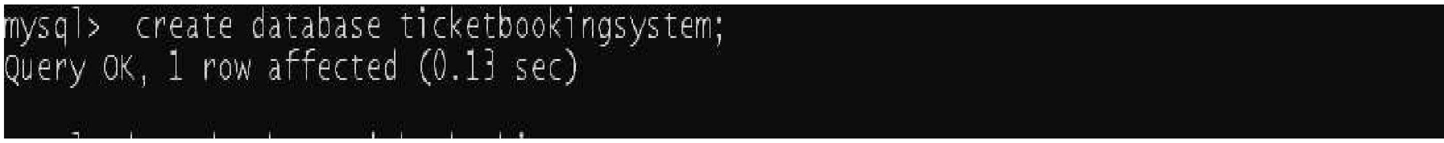
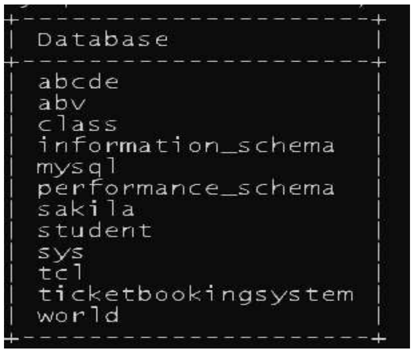
TICKET BOOKING SYSTEM

Tasks 1: Database Design:

1. Create the database named "TicketBookingSystem"



```
mysql>  create database ticketbookingsystem;
Query OK, 1 row affected (0.13 sec)
```



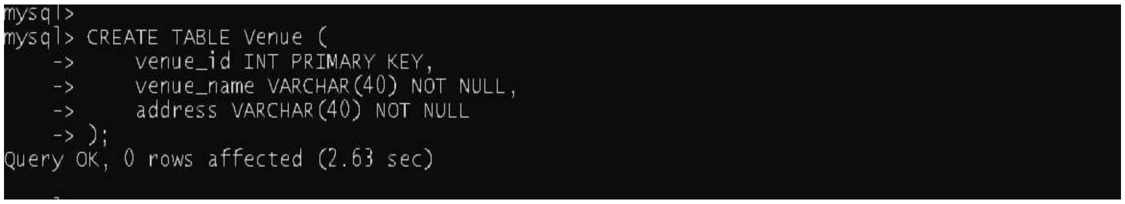
```
+--------------------+
| Database           |
+--------------------+
| abcde              |
| abv                |
| class              |
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| student            |
| sys                |
| tcl                |
| ticketbookingsystem |
| world              |
+--------------------+
```

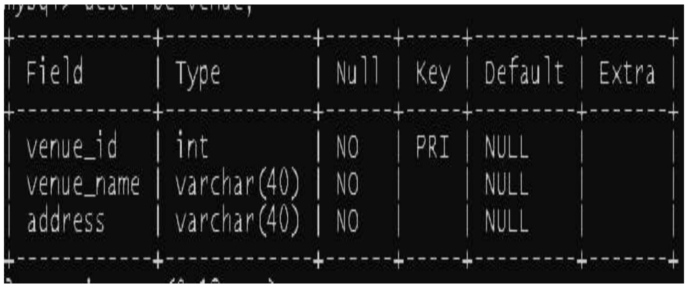
2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

- Venu
- Event
- Customers
- Booking

1. Venu Table

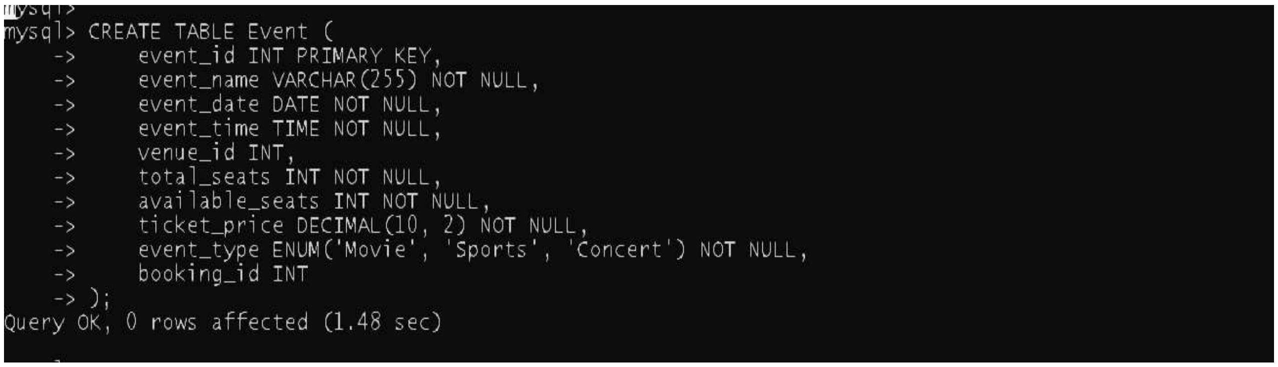
- venue_id (Primary Key)
- venue_name,
- address

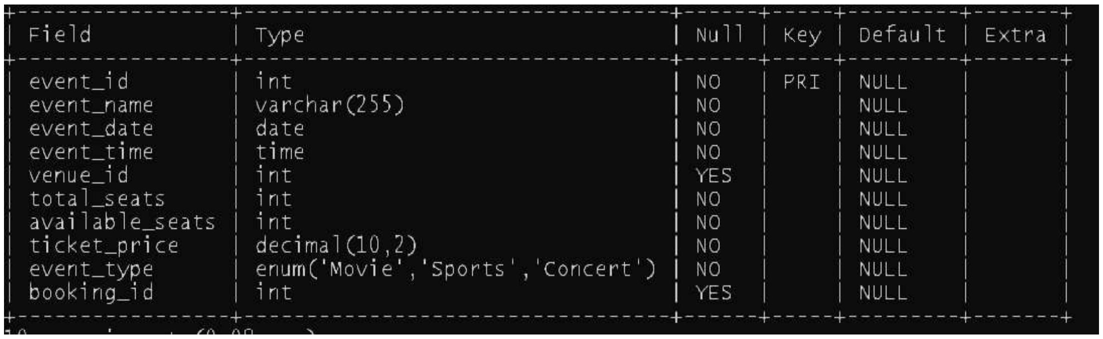

```
mysql>
mysql> CREATE TABLE Venue (
    ->     venue_id INT PRIMARY KEY,
    ->     venue_name VARCHAR(40) NOT NULL,
    ->     address VARCHAR(40) NOT NULL
    -> );
Query OK, 0 rows affected (2.63 sec)
```


```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| venue_id   | int         | NO   | PRI | NULL    |       |
| venue_name | varchar(40) | NO   |     | NULL    |       |
| address    | varchar(40) | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

2. Event Table

- event_id (Primary Key)
- event_name,
- event_date DATE,
- event_time TIME,
- venue_id (Foreign Key),
- total_seats,
- available_seats,
- ticket_price DECIMAL,
- event_type ('Movie', 'Sports', 'Concert')
- booking_id (Foreign Key)


```
mysql>
mysql> CREATE TABLE Event (
    ->     event_id INT PRIMARY KEY,
    ->     event_name VARCHAR(255) NOT NULL,
    ->     event_date DATE NOT NULL,
    ->     event_time TIME NOT NULL,
    ->     venue_id INT,
    ->     total_seats INT NOT NULL,
    ->     available_seats INT NOT NULL,
    ->     ticket_price DECIMAL(10, 2) NOT NULL,
    ->     event_type ENUM('Movie', 'Sports', 'Concert') NOT NULL,
    ->     booking_id INT
    -> );
Query OK, 0 rows affected (1.48 sec)
```



```
+-----------------+-----------------------------------+------+-----+---------+-------+
| Field           | Type                              | Null | Key | Default | Extra |
+-----------------+-----------------------------------+------+-----+---------+-------+
| event_id        | int                               | NO   | PRI | NULL    |       |
| event_name      | varchar(255)                      | NO   |     | NULL    |       |
| event_date      | date                              | NO   |     | NULL    |       |
| event_time      | time                              | NO   |     | NULL    |       |
| venue_id        | int                               | YES  |     | NULL    |       |
| total_seats     | int                               | NO   |     | NULL    |       |
| available_seats | int                               | NO   |     | NULL    |       |
| ticket_price    | decimal(10,2)                     | NO   |     | NULL    |       |
| event_type      | enum('Movie','Sports','Concert')  | NO   |     | NULL    |       |
| booking_id      | int                               | YES  |     | NULL    |       |
+-----------------+-----------------------------------+------+-----+---------+-------+
```

3. Customer Table

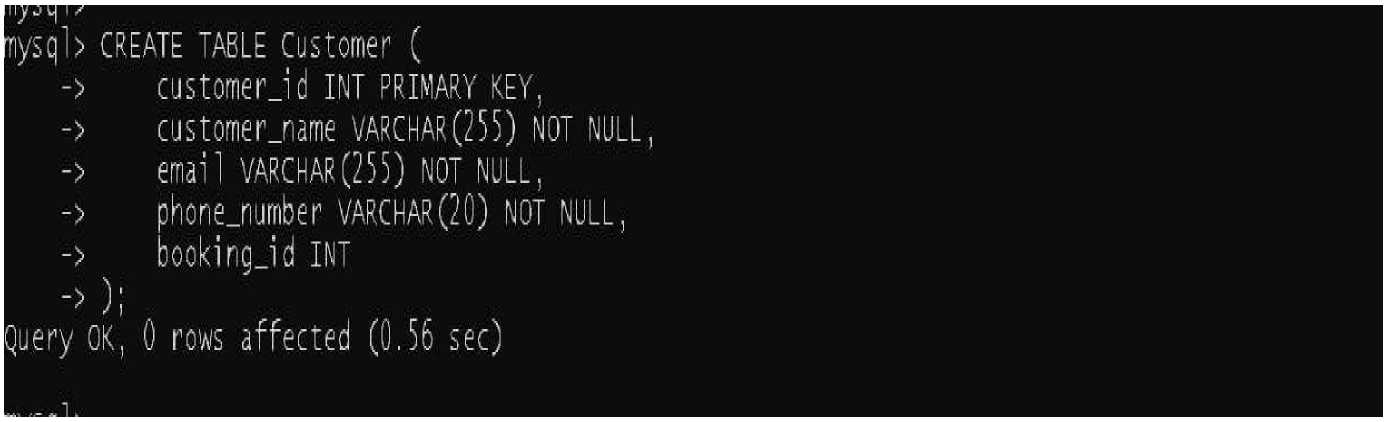
• customer_id (Primary key)

• customer_name,

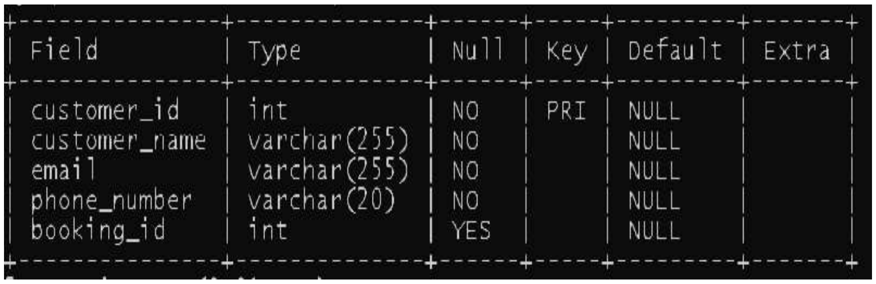
• email,

• phone_number,

• booking_id (Foreign Key)



```
mysql> CREATE TABLE Customer (
    ->     customer_id INT PRIMARY KEY,
    ->     customer_name VARCHAR(255) NOT NULL,
    ->     email VARCHAR(255) NOT NULL,
    ->     phone_number VARCHAR(20) NOT NULL,
    ->     booking_id INT
    -> );
Query OK, 0 rows affected (0.56 sec)
```



```
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| customer_id   | int          | NO   | PRI | NULL    |       |
| customer_name | varchar(255) | NO   |     | NULL    |       |
| email         | varchar(255) | NO   |     | NULL    |       |
| phone_number  | varchar(20)  | NO   |     | NULL    |       |
| booking_id    | int          | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
```

4. Booking Table

• booking_id (Primary Key),

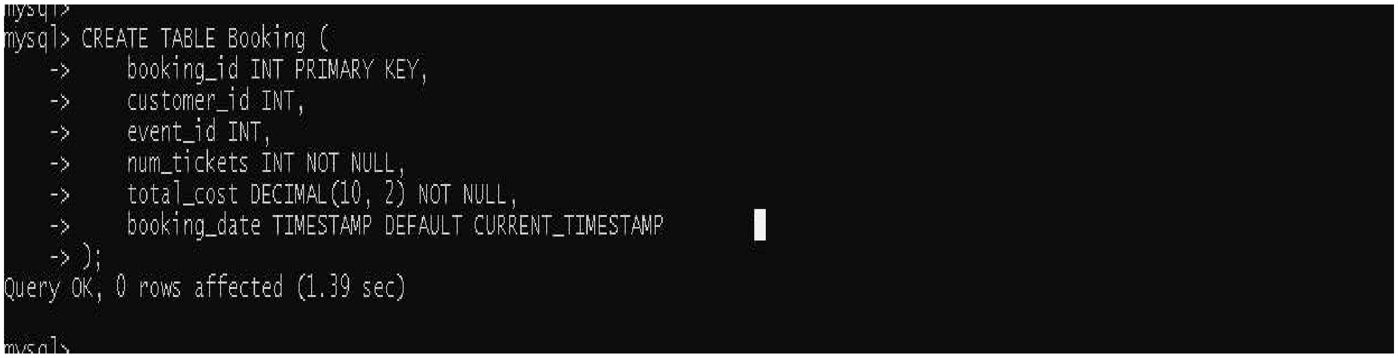
• customer_id (Foreign Key),

• event_id (Foreign Key),

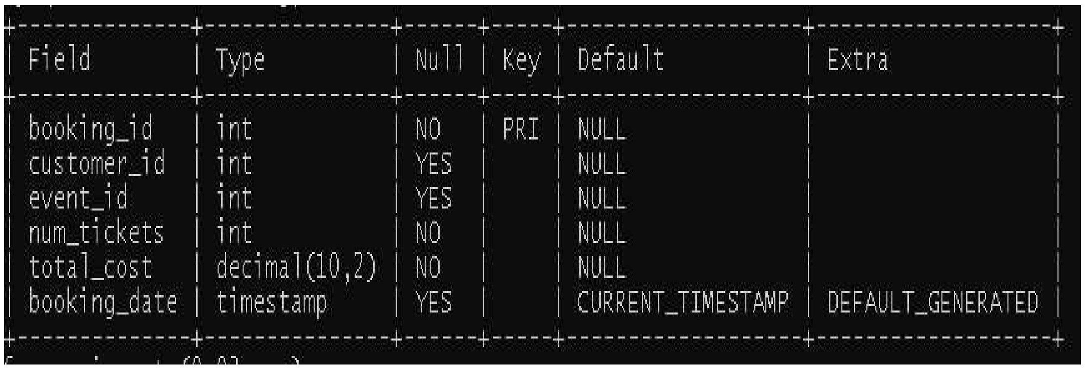
• num_tickets,

• total_cost,

• booking_date

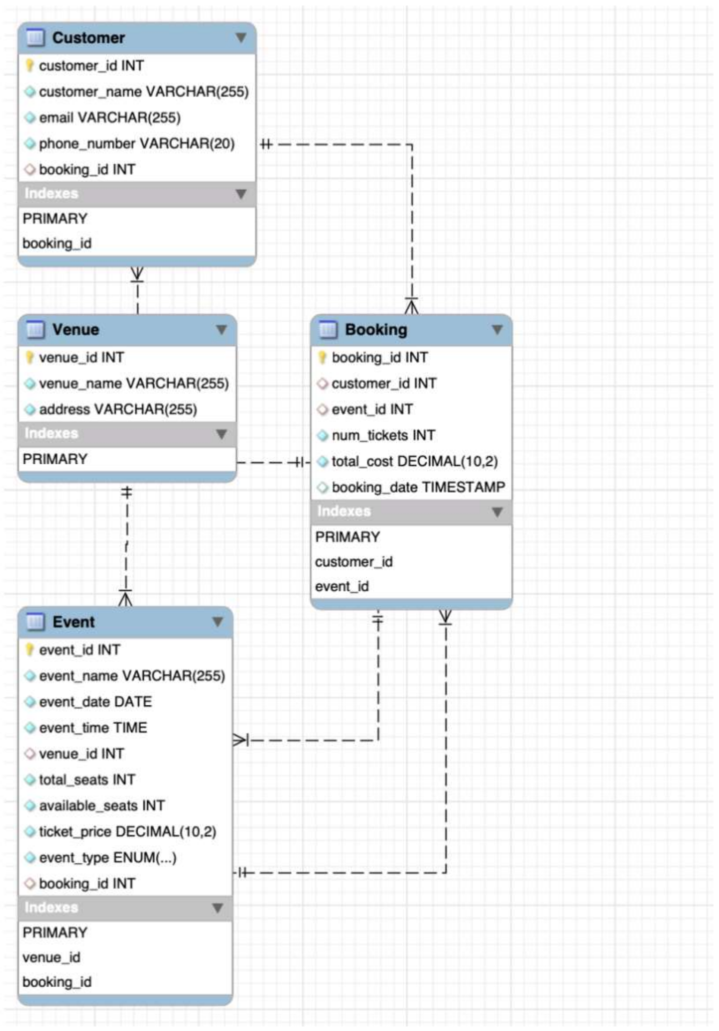


```
mysql>
mysql> CREATE TABLE Booking (
    ->     booking_id INT PRIMARY KEY,
    ->     customer_id INT,
    ->     event_id INT,
    ->     num_tickets INT NOT NULL,
    ->     total_cost DECIMAL(10, 2) NOT NULL,
    ->     booking_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (1.39 sec)

mysql>
```

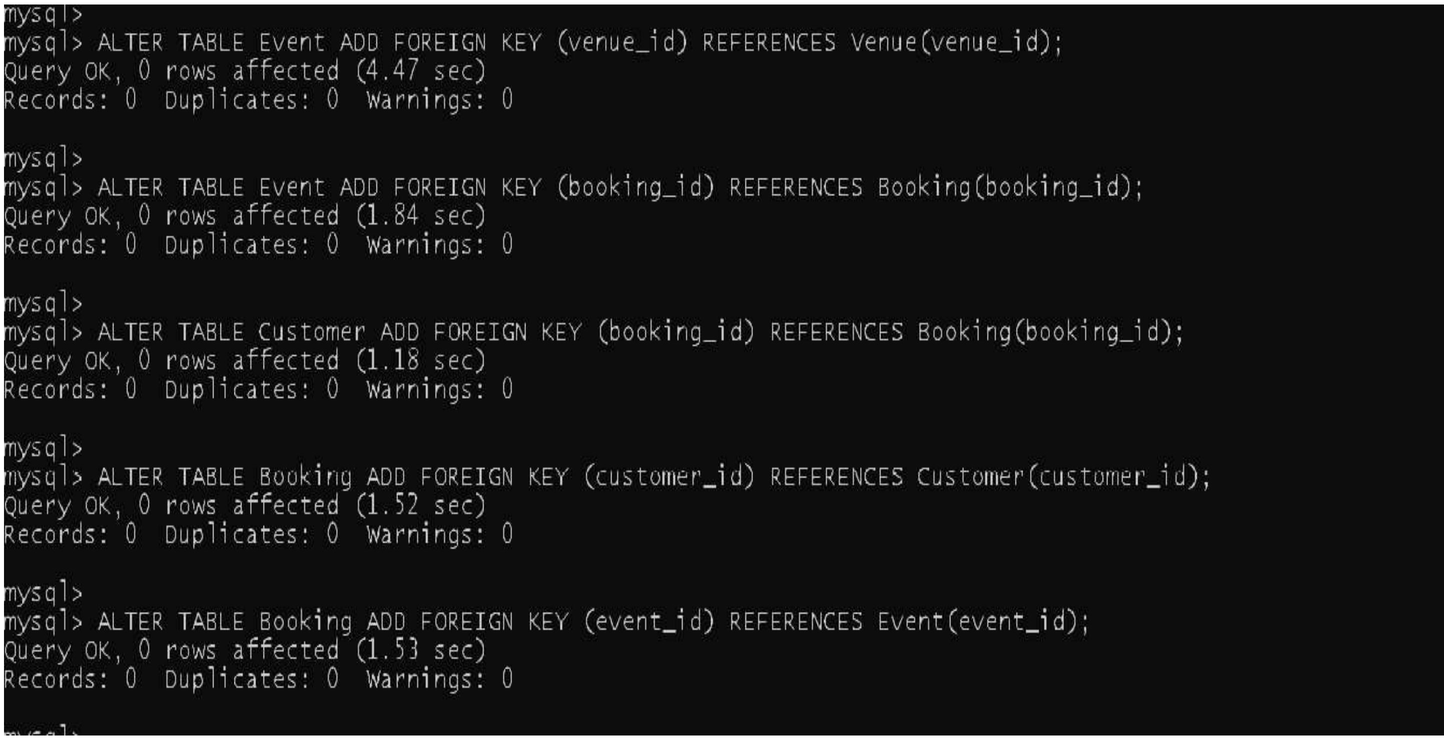


```
+--------------+---------------+------+-----+-------------------+-------------------+
| Field        | Type          | Null | Key | Default           | Extra             |
+--------------+---------------+------+-----+-------------------+-------------------+
| booking_id   | int           | NO   | PRI | NULL              |                   |
| customer_id  | int           | YES  |     | NULL              |                   |
| event_id     | int           | YES  |     | NULL              |                   |
| num_tickets  | int           | NO   |     | NULL              |                   |
| total_cost   | decimal(10,2) | NO   |     | NULL              |                   |
| booking_date | timestamp     | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+--------------+---------------+------+-----+-------------------+-------------------+
```

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

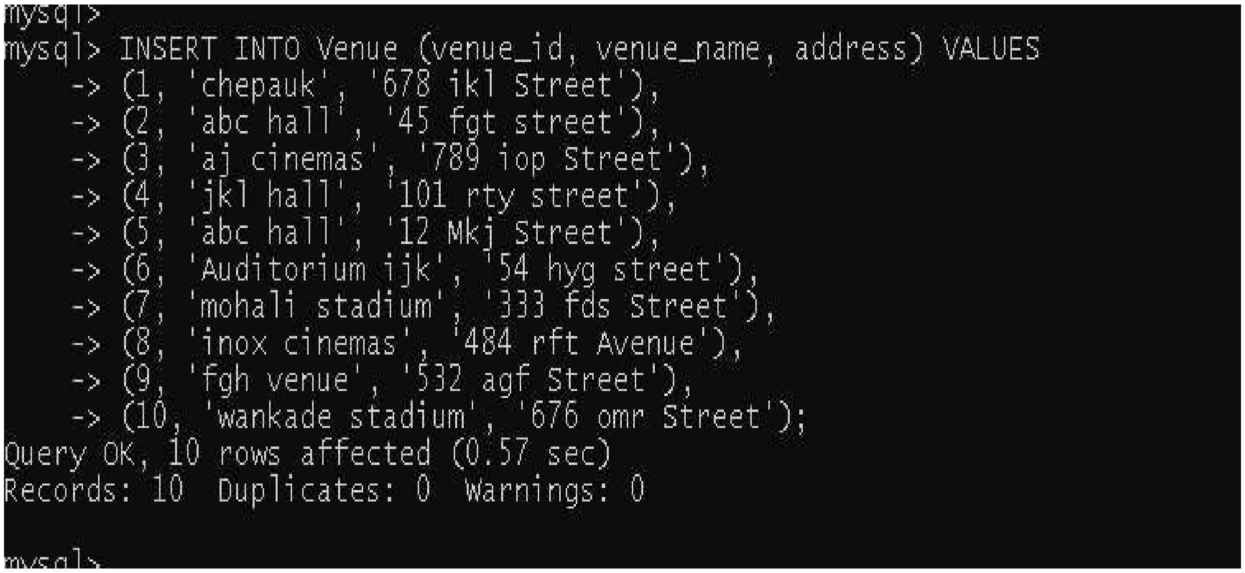


4. Create appropriate Primary Key and Foreign Key constraints for referential integrity

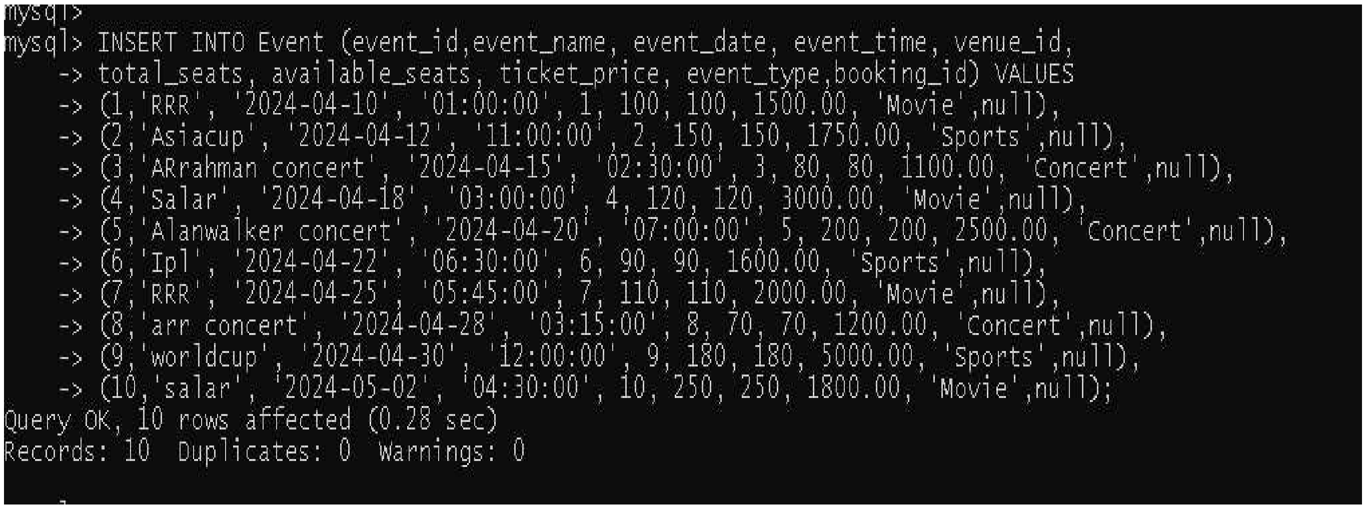

```
mysql>
mysql> ALTER TABLE Event ADD FOREIGN KEY (venue_id) REFERENCES Venue(venue_id);
Query OK, 0 rows affected (4.47 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Event ADD FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
Query OK, 0 rows affected (1.84 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Customer ADD FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
Query OK, 0 rows affected (1.18 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Booking ADD FOREIGN KEY (customer_id) REFERENCES Customer(customer_id);
Query OK, 0 rows affected (1.52 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Booking ADD FOREIGN KEY (event_id) REFERENCES Event(event_id);
Query OK, 0 rows affected (1.53 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

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

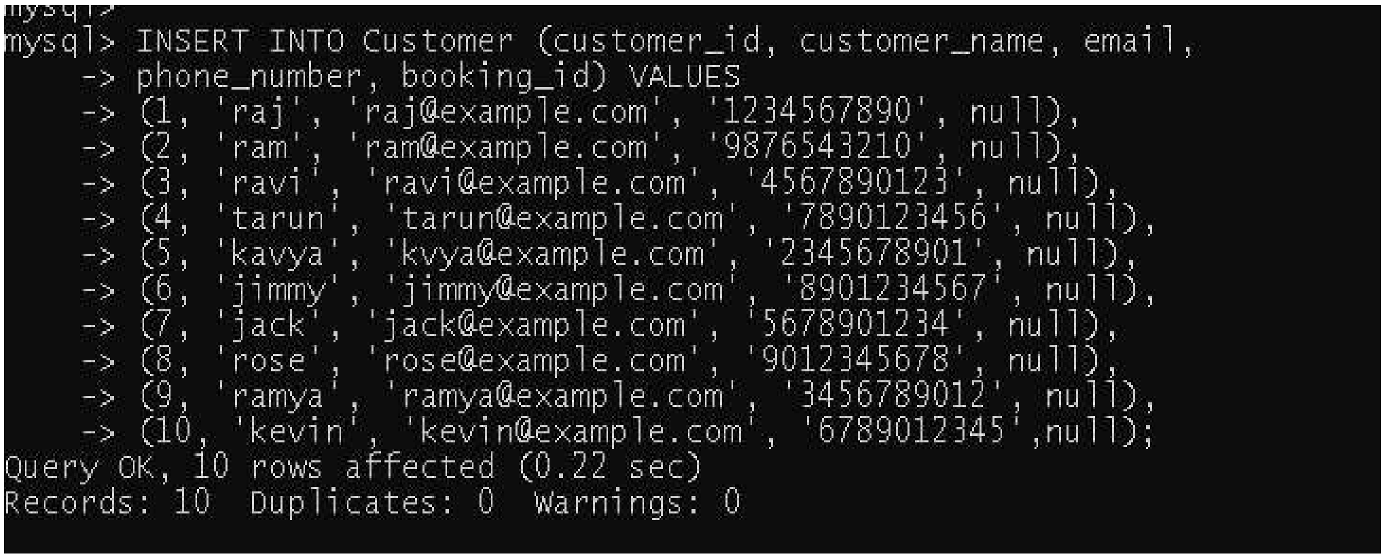
1. Write a SQL query to insert at least 10 sample records into each table.



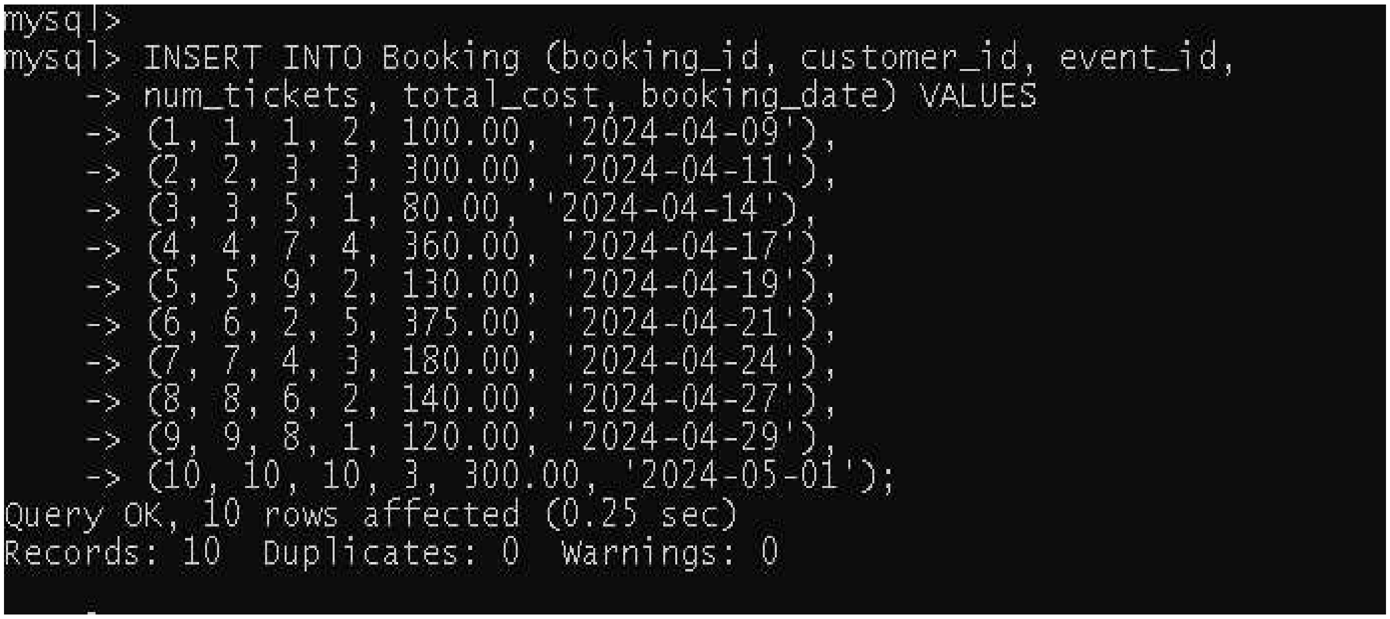
```
mysql>
mysql> INSERT INTO Venue (venue_id, venue_name, address) VALUES
    -> (1, 'chepauk', '678 ikl Street'),
    -> (2, 'abc hall', '45 fgt street'),
    -> (3, 'aj cinemas', '789 iop Street'),
    -> (4, 'jkl hall', '101 rty street'),
    -> (5, 'abc hall', '12 Mkj Street'),
    -> (6, 'Auditorium ijk', '54 hyg street'),
    -> (7, 'mohali stadium', '333 fds Street'),
    -> (8, 'inox cinemas', '484 rft Avenue'),
    -> (9, 'fgh venue', '532 agf Street'),
    -> (10, 'wankade stadium', '676 omr Street');
Query OK, 10 rows affected (0.57 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
```



```
mysql>
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
    -> (1,'RRR', '2024-04-10', '01:00:00', 1, 100, 100, 1500.00, 'Movie',null),
    -> (2,'Asiacup', '2024-04-12', '11:00:00', 2, 150, 150, 1750.00, 'Sports',null),
    -> (3,'ARrahman concert', '2024-04-15', '02:30:00', 3, 80, 80, 1100.00, 'Concert',null),
    -> (4,'Salar', '2024-04-18', '03:00:00', 4, 120, 120, 3000.00, 'Movie',null),
    -> (5,'Alanwalker concert', '2024-04-20', '07:00:00', 5, 200, 200, 2500.00, 'Concert',null),
    -> (6,'Ipl', '2024-04-22', '06:30:00', 6, 90, 90, 1600.00, 'Sports',null),
    -> (7,'RRR', '2024-04-25', '05:45:00', 7, 110, 110, 2000.00, 'Movie',null),
    -> (8,'arr concert', '2024-04-28', '03:15:00', 8, 70, 70, 1200.00, 'Concert',null),
    -> (9,'worldcup', '2024-04-30', '12:00:00', 9, 180, 180, 5000.00, 'Sports',null),
    -> (10,'salar', '2024-05-02', '04:30:00', 10, 250, 250, 1800.00, 'Movie',null);
Query OK, 10 rows affected (0.28 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

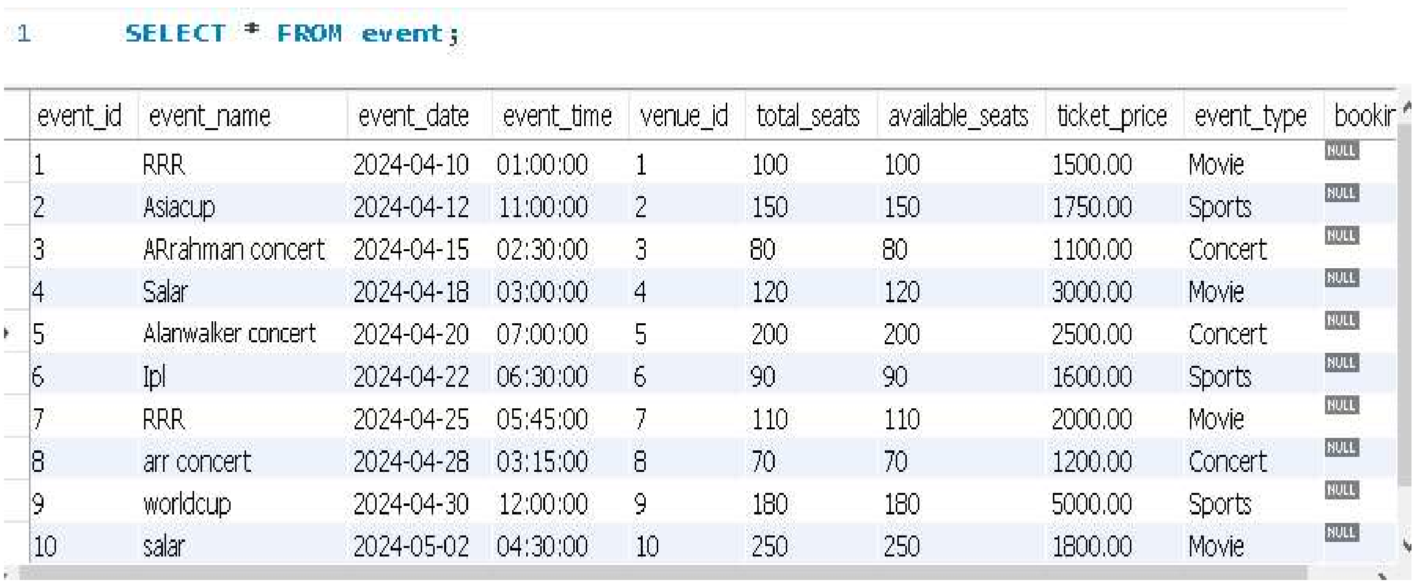


```
mysql>
mysql> INSERT INTO Customer (customer_id, customer_name, email,
    -> phone_number, booking_id) VALUES
    -> (1, 'raj', 'raj@example.com', '1234567890', null),
    -> (2, 'ram', 'ram@example.com', '9876543210', null),
    -> (3, 'ravi', 'ravi@example.com', '4567890123', null),
    -> (4, 'tarun', 'tarun@example.com', '7890123456', null),
    -> (5, 'kavya', 'kvya@example.com', '2345678901', null),
    -> (6, 'jimmy', 'jimmy@example.com', '8901234567', null),
    -> (7, 'jack', 'jack@example.com', '5678901234', null),
    -> (8, 'rose', 'rose@example.com', '9012345678', null),
    -> (9, 'ramya', 'ramya@example.com', '3456789012', null),
    -> (10, 'kevin', 'kevin@example.com', '6789012345',null);
Query OK, 10 rows affected (0.22 sec)
Records: 10  Duplicates: 0  Warnings: 0
```



```
mysql>
mysql> INSERT INTO Booking (booking_id, customer_id, event_id,
    -> num_tickets, total_cost, booking_date) VALUES
    -> (1, 1, 1, 2, 100.00, '2024-04-09'),
    -> (2, 2, 3, 3, 300.00, '2024-04-11'),
    -> (3, 3, 5, 1, 80.00, '2024-04-14'),
    -> (4, 4, 7, 4, 360.00, '2024-04-17'),
    -> (5, 5, 9, 2, 130.00, '2024-04-19'),
    -> (6, 6, 2, 5, 375.00, '2024-04-21'),
    -> (7, 7, 4, 3, 180.00, '2024-04-24'),
    -> (8, 8, 6, 2, 140.00, '2024-04-27'),
    -> (9, 9, 8, 1, 120.00, '2024-04-29'),
    -> (10, 10, 10, 3, 300.00, '2024-05-01');
Query OK, 10 rows affected (0.25 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

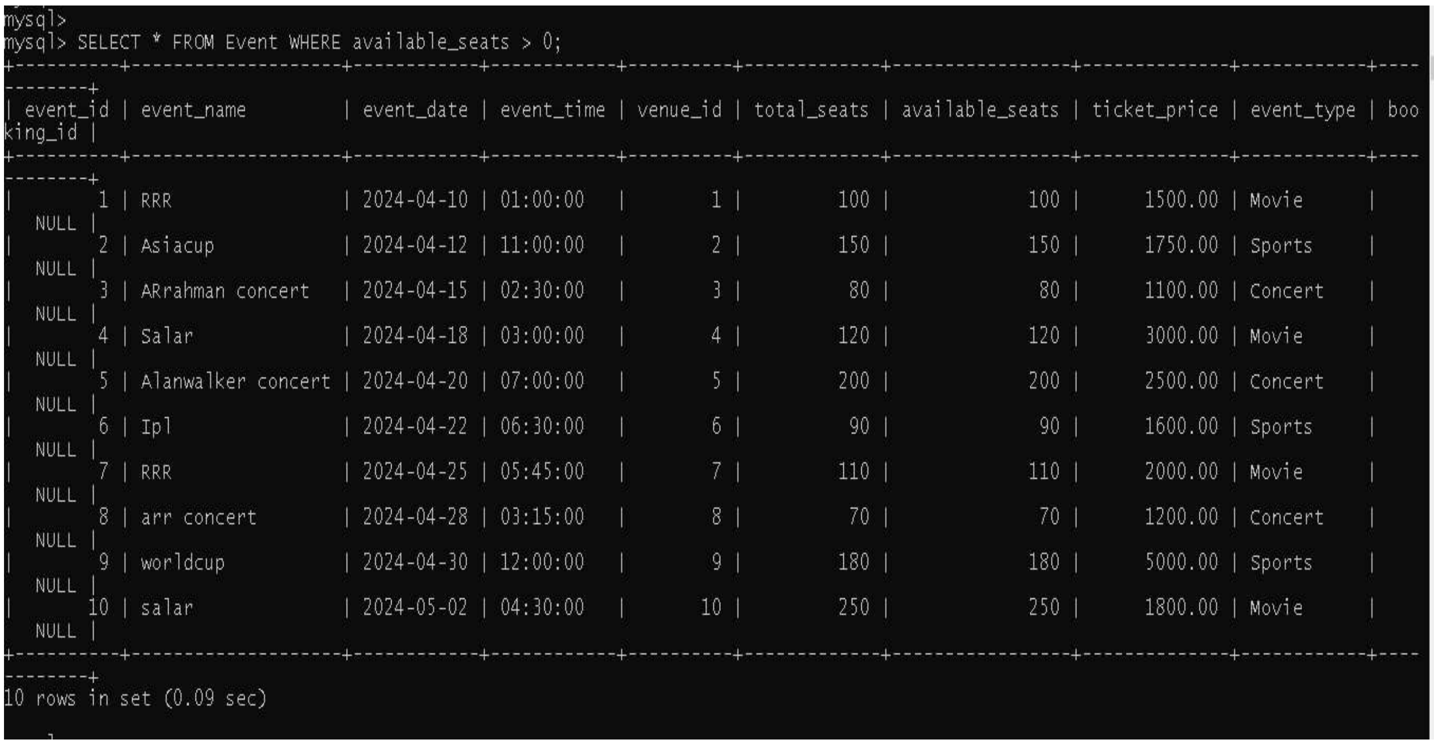
2. Write a SQL query to list all Events.



```sql
SELECT * FROM event;
```

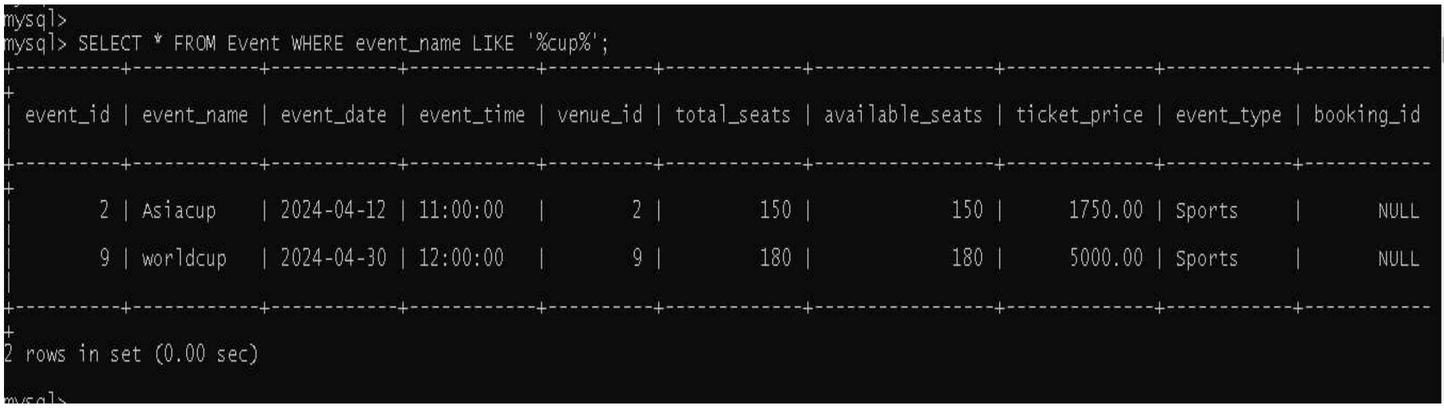
event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	bookir
1	RRR	2024-04-10	01:00:00	1	100	100	1500.00	Movie	NULL
2	Asiacup	2024-04-12	11:00:00	2	150	150	1750.00	Sports	NULL
3	ARrahman concert	2024-04-15	02:30:00	3	80	80	1100.00	Concert	NULL
4	Salar	2024-04-18	03:00:00	4	120	120	3000.00	Movie	NULL
5	Alanwalker concert	2024-04-20	07:00:00	5	200	200	2500.00	Concert	NULL
6	Ipl	2024-04-22	06:30:00	6	90	90	1600.00	Sports	NULL
7	RRR	2024-04-25	05:45:00	7	110	110	2000.00	Movie	NULL
8	arr concert	2024-04-28	03:15:00	8	70	70	1200.00	Concert	NULL
9	worldcup	2024-04-30	12:00:00	9	180	180	5000.00	Sports	NULL
10	salar	2024-05-02	04:30:00	10	250	250	1800.00	Movie	NULL

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



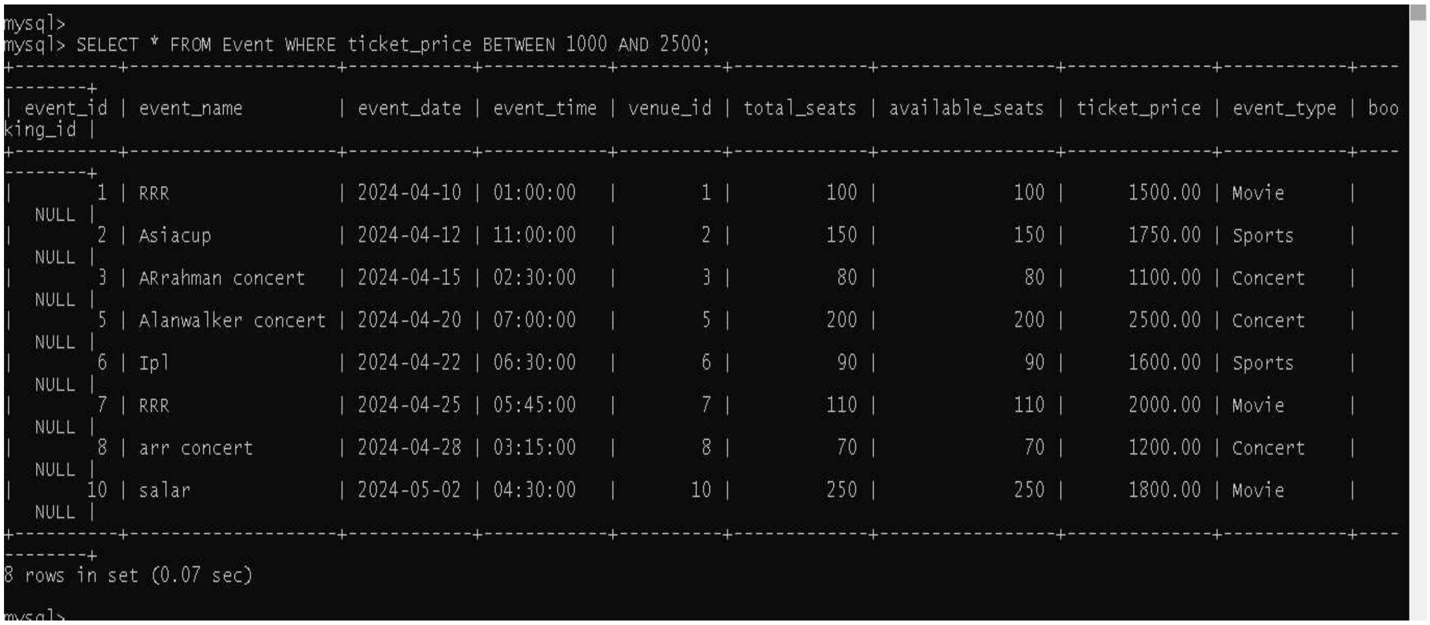
```
mysql>
mysql> SELECT * FROM Event WHERE available_seats > 0;
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
| event_id | event_name         | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | boo
king_id |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
|        1 | RRR                | 2024-04-10 | 01:00:00   |        1 |         100 |             100 |      1500.00 | Movie      |
    NULL |
|        2 | Asiacup            | 2024-04-12 | 11:00:00   |        2 |         150 |             150 |      1750.00 | Sports     |
    NULL |
|        3 | ARrahman concert   | 2024-04-15 | 02:30:00   |        3 |          80 |              80 |      1100.00 | Concert    |
    NULL |
|        4 | Salar              | 2024-04-18 | 03:00:00   |        4 |         120 |             120 |      3000.00 | Movie      |
    NULL |
|        5 | Alanwalker concert | 2024-04-20 | 07:00:00   |        5 |         200 |             200 |      2500.00 | Concert    |
    NULL |
|        6 | Ipl                | 2024-04-22 | 06:30:00   |        6 |          90 |              90 |      1600.00 | Sports     |
    NULL |
|        7 | RRR                | 2024-04-25 | 05:45:00   |        7 |         110 |             110 |      2000.00 | Movie      |
    NULL |
|        8 | arr concert        | 2024-04-28 | 03:15:00   |        8 |          70 |              70 |      1200.00 | Concert    |
    NULL |
|        9 | worldcup           | 2024-04-30 | 12:00:00   |        9 |         180 |             180 |      5000.00 | Sports     |
    NULL |
|       10 | salar              | 2024-05-02 | 04:30:00   |       10 |         250 |             250 |      1800.00 | Movie      |
    NULL |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
10 rows in set (0.09 sec)
```

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



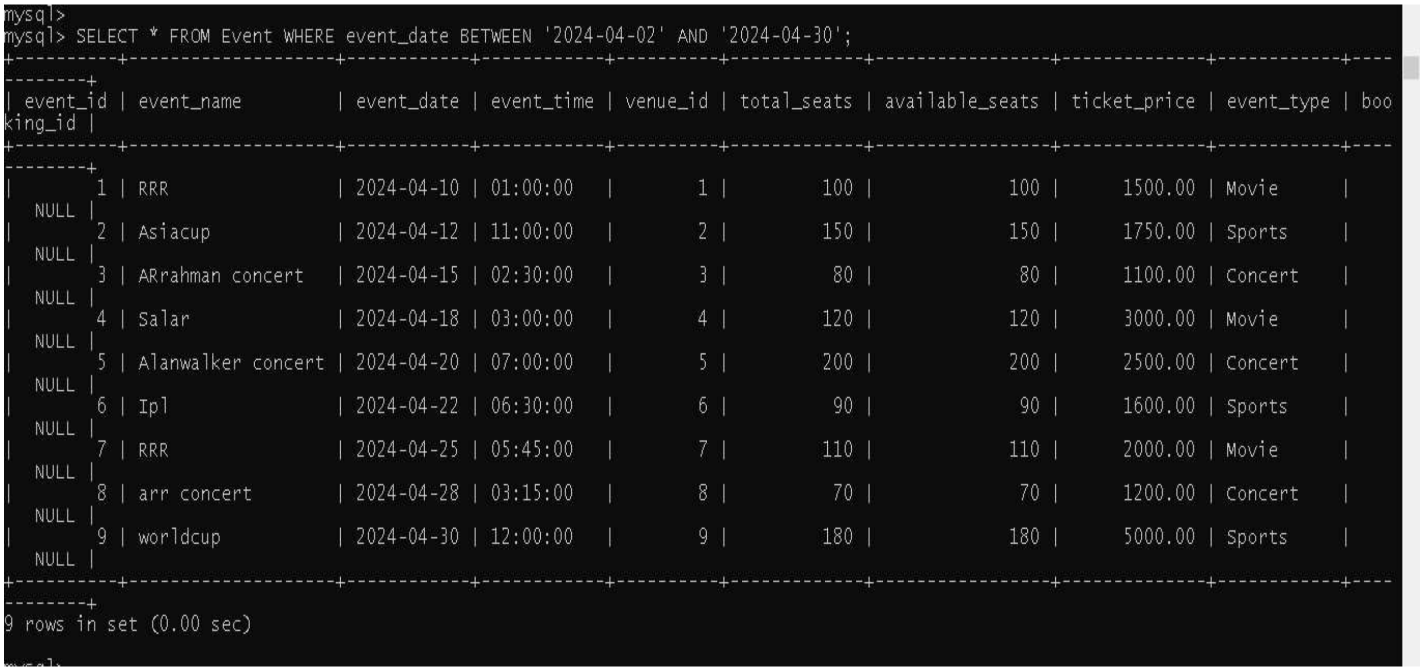
```
mysql>
mysql> SELECT * FROM Event WHERE event_name LIKE '%cup%';
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+-----------
+
| event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id
|
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+-----------
+
|        2 | Asiacup    | 2024-04-12 | 11:00:00   |        2 |         150 |             150 |      1750.00 | Sports     |       NULL
|
|        9 | worldcup   | 2024-04-30 | 12:00:00   |        9 |         180 |             180 |      5000.00 | Sports     |       NULL
|
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+-----------
+
2 rows in set (0.00 sec)

mysql>
```

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



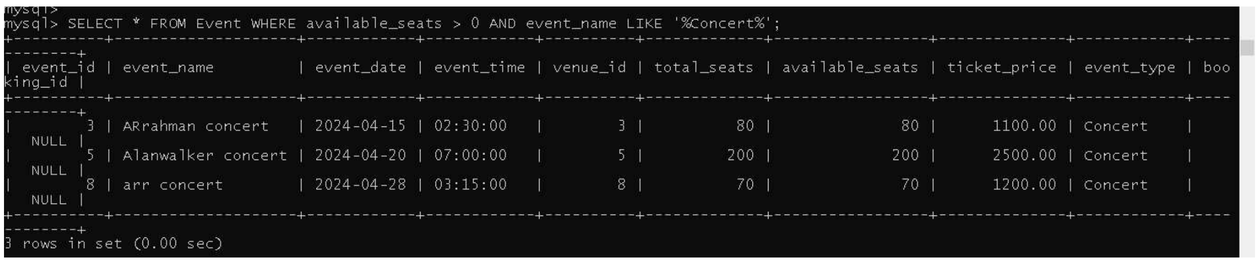
```
mysql>
mysql> SELECT * FROM Event WHERE ticket_price BETWEEN 1000 AND 2500;
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
| event_id | event_name         | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | boo
king_id |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
|        1 | RRR                | 2024-04-10 | 01:00:00   |        1 |         100 |             100 |      1500.00 | Movie      |
   NULL |
|        2 | Asiacup            | 2024-04-12 | 11:00:00   |        2 |         150 |             150 |      1750.00 | Sports     |
   NULL |
|        3 | ARrahman concert   | 2024-04-15 | 02:30:00   |        3 |          80 |              80 |      1100.00 | Concert    |
   NULL |
|        5 | Alanwalker concert | 2024-04-20 | 07:00:00   |        5 |         200 |             200 |      2500.00 | Concert    |
   NULL |
|        6 | Ipl                | 2024-04-22 | 06:30:00   |        6 |          90 |              90 |      1600.00 | Sports     |
   NULL |
|        7 | RRR                | 2024-04-25 | 05:45:00   |        7 |         110 |             110 |      2000.00 | Movie      |
   NULL |
|        8 | arr concert        | 2024-04-28 | 03:15:00   |        8 |          70 |              70 |      1200.00 | Concert    |
   NULL |
|       10 | salar              | 2024-05-02 | 04:30:00   |       10 |         250 |             250 |      1800.00 | Movie      |
   NULL |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
8 rows in set (0.07 sec)

mysql>
```

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



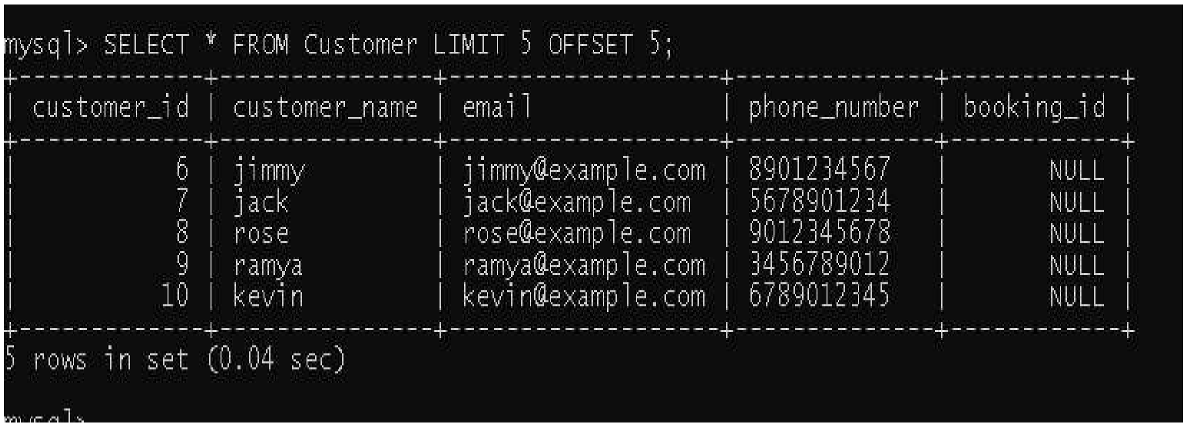
```
mysql>
mysql> SELECT * FROM Event WHERE event_date BETWEEN '2024-04-02' AND '2024-04-30';
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
| event_id | event_name         | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | boo
king_id |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
|        1 | RRR                | 2024-04-10 | 01:00:00   |        1 |         100 |             100 |      1500.00 | Movie      |
   NULL |
|        2 | Asiacup            | 2024-04-12 | 11:00:00   |        2 |         150 |             150 |      1750.00 | Sports     |
   NULL |
|        3 | ARrahman concert   | 2024-04-15 | 02:30:00   |        3 |          80 |              80 |      1100.00 | Concert    |
   NULL |
|        4 | Salar              | 2024-04-18 | 03:00:00   |        4 |         120 |             120 |      3000.00 | Movie      |
   NULL |
|        5 | Alanwalker concert | 2024-04-20 | 07:00:00   |        5 |         200 |             200 |      2500.00 | Concert    |
   NULL |
|        6 | Ipl                | 2024-04-22 | 06:30:00   |        6 |          90 |              90 |      1600.00 | Sports     |
   NULL |
|        7 | RRR                | 2024-04-25 | 05:45:00   |        7 |         110 |             110 |      2000.00 | Movie      |
   NULL |
|        8 | arr concert        | 2024-04-28 | 03:15:00   |        8 |          70 |              70 |      1200.00 | Concert    |
   NULL |
|        9 | worldcup           | 2024-04-30 | 12:00:00   |        9 |         180 |             180 |      5000.00 | Sports     |
   NULL |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
9 rows in set (0.00 sec)

mysql>
```

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



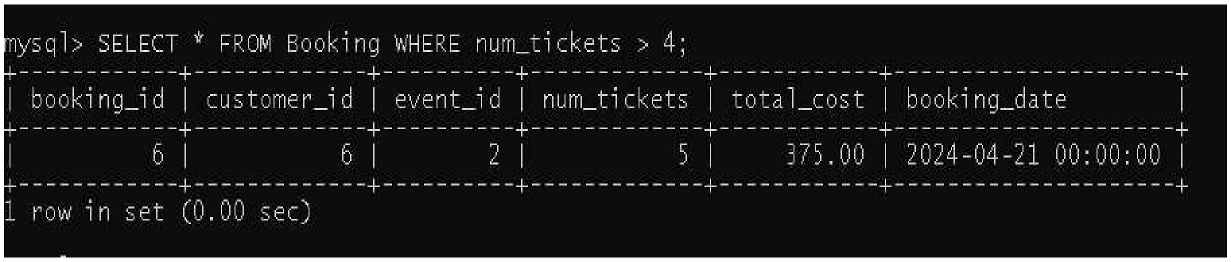
```
mysql>
mysql> SELECT * FROM Event WHERE available_seats > 0 AND event_name LIKE '%Concert%';
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
| event_id | event_name         | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | boo
king_id |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
|        3 | ARrahman concert   | 2024-04-15 | 02:30:00   |        3 |          80 |              80 |      1100.00 | Concert    |
   NULL |
|        5 | Alanwalker concert | 2024-04-20 | 07:00:00   |        5 |         200 |             200 |      2500.00 | Concert    |
   NULL |
|        8 | arr concert        | 2024-04-28 | 03:15:00   |        8 |          70 |              70 |      1200.00 | Concert    |
   NULL |
+----------+--------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
3 rows in set (0.00 sec)
```

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



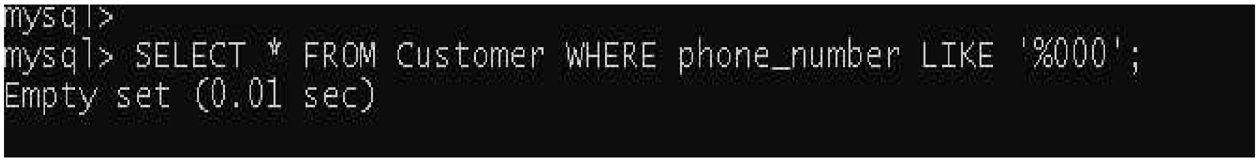
```
mysql> SELECT * FROM Customer LIMIT 5 OFFSET 5;
+-------------+---------------+-------------------+--------------+------------+
| customer_id | customer_name | email             | phone_number | booking_id |
+-------------+---------------+-------------------+--------------+------------+
|           6 | jimmy         | jimmy@example.com | 8901234567   |       NULL |
|           7 | jack          | jack@example.com  | 5678901234   |       NULL |
|           8 | rose          | rose@example.com  | 9012345678   |       NULL |
|           9 | ramya         | ramya@example.com | 3456789012   |       NULL |
|          10 | kevin         | kevin@example.com | 6789012345   |       NULL |
+-------------+---------------+-------------------+--------------+------------+
5 rows in set (0.04 sec)
```

9. Write a 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        |
+------------+-------------+----------+-------------+------------+---------------------+
|          6 |           6 |        2 |           5 |     375.00 | 2024-04-21 00:00:00 |
+------------+-------------+----------+-------------+------------+---------------------+
1 row in set (0.00 sec)
```

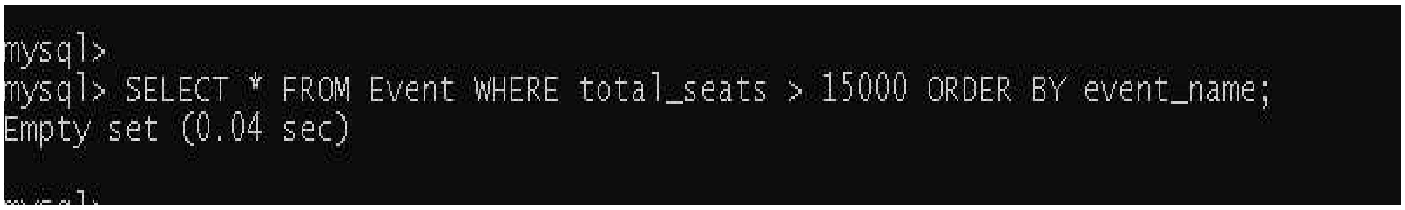
10. Write a SQL query to retrieve customer information whose phone number end with '000'



```
mysql>
mysql> SELECT * FROM Customer WHERE phone_number LIKE '%000';
Empty set (0.01 sec)
```

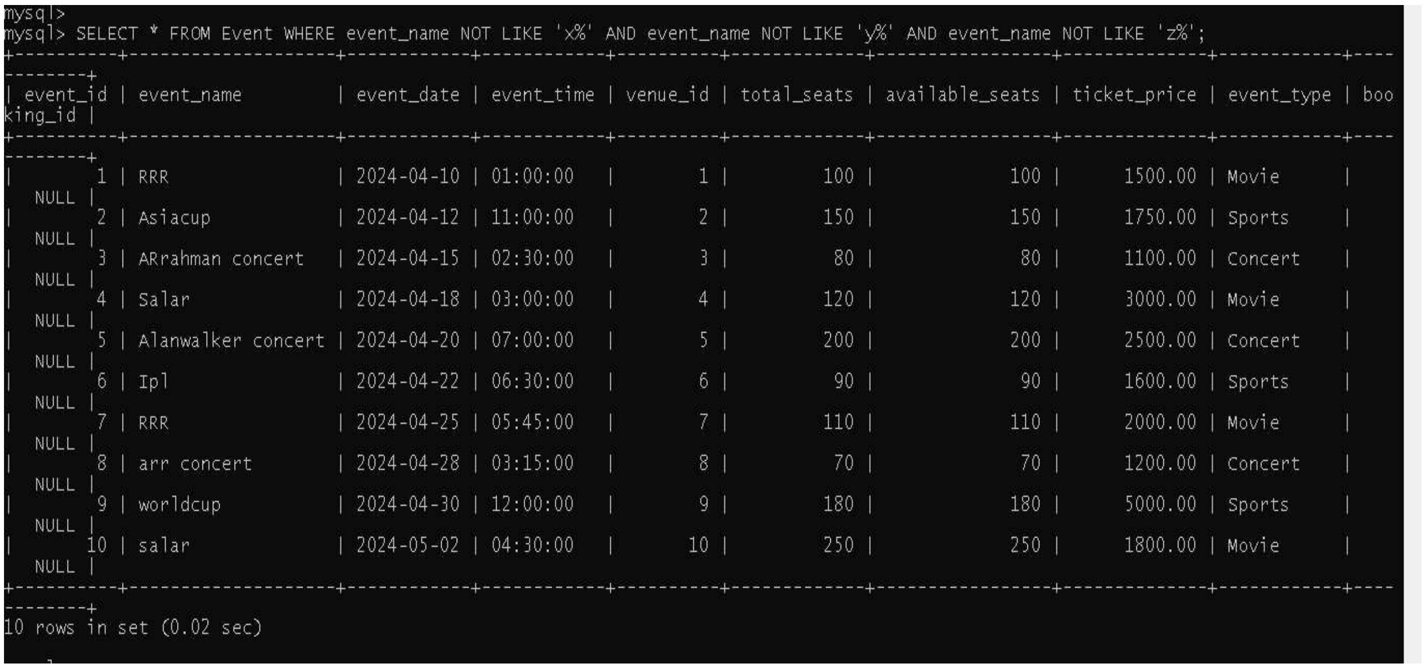
11. Write a SQL query to retrieve the events

in order whose seat capacity more than 15000.



```
mysql>
mysql> SELECT * FROM Event WHERE total_seats > 15000 ORDER BY event_name;
Empty set (0.04 sec)

mysql>
```

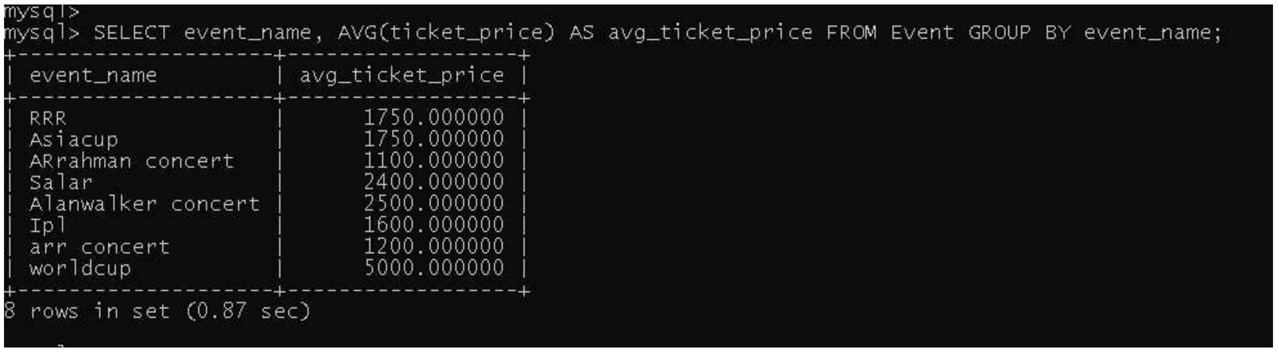
12. Write a SQL query to select events name not start with ‘x’, ‘y’, ‘z’



```
mysql>
mysql> SELECT * FROM Event WHERE event_name NOT LIKE 'x%' AND event_name NOT LIKE 'y%' AND event_name NOT LIKE 'z%';
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
| event_id | event_name        | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | boo
king_id |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
|        1 | RRR               | 2024-04-10 | 01:00:00   |        1 |         100 |             100 |      1500.00 | Movie      |
   NULL |
|        2 | Asiacup           | 2024-04-12 | 11:00:00   |        2 |         150 |             150 |      1750.00 | Sports     |
   NULL |
|        3 | ARrahman concert  | 2024-04-15 | 02:30:00   |        3 |          80 |              80 |      1100.00 | Concert    |
   NULL |
|        4 | Salar             | 2024-04-18 | 03:00:00   |        4 |         120 |             120 |      3000.00 | Movie      |
   NULL |
|        5 | Alanwalker concert | 2024-04-20 | 07:00:00   |        5 |         200 |             200 |      2500.00 | Concert    |
   NULL |
|        6 | Ipl               | 2024-04-22 | 06:30:00   |        6 |          90 |              90 |      1600.00 | Sports     |
   NULL |
|        7 | RRR               | 2024-04-25 | 05:45:00   |        7 |         110 |             110 |      2000.00 | Movie      |
   NULL |
|        8 | arr concert       | 2024-04-28 | 03:15:00   |        8 |          70 |              70 |      1200.00 | Concert    |
   NULL |
|        9 | worldcup          | 2024-04-30 | 12:00:00   |        9 |         180 |             180 |      5000.00 | Sports     |
   NULL |
|       10 | salar             | 2024-05-02 | 04:30:00   |       10 |         250 |             250 |      1800.00 | Movie      |
   NULL |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+----
--------+
10 rows in set (0.02 sec)
```

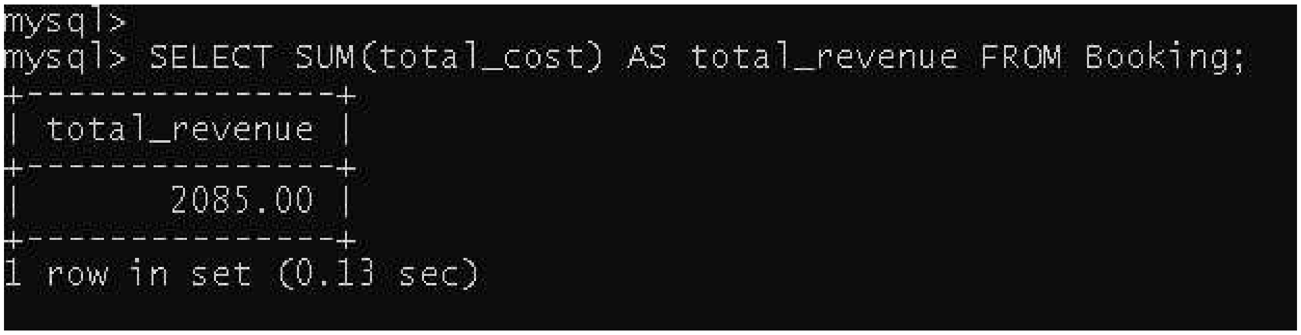
Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:

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



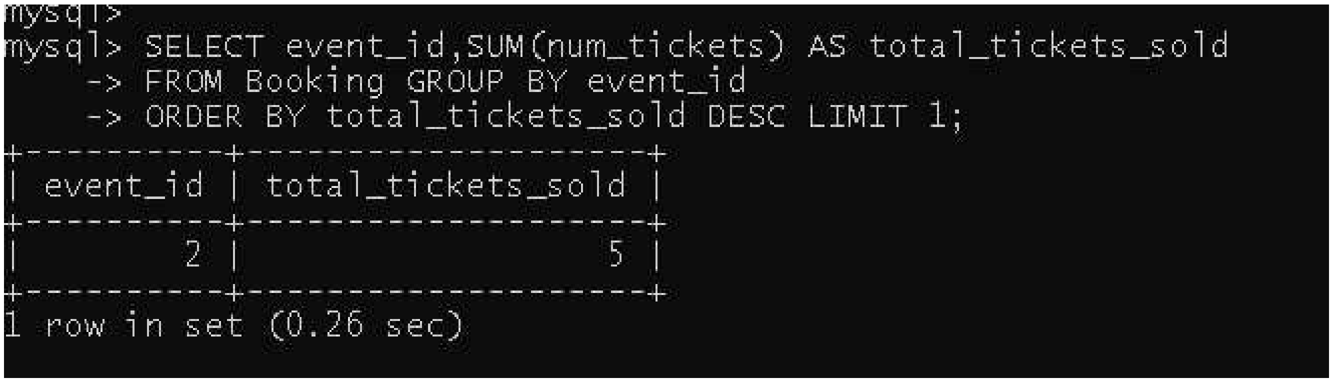
```
mysql>
mysql> SELECT event_name, AVG(ticket_price) AS avg_ticket_price FROM Event GROUP BY event_name;
+--------------------+------------------+
| event_name         | avg_ticket_price |
+--------------------+------------------+
| RRR                |      1750.000000 |
| Asiacup            |      1750.000000 |
| ARrahman concert   |      1100.000000 |
| Salar              |      2400.000000 |
| Alanwalker concert |      2500.000000 |
| Ipl                |      1600.000000 |
| arr concert        |      1200.000000 |
| worldcup           |      5000.000000 |
+--------------------+------------------+
8 rows in set (0.87 sec)
```

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



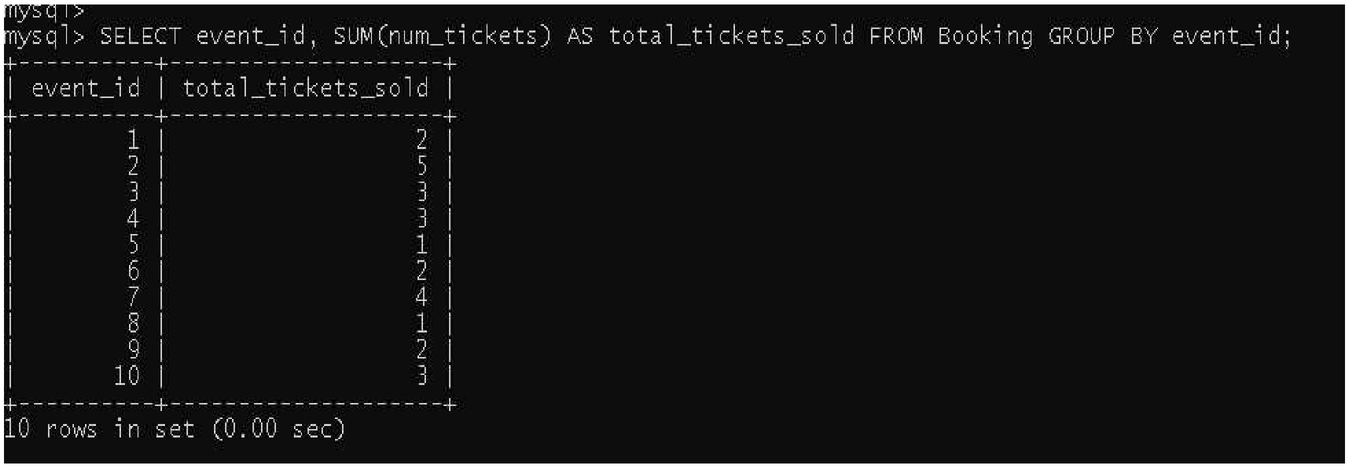
```
mysql>
mysql> SELECT SUM(total_cost) AS total_revenue FROM Booking;
+---------------+
| total_revenue |
+---------------+
|       2085.00 |
+---------------+
1 row in set (0.13 sec)
```

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



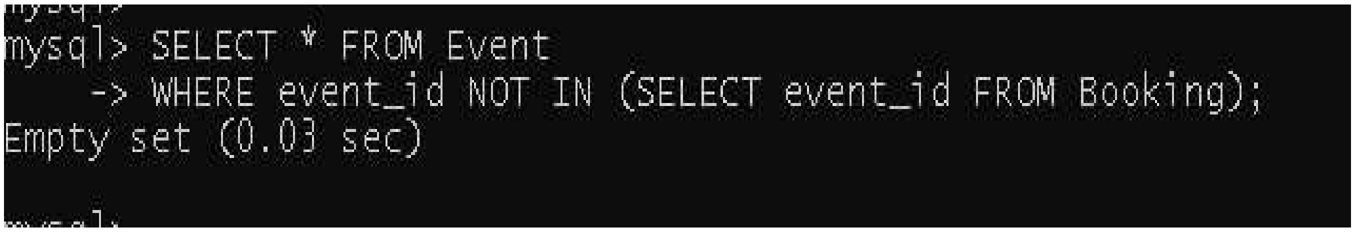
```
mysql>
mysql> SELECT event_id,SUM(num_tickets) AS total_tickets_sold
    -> FROM Booking GROUP BY event_id
    -> ORDER BY total_tickets_sold DESC LIMIT 1;
+----------+--------------------+
| event_id | total_tickets_sold |
+----------+--------------------+
|        2 |                  5 |
+----------+--------------------+
1 row in set (0.26 sec)
```

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



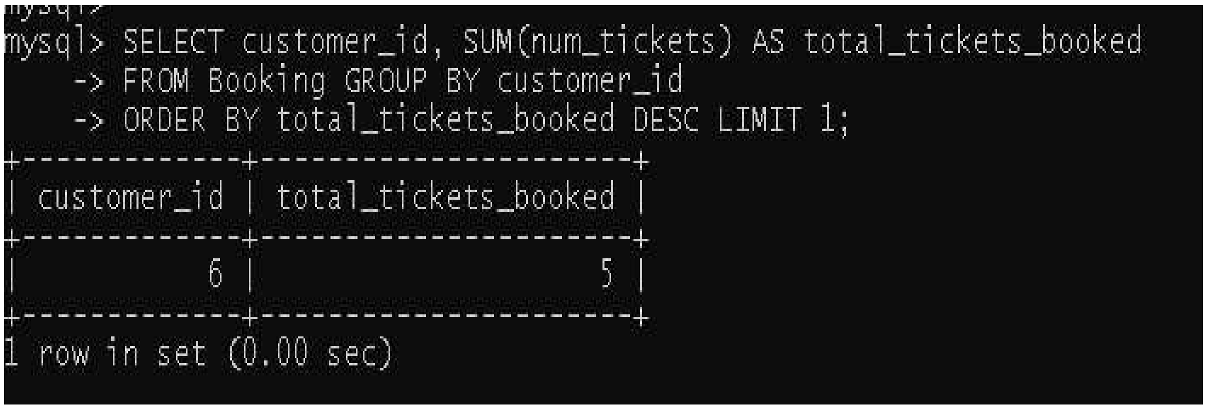
```
mysql>
mysql> SELECT event_id, SUM(num_tickets) AS total_tickets_sold FROM Booking GROUP BY event_id;
+----------+--------------------+
| event_id | total_tickets_sold |
+----------+--------------------+
|        1 |                  2 |
|        2 |                  5 |
|        3 |                  3 |
|        4 |                  3 |
|        5 |                  1 |
|        6 |                  2 |
|        7 |                  4 |
|        8 |                  1 |
|        9 |                  2 |
|       10 |                  3 |
+----------+--------------------+
10 rows in set (0.00 sec)
```

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



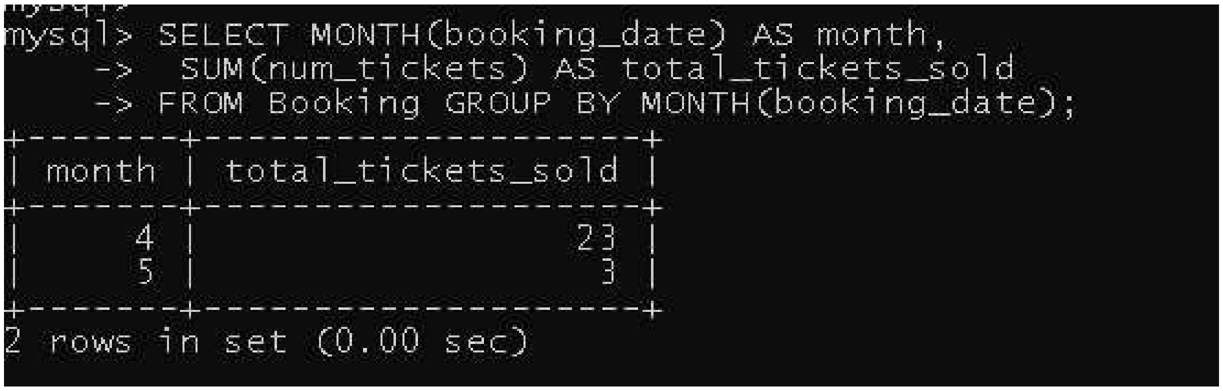
```
mysql> SELECT * FROM Event
    -> WHERE event_id NOT IN (SELECT event_id FROM Booking);
Empty set (0.03 sec)
```

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



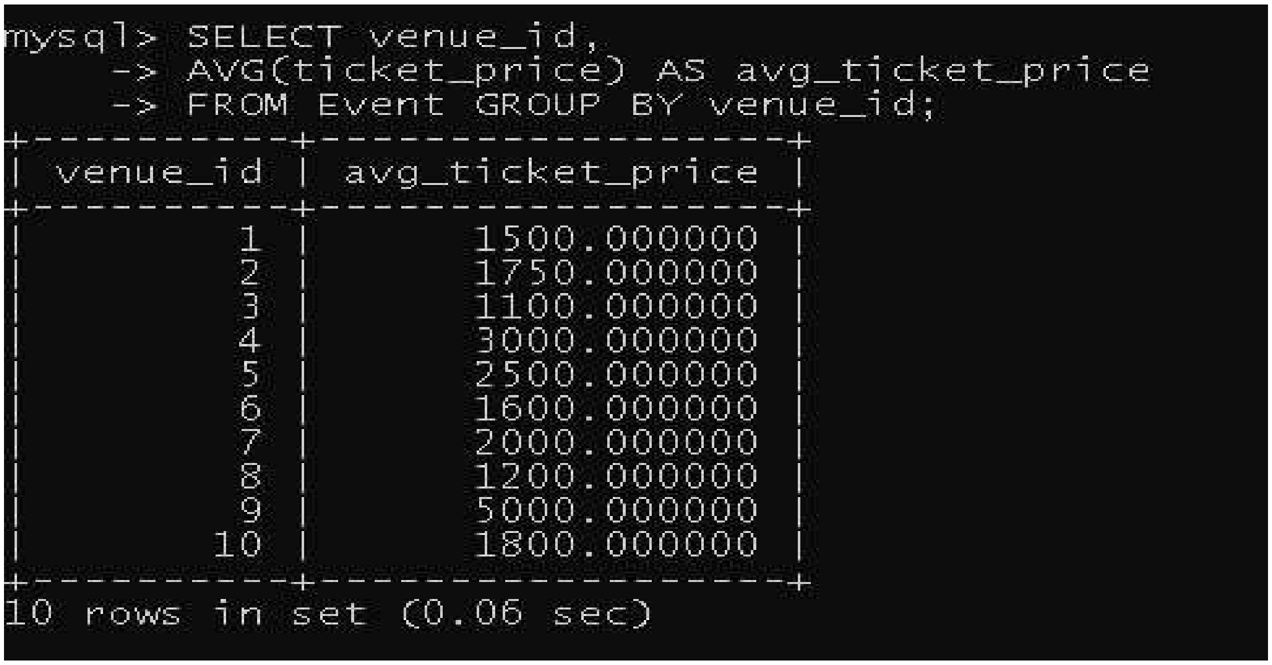
```
mysql> SELECT customer_id, SUM(num_tickets) AS total_tickets_booked
    -> FROM Booking GROUP BY customer_id
    -> ORDER BY total_tickets_booked DESC LIMIT 1;
+-------------+----------------------+
| customer_id | total_tickets_booked |
+-------------+----------------------+
|           6 |                    5 |
+-------------+----------------------+
1 row in set (0.00 sec)
```

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



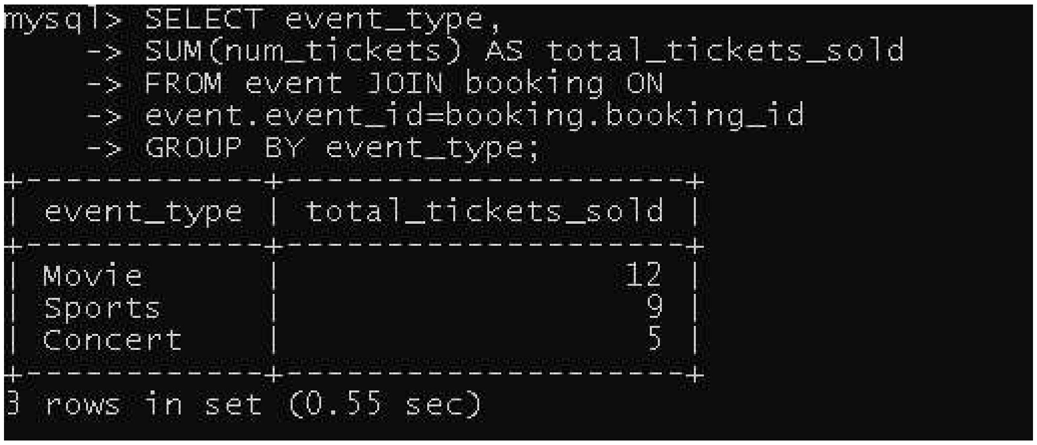
```
mysql> SELECT MONTH(booking_date) AS month,
    ->  SUM(num_tickets) AS total_tickets_sold
    -> FROM Booking GROUP BY MONTH(booking_date);
+-------+--------------------+
| month | total_tickets_sold |
+-------+--------------------+
|     4 |                 23 |
|     5 |                  3 |
+-------+--------------------+
2 rows in set (0.00 sec)
```

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



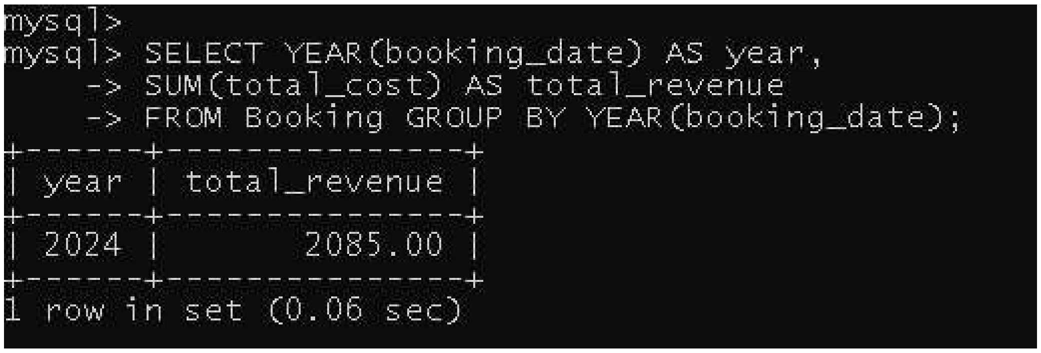
```
mysql> SELECT venue_id,
    -> AVG(ticket_price) AS avg_ticket_price
    -> FROM Event GROUP BY venue_id;
+----------+------------------+
| venue_id | avg_ticket_price |
+----------+------------------+
|        1 |      1500.000000 |
|        2 |      1750.000000 |
|        3 |      1100.000000 |
|        4 |      3000.000000 |
|        5 |      2500.000000 |
|        6 |      1600.000000 |
|        7 |      2000.000000 |
|        8 |      1200.000000 |
|        9 |      5000.000000 |
|       10 |      1800.000000 |
+----------+------------------+
10 rows in set (0.06 sec)
```

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



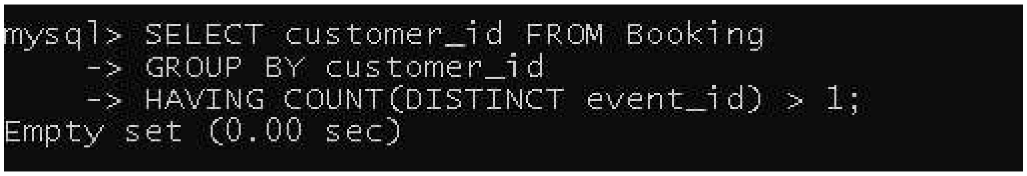
```
mysql> SELECT event_type,
    -> SUM(num_tickets) AS total_tickets_sold
    -> FROM event JOIN booking ON
    -> event.event_id=booking.booking_id
    -> GROUP BY event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Movie      |                 12 |
| Sports     |                  9 |
| Concert    |                  5 |
+------------+--------------------+
3 rows in set (0.55 sec)
```

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



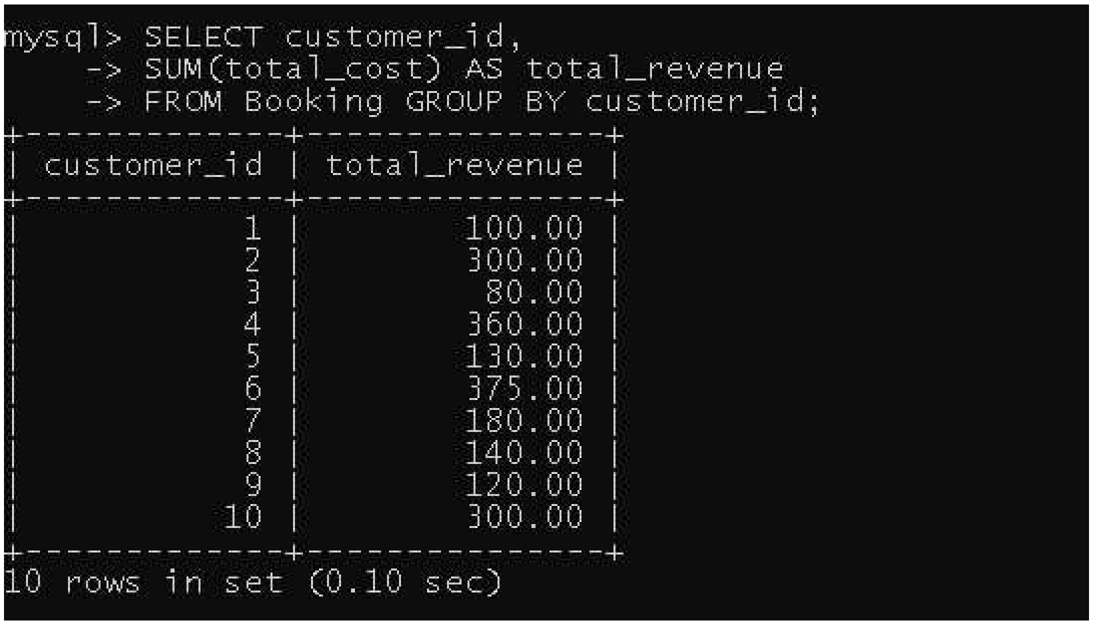
```
mysql>
mysql> SELECT YEAR(booking_date) AS year,
    -> SUM(total_cost) AS total_revenue
    -> FROM Booking GROUP BY YEAR(booking_date);
+------+---------------+
| year | total_revenue |
+------+---------------+
| 2024 |       2085.00 |
+------+---------------+
1 row in set (0.06 sec)
```

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



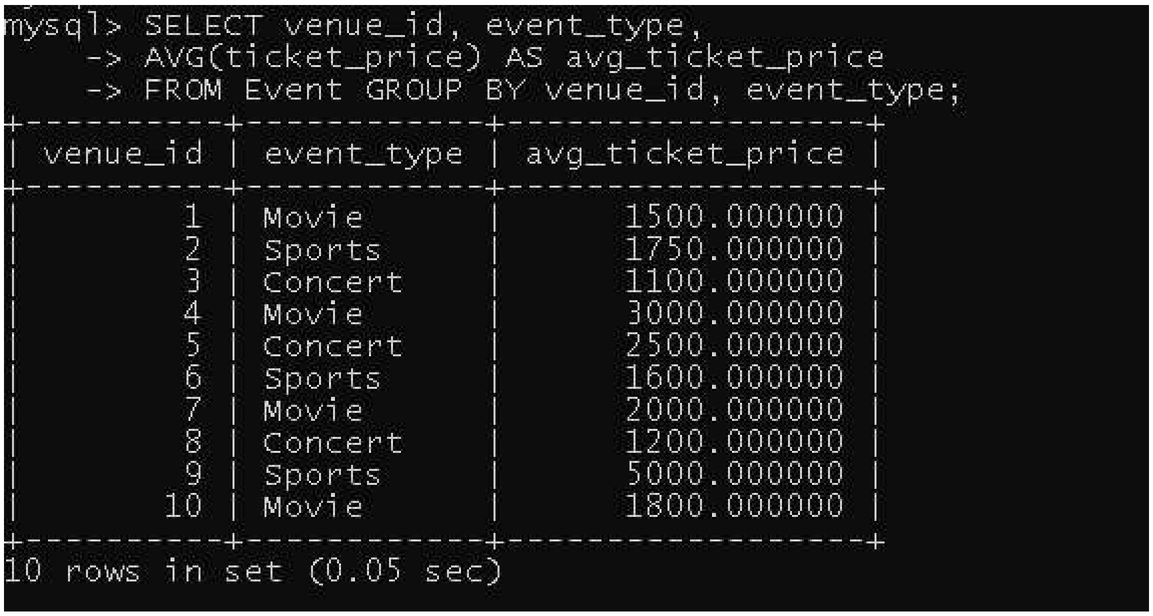
```
mysql> SELECT customer_id FROM Booking
    -> GROUP BY customer_id
    -> HAVING COUNT(DISTINCT event_id) > 1;
Empty set (0.00 sec)
```

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



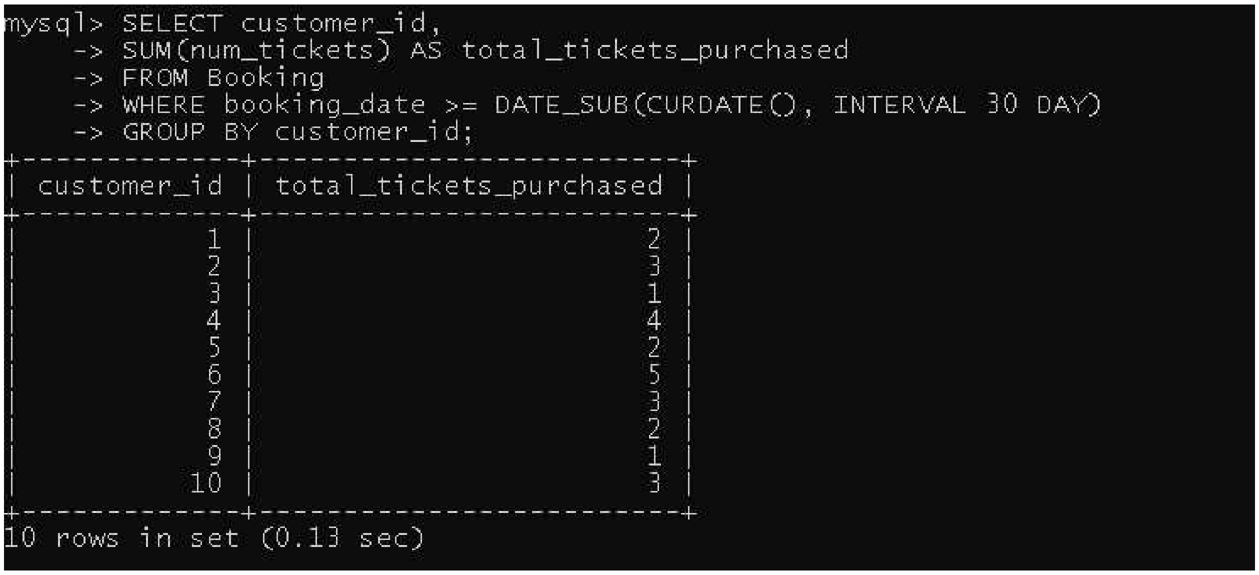
```
mysql> SELECT customer_id,
    -> SUM(total_cost) AS total_revenue
    -> FROM Booking GROUP BY customer_id;
+-------------+---------------+
| customer_id | total_revenue |
+-------------+---------------+
|           1 |        100.00 |
|           2 |        300.00 |
|           3 |         80.00 |
|           4 |        360.00 |
|           5 |        130.00 |
|           6 |        375.00 |
|           7 |        180.00 |
|           8 |        140.00 |
|           9 |        120.00 |
|          10 |        300.00 |
+-------------+---------------+
10 rows in set (0.10 sec)
```

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



```
mysql> SELECT venue_id, event_type,
    -> AVG(ticket_price) AS avg_ticket_price
    -> FROM Event GROUP BY venue_id, event_type;
+----------+------------+------------------+
| venue_id | event_type | avg_ticket_price |
+----------+------------+------------------+
|        1 | Movie      |      1500.000000 |
|        2 | Sports     |      1750.000000 |
|        3 | Concert    |      1100.000000 |
|        4 | Movie      |      3000.000000 |
|        5 | Concert    |      2500.000000 |
|        6 | Sports     |      1600.000000 |
|        7 | Movie      |      2000.000000 |
|        8 | Concert    |      1200.000000 |
|        9 | Sports     |      5000.000000 |
|       10 | Movie      |      1800.000000 |
+----------+------------+------------------+
10 rows in set (0.05 sec)
```

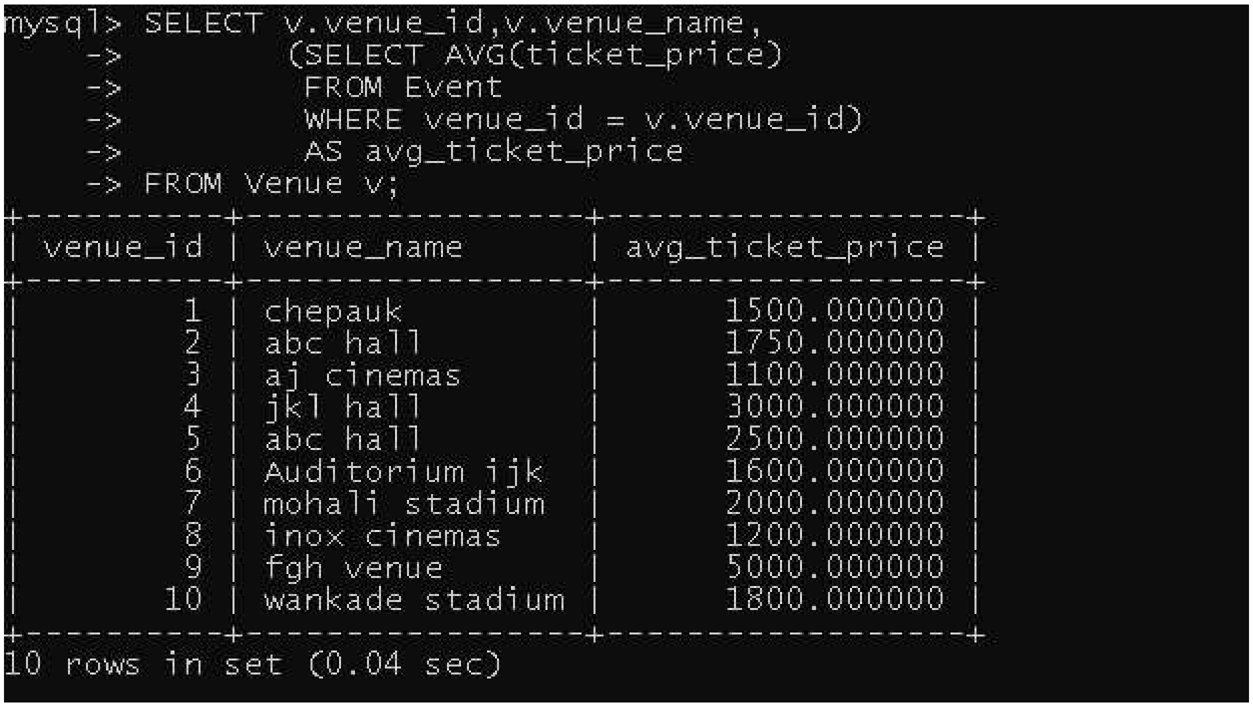
14. Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days.



```
mysql> SELECT customer_id,
    -> SUM(num_tickets) AS total_tickets_purchased
    -> FROM Booking
    -> WHERE booking_date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)
    -> GROUP BY customer_id;
+-------------+-------------------------+
| customer_id | total_tickets_purchased |
+-------------+-------------------------+
|           1 |                       2 |
|           2 |                       3 |
|           3 |                       1 |
|           4 |                       4 |
|           5 |                       2 |
|           6 |                       5 |
|           7 |                       3 |
|           8 |                       2 |
|           9 |                       1 |
|          10 |                       3 |
+-------------+-------------------------+
10 rows in set (0.13 sec)
```

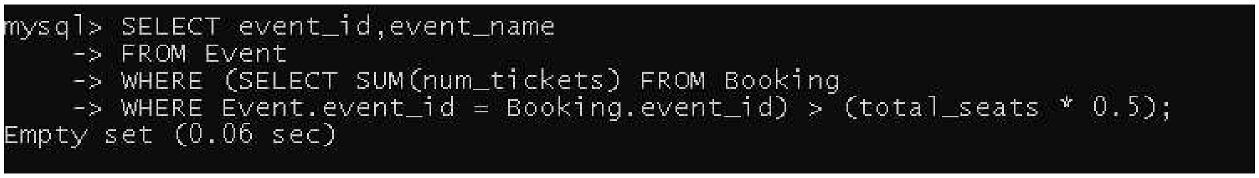
Tasks 4: Subquery and its types

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



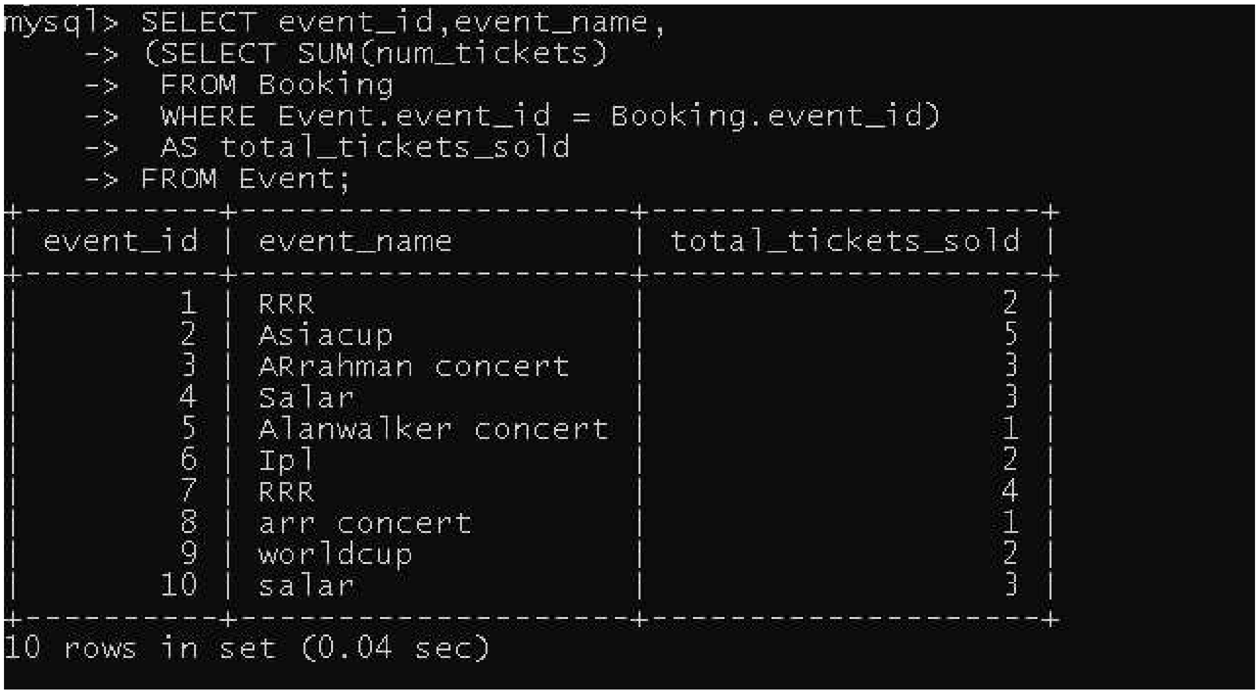
```
mysql> SELECT v.venue_id,v.venue_name,
    ->        (SELECT AVG(ticket_price)
    ->         FROM Event
    ->         WHERE venue_id = v.venue_id)
    ->         AS avg_ticket_price
    -> FROM Venue v;
+----------+-----------------+------------------+
| venue_id | venue_name      | avg_ticket_price |
+----------+-----------------+------------------+
|        1 | chepauk         |      1500.000000 |
|        2 | abc hall        |      1750.000000 |
|        3 | aj cinemas      |      1100.000000 |
|        4 | jkl hall        |      3000.000000 |
|        5 | abc hall        |      2500.000000 |
|        6 | Auditorium ijk  |      1600.000000 |
|        7 | mohali stadium  |      2000.000000 |
|        8 | inox cinemas    |      1200.000000 |
|        9 | fgh venue       |      5000.000000 |
|       10 | wankade stadium |      1800.000000 |
+----------+-----------------+------------------+
10 rows in set (0.04 sec)
```

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



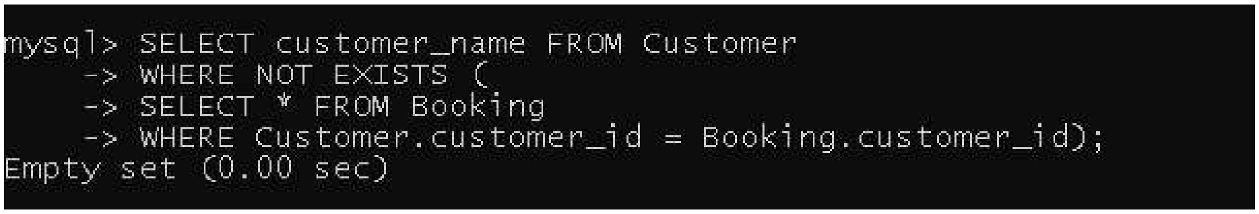
```
mysql> SELECT event_id,event_name
    -> FROM Event
    -> WHERE (SELECT SUM(num_tickets) FROM Booking
    -> WHERE Event.event_id = Booking.event_id) > (total_seats * 0.5);
Empty set (0.06 sec)
```

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



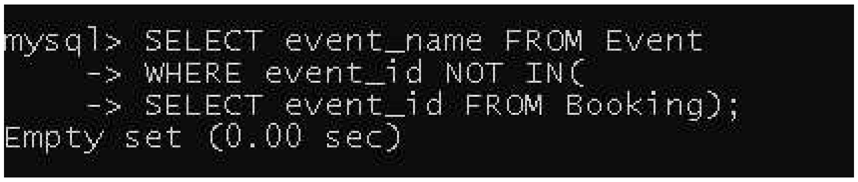
```
mysql> SELECT event_id,event_name,
    -> (SELECT SUM(num_tickets)
    ->  FROM Booking
    ->  WHERE Event.event_id = Booking.event_id)
    ->  AS total_tickets_sold
    -> FROM Event;
+----------+--------------------+--------------------+
| event_id | event_name         | total_tickets_sold |
+----------+--------------------+--------------------+
|        1 | RRR                |                  2 |
|        2 | Asiacup            |                  5 |
|        3 | ARrahman concert   |                  3 |
|        4 | Salar              |                  3 |
|        5 | Alanwalker concert |                  1 |
|        6 | Ipl                |                  2 |
|        7 | RRR                |                  4 |
|        8 | arr concert        |                  1 |
|        9 | worldcup           |                  2 |
|       10 | salar              |                  3 |
+----------+--------------------+--------------------+
10 rows in set (0.04 sec)
```

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



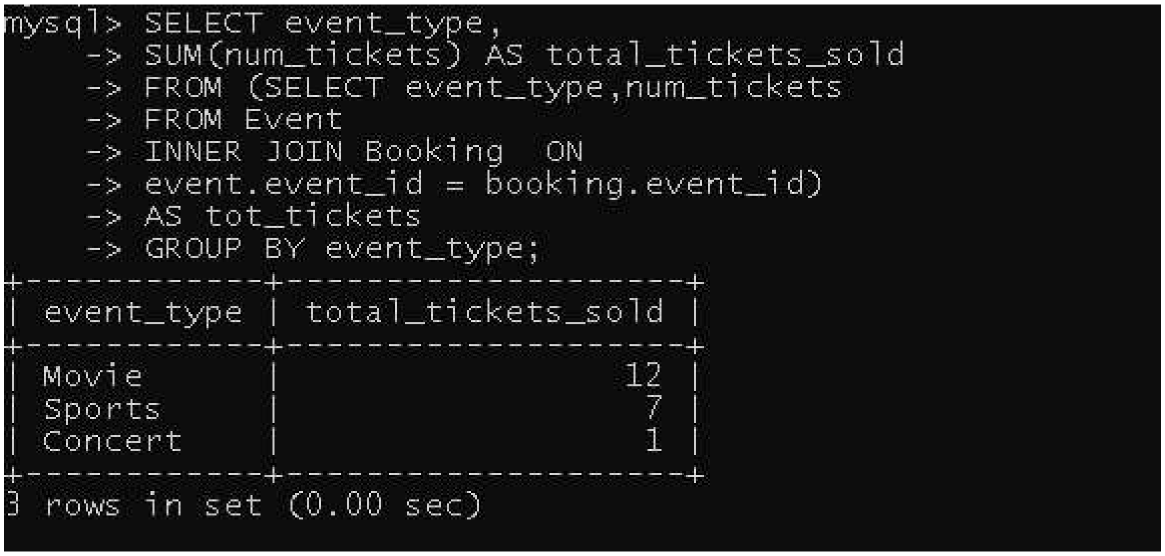
```
mysql> SELECT customer_name FROM Customer
    -> WHERE NOT EXISTS (
    -> SELECT * FROM Booking
    -> WHERE Customer.customer_id = Booking.customer_id);
Empty set (0.00 sec)
```

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



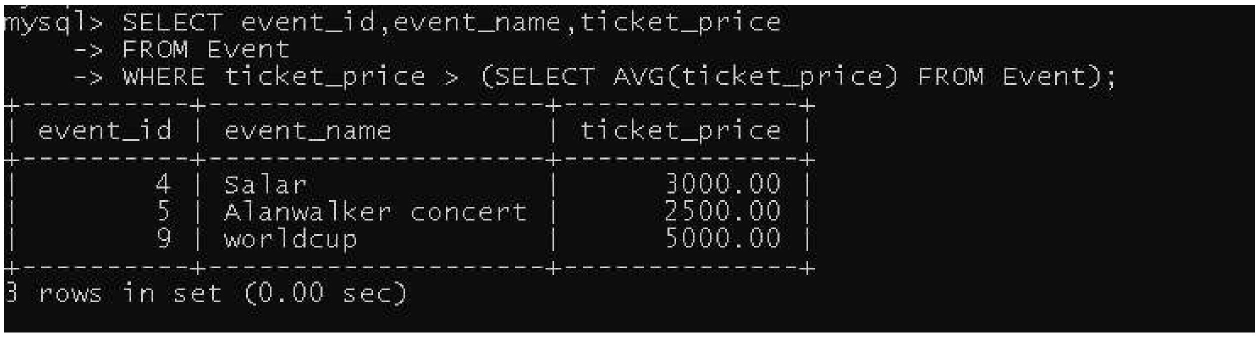
```
mysql> SELECT event_name FROM Event
    -> WHERE event_id NOT IN(
    -> SELECT event_id FROM Booking);
Empty set (0.00 sec)
```

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



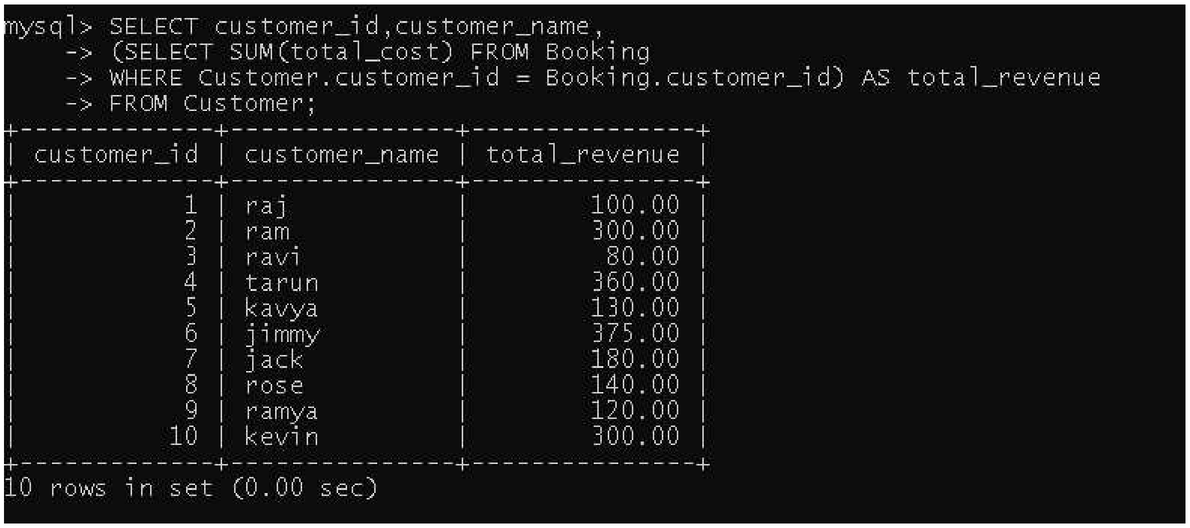
```
mysql> SELECT event_type,
    -> SUM(num_tickets) AS total_tickets_sold
    -> FROM (SELECT event_type,num_tickets
    -> FROM Event
    -> INNER JOIN Booking  ON
    -> event.event_id = booking.event_id)
    -> AS tot_tickets
    -> GROUP BY event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Movie      |                 12 |
| Sports     |                  7 |
| Concert    |                  1 |
+------------+--------------------+
3 rows in set (0.00 sec)
```

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



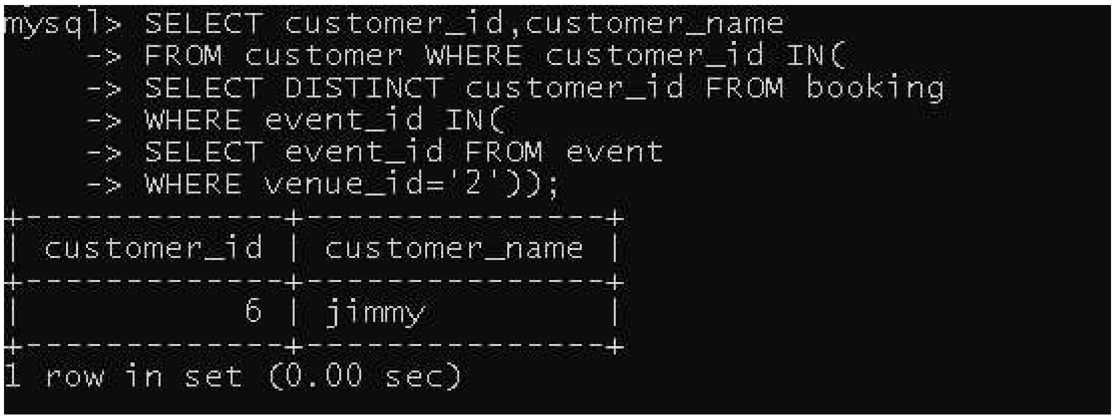
```
mysql> SELECT event_id,event_name,ticket_price
    -> FROM Event
    -> WHERE ticket_price > (SELECT AVG(ticket_price) FROM Event);
+----------+--------------------+--------------+
| event_id | event_name         | ticket_price |
+----------+--------------------+--------------+
|        4 | Salar              |      3000.00 |
|        5 | Alanwalker concert |      2500.00 |
|        9 | worldcup           |      5000.00 |
+----------+--------------------+--------------+
3 rows in set (0.00 sec)
```

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



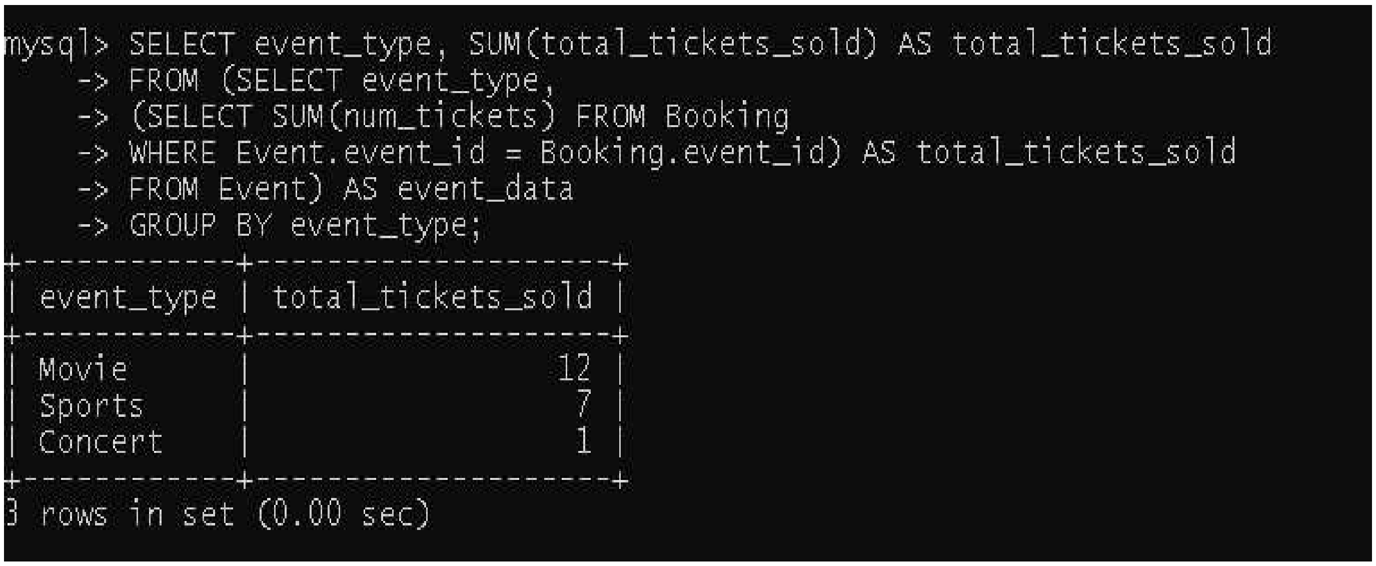
```
mysql> SELECT customer_id,customer_name,
    -> (SELECT SUM(total_cost) FROM Booking
    -> WHERE Customer.customer_id = Booking.customer_id) AS total_revenue
    -> FROM Customer;
+-------------+---------------+---------------+
| customer_id | customer_name | total_revenue |
+-------------+---------------+---------------+
|           1 | raj           |        100.00 |
|           2 | ram           |        300.00 |
|           3 | ravi          |         80.00 |
|           4 | tarun         |        360.00 |
|           5 | kavya         |        130.00 |
|           6 | jimmy         |        375.00 |
|           7 | jack          |        180.00 |
|           8 | rose          |        140.00 |
|           9 | ramya         |        120.00 |
|          10 | kevin         |        300.00 |
+-------------+---------------+---------------+
10 rows in set (0.00 sec)
```

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



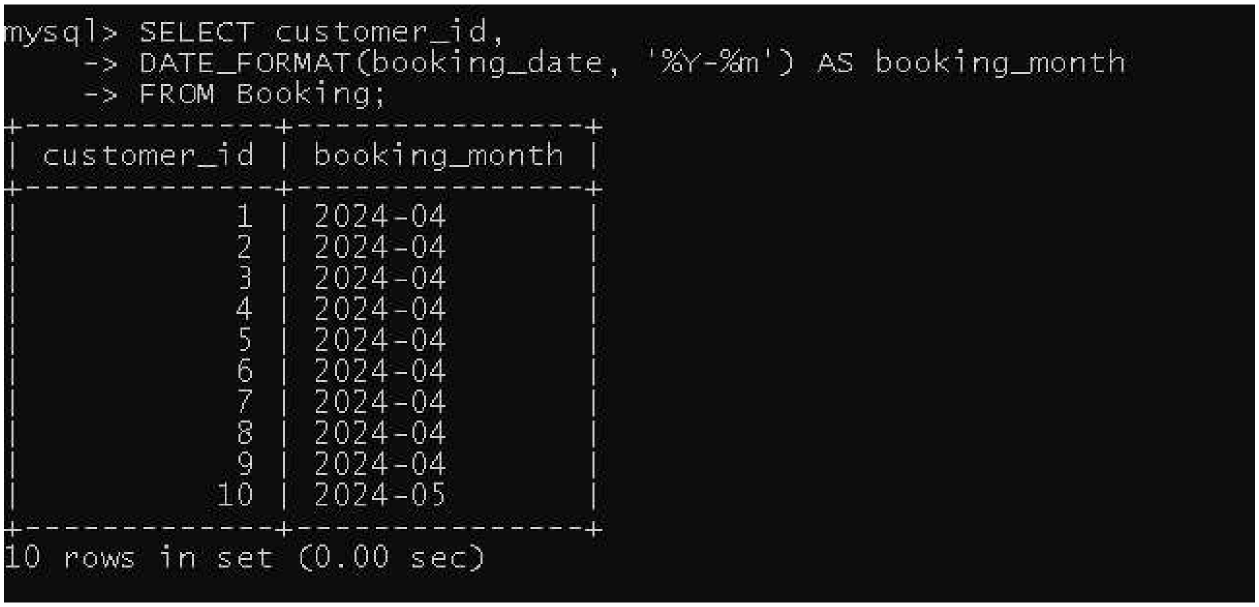
```
mysql> SELECT customer_id,customer_name
    -> FROM customer WHERE customer_id IN(
    -> SELECT DISTINCT customer_id FROM booking
    -> WHERE event_id IN(
    -> SELECT event_id FROM event
    -> WHERE venue_id='2'));
+-------------+---------------+
| customer_id | customer_name |
+-------------+---------------+
|           6 | jimmy         |
+-------------+---------------+
1 row in set (0.00 sec)
```

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

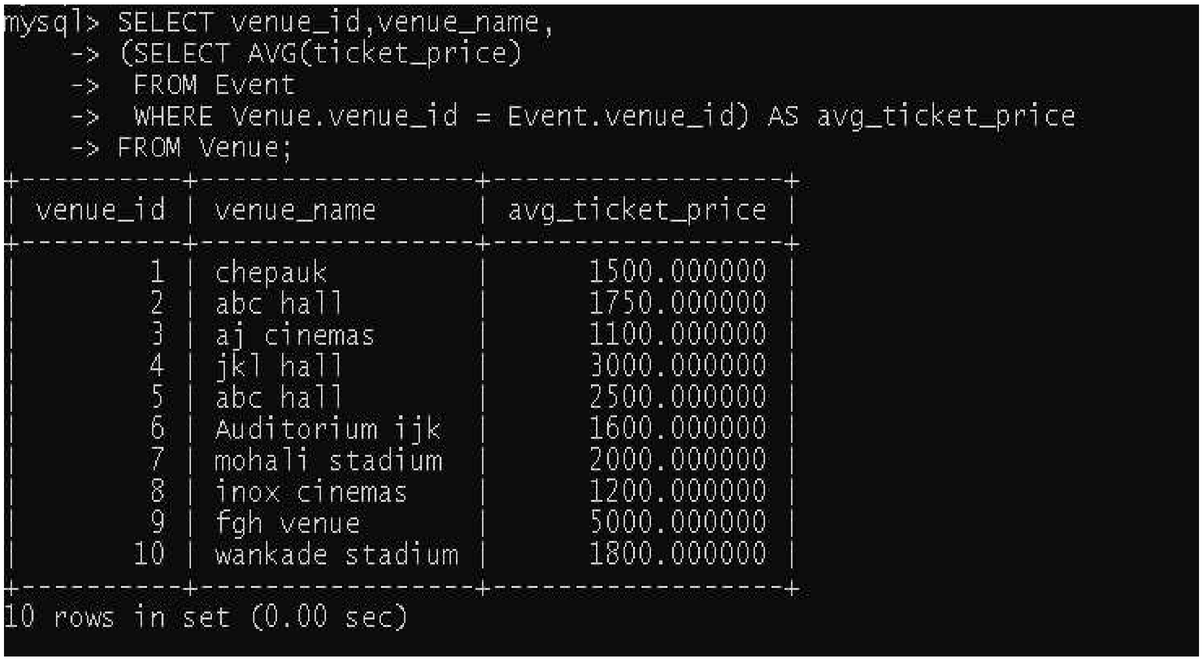


```
mysql> SELECT event_type, SUM(total_tickets_sold) AS total_tickets_sold
    -> FROM (SELECT event_type,
    -> (SELECT SUM(num_tickets) FROM Booking
    -> WHERE Event.event_id = Booking.event_id) AS total_tickets_sold
    -> FROM Event) AS event_data
    -> GROUP BY event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Movie      |                 12 |
| Sports     |                  7 |
| Concert    |                  1 |
+------------+--------------------+
3 rows in set (0.00 sec)
```

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


```
mysql> SELECT customer_id,
    -> DATE_FORMAT(booking_date, '%Y-%m') AS booking_month
    -> FROM Booking;
+-------------+---------------+
| customer_id | booking_month |
+-------------+---------------+
|           1 | 2024-04       |
|           2 | 2024-04       |
|           3 | 2024-04       |
|           4 | 2024-04       |
|           5 | 2024-04       |
|           6 | 2024-04       |
|           7 | 2024-04       |
|           8 | 2024-04       |
|           9 | 2024-04       |
|          10 | 2024-05       |
+-------------+---------------+
10 rows in set (0.00 sec)
```

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


```
mysql> SELECT venue_id,venue_name,
    -> (SELECT AVG(ticket_price)
    ->  FROM Event
    ->  WHERE Venue.venue_id = Event.venue_id) AS avg_ticket_price
    -> FROM Venue;
+----------+-----------------+------------------+
| venue_id | venue_name      | avg_ticket_price |
+----------+-----------------+------------------+
|        1 | chepauk         |      1500.000000 |
|        2 | abc hall        |      1750.000000 |
|        3 | aj cinemas      |      1100.000000 |
|        4 | jkl hall        |      3000.000000 |
|        5 | abc hall        |      2500.000000 |
|        6 | Auditorium ijk  |      1600.000000 |
|        7 | mohali stadium  |      2000.000000 |
|        8 | inox cinemas    |      1200.000000 |
|        9 | fgh venue       |      5000.000000 |
|       10 | wankade stadium |      1800.000000 |
+----------+-----------------+------------------+
10 rows in set (0.00 sec)
```