**1. What are the key differences between SQL and NoSQL databases?**

* **SQL** databases are relational, use structured schemas, and store data in tables.
* **NoSQL** databases (like MongoDB) are non-relational, schema-less (or dynamic), and store data in flexible formats like documents (JSON/BSON).

**2. What makes MongoDB a good choice for modern applications?**

* Flexible document model (BSON).
* Schema-less design allows agile development.
* Built-in replication and sharding for high availability and scalability.
* Rich querying, indexing, and aggregation features.
* Great fit for microservices, IoT, mobile, and real-time analytics.

**3. Explain the concept of collections in MongoDB.**

* A **collection** is similar to a table in SQL.
* It stores **documents**, which are BSON-formatted records.
* Documents within a collection can have different fields and structures.

**4. How does MongoDB ensure high availability using replication?**

* MongoDB uses **replica sets**, which include:
  + One **primary** node (handles all writes).
  + Multiple **secondary** nodes (replicate data).
* If the primary fails, an automatic **failover** occurs and a secondary is elected as the new primary.

**5. What are the main benefits of MongoDB Atlas?**

* Fully managed cloud-based MongoDB service.
* Automatic backups, monitoring, and scaling.
* Global clusters and multi-region deployment.
* Integrated security, compliance, and performance tools.
* Easy integration with AWS, Azure, GCP.

**6. What is the role of indexes in MongoDB, and how do they improve performance?**

* **Indexes** allow faster data retrieval by avoiding full collection scans.
* Improve performance on read-heavy workloads.
* Support various types: single field, compound, text, geospatial, etc.

**7. Describe the stages of the MongoDB aggregation pipeline.**

Common stages include:

* $match – Filters documents.
* $group – Groups data by a key.
* $project – Shapes returned documents.
* $sort – Orders documents.
* $limit / $skip – Controls result set size.
* $lookup – Performs joins.
* $unwind – Flattens arrays.

**8. What is sharding in MongoDB? How does it differ from replication?**

* **Sharding** distributes data across multiple machines for horizontal scalability.
* **Replication** duplicates data across servers for redundancy and high availability.
* Sharding improves **scalability**; replication improves **availability**.

**9. What is PyMongo, and why is it used?**

* **PyMongo** is the official Python driver for MongoDB.
* Enables Python applications to interact with MongoDB databases.
* Supports CRUD, aggregation, transactions, and more.

**10. What are the ACID properties in the context of MongoDB transactions?**

* **A**tomicity: All-or-nothing execution.
* **C**onsistency: Keeps data in valid state.
* **I**solation: Transactions don’t interfere with each other.
* **D**urability: Changes persist after completion.
* MongoDB supports multi-document ACID transactions since version 4.0.

**11. What is the purpose of MongoDB’s explain() function?**

* Analyzes and displays the execution plan of a query.
* Helps identify whether indexes are being used and reveals potential bottlenecks.

**12. How does MongoDB handle schema validation?**

* Uses **JSON Schema**-based validation rules.
* You can define required fields, field types, and value constraints.
* Enforced during document insert and update operations.

**13. What is the difference between a primary and a secondary node in a replica set?**

* **Primary**: Accepts write and read operations.
* **Secondary**: Replicates data from primary and can serve reads (if configured).
* In case of failure, a new primary is elected from the secondaries.

**14. What security mechanisms does MongoDB provide for data protection?**

* **Authentication**: SCRAM, LDAP, Kerberos.
* **Authorization**: Role-Based Access Control (RBAC).
* **Encryption**: TLS/SSL (in transit), encryption at rest.
* **Auditing**, IP whitelisting, firewalls, x.509 certificates.

**15. Explain the concept of embedded documents and when they should be used.**

* Embedded documents are nested documents within a parent document.
* Useful for data that is frequently read together (one-to-few relationships).
* Reduces the need for joins and increases read efficiency.

**16. What is the purpose of MongoDB’s $lookup stage in aggregation?**

* Performs **joins** between documents in different collections.
* Enables combining related data, similar to SQL joins.

**17. What are some common use cases for MongoDB?**

* Content management systems (CMS).
* Real-time analytics and dashboards.
* Product catalogs.
* IoT applications.
* Mobile and social apps.

**18. What are the advantages of using MongoDB for horizontal scaling?**

* Uses **sharding** to distribute data across multiple servers.
* Handles large datasets and high throughput.
* Maintains performance as data volume grows.

**19. How do MongoDB transactions differ from SQL transactions?**

* MongoDB transactions are newer (since v4.0).
* Traditionally designed for high availability and scalability, not strict ACID compliance.
* Transactions can be slower than in SQL for complex operations but offer similar guarantees.

**20. What are the main differences between capped collections and regular collections?**

* **Capped Collections**:
  + Fixed size.
  + Auto-overwrite oldest data (FIFO).
  + High performance for logging and streaming.
* **Regular Collections**: No size limit; documents persist unless explicitly removed.

**21. What is the purpose of the $match stage in MongoDB’s aggregation pipeline?**

* Filters documents based on specified conditions.
* Functions like SQL’s WHERE clause.
* Should be used early to reduce document processing.

**22. How can you secure access to a MongoDB database?**

* Use **authentication** (SCRAM, LDAP, etc.).
* Apply **RBAC** (role-based access control).
* Enable **TLS/SSL**.
* Use IP whitelisting and firewalls.
* Audit logs and enable encryption at rest.

**23. What is MongoDB’s WiredTiger storage engine, and why is it important?**

* Default storage engine in MongoDB.
* Provides document-level locking for high concurrency.
* Offers compression for better storage efficiency.
* Enables high performance and scalability.

Practical Questions

import pandas as pd

from pymongo import MongoClient

1. Load the Superstore dataset into MongoDB

client = MongoClient("mongodb://localhost:27017/") # Adjust if using MongoDB Atlas or other setup

db = client["SuperstoreDB"]

orders\_col = db["Orders"]

df = pd.read\_csv("Superstore.csv")

df.columns = df.columns.str.strip()

orders\_col.delete\_many({})

orders\_col.insert\_many(df.to\_dict(orient="records"))

2. Retrieve and print all documents

print("All documents:")

for doc in orders\_col.find():

print(doc)

3. Count total number of documents

total\_docs = orders\_col.count\_documents({})

print(f"\nTotal number of documents: {total\_docs}")

4. Fetch all orders from the "West" region

print("\nOrders from the 'West' region:")

for doc in orders\_col.find({"Region": "West"}):

print(doc)

5. Find orders where Sales is greater than 500

print("\nOrders with Sales > 500:")

for doc in orders\_col.find({"Sales": {"$gt": 500}}):

print(doc)

6. Top 3 orders with highest Profit

print("\nTop 3 orders with highest Profit:")

for doc in orders\_col.find().sort("Profit", -1).limit(3):

print(doc)

7. Update Ship Mode "First Class" to "Premium Class"

result = orders\_col.update\_many(

{"Ship Mode": "First Class"},

{"$set": {"Ship Mode": "Premium Class"}}

)

print(f"\nUpdated {result.modified\_count} documents from 'First Class' to 'Premium Class'.")

8. Delete orders where Sales < 50

delete\_result = orders\_col.delete\_many({"Sales": {"$lt": 50}})

print(f"\nDeleted {delete\_result.deleted\_count} documents with Sales < 50.")

9. Aggregation: Total Sales by Region

print("\nTotal Sales by Region:")

pipeline = [

{"$group": {"\_id": "$Region", "TotalSales": {"$sum": "$Sales"}}}

]

for doc in orders\_col.aggregate(pipeline):

print(doc)

10. Distinct values for Ship Mode

ship\_modes = orders\_col.distinct("Ship Mode")

print("\nDistinct Ship Modes:")

print(ship\_modes)

11. Count number of orders per Category

print("\nNumber of orders per Category:")

pipeline = [

{"$group": {"\_id": "$Category", "Count": {"$sum": 1}}}

]

for doc in orders\_col.aggregate(pipeline):

print(doc)