# MongoDB Assignment 2

Data

**Product Collection:**

db.products.insertMany([

{ \_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:**

db.orders.insertMany([

{ \_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

}

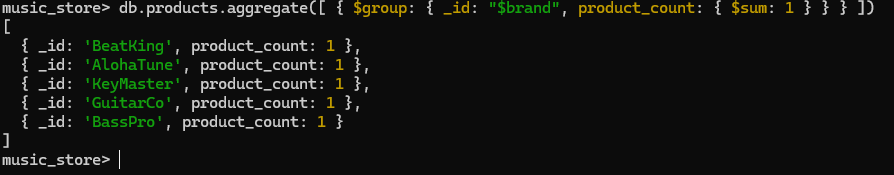
])

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

Ans.

* 1. Count Products per Brand.

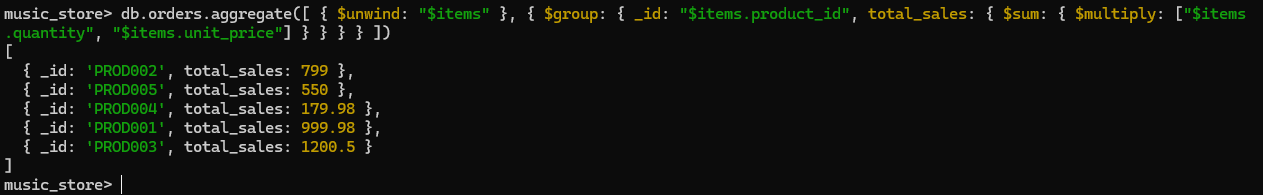
Ans.



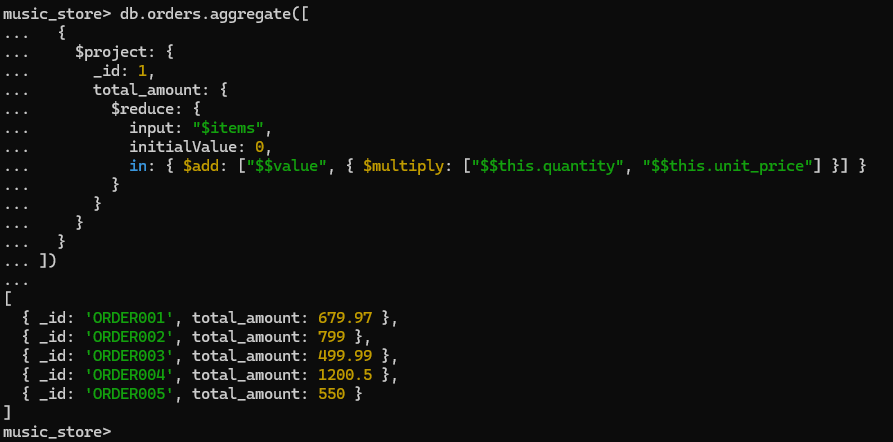
* 1. Find the Average Rating for Each Product.

Ans.

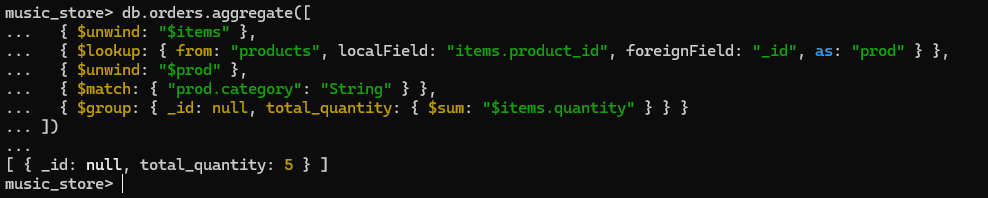
* 1. Calculate Total Sales for Each Product.

Ans.

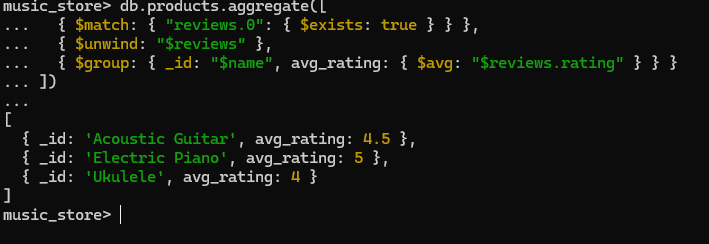
* 1. Recalculate each order's total amount using $reduce.

Ans.

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

Ans.

* 1. Find the average rating for each product that has received at least one review.

Ans.

* 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.

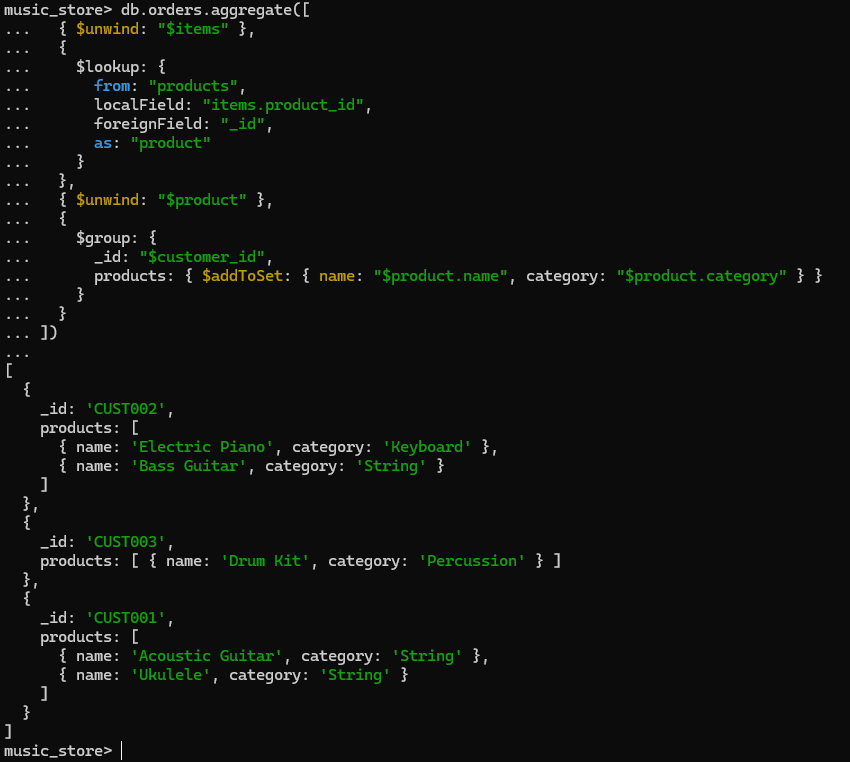
* 1. Retrieve a list of all orders, but instead of just showing product IDs in the items array, replace them with a more detailed object containing the product's name, brand, and category.

Ans.



* 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 in MongoDB**

* MapReduce is a **data processing paradigm** in MongoDB used to handle large-scale aggregations by applying **map** and **reduce** functions written in JavaScript.
* It’s flexible but slower compared to the modern aggregation pipeline.

**Core Components**

1. **Map Function**
   * Applied to each document in the collection.
   * Emits key-value pairs.
   * Example:

function map() {

this.tags.forEach(tag => emit(tag, 1));

}

1. **Reduce Function**
   * Groups values by key and reduces them to a single result.
   * Example:

function reduce(key, values) {

return Array.sum(values);

}

**Input-Output Model in MongoDB**

* **Concept:** 
  + MongoDB follows a **document-based Input-Process-Output (IPO) model**:
    1. Input
* A set of documents from a collection (e.g., products, orders).
* Query conditions (find, $match) decide which documents enter the pipeline.
  + 1. Process
* Transformations using operators or functions:
  + MapReduce (map, reduce, finalize)
  + Aggregation Pipeline ($group, $project, $lookup, $unwind)
    1. Output
* Three forms:
  + Direct result returned to client.
  + Written to a collection (using out in MapReduce or $out in aggregation).
  + Inline temporary results (not stored).

**Example:** Aggregation IPO

db.orders.aggregate([

{ $unwind: "$items" }, // Process: expand items

{ $group: { \_id: "$items.product\_id", // Process: group sales per product

totalQty: { $sum: "$items.quantity" } } },

{ $out: "sales\_summary" } // Output: new collection

])

* **Input**: documents from orders.
* **Process**: $unwind, $group.
* **Output**: sales\_summary collection.