MONGODB

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.

```
db> db.students.find({ gpa:{$gt:3}})
[
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a0'),
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Phy
sics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a3'),
    name: 'Student 346',
    age: 25,
    courses: "['Mathematics', 'History', 'English'
]",
    gpa: 3.31,
    home_city: 'City 8',
    blood_group: 'O-',
    is_hotel_resident: true
  },
```

In this example we use the condition gpa greater than 3. So the result is shown above is based on this condition

Here '$gt' means greater than

AND

Given a Collection you want to FILTER a subset based on multiple conditions

```
Type "it" for more
db> db.students.find({
... $and:[
... {home_city:"City 1"},
... {blood_group:"O-"}
... ]
... })
[
  {
    _id: ObjectId('66587b4a0a3749dfd07d78c0'),
    name: 'Student 384',
    age: 18,
    courses: "['Mathematics', 'Computer Science']"
,
    gpa: 3.9,
    home_city: 'City 1',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('66587b4a0a3749dfd07d7950'),
    name: 'Student 702',
    age: 22,
    courses: "['History', 'Mathematics', 'English'
]",
    gpa: 3.74,
    home_city: 'City 1',
    blood_group: 'O-',
    is_hotel_resident: false
  },
```

Above example is filtered based on 2 condition i.e

'home_city: Cityl' and 'blood group: O-

OR

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

```
db> db.students.find({
... $or:[
... {blood_group:"O+"},
... {gpa:{$lt:3.5}}
... ]
... })
[
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a0'),
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Phy
sics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a1'),
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
```

Above example, the students database is filtered based on either 'blood_group: O+' or 'gpa less than 3.5'

NOTE: $It is less than.

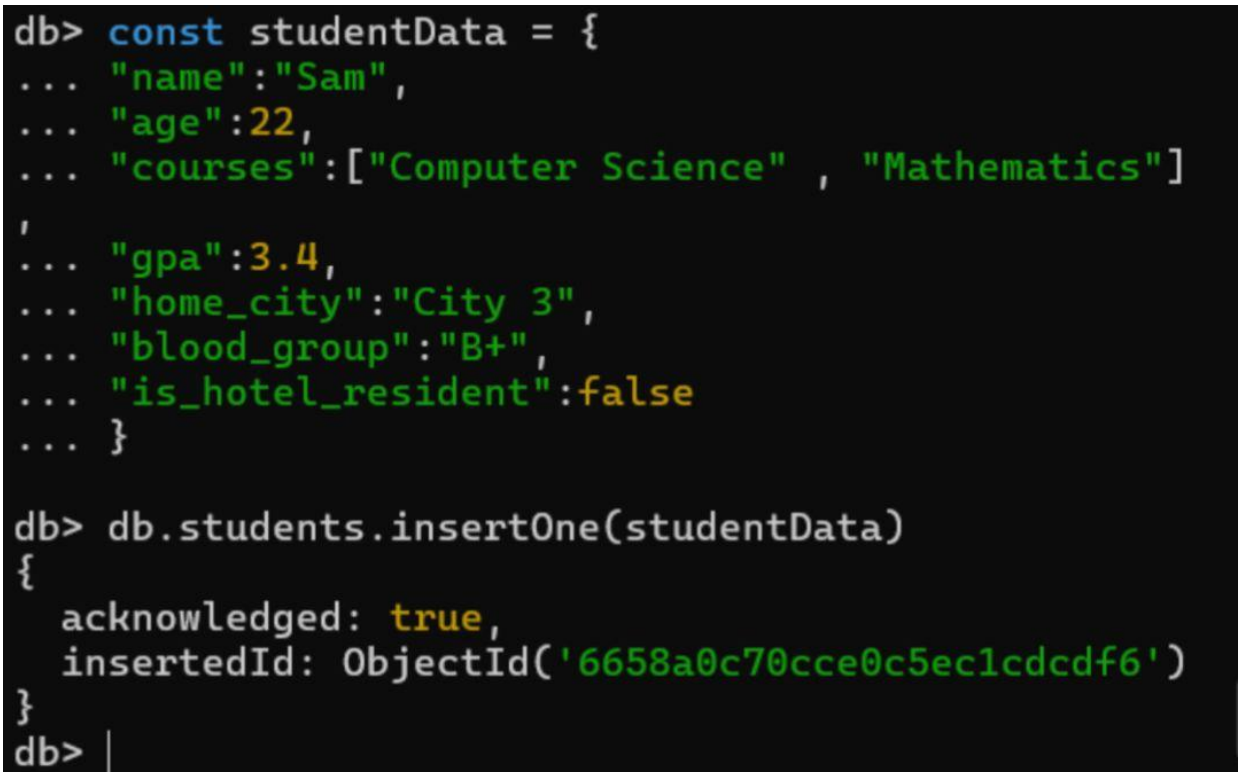
CRUD

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

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

Create/Insert

- To create a new document in a collection, use the `insertOne` method for inserting a single document or `insertMany` for inserting multiple documents at once.



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

Update

As information changes, you'll need to update existing documents. MongoDB provides methods like `updateOne` (for a single document) and `updateMany` (for multiple documents) to modify specific fields.

```
// Find a student by name and update their GPA
db.students.updateOne({ name: "Alice Smith" }, { $set: { gpa: 3.8 } });
```

In the above example we can able to “updateOne”

```
// 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 } });
```

In the above example we can able to update many time “updateMany”

Delete

When data is no longer needed, you can delete documents. Similar to updates, `deleteOne` removes a single document based on criteria, while `deleteMany` removes multiple documents.

```
// Delete a student by name
db.students.deleteOne({ name: "John Doe" });
```

In the above example we can able to “deleteOne”

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

In the above example we can able to delete many times “deleteMany”

PROJECTION

In MongoDB, projections are a powerful way to control which fields get returned when you query your collections. This allows you to optimize your queries by fetching only the data you actually need, reducing network traffic and improving performance.

```
// Get only the name and gpa for all students
db.students.find({}, { name: 1, gpa: 1 });

// Exclude the "_id" field from all queries by default
db.students.find({}, { _id: 0 });
```