BERR2243

Database and Cloud System

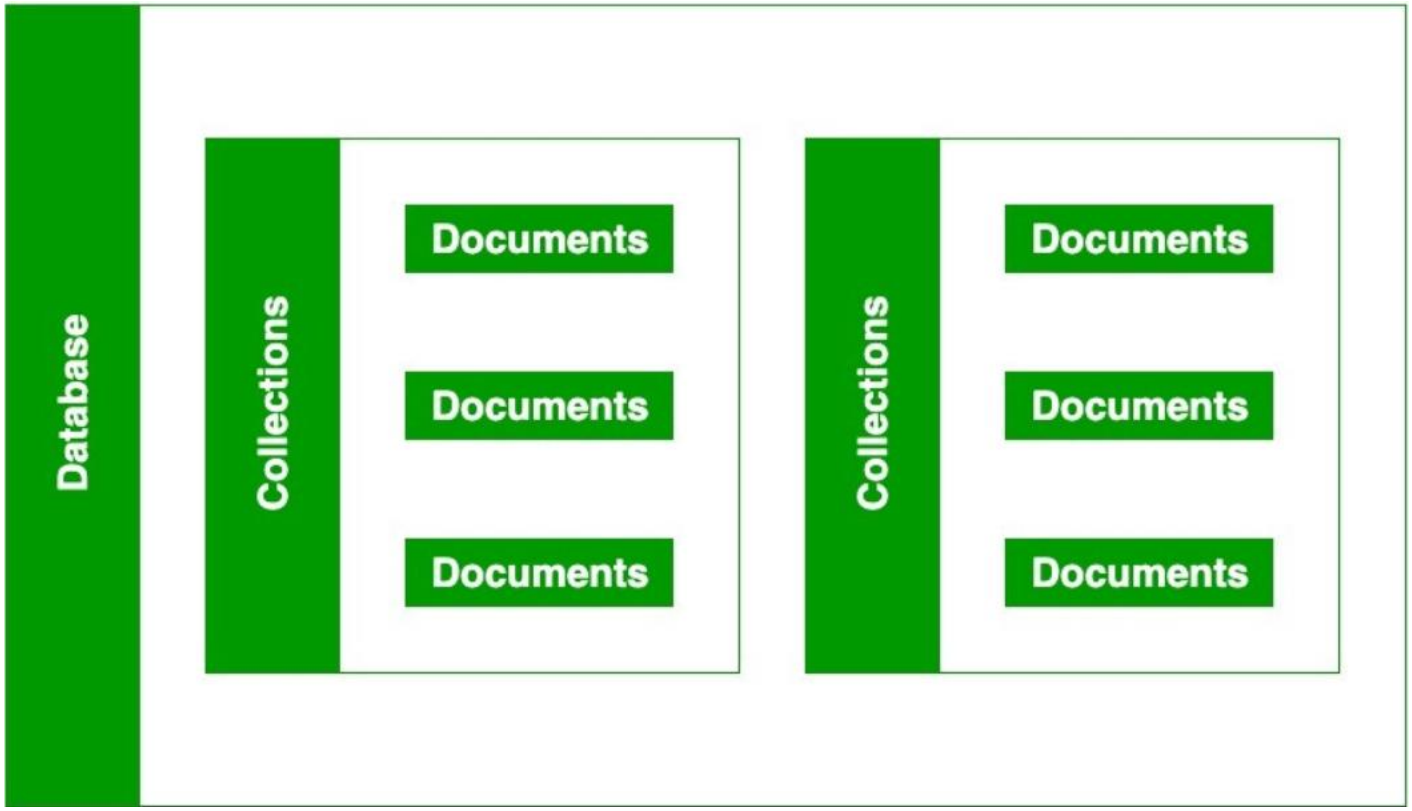
Chapter 3:

CRUD Operations

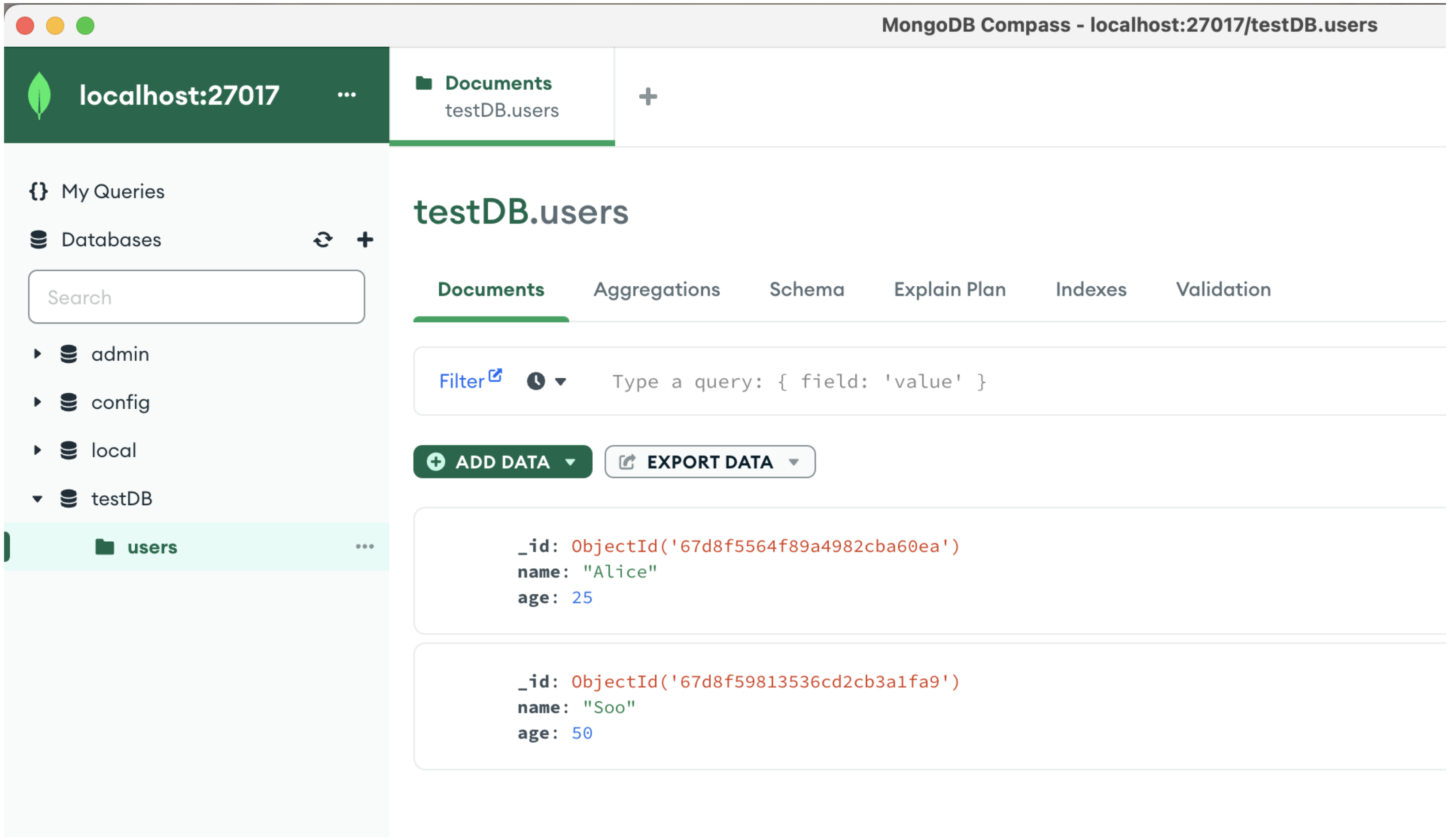
Learning Outcome

- To understand the basic database operations
- To develop basic javascript program to manipulate the database

- A Database contains a collection, and a collection contains documents, and the documents contain data



Document Data Model



```
_id: ObjectId('67d8f5564f89a4982cba60ea')
name: "Alice"
age: 25
```

```
_id: ObjectId('67d8f59813536cd2cb3a1fa9')
name: "Soo"
age: 50
```

MongoDB Data Model

Document Data Model

ID	Name	Email	...
1	Jack	jack@example.com	
2	Jill	jill@example.net	
3	Alex	alex@example.org	

Document 1
{ “id”: 1, “name”:”Jack”, “email”: “jack@example.com”, “address”: {“street”: “900 university ave”, “city”: “Riv state: “CA”}, “friend_ids”: [3, 55, 123]}

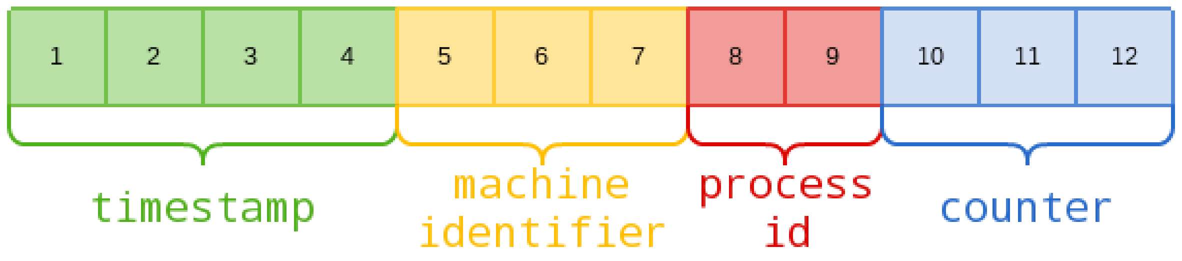
Document 2
{ “id”: 2, “name”: “Jill”, “email”: “jill@example.net”, “hobbies”: [“hiking”, “cooking”]}

Document Format

- MongoDB natively works with JSON documents
- For efficiency, documents are stored in a binary format called BSON (i.e., binary JSON)
- Like JSON, both schema and data are stored in each document
- Individual documents stored in a collection
- Each document in collection has unique ObjectID field called _id

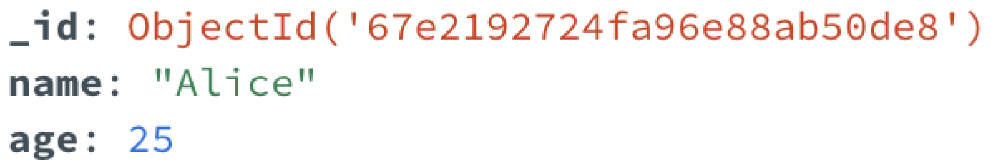
ObjectId

- MongoDB Object IDs are 12-byte hexadecimal or 24 hexadecimal numbers
- These numbers are not entirely random, but generated with the following pattern:



ObjectId

- Example:



```
_id: ObjectId('67e2192724fa96e88ab50de8')
name: "Alice"
age: 25
```

67e21927 (1742870823_{10}) > Tuesday, 25 March 2025 02:47:03

24fa96: Machine Identifier

E88a: Process ID

B50de8: An incremental counter

What are CRUD Operations?

- The basic operations performed on data stored in databases are known as CRUD operations.
- CRUD stands for Create, Read, Update, and Delete.
- Exercise:

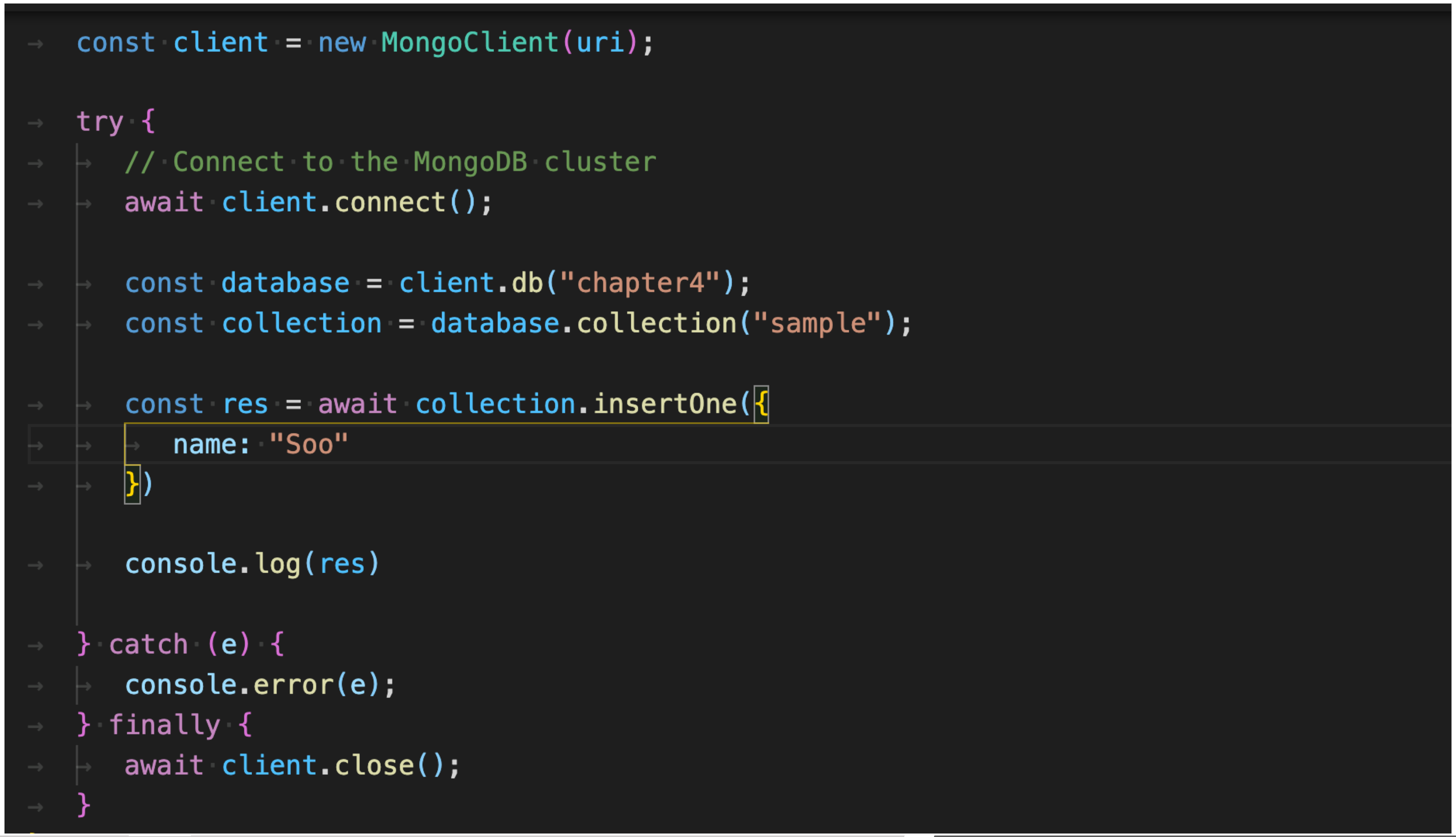
	User Management	
C	User Registering	
R	User Login	
U	User Updating Profile	
D	User Remove Account	

- To insert or add new documents in the collection.
- If a collection does not exist, then it will create a new collection in the database.

`insertOne()`	insert a single document in the collection.
`insertMany()`	insert multiple documents in the collection.

Create

- Insert a new Document into the database **chapter4** and collection **sample**



```
const client = new MongoClient(uri);

try {
    // Connect to the MongoDB cluster
    await client.connect();

    const database = client.db("chapter4");
    const collection = database.collection("sample");

    const res = await collection.insertOne({
        name: "Soo"
    })

    console.log(res)

} catch (e) {
    console.error(e);
} finally {
    await client.close();
}
```

Create

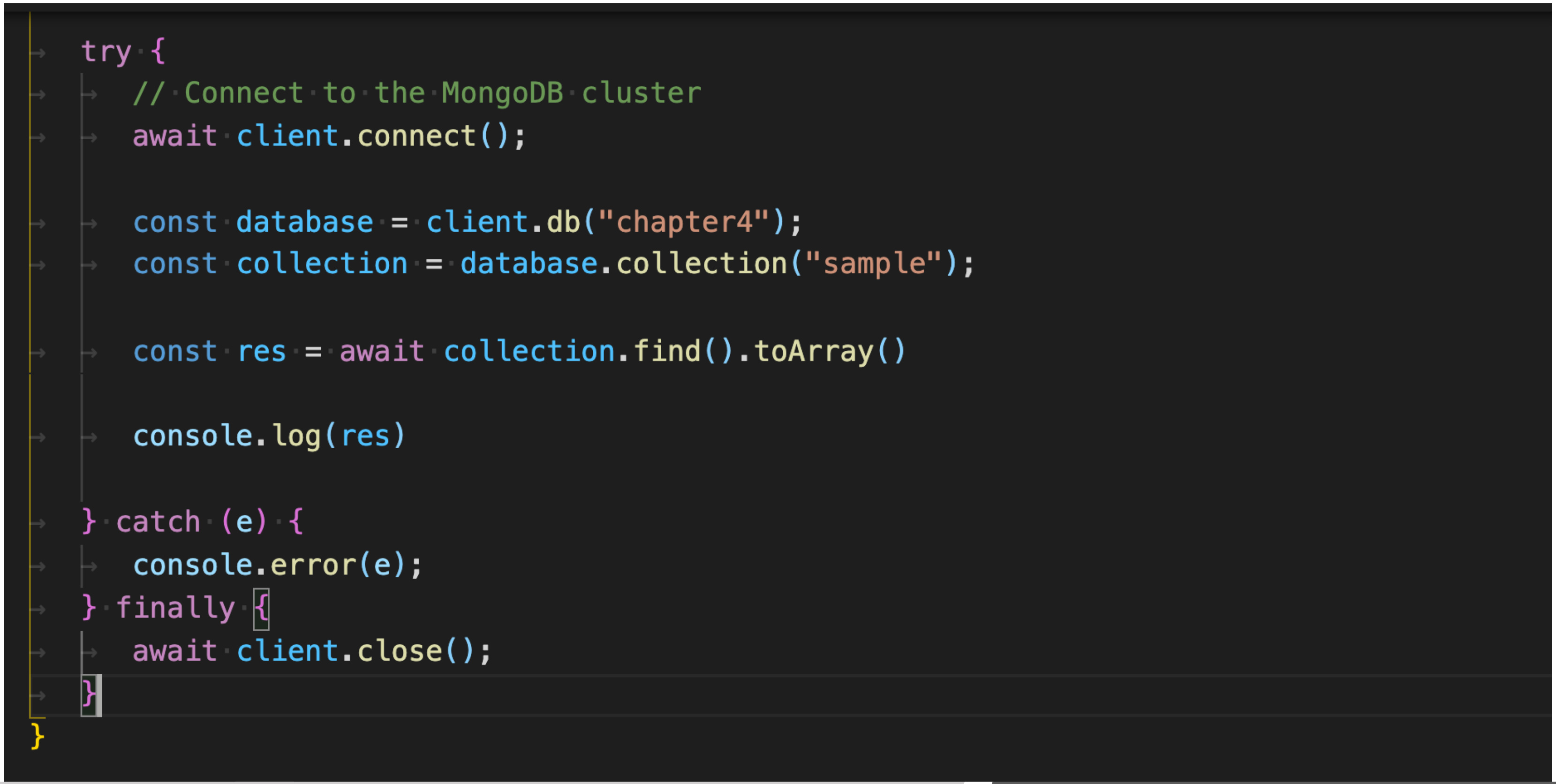
Read

- Read operations, or queries, retrieve data stored in the database
- Queries select documents from a single collection.
- Queries specify criteria, or conditions, that identify the documents that MongoDB returns to the clients
- A query may include a projection that specifies the fields from the matching documents

`find()` to retrieve documents from the collection.

`findOne()` to retrieve documents from the collection.

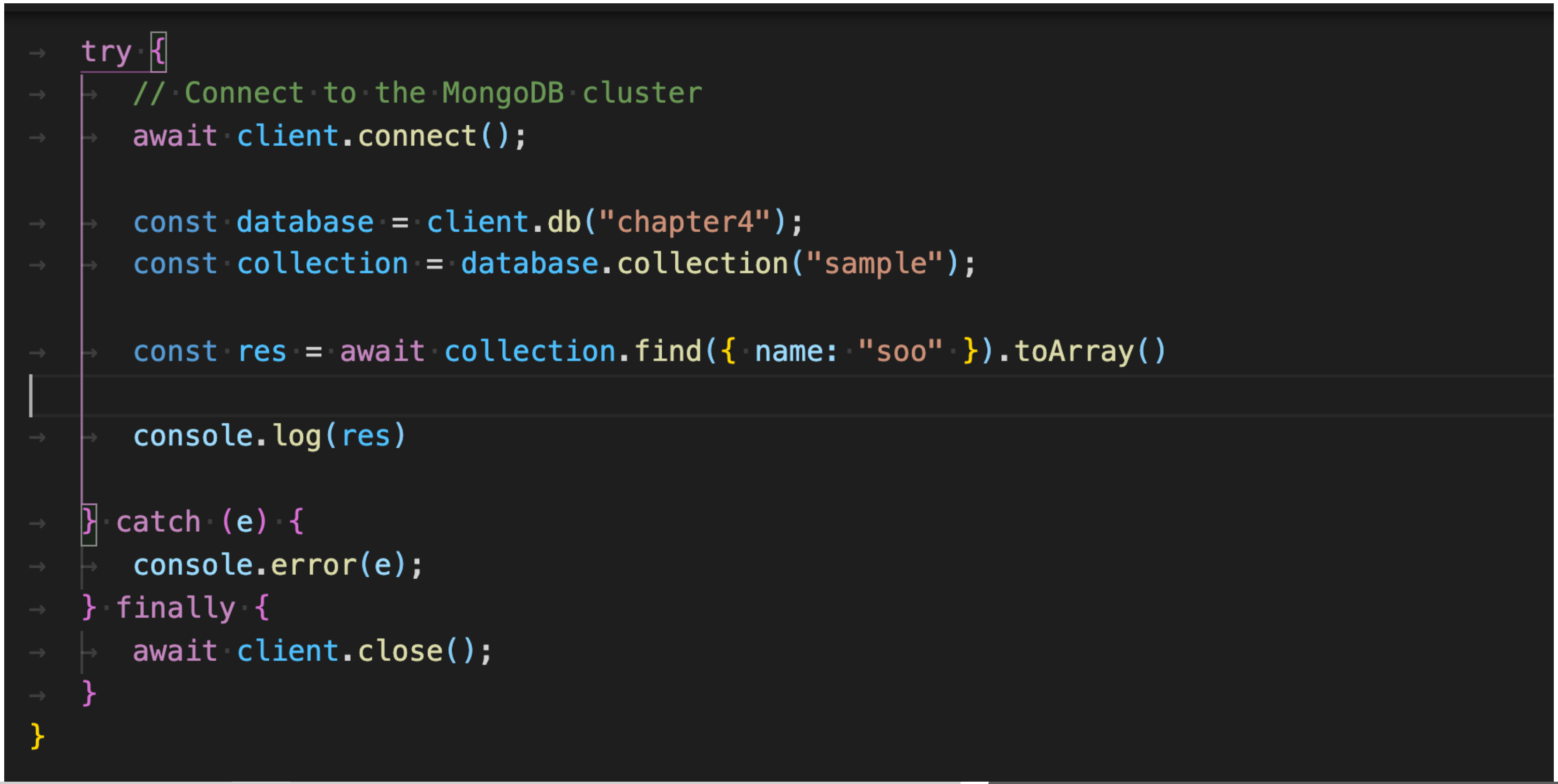
- Find all documents from collection **sample**.



```
    try {
        // Connect to the MongoDB cluster
        await client.connect();

        const database = client.db("chapter4");
        const collection = database.collection("sample");

        const res = await collection.find().toArray()

        console.log(res)

    } catch (e) {
        console.error(e);
    } finally {
        await client.close();
    }
}
```

Read

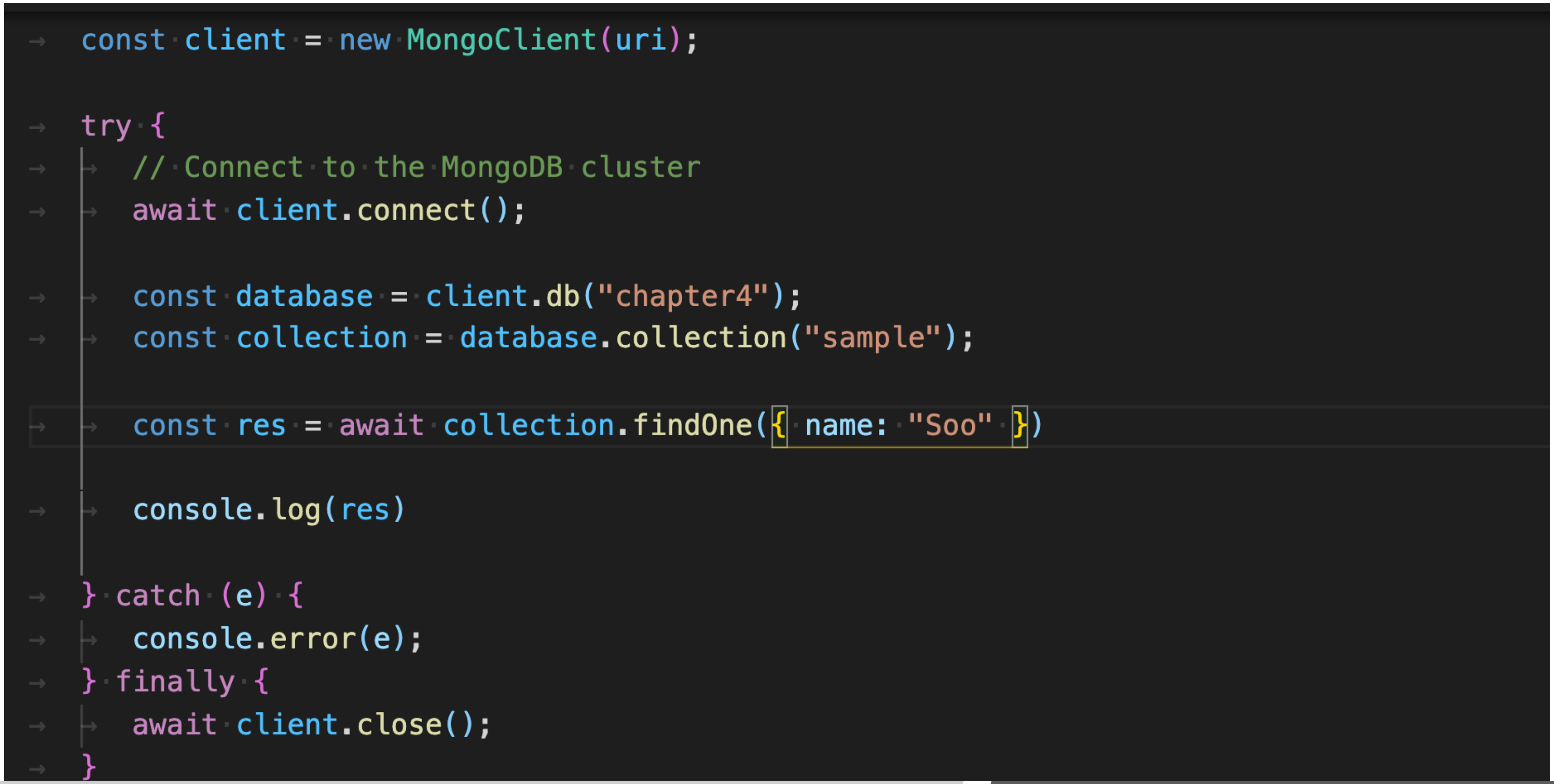
- Find all documents that matched the **conditions** from collection **sample**.



```
    try {
        // Connect to the MongoDB cluster
        await client.connect();

        const database = client.db("chapter4");
        const collection = database.collection("sample");

        const res = await collection.find({ name: "soo" }).toArray()

        console.log(res)

    } catch (e) {
        console.error(e);
    } finally {
        await client.close();
    }
}
```

find()

- Find **ONE** documents that matched the **conditions** from collection **sample**.



```
const client = new MongoClient(uri);

try {
    // Connect to the MongoDB cluster
    await client.connect();

    const database = client.db("chapter4");
    const collection = database.collection("sample");

    const res = await collection.findOne({ name: "Soo" })

    console.log(res)

} catch (e) {
    console.error(e);
} finally {
    await client.close();
}
```

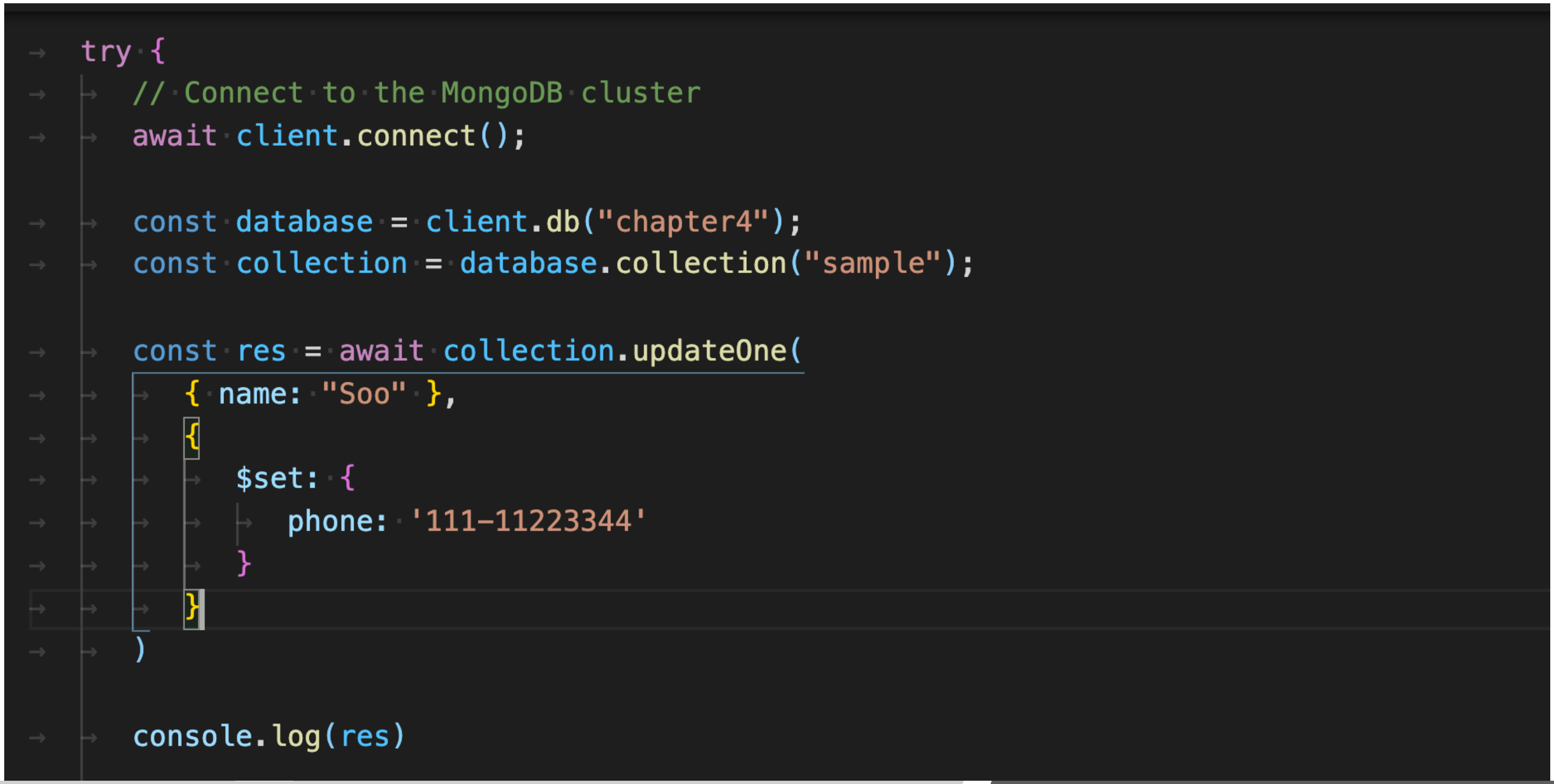
findOne()

Update

- To update or modify the existing document in the collection.
- Specify criteria, or conditions, that identify the documents that MongoDB to be updated

`updateOne()`	update a single document in the collection that satisfy the given conditions.
`updateMany()`	update multiple documents in the collection that satisfy the given conditions.

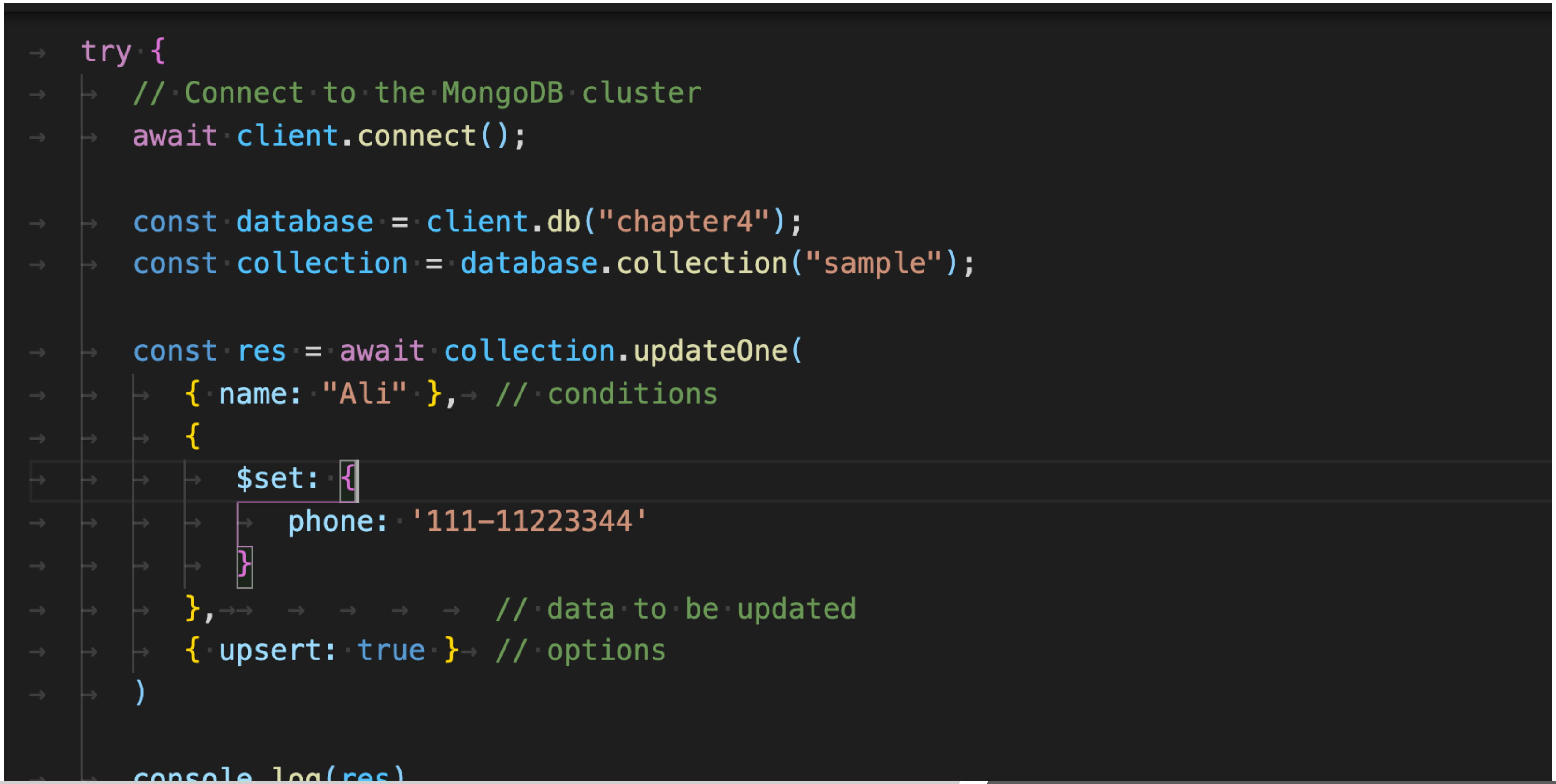
- Update **ONE** documents that matched the **conditions** from collection **sample**.



```
try {
    // Connect to the MongoDB cluster
    await client.connect();

    const database = client.db("chapter4");
    const collection = database.collection("sample");

    const res = await collection.updateOne(
        { name: "Soo" },
        {
            $set: {
                phone: '111-11223344'
            }
        }
    )

    console.log(res)
```

updateOne()

- Update **ONE** documents with **upsert** option



```
try {
    // Connect to the MongoDB cluster
    await client.connect();

    const database = client.db("chapter4");
    const collection = database.collection("sample");

    const res = await collection.updateOne(
        { name: "Ali" },    // conditions
        {
            $set: {
                phone: '111-11223344'
            }
        },                  // data to be updated
        { upsert: true }    // options
    )

    console.log(res)
```

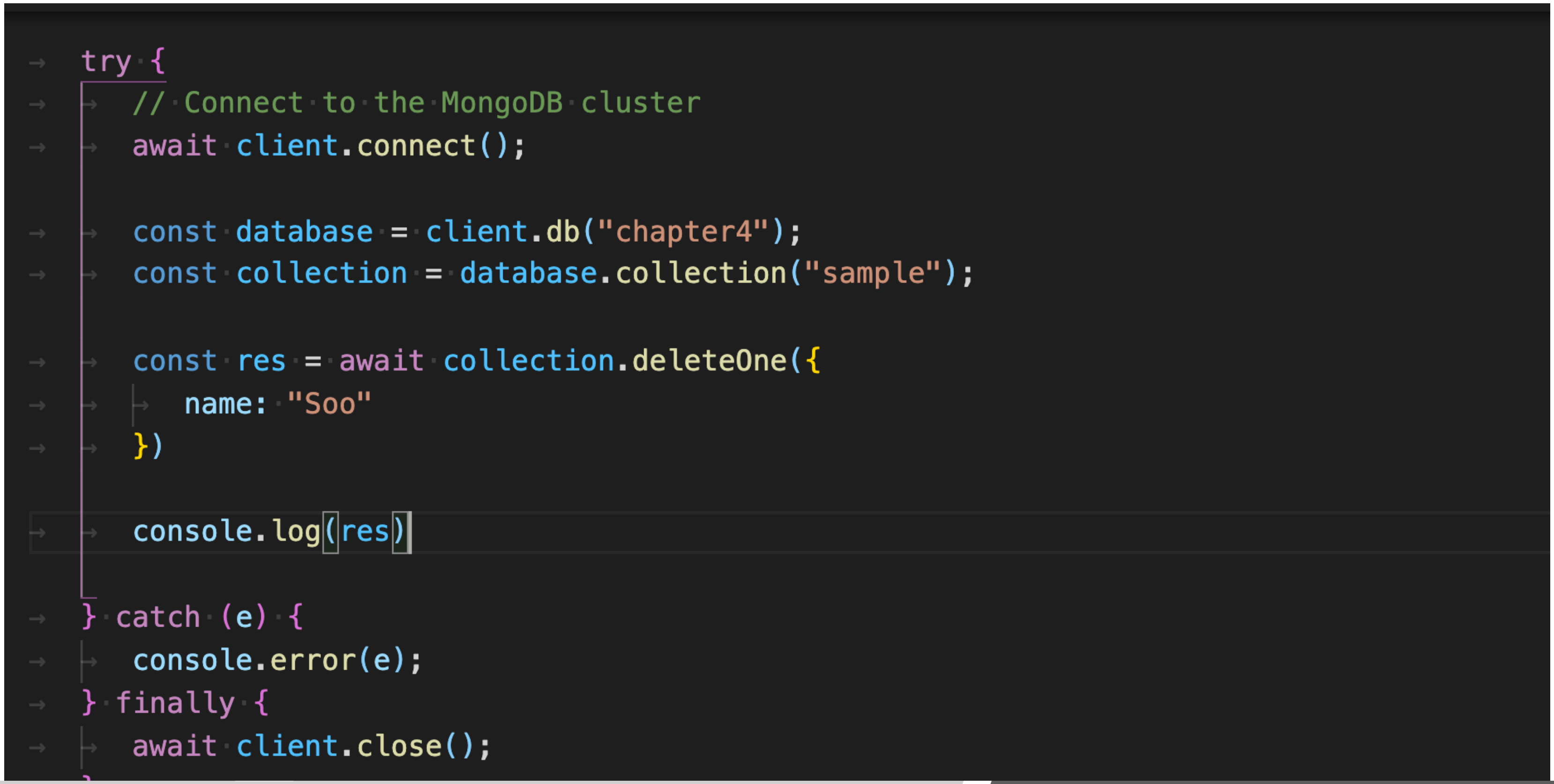
updateOne()

- To delete or remove the documents from a collection.
- Specify criteria, or conditions, that identify the documents that MongoDB to be deleted

`deleteOne()`	delete a single document in the collection that satisfy the given conditions.
`deleteMany()`	delete multiple documents in the collection that satisfy the given conditions.

Delete

- Delete **ONE** documents that matched the **conditions** from collection **sample**.



```
try {
    // Connect to the MongoDB cluster
    await client.connect();

    const database = client.db("chapter4");
    const collection = database.collection("sample");

    const res = await collection.deleteOne({
        name: "Soo"
    })

    console.log(res)

} catch (e) {
    console.error(e);
} finally {
    await client.close();
```

deleteOne()