Organized Practical Questions and Topics (Beginner to Advanced, Topic-Wise)

1. Basic MongoDB Operations (CRUD)

Beginner

- Create an empty collection (`db.createCollection()`).
- Insert a document into a collection (`insertOne`, `insertMany`).
- Find documents (`find`, `findOne`).
- Sort documents (`sort`).
- Limit results (`limit`).
- Skip documents (`skip`).
- Update a single document (`updateOne`).
- Update multiple documents (`updateMany`).
- Delete documents (`deleteOne`, `deleteMany`).
- Count documents (`countDocuments`).
- Practical: Show only name and salary of employees.
- Practical: Find records using less than operator (`$lt`).
- Practical: Sort a list using `find` (e.g., sort by salary).

Intermediate

- Update operators: `$set`, `$unset`, `$inc`.
- Practical: Increment the price of books by 50 rupees.
- Practical: Reduce the salary of all employees by 500.
- Practical: Increase the bonus of all employees by 1000.
- Practical: Decrement each book's rating by 1.
- Practical: Reduce the age of all by 5.
- Practical: Reduce the goals of all players by 25.
- Practical: Increase all marks by 10%.
- Practical: Reduce 2 marks from every student's score.
- Practical: Rename a field (`$rename`).
- Practical: Rename a collection (`renameCollection`).

- Practical: Remove a key from a document (`$unset`).

Advanced

- Upsert operations (`upsert: true`).
- Practical: Use `upsert` to update or insert a document if it doesn't exist.
- Practical: Difference between `save` vs `insert` vs `upsert`.
- Practical: Find one and update (`findOneAndUpdate`).
- Practical: Update documents in an array (e.g., update specific elements in an array field).

2. Querying with Logical and Comparison Operators

Beginner

- Use comparison operators: `$lt`, `$gt`, `$eq`, `$ne`.
- Use logical operators: `$and`, `$or`, `$not`, `$nor`.
- Practical: Find employees whose department is 2.
- Practical: Find fruits with price less than 2 and higher than 5.

Intermediate

- Use `$in` and `$nin` for querying.
- Practical: Difference between `$in` vs `$nin`.
- Practical: Find names starting with 'j' or ending with 'n', 'i', 'e', 'p', or vowels using `$regex`.
- Practical: Find names that end with 's' or 'j' (case-insensitive regex).
- Practical: Pattern matching using regex (e.g., names starting with a vowel).
- Practical: Find the first value from an array (`$slice`).

Advanced

- Use `$expr` for advanced comparisons.
- Practical: Find employees whose salary is above the average salary.
- Practical: Use `$elemMatch` to query arrays with multiple conditions.
- Practical: Query documents where a field exists (`$exists`).
- Practical: Conditionally add a field to documents that don't have it using `$exists`.
- Practical: Use `$all` to match all elements in an array.

- Practical: Bitwise query operators (`$bitsAnySet`).

- Practical: Find records in arrays (e.g., query specific array elements).

3. Aggregation Framework

Beginner

- Basic aggregation pipeline: `$match`, `$group`, `$sort`.

- Practical: Find the count of students in each semester (`$group`).

- Practical: Find the total salary and total employees in each department.

- Practical: Find the average score of class A.

- Practical: Find the average salary per department.

- Practical: Find the average score of students with score > 70.

- Practical: Find the average mark in class 10.

Intermediate

- Use `$group` with accumulators: `$sum`, `$avg`, `$max`, `$min`.

- Practical: Find the highest salary (`$max`).

- Practical: Find the second highest salary.

- Practical: Find the difference between the highest and lowest salary.

- Practical: Find the average CGPA where semester is between 5 and 10.

- Practical: Find the student with the highest mark.

- Practical: Find the count of employees who have the same salary.

- Practical: Find the average age of each team.

Advanced

- Use advanced aggregation stages: `$lookup`, `$unwind`, `$project`, `$out`, `$merge`.

- Practical: Perform a `$lookup` to join collections (e.g., join employee and department data).

- Practical: Use pipeline inside `$lookup` for complex joins.

- Practical: Unwind an array field (`$unwind`) and process it.

- Practical: Use `$facet` for multiple aggregation pipelines in a single query.

- Practical: Use `$setUnion` to combine results.

- Practical: Write results to a new collection using `$out`.

- Practical: Compare `$out` vs `$merge`.

- Practical: Practice aggregation workouts involving multiple stages (e.g., `$match`, `$group`, `$sort`).

- Practical: Find the second youngest person.

- Practical: Find the minimum aged whole persons.

4. Indexing

Beginner

- Create an index (`createIndex`).

- Practical: Create a basic index on a field (e.g., salary).

- Practical: List all indices (`getIndexes`).

Intermediate

- Types of indexes: single field, compound index, TTL index, geospatial index, hashed index.

- Practical: Create a compound index.

- Practical: Create a TTL index.

- Practical: Create a geospatial index.

- Practical: Create a hashed index.

- Practical: Understand default index on `_id`.

Advanced

- Understand covered queries.

- Practical: Write a covered query (query fully satisfied by an index).

- Practical: Drawbacks of indexing (e.g., storage overhead, write performance impact).

- Practical: Create and list indices for a collection.

- Practical: Compare compound index vs TTL index.

- Practical: Background working of indexing (how indexes are created and queried).

5. Array Operations

Beginner

- Add elements to an array: `$push`, `$addToSet`.

- Practical: Use `$addToSet` to add unique elements to an array.

- Practical: Compare `$addToSet` vs `$push`.

Intermediate

- Remove elements from an array: `$pop`, `$pull`, `$pullAll`.
- Practical: Compare `$pop` vs `$pull`.
- Practical: Find the first value from an array (`$slice`).
- Practical: Query arrays using `$elemMatch`.

Advanced

- Update specific array elements.
- Practical: Find and update elements in an array.
- Practical: Find employees with more than one phone number (array query).

6. Data Modeling

Beginner

- Understand embedded documents vs referenced documents.
- Practical: Create a collection with embedded documents.

Intermediate

- Practical: Embedding vs referencing (when to use which).
- Practical: Create a one-to-one or many-to-many relationship.

Advanced

- Data modeling best practices and anti-patterns.
- Practical: Design a schema for a given use case (e.g., employees and departments).
- Practical: Avoid common anti-patterns in MongoDB schema design.

7. Capped Collections

Beginner

- Create a capped collection (`db.createCollection("name", { capped: true, size: 100000 })`).
- Practical: Create a capped collection with size constraints.

Intermediate

- Understand `isCapped` to check if a collection is capped.

Advanced

- Practical: Use capped collections for logging or fixed-size data.
- Practical: Syntax issues with capped collection creation.

8. Transactions

Beginner

- Understand transactions in MongoDB.
- Practical: Write a basic transaction.

Intermediate

- Practical: Use transactions for multi-document updates.
- Practical: Understand ACID properties in MongoDB.

Advanced

- Practical: Implement a transaction for a complex operation (e.g., transferring funds).
- Practical: Understand isolation levels in MongoDB transactions.

9. Sharding

Beginner

- Understand sharding concepts and shard keys.
- Practical: Choose a shard key for a collection.

Intermediate

- Practical: Types of sharding (range-based, hash-based).
- Practical: Compare sharding vs replication.

Advanced

- Practical: Implement sharding in a MongoDB cluster.
- Practical: Understand `mongos` and its role in sharding.

10. Replication

Beginner

- Understand replica sets and their purpose.
- Practical: Minimum nodes for replication (3 nodes).

Intermediate

- Practical: Pros and cons of replication.
- Practical: Understand primary-secondary election in replica sets.

Advanced

- Practical: Set up a replica set.
- Practical: Handle failover scenarios in replication.

11. GridFS

Beginner

- Understand GridFS for storing large files.
- Practical: Store a file using GridFS.

Intermediate

- Practical: Retrieve a file from GridFS.

Advanced

- Practical: Use GridFS for a real-world use case (e.g., storing images).

12. Backup and Restore

Beginner

- Use `mongodump` and `mongorestore` for backup and restore.
- Practical: Perform a backup of a collection.

Intermediate

- Practical: Restore a collection using `mongorestore`.

- Practical: Compare `mongodump` vs `mongorestore`.

Advanced

- Practical: Automate backups using MongoDB utilities.

- Practical: Backup a sharded cluster.

13. Change Streams

Beginner

- Understand change streams in MongoDB.

- Practical: Set up a basic change stream to monitor collection changes.

Intermediate

- Practical: Use change streams to trigger actions on data changes.

Advanced

- Practical: Implement change streams in a real-time application.

14. Views

Beginner

- Create a view in MongoDB.

- Practical: Create a view using `createView`.

Intermediate

- Practical: Understand materialized views.

Advanced

- Practical: Use views for read-only aggregated data.

15. MongoDB Utilities

Beginner

- Use `mongoimport` and `mongoexport` for data import/export.
- Practical: Import data using `mongoimport`.

Intermediate

- Practical: Export data using `mongoexport`.
- Practical: Use `mongotop` to monitor MongoDB performance.

Advanced

- Practical: Use the database profiler to analyze query performance.
- Practical: Understand `allowDiskUse` for large aggregations.

16. Bulk Write Operations

Beginner

- Understand bulk write operations (`bulkWrite`).
- Practical: Perform a basic bulk write operation.

Intermediate

- Practical: Use `bulkWrite` for multiple insert/update/delete operations.
- Practical: Understand batch sizing in bulk writes.

Advanced

- Practical: Optimize bulk write operations for performance.

17. Miscellaneous

Beginner

- Understand BSON vs JSON.
- Practical: Advantages of BSON over JSON.
- Practical: Default port number of MongoDB (27017).
- Practical: Data type of `_id` (ObjectId, 12-byte hexadecimal).

- Practical: Components of `_id` (timestamp, machine ID, etc.).

- Practical: NoSQL full form (Not Only SQL).

Intermediate

- Understand CAP theorem and its application in MongoDB.

- Practical: Explain partition tolerance in MongoDB.

- Practical: MongoDB as a schema-less database.

- Practical: Use `$cond` for conditional logic in aggregations.

- Practical: Namespace in MongoDB (database.collection).

- Practical: Journaling in MongoDB (WiredTiger storage engine).
