Experiment:2

Projection

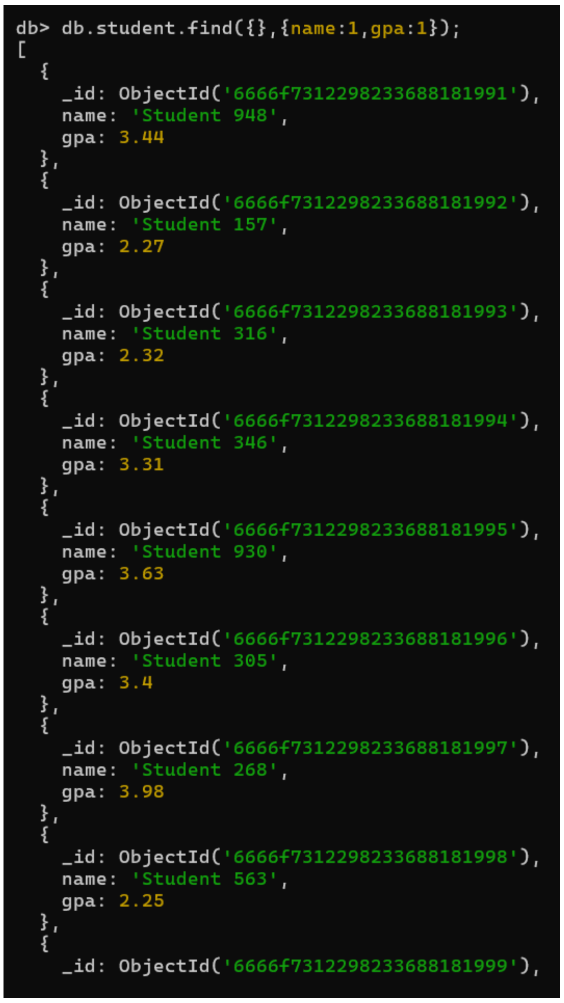
```
// Get only the name and gpa for all students
db.students.find({}, { name: 1, gpa: 1 });

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

`db.students.find({}, { name: 1, gpa: 1 })` - This line uses the `find` method of the `students` collection to retrieve documents from the database.

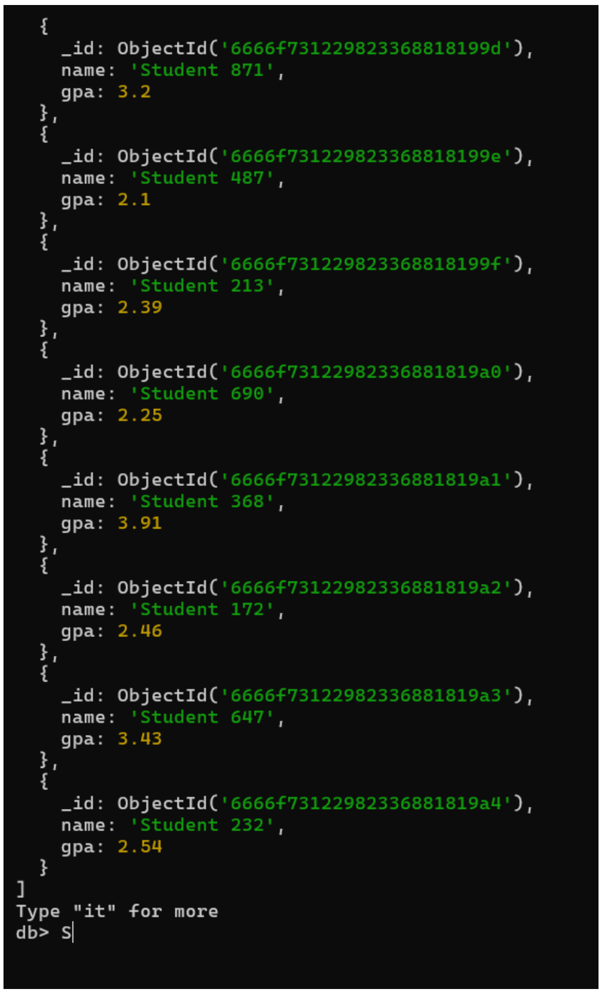
- `db.students`: This refers to the `students` collection within the MongoDB database.
- `.find({})`: The `.find` method is used to find documents in a collection. The first argument, `{}` in this case, is an empty filter document, which means it will retrieve all documents in the collection.
- `{ name: 1, gpa: 1 }`: This is the projection document that specifies which fields to include in the retrieved documents.
  - `name: 1`: Includes the `name` field in the results.
  - `gpa: 1`: Includes the `gpa` field in the results.

Output :

In essence, this code snippet retrieves all documents from the `students` collection and projects only the `name` and `gpa` fields for each document. This means the query will return an array of documents where each document only contains the `name` and `gpa` fields.



```
db> db.student.find({},{name:1,gpa:1});
[
  {
    _id: ObjectId('6666f7312298233688181991'),
    name: 'Student 948',
    gpa: 3.44
  },
  {
    _id: ObjectId('6666f7312298233688181992'),
    name: 'Student 157',
    gpa: 2.27
  },
  {
    _id: ObjectId('6666f7312298233688181993'),
    name: 'Student 316',
    gpa: 2.32
  },
  {
    _id: ObjectId('6666f7312298233688181994'),
    name: 'Student 346',
    gpa: 3.31
  },
  {
    _id: ObjectId('6666f7312298233688181995'),
    name: 'Student 930',
    gpa: 3.63
  },
  {
    _id: ObjectId('6666f7312298233688181996'),
    name: 'Student 305',
    gpa: 3.4
  },
  {
    _id: ObjectId('6666f7312298233688181997'),
    name: 'Student 268',
    gpa: 3.98
  },
  {
    _id: ObjectId('6666f7312298233688181998'),
    name: 'Student 563',
    gpa: 2.25
  },
  {
    _id: ObjectId('6666f7312298233688181999'),
```



```
  {
    _id: ObjectId('6666f731229823368818199d'),
    name: 'Student 871',
    gpa: 3.2
  },
  {
    _id: ObjectId('6666f731229823368818199e'),
    name: 'Student 487',
    gpa: 2.1
  },
  {
    _id: ObjectId('6666f731229823368818199f'),
    name: 'Student 213',
    gpa: 2.39
  },
  {
    _id: ObjectId('6666f73122982336881819a0'),
    name: 'Student 690',
    gpa: 2.25
  },
  {
    _id: ObjectId('6666f73122982336881819a1'),
    name: 'Student 368',
    gpa: 3.91
  },
  {
    _id: ObjectId('6666f73122982336881819a2'),
    name: 'Student 172',
    gpa: 2.46
  },
  {
    _id: ObjectId('6666f73122982336881819a3'),
    name: 'Student 647',
    gpa: 3.43
  },
  {
    _id: ObjectId('6666f73122982336881819a4'),
    name: 'Student 232',
    gpa: 2.54
  }
]
Type "it" for more
db> S
```

.

Get Selected Attributes

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

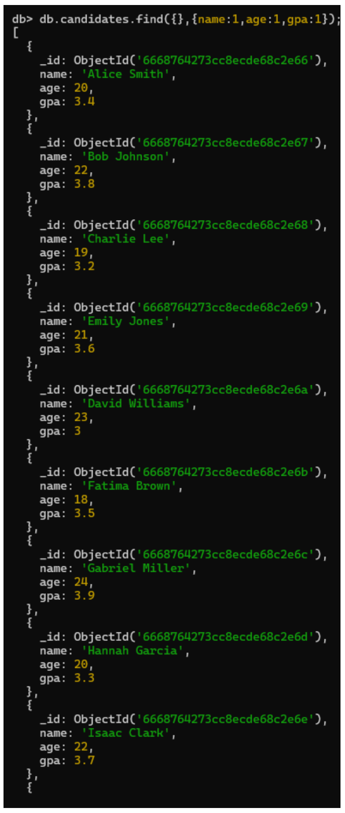
db.students.find({}, { name: 1, age: 1 }) : This line uses the `find` method of the `students` collection to retrieve documents from the database.

- **db.students** : This refers to the `students` collection within the MongoDB database.
- **.find({})** : The `.find` method is used to find documents in a collection. The first argument, `{}` in this case, is an empty filter document, which means it will retrieve all documents in the collection.
- **{ name: 1, age: 1 }** : This is the projection document that specifies which fields to include in the retrieved documents.
  - **name: 1** : Includes the `name` field in the results.

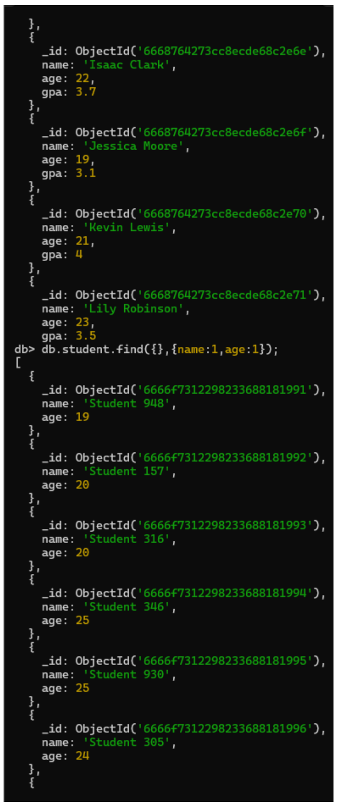
- **age: 1** : Includes the `age` field in the results.

Output :

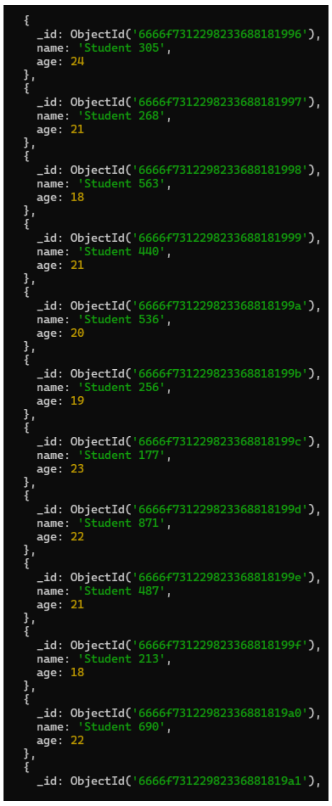
This code retrieves all documents from the `students` collection and projects only the `name` and `age` fields for each document. This means the query will return an array of documents where each document only contains the `name` and `age` fields.



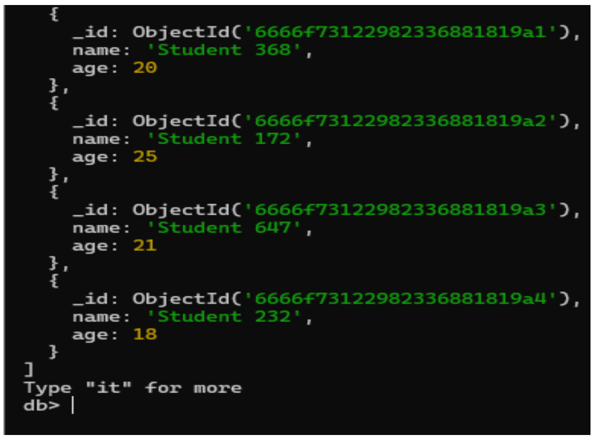
```
db> db.candidates.find({},{name:1,age:1,gpa:1});
[
  {
    _id: ObjectId('6668764273cc8ecde68c2e66'),
    name: 'Alice Smith',
    age: 20,
    gpa: 3.4
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e67'),
    name: 'Bob Johnson',
    age: 22,
    gpa: 3.8
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e68'),
    name: 'Charlie Lee',
    age: 19,
    gpa: 3.2
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e69'),
    name: 'Emily Jones',
    age: 21,
    gpa: 3.6
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6a'),
    name: 'David Williams',
    age: 23,
    gpa: 3
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6b'),
    name: 'Fatima Brown',
    age: 18,
    gpa: 3.5
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6c'),
    name: 'Gabriel Miller',
    age: 24,
    gpa: 3.9
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6d'),
    name: 'Hannah Garcia',
    age: 20,
    gpa: 3.3
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6e'),
    name: 'Isaac Clark',
    age: 22,
    gpa: 3.7
  },
  {
```



```
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6e'),
    name: 'Isaac Clark',
    age: 22,
    gpa: 3.7
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6f'),
    name: 'Jessica Moore',
    age: 19,
    gpa: 3.1
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e70'),
    name: 'Kevin Lewis',
    age: 21,
    gpa: 4
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e71'),
    name: 'Lily Robinson',
    age: 23,
    gpa: 3.5
db> db.student.find({},{name:1,age:1});
[
  {
    _id: ObjectId('6666f7312298233688181991'),
    name: 'Student 948',
    age: 19
  },
  {
    _id: ObjectId('6666f7312298233688181992'),
    name: 'Student 157',
    age: 20
  },
  {
    _id: ObjectId('6666f7312298233688181993'),
    name: 'Student 316',
    age: 20
  },
  {
    _id: ObjectId('6666f7312298233688181994'),
    name: 'Student 346',
    age: 25
  },
  {
    _id: ObjectId('6666f7312298233688181995'),
    name: 'Student 930',
    age: 25
  },
  {
    _id: ObjectId('6666f7312298233688181996'),
    name: 'Student 305',
    age: 24
  },
  {
```



```
  {
    _id: ObjectId('6666f7312298233688181996'),
    name: 'Student 305',
    age: 24
  },
  {
    _id: ObjectId('6666f7312298233688181997'),
    name: 'Student 268',
    age: 21
  },
  {
    _id: ObjectId('6666f7312298233688181998'),
    name: 'Student 563',
    age: 18
  },
  {
    _id: ObjectId('6666f7312298233688181999'),
    name: 'Student 440',
    age: 21
  },
  {
    _id: ObjectId('6666f731229823368818199a'),
    name: 'Student 536',
    age: 20
  },
  {
    _id: ObjectId('6666f731229823368818199b'),
    name: 'Student 256',
    age: 19
  },
  {
    _id: ObjectId('6666f731229823368818199c'),
    name: 'Student 177',
    age: 23
  },
  {
    _id: ObjectId('6666f731229823368818199d'),
    name: 'Student 871',
    age: 22
  },
  {
    _id: ObjectId('6666f731229823368818199e'),
    name: 'Student 487',
    age: 21
  },
  {
    _id: ObjectId('6666f731229823368818199f'),
    name: 'Student 213',
    age: 18
  },
  {
    _id: ObjectId('6666f73122982336881819a0'),
    name: 'Student 690',
    age: 22
  },
  {
    _id: ObjectId('6666f73122982336881819a1'),
```



```
  {
    _id: ObjectId('6666f73122982336881819a1'),
    name: 'Student 368',
    age: 20
  },
  {
    _id: ObjectId('6666f73122982336881819a2'),
    name: 'Student 172',
    age: 25
  },
  {
    _id: ObjectId('6666f73122982336881819a3'),
    name: 'Student 647',
    age: 21
  },
  {
    _id: ObjectId('6666f73122982336881819a4'),
    name: 'Student 232',
    age: 18
  }
]
Type "it" for more
db> |
```

Ignore Attributes

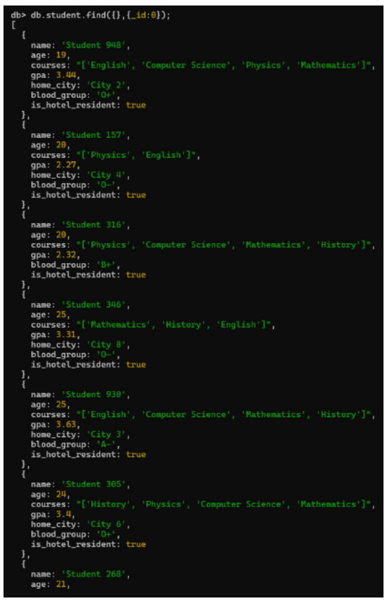
```
// Get all student data but exclude the _id field
db.students.find({}, { _id: 0 });
```

db.students.find({}, { id: 0 }) : This line uses the `find` method of the `students` collection to retrieve documents from the database.

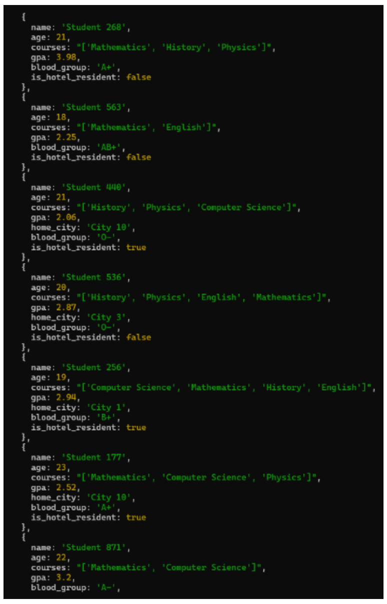
- **db.students** : This refers to the `students` collection within the MongoDB database.
- **.find({})** : The `.find` method is used to find documents in a collection. The first argument, `{}` in this case, is an empty filter document, which means it will retrieve all documents in the collection.
- **{ id: 0 }** : This is the projection document that specifies which fields to exclude in the retrieved documents.
    - **id: 0** : Excludes the `id` field from the results. Setting a field to `0` in the projection document tells MongoDB not to include that field in the retrieved documents.

Output :

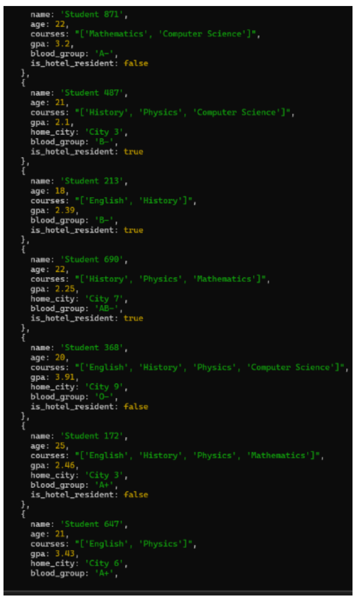
This code retrieves all documents from the `students` collection and excludes the `_id` field from the results. This means the query will return an array of documents where each document contains all its fields except for the `_id` field.



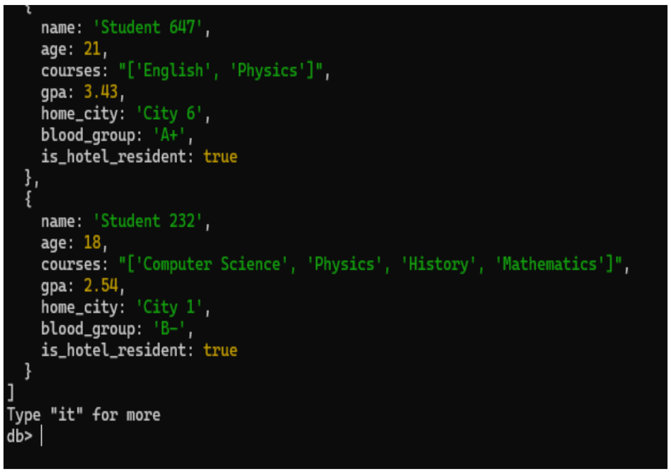
```
db> db.student.find({},{_id:0});
[
  {
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    name: 'Student 157',
    age: 20,
    courses: "['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
  },
  {
    name: 'Student 305',
    age: 24,
    courses: "['History', 'Physics', 'Computer Science', 'Mathematics']",
    gpa: 3.4,
    home_city: 'City 6',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    name: 'Student 268',
    age: 21,
```



```
  {
    name: 'Student 268',
    age: 21,
    courses: "['Mathematics', 'History', 'Physics']",
    gpa: 3.98,
    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 440',
    age: 21,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 2.06,
    home_city: 'City 10',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    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 256',
    age: 19,
    courses: "['Computer Science', 'Mathematics', 'History', 'English']",
    gpa: 2.94,
    home_city: 'City 1',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Student 177',
    age: 23,
    courses: "['Mathematics', 'Computer Science', 'Physics']",
    gpa: 2.52,
    home_city: 'City 10',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    name: 'Student 871',
    age: 22,
    courses: "['Mathematics', 'Computer Science']",
    gpa: 3.2,
    blood_group: 'A-',
```



```
    name: 'Student 871',
    age: 22,
    courses: "['Mathematics', 'Computer Science']",
    gpa: 3.2,
    blood_group: 'A-',
    is_hotel_resident: false
  },
  {
    name: 'Student 487',
    age: 21,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 2.1,
    home_city: 'City 3',
    blood_group: 'B-',
    is_hotel_resident: true
  },
  {
    name: 'Student 213',
    age: 18,
    courses: "['English', 'History']",
    gpa: 2.39,
    blood_group: 'B-',
    is_hotel_resident: true
  },
  {
    name: 'Student 690',
    age: 22,
    courses: "['History', 'Physics', 'Mathematics']",
    gpa: 2.25,
    home_city: 'City 7',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    name: 'Student 368',
    age: 20,
    courses: "['English', 'History', 'Physics', 'Computer Science']",
    gpa: 3.91,
    home_city: 'City 9',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Student 172',
    age: 25,
    courses: "['English', 'History', 'Physics', 'Mathematics']",
    gpa: 2.46,
    home_city: 'City 3',
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    name: 'Student 647',
    age: 21,
    courses: "['English', 'Physics']",
    gpa: 3.43,
    home_city: 'City 6',
    blood_group: 'A+',
```



```
    name: 'Student 647',
    age: 21,
    courses: "['English', 'Physics']",
    gpa: 3.43,
    home_city: 'City 6',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    name: 'Student 232',
    age: 18,
    courses: "['Computer Science', 'Physics', 'History', 'Mathematics']",
    gpa: 2.54,
    home_city: 'City 1',
    blood_group: 'B-',
    is_hotel_resident: true
  }
]
Type "it" for more
db> 
```

Retrieving Specific Fields from Nested Objects

```
// Get student name and only the first course from the courses array
db.students.find({}, {
  name: 1,
  courses: { $slice: 1 }
});
```

- **db.students.find({}, {** : This line uses the `find` method of the `students` collection to retrieve documents from the database.

  - **db.students** : This refers to the `students` collection within the MongoDB database.
  - **.find({})** : The `.find` method is used to find documents in a collection. The first argument, `{}` in this case, is an empty filter document, which means it will retrieve all documents in the collection.

- **{** : This curly brace opens the projection document which specifies which fields to include or exclude in the retrieved documents.

- **name: 1,** : This line includes the `name` field in the projection. Setting a field name to `1` (or `true`) in the projection document tells MongoDB to include that field in the retrieved documents.

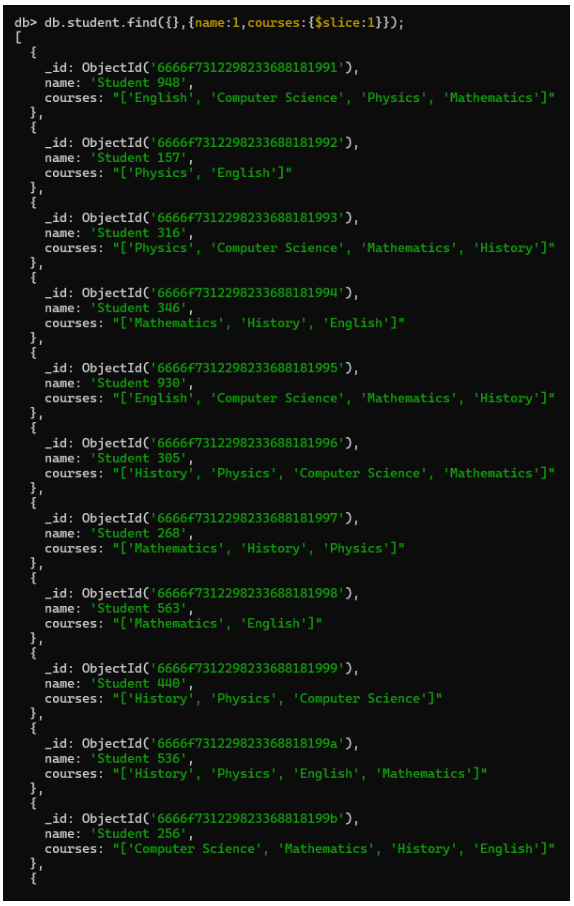
- **courses: { $slice: 1}** : This line uses the MongoDB aggregation operator `$slice` to limit the number of elements returned from the `courses` array to **1**. In other words, it will only include the first course from the `courses` array in the projection.

- **}** : This curly brace closes the projection document.

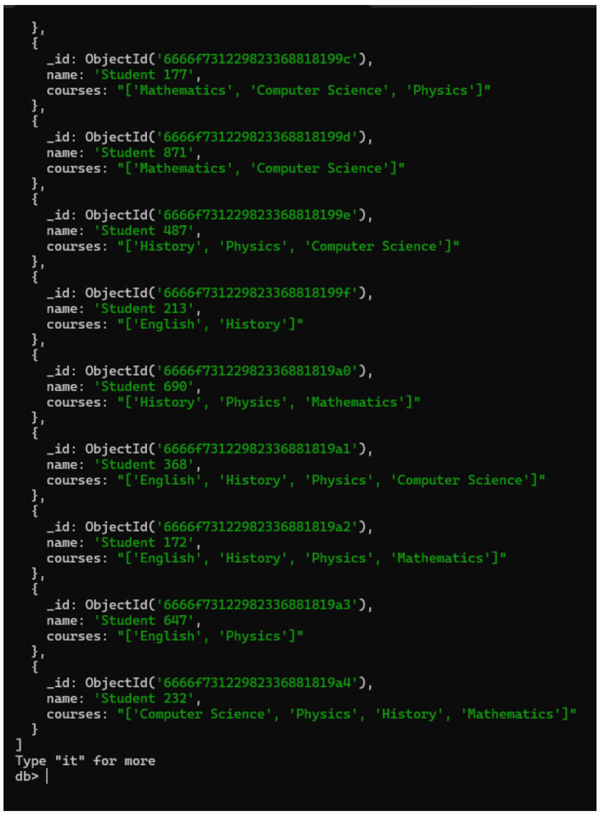
- **});** : This curly brace and the closing parenthesis close the `.find` method call.

Output :

This code retrieves all documents from the `students` collection and projects only the `name` field and the first element from the `courses` array for each student. This means the query result will return an array of documents where each document contains the student's name and only the first course from their enrolled courses.



```
db> db.student.find({},{name:1,courses:{$slice:1}});
[
  {
    _id: ObjectId('6666f7312298233688181991'),
    name: 'Student 948',
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']"
  },
  {
    _id: ObjectId('6666f7312298233688181992'),
    name: 'Student 157',
    courses: "['Physics', 'English']"
  },
  {
    _id: ObjectId('6666f7312298233688181993'),
    name: 'Student 316',
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']"
  },
  {
    _id: ObjectId('6666f7312298233688181994'),
    name: 'Student 346',
    courses: "['Mathematics', 'History', 'English']"
  },
  {
    _id: ObjectId('6666f7312298233688181995'),
    name: 'Student 930',
    courses: "['English', 'Computer Science', 'Mathematics', 'History']"
  },
  {
    _id: ObjectId('6666f7312298233688181996'),
    name: 'Student 305',
    courses: "['History', 'Physics', 'Computer Science', 'Mathematics']"
  },
  {
    _id: ObjectId('6666f7312298233688181997'),
    name: 'Student 268',
    courses: "['Mathematics', 'History', 'Physics']"
  },
  {
    _id: ObjectId('6666f7312298233688181998'),
    name: 'Student 563',
    courses: "['Mathematics', 'English']"
  },
  {
    _id: ObjectId('6666f7312298233688181999'),
    name: 'Student 440',
    courses: "['History', 'Physics', 'Computer Science']"
  },
  {
    _id: ObjectId('6666f731229823368818199a'),
    name: 'Student 536',
    courses: "['History', 'Physics', 'English', 'Mathematics']"
  },
  {
    _id: ObjectId('6666f731229823368818199b'),
    name: 'Student 256',
    courses: "['Computer Science', 'Mathematics', 'History', 'English']"
  },
  {
```



```
  },
  {
    _id: ObjectId('6666f731229823368818199c'),
    name: 'Student 177',
    courses: "['Mathematics', 'Computer Science', 'Physics']"
  },
  {
    _id: ObjectId('6666f731229823368818199d'),
    name: 'Student 871',
    courses: "['Mathematics', 'Computer Science']"
  },
  {
    _id: ObjectId('6666f731229823368818199e'),
    name: 'Student 487',
    courses: "['History', 'Physics', 'Computer Science']"
  },
  {
    _id: ObjectId('6666f731229823368818199f'),
    name: 'Student 213',
    courses: "['English', 'History']"
  },
  {
    _id: ObjectId('6666f73122982336881819a0'),
    name: 'Student 690',
    courses: "['History', 'Physics', 'Mathematics']"
  },
  {
    _id: ObjectId('6666f73122982336881819a1'),
    name: 'Student 368',
    courses: "['English', 'History', 'Physics', 'Computer Science']"
  },
  {
    _id: ObjectId('6666f73122982336881819a2'),
    name: 'Student 172',
    courses: "['English', 'History', 'Physics', 'Mathematics']"
  },
  {
    _id: ObjectId('6666f73122982336881819a3'),
    name: 'Student 647',
    courses: "['English', 'Physics']"
  },
  {
    _id: ObjectId('6666f73122982336881819a4'),
    name: 'Student 232',
    courses: "['Computer Science', 'Physics', 'History', 'Mathematics']"
  }
]
Type "it" for more
db> |
```

Benefits of Projection

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

2.Improves query performance by retrieving only necessary data.

3.Simplifies your code by focusing on the specific information you need.

LIMIT

1.The limit operator is used with the find method.
2.It's chained after the filter criteria or any sorting operations.
3.Syntax: db.collection.find({filter}, {projection}).limit(number)

Get First 5 document

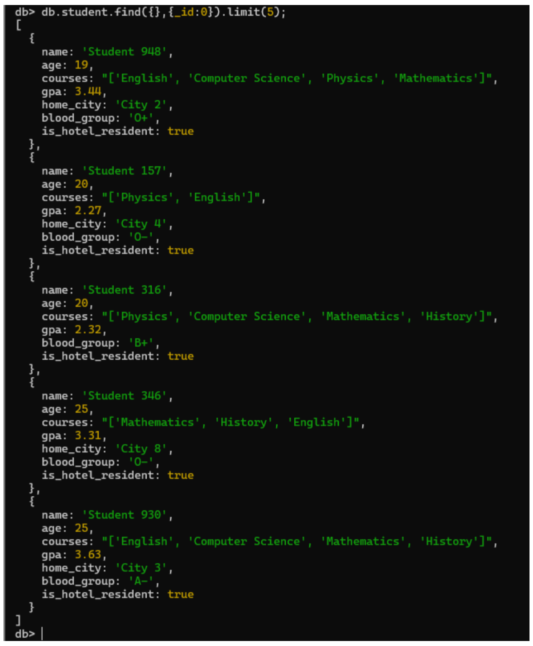
```
// Assuming you have already executed a query on the student collection
// Limit the results to the first 5 documents
db.students.find({}, { _id: 0 }).limit(5);
```

db.students.find({}, { id: 0 }).limit(5) : This line uses the `.find` method of the `students` collection to retrieve documents from the database and then uses the `.limit` method to limit the number of documents returned.

- **db.students.find({}, { id: 0 })** : This part is the same as the previous code snippets we discussed. It retrieves all documents from the `students` collection and excludes the `_id` field from the results using the projection document `{ id: 0 }`.
- **.limit(5)** : The `.limit` method limits the number of documents returned by the query to the specified value. In this case, `.limit(5)` limits the results to the first 5 documents retrieved from the `students` collection.

Output :

This code snippet assumes you've already executed a query on the `students` collection and then limits the results of that query to only the first 5 documents. The projection document `{ id: 0 }` also excludes the `_id` field from the final results.



```
db> db.student.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
  }
]
db> |
```

Limiting Results

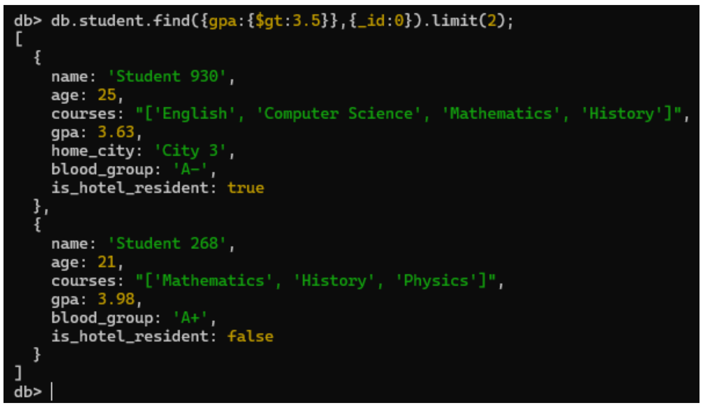
```
// Find all students with GPA greater than 3.5 and limit to 2 documents
db.students.find({ gpa: { $gt: 3.5 } }, { _id: 0 }).limit(2);
```

db.students.find({ gpa: { $gt: 3.5 } }, { id: 0 }).limit(2) : This line uses the `find` method of the `students` collection to retrieve documents from the database. It uses chaining to perform the find operation, then apply a projection, and then limit the results.

- **db.students.find({ gpa: { $gt: 3.5 } })** : This part finds all students where the `gpa` field is greater than 3.5.
- `db.students`: This refers to the `students` collection within the MongoDB database.
- `.find({ gpa: { $gt: 3.5 } })`: The `.find` method is used to find documents in a collection.
- `{ gpa: { $gt: 3.5 } }`: This is the filter document that specifies which documents to retrieve.
- `gpa`: The field to filter on.
- `{ $gt: 3.5 }`: The filter operator. In this case, `$gt` stands for "greater than" and 3.5 is the value to compare against. So this filters for documents where the `gpa` field is greater than 3.5.
- **{ id: 0 }** : This is the projection document that specifies which fields to exclude in the retrieved documents.
    - `id: 0` : Excludes the `_id` field from the results. Setting a field to `0` in the projection document tells MongoDB not to include that field in the retrieved documents.
- **.limit(2)** : The `.limit` method limits the number of documents returned by the query to the specified value. In this case, `.limit(2)` limits the results to the first 2 documents that match the filter criteria.

Output :

this code snippet retrieves documents from the `students` collection where the `gpa` field is greater than 3.5, excludes the `_id` field from the results and limits the final results to only the first 2 matching documents.



```
db> db.student.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
  }
]
db> |
```

To get Top 5 Results

```
// Sort documents in descending order by _id and limit to 5
db.students.find({}, { _id: 0 }).sort({ _id: -1 }).limit(5);
```

- **Student.find()** : This line uses the `find` method of the `Student` model, which is likely defined elsewhere in your code. The `Student` model presumably corresponds to the `students` collection in your MongoDB database. This line initiates the query to find documents from the `Student` collection.

- **.sort({ _id: -1 })** : This line sorts the results of the query based on the specified field.

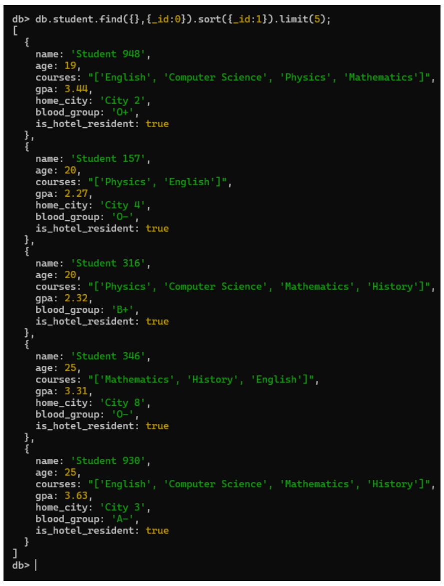
  - `.sort({ _id: -1 }):`
  - `.sort`: This method is used to sort the query results.
  - `{ _id: -1 }`: This is the sort document that specifies the field and order to sort by.
  - `_id`: The field to sort by. In this case, it's the `_id` field, which is the default unique identifier field in MongoDB documents.
  - `-1`: Sorts in descending order. A value of 1 would sort in ascending order.

- **.limit(5)** : This line limits the number of documents returned by the query to the specified value.

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

Output :

This code snippet retrieves documents from the `Student` collection, sorts them by their `_id` field in descending order (most recent first), and limits the final results to the first 5 documents.



```
db> db.student.find({},{_id:0}).sort({_id:1}).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
  }
]
db> |
```