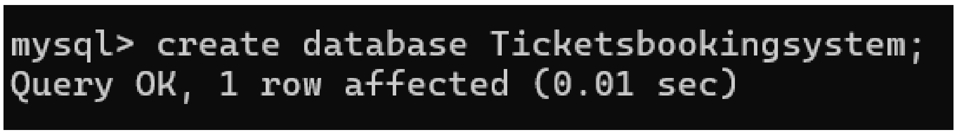
Assignment

Ticket booking system

Task 1:

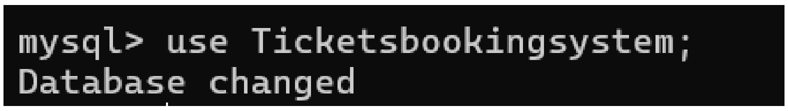
Database design:

1.Create the database named "Tickets booking system":



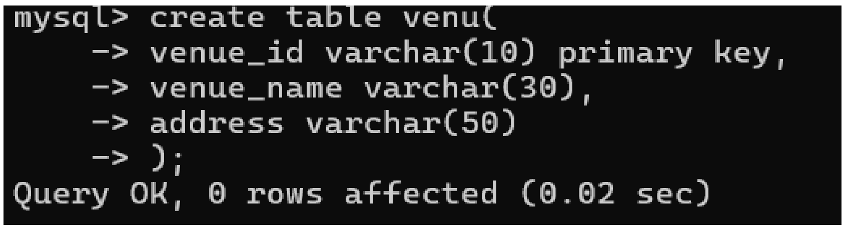
```
mysql> create database Ticketsbookingsystem;
Query OK, 1 row affected (0.01 sec)
```

Use database:

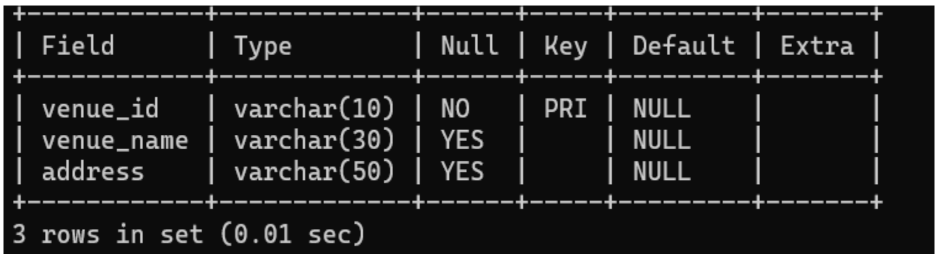


```
mysql> use Ticketsbookingsystem;
Database changed
```

2.Creating table venue:

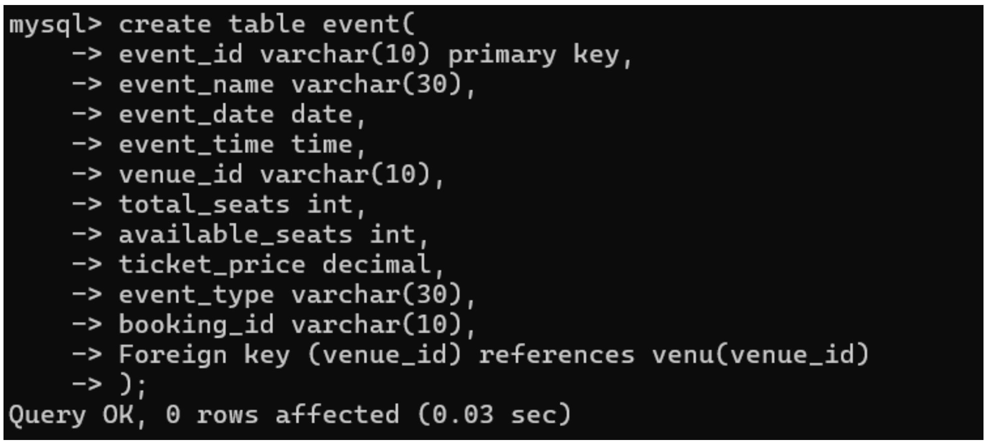


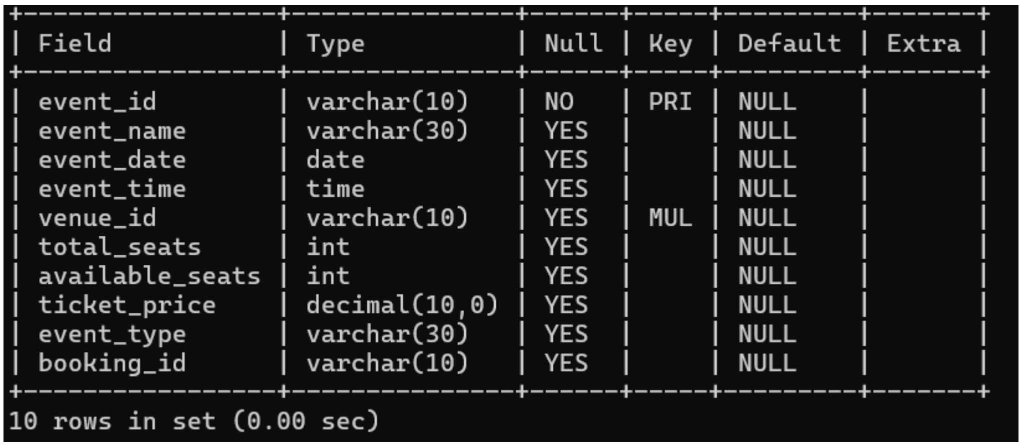
```
mysql> create table venu(
    -> venue_id varchar(10) primary key,
    -> venue_name varchar(30),
    -> address varchar(50)
    -> );
Query OK, 0 rows affected (0.02 sec)
```



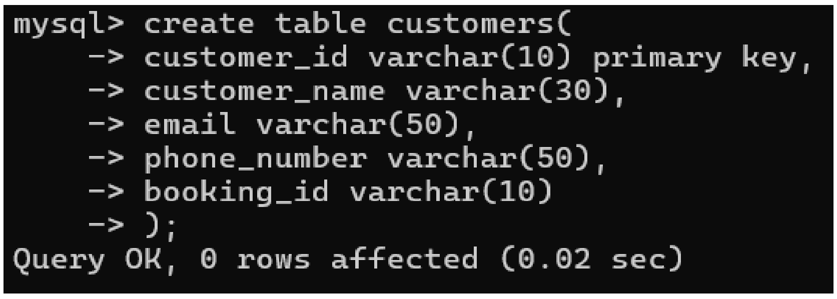
```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| venue_id   | varchar(10) | NO   | PRI | NULL    |       |
| venue_name | varchar(30) | YES  |     | NULL    |       |
| address    | varchar(50) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

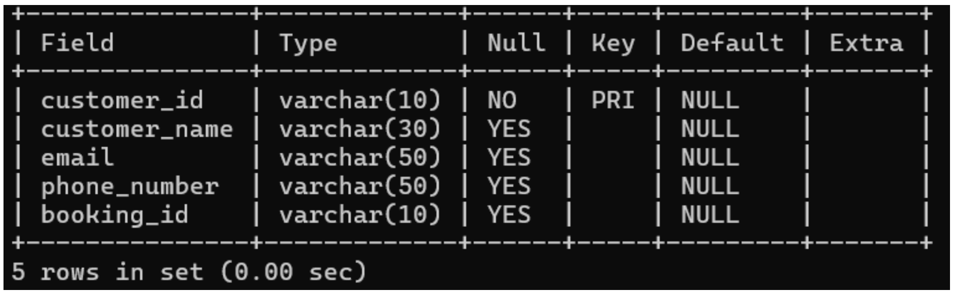
Creating table event:


```
mysql> create table event(
    -> event_id varchar(10) primary key,
    -> event_name varchar(30),
    -> event_date date,
    -> event_time time,
    -> venue_id varchar(10),
    -> total_seats int,
    -> available_seats int,
    -> ticket_price decimal,
    -> event_type varchar(30),
    -> booking_id varchar(10),
    -> Foreign key (venue_id) references venu(venue_id)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

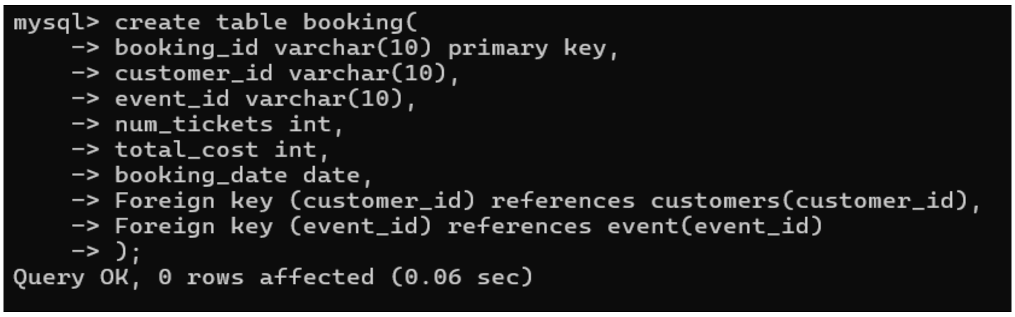

```
+-----------------+---------------+------+-----+---------+-------+
| Field           | Type          | Null | Key | Default | Extra |
+-----------------+---------------+------+-----+---------+-------+
| event_id        | varchar(10)   | NO   | PRI | NULL    |       |
| event_name      | varchar(30)   | YES  |     | NULL    |       |
| event_date      | date          | YES  |     | NULL    |       |
| event_time      | time          | YES  |     | NULL    |       |
| venue_id        | varchar(10)   | YES  | MUL | NULL    |       |
| total_seats     | int           | YES  |     | NULL    |       |
| available_seats | int           | YES  |     | NULL    |       |
| ticket_price    | decimal(10,0) | YES  |     | NULL    |       |
| event_type      | varchar(30)   | YES  |     | NULL    |       |
| booking_id      | varchar(10)   | YES  |     | NULL    |       |
+-----------------+---------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

Creating customers table:

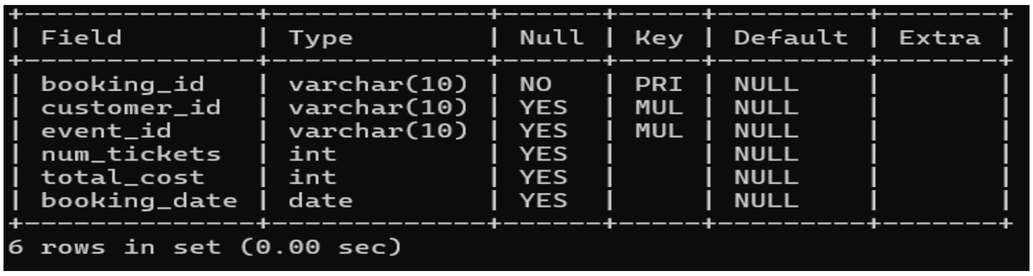

```
mysql> create table customers(
    -> customer_id varchar(10) primary key,
    -> customer_name varchar(30),
    -> email varchar(50),
    -> phone_number varchar(50),
    -> booking_id varchar(10)
    -> );
Query OK, 0 rows affected (0.02 sec)
```


```
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| customer_id   | varchar(10) | NO   | PRI | NULL    |       |
| customer_name | varchar(30) | YES  |     | NULL    |       |
| email         | varchar(50) | YES  |     | NULL    |       |
| phone_number  | varchar(50) | YES  |     | NULL    |       |
| booking_id    | varchar(10) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Creating booking table:

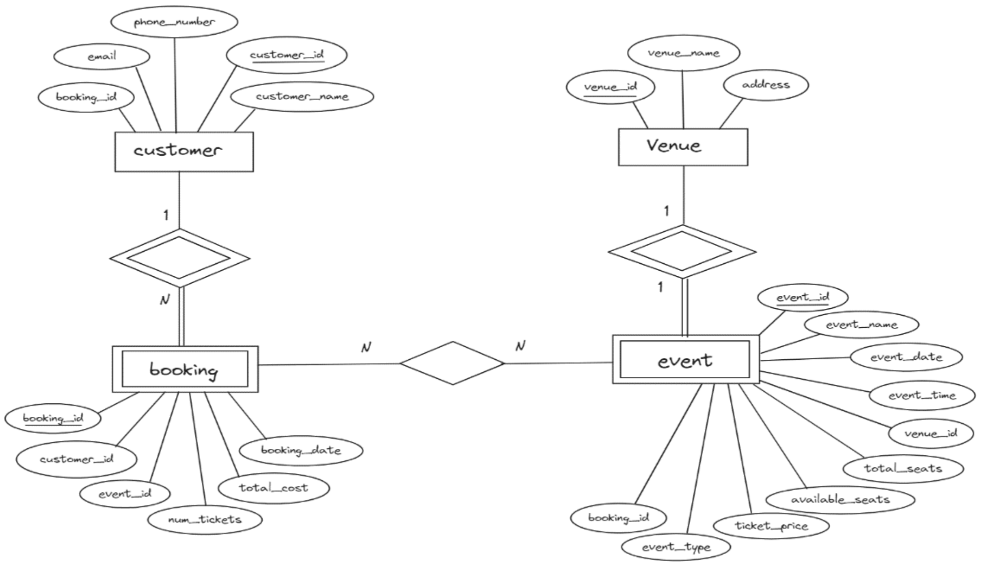


```
mysql> create table booking(
    -> booking_id varchar(10) primary key,
    -> customer_id varchar(10),
    -> event_id varchar(10),
    -> num_tickets int,
    -> total_cost int,
    -> booking_date date,
    -> Foreign key (customer_id) references customers(customer_id),
    -> Foreign key (event_id) references event(event_id)
    -> );
Query OK, 0 rows affected (0.06 sec)
```



```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| booking_id   | varchar(10) | NO   | PRI | NULL    |       |
| customer_id  | varchar(10) | YES  | MUL | NULL    |       |
| event_id     | varchar(10) | 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)
```

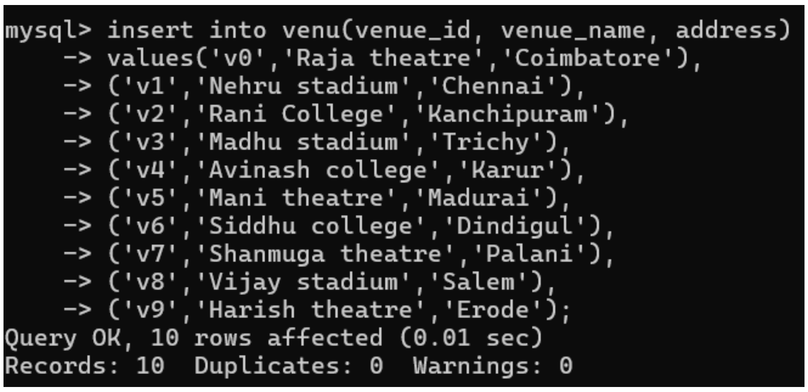
3. Create an ERD (Entity Relationship Diagram) for the database:



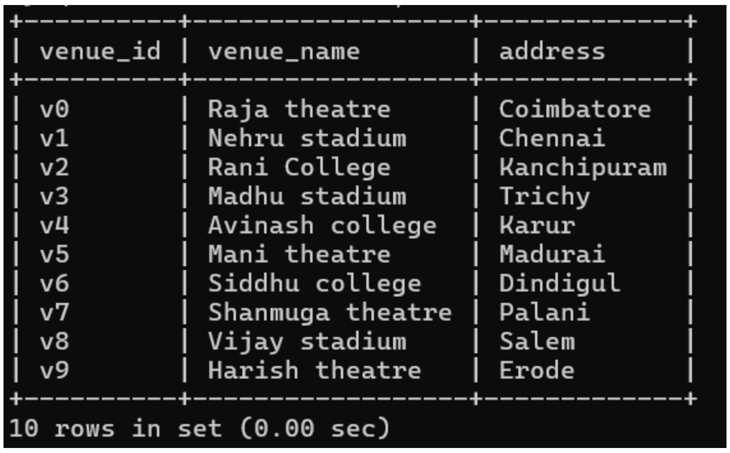
Task 2:

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

Inserting 10 records in the venu table:

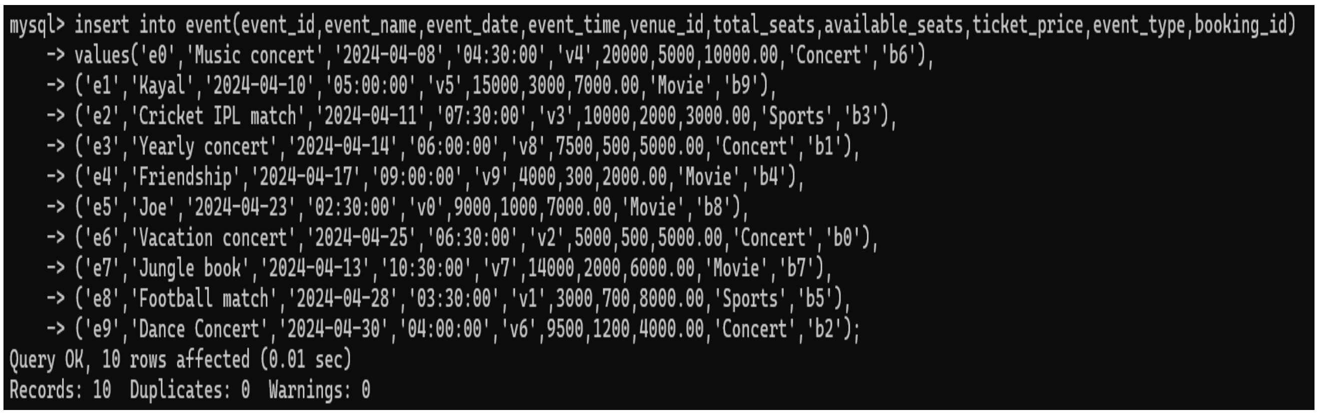


```
mysql> insert into venu(venue_id, venue_name, address)
    -> values('v0','Raja theatre','Coimbatore'),
    -> ('v1','Nehru stadium','Chennai'),
    -> ('v2','Rani College','Kanchipuram'),
    -> ('v3','Madhu stadium','Trichy'),
    -> ('v4','Avinash college','Karur'),
    -> ('v5','Mani theatre','Madurai'),
    -> ('v6','Siddhu college','Dindigul'),
    -> ('v7','Shanmuga theatre','Palani'),
    -> ('v8','Vijay stadium','Salem'),
    -> ('v9','Harish theatre','Erode');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

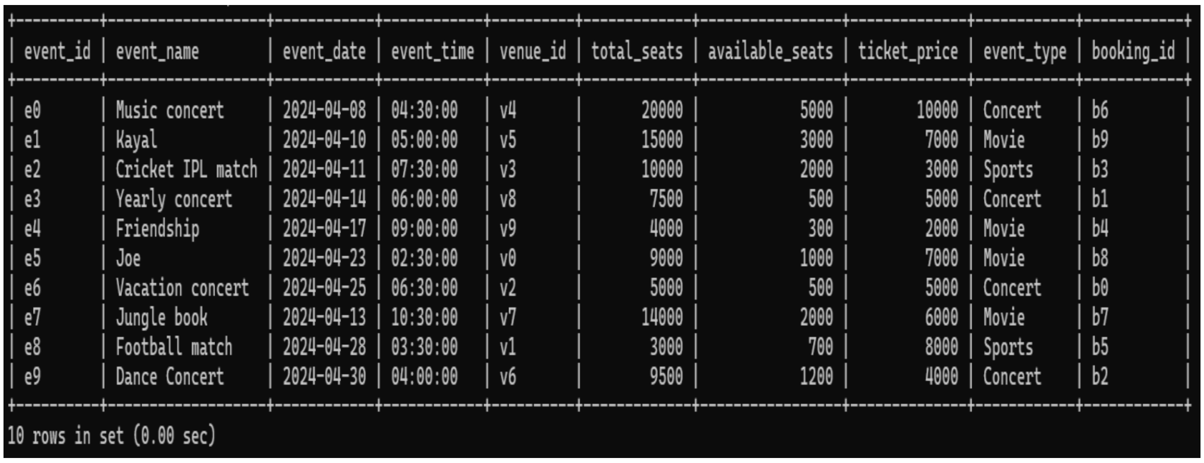


```
+----------+------------------+-------------+
| venue_id | venue_name       | address     |
+----------+------------------+-------------+
| v0       | Raja theatre     | Coimbatore  |
| v1       | Nehru stadium    | Chennai     |
| v2       | Rani College     | Kanchipuram |
| v3       | Madhu stadium    | Trichy      |
| v4       | Avinash college  | Karur       |
| v5       | Mani theatre     | Madurai     |
| v6       | Siddhu college   | Dindigul    |
| v7       | Shanmuga theatre | Palani      |
| v8       | Vijay stadium    | Salem       |
| v9       | Harish theatre   | Erode       |
+----------+------------------+-------------+
10 rows in set (0.00 sec)
```

Inserting 10 records in the event table:

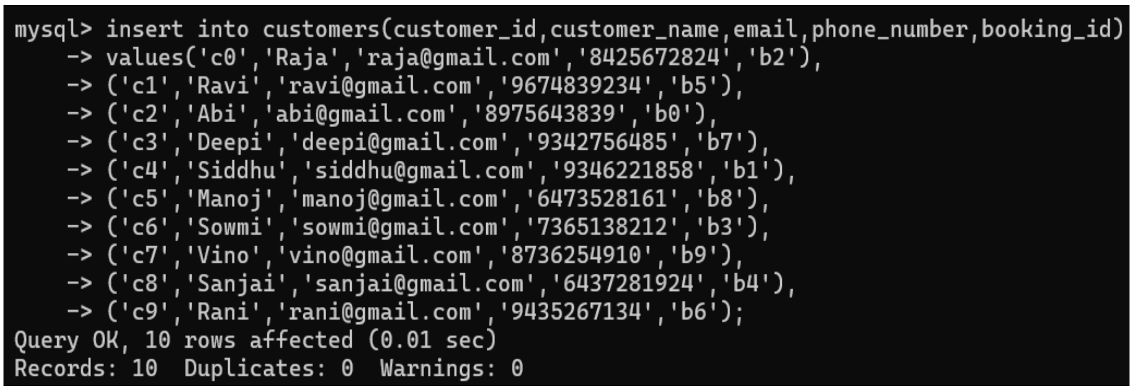


```
mysql> insert into event(event_id,event_name,event_date,event_time,venue_id,total_seats,available_seats,ticket_price,event_type,booking_id)
    -> values('e0','Music concert','2024-04-08','04:30:00','v4',20000,5000,10000.00,'Concert','b6'),
    -> ('e1','Kayal','2024-04-10','05:00:00','v5',15000,3000,7000.00,'Movie','b9'),
    -> ('e2','Cricket IPL match','2024-04-11','07:30:00','v3',10000,2000,3000.00,'Sports','b3'),
    -> ('e3','Yearly concert','2024-04-14','06:00:00','v8',7500,500,5000.00,'Concert','b1'),
    -> ('e4','Friendship','2024-04-17','09:00:00','v9',4000,300,2000.00,'Movie','b4'),
    -> ('e5','Joe','2024-04-23','02:30:00','v0',9000,1000,7000.00,'Movie','b8'),
    -> ('e6','Vacation concert','2024-04-25','06:30:00','v2',5000,500,5000.00,'Concert','b0'),
    -> ('e7','Jungle book','2024-04-13','10:30:00','v7',14000,2000,6000.00,'Movie','b7'),
    -> ('e8','Football match','2024-04-28','03:30:00','v1',3000,700,8000.00,'Sports','b5'),
    -> ('e9','Dance Concert','2024-04-30','04:00:00','v6',9500,1200,4000.00,'Concert','b2');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

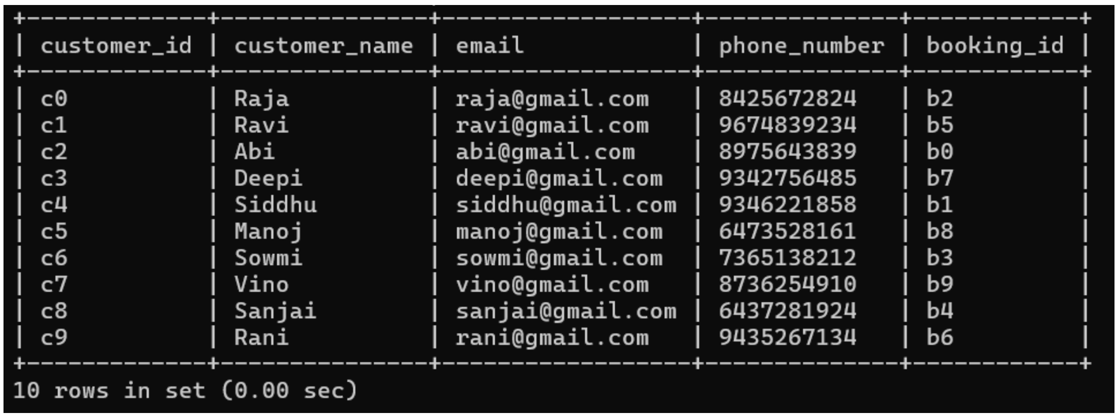


```
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name        | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| e0       | Music concert     | 2024-04-08 | 04:30:00   | v4       |       20000 |            5000 |        10000 | Concert    | b6         |
| e1       | Kayal             | 2024-04-10 | 05:00:00   | v5       |       15000 |            3000 |         7000 | Movie      | b9         |
| e2       | Cricket IPL match | 2024-04-11 | 07:30:00   | v3       |       10000 |            2000 |         3000 | Sports     | b3         |
| e3       | Yearly concert    | 2024-04-14 | 06:00:00   | v8       |        7500 |             500 |         5000 | Concert    | b1         |
| e4       | Friendship        | 2024-04-17 | 09:00:00   | v9       |        4000 |             300 |         2000 | Movie      | b4         |
| e5       | Joe               | 2024-04-23 | 02:30:00   | v0       |        9000 |            1000 |         7000 | Movie      | b8         |
| e6       | Vacation concert  | 2024-04-25 | 06:30:00   | v2       |        5000 |             500 |         5000 | Concert    | b0         |
| e7       | Jungle book       | 2024-04-13 | 10:30:00   | v7       |       14000 |            2000 |         6000 | Movie      | b7         |
| e8       | Football match    | 2024-04-28 | 03:30:00   | v1       |        3000 |             700 |         8000 | Sports     | b5         |
| e9       | Dance Concert     | 2024-04-30 | 04:00:00   | v6       |        9500 |            1200 |         4000 | Concert    | b2         |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
10 rows in set (0.00 sec)
```

Inserting 10 records in the customers table:

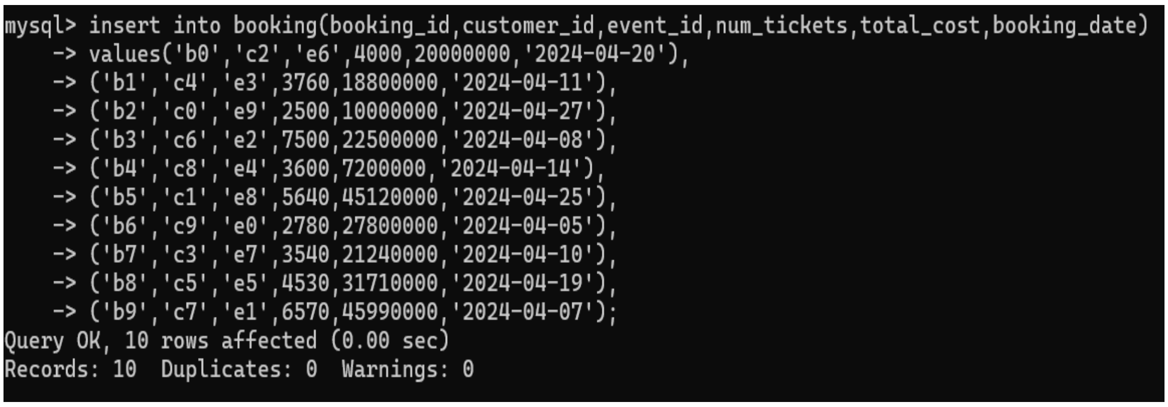


```
mysql> insert into customers(customer_id,customer_name,email,phone_number,booking_id)
    -> values('c0','Raja','raja@gmail.com','8425672824','b2'),
    -> ('c1','Ravi','ravi@gmail.com','9674839234','b5'),
    -> ('c2','Abi','abi@gmail.com','8975643839','b0'),
    -> ('c3','Deepi','deepi@gmail.com','9342756485','b7'),
    -> ('c4','Siddhu','siddhu@gmail.com','9346221858','b1'),
    -> ('c5','Manoj','manoj@gmail.com','6473528161','b8'),
    -> ('c6','Sowmi','sowmi@gmail.com','7365138212','b3'),
    -> ('c7','Vino','vino@gmail.com','8736254910','b9'),
    -> ('c8','Sanjai','sanjai@gmail.com','6437281924','b4'),
    -> ('c9','Rani','rani@gmail.com','9435267134','b6');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

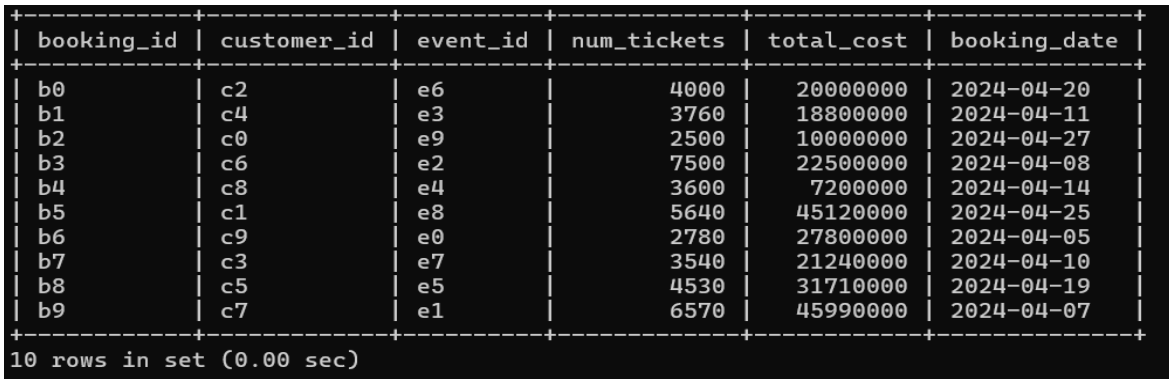


```
+-------------+---------------+------------------+--------------+------------+
| customer_id | customer_name | email            | phone_number | booking_id |
+-------------+---------------+------------------+--------------+------------+
| c0          | Raja          | raja@gmail.com   | 8425672824   | b2         |
| c1          | Ravi          | ravi@gmail.com   | 9674839234   | b5         |
| c2          | Abi           | abi@gmail.com    | 8975643839   | b0         |
| c3          | Deepi         | deepi@gmail.com  | 9342756485   | b7         |
| c4          | Siddhu        | siddhu@gmail.com | 9346221858   | b1         |
| c5          | Manoj         | manoj@gmail.com  | 6473528161   | b8         |
| c6          | Sowmi         | sowmi@gmail.com  | 7365138212   | b3         |
| c7          | Vino          | vino@gmail.com   | 8736254910   | b9         |
| c8          | Sanjai        | sanjai@gmail.com | 6437281924   | b4         |
| c9          | Rani          | rani@gmail.com   | 9435267134   | b6         |
+-------------+---------------+------------------+--------------+------------+
10 rows in set (0.00 sec)
```

Inserting 10 records in the booking table:

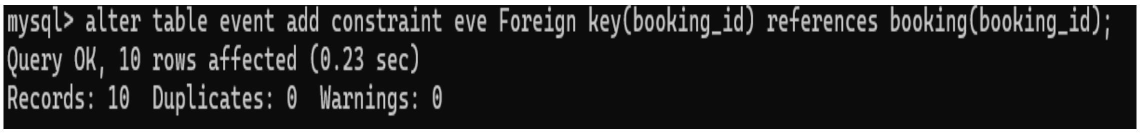


```
mysql> insert into booking(booking_id,customer_id,event_id,num_tickets,total_cost,booking_date)
    -> values('b0','c2','e6',4000,20000000,'2024-04-20'),
    -> ('b1','c4','e3',3760,18800000,'2024-04-11'),
    -> ('b2','c0','e9',2500,10000000,'2024-04-27'),
    -> ('b3','c6','e2',7500,22500000,'2024-04-08'),
    -> ('b4','c8','e4',3600,7200000,'2024-04-14'),
    -> ('b5','c1','e8',5640,45120000,'2024-04-25'),
    -> ('b6','c9','e0',2780,27800000,'2024-04-05'),
    -> ('b7','c3','e7',3540,21240000,'2024-04-10'),
    -> ('b8','c5','e5',4530,31710000,'2024-04-19'),
    -> ('b9','c7','e1',6570,45990000,'2024-04-07');
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

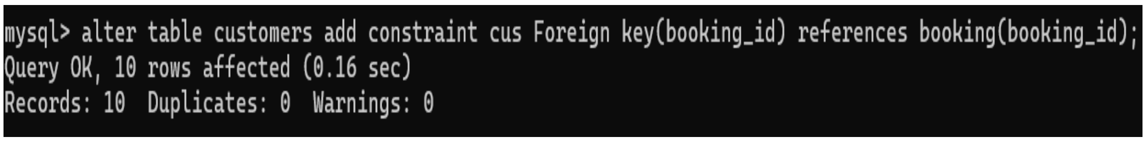


```
+------------+-------------+----------+-------------+------------+--------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
+------------+-------------+----------+-------------+------------+--------------+
| b0         | c2          | e6       |        4000 |   20000000 | 2024-04-20   |
| b1         | c4          | e3       |        3760 |   18800000 | 2024-04-11   |
| b2         | c0          | e9       |        2500 |   10000000 | 2024-04-27   |
| b3         | c6          | e2       |        7500 |   22500000 | 2024-04-08   |
| b4         | c8          | e4       |        3600 |    7200000 | 2024-04-14   |
| b5         | c1          | e8       |        5640 |   45120000 | 2024-04-25   |
| b6         | c9          | e0       |        2780 |   27800000 | 2024-04-05   |
| b7         | c3          | e7       |        3540 |   21240000 | 2024-04-10   |
| b8         | c5          | e5       |        4530 |   31710000 | 2024-04-19   |
| b9         | c7          | e1       |        6570 |   45990000 | 2024-04-07   |
+------------+-------------+----------+-------------+------------+--------------+
10 rows in set (0.00 sec)
```

Adding foreign key constraints:

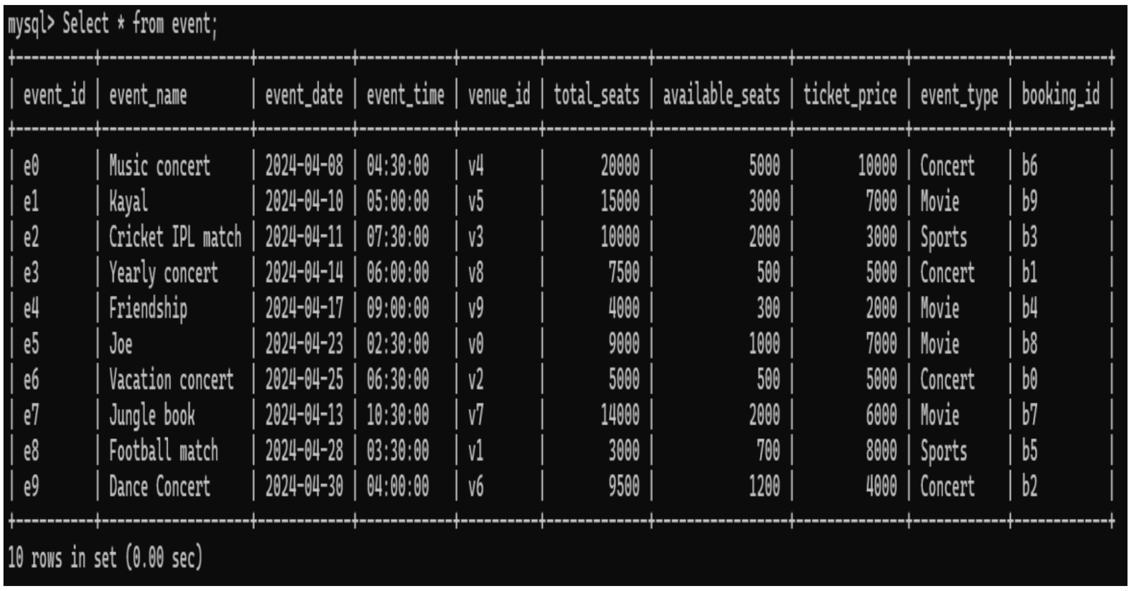


```
mysql> alter table event add constraint eve Foreign key(booking_id) references booking(booking_id);
Query OK, 10 rows affected (0.23 sec)
Records: 10  Duplicates: 0  Warnings: 0
```



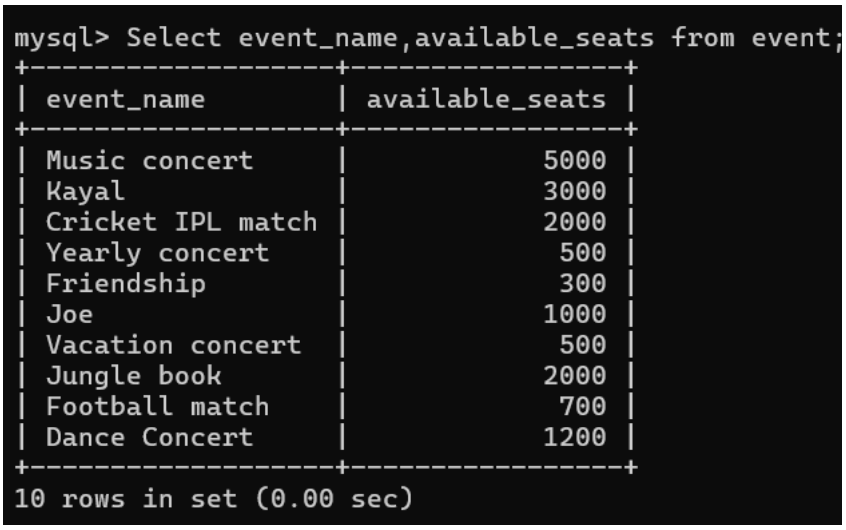
```
mysql> alter table customers add constraint cus Foreign key(booking_id) references booking(booking_id);
Query OK, 10 rows affected (0.16 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

2. SQL query to list all Events:



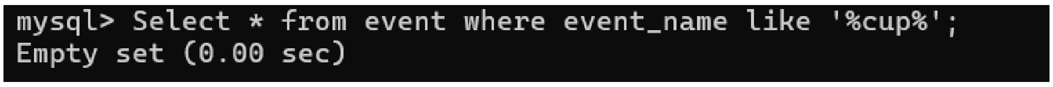
```
mysql> Select * from event;
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name        | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| e0       | Music concert     | 2024-04-08 | 04:30:00   | v4       |       20000 |            5000 |        10000 | Concert    | b6         |
| e1       | Kayal             | 2024-04-10 | 05:00:00   | v5       |       15000 |            3000 |         7000 | Movie      | b9         |
| e2       | Cricket IPL match | 2024-04-11 | 07:30:00   | v3       |       10000 |            2000 |         3000 | Sports     | b3         |
| e3       | Yearly concert    | 2024-04-14 | 06:00:00   | v8       |        7500 |             500 |         5000 | Concert    | b1         |
| e4       | Friendship        | 2024-04-17 | 09:00:00   | v9       |        4000 |             300 |         2000 | Movie      | b4         |
| e5       | Joe               | 2024-04-23 | 02:30:00   | v0       |        9000 |            1000 |         7000 | Movie      | b8         |
| e6       | Vacation concert  | 2024-04-25 | 06:30:00   | v2       |        5000 |             500 |         5000 | Concert    | b0         |
| e7       | Jungle book       | 2024-04-13 | 10:30:00   | v7       |       14000 |            2000 |         6000 | Movie      | b7         |
| e8       | Football match    | 2024-04-28 | 03:30:00   | v1       |        3000 |             700 |         8000 | Sports     | b5         |
| e9       | Dance Concert     | 2024-04-30 | 04:00:00   | v6       |        9500 |            1200 |         4000 | Concert    | b2         |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
10 rows in set (0.00 sec)
```

3. SQL query to select events with available tickets:



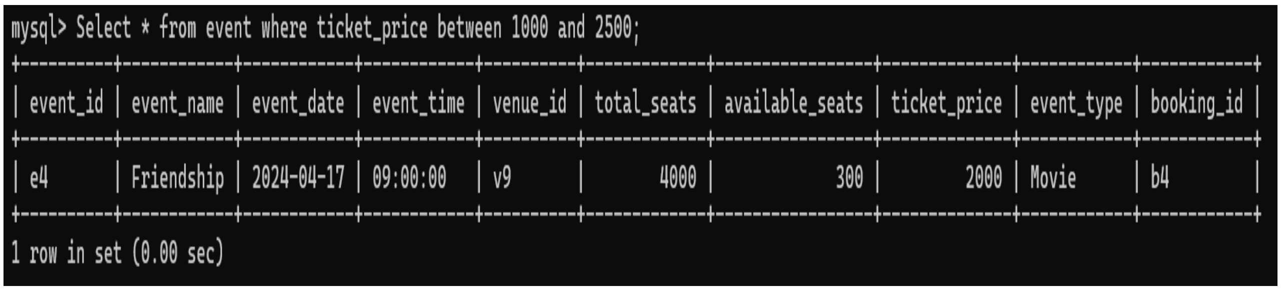
```
mysql> Select event_name,available_seats from event;
+-------------------+-----------------+
| event_name        | available_seats |
+-------------------+-----------------+
| Music concert     |            5000 |
| Kayal             |            3000 |
| Cricket IPL match |            2000 |
| Yearly concert    |             500 |
| Friendship        |             300 |
| Joe               |            1000 |
| Vacation concert  |             500 |
| Jungle book       |            2000 |
| Football match    |             700 |
| Dance Concert     |            1200 |
+-------------------+-----------------+
10 rows in set (0.00 sec)
```

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



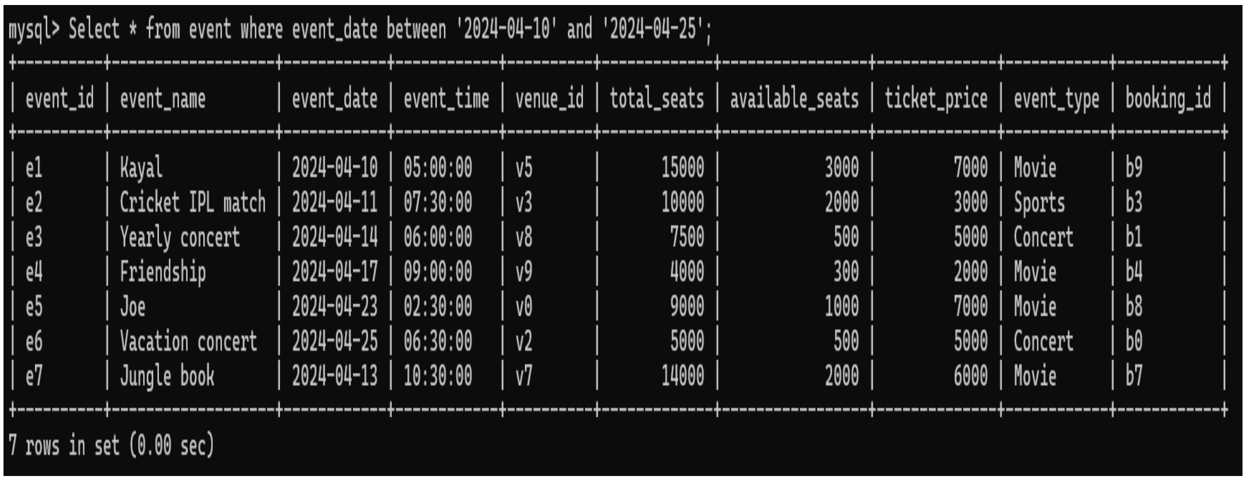
```
mysql> Select * from event where event_name like '%cup%';
Empty set (0.00 sec)
```

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



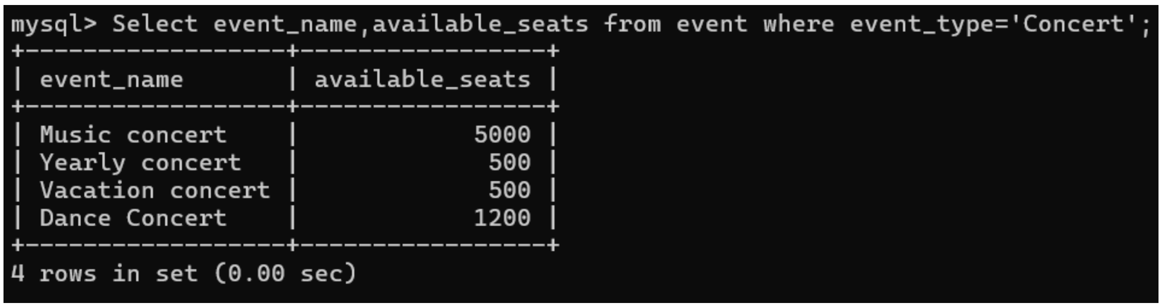
```
mysql> Select * from event where ticket_price between 1000 and 2500;
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| e4       | Friendship | 2024-04-17 | 09:00:00   | v9       |        4000 |             300 |         2000 | Movie      | b4         |
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
1 row in set (0.00 sec)
```

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



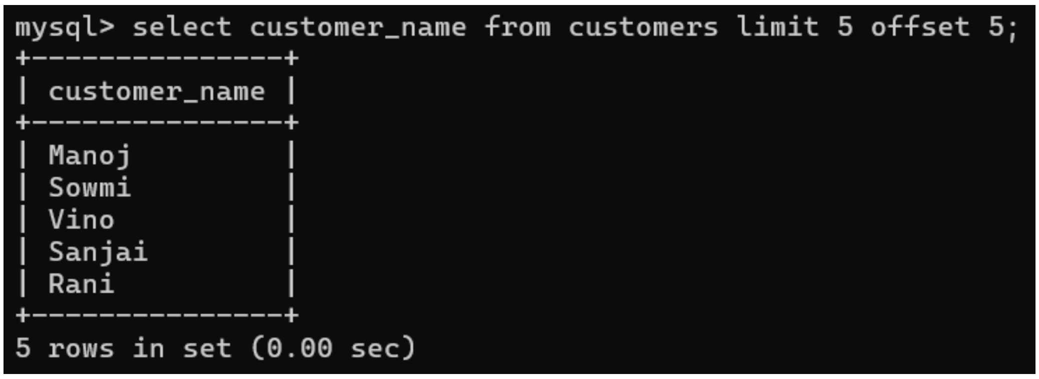
```
mysql> Select * from event where event_date between '2024-04-10' and '2024-04-25';
+----------+------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name       | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| e1       | Kayal            | 2024-04-10 | 05:00:00   | v5       |       15000 |            3000 |         7000 | Movie      | b9         |
| e2       | Cricket IPL match| 2024-04-11 | 07:30:00   | v3       |       10000 |            2000 |         3000 | Sports     | b3         |
| e3       | Yearly concert   | 2024-04-14 | 06:00:00   | v8       |        7500 |             500 |         5000 | Concert    | b1         |
| e4       | Friendship       | 2024-04-17 | 09:00:00   | v9       |        4000 |             300 |         2000 | Movie      | b4         |
| e5       | Joe              | 2024-04-23 | 02:30:00   | v0       |        9000 |            1000 |         7000 | Movie      | b8         |
| e6       | Vacation concert | 2024-04-25 | 06:30:00   | v2       |        5000 |             500 |         5000 | Concert    | b0         |
| e7       | Jungle book      | 2024-04-13 | 10:30:00   | v7       |       14000 |            2000 |         6000 | Movie      | b7         |
+----------+------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
7 rows in set (0.00 sec)
```

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



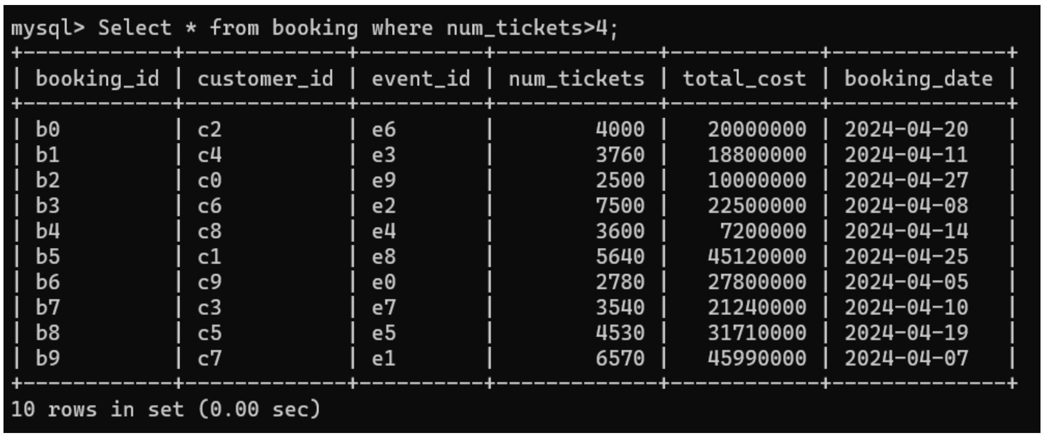
```
mysql> Select event_name,available_seats from event where event_type='Concert';
+------------------+-----------------+
| event_name       | available_seats |
+------------------+-----------------+
| Music concert    |            5000 |
| Yearly concert   |             500 |
| Vacation concert |             500 |
| Dance Concert    |            1200 |
+------------------+-----------------+
4 rows in set (0.00 sec)
```

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



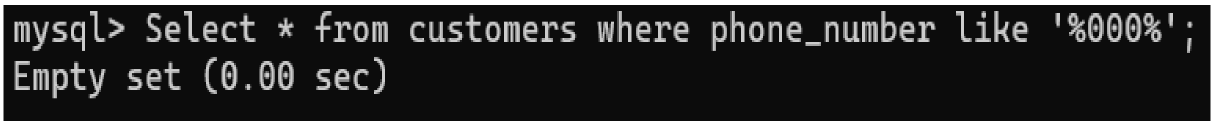
```
mysql> select customer_name from customers limit 5 offset 5;
+---------------+
| customer_name |
+---------------+
| Manoj         |
| Sowmi         |
| Vino          |
| Sanjai        |
| Rani          |
+---------------+
5 rows in set (0.00 sec)
```

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



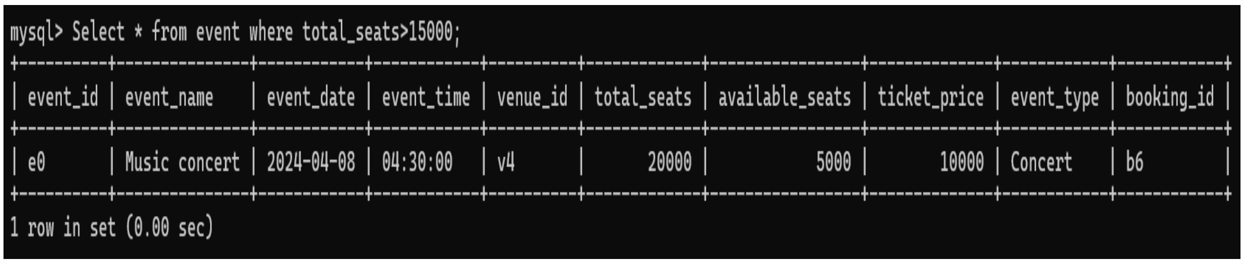
```
mysql> Select * from booking where num_tickets>4;
+------------+-------------+----------+-------------+------------+--------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
+------------+-------------+----------+-------------+------------+--------------+
| b0         | c2          | e6       |        4000 |   20000000 | 2024-04-20   |
| b1         | c4          | e3       |        3760 |   18800000 | 2024-04-11   |
| b2         | c0          | e9       |        2500 |   10000000 | 2024-04-27   |
| b3         | c6          | e2       |        7500 |   22500000 | 2024-04-08   |
| b4         | c8          | e4       |        3600 |    7200000 | 2024-04-14   |
| b5         | c1          | e8       |        5640 |   45120000 | 2024-04-25   |
| b6         | c9          | e0       |        2780 |   27800000 | 2024-04-05   |
| b7         | c3          | e7       |        3540 |   21240000 | 2024-04-10   |
| b8         | c5          | e5       |        4530 |   31710000 | 2024-04-19   |
| b9         | c7          | e1       |        6570 |   45990000 | 2024-04-07   |
+------------+-------------+----------+-------------+------------+--------------+
10 rows in set (0.00 sec)
```

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



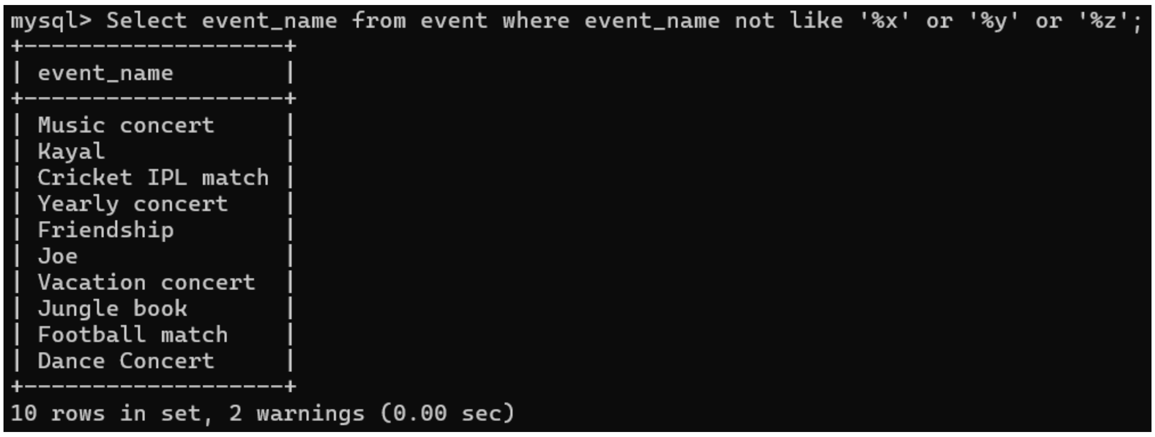
```
mysql> Select * from customers where phone_number like '%000%';
Empty set (0.00 sec)
```

11. SQL query to retrieve the events in order whose seat capacity more than 15000:



```
mysql> Select * from event where total_seats>15000;
+----------+---------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name    | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+---------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| e0       | Music concert | 2024-04-08 | 04:30:00   | v4       |       20000 |            5000 |        10000 | Concert    | b6         |
+----------+---------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
1 row in set (0.00 sec)
```

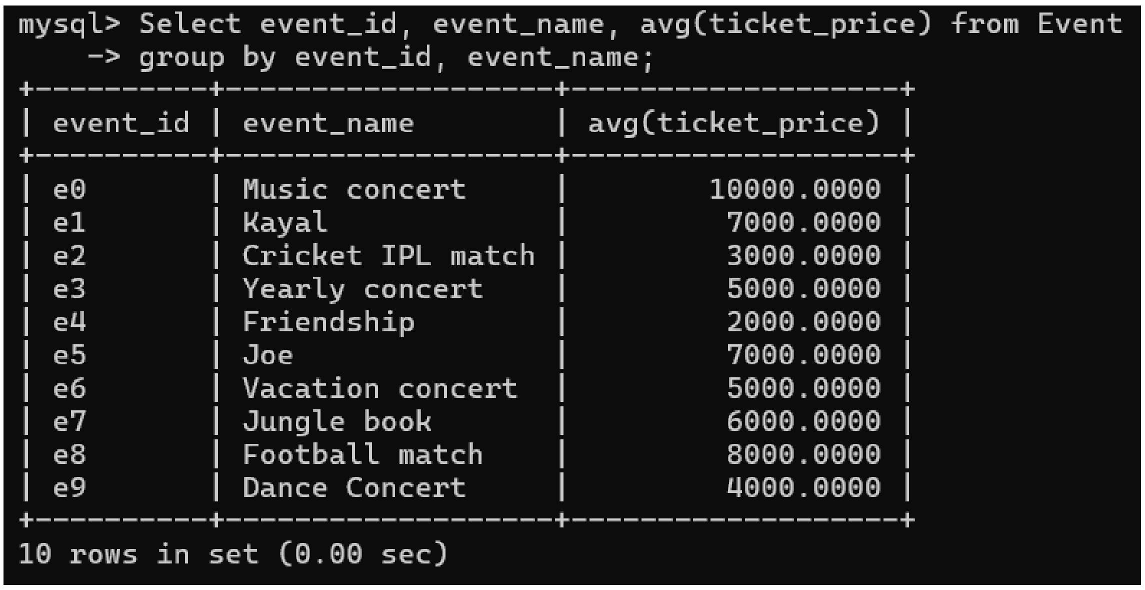
12. SQL query to select events name not start with ‘x’, ‘y’, ‘z’:



```
mysql> Select event_name from event where event_name not like '%x' or '%y' or '%z';
+-------------------+
| event_name        |
+-------------------+
| Music concert     |
| Kayal             |
| Cricket IPL match |
| Yearly concert    |
| Friendship        |
| Joe               |
| Vacation concert  |
| Jungle book       |
| Football match    |
| Dance Concert     |
+-------------------+
10 rows in set, 2 warnings (0.00 sec)
```

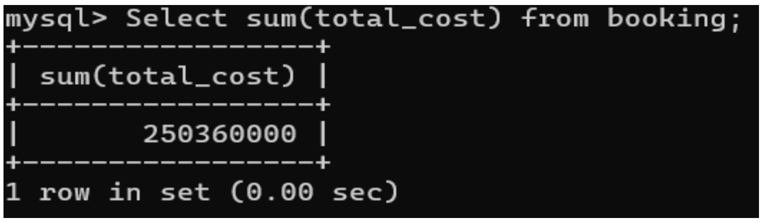
Task 3:

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



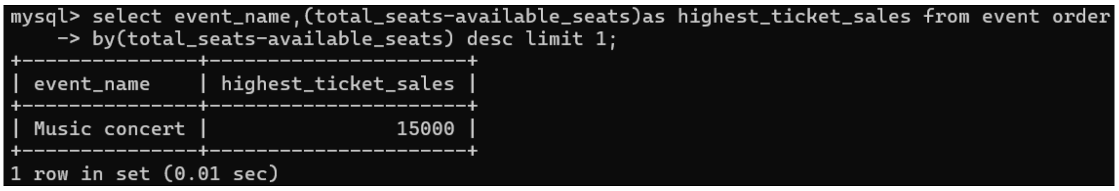
```
mysql> Select event_id, event_name, avg(ticket_price) from Event
    -> group by event_id, event_name;
+----------+-------------------+-------------------+
| event_id | event_name        | avg(ticket_price) |
+----------+-------------------+-------------------+
| e0       | Music concert     |        10000.0000 |
| e1       | Kayal             |         7000.0000 |
| e2       | Cricket IPL match |         3000.0000 |
| e3       | Yearly concert    |         5000.0000 |
| e4       | Friendship        |         2000.0000 |
| e5       | Joe               |         7000.0000 |
| e6       | Vacation concert  |         5000.0000 |
| e7       | Jungle book       |         6000.0000 |
| e8       | Football match    |         8000.0000 |
| e9       | Dance Concert     |         4000.0000 |
+----------+-------------------+-------------------+
10 rows in set (0.00 sec)
```

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



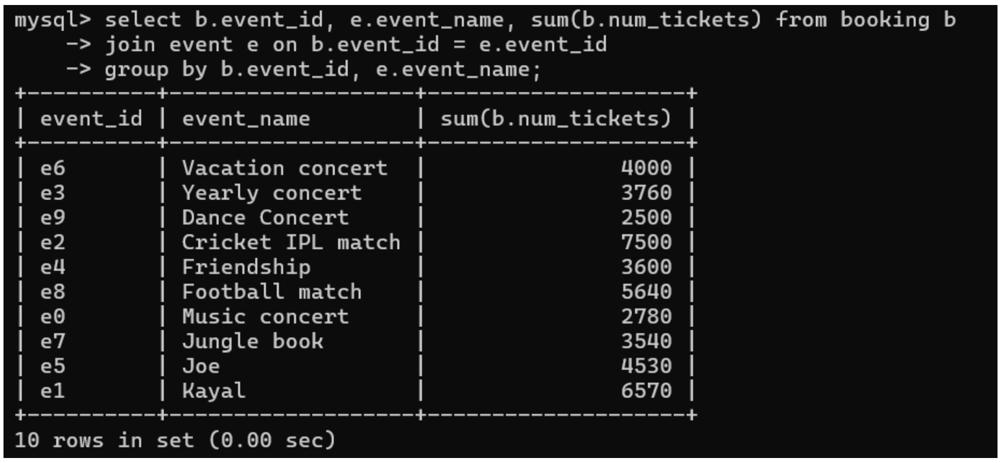
```
mysql> Select sum(total_cost) from booking;
+-----------------+
| sum(total_cost) |
+-----------------+
|       250360000 |
+-----------------+
1 row in set (0.00 sec)
```

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



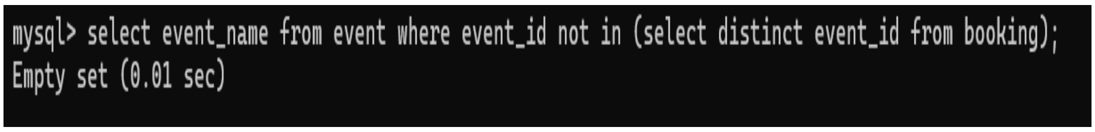
```
mysql> select event_name,(total_seats-available_seats)as highest_ticket_sales from event order
    -> by(total_seats-available_seats) desc limit 1;
+---------------+----------------------+
| event_name    | highest_ticket_sales |
+---------------+----------------------+
| Music concert |                15000 |
+---------------+----------------------+
1 row in set (0.01 sec)
```

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



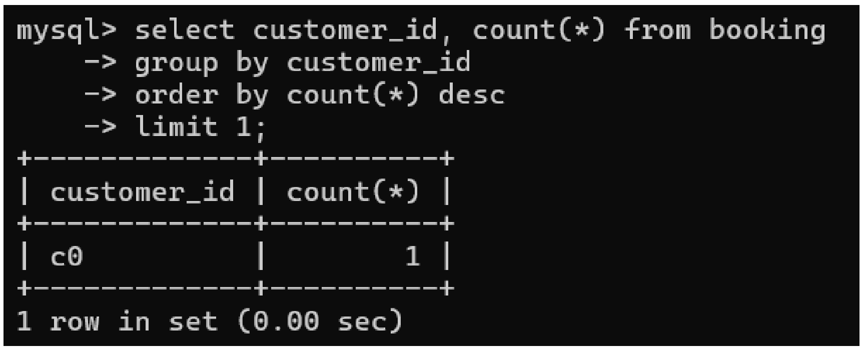
```
mysql> select b.event_id, e.event_name, sum(b.num_tickets) from booking b
    -> join event e on b.event_id = e.event_id
    -> group by b.event_id, e.event_name;
+----------+------------------+--------------------+
| event_id | event_name       | sum(b.num_tickets) |
+----------+------------------+--------------------+
| e6       | Vacation concert |               4000 |
| e3       | Yearly concert   |               3760 |
| e9       | Dance Concert    |               2500 |
| e2       | Cricket IPL match|               7500 |
| e4       | Friendship       |               3600 |
| e8       | Football match   |               5640 |
| e0       | Music concert    |               2780 |
| e7       | Jungle book      |               3540 |
| e5       | Joe              |               4530 |
| e1       | Kayal            |               6570 |
+----------+------------------+--------------------+
10 rows in set (0.00 sec)
```

5. SQL query to Find Events with No Ticket Sales:



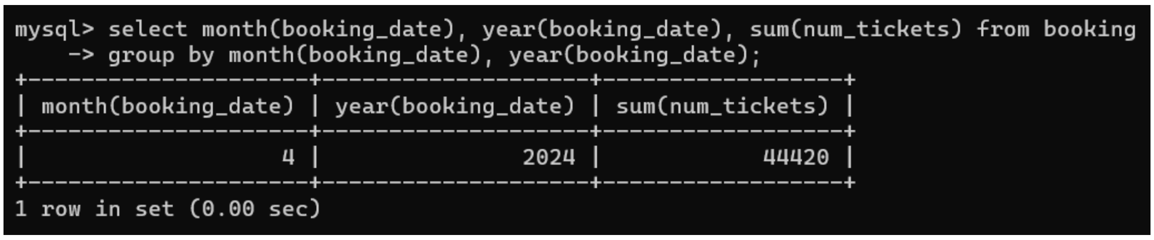
```
mysql> select event_name from event where event_id not in (select distinct event_id from booking);
Empty set (0.01 sec)
```

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



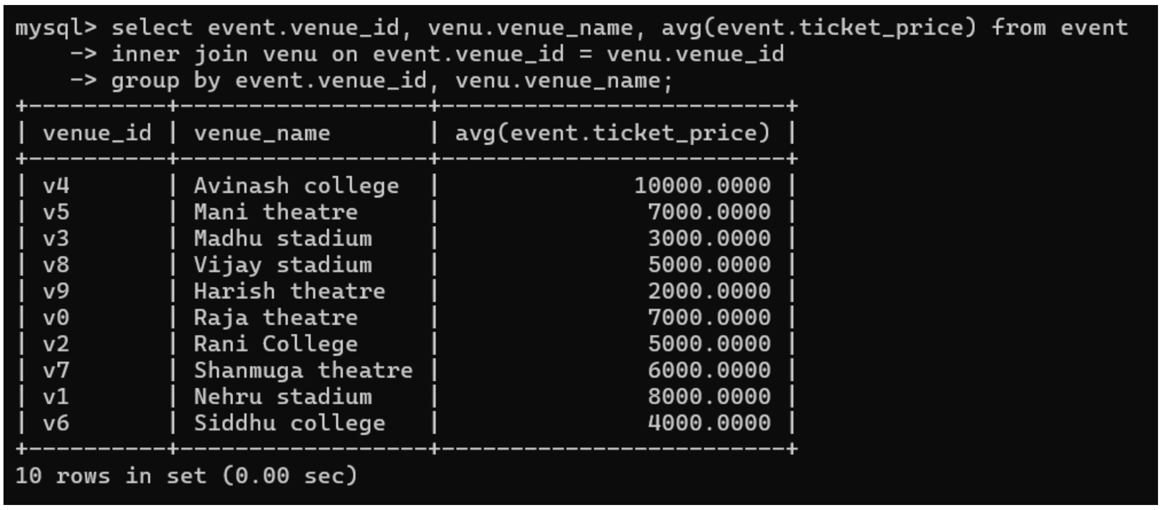
```
mysql> select customer_id, count(*) from booking
    -> group by customer_id
    -> order by count(*) desc
    -> limit 1;
+-------------+----------+
| customer_id | count(*) |
+-------------+----------+
| c0          |        1 |
+-------------+----------+
1 row in set (0.00 sec)
```

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



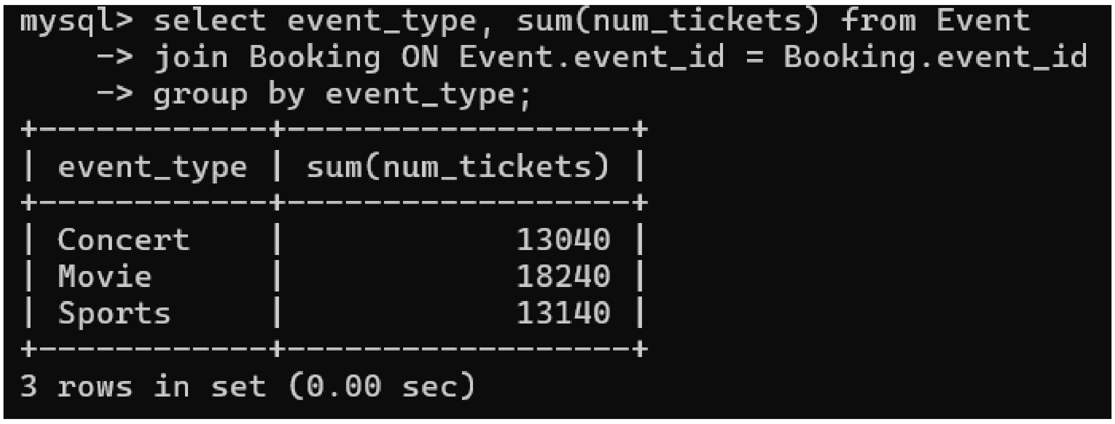
```
mysql> select month(booking_date), year(booking_date), sum(num_tickets) from booking
    -> group by month(booking_date), year(booking_date);
+---------------------+--------------------+------------------+
| month(booking_date) | year(booking_date) | sum(num_tickets) |
+---------------------+--------------------+------------------+
|                   4 |               2024 |            44420 |
+---------------------+--------------------+------------------+
1 row in set (0.00 sec)
```

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



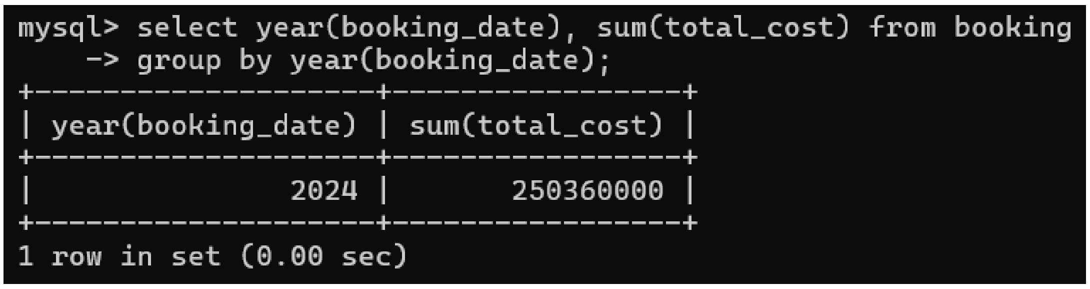
```
mysql> select event.venue_id, venu.venue_name, avg(event.ticket_price) from event
    -> inner join venu on event.venue_id = venu.venue_id
    -> group by event.venue_id, venu.venue_name;
+----------+-----------------+-------------------------+
| venue_id | venue_name      | avg(event.ticket_price) |
+----------+-----------------+-------------------------+
| v4       | Avinash college |              10000.0000 |
| v5       | Mani theatre    |               7000.0000 |
| v3       | Madhu stadium   |               3000.0000 |
| v8       | Vijay stadium   |               5000.0000 |
| v9       | Harish theatre  |               2000.0000 |
| v0       | Raja theatre    |               7000.0000 |
| v2       | Rani College    |               5000.0000 |
| v7       | Shanmuga theatre|               6000.0000 |
| v1       | Nehru stadium   |               8000.0000 |
| v6       | Siddhu college  |               4000.0000 |
+----------+-----------------+-------------------------+
10 rows in set (0.00 sec)
```

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



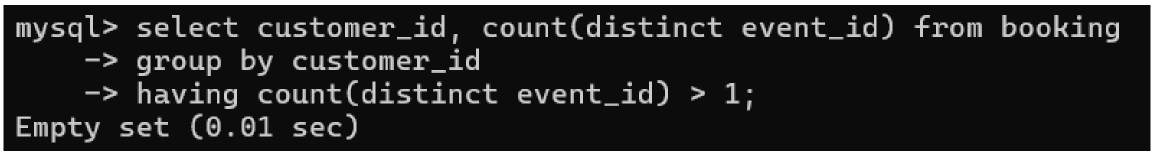
```
mysql> select event_type, sum(num_tickets) from Event
    -> join Booking ON Event.event_id = Booking.event_id
    -> group by event_type;
+------------+------------------+
| event_type | sum(num_tickets) |
+------------+------------------+
| Concert    |            13040 |
| Movie      |            18240 |
| Sports     |            13140 |
+------------+------------------+
3 rows in set (0.00 sec)
```

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



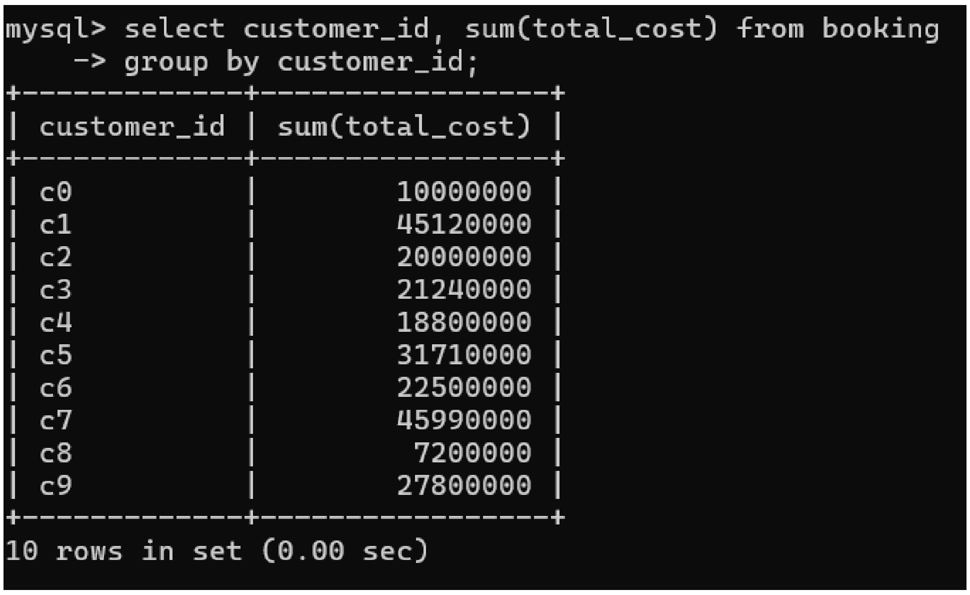
```
mysql> select year(booking_date), sum(total_cost) from booking
    -> group by year(booking_date);
+--------------------+-----------------+
| year(booking_date) | sum(total_cost) |
+--------------------+-----------------+
|               2024 |       250360000 |
+--------------------+-----------------+
1 row in set (0.00 sec)
```

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



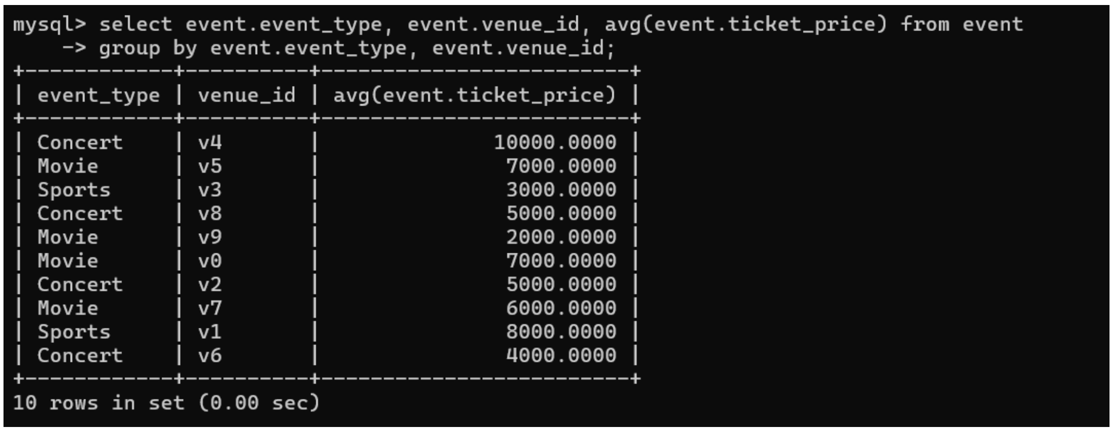
```
mysql> select customer_id, count(distinct event_id) from booking
    -> group by customer_id
    -> having count(distinct event_id) > 1;
Empty set (0.01 sec)
```

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



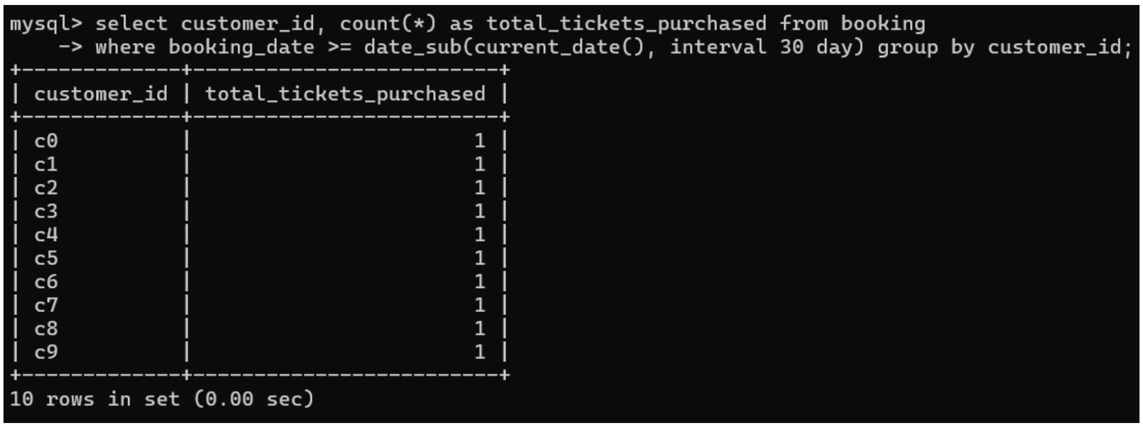
```
mysql> select customer_id, sum(total_cost) from booking
    -> group by customer_id;
+-------------+-----------------+
| customer_id | sum(total_cost) |
+-------------+-----------------+
| c0          |        10000000 |
| c1          |        45120000 |
| c2          |        20000000 |
| c3          |        21240000 |
| c4          |        18800000 |
| c5          |        31710000 |
| c6          |        22500000 |
| c7          |        45990000 |
| c8          |         7200000 |
| c9          |        27800000 |
+-------------+-----------------+
10 rows in set (0.00 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    | v4       |              10000.0000 |
| Movie      | v5       |               7000.0000 |
| Sports     | v3       |               3000.0000 |
| Concert    | v8       |               5000.0000 |
| Movie      | v9       |               2000.0000 |
| Movie      | v0       |               7000.0000 |
| Concert    | v2       |               5000.0000 |
| Movie      | v7       |               6000.0000 |
| Sports     | v1       |               8000.0000 |
| Concert    | v6       |               4000.0000 |
+------------+----------+-------------------------+
10 rows in set (0.00 sec)
```

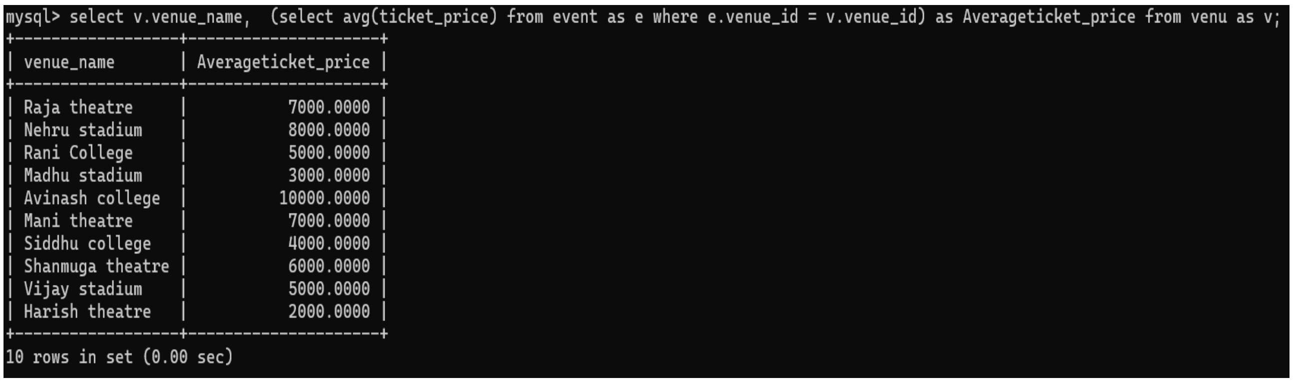
14. SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days:



```
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 |
+-------------+-------------------------+
| c0          |                       1 |
| c1          |                       1 |
| c2          |                       1 |
| c3          |                       1 |
| c4          |                       1 |
| c5          |                       1 |
| c6          |                       1 |
| c7          |                       1 |
| c8          |                       1 |
| c9          |                       1 |
+-------------+-------------------------+
10 rows in set (0.00 sec)
```

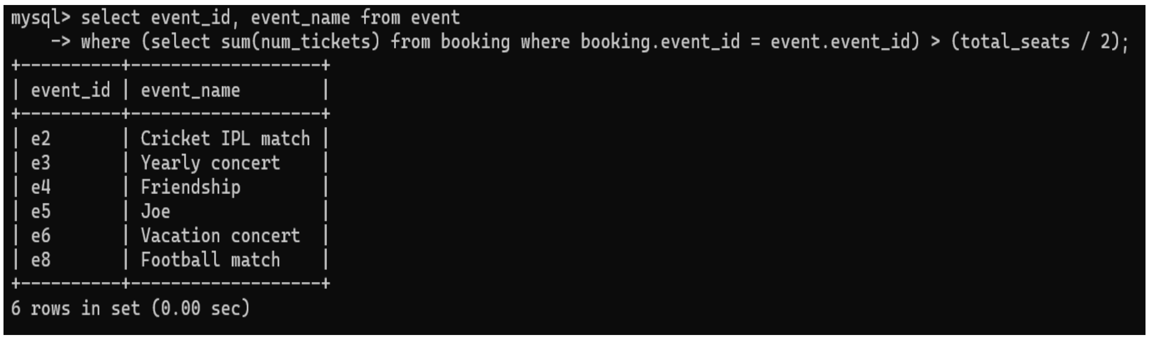
Task 4:

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



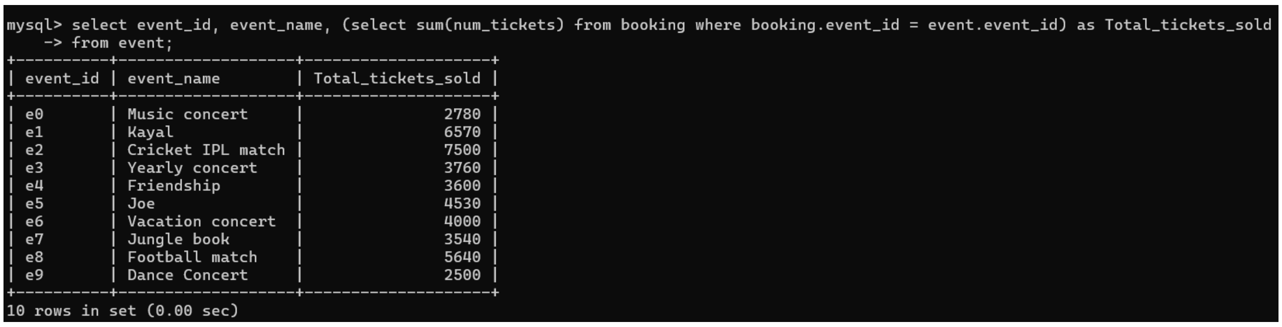
```
mysql> select v.venue_name,  (select avg(ticket_price) from event as e where e.venue_id = v.venue_id) as Averageticket_price from venu as v;
+-----------------+---------------------+
| venue_name      | Averageticket_price |
+-----------------+---------------------+
| Raja theatre    |           7000.0000 |
| Nehru stadium   |           8000.0000 |
| Rani College    |           5000.0000 |
| Madhu stadium   |           3000.0000 |
| Avinash college |          10000.0000 |
| Mani theatre    |           7000.0000 |
| Siddhu college  |           4000.0000 |
| Shanmuga theatre |          6000.0000 |
| Vijay stadium   |           5000.0000 |
| Harish theatre  |           2000.0000 |
+-----------------+---------------------+
10 rows in set (0.00 sec)
```

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



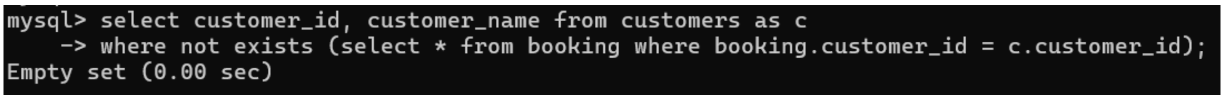
```
mysql> select event_id, event_name from event
    -> where (select sum(num_tickets) from booking where booking.event_id = event.event_id) > (total_seats / 2);
+----------+-------------------+
| event_id | event_name        |
+----------+-------------------+
| e2       | Cricket IPL match |
| e3       | Yearly concert    |
| e4       | Friendship        |
| e5       | Joe               |
| e6       | Vacation concert  |
| e8       | Football match    |
+----------+-------------------+
6 rows in set (0.00 sec)
```

3.Total Number of Tickets Sold for Each Event:



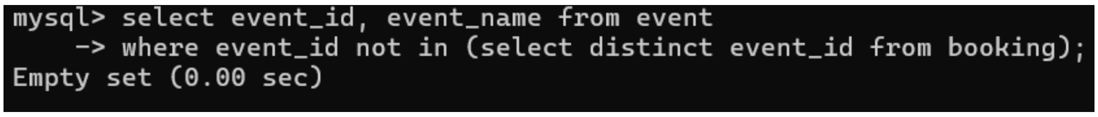
```
mysql> select event_id, event_name, (select sum(num_tickets) from booking where booking.event_id = event.event_id) as Total_tickets_sold
    -> from event;
+----------+-------------------+--------------------+
| event_id | event_name        | Total_tickets_sold |
+----------+-------------------+--------------------+
| e0       | Music concert     |               2780 |
| e1       | Kayal             |               6570 |
| e2       | Cricket IPL match |               7500 |
| e3       | Yearly concert    |               3760 |
| e4       | Friendship        |               3600 |
| e5       | Joe               |               4530 |
| e6       | Vacation concert  |               4000 |
| e7       | Jungle book       |               3540 |
| e8       | Football match    |               5640 |
| e9       | Dance Concert     |               2500 |
+----------+-------------------+--------------------+
10 rows in set (0.00 sec)
```

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



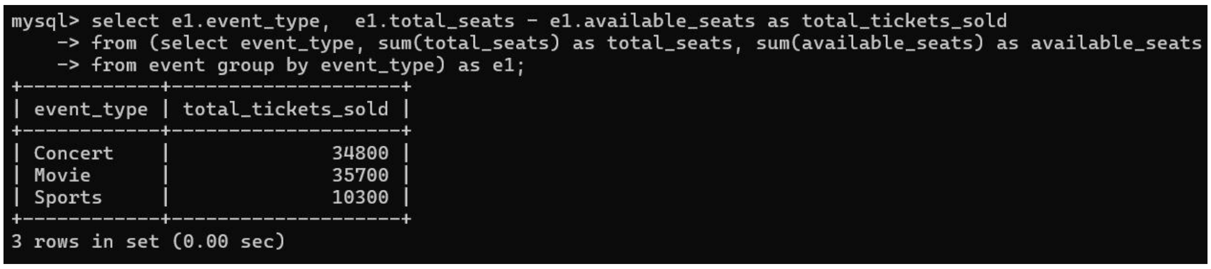
```
mysql> select customer_id, customer_name from customers as c
    -> where not exists (select * from booking where booking.customer_id = c.customer_id);
Empty set (0.00 sec)
```

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



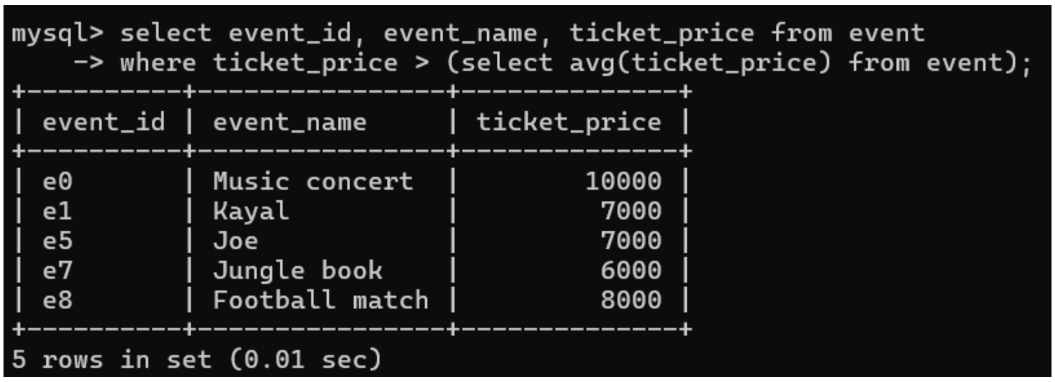
```
mysql> select event_id, event_name from event
    -> where event_id not in (select distinct event_id from booking);
Empty set (0.00 sec)
```

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



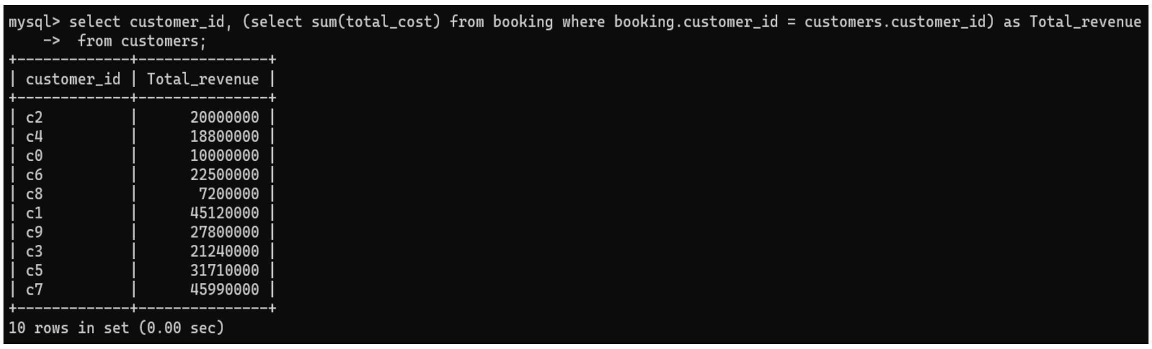
```
mysql> select e1.event_type,  e1.total_seats - e1.available_seats as total_tickets_sold
    -> from (select event_type, sum(total_seats) as total_seats, sum(available_seats) as available_seats
    -> from event group by event_type) as e1;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Concert    |              34800 |
| Movie      |              35700 |
| Sports     |              10300 |
+------------+--------------------+
3 rows in set (0.00 sec)
```

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



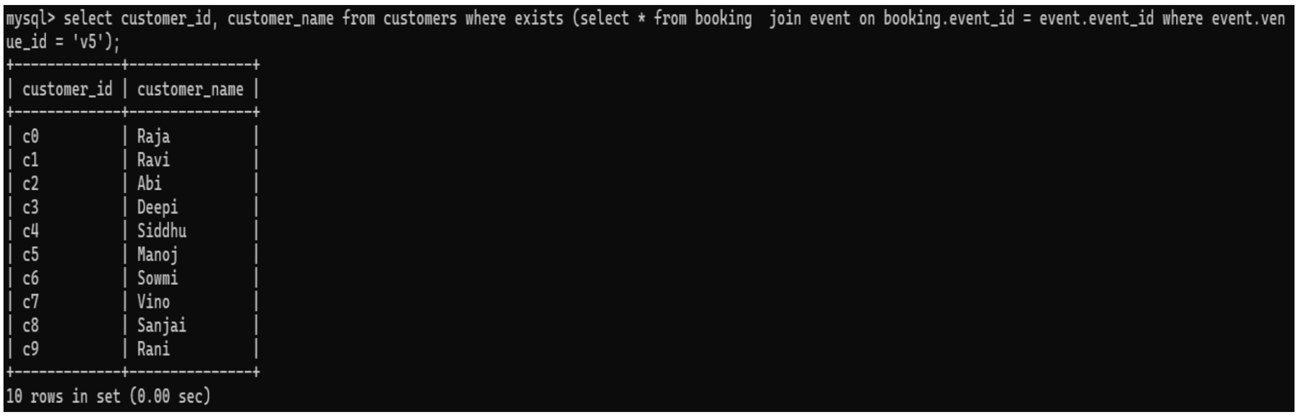
```
mysql> select event_id, event_name, ticket_price from event
    -> where ticket_price > (select avg(ticket_price) from event);
+----------+----------------+--------------+
| event_id | event_name     | ticket_price |
+----------+----------------+--------------+
| e0       | Music concert  |        10000 |
| e1       | Kayal          |         7000 |
| e5       | Joe            |         7000 |
| e7       | Jungle book    |         6000 |
| e8       | Football match |         8000 |
+----------+----------------+--------------+
5 rows in set (0.01 sec)
```

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



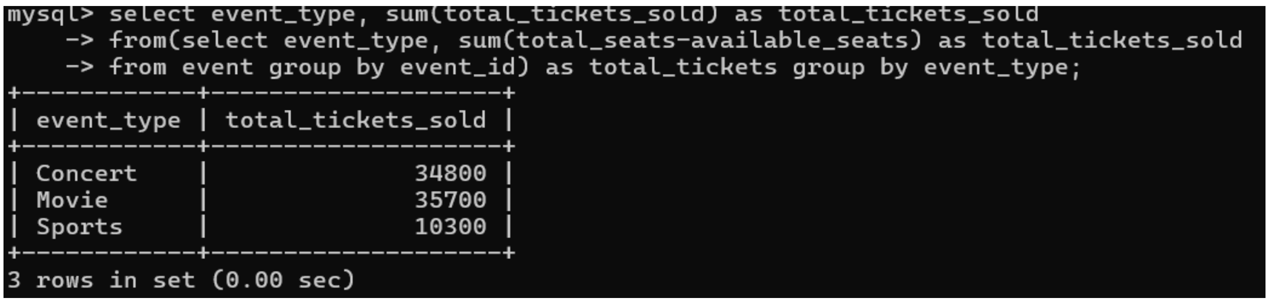
```
mysql> select customer_id, (select sum(total_cost) from booking where booking.customer_id = customers.customer_id) as Total_revenue
    ->  from customers;
+-------------+---------------+
| customer_id | Total_revenue |
+-------------+---------------+
| c2          |      20000000 |
| c4          |      18800000 |
| c0          |      10000000 |
| c6          |      22500000 |
| c8          |       7200000 |
| c1          |      45120000 |
| c9          |      27800000 |
| c3          |      21240000 |
| c5          |      31710000 |
| c7          |      45990000 |
+-------------+---------------+
10 rows in set (0.00 sec)
```

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



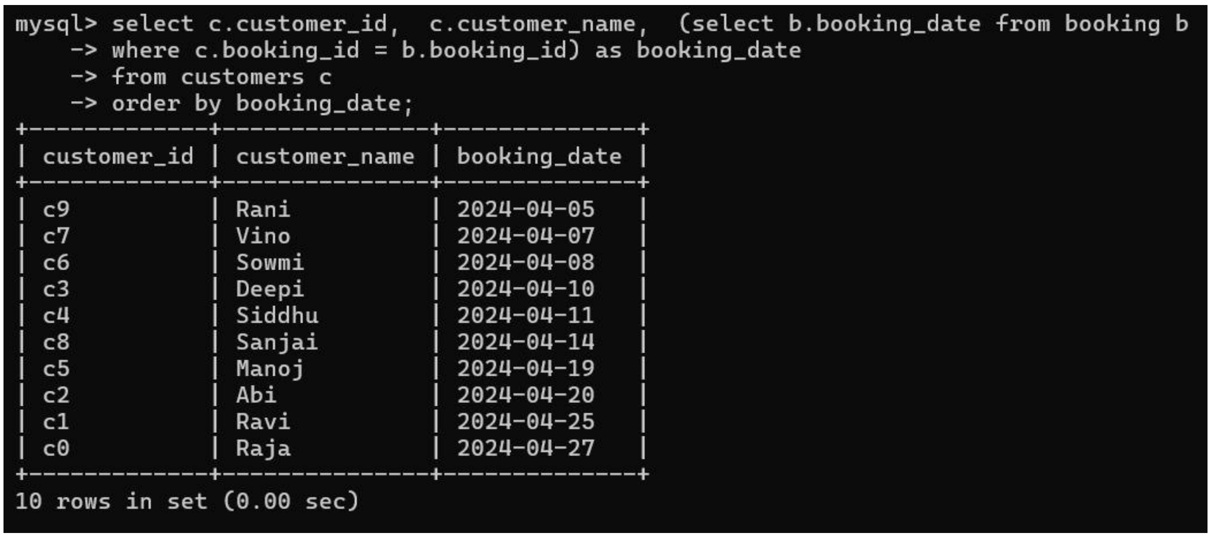
```
mysql> select customer_id, customer_name from customers where exists (select * from booking  join event on booking.event_id = event.event_id where event.ven
ue_id = 'v5');
+-------------+---------------+
| customer_id | customer_name |
+-------------+---------------+
| c0          | Raja          |
| c1          | Ravi          |
| c2          | Abi           |
| c3          | Deepi         |
| c4          | Siddhu        |
| c5          | Manoj         |
| c6          | Sowmi         |
| c7          | Vino          |
| c8          | Sanjai        |
| c9          | Rani          |
+-------------+---------------+
10 rows in set (0.00 sec)
```

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



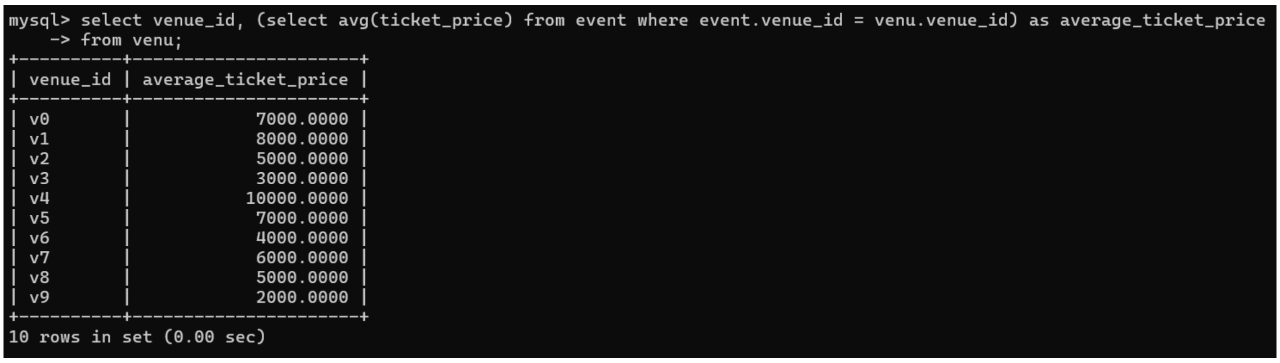
```
mysql> select event_type, sum(total_tickets_sold) as total_tickets_sold
    -> from(select event_type, sum(total_seats-available_seats) as total_tickets_sold
    -> from event group by event_id) as total_tickets group by event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Concert    |              34800 |
| Movie      |              35700 |
| Sports     |              10300 |
+------------+--------------------+
3 rows in set (0.00 sec)
```

11. 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. Average Ticket Price for Events in Each Venue Using a Subquery:



```
mysql> select venue_id, (select avg(ticket_price) from event where event.venue_id = venu.venue_id) as average_ticket_price
    -> from venu;
+----------+----------------------+
| venue_id | average_ticket_price |
+----------+----------------------+
| v0       |            7000.0000 |
| v1       |            8000.0000 |
| v2       |            5000.0000 |
| v3       |            3000.0000 |
| v4       |           10000.0000 |
| v5       |            7000.0000 |
| v6       |            4000.0000 |
| v7       |            6000.0000 |
| v8       |            5000.0000 |
| v9       |            2000.0000 |
+----------+----------------------+
10 rows in set (0.00 sec)
```