WHERE,AND,OR AND CRUD

WHERE:

To filter a subset based on condition **where** is used.

To find the collection with gpa greater than 2.0 we use a command

here, gt→greater than

db.students.find({gpa:{$gt:2.0}});

```
db> db.students.find({gpa:{$gt:2.0}});
[
  {
    _id: ObjectId('6663dac4f24355f2c2a837e5'),
    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('6663dac4f24355f2c2a837e6'),
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837e7'),
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.32,
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837e8'),
    name: 'Student 346',
    age: 25,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 3.31,
    home_city: 'City 8',
    blood_group: 'O-',
    is_hotel_resident: true
  },
```

This is how an output is formed.we can analyse this by checking it with data stored in document and collection imported into mongocompass.

Can also try to find collection whose gpa is less than 3.9 by a command here, lt→is lesser than.

db.students.find({gpa:{$lt:3.9}});

```
db> db.students.find({gpa:{$lt:3.9}});
[
  {
    _id: ObjectId('6663dac4f24355f2c2a837e5'),
    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('6663dac4f24355f2c2a837e6'),
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837e7'),
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.32,
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837e8'),
    name: 'Student 346',
    age: 25,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 3.31,
    home_city: 'City 8',
    blood_group: 'O-',
    is_hotel_resident: true
  },
```

To know the countings of students who have got less than 3.9 gpa we use

db.students.find({gpa:{$lt:3.9}}).count();

```
db> db.students.find({gpa:{$lt:3.9}}).count();
473
```

We can say that out of 500 ,473 students got lesser than 3.9 gpa.

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();

```
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>
```

Here also checked 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 0

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 want to filter a subset based on multiple conditions but any one is sufficient among them **OR** is used.

Here we are checking for students who are hotek 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({$or:[{is_hotel_resident:true},{gpa:{$lt:3.0}}]});
[
  {
    _id: ObjectId('6663dac4f24355f2c2a837e5'),
    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('6663dac4f24355f2c2a837e6'),
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837e7'),
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.32,
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837e8'),
    name: 'Student 346',
    age: 25,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 3.31,
    home_city: 'City 8',
```

here to find students having either blood group “A-“ and gpa less than 3.0 used a command

db.students.find([$or:[{blood_group:”A-“},{gpa:{$lt:3.0}}]]);

to find countings we use

db.students.find([$or:[{blood_group:”A-“},{gpa:{$lt:3.0}}]]).count();

```
db> db.students.find({$or:[{blood_group:"A-"},{gpa:{$lt:3.0}}]}).count();
286
db> db.students.find({$or:[{blood_group:"A-"},{gpa:{$lt:3.0}}]});
[
  {
    _id: ObjectId('6663dac4f24355f2c2a837e6'),
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837e7'),
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.32,
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6663dac4f24355f2c2a837e9'),
    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('6663dac4f24355f2c2a837ec'),
    name: 'Student 563',
    age: 18,
```

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

CRUD:

C→Create/Insert

R→Remove

U→Update

D→Delete

CREATE: For the imported collection need to add some more details of a particular individual so first should create a data of a student and then insertion has been made .

1.Const studentsData={“name”:”Meghana”,

“age”:20,

“courses”:[“Mathematics”,”Computer Science”,”English”],

“gpa”:3.8,

“home_city”:”New York”,

”blood_group”:”A+”,

is_hotel_resident”:false};

Data is defined now to **insert** we use a command

db.students.insertOne(studentsData);

To count we use a command

db.students.insertOne(studentsData).count();

2.Const studentsData={“name”:”Shaila”,

“age”:20,

“courses”:[“Mathematics”,”Computer Science”,”English”],

“gpa”:3.4,

“home_city”:”New York”,

”blood_group”:”A+”,

is_hotel_resident”:false};

```
db> const studentsData = { "name": "Meghana", "age": 20, "courses":
 ["Mathematics", "Computer Science", "English"], "gpa": 3.8, "home_
city": "New York", "blood_group": "A+", "is_hotel_resident": false
};

db> db.students.insertOne(studentsData);
{
  acknowledged: true,
  insertedId: ObjectId('6663eebbaa55c98dbbcdcdf7')
}
db> const studentsData = { "name": "Shaila", "age": 20, "courses":
["Mathematics", "Computer Science", "English"], "gpa": 3.4, "home_c
ity": "New York", "blood_group": "B+", "is_hotel_resident": false }
;

db> db.students.insertOne(studentsData);
{
  acknowledged: true,
  insertedId: ObjectId('6663ef29aa55c98dbbcdcdf8')
}
db>
```

here, three more students data is inserted into to already existing document to check total countings of collection we use a command

db.students.count();

```
}
db> const studentsData = { "name": "Meghana", "age": 20, "courses":
 ["Mathematics", "Computer Science", "English"], "gpa": 3.8, "home_
city": "New York", "blood_group": "A+", "is_hotel_resident": false
};

db> db.students.insertOne(studentsData);
{
  acknowledged: true,
  insertedId: ObjectId('6663eebbaa55c98dbbcdcdf7')
}
db> const studentsData = { "name": "Shaila", "age": 20, "courses":
["Mathematics", "Computer Science", "English"], "gpa": 3.4, "home_c
ity": "New York", "blood_group": "B+", "is_hotel_resident": false }
;

db> db.students.insertOne(studentsData);
{
  acknowledged: true,
  insertedId: ObjectId('6663ef29aa55c98dbbcdcdf8')
}
db> db.students.count();
DeprecationWarning: Collection.count() is deprecated. Use countDocu
ments or estimatedDocumentCount.
503
db>
```

UPDATE: To update a name of a student and gpa we use

db.students.updateOne({name:”Meghana”},{$ser:{gpa:3.8}});

```
db> db.students.updateOne({name:"Meghana"},{$set:{gpa:3.8}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
db>
```

db.students.updateOne({name:”Shaila”},{$ser:{gpa:3.4}});

```
db> db.students.updateOne({name:"Shaila"},{$set:{gpa:3.4}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
db>
```

Both students data is uploaded and updated into a collection so otput is acknowleged as :true

DELETE: To delete a student need to enter a name of a individual so its been information has been deleted .we use

db.students.deleteOne({name:”Meghana”});

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

After deleting the count is

db.students.count()

```
db> db.students.count();
503
db> db.students.deleteOne({name:"Meghana"});
{ acknowledged: true, deletedCount: 1 }
db> db.students.count();
502
```

The total number of students in a collection is503 now one student data deleted so counting is updated that is 502

UPDATE MANY: update some aspects(saved information) of a students

To update data of students with a gpa greater than 3.5 by increasing 0.5 we use

db.students.updateMany({gpa:3.5}},{$inc:{gpa:0.5}});

```
db> db.students.updateMany({gpa:{$gt:3.5}},{$inc:{gpa:0.5}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 124,
  modifiedCount: 124,
  upsertedCount: 0
}
db>
```

DELETE MANY: To delete data of students who are hotel_residents we use a command

db.students.deleteMany({is_hotel_resident:true});

and the countings of deleted students is 246.

```
db> db.students.deleteMany({is_hotel_resident:true});
{ acknowledged: true, deletedCount: 246 }
db>
```

Delete all the students who's blood group "A-" we us

db.students.deleteMany({blood_group:"A-"});

```
db> db.students.deleteMany({blood_group:"A-"});
{ acknowledged: true, deletedCount: 20 }
db>
```