WHERE , AND,OR & CRUD

LOAD THE DOCUMENT

#DOWNLOAD THE student csv from this link

#IMPORT THE DATA TO THE COLLECTION CTREATED DATA

WHERE:

In MongoDB, the "where" operator is not used like in traditional SQL databases. Instead, MongoDB uses the "find" method to query documents in a collection based on specific criteria. You can specify the criteria inside the "find" method to filter the results you want. It's a bit different from SQL, but once you get the hang of it, it's pretty cool! Let me know if you need more info on querying in MongoDB.

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.

EXAMPLE:

```
db.products.find({
  $where: function() {
    // Calculate total price with discount
    const discount = this.price * this.discount / 100;
    const totalPrice = this.price- discount;

    // Return documents where total price is greater than 100
    return totalPrice > 100;
  }
})
```

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

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

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

AND:

In MongoDB, you can use the "and" operator by specifying multiple conditions within the same query. To do this, you can use the "$and" operator, which allows you to combine multiple expressions. Each expression inside the "$and" operator must evaluate to true for the document to be included in the result set.

```
db> db.students.find({ $and: [{ gpa:{$lt:3.0} }, { blood_group: "B+" }] }).count();
16
db>
```

OR:

In MongoDB, you can use the "or" operator to query documents based on multiple conditions where at least one of the conditions needs to be true for the document to be included in the result set. To use the "or" operator, you can use the "$or" operator in your query.

```
db> db.students.find({ $or: [{ gpa:{$lt:3.0} }, { blood_group: "B+" }] }).count();
140
db>
```

CRUD:

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

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

INSERT:

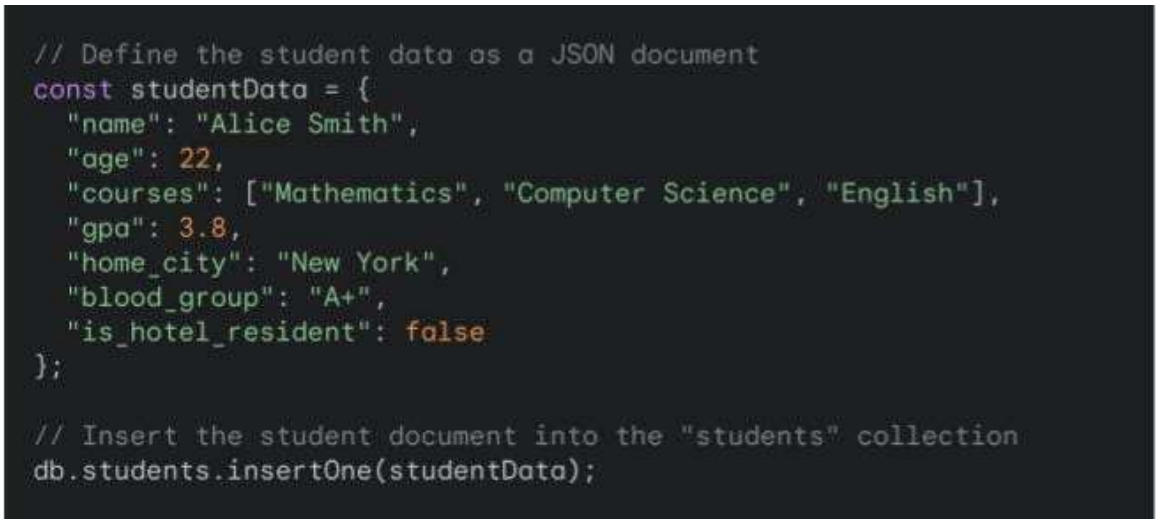
In MongoDB, to insert data into a collection, you can use the "insertOne" or "insertMany" methods.

If you want to insert a single document, you can use the "insertOne" method like this:

```
db.collection.insertOne({ key: value, key2: value2, ... })
```

If you want to insert multiple documents at once, you can use the "insertMany" method like this:

```
db.collection.insertMany([
  { key: value, key2: value2, ... },
  { key: value, key2: value2, ... },
  ...
])
```



```
// Define the student data as a JSON document
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
};

// Insert the student document into the "students" collection
db.students.insertOne(studentData);
```

```
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:

n MongoDB, to update documents in a collection, you can use the "updateOne" or "updateMany" methods.

If you want to update a single document, you can use the "updateOne" method like this:

db.collection.updateOne({ filter criteria }, { $set: { key: value, key2: value2, ... } })

If you want to update multiple documents that match a filter, you can use the "updateMany" method like this:

db.collection.updateMany({ filter criteria }, { $set: { key: value, key2: value2, ... } })

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

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

DELETE:

To delete documents in MongoDB, you can use the "deleteOne" or "deleteMany" methods.

If you want to delete a single document, you can use the "deleteOne" method like this:

db.collection.deleteOne({ filter criteria })

If you want to delete multiple documents that match a filter, you can use the "deleteMany" method like this:

db.collection.deleteMany({ filter criteria })

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

UPDATE MANY:

To update multiple documents in MongoDB, you can use the "updateMany" method. This method allows you to update all documents in a collection that match a specified filter.

Here's an example of how you can use the "updateMany" method in MongoDB:

```
db.collection.updateMany(
{ filter criteria },
{ $set: { key: value, key2: value2, ... } }
)
```

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

PROJECTION:

The project operator in MongoDB is used to specify or restrict the fields that should be returned in the query results. It allows you to control which fields are included or excluded in the output documents.

```
// 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 });
```