MongoDB

## Introduction

|  |
| --- |
|  |

JSON and BSON <https://www.educba.com/json-vs-bson/>

|  |
| --- |
|  |

## Collections

Collections are nothing but tables in sql database, each collection is used to a particular form of data. For example, User collection of storing user details

|  |
| --- |
|  |

## Documents

Documents are nothing but a record in sql database, each one represents a single item of data.

|  |
| --- |
|  |

### Query

#### Show all database

Also remember, if any database does not contain any collection it will show that database.

|  |
| --- |
| show dbs // to show all the database |

#### Creating a database

It creates the new database if the database is already exists, then it just switch to that database.

|  |
| --- |
| use <database name> |

#### Checking the current active database

|  |
| --- |
| db |

#### Creating the new collection with insert document

First use database to redirect to that database and execute the below query.

|  |
| --- |
| db.<collection name>.insertOne({name:"Chiranjit Saha", Gender:"M"}) |

#### Checking all the collection in the database

|  |
| --- |
| show collections |

#### Checking all the documents inside the collection

|  |
| --- |
| db.<collection name>.find()  db.<collection name>.find().pretty() // to show it in pretty way |

### CRUD of documents

#### Create

|  |
| --- |
| db.collection.insertOne({})  db.collection.insertMany([{}, {}, {}…]) |

#### Read

|  |
| --- |
| db.chiranjitData.find({gender:"M"}, {\_id:0,name:1}).limit(1).pretty()  db.chiranjitData.findOne({gender:"M"}, {\_id:0,name:1}) |

#### Update

|  |
| --- |
| db.chiranjitData.updateOne({name:"Arijit"}, {$set: {age:17}})  db.chiranjitData.updateMany({gender: "M"}, {$set: {gender:"F"}}) |

#### Delete

|  |
| --- |
| db.chiranjitData.deleteMany({country:"USA"})  db.chiranjitData.deleteMany({}) // delete all documents |

### Comparison query operators

<https://docs.mongodb.com/manual/reference/operator/query-comparison/index.html>

|  |
| --- |
|  |

|  |
| --- |
| Playlist  .find({ videos: { $gte: 60 } })  .select({name: 1}); |

|  |
| --- |
| Playlist.find({        ctype: { $in: ["Back End", "Databases"] },      }).select({name: 1,}); |

### Logical Operators

<https://docs.mongodb.com/manual/reference/operator/query-logical/index.html>

|  |
| --- |
|  |

|  |
| --- |
| Playlist.find({        $or: [{ ctype: "Back End" }, { author: "Chiranjit" }],      }).select({ name: 1 }); |

### Count

|  |
| --- |
| const result = await Playlist.find({        $and: [{ ctype: "Back End" }, { author: "Chiranjit" }],      })        .select({ name: 1 })        .countDocuments();      console.log(result); |

### Sort

Below example will do as ascending order.

|  |
| --- |
| const result = await Playlist.find({        $and: [{ author: "Chiranjit" }],      })        .select({ name: 1 })        .sort("name : 1");      console.log(result); |

### Aggregation

<https://www.mikedane.com/databases/mongodb/aggregation/>

## Mongoose

<https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/>

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.

|  |
| --- |
|  |

|  |
| --- |
|  |

### Schema and Model

|  |
| --- |
|  |

|  |
| --- |
|  |