

MongoDB Basics (Common For All Assignments)

Operation	Command	Meaning
Show databases	<code>show dbs</code>	View all databases
Create/Select DB	<code>use mydb</code>	Creates if not exists
Show collections	<code>show collections</code>	View all collections in DB
Drop DB	<code>db.dropDatabase()</code>	Deletes the selected DB
Create collection	<code>db.createCollection("stud")</code>	Explicit collection creation
Drop collection	<code>db.stud.drop()</code>	Deletes a collection

ASSIGNMENT 1 – CRUD + Queries

Create Collection + Insert

```
use collegeDB
```

```
db.student.insert({
  roll: 1,
  name: "Amit",
  branch: "IT",
  marks: 85
})
```

Batch Insert

```
db.student.insertMany([
  {roll: 2, name: "Pooja", branch: "CSE", marks: 92},
  {roll: 3, name: "Raj", branch: "IT", marks: 76}
])
```

Find Documents

```
db.student.find()  
db.student.find().pretty()
```

Find With Condition

```
db.student.find({branch: "IT"})  
db.student.find({marks: {$gt: 80}})  
db.student.find({marks: {$lt: 50}})  
db.student.find({marks: {$ne: 40}})
```

Logical Queries

```
db.student.find({marks: {$gt: 80}, branch: "IT"}) // AND  
db.student.find({$or: [{branch: "IT"}, {branch: "CSE"}]}) // OR
```

Update

```
db.student.update({roll: 1}, {$set: {marks: 95}})  
db.student.update({}, {$inc: {marks: 5}}, {multi: true}) // Increase  
marks for all
```

Replace Document

```
db.student.save({_id: ObjectId("..."), roll: 1, name: "Amit", branch:  
"EEE", marks: 88})
```

Delete

```
db.student.remove({roll: 3})  
db.student.remove({}, {justOne: true})  
db.student.remove({}) // delete all docs (truncate)
```

ASSIGNMENT 2 – Aggregation + Indexing

Aggregation Framework

```
db.student.aggregate([
  {$group: {_id: "$branch", total_students: {$sum: 1}}}
])
```

Meaning: Count how many students are in each branch.

Expressio n	Meaning
<code>\$sum:</code>	total count / sum
<code>\$avg:</code>	average
<code>\$min:</code>	minimum value
<code>\$max:</code>	maximum value

Example: Average marks branch-wise

```
db.student.aggregate([
  {$group: {_id: "$branch", avg_marks: {$avg: "$marks"}}}
])
```

Indexing

Create Index:

```
db.student.createIndex({name: 1})
```

Drop Index:

```
db.student.dropIndex({name: 1})
```

Check Indexes:

```
db.student.getIndexes()
```

Use: **Speeds up searching but uses extra storage**

MAP REDUCE IN MONGODB

Why Map-Reduce?

Map-Reduce is used for **large data processing** where aggregation may not be enough. It **splits work**, processes in parallel, and **combines results**.

General Structure

1) Map Function

Used to **emit** key-value pairs.

```
var mapFunction = function() {  
    emit(key, value);  
};
```

2) Reduce Function

Used to **merge values** having same key.

```
var reduceFunction = function(key, values) {  
    return reduced_value;  
};
```

3) Execute Map-Reduce

```
db.collection.mapReduce(  
    mapFunction,  
    reduceFunction,  
    { out: "output_collection_name" }  
)
```

Full Working Example (Use this in Exam)

Collection

```
use shopDB
```

```
db.orders.insertMany([
  {cust_id: "C1", price: 200},
  {cust_id: "C2", price: 150},
  {cust_id: "C1", price: 300},
  {cust_id: "C3", price: 100}
])
```

1) Map Function

```
var mapFunc = function() {
  emit(this.cust_id, this.price);
};
```

2) Reduce Function

```
var reduceFunc = function(key, values) {
  return Array.sum(values);
};
```

3) Call Map Reduce

```
db.orders.mapReduce(
  mapFunc,
  reduceFunc,
  { out: "total_spent_by_customers" }
)
```

4) View the Output

```
db.total_spent_by_customers.find().pretty()
```

Output

```
{ "_id" : "C1", "value" : 500 }  
{ "_id" : "C2", "value" : 150 }  
{ "_id" : "C3", "value" : 100 }
```

This means:

- Customer C1 spent **₹500**
- Customer C2 spent **₹150**
- Customer C3 spent **₹100**