

Mongoose MongoDB Operations

Connecting to MongoDB

Establishes a connection to the MongoDB database.

```
const mongoose = require('mongoose');

mongoose.connect('YOUER_CONNECTION_STRING');
const db = mongoose.connection;
db.on('error', (error) => console.error(error));
db.once('open', () => console.log('Connected to Database'));
```

Defining a Model

Defines the structure of the data in the database.

```
const { Schema, model } = mongoose;


const userSchema = new Schema({
  name: String,
  age: Number,
  email: String
});

const User = model('User', userSchema);
```

Creating a Document

Adds new data to the database.

```
const newUser = new User({ name: 'John Doe', age: 30, email: 'john@example.com' }  
newUser.save();
```



Finding Documents

Retrieves data from the database.

```
// Find all documents  
User.find();  
  
// Find one document by criteria  
User.findOne({ name: 'John Doe' });  
  
// Find a document by ID  
User.findById('someObjectId');
```

Updating Documents

Modifies existing data in the database.

```
// Update a document by ID  
User.findByIdAndUpdate('someObjectId', { $set: { age: 31 } }, { new: true });
```

Deleting Documents

Removes data from the database.

```
// Delete a document by ID
```

```
User.findByIdAndDelete('someObjectId');
```

Using populate for Relationship Handling

Automatically replaces specified paths in the document with data from other collections.

```
// Assuming you have two collections, User and Post
// To find all posts and populate their corresponding user details
Post.find().populate('userId').exec();
```

Sorting Documents

Arranges retrieved data in a specific order.

```
// Find all users and sort by age in descending order
User.find().sort({ age: -1 });
```