neo4j

cassandra

NoSQL

redis

Elastic Search

MongoDB

MongoDB

- מסד הנתונים המוביל בעולם בקטגוריית NoSQL
- מערכת ניהול נתונים המבוססת על מבנה של מסמך (Document Store).
- הנתונים נשמרים כאוסף של נתונים בצורה של key-value לכל מפתח יש קובץ נתונים.
- הנתונים נכתבים בדרך כלל ב-XML או JSON.
- לבסיס הנתונים יש יכולת 'הבנה' של נתונים (ולא רק של המפתח).

השוואה ל-SQL

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)

- כדי לתרגל את המסד נתונים הזה נשתמש באתר שיש בו הדגמה:

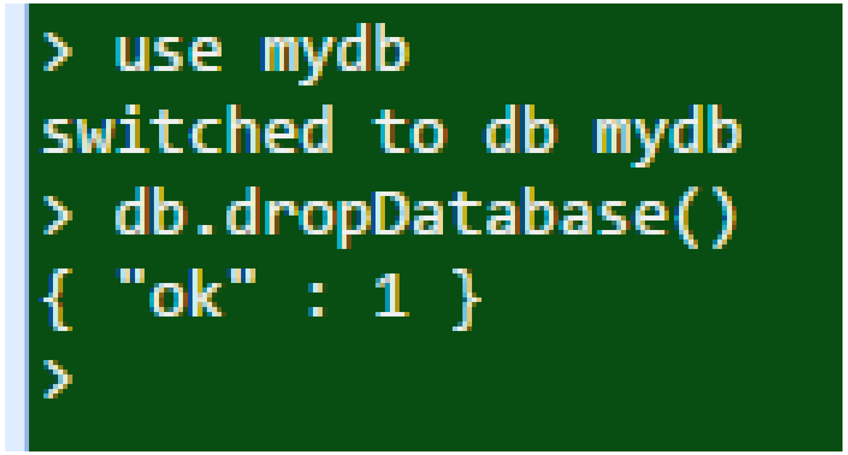
https://www.jdoodle.com/online-mongodb-terminal

יצירת מסד נתונים (או מעבר למסד נתונים, אם קיים)

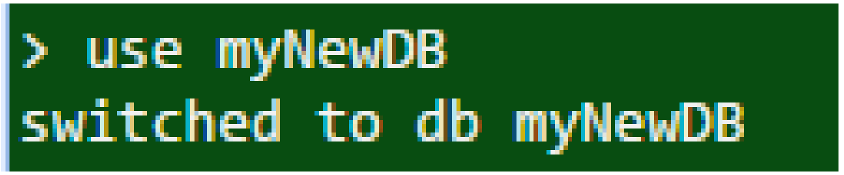
use DATABASE_NAME

- מחיקת מסד נתונים

db.dropDatabase()



```
> use mydb
switched to db mydb
> db.dropDatabase()
{ "ok" : 1 }
>
```

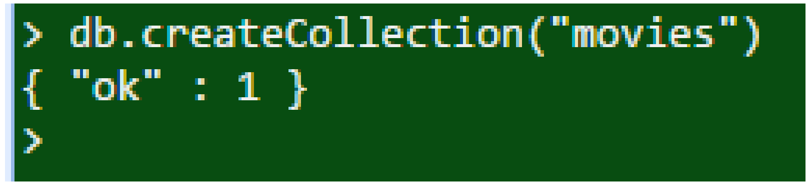


```
> use myNewDB
switched to db myNewDB
```

Collection

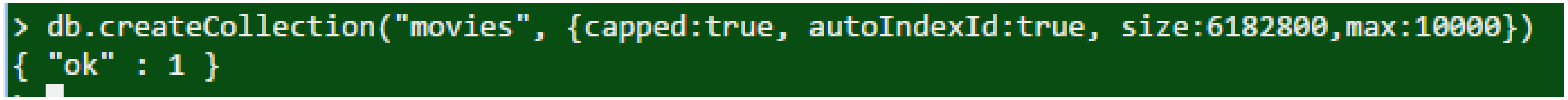
- יצירת collection (טבלה):

db.createCollection(name, options)



```
> db.createCollection("movies")
{ "ok" : 1 }
>
```

- לא חייבים ליצור collection, אפשר ליצור רק על מנת להגדיר פרמטרים נוספים:

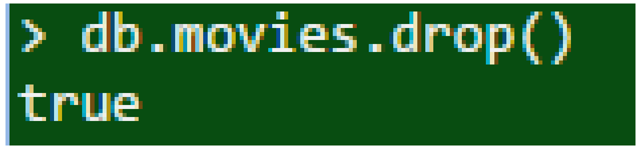


```
> db.createCollection("movies", {capped:true, autoIndexId:true, size:6182800,max:10000})
{ "ok" : 1 }
```

Collection

- מחיקת collection:

db.COLLECTION_NAME.drop()



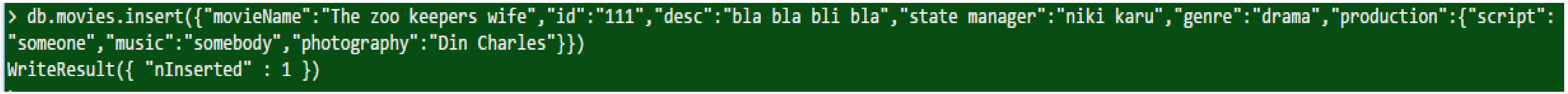
```
> db.movies.drop()
true
```

Document

- יצירת document – רשומה:

```
db.COLLECTION_NAME.insert(document)

db.movies.insert({"movieName": "The Zookeepers Wife", "id": "111",
                 "desc": "bla blab la bla",  "stage manager":"Nici Karu",
                 "genre" : "drama",
                 "production": {
                    "script": "someone",
                    "music": "somebody",
                    "Photography": "din charles"}
                 })
```



```
> db.movies.insert({"movieName":"The zoo keepers wife","id":"111","desc":"bla bla bli bla","state manager":"niki karu","genre":"drama","production":{"script":
"someone","music":"somebody","photography":"Din Charles"}})
WriteResult({ "nInserted" : 1 })
```

Documents

- ניתן ליצור מספר רשומות:

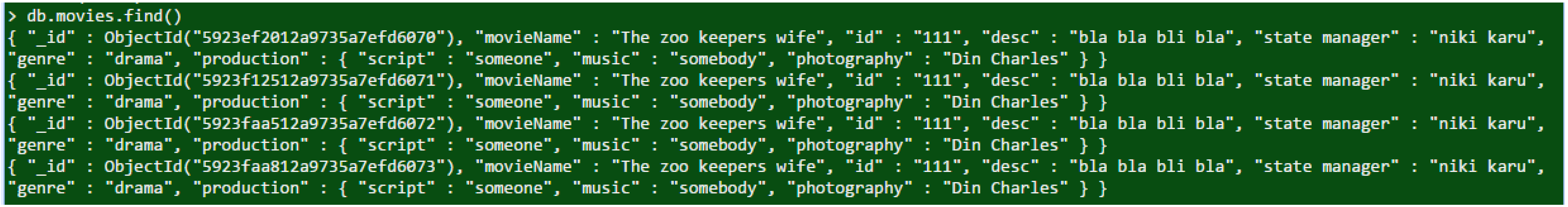
```
db.COLLECTION_NAME.insert([document, document,…])


db.movies.insert([{"movieName": " The SpongeBob Movie", "id": "222",
"desc": "bla blab la bla", "stage manager":"Paul Tibbit", "genre" : "children",
"production": { "script": "someone",  "music": "somebody",  "Photography":
"din charles"} }, {"movieName": "Baby Boss",  "id": "333",
"desc": "bla blab la bla",  "stage manager":"Tom makarts", "genre""comedy",
"production": {  "script": "someone",  "music": "somebody",   "Photography":
"din charles"} }, {"movieName": "Ben Gurion Epilogue",       "id": "444",
"desc": "bla blab la bla",  "stage manager":"Yariv muzar", "genre" : "israeli",
"production": { "script": "someone", "music": "somebody",  "Photography":
"din charles"} }])
```

Query the Document

- שליפת נתונים:

db.COLLECTION_NAME.find()

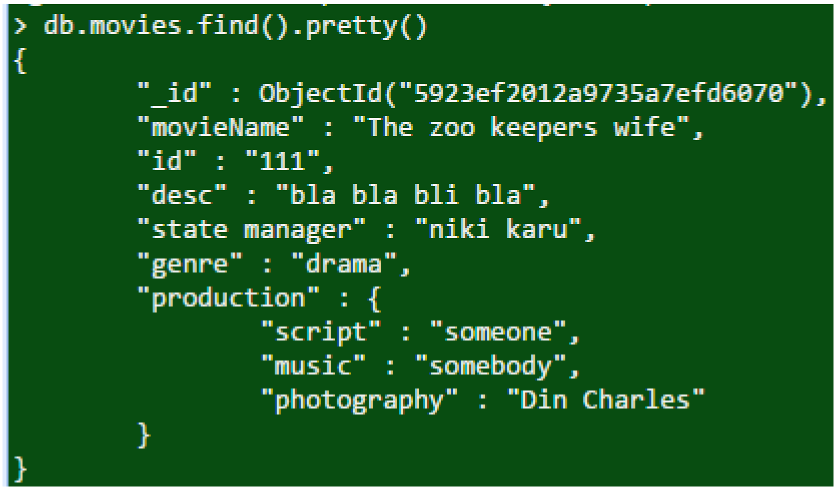


```
> db.movies.find()
{ "_id" : ObjectId("5923ef2012a9735a7efd6070"), "movieName" : "The zoo keepers wife", "id" : "111", "desc" : "bla bla bli bla", "state manager" : "niki karu",
"genre" : "drama", "production" : { "script" : "someone", "music" : "somebody", "photography" : "Din Charles" } }
{ "_id" : ObjectId("5923f12512a9735a7efd6071"), "movieName" : "The zoo keepers wife", "id" : "111", "desc" : "bla bla bli bla", "state manager" : "niki karu",
"genre" : "drama", "production" : { "script" : "someone", "music" : "somebody", "photography" : "Din Charles" } }
{ "_id" : ObjectId("5923faa512a9735a7efd6072"), "movieName" : "The zoo keepers wife", "id" : "111", "desc" : "bla bla bli bla", "state manager" : "niki karu",
"genre" : "drama", "production" : { "script" : "someone", "music" : "somebody", "photography" : "Din Charles" } }
{ "_id" : ObjectId("5923faa812a9735a7efd6073"), "movieName" : "The zoo keepers wife", "id" : "111", "desc" : "bla bla bli bla", "state manager" : "niki karu",
"genre" : "drama", "production" : { "script" : "someone", "music" : "somebody", "photography" : "Din Charles" } }
```

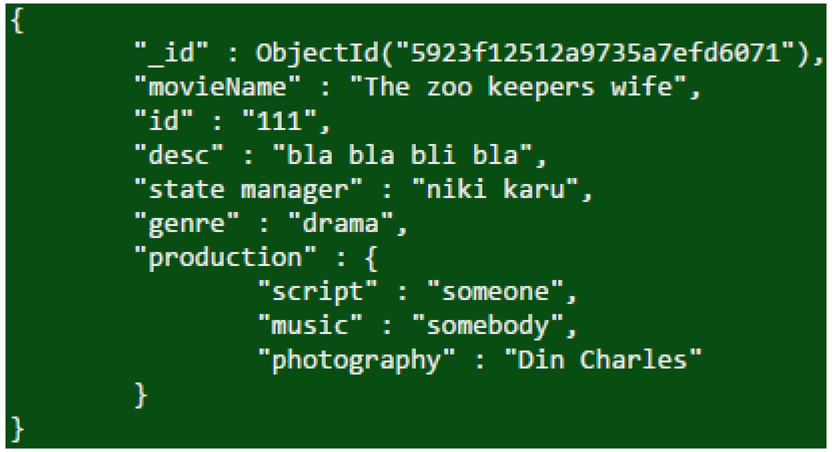
Query the Document

- שליפת הנתונים בצורה מסודרת:

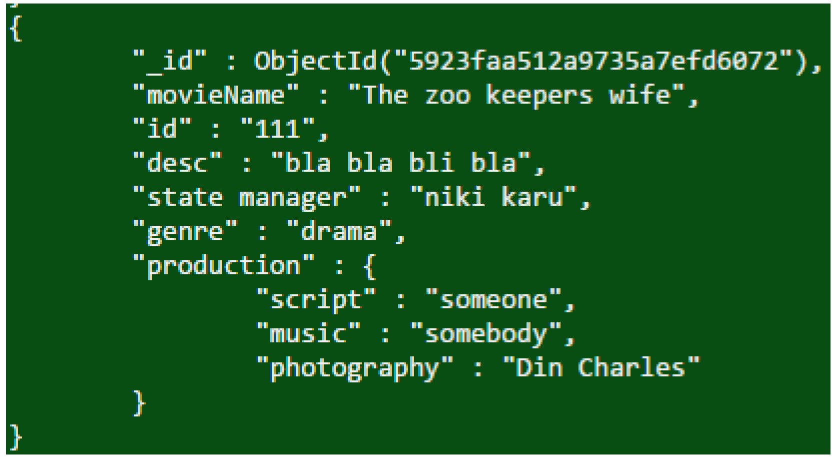
db.mycol.find().pretty()



```
> db.movies.find().pretty()
{
        "_id" : ObjectId("5923ef2012a9735a7efd6070"),
        "movieName" : "The zoo keepers wife",
        "id" : "111",
        "desc" : "bla bla bli bla",
        "state manager" : "niki karu",
        "genre" : "drama",
        "production" : {
                "script" : "someone",
                "music" : "somebody",
                "photography" : "Din Charles"
        }
}
```



```
{
        "_id" : ObjectId("5923f12512a9735a7efd6071"),
        "movieName" : "The zoo keepers wife",
        "id" : "111",
        "desc" : "bla bla bli bla",
        "state manager" : "niki karu",
        "genre" : "drama",
        "production" : {
                "script" : "someone",
                "music" : "somebody",
                "photography" : "Din Charles"
        }
}
```

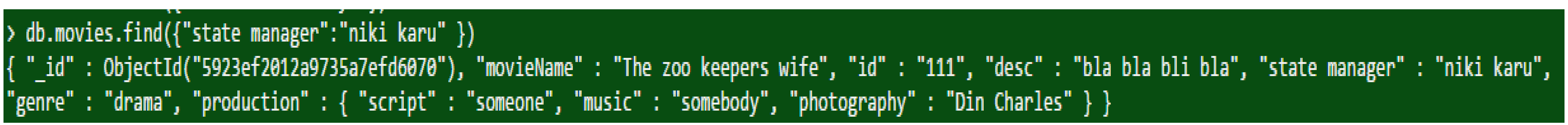


```
{
        "_id" : ObjectId("5923faa512a9735a7efd6072"),
        "movieName" : "The zoo keepers wife",
        "id" : "111",
        "desc" : "bla bla bli bla",
        "state manager" : "niki karu",
        "genre" : "drama",
        "production" : {
                "script" : "someone",
                "music" : "somebody",
                "photography" : "Din Charles"
        }
}
```

Query the Document

• שליפה עם תנאי:

db.COLLECTION_NAME.find({key: value})



```
> db.movies.find({"state manager":"niki karu" })
{ "_id" : ObjectId("5923ef2012a9735a7efd6070"), "movieName" : "The zoo keepers wife", "id" : "111", "desc" : "bla bla bli bla", "state manager" : "niki karu",
"genre" : "drama", "production" : { "script" : "someone", "music" : "somebody", "photography" : "Din Charles" } }
```

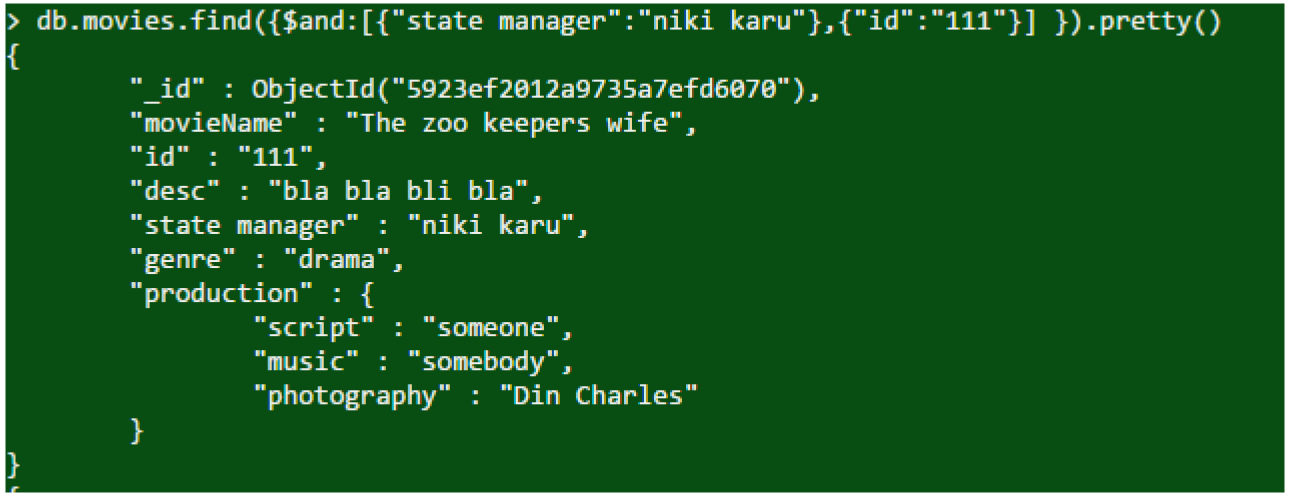
Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:<value>}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{$lt:<value>}}	db.mycol.find({"likes":{$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{$lte:<value>}}	db.mycol.find({"likes":{$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{$gt:<value>}}	db.mycol.find({"likes":{$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{$gte:<value>}}	db.mycol.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{$ne:<value>}}	db.mycol.find({"likes":{$ne:50}}).pretty()	where likes != 50

Query the Document

- שליפה עם AND:

db.mycol.find({ $and: [{key1: value1}, {key2:value2}] }).pretty()

db.mycol.find({ key1: value1, key2:value2}).pretty()



```
> db.movies.find({$and:[{"state manager":"niki karu"},{"id":"111"}] }).pretty()
{
        "_id" : ObjectId("5923ef2012a9735a7efd6070"),
        "movieName" : "The zoo keepers wife",
        "id" : "111",
        "desc" : "bla bla bli bla",
        "state manager" : "niki karu",
        "genre" : "drama",
        "production" : {
                "script" : "someone",
                "music" : "somebody",
                "photography" : "Din Charles"
        }
}
```

Query the Document

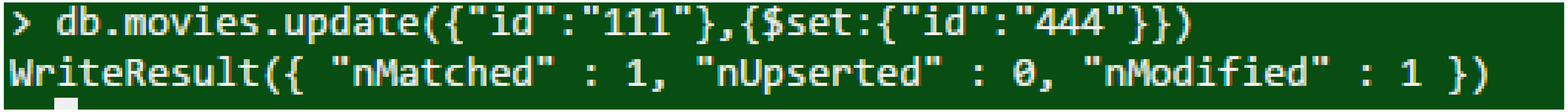
- שליפה עם OR:

```
db.mycol.find( { $or: [ {key1: value1}, {key2:value2} ] } ).pretty()
```

Update

- עדכון נתונים ב- document:

>db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA,[optional: {multi:true}])



```
> db.movies.update({"id":"111"},{$set:{"id":"444"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

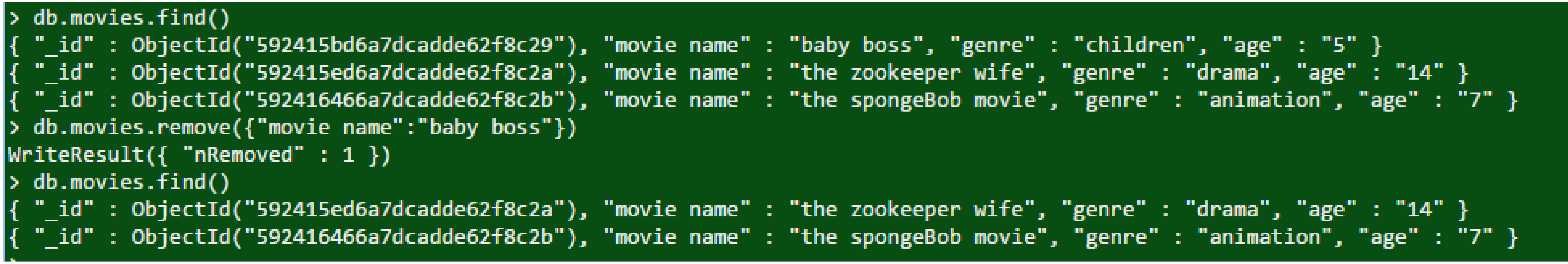
Remove

- הסרת : document

>db.COLLECTION_NAME.remove(DELETION_CRITTERIA)

>db.COLLECTION_NAME.remove(DELETION_CRITERIA,1)

>db.mycol.remove()

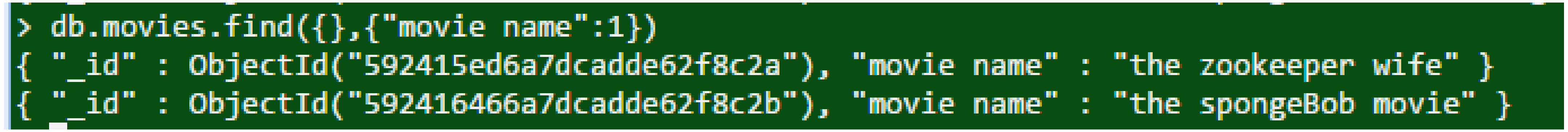


```
> db.movies.find()
{ "_id" : ObjectId("592415bd6a7dcadde62f8c29"), "movie name" : "baby boss", "genre" : "children", "age" : "5" }
{ "_id" : ObjectId("592415ed6a7dcadde62f8c2a"), "movie name" : "the zookeeper wife", "genre" : "drama", "age" : "14" }
{ "_id" : ObjectId("592416466a7dcadde62f8c2b"), "movie name" : "the spongeBob movie", "genre" : "animation", "age" : "7" }
> db.movies.remove({"movie name":"baby boss"})
WriteResult({ "nRemoved" : 1 })
> db.movies.find()
{ "_id" : ObjectId("592415ed6a7dcadde62f8c2a"), "movie name" : "the zookeeper wife", "genre" : "drama", "age" : "14" }
{ "_id" : ObjectId("592416466a7dcadde62f8c2b"), "movie name" : "the spongeBob movie", "genre" : "animation", "age" : "7" }
>
```

Projection

- שליפה של חלק מהנתונים בתוך ה- document:

>db.COLLECTION_NAME.find({},{KEY:1})

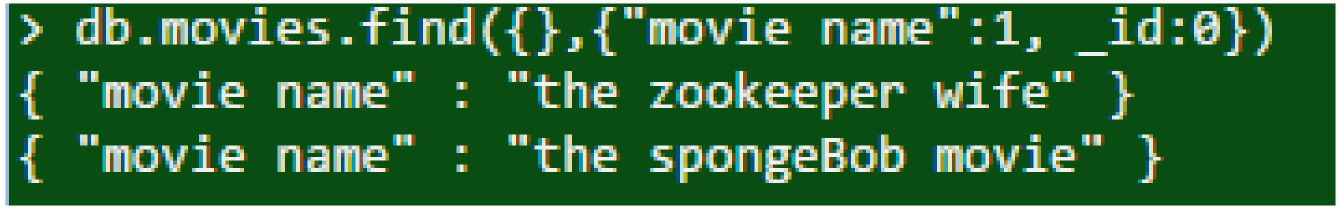


```
> db.movies.find({},{"movie name":1})
{ "_id" : ObjectId("592415ed6a7dcadde62f8c2a"), "movie name" : "the zookeeper wife" }
{ "_id" : ObjectId("592416466a7dcadde62f8c2b"), "movie name" : "the spongeBob movie" }
```

Projection

- שליפה ללא שדה _id:

```
>db.COLLECTION_NAME.find({},{KEY_NAME:1,_id:0})
```

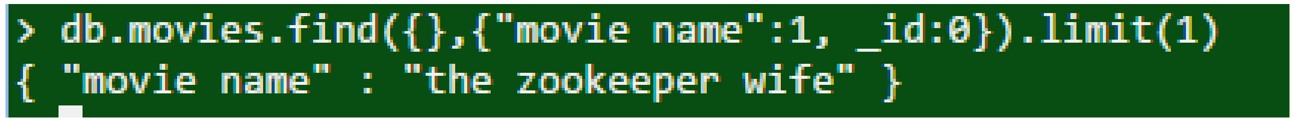


```
> db.movies.find({},{"movie name":1, _id:0})
{ "movie name" : "the zookeeper wife" }
{ "movie name" : "the spongeBob movie" }
```

More Functions

- Limit() – הגבלה של מס' המסמכים לשליפה:

>db.COLLECTION_NAME.find().limit(NUMBER)

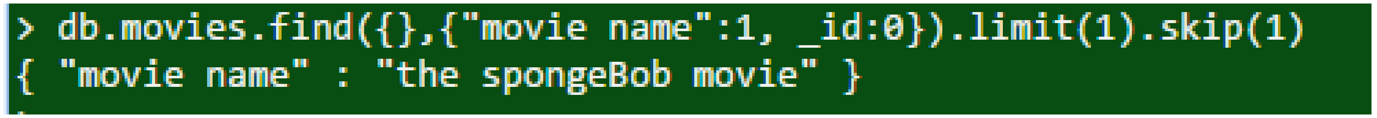


```
> db.movies.find({},{"movie name":1, _id:0}).limit(1)
{ "movie name" : "the zookeeper wife" }
```

More Functions

- Skip() – דילוג על תוצאות

>db.COLLECTION_NAME.find().limit(NUMBER).skip(NUMBER)

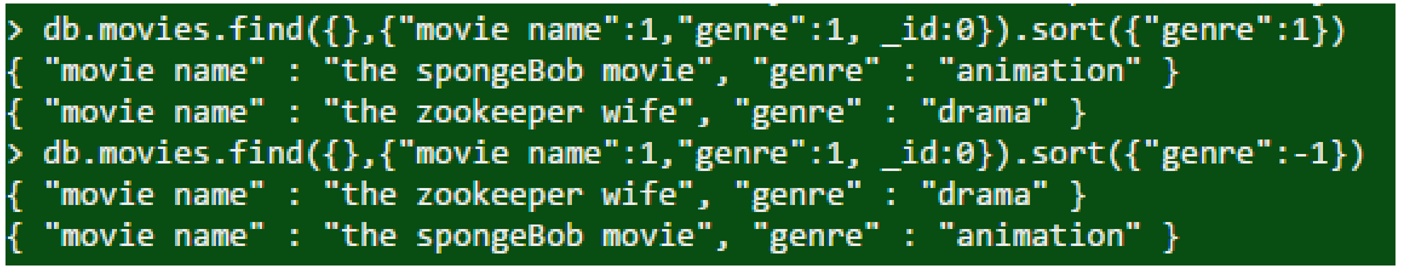


```
> db.movies.find({},{"movie name":1, _id:0}).limit(1).skip(1)
{ "movie name" : "the spongeBob movie" }
```

More Functions

- Sort() – שליפה לפי סדר:

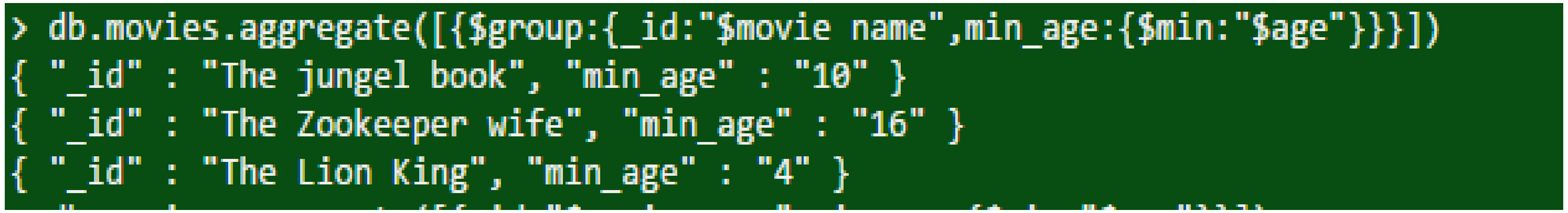
>db.COLLECTION_NAME.find().sort({KEY_NAME:1})



```
> db.movies.find({},{"movie name":1,"genre":1, _id:0}).sort({"genre":1})
{ "movie name" : "the spongeBob movie", "genre" : "animation" }
{ "movie name" : "the zookeeper wife", "genre" : "drama" }
> db.movies.find({},{"movie name":1,"genre":1, _id:0}).sort({"genre":-1})
{ "movie name" : "the zookeeper wife", "genre" : "drama" }
{ "movie name" : "the spongeBob movie", "genre" : "animation" }
```

Aggregation

- >db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)

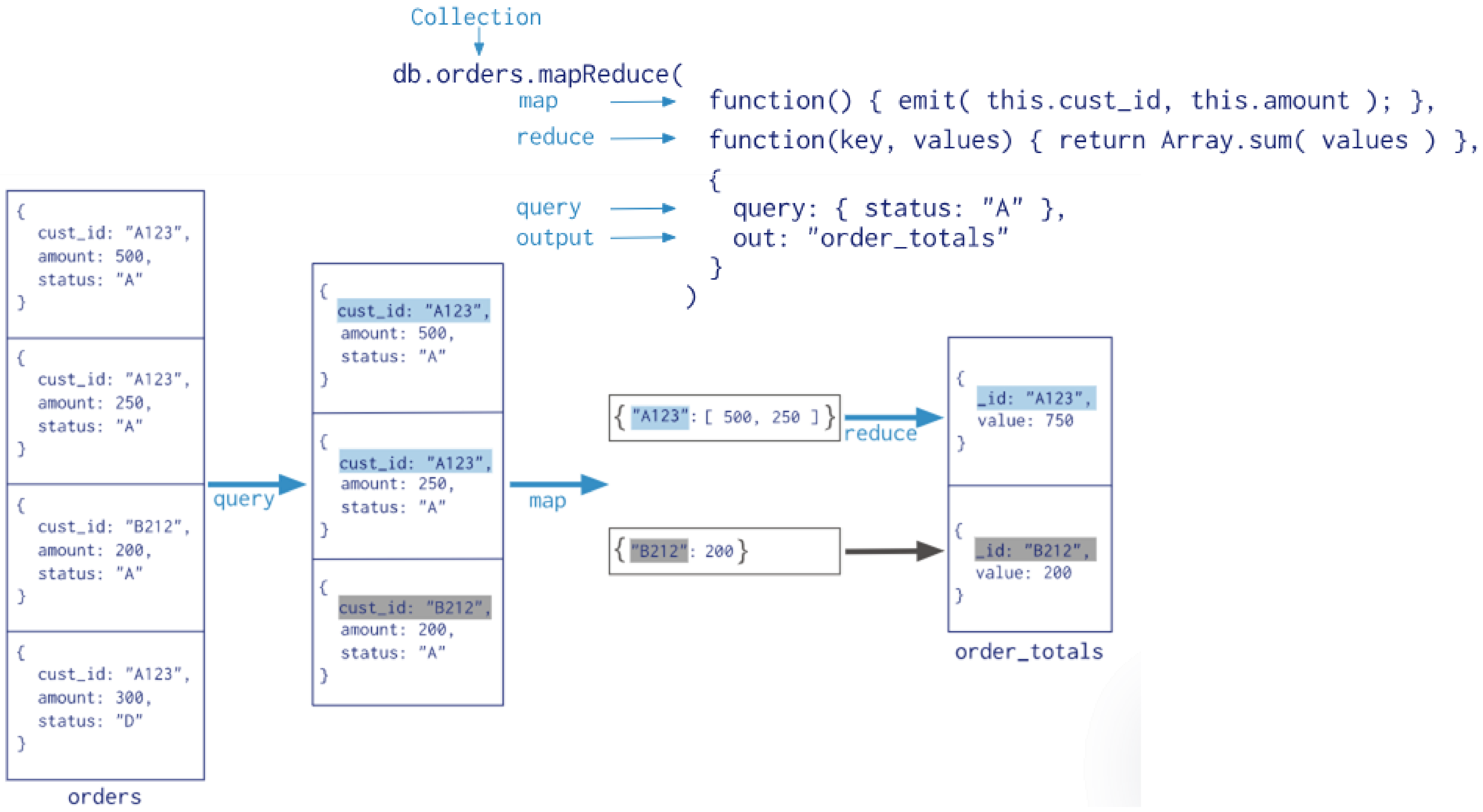


```
> db.movies.aggregate([{$group:{_id:"$movie name",min_age:{$min:"$age"}}}])
{ "_id" : "The jungel book", "min_age" : "10" }
{ "_id" : "The Zookeeper wife", "min_age" : "16" }
{ "_id" : "The Lion King", "min_age" : "4" }
```

Expression	Description	Example
$sum	Sums up the defined value from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
$avg	Calculates the average of all given values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
$min	Gets the minimum of the corresponding values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
$max	Gets the maximum of the corresponding values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
$push	Inserts the value to an array in the resulting document.	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
$first	Gets the first document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage.	db.mycol.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
$last	Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage.	db.mycol.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])

mapReduce()

```
db.collection_name.mapReduce(
    {
        mapper,
        reducer,
        {
            out : “out_name”,
            finalize: finalizeFunction
        }
    }
);
db.collection_name.find(out_name);
```



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

Collection
map
reduce
query
output
{ 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" }
orders
query
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
map
{ "A123": [500, 250] }
{ "B212": 200 }
reduce
{ _id: "A123", value: 750 }
{ _id: "B212", value: 200 }
order_totals

דוגמא

```
mapper = function () {
            emit(this.gender,  1);
};
reducer = function(gender, count){
            return Array.sum(count);
};
db.sourceData.mapReduce(
            mapper,
            reducer,
            {
                        out : "example1_results"
            }
 );
 db.example1_results.find()
```

דרך אחרת לעשות את אותו הדבר:

```
db.students.aggregate([{$group:{_id:"$gender", count:{$sum:1 }}}])
```

דוגמא נוספת:

```
var mapper = function () {
                var x = {age : this.age, name : this.name};
                emit(this.gender, {min : x , max : x});
};
 var reducer = function(key, values){
                var res = values[0];
                for (var i = 1; i < values.length; i++) {
                                if(values[i].min.age < res.min.age)
                                                res.min = {name : values[i].min.name, age : values[i].min.age};
       if (values[i].max.age > res.max.age)
         res.max = {name : values[i].max.name, age : values[i].max.age};
                };
                return res;
};
db.students.mapReduce(
                mapper,
                reducer,
                {out : "example2_results"}
 );
```

```
var mapper = function(){
        for(var i = 0; i< this.grades.length; i++)
        {
                var key = this.grades[i].subject;
                var value = {count : 1, qty: this.grades[i].grade};
                emit(key, value);     }
}
var reducer = function(key, countObj){
                reducValue = {count:0, qty:0};
                for(var i=0; i< countObj.length; i++)
                {
                        reducValue.count ++;
                        reducValue.qty += countObj[i].value;
                }
                return reducValue;};
}
var finalizeAddAvgField =  function (key, reducedVal) {
        reducedVal.avg = reducedVal.qty/reducedVal.count;
        return reducedVal; };

db.students.mapReduce(mapper, reducer, {out:"exp3", finalize: finalizeAddAvgField});
```

תרגילים לעבודה עצמית

נתון בסיס נתונים ב-mongo DB המכיל רשימה של הזמנות מוצרים ופרטי ההזמנה.

דוגמא למבנה של רשומת הזמנה:

```
{
          id:ObjectId("50a8240b927d5d8b5891743c“),
           cust_id: "abc123“,
                     ord_date: new Date("Oct 2012 ,04“),
           status: ‘A’,
           price: 25,
           items :[{ sku: "mmm", qty :5, price:2.5},
               {sku: "nnn", qty:5, price:2.5} ]
    }
```

כתבו פונקציית mapReduce שתחשב עבור כל לקוח את העלות הכוללת של ההזמנות שלו.

```
var mapFunction1 = function() {
                emit(this.cust_id, this.price);
            };
var reduceFunction1 = function(keyCustId, valuesPrices) {
                 return Array.sum(valuesPrices);
             };
db.orders.mapReduce(
              mapFunction1,
              reduceFunction1,
              { out: "map_reduce_example" }
            )
```

כתבו פונקציית mapReduce שתחשב עבור כל מוצר את הכמות הממוצעת שהוזמנה ממנו בהזמנות שבוצעו אחרי ה01/01/2012.

```
var mapFunction2 = function() {
            for (var idx = 0; idx < this.items.length; idx++) {
              var key = this.items[idx].sku;
              var value = {
                     count: 1,
                     qty: this.items[idx].qty
                    };
              emit(key, value);
            }
          };
```

```
var reduceFunction2 = function(keySKU, countObjVals) {
                reducedVal = { count: 0, qty: 0 };

                for (var idx = 0; idx < countObjVals.length; idx++) {
                   reducedVal.count += countObjVals[idx].count;
                   reducedVal.qty += countObjVals[idx].qty;
                }

                return reducedVal;
              };
```

```
var finalizeFunction2 = function (key, reducedVal) {

                    reducedVal.avg = reducedVal.qty/reducedVal.count;

                    return reducedVal;

                  };
```

```
db.orders.mapReduce( mapFunction2,
                     reduceFunction2,
                     {
                       out: { merge: "map_reduce_example" },
                       query: { ord_date:
                                  { $gt: new Date('01/01/2012') }
                              },
                       finalize: finalizeFunction2
                     }
                   )
```

שאלת מבחן – 2017 סמסטר ב' מועד ב'

Consider the car collection which contains the following documents:

```
db.car.insert([
    {car_id:"c1", name:"Audi", color:"Black", current_speed:50},
    {car_id:"c2", name:"Polo", color:"White", current_speed:65},
    {car_id:"c3", name:"Alto", color:"White", current_speed:75},
    {car_id:"c4", name:"Santro", color:"Black", current_speed:150},
    {car_id:"c5", name:"Subaru", color:"Black", current_speed:100},
    {car_id:"c6", name:"Zen", color:"Blue", current_speed:97} ] )
```

What will be the content of my_out after the following mapreduce call (i.e. what will we get when we type:

```
db.my_out.find()):
db.car.mapReduce( function () {
           if ( this.current_speed < 120 )
                      { emit(this.color,   this.current_speed); } },
function(key, val1) {
          var total =0;
          for (var i = 0; i < val1.length; i++) {
                     total += val1[i]; }
          return total / val1.length; },
 {out: "my_out"});
```

my_out:{“Black”: 75, “White”:70, “Blue” : 97 }

שאלת מבחן – 2017 סמסטר קיץ מועד א'

בהינתן רשימת פרטי תלמידים הבאה:

```
db.student.insert([
        {student_id:"111", name:"Dani", course_name:"Databases", grade:42},
        {student_id:"222", name:"Dafna", course_name:"Operating systems", grade:81},
        {student_id:"333", name:"Miri", course_name:"Software structure", grade:78},
        {student_id:"444", name:"Nati", course_name:"Databases", grade:52},
        {student_id:"555", name:"Yaffa", course_name:"Software structure", grade:62},
        {student_id:"666", name:"Zohar", course_name:"Operating systems", grade:95},
        {student_id:"777", name:"Ari", course_name:"Operating systems", grade:48} ,
        {student_id:"888",name:"Miki",course_name:"Databases",grade:65}) ] )
```

ופונקציית ה- mapReduce הבאה:

```
db.student.mapReduce(
        function (){
                if ( this.grade > 59 )
                        { emit(this.course_name, this.grade); } },
         function(key, val1) {
                var total =0;
                for (var i = 0; i < val1.length; i++)
                        { total += val1[i]; } return total / val1.length; },
 {out: "my_out"});
```

```
 “my_out” : { {_id: “Operating systems”, Value:88 } ,
                  {_id:"Software structure", Value:70 },
                  {_id:"Databases", Value:65 } }
```

שאלת מבחן – 2018 סמסטר ב' מועד א'

נתון בסיס נתונים ב-mongo DB המכיל רשימה של שחקני כדורגל המשחקים במונדיאל 2018 ופרטיהם.

דוגמא למבנה של רשומת שחקן:

```
{“_id”: ObjectId("5691048096b75aa53104b939"),
“name”:" Cristiano Ronaldo",
“nationality”:"Portuguese",
“height”:185,
“age”:33,
“playingPosition”:”Forward”}
```

כתוב פונקציית mapReduce שתחשב עבור שחקנים שגילם יותר מ-30 את הגובה הממוצע של שחקן לכל playingPosition.

(playingPosition – הכוונה לתפקיד של השחקן במהלך המשחק, שימו לב שלכל שחקן יש playingPosition בודד במהלך כל הקריירה)