Ticket Booking System

Task-1:

1) Database Design:

Create the database named "TicketBookingSystem"

```
mysql> create database TicketBookingSystem;
Query OK, 1 row affected (1.92 sec)
```

2)Creating venue table

```
mysql> create table venue(
    -> venue_id int primary key,
    -> venue_name varchar(30),
    -> address varchar(50)
    -> );
Query OK, 0 rows affected (4.46 sec)
```

```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| venue_id   | int         | NO   | PRI | NULL    |       |
| venue_name | varchar(30) | YES  |     | NULL    |       |
| address    | varchar(50) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.92 sec)
```

Creating event table

```
mysql> Create table event (
    ->     event_id int primary key,
    ->     event_name varchar(50),
    ->     event_date DATE,
    ->     event_time TIME,
    ->     venue_id int,
    ->     total_seats int,
    ->     available_seats int,
    ->     ticket_price decimal,
    ->     event_type varchar(20),
    ->     booking_id int,
    ->     Foreign key (venue_id) references venue(venue_id)
    ->
    -> );
Query OK, 0 rows affected (2.18 sec)
```

```
+-----------------+---------------+------+-----+---------+-------+
| Field           | Type          | Null | Key | Default | Extra |
+-----------------+---------------+------+-----+---------+-------+
| event_id        | int           | NO   | PRI | NULL    |       |
| event_name      | varchar(50)   | YES  |     | NULL    |       |
| event_date      | date          | YES  |     | NULL    |       |
| event_time      | time          | YES  |     | NULL    |       |
| venue_id        | int           | YES  | MUL | NULL    |       |
| total_seats     | int           | YES  |     | NULL    |       |
| available_seats | int           | YES  |     | NULL    |       |
| ticket_price    | decimal(10,0) | YES  |     | NULL    |       |
| event_type      | varchar(20)   | YES  |     | NULL    |       |
| booking_id      | int           | YES  |     | NULL    |       |
+-----------------+---------------+------+-----+---------+-------+
10 rows in set (0.08 sec)
```

Creating customer table

```
mysql> Create table customer (
    ->     customer_id int primary key,
    ->     customer_name varchar(50),
    ->     email varchar(50),
    ->     phone_number varchar(15),
    ->     booking_id int
    ->
    -> );
Query OK, 0 rows affected (1.29 sec)
```

```
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| customer_id   | int         | NO   | PRI | NULL    |       |
| customer_name | varchar(50) | YES  |     | NULL    |       |
| email         | varchar(50) | YES  |     | NULL    |       |
| phone_number  | varchar(15) | YES  |     | NULL    |       |
| booking_id    | int         | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

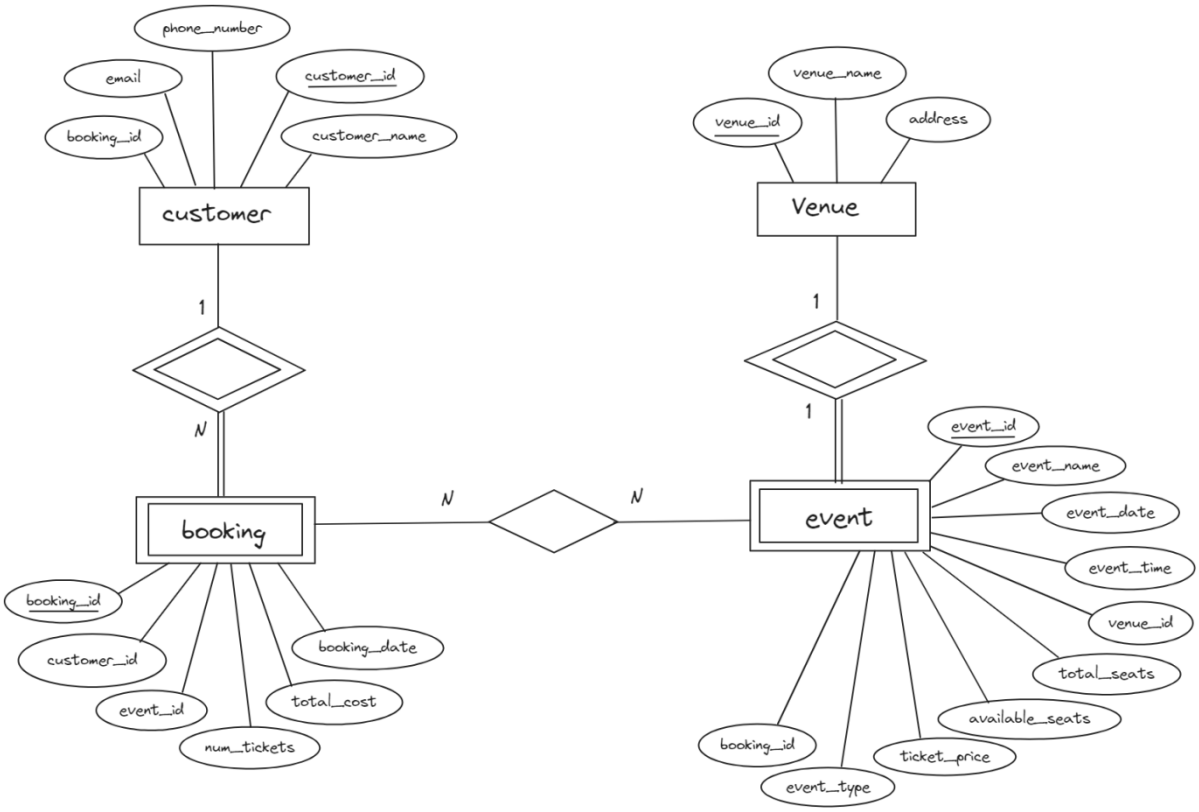
Creating booking table

```
mysql> Create table booking (
    ->     booking_id int primary key,
    ->     customer_id int,
    ->     event_id int,
    ->     num_tickets int,
    ->     total_cost int,
    ->     booking_date DATE,
    ->     Foreign key (customer_id) references customer(customer_id),
    ->     Foreign key (event_id) references event(event_id)
    -> );
Query OK, 0 rows affected (1.82 sec)
```

Field	Type	Null	Key	Default	Extra
booking_id	int	NO	PRI	NULL	
customer_id	int	YES	MUL	NULL	
event_id	int	YES	MUL	NULL	
num_tickets	int	YES		NULL	
total_cost	int	YES		NULL	
booking_date	date	YES		NULL	

6 rows in set (0.00 sec)

3) Entity Relationship Diagram



Task-2:

Inserting values into venue table

```
mysql> insert into venue values(101,'Auditorium','Dindigul'),
    -> (102,'raja mahal','Trichy'),
    -> (103,'rani palace','Coimbatore'),
    -> (104,'open auditorium','Chennai'),
    -> (105,'devi complex','Bangalore'),
    -> (106,'priya theatre','Erode'),
    -> (107,'kannan bazaar','Karur'),
    -> (108,'kanmani college','Madurai'),
    -> (109,'mathi theatre','Mumbai'),
    -> (110,'dharshini bazaar','Gujarat');
Query OK, 10 rows affected (0.55 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
+----------+------------------+------------+
| venue_id | venue_name       | address    |
+----------+------------------+------------+
|      101 | Auditorium       | Dindigul   |
|      102 | raja mahal       | Trichy     |
|      103 | rani palace      | Coimbatore |
|      104 | open auditorium  | Chennai    |
|      105 | devi complex     | Bangalore  |
|      106 | priya theatre    | Erode      |
|      107 | kannan bazaar    | Karur      |
|      108 | kanmani college  | Madurai    |
|      109 | mathi theatre    | Mumbai     |
|      110 | dharshini bazaar | Gujarat    |
+----------+------------------+------------+
10 rows in set (0.03 sec)
```

Inserting values into event table

```
mysql> insert into event values(1,'Beats concert','2024-04-02','05:00',101,1000,700,2000.0,'Concert',500),
    -> (2,'summer concert','2024-04-03','05:30',101,2000,800,3000.0,'Concert',600),
    -> (3,'harmony concert','2024-04-04','06:00',102,3000,900,4000.0,'Concert',700),
    -> (4,'world_cup match','2024-04-05','06:30',103,4000,1000,5000.0,'Sports',800),
    -> (5,'cricket world cup','2024-04-06','07:00',103,5000,1100,6000.0,'Sports',800),
    -> (6,'kabaddi','2024-04-07','07:30',104,6000,1200,7000.0,'Sports',900),
    -> (7,'Harry potter','2024-04-08','08:00',105,7000,1300,8000.0,'Movie',900),
    -> (8,'Joe','2024-04-09','08:30',105,7500,1500,9000.0,'Movie',500),
    -> (9,'Life_of_pi','2024-04-10','09:00',105,8000,1600,9500.0,'Movie',700),
    -> (10,'Jurassic_Park','2024-04-11','09:30',106,9000,2000,10000.0,'Movie',600);
Query OK, 10 rows affected (0.15 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
+----------+-----------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
| event_id | event_name      | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+-----------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
|        1 | Beats concert   | 2024-04-02 | 05:00:00   |      101 |        1000 |            700 |         2000 | Concert    |        500 |
|        2 | summer concert  | 2024-04-03 | 05:30:00   |      101 |        2000 |            800 |         3000 | Concert    |        600 |
|        3 | harmony concert | 2024-04-04 | 06:00:00   |      102 |        3000 |            900 |         4000 | Concert    |        700 |
|        4 | world_cup match | 2024-04-05 | 06:30:00   |      103 |        4000 |           1000 |         5000 | Sports     |        800 |
|        5 | cricket world cup | 2024-04-06 | 07:00:00 |      103 |        5000 |           1100 |         6000 | Sports     |        800 |
|        6 | kabaddi         | 2024-04-07 | 07:30:00   |      104 |        6000 |           1200 |         7000 | Sports     |        900 |
|        7 | Harry potter    | 2024-04-08 | 08:00:00   |      105 |        7000 |           1300 |         8000 | Movie      |        900 |
|        8 | Joe             | 2024-04-09 | 08:30:00   |      105 |        7500 |           1500 |         9000 | Movie      |        500 |
|        9 | Life_of_pi      | 2024-04-10 | 09:00:00   |      105 |        8000 |           1600 |         9500 | Movie      |        700 |
|       10 | Jurassic_Park   | 2024-04-11 | 09:30:00   |      106 |        9000 |           2000 |        10000 | Movie      |        600 |
+----------+-----------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
10 rows in set (0.00 sec)
```

Inserting values into customer table

```
mysql> insert into customer values(201,'kavi','kavi@gmail.com','9038606367',200),
    -> (202,'diya','diya@gmail.com','9898606367',200),
    -> (203,'nithya','nithya@gmail.com','8768608000',300),
    -> (204,'deva','deva@gmail.com','9038456792',400),
    -> (205,'sowmiya','sowmiya@gmail.com','8745632907',100),
    -> (206,'vino','vino@gmail.com','7576975000',500),
    -> (207,'divya','divya@gmail.com','9875623417',300),
    -> (208,'siva','siva@gmail.com','9465787843',500),
    -> (209,'pravin','pravin@gmail.com','9067854325',600),
    -> (210,'bagavathi','bagavathi@gmail.com','9987656788',700);
Query OK, 10 rows affected (0.09 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
+-------------+---------------+---------------------+--------------+------------+
| customer_id | customer_name | email               | phone_number | booking_id |
+-------------+---------------+---------------------+--------------+------------+
|         201 | kavi          | kavi@gmail.com      | 9038606367   |        200 |
|         202 | diya          | diya@gmail.com      | 9898606367   |        200 |
|         203 | nithya        | nithya@gmail.com    | 8768608000   |        300 |
|         204 | deva          | deva@gmail.com      | 9038456792   |        400 |
|         205 | sowmiya       | sowmiya@gmail.com   | 8745632907   |        100 |
|         206 | vino          | vino@gmail.com      | 7576975000   |        500 |
|         207 | divya         | divya@gmail.com     | 9875623417   |        300 |
|         208 | siva          | siva@gmail.com      | 9465787843   |        500 |
|         209 | pravin        | pravin@gmail.com    | 9067854325   |        600 |
|         210 | bagavathi     | bagavathi@gmail.com | 9987656788   |        700 |
+-------------+---------------+---------------------+--------------+------------+
10 rows in set (0.00 sec)
```

Creating values in booking table

```
+------------+-------------+----------+-------------+------------+--------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
+------------+-------------+----------+-------------+------------+--------------+
|        100 |         201 |        1 |          50 |     100000 | 2024-05-12   |
|        200 |         201 |        1 |          60 |     180000 | 2024-05-13   |
|        300 |         202 |        2 |          70 |     280000 | 2024-05-14   |
|        400 |         209 |        1 |          80 |     400000 | 2024-05-15   |
|        500 |         210 |        3 |          30 |     180000 | 2024-05-16   |
|        600 |         203 |        4 |          20 |     140000 | 2024-05-17   |
|        700 |         203 |        5 |          90 |     720000 | 2024-05-18   |
|        800 |         206 |        6 |          40 |     360000 | 2024-05-19   |
|        900 |         207 |        6 |         100 |     950000 | 2024-05-20   |
|       1000 |         208 |        8 |          10 |     100000 | 2024-05-21   |
+------------+-------------+----------+-------------+------------+--------------+
10 rows in set (0.00 sec)
```

```
mysql> insert into booking values(100,201,1,50,100000,'2024-05-12'),
    -> (200,201,1,60,180000,'2024-05-13'),
    -> (300,202,2,70,280000,'2024-05-14'),
    -> (400,209,1,80,400000,'2024-05-15'),
    -> (500,210,3,30,180000,'2024-05-16'),
    -> (600,203,4,20,140000,'2024-05-17'),
    -> (700,203,5,90,720000,'2024-05-18'),
    -> (800,206,6,40,360000,'2024-05-19'),
    -> (900,207,6,100,950000,'2024-05-20'),
    -> (1000,208,8,10,100000,'2024-05-21');
Query OK, 10 rows affected (0.14 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Adding foreign key constraint to event and customer table

```
mysql> alter table event add constraint eve foreign key(booking_id) references booking(booking_id);
Query OK, 10 rows affected (2.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> alter table customer add constraint cus foreign key(booking_id) references booking(booking_id);
Query OK, 10 rows affected (1.25 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> desc event;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| event_id       | int          | NO   | PRI | NULL    |       |
| event_name     | varchar(50)  | YES  |     | NULL    |       |
| event_date     | date         | YES  |     | NULL    |       |
| event_time     | time         | YES  |     | NULL    |       |
| venue_id       | int          | YES  | MUL | NULL    |       |
| total_seats    | int          | YES  |     | NULL    |       |
| available_seats | int         | YES  |     | NULL    |       |
| ticket_price   | decimal(10,0) | YES  |     | NULL    |       |
| event_type     | varchar(20)  | YES  |     | NULL    |       |
| booking_id     | int          | YES  | MUL | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

```
mysql> desc customer;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| customer_id   | int         | NO   | PRI | NULL    |       |
| customer_name | varchar(50) | YES  |     | NULL    |       |
| email         | varchar(50) | YES  |     | NULL    |       |
| phone_number  | varchar(15) | YES  |     | NULL    |       |
| booking_id    | int         | YES  | MUL | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

2) SQL query to list all Events.

```
mysql> select * from event;
+----------+-----------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
| event_id | event_name      | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+-----------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
|        1 | Beats concert   | 2024-04-02 | 05:00:00   |      101 |        1000 |            700 |         2000 | Concert    |        500 |
|        2 | summer concert  | 2024-04-03 | 05:30:00   |      101 |        2000 |            800 |         3000 | Concert    |        600 |
|        3 | harmony concert | 2024-04-04 | 06:00:00   |      102 |        3000 |            900 |         4000 | Concert    |        700 |
|        4 | world_cup match | 2024-04-05 | 06:30:00   |      103 |        4000 |           1000 |         5000 | Sports     |        800 |
|        5 | cricket world cup | 2024-04-06 | 07:00:00 |      103 |        5000 |           1100 |         6000 | Sports     |        800 |
|        6 | kabaddi         | 2024-04-07 | 07:30:00   |      104 |        6000 |           1200 |         7000 | Sports     |        900 |
|        7 | Harry potter    | 2024-04-08 | 08:00:00   |      105 |        7000 |           1300 |         8000 | Movie      |        900 |
|        8 | Joe             | 2024-04-09 | 08:30:00   |      105 |        7500 |           1500 |         9000 | Movie      |        500 |
|        9 | Life_of_pi      | 2024-04-10 | 09:00:00   |      105 |        8000 |           1600 |         9500 | Movie      |        700 |
|       10 | Jurassic_Park   | 2024-04-11 | 09:30:00   |      106 |        9000 |           2000 |        10000 | Movie      |        600 |
+----------+-----------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
10 rows in set (0.00 sec)
```

3) SQL query to select events with available tickets

```
mysql> select event_name,available_seats from event;
+--------------------+-----------------+
| event_name         | available_seats |
+--------------------+-----------------+
| Beats concert      |             700 |
| summer concert     |             800 |
| harmony concert    |             900 |
| world_cup match    |            1000 |
| cricket world cup  |            1100 |
| kabaddi            |            1200 |
| Harry potter       |            1300 |
| Joe                |            1500 |
| Life_of_pi         |            1600 |
| Jurassic_Park      |            2000 |
+--------------------+-----------------+
10 rows in set (0.00 sec)
```

4)SQL query to select events name partial match with ‘cup’

```
mysql> select event_name from event where event_name like '%cup%';
+-------------------+
| event_name        |
+-------------------+
| world_cup match   |
| cricket world cup |
+-------------------+
2 rows in set (0.00 sec)
```

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

```
mysql> Select event_name from event where ticket_price  between 1000 and 2500;
+---------------+
| event_name    |
+---------------+
| Beats concert |
+---------------+
1 row in set (0.06 sec)
```

6) SQL query to retrieve events with dates falling within a specific range

```
mysql> select event_name,event_date from event where event_date between '2024-04-03' and '2024-04-06';
+------------------+------------+
| event_name       | event_date |
+------------------+------------+
| summer concert   | 2024-04-03 |
| harmony concert  | 2024-04-04 |
| world_cup match  | 2024-04-05 |
| cricket world cup | 2024-04-06 |
+------------------+------------+
4 rows in set (0.04 sec)
```

7) SQL query to retrieve events with available tickets that also have "Concert" in their name

```
mysql> select event_name,available_seats from event where event_name like '%concert%';
+-----------------+-----------------+
| event_name      | available_seats |
+-----------------+-----------------+
| Beats concert   |             700 |
| summer concert  |             800 |
| harmony concert |             900 |
+-----------------+-----------------+
3 rows in set (0.00 sec)
```

8) SQL query to retrieve users in batches of 5, starting from the 6th user.

```
mysql> select customer_name from Customer limit 5 offset 5;
+---------------+
| customer_name |
+---------------+
| vino          |
| divya         |
| siva          |
| pravin        |
| bagavathi     |
+---------------+
5 rows in set (0.80 sec)
```

9) SQL query to retrieve bookings details contains booked no of ticket more than 4

```
mysql> select * from booking where num_tickets>4;
+------------+-------------+----------+-------------+------------+--------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
+------------+-------------+----------+-------------+------------+--------------+
|        100 |         201 |        1 |          50 |     100000 | 2024-05-12   |
|        200 |         201 |        1 |          60 |     180000 | 2024-05-13   |
|        300 |         202 |        2 |          70 |     280000 | 2024-05-14   |
|        400 |         209 |        1 |          80 |     400000 | 2024-05-15   |
|        500 |         210 |        3 |          30 |     180000 | 2024-05-16   |
|        600 |         203 |        4 |          20 |     140000 | 2024-05-17   |
|        700 |         203 |        5 |          90 |     720000 | 2024-05-18   |
|        800 |         206 |        6 |          40 |     360000 | 2024-05-19   |
|        900 |         207 |        6 |         100 |     950000 | 2024-05-20   |
|       1000 |         208 |        8 |          10 |     100000 | 2024-05-21   |
+------------+-------------+----------+-------------+------------+--------------+
10 rows in set (0.13 sec)
```

10) SQL query to retrieve customer information whose phone number end with ‘000’

```
mysql> select * from customer where phone_number like '%000';
+-------------+---------------+-----------------+--------------+------------+
| customer_id | customer_name | email           | phone_number | booking_id |
+-------------+---------------+-----------------+--------------+------------+
|         203 | nithya        | nithya@gmail.com | 8768608000   |        300 |
|         206 | vino          | vino@gmail.com   | 7576975000   |        500 |
+-------------+---------------+-----------------+--------------+------------+
2 rows in set (0.00 sec)
```

11) SQL query to retrieve the events in order whose seat capacity more than 1500

```
mysql> select * from event where available_seats>1500 order by available_seats;
+----------+-------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name  | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+-------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        9 | Life_of_pi  | 2024-04-10 | 09:00:00   |      105 |        8000 |            1600 |         9500 | Movie      |        700 |
|       10 | Jurassic_Park | 2024-04-11 | 09:30:00   |      106 |        9000 |            2000 |        10000 | Movie      |        600 |
+----------+-------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
2 rows in set (0.00 sec)
```

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

```
mysql> select event_name from event where event_name not like 'X%' or 'Y%' or 'Z%';
+-------------------+
| event_name        |
+-------------------+
| Beats concert     |
| summer concert    |
| harmony concert   |
| world_cup match   |
| cricket world cup |
| kabaddi           |
| Harry potter      |
| Joe               |
| Life_of_pi        |
| Jurassic_Park     |
+-------------------+
10 rows in set, 2 warnings (0.00 sec)
```

Task-3:

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

```
mysql> select event_id, event_name, avg(ticket_price) from event group by event_id, event_name;
+----------+-------------------+-------------------+
| event_id | event_name        | avg(ticket_price) |
+----------+-------------------+-------------------+
|        1 | Beats concert     |         2000.0000 |
|        2 | summer concert    |         3000.0000 |
|        3 | harmony concert   |         4000.0000 |
|        4 | world_cup match   |         5000.0000 |
|        5 | cricket world cup |         6000.0000 |
|        6 | kabaddi           |         7000.0000 |
|        7 | Harry potter      |         8000.0000 |
|        8 | Joe               |         9000.0000 |
|        9 | Life_of_pi        |         9500.0000 |
|       10 | Jurassic_Park     |        10000.0000 |
+----------+-------------------+-------------------+
10 rows in set (0.37 sec)
```

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

```
mysql> select sum(ticket_price) from event;
+--------------------+
| sum(ticket_price) |
+--------------------+
|              63500 |
+--------------------+
1 row in set (0.05 sec)
```

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

```
mysql> select event_name,(total_seats-available_seats) as highest_ticket_sales from event order by
    -> (highest_ticket_sales) desc limit 1;
+-------------+--------------------+
| event_name  | highest_ticket_sales |
+-------------+--------------------+
| Jurassic_Park |               7000 |
+-------------+--------------------+
1 row in set (0.39 sec)
```

4) SQL query to Calculate the Total Number of Tickets Sold for Each Event

```
mysql> select event_name, sum(num_tickets)  from booking
    -> join event on booking.event_id = event.event_id group by event_name;
+--------------------+------------------+
| event_name         | sum(num_tickets) |
+--------------------+------------------+
| Beats concert      |              190 |
| summer concert     |               70 |
| harmony concert    |               30 |
| world_cup match    |               20 |
| cricket world cup  |               90 |
| kabaddi            |              140 |
| Joe                |               10 |
+--------------------+------------------+
7 rows in set (0.00 sec)
```

5) a SQL query to Find Events with No Ticket Sales

```
mysql> select event_name from event where event_id not in (select distinct event_id from
    -> booking);
+--------------+
| event_name   |
+--------------+
| Harry potter |
| Life_of_pi   |
| Jurassic_Park |
+--------------+
3 rows in set (0.00 sec)
```

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

```
mysql> select customer_name, sum(num_tickets) from customer  join booking on customer.customer_id = booking.customer_id
    -> group by customer_name order by sum(num_tickets) desc limit 1;
+-------------+-----------------+
| customer_name | sum(num_tickets) |
+-------------+-----------------+
| kavi        |             110 |
+-------------+-----------------+
1 row in set (0.01 sec)
```

7) SQL query to List Events and the total number of tickets sold for each month.

```
mysql> select month(booking_date),year(booking_date),sum(num_tickets) from booking
    -> group by month(booking_date),year(booking_date);
+--------------------+-------------------+------------------+
| month(booking_date) | year(booking_date) | sum(num_tickets) |
+--------------------+-------------------+------------------+
|                  5 |              2024 |              550 |
+--------------------+-------------------+------------------+
1 row in set (0.06 sec)
```

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

```
mysql> select venue.venue_id,venue.venue_name,avg(event.ticket_price) from venue
    -> join event on venue.venue_id = event.venue_id
    -> group by venue.venue_id,venue.venue_name;
+----------+-----------------+-------------------------+
| venue_id | venue_name      | avg(event.ticket_price) |
+----------+-----------------+-------------------------+
|      101 | Auditorium      |               2500.0000 |
|      102 | raja mahal      |               4000.0000 |
|      103 | rani palace     |               5500.0000 |
|      104 | open auditorium |               7000.0000 |
|      105 | devi complex    |               8833.3333 |
|      106 | priya theatre   |              10000.0000 |
+----------+-----------------+-------------------------+
6 rows in set (0.00 sec)
```

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

```
mysql>  select event_type, sum(num_tickets) from booking join
    -> event on booking.event_id = event.event_id group by event_type;
+------------+------------------+
| event_type | sum(num_tickets) |
+------------+------------------+
| Concert    |              290 |
| Sports     |              250 |
| Movie      |               10 |
+------------+------------------+
3 rows in set (0.00 sec)
```

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

```
mysql> select year(booking_date),sum(total_cost) from booking
    -> group by year(booking_date);
+---------------------+------------------+
| year(booking_date) | sum(total_cost) |
+---------------------+------------------+
|                2024 |          3410000 |
+---------------------+------------------+
1 row in set (0.00 sec)
```

11) SQL query to list users who have booked tickets for multiple events

```
mysql> select customer_id,count(distinct event_id) from booking group by customer_id
   -> having count(distinct event_id) > 1;
+------------+------------------------+
| customer_id | count(distinct event_id) |
+------------+------------------------+
|        203 |                      2 |
+------------+------------------------+
1 row in set (0.00 sec)
```

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

```
mysql> select customer_id,sum(total_cost) from booking group by customer_id;
+-------------+-----------------+
| customer_id | sum(total_cost) |
+-------------+-----------------+
|         201 |          280000 |
|         202 |          280000 |
|         203 |          860000 |
|         206 |          360000 |
|         207 |          950000 |
|         208 |          100000 |
|         209 |          400000 |
|         210 |          180000 |
+-------------+-----------------+
8 rows in set (0.02 sec)
```

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

```
mysql> select event.event_type,event.venue_id,avg(event.ticket_price) from event
    -> group by event.event_type,event.venue_id;
+------------+----------+--------------------------+
| event_type | venue_id | avg(event.ticket_price) |
+------------+----------+--------------------------+
| Concert    |      101 |               2500.0000 |
| Concert    |      102 |               4000.0000 |
| Sports     |      103 |               5500.0000 |
| Sports     |      104 |               7000.0000 |
| Movie      |      105 |               8833.3333 |
| Movie      |      106 |              10000.0000 |
+------------+----------+--------------------------+
6 rows in set (0.00 sec)
```

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

```
mysql> select customer_id, count(*) as total_tickets_purchased from booking
    -> where booking_date >= date_sub(current_date(), interval 30 day) group by customer_id;
+-------------+------------------------+
| customer_id | total_tickets_purchased |
+-------------+------------------------+
|         201 |                      2 |
|         202 |                      1 |
|         203 |                      2 |
|         206 |                      1 |
|         207 |                      1 |
|         208 |                      1 |
|         209 |                      1 |
|         210 |                      1 |
+-------------+------------------------+
8 rows in set (0.17 sec)
```

Task-4:

1)Average Ticket Price for Events in Each Venue Using a Subquery

```
mysql> select v.venue_name,(select avg(ticket_price) from event where venue_id =
    -> v.venue_id) as avg_ticket_price from venue v;
+-----------------+------------------+
| venue_name      | avg_ticket_price |
+-----------------+------------------+
| Auditorium      |        2500.0000 |
| raja mahal      |        4000.0000 |
| rani palace     |        5500.0000 |
| open auditorium |        7000.0000 |
| devi complex    |        8833.3333 |
| priya theatre   |       10000.0000 |
| kannan bazaar   |             NULL |
| kanmani college |             NULL |
| mathi theatre   |             NULL |
| dharshini bazaar|             NULL |
+-----------------+------------------+
10 rows in set (0.07 sec)
```

2)Finding Events with More Than 50% of Tickets Sold using subquery

```
mysql> select event_id,event_name from event
    -> where (select sum(num_tickets) from booking where booking.event_id=event.event_id) >(total_seats/2);
Empty set (0.06 sec)
```

3) Calculating the Total Number of Tickets Sold for Each Event.

```
mysql> select event_id, event_name, (select sum(num_tickets) from Booking where
    -> Booking.event_id = Event.event_id) as total_tickets_sold from Event;
+----------+-------------------+--------------------+
| event_id | event_name        | total_tickets_sold |
+----------+-------------------+--------------------+
|        1 | Beats concert     |                190 |
|        2 | summer concert    |                 70 |
|        3 | harmony concert   |                 30 |
|        4 | world_cup match   |                 20 |
|        5 | cricket world cup |                 90 |
|        6 | kabaddi           |                140 |
|        7 | Harry potter      |               NULL |
|        8 | Joe               |                 10 |
|        9 | Life_of_pi        |               NULL |
|       10 | Jurassic_Park     |               NULL |
+----------+-------------------+--------------------+
10 rows in set (0.00 sec)
```

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

```
mysql> select customer_id, customer_name from Customer c where not exists (select *
    -> from Booking where Booking.customer_id = c.customer_id);
+-------------+---------------+
| customer_id | customer_name |
+-------------+---------------+
|         204 | deva          |
|         205 | sowmiya       |
+-------------+---------------+
2 rows in set (0.12 sec)
```

5) Listing Events with No Ticket Sales Using a NOT IN Subquery.

```
mysql> select event_id, event_name from event where event_id not in (select distinct
    -> event_id from Booking);
+----------+---------------+
| event_id | event_name    |
+----------+---------------+
|        7 | Harry potter  |
|        9 | Life_of_pi    |
|       10 | Jurassic_Park |
+----------+---------------+
3 rows in set (0.04 sec)
```

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

```
mysql> select e1.event_type,  e1.total_seats - e1.available_seats as total_tickets_sold
    -> from (select event_type, sum(total_seats) as total_seats, sum(available_seats) as available_seats
    -> from event group by event_type) as e1;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Concert    |               3600 |
| Sports     |              11700 |
| Movie      |              25100 |
+------------+--------------------+
3 rows in set (0.17 sec)
```

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

```
mysql> select event_id, event_name, ticket_price from event where ticket_price >(select
    -> avg(ticket_price) from event);
+----------+---------------+--------------+
| event_id | event_name    | ticket_price |
+----------+---------------+--------------+
|        6 | kabaddi       |         7000 |
|        7 | Harry potter  |         8000 |
|        8 | Joe           |         9000 |
|        9 | Life_of_pi    |         9500 |
|       10 | Jurassic_Park |        10000 |
+----------+---------------+--------------+
5 rows in set (0.08 sec)
```

8) Calculating the Total Revenue Generated by Events for Each User Using a Correlated Subquery

```
mysql> select customer_id, customer_name,(select sum(total_cost) from booking where
    -> booking.customer_id = customer.customer_id) as total_revenue from customer;
+-------------+---------------+---------------+
| customer_id | customer_name | total_revenue |
+-------------+---------------+---------------+
|         201 | kavi          |        280000 |
|         202 | diya          |        280000 |
|         203 | nithya        |        860000 |
|         204 | deva          |          NULL |
|         205 | sowmiya       |          NULL |
|         206 | vino          |        360000 |
|         207 | divya         |        950000 |
|         208 | siva          |        100000 |
|         209 | pravin        |        400000 |
|         210 | bagavathi     |        180000 |
+-------------+---------------+---------------+
10 rows in set (0.00 sec)
```

9) Listing Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause

```
mysql> select customer_id, customer_name from customer where customer_id in (select
    -> distinct customer_id from Booking where event_id in (select event_id from event
    -> where venue_id = (select venue_id from venue where venue_name ='Auditorium')));
+-------------+---------------+
| customer_id | customer_name |
+-------------+---------------+
|         201 | kavi          |
|         209 | pravin        |
|         202 | diya          |
+-------------+---------------+
3 rows in set (0.25 sec)
```

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

```
mysql> select event_type, sum(total_tickets_sold) as total_tickets_sold
    -> from(select event_type, sum(total_seats-available_seats) as total_tickets_sold
    -> from event group by event_id) as total_tickets group by event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Concert    |               3600 |
| Sports     |              11700 |
| Movie      |              25100 |
+------------+--------------------+
3 rows in set (0.07 sec)
```

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

```
mysql> select c.customer_id,  c.customer_name,  (select b.booking_date from booking b
    -> where c.booking_id = b.booking_id) as booking_date
    -> from customers c
    -> order by booking_date;
+-------------+---------------+--------------+
| customer_id | customer_name | booking_date |
+-------------+---------------+--------------+
| c9          | Rani          | 2024-04-05   |
| c7          | Vino          | 2024-04-07   |
| c6          | Sowmi         | 2024-04-08   |
| c3          | Deepi         | 2024-04-10   |
| c4          | Siddhu        | 2024-04-11   |
| c8          | Sanjai        | 2024-04-14   |
| c5          | Manoj         | 2024-04-19   |
| c2          | Abi           | 2024-04-20   |
| c1          | Ravi          | 2024-04-25   |
| c0          | Raja          | 2024-04-27   |
+-------------+---------------+--------------+
10 rows in set (0.00 sec)
```

12) Calculating the Average Ticket Price for Events in Each Venue Using a Subquery

```
mysql> select venue.venue_name,(select avg(ticket_price) from event where event.venue_id =
    -> venue.venue_id) as avg_ticket_price from venue;
+-----------------+------------------+
| venue_name      | avg_ticket_price |
+-----------------+------------------+
| Auditorium      |        2500.0000 |
| raja mahal      |        4000.0000 |
| rani palace     |        5500.0000 |
| open auditorium |        7000.0000 |
| devi complex    |        8833.3333 |
| priya theatre   |       10000.0000 |
| kannan bazaar   |             NULL |
| kanmani college |             NULL |
| mathi theatre   |             NULL |
| dharshini bazaar|             NULL |
+-----------------+------------------+
10 rows in set (0.00 sec)
```