MongoDB Script for Creating Database and Collection with Index

1. Create Database:

To create a database named **ecommerce** (if it doesn't exist), use the following MongoDB script:

```
// Check if the database exists, and switch to it. If it d
use ecommerce;
```

This script switches to the **ecommerce** database. If the database does not already exist, MongoDB will create it when you insert data into it.

---

2. Create Collection with Index:

Next, we create a collection named **products** in the **ecommerce** database and add an index on the **productName** field.

```
// Create the "products" collection with fields: productNar
db.createCollection("products");

// Insert sample product data into the collection
db.products.insertMany([
    { productName: "Laptop", price: 1200 },
    { productName: "Smartphone", price: 800 },
    { productName: "Tablet", price: 600 }
]);

// Create an index on the "productName" field for faster qu
db.products.createIndex({ productName: 1 });
```

Explanation:

1. `db.createCollection("products")`: Creates the **products** collection in the **ecommerce** database.
2. `db.products.insertMany([...])`: Inserts sample product data into the **products** collection. Each product has a **productName** (string) and **price** (integer).
3. `db.products.createIndex({ productName: 1 })`: Creates an index on the **productName** field. The **1** indicates an ascending order for indexing. This index will speed up queries searching for products by their name.

Example Usage After Script Execution:

- **Query to find a product by its name (using the index):**

```
db.products.find({ productName: "Laptop" });
```

- **Query to find all products and sort by name (using the index):**

```
db.products.find().sort({ productName: 1 });
```

This setup ensures that the **products** collection in the **ecommerce** database is optimized for querying the **productName** field efficiently.