NoSQL - Mongo DB

Connecting to database :

```
use name_databse
```

Creating A collection :

```
db.name_collection.insertOne (
    {
        name_of_field : value
        ...
    }
)
```

Example :

```
db.author.insertOne(
    {
        name: "ashraf",
        lastName: "khabar",
        phoneNumber: 0699530916
    }
)
```

Authomatically the id with be created, and also the collection if not created will be created too.

Inserting in collection :

we can insert into the collection like we did above , or we can insert many documents at the same time using `insertMany()` function :

```
db.author.insertOne([
    {
        name: "ashraf",
        lastName: "khabar",
        phoneNumber: 0699530916
    },
    {
        name: "sami",
        lastName: "Aouad",
        phoneNumber: 02588266952
        }
    ])
```

Getting all docs :

```
db.author.find()
```

Or :

```
db.author.find({})
```

Using an empty object

Getting the count and the limit of docs :

```
db.Author.find().count()
```

```
db.Author.find().limit(3)
```

Filtering the Docs :

Getting all the doc with the **name** = *Ashraf*

```
db.Author.find({name: "Ashraf"})
```

If i want to get all the docs but only the **name** :

```
db.Author.find({}, {name: 1})
```

Sorting the Docs :

How about sorting a doc by the **phone number** :

```
db.Author.find().sort({phoneNumber : 1})
```

Sorting the docs at the same time With limit :

```
db.Author.find().sort({name : 1}).limit(3)
```

Nested docs :

Sometimes we need to insert a data non atomic value (without respecting the 1NF in SGBDOR) :

```
db.books.insertOne(
    {
        title: "La boite a merveuille",
        author: "Ahmed safrioui",
        rating: 9,
        genres: ["6eme annee", "Jihaoui"],
        reviews: [
            {
                name: "ashraf khabar",
                body : "The worst book i had ever read"
            },
            {
                name : "Sami Aouad",
                body : "As ashraf khabar"
            }
        ]
    }
)
```

And the **id** gonna be created automatically .

Operators :

If we want to select the data based on a creteria of value of rating (greater than, less than, greater or equal than ...) :

```
db.books.find (
    {
        rating : { $gt : 7 },
        author : "Ahmed safrioui"
    }
)
```

PS : We have **lt**, **gt**, **le**, **ge**

How about selection using **Or** :

```
db.books.find(
    {
        $or : [
            {rating : 7},
            [rating : 9]
        ]
    }
)
```

How about combining both of operators :

```
    db.books.find(
        {
            $or : [
                {
                    pages : {$lt : 300}
                },
                {
                    pages: {$gt : 400}
                }
            ]
        }
    )
```

$in and $nin :

Sometimes it so over to user **or** when we have a lot of values , so we can use **$in** operator :

```
    db.books.find(
        {
            rating : {

                $in : [7, 9, 8]
            }
        }
    )
```

Or we can use the not in : $nin

```
    db.books.find(
        {
            rating : {

                $nin : [7, 9, 8]
            }
        }
    )
```

Querying array :

In this case , i want to fetch the docs that the array genres has a value in it with the name **jihaoui** :

```
db.books.find(
    {
        genres: "Jihaoui"
    }
)
```

But if i want to fetch the array with the exact value , i need to make it inside the array :

```
db.books.find(
    {
        genres: ["Jihaoui"]
    }
)
```

And how about if i want to have two values if they are inside the holle array , like if [c, d, b] is inside [a, b, c, d, e] , in this case we gonna use the operator **$all** :

```
db.books.find(
    {
        genres : {
            $all : [
                "6eme annee",
                "Jihaoui"
            ]
        }
    }
)
```

And how we can query in the nested docs :

```
db.books.find(
    {
        "reviews.name" : "Ashraf khabar"
    }
)
```

PS : when we have a nested docs and we make the dot notation, we add the brakets => $reviews.name$ is false , but $"reviews.name"$ is true .

Deleting a doc :

```
db.books.deleteOne(
    {
        _id : ObjectId('63fa75616af9c77a2e1b98ee')
    }
)
```

PS : if we delete based on other thing but id, only the first occurence gonna be deleting .

If u want to delete all the occurences , we use $deleteMany()$

```
db.books.deleteMany(
    {
        author : "Ahmed safrioui"
    }
)
```

Update a doc :

```
db.books.updateOne(
    {
        _id : ObjectId('63fa75616af9c77a2e1b98ee')
    },
    {
        $set : {
            rating : 8,
            pages : 1000
        }
    }
)
```

Modify more than one document at the same time :

```
db.books.updateMany(
    {
        author : "Ahmed safrioui"
    },
    {
        $set : {
            author : "Ashraf khabar"
        }
    }
)
```

How about increasing or decreasing a value (by one or two or ....) :

```
    db.books.updateOne(
        {
            _id : ObjectId('63fa75616af9c77a2e1b98ee')
        },
        {
            $inc : {
                pages : 2
            }
        }
    )
```

How about pulling an element from an array :

```
    db.books.updateOne(
        {
            _id : ObjectId('63fa75616af9c77a2e1b98ee')
        },
        {
            $pull : {
                genres : "Jihaoui"
            }
        }
    )
```

PS : Same for $push$.

But in push we gonna use $each$ operator in order to insert elements into the array :

```
    db.books.updateOne(
        {
            _id : ObjectId('63fa75616af9c77a2e1b98ee')
        },
        {
            $push : {
                genres : {
                    $each : [
                        "Chouawafa",
                        "Gnawa"
                    ]
                }
            }
        }
    )
```