WebX Exp 6 : MongoDB

<u>Name</u>	Jai Talreja
<u>Class</u>	D15A
<u>Roll No.</u>	59

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

3) **Theory:**

Features of MongoDB:

MongoDB is a widely used NoSQL (Not Only SQL) database that stores data in a flexible, JSON-like format known as BSON (Binary JSON). Some of the key features of MongoDB include:

1. **Schema-less Design**: MongoDB allows for dynamic schema, meaning the structure of a document can change over time. This flexibility makes it a great option for working with data that doesn't fit a rigid schema, such as user-generated content or rapidly evolving application features.

2. **High Performance**: MongoDB provides high throughput for both reads and writes. It is designed to scale easily by distributing data across many servers, providing performance enhancements for large datasets.

3. **Horizontal Scalability**: MongoDB has built-in sharding capabilities, allowing it to horizontally scale across many servers or clusters. This makes it an ideal solution for handling big data or applications with high throughput requirements.

4. **Rich Query Language**: MongoDB supports a rich query language that includes basic CRUD (Create, Read, Update, Delete) operations, as well as more advanced features such as aggregation, text search, and geospatial queries.

5. **Replication**: MongoDB provides replication through a feature called replica sets. A replica set is a group of MongoDB servers that maintain the same data set, ensuring high availability and fault tolerance.

6. **Indexing**: MongoDB supports various types of indexes, including single-field, compound, geospatial, and text indexes, which help in improving query performance.

7. **Aggregation Framework**: MongoDB includes an aggregation framework that allows developers to perform complex data transformations and computations on the data directly within the database, without needing to write additional code in the application layer.

8. **GridFS**: For storing and retrieving large files, MongoDB offers GridFS, a specification that divides large files into smaller chunks and stores them across multiple documents. This is useful for applications that handle large media files like images, audio, and video.

Documents and Collections in MongoDB:

1. **Documents**:

    - A document in MongoDB is a set of key-value pairs, similar to a JSON object. The keys are strings, and the values can be strings, numbers, arrays, or even other documents. Each document is a record in the database and has a unique identifier known as _id.
    - Documents are flexible and schema-less, meaning fields can vary from one document to another within the same collection.

Example of a document in MongoDB:

```
 {
  "_id": 1,
  "name": "Jai Talreja",
  "age": 20,
  "address": {
    "street": "Chembur",
    "city": "Mumbai"
  }
}
```

2. **Collections**:

    - A collection in MongoDB is a group of related documents. Collections are the equivalent of tables in relational databases, but unlike tables, collections do not enforce a schema. They can contain documents with different structures.
    - Collections are typically used to store entities that share a similar nature, like "users," "orders," or "products."
    - MongoDB automatically creates collections when documents are inserted.

When to Use MongoDB:

MongoDB is particularly well-suited for certain types of applications and use cases. You might consider using MongoDB in the following scenarios:

1. **Unstructured or Semi-Structured Data**: If your data doesn't fit neatly into a relational schema or if you expect frequent changes to the structure of your data, MongoDB's schema-less design makes it a good choice.

2. **Big Data Applications**: MongoDB is designed to handle large volumes of data efficiently, making it ideal for big data applications that require fast write and read operations.

3. **Real-Time Analytics**: MongoDB's aggregation framework and indexing capabilities make it a good choice for applications that require real-time analytics and data processing.

4. **Content Management Systems**: Since MongoDB stores documents in JSON-like format, it can handle content management systems (CMS) that deal with varied types of content (e.g., text, images, metadata).

5. **Internet of Things (IoT)**: MongoDB’s ability to handle large amounts of time-series data (e.g., sensor data) and its scalability makes it a strong candidate for IoT applications.

6. **Mobile and Web Applications**: MongoDB’s flexible structure and ability to scale horizontally are advantageous for mobile and web applications with varying user data and high read/write demands.

Sharding in MongoDB:

Sharding in MongoDB refers to the process of distributing data across multiple servers to ensure horizontal scalability and high availability. This is particularly useful when dealing with large datasets that cannot fit on a single machine. Sharding helps MongoDB handle workloads that would otherwise overwhelm a single server by splitting data into smaller, more manageable chunks and distributing these chunks across multiple servers.

- **Sharding Key**: MongoDB uses a "sharding key" to determine how data is distributed across the shards. The choice of sharding key is crucial for performance and data distribution.
- **Chunks**: MongoDB divides data into chunks based on the value of the sharding key. These chunks are distributed across different shards in the cluster.
- **Shard**: A shard is a single MongoDB instance (or replica set) that holds a subset of the data.
- **Balancer**: MongoDB has a built-in balancer that redistributes chunks across shards to ensure data is evenly distributed.
- **Advantages**:
  - **Scalability**: Sharding allows MongoDB to scale horizontally across multiple servers.
  - **High Availability**: With sharded clusters, MongoDB can handle large-scale operations while maintaining redundancy through replica sets.
  - **Improved Performance**: By distributing the workload, sharding can enhance the performance of queries, writes, and data processing tasks.

Sharding is typically employed when the data grows beyond the capacity of a single machine, and when the system needs to handle large traffic or large datasets across distributed environments.

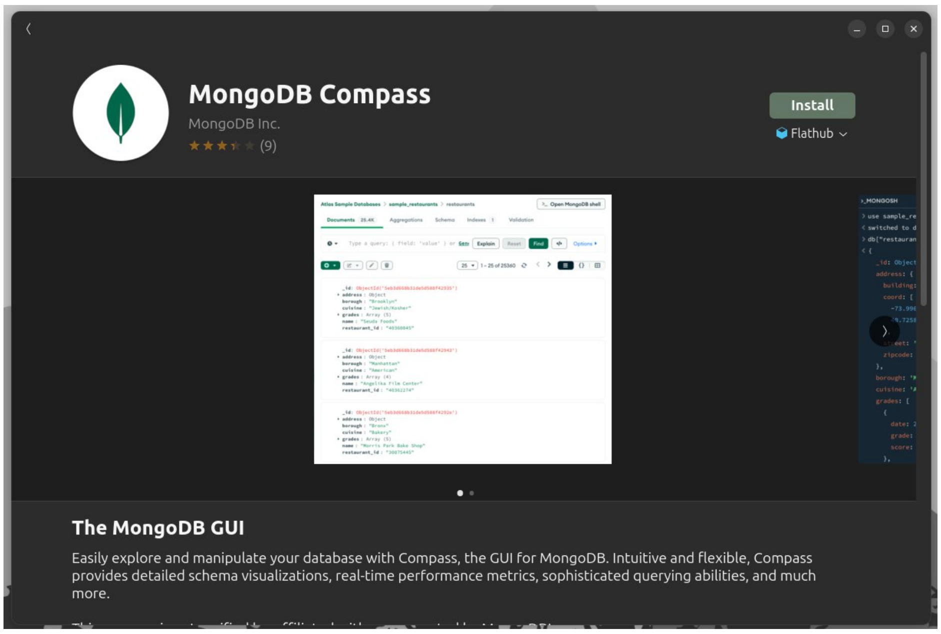
Output :

MongoDB Installation

This step was performed on a Linux distro, Ubuntu 24, with Flatpak installed and configured. Flatpak, is a packaging format and package manager, that makes installation of software very easy, on Linux (it only exists on Linux).

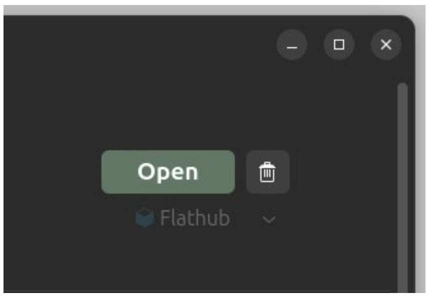
Flatpak apps are containerized and come pre-packaged with their dependencies so the user does not need to manage the dependencies and versions compatibility.

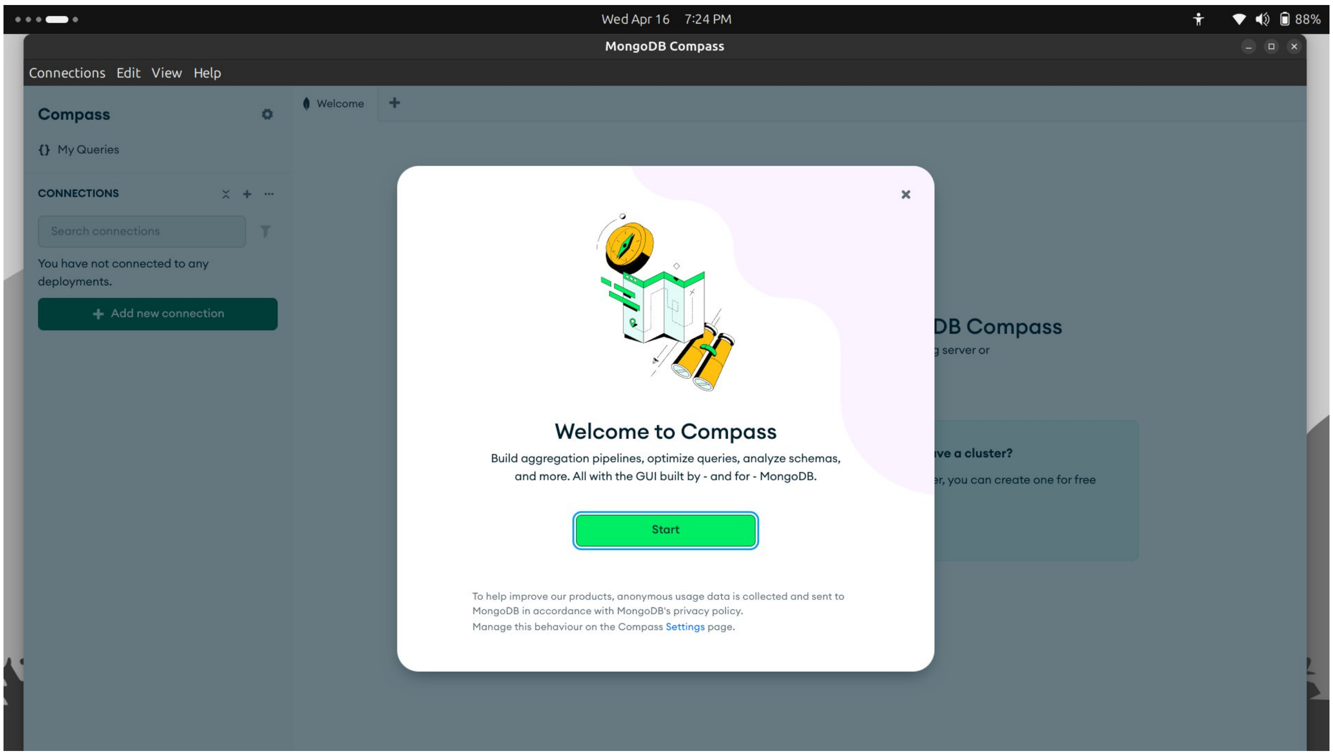
Also, Flatpaks can be installed on any Linux system with the Flatpak software manager installed and configured.



The MongoDB GUI

Easily explore and manipulate your database with Compass, the GUI for MongoDB. Intuitive and flexible, Compass provides detailed schema visualizations, real-time performance metrics, sophisticated querying abilities, and much more.




Wed Apr 16 7:24 PM

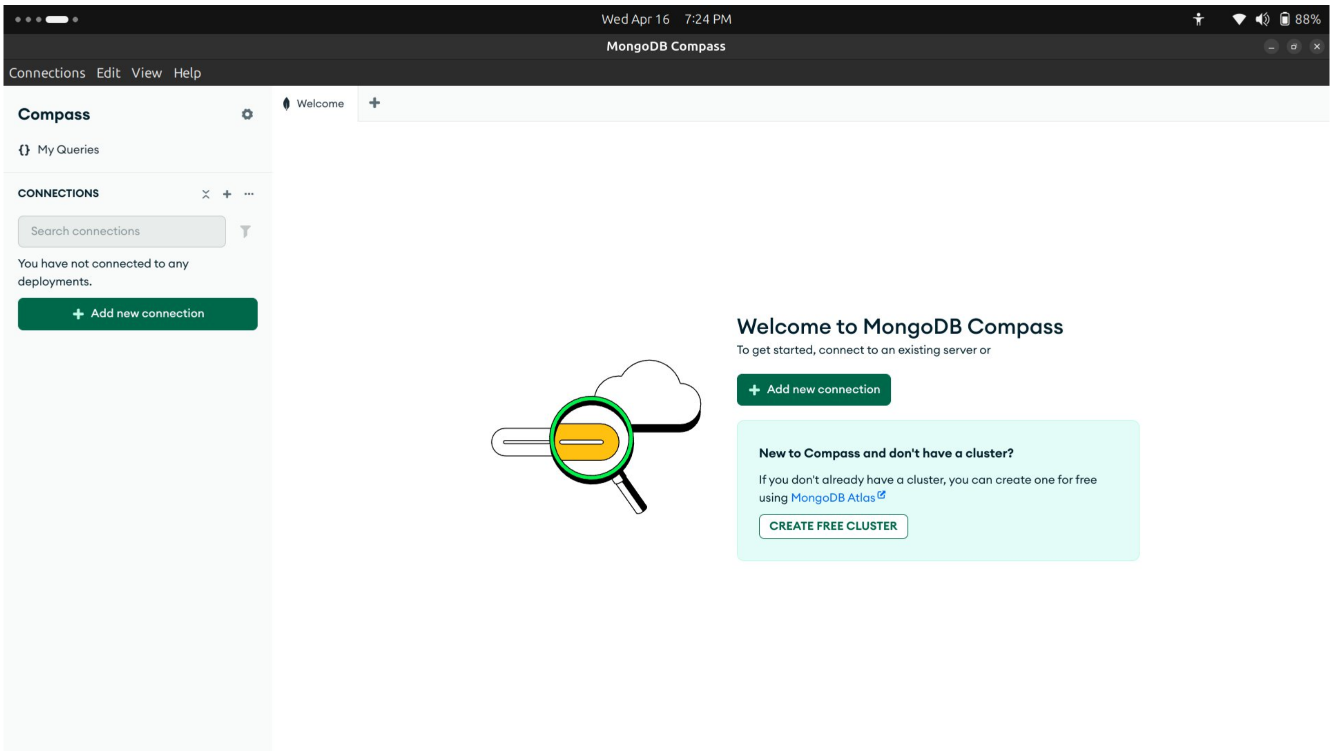
MongoDB Compass

You have not connected to any deployments.

Welcome to Compass

Build aggregation pipelines, optimize queries, analyze schemas, and more. All with the GUI built by - and for - MongoDB.

To help improve our products, anonymous usage data is collected and sent to MongoDB in accordance with MongoDB's privacy policy.
Manage this behaviour on the Compass Settings page.


Wed Apr 16 7:24 PM

MongoDB Compass

You have not connected to any deployments.

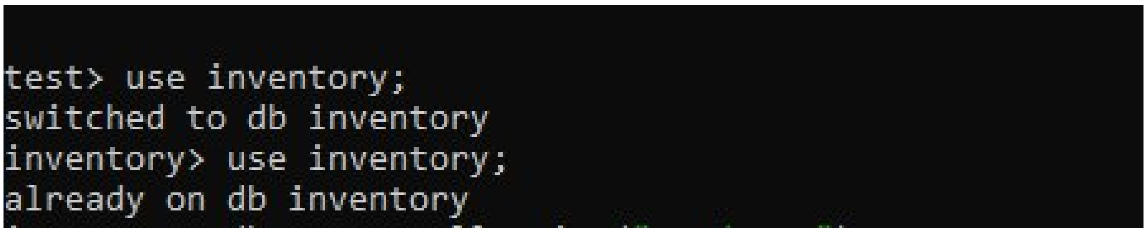
Welcome to MongoDB Compass

To get started, connect to an existing server or

New to Compass and don't have a cluster?

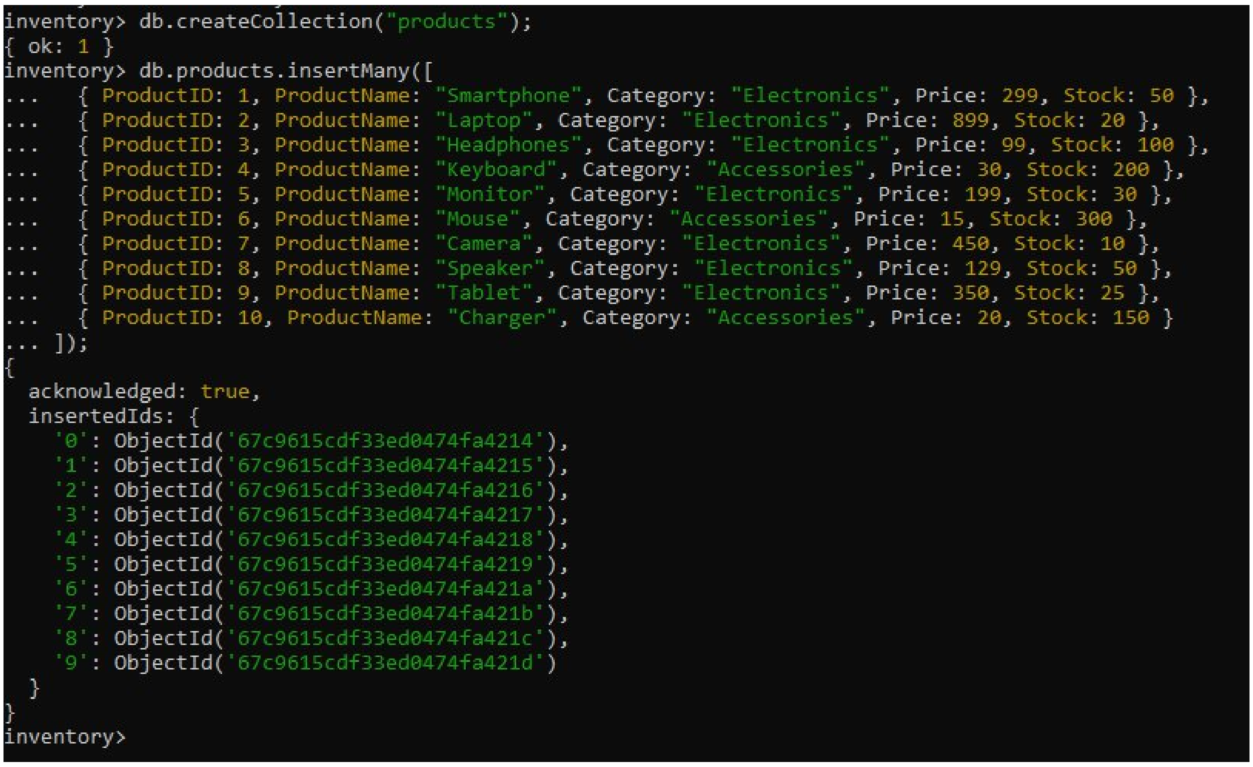
If you don't already have a cluster, you can create one for free using MongoDB Atlas

Create a database named "inventory".



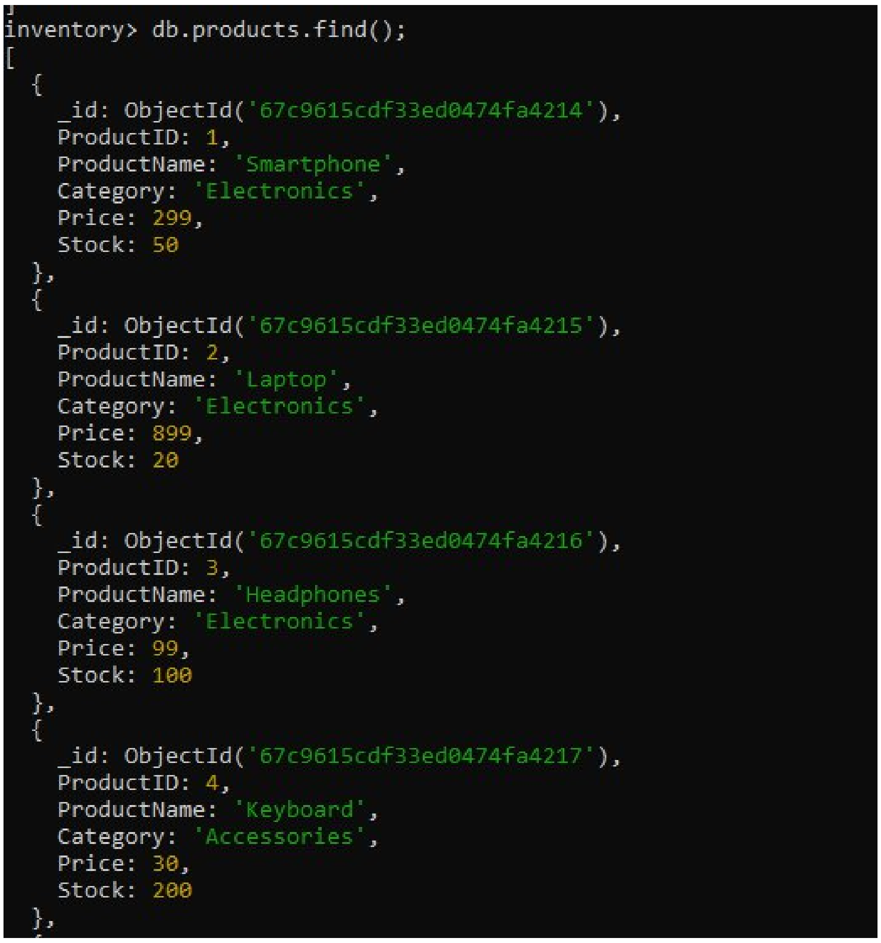
```
test> use inventory;
switched to db inventory
inventory> use inventory;
already on db inventory
```

Create a collection named "products" with the fields: (ProductID, ProductName, Category, Price, Stock).
Insert 10 documents into the "products" collection.

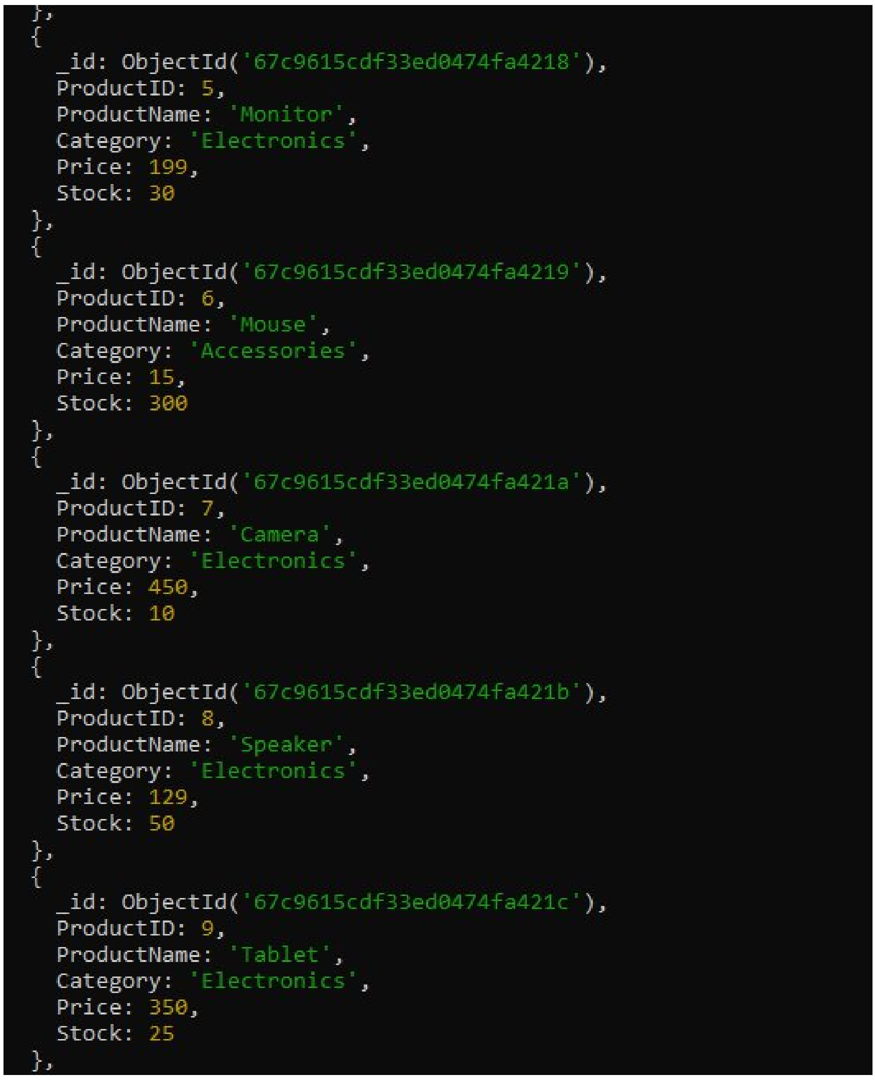


```
inventory> db.createCollection("products");
{ ok: 1 }
inventory> db.products.insertMany([
...   { ProductID: 1, ProductName: "Smartphone", Category: "Electronics", Price: 299, Stock: 50 },
...   { ProductID: 2, ProductName: "Laptop", Category: "Electronics", Price: 899, Stock: 20 },
...   { ProductID: 3, ProductName: "Headphones", Category: "Electronics", Price: 99, Stock: 100 },
...   { ProductID: 4, ProductName: "Keyboard", Category: "Accessories", Price: 30, Stock: 200 },
...   { ProductID: 5, ProductName: "Monitor", Category: "Electronics", Price: 199, Stock: 30 },
...   { ProductID: 6, ProductName: "Mouse", Category: "Accessories", Price: 15, Stock: 300 },
...   { ProductID: 7, ProductName: "Camera", Category: "Electronics", Price: 450, Stock: 10 },
...   { ProductID: 8, ProductName: "Speaker", Category: "Electronics", Price: 129, Stock: 50 },
...   { ProductID: 9, ProductName: "Tablet", Category: "Electronics", Price: 350, Stock: 25 },
...   { ProductID: 10, ProductName: "Charger", Category: "Accessories", Price: 20, Stock: 150 }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67c9615cdf33ed0474fa4214'),
    '1': ObjectId('67c9615cdf33ed0474fa4215'),
    '2': ObjectId('67c9615cdf33ed0474fa4216'),
    '3': ObjectId('67c9615cdf33ed0474fa4217'),
    '4': ObjectId('67c9615cdf33ed0474fa4218'),
    '5': ObjectId('67c9615cdf33ed0474fa4219'),
    '6': ObjectId('67c9615cdf33ed0474fa421a'),
    '7': ObjectId('67c9615cdf33ed0474fa421b'),
    '8': ObjectId('67c9615cdf33ed0474fa421c'),
    '9': ObjectId('67c9615cdf33ed0474fa421d')
  }
}
inventory>
```

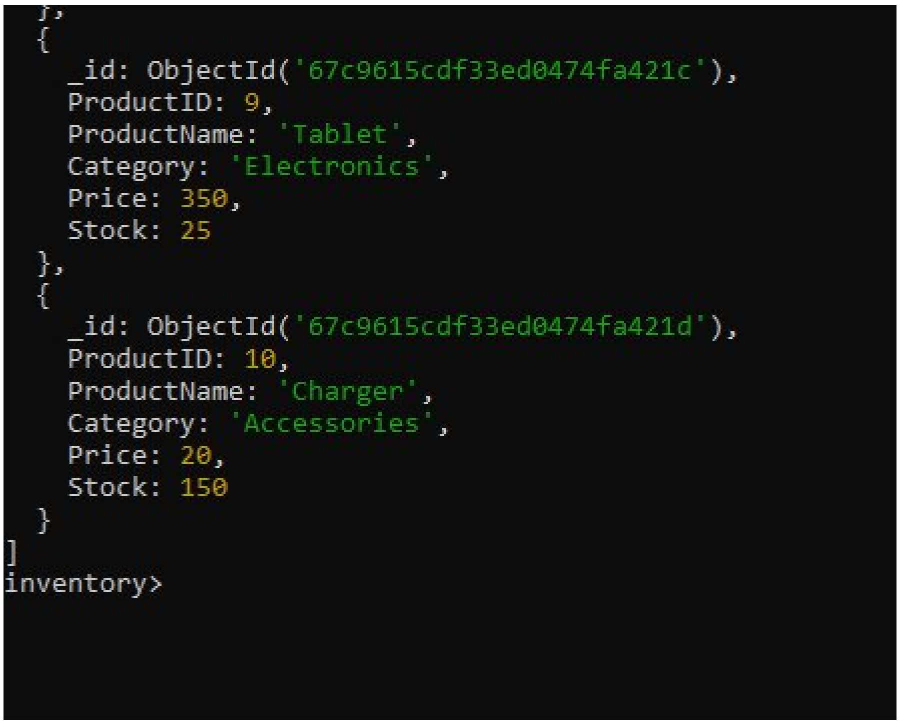
Display all the documents in the "products" collection.



```
inventory> db.products.find();
[
  {
    _id: ObjectId('67c9615cdf33ed0474fa4214'),
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 50
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4215'),
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 899,
    Stock: 20
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4216'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 99,
    Stock: 100
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4217'),
    ProductID: 4,
    ProductName: 'Keyboard',
    Category: 'Accessories',
    Price: 30,
    Stock: 200
  },
```

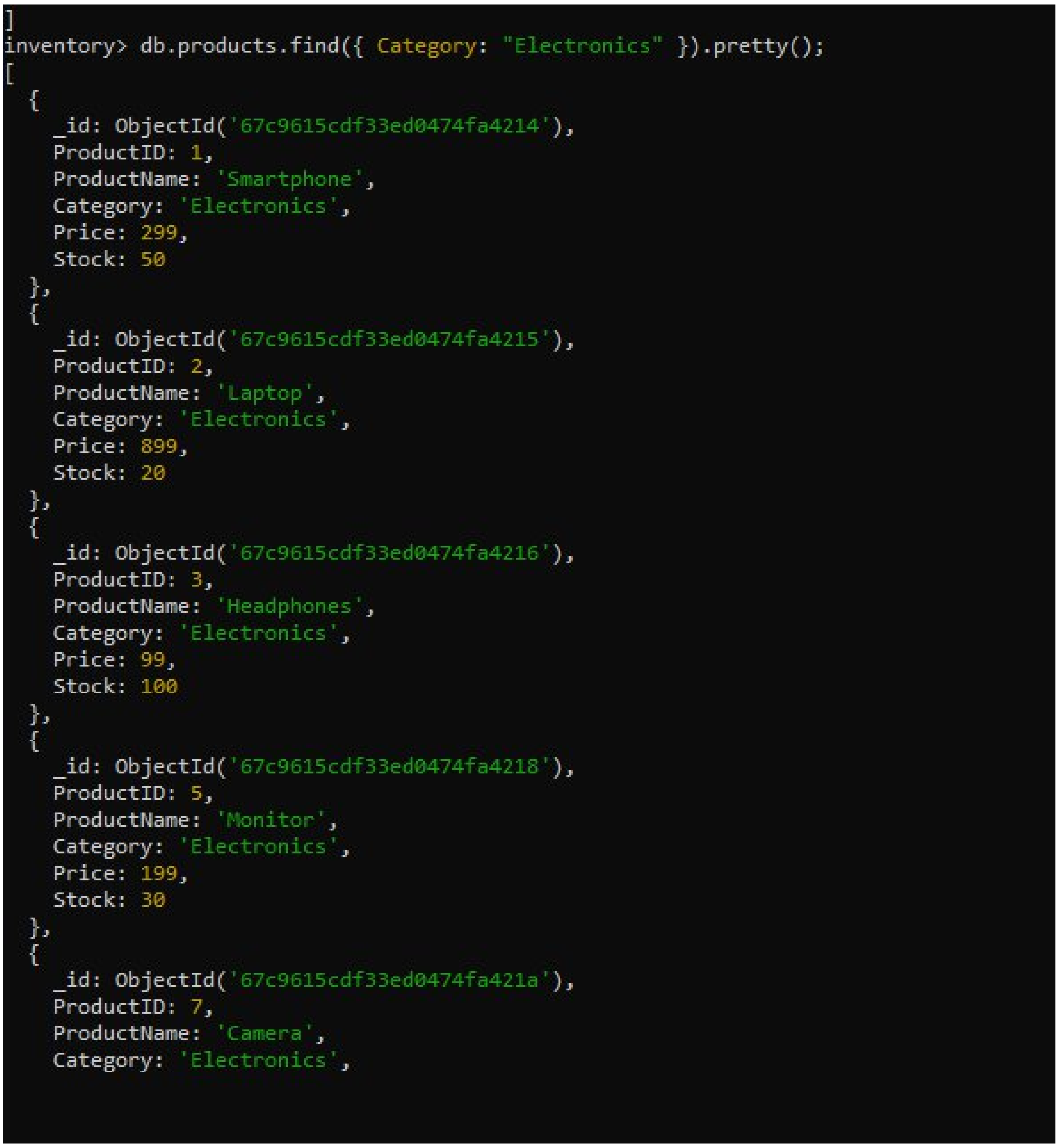


```
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4218'),
    ProductID: 5,
    ProductName: 'Monitor',
    Category: 'Electronics',
    Price: 199,
    Stock: 30
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4219'),
    ProductID: 6,
    ProductName: 'Mouse',
    Category: 'Accessories',
    Price: 15,
    Stock: 300
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa421a'),
    ProductID: 7,
    ProductName: 'Camera',
    Category: 'Electronics',
    Price: 450,
    Stock: 10
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa421b'),
    ProductID: 8,
    ProductName: 'Speaker',
    Category: 'Electronics',
    Price: 129,
    Stock: 50
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa421c'),
    ProductID: 9,
    ProductName: 'Tablet',
    Category: 'Electronics',
    Price: 350,
    Stock: 25
  },
```



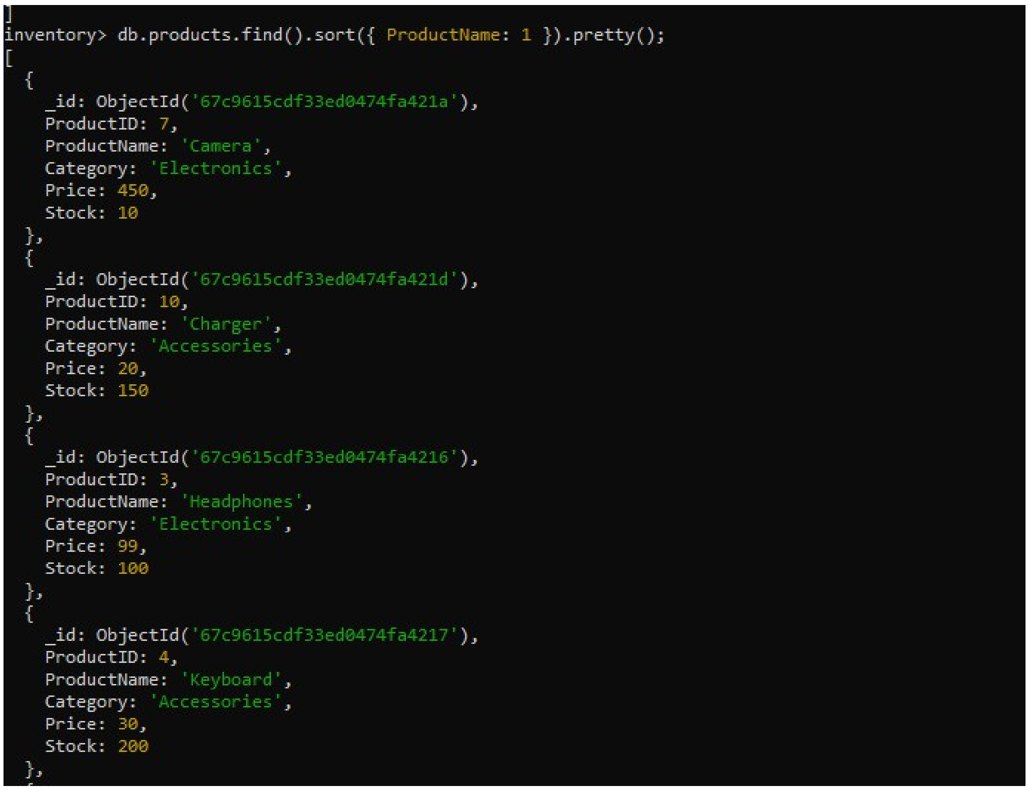
```
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa421c'),
    ProductID: 9,
    ProductName: 'Tablet',
    Category: 'Electronics',
    Price: 350,
    Stock: 25
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa421d'),
    ProductID: 10,
    ProductName: 'Charger',
    Category: 'Accessories',
    Price: 20,
    Stock: 150
  }
]
inventory>
```

Display all the products in the "Electronics" category.

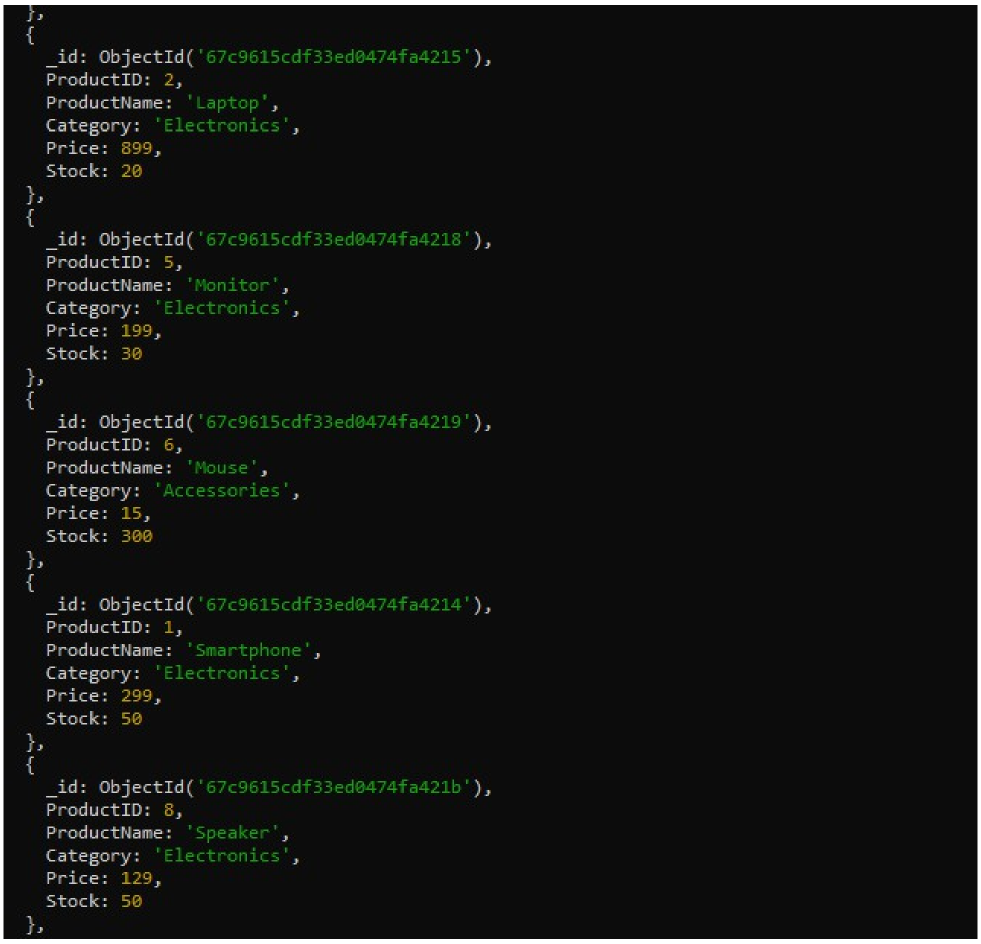


```
]
inventory> db.products.find({ Category: "Electronics" }).pretty();
[
  {
    _id: ObjectId('67c9615cdf33ed0474fa4214'),
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 50
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4215'),
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 899,
    Stock: 20
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4216'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 99,
    Stock: 100
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4218'),
    ProductID: 5,
    ProductName: 'Monitor',
    Category: 'Electronics',
    Price: 199,
    Stock: 30
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa421a'),
    ProductID: 7,
    ProductName: 'Camera',
    Category: 'Electronics',
```

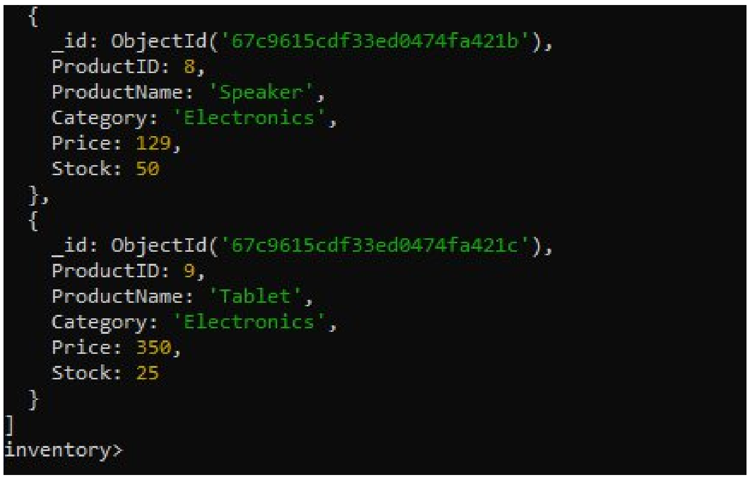
Display all the products in ascending order of their names.



```
]
inventory> db.products.find().sort({ ProductName: 1 }).pretty();
[
  {
    _id: ObjectId('67c9615cdf33ed0474fa421a'),
    ProductID: 7,
    ProductName: 'Camera',
    Category: 'Electronics',
    Price: 450,
    Stock: 10
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa421d'),
    ProductID: 10,
    ProductName: 'Charger',
    Category: 'Accessories',
    Price: 20,
    Stock: 150
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4216'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 99,
    Stock: 100
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4217'),
    ProductID: 4,
    ProductName: 'Keyboard',
    Category: 'Accessories',
    Price: 30,
    Stock: 200
  },
```

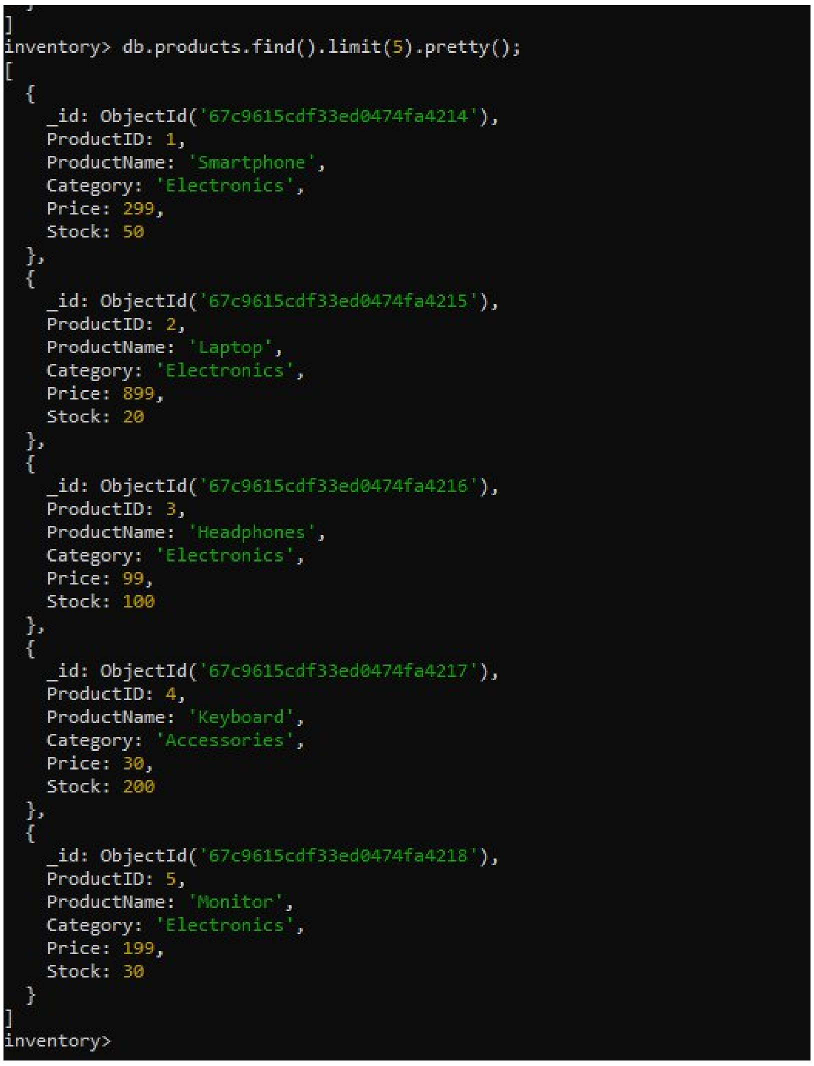


```
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4215'),
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 899,
    Stock: 20
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4218'),
    ProductID: 5,
    ProductName: 'Monitor',
    Category: 'Electronics',
    Price: 199,
    Stock: 30
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4219'),
    ProductID: 6,
    ProductName: 'Mouse',
    Category: 'Accessories',
    Price: 15,
    Stock: 300
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4214'),
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 50
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa421b'),
    ProductID: 8,
    ProductName: 'Speaker',
    Category: 'Electronics',
    Price: 129,
    Stock: 50
  },
```



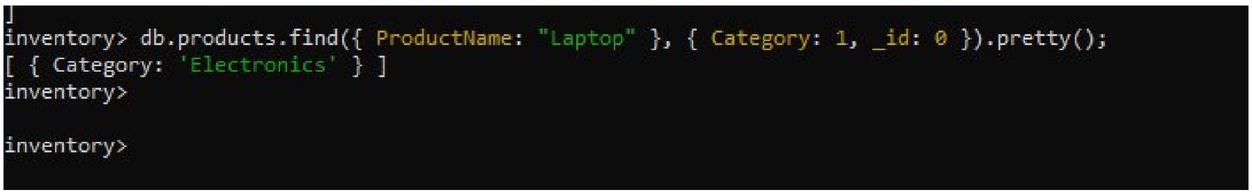
```
  {
    _id: ObjectId('67c9615cdf33ed0474fa421b'),
    ProductID: 8,
    ProductName: 'Speaker',
    Category: 'Electronics',
    Price: 129,
    Stock: 50
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa421c'),
    ProductID: 9,
    ProductName: 'Tablet',
    Category: 'Electronics',
    Price: 350,
    Stock: 25
  }
]
inventory>
```

Display the details of the first 5 products.



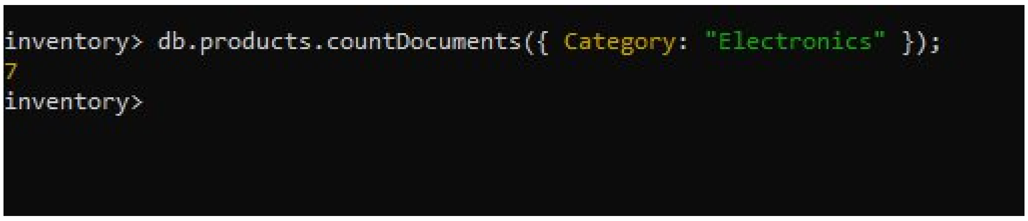
```
]
inventory> db.products.find().limit(5).pretty();
[
  {
    _id: ObjectId('67c9615cdf33ed0474fa4214'),
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 50
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4215'),
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 899,
    Stock: 20
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4216'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 99,
    Stock: 100
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4217'),
    ProductID: 4,
    ProductName: 'Keyboard',
    Category: 'Accessories',
    Price: 30,
    Stock: 200
  },
  {
    _id: ObjectId('67c9615cdf33ed0474fa4218'),
    ProductID: 5,
    ProductName: 'Monitor',
    Category: 'Electronics',
    Price: 199,
    Stock: 30
  }
]
inventory>
```

Display the categories of products with a specific name.



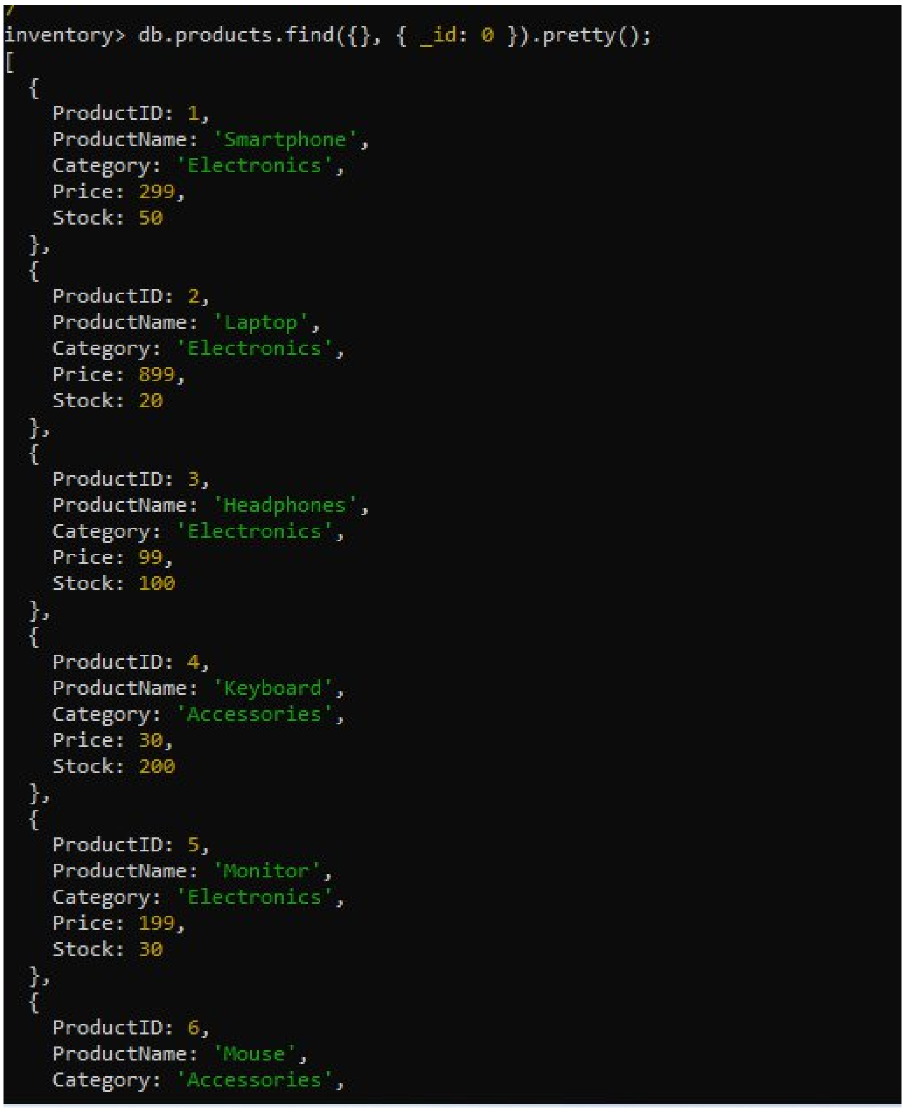
```
]
inventory> db.products.find({ ProductName: "Laptop" }, { Category: 1, _id: 0 }).pretty();
[ { Category: 'Electronics' } ]
inventory>

inventory>
```

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

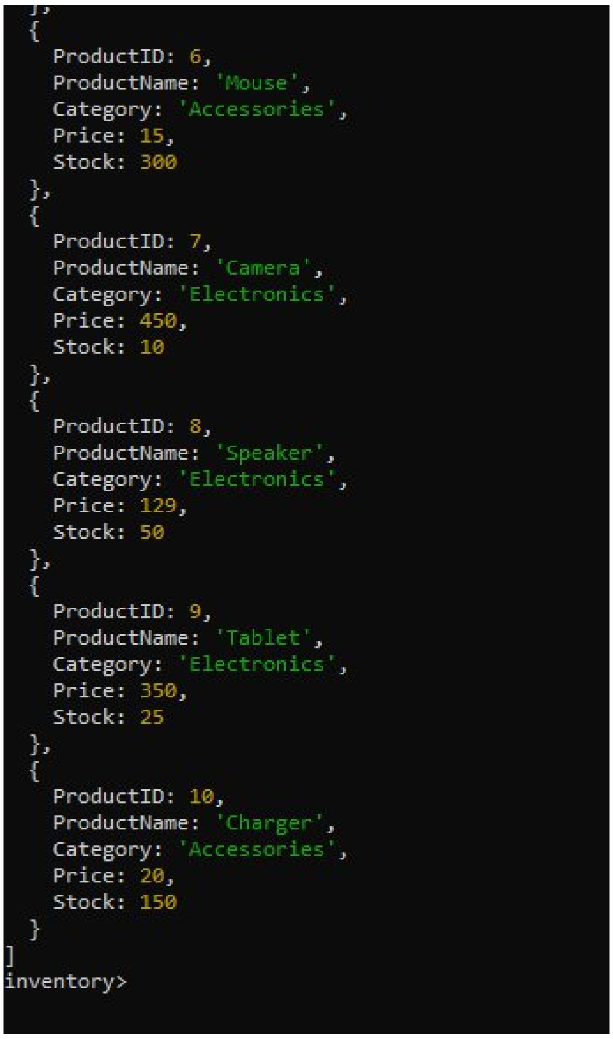


```
inventory> db.products.countDocuments({ Category: "Electronics" });
7
inventory>
```

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

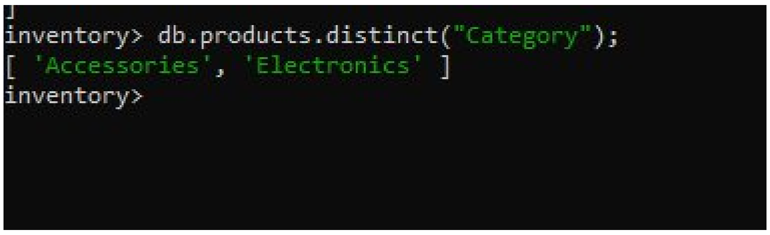


```
inventory> db.products.find({}, { _id: 0 }).pretty();
[
  {
    ProductID: 1,
    ProductName: 'Smartphone',
    Category: 'Electronics',
    Price: 299,
    Stock: 50
  },
  {
    ProductID: 2,
    ProductName: 'Laptop',
    Category: 'Electronics',
    Price: 899,
    Stock: 20
  },
  {
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 99,
    Stock: 100
  },
  {
    ProductID: 4,
    ProductName: 'Keyboard',
    Category: 'Accessories',
    Price: 30,
    Stock: 200
  },
  {
    ProductID: 5,
    ProductName: 'Monitor',
    Category: 'Electronics',
    Price: 199,
    Stock: 30
  },
  {
    ProductID: 6,
    ProductName: 'Mouse',
    Category: 'Accessories',
```



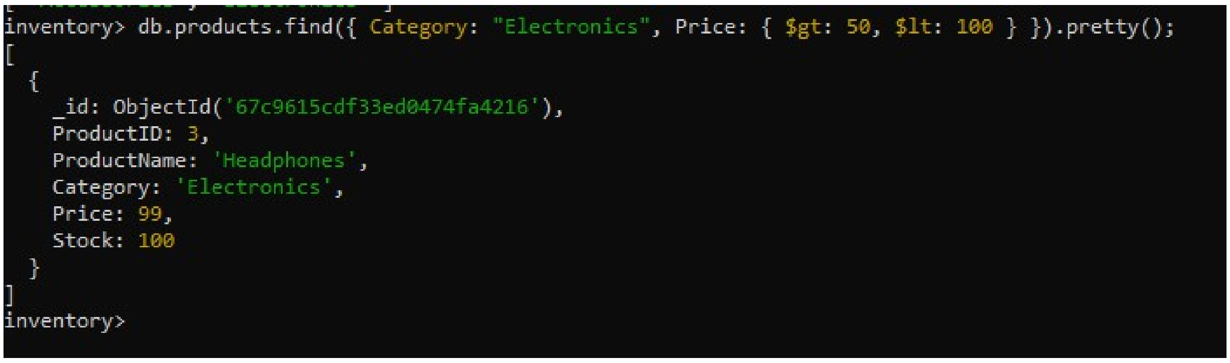
```
  },
  {
    ProductID: 6,
    ProductName: 'Mouse',
    Category: 'Accessories',
    Price: 15,
    Stock: 300
  },
  {
    ProductID: 7,
    ProductName: 'Camera',
    Category: 'Electronics',
    Price: 450,
    Stock: 10
  },
  {
    ProductID: 8,
    ProductName: 'Speaker',
    Category: 'Electronics',
    Price: 129,
    Stock: 50
  },
  {
    ProductID: 9,
    ProductName: 'Tablet',
    Category: 'Electronics',
    Price: 350,
    Stock: 25
  },
  {
    ProductID: 10,
    ProductName: 'Charger',
    Category: 'Accessories',
    Price: 20,
    Stock: 150
  }
]
inventory>
```

Display all the distinct categories of products.



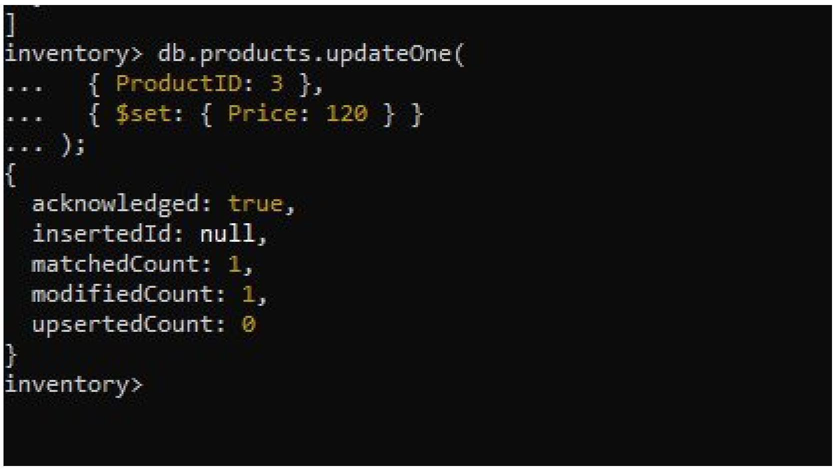
```
]
inventory> db.products.distinct("Category");
[ 'Accessories', 'Electronics' ]
inventory>
```

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



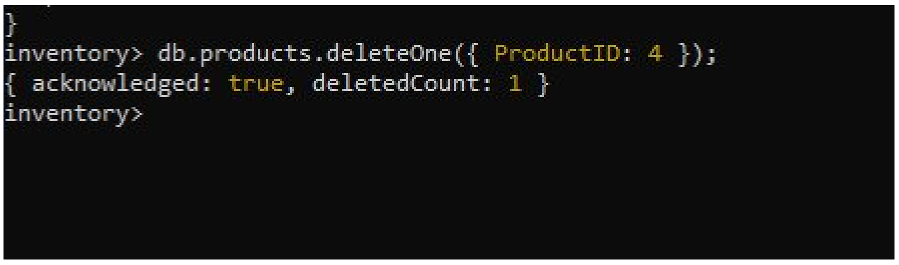
```
inventory> db.products.find({ Category: "Electronics", Price: { $gt: 50, $lt: 100 } }).pretty();
[
  {
    _id: ObjectId('67c9615cdf33ed0474fa4216'),
    ProductID: 3,
    ProductName: 'Headphones',
    Category: 'Electronics',
    Price: 99,
    Stock: 100
  }
]
inventory>
```

Change the price of a product.



```
]
inventory> db.products.updateOne(
...   { ProductID: 3 },
...   { $set: { Price: 120 } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
inventory>
```

Delete a particular product entry.



```
}
inventory> db.products.deleteOne({ ProductID: 4 });
{ acknowledged: true, deletedCount: 1 }
inventory>
```