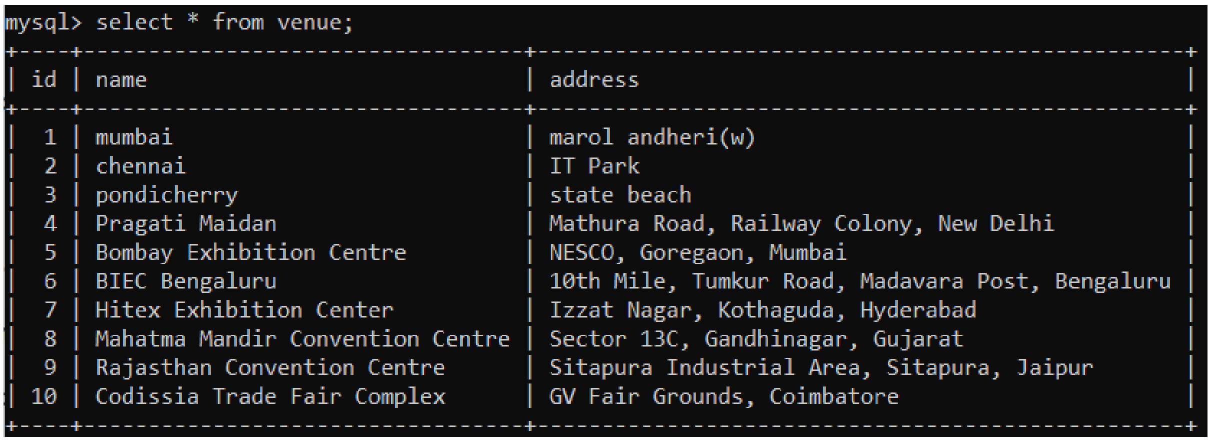
```
-- -----------------------------------------------------
-- Table `mydb`.`venue`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`venue` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `address` VARCHAR(255) NOT NULL,
  PRIMARY KEY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



-- -----------------------------------------------------
-- Table `mydb`.`event`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`event` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `date` DATE NOT NULL,
  `time` DATETIME NOT NULL,
  `total_seats` INT NOT NULL,
  `available_seats` INT NOT NULL,
  `price` DOUBLE NOT NULL,
  `event_type` VARCHAR(255) NOT NULL,
  `venue_id1` INT NOT NULL,
  PRIMARY KEY (`id`, `venue_id1`),
  INDEX `fk_event_venue_idx` (`venue_id1` ASC) ,
  CONSTRAINT `fk_event_venue`
    FOREIGN KEY (`venue_id1`)
    REFERENCES `mydb`.`venue` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;



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



-- -----------------------------------------------------
-- Table `mydb`.`booking`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`booking` (
  `event_id` INT NOT NULL AUTO_INCREMENT,
  `customer_id` INT NOT NULL,
  `id` INT NOT NULL,
  `num_tickets` INT NOT NULL,
  `total_cost` INT NOT NULL,
  `booking_date` DATE NOT NULL,
  INDEX `fk_event_has_customer_customer1_idx` (`customer_id` ASC) ,
  INDEX `fk_event_has_customer_event1_idx` (`event_id` ASC) ,
  PRIMARY KEY (`event_id`, `customer_id`, `id`),
  CONSTRAINT `fk_event_has_customer_event1`
    FOREIGN KEY (`event_id`)
    REFERENCES `mydb`.`event` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_event_has_customer_customer1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `mydb`.`customer` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

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

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

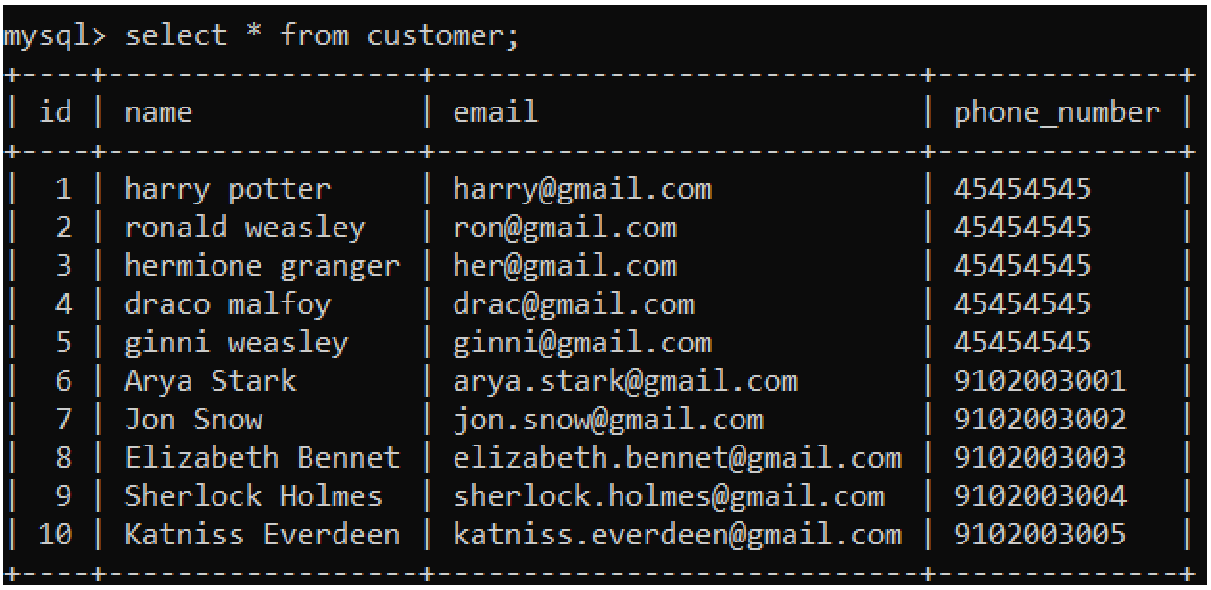
```
-- venue insertion
insert into venue(name,address)values
('mumbai','marol andheri(w)'),
('chennai','IT'Park'),
('pondicherry','state beach'),
('Pragati Maidan', 'Mathura Road, Railway Colony, New Delhi'),
('Bombay Exhibition Centre', 'NESCO, Goregaon, Mumbai'),
('BIEC Bengaluru', '10th Mile, Tumkur Road, Madavara Post, Bengaluru'),
('Hitex Exhibition Center', 'Izzat Nagar, Kothaguda, Hyderabad'),
('Mahatma Mandir Convention Centre', 'Sector 13C, Gandhinagar, Gujarat'),
('Rajasthan Convention Centre', 'Sitapura Industrial Area, Sitapura, Jaipur'),
('Codissia Trade Fair Complex', 'GV Fair Grounds, Coimbatore');
```



```
mysql> select * from venue;
+----+--------------------------------+---------------------------------------------+
| id | name                           | address                                     |
+----+--------------------------------+---------------------------------------------+
|  1 | mumbai                         | marol andheri(w)                            |
|  2 | chennai                        | IT Park                                     |
|  3 | pondicherry                    | state beach                                 |
|  4 | Pragati Maidan                 | Mathura Road, Railway Colony, New Delhi     |
|  5 | Bombay Exhibition Centre       | NESCO, Goregaon, Mumbai                     |
|  6 | BIEC Bengaluru                 | 10th Mile, Tumkur Road, Madavara Post, Bengaluru |
|  7 | Hitex Exhibition Center        | Izzat Nagar, Kothaguda, Hyderabad           |
|  8 | Mahatma Mandir Convention Centre | Sector 13C, Gandhinagar, Gujarat          |
|  9 | Rajasthan Convention Centre    | Sitapura Industrial Area, Sitapura, Jaipur  |
| 10 | Codissia Trade Fair Complex    | GV Fair Grounds, Coimbatore                 |
+----+--------------------------------+---------------------------------------------+
```

-- customer insertion

```
INSERT INTO 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','draco@gmail.com',45454545),
('ginni weasley', 'ginni@gmail.com', '45454545')
('Arya Stark', 'arya.stark@gmail.com', '9102003001'),
('Jon Snow', 'jon.snow@gmail.com', '9102003002'),
('Elizabeth Bennet', 'elizabeth.bennet@gmail.com', '9102003003'),
('Sherlock Holmes', 'sherlock.holmes@gmail.com', '9102003004'),
('Katniss Everdeen', 'katniss.everdeen@gmail.com', '9102003005');
```



```
mysql> select * from customer;
+----+------------------+----------------------------+--------------+
| id | name             | email                      | phone_number |
+----+------------------+----------------------------+--------------+
|  1 | harry potter     | harry@gmail.com            | 45454545     |
|  2 | ronald weasley   | ron@gmail.com              | 45454545     |
|  3 | hermione granger | her@gmail.com              | 45454545     |
|  4 | draco malfoy     | drac@gmail.com             | 45454545     |
|  5 | ginni weasley    | ginni@gmail.com            | 45454545     |
|  6 | Arya Stark       | arya.stark@gmail.com       | 9102003001   |
|  7 | Jon Snow         | jon.snow@gmail.com         | 9102003002   |
|  8 | Elizabeth Bennet | elizabeth.bennet@gmail.com | 9102003003   |
|  9 | Sherlock Holmes  | sherlock.holmes@gmail.com  | 9102003004   |
| 10 | Katniss Everdeen | katniss.everdeen@gmail.com | 9102003005   |
+----+------------------+----------------------------+--------------+
```

-- event insertion

insert into event(event_name,event_date,event_time,total_seats,available_seats,ticket_price,event_type, venue_id)

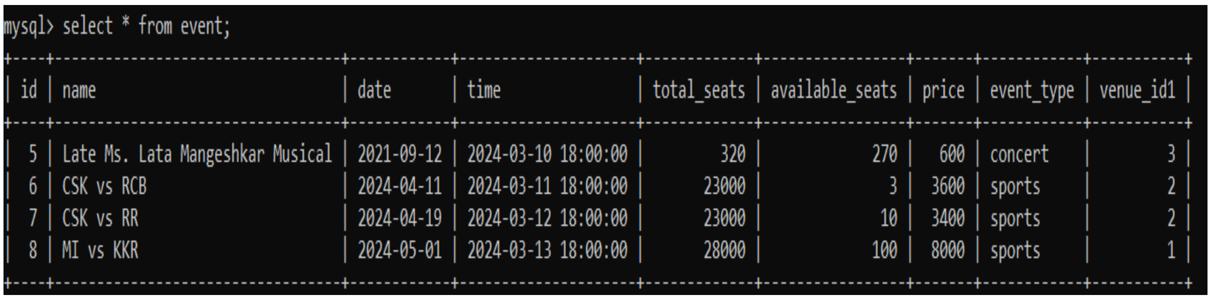
values

('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);



```
mysql> select * from event;
+----+--------------------------------+------------+---------------------+-------------+-----------------+-------+------------+-----------+
| id | name                           | date       | time                | total_seats | available_seats | price | event_type | venue_id1 |
+----+--------------------------------+------------+---------------------+-------------+-----------------+-------+------------+-----------+
|  5 | Late Ms. Lata Mangeshkar Musical | 2021-09-12 | 2024-03-10 18:00:00 |         320 |             270 |   600 | concert    |         3 |
|  6 | CSK vs RCB                     | 2024-04-11 | 2024-03-11 18:00:00 |       23000 |               3 |  3600 | sports     |         2 |
|  7 | CSK vs RR                      | 2024-04-19 | 2024-03-12 18:00:00 |       23000 |              10 |  3400 | sports     |         2 |
|  8 | MI vs KKR                      | 2024-05-01 | 2024-03-13 18:00:00 |       28000 |             100 |  8000 | sports     |         1 |
+----+--------------------------------+------------+---------------------+-------------+-----------------+-------+------------+-----------+
```

-- booking insertion

insert into booking values

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

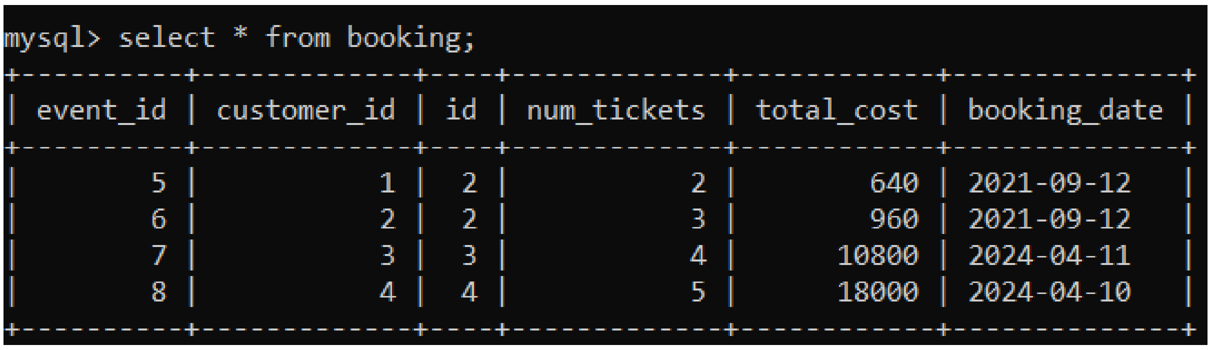
(4,4,3,960,'2021-09-12'),

(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');



```
mysql> select * from booking;
+----------+-------------+----+-------------+------------+--------------+
| event_id | customer_id | id | num_tickets | total_cost | booking_date |
+----------+-------------+----+-------------+------------+--------------+
|        5 |           1 |  2 |           2 |        640 | 2021-09-12   |
|        6 |           2 |  2 |           3 |        960 | 2021-09-12   |
|        7 |           3 |  3 |           4 |      10800 | 2024-04-11   |
|        8 |           4 |  4 |           5 |      18000 | 2024-04-10   |
+----------+-------------+----+-------------+------------+--------------+
```

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;
```

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

```
select * from event where name like '%cup%';
```

5. Write a SQL query to select events with ticket price range is between 1000 to 2500.

```
select * from event where 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 '2024-04-11' 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 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 offset 5;
```

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 order by total_seats ASC ;
```

12. Write a SQL query to select events name not start with ‘x’, ‘y’, ‘z’

```
select *  from event where name NOT LIKE 'c%' AND name NOT LIKE 'x%';
```

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

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

```
select e.id,v.name,AVG(e.price )  from event e, venue v where v.id = e.id group by
e.id;
```

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

```
select SUM((total_seats  - available_seats) *  price) from event;
```

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

```
select name,MAX((total_seats  - available_seats) *  price) as total_sales

from event group by 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 name, total_seats  - available_seats as total_tickets_sold from event
group by name;
```

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

```
select e.* from event e left join booking b on e.id = b.event_id where b.id is null;
```

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

```
select name, SUM(b.num_tickets) as tickets_booked  from booking b, customer c
where b.customer_id = c.id group by 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 e.id, e.name, year(b.booking_date) as year, month(b.booking_date) as month,
sum(b.num_tickets) as tickets_sold from event e join booking b on e.id = b.event_id
```

```sql
group by e.id, e.name, year(b.booking_date), month(b.booking_date) order by year, month;
```

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

```sql
select v.name as venue_name, avg(e.price) as average_ticket_price
from event e join venue v on e.venue_id1 = v.id group by v.name;
```

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

```sql
select e.event_type, sum(b.num_tickets) as total_tickets_sold from event e
join booking b on e.id = b.event_id group by e.event_type;
```

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

```sql
select year(b.booking_date) as year, sum(b.total_cost) as total_revenue from booking b
join event e on b.event_id = e.id group by year(b.booking_date);
```

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

```sql
select c.id, c.name, count(distinct b.event_id) as events_booked
from customer c join booking b on c.id = b.customer_id group by c.id, c.name having
count(distinct b.event_id) > 1;
```

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

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

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

```sql
select e.event_type, v.name as venue_name, avg(e.price) as average_ticket_price
from event e join venue v on e.venue_id1 = v.id group by e.event_type, v.name;
```

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

```
select c.id, c.name, sum(b.num_tickets) as total_tickets_purchased
from customer c join booking b on c.id = b.customer_id  where b.booking_date >= curdate()
 interval 30 day group by c.id, c.name;
```

Tasks 4: Subquery and its types

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

```
select v.name as VenueName, av.AveragePrice from venue v join (
select venue_id1, avg(price) as AveragePrice from event group by venue_id1) av on v.id =   a
v.venue_id1;
```

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

```
select e.id, e.name, e.total_seats,  (e.total_seats - e.available_seats) as tickets_sold,
((e.total_seats - e.available_seats) / e.total_seats) * 100 as percentage_sold from event e
where ((e.total_seats - e.available_seats) / e.total_seats) * 100 > 50;
```

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

```
select e.id, e.name, sum(b.num_tickets) as total_tickets_sold from event e
join booking b on e.id = b.event_id group by e.id, e.name;
```

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

```
select c.* 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 e.*from event e where e.id not in (select distinct b.event_id from booking b);
```

6. Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause.

```
select et.event_type, sum(et.tickets_sold) as total_tickets_sold from (select e.event_type,
b.num_tickets as tickets_sold from event e join booking b on e.id = b.event_id
) et group by et.event_type;
```

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

```
select * from event where price > ( select avg(price) from event);
```

8. Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery.

```
select c.id, c.name, (select sum(b.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 distinct c.* from customer c join booking b on c.id = b.customer_id
where b.event_id in ( select e.id  from event e where e.venue_id1 =1);
```

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

```
select e.event_type, sum(ts.total_tickets_sold) as total_tickets_sold_for_category
from (select event_id, sum(num_tickets) as total_tickets_sold from booking
group by event_id) ts join event e on ts.event_id = e.id group by e.event_type;
```

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

```
select c.id, c.name, DATE_FORMAT(b.booking_date, '%Y-%m') as booking_month
from customer c join booking b on c.id = b.customer_id group by c.id, c.name,
booking_month order by c.id, booking_month;
```

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

```
select v.id, v.name, avg(ep.average_price) as average_ticket_price
from venue v join ( select venue_id1, avg(price) as average_price from event
group by venue_id1) ep on v.id = ep.venue_id1 group by v.id, v.name;
```