SSD Monsoon 2024

Lab 8 (11th Oct) - aggregate, count, distinct, mapReduce

Syntax

map,JavaScript function to map the data

reduce,JavaScript function to reduce the mapped data

out:Output collection or "inline" for results in-memory

query: Optional filter to select documents to process

sort: Optional sorting of documents before applying mapReduce

limit: Optional limit on the number of input documents

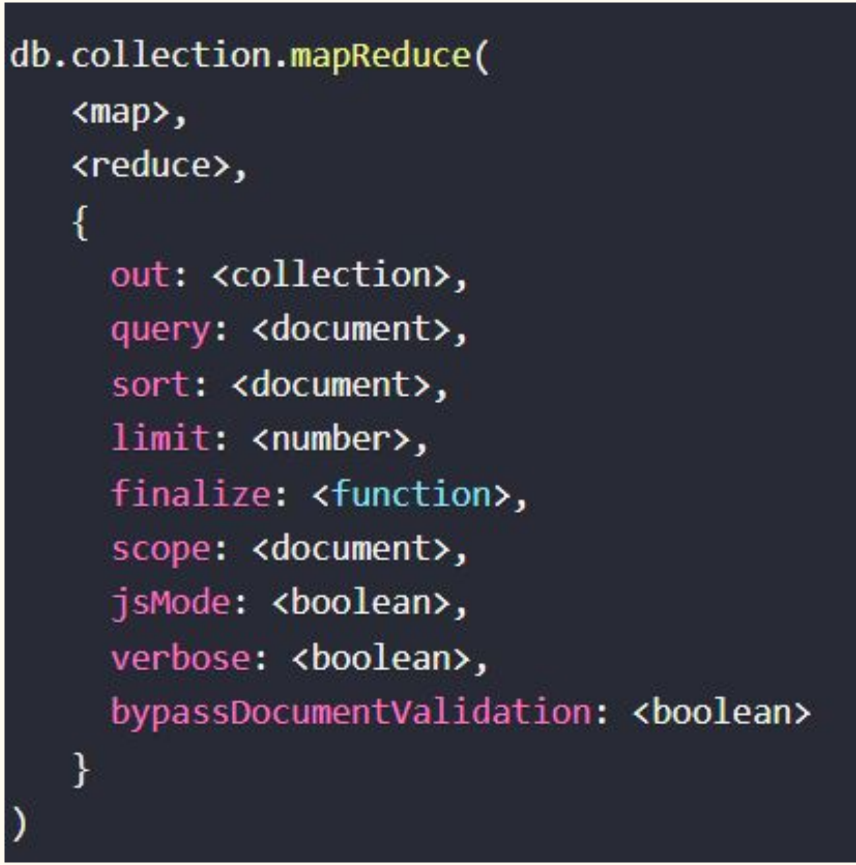
finalize: Optional function to modify the final output

scope:Optional variables accessible within map and reduce functions

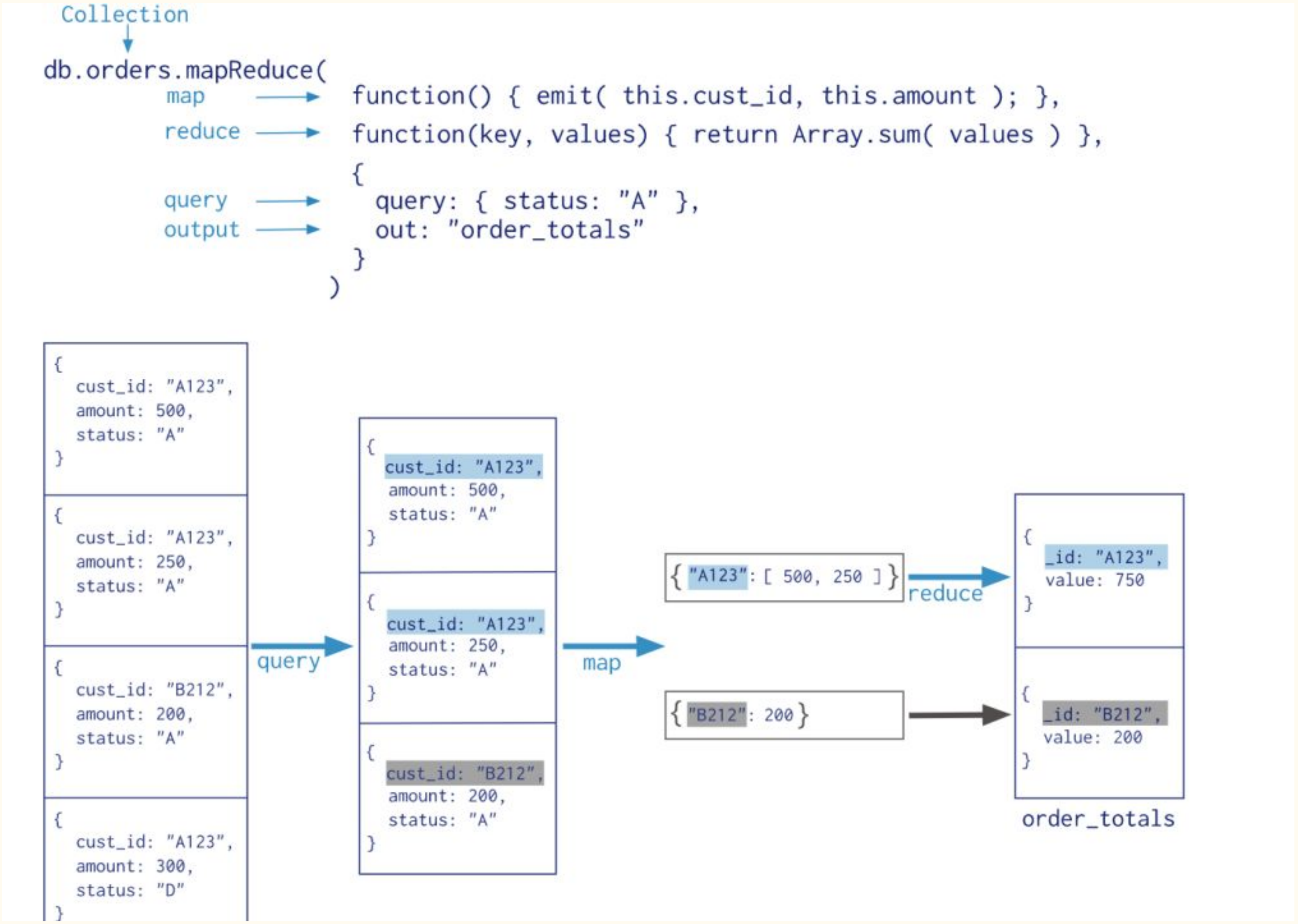
jsMode: Optional. If true, it runs the reduce function in JavaScript mode

verbose: Optional. If true, adds additional statistics to output

bypassDocumentValidation Optional. If true, skips validation for insertion



```
db.collection.mapReduce(
   <map>,
   <reduce>,
   {
     out: <collection>,
     query: <document>,
     sort: <document>,
     limit: <number>,
     finalize: <function>,
     scope: <document>,
     jsMode: <boolean>,
     verbose: <boolean>,
     bypassDocumentValidation: <boolean>
   }
)
```



```
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"
}
```

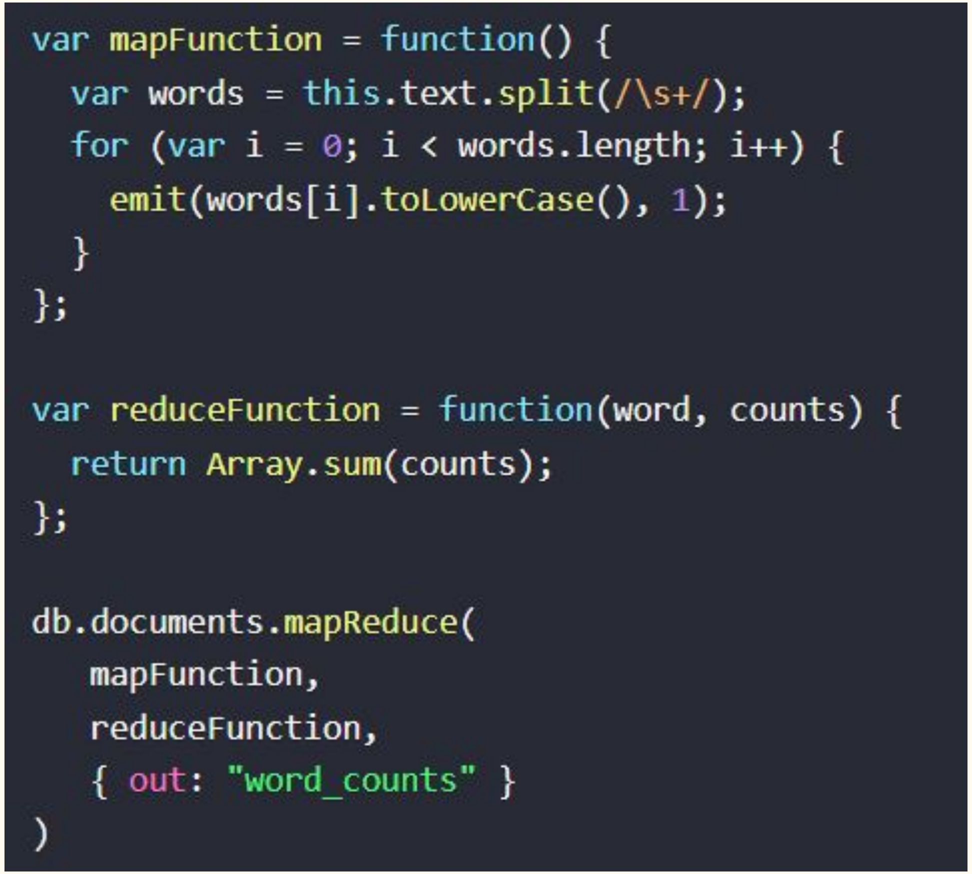
```
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
```

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

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

Counting word occurrences

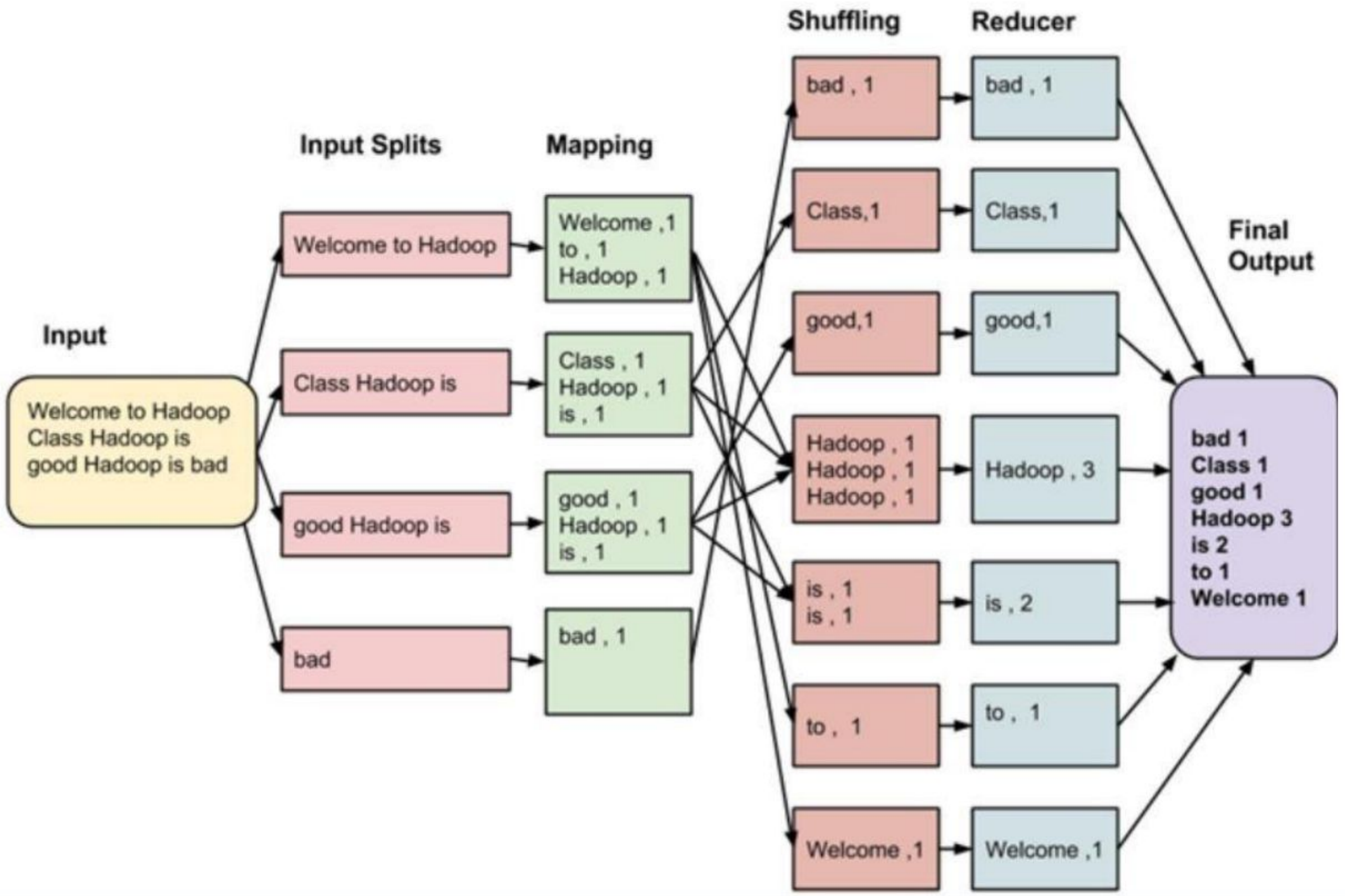
```
db.text.mapReduce(
  function() {
    emit(this.word, 1);
  },
  function(key, values) {
    return Array.sum(values);
  },
  { out: "word_count" }
)
```



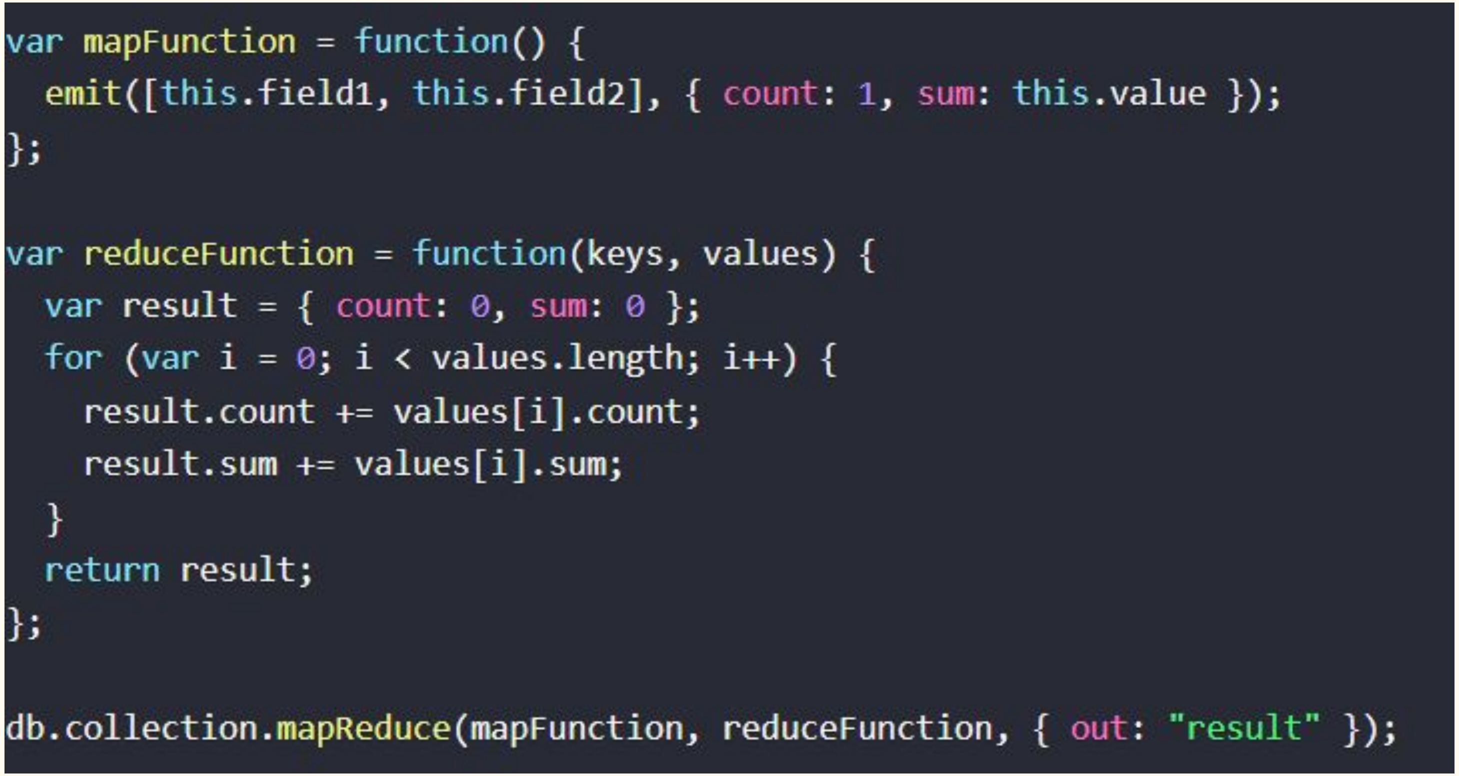
```
var mapFunction = function() {
  var words = this.text.split(/\s+/);
  for (var i = 0; i < words.length; i++) {
    emit(words[i].toLowerCase(), 1);
  }
};

var reduceFunction = function(word, counts) {
  return Array.sum(counts);
};

db.documents.mapReduce(
   mapFunction,
   reduceFunction,
   { out: "word_counts" }
)
```


Shuffling
Reducer
Input Splits
Mapping
Input
Final Output
Welcome to Hadoop Class Hadoop is good Hadoop is bad
Welcome to Hadoop
Class Hadoop is
good Hadoop is
bad
Welcome ,1 to , 1 Hadoop , 1
Class , 1 Hadoop , 1 is , 1
good , 1 Hadoop , 1 is , 1
bad , 1
bad , 1
Class,1
good,1
Hadoop , 1 Hadoop , 1 Hadoop , 1
is , 1 is , 1
to , 1
Welcome ,1
bad , 1
Class,1
good,1
Hadoop , 3
is , 2
to , 1
Welcome ,1
bad 1 Class 1 good 1 Hadoop 3 is 2 to 1 Welcome 1

Group documents by multiple fields and calculate sums for each group.

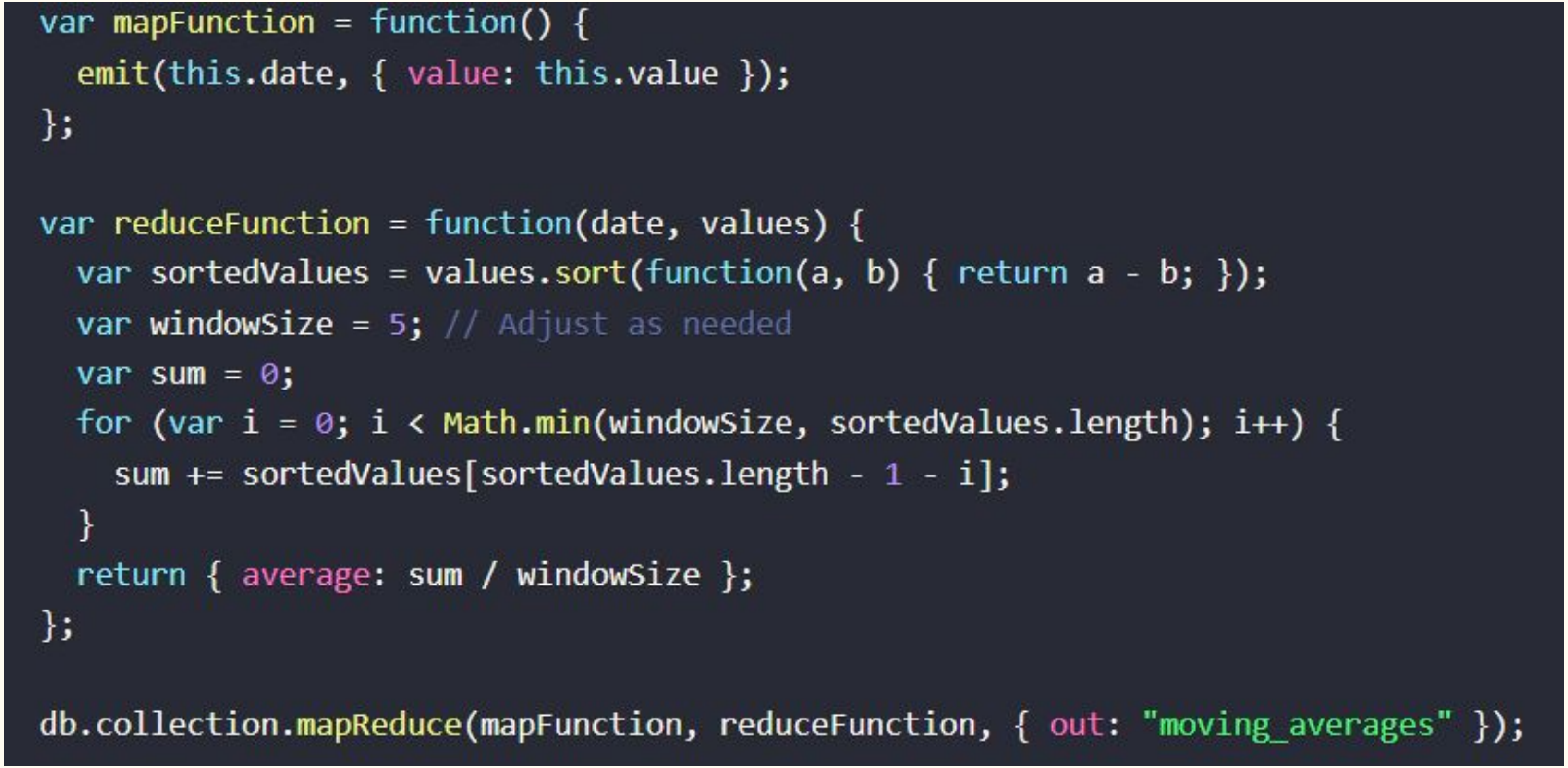


```
var mapFunction = function() {
  emit([this.field1, this.field2], { count: 1, sum: this.value });
};

var reduceFunction = function(keys, values) {
  var result = { count: 0, sum: 0 };
  for (var i = 0; i < values.length; i++) {
    result.count += values[i].count;
    result.sum += values[i].sum;
  }
  return result;
};

db.collection.mapReduce(mapFunction, reduceFunction, { out: "result" });
```

Calculate moving averages over a series of documents

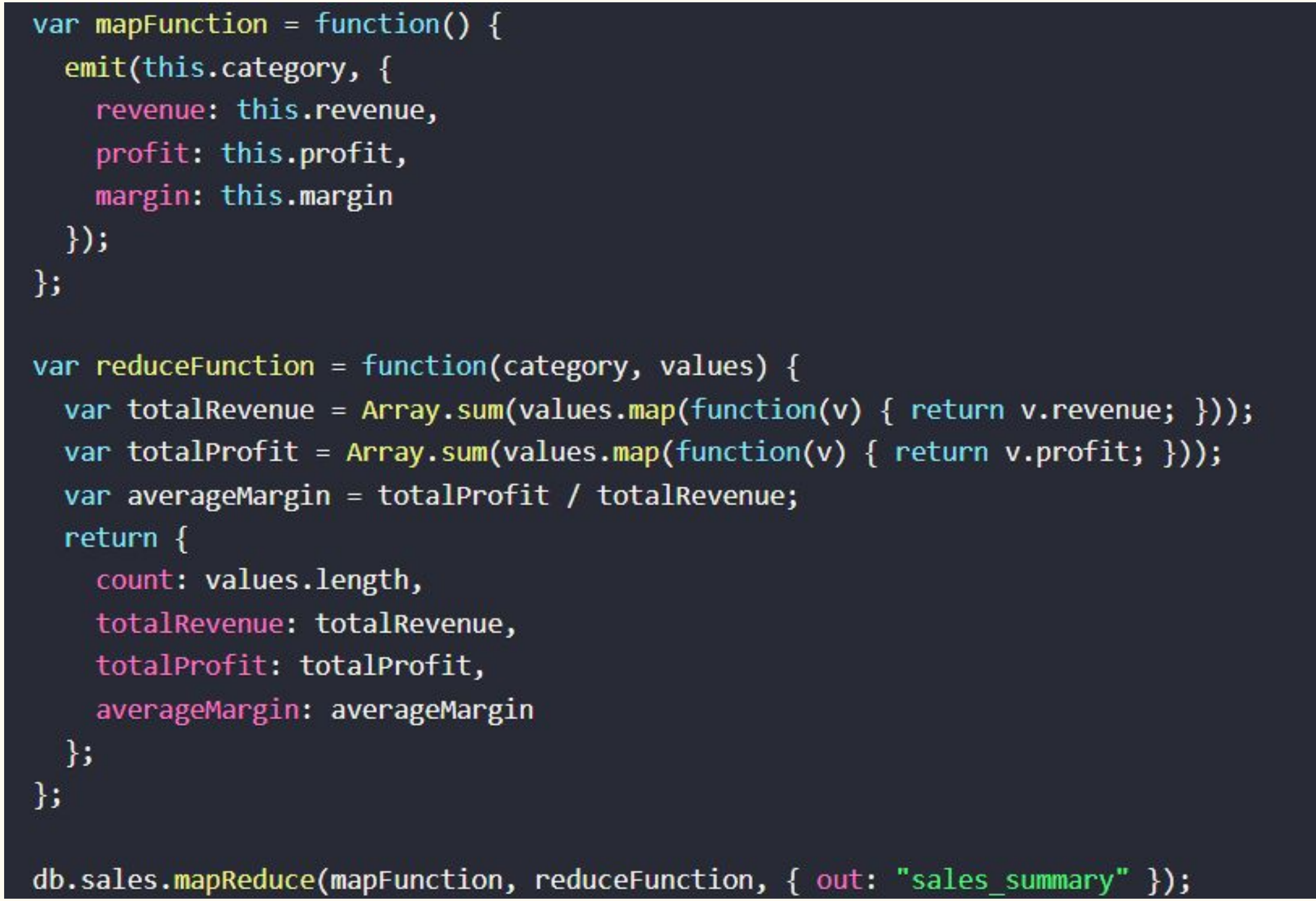


```javascript
var mapFunction = function() {
  emit(this.date, { value: this.value });
};

var reduceFunction = function(date, values) {
  var sortedValues = values.sort(function(a, b) { return a - b; });
  var windowSize = 5; // Adjust as needed
  var sum = 0;
  for (var i = 0; i < Math.min(windowSize, sortedValues.length); i++) {
    sum += sortedValues[sortedValues.length - 1 - i];
  }
  return { average: sum / windowSize };
};

db.collection.mapReduce(mapFunction, reduceFunction, { out: "moving_averages" });
```

Calculating Compound Metrics

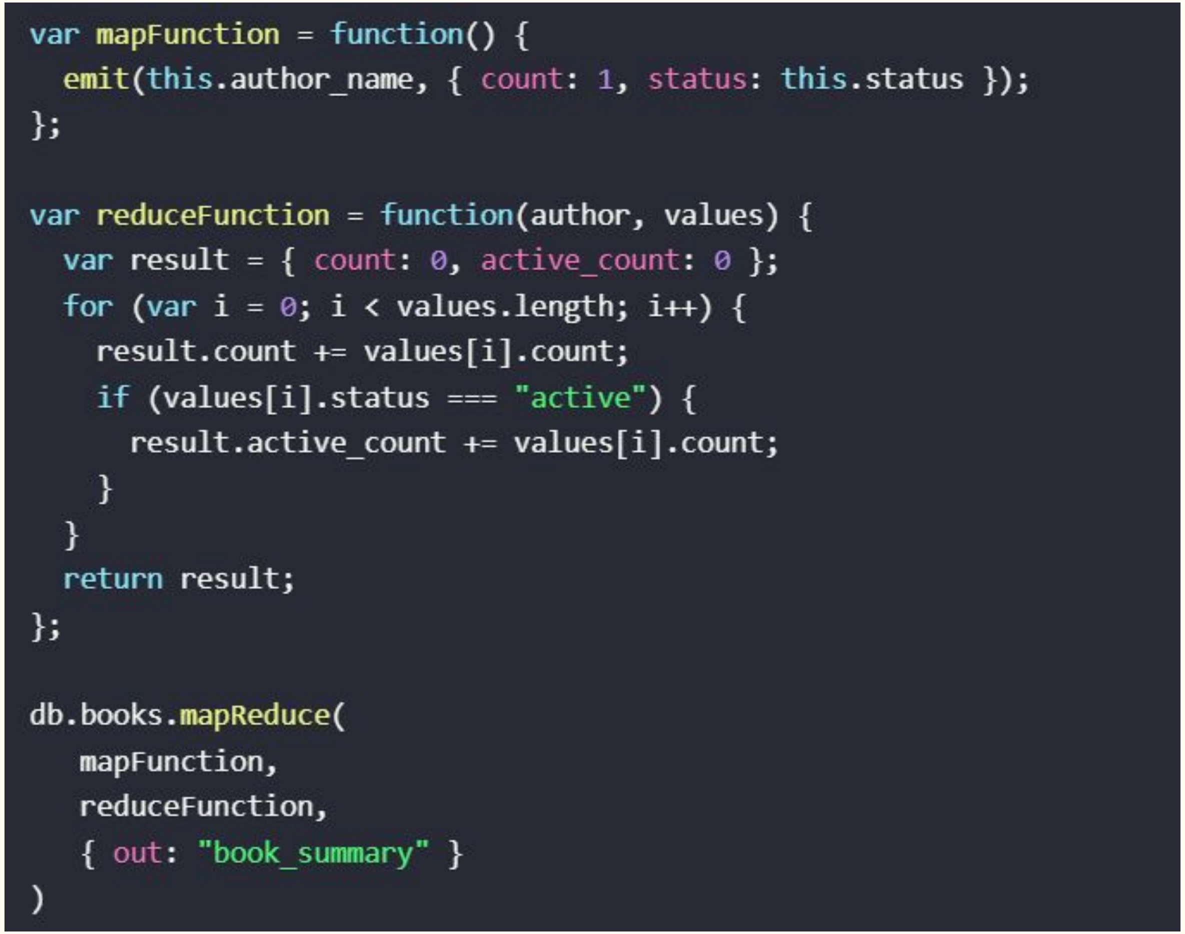


```
var mapFunction = function() {
  emit(this.category, {
    revenue: this.revenue,
    profit: this.profit,
    margin: this.margin
  });
};

var reduceFunction = function(category, values) {
  var totalRevenue = Array.sum(values.map(function(v) { return v.revenue; }));
  var totalProfit = Array.sum(values.map(function(v) { return v.profit; }));
  var averageMargin = totalProfit / totalRevenue;
  return {
    count: values.length,
    totalRevenue: totalRevenue,
    totalProfit: totalProfit,
    averageMargin: averageMargin
  };
};

db.sales.mapReduce(mapFunction, reduceFunction, { out: "sales_summary" });
```

Grouping Books by Author and Counting Active Books

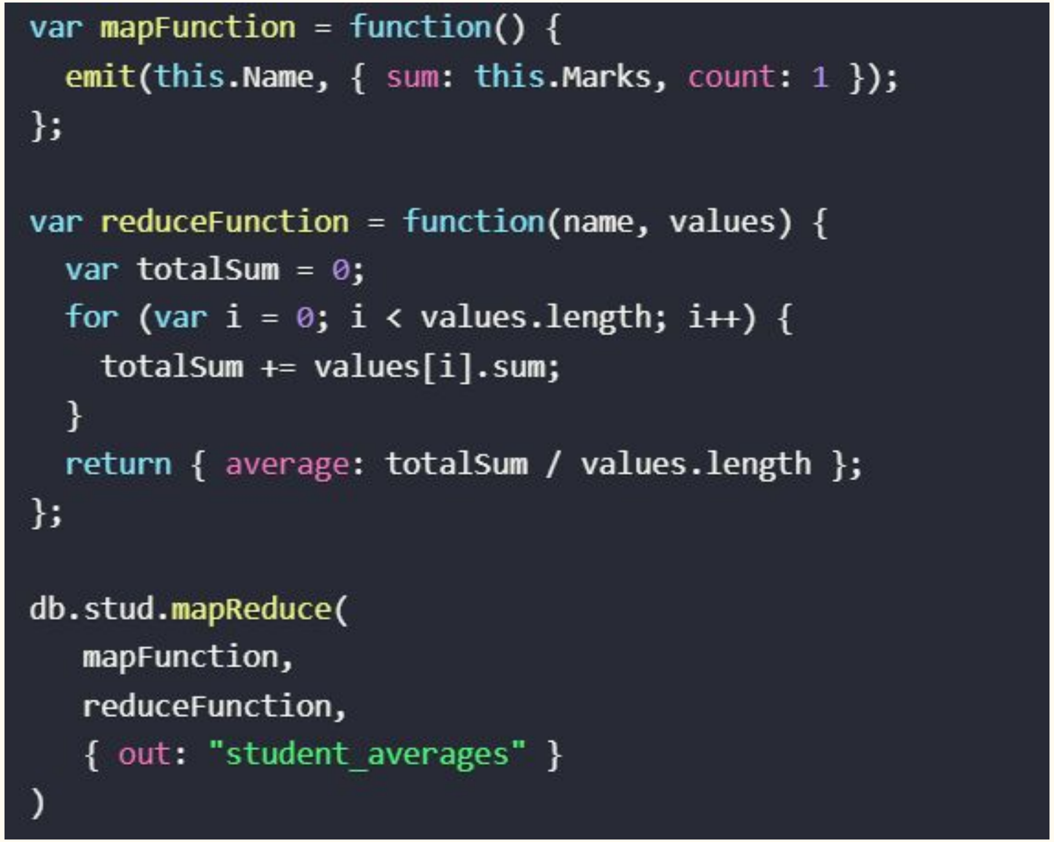


```
var mapFunction = function() {
  emit(this.author_name, { count: 1, status: this.status });
};

var reduceFunction = function(author, values) {
  var result = { count: 0, active_count: 0 };
  for (var i = 0; i < values.length; i++) {
    result.count += values[i].count;
    if (values[i].status === "active") {
      result.active_count += values[i].count;
    }
  }
  return result;
};

db.books.mapReduce(
   mapFunction,
   reduceFunction,
   { out: "book_summary" }
)
```

Calculating Average Marks for Students

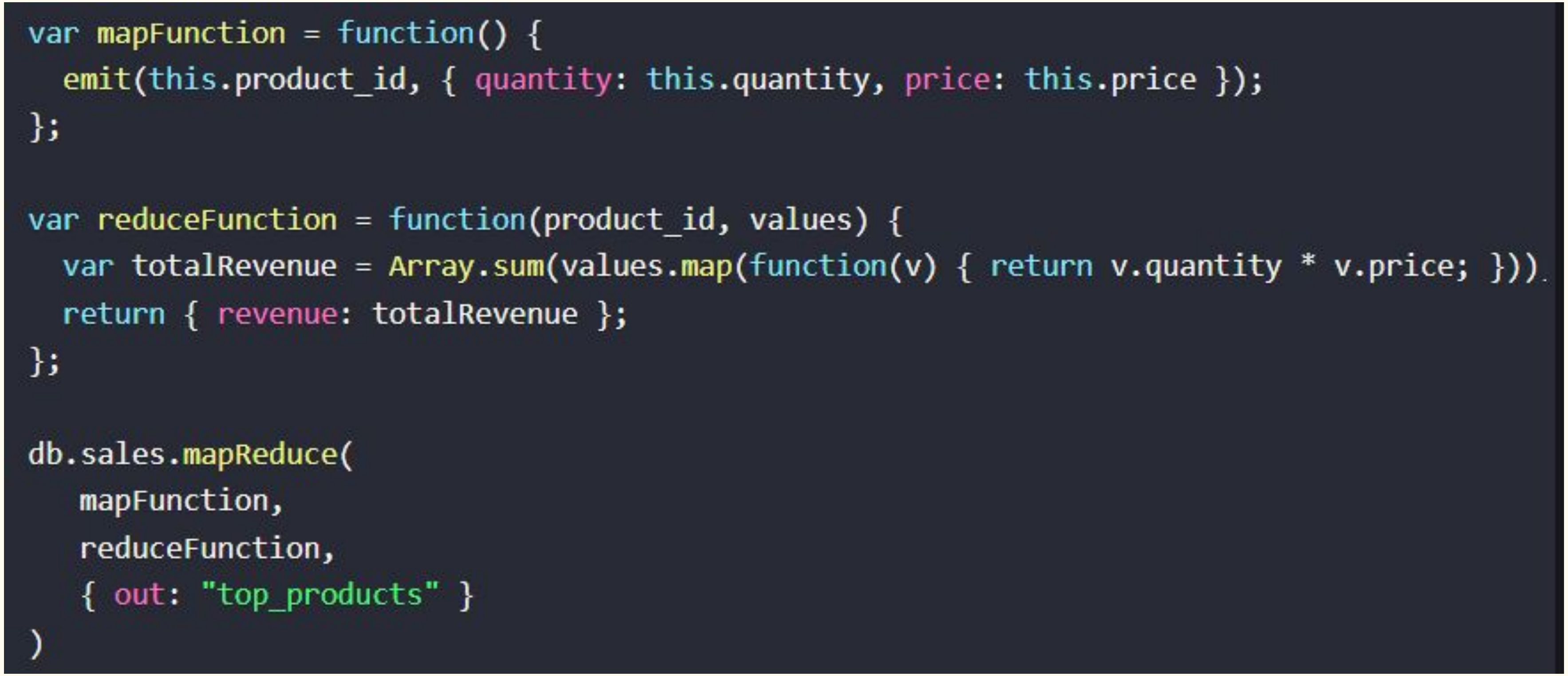


```
var mapFunction = function() {
  emit(this.Name, { sum: this.Marks, count: 1 });
};

var reduceFunction = function(name, values) {
  var totalSum = 0;
  for (var i = 0; i < values.length; i++) {
    totalSum += values[i].sum;
  }
  return { average: totalSum / values.length };
};

db.stud.mapReduce(
   mapFunction,
   reduceFunction,
   { out: "student_averages" }
)
```

Finding Top-Selling Products



```
var mapFunction = function() {
  emit(this.product_id, { quantity: this.quantity, price: this.price });
};

var reduceFunction = function(product_id, values) {
  var totalRevenue = Array.sum(values.map(function(v) { return v.quantity * v.price; }))
  return { revenue: totalRevenue };
};

db.sales.mapReduce(
   mapFunction,
   reduceFunction,
   { out: "top_products" }
)
```