Assignment – 05

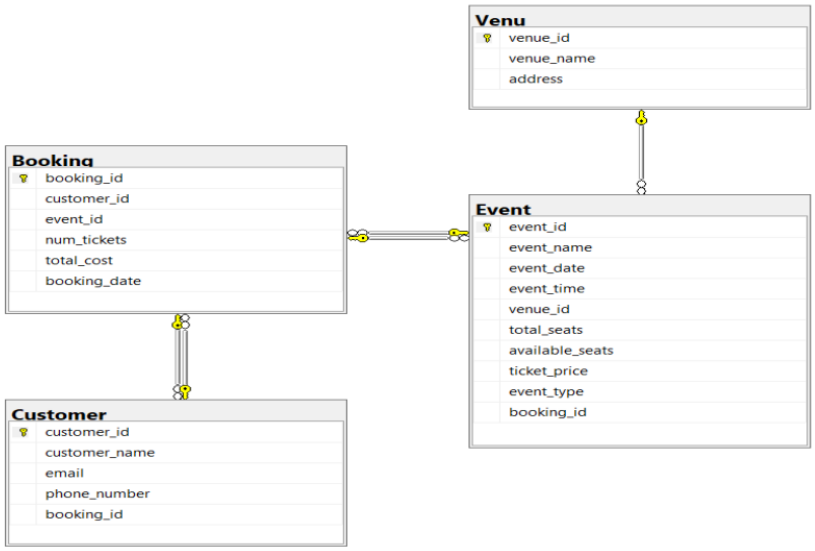
Task-1 Database Design:

1. Create the database named "TicketBookingSystem".

```
mysql> CREATE DATABASE TicketBookingSystem;
Query OK, 1 row affected (0.02 sec)

mysql> USE TicketBookingSystem;
Database changed
```

2. ERD.



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

   Venu:

```
mysql> CREATE TABLE Venu (
    ->     venue_id INT PRIMARY KEY,
    ->     venue_name VARCHAR(255),
    ->     address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

   Event:

```
mysql> CREATE TABLE Event (
    ->     event_id INT PRIMARY KEY,
    ->     event_name VARCHAR(255),
    ->     event_date DATE,
    ->     event_time TIME,
    ->     venue_id INT,
    ->     total_seats INT,
    ->     available_seats INT,
    ->     ticket_price DECIMAL(10, 2),
    ->     event_type VARCHAR(50),
    ->     booking_id INT
    -> );
Query OK, 0 rows affected (0.02 sec)
```

Customers:

```
mysql> CREATE TABLE Customer (
    ->     customer_id INT PRIMARY KEY,
    ->     customer_name VARCHAR(255),
    ->     email VARCHAR(255),
    ->     phone_number VARCHAR(15),
    ->     booking_id INT
    -> );
Query OK, 0 rows affected (0.02 sec)
```

Booking:

```
mysql> CREATE TABLE Booking (
    ->     booking_id INT PRIMARY KEY,
    ->     customer_id INT,
    ->     event_id INT,
    ->     num_tickets INT,
    ->     total_cost DECIMAL(10, 2),
    ->     booking_date DATE
    -> );
Query OK, 0 rows affected (0.02 sec)
```

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

```
mysql> ALTER TABLE Customer
    -> ADD FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> ALTER TABLE Event
    -> ADD CHECK (event_type IN ('Movie', 'Sports', 'Concert'));
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Task-2 Select, Where, Between, AND, LIKE:

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

Venu:

```
mysql> INSERT INTO Venu (venue_id, venue_name, address) VALUES
    -> (1, 'Grand Theater', '123 Main Street, Cityville'),
    -> (2, 'City Arena', '456 Center Avenue, Townsville'),
    -> (3, 'Sports Stadium', '789 Stadium Road, Sportstown'),
    -> (4, 'Film Palace', '101 Movie Lane, Cinemaville'),
    -> (5, 'Concert Hall', '202 Melody Street, Harmonytown'),
    -> (6, 'Community Center', '303 Social Square, Gatherburg'),
    -> (7, 'Live Lounge', '404 Entertainment Avenue, Showville'),
    -> (8, 'Cinematic Complex', '505 Film Street, Filmington'),
    -> (9, 'Soccer Park', '606 Goal Street, Kicksville'),
    -> (10, 'Music Dome', '707 Harmony Road, Concertburg');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Event:

```
mysql> INSERT INTO Event (event_id, event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_t
ype, booking_id) VALUES
    -> (1, 'Movie Night: Inception', '2023-01-15', '18:00:00', 1, 150, 120, 2220.00, 'Movie', NULL),
    -> (2, 'Concert: Acoustic Vibes', '2023-02-20', '20:00:00', 2, 300, 250, 1235.00, 'Concert', NULL),
    -> (3, 'Soccer Match: City Rivals', '2023-03-25', '19:30:00', 3, 200, 180, 1525.00, 'Sports', NULL),
    -> (4, 'Movie Night: The Great Gatsby', '2023-04-10', '21:00:00', 4, 120, 80, 1555.00, 'Movie', NULL),
    -> (5, 'Concert: Pop Explosion', '2023-05-05', '17:45:00', 5, 250, 200, 3330.00, 'Concert', NULL),
    -> (6, 'Live Music: Jazz Evening', '2023-06-12', '19:00:00', 6, 300, 280, 1440.00, 'Concert', NULL),
    -> (7, 'Basketball Game: Finals', '2023-07-08', '18:30:00', 7, 350, 300, 1330.00, 'Sports', NULL),
    -> (8, 'Movie Night: Casablanca', '2023-08-20', '20:15:00', 8, 150, 120, 2220.00, 'Movie', NULL),
    -> (9, 'Soccer Match: International Clash', '2023-09-18', '19:45:00', 9, 200, 180, 1225.00, 'Sports', NULL),
    -> (10, 'Concert: Rock Revolution', '2023-10-30', '22:00:00', 10, 250, 200, 3310.00, 'Concert', NULL);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Customers:

```
mysql> INSERT INTO Customer (customer_id, customer_name, email, phone_number, booking_id) VALUES
    -> (1, 'John Doe', 'john.doe@email.com', '555-1234', NULL),
    -> (2, 'Jane Smith', 'jane.smith@email.com', '555-5678', NULL),
    -> (3, 'Robert Johnson', 'robert.j@email.com', '555-9012', NULL),
    -> (4, 'Samantha Brown', 'samantha.b@email.com', '555-3456', NULL),
    -> (5, 'Chris Miller', 'chris.m@email.com', '555-7890', NULL),
    -> (6, 'Emma White', 'emma.w@email.com', '555-2345', NULL),
    -> (7, 'Michael Davis', 'michael.d@email.com', '555-6789', NULL),
    -> (8, 'Olivia Taylor', 'olivia.t@email.com', '555-1234', NULL),
    -> (9, 'Daniel Wilson', 'daniel.w@email.com', '555-5678', NULL),
    -> (10, 'Sophia Adams', 'sophia.a@email.com', '555-9012', NULL);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Booking:

```
mysql> INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date) VALUES
    -> (1, 1, 1, 2, 4440.00, '2023-01-15'),
    -> (2, 2, 2, 3, 3705.00, '2023-02-20'),
    -> (3, 3, 3, 1, 1525.00, '2023-03-25'),
    -> (4, 4, 4, 4, 6220.00, '2023-04-10'),
    -> (5, 5, 5, 2, 6660.00, '2023-05-05'),
    -> (6, 6, 6, 3, 4320.00, '2023-06-12'),
    -> (7, 7, 7, 5, 6650.00, '2023-07-08'),
    -> (8, 8, 8, 1, 2220.00, '2023-08-20'),
    -> (9, 9, 9, 2, 2450.00, '2023-09-18'),
    -> (10, 10, 10, 3, 9930.00, '2023-10-30');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

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 | Movie Night: Inception            | 2023-01-15 | 18:00:00   |        1 |         150 |             120 |      2220.00 | Movie      |          1 |
|        2 | Concert: Acoustic Vibes           | 2023-02-20 | 20:00:00   |        2 |         300 |             250 |      1235.00 | Concert    |          2 |
|        3 | Soccer Match: City Rivals         | 2023-03-25 | 19:30:00   |        3 |         200 |             180 |      1525.00 | Sports     |          3 |
|        4 | Movie Night: The Great Gatsby     | 2023-04-10 | 21:00:00   |        4 |         120 |              80 |      1555.00 | Movie      |          4 |
|        5 | Concert: Pop Explosion            | 2023-05-05 | 17:45:00   |        5 |         250 |             200 |      3330.00 | Concert    |          5 |
|        6 | Live Music: Jazz Evening          | 2023-06-12 | 19:00:00   |        6 |         300 |             280 |      1440.00 | Concert    |          6 |
|        7 | Basketball Game: Finals           | 2023-07-08 | 18:30:00   |        7 |         350 |             300 |      1330.00 | Sports     |          7 |
|        8 | Movie Night: Casablanca           | 2023-08-20 | 20:15:00   |        8 |         150 |             120 |      2220.00 | Movie      |          8 |
|        9 | Soccer Match: International Clash | 2023-09-18 | 19:45:00   |        9 |         200 |             180 |      1225.00 | Sports     |          9 |
|       10 | Concert: Rock Revolution          | 2023-10-30 | 22:00:00   |       10 |         250 |             200 |      3310.00 | 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 | Movie Night: Inception            | 2023-01-15 | 18:00:00   |        1 |         150 |             120 |      2220.00 | Movie      |          1 |
|        2 | Concert: Acoustic Vibes           | 2023-02-20 | 20:00:00   |        2 |         300 |             250 |      1235.00 | Concert    |          2 |
|        3 | Soccer Match: City Rivals         | 2023-03-25 | 19:30:00   |        3 |         200 |             180 |      1525.00 | Sports     |          3 |
|        4 | Movie Night: The Great Gatsby     | 2023-04-10 | 21:00:00   |        4 |         120 |              80 |      1555.00 | Movie      |          4 |
|        5 | Concert: Pop Explosion            | 2023-05-05 | 17:45:00   |        5 |         250 |             200 |      3330.00 | Concert    |          5 |
|        6 | Live Music: Jazz Evening          | 2023-06-12 | 19:00:00   |        6 |         300 |             280 |      1440.00 | Concert    |          6 |
|        7 | Basketball Game: Finals           | 2023-07-08 | 18:30:00   |        7 |         350 |             300 |      1330.00 | Sports     |          7 |
|        8 | Movie Night: Casablanca           | 2023-08-20 | 20:15:00   |        8 |         150 |             120 |      2220.00 | Movie      |          8 |
|        9 | Soccer Match: International Clash | 2023-09-18 | 19:45:00   |        9 |         200 |             180 |      1225.00 | Sports     |          9 |
|       10 | Concert: Rock Revolution          | 2023-10-30 | 22:00:00   |       10 |         250 |             200 |      3310.00 | 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 (0.00 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;
+----------+-----------------------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name                        | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+-----------------------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Movie Night: Inception            | 2023-01-15 | 18:00:00   |        1 |         150 |             120 |      2220.00 | Movie      |          1 |
|        2 | Concert: Acoustic Vibes           | 2023-02-20 | 20:00:00   |        2 |         300 |             250 |      1235.00 | Concert    |          2 |
|        3 | Soccer Match: City Rivals         | 2023-03-25 | 19:30:00   |        3 |         200 |             180 |      1525.00 | Sports     |          3 |
|        4 | Movie Night: The Great Gatsby     | 2023-04-10 | 21:00:00   |        4 |         120 |              80 |      1555.00 | Movie      |          4 |
|        6 | Live Music: Jazz Evening          | 2023-06-12 | 19:00:00   |        6 |         300 |             280 |      1440.00 | Concert    |          6 |
|        7 | Basketball Game: Finals           | 2023-07-08 | 18:30:00   |        7 |         350 |             300 |      1330.00 | Sports     |          7 |
|        8 | Movie Night: Casablanca           | 2023-08-20 | 20:15:00   |        8 |         150 |             120 |      2220.00 | Movie      |          8 |
|        9 | Soccer Match: International Clash | 2023-09-18 | 19:45:00   |        9 |         200 |             180 |      1225.00 | Sports     |          9 |
+----------+-----------------------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
8 rows in set (0.00 sec)
```

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

```
mysql> SELECT * FROM Event WHERE event_date BETWEEN '2023-01-01' AND '2023-05-31';
+----------+-----------------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name                  | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+-----------------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Movie Night: Inception      | 2023-01-15 | 18:00:00   |        1 |         150 |             120 |      2220.00 | Movie      |          1 |
|        2 | Concert: Acoustic Vibes     | 2023-02-20 | 20:00:00   |        2 |         300 |             250 |      1235.00 | Concert    |          2 |
|        3 | Soccer Match: City Rivals   | 2023-03-25 | 19:30:00   |        3 |         200 |             180 |      1525.00 | Sports     |          3 |
|        4 | Movie Night: The Great Gatsby | 2023-04-10 | 21:00:00   |        4 |         120 |              80 |      1555.00 | Movie      |          4 |
|        5 | Concert: Pop Explosion      | 2023-05-05 | 17:45:00   |        5 |         250 |             200 |      3330.00 | Concert    |          5 |
+----------+-----------------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
5 rows in set (0.00 sec)
```

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

```
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 | booking_id |
+----------+-----------------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        2 | Concert: Acoustic Vibes     | 2023-02-20 | 20:00:00   |        2 |         300 |             250 |      1235.00 | Concert    |          2 |
|        5 | Concert: Pop Explosion      | 2023-05-05 | 17:45:00   |        5 |         250 |             200 |      3330.00 | Concert    |          5 |
|       10 | Concert: Rock Revolution    | 2023-10-30 | 22:00:00   |       10 |         250 |             200 |      3310.00 | Concert    |         10 |
+----------+-----------------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
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 | Emma White    | emma.w@email.com    | 555-2345     |          6 |
|           7 | Michael Davis | michael.d@email.com | 555-6789     |          7 |
|           8 | Olivia Taylor | olivia.t@email.com  | 555-1234     |          8 |
|           9 | Daniel Wilson | daniel.w@email.com  | 555-5678     |          9 |
|          10 | Sophia Adams  | sophia.a@email.com  | 555-9012     |         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 |
+------------+-------------+----------+-------------+------------+--------------+
|          7 |           7 |        7 |           5 |    6650.00 | 2023-07-08   |
+------------+-------------+----------+-------------+------------+--------------+
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 NOT event_name LIKE '[x-z]%';
+----------+----------------------------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
| event_id | event_name                       | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+----------------------------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
|        1 | Movie Night: Inception           | 2023-01-15 | 18:00:00   |        1 |         150 |            120 |      2220.00 | Movie      |          1 |
|        2 | Concert: Acoustic Vibes          | 2023-02-20 | 20:00:00   |        2 |         300 |            250 |      1235.00 | Concert    |          2 |
|        3 | Soccer Match: City Rivals        | 2023-03-25 | 19:30:00   |        3 |         200 |            180 |      1525.00 | Sports     |          3 |
|        4 | Movie Night: The Great Gatsby    | 2023-04-10 | 21:00:00   |        4 |         120 |             80 |      1555.00 | Movie      |          4 |
|        5 | Concert: Pop Explosion           | 2023-05-05 | 17:45:00   |        5 |         250 |            200 |      3330.00 | Concert    |          5 |
|        6 | Live Music: Jazz Evening         | 2023-06-12 | 19:00:00   |        6 |         300 |            280 |      1440.00 | Concert    |          6 |
|        7 | Basketball Game: Finals          | 2023-07-08 | 18:30:00   |        7 |         350 |            300 |      1330.00 | Sports     |          7 |
|        8 | Movie Night: Casablanca          | 2023-08-20 | 20:15:00   |        8 |         150 |            120 |      2220.00 | Movie      |          8 |
|        9 | Soccer Match: International Clash | 2023-09-18 | 19:45:00   |        9 |         200 |            180 |      1225.00 | Sports     |          9 |
|       10 | Concert: Rock Revolution         | 2023-10-30 | 22:00:00   |       10 |         250 |            200 |      3310.00 | Concert    |         10 |
+----------+----------------------------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
10 rows in set (0.00 sec)
```

Task-3 Aggregate functions, Having, Order By, Group By and Joins:

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 |
+----------------------------------+---------------------+
| Movie Night: Inception           |          2220.000000 |
| Concert: Acoustic Vibes          |          1235.000000 |
| Soccer Match: City Rivals        |          1525.000000 |
| Movie Night: The Great Gatsby    |          1555.000000 |
| Concert: Pop Explosion           |          3330.000000 |
| Live Music: Jazz Evening         |          1440.000000 |
| Basketball Game: Finals          |          1330.000000 |
| Movie Night: Casablanca          |          2220.000000 |
| Soccer Match: International Clash |          1225.000000 |
| Concert: Rock Revolution         |          3310.000000 |
+----------------------------------+---------------------+
10 rows in set (0.00 sec)
```

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

```
mysql> SELECT SUM(total_cost) AS total_revenue FROM Booking;
+---------------+
| total_revenue |
+---------------+
|      48120.00 |
+---------------+
1 row in set (0.00 sec)
```

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

```
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 |
+----------+--------------------+
|        7 |                  5 |
+----------+--------------------+
1 row in set (0.00 sec)
```

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

```
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 |                  3 |
|        3 |                  1 |
|        4 |                  4 |
|        5 |                  2 |
|        6 |                  3 |
|        7 |                  5 |
|        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 event_id, event_name FROM Event
    -> WHERE event_id NOT IN (SELECT DISTINCT event_id FROM Booking);
Empty set (0.00 sec)
```

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

```
mysql> SELECT c.customer_id, c.customer_name, COUNT(b.booking_id) AS total_tickets_booked
    -> FROM Customer c JOIN Booking b ON c.customer_id = b.customer_id
    -> GROUP BY c.customer_id, c.customer_name
    -> ORDER BY total_tickets_booked DESC LIMIT 1;
+-------------+---------------+---------------------+
| customer_id | customer_name | total_tickets_booked |
+-------------+---------------+---------------------+
|           1 | John Doe      |                   1 |
+-------------+---------------+---------------------+
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, event_id, SUM(num_tickets) AS total_tickets_sold
    -> FROM Booking GROUP BY MONTH(booking_date), event_id;
+-------+----------+--------------------+
| month | event_id | total_tickets_sold |
+-------+----------+--------------------+
|     1 |        1 |                  2 |
|     2 |        2 |                  3 |
|     3 |        3 |                  1 |
|     4 |        4 |                  4 |
|     5 |        5 |                  2 |
|     6 |        6 |                  3 |
|     7 |        7 |                  5 |
|     8 |        8 |                  1 |
|     9 |        9 |                  2 |
|    10 |       10 |                  3 |
+-------+----------+--------------------+
10 rows in set (0.00 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 | Grand Theater    |          2220.000000 |
|        2 | City Arena       |          1235.000000 |
|        3 | Sports Stadium   |          1525.000000 |
|        4 | Film Palace      |          1555.000000 |
|        5 | Concert Hall     |          3330.000000 |
|        6 | Community Center |          1440.000000 |
|        7 | Live Lounge      |          1330.000000 |
|        8 | Cinematic Complex |          2220.000000 |
|        9 | Soccer Park      |          1225.000000 |
|       10 | Music Dome       |          3310.000000 |
+----------+------------------+----------------------+
10 rows in set (0.00 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.event_id
    -> GROUP BY event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Movie      |                  7 |
| Concert    |                 11 |
| Sports     |                  8 |
+------------+--------------------+
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(booking_date) AS year, SUM(total_cost) AS total_revenue
    -> FROM Booking
    -> GROUP BY YEAR(booking_date);
+------+---------------+
| year | total_revenue |
+------+---------------+
| 2023 |      48120.00 |
+------+---------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT c.customer_id, c.customer_name
    -> FROM Customer c
    -> JOIN Booking b ON c.customer_id = b.customer_id
    -> GROUP BY c.customer_id, c.customer_name
    -> HAVING COUNT(DISTINCT b.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 c.customer_id, c.customer_name, SUM(total_cost) AS total_revenue
    -> FROM Customer c
    -> JOIN Booking b ON c.customer_id = b.customer_id
    -> GROUP BY c.customer_id, c.customer_name;
+-------------+---------------+---------------+
| customer_id | customer_name | total_revenue |
+-------------+---------------+---------------+
|           1 | John Doe      |       4440.00 |
|           2 | Jane Smith    |       3705.00 |
|           3 | Robert Johnson |       1525.00 |
|           4 | Samantha Brown |       6220.00 |
|           5 | Chris Miller  |       6660.00 |
|           6 | Emma White    |       4320.00 |
|           7 | Michael Davis |       6650.00 |
|           8 | Olivia Taylor |       2220.00 |
|           9 | Daniel Wilson |       2450.00 |
|          10 | Sophia Adams  |       9930.00 |
+-------------+---------------+---------------+
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 | Grand Theater    | Movie      |          2220.000000 |
|        2 | City Arena       | Concert    |          1235.000000 |
|        3 | Sports Stadium   | Sports     |          1525.000000 |
|        4 | Film Palace      | Movie      |          1555.000000 |
|        5 | Concert Hall     | Concert    |          3330.000000 |
|        6 | Community Center | Concert    |          1440.000000 |
|        7 | Live Lounge      | Sports     |          1330.000000 |
|        8 | Cinematic Complex | Movie      |          2220.000000 |
|        9 | Soccer Park      | Sports     |          1225.000000 |
|       10 | Music Dome       | Concert    |          3310.000000 |
+----------+------------------+------------+----------------------+
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 c.customer_id, c.customer_name, COUNT(b.booking_id) AS total_tickets_purchased
    -> FROM Customer c
    -> JOIN Booking b ON c.customer_id = b.customer_id
    -> WHERE b.booking_date >= DATE_SUB(NOW(), INTERVAL 30 DAY)
    -> GROUP BY c.customer_id, c.customer_name;
+-------------+---------------+-------------------------+
| customer_id | customer_name | total_tickets_purchased |
+-------------+---------------+-------------------------+
|           1 | John Doe      |                       1 |
|           2 | Jane Smith    |                       1 |
+-------------+---------------+-------------------------+
2 rows in set (0.00 sec)
```

Task-4 Subquery and its types:

1. 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_id = v.venue_id
    -> ) AS average_ticket_price
    -> FROM Venu v;
+----------+------------------+----------------------+
| venue_id | venue_name       | average_ticket_price |
+----------+------------------+----------------------+
|        1 | Grand Theater    |          2220.000000 |
|        2 | City Arena       |          1235.000000 |
|        3 | Sports Stadium   |          1525.000000 |
|        4 | Film Palace      |          1555.000000 |
|        5 | Concert Hall     |          3330.000000 |
|        6 | Community Center |          1440.000000 |
|        7 | Live Lounge      |          1330.000000 |
|        8 | Cinematic Complex |          2220.000000 |
|        9 | Soccer Park      |          1225.000000 |
|       10 | Music Dome       |          3310.000000 |
+----------+------------------+----------------------+
10 rows in set (0.00 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 Booking.event_id = Event.event_id
    -> ) > 0.5 * total_seats;
Empty set (0.00 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 Booking.event_id = Event.event_id
    -> ) AS total_tickets_sold
    -> FROM Event;
+----------+----------------------------------+--------------------+
| event_id | event_name                       | total_tickets_sold |
+----------+----------------------------------+--------------------+
|        1 | Movie Night: Inception           |                  4 |
|        2 | Concert: Acoustic Vibes          |                  6 |
|        3 | Soccer Match: City Rivals        |                  1 |
|        4 | Movie Night: The Great Gatsby    |                  4 |
|        5 | Concert: Pop Explosion           |                  2 |
|        6 | Live Music: Jazz Evening         |                  3 |
|        7 | Basketball Game: Finals          |                  5 |
|        8 | Movie Night: Casablanca          |                  1 |
|        9 | Soccer Match: International Clash |                  2 |
|       10 | Concert: Rock Revolution         |                  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, customer_name
    -> FROM Customer c
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM Booking b
    ->     WHERE b.customer_id = c.customer_id
    -> );
Empty set (0.00 sec)
```

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 event_type, SUM(total_tickets_sold) AS total_tickets_sold
    -> FROM (
    ->     SELECT event_type, event_id, (
    ->         SELECT SUM(num_tickets)
    ->         FROM Booking
    ->         WHERE Booking.event_id = Event.event_id
    ->     ) AS total_tickets_sold
    ->     FROM Event
    -> ) AS Subquery
    -> GROUP BY event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Movie      |                  9 |
| Concert    |                 14 |
| Sports     |                  8 |
+------------+--------------------+
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 |
+----------+-------------------------+--------------+
|        1 | Movie Night: Inception  |      2220.00 |
|        5 | Concert: Pop Explosion  |      3330.00 |
|        8 | Movie Night: Casablanca |      2220.00 |
|       10 | Concert: Rock Revolution|      3310.00 |
+----------+-------------------------+--------------+
4 rows in set (0.00 sec)
```

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

```
mysql> SELECT customer_id, customer_name, (
    ->     SELECT SUM(total_cost)
    ->     FROM Booking
    ->     WHERE Booking.customer_id = c.customer_id
    -> ) AS total_revenue
    -> FROM Customer c;
+-------------+---------------+---------------+
| customer_id | customer_name | total_revenue |
+-------------+---------------+---------------+
|           1 | John Doe      |       8880.00 |
|           2 | Jane Smith    |       7410.00 |
|           3 | Robert Johnson|       1525.00 |
|           4 | Samantha Brown|       6220.00 |
|           5 | Chris Miller  |       6660.00 |
|           6 | Emma White    |       4320.00 |
|           7 | Michael Davis |       6650.00 |
|           8 | Olivia Taylor |       2220.00 |
|           9 | Daniel Wilson |       2450.00 |
|          10 | Sophia Adams  |       9930.00 |
+-------------+---------------+---------------+
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 EXISTS (
    ->     SELECT 1
    ->     FROM Booking
    ->     JOIN Event ON Booking.event_id = Event.event_id
    ->     WHERE Event.venue_id = 1
    ->     AND Booking.customer_id = Customer.customer_id
    -> );
+-------------+---------------+
| customer_id | customer_name |
+-------------+---------------+
|           1 | John Doe      |
+-------------+---------------+
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 Booking.event_id = Event.event_id
    ->     ) AS total_tickets_sold
    ->     FROM Event
    -> ) AS Subquery
    -> GROUP BY event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Movie      |                  9 |
| Concert    |                 14 |
| Sports     |                  8 |
+------------+--------------------+
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, customer_name
    -> FROM Customer c
    -> WHERE EXISTS (
    ->     SELECT *
    ->     FROM Booking b
    ->     WHERE b.customer_id = c.customer_id
    ->     AND FORMAT(b.booking_date, 'yyyy-MM') = '2023-03'
    -> );
Empty set, 12 warnings (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_id = v.venue_id
    -> ) AS average_ticket_price
    -> FROM Venu v;
+----------+------------------+----------------------+
| venue_id | venue_name       | average_ticket_price |
+----------+------------------+----------------------+
|        1 | Grand Theater    |          2220.000000 |
|        2 | City Arena       |          1235.000000 |
|        3 | Sports Stadium   |          1525.000000 |
|        4 | Film Palace      |          1555.000000 |
|        5 | Concert Hall     |          3330.000000 |
|        6 | Community Center |          1440.000000 |
|        7 | Live Lounge      |          1330.000000 |
|        8 | Cinematic Complex |          2220.000000 |
|        9 | Soccer Park      |          1225.000000 |
|       10 | Music Dome       |          3310.000000 |
+----------+------------------+----------------------+
10 rows in set (0.00 sec)
```