**Wilson, Alex**

**CS214 – Database Technologies**

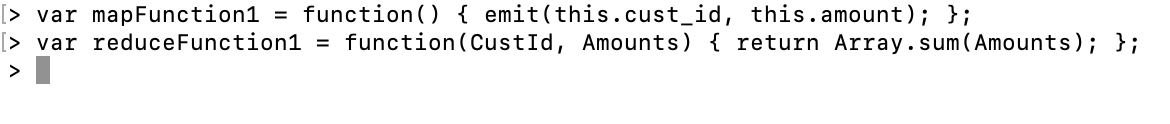
**HOP NUMBER 08**

**Part 1**

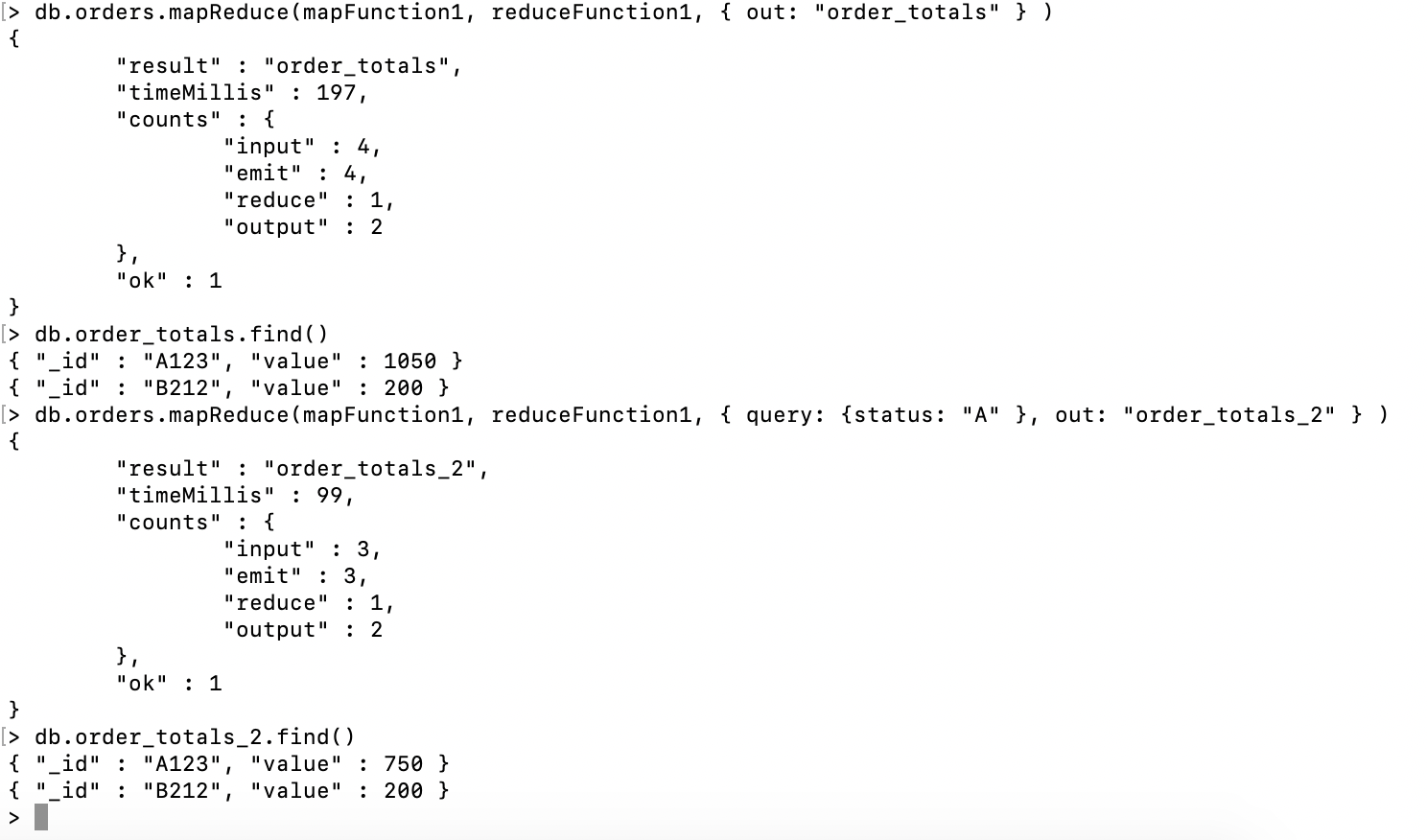
**SCREENSHOT #1**



**SCREENSHOT #2**



**SCREENSHOT #3**

****

**PART 2**

The map reduce function in Mongo DB is way to consolidate records for ease of understanding.

In this Hands-on Practice we created a database, populated some documents into the database, defined a map function and reduce function, then conducted two different map-reduce operations on the created data.

The database we created is called orders:

use orders

Then we populated the database:

> db.orders.insert({cust\_id: "A123", amount: 500, status: "A"})

WriteResult({ "nInserted" : 1 })

> db.orders.insert({cust\_id: "A123", amount: 250, status: "A"})

WriteResult({ "nInserted" : 1 })

> db.orders.insert({cust\_id: "B212", amount: 200, status: "A"})

WriteResult({ "nInserted" : 1 })

> db.orders.insert({cust\_id: "A123", amount: 300, status: "D"})

WriteResult({ "nInserted" : 1 })

Then we defined the functions that we’ll need in order to execute the MapReduce:

> var mapFunction1 = function() { emit(this.cust\_id, this.amount); };

> var reduceFunction1 = function(CustId, Amounts) { return Array.sum(Amounts); };

execute When we go to execute the MapReduce, the combined function will map the data in our database, and reduce it down to its common elements:

Command:

db.orders.mapReduce(mapFunction1, reduceFunction1, { out: "order\_totals" } )

Result:

{ "\_id" : "A123", "value" : 1050 }

{ "\_id" : "B212", "value" : 200 }

Here’s another example, utilizing the same functions:

Command:

> db.orders.mapReduce(mapFunction1, reduceFunction1, { query: {status: "A" }, out: "order\_totals\_2" } )

Result:

{ "\_id" : "A123", "value" : 750 }

{ "\_id" : "B212", "value" : 200 }