// Connect to MongoDB and create a new collection

db = connect("mongodb://localhost:27017/demoDB");

db.createCollection("Products");

// Insert sample products into the collection without attributes

db.Products.insertMany([

    { name: "Laptop", category: "Electronics", price: 1000, quantity: 5 },

    { name: "Phone", category: "Electronics", price: 600, quantity: 10 },

    { name: "Tablet", category: "Electronics", price: 300, quantity: 15 },

    { name: "TV", category: "Electronics", price: 1500, quantity: 3 },

    { name: "T-shirt", category: "Clothing", price: 20, quantity: 50 },

    { name: "Jeans", category: "Clothing", price: 40, quantity: 40 },

    { name: "Sneakers", category: "Footwear", price: 60, quantity: 25 },

    { name: "Boots", category: "Footwear", price: 100, quantity: 15 },

    { name: "Sofa", category: "Furniture", price: 2000, quantity: 2 },

    { name: "Table", category: "Furniture", price: 300, quantity: 10 }

]);

**Sum of prices by category**javascript  
Copy code  
db.Products.aggregate([

    { $group: { \_id: "$category", totalPrice: { $sum: "$price" } } }

]);

**Average price by category**javascript  
Copy code  
db.Products.aggregate([

    { $group: { \_id: "$category", avgPrice: { $avg: "$price" } } }

]);

**Maximum and minimum price in each category**javascript  
Copy code  
db.Products.aggregate([

    { $group: { \_id: "$category", maxPrice: { $max: "$price" }, minPrice: { $min: "$price" } } }

]);

**Count of products by category**javascript  
Copy code  
db.Products.aggregate([

    { $group: { \_id: "$category", productCount: { $count: {} } } }

]);

**Sum of total inventory value by product (price \* quantity)**javascript  
Copy code  
db.Products.aggregate([

    { $project: { name: 1, inventoryValue: { $multiply: ["$price", "$quantity"] } } }

]);

**Filter products with a price less than $500**javascript  
Copy code  
db.Products.aggregate([

    { $match: { price: { $lt: 500 } } }

]);

**Sort products by price in descending order**javascript  
Copy code  
db.Products.aggregate([

    { $sort: { price: -1 } }

]);

**Get products with the number of reviews counted (assuming reviews field is added)**javascript  
Copy code  
db.Products.aggregate([

    { $project: { name: 1, price: 1, no\_of\_reviews: { $size: "$reviews" } } }

]);

**Indexing Examples**

**Create an index on name and price fields**javascript  
Copy code  
db.Products.createIndex({ name: 1, price: -1 });

**Use index to find products by name and price**javascript  
Copy code  
db.Products.find({ name: "Laptop", price: 1000 }).hint("name\_1\_price\_-1");

1. **Create a category index to speed up aggregation by category**javascript  
   Copy code  
   db.Products.createIndex({ category: 1 });

**Steps to Test Query Performance With and Without Index**

1. **Run the Query Without an Index**
   * Run your query and check the execution statistics before creating an index.

Example:  
javascript  
Copy code  
db.Products.find({ name: "Laptop" }).explain("executionStats");

* This will show detailed statistics, including how many documents were scanned and how long it took to execute the query.

1. **Create the Index**

Now, create the index.  
javascript  
Copy code  
db.Products.createIndex({ name: 1 });

1. **Run the Query Again with the Index**
   * Re-run the query with the index now created and again check the execution statistics.  
     javascript  
     Copy code  
     db.Products.find({ name: "Laptop" }).explain("executionStats");

**Example: Count the total number of products in the Products collection**

javascript

Copy code

db.Products.aggregate([

    {

        $count: "totalProducts" // Count all documents in the collection

    }

]);

**Example: Count products by category**

javascript

Copy code

db.Products.aggregate([

    {

        $group: {

            \_id: "$category",

            productCount: { $sum: 1 } // Count the number of products in each category

        }

    }

]);

The $size operator helps you determine the length of an array, which can be useful when you need to analyze the number of items in a group.

The $count operator is straightforward for counting documents in your queries, both overall and within specific groups.