Projection operators

Projection operators in MongoDB are used to specify the fields to include or exclude in documents returned by a query. By default, all fields of a document are returned. However, with projection, you can explicitly define which fields to retrieve, which is especially useful in large documents or collections.

Purpose of using Projection Operators in queries:

- Performance improvement
- Resource Optimization
- Focused data retrieval

Create and demonstrate how projection operators ($,$elements and $slice) would be used in the MongoDB.

Types of projection operators:

- Inclusion:

The Inclusion is a basic form of projection where you specify which fields to include in the result of your query. Here only the id field is included, but you can include other fields by setting them to 1 in your projection document.

```
db.collection.find({}, { field1: 1, field2: 1 })
```

- Exclusion:

Here in the Exclusion involves specifying which fields not to include in the query results. This is done by setting the field value to 0 in the projection document.

```
db.collection.find({}, { field1: 0, field2: 0 }
```

Positional Operators:

- **$elemMatch:**

The $elemMatch operator matches documents in a collection that contain an array field with at least one element that satisfies multiple given conditions.

```
db.collection.find({},{arrayField:{$elemMatch:{field:value}}})
```

- $slice:

The $slice operator is used to limit the number of elements projected from an array. It can either return the first N elements, skip the first N elements, or return elements after skipping N elements.

```
db.collection.find({}, { arrayField: { $slice: [skip, limit] } })
```

- $:

The $ operator is used to project the first element in an array that matches the specified condition. It is especially useful when dealing with large arrays, and you only need the first element matching a given condition.

```
db.collection.find({ "arrayField.field": value }, { "arrayField.$":
1 })
```

Computed Fields and Expressions:

- $meta:

Projects the metadata (e.g., text search score) for a document.

```
db.collection.find({}, { score: { $meta: "textScore" } })
```

- $slice:

Projects a subset od an array.

```
db.collection.find({}, { arrayField: { $slice: limit } })
```

Aggregation Framework operators:

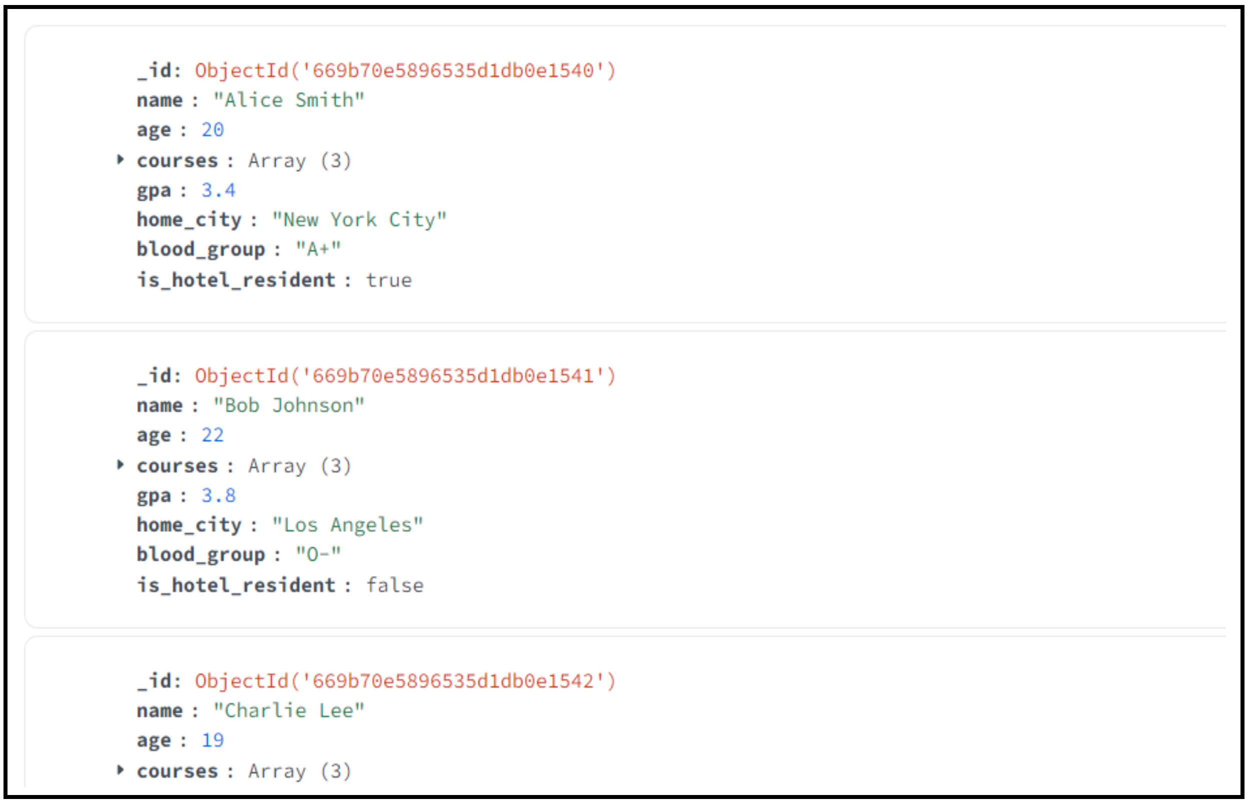
- $project:

The $project is a projection operator in MongoDB, which is used during the aggregation process to reshape/output a document by specifying the fields to include or exclude. This is particularly helpful when you need to limit the amount of data retrieved from the database or modify the structure of the result.

```
db.collection.aggregate([

 { $project: { field1: 1, field2: 1, computedField: { $add: ["$fieldA", "$fieldB"] } } }

]);
```

For example :

Let’s assume we have a “stu” collection with documents that look like this:



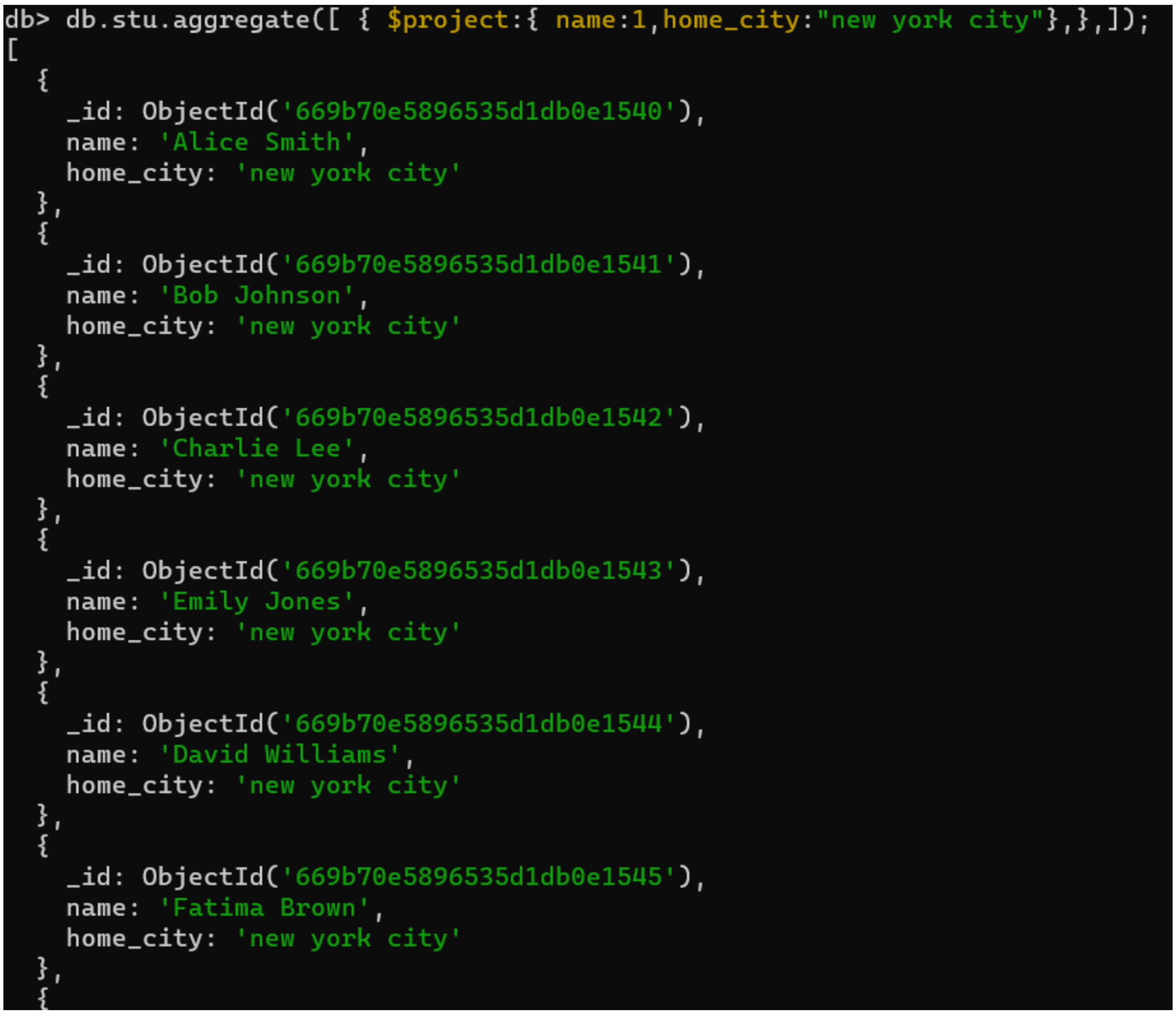
```
_id: ObjectId('669b70e5896535d1db0e1540')
name : "Alice Smith"
age : 20
▸ courses : Array (3)
gpa : 3.4
home_city : "New York City"
blood_group : "A+"
is_hotel_resident : true

_id: ObjectId('669b70e5896535d1db0e1541')
name : "Bob Johnson"
age : 22
▸ courses : Array (3)
gpa : 3.8
home_city : "Los Angeles"
blood_group : "O-"
is_hotel_resident : false

_id: ObjectId('669b70e5896535d1db0e1542')
name : "Charlie Lee"
age : 19
▸ courses : Array (3)
```

If you want to retrieve only the name and the total number of posts for each user, you can execute the following aggregate query with a $project operator:

```
db.users.aggregate([
   {
     $project:{
        name:1,
        home_city: "new york city"
       },
    },
]);
```

Here, we are including the name field and calculating the totalPosts value with the $size operator.

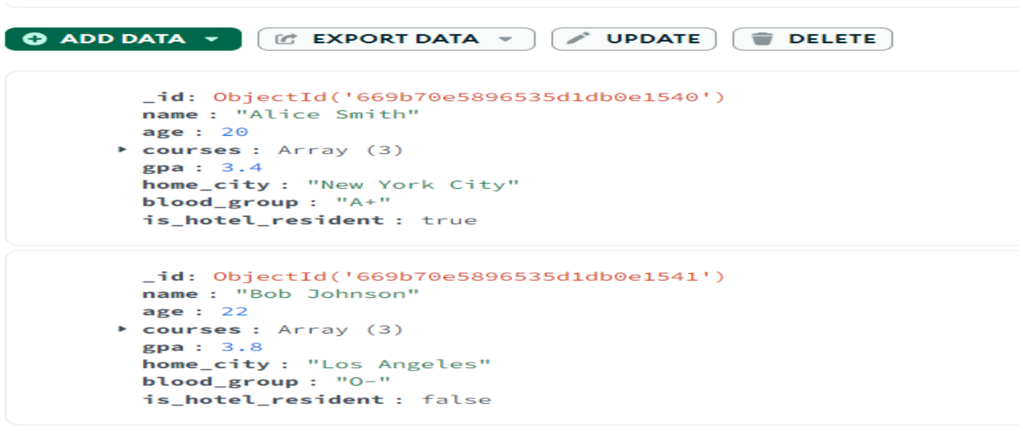
Output look like this:



```
db> db.stu.aggregate([ { $project:{ name:1,home_city:"new york city"},},]);
[
  {
    _id: ObjectId('669b70e5896535d1db0e1540'),
    name: 'Alice Smith',
    home_city: 'new york city'
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1541'),
    name: 'Bob Johnson',
    home_city: 'new york city'
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1542'),
    name: 'Charlie Lee',
    home_city: 'new york city'
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1543'),
    name: 'Emily Jones',
    home_city: 'new york city'
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1544'),
    name: 'David Williams',
    home_city: 'new york city'
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1545'),
    name: 'Fatima Brown',
    home_city: 'new york city'
  },
  {
```

An example of using projection operators in MongoDB:

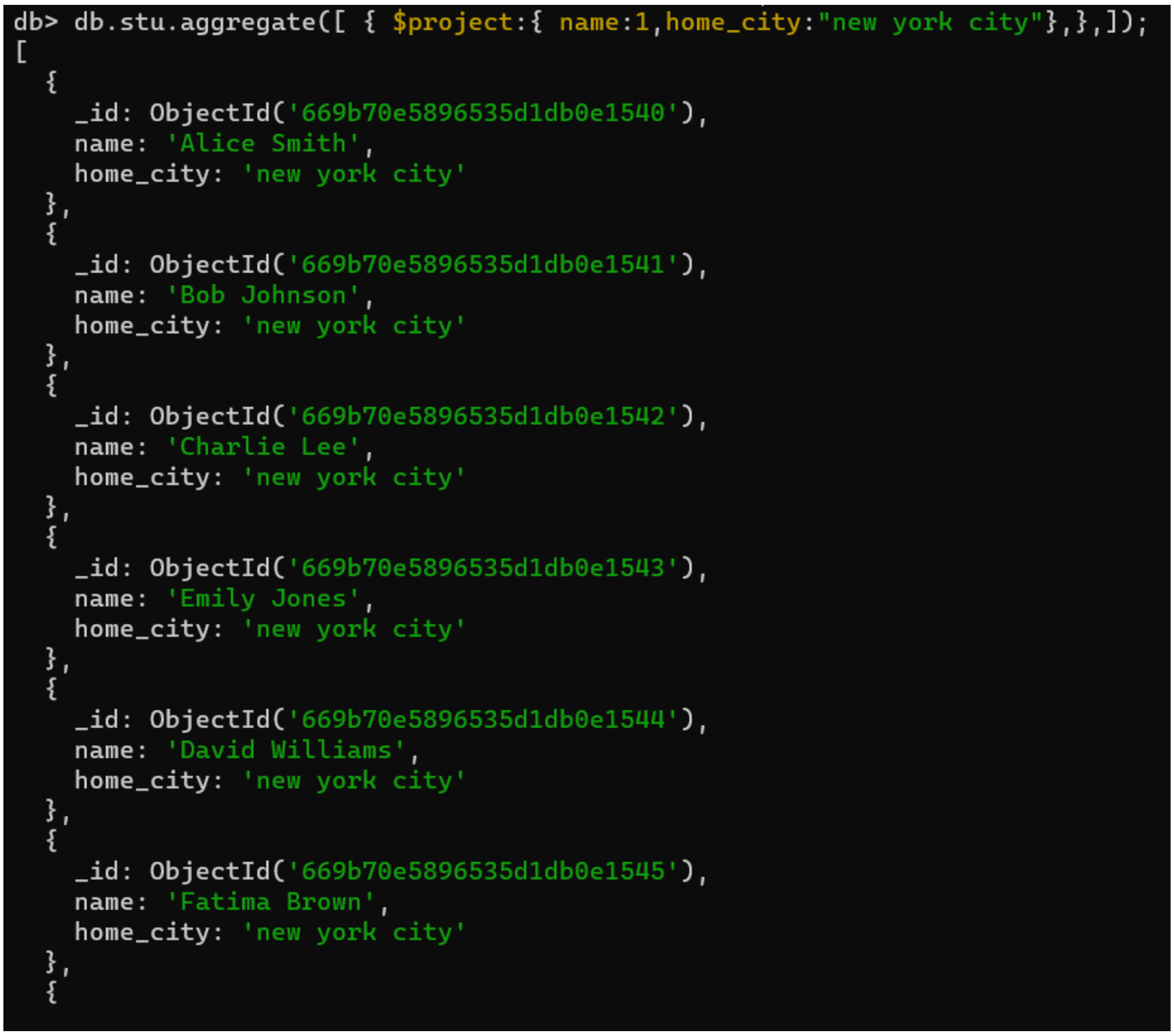
Let’s we assume that we have a MongoDB collection named stu with the following documents:



```
_id: ObjectId('669b70e5896535d1db0e1540')
name : "Alice Smith"
age : 20
courses : Array (3)
gpa : 3.4
home_city : "New York City"
blood_group : "A+"
is_hotel_resident : true

_id: ObjectId('669b70e5896535d1db0e1541')
name : "Bob Johnson"
age : 22
courses : Array (3)
gpa : 3.8
home_city : "Los Angeles"
blood_group : "O-"
is_hotel_resident : false
```

Lets we retrieve only name,age,and gpa.

Here the code,



```
db> db.stu.aggregate([ { $project:{ name:1,home_city:"new york city"},},]);
[
  {
    _id: ObjectId('669b70e5896535d1db0e1540'),
    name: 'Alice Smith',
    home_city: 'new york city'
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1541'),
    name: 'Bob Johnson',
    home_city: 'new york city'
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1542'),
    name: 'Charlie Lee',
    home_city: 'new york city'
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1543'),
    name: 'Emily Jones',
    home_city: 'new york city'
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1544'),
    name: 'David Williams',
    home_city: 'new york city'
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1545'),
    name: 'Fatima Brown',
    home_city: 'new york city'
  },
  {
```

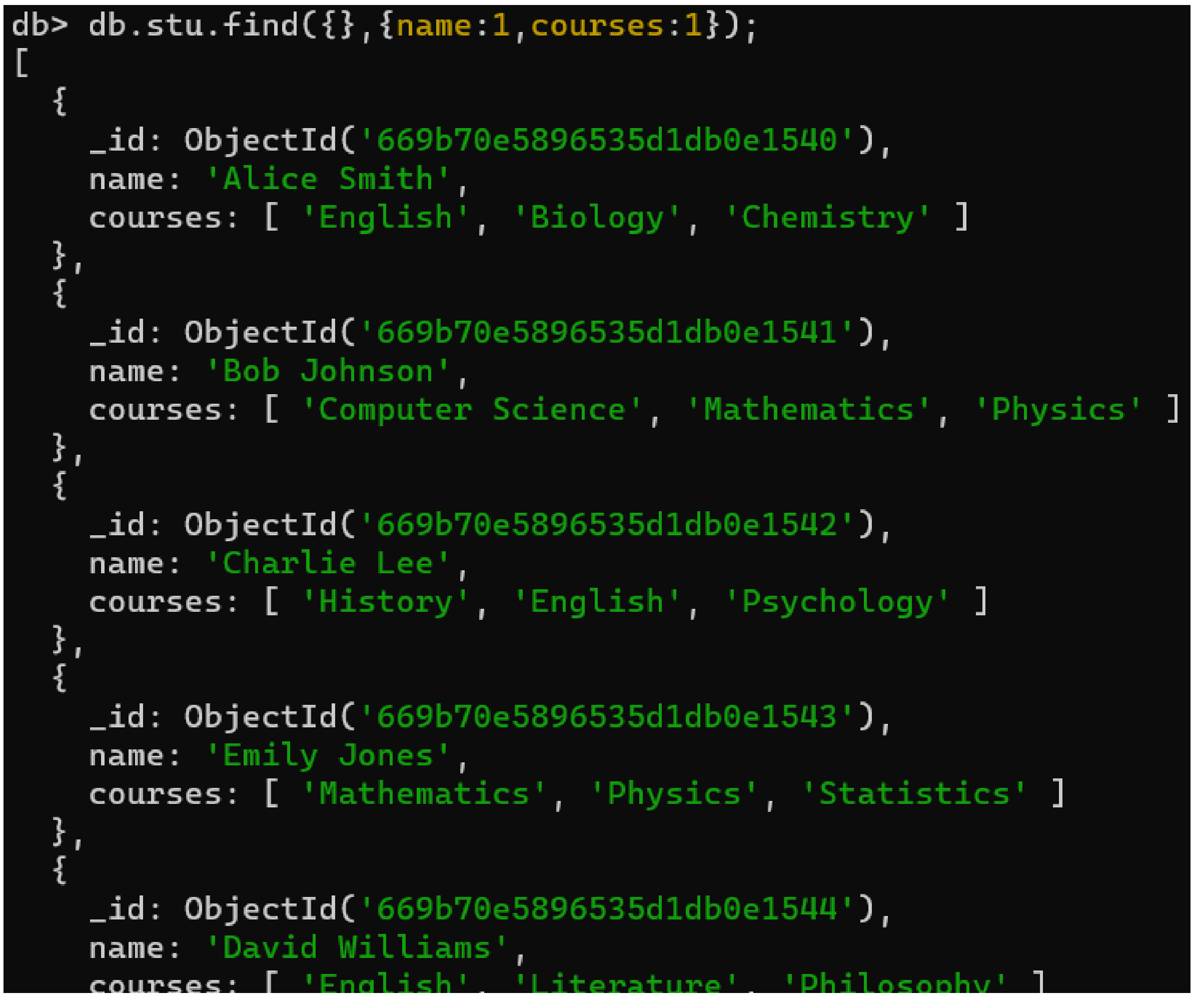
As per the above code we got ‘stu’ collection of name ,age and gpa.

1.Including specific fields:

In including field we retrieve only we wanted data others are excluded.

Now we retrieve only *name* and *courses* using below inclusion command.

db.users.find({}, { name: 1, email: 1, _id: 0 })



```
db> db.stu.find({},{name:1,courses:1});
[
  {
    _id: ObjectId('669b70e5896535d1db0e1540'),
    name: 'Alice Smith',
    courses: [ 'English', 'Biology', 'Chemistry' ]
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1541'),
    name: 'Bob Johnson',
    courses: [ 'Computer Science', 'Mathematics', 'Physics' ]
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1542'),
    name: 'Charlie Lee',
    courses: [ 'History', 'English', 'Psychology' ]
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1543'),
    name: 'Emily Jones',
    courses: [ 'Mathematics', 'Physics', 'Statistics' ]
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1544'),
    name: 'David Williams',
    courses: [ 'English', 'Literature', 'Philosophy' ]
```

As you can see here the name and courses is included and rest are excluded.

Here in the example,

- {} is the query part means want to retrieve all documents.
- {name:1,courses:1}is the projection part,
- where name:1 includes the name field.
- If give name:0 and courses:1,it includes only name.
- In the above name:1 includes the name.
- Courses:1 includes the courses.

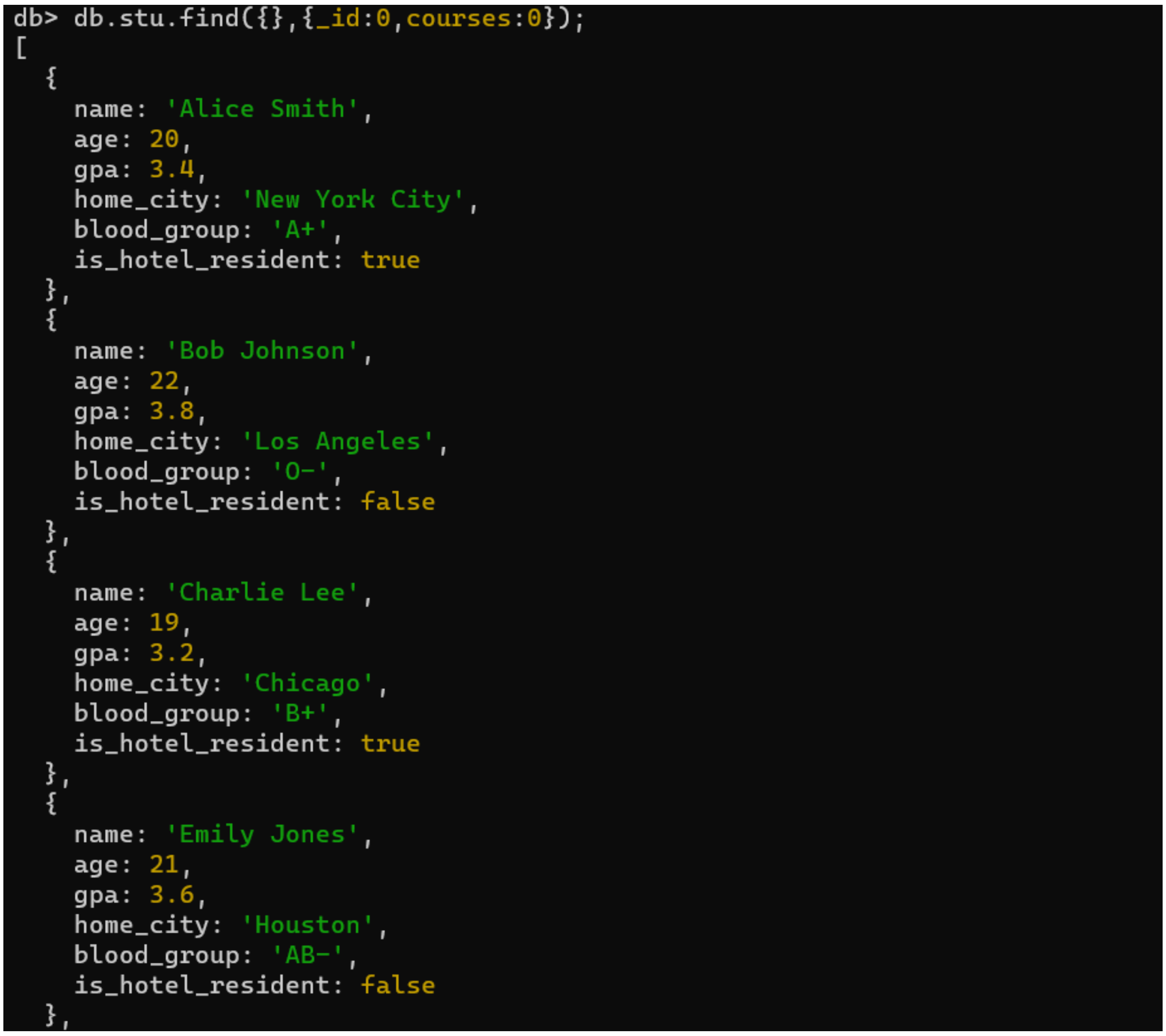
2.Excluding specific fields:

The exclude operator, as the name suggests, helps you to exclude certain fields from the result.

To exclude a field from the query result, you need to set its value to 0 in the projection document. Let's understand it better with an example.

Now we exclude the _id and courses using exclusion command.

db.stu.find({},{_id:0,courses:0});



```
db> db.stu.find({},{_id:0,courses:0});
[
  {
    name: 'Alice Smith',
    age: 20,
    gpa: 3.4,
    home_city: 'New York City',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    name: 'Bob Johnson',
    age: 22,
    gpa: 3.8,
    home_city: 'Los Angeles',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Charlie Lee',
    age: 19,
    gpa: 3.2,
    home_city: 'Chicago',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Emily Jones',
    age: 21,
    gpa: 3.6,
    home_city: 'Houston',
    blood_group: 'AB-',
    is_hotel_resident: false
  },
```

As you can see here the _id and courses is excluded and rest are included.

Here in the example,

- {} is the query part means want to retrieve all documents.
- {_id:0,courses:1}is the projection part,

- where _id:0 excludes the id field.
- Courses:0 excludes the courses and rest included.

3.$eleMatch:

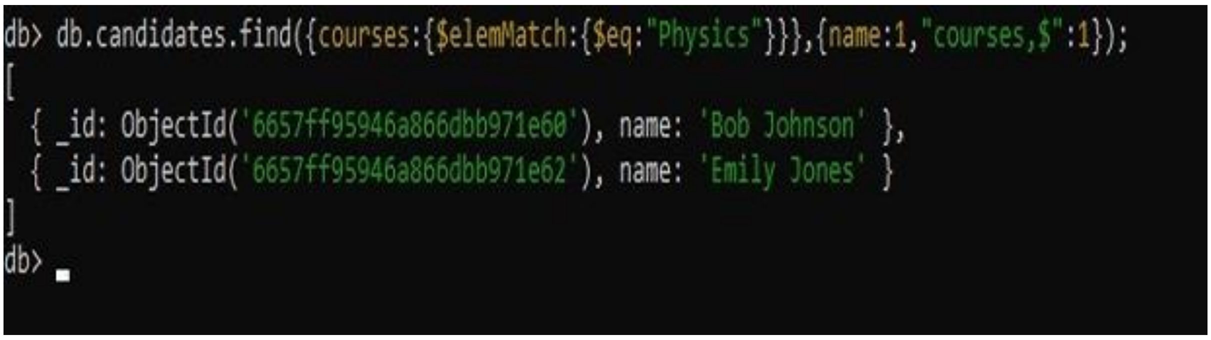
$elemMatch is an array operator in MongoDB that is used to select documents that contain an array field with at least one element matching the specified query criteria. This is useful in situations when you need to match multiple criteria within the same array element.

Usage:

To use $elemMatch, you need to include it in your query with the syntax **{ <field>: { $elemMatch: { <query> } } }.**

- **<field>:** The name of the array field for which you want to apply the $elemMatch operator.
- **<query>:** A document containing the query conditions to be matched against the elements in the array.

Here the example :



```
db> db.candidates.find({courses:{$elemMatch:{$eq:"Physics"}}},{name:1,"courses,$":1});
[
  { _id: ObjectId('6657ff95946a866dbb971e60'), name: 'Bob Johnson' },
  { _id: ObjectId('6657ff95946a866dbb971e62'), name: 'Emily Jones' }
]
db> 
```

- **Collection:** The query is executed on the candidates collection.

4.$slice:

The $slice projection operator is a MongoDB feature that allows you to limit the number of elements returned for an array field within the documents. This is particularly useful when you have large arrays in your documents, and you only need to work with a specific portion of them. By applying the $slice operator, you can optimize your queries and minimize memory usage.

Syntax:

The $slice has one of the following syntax forms:

```
db.collection.find(
  <query>,
  { <arrayField>: { $slice: <number> } }
);
```

Usage

The $slice operator can be used in two forms:

- Limit the number of array elements returned, starting from the beginning of the array.
- Limit the number of array elements returned, starting from a specific position in the array.

Value	Description
`$slice: <number>`	Specifies the number of elements to return in the `<arrayField>`. For `<number>`: • Specify a positive number `n` to return the first `n` elements. • Specify a negative number `n` to return the last `n` elements. If the `<number>` is greater than the number of array elements, the query returns all array elements.
`$slice: [ <number to skip>, <number to return> ]`	Specifies the number of elements to return in the `<arrayField>` after skipping the specified number of elements starting from the first element. You must specify both elements.

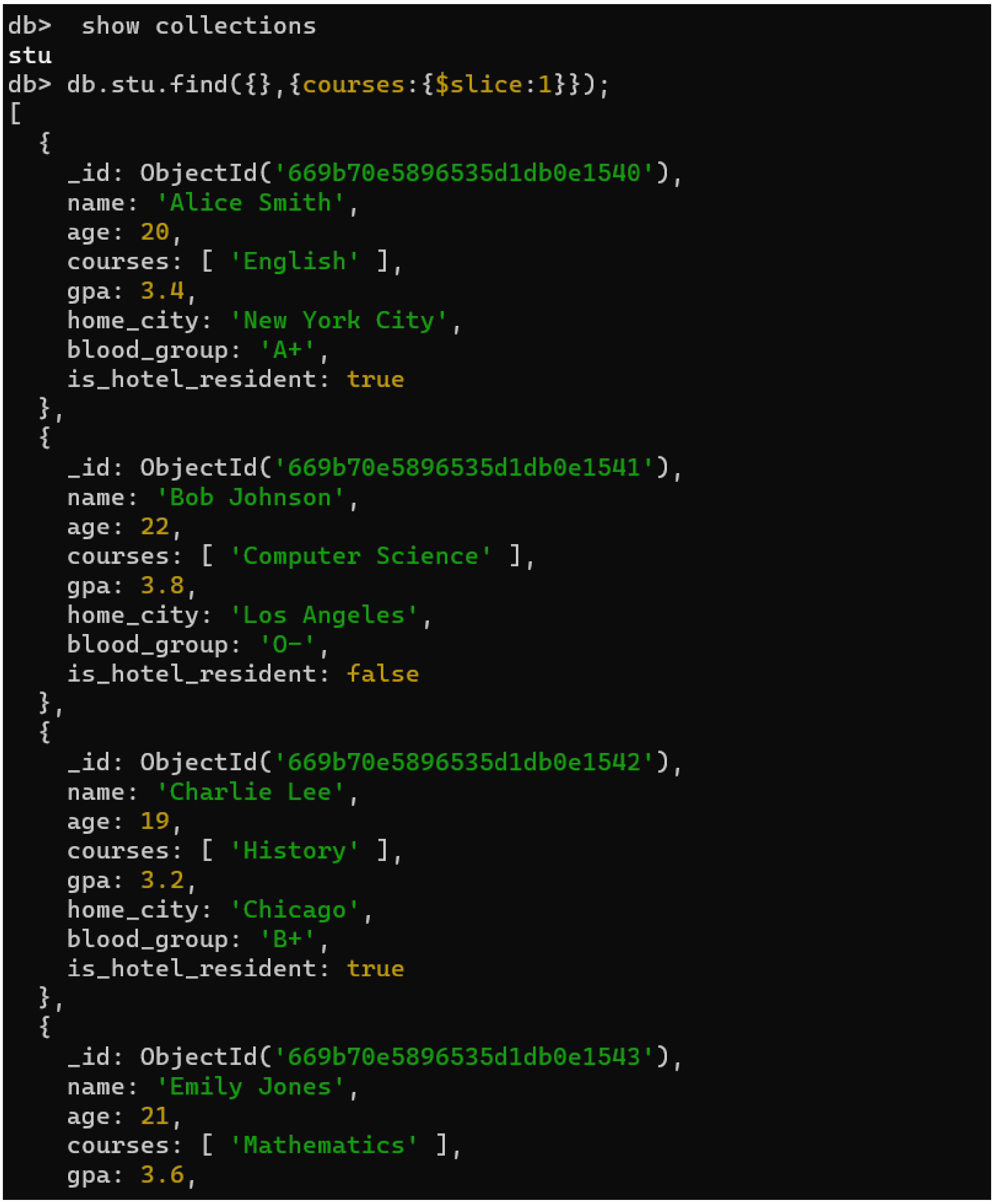
Value	Description
	For the `<number to skip>`: • Specify a positive number `n` to skip `n` elements from the start of the array; i.e. 0th index position. Based on a zero-based array index, `1` indicates the starting position of the 2nd element, etc. If `n` is greater than the number of array elements, the query returns an empty array for the `<arrayField>`. • Specify a negative number `n` to skip backward `n` elements from the start of the array; i.e. 0th index position Based on a zero-based array index (i.e. the first element is at index 0), `-1` indicates the starting position of the last element, etc. If the absolute value of the negative number is greater than the number of array elements, the starting position is the start of the array. For the `<number to return>`, you must specify a *positive* number `n` to return the next `n` elements, starting after skipping the specified number.

$slice operation:

The $slice operator in MongoDB is used to project a subset of elements from an array. It can be used both in queries and updates, as well as in aggregation pipelines to control which elements of an array field are included in the result.

db.candidates.find({}): This part of the query is selecting all documents in the candidates collection.

db.candidates.find({}, {courses: {$slice: 1}})

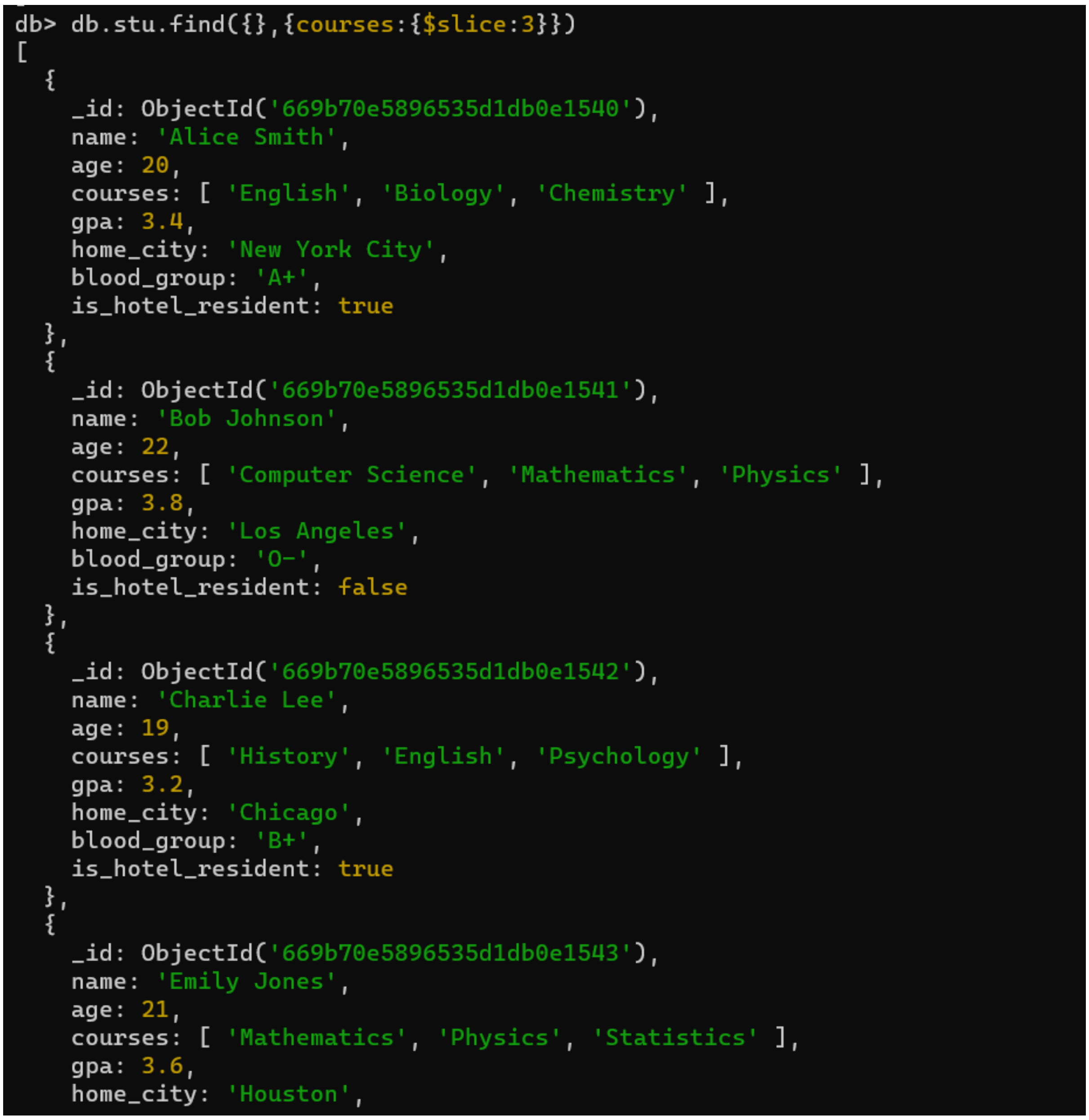


```
db>  show collections
stu
db> db.stu.find({},{courses:{$slice:1}});
[
  {
    _id: ObjectId('669b70e5896535d1db0e1540'),
    name: 'Alice Smith',
    age: 20,
    courses: [ 'English' ],
    gpa: 3.4,
    home_city: 'New York City',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1541'),
    name: 'Bob Johnson',
    age: 22,
    courses: [ 'Computer Science' ],
    gpa: 3.8,
    home_city: 'Los Angeles',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1542'),
    name: 'Charlie Lee',
    age: 19,
    courses: [ 'History' ],
    gpa: 3.2,
    home_city: 'Chicago',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1543'),
    name: 'Emily Jones',
    age: 21,
    courses: [ 'Mathematics' ],
    gpa: 3.6,
```

This query does the following:

- **db.stu.find({}):** This part of the query is selecting all documents in the candidates collection.
- **{courses:{$slice: 1}}:** This projection specifies that only the first element of the courses array should be included in the result.

$slice operation[1:3]:



```
db> db.stu.find({},{courses:{$slice:3}})
[
  {
    _id: ObjectId('669b70e5896535d1db0e1540'),
    name: 'Alice Smith',
    age: 20,
    courses: [ 'English', 'Biology', 'Chemistry' ],
    gpa: 3.4,
    home_city: 'New York City',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1541'),
    name: 'Bob Johnson',
    age: 22,
    courses: [ 'Computer Science', 'Mathematics', 'Physics' ],
    gpa: 3.8,
    home_city: 'Los Angeles',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1542'),
    name: 'Charlie Lee',
    age: 19,
    courses: [ 'History', 'English', 'Psychology' ],
    gpa: 3.2,
    home_city: 'Chicago',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('669b70e5896535d1db0e1543'),
    name: 'Emily Jones',
    age: 21,
    courses: [ 'Mathematics', 'Physics', 'Statistics' ],
    gpa: 3.6,
    home_city: 'Houston',
```

- **db.stu.find({}):** Selects all documents in the candidates collection.
- **{courses: {$slice: 3}}:** Specifies that only the first 3 elements of the courses array should be included in the result.
- The query returns all documents in the candidates collection, but for each document, it only includes the first 3 elements of the courses array.

Before Slicing:

Consider a document in the candidates collection before applying the $slice

operation:

```
                {
  "_id": ObjectId("6657f9f9946a866dbb971e5f"),
  "name": "Alice Smith",
  "age": 20,
  "courses": ["English", "Biology", "Chemistry", "Physics"],
  "gpa": 3.4,
  "home_city": "New York City",
  "blood_group": "A+",
  "is_hotel_resident": true
}
```

After Slicing:

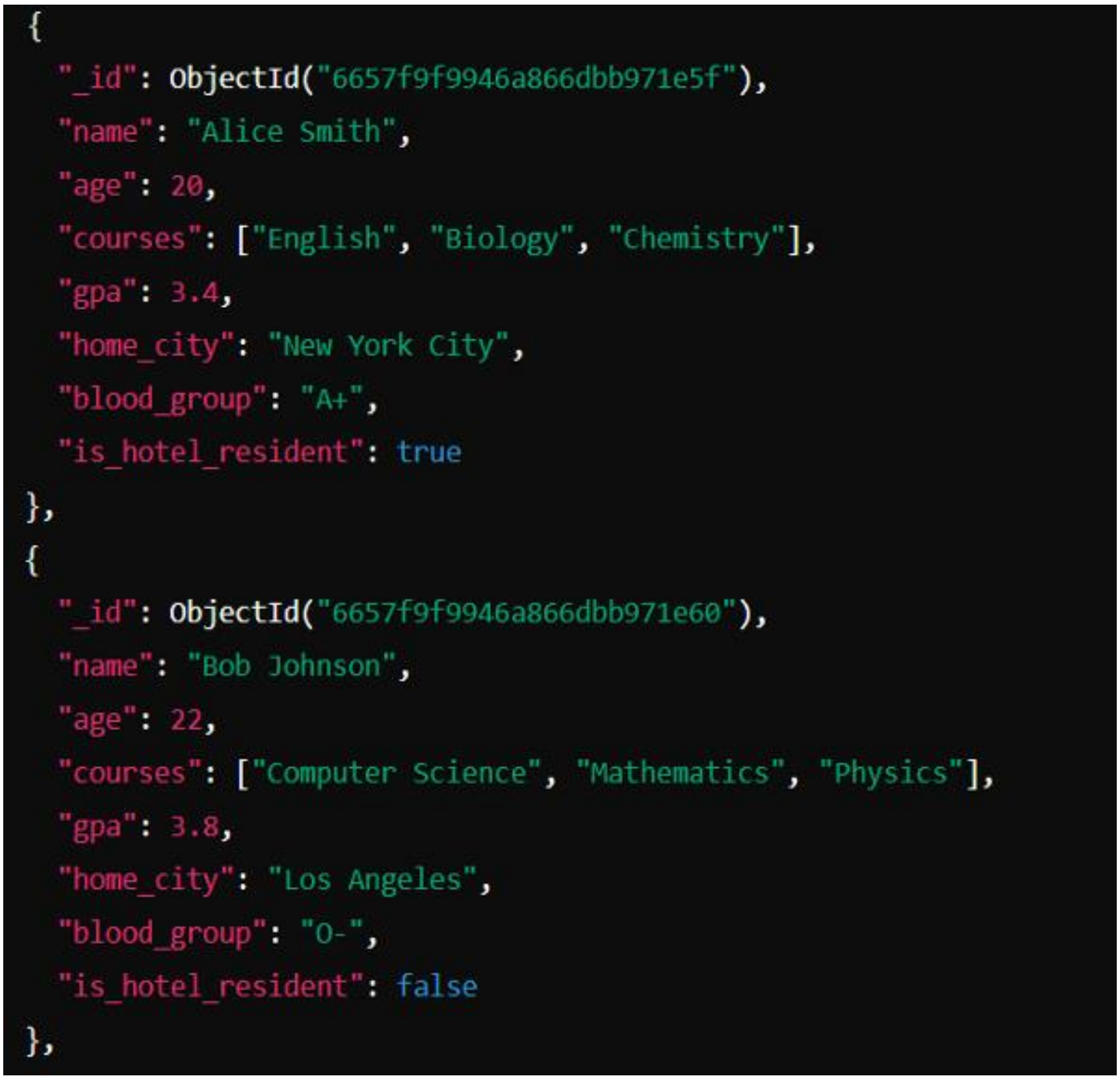
After applying the $slice: 3 projection, the document would look like:

```
                              {
  "_id": ObjectId("6657f9f9946a866dbb971e5f"),
  "name": "Alice Smith",
  "age": 20,
  "courses": ["English", "Biology", "Chemistry"],
  "gpa": 3.4,
  "home_city": "New York City",
  "blood_group": "A+",
  "is_hotel_resident": true
}
```

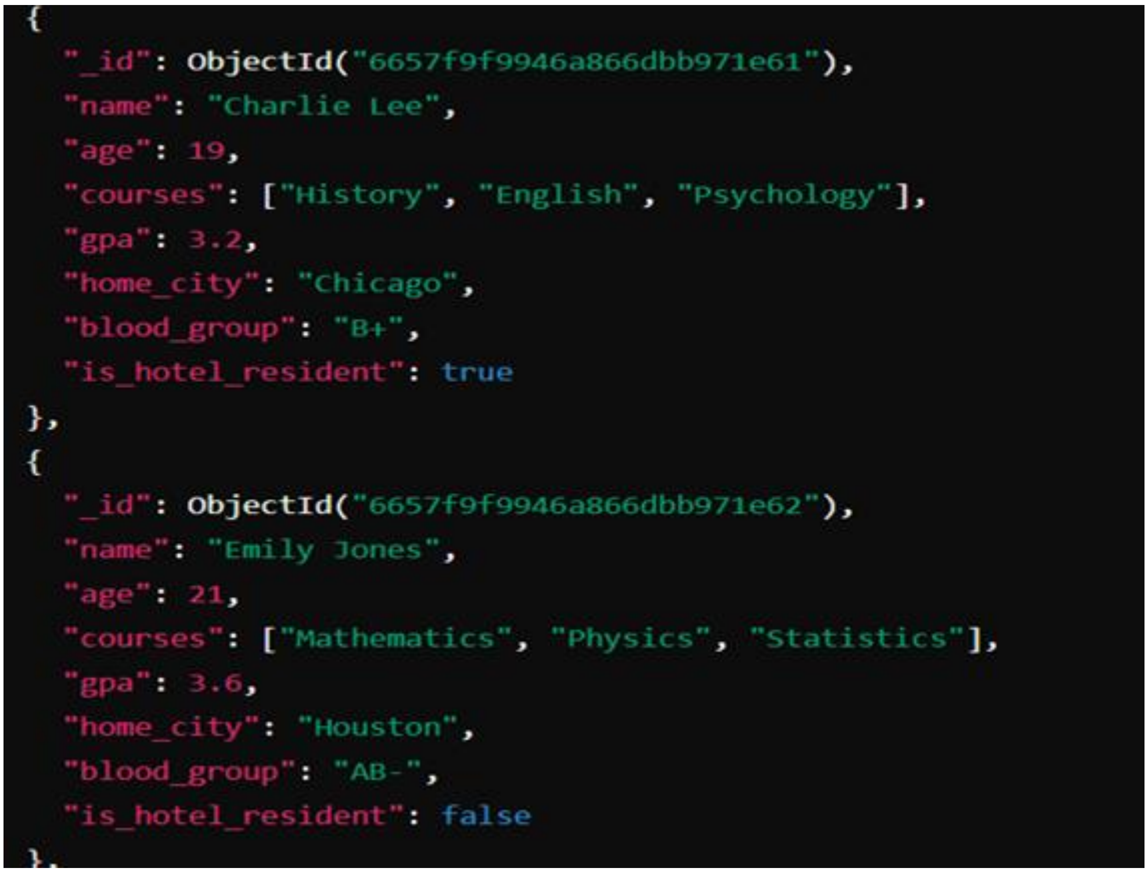
Only the first 3 elements of the courses array are included in the output.

Result:

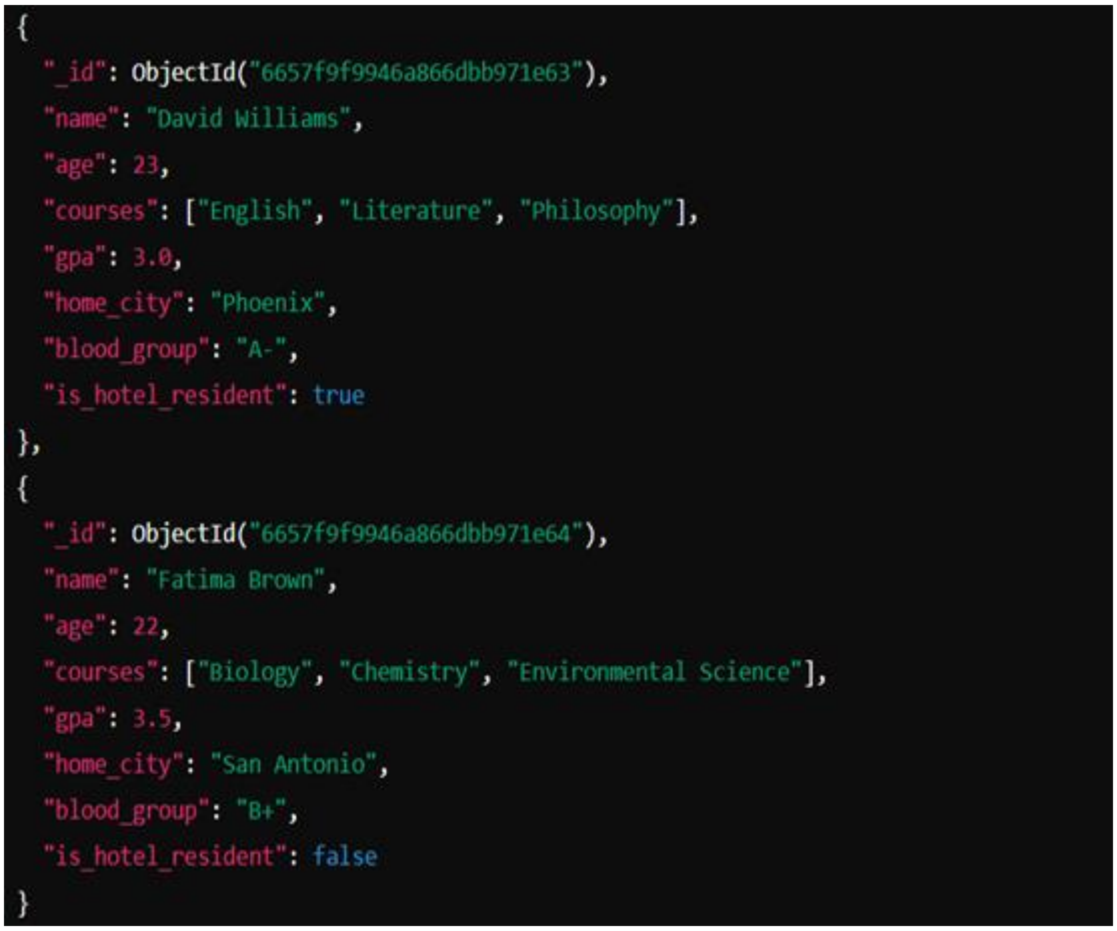
The results in the image show several documents from the candidates collection after applying the $slice operation. Each document includes only the first 3 elements of the courses array:



```
{
  "_id": ObjectId("6657f9f9946a866dbb971e5f"),
  "name": "Alice Smith",
  "age": 20,
  "courses": ["English", "Biology", "Chemistry"],
  "gpa": 3.4,
  "home_city": "New York City",
  "blood_group": "A+",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6657f9f9946a866dbb971e60"),
  "name": "Bob Johnson",
  "age": 22,
  "courses": ["Computer Science", "Mathematics", "Physics"],
  "gpa": 3.8,
  "home_city": "Los Angeles",
  "blood_group": "O-",
  "is_hotel_resident": false
},
```



```
{
  "_id": ObjectId("6657f9f9946a866dbb971e61"),
  "name": "Charlie Lee",
  "age": 19,
  "courses": ["History", "English", "Psychology"],
  "gpa": 3.2,
  "home_city": "Chicago",
  "blood_group": "B+",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6657f9f9946a866dbb971e62"),
  "name": "Emily Jones",
  "age": 21,
  "courses": ["Mathematics", "Physics", "Statistics"],
  "gpa": 3.6,
  "home_city": "Houston",
  "blood_group": "AB-",
  "is_hotel_resident": false
},
```



```
{
  "_id": ObjectId("6657f9f9946a866dbb971e63"),
  "name": "David Williams",
  "age": 23,
  "courses": ["English", "Literature", "Philosophy"],
  "gpa": 3.0,
  "home_city": "Phoenix",
  "blood_group": "A-",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6657f9f9946a866dbb971e64"),
  "name": "Fatima Brown",
  "age": 22,
  "courses": ["Biology", "Chemistry", "Environmental Science"],
  "gpa": 3.5,
  "home_city": "San Antonio",
  "blood_group": "B+",
  "is_hotel_resident": false
}
```

The query db.candidates.find({}, {courses: {$slice: 3}}) retrieves all documents from the candidates collection but limits the courses array in each document to only include the first 3 elements.