Class 4:

Projection, Limit & Selectors

Projection:

In MongoDB, projection means selecting only the necessary data rather than selecting whole of the data of a document. If a document has 5 fields and you need to show only 3, then select only 3 fields from them. MongoDB projection uses the same find syntax, but we also add a set of parameters to the find function. This set of parameters informs the MongoDB instance of which fields to be returned.

MongoDB projection method positively impacts database performance as it reduces the workload of the find query when trying to retrieve specific data from a document, minimizing resource usage.

To enhance the querying and reduce the workload, multiple operators can be used within a projection query like the ones below:

- $
- $elemMatch
- $slice

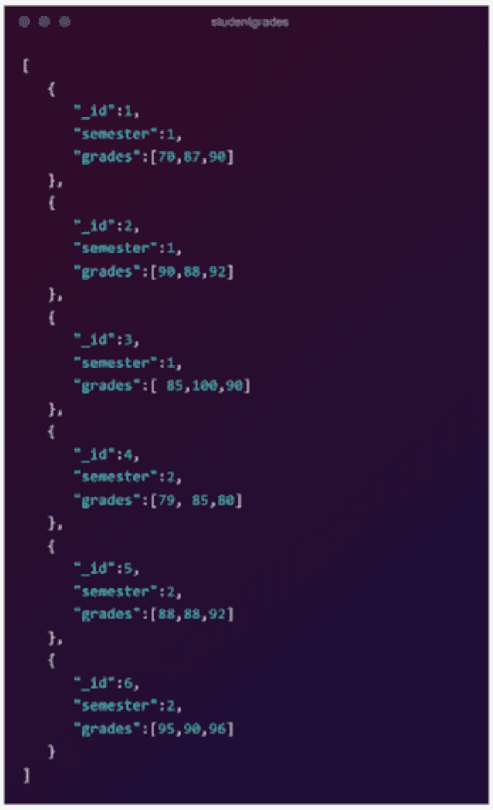
1. $ Operator:

The $ operator is utilized in scenarios where there is a need to limit an array to project only the first element that matches the condition of the query, if no query condition is present, the first element is returned in the specified MongoDB projection array, and a sample syntax of the same can be seen below:

```
db.collection.find( { <array>: <condition> ... }, { "<array>.$": 1 } )
```

Example:

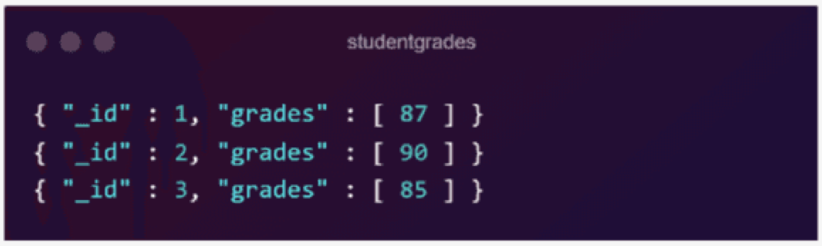
Let us check out the "students" Collection this time to understand the functioning of the $ operator.



```
[
   {
      "_id":1,
      "semester":1,
      "grades":[70,87,90]
   },
   {
      "_id":2,
      "semester":1,
      "grades":[90,88,92]
   },
   {
      "_id":3,
      "semester":1,
      "grades":[ 85,100,90]
   },
   {
      "_id":4,
      "semester":2,
      "grades":[79, 85,80]
   },
   {
      "_id":5,
      "semester":2,
      "grades":[88,88,92]
   },
   {
      "_id":6,
      "semester":2,
      "grades":[95,90,96]
   }
]
```

```
db.students.find( { semester: 1, grades: { $gte: 85 } }, { "grades.$": 1 } )
```

Using the $ operator will only return the MongoDB projection first element of array where it is equal to or greater than 85.



```
{ "_id" : 1, "grades" : [ 87 ] }
{ "_id" : 2, "grades" : [ 90 ] }
{ "_id" : 3, "grades" : [ 85 ] }
```

Limitations of the $ operator:

- In a single MongoDB projection query, only a single $ operator can be used.
- It will only be applied to a single condition on the array field
- The sort() function in the find() is applied before the $ operator and this function may cause the sort order to get distorted.

2. $elemMatch Operator:

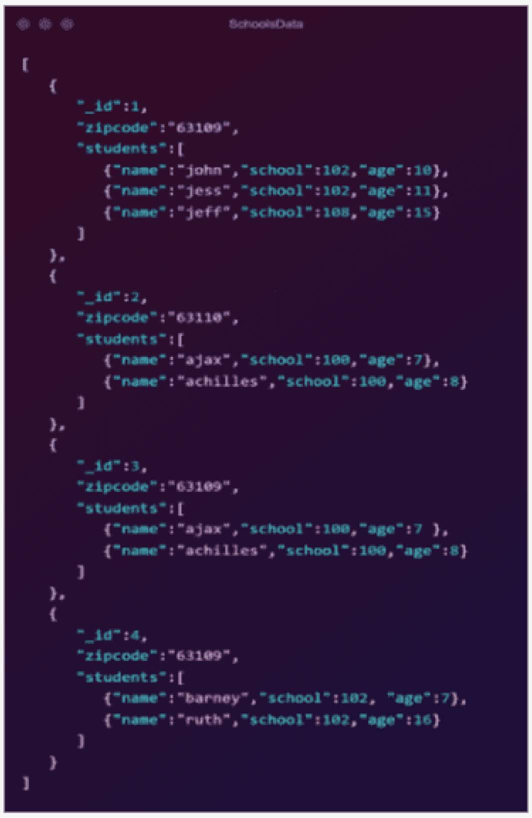
Similar to the $ operator, the $elemMatch operator also limits the contents of an array to the first element that fits the given constraint. Though, there is a minor difference from the $ operator because the $elemMatch projection operator needs an explicit condition argument.

Syntax as follows:

```
db.collection.find( { <array>: <condition> ... }, { "<array>.$elemMatch": ($elemMatch operator) } )
```

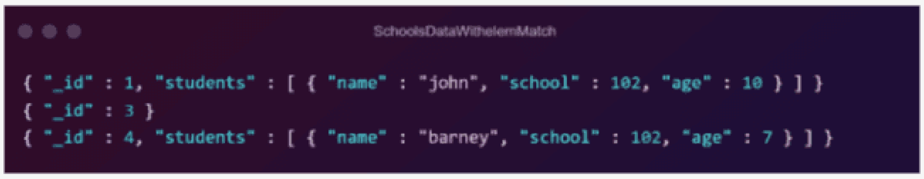
Example:

Below you can check the "schoolsdData" collection that we will use to demonstrate the $elemMatch operator.



```
[
   {
      "_id":1,
      "zipcode":"63109",
      "students":[
         {"name":"john","school":102,"age":10},
         {"name":"jess","school":102,"age":11},
         {"name":"jeff","school":108,"age":15}
      ]
   },
   {
      "_id":2,
      "zipcode":"63110",
      "students":[
         {"name":"ajax","school":100,"age":7},
         {"name":"achilles","school":100,"age":8}
      ]
   },
   {
      "_id":3,
      "zipcode":"63109",
      "students":[
         {"name":"ajax","school":100,"age":7 },
         {"name":"achilles","school":100,"age":8}
      ]
   },
   {
      "_id":4,
      "zipcode":"63109",
      "students":[
         {"name":"barney","school":102, "age":7},
         {"name":"ruth","school":102,"age":16}
      ]
   }
]
```

```
db.schools.find( { zipcode: "63109" }, { students: { $elemMatch: { school: 102 } } } )
```



```
{ "_id" : 1, "students" : [ { "name" : "john", "school" : 102, "age" : 10 } ] }
{ "_id" : 3 }
{ "_id" : 4, "students" : [ { "name" : "barney", "school" : 102, "age" : 7 } ] }
```

Limitations of $elemMatch operator are as follows:

- The field to which the $elemMatch projection is applied is returned as the last field of the document, regardless of the order in which the fields are ordered.
- $elemMatch projection operator is not supported by find() operations on MongoDB views.
- The $elemMatch operator does not handle $text query expressions.

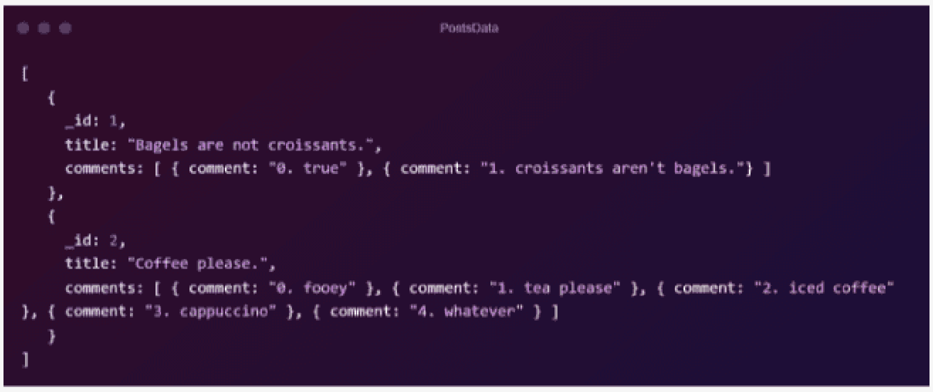
3. $slice Projection Operator:

The $slice operator bounds the number of elements that should be returned as the output of a MongoDB projection query.

Syntax:

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

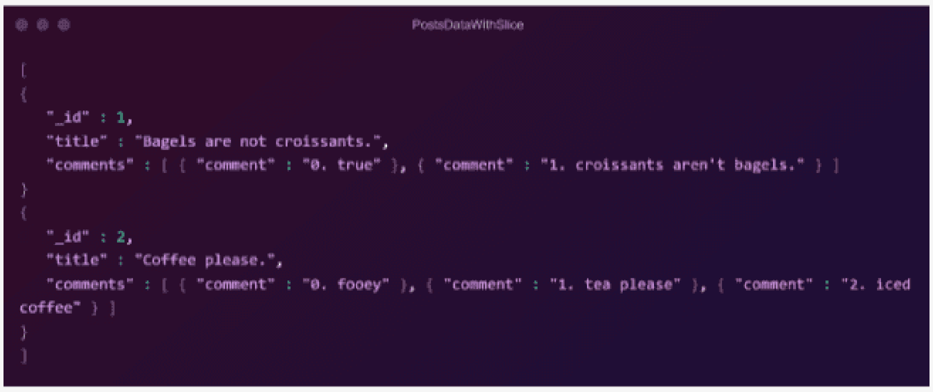
Now we'll use the "postsData" collection to demonstrate the $slice operator.



```
[
  {
    _id: 1,
    title: "Bagels are not croissants.",
    comments: [ { comment: "0. true" }, { comment: "1. croissants aren't bagels."} ]
  },
  {
    _id: 2,
    title: "Coffee please.",
    comments: [ { comment: "0. fooey" }, { comment: "1. tea please" }, { comment: "2. iced coffee"
}, { comment: "3. cappuccino" }, { comment: "4. whatever" } ]
  }
]
```

We want to get data from the comments array to return the array with its first three elements. If the array contains less than three elements, all elements of the array are returned.

```
db.posts.find( {}, { comments: { $slice: 3 } } )
```



```
[
{
   "_id" : 1,
   "title" : "Bagels are not croissants.",
   "comments" : [ { "comment" : "0. true" }, { "comment" : "1. croissants aren't bagels." } ]
}
{
   "_id" : 2,
   "title" : "Coffee please.",
   "comments" : [ { "comment" : "0. fooey" }, { "comment" : "1. tea please" }, { "comment" : "2. iced
coffee" } ]
}
]
```

Limitations in $slice operator:

- In a nested array the **$slice** operator will only return the sliced element and will not return any other item, especially from MongoDB 4.4.
- The find() action is not supported by the $slice operator on MongoDB views.
- Due to a constraint imposed by MongoDB, the $slice operator cannot be combined with the $ projection operator. This is because top-level fields are not permitted to contain the $ symbol as part of the field name.

Limit:

In MongoDB, a limit is a query parameter that specifies the maximum number of documents to return in a query result set. It is similar to the LIMIT clause in SQL. The limit parameter can be used in conjunction with other query parameters, such as sorting and skipping. In MongoDB, several types of limits may apply, depending on the context

Size limit:

MongoDB has a maximum document size of 16 megabytes (MB). If a document exceeds this limit, it cannot be stored in MongoDB. Similarly, a MongoDB database cannot exceed 64 terabytes (TB) in size.

Index limit:

MongoDB imposes a limit on the number of indexes that can be created for a single collection. The exact limit depends on the size of the indexes and the available memory on the server.

Cursor limit:

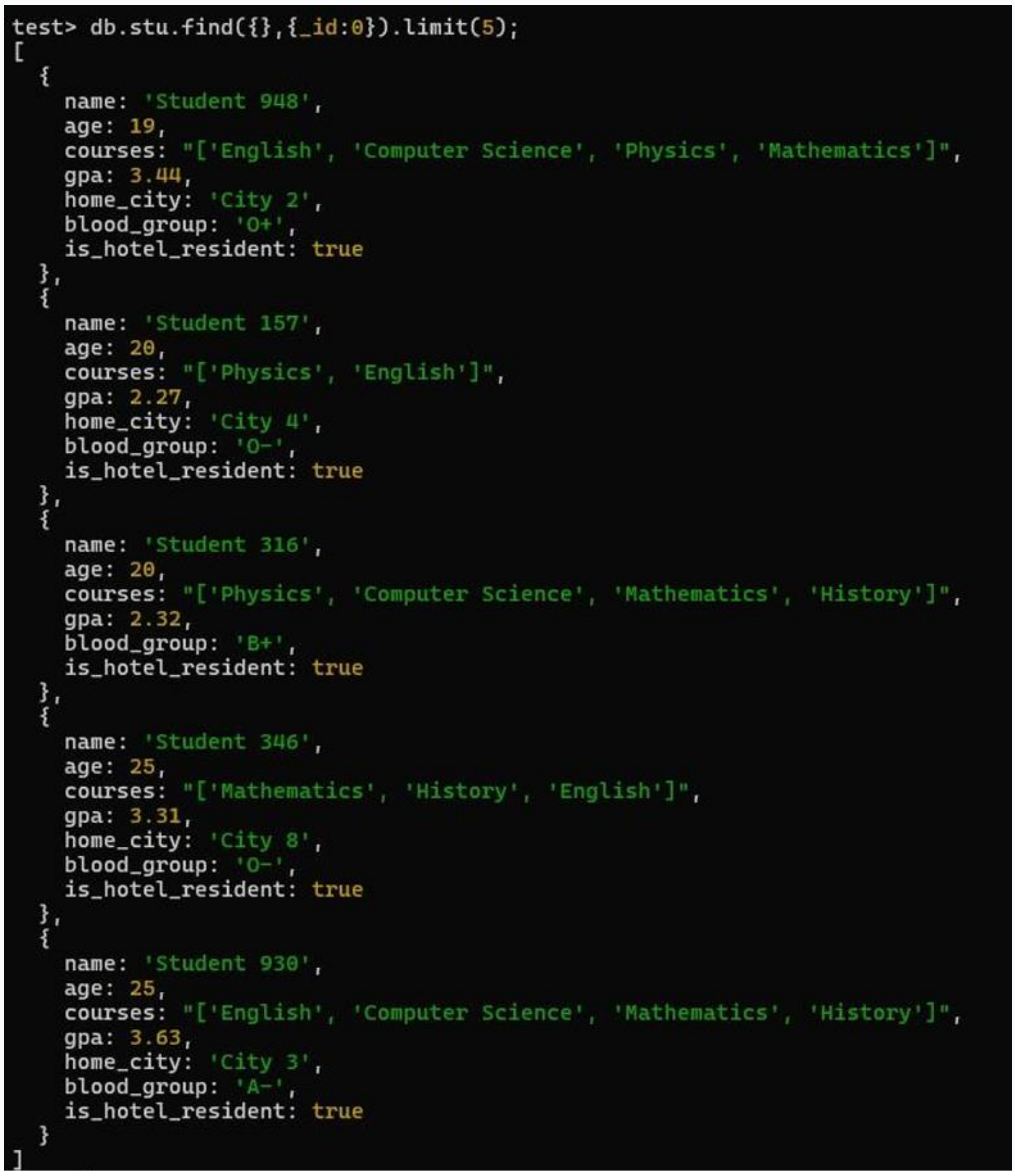
When querying large amounts of data, MongoDB may limit the number of documents that can be returned in a single query. This limit is controlled by the batchSize parameter and can be adjusted to balance performance and resource usage.

Connection limit:

MongoDB imposes a limit on the number of concurrent connections that can be made to a single MongoDB instance. This limit depends on the available memory and CPU resources of the server.

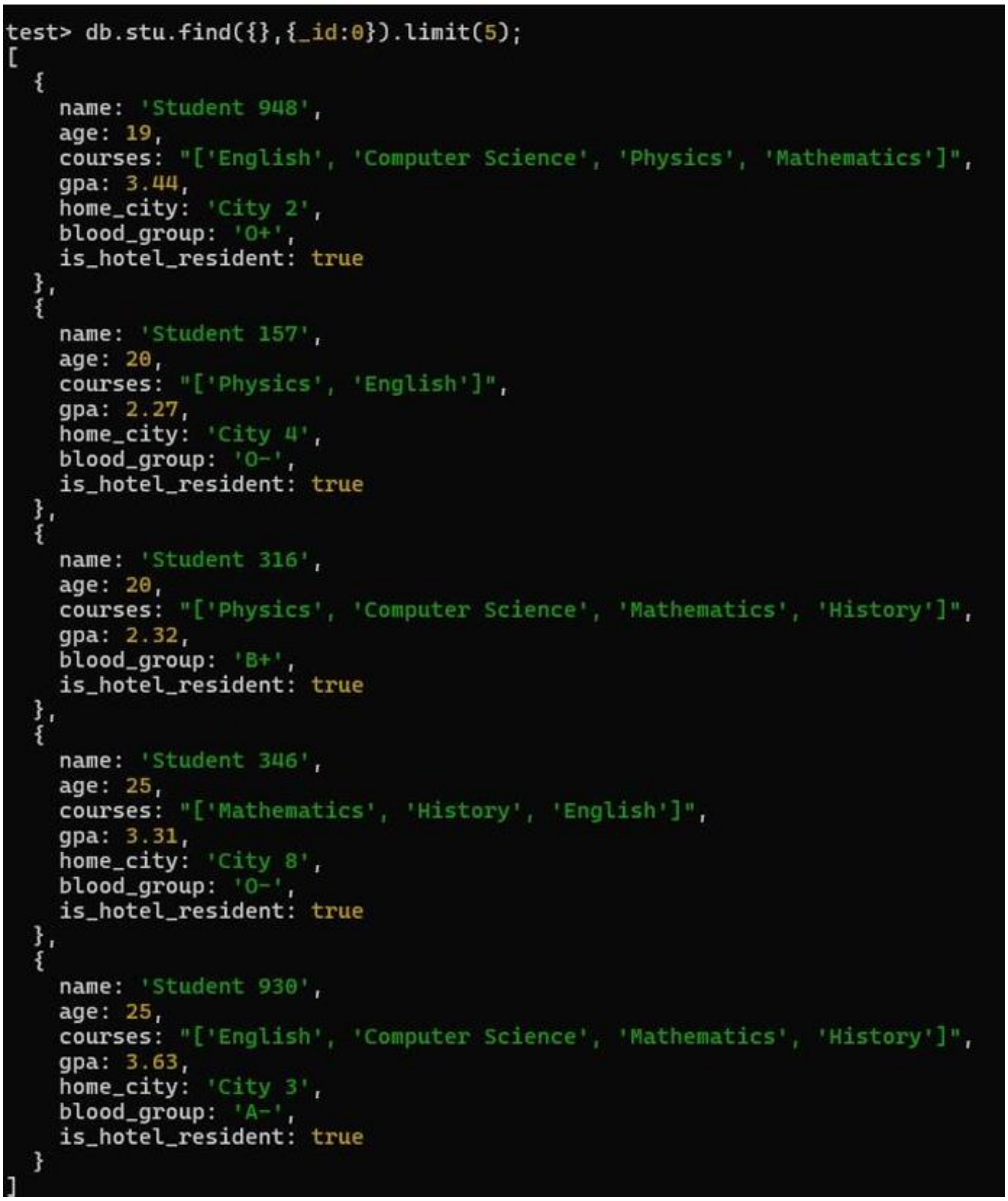
Collection and database limit:

There is no hard limit on the number of collections or databases that can be created in MongoDB, but the practical limit depends on the available resources and the workload of the server.



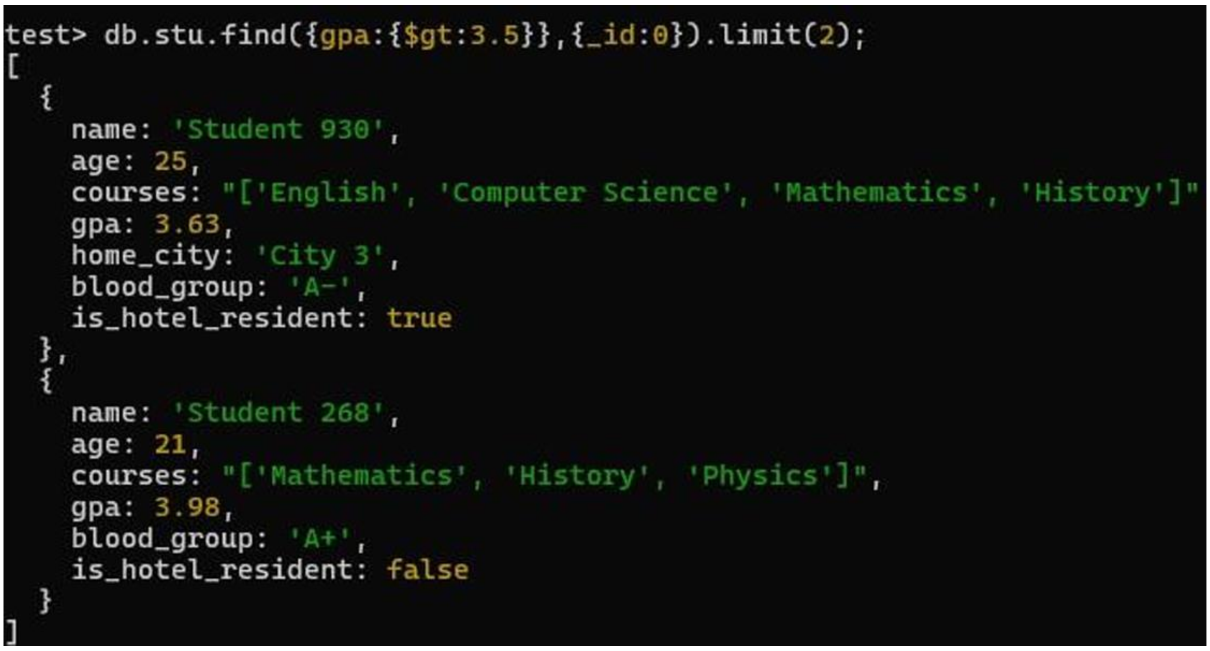
```
test> db.stu.find({},{_id:0}).limit(5);
[
  {
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.32,
    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
  },
  {
    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
  }
]
```

Getting first five document:



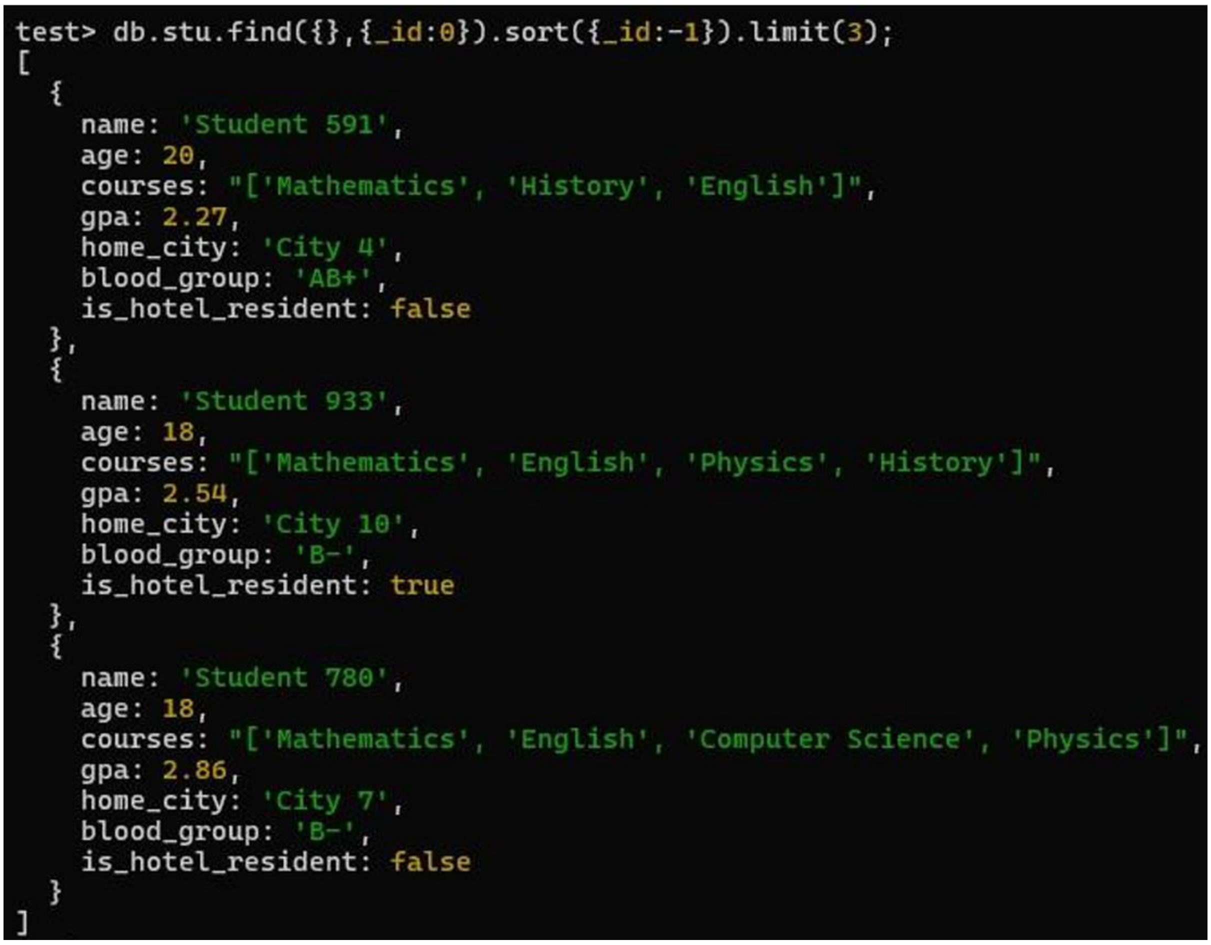
```
test> db.stu.find({},{_id:0}).limit(5);
[
  {
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.32,
    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
  },
  {
    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
  }
]
```

Limiting results :



```
test> db.stu.find({gpa:{$gt:3.5}},{_id:0}).limit(2);
[
  {
    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
  },
  {
    name: 'Student 268',
    age: 21,
    courses: "['Mathematics', 'History', 'Physics']",
    gpa: 3.98,
    blood_group: 'A+',
    is_hotel_resident: false
  }
]
```

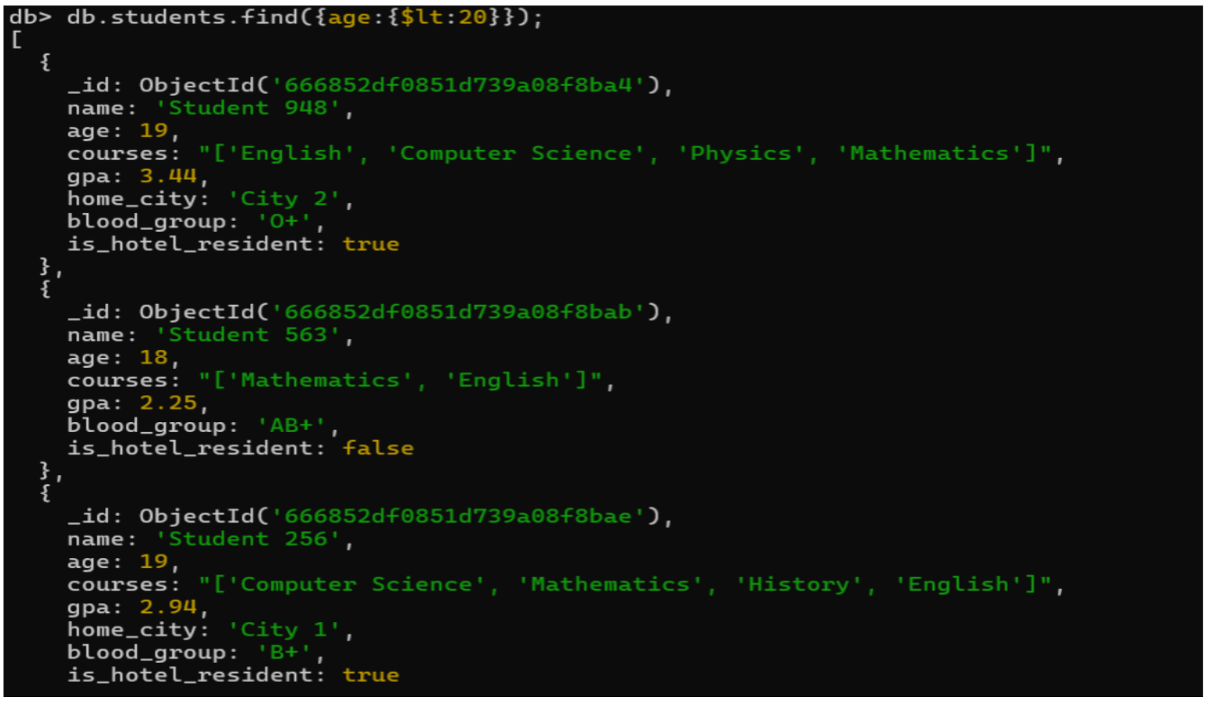
Top ten results:



```
test> db.stu.find({},{_id:0}).sort({_id:-1}).limit(3);
[
  {
    name: 'Student 591',
    age: 20,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    name: 'Student 933',
    age: 18,
    courses: "['Mathematics', 'English', 'Physics', 'History']",
    gpa: 2.54,
    home_city: 'City 10',
    blood_group: 'B-',
    is_hotel_resident: true
  },
  {
    name: 'Student 780',
    age: 18,
    courses: "['Mathematics', 'English', 'Computer Science', 'Physics']",
    gpa: 2.86,
    home_city: 'City 7',
    blood_group: 'B-',
    is_hotel_resident: false
  }
]
```

Selectors:

Comparing the greater than and less than operators by using the operators such as $gt,$lt. In the below example we got data of the students age less than 20.



```
db> db.students.find({age:{$lt:20}});
[
  {
    _id: ObjectId('666852df0851d739a08f8ba4'),
    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('666852df0851d739a08f8bab'),
    name: 'Student 563',
    age: 18,
    courses: "['Mathematics', 'English']",
    gpa: 2.25,
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('666852df0851d739a08f8bae'),
    name: 'Student 256',
    age: 19,
    courses: "['Computer Science', 'Mathematics', 'History', 'English']",
    gpa: 2.94,
    home_city: 'City 1',
    blood_group: 'B+',
    is_hotel_resident: true
```

GEOSPATIAL:

To perform a geospatial query, create a query filter with a field name and a geospatial query operator. In MongoDB, you can store geospatial data as GeoJSON objects or as legacy coordinate pairs.

GeoJSON Objects:

To calculate geometry over an Earth-like sphere, store your location data as GeoJSON objects.

To specify GeoJSON data, use an embedded document with:

● a field named type that specifies the GeoJSON object type.

● a field named coordinates that specifies the object's coordinates.