**1) What is MongoDB?**

🡺MongoDB is a popular NoSQL database that uses a document-oriented data model.

🡺 It stores data in flexible JSON-like documents, making it ideal for handling complex and unstructured data. MongoDB is known for its scalability, performance, and ease of use.

**2) What is the difference between MongoDB and SQL?**

|  |  |  |
| --- | --- | --- |
| **🡺Feature** | **MongoDB** | **SQL** |
|  |  |  |
| Data Model | Document-oriented | Relational |
| Schema | Flexible, schema-less | Rigid, defined schema |
| Data Structure | JSON-like documents | Tables, rows, and columns |
| Query Language | MongoDB Query Language (MQL) | SQL (Structured Query Language) |
| Use Cases | Real-time applications, content management, IoT, analytics | Traditional business applications, financial data, inventory management |

**3) Create a database for an online shopping app.**

🡺 **Requirements:**

* **User Management:** Registration, login, profile management.
* **Product Catalog:** Listing, searching, filtering products.
* **Shopping Cart:** Adding, removing, and saving items.
* **Checkout:** Payment processing, order placement, shipping information.
* **Order Tracking:** Order status updates, delivery tracking.
* **Reviews and Ratings:** User feedback on products.

**Database Schema:**

**1. Users**

* user\_id (Primary Key, Auto-Increment): Unique identifier for each user.
* username (Unique): Username for login.
* email (Unique): Email address for login and communication.
* password (Hash): Password stored securely.
* first\_name
* last\_name
* address
* city
* state
* country

**2. Products**

* product\_id (Primary Key, Auto-Increment): Unique identifier for each product.
* name
* description
* price
* quantity

**3. Categories**

* category\_id (Primary Key, Auto-Increment): Unique identifier for each category.
* name
* description

**4. Shopping\_Cart**

* cart\_id (Primary Key, Auto-Increment): Unique identifier for each shopping cart.
* user\_id (Foreign Key referencing Users table)
* product\_id (Foreign Key referencing Products table)

**5. Orders**

* order\_id (Primary Key, Auto-Increment): Unique identifier for each order.
* user\_id (Foreign Key referencing Users table)
* order\_date
* shipping\_address
* billing\_address
* status (e.g., "pending", "shipped", "delivered")

**6. Order\_Items**

* order\_item\_id (Primary Key, Auto-Increment): Unique identifier for each item in an order.
* order\_id (Foreign Key referencing Orders table)
* product\_id (Foreign Key referencing Products table)
* quantity
* price

**7. Reviews**

* review\_id (Primary Key, Auto-Increment): Unique identifier for each review.
* user\_id (Foreign Key referencing Users table)
* product\_id (Foreign Key referencing Products table)
* rating (e.g., 1-5 stars)

4)**Create required collections for an online shopping app and documents.**

**🡺Collections:**

* **users:** Stores information about users, such as name, email, password, and address.
* **product\_categories:** Stores information about product categories, such as name and description.
* **products:** Stores information about products, such as name, description, price, quantity, and category.
* **orders:** Stores information about orders, such as order ID, user ID, product IDs, quantity, and status.
* **reviews:** Stores information about product reviews, such as user ID, product ID, rating, and comment.

**Example**

// users collection

{

"\_id": ObjectId("64d357175a15175000000001"),

"name": "John Doe",

"email": "john@example.com",

"password": "hashed\_password",

"address": "123 Main Street"

}

// product\_categories collection

{

"\_id": ObjectId("64d357175a15175000000002"),

"name": "Electronics"

}

// products collection

{

"\_id": ObjectId("64d357175a15175000000003"),

"name": "Smartphone",

"description": "A high-end smartphone with powerful features.",

"price": 599.99,

"quantity": 100,

"category": ObjectId("64d357175a15175000000002")

}

**5) Write a command to show all data from the products collection and sort in ascending order.**

🡺db.products.find().sort({ price: 1 });

This command finds all documents in the products collection and sorts them in ascending order based on the price field.

**Q6. Update the product price for a particular product.**

db.products.updateOne({ \_id: ObjectId("64d357175a15175000000003") }, { $set: { price: 649.99 } });

**7) Write a command to delete a particular document and collection.**

🡺db.products.deleteOne({ \_id: ObjectId("64d357175a15175000000003") });

**🡺Deleting a collection:**

🡺db.dropCollection("products");