Creating a Database Using MongoDB and Mongosh.

NAME : Sravani Medida

Email : medidasravani88@gmail.com

Phone no : 7337073964

Roll No : 20NN1A05F1(CSE)

College Name : Vignans Nirula Institute of Technology and Science for Women

starting mongosh:

To run MongoDB commands in the command prompt (CMD) or terminal, you need to use the mongo or mongosh commands.

```
C:\Users\4727y>mongosh
Current Mongosh Log ID: 65fea25d0e4baeedf3d14a0d
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.1
Using MongoDB:          7.0.6
Using Mongosh:          2.2.1

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

------
   The server generated these startup warnings when booting
   2024-03-20T06:04:12.864+05:30: Access control is not enabled for the database. Read and write access to data and configuration is
 unrestricted
------
```

creating a Database :

use command is used to create database. If database is already exits is switch to current database to specified database.

```
test> use myDatabase
switched to db myDatabase
```

creating a collection :

To create a new collection, use the **db.createCollection()** command followed by the collection name. For example, to create a collection named "users":

```
myDatabase> db.createCollection("users");
{ ok: 1 }
```

Insert Document(s):

To insert a single document into a collection, use the insertOne() method. For example, to insert a document into a collection named "users":

```
myDatabase> db.users.insertOne({name:"tesla",email:"tesla@gmail.com",age:29});
{
  acknowledged: true,
  insertedId: ObjectId('65fea43ae54fe4d0d3d14a0e')
}
```

```
myDatabase> db.users.insertOne({name:"elon",email:"elon@gmail.com",age:52});
{
  acknowledged: true,
  insertedId: ObjectId('65fea488e54fe4d0d3d14a0f')
}
```

```
myDatabase> db.users.insertOne({name:"zuck",email:"zuck@gmail.com",age:39});
{
  acknowledged: true,
  insertedId: ObjectId('65fea4c1e54fe4d0d3d14a10')
}
```

```
myDatabase> db.users.insertOne({name:"john",email:"john@gmail.com",age:46});
{
  acknowledged: true,
  insertedId: ObjectId('65fea549e54fe4d0d3d14a12')
}
```

```
myDatabase> db.users.insertOne({name:"ronaldo",email:"ronaldo@gmail.com",age:39});
{
  acknowledged: true,
  insertedId: ObjectId('65fea513e54fe4d0d3d14a11')
}
```

Find All Documents:

To find all documents in a collection, use the find() method without any filters. For example, to find all documents in a collection named "users":

```
myDatabase> db.users.find();
[
  {
    _id: ObjectId('65fea43ae54fe4d0d3d14a0e'),
    name: 'tesla',
    email: 'tesla@gmail.com',
    age: 29
  },
  {
    _id: ObjectId('65fea488e54fe4d0d3d14a0f'),
    name: 'elon',
    email: 'elon@gmail.com',
    age: 52
  },
  {
    _id: ObjectId('65fea4c1e54fe4d0d3d14a10'),
    name: 'zuck',
    email: 'zuck@gmail.com',
    age: 39
  },
  {
    _id: ObjectId('65fea513e54fe4d0d3d14a11'),
    name: 'ronaldo',
    email: 'ronaldo@gmail.com',
    age: 39
  },
  {
    _id: ObjectId('65fea549e54fe4d0d3d14a12'),
    name: 'john',
    email: 'john@gmail.com',
    age: 46
  }
]
```

Users with an age greater than or equal to 30 :

To find users with an age greater than or equal to 30 using the MongoDB shell (mongosh), you can use the following command:

```
myDatabase> db.users.find({age:{$gte:30}});
[
  {
    _id: ObjectId('65fea488e54fe4d0d3d14a0f'),
    name: 'elon',
    email: 'elon@gmail.com',
    age: 52
  },
  {
    _id: ObjectId('65fea4c1e54fe4d0d3d14a10'),
    name: 'zuck',
    email: 'zuck@gmail.com',
    age: 39
  },
  {
    _id: ObjectId('65fea513e54fe4d0d3d14a11'),
    name: 'ronaldo',
    email: 'ronaldo@gmail.com',
    age: 39
  },
  {
    _id: ObjectId('65fea549e54fe4d0d3d14a12'),
    name: 'john',
    email: 'john@gmail.com',
    age: 46
  }
]
```

db.users: Specifies the collection where you want to perform the search (replace "users" with your actual collection name).

find(): Specifies that you want to find documents in the collection that match a certain condition.

**{ age: { $gte: 30 } }**: Specifies the condition for the search. In this case, it's looking for documents where the "age" field is greater than or equal to 30 ($gte is the greater than or equal to operator).

Update the age of a user with a specific email address :

To update the age of a user with a specific email address in MongoDB using the MongoDB shell (mongosh), you can use the following command:

```
myDatabase> db.users.updateOne({email:'tesla@gmail.com'},{$set:{age:22}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
myDatabase> db.users.findOne({email:"tesla@gmail.com"});
{
  _id: ObjectId('65fea43ae54fe4d0d3d14a0e'),
  name: 'tesla',
  email: 'tesla@gmail.com',
  age: 22
}
```

db.users.updateOne() : Specifies that you want to update one document that matches the specified filter.

{ email: "specific@example.com" } : Specifies the filter to find the user with the specific email address ("specific@example.com"). Replace this email address with the actual email you want to target.

{ $set: { age: newAgeValue } } : Specifies the update operation. $set is the update operator that sets the value of the "age" field to the newAgeValue. Replace newAgeValue with the actual new age value you want to set for the user.

Delete a user document based on a specific email address :

To delete a user document based on a specific email address in MongoDB using the MongoDB shell (mongosh), you can use the following command:

```
myDatabase> db.users.deleteOne({email:"john@gmail.com"});
{ acknowledged: true, deletedCount: 1 }
myDatabase> db.users.find();
[
  {
    _id: ObjectId('65fea43ae54fe4d0d3d14a0e'),
    name: 'tesla',
    email: 'tesla@gmail.com',
    age: 22
  },
  {
    _id: ObjectId('65fea488e54fe4d0d3d14a0f'),
    name: 'elon',
    email: 'elon@gmail.com',
    age: 52
  },
  {
    _id: ObjectId('65fea4c1e54fe4d0d3d14a10'),
    name: 'zuck',
    email: 'zuck@gmail.com',
    age: 39
  },
  {
    _id: ObjectId('65fea513e54fe4d0d3d14a11'),
    name: 'ronaldo',
    email: 'ronaldo@gmail.com',
    age: 39
  }
]
```

db.users.deleteOne(): Specifies that you want to delete one document that matches the specified filter.

{ email: "specific@example.com" }: Specifies the filter to find the user with the specific email address ("specific@example.com"). Replace this email address with the actual email you want to target for deletion.

Create an index on the email field of the users collection :

To create an index on the email field of the users collection in MongoDB using the MongoDB shell (mongosh), you can use the following command:

```
myDatabase> db.users.createIndex({email:1},{unique:true});
email_1
```

db.users.createIndex() : Specifies that you want to create an index on the users collection.

{ email: 1 }: Specifies the field to create the index on (in this case, the "email" field) and the direction of the index (1 for ascending order).

{ unique: true }: Optional parameter that ensures uniqueness of values in the indexed field. If you want to allow duplicate values, you can omit this parameter.

After running this command, MongoDB will create an index on the "email" field of the "users" collection. The unique: true option ensures that each email address in the collection is unique.