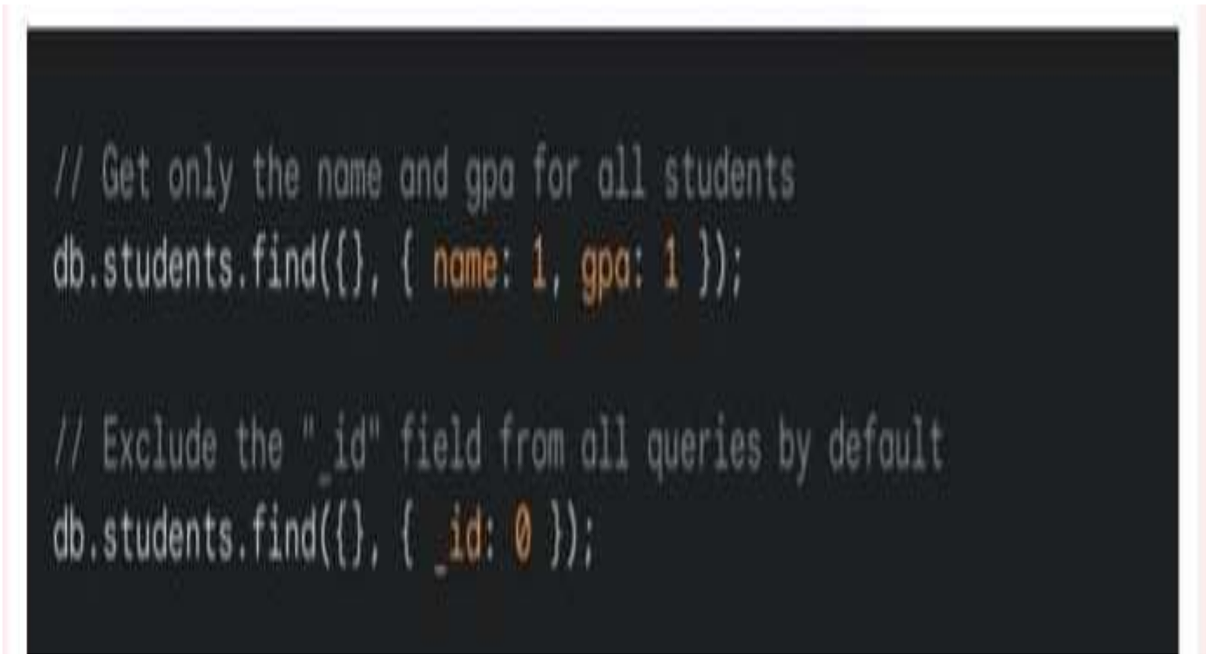
PROJECTION,LIMIT &SELECTORS

IT IS USED TO WHEN YOU DON'T WANT TO DISPLAY ALL THE ATTRIBUTES IN THE OUTPUT.

db.students.find({},{name:1,gpa:1}).count();

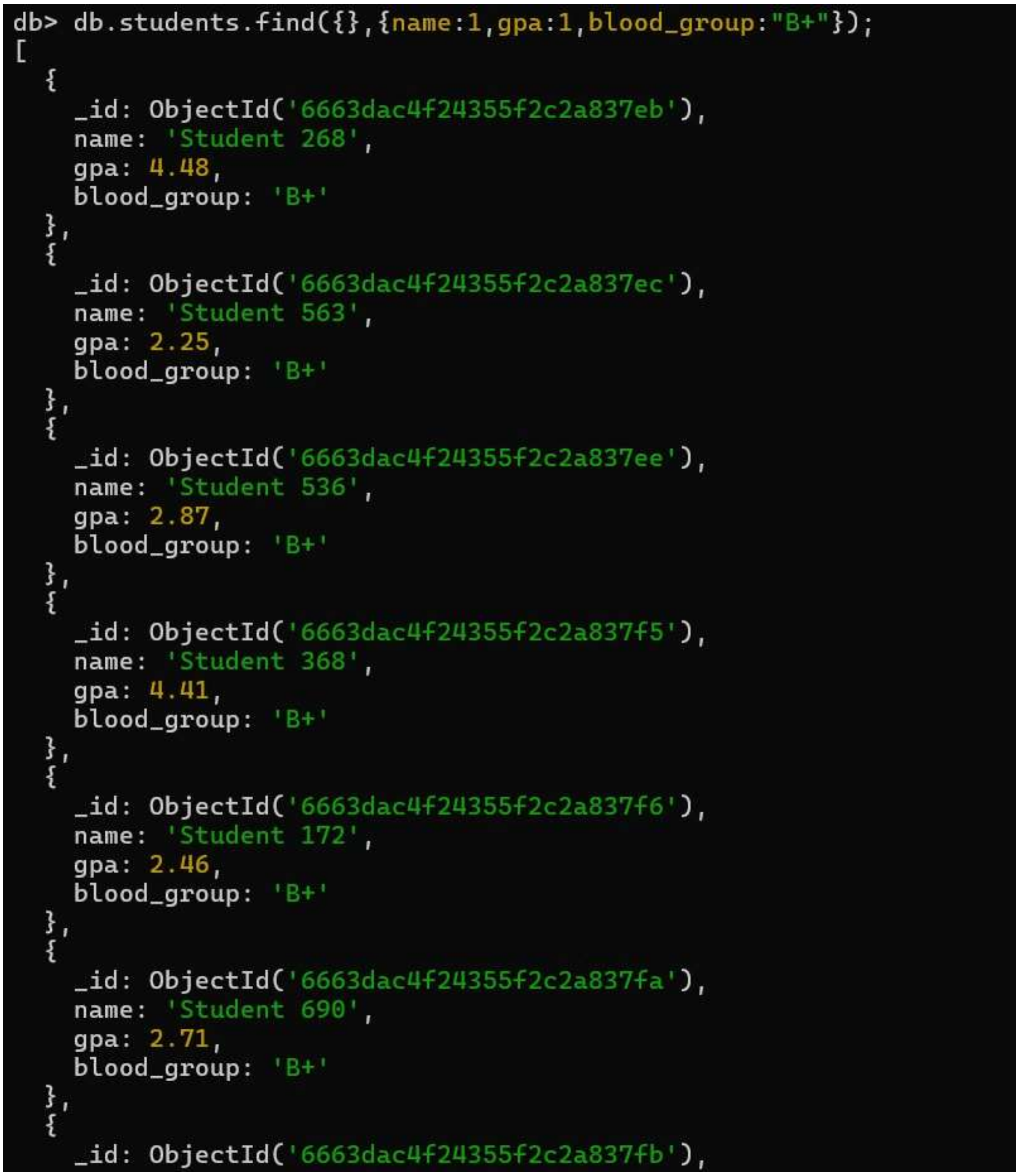


```
// Get only the name and gpa for all students
db.students.find({}, { name: 1, gpa: 1 });

// Exclude the "_id" field from all queries by default
db.students.find({}, { _id: 0 });
```

<u>To find countings</u>:

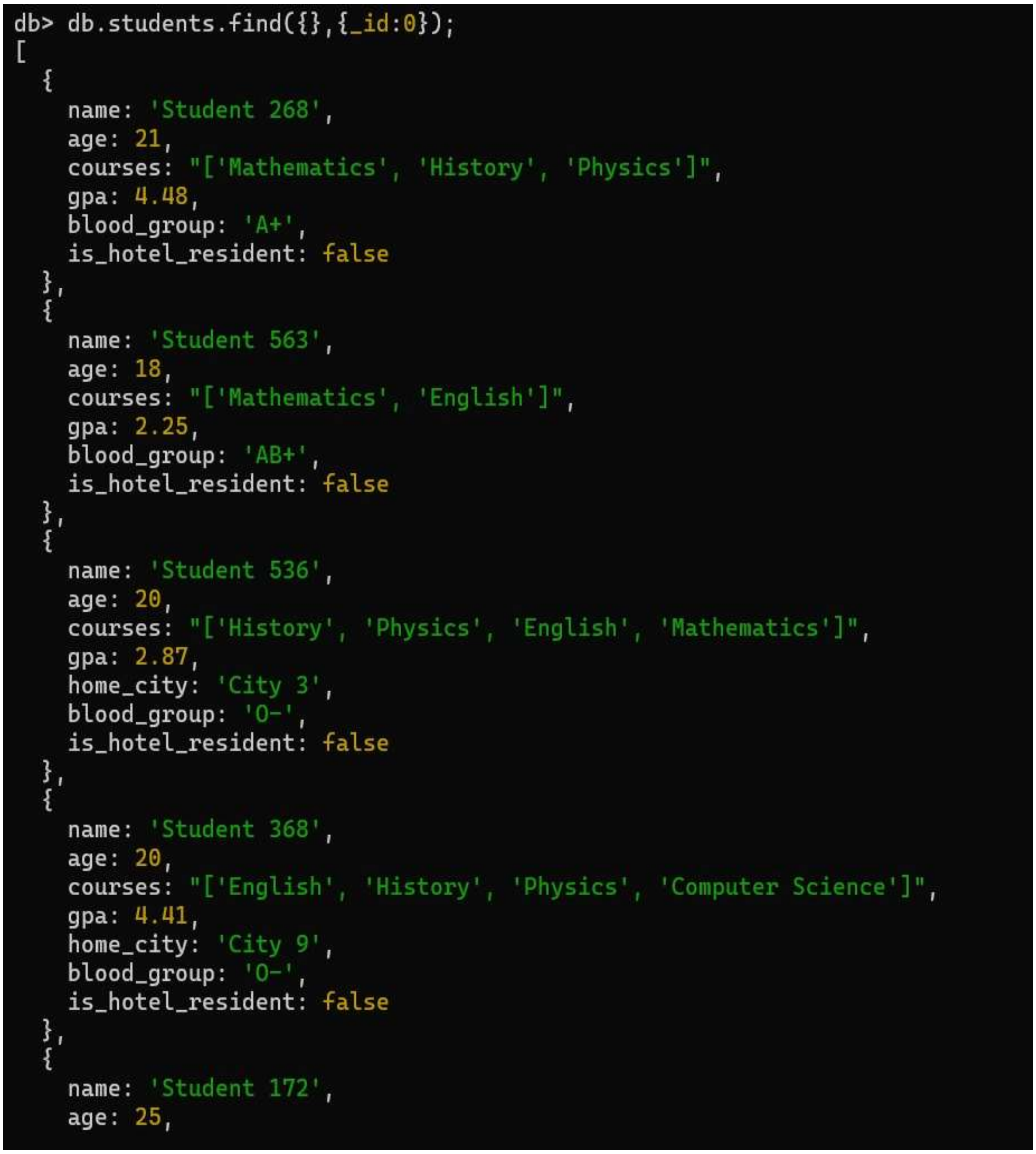
db.students.find({},{name:1,gpa:1,blood_group:"B+"});



```
db> db.students.find({},{name:1,gpa:1,blood_group:"B+"});
[
  {
    _id: ObjectId('6663dac4f24355f2c2a837eb'),
    name: 'Student 268',
    gpa: 4.48,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837ec'),
    name: 'Student 563',
    gpa: 2.25,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837ee'),
    name: 'Student 536',
    gpa: 2.87,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837f5'),
    name: 'Student 368',
    gpa: 4.41,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837f6'),
    name: 'Student 172',
    gpa: 2.46,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837fa'),
    name: 'Student 690',
    gpa: 2.71,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837fb'),
```

To display collection without an id

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



```
db> db.students.find({},{_id:0});
[
  {
    name: 'Student 268',
    age: 21,
    courses: "['Mathematics', 'History', 'Physics']",
    gpa: 4.48,
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    name: 'Student 563',
    age: 18,
    courses: "['Mathematics', 'English']",
    gpa: 2.25,
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    name: 'Student 536',
    age: 20,
    courses: "['History', 'Physics', 'English', 'Mathematics']",
    gpa: 2.87,
    home_city: 'City 3',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Student 368',
    age: 20,
    courses: "['English', 'History', 'Physics', 'Computer Science']",
    gpa: 4.41,
    home_city: 'City 9',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Student 172',
    age: 25,
```

GET SELECTED ATTRIBUTES:

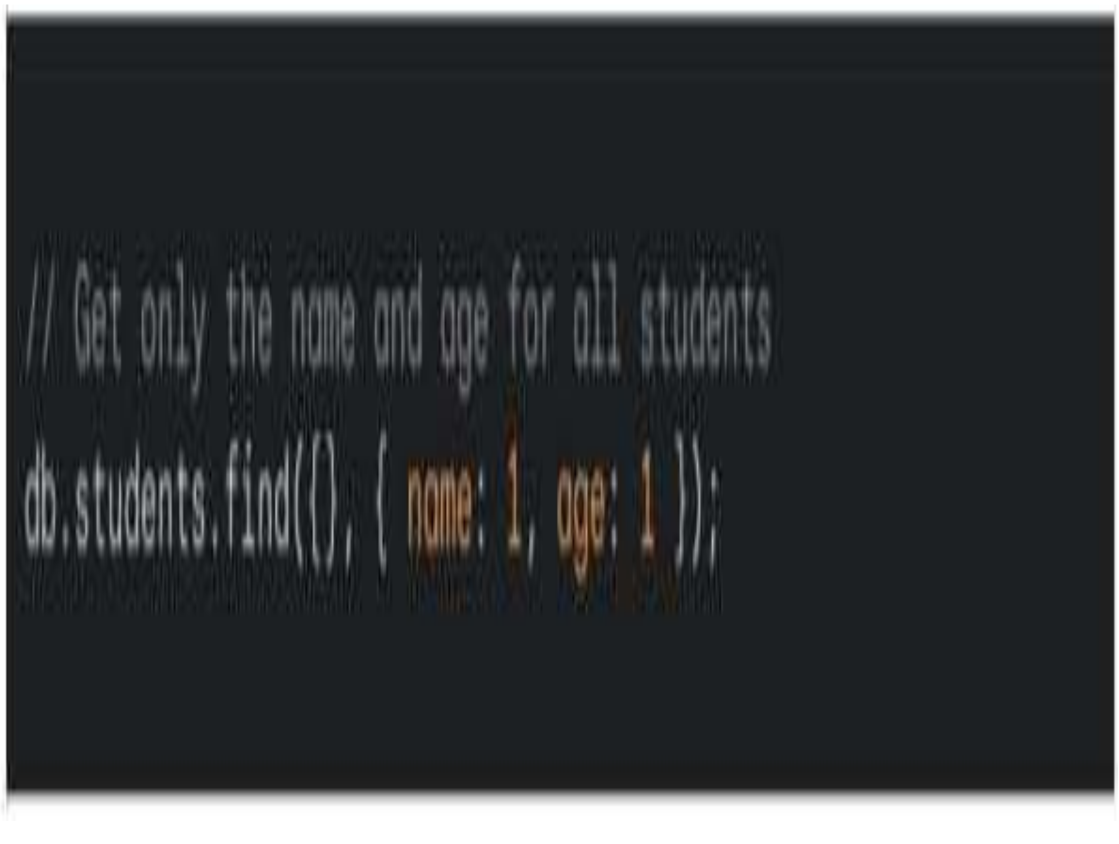
To retrieve selected attributes in MongoDB, use the projection operator in the find method. This allows you to specify the fields you want to include or exclude in the query results.

Here's an example:

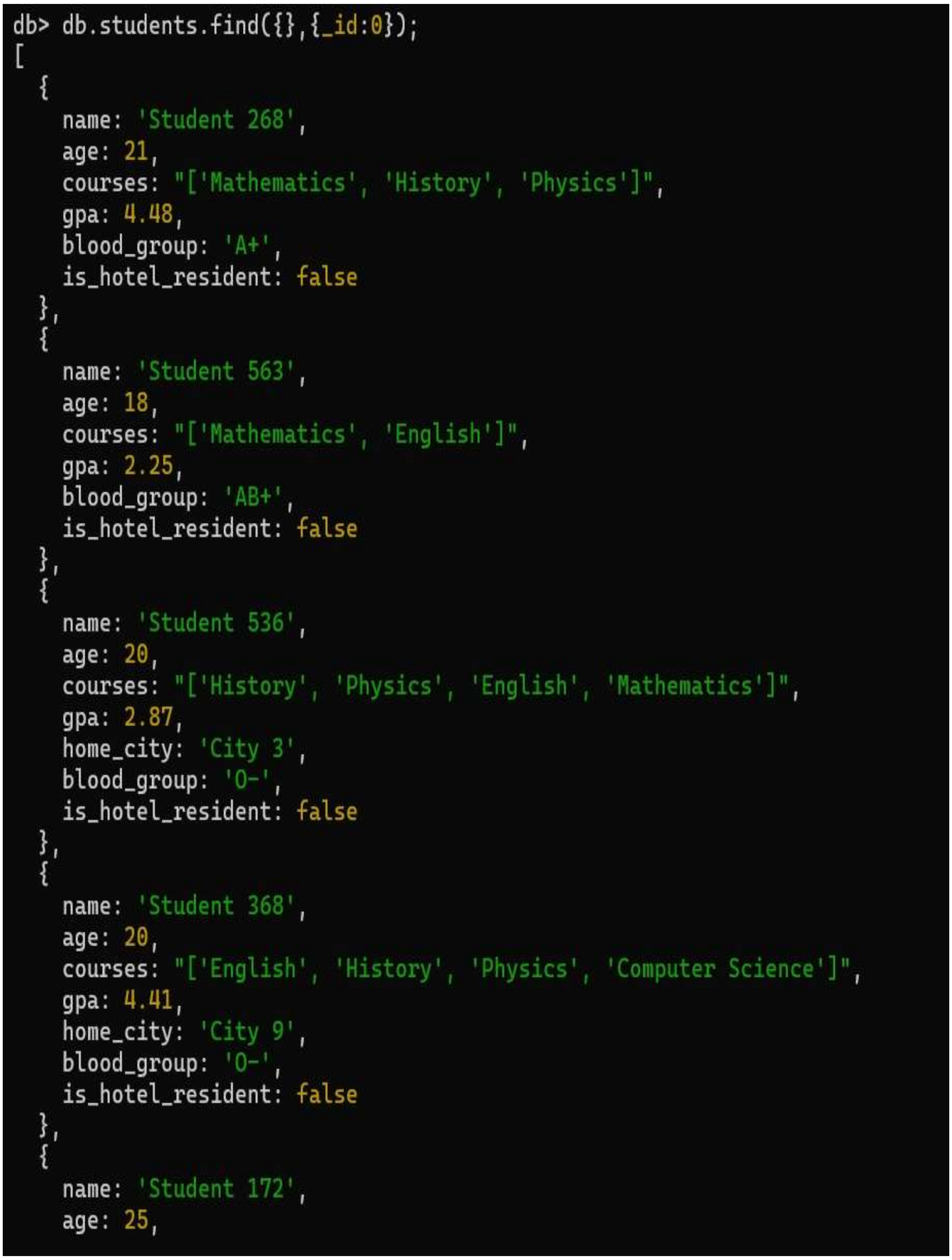
db.collection.find({}, { attribute1: 1, attribute2: 1, ... })

In this query:

- "attribute1, attribute2, ..." are the specific attributes you want to include in the output.
- You can set the attribute to 1 to include it or 0 to exclude it.



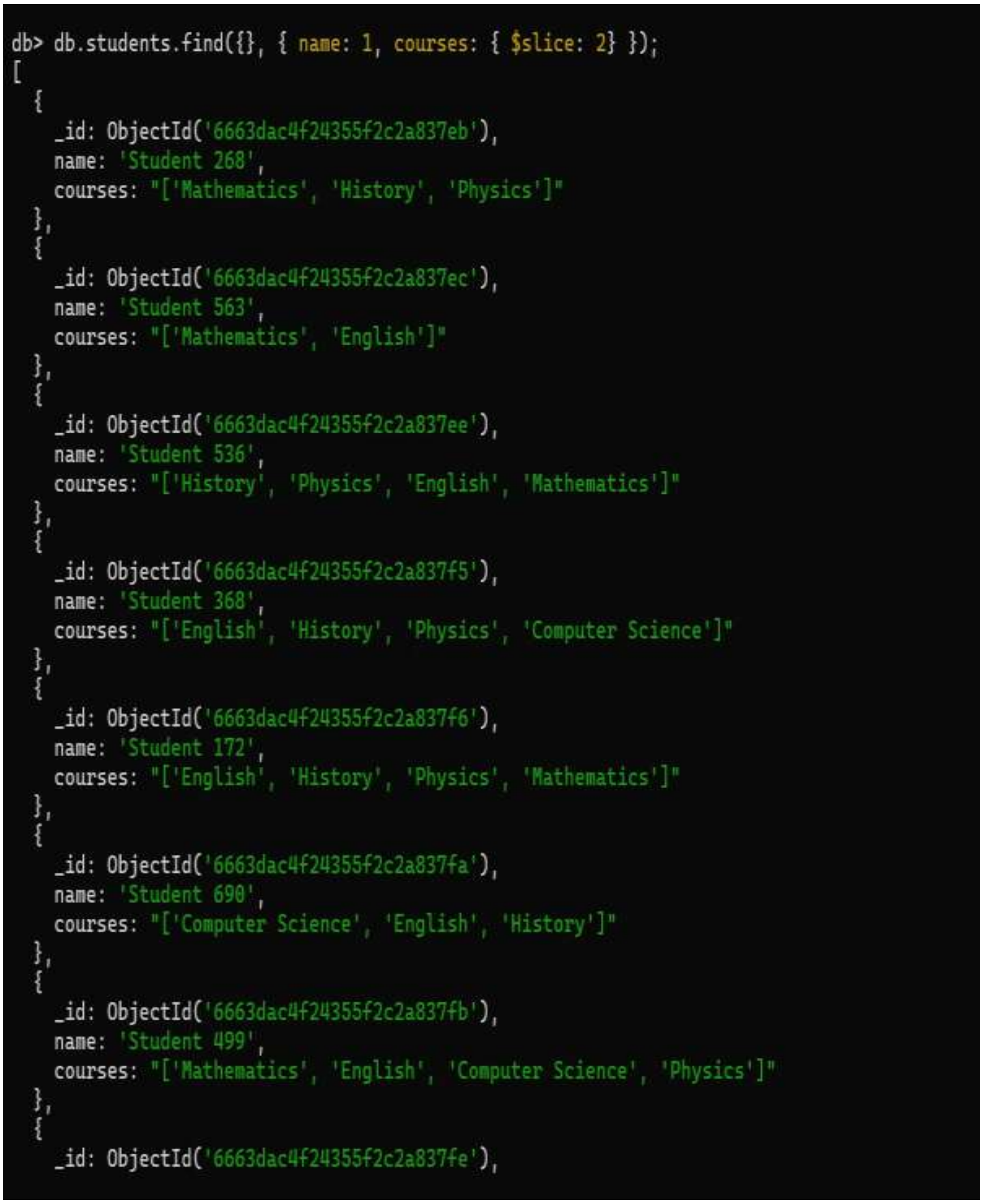
```
// Get only the name and age for all students
db.students.find({}, { name: 1, age: 1 });
```



```
db> db.students.find({},{_id:0});
[
  {
    name: 'Student 268',
    age: 21,
    courses: "['Mathematics', 'History', 'Physics']",
    gpa: 4.48,
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    name: 'Student 563',
    age: 18,
    courses: "['Mathematics', 'English']",
    gpa: 2.25,
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    name: 'Student 536',
    age: 20,
    courses: "['History', 'Physics', 'English', 'Mathematics']",
    gpa: 2.87,
    home_city: 'City 3',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Student 368',
    age: 20,
    courses: "['English', 'History', 'Physics', 'Computer Science']",
    gpa: 4.41,
    home_city: 'City 9',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Student 172',
    age: 25,
```

To display only name and age:

db.students.find({},{name:1,age:1});

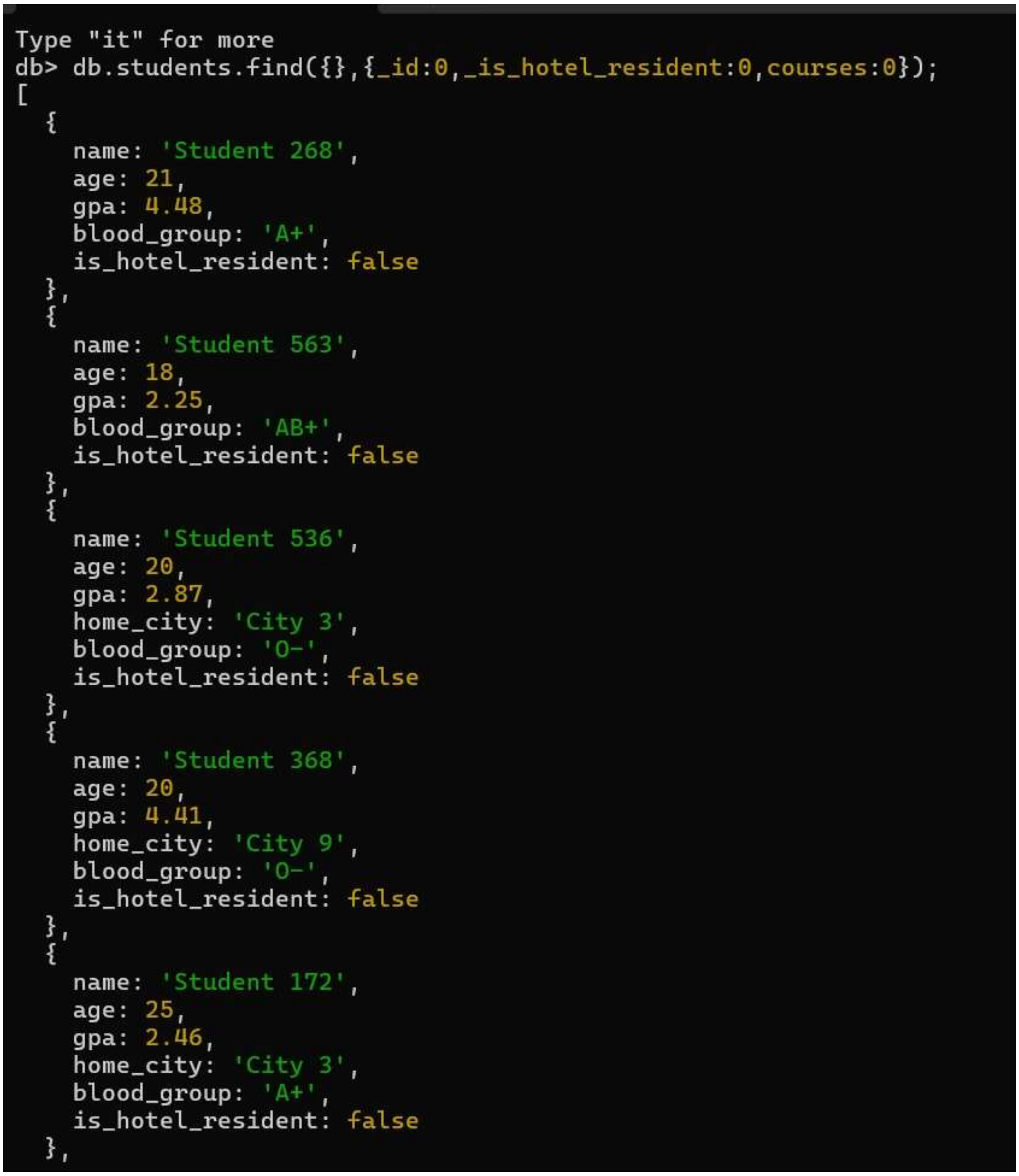


```
db> db.students.find({}, { name: 1, courses: { $slice: 2} });
[
  {
    _id: ObjectId('6663dac4f24355f2c2a837eb'),
    name: 'Student 268',
    courses: "['Mathematics', 'History', 'Physics']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837ec'),
    name: 'Student 563',
    courses: "['Mathematics', 'English']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837ee'),
    name: 'Student 536',
    courses: "['History', 'Physics', 'English', 'Mathematics']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837f5'),
    name: 'Student 368',
    courses: "['English', 'History', 'Physics', 'Computer Science']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837f6'),
    name: 'Student 172',
    courses: "['English', 'History', 'Physics', 'Mathematics']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837fa'),
    name: 'Student 690',
    courses: "['Computer Science', 'English', 'History']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837fb'),
    name: 'Student 499',
    courses: "['Mathematics', 'English', 'Computer Science', 'Physics']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837fe'),
```

IGNORE ATTRIBUTES:

SET ALL THE ATTRIBUTES TO 0.

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



```
Type "it" for more
db> db.students.find({},{_id:0,_is_hotel_resident:0,courses:0});
[
  {
    name: 'Student 268',
    age: 21,
    gpa: 4.48,
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    name: 'Student 563',
    age: 18,
    gpa: 2.25,
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    name: 'Student 536',
    age: 20,
    gpa: 2.87,
    home_city: 'City 3',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Student 368',
    age: 20,
    gpa: 4.41,
    home_city: 'City 9',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Student 172',
    age: 25,
    gpa: 2.46,
    home_city: 'City 3',
    blood_group: 'A+',
    is_hotel_resident: false
  },
```

RETRIEVING SPECIFIC FIELDS FROM NESTED OBJECTS:

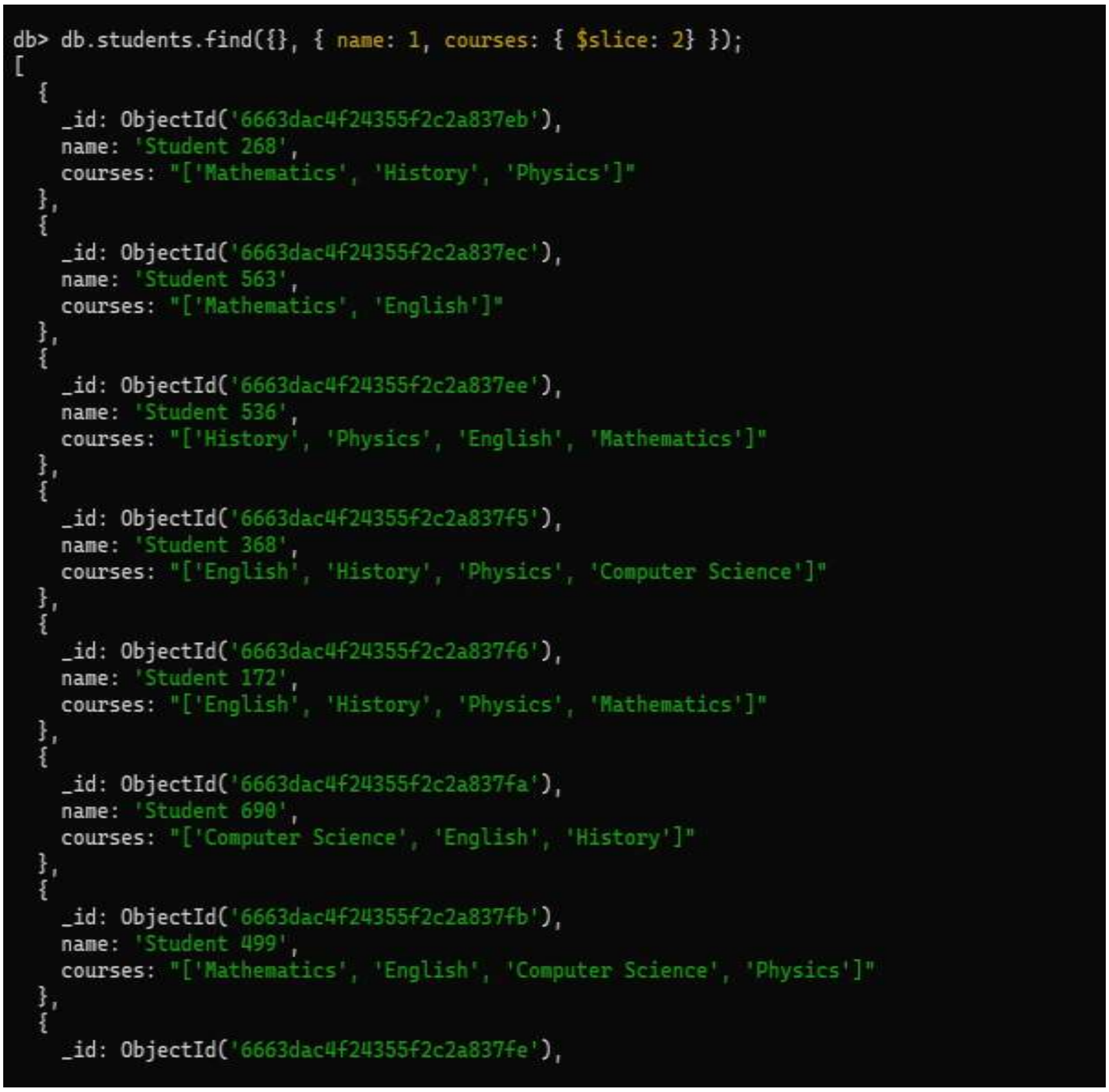
Find the output for $slice:2

Db.students.find({},{

Name:1,

Courses:{$slice:1}

});



```
db> db.students.find({}, { name: 1, courses: { $slice: 2} });
[
  {
    _id: ObjectId('6663dac4f24355f2c2a837eb'),
    name: 'Student 268',
    courses: "['Mathematics', 'History', 'Physics']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837ec'),
    name: 'Student 563',
    courses: "['Mathematics', 'English']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837ee'),
    name: 'Student 536',
    courses: "['History', 'Physics', 'English', 'Mathematics']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837f5'),
    name: 'Student 368',
    courses: "['English', 'History', 'Physics', 'Computer Science']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837f6'),
    name: 'Student 172',
    courses: "['English', 'History', 'Physics', 'Mathematics']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837fa'),
    name: 'Student 690',
    courses: "['Computer Science', 'English', 'History']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837fb'),
    name: 'Student 499',
    courses: "['Mathematics', 'English', 'Computer Science', 'Physics']"
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837fe'),
```

Benefits of Projection:

● Reduces data transferred between the database and your application.

● Improves query performance by retrieving only necessary data.

● Simplifies your code by focusing on the specific information yo[9:11 pm, 11/6/2024] !: Projection

● Use the projection document as the second argument to the find method.

● Include field names with a value of 1 to specify fields to be returned.

● Omit fields or set them to 0 to exclude them from the results.

In MongoDB, projection refers to the process of specifying which fields you want to include or exclude in the query results. It allows you to control the data that is returned from a query. By using the projection operator in MongoDB, you can tailor your query results to only include the fields that are relevant to your needs.

Limits:

● The limit operator is used with the find method.

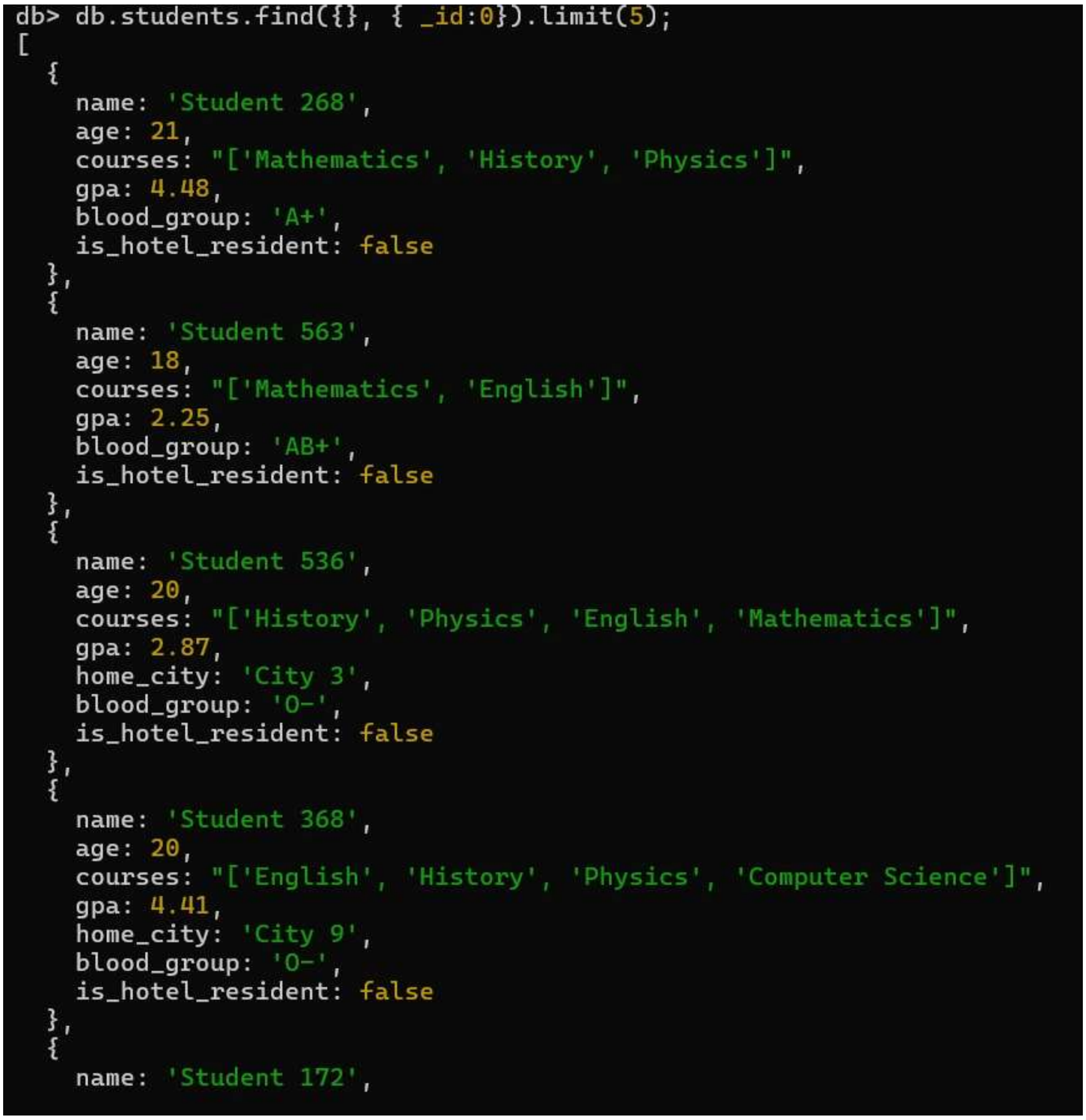
● It's chained after the filter criteria or any sorting operations.

● Syntax: db.collection.find({filter}, {projection}).limit(number)

● Syntax: db.collection.find({filter}, {projection}).limit(number)

GET FIRST 5 DOCUMENT:

```
db.students.find({},{_id:0}).limit(5);
```

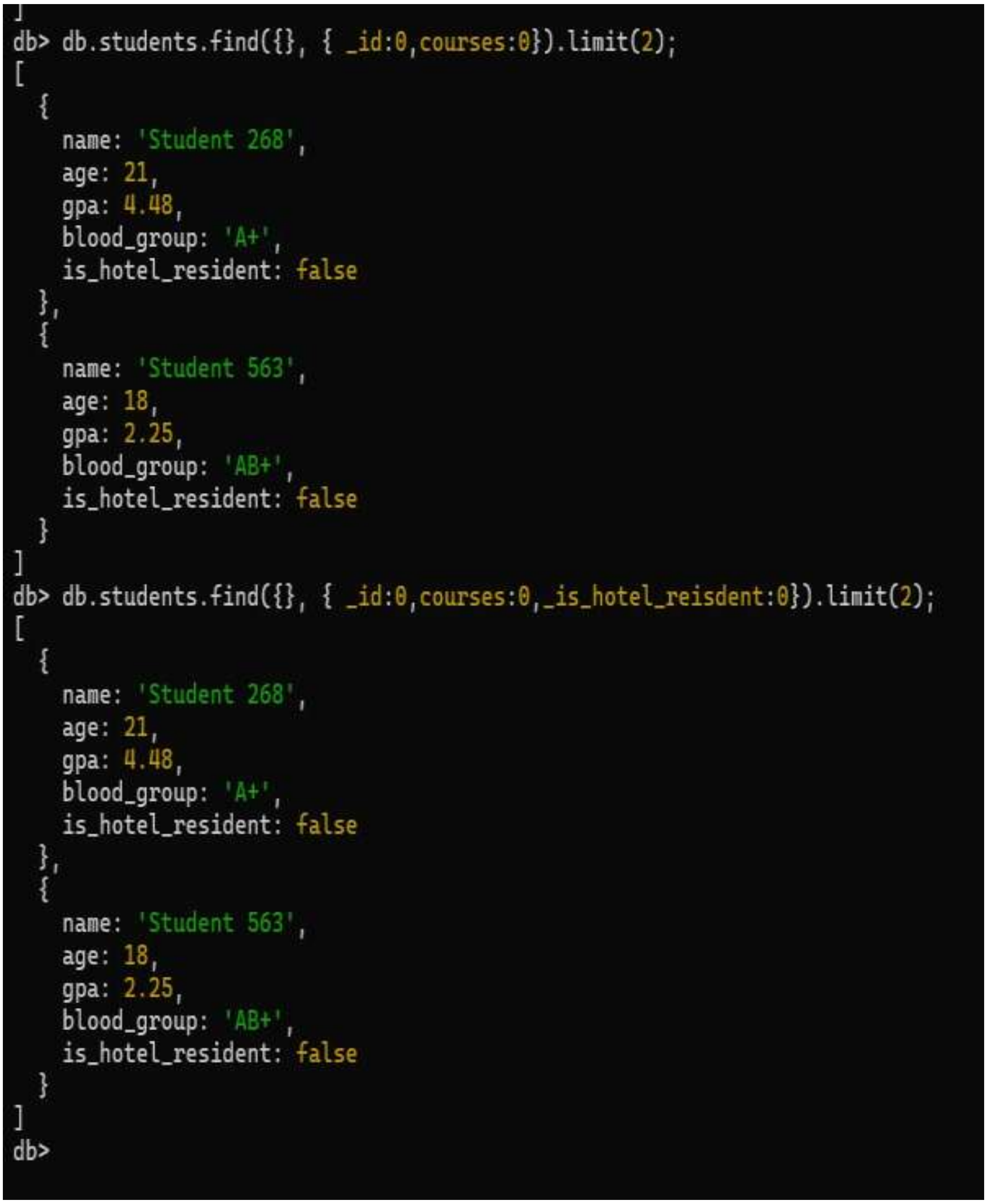


```
db> db.students.find({}, { _id:0}).limit(5);
[
  {
    name: 'Student 268',
    age: 21,
    courses: "['Mathematics', 'History', 'Physics']",
    gpa: 4.48,
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    name: 'Student 563',
    age: 18,
    courses: "['Mathematics', 'English']",
    gpa: 2.25,
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    name: 'Student 536',
    age: 20,
    courses: "['History', 'Physics', 'English', 'Mathematics']",
    gpa: 2.87,
    home_city: 'City 3',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Student 368',
    age: 20,
    courses: "['English', 'History', 'Physics', 'Computer Science']",
    gpa: 4.41,
    home_city: 'City 9',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Student 172',
```

LIMITING RESULTS:

```
db.students.find({},(_id:0}).sort({_id:-1}).limit(5);
```

here,we have limited to 5 .

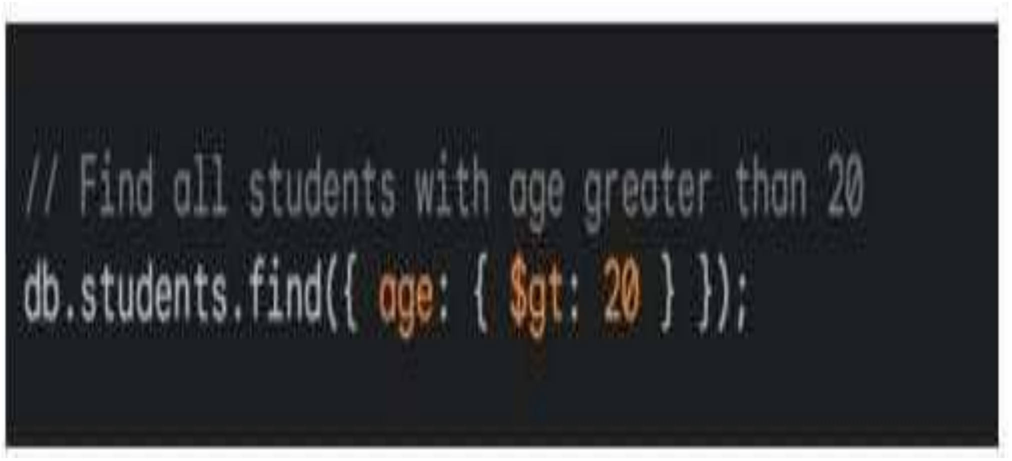


```
]
db> db.students.find({}, { _id:0,courses:0}).limit(2);
[
  {
    name: 'Student 268',
    age: 21,
    gpa: 4.48,
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    name: 'Student 563',
    age: 18,
    gpa: 2.25,
    blood_group: 'AB+',
    is_hotel_resident: false
  }
]
db> db.students.find({}, { _id:0,courses:0,_is_hotel_reisdent:0}).limit(2);
[
  {
    name: 'Student 268',
    age: 21,
    gpa: 4.48,
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    name: 'Student 563',
    age: 18,
    gpa: 2.25,
    blood_group: 'AB+',
    is_hotel_resident: false
  }
]
db>
```

SELECTORS:

Comparision gt it:

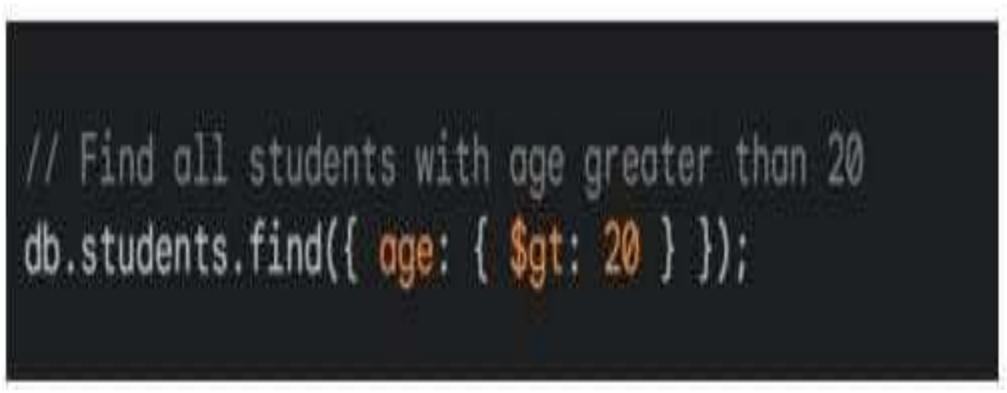
Db.students.find({age:{$gt:20}});



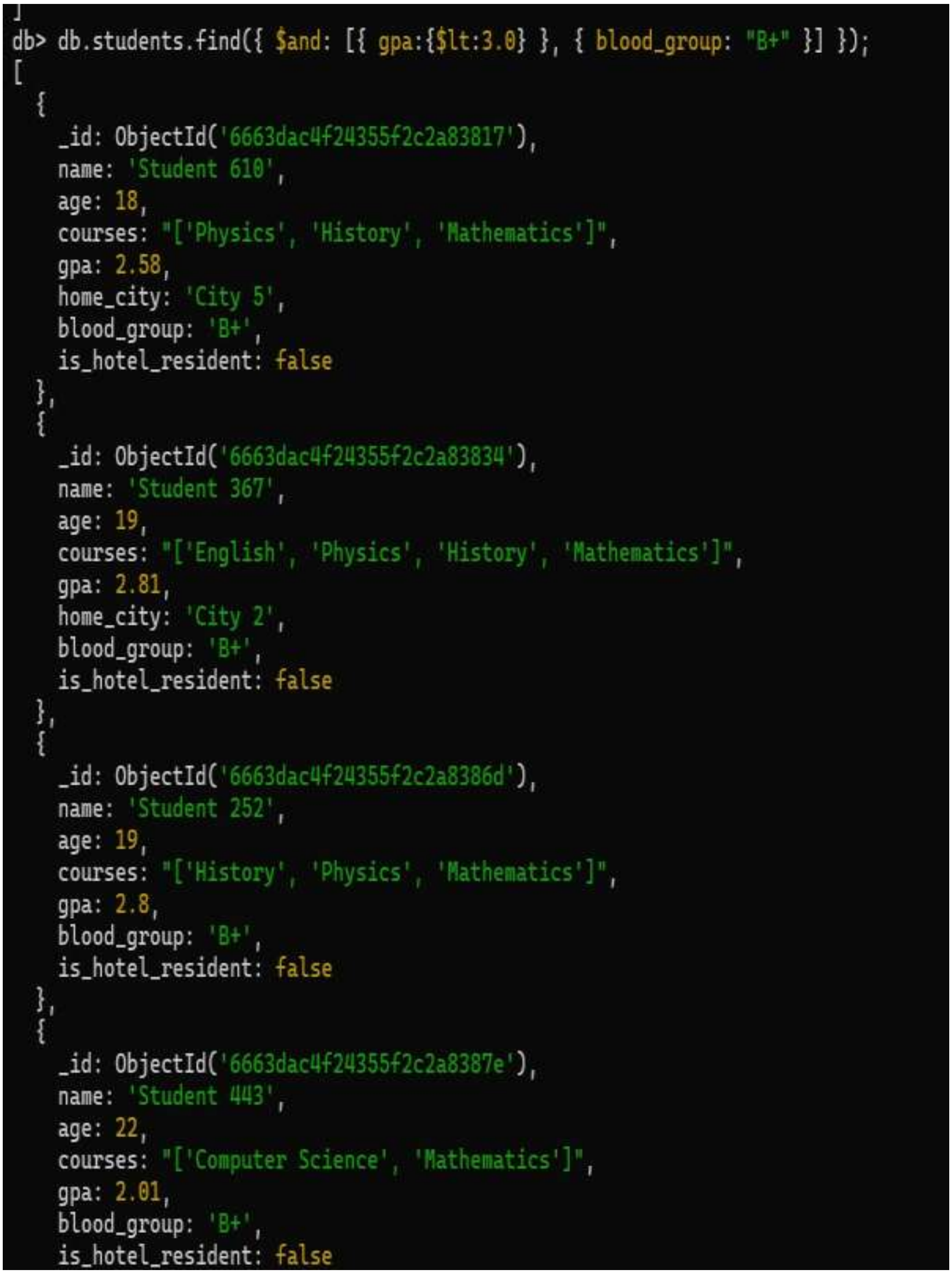
```
// Find all students with age greater than 20
db.students.find({ age: { $gt: 20 } });
```

AND OPERATOR:

In MongoDB, the $and operator is used to combine multiple expressions so that all expressions must be true for a document to be included in the query results. It allows you to specify multiple conditions that must all be satisfied for a document to match the query.



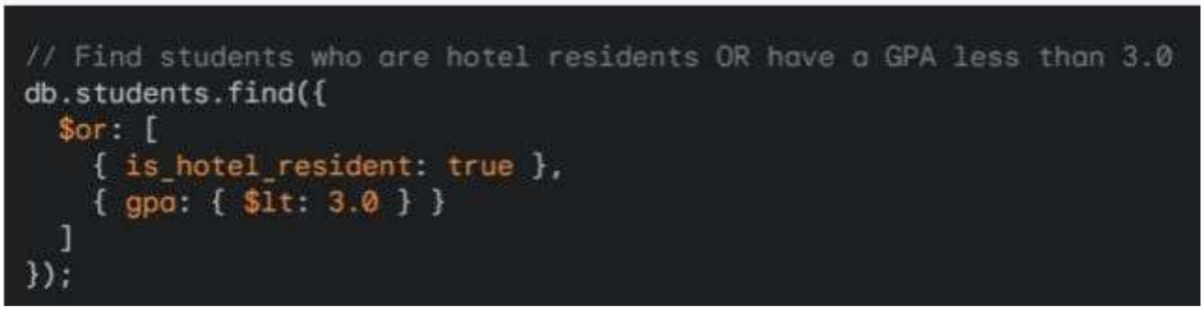
```
// Find all students with age greater than 20
db.students.find({ age: { $gt: 20 } });
```



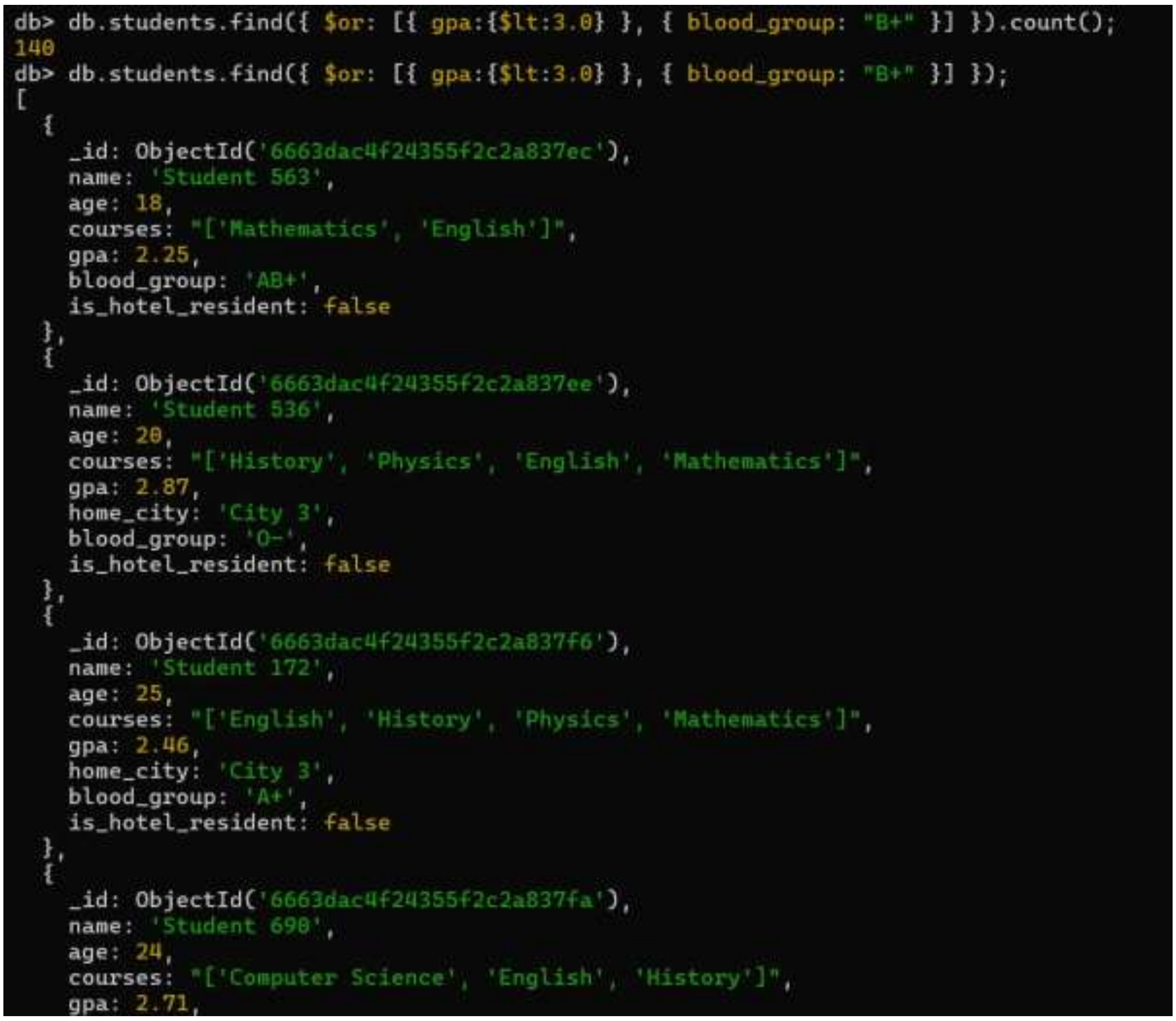
```
]
db> db.students.find({ $and: [{ gpa:{$lt:3.0} }, { blood_group: "B+" }] });
[
  {
    _id: ObjectId('6663dac4f24355f2c2a83817'),
    name: 'Student 610',
    age: 18,
    courses: "['Physics', 'History', 'Mathematics']",
    gpa: 2.58,
    home_city: 'City 5',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a83834'),
    name: 'Student 367',
    age: 19,
    courses: "['English', 'Physics', 'History', 'Mathematics']",
    gpa: 2.81,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a8386d'),
    name: 'Student 252',
    age: 19,
    courses: "['History', 'Physics', 'Mathematics']",
    gpa: 2.8,
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a8387e'),
    name: 'Student 443',
    age: 22,
    courses: "['Computer Science', 'Mathematics']",
    gpa: 2.01,
    blood_group: 'B+',
    is_hotel_resident: false
```

OR OPERATOR:

To find students who are hostel residents or have a gpa less than 3.0



```
// Find students who are hotel residents OR have a GPA less than 3.0
db.students.find({
  $or: [
    { is_hotel_resident: true },
    { gpa: { $lt: 3.0 } }
  ]
});
```



```
db> db.students.find({ $or: [{ gpa:{$lt:3.0} }, { blood_group: "B+" }] }).count();
140
db> db.students.find({ $or: [{ gpa:{$lt:3.0} }, { blood_group: "B+" }] });
[
  {
    _id: ObjectId('6663dac4f24355f2c2a837ec'),
    name: 'Student 563',
    age: 18,
    courses: "['Mathematics', 'English']",
    gpa: 2.25,
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837ee'),
    name: 'Student 536',
    age: 20,
    courses: "['History', 'Physics', 'English', 'Mathematics']",
    gpa: 2.87,
    home_city: 'City 3',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837f6'),
    name: 'Student 172',
    age: 25,
    courses: "['English', 'History', 'Physics', 'Mathematics']",
    gpa: 2.46,
    home_city: 'City 3',
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837fa'),
    name: 'Student 690',
    age: 24,
    courses: "['Computer Science', 'English', 'History']",
    gpa: 2.71,
```

Take new Data set

● New Students Permission dataset link

Explanation: Collection name: students_permission

● name: Student's name (string)

● age: Student's age (number)

● permissions: Bitmask representing user permissions (number)

Bitwise Value

● In our example its a 32 bit each bit representing different things

● Bitwise value 7 means all access 7-> 111

Bit 3 Bit 2 Bit 1

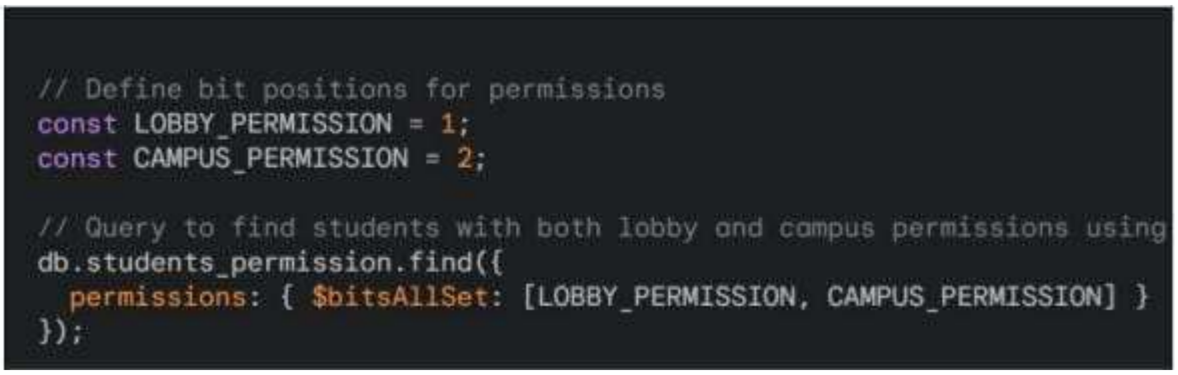
cafe campus lobby

In MongoDB, you can actually use bitwise operations on integer fields within documents. MongoDB provides bitwise query operators such as $bitsAllSet, $bitsAnySet, and $bitsAllClear that allow you to perform bitwise operations on integer fields in your documents.

For example, you can use the $bitsAllSet operator to find documents where a specific set of bits is set in an integer field. Similarly, the $bitsAnySet operator can be used to find documents where any of the specified bits are set. Lastly, the $bitsAllClear operator helps you find documents where all the specified bits are clear.

These operators can be handy when working with integer fields that store bit flags or other bitwise data.

QUERY:



```
// Define bit positions for permissions
const LOBBY_PERMISSION = 1;
const CAMPUS_PERMISSION = 2;

// Query to find students with both lobby and campus permissions using
db.students_permission.find({
  permissions: { $bitsAllSet: [LOBBY_PERMISSION, CAMPUS_PERMISSION] }
});
```

GEOSPATIAL QUERY:

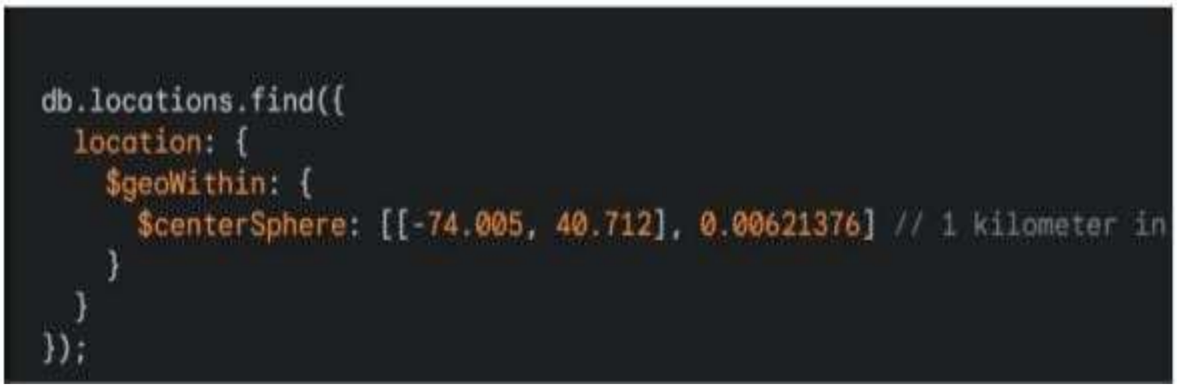
Now,we need to upload a new collection names “location” in json format.

Later,

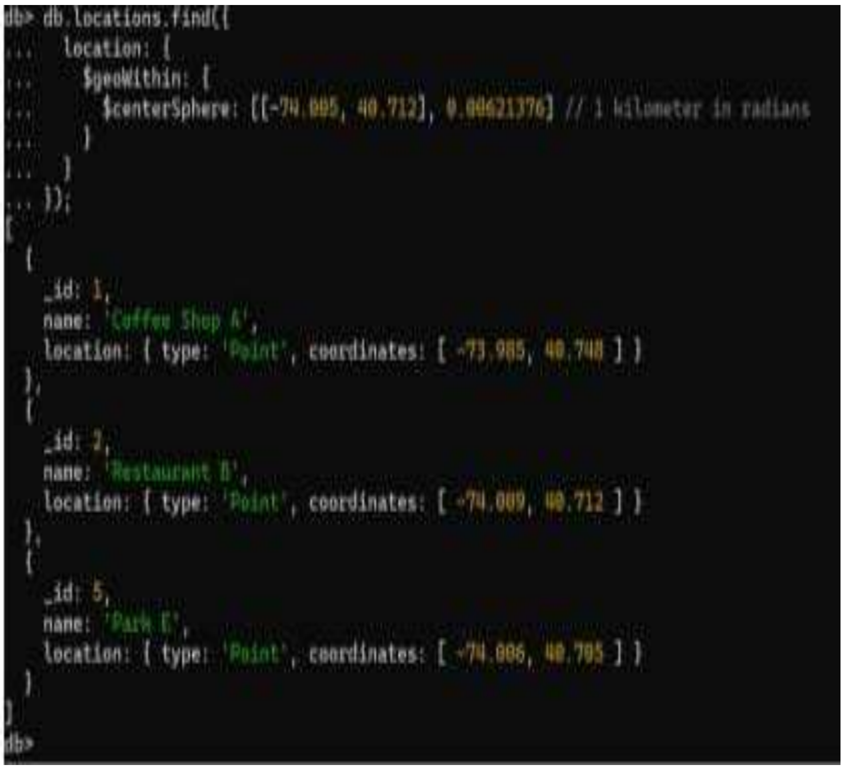
Use db

Show dbs

Show collections.



```
db.locations.find({
  location: {
    $geoWithin: {
      $centerSphere: [[-74.005, 40.712], 0.00621376] // 1 kilometer in
    }
  }
});
```



```
db> db.locations.find({
...   location: {
...     $geoWithin: {
...       $centerSphere: [[-74.005, 40.712], 0.00621376] // 1 kilometer in radians
...     }
...   }
... });
[
  {
    _id: 1,
    name: 'Coffee Shop A',
    location: { type: 'Point', coordinates: [ -73.985, 40.748 ] }
  },
  {
    _id: 2,
    name: 'Restaurant B',
    location: { type: 'Point', coordinates: [ -74.009, 40.712 ] }
  },
  {
    _id: 5,
    name: 'Park E',
    location: { type: 'Point', coordinates: [ -74.006, 40.705 ] }
  }
]
db>
```

DATA TYPES AND OPERATIONS:

<u>DATA TYPES:</u>

*POINT

*LINE STRING

*POLYGON