**Tasks 1: Database Design:**

1. Create the database named "TicketBookingSystem"

   CREATE DATABASE TicketBookingSystem;

   USE TicketBookingSystem;

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

   ✓ Venu

   ```
   CREATE TABLE Venue (
      venue_id INT AUTO_INCREMENT PRIMARY KEY,
      venue_name VARCHAR(100) NOT NULL,
      address VARCHAR(255) NOT NULL
   );
   ```

| * venue_id int | * venue_name varchar(100) | * address varchar(255) |
|---|---|---|

   ✓ Event

   ```
   CREATE TABLE Event (
      event_id INT AUTO_INCREMENT PRIMARY KEY,
      event_name VARCHAR(100) NOT NULL,
      event_date DATE NOT NULL,
      event_time TIME NOT NULL,
      venue_id INT NOT NULL,
      total_seats INT NOT NULL,
      available_seats INT NOT NULL,
      ticket_price DECIMAL(10, 2) NOT NULL,
      event_type ENUM('Movie', 'Sports', 'Concert') NOT NULL,

      FOREIGN KEY (venue_id) REFERENCES Venue(venue_id)
   );
   ```

| * event_id int | * event_name varchar(100) | * event_date date | * event_time time | * venue int | * total_seats int | * available_seats int | * ticket_price decimal(10,2) | * event_type enum('Movie','Sports', |
|---|---|---|---|---|---|---|---|---|

   ✓ Customers

ygokul720@gmail.com

```
CREATE TABLE Customer (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    phone_number VARCHAR(15) NOT NULL
);
```

| customer_id int | customer_name varchar(100) | email varchar(100) | phone_numbe varchar(15) |
|---|---|---|---|

✓ Booking

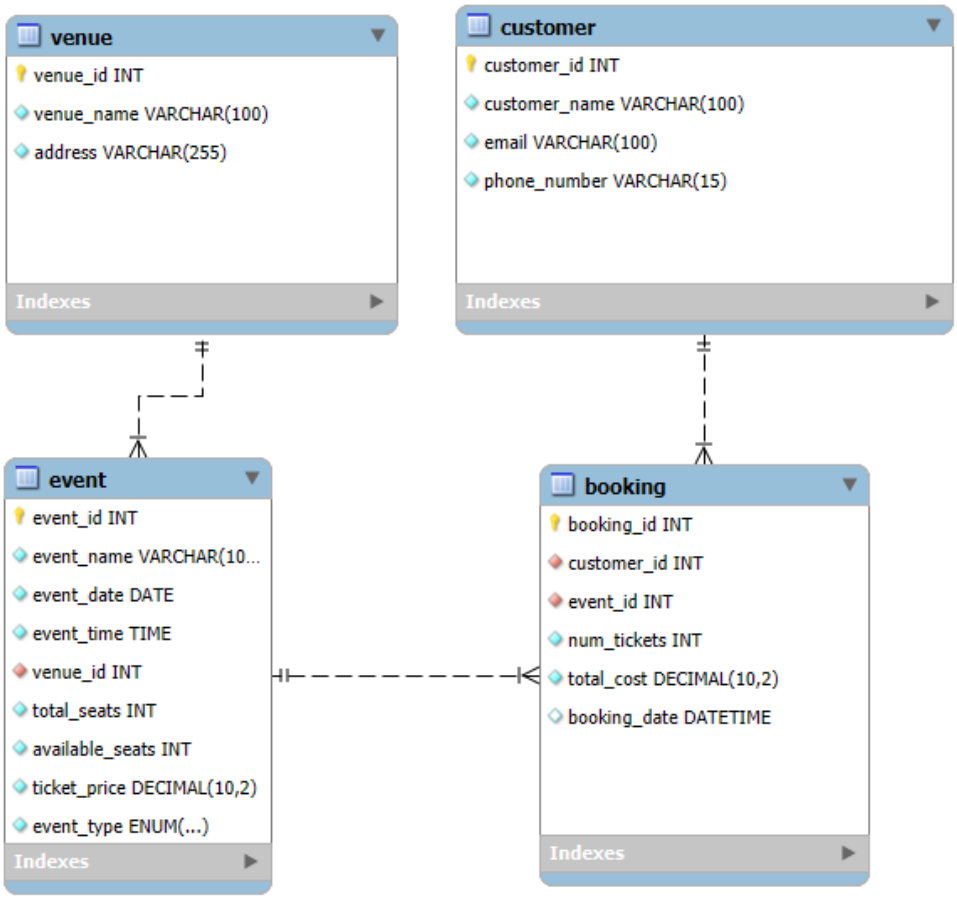
```
CREATE TABLE Booking (
    booking_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    event_id INT NOT NULL,
    num_tickets INT NOT NULL,
    total_cost DECIMAL(10, 2) NOT NULL,
    booking_date DATETIME DEFAULT CURRENT_TIMESTAMP,

    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (event_id) REFERENCES Event(event_id)
);
```

| booking_id int | customer_i int | event int | num_tickets int | total_cost decimal(10,2) | booking_date datetime |
|---|---|---|---|---|---|

ygokul720@gmail.com

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




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

   All primary keys and foreign keys were properly defined in the SQL provided

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

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

   **-- Insert into Venue --**

   INSERT INTO Venue (venue_id, venue_name, address) VALUES

   (1, 'Arena Hall', 'Chennai'),

   (2, 'Stadium Dome', 'Bangalore'),

   (3, 'Open Sky Theater', 'Hyderabad'),

   (4, 'Classic Cineplex', 'Mumbai'),

   (5, 'Sports Arena', 'Delhi'),

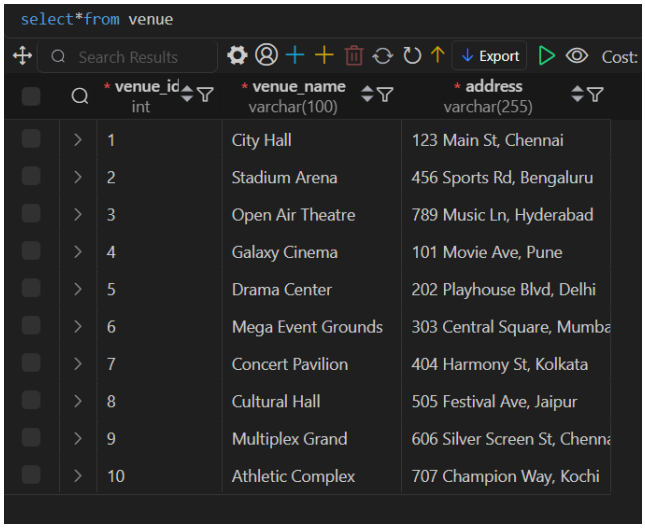
   (6, 'City Stage', 'Kolkata'),

   (7, 'Royal Opera House', 'Chennai'),

(8, 'Metro Square', 'Pune'),

(9, 'Moonlight Grounds', 'Ahmedabad'),

(10, 'Sunset Point', 'Jaipur');



```
select*from venue
```

| venue_id int | venue_name varchar(100) | address varchar(255) |
|---|---|---|
| 1 | City Hall | 123 Main St, Chennai |
| 2 | Stadium Arena | 456 Sports Rd, Bengaluru |
| 3 | Open Air Theatre | 789 Music Ln, Hyderabad |
| 4 | Galaxy Cinema | 101 Movie Ave, Pune |
| 5 | Drama Center | 202 Playhouse Blvd, Delhi |
| 6 | Mega Event Grounds | 303 Central Square, Mumba |
| 7 | Concert Pavilion | 404 Harmony St, Kolkata |
| 8 | Cultural Hall | 505 Festival Ave, Jaipur |
| 9 | Multiplex Grand | 606 Silver Screen St, Chenna |
| 10 | Athletic Complex | 707 Champion Way, Kochi |

**-- Insert into Event --**

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

VALUES

(1, 'World Cup Finals', '2025-04-15', '18:00:00', 2, 25000, 10000, 2000.00, 'Sports'),

(2, 'Sunburn Concert', '2025-05-20', '19:00:00', 1, 20000, 15000, 1800.00, 'Concert'),

(3, 'Drama Nights', '2025-06-10', '17:30:00', 3, 1500, 200, 750.00, 'Play'),

(4, 'Indie Rock Live', '2025-05-25', '20:00:00', 4, 1800, 0, 1200.00, 'Concert'),

(5, 'Comedy Show Cup', '2025-04-30', '16:00:00', 5, 1000, 300, 900.00, 'Play'),

(6, 'Romantic Movie Gala', '2025-05-01', '15:00:00', 6, 120, 15, 500.00, 'Movie'),

(7, 'Jazz Concert Cup', '2025-07-15', '20:30:00', 7, 15000, 7500, 2200.00, 'Concert'),

(8, 'Bollywood Blockbuster', '2025-04-18', '14:00:00', 8, 500, 100, 600.00, 'Movie'),

(9, 'Classical Night', '2025-04-20', '19:30:00', 9, 2500, 1500, 1300.00, 'Concert'),

(10, 'Science Fest', '2025-06-05', '10:00:00', 10, 3000, 2700, 100.00, 'Play');

ygokul720@gmail.com

| event_id int | event_name varchar(100) | event_date date | event_time time | venue int | total_seats int | available_seats int | ticket_price decimal(10,2) | event_type enum('Movie','Sports', |
|---|---|---|---|---|---|---|---|---|
| 1 | IPL Cup Final | 2025-04-25 | 18:30:00 | 2 | 30000 | 10000 | 2500.00 | Sports |
| 2 | Rocking Beats Live | 2025-04-10 | 20:00:00 | 3 | 8000 | 4500 | 1800.00 | Concert |
| 3 | Blockbuster Premiere | 2025-03-22 | 17:00:00 | 4 | 200 | 50 | 350.00 | Movie |
| 4 | Shakespeare Drama | 2025-04-01 | 19:30:00 | 5 | 500 | 200 | 500.00 | Movie |
| 5 | EDM Night Festival | 2025-04-12 | 21:00:00 | 6 | 10000 | 7000 | 2200.00 | Concert |
| 6 | Comedy Show | 2025-04-08 | 18:00:00 | 8 | 300 | 100 | 700.00 | Movie |
| 7 | Classical Music Evening | 2025-04-15 | 19:00:00 | 7 | 1500 | 750 | 1500.00 | Concert |
| 8 | Championship League | 2025-04-30 | 16:00:00 | 10 | 25000 | 5000 | 2000.00 | Sports |
| 9 | Action Movie Release | 2025-03-29 | 15:00:00 | 9 | 300 | 150 | 300.00 | Movie |
| 10 | Folk Music Night | 2025-04-18 | 18:30:00 | 1 | 600 | 250 | 900.00 | Concert |

**-- Insert into Customer --**

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

(1, 'Arjun Kumar', 'arjun@mail.com', '9876543000'),

(2, 'Divya Sharma', 'divya@mail.com', '9998877660'),

(3, 'Rohit Mehra', 'rohit@mail.com', '8888877661'),

(4, 'Anjali Gupta', 'anjali@mail.com', '7778877662'),

(5, 'Vikram Patel', 'vikram@mail.com', '6666677663'),

(6, 'Sahana Rao', 'sahana@mail.com', '9999900000'),

(7, 'Pooja Iyer', 'pooja@mail.com', '9123456000'),

(8, 'Tarun Singh', 'tarun@mail.com', '8899776600'),

(9, 'Neha Jain', 'neha@mail.com', '8000000000'),

(10, 'Gokul BT', 'gokul@mail.com', '7010000000');

| customer_id int | customer_name varchar(100) | email varchar(100) | phone_number varchar(15) |
|---|---|---|---|
| 1 | Arjun Rao | arjun@example.com | 9876543000 |
| 2 | Meera Iyer | meera@example.com | 9123456700 |
| 3 | Ravi Kumar | ravi@example.com | 9988776600 |
| 4 | Divya Sharma | divya@example.com | 9011223344 |
| 5 | Krishna Menon | krishna@example.com | 9345678900 |
| 6 | Sneha Kapoor | sneha@example.com | 9456123000 |
| 7 | Vikram Joshi | vikram@example.com | 9500654321 |
| 8 | Pooja Nair | pooja@example.com | 9234567890 |
| 9 | Aditya Verma | aditya@example.com | 9876500000 |
| 10 | Lakshmi Rao | lakshmi@example.com | 9998800000 |

**-- Insert into Booking --**

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

(1, 1, 1, 3, 6000.00, '2025-03-01'),

(2, 2, 2, 5, 9000.00, '2025-03-01'),

(3, 3, 5, 2, 1800.00, '2025-03-02'),

(4, 4, 4, 1, 1200.00, '2025-03-03'),

(5, 5, 6, 6, 3000.00, '2025-03-04'),

(6, 6, 7, 2, 4400.00, '2025-03-05'),

(7, 7, 3, 4, 3000.00, '2025-03-06'),

(8, 8, 8, 1, 600.00, '2025-03-07'),

(9, 9, 9, 7, 9100.00, '2025-03-08'),

(10, 10, 10, 10, 1000.00, '2025-03-09');

| booking_id int | customer_i int | event_ int | num_tickets int | total_cost decimal(10,2) | booking_date datetime |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 3 | 6000.00 | 2025-03-01 00:00:00 |
| 2 | 2 | 2 | 5 | 9000.00 | 2025-03-01 00:00:00 |
| 3 | 3 | 5 | 2 | 1800.00 | 2025-03-02 00:00:00 |
| 4 | 4 | 4 | 1 | 1200.00 | 2025-03-03 00:00:00 |
| 5 | 5 | 6 | 6 | 3000.00 | 2025-03-04 00:00:00 |
| 6 | 6 | 7 | 2 | 4400.00 | 2025-03-05 00:00:00 |
| 7 | 7 | 3 | 4 | 3000.00 | 2025-03-06 00:00:00 |
| 8 | 8 | 8 | 1 | 600.00 | 2025-03-07 00:00:00 |
| 9 | 9 | 9 | 7 | 9100.00 | 2025-03-08 00:00:00 |
| 10 | 10 | 10 | 10 | 1000.00 | 2025-03-09 00:00:00 |

ygokul720@gmail.com

2. Write a SQL query to list all Events.

SELECT * FROM Event;

| event_id int | event_name varchar(100) | event_date date | event_time time | venue int | total_seats int | available_seats int | ticket_price decimal(10,2) | event_type enum('Movie','Sports', |
|---|---|---|---|---|---|---|---|---|
| 1 | IPL Cup Final | 2025-04-25 | 18:30:00 | 2 | 30000 | 10000 | 2500.00 | Sports |
| 2 | Rocking Beats Live | 2025-04-10 | 20:00:00 | 3 | 8000 | 4500 | 1800.00 | Concert |
| 3 | Blockbuster Premiere | 2025-03-22 | 17:00:00 | 4 | 200 | 50 | 350.00 | Movie |
| 4 | Shakespeare Drama | 2025-04-01 | 19:30:00 | 5 | 500 | 200 | 500.00 | Movie |
| 5 | EDM Night Festival | 2025-04-12 | 21:00:00 | 6 | 10000 | 7000 | 2200.00 | Concert |
| 6 | Comedy Show | 2025-04-08 | 18:00:00 | 8 | 300 | 100 | 700.00 | Movie |
| 7 | Classical Music Evening | 2025-04-15 | 19:00:00 | 7 | 1500 | 750 | 1500.00 | Concert |
| 8 | Championship League | 2025-04-30 | 16:00:00 | 10 | 25000 | 5000 | 2000.00 | Sports |
| 9 | Action Movie Release | 2025-03-29 | 15:00:00 | 9 | 300 | 150 | 300.00 | Movie |
| 10 | Folk Music Night | 2025-04-18 | 18:30:00 | 1 | 600 | 250 | 900.00 | Concert |

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

SELECT * FROM Event WHERE available_seats > 0;

| event_name varchar(100) | available_seats int |
|---|---|
| IPL Cup Final | 10000 |
| Rocking Beats Live | 4500 |
| Blockbuster Premiere | 50 |
| Shakespeare Drama | 200 |
| EDM Night Festival | 7000 |
| Comedy Show | 100 |
| Classical Music Evening | 750 |
| Championship League | 5000 |
| Action Movie Release | 150 |
| Folk Music Night | 250 |

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

SELECT * FROM Event WHERE event_name LIKE '%cup%';

| event_id int | event_name varchar(100) | event_date date | event_time time | venue int | total_seats int | available_seats int | ticket_price decimal(10,2) | event_type enum('Movie','Sports', |
|---|---|---|---|---|---|---|---|---|
| 1 | IPL Cup Final | 2025-04-25 | 18:30:00 | 2 | 30000 | 10000 | 2500.00 | Sports |

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

SELECT * FROM Event WHERE ticket_price BETWEEN 1000 AND 2500;

| event_name varchar(100) | ticket_price decimal(10,2) |
|---|---|
| IPL Cup Final | 2500.00 |
| Rocking Beats Live | 1800.00 |
| EDM Night Festival | 2200.00 |
| Classical Music Evening | 1500.00 |
| Championship League | 2000.00 |

ygokul720@gmail.com

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

SELECT * FROM Event WHERE event_date BETWEEN '2025-04-01' AND '2025-05-15';

| event_id int | event_name varchar(100) | event_date date | event_time time | venue int | total_seats int | available_seats int | ticket_price decimal(10,2) | event_type enum('Movie','Sports', |
|---|---|---|---|---|---|---|---|---|
| 2 | Rocking Beats Live | 2025-04-10 | 20:00:00 | 3 | 8000 | 4500 | 1800.00 | Concert |
| 4 | Shakespeare Drama | 2025-04-01 | 19:30:00 | 5 | 500 | 200 | 500.00 | Movie |
| 5 | EDM Night Festival | 2025-04-12 | 21:00:00 | 6 | 10000 | 7000 | 2200.00 | Concert |
| 6 | Comedy Show | 2025-04-08 | 18:00:00 | 8 | 300 | 100 | 700.00 | Movie |
| 7 | Classical Music Evening | 2025-04-15 | 19:00:00 | 7 | 1500 | 750 | 1500.00 | Concert |
| 10 | Folk Music Night | 2025-04-18 | 18:30:00 | 1 | 600 | 250 | 900.00 | Concert |

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

SELECT * FROM Event

WHERE available_seats > 0 AND event_name LIKE '%Concert%';

| event_id int | event_name varchar(100) | event_date date | event_time time | venue int | total_seats int | available_seats int | ticket_price decimal(10,2) | event_type enum('Movie','Sports', |
|---|---|---|---|---|---|---|---|---|

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

SELECT * FROM Customer LIMIT 5 OFFSET 5;

| customer_id int | customer_name varchar(100) | email varchar(100) | phone_number varchar(15) |
|---|---|---|---|
| 6 | Sneha Kapoor | sneha@example.com | 9456123000 |
| 7 | Vikram Joshi | vikram@example.com | 9500654321 |
| 8 | Pooja Nair | pooja@example.com | 9234567890 |
| 9 | Aditya Verma | aditya@example.com | 9876500000 |
| 10 | Lakshmi Rao | lakshmi@example.com | 9998800000 |

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

SELECT * FROM Booking WHERE num_tickets > 4

| booking_id int | customer_i int | event_ int | num_tickets int | total_cost decimal(10,2) | booking_date datetime |
|---|---|---|---|---|---|
| 2 | 2 | 2 | 5 | 9000.00 | 2025-03-01 00:00:00 |
| 5 | 5 | 6 | 6 | 3000.00 | 2025-03-04 00:00:00 |
| 9 | 9 | 9 | 7 | 9100.00 | 2025-03-08 00:00:00 |
| 10 | 10 | 10 | 10 | 1000.00 | 2025-03-09 00:00:00 |

ygokul720@gmail.com

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

SELECT * FROM Customer WHERE phone_number LIKE '%000';

| customer_id int | customer_name varchar(100) | email varchar(100) | phone_number varchar(15) |
|---|---|---|---|
| 1 | Arjun Rao | arjun@example.com | 9876543000 |
| 6 | Sneha Kapoor | sneha@example.com | 9456123000 |
| 9 | Aditya Verma | aditya@example.com | 9876500000 |
| 10 | Lakshmi Rao | lakshmi@example.com | 9998800000 |

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

SELECT * FROM Event WHERE total_seats > 15000 ORDER BY total_seats DESC;

| event_id int | event_name varchar(100) | event_date date | event_time time | venue int | total_seats int | available_seats int | ticket_price decimal(10,2) | event_type enum('Movie','Sports', |
|---|---|---|---|---|---|---|---|---|
| 1 | IPL Cup Final | 2025-04-25 | 18:30:00 | 2 | 30000 | 10000 | 2500.00 | Sports |
| 8 | Championship League | 2025-04-30 | 16:00:00 | 10 | 25000 | 5000 | 2000.00 | Sports |

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

SELECT * FROM Event

WHERE event_name NOT LIKE 'x%'

AND event_name NOT LIKE 'y%'

AND event_name NOT LIKE 'z%';

| event_id int | event_name varchar(100) | event_date date | event_time time | venue int | total_seats int | available_seats int | ticket_price decimal(10,2) | event_type enum('Movie','Sports', |
|---|---|---|---|---|---|---|---|---|
| 1 | IPL Cup Final | 2025-04-25 | 18:30:00 | 2 | 30000 | 10000 | 2500.00 | Sports |
| 2 | Rocking Beats Live | 2025-04-10 | 20:00:00 | 3 | 8000 | 4500 | 1800.00 | Concert |
| 3 | Blockbuster Premiere | 2025-03-22 | 17:00:00 | 4 | 200 | 50 | 350.00 | Movie |
| 4 | Shakespeare Drama | 2025-04-01 | 19:30:00 | 5 | 500 | 200 | 500.00 | Movie |
| 5 | EDM Night Festival | 2025-04-12 | 21:00:00 | 6 | 10000 | 7000 | 2200.00 | Concert |
| 6 | Comedy Show | 2025-04-08 | 18:00:00 | 8 | 300 | 100 | 700.00 | Movie |
| 7 | Classical Music Evening | 2025-04-15 | 19:00:00 | 7 | 1500 | 750 | 1500.00 | Concert |
| 8 | Championship League | 2025-04-30 | 16:00:00 | 10 | 25000 | 5000 | 2000.00 | Sports |
| 9 | Action Movie Release | 2025-03-29 | 15:00:00 | 9 | 300 | 150 | 300.00 | Movie |
| 10 | Folk Music Night | 2025-04-18 | 18:30:00 | 1 | 600 | 250 | 900.00 | Concert |

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

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

```
SELECT event_name, AVG(ticket_price) AS average_price
FROM event
GROUP BY event_name;
```

| * event_name varchar(100) | average_price |
|---|---|
| IPL Cup Final | 2500.000000 |
| Rocking Beats Live | 1800.000000 |
| Blockbuster Premiere | 350.000000 |
| Shakespeare Drama | 500.000000 |
| EDM Night Festival | 2200.000000 |
| Comedy Show | 700.000000 |
| Classical Music Evening | 1500.000000 |
| Championship League | 2000.000000 |
| Action Movie Release | 300.000000 |
| Folk Music Night | 900.000000 |

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

```
SELECT e.event_name, SUM(b.total_cost) AS total_revenue
FROM booking b
JOIN event e ON b.event_id = e.event_id
GROUP BY e.event_name;
```

| event_name varchar | total_revenue decimal |
|---|---|
| IPL Cup Final | 6000.00 |
| Rocking Beats Live | 9000.00 |
| EDM Night Festival | 1800.00 |
| Shakespeare Drama | 1200.00 |
| Comedy Show | 3000.00 |
| Classical Music Evening | 4400.00 |
| Blockbuster Premiere | 3000.00 |
| Championship League | 600.00 |
| Action Movie Release | 9100.00 |
| Folk Music Night | 1000.00 |

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

```
SELECT e.event_name, SUM(b.num_tickets) AS total_tickets
FROM booking b
JOIN event e ON b.event_id = e.event_id
```

```
GROUP BY e.event_name
ORDER BY total_tickets DESC
LIMIT 1;
```

| event_name varchar | total_tickets decimal |
|---|---|
| Folk Music Night | 10 |

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

```
SELECT 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 e.event_name;
```

| event_name varchar | total_tickets_sold decimal |
|---|---|
| IPL Cup Final | 3 |
| Rocking Beats Live | 5 |
| EDM Night Festival | 2 |
| Shakespeare Drama | 1 |
| Comedy Show | 6 |
| Classical Music Evening | 2 |
| Blockbuster Premiere | 4 |
| Championship League | 1 |
| Action Movie Release | 7 |
| Folk Music Night | 10 |

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

```
SELECT e.event_name
FROM event e
LEFT JOIN booking b ON e.event_id = b.event_id
WHERE b.booking_id IS NULL;
```

| event_name varchar |
|---|

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

```
SELECT c.customer_name, SUM(b.num_tickets) AS total_tickets
FROM booking b
```

ygokul720@gmail.com

```
JOIN customer c ON b.customer_id = c.customer_id
GROUP BY c.customer_id
ORDER BY total_tickets DESC
LIMIT 1;
```

| customer_name varchar | total_tickets decimal |
|---|---|
| Lakshmi Rao | 10 |

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

```
SELECT
  e.event_name,
  MONTH(b.booking_date) AS booking_month,
  YEAR(b.booking_date) AS booking_year,
  SUM(b.num_tickets) AS total_tickets
FROM booking b
JOIN event e ON b.event_id = e.event_id
GROUP BY e.event_name, YEAR(b.booking_date), MONTH(b.booking_date);
```

| event_name varchar | booking_month int | booking_year int | total_tickets decimal |
|---|---|---|---|
| IPL Cup Final | 3 | 2025 | 3 |
| Rocking Beats Live | 3 | 2025 | 5 |
| EDM Night Festival | 3 | 2025 | 2 |
| Shakespeare Drama | 3 | 2025 | 1 |
| Comedy Show | 3 | 2025 | 6 |
| Classical Music Evening | 3 | 2025 | 2 |
| Blockbuster Premiere | 3 | 2025 | 4 |
| Championship League | 3 | 2025 | 1 |
| Action Movie Release | 3 | 2025 | 7 |
| Folk Music Night | 3 | 2025 | 10 |

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

```
SELECT v.venue_name, AVG(e.ticket_price) AS avg_price
FROM event e
JOIN venue v ON e.venue_id = v.venue_id
GROUP BY v.venue_name;
```

ygokul720@gmail.com

| venue_name varchar | avg_price decimal |
|---|---|
| Stadium Arena | 2500.000000 |
| Open Air Theatre | 1800.000000 |
| Galaxy Cinema | 350.000000 |
| Drama Center | 500.000000 |
| Mega Event Grounds | 2200.000000 |
| Cultural Hall | 700.000000 |
| Concert Pavilion | 1500.000000 |
| Athletic Complex | 2000.000000 |
| Multiplex Grand | 300.000000 |
| City Hall | 900.000000 |

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

```
SELECT e.event_type, SUM(b.num_tickets) AS total_tickets
FROM booking b
JOIN event e ON b.event_id = e.event_id
GROUP BY e.event_type;
```

| event_type string | total_tickets decimal |
|---|---|
| Sports | 4 |
| Concert | 19 |
| Movie | 18 |

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

```
SELECT YEAR(b.booking_date) AS year, SUM(b.total_cost) AS total_revenue
FROM booking b
GROUP BY YEAR(b.booking_date);
```

| year | total_revenue |
|---|---|
| 2025 | 39100.00 |

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

```
SELECT c.customer_name, COUNT(DISTINCT b.event_id) AS event_count
FROM booking b
JOIN customer c ON b.customer_id = c.customer_id
GROUP BY c.customer_id
HAVING event_count > 1;
```

| customer_name varchar | event_count bigint |
|---|---|

ygokul720@gmail.com

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

```
SELECT c.customer_name, SUM(b.total_cost) AS total_spent
FROM booking b
JOIN customer c ON b.customer_id = c.customer_id
GROUP BY c.customer_id;
```

| customer_name varchar | total_spent decimal |
|---|---|
| Arjun Rao | 6000.00 |
| Meera Iyer | 9000.00 |
| Ravi Kumar | 1800.00 |
| Divya Sharma | 1200.00 |
| Krishna Menon | 3000.00 |
| Sneha Kapoor | 4400.00 |
| Vikram Joshi | 3000.00 |
| Pooja Nair | 600.00 |
| Aditya Verma | 9100.00 |
| Lakshmi Rao | 1000.00 |

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

```
SELECT e.event_type, v.venue_name, AVG(e.ticket_price) AS avg_price
FROM event e
JOIN venue v ON e.venue_id = v.venue_id
GROUP BY e.event_type, v.venue_name;
```

| event_type string | venue_name varchar | avg_price decimal |
|---|---|---|
| Sports | Stadium Arena | 2500.000000 |
| Concert | Open Air Theatre | 1800.000000 |
| Movie | Galaxy Cinema | 350.000000 |
| Movie | Drama Center | 500.000000 |
| Concert | Mega Event Grounds | 2200.000000 |
| Movie | Cultural Hall | 700.000000 |
| Concert | Concert Pavilion | 1500.000000 |
| Sports | Athletic Complex | 2000.000000 |
| Movie | Multiplex Grand | 300.000000 |
| Concert | City Hall | 900.000000 |

ygokul720@gmail.com

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

```
SELECT c.customer_name, SUM(b.num_tickets) AS tickets_last_30_days
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;
```

| customer_name varchar | tickets_last_30_days decimal |
|---|---|
| Arjun Rao | 3 |
| Meera Iyer | 5 |
| Ravi Kumar | 2 |
| Divya Sharma | 1 |
| Krishna Menon | 6 |
| Sneha Kapoor | 2 |
| Vikram Joshi | 4 |
| Pooja Nair | 1 |
| Aditya Verma | 7 |
| Lakshmi Rao | 10 |

**Tasks 4: Subquery and its types**

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

```
SELECT venue_name,
    (SELECT AVG(ticket_price)
     FROM event e
     WHERE e.venue_id = v.venue_id) AS avg_ticket_price
FROM venue v;
```

| venue_name varchar | avg_ticket_price decimal |
|---|---|
| City Hall | 900.000000 |
| Stadium Arena | 2500.000000 |
| Open Air Theatre | 1800.000000 |
| Galaxy Cinema | 350.000000 |
| Drama Center | 500.000000 |
| Mega Event Grounds | 2200.000000 |
| Concert Pavilion | 1500.000000 |
| Cultural Hall | 700.000000 |
| Multiplex Grand | 300.000000 |
| Athletic Complex | 2000.000000 |

ygokul720@gmail.com

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

```
SELECT event_name
FROM event
WHERE available_seats < (total_seats / 2);
```

| * event_name varchar(100) |
|---|
| IPL Cup Final |
| Blockbuster Premiere |
| Shakespeare Drama |
| Comedy Show |
| Championship League |
| Folk Music Night |

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

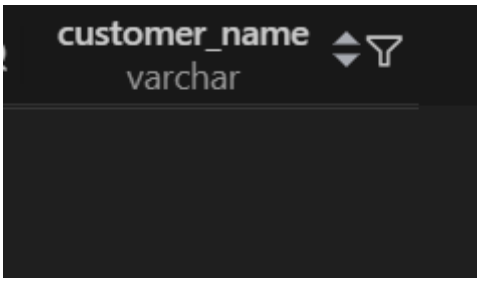
```
SELECT e.event_name,
    SUM(b.num_tickets) AS total_tickets_sold
FROM event e
JOIN booking b ON e.event_id = b.event_id
GROUP BY e.event_name;
```

| event_name varchar | total_tickets_sold decimal |
|---|---|
| IPL Cup Final | 3 |
| Rocking Beats Live | 5 |
| EDM Night Festival | 2 |
| Shakespeare Drama | 1 |
| Comedy Show | 6 |
| Classical Music Evening | 2 |
| Blockbuster Premiere | 4 |
| Championship League | 1 |
| Action Movie Release | 7 |
| Folk Music Night | 10 |

ygokul720@gmail.com

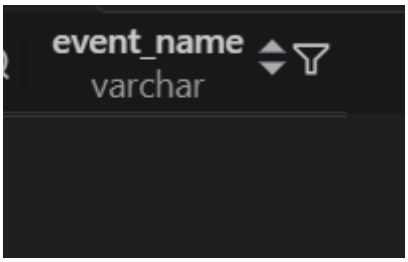
4. Find Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.

```
SELECT c.customer_name
FROM customer c
WHERE NOT EXISTS (
  SELECT 1
  FROM booking b
  WHERE b.customer_id = c.customer_id
);
```




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

```
SELECT event_name
FROM event
WHERE event_id NOT IN (
  SELECT DISTINCT event_id
  FROM booking
);
```




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

```
SELECT e.event_type, SUM(t.total_tickets) AS total_tickets
FROM (
  SELECT event_id, SUM(num_tickets) AS total_tickets
  FROM booking
  GROUP BY event_id
```

ygokul720@gmail.com

```
) t
JOIN event e ON e.event_id = t.event_id
GROUP BY e.event_type;
```

| event_type string | total_tickets decimal |
|---|---|
| Sports | 4 |
| Concert | 19 |
| Movie | 18 |

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

```
SELECT event_name, ticket_price
FROM event
WHERE ticket_price > (
   SELECT AVG(ticket_price) FROM event
);
```

| event_name varchar | ticket_price decimal |
|---|---|
| IPL Cup Final | 2500.00 |
| Rocking Beats Live | 1800.00 |
| EDM Night Festival | 2200.00 |
| Classical Music Evening | 1500.00 |
| Championship League | 2000.00 |

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

```
SELECT c.customer_name,
      (SELECT SUM(b.num_tickets * e.ticket_price)
       FROM booking b
       JOIN event e ON b.event_id = e.event_id
       WHERE b.customer_id = c.customer_id) AS total_revenue
```

ygokul720@gmail.com

```
FROM customer c;
```

| customer_name varchar | total_revenue decimal |
|---|---|
| Arjun Rao | 7500.00 |
| Meera Iyer | 9000.00 |
| Ravi Kumar | 4400.00 |
| Divya Sharma | 500.00 |
| Krishna Menon | 4200.00 |
| Sneha Kapoor | 3000.00 |
| Vikram Joshi | 1400.00 |
| Pooja Nair | 2000.00 |
| Aditya Verma | 2100.00 |
| Lakshmi Rao | 9000.00 |

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

```
SELECT DISTINCT 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_name varchar |
|---|
| Lakshmi Rao |

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

```
SELECT event_type, SUM(total_tickets) AS total_tickets_sold
FROM (
```

```
    SELECT e.event_type, b.num_tickets AS total_tickets
    FROM booking b
    JOIN event e ON b.event_id = e.event_id
) sub
GROUP BY event_type;
```

| event_type string | total_tickets_sold decimal |
|---|---|
| Sports | 4 |
| Concert | 19 |
| Movie | 18 |

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

```
SELECT DISTINCT customer_id
FROM booking
WHERE DATE_FORMAT(booking_date, '%Y-%m') IN (
    SELECT DISTINCT DATE_FORMAT(booking_date, '%Y-%m')
    FROM booking
);
```

| customer_id int |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |

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

```
SELECT venue_name,
    (SELECT AVG(ticket_price)
     FROM event e
     WHERE e.venue_id = v.venue_id) AS avg_ticket_price
```

FROM venue v;

| venue_name<br>varchar | avg_ticket_price<br>decimal |
|---|---|
| City Hall | 900.000000 |
| Stadium Arena | 2500.000000 |
| Open Air Theatre | 1800.000000 |
| Galaxy Cinema | 350.000000 |
| Drama Center | 500.000000 |
| Mega Event Grounds | 2200.000000 |
| Concert Pavilion | 1500.000000 |
| Cultural Hall | 700.000000 |
| Multiplex Grand | 300.000000 |
| Athletic Complex | 2000.000000 |

ygokul720@gmail.com