

**Name : Chirag A Choudhary**

**Class : D15A**

**Roll no. : 10**

### **Experiment – 6: MongoDB**

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 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.

3) **Theory:**

A. Describe some of the features of MongoDB?

- **Flexible Schema:** MongoDB is schema-less, meaning it can store documents with different structures in the same collection.
- **Scalability:** It supports horizontal scaling using sharding.
- **High Performance:** Efficient for read and write operations.
- **Replication:** Provides data redundancy and high availability using replica sets.
- **Indexing:** Supports various types of indexes for efficient query execution.
- **Aggregation Framework:** Allows powerful data aggregation and transformation.

B. What are Documents and Collections in MongoDB?

- **Document:** A document in MongoDB is a JSON-like data structure called BSON (Binary JSON). It consists of field-value pairs, similar to a row in a relational database.
- **Collection:** A collection is a group of MongoDB documents, equivalent to a table in relational databases. Documents within a collection can have varying structures.

### C. When to use MongoDB?

- When dealing with large volumes of unstructured or semi-structured data.
- For applications requiring horizontal scalability.
- When frequent schema changes are expected.
- For real-time analytics and content management systems.

### D. What is Sharding in MongoDB?

- **Sharding** is a method of horizontally partitioning data across multiple servers to handle large datasets.
- MongoDB uses **shards** to store subsets of data, ensuring improved read and write performance.
- A **Shard Key** is used to distribute data evenly across shards.

## 4) Output:

### 1) Create a database and collection

## Create Database

Database Name

Inventory

Collection Name

☐ **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)

**i** Before MongoDB can save your new database, a collection name must also be specified at the time of creation. [More Information](#)

Cancel

Create Database

## Create Collection

Collection Name

☐ **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)

Cancel

Create Collection

## 2) Insert Data

## Insert Document

To collection Inventory.products

VIEW

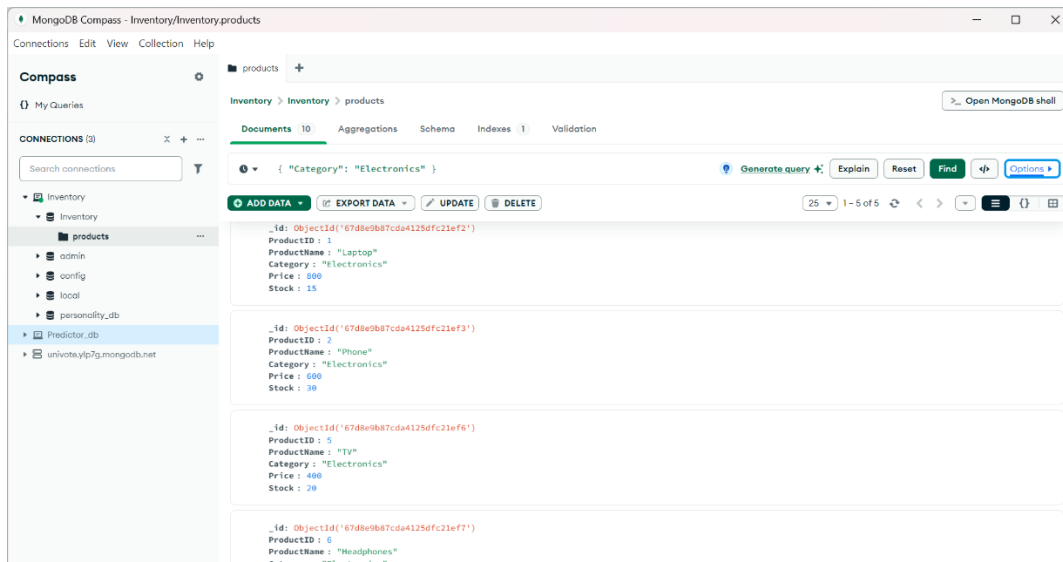


```
1 {  
2   "ProductID": 1,  
3   "ProductName": "Laptop",  
4   "Category": "Electronics",  
5   "Price": 900,  
6   "Stock": 15  
7 }
```

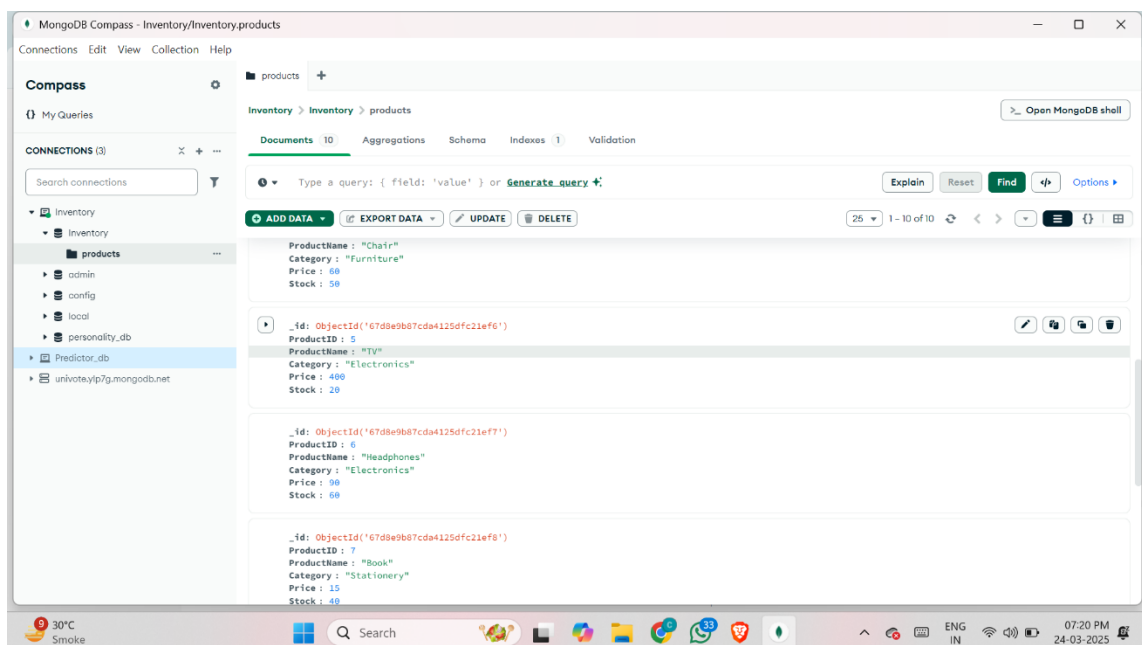
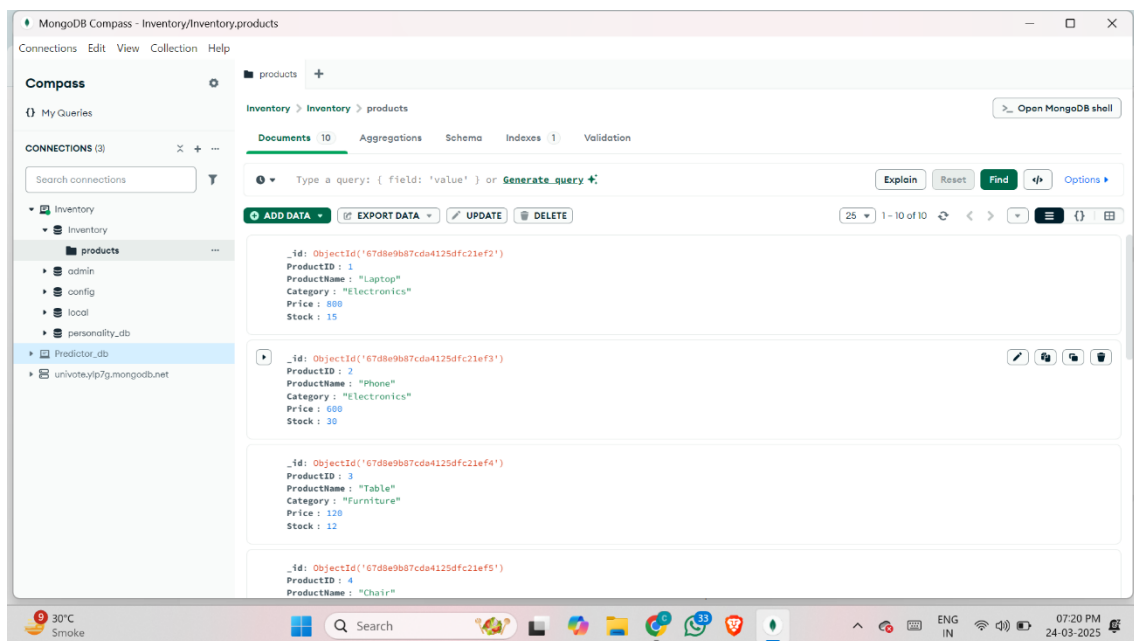


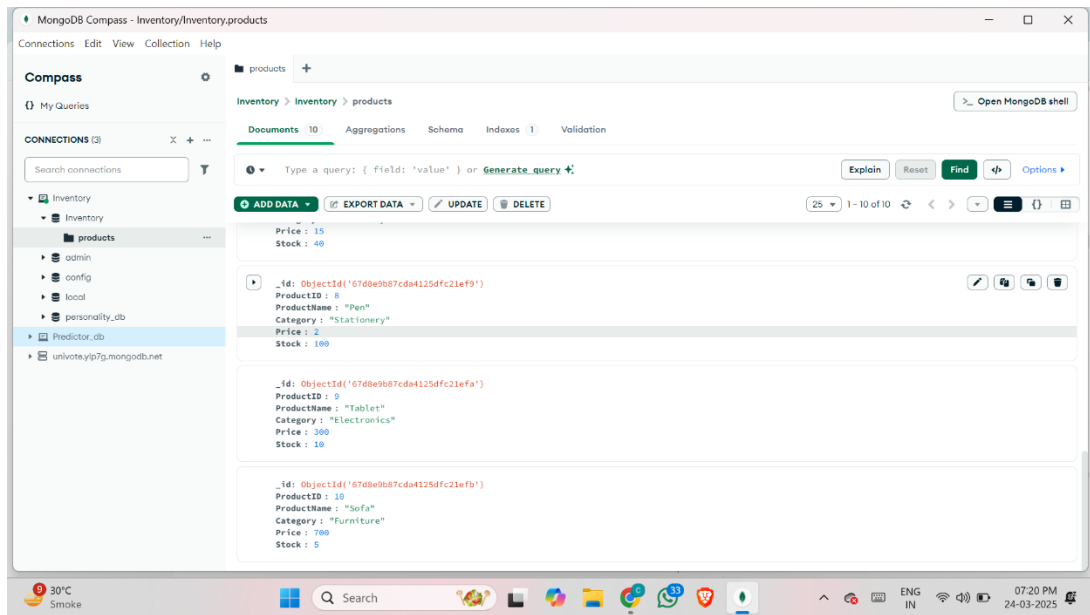
Cancel

Insert

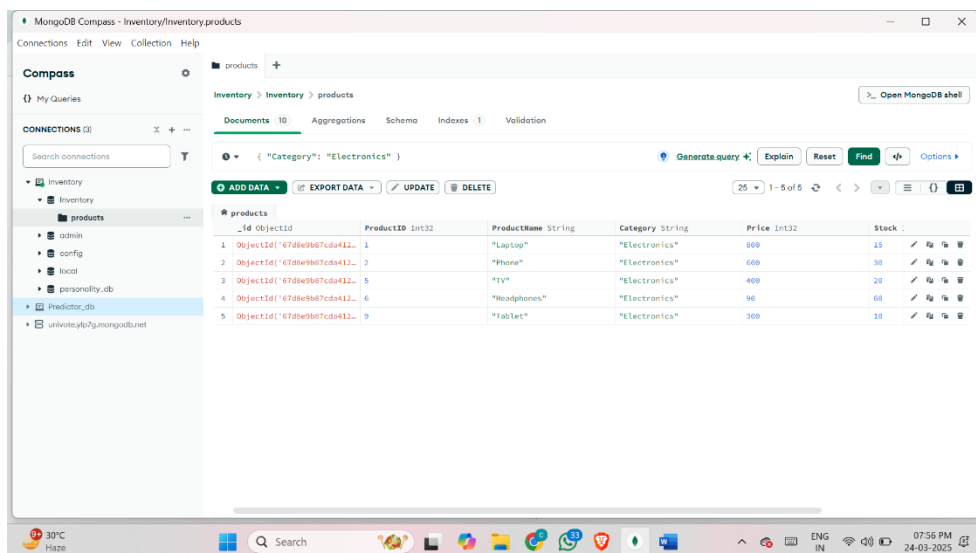


### 3) Display all Documents

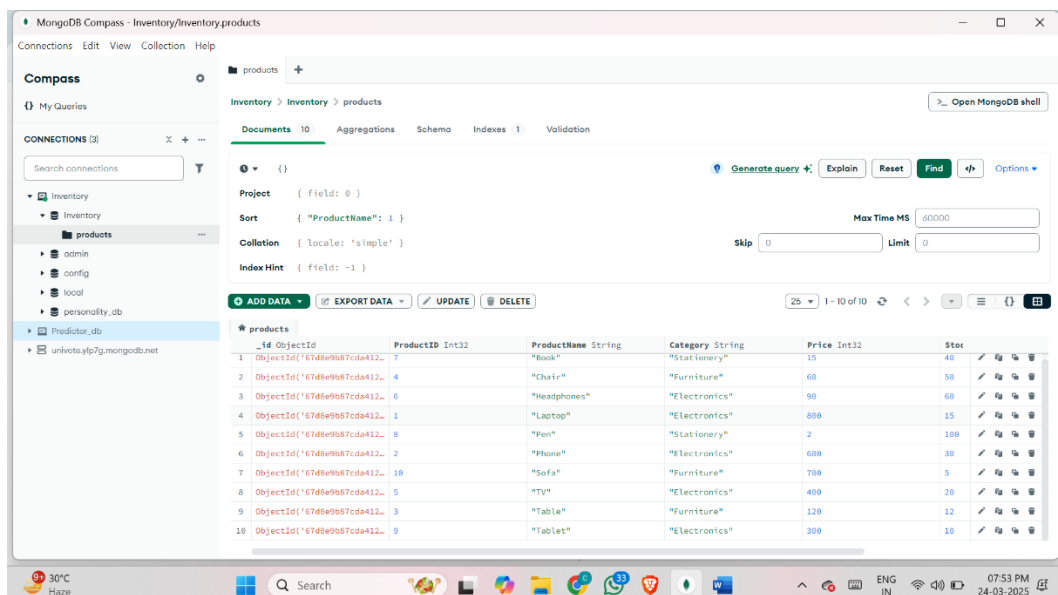




#### 4) Display all Products in the Electronics Category



#### 5) Display Products in Ascending Order of Names



## 6) Display First 5 Products

The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' panel lists the 'Inventory' database and the 'products' collection. The main panel displays the 'products' collection with a table of 5 documents. The table has columns: \_id, ProductID, ProductName, Category, Price, and Stock. The first 5 products are: 1. Laptop (Price: 800, Stock: 15), 2. Phone (Price: 600, Stock: 30), 3. Table (Price: 120, Stock: 12), 4. Chair (Price: 80, Stock: 50), and 5. TV (Price: 400, Stock: 20).

_id	ProductID	ProductName	Category	Price	Stock
ObjectID('67d8e9b87cda412...')	1	Laptop	Electronics	800	15
ObjectID('67d8e9b87cda412...')	2	Phone	Electronics	600	30
ObjectID('67d8e9b87cda412...')	3	Table	Furniture	120	12
ObjectID('67d8e9b87cda412...')	4	Chair	Furniture	80	50
ObjectID('67d8e9b87cda412...')	5	TV	Electronics	400	20

## 7) Display Products with a Specific Name

The screenshot shows the MongoDB Compass interface with a query filter applied: `{ ProductName: "Laptop" }`. The main panel displays the result of the query, showing a single document for the product with ID 1, named 'Laptop', in the 'Electronics' category, with a price of 800 and a stock of 15.

```
{
  "_id": "ObjectID('67d8e9b87cda4125d4c21e2')",
  "ProductID": 1,
  "ProductName": "Laptop",
  "Category": "Electronics",
  "Price": 800,
  "Stock": 15
}
```

## 8) Count Products in Electronics Category

The screenshot shows the MongoDB Compass interface. The left sidebar displays the 'Inventory' database with the 'products' collection selected. The main panel shows the 'products' collection with a filter of `{ "Category": "Electronics" }`. The data is displayed in a table with columns: `_id`, `ProductID`, `ProductName`, `Category`, `Price`, and `Stock`. The table contains 5 documents.

	<code>_id</code>	<code>ProductID</code>	<code>ProductName</code>	<code>Category</code>	<code>Price</code>	<code>Stock</code>
1	<code>ObjectId('67d8e9b87cda412...')</code>	1	"Laptop"	"Electronics"	890	15
2	<code>ObjectId('67d8e9b87cda412...')</code>	2	"Phone"	"Electronics"	690	30
3	<code>ObjectId('67d8e9b87cda412...')</code>	5	"TV"	"Electronics"	490	20
4	<code>ObjectId('67d8e9b87cda412...')</code>	6	"Headphones"	"Electronics"	90	60
5	<code>ObjectId('67d8e9b87cda412...')</code>	9	"Tablet"	"Electronics"	390	10

## 9) Hide the “\_id” Field

The screenshot shows the MongoDB Compass interface with a query applied to the 'products' collection. The query is `{ "projection": { "_id": 0 } }`. The data is displayed in a table with columns: `ProductID`, `ProductName`, `Category`, `Price`, and `Stock`. The table contains 10 documents.

	<code>ProductID</code>	<code>ProductName</code>	<code>Category</code>	<code>Price</code>	<code>Stock</code>
1	1	"Laptop"	"Electronics"	890	15
2	2	"Phone"	"Electronics"	690	30
3	3	"Table"	"Furniture"	120	12
4	4	"Chair"	"Furniture"	60	50
5	5	"TV"	"Electronics"	490	20
6	6	"Headphones"	"Electronics"	90	60
7	7	"Book"	"Stationery"	15	40
8	8	"Pen"	"Stationery"	2	100
9	9	"Tablet"	"Electronics"	390	10
10	10	"Sofa"	"Furniture"	700	5

## 10) Display Distinct Categories

The screenshot shows the MongoDB Compass interface with the 'Aggregations' tab selected. The pipeline consists of a single stage: `$group: { _id: "$Category" }`. The pipeline output shows three distinct categories: Furniture, Stationery, and Electronics.

```
1 {
2   $group: { _id: "$Category" }
3 }
4 {}
```

PIPELINE OUTPUT  
Sample of 3 documents

- \_id: "Furniture"
- \_id: "Stationery"
- \_id: "Electronics"

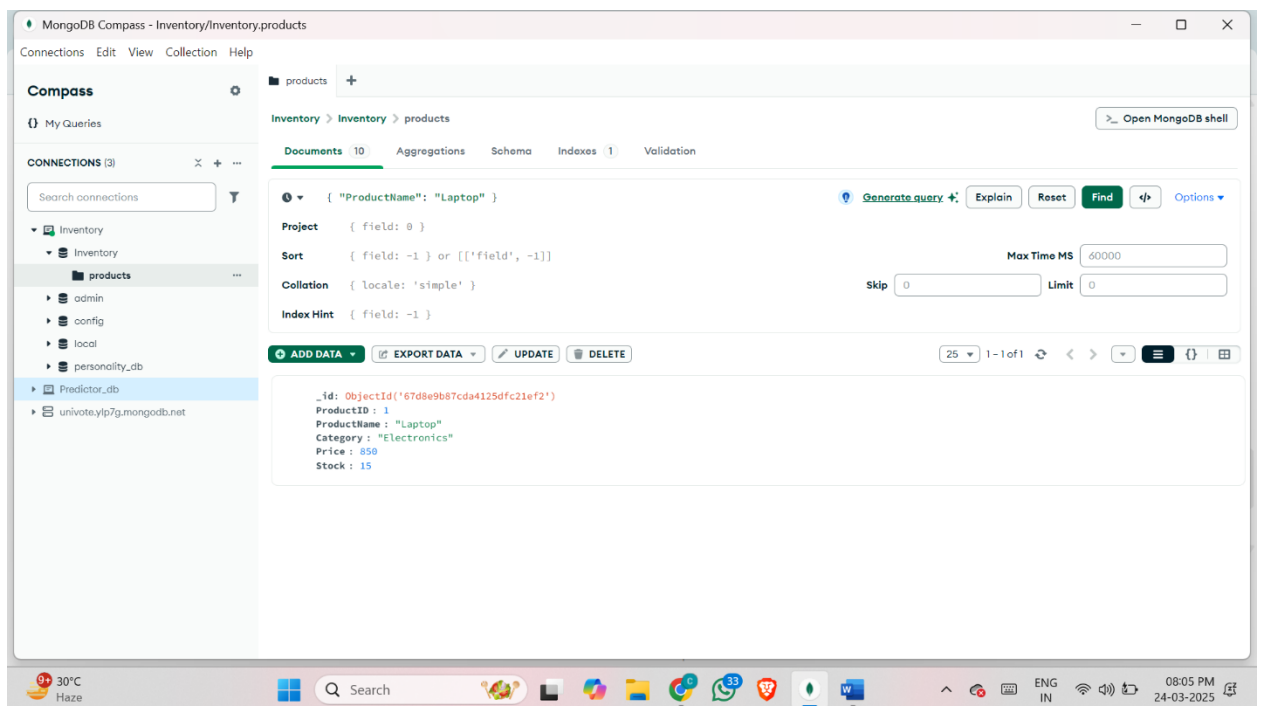
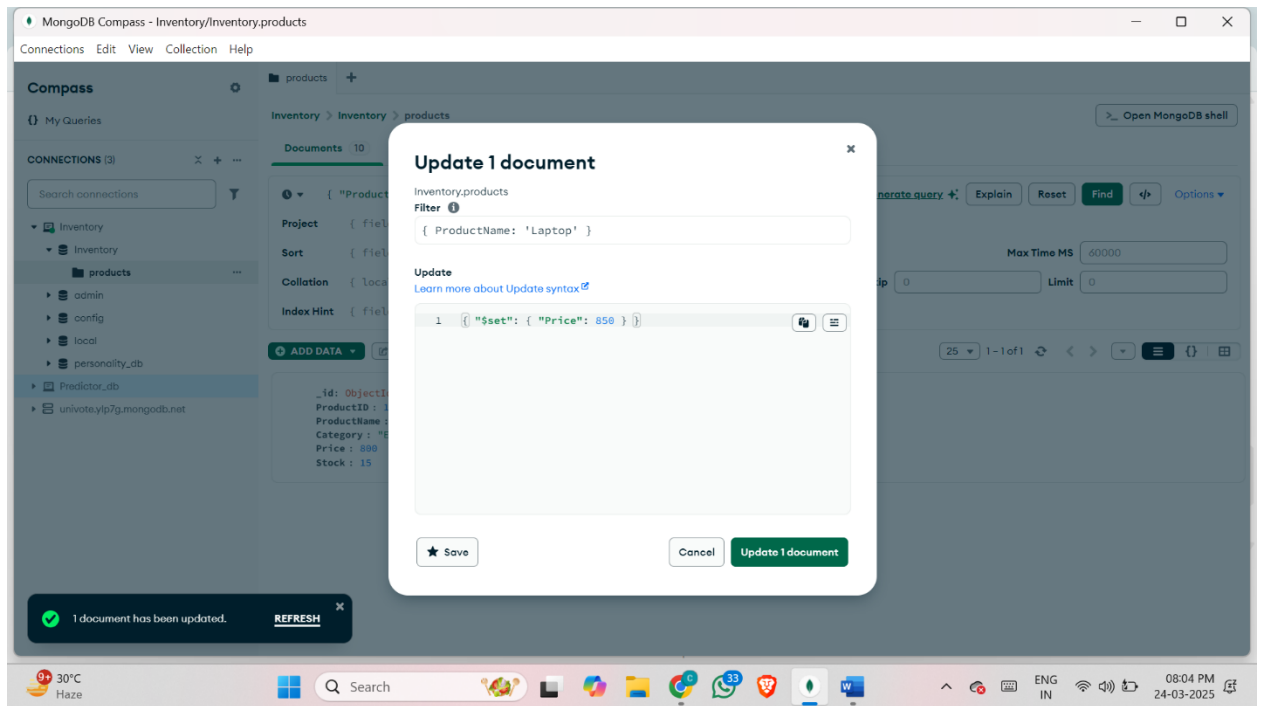
## 11) Display Products in Electronics Category with Price > 50 and < 100

The screenshot shows the MongoDB Compass interface with the 'Documents' tab selected. The query is: `{ "Category": "Electronics", "Price": { "$gt": 50, "$lt": 100 } }`. The results show one product: Headphones.

ProductID	ProductName	Category	Price	Stock
6	Headphones	Electronics	90	60



## 12) Change the Price of a Product



### 13) Delete a Product

