ASSIGNMENT 1

Title: Exploring MongoDB: Installation, Database Creation, and CRUD Operations.

Aim: To provide a comprehensive guide for installing MongoDB, creating databases and collections, and performing CRUD operations on documents.

Objective:
1. Understand the installation process of MongoDB and its basic configuration settings.
2. Learn how to create databases and collections in MongoDB to organize data efficiently.
3. Master the CRUD operations (Create, Read, Update, Delete) for managing documents within MongoDB collections.

Theory:
MongoDB Overview: Introduction to MongoDB as a NoSQL document-oriented database, highlighting its advantages and use cases.
Installation Process: Explanation of the steps required to download, install, and configure MongoDB on various operating systems.
Database and Collection Creation: Description of how to create databases and collections in MongoDB using the MongoDB Shell or graphical user interfaces (GUIs) like MongoDB Compass.
CRUD Operations: Overview of the four fundamental CRUD operations in MongoDB:
- Insert: Adding new documents to a collection.
- Query: Retrieving documents from a collection based on specified criteria.
- Update: Modifying existing documents in a collection.
- Delete: Removing documents from a collection.

Step 1: MongoDB Installation on Windows: Download the MongoDB Community Server from the MongoDB Download Center. Run the installer and follow the setup wizard. Add MongoDB's bin folder to the PATH environment variable for easy command-line access.
https://www.mongodb.com/try/download/community

Step 2: Create a Database and Collection:

- Switch to Your New Database:

  use myNewDatabase

- Create a Collection by Inserting a Document:

db.myNewCollection.insertOne({name: "John Doe", age: 30})

MongoDB creates the database and collection upon inserting the first document.

Step 3: CRUD Operations

- Create (Insert Document): Insert a single document:

db.myNewCollection.insertOne({name: "Jane Doe", age: 25})

- Read (Query Document): Find one document:

db.myNewCollection.findOne({name: "John Doe"})

• Update Document: Update a single document:

db.myNewCollection.update One ({name: "John Doe"}, {$set: {age: 31}})

- Delete Document: Delete a single document:

db.myNewCollection.deleteOne({name: "Bob"})

Conclusion:

The guide provides a step-by-step approach to installing MongoDB, creating databases and collections, and performing CRUD operations on documents. By mastering these fundamental operations, users can harness the power and flexibility of MongoDB for storing and managing data efficiently. This serves as a foundation for further exploration of MongoDB's advanced features and capabilities in application development and data management.

```
ubuntu@ubuntu-OptiPlex-3090:~$ sudo service mongodb start
[sudo] password for ubuntu:
ubuntu@ubuntu-OptiPlex-3090:~$ mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("25635e6a-4336-48e3-be23-a8bb665e00c8") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-09-08T15:44:54.646+0530 I STORAGE  [initandlisten]
2022-09-08T15:44:54.646+0530 I STORAGE  [initandlisten] ** WARNING: Using the XFS
filesystem is strongly recommended with the WiredTiger storage engine
2022-09-08T15:44:54.646+0530 I STORAGE  [initandlisten] **          See
http://dochub.mongodb.org/core/prodnotes-filesystem
2022-09-08T15:44:55.315+0530 I CONTROL  [initandlisten]
2022-09-08T15:44:55.315+0530 I CONTROL  [initandlisten] ** WARNING: Access control is not
enabled for the database.
2022-09-08T15:44:55.315+0530 I CONTROL  [initandlisten] **          Read and write access to data
and configuration is unrestricted.
2022-09-08T15:44:55.315+0530 I CONTROL  [initandlisten]
> use samarth
switched to db samarth
> db.createCollection("emp_info")
{ "ok" : 1 }
> show collections
emp_info
> db.emp_info.insert({id:"E101",ename:"abc",age:25,dept:"tester",sal:25000})
WriteResult({ "nInserted" : 1 })
> db.emp_info.insert({id:"E102",ename:"pqr",age:50,dept:"R&D",sal:50000})
WriteResult({ "nInserted" : 1 })
> db.emp_info.insert({id:"E103",ename:"def",age:56,dept:"R&D",sal:75000})
WriteResult({ "nInserted" : 1 })
> db.emp_info.insert({id:"E104",ename:"xyz",age:28,dept:"dev",sal:50000})
WriteResult({ "nInserted" : 1 })
> db.emp_info.insert({id:"E105",ename:"mno",age:30,dept:"tester",sal:55000})
WriteResult({ "nInserted" : 1 })
> db.emp_info.find()
{ "_id" : ObjectId("6319c913eb1ee6a2f9b3d141"), "id" : "E101", "ename" : "abc", "age" : 25, "dept" :
"tester", "sal" : 25000 }
{ "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"), "id" : "E102", "ename" : "pqr", "age" : 50, "dept" :
"R&D", "sal" : 50000 }
{ "_id" : ObjectId("6319c952eb1ee6a2f9b3d143"), "id" : "E103", "ename" : "def", "age" : 56, "dept" :
"R&D", "sal" : 75000 }
{ "_id" : ObjectId("6319c976eb1ee6a2f9b3d144"), "id" : "E104", "ename" : "xyz", "age" : 28, "dept" :
"dev", "sal" : 50000 }
{ "_id" : ObjectId("6319c993eb1ee6a2f9b3d145"), "id" : "E105", "ename" : "mno", "age" : 30, "dept" :
"tester", "sal" : 55000 }
> db.emp_info.find().pretty()
{
        "_id" : ObjectId("6319c913eb1ee6a2f9b3d141"),
```

```
        "id" : "E101",
        "ename" : "abc",
        "age" : 25,
        "dept" : "tester",
        "sal" : 25000
}
{
        "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"),
        "id" : "E102",
        "ename" : "pqr",
        "age" : 50,
        "dept" : "R&D",
        "sal" : 50000
}
{
        "_id" : ObjectId("6319c952eb1ee6a2f9b3d143"),
        "id" : "E103",
        "ename" : "def",
        "age" : 56,
        "dept" : "R&D",
        "sal" : 75000
}
{
        "_id" : ObjectId("6319c976eb1ee6a2f9b3d144"),
        "id" : "E104",
        "ename" : "xyz",
        "age" : 28,
        "dept" : "dev",
        "sal" : 50000
}
{
        "_id" : ObjectId("6319c993eb1ee6a2f9b3d145"),
        "id" : "E105",
        "ename" : "mno",
        "age" : 30,
        "dept" : "tester",
        "sal" : 55000
}
> db.emp_info.find({sal:{$gte:50000}})
{ "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"), "id" : "E102", "ename" : "pqr", "age" : 50, "dept" :
"R&D", "sal" : 50000 }
{ "_id" : ObjectId("6319c952eb1ee6a2f9b3d143"), "id" : "E103", "ename" : "def", "age" : 56, "dept" :
"R&D", "sal" : 75000 }
{ "_id" : ObjectId("6319c976eb1ee6a2f9b3d144"), "id" : "E104", "ename" : "xyz", "age" : 28, "dept" :
"dev", "sal" : 50000 }
{ "_id" : ObjectId("6319c993eb1ee6a2f9b3d145"), "id" : "E105", "ename" : "mno", "age" : 30, "dept" :
"tester", "sal" : 55000 }
> db.emp_info.find({age:50},{dept:"R&D"})
{ "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"), "dept" : "R&D" }
```

```
> db.emp_info.findOne({age:50},{dept:"R&D"})
{ "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"), "dept" : "R&D" }
> db.emp_info.remove({dept:"tester"})
WriteResult({ "nRemoved" : 2 })
> db.emp_info.update({dept="dev"},{$set:{dept:"R&D"}},{multi:true})
2022-09-08T16:29:10.180+0530 E QUERY    [thread1] SyntaxError: missing : after property id
@(shell):1:24
> db.emp_info.update({dept:"dev"},{$set:{dept:"R&D"}},{multi:true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.emp_info.update({dept:"tester"},{dept:"R&D"})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.emp_info.find({dept:"tester"},{dept:"R&D"})
> db.emp_info.find({dept:"tester","R&D"})
2022-09-08T16:31:55.732+0530 E QUERY    [thread1] SyntaxError: missing : after property id
@(shell):1:37
> db.emp_info.find({$or[{dept:"tester"},{dept:"R&D"}]})
2022-09-08T16:33:14.580+0530 E QUERY    [thread1] SyntaxError: missing : after property id
@(shell):1:21
> db.emp_info.find()
{ "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"), "id" : "E102", "ename" : "pqr", "age" : 50, "dept" :
"R&D", "sal" : 50000 }
{ "_id" : ObjectId("6319c952eb1ee6a2f9b3d143"), "id" : "E103", "ename" : "def", "age" : 56, "dept" :
"R&D", "sal" : 75000 }
{ "_id" : ObjectId("6319c976eb1ee6a2f9b3d144"), "id" : "E104", "ename" : "xyz", "age" : 28, "dept" :
"R&D", "sal" : 50000 }
```