EXPERIMENT NO. 6 - MongoDB

Name of Student	**Sneha Patra**
Class Roll No	**D15A_40**
D.O.P.	**27/02/2025**
D.O.S.	**06/03/2025**
Sign and Grade	

AIM: To study CRUD operations in MongoDB

PROBLEM STATEMENT:

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

a. Create a database named "inventory".

b. Create a collection named "products" with the fields: (ProductID, ProductName, Category, Price, Stock).

c. Insert 10 documents into the "products" collection.

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

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

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

g. Display the details of the first 5 products.

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

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

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

k. Display all the distinct categories of products.

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

m. Change the price of a product.

n. Delete a particular product entry.

THEORY:

1. Describe some of the features of MongoDB?

   - **Document-Oriented:** Stores data as flexible, JSON-like documents (BSON).
   - **Flexible Schema:** No fixed structure, supports dynamic data.
   - **Horizontal Scalability:** Uses sharding to manage large datasets.
   - **Replication:** Ensures high availability with replica sets.
   - **Indexing:** Supports various indexes for faster query execution.
   - **Aggregation Framework:** Provides powerful data processing using pipelines.
   - **Ad-hoc Queries:** Enables complex queries with ease.

2. **What are Documents and Collections in MongoDB?**

   Documents: JSON-like records storing data in key-value pairs. Example:

   ```
   {
   "_id": "101",
   "name": "Alice", "age": 28,
   "email": "alice@example.com"
   }
   ```

   Collections: A group of documents, equivalent to tables in relational databases. They don't enforce strict schemas, allowing flexibility.

3. **When to use MongoDB?**

   - **Big Data Applications:** Efficient for large, unstructured data.
   - **E-commerce Platforms:** Ideal for product catalogs with dynamic attributes.
   - **Content Management Systems (CMS):** Supports frequent changes in data models.
   - **Real-Time Analytics:** Processes and analyzes data rapidly.
   - **IoT and Mobile Apps:** Manages sensor data and app data effectively.

● **Social Networks:** Scales well for user-generated content.

4. **What is Sharding in MongoDB?**

Sharding: Distributes data across multiple servers to handle large datasets.

Shard Key: A field in documents used to split data across shards.

Components:

- Shards: Store actual data.
- Config Servers: Maintain metadata and sharding configuration.
- Mongos: Routes queries to the appropriate shards.

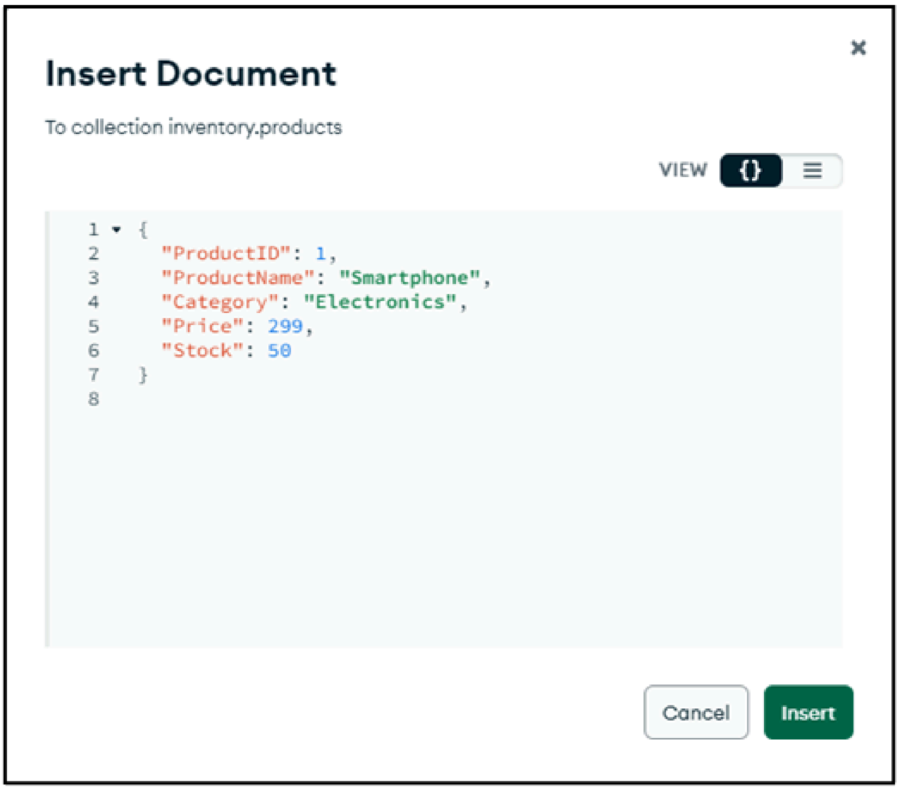
Benefits:

- Supports large-scale data management.
- Improves read and write performance.
- Ensures fault tolerance and high availability.

OUTPUT:

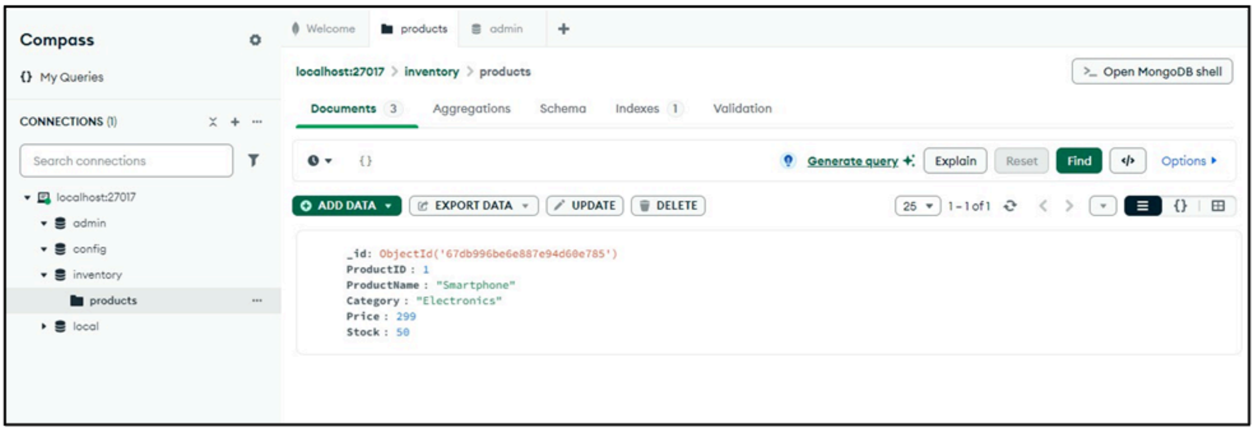
Insert Data (Create Operation)

1. Open your inventory collection.
2. Click "Insert Document" (top-right).


Insert Document

To collection inventory.products

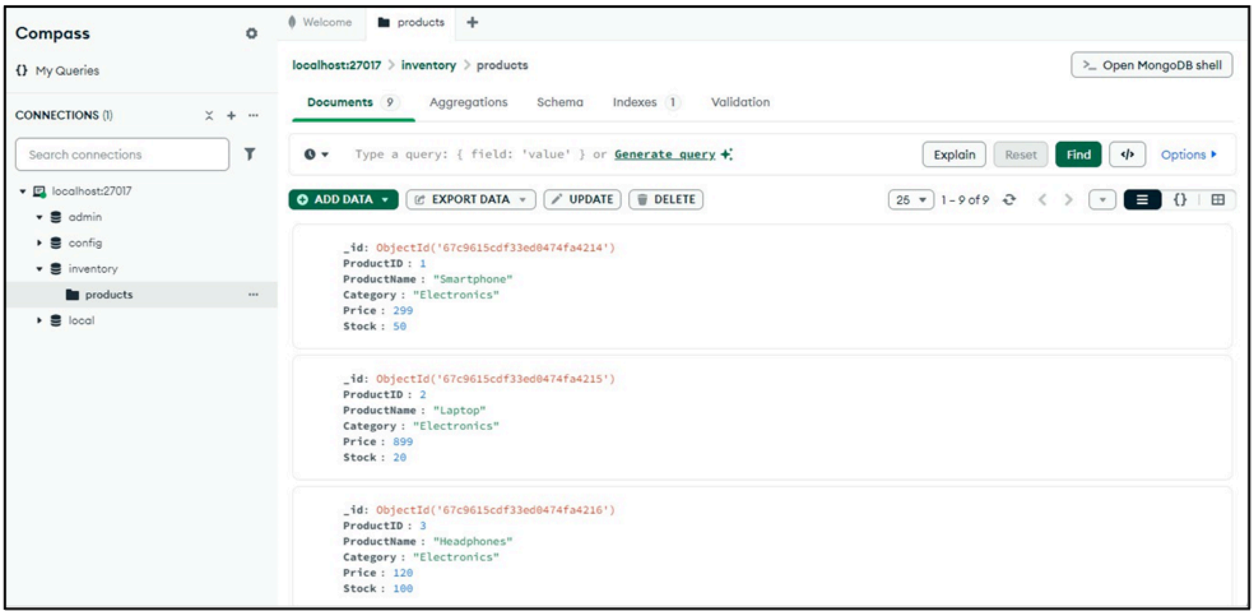
```
{
    "ProductID": 1,
    "ProductName": "Smartphone",
    "Category": "Electronics",
    "Price": 299,
    "Stock": 50
}
```



```
{}
```

```
_id: ObjectId('67db996be6e887e94d60e785')
ProductID : 1
ProductName : "Smartphone"
Category : "Electronics"
Price : 299
Stock : 50
```

Added more data to the database -



```
_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 : 120
Stock : 100
```

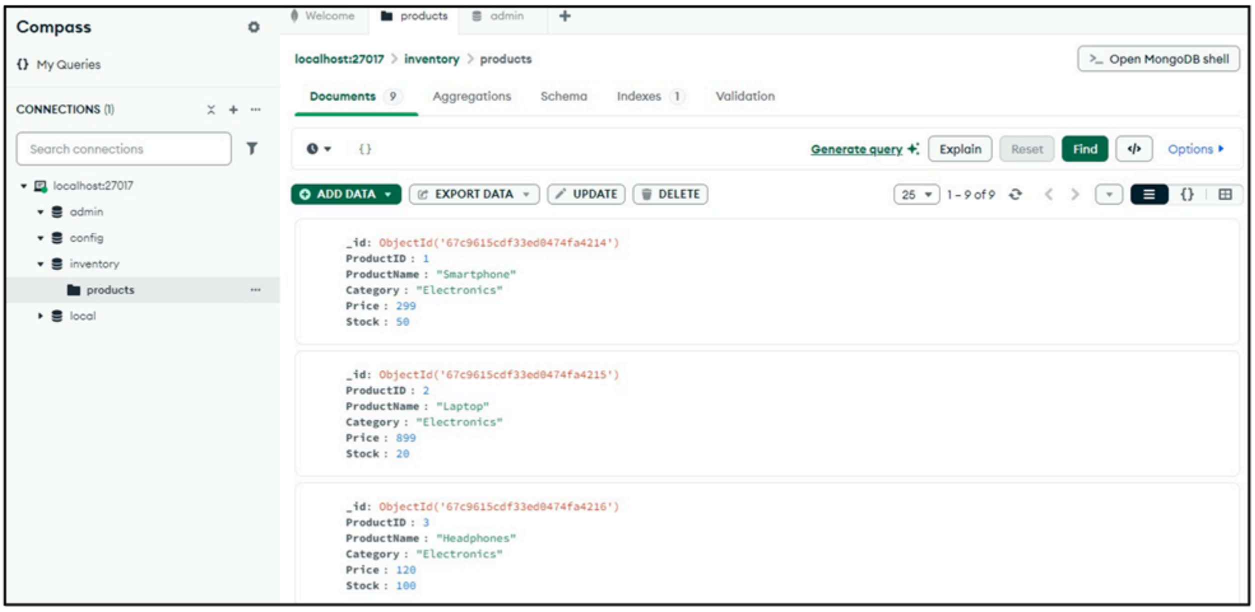
Read Data (Retrieve Documents)

1. Click on the inventory collection.
2. In the "FILTER" field, enter queries to retrieve data.

a) Get all products:

- Query:

  {}

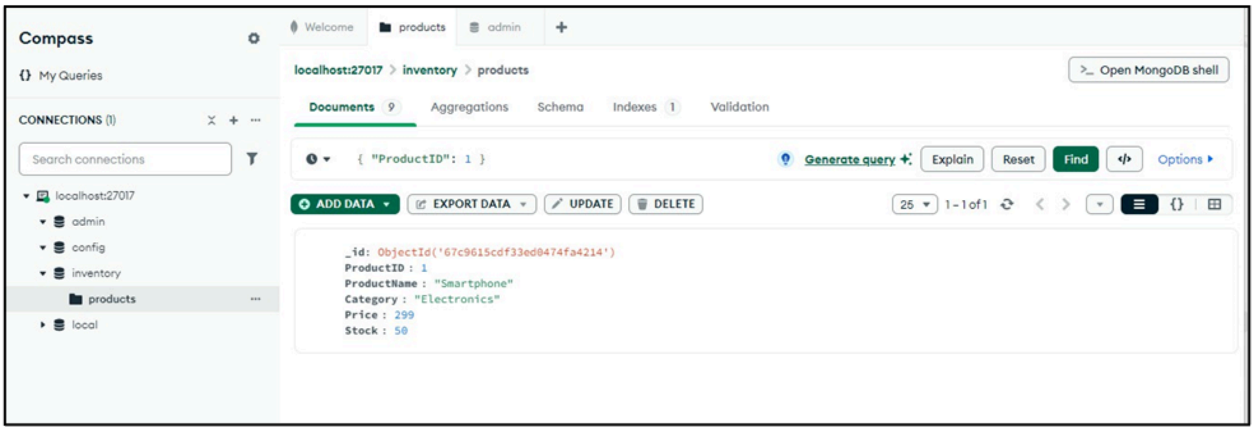


```
_id: ObjectId('67c9615cdf33ed0474fa4214')
ProductID : 1
ProductName : "Smartphone"
Category : "Electronics"
Price : 299
Stock : 50

_id: ObjectId('67c9615cdf33ed0474fa4215')
ProductID : 2
ProductName : "Laptop"
Category : "Electronics"
Price : 899
Stock : 20

_id: ObjectId('67c9615cdf33ed0474fa4216')
ProductID : 3
ProductName : "Headphones"
Category : "Electronics"
Price : 120
Stock : 100
```

b) Get a specific product by ProductID:

- Query:

{ "ProductID": 1 }

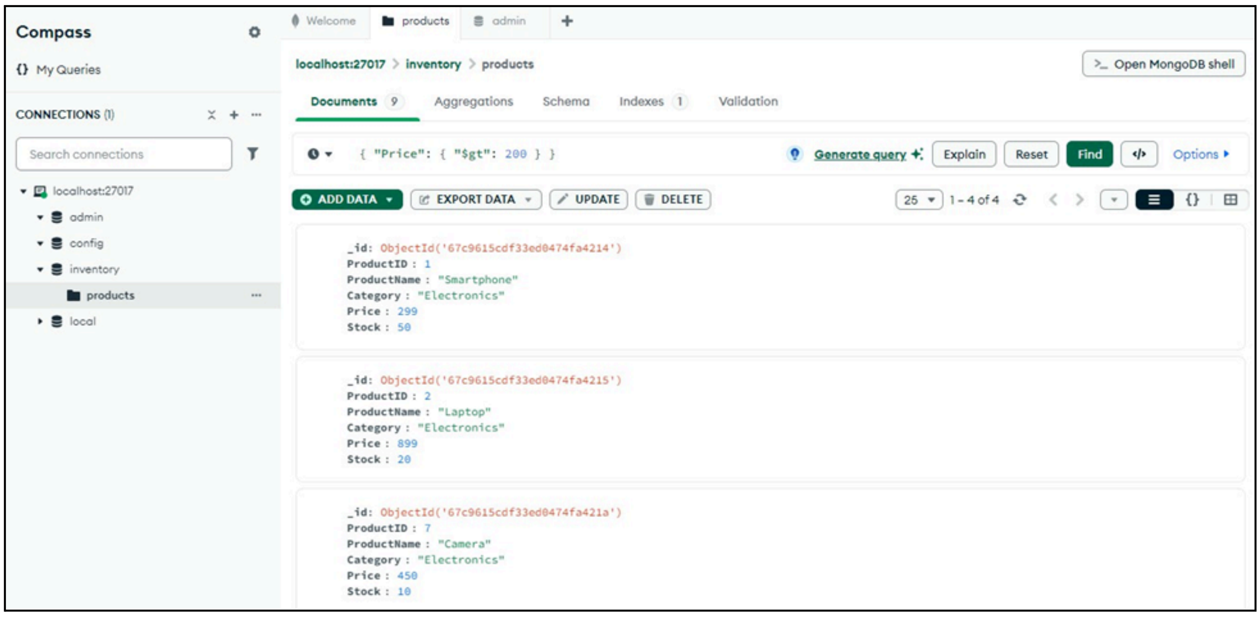


```
{ "ProductID": 1 }

_id: ObjectId('67c9615cdf33ed0474fa4214')
ProductID : 1
ProductName : "Smartphone"
Category : "Electronics"
Price : 299
Stock : 50
```

c) Get products with price greater than 200:

- Query:

{ "Price": { "$gt": 200 } }



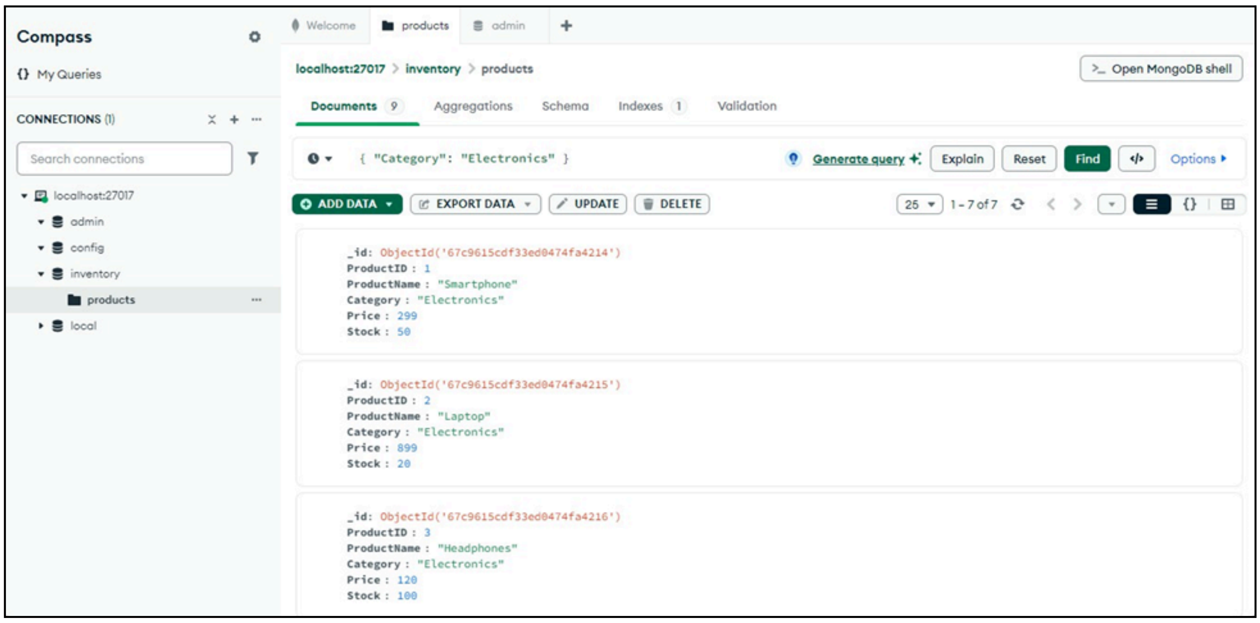
```
{ "Price": { "$gt": 200 } }
```

```
_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('67c9615cdf33ed0474fa421a')
ProductID : 7
ProductName : "Camera"
Category : "Electronics"
Price : 450
Stock : 10
```

d) Get all products in the "Electronics" category:

- Query:

{ "Category": "Electronics" }



```
{ "Category": "Electronics" }
```

```
_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 : 120
Stock : 100
```

<u>Update Data</u>

a) Update the price of a product:

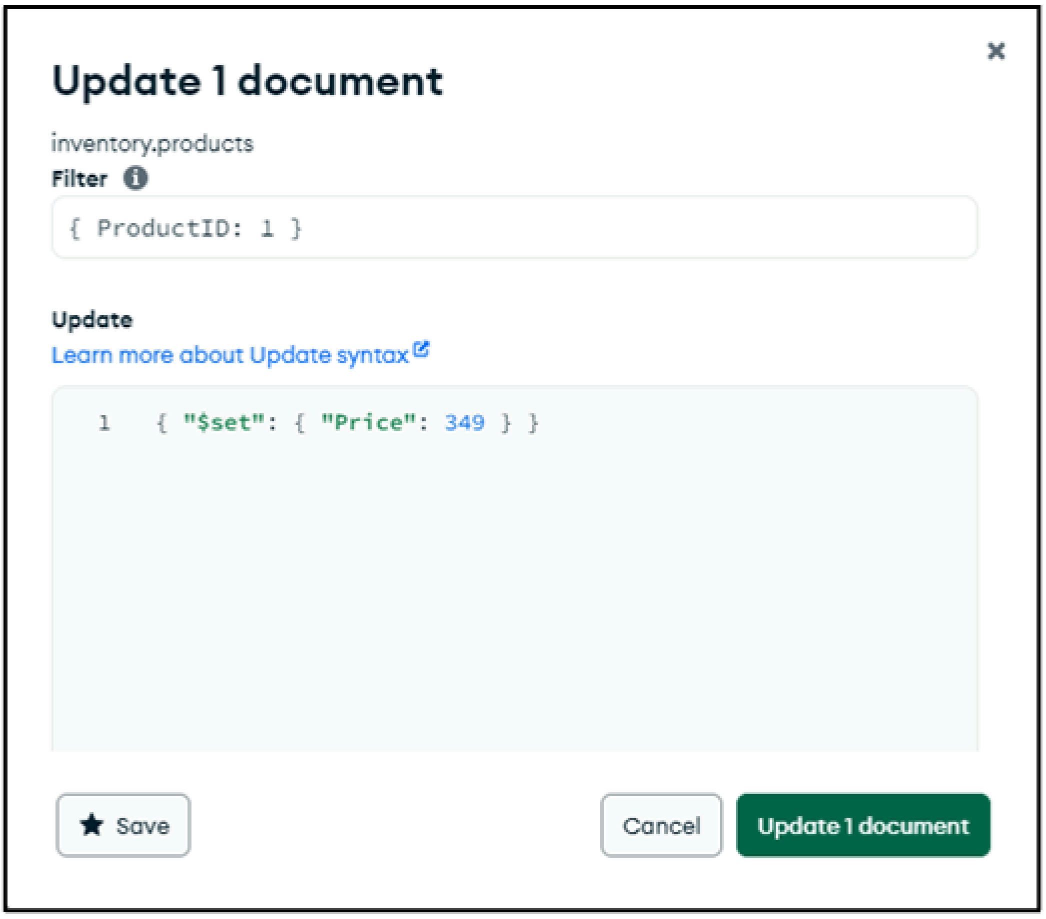
Filter Query (to find the product):

{ "ProductID": 1 }

Update Query:

{ "$set": { "Price": 349 } }

- Click "Update".


Update 1 document

inventory.products

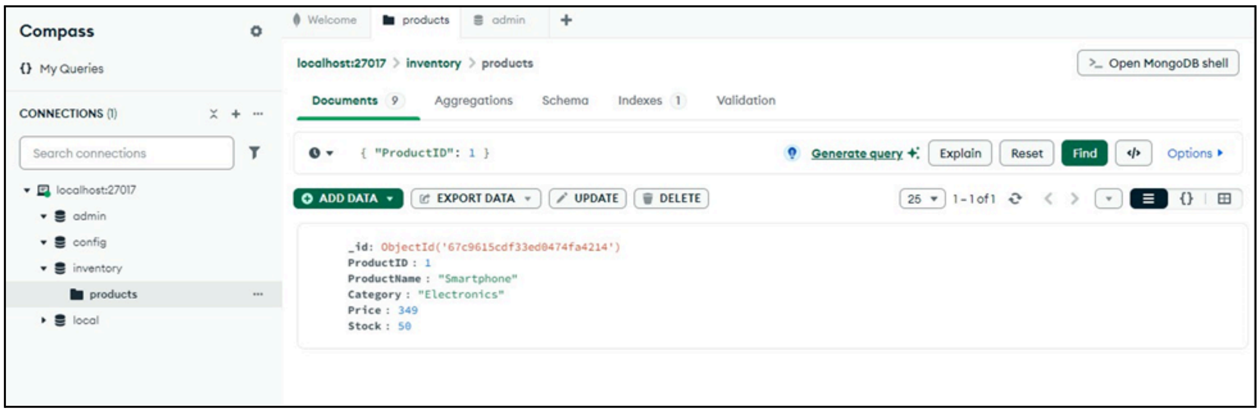
Filter

```
{ ProductID: 1 }
```

Update

Learn more about Update syntax

```
{ "$set": { "Price": 349 } }
```



```
{ "ProductID": 1 }
```

```
_id: ObjectId('67c9615cdf33ed0474fa4214')
ProductID : 1
ProductName : "Smartphone"
Category : "Electronics"
Price : 349
Stock : 50
```

b) Add a new field "Discount" to all products:

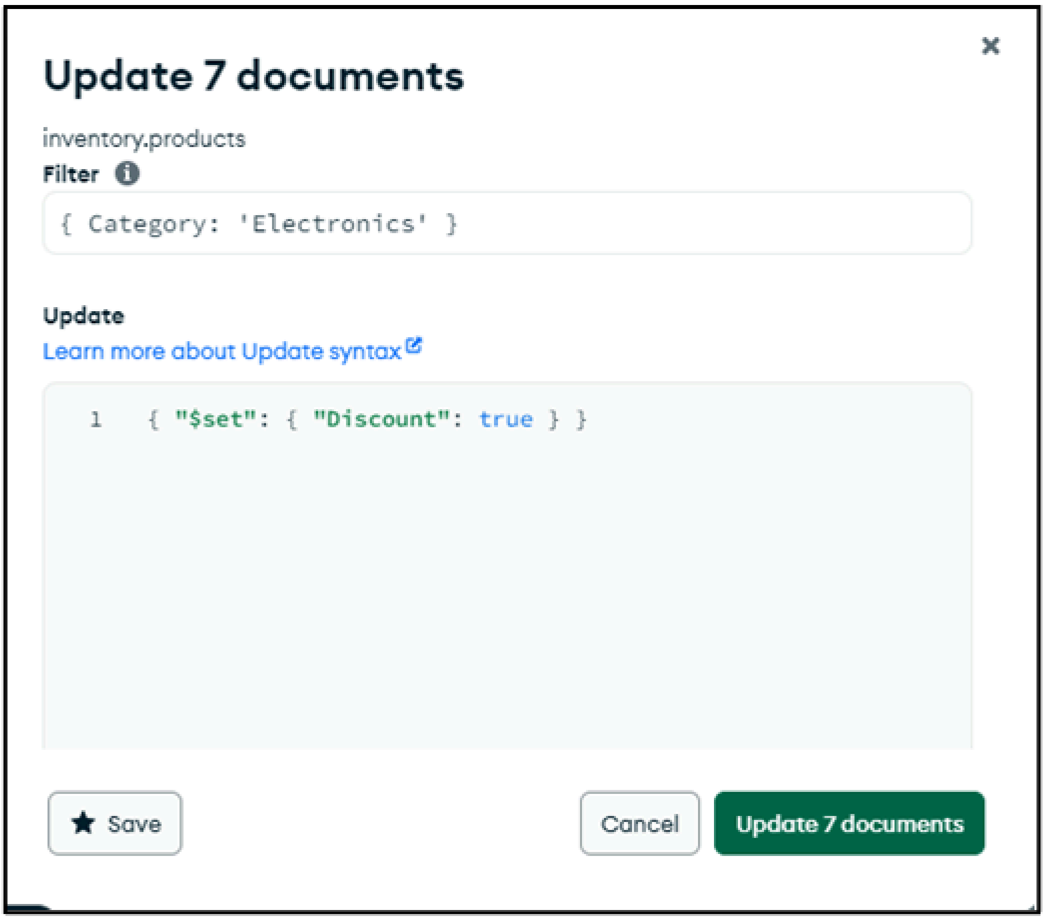
Filter Query:

{ "Category": "Electronics" }

Update Query:

{ "$set": { "Discount": true } }

- Click "Update Many".


Update 7 documents

inventory.products

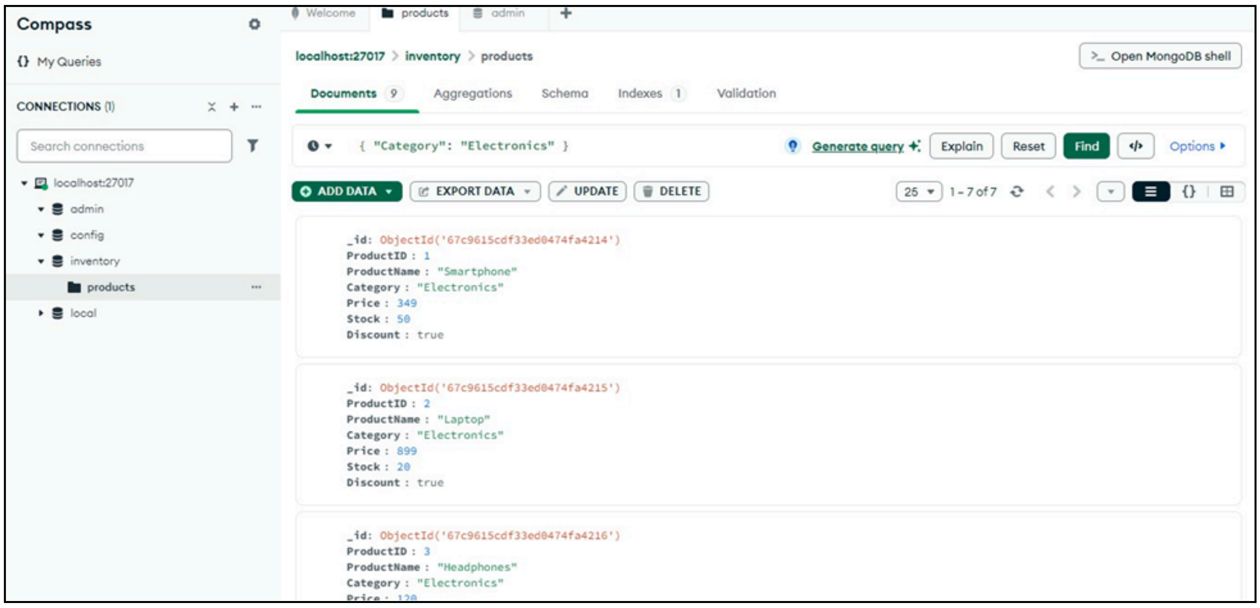
Filter

```
{ Category: 'Electronics' }
```

Update

Learn more about Update syntax

```
{ "$set": { "Discount": true } }
```



```
{ "Category": "Electronics" }
```

```
_id: ObjectId('67c9615cdf33ed0474fa4214')
ProductID : 1
ProductName : "Smartphone"
Category : "Electronics"
Price : 349
Stock : 50
Discount : true

_id: ObjectId('67c9615cdf33ed0474fa4215')
ProductID : 2
ProductName : "Laptop"
Category : "Electronics"
Price : 899
Stock : 20
Discount : true

_id: ObjectId('67c9615cdf33ed0474fa4216')
ProductID : 3
ProductName : "Headphones"
Category : "Electronics"
Price : 120
```

Delete Data

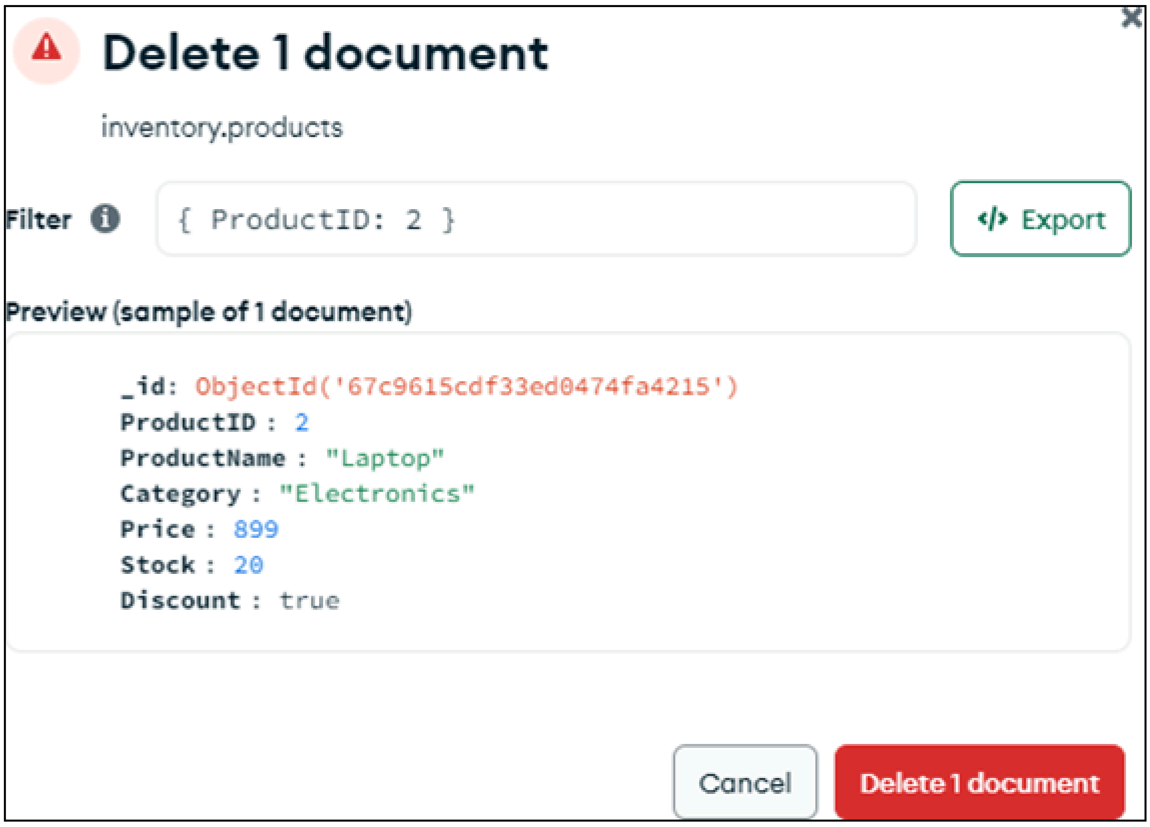
1. Click on the inventory collection.
2. Click "FILTER" and enter the query to find the document you want to delete.
3. Click "DELETE".

a) Delete a specific product:

Filter Query:

{ "ProductID": 2 }

- Click "Delete One".



Delete 1 document

inventory.products

Filter `{ ProductID: 2 }` Export

Preview (sample of 1 document)

```
_id: ObjectId('67c9615cdf33ed0474fa4215')
ProductID : 2
ProductName : "Laptop"
Category : "Electronics"
Price : 899
Stock : 20
Discount : true
```

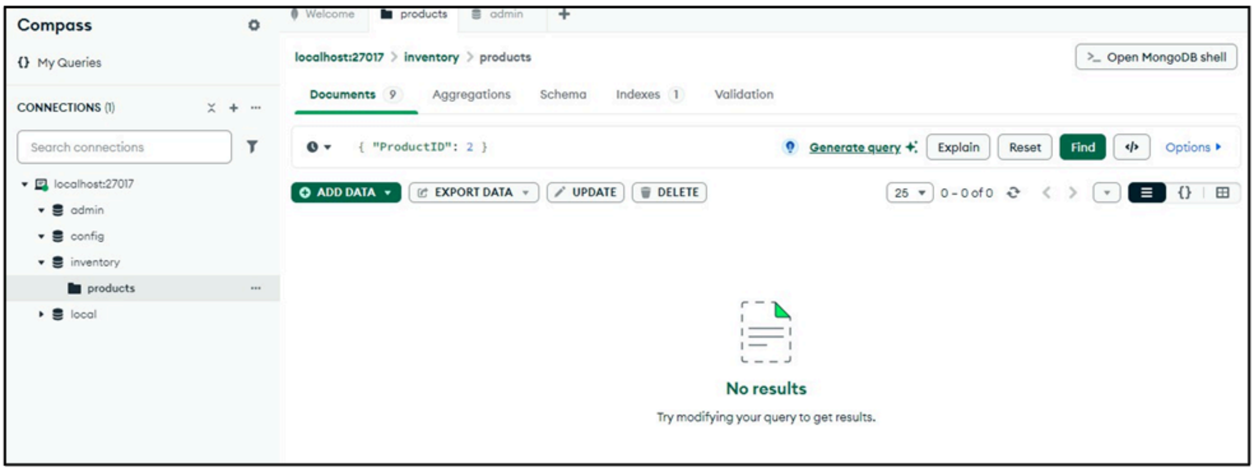
Cancel | Delete 1 document

b) Delete all products in the "Electronics" category:

Filter Query:

{ "Category": "Electronics" }

- Click "Delete Many".



`{ "ProductID": 2 }`

No results

Try modifying your query to get results.

CONCLUSION

Through this experiment, we successfully performed CRUD operations in MongoDB, including creating a database, inserting documents, querying data, updating records, and deleting entries. We also explored filtering data, sorting, and aggregation queries.

MongoDB's document-oriented structure and flexible schema make it an ideal choice for handling large-scale, unstructured data in real-world applications.