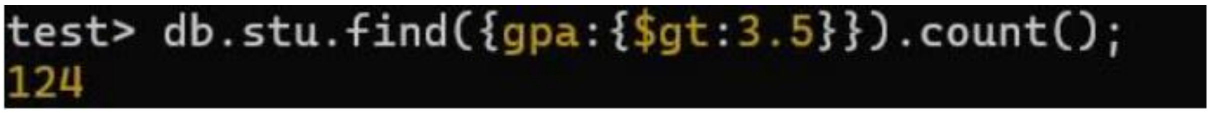
MONGO DB

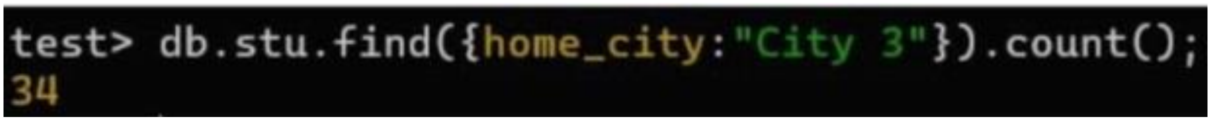
CLASS 3: WHERE, AND, OR & CRUD

WHERE:

Given a Collection you want to FILTER a subset based on a condition. That is the place WHERE is used. For queries that cannot be done any other way, there are "$where" clauses, which allow you to execute arbitrary JavaScript as part of your query. This allows you to do (almost) anything within a query. For security, use of "$where" clauses should be highly restricted or eliminated. End users should never be allowed to execute arbitrary "$where" clauses.



```
test> db.stu.find({gpa:{$gt:3.5}}).count();
124
```



```
test> db.stu.find({home_city:"City 3"}).count();
34
```

AND:

To find data by considering both the condition what is mentioned we use AND.

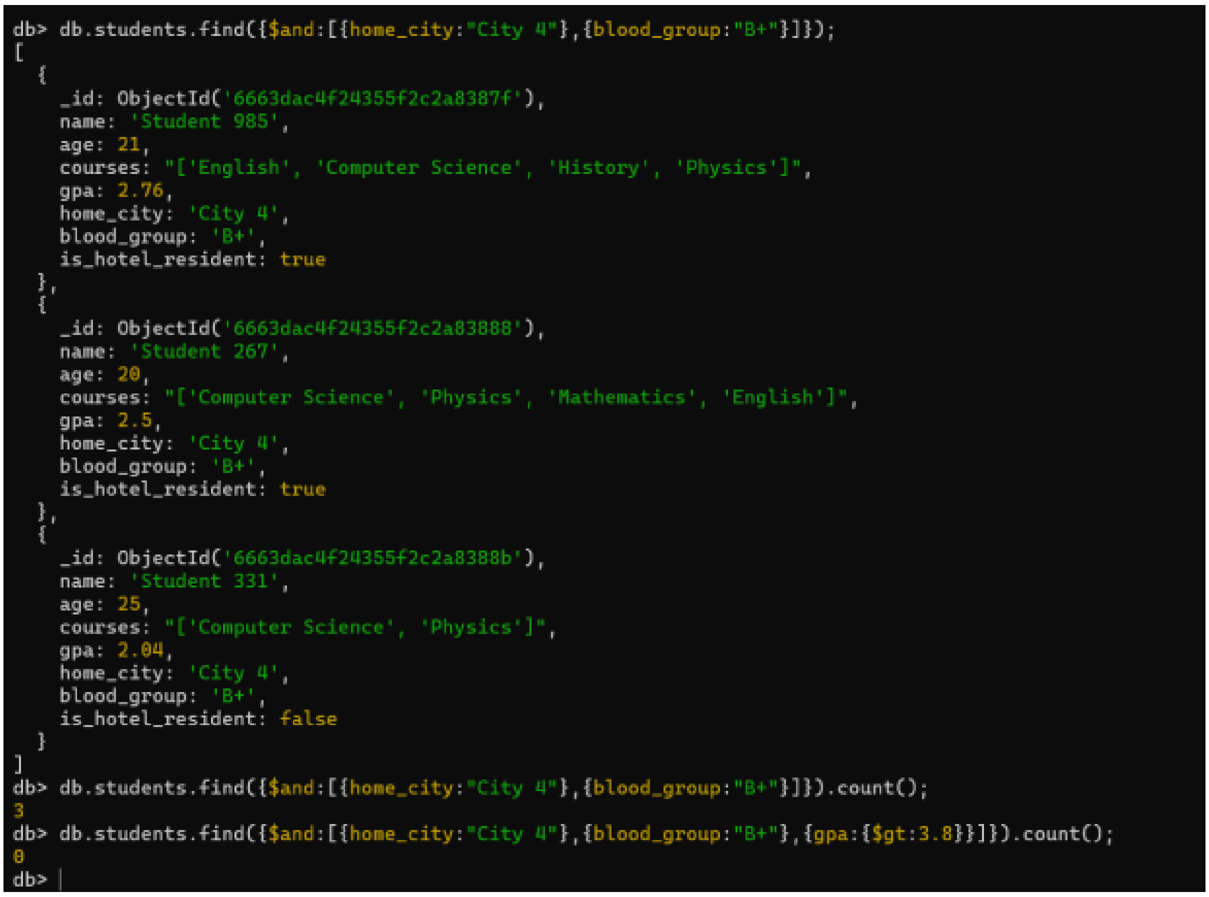
To find a students who are lived in city 4 and having a blood group “B+” we use db.students.find({$and :[{home_city:”City4”},{blood_ group: ”B”}]});

and to find its count

db.students.find({$and:[{home_city:”City4”},{blood_group:”B+”}]}).

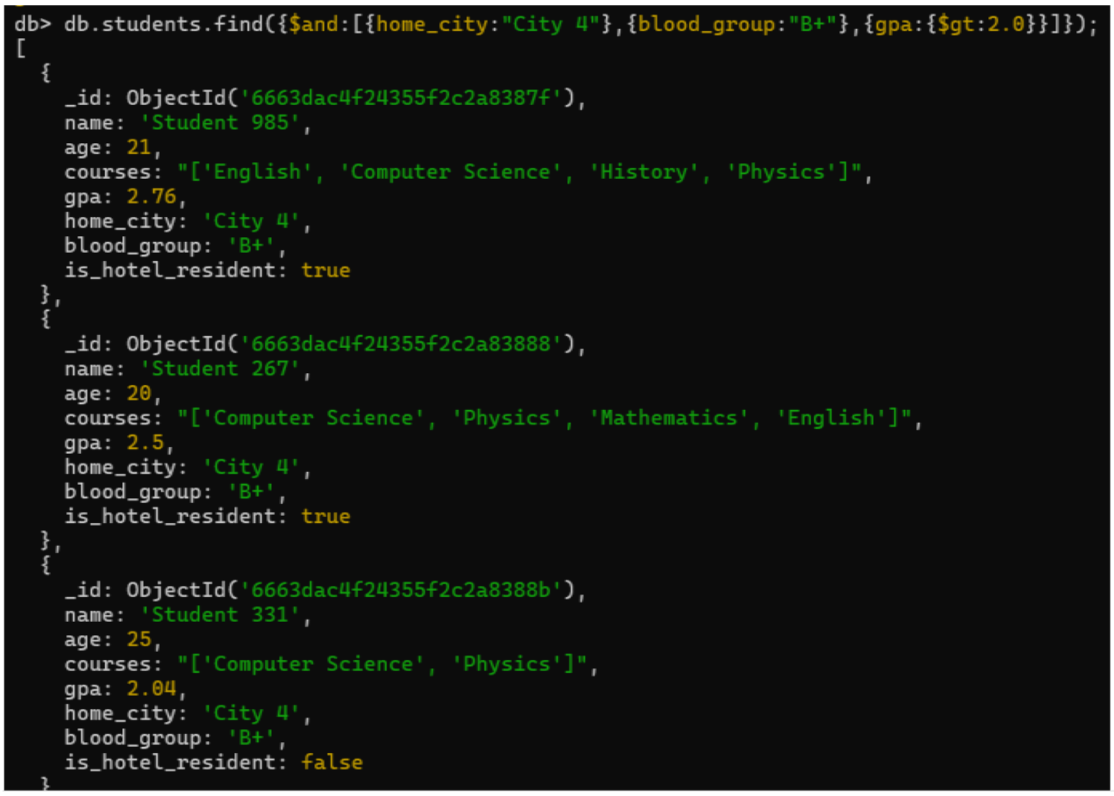
Count ();

Here we can also check a condition of students residing in City 4 having a blood group “B+” and who have gpa greater then 3.8 the result we got is



```
db> db.students.find({$and:[{home_city:"City 4"},{blood_group:"B+"}]});
[
  {
    _id: ObjectId('6663dac4f24355f2c2a8387f'),
    name: 'Student 985',
    age: 21,
    courses: "['English', 'Computer Science', 'History', 'Physics']",
    gpa: 2.76,
    home_city: 'City 4',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a83888'),
    name: 'Student 267',
    age: 20,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'English']",
    gpa: 2.5,
    home_city: 'City 4',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a8388b'),
    name: 'Student 331',
    age: 25,
    courses: "['Computer Science', 'Physics']",
    gpa: 2.04,
    home_city: 'City 4',
    blood_group: 'B+',
    is_hotel_resident: false
  }
]
db> db.students.find({$and:[{home_city:"City 4"},{blood_group:"B+"}]}).count();
3
db> db.students.find({$and:[{home_city:"City 4"},{blood_group:"B+"},{gpa:{$gt:3.8}}]}).count();
0
db> |
```

One more condition that is with a condition bit changes is made in gpa whose is greater than 2.0



```
db> db.students.find({$and:[{home_city:"City 4"},{blood_group:"B+"},{gpa:{$gt:2.0}}]});
[
  {
    _id: ObjectId('6663dac4f24355f2c2a8387f'),
    name: 'Student 985',
    age: 21,
    courses: "['English', 'Computer Science', 'History', 'Physics']",
    gpa: 2.76,
    home_city: 'City 4',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a83888'),
    name: 'Student 267',
    age: 20,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'English']",
    gpa: 2.5,
    home_city: 'City 4',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a8388b'),
    name: 'Student 331',
    age: 25,
    courses: "['Computer Science', 'Physics']",
    gpa: 2.04,
    home_city: 'City 4',
    blood_group: 'B+',
    is_hotel_resident: false
```

OR:

Given a Collection you want to FILTER a subset based on multiple conditions but Any One is Sufficient.

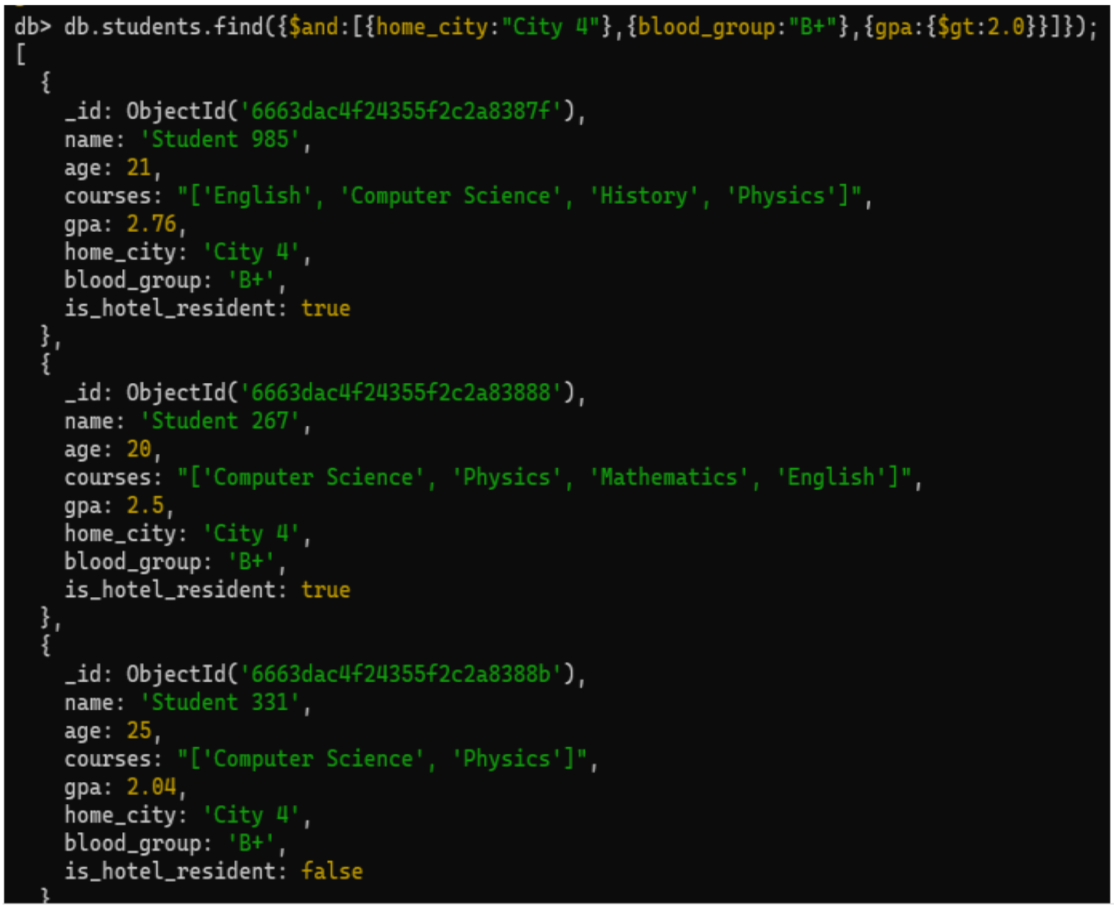
● You can use this operator in methods like find (), update (), etc. according to your requirements.

● You can also use this operator with text queries, Geo Spatial queries, and sort operations.

● When MongoDB evaluating the clauses in the $or expression, it performs a collection scan.

Here we are checking for students who are hostel resident and scored gpa less than 3.0 we use

db.students.find({$or:[{is_hotel _resident:true},{gpa:{$lt:3.000}}]});



```
db> db.students.find({$and:[{home_city:"City 4"},{blood_group:"B+"},{gpa:{$gt:2.0}}]});
[
  {
    _id: ObjectId('6663dac4f24355f2c2a8387f'),
    name: 'Student 985',
    age: 21,
    courses: "['English', 'Computer Science', 'History', 'Physics']",
    gpa: 2.76,
    home_city: 'City 4',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a83888'),
    name: 'Student 267',
    age: 20,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'English']",
    gpa: 2.5,
    home_city: 'City 4',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a8388b'),
    name: 'Student 331',
    age: 25,
    courses: "['Computer Science', 'Physics']",
    gpa: 2.04,
    home_city: 'City 4',
    blood_group: 'B+',
    is_hotel_resident: false
```

We found that out of 500 ,286 students matches for the above condition in a collection.

CRUD:

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

C- Create/ Insert

U- Update

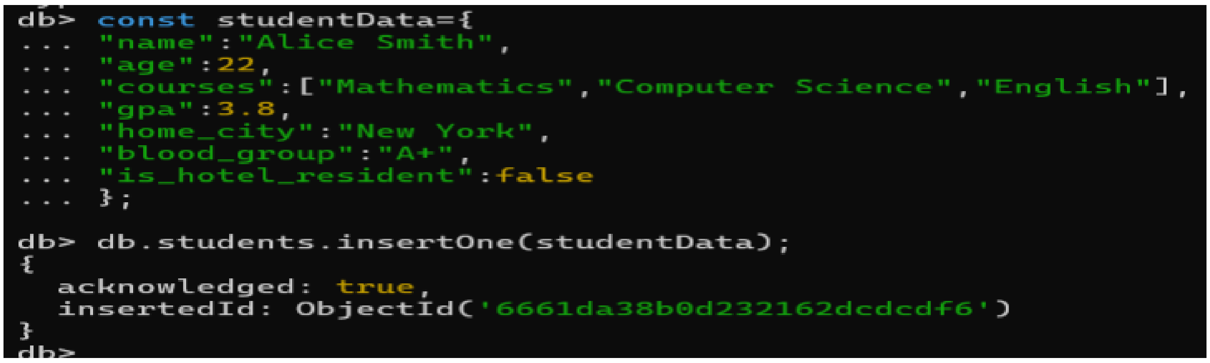
R- Remove

D- Delete

Create/Insert The "Create" or "Insert" operation in CRUD is used to add new data to a collection or table. This could be creating a new user account, adding a product to an inventory, or inserting a new row of data into a database. The "Create" operation is essential for expanding and populating a data store with new information as needed.	**Remove** The "Remove" operation allows you to retrieve data from a collection or table. This could involve querying the database to find all users with a certain email address, or pulling a specific product record.
Update The "Update" operation is used to modify existing data in a collection or table. This could include changing a user's contact information, updating a product's price, or editing a record in the database. The "Update" function is essential for maintaining accurate and up-to-date data in your system.	**Delete** The "Delete" operation allows you to remove data from a collection or table. This could involve deleting a user account, removing a product from inventory, or erasing a specific record from the database. The "Delete" function is important for cleaning up and managing the data in your system as needed.

INSERT ONE:

In MongoDB, the `insertOne` method is used to insert a single document into a collection. It takes a single parameter, which is an object representing the document to be inserted. Upon successful insertion, it returns an object with an `acknowledged` field set to true and an `insertedId` field containing the unique identifier (`ObjectId`) assigned to the newly inserted document. This method is useful for adding individual documents to a collection in MongoDB.



```
db> 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
... };

db> db.students.insertOne(studentData);
{
  acknowledged: true,
  insertedId: ObjectId('6661da38b0d232162dcdcdf6')
}
db>
```

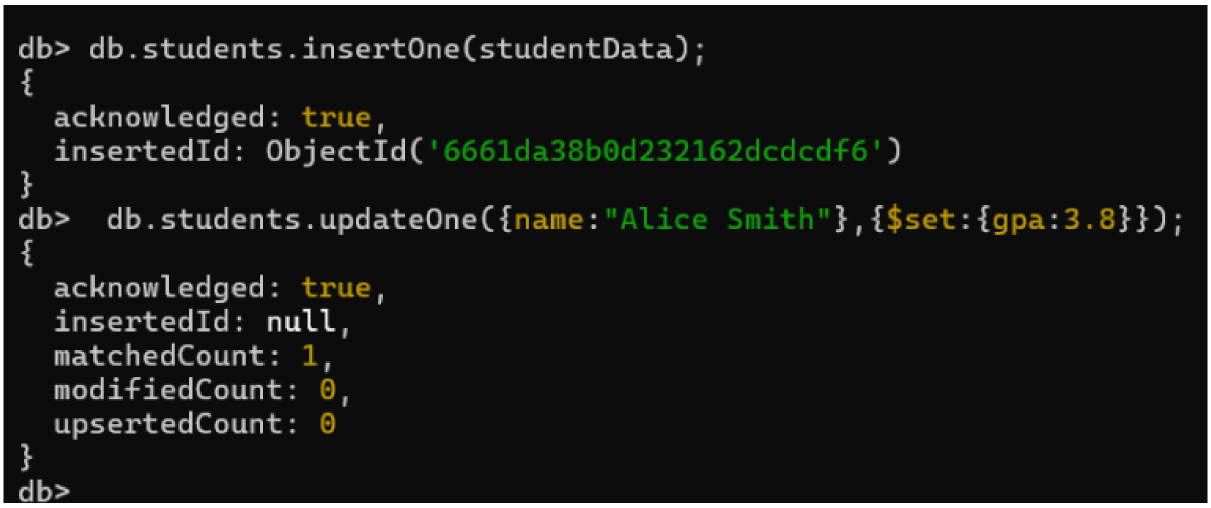
The JavaScript object `studentData` contains information about a student, including their name, age, courses, GPA, home city, blood group, and hotel residency status. Using the `insertOne` method in MongoDB, this data is inserted into the `students` collection as a single document. The `insertedId` field in the returned object confirms that the document was successfully inserted and includes the unique identifier (`ObjectId`) assigned to the newly inserted document. This identifier can be used to uniquely identify and retrieve the inserted document from the collection.

INSERT MANY:

The insertMany() method inserts one or more documents in the collection. It takes array of documents to insert in the collection.

UPDATE ONE:

In MongoDB, the `updateOne` method is used to update a single document that matches a specified filter. It takes two parameters: a filter object to identify the document to update, and an update object containing the modifications to apply. Only the first document that matches the filter is updated. The method returns an object indicating whether the update operation was acknowledged and the number of documents modified.

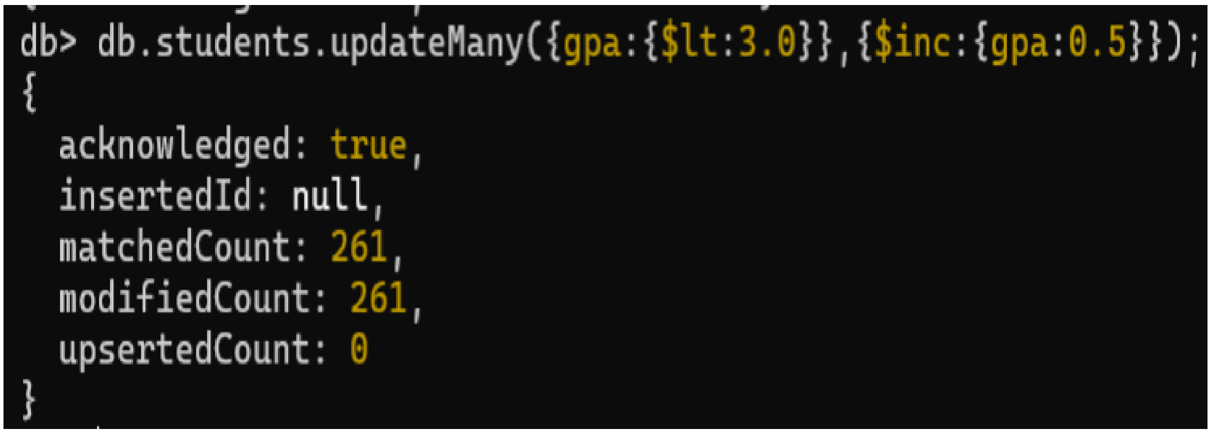


```
db> db.students.insertOne(studentData);
{
  acknowledged: true,
  insertedId: ObjectId('6661da38b0d232162dcdcdf6')
}
db>  db.students.updateOne({name:"Alice Smith"},{$set:{gpa:3.8}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
db>
```

The first MongoDB operation inserts a new document into the `students` collection using `insertOne`, which is acknowledged with `acknowledged: true` and provides the `insertedId` of the new document. The second operation uses `updateOne` to find a document with the name "Alice Smith" and update its `gpa` field to 3.8. This update operation is also acknowledged with `acknowledged: true`, and it indicates that one document was matched (`matchedCount: 1`), but no actual modifications were made (`modifiedCount: 0`) because the existing `gpa` value was already 3.8.

UPDATE MANY:

In MongoDB, the `updateMany` method is used to update multiple documents that match a specified filter. It takes two parameters: a filter object to identify the documents to update, and an update object containing the modifications to apply. All documents that match the filter will be updated. The method returns an object indicating whether the update operation was acknowledged and the number of documents modified.

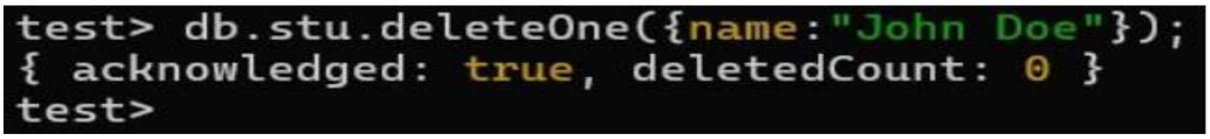


```
db> db.students.updateMany({gpa:{$lt:3.0}},{$inc:{gpa:0.5}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 261,
  modifiedCount: 261,
  upsertedCount: 0
}
```

The MongoDB operation `updateMany` is used to update multiple documents in the `students` collection where the `gpa` is less than 3.0. The update increments the `gpa` field by 0.5 for each matched document. The operation is acknowledged with `acknowledged: true` and indicates that 261 documents were matched (`matchedCount: 261`) and modified (`modifiedCount: 261`). This means that all documents with a `gpa` less than 3.0 were successfully updated to increase their `gpa` by 0.5.

DELETE ONE:

In MongoDB, the `deleteOne` method is used to delete a single document that matches a specified filter. It takes a filter object to identify the document to delete and removes only the first document that matches the filter. The method returns an object indicating whether the delete operation was acknowledged and the number of documents deleted, which is typically either 0 or 1

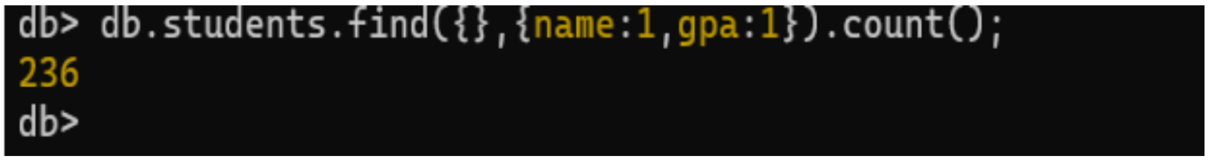


```
test> db.stu.deleteOne({name:"John Doe"});
{ acknowledged: true, deletedCount: 0 }
test>
```

PROJECTION:

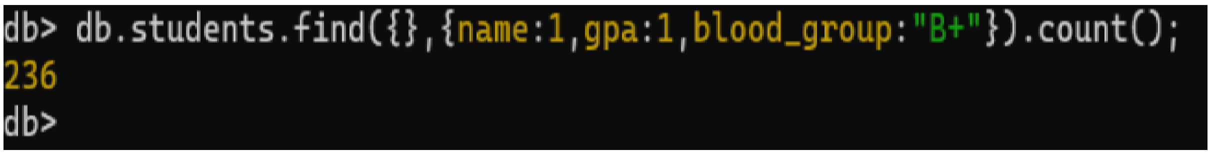
MongoDB provides a special feature that is known as Projection. It allows you to select only the necessary data rather than selecting whole data from the document.

if we use projection, only mentioned part (name and gpa)of students is displayed in the ouput .

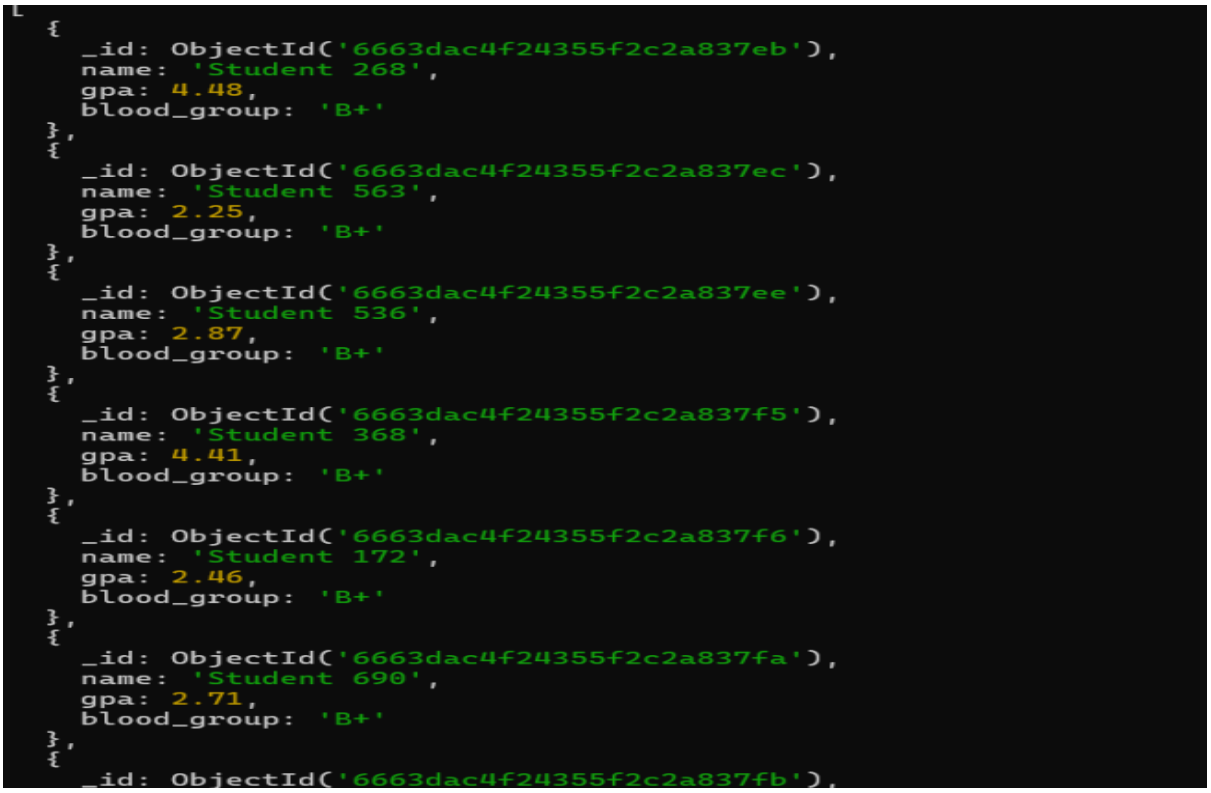


```
db> db.students.find({},{name:1,gpa:1}).count();
236
db>
```

To find only students name , gpa and blood group we use



```
db> db.students.find({},{name:1,gpa:1,blood_group:"B+"}).count();
236
db>
```



```
[
  {
    _id: ObjectId('6663dac4f24355f2c2a837eb'),
    name: 'Student 268',
    gpa: 4.48,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837ec'),
    name: 'Student 563',
    gpa: 2.25,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837ee'),
    name: 'Student 536',
    gpa: 2.87,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837f5'),
    name: 'Student 368',
    gpa: 4.41,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837f6'),
    name: 'Student 172',
    gpa: 2.46,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837fa'),
    name: 'Student 690',
    gpa: 2.71,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837fb'),
```