Practical 04 :- MongoDB CRUD Operations

1) Create Database :- Let us create a database name userdb

```
> use userdb
< switched to db userdb
```

2) Creating a New Collection

```
> db.createCollection("users31")
< { ok: 1 }
```

3) Create Operation : There are 2 ways to create new documents to a collection in MongoDB:

1. insertOne():

Syntax:- db.collectionName.insertOne()

```
> db.users31.insertOne({Name:"Ankita",Age:21,Address:"Thane"})
< {
    acknowledged: true,
    insertedId: ObjectId("64d48bded302cc29d0b4f2e9")
  }
> db.users31.find()
< {
    _id: ObjectId("64d47f63d302cc29d0b4f2e6"),
    Name: 'Ankita',
    Age: 21,
    Address: 'Thane'
  }
```

2. insertMany()

Syntax:

db.collectionName.insertMany();

```
> db.users31.insertMany([{Name:"Ankita",Age:21,Address:"Thane"},{Name:"Aashu",Age:47,Address:"Vikhroli"},{Name:"Ram",Age:46,Address:"Ghatkopar"}])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("64d47f63d302cc29d0b4f2e6"),
      '1': ObjectId("64d47f63d302cc29d0b4f2e7"),
      '2': ObjectId("64d47f63d302cc29d0b4f2e8")
    }
  }
```

4) Read Operations: In MongoDB, read operations are used to retrieve data from the database.

1. find()

Syntax:- db.collectionName.find(query, projection)

```
> db.users31.find()
< {
    _id: ObjectId("64d47f63d302cc29d0b4f2e6"),
    Name: 'Ankita',
    Age: 21,
    Address: 'Thane'
  }
  {
    _id: ObjectId("64d47f63d302cc29d0b4f2e7"),
    Name: 'Aashu',
    Age: 47,
    Address: 'Vikhroli'
  }
  {
    _id: ObjectId("64d47f63d302cc29d0b4f2e8"),
    Name: 'Ram',
    Age: 46,
    Address: 'Ghatkopar'
  }
```

```
> db.users31.find({Age:{$gt:29}},{Name:1,Age:1})
< {
    _id: ObjectId("64d47f63d302cc29d0b4f2e7"),
    Name: 'Aashu',
    Age: 47
  }
  {
    _id: ObjectId("64d47f63d302cc29d0b4f2e8"),
    Name: 'Ram',
    Age: 46
  }
```

```
> db.users31.find({Age:{$gt:29}},{Name:1,Age:1,Address:1})
< {
    _id: ObjectId("64d47f63d302cc29d0b4f2e7"),
    Name: 'Aashu',
    Age: 47,
    Address: 'Vikhroli'
  }
  {
    _id: ObjectId("64d47f63d302cc29d0b4f2e8"),
    Name: 'Ram',
    Age: 46,
    Address: 'Ghatkopar'
  }
```

```
> db.users31.find({Age:{$lt:29}},{Name:1,Age:1,Address:1})
< {
    _id: ObjectId("64d47f63d302cc29d0b4f2e6"),
    Name: 'Ankita',
    Age: 21,
    Address: 'Thane'
  }
```

2. findOne()

The findOne() method returns a single document object, or null if no document is found. You can pass a query object to this method to filter the results.

Syntax:

db.collectionName.findOne()

```
> db.part_2.findOne({ name: "Jim" })
< {
    _id: ObjectId("64d5ad88d47c56bff07a5800"),
    name: 'Jim',
    age: 29,
    status: 'active'
  }
```

5) Update Operations

In MongoDB, the "update" operation is used to modify existing documents in a collection.

Methods:

There are several ways to perform an update operation, including the following:

1. updateOne()

The updateOne() method is used to update a single document that matches a specified filter.

Syntax:

db.collectionName.updateOne(filter, update, options)

```
> db.part_2.updateOne({ name: "Angela" }, { $set: { email: "angela@gmail.com" }})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 0,
    upsertedCount: 0
  }
> db.part_2.find()
< {
    _id: ObjectId("64d59fb9d47c56bff07a57e2"),
    name: 'Angela',
    age: 27,
    email: 'angela@gmail.com'
  }
  {
    _id: ObjectId("64d59fb9d47c56bff07a57e3"),
    name: 'Dwight',
    age: 30
  }
  {
    _id: ObjectId("64d59fb9d47c56bff07a57e4"),
    name: 'Jim',
    age: 29
  }
```

2. updateMany

The updateMany() method is used to update multiple documents that match a specified filter.

Syntax:

db.collectionName.updateMany(filter, update, options)

```
> db.part_2.updateMany({ age: { $lt: 30 } }, { $set: { status: "active" }})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 2,
    modifiedCount: 2,
    upsertedCount: 0
  }
> db.part_2.find()
< {
    _id: ObjectId("64d59fb9d47c56bff07a57e3"),
    name: 'Dwight',
    age: 30
  }
  {
    _id: ObjectId("64d5ad88d47c56bff07a57fe"),
    name: 'Angela',
    age: 27,
    status: 'active'
  }
  {
    _id: ObjectId("64d5ad88d47c56bff07a57ff"),
    name: 'Dwight',
    age: 30
  }
  {
    _id: ObjectId("64d5ad88d47c56bff07a5800"),
    name: 'Jim',
    age: 29,
    status: 'active'
  }
```

6) Delete Operations

In MongoDB, the "delete" operation is used to remove documents from a collection.

There are several ways to perform a delete operation, including the following:

1. deleteOne()

The deleteOne() method is used to remove a single document that matches a specified filter.

Syntax:

db.collectionName.deleteOne(filter, options)

```
> db.part_2.deleteOne({name:"Angela"})
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.part_2.find()
< {
    _id: ObjectId("64d59fb9d47c56bff07a57e3"),
    name: 'Dwight',
    age: 30
  }
  {
    _id: ObjectId("64d59fb9d47c56bff07a57e4"),
    name: 'Jim',
    age: 29
  }
```

2. deleteMany()

The deleteMany() method is used to remove multiple documents that match a specified filter.

Syntax:

db.collectionName.deleteMany(filter, options)

```
> db.part_2.find()
< {
    _id: ObjectId("64d59fb9d47c56bff07a57e3"),
    name: 'Dwight',
    age: 30
  }
  {
    _id: ObjectId("64d59fb9d47c56bff07a57e4"),
    name: 'Jim',
    age: 29
  }
> db.part_2.deleteMany({age:{$lt:30}})
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.part_2.find()
< {
    _id: ObjectId("64d59fb9d47c56bff07a57e3"),
    name: 'Dwight',
    age: 30
  }
```

3. drop()

The drop() method is used to remove an entire collection.

Syntax:

db.collectionName.drop()

```
> db.sales.drop()
< true
```

Practical 5: MongoDB aggregation operations

The operations on each stage can be one of the following:

- $project – select fields for the output documents.
- $match – select documents to be processed.
- $limit – limit the number of documents to be passed to the next stage.
- $skip – skip a specified number of documents.
- $sort – sort documents.
- $group – group documents by a specified key.

The following shows the syntax for defining an aggregation pipeline:

db.collection.aggregate([{ $match:...},{$group:...},{$sort:...}]);

In this syntax:

- First, call the aggregate() method on the collection.
- Second, pass an array of documents, where each document describes a stage in the pipeline.

MongoDB aggregation example

First, switch to the coffeeshop database that stores the coffee sales:

– use coffeeshop

Second, insert documents into the sales collection:

– db.sales.insertMany([

{ "_id" : 1, "item" : "Americanos", "price" : 5, "size": "Short", "quantity" : 22, "date" : ISODate("2022-01-15T08:00:00Z") },

{ "_id" : 2, "item" : "Cappuccino", "price" : 6, "size": "Short","quantity" : 12, "date" : ISODate("2022-01-16T09:00:00Z") },

{ "_id" : 3, "item" : "Lattes", "price" : 15, "size": "Grande","quantity" : 25, "date" : ISODate("2022-01-16T09:05:00Z") },

{ "_id" : 4, "item" : "Mochas", "price" : 25,"size": "Tall", "quantity" : 11, "date" : ISODate("2022-02-17T08:00:00Z") },

{ "_id" : 5, "item" : "Americanos", "price" : 10, "size": "Grande","quantity" : 12, "date" : ISODate("2022-02-18T21:06:00Z") },

{ "_id" : 6, "item" : "Cappuccino", "price" : 7, "size": "Tall","quantity" : 20, "date" : ISODate("2022-02-20T10:07:00Z") },

{ "_id" : 7, "item" : "Lattes", "price" : 25,"size": "Tall", "quantity" : 30, "date" : ISODate("2022-02-21T10:08:00Z") },

{ "_id" : 8, "item" : "Americanos", "price" : 10, "size": "Grande","quantity" : 21, "date" : ISODate("2022-02-22T14:09:00Z") },

{ "_id" : 9, "item" : "Cappuccino", "price" : 10, "size": "Grande","quantity" : 17, "date" : ISODate("2022-02-23T14:09:00Z") },

{ "_id" : 10, "item" : "Americanos", "price" : 8, "size": "Tall","quantity" : 15, "date" : ISODate("2022-02-25T14:09:00Z")}

]);

Third, use an aggregation pipeline to filter the sales by the Americanos, calculate the sum of quantity grouped by sizes, and sort the result document by the total quantity in descending order.

```
db.sales.aggregate([
	{
		$match: { item: "Americanos" }
	},
	{
		$group: {
			_id: "$size",
			totalQty: {$sum: "$quantity"}
		}
	},
	{
		$sort: { totalQty : -1}
```

```
        }
]);
[
  { _id: 'Grande', totalQty: 33 },
  { _id: 'Short', totalQty: 22 },
  { _id: 'Tall', totalQty: 15 }
]
```

```
> db.sales.aggregate([{$match:{item:"Americanos"}},{$group:{_id:"$size",totalQty:{$sum:"$quantity"}}},{$sort:{totalQty:-1}}]);
< {
    _id: 'short',
    totalQty: 110
  }
```

```
> db.sales.aggregate([{$match:{item:"Americanos"}},{$group:{_id:"$size",totalQty:{$sum:"$quantity"}}},{$sort:{totalQty:1}}]);
< {
    _id: 'short',
    totalQty: 44
  }
  {
    _id: 'tall',
    totalQty: 44
  }
```