**1. Create Collections**

db.createCollection("users")

db.createCollection("orders", { capped: true, size: 5000, max: 1000 })

db.createCollection("logs", { validator: { level: { $in: ["info", "warn", "error"] } } })

db.createCollection("products", { autoIndexId: false })

db.createCollection("students", { timeseries: { timeField: "createdAt", metaField: "metadata" } })

**2. Drop Collections**

db.users.drop()

db.getCollection("orders").drop()

db["products"].drop()

db.collectionToDelete.drop()

if (db.logs.exists()) { db.logs.drop(); }

**3. List Collections**

db.getCollectionNames()

db.runCommand({ listCollections: 1 })

db.getCollectionInfos()

db.adminCommand({ listCollections: 1, nameOnly: true })

db.listCollections({ name: "users" }).toArray()

**4. Rename Collections**

db.oldCollection.renameCollection("newCollection")

db.tempData.renameCollection("permanentData", true) // dropTarget if exists

db.logs.renameCollection("archivedLogs")

db.getCollection("orders").renameCollection("completedOrders")

db.products.renameCollection("inventory")

**5. Create Indexes**

db.users.createIndex({ email: 1 })

db.orders.createIndex({ orderDate: -1 })

db.products.createIndex({ name: "text" })

db.logs.createIndex({ level: 1, timestamp: -1 })

db.students.createIndex({ age: 1 }, { unique: true })

**6. Drop Indexes**

db.users.dropIndex("email\_1")

db.orders.dropIndex({ orderDate: -1 })

db.products.dropIndexes()

db.logs.dropIndex({ level: 1, timestamp: -1 })

db.students.dropIndexes()

**7. View Indexes**

db.users.getIndexes()

db.orders.getIndexes()

db.products.getIndexes()

db.logs.getIndexes()

db.students.getIndexes()

**8. Modify Collection Options**

db.runCommand({ collMod: "orders", validator: { price: { $gte: 0 } } })

db.runCommand({ collMod: "users", validationLevel: "moderate" })

db.runCommand({ collMod: "logs", validationAction: "warn" })

db.runCommand({ collMod: "inventory", index: { keyPattern: { name: 1 }, expireAfterSeconds: 3600 } })

db.runCommand({ collMod: "employees", validationLevel: "strict" })

**9. Enable/Disable Sharding (Only in Clustered Mode)**

sh.enableSharding("myDatabase")

sh.shardCollection("myDatabase.users", { userId: 1 })

sh.disableBalancing("myDatabase.orders")

sh.isBalancerRunning()

sh.status()

**10. Database Operations**

db.dropDatabase()

db.runCommand({ renameCollection: "oldCollection", to: "newCollection" })

db.runCommand({ convertToCapped: "logs", size: 100000 })

db.runCommand({ createIndexes: "orders", indexes: [{ key: { orderId: 1 }, name: "orderIndex" }] })

db.runCommand({ dropDatabase: 1 })