2. Where, AND, OR & CRUD

Sample Data:

As a first step I want to insert some sample data into **user** collection under **roytuts** database. Execute below insert() statements in the MongoDB shell to insert these data.

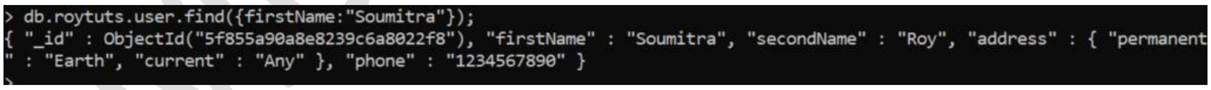
```
db.roytuts.user.insert({"firstName":"Soumitra","secondNam
e":"Roy","address":{"permanent":"Earth","current":"Any"},
"phone":"1234567890"});
db.roytuts.user.insert({"firstName":"John","secondName":"
Smith","address":{"permanent":"Earth","current":"Jupitor"
},"phone":"1234562890"});
db.roytuts.user.insert({"firstName":"Ikra","secondName":"
Michell","address":{"permanent":"Earth","current":"Mars"}
,"phone":"1234567924"});
db.roytuts.user.insert({"firstName":"Jolly","secondName":
"LLB","address":{"permanent":"Earth","current":"Saturn"},
"phone":"1284167924"});
```

WHERE Condition:

let's say you want to fetch data based on a firstName key from the user collection. The command can be written as, for example, where firstName is Soumitra.

```
db.roytuts.user.find({firstName:"Soumitra"});
```

The above command will give you the following output on your Mongo shell. If you want to pretty print your output then you can use pretty() function.



```
> db.roytuts.user.find({firstName:"Soumitra"});
{ "_id" : ObjectId("5f855a90a8e8239c6a8022f8"), "firstName" : "Soumitra", "secondName" : "Roy", "address" : { "permanent
" : "Earth", "current" : "Any" }, "phone" : "1234567890" }
>
```

The equivalent query in MySQL can be written as:

```
SELECT * FROM roytuts.user WHERE firstName = "Soumitra";
```

AND Condition:

AND condition will fetch data if both conditions match. Let's check out the following query.

```
db.roytuts.user.find({firstName:"Ikra",
"secondName":"Michell"});
```

The above command will give you the following output.

```
> db.roytuts.user.find({firstName:"Ikra",
"secondName":"Michell"});

{ "_id" : ObjectId("5f855a91a8e8239c6a8022fa"), "firstName"
: "Ikra", "secondName" : "Michell", "address" : {
"permanent" : "Earth", "current" : "Mars" }, "phone" :
"1234567924"
```

The equivalent MySQL query would be:

```
SELECT * FROM roytuts.user WHERE firstName = "Ikra" and
secondName = "Michell";
```

You can also query nested object. Let's say you want to fetch records for permanent address having Earth.

```
db.roytuts.user.find({"address.permanent":"Earth"});
```

The output would be for the above query.

```
> db.roytuts.user.find({"address.permanent":"Earth"});
{ "_id" : ObjectId("5f855a90a8e8239c6a8022f8"), "firstName"
: "Soumitra", "secondName" : "Roy", "address" : {
"permanent" : "Earth", "current" : "Any" }, "phone" :
"1234567890" }
{ "_id" : ObjectId("5f855a91a8e8239c6a8022f9"), "firstName"
: "John", "secondName" : "Smith", "address" : { "permanent"
: "Earth", "current" : "Jupitor" }, "phone" : "1234562890"
}
{ "_id" : ObjectId("5f855a91a8e8239c6a8022fa"), "firstName"
: "Ikra", "secondName" : "Michell", "address" : {
"permanent" : "Earth", "current" : "Mars" }, "phone" :
"1234567924" }
{ "_id" : ObjectId("5f855a92a8e8239c6a8022fb"), "firstName"
: "Jolly", "secondName" : "LLB", "address" : { "permanent"
: "Earth", "current" : "Saturn" }, "phone" : "1284167924" }
```

Let’s take an another example, you want to fetch records having permanent address Earth and phone number greater than 1234567890. Note here I have put double quote (“”) around the phone number value because I had inserted with double quotation for phone numbers.

```
db.roytuts.user.find({"address.permanent":"Earth",
"phone":{"$gte":"1234567890"}});
```

The above query will give you the following output:

```
> db.roytuts.user.find({"address.permanent":"Earth",
"phone":{"$gte":"1234567890"}});
{ "_id" : ObjectId("5f855a90a8e8239c6a8022f8"), "firstName" :
"Soumitra", "secondName" : "Roy", "address" : { "permanent" :
"Earth", "current" : "Any" }, "phone" : "1234567890" }
{ "_id" : ObjectId("5f855a91a8e8239c6a8022fa"), "firstName" :
"Ikra", "secondName" : "Michell", "address" : { "permanent" :
"Earth", "current" : "Mars" }, "phone" : "1234567924" }
{ "_id" : ObjectId("5f855a92a8e8239c6a8022fb"), "firstName" :
"Jolly", "secondName" : "LLB", "address" : { "permanent" :
"Earth", "current" : "Saturn" }, "phone" : "1284167924" }
```

You can also perform like operation, for example, you want to fetch records having permanent address as art. You can use any one of the query to achieve the same.

```
db.roytuts.user.find({"address.permanent":/art/});

Or

db.roytuts.user.find({"address.permanent":/.*art.*/});
```

The above queries will produce below outputs.

```
> db.roytuts.user.find({"address.permanent":/art/});
{ "_id" : ObjectId("5f855a90a8e8239c6a8022f8"),
"firstName" : "Soumitra", "secondName" : "Roy", "address"
: { "permanent" : "Earth", "current" : "Any" }, "phone" :
"1234567890" }
{ "_id" : ObjectId("5f855a91a8e8239c6a8022f9"),
"firstName" : "John", "secondName" : "Smith", "address" :
{ "permanent" : "Earth", "current" : "Jupitor" }, "phone"
: "1234562890" }
{ "_id" : ObjectId("5f855a91a8e8239c6a8022fa"),
"firstName" : "Ikra", "secondName" : "Michell", "address"
: { "permanent" : "Earth", "current" : "Mars" }, "phone"
: "1234567924" }
{ "_id" : ObjectId("5f855a92a8e8239c6a8022fb"),
"firstName" : "Jolly", "secondName" : "LLB", "address" :
{ "permanent" : "Earth", "current" : "Saturn" }, "phone"
: "1284167924" }

> db.roytuts.user.find({"address.permanent":/.*art.*/});
{ "_id" : ObjectId("5f855a90a8e8239c6a8022f8"),
"firstName" : "Soumitra", "secondName" : "Roy", "address"
: { "permanent" : "Earth", "current" : "Any" }, "phone" :
"1234567890" }
{ "_id" : ObjectId("5f855a91a8e8239c6a8022f9"),
"firstName" : "John", "secondName" : "Smith", "address" :
{ "permanent" : "Earth", "current" : "Jupitor" }, "phone"
: "1234562890" }
{ "_id" : ObjectId("5f855a91a8e8239c6a8022fa"),
"firstName" : "Ikra", "secondName" : "Michell", "address"
: { "permanent" : "Earth", "current" : "Mars" }, "phone"
: "1234567924" }
     { "_id" : ObjectId("5f855a92a8e8239c6a8022fb"),
 "firstName" : "Jolly", "secondName" : "LLB", "address" :
 { "permanent" : "Earth", "current" : "Saturn" }, "phone"
                    : "1284167924" }
```

OR Condition:

Matching any one of the given conditions will fetch records from database. In the AND condition you did not have to mention any AND operator but for OR condition you need to specify OR operator.

```
db.roytuts.user.find({"$or":[{firstName:"Ikra",
"secondName":"Michell"}]});
```

The above query is equivalent to the following MySQL query:

```
SELECT * FROM roytuts.user WHERE firstName = "Ikra" or
secondName = "Michell";
```

The above query will produce the following output:

```
> db.roytuts.user.find({"$or":[{firstName:"Ikra",
"secondName":"Michell"}]});
{ "_id" : ObjectId("5f855a91a8e8239c6a8022fa"), "firstName"
: "Ikra", "secondName" : "Michell", "address" : {
"permanent" : "Earth", "current" : "Mars" }, "phone" :
"1234567924" }
```

CRUD:

CRUD Operations (Create, Read, Update, and Delete) are the basic set of operations that allow users to interact with the MongoDB server.
As we know, to use MongoDB we need to interact with the MongoDB server to perform certain operations like entering new data into the application, updating data into the application, deleting data from the application, and reading the application data.

C
Create
R
Read
U
Update
D
Delete

In this article, we will learn all **4 major operations**– **CREATE**, **READ**, **UPDATE**, and **DELETE** that form the CRUD operations in MongoDB.

Perform CRUD Operations in MongoDB:

Now that we know the components of the CRUD operation, let's learn about each individual operation in MongoDB. We will know what each operation does, and the methods to perform these operations in MongoDB.
We will create, read, update and delete documents from MongoDB server.

1. Create Operations:

The **create or insert operations** are used to insert or add new documents in the collection. If a collection does not exist, then it will create a new collection in the database.

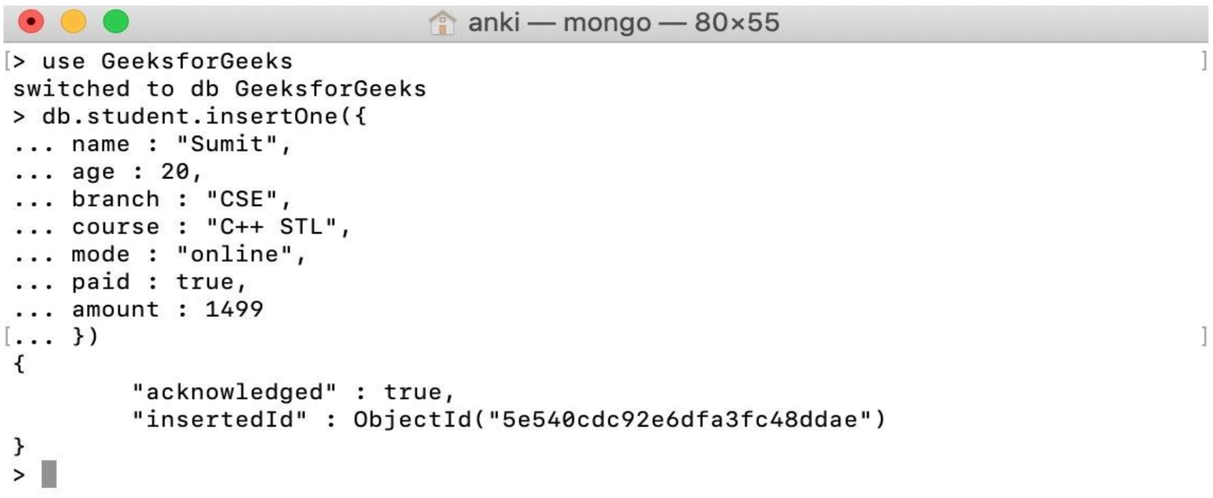
You can perform, create operations using the following methods provided by the MongoDB.

Method	Description
db.collection.insertOne()	It is used to insert a single document in the collection.
db.collection.insertMany()	It is used to insert multiple document I the collection.
db.createCollection()	It is used to create an empty collection.

Create Operations Example

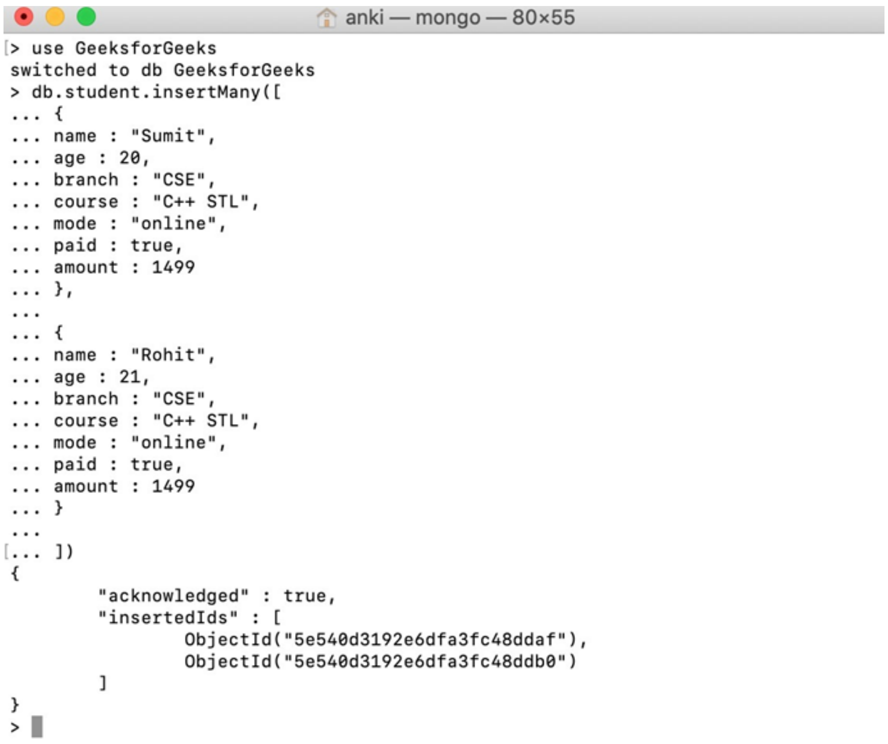
Let's look at some examples of the Create operation from CRUD in MongoDB.

Example 1: In this example, we are inserting details of a single student in the form of document in the student collection using **db.collection.insertOne() method**.



```
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.student.insertOne({
... name : "Sumit",
... age : 20,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... })
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5e540cdc92e6dfa3fc48ddae")
}
>
```

Example 2: In this example, we are inserting details of the multiple students in the form of documents in the student collection using **db.collection.insertMany() method**.



```
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.student.insertMany([
... {
... name : "Sumit",
... age : 20,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... },
...
... {
... name : "Rohit",
... age : 21,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... }
...
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5e540d3192e6dfa3fc48ddaf"),
                ObjectId("5e540d3192e6dfa3fc48ddb0")
        ]
}
>
```

2.Read Operations:

The Read operations are used to retrieve documents from the collection, or in other words, read operations are used to query a collection for a document.

You can perform read operation using the following method provided by the MongoDB:

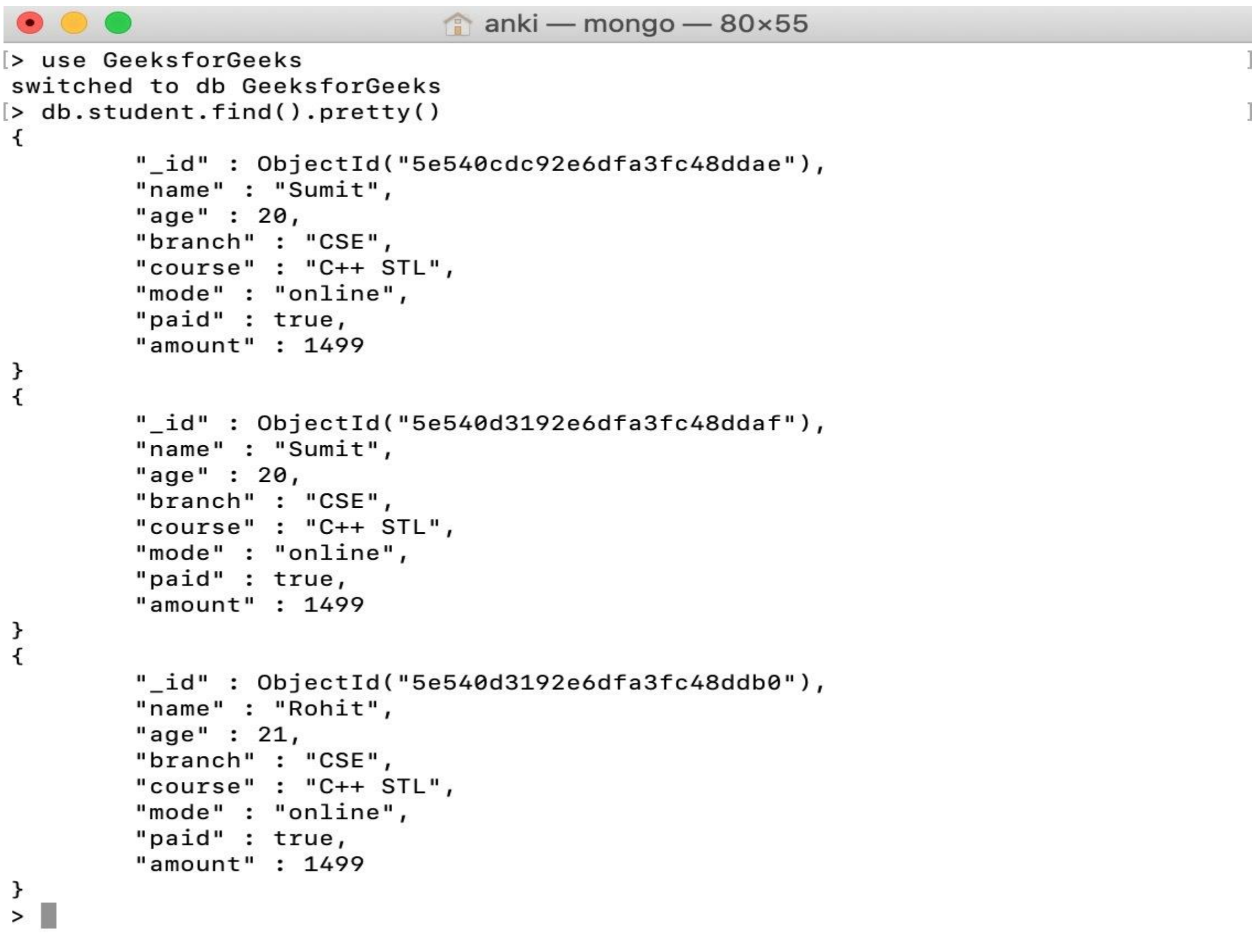
Method	Description
db.collection.find()	It is used to retrieve documents from the collection.

Note: pretty() method is used to decorate the result such that it is easy to read.

Read Operations Example

Let's look at some examples of Read operation from CRUD in MongoDB.

Example: In this example, we are retrieving the details of students from the student collection using **db.collection.find()** method.



```
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddb0"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
>
```

3.Update Operations:

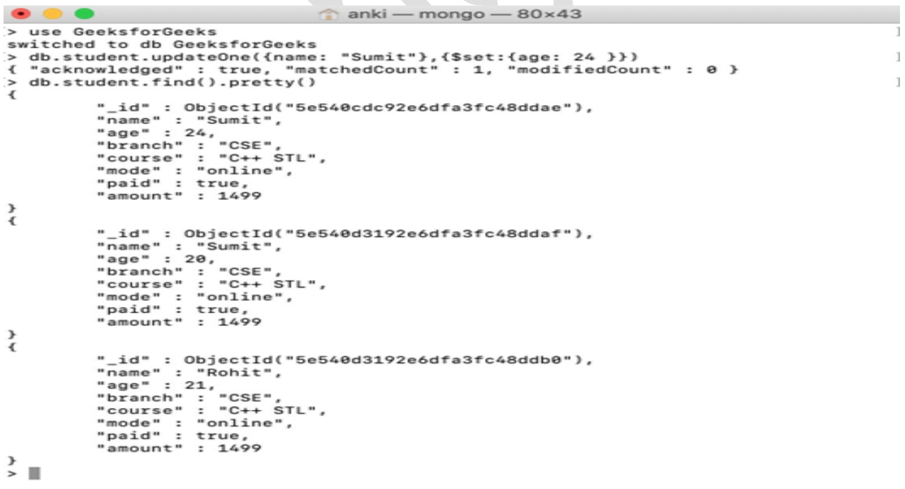
The update operations are used to update or modify the existing document in the collection. You can perform update operations using the following methods provided by the MongoDB:

Method	Description
db.collection.updateOne()	It is used to update a single document in the collection that satisfy the given criteria.
db.collection.updateMany()	It is used to update multiple document in the collection that satisfy the given criteria.
db.collection.replaceOne()	It is used to replace single document in the collection that satisfy the given criteria.

Update Operations Example:

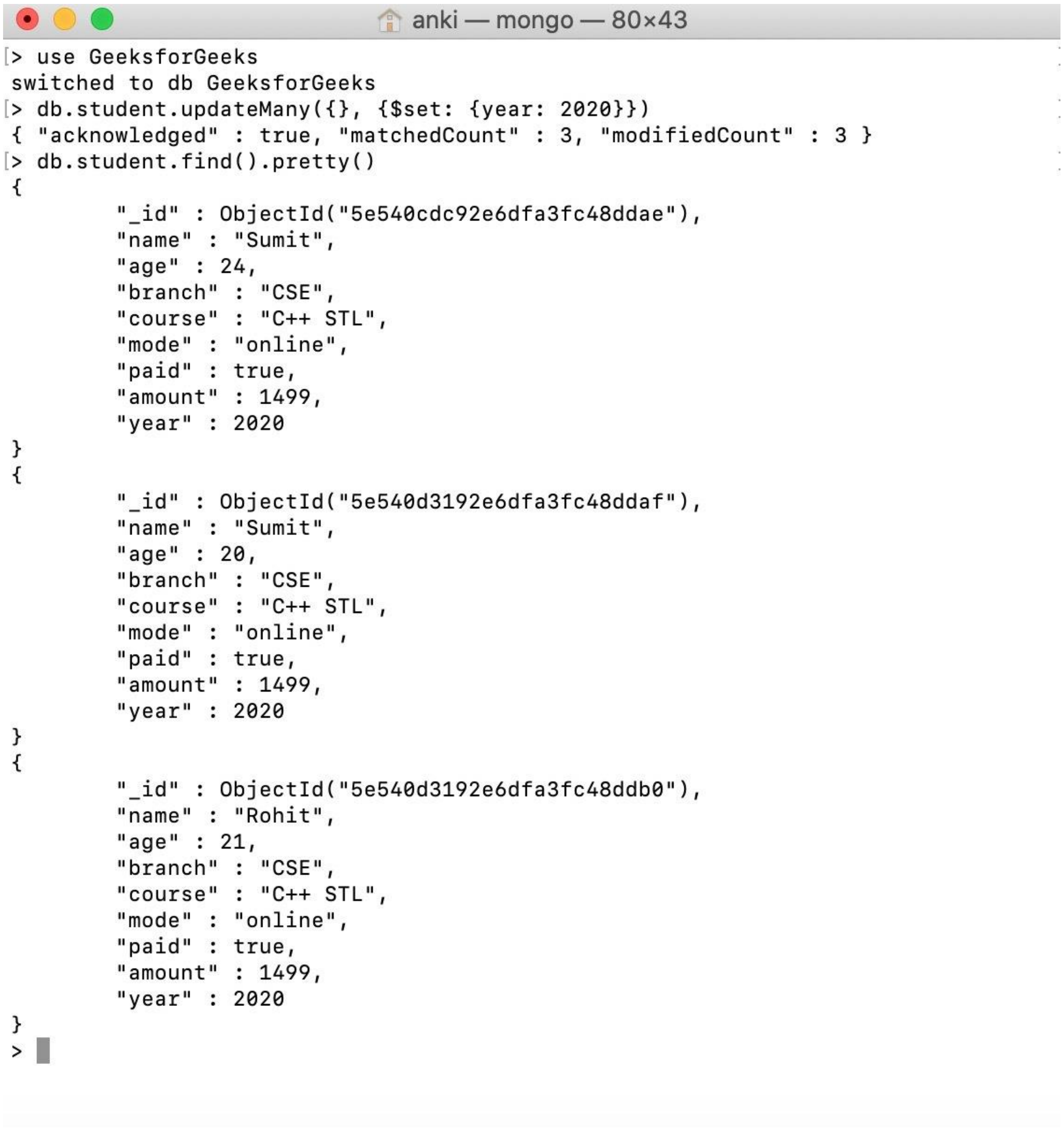
Let's look at some examples of the update operation from CRUD in MongoDB.

Example 1: In this example, we are updating the age of Sumit in the student collection using **db.collection.updateOne()** method.



```
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.student.updateOne({name: "Sumit"},{$set:{age: 24 }})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 0 }
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 24,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddb0"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
> 
```

Example 2: In this example, we are updating the year of course in all the documents in the student collection using db.collection.updateMany() method.



```
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.student.updateMany({}, {$set: {year: 2020}})
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 24,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddb0"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
>
```

4.Delete Operations:

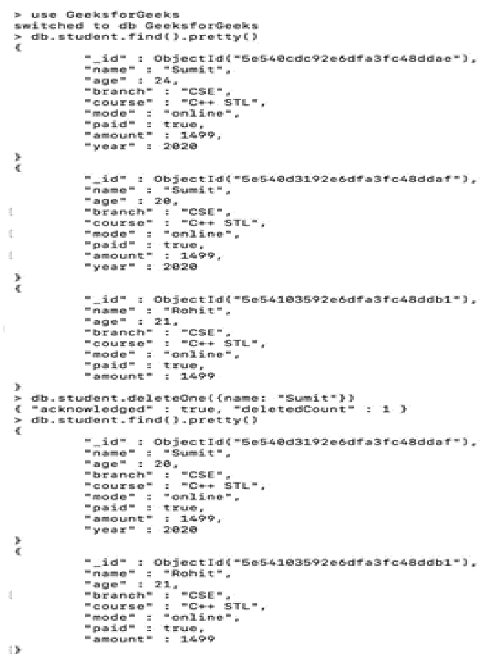
The delete operation are used to delete or remove the documents from a collection. You can perform delete operations using the following methods provided by the MongoDB:

Method	**Description**
db.collection.deleteOne()	It is used to delete a single document from the collection that satisfy the given criteria.
db.collection.deleteMany()	It is used to delete multiple documents from the collection that satisfy the given criteria.

Delete Operations Examples

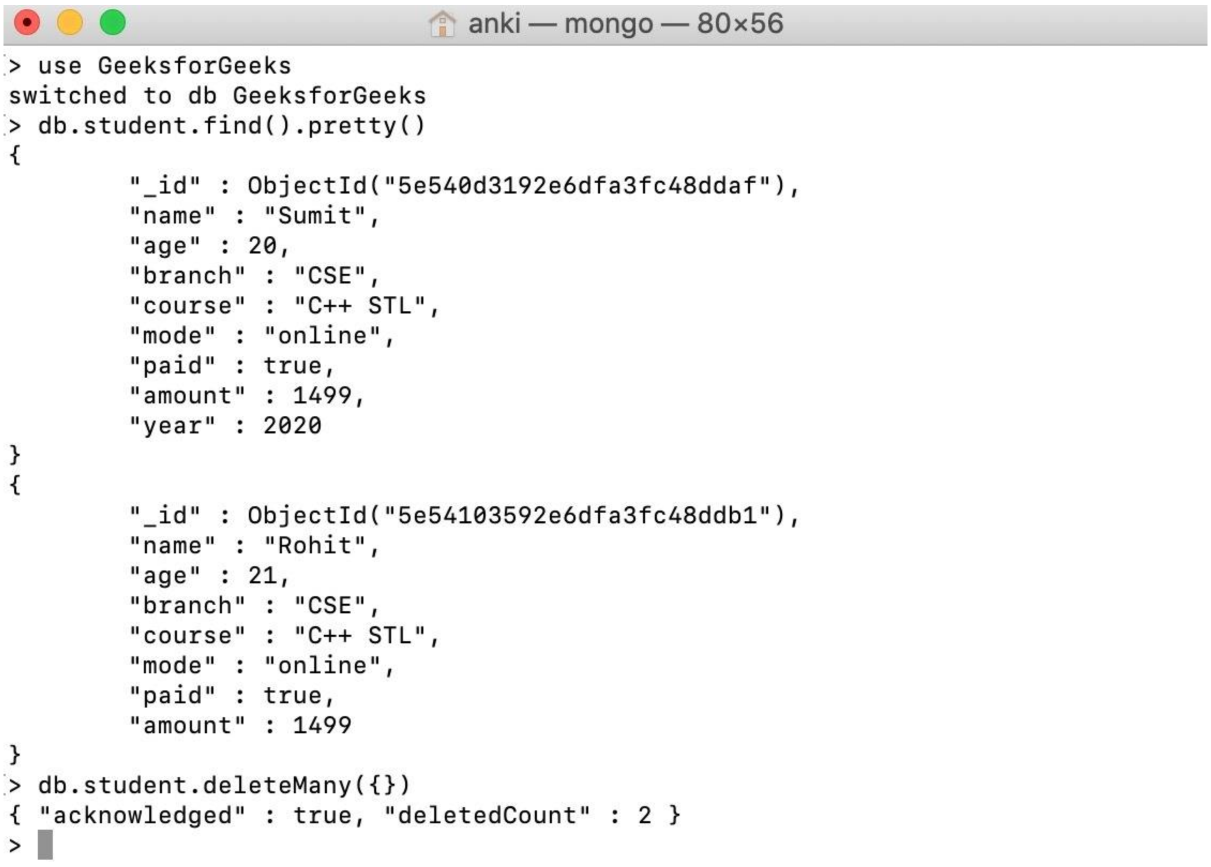
Let’s look at some examples of delete operation from CRUD in MongoDB.

Example 1: In this example, we are deleting a document from the student collection using **db.collection.deleteOne()** method.



```
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 24,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e54103592e6dfa3fc48ddb1"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
> db.student.deleteOne({name: "Sumit"})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e54103592e6dfa3fc48ddb1"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
```

Example 2: In this example, we are deleting all the documents from the student collection using **db.collection.deleteMany()** method.



```
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e54103592e6dfa3fc48ddb1"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
> db.student.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 2 }
>
```

5.Insert Operation:

The **insert() method** in MongoDB inserts documents in the MongoDB collection. This method is also used to create a new **collection** by inserting documents.

Note: This method is deprecated in Mongosh. Instead, use insertOne() and Insert()many methods to insert new documents in the MongoDB collection.

Syntax:

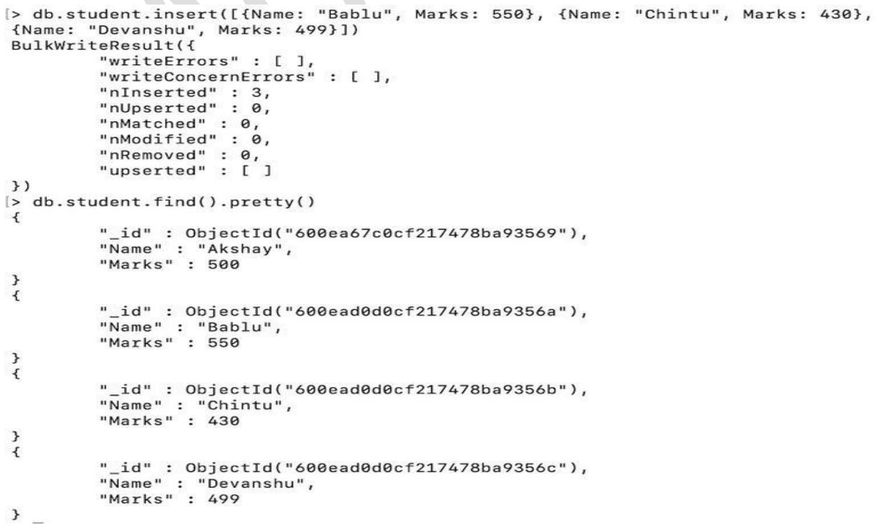
```
db.Collection_name.insert(
<document or [document1, document2,...]>,
{
   writeConcern: <document>,
   ordered: <boolean>
})
```

Insert Single or multiple Document in MongoDB Collection Example

Here, we insert multiple documents in the collection by passing an array of documents in the insert method.

```
db.student.insert([{Name: "Bablu", Marks: 550},
                   {Name: "Chintu", Marks: 430},
                   {Name: "Devanshu", Marks: 499}
])
```

Output:

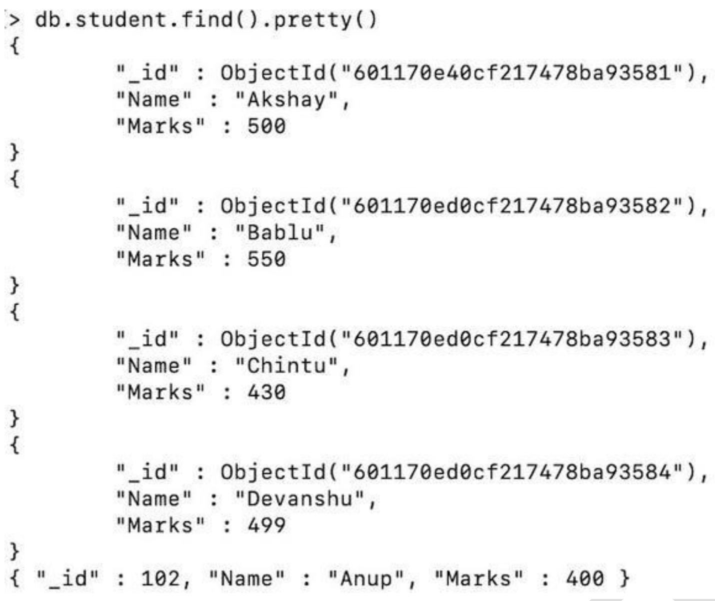


```
> db.student.insert([{Name: "Bablu", Marks: 550}, {Name: "Chintu", Marks: 430},
{Name: "Devanshu", Marks: 499}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 3,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.student.find().pretty()
{
        "_id" : ObjectId("600ea67c0cf217478ba93569"),
        "Name" : "Akshay",
        "Marks" : 500
}
{
        "_id" : ObjectId("600ead0d0cf217478ba9356a"),
        "Name" : "Bablu",
        "Marks" : 550
}
{
        "_id" : ObjectId("600ead0d0cf217478ba9356b"),
        "Name" : "Chintu",
        "Marks" : 430
}
{
        "_id" : ObjectId("600ead0d0cf217478ba9356c"),
        "Name" : "Devanshu",
        "Marks" : 499
} _
```

Here, we insert a document in the student collection with _id field.

```
db.student.insert({_id: 102,Name: "Anup", Marks: 400})
```

Output:



```
> db.student.find().pretty()
{
        "_id" : ObjectId("601170e40cf217478ba93581"),
        "Name" : "Akshay",
        "Marks" : 500
}
{
        "_id" : ObjectId("601170ed0cf217478ba93582"),
        "Name" : "Bablu",
        "Marks" : 550
}
{
        "_id" : ObjectId("601170ed0cf217478ba93583"),
        "Name" : "Chintu",
        "Marks" : 430
}
{
        "_id" : ObjectId("601170ed0cf217478ba93584"),
        "Name" : "Devanshu",
        "Marks" : 499
}
{ "_id" : 102, "Name" : "Anup", "Marks" : 400 }
```

6.Remove Operation:

The remove() method is a part of MongoDB’s native CRUD (Create, Read, Update, Delete) operations. Using this method, you can delete single or multiple documents from a collection that match a given query criteria.

```
db.users.remove({ 'name': 'John' }, true)
```

You can also perform removals that meet certain conditions. For example, if you want to delete users who are older than 30:

```
db.users.remove({ 'age': { '$gt': 30 } })
```

To remove all documents from a collection, simply use an empty query object:

db.users.remov({})

we wanted to remove all users that either have an ‘age’ less than 20 or a ‘name’ that starts with ‘S’, we would use the $or operator:

```
db.users.remove({
'$or': [
 { 'age':
{ '$lt': 20 } },
 { 'name': /^S/ }
]
 })
```

7. Projection Operation:

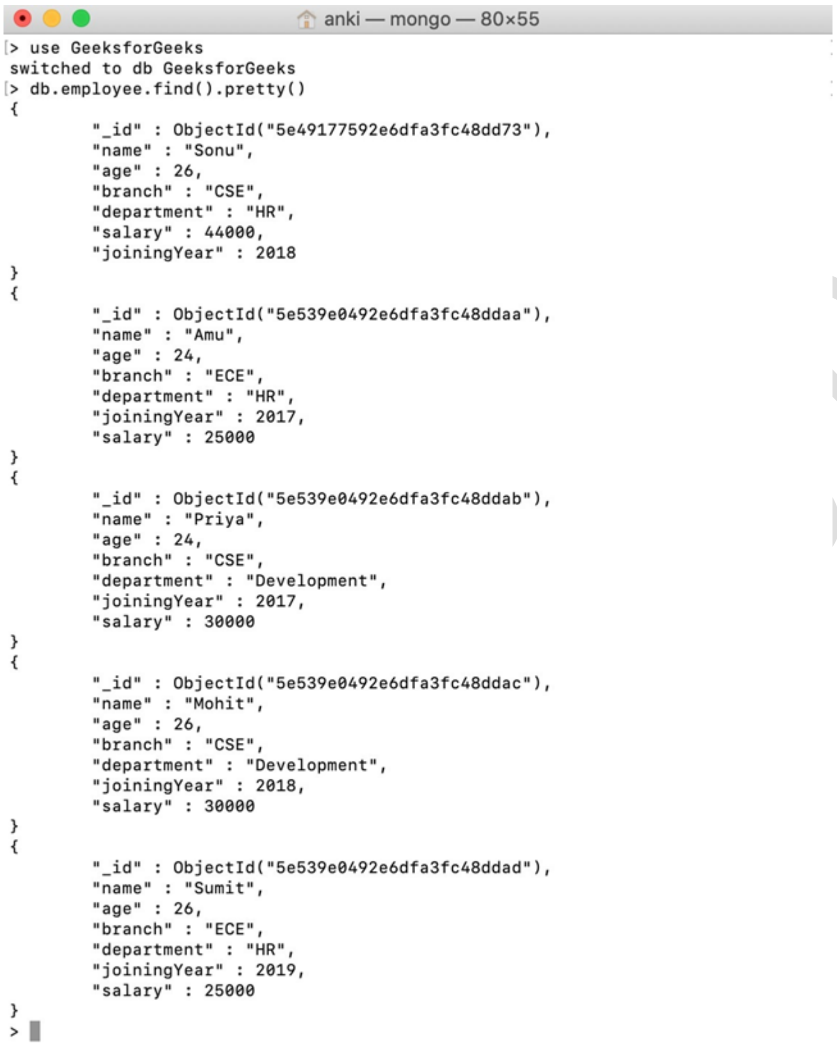
MongoDB provides a special feature that is known as **Projection**. It allows you to select only the necessary data rather than selecting whole data from the document. For example, a document contains 5 fields, i.e.,

```
{
name: "Roma",
age: 30,
branch: EEE,
department: "HR",
salary: 20000
}
```

Syntax:

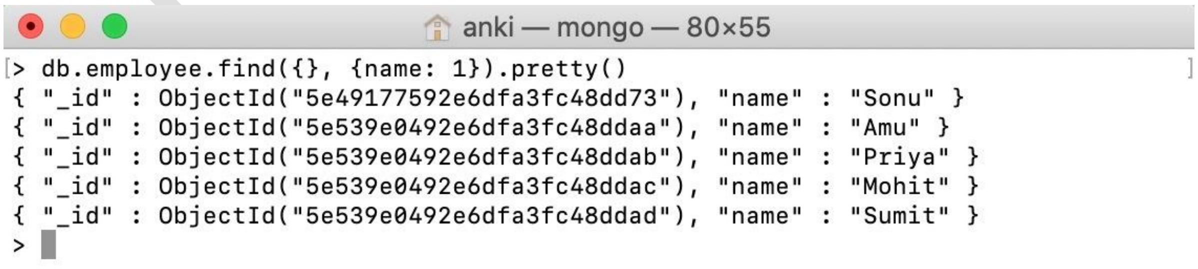
```
db.collection.find({}, {field1: value2, field2: value2, ..})
```

Example:



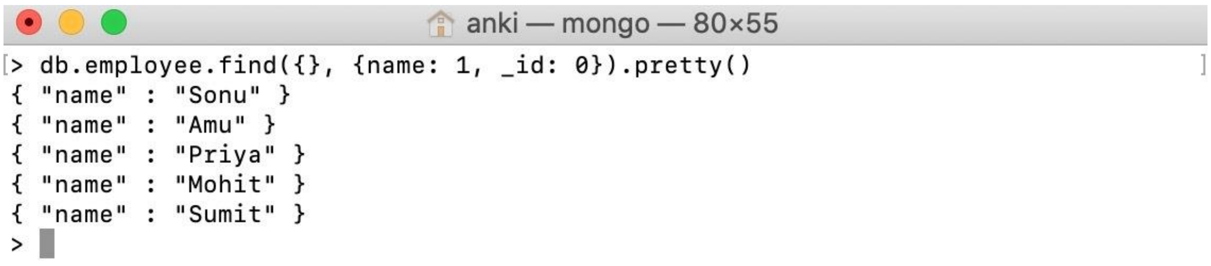
```
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.employee.find().pretty()
{
        "_id" : ObjectId("5e49177592e6dfa3fc48dd73"),
        "name" : "Sonu",
        "age" : 26,
        "branch" : "CSE",
        "department" : "HR",
        "salary" : 44000,
        "joiningYear" : 2018
}
{
        "_id" : ObjectId("5e539e0492e6dfa3fc48ddaa"),
        "name" : "Amu",
        "age" : 24,
        "branch" : "ECE",
        "department" : "HR",
        "joiningYear" : 2017,
        "salary" : 25000
}
{
        "_id" : ObjectId("5e539e0492e6dfa3fc48ddab"),
        "name" : "Priya",
        "age" : 24,
        "branch" : "CSE",
        "department" : "Development",
        "joiningYear" : 2017,
        "salary" : 30000
}
{
        "_id" : ObjectId("5e539e0492e6dfa3fc48ddac"),
        "name" : "Mohit",
        "age" : 26,
        "branch" : "CSE",
        "department" : "Development",
        "joiningYear" : 2018,
        "salary" : 30000
}
{
        "_id" : ObjectId("5e539e0492e6dfa3fc48ddad"),
        "name" : "Sumit",
        "age" : 26,
        "branch" : "ECE",
        "department" : "HR",
        "joiningYear" : 2019,
        "salary" : 25000
}
>
```

Displaying the names of the employees



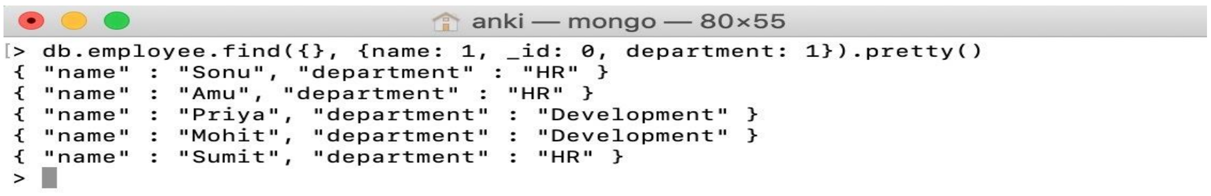
```
> db.employee.find({}, {name: 1}).pretty()
{ "_id" : ObjectId("5e49177592e6dfa3fc48dd73"), "name" : "Sonu" }
{ "_id" : ObjectId("5e539e0492e6dfa3fc48ddaa"), "name" : "Amu" }
{ "_id" : ObjectId("5e539e0492e6dfa3fc48ddab"), "name" : "Priya" }
{ "_id" : ObjectId("5e539e0492e6dfa3fc48ddac"), "name" : "Mohit" }
{ "_id" : ObjectId("5e539e0492e6dfa3fc48ddad"), "name" : "Sumit" }
>
```

Displaying the names of the employees without the _id field –



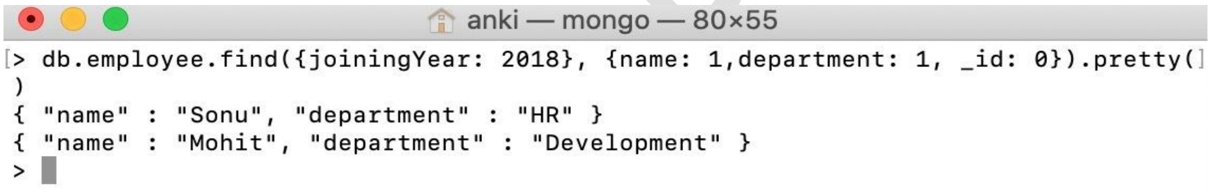
```
> db.employee.find({}, {name: 1, _id: 0}).pretty()
{ "name" : "Sonu" }
{ "name" : "Amu" }
{ "name" : "Priya" }
{ "name" : "Mohit" }
{ "name" : "Sumit" }
>
```

Displaying the name and the department of the employees without the _id field –



```
> db.employee.find({}, {name: 1, _id: 0, department: 1}).pretty()
{ "name" : "Sonu", "department" : "HR" }
{ "name" : "Amu", "department" : "HR" }
{ "name" : "Priya", "department" : "Development" }
{ "name" : "Mohit", "department" : "Development" }
{ "name" : "Sumit", "department" : "HR" }
>
```

Displaying the names and the department of the employees whose joining year is 2018 –



```
> db.employee.find({joiningYear: 2018}, {name: 1,department: 1, _id: 0}).pretty(
)
{ "name" : "Sonu", "department" : "HR" }
{ "name" : "Mohit", "department" : "Development" }
>
```