Exercise 5

importing `JSON` data into mongo:

```
cd /var/lib/mongo
mongoimport --db test --collection collectionName /location/of/JSON.json
```

Aggregation functions

Counting all records: with "param" = "something"

```
db.collectionName.count({"param": "something"})
```

Give a list of all unique elements in the database

```
db.collectionName.distinct("param")
```

With `.aggregate()`, one can use a multitude of clauses, which get executed in the order they are given.

Example:

```
db.collectionName.aggregate({$match: {"param": "something"}},
                            {$group: {_id: "$paramX", "totalSum": {$sum: "$paramY"}}})
```

`$project:` extracts the selected fields, _id is always included unless `_id:0` is given, can rename given fiels by passing `newName: "$oldName"`

`$group:` groups by a field selected by `_id: "$field"` or `_id: {"$field1", "$field2"}` for multiple fields.
There can be calculations on parameters too by using one of the following accumulators:

- **`$addToSet:`** collects unique values of the given field.
- **`$avg:`** gives the average of all numerical values in this field
- **`$min and $max:`** give the min/max value of this field
- **`$first and $last:`** give the first/last value of this field, only useful after a sort
- **`$push:`** gives all values for this field
- **`$sum`** gives the sum of all numerical values in this field

`$sort:` sorts the output by the given fields, `-1` for `A-Z`, `1` for `Z-A`

`$skip:` takes only one number and drops the given amount of results

`$unwind:` opens up the given array, outputs a line for each value of the array with the value of the array field replaced by the element

`$limit:` Limits the number of results to the given amount

`$match:` Only passes on the entries that conform to the given query

Other operators

`$multiply, $divide, $add, $subtract` all work similar and take an array: `$multiply: ["$pop", 2]` would double the value of `$pop`

`$year, $month, $week, $dayOfMonth, $dayOfWeek, $dayOfYear, $hour, $minute, $second` can extract data from a date, like so:
`{ $year: "$date" }`

`$substr` takes an array with `[expr, startOffset, numOfChars]`

`$concat` concatenates all strings in a given array

`$toLower and $toUpper` both take a string

Exercise 6

Introduction

The mongoDB Mapreduce works as follows: `db.collectionName.mapReduce(map, reduce, options)`:

The first parameter is the mapper function, which gets applied to each result.

The second parameter is the reducer function, that gets applied to the key-valuearray that comes from the mapper.

The options must contain an `out`-value, which is a collection where the output will be stored, use `{inline:1}` to simply receive them in the console. A `query`-value can also be given, which gets applied before the mapper. The functions are in JavaScript.

An example mapper function is:

```
function() {
    emit(this.theKey, this.theValue);
}
```

An example reducer function is:

```
function(key, values) {
    return Array.sum(values);
}
```

More examples

To count the 5 most popular items in an array `theArray` over all elements in the collection.

```
db.collectionName.mapReduce(
    function() { // mapper
        if (!this.theArray) {
            return;
        }
        for(index in this.theArray) {
            emit(this.theArray[index], 1);
        }
    }, // end mapper
    function(key, values) { // reducer
        return Array.sum(values);
    }, // end reducer
    {out: {inline:1}}
);
```

Useful mongoDB commands (© JurgenVM)

Command	Meaning
`mongo`	Load mongo shell
`show dbs`	Show current databases
`use movies`	Use the database `movies`
`db.getName()`	Get name current database
`show collections`	Show all tables