# Ticket Booking Assignment SQL

## By Popuri Keerthika

Tasks 1: Database Design: 1. Create the database named "TicketBooking"

```
mysql> create database ticketbooking;
Query OK, 1 row affected (0.02 sec)
```

2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships. • Venu • Event • Customers • Booking

```
mysql> CREATE TABLE Venu (
    ->     venue_id INT PRIMARY KEY,
    ->     venue_name VARCHAR(255) NOT NULL,
    ->     address VARCHAR(255) NOT NULL
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> desc venu;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| venue_id   | int          | NO   | PRI | NULL    |       |
| venue_name | varchar(255) | NO   |     | NULL    |       |
| address    | varchar(255) | NO   |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

```
mysql> CREATE TABLE Event (
    ->     event_id INT PRIMARY KEY,
    ->     event_name VARCHAR(255) NOT NULL,
    ->     event_date DATE,
    ->     event_time TIME,
    ->     venue_id INT,
    ->     total_seats INT,
    ->     available_seats INT,
    ->     ticket_price DECIMAL(10,2),
    ->     event_type ENUM('Movie', 'Sports', 'Concert'),
    ->     booking_id INT,
    ->     FOREIGN KEY (venue_id) REFERENCES Venu(venue_id)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> desc Event;
+-----------------+---------------------------------+------+-----+-------
--+-------+
| Field           | Type                            | Null | Key | Defaul
t | Extra |
+-----------------+---------------------------------+------+-----+-------
--+-------+
| event_id        | int                             | NO   | PRI | NULL
  |       |
| event_name      | varchar(255)                    | NO   |     | 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  |     | NULL
  |       |
+-----------------+---------------------------------+------+-----+-------
--+-------+
```

```
mysql> CREATE TABLE Customer (
    ->     customer_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     customer_name VARCHAR(255) NOT NULL,
    ->     email VARCHAR(255),
    ->     phone_number VARCHAR(15),
    ->     booking_id INT
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> desc customer;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| customer_id   | int         | NO   | PRI | NULL    | auto_increment |
| customer_name | varchar(255) | NO   |     | NULL    |                |
| email         | varchar(255) | YES  |     | NULL    |                |
| phone_number  | varchar(15)  | YES  |     | NULL    |                |
| booking_id    | int         | YES  |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

```
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 TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ->     FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    ->     FOREIGN KEY (event_id) REFERENCES Event(event_id)
    -> );
Query OK, 0 rows affected (0.08 sec)
```

```
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 | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_
GENERATED |
+--------------+--------------+------+-----+-------------------+---------
----------+
6 rows in set (0.00 sec)
```

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

ysql> desc customer;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| customer_id   | int          | NO   | PRI | NULL    | auto_increment |
| customer_name | varchar(255) | NO   |     | NULL    |                |
| email         | varchar(255) | YES  |     | NULL    |                |
| phone_number  | varchar(15)  | YES  |     | NULL    |                |
| booking_id    | int          | YES  | MUL | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

```
mysql> ALTER TABLE Event
    -> ADD CONSTRAINT fk_event_booking_id
    -> FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc event;
```

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| event_id | int | NO | PRI | NULL | |
| event_name | varchar(255) | NO | | 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 | |

Tasks 2: Select, Where, Between, AND, LIKE: 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 1', 'Address 1'),
    -> (2, 'Venue 2', 'Address 2'),
    -> (3, 'Venue 3', 'Address 3'),
    -> (4, 'Venue 4', 'Address 4'),
    -> (5, 'Venue 5', 'Address 5'),
    -> (6, 'Venue 6', 'Address 6'),
    -> (7, 'Venue 7', 'Address 7'),
    -> (8, 'Venue 8', 'Address 8'),
    -> (9, 'Venue 9', 'Address 9'),
    -> (10, 'Venue 10', 'Address 10');
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0

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, 'Event 1', '2024-04-08', '14:00:00', 1, 200, 200, 25.00, 'Sports', NULL),
    -> (2, 'Event 2', '2024-04-09', '15:00:00', 2, 150, 100, 35.00, 'Concert', NULL),
    -> (3, 'Event 3', '2024-04-10', '16:00:00', 3, 300, 250, 20.00, 'Movie', NULL),
    -> (4, 'Event 4', '2024-04-11', '17:00:00', 4, 100, 50, 50.00, 'Concert', NULL),
    -> (5, 'Event 5', '2024-04-12', '18:00:00', 5, 250, 200, 30.00, 'Sports', NULL),
    -> (6, 'Event 6', '2024-04-13', '19:00:00', 6, 150, 100, 40.00, 'Concert', NULL),
    -> (7, 'Event 7', '2024-04-14', '20:00:00', 7, 200, 150, 45.00, 'Movie', NULL),
    -> (8, 'Event 8', '2024-04-15', '21:00:00', 8, 300, 250, 20.00, 'Concert', NULL),
    -> (9, 'Event 9', '2024-04-16', '22:00:00', 9, 100, 50, 60.00, 'Sports', NULL),
    -> (10, 'Event 10', '2024-04-17', '23:00:00', 10, 150, 100, 35.00, 'Concert', NULL);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO Customer (customer_id, customer_name, email, phone_number, booking_id
    -> VALUES
    -> (1, 'Rahul Kumar', 'rahul.kumar@gmail.com', '1234567890', NULL),
    -> (2, 'Priya Patel', 'priya.patel@gmail.com', '9876543210', NULL),
    -> (3, 'Amit Singh', 'amit.singh@gmail.com', '4561237890', NULL),
    -> (4, 'Divya Gupta', 'divya.gupta@gmail.com', '7890123456', NULL),
    -> (5, 'Anjali Sharma', 'anjali.sharma@gmail.com', '3216549870', NULL),
    -> (6, 'Rohit Mehta', 'rohit.mehta@gmail.com', '6549873210', NULL),
    -> (7, 'Pooja Verma', 'pooja.verma@gmail.com', '0123456789', NULL),
    -> (8, 'Karan Shah', 'karan.shah@gmail.com', '9870123456', NULL),
    -> (9, 'Neha Joshi', 'neha.joshi@gmail.com', '6543210987', NULL),
    -> (10, 'Sandeep Agrawal', 'sandeep.agrawal@gmail.com', '0129876543', NULL);
```

```
mysql> INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date) VALUES
    -> (1, 1, 1, 2, 50.00, '2024-04-07 10:00:00'),
    -> (2, 2, 2, 3, 105.00, '2024-04-08 11:00:00'),
    -> (3, 3, 3, 4, 80.00, '2024-04-09 12:00:00'),
    -> (4, 4, 4, 1, 50.00, '2024-04-10 13:00:00'),
    -> (5, 5, 5, 5, 150.00, '2024-04-11 14:00:00'),
    -> (6, 6, 6, 2, 80.00, '2024-04-12 15:00:00'),
    -> (7, 7, 7, 3, 135.00, '2024-04-13 16:00:00'),
    -> (8, 8, 8, 4, 80.00, '2024-04-14 17:00:00'),
    -> (9, 9, 9, 1, 60.00, '2024-04-15 18:00:00'),
    -> (10, 10, 10, 2, 70.00, '2024-04-16 19:00:00');
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 | Event 1    | 2024-04-08 | 14:00:00   |        1 |         200 |             200 |        25.00 | Sports     |       NULL
 |
|        2 | Event 2    | 2024-04-09 | 15:00:00   |        2 |         150 |             100 |        35.00 | Concert    |       NULL
 |
|        3 | Event 3    | 2024-04-10 | 16:00:00   |        3 |         300 |             250 |        20.00 | Movie      |       NULL
 |
|        4 | Event 4    | 2024-04-11 | 17:00:00   |        4 |         100 |              50 |        50.00 | Concert    |       NULL
 |
|        5 | Event 5    | 2024-04-12 | 18:00:00   |        5 |         250 |             200 |        30.00 | Sports     |       NULL
 |
|        6 | Event 6    | 2024-04-13 | 19:00:00   |        6 |         150 |             100 |        40.00 | Concert    |       NULL
 |
|        7 | Event 7    | 2024-04-14 | 20:00:00   |        7 |         200 |             150 |        45.00 | Movie      |       NULL
 |
|        8 | Event 8    | 2024-04-15 | 21:00:00   |        8 |         300 |             250 |        20.00 | Concert    |       NULL
 |
|        9 | Event 9    | 2024-04-16 | 22:00:00   |        9 |         100 |              50 |        60.00 | Sports     |       NULL
 |
|       10 | Event 10   | 2024-04-17 | 23:00:00   |       10 |         150 |             100 |        35.00 | Concert    |       NULL
 |
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+-----------
-+
```

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 1    | 2024-04-08 | 14:00:00   |        1 |         200 |             200 |        25.00 | Sports     |       NULL
 |
|        2 | Event 2    | 2024-04-09 | 15:00:00   |        2 |         150 |             100 |        35.00 | Concert    |       NULL
 |
|        3 | Event 3    | 2024-04-10 | 16:00:00   |        3 |         300 |             250 |        20.00 | Movie      |       NULL
 |
|        4 | Event 4    | 2024-04-11 | 17:00:00   |        4 |         100 |              50 |        50.00 | Concert    |       NULL
 |
|        5 | Event 5    | 2024-04-12 | 18:00:00   |        5 |         250 |             200 |        30.00 | Sports     |       NULL
 |
|        6 | Event 6    | 2024-04-13 | 19:00:00   |        6 |         150 |             100 |        40.00 | Concert    |       NULL
 |
|        7 | Event 7    | 2024-04-14 | 20:00:00   |        7 |         200 |             150 |        45.00 | Movie      |       NULL
 |
|        8 | Event 8    | 2024-04-15 | 21:00:00   |        8 |         300 |             250 |        20.00 | Concert    |       NULL
 |
|        9 | Event 9    | 2024-04-16 | 22:00:00   |        9 |         100 |              50 |        60.00 | Sports     |       NULL
 |
|       10 | Event 10   | 2024-04-17 | 23:00:00   |       10 |         150 |             100 |        35.00 | Concert    |       NULL
 |
```

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;
+----------+---------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name    | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+---------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|       11 | Taylor Concert | 2024-04-08 | 14:00:00   |        1 |         200 |             200 |      1500.00 | Concert    |       NULL |
+----------+---------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
1 row 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 '2024-04-01' AND '2024-04-30';
+----------+------------+------------+------------+----------+
-+
| event_id | event_name | event_date | event_time | venue_id |
 |
+----------+------------+------------+------------+----------+
-+
|        1 | Event 1    | 2024-04-08 | 14:00:00   |        1 |
 |
|        2 | Event 2    | 2024-04-09 | 15:00:00   |        2 |
 |
|        3 | Event 3    | 2024-04-10 | 16:00:00   |        3 |
 |
|        4 | Event 4    | 2024-04-11 | 17:00:00   |        4 |
 |
|        5 | Event 5    | 2024-04-12 | 18:00:00   |        5 |
 |
|        6 | Event 6    | 2024-04-13 | 19:00:00   |        6 |
 |
|        7 | Event 7    | 2024-04-14 | 20:00:00   |        7 |
 |
|        8 | Event 8    | 2024-04-15 | 21:00:00   |        8 |
 |
|        9 | Event 9    | 2024-04-16 | 22:00:00   |        9 |
 |
|       10 | Event 10   | 2024-04-17 | 23:00:00   |       10 |
```

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 |
+----------+---------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|       11 | Taylor Concert | 2024-04-08 | 14:00:00   |        1 |         200 |             200 |      1500.00 | Concert    |       NULL |
+----------+---------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
1 row 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
    -> ORDER BY customer_id
    -> LIMIT 5 OFFSET 5;
+-------------+---------------+--------------------------+--------------+------------+
| customer_id | customer_name | email                    | phone_number | booking_id |
+-------------+---------------+--------------------------+--------------+------------+
|           6 | Rohit Mehta   | rohit.mehta@gmail.com    | 6549873210   |       NULL |
|           7 | Pooja Verma   | pooja.verma@gmail.com    | 0123456789   |       NULL |
|           8 | Karan Shah    | karan.shah@gmail.com     | 9870123456   |       NULL |
|           9 | Neha Joshi    | neha.joshi@gmail.com     | 6543210987   |       NULL |
|          10 | Sandeep Agrawal | sandeep.agrawal@gmail.com | 0129876543 |       NULL |
+-------------+---------------+--------------------------+--------------+------------+
5 rows in set (0.00 sec)
```

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

```
ERROR 1062 (23000): Duplicate entry '1' for key 'customer.PRIMARY'
mysql> SELECT *
    -> FROM Booking
    -> WHERE num_tickets > 4;
+------------+-------------+----------+-------------+------------+---------------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date        |
+------------+-------------+----------+-------------+------------+---------------------+
|          5 |           5 |        5 |           5 |     150.00 | 2024-04-11 14:00:00 |
+------------+-------------+----------+-------------+------------+---------------------+
1 row in set (0.00 sec)
```

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

```
mysql> 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 |
+----------+----------------+------------+------------+----------+-------------+-----------------+-------------+------------+
|        1 | Event 1        | 2024-04-08 | 14:00:00   |        1 |         200 |             200 |        25.00 | Sports     |
|        2 | Event 2        | 2024-04-09 | 15:00:00   |        2 |         150 |             100 |        35.00 | Concert    |
|        3 | Event 3        | 2024-04-10 | 16:00:00   |        3 |         300 |             250 |        20.00 | Movie      |
|        4 | Event 4        | 2024-04-11 | 17:00:00   |        4 |         100 |              50 |        50.00 | Concert    |
|        5 | Event 5        | 2024-04-12 | 18:00:00   |        5 |         250 |             200 |        30.00 | Sports     |
|        6 | Event 6        | 2024-04-13 | 19:00:00   |        6 |         150 |             100 |        40.00 | Concert    |
|        7 | Event 7        | 2024-04-14 | 20:00:00   |        7 |         200 |             150 |        45.00 | Movie      |
|        8 | Event 8        | 2024-04-15 | 21:00:00   |        8 |         300 |             250 |        20.00 | Concert    |
|        9 | Event 9        | 2024-04-16 | 22:00:00   |        9 |         100 |              50 |        60.00 | Sports     |
|       10 | Event 10       | 2024-04-17 | 23:00:00   |       10 |         150 |             100 |        35.00 | Concert    |
|       11 | Taylor Concert | 2024-04-08 | 14:00:00   |        1 |         200 |             200 |      1500.00 | Concert    |
+----------+----------------+------------+------------+----------+-------------+-----------------+-------------+------------+
11 rows in set (0.00 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> SELECT event_name, AVG(ticket_price) AS average_ticket_price
    -> FROM Event
    -> GROUP BY event_name;
+----------------+----------------------+
| event_name     | average_ticket_price |
+----------------+----------------------+
| Event 1        |            25.000000 |
| Event 2        |            35.000000 |
| Event 3        |            20.000000 |
| Event 4        |            50.000000 |
| Event 5        |            30.000000 |
| Event 6        |            40.000000 |
| Event 7        |            45.000000 |
| Event 8        |            20.000000 |
| Event 9        |            60.000000 |
| Event 10       |            35.000000 |
| Taylor Concert |          1500.000000 |
+----------------+----------------------+
11 rows in set (0.01 sec)
```

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

```
mysql> SELECT event_name, SUM(total_cost) AS total_revenue
    -> FROM Booking
    -> JOIN Event ON Booking.event_id = Event.event_id
    -> GROUP BY event_name;
+------------+---------------+
| event_name | total_revenue |
+------------+---------------+
| Event 1    |         50.00 |
| Event 2    |        105.00 |
| Event 3    |         80.00 |
| Event 4    |         50.00 |
| Event 5    |        150.00 |
| Event 6    |         80.00 |
| Event 7    |        135.00 |
| Event 8    |         80.00 |
| Event 9    |         60.00 |
| Event 10   |         70.00 |
+------------+---------------+
10 rows in set (0.00 sec)
```

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

```
mysql> SELECT b.event_id, e.event_name, SUM(b.num_tickets) AS total_tickets_sold
    -> FROM Booking b
    -> JOIN Event e ON b.event_id = e.event_id
    -> GROUP BY b.event_id, e.event_name
    -> ORDER BY total_tickets_sold DESC
    -> LIMIT 1;
+----------+------------+--------------------+
| event_id | event_name | total_tickets_sold |
+----------+------------+--------------------+
|        5 | Event 5    |                  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.event_id, Event.event_name, SUM(Booking.num_tickets) AS total_tickets_sold
    -> FROM Event
    -> LEFT JOIN Booking ON Event.event_id = Booking.event_id
    -> GROUP BY Event.event_id, Event.event_name;
+----------+---------------+--------------------+
| event_id | event_name    | total_tickets_sold |
+----------+---------------+--------------------+
|        1 | Event 1       |                  2 |
|        2 | Event 2       |                  3 |
|        3 | Event 3       |                  4 |
|        4 | Event 4       |                  1 |
|        5 | Event 5       |                  5 |
|        6 | Event 6       |                  2 |
|        7 | Event 7       |                  3 |
|        8 | Event 8       |                  4 |
|        9 | Event 9       |                  1 |
|       10 | Event 10      |                  2 |
|       11 | Taylor Concert |               NULL |
+----------+---------------+--------------------+
11 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);
+----------+----------------+
| event_id | event_name     |
+----------+----------------+
|       11 | Taylor Concert |
+----------+----------------+
1 row in set (0.01 sec)
```

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

```
mysql> SELECT b.customer_id, c.customer_name, SUM(b.num_tickets) AS total_tickets_booked
    -> FROM Booking b
    -> JOIN Customer c ON b.customer_id = c.customer_id
    -> GROUP BY b.customer_id, c.customer_name
    -> ORDER BY total_tickets_booked DESC
    -> LIMIT 1;
+-------------+---------------+----------------------+
| customer_id | customer_name | total_tickets_booked |
+-------------+---------------+----------------------+
|           5 | Anjali Sharma |                    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(event_date) AS month, YEAR(event_date) AS year, event_name, SUM(num_tickets) AS total_tickets_sold
   -> FROM Booking b
   -> JOIN Event e ON b.event_id = e.event_id
   -> GROUP BY YEAR(event_date), MONTH(event_date), event_name;
+-------+------+------------+--------------------+
| month | year | event_name | total_tickets_sold |
+-------+------+------------+--------------------+
|     4 | 2024 | Event 1    |                  2 |
|     4 | 2024 | Event 2    |                  3 |
|     4 | 2024 | Event 3    |                  4 |
|     4 | 2024 | Event 4    |                  1 |
|     4 | 2024 | Event 5    |                  5 |
|     4 | 2024 | Event 6    |                  2 |
|     4 | 2024 | Event 7    |                  3 |
|     4 | 2024 | Event 8    |                  4 |
|     4 | 2024 | Event 9    |                  1 |
|     4 | 2024 | Event 10   |                  2 |
+-------+------+------------+--------------------+
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 Event e
    -> JOIN Venu v ON e.venue_id = v.venue_id
    -> GROUP BY v.venue_id, v.venue_name;
+----------+------------+----------------------+
| venue_id | venue_name | average_ticket_price |
+----------+------------+----------------------+
|        1 | Venue 1    |           762.500000 |
|        2 | Venue 2    |            35.000000 |
|        3 | Venue 3    |            20.000000 |
|        4 | Venue 4    |            50.000000 |
|        5 | Venue 5    |            30.000000 |
|        6 | Venue 6    |            40.000000 |
|        7 | Venue 7    |            45.000000 |
|        8 | Venue 8    |            20.000000 |
|        9 | Venue 9    |            60.000000 |
|       10 | Venue 10   |            35.000000 |
+----------+------------+----------------------+
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 e.event_type, SUM(b.num_tickets) AS total_tickets_sold
    -> FROM Booking b
    -> JOIN Event e ON b.event_id = e.event_id
    -> GROUP BY e.event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Sports     |                  8 |
| Concert    |                 12 |
| Movie      |                  7 |
+------------+--------------------+
3 rows in set (0.00 sec)
```

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

```
Database changed
mysql> SELECT YEAR(b.booking_date) AS year, SUM(b.total_cost) AS total_revenue
    -> FROM Booking b
    -> JOIN Event e ON b.event_id = e.event_id
    -> GROUP BY YEAR(b.booking_date);
+------+---------------+
| year | total_revenue |
+------+---------------+
| 2024 |        860.00 |
+------+---------------+
1 row in set (0.14 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 Booking b
    -> JOIN Customer c ON b.customer_id = c.customer_id
    -> GROUP BY c.customer_id, c.customer_name
    -> HAVING COUNT(DISTINCT b.event_id) > 1;
Empty set (0.01 sec)

mysql> select * from booking;
+------------+-------------+----------+-------------+------------+---------------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date        |
+------------+-------------+----------+-------------+------------+---------------------+
|          1 |           1 |        1 |           2 |      50.00 | 2024-04-07 10:00:00 |
|          2 |           2 |        2 |           3 |     105.00 | 2024-04-08 11:00:00 |
|          3 |           3 |        3 |           4 |      80.00 | 2024-04-09 12:00:00 |
|          4 |           4 |        4 |           1 |      50.00 | 2024-04-10 13:00:00 |
|          5 |           5 |        5 |           5 |     150.00 | 2024-04-11 14:00:00 |
|          6 |           6 |        6 |           2 |      80.00 | 2024-04-12 15:00:00 |
|          7 |           7 |        7 |           3 |     135.00 | 2024-04-13 16:00:00 |
|          8 |           8 |        8 |           4 |      80.00 | 2024-04-14 17:00:00 |
|          9 |           9 |        9 |           1 |      60.00 | 2024-04-15 18:00:00 |
|         10 |          10 |       10 |           2 |      70.00 | 2024-04-16 19:00:00 |
+------------+-------------+----------+-------------+------------+---------------------+
10 rows in 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(b.total_cost) AS total_revenue
    -> FROM Booking b
    -> JOIN Customer c ON b.customer_id = c.customer_id
    -> GROUP BY c.customer_id, c.customer_name;
+-------------+-----------------+---------------+
| customer_id | customer_name   | total_revenue |
+-------------+-----------------+---------------+
|           1 | Rahul Kumar     |         50.00 |
|           2 | Priya Patel     |        105.00 |
|           3 | Amit Singh      |         80.00 |
|           4 | Divya Gupta     |         50.00 |
|           5 | Anjali Sharma   |        150.00 |
|           6 | Rohit Mehta     |         80.00 |
|           7 | Pooja Verma     |        135.00 |
|           8 | Karan Shah      |         80.00 |
|           9 | Neha Joshi      |         60.00 |
|          10 | Sandeep Agrawal |         70.00 |
+-------------+-----------------+---------------+
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 e.event_type, v.venue_name, AVG(e.ticket_price) AS average_ticket_price
    -> FROM Event e
    -> JOIN Venu v ON e.venue_id = v.venue_id
    -> GROUP BY e.event_type, v.venue_name;
+------------+------------+---------------------+
| event_type | venue_name | average_ticket_price |
+------------+------------+---------------------+
| Sports     | Venue 1    |           25.000000 |
| Concert    | Venue 2    |           35.000000 |
| Movie      | Venue 3    |           20.000000 |
| Concert    | Venue 4    |           50.000000 |
| Sports     | Venue 5    |           30.000000 |
| Concert    | Venue 6    |           40.000000 |
| Movie      | Venue 7    |           45.000000 |
| Concert    | Venue 8    |           20.000000 |
| Sports     | Venue 9    |           60.000000 |
| Concert    | Venue 10   |           35.000000 |
| Concert    | Venue 1    |         1500.000000 |
+------------+------------+---------------------+
11 rows in set (0.06 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, SUM(b.num_tickets) AS total_tickets_purchased
    -> FROM Booking b
    -> JOIN Customer c ON b.customer_id = c.customer_id
    -> WHERE b.booking_date >= CURDATE() - INTERVAL 30 DAY
    -> GROUP BY c.customer_id, c.customer_name;
+-------------+-----------------+--------------------------+
| customer_id | customer_name   | total_tickets_purchased |
+-------------+-----------------+--------------------------+
|           1 | Rahul Kumar     |                        2 |
|           2 | Priya Patel     |                        3 |
|           3 | Amit Singh      |                        4 |
|           4 | Divya Gupta     |                        1 |
|           5 | Anjali Sharma   |                        5 |
|           6 | Rohit Mehta     |                        2 |
|           7 | Pooja Verma     |                        3 |
|           8 | Karan Shah      |                        4 |
|           9 | Neha Joshi      |                        1 |
|          10 | Sandeep Agrawal |                        2 |
+-------------+-----------------+--------------------------+
10 rows in set (0.00 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(e.ticket_price) FROM Event e WHERE e.venue_id = v.venue_id) AS avera
ge_ticket_price FROM Venu v;
+----------+------------+---------------------+
| venue_id | venue_name | average_ticket_price |
+----------+------------+---------------------+
|        1 | Venue 1    |          762.500000 |
|        2 | Venue 2    |           35.000000 |
|        3 | Venue 3    |           20.000000 |
|        4 | Venue 4    |           50.000000 |
|        5 | Venue 5    |           30.000000 |
|        6 | Venue 6    |           40.000000 |
|        7 | Venue 7    |           45.000000 |
|        8 | Venue 8    |           20.000000 |
|        9 | Venue 9    |           60.000000 |
|       10 | Venue 10   |           35.000000 |
+----------+------------+---------------------+
10 rows in set (0.06 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) > (total_seats * 0.5);
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 to
tal_tickets_sold FROM Event;
+----------+---------------+--------------------+
| event_id | event_name    | total_tickets_sold |
+----------+---------------+--------------------+
|        1 | Event 1       |                  2 |
|        2 | Event 2       |                  3 |
|        3 | Event 3       |                  4 |
|        4 | Event 4       |                  1 |
|        5 | Event 5       |                  5 |
|        6 | Event 6       |                  2 |
|        7 | Event 7       |                  3 |
|        8 | Event 8       |                  4 |
|        9 | Event 9       |                  1 |
|       10 | Event 10      |                  2 |
|       11 | Taylor Concert |               NULL |
+----------+---------------+--------------------+
11 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 * FROM Booking WHERE Booking.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);
+----------+----------------+
| event_id | event_name     |
+----------+----------------+
|       11 | Taylor Concert |
+----------+----------------+
1 row in 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, b.total_tickets_sold FROM (SELECT DISTINCT event_type FROM Event) e LEFT JOIN (SELECT event_
id, SUM(num_tickets) AS total_tickets_sold FROM Booking GROUP BY event_id) b ON e.event_type = (SELECT event_type FROM E
vent WHERE event_id = b.event_id);
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Sports     |                  1 |
| Sports     |                  5 |
| Sports     |                  2 |
| Concert    |                  2 |
| Concert    |                  4 |
| Concert    |                  2 |
| Concert    |                  1 |
| Concert    |                  3 |
| Movie      |                  3 |
| Movie      |                  4 |
+------------+--------------------+
10 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 |
+----------+----------------+--------------+
|       11 | Taylor Concert |      1500.00 |
+----------+----------------+--------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT
    ->     c.customer_id,
    ->     c.customer_name,
    ->     (SELECT SUM(total_cost) FROM Booking WHERE customer_id = c.customer_id) AS total_revenue
    -> FROM
    ->     Customer c;
+-------------+----------------+---------------+
| customer_id | customer_name  | total_revenue |
+-------------+----------------+---------------+
|           1 | Rahul Kumar    |         50.00 |
|           2 | Priya Patel    |        105.00 |
|           3 | Amit Singh     |         80.00 |
|           4 | Divya Gupta    |         50.00 |
|           5 | Anjali Sharma  |        150.00 |
|           6 | Rohit Mehta    |         80.00 |
|           7 | Pooja Verma    |        135.00 |
|           8 | Karan Shah     |         80.00 |
|           9 | Neha Joshi     |         60.00 |
|          10 | Sandeep Agrawal |         70.00 |
+-------------+----------------+---------------+
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 customer_id
    ->     FROM Booking
    ->     WHERE event_id IN (
    ->         SELECT event_id
    ->         FROM Event
    ->         WHERE venue_id = 1
    ->     )
    -> );
+-------------+---------------+
| customer_id | customer_name |
+-------------+---------------+
|           1 | Rahul Kumar   |
+-------------+---------------+
1 row in set (0.01 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(b.num_tickets) AS total_tickets_sold FROM Event e LEFT JOIN Booking b ON e.event_id = b.event_id GROU
P BY e.event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Sports     |                  8 |
| Concert    |                 12 |
| Movie      |                  7 |
+------------+--------------------+
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
    ->     c.customer_id,
    ->     c.customer_name,
    ->     DATE_FORMAT(booking_date, '%Y-%m') AS booking_month
    -> FROM
    ->     Booking b
    -> JOIN
    ->     Customer c ON b.customer_id = c.customer_id
    -> GROUP BY
    ->     c.customer_id,
    ->     c.customer_name,
    ->     DATE_FORMAT(booking_date, '%Y-%m');
+-------------+----------------+--------------+
| customer_id | customer_name  | booking_month |
+-------------+----------------+--------------+
|           1 | Rahul Kumar    | 2024-04      |
|           2 | Priya Patel    | 2024-04      |
|           3 | Amit Singh     | 2024-04      |
|           4 | Divya Gupta    | 2024-04      |
|           5 | Anjali Sharma  | 2024-04      |
|           6 | Rohit Mehta    | 2024-04      |
|           7 | Pooja Verma    | 2024-04      |
|           8 | Karan Shah     | 2024-04      |
|           9 | Neha Joshi     | 2024-04      |
|          10 | Sandeep Agrawal | 2024-04      |
+-------------+----------------+--------------+
10 rows in set (0.00 sec)
```

12. 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 average_ticket_price
    -> FROM
    ->     Venu v;
+----------+------------+----------------------+
| venue_id | venue_name | average_ticket_price |
+----------+------------+----------------------+
|        1 | Venue 1    |           762.500000 |
|        2 | Venue 2    |            35.000000 |
|        3 | Venue 3    |            20.000000 |
|        4 | Venue 4    |            50.000000 |
|        5 | Venue 5    |            30.000000 |
|        6 | Venue 6    |            40.000000 |
|        7 | Venue 7    |            45.000000 |
|        8 | Venue 8    |            20.000000 |
|        9 | Venue 9    |            60.000000 |
|       10 | Venue 10   |            35.000000 |
+----------+------------+----------------------+
10 rows in set (0.00 sec)
```