MongoDB Tutorial

For Beginners

MongoDB- a NoSQL Database

what is mongodb?

MongoDB is a popular, open-source NoSQL document database used for modern applications, storing data in flexible, JSON-like BSON documents instead of traditional tables. It offers high scalability and availability through a horizontal scaling architecture, a flexible schema for varied data types, and developer-focused features like a rich query API and built-in drivers for popular programming languages.

Key Characteristics

- **Document-Oriented:** Data is stored in self-contained documents, similar to JSON objects, within collections.
- **BSON Format:** MongoDB uses BSON (Binary JSON), an extension of JSON, to store data, allowing for more data types beyond standard JSON.
- **Flexible Schema:** Documents in a collection can have different structures, providing flexibility for evolving applications and varied data types.
- **NoSQL Database:** It's a non-relational database, contrasting with SQL databases that use structured tables.
- **Scalability & Availability:** Designed for horizontal scaling ("scaling out") across multiple systems to handle large datasets and maintain high availability.

MongoDB- a NoSQL Database

Key Features & Benefits

•**Developer-Friendly:**Simplifies development with a flexible data model and is designed for common CRUD (Create, Read, Update, Delete) operations.

•**Rich Query Language:**Provides a powerful and expressive query API to work with data effectively.

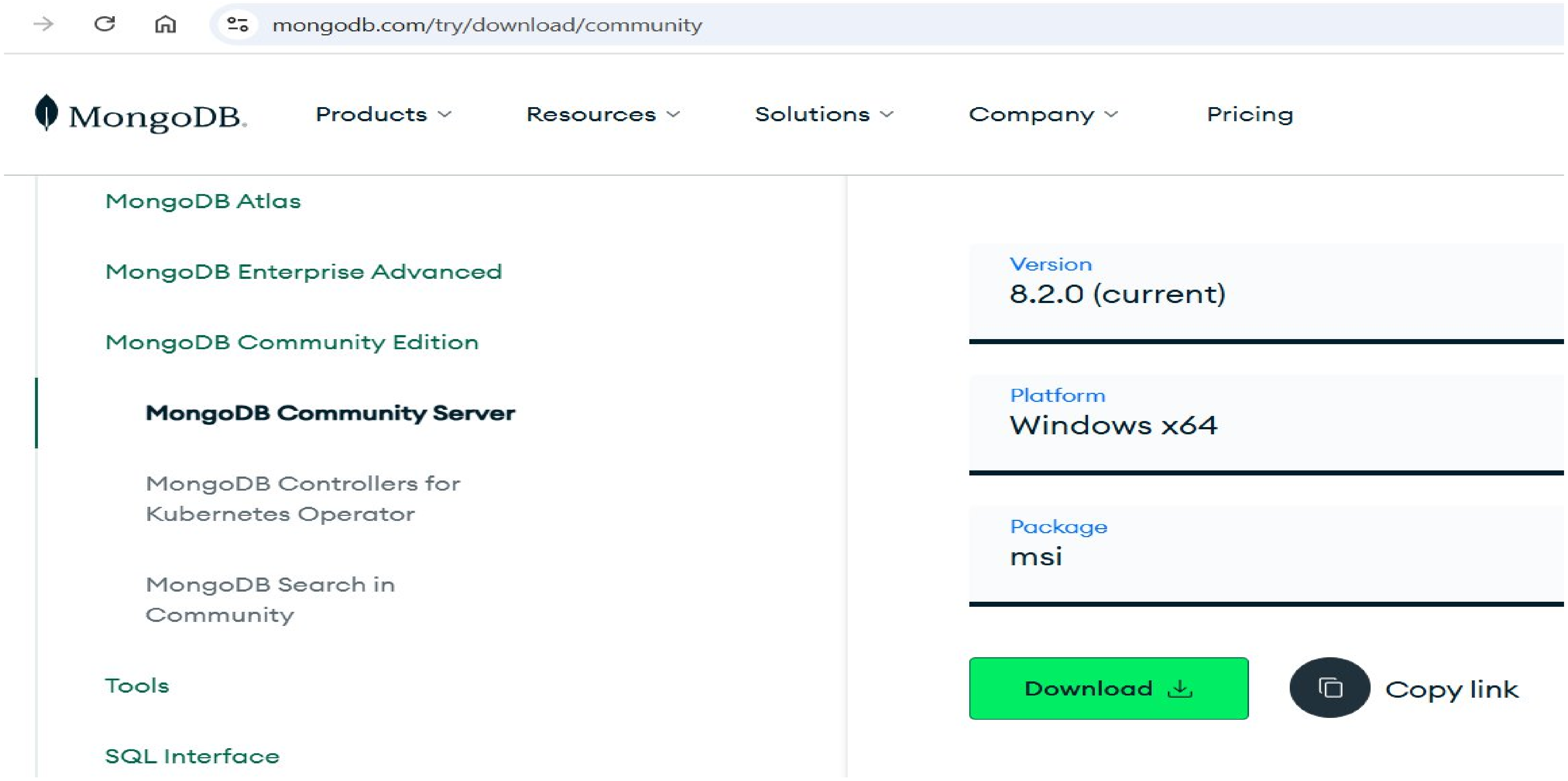
•**Indexing:**Supports indexing for faster data retrieval, including specialized indexing for time-series data.

•**Distributed Architecture:**Built for high availability and geographic distribution of data.

•**Integrated Services:**MongoDB Atlas offers a comprehensive platform with database, search, and data visualization services.

MongoDB- a NoSQL Database

Installation- MongoDB and Compass



mongodb.com/try/download/community
MongoDB
Products
Resources
Solutions
Company
Pricing
MongoDB Atlas
MongoDB Enterprise Advanced
MongoDB Community Edition
MongoDB Community Server
MongoDB Controllers for Kubernetes Operator
MongoDB Search in Community
Tools
SQL Interface
Download
Copy link

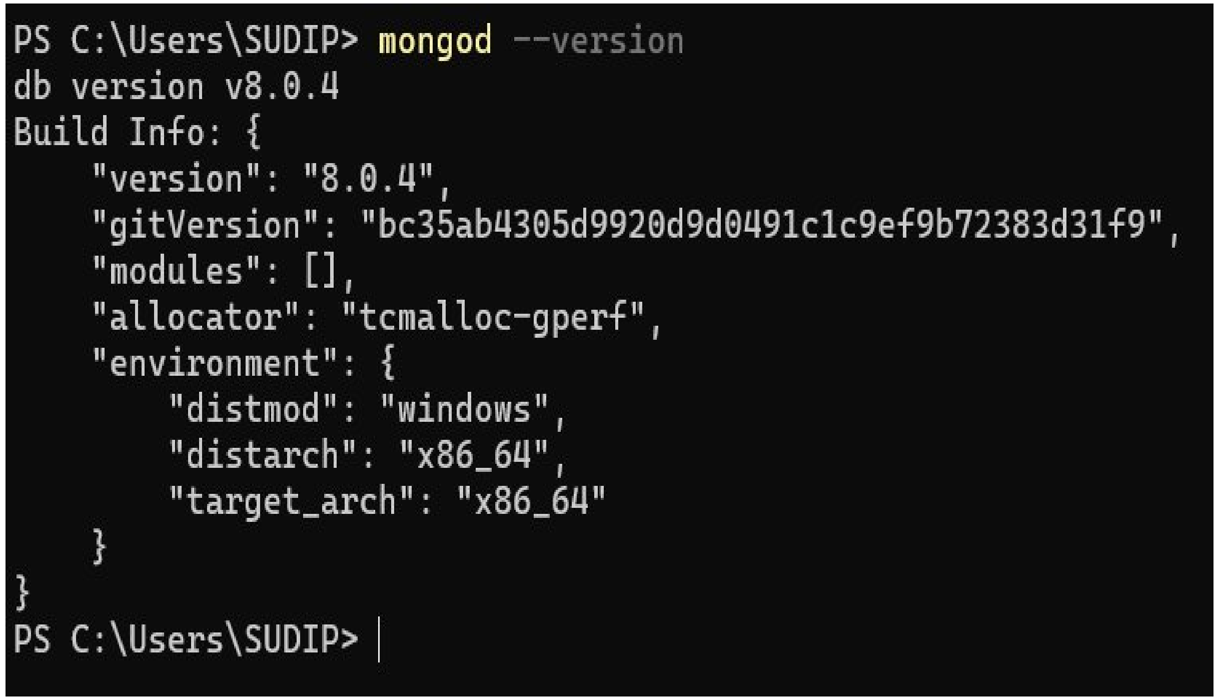
Version
8.2.0 (current)

Platform
Windows x64

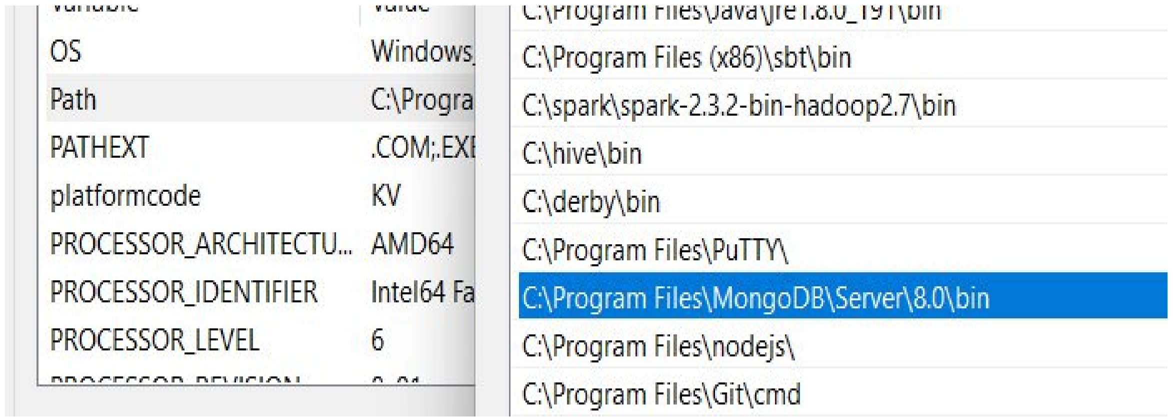
Package
msi

MongoDB- a NoSQL Database

After installation ->>setup required
>> C:\Program Files\MongoDB\Server\8.0\bin
>> system env variable
>>open power shell
>>mongod --version



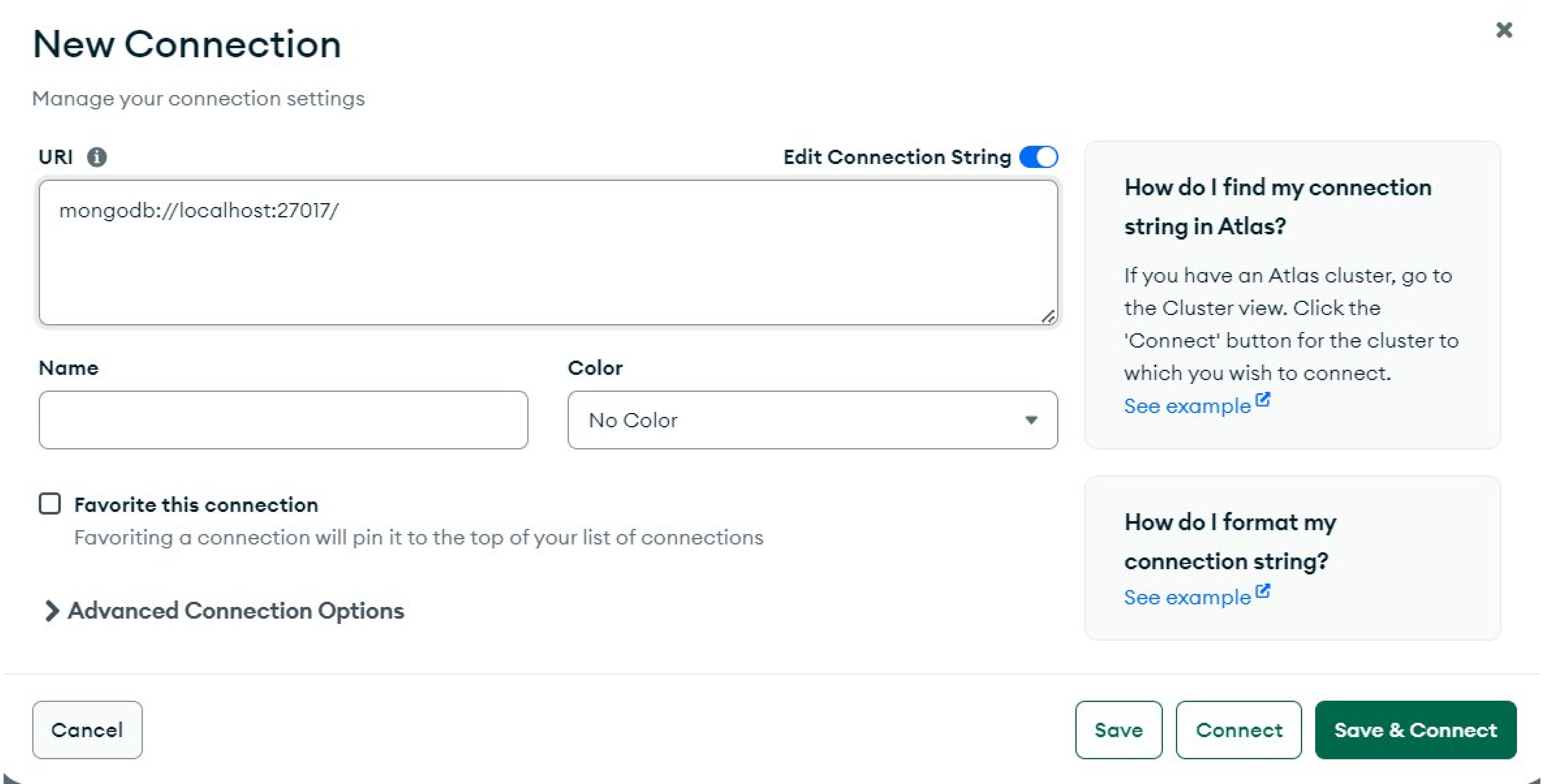
```
PS C:\Users\SUDIP> mongod --version
db version v8.0.4
Build Info: {
    "version": "8.0.4",
    "gitVersion": "bc35ab4305d9920d9d0491c1c9ef9b72383d31f9",
    "modules": [],
    "allocator": "tcmalloc-gperf",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}
PS C:\Users\SUDIP>
```



OS	Windows_
Path	C:\Progra
PATHEXT	.COM;.EXI
platformcode	KV
PROCESSOR_ARCHITECTU...	AMD64
PROCESSOR_IDENTIFIER	Intel64 Fa
PROCESSOR_LEVEL	6

C:\Program Files\Java\jre1.8.0_191\bin
C:\Program Files (x86)\sbt\bin
C:\spark\spark-2.3.2-bin-hadoop2.7\bin
C:\hive\bin
C:\derby\bin
C:\Program Files\PuTTY\
C:\Program Files\MongoDB\Server\8.0\bin
C:\Program Files\nodejs\
C:\Program Files\Git\cmd

MongoDB- a NoSQL Database



New Connection

Manage your connection settings

URI

Edit Connection String

```
mongodb://localhost:27017/
```

Name

Color: No Color

Favorite this connection

Favoriting a connection will pin it to the top of your list of connections

Advanced Connection Options

How do I find my connection string in Atlas?

If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.

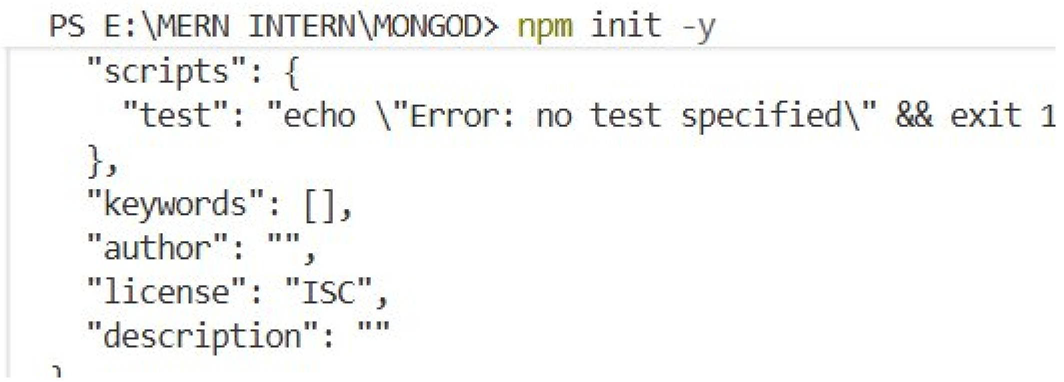
See example

How do I format my connection string?

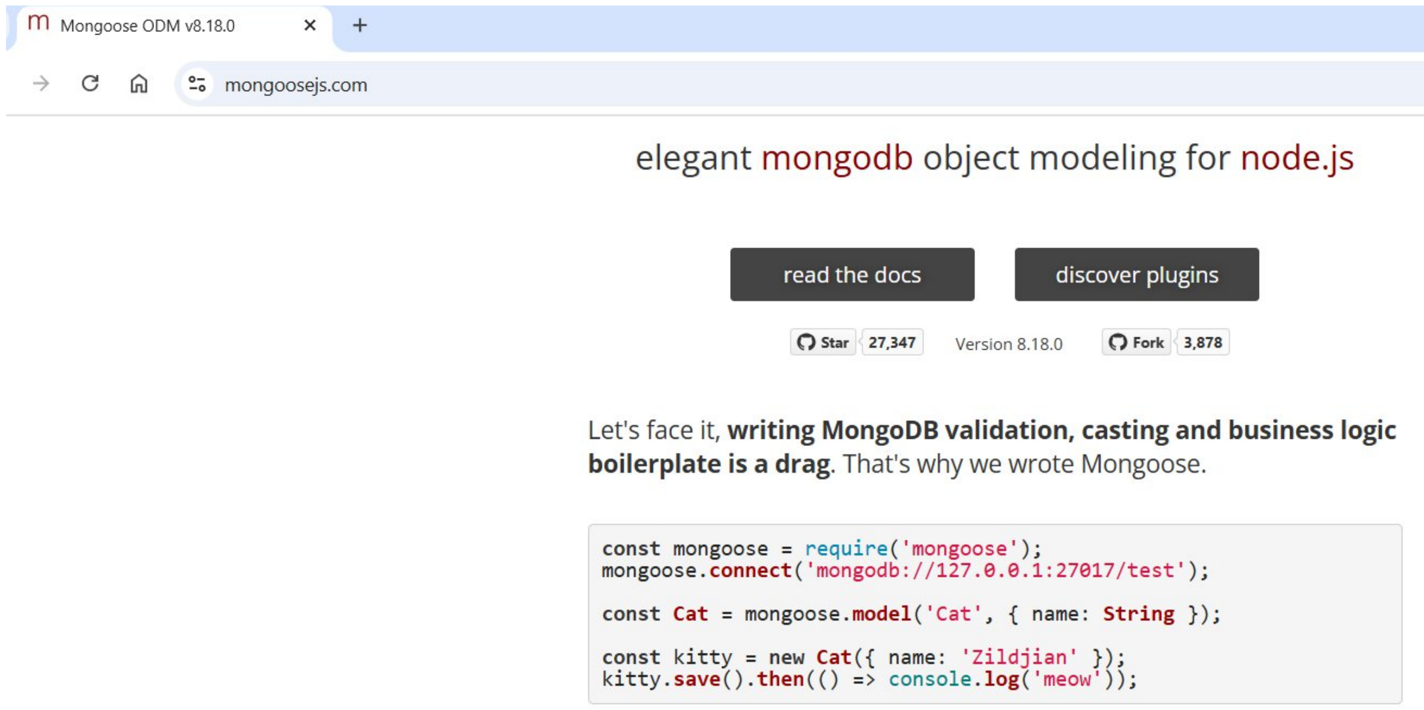
See example

MongoDB- a NoSQL Database

Connecting to MongoDB
>>npm init –y
>>npm i mongoose
>>



```
PS E:\MERN INTERN\MONGOD> npm init -y
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
```



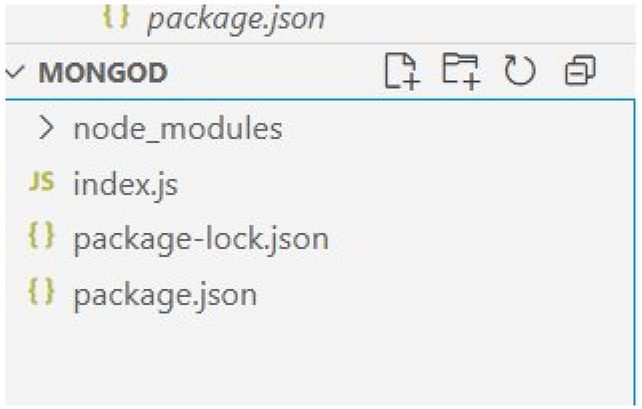
elegant mongodb object modeling for node.js

Let's face it, **writing MongoDB validation, casting and business logic boilerplate is a drag**. That's why we wrote Mongoose.

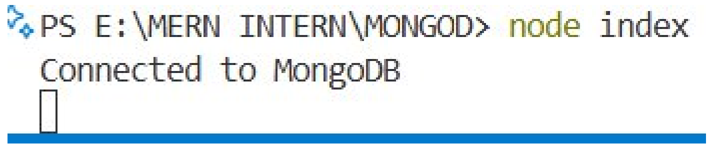
```
const mongoose = require('mongoose');
mongoose.connect('mongodb://127.0.0.1:27017/test');

const Cat = mongoose.model('Cat', { name: String });

const kitty = new Cat({ name: 'Zildjian' });
kitty.save().then(() => console.log('meow'));
```

MongoDB- a NoSQL Database

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/test').then(() => console.log
('Connected to MongoDB')).catch(err => console.error('Could not connect to MongoDB', err));
```



```
mongoose.connect('mongodb://loc
('Connected to MongoDB')).catch
```



```
PS E:\MERN INTERN\MONGOD> node index
Connected to MongoDB
```

MongoDB- a NoSQL Database

How to connect schema and models

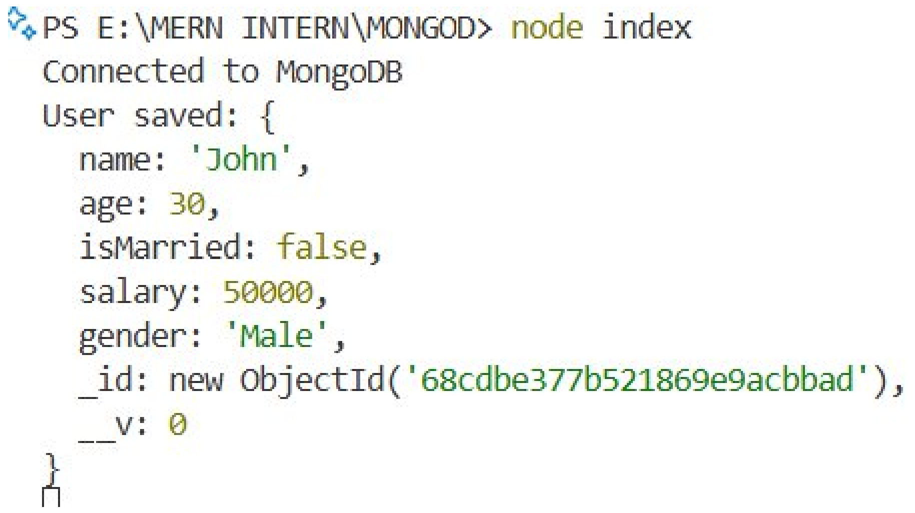
https://mongoosejs.com/docs/guide.html

```
//schema (shape of the document)
//Document - record in the database
//Collection - table in the database
//database - group of collections
const user  = [
    {name: 'John', age: 30},
    {name: 'Jane', age: 25},
    {name: 'Jim', age: 35}
]
```

```
//schema (shape of the document)
const userSchema = new mongoose.Schema({
    name: String,
    age: Number
    isMarried: Boolean,
    salary: Number,
    gender:String
});

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

MongoDB- a NoSQL Database

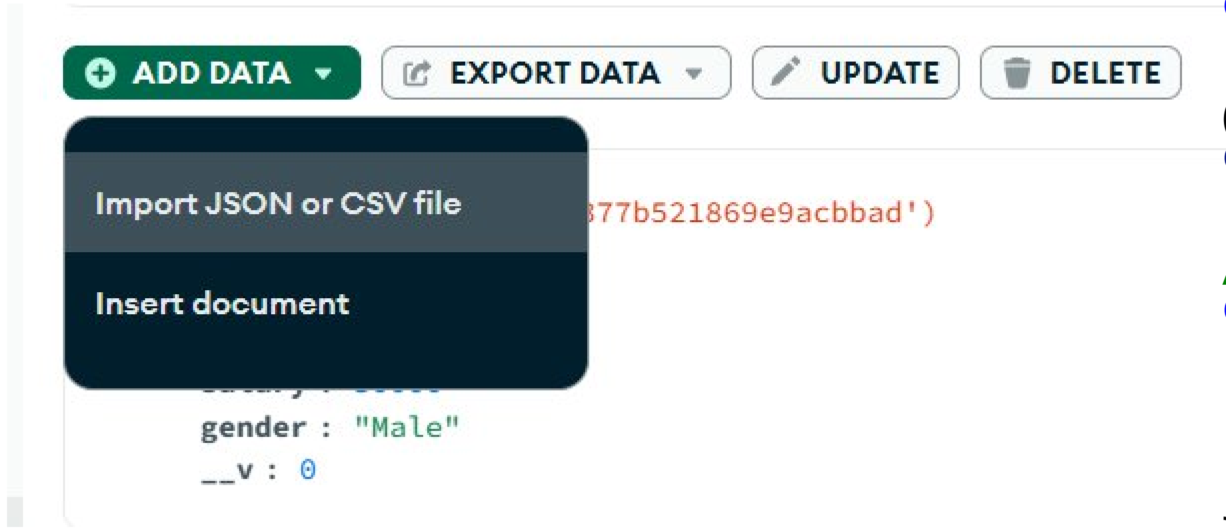
```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/test').then(() => console.log
('Connected to MongoDB')).catch(err => console.error('Could not connect to
MongoDB', err));


//schema (shape of the document)
const userSchema = new mongoose.Schema({
    name: String,
    age: Number,
    isMarried: Boolean,
    salary: Number,
    gender:String
});


const User = mongoose.model('User', userSchema);
async function storeInformation() {
const user = new User({
    name: 'John',
    age: 30,
    isMarried: false,
    salary: 50000,
    gender:'Male'
});
await user.save();
console.log('User saved:', user);
}


storeInformation();
```



```
PS E:\MERN INTERN\MONGOD> node index
Connected to MongoDB
User saved: {
  name: 'John',
  age: 30,
  isMarried: false,
  salary: 50000,
  gender: 'Male',
  _id: new ObjectId('68cdbe377b521869e9acbbad'),
  __v: 0
}
```

MongoDB- a NoSQL Database

Find documents in multiple ways-

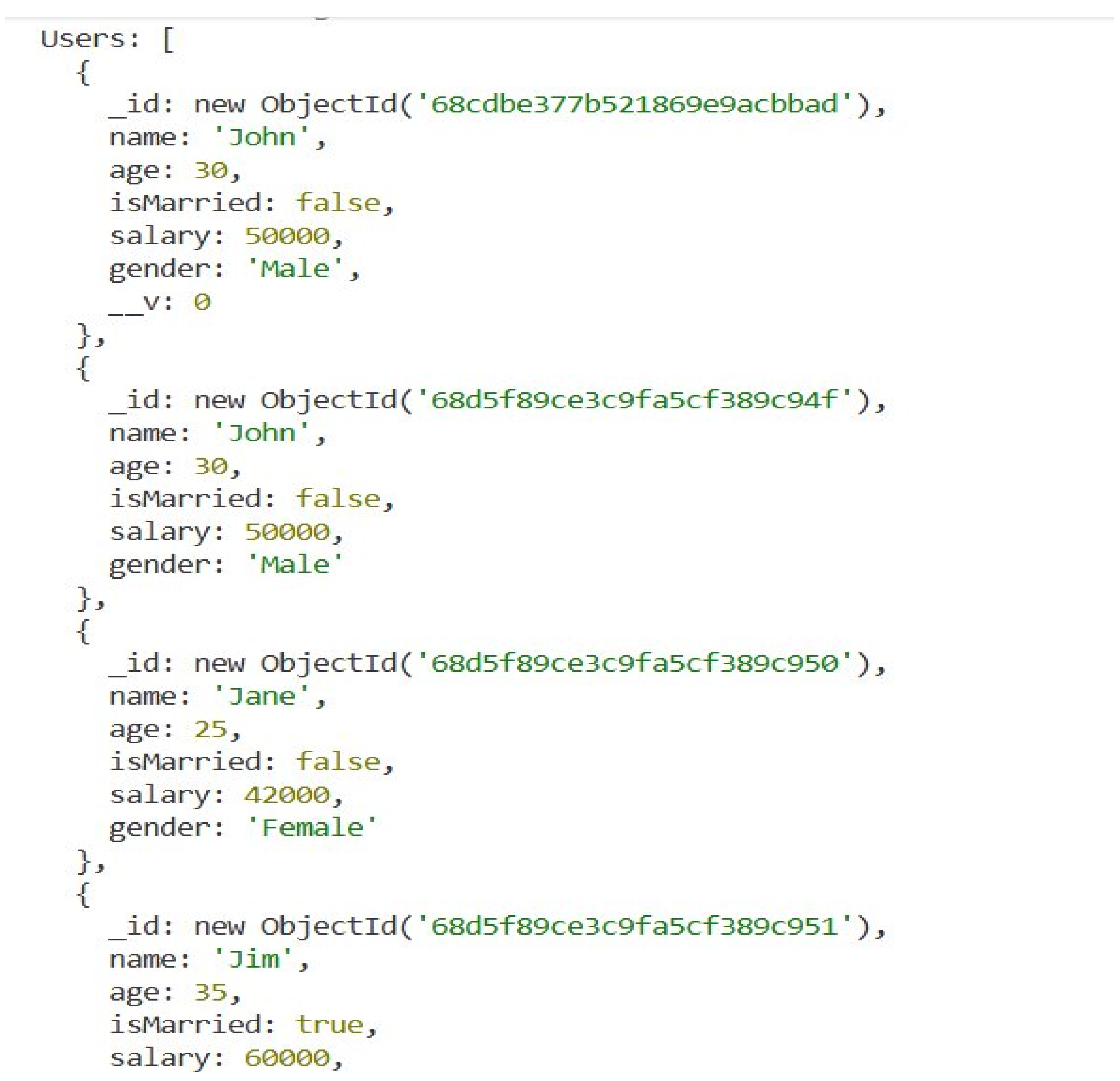


```
377b521869e9acbbad')
gender : "Male"
__v : 0
```

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/test').t
hen(() => console.log
('Connected to MongoDB')).catch(err =>
console.error('Could not connect to MongoDB', err));


//schema (shape of the document)
const userSchema = new mongoose.Schema({
    name: String,
    age: Number,
    isMarried: Boolean,
    salary: Number,
    gender:String
});


const User = mongoose.model('User', userSchema);
async function fetchInformation() {
    const users = await User.find({});
    console.log('Users:', users);
}


fetchInformation();
```

MongoDB- a NoSQL Database

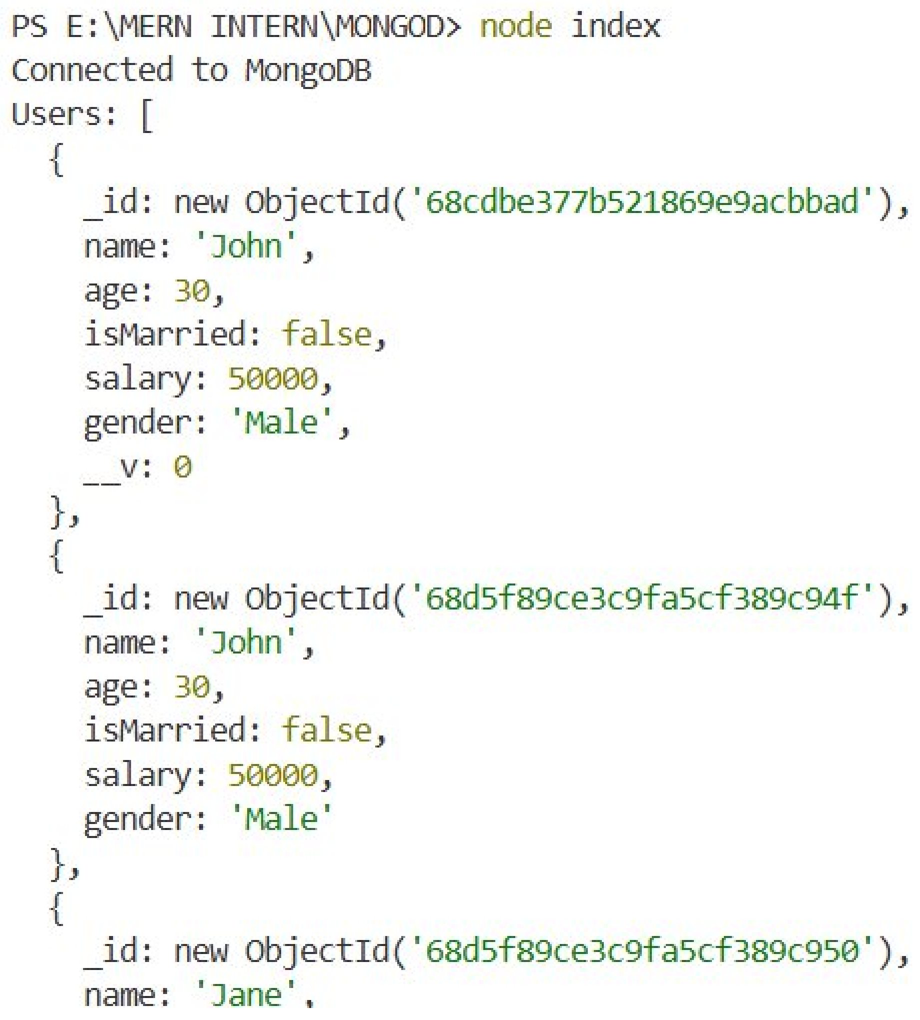
Array object data-



```
Users: [
  {
    _id: new ObjectId('68cdbe377b521869e9acbbad'),
    name: 'John',
    age: 30,
    isMarried: false,
    salary: 50000,
    gender: 'Male',
    __v: 0
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c94f'),
    name: 'John',
    age: 30,
    isMarried: false,
    salary: 50000,
    gender: 'Male'
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c950'),
    name: 'Jane',
    age: 25,
    isMarried: false,
    salary: 42000,
    gender: 'Female'
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c951'),
    name: 'Jim',
    age: 35,
    isMarried: true,
    salary: 60000,
```

MongoDB- a NoSQL Database

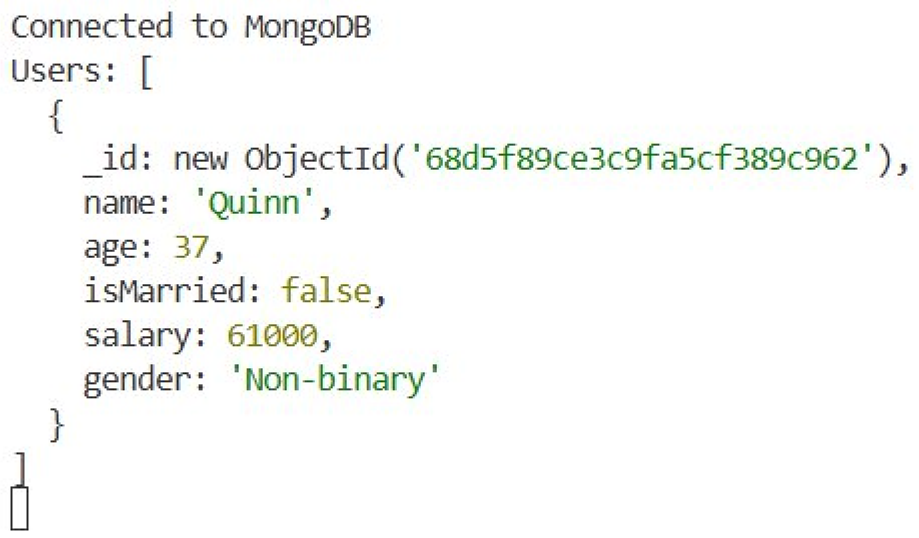
```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
    const users = await User.find({isMarried:false});
  console.log('Users:', users);
}

fetchInformation();
```



```
PS E:\MERN INTERN\MONGOD> node index
Connected to MongoDB
Users: [
  {
    _id: new ObjectId('68cdbe377b521869e9acbbad'),
    name: 'John',
    age: 30,
    isMarried: false,
    salary: 50000,
    gender: 'Male',
    __v: 0
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c94f'),
    name: 'John',
    age: 30,
    isMarried: false,
    salary: 50000,
    gender: 'Male'
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c950'),
    name: 'Jane',
```

MongoDB- a NoSQL Database

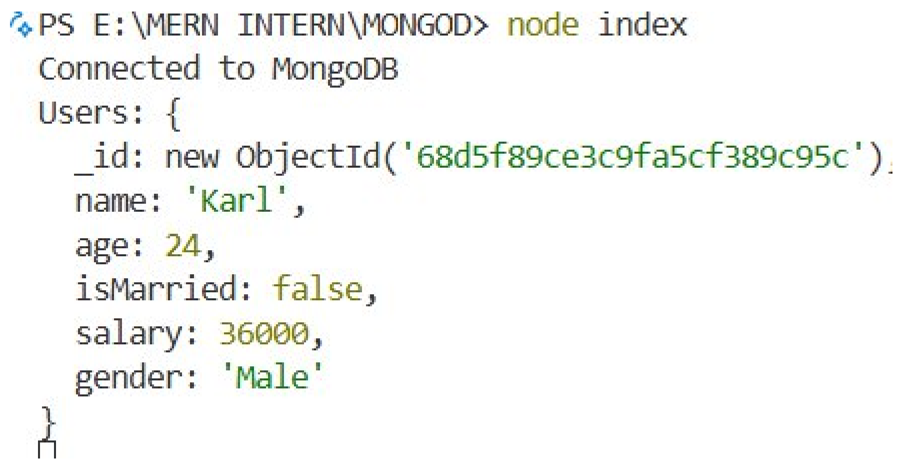
```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
    //const users = await User.find({});
    const users = await
User.find({isMarried:false,salary:61000});
    console.log('Users:', users);
}


fetchInformation();
```



```
Connected to MongoDB
Users: [
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c962'),
    name: 'Quinn',
    age: 37,
    isMarried: false,
    salary: 61000,
    gender: 'Non-binary'
  }
]
```

MongoDB- a NoSQL Database

```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
      const users = await
User.findById('68d5f89ce3c9fa5cf389c95c');
    console.log('Users:', users);
}


fetchInformation();
```

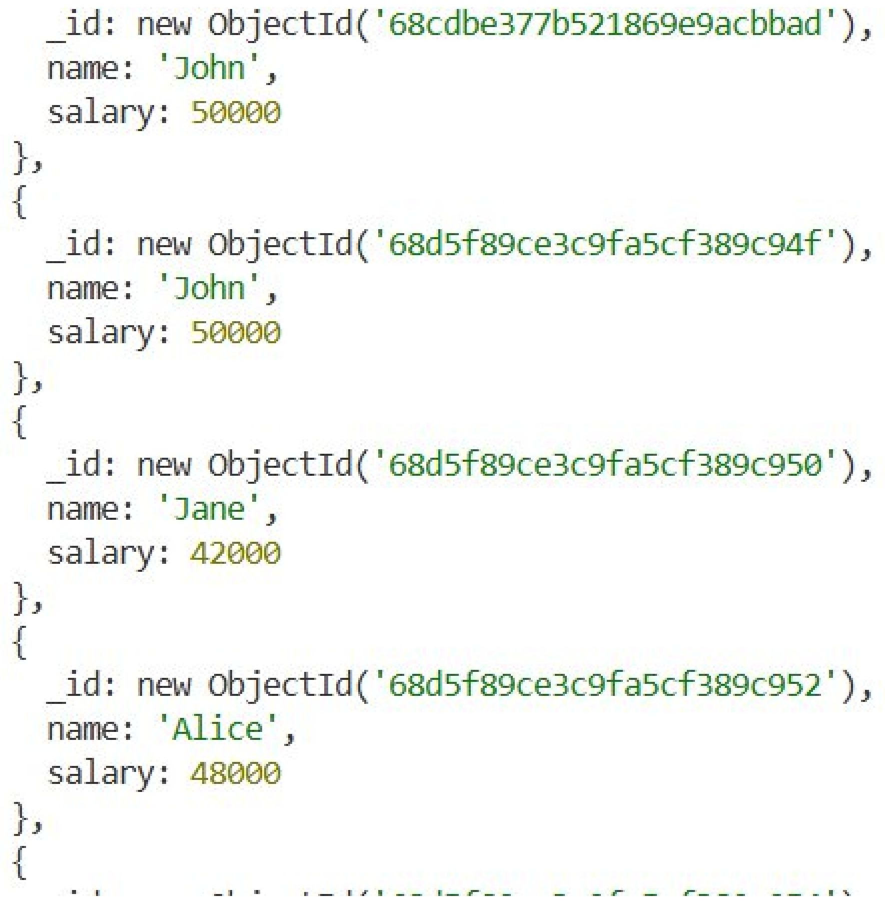


```
PS E:\MERN INTERN\MONGOD> node index
Connected to MongoDB
Users: {
  _id: new ObjectId('68d5f89ce3c9fa5cf389c95c'),
  name: 'Karl',
  age: 24,
  isMarried: false,
  salary: 36000,
  gender: 'Male'
}
```

MongoDB- a NoSQL Database

Query API Select, Sort, Limit, Count Documents-

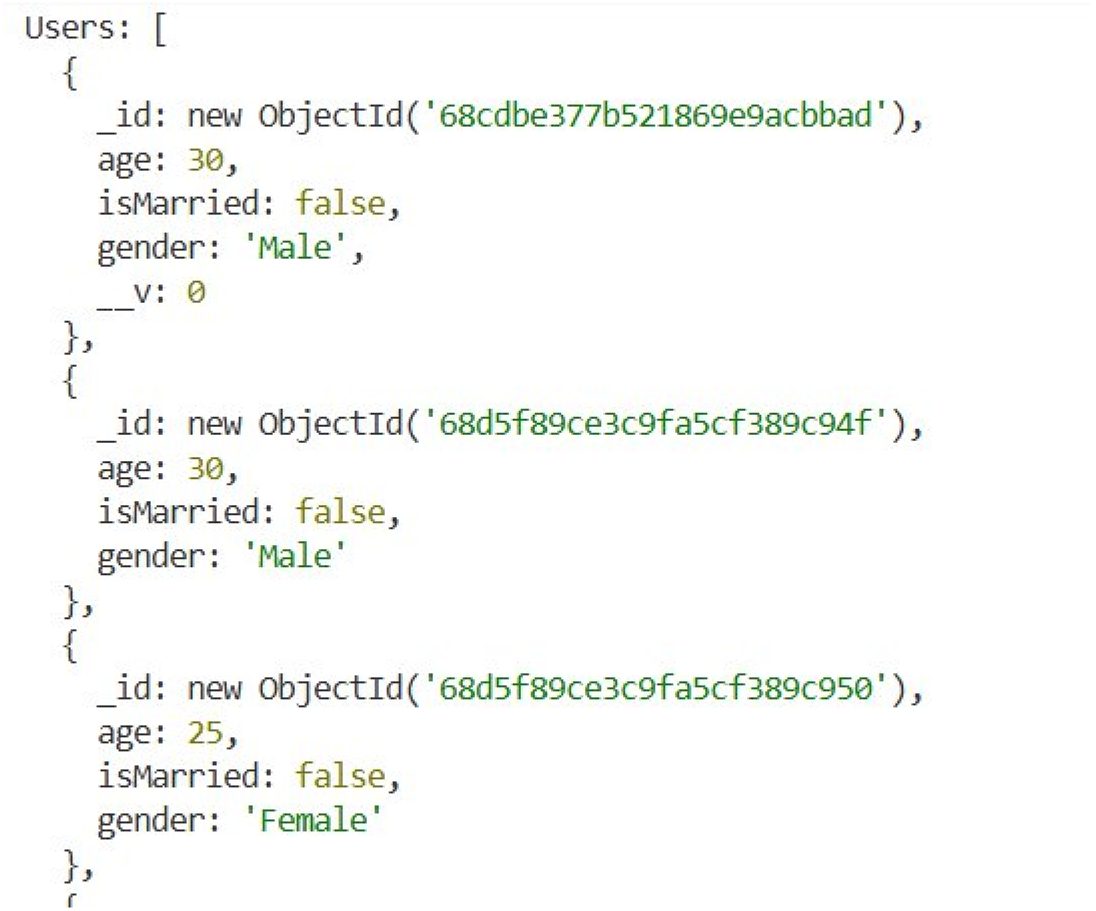
```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
    const users = await User.find({isMarried:false}).select('name
salary');
    console.log('Users:', users);
}

fetchInformation();
```



```
  _id: new ObjectId('68cdbe377b521869e9acbbad'),
  name: 'John',
  salary: 50000
},
{
  _id: new ObjectId('68d5f89ce3c9fa5cf389c94f'),
  name: 'John',
  salary: 50000
},
{
  _id: new ObjectId('68d5f89ce3c9fa5cf389c950'),
  name: 'Jane',
  salary: 42000
},
{
  _id: new ObjectId('68d5f89ce3c9fa5cf389c952'),
  name: 'Alice',
  salary: 48000
},
{
```

MongoDB- a NoSQL Database

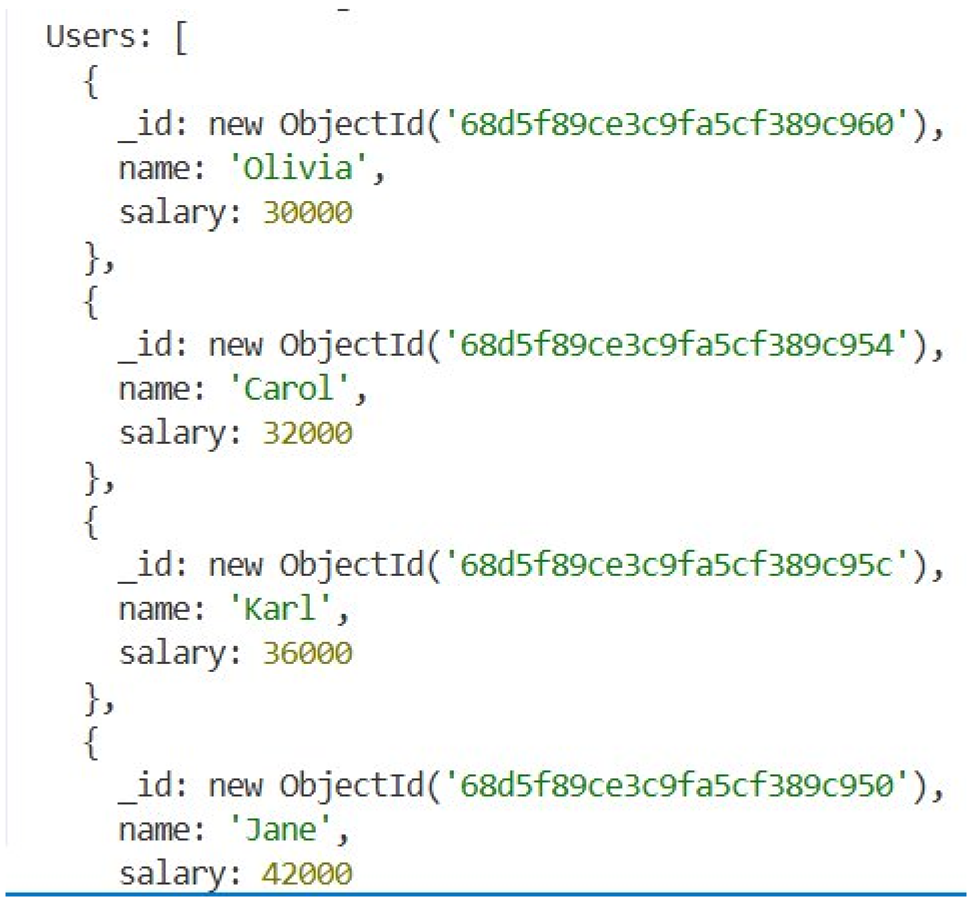
```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
    const users = await User.find({isMarried:false}).select('-name -
salary');
    console.log('Users:', users);
}

fetchInformation();
```



```
Users: [
  {
    _id: new ObjectId('68cdbe377b521869e9acbbad'),
    age: 30,
    isMarried: false,
    gender: 'Male',
    __v: 0
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c94f'),
    age: 30,
    isMarried: false,
    gender: 'Male'
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c950'),
    age: 25,
    isMarried: false,
    gender: 'Female'
  },
  {
```

MongoDB- a NoSQL Database

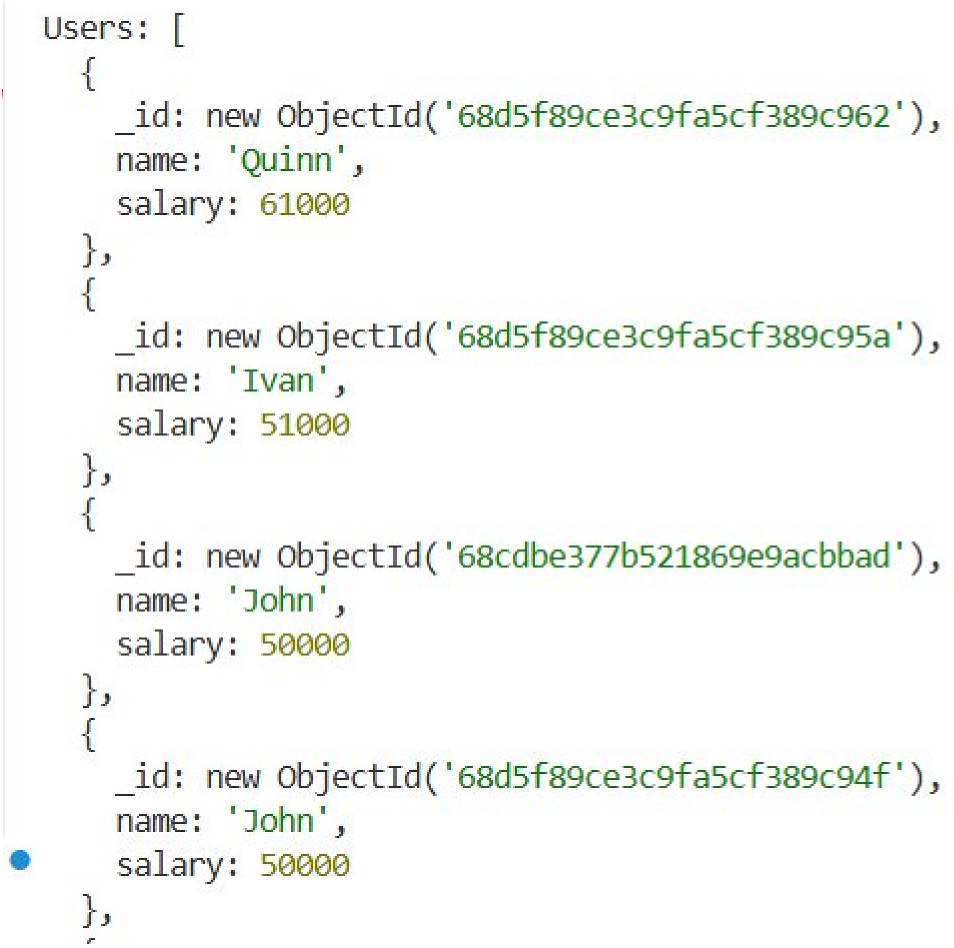
```
async function fetchInformation() {
    const users = await
User.find({isMarried:false}).select('name
salary').sort('salary');
    console.log('Users:', users);
}


fetchInformation();
```



```
Users: [
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c960'),
    name: 'Olivia',
    salary: 30000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c954'),
    name: 'Carol',
    salary: 32000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c95c'),
    name: 'Karl',
    salary: 36000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c950'),
    name: 'Jane',
    salary: 42000
```

MongoDB- a NoSQL Database

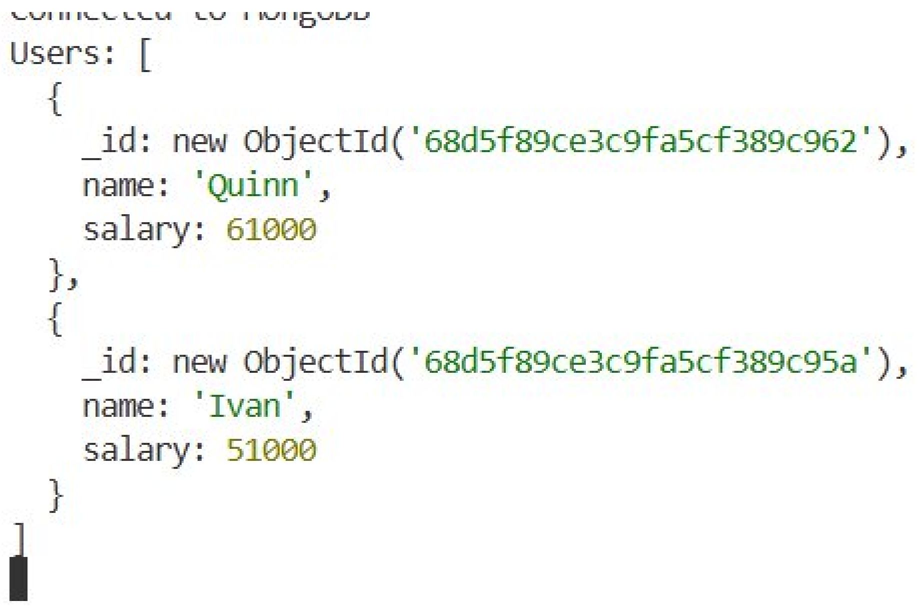
```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
    const users = await
User.find({isMarried:false}).select('name salary').sort('
salary');
    console.log('Users:', users);
}


fetchInformation();
```



```
Users: [
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c962'),
    name: 'Quinn',
    salary: 61000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c95a'),
    name: 'Ivan',
    salary: 51000
  },
  {
    _id: new ObjectId('68cdbe377b521869e9acbbad'),
    name: 'John',
    salary: 50000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c94f'),
    name: 'John',
    salary: 50000
  },
```

MongoDB- a NoSQL Database

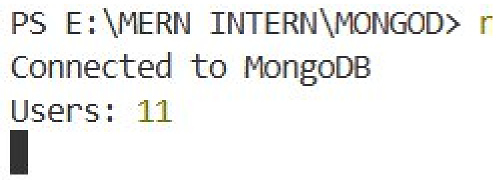
```
async function fetchInformation() {
    const users = await
User.find({isMarried:false}).select('name salary').sort('-
salary').limit(2);
    console.log('Users:', users);
}

fetchInformation();
```



```
Connected to MongoDB
Users: [
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c962'),
    name: 'Quinn',
    salary: 61000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c95a'),
    name: 'Ivan',
    salary: 51000
  }
]
```

MongoDB- a NoSQL Database

```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
    const users = await
User.find({isMarried:false}).select('name salary').sort('-
salary').countDocuments();
    console.log('Users:', users);
}

fetchInformation();
```

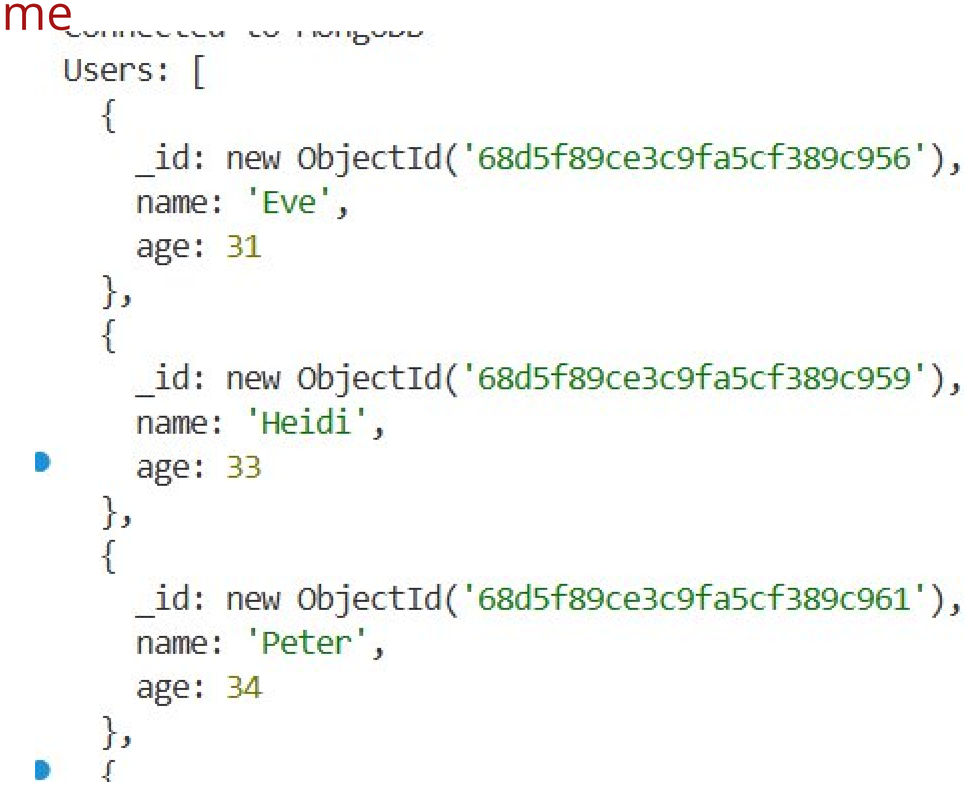


```
PS E:\MERN INTERN\MONGOD> r
Connected to MongoDB
Users: 11
```

MongoDB- a NoSQL Database

Comparation Operator-

```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
    const users = await User.find({age:{$gt:30}}).select('name
age').sort('age');
    console.log('Users:', users);
}

fetchInformation();
```

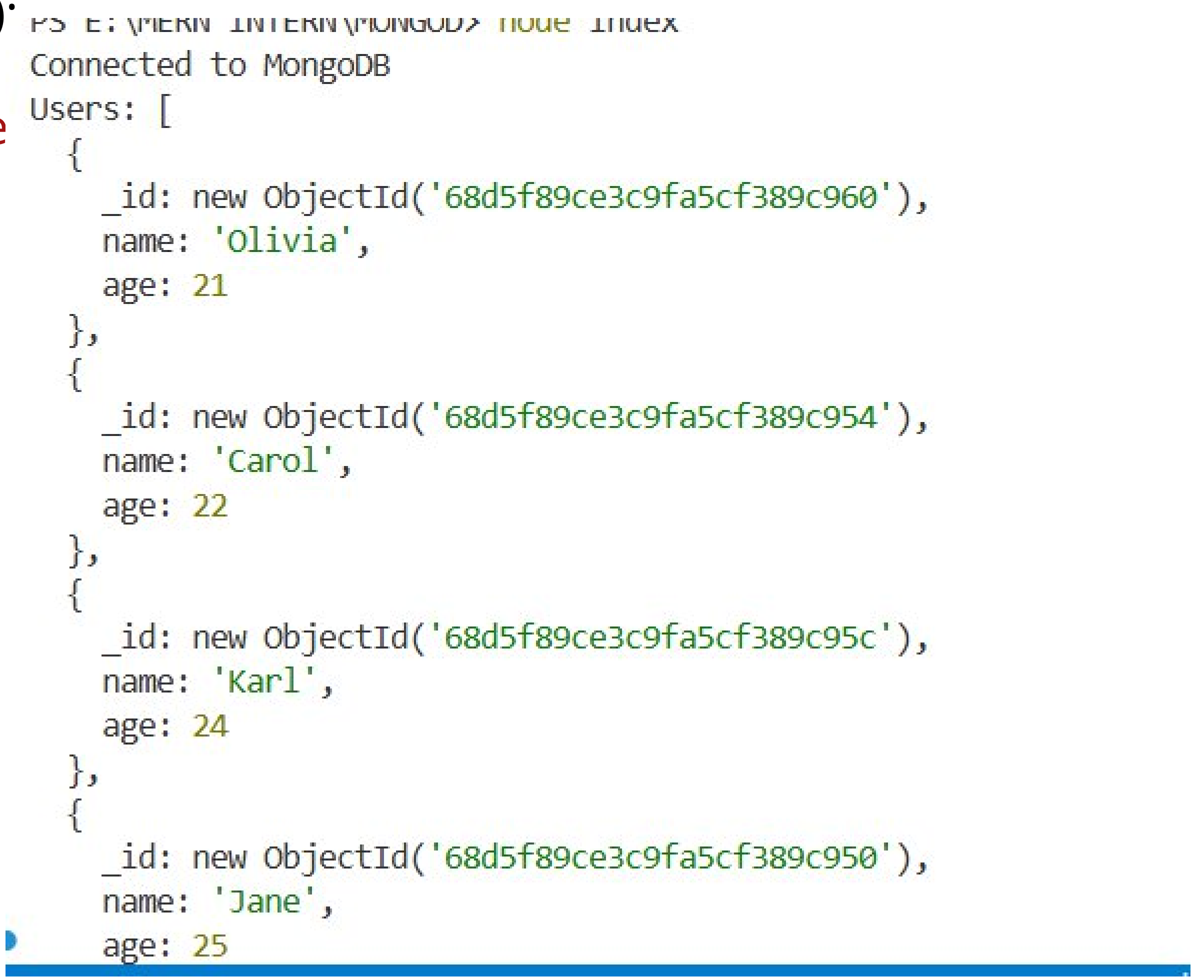


```
Users: [
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c956'),
    name: 'Eve',
    age: 31
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c959'),
    name: 'Heidi',
    age: 33
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c961'),
    name: 'Peter',
    age: 34
  },
  {
```

MongoDB- a NoSQL Database

```
const User = mongoose.model('User', userSchema)·
async function fetchInformation() {
    const users = await
User.find({age:{$lt:30}}).select('name age').sort('age
    console.log('Users:', users);
}


fetchInformation();
```

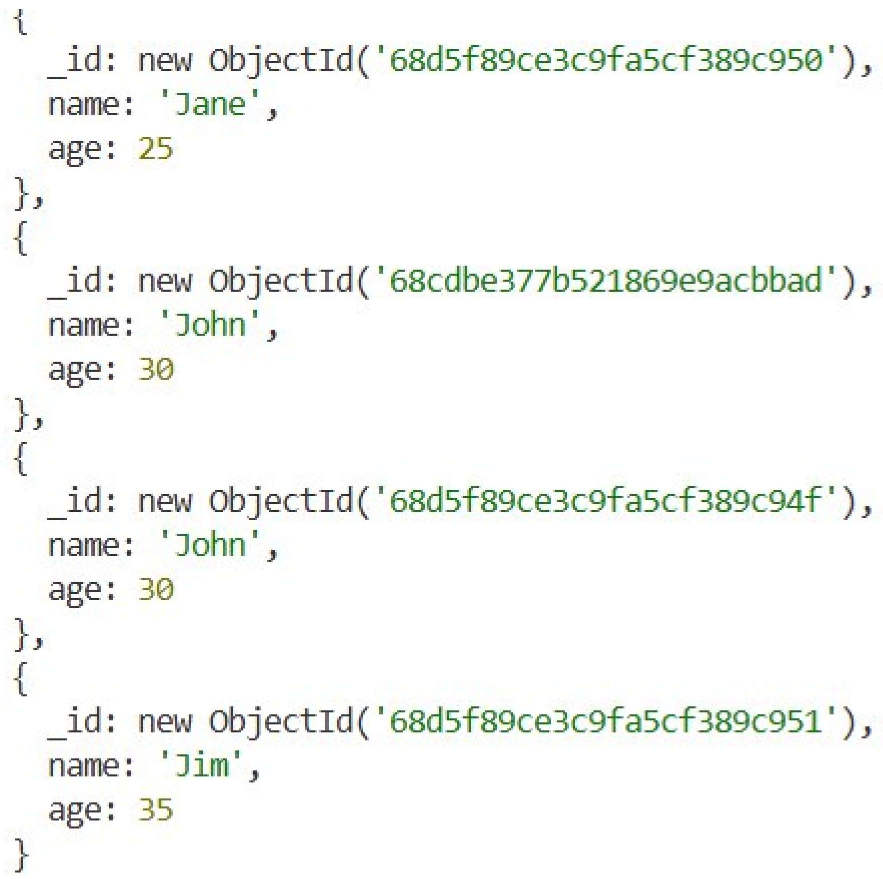
$gt
$lt
$lte
$gte
$lte
$in
$nin
$and
$or
$not
$nor
$exists
$expr
$regex
$mod
$size
$all
$elemMatch
$type



```
PS E:\MERN INTERN\MONGOD> node index
Connected to MongoDB
Users: [
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c960'),
    name: 'Olivia',
    age: 21
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c954'),
    name: 'Carol',
    age: 22
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c95c'),
    name: 'Karl',
    age: 24
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c950'),
    name: 'Jane',
    age: 25
```

MongoDB- a NoSQL Database

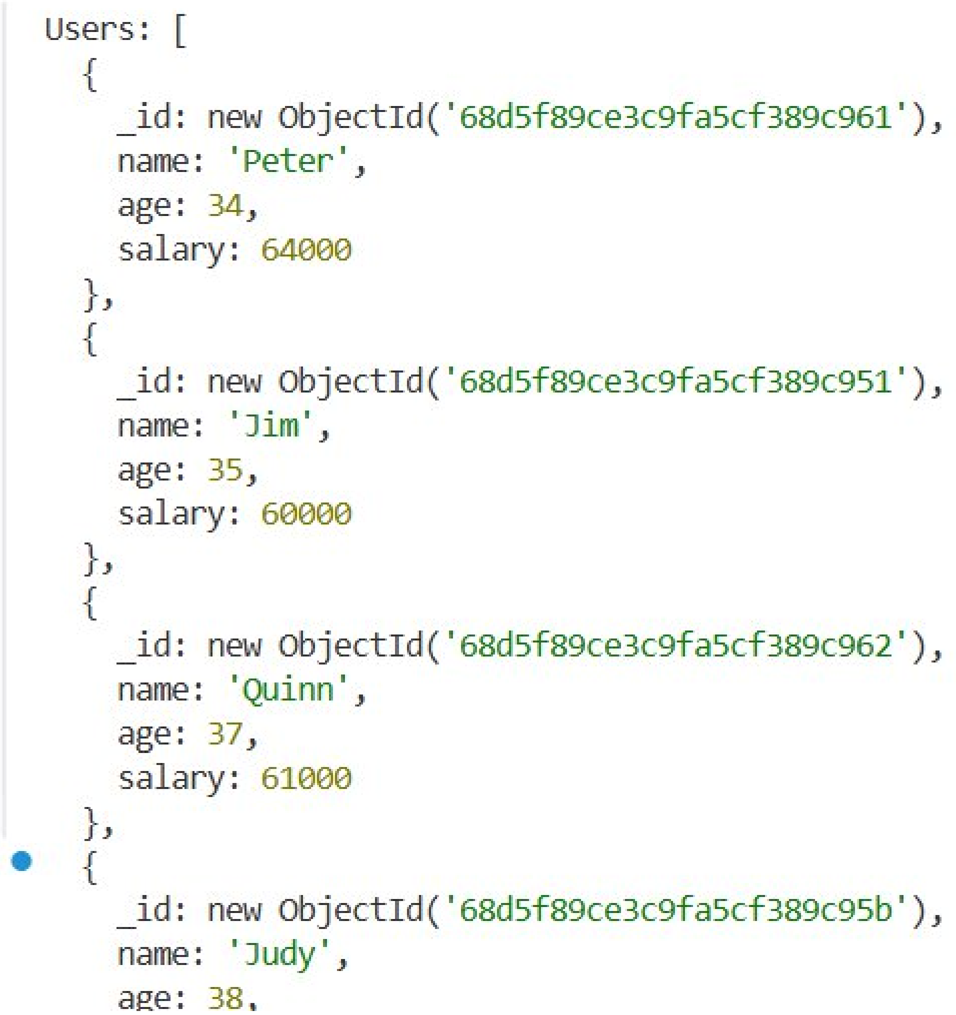
```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
    //in
    const users = await 
User.find({age:{$in:[25,30,35]}}).select('name 
age').sort('age');
    console.log('Users:', users);
}


fetchInformation();
```



```
{
  _id: new ObjectId('68d5f89ce3c9fa5cf389c950'),
  name: 'Jane',
  age: 25
},
{
  _id: new ObjectId('68cdbe377b521869e9acbbad'),
  name: 'John',
  age: 30
},
{
  _id: new ObjectId('68d5f89ce3c9fa5cf389c94f'),
  name: 'John',
  age: 30
},
{
  _id: new ObjectId('68d5f89ce3c9fa5cf389c951'),
  name: 'Jim',
  age: 35
}
```

MongoDB- a NoSQL Database

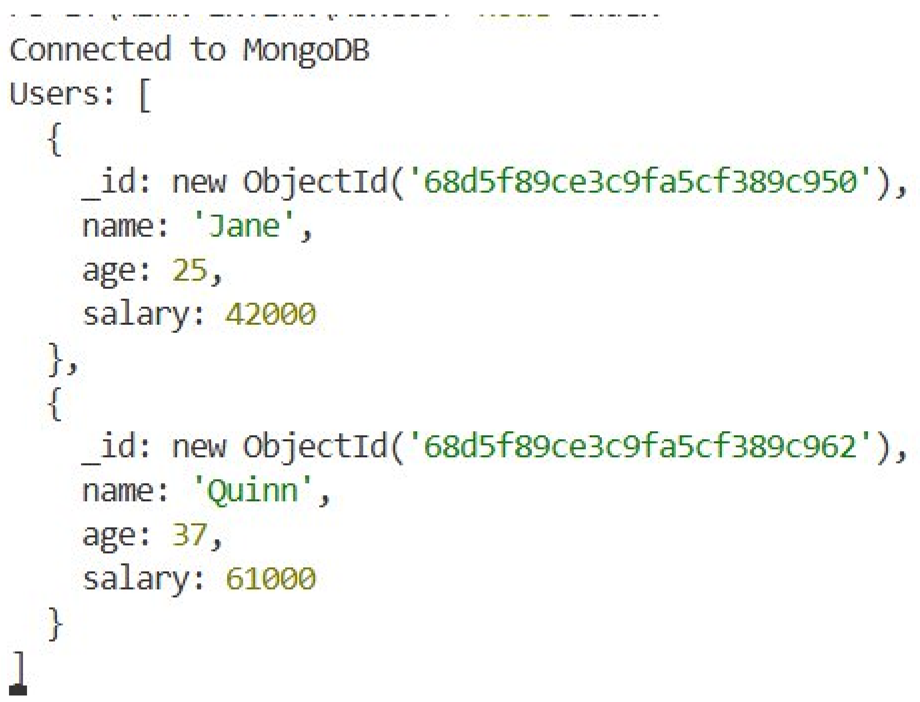
```
const User = mongoose.model('User',
userSchema);
async function fetchInformation() {
    //and
    const users = await
User.find({$and:[{age:{$gte:30}},{salary:{$gte:600
00}}]}).select('name age salary').sort('age');


    console.log('Users:', users);
}


fetchInformation();
```



```
Users: [
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c961'),
    name: 'Peter',
    age: 34,
    salary: 64000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c951'),
    name: 'Jim',
    age: 35,
    salary: 60000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c962'),
    name: 'Quinn',
    age: 37,
    salary: 61000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c95b'),
    name: 'Judy',
    age: 38,
```

MongoDB- a NoSQL Database

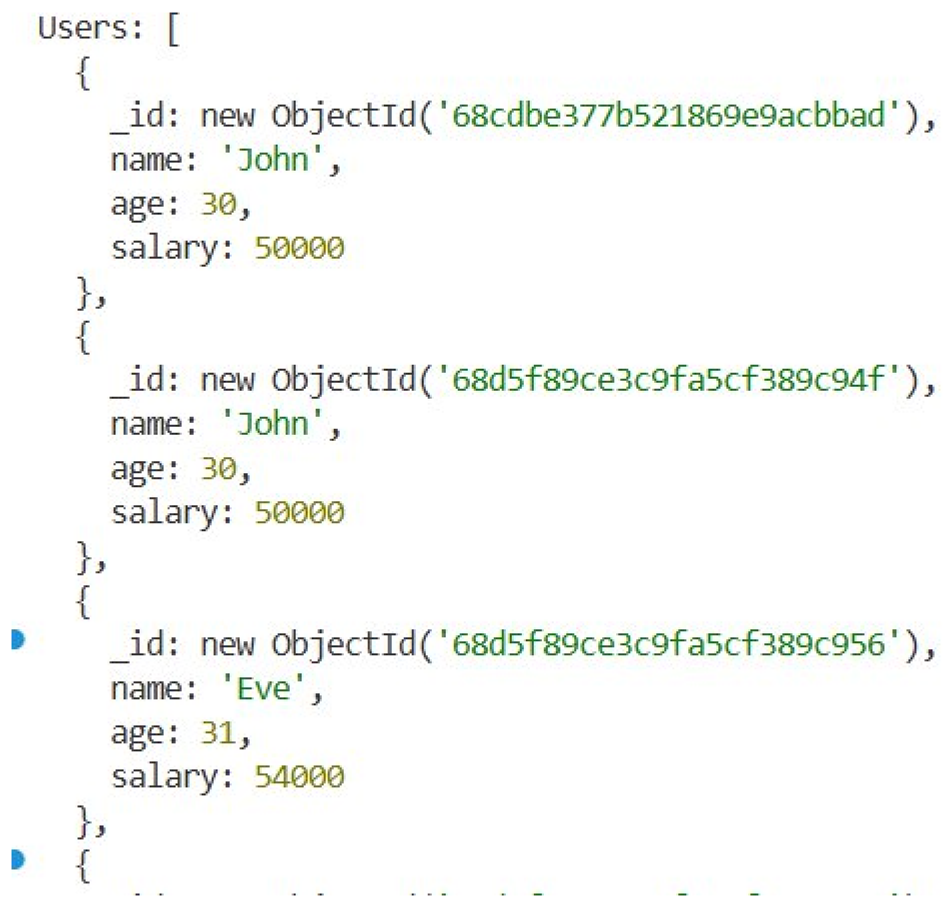
```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
    //or
    const users = await
User.find({$or:[{age:25},{salary:61000}]}).select('name
age salary').sort('age');

    console.log('Users:', users);
}


fetchInformation();
```



```
Connected to MongoDB
Users: [
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c950'),
    name: 'Jane',
    age: 25,
    salary: 42000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c962'),
    name: 'Quinn',
    age: 37,
    salary: 61000
  }
]
```

MongoDB- a NoSQL Database

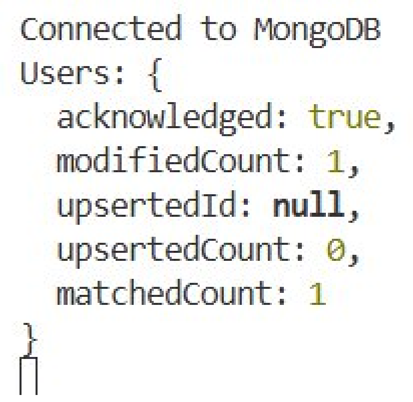
```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
    const users = await User.find({$or:[{age:{$gte:30}},{salary:{$gte:60000}}]}).select('name age
salary').sort('age');

    console.log('Users:', users);
}


fetchInformation();
```



```
Users: [
  {
    _id: new ObjectId('68cdbe377b521869e9acbbad'),
    name: 'John',
    age: 30,
    salary: 50000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c94f'),
    name: 'John',
    age: 30,
    salary: 50000
  },
  {
    _id: new ObjectId('68d5f89ce3c9fa5cf389c956'),
    name: 'Eve',
    age: 31,
    salary: 54000
  },
  {
```

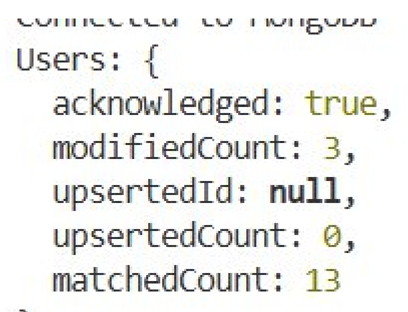
MongoDB- a NoSQL Database

```
const User = mongoose.model('User', userSchema);
async function fetchInformation() {
   //update a document
   const users = await
User.updateOne({_id:'68d5f89ce3c9fa5cf389c95c'},{$set:{name:'Ravi Kumar'}});
      console.log('Users:', users);
}
```



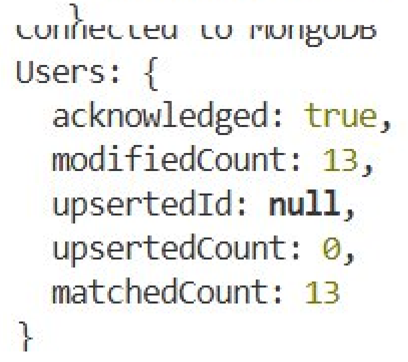
```
Connected to MongoDB
Users: {
  acknowledged: true,
  modifiedCount: 1,
  upsertedId: null,
  upsertedCount: 0,
  matchedCount: 1
}
```

```
 const users = await
User.updateMany({age:{$gte:30}},{$set:{isMarried:true}
});
```



```
Connected to MongoDB
Users: {
  acknowledged: true,
  modifiedCount: 3,
  upsertedId: null,
  upsertedCount: 0,
  matchedCount: 13
}
```

```
 const users = await
User.updateMany({age:{$gte:30}},{$set:{salary:70000}},{runValid
ators:true}});
```



```
Connected to MongoDB
Users: {
  acknowledged: true,
  modifiedCount: 13,
  upsertedId: null,
  upsertedCount: 0,
  matchedCount: 13
}
```

MongoDB- a NoSQL Database

```
//delete a document
//const users = await User.deleteOne({_id:'68d5f89ce3c9fa5cf389c95c'});
//delete multiple documents
const users = await User.deleteMany({age:{$gte:30}});
```

```
{ name: 'Alice', age: 28, isMarried: false, salary: 48000, gender: 'Female' },
{ name: 'Brian', age: 42, isMarried: true, salary: 72000, gender: 'Male' },
{ name: 'Charles', age: 33, isMarried: false, salary: 56000, gender: 'Male' },
{ name: 'Diana', age: 26, isMarried: false, salary: 41000, gender: 'Female' },
{ name: 'Ethan', age: 59, isMarried: true, salary: 95000, gender: 'Male' },
{ name: 'Fiona', age: 31, isMarried: true, salary: 60000, gender: 'Female' },
{ name: 'Gavin', age: 24, isMarried: false, salary: 35000, gender: 'Male' },
{ name: 'Hannah', age: 29, isMarried: false, salary: 45000, gender: 'Female' },
{ name: 'Ian', age: 38, isMarried: true, salary: 67000, gender: 'Male' },
{ name: 'Janet', age: 22, isMarried: false, salary: 30000, gender: 'Female' }
```

```
//mongo db problems case study
1. create a database named company
2. create a collection named employees
3. insert 10 documents
4. fetch all employees
5. fetch employees whose age is greater than 30
6. fetch employees whose salary is less than or equal to 50000
7. fetch employees who are not married
8. fetch employees whose age is between 25 and 35
9. fetch employees whose name starts with 'A'
10. fetch employees whose name ends with 'n'
11. fetch employees whose name contains ‘Br'
12. fetch employees whose salary is in the range of 40000 to 60000
13. fetch employees whose age is either 25 or 30
14. fetch employees whose age is greater than 30 and salary is greater than 50000
15. fetch employees whose age is less than 30 or salary is less than 40000
```

Thank you!!