MongoDB Assignment

Part 1 – Basics of MongoDB

1. What is MongoDB and how is it different from relational databases?

   - Ans : . It's a NoSQL Database.
   - It's a document oriented data model.
   - MongoDb stores data in a flexible Json-like format
   - Relational databases consists of rows and coloumns and a unique identifier key is used it is a sql database.

2. In MongoDB, what are **databases, collections, and documents**?

   Databases : database is a highest level of data organization in a mongodb instance.A single mongodb server can handle multiple databases.it act as a container for one or more collection.

   Collections : a collection is a grouping of MongoDb documents within a specific database. collections in MongoDb are schema-less by default.Documents within the same collection have different fields and structures.

   Documents : it is a basic unit of data storage in mongoDb.It is a single record within the collection. Stored in binary or Json.

3. What format does MongoDB use to store documents ?
   MongoDb uses (Binary,Json) to store documents.

4. List three different ways to install or use MongoDB.
   1 Download and install MongoDB Community Server from the official MongoDB website.

   2 Use MongoDB Atlas.

   3 Run MongoDB with Docker.

5. Which MongoDB tool provides a graphical interface?
   MongoDB Compass is the official GUI for MongoDB. It allows you to visually explore your data, run queries, create and modify documents, view schema analysis.

6. How can you verify MongoDB installation using the terminal?
   mongod --version

Part 2 – MongoDB Commands

7. Write the command to show all databases.

   ans : show dbs

7. How do you switch to or create a new database called mydb?
Ans: use mydb

8. Write the command to create a collection called users.
db.createcollection(“users”)

9. Insert one user with fields name, age, and city.
db.user.insertone({ name:”Drishti”,age:21,city: “Delhi”})

10. Insert multiple users at once.
db.user.insertMany([{ name:”Drishti”,age:21,city: “Delhi”}, { name:”Srishti”,age:31,city: “Dehradun”}, { name:”kirti”,age:20,city: “Ambala”}])

11. Write the command to view all documents in the users collection.
Ans : db.users.find()

12. Write the command to find a user named "Alice".
db.users.find({ name: “Drishti” })

13. Write the command to find users with age > 25.
db.users.find({ age: { $gt:25 } })

Part 3 – CRUD Operations

15. Write a query to update one user’s city to "Berlin".
db.users.updateOne({ name: "Kirti" }, { $set: { city: "Berlin" } })

16. Write a query to update all users from "London" by adding a new field status: "active".
db.users.updateMany({ city: "London" },{ $set: { status: "active" } })

17. Write a query to delete one user named "Alice".
db.users.deleteOne({ name: “srishti”})

18. Write a query to delete all users from "Paris".
db.users.deleteMany({ city: “Paris”})

Part 4 – Query Operators & Projection

19. Which operator is used to check if a value is greater than another?
$gt Example : { age : {$gt:25}}

20. Which operator is used to find documents where a value is not equal?
$ne example : { age : {$ne:25}}

21. Write a query to find users whose city is either "London" or "Paris".
db.users.find({city: { $in: ["London", "Paris"] }})

22. Write a query to find users from "London OR Berlin".
db.users.find({city: { $in: ["London", "Berlin"] }})

23. Write a query to show only name and city fields while hiding _id.
db.users.find({}, { name: 1, city: 1, _id: 0 })

Part 5 – Relations & Schema Design

24. What does it mean when we say MongoDB is "schema-less"?
Ans : MongoDB being "schema-less" signifies its flexible data model, **No Predefined Table Structure:** Unlike relational databases where you must define tables and columns with specific data types before inserting data, MongoDB does not require a fixed structure for documents within a collection.

25. What is the difference between **embedding** and **referencing** in MongoDB schema design?
Ans : Embedding involves nesting related data directly within a single document. This creates a denormalized data model where all relevant information for a specific entity is contained within one document.

Advantages:

- Faster read performance as all related data is retrieved in a single query, eliminating the need for joins.
- Simpler queries and application code due to the consolidated data structure.

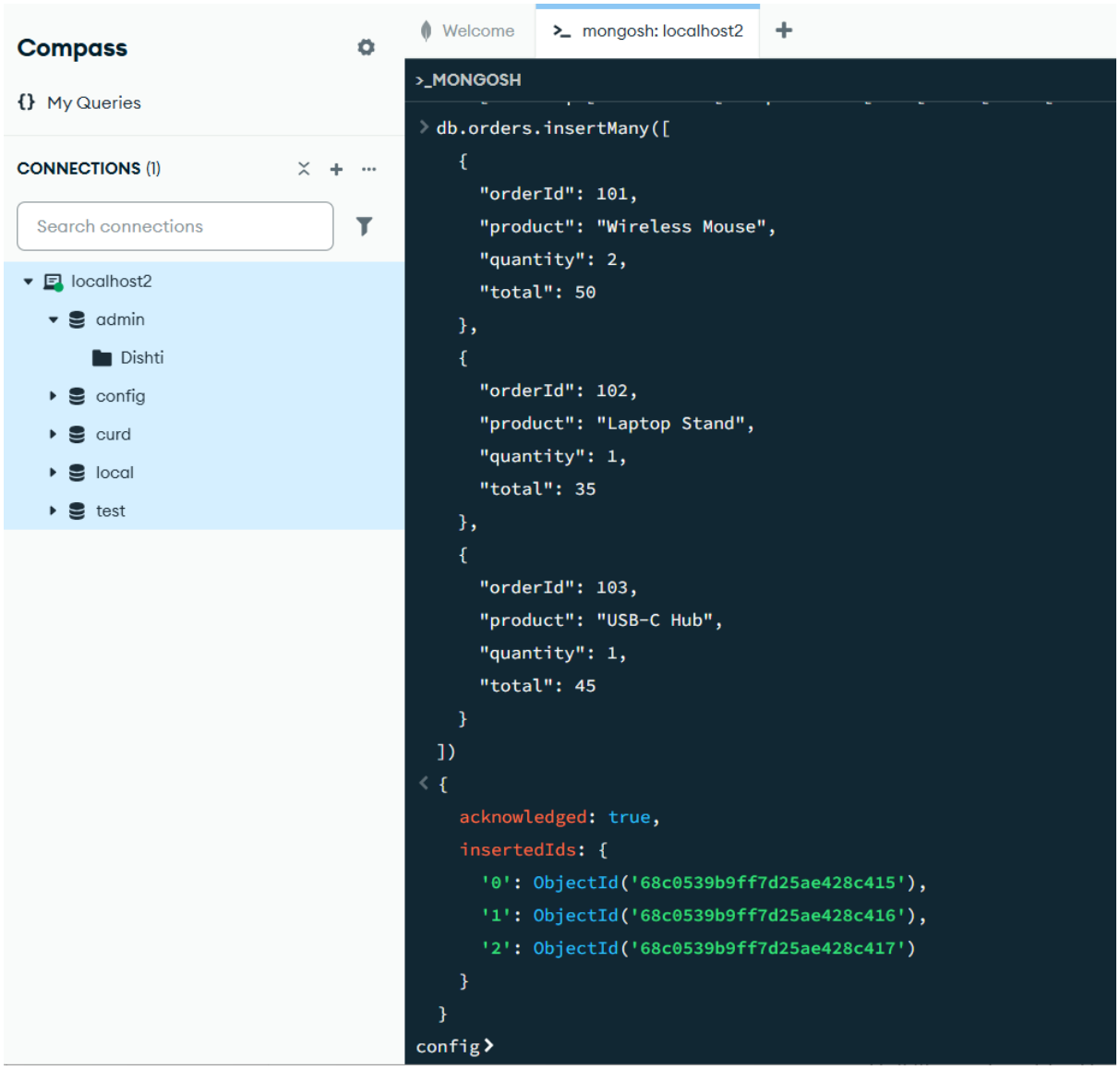
Referencing involves storing related data in separate documents and collections, linking them through identifiers (typically _id values). This creates a normalized data model, similar to relational databases.

Advantages:

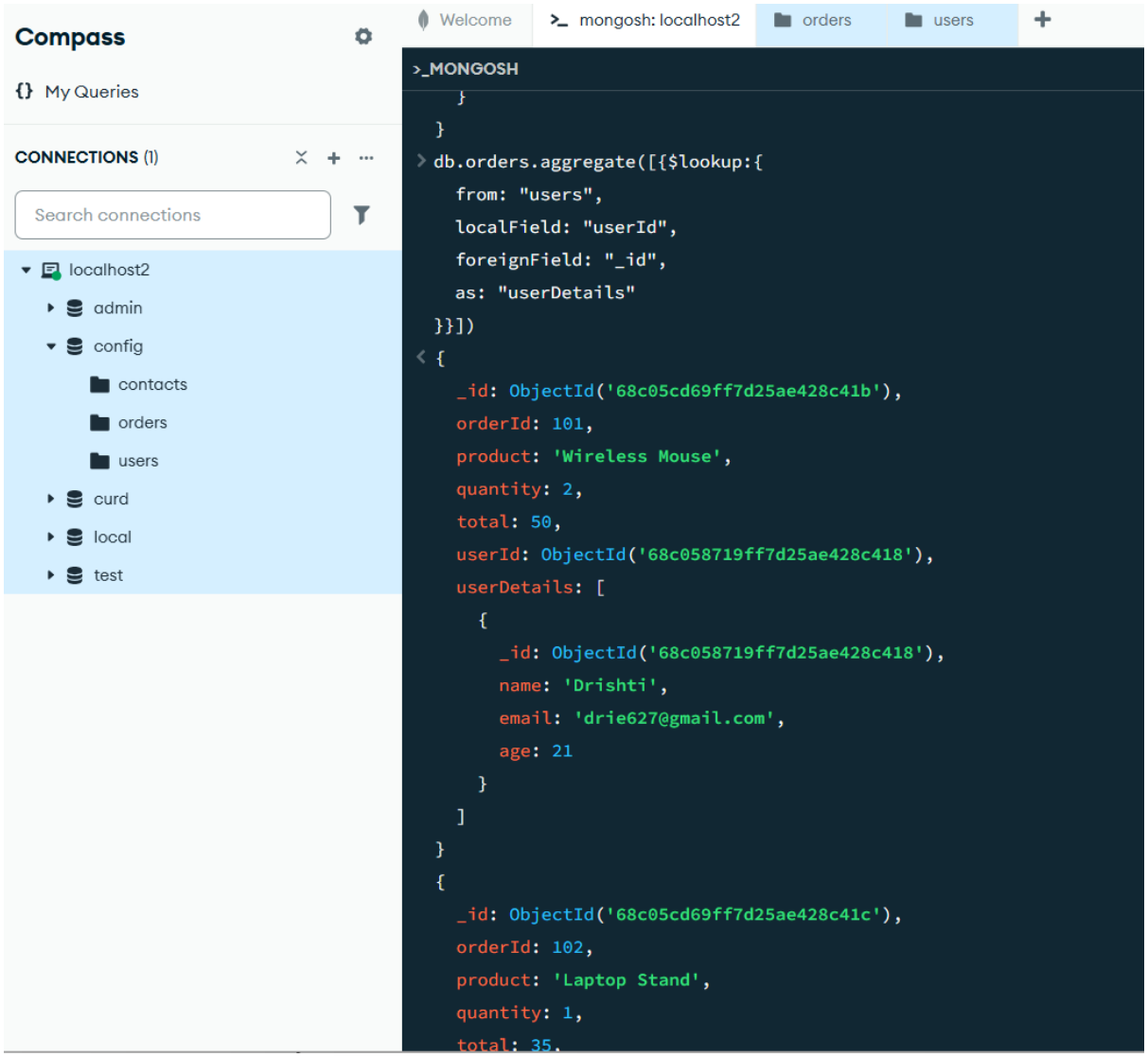
- Manages document size effectively, preventing documents from becoming too large.
- Reduces data duplication for shared or frequently updated data.
- Provides flexibility for complex relationships and independent data management.

26. Create an **embedded document** for a user with multiple orders.

Ans:



27. Create a **referenced schema** for users and orders using ObjectId.



28. Which approach is better for small related data: embedding or referencing?

Ans : For small related data embedding is preferred Related data is stored together in the same document, making reads faster and simpler. Fewer queries, Updates or inserts can be done in one operation without worrying about consistency across documents.

29. Why is schema design important even though MongoDB is schema-less?

Ans : Consistency of data, Efficient queries, Maintainability and scalability, Preventing redundancy and duplication, Validation and error prevention

---

Part 6 – Aggregation Framework

30. What is aggregation in MongoDB and why is it more powerful than find()?

Ans : Aggregation in MongoDB is a process of transforming and analysing data to compute results

like sums, averages, counts, grouping, filtering, reshaping, or sorting documents. It's used to process large datasets and return computed information.

31. Write an aggregation query using $match to find only delivered orders.
Ans : db.orders.aggregate([{ $match: { status: "delivered" } }])

32. Write an aggregation query using $project to show only customer Name and total fields.
Ans : db.orders.aggregate([{ $project: { _id: 0,customerName: 1,total: 1 } }])

33. Write an aggregation query using $group to calculate the total amount spent by each customer.
Ans : db.orders.aggregate([{ $group: { _id: "$customerName",totalSpent: { $sum: "$total" }}}])

34. What does the $unwind stage do in aggregation?
Ans : If a document has an array field with multiple elements, $unwind will create a separate document for each element in that array. This is useful when you want to work with individual items inside an array rather than the array as a whole.

35. Write an aggregation query using $lookup to join orders with customers on customerId.
Ans : db.orders.aggregate([{$lookup: {from: "customers", localField: "customerId", foreignField: "_id", as: "customerDetails" } }])

36. Write an aggregation query to find the top 2 highest orders using $sort and $limit.

Ans : db.orders.aggregate([{ $sort: { total: -1 } }, { $limit: 2 }])