CLASS 2:-ADD,UPDATE & DELETE

FEW COMMANDS TO TEST AFTER CONNECTION:-

1. Show dbs :-

   All databases are shown

   Expected output:-

```
admin 40.00 KiB
config 72.00 KiB
db 128.00 KiB
local 40.00 KiB
```

These are default databases (`admin`, `config`, and `local`) that MongoDB creates for administrative purposes. Any additional databases you've created or that have been created by applications will also appear in this list.

2. use db:-

   the `use` command is used to switch to a specific database within the MongoDB shell

   expected output:-

```
switched to db db
```

3. show collections:-
   To list all collections in the currently selected database, you can use the `show collections`

Expected output:-

```
Students
```

4. db.foo.insert({"bar" : "baz"}):-

   The command `db.foo.insert({"bar" : "baz"})` is used in MongoDB to insert a document into a

collection named `foo` within the currently selected database (`db`).

5. db.foo.batchInsert([{"_id" : 0}, {"_id" : 1}, {"_id" : 2}])

the `batchInsert` method has been deprecated since version 2.6. The recommended approach to insert multiple documents into a collection is to use the `insertMany` method. Here's how you would achieve this:

6. db.foo.find()

   To retrieve documents from the `foo` collection in MongoDB using the `mongo` shell, you would use the `find()` method. Here's how you would do it:

7. db.foo.remove()

   The `db.foo.remove()` command in MongoDB is used to remove documents from the `foo` collection. However, it's important to note that the `remove()` method without any parameters is deprecated as of MongoDB 4.0 and has been removed in MongoDB 5.0. Instead, you should use `deleteOne()` or `deleteMany()` methods to specify which documents to remove.

DOCUMENTS, COLLECTIONS , DATABASE

DOCUNENTS:-

At the heart of MongoDB is the document:

an ordered set of keys with associated values.

The representation of a document varies by programming language, but most languages have a data structure that is a natural fit, such as a map, hash, or dictionary. ***{"greeting" : "Hello, world!"}***

COLLECTIONS:-

Collections A collection is a group of documents.

If a document is the MongoDB analog of a row in a relational database, then a collection can be thought of as the analog to a table.

DATABASE:-

MongoDB groups collections into databases.

A single instance of MongoDB can host several databases, each grouping together zero or more collections.

A database has its own permissions, and each database is stored in separate files on disk.

A good rule of thumb is to store all data for a single application in the same database.

DATATYPE:-

Basically each document will be in JSON format which will be as follows. Where each attributes inside can be of multiple data types

```
{
    "name" : "John Doe",
    "address" : {
        "street" : "123 Park Street",
        "city" : "Anytown",
        "state" : "NY"
    }
}
```

CLASS 3:-WHERE,AND,OR &CRUD

WHERE:-

In MongoDB, unlike SQL databases, there isn't a direct equivalent of the `WHERE` clause used in querying data. Instead, MongoDB uses the `.find()` method along with query operators to filter documents based on specific criteria.

```
// Find all students with GPA greater than 3.5
db.students.find({ gpa: { $gt: 3.5 } });

// Find all students from "City 3"
db.students.find({ home_city: "City 3" });
```

In tis program we use the $gt (greater than) and there is so much commends are ther like:- $lt, $eq, $ne, $in

AND:-

The explicit AND operation uses the `$and` operator to combine multiple conditions explicitly. This is particularly useful when the conditions are more complex or when combining nested conditions.

```
// Find all students who live in "City 5" AND have a blood group of "A+"
db.students.find({
  $and: [
    { home_city: "City 5" },
    { blood_group: "A+" }
  ]
});
```

In this above example we are find both the blood_group and home_city

OR:-

The logical OR operation in MongoDB can be used to filter documents based on multiple conditions where any of the specified conditions can be true. This is achieved using the `$or` operator. Here's a deeper explanation of the usage of the OR operation:

```
// Find all students who are hotel residents OR have a GPA less than 3.
db.students.find({
  $or: [
    { is_hotel_resident: true },
    { gpa: { $lt: 3.0 } }
  ]
});
```

In this above example we are finding the either hotel_resident or gpa

CRUD:-

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

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

INSERT:-

Inserting documents into a collection can be done using various methods provided by the PyMongo library. The main methods for insertion are `insert_one` for a single document and `insert_many` for multiple documents.

```
// 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);
```

In this above example we are insert the data into document. insertOne()

UPDATE:-

In MongoDB, you can update documents in a collection using the `update_one`, `update_many`, and `replace_one` methods provided by PyMongo. These methods allow you to modify existing documents based on specified criteria.

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

In this above example we are update the name as Alice smith and gpa as 3.8

DELETE:-

In MongoDB, you can delete documents from a collection using the `delete_one`, `delete_many`, and `drop` methods provided by PyMongo. These methods allow you to

remove documents based on specified criteria or to clear an entire collection.

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

In this above example we are delete the document one time “deleteOne()”

UPDATE MANY:-

In MongoDB, you can update multiple documents that match a specified filter using the `update_many` method provided by PyMongo. This method allows you to modify multiple documents in one operation based on specified criteria.

```
// 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 this above example we are update the document many time “updateMany()”

DELETE MANY:-

Deleting multiple documents in MongoDB can be accomplished using the `delete_many` method provided by PyMongo. This method allows you to remove all documents that match a specified filter criteria from a collection.

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

In this above example we are delete the document multiple time