PRACTICAL 3

```
Microsoft Windows [Version 10.0.22631.4602]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>mongod --version
db version v7.0.6
Build Info: {
    "version": "7.0.6",
    "gitVersion": "66cdc1f28172cb33ff68263050d73d4ade73b9a4",
    "modules": [],
    "allocator": "tcmalloc",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}

C:\Users\Admin>mongosh
Current Mongosh Log ID: 6780ea5936b3d9045a8bf201
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
&appName=mongosh+2.2.0
Using MongoDB:          6.0.13
Using Mongosh:          2.2.0
mongosh 2.2.15 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-01-09T10:15:35.066+05:30: Access control is not enabled for the database. Read and write acces
s to data and configuration is unrestricted
------
```

Create a database userdb and new collection users

```
test> use userdb;
switched to db userdb
userdb> db.createCollection("users")
{ ok: 1 }
```

Create Operations:
1.insertOne()

```
userdb> db.users.insertOne({
...    name: "Angela",
...    age: 27,
... });
{
  acknowledged: true,
  insertedId: ObjectId('6780ef5f36b3d9045a8bf202')
}
```

PRACTICAL 3

2.insertMany()

```
userdb> db.users.insertMany([{
... name:"Dwight",
... age:30,
... },
... {
... name:"Jim",
... age:29,
... }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6780f05936b3d9045a8bf203'),
    '1': ObjectId('6780f05936b3d9045a8bf204')
  }
}
```

Read Operations:

1.Find()

```
userdb> db.users.find()
[
  {
    _id: ObjectId('6780ef5f36b3d9045a8bf202'),
    name: 'Angela',
    age: 27
  },
  {
    _id: ObjectId('6780f05936b3d9045a8bf203'),
    name: 'Dwight',
    age: 30
  },
  { _id: ObjectId('6780f05936b3d9045a8bf204'), name: 'Jim', age: 29 }
]
```

Return all the documents in the "users" collection where the age is greater than 29, and only return the "name" and "age" fields.

```
userdb> db.users.find({ age: { $gt: 29 } }, { name: 1, age: 1 })
[
  {
    _id: ObjectId('6780f05936b3d9045a8bf203'),
    name: 'Dwight',
    age: 30
  }
]
```

PRACTICAL 3

2.findOne()

```
userdb> db.users.findOne({ name: "Jim" })
{ _id: ObjectId('6780f05936b3d9045a8bf204'), name: 'Jim', age: 29 }
```

Update Operations:

1.updateOne()

```
userdb> db.users.updateOne({ name: "Angela" }, { $set: { email: "angela@gmail.com" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2.updateMany()

```
userdb> db.users.updateMany({ age: { $lt: 30 } }, { $set: { status: "active" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

Delete Operations:

1.deleteOne()

```
userdb> db.users.deleteOne({ name: "Angela" })
{ acknowledged: true, deletedCount: 1 }
```

2.deleteMany()

```
userdb> db.users.deleteMany({ age: { $lt: 30 } })
{ acknowledged: true, deletedCount: 1 }
```

3.drop()

```
userdb> db.users.drop()
true
```