# Documentation on Map-Reduce and Input-Output Model in MongoDB

## 1. Map-Reduce in MongoDB

Map-Reduce is a data processing paradigm for condensing large volumes of data into aggregated results. MongoDB supports Map-Reduce to perform complex data processing and aggregation tasks.

It consists of two key functions:

• Map Function: Processes each input document and emits one or more key-value pairs.  
• Reduce Function: Merges all values associated with the same key into a single value.

Map-Reduce is useful for tasks like counting, summing, or averaging data across large collections. While aggregation pipelines are generally more efficient, Map-Reduce is still valuable for custom aggregation logic that cannot be expressed easily in pipelines.

### Example:

db.orders.mapReduce(  
 function() { emit(this.customer\_id, this.total\_amount); }, // Map  
 function(key, values) { return Array.sum(values); }, // Reduce  
 { out: "total\_sales\_per\_customer" }  
)

## 2. Input-Output Model in MongoDB Map-Reduce

The Input-Output (I/O) model in MongoDB's Map-Reduce describes how data flows through the process:

1. Input Phase:  
 • The input documents are selected from a collection using a query.  
 • Each document is passed to the Map function.  
  
2. Map Phase:  
 • The Map function processes each document and emits key-value pairs.  
  
3. Shuffle Phase:  
 • MongoDB groups all emitted values by their keys.  
  
4. Reduce Phase:  
 • The Reduce function takes each key and its grouped values and reduces them to a single value.  
  
5. Output Phase:  
 • The results are written to a collection (specified in 'out') or returned inline.

### Example with Input-Output:

db.orders.mapReduce(  
 function() { emit(this.status, this.total\_amount); }, // Map: emit order status and amount  
 function(key, values) { return Array.sum(values); }, // Reduce: sum amounts per status  
 { out: "sales\_by\_status" } // Output collection  
)

In this example:  
• Input: documents from the 'orders' collection.  
• Output: a new collection 'sales\_by\_status' containing aggregated totals per status.