EXPERIMENT NO. 6 - MongoDB

Name of Student	**Gunjan Chandnani**
Class Roll No	**D15A_28**
D.O.P.	
D.O.S.	
Sign and Grade	

AIM: To study CRUD operations in MongoDB

Problem Statement:

The objective of this experiment is to perform various **CRUD operations** (Create, Read, Update, Delete) in MongoDB. This involves setting up a database, inserting data, and executing different queries to retrieve and manipulate data effectively.

Theory

1. Features of MongoDB

- Document-Oriented: Stores data as flexible, JSON-like BSON documents.
- Flexible Schema: No fixed structure, allowing dynamic data storage.
- Horizontal Scalability: Uses sharding to distribute large datasets across multiple servers.
- Replication: Ensures high availability through replica sets.
- Indexing: Provides various indexing methods for faster query execution.
- Aggregation Framework: Supports powerful data processing using pipelines.
- Ad-hoc Queries: Enables complex queries without requiring predefined schemas.

Tasks:

a. Create a database named "inventory".
b. Create a collection called "products", containing the fields: ProductID, ProductName, Category, Price, Stock.
c. Insert 10 records into the "products" collection.
d. Retrieve all documents stored in the "products" collection.
e. Filter and display all products that belong to the "Electronics" category.
f. Sort and display products in ascending order based on their names.
g. Retrieve details of the first 5 products in the collection.
h. Find and display the categories of products for a given product name.
i. Count the number of products under the "Electronics" category.
j. Display all product details excluding the "_id" field.
k. Retrieve a list of all distinct categories available in the collection.
l. Filter and display products in the "Electronics" category whose price is between 50 and 100.
m. Update the price of a specific product.
n. Delete a product from the collection.

Collections:

- A collection is a container for multiple documents, similar to a table in SQL.
- Collections do not enforce strict schemas, providing flexibility to store various types of data.

---

3. When to Use MongoDB?

MongoDB is ideal for applications that require:

- **Handling Large Data Sets:** Suitable for **Big Data** applications due to its ability to manage large volumes of unstructured data.
- **Dynamic Schema Requirements:** Useful for **E-commerce platforms**, where product attributes change frequently.
- **Content Management Systems (CMS):** Supports **flexible and scalable** data storage.
- **Real-Time Analytics:** Efficiently processes and analyzes high-speed data streams.
- **Internet of Things (IoT) and Mobile Apps:** Capable of managing sensor data and dynamic app-based data.
- **Social Media Platforms:** Scales effectively for **user-generated content** like posts, comments, and likes.

---

4. Understanding Sharding in MongoDB

Sharding is a **data partitioning** method used in MongoDB to distribute large datasets across **multiple servers**. It helps in improving performance and scalability by handling high loads efficiently.

Sharding Components:

- **Shards:** Contain partitions of data and collectively form the entire database.
- **Config Servers:** Store metadata and manage sharding configurations.
- **Mongos Router:** Acts as a **query router**, directing requests to the appropriate shard.

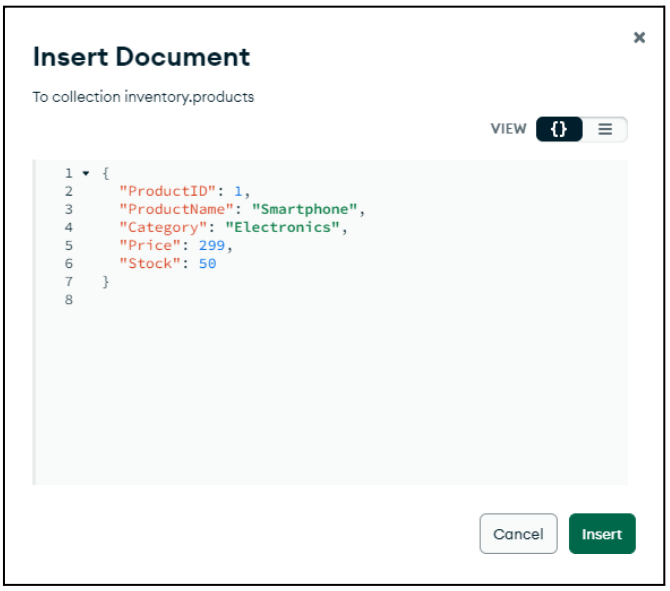
Benefits of Sharding:

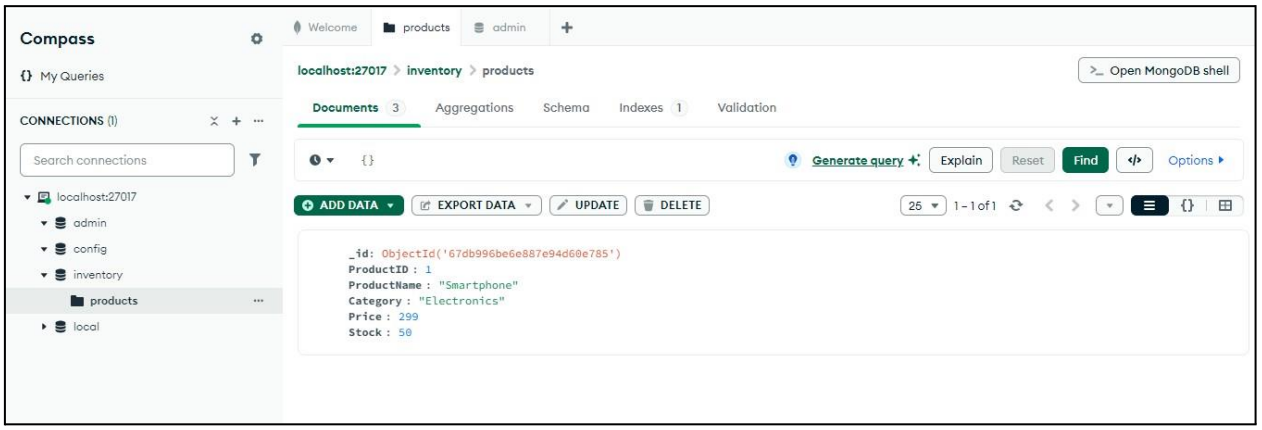
- Enhances **scalability** for managing massive amounts of data.
- Improves **query performance** by distributing workload across multiple machines.
- Provides **high availability** and **fault tolerance** by ensuring data redundancy.

OUTPUT:

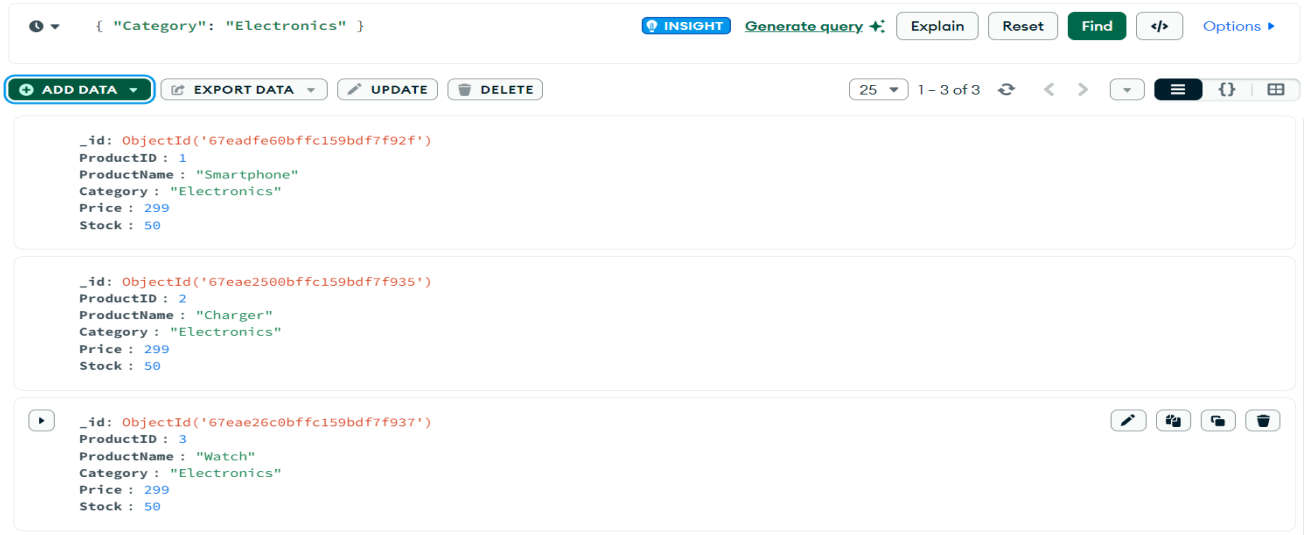
Insert Data (Create Operation)

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





Added more data to the database -



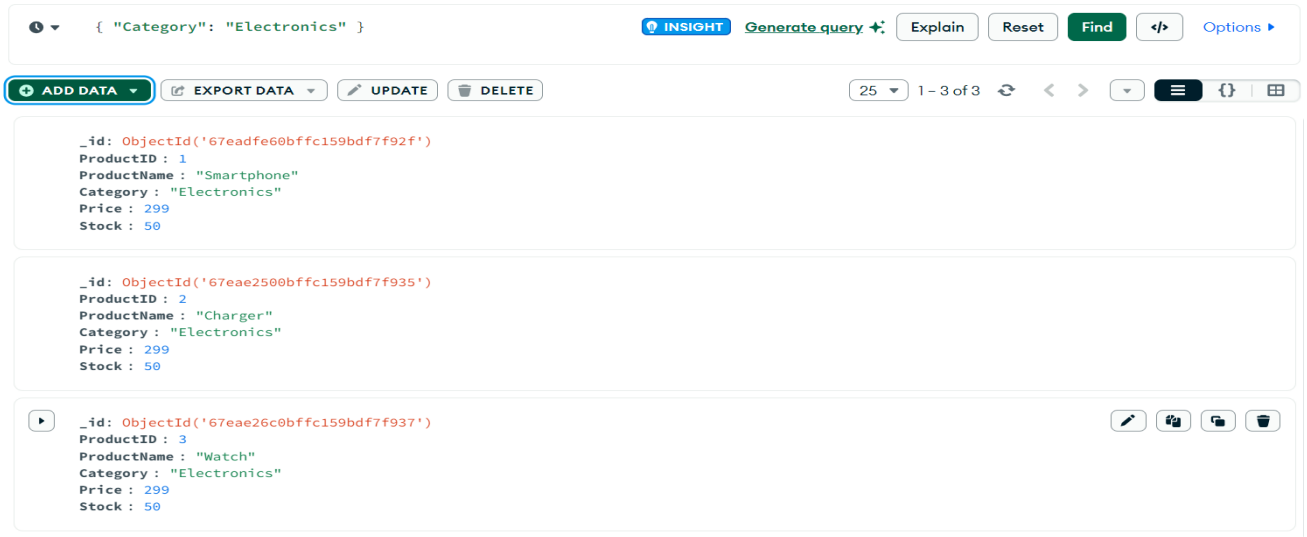
Read Data (Retrieve Documents)

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

a) Get all products:

- Query:

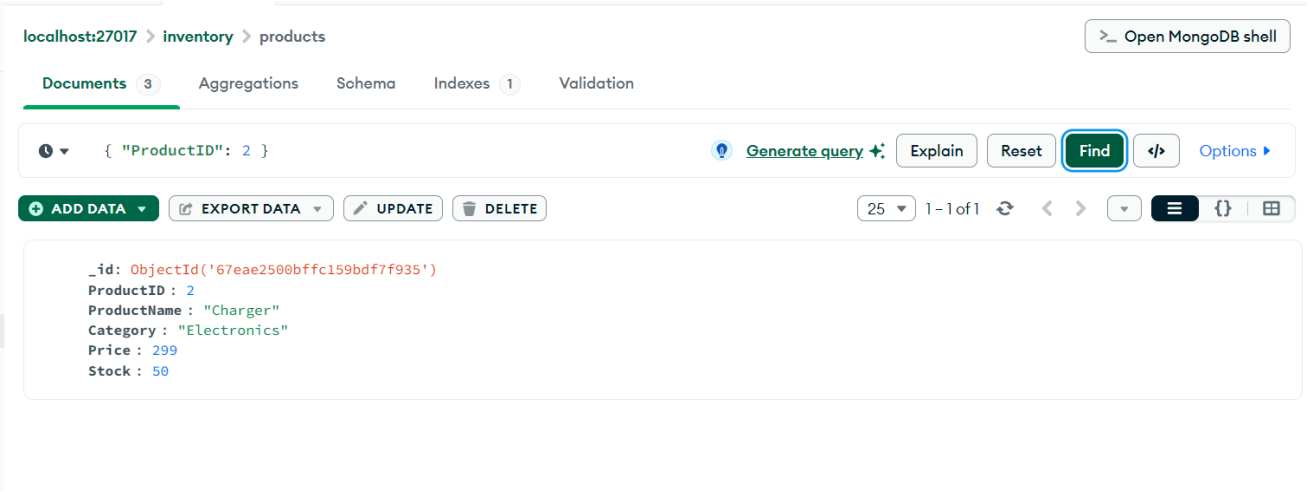
```
{ }
```



b) Get a specific product by `ProductID`:

- Query:

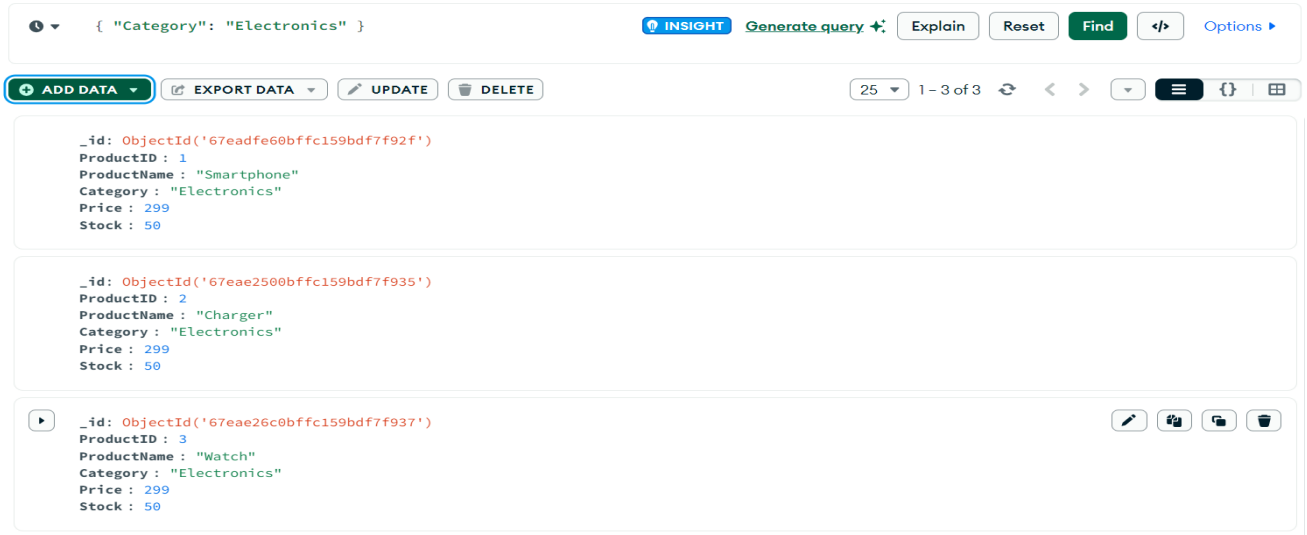
```
{ "ProductID": 2 }
```



c) Get products with price greater than 200:

- Query:

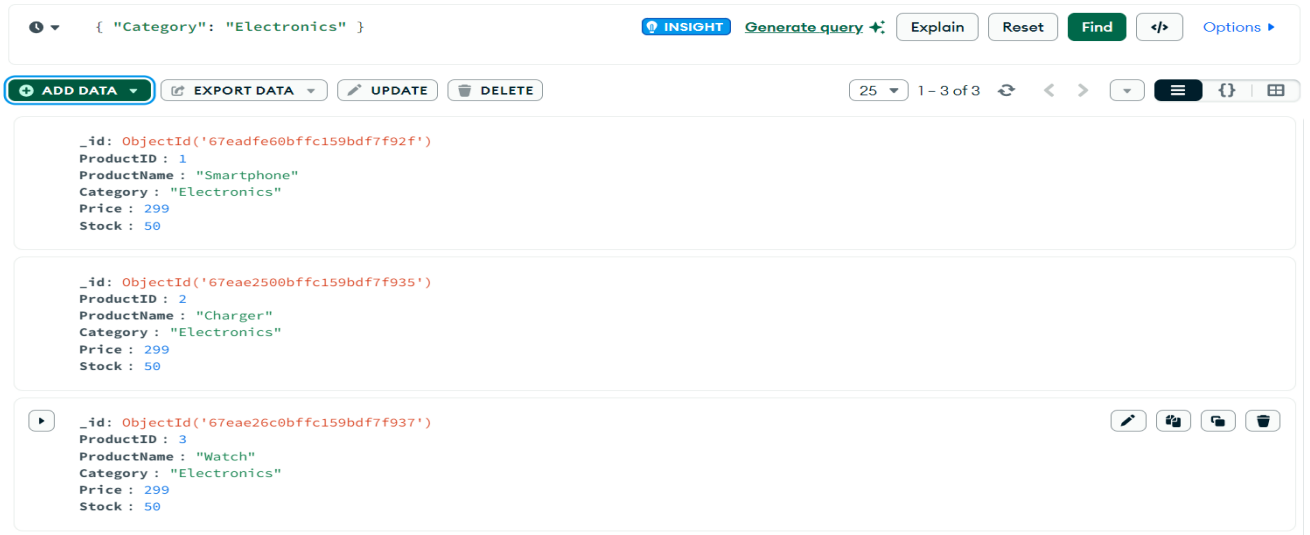
```
{ "Price": { "$gt": 299 } }
```



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

- Query:

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



Update Data

a) Update the price of a product:

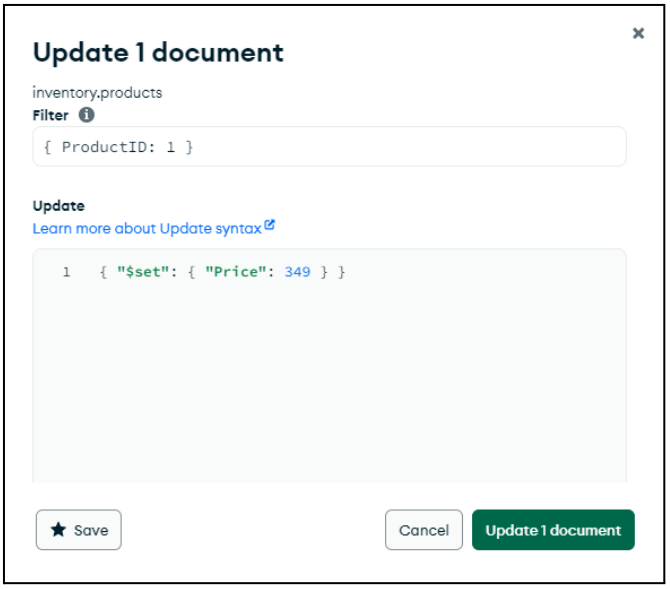
Filter Query (to find the product):

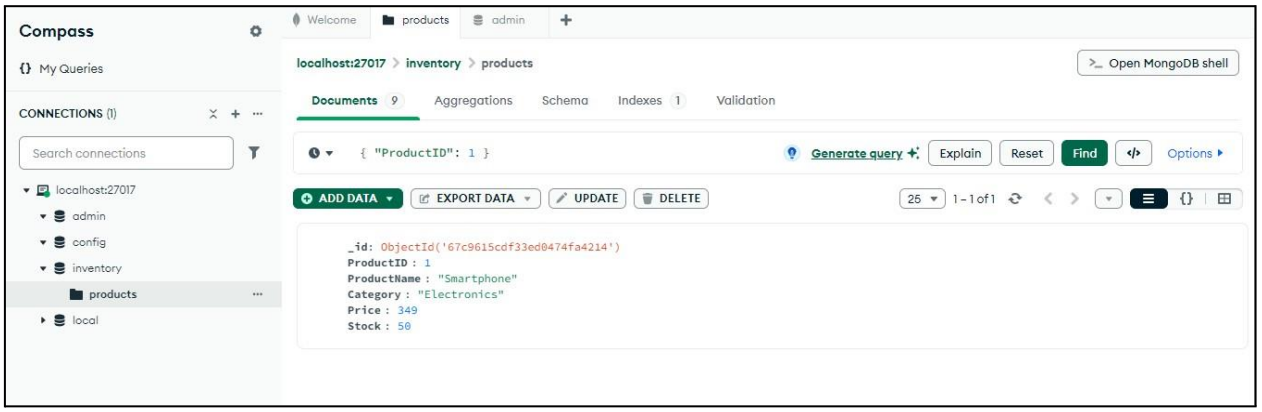
```
{ "ProductID": 1 }
```

Update Query:

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

- Click **"Update"**.





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

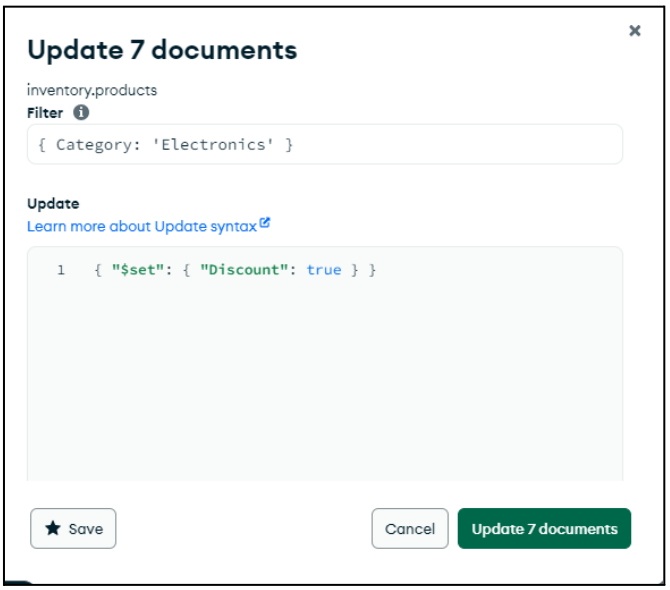
Filter Query:

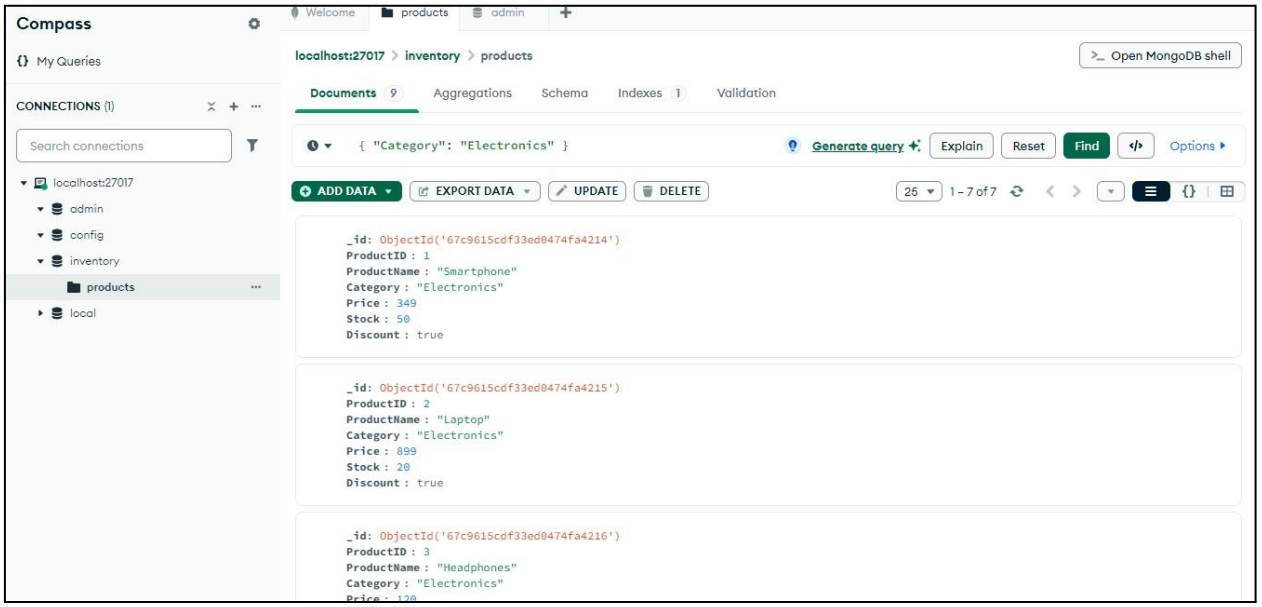
```
{ "Category": "Electronics" }
```

Update Query:

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

- Click **"Update Many"**.





Delete Data

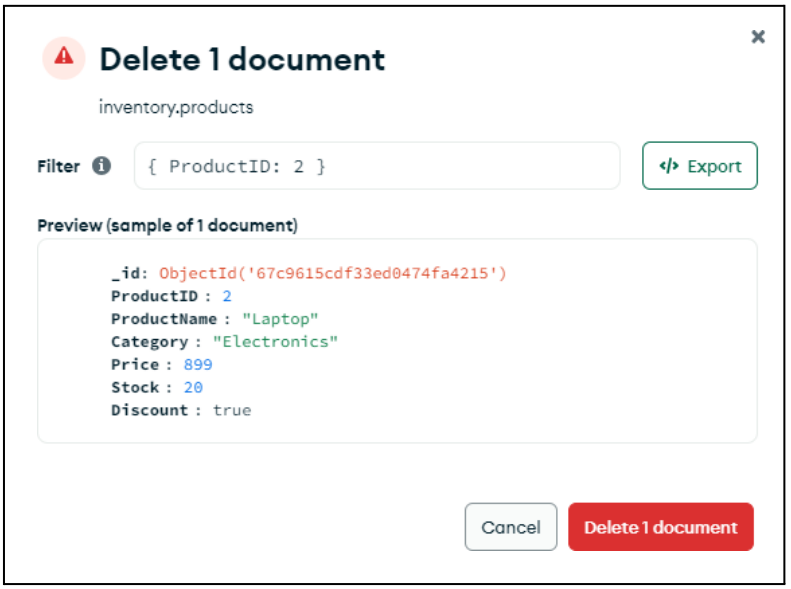
1. Click on the `inventory` collection.
2. Click "**FILTER**" and enter the query to find the document you want to delete.
3. Click "**DELETE**".

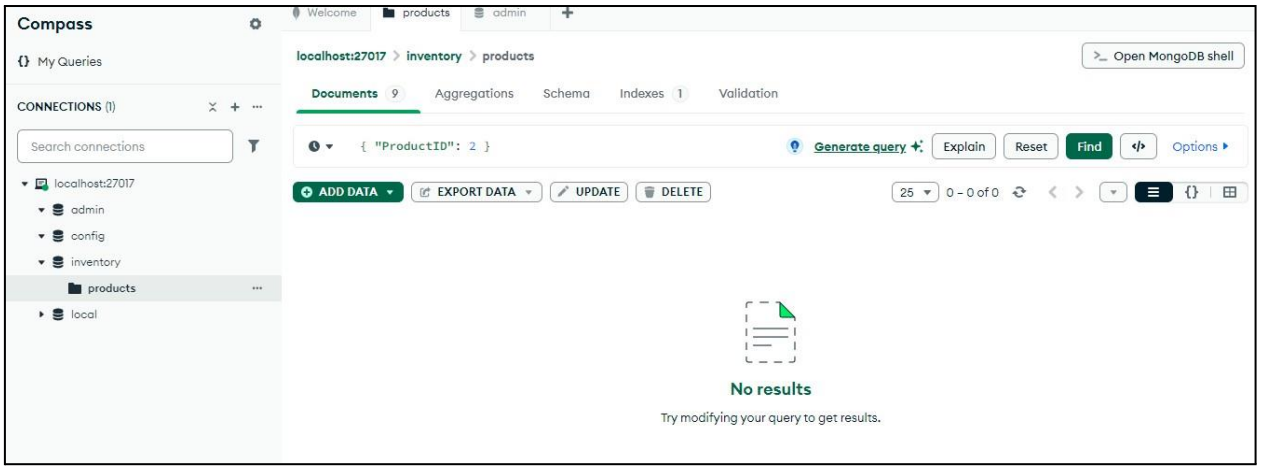
a) Delete a specific product:

Filter Query:

```
{ "ProductID": 2 }
```

- **Click "Delete One".**



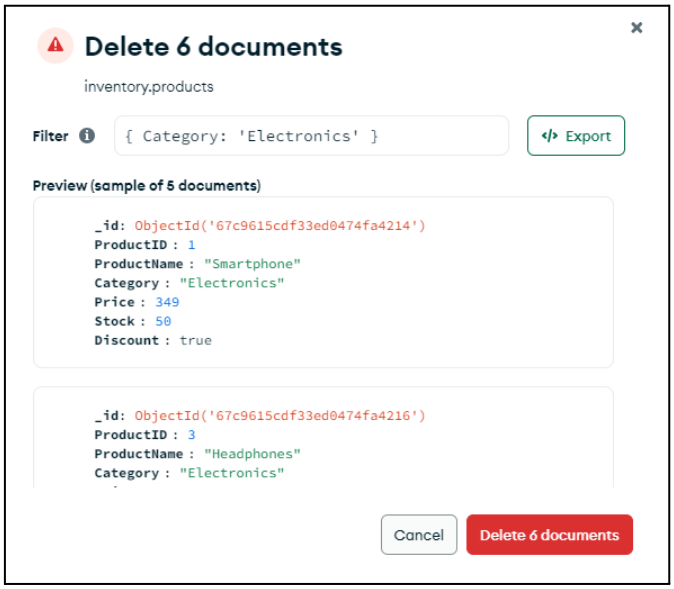


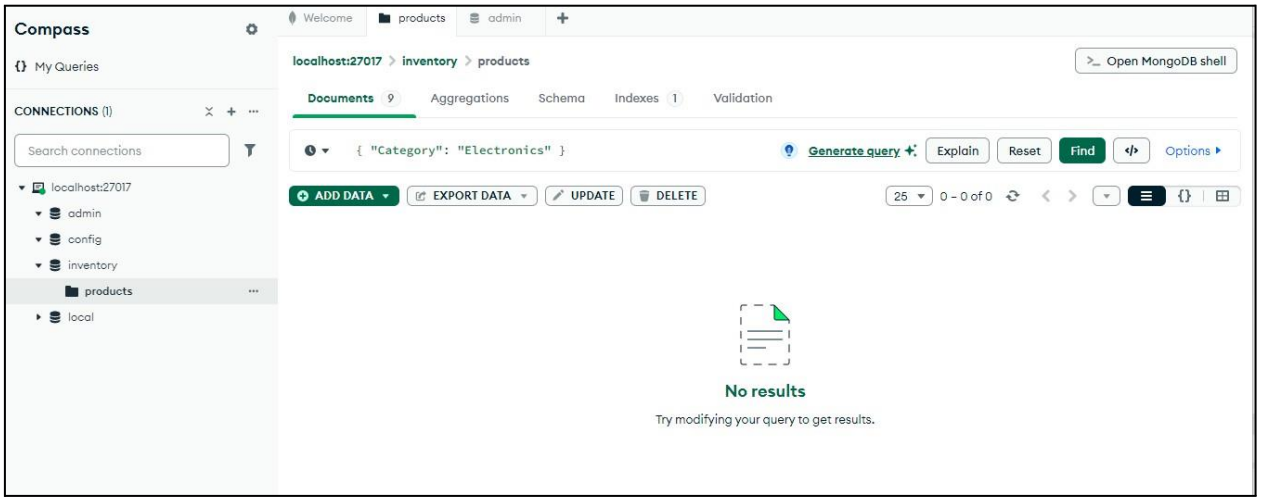
b) Delete all products in the "Electronics" category:

Filter Query:

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

- **Click "Delete Many".**





Conclusion

In this experiment, we successfully carried out CRUD operations in MongoDB, including creating a database, inserting documents, retrieving data, updating records, and deleting entries. Additionally, we explored filtering, sorting, and aggregation queries to manipulate and analyze data efficiently.

MongoDB's document-based structure and flexible schema make it well-suited for managing large-scale and unstructured data. Its scalability and efficiency make it a strong choice for real-world applications requiring dynamic and high-performance data handling.