MongoDB Assignment — "BookBazaar" Marketplace

0) Learning Outcomes

- Model documents (embed vs reference) with basic validation.
- Perform CRUD confidently.
- Write multi-stage **aggregation pipelines** (lookup, unwind, group, facet, window).
- Apply **indexing** + verify with explain.

1) Case Study Overview

BookBazaar is an online marketplace where:

- **Customers** browse books, place orders, and leave reviews.
- **Vendors** list books, manage price/stock.
- The platform needs fast search, analytics, and clean operations.

1.1 Suggested Collections (students may adjust with justification)

1. users — _id, name, email, phone, role(customer|vendor|admin), addresses[], createdAt
2. vendors — _id, name, legalName, city, tags[], rating, status(ACTIVE|SUSPENDED), createdAt
3. books — _id, title, subtitle, authors[], publisher, publishedYear, categories[], price, currency, isbn13, pages, language, vendorId, tags[], stock, description
4. orders — _id, userId, items[{bookId, qty, priceAtPurchase}], status(PLACED|PAID|SHIPPED|DELIVERED|CANCELLED), shippingAddress, payment{method, paidAt}, createdAt, deliveredAt
5. reviews — _id, bookId, userId, rating(1–5), title, body, createdAt
6. inventory_logs — _id, bookId, vendorId, delta(+/-int), reason(restock|sale|return|adjustment), createdAt
7. sessions — _id, userId, token, createdAt (for TTL demo)

1.2 Seed Size (minimums)

- users ≥ 12, vendors ≥ 5, books ≥ 60, orders ≥ 80, reviews ≥ 120, inventory_logs ≥ 150, sessions ≥ 10
  Data must be plausible (orders reference real books, stocks align with logs, etc.).

2) Environment & Submission

- DB: bookbazaar_db (local MongoDB).
- Tools: MongoDB Shell or Compass; optional driver scripts.

- Timestamps: ISODate.

3) Part A — Modeling & Validation

Tasks

1. Create all 7 collections.
2. Add JSON Schema **validators**:
    - books: required title, price, vendorId, stock(≥0), authors(array ≥1).
    - reviews: required bookId, userId, rating(int 1–5).
3. Insert ≥ 5 test docs per collection to verify validators; then bulk seed.
4. Show validator **rejections** (screenshots of failures).

Deliverable: 01_modeling_setup.md with schema snippets + screenshots.

4) Part B — CRUD Scenarios

Write the **exact commands** (shell or driver). Briefly label each.

Create

B1) Vendor adds a **new book** (initial stock + tags).
B2) Register a **user** with two addresses (role customer).
B3) **Bulk insert** 10 books for a vendor (same publisher, different titles).

Read

B4) Books in category "Data Science" with 20 ≤ price ≤ 60; **project** title, price, vendorId, **sort** by price asc.
B5) **Top 5 newest** books (publishedYear) with stock > 0.
B6) For a userId, fetch **last 3 orders** with fields: status, createdAt, itemCount.

Update

B7) **Increase price by 5%** for all books of a given publisher where price < 25.
B8) **Upsert review** (if exists: update rating/body; else insert).
B9) **Normalize city**: change user address "Lahore" → "Lahore City" only where label: "Home" (arrayFilters).

Delete

B10) Delete **orphan reviews** (where bookId no longer exists).
B11) Delete vendors with status: "SUSPENDED" and **no books**.
B12) Delete sessions older than 7 days (if TTL not used yet).

Deliverable: 02_crud_queries.js (+ notes if anything tricky).

5) Part C — Aggregations

Each pipeline should be commented by stage.

C1) **Monthly sales by category (last 6 months)**: orders.items.qty priceAtPurchase, $lookup books, $unwind categories, $group by {month, category}.
C2) **Top 10 best-sellers**: rank by **total qty**, output bookId, title, totalQty, totalRevenue.
C3) **Also-bought**: for a target bookId, top 5 co-purchased books from same orders.
C4) **Vendor dashboard (facet)**:

- salesLast30d (sum revenue)
- avgDeliveryDays (DELIVERED: deliveredAt - createdAt)
- lowStockCount (stock < 5)

C5) **Ratings by category**: join reviews→books, unwind categories, compute **median** (MongoDB 7 $percentile if available; otherwise average + note limitation).
C6) **Author leaderboard (window)**: total revenue per author in "Programming"; add rank via $setWindowFields sorted by revenue desc.
C7) **Cohort retention (60-day)**: cohort by month of **first order**; share of users ordering again within 60 days.
C8) **Inventory anomalies**: sum(delta) in inventory_logs vs books.stock; report mismatches with difference.
C9) **Price buckets**: [0–10], (10–25], (25–50], (50–100], >100 + average rating via join.
C10) **Text search & recency boost**: text index search for "machine learning deep learning" → top 10 with textScore; then **secondary sort** by publishedYear desc.

Deliverable: 03_aggregations.js (10 pipelines) + 1–2 line rationale per pipeline.

6) Part D — Indexing (Simplified)

D1) Create 4 Indexes

- **books**
  - (1) Single-field: price (asc)
  - (2) **Compound**: { categories: 1, price: 1 }
  - (3) **Text**: title, subtitle, description (single text index)
- **orders**

- (4) Compound: { userId: 1, createdAt: -1 }

Deliverables: brief reason (2–3 lines each) + list indexes screenshot.

D2) Verify with explain("executionStats")

Pick 3 queries (from B/C):

- A) category+price (uses {categories:1, price:1})
- B) user's last 3 orders (uses {userId:1, createdAt:-1})
- C) text search (uses text index)

For each: paste query, predict index usage (1 sentence), attach **post-index** explain, then 1–2 lines interpreting nReturned, totalDocsExamined.

D3) Maintenance

- List all indexes; **drop** one non-_id_ index; recreate.
- Add a short **best-practices note** (5–7 bullets).

Optional +3%: multikey index on authors and a benefiting query + explain summary.

Deliverable: 04_indexing_simplified.md.

What to Submit (folder)

bookbazaar_submission_<rollNo>/

- 01_modeling_setup.md (schemas, validators, sample inserts, screenshots)
- 02_crud_queries.js (or .ipynb) + short notes
- 03_aggregations.js + brief explanations per pipeline
- 04_indexing_simplified.md (indexes, explains, notes)
- /screenshots/ (import, validator rejections, explain outputs)
- /exports/ (small JSON exports OK)
- A brief report