Experiment 6 : MongoDB basic operations

Name of Student	**KUNAL PUNJABI**
Class Roll No	43
D.O.P.	
D.O.S.	
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.

Theory:-

1. Features of MongoDB:

MongoDB is a NoSQL, document-oriented database that provides high performance, scalability, and flexibility. Some of its key features include:

- **Document-Oriented Storage:** Data is stored in BSON (Binary JSON) format, making it flexible and easy to use.
- **Schema-less:** Unlike relational databases, MongoDB does not require a fixed schema, allowing dynamic changes to the structure of documents.
- **Scalability:** It supports horizontal scaling using **sharding**, allowing large-scale applications to distribute data efficiently.
- **High Performance:** Indexing, replication, and in-memory processing improve query execution speed.
- **Replication:** Provides **automatic failover** and **data redundancy** using Replica Sets.
- **Aggregation Framework:** MongoDB offers an advanced aggregation pipeline for complex data analysis and transformations.
- **Rich Query Language:** Supports CRUD operations, indexing, text search, and geospatial queries.
- **Load Balancing:** Distributes queries across multiple nodes for better efficiency and reliability.

2. Documents and Collections in MongoDB:

- **Documents:**
  - A document in MongoDB is the basic unit of data storage, similar to a row in a relational database.

○ It is stored in **BSON (Binary JSON)** format and contains key-value pairs. Example

of a document in JSON format:

```
{
  "ProductID": 101,
  "ProductName": "Laptop",
  "Category": "Electronics",
  "Price": 55000,
  "Stock": 30
}
```

- **Collections:**
  - A collection is a group of MongoDB documents, similar to a table in a relational database.
  - Collections do not enforce a fixed schema, allowing flexibility in data storage.
  - Example: A **products** collection may store different types of product documents with varying fields.

3. When to Use MongoDB?

MongoDB is suitable for applications that require:

- **Handling Large Amounts of Unstructured or Semi-Structured Data:** ○ Example: Social media platforms, content management systems.
- **Real-Time Data Processing:**
  - Example: E-commerce websites tracking live user activity.
- **Big Data and High-Throughput Applications:**
  - Example: IoT (Internet of Things) applications, streaming analytics.
- **Flexible Schema Requirements:**
  - Example: Applications where data structure frequently changes, such as product catalogs.
- **Geospatial Data Storage and Processing:**
  - Example: Location-based services and mapping applications.
- **Cloud-Based and Distributed Systems:**
  - Example: Applications requiring horizontal scalability and high availability.

4. What is Sharding in MongoDB?

Sharding is a method used to **distribute large datasets** across multiple servers, improving performance and scalability.

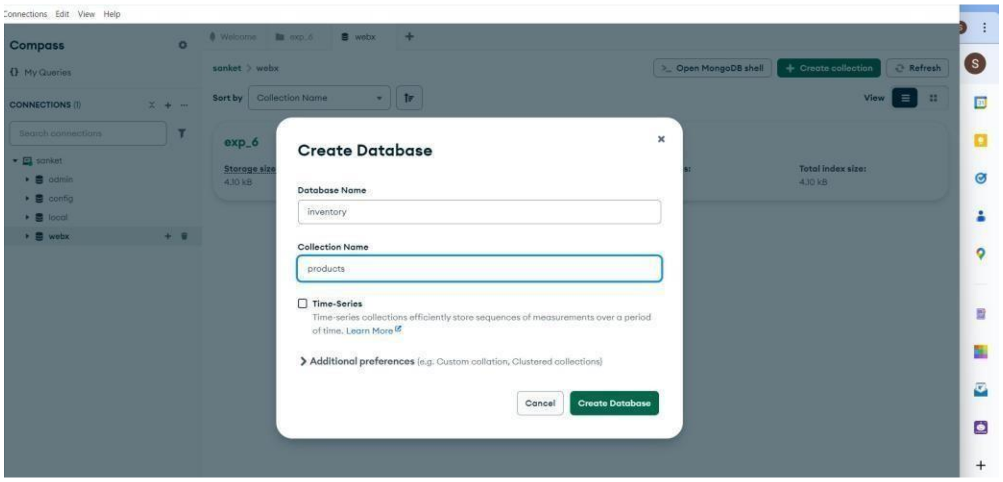
- **Why is Sharding Needed?**
  - When a single server cannot handle large amounts of data or high traffic, sharding helps distribute the load across multiple machines. ● **How Sharding Works?**

  - **Shard Key Selection:** A field is chosen as a shard key to distribute data.
  - **Data Distribution:** Data is partitioned into chunks and distributed among multiple **shards (servers).**
  - **Query Routing:** A **mongos** process directs queries to the correct shard.
  - **Balancing and Replication:** MongoDB ensures data is balanced across shards and replicated for fault tolerance.
- **Example Scenario:**
  - A **large e-commerce website** with millions of products and users may use sharding to distribute product data across multiple servers, ensuring efficient query performance.

Sharding helps MongoDB handle **big data workloads** efficiently by ensuring **high availability, better performance, and fault tolerance.**

OUTPUT:-

Create a Database (`inventory`)



Create Database

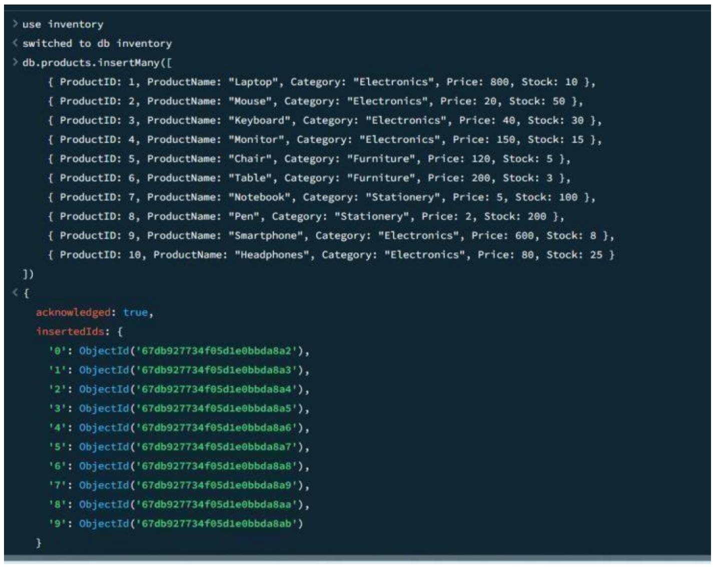
Database Name: inventory

Collection Name: products

Time-Series — Time-series collections efficiently store sequences of measurements over a period of time. Learn More

Additional preferences (e.g. Custom collation, Clustered collections)

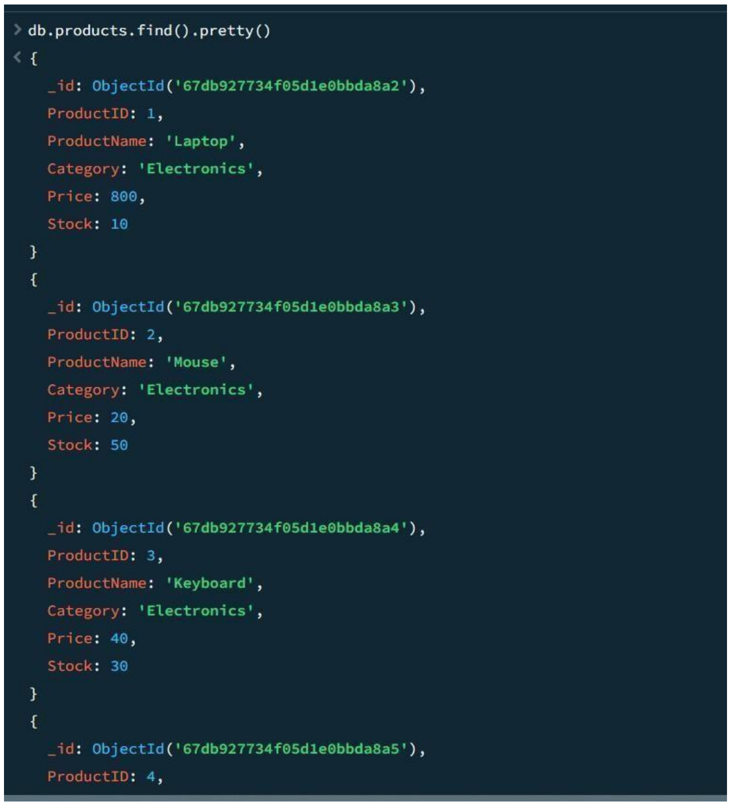
Create a Collection (`products`) & Insert 10 Documents



```
> use inventory
< switched to db inventory
> db.products.insertMany([
      { ProductID: 1, ProductName: "Laptop", Category: "Electronics", Price: 800, Stock: 10 },
      { ProductID: 2, ProductName: "Mouse", Category: "Electronics", Price: 20, Stock: 50 },
      { ProductID: 3, ProductName: "Keyboard", Category: "Electronics", Price: 40, Stock: 30 },
      { ProductID: 4, ProductName: "Monitor", Category: "Electronics", Price: 150, Stock: 15 },
      { ProductID: 5, ProductName: "Chair", Category: "Furniture", Price: 120, Stock: 5 },
      { ProductID: 6, ProductName: "Table", Category: "Furniture", Price: 200, Stock: 3 },
      { ProductID: 7, ProductName: "Notebook", Category: "Stationery", Price: 5, Stock: 100 },
      { ProductID: 8, ProductName: "Pen", Category: "Stationery", Price: 2, Stock: 200 },
      { ProductID: 9, ProductName: "Smartphone", Category: "Electronics", Price: 600, Stock: 8 },
      { ProductID: 10, ProductName: "Headphones", Category: "Electronics", Price: 80, Stock: 25 }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('67db927734f05d1e0bbda8a2'),
      '1': ObjectId('67db927734f05d1e0bbda8a3'),
      '2': ObjectId('67db927734f05d1e0bbda8a4'),
      '3': ObjectId('67db927734f05d1e0bbda8a5'),
      '4': ObjectId('67db927734f05d1e0bbda8a6'),
      '5': ObjectId('67db927734f05d1e0bbda8a7'),
      '6': ObjectId('67db927734f05d1e0bbda8a8'),
      '7': ObjectId('67db927734f05d1e0bbda8a9'),
      '8': ObjectId('67db927734f05d1e0bbda8aa'),
      '9': ObjectId('67db927734f05d1e0bbda8ab')
    }
```

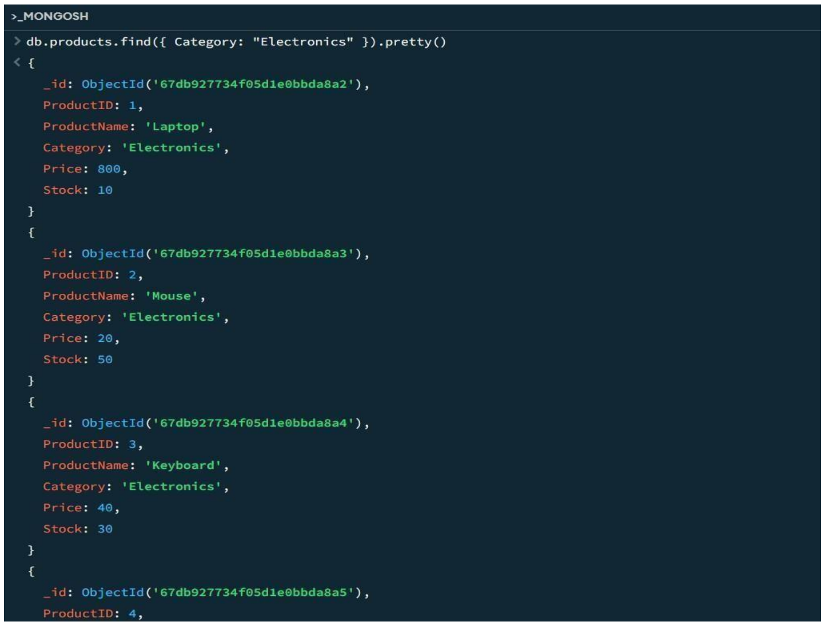
Display

all documents



```
> db.products.find().pretty()
< {
    _id: ObjectId('67db927734f05d1e0bbda8a2'),
    ProductID: 1,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 800,
    Stock: 10
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a3'),
    ProductID: 2,
    ProductName: 'Mouse',
    Category: 'Electronics',
    Price: 20,
    Stock: 50
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a4'),
    ProductID: 3,
    ProductName: 'Keyboard',
    Category: 'Electronics',
    Price: 40,
    Stock: 30
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a5'),
    ProductID: 4,
```

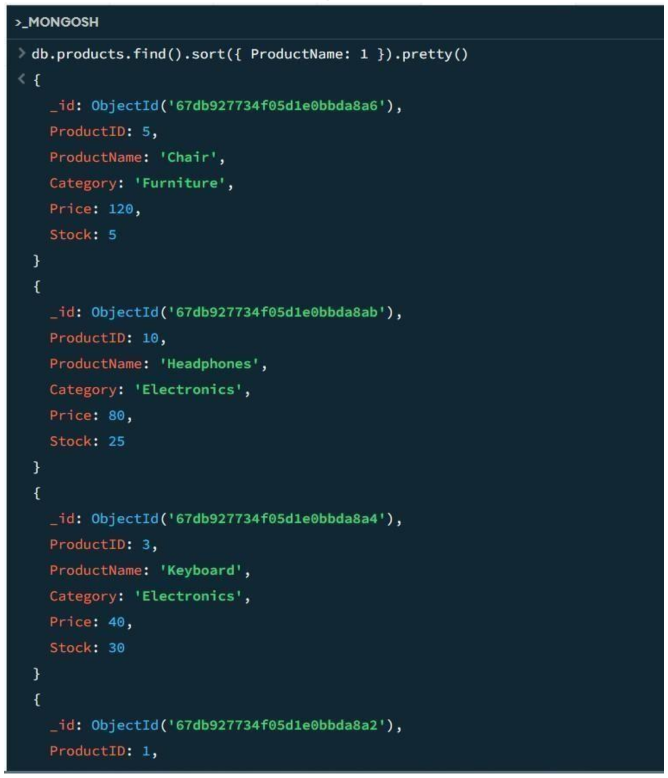
Display all products in the "Electronics" category



```
>_MONGOSH
> db.products.find({ Category: "Electronics" }).pretty()
< {
    _id: ObjectId('67db927734f05d1e0bbda8a2'),
    ProductID: 1,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 800,
    Stock: 10
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a3'),
    ProductID: 2,
    ProductName: 'Mouse',
    Category: 'Electronics',
    Price: 20,
    Stock: 50
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a4'),
    ProductID: 3,
    ProductName: 'Keyboard',
    Category: 'Electronics',
    Price: 40,
    Stock: 30
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a5'),
    ProductID: 4,
```

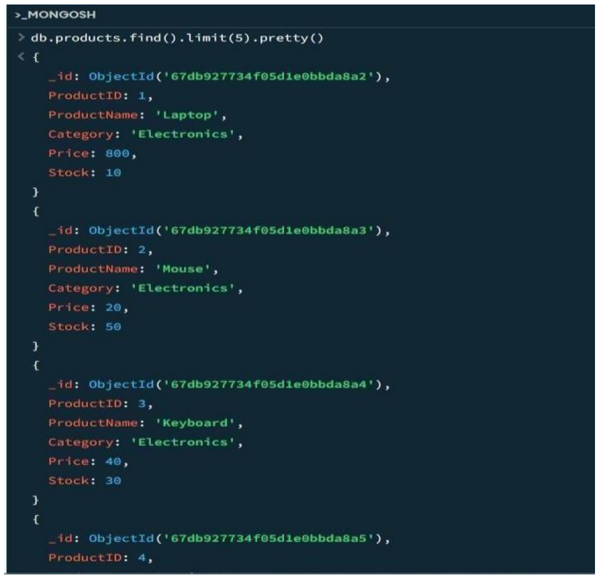
Display

all products in ascending order of their names



```
>_MONGOSH
> db.products.find().sort({ ProductName: 1 }).pretty()
< {
    _id: ObjectId('67db927734f05d1e0bbda8a6'),
    ProductID: 5,
    ProductName: 'Chair',
    Category: 'Furniture',
    Price: 120,
    Stock: 5
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8ab'),
    ProductID: 10,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 80,
    Stock: 25
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a4'),
    ProductID: 3,
    ProductName: 'Keyboard',
    Category: 'Electronics',
    Price: 40,
    Stock: 30
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a2'),
    ProductID: 1,
```

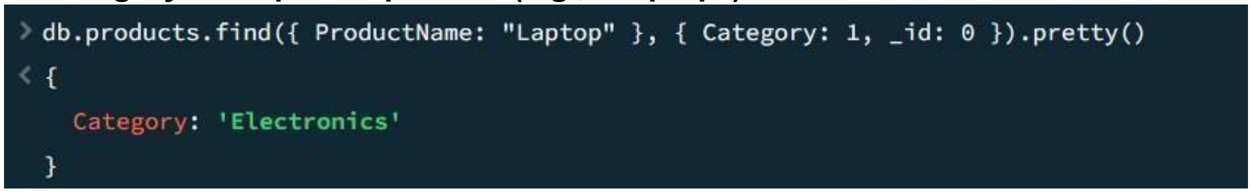
Display the first 5 products



```
>_MONGOSH
> db.products.find().limit(5).pretty()
< {
    _id: ObjectId('67db927734f05d1e0bbda8a2'),
    ProductID: 1,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 800,
    Stock: 10
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a3'),
    ProductID: 2,
    ProductName: 'Mouse',
    Category: 'Electronics',
    Price: 20,
    Stock: 50
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a4'),
    ProductID: 3,
    ProductName: 'Keyboard',
    Category: 'Electronics',
    Price: 40,
    Stock: 30
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a5'),
    ProductID: 4,
```

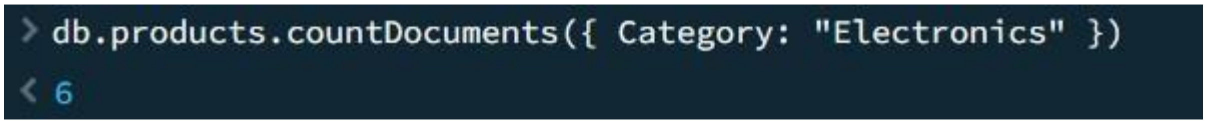
Display

the category of a specific product (e.g., "Laptop")



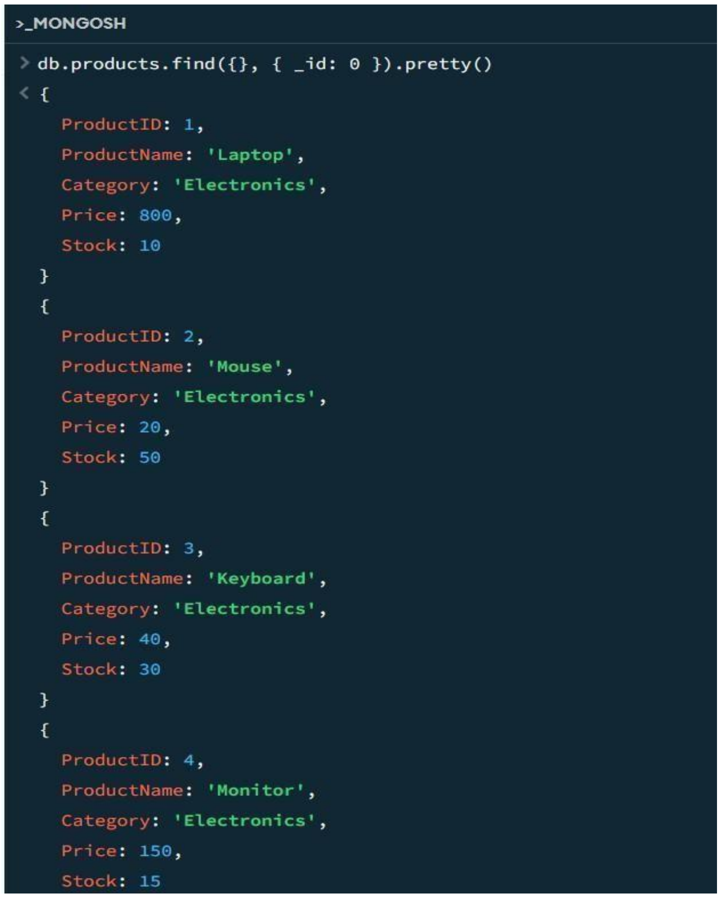
```
> db.products.find({ ProductName: "Laptop" }, { Category: 1, _id: 0 }).pretty()
< {
    Category: 'Electronics'
  }
```

Count the number of products in the "Electronics" category



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

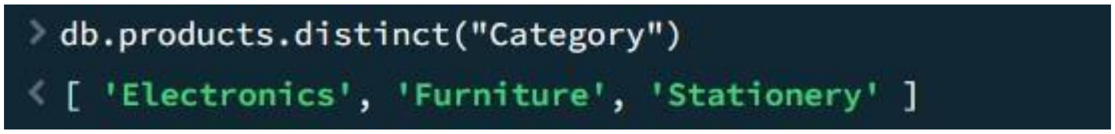
Display all products without the `_id` field



```
> db.products.find({}, { _id: 0 }).pretty()
< {
    ProductID: 1,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 800,
    Stock: 10
  }
  {
    ProductID: 2,
    ProductName: 'Mouse',
    Category: 'Electronics',
    Price: 20,
    Stock: 50
  }
  {
    ProductID: 3,
    ProductName: 'Keyboard',
    Category: 'Electronics',
    Price: 40,
    Stock: 30
  }
  {
    ProductID: 4,
    ProductName: 'Monitor',
    Category: 'Electronics',
    Price: 150,
    Stock: 15
```

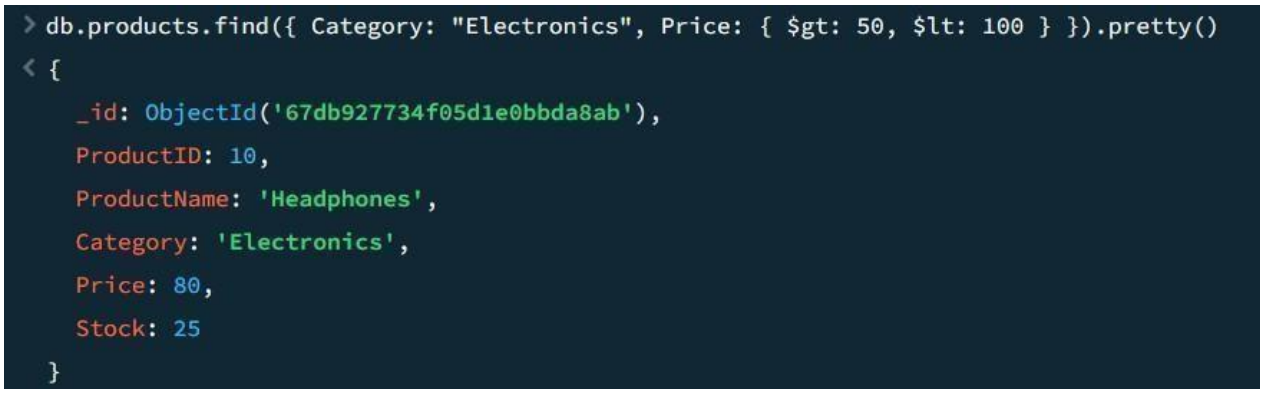
Display

all distinct categories



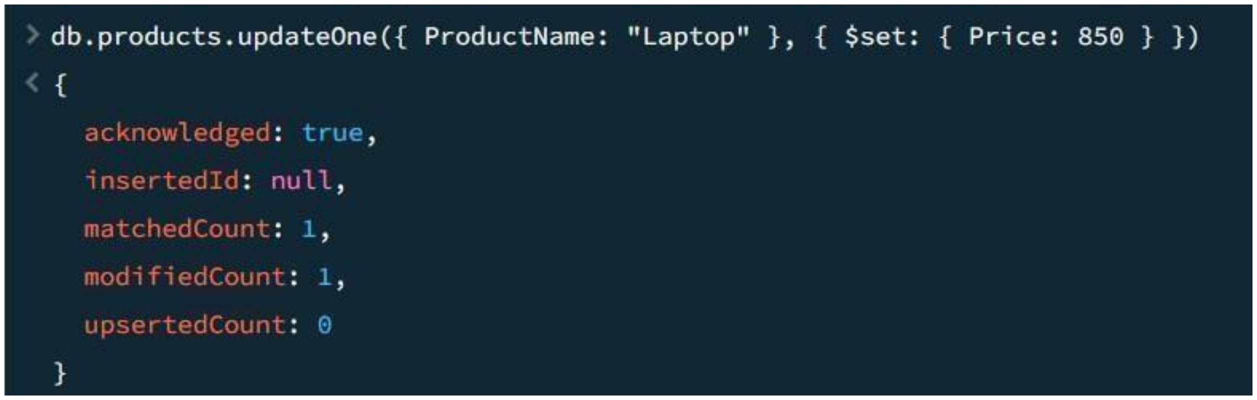
```
> db.products.distinct("Category")
< [ 'Electronics', 'Furniture', 'Stationery' ]
```

Display products in "Electronics" with price between 50 and 100



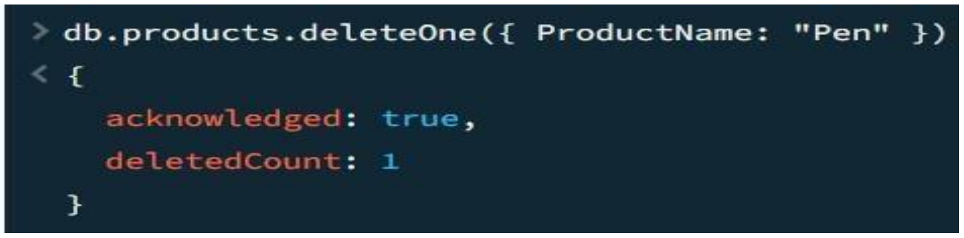
```
> db.products.find({ Category: "Electronics", Price: { $gt: 50, $lt: 100 } }).pretty()
< {
    _id: ObjectId('67db927734f05d1e0bbda8ab'),
    ProductID: 10,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 80,
    Stock: 25
  }
```

Change the price of a product (e.g., Update Laptop price to 850)



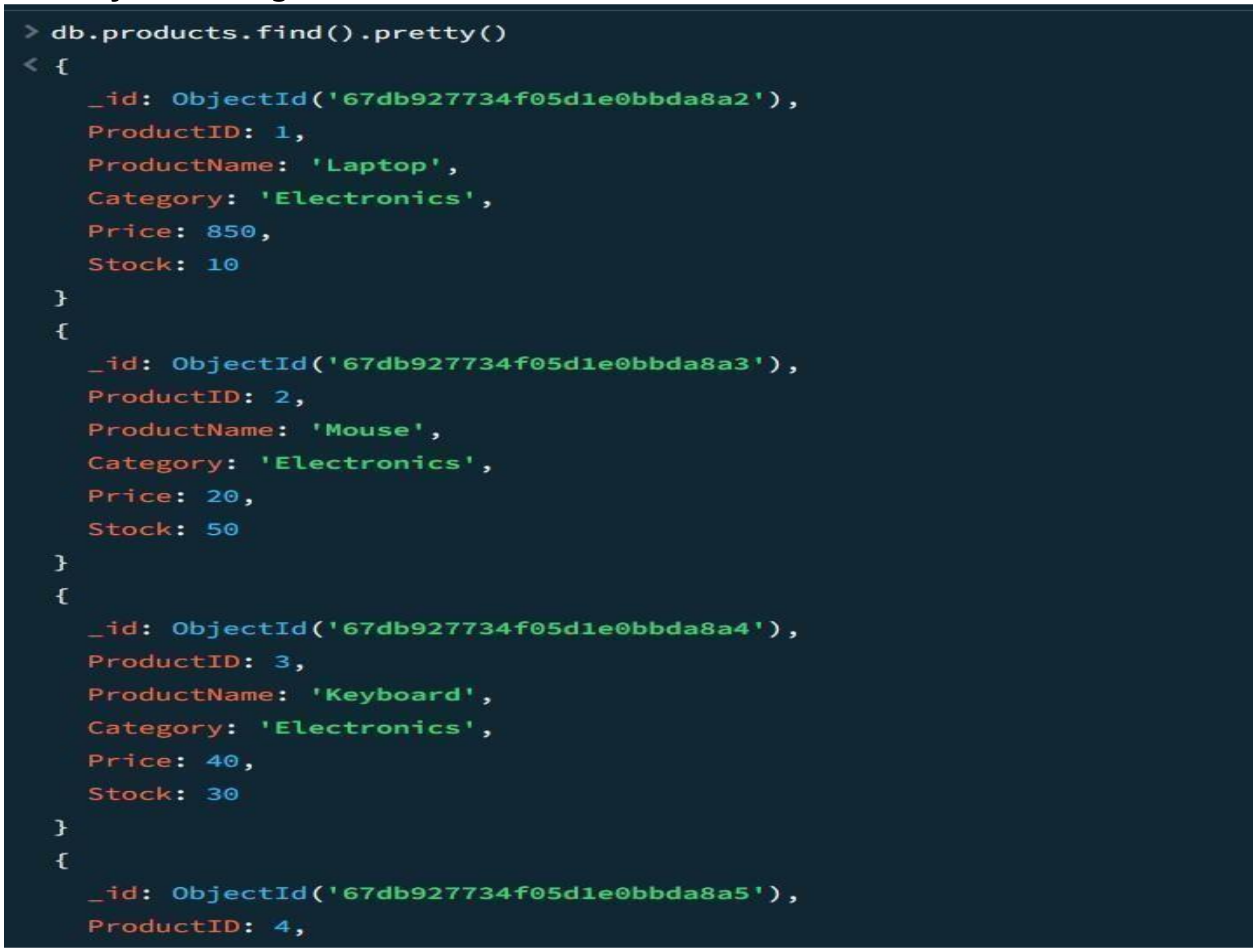
```
> db.products.updateOne({ ProductName: "Laptop" }, { $set: { Price: 850 } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Delete a specific product (e.g., Remove "Pen")



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

Verify the Changes



```
> db.products.find().pretty()
< {
    _id: ObjectId('67db927734f05d1e0bbda8a2'),
    ProductID: 1,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 850,
    Stock: 10
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a3'),
    ProductID: 2,
    ProductName: 'Mouse',
    Category: 'Electronics',
    Price: 20,
    Stock: 50
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a4'),
    ProductID: 3,
    ProductName: 'Keyboard',
    Category: 'Electronics',
    Price: 40,
    Stock: 30
  }
  {
    _id: ObjectId('67db927734f05d1e0bbda8a5'),
    ProductID: 4,
```

Conclusion

In this experiment, we successfully performed CRUD operations in MongoDB. We created a database and collection, inserted multiple documents, and retrieved data using various queries. We also updated and deleted specific entries.
This helped us understand the practical implementation of MongoDB's flexible schema, document-based structure, and powerful querying capabilities.