1. What is MongoDB?

- **Introduction**:
  - MongoDB is a **NoSQL, document-oriented database** that stores data in flexible, JSON-like documents, allowing for dynamic schemas. It's ideal for applications needing high scalability, flexibility, and quick iterative development.
- **Benefits of MongoDB**:
  - **Scalability**: Horizontal scaling through sharding.
  - **Schema-less**: Allows each document to be different.
  - **Flexibility and Speed**: Easy to work with JSON data, fast for reads and writes.
- **Use Cases**: Real-time data analytics, content management, IoT applications, mobile applications, and social media platforms.

2. NoSQL vs SQL

- **Key Differences**:
  - **Data Model**: SQL databases use tables with rows and columns, while NoSQL databases (like MongoDB) use collections and documents.
  - **Schema**: SQL databases have a rigid schema, while NoSQL databases are schema-less or schema-flexible.
  - **Scalability**: SQL typically scales vertically, while NoSQL databases are built for horizontal scaling.
- **Advantages and Limitations**:
  - SQL is best for complex querying and structured data.
  - NoSQL is optimized for unstructured data, flexibility, and scalability.

3. Installing MongoDB and MongoDB Atlas

- **Installing MongoDB Locally**:
  - **Download MongoDB** from the official website, available for major OS (Windows, Mac, Linux).
  - Walk through **setting up and running MongoDB** on a local server, configuring paths, and starting the MongoDB service.
- **Setting Up MongoDB Atlas (Cloud)**:
  - MongoDB Atlas is a cloud-hosted version of MongoDB, ideal for scaling without local setup.
  - **Creating an Account**: Sign up for a MongoDB Atlas account and create a new cluster.
  - **Connection String**: Show how to create a connection string to access the cluster.
  - **Network Access**: Guide students to set up IP whitelisting to allow secure access.
- **Connecting a Node.js Application**:
  - Using the MongoDB Atlas connection string, connect it to a sample Node.js application to show remote database connectivity.

4. Introduction to MongoDB Shell

- **What is the MongoDB Shell**:
  - The MongoDB Shell (mongosh) is an interactive JavaScript interface to communicate with MongoDB databases.
- **Basic Shell Commands**:
  - **Viewing Databases**: show dbs
  - **Selecting a Database**: use databaseName
  - **Viewing Collections**: show collections
  - **Inserting Data**: db.collectionName.insert({ key: value })
  - **Basic Queries**: db.collectionName.find({ key: value })
  - **Updating Data**: db.collectionName.updateOne({ key: value }, { $set: { key: newValue } })
  - **Deleting Data**: db.collectionName.deleteOne({ key: value })

5. CRUD Operations in MongoDB

- **CRUD Overview**: CRUD stands for Create, Read, Update, and Delete – fundamental operations for managing data in MongoDB.
- **1. Create**:
  - **Insert Documents**: db.collectionName.insertOne({ key: value }) and db.collectionName.insertMany([{ key1: value1 }, { key2: value2 }]).
  - Show how to create records in a collection with sample data.
- **2. Read**:
  - **Finding Documents**: db.collectionName.find({ key: value }) and db.collectionName.findOne({ key: value }).
  - Explain filtering and projection to narrow down data.
- **3. Update**:
  - **Updating Documents**: db.collectionName.updateOne({ key: value }, { $set: { key: newValue } }).
  - **Update Operators**: $set, $inc, $rename, etc.
  - Discuss scenarios where updating multiple records is needed (updateMany).
- **4. Delete**:
  - **Deleting Documents**: db.collectionName.deleteOne({ key: value }) and db.collectionName.deleteMany({ key: value }).
  - Explain deleting based on conditions and the importance of careful deletion.

Create Multiple Records

- To insert multiple documents into a collection at once, you can use insertMany()

```
db.HemaCodingSchool.insertMany([
  { name: 'Hema', role: 'Organizer' },
  { name: 'Mahesh', role: 'Employee' },
  { name: 'Maruthi', role: 'Student' }
])
```

This command inserts an array of documents into the specified collection. Each object within the array represents a document.

Read Multiple Records

- To retrieve multiple documents that meet certain criteria, use find() with an optional filter.

```
Read:
db.HemaCodingSchool.find({})
```

- This command fetches all documents from collectionName where the city field is "New York". Omitting the filter (e.g., find({})) retrieves all documents.

Projection (optional):

- You can specify fields to include or exclude using projection.

```
db.collectionName.find({ city: "HYD" }, { name: 1, age: 1, _id: 0 });
```

This returns only the name and age fields for matching documents, excluding the _id field.

Update Multiple Records

- To modify multiple documents simultaneously, use updateMany().

```
Update Many:

db.HemaCodingSchool.updateMany(
  { name: { $in: ['Hema', 'Mahesh', 'Maruthi'] } },
  { $set: { role: 'NewRole' } }
)
```

This updates the city field to "San Francisco" for all documents where the city was "Chicago".

Additional Update Operators:

- $inc to increment values: { $inc: { age: 1 } }
- $rename to rename fields: { $rename: { "oldFieldName": "newFieldName" } }

Delete Multiple Records

- To delete multiple documents based on a condition, use deleteMany().

```
Delete Many:

db.HemaCodingSchool.deleteMany(
  { name: { $in: ['Hema', 'Mahesh', 'Maruthi'] } }
)
```

This deletes all documents where the age field is 30 or greater.

Extra Operation:

---------- Insert Many

db.CollegeStudentData.insertMany([

```
  {
    "name": "Hema",
    "age": 24,
    "major": "Computer Science",
    "skills": ["JavaScript", "Node.js"],
    "gpa": 3.9
  },
  {
    "name": "Mahesh",
    "age": 22,
    "major": "Mathematics",
    "skills": ["Python", "Statistics"],
    "gpa": 3.7
  },
  {
    "name": "TATA",
    "age": 23,
    "major": "Physics",
    "skills": ["C++", "Data Analysis"],
    "gpa": 3.8
  },
```

```
  {
    "name": "Sita",
    "age": 21,
    "major": "Biology",
    "skills": ["Research", "Lab Work"],
    "gpa": 3.6
  },
  {
    "name": "Radha",
    "age": 25,
    "major": "Economics",
    "skills": ["R", "Data Visualization"],
    "gpa": 4.0
  }
])
```

---------- Find Many

```
db.CollegeStudentData.find()
db.CollegeStudentData.find().pretty()
 db.CollegeStudentData.find({age:{$gt:23}})
```

----------updateMany

```
 db.CollegeStudentData.updateMany({major:'Computer Science'},{$push:{skills:'Express.js'}})
  db.CollegeStudentData.updateMany({age:{$gt:23}},{$inc:{gpa:0.1}})
```

----------Delete Many

```
db.CollegeStudentData.deleteMany({major:'Mathematics'})
db.CollegeStudentData.deleteMany({gpa:{$gt:3.9}})
db.CollegeStudentData.deleteMany({})
```

----------Drop the collection

db.CollegeStudentData.drop()

Interview Questions:

1.What is MongoDB, and how is it different from SQL databases?

- **Answer**: MongoDB is a NoSQL, document-oriented database that stores data in BSON (Binary JSON) format. Unlike SQL databases, which use a structured schema of tables and rows, MongoDB uses collections and documents, making it schema-flexible. This allows each document to have a unique structure, supporting unstructured or semi-structured data.

2. Explain the CRUD operations in MongoDB.

- **Answer**: CRUD stands for Create, Read, Update, and Delete—fundamental operations for managing data in a database.
  - **Create**: Adds new documents to a collection (e.g., insertOne, insertMany).
  - **Read**: Retrieves documents from a collection (e.g., find, findOne).
  - **Update**: Modifies existing documents (e.g., updateOne, updateMany).
  - **Delete**: Removes documents from a collection (e.g., deleteOne, deleteMany).

3. How do you insert multiple documents in MongoDB?

- **Answer**: MongoDB allows inserting multiple documents using the insertMany() method. This method takes an array of documents as an argument. For example:

```
db.HemaCodingSchool.insertMany([
  { name: 'Hema', role: 'Organizer' },
  { name: 'Mahesh', role: 'Employee' },
  { name: 'Maruthi', role: 'Student' }
])
```

4.What is the difference between find() and findOne() in MongoDB?

- **Answer**:
  - **find()**: Returns all documents that match the query criteria as a cursor, which can then be iterated over.
  - **findOne()**: Returns only the first document that matches the query criteria, without creating a cursor.

```
db.collectionName.find({ age: 25 });     // Returns all matching documents
```

```
db.collectionName.findOne({ age: 25 });  // Returns the first matching document
```

5.How would you update multiple documents in MongoDB?

- **Answer**: To update multiple documents, you can use updateMany(). It updates all documents that match the specified filter. For example:

```
db.HemaCodingSchool.updateMany(
  { name: { $in: ['Hema', 'Mahesh', 'Maruthi'] } },
  { $set: { role: 'NewRole' } }
)
```

- This command changes the status field to "inactive" for all documents with status equal to "active".

6. What is the purpose of the $set operator in MongoDB?

- **Answer**: The $set operator in MongoDB is used to update specific fields in a document. If the field exists, it modifies the field; if it doesn't exist, it adds the field. For example:

```
db.HemaCodingSchool.updateOne(
  { name: 'Mahesh'}, {$set:{ role: 'Emloyee'}}
)
```

This updates the role field of the document where name is "Mahesh" to 'Emloyee'.

7. How do you delete multiple documents from a collection?

- **Answer**: Use deleteMany() to remove all documents that match a specified condition

```
db.HemaCodingSchool.deleteMany(
  { name: { $in: ['Hema', 'Mahesh', 'Maruthi'] } }
)
```

8.What is Mongoose, and why is it used with MongoDB?

- **Answer**: Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It provides schema-based solutions, allowing developers to define the structure of documents, validation, and query-building capabilities. Mongoose is widely used because it brings structure to MongoDB's flexible data model and simplifies interactions with the database.

09. What is MongoDB Atlas?

- **Answer**: MongoDB Atlas is a fully managed cloud database service provided by MongoDB, Inc. It offers automated scaling, backup, and monitoring features and allows developers to deploy, operate, and scale MongoDB databases without managing the underlying infrastructure.

10. Explain sharding in MongoDB.

- **Answer**: Sharding is a method of distributing data across multiple servers, or shards, to enable horizontal scaling. MongoDB supports sharding to manage large datasets and high traffic by partitioning data based on a shard key. Each shard holds a portion of the data, allowing MongoDB to distribute workload and scale easily.

11. What are the differences between updateOne() and updateMany()?

- **Answer**:
    - **updateOne()**: Updates only the first document that matches the filter criteria.
    - **updateMany()**: Updates all documents that match the filter criteria.