EXPERIMENT NO. 6 : MongoDB

Name of Student	**Komal Milind Deolekar**
Class Roll No	**14**
D.O.P.	**18-03-2025**
D.O.S.	**25-03-2025**
Sign and Grade	

Aim : To study CRUD operations in MongoDB

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.

Github Link :
https://github.com/KomalDeolekar0607/Webx_Lab/tree/main/Webx_Lab_Exp_6

Theory :

A. Describe some of the features of MongoDB?

MongoDB is a NoSQL document-oriented database designed for high performance, scalability, and flexibility. Unlike relational databases, it does not use tables and rows but instead stores data in JSON-like BSON (Binary JSON) documents.

Key Features:

1. Schema-Less Database:

- No fixed schema is required, meaning each document in a collection can have a different structure.
- Provides flexibility in handling dynamic and evolving data models.

```
{
   "name": "Alice",
   "email": "alice@example.com",
   "age": 25
}

{
   "name": "Bob",
   "phone": "1234567890",
   "city": "New York"
}
```

- **Note:** Unlike SQL, Alice's document has an email field, while Bob's has a phone field.

2. High Scalability & Horizontal Scaling:

- MongoDB supports **sharding**, which allows data to be distributed across multiple servers for load balancing.
- Enables **horizontal scaling**, making it ideal for applications handling **big data** and high user traffic.

3. Replication for High Availability:

- Supports **replica sets**, which ensure database availability in case of failure.
- Data is automatically synchronized between multiple servers.

4. Indexing for Fast Query Performance:

- MongoDB uses **indexes** to speed up query performance.
- Supports **single field, compound, geospatial, text, and hashed indexes**.

5. Rich Query Language:

Supports **filtering, sorting, aggregation, and geospatial queries**.

Allows **document-based queries** without requiring complex joins.

Example query:

```
db.users.find({"age": {"$gt": 18}})
```

Returns all users where age > 18.

B. What are Documents and Collections in MongoDB?

MongoDB stores data in documents, which are grouped into collections. These replace rows and tables in relational databases.

What is a Document?

- A **document** is a JSON-like structure that stores **key-value pairs**.
- Documents **do not require a predefined schema**, making them flexible.

Example:

```
{
  "_id": 1,
  "name": "Alice",
```

```
    "email": "alice@example.com",
    "age": 25
}
```

- _id: A unique identifier (default generated by MongoDB).
- name, email, and age are user-defined fields.

What is a Collection?

- A **collection** is a group of documents (similar to an SQL table but without strict structure).

  Example: users collection with multiple documents:

```
[
  { "_id": 1, "name": "Alice", "email": "alice@example.com", "age": 25 },
  { "_id": 2, "name": "Bob", "phone": "1234567890", "city": "New York" }
]
```

Comparison with SQL:

SQL Concept	MongoDB Equivalent
Table	Collection
Row	Document
Column	Field
Primary Key	_id field (unique identifier)

C. When to use MongoDB?

Best Use Cases for MongoDB:

Use Case	Why MongoDB?
Big Data Applications	Handles large datasets with high speed.
Real-Time Analytics	Fast data ingestion and querying.
E-commerce Platforms	Stores flexible product catalogs.
Social Media Applications	Handles unstructured and semi-structured data efficiently.
IoT Applications	Stores high-velocity sensor data.
Cloud-based Apps	Works well with cloud services like AWS, Azure, and Google Cloud.

When NOT to Use MongoDB?

Use Case	Why Not MongoDB?
Financial/Banking Applications	Requires ACID transactions and strict consistency.
Relational Data with Complex Joins	MongoDB does not support traditional SQL joins.
Small Applications with Low Data Volume	Relational databases might be simpler to manage.

D. What is Sharding in MongoDB?

Definition:

Sharding is MongoDB's horizontal scaling feature that distributes data across multiple servers (shards) to improve performance and data availability.

Why Use Sharding?

To handle **large datasets** (terabytes or petabytes).
To improve **read/write performance** by distributing queries.
To enable **fault tolerance** and **high availability**.

Components of Sharding:

1. **Shards:** Store the actual data.
2. **Config Servers:** Manage metadata and routing.
3. **Query Routers (mongos):** Direct queries to the correct shard.

Example of Sharded Data:

- Suppose we have a large **user database** and decide to shard based on **location (country)**.

Shard 1 (USA)	Shard 2 (India)	Shard 3 (UK)
John (NY)	Rahul (Delhi)	Alice (London)
Mike (LA)	Priya (Mumbai)	Bob (Manchester)

- Now, when a user searches for "Alice," the query router (mongos) **only queries Shard 3 (UK)**, reducing the load.

How to Implement Sharding?

Enable sharding on a database:

```
sh.enableSharding("mydatabase")
```

Create an **index on the shard key**:

```
db.users.createIndex({ "country": 1 })
```

Enable **sharding on the collection**:

```
sh.shardCollection("mydatabase.users", { "country": 1 })
```

Feature	Sharding	Replication
Purpose	Splits data across servers	Copies data across servers
Scaling Type	**Horizontal Scaling**	**Fault Tolerance & Read Scaling**
Load Balancing	Distributes queries efficiently	Handles failover in case of a crash
Use Case	**Large-scale databases**	**High availability & backups**

Output :

In Mongosh

Starting mongoDB

```
C:\Users\Komal>mongosh
Current Mongosh Log ID: 67f9392af0ade1a7194eeb85
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.3.7
Using MongoDB:          8.0.4
Using Mongosh:          2.3.7

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-04-09T01:51:48.803+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------

test>
```

```
test> show dbs
StudentsDB_old   40.00 KiB
admin            40.00 KiB
config          108.00 KiB
inventory_old    40.00 KiB
local            80.00 KiB
student_db       72.00 KiB
test>
```

Creating a MongoDB Database Named inventory

```
test> use inventory
```

```
inventory>
switched to db inventory
```

Creating a Collection Named products

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

Inserting 10 Documents into the products Collection

```
inventory> db.products.insertMany([
...   { ProductID: 1, ProductName: "Smartphone", Category: "Electronics", Price: 299, Stock: 50 },
...   { ProductID: 2, ProductName: "Laptop", Category: "Electronics", Price: 999, Stock: 30 },
...   { ProductID: 3, ProductName: "Tablet", Category: "Electronics", Price: 199, Stock: 40 },
...   { ProductID: 4, ProductName: "Headphones", Category: "Electronics", Price: 89, Stock: 100 },
...   { ProductID: 5, ProductName: "Keyboard", Category: "Electronics", Price: 49, Stock: 70 },
...   { ProductID: 6, ProductName: "Shoes", Category: "Footwear", Price: 60, Stock: 80 },
...   { ProductID: 7, ProductName: "Chair", Category: "Furniture", Price: 120, Stock: 25 },
...   { ProductID: 8, ProductName: "Watch", Category: "Accessories", Price: 150, Stock: 45 },
...   { ProductID: 9, ProductName: "Camera", Category: "Electronics", Price: 450, Stock: 20 },
...   { ProductID: 10, ProductName: "Mouse", Category: "Electronics", Price: 30, Stock: 90 }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67f94d85c80e1146104eeb86'),
    '1': ObjectId('67f94d85c80e1146104eeb87'),
    '2': ObjectId('67f94d85c80e1146104eeb88'),
    '3': ObjectId('67f94d85c80e1146104eeb89'),
    '4': ObjectId('67f94d85c80e1146104eeb8a'),
    '5': ObjectId('67f94d85c80e1146104eeb8b'),
    '6': ObjectId('67f94d85c80e1146104eeb8c'),
    '7': ObjectId('67f94d85c80e1146104eeb8d'),
    '8': ObjectId('67f94d85c80e1146104eeb8e'),
    '9': ObjectId('67f94d85c80e1146104eeb8f')
  }
}
```

Displaying All Documents in the products Collection

```
inventory> db.products.find()
[
  {
    _id: ObjectId('67f94d85c80e1146104eeb86'),
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 50
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb87'),
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 999,
    Stock: 30
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb88'),
    ProductID: 3,
    ProductName: 'Tablet',
    Category: 'Electronics',
    Price: 199,
    Stock: 40
  },
```

```
  {
    _id: ObjectId('67f94d85c80e1146104eeb89'),
    ProductID: 4,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 89,
    Stock: 100
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb8a'),
    ProductID: 5,
    ProductName: 'Keyboard',
    Category: 'Electronics',
    Price: 49,
    Stock: 70
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb8b'),
    ProductID: 6,
    ProductName: 'Shoes',
    Category: 'Footwear',
    Price: 60,
    Stock: 80
  },
```

```
  {
    _id: ObjectId('67f94d85c80e1146104eeb8c'),
    ProductID: 7,
    ProductName: 'Chair',
    Category: 'Furniture',
    Price: 120,
    Stock: 25
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb8d'),
    ProductID: 8,
    ProductName: 'Watch',
    Category: 'Accessories',
    Price: 150,
    Stock: 45
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb8e'),
    ProductID: 9,
    ProductName: 'Camera',
    Category: 'Electronics',
    Price: 450,
    Stock: 20
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb8f'),
    ProductID: 10,
    ProductName: 'Mouse',
    Category: 'Electronics',
    Price: 30,
    Stock: 90
  }
]
inventory>
```

Displaying All Products in the Electronics Category.

```
inventory> db.products.find({ Category: "Electronics" })
[
  {
    _id: ObjectId('67f94d85c80e1146104eeb86'),
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 50
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb87'),
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 999,
    Stock: 30
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb88'),
    ProductID: 3,
inventory>
    Category: 'Electronics',
    Price: 199,
    Stock: 40
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb89'),
    ProductID: 4,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 89,
    Stock: 100
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb8a'),
    ProductID: 5,
    ProductName: 'Keyboard',
    Category: 'Electronics',
    Price: 49,
    Stock: 70
  },
```

Displaying All Products in Ascending Order of Their Names

```
inventory> db.products.find().sort({ ProductName: 1 })
[
  {
    _id: ObjectId('67f94d85c80e1146104eeb8e'),
    ProductID: 9,
    ProductName: 'Camera',
    Category: 'Electronics',
    Price: 450,
    Stock: 20
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb8c'),
    ProductID: 7,
    ProductName: 'Chair',
    Category: 'Furniture',
    Price: 120,
    Stock: 25
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb89'),
    ProductID: 4,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 89,
    Stock: 100
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb8a'),
    ProductID: 5,
    ProductName: 'Keyboard',
    Category: 'Electronics',
    Price: 49,
    Stock: 70
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb87'),
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 999,
    Stock: 30
  },
```

```
  {
    _id: ObjectId('67f94d85c80e1146104eeb8f'),
    ProductID: 10,
    ProductName: 'Mouse',
    Category: 'Electronics',
    Price: 30,
    Stock: 90
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb8b'),
    ProductID: 6,
    ProductName: 'Shoes',
    Category: 'Footwear',
    Price: 60,
    Stock: 80
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb86'),
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 50
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb88'),
    ProductID: 3,
    ProductName: 'Tablet',
    Category: 'Electronics',
    Price: 199,
    Stock: 40
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb8d'),
    ProductID: 8,
    ProductName: 'Watch',
    Category: 'Accessories',
    Price: 150,
    Stock: 45
  }
]
inventory>
```

Displaying Details of the First 5 Products

```
inventory> db.products.find().limit(5)
[
  {
    _id: ObjectId('67f94d85c80e1146104eeb86'),
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 50
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb87'),
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 999,
    Stock: 30
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb88'),
    ProductID: 3,
    ProductName: 'Tablet',
    Category: 'Electronics',
    Price: 199,
    Stock: 40
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb89'),
    ProductID: 4,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 89,
    Stock: 100
  },
  {
    _id: ObjectId('67f94d85c80e1146104eeb8a'),
    ProductID: 5,
    ProductName: 'Keyboard',
    Category: 'Electronics',
    Price: 49,
    Stock: 70
  }
]
inventory>
```

Displaying the Category of Products with a Specific Name

```
inventory> db.products.find({ ProductName: "Laptop" }, { Category: 1, _id: 0 })
[ { Category: 'Electronics' } ]
inventory>
```

Displaying the Number of Products in the Electronics Category

```
inventory> db.products.countDocuments({ Category: "Electronics" })
7
inventory>
```

Displaying All Products Without Showing the _id Field

```
inventory> db.products.find({}, { _id: 0 })
[
  {
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 50
  },
  {
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 999,
    Stock: 30
  },
  {
    ProductID: 3,
    ProductName: 'Tablet',
    Category: 'Electronics',
    Price: 199,
    Stock: 40
  },
inventory>
    ProductID: 4,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 89,
    Stock: 100
  },
  {
    ProductID: 5,
    ProductName: 'Keyboard',
    Category: 'Electronics',
    Price: 49,
    Stock: 70
  },
```

```
  {
    ProductID: 6,
    ProductName: 'Shoes',
    Category: 'Footwear',
    Price: 60,
    Stock: 80
  },
  {
    ProductID: 7,
    ProductName: 'Chair',
    Category: 'Furniture',
    Price: 120,
    Stock: 25
  },
  {
    ProductID: 8,
    ProductName: 'Watch',
    Category: 'Accessories',
    Price: 150,
    Stock: 45
  },
  {
    ProductID: 9,
    ProductName: 'Camera',
    Category: 'Electronics',
    Price: 450,
    Stock: 20
  },
  {
    ProductID: 10,
    ProductName: 'Mouse',
    Category: 'Electronics',
    Price: 30,
    Stock: 90
  }
]
inventory>
```

Displaying All Distinct Product Categories in the Collection

```
inventory> db.products.distinct("Category")
[ 'Accessories', 'Electronics', 'Footwear', 'Furniture' ]
inventory>
```

Displaying Products in Electronics Category Priced Between 50 and 100

```
inventory> db.products.find({ Category: "Electronics", Price: { $gt: 50, $lt: 100 } })
[
  {
    _id: ObjectId('67f94d85c80e1146104eeb89'),
    ProductID: 4,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 89,
    Stock: 100
  }
]
inventory>
```

Changing the Price of a Specific Product (Mouse price to 35)

```
inventory> db.products.updateOne(
...   { ProductName: "Mouse" },
...   { $set: { Price: 35 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

After Updation

```
inventory> db.products.findOne( { ProductName: "Mouse" } )
{
  _id: ObjectId('67f94d85c80e1146104eeb8f'),
  ProductID: 10,
  ProductName: 'Mouse',
  Category: 'Electronics',
  Price: 35,
  Stock: 90
}
inventory>
```

Deleting a Particular Product(Camera) Entry from the Collection

```
inventory> db.products.deleteOne({ ProductName: "Camera" })
{ acknowledged: true, deletedCount: 1 }
```

After deletion

```
inventory> db.products.findOne({ ProductName: "Camera" })
null
inventory>
```

With Python (pymongo)

Code :

exp6.py

```
from pymongo import MongoClient


# Connect to MongoDB
client = MongoClient("mongodb://localhost:27017/")  # Update if using a different host or port
print("Connected to MongoDb")


db = client["inventory"]  # Create or connect to the 'inventory' database
print("Created Inventory DataBase")


collection = db["products"]  # Create or connect to the 'products' collection
print("Created Products collection")


# Insert 10 documents
products = [
    {"ProductID": 1, "ProductName": "Laptop", "Category": "Electronics", "Price": 800, "Stock": 10},
```

```
    {"ProductID": 2, "ProductName": "Smartphone", "Category": "Electronics", "Price": 500, "Stock": 25},
    {"ProductID": 3, "ProductName": "Headphones", "Category": "Electronics", "Price": 50, "Stock": 50},
    {"ProductID": 4, "ProductName": "Table", "Category": "Furniture", "Price": 150, "Stock": 15},
    {"ProductID": 5, "ProductName": "Chair", "Category": "Furniture", "Price": 80, "Stock": 30},
    {"ProductID": 6, "ProductName": "T-shirt", "Category": "Clothing", "Price": 20, "Stock": 100},
    {"ProductID": 7, "ProductName": "Shoes", "Category": "Clothing", "Price": 60, "Stock": 40},
    {"ProductID": 8, "ProductName": "TV", "Category": "Electronics", "Price": 900, "Stock": 5},
    {"ProductID": 9, "ProductName": "Watch", "Category": "Accessories", "Price": 120, "Stock": 20},
    {"ProductID": 10, "ProductName": "Bag", "Category": "Accessories", "Price": 40, "Stock": 35},
    {"ProductID": 11, "ProductName": "Laptop", "Category": "Electronics", "Price": 1000, "Stock": 15},
    {"ProductID": 12, "ProductName": "Jeans", "Category": "Clothing", "Price": 500, "Stock": 12},
     {"ProductID": 13, "ProductName": "Lesor", "Category": "Electronics", "Price": 90, "Stock": 15},
      {"ProductID": 14, "ProductName": "Jacket", "Category": "Clothing", "Price": 400, "Stock": 12},
       {"ProductID": 15, "ProductName": "Bulb", "Category": "Electronics", "Price": 70, "Stock": 25},
]


# Insert data into MongoDB
collection.insert_many(products)
print("\n Inserted 10 products successfully.")


# Display all documents
print("\n All Products:")
for product in collection.find():
    print(product)
```

```
# Display all products in 'Electronics' category
print("\n Electronics Products:")
for product in collection.find({"Category": "Electronics"}):
    print(product)

# Display all products in ascending order of ProductName
print("\n Products Sorted by Name:")
for product in collection.find().sort("ProductName",1):
    print(product)



# Display categories of products with a specific name
product_name = "Laptop"
print(f"\n Category of '{product_name}':")
for product in collection.find({"ProductName":product_name}):
    print(product)

# Display categories of products with a specific name
product_name = "Laptop"
print(f"\n Category of '{product_name}':")
for product in collection.find({"ProductName":product_name},{ "ProductName" : 1 ,"Category":1,"_id" : 0}):
    print(product)
```

```
# Display the number of products in the 'Electronics' category
electronics_count = collection.count_documents({"Category":"Electronics"})
print(f"\n Number of Electronics Products: {electronics_count}")

# Display all products without the "_id" field
print("\n Products Without '_id':")
for product in collection.find({},{"_id":0}):
    print(product)

# Display all distinct categories
# for category in collection.distinct("Category"):
#     print(category)

# Display all distinct categories
categories = collection.distinct("Category")
print("\n Distinct Product Categories:", categories)

# Display products in 'Electronics' category with prices >50 and <100
print("\n Electronics Products with Price Between 50 and 100:")
for product in collection.find({"Category":"Electronics" , "Price" : {"$gt" : 50 , "$lt" : 100 }}):
    print(product)

# Update the price of a product
collection.update_one({'ProductName': 'TV'}, {"$set" : {"Price" : 850}})
```

```
print("\n Price of Laptop updated to 850.")


product = collection.find_one({"ProductName" : "TV"})
print(product)


# Delete a particular product entry
collection.delete_one({"ProductName" : "Chair"})
print("\n Product 'Chair' deleted.")


for product in collection.find({"ProductName" : "Chair"}):
    print(product)
```

Output :

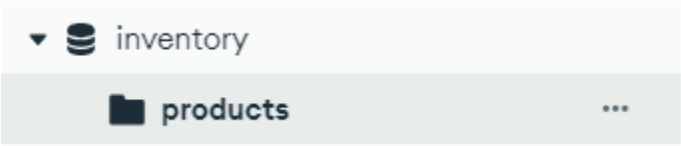
```
pip install pymongo
```

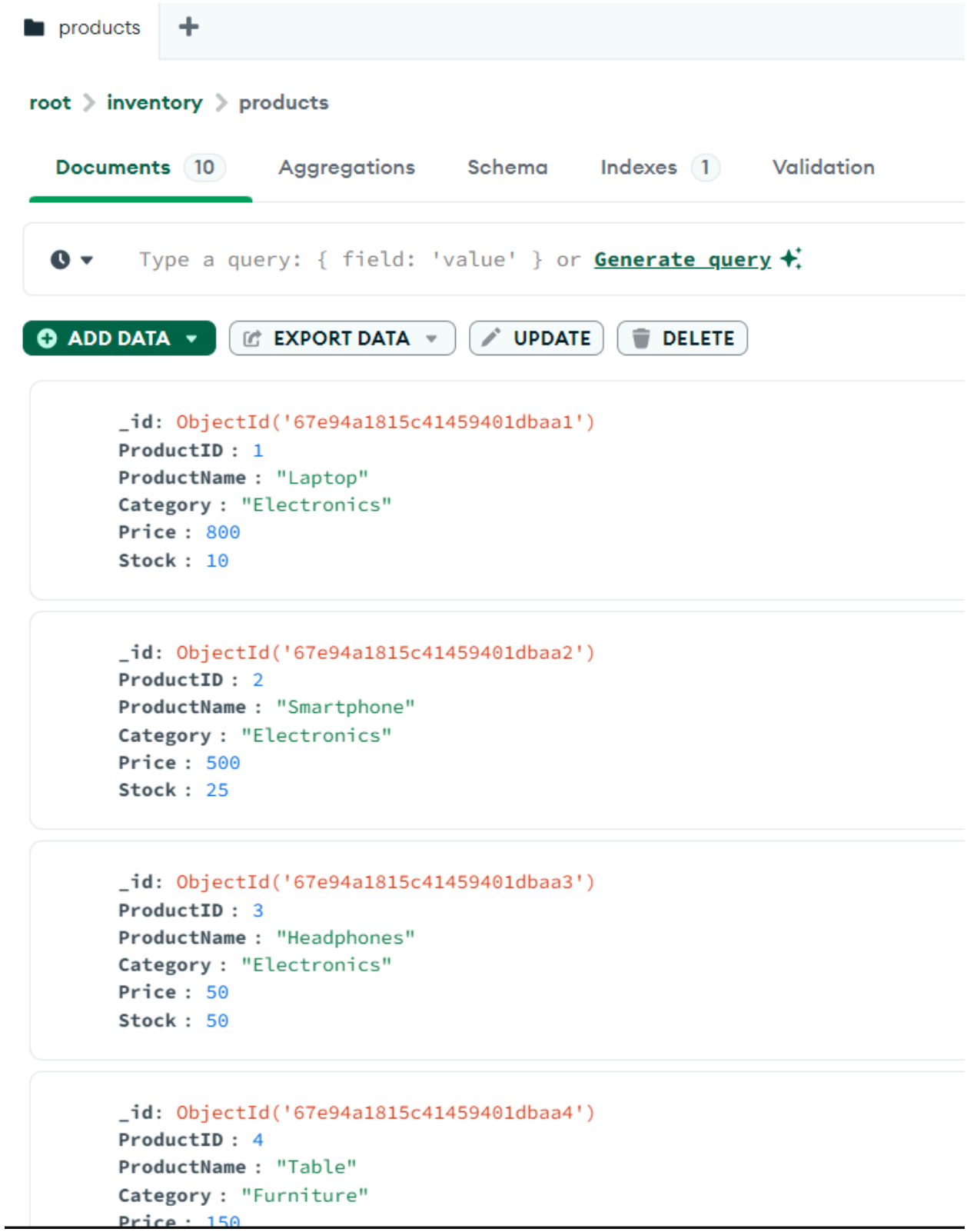
Start your **MongoDB server** (if not running):

```
bash

mongod
```

```
D:\Users\Komal\OneDrive\Desktop\sem 6\webx_lab\mongodb_lab_6>pip install pymongo
Collecting pymongo
  Downloading pymongo-4.11.3-cp313-cp313-win_amd64.whl.metadata (22 kB)
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.7.0-py3-none-any.whl.metadata (5.8 kB)
Downloading pymongo-4.11.3-cp313-cp313-win_amd64.whl (932 kB)
   ---------------------------------------- 932.9/932.9 kB 6.1 MB/s eta 0:00:00
Downloading dnspython-2.7.0-py3-none-any.whl (313 kB)
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.7.0 pymongo-4.11.3

[notice] A new release of pip is available: 24.3.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip

D:\Users\Komal\OneDrive\Desktop\sem 6\webx_lab\mongodb_lab_6>
```

```
D:\Users\Komal\OneDrive\Desktop\sem 6\webx_lab\mongodb_lab_6>mongod
{"t":{"$date":"2025-03-30T19:07:47.666+05:30"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"thread1","msg":"Automatically disabling TLS 1.0, to for
ce-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2025-03-30T19:07:55.097+05:30"},"s":"I",  "c":"CONTROL",  "id":5945603, "ctx":"thread1","msg":"Multi threading initialized"}
{"t":{"$date":"2025-03-30T19:07:55.102+05:30"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"thread1","msg":"Implicit TCP FastOpen unavailable. If T
CP FastOpen is required, set at least one of the related parameters","attr":{"relatedParameters":["tcpFastOpenServer","tcpFastOpenClient","tcpFastOpe
nQueueSize"]}}
{"t":{"$date":"2025-03-30T19:07:55.136+05:30"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"thread1","msg":"Initialized wire specification","attr":
{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":25},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":25},"outgoin
g":{"minWireVersion":6,"maxWireVersion":25},"isInternalClient":true}}}
{"t":{"$date":"2025-03-30T19:07:55.143+05:30"},"s":"I",  "c":"TENANT_M", "id":7091600, "ctx":"thread1","msg":"Starting TenantMigrationAccessBlockerRe
gistry"}
```




products
root > inventory > products
Documents 10
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
ADD DATA
EXPORT DATA
UPDATE
DELETE
_id: ObjectId('67e94a1815c41459401dbaa1')
ProductID : 1
ProductName : "Laptop"
Category : "Electronics"
Price : 800
Stock : 10
_id: ObjectId('67e94a1815c41459401dbaa2')
ProductID : 2
ProductName : "Smartphone"
Category : "Electronics"
Price : 500
Stock : 25
_id: ObjectId('67e94a1815c41459401dbaa3')
ProductID : 3
ProductName : "Headphones"
Category : "Electronics"
Price : 50
Stock : 50
_id: ObjectId('67e94a1815c41459401dbaa4')
ProductID : 4
ProductName : "Table"
Category : "Furniture"
Price : 150

```
D:\Users\Komal\OneDrive\Desktop\sem 6\webx_lab\mongodb_lab_6>py exp_6.py
Connected to MongoDb
Created Inventory DataBase
Created Products collection
```

```
Inserted 10 products successfully.

All Products:
{'_id': ObjectId('67eb2b160eaea0bbc46996cb'), 'ProductID': 1, 'ProductName': 'Laptop', 'Category': 'Electronics', 'Price': 800, 'Stock': 10}
{'_id': ObjectId('67eb2b160eaea0bbc46996cc'), 'ProductID': 2, 'ProductName': 'Smartphone', 'Category': 'Electronics', 'Price': 500, 'Stock': 25}
{'_id': ObjectId('67eb2b160eaea0bbc46996cd'), 'ProductID': 3, 'ProductName': 'Headphones', 'Category': 'Electronics', 'Price': 50, 'Stock': 50}
{'_id': ObjectId('67eb2b160eaea0bbc46996ce'), 'ProductID': 4, 'ProductName': 'Table', 'Category': 'Furniture', 'Price': 150, 'Stock': 15}
{'_id': ObjectId('67eb2b160eaea0bbc46996cf'), 'ProductID': 5, 'ProductName': 'Chair', 'Category': 'Furniture', 'Price': 80, 'Stock': 30}
{'_id': ObjectId('67eb2b160eaea0bbc46996d0'), 'ProductID': 6, 'ProductName': 'T-shirt', 'Category': 'Clothing', 'Price': 20, 'Stock': 100}
{'_id': ObjectId('67eb2b160eaea0bbc46996d1'), 'ProductID': 7, 'ProductName': 'Shoes', 'Category': 'Clothing', 'Price': 60, 'Stock': 40}
{'_id': ObjectId('67eb2b160eaea0bbc46996d2'), 'ProductID': 8, 'ProductName': 'TV', 'Category': 'Electronics', 'Price': 900, 'Stock': 5}
{'_id': ObjectId('67eb2b160eaea0bbc46996d3'), 'ProductID': 9, 'ProductName': 'Watch', 'Category': 'Accessories', 'Price': 120, 'Stock': 20}
{'_id': ObjectId('67eb2b160eaea0bbc46996d4'), 'ProductID': 10, 'ProductName': 'Bag', 'Category': 'Accessories', 'Price': 40, 'Stock': 35}
{'_id': ObjectId('67eb2b160eaea0bbc46996d5'), 'ProductID': 11, 'ProductName': 'Laptop', 'Category': 'Electronics', 'Price': 1000, 'Stock': 15}
{'_id': ObjectId('67eb2b160eaea0bbc46996d6'), 'ProductID': 12, 'ProductName': 'Jeans', 'Category': 'Clothing', 'Price': 500, 'Stock': 12}
{'_id': ObjectId('67eb2b160eaea0bbc46996d7'), 'ProductID': 13, 'ProductName': 'Lesor', 'Category': 'Electronics', 'Price': 90, 'Stock': 15}
{'_id': ObjectId('67eb2b160eaea0bbc46996d8'), 'ProductID': 14, 'ProductName': 'Jacket', 'Category': 'Clothing', 'Price': 400, 'Stock': 12}
{'_id': ObjectId('67eb2b160eaea0bbc46996d9'), 'ProductID': 15, 'ProductName': 'Bulb', 'Category': 'Electronics', 'Price': 70, 'Stock': 25}
```

```
Electronics Products:
{'_id': ObjectId('67eb2b160eaea0bbc46996cb'), 'ProductID': 1, 'ProductName': 'Laptop', 'Category': 'Electronics', 'Price': 800, 'Stock': 10}
{'_id': ObjectId('67eb2b160eaea0bbc46996cc'), 'ProductID': 2, 'ProductName': 'Smartphone', 'Category': 'Electronics', 'Price': 500, 'Stock': 25}
{'_id': ObjectId('67eb2b160eaea0bbc46996cd'), 'ProductID': 3, 'ProductName': 'Headphones', 'Category': 'Electronics', 'Price': 50, 'Stock': 50}
{'_id': ObjectId('67eb2b160eaea0bbc46996d2'), 'ProductID': 8, 'ProductName': 'TV', 'Category': 'Electronics', 'Price': 900, 'Stock': 5}
{'_id': ObjectId('67eb2b160eaea0bbc46996d5'), 'ProductID': 11, 'ProductName': 'Laptop', 'Category': 'Electronics', 'Price': 1000, 'Stock': 15}
{'_id': ObjectId('67eb2b160eaea0bbc46996d7'), 'ProductID': 13, 'ProductName': 'Lesor', 'Category': 'Electronics', 'Price': 90, 'Stock': 15}
{'_id': ObjectId('67eb2b160eaea0bbc46996d9'), 'ProductID': 15, 'ProductName': 'Bulb', 'Category': 'Electronics', 'Price': 70, 'Stock': 25}
```

```
Products Sorted by Name:
{'_id': ObjectId('67eb2b160eaea0bbc46996d4'), 'ProductID': 10, 'ProductName': 'Bag', 'Category': 'Accessories', 'Price': 40, 'Stock': 35}
{'_id': ObjectId('67eb2b160eaea0bbc46996d9'), 'ProductID': 15, 'ProductName': 'Bulb', 'Category': 'Electronics', 'Price': 70, 'Stock': 25}
{'_id': ObjectId('67eb2b160eaea0bbc46996cf'), 'ProductID': 5, 'ProductName': 'Chair', 'Category': 'Furniture', 'Price': 80, 'Stock': 30}
{'_id': ObjectId('67eb2b160eaea0bbc46996cd'), 'ProductID': 3, 'ProductName': 'Headphones', 'Category': 'Electronics', 'Price': 50, 'Stock': 50}
{'_id': ObjectId('67eb2b160eaea0bbc46996d8'), 'ProductID': 14, 'ProductName': 'Jacket', 'Category': 'Clothing', 'Price': 400, 'Stock': 12}
{'_id': ObjectId('67eb2b160eaea0bbc46996d6'), 'ProductID': 12, 'ProductName': 'Jeans', 'Category': 'Clothing', 'Price': 500, 'Stock': 12}
{'_id': ObjectId('67eb2b160eaea0bbc46996cb'), 'ProductID': 1, 'ProductName': 'Laptop', 'Category': 'Electronics', 'Price': 800, 'Stock': 10}
{'_id': ObjectId('67eb2b160eaea0bbc46996d5'), 'ProductID': 11, 'ProductName': 'Laptop', 'Category': 'Electronics', 'Price': 1000, 'Stock': 15}
{'_id': ObjectId('67eb2b160eaea0bbc46996d7'), 'ProductID': 13, 'ProductName': 'Lesor', 'Category': 'Electronics', 'Price': 90, 'Stock': 15}
{'_id': ObjectId('67eb2b160eaea0bbc46996d1'), 'ProductID': 7, 'ProductName': 'Shoes', 'Category': 'Clothing', 'Price': 60, 'Stock': 40}
{'_id': ObjectId('67eb2b160eaea0bbc46996cc'), 'ProductID': 2, 'ProductName': 'Smartphone', 'Category': 'Electronics', 'Price': 500, 'Stock': 25}
{'_id': ObjectId('67eb2b160eaea0bbc46996d0'), 'ProductID': 6, 'ProductName': 'T-shirt', 'Category': 'Clothing', 'Price': 20, 'Stock': 100}
{'_id': ObjectId('67eb2b160eaea0bbc46996d2'), 'ProductID': 8, 'ProductName': 'TV', 'Category': 'Electronics', 'Price': 900, 'Stock': 5}
{'_id': ObjectId('67eb2b160eaea0bbc46996ce'), 'ProductID': 4, 'ProductName': 'Table', 'Category': 'Furniture', 'Price': 150, 'Stock': 15}
{'_id': ObjectId('67eb2b160eaea0bbc46996d3'), 'ProductID': 9, 'ProductName': 'Watch', 'Category': 'Accessories', 'Price': 120, 'Stock': 20}
```

```
Category of 'Laptop':
{'_id': ObjectId('67eb2b160eaea0bbc46996cb'), 'ProductID': 1, 'ProductName': 'Laptop', 'Category': 'Electronics', 'Price': 800, 'Stock': 10}
{'_id': ObjectId('67eb2b160eaea0bbc46996d5'), 'ProductID': 11, 'ProductName': 'Laptop', 'Category': 'Electronics', 'Price': 1000, 'Stock': 15}

Category of 'Laptop':
{'ProductName': 'Laptop', 'Category': 'Electronics'}
{'ProductName': 'Laptop', 'Category': 'Electronics'}
```

```
Number of Electronics Products: 7
```

```
 Products Without '_id':
{'ProductID': 1, 'ProductName': 'Laptop', 'Category': 'Electronics', 'Price': 800, 'Stock': 10}
{'ProductID': 2, 'ProductName': 'Smartphone', 'Category': 'Electronics', 'Price': 500, 'Stock': 25}
{'ProductID': 3, 'ProductName': 'Headphones', 'Category': 'Electronics', 'Price': 50, 'Stock': 50}
{'ProductID': 4, 'ProductName': 'Table', 'Category': 'Furniture', 'Price': 150, 'Stock': 15}
{'ProductID': 5, 'ProductName': 'Chair', 'Category': 'Furniture', 'Price': 80, 'Stock': 30}
{'ProductID': 6, 'ProductName': 'T-shirt', 'Category': 'Clothing', 'Price': 20, 'Stock': 100}
{'ProductID': 7, 'ProductName': 'Shoes', 'Category': 'Clothing', 'Price': 60, 'Stock': 40}
{'ProductID': 8, 'ProductName': 'TV', 'Category': 'Electronics', 'Price': 900, 'Stock': 5}
{'ProductID': 9, 'ProductName': 'Watch', 'Category': 'Accessories', 'Price': 120, 'Stock': 20}
{'ProductID': 10, 'ProductName': 'Bag', 'Category': 'Accessories', 'Price': 40, 'Stock': 35}
{'ProductID': 11, 'ProductName': 'Laptop', 'Category': 'Electronics', 'Price': 1000, 'Stock': 15}
{'ProductID': 12, 'ProductName': 'Jeans', 'Category': 'Clothing', 'Price': 500, 'Stock': 12}
{'ProductID': 13, 'ProductName': 'Lesor', 'Category': 'Electronics', 'Price': 90, 'Stock': 15}
{'ProductID': 14, 'ProductName': 'Jacket', 'Category': 'Clothing', 'Price': 400, 'Stock': 12}
{'ProductID': 15, 'ProductName': 'Bulb', 'Category': 'Electronics', 'Price': 70, 'Stock': 25}
```

```
 Distinct Product Categories: ['Accessories', 'Clothing', 'Electronics', 'Furniture']
```

```
 Electronics Products with Price Between 50 and 100:
{'_id': ObjectId('67eb2b160eaea0bbc46996d7'), 'ProductID': 13, 'ProductName': 'Lesor', 'Category': 'Electronics', 'Price': 90, 'Stock': 15}
{'_id': ObjectId('67eb2b160eaea0bbc46996d9'), 'ProductID': 15, 'ProductName': 'Bulb', 'Category': 'Electronics', 'Price': 70, 'Stock': 25}
```

```
 Price of Laptop updated to 850.
{'_id': ObjectId('67eb2b160eaea0bbc46996d2'), 'ProductID': 8, 'ProductName': 'TV', 'Category': 'Electronics', 'Price': 850, 'Stock': 5}
```

```
 Product 'Chair' deleted.
```

Conclusion :

- MongoDB is a NoSQL database that provides high performance, scalability, and flexibility.
- It stores data in documents, grouped into collections, making it schema-less and dynamic.
- Sharding enables horizontal scaling, making MongoDB suitable for large, distributed applications.
- Best suited for big data, cloud apps, and real-time analytics, but not ideal for transactional systems like banking.

MongoDB is a powerful choice for modern, high-scale applications!