Database & Collection

1. Create a database named company.
   ANSWER: use company
2. Create a collection named employee.
   ANSWER: db.createCollection("employee")
3. Show all collections in the company database.
   ANSWER: show collections

Insert Operations

4. Insert one employee record with name "Raj", job "Manager", and salary 50000.
   ANSWER: db.employee.insertOne({ename: "Raj", job: "Manager", salary: 50000})
5. Insert multiple employees: "Amit" (Developer, 40000) and "Sita" (HR, 35000).
   ANSWER: db.employee.insertMany([

   {ename: "Amit", job: "Developer", salary: 40000},

   {ename: "Sita", job: "HR", salary: 35000}

   ])
6. Perform an unordered insert of two employees.
   ANSWER: db.employee.insertMany(

   [{ename: "John", job: "Tester"}, {ename: "Meena", job: "Developer"}],

   {ordered: false})

Read Operations

7. Display all employee documents.
   ANSWER: db.employee.find()
8. Find the first employee document.
   ANSWER: db.employee.findOne()

9. Display only ename and job of all employees.
   ANSWER: db.employee.find({}, {ename:1, job:1})
10. Find all employees with the job "Developer".
    ANSWER: db.employee.find({job: "Developer"})
11. Find employees with salaries greater than 40000.
    ANSWER: db.employee.find({salary: {$gt: 40000}})
12. Find employees with salaries between 30000 and 45000.
    ANSWER: db.employee.find({salary: {$gte: 30000, $lte: 45000}})
13. Find employees whose job is either "Manager" or "HR".
    ANSWER: db.employee.find({job: {$in: ["Manager", "HR"]}})
14. Find employees who are not Managers.
    ANSWER: db.employee.find({job: {$ne: "Manager"}})
15. Find employees with salaries not equal to 40000.
    ANSWER: db.employee.find({salary: {$ne: 40000}})

Projection, Sort, Limit, Skip

16. Display only employee names, hide _id.
    ANSWER: db.employee.find({}, {ename:1, _id:0})
17. Display employees sorted by salary (ascending).
    ANSWER: db.employee.find().sort({salary: 1})
18. Display employees sorted by name (descending).
    ANSWER: db.employee.find().sort({ename: -1})
19. Display top 2 employees (limit).
    ANSWER: db.employee.find().limit(2)
20. Skip the first 2 employees and show the rest.
    ANSWER: db.employee.find().skip(2)
21. Count the number of employees.
    ANSWER: db.employee.find().count()
22. Find distinct job roles in employees.
    ANSWER: db.employee.distinct("job")

Update Operations

23. Update salary of Raj to 60000.
    ANSWER: db.employee.updateOne({ename: "Raj"}, {$set: {salary: 60000}})
24. Update all Developers' salary to 45000.
    ANSWER: db.employee.updateMany({job: "Developer"}, {$set: {salary: 45000}})
25. Remove the job field from Amit's record.
    ANSWER: db.employee.updateOne({ename: "Amit"}, {$unset: {job: 1}})
26. Rename salary field to sal.
    ANSWER: db.employee.updateMany({}, {$rename: {salary: "sal"}})
27. Add a skill array with "MongoDB" to Raj.
    ANSWER: db.employee.updateOne({ename: "Raj"}, {$push: {skills: "MongoDB"}})
28. Remove the last element from Raj's skills array.
    ANSWER: db.employee.updateOne({ename: "Raj"}, {$pop: {skills: 1}})
29. Replace the entire document of Sita with {ename:"Sita", dept:"HR"}.
    ANSWER: db.employee.replaceOne({ename:"Sita"}, {ename:"Sita", dept:"HR"})

Delete Operations

30. Delete one employee whose job is "Tester".
    ANSWER: db.employee.deleteOne({job: "Tester"})
31. Delete all employees with the job "HR".
    ANSWER: db.employee.deleteMany({job: "HR"})
32. Delete all documents from collection.
    ANSWER: db.employee.deleteMany({})