Top 50 MongoDB Interview Questions and Answers

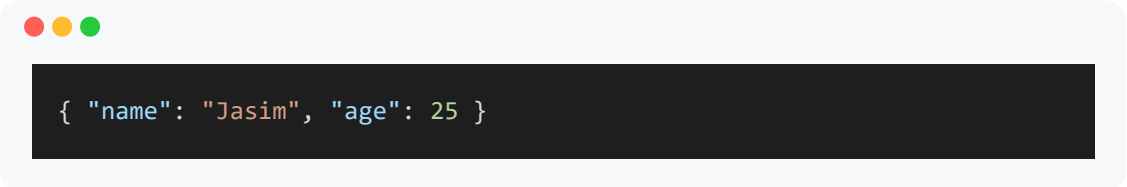
Here's a comprehensive list of **Top 50 MongoDB Interview Questions and Answers**, covering beginner to advanced concepts across CRUD operations, indexing, performance, aggregation, replication, sharding, schema design, and more:

🔰 Basic MongoDB Concepts

1. **What is MongoDB?**
    - MongoDB is a NoSQL, document-oriented database that stores data in BSON (binary JSON) format.
2. **Difference between SQL and MongoDB?**

SQL DB	MongoDB
Table	Collection
Row	Document
Column	Field
Join	Embedding/Manual

3. **What is a document in MongoDB?**
    - A JSON-like object with key-value pairs. Example:



```
{ "name": "Jasim", "age": 25 }
```

4. **What is a collection in MongoDB?**
    - A group of related documents. Similar to a table in RDBMS.
5. **What is BSON?**

- Binary JSON format used internally by MongoDB for faster traversal and storage.

🛠️ CRUD Operations

6. **How do you insert a document?**

```
db.users.insertOne({ name: "Ali", age: 30 });
```

7. **How do you update a document?**

```
db.users.updateOne({ name: "Ali" }, { $set: { age: 31 } });
```

8. **How do you delete a document?**

```
db.users.deleteOne({ name: "Ali" });
```

9. **How do you read documents?**

```
db.users.find({ age: { $gt: 20 } });
```

10. **Difference between `insertOne` and `insertMany`?**

- `insertOne`: inserts a single document.
- `insertMany`: inserts multiple documents at once.

🔎 Querying and Filters

11. **What is the use of `$in` operator?**

```
db.users.find({ age: { $in: [25, 30] } });
```

12. **Explain projection in MongoDB?**

- Include/exclude fields in output.

```
db.users.find({}, { name: 1, _id: 0 });
```

13. **What is the use of `$regex`?**

- Pattern matching.

```
db.users.find({ name: { $regex: /^A/ } });
```

14. **What is `$and`, `$or`, `$not` in MongoDB?**

```
db.users.find({ $or: [{ age: 25 }, { age: 30 }] });
```

15. **How do you sort documents?**

```
db.users.find().sort({ age: -1 }); // Descending
```

📊 Aggregation Framework

16. **What is aggregation in MongoDB?**

- Framework to process data and return computed results (like SQL `GROUP BY`).

17. **Basic aggregation example?**

```
db.users.aggregate([
  { $group: { _id: "$city", total: { $sum: 1 } } }
]);
```

18. **What is `$match` and `$group`?**

- `$match`: filters documents.
- `$group`: groups by a field and performs operations like `$sum`, `$avg`.

19. **How to use `$project` in aggregation?**

```
db.users.aggregate([
  { $project: { name: 1, age: 1 } }
]);
```

20. **Use of `$lookup`?**

- Performs a left outer join.

```
db.orders.aggregate([
  {
    $lookup: {
      from: "products",
      localField: "productId",
      foreignField: "_id",
      as: "productDetails"
    }
  }
]);
```

⚙️ Indexing and Performance

21. **Why use indexes in MongoDB?**

- Improve query performance.

22. **Create an index on a field?**

```
db.users.createIndex({ age: 1 });
```

23. **What is a compound index?**

- Index on multiple fields.

```
db.users.createIndex({ age: 1, name: -1 });
```

24. **Explain covered query.**

- When all fields in query/projection are in the index, no need to fetch document.

25. **What is an explain plan?**

- Shows how MongoDB will execute a query.

```
db.users.find({ age: 25 }).explain("executionStats");
```

🧩 Schema Design

26. **Should I embed or reference documents?**

- Embed for one-to-few relationships.
- Reference for one-to-many or many-to-many.

27. **Example of embedded documents?**

```
{ name: "John", address: { city: "Delhi", pincode: "110011" } }
```

28. **Example of referencing?**

```
{ name: "John", addressId: ObjectId("...") }
```

29. **How to ensure data consistency in MongoDB?**

- Use transactions (from MongoDB 4.0+) or design schemas carefully.

30. **Denormalization vs Normalization in MongoDB?**

- Denormalization improves read performance at cost of redundancy.

🚦 Transactions & ACID

31. **Does MongoDB support transactions?**

- Yes, multi-document ACID transactions are supported from version 4.0+ (Replica Set) and 4.2+ (Sharded).

32. **Transaction example?**

```
const session = client.startSession();
session.withTransaction(() => {
  db.accounts.updateOne({ name: "A" }, { $inc: { balance: -100 } }, { session });
  db.accounts.updateOne({ name: "B" }, { $inc: { balance: 100 } }, { session });
});
session.endSession();
```

33. **What is write concern?**

- Level of acknowledgment requested from MongoDB for write operations (e.g. `w: 1`, `w: "majority"`).

34. **What is read concern?**

- Determines the consistency and isolation properties of the data read (e.g. `local`, `majority`, `snapshot`).

35. **What is durability in MongoDB?**

- Guarantees data is persisted on disk.

🌍 Replication and High Availability

36. **What is replication in MongoDB?**

- Copies data across multiple servers for high availability.

37. **What is a replica set?**

- Group of MongoDB servers: 1 primary, multiple secondaries.

38. **How failover works in replica set?**

- If primary goes down, an eligible secondary is elected as new primary.

39. **How to check replication status?**

```
rs.status();
```

40. **What is an arbiter?**

- A member of replica set that doesn't store data but helps in elections.

🗺️ Sharding (Scalability)

41. **What is sharding in MongoDB?**

- Horizontal scaling by distributing data across multiple machines.

42. **What is a shard key?**

- Field used to partition data in sharding.

43. **What is a mongos?**

- Query router in sharded cluster.

44. **What is config server?**

- Stores metadata about sharded cluster.

45. **When should you shard a collection?**

- When a collection exceeds a certain size or traffic grows beyond a single server.

🧪 Admin and Miscellaneous

46. **How to create a user in MongoDB?**

```
db.createUser({
  user: "admin",
  pwd: "password",
  roles: ["readWrite", "dbAdmin"]
});
```

47. **How to backup MongoDB?**

- Use `mongodump` and restore with `mongorestore`.

48. **What is capped collection?**

- Fixed size collection that overwrites oldest entries when full.

49. **What is TTL index?**

- Automatically deletes documents after a certain time.

```
db.logs.createIndex({ createdAt: 1 }, { expireAfterSeconds: 3600 });
```

50. **What is the aggregation pipeline limit?**

- As of latest versions: 100 MB without `allowDiskUse: true`.
