Name : Atharva Prabhu

Class : D15A

Roll no. : 42

Experiment – 6: MongoDB

1) **Aim:** To study CRUD operations in MongoDB
2) **Problem Statement:**

A) Create a database, create a collection, insert data, query and manipulate data using various MongoDB operations.

1. Create a database named "inventory".
2. Create a collection named "products" with the fields: (ProductID, ProductName, Category, Price, Stock).
3. Insert 10 documents into the "products" collection.
4. Display all the documents in the "products" collection.
5. Display all the products in the "Electronics" category.
6. Display all the products in ascending order of their names.
7. Display the details of the first 5 products.
8. Display the categories of products with a specific name.
9. Display the number of products in the "Electronics" category.
10. Display all the products without showing the "_id" field.
11. Display all the distinct categories of products.
12. Display products in the "Electronics" category with prices greater than 50 but less than 100.
13. Change the price of a product.
14. Delete a particular product entry.

3) **Theory:**

1. Describe some of the features of MongoDB?

MongoDB is a **NoSQL document-oriented database** known for its flexibility, scalability, and high performance. Some key features include:

Document-Oriented Storage – Stores data in JSON-like BSON format, allowing flexible schemas.

Scalability – Supports **horizontal scaling** via **sharding**, distributing data across multiple servers.

High Performance – Indexing, in-memory computing, and query optimization ensure fast read/write operations.

Schema Flexibility – No fixed schema; fields can vary across documents in a collection.

Replication (High Availability) – Uses **replica sets** to ensure data availability and fault tolerance.

Aggregation Framework – Provides powerful data processing similar to SQL's GROUP BY and JOIN operations.

Full-Text Search – Built-in indexing and text search capabilities.

2. What are Documents and Collections in MongoDB?
   A **document** is a key-value pair structure (JSON-like BSON format).

Example of a document:

```
 json
CopyEdit
{
"_id": 1,
"name": "John Doe",
"age": 30,
"city": "New York"
}
```

Collections:

a. A **collection** is a group of **documents** (similar to a table in relational databases).

Example of a collection named **"users"** containing multiple documents:

```
 json
CopyEdit
[
  { "_id": 1, "name": "John", "age": 25 },
  { "_id": 2, "name": "Jane", "age": 30 }
]
```

3. When to use MongoDB?

   MongoDB is best suited for:

   Big Data Applications – Handles large amounts of unstructured data efficiently.

   Real-Time Analytics – Fast read/write operations for analytics and dashboards.

   Content Management Systems (CMS) – Stores diverse and dynamic content types.

   IoT & Mobile Applications – Ideal for handling sensor data, logs, and real-time updates.

   E-commerce Platforms – Flexible schema supports varied product catalogs.

   Cloud-based Applications – Highly scalable, making it a good fit for cloud computing.

   Social Media & Chat Applications – Fast and scalable for handling user-generated content.

4. What is Sharding in MongoDB?
   Sharding is the process of **splitting large datasets** across multiple servers to ensure high performance and scalability.

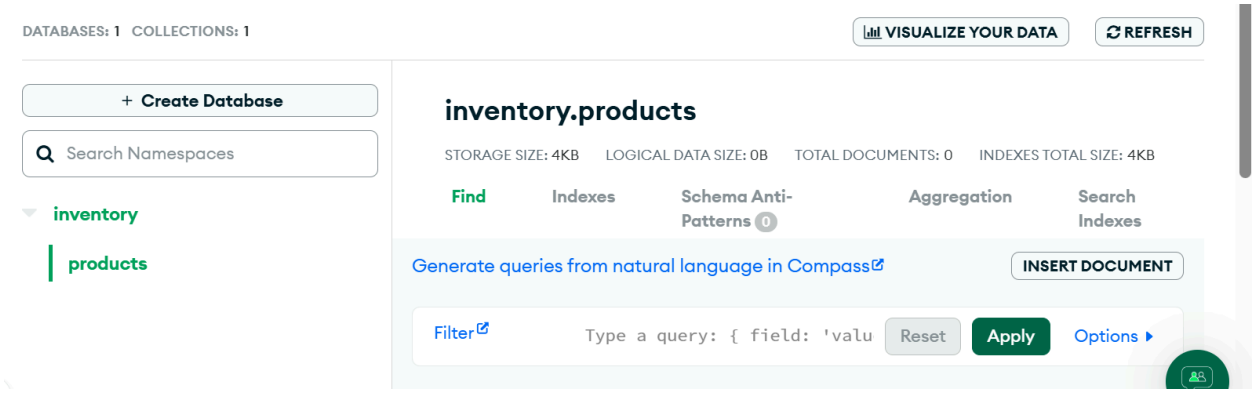
   - **Why Use Sharding?**
     1. Overcomes hardware limitations of a single server.
     2. Ensures **high availability** and **fault tolerance**.
     3. Improves **read and write** performance by distributing queries.

   - **How It Works?**

     1. **Shards** – Data is stored in different partitions (shards).
     2. **Shard Key** – A unique key is used to distribute data across shards.
     3. **Config Servers** – Stores metadata about shards.
     4. **Query Routing (Mongos)** – Routes queries to the correct shard.

   Example: A large e-commerce website may use **sharding** to distribute **order data** based on `UserID` across multiple servers to balance the load.

4) **Output:**




```
> db["products"].find()
<
> use inventory
< switched to db inventory
> db.products.insertMany([
    { ProductID: 1, ProductName: "Laptop", Category: "Electronics", Price: 800, Stock: 10 },
    { ProductID: 2, ProductName: "Smartphone", Category: "Electronics", Price: 600, Stock: 20 },
    { ProductID: 3, ProductName: "Headphones", Category: "Electronics", Price: 50, Stock: 50 },
    { ProductID: 4, ProductName: "Mouse", Category: "Electronics", Price: 30, Stock: 40 },
    { ProductID: 5, ProductName: "Keyboard", Category: "Electronics", Price: 45, Stock: 35 },
    { ProductID: 6, ProductName: "Table", Category: "Furniture", Price: 150, Stock: 5 },
    { ProductID: 7, ProductName: "Chair", Category: "Furniture", Price: 80, Stock: 15 },
    { ProductID: 8, ProductName: "Notebook", Category: "Stationery", Price: 5, Stock: 100 },
    { ProductID: 9, ProductName: "Pen", Category: "Stationery", Price: 2, Stock: 200 },
    { ProductID: 10, ProductName: "Monitor", Category: "Electronics", Price: 120, Stock: 25 }
])
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67e4cf65aed5a843c28697f3'),
    '1': ObjectId('67e4cf65aed5a843c28697f4'),
    '2': ObjectId('67e4cf65aed5a843c28697f5'),
    '3': ObjectId('67e4cf65aed5a843c28697f6'),
    '4': ObjectId('67e4cf65aed5a843c28697f7'),
    '5': ObjectId('67e4cf65aed5a843c28697f8'),
```

```
>_MONGOSH
> db.products.find().pretty()
< {
    _id: ObjectId('67e4cf65aed5a843c28697f3'),
    ProductID: 1,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 800,
    Stock: 10
  }
  {
    _id: ObjectId('67e4cf65aed5a843c28697f4'),
    ProductID: 2,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 600,
    Stock: 20
  }
  {
    _id: ObjectId('67e4cf65aed5a843c28697f5'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 50,
    Stock: 50
  }
```

```
> db.products.distinct("Category")
< [ 'Electronics', 'Furniture', 'Stationery' ]
> db.products.find({ Category: "Electronics", Price: { $gt: 50, $lt: 100 } }).pretty()
<
> db.products.updateOne({ ProductName: "Laptop" }, { $set: { Price: 750 } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.products.deleteOne({ ProductName: "Pen" })
< {
    acknowledged: true,
    deletedCount: 1
  }
Atlas atlas-jsa0wb-shard-0 [primary] inventory>
```

```
>_MONGOSH
> db.products.find({ ProductName: "Laptop" }, { Category: 1, _id: 0 }).pretty()
< {
    Category: 'Electronics'
  }
> db.products.countDocuments({ Category: "Electronics" })
< 6
> db.products.find({}, { _id: 0 }).pretty()
< {
    ProductID: 1,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 800,
    Stock: 10
  }
  {
    ProductID: 2,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 600,
    Stock: 20
  }
  {
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
```

>_MONGOSH

```
> db.products.find().limit(5).pretty()
< {
    _id: ObjectId('67e4cf65aed5a843c28697f3'),
    ProductID: 1,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 800,
    Stock: 10
  }
  {
    _id: ObjectId('67e4cf65aed5a843c28697f4'),
    ProductID: 2,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 600,
    Stock: 20
  }
  {
    _id: ObjectId('67e4cf65aed5a843c28697f5'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 50,
    Stock: 50
  }
```

>_MONGOSH

```
> db.products.find({ Category: "Electronics" }).pretty()
< {
    _id: ObjectId('67e4cf65aed5a843c28697f3'),
    ProductID: 1,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 800,
    Stock: 10
  }
  {
    _id: ObjectId('67e4cf65aed5a843c28697f4'),
    ProductID: 2,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 600,
    Stock: 20
  }
  {
    _id: ObjectId('67e4cf65aed5a843c28697f5'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 50,
    Stock: 50
  }
```