**Basic CRUD Operations**

**Importing documents**

**A screen shot of a computer code

Description automatically generated**

1. **Insert a Single Document**: Insert a new user document into a users collection.

db.users.insertOne({

"\_id": 11,

"firstName": "Lisa",

"lastName": "Wong",

"email": "lisa.wong@example.com",

"age": 30,

"username": "lisaw",

"lastLogin": "2024-10-01"

})

A screen shot of a computer

Description automatically generated

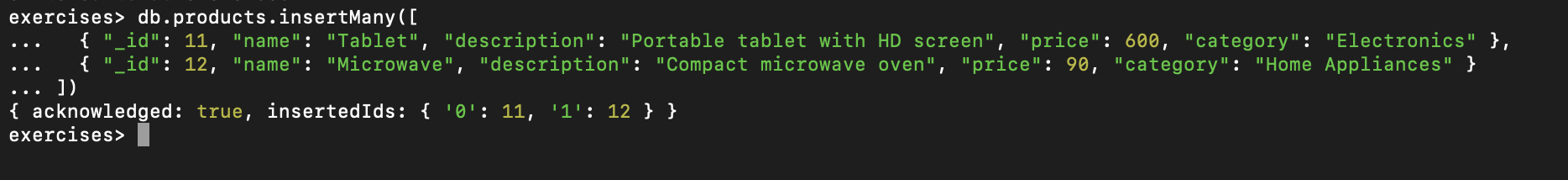
1. **Insert Multiple Documents**: Insert multiple product documents into a products collection.

db.products.insertMany([

{ "\_id": 11, "name": "Tablet", "description": "Portable tablet with HD screen", "price": 600, "category": "Electronics" },

{ "\_id": 12, "name": "Microwave", "description": "Compact microwave oven", "price": 90, "category": "Home Appliances" }

])



1. **Find a Document by ID**: Retrieve a user document by its \_id from the users collection.

db.users.findOne({ "\_id": 1 })

A computer screen with white text and green text

Description automatically generated

1. **Update a Document**: Update the email address of a user in the users collection.

db.users.updateOne(

{ "\_id": 1 },

{ $set: { "email": "john.doe.new@example.com" } }

)

A screen shot of a computer

Description automatically generated

1. **Delete a Document**: Remove a product from the products collection by its \_id.

db.products.deleteOne({ "\_id": 1 })

A black background with white text

Description automatically generated

1. **Find Documents with a Condition**: Retrieve all users who are older than 30 from the users collection.

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

A screen shot of a computer

Description automatically generated

1. **Update Multiple Documents**: Increase the price of all products in the products collection by 10%.

db.products.updateMany(

{},

{ $mul: { "price": 1.1 } }

)

A screen shot of a computer code

Description automatically generated

1. **Delete Multiple Documents**: Remove all users who have not logged in for over a six months from the users collection.

db.users.deleteMany(

{ "lastLogin": { $lt: "2024-05-01" } }

)

A computer screen with white text

Description automatically generated

1. **Upsert Operation**: Insert a new user if it does not exist, or update the existing user’s information.

db.users.updateOne(

{ "username": "newuser" },

{

$set: {

"firstName": "New",

"lastName": "User",

"email": "new.user@example.com",

"age": 25,

"lastLogin": "2024-11-01"

}

},

{ upsert: true }

)

A computer screen with white and green text

Description automatically generated

1. **Regular Expression**: Find places that have the word “of” in their names using a regular expression.

db.places.find(

{ "name": { $regex: /of/, $options: "i" } }

)

A computer screen with white text and green text

Description automatically generated

1. **Compound Index**: Create a compound index on firstName and lastName in the users collection.

db.users.createIndex(

{ "firstName": 1, "lastName": 1 }

)

A computer screen with white text

Description automatically generated

1. **Text Index**: Create a text index on the description field in the products collection for full-text search.

db.products.createIndex(

{ "description": "text" }

)

A screen shot of a computer

Description automatically generated

1. **Unique Index**: Ensure that the username field in the users collection is unique.

db.users.createIndex(

{ "username": 1 },

{ unique: true }

)

A screen shot of a computer

Description automatically generated

1. **Find text**: Using the text index, find the products with “laptop” word in their description.

db.products.find(

{ $text: { $search: "laptop" } }

)

I have deleted laptop, so I’ll use another query

db.products.find(

{ $text: { $search: "oven" } }

)

A screen shot of a computer code

Description automatically generated

1. **Geospatial Index**: Create a 2dsphere index on the location field in the places collection.

db.places.createIndex(

{ "location": "2dsphere" }

)

A screen shot of a computer

Description automatically generated

1. **Indexed search**: Retrieve all products with prices higher than 1000 and with the word “laptop” in the description.

db.products.find(

{

$and: [

{ "price": { $gt: 1000 } },

{ $text: { $search: "laptop" } }

]

}

)

I have deleted laptop, so I’ll use another query

db.products.find(

{

$and: [

{ "price": { $gt: 50 } },

{ $text: { $search: "oven" } }

]

}

)

A screen shot of a computer program

Description automatically generated

1. **Find Documents Near a Point**: Retrieve all places within 5 kilometers of a given point.

db.places.find({

"location": {

$near: {

$geometry: {

type: "Point",

coordinates: [-73.9857, 40.7484]

},

$maxDistance: 5000

}

}

})

A screen shot of a computer code

Description automatically generated

1. **Geospatial Sorting**: Sort places by their distance from a given point.

db.places.find({

"location": {

$near: {

$geometry: {

type: "Point",

coordinates: [-73.9857, 40.7484]

}

}

}

})

A screen shot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

1. **Find documents using $or** : Find places with the word “the” or “of” in their names.

db.places.find({

$or: [

{ "name": { $regex: "the", $options: "i" } },

{ "name": { $regex: "of", $options: "i" } }

]

})

A computer screen shot of a black screen

Description automatically generated

1. **Find documents using $in** : Find products of Electronics and Furniture type.

db.products.find({

"category": { $in: ["Electronics", "Furniture"] }

})

A screenshot of a computer program

Description automatically generated

1. **Find documents by array value**: Find places with both coordinates less than 0.

db.places.find({

"location.coordinates.0": { $lt: 0 },

"location.coordinates.1": { $lt: 0 }

})

A screen shot of a computer

Description automatically generated