U18ISI6203T-Internet and Web Programming

KUMARAGURU COLLEGE OF TECHNOLOGY

LABORATORY MANUAL

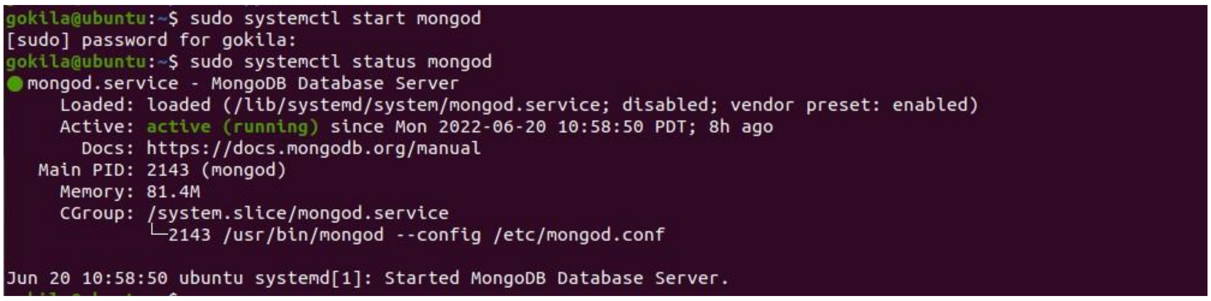
NAME	: N.M.GOKILA
ROLL NO	:19BIS040
DEPARTMENT	:ISE
COURSE CODE	: U18ISI6203T
YEAR	:III YEAR
SEMESTER	: VI SEMESTER

Internet and Web Programming

LAB MANUAL

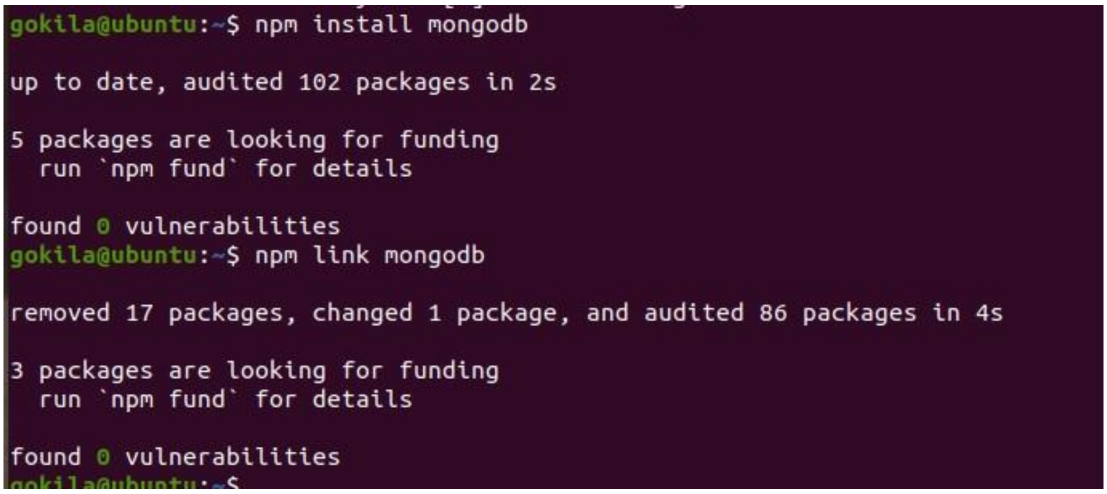
SIGN: **DATE:21-06-2021**

Start a mongodb server:



```
gokila@ubuntu:~$ sudo systemctl start mongod
[sudo] password for gokila:
gokila@ubuntu:~$ sudo systemctl status mongod
● mongod.service - MongoDB Database Server
     Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset: enabled)
     Active: active (running) since Mon 2022-06-20 10:58:50 PDT; 8h ago
       Docs: https://docs.mongodb.org/manual
   Main PID: 2143 (mongod)
     Memory: 81.4M
     CGroup: /system.slice/mongod.service
             └─2143 /usr/bin/mongod --config /etc/mongod.conf

Jun 20 10:58:50 ubuntu systemd[1]: Started MongoDB Database Server.
```

Link nodejs and mongodb:



```
gokila@ubuntu:~$ npm install mongodb

up to date, audited 102 packages in 2s

5 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
gokila@ubuntu:~$ npm link mongodb

removed 17 packages, changed 1 package, and audited 86 packages in 4s

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
gokila@ubuntu:~$
```

1. Creating a Database

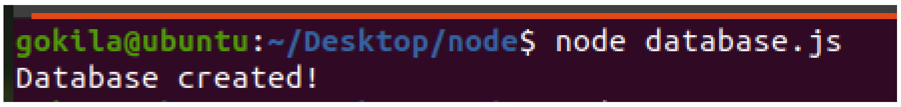
To create a database in MongoDB, start by creating a MongoClient object, then specify a connection URL with the correct ip address and the name of the database you want to create.

Code:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

Save the code above in a file called "database.js" and run the file:

Snap:



```
gokila@ubuntu:~/Desktop/node$ node database.js
Database created!
```

2. Create Collection:

A collection in MongoDB is the same as a table in MySQL

Create a collection called "customer":

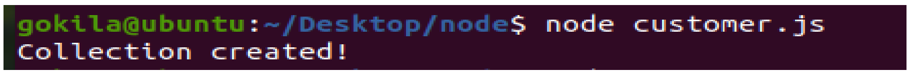
CODE:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.createCollection("customer", function(err, res) {
    if (err) throw err;
    console.log("Collection created!");
    db.close();
  });
});
```

Save the code above in a file called "customer.js" and run the file:

Run "node customer.js"

SNAP:



```
gokila@ubuntu:~/Desktop/node$ node customer.js
Collection created!
```

3. Insert Into Collection:

A collection in MongoDB is the same as a table in MySQL.

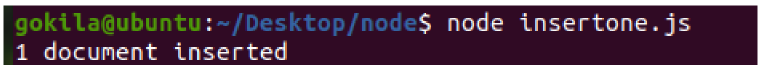
To insert a record, or document as it is called in MongoDB, into a collection, we use the insertOne() method.

CODE:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myobj = { name: "Company Inc", address: "Highway 37" };
  dbo.collection("customer").insertOne(myobj, function(err, res) {
    if (err) throw err;
    console.log("1 document inserted");
    db.close();
  });
});
```

Save the code above in a file called "insertone.js" and run the file:

SNAP:



```
gokila@ubuntu:~/Desktop/node$ node insertone.js
1 document inserted
```

4. Insert Multiple Documents

To insert multiple documents into a collection in MongoDB, we use the insertMany() method.

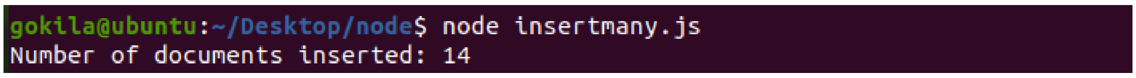
CODE:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myobj = [
    { name: 'John', address: 'Highway 71'},
    { name: 'Peter', address: 'Lowstreet 4'},
    { name: 'Amy', address: 'Apple st 652'},
    { name: 'Hannah', address: 'Mountain 21'},
    { name: 'Michael', address: 'Valley 345'},
    { name: 'Sandy', address: 'Ocean blvd 2'},
    { name: 'Betty', address: 'Green Grass 1'},
    { name: 'Richard', address: 'Sky st 331'},
    { name: 'Susan', address: 'One way 98'},
    { name: 'Vicky', address: 'Yellow Garden 2'},
    { name: 'Ben', address: 'Park Lane 38'},
    { name: 'William', address: 'Central st 954'},
    { name: 'Chuck', address: 'Main Road 989'},
    { name: 'Viola', address: 'Sideway 1633'}
  ];
  dbo.collection("customer").insertMany(myobj, function(err, res) {
    if (err) throw err;
    console.log("Number of documents inserted: " + res.insertedCount);
    db.close();
  });
});
```

Save the code above in a file called "insertmany.js" and run the file:

SNAP:



```
gokila@ubuntu:~/Desktop/node$ node insertmany.js
Number of documents inserted: 14
```

5.Find One:

To select data from a collection in MongoDB, we can use the findOne() method.The findOne() method returns the first occurrence in the selection.

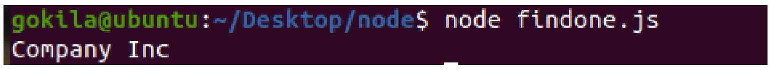
Find the first document in the customers collection:

Code:

```
var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {

  if (err) throw err;

  var dbo = db.db("mydb");

  dbo.collection("customer").findOne({}, function(err, result) {

    if (err) throw err;

    console.log(result.name);

    db.close();

  });

});
```

Save the code above in a file called "findone.js" and run the file:

Snap:



```
gokila@ubuntu:~/Desktop/node$ node findone.js
Company Inc
```

5. Find All

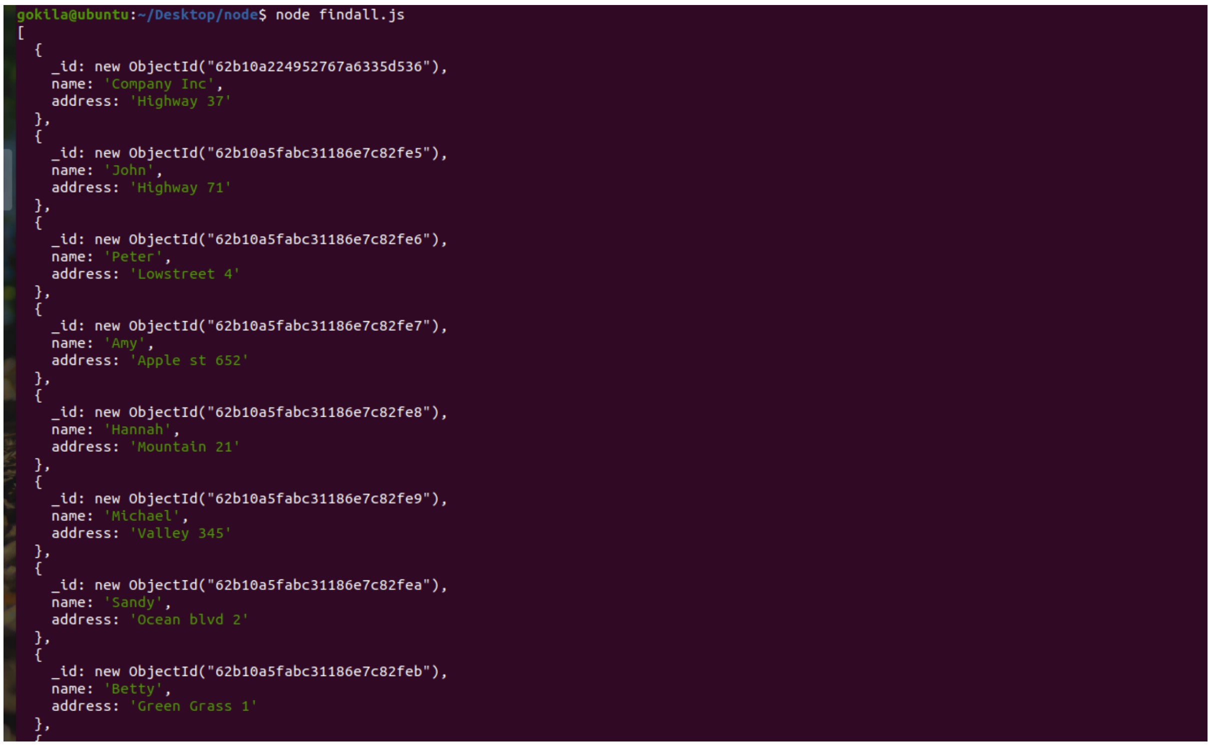
Select data from a table in MongoDB, we can also use the find() method.The find() method returns all occurrences in the selection.

Code:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.collection("customer").find({}).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Save the code above in a file called "findall.js" and run the file:

Snap:



```
gokila@ubuntu:~/Desktop/node$ node findall.js
[
  {
    _id: new ObjectId("62b10a224952767a6335d536"),
    name: 'Company Inc',
    address: 'Highway 37'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe5"),
    name: 'John',
    address: 'Highway 71'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe6"),
    name: 'Peter',
    address: 'Lowstreet 4'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe7"),
    name: 'Amy',
    address: 'Apple st 652'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe8"),
    name: 'Hannah',
    address: 'Mountain 21'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe9"),
    name: 'Michael',
    address: 'Valley 345'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fea"),
    name: 'Sandy',
    address: 'Ocean blvd 2'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82feb"),
    name: 'Betty',
    address: 'Green Grass 1'
  },
  {
```

7.Find Some

The second parameter of the find() method is the projection object that describes which fields to include in the result.

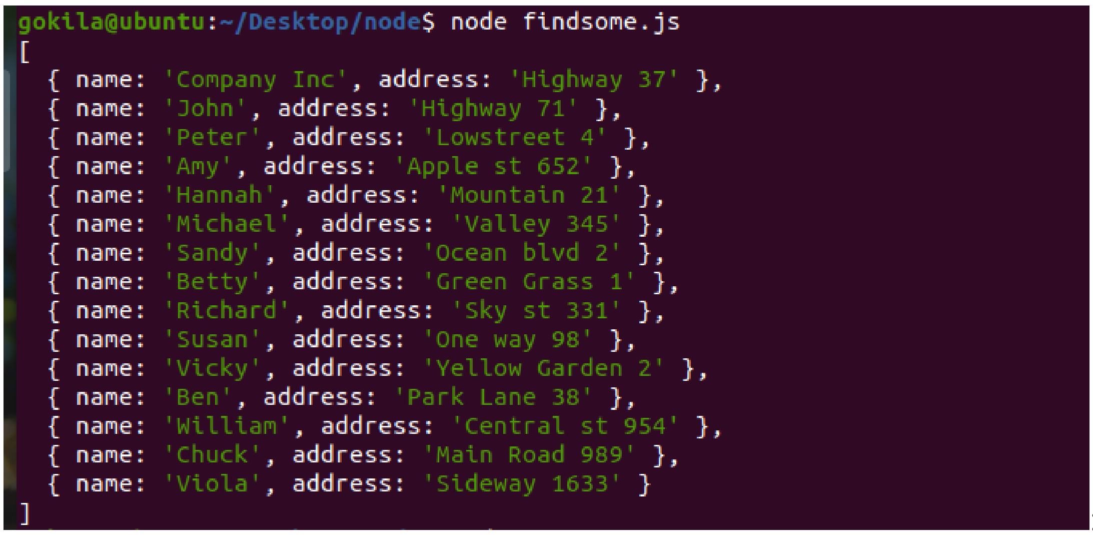
Return the fields "name" and "address" of all documents in the customers collection:

Code:

```
var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {

  if (err) throw err;

  var dbo = db.db("mydb");

  dbo.collection("customer").find({}, { projection: { _id: 0, name: 1, address: 1
} }).toArray(function(err, result) {

    if (err) throw err;

    console.log(result);

    db.close();

  });

});
```

Save the code above in a file called "findsome.js" and run the file:

Snap:



```
gokila@ubuntu:~/Desktop/node$ node findsome.js
[
  { name: 'Company Inc', address: 'Highway 37' },
  { name: 'John', address: 'Highway 71' },
  { name: 'Peter', address: 'Lowstreet 4' },
  { name: 'Amy', address: 'Apple st 652' },
  { name: 'Hannah', address: 'Mountain 21' },
  { name: 'Michael', address: 'Valley 345' },
  { name: 'Sandy', address: 'Ocean blvd 2' },
  { name: 'Betty', address: 'Green Grass 1' },
  { name: 'Richard', address: 'Sky st 331' },
  { name: 'Susan', address: 'One way 98' },
  { name: 'Vicky', address: 'Yellow Garden 2' },
  { name: 'Ben', address: 'Park Lane 38' },
  { name: 'William', address: 'Central st 954' },
  { name: 'Chuck', address: 'Main Road 989' },
  { name: 'Viola', address: 'Sideway 1633' }
]
```

:

8. Filter With Regular Expressions:

You can write regular expressions to find exactly what you are searching for.Regular expressions can only be used to query strings.

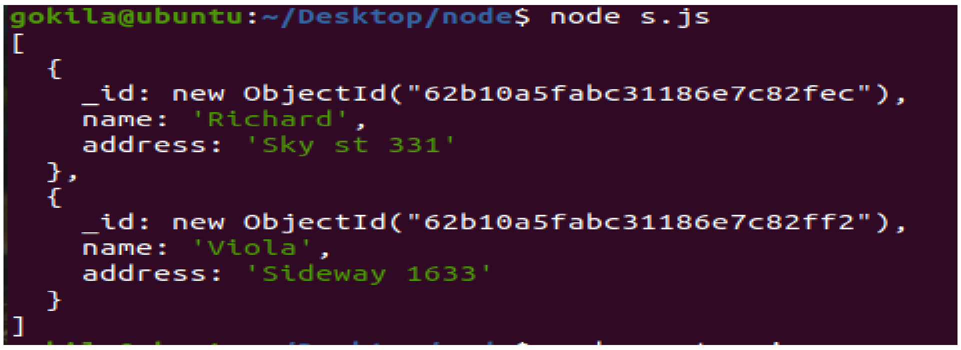
To find only the documents where the "address" field starts with the letter "S", use the regular expression /^S/:

CODE:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";


MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var query = { address: /^S/ };
  dbo.collection("customer").find(query).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Save the code above in a file called "s.js" and run the file:

SNAP:



```
gokila@ubuntu:~/Desktop/node$ node s.js
[
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fec"),
    name: 'Richard',
    address: 'Sky st 331'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82ff2"),
    name: 'Viola',
    address: 'Sideway 1633'
  }
]
```

9.Filter the Result

When finding documents in a collection, you can filter the result by using a query object. The first argument of the find() method is a query object, and is used to limit the search.

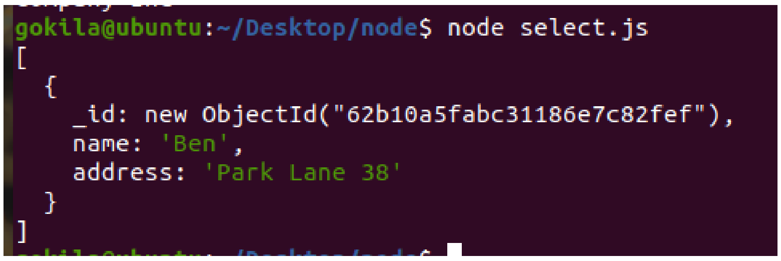
Find documents with the address "Park Lane 38":

CODE:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var query = { address: "Park Lane 38" };
  dbo.collection("customer").find(query).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Save the code above in a file called "select.js" and run the file:

SNAP:



```
gokila@ubuntu:~/Desktop/node$ node select.js
[
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fef"),
    name: 'Ben',
    address: 'Park Lane 38'
  }
]
```

10. Sort the result alphabetically by name:

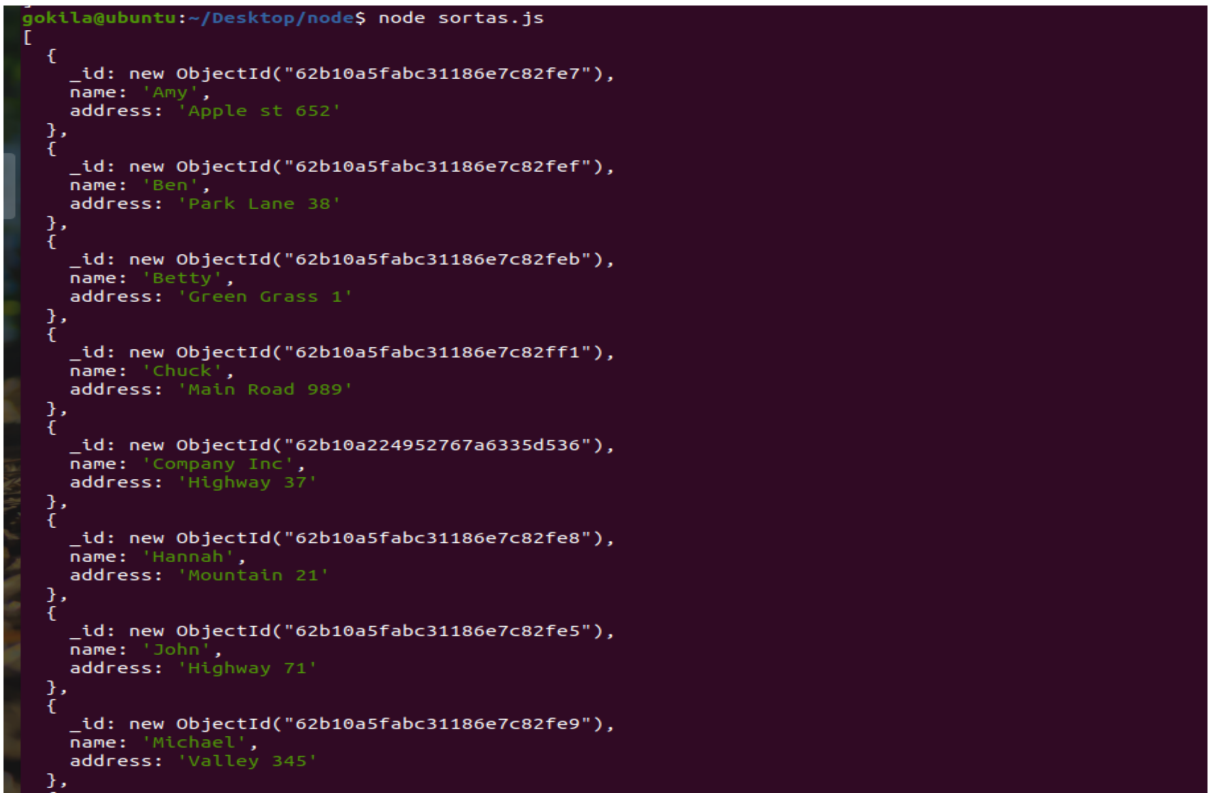
Use the sort() method to sort the result in ascending or descending order.

Code:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var mysort = { name: 1 };
  dbo.collection("customer").find().sort(mysort).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Save the code above in a file called "sortas.js" and run the file:

Snap:



```
gokila@ubuntu:~/Desktop/node$ node sortas.js
[
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe7"),
    name: 'Amy',
    address: 'Apple st 652'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fef"),
    name: 'Ben',
    address: 'Park Lane 38'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82feb"),
    name: 'Betty',
    address: 'Green Grass 1'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82ff1"),
    name: 'Chuck',
    address: 'Main Road 989'
  },
  {
    _id: new ObjectId("62b10a224952767a6335d536"),
    name: 'Company Inc',
    address: 'Highway 37'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe8"),
    name: 'Hannah',
    address: 'Mountain 21'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe5"),
    name: 'John',
    address: 'Highway 71'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe9"),
    name: 'Michael',
    address: 'Valley 345'
  },
```

11. Sort Descending

Use the value -1 in the sort object to sort descending.

{ name: 1 } // ascending

{ name: -1 } // descending

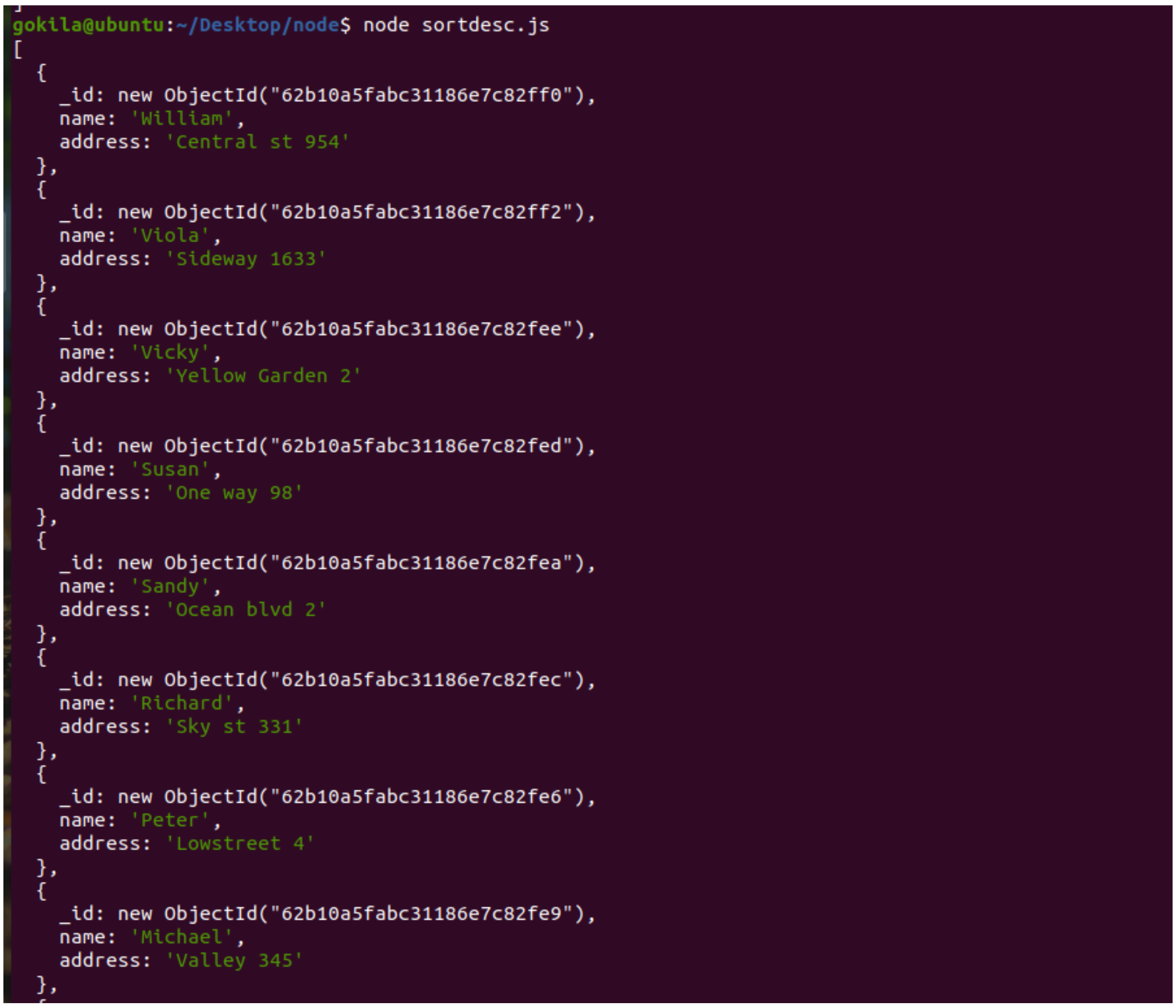
Sort the result reverse alphabetically by name:

Code:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var mysort = { name: -1 };
  dbo.collection("customer").find().sort(mysort).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Save the code above in a file called "sortdesc.js" and run the file:

Snap:



```
gokila@ubuntu:~/Desktop/node$ node sortdesc.js
[
  {
    _id: new ObjectId("62b10a5fabc31186e7c82ff0"),
    name: 'William',
    address: 'Central st 954'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82ff2"),
    name: 'Viola',
    address: 'Sideway 1633'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fee"),
    name: 'Vicky',
    address: 'Yellow Garden 2'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fed"),
    name: 'Susan',
    address: 'One way 98'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fea"),
    name: 'Sandy',
    address: 'Ocean blvd 2'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fec"),
    name: 'Richard',
    address: 'Sky st 331'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe6"),
    name: 'Peter',
    address: 'Lowstreet 4'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe9"),
    name: 'Michael',
    address: 'Valley 345'
  },
```

12. Limit the Result

To limit the result in MongoDB, we use the limit() method.The limit() method takes one parameter, a number defining how many documents to return.

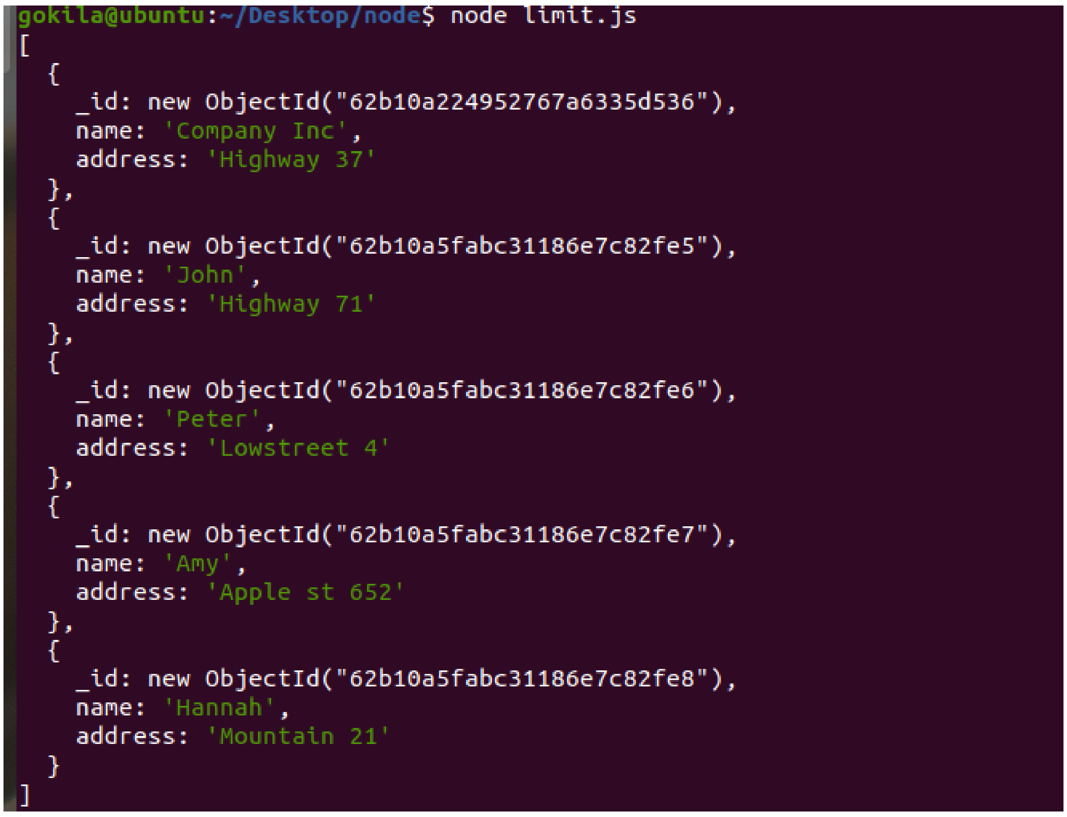
Limit the result to only return 5 documents:

Code:

```
var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/";


MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.collection("customer").find().limit(5).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Save the code above in a file called "limit.js" and run the file:

Snap:



```
gokila@ubuntu:~/Desktop/node$ node limit.js
[
  {
    _id: new ObjectId("62b10a224952767a6335d536"),
    name: 'Company Inc',
    address: 'Highway 37'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe5"),
    name: 'John',
    address: 'Highway 71'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe6"),
    name: 'Peter',
    address: 'Lowstreet 4'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe7"),
    name: 'Amy',
    address: 'Apple st 652'
  },
  {
    _id: new ObjectId("62b10a5fabc31186e7c82fe8"),
    name: 'Hannah',
    address: 'Mountain 21'
  }
]
```

13. Update Document

You can update a record, or document as it is called in MongoDB, by using the updateOne() method.The first parameter of the updateOne() method is a query object defining which document to update.

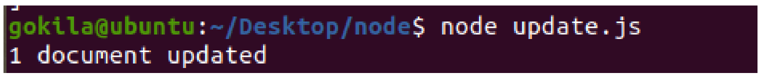
Update the document with the address "Valley 345" to name="Mickey" and address="Canyon 123":

Code:

```
var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://127.0.0.1:27017/";

MongoClient.connect(url, function(err, db) {

  if (err) throw err;

  var dbo = db.db("mydb");

  var myquery = { address: "Valley 345" };

  var newvalues = { $set: {name: "Mickey", address: "Canyon 123" } };

  dbo.collection("customer").updateOne(myquery, newvalues, function(err, res)
{

    if (err) throw err;

    console.log("1 document updated");

    db.close();

  });

});
```

Save the code above in a file called "update.js" and run the file:

Snap:



```
gokila@ubuntu:~/Desktop/node$ node update.js
1 document updated
```

14.Delete Document

To delete a record, or document as it is called in MongoDB, we use the deleteOne() method.The first parameter of the deleteOne() method is a query object defining which document to delete.

Delete the document with the address "Mountain 21":

Code:

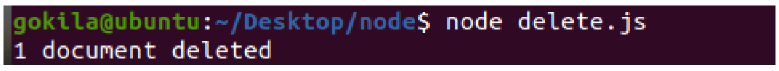
var MongoClient = require('mongodb').MongoClient;

Code:

```
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myquery = { address: 'Mountain 21' };
  dbo.collection("customer").deleteOne(myquery, function(err, obj) {
    if (err) throw err;
    console.log("1 document deleted");
    db.close();
  });
});
```

Save the code above in a file called "delete.js" and run the file:

Snap:



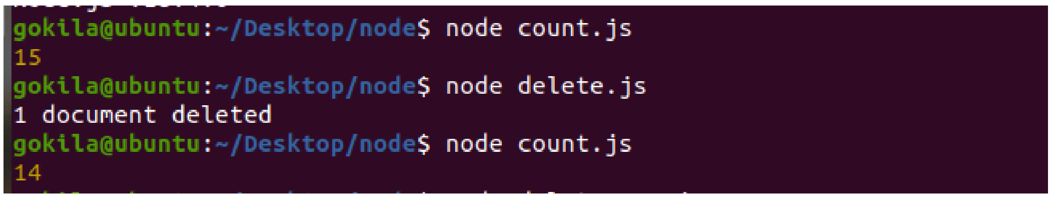
```
gokila@ubuntu:~/Desktop/node$ node delete.js
1 document deleted
```

15.Count:

Code:

```
const databasename = "mydb";
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url).then((client) => {
const connect = client.db(databasename);
const collection = connect.collection("customer");
collection.countDocuments().then((count_documents) => {
        console.log(count_documents);
        client.close();
}).catch((err) => {
        console.log(err.Message);
})
}).catch((err) => {
// Printing the error message
console.log(err.Message);
})
```

Snap:



```
gokila@ubuntu:~/Desktop/node$ node count.js
15
gokila@ubuntu:~/Desktop/node$ node delete.js
1 document deleted
gokila@ubuntu:~/Desktop/node$ node count.js
14
```

16.Drop Collection:

You can delete a table, or collection as it is called in MongoDB, by using the drop() method.The drop() method takes a callback function containing the error object and the result parameter which returns true if the collection was dropped successfully, otherwise it returns false.

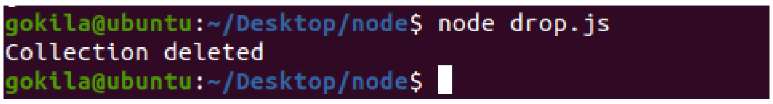
Delete the "customers" table:

Code:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";


MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.collection("customer").drop(function(err, delOK) {
    if (err) throw err;
    if (delOK) console.log("Collection deleted");
    db.close();
  });
});
```

Save the code above in a file called "drop.js" and run the file:

Snap:



```
gokila@ubuntu:~/Desktop/node$ node drop.js
Collection deleted
gokila@ubuntu:~/Desktop/node$ 
```