LAB-9 :

Part – A :

1. Retrieve/Display every document of Deposit collection.

➔ use BANK_INFO

➔ db.createCollection("Deposit")

➔ db.Deposit.insertMany([

{"ACTNO": 101,"CNAME": "ANIL","BNAME": "VRCE","AMOUNT": 1000,"ADATE": "1995-03-01"},

{"ACTNO": 102,"CNAME": "SUNIL",BNAME": "AJNI","AMOUNT": 5000,"ADATE": "1996-01-04"},

{"ACTNO": 103,"CNAME": "MEHUL","BNAME": "KAROLBAGH","AMOUNT": 3500,"ADATE": "1995-11-17"},

{"ACTNO": 104, "CNAME": "MADHURI","BNAME": "CHANDI","AMOUNT": 1200,"ADATE": "1995-12-17"},

{"ACTNO": 105, "CNAME": "PRMOD","BNAME": "M.G. ROAD","AMOUNT": 3000, "ADATE": "1996-03-27"},

{"ACTNO": 106, "CNAME": "SANDIP","BNAME": "ANDHERI","AMOUNT": 2000,"ADATE": "1996-03-31"},

{"ACTNO": 107,"CNAME": "SHIVANI","BNAME": "VIRAR","AMOUNT": 1000,"ADATE": "1995-09-05"},

{"ACTNO": 108,"CNAME": "KRANTI","BNAME": "NEHRU PLACE","AMOUNT": 5000,"ADATE": "1995-07-02"}

])

➔ db.Deposit.find()

2. Display only one document of Deposit collection. (Use: findOne())

➔ db.Deposit.findOne()

3. Insert following document into Deposit collection. (Use: insertOne())

109 KIRTI VIRAR 3000 3-5-97

➔ db.Deposit.insertOne({ ACTNO : 109 , CNAME : 'KIRTI' , BNAME : 'VIRAR' , AMOUNT : 3000 , ADATE : '1997-05-03' })

4. Insert following documents into Deposit collection. (Use: insertMany())

110 MITALI ANDHERI 4500 4-9-95

111 RAJIV NEHRU PLACE 7000 2-10-98

➔ db.Deposit.insertOne([{ ACTNO : 110 , CNAME : 'MITALI' , BNAME : 'ANDHERI' , AMOUNT : 4500 , ADATE : '1995-09-04' } , { ACTNO : 111 , CNAME : 'RAJIV' , BNAME : 'NEHRU PLACE' , AMOUNT : 7000 , ADATE : '1998-10-02' }])

5. Display all the documents of 'VIRAR' branch from Deposit collection.

➔ db.Deposit.find({ BNAME : 'VIRAR' })

6. Display all the documents of Deposit collection whose amount is between 3000 and 5000.

➔ db.Deposit.find({ AMOUNT : {$gt : 3000 , $lt : 5000} })

7. Display all the documents of Deposit collection whose amount is greater than 2000 and branch is VIRAR.

➔ db.Deposit.find({ AMOUNT : {$gt : 2000} , BNAME : 'VIRAR' })

8. Display all the documents with CNAME, BNAME and AMOUNT fields from Deposit collection.

➔ db.Deposit.find({} , { _id : 0 , CNAME : 1 , BNAME : 1 , AMOUNT : 1 }).forEach(printjson)

9. Display all the documents of Deposit collection on ascending order by CNAME.

➔ db.Deposit.find().sort({ CNAME : 1 })

10. Display all the documents of Deposit collection on descending order by BNAME.

➔ db.Deposit.find().sort({ BNAME : -1 })

11. Display all the documents of Deposit collection on ascending order by ACTNO and descending order by AMOUNT.

➔ db.Deposit.find().sort({ ACTNO : 1 , AMOUNT : -1 })

12. Display only two documents of Deposit collection.

➔ db.Deposit.find().limit(2)

13. Display 3rd document of Deposit collection.

➔ db.Depsoit.find().skip(2).limit(1)

14. Display 6th and 7th documents of Deposit collection.

➔ db.Depsoit.find().skip(5).limit(2)

15. Display the count of documents in Deposit collection.

➔ db.Deposit.countDocuments()

Part- B :

1. Insert following documents into “Student” collection. (Use: insertMany())

➔ use Darshan

➔ db.Student.insertMany([
{ "_id": 1, "name": "John", "age": 30, "city": "New York", "isActive": true },
{ "_id": 2, "name": "Jane", "age": 25, "city": "Los Angeles", "isActive": false },
{ "_id": 3, "name": "Tom", "age": 35, "city": "Chicago", "isActive": true },
{ "_id": 4, "name": "Lucy", "age": 28, "city": "San Francisco", "isActive": true },
{ "_id": 5, "name": "David", "age": 40, "city": "Miami", "isActive": false },
{ "_id": 6, "name": "Eva", "age": 23, "city": "Boston", "isActive": true },
{ "_id": 7, "name": "Nick", "age": 38, "city": "Seattle", "isActive": false },
{ "_id": 8, "name": "Sophia", "age": 27, "city": "New York", "isActive": true },
{ "_id": 9, "name": "Liam", "age": 32, "city": "Los Angeles", "isActive": false },
{ "_id": 10, "name": "Olivia", "age": 29, "city": "San Diego", "isActive": true }
])

2. Display all documents of "Student" collection.

➔ db.Student.find()

3. Display all documents of "Student" collection whose age is 30.

➔ db.Student.find({ age : {$eq : 30} })

4. Display all documents of "Student" collection whose age is greater than 25.

➔ db.Student.find({ age : {$gt : 25} })

5. Display all documents of "Student" collection whose name is "John" and age is 30.

➔ db.Student.find({ name : ‘John’ , age : 30 })

6. Display all documents of "Student" collection whose age is not equal to 25.

➔ db.Student.find({ age : {$ne : 25} })

7. Display all documents of "Student" collection whose age is equal to 25 or 30 or 35. (using $or as well as using $in).

➔ db.Student.find({ $or : [{age : 25} , {age:30} , {age:35}] })
➔ db.Student.find({ age : { $in : [25 , 30 , 35] } })

8. Display all documents of "Student" collection whose name is "John" or age is 30.

➔ db.Student.find({ $or : [{name : 'John'} , {age : 30}] })

9. Display all documents of "Student" collection whose name is "John" and city is New York.

➔ db.Student.find({ $and : [{name : 'John'} , {city : 'New York'}] })

10. Display name and age of students from "Student" collection whose name is "John" and city is New York.

➔ db.Student.find({ $and : [{name : 'John'} , {city : 'New York'}] } ,
{ _id : 0 , name : 1 , age : 1 })

Part – C :

1. Display name of students from "Student" collection whose age is between to 25 and 35 and sort output by age in ascending order.
   ➔ db.Student.find({ age: { $gte : 25 , $lte : 35 } } ,
   { _id : 0 , name : 1 }).sort({ age : 1 }).forEach(printjson)

2. Display all documents of "Student" collection and sort all the documents by name in ascending order and then by age in descending.

➔ db.Student.find().sort({ name : 1 , age : -1 })

3. Display first five documents of "Student" collection.

➔ db.Student.find().limit(5)

4. Display fourth and fifth documents of "Student" collection.

➔ db.Student.find().skip(3).limit(2)

5. Display the name of oldest student from "Student" collection.

➔ db.Student.find({} , { _id : 0 , name : 1 }).sort({ age : -1 }).limit(1).forEach(printjson)

6. Display all documents of "Student" collection in such a way that skip the first 2 documents and return the rest documents.

➔ db.Student.find().skip(2)