Practical no 1

Aim: Practical of Data collection, Data curation and management for Large-scale Data

system (such as MongoDB) CRUD operations using MongoDB

- MongoDB Create database
- MongoDB Drop Database
- MongoDB Create collection
- MongoDB Drop collection
- MongoDB Insert Document
- MongoDB Query Document
- MongoDB Update Document
- Delete document in MongoDB
- MongoDB Projection
- limit() and skip() method in MongoDB
- Sorting of Documents in MongoDB
- MongoDB Indexing

Starting server with mongo or mongodb

C:\>**mongo**

>db
Test

- **Create Database in MongoDB**

Once you are in the MongoDB shell, create the database in MongoDB by typing this command:

```
use database_name
```

For example: create a database “tycs”:

> use tycs
switched to db tycs

```
MongoDB Enterprise > use tycs
switched to db tycs
MongoDB Enterprise > show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
tycs    0.000GB
MongoDB Enterprise >
```

create a collection **user** and insert a document in it.
> db.user.insert({name: "Asif", age: 20})

O/P: WriteResult({ "nInserted" : 1 })

>show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

tycs 0.000GB

- **MongoDB Drop Database**

The syntax to drop a Database is:

```
>db.dropDatabase()
```

O/P:

{ "dropped" : "Testdb", "ok" : 1 }

MongoDB Enterprise > show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

tycs 0.000GB

O/P:

```
MongoDB Enterprise > db.dropDatabase()
{ "dropped" : "tycs", "ok" : 1 }
MongoDB Enterprise > show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
MongoDB Enterprise >
```

- **Create Collection in MongoDB**

Method 1: Creating the Collection in MongoDB on the fly

MongoDB Enterprise > use tycs

switched to db tycs

MongoDB Enterprise > db.tycs.insert({name:"Asif khan",age:21,website:"www.google.com"})

O/P:

WriteResult({ "nInserted" : 1 })

Syntax: **db.collection_name.find()**

MongoDB Enterprise > db.tycs.find()

o/p:

{ "_id" : ObjectId("5e410808e3755b1e06a63d1d"), "name" : "Asif khan", "age" : 21, "website" : "www.google.com" }

```
show collections
```

MongoDB Enterprise > show collections

O/P:

tycs

user

- **Drop collection in MongoDB**

SYNTAX:

```
db.collection_name.drop()
```

MongoDB Enterprise > use students

switched to db students

MongoDB Enterprise > **show collections**

students

teachers

tycs

user

MongoDB Enterprise > **db.user.drop()**

true

MongoDB Enterprise > **show collections**

students

teacher

tycs

MongoDB Insert Document

Syntax to insert a document into the collection:

```
db.collection_name.insert()
```

```
> db.tycs.insert(
...    {
...       name: "ASIF",
...       age: 20,
...       email: "asif@gmail.com",
...       course: [ { name: "MongoDB", duration: 7 }, { name: "Java", duration: 30 } ]
...    }
... )
```

O/P:
WriteResult({ "nInserted" : 1 })

Verification:
Syntax:

```
db.collection_name.find()
```

```
> db.tycs.find()
{ "_id" : ObjectId("5c2d37734fa204bd77e7fc1c"), "name" : "ASIF", "age" : 20, "email" : "asif@gmail.com", "course" : [ { "name" : "MongoDB", "duration" : 7 }, { "name" : "Java", "duration" : 30 } ] }
```

MongoDB Example: Insert Multiple Documents in collection

```
MongoDB Enterprise > var beginners=
... [
... "studentID":1001,
... "studentName":"Asif",
... "age":20
... },
... ]
```

- **MongoDB Query Document using find() method**

Querying all the documents in JSON format

```
MongoDB Enterprise > db.students.find().pretty()
```

```
{
        "_id" : ObjectId("5e410f3fe3755b1e06a63d1e"),
        "studentID" : 1001,
        "studentName" : "Asif",
        "age" : 20
}
```

- **Query Document based on the criteria**

```
> db.students.find({StudentName : "Asif"}).pretty()

{
        "_id" : ObjectId("5c281c90c23e08d1515fd9cc"),
        "StudentId" : 1001,
        "StudentName" : "Asif",
        "age" : 20
}
```

- **Updating Document using update() method**

Syntax:

```
db.collection_name.update(criteria, update_data)
```

```
> use tycs
switched to db tycs
> show collections
beginnersbook
students
tycs
> db.createCollection("got")
{ "ok" : 1 }
> var abc = [
... {
...         "_id" : ObjectId("59bd2e73ce524b733f14dd65"),
...         "name" : "Asif",
...         "age" : 20
... },
... ];
```

```
> db.got.find().pretty()
{
        "_id" : ObjectId("59bd2e73ce524b733f14dd65"),
        "name" : "steve",
        "age" : 20
}
```

To update multiple documents with the update() method:

```
db.got.update({"name":"Jon Snow"},
 {$set:{"name":"Kit Harington"}},{multi:true})
```

Updating Document using save() method

Syntax:

```
db.collection_name.save( {_id:ObjectId(), new_document} )
```

To get the _id of a document, you can either type this command:

```
db.got.find().pretty()
```

```
> db.got.find({"name": "Asif"}).pretty()
{
        "_id" : ObjectId("59bd2e73ce524b733f14dd65"),
        "name" : "Asif",
        "age" : 20
}
> db.got.find().pretty()
{
        "_id" : ObjectId("59bd2e73ce524b733f14dd65"),
        "name" : "Steve",
        "age" : 20
}
```

- **MongoDB Delete Document from a Collection**

Syntax of remove() method:

```
db.collection_name.remove(delete_criteria)
```

Delete Document using remove() method

```
> db.students.find().pretty()
{
        "_id" : ObjectId("59bcecc7668dcce02aaa6fed"),
        "StudentId" : 1001,
        "StudentName" : "Steve",
        "age" : 30
}
```

```
db.students.remove({"StudentId": 3333})
```

Output:

```
WriteResult({ "nRemoved" : 1 })
```

To verify whether the document is actually deleted. Type the following command:

```
db.students.find().pretty()
```

It will list all the documents of students collection.

```
> use tycs
switched to db tycs
> db.students.find().pretty()
{
        "_id" : ObjectId("5c281c90c23e08d1515fd9cc"),
        "StudentId" : 1001,
        "StudentName" : "Asif",
        "age" : 20
}
{
        "_id" : ObjectId("5c2d38934fa204bd77e7fc1d"),
        "StudentId" : 1001,
        "StudentName" : "Steve",
        "age" : 30
}
```

Remove all Documents

```
db.collection_name.remove({})
```

- **MongoDB Projection**

Syntax:

```
db.collection_name.find({},{field_key:1 or 0})
```

```
> db.students.find().pretty()
{
        "_id" : ObjectId("5c281c90c23e08d1515fd9cc"),
        "StudentId" : 1001,
        "StudentName" : "Steve",
        "age" : 20
}
> db.students.find({}, {"_id": 0, "StudentId" : 1})
{ "StudentId" : 1001 }
{ "StudentId" : 1002 }


> db.students.find({}, {"_id": 0, "StudentName" : 0, "age" : 0})
{ "StudentId" : 1001 }
{ "StudentId" : 1002 }
```

- **MongoDB – limit( ) and skip( ) method**

The **limit() method in MongoDB**

Syntax:

```
db.collection_name.find().limit(number_of_documents)
db.studentdata.find({student_id : {$gt:2002}}).pretty()
db.studentdata.find({student_id : {$gt:2002}}).limit(1).pretty()
```

MongoDB Skip() Method

```
db.studentdata.find({student_id : {$gt:2002}}).limit(1).skip(1).pretty()
```

- **MongoDB sort() method**

Sorting Documents using sort() method

Syntax of sort() method:

```
db.collecttion_name.find().sort({field_key:1 or -1})
```

1 is for ascending order and -1 is for descending order. The default value is 1.

For example: collection studentdata contains following documents:

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

{
        "_id" : ObjectId("59bf63380be1d7770c3982af"),
        "student_name" : "Steve",
        "student_id" : 1001,
        "student_age"  :1002
}
```

Let's display the student_id of all the documents in **descending order**:

```
> db.studentdata.find({}, {"student_id": 1, _id:0}).sort({"student_id": -1})
{ "student_id" : 1001 }
{ "student_id" : 1002 }
```

To display the student_id field of all the students in **ascending order**:

```
> db.studentdata.find({}, {"student_id": 1, _id:0}).sort({"student_id": 1})
{ "student_id" : 1001 }
{ "student_id" : 1002 }
```

```
> db.studentdata.find({}, {"student_id": 0, _id:0}).sort({"st
udent_id": 1})
{ "student_name" : "Steve", "student_age" : 22 }
{ "student_name" : "Carol", "student_age" : 22 }
{ "student_name" : "Tim", "student_age" : 23 }
>
```

- **MongoDB Indexing with Example**

How to create index in MongoDB

```
db.collection_name.createIndex({field_name: 1 or -1})
```

The value 1 is for ascending order and -1 is for **descending order.**

Let's create the index on student_name field in **ascending order:**

```
db.studentdata.createIndex({student_name: 1})
```

Output:

```
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

- **MongoDB – Finding the indexes in a collection**

```
db.collection_name.getIndexes()
> db.studentdata.getIndexes()
[
```

```
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "test.studentdata"
        },
]
```


Conclusion: Hence we have successfully learned and performed how to create and delete databases in MongoDB.