Assignment No.-7
Implement Mongo CRUD Operations.

1. Explain Mongo CRUD operations
We can use MongoDB for various things like building an application or analysis of data or administrator of a MongoDB database. In all these cases, we need to interact with the MongoDB Server to perform certain operations like entering new data into application, updating data into the application, deleting data from the application and reading data of the application.
MongoDB provides a set of essential operations that will help you to easily interact with the MongoDB Server and these operations are known as CRUD operations.
C→ Create, R→ Read, U→ Update, D→ Delete

Create Operations- Create/insert operations are used to insert/add new documents in the Collection.

Method	Description
db.collection.insertOne()	It is used to insert a single document in the Collection.
db.collection.insertMany()	It is used to insert multiple documents in the Collection.

```
> db.student.insertOne({
... name: "Shreya",
... age: 20,
... branch: "CSE",
... amount: 1499
...})
```

```
> db.student.insertMany([
... {
... name: "Anushka",
... age: 21,
... branch: "EEE",
... amount: 1499
... },
...
... {
... name: "Prajakta",
... age: 20,
... branch: "CSE",
... amount: 1499
... }
...])
```

<u>Read operations</u> - Used to retrieve documents from the Collection or query a Collection for a document.

db.collection.find() → It is used to retrieve documents from the Collection.

example -

```
> db.student.find().pretty()
```

Update operations - Update operations are used to update / modify the existing document in the collection. You perform update operations using -

Method	Description
db.collection.updateOne()	It is used to update a single document in the collection that satisfies the given criteria.
db.collection.updateMany()	It is used to update multiple documents in the collection that satisfy the given criteria.
db.collection.replaceOne()	It is used to replace single document in the collection that satisfy the given criteria.

```
> db.student.updateOne(
    {name: ~~"Sumit"~~ "Shreya"},
    {$set: {age: 24}})
```

```
>db.student.updateMany(
  {},
  {$set: {year: 2021}}
)
```

<u>Delete Operations</u>– The delete operation is used to delete / remove the documents from a collection.

Method	Description
db.collection.deleteOne()	It is used to delete a single document from the collection that
db.collection.deleteMany()	It is used to delete multiple documents from the collection that satisfy the given criteria.

```
>db.student.deleteOne({name:"Shreya"})

>db.student.deleteMany({})

{ "acknowledged" : true,
  "deletedCount" : 2 }
```

2. What are the datatypes in MongoDB?
In MongoDB, documents are stored in BSON which is the binary encoded format of JSON.

1. String – This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.

```
> db.student.insertMany([
  { name : "Akshay" },
  { name : "Vikash" }
])
```

the data type of the value of the name field is a String.

2. Integer – Integer data type is used to store a numeric value. Integer can be 32 bit or 64 bit depending upon your server.

```
> db.student.insertOne({ name : "Akash",
                         age : 19 })
```

In the following eg, we are storing the age of the student in the student collection.

3. Double – The double data type is used to store the floating point values.

```
> db.student.insertOne(
  { name : "Sagar,
    age : 20,
    marks : 546.43 })
```

double data type is used to store the marks of the student

4. Boolean - The boolean data type is used to store either true or false.

```
> db.student.insertOne ({
    name:"vishal",
    pass: true })
```

5. Null - The null data type is used to store the null ~~dat~~ value.

```
>db.student.insertOne({
    name: "Akash",
    MobNo: null,
    Skills: ["c", "c++", "java", "python", "JS"] })
```

6. Array - It is a set of values. It can store the same or different data type values in it. The array is created using square brackets. ([]).

7. Object - Object data type stores embedded documents. Embedded documents are also called as nested documents.
They are those types of documents that contain a document inside another document.

```
>db.book.insertOne(
  { Book: { name:"C in depth",
            writer:"Akash" }
  })
```

8. Object ID – Whenever we create a new document in a collection, MongoDB automatically creates a unique object id for that document (if the document does not have it)
There is an _id field in MongoDB for each document. The data stored in the id is of hexadecimal format & the length of the id is 12 bytes that consists of –
→ 4 bytes for timestamp value
→ 5 bytes for random values
→ 3 bytes for counter.
You can create your own id field but make sure that the value of that id field must be unique.

9. ~~Unique~~ Undefined – This data type stores undefined values.

```
>db.project.insertOne(
  { name : "FauG",
    duration : undefined,
    binaryValue : "11001001" })
```

10. Binary data – This datatype is used to store binary data.

```
>db.project.insertOne(
  { name : "FauG",
    duration : undefined,
    binaryValue : "11001001" })
```

13. Symbol - This datatype is similar to the string data type.
>var symbol = "nik432#"

14. Regular Expression - This datatype is used to store regular expressions.

>var expression = new RegExp("/.gfg")

15. JavaScript - This datatype is used to store Javascript code into the document without scope
>var expression = new RegExp("/.gfg")

16. Javascript with Scope - This MongoDB data type stores javascript data with a scope.

17. Timestamp - In MongoDB, this data type is used to store a timestamp. The value of the timestamp data type is always unique.

>var timestmp = new Timestamp()
>time stmp
Timestamp(0,0)

3. Explain find() in mongoDB. with examples.
In MongoDB, find() method is used to select documents in a collection & return a cursor to the selected documents. Cursor means a pointer that points to a document, when we use find() method it returns

a pointer on the selected documents & returns one by one. It takes only some optional parameters.
If we want to return pointer on all documents then use empty document ({}).
eg - find all the documents in the Collection
→ db.student.find()
→ db.student.find({ age : 18})

4. What is the id field in MongoDB?
The field name _id is reserved for use as a primary key. Its value must be unique in the Collection. It is immutable & May be of any type other than an array. If the _id contains subfields, the subfield name cannot begin with a $ symbol.
The _id field is always the first field in the documents.

5. Explain the following terms in MongoDB -
a) document
b) collection
c) embedded document
d) eventual consistency
e) Sharding

1. Document - A record in MongoDB is a document, which is data structure composed of field and value pairs. MongoDB documents are similar to JSON

objects. The values of the fields may include other documents, arrays & arrays of documents.

The advantages of using documents are -

- Documents correspond to native data types in many programming languages
- Embedded documents and arrays reduce the need for expensive joins.
- Dynamic schema supports fluent polymorphism.

2. Collection - Collection is a grouping of MongoDB documents. A collection is equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce schema. Documents within a collection can have different fields. Typically all documents in a collection have a similar or related purpose.

3. Embedded document - Also called Nested document. They are those type of documents that contain a document inside another document. In other words when a collection has a document this document contains another document, another document contains another sub document & so on then such types of documents are called as embedded documents or nested documents. In MongoDB, you can only nest documents upto 100 levels. The overall document size must not exceed 16 MB.

Syntax -

```
{
  ...
  field : { field1 : value1,
            field2 = value2 }
  ...
}
```

eg -

Database - Students

Collection - Courses

```
{
  name : { first : "Arti", middle : "Ravi", last : "Jha" }
  branch : "CSE"
  courseName : "DMS"
}
```

4. <u>Eventual Consistency</u> - Eventual consistency is a consistency model that guarantees that if no new updates are made to a given object, eventually all accesses to that object will return the last updated value.

5. Sharding - It is the process of storing data records across multiple machines & it is MongoDB's approach to meeting the demands of data growth.

As the size of data increases, a single machine may not be sufficient to store the data nor provide an acceptable read & write throughput. Sharding solves the problem with horizontal scaling.
With sharding, you add more machines to support data growth and the demands of read & write operations.

Why Sharding is important?
1. In replication all writes go to master node
2. Latency sensitive querries still go to master.
3. Single replica set has limitation of 12 nodes.
4. Memory can't be large enough when active dataset is big.
5. Local disk is not big enough.
6. Vertical scaling is too expensive.