

# Creating a Database Using MongoDB and Mongosh

NAME : KOTHA VENKATA SATYA SAI SURYA

EMAIL ID : [208X1a05b0@khitguntur.ac.in](mailto:208X1a05b0@khitguntur.ac.in)

PHONE NO : 7386675365

ROLL NO : 208X1A05B0 (CSE)

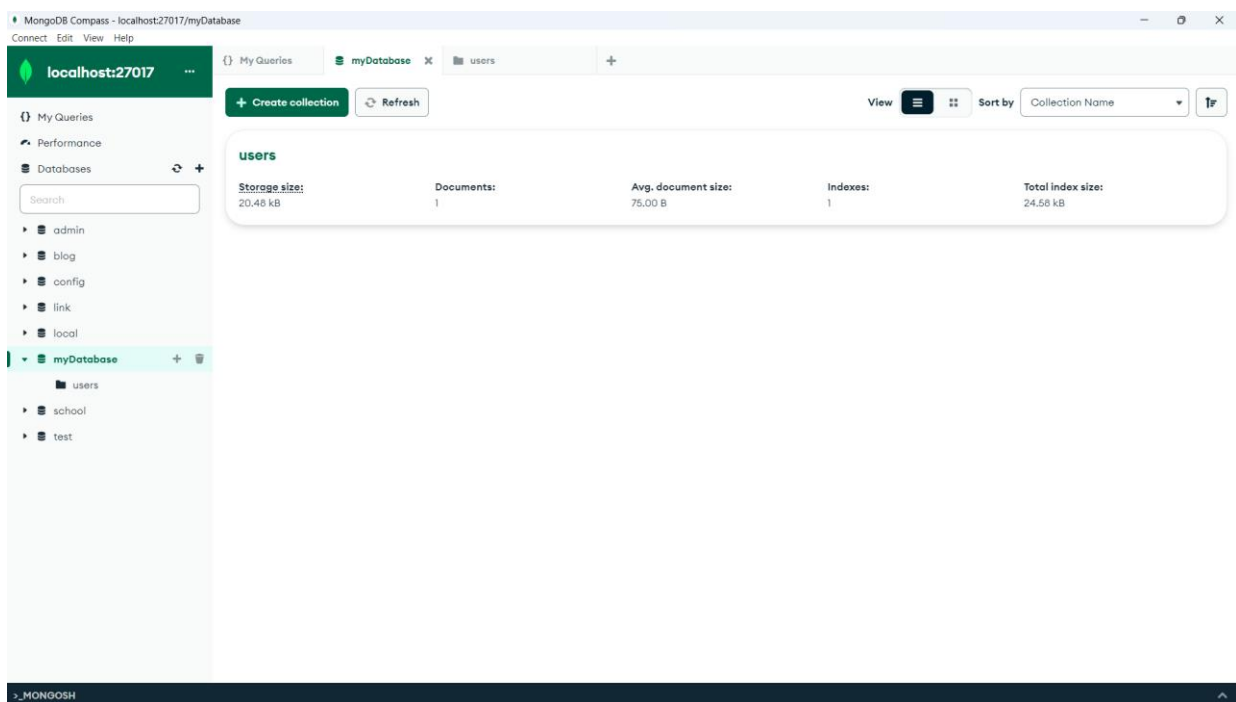
COLLEGE : KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY, GUNTUR

## 1. Database Setup:

- Open the mongoDB compass
- Create a new MongoDB database: myDatabase

## 2. Collection Creation:

- Create a collection within database: users



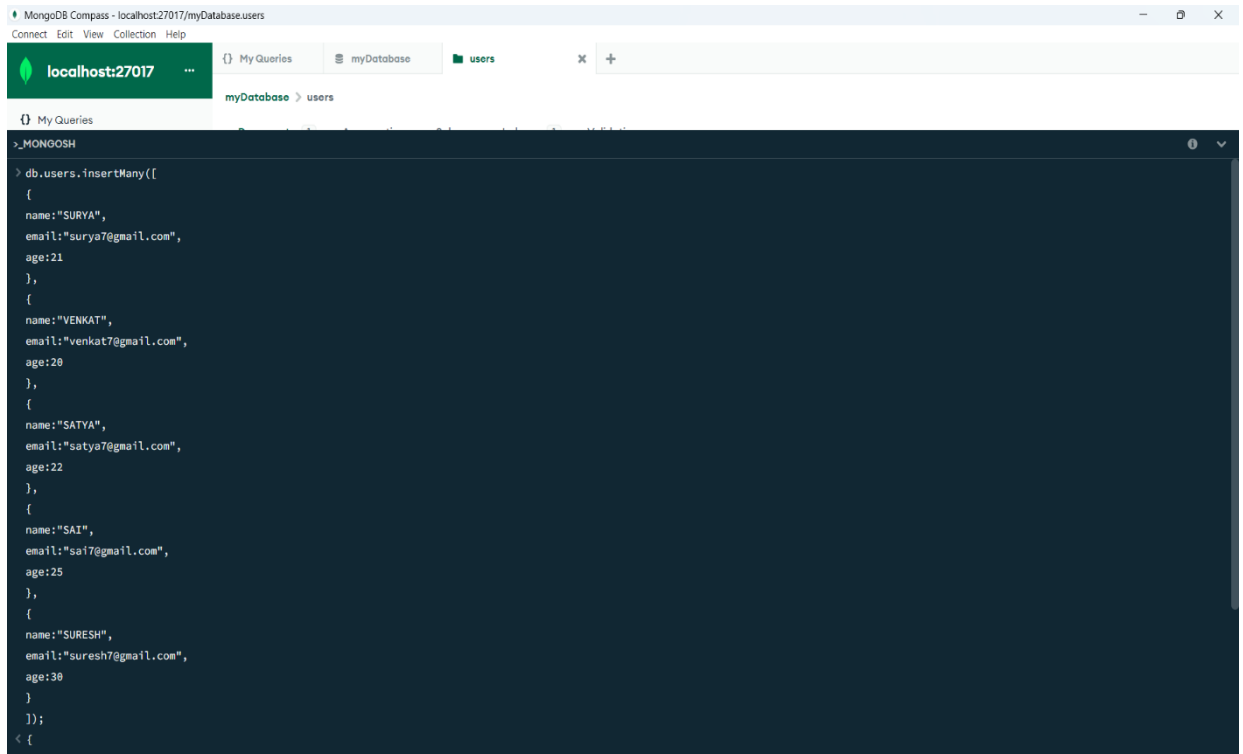
## 3. Document Insertion:

- Insert five documents into the users collection, each representing a user with fields such as name, email, and age.
- Before inserting into collection “use db” command to switch the current database context within MongoDB.

```
>_MONGOSH
> use myDatabase
< switched to db myDatabase
```

- `db.users.insertMany()` method is used to insert the documents into users collection as shown below.

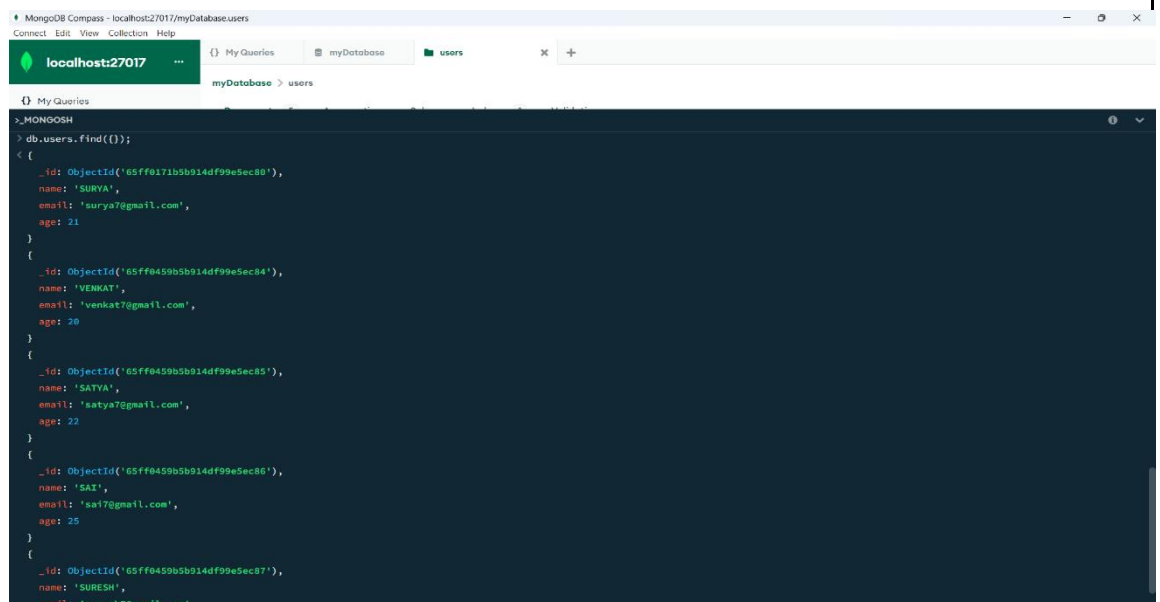
#### 4. Queries to retrieve:



The screenshot shows the MongoDB Compass interface. The top bar indicates the connection to 'localhost:27017' and the current database is 'myDatabase'. The 'users' collection is selected. In the command window, the following JavaScript code is entered:

```
>_MONGOSH
> db.users.insertMany([
  {
    name:"SURYA",
    email:"surya7@gmail.com",
    age:21
  },
  {
    name:"VENKAT",
    email:"venkat7@gmail.com",
    age:20
  },
  {
    name:"SATYA",
    email:"satya7@gmail.com",
    age:22
  },
  {
    name:"SAI",
    email:"sai7@gmail.com",
    age:25
  },
  {
    name:"SURESH",
    email:"suresh7@gmail.com",
    age:30
  }
]);
< {
```

- To retrieve all the users from the users collection. ○  
`db.users.find({ });`



The screenshot shows the MongoDB Compass interface. The top bar indicates the connection to 'localhost:27017' and the current database is 'myDatabase'. The 'users' collection is selected. In the command window, the following JavaScript code is entered:

```
>_MONGOSH
> db.users.find({});
< [
  {
    _id: ObjectId('65ff0171b5b914df99eSec80'),
    name: 'SURYA',
    email: 'surya7@gmail.com',
    age: 21
  },
  {
    _id: ObjectId('65ff0459b5b914df99eSec84'),
    name: 'VENKAT',
    email: 'venkat7@gmail.com',
    age: 20
  },
  {
    _id: ObjectId('65ff0459b5b914df99eSec85'),
    name: 'SATYA',
    email: 'satya7@gmail.com',
    age: 22
  },
  {
    _id: ObjectId('65ff0459b5b914df99eSec86'),
    name: 'SAI',
    email: 'sai7@gmail.com',
    age: 25
  },
  {
    _id: ObjectId('65ff0459b5b914df99eSec87'),
    name: 'SURESH',
    email: 'suresh7@gmail.com',
    age: 30
  }
]
```

- To retrieve the specific users with an age greater than or equal to 30
  - `db.users.find({ age: { $gte: 30 } });`

```
> db.users.find({age:{$gte:30}});
< {
  _id: ObjectId('65ff0459b5b914df99e5ec87'),
  name: 'SURESH',
  email: 'suresh7@gmail.com',
  age: 30
}
myDatabase > |
```

## 5. Update Operation:

- To update the age of a user with a specific email address in MongoDB, use the `updateOne()` method. • Example:
  - `db.users.updateOne(`

```
    { email: "surya7@gmail.com" },
    { $set: { age: 20 } }
```

```
);
```

```
> db.users.updateOne(
  { email:"surya7@gmail.com"},
  {$set:{age:20}}
);
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
myDatabase >
```

## 6. Deletion Operation:

- To delete the user document based on a specific email address in MongoDB, you can use the `deleteOne()` method.

○ `db.users.deleteOne({email:sai7@gmail.com});`

```
> db.users.deleteOne({email:"sai7@gmail.com"});
< {
  acknowledged: true,
  deletedCount: 1
}
myDatabase >
```

## 7. Index Creation:

To create an index on the email field of the users collection in MongoDB, use the `db.users.createIndex()` method.

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

```
> db.users.createIndex({ email: 1});
< email_1
> db.users.getIndexes();
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { email: 1 }, name: 'email_1' }
]
```

# Final Outcome:

