PROJECTION OPERATORS

Projection operators in MongoDB are used to shape the documents returned in the query results by including, excluding, or manipulating fields. Here are some commonly used projection operators.

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

In MongoDB, projection operators are used to control the inclusion, exclusion, and transformation of fields in the documents returned by a query. Here are the common projection operators available in MongoDB:

Field Inclusion and Exclusion Operators:

1.Inclusion: Include specific fields in the output.

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

2.Exclusion: Exclude specific fields from the output.

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

Positional Operators:

3.$elemMatch: Projects the first matching element from an array based on a specified condition.

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

4.$slice: Limits the number of elements projected from an array.

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

5.$: Projects the first element in an array that matches the query condition.

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

Computed Fields and Expressions:

6.$meta: Projects the metadata (e.g., text search score) for a document.

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

7.$slice: Projects a subset of an array.

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

Aggregation Framework Operators:

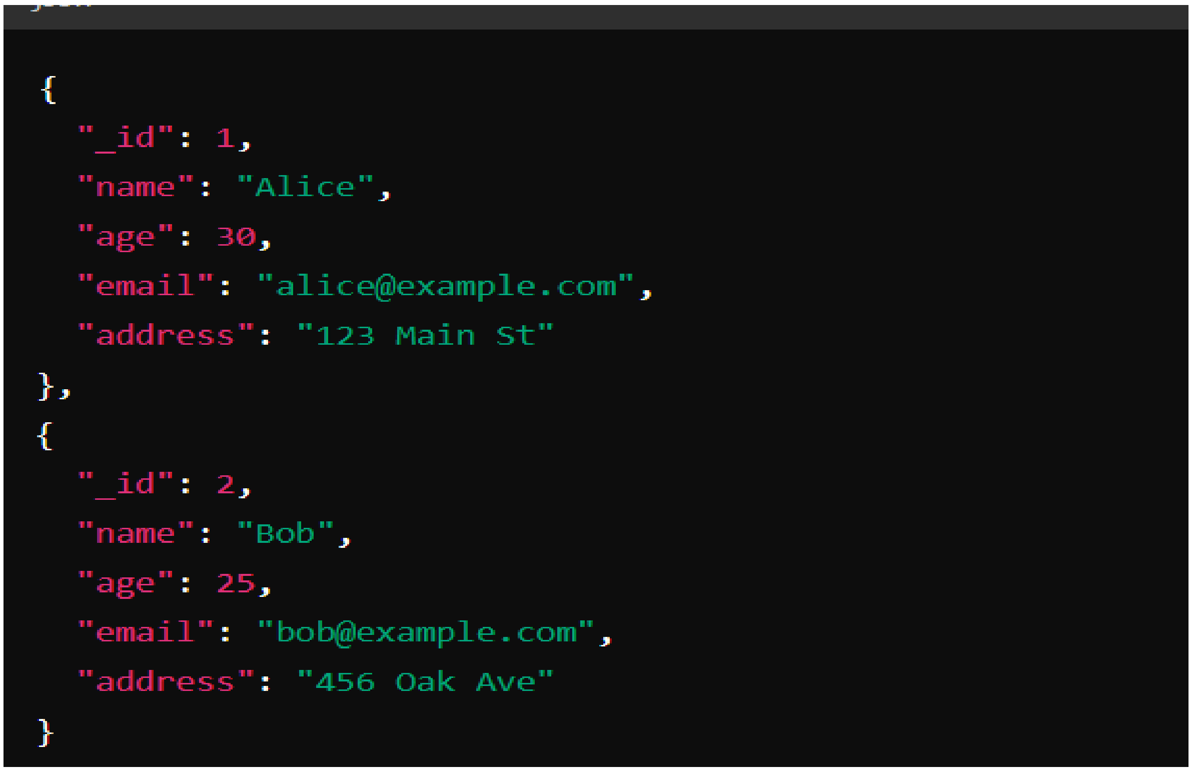
When using the aggregation framework, you can use the $project stage to include/exclude fields and add computed fields.

8.$project: Used in the aggregation pipeline to reshape each document in the stream.

```
db.collection.aggregate([
 { $project: { field1: 1, field2: 1, computedField: { $add: ["$fieldA", "$fieldB"] } } }
]);
```

An example of using projection operators in MongoDB:

Assume we have a MongoDB collection named `users` with the following documents:



```json
{
  "_id": 1,
  "name": "Alice",
  "age": 30,
  "email": "alice@example.com",
  "address": "123 Main St"
},
{
  "_id": 2,
  "name": "Bob",
  "age": 25,
  "email": "bob@example.com",
  "address": "456 Oak Ave"
}
```

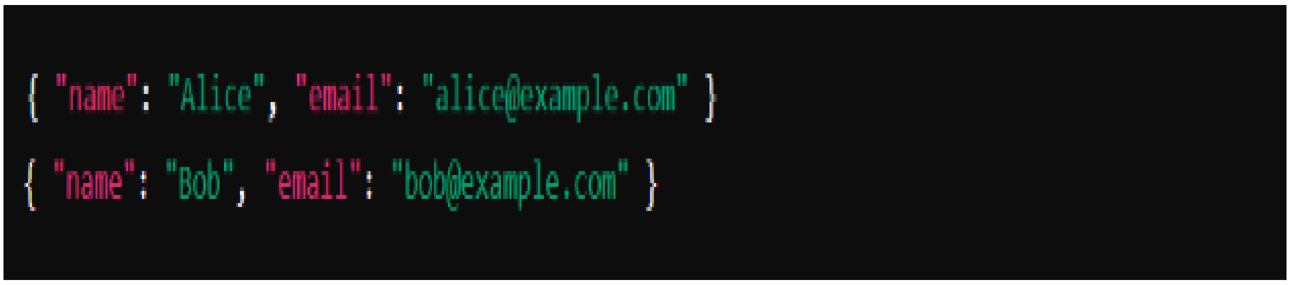
Example 1: Including Specific Fields

Let's say we want to retrieve only the `name` and `email` fields for each user, excluding the rest.

Query:

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

Result:



```
{ "name": "Alice", "email": "alice@example.com" }
{ "name": "Bob", "email": "bob@example.com" }
```

In this example:

- {} is the query part, meaning we are not filtering any documents and want to retrieve all documents.
- { name: 1, email: 1, _id: 0 } is the projection part, where:
    - name: 1 includes the name field.
    - email: 1 includes the email field.
    - _id: 0 excludes the _id field (which is included by default).

Example 2: Excluding Specific Fields

Now, let's say we want to retrieve all fields except the `address` field.

Query:

db.users.find({}, { address: 0 })

Result:

```
{
  "_id": 1,
  "name": "Alice",
  "age": 30,
  "email": "alice@example.com"
},
{
  "_id": 2,
  "name": "Bob",
  "age": 25,
  "email": "bob@example.com"
}
```

In this example:

- {} is the query part, meaning we are not filtering any documents and want to retrieve all documents.
- { address: 0 } is the projection part, where address: 0 excludes the address field.

Example 3: Using Positional Operator

Suppose we have a more complex document structure with an array of scores and we want to retrieve only the first element of the scores array.

```
{
  "_id": 1,
  "name": "Alice",
  "scores": [
    { "subject": "Math", "score": 85 },
    { "subject": "English", "score": 90 }
  ]
},
{
  "_id": 2,
  "name": "Bob",
  "scores": [
    { "subject": "Math", "score": 80 },
    { "subject": "English", "score": 88 }
  ]
}
```

Query:

```
db.users.find({}, { name: 1, "scores": { $slice: 1 }, _id: 0 })
```

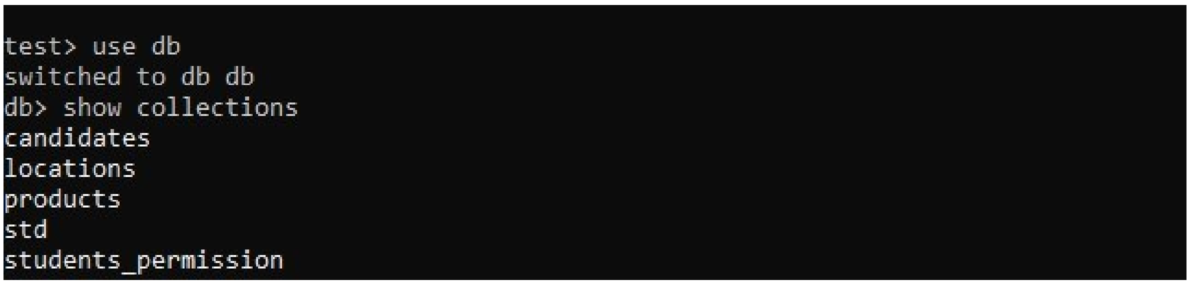
Result:

```
{
  "name": "Alice",
  "scores": [
    { "subject": "Math", "score": 85 }
  ]
},
{
  "name": "Bob",
  "scores": [
    { "subject": "Math", "score": 80 }
  ]
}
```

In this example:

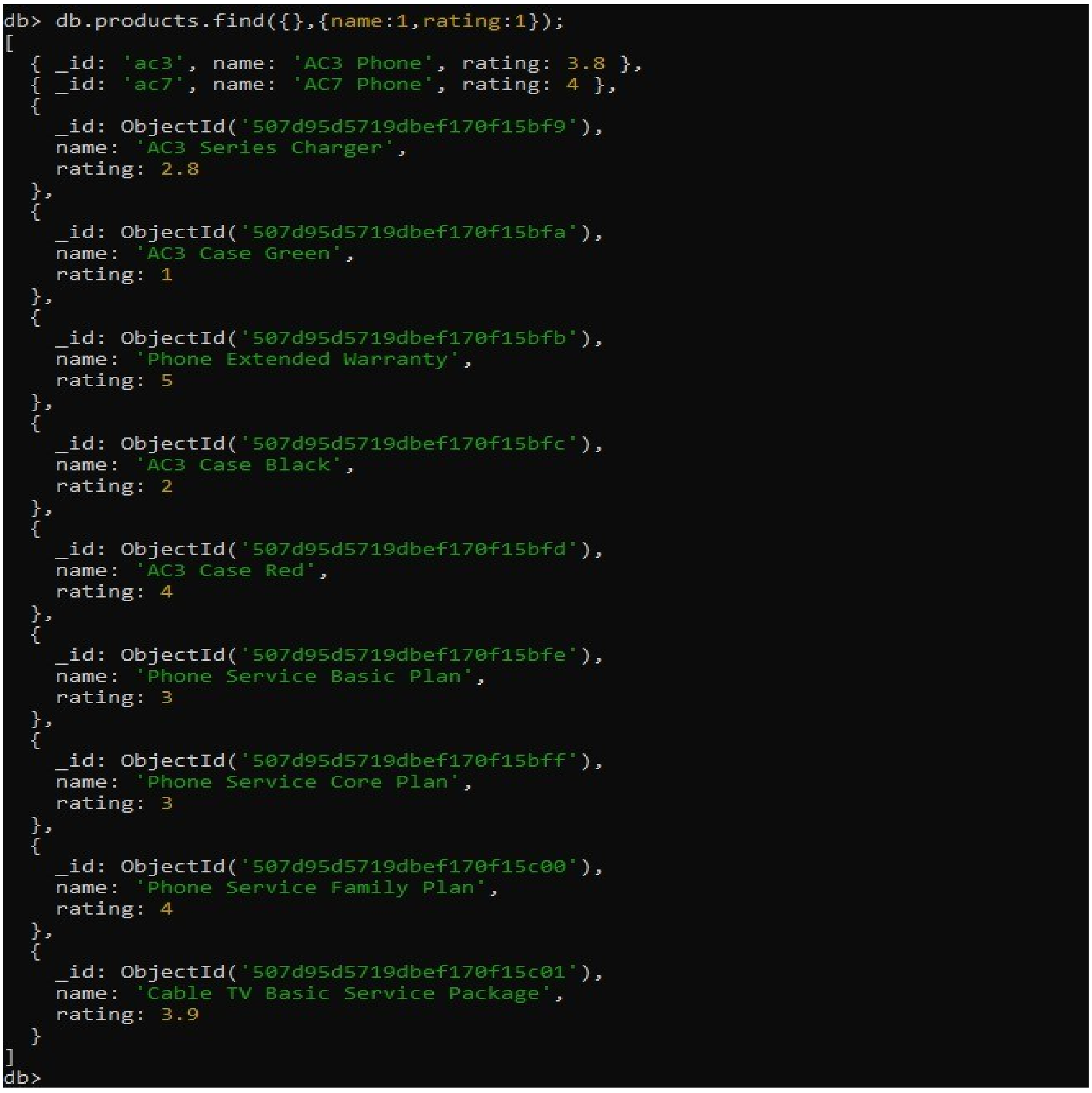
- `{ }` is the query part, meaning we are not filtering any documents and want to retrieve all documents.
- `{ name: 1, "scores": { $slice: 1 }, _id: 0 }` is the projection part, where:
  - `name: 1` includes the `name` field.
  - `"scores": { $slice: 1 }` includes only the first element of the `scores` array.
  - `_id: 0` excludes the `_id` field.

ADDING NEW DATASET CALLED PRODUCTS:



```
test> use db
switched to db db
db> show collections
candidates
locations
products
std
students_permission
```

RETRIVING NAME AND RATINGS OF THE PRODUCTS:



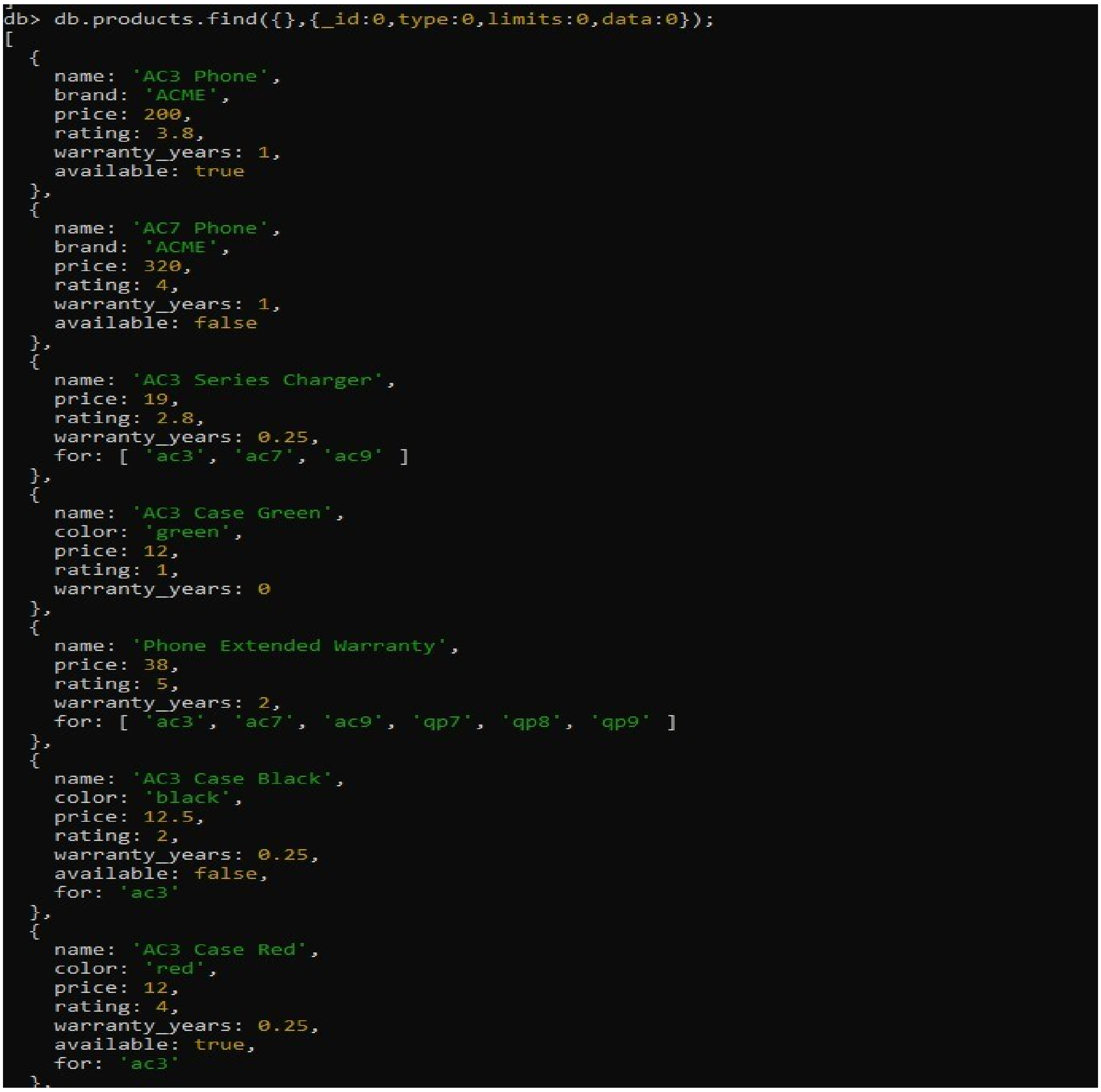
```
db> db.products.find({},{name:1,rating:1});
[
  { _id: 'ac3', name: 'AC3 Phone', rating: 3.8 },
  { _id: 'ac7', name: 'AC7 Phone', rating: 4 },
  {
    _id: ObjectId('507d95d5719dbef170f15bf9'),
    name: 'AC3 Series Charger',
    rating: 2.8
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bfa'),
    name: 'AC3 Case Green',
    rating: 1
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bfb'),
    name: 'Phone Extended Warranty',
    rating: 5
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bfc'),
    name: 'AC3 Case Black',
    rating: 2
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bfd'),
    name: 'AC3 Case Red',
    rating: 4
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bfe'),
    name: 'Phone Service Basic Plan',
    rating: 3
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bff'),
    name: 'Phone Service Core Plan',
    rating: 3
  },
  {
    _id: ObjectId('507d95d5719dbef170f15c00'),
    name: 'Phone Service Family Plan',
    rating: 4
  },
  {
    _id: ObjectId('507d95d5719dbef170f15c01'),
    name: 'Cable TV Basic Service Package',
    rating: 3.9
  }
]
db>
```

- **db**: This refers to the current database connection.
- **products:** This refers to the collection named products within the current database.
- **.find({}):** This is the find method which is used to find documents in a collection. The empty curly braces {} specify that all documents in the collection should be returned.

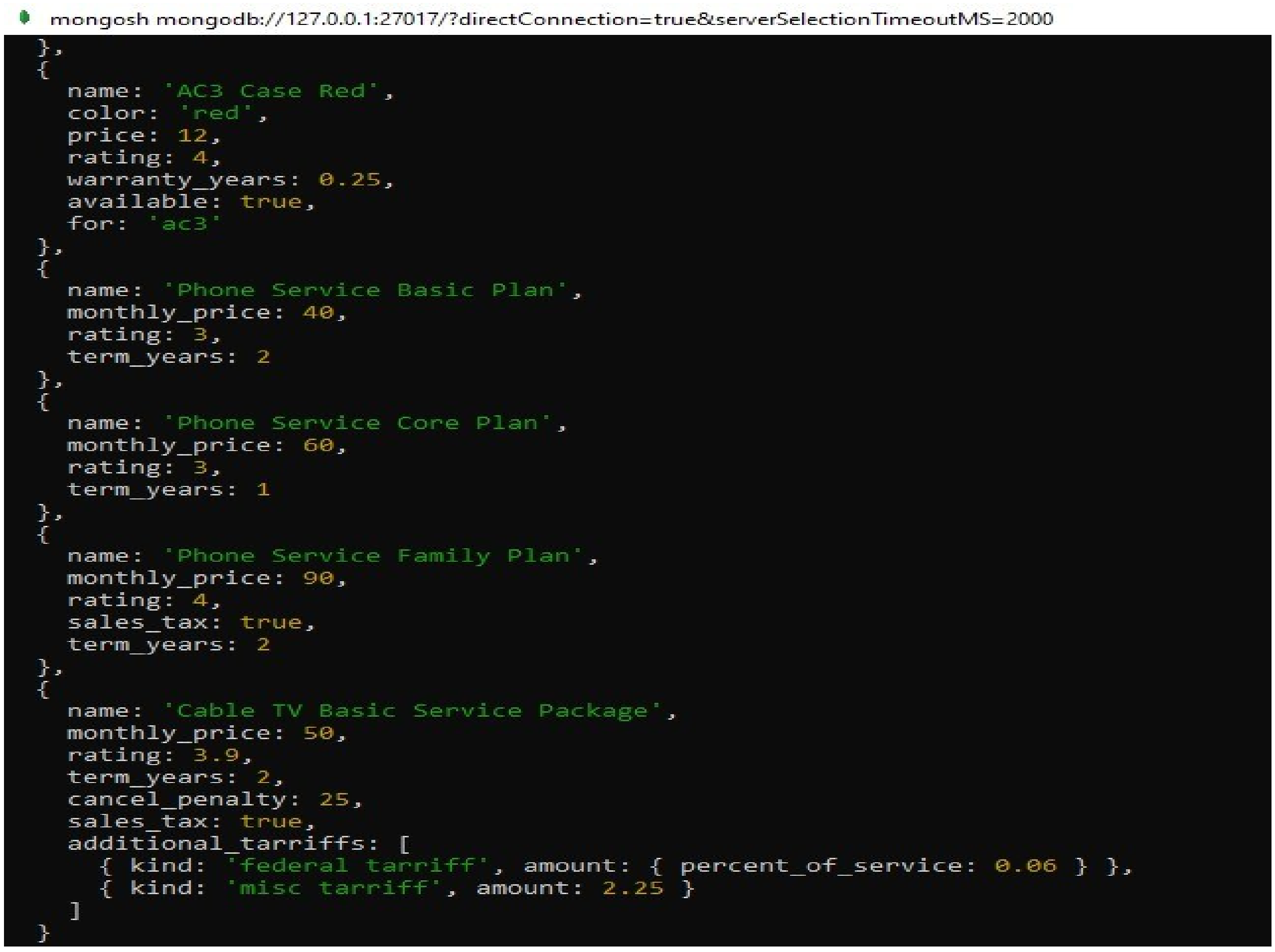
- **({name: 1, rating: 1}):** This is the projection document which specifies which fields to include in the returned documents. In this case, only the name and rating fields are included by setting their values to 1.

The output of the query shows seven documents from the products collection. Each document has an _id field, which is the unique identifier for the document, and the two fields specified in the projection, name and rating.

EXCLUDING FIELDS:



```
db> db.products.find({},{_id:0,type:0,limits:0,data:0});
[
  {
    name: 'AC3 Phone',
    brand: 'ACME',
    price: 200,
    rating: 3.8,
    warranty_years: 1,
    available: true
  },
  {
    name: 'AC7 Phone',
    brand: 'ACME',
    price: 320,
    rating: 4,
    warranty_years: 1,
    available: false
  },
  {
    name: 'AC3 Series Charger',
    price: 19,
    rating: 2.8,
    warranty_years: 0.25,
    for: [ 'ac3', 'ac7', 'ac9' ]
  },
  {
    name: 'AC3 Case Green',
    color: 'green',
    price: 12,
    rating: 1,
    warranty_years: 0
  },
  {
    name: 'Phone Extended Warranty',
    price: 38,
    rating: 5,
    warranty_years: 2,
    for: [ 'ac3', 'ac7', 'ac9', 'qp7', 'qp8', 'qp9' ]
  },
  {
    name: 'AC3 Case Black',
    color: 'black',
    price: 12.5,
    rating: 2,
    warranty_years: 0.25,
    available: false,
    for: 'ac3'
  },
  {
    name: 'AC3 Case Red',
    color: 'red',
    price: 12,
    rating: 4,
    warranty_years: 0.25,
    available: true,
    for: 'ac3'
  },
```

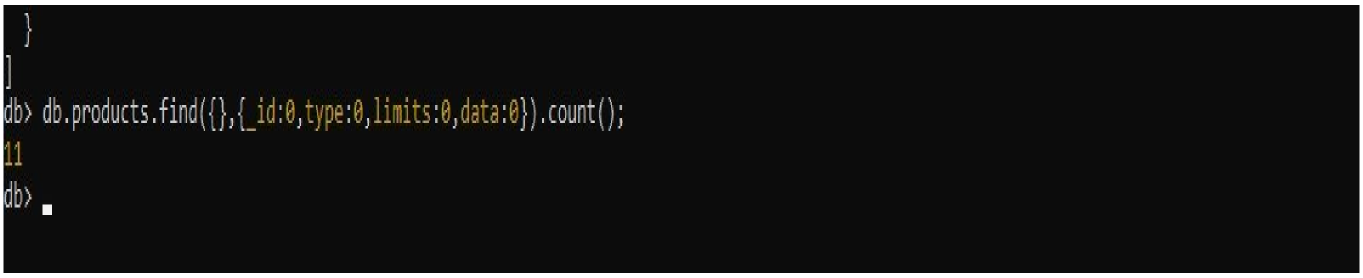


```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
  },
  {
    name: 'AC3 Case Red',
    color: 'red',
    price: 12,
    rating: 4,
    warranty_years: 0.25,
    available: true,
    for: 'ac3'
  },
  {
    name: 'Phone Service Basic Plan',
    monthly_price: 40,
    rating: 3,
    term_years: 2
  },
  {
    name: 'Phone Service Core Plan',
    monthly_price: 60,
    rating: 3,
    term_years: 1
  },
  {
    name: 'Phone Service Family Plan',
    monthly_price: 90,
    rating: 4,
    sales_tax: true,
    term_years: 2
  },
  {
    name: 'Cable TV Basic Service Package',
    monthly_price: 50,
    rating: 3.9,
    term_years: 2,
    cancel_penalty: 25,
    sales_tax: true,
    additional_tarriffs: [
      { kind: 'federal tarriff', amount: { percent_of_service: 0.06 } },
      { kind: 'misc tarriff', amount: 2.25 }
    ]
  }
```

• ({id: 0, type: 0, limits: 0, data: 0}): This is the projection document which specifies which fields to exclude from the returned documents. In this case, the following fields are excluded by setting their values to 0: id, type, limits, and data.

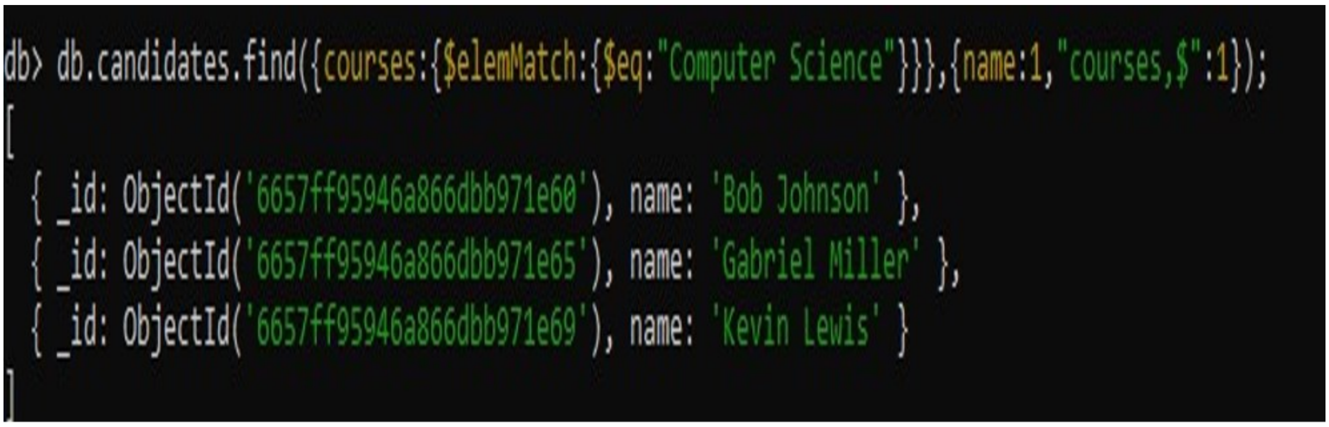
The output of the query shows seven documents from the products collection. However, it only displays specific fields for each document based on what wasn't excluded in the projection

To get total number of counts for the above operation:



```
  }
]
db> db.products.find({},{_id:0,type:0,limits:0,data:0}).count();
11
db> ▪
```

Now lets use candidates.json dataset:



```
db> db.candidates.find({courses:{$elemMatch:{$eq:"Computer Science"}}},{name:1,"courses,$":1});
[
  { _id: ObjectId('6657ff95946a866dbb971e60'), name: 'Bob Johnson' },
  { _id: ObjectId('6657ff95946a866dbb971e65'), name: 'Gabriel Miller' },
  { _id: ObjectId('6657ff95946a866dbb971e69'), name: 'Kevin Lewis' }
]
```

Each document in the output includes:

- The _id field (automatically included unless explicitly excluded).
- The name field of the candidate.
- The courses array with only the matching element "Computer Science".

The output of the query shows three documents where the "courses" field contains the value "Computer Science". The output only shows the "name" field and the matched course ("Computer Science") from the "courses" field for each document.

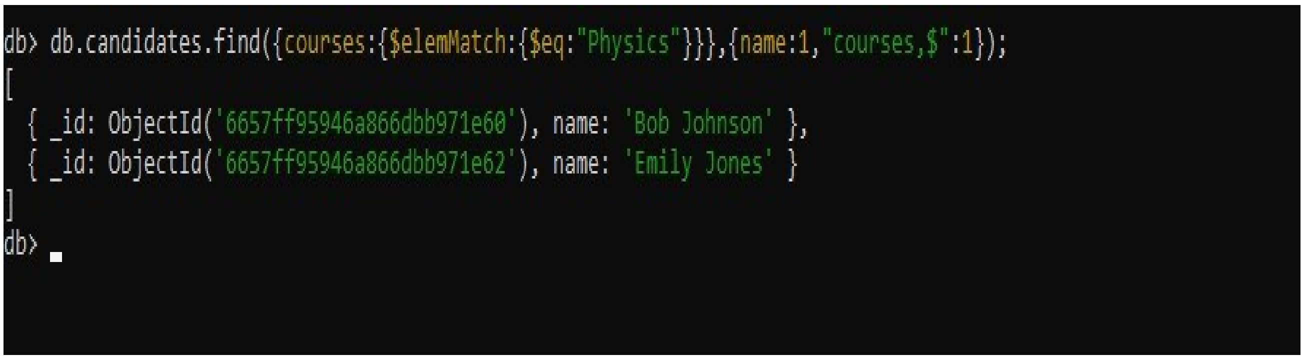
$elemMatch:

$elemMatch operator and its use in the given query. The **$elemMatch** operator is used in MongoDB to match documents that contain an array field with at least one element that matches all the specified query criteria.

Usage of $elemMatch

The $elemMatch operator is typically used in two contexts:

1. **Querying Arrays**: To find documents where at least one element in an array field matches the specified criteria.
2. **Projection**: To project the first element in an array that matches the specified criteria.



```
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.

Query Condition:

{ courses: { $elemMatch: { $eq: "Physics" } } }

- courses: This field is expected to be an array within the documents in the candidates collection.
- $elemMatch: This operator specifies that at least one element in the courses array must match the given condition.
- { $eq: "Physics" }: The condition to be matched, which means the array should contain an element that is exactly equal to "Physics".

The output of the query shows two documents where the "courses" field contains the value "Physics". The output only shows the "name" field and the matched course ("Physics") from the "courses" field for each document.

$slice:

In MongoDB, the **$slice** operator is used to retrieve a subset of elements from an array in a document. This operator can be used within a projection to limit the number of array elements returned by a query.

Syntax:

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

Usage of $slice

There are three common ways to use the $slice operator:

1.Retrieve the first n elements from an array:

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

This will return the first n elements of the array arrayField.

2.Retrieve the last n elements from an array:

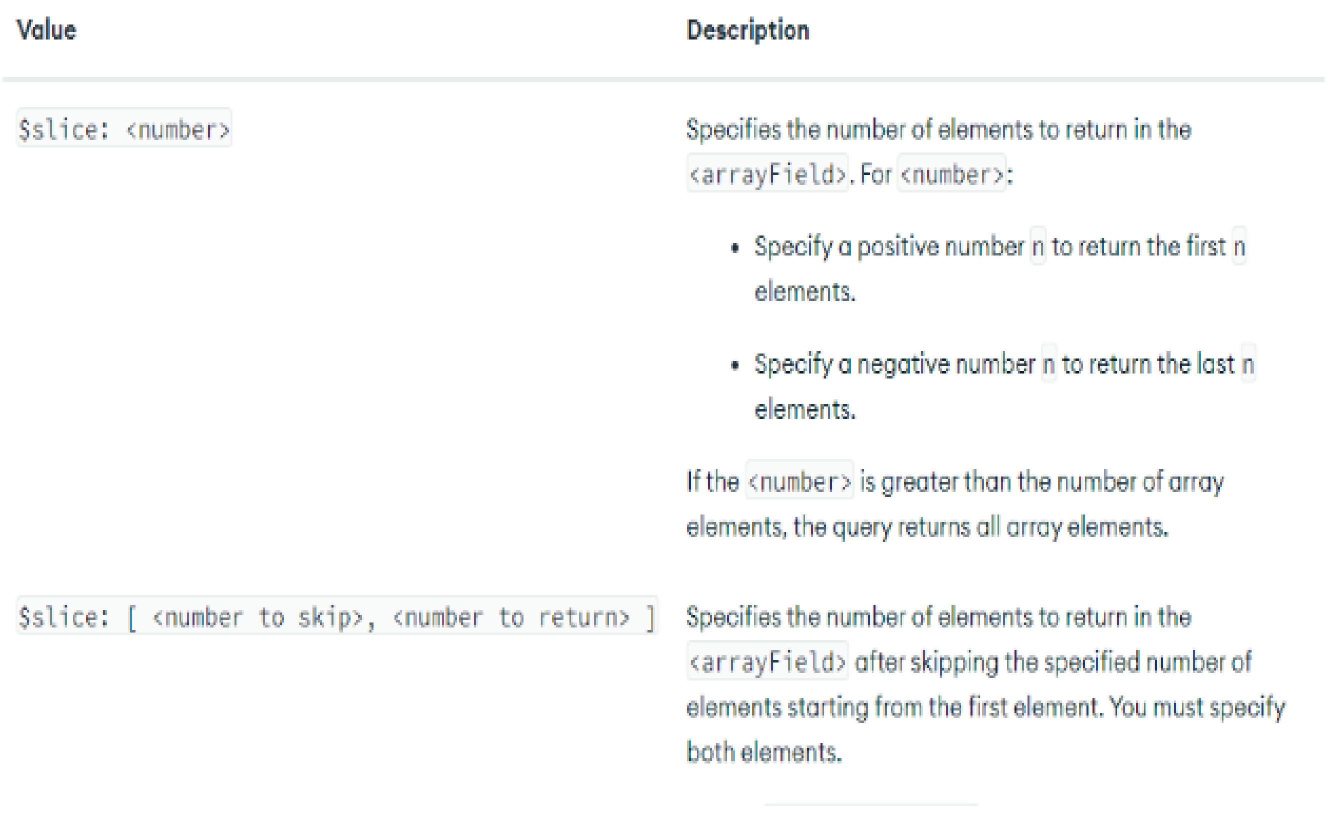
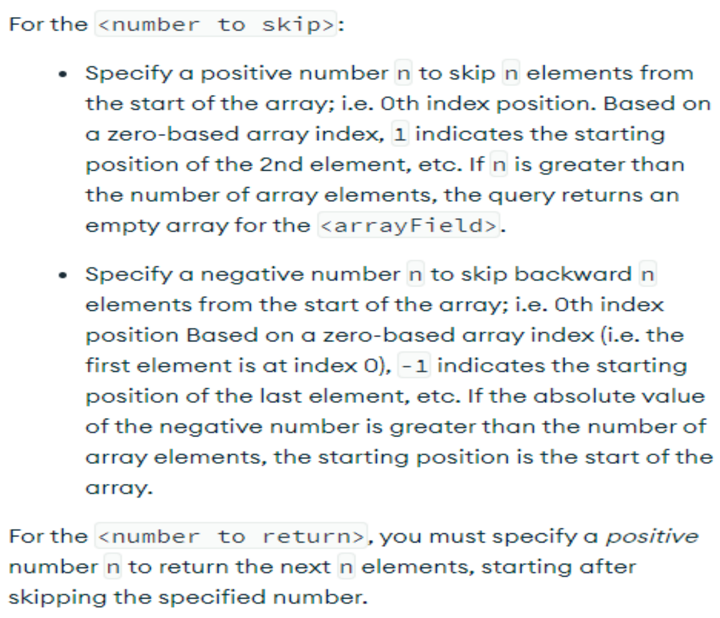
```
db.collection.find({}, { arrayField: { $slice: -n } })
```

This will return the last n elements of the array arrayField.

3.Retrieve a subset starting at a specific position:

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

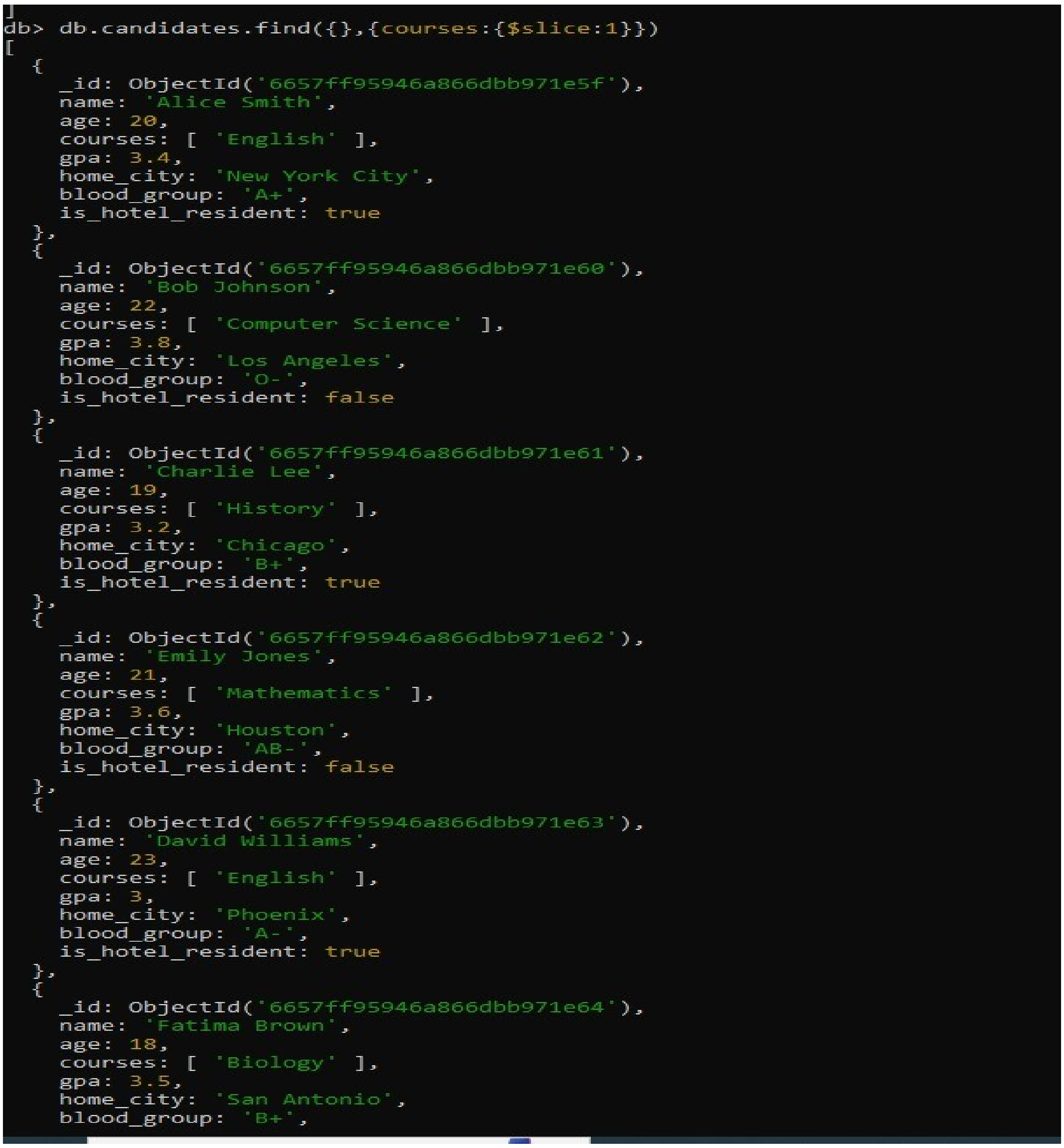
This will return length elements starting from the start position of the array arrayField.

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. 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 operation in MongoDB is a powerful tool for limiting the number of elements returned from an array within a document. It can be used in projections to retrieve a subset of an array, which is particularly useful for applications like pagination, data previews, and efficient data retrieval.



```
]
db> db.candidates.find({},{courses:{$slice:1}})
[
  {
    _id: ObjectId('6657ff95946a866dbb971e5f'),
    name: 'Alice Smith',
    age: 20,
    courses: [ 'English' ],
    gpa: 3.4,
    home_city: 'New York City',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e60'),
    name: 'Bob Johnson',
    age: 22,
    courses: [ 'Computer Science' ],
    gpa: 3.8,
    home_city: 'Los Angeles',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e61'),
    name: 'Charlie Lee',
    age: 19,
    courses: [ 'History' ],
    gpa: 3.2,
    home_city: 'Chicago',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e62'),
    name: 'Emily Jones',
    age: 21,
    courses: [ 'Mathematics' ],
    gpa: 3.6,
    home_city: 'Houston',
    blood_group: 'AB-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e63'),
    name: 'David Williams',
    age: 23,
    courses: [ 'English' ],
    gpa: 3,
    home_city: 'Phoenix',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e64'),
    name: 'Fatima Brown',
    age: 18,
    courses: [ 'Biology' ],
    gpa: 3.5,
    home_city: 'San Antonio',
    blood_group: 'B+',
db.candidates.find({}, {courses: {$slice: 1}})
```

This query does the following:

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

Explanation:

The query returns all documents in the candidates collection, but for each document, it only includes the first element 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", "Math", "Science"],
"gpa": 3.4,
"home_city": "New York City",
"blood_group": "A+",
"is_hotel_resident": true
}
```

After Slicing

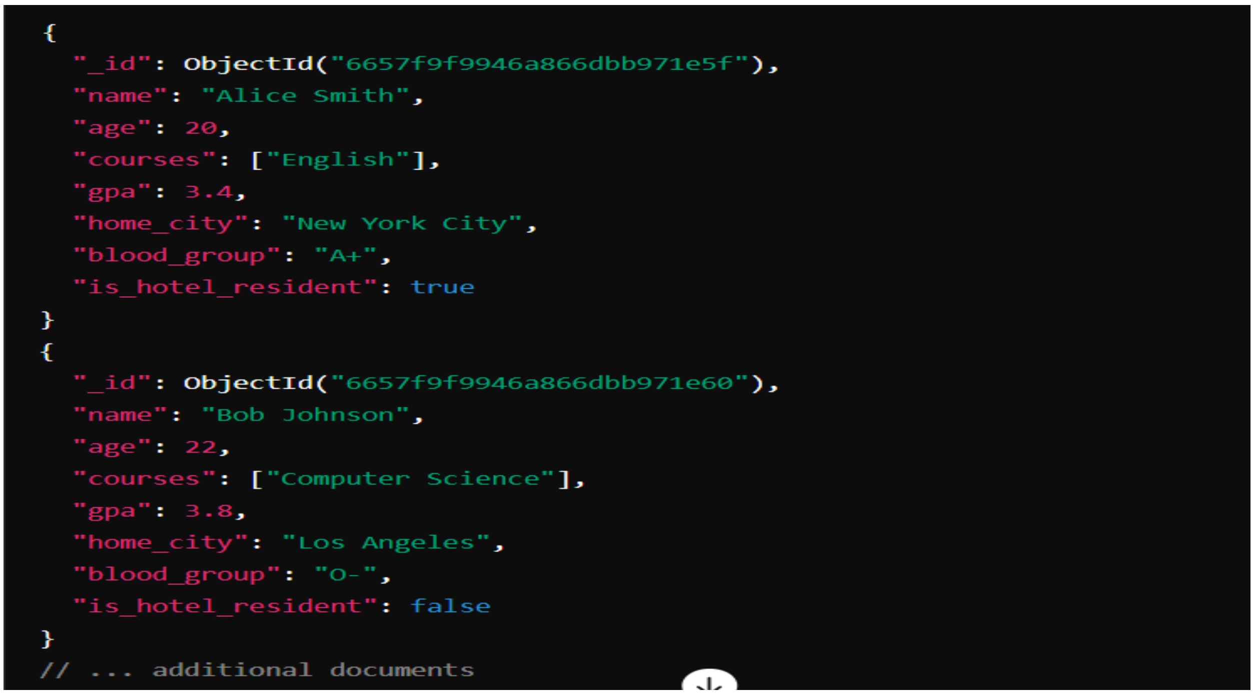
After applying the $slice: 1 projection, the document would look like:

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

Only the first element of the `courses` array ("English") is included in the output.

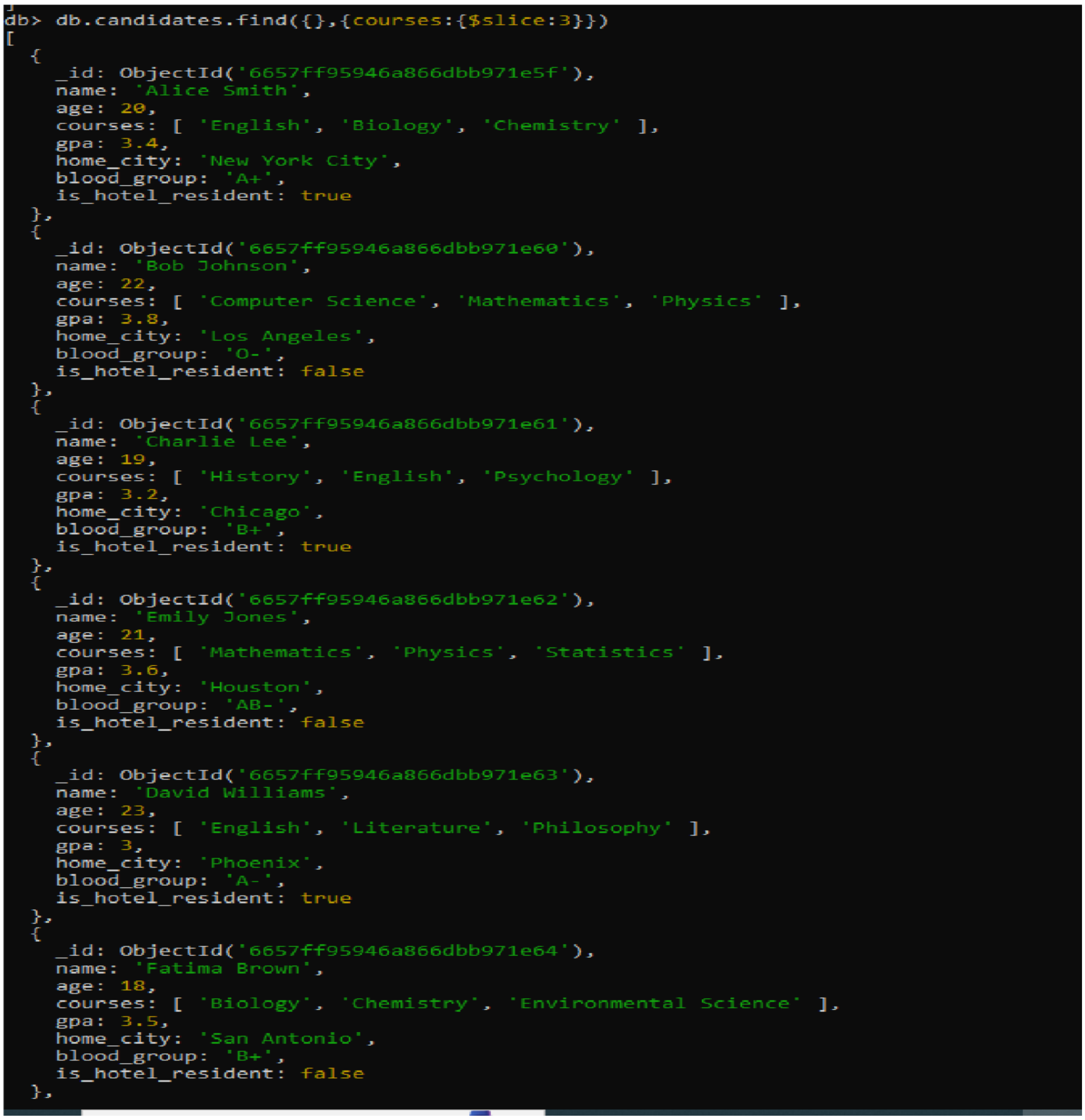
Result Set:

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



```
{
  "_id": ObjectId("6657f9f9946a866dbb971e5f"),
  "name": "Alice Smith",
  "age": 20,
  "courses": ["English"],
  "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"],
  "gpa": 3.8,
  "home_city": "Los Angeles",
  "blood_group": "O-",
  "is_hotel_resident": false
}
// ... additional documents
```

$slice operation[1:3]



```
db> db.candidates.find({},{courses:{$slice:3}})
[
  {
    _id: ObjectId('6657ff95946a866dbb971e5f'),
    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('6657ff95946a866dbb971e60'),
    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('6657ff95946a866dbb971e61'),
    name: 'Charlie Lee',
    age: 19,
    courses: [ 'History', 'English', 'Psychology' ],
    gpa: 3.2,
    home_city: 'Chicago',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e62'),
    name: 'Emily Jones',
    age: 21,
    courses: [ 'Mathematics', 'Physics', 'Statistics' ],
    gpa: 3.6,
    home_city: 'Houston',
    blood_group: 'AB-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e63'),
    name: 'David Williams',
    age: 23,
    courses: [ 'English', 'Literature', 'Philosophy' ],
    gpa: 3,
    home_city: 'Phoenix',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e64'),
    name: 'Fatima Brown',
    age: 18,
    courses: [ 'Biology', 'Chemistry', 'Environmental Science' ],
    gpa: 3.5,
    home_city: 'San Antonio',
    blood_group: 'B+',
    is_hotel_resident: false
  },
```

- db.candidates.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:

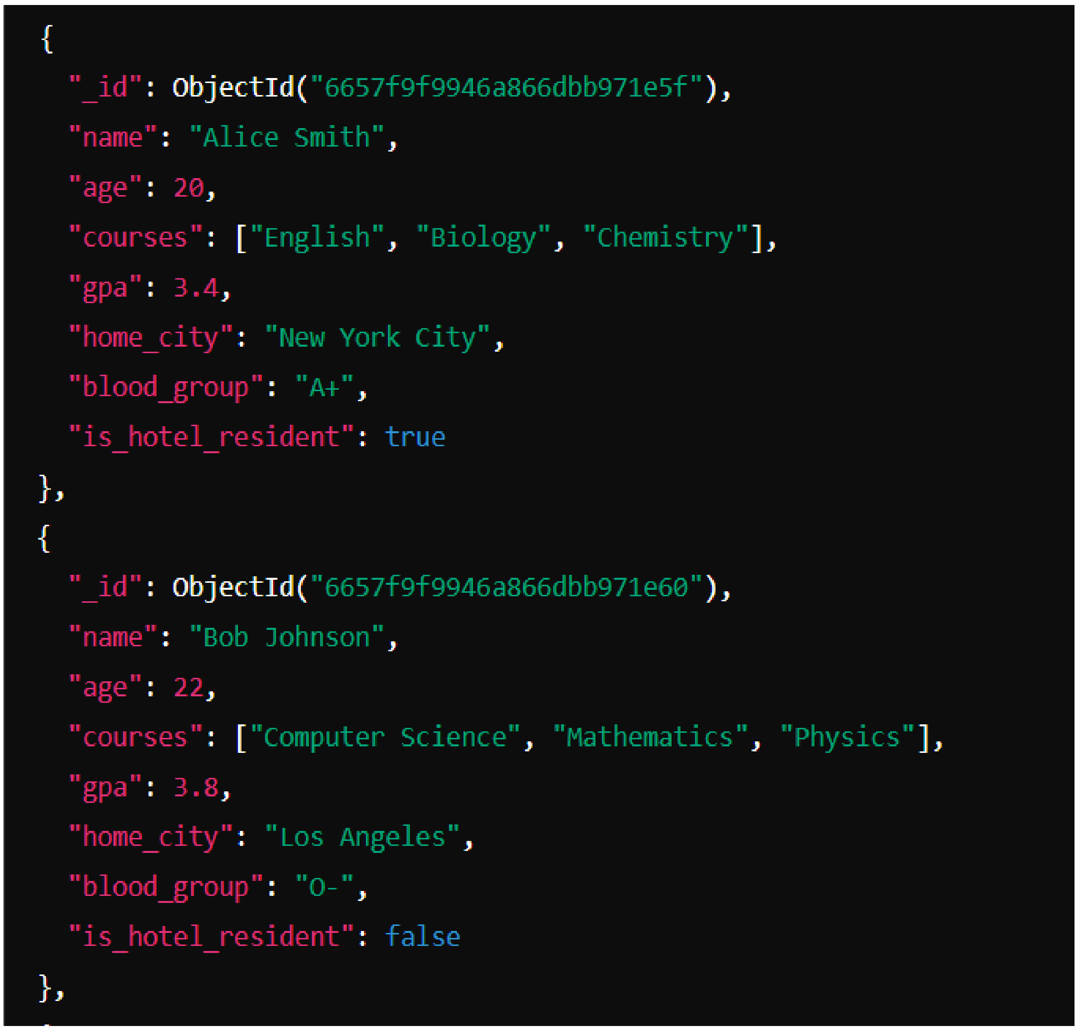
After applying the $slice: 3 projection, the document would look like:

```
             {
        "_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
}
```

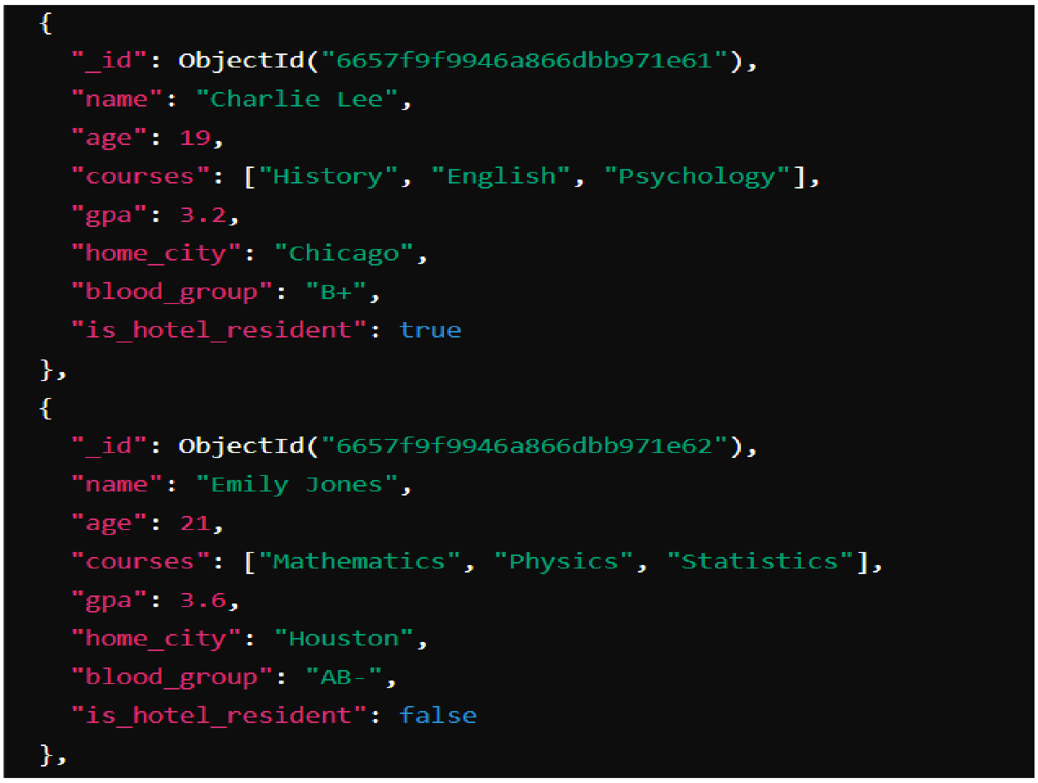
Only the first 3 elements of the `courses` array are included in the output.

Result:

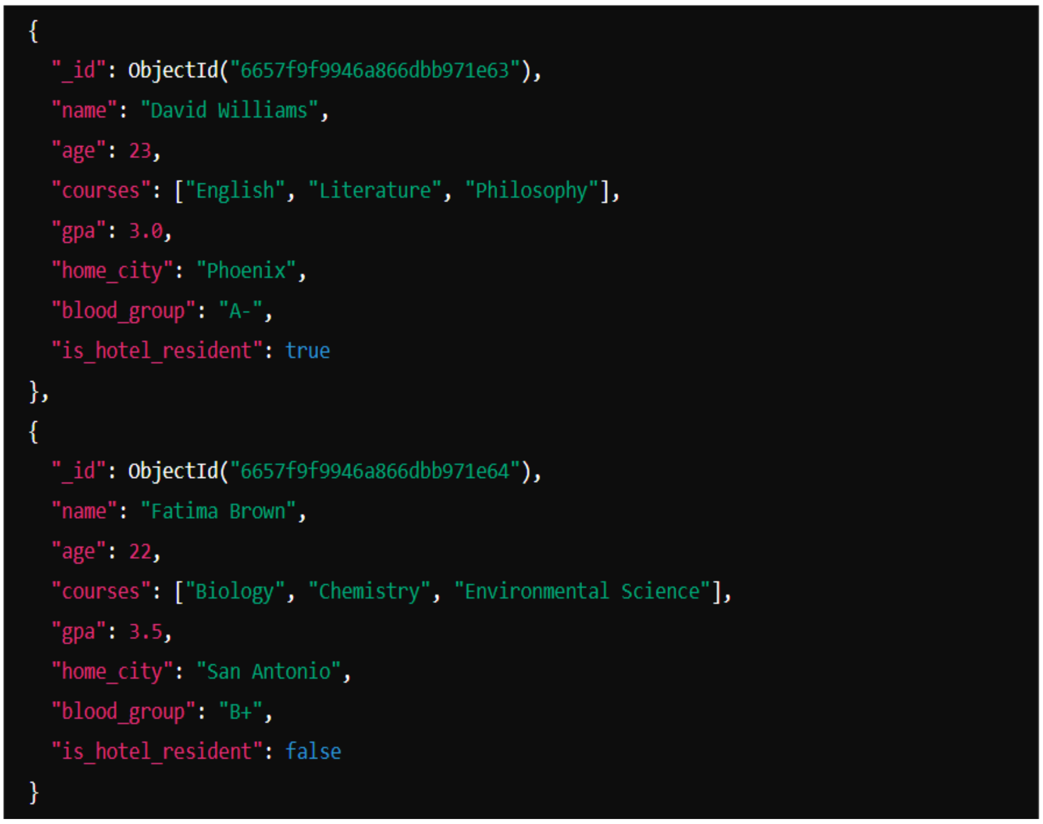
The results in the image show several documents from the candidates collection after applying the $slice operation. Each document includes only the first 3 elements of the courses array:



```
{
  "_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.