Assignment – 5

Tasks 1: Database Design:

1. Create the database named "TicketBookingSystem"

Ans.

CREATE DATABASE TicketBookingSystem;

USE TicketBookingSystem;

Command Prompt - mysql -u root -p

```
Empty set (0.00 sec)

mysql> CREATE DATABASE TicketBookingSystem;
Query OK, 1 row affected (0.04 sec)

mysql> USE TicketBookingSystem;
Database changed
mysql> select database();
+--------------------+
| database()         |
+--------------------+
| ticketbookingsystem |
+--------------------+
1 row in set (0.00 sec)

mysql>
```

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

Venue

Event

Customers

Booking

Ans.

-- Venu Table

CREATE TABLE Venu (

venue_id INT PRIMARY KEY,

venue_name VARCHAR(255),

address VARCHAR(255)

);

-- Event Table

```
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,
    event_type ENUM('Movie', 'Sports', 'Concert'),
    booking_id INT,
    FOREIGN KEY (venue_id) REFERENCES Venu(venue_id),
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
);

-- Customer Table
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    email VARCHAR(255),
    phone_number VARCHAR(20),
    booking_id INT,
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
);

-- Booking Table
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY,
    customer_id INT,
    event_id INT,
    num_tickets INT,
```

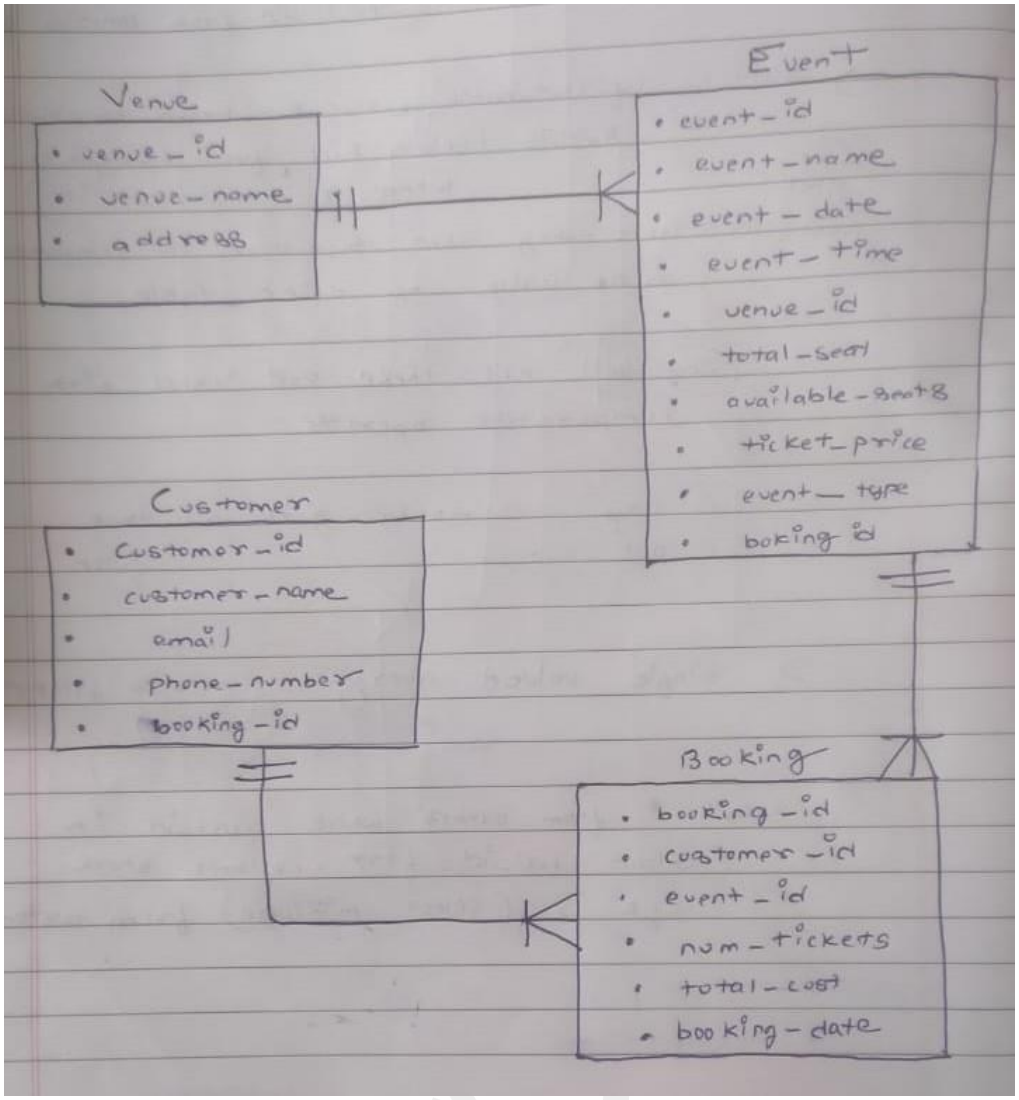
```
    total_cost DECIMAL,
    booking_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (event_id) REFERENCES Event(event_id)
);
```

Command Prompt - mysql -u root -p

```
ERROR 1824 (HY000): Failed to open the referenced table 'customer'
mysql> -- Create Event Table
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,
    ->     event_type ENUM('Movie', 'Sports', 'Concert'),
    ->     booking_id INT
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> -- Create Customer Table
mysql> CREATE TABLE Customer (
    ->     customer_id INT PRIMARY KEY,
    ->     customer_name VARCHAR(255),
    ->     email VARCHAR(255),
    ->     phone_number VARCHAR(20),
    ->     booking_id INT
    -> );
Query OK, 0 rows affected (0.01 sec)

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

mysql>
mysql> -- Add Foreign Key Constraints
mysql> ALTER TABLE Event ADD FOREIGN KEY (venue_id) REFERENCES Venu(venue_id);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Event ADD FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

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

Ans.



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

Ans. The primary key constraints are already included in the table definitions. Foreign key constraints are also included in the table definitions, ensuring that each foreign key refers to the primary key of the corresponding table.

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

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

Ans.

INSERT INTO Venu (venue_id, venue_name, address) VALUES

(1, 'Venue1', 'Address1'),

(2, 'Venue2', 'Address2'),

(3, 'Venue3', 'Address3'),

(4, 'Venue4', 'Address4'),

(5, 'Venue5', 'Address5'),

(6, 'Venue6', 'Address6'),

(7, 'Venue7', 'Address7'),

(8, 'Venue8', 'Address8'),

(9, 'Venue9', 'Address9'),

(10, 'Venue10', 'Address10');

Command Prompt - mysql -u root -p

```
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> INSERT INTO Venu (venue_id, venue_name, address) VALUES
    -> (1, 'Venue1', 'Address1'),
    -> (2, 'Venue2', 'Address2'),
    -> (3, 'Venue3', 'Address3'),
    -> (4, 'Venue4', 'Address4'),
    -> (5, 'Venue5', 'Address5'),
    -> (6, 'Venue6', 'Address6'),
    -> (7, 'Venue7', 'Address7'),
    -> (8, 'Venue8', 'Address8'),
    -> (9, 'Venue9', 'Address9'),
    -> (10, 'Venue10', 'Address10');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
```

-- Insert 10 sample rows into Event table

INSERT INTO Event (event_id, event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type, booking_id) VALUES

(1, 'Event1', '2023-12-15', '18:00:00', 1, 200, 200, 20.00, 'Movie', 1),

(2, 'Event2', '2023-12-16', '19:30:00', 2, 150, 150, 25.00, 'Sports', 2),

(3, 'Event3', '2023-12-17', '20:00:00', 3, 300, 300, 30.00, 'Concert', 3),

(4, 'Event4', '2023-12-18', '17:00:00', 4, 180, 180, 18.00, 'Movie', 4),

(5, 'Event5', '2023-12-19', '21:00:00', 5, 250, 250, 22.50, 'Concert', 5),

(6, 'Event6', '2023-12-20', '19:30:00', 6, 120, 120, 15.00, 'Sports', 6),

(7, 'Event7', '2023-12-21', '20:30:00', 7, 200, 200, 20.00, 'Concert', 7),

(8, 'Event8', '2023-12-22', '18:45:00', 8, 300, 300, 25.00, 'Movie', 8),

(9, 'Event9', '2023-12-23', '19:00:00', 9, 150, 150, 18.00, 'Sports', 9),

(10, 'Event10', '2023-12-24', '22:00:00', 10, 180, 180, 20.00, 'Concert', 10);

Command Prompt - mysql -u root -p

```
    -> (1, 'Event1', '2023-12-15', '18:00:00', 1, 200, 200, 20.00, 'Movie', 1),
    -> (2, 'Event2', '2023-12-16', '19:30:00', 2, 150, 150, 25.00, 'Sports', 2),
    -> (3, 'Event3', '2023-12-17', '20:00:00', 3, 300, 300, 30.00, 'Concert', 3),
    -> (4, 'Event4', '2023-12-18', '17:00:00', 4, 180, 180, 18.00, 'Movie', 4),
    -> (5, 'Event5', '2023-12-19', '21:00:00', 5, 250, 250, 22.50, 'Concert', 5),
    -> (6, 'Event6', '2023-12-20', '19:30:00', 6, 120, 120, 15.00, 'Sports', 6),
    -> (7, 'Event7', '2023-12-21', '20:30:00', 7, 200, 200, 20.00, 'Concert', 7),
    -> (8, 'Event8', '2023-12-22', '18:45:00', 8, 300, 300, 25.00, 'Movie', 8),
    -> (9, 'Event9', '2023-12-23', '19:00:00', 9, 150, 150, 18.00, 'Sports', 9),
    -> (10, 'Event10', '2023-12-24', '22:00:00', 10, 180, 180, 20.00, 'Concert', 10);
Query OK, 10 rows affected, 1 warning (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 1

mysql>
```

-- Insert 10 sample rows into Customer table

INSERT INTO Customer (customer_id, customer_name, email, phone_number, booking_id) VALUES

(1, 'Customer1', 'customer1@email.com', '123-456-7890', 1),

(2, 'Customer2', 'customer2@email.com', '987-654-3210', 2),

(3, 'Customer3', 'customer3@email.com', '111-222-3333', 3),

(4, 'Customer4', 'customer4@email.com', '444-555-6666', 4),

(5, 'Customer5', 'customer5@email.com', '555-666-7777', 5),

(6, 'Customer6', 'customer6@email.com', '666-777-8888', 6),

(7, 'Customer7', 'customer7@email.com', '777-888-9999', 7),

(8, 'Customer8', 'customer8@email.com', '888-999-0000', 8),

(9, 'Customer9', 'customer9@email.com', '999-000-1111', 9),

(10, 'Customer10', 'customer10@email.com', '000-111-2222', 10);

Command Prompt - mysql -u root -p

```
mysql> -- Insert 10 sample rows into Customer table
mysql> INSERT INTO Customer (customer_id, customer_name, email, phone_number, booking_id) VALUES
    -> (1, 'Customer1', 'customer1@email.com', '123-456-7890', 1),
    -> (2, 'Customer2', 'customer2@email.com', '987-654-3210', 2),
    -> (3, 'Customer3', 'customer3@email.com', '111-222-3333', 3),
    -> (4, 'Customer4', 'customer4@email.com', '444-555-6666', 4),
    -> (5, 'Customer5', 'customer5@email.com', '555-666-7777', 5),
    -> (6, 'Customer6', 'customer6@email.com', '666-777-8888', 6),
    -> (7, 'Customer7', 'customer7@email.com', '777-888-9999', 7),
    -> (8, 'Customer8', 'customer8@email.com', '888-999-0000', 8),
    -> (9, 'Customer9', 'customer9@email.com', '999-000-1111', 9),
    -> (10, 'Customer10', 'customer10@email.com', '000-111-2222', 10);
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
```

-- Insert 10 sample rows into Booking table

INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date) VALUES

(1, 1, 1, 3, 60.00, '2023-12-14'),

(2, 2, 2, 2, 50.00, '2023-12-15'),

(3, 3, 3, 5, 150.00, '2023-12-16'),

(4, 4, 4, 2, 36.00, '2023-12-17'),

(5, 5, 5, 4, 90.00, '2023-12-18'),

(6, 6, 6, 1, 15.00, '2023-12-19'),

(7, 7, 7, 3, 60.00, '2023-12-20'),

(8, 8, 8, 5, 125.00, '2023-12-21'),

(9, 9, 9, 2, 36.00, '2023-12-22'),

(10, 10, 10, 3, 60.00, '2023-12-23');

Command Prompt - mysql -u root -p

```
mysql> INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date) VALUES
    -> (1, 1, 1, 3, 60.00, '2023-12-14'),
    -> (2, 2, 2, 2, 50.00, '2023-12-15'),
    -> (3, 3, 3, 5, 150.00, '2023-12-16'),
    -> (4, 4, 4, 2, 36.00, '2023-12-17'),
    -> (5, 5, 5, 4, 90.00, '2023-12-18'),
    -> (6, 6, 6, 1, 15.00, '2023-12-19'),
    -> (7, 7, 7, 3, 60.00, '2023-12-20'),
    -> (8, 8, 8, 5, 125.00, '2023-12-21'),
    -> (9, 9, 9, 2, 36.00, '2023-12-22'),
    -> (10, 10, 10, 3, 60.00, '2023-12-23');
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
```

2. Write a SQL query to list all Events.

Ans. SELECT *

FROM Event;

Command Prompt - mysql -u root -p

```
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 | Event1     | 2023-12-15 | 18:00:00   |        1 |         200 |            200 |           20 | Movie      |          1 |
|        2 | Event2     | 2023-12-16 | 19:30:00   |        2 |         150 |            150 |           25 | Sports     |          2 |
|        3 | Event3     | 2023-12-17 | 20:00:00   |        3 |         300 |            300 |           30 | Concert    |          3 |
|        4 | Event4     | 2023-12-18 | 17:00:00   |        4 |         180 |            180 |           18 | Movie      |          4 |
|        5 | Event5     | 2023-12-19 | 21:00:00   |        5 |         250 |            250 |           23 | Concert    |          5 |
|        6 | Event6     | 2023-12-20 | 19:30:00   |        6 |         120 |            120 |           15 | Sports     |          6 |
|        7 | Event7     | 2023-12-21 | 20:30:00   |        7 |         200 |            200 |           20 | Concert    |          7 |
|        8 | Event8     | 2023-12-22 | 18:45:00   |        8 |         300 |            300 |           25 | Movie      |          8 |
|        9 | Event9     | 2023-12-23 | 19:00:00   |        9 |         150 |            150 |           18 | Sports     |          9 |
|       10 | Event10    | 2023-12-24 | 22:00:00   |       10 |         180 |            180 |           20 | Concert    |         10 |
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
10 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT *

FROM Event

WHERE available_seats > 0;

Command Prompt - mysql -u root -p

```
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
10 rows in set (0.00 sec)

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 | Event1     | 2023-12-15 | 18:00:00   |        1 |         200 |            200 |           20 | Movie      |          1 |
|        2 | Event2     | 2023-12-16 | 19:30:00   |        2 |         150 |            150 |           25 | Sports     |          2 |
|        3 | Event3     | 2023-12-17 | 20:00:00   |        3 |         300 |            300 |           30 | Concert    |          3 |
|        4 | Event4     | 2023-12-18 | 17:00:00   |        4 |         180 |            180 |           18 | Movie      |          4 |
|        5 | Event5     | 2023-12-19 | 21:00:00   |        5 |         250 |            250 |           23 | Concert    |          5 |
|        6 | Event6     | 2023-12-20 | 19:30:00   |        6 |         120 |            120 |           15 | Sports     |          6 |
|        7 | Event7     | 2023-12-21 | 20:30:00   |        7 |         200 |            200 |           20 | Concert    |          7 |
|        8 | Event8     | 2023-12-22 | 18:45:00   |        8 |         300 |            300 |           25 | Movie      |          8 |
|        9 | Event9     | 2023-12-23 | 19:00:00   |        9 |         150 |            150 |           18 | Sports     |          9 |
|       10 | Event10    | 2023-12-24 | 22:00:00   |       10 |         180 |            180 |           20 | Concert    |         10 |
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
10 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT *

FROM Event

WHERE event_name LIKE '%cup%';

Command Prompt - mysql -u root -p

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

mysql> _
```

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

Ans. SELECT *

FROM Event

WHERE ticket_price BETWEEN 1000 AND 2500;

Command Prompt - mysql -u root -p

```
mysql> SELECT *
    -> FROM Event
    -> WHERE ticket_price BETWEEN 1000 AND 2500;
Empty set (0.00 sec)

mysql> _
```

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

Ans. SELECT *

FROM Event

WHERE event_date BETWEEN '2023-12-15' AND '2023-12-22';

Command Prompt - mysql -u root -p

```
Empty set (0.00 sec)

mysql> SELECT *
    -> FROM Event
    -> WHERE event_date BETWEEN '2023-12-15' AND '2023-12-22';
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|        1 | Event1     | 2023-12-15 | 18:00:00   |        1 |         200 |             200 |           20 | Movie      |          1 |
|        2 | Event2     | 2023-12-16 | 19:30:00   |        2 |         150 |             150 |           25 | Sports     |          2 |
|        3 | Event3     | 2023-12-17 | 20:00:00   |        3 |         300 |             300 |           30 | Concert    |          3 |
|        4 | Event4     | 2023-12-18 | 17:00:00   |        4 |         180 |             180 |           18 | Movie      |          4 |
|        5 | Event5     | 2023-12-19 | 21:00:00   |        5 |         250 |             250 |           23 | Concert    |          5 |
|        6 | Event6     | 2023-12-20 | 19:30:00   |        6 |         120 |             120 |           15 | Sports     |          6 |
|        7 | Event7     | 2023-12-21 | 20:30:00   |        7 |         200 |             200 |           20 | Concert    |          7 |
|        8 | Event8     | 2023-12-22 | 18:45:00   |        8 |         300 |             300 |           25 | Movie      |          8 |
+----------+------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
8 rows in set (0.00 sec)
```

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

Ans. SELECT *

FROM Event

WHERE available_seats > 0

AND event_type = 'Concert';

Command Prompt - mysql -u root -p

```
mysql> SELECT *
    -> FROM Event
    -> WHERE available_seats > 0
    ->   AND event_type = 'Concert';
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
| event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
|        3 | Event3     | 2023-12-17 | 20:00:00   |        3 |         300 |            300 |           30 | Concert    |          3 |
|        5 | Event5     | 2023-12-19 | 21:00:00   |        5 |         250 |            250 |           23 | Concert    |          5 |
|        7 | Event7     | 2023-12-21 | 20:30:00   |        7 |         200 |            200 |           20 | Concert    |          7 |
|       10 | Event10    | 2023-12-24 | 22:00:00   |       10 |         180 |            180 |           20 | Concert    |         10 |
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
4 rows in set (0.00 sec)
```

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

Ans. SELECT *

FROM Customer

ORDER BY customer_id

LIMIT 5 OFFSET 5;

Command Prompt - mysql -u root -p

```
mysql> SELECT *
    -> FROM Customer
    -> ORDER BY customer_id
    -> LIMIT 5 OFFSET 5;
+-------------+---------------+----------------------+--------------+------------+
| customer_id | customer_name | email                | phone_number | booking_id |
+-------------+---------------+----------------------+--------------+------------+
|           6 | Customer6     | customer6@email.com  | 666-777-8888 |          6 |
|           7 | Customer7     | customer7@email.com  | 777-888-9999 |          7 |
|           8 | Customer8     | customer8@email.com  | 888-999-0000 |          8 |
|           9 | Customer9     | customer9@email.com  | 999-000-1111 |          9 |
|          10 | Customer10    | customer10@email.com | 000-111-2222 |         10 |
+-------------+---------------+----------------------+--------------+------------+
5 rows in set (0.01 sec)

mysql>
```

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

Ans. SELECT Booking.*, Event.event_name, Event.event_date, Event.event_time

FROM Booking

JOIN Event ON Booking.event_id = Event.event_id

WHERE Booking.num_tickets > 4;

Command Prompt - mysql -u root -p

```
5 rows in set (0.01 sec)

mysql> SELECT Booking.*, Event.event_name, Event.event_date, Event.event_time
    -> FROM Booking
    -> JOIN Event ON Booking.event_id = Event.event_id
    -> WHERE Booking.num_tickets > 4;
+------------+------------+----------+------------+------------+--------------+------------+------------+------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date | event_name | event_date | event_time |
+------------+------------+----------+------------+------------+--------------+------------+------------+------------+
|          3 |          3 |        3 |          5 |        150 | 2023-12-16   | Event3     | 2023-12-17 | 20:00:00   |
|          8 |          8 |        8 |          5 |        125 | 2023-12-21   | Event8     | 2023-12-22 | 18:45:00   |
+------------+------------+----------+------------+------------+--------------+------------+------------+------------+
2 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT *

FROM Customer

WHERE phone_number LIKE '%000';

Command Prompt - mysql -u root -p

```
2 rows in set (0.00 sec)

mysql> SELECT *
    -> FROM Customer
    -> WHERE phone_number LIKE '%000';
+-------------+---------------+---------------------+--------------+------------+
| customer_id | customer_name | email               | phone_number | booking_id |
+-------------+---------------+---------------------+--------------+------------+
|           8 | Customer8     | customer8@email.com | 888-999-0000 |          8 |
+-------------+---------------+---------------------+--------------+------------+
1 row in set (0.00 sec)
```

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

Ans. SELECT *

FROM Customer

WHERE phone_number LIKE '%000';

Command Prompt - mysql -u root -p

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

mysql>
```

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

Ans. SELECT *

FROM Event

WHERE event_name NOT LIKE 'x%'

AND event_name NOT LIKE 'y%'

AND event_name NOT LIKE 'z%';

Command Prompt - mysql -u root -p

```
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 | Event1     | 2023-12-15 | 18:00:00   |        1 |         200 |            200 |           20 | Movie      |          1 |
|        2 | Event2     | 2023-12-16 | 19:30:00   |        2 |         150 |            150 |           25 | Sports     |          2 |
|        3 | Event3     | 2023-12-17 | 20:00:00   |        3 |         300 |            300 |           30 | Concert    |          3 |
|        4 | Event4     | 2023-12-18 | 17:00:00   |        4 |         180 |            180 |           18 | Movie      |          4 |
|        5 | Event5     | 2023-12-19 | 21:00:00   |        5 |         250 |            250 |           23 | Concert    |          5 |
|        6 | Event6     | 2023-12-20 | 19:30:00   |        6 |         120 |            120 |           15 | Sports     |          6 |
|        7 | Event7     | 2023-12-21 | 20:30:00   |        7 |         200 |            200 |           20 | Concert    |          7 |
|        8 | Event8     | 2023-12-22 | 18:45:00   |        8 |         300 |            300 |           25 | Movie      |          8 |
|        9 | Event9     | 2023-12-23 | 19:00:00   |        9 |         150 |            150 |           18 | Sports     |          9 |
|       10 | Event10    | 2023-12-24 | 22:00:00   |       10 |         180 |            180 |           20 | Concert    |         10 |
+----------+------------+------------+------------+----------+-------------+----------------+--------------+------------+------------+
10 rows in set (0.01 sec)

mysql>
```

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

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

Ans. SELECT

SUM(Booking.total_cost) AS total_revenue

FROM

Booking;

Command Prompt - mysql -u root -p

```
mysql> SELECT
    ->     Event.event_id,
    ->     Event.event_name,
    ->     AVG(Event.ticket_price) AS average_ticket_price
    -> FROM
    ->     Event
    -> GROUP BY
    ->     Event.event_id, Event.event_name;
+----------+------------+----------------------+
| event_id | event_name | average_ticket_price |
+----------+------------+----------------------+
|        1 | Event1     |              20.0000 |
|        2 | Event2     |              25.0000 |
|        3 | Event3     |              30.0000 |
|        4 | Event4     |              18.0000 |
|        5 | Event5     |              23.0000 |
|        6 | Event6     |              15.0000 |
|        7 | Event7     |              20.0000 |
|        8 | Event8     |              25.0000 |
|        9 | Event9     |              18.0000 |
|       10 | Event10    |              20.0000 |
+----------+------------+----------------------+
10 rows in set (0.03 sec)

mysql>
```

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

Ans. SELECT

SUM(Booking.total_cost) AS total_revenue

FROM

Booking

JOIN Event ON Booking.event_id = Event.event_id;

Command Prompt - mysql -u root -p

```
mysql> SELECT
    ->     SUM(Booking.total_cost) AS total_revenue
    -> FROM
    ->     Booking;
+---------------+
| total_revenue |
+---------------+
|           682 |
+---------------+
1 row in set (0.00 sec)

mysql>
```

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

Ans. SELECT

Event.event_id,

Event.event_name,

SUM(Booking.num_tickets) AS total_tickets_sold

FROM

Event

JOIN Booking ON Event.event_id = Booking.event_id

GROUP BY

Event.event_id, Event.event_name

ORDER BY

total_tickets_sold DESC

LIMIT 1;

Command Prompt - mysql -u root -p

```
1 row in set (0.00 sec)

mysql> SELECT
    ->     Event.event_id,
    ->     Event.event_name,
    ->     SUM(Booking.num_tickets) AS total_tickets_sold
    -> FROM
    ->     Event
    -> JOIN Booking ON Event.event_id = Booking.event_id
    -> GROUP BY
    ->     Event.event_id, Event.event_name
    -> ORDER BY
    ->     total_tickets_sold DESC
    -> LIMIT 1;
+----------+------------+--------------------+
| event_id | event_name | total_tickets_sold |
+----------+------------+--------------------+
|        3 | Event3     |                  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.

Ans. SELECT

Event.event_id,

Event.event_name,

SUM(Booking.num_tickets) AS total_tickets_sold

FROM

Event

JOIN Booking ON Event.event_id = Booking.event_id

GROUP BY

Event.event_id, Event.event_name;

Command Prompt - mysql -u root -p

```
mysql> SELECT
    ->     Event.event_id,
    ->     Event.event_name,
    ->     SUM(Booking.num_tickets) AS total_tickets_sold
    -> FROM
    ->     Event
    -> 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 | Event1     |                  3 |
|        2 | Event2     |                  2 |
|        3 | Event3     |                  5 |
|        4 | Event4     |                  2 |
|        5 | Event5     |                  4 |
|        6 | Event6     |                  1 |
|        7 | Event7     |                  3 |
|        8 | Event8     |                  5 |
|        9 | Event9     |                  2 |
|       10 | Event10    |                  3 |
+----------+------------+--------------------+
10 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT

Event.event_id,

Event.event_name

FROM

Event

LEFT JOIN Booking ON Event.event_id = Booking.event_id

WHERE

Booking.event_id IS NULL;

Command Prompt - mysql -u root -p

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

mysql>
```

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

Ans. SELECT

Customer.customer_id,

Customer.customer_name,

SUM(Booking.num_tickets) AS total_tickets_booked

FROM

Customer

JOIN Booking ON Customer.customer_id = Booking.customer_id

GROUP BY

Customer.customer_id, Customer.customer_name

ORDER BY

total_tickets_booked DESC

LIMIT 1;

Command Prompt - mysql -u root -p

```
Empty set (0.00 sec)

mysql> SELECT
    ->     Customer.customer_id,
    ->     Customer.customer_name,
    ->     SUM(Booking.num_tickets) AS total_tickets_booked
    -> FROM
    ->     Customer
    -> JOIN Booking ON Customer.customer_id = Booking.customer_id
    -> GROUP BY
    ->     Customer.customer_id, Customer.customer_name
    -> ORDER BY
    ->     total_tickets_booked DESC
    -> LIMIT 1;
+-------------+---------------+----------------------+
| customer_id | customer_name | total_tickets_booked |
+-------------+---------------+----------------------+
|           3 | Customer3     |                    5 |
+-------------+---------------+----------------------+
1 row in set (0.00 sec)

mysql>
```

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

Ans. SELECT

Event.event_id,

Event.event_name,

EXTRACT(MONTH FROM Booking.booking_date) AS month,

SUM(Booking.num_tickets) AS total_tickets_sold

FROM

Event

JOIN Booking ON Event.event_id = Booking.event_id

GROUP BY

Event.event_id, Event.event_name, month

ORDER BY

Event.event_id, month;

Command Prompt - mysql -u root -p

```
mysql> SELECT
    ->     Event.event_id,
    ->     Event.event_name,
    ->     EXTRACT(MONTH FROM Booking.booking_date) AS month,
    ->     SUM(Booking.num_tickets) AS total_tickets_sold
    -> FROM
    ->     Event
    -> JOIN Booking ON Event.event_id = Booking.event_id
    -> GROUP BY
    ->     Event.event_id, Event.event_name, month
    -> ORDER BY
    ->     Event.event_id, month;
+----------+------------+-------+--------------------+
| event_id | event_name | month | total_tickets_sold |
+----------+------------+-------+--------------------+
|        1 | Event1     |    12 |                  3 |
|        2 | Event2     |    12 |                  2 |
|        3 | Event3     |    12 |                  5 |
|        4 | Event4     |    12 |                  2 |
|        5 | Event5     |    12 |                  4 |
|        6 | Event6     |    12 |                  1 |
|        7 | Event7     |    12 |                  3 |
|        8 | Event8     |    12 |                  5 |
|        9 | Event9     |    12 |                  2 |
|       10 | Event10    |    12 |                  3 |
+----------+------------+-------+--------------------+
10 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT

Event.venue_id,

Venu.venue_name,

AVG(Event.ticket_price) AS average_ticket_price

FROM

Event

JOIN Venu ON Event.venue_id = Venu.venue_id

GROUP BY

Event.venue_id, Venu.venue_name;

Command Prompt - mysql -u root -p

```
10 rows in set (0.00 sec)

mysql> SELECT
    ->     Event.venue_id,
    ->     Venu.venue_name,
    ->     AVG(Event.ticket_price) AS average_ticket_price
    -> FROM
    ->     Event
    -> JOIN Venu ON Event.venue_id = Venu.venue_id
    -> GROUP BY
    ->     Event.venue_id, Venu.venue_name;
+----------+------------+----------------------+
| venue_id | venue_name | average_ticket_price |
+----------+------------+----------------------+
|        1 | Venue1     |              20.0000 |
|        2 | Venue2     |              25.0000 |
|        3 | Venue3     |              30.0000 |
|        4 | Venue4     |              18.0000 |
|        5 | Venue5     |              23.0000 |
|        6 | Venue6     |              15.0000 |
|        7 | Venue7     |              20.0000 |
|        8 | Venue8     |              25.0000 |
|        9 | Venue9     |              18.0000 |
|       10 | Venue10    |              20.0000 |
+----------+------------+----------------------+
10 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT

Event.event_type,

SUM(Booking.num_tickets) AS total_tickets_sold

FROM

Event

JOIN Booking ON Event.event_id = Booking.event_id

GROUP BY

Event.event_type

ORDER BY

Event.event_type;

Command Prompt - mysql -u root -p

```
+----------+------------+---------------------+
10 rows in set (0.00 sec)

mysql> SELECT
    ->     Event.event_type,
    ->     SUM(Booking.num_tickets) AS total_tickets_sold
    -> FROM
    ->     Event
    -> JOIN Booking ON Event.event_id = Booking.event_id
    -> GROUP BY
    ->     Event.event_type
    -> ORDER BY
    ->     Event.event_type;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Movie      |                 10 |
| Sports     |                  5 |
| Concert    |                 15 |
+------------+--------------------+
3 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT

EXTRACT(YEAR FROM Booking.booking_date) AS event_year,

SUM(Booking.total_cost) AS total_revenue

FROM

Booking

GROUP BY

event_year

ORDER BY

event_year;

Command Prompt - mysql -u root -p

```
mysql> SELECT
    ->     EXTRACT(YEAR FROM Booking.booking_date) AS event_year,
    ->     SUM(Booking.total_cost) AS total_revenue
    -> FROM
    ->     Booking
    -> GROUP BY
    ->     event_year
    -> ORDER BY
    ->     event_year;
+------------+---------------+
| event_year | total_revenue |
+------------+---------------+
|       2023 |           682 |
+------------+---------------+
1 row in set (0.00 sec)

mysql>
```

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

Ans. SELECT

Customer.customer_id,

Customer.customer_name,

COUNT(DISTINCT Booking.event_id) AS num_events_booked

FROM

Customer

JOIN Booking ON Customer.customer_id = Booking.customer_id

GROUP BY

Customer.customer_id, Customer.customer_name

HAVING

num_events_booked > 1;

Command Prompt - mysql -u root -p

```
mysql> SELECT
    ->     Customer.customer_id,
    ->     Customer.customer_name,
    ->     COUNT(DISTINCT Booking.event_id) AS num_events_booked
    -> FROM
    ->     Customer
    -> JOIN Booking ON Customer.customer_id = Booking.customer_id
    -> GROUP BY
    ->     Customer.customer_id, Customer.customer_name
    -> HAVING
    ->     num_events_booked > 1;
Empty set (0.03 sec)

mysql>
```

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

Ans. SELECT

Customer.customer_id,

Customer.customer_name,

SUM(Booking.total_cost) AS total_revenue

FROM

Customer

JOIN Booking ON Customer.customer_id = Booking.customer_id

GROUP BY

Customer.customer_id, Customer.customer_name

ORDER BY

total_revenue DESC;

Command Prompt - mysql -u root -p

```
Empty set (0.03 sec)

mysql> SELECT
    ->     Customer.customer_id,
    ->     Customer.customer_name,
    ->     SUM(Booking.total_cost) AS total_revenue
    -> FROM
    ->     Customer
    -> JOIN Booking ON Customer.customer_id = Booking.customer_id
    -> GROUP BY
    ->     Customer.customer_id, Customer.customer_name
    -> ORDER BY
    ->     total_revenue DESC;
+-------------+---------------+---------------+
| customer_id | customer_name | total_revenue |
+-------------+---------------+---------------+
|           3 | Customer3     |           150 |
|           8 | Customer8     |           125 |
|           5 | Customer5     |            90 |
|           1 | Customer1     |            60 |
|           7 | Customer7     |            60 |
|          10 | Customer10    |            60 |
|           2 | Customer2     |            50 |
|           4 | Customer4     |            36 |
|           9 | Customer9     |            36 |
|           6 | Customer6     |            15 |
+-------------+---------------+---------------+
10 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT

Venu.venue_id,

Venu.venue_name,

Event.event_type,

AVG(Event.ticket_price) AS average_ticket_price

FROM

Venu

JOIN Event ON Venu.venue_id = Event.venue_id

GROUP BY

Venu.venue_id, Venu.venue_name, Event.event_type

ORDER BY

Venu.venue_id, Event.event_type;

Command Prompt - mysql -u root -p

```
10 rows in set (0.00 sec)

mysql> SELECT
    ->     Venu.venue_id,
    ->     Venu.venue_name,
    ->     Event.event_type,
    ->     AVG(Event.ticket_price) AS average_ticket_price
    -> FROM
    ->     Venu
    -> JOIN Event ON Venu.venue_id = Event.venue_id
    -> GROUP BY
    ->     Venu.venue_id, Venu.venue_name, Event.event_type
    -> ORDER BY
    ->     Venu.venue_id, Event.event_type;
+----------+------------+------------+----------------------+
| venue_id | venue_name | event_type | average_ticket_price |
+----------+------------+------------+----------------------+
|        1 | Venue1     | Movie      |              20.0000 |
|        2 | Venue2     | Sports     |              25.0000 |
|        3 | Venue3     | Concert    |              30.0000 |
|        4 | Venue4     | Movie      |              18.0000 |
|        5 | Venue5     | Concert    |              23.0000 |
|        6 | Venue6     | Sports     |              15.0000 |
|        7 | Venue7     | Concert    |              20.0000 |
|        8 | Venue8     | Movie      |              25.0000 |
|        9 | Venue9     | Sports     |              18.0000 |
|       10 | Venue10    | Concert    |              20.0000 |
+----------+------------+------------+----------------------+
10 rows in set (0.00 sec)

mysql>
```

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

Ans.

SELECT

Customer.customer_id,

Customer.customer_name,

SUM(Booking.num_tickets) AS total_tickets_purchased

FROM

Customer

JOIN Booking ON Customer.customer_id = Booking.customer_id

WHERE

Booking.booking_date >= CURRENT_DATE - INTERVAL 30 DAY

GROUP BY

Customer.customer_id, Customer.customer_name

ORDER BY

total_tickets_purchased DESC;

Command Prompt - mysql -u root -p

```
+----------+------------+------------+---------------------+
10 rows in set (0.00 sec)

mysql> SELECT
    ->     Customer.customer_id,
    ->     Customer.customer_name,
    ->     SUM(Booking.num_tickets) AS total_tickets_purchased
    -> FROM
    ->     Customer
    -> JOIN Booking ON Customer.customer_id = Booking.customer_id
    -> WHERE
    ->     Booking.booking_date >= CURRENT_DATE - INTERVAL 30 DAY
    -> GROUP BY
    ->     Customer.customer_id, Customer.customer_name
    -> ORDER BY
    ->     total_tickets_purchased DESC;
+-------------+---------------+------------------------+
| customer_id | customer_name | total_tickets_purchased |
+-------------+---------------+------------------------+
|           3 | Customer3     |                      5 |
|           8 | Customer8     |                      5 |
|           5 | Customer5     |                      4 |
|           1 | Customer1     |                      3 |
|           7 | Customer7     |                      3 |
|          10 | Customer10    |                      3 |
|           2 | Customer2     |                      2 |
|           4 | Customer4     |                      2 |
|           9 | Customer9     |                      2 |
|           6 | Customer6     |                      1 |
+-------------+---------------+------------------------+
10 rows in set (0.00 sec)

mysql>
```

Tasks 4: Subquery and its types

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

Ans. 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;

Command Prompt - mysql -u root -p

```
10 rows in set (0.00 sec)

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 | Venue1     |              20.0000 |
|        2 | Venue2     |              25.0000 |
|        3 | Venue3     |              30.0000 |
|        4 | Venue4     |              18.0000 |
|        5 | Venue5     |              23.0000 |
|        6 | Venue6     |              15.0000 |
|        7 | Venue7     |              20.0000 |
|        8 | Venue8     |              25.0000 |
|        9 | Venue9     |              18.0000 |
|       10 | Venue10    |              20.0000 |
+----------+------------+---------------------+
10 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT

E.event_id,

E.event_name,

E.event_date,

E.event_time,

E.total_seats,

E.available_seats,

E.ticket_price,

(E.total_seats - E.available_seats) AS tickets_sold

FROM

Event E

WHERE

(E.total_seats - E.available_seats) > (0.5 * E.total_seats);

Command Prompt - mysql -u root -p

```
mysql> SELECT
    ->     E.event_id,
    ->     E.event_name,
    ->     E.event_date,
    ->     E.event_time,
    ->     E.total_seats,
    ->     E.available_seats,
    ->     E.ticket_price,
    ->     (E.total_seats - E.available_seats) AS tickets_sold
    -> FROM
    ->     Event E
    -> WHERE
    ->     (E.total_seats - E.available_seats) > (0.5 * E.total_seats);
Empty set (0.03 sec)

mysql>
```

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

Ans. SELECT

E.event_id,

E.event_name,

E.event_date,

E.event_time,

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, E.event_date, E.event_time;

Command Prompt - mysql -u root -p

```
Empty set (0.03 sec)

mysql> SELECT
    ->     E.event_id,
    ->     E.event_name,
    ->     E.event_date,
    ->     E.event_time,
    ->     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, E.event_date, E.event_time;
+----------+------------+------------+------------+--------------------+
| event_id | event_name | event_date | event_time | total_tickets_sold |
+----------+------------+------------+------------+--------------------+
|        1 | Event1     | 2023-12-15 | 18:00:00   |                  3 |
|        2 | Event2     | 2023-12-16 | 19:30:00   |                  2 |
|        3 | Event3     | 2023-12-17 | 20:00:00   |                  5 |
|        4 | Event4     | 2023-12-18 | 17:00:00   |                  2 |
|        5 | Event5     | 2023-12-19 | 21:00:00   |                  4 |
|        6 | Event6     | 2023-12-20 | 19:30:00   |                  1 |
|        7 | Event7     | 2023-12-21 | 20:30:00   |                  3 |
|        8 | Event8     | 2023-12-22 | 18:45:00   |                  5 |
|        9 | Event9     | 2023-12-23 | 19:00:00   |                  2 |
|       10 | Event10    | 2023-12-24 | 22:00:00   |                  3 |
+----------+------------+------------+------------+--------------------+
10 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT

```
  C.customer_id,
  C.customer_name
FROM
  Customer C
WHERE
  NOT EXISTS (
    SELECT 1
    FROM
      Booking B
    WHERE
      B.customer_id = C.customer_id
  );
```

Command Prompt - mysql -u root -p

```
+----------+------------+------------+------------
10 rows in set (0.00 sec)

mysql> SELECT
    ->     C.customer_id,
    ->     C.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)

mysql>
```

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

Ans.

```
SELECT
  E.event_id,
  E.event_name,
  E.event_date,
  E.event_time
FROM
  Event E
WHERE
  E.event_id NOT IN (
    SELECT DISTINCT
      B.event_id
    FROM
      Booking B
  );
```

Command Prompt - mysql -u root -p

```
mysql> SELECT
    ->     E.event_id,
    ->     E.event_name,
    ->     E.event_date,
    ->     E.event_time
    -> FROM
    ->     Event E
    -> WHERE
    ->     E.event_id NOT IN (
    ->         SELECT DISTINCT
    ->             B.event_id
    ->         FROM
    ->             Booking B
    ->     );
Empty set (0.00 sec)

mysql>
```

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

Ans. 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;

Command Prompt - mysql -u root -p

```
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 |
+------------+--------------------+
| Movie      |                 10 |
| Sports     |                  5 |
| Concert    |                 15 |
+------------+--------------------+
3 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT

```
  event_id,
  event_name,
  event_date,
  event_time,
  ticket_price
FROM
  Event
WHERE
  ticket_price > (
    SELECT AVG(ticket_price)
    FROM Event
  );
```

Command Prompt - mysql -u root -p

```
mysql> SELECT
    ->     event_id,
    ->     event_name,
    ->     event_date,
    ->     event_time,
    ->     ticket_price
    -> FROM
    ->     Event
    -> WHERE
    ->     ticket_price > (
    ->         SELECT AVG(ticket_price)
    ->         FROM Event
    ->     );
+----------+------------+------------+------------+--------------+
| event_id | event_name | event_date | event_time | ticket_price |
+----------+------------+------------+------------+--------------+
|        2 | Event2     | 2023-12-16 | 19:30:00   |           25 |
|        3 | Event3     | 2023-12-17 | 20:00:00   |           30 |
|        5 | Event5     | 2023-12-19 | 21:00:00   |           23 |
|        8 | Event8     | 2023-12-22 | 18:45:00   |           25 |
+----------+------------+------------+------------+--------------+
4 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT

C.customer_id,

C.customer_name,

SUM(B.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;

Command Prompt - mysql -u root -p

```
10 rows in set (0.00 sec)

mysql> SELECT
    ->     customer_id,
    ->     customer_name
    -> FROM
    ->     Customer
    -> WHERE
    ->     customer_id IN (
    ->         SELECT DISTINCT
    ->             B.customer_id
    ->         FROM
    ->             Booking B
    ->             JOIN Event E ON B.event_id = E.event_id
    ->         WHERE
    ->             E.venue_id = 1
    ->     );
+-------------+---------------+
| customer_id | customer_name |
+-------------+---------------+
|           1 | Customer1     |
+-------------+---------------+
1 row in set (0.00 sec)

mysql>
```

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

Ans. SELECT

customer_id,

customer_name

FROM

Customer

WHERE

customer_id IN (

SELECT DISTINCT

```
        B.customer_id
    FROM
        Booking B
        JOIN Event E ON B.event_id = E.event_id
    WHERE
        E.venue_id = 1
);
```

Command Prompt - mysql -u root -p

```
+-------------+---------------+
1 row in set (0.00 sec)

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 |
+------------+--------------------+
| Movie      |                 10 |
| Sports     |                  5 |
| Concert    |                 15 |
+------------+--------------------+
3 rows in set (0.00 sec)

mysql>
```

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

Ans. 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;
```

Command Prompt - mysql -u root -p

```
+-------------+---------------+
1 row in set (0.00 sec)

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 |
+------------+--------------------+
| Movie      |                 10 |
| Sports     |                  5 |
| Concert    |                 15 |
+------------+--------------------+
3 rows in set (0.00 sec)

mysql>
```

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

Ans. SELECT

C.customer_id,

C.customer_name,

DATE_FORMAT(B.booking_date, '%Y-%m') AS booking_month

FROM

Customer C

JOIN

Booking B ON C.booking_id = B.booking_id

GROUP BY

C.customer_id, C.customer_name, booking_month;

Command Prompt - mysql -u root -p

```
mysql> SELECT
    ->     C.customer_id,
    ->     C.customer_name,
    ->     DATE_FORMAT(B.booking_date, '%Y-%m') AS booking_month
    -> FROM
    ->     Customer C
    -> JOIN
    ->     Booking B ON C.booking_id = B.booking_id
    -> GROUP BY
    ->     C.customer_id, C.customer_name, booking_month;
+-------------+---------------+---------------+
| customer_id | customer_name | booking_month |
+-------------+---------------+---------------+
|           1 | Customer1     | 2023-12       |
|           2 | Customer2     | 2023-12       |
|           3 | Customer3     | 2023-12       |
|           4 | Customer4     | 2023-12       |
|           5 | Customer5     | 2023-12       |
|           6 | Customer6     | 2023-12       |
|           7 | Customer7     | 2023-12       |
|           8 | Customer8     | 2023-12       |
|           9 | Customer9     | 2023-12       |
|          10 | Customer10    | 2023-12       |
+-------------+---------------+---------------+
10 rows in set (0.01 sec)

mysql>
```

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

Ans. 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;

Command Prompt - mysql -u root -p

```
10 rows in set (0.01 sec)

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 | Venue1     |              20.0000 |
|        2 | Venue2     |              25.0000 |
|        3 | Venue3     |              30.0000 |
|        4 | Venue4     |              18.0000 |
|        5 | Venue5     |              23.0000 |
|        6 | Venue6     |              15.0000 |
|        7 | Venue7     |              20.0000 |
|        8 | Venue8     |              25.0000 |
|        9 | Venue9     |              18.0000 |
|       10 | Venue10    |              20.0000 |
+----------+------------+---------------------+
10 rows in set (0.00 sec)

mysql>
```