Assignment - 5 Ticket Booking System

Tasks 1: Database Design:

1. Create the database named "TicketBookingSystem"

```
mysql> create database TicketBookingSystem;
Query OK, 1 row affected (0.03 sec)

mysql> show databases;
+---------------------------+
| Database                  |
+---------------------------+
| careerhub                 |
| courier_management_system |
| d1                        |
| hexaware                  |
| hmbank                    |
| information_schema        |
| mysql                     |
| performance_schema        |
| petpals                   |
| sakila                    |
| sisdb                     |
| subquery                  |
| sys                       |
| techshop                  |
| ticketbookingsystem       |
| world                     |
+---------------------------+
16 rows in set (0.02 sec)

mysql> use TicketBookingSystem;
Database changed
mysql>
```

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

- Venu

```
Database changed
mysql> CREATE TABLE Venu (
    ->     venue_id INT PRIMARY KEY,
    ->     venue_name VARCHAR(50),
    ->     address VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> desc venue;
ERROR 1146 (42S02): Table 'ticketbookingsystem.venue' doesn't exist
mysql> desc venu;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| venue_id   | int          | NO   | PRI | NULL    |       |
| venue_name | varchar(50)  | YES  |     | NULL    |       |
| address    | varchar(100) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

• Event

```
mysql> desc event;
+----------------+---------------------------------+------+-----+---------+-------+
| Field          | Type                            | Null | Key | Default | Extra |
+----------------+---------------------------------+------+-----+---------+-------+
| event_id       | int                             | NO   | PRI | NULL    |       |
| event_name     | varchar(50)                     | YES  |     | NULL    |       |
| event_date     | date                            | YES  |     | NULL    |       |
| event_time     | time                            | YES  |     | NULL    |       |
| venue_id       | int                             | YES  | MUL | NULL    |       |
| total_seats    | int                             | YES  |     | NULL    |       |
| available_seats | int                            | YES  |     | NULL    |       |
| ticket_price   | decimal(10,2)                   | YES  |     | NULL    |       |
| event_type     | enum('Movie','Sports','Concert') | YES  |     | NULL    |       |
| booking_id     | int                             | YES  | MUL | NULL    |       |
+----------------+---------------------------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

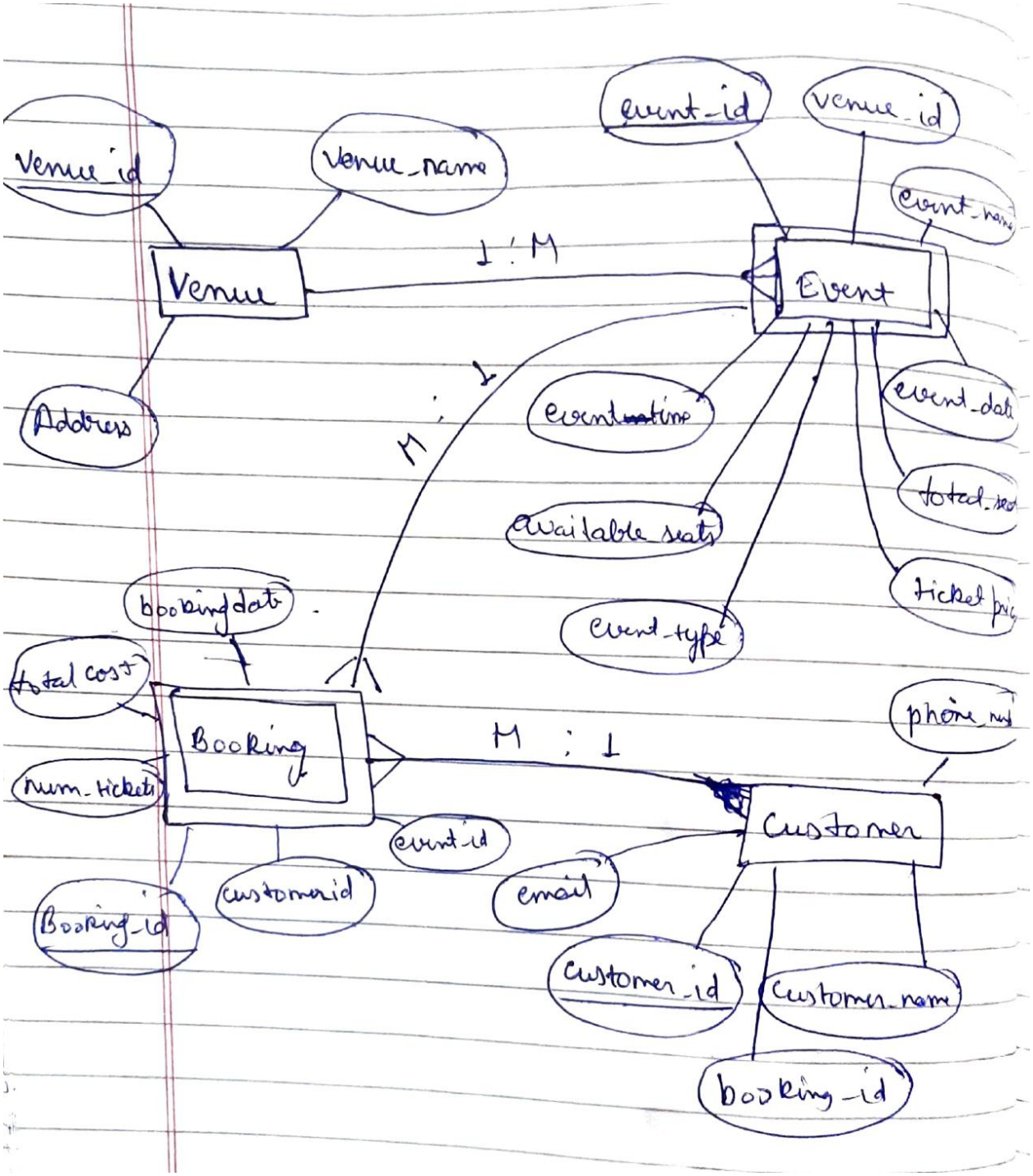
• Customers

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

• Booking

```
mysql> desc booking;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| booking_id   | int           | NO   | PRI | NULL    |       |
| customer_id  | int           | YES  | MUL | NULL    |       |
| event_id     | int           | YES  | MUL | NULL    |       |
| num_tickets  | int           | YES  |     | NULL    |       |
| total_cost   | decimal(10,2) | YES  |     | NULL    |       |
| booking_date | date          | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

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



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

Already done above

Tasks 2:

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

```
mysql> INSERT INTO Venu (venue_id, venue_name, address)
    -> VALUES
    ->     (1, 'Venue A', 'Address A1'),
    ->     (2, 'Venue B', 'Address B2'),
    ->     (3, 'Venue C', 'Address C3'),
    ->     (4, 'Venue D', 'Address D4'),
    ->     (5, 'Venue E', 'Address E5'),
    ->     (6, 'Venue F', 'Address F6'),
    ->     (7, 'Venue G', 'Address G7'),
    ->     (8, 'Venue H', 'Address H8'),
    ->     (9, 'Venue I', 'Address I9'),
    ->     (10, 'Venue J', 'Address J10');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from venu;
+----------+------------+-------------+
| venue_id | venue_name | address     |
+----------+------------+-------------+
|        1 | Venue A    | Address A1  |
|        2 | Venue B    | Address B2  |
|        3 | Venue C    | Address C3  |
|        4 | Venue D    | Address D4  |
|        5 | Venue E    | Address E5  |
|        6 | Venue F    | Address F6  |
|        7 | Venue G    | Address G7  |
|        8 | Venue H    | Address H8  |
|        9 | Venue I    | Address I9  |
|       10 | Venue J    | Address J10 |
+----------+------------+-------------+
10 rows in set (0.00 sec)
```

Event

```
mysql> select * from event;
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
| event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
|        1 | Event X    | 2023-12-15 | 18:00:00   |        1 |         100 |             80 |        20.00 | Concert    |          7 |
|        2 | Event Y    | 2023-12-20 | 15:30:00   |        2 |         150 |            120 |        15.00 | Movie      |          9 |
|        3 | Event Z    | 2023-12-25 | 20:00:00   |        3 |          80 |             60 |        25.00 | Sports     |          6 |
|        4 | Event P    | 2023-12-18 | 19:45:00   |        4 |         120 |            100 |        18.00 | Concert    |          4 |
|        5 | Event Q    | 2023-12-22 | 16:20:00   |        5 |          90 |             70 |        22.50 | Movie      |          8 |
|        6 | Event R    | 2023-12-28 | 21:15:00   |        6 |          70 |             50 |        30.00 | Sports     |          6 |
|        7 | Event S    | 2023-12-17 | 17:45:00   |        7 |         110 |             90 |        17.50 | Concert    |          7 |
|        8 | Event T    | 2023-12-23 | 14:00:00   |        8 |         130 |            110 |        14.00 | Movie      |          8 |
|        9 | Event U    | 2023-12-29 | 19:30:00   |        9 |          85 |             65 |        26.00 | Sports     |          9 |
|       10 | Event V    | 2023-12-21 | 22:10:00   |       10 |          75 |             55 |        19.50 | Concert    |         10 |
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
10 rows in set (0.00 sec)
```

Customer

```
mysql> select * from customer;
+-------------+-----------------+---------------------+--------------+------------+
| customer_id | customer_name   | email               | phone_number | booking_id |
+-------------+-----------------+---------------------+--------------+------------+
|           1 | John Doe        | john@example.com    | 123-456-7890 |          1 |
|           2 | Alice Smith     | alice@example.com   | 987-654-3210 |          2 |
|           3 | Bob Johnson     | bob@example.com     | 111-222-3333 |          3 |
|           4 | Emma Brown      | emma@example.com    | 444-555-6666 |          4 |
|           5 | James Wilson    | james@example.com   | 777-888-9999 |          5 |
|           6 | Sophia Lee      | sophia@example.com  | 333-999-1111 |          6 |
|           7 | William Davis   | william@example.com | 222-777-4444 |          7 |
|           8 | Olivia Garcia   | olivia@example.com  | 888-111-7777 |          8 |
|           9 | Michael Martinez | michael@example.com | 555-222-8888 |          9 |
|          10 | Ava Rodriguez   | ava@example.com     | 666-666-6666 |         10 |
+-------------+-----------------+---------------------+--------------+------------+
10 rows in set (0.00 sec)
```

Booking

```
mysql> select * from booking;
+------------+-------------+----------+-------------+------------+--------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
+------------+-------------+----------+-------------+------------+--------------+
|          1 |           1 |        1 |           2 |      40.00 | 2023-12-10   |
|          2 |           2 |        2 |           3 |      45.00 | 2023-12-12   |
|          3 |           3 |        3 |           1 |      25.00 | 2023-12-14   |
|          4 |           4 |        4 |           4 |      72.00 | 2023-12-16   |
|          5 |           5 |        5 |           2 |      45.00 | 2023-12-18   |
|          6 |           6 |        6 |           3 |      90.00 | 2023-12-20   |
|          7 |           7 |        7 |           1 |      17.50 | 2023-12-22   |
|          8 |           8 |        8 |           5 |      70.00 | 2023-12-24   |
|          9 |           9 |        9 |           2 |      52.00 | 2023-12-26   |
|         10 |          10 |       10 |           3 |      58.50 | 2023-12-28   |
+------------+-------------+----------+-------------+------------+--------------+
10 rows in set (0.00 sec)
```

2. Write a SQL query to list all Events.

```
mysql> select * from event;
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Event X    | 2023-12-15 | 18:00:00   |        1 |         100 |              80 |        20.00 | Concert    |          7 |
|        2 | Event Y    | 2023-12-20 | 15:30:00   |        2 |         150 |             120 |        15.00 | Movie      |          9 |
|        3 | Event Z    | 2023-12-25 | 20:00:00   |        3 |          80 |              60 |        25.00 | Sports     |          6 |
|        4 | Event P    | 2023-12-18 | 19:45:00   |        4 |         120 |             100 |        18.00 | Concert    |          4 |
|        5 | Event Q    | 2023-12-22 | 16:20:00   |        5 |          90 |              70 |        22.50 | Movie      |          8 |
|        6 | Event R    | 2023-12-28 | 21:15:00   |        6 |          70 |              50 |        30.00 | Sports     |          6 |
|        7 | Event S    | 2023-12-17 | 17:45:00   |        7 |         110 |              90 |        17.50 | Concert    |          7 |
|        8 | Event T    | 2023-12-23 | 14:00:00   |        8 |         130 |             110 |        14.00 | Movie      |          8 |
|        9 | Event U    | 2023-12-29 | 19:30:00   |        9 |          85 |              65 |        26.00 | Sports     |          9 |
|       10 | Event V    | 2023-12-21 | 22:10:00   |       10 |          75 |              55 |        19.50 | Concert    |         10 |
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
10 rows in set (0.00 sec)
```

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

```
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 | booking_id |
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Event X    | 2023-12-15 | 18:00:00   |        1 |         100 |              80 |        20.00 | Concert    |          7 |
|        2 | Event Y    | 2023-12-20 | 15:30:00   |        2 |         150 |             120 |        15.00 | Movie      |          9 |
|        3 | Event Z    | 2023-12-25 | 20:00:00   |        3 |          80 |              60 |        25.00 | Sports     |          6 |
|        4 | Event P    | 2023-12-18 | 19:45:00   |        4 |         120 |             100 |        18.00 | Concert    |          4 |
|        5 | Event Q    | 2023-12-22 | 16:20:00   |        5 |          90 |              70 |        22.50 | Movie      |          8 |
|        6 | Event R    | 2023-12-28 | 21:15:00   |        6 |          70 |              50 |        30.00 | Sports     |          6 |
|        7 | Event S    | 2023-12-17 | 17:45:00   |        7 |         110 |              90 |        17.50 | Concert    |          7 |
|        8 | Event T    | 2023-12-23 | 14:00:00   |        8 |         130 |             110 |        14.00 | Movie      |          8 |
|        9 | Event U    | 2023-12-29 | 19:30:00   |        9 |          85 |              65 |        26.00 | Sports     |          9 |
|       10 | Event V    | 2023-12-21 | 22:10:00   |       10 |          75 |              55 |        19.50 | Concert    |         10 |
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
10 rows in set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_name LIKE '%cup%';
Empty set (0.01 sec)
```

5. Write a 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;
Empty set (0.01 sec)
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_date BETWEEN '2023-12-01' AND '2023-12-31';
+----------+------------+------------+------------+----------+-------------+----------------+-------------+------------+------------+
| event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+------------+------------+------------+----------+-------------+----------------+-------------+------------+------------+
|        1 | Event X    | 2023-12-15 | 18:00:00   |        1 |         100 |             80 |        20.00 | Concert    |          7 |
|        2 | Event Y    | 2023-12-20 | 15:30:00   |        2 |         150 |            120 |        15.00 | Movie      |          9 |
|        3 | Event Z    | 2023-12-25 | 20:00:00   |        3 |          80 |             60 |        25.00 | Sports     |          6 |
|        4 | Event P    | 2023-12-18 | 19:45:00   |        4 |         120 |            100 |        18.00 | Concert    |          4 |
|        5 | Event Q    | 2023-12-22 | 16:20:00   |        5 |          90 |             70 |        22.50 | Movie      |          8 |
|        6 | Event R    | 2023-12-28 | 21:15:00   |        6 |          70 |             50 |        30.00 | Sports     |          6 |
|        7 | Event S    | 2023-12-17 | 17:45:00   |        7 |         110 |             90 |        17.50 | Concert    |          7 |
|        8 | Event T    | 2023-12-23 | 14:00:00   |        8 |         130 |            110 |        14.00 | Movie      |          8 |
|        9 | Event U    | 2023-12-29 | 19:30:00   |        9 |          85 |             65 |        26.00 | Sports     |          9 |
|       10 | Event V    | 2023-12-21 | 22:10:00   |       10 |          75 |             55 |        19.50 | Concert    |         10 |
+----------+------------+------------+------------+----------+-------------+----------------+-------------+------------+------------+
10 rows in set (0.00 sec)
```

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

```
10 rows in set (0.00 sec)

mysql> SELECT *
    -> FROM Event
    -> WHERE available_seats > 0
    -> AND event_name LIKE '%Concert%';
Empty set (0.00 sec)
```

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

```
ERROR 1146 (42S02): Table 'ticketbookingsystem.user' doesn't exist
mysql> SELECT *
    -> FROM customer
    -> ORDER BY customer_id
    -> LIMIT 5 OFFSET 5;
+-------------+------------------+---------------------+--------------+------------+
| customer_id | customer_name    | email               | phone_number | booking_id |
+-------------+------------------+---------------------+--------------+------------+
|           6 | Sophia Lee       | sophia@example.com  | 333-999-1111 |          6 |
|           7 | William Davis    | william@example.com | 222-777-4444 |          7 |
|           8 | Olivia Garcia    | olivia@example.com  | 888-111-7777 |          8 |
|           9 | Michael Martinez | michael@example.com | 555-222-8888 |          9 |
|          10 | Ava Rodriguez    | ava@example.com     | 666-666-6666 |         10 |
+-------------+------------------+---------------------+--------------+------------+
5 rows in set (0.00 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 |
+------------+-------------+----------+-------------+------------+--------------+
|          8 |           8 |        8 |           5 |      70.00 | 2023-12-24   |
+------------+-------------+----------+-------------+------------+--------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Customer
    -> WHERE phone_number LIKE '%000';
Empty set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE total_seats > 15000
    -> ORDER BY total_seats;
Empty set (0.00 sec)
```

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

```
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 | booking_id |
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
|        1 | Event X    | 2023-12-15 | 18:00:00   |        1 |         100 |             80 |        20.00 | Concert    |          7 |
|        2 | Event Y    | 2023-12-20 | 15:30:00   |        2 |         150 |            120 |        15.00 | Movie      |          9 |
|        3 | Event Z    | 2023-12-25 | 20:00:00   |        3 |          80 |             60 |        25.00 | Sports     |          6 |
|        4 | Event P    | 2023-12-18 | 19:45:00   |        4 |         120 |            100 |        18.00 | Concert    |          4 |
|        5 | Event Q    | 2023-12-22 | 16:20:00   |        5 |          90 |             70 |        22.50 | Movie      |          8 |
|        6 | Event R    | 2023-12-28 | 21:15:00   |        6 |          70 |             50 |        30.00 | Sports     |          6 |
|        7 | Event S    | 2023-12-17 | 17:45:00   |        7 |         110 |             90 |        17.50 | Concert    |          7 |
|        8 | Event T    | 2023-12-23 | 14:00:00   |        8 |         130 |            110 |        14.00 | Movie      |          8 |
|        9 | Event U    | 2023-12-29 | 19:30:00   |        9 |          85 |             65 |        26.00 | Sports     |          9 |
|       10 | Event V    | 2023-12-21 | 22:10:00   |       10 |          75 |             55 |        19.50 | Concert    |         10 |
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
```

Tasks 3

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

```
mysql> SELECT event_name, AVG(ticket_price) AS average_ticket_price
    -> FROM Event
    -> GROUP BY event_name;
+------------+---------------------+
| event_name | average_ticket_price |
+------------+---------------------+
| Event X    |           20.000000 |
| Event Y    |           15.000000 |
| Event Z    |           25.000000 |
| Event P    |           18.000000 |
| Event Q    |           22.500000 |
| Event R    |           30.000000 |
| Event S    |           17.500000 |
| Event T    |           14.000000 |
| Event U    |           26.000000 |
| Event V    |           19.500000 |
+------------+---------------------+
10 rows in set (0.01 sec)
```

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

```
mysql> SELECT SUM(total_cost) AS total_revenue_generated
    -> FROM Booking;
+-------------------------+
| total_revenue_generated |
+-------------------------+
|                  515.00 |
+-------------------------+
1 row in set (0.00 sec)
```

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

```
ERROR 1054 (42S22): Unknown column 'event_name' in 'field list'
mysql> SELECT e.event_id, e.event_name, MAX(b.num_tickets) AS max_tickets_sold
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY e.event_id, e.event_name
    -> ORDER BY max_tickets_sold DESC
    -> LIMIT 1;
+----------+------------+------------------+
| event_id | event_name | max_tickets_sold |
+----------+------------+------------------+
|        8 | Event T    |                5 |
+----------+------------+------------------+
1 row in set (0.00 sec)

mysql>
```

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

```
mysql> SELECT e.event_id, e.event_name, SUM(b.num_tickets) AS total_tickets_sold
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY e.event_id, e.event_name;
+----------+------------+--------------------+
| event_id | event_name | total_tickets_sold |
+----------+------------+--------------------+
|        1 | Event X    |                  2 |
|        2 | Event Y    |                  3 |
|        3 | Event Z    |                  1 |
|        4 | Event P    |                  4 |
|        5 | Event Q    |                  2 |
|        6 | Event R    |                  3 |
|        7 | Event S    |                  1 |
|        8 | Event T    |                  5 |
|        9 | Event U    |                  2 |
|       10 | Event V    |                  3 |
+----------+------------+--------------------+
10 rows in set (0.00 sec)
```

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

```
mysql> SELECT e.event_id, e.event_name
    -> FROM Event e
    -> LEFT JOIN Booking b ON e.event_id = b.event_id
    -> WHERE b.booking_id IS NULL;
Empty set (0.00 sec)
```

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

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

mysql> SELECT customer_id, COUNT(*) AS total_tickets_booked
    -> FROM Booking
    -> GROUP BY customer_id
    -> ORDER BY total_tickets_booked DESC;
+-------------+----------------------+
| customer_id | total_tickets_booked |
+-------------+----------------------+
|           1 |                    1 |
|           2 |                    1 |
|           3 |                    1 |
|           4 |                    1 |
|           5 |                    1 |
|           6 |                    1 |
|           7 |                    1 |
|           8 |                    1 |
|           9 |                    1 |
|          10 |                    1 |
+-------------+----------------------+
10 rows 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 e.event_id, e.event_name, MONTH(b.booking_date) AS month, SUM(b.num_tickets) AS total_tickets_sold
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY e.event_id, e.event_name, MONTH(b.booking_date)
    -> ORDER BY month, total_tickets_sold DESC;
+----------+------------+-------+--------------------+
| event_id | event_name | month | total_tickets_sold |
+----------+------------+-------+--------------------+
|        8 | Event T    |    12 |                  5 |
|        4 | Event P    |    12 |                  4 |
|        2 | Event Y    |    12 |                  3 |
|        6 | Event R    |    12 |                  3 |
|       10 | Event V    |    12 |                  3 |
|        1 | Event X    |    12 |                  2 |
|        5 | Event Q    |    12 |                  2 |
|        9 | Event U    |    12 |                  2 |
|        3 | Event Z    |    12 |                  1 |
|        7 | Event S    |    12 |                  1 |
+----------+------------+-------+--------------------+
10 rows in set (0.01 sec)
```

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

```
mysql> SELECT v.venue_id, v.venue_name, AVG(e.ticket_price) AS average_ticket_price
    -> FROM Venu v
    -> JOIN Event e ON v.venue_id = e.venue_id
    -> GROUP BY v.venue_id, v.venue_name;
+----------+------------+----------------------+
| venue_id | venue_name | average_ticket_price |
+----------+------------+----------------------+
|        1 | Venue A    |            20.000000 |
|        2 | Venue B    |            15.000000 |
|        3 | Venue C    |            25.000000 |
|        4 | Venue D    |            18.000000 |
|        5 | Venue E    |            22.500000 |
|        6 | Venue F    |            30.000000 |
|        7 | Venue G    |            17.500000 |
|        8 | Venue H    |            14.000000 |
|        9 | Venue I    |            26.000000 |
|       10 | Venue J    |            19.500000 |
+----------+------------+----------------------+
10 rows in set (0.01 sec)
```

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

```
mysql> SELECT e.event_type, SUM(b.num_tickets) AS total_tickets_sold
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY e.event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Concert    |                 10 |
| Movie      |                 10 |
| Sports     |                  6 |
+------------+--------------------+
3 rows in set (0.00 sec)
```

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

```
mysql> SELECT YEAR(b.booking_date) AS year, SUM(b.total_cost) AS total_revenue
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY YEAR(b.booking_date);
+------+---------------+
| year | total_revenue |
+------+---------------+
| 2023 |        515.00 |
+------+---------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT customer_id, COUNT(DISTINCT event_id) AS num_events_booked
    -> FROM Booking
    -> GROUP BY customer_id
    -> HAVING COUNT(DISTINCT event_id) > 1;
Empty set (0.01 sec)
```

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

```
mysql> SELECT b.customer_id, SUM(e.ticket_price * b.num_tickets) AS total_revenue
    -> FROM Booking b
    -> JOIN Event e ON b.event_id = e.event_id
    -> GROUP BY b.customer_id;
+-------------+---------------+
| customer_id | total_revenue |
+-------------+---------------+
|           1 |         40.00 |
|           2 |         45.00 |
|           3 |         25.00 |
|           4 |         72.00 |
|           5 |         45.00 |
|           6 |         90.00 |
|           7 |         17.50 |
|           8 |         70.00 |
|           9 |         52.00 |
|          10 |         58.50 |
+-------------+---------------+
10 rows in set (0.00 sec)
```

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

```
mysql> SELECT v.venue_id, v.venue_name, e.event_type, AVG(e.ticket_price) AS average_ticket_price
    -> FROM Venu v
    -> JOIN Event e ON v.venue_id = e.venue_id
    -> GROUP BY v.venue_id, v.venue_name, e.event_type;
+----------+------------+------------+----------------------+
| venue_id | venue_name | event_type | average_ticket_price |
+----------+------------+------------+----------------------+
|        1 | Venue A    | Concert    |            20.000000 |
|        2 | Venue B    | Movie      |            15.000000 |
|        3 | Venue C    | Sports     |            25.000000 |
|        4 | Venue D    | Concert    |            18.000000 |
|        5 | Venue E    | Movie      |            22.500000 |
|        6 | Venue F    | Sports     |            30.000000 |
|        7 | Venue G    | Concert    |            17.500000 |
|        8 | Venue H    | Movie      |            14.000000 |
|        9 | Venue I    | Sports     |            26.000000 |
|       10 | Venue J    | Concert    |            19.500000 |
+----------+------------+------------+----------------------+
10 rows in set (0.00 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 >= CURDATE() - INTERVAL 30 DAY
    -> GROUP BY customer_id;
+-------------+--------------------------+
| customer_id | total_tickets_purchased |
+-------------+--------------------------+
|           1 |                        2 |
|           2 |                        3 |
|           3 |                        1 |
|           4 |                        4 |
|           5 |                        2 |
|           6 |                        3 |
|           7 |                        1 |
|           8 |                        5 |
|           9 |                        2 |
|          10 |                        3 |
+-------------+--------------------------+
10 rows in set (0.00 sec)
```

Tasks- 4

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

```
mysql> SELECT v.venue_id, v.venue_name, AVG(e.ticket_price) AS average_ticket_price
    -> FROM Venu v
    -> JOIN (
    ->     SELECT venue_id, ticket_price
    ->     FROM Event
    -> ) AS e ON v.venue_id = e.venue_id
    -> GROUP BY v.venue_id, v.venue_name;
+----------+------------+----------------------+
| venue_id | venue_name | average_ticket_price |
+----------+------------+----------------------+
|        1 | Venue A    |            20.000000 |
|        2 | Venue B    |            15.000000 |
|        3 | Venue C    |            25.000000 |
|        4 | Venue D    |            18.000000 |
|        5 | Venue E    |            22.500000 |
|        6 | Venue F    |            30.000000 |
|        7 | Venue G    |            17.500000 |
|        8 | Venue H    |            14.000000 |
|        9 | Venue I    |            26.000000 |
|       10 | Venue J    |            19.500000 |
+----------+------------+----------------------+
```

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 Booking.event_id = Event.event_id
    -> ) > (total_seats * 0.5);
Empty set (0.00 sec)
```

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

```
mysql> SELECT e.event_id, e.event_name, SUM(b.num_tickets) AS total_tickets_sold
    -> FROM Event e
    -> JOIN Booking b ON e.event_id = b.event_id
    -> GROUP BY e.event_id, e.event_name;
+----------+------------+--------------------+
| event_id | event_name | total_tickets_sold |
+----------+------------+--------------------+
|        1 | Event X    |                  2 |
|        2 | Event Y    |                  3 |
|        3 | Event Z    |                  1 |
|        4 | Event P    |                  4 |
|        5 | Event Q    |                  2 |
|        6 | Event R    |                  3 |
|        7 | Event S    |                  1 |
|        8 | Event T    |                  5 |
|        9 | Event U    |                  2 |
|       10 | Event V    |                  3 |
+----------+------------+--------------------+
10 rows in set (0.00 sec)
```

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

```
mysql> SELECT customer_id
    -> FROM Customer c
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM Booking b
    ->     WHERE c.customer_id = b.customer_id
    -> );
Empty set (0.01 sec)

mysql>
```

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

```
mysql> SELECT e.event_type, COALESCE(b.total_tickets_sold, 0) AS total_tickets_sold
    -> FROM (
    ->     SELECT event_type, SUM(num_tickets) AS total_tickets_sold
    ->     FROM Event JOIN Booking ON Event.event_id = Booking.event_id
    ->     GROUP BY event_type
    -> ) AS b
    -> RIGHT JOIN (
    ->     SELECT DISTINCT event_type
    ->     FROM Event
    -> ) AS e ON b.event_type = e.event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Concert    |                 10 |
| Movie      |                 10 |
| Sports     |                  6 |
+------------+--------------------+
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 |
+----------+------------+--------------+
|        3 | Event Z    |        25.00 |
|        5 | Event Q    |        22.50 |
|        6 | Event R    |        30.00 |
|        9 | Event U    |        26.00 |
+----------+------------+--------------+
4 rows in set (0.00 sec)
```

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

```
mysql> SELECT b.customer_id, SUM(b.num_tickets * e.ticket_price) AS total_revenue
    -> FROM Booking b
    -> JOIN Event e ON b.event_id = e.event_id
    -> GROUP BY b.customer_id;
+-------------+---------------+
| customer_id | total_revenue |
+-------------+---------------+
|           1 |         40.00 |
|           2 |         45.00 |
|           3 |         25.00 |
|           4 |         72.00 |
|           5 |         45.00 |
|           6 |         90.00 |
|           7 |         17.50 |
|           8 |         70.00 |
|           9 |         52.00 |
|          10 |         58.50 |
+-------------+---------------+
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
    -> FROM Booking
    -> WHERE event_id IN (
    ->     SELECT event_id
    ->     FROM Event
    ->     WHERE venue_id = 3
    -> );
+-------------+
| customer_id |
+-------------+
|           3 |
+-------------+
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 e.event_type, SUM(COALESCE(b.total_tickets_sold, 0)) AS total_tickets_sold
    -> FROM (
    ->     SELECT event_id, SUM(num_tickets) AS total_tickets_sold
    ->     FROM Booking
    ->     GROUP BY event_id
    -> ) AS b
    -> RIGHT JOIN (
    ->     SELECT DISTINCT event_id, event_type
    ->     FROM Event
    -> ) AS e ON b.event_id = e.event_id
    -> GROUP BY e.event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Concert    |                 10 |
| Movie      |                 10 |
| Sports     |                  6 |
+------------+--------------------+
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 DISTINCT customer_id, MONTH(booking_date) AS month
    -> FROM Booking
    -> WHERE event_id IN (
    ->     SELECT event_id
    ->     FROM Event
    ->     WHERE MONTH(booking_date) = MONTH(Event.event_date)
    -> );
+-------------+-------+
| customer_id | month |
+-------------+-------+
|           1 |    12 |
|           2 |    12 |
|           3 |    12 |
|           4 |    12 |
|           5 |    12 |
|           6 |    12 |
|           7 |    12 |
|           8 |    12 |
|           9 |    12 |
|          10 |    12 |
+-------------+-------+
10 rows in set (0.01 sec)
```

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

```
ERROR 1146 (42S02): Table 'ticketbookingsystem.venue' doesn't exist
mysql> SELECT v.venue_id, v.venue_name, COALESCE(avg_price, 0) AS average_ticket_price
    -> FROM Venu v
    -> LEFT JOIN (
    ->     SELECT venue_id, AVG(ticket_price) AS avg_price
    ->     FROM Event
    ->     GROUP BY venue_id
    -> ) AS sub ON v.venue_id = sub.venue_id;
+----------+------------+---------------------+
| venue_id | venue_name | average_ticket_price |
+----------+------------+---------------------+
|        1 | Venue A    |           20.000000 |
|        2 | Venue B    |           15.000000 |
|        3 | Venue C    |           25.000000 |
|        4 | Venue D    |           18.000000 |
|        5 | Venue E    |           22.500000 |
|        6 | Venue F    |           30.000000 |
|        7 | Venue G    |           17.500000 |
|        8 | Venue H    |           14.000000 |
|        9 | Venue I    |           26.000000 |
|       10 | Venue J    |           19.500000 |
+----------+------------+---------------------+
10 rows in set (0.00 sec)
```