INTRUDUCTION:

MongoDB is a popular NoSQL database system that stores data in flexible, JSON-like documents.Unlike relational databases that use tables with predefined schemas, MongoDB offers a more scalable and adaptable way to store information.

Breakdown of its key features

- **Document-oriented:** Data is stored in JSON-like documents, allowing you to represent complex data structures naturally.
- **Schema-less:** While schemas can be defined for better data organization, MongoDB doesn't enforce them strictly, providing flexibility.
- **Scalability:** MongoDB excels at handling large datasets and can be easily scaled horizontally by adding more servers.

- **Document-oriented:** Data is stored in JSON-like documents, allowing you to represent complex data structures naturally.
- **Schema-less:** While schemas can be defined for better data organization, MongoDB doesn't enforce them strictly, providing flexibility.
- **Scalability:** MongoDB excels at handling large datasets and can be easily scaled horizontally by adding more servers.

What is data base?

A dataBase is essentially a big organized collection of information stored electronically on a computer system. This data can include things like words,numbers,images,and even videos. Datanase are used to store and mange information efficiently, so it can be easily accessed,searched and updated.

Data base can be work based on:

1)data organization

2)data base manegment system

Future of mango DB:

- **AI-powered features:** MongoDB is exploring ways to integrate AI for data analysis and automation within the database itself.
- **Hybrid and multi-cloud adoption:** As cloud computing evolves, MongoDB's ability to run on various cloud platforms or even a user's desktop positions it well for the future .
- **Focus on sustainability:** MongoDB's commitment to reducing its carbon footprint through renewable energy aligns with the growing focus on eco-friendly technology.

Where we use mongo DB:

 Large and Diverse Data: If you're working with a massive amount of information from various sources, MongoDB's flexible schema lets you integrate it seamlessly into a unified view.

 Complex and Evolving Data Structures: Need to store data with intricate structures that might change over time? MongoDB's document model allows for nesting documents and accommodates variations effortlessly.

 High-Performance Applications: MonMongoDB is built for speed, making it ideal for applications that require fast data retrieval, updates, and delivery.

Cloud-Based Deployments: MongoDB integrates well with cloud environments, making it a perfect fit for applications leveraging cloud infrastructure.

Agile Development: The flexible schema and ease of use make MongoDB popular with agile development teams, allowing for rapid prototyping and iterative changes.

INSTALLATION OF MONGO DB:

- Mongo Shell download link
- All the work is expected to do it in mongo shell not in mongo compass

OR

- You can also install Studio3T
- Connect to mongodb://localhost:27017

Few Commends to test after connections :

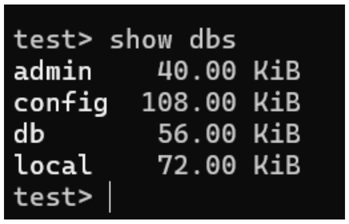
Command		Notes
show dbs		All Databases are shown
use db		Connect and use db
show collections		Show all tables
db.foo.insert({"bar" : "baz"})		Insert a record to collection. Create Collection if not exists
db.boo.find()		Print all rows

db.boo.remove()		Remove foo table

Add ,Update & Delete Data:

First we want to view all data base using command,

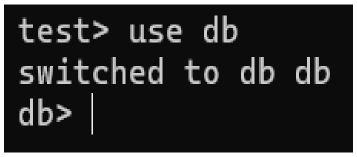
“Show dbs”



```
test> show dbs
admin    40.00 KiB
config  108.00 KiB
db       56.00 KiB
local    72.00 KiB
test> |
```

Second we want to create a new database using commend ,

“Use db”

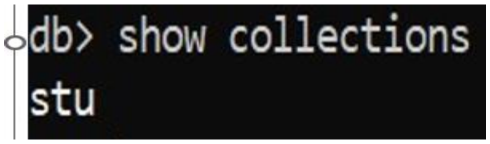


```
test> use db
switched to db db
db> |
```

Now the data base is switched to db.

To find where the given data present in the collection, use commend,

“Show collection”



```
db> show collections
stu
```

To create stud collections:

db.create collection(“stu”)

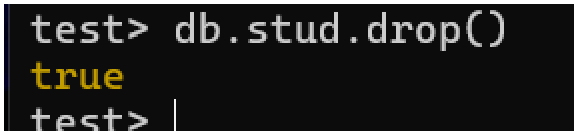
In the above ex the collection name is stu

To insert a file to the collection:

db.stu.insert({name:”alice”})

To drop the stud collection:

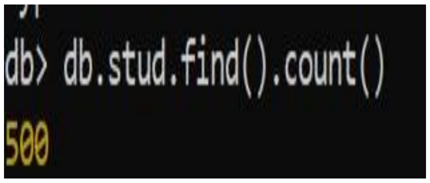
db.stud.drop()



```
test> db.stud.drop()
true
test>
```

To find the total number of collection of the database use the command.

“db.stud.find().count()”

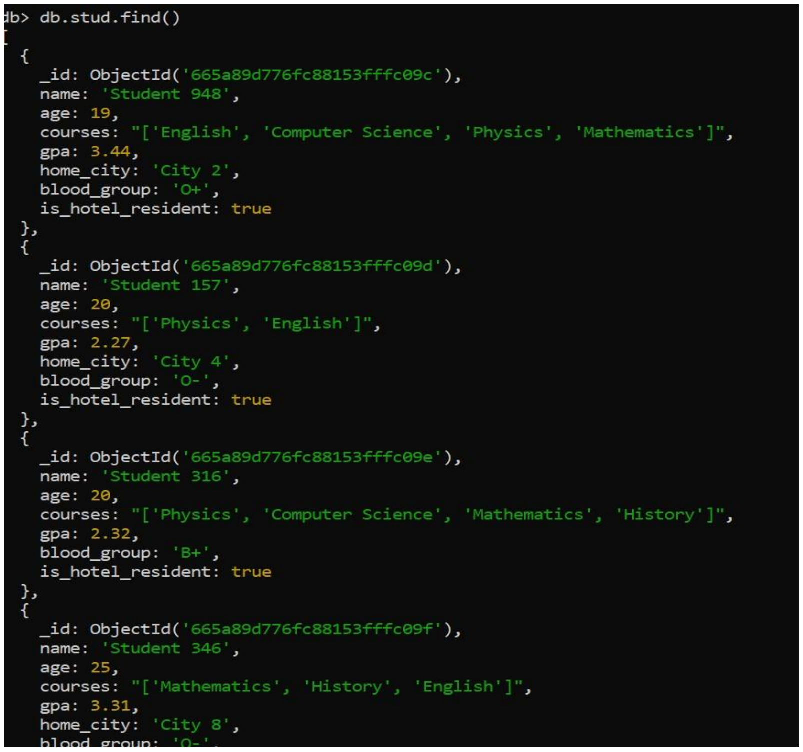


```
db> db.stud.find().count()
500
```

It shows the total collections of students in the database.

To find the data from the collection:

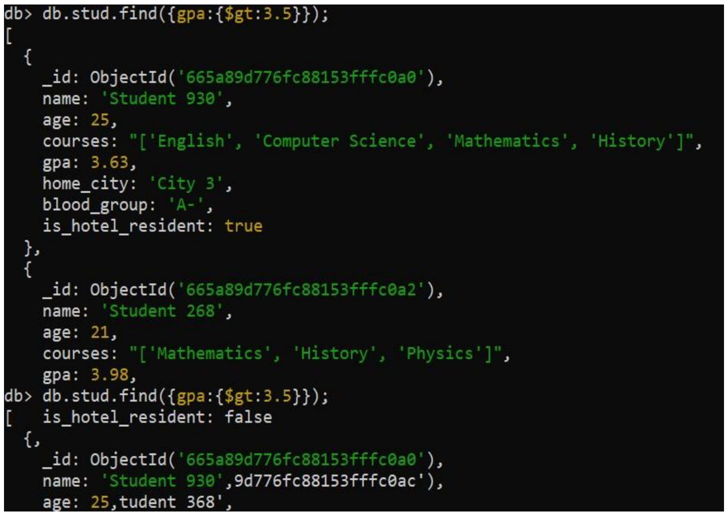
db.stud.find()



```
db> db.stud.find()
[
  {
    _id: ObjectId('665a89d776fc88153fffc09c'),
    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('665a89d776fc88153fffc09d'),
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665a89d776fc88153fffc09e'),
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.32,
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665a89d776fc88153fffc09f'),
    name: 'Student 346',
    age: 25,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 3.31,
    home_city: 'City 8',
    blood_group: 'O-',
```

WHERE ,AND ,OR & CRUD:

- ### Where



```
db> db.stud.find({gpa:{$gt:3.5}});
[
  {
    _id: ObjectId('665a89d776fc88153fffc0a0'),
    name: 'Student 930',
    age: 25,
    courses: "['English', 'Computer Science', 'Mathematics', 'History']",
    gpa: 3.63,
    home_city: 'City 3',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665a89d776fc88153fffc0a2'),
    name: 'Student 268',
    age: 21,
    courses: "['Mathematics', 'History', 'Physics']",
    gpa: 3.98,
db> db.stud.find({gpa:{$gt:3.5}});
[   is_hotel_resident: false
  {,
    _id: ObjectId('665a89d776fc88153fffc0a0'),
    name: 'Student 930',9d776fc88153fffc0ac'),
    age: 25,tudent 368',
```

Given a collection you want to filter a subset based on multiple condition that is the place WHERE is used.

To find the students based on gpa more than 3.5 with query

db.stud.find({gpa:{$gt:3.5}});

Here "$gt" text represents the greater than.and in above example it give the collection with students greater than 3.5.

- ### AND

Given a collection you want to filter a subset based on multiple condition. That is the place AND Is used

To find students belongs to "city 5" AND "blood group is "A+".Using command,

db.students.find({

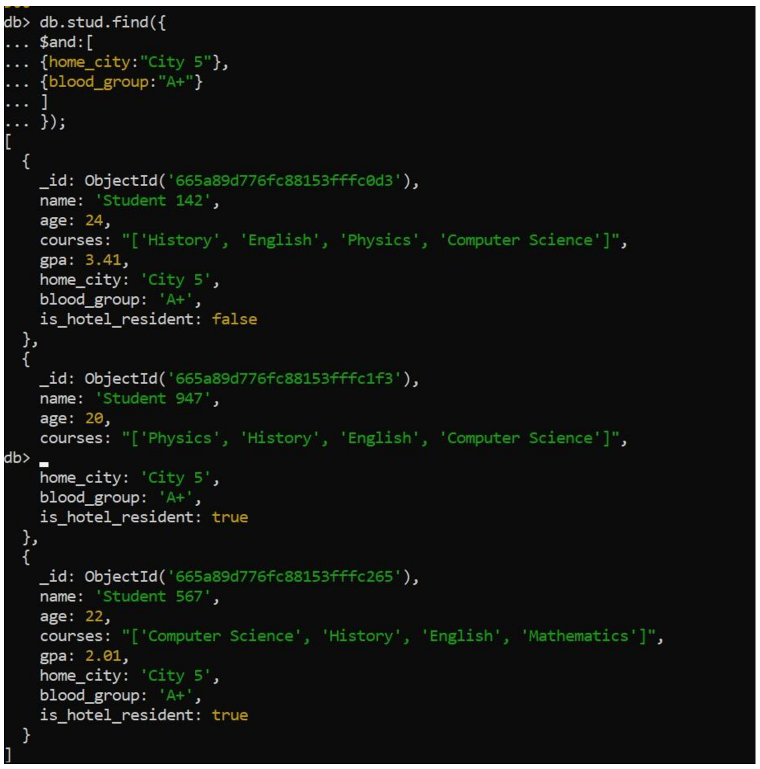
$and:[

(home_city:"city b").

{blood group: "A+")

]

});



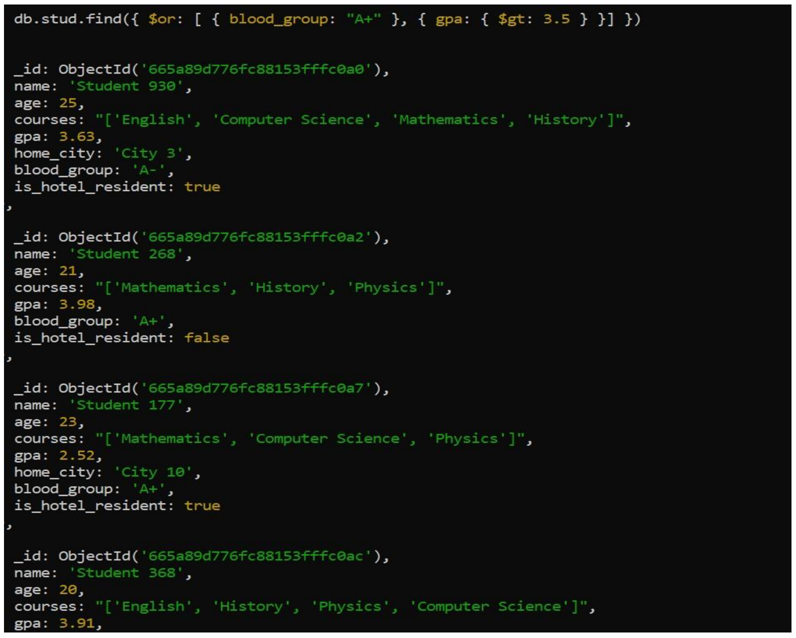
```
db> db.stud.find({
... $and:[
... {home_city:"City 5"},
... {blood_group:"A+"}
... ]
... });
[
  {
    _id: ObjectId('665a89d776fc88153fffc0d3'),
    name: 'Student 142',
    age: 24,
    courses: "['History', 'English', 'Physics', 'Computer Science']",
    gpa: 3.41,
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('665a89d776fc88153fffc1f3'),
    name: 'Student 947',
    age: 20,
    courses: "['Physics', 'History', 'English', 'Computer Science']",
db> 
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665a89d776fc88153fffc265'),
    name: 'Student 567',
    age: 22,
    courses: "['Computer Science', 'History', 'English', 'Mathematics']",
    gpa: 2.01,
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: true
  }
]
```

Here we use AND operation it give the students who belongs to “city 5” and “blood_group A+”.

- **OR**

Given a collection you want to filter a subset based on multiple condition that is the place OR is used .

To find students belongs to “city 5” AND “blood group is “A+”.Using command



```
 db.stud.find({ $or: [ { blood_group: "A+" }, { gpa: { $gt: 3.5 } }] })


 _id: ObjectId('665a89d776fc88153fffc0a0'),
 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
,

 _id: ObjectId('665a89d776fc88153fffc0a2'),
 name: 'Student 268',
 age: 21,
 courses: "['Mathematics', 'History', 'Physics']",
 gpa: 3.98,
 blood_group: 'A+',
 is_hotel_resident: false
,

 _id: ObjectId('665a89d776fc88153fffc0a7'),
 name: 'Student 177',
 age: 23,
 courses: "['Mathematics', 'Computer Science', 'Physics']",
 gpa: 2.52,
 home_city: 'City 10',
 blood_group: 'A+',
 is_hotel_resident: true
,

 _id: ObjectId('665a89d776fc88153fffc0ac'),
 name: 'Student 368',
 age: 20,
 courses: "['English', 'History', 'Physics', 'Computer Science']",
 gpa: 3.91,
```

- **CRUD**

- C - Create / Insert
- R - Remove
- U - update
- D - Delete

This is applicable for a Collection (Table) or a Document (Row)

- Insert

To insert new data into collection use the below query

Const studentsData={

“name”:“Alice Smith”,

“age”:12,,

“coueses”:[“Mathematics”,“computer science”,“ English”],

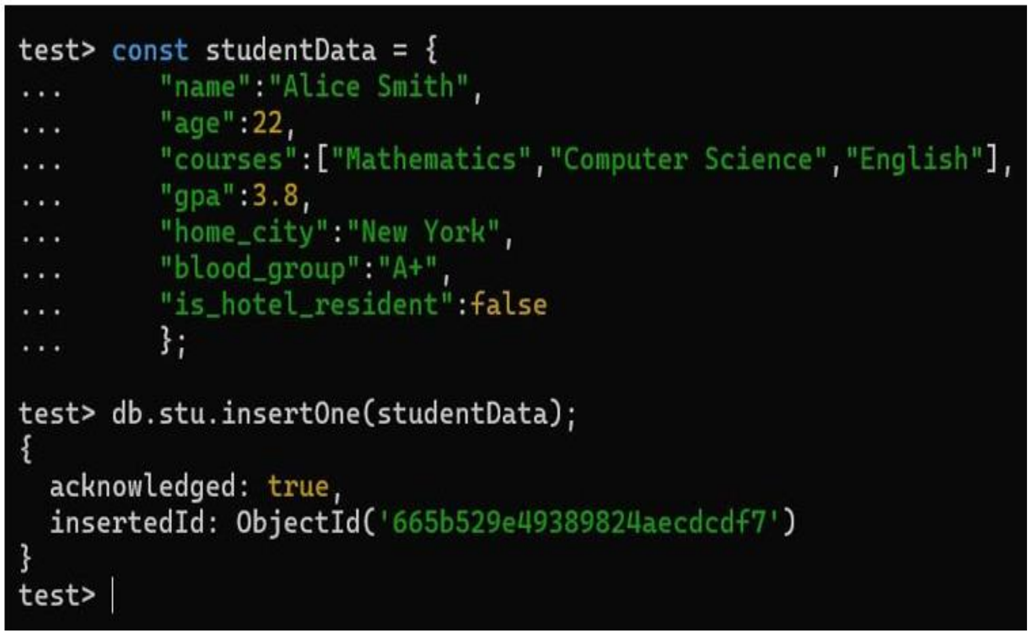
“gpa”:3.5

“home_city”:“New York”,

“blood_group”:“A+”,

“is_hotel_resident”:false

};



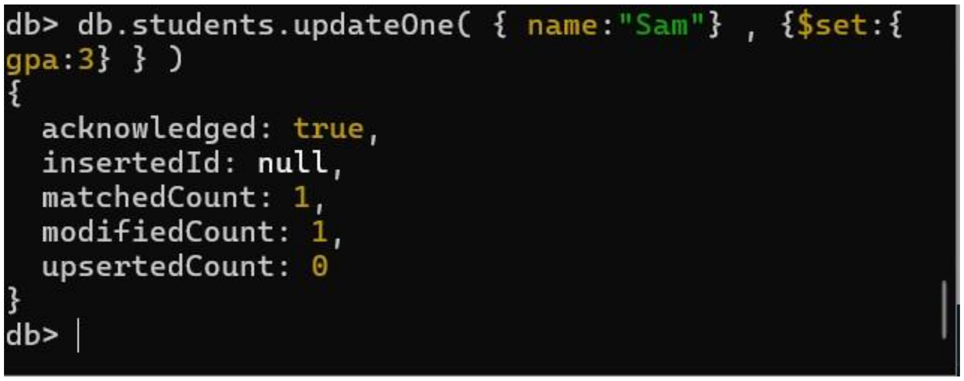
```
test> const studentData = {
...       "name":"Alice Smith",
...       "age":22,
...       "courses":["Mathematics","Computer Science","English"],
...       "gpa":3.8,
...       "home_city":"New York",
...       "blood_group":"A+",
...       "is_hotel_resident":false
...       };

test> db.stu.insertOne(studentData);
{
  acknowledged: true,
  insertedId: ObjectId('665b529e49389824aecdcdf7')
}
test> |
```

- **Update**

To update the given students data using query

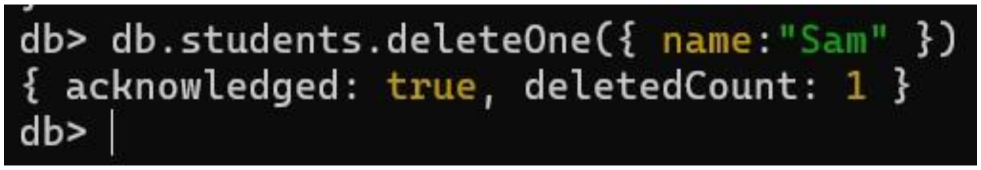
db.students.updateOne({ name: “Sam”} , {$set: {gpa:3.5} })



```
db> db.students.updateOne( { name:"Sam"} , {$set:{
gpa:3} } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
db> |
```

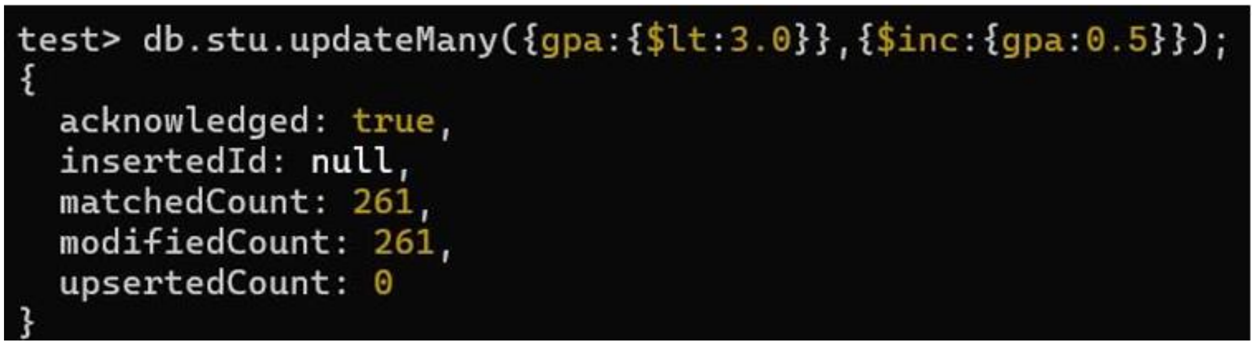
- **Delete**

Delete is used to delete one student data from the given collection.



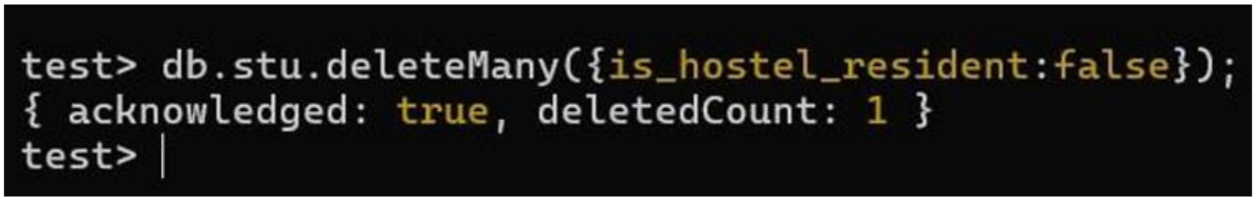
```
db> db.students.deleteOne({ name:"Sam" })
{ acknowledged: true, deletedCount: 1 }
db> |
```

- **Update Many**



```
test> db.stu.updateMany({gpa:{$lt:3.0}},{$inc:{gpa:0.5}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 261,
  modifiedCount: 261,
  upsertedCount: 0
}
```

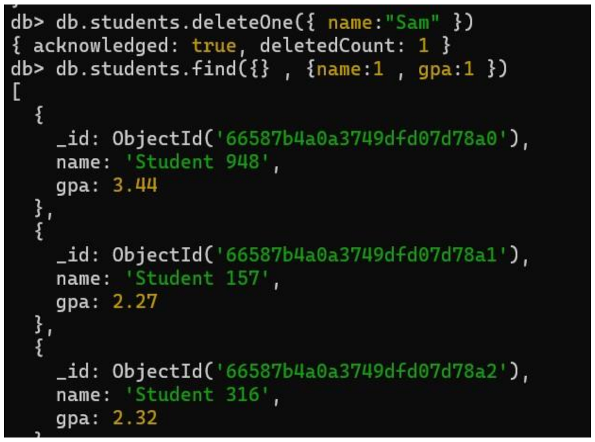
- **Delete Many**



```
test> db.stu.deleteMany({is_hostel_resident:false});
{ acknowledged: true, deletedCount: 1 }
test> |
```

- ## Projection

this is used when we don’t need all columns. MongoDB provides a special feature that is known as Projection,



```
db> db.students.deleteOne({ name:"Sam" })
{ acknowledged: true, deletedCount: 1 }
db> db.students.find({} , {name:1 , gpa:1 })
[
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a0'),
    name: 'Student 948',
    gpa: 3.44
  },
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a1'),
    name: 'Student 157',
    gpa: 2.27
  },
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a2'),
    name: 'Student 316',
    gpa: 2.32
```