**eCommerce Application - MongoDB Problem Statement**

**Objective:**  
Design and implement a scalable and flexible eCommerce database using MongoDB that supports product catalog management, user management, order processing, and analytics.

**🧩 Background:**

An eCommerce platform needs to store and manage large volumes of data, including users, products, categories, shopping carts, orders, and reviews. MongoDB’s document-oriented model provides the flexibility needed for fast iteration and scale.

**✅ Requirements:**

**1. Users Collection**

* Store user data including name, email, password (hashed), address(es), and order history.
* Support multiple addresses and roles (e.g., customer, admin).

**2. Products Collection**

* Store product data such as name, description, price, category, available stock, images, and specifications (e.g., size, color).
* Products can belong to multiple categories and have dynamic fields (e.g., electronics vs. clothing).

**3. Categories Collection**

* Hierarchical structure of categories (e.g., Electronics > Phones > Smartphones).
* Allow querying products by category/sub-category.

**4. Orders Collection**

* Store customer orders including list of items, quantities, total amount, shipping info, status (placed, shipped, delivered), and payment details.
* Support updates to order status and audit history.

**5. Cart Collection**

* Store real-time cart information per user.
* Allow adding/removing items, adjusting quantities.

**6. Reviews Collection**

* Allow customers to review products with ratings and comments.
* Enable querying average ratings per product.

**7. Analytics and Reporting**

* Query top-selling products, user purchase patterns, inventory reports.
* Support aggregation queries for sales per month, user order frequency, etc.

**🛠️ Challenges MongoDB Should Solve:**

* Schema flexibility for product types with different attributes.
* Embedding vs. referencing (e.g., orders referencing products, or embedding item snapshots).
* Performance with large datasets using indexes and aggregation pipelines.
* Sharding for scalability and high availability.