

# **EventManager&TicketingSystem**

## **MongoDB Full Code Assignment**

**Student Name:** AMBER SINGH

**Submission Date:** 22 January 2026

## 1. schemas/users.schema.js

Defines structure of Users collection.

```
db.createCollection("users")

{
  name: "Riya Kapoor",
  email: "riya.kapoor@gmail.com",
  phone: "9876543211",
  role: "attendee",
  createdAt: new Date()
}
```

## 2. schemas/categories.schema.js

Defines structure of Categories collection.

```
db.createCollection("categories")

{
  name: "Technology",
  description: "Tech conferences, hackathons and workshops"
}
```

## 3. schemas/events.schema.js

Defines structure of Events collection.

```
db.createCollection("events")

{
  title: "AI Innovation Summit 2026",
  description: "Future of Artificial Intelligence",
  categoryId: ObjectId("CATEGORY_ID"),
  dateTime: ISODate("2026-04-20T10:00:00Z"),
  venue: "New Delhi Convention Center",
  organizerId: ObjectId("ORGANIZER_ID"),
  price: 3000,
  totalTickets: 500,
  availableTickets: 500,
  status: "upcoming"
}
```

## 4. schemas/tickets.schema.js

Defines structure of Tickets collection.

```
db.createCollection("tickets")

{
  eventId: ObjectId("EVENT_ID"),
  userId: ObjectId("USER_ID"),
  bookingDate: new Date(),
  quantity: 2,
  totalAmount: 6000,
  status: "booked"
}
```

## 5. crud-operations/users.crud.js

```
db.users.insertOne({
  name: "Aman Verma",
  email: "aman.verma@gmail.com",
  phone: "9988776655",
  role: "organizer",
  createdAt: new Date()
})

db.users.find({role:"organizer"}) db.users.updateOne(
  { email: "aman.verma@gmail.com"},
  { $set: { phone: "9123456789" } }
)

db.users.deleteOne({ email: "aman.verma@gmail.com"})
```

## 6. crud-operations/events.crud.js

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

db.events.updateOne(
  { title: "AI Innovation Summit 2026"},
  { $inc: { availableTickets: -1 } }
)
```

## 7. crud-operations/tickets.crud.js

```
db.tickets.insertOne({
  eventId:ObjectId("EVENT_ID"),
  userId: ObjectId("USER_ID"),
  bookingDate: new Date(),
  quantity: 3,
  totalAmount:9000,
  status: "booked"
})

db.tickets.updateOne(
  { _id: ObjectId("TICKET_ID") },
  { $set: { status: "cancelled" } }
)
```

## 8. aggregations/topEventsByTicketSales.js

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

## 9. aggregations/revenueByOrganizer.js

```

db.events.aggregate([
  {
    $lookup: {
      from: "tickets",
      localField: "_id",
      foreignField: "eventId",
      as: "ticketData"
    }
  },
  { $unwind: "$ticketData"},
  { $match: { "ticketData.status": "booked" } },
  {
    $group: {
      _id: "$organizerId",
      totalRevenue: { $sum: "$ticketData.totalAmount" }
    }
  }
])

```

## 10. aggregations/attendeesPerEvent.js

```

db.tickets.aggregate([
  { $match: { status: "booked" } },
  { $group: { _id: "$eventId", attendees: { $sum: "$quantity" } } }
])

```

## 11. indexes/createIndexes.js

```

db.users.createIndex({email:1},{unique:true})
db.events.createIndex({ dateTime: 1 })
db.events.createIndex({ categoryId: 1 })

```

## 12. End Note

This PDF contains all MongoDB codes organized exactly according to the project folder structure. Each file represents real MongoDB shell commands suitable for execution and evaluation.