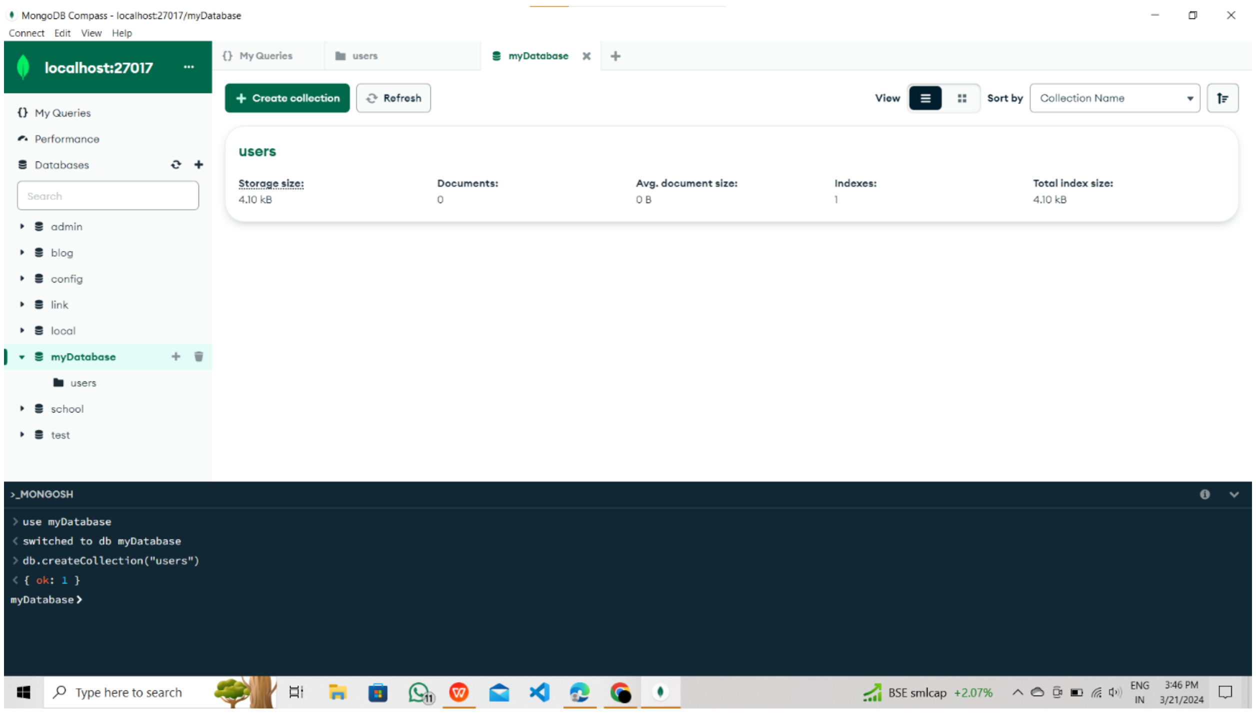
Creating a Database Using MongoDB and Mongosh

Name : Yarrakula Muvvana
Email : yarrakulamuvvana@gmail.com
Roll No:20NN1A0558
College : Vignan’s Nirula Institute of Technology & Science For Women’s

Here's how you can perform the tasks using Mongosh:

1.Database Setup and Collection Creation:

To create database type Use myDatabase & then db.createCollection(“users”) in mongosh shell to create collection.



users

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
4.10 kB	0	0 B	1	4.10 kB

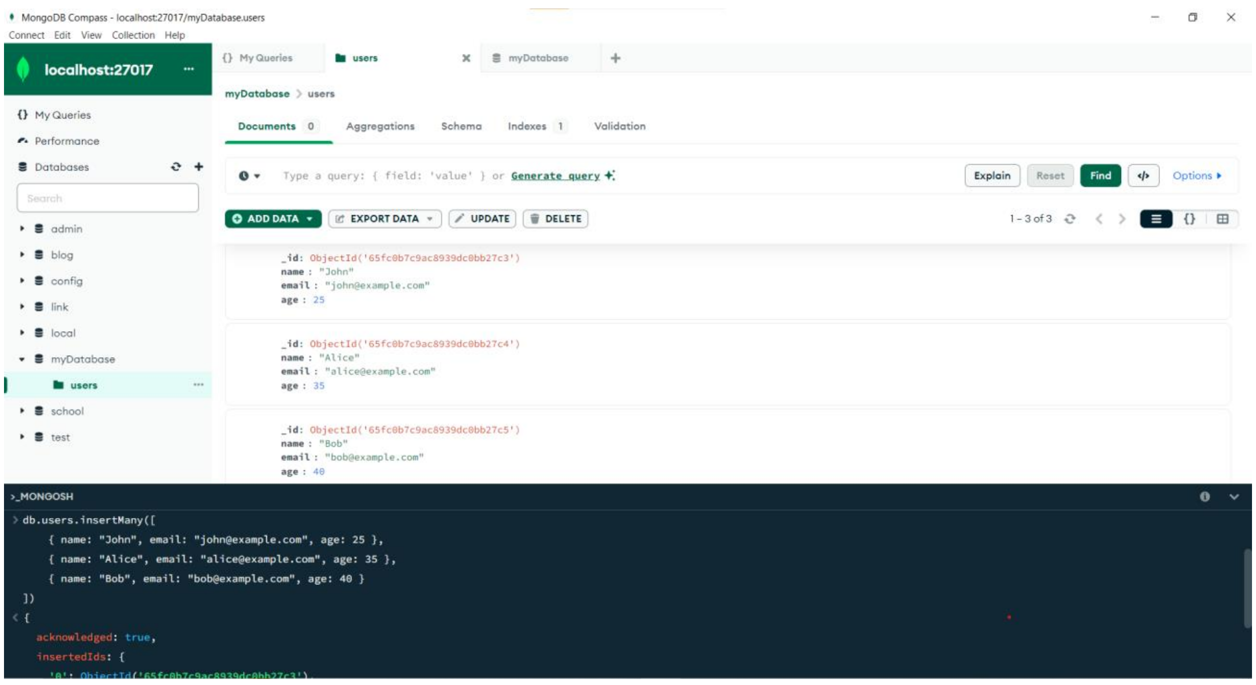
```
> use myDatabase
< switched to db myDatabase
> db.createCollection("users")
< { ok: 1 }
myDatabase>
```

Now we have successfully created a database and collection as well.

2. **Document Insertion:**

To insert documents into the collection, including the structure of the documents and the **insertMany()** command.

```
db.users.insertMany([
  { name: "John", email: "john@example.com", age: 25 },
  { name: "Alice", email: "alice@example.com", age: 35 },
  { name: "Bob", email: "bob@example.com", age: 40 }
])
```



```
_id: ObjectId('65fc0b7c9ac8939dc0bb27c3')
name : "John"
email : "john@example.com"
age : 25

_id: ObjectId('65fc0b7c9ac8939dc0bb27c4')
name : "Alice"
email : "alice@example.com"
age : 35

_id: ObjectId('65fc0b7c9ac8939dc0bb27c5')
name : "Bob"
email : "bob@example.com"
age : 40
```

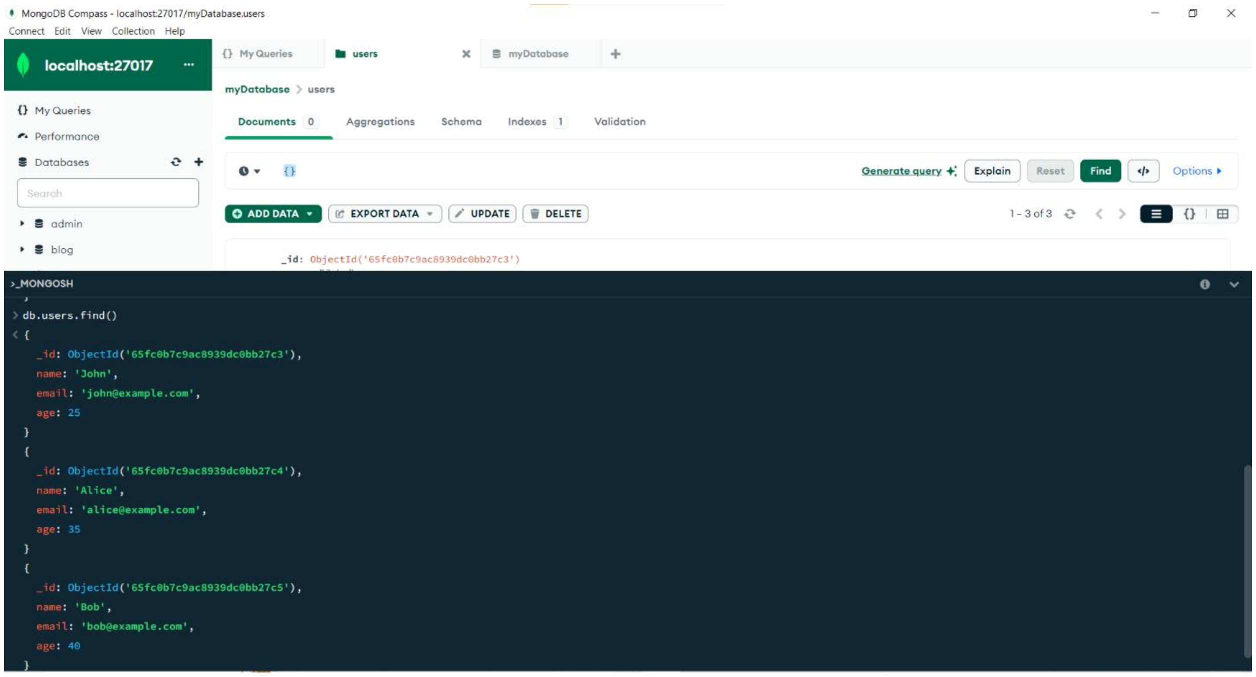
```
> db.users.insertMany([
      { name: "John", email: "john@example.com", age: 25 },
      { name: "Alice", email: "alice@example.com", age: 35 },
      { name: "Bob", email: "bob@example.com", age: 40 }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65fc0b7c9ac8939dc0bb27c3'),
```

We have created documents in the collections,
you can see the documents are visible in the above picture .

3. Querying:

To get all users from the users collection.

Use following command: **db.users.find()**



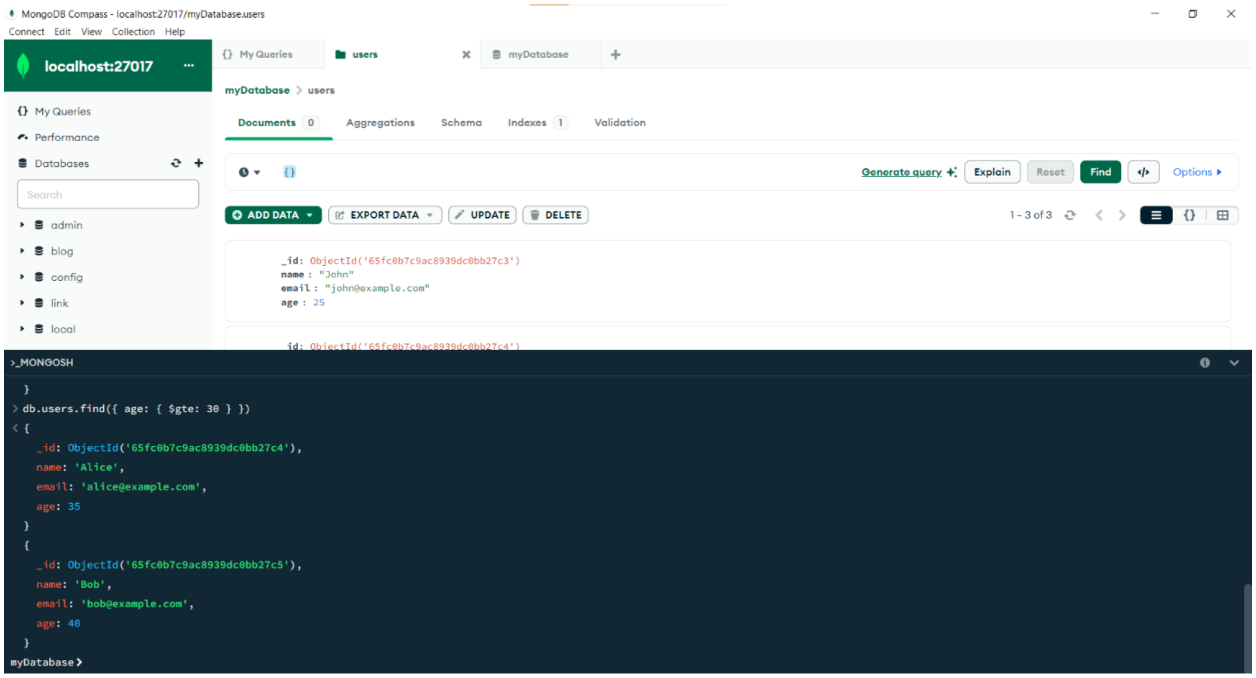
```
_id: ObjectId('65fc0b7c9ac8939dc0bb27c3')
```

```
> db.users.find()
< {
    _id: ObjectId('65fc0b7c9ac8939dc0bb27c3'),
    name: 'John',
    email: 'john@example.com',
    age: 25
  }
  {
    _id: ObjectId('65fc0b7c9ac8939dc0bb27c4'),
    name: 'Alice',
    email: 'alice@example.com',
    age: 35
  }
  {
    _id: ObjectId('65fc0b7c9ac8939dc0bb27c5'),
    name: 'Bob',
    email: 'bob@example.com',
    age: 40
  }
```

Users with age greater than or equal to 30:

Use following command

db.users.find({ age: { $gte: 30 } })



```
_id: ObjectId('65fc0b7c9ac8939dc0bb27c3')
name : "John"
email : "john@example.com"
age : 25

_id: ObjectId('65fc0b7c9ac8939dc0bb27c4')
```

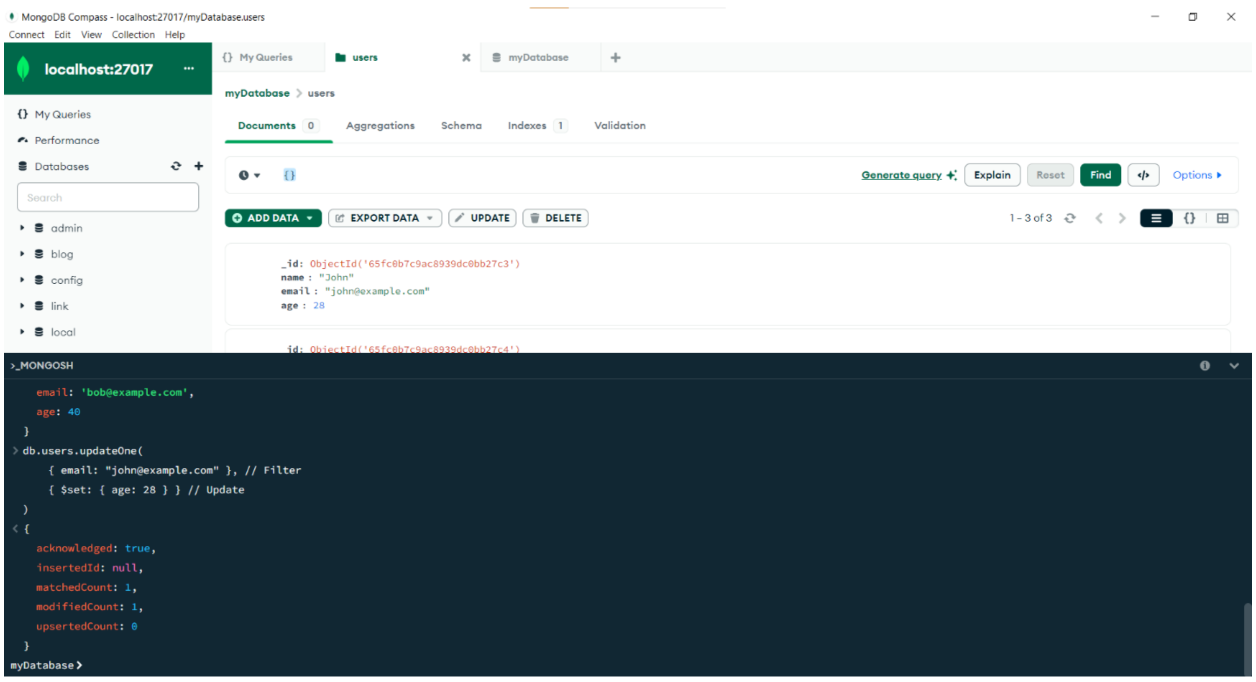
```
  }
> db.users.find({ age: { $gte: 30 } })
< {
    _id: ObjectId('65fc0b7c9ac8939dc0bb27c4'),
    name: 'Alice',
    email: 'alice@example.com',
    age: 35
  }
  {
    _id: ObjectId('65fc0b7c9ac8939dc0bb27c5'),
    name: 'Bob',
    email: 'bob@example.com',
    age: 40
  }
myDatabase>
```

4.Update Operation:

Updating the age of a user with a specific email address

The following command:

```
db.users.updateOne(
   {email: "john@example.com"}, // Filter
   {$set: {age: 28}} // Update
)
```



```
_id: ObjectId('65fc0b7c9ac8939dc0bb27c3')
name : "John"
email : "john@example.com"
age : 28

_id: ObjectId('65fc0b7c9ac8939dc0bb27c4')
```

```
    email: 'bob@example.com',
    age: 40
  }
> db.users.updateOne(
      { email: "john@example.com" }, // Filter
      { $set: { age: 28 } } // Update
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
myDatabase >
```

As u can see the age of a user has been updated. To update one or more users at a time we must use the following command:

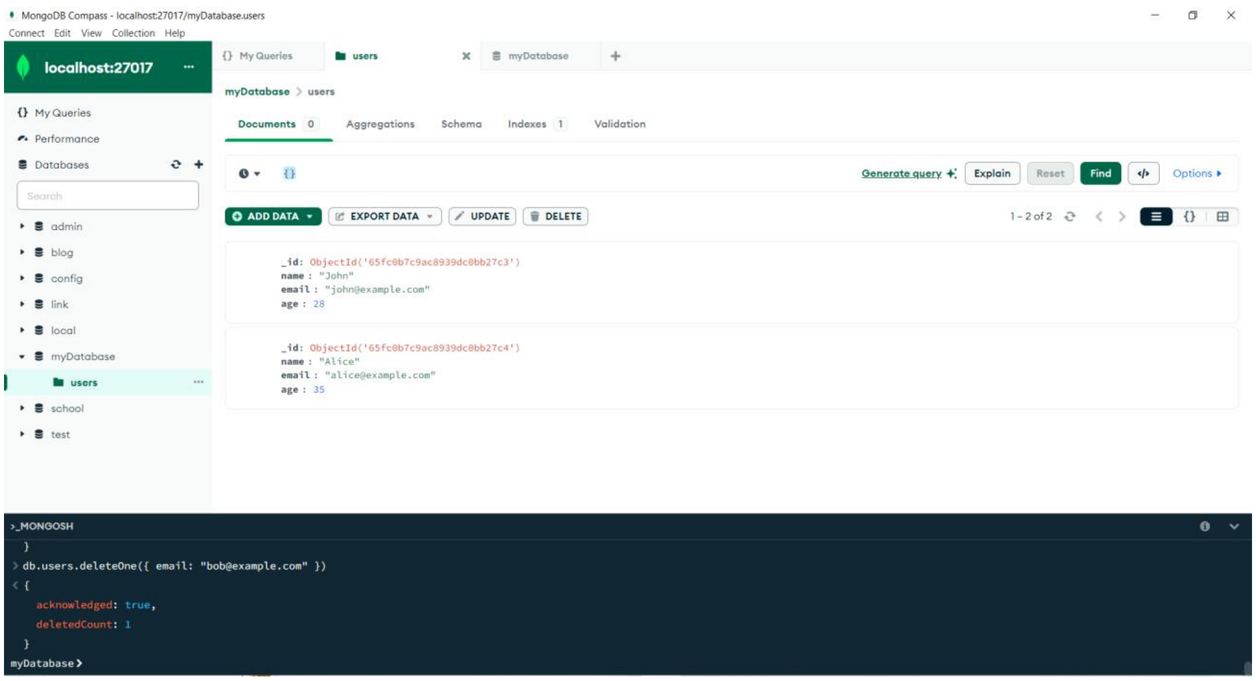
db.users.updateMany()

With the above command you can update more users at a time.
This is how we can update the user's documents.

5. **Deletion Operation:**

Delete a user document based on a specific email address.
The following command:

```
db.users.deleteOne({ email: "bob@example.com" })
```



```
_id: ObjectId('65fc0b7c9ac8939dc0bb27c3')
name : "John"
email : "john@example.com"
age : 28

_id: ObjectId('65fc0b7c9ac8939dc0bb27c4')
name : "Alice"
email : "alice@example.com"
age : 35
```

```
>_MONGOSH
  }
> db.users.deleteOne({ email: "bob@example.com" })
< {
    acknowledged: true,
    deletedCount: 1
  }
myDatabase >
```

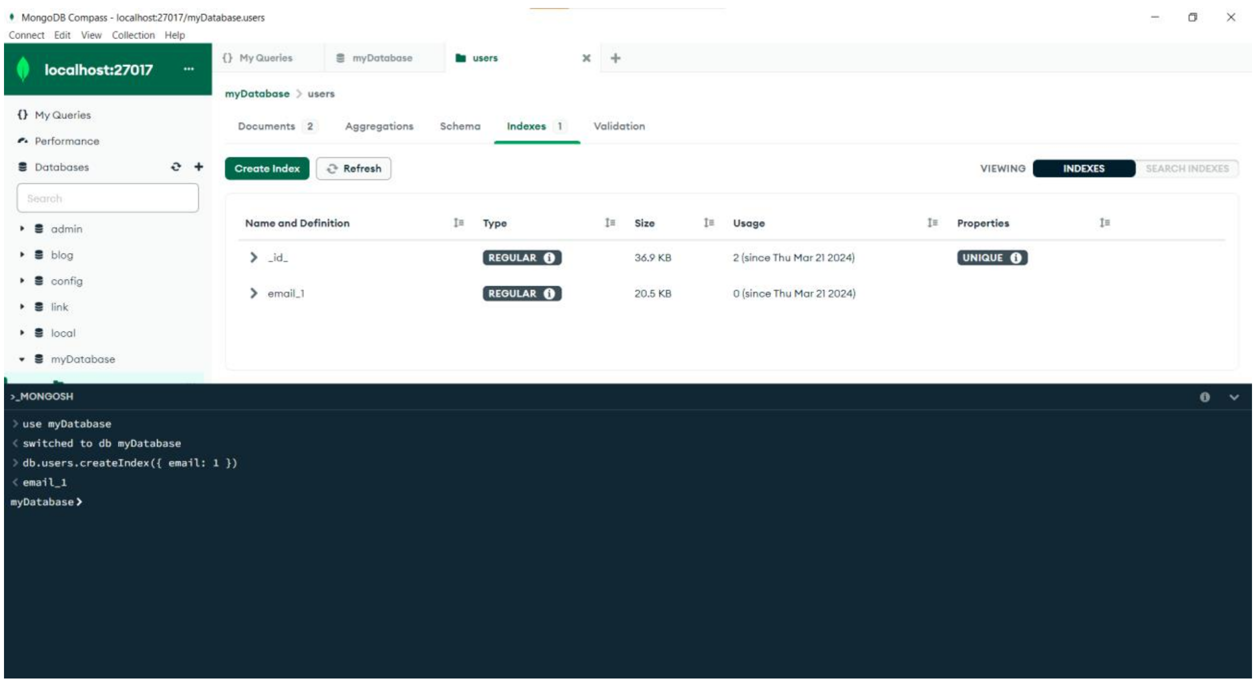
As you can see, we have deleted one user with the help of delete one command.
Also, you can delete more than one document at the same time by using the following command in the Mongosh shell.

```
Db.users.deleteMany()
```

6. Index Creation:

Creating an index on the email field of the users collection.
Use the following command:

```
db.users.createIndex({ email: 1 })
```



Name and Definition	Type	Size	Usage	Properties
_id_	REGULAR	36.9 KB	2 (since Thu Mar 21 2024)	UNIQUE
email_1	REGULAR	20.5 KB	0 (since Thu Mar 21 2024)	

```
> use myDatabase
< switched to db myDatabase
> db.users.createIndex({ email: 1 })
< email_1
myDatabase >
```

Creating indexes is crucial for optimizing query performance in MongoDB, especially in collections with large amounts of data.

CONCLUSION:

Through this assignment, i have gained practical experience in working with MongoDB and Mongosh. i have learned how to create databases, collections, insert documents, query data, and perform basic operations in MongoDB.