1. Setup & Management:

- mongod: Start the MongoDB server.
- mongo: Connect to a running MongoDB server.
- mongos: Start a mongos server for sharding.
- mongodump: Backup MongoDB database.
- mongorestore: Restore MongoDB database from backup.
- mongoexport: Export a collection to JSON or CSV.
- mongoimport: Import content from JSON or CSV into a collection.

2. Basic Shell Commands:

- show dbs: List all databases.
- use <dbname>: Switch to database or create it if it doesn't exist.
- show collections: List all collections in the current database.
- db.help(): Display help for MongoDB commands in the shell.

3. CRUD Operations:

- db.collection.insert(document): Insert a document.
- db.collection.find(): Retrieve documents.
- db.collection.update(criteria, changes): Update documents.
- db.collection.delete(criteria): Delete documents.
- db.collection.save(document): Update an existing document or insert a new one.
- db.collection.replaceOne(filter, replacement): Replace a document.

4. Querying:

- db.collection.find({field: value}): Query by specific field value.
- db.collection.find({field: {$gt: value}}): Query for documents where field value is greater than a specified value.
- db.collection.find().sort({field: 1}): Sort query results in ascending order by field.
- db.collection.find().limit(n): Limit query results to n documents.

- `db.collection.find().skip(n)`: Skip the first n results of a query.
- `db.collection.findOne()`: Retrieve a single document.
- `db.collection.count()`: Count documents that match a query.

5. Advanced Queries:

- `db.collection.find({$or: [{field1: value1}, {field2: value2}]})`: Query using the OR condition.
- `db.collection.find({field: {$in: [value1, value2, ...]}})`: Query where field value is in the specified array.
- `db.collection.find({field: {$exists: true/false}})`: Query documents where a field does or doesn't exist.
- `db.collection.find({field: {$type: BSONType}})`: Query by BSON type.

6. Aggregation:

- `db.collection.aggregate()`: Process data and return computed results.
- `$group`: Group by some specified expression.
- `$sort`: Sort documents.
- `$limit`: Limit the number of documents.
- `$skip`: Skip specified number of documents.
- `$unwind`: Deconstructs an array field.

7. Indexing:

- `db.collection.createIndex({field: 1})`: Create an index on a field.
- `db.collection.getIndexes()`: List all indexes on a collection.
- `db.collection.dropIndex(indexName)`: Remove an index.
- `db.collection.reIndex()`: Rebuild indexes.

8. Administration:

- `db.runCommand({commandName: 1})`: Run a database command.
- `db.dropDatabase()`: Drop the current database.
- `db.collection.drop()`: Drop a collection.
- `db.collection.stats()`: Returns statistics about the collection.

9. Replication:

- `rs.initiate()`: Initialize a new replica set.
- `rs.add(host)`: Add a member to the replica set.
- `rs.status()`: View the status of the replica set.
- `rs.conf()`: View the replica set's configuration.
- `rs.remove(host)`: Remove a member from the replica set.

10. Sharding:

- `sh.status()`: Show sharding status.
- `sh.addShard(shardURL)`: Add a shard to a cluster.
- `sh.enableSharding(database)`: Enable sharding for a database.
- `sh.shardCollection(namespace, key)`: Enable sharding for a collection.

11. Monitoring & Diagnostics:

- `mongostat`: Provide statistics about MongoDB's state.
- `mongotop`: Monitor read/write activity on a per-collection level.
- `db.serverStatus()`: Retrieve server status.
- `db.currentOp()`: Show in-progress operations.
- `db.killOp(opid)`: Kill a specific operation.

12. User and Role Management:

- `db.createUser(userDocument)`: Create a new user.
- `db.dropUser(username)`: Delete a user.
- `db.updateUser(username, fields)`: Update user data.
- `db.createRole(roleDocument)`: Create a new role.
- `db.dropRole(roleName)`: Delete a role.

13. Security & Authentication:

- `--auth`: Start MongoDB with authentication.
- `db.auth(username, password)`: Authenticate a user.
- `--sslMode requireSSL`: Start MongoDB with SSL.

14. Storage:

- `--storageEngine wiredTiger`: Specify the WiredTiger storage engine.
- `db.collection.storageSize()`: Retrieve the storage size for a collection.
- `db.collection.totalSize()`: Retrieve the total size of a collection including all indexes.

15. Backup & Restore:

- `mongodump --db mydb`: Backup a specific database.
- `mongorestore --dir /path/to/backup`: Restore from a specific backup directory.

16. Import & Export:

- `mongoexport --collection mycol --out mycol.json`: Export a collection to JSON.
- `mongoimport --collection mycol --file mycol.json`: Import a collection from JSON.

17. Utilities:

- `db.fsyncLock()`: Lock a mongod instance for maintenance.
- `db.fsyncUnlock()`: Unlock a mongod instance.

18. Text Search:

- `db.collection.createIndex({field: "text"})`: Create a text index.
- `db.collection.find({$text: {$search: "query"}})`: Perform a text search.

19. GridFS:

- `mongofiles`: Utility for putting and getting files from MongoDB GridFS.
- `db.fs.files.find()`: Find files stored in GridFS.

20. Miscellaneous Commands & Concepts:

- `$elemMatch`: Query for array elements.
- `$upsert`: Insert a new document if no document matches the query criteria.
- `$set`: Set the value of a field.
- `$unset`: Remove a field.
- `$push`: Append a value to an array.
- `$pop`: Remove the first or last element of an array.
- `$addToSet`: Add a value to an array unless it already exists.
- `db.collection.explain()`: Explain a query.
- `db.collection.distinct(field)`: Find distinct values for a field.
- `db.collection.mapReduce(map, reduce, options)`: Perform a map-reduce operation.
- `ObjectId()`: Create a new ObjectId.
- `db.printReplicationInfo()`: Print replication info.
- `db.printShardingStatus()`: Print sharding info.

21. Advanced Features & Options:

- `db.runCommand({collMod: "collection", usePowerOf2Sizes: true})`: Modify collection options.
- `db.collection.isCapped()`: Check if a collection is capped.
- `--oplogSize`: Set the oplog size.
- `--replSet "rsname"`: Start mongod as a member of replica set "rsname".
- `db.setProfilingLevel(level)`: Set the database profiling level.
- `db.getProfilingStatus()`: Check the database profiling status.
- `db.collection.aggregate([{ $lookup: options }])`: Left outer join of two collections.
- `--bind_ip`: Bind MongoDB to listen to a specific IP address.
- `--fork`: Run MongoDB in the background.
- `--logpath`: Specify the log file path.

22. Troubleshooting & Help:

- `db.getLastError()`: Return the error message from the last operation.
- `db.printSlaveReplicationInfo()`: Print status of all replica set members.
- `mongo --help`: Display help about command line options.
- `db.getCollection("collection").help()`: Display help on collection methods.

23. Connection & Networking:

- `--port`: Specify the port number for MongoDB to listen on.
- `db.getMongo()`: Returns the current connection instance.
- `db.getMongo().setReadPref("secondary")`: Set read preference to secondary nodes.
- `db.getMongo().setSlaveOk()`: Allows queries on replica set secondaries.

24. Diagnostic Commands:

- `db.collection.validate()`: Validates the structure of data.
- `db.runCommand({whatsmyuri: 1})`: Returns the client's connection string.
- `db.hostInfo()`: Provides details about the system MongoDB is running on.
- `db.collection.getShardDistribution()`: Shows data distribution among shards.

25. Schema Design:

- `db.createCollection("name", options)`: Explicitly creates a collection with options.
- `db.collection.isCapped()`: Determines if a collection is a capped collection.
- `$jsonSchema`: Allows you to specify schema validation in the collection level.

26. Advanced Aggregation:

- `$lookup`: Joins documents from another collection.
- `$graphLookup`: Performs recursive search on a collection.
- `$facet`: Processes multiple aggregation pipelines within a single stage.
- `$bucket`: Categorizes incoming documents into specific ranges.
- `$bucketAuto`: Categorizes incoming documents into a specified number of equally sized ranges.

27. Concurrency & Transactions:

- `db.beginTransaction()`: Starts a multi-document transaction.
- `db.commitTransaction()`: Commits the active transaction.
- `db.abortTransaction()`: Aborts the active transaction.

28. Maintenance & Clean-up:

- `db.repairDatabase()`: Repairs the current database. Use with caution as this can be blocking and lengthy.
- `db.compactCollection()`: Compacts a collection, reducing storage use.
- `db.collection.reIndex()`: Rebuilds all indexes on a collection.

29. BSON Types & Handling:

- `ObjectId("string")`: Creates an ObjectId from the given string.
- `$binary`: Represents binary data.
- `$date`: Represents a specific date.
- `$timestamp`: Represents a BSON timestamp value.

30. Geospatial Queries:

- `db.collection.createIndex({location: "2d"})`: Creates a 2D geospatial index.
- `$near`: Returns geospatial objects closest to a point.
- `$geoWithin`: Returns objects within a certain geospatial shape.

31. TTL (Time-To-Live) Collections:

- `db.collection.createIndex({field: 1}, {expireAfterSeconds: seconds})`: Sets up a TTL index which automatically removes documents after a certain time.

32. Miscellaneous:

- `db.setLogLevel(level)`: Changes the log level.
- `$redact`: Reshapes each document in the stream by restricting the content for each document based on information stored in the documents themselves.
- `$merge`: Writes the results of the aggregation pipeline to a specified collection.
- `db.getReplicaSetStatus()`: Returns the status of the replica set from the POV of the server it's run on.
- `db.setFeatureCompatibilityVersion("version")`: Sets the feature compatibility version of MongoDB.