|  |
| --- |
| **Aggregation Pipelining** |
| { "product": "apple", "quantity": 10 }  { "product": "banana", "quantity": 5 }  { "product": "apple", "quantity": 20 }  { "product": "orange", "quantity": 15 } |
| db.orders.aggregate([  { $group: { \_id: "$product", totalQuantity: { $sum: "$quantity" } } }  ])  **Explanation**:   * **$group**: This stage groups documents by the specified **\_id**. Here, we group by **product**. * **$sum**: This accumulator operator calculates the total quantity for each group. |
| Filtering Documents Before Aggregation **Step-by-step explanation:**   1. **Data Setup**: Using the same **orders** collection. 2. **Aggregation Pipeline**: We want to calculate the total quantity only for products with a quantity greater than 10.   db.orders.aggregate([  { $match: { quantity: { $gt: 10 } } },  { $group: { \_id: "$product", totalQuantity: { $sum: "$quantity" } } }  ])   1. **Explanation**:    * **$match**: This stage filters documents before passing them to the next stage. Here, we filter documents where **quantity** is greater than 10. |
| Sorting Aggregated Results  1. **Data Setup**: Same **orders** collection. 2. **Aggregation Pipeline**: We want to find the top three products by total quantity.   db.orders.aggregate([  { $group: { \_id: "$product", totalQuantity: { $sum: "$quantity" } } },  { $sort: { totalQuantity: -1 } },  { $limit: 3 }  ])   1. **Explanation**:    * **$sort**: This stage sorts the documents based on the specified field in descending order (**-1** for descending).    * **$limit**: This stage limits the number of documents passed to the next stage. |
| Projecting Fields in Aggregation  1. **Data Setup**: Using the same **orders** collection. 2. **Aggregation Pipeline**: We want to find the average quantity for each product, along with the total quantity.  db.orders.aggregate([{ $group: { \_id: "$product", totalQuantity: { $sum: "$quantity" } } },{ $project: { \_id: 0, product: "$\_id", totalQuantity: 1, avgQuantity: { $avg: "$totalQuantity" } } }]) |
| Using Unwind to Denormalize Arrays **Data Setup**: Assume we have a collection named **orders** with documents containing an array of **items**. { "\_id": 1, "items": ["apple", "banana", "orange"] }{ "\_id": 2, "items": ["banana", "orange"] } **Aggregation Pipeline**: We want to count occurrences of each item. db.orders.aggregate([{ $unwind: "$items" },{ $group: { \_id: "$items", count: { $sum: 1 } } }]) **Explanation**:   * **$unwind**: This stage deconstructs the array field **items** into separate documents for each element. * **$group**: This stage groups the documents by item and counts occurrences using **$sum**. |
|  |
|  |