Here's a list of **MongoDB commands** to help you complete **Week 12 & 13 tasks**:

✅ 1. Create a Database

use myDatabase

This creates or switches to the database named myDatabase.

✅ 2. Create a Collection

db.createCollection("students")

Creates a collection named students.

✅ 3. Insert Documents

```
db.students.insertOne({
  name: "Alice",
  age: 22,
  course: "Computer Science"
})
db.students.insertMany([
  { name: "Bob", age: 23, course: "Electronics" },
  { name: "Charlie", age: 21, course: "Mechanical" }
])
```

✅ 4. Read Data (Basic Queries)

```
db.students.find()                  // View all documents
db.students.find({ name: "Alice" })   // Find by condition
```

✅ 5. Update Documents

```
db.students.updateOne(
  { name: "Alice" },
  { $set: { age: 23 } }
)
```

✅ 6. Delete Documents

db.students.deleteOne({ name: "Charlie" })

or

db.students.remove({ name: “Charlie” }) (remove is deprecated in latest versions)

✅ 7. Explore Data Types in MongoDB

Data Type	Example Usage
String	"Alice"
Number (int/double)	22 or 22.5
Boolean	true, false
Array	["Math", "Science"]
Object	{ city: "Hyd", pin: 500032 }
Null	null
Date	new Date("2025-05-18")

Example: (execute this)

```
db.students.insertOne({
  name: "David",
  age: 24,
  isActive: true,
  subjects: ["Math", "Science"],
  address: { city: "Hyderabad", pin: 500032 },
  joined: new Date()
})
```

(Extra MongoDB Query Language)

✅ 8. Count

db.students.count()

✅ **9. Sort (ascending order of name)**

db.students.find().sort({name:1})

✅ **10. skip (skip the first 2 documents)**

db.students.find().skip(2)

Arrays

db.createCollection("food")

db.food.insert({_id:1,fruits:['banana','apple','cherry']})

db.food.insertOne({_id:2,fruits:['orange','mango']})

db.food.insertOne({_id:3,fruits:['orange','strawberry','grapes']})

db.food.insertOne({_id:4,fruits:['banana','strawberry','grapes']})

db.food.insertOne({_id:5,fruits:['strawberry','grapes']})

find in arrays

db.food.find({fruits:['banana','apple','cherry']})

db.food.find({'fruits.1':'grapes'})

(To find those documents from the “food” collection which have the “fruits” array having “grapes” in the first index position. The index position begins at 0.)

Aggregrate Functions

db.createCollection("Customers")

db.Customers.insertMany([{CustID:"C123",AccBal:500,AccType:"S"},

{CustID:"C123",AccBal:900,AccType:"S"},

{CustID:"C111",AccBal:1200,AccType:"S"},

{CustID:"C123",AccBal:1500,AccType:"C"}])

(first filter on “AccType:S” and then group it on “CustID” and then compute the sum of “AccBal”)

db.Customers.aggregate({$match: {AccType: "S"}}, {$group: {_id: "$CustID",TotAccBal: {$sum: "$AccBal"}}});

MapReduce Function

```
var map = function() { emit (this.CustID, this.AccBal);}
var reduce = function(key, values){ return Array.sum(values);}
db.Customers.mapReduce(map, reduce,{out: "Customer_Totals", query:{AccType:"S"}});
```