NAME:AVINASH BHIWALE ROLL NO.:T1851050 PRN NO.:71901284J DIV:A

Assignment No. 15

AIM: Implement Map reduce example with suitable example.

Theory:

MapReduce is a generic multi-phase data aggregation modality for processing quantities of data. MongoDB provides map-reduce with the MapReduce database command.

The *map* function is a JavaScript function that associates or **"***maps***"** a value with a key and *emits* the key and value pair during a *map-reduce* operation.

In general, map-reduce operations have two phases: a map stage that processes each document and emits one or more objects for each input document, and reduce phase that combines the output of the map operation. Optionally, map-reduce can have a finalize stage to make final modifications to the result. Like other aggregation operations, map-reduce can specify a query condition to select the input documents as well as sort and limit the results.

Map-reduce uses custom JavaScript functions to perform the map and reduce operations, as well as the optional finalize operation. While the custom JavaScript provides great flexibility compared to the aggregation pipeline, in general, MapReduce is less efficient and more complex than the aggregation pipeline.

Additionally, map-reduce operations can have output sets that exceed the 16 megabyte output limitation of the aggregation pipeline.

Example:

Problem: A king wants to count the total population in his country.

Solution 1: He can send one person to count the population. The assigned person will visit every city serially and return with the total population in the country.

Solution 2: King sends one person to each city. They will count the population in each city and after returning to kingdom the total count will reduce to single count (By adding population of each city). Here all persons are counting the population in every city.

First solution is serial and second is parallel solution.

Assuming there are 10 cities in the kingdom, if it takes one day to count the population in each city then counting the total population takes 10 days in solution 1 and only one day in solution 2.

In map Reduce we have to write 3 functions:

1. Map Function (Ex: Person to each city to count population)

2. Reduce Function (Ex: Reducing the total population count to single value.)

3. Map Reduce Function (It will create a new collection it contains the total population)

MapReduce Operation with Example:

We first create an orders collection and apply MapReduce operation on the same.

db.orders.insert({cust_id: "abc123", ord_date: new Date("Oct 04, 2012“), price: 25})

(Assume such 10 documents are inserted)

This collection contains a customer id, order date and price.
Now we can write a MapReduce function which will return the Total Price per Customer.
Step 1: Map

Map function to process each input document:

```
var mapFunction1 = function() {
   emit(this.cust_id, this.price);

 };
```

Above function maps the price to the cust_id for each document and emits the cust_id and price pair.

Step 2: Reduce

```
var reduceFunction1 = function(keyCustId, valuesPrices)
{ return Array.sum(valuesPrices);

    };
```

Above function defines the corresponding reduce function with two arguments keyCustId and valuesPrices.

The valuesPrices is an array whose elements are the price values emitted by the map function and grouped by keyCustId. The function reduces the valuesPrice array to the sum of its elements.

Step 3: MapReduce

This function performs the map-reduce on all documents in the orders collection using themapFunction1 map function and the reduceFunction1 reduce function.

```
db.orders.mapReduce(

    mapFunction1,

    reduceFunction1,

   { out: "map_example" }

    )
```

This operation outputs the results to a collection named map_example.

Conclusion:

Implement Map reduce operation in MongoDB.

Output:

```
connecting to: test
> db.createCollection("order")
{ "ok" : 1 }
> show collections;
Rutuja
employe
e order
rutuja
shop
student
system.indexes
> db.order.insert({"id":1,"cust_id":1,"itm":bat,"price":1000})
2019-10-18T19:42:20.772+0530 ReferenceError: bat is not defined
> db.order.insert({"id":1,"cust_id":1,"itm":"bat","price":1000
}) WriteResult({ "nInserted" : 1 })
> db.order.insert({"id":2,"cust_id":2,"itm":"ball","price":100
}) WriteResult({ "nInserted" : 1 })
> db.order.insert({"id":3,"cust_id":3,"itm":"pen","price":10
}) WriteResult({ "nInserted" : 1 })
> db.order.insert({"id":4,"cust_id":4,"itm":"shirt","price":500
}) WriteResult({ "nInserted" : 1 })
> db.order.insert({"id":4,"cust_id":4,"itm":"jeans","price":5,00
0}) 2019-10-18T19:44:06.947+0530 SyntaxError:
Unexpected token }
```

```
> db.order.insert({"id":4,"cust_id":4,"itm":"jeans","price":500
0}) WriteResult({ "nInserted" : 1 })
> db.order.insert({"id":5,"cust_id":5,"itm":"gloves","price":70
0}) WriteResult({ "nInserted" : 1 })
> db.order.insert({"id":6,"cust_id":6,"itm":"mobile","price":7000
}) WriteResult({ "nInserted" : 1 })
> db.order.insert({"id":7,"cust_id":7,"itm":"helmet","price":4000
}) WriteResult({ "nInserted" : 1 })
> db.order.find().pretty()
{
        "_id"                                    :
        ObjectId("5da9c84f30d45d57f00a68f2"),
        "id" : 1,
        "cust_id" : 1,
        "itm"  :  "bat",
        "price"       :
        1000
}
{
        "_id" : ObjectId("5da9c86e30d45d57f00a68f3"),
        "id" : 2,
        "cust_id"    :
        2,    "itm"    :
        "ball",
        "price" : 100
}
{
        "_id" : ObjectId("5da9c88030d45d57f00a68f4"),
        "id" : 3,
        "cust_id"    :
        3,    "itm"    :
        "pen",
        "price" : 10
}
{
        "_id" : ObjectId("5da9c89630d45d57f00a68f5"),
        "id" : 4,
        "cust_id" : 4,
        "itm" : "shirt",
        "price" : 500
}
{
        "_id" : ObjectId("5da9c8b430d45d57f00a68f6"),
        "id" : 4,
        "cust_id" : 4,
        "itm" :
        "jeans",
        "price" : 5000
}
{
        "_id" : ObjectId("5da9c8d130d45d57f00a68f7"),
"id" : 5,
"cust_id" : 5,
        "itm" :
        "gloves",
        "price" : 700
}
{
        "_id" : ObjectId("5da9c8e830d45d57f00a68f8"),
        "id" : 6,
        "cust_id" : 6,
        "itm" : "mobile",
        "price" : 7000
}
{
        "_id" : ObjectId("5da9c90430d45d57f00a68f9"),
        "id" : 7,
        "cust_id" : 7,
        "itm" : "helmet",
        "price" : 4000
}
> var     mapFunction=functon(){emit(this.cust_id,this.price)}
2019- 10-18T19:47:57.338+0530 SyntaxError: Unexpected
token        {        >        var        mapFunction=functon()
{emit(this.cust_id,this.price)}
2019-10-18T19:48:25.004+0530 SyntaxError: Unexpected token
{ > var mapFunction=functon() {emit(this.custid,this.price)}
2019-10-18T19:48:50.998+0530 SyntaxError: Unexpected token {
> var mapFunction=functon(){emit(this.cust_id,this.price)};
2019-10-18T19:49:29.118+0530 SyntaxError: Unexpected token {
> var mapFunction=functon(){emit(this.cust_id,this.price)};
2019-10-18T19:49:39.212+0530 SyntaxError: Unexpected token
{ > var mapFunction=functon()emit(this.cust_id,this.price)
2019-10-18T19:50:33.611+0530 SyntaxError: Unexpected
identifier > var mapFunction=functon()
emit(this.cust_id,this.price)
2019-10-18T19:50:45.341+0530 SyntaxError: Unexpected
identifier > var mapFunction=functon()
emit(this.cust_id,this.price)
2019-10-18T19:50:53.471+0530 SyntaxError: Unexpected
identifier > var mapFunction=functon().emit(this.cust_id,this.price)
2019-10- 18T19:51:02.446+0530 ReferenceError: functon is not
defined
> var mapFunction=functon(){emit(this.cust_id,this.price);};
2019- 10-18T19:52:04.992+0530 SyntaxError: Unexpected
token {
> var mapFunction=function() {emit(this.cust_id,this.price)}
> var reduceFunction=function(keycust_id,valueprice) {return Array.sum(valueprice);};
> db.order.mapReduce(mapFunction,reduceFunction,{out:"total"})
{
        "result" : "total",
        "timeMillis" : 408,
        "counts" : {
                "input" : 8,
                "emit" : 8,
"reduce" : 1,
"output" : 7
```

```
        },
        "ok" : 1
}
> db.total.find()
{ "_id" : 1, "value" : 1000 }
{ "_id" : 2, "value" : 100 }
{ "_id" : 3, "value" : 10 }
{ "_id" : 4, "value" : 5500 }
{ "_id" : 5, "value" : 700 }
{ "_id" : 6, "value" : 7000 }
{ "_id" : 7, "value" : 4000 }
> show collections; Rutuja
employee
order rutuja
shop student
system.index
es total


> db.total.insert("cust_id":1,"value":1000)
2019-10-18T20:16:03.015+0530 SyntaxError: Unexpected token
: > db.total.insert({"cust_id":1,"value":1000})
WriteResult({ "nInserted" : 1 })
> db.total.insert({"cust_id":2,"value":100
}) WriteResult({ "nInserted" : 1 })
> db.total.insert({"cust_id":3,"value":10
}) WriteResult({ "nInserted" : 1 })
> db.total.insert({"cust_id":4,"value":500
}) WriteResult({ "nInserted" : 1 })
> db.total.insert({"cust_id":4,"value":500
0}) WriteResult({ "nInserted" : 1 })
> db.total.insert({"cust_id":5,"value":700
}) WriteResult({ "nInserted" : 1 })
> db.total.insert({"cust_id":6,"value":700
0}) WriteResult({ "nInserted" : 1 })
> db.total.insert({"cust_id":7,"value":400
0}) WriteResult({ "nInserted" : 1 })
> db.total.find()
{ "_id" : 1, "value" : 1000 }
{ "_id" : 2, "value" : 100 }
{ "_id" : 3, "value" : 10 }
{ "_id" : 4, "value" : 5500 }
{ "_id" : 5, "value" : 700 }
{ "_id" : 6, "value" : 7000 }
{ "_id" : 7, "value" : 4000 }
{ "_id" : ObjectId("5da9d04230d45d57f00a68fa"), "cust_id" : 1, "value" : 1000 }
{ "_id" : ObjectId("5da9d04e30d45d57f00a68fb"), "cust_id" : 2, "value" : 100 }
{ "_id" : ObjectId("5da9d05730d45d57f00a68fc"), "cust_id" : 3, "value" : 10 }
{ "_id" : ObjectId("5da9d07430d45d57f00a68fd"), "cust_id" : 4, "value" : 500 }
{ "_id" : ObjectId("5da9d07a30d45d57f00a68fe"), "cust_id" : 4, "value" : 5000 }
{ "_id" : ObjectId("5da9d08830d45d57f00a68ff"), "cust_id" : 5, "value" : 700 }
{ "_id" : ObjectId("5da9d09030d45d57f00a6900"), "cust_id" : 6, "value" : 7000 }
{ "_id" : ObjectId("5da9d09a30d45d57f00a6901"), "cust_id" : 7, "value" : 4000 }
> db.total.update({$set:{"cust_id":,"value":}})
2019-10-18T20:21:53.680+0530 SyntaxError:
Unexpected token , >
db.total.update({$set:{"cust_id"},{"value"}})
2019-10-18T20:22:23.317+0530 SyntaxError:
Unexpected token } >
db.total.update({$set:{"cust_id","value"}})
2019-10-18T20:22:35.974+0530 SyntaxError:
Unexpected token , >
db.total.update({$set:{"cust_id":1,"value:1000"}})
2019-10-18T20:23:16.980+0530 SyntaxError:
Unexpected token } >
```