MONGODB QUICK REFERENCE

CRUD, queries, aggregation, indexes, schema design

Connecting

Connection String

```
mongosh "mongodb://localhost:27017"
mongosh "mongodb://user:pass@host:27017/mydb"
mongosh "mongodb+srv://user:pass@cluster.mongodb.net/mydb"
```

Driver Connection (Node.js)

```
const { MongoClient } = require('mongodb');
const client = new MongoClient(uri);
await client.connect();
const db = client.db('mydb');
```

Databases & Collections

Database Operations

```
show dbs
use mydb
db.dropDatabase()
```

Collection Operations

```
db.createCollection("users")
show collections
db.users.drop()
```

Capped Collection

```
db.createCollection("logs", {
  capped: true, size: 10485760, max: 5000
})
```

CRUD Operations

Insert

```
db.users.insertOne({ name: "Alice", age: 30 })
db.users.insertMany([
  { name: "Bob", age: 25 },
  { name: "Carol", age: 28 }
])
```

Find

```
db.users.findOne({ name: "Alice" })
db.users.find({ age: { $gte: 25 } })
db.users.find({}, { name: 1, _id: 0 })
db.users.find().sort({ age: -1 }).limit(10)
```

Update

```
db.users.updateOne(
  { name: "Alice" },
  { $set: { age: 31, city: "Boston" } }
)
db.users.updateMany(
  { active: false },
  { $set: { archived: true } }
)
```

Delete

```
db.users.deleteOne({ name: "Alice" })
db.users.deleteMany({ active: false })
```

Replace & Upsert

```
db.users.replaceOne(
  { name: "Alice" },
  { name: "Alice", age: 32, city: "NYC" }
)
db.users.updateOne(
  { email: "a@ex.com" },
  { $set: { name: "Alice" } },
  { upsert: true }
)
```

Query Operators

Comparison

`$eq` / `$ne`	Equal / not equal
`$gt` / `$gte`	Greater than / greater or equal
`$lt` / `$lte`	Less than / less or equal
`$in` / `$nin`	In array / not in array

Logical

`$and`	All conditions must match
`$or`	At least one condition matches
`$not`	Negates a condition
`$exists`	Field exists (true/false)
`$regex`	Regular expression match

Update Operators

`$set`	Set field value
`$unset`	Remove field
`$inc`	Increment numeric value
`$push` / `$pull`	Add / remove array element
`$addToSet`	Add to array if not present
`$rename`	Rename a field

Aggregation

Pipeline Stages

`$match`	Filter documents (like WHERE)
`$group`	Group and aggregate
`$project`	Reshape documents (like SELECT)
`$sort`	Sort results
`$limit` / `$skip`	Pagination
`$lookup`	Left outer join with another collection
`$unwind`	Deconstruct array into documents

Aggregation Example

```
db.orders.aggregate([
  { $match: { status: "completed" } },
  { $group: {
      _id: "$customer_id",
      total: { $sum: "$amount" },
      count: { $sum: 1 }
  }},
  { $sort: { total: -1 } },
  { $limit: 10 }
])
```

Indexes

Create & Drop

```
db.users.createIndex({ email: 1 }, { unique: true })
db.users.createIndex({ name: 1, age: -1 })
db.users.createIndex({ location: "2dsphere" })
db.users.dropIndex("email_1")
```

Index Types

`Single field`	Index on one field ({ name: 1 })
`Compound`	Multiple fields ({ a: 1, b: -1 })
`Text`	Full-text search ({ field: 'text' })
`2dsphere`	Geospatial queries
`TTL`	Auto-expire documents after time

Index Info

```
db.users.getIndexes()
db.users.find({ name: "Alice" }).explain()
```

Schema Design

Embedding vs Referencing

`Embed`	1:1 or 1:few, data read together
`Reference`	1:many, data accessed independently
`Embed`	Sub-document rarely exceeds 16 MB
`Reference`	Many-to-many relationships

Schema Validation

```
db.createCollection("users", {
  validator: { $jsonSchema: {
    bsonType: "object",
    required: ["name", "email"],
    properties: {
      name: { bsonType: "string" },
      email: { bsonType: "string" }
}}})
```

Replication

Replica Set Concepts

`Primary`	Receives all writes
`Secondary`	Replicates from primary, can serve reads
`Arbiter`	Votes in elections, holds no data

Replica Set Commands

```
rs.initiate()
rs.add("mongo2:27017")
rs.addArb("mongo3:27017")
rs.status()
rs.conf()
```

Common Patterns

Transactions

```
const session = client.startSession();
session.startTransaction();
await db.collection("accounts").updateOne(
  { _id: 1 }, { $inc: { bal: -100 } }, { session });
await db.collection("accounts").updateOne(
  { _id: 2 }, { $inc: { bal: 100 } }, { session });
await session.commitTransaction();
```

Bulk Write

```
db.users.bulkWrite([
  { insertOne: { document: { name: "Dan" } } },
  { updateOne: {
      filter: { name: "Alice" },
      update: { $set: { age: 31 } }
  }},
  { deleteOne: { filter: { name: "old" } } }
])
```

Change Streams

```
const stream = db.collection("orders")
  .watch([{ $match: { "fullDocument.status": "new" } }]);
stream.on("change", (change) => {
  console.log(change.fullDocument);
});
```

mongosh Commands

Shell Helpers

`show dbs`	List databases
`show collections`	List collections in current db
`db.stats()`	Database statistics
`db.collection.stats()`	Collection statistics
`db.collection.countDocuments({})`	Count documents
`db.collection.distinct('field')`	Distinct values for a field

Export & Import

```
mongoexport --db=mydb --collection=users \
  --out=users.json
mongoimport --db=mydb --collection=users \
  --file=users.json
mongodump --db=mydb --out=/backup/
mongorestore --db=mydb /backup/mydb/
```