CLASS-4

Projection, Limit & Selectors

Projection:

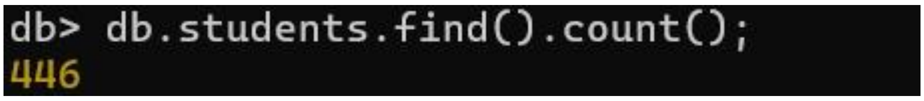
In MongoDB, projections are used to specify or restrict which fields should be returned in the documents that match a query. This can help reduce the amount of data transferred over the network and improve query performance by retrieving only the necessary fields. This is used when we don’t need all columns/attributes

Syntax:

To use projections, you include a second parameter in the find method, specifying the fields to include or exclude.

```
db.collection.find(query, projection);
```

Collection count:

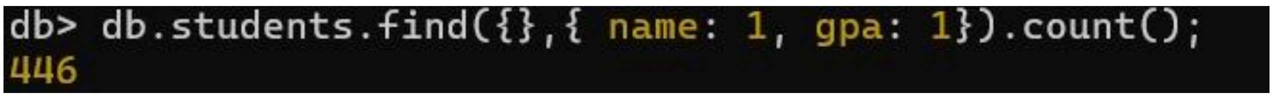


```
db> db.students.find().count();
446
```

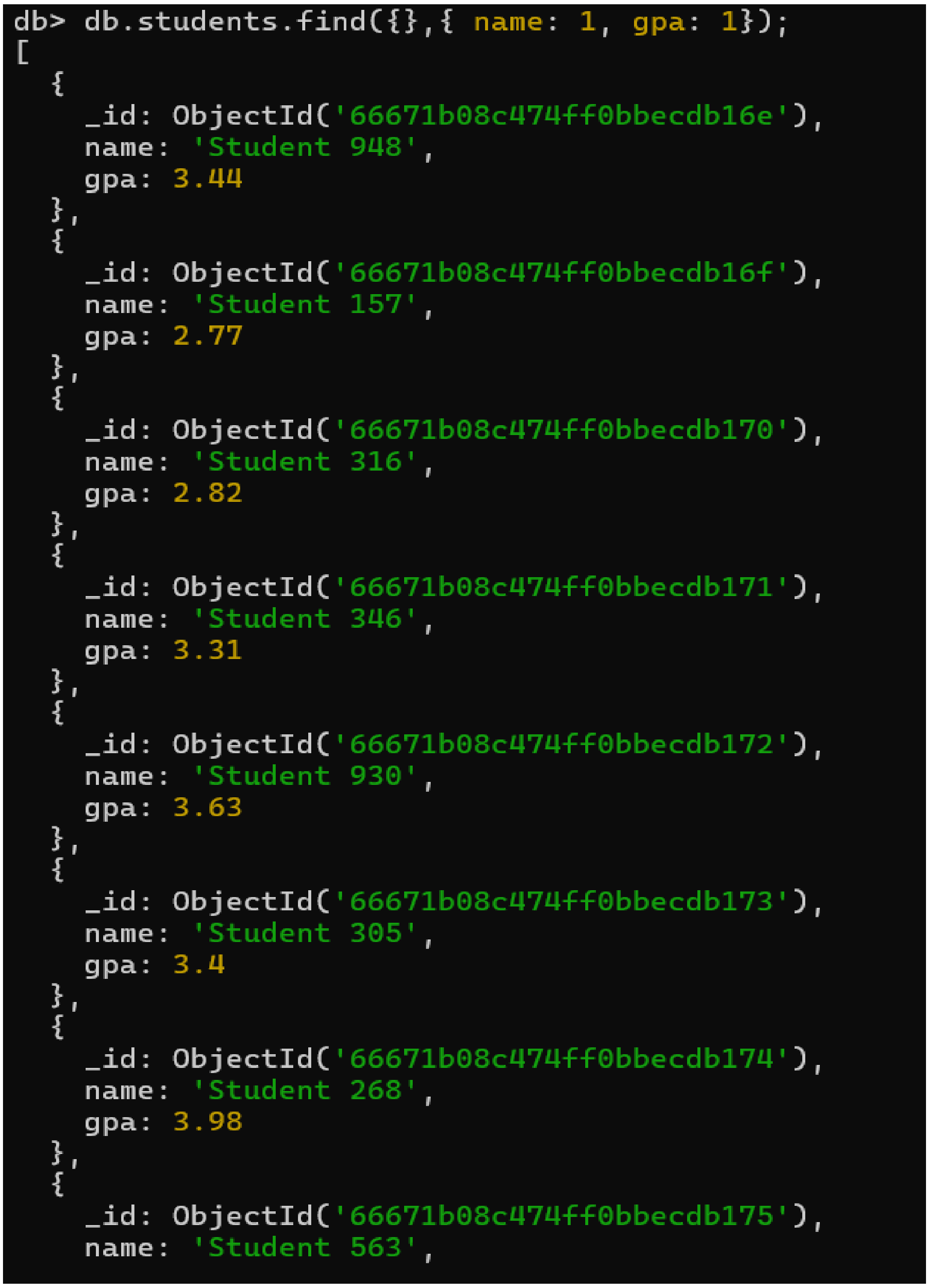
Example:

To get the selected attributes

Gets only the name and gpa of all students.

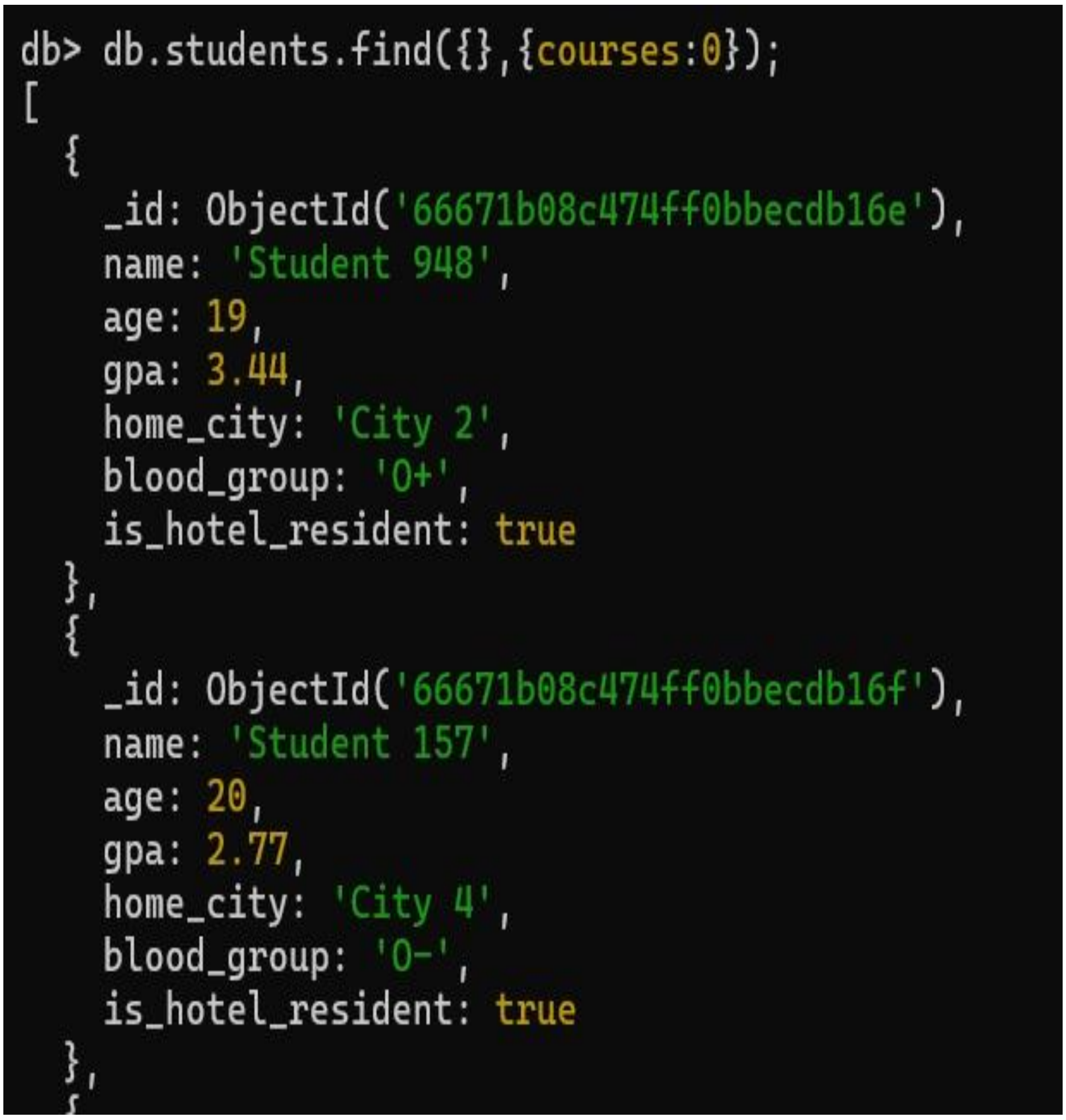


```
db> db.students.find({},{ name: 1, gpa: 1}).count();
446
```



```
db> db.students.find({},{ name: 1, gpa: 1});
[
  {
    _id: ObjectId('66671b08c474ff0bbecdb16e'),
    name: 'Student 948',
    gpa: 3.44
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb16f'),
    name: 'Student 157',
    gpa: 2.77
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb170'),
    name: 'Student 316',
    gpa: 2.82
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb171'),
    name: 'Student 346',
    gpa: 3.31
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb172'),
    name: 'Student 930',
    gpa: 3.63
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb173'),
    name: 'Student 305',
    gpa: 3.4
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb174'),
    name: 'Student 268',
    gpa: 3.98
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb175'),
    name: 'Student 563',
```

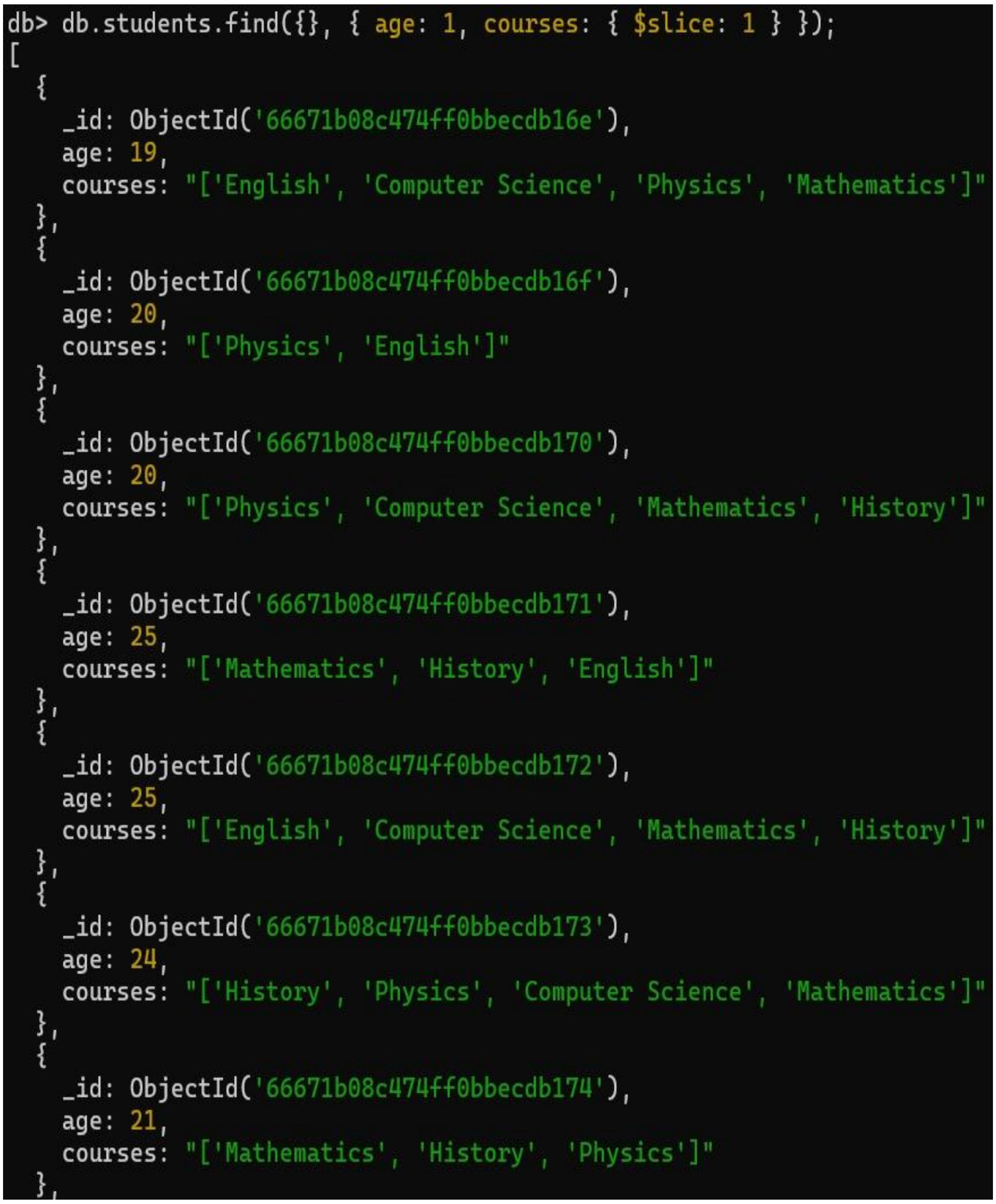
Ignoring Attributes:



```
db> db.students.find({},{courses:0});
[
  {
    _id: ObjectId('66671b08c474ff0bbecdb16e'),
    name: 'Student 948',
    age: 19,
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb16f'),
    name: 'Student 157',
    age: 20,
    gpa: 2.77,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
```

Retrieving Specific Fields from Nested Objects

The $slice operator in MongoDB is used to select a subset of an array. It is particularly useful when you have large arrays stored in your documents and you only need to retrieve certain elements, optimizing data retrieval and reducing overhead.



```
db> db.students.find({}, { age: 1, courses: { $slice: 1 } });
[
  {
    _id: ObjectId('66671b08c474ff0bbecdb16e'),
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']"
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb16f'),
    age: 20,
    courses: "['Physics', 'English']"
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb170'),
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']"
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb171'),
    age: 25,
    courses: "['Mathematics', 'History', 'English']"
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb172'),
    age: 25,
    courses: "['English', 'Computer Science', 'Mathematics', 'History']"
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb173'),
    age: 24,
    courses: "['History', 'Physics', 'Computer Science', 'Mathematics']"
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb174'),
    age: 21,
    courses: "['Mathematics', 'History', 'Physics']"
  },
```

Benefits of Projection

1. Improved Query Performance

- Reduced Data Transfer: By retrieving only the necessary fields, projections reduce the amount of data sent over the network, leading to faster query response times.

- Lower Memory Usage: Projections decrease the amount of memory required to store query results on both the client and server side.

2. Efficient Resource Utilization

- Server Resources: The server uses fewer resources (CPU, memory) when processing and returning smaller subsets of documents.
- Client Resources: The client application can process and handle smaller result sets more efficiently, leading to better performance and responsiveness.

3. Optimized Application Performance
- Faster Data Processing: Applications can process query results more quickly when only the relevant data is included, improving overall application performance.
- Simplified Data Handling: By receiving only the necessary fields, the complexity of data handling within the application is reduced, making the codebase cleaner and more maintainable.

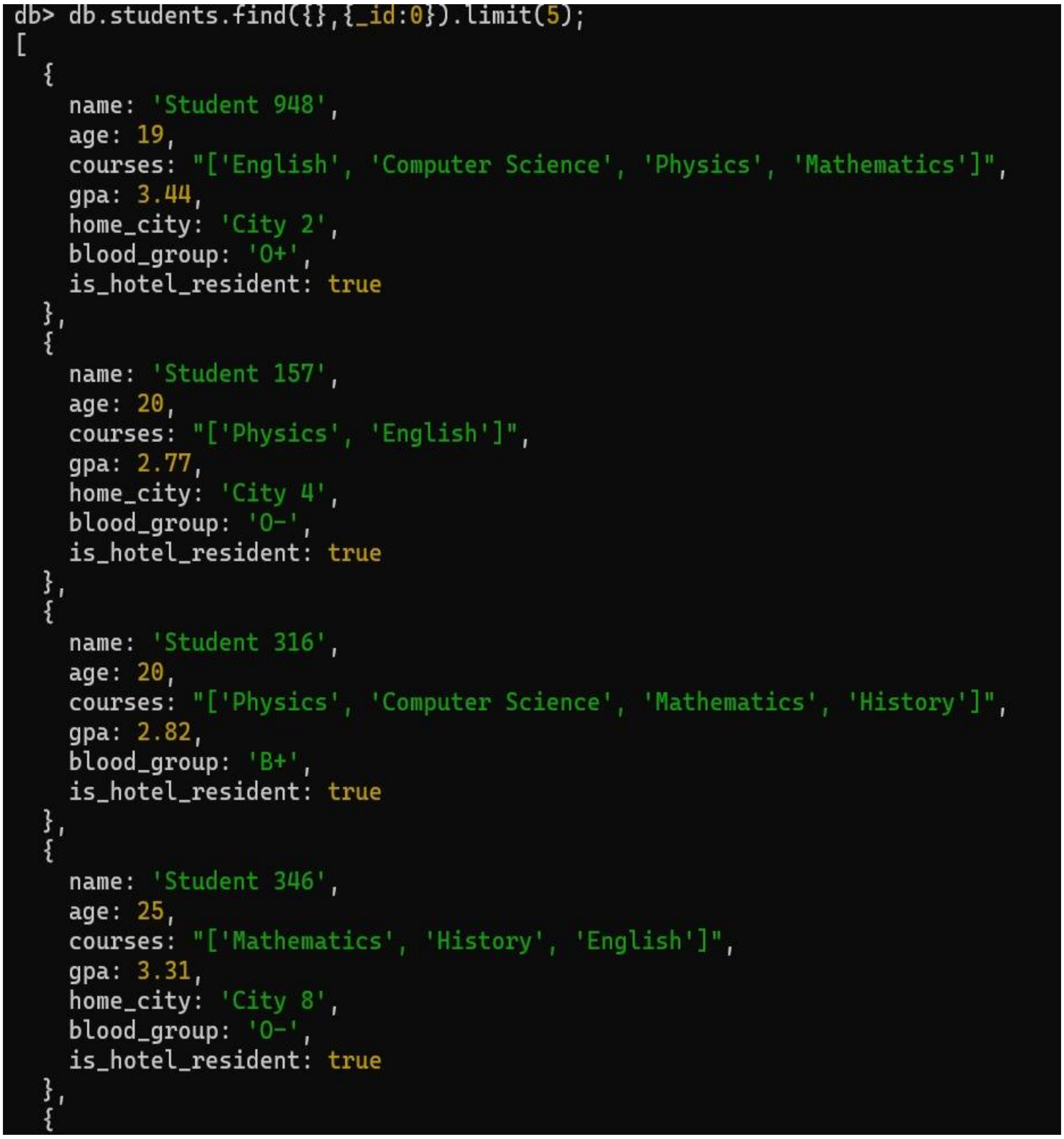
4. Reduces data transferred between the database and your application.

Limit:
In MongoDB, the limit method is used to restrict the number of documents returned by a query. This can be useful when you only need a subset of the results or when you want to paginate through a large set of documents.

- The limit operator is used with the find method.
- It's chained after the filter criteria or any sorting operations.

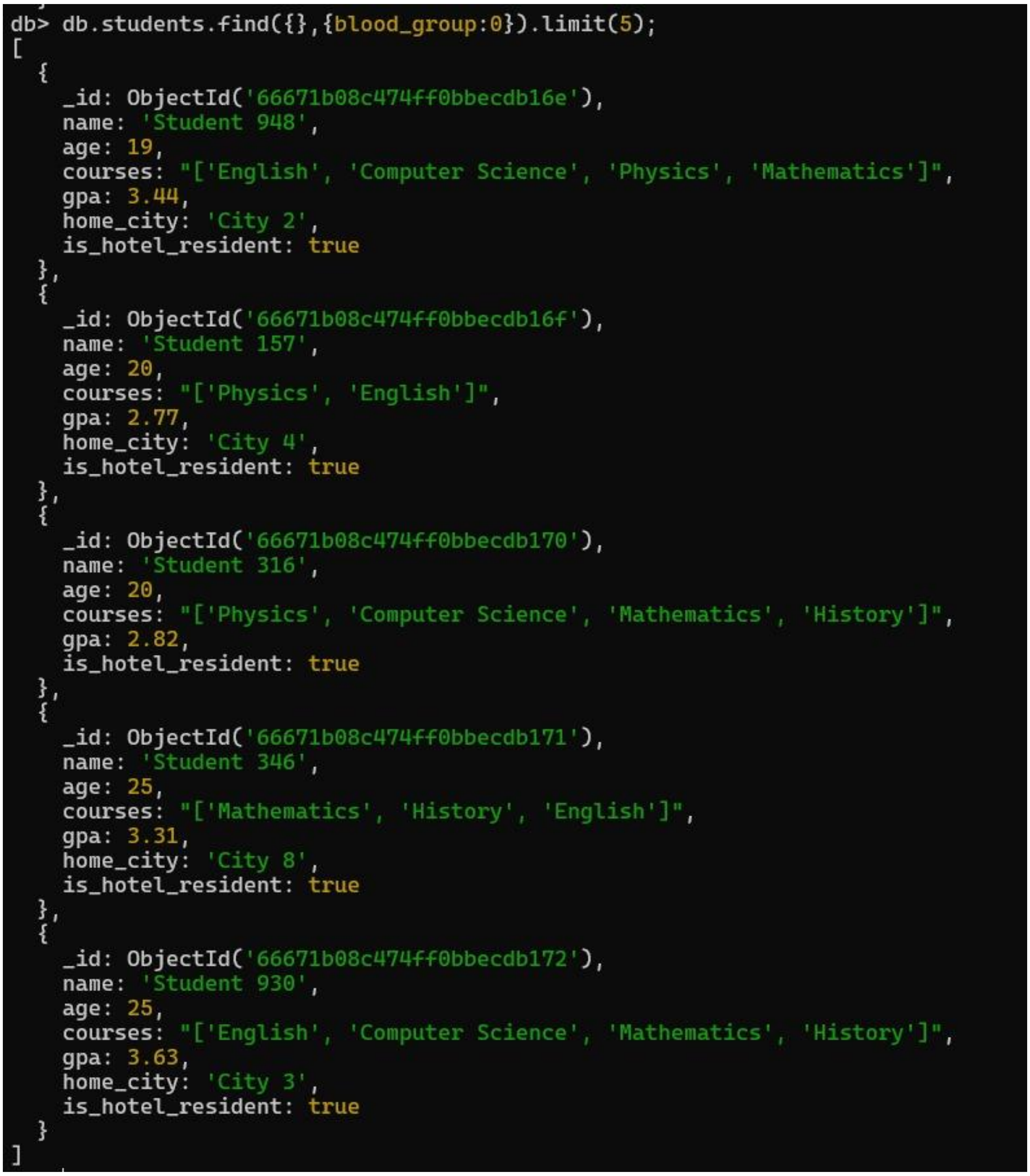
Syntax:

```
db.collection.find(query).limit(number);
```



```
db> db.students.find({},{_id:0}).limit(5);
[
  {
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.77,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.82,
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Student 346',
    age: 25,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 3.31,
    home_city: 'City 8',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
```

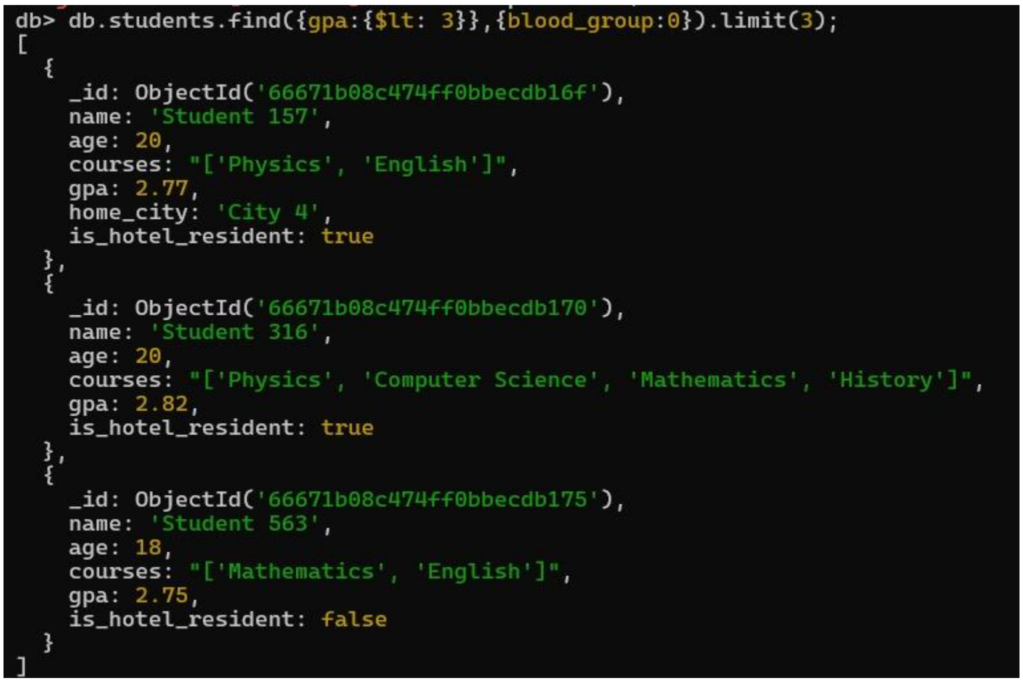
Getting First documents:



```
db> db.students.find({},{blood_group:0}).limit(5);
[
  {
    _id: ObjectId('66671b08c474ff0bbecdb16e'),
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb16f'),
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.77,
    home_city: 'City 4',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb170'),
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.82,
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb171'),
    name: 'Student 346',
    age: 25,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 3.31,
    home_city: 'City 8',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb172'),
    name: 'Student 930',
    age: 25,
    courses: "['English', 'Computer Science', 'Mathematics', 'History']",
    gpa: 3.63,
    home_city: 'City 3',
    is_hotel_resident: true
  }
]
```

Limiting Results:

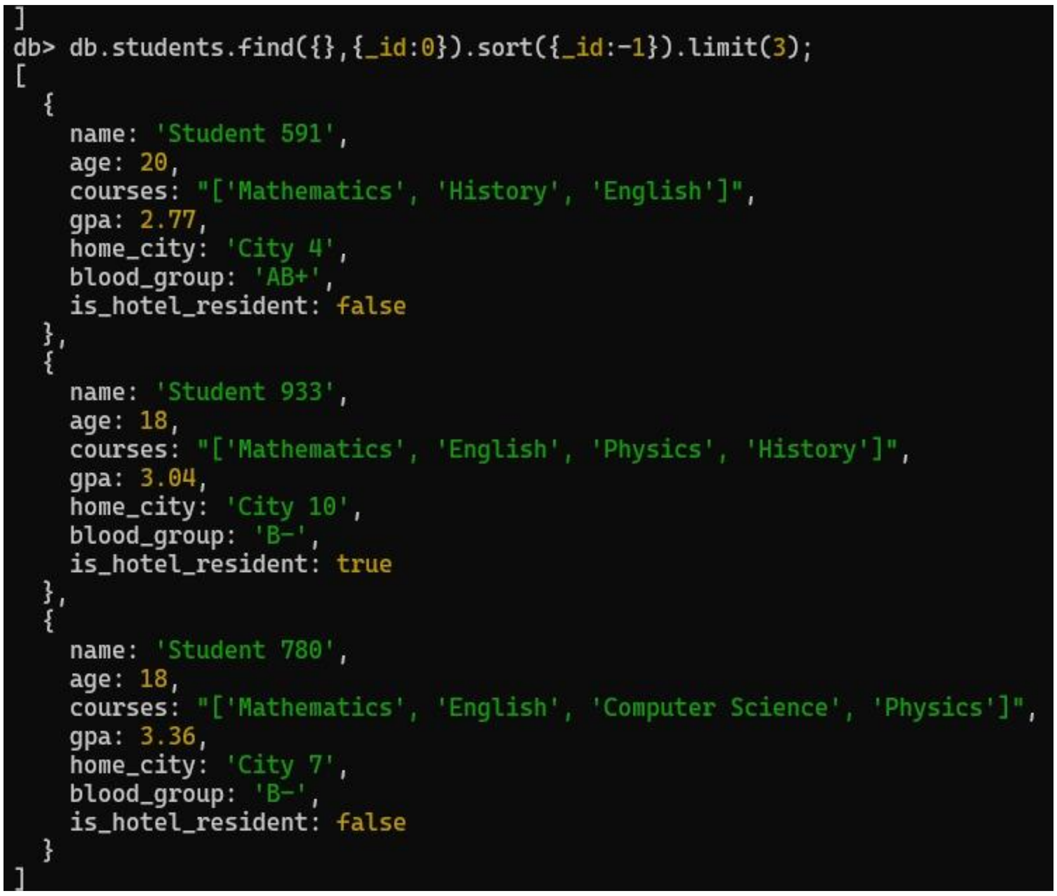
Finding all students with gpa lesser than 3 and limiting to 3 documents.



```
db> db.students.find({gpa:{$lt: 3}},{blood_group:0}).limit(3);
[
  {
    _id: ObjectId('66671b08c474ff0bbecdb16f'),
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.77,
    home_city: 'City 4',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb170'),
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.82,
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb175'),
    name: 'Student 563',
    age: 18,
    courses: "['Mathematics', 'English']",
    gpa: 2.75,
    is_hotel_resident: false
  }
]
```

Top 10 results:

Sorting documents in descending order by _id and limiting to 3



```
]
db> db.students.find({},{_id:0}).sort({_id:-1}).limit(3);
[
  {
    name: 'Student 591',
    age: 20,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 2.77,
    home_city: 'City 4',
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    name: 'Student 933',
    age: 18,
    courses: "['Mathematics', 'English', 'Physics', 'History']",
    gpa: 3.04,
    home_city: 'City 10',
    blood_group: 'B-',
    is_hotel_resident: true
  },
  {
    name: 'Student 780',
    age: 18,
    courses: "['Mathematics', 'English', 'Computer Science', 'Physics']",
    gpa: 3.36,
    home_city: 'City 7',
    blood_group: 'B-',
    is_hotel_resident: false
  }
]
```

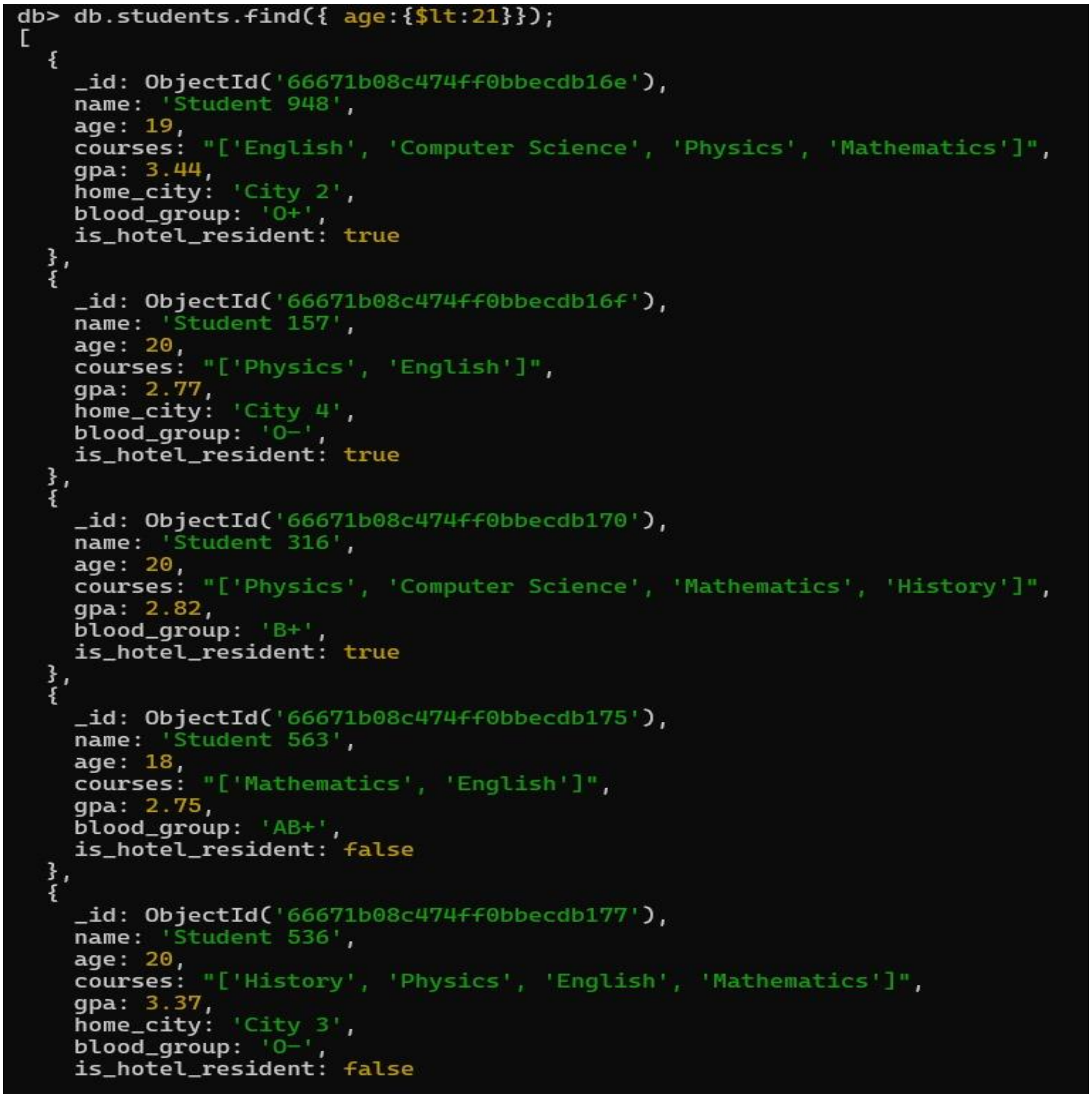
Selectors:

In MongoDB, selectors are used to define criteria for querying documents in a collection. They specify the conditions that documents must meet to be included in the query results. MongoDB provides various selectors to filter documents based on different criteria.

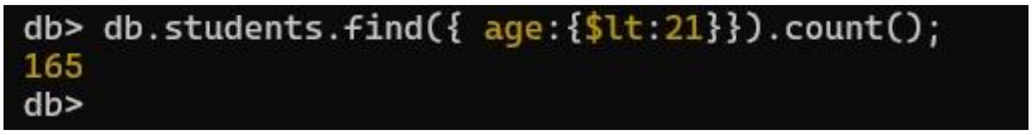
Comparison Selectors:

Used to match documents based on comparison operators such as greater than, less than, etc.

Example:



```
db> db.students.find({ age:{$lt:21}});
[
  {
    _id: ObjectId('66671b08c474ff0bbecdb16e'),
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb16f'),
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.77,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb170'),
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.82,
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb175'),
    name: 'Student 563',
    age: 18,
    courses: "['Mathematics', 'English']",
    gpa: 2.75,
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb177'),
    name: 'Student 536',
    age: 20,
    courses: "['History', 'Physics', 'English', 'Mathematics']",
    gpa: 3.37,
    home_city: 'City 3',
    blood_group: 'O-',
    is_hotel_resident: false
```

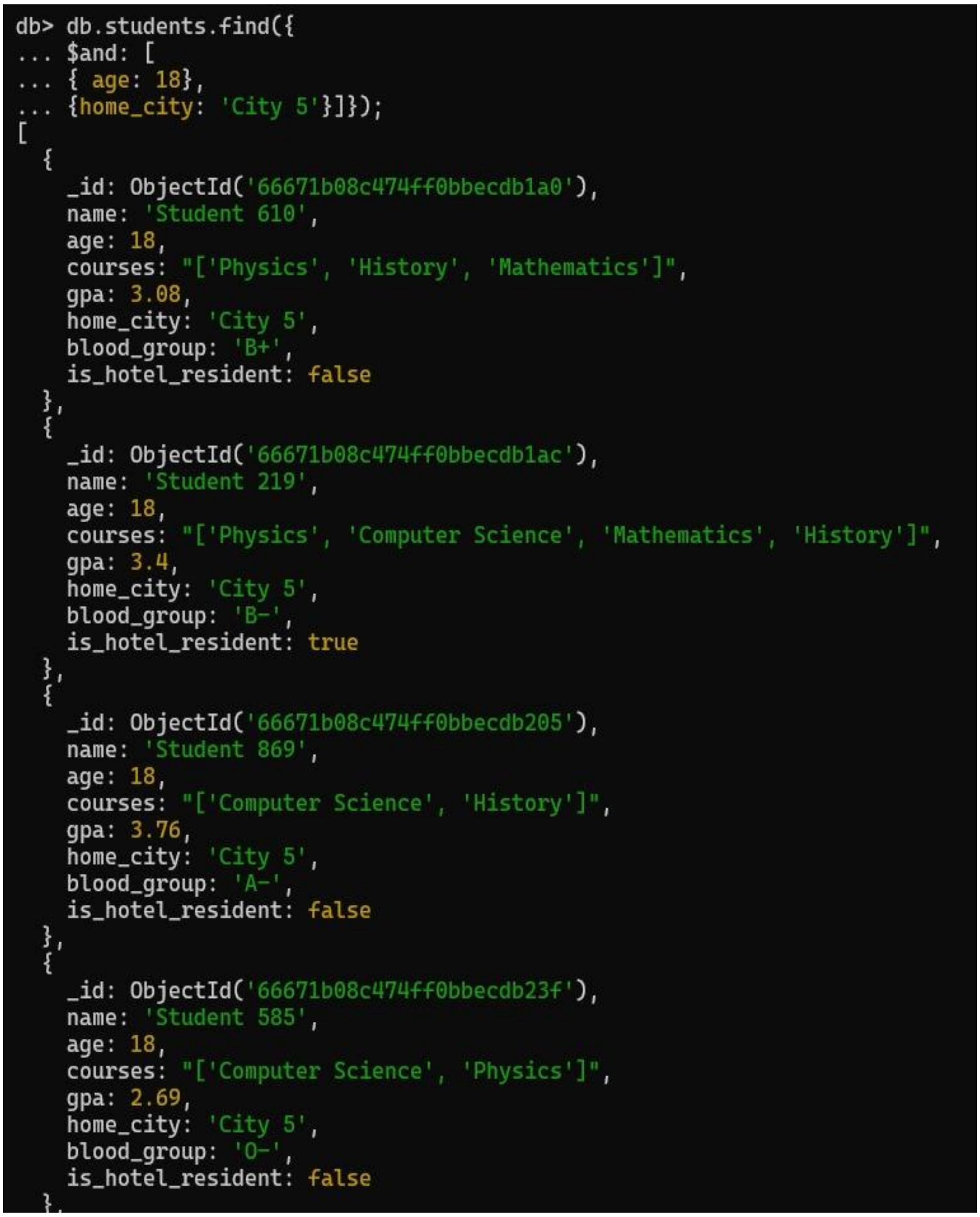


```
db> db.students.find({ age:{$lt:21}}).count();
165
db>
```

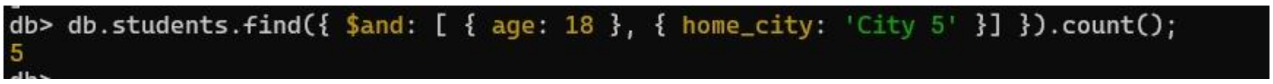
AND OPERATOR:

Example:

Finding students of age 18 from home_city – ‘City 5’.



```
db> db.students.find({
... $and: [
... { age: 18},
... {home_city: 'City 5'}]});
[
  {
    _id: ObjectId('66671b08c474ff0bbecdb1a0'),
    name: 'Student 610',
    age: 18,
    courses: "['Physics', 'History', 'Mathematics']",
    gpa: 3.08,
    home_city: 'City 5',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb1ac'),
    name: 'Student 219',
    age: 18,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 3.4,
    home_city: 'City 5',
    blood_group: 'B-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb205'),
    name: 'Student 869',
    age: 18,
    courses: "['Computer Science', 'History']",
    gpa: 3.76,
    home_city: 'City 5',
    blood_group: 'A-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb23f'),
    name: 'Student 585',
    age: 18,
    courses: "['Computer Science', 'Physics']",
    gpa: 2.69,
    home_city: 'City 5',
    blood_group: 'O-',
    is_hotel_resident: false
  },
```

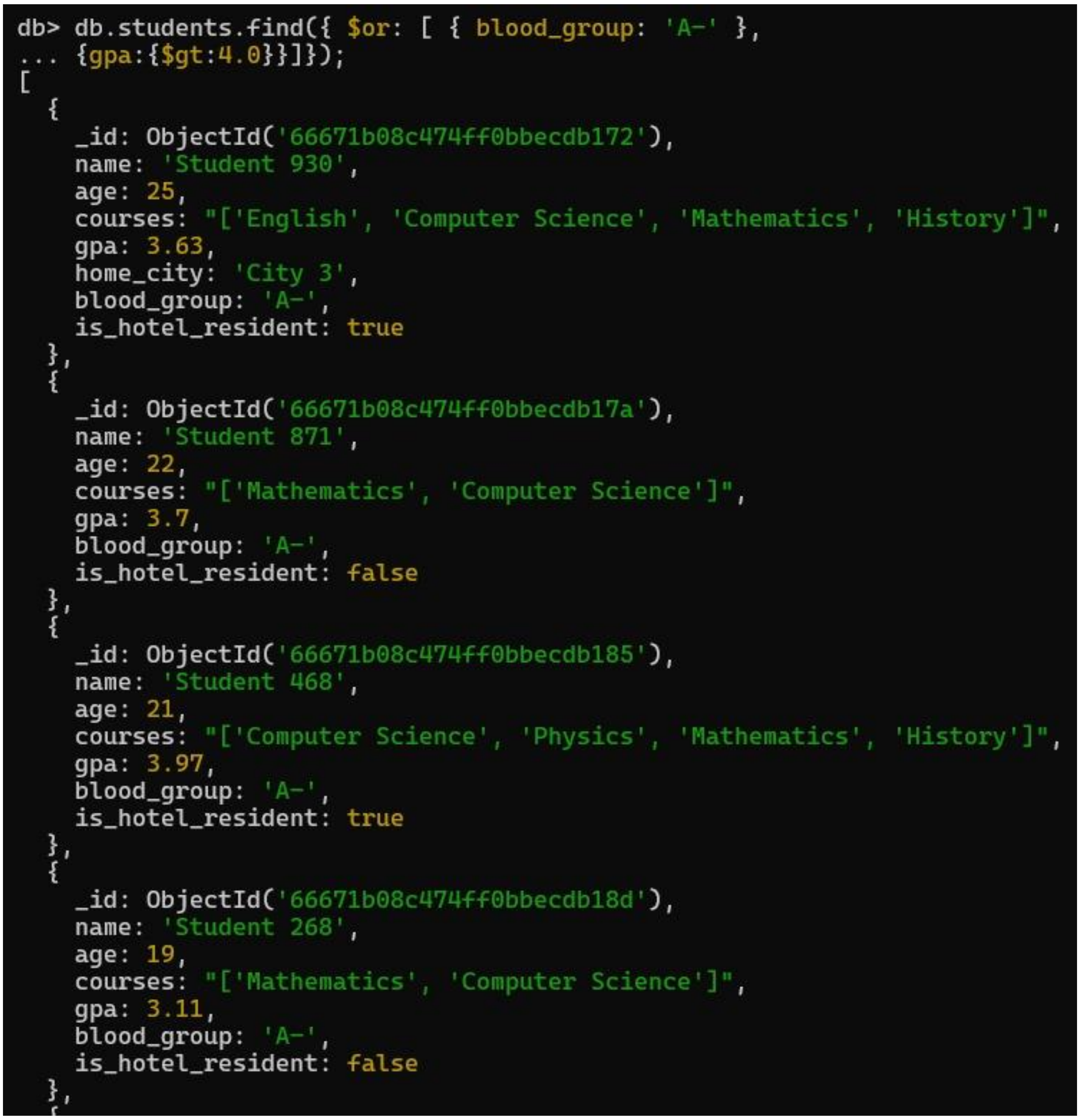


```
db> db.students.find({ $and: [ { age: 18 }, { home_city: 'City 5' }] }).count();
5
```

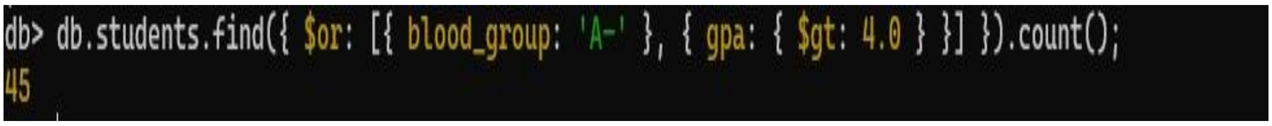
OR OPERATOR:

Example:

Finding students who have ‘A-‘ blood group or have gpa greater than 4.0



```
db> db.students.find({ $or: [ { blood_group: 'A-' },
... {gpa:{$gt:4.0}}]});
[
  {
    _id: ObjectId('66671b08c474ff0bbecdb172'),
    name: 'Student 930',
    age: 25,
    courses: "['English', 'Computer Science', 'Mathematics', 'History']",
    gpa: 3.63,
    home_city: 'City 3',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb17a'),
    name: 'Student 871',
    age: 22,
    courses: "['Mathematics', 'Computer Science']",
    gpa: 3.7,
    blood_group: 'A-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb185'),
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66671b08c474ff0bbecdb18d'),
    name: 'Student 268',
    age: 19,
    courses: "['Mathematics', 'Computer Science']",
    gpa: 3.11,
    blood_group: 'A-',
    is_hotel_resident: false
  },
```



```
db> db.students.find({ $or: [{ blood_group: 'A-' }, { gpa: { $gt: 4.0 } }] }).count();
45
```