Perform following queries using find, limit, skip and sort method.

Create and Select Database Named: “BANK\_INFO”

“use BANK\_INFO”

From the above given collection perform the following queries using find, limit, skip and sort method:

Part – A

1. Retrieve/Display every document of Deposit collection.

db.Deposit.find()

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

db.Deposit.findOne()

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

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

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

db.Deposit.insertMany([

{ACTNO:110, CNAME:'MITALI', BNAME:'ANDHERI', AMOUNT:4500, ADATE:'1995-09-04'},

{ACTNO:111, CNAME:'RAJIV', BNAME:'NEHRU PALACE', AMOUNT:7000, ADATE:'1998-10-02'}

])

1. Display all the documents of ‘VIRAR’ branch from Deposit collection.

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

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

db.Deposit.find({$and:[{AMOUNT:{$gte:3000}}, {AMOUNT:{$lte:5000}}]})

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

db.Deposit.find({$and:[{AMOUNT:{$gt:2000}}, {BNAME:{$eq:'VIRAR'}}]})

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

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

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

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

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

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

1. 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})

1. Display only two documents of Deposit collection.

db.Deposit.find().limit(2)

1. Display 3rd document of Deposit collection.

db.Deposit.find().skip(2).limit(1)

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

db.Deposit.find().skip(5).limit(2)

1. Display the count of documents in Deposit collection.

db.Deposit.countDocuments()

Part- B

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

* 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: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({$and:[{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({$and:[{age:{$gte:25}}, {age:{$lte:35}}]}, {\_id:0, name:1}).sort({age:1})

1. 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})

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

db.Student.find().limit(5)

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

db.Student.find().skip(3).limit(2)

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

db.Student.aggregate([

{ $sort: { age: -1 } },

{ $limit: 1 }

])

1. 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)