PROJECTION:

We can get documents with only required fields instead of all fields. This is called projection.

Relational databases/sql dabases:
----
without projection: select * from employees;
with projection: select ename,esal from employees;

Projection in MongoDB?
----
db.collection.find({filter}) ===>without projection
db.collection.find({filter},{projection fields}) ===>with projection

Note: If we are providing projection list, compulsory we should provide filter object also, atleast empty java script object. i.e without providing first argument, we cannot talk about second argument.

eg: db.collection.find({},{projection fields})

Case Study:
----
books collection: sample document

```
{
     "title": "Linux in simple way",
     "isbn":  6677,
     "downloadable": false,
     "no_of_reviews": 1,
     "tags": ["os","freeware","shell programming"],
"languages": ["english","hindi","telugu"],
"author": {
        "name": "Shiva Ramachandran",
            "callname": "Shiv",
            "profile": {
                     "exp":8,
                      "courses":3,
                      "books":2
                      }
           }
   }
```

```
db.collection.find({},{projection fields})
```

Q1. To project only title and no_of_reviews?

```
> db.books.find({},{title: 1,no_of_reviews: 1}).pretty()
```

field: 1 ===>means project/include this field in the result
field: 0 ===>means not to project/exclude this field in the result

If we are not taking any field in the projected list, bydefault that field will be excluded. ie default value is 0.

_id field will be included always. But we can exclude this field by assigning with 0 explicitly.

```
> db.books.find({},{title: 1,no_of_reviews: 1}).pretty()
{
        "_id" : ObjectId("5fe95428fe935cdac43627c9"),
        "title" : "Java in simple way",
        "no_of_reviews" : 2
}
{
        "_id" : ObjectId("5fe95428fe935cdac43627ca"),
        "title" : "Linux in simple way",
        "no_of_reviews" : 1
}
{
        "_id" : ObjectId("5fe95428fe935cdac43627cb"),
        "title" : "MongoDB in simple way",
        "no_of_reviews" : 4
}
{
        "_id" : ObjectId("5fe95428fe935cdac43627cc"),
        "title" : "Python in simple way",
        "no_of_reviews" : 5
}
{
        "_id" : ObjectId("5fe95428fe935cdac43627cd"),
        "title" : "Shell Scripting in simple way",
        "no_of_reviews" : 1
}
{
        "_id" : ObjectId("5fe95428fe935cdac43627ce"),
        "title" : "Devops in simple way",
        "no_of_reviews" : 3
}
{
        "_id" : ObjectId("5fe95428fe935cdac43627cf"),
        "title" : "Oracle in simple way",
        "no_of_reviews" : 3
}
```

Note:

```
> db.books.find({},{}).pretty()
```

We will get all documents with all fields. Simply it is equals to:

```
> db.books.find().pretty()
```

Q2. To project only title and no_of_reviews without _id ?

```
> db.books.find({},{title: 1,no_of_reviews: 1, _id: 0}).pretty()

{ "title" : "Java in simple way", "no_of_reviews" : 2 }
{ "title" : "Linux in simple way", "no_of_reviews" : 1 }
{ "title" : "MongoDB in simple way", "no_of_reviews" : 4 }
{ "title" : "Python in simple way", "no_of_reviews" : 5 }
{ "title" : "Shell Scripting in simple way", "no_of_reviews" : 1 }
{ "title" : "Devops in simple way", "no_of_reviews" : 3 }
{ "title" : "Oracle in simple way", "no_of_reviews" : 3 }
```

Q3. Select all documents where no_of_reviews is greater than or equal to 3. Project only the following fields in every document?

1. title
2. no_of_reviews
3. isbn

```
> db.books.find({ no_of_reviews: {$gte: 3}}, {title: 1, no_of_reviews:1,
isbn:1, _id: 0 }).pretty()

{ "title" : "MongoDB in simple way", "isbn" : 6677, "no_of_reviews" : 4 }
{ "title" : "Python in simple way", "isbn" : 1234, "no_of_reviews" : 5 }
{ "title" : "Devops in simple way", "isbn" : 6677, "no_of_reviews" : 3 }
{ "title" : "Oracle in simple way", "isbn" : 6677, "no_of_reviews" : 3 }
```

Projection of Nested Document Fields:

Q4. Project title, author's name and number of books in every document?

```
> db.books.find({},{title: 1, "author.name": 1, "author.profile.books":1,
_id:0 }).pretty()

{
        "title" : "Java in simple way",
        "author" : {
                "name" : "Karhik Ramachandran",
                "profile" : {
                        "books" : 3
                }
        }
}
```

Projection of arrays:

Q. Project title, tags in every document of books collection?

```
> db.books.find({},{ title:1, tags: 1, _id:0}).pretty()

{
        "title" : "Java in simple way",
        "tags" : [
                "language",
                "freeware",
                "programming"
        ]
}
{
        "title" : "Linux in simple way",
        "tags" : [
                "os",
                "freeware",
                "shell programming"
        ]
```

}

Projection of Array Elements | Array Elements Projection Operators:

```
> db.books.find({tags:"programming"}).pretty()
> db.books.find({tags:"programming"},{title:1, tags:1, _id:0}).pretty()
> db.books.find({tags:"programming"},{title:1, "tags.$":1, _id:0}).pretty()
```

We can project array elements by using the following operators:

1. $

2. $elemMatch

3. $slice

1. $ Operator:

We can use $ operator to project first element in an array that matches query condition.

Syntax:

```
db.collection.find({<array>:<condition>,...},{"<array>.$":1})
```

Case Study:

```
db.students.insertOne({_id:1, name:"Durga", year:1, marks:[70,87,90]})
db.students.insertOne({_id:2, name:"Ravi", year:1, marks:[90,88,92]})
db.students.insertOne({_id:3, name:"Shiva", year:1, marks:[85,100,90]})
db.students.insertOne({_id:4, name:"Durga", year:2, marks:[79,85,80]})
db.students.insertOne({_id:5, name:"Ravi", year:2, marks:[88,88,92]})
db.students.insertOne({_id:6, name:"Shiva", year:2, marks:[95,90,96]})
```

Q1. db.students.find({marks:{$gte: 85}},{_id:0,marks:1})

```
> db.students.find({marks:{$gte: 85}},{_id:0,marks:1})
{ "marks" : [ 70, 87, 90 ] }
{ "marks" : [ 90, 88, 92 ] }
{ "marks" : [ 85, 100, 90 ] }
{ "marks" : [ 79, 85, 80 ] }
{ "marks" : [ 88, 88, 92 ] }
{ "marks" : [ 95, 90, 96 ] }
```

In this case all elements of array projected.

Q2. db.students.find({marks:{$gte: 85}},{_id:0,name: 1, "marks.$":1})
Now instead of all elements, only first matched element will be projected.

```
> db.students.find({marks:{$gte: 85}},{_id:0,name: 1, "marks.$":1})
{ "name" : "Durga", "marks" : [ 87 ] }
{ "name" : "Ravi", "marks" : [ 90 ] }
{ "name" : "Shiva", "marks" : [ 85 ] }
{ "name" : "Durga", "marks" : [ 85 ] }
{ "name" : "Ravi", "marks" : [ 88 ] }
{ "name" : "Shiva", "marks" : [ 95 ] }
```

Q3. db.students.find({marks:{$all: [88,90]}},{_id:0,name: 1, "marks.$":1})

```
{ "name" : "Ravi", "marks" : [ 90 ] }
```

Note: If there is no query condition or if query condition won't include array then we cannot use $ operator, otherwise we will get error.

eg-1:

```
> db.students.find({},{_id:0,name: 1, "marks.$":1})
Error: error: {
	"ok" : 0,
	"errmsg" : "positional operator '.$' couldn't find a matching
element in the array",
	"code" : 51246,
	"codeName" : "Location51246"
}
```

eg1:

```
> db.students.find({year: 1},{_id:0,name: 1, "marks.$":1})
Error: error: {
	"ok" : 0,
	"errmsg" : "positional operator '.$' couldn't find a matching
element in the array",
	"code" : 51246,
	"codeName" : "Location51246"
}
```

***Note: $ operator selects only one element which is first matched element based on query condition.

2. $elemMatch operator:

----

1. selects only one element
2. which is matched element where condition is specified by $elemMatch explicitly.

It never considers query condition.

We can use $elemMatch to project first element in the array that matches specified $elemMatch condition.

Q1.

```
> db.students.find({},{_id:0, name:1,year:1,marks:{$elemMatch:{$lt:
95}}})

{ "marks" : [ 70, 87, 90 ] }
{ "marks" : [ 90, 88, 92 ] }
{ "marks" : [ 85, 100, 90 ] }
{ "marks" : [ 79, 85, 80 ] }
{ "marks" : [ 88, 88, 92 ] }
{ "marks" : [ 95, 90, 96 ] }

{ "name" : "Durga", "year" : 1, "marks" : [ 70 ] }
{ "name" : "Ravi", "year" : 1, "marks" : [ 90 ] }
{ "name" : "Shiva", "year" : 1, "marks" : [ 85 ] }
{ "name" : "Durga", "year" : 2, "marks" : [ 79 ] }
{ "name" : "Ravi", "year" : 2, "marks" : [ 88 ] }
{ "name" : "Shiva", "year" : 2, "marks" : [ 90 ] }


> db.students.find({year:1},{_id:0,
name:1,year:1,marks:{$elemMatch:{$gt: 85}}})
{ "name" : "Durga", "year" : 1, "marks" : [ 87 ] }
{ "name" : "Ravi", "year" : 1, "marks" : [ 90 ] }
{ "name" : "Shiva", "year" : 1, "marks" : [ 100 ] }
```

What is the difference between $ and $elemMatch operators:
--
Both operators project the first matching element from an array based on a condition.

$ operator will select array element based on query condition. But $elemMatch will select array element based on explicit condition specified by $elemMatch but not based on query condition.

```
> db.students.find({year:1,marks:{$gte: 85}},{_id:0,name:1,"marks.$":1})
{ "name" : "Durga", "marks" : [ 87 ] }
{ "name" : "Ravi", "marks" : [ 90 ] }
{ "name" : "Shiva", "marks" : [ 85 ] }

> db.students.find({year:1,marks:{$gte:
85}},{_id:0,name:1,marks:{$elemMatch:{$gt:89}}})

{ "name" : "Durga", "marks" : [ 90 ] }
{ "name" : "Ravi", "marks" : [ 90 ] }
{ "name" : "Shiva", "marks" : [ 100 ] }
```

3. $slice operator:

By using $slice operator we can select required number of elements in the array.

Syntax-1:

```
db.collection.find({query},{<array>:{$slice: n}})
```

n-->number of elements to be selected.
Specify a positive number n to return the first n elements.
Specify a negative number n to return the last n elements.
If n is greater than number of elements in the array then all elements will be selected.

eg-1:

```
> db.students.find({},{_id:0,name:1,year:1, marks:{$slice: 2}})
```

In the array only first 2 elements will be selected.

```
{ "name" : "Durga", "year" : 1, "marks" : [ 70, 87 ] }
{ "name" : "Ravi", "year" : 1, "marks" : [ 90, 88 ] }
{ "name" : "Shiva", "year" : 1, "marks" : [ 85, 100 ] }
{ "name" : "Durga", "year" : 2, "marks" : [ 79, 85 ] }
{ "name" : "Ravi", "year" : 2, "marks" : [ 88, 88 ] }
{ "name" : "Shiva", "year" : 2, "marks" : [ 95, 90 ] }
```

eg-2:

```
> db.students.find({},{_id:0,name:1,year:1, marks:{$slice: -2}})
```

In the array only last 2 elements will be selected.

```
> db.students.find({},{_id:0,name:1,year:1, marks:{$slice: -2}})
{ "name" : "Durga", "year" : 1, "marks" : [ 87, 90 ] }
{ "name" : "Ravi", "year" : 1, "marks" : [ 88, 92 ] }
{ "name" : "Shiva", "year" : 1, "marks" : [ 100, 90 ] }
{ "name" : "Durga", "year" : 2, "marks" : [ 85, 80 ] }
{ "name" : "Ravi", "year" : 2, "marks" : [ 88, 92 ] }
{ "name" : "Shiva", "year" : 2, "marks" : [ 90, 96 ] }
```

eg-3:

```
> db.students.find({},{_id:0,name:1,year:1, marks:{$slice: 100}})
```

In this case all elements will be included.

```
{ "name" : "Durga", "year" : 1, "marks" : [ 70, 87, 90 ] }
{ "name" : "Ravi", "year" : 1, "marks" : [ 90, 88, 92 ] }
{ "name" : "Shiva", "year" : 1, "marks" : [ 85, 100, 90 ] }
{ "name" : "Durga", "year" : 2, "marks" : [ 79, 85, 80 ] }
{ "name" : "Ravi", "year" : 2, "marks" : [ 88, 88, 92 ] }
{ "name" : "Shiva", "year" : 2, "marks" : [ 95, 90, 96 ] }
```

Syntax-2:

----

db.collection.find({query},{<array>:{$slice: [n1,n2]}})
skip n1 number of elements and then select n2 number of elements.

n1--->number to skip
n2--->number to return

eg-1:
skip first element and then select next two elements.

```
> db.students.find({year:1},{_id:0,name:1, marks:{$slice: [1,2]}})
{ "name" : "Durga", "marks" : [ 87, 90 ] }
{ "name" : "Ravi", "marks" : [ 88, 92 ] }
{ "name" : "Shiva", "marks" : [ 100, 90 ] }
```

eg-2: skip first 2 elements and select next 10 elements.

```
> db.students.find({year:1},{_id:0,name:1, marks:{$slice: [2,10]}})

{ "name" : "Durga", "marks" : [ 90 ] }
{ "name" : "Ravi", "marks" : [ 92 ] }
{ "name" : "Shiva", "marks" : [ 90 ] }
```

eg-3: required only 7th element in the array?

```
> db.students.find({},{_id:0,name:1, marks:{$slice: [6,1]}})
```

eg-4: required from 3rd to 10th elements

```
> db.students.find({},{_id:0,name:1, marks:{$slice: [2,8]}})
```

CRUD Operations
C--->Create Operation | Insert Operation
R--->Retrieve Operation | Read Operation
U--->Update Operation