# MongoDB Assignment 2 [ Deadline: 2025-09-03]

Data

**Product Collection:**

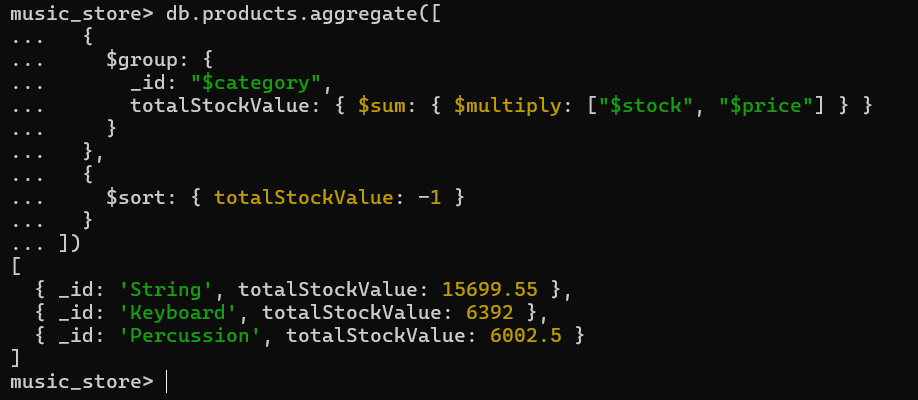
[{ \_id: "PROD001", name: "Acoustic Guitar", category: "String", brand: "GuitarCo", price: 499.99, stock: 15, features: ["Solid Spruce Top", "Mahogany Back & Sides"], reviews: [ { user: "Alice", rating: 5, comment: "Amazing sound!" }, { user: "Bob", rating: 4, comment: "Great for beginners." }, ],},{ \_id: "PROD002", name: "Electric Piano", category: "Keyboard", brand: "KeyMaster", price: 799.0, stock: 8, features: ["88 Weighted Keys", "Multiple Voices"], reviews: [ { user: "Charlie", rating: 5, comment: "Love the feel of the keys." }, ],},{ \_id: "PROD003", name: "Drum Kit", category: "Percussion", brand: "BeatKing", price: 1200.5, stock: 5, features: ["5-Piece Kit", "Cymbals Included"], reviews: [],},{ \_id: "PROD004", name: "Ukulele", category: "String", brand: "AlohaTune", price: 89.99, stock: 30, features: ["Soprano Size", "Mahogany Body"], reviews: [{ user: "Alice", rating: 4, comment: "Cute and fun!" }],},{ \_id: "PROD005", name: "Bass Guitar", category: "String", brand: "BassPro", price: 550.0, stock: 10, features: ["4-String", "Active Pickups"], reviews: [],}];

**Order Collection:**

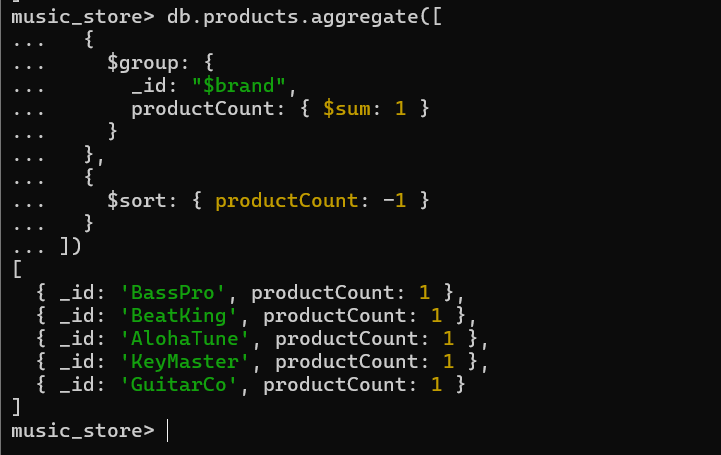
[{ \_id: "ORDER001", customer\_id: "CUST001", order\_date: ISODate("2023-01-10T10:00:00Z"), items: [ { product\_id: "PROD001", quantity: 1, unit\_price: 499.99 }, { product\_id: "PROD004", quantity: 2, unit\_price: 89.99 }, ], status: "completed", total\_amount: 679.97,},{ \_id: "ORDER002", customer\_id: "CUST002", order\_date: ISODate("2023-01-15T14:30:00Z"), items: [{ product\_id: "PROD002", quantity: 1, unit\_price: 799.0 }], status: "pending", total\_amount: 799.0,},{ \_id: "ORDER003", customer\_id: "CUST001", order\_date: ISODate("2023-02-01T09:00:00Z"), items: [{ product\_id: "PROD001", quantity: 1, unit\_price: 499.99 }], status: "completed", total\_amount: 499.99,},{ \_id: "ORDER004", customer\_id: "CUST003", order\_date: ISODate("2023-02-05T11:45:00Z"), items: [{ product\_id: "PROD003", quantity: 1, unit\_price: 1200.5 }], status: "completed", total\_amount: 1200.5,},{ \_id: "ORDER005", customer\_id: "CUST002", order\_date: ISODate("2023-03-01T16:00:00Z"), items: [{ product\_id: "PROD005", quantity: 1, unit\_price: 550.0 }], status: "pending", total\_amount: 550.0,}]

1. Write queries for the given questions using the above data.
   1. Calculate Total Stock Value by Category.

Ans:



* 1. Count Products per Brand.

Ans: 

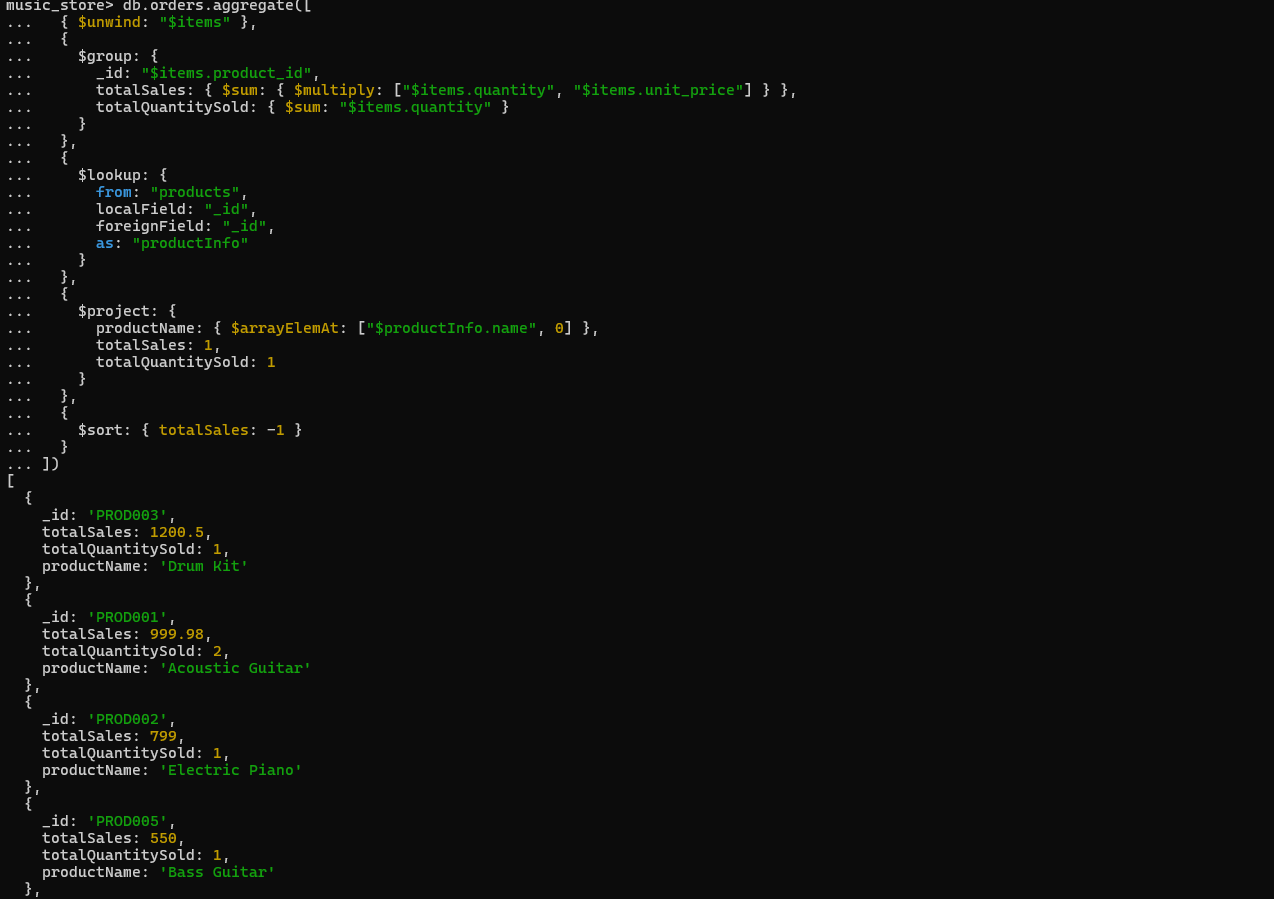
* 1. Find the Average Rating for Each Product.

Ans:

A screenshot of a computer program

AI-generated content may be incorrect.

* 1. Calculate Total Sales for Each Product.

Ans: 

* 1. Recalculate each order's total amount using $reduce.

Ans:

A computer screen shot of a program code

AI-generated content may be incorrect.

* 1. Find the total quantity of 'String' products sold.

Ans: A screen shot of a computer

AI-generated content may be incorrect.Find the average rating for each product that has received at least one review.

Ans: A screen shot of a computer screen

AI-generated content may be incorrect.

* 1. For each product, create a summary that includes its name and price, a simple list of the usernames who reviewed it, and a list of the order IDs in which it was sold.

Ans: A computer screen shot of a black screen

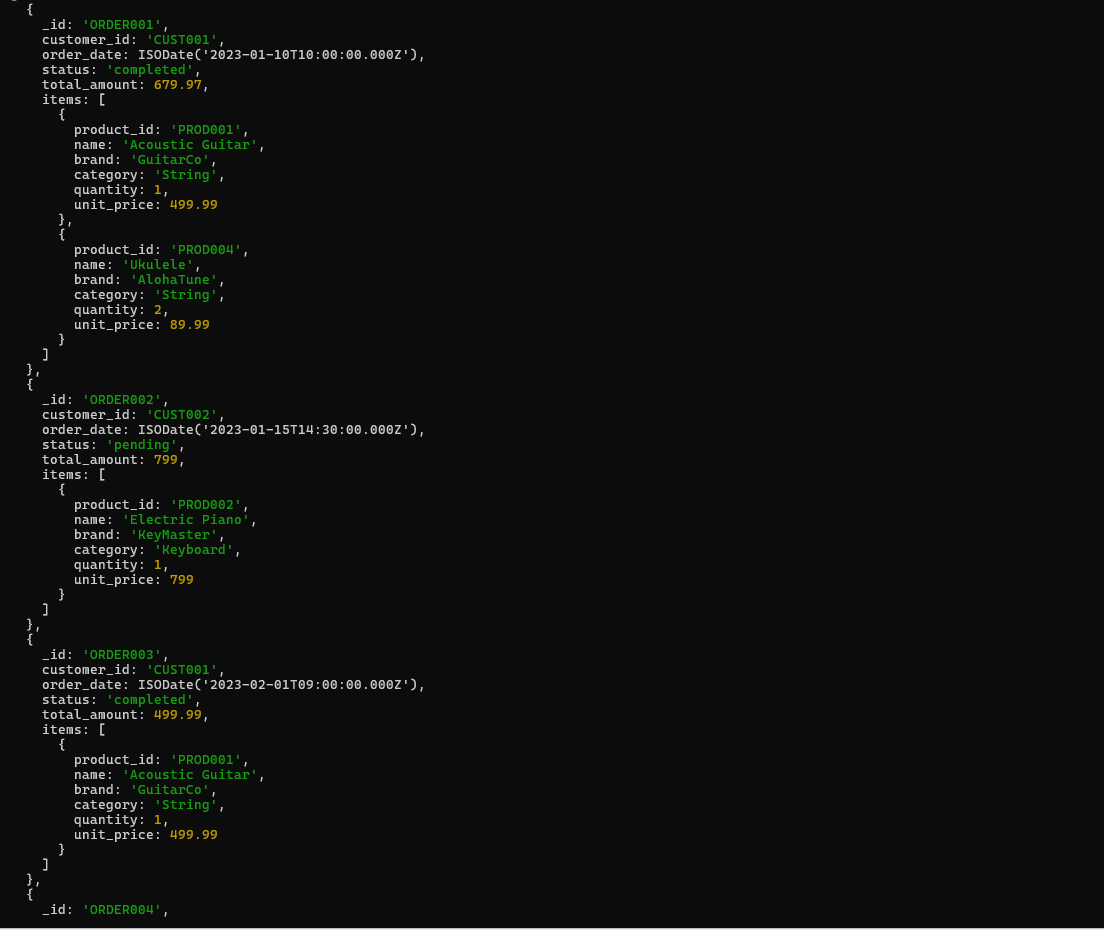
AI-generated content may be incorrect.

* 1. Retrieve a list of all orders, but instead of just showing product IDs in the items array, replace them with a more detailed object containing the product's name, brand, and category.

Ans:

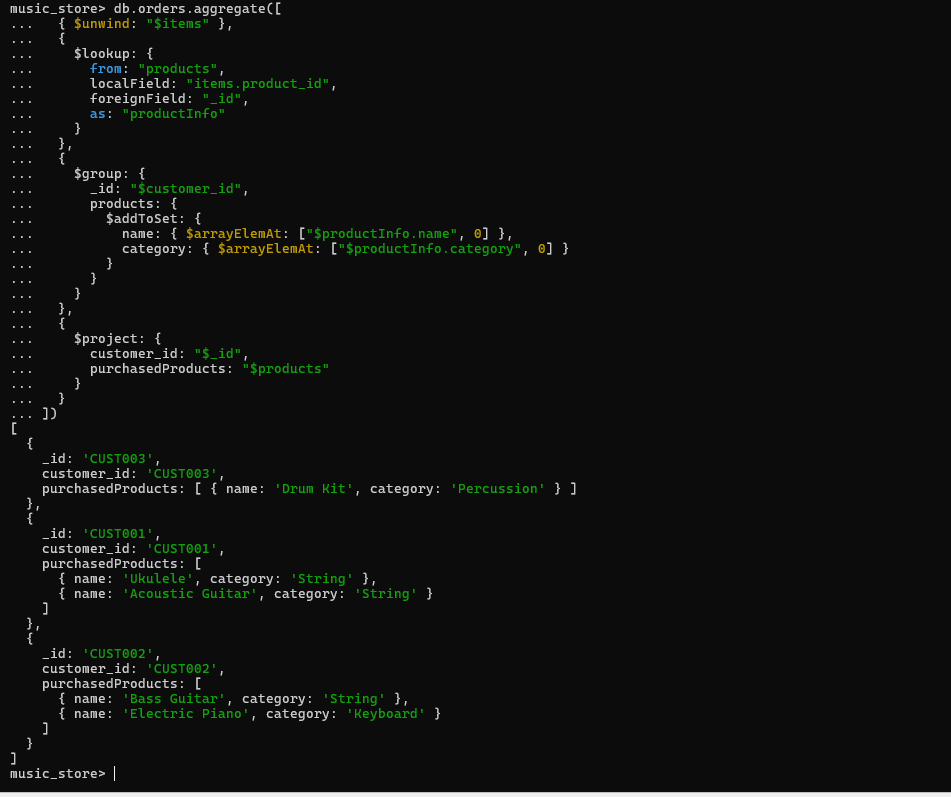
A computer screen shot of a computer code

AI-generated content may be incorrect.



* 1. Create a profile for each customer showing a list of products they have purchased. The list should not contain duplicates and should include the product name and category.

Ans:



1. Create documentation on Map-Reduce and the Input-Output model in MongoDB.

Ans: Map-Reduce is a data processing paradigm that uses a distributed, parallel algorithm to process large data sets. Complex aggregation tasks that might be challenging to express using the conventional aggregation pipeline can be accomplished with MongoDB's Map-Reduce operations.

Map-Reduce Components

**1. Map Function**

* **Purpose**: Processes each document in the collection and emits key-value pairs
* **Input**: Individual documents from the collection
* **Output**: Zero or more key-value pairs using emit(key, value)

function mapFunction() {

*// Process the document (this)*

*// Emit key-value pairs* emit(key, value);

}

**2. Reduce Function**

* **Purpose**: Processes the emitted key-value pairs and combines values for the same key
* **Input**: A key and an array of values for that key
* **Output**: A single reduced value for the key

function reduceFunction(key, values) {

*// Process the array of values*

*// Return a single reduced value* return reducedValue;

}

**Map-Reduce Input-Output Model**

**Input Model**

1. **Collection: Source collection containing documents to process**
2. **Query: Optional query to filter input documents**
3. **Sort: Optional sort specification for input documents**
4. **Limit: Optional limit on number of input documents**

**Input Documents → Map Function → Intermediate Key-Value Pairs → Reduce Function → Final Results → Output Collection/Inline Results**

**Output Model**

1. **Inline: Results returned directly (limited to 16MB)**
2. **Replace: Replace existing collection with results**
3. **Merge: Merge results with existing collection**
4. **Reduce: Use reduce function to combine with existing data**

**Map-Reduce vs Aggregation Pipeline**

**Map-Reduce Advantages:**

More flexible for complex custom logic

Can handle very large datasets Suitable for JavaScript-heavy processing

**Map-Reduce Disadvantages:**

Slower than aggregation pipeline

More complex to write and maintain JavaScript execution overhead

**When to Use Map-Reduce:**

Complex custom aggregations not possible with pipeline

Need for custom JavaScript logic Processing extremely large datasets with custom requirements Performance

**Considerations Indexing:** Ensure proper indexes on query fields

**Query Optimization**: Use specific queries to reduce input documents

**Sharding**: Map-Reduce works well with sharded collections

**Memory:** Be aware of memory limitations for reduce operations

**Alternative**: Consider aggregation pipeline first for better performance