3-Day MongoDB + Mongoose Training Program

Day 1 — MongoDB & Mongoose Fundamentals

What to Learn (Core Basics)

- What is MongoDB? Document-based NoSQL database.
- Install MongoDB & Compass (GUI).
- MongoDB data types (String, Number, Boolean, Date, ObjectId, Array, Object).
- Create Database & Collections.
- Basic CRUD operations in MongoDB:
    - `insertOne`, `insertMany`
    - `find`, `findOne`
    - `updateOne`, `updateMany`
    - `deleteOne`, `deleteMany`
- Introduction to **Mongoose**:
    - Connecting Node.js to MongoDB.
    - Creating Schema & Model.
    - Performing CRUD with Mongoose.

Task: ***Student Record System***

1. Create `students` collection with fields: name, age, email, course.
2. Insert at least 10 sample records.
3. Write queries to:
    - Fetch all students in a specific course.
    - Update a student's email.
    - Delete a student by name.
4. Use **Mongoose schema** to enforce rules (e.g., email is required, age must be a number).

Deliverable: Working Node.js app connected to MongoDB with student data CRUD.

Day 2 — Relationships, Validation & Aggregation

What to Learn (Intermediate Level)

- Schema Types & Validations (required, min/max, enum, match, custom validators).
- Embedding vs Referencing (One-to-One, One-to-Many, Many-to-Many).
- `populate()` for relationships.
- Aggregation Framework basics:
  - `$match`, `$group`, `$sort`, `$sum`, `$avg`, `$lookup`.
- Indexing basics: Why indexes improve performance.

Task: *Library Management System*

1. Create 2 collections:
   - `books`: title, author, category, price.
   - `users`: name, email, borrowedBooks (array of ObjectIds).
2. Insert sample data (10 books, 5 users).
3. Implement queries:
   - Find all books by category.
   - Find which books a user borrowed (`populate`).
   - Get average price of books per category (aggregation).
   - Create an index on `email` (users).
4. Add validations in Mongoose: email format, price > 0.

Deliverable: Library app with relationships, aggregation queries, and indexes.

Day 3 — Advanced MongoDB Features

What to Learn (Advanced Level)

- Indexing in detail: compound indexes, unique indexes, text indexes.
- Aggregation Advanced: `$unwind`, `$facet`, `$bucket`, `$project`.
- Transactions in MongoDB (atomic operations).
- Sharding basics: Why/when it's needed.
- MongoDB Atlas overview (cloud database).
- Performance tuning & best practices.

Task: *E-Commerce Analytics System*

1. Create collections:
   - `products`: name, category, price, stock.
   - `orders`: userId, productId, quantity, totalPrice, date.
2. Queries to perform:

    - Total revenue per category (aggregation).
    - Top 3 most sold products.
    - Daily sales report using `$group` by date.
    - Use **text index** on product name for search.
    - Demonstrate a simple **transaction**: deduct stock when an order is placed.
3. (Optional) Discuss how **sharding** would apply if the data grows to millions of orders.

Deliverable: E-Commerce database with analytics queries, indexes, and transactions.

Capstone Project (End of Day 3)

Build a **Mini E-Commerce Backend** using **Node.js + Express + MongoDB + Mongoose**:

- User signup/login (with validation).
- CRUD for products & orders.
- Aggregation for analytics (e.g., revenue, top products).
- Indexes for performance.
- Simulated transaction for stock deduction.
