MongoDB Fundamentals – Self-Study Guide

---

1. Difference Between MySQL and MongoDB (NoSQL) Terminology

MySQL Terminology	MongoDB Terminology
database	database
table	collection
row	document
column	field
index	index
transaction	transaction
primary key	primary key (default is `_id`)

---

2. Database Operations

Operation	Command Example
Show all databases	`show dbs`
Show current database	`db`
Create/Switch database	`use <DB_name>`
Delete current database	`db.dropDatabase()`

> **Note:** You must be in the database you want to drop.

---

3. Collections Operations

Operation	Command Example

Operation	Command Example
Show all collections	`show collections`
Create a new collection	`db.createCollection("collection_name")`
Auto-create collection on insert	Insert directly into a non-existent collection

---

4. CRUD Operations

Create

Method	Description
`insert()`	Insert one or more documents
`insertOne()`	Insert a single document
`insertMany()`	Insert multiple documents

Example:

```
db.students.insertMany([
  {
    _id: 1,
    name: "ziad",
    grades: { arabic: 30, english: 40 },
    address: ["cairo", "fayiom"]
  },
  {
    _id: 2,
    name: "ahmed"
  }
])
```

> **Note**: Use `[]` for inserting multiple documents.

Explanation:

- `_id` : Document identifier (auto-generated if not specified)
- `grades` : A nested (embedded) document
- `address` : An array of strings

Read

SQL vs MongoDB Queries

SQL	MongoDB
`SELECT * FROM people`	`db.people.find()`
`SELECT id, user_id, status FROM people`	`db.people.find({}, { user_id: 1, status: 1 })`
`SELECT user_id, status FROM people`	`db.people.find({}, { user_id: 1, status: 1, _id: 0 })`
`SELECT * FROM people WHERE status = "A"`	`db.people.find({ status: "A" })`
`SELECT * FROM people WHERE status != "A"`	`db.people.find({ status: { $ne: "A" } })`
`SELECT * FROM people WHERE status = "A" AND age = 50`	`db.people.find({ status: "A", age: 50 })`
`SELECT * FROM people WHERE status = "A" OR age = 50`	`db.people.find({ $or: [{ status: "A" }, { age: 50 }] })`

Comparison Operators

SQL Operator	Description	MongoDB Equivalent
`=`	Equal to	`$eq`
`!=`	Not equal to	`$ne`
`>`	Greater than	`$gt`
`<`	Less than	`$lt`
`>=`	Greater than or equal to	`$gte`
`<=`	Less than or equal to	`$lte`

Examples on Comparison Operators

SQL	MongoDB
`SELECT * FROM people WHERE age > 25`	`db.people.find({ age: { $gt: 25 } })`

SQL	MongoDB
`SELECT * FROM people WHERE age < 25`	`db.people.find({ age: { $lt: 25 } })`
`SELECT * FROM people WHERE age > 25 AND age <= 50`	`db.people.find({ age: { $gt: 25, $lte: 50 } })`
`SELECT * FROM people WHERE age < 25 AND age >= 50`	`db.people.find({ age: { $lt: 25, $gte: 50 } })` (Logically contradictory)

Example 1:

```
db.people.find({}, { user_id: 1, status: 1, _id: 0 })
```

Explanation:

- `{}` = no filter → return all documents
- Second argument = projection (include/exclude fields)

Example 2:

```
db.people.find({ status: "A" }, { user_id: 1, status: 1, _id: 0 })
```

Other Read Commands

Purpose	Command Example
Find one document	`db.collection_name.findOne({ })`
Find multiple documents	`db.collection_name.find({ })`
Find with condition	`db.collection_name.find({ "field": "value" })`
Pretty print output	`db.collection_name.find().pretty()`

Sort Operation

MongoDB supports two sorting types:

- `1` = ascending (ASC)
- `-1` = descending (DESC)

SQL vs MongoDB Sorting Examples

SQL	MongoDB
`SELECT * FROM people WHERE status = "A"`	`db.people.find({ status: "A" })`
`ORDER BY user_id ASC`	`db.people.find({ status: "A" }).sort({ user_id: 1 })`
`ORDER BY user_id DESC`	`db.people.find({ status: "A" }).sort({ user_id: -1 })`

Count Operation

SQL	MongoDB
`SELECT COUNT(*) FROM people`	`db.people.count()` or `db.people.find().count()`
`SELECT COUNT(user_id) FROM people`	`db.people.count({ user_id: { $exists: true } })`
`SELECT COUNT(*) FROM people WHERE age > 30`	`db.people.count({ age: { $gt: 30 } })`

`$exists` Operator Example:

```
db.people.find({ user_id: { $exists: true } })
```

> In MySQL, `*` includes all columns (even nulls); in MongoDB, you can count conditionally or all.

Distinct Operation

Used to remove repeated data.

SQL	MongoDB
`SELECT DISTINCT (status) FROM people`	`db.people.distinct("status")`

Limit & Skip

SQL	MongoDB
`SELECT * FROM people LIMIT 1`	`db.people.findOne()` or `.find().limit(1)`
`SELECT * FROM people LIMIT 5 SKIP 10`	`db.people.find().limit(5).skip(10)`

Flexible Search – `LIKE` Operator Equivalent

SQL	MongoDB
`SELECT * FROM people WHERE user_id LIKE "%bc%"`	`db.people.find({ user_id: /bc/ })` or `{ $regex: /bc/ }`
`SELECT * FROM people WHERE user_id LIKE "bc%"`	`db.people.find({ user_id: /^bc/ })` or `{ $regex: /^bc/ }`

Update

Update Functions:

- `db.collection.updateOne()` → Updates one document even if more match.
- `db.collection.updateMany()` → Updates all matching documents.
- `db.collection.update()` → Updates one by default; add `{ multi: true }` to update many (legacy method).

SQL vs MongoDB Update Examples

SQL	MongoDB
`UPDATE people SET status = "C" WHERE age > 25`	`db.people.updateMany({ age: { $gt: 25 } }, { $set: { status: "C" } })`
`UPDATE people SET age = age + 3 WHERE status = "A"`	`db.people.updateMany({ status: "A" }, { $inc: { age: 3 } })`

Operation	MongoDB Command Example
Update one document	`db.collection.updateOne({ _id: 1 }, { $set: { name: "ziad" } })`

Operation	MongoDB Command Example
Update multiple documents	`db.collection.updateMany({ age: { $gt: 25 } }, { $set: { status: "B" } })`
Replace a document	`db.collection.replaceOne({ _id: 1 }, { name: "ziad" })`

> `$set` only updates specified fields. `replaceOne` replaces the entire document.

ALTER TABLE Equivalent

Add a Field

SQL	MongoDB
`ALTER TABLE people ADD join_date DATETIME`	`db.people.updateMany({}, { $set: { join_date: new Date() } })`

> `new Date()` will insert the current date.

Drop a Field

SQL	MongoDB
`ALTER TABLE people DROP COLUMN join_date`	`db.people.updateMany({}, { $unset: { "join_date": "" } })`

Delete

Operation	Command
Delete a collection	`db.collection_name.drop()`
Delete a database	`db.dropDatabase()`

> **Reminder:** You must be in the database you want to delete.

DELETE Statements

SQL	MongoDB
`DELETE FROM people WHERE status = "D"`	`db.people.deleteMany({ status: "D" })`

SQL	MongoDB
`DELETE FROM people`	`db.people.deleteMany({})`