Projection Operations

Projection:

MongoDB provides a special feature that is known as **Projection**. It allows you to select only the necessary data rather than selecting whole data from the document. For example, a document contains 5 fields, i.e.,

```
{
name: "Roma",
age: 30,
branch: EEE,
department: "HR",
salary: 20000
}
```

But we only want to display the *name* and the *age* of the employee rather than displaying whole details. Now, here we use projection to display the name and age of the employee. One can use projection with `db.collection.find()` method. In this method, the second parameter is the projection parameter, which is used to specify which fields are returned in the matching documents.

Syntax:

```
db.collection.find({}, {field1: value2, field2: value2, ..})
```

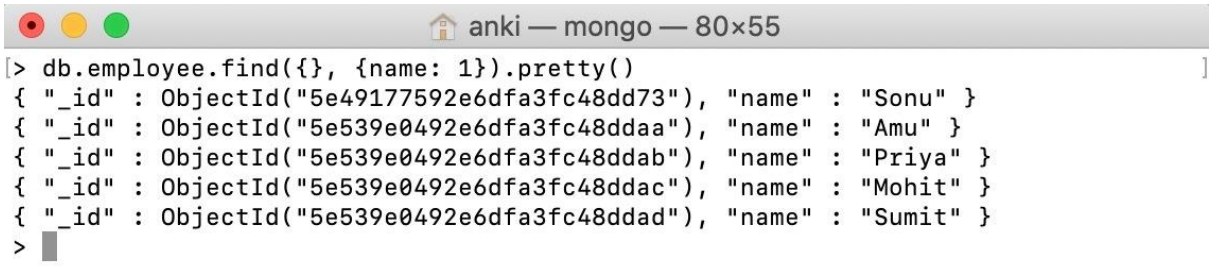
- If the value of the field is set to 1 or true, then it means the field will include in the return document.
- If the value of the field is set to 0 or false, then it means the field will not include in the return document.
- You are allowed to use projection operators, but `find()` method does not support following projection operators, i.e., `$`, `$elemMatch`, `$slice`, and `$meta`.
- There is no need to set `_id` field to 1 to return `_id` field, the `find()` method always return `_id` unless you set a `_id` field to 0.

Examples:

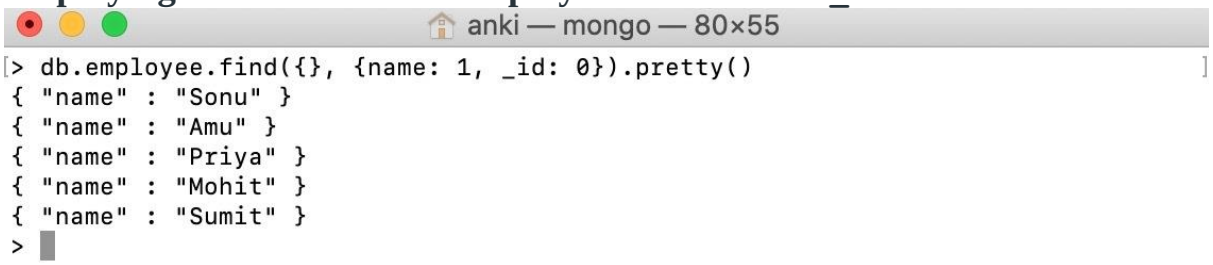
In the following examples, we are working with:

```
Database: GeeksforGeeks
Collection: employee
Document: five documents that contain the details of the employees
in the form of field-value pairs.
```

Displaying the names of the employees –



Displaying the names of the employees without the `_id` field –



Displaying the name and the department of the employees without the `_id` field –

Displaying the names and the department of the employees whose joining year is 2018 –

```
anki — mongo — 80×55
> db.employee.find({joiningYear: 2018}, {name: 1,department: 1, _id: 0}).pretty()
)
{ "name" : "Sonu", "department" : "HR" }
{ "name" : "Mohit", "department" : "Development" }
>
```

Summer-time is here and so is the time to skill-up! More than 5,000 learners have now completed their journey from **basics of DSA to advanced level development programs** such as Full-Stack, Backend Development, Data Science.

```
_id: ObjectId('665752830959f4120ac93d06')
name : "Emily Jones"
age : 21
▸ courses : Array (3)
gpa : 3.6
home_city : "Houston"
blood_group : "AB-"
is_hotel_resident : false
```

Example 1:Retrieve name,age and cgpa

Input:

Example 1: Retrieve Name, Age, and GPA

JavaScript

```
db.candidates.find({}, { name: 1, age: 1, gpa: 1 });
```

Output:

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
local    72.00 KiB
test> use db
switched to db db
db> show collections
candidates
```

Output explanation:

db.students.find({},{_id: 0}): This line uses the find method to retrieve documents from the "students" collection.

The first curly braces {} specify the query criteria. In this case, it selects all documents because it's empty.

The second curly braces {_id: 0} specify which fields to exclude from the output. The 0 here indicates that the "_id" field should not be included.

.limit(5): This limits the number of documents returned to 5.

The output shows five student documents, each with an "_id" field (which is hidden by the query) and other fields like "name", "age", and "permissions".

Here are some additional points to note:

MongoDB uses documents to store data. Documents are similar to JSON objects with key-value pairs.

The "_id" field is a special field in MongoDB that uniquely identifies each document in a collection. It's automatically generated by MongoDB when a document is inserted.

The find method is a versatile method for retrieving data from MongoDB collections. You can use it with various query filters and options to target specific documents.

JavaScript

db.candidates.find({},{name: 1, age: 1, gpa: 1})

Use code with caution

content_copy

db.candidates.find({}): This part uses the find method to retrieve documents from the "candidates" collection. The empty curly braces {} specify that all documents in the collection should be returned.

{name: 1, age: 1, gpa: 1}: This part projects which fields to include in the output documents. Specifying 1 for each field like "name", "age", and "gpa" instructs the query to include only those fields and exclude any other fields from the output.

The output shows multiple student documents, each containing only the "name", "age", and "gpa" fields.

Here are some additional points to note:

MongoDB collections store data in JSON-like documents.

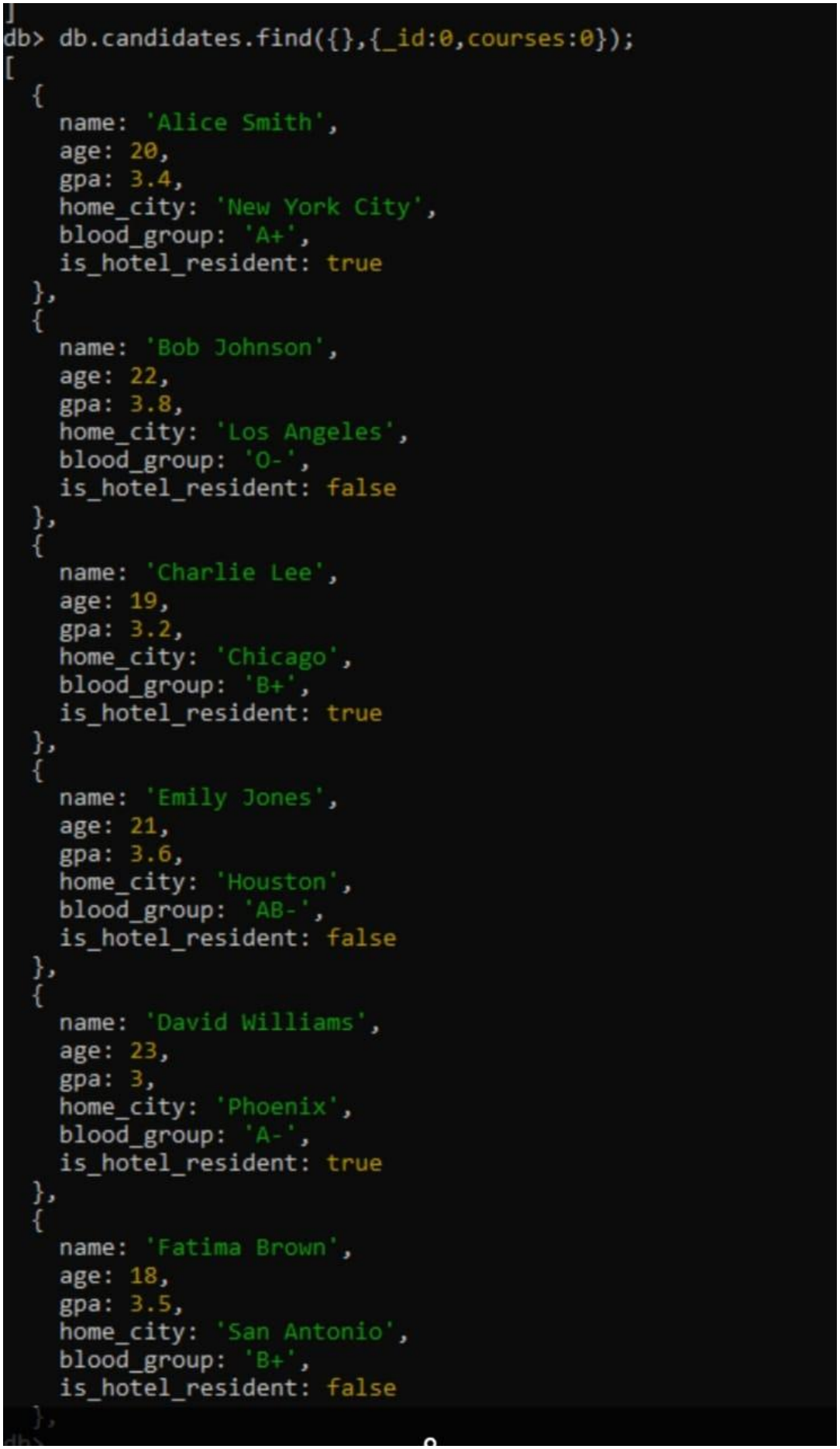
The find method is a versatile method for retrieving data from MongoDB collections. You can use it with various query filters and options to target specific documents.

You can use projection to control which fields are returned in the query results. In this case, the projection ensures that only the specified fields are included in the output, reducing the amount of data returned and improving performance.

Example 2:

Input:

Output:



```
]
db> db.candidates.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
  },
  {
    name: 'David Williams',
    age: 23,
    gpa: 3,
    home_city: 'Phoenix',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Fatima Brown',
    age: 18,
    gpa: 3.5,
    home_city: 'San Antonio',
    blood_group: 'B+',
    is_hotel_resident: false
  },
```

Output explanation:

JavaScript

db.candidates.find({},{name: 1, age: 1, gpa: 1})

Use code with caution

content_copy

db.candidates.find({}): This part uses the find method to retrieve documents from the "candidates" collection. The empty curly braces {} specify that all documents in the collection should be returned

{name: 1, age: 1, gpa: 1}: This part projects which fields to include in the output documents. Specifying 1 for each field like "name", "age", and "gpa" instructs the query to include only those fields and exclude any other fields from the output.

The output shows multiple student documents, each containing only the "name", "age", and "gpa" fields.

Here are some additional points to note:

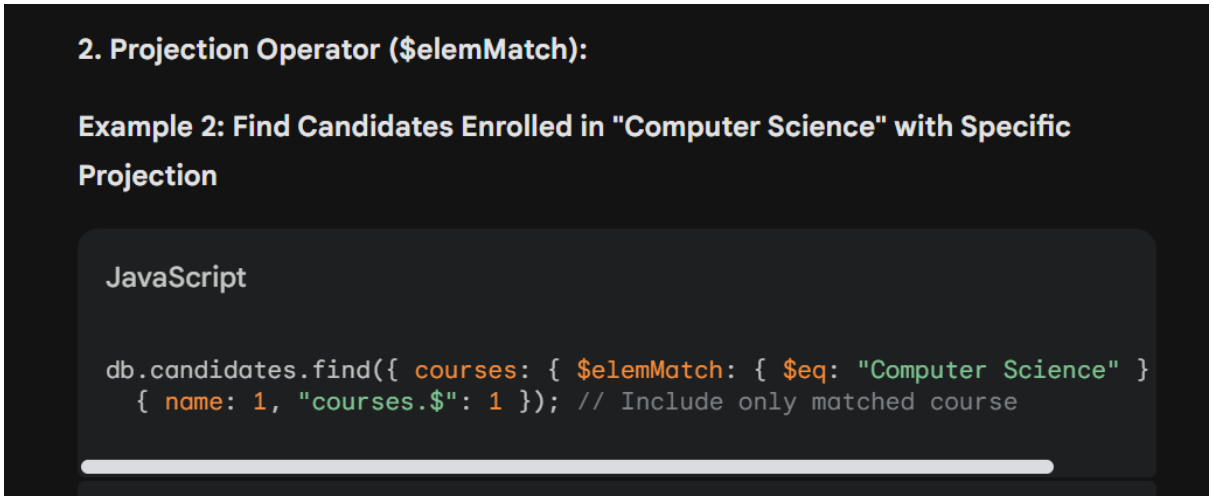
MongoDB collections store data in JSON-like documents

The find method is a versatile method for retrieving data from MongoDB collections. You can use it with various query filters and options to target specific documents

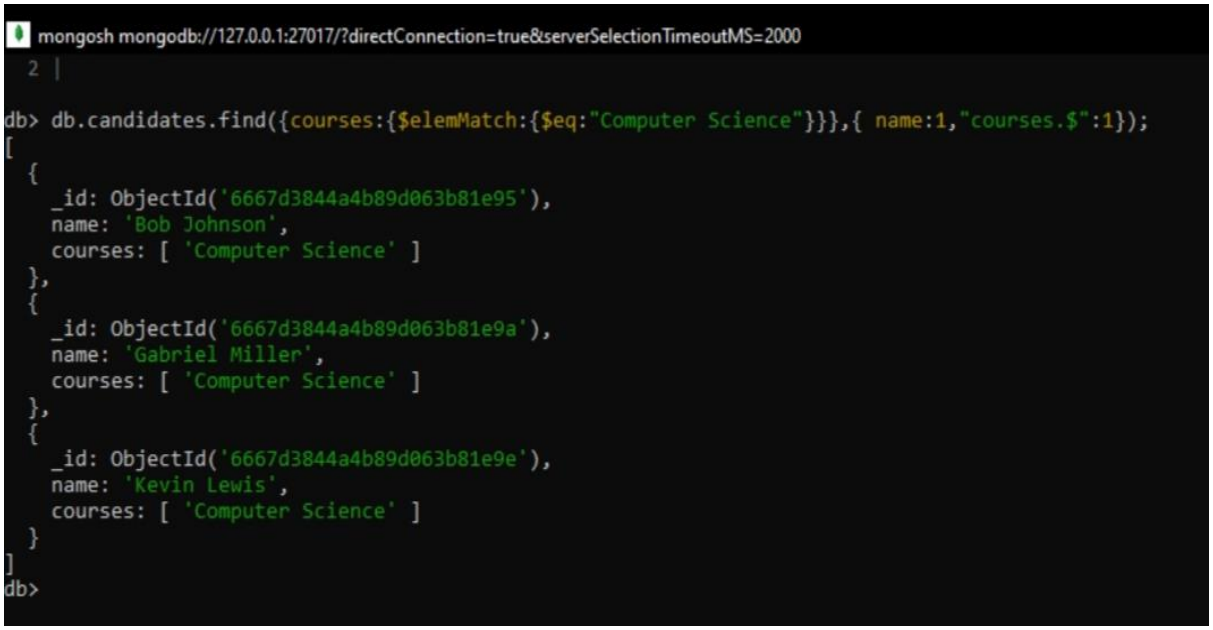
You can use projection to control which fields are returned in the query results. In this case, the projection ensures that only the specified fields are included in the output, reducing the amount of data returned and improving performance.

Example 3:

Input:



Output:



Output explanation:

db.candidates.find({},{name: 1, age: 1, gpa: 1})

Use code with caution.

content_cop

db.candidates.find({}): This part uses the find method to retrieve documents from the "candidates" collection. The empty curly braces {} specify that all documents in the collection

{name: 1, age: 1, gpa: 1}: This part projects which fields to include in the output documents. Specifying 1 for each field like "name", "age", and "gpa" instructs the query to include only those fields and exclude any other fields from the output.

The output shows multiple student documents, each containing only the "name", "age", and "gpa" fields.

Here are some additional points to note

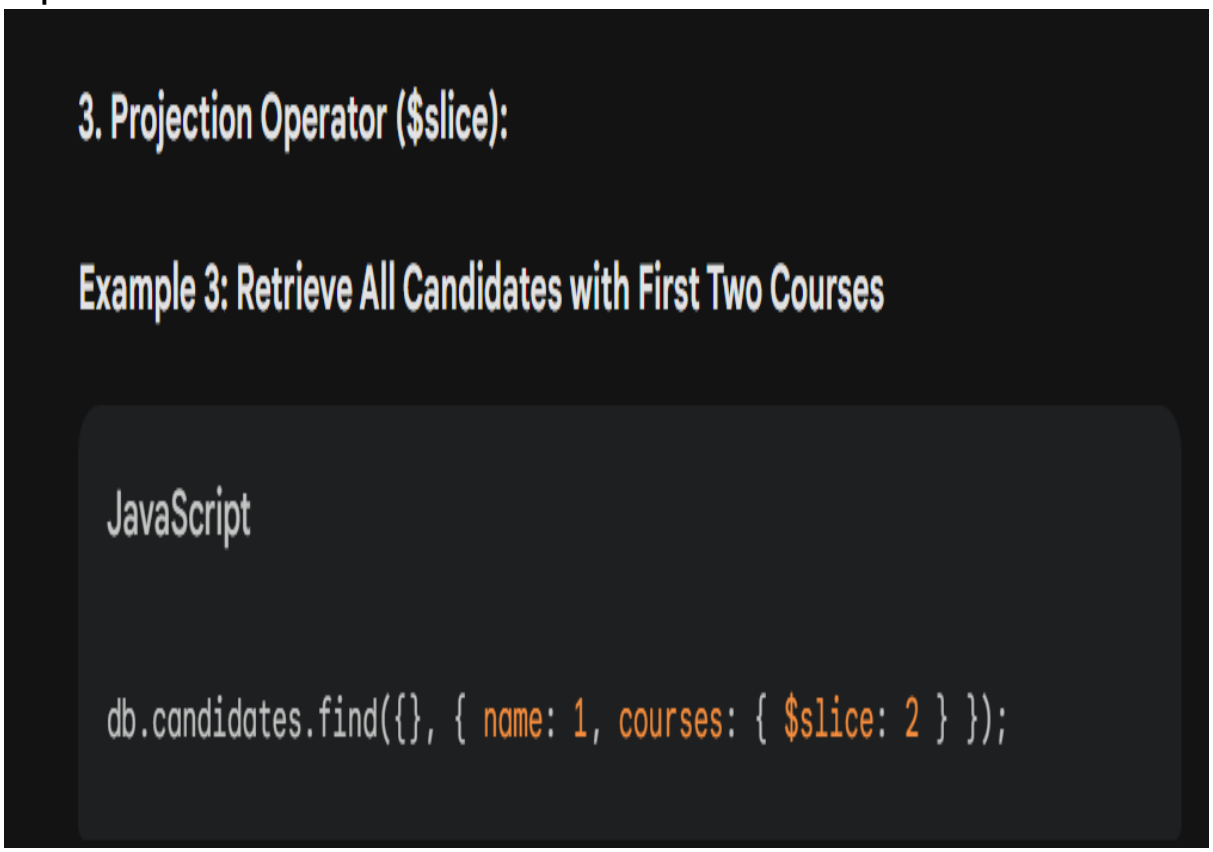
MongoDB collections store data in JSON-like documents.

The find method is a versatile method for retrieving data from MongoDB collections. You can use it with various query filters and options to target specific documents.

You can use projection to control which fields are returned in the query results. In this case, the projection ensures that only the specified fields are included in the output, reducing the amount of data returned and improving performance.

Projection operator ($slice):

Input:



Output:

```
b> db.candidates.find({},{name:1,courses:{$slice:2}});

[
  {
    _id: ObjectId('6667d3844a4b89d063b81e94'),
    name: 'Alice Smith',
    courses: [ 'English', 'Biology' ]
  },
  {
    _id: ObjectId('6667d3844a4b89d063b81e95'),
    name: 'Bob Johnson',
    courses: [ 'Computer Science', 'Mathematics' ]
  },
  {
    _id: ObjectId('6667d3844a4b89d063b81e96'),
    name: 'Charlie Lee',
    courses: [ 'History', 'English' ]
  },
  {
    _id: ObjectId('6667d3844a4b89d063b81e97'),
    name: 'Emily Jones',
    courses: [ 'Mathematics', 'Physics' ]
  },
  {
    _id: ObjectId('6667d3844a4b89d063b81e98'),
    name: 'David Williams',
    courses: [ 'English', 'Literature' ]
  },
  {
    _id: ObjectId('6667d3844a4b89d063b81e99'),
    name: 'Fatima Brown',
    courses: [ 'Biology', 'Chemistry' ]
  },
  {
    _id: ObjectId('6667d3844a4b89d063b81e9a'),
    name: 'Gabriel Miller',
    courses: [ 'Computer Science', 'Engineering' ]
  },
  {
    _id: ObjectId('6667d3844a4b89d063b81e9b'),
    name: 'Hannah Garcia',
    courses: [ 'History', 'Political Science' ]
  },
  {
    _id: ObjectId('6667d3844a4b89d063b81e9c'),
    name: 'Isaac Clark',
    courses: [ 'English', 'Creative Writing' ]
  },
  {
    _id: ObjectId('6667d3844a4b89d063b81e9d'),
    name: 'Jessica Moore',
    courses: [ 'Biology', 'Ecology' ]
b>
  {
    _id: ObjectId('6667d3844a4b89d063b81e9e'),
    name: 'Kevin Lewis',
    courses: [ 'Computer Science', 'Artificial Intelligence' ]
  },
  {
    _id: ObjectId('6667d3844a4b89d063b81e9f'),
    name: 'Lily Robinson',
    courses: [ 'History', 'Art History' ]
```

Output explanation:

db.students.find({}): This part uses the find method to retrieve documents from the "students" collection.

{age:{$gt:20}}: This part filters the documents based on the value of the "age" field. The $gt operator is a comparison operator that checks if a field is greater than a specified value. In this case, the query selects documents where the "age" field is greater than 20.

The output shows multiple student documents where the "age" field is greater than 20. Each document contains all fields stored for that student.

Here are some additional points to note:

MongoDB collections store data in JSON-like documents.

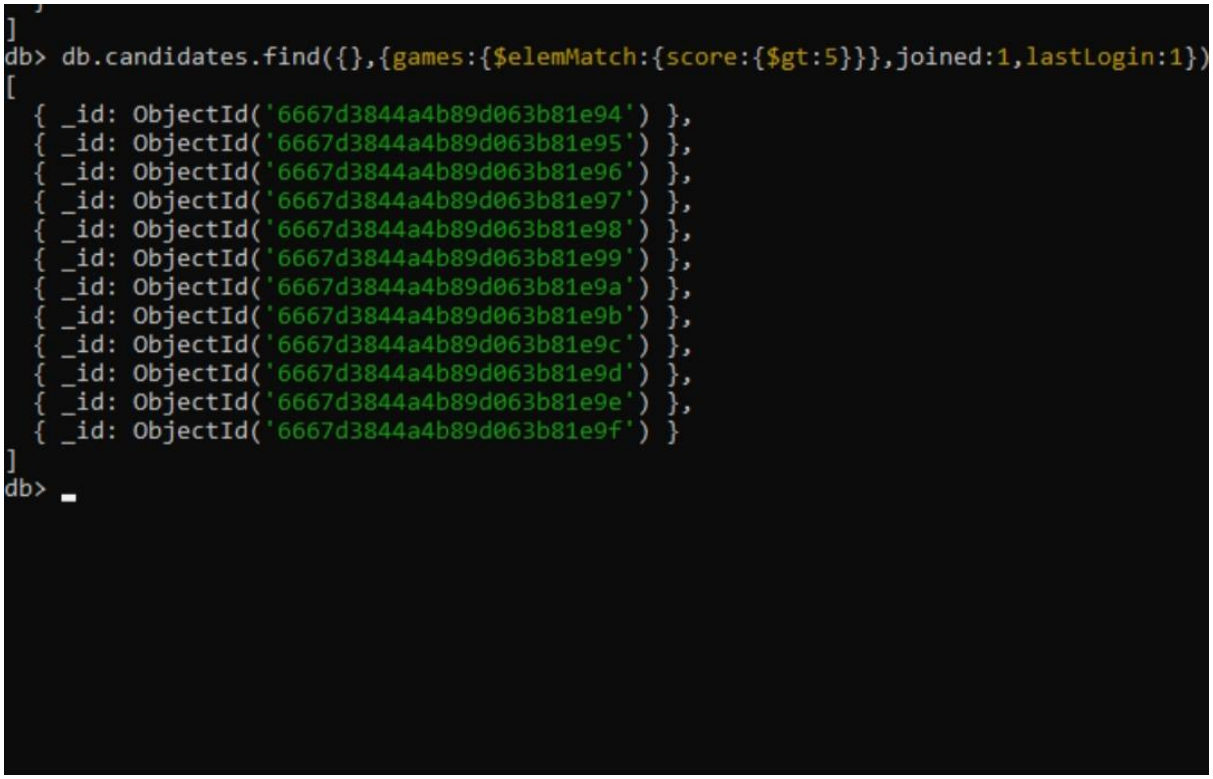
The find method is a versatile method for retrieving data from MongoDB collections. You can use it with various query filters and options to target specific documents.

The $gt operator is one of many comparison operators available in MongoDB. You can use it to filter documents based on equality, greater than, less than, and other comparisons.

$elemMatch:

Input:

Output:



output explanation:

db.candidates.find({}): This part uses the find method to retrieve documents from the "candidates" collection. The empty curly braces {} specify that all documents in the collection should be returned.

{name: 1, age: 1, gpa: 1}: This part projects which fields to include in the output documents. Specifying 1 for each field like "name", "age", and "gpa" instructs the query to include only those fields and exclude any other fields from the output.

The output shows multiple student documents, each containing only the "name", "age", and "gpa" fields.

Here are some additional points to note:

MongoDB collections store data in JSON-like documents.

The find method is a versatile method for retrieving data from MongoDB collections. You can use it with various query filters and options to target specific documents.

You can use projection to control which fields are returned in the query results. In this case, the projection ensures that only the specified fields are included in the output, reducing the amount of data returned and improving performance.