Name: - Aayush Darange
Course: - Database Management System Lab
Division: -TY CSF 2

Roll No: - 2213789
Batch: - B

Experiment No. -9

Aim - Implement MapReduce example in MongoDB with suitable dataset.

A. Create a sample collection order with 10 documents.
B. Perform the map-reduce operation on the orders collection to group by the cust_id, and calculate the sum of the price for each cust_id.

Software Required - MongoDB

Theory :-

Map-reduce is a data processing paradigm for condensing large volumes of data into useful *aggregated* results. To perform map-reduce operations, MongoDB provides the mapReduce database command.

Map-Reduce Syntax

```
 db.collection.mapReduce( function() {emit(key, value);},

 //Define map function

function(key,values) {return reduceFunction}, {

 //Define reduce function

 out: collection,

 query: document,

sort: document,

 limit: number

 }

)
```

Map-Reduce Syntax Explanation •

The above map-reduce function will query the collection, and then map the output documents to the emit key-value pairs.

After this, it is reduced based on the keys that have multiple values. Here, we have used the following functions and parameters.

• Map: – It is a JavaScript function. It is used to map a value with a key and produces a key-value pair

. • Reduce: – It is a JavaScript function. It is used to reduce or group together all the documents which have the same key.

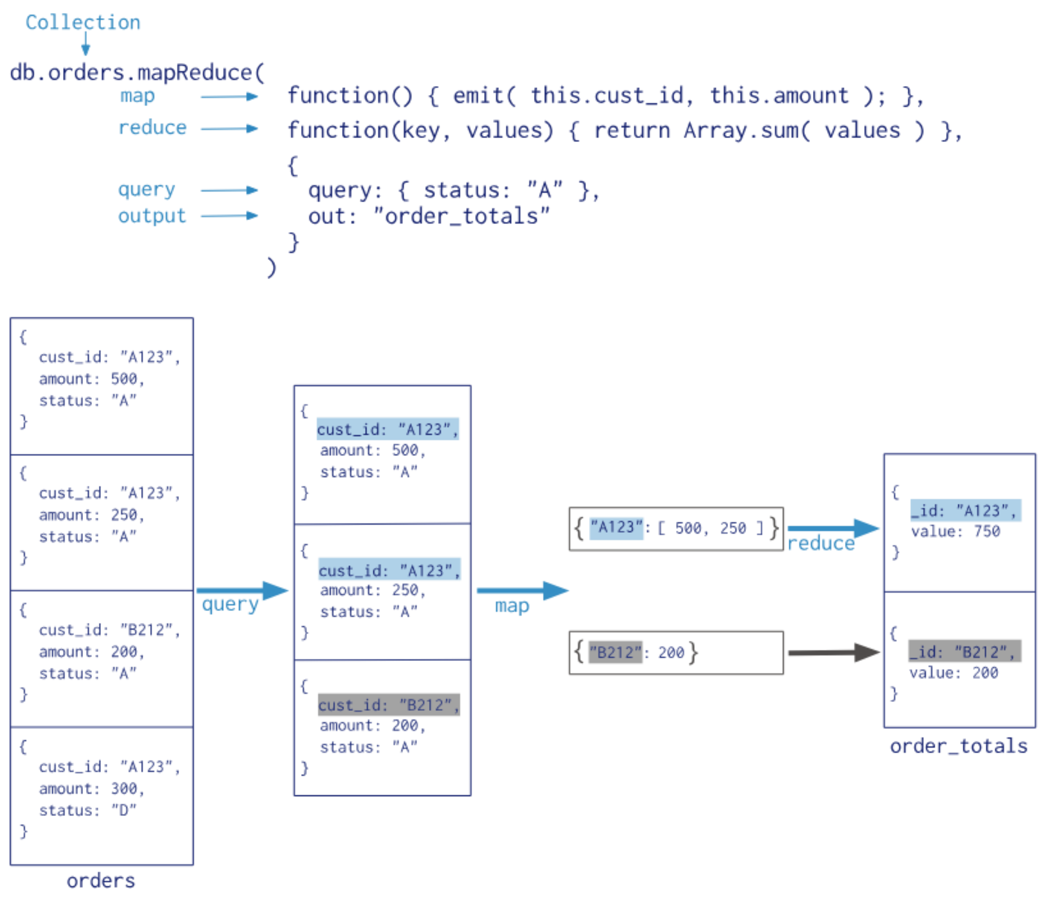
• Out: – It is used to specify the location of the map-reduce query output

. • Query: – It is used to specify the optional selection criteria for selecting documents.

• Sort: – It is used to specify the optional sort criteria.

• Limit: – It is used to specify the optional maximum number of documents which are desired to be returned.

Consider the following map-reduce operation:



```
db.orders.mapReduce(
    function() { emit( this.cust_id, this.amount ); },
    function(key, values) { return Array.sum( values ) },
    {
      query: { status: "A" },
      out: "order_totals"
    }
)
```

```
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
{ cust_id: "A123", amount: 300, status: "D" }

{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }

{ "A123": [ 500, 250 ] }
{ "B212": 200 }

{ _id: "A123", value: 750 }
{ _id: "B212", value: 200 }
```

In this map-reduce operation, MongoDB applies the *map* phase to each input document (i.e. the documents in the collection that match the query condition). The map function emits key-value pairs. For those keys that have multiple values, MongoDB applies the *reduce* phase, which collects and condenses the aggregated data. MongoDB then stores the results in a collection. Optionally, the output of the reduce function may pass through a *finalize* function to further condense or process the results of the aggregation.

Consider the following document structure that stores book details author wise.

• > db.author.save({ "book_title" : "MongoDB Tutorial", "author_name" : "aparajita", "status" : "active", "publish_year": "2016" })

• > db.author.save({ "book_title" : "Software Testing Tutorial", "author_name" : "aparajita", "status" : "active", "publish_year": "2015" })

• > db.author.save({ "book_title" : "Node.js Tutorial", "author_name" : "Kritika", "status" : "active", "publish_year": "2016" })

• > db.author.save({ "book_title" : "PHP7 Tutorial", "author_name" : "aparajita", "status" : "passive", "publish_year": "2016" })

Perform Below Tasks using Mapreduce

1. To select all the active books
2. Group them together on the basis of author_name and Then count the number of books by each author

Let us consider school DB, where the student is a collection, and the collection contains documents, each of which includes a student's name and the marks they received in a particular subject.

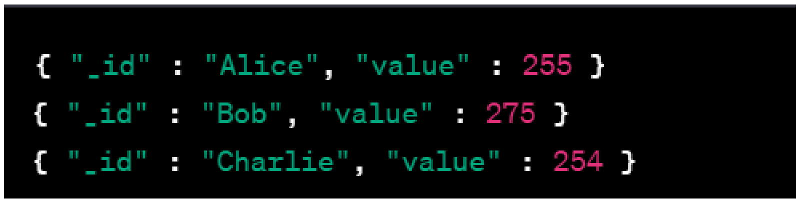
Write the Mapreduce Program to Calculate the Total Marks Secured by each student in all Subjects

Program: -

```
// Step 1: Create a sample 'students' collection with dummy data
db.students.insertMany([
  { name: "Alice", marks: { math: 85, science: 92, history: 78 } },
  { name: "Bob", marks: { math: 92, science: 88, history: 95 } },
  { name: "Charlie", marks: { math: 78, science: 86, history: 90 } }
]);

// Step 2: Define the Map and Reduce functions
var mapFunction = function () {
  for (var subject in this.marks) {
    emit(this.name, this.marks[subject]);
  }
};

var reduceFunction = function (name, marks) {
  return Array.sum(marks);
};

// Step 3: Run the MapReduce operation
```

```
db.students.mapReduce(
  mapFunction,
  reduceFunction,
  {
    out: "total_marks"  // Create a collection to store the results
  }
);

// Step 4: Query the 'total_marks' collection to see the results
db.total_marks.find();

// Step 5: Explain the MapReduce process
// - The map function emits the student's name as the key and each subject's
marks as the value.
// - The reduce function sums up the marks for each student.
// - The results are stored in the 'total_marks' collection.
```

OUTPUT: -



```
{ "_id" : "Alice", "value" : 255 }
{ "_id" : "Bob", "value" : 275 }
{ "_id" : "Charlie", "value" : 254 }
```