Experiment – 6: MongoDB

Name of Student	Mahvish Siddiqui
Class Roll No	D15A 56
D.O.P.	
D.O.S.	
Sign and Grade	

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:**
   A. Describe some of the features of MongoDB?

   <u>Features of MongoDB:</u>

   MongoDB is a popular NoSQL database that is known for its scalability, flexibility, and performance. Here are some of the key features:

   - Document-Oriented Storage: MongoDB stores data in BSON (Binary JSON) format, which allows for flexible and hierarchical data structures.
   - Scalability: MongoDB is designed to scale horizontally, meaning you can distribute data across multiple machines (sharding).
   - High Availability: With replica sets, MongoDB provides automatic failover and data redundancy, ensuring high availability.

- Indexing: MongoDB supports a wide range of indexes (e.g., single field, compound, geospatial, and text indexes) to optimize query performance.
- Aggregation Framework: MongoDB provides an aggregation framework that allows for complex data transformations and computations.
- Schema Flexibility: MongoDB allows you to have a dynamic schema, meaning documents in the same collection do not need to have the same structure.
- Rich Query Language: MongoDB supports a rich query language that allows for querying based on various data types and operations like equality, range, and pattern matching.
- Replication: MongoDB uses replica sets for data replication, which enhances data availability and fault tolerance.
- Built-in Sharding: MongoDB can distribute data across multiple servers using sharding to manage large datasets.

B. What are Documents and Collections in MongoDB?

Documents:

- A document is a basic unit of data in MongoDB, which is represented in BSON (Binary JSON) format.
- It is similar to a row in a relational database, but it is more flexible, as each document can have a different structure.
- Documents can contain arrays, nested documents, and various data types (e.g., strings, integers, dates, etc.).
- Eg.

```
{
  "_id": ObjectId("605c72ef1532074b8b10a7e4"),
  "name": "John Doe",
  "age": 30,
  "address": {
    "street": "123 Main St",
    "city": "New York"
  }
}
```

Collections:

- A collection is a group of documents in MongoDB. It is equivalent to a table in relational databases but does not enforce any schema.
- Collections are where the documents are stored. You can think of a collection as a container for similar types of documents.
- Collections are schema-less, meaning documents within the same collection do not have to have the same fields or structure.
- Example: A collection named users might contain documents about different users.

C. When to use MongoDB?

MongoDB is ideal in the following scenarios:

- Unstructured or Semi-Structured Data: If your data is not rigidly structured, MongoDB allows you to store documents with varying fields.
- Big Data Applications: When you have large datasets or need horizontal scaling, MongoDB is designed to scale across many machines.
- Rapid Development: MongoDB's flexible schema and rich query capabilities make it a great choice for applications that require quick iteration and development.
- Real-Time Analytics: MongoDB's aggregation framework makes it well-suited for real-time data processing and analysis.
- Geospatial Data: If your application needs to store and query geospatial data, MongoDB supports geospatial indexes and queries.
- Content Management and Cataloging: MongoDB's flexibility is beneficial for systems managing content and product catalogs, where the data model may evolve over time.
- High Write Throughput: MongoDB can handle high write volumes, making it suitable for logging, session management, and real-time data collection.

D. What is Sharding in MongoDB?

Sharding is the process of distributing data across multiple servers, or shards, to ensure horizontal scaling. It allows MongoDB to handle large amounts of data and high-throughput applications by spreading the load.

- Sharding Key: When sharding, a shard key is chosen to determine how data is distributed. The shard key can be a single field or compound key.
- Chunks: Data is divided into chunks based on the shard key, and each chunk is distributed to different servers or shards.

- Shard: A shard is a single MongoDB server or replica set that stores a subset of the data.
- Config Servers: These servers store metadata about the distribution of data and the location of chunks across the shards.
- Query Router: Also known as mongos, it is responsible for routing client requests to the appropriate shard based on the shard key.
- Sharding helps MongoDB scale horizontally, allowing it to handle massive datasets and high-velocity applications while maintaining performance.

4) **Output:**

1. Create a database named "inventory"

```
test> use inventory
switched to db inventory
inventory>
```

2. Create a collection “Products”

```
inventory> db.createCollection('products');
{ ok: 1 }
inventory>
```

3. Insert 10 items

```
inventory> db.products.insertMany([
...   { ProductID: 1, ProductName: "Smartphone", Category: "Electronics", Price: 299, Stock: 100 },
...   { ProductID: 2, ProductName: "Laptop", Category: "Electronics", Price: 799, Stock: 50 },
...   { ProductID: 3, ProductName: "Headphones", Category: "Electronics", Price: 59, Stock: 200 },
...   { ProductID: 4, ProductName: "Smartwatch", Category: "Electronics", Price: 199, Stock: 150 },
...   { ProductID: 5, ProductName: "Coffee Maker", Category: "Appliances", Price: 49, Stock: 120 },
...   { ProductID: 6, ProductName: "Blender", Category: "Appliances", Price: 69, Stock: 80 },
...   { ProductID: 7, ProductName: "Gaming Console", Category: "Electronics", Price: 399, Stock: 30 },
...   { ProductID: 8, ProductName: "Microwave", Category: "Appliances", Price: 129, Stock: 60 },
...   { ProductID: 9, ProductName: "Refrigerator", Category: "Appliances", Price: 599, Stock: 40 },
...   { ProductID: 10, ProductName: "TV", Category: "Electronics", Price: 499, Stock: 70 }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67c95f613fdb03e86afa4214'),
    '1': ObjectId('67c95f613fdb03e86afa4215'),
    '2': ObjectId('67c95f613fdb03e86afa4216'),
    '3': ObjectId('67c95f613fdb03e86afa4217'),
    '4': ObjectId('67c95f613fdb03e86afa4218'),
    '5': ObjectId('67c95f613fdb03e86afa4219'),
    '6': ObjectId('67c95f613fdb03e86afa421a'),
    '7': ObjectId('67c95f613fdb03e86afa421b'),
    '8': ObjectId('67c95f613fdb03e86afa421c'),
    '9': ObjectId('67c95f613fdb03e86afa421d')
  }
}
inventory>
```

4. Display all the documents in the "products" collection.

```
inventory> db.products.find();
[
  {
    _id: ObjectId('67c95f613fdb03e86afa4214'),
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 100
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa4215'),
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 799,
    Stock: 50
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa4216'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 59,
    Stock: 200
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa4217'),
    ProductID: 4,
    ProductName: 'Smartwatch',
    Category: 'Electronics',
    Price: 199,
    Stock: 150
```

5. Display all the products in the "Electronics" category

```
inventory> db.products.find({ Category: "Electronics" });
[
  {
    _id: ObjectId('67c95f613fdb03e86afa4214'),
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 100
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa4215'),
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 799,
    Stock: 50
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa4216'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 59,
    Stock: 200
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa4217'),
    ProductID: 4,
    ProductName: 'Smartwatch',
    Category: 'Electronics',
```

6. Display all the products in ascending order of their names.

```
inventory> db.products.find().sort({ ProductName: 1 });
[
  {
    _id: ObjectId('67c95f613fdb03e86afa4219'),
    ProductID: 6,
    ProductName: 'Blender',
    Category: 'Appliances',
    Price: 69,
    Stock: 80
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa4218'),
    ProductID: 5,
    ProductName: 'Coffee Maker',
    Category: 'Appliances',
    Price: 49,
    Stock: 120
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa421a'),
    ProductID: 7,
    ProductName: 'Gaming Console',
    Category: 'Electronics',
    Price: 399,
    Stock: 30
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa4216'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 59,
    Stock: 200
```

7. Display the details of the first 5 products.

```
inventory> db.products.find().limit(5);
[
  {
    _id: ObjectId('67c95f613fdb03e86afa4214'),
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 100
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa4215'),
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 799,
    Stock: 50
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa4216'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 59,
    Stock: 200
  },
  {
    _id: ObjectId('67c95f613fdb03e86afa4217'),
    ProductID: 4,
    ProductName: 'Smartwatch',
    Category: 'Electronics',
    Price: 199,
    Stock: 150
  },
  {
```

8. Display the categories of products with a specific name.

```
inventory> db.products.find({ ProductName: "Smartphone" }, { Category: 1 });
[
  {
    _id: ObjectId('67c95f613fdb03e86afa4214'),
    Category: 'Electronics'
  }
]
inventory>
```

9. Display the number of products in the "Electronics" category.

```
inventory> db.products.countDocuments({ Category: "Electronics" });
6
inventory>
```

10. Display all the products without showing the "_id" field.

```
inventory> db.products.find({}, { _id: 0 });
[
  {
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 100
  },
  {
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 799,
    Stock: 50
  },
  {
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 59,
    Stock: 200
  },
  {
    ProductID: 4,
    ProductName: 'Smartwatch',
    Category: 'Electronics',
    Price: 199,
    Stock: 150
  },
  {
    ProductID: 5,
```

11. Display all the distinct categories of products.

```
inventory> db.products.distinct("Category");
[ 'Appliances', 'Electronics' ]
inventory>
```

12. Display products in the "Electronics" category with prices greater than 50 but less than 100.

```
inventory> db.products.find({
...   Category: "Electronics",
...   Price: { $gt: 50, $lt: 100 }
... });
[
  {
    _id: ObjectId('67c95f613fdb03e86afa4216'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 59,
    Stock: 200
  }
]
inventory>
```

13. Change the price of a product.

```
inventory> db.products.updateOne(
...   { ProductID: 1 },
...   { $set: { Price: 350 } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
inventory>
```

14. Delete a particular product entry.

```
inventory> db.products.deleteOne({ ProductID: 2 });
{ acknowledged: true, deletedCount: 1 }
inventory>
```

CONCLUSION

In this experiment, we successfully explored CRUD operations in MongoDB by creating and managing a database named inventory and a collection of products. We performed various operations, including inserting, querying, sorting, filtering, updating, and deleting data. These exercises provided a hands-on understanding of MongoDB's powerful data manipulation capabilities, reinforcing its flexibility and efficiency in handling NoSQL databases.