Assignment: Event Management & Ticketing System (MongoDB Project)

Objective: Design a backend MongoDB schema for managing events, ticketing, and user interactions such as browsing, booking, and managing tickets.

1. Database creation:

```
use eventmanagementdb
```

2. MongoDB Collections:

Users: Stores details of users including attendees and organizers.

```
db.createCollection("users");
db.users.insertMany([
  { name: "Kavya Nair", email: "kavyan@gmail.com", role: "attendee", phone: "9523989523" },
  { name: "Himani Dhawan", email: "himani@gmail.com", role: "organizer", phone: "9952398562" },
  { name: "Aarav Khanna", email: "aaravk@gmail.com", role: "attendee", phone: "9239856321" },
  { name: "Lavanya Mishra", email: "lavanya@gmail.com", role: "attendee", phone: "9859756945" },
  { name: "Kunal Kumar", email: "kunal@gmail.com", role: "organizer", phone: "9888965789" }
])
```

Events: Stores details of event including title, category, etc.

```
db.createCollection("events");
db.events.insertMany([
  {
    title: "Soul Night",
    category: "Music",
    venue: "Delhi Stadium",
    date: "2026-02-20",
    price: 1200,
```

```
  totalTickets: 500,
  availableTickets: 300,
  status: "upcoming"
},
{
  title: "Tech Conference",
  category: "Tech",
  venue: "Bangalore Convention Center",
  date: "2026-03-10",
  price: 1500,
  totalTickets: 400,
  availableTickets: 250,
  status: "upcoming"
},
{
  title: "Art Exhibition",
  category: "Arts",
  venue: "Delhi Art Gallery",
  date: "2026-01-15",
  price: 500,
  totalTickets: 200,
  availableTickets: 0,
  status: "completed"
},
{
  title: "Startup Meetup",
  category: "Business",
  venue: "Noida Hub",
  date: "2026-04-05",
  price: 800,
  totalTickets: 300,
  availableTickets: 180,
  status: "upcoming"
```

```
  },
  {
    title: "Badminton Match",
    category: "Sports",
    venue: "Delhi Stadium",
    date: "2026-02-28",
    price: 1000,
    totalTickets: 600,
    availableTickets: 420,
    status: "upcoming"
  }
])
```

Tickets: Stores details of tickets including user, quantity, etc.

```
db.createCollection("tickets");
db.tickets.insertMany([
  {
    userEmail: "kavyan@gmail.com",
    eventTitle: "Soul Night",
    quantity: 2,
    totalAmount: 2400,
    bookingDate: new Date(),
    status: "booked"
  },
  {
    userEmail: "himani@gmail.com",
    eventTitle: "Tech Conference",
    quantity: 1,
    totalAmount: 1500,
    bookingDate: new Date(),
    status: "booked"
  },
```

```
  {
    userEmail: "lavanya@gmail.com",
    eventTitle: "Badminton Match",
    quantity: 3,
    totalAmount: 3000,
    bookingDate: new Date(),
    status: "booked"
  },
  {
    userEmail: "himani@gmail.com",
    eventTitle: "Startup Meetup",
    quantity: 1,
    totalAmount: 800,
    bookingDate: new Date(),
    status: "cancelled"
  },
  {
    userEmail: "kunal@gmail.com",
    eventTitle: "Art Exhibition",
    quantity: 2,
    totalAmount: 1000,
    bookingDate: new Date(),
    status: "booked"
  }
])
```

Categories: Stores types of category.

```
db.createCollection("categories");
db.categories.insertMany([
  { name: "Music", description: "Music concerts and shows" },
  { name: "Tech", description: "Technology events" },
  { name: "Sports", description: "Sports and games" },
```

```
  { name: "Arts", description: "Art exhibitions and workshops" },
  { name: "Business", description: "Business seminars and talks" }
])
```

3. Required Functionality

CRUD Operations: CRUD operations are used to create, retrieve, update, and delete event records.

FIND

```
db.events.find({ status: "upcoming" })
```

UPDATION

```
db.events.updateOne(
  { title: "Soul Night" },
  { $set: { availableTickets: 280 } }
)
db.events.deleteOne({ status: "cancelled" })
```

Aggregation Pipelines: Aggregation pipelines are used to generate analytical reports such as ticket sales and revenue.

```
db.tickets.aggregate([
  { $group: { _id: "$eventTitle", ticketsSold: { $sum: "$quantity" } } },
  { $sort: { ticketsSold: -1 } },
  { $limit: 5 }
])

db.tickets.aggregate([
  { $group: { _id: "$eventTitle", revenue: { $sum: "$totalAmount" } } }
])
```

```
db.tickets.aggregate([
  { $group: { _id: "$eventTitle", attendees: { $sum: "$quantity" } } }
])
```

Indexes: Creates a unique index on the email field to ensure that no two users can register with the same email address.

```
db.events.createIndex({ date: 1 })
db.users.createIndex({ email: 1 }, { unique: true })
```