PROJECTION , LIMIT AND SELECTORS

PROJECTION:

In MongoDB, projection is a way to select only the necessary data from a document, rather than retrieving the entire document. It allows you to specify which fields to include or exclude from the result set.

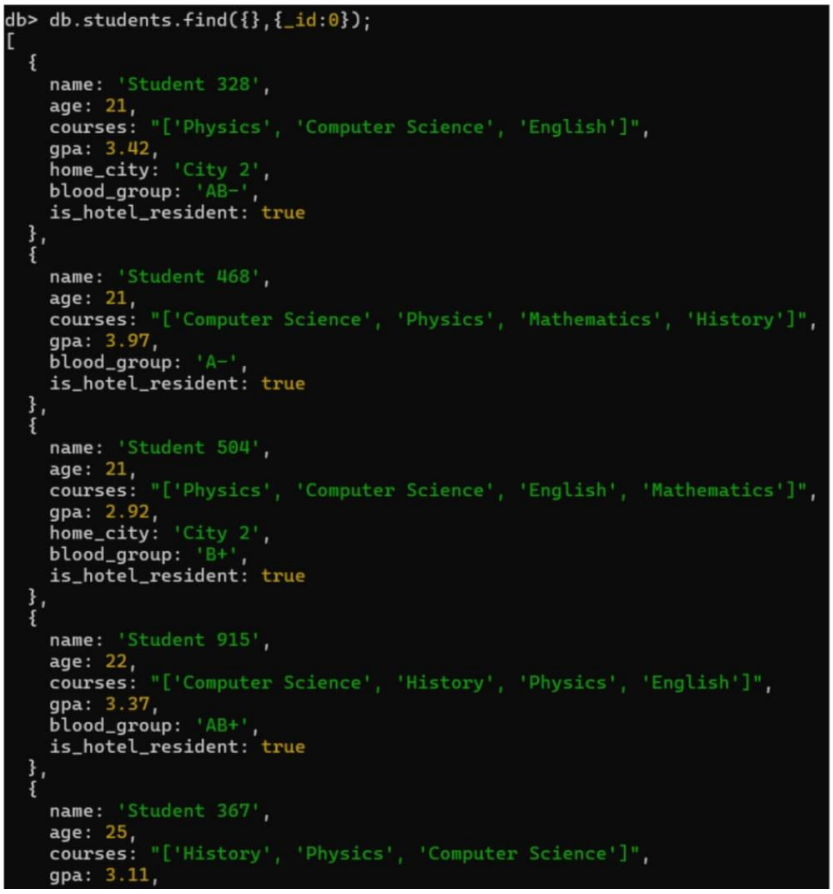
BENEFITS:

1. **Reduced Data Transfer**: Projection reduces the amount of data transferred over the network, which can lead to significant performance improvements, especially when working with large datasets.
2. **Improved Performance**: By only retrieving the necessary fields, MongoDB can process the data more efficiently, reducing the load on the server and improving overall performance.
3. **Simplified Data Processing**: Projection simplifies data processing by only returning the necessary data, making it easier to work with and manipulate the data.
4. **Reduced Memory Usage**: By only retrieving the necessary fields, MongoDB uses less memory to store the data, which can be especially important when working with large datasets.
5. **Improved Security**: Projection can help improve security by only returning the necessary fields, reducing the risk of sensitive data being exposed.

Get Selected Attributes:

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

It is used to retrieve all documents from the students collection while excluding the _id field from the results.



```
db> db.students.find({},{_id:0});
[
  {
    name: 'Student 328',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English']",
    gpa: 3.42,
    home_city: 'City 2',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Student 915',
    age: 22,
    courses: "['Computer Science', 'History', 'Physics', 'English']",
    gpa: 3.37,
    blood_group: 'AB+',
    is_hotel_resident: true
  },
  {
    name: 'Student 367',
    age: 25,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 3.11,
```

Retrieving specific fields from nested objects:

Retrieving specific fields from nested objects in MongoDB involves using the projection feature to specify the fields within the nested documents that you want to include or exclude.

db. Students. find({},{

name:1,

courses:{$slice:1}

});

```
db> db.students.find({},{name:1, courses:{$slice:1}});
[
  {
    _id: ObjectId('665609bde575a33e43e69479'),
    name: 'Student 328',
    courses: "['Physics', 'Computer Science', 'English']"
  },
  {
    _id: ObjectId('665609bde575a33e43e6947c'),
    name: 'Student 468',
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']"
  },
  {
    _id: ObjectId('665609bde575a33e43e6947d'),
    name: 'Student 504',
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']"
  },
  {
    _id: ObjectId('665609bde575a33e43e6947f'),
    name: 'Student 915',
    courses: "['Computer Science', 'History', 'Physics', 'English']"
  },
  {
    _id: ObjectId('665609bde575a33e43e69483'),
    name: 'Student 367',
    courses: "['History', 'Physics', 'Computer Science']"
  },
  {
    _id: ObjectId('665609bde575a33e43e69488'),
    name: 'Student 969',
    courses: "['History', 'Mathematics', 'Physics', 'English']"
  },
  {
    _id: ObjectId('665609bde575a33e43e69489'),
    name: 'Student 502',
    courses: "['Computer Science', 'Mathematics', 'English', 'Physics']"
  },
  {
    _id: ObjectId('665609bde575a33e43e6948a'),
```

Ignore attributes:

In MongoDB, ignoring an attribute (or field) in query results is done using projection. By setting the value of a field to 0 in the projection document, you can exclude that field from the returned documents.

```
test> db.stu.find({},{name:1,age:1}).count();
500
test>
```

LIMIT:

Getting first five documents:

```
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:

Syntax: db. collection. find(<query criteria>).limit(<number>)

db. students. find({ gpa :{$gt:3.5} },{_id:0}).limit(2);

Only documents where the gpa field is greater than 3.5 should be selected.

```
db> db.students.find({gpa:{$gt:3.5}},{_id:0}).limit(2);
[
  {
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Student 969',
    age: 24,
    courses: "['History', 'Mathematics', 'Physics', 'English']",
    gpa: 3.71,
    blood_group: 'B+',
    is_hotel_resident: true
  }
]
db>
```

Top 10 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
  }
]
test>
```

SELECTORS:

Grater than: The $gt operator in MongoDB is used to specify a query condition where the field value must be greater than a specified value. It stands for "greater than."

• Example: db. collection. find({ field: { $gt: value } }) selects documents where the field value is greater than value.

Lesser than: the less than operation is performed using the $lt operator. This operator allows you to query documents where a specific field's value is less than a given value.

• Example: db. collection. find({ field: { $lt: value } }) selects documents where the field value is less than value.

```
db> db.students.find({age:{$gt:20}});
[
  {
    _id: ObjectId('665609bde575a33e43e69479'),
    name: 'Student 328',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English']",
    gpa: 3.42,
    home_city: 'City 2',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665609bde575a33e43e6947c'),
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665609bde575a33e43e6947d'),
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665609bde575a33e43e6947f'),
    name: 'Student 915',
    age: 22,
    courses: "['Computer Science', 'History', 'Physics', 'English']",
    gpa: 3.37,
    blood_group: 'AB+',
    is_hotel_resident: true
  },
  {
```

It gives the student collection with age greater than 20 .

AND operator:

The logical AND operator ($and) is used to combine multiple query conditions. The $and operator is particularly useful when you need to match documents that satisfy all the specified conditions.

db. student. find({$and :{{home_city: “City 2”}, {blood_ group: “B+”}]});

```
Type "it" for more
db> db.students.find({$and:[{home_city:"City 2"},{blood_group:"B+"}]});
[
  {
    _id: ObjectId('665609bde575a33e43e6947d'),
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665609bde575a33e43e6962b'),
    name: 'Student 872',
    age: 24,
    courses: "['English', 'Mathematics', 'History']",
    gpa: 3.36,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665df4511655576509ad81c8'),
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665df4511655576509ad8376'),
    name: 'Student 872',
    age: 24,
    courses: "['English', 'Mathematics', 'History']",
    gpa: 3.36,
    home_city: 'City 2',
```

OR Operator:

MongoDB provides different types of logical query operators and $or operator is one of them. This operator is used to perform logical OR operation on the array of two or more expressions and select or retrieve only those documents that match at least one of the given expression in the array.

Syntax: { $or: [{ Expression1 }, { Expression2 }, ..., { ExpressionN }] }

```
// 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 } }
  ]
});
```