NBD MongoDB lab. 1

Tasks for the student are denoted with “Taskx” where x is a number. Solutions to such tasks should be submitted to eNauczanie. “Task*” denotes practice tasks not requiring submission.

Please open command terminal and move to the directory with MongoDB binaries.

Run *.\mongod*. You will get an error stating x:\data\db is missing.

Create data_db directory than run:

mongod --dbpath .\data_db (or choose other empty directory).

Result: a lot of messages containing: "ctx":"listener","msg":"Waiting for connections","attr":{"port":27017,"ssl":"off"}} (default, can be changed).

Move to the directory containing Mongo shell binaries, open new command terminal window and run

mongosh   (will connect with localhost:27017, other address can be given)

```
test> db.getCollectionNames()
[ ]
```

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

```
test > db.getCollectionNames()
[ "AAA" ]
```

```
test> db.test.insertOne({"test":"document"})
{
  acknowledged: true,
  insertedId: ObjectId("637375babde66388c8eda792")
}
```

```
test > db.getCollectionNames()
[ "AAA", "test" ]
```
--When inserting data, collection needs to be specified. If collection does not exist, it is created.

```
test> db.AAA.drop()
true
```

```
test> db.getCollectionNames()
[ "test" ]
```

```
test> db
test
```
--db represents currently selected database, database test selected by default

```
test> show dbs
```

```
admin   40.00 KiB         -- store system collections and user authentication and authorization data
config  60.00 KiB         --support for sharding and causally consistent sessions
local   40.00 KiB         --data for replication and instance-specific data
test    72.00 KiB         --default empty db
                          --4 databases present by default.

test> use local
switched to db local

local> db.getCollectionNames()
[ "startup_log" ]

local> use config
switched to db config

config> db.getCollectionNames()
[ "system.sessions" ]

config > use admin
switched to db admin

admin> db.getCollectionNames()
[ "system.version" ]

admin> db.system.version.find()
[ { _id: 'featureCompatibilityVersion', version: '6.0' } ]

admin> use test
switched to db test       --other databases store important information

test> db.test.insertOne({"test":"document"})
{
  acknowledged: true,
  insertedId: ObjectId("6373782dbde66388c8eda793")
}                   --successfully inserting the same document twice

test> db.test.find()
[
  { _id: ObjectId("637375babde66388c8eda792"), test: 'document' },
  { _id: ObjectId("6373782dbde66388c8eda793"), test: 'document' }
]         -- unique _id will be created by MongoDB if not provided by the user

test> db.test.insertOne({"_id":1, "test": 2})
{ acknowledged: true, insertedId: 1 }

test > db.test.insertOne({"_id":1, "test": 3})
MongoServerError: E11000 duplicate key error collection: test.test index: _id_ dup key: { _id: 1 }
        --when providing own _id value, duplicates will not be accepted (_id_ unique index is created
on _id field of every collection)
```

```
test > db.test.insertOne({
... name: "Anthony",
... surname: "Smith",
... age: 47,
... address: {
... street: "Queen's road",
... number: 15,
... "zip-code": "45-236"
... },
... phones: [ "423-453-4534", "222-222-2222"],
... associates: [ {name: "Bart", surname: "Smith"}, {name: "Joan", surname: "Bean"} ]
... })
{
  acknowledged: true,
  insertedId: ObjectId("6373795bbde66388c8eda794")
}
```

```
test> db.test.find({name:"Anthony"})
[
  {
    _id: ObjectId("6373795bbde66388c8eda794"),
    name: 'Anthony',
    surname: 'Smith',
    age: 47,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '423-453-4534', '222-222-2222' ],
    associates: [
      { name: 'Bart', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  }
]
```

```
test> db.test.find({name:"Anthony"}).pretty()
[
  {
    _id: ObjectId("6373795bbde66388c8eda794"),
    name: 'Anthony',
    surname: 'Smith',
    age: 47,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '423-453-4534', '222-222-2222' ],
    associates: [
      { name: 'Bart', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
```

```
 }
]
```
--formatted vs. also formatted (in old shell find() used to return results in a single line)

--you can find documents to be inserted by this command in “Sample documents for lab 1” file on eNauczanie

```
test> db.test.insertMany([
... {
... name: "Bart",
... surname: "Smith",
... age: 52,
... address: {
... street: "Queen's road",
... number: 18,
... "zip-code": "45-236"
... },
... phones: [ "654-234-5466", "546-243-3455"],
... associates: [ {name: "Anthony", surname: "Smith"}, {name: "Joan", surname: "Bean"} ]
... },
... {
... name: "Joan",
... surname: "Bean",
... age: 43,
... address: {
... street: "Queen's road",
... number: 15,
... "zip-code": "45-236"
... },
... phones: [ "123-456-7890", "222-222-2222", "423-453-4534"],
... associates: [ {surname: "Smith", name: "Anthony" }, {name: "Jonny", surname: "Bean"} ],
... marks: null
... },
... {
... name: "Jonny",
... surname: "Bean",
... age: 23,
... address: {
... street: "Mulholand drive",
... number: 5,
... "zip-code": "45-236"
... },
... phones: [ "423-453-4534", "999-342-2344"],
... associates: [ {name: "All", surname: "o'Brian"}, {name: "Joan", surname: "Bean"} ],
... marks: [1,3,7,8]
... },
... {
```

```
... name: "Burt",
... surname: "O'Donnel",
... age: 39,
... address: {
... street: "Dunno ally",
... number: 13,
... "zip-code": "11-435"
... },
... phones: [ "543-546-5355", "111-111-1111"],
... associates: [ {name: "Kathy", surname: "Jenkins"} ],
... marks: [4,5]
... },
... {
... name: "Abe",
... surname: "Kendricks",
... age: 47,
... address: {
... street: "King's highway",
... number: 1,
... "zip-code": "98-435"
... },
... phones: [ "774-425-4324", "554-542-8997"],
... associates: [ {name: "Joan", surname: "Black"}, {name: "Patty", surname: "Fuller"} ],
... marks: []
... }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63737ac2bde66388c8eda795"),
    '1': ObjectId("63737ac2bde66388c8eda796"),
    '2': ObjectId("63737ac2bde66388c8eda797"),
    '3': ObjectId("63737ac2bde66388c8eda798"),
    '4': ObjectId("63737ac2bde66388c8eda799")
  }
}           --inserted 5 more documents
```

Task* Find people that are 47 years old

.

.

.

```
test> db.test.find({age:47})
[
  {
    _id: ObjectId("6373795bbde66388c8eda794"),
    name: 'Anthony',
    surname: 'Smith',
    age: 47,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '423-453-4534', '222-222-2222' ],
    associates: [
      { name: 'Bart', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  },
  {
    _id: ObjectId("63737ac2bde66388c8eda799"),
    name: 'Abe',
    surname: 'Kendricks',
    age: 47,
    address: { street: "King's highway", number: 1, 'zip-code': '98-435' },
    phones: [ '774-425-4324', '554-542-8997' ],
    associates: [
      { name: 'Joan', surname: 'Black' },
      { name: 'Patty', surname: 'Fuller' }
    ],
    marks: []
  }
]                        --a bit too much information

test> db.test.find({age:47}, {name: 1, surname:1})          --only interesting information can be
retrieved
[
  {
    _id: ObjectId("6373795bbde66388c8eda794"),
    name: 'Anthony',
    surname: 'Smith'
  },
  {
    _id: ObjectId("63737ac2bde66388c8eda799"),
    name: 'Abe',
    surname: 'Kendricks'
  }
]            --_id treated differently: if not explicitly excluded, it will be included

test> db.test.find({age:47}, {"_id":0, name: 1, surname:1})
[
  { name: 'Anthony', surname: 'Smith' },
  { name: 'Abe', surname: 'Kendricks' }
]            --now only name and surname
```

You can specify which fields are to be retrieved using “:1”, or which are to be removed from found documents with “:0”, but not both. _id is an exception to this rule.

Task 1. Find Kendricks’s phone numbers (1 point).

```
--people having 40 years or more
test> db.test.find({age: {$gte: 40}}, {"_id":0, age: 1, name:1, surname:1})
[
  { name: 'Anthony', surname: 'Smith', age: 47 },
  { name: 'Bart', surname: 'Smith', age: 52 },
  { name: 'Joan', surname: 'Bean', age: 43 },
  { name: 'Abe', surname: 'Kendricks', age: 47 }
]
```

```
--Smiths having 40 years or more
test> db.test.find({age: {$gte: 40}, surname: "Smith"}, {"_id":0, age: 1, name:1, surname:1})
[
  { name: 'Anthony', surname: 'Smith', age: 47 },
  { name: 'Bart', surname: 'Smith', age: 52 }
]
```

```
--people between 40 and 50 years old
test> db.test.find({age: {$gte: 40}, age: {$lt: 50}}, {"_id":0, age: 1, name:1, surname:1})
[
  { name: 'Anthony', surname: 'Smith', age: 47 },
  { name: 'Joan', surname: 'Bean', age: 43 },
  { name: 'Jonny', surname: 'Bean', age: 23 },
  { name: 'Burt', surname: "O'Donnel", age: 39 },
  { name: 'Abe', surname: 'Kendricks', age: 47 }
]        --doesn’t work properly, because the same field used twice, only second condition is checked
```

```
test> db.test.find({age: {$gte: 40, $lt: 50}}, {"_id":0, age: 1, name:1, surname:1})
[
  { name: 'Anthony', surname: 'Smith', age: 47 },
  { name: 'Joan', surname: 'Bean', age: 43 },
  { name: 'Abe', surname: 'Kendricks', age: 47 }
]        --works correctly this time
```

```
--other way
test> db.test.find({$and :[{age: {$gte: 40}}, {age: {$lt: 50}}]}, {"_id":0, age: 1, name:1, surname:1})
[
  { name: 'Anthony', surname: 'Smith', age: 47 },
  { name: 'Joan', surname: 'Bean', age: 43 },
  { name: 'Abe', surname: 'Kendricks', age: 47 }
]        --same result
```

Task* Find people younger than 40 or older than 50 (reverse previous query)

.

.

.

```
test> db.test.find({$or :[{age: {$lt: 40}}, {age: {$gte: 50}}]}, {"_id":0, age: 1, name:1, surname:1})
[
  { name: 'Bart', surname: 'Smith', age: 52 },
  { name: 'Jonny', surname: 'Bean', age: 23 },
  { name: 'Burt', surname: "O'Donnel", age: 39 }
]         --operator needed for OR
```

```
--people living on Queen's road 15
test> db.test.find({address: {street: "Queen's road", number: 15, "zip-code": "45-236" }})
[
  {
    _id: ObjectId("6373795bbde66388c8eda794"),
    name: 'Anthony',
    surname: 'Smith',
    age: 47,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '423-453-4534', '222-222-2222' ],
    associates: [
      { name: 'Bart', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  },
  {
    _id: ObjectId("63737ac2bde66388c8eda796"),
    name: 'Joan',
    surname: 'Bean',
    age: 43,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '123-456-7890', '222-222-2222', '423-453-4534' ],
    associates: [
      { surname: 'Smith', name: 'Anthony' },
      { name: 'Jonny', surname: 'Bean' }
    ],
    marks: null
  }
]         --works ok, by why include zip-code?
```

```
test> db.test.find({address: {street: "Queen's road", number: 15}})       --nothing found
```

```
--let's use zip-code, but change order of the fields
test> db.test.find({address: {"zip-code": "45-236", street: "Queen's road", number: 15 }})         --
```

nothing again, such query finds documents that have field address containing exactly the specified (sub)document

```
--all people with a certain zip-code?
test> db.test.find({"address.zip-code": "45-236"})
[
  {
    _id: ObjectId("6373795bbde66388c8eda794"),
    name: 'Anthony',
    surname: 'Smith',
    age: 47,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '423-453-4534', '222-222-2222' ],
    associates: [
      { name: 'Bart', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  },
  {
    _id: ObjectId("63737ac2bde66388c8eda795"),
    name: 'Bart',
    surname: 'Smith',
    age: 52,
    address: { street: "Queen's road", number: 18, 'zip-code': '45-236' },
    phones: [ '654-234-5466', '546-243-3455' ],
    associates: [
      { name: 'Anthony', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  },
  {
    _id: ObjectId("63737ac2bde66388c8eda796"),
    name: 'Joan',
    surname: 'Bean',
    age: 43,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '123-456-7890', '222-222-2222', '423-453-4534' ],
    associates: [
      { surname: 'Smith', name: 'Anthony' },
      { name: 'Jonny', surname: 'Bean' }
    ],
    marks: null
  },
  {
    _id: ObjectId("63737ac2bde66388c8eda797"),
    name: 'Jonny',
    surname: 'Bean',
    age: 23,
    address: { street: 'Mulholand drive', number: 5, 'zip-code': '45-236' },
    phones: [ '423-453-4534', '999-342-2344' ],
    associates: [
      { name: 'All', surname: "o'Brian" },
      { name: 'Joan', surname: 'Bean' }
    ],
    marks: [ 1, 3, 7, 8 ]
  }
]
```

Task 2. Find people with building number lower than 14 (1 point).

```
--searching lists, let's start by finding Anthony by his phone numbers
test> db.test.find({phones: [ "423-453-4534", "222-222-2222"]})
[
  {
    _id: ObjectId("6373795bbde66388c8eda794"),
    name: 'Anthony',
    surname: 'Smith',
    age: 47,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '423-453-4534', '222-222-2222' ],
    associates: [
      { name: 'Bart', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  }
]                          --Anthony found, but Joan has the same phones

test> db.test.find({phones: {$all:[ "423-453-4534", "222-222-2222"]}})
[
  {
    _id: ObjectId("6373795bbde66388c8eda794"),
    name: 'Anthony',
    surname: 'Smith',
    age: 47,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '423-453-4534', '222-222-2222' ],
    associates: [
      { name: 'Bart', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  },
  {
    _id: ObjectId("63737ac2bde66388c8eda796"),
    name: 'Joan',
    surname: 'Bean',
    age: 43,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '123-456-7890', '222-222-2222', '423-453-4534' ],
    associates: [
      { surname: 'Smith', name: 'Anthony' },
      { name: 'Jonny', surname: 'Bean' }
    ],
    marks: null
  }
]           --now both found
```

Task* What was the previous query looking for exactly?
.
.
.
Field phones with the exact value of the specified list.
Last query looked for field phones being a list and containing both specified values.

```
--everyone having phone "222-222-22222" (among others)
test> db.test.find({phones: "222-222-2222"})
[
  {
    _id: ObjectId("6373795bbde66388c8eda794"),
    name: 'Anthony',
    surname: 'Smith',
    age: 47,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '423-453-4534', '222-222-2222' ],
    associates: [
      { name: 'Bart', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  },
  {
    _id: ObjectId("63737ac2bde66388c8eda796"),
    name: 'Joan',
    surname: 'Bean',
    age: 43,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '123-456-7890', '222-222-2222', '423-453-4534' ],
    associates: [
      { surname: 'Smith', name: 'Anthony' },
    { name: 'Jonny', surname: 'Bean' }
   ],
   marks: null
  }
]
```

--finding field with given value or finding field with a list containing the value is the exact same query

Task 3. Find people that know Jonny (1 point).

--people with mark greater than 5

```
test> db.test.find({marks: {$gt: 5}})
[
  {
   _id: ObjectId("63737ac2bde66388c8eda797"),
   name: 'Jonny',
   surname: 'Bean',
   age: 23,
   address: { street: 'Mulholand drive', number: 5, 'zip-code': '45-236' },
   phones: [ '423-453-4534', '999-342-2344' ],
   associates: [
    { name: 'All', surname: "o'Brian" },
    { name: 'Joan', surname: 'Bean' }
   ],
   marks: [ 1, 3, 7, 8 ]
  }
]
```

--marks between 4 and 6 inclusively

```
test> db.test.find({$and: [{marks: {$gte: 4, $lte:6}}]})
[
  {
   _id: ObjectId("63737ac2bde66388c8eda797"),
   name: 'Jonny',
   surname: 'Bean',
   age: 23,
   address: { street: 'Mulholand drive', number: 5, 'zip-code': '45-236' },
   phones: [ '423-453-4534', '999-342-2344' ],
   associates: [
    { name: 'All', surname: "o'Brian" },
    { name: 'Joan', surname: 'Bean' }
   ],
   marks: [ 1, 3, 7, 8 ]
  },

  {
   _id: ObjectId("63737ac2bde66388c8eda798"),
   name: 'Burt',
   surname: "O'Donnel",
   age: 39,
   address: { street: 'Dunno ally', number: 13, 'zip-code': '11-435' },
   phones: [ '543-546-5355', '111-111-1111' ],
   associates: [ { name: 'Kathy', surname: 'Jenkins' } ],
   marks: [ 4, 5 ]
  }
]           --error
```

Task* What was found?

```
.
.
.
test> db.test.find({$and: [{marks: {$gte: 4, $lte:6}}]},{"marks.$":1})
[
  { _id: ObjectId("63737ac2bde66388c8eda797"), marks: [ 1 ] },
  { _id: ObjectId("63737ac2bde66388c8eda798"), marks: [ 4 ] }
]           --marks.$ - returning first element of a list marks matched by the query; a list with an
element greater or equal to 4 and an element lesser or equal to 6 was found
```

Task 4. Find which Smiths (surname) are known by others (1 point).

```
test> db.test.find({marks: {$elemMatch: {$gte: 4, $lte:6}}})
[
  {
    _id: ObjectId("63737ac2bde66388c8eda798"),
    name: 'Burt',
    surname: "O'Donnel",
    age: 39,
    address: { street: 'Dunno ally', number: 13, 'zip-code': '11-435' },
    phones: [ '543-546-5355', '111-111-1111' ],
    associates: [ { name: 'Kathy', surname: 'Jenkins' } ],
    marks: [ 4, 5 ]
  }
]           --now it's a single element fulfilling both conditions
```

```
--people without any marks
test> db.test.find({$or:[{marks: null},{marks:{$size:0}}]},{marks:1})
[
  { _id: ObjectId("637375babde66388c8eda792") },
  { _id: ObjectId("6373782dbde66388c8eda793") },
  { _id: 1 },
  { _id: ObjectId("6373795bbde66388c8eda794") },
  { _id: ObjectId("63737ac2bde66388c8eda795") },
  { _id: ObjectId("63737ac2bde66388c8eda796"), marks: null },
  { _id: ObjectId("63737ac2bde66388c8eda799"), marks: [] }
]

-- people with some marks
> db.test.find({marks: {$not: {$size: 0}}},{marks:1})
[
  { _id: ObjectId("637375babde66388c8eda792") },
  { _id: ObjectId("6373782dbde66388c8eda793") },
  { _id: 1 },
  { _id: ObjectId("6373795bbde66388c8eda794") },
  { _id: ObjectId("63737ac2bde66388c8eda795") },
  { _id: ObjectId("63737ac2bde66388c8eda796"), marks: null },
  { _id: ObjectId("63737ac2bde66388c8eda797"), marks: [ 1, 3, 7, 8 ] },
  { _id: ObjectId("63737ac2bde66388c8eda798"), marks: [ 4, 5 ] }
]           --no marks or null marks are both not marks with empty list

test> db.test.find({marks: {$not: {$type: 10}}},{marks:1})
[
  { _id: ObjectId("637375babde66388c8eda792") },
  { _id: ObjectId("6373782dbde66388c8eda793") },
  { _id: 1 },
  { _id: ObjectId("6373795bbde66388c8eda794") },
  { _id: ObjectId("63737ac2bde66388c8eda795") },
  { _id: ObjectId("63737ac2bde66388c8eda797"), marks: [ 1, 3, 7, 8 ] },
  { _id: ObjectId("63737ac2bde66388c8eda798"), marks: [ 4, 5 ] },
  { _id: ObjectId("63737ac2bde66388c8eda799"), marks: [] }
]           --documents without null as marks value

test> db.test.find({marks: {$exists: true}},{marks:1})
[
  { _id: ObjectId("63737ac2bde66388c8eda796"), marks: null },
  { _id: ObjectId("63737ac2bde66388c8eda797"), marks: [ 1, 3, 7, 8 ] },
  { _id: ObjectId("63737ac2bde66388c8eda798"), marks: [ 4, 5 ] },
  { _id: ObjectId("63737ac2bde66388c8eda799"), marks: [] }
]           --documents with a marks field

--combined and correct
test> db.test.find({$and: [{marks: {$not: {$size: 0}}},{marks: {$not: {$type: 10}}},{marks: {$exists: 
true}}]},{marks:1})
[
  { _id: ObjectId("63737ac2bde66388c8eda797"), marks: [ 1, 3, 7, 8 ] },
  { _id: ObjectId("63737ac2bde66388c8eda798"), marks: [ 4, 5 ] }
]

--what will also work would be
test> db.test.find({$nor:[{marks: null},{marks:{$size:0}}]},{marks:1})
```

Task 5. Find people not living on Queen's road (1 point).

```
--searching lists of documents, people that know Anthony Smith
test> db.test.find({associates: {name:"Anthony", surname: "Smith"}},{associates:1})
[
  {
    _id: ObjectId("63737ac2bde66388c8eda795"),
    associates: [
      { name: 'Anthony', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  }
]            --is it correct? let's find people knowing any Smith

test> db.test.find({"associates.surname": "Smith"},{associates:1})
[
  {
    _id: ObjectId("6373795bbde66388c8eda794"),
    associates: [
      { name: 'Bart', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  },
  {
    _id: ObjectId("63737ac2bde66388c8eda795"),
    associates: [
      { name: 'Anthony', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  },
  {
    _id: ObjectId("63737ac2bde66388c8eda796"),
    associates: [
      { surname: 'Smith', name: 'Anthony' },
      { name: 'Jonny', surname: 'Bean' }
    ]
  }
]            --we have Anthony Smith but also Smith Anthony

test> db.test.find({associates: {$elemMatch:{name:"Anthony", surname: "Smith"}}},{associates:1})
[
  {
    _id: ObjectId("63737ac2bde66388c8eda795"),
    associates: [
      { name: 'Anthony', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  },
  {
    _id: ObjectId("63737ac2bde66388c8eda796"),
    associates: [
      { surname: 'Smith', name: 'Anthony' },
      { name: 'Jonny', surname: 'Bean' }
    ]
  }
]           --now getting correct result
```

--let’s change Anthony’s name to Antony

```
test> db.test.updateOne({name: "Anthony"}, {name: "Antony"})
MongoInvalidArgumentError: Update document requires atomic operators       --strange error, let’s try differently
```

```
test> db.test.replaceOne({name: "Anthony"}, {name: "Antony"})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}           --worked, what the result is?
```

```
test> db.test.find({name: "Antony"})
[ { _id: ObjectId("6373795bbde66388c8eda794"), name: 'Antony' } ]      --not really the desired result
```

```
test> db.test.deleteOne({name: "Antony"})
{ acknowledged: true, deletedCount: 1 }
```

```
test> db.test.insertOne({
... name: "Anthony",
... surname: "Smith",
... age: 47,
... address: {
... street: "Queen's road",
... number: 15,
... "zip-code": "45-236"
... },
... phones: [ "423-453-4534", "222-222-2222"],
... associates: [ {name: "Bart", surname: "Smith"}, {name: "Joan", surname: "Bean"} ]
... })
{
  acknowledged: true,
  insertedId: ObjectId("63738505bde66388c8eda79a")
}                       --reinserting Anthony, now let’s do it properly
```

```
test> db.test.updateOne({name: "Anthony"},{$set: {name: "Antony"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}           --what’s the result?
```

```
test> db.test.find({name: "Antony"})
[
  {
    _id: ObjectId("63738505bde66388c8eda79a"),
    name: 'Antony',
    surname: 'Smith',
    age: 47,
    address: { street: "Queen's road", number: 15, 'zip-code': '45-236' },
    phones: [ '423-453-4534', '222-222-2222' ],
    associates: [
      { name: 'Bart', surname: 'Smith' },
      { name: 'Joan', surname: 'Bean' }
    ]
  }
]                       --result is correct
```

updateOne, updateMany, replaceOne can have parameter upsert: true set – than if no document is matched, one is inserted

--let’s correct references

```
test> db.test.updateMany({"associates.name": "Anthony"},{$set: {"associates.name": "Antony"}})
MongoServerError: Cannot create field 'name' in element {associates: [ { name: "Anthony", surname: "Smith" }, { name: "Joan", surname: "Bean" } ]}
```

error because associates is a list and $set does not know which element of the list to modify

```
test> db.test.updateMany({associates: {$elemMatch:{name:"Anthony", surname: "Smith"}}},{$set: {"associates.$.name": "Antony"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}           --first element of the list that matched the search criterion are modified
```

```
test> db.test.updateMany({associates: {$elemMatch: {name:"Anthony", surname: "Smith"}}},{$set: {"associates.$[test].name": "Antony"}}, {arrayFilters: [ { "test.name":"Anthony", "test.surname": "Smith"}]})                     --this query would modify all matching element of the list
```

```
--increasing everyone’s age by 1
test> db.test.updateMany({},{$inc: {age: 1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 9,
  modifiedCount: 9,
  upsertedCount: 0
}

--increasing everyone’s age to at least 30
test> db.test.updateMany({},{$max:{age: 30}})  --$min is also available
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 9,
  modifiedCount: 4,
  upsertedCount: 0
}  --4 of the 9 matched documents had age under 30

--adding everyone with marks list (type 4/”array”) two marks 5 and 7
test> db.test.updateMany({marks: {$type: "array"}},{$push: {marks:{$each: [5,7]}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}           --$push adds an element to a list (without $each modifier an element [5,7](a sublist) would
be added to marks)

test> db.test.updateMany({},{$unset: {marks: null}})      --removing marks from everyone
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 9,
  modifiedCount: 4,
  upsertedCount: 0
}

test> db.test.deleteMany({})      --removing all documents
{ acknowledged: true, deletedCount: 9 }

test> db.test.drop()                  --removing test collection
true
```