Add, Update, and Delete Data:

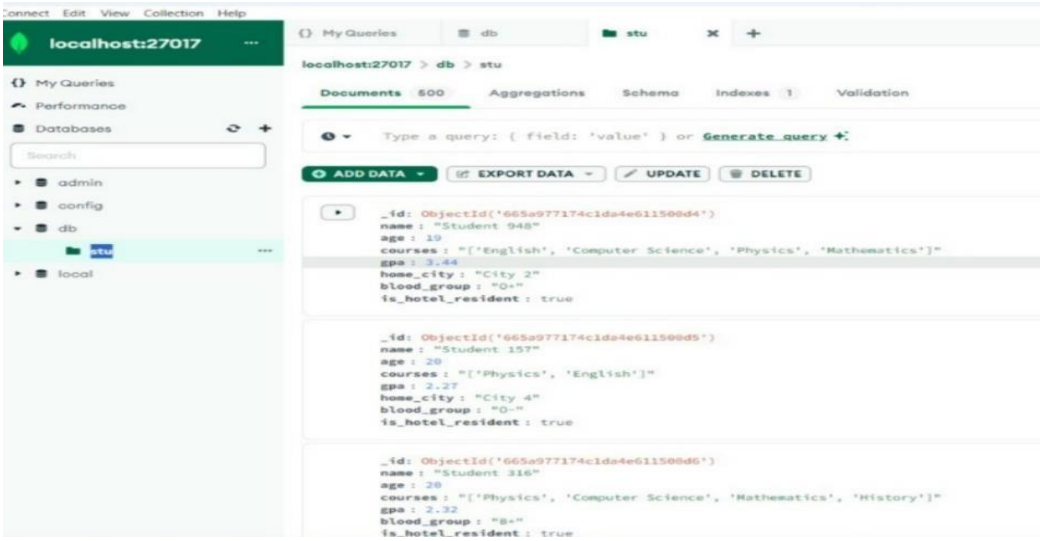
First step is we want to switch our database to the given collection by using command.

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

Now database is switched to db .

To find the data is present in the given set we can use “show collections “

```
test> use db
switched to db db
db> show collections
stu
stud
```



In the above example collections name is stu

If we give “show dbs”it show all the database.

```
]
Type "it" for more
db> show dbs
admin   40.00 KiB
config  84.00 KiB
db      56.00 KiB
local   40.00 KiB
db>
```

If we want to know the total number of collections we have to give db.stu .find().count()

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

If we want to find the collections use this command

db.stu.find()

```
b> 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+',
```

Collections:

MongoDB database stores its data in collections. A collection holds one or more BSON documents. Documents are analogous to records or rows in a relational database table. Each document has one or more fields; fields are similar to the columns in a relational database table.

Database: A database is an organized collection of data stored in a computer system and usually controlled by a database management system (DBMS). The data in common databases is modeled in tables, making querying and processing efficient. Structured query language (SQL) is commonly used for data querying and writing.

Data type:

MongoDB Server stores data using the BSON format which supports some additional data types that are not available using the JSON format.

WHERE,AND,OR &CURD:

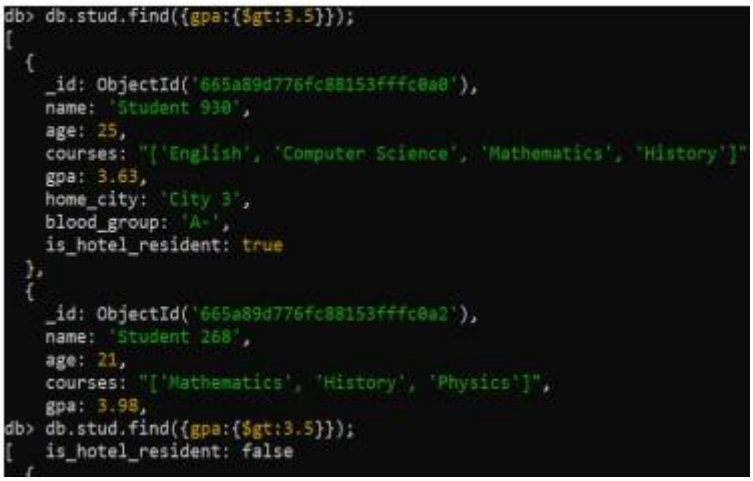
WHERE:

Given a collection you want to filter a subset based on a condition. That is the place WHERE isused.

to find the collections which is greater than 3.5 we use this command

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

$gt------ greater than



The gpa having greater than 3.5 will appear in the output window.

AND:

Given a collection you want to filter a subset based on multiple conditions.

To find all students who live in “City 5” AND have a based group of “A+”

We use this command to find the given all students who lived in “city 5” and group “A+”.

```
Db.stud.find({
$and: [
 {home_city : “City 5”},
 {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
  },
```

Above example is filtered based up on some conditions like:'home_city:'City5' and 'blood_group:A+'.
Here the output will give the students who are living in "city 5" and group "A+".

OR:

In the given collection that is student we want to filter a subset based on multiple conditions but any one is sufficient.

To find all the students who are all having either of the blood_group:"A+" or $gpa:"3.5".

```
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
```

$lt-represents less than

CURD:

C-Create/Insert

R-Remove

U-Update

D-Delete

INSERT:

This command will show how to insert a data to the collections.

Const studentData={

“name”: “Jam”,

“age”:12,,

“courses”:[“CS”, “MATHS”, “KANNADA”],

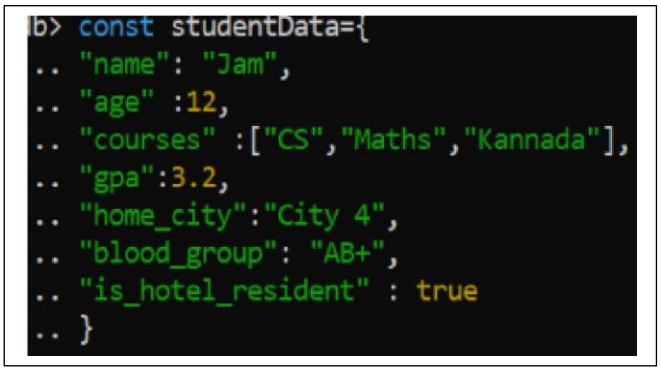
“gpa”:3.2,

“home_city”: “City 4”,

“blood_group”: “AB+”,

“is_hotel_resident”:true

}



In the above example we insert a name and other information of “Jam” to the database. The collection of database called studentData.In the above example insertion take place only one time but we can insertan information for many times to the collcetions.

Update:

Update cpmmand is used to update the new data to the dataset.

$set is used to update the command.

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

Figure shows it update the exist database and new collections was added to the database.

Delete:

Delete command is used to delete the data from the present collections.

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

Here the data “Sam” will be deleted from the present database.

Update many:

Here update many is used to update many student having the gpa lessthan 3.0 by increasing it by 0.5

```
// Update all students with a GPA less than 3.0 by increasing it by 0.5
db.students.updateMany({ gpa: { $lt: 3.0 } }, { $inc: { gpa: 0.5 } });
```

DeleteMany:

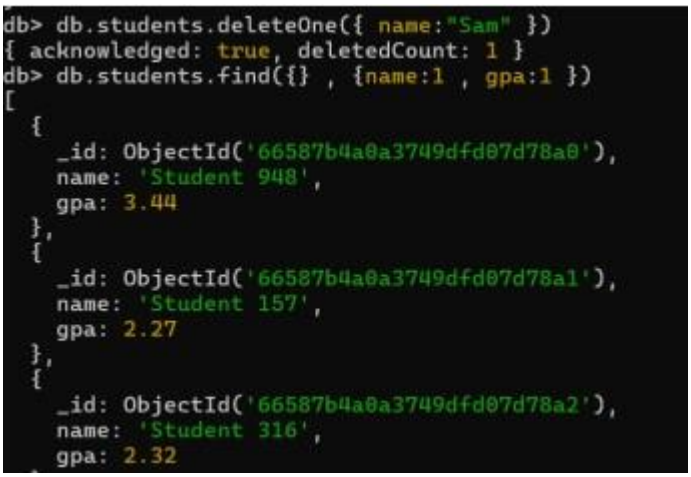
DeleteMany is used to delete all the students who are belongs th the group.

```
// Delete all students who are not hotel residents
db.students.deleteMany({ is_hotel_resident: false });
```

Projection:

Projection is a powerfull tool that can be used to extract only the fields you need from a document—not all fields.

This is used when we don't need all columns or attributes.



In the above example it shows only the name and gpa, because the command is given as 'name:1' and 'gpa:1'.

o Get Selected Attributes:

In the given collection if we want to FILTER a subset of attribute. That is the region where the projection is used.

o Ignore Attributes:

This attributes is used to print the exact data by excluding the object_id. Here _id is Used to find the exact student rather than searching here and there in the database It just saw the _id and find the specific group.

Benefits of Projection:

➢ Reduced data transferred between the database and your application.

➢ Simplifies your code by focusing on the specific information you need.

➢ Improve query performance by retrieving only necessary data.

LIMIT AND SELECTIONS:

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)

```
db> db.students.find({} , {_id:0}).limit(5)
[
  {
    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
  },
  {
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
```

The above example gives the complete information about limit, Here we take _id as zero because To prevent the _id numbers and we restricted or we use the limit to print only the minimum number of characters in the above example we mentioned only 3 so it prints the collections upto three collections.

• Selectors:

➢ Comparison gt and lt

➢ AND operator

➢ OR operator

Comparision gt lt:

To find the students who are greater than 20 we use this command.

This is used to find the database greaterthan or lessthan

gt-greaterthan

lt-lessthan

Here the example to find all the students whose age is lessthan 30.

```
b> db.stud.find({age:{$gt:20}});

{
  _id: ObjectId('665a89d776fc88153fffc09f'),
  name: 'Student 346',
  age: 25,
  courses: "['Mathematics', 'History', 'English']",
  gpa: 3.31,
  home_city: 'City 8',
  blood_group: 'O-',
  is_hotel_resident: true
},
{
  _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('665a89d776fc88153fffc0a1'),
  name: 'Student 305',
  age: 24,
  courses: "['History', 'Physics', 'Computer Science', 'Mathematics']",
  gpa: 3.4,
  home_city: 'City 6',
  blood_group: 'O+',
  is_hotel_resident: true
```

Here the output will shown the students whose age is lessthan 30.

AND operator:

AND operation is used to find the specific details about the collection. Here is some Examples on AND operation

And operator is used to find the student from “city 2”with blood_group “B+”.

```
db.stud.find({
$and:[
{home_city: "City 2"},
{blood_group: "B+" }
]
});


_id: ObjectId('665a89d776fc88153fffc0b4'),
name: 'Student 504',
age: 21,
courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
gpa: 2.42,
home_city: 'City 2',
blood_group: 'B+',
is_hotel_resident: true
},
{
_id: ObjectId('665a89d776fc88153fffc0eb'),
name: 'Student 367',
age: 19,
courses: "['English', 'Physics', 'History', 'Mathematics']",
gpa: 2.81,
home_city: 'City 2',
blood_group: 'B+',
is_hotel_resident: false
```

In the above given example shows the collections of the students who are from the city 2 and the bloodGroup are of type B+. This can be easily done by using the AND operation

OR operation:

The OR operation can be explained by using the following example,here we take a example to find the Student who are hostel residents OR have a GPA less than 3.0 It means it takes either the students how are hostel residents or the students whose gpa is less than 3.0.

```
db> db.stud.find({
... $or:[
... {is_hostel_resident:true},
... {gpa:{$lt:3.0}}
... ]
... });
[
  {
    _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('665a89d776fc88153fffc0a3'),
    name: 'Student 563',
    age: 18,
    courses: "['Mathematics', 'English']",
    gpa: 2.25,
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('665a89d776fc88153fffc0a4'),
    name: 'Student 440',
    age: 21,
```

Here we take the example that wants to give the output as the students who are hostel residents OR the students whose gpa is less than 3.0