PROPAGATION ,LIMIT AND SELECTORS:

PROPAGATION:

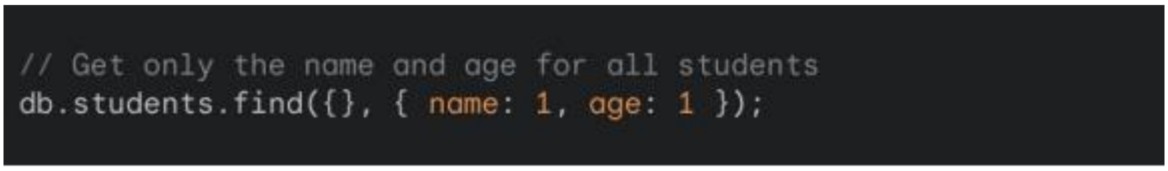
Use the projection document as the second argument to the find method.

- 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.

Get Selected Attributes :

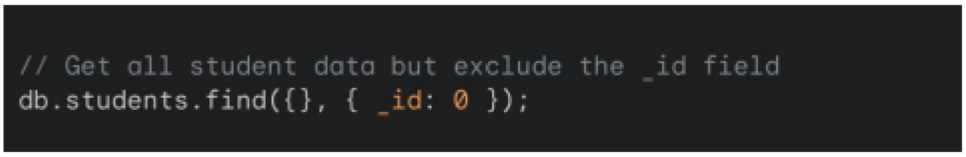
- Given a Collection you want to FILTER a subset of attributes.

That is the place Projection is used



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

Ignore attributes:



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

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 you need

Limit:

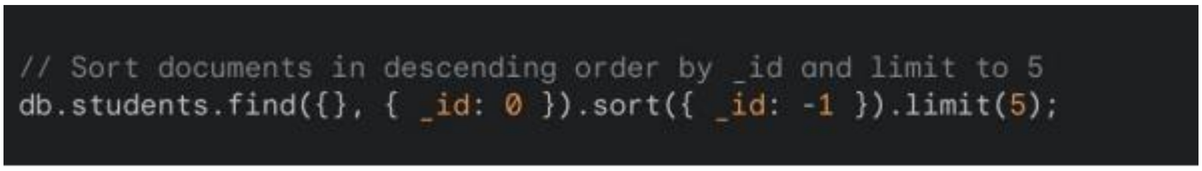
The limit operator is used with the find method.

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

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

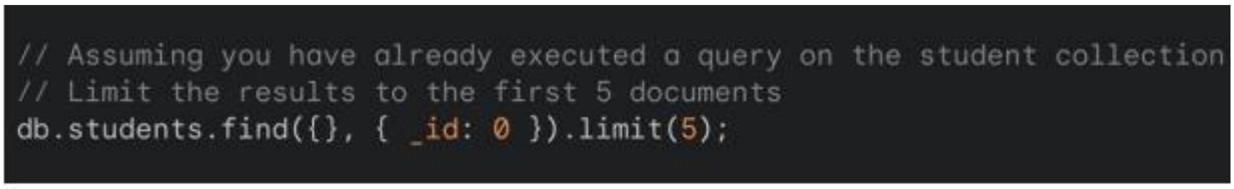
Examples:

1)I want top 10 results:



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

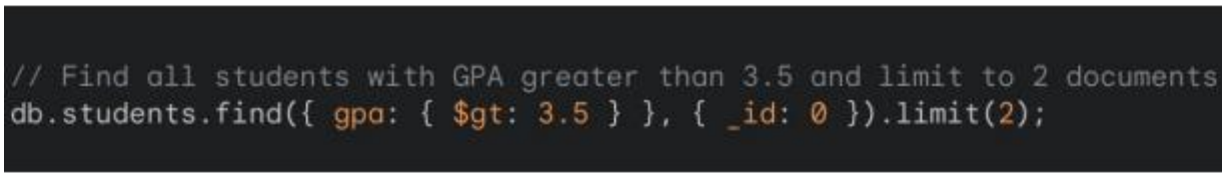
2)get first 5 documents:



```
// 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);
```

---

3)limiting result:

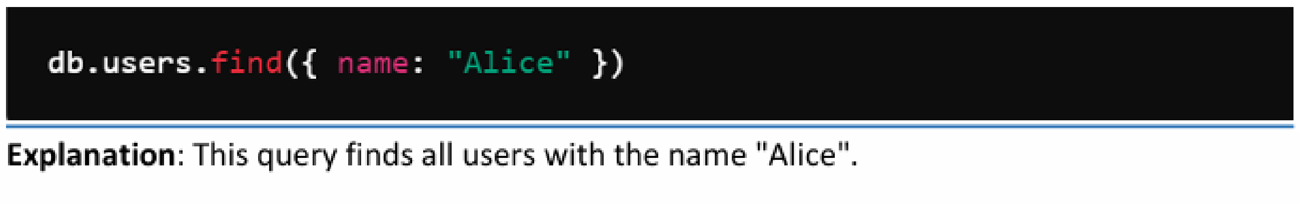


```
// Find all students with GPA greater than 3.5 and limit to 2 documents
db.students.find({ gpa: { $gt: 3.5 } }, { _id: 0 }).limit(2);
```

SELECTORS:

In MongoDB, selectors are part of query documents used to find specific data within a collection. These selectors utilize various operators to filter and target documents based on field values and conditions. Common operators include matching for equality, greater than/less than, and existence of fields. You can combine these with logical operators (AND, OR) to create complex queries for precise data retrieval.

EQUALITY SELECTOR:

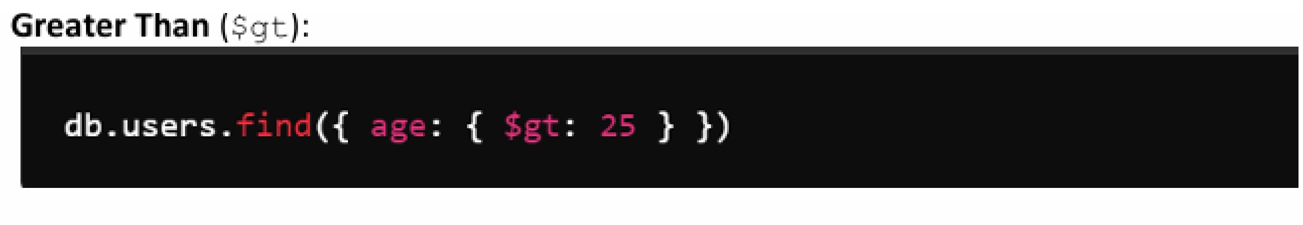


```
db.users.find({ name: "Alice" })
```

Explanation: This query finds all users with the name "Alice".

COMPARISION SELECTOR:

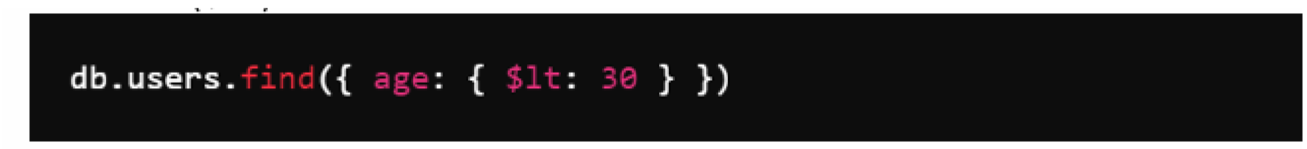
GREATER THAN:



Greater Than (`$gt`):

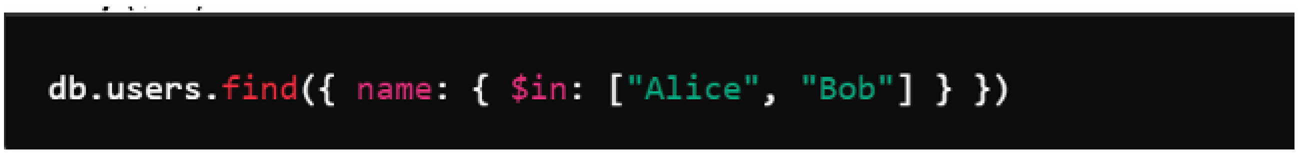
```
db.users.find({ age: { $gt: 25 } })
```

LESS THAN:



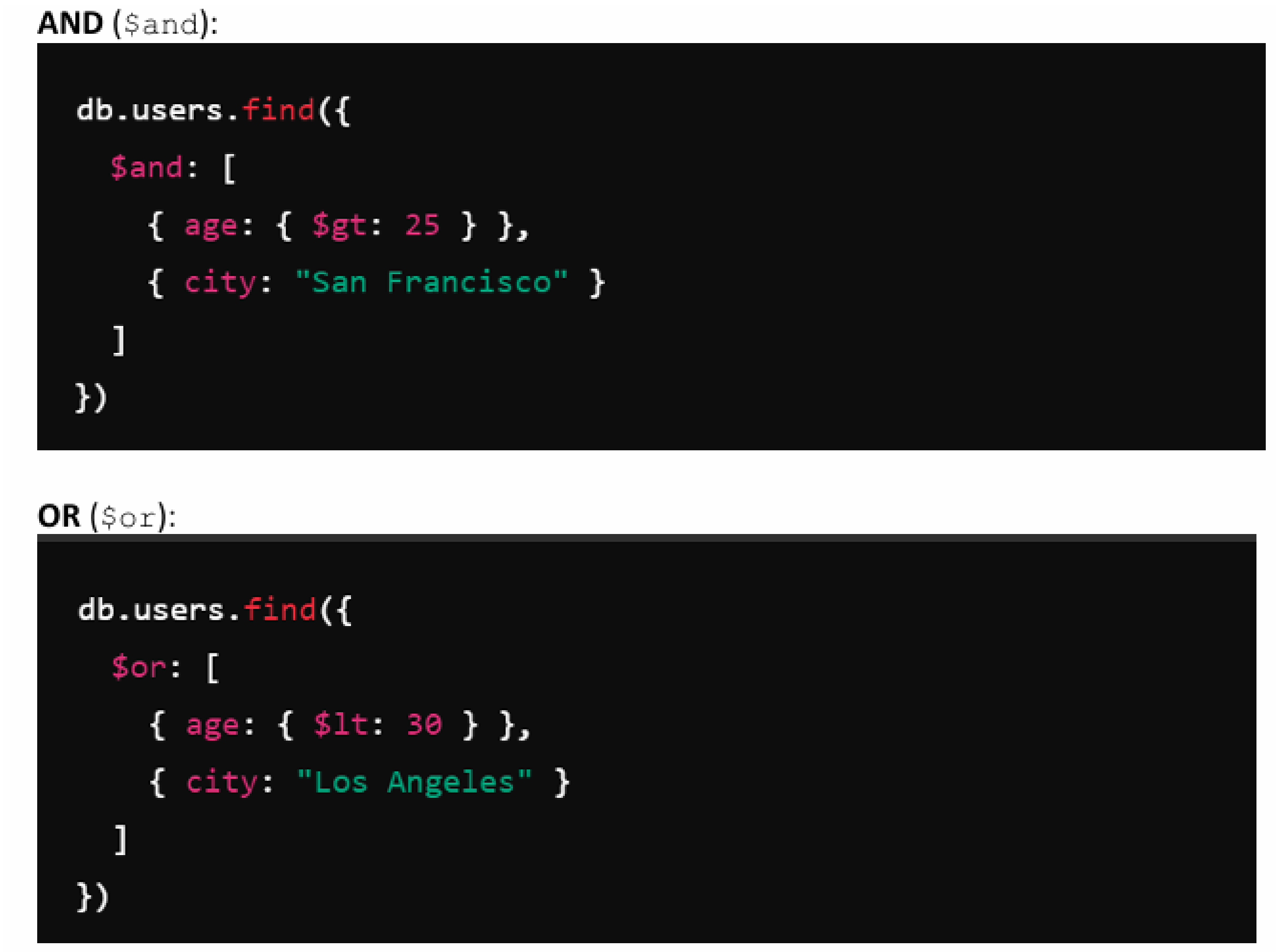
```
db.users.find({ age: { $lt: 30 } })
```

IN ARRAY:



```
db.users.find({ name: { $in: ["Alice", "Bob"] } })
```

LOGICAL SELECTORS:



AND ($and):

```
db.users.find({
  $and: [
    { age: { $gt: 25 } },
    { city: "San Francisco" }
  ]
})
```

OR ($or):

```
db.users.find({
  $or: [
    { age: { $lt: 30 } },
    { city: "Los Angeles" }
  ]
})
```