Mongodb

>Document -A document is the basic unit of data stored in a collection.
.Documents are similar to json objects and contain key-value pair.
.They can include nested structures and array.

>A group of documents is known as collection

>Distinct is used to return unique values across the fields

>ObjectId is a unique identifier automatically generated by the Mongodb for each document.
.It is a 12 byte identifier and ensures uniqueness across documents and includes timestamp.
.Timestamp: A 4-byte timestamp representing the ObjectId's creation time.
.Machine Identifier: A 3-byte identifier unique to the machine generating the ObjectId.
.Process Identifier: A 2-byte identifier unique to the process generating the ObjectId.
.Counter: A 3-byte incrementing counter, initialized to a random value

>Sharding is a method for distributing data across multiple servers or clusters to ensure horizontal scalability and to handle large datasets

>DRAWBACKS OF SHARDING
.Increased Complexity
.Increased Latency

>REPLICATION
.Replication in MongoDB is a key feature that enhances data availability, fault tolerance, and redundancy by duplicating data across multiple servers.
.It ensures that even if one server fails, the data remains accessible through other servers in the replica set.
.Replication in MongoDB is the process of synchronizing data across multiple MongoDB servers.
.A replica set is a group of MongoDB servers that maintain the same dataset.
.The primary server receives all write operations, while secondary servers replicate the data from the primary.

>WIREDTIGER
.WiredTiger is the default storage engine for mongodb.
.It provides better concurrency, compression and performance.

>BSON
.Bson stands to binary json.
.It is a binary encoded serialization format used by mongodb to represent document.
.Unlike json it has additional datatypes like date,binary and objectId.

>Capped Collection is a special type of collection with a fixed size and automatic overwriting of the old files when the size limit is reached.
db.createCollection('capped',{capped:true,size:1000,max:1})

>Lookup perfoms a leftouter join with another collection allowing you to combine multiple documents based on a specified field.
.Syntax:db.orders.aggregate([{$lookup: {from: "products",localField: "productId",foreignField: "_id",as: "productDetails"}}])
.from : the name of the collection we want to join with
.localField:The field in the documents of the input collection (the collection being aggregated) that you want to match.
.foreignField : The field in the documents of the from collection (the collection being j
oined) that will be matched against the localField.
.as:The name of the field to add to the documents in the input collection that will contain the results of the join.

>db.collection.validate checks for the integrity of the collection and its indexes ensuring that they are free of corruption and in a consistent state.

>Write Concern specifies the level of acknowledgment requested from MongoDB for write operations.
.It can be set to various levels, such as w: 1 for acknowledgment from the primary only, or w: "majority" for acknowledgment from the majority of replica set members.

>CAP theorem
The CAP Theorem, also known as Brewer's Theorem, is a fundamental principle in distributed systems that states that a distributed data store can only guarantee two out of the following three properties simultaneously:

>Consistency: Every read receives the most recent write, ensuring that all nodes in the system see the same data at the same time.
.This means that once a write is acknowledged, all subsequent reads will reflect that write.

>Availability: Every request (read or write) receives a response, even if some of the nodes are down.
.This means that the system remains operational and responsive, but it might not always return the most recent data.

>Partition Tolerance :The system continues to function even if network partitions (communication breakdowns between nodes) occur.
.This means the system can handle and recover from network failures without losing data or becoming completely unavailable.

>Normalization refers to the process of organising data in database to avoid data redundancy and increase data integrity.
.By references
.Data duplication

>Denormalization refers to the process of embedding related data into a single document rather than storing it in separate database or as references.

>INDEXES
.Indexes in mongodb are used to improve the query performance by allowing the database to quickly locate and access data and doesn't need to scan the entire document in the collection.
.Mongodb supports various types of indexes including single field index,compound index and geospatial index.

>Types of indexing
.Single Field Indexing: An index on a single field within a collection
db.users.createIndex({ email: 1 })

>COMPOUND INDEX
.An index on multiple fields within a collection
.db.users.createIndex({ firstName: 1, lastName: 1 })

.Multikey Index-An index given on field that contain array.
.Mongodb creates an index for each element in the array.
db.orders.createIndex({ items: 1 })

>DISADVANTAGES OF INDEXING
.Increased Disk Space Usage:
.Write Performance Impact:

>GridFS is a specification for storing and retrieving large files such as images, videos, and other binary data in MongoDB.
.It allows you to store files in chunks and provides efficient querying and retrieval.
fs.files: Stores metadata about the files (e.g., filename, upload date).
fs.chunks: Stores the file chunks. Each chunk is a piece of the file, and the chunks are stored in sequence.

>The $ne operator is used to match documents where the value of a field is not equal to a specified value.
>The $nin operator is used to match documents where the value of a field is not in a specified array of values.

>In MongoDB, the $project stage is used in the aggregation pipeline to reshape documents.
.It allows you to specify which fields to include or exclude, add new fields, or compute new values based on existing fields.
.It’s a powerful tool for transforming data into a format that suits your needs.

>$UPSERT
>The term upsert is a combination of update and insert.
.It refers to the operation where it either update an existing document or if no such document is there it

creates a new one.
.db.users.updateOne({ username: "jdoe" },{ $set: { email: "jdoe@example.com" } }{ upsert: true })

>acknowledged: true
.Indicates that the operation was acknowledged by the MongoDB server.
.This means the server received and processed the request successfully.
>insertedId
.It is crucial when an insert operation occurs it shows the id in which the insert operation occurs.
>matchedCount
.The number of documents that matched the query filter
>modifiedCount
.the number of documents modified in the update
>upsertedCount
.the number of documents that we upserted

>FACET
.

KaTeX parse error: Undefined control sequence: \[at position 279: …ruit.aggregate(\[{
facet:{countByVitamin:[{
KaTeX parse error: Expected '}', got 'EOF' at end of input: group:{_id:"
vitamin",count:{
KaTeX parse error: Expected 'EOF', got '}' at position 6: sum:1}}}\],average:\[…
group:{_id:"
KaTeX parse error: Expected '}', got 'EOF' at end of input: …amin",average:{
avg:"$price"}}}]}}])

>BULKWRITE
>The bulkWrite operation in MongoDB is used to perform multiple write operations in a single request.
.This can be more efficient as it reduces the number of round trips between the application and the database server.
test> db.fruit.bulkWrite([{updateOne:{filter:{name:"Mango"},update:{$set:{stock:2.5}}}},{deleteOne:{filter:{name:"Watermelon"}}}])

>HASHED INDEXING
.Hashed Indexing is a type of index that uses a hash of the indexed field's value for faster equality checks.
.It’s particularly useful when you need to ensure even distribution of data across shards in a sharded cluster or want to optimize the performance of equality queries.

>DATABASE PROFILING
.Profiling is a useful tool for monitoring and analyzing the performance of your queries and operations.
.The MongoDB profiler provides insights into the performance of database operations, allowing you to identify slow queries and performance bottlenecks.

>COVERED QUERY
.A covered query refers to a query that can be satisfied entirely using the indexes without needing to access the actual documents in the collection.
.This is a performance optimization technique that can significantly speed up query execution.

>all.all operator is used to match documents in an array that contains all the specified elements.

>JOURNALING
.Journaling is a feature designed to enhance data durability and crash recovery.
.It ensures that write operations are preserved in case of a server crash or power failure.
.Journaling is an essential part of MongoDB's Write Concern system and helps to maintain data integrity.

>$addToSet$.addToSet operator is used in update operations to add a value to an array field only if that value does not already exist in the array.
.This ensures that the array will not have duplicate values

>$EXPR$.expr operator is used to allow the use of aggregation expressions within the query language.
.This is useful when you need to compare fields within the same document or perform complex calculations in queries, which is not possible with the standard query operators.

>$SET
. The $set operator is used to update the value of a specific field in a document.
. If the field does not exist, $set will create it.

>$PUSH
.The $push operator is used to add an element to an array field in a document.
.If the array field does not exist, $push will create it and add the element.
.Push doesnt check for duplicates

>$ELEMMATCH$.elemMatch operator is used to match elements of an array that satisfy specific condition.
.This operator is particularly useful when you need to filter documents based on conditions applied to array elements.

>GEOSPATIAL INDEXING
.Geospatial indexing is a technique used in databases to efficiently query and manage spatial data, such as geographical locations, shapes, and boundaries.
.This type of indexing is crucial for applications that involve geographical or spatial queries, such as mapping services, location-based searches, and geographic information systems (GIS).

>CLUSTERED COLLECTION
.A clustered collection typically refers to a collection of documents or records in a database that are physically stored on disk in a sorted or ordered manner based on a specific field or index.

.This concept is most commonly associated with database systems that support clustering as a means to optimize query performance and data retrieval.

>OUT.out operator is used in aggregation pipelines to write the results of the aggregation directly to a collection.
.It either creates a new collection with the results or replaces an existing collection with the same name.

>$REDACT$.redact operator is used in aggregation pipelines to control the visibility of documents and their subfields based on certain conditions.

>$COND
.The $cond operator is a conditional operator that functions like an if-else statement in programming languages.
.It allows you to evaluate a condition and return different values based on whether that condition is true or false.
.This operator is particularly useful in aggregation pipelines, where it can be used to manipulate or transform data.

>PRETTY()
.pretty() method is used to format the output of a query in a more readable way.
.When you run a query, especially with find(), the results can be presented in a compact JSON format, which may not be very human-friendly.

>SCALING
.Scaling refers to the approach you take to improve database performance as the amount of data or the number of requests increases.
.There are two primary methods of scaling: horizontal scaling and vertical scaling.
1.Vertical Scaling:
.Vertical scaling involves increasing the capacity of a single server by adding more resources such as CPU, RAM, or disk space.
2.Horizontal Scaling
.Horizontal scaling involves adding more servers (or nodes) to distribute the load across multiple machines, also known as sharding in MongoDB.

>MONGODB UTILITY COMMANDS
.MongoDB utility commands that you can use to interact with the MongoDB database system.
1.Show collections
2.Switch to databases
3.Create collection
4.Find Documents

>$SIZE$.size operator is used to determine the number of elements in an array

>ATLAS FUNCTION
.Atlas Functions allow us to write and execute serverless functions within the MongoDB cloud

infrastructure.
.MongoDB Atlas is a cloud-hosted database service that provides built-in support for serverless computing, and its functions are executed in response to triggers or can be invoked directly via APIs.

>FEATURES OF MONGODB
.Document-Oriented
.Sharding and Replication
.High Performance
.Aggregation Framework
.ACID Transactions
.GridFS
.Cloud Integration
.Security

$PULL.pull operator is used to remove elements from an array that match a specified condition.

**TIME TO LIVE INDEX (TTL)
**.TTL (Time-To-Live) indexing in MongoDB is a feature that allows you to automatically delete documents from a collection after a certain period of time