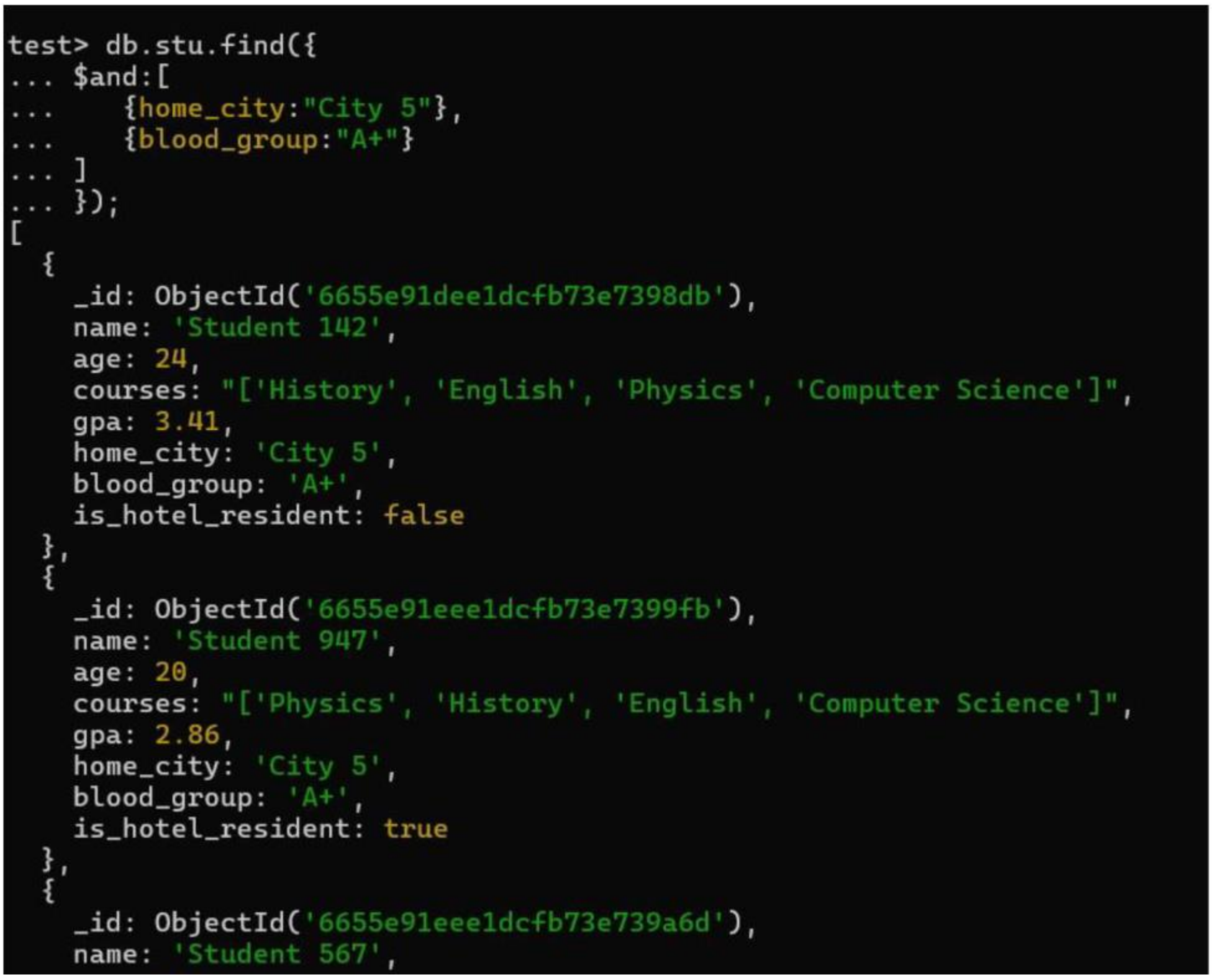
# Where, AND, OR & CRUD

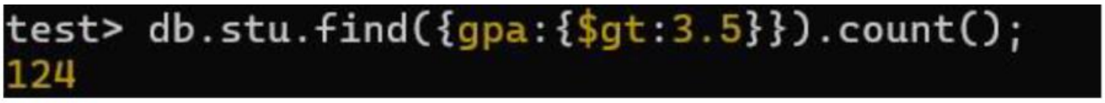
## Where:

Given a Collection you want to FILTER a subset based on a condition. That is the place WHERE is used

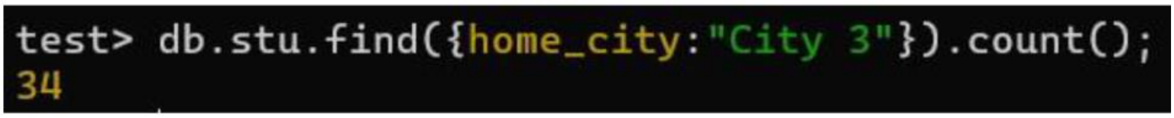
In MongoDB, the `$where` operator is indeed used to filter documents in a collection based on a condition. It allows you to specify conditions using JavaScript expressions that are evaluated for each document in the collection. However, it's important to use `$where` judiciously due to performance and security considerations.



```
test> db.stu.find({
... $and:[
...    {home_city:"City 5"},
...    {blood_group:"A+"}
... ]
... });
[
  {
    _id: ObjectId('6655e91dee1dcfb73e7398db'),
    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('6655e91eee1dcfb73e7399fb'),
    name: 'Student 947',
    age: 20,
    courses: "['Physics', 'History', 'English', 'Computer Science']",
    gpa: 2.86,
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6655e91eee1dcfb73e739a6d'),
    name: 'Student 567',
```



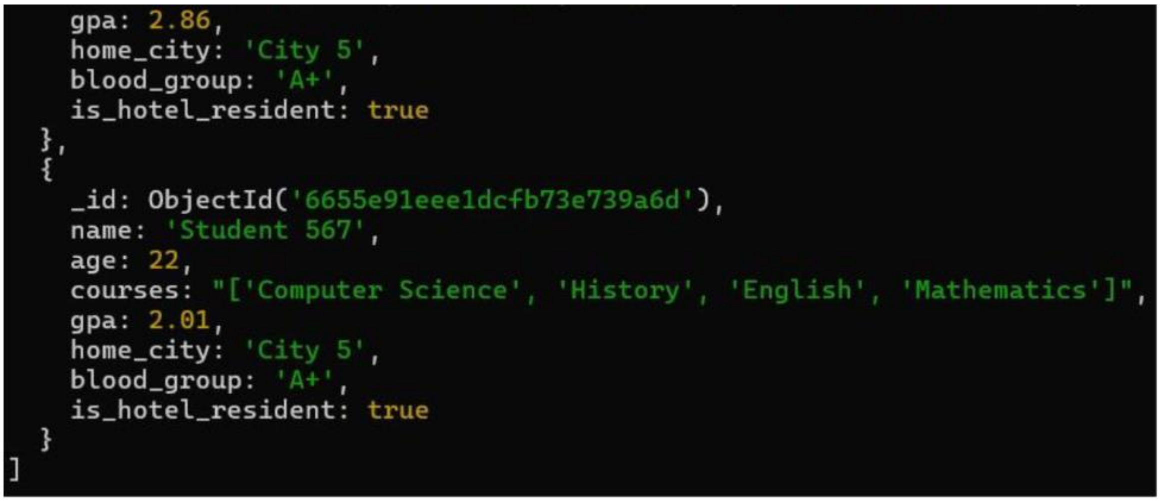
```
test> db.stu.find({gpa:{$gt:3.5}}).count();
124
```



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

AND: Given a Collection you want to FILTER a subsetbased on multiple conditions

In MongoDB, the `$and` operator is used to combine multiple query conditions, allowing you to filter documents that meet all the specified conditions. This is analogous to the logical AND operator in programming languages.

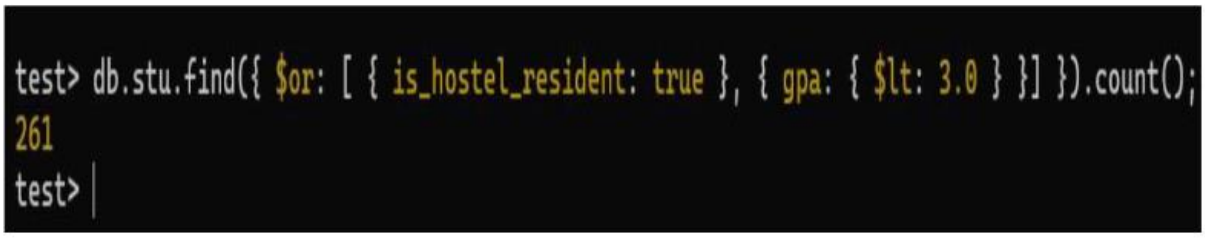


```
    gpa: 2.86,
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6655e91eee1dcfb73e739a6d'),
    name: 'Student 567',
    age: 22,
    courses: "['Computer Science', 'History', 'English', 'Mathematics']",
    gpa: 2.01,
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: true
  }
]
```

## 0R:

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

In MongoDB, the `$or` operator is used to combine multiple query conditions in such a way that a document will be returned if it satisfies at least one of the specified conditions. This is similar to the logical OR operator in many programming languages



```
test> db.stu.find({ $or: [ { is_hostel_resident: true }, { gpa: { $lt: 3.0 } }] }).count();
261
test>
```

## CRUD:

1.C - Create / Insert

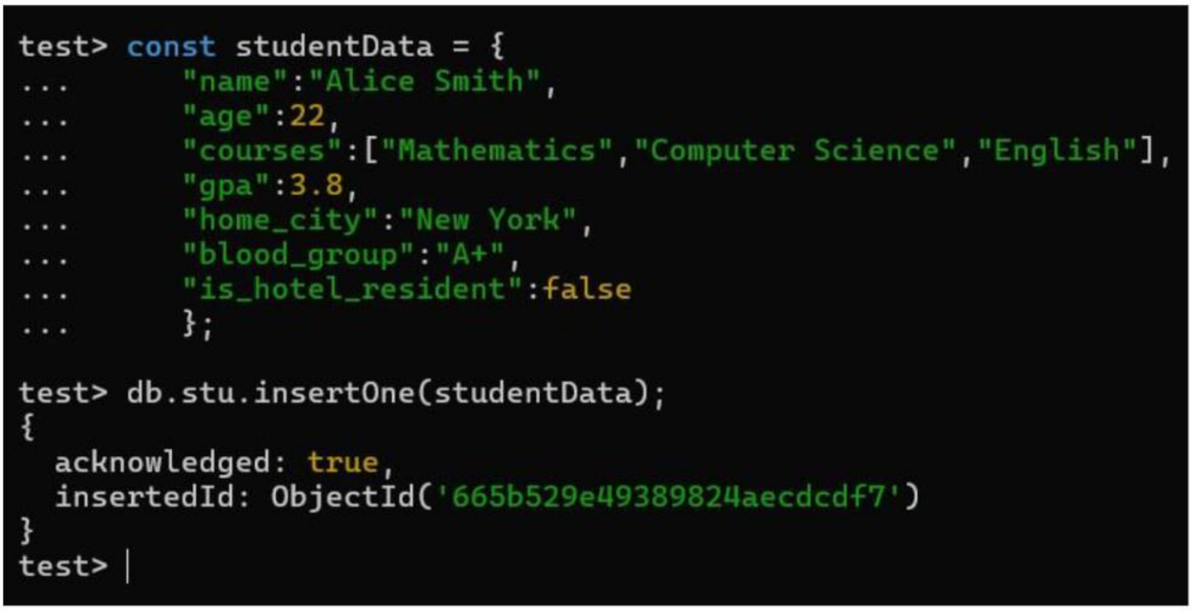
2.R - Remove

3.U – update

4.D – Delete

## INSERT:

The insert function in MongoDB is used to add documents to a collection. In the context of MongoDB, a document is a set of key-value pairs (similar to a JSON object), and a collection is a group of documents.

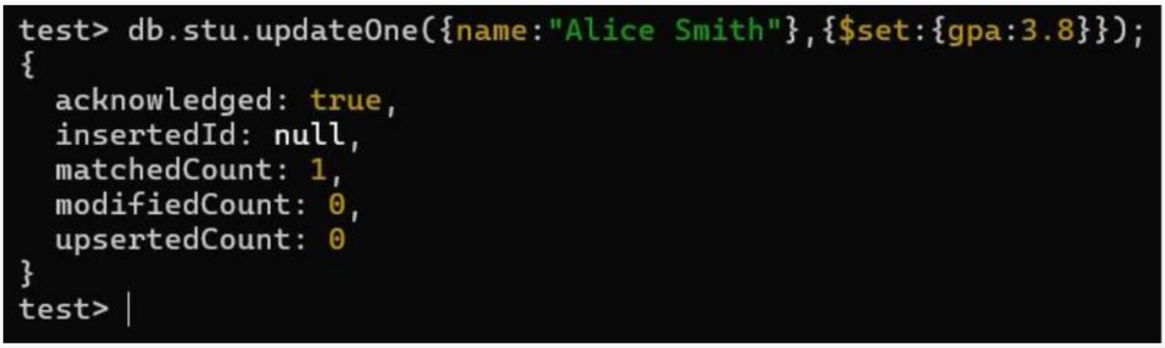


```
test> const studentData = {
...       "name":"Alice Smith",
...       "age":22,
...       "courses":["Mathematics","Computer Science","English"],
...       "gpa":3.8,
...       "home_city":"New York",
...       "blood_group":"A+",
...       "is_hotel_resident":false
...       };

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

## UPDATE:

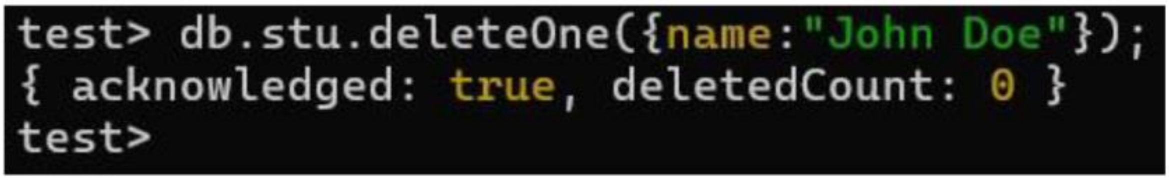
In MongoDB, the `update` operations are used to modify documents in a collection. MongoDB provides several methods and operators to update documents, allowing you to update one or more fields, entire documents, or multiple documents in a collection



```
test> db.stu.updateOne({name:"Alice Smith"},{$set:{gpa:3.8}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
test> |
```

## DELETE:

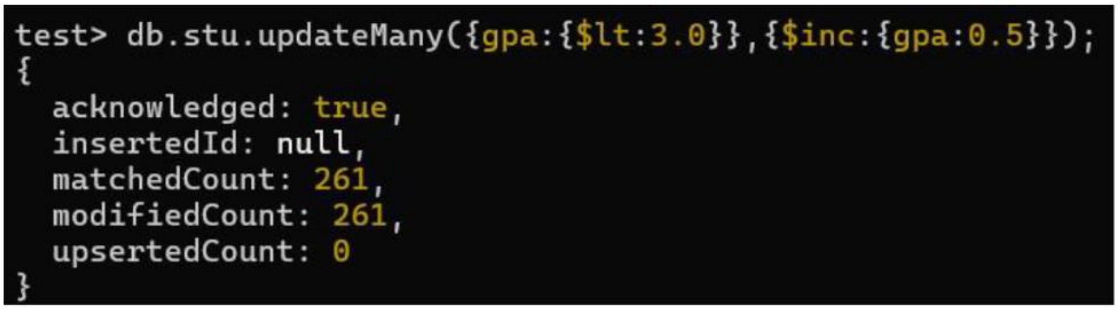
delete operations (`deleteOne` and `deleteMany`) provide powerful and flexible ways to MongoDB's remove documents from a collection. These operations should be used with care, especially when deleting multiple documents or performing bulk deletions. Understanding and using the appropriate delete method for your specific use case can help manage and maintain your database efficiently



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

## UPDATE MANY:

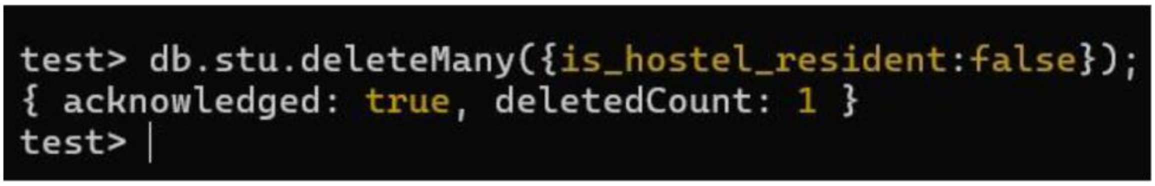
The update Many function in MongoDB is used to update multiple documents in a collection that match a given filter. This function is particularly useful when you need to modify several documents at once based on a specific condition.



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

## DELETE MANY:

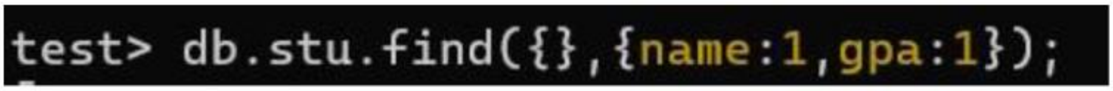
The `deleteMany` method in MongoDB is a powerful tool for bulk deleting documents based on specified criteria. Proper use of this method can help you efficiently manage and maintain your collections. Always use `deleteMany` with caution, especially with broad filters, to avoid unintended data loss. Ensuring proper indexing and considering the impact of the deletion operation are key to effectively using this method.



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

PROJECTION:

Projection in MongoDB allows you to control which fields are included or excluded from query results. By specifying the fields to include or exclude, you can tailor query results to meet your specific requirements, improve query performance, and reduce network overhead. Understanding how to use projection effectively can help you optimize your MongoDB queries and improve the efficiency of your applications.



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

## Practical Considerations

1. **Impact on Performance**: Projection can significantly improve query performance by reducing the amount of data transmitted over the network and the amount of memory required to process query results.
2. **Indexing**: Ensure that fields used in the filter condition are indexed to optimize query performance, especially when projection is applied.
3. **Atomicity**: Projection is applied after the documents are retrieved from the database but before they are returned to the client. This means that projection does not affect the atomicity of the query operation.
4. **Readability**: Projection can enhance the readability of query results by returning only the relevant fields.