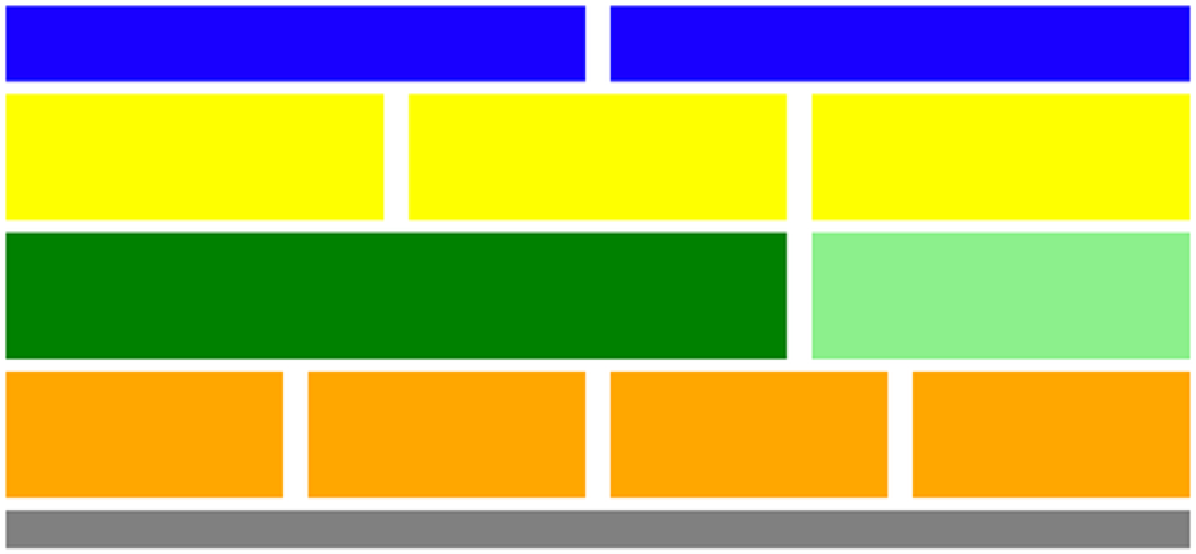
Neha Nariya **MCA** **Roll No. : 31**

Tutorial-01

1. Design html page to get below given output using bootstrap css.

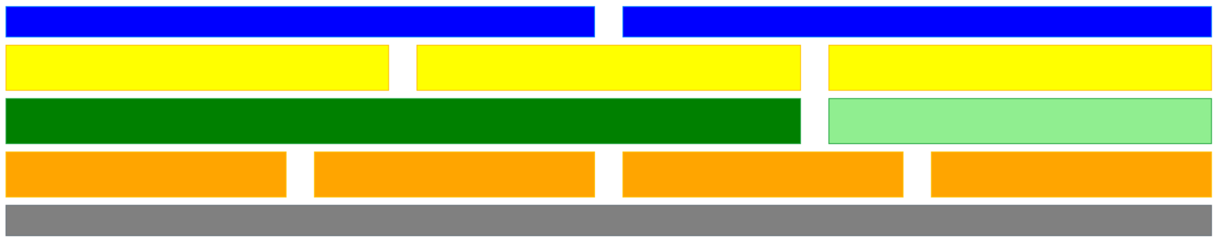


Answer:

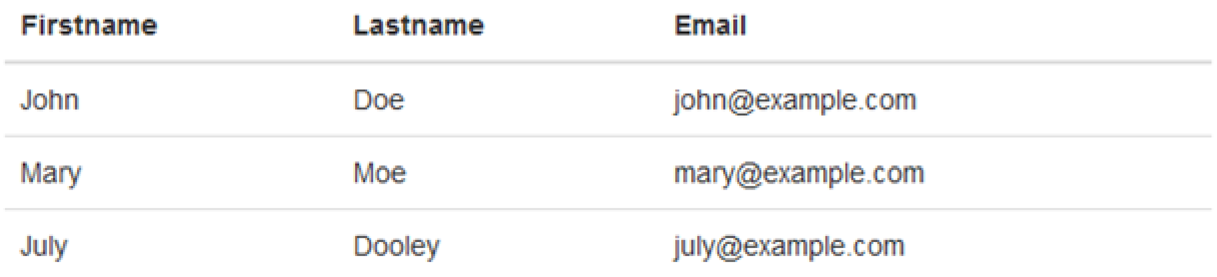
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Tutorial-01</title>

   <!--CSS Link-->
   <link href="css/bootstrap.min.css" rel="stylesheet">

   <!--JS Link-->
   <script src="js/bootstrap.min.js"></script>
   <script src="js/bootstrap.bundle.js"></script>

</head>
<body>
   <div class="container-fuied px-4 mt-4">

      <div class="row ">
         <div class="col-md-6 gx-3">
            <div class="p-3 border border-primary" style="background-color:blue;"></div>
         </div>
         <div class="col-md-6">
            <div class="p-3 border border-primary" style="background-color:blue;"></div>
         </div>
```

```
        </div>

        <div class="row gx-3 mt-2">
            <div class="col-md-4">
                <div class="p-4 border border-warning" style="background-color:yellow;"></div>
            </div>
            <div class="col-md-4">
                <div class="p-4 border border-warning" style="background-color:yellow;"></div>
            </div>
            <div class="col-md-4">
                <div class="p-4 border border-warning" style="background-color:yellow;"></div>
            </div>
        </div>

        <div class="row gx-3 mt-2">
            <div class="col-md-8">
                <div class="p-4 border border-success" style="background-color:green;"></div>
            </div>
            <div class="col-md-4">
                <div class="p-4 border border-success" style="background-color:lightgreen;"></div>
            </div>
        </div>

        <div class="row gx-3 mt-2">
            <div class="col-md-3">
                <div class="p-4 border border-warning" style="background-color:orange;"></div>
            </div>
            <div class="col-md-3">
                <div class="p-4 border border-warning" style="background-color:orange;"></div>
            </div>
            <div class="col-md-3">
                <div class="p-4 border border-warning" style="background-color:orange;"></div>
            </div>
            <div class="col-md-3">
                <div class="p-4 border border-warning" style="background-color:orange;"></div>
            </div>
        </div>

        <div class="row gx-3 mt-2">
            <div class="col">
                <div class="p-3 border border-secondary" style="background-color:gray;"></div>
            </div>
        </div>

    </body>
    </html>
```

Output:



2. Use a Bootstrap class to style the table properly and get the following output (with padding and horizontal dividers).



Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

Add zebra-stripes to the table.

Add borders on all sides of the table and cells.
Enable a hover state on table rows.
Make the table more compact by cutting cell padding in half.
Use contextual classes to add the following:

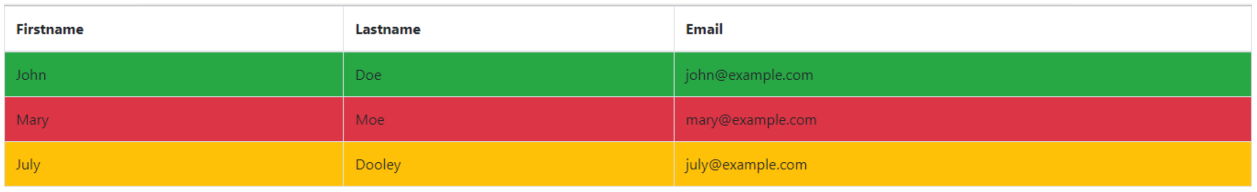
Green color to the table row containing John.
Red color to the table row containing Mary.
Orange color to the last table row.

Answer:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Tutorial-01</title>

   <!--CSS Link-->
   <link href="css/bootstrap.min.css" rel="stylesheet">
```

```
    <!--JS Link-->
    <script src="js/bootstrap.min.js"></script>
    <script src="js/bootstrap.bundle.js"></script>


</head>
<body>
  <!-- <div class="container-fluid mt-5">
    <div class="row justify-content-center">
        <div class="col-md-8">   -->
          <table class="table table-bordered table-striped table-hover table-condensed">

            <thead>
              <tr>
                <th scope="col">Firstname</th>
                <th scope="col">Lastname</th>
                <th scope="col">Email</th>
              </tr>
            </thead>

            <tbody>
              <tr class="bg-success">
                <td>John</td>
                <td>Doe</td>
                <td>john@example.com</td>
              </tr>
              <tr class="bg-danger">
                <td>Mary</td>
                <td>Moe</td>
                <td>mary@example.com</td>
              </tr>
              <tr class="bg-warning">
                <td>July</td>
                <td>Dooley</td>
                <td>july@example.com</td>
              </tr>
            </tbody>

          </table>

        <!-- </div>
      </div>
    </div>  -->
</body>
</html>
```

Output:



Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

3. Bootstrapping with Buttons.

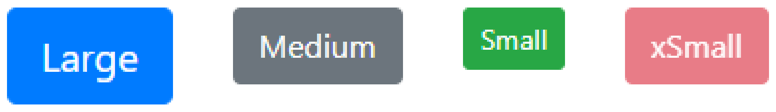
a. Use a Bootstrap class to style the button properly with a red color.
b. Change the size of the buttons in the following order: large, medium, small and xsmall.
c. Make the button span the entire width of the parent element.
d. Use a Bootstrap class to disable the button.

Answer:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tutorial-01</title>

  <!--CSS Link-->
  <link href="css/bootstrap.min.css" rel="stylesheet">

  <!--JS Link-->
  <script src="js/bootstrap.min.js"></script>
  <script src="js/bootstrap.bundle.js"></script>

</head>
<body>
  <!-- <div class="container">
    <button type="button" class="btn btn-primary btn-lg">Primary</button>

   <button type="button" class="btn btn-secondary btn-md">Secondary</button>

   <button type="button" class="btn btn-success btn-sm">Success</button>
```

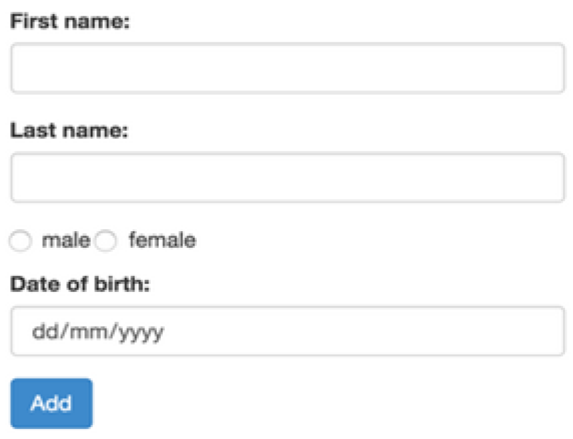
```
    <button type="button" class="btn btn-danger btn-xs">xsmall</button>

    <button type="button" class="btn btn-primary" disabled>Disabled</button>
      </div> -->
  <div class="container-fluid mt-5 d-flex">
    <div class="row">
      <div class="col">
        <button type="button" class="btn btn-primary btn-lg ">Large</button>
      </div>
      <div class="col">
        <button type="button" class="btn btn-secondary btn-md">Medium</button>
      </div>
      <div class="col">
        <button type="button" class="btn btn-success btn-sm">Small</button>
      </div>
      <div class="col">
        <button type="button" class="btn btn-danger btn-xs disabled">xSmall</button>
      </div>
    </div>
  </div>
  <!-- <div class="container btn-group d-flex" role="group">
    <button type="button" class="btn btn-danger">Button</button>
    <button type="button" class="btn btn-primary btn-lg ">Button Large </button>
    <button type="button" class="btn btn-success btn-lg">Button Large </button>
    <button type="button" class="btn btn-info btn-sm">Button Small </button>
    <button type="button" class="btn btn-warning btn-sm">Button Small </button>

    <button type="button" class="btn btn-danger disabled">Button Disabled</button>
  </div>  -->
</body>
</html>
```

Output:



4. Style the below given html form using bootstrap to get the output shown below.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Exercise #6: Simple form</title>
</head>
<body>
<form action="#">
    <div>
       <label for="first_name">First name:</label>
       <input type="text" name="first_name" id="first_name"/>
    </div>
    <div>
       <label for="last_name">Last name:</label>
       <input type="text" name="last_name" id="last_name"/>
    </div>
    <div>
       <label><input type="radio" name="gender" value="male"/>male</label>
 <label><input type="radio" name="gender" value="female"/>female</label>
    </div>
    <div>
       <label for="birth_date">Date of birth:</label>
       <input type="date" name="birth_date" id="birth_date"/>
    </div>
    <input type="submit" value="Add"/>
</form>
</body>
</html>
```



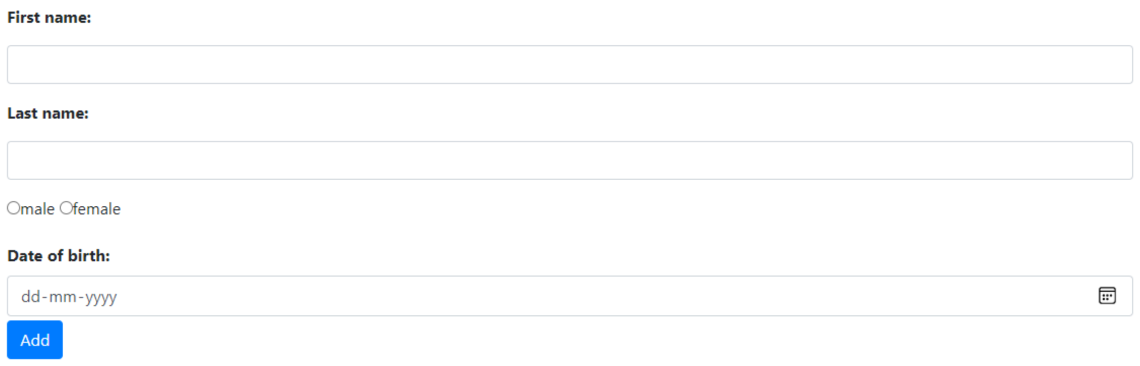
Answer:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exercise #6: Simple form</title>
  <!--CSS Link-->
  <link href="css/bootstrap.min.css" rel="stylesheet">

  <!--JS Link-->
  <!-- <script href="js/bootstrap.min.js"></script> -->
  <script src="js/bootstrap.min.js"></script>
  <script src="js/bootstrap.bundle.js"></script>
</head>
<body>
  <!-- <div class="container-fluid mt-5">
    <div class="row justify-content-center">
      <div class="col-md-4">

        <form action="#">

          <div class="row p-2">

            <label for="first_name" class="form-label">First name:</label>

            <input class="form-control" type="text" name="first_name" id="first_name" />

          </div>

          <div class="row p-2">

            <label for="last_name" class="form-label">Last name:</label>

            <input class="form-control" type="text" name="last_name" id="last_name" />

          </div>

          <div class="row p-2">

            <label class="form-label"> <input type="radio" class="form-check-input" name="gender"
    value="male" /> male</label>

            <label class="form-label"> <input type="radio" class="form-check-input" name="gender"
    value="female" /> female</label>

          </div>
```

```
              <div class="row p-2">

                <label for="birth_date" class="form-label">Date of birth:</label>

                <input type="date" class="form-control" name="birth_date" id="birth_date" />

              </div>
              <div class="row pt-2">
                <div class="col-md-3">
                  <input class="bg-primary text-white form-control" type="submit" value="Add" />
                </div>
              </div>

            </form>
          </div>
        </div>
      </div> -->
      <form action="#">
        <div class="container">
          <div class="form-group">
          <label for="first_name" class="form-group font-weight-bold">First name:</label>
          <input type="text" class=" form-control"  name="first_name" id="first_name"/>

        </div>

        <div class="form-group">

          <label for="last_name" class="form-group font-weight-bold">Last name:</label>
          <input type="text"  class="form-control"name="last_name" id="last_name"/>

        </div>

        <div >

          <label><input type="radio" name="gender" value="male" class="form-group"/>male</label>

        <label><input type="radio" name="gender" value="female" class="form-group"/>female</label>

        </div>

        <div>

          <label for="birth_date" class="form-text font-weight-bold">Date of birth:</label>

          <input type="date" name="birth_date" class="form-control " id="birth_date"/>

        </div>
        <div>
```

```
            <input type="submit" class="btn btn-primary form-text" value="Add"/>
        </div>

    </div>
</form>
</body>
</html>
```

Output:



Tutorial-02

(MongoDB)

Import restaurant.json file using below command.
mongoimport --db databasename --collection res --file D:\restaurants.json
Note: Don't write above command in mongo shell. Directly execute it from the command prompt.

1. **Write a MongoDB query to display all the documents in the collection restaurants.**

```
db.restaurants.find()
```

2. **Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.**

```
db.restaurants.find({},{restaurant_id:1,name:1,borough:1,cuisine:1})
```

3. **Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine, but exclude the field _id for all the documents in the collection restaurant.**

```
db.restaurants.find({},{restaurant_id:1,borough :1,cuisine:1,_id:0})
```

4. **Write a MongoDB query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant.**

```
db.restaurants.find({},{restaurant_id:1,borough:1,cuisine:1,_id:0,"address.zipcode":1)
```

5. **Write a MongoDB query to display all the restaurant which is in the borough Bronx.**

```
db.restaurants.find({"borough":"Bronx"}).pretty()
```

6. **Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.**

```
db.restaurants.find({"borough":"Bronx"}).limit(5).pretty()
```

7. **Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx**

```
db.restaurants.find({"borough":"Bronx"}).skip(5).limit(5).pretty()
```

8. **Write a MongoDB query to find the restaurants who achieved a score more than 90.**

```
db.restaurants.find({"grades.score":{$gt:90}})
```

9. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100

```
db.restaurants.find({"grades.score":{$gt:80,$lt:100}}).pretty()
```

10. Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168.

```
 db.restaurants.find({"address.coord" : {$lt : -95.754168}})
```

11. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.

```
db.restaurants.find({$and:
[
   {"cuisine":{$ne:"American"}},
   {"grades.score":{$gt:70}},
   {"address.coord":{$lt: -65.754168}}
]
}
)
```

12. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168.

Note : Do this query without using $and operator.

```
db.restaurants.find(
{
  "cuisine":{$ne:"American"},
  "grades.score":{$gt:70},
  "address.coord":{$lt:-65.754168}
})
```

13. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.

```
db.restaurants.find(

{

"cuisine":{$ne:"American"},

"grades.grade":"A",

"borough":{$ne:"Brooklyn"}

}).sort({"cuisine":-1})
```

14. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.

```
db.restaurants.find({name:/^Wil/},{"restaurantId":1,"name":1,"borough":1,"cuisine":1})
```

15. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

```
db.restaurants.find({name:/ces$/},{"restaurant Id":1,"name":1,"borough":1,"cuisine":1})
```

16. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.

```
 db.restaurants.find({name:/.*Reg.*/},{"restaurant Id":1,"name":1,"borough":1,"cuisine":1})
```

17. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish

```
db.restaurants.find({"borough":"Bronx",$or:[{"cuisine":"American"},{"cuisine":"Chinese"}]})
```

18. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronxor Brooklyn.

```
db.restaurants.find(
{"borough" :{$in :["Staten Island","Queens","Bronx","Brooklyn"]}},

{

"restaurant_id" : 1,

"name":1,"borough":1,

"cuisine" :1

}

);
```

19. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronxor Brooklyn.

```
db.restaurants.find(

{"borough" :{$nin :["Staten Island","Queens","Bronx","Brooklyn"]}},

{

"restaurant_id" : 1,

"name":1,"borough":1,

"cuisine" :1

}

);
```

20. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.

```
db.restaurants.find(

{"grades.score" :

{ $not:

{$gt : 10}

}
},
{
"restaurant_id" : 1,
"name":1,"borough":1,
"cuisine" :1
}
);
```

21. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns

```
db.restaurants.find(
{$or: [
  {name: /^Wil/},
  {"$and": [
      {"cuisine" : {$ne :"American "}},
      {"cuisine" : {$ne :"Chinees"}}
   ]}
]}
,{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
);
```

22. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

```
db.restaurants.find(
              {
               "grades.date": ISODate("2014-08-11T00:00:00Z"),
               "grades.grade":"A" ,
               "grades.score" : 11
              },
    {"restaurant_id" : 1,"name":1,"grades":1}
  );
```

23. Write a MongoDB query to arrange the name of the cuisine in ascending order and for that same cuisine borough should be in descending order

```
db.restaurants.find(
            { "grades.1.date": ISODate("2014-08-11T00:00:00Z"),
              "grades.1.grade":"A" ,
              "grades.1.score" : 9
            },
             {"restaurant_id" : 1,"name":1,"grades":1}
          );
```

24. Find out how many times each cuisine is offered at various restaurants.

```
db.restaurants.find(
            {
              "address.coord.1": {$gt : 42, $lte : 52}
            },
              {"restaurant_id" : 1,"name":1,"address":1,"coord":1}
          );
```

25. Find out how many times each cuisine is offered at various restaurants in descending order.

```
db.restaurants.find().sort({"name":1});
```

26. Which cuisine is highly offered among all restaurants?.

```
db.restaurants.find().sort(
              {"name":-1}
              );
```

27. Find out the top 5 highly offered cuisines among all restaurants?

```
db.restaurants.find().sort(
                         {"cuisine":1,"borough" : -1,}
                        );
```

Tutorial-03

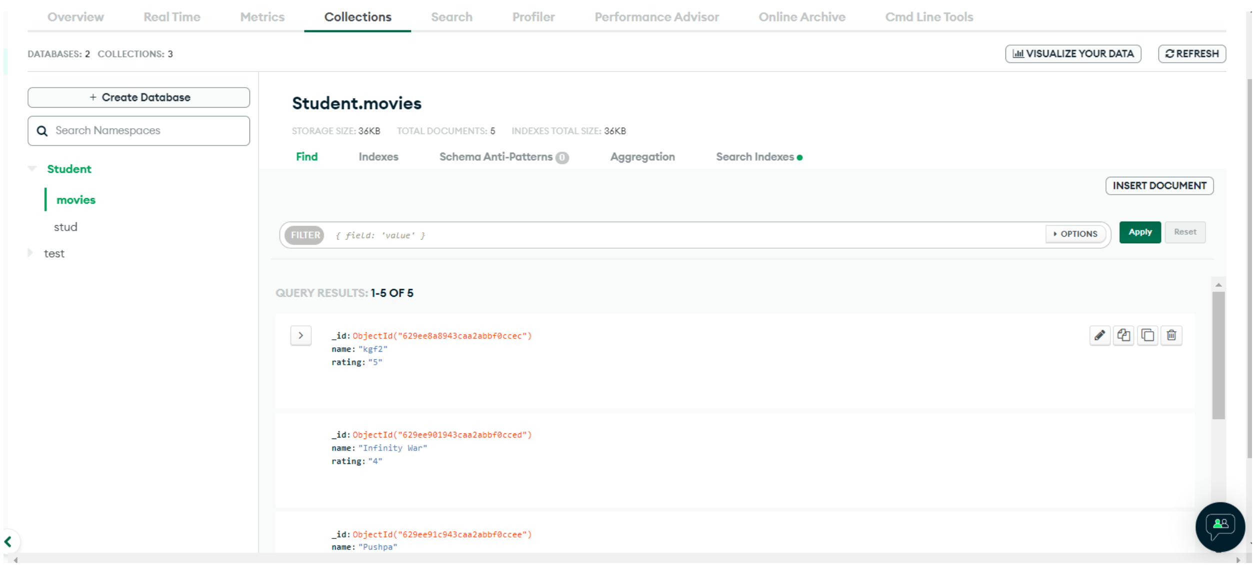
CRUD with Nodejs

1. Create and Emit a custom event that checks whether the age of the person is greater than 18 or not depending on the date of birth passed to the event.
2. Create a node script that gets the parameters using the GET method from the form.html file and log it on the console.
3. Create a node script that gets the parameters using POST method from the form.html file and log it on console.
4. Create mongodb for students and nodejs that connects to mongodb using mongojs module.(install monojs and nodemon packages).
[Student document must contain: s_id,s_name, s_branch, s_city, s_mobilenos, s_add]
5. Modify the above created script to insert static data in student db.
6. Create a nodejs that fetches all the documents from student db and logs it on the console.
7. Modify the program 6, in order to save records from the form.html file.
8. Create a nodejs script to update student records based on the student id.
9. Create a noejs script to delete student records based on student id.

```
var event = require('events');

var em = new event.EventEmitter();

em.on("myEvent",(data)=>{
   console.log(data);
})

em.on("checkAge",(age)=>{
   if(age>18)
   {
      console.log("eligible");
   }
   else
   {
      console.log("Not eligible")
   }

})
em.emit('checkAge',22)

em.emit('myEvent',"This is my data");
```

```
constexpress = require('express')
varrouter = express.Router();
varMovie = require('./Model/movies')
```

```
router.get('/movies',async(req,res)=>{
    constiMovie = awaitMovie.find();
    res.send(iMovie);
})

// for posting data
router.post('/movies',async(req,res) => {
    constiMovie = newMovie({
        name:req.body.name,
        rating:req.body.rating
    })

    console.log(iMovie);

    awaitiMovie.save((err,msg)=>{
        if(err){
            res.send(err);
        }
        else{
            res.send(msg);
        }
    });
})
module.exports = router;
```



DATABASES: 2 COLLECTIONS: 3

Student
movies
stud
test

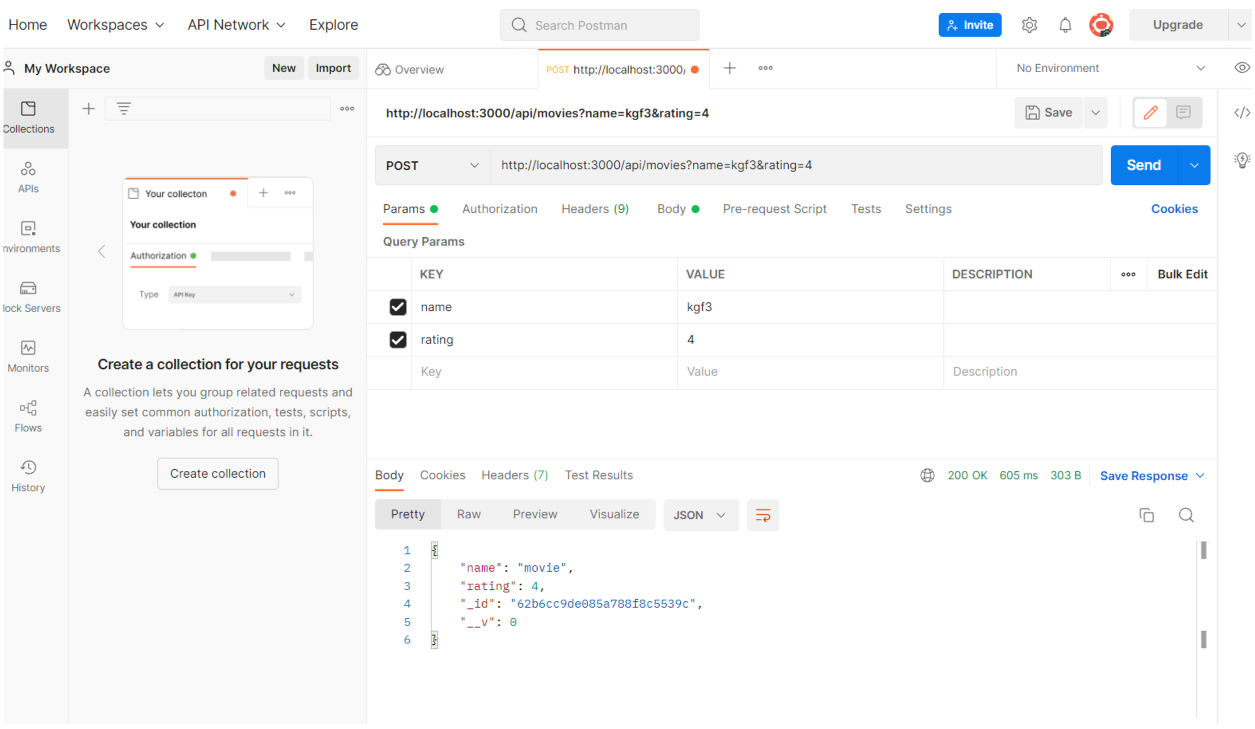
Student.movies

STORAGE SIZE: 36KB TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 36KB

{ field: 'value' }

QUERY RESULTS: 1-5 OF 5

```
_id: ObjectId("629ee8a8943caa2abbf0ccec")
name: "kgf2"
rating: "5"

_id: ObjectId("629ee901943caa2abbf0cced")
name: "Infinity War"
rating: "4"

_id: ObjectId("629ee91c943caa2abbf0ccee")
name: "Pushpa"
```



http://localhost:3000/api/movies?name=kgf3&rating=4

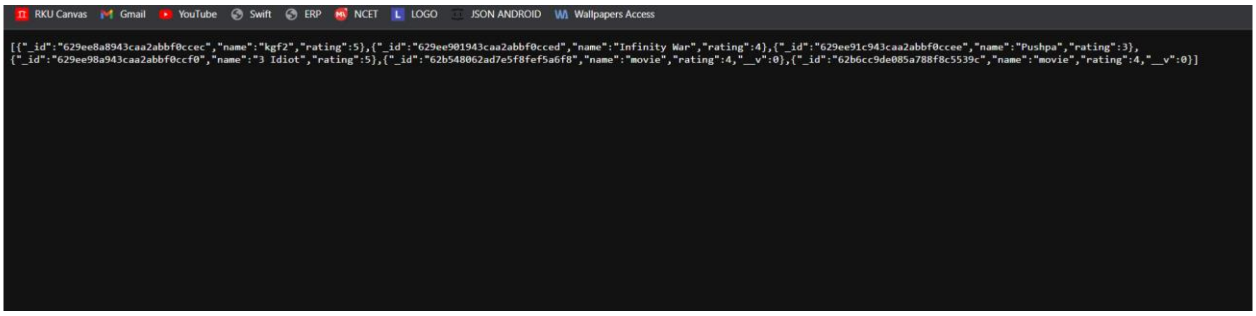
POST http://localhost:3000/api/movies?name=kgf3&rating=4

Query Params

KEY	VALUE	DESCRIPTION
✓ name	kgf3	
✓ rating	4	
Key	Value	Description

Body Cookies Headers (7) Test Results — 200 OK 605 ms 303 B

```
{
    "name": "movie",
    "rating": 4,
    "_id": "62b6cc9de085a788f8c5539c",
    "__v": 0
}
```



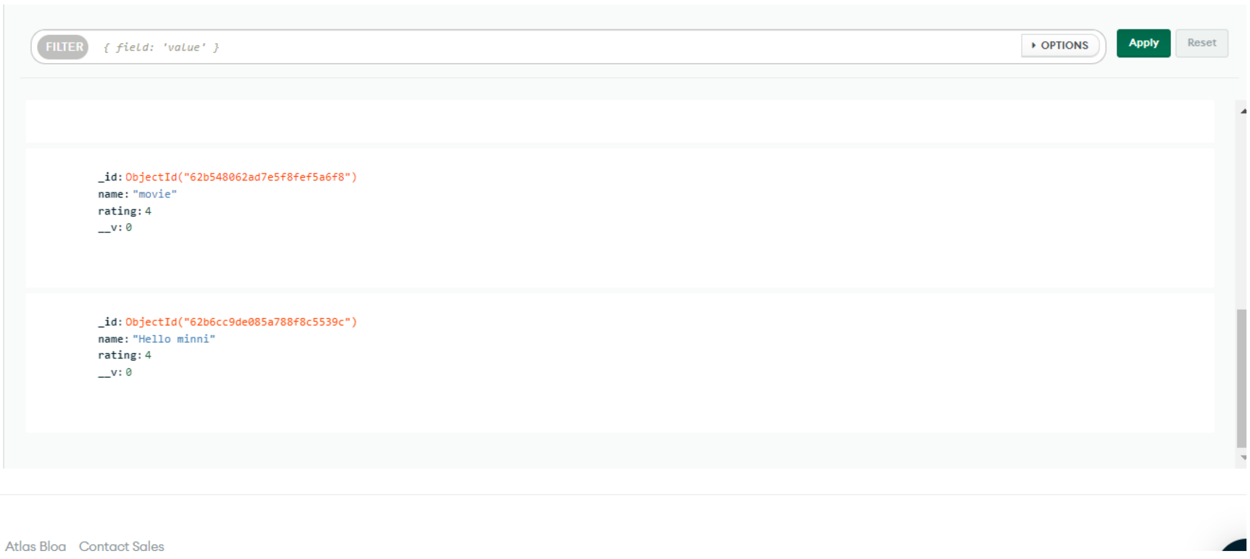
```
[{"_id":"629ee8a8943caa2abbf0ccec","name":"kgf2","rating":5},{"_id":"629ee901943caa2abbf0cced","name":"Infinity War","rating":4},{"_id":"629ee91c943caa2abbf0ccee","name":"Pushpa","rating":3},
{"_id":"629ee98a943caa2abbf0ccf0","name":"3 Idiot","rating":5},{"_id":"62b548062ad7e5f8fef5a6f8","name":"movie","rating":4,"__v":0},{"_id":"62b6cc9de085a788f8c5539c","name":"movie","rating":4,"__v":0}]
```

Tutorial-04

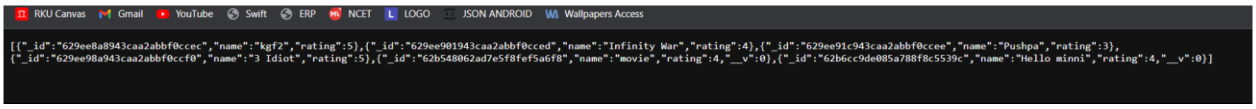
Creating APIS using Mongoose

1. Create a NodeAPI to insert records into products database using mongoose library.
2. Create a NodeAPI to get products from the database using mongoose library.
3. Create a NodeAPI to update products from the database using mongoose library.
4. Create a NodeAPI to delete products from the database using mongoose library.

```
varexpress = require('express')
varrouter = express.Router();
varMovie = require('./Model/movie')

router.get("/movies",async(req,res)=>{
    constiMovie = await
    Movie.find();
    res.send(iMovie);
});

//Update Specific Record
router.patch("/movies/:id",async(req,res)=>{
    constiMovie = awaitMovie.findOne({_id:req.params.id});
    iMovie.name = req.body.name;
    iMovie.rating = req.body.rating;

    awaitiMovie.save((err,msg)=>{
        if(err){
            res.status(500).json({
                "err":err
            })
        }else{
            res.status(200).json({
                "message":msg
            })
        }
    })
})

router.delete("/movies/:id",async(req,res)=>{
    awaitMovie.deleteOne({_id:req.params.id},(err,msg)=>{
        if(err){
            res.status(500).json({
                "err":err
            })
        }else{
            res.status(200).json({
                "message":msg
            })
        }
    })
})

module.exports = router
```



```
_id: ObjectId("62b548062ad7e5f8fef5a6f8")
name: "movie"
rating: 4
__v: 0

_id: ObjectId("62b6cc9de085a788f8c5539c")
name: "Hello minni"
rating: 4
__v: 0
```



[{"_id":"629ee8a8943caa2abbf0ccec","name":"kgf2","rating":5},{"_id":"629ee901943caa2abbf0cced","name":"Infinity War","rating":4},{"_id":"629ee91c943caa2abbf0ccee","name":"Pushpa","rating":3},
{"_id":"629ee98a943caa2abbf0ccf0","name":"3 Idiot","rating":5},{"_id":"62b548062ad7e5f8fef5a6f8","name":"movie","rating":4,"__v":0},{"_id":"62b6cc9de085a788f8c5539c","name":"Hello minni","rating":4,"__v":0}]

Tutorial-05

Hosting CRUD API on Heroku with Mongo Cluster database

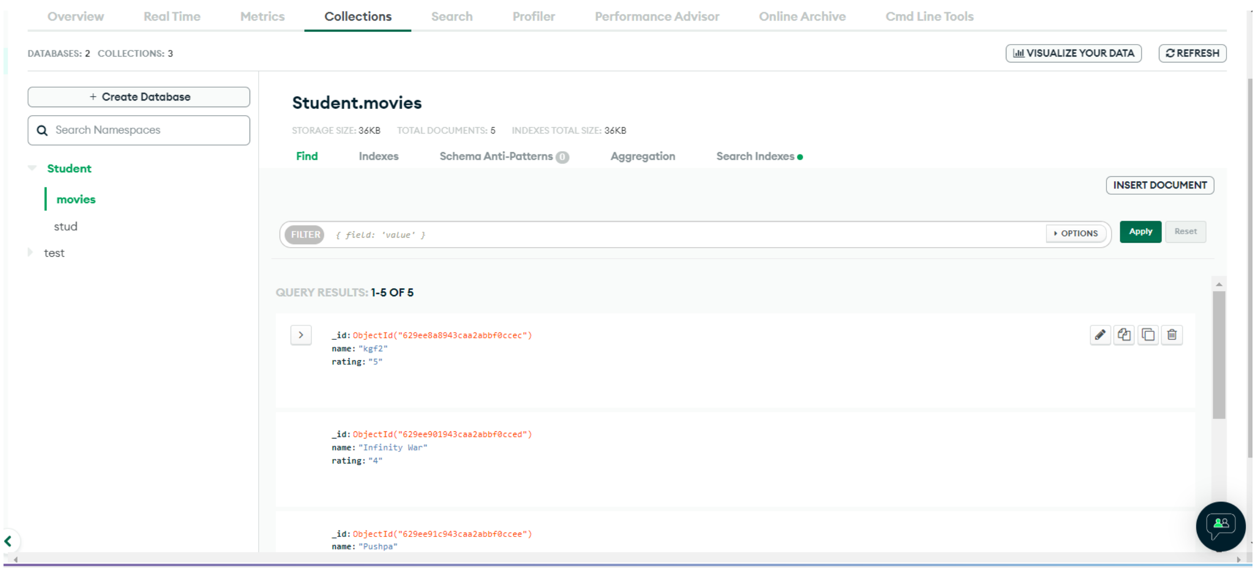
Note: You need to submit live working url as a part of assignment submission

1. Create a database of your choice with at least 2 collections on the mongo cluster.

   Note: Collections should be different from your classmates.

2. Create a NodeAPI to perform CRUD operations on the above created collections for the data stored on mlab cluster.
3. Host the same project on Heroku and need to present the same.

```
varexpress = require('express')
varrouter = express.Router();
varMovie = require('./Model/movie')

router.get("/movies",async(req,res)=>{
    constiMovie = await
    Movie.find();
    res.send(iMovie);
});

//Update Specific Record
router.patch("/movies/:id",async(req,res)=>{
    constiMovie = awaitMovie.findOne({_id:req.params.id});
    iMovie.name = req.body.name;
    iMovie.rating = req.body.rating;

    awaitiMovie.save((err,msg)=>{
        if(err){
            res.status(500).json({
                "err":err
            })
        }else{
            res.status(200).json({
                "message":msg
            })
        }
    })
})

router.delete("/movies/:id",async(req,res)=>{
    awaitMovie.deleteOne({_id:req.params.id},(err,msg)=>{
        if(err){
            res.status(500).json({
                "err":err
            })
        }else{
            res.status(200).json({
                "message":msg
            })
        }
    })
})
```



DATABASES: 2 COLLECTIONS: 3

Student
- movies
- stud

test

Student.movies

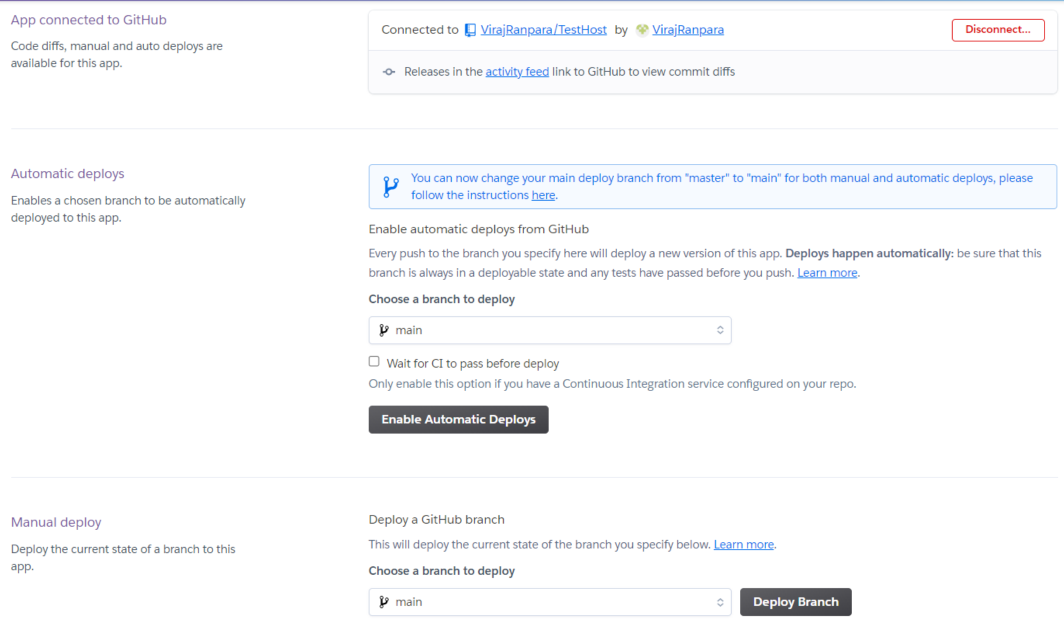
STORAGE SIZE: 36KB TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 36KB

Find | Indexes | Schema Anti-Patterns 0 | Aggregation | Search Indexes ●

`{ field: 'value' }`

QUERY RESULTS: 1-5 OF 5

```
_id: ObjectId("629ee8a8943caa2abbf0ccec")
name: "kgf2"
rating: "5"

_id: ObjectId("629ee901943caa2abbf0cced")
name: "Infinity War"
rating: "4"

_id: ObjectId("629ee91c943caa2abbf0ccee")
name: "Pushpa"
```



App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to VirajRanpara/TestHost by VirajRanpara

Releases in the activity feed link to GitHub to view commit diffs

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. Learn more.

Choose a branch to deploy

main

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Manual deploy

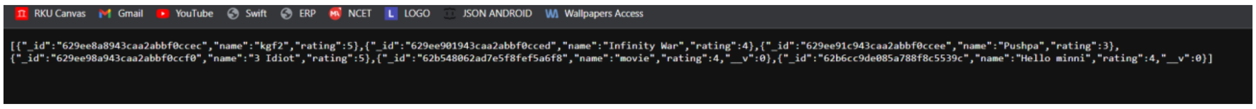
Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy

main



```
[{"_id":"629ee8a8943caa2abbf0ccec","name":"kgf2","rating":5},{"_id":"629ee901943caa2abbf0cced","name":"Infinity War","rating":4},{"_id":"629ee91c943caa2abbf0ccee","name":"Pushpa","rating":3},
{"_id":"629ee98a943caa2abbf0ccf0","name":"3 Idiot","rating":5},{"_id":"62b548062ad7e5f8fef5a6f8","name":"movie","rating":4,"__v":0},{"_id":"62b6cc9de085a788f8c5539c","name":"Hello minni","rating":4,"__v":0}]
```

Tutorial-07

Create a component that displays data the from the live API created using node(refer prev tutorials).

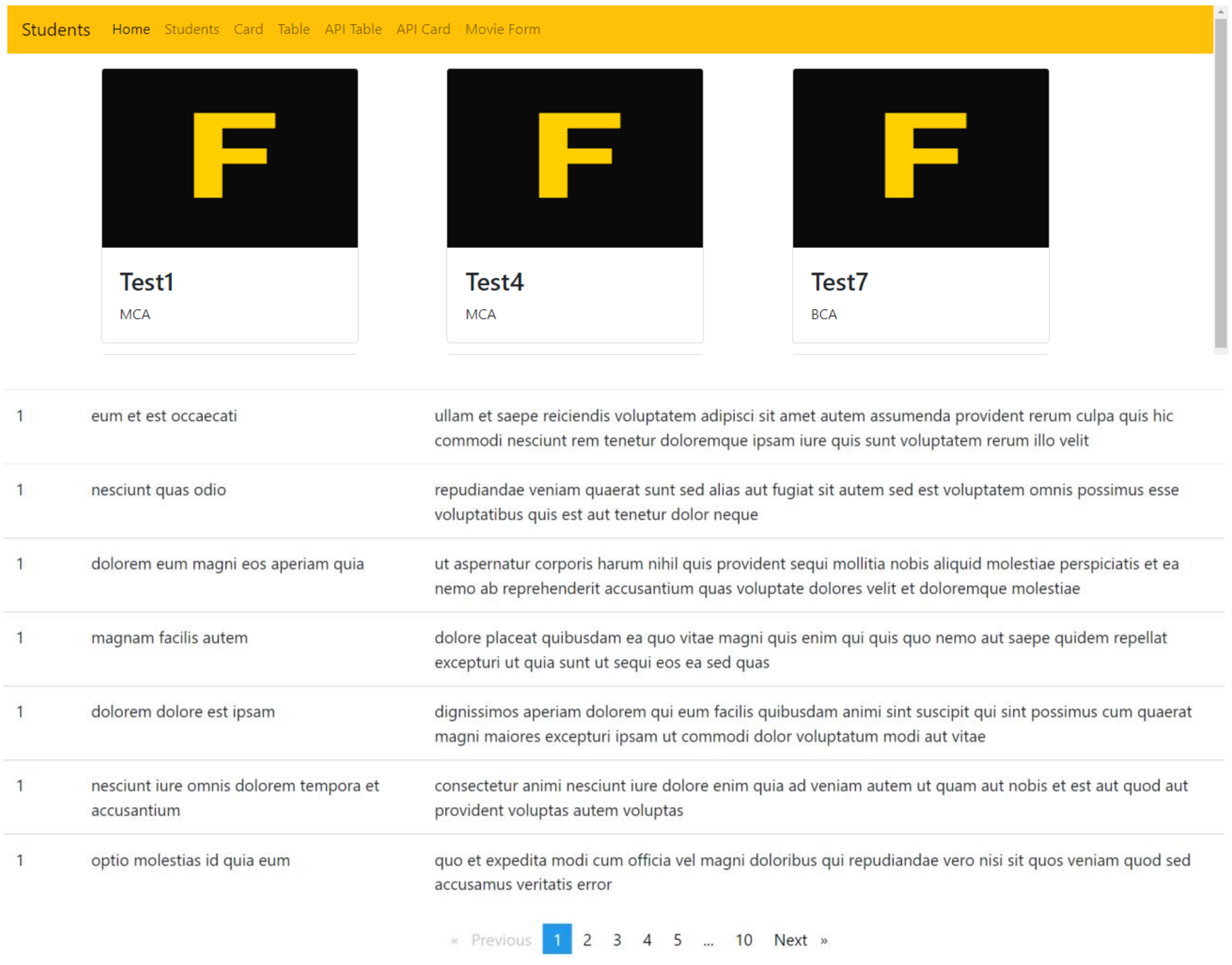
Make sure to use following things:

- ngFor
- bootstrap (any)
- Pagination
- Sorting
- Searching
- Service
- Interface

```
<form#login="ngForm"(ngSubmit)="submit(login)">
   <divclass="form-group">
     <labelfor="exampleInputEmail1">Email address</label>
     <inputtype="email"name="Email"#Email="ngModel"[(ngModel)]="email"class="form-
control"id="exampleInputEmail1"aria-describedby="emailHelp"placeholder="Enter email">
   </div>
   <divclass="form-group">
     <labelfor="exampleInputPassword1">Password</label>
     <inputtype="password"name="Pswd"#Pswd="ngModel"[(ngModel)]="pswd"class="form-
control"id="exampleInputPassword1"placeholder="Password">
   </div>
   <buttontype="submit"class="btn btn-primary">Submit</button>
 </form>
```

```
<tableclass="table">
   <thead>
       <th(click)="sort('userId')">UserID</th>
       <th(click)="sort('title')">Title</th>
       <th(click)="sort('body')">Body</th>
   </thead>
   <tbody>
       <tr*ngFor = "let p of posts | paginate: { itemsPerPage: 10, currentPage: p } | orderBy:Key">
           <td>{{p.userId}}</td>
           <td>{{p.title}}</td>
           <td>{{p.body}}</td>
       </tr>
   </tbody>
</table>
<pagination-controlsclass="text-center"(pageChange)="p = $event"></pagination-controls>
```

```
<divclass="container">
  <h1>Photo Details..</h1>
  <divclass="row">
    <divclass="card-columns">
      <divclass="card"*ngFor="let ph of photo | paginate: { itemsPerPage: 6, currentPage: p }
">
        <divclass="card-body text-center">
          <h5class="card-title">{{ph.title}}</h5>
          <imgclass="m-3"[src]="ph.url">
          <!-- <p class="card-text">Thumbnail Url: {{p.thumbnailUrl}}</p> -->
        </div>
      </div>
    </div>
  </div>
</div>
<pagination-controlsclass="text-center"(pageChange)="p = $event"></pagination-controls>
```



Test1
MCA

Test4
MCA

Test7
BCA

1	eum et est occaecati	ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit
1	nesciunt quas odio	repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque
1	dolorem eum magni eos aperiam quia	ut aspernatur corporis harum nihil quis provident sequi mollitia nobis aliquid molestiae perspiciatis et ea nemo ab reprehenderit accusantium quas voluptate dolores velit et doloremque molestiae
1	magnam facilis autem	dolore placeat quibusdam ea quo vitae magni quis enim qui quis quo nemo aut saepe quidem repellat excepturi ut quia sunt ut sequi eos ea sed quas
1	dolorem dolore est ipsam	dignissimos aperiam dolorem qui eum facilis quibusdam animi sint suscipit qui sint possimus cum quaerat magni maiores excepturi ipsam ut commodi dolor voluptatum modi aut vitae
1	nesciunt iure omnis dolorem tempora et accusantium	consectetur animi nesciunt iure dolore enim quia ad veniam autem ut quam aut nobis et est aut quod aut provident voluptas autem voluptas
1	optio molestias id quia eum	quo et expedita modi cum officia vel magni doloribus qui repudiandae vero nisi sit quos veniam quod sed accusamus veritatis error

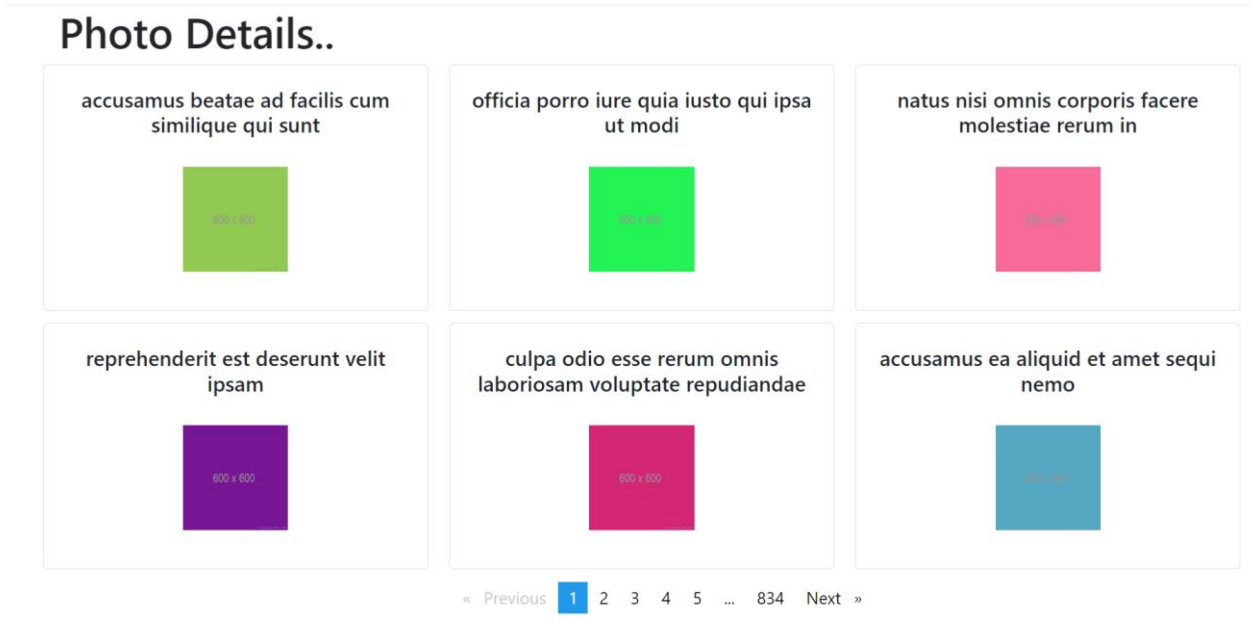


Photo Details..

accusamus beatae ad facilis cum similique qui sunt

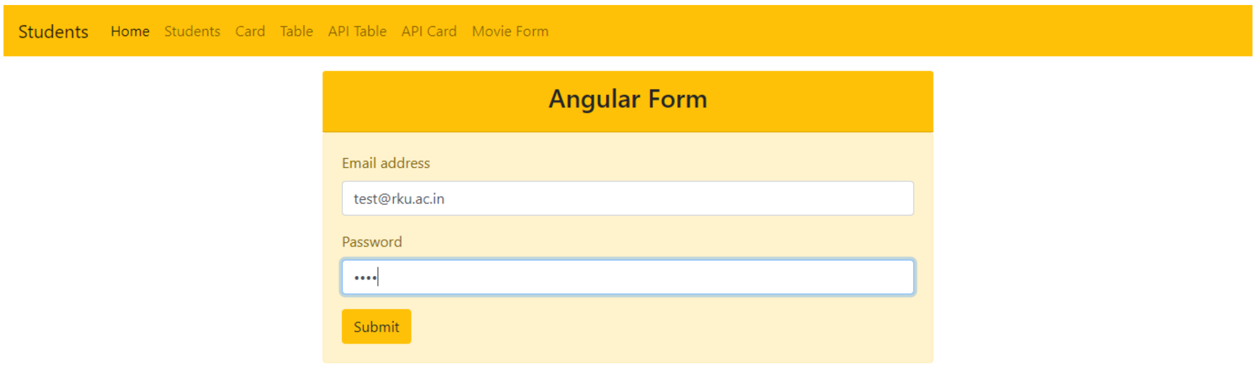
officia porro iure quia iusto qui ipsa ut modi

natus nisi omnis corporis facere molestiae rerum in

reprehenderit est deserunt velit ipsam

culpa odio esse rerum omnis laboriosam voluptate repudiandae

accusamus ea aliquid et amet sequi nemo



Students Home Students Card Table API Table API Card Movie Form

Angular Form

Email address: test@rku.ac.in

Password: ••••

Practice for Property Binding in Angular

1..apicard.component.html

```
<!-- <p>apicard works!</p> -->
<div class="row row-cols-1 row-cols-md-5 g-4">
    <div class="col-md-3" *ngFor="let dt of data | paginate: { itemsPerPage: 12,
currentPage: page } ">
        <div class="card mt-1" style="height: auto;">
            <img [src]="dt.url" height="100px" width="100px" class="card-img-top" alt="...">
            <div class="card-body">
                <p class="card-text"><span>ID:-</span>{{dt.id}}</p>
                <hr>
                <h5 class="card-title">{{dt.title}}</h5>
            </div>
        </div>
        <br>
    </div>
</div>
<div class="text-center">
    <pagination-controls (pageChange)="page = $event " class="text-
center"></pagination-controls>
</div>
```

2..apicard.component.ts

```
import { Component, OnInit } from '@angular/core';
import { PostService } from '../Service/post.service';


@Component({
  selector: 'app-apicard',
  templateUrl: './apicard.component.html',
  styleUrls: ['./apicard.component.css']
})
export class APICardComponent implements OnInit {
  page:number =1;

  constructor(private _postService:PostService) { }

  data:any;

  ngOnInit(): void {
    this._postService.getData().subscribe(d_data=>{
      this.data=d_data;
    })
  }

}
```

3..nav-bar.component.html

```
<nav class="navbar navbar-expand-lg navbar-light bg-primary">
    <a class="navbar-brand" href="#">21SOECA21032</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-
label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
      <div class="navbar-nav">
        <a class="nav-link active" routerLink="apicard" href="#">Home <span class="sr-
only">(current)</span></a>
        <a class="nav-link" routerLink="apicard" href="#">Student</a>
        <a class="nav-link" routerLink="apicard" href="#">Sign Up</a>
      </div>
    </div>
  </nav>
```

4..nav-bar.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-nav-bar',
  templateUrl: './nav-bar.component.html',
  styleUrls: ['./nav-bar.component.css']
})
export class NavBarComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

5..Service/Post

Post.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class PostService {

  url = "https://jsonplaceholder.typicode.com/photos"

  constructor(private _http:HttpClient) { }
  getData(){
    return this._http.get(this.url);
      }

}
```

6..app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { APICardComponent } from './apicard/apicard.component';
import { FormComponent } from './form/form.component';
import { NotFoundComponent } from './not-found/not-found.component';


const routes: Routes = [
  {path:"apicard",component:APICardComponent},
  {path:'form',component:FormComponent},
  {path:'',redirectTo:'apicard',pathMatch:'full'},
  {path:'**',component:NotFoundComponent},

];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

7..app.component.html

```
<app-nav-bar></app-nav-bar>
<!-- <app-apicard></app-apicard> -->
<router-outlet></router-outlet>
```

8..app.module.ts

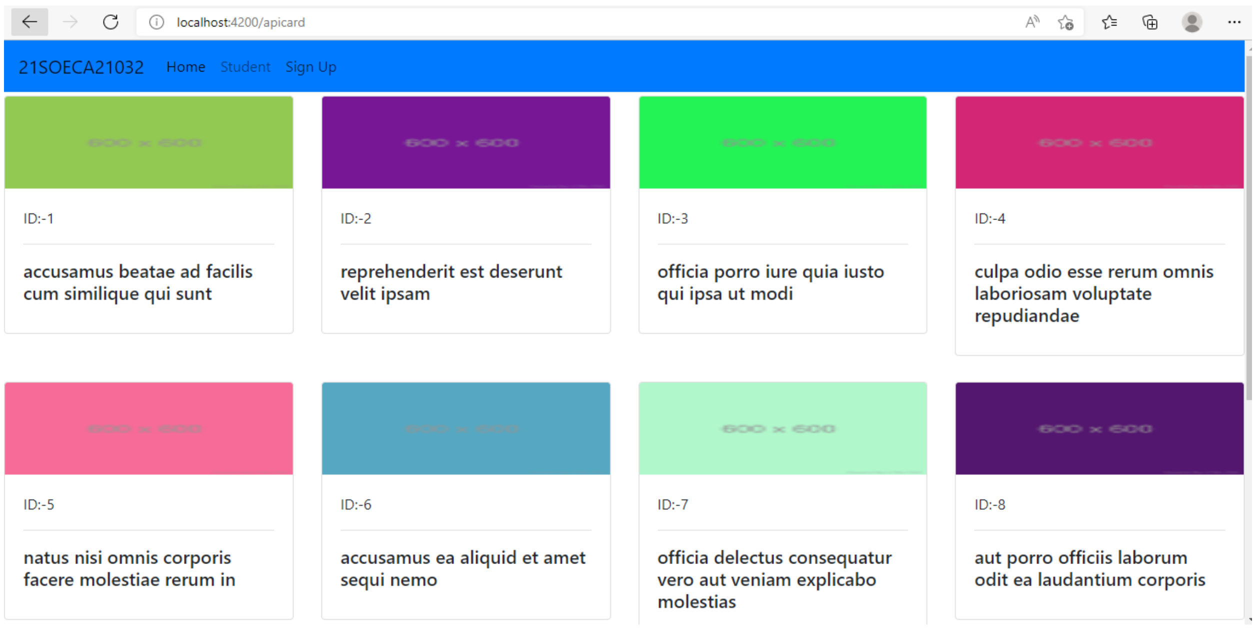
```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NavBarComponent } from './nav-bar/nav-bar.component';
import { APICardComponent } from './apicard/apicard.component';
import { NotFoundComponent } from './not-found/not-found.component';
import { HttpClientModule } from '@angular/common/http';
import {NgxPaginationModule} from 'ngx-pagination';
import { FormComponent } from './form/form.component';
import { FormsModule } from '@angular/forms'




@NgModule({
  declarations: [
    AppComponent,
    NavBarComponent,
    APICardComponent,
    NotFoundComponent,
    FormComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    NgxPaginationModule,
    FormsModule,



  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

9..index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Placeholder1</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css"
integrity="sha384-
zCbKRCUGaJDkqS1kPbPd7TveP5iyJE0EjAuZQTgFLD2ylzuqKfdKlfG/eSrtxUkn"
crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js"
integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
fQybjgWLrvvRgtW6bFlB7jaZrFsaBXjsOMm/tB9LTS58ONXgqbR9W8oWht/amnpF"
crossorigin="anonymous"></script>
</head>
<body>
```

```
  <app-root></app-root>
</body>
</html>
```

Output:



localhost:4200/apicard

21SOECA21032 Home Student Sign Up

ID:-1
accusamus beatae ad facilis cum similique qui sunt

ID:-2
reprehenderit est deserunt velit ipsam

ID:-3
officia porro iure quia iusto qui ipsa ut modi

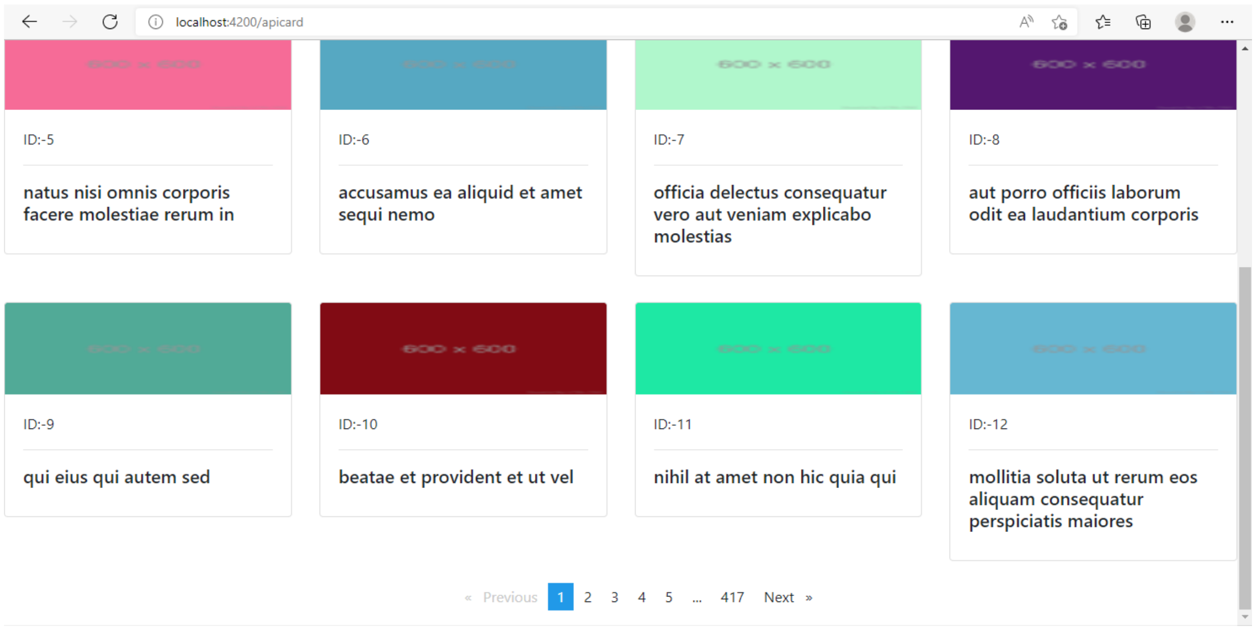
ID:-4
culpa odio esse rerum omnis laboriosam voluptate repudiandae

ID:-5
natus nisi omnis corporis facere molestiae rerum in

ID:-6
accusamus ea aliquid et amet sequi nemo

ID:-7
officia delectus consequatur vero aut veniam explicabo molestias

ID:-8
aut porro officiis laborum odit ea laudantium corporis



localhost:4200/apicard

ID:-5
natus nisi omnis corporis facere molestiae rerum in

ID:-6
accusamus ea aliquid et amet sequi nemo

ID:-7
officia delectus consequatur vero aut veniam explicabo molestias

ID:-8
aut porro officiis laborum odit ea laudantium corporis

ID:-9
qui eius qui autem sed

ID:-10
beatae et provident et ut vel

ID:-11
nihil at amet non hic quia qui

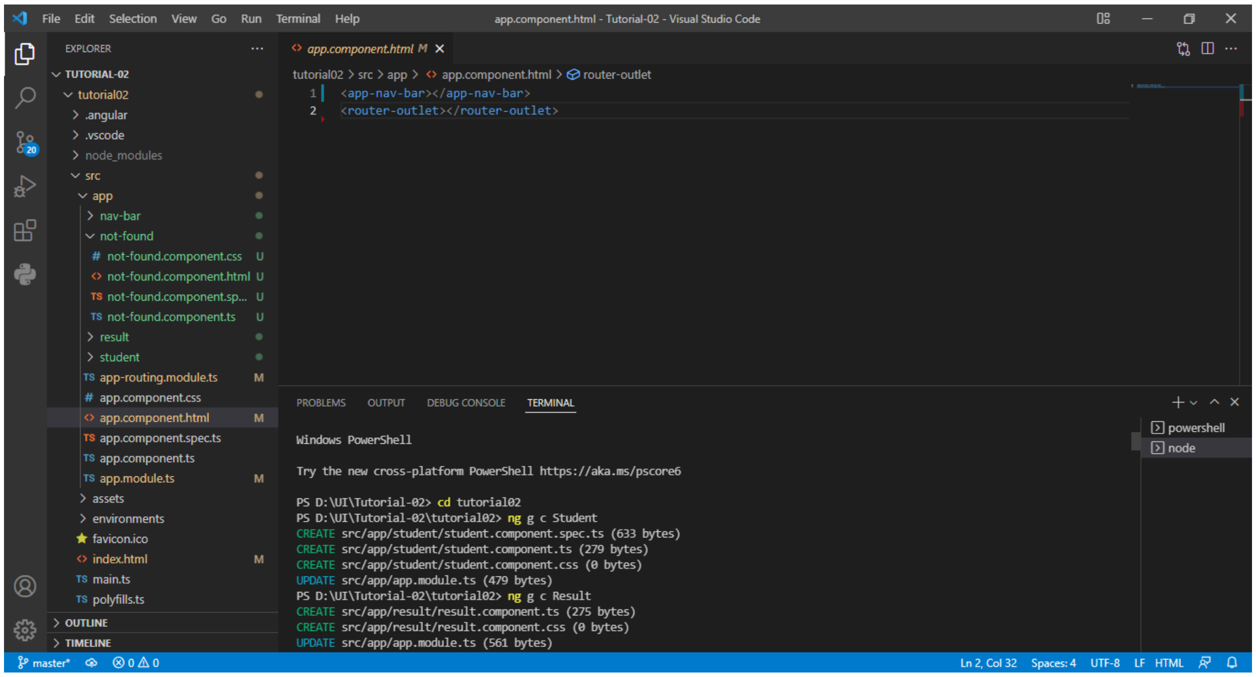
ID:-12
mollitia soluta ut rerum eos aliquam consequatur perspiciatis maiores

« Previous 1 2 3 4 5 ... 417 Next »

Practice with NavBar and Components(UI)

1. app.component.html

```
<app-nav-bar></app-nav-bar>
<router-outlet></router-outlet>
```



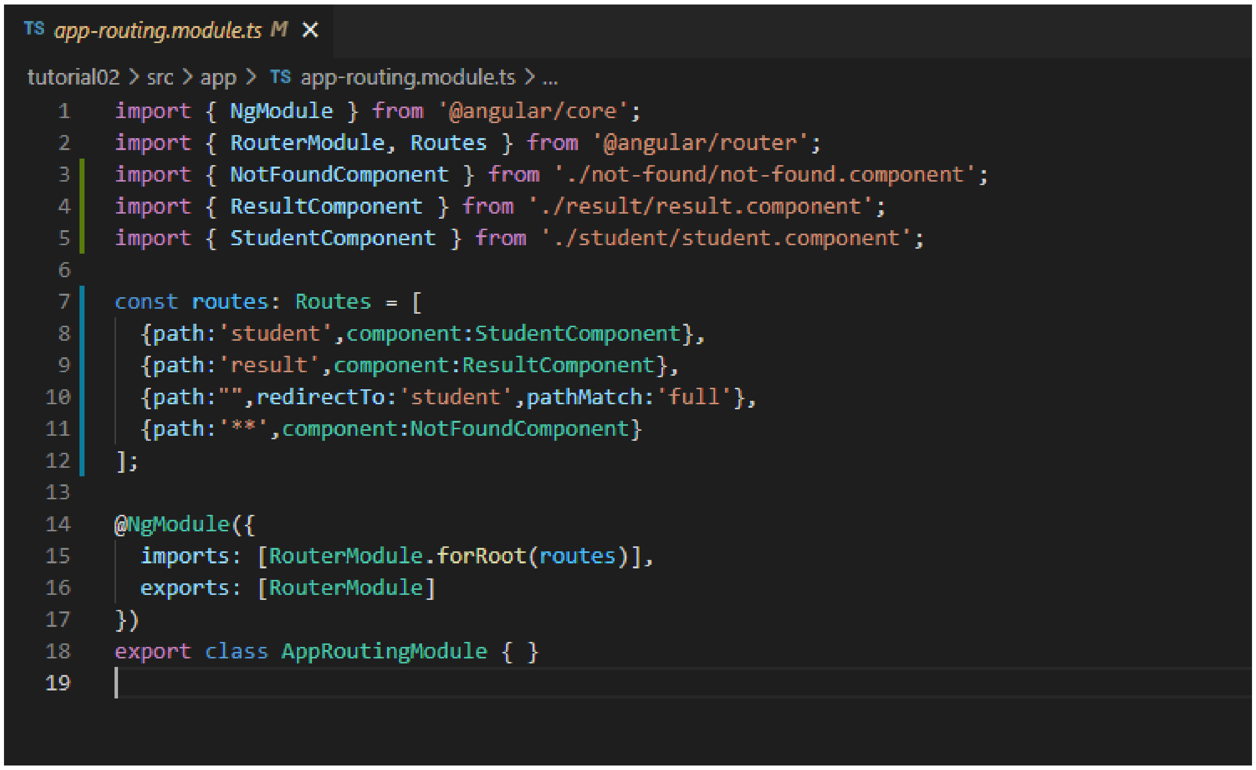
```
<app-nav-bar></app-nav-bar>
<router-outlet></router-outlet>
```

```
Windows PowerShell

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\UI\Tutorial-02> cd tutorial02
PS D:\UI\Tutorial-02\tutorial02> ng g c Student
CREATE src/app/student/student.component.spec.ts (633 bytes)
CREATE src/app/student/student.component.ts (279 bytes)
CREATE src/app/student/student.component.css (0 bytes)
UPDATE src/app/app.module.ts (479 bytes)
PS D:\UI\Tutorial-02\tutorial02> ng g c Result
CREATE src/app/result/result.component.ts (275 bytes)
CREATE src/app/result/result.component.css (0 bytes)
UPDATE src/app/app.module.ts (561 bytes)
```

2. app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { NotFoundComponent } from './not-found/not-found.component';
import { ResultComponent } from './result/result.component';
import { StudentComponent } from './student/student.component';

const routes: Routes = [
  {path:'student',component:StudentComponent},
  {path:'result',component:ResultComponent},
  {path:"",redirectTo:'student',pathMatch:'full'},
  {path:'**',component:NotFoundComponent}
];
```

```
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

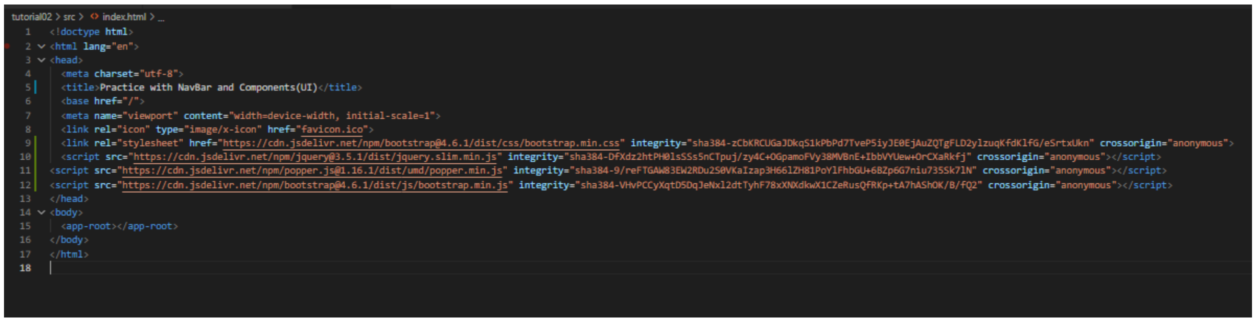


```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { NotFoundComponent } from './not-found/not-found.component';
import { ResultComponent } from './result/result.component';
import { StudentComponent } from './student/student.component';

const routes: Routes = [
  {path:'student',component:StudentComponent},
  {path:'result',component:ResultComponent},
  {path:"",redirectTo:'student',pathMatch:'full'},
  {path:'**',component:NotFoundComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

3. index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Practice with NavBar and Components(UI)</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css"
integrity="sha384-
zCbKRCUGaJDkqS1kPbPd7TveP5iyJE0EjAuZQTgFLD2ylzuqKfdKlfG/eSrtxUkn"
crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js"
integrity="sha384-
```

```
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.min.js"
integrity="sha384-
VHvPCCyXqtD5DqJeNxl2dtTyhF78xXNXdkwX1CZeRusQfRKp+tA7hAShOK/B/fQ2"
crossorigin="anonymous"></script>
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

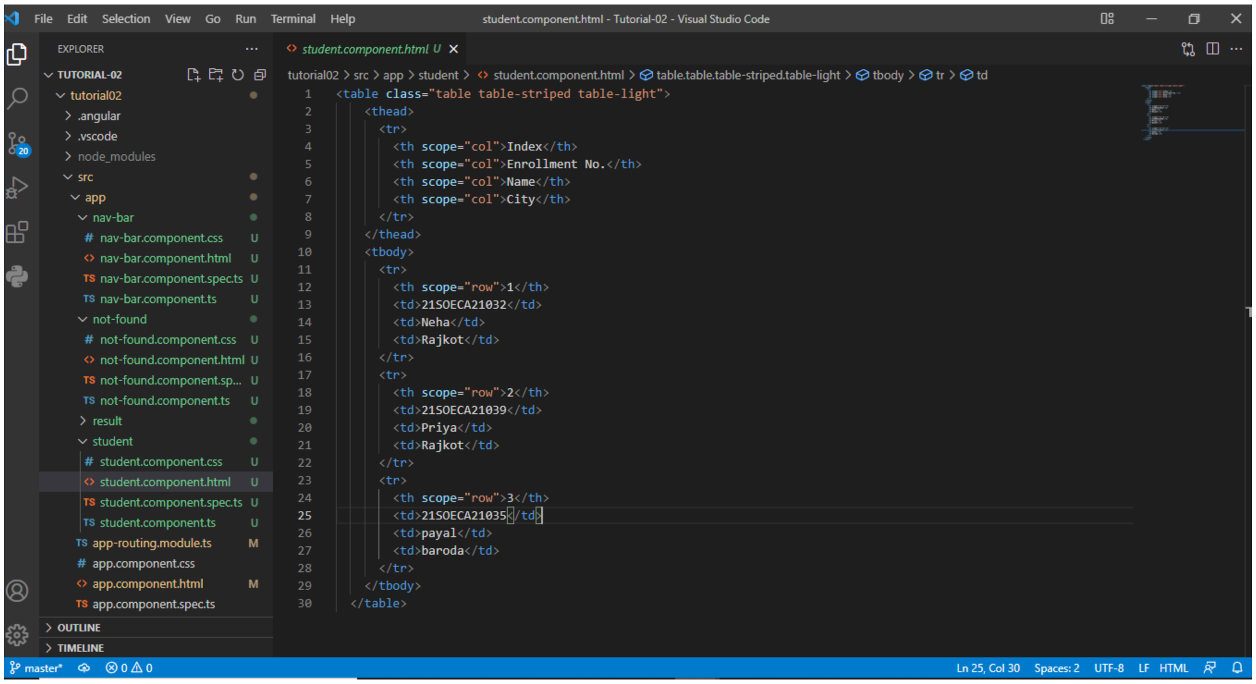


tutorial02 > src > index.html > ...

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Practice with NavBar and Components(UI)</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css" integrity="sha384-zCbKRCUGaJDkqS1kPbPd7TveP5iyJE0EjAuZQTgFLD2ylzuqKfdKlfG/eSrtxUkn" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.min.js" integrity="sha384-VHvPCCyXqtD5DqJeNxl2dtTyhF78xXNXdkwX1CZeRusQfRKp+tA7hAShOK/B/fQ2" crossorigin="anonymous"></script>
</head>
<body>
  <app-root></app-root>
</body>
</html>

```

4. student.component.html

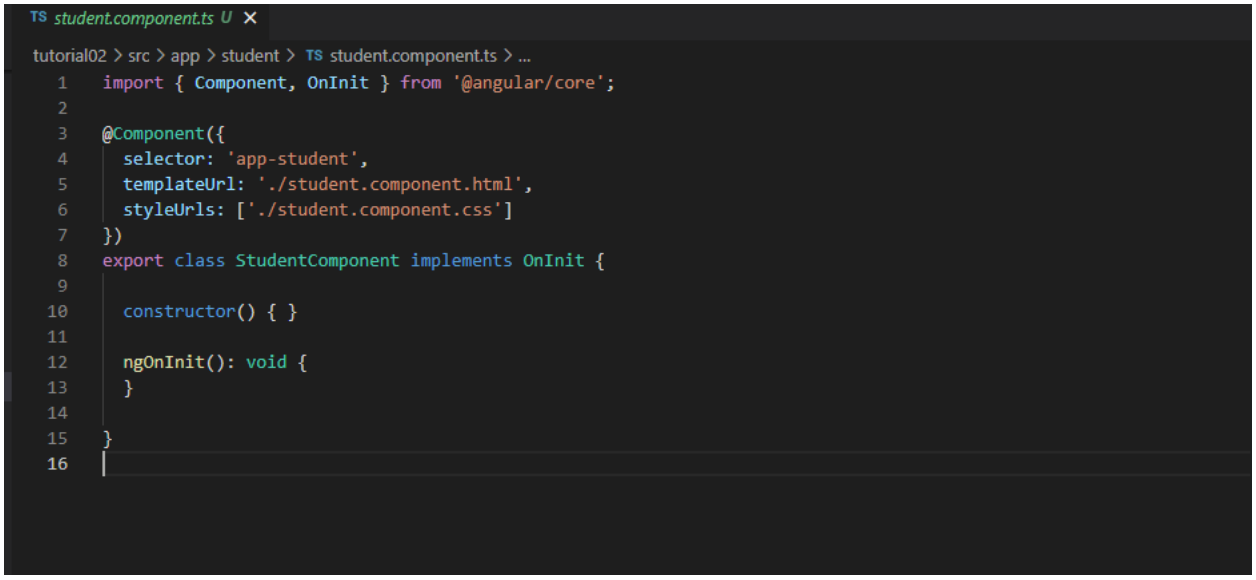
```
<table class="table table-striped table-light">
    <thead>
      <tr>
        <th scope="col">Index</th>
        <th scope="col">Enrollment No.</th>
        <th scope="col">Name</th>
        <th scope="col">City</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <th scope="row">1</th>
        <td>21SOECA21032</td>
        <td>Neha</td>
        <td>Rajkot</td>
      </tr>
      <tr>
        <th scope="row">2</th>
        <td>21SOECA21039</td>
        <td>Priya</td>
        <td>Rajkot</td>
      </tr>
      <tr>
        <th scope="row">3</th>
        <td>21SOECA21035</td>
        <td>payal</td>
        <td>baroda</td>
      </tr>
    </tbody>
  </table>
```



```
<table class="table table-striped table-light">
  <thead>
    <tr>
      <th scope="col">Index</th>
      <th scope="col">Enrollment No.</th>
      <th scope="col">Name</th>
      <th scope="col">City</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
      <td>21SOECA21032</td>
      <td>Neha</td>
      <td>Rajkot</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>21SOECA21039</td>
      <td>Priya</td>
      <td>Rajkot</td>
    </tr>
    <tr>
      <th scope="row">3</th>
      <td>21SOECA21035</td>
      <td>payal</td>
      <td>baroda</td>
    </tr>
  </tbody>
</table>
```

4.. student.component.ts

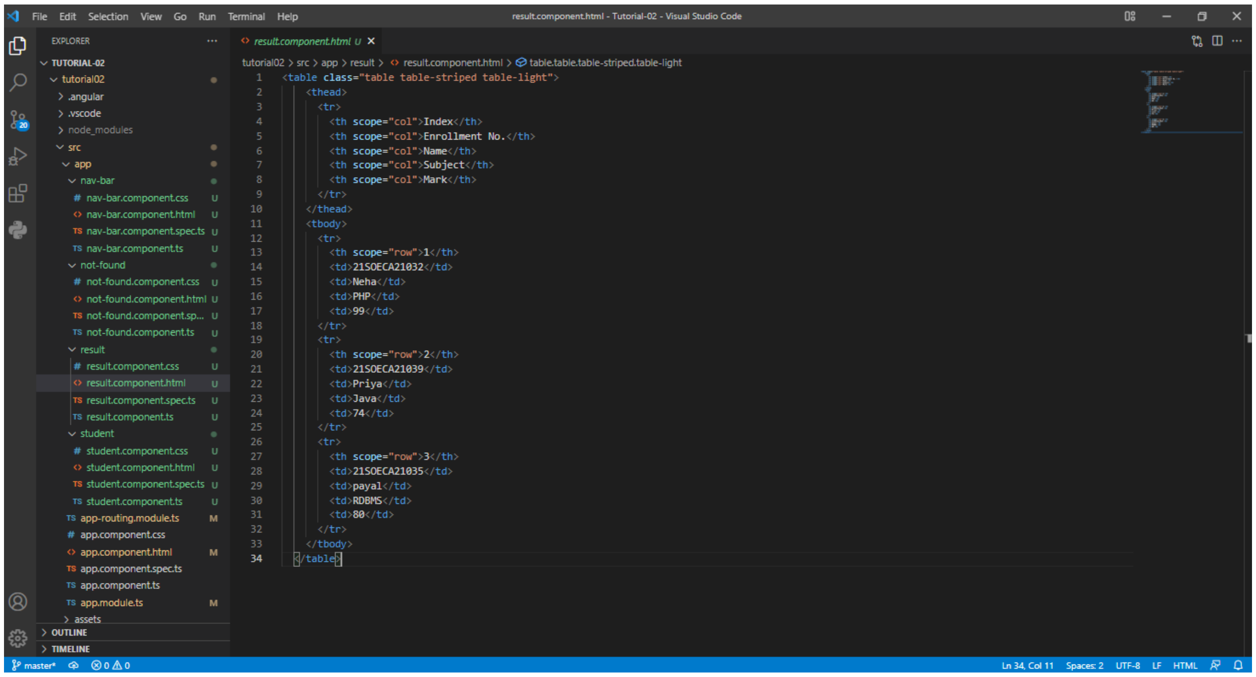
```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-student',
  templateUrl: './student.component.html',
  styleUrls: ['./student.component.css']
})
export class StudentComponent implements OnInit {
  constructor() { }

  ngOnInit(): void {
  }

}
```


student.component.ts U
tutorial02 > src > app > student > student.component.ts > ...

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-student',
  templateUrl: './student.component.html',
  styleUrls: ['./student.component.css']
})
export class StudentComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

5. result.component.html

```
<table class="table table-striped table-light">
    <thead>
      <tr>
        <th scope="col">Index</th>
        <th scope="col">Enrollment No.</th>
        <th scope="col">Name</th>
        <th scope="col">Subject</th>
        <th scope="col">Mark</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <th scope="row">1</th>
        <td>21SOECA21032</td>
        <td>Neha</td>
        <td>PHP</td>
        <td>99</td>
      </tr>
      <tr>
        <th scope="row">2</th>
        <td>21SOECA21039</td>
        <td>Priya</td>
        <td>Java</td>
        <td>74</td>
      </tr>
      <tr>
        <th scope="row">3</th>
        <td>21SOECA21035</td>
        <td>payal</td>
        <td>RDBMS</td>
        <td>80</td>
      </tr>
    </tbody>
  </table>
```

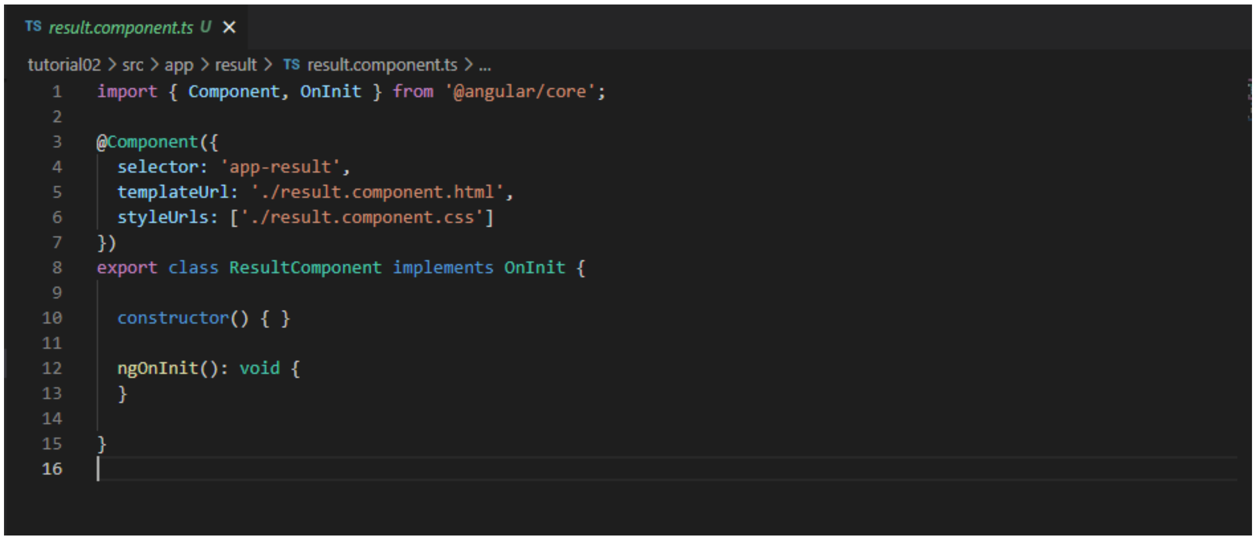

File Edit Selection View Go Run Terminal Help
result.component.html - Tutorial-02 - Visual Studio Code
EXPLORER
result.component.html U
tutorial02 > src > app > result > result.component.html > table.table.table-striped.table-light
OUTLINE
TIMELINE
master*
Ln 34, Col 11 Spaces: 2 UTF-8 LF HTML

```
<table class="table table-striped table-light">
  <thead>
    <tr>
      <th scope="col">Index</th>
      <th scope="col">Enrollment No.</th>
      <th scope="col">Name</th>
      <th scope="col">Subject</th>
      <th scope="col">Mark</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
      <td>21SOECA21032</td>
      <td>Neha</td>
      <td>PHP</td>
      <td>99</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>21SOECA21039</td>
      <td>Priya</td>
      <td>Java</td>
      <td>74</td>
    </tr>
    <tr>
      <th scope="row">3</th>
      <td>21SOECA21035</td>
      <td>payal</td>
      <td>RDBMS</td>
      <td>80</td>
    </tr>
  </tbody>
</table>
```

5.. result.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-result',
  templateUrl: './result.component.html',
  styleUrls: ['./result.component.css']
})
```

```
export class ResultComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```



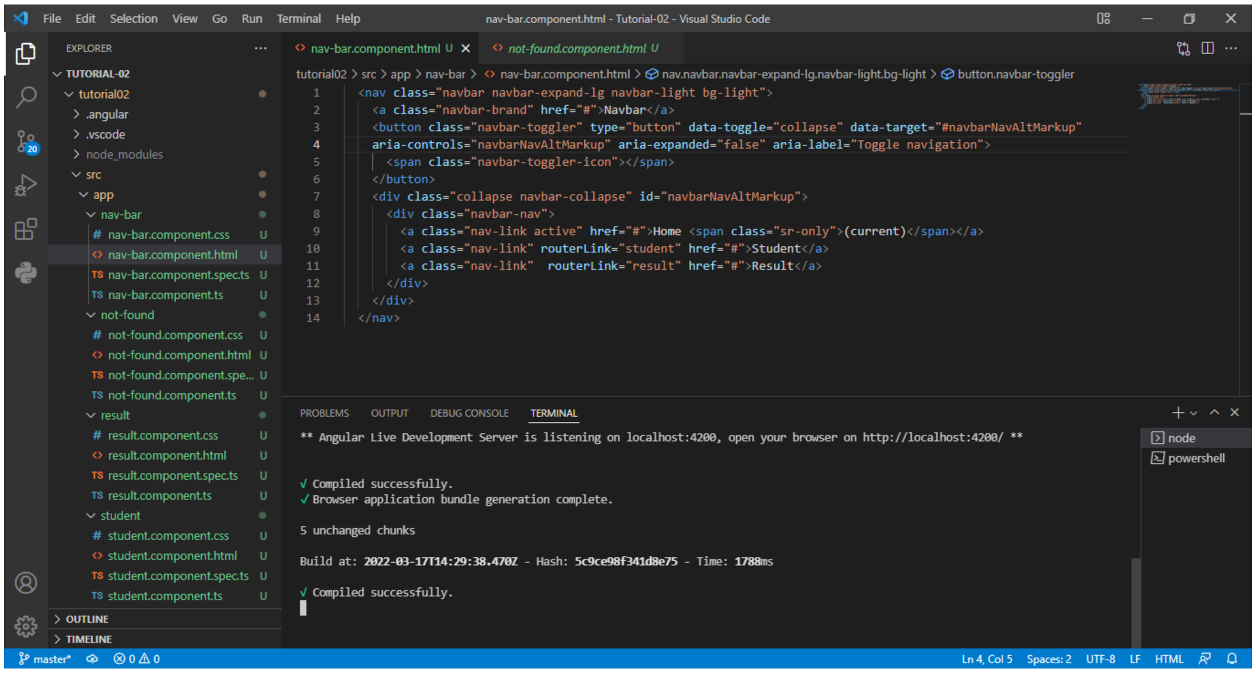
result.component.ts U ×

tutorial02 > src > app > result > result.component.ts > ...

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-result',
  templateUrl: './result.component.html',
  styleUrls: ['./result.component.css']
})
export class ResultComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}

```

6. nav-bar.component.html

```
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
      <div class="navbar-nav">
        <a class="nav-link active" href="#">Home <span class="sr-only">(current)</span></a>
        <a class="nav-link" routerLink="student" href="#">Student</a>
        <a class="nav-link"  routerLink="result" href="#">Result</a>
      </div>
    </div>
  </nav>
```

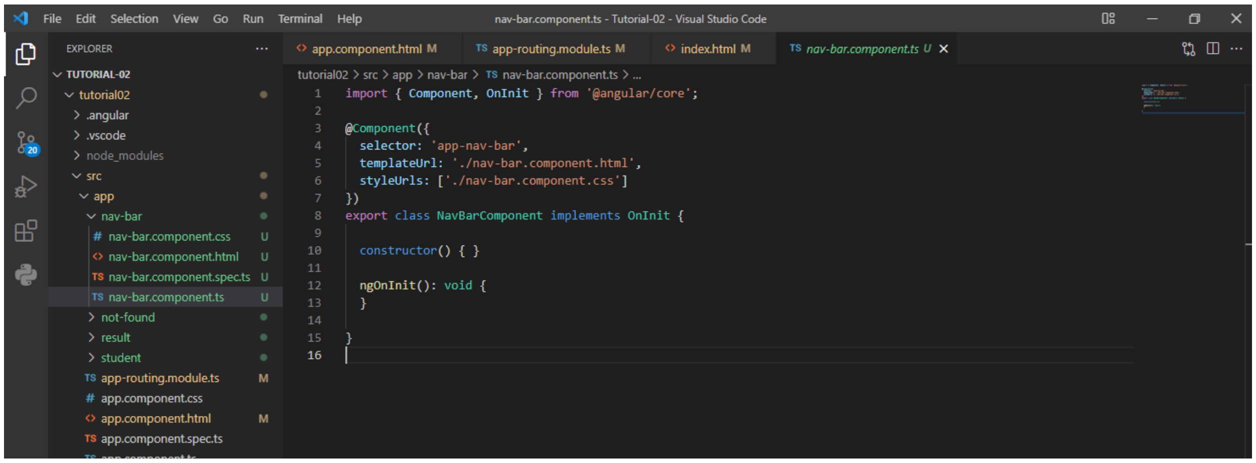

nav-bar.component.html - Tutorial-02 - Visual Studio Code
nav-bar.component.html U
not-found.component.html U
tutorial02 > src > app > nav-bar > nav-bar.component.html > nav.navbar.navbar-expand-lg.navbar-light.bg-light > button.navbar-toggler
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
node
powershell
master* ⊗ 0 ⚠ 0
Ln 4, Col 5 Spaces: 2 UTF-8 LF HTML

```html
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup"
  aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
    <div class="navbar-nav">
      <a class="nav-link active" href="#">Home <span class="sr-only">(current)</span></a>
      <a class="nav-link" routerLink="student" href="#">Student</a>
      <a class="nav-link"  routerLink="result" href="#">Result</a>
    </div>
  </div>
</nav>
```

```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **


√ Compiled successfully.
√ Browser application bundle generation complete.

5 unchanged chunks

Build at: 2022-03-17T14:29:38.470Z - Hash: 5c9ce98f341d8e75 - Time: 1788ms

√ Compiled successfully.
```

6.. nav-bar.component.ts

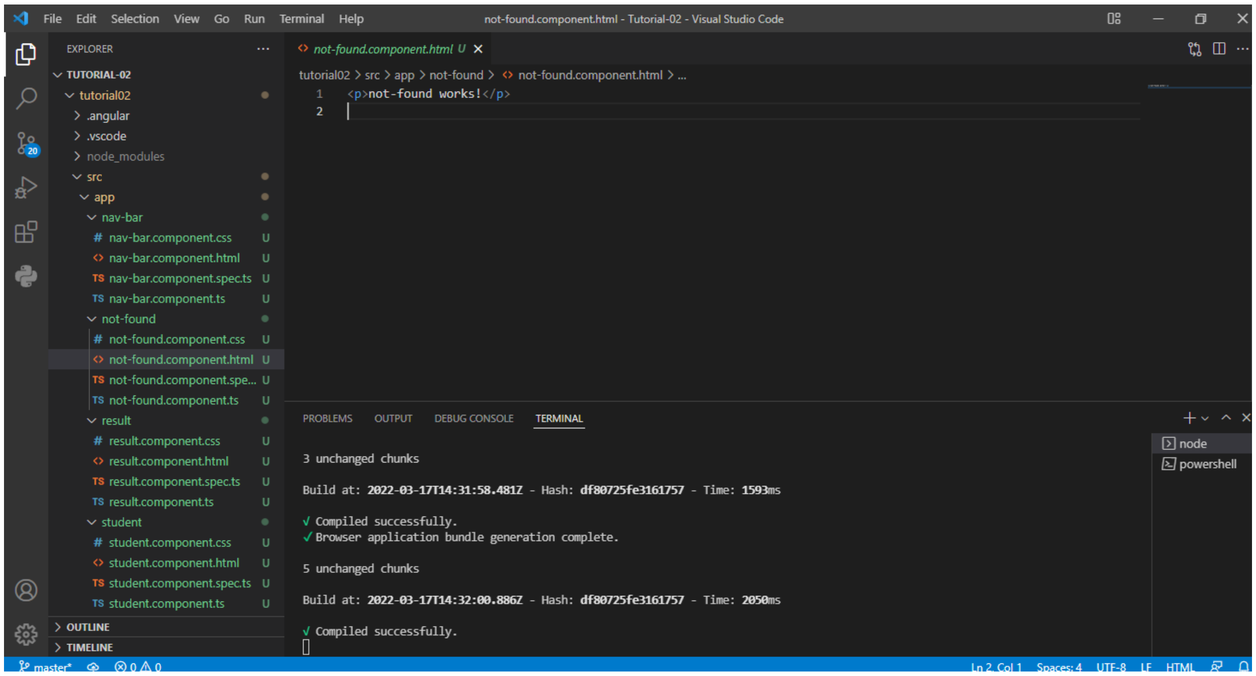
```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-nav-bar',
  templateUrl: './nav-bar.component.html',
  styleUrls: ['./nav-bar.component.css']
})
export class NavBarComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```



```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-nav-bar',
  templateUrl: './nav-bar.component.html',
  styleUrls: ['./nav-bar.component.css']
})
export class NavBarComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

7. not-found.component.html

<p>not-found works!</p>



```
<p>not-found works!</p>
```

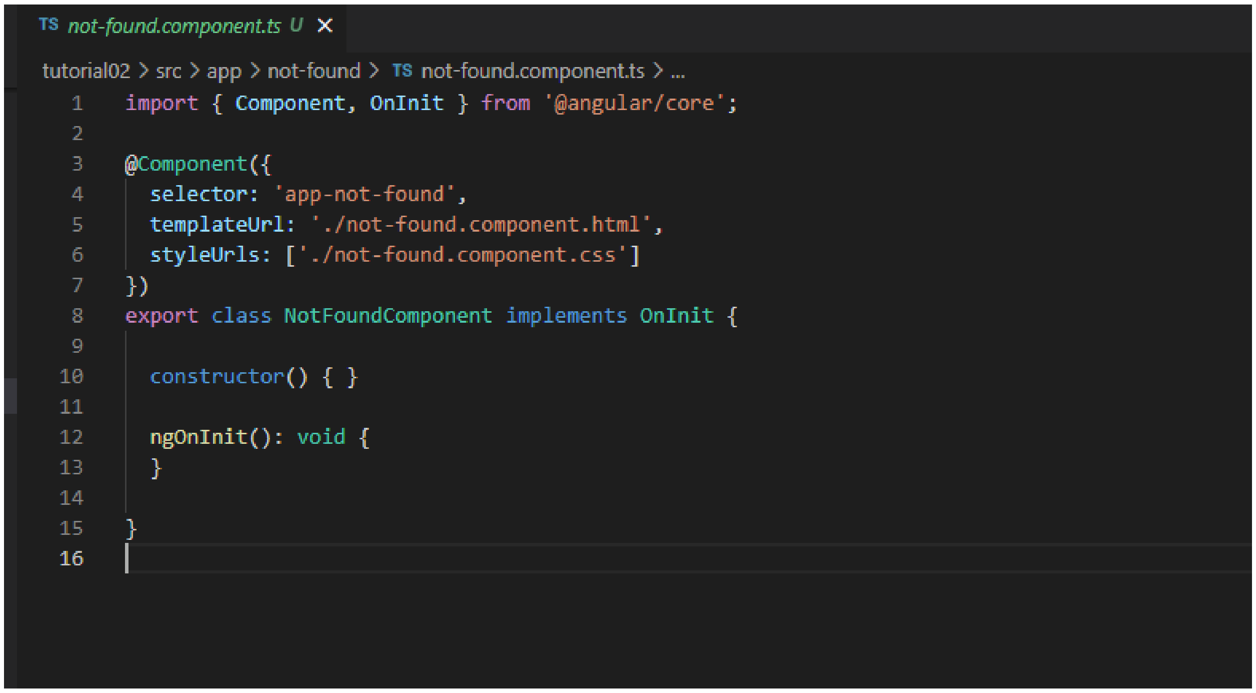
```
3 unchanged chunks

Build at: 2022-03-17T14:31:58.481Z - Hash: df80725fe3161757 - Time: 1593ms

√ Compiled successfully.
√ Browser application bundle generation complete.

5 unchanged chunks

Build at: 2022-03-17T14:32:00.886Z - Hash: df80725fe3161757 - Time: 2050ms

√ Compiled successfully.
```

7.. not-found.component.ts

import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-not-found',
  templateUrl: './not-found.component.html',
  styleUrls: ['./not-found.component.css']

```
})
export class NotFoundComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```



TS not-found.component.ts U ×

tutorial02 > src > app > not-found > TS not-found.component.ts > ...

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-not-found',
  templateUrl: './not-found.component.html',
  styleUrls: ['./not-found.component.css']
})
export class NotFoundComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

Practice Basic CRUD

Perform the following tasks:

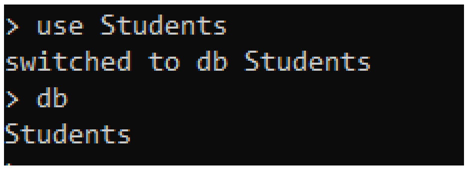
1. Create a database named "Students"

> use Students

switched to db Students

> db

Students

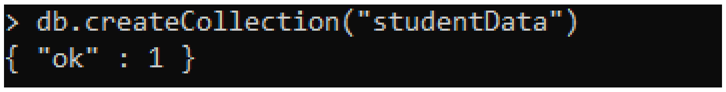


```
> use Students
switched to db Students
> db
Students
```

2. Create an empty collection named "studentData"

> db.createCollection("studentData")

{ "ok" : 1 }

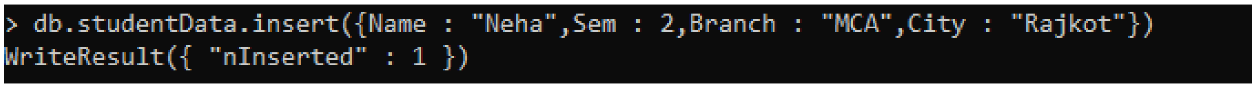


```
> db.createCollection("studentData")
{ "ok" : 1 }
```

3. Insert only one record with appropriate fields.

> db.studentData.insert({Name : "Neha",Sem : 2,Branch : "MCA",City : "Rajkot"})

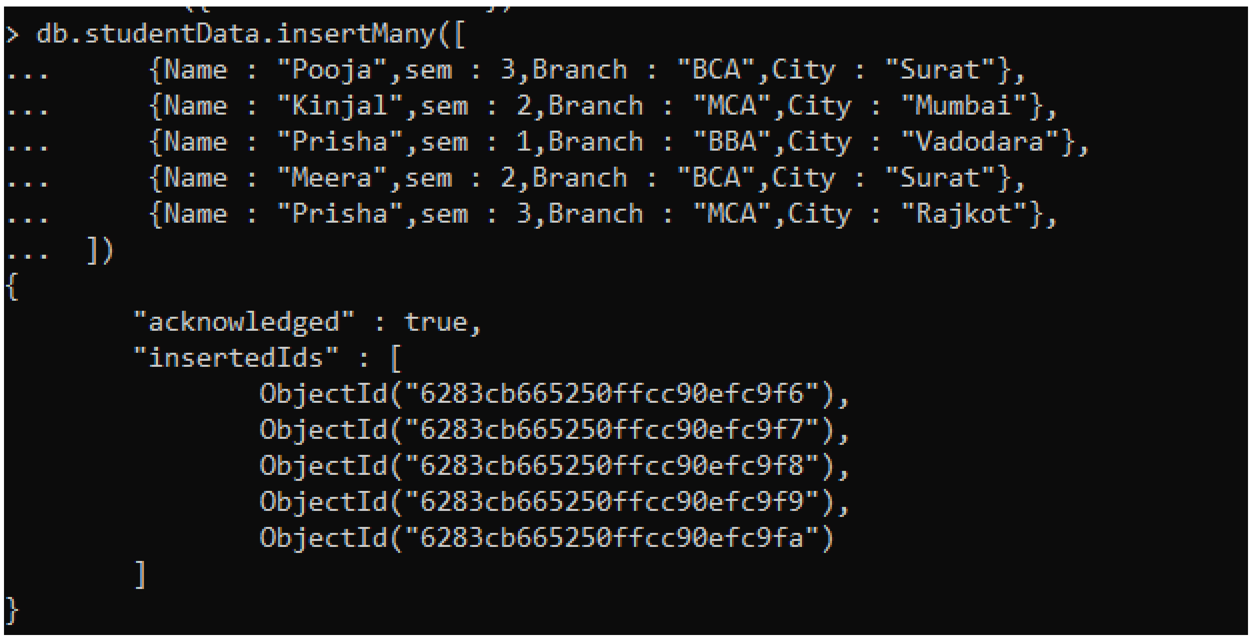
WriteResult({ "nInserted" : 1 })



```
> db.studentData.insert({Name : "Neha",Sem : 2,Branch : "MCA",City : "Rajkot"})
WriteResult({ "nInserted" : 1 })
```

4. Try to insert 5 records together using single query

```
> db.studentData.insertMany([
...     {Name : "Pooja",sem : 3,Branch : "BCA",City : "Surat"},
...     {Name : "Kinjal",sem : 2,Branch : "MCA",City : "Mumbai"},
...     {Name : "Prisha",sem : 1,Branch : "BBA",City : "Vadodara"},
...     {Name : "Meera",sem : 2,Branch : "BCA",City : "Surat"},
...     {Name : "Prisha",sem : 3,Branch : "MCA",City : "Rajkot"},
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6283cb665250ffcc90efc9f6"),
                ObjectId("6283cb665250ffcc90efc9f7"),
                ObjectId("6283cb665250ffcc90efc9f8"),
                ObjectId("6283cb665250ffcc90efc9f9"),
                ObjectId("6283cb665250ffcc90efc9fa")
        ]
}
```

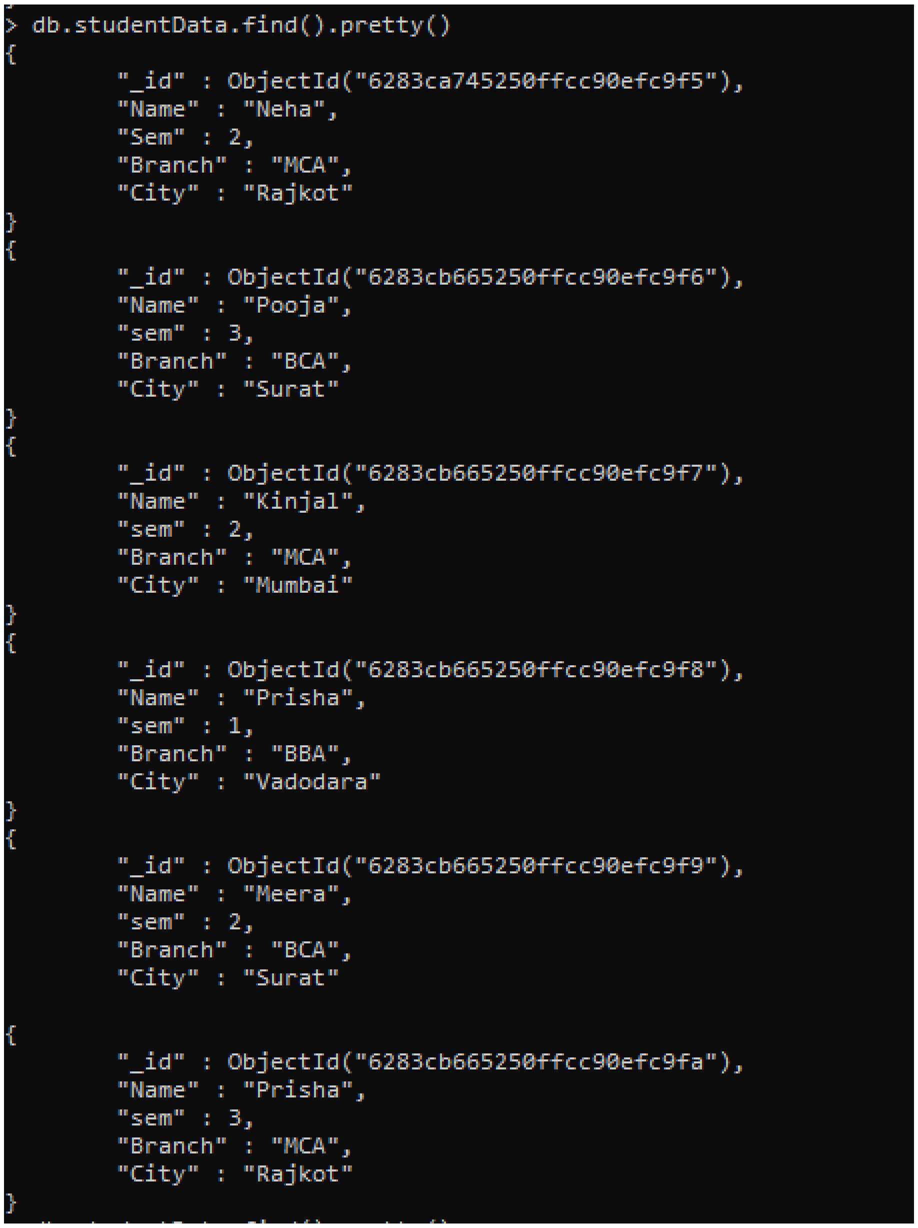


```
> db.studentData.insertMany([
...      {Name : "Pooja",sem : 3,Branch : "BCA",City : "Surat"},
...      {Name : "Kinjal",sem : 2,Branch : "MCA",City : "Mumbai"},
...      {Name : "Prisha",sem : 1,Branch : "BBA",City : "Vadodara"},
...      {Name : "Meera",sem : 2,Branch : "BCA",City : "Surat"},
...      {Name : "Prisha",sem : 3,Branch : "MCA",City : "Rajkot"},
...   ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6283cb665250ffcc90efc9f6"),
                ObjectId("6283cb665250ffcc90efc9f7"),
                ObjectId("6283cb665250ffcc90efc9f8"),
                ObjectId("6283cb665250ffcc90efc9f9"),
                ObjectId("6283cb665250ffcc90efc9fa")
        ]
}
```

5. Fetch all the documents in formatted manner.

```
> db.studentData.find().pretty()

{

        "_id" : ObjectId("6283ca745250ffcc90efc9f5"),

        "Name" : "Neha",

        "Sem" : 2,

        "Branch" : "MCA",

        "City" : "Rajkot"

}

{

        "_id" : ObjectId("6283cb665250ffcc90efc9f6"),

        "Name" : "Pooja",

        "sem" : 3,

        "Branch" : "BCA",

        "City" : "Surat"
```

```
}
{
        "_id" : ObjectId("6283cb665250ffcc90efc9f7"),
        "Name" : "Kinjal",
        "sem" : 2,
        "Branch" : "MCA",
        "City" : "Mumbai"
}
{
        "_id" : ObjectId("6283cb665250ffcc90efc9f8"),
        "Name" : "Prisha",
        "sem" : 1,
        "Branch" : "BBA",
        "City" : "Vadodara"
}
{
        "_id" : ObjectId("6283cb665250ffcc90efc9f9"),
        "Name" : "Meera",
        "sem" : 2,
        "Branch" : "BCA",
        "City" : "Surat"
}
{
        "_id" : ObjectId("6283cb665250ffcc90efc9fa"),
        "Name" : "Prisha",
        "sem" : 3,
        "Branch" : "MCA",

        "City" : "Rajkot"

}
```

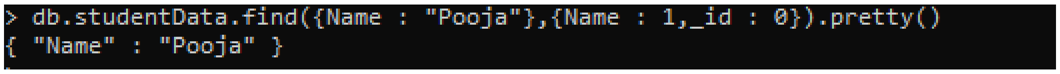


```
> db.studentData.find().pretty()
{
        "_id" : ObjectId("6283ca745250ffcc90efc9f5"),
        "Name" : "Neha",
        "Sem" : 2,
        "Branch" : "MCA",
        "City" : "Rajkot"
}
{
        "_id" : ObjectId("6283cb665250ffcc90efc9f6"),
        "Name" : "Pooja",
        "sem" : 3,
        "Branch" : "BCA",
        "City" : "Surat"
}
{
        "_id" : ObjectId("6283cb665250ffcc90efc9f7"),
        "Name" : "Kinjal",
        "sem" : 2,
        "Branch" : "MCA",
        "City" : "Mumbai"
}
{
        "_id" : ObjectId("6283cb665250ffcc90efc9f8"),
        "Name" : "Prisha",
        "sem" : 1,
        "Branch" : "BBA",
        "City" : "Vadodara"
}
{
        "_id" : ObjectId("6283cb665250ffcc90efc9f9"),
        "Name" : "Meera",
        "sem" : 2,
        "Branch" : "BCA",
        "City" : "Surat"

{
        "_id" : ObjectId("6283cb665250ffcc90efc9fa"),
        "Name" : "Prisha",
        "sem" : 3,
        "Branch" : "MCA",
        "City" : "Rajkot"
}
```

6. Fetch the document based on the filter criteria.

> db.studentData.find({Name : "Pooja"},{Name : 1,_id : 0}).pretty()

{ "Name" : "Pooja" }



```
> db.studentData.find({Name : "Pooja"},{Name : 1,_id : 0}).pretty()
{ "Name" : "Pooja" }
```

7. Delete one document from the "studentData".

> db.studentData.deleteOne({Name : "Prisha"})

{ "acknowledged" : true, "deletedCount" : 1 }

> db.studentData.find()

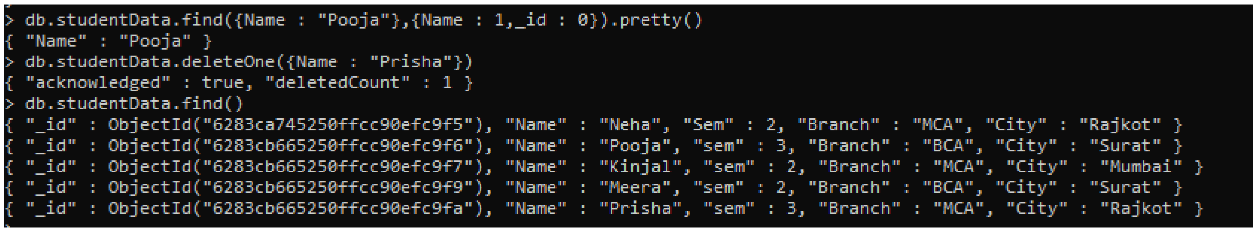
{ "_id" : ObjectId("6283ca745250ffcc90efc9f5"), "Name" : "Neha", "Sem" : 2, "Branch" : "MCA", "City" : "Rajkot" }

{ "_id" : ObjectId("6283cb665250ffcc90efc9f6"), "Name" : "Pooja", "sem" : 3, "Branch" : "BCA", "City" : "Surat" }

{ "_id" : ObjectId("6283cb665250ffcc90efc9f7"), "Name" : "Kinjal", "sem" : 2, "Branch" : "MCA", "City" : "Mumbai" }

{ "_id" : ObjectId("6283cb665250ffcc90efc9f9"), "Name" : "Meera", "sem" : 2, "Branch" : "BCA", "City" : "Surat" }

{ "_id" : ObjectId("6283cb665250ffcc90efc9fa"), "Name" : "Prisha", "sem" : 3, "Branch" : "MCA", "City" : "Rajkot" }



```
> db.studentData.find({Name : "Pooja"},{Name : 1,_id : 0}).pretty()
{ "Name" : "Pooja" }
> db.studentData.deleteOne({Name : "Prisha"})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.studentData.find()
{ "_id" : ObjectId("6283ca745250ffcc90efc9f5"), "Name" : "Neha", "Sem" : 2, "Branch" : "MCA", "City" : "Rajkot" }
{ "_id" : ObjectId("6283cb665250ffcc90efc9f6"), "Name" : "Pooja", "sem" : 3, "Branch" : "BCA", "City" : "Surat" }
{ "_id" : ObjectId("6283cb665250ffcc90efc9f7"), "Name" : "Kinjal", "sem" : 2, "Branch" : "MCA", "City" : "Mumbai" }
{ "_id" : ObjectId("6283cb665250ffcc90efc9f9"), "Name" : "Meera", "sem" : 2, "Branch" : "BCA", "City" : "Surat" }
{ "_id" : ObjectId("6283cb665250ffcc90efc9fa"), "Name" : "Prisha", "sem" : 3, "Branch" : "MCA", "City" : "Rajkot" }
```

8. Delete the documents based on the filter criteria.

> db.studentData.deleteOne({Name : "Meera"})

{ "acknowledged" : true, "deletedCount" : 1 }

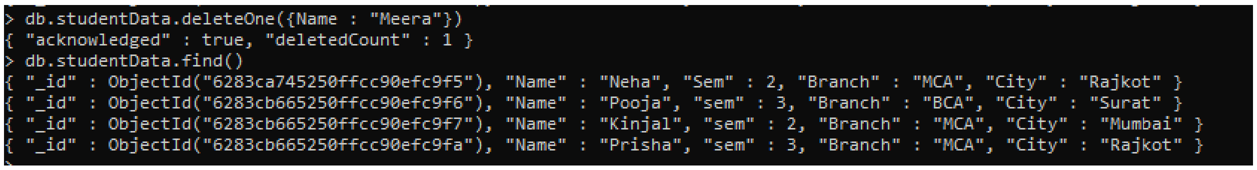
> db.studentData.find()

{ "_id" : ObjectId("6283ca745250ffcc90efc9f5"), "Name" : "Neha", "Sem" : 2, "Branch" : "MCA", "City" : "Rajkot" }

{ "_id" : ObjectId("6283cb665250ffcc90efc9f6"), "Name" : "Pooja", "sem" : 3, "Branch" : "BCA", "City" : "Surat" }

{ "_id" : ObjectId("6283cb665250ffcc90efc9f7"), "Name" : "Kinjal", "sem" : 2, "Branch" : "MCA", "City" : "Mumbai" }

{ "_id" : ObjectId("6283cb665250ffcc90efc9fa"), "Name" : "Prisha", "sem" : 3, "Branch" : "MCA", "City" : "Rajkot" }



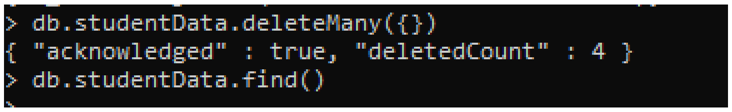
```
> db.studentData.deleteOne({Name : "Meera"})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.studentData.find()
{ "_id" : ObjectId("6283ca745250ffcc90efc9f5"), "Name" : "Neha", "Sem" : 2, "Branch" : "MCA", "City" : "Rajkot" }
{ "_id" : ObjectId("6283cb665250ffcc90efc9f6"), "Name" : "Pooja", "sem" : 3, "Branch" : "BCA", "City" : "Surat" }
{ "_id" : ObjectId("6283cb665250ffcc90efc9f7"), "Name" : "Kinjal", "sem" : 2, "Branch" : "MCA", "City" : "Mumbai" }
{ "_id" : ObjectId("6283cb665250ffcc90efc9fa"), "Name" : "Prisha", "sem" : 3, "Branch" : "MCA", "City" : "Rajkot" }
```

9. Delete all the documents from the "studentData: collection

> db.studentData.deleteMany({})

{ "acknowledged" : true, "deletedCount" : 4 }

> db.studentData.find()

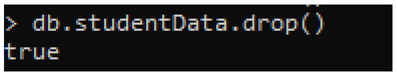


```
> db.studentData.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 4 }
> db.studentData.find()
```

10. Drop the collection.

> db.studentData.drop()

true



```
> db.studentData.drop()
true
```

Hands on with Querying using Regex Expression

In order to perform following queries use a collection `products` with the following documents:

{ "_id" : 100, "sku" : "abc123", "description" : "Single line description." }

{ "_id" : 101, "sku" : "abc789", "description" : "First line\nSecond line" }

{ "_id" : 102, "sku" : "xyz456", "description" : "Many spaces before   line" }

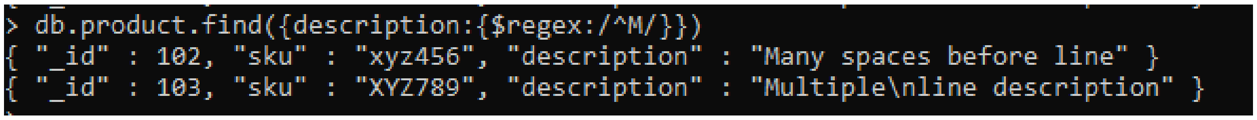
{ "_id" : 103, "sku" : "XYZ789", "description" : "Multiple\nline description" }

1. find the documents whose description starts with "M".

> db.product.find({description:{$regex:/^M/}})

{ "_id" : 102, "sku" : "xyz456", "description" : "Many spaces before line" }

{ "_id" : 103, "sku" : "XYZ789", "description" : "Multiple\nline description" }

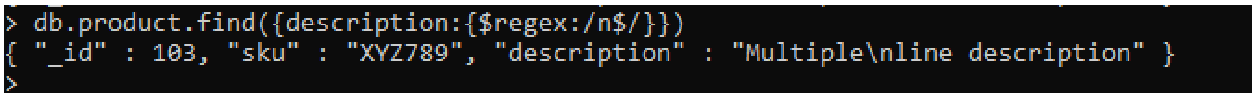


```
> db.product.find({description:{$regex:/^M/}})
{ "_id" : 102, "sku" : "xyz456", "description" : "Many spaces before line" }
{ "_id" : 103, "sku" : "XYZ789", "description" : "Multiple\nline description" }
```

2. find the documents whose description end with "n"

> db.product.find({description:{$regex:/n$/}})

{ "_id" : 103, "sku" : "XYZ789", "description" : "Multiple\nline description" }



```
> db.product.find({description:{$regex:/n$/}})
{ "_id" : 103, "sku" : "XYZ789", "description" : "Multiple\nline description" }
>
```

3. find the documents whose description contains "line" word"

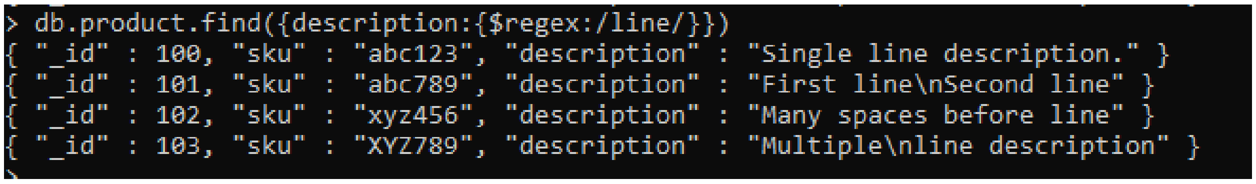
> db.product.find({description:{$regex:/line/}})

{ "_id" : 100, "sku" : "abc123", "description" : "Single line description." }

{ "_id" : 101, "sku" : "abc789", "description" : "First line\nSecond line" }

{ "_id" : 102, "sku" : "xyz456", "description" : "Many spaces before line" }

{ "_id" : 103, "sku" : "XYZ789", "description" : "Multiple\nline description" }



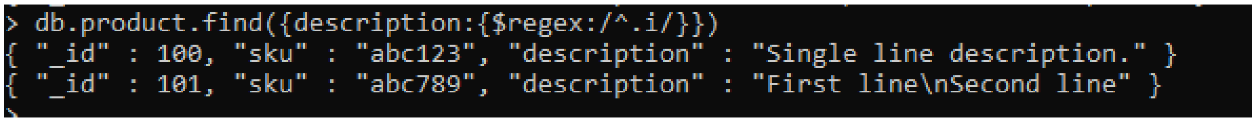
```
> db.product.find({description:{$regex:/line/}})
{ "_id" : 100, "sku" : "abc123", "description" : "Single line description." }
{ "_id" : 101, "sku" : "abc789", "description" : "First line\nSecond line" }
{ "_id" : 102, "sku" : "xyz456", "description" : "Many spaces before line" }
{ "_id" : 103, "sku" : "XYZ789", "description" : "Multiple\nline description" }
```

4. find the documents whose description contains second character "i".

> db.product.find({description:{$regex:/^.i/}})

{ "_id" : 100, "sku" : "abc123", "description" : "Single line description." }

{ "_id" : 101, "sku" : "abc789", "description" : "First line\nSecond line" }



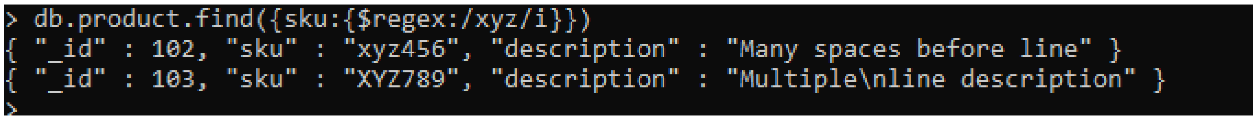
```
> db.product.find({description:{$regex:/^.i/}})
{ "_id" : 100, "sku" : "abc123", "description" : "Single line description." }
{ "_id" : 101, "sku" : "abc789", "description" : "First line\nSecond line" }
```

5. find the documents where "sku" fields contains "xyz", ignore the case sensitivity.

> db.product.find({sku:{$regex:/xyz/i}})

{ "_id" : 102, "sku" : "xyz456", "description" : "Many spaces before line" }

{ "_id" : 103, "sku" : "XYZ789", "description" : "Multiple\nline description" }

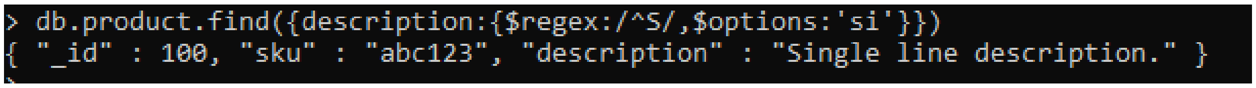


```
> db.product.find({sku:{$regex:/xyz/i}})
{ "_id" : 102, "sku" : "xyz456", "description" : "Many spaces before line" }
{ "_id" : 103, "sku" : "XYZ789", "description" : "Multiple\nline description" }
```

6. find the documents where any line from the description starts with 'S..

> db.product.find({description:{$regex:/^S/,$options:'si'}})

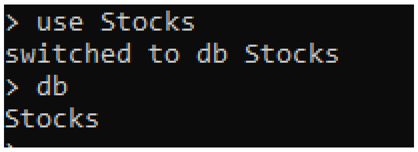
{ "_id" : 100, "sku" : "abc123", "description" : "Single line description." }



```
> db.product.find({description:{$regex:/^S/,$options:'si'}})
{ "_id" : 100, "sku" : "abc123", "description" : "Single line description." }
```

1. Create a Database named "Stocks".

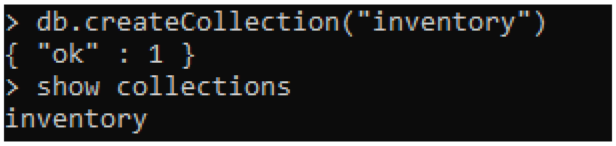
```
> use Stocks
switched to db Stocks
> db
Stocks
```



```
> use Stocks
switched to db Stocks
> db
Stocks
```

2. Create an empty collection named "inventory".

```
> db.createCollection("inventory")
{ "ok" : 1 }
> show collections
inventory
```

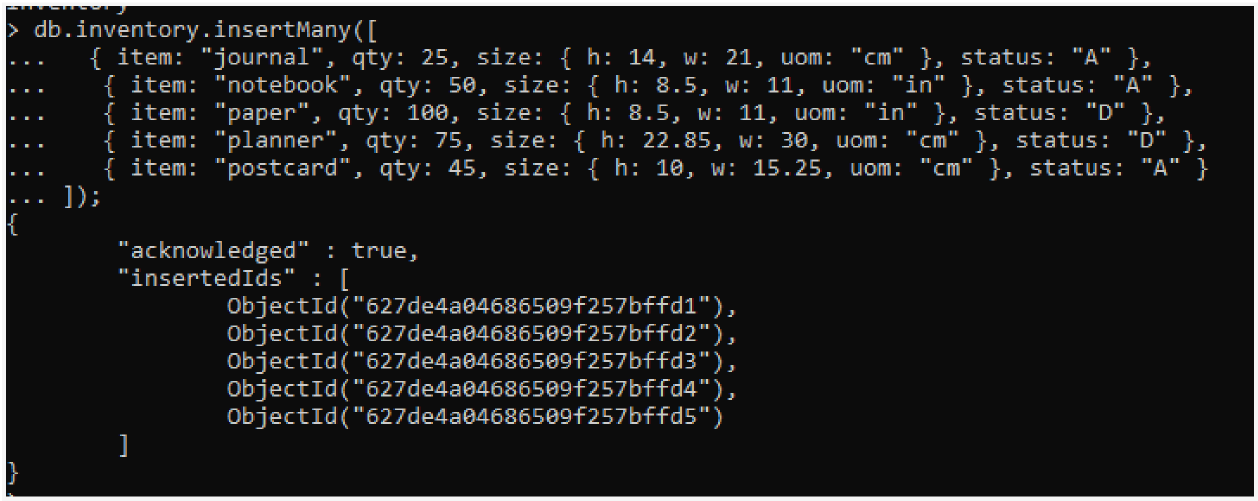


```
> db.createCollection("inventory")
{ "ok" : 1 }
> show collections
inventory
```

3. Insert below records in the inventory collection all together.

```
db.inventory.insertMany([

  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" },
status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" },
status: "A" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" },
status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" },
status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" },
status: "A" }
]);
```

```
> db.inventory.insertMany([
...    { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" },
status: "A" },

...     { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in"
}, status: "A" },

...     { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" },
status: "D" },

...     { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm"
}, status: "D" },

...     { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm"
}, status: "A" }

... ]);

{

        "acknowledged" : true,

        "insertedIds" : [

                ObjectId("627de4a04686509f257bffd1"),

                ObjectId("627de4a04686509f257bffd2"),

                ObjectId("627de4a04686509f257bffd3"),

                ObjectId("627de4a04686509f257bffd4"),

                ObjectId("627de4a04686509f257bffd5")

        ]

}
```

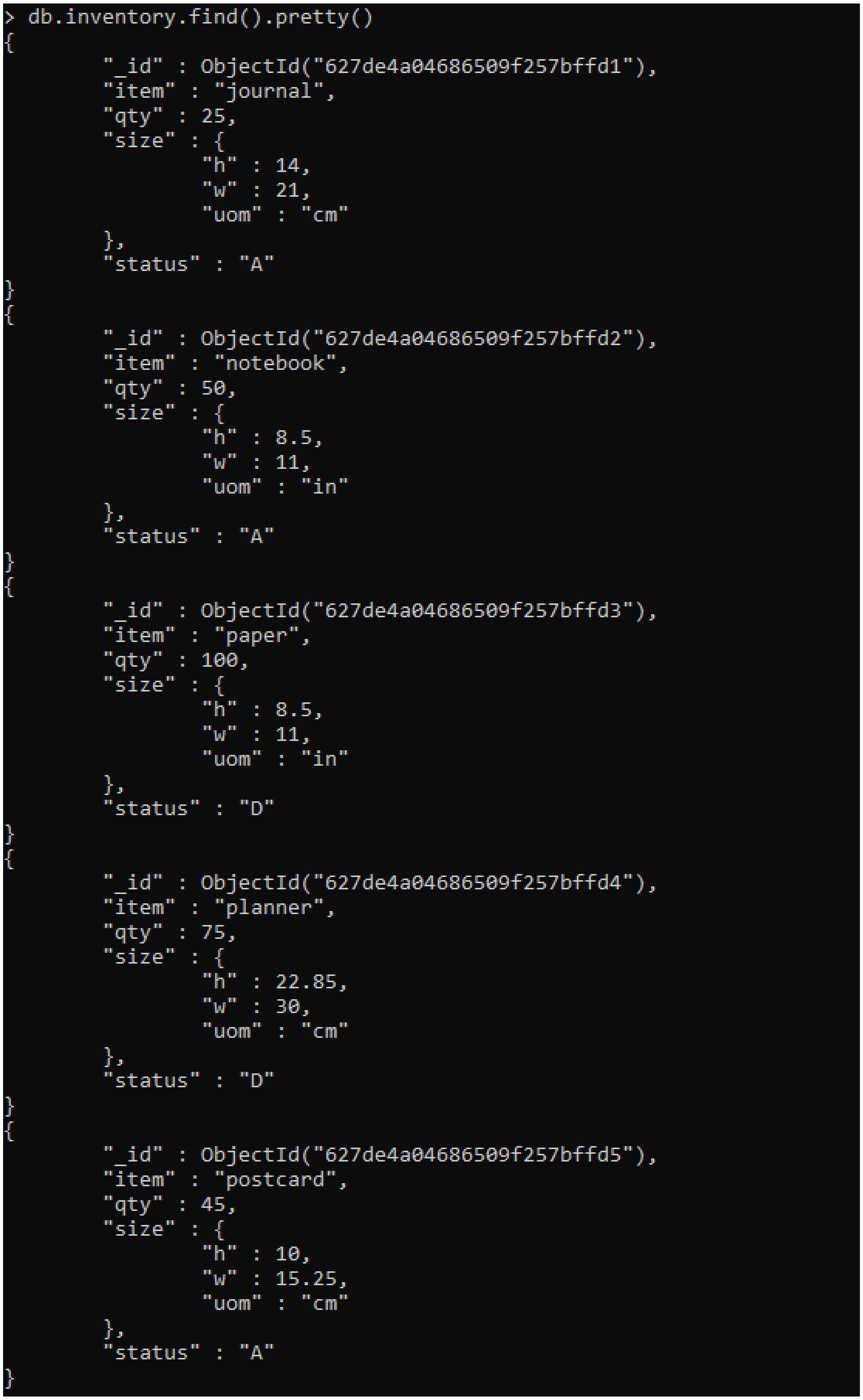


```
> db.inventory.insertMany([
...    { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
...     { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
...     { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
...     { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
...     { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
... ]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("627de4a04686509f257bffd1"),
                ObjectId("627de4a04686509f257bffd2"),
                ObjectId("627de4a04686509f257bffd3"),
                ObjectId("627de4a04686509f257bffd4"),
                ObjectId("627de4a04686509f257bffd5")
        ]
}
```

4. Select all documents from inventory collection

```
> db.inventory.find().pretty()
{
        "_id" : ObjectId("627de4a04686509f257bffd1"),
        "item" : "journal",
        "qty" : 25,
        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd2"),
        "item" : "notebook",
        "qty" : 50,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "A"
```

```
}
{
	"_id" : ObjectId("627de4a04686509f257bffd3"),
	"item" : "paper",
	"qty" : 100,
	"size" : {
		"h" : 8.5,
		"w" : 11,
		"uom" : "in"
	},
	"status" : "D"
}
{
	"_id" : ObjectId("627de4a04686509f257bffd4"),
	"item" : "planner",
	"qty" : 75,
	"size" : {
		"h" : 22.85,
		"w" : 30,
		"uom" : "cm"
	},
	"status" : "D"
}
{
	"_id" : ObjectId("627de4a04686509f257bffd5"),
	"item" : "postcard",
    "qty" : 45,
    "size" : {
        "h" : 10,
        "w" : 15.25,
        "uom" : "cm"
    },
    "status" : "A"
}
```



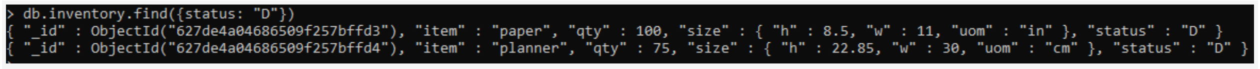
```
> db.inventory.find().pretty()
{
        "_id" : ObjectId("627de4a04686509f257bffd1"),
        "item" : "journal",
        "qty" : 25,
        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd2"),
        "item" : "notebook",
        "qty" : 50,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd3"),
        "item" : "paper",
        "qty" : 100,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "D"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd4"),
        "item" : "planner",
        "qty" : 75,
        "size" : {
                "h" : 22.85,
                "w" : 30,
                "uom" : "cm"
        },
        "status" : "D"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd5"),
        "item" : "postcard",
        "qty" : 45,
        "size" : {
                "h" : 10,
                "w" : 15.25,
                "uom" : "cm"
        },
        "status" : "A"
}
```

5. selects from the `inventory` collection all documents where the `status` equals `"D"`

```
> db.inventory.find({status: "D"})
```

```
{ "_id" : ObjectId("627de4a04686509f257bffd3"), "item" :
"paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" :
"in" }, "status" : "D" }
```

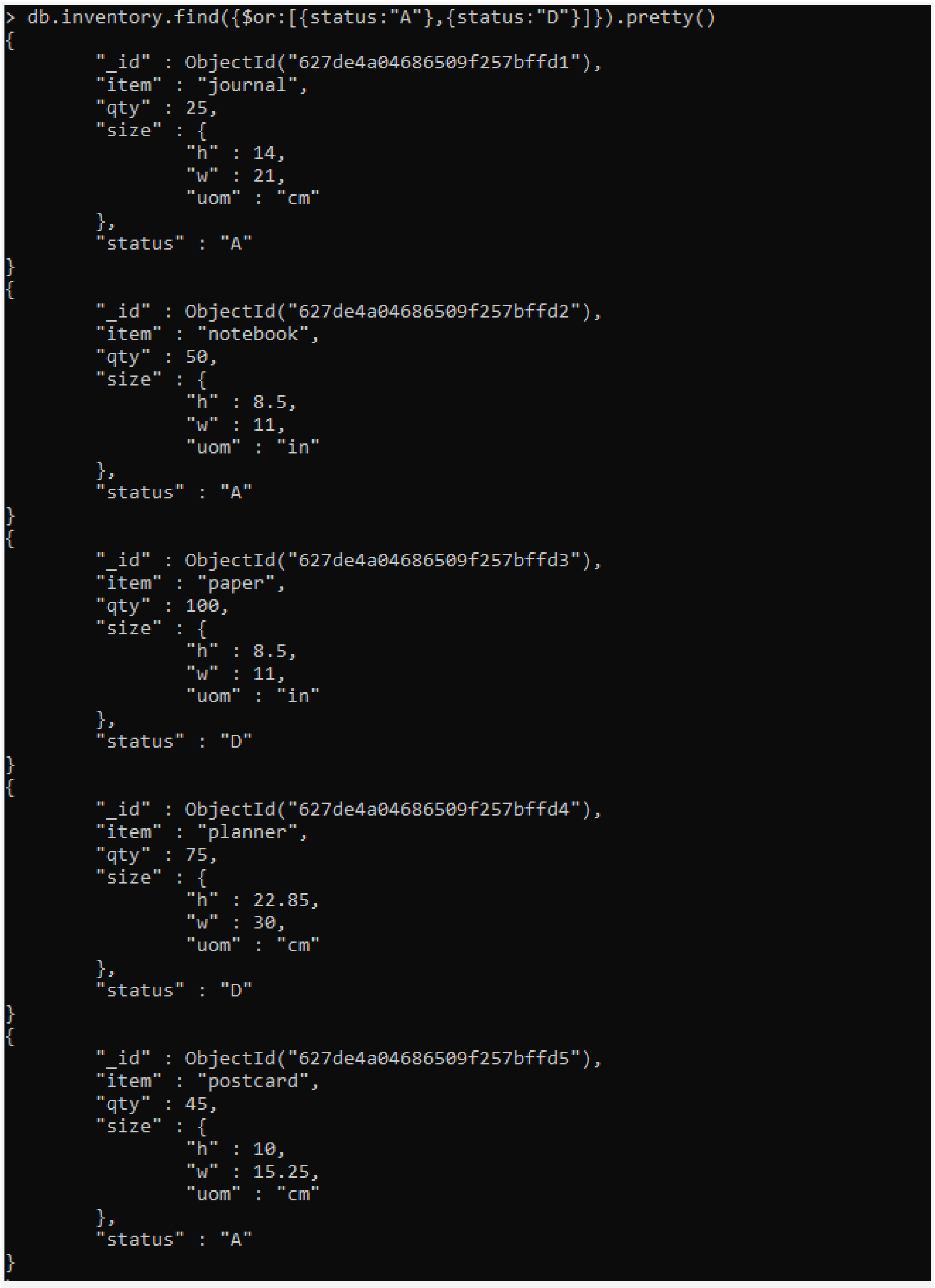
```
{ "_id" : ObjectId("627de4a04686509f257bffd4"), "item" :
"planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" :
"cm" }, "status" : "D" }
```



```
> db.inventory.find({status: "D"})
{ "_id" : ObjectId("627de4a04686509f257bffd3"), "item" : "paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "D" }
{ "_id" : ObjectId("627de4a04686509f257bffd4"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "D" }
```

6. Retrieves all documents from the `inventory` collection where `status` equals either `"A"` or `"D"`.

```
> db.inventory.find({$or:[{status:"A"},{status:"D"}]}).pretty()
{
        "_id" : ObjectId("627de4a04686509f257bffd1"),
        "item" : "journal",
        "qty" : 25,
        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd2"),
        "item" : "notebook",
        "qty" : 50,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd3"),
        "item" : "paper",
        "qty" : 100,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "D"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd4"),
        "item" : "planner",
        "qty" : 75,
        "size" : {
                "h" : 22.85,
                "w" : 30,
                "uom" : "cm"
        },
        "status" : "D"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd5"),
        "item" : "postcard",
        "qty" : 45,
        "size" : {
                "h" : 10,
                "w" : 15.25,
                "uom" : "cm"
        },
        "status" : "A"
}
```



```
> db.inventory.find({$or:[{status:"A"},{status:"D"}]}).pretty()
{
        "_id" : ObjectId("627de4a04686509f257bffd1"),
        "item" : "journal",
        "qty" : 25,
        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd2"),
        "item" : "notebook",
        "qty" : 50,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd3"),
        "item" : "paper",
        "qty" : 100,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "D"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd4"),
        "item" : "planner",
        "qty" : 75,
        "size" : {
                "h" : 22.85,
                "w" : 30,
                "uom" : "cm"
        },
        "status" : "D"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd5"),
        "item" : "postcard",
        "qty" : 45,
        "size" : {
                "h" : 10,
                "w" : 15.25,
                "uom" : "cm"
        },
        "status" : "A"
}
```

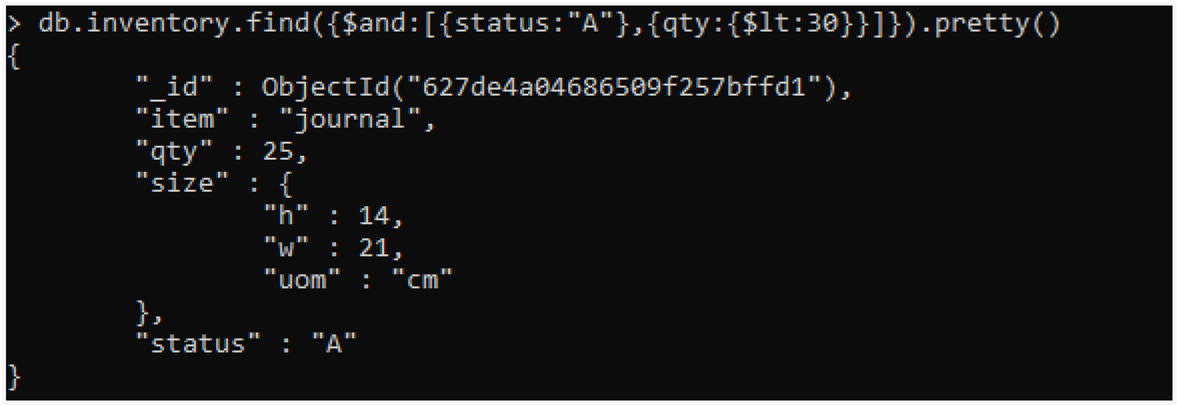
7. Retrieve documents in the `inventory` collection where the `status` equals `"A"` and `qty` is less than (`$lt`

(Links to an external site.)

Links to an external site.

) `30`

```
>
db.inventory.find({$and:[{status:"A"},{qty:{$lt:30}}]}).pretty()
{
        "_id" : ObjectId("627de4a04686509f257bffd1"),
        "item" : "journal",
        "qty" : 25,
        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
```



```
> db.inventory.find({$and:[{status:"A"},{qty:{$lt:30}}]}).pretty()
{
        "_id" : ObjectId("627de4a04686509f257bffd1"),
        "item" : "journal",
        "qty" : 25,
        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
```

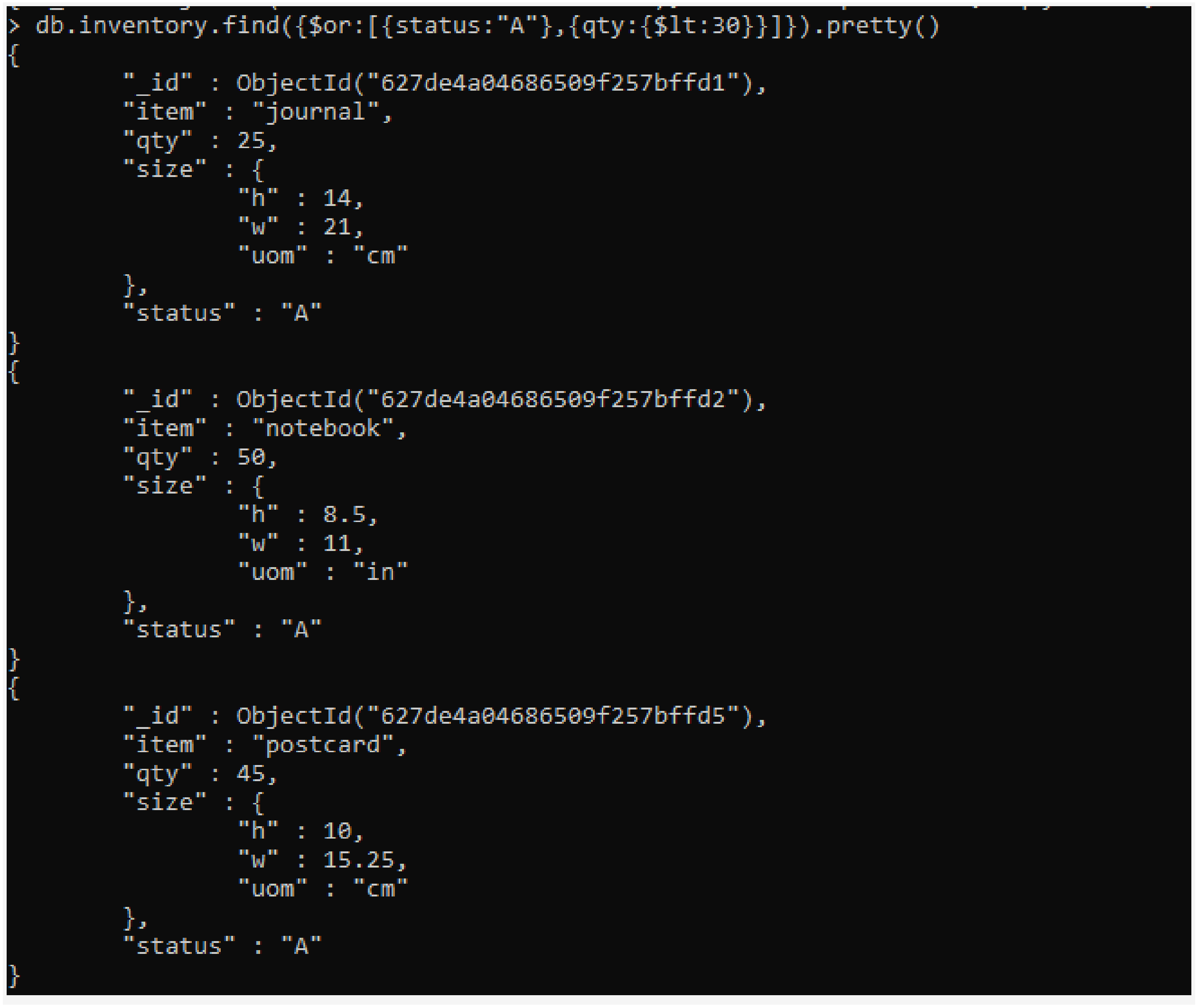
8. Retrieve all documents in the collection where the `status` equals `"A"` or `qty` is less than (`$lt`

(Links to an external site.)

Links to an external site.

) `30`

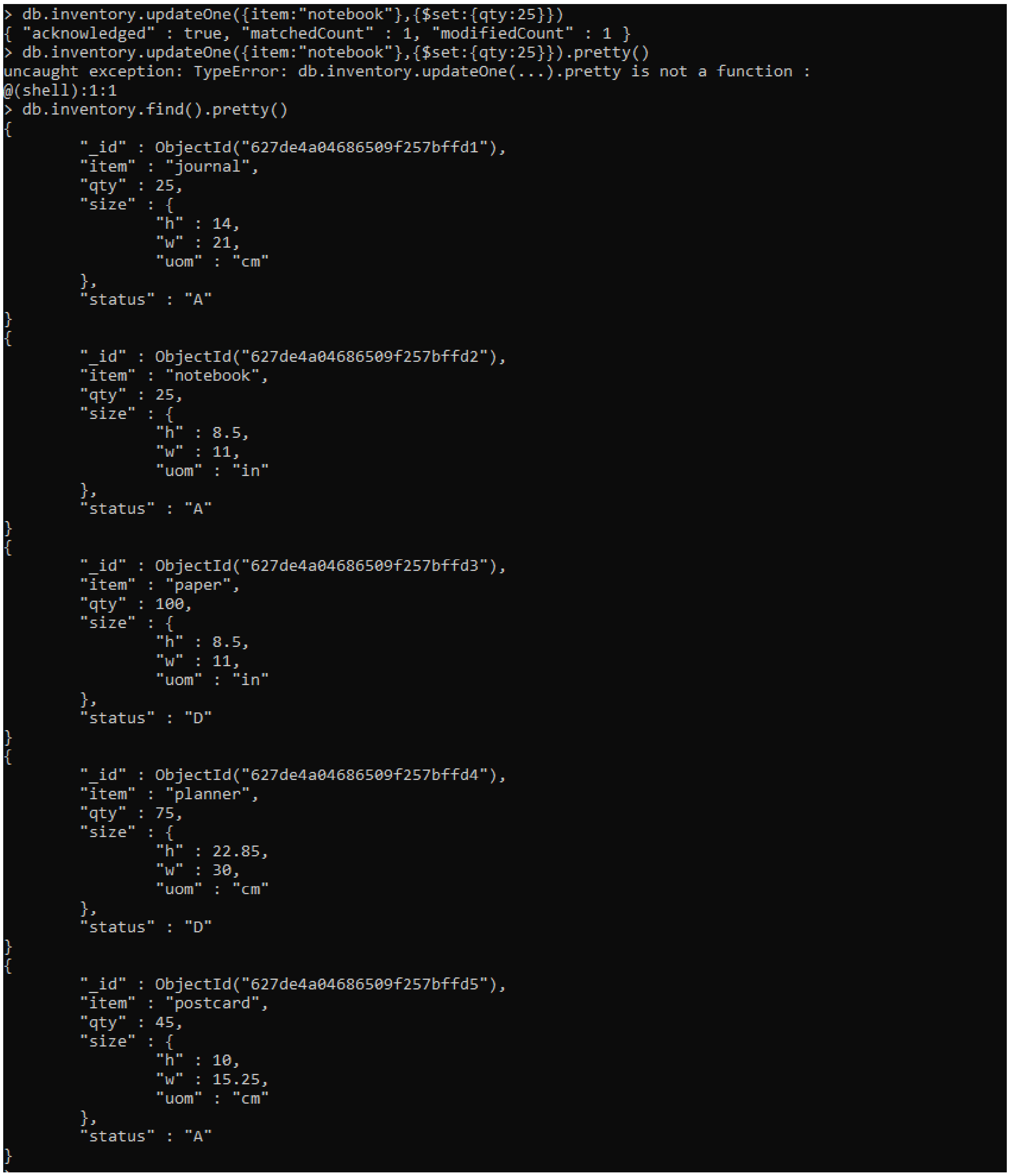
```
>
db.inventory.find({$or:[{status:"A"},{qty:{$lt:30}}]}).pretty()
{
        "_id" : ObjectId("627de4a04686509f257bffd1"),
        "item" : "journal",
        "qty" : 25,
        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd2"),
        "item" : "notebook",
        "qty" : 50,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd5"),
        "item" : "postcard",
        "qty" : 45,
        "size" : {
                "h" : 10,
                "w" : 15.25,
                "uom" : "cm"
        },
        "status" : "A"
}
```



```
> db.inventory.find({$or:[{status:"A"},{qty:{$lt:30}}]}).pretty()
{
        "_id" : ObjectId("627de4a04686509f257bffd1"),
        "item" : "journal",
        "qty" : 25,
        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd2"),
        "item" : "notebook",
        "qty" : 50,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd5"),
        "item" : "postcard",
        "qty" : 45,
        "size" : {
                "h" : 10,
                "w" : 15.25,
                "uom" : "cm"
        },
        "status" : "A"
}
```

9. Update qty to 25 where item is notebook

```
> db.inventory.updateOne({item:"notebook"},{$set:{qty:25}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.inventory.find().pretty()

{

        "_id" : ObjectId("627de4a04686509f257bffd1"),

        "item" : "journal",

        "qty" : 25,

        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd2"),
        "item" : "notebook",
        "qty" : 25,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd3"),
        "item" : "paper",
        "qty" : 100,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "D"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd4"),
        "item" : "planner",
        "qty" : 75,
        "size" : {
                "h" : 22.85,
                "w" : 30,
                "uom" : "cm"
        },
        "status" : "D"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd5"),
        "item" : "postcard",
        "qty" : 45,
        "size" : {
                "h" : 10,
                "w" : 15.25,
                "uom" : "cm"
        },
        "status" : "A"
}
```

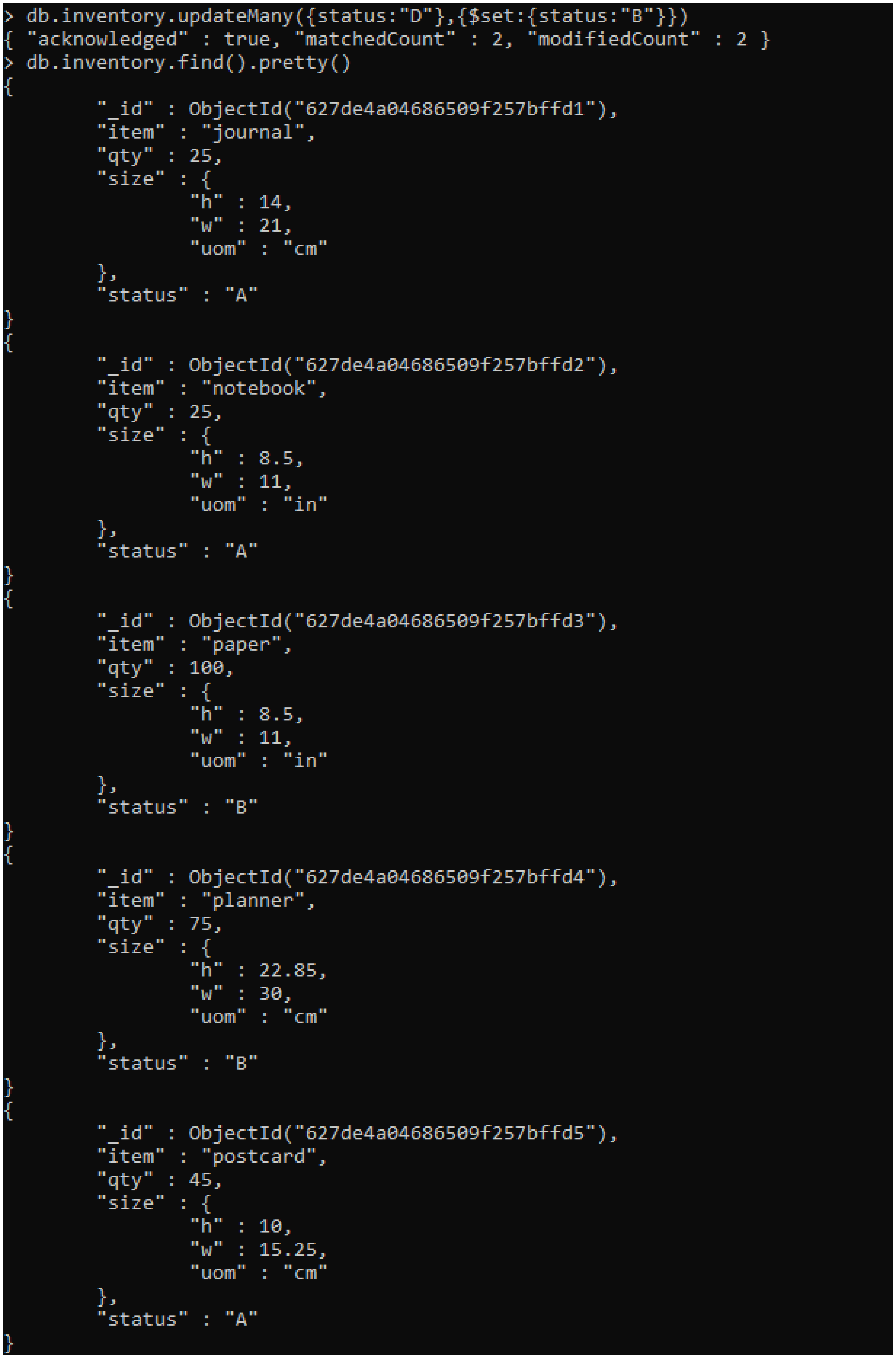


```
> db.inventory.updateOne({item:"notebook"},{$set:{qty:25}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.inventory.updateOne({item:"notebook"},{$set:{qty:25}}).pretty()
uncaught exception: TypeError: db.inventory.updateOne(...).pretty is not a function :
@(shell):1:1
> db.inventory.find().pretty()
{
        "_id" : ObjectId("627de4a04686509f257bffd1"),
        "item" : "journal",
        "qty" : 25,
        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd2"),
        "item" : "notebook",
        "qty" : 25,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd3"),
        "item" : "paper",
        "qty" : 100,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "D"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd4"),
        "item" : "planner",
        "qty" : 75,
        "size" : {
                "h" : 22.85,
                "w" : 30,
                "uom" : "cm"
        },
        "status" : "D"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd5"),
        "item" : "postcard",
        "qty" : 45,
        "size" : {
                "h" : 10,
                "w" : 15.25,
                "uom" : "cm"
        },
        "status" : "A"
}
```

10. Update all status to 'B' where status is 'D'

```
> db.inventory.updateMany({status:"D"},{$set:{status:"B"}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
> db.inventory.find().pretty()
{
        "_id" : ObjectId("627de4a04686509f257bffd1"),
        "item" : "journal",
        "qty" : 25,
        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd2"),
        "item" : "notebook",
        "qty" : 25,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd3"),
        "item" : "paper",
        "qty" : 100,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "B"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd4"),
        "item" : "planner",
        "qty" : 75,
        "size" : {
                "h" : 22.85,
                "w" : 30,
                "uom" : "cm"
        },
        "status" : "B"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd5"),
        "item" : "postcard",
        "qty" : 45,
        "size" : {
```

```
        "h" : 10,
        "w" : 15.25,
        "uom" : "cm"
    },
    "status" : "A"
}
```



```
> db.inventory.updateMany({status:"D"},{$set:{status:"B"}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
> db.inventory.find().pretty()
{
        "_id" : ObjectId("627de4a04686509f257bffd1"),
        "item" : "journal",
        "qty" : 25,
        "size" : {
                "h" : 14,
                "w" : 21,
                "uom" : "cm"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd2"),
        "item" : "notebook",
        "qty" : 25,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "A"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd3"),
        "item" : "paper",
        "qty" : 100,
        "size" : {
                "h" : 8.5,
                "w" : 11,
                "uom" : "in"
        },
        "status" : "B"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd4"),
        "item" : "planner",
        "qty" : 75,
        "size" : {
                "h" : 22.85,
                "w" : 30,
                "uom" : "cm"
        },
        "status" : "B"
}
{
        "_id" : ObjectId("627de4a04686509f257bffd5"),
        "item" : "postcard",
        "qty" : 45,
        "size" : {
                "h" : 10,
                "w" : 15.25,
                "uom" : "cm"
        },
        "status" : "A"
}
```

11. Write a query to demonstrate usage of upsert.

> db.inventory.update({status:"D"},{item:"z"},{upsert:true})

WriteResult({

"nMatched" : 0,

"nUpserted" : 1,

"nModified" : 0,

"_id" : ObjectId("627df05b3f0a383f2c883bc5")

})

> db.inventory.find()

{ "_id" : ObjectId("627de4a04686509f257bffd1"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }

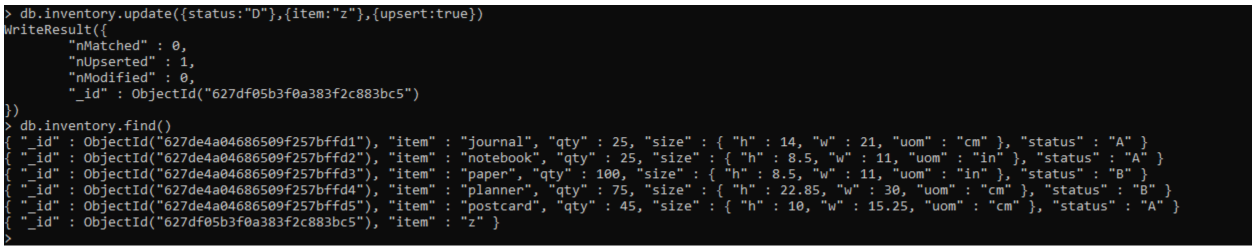
{ "_id" : ObjectId("627de4a04686509f257bffd2"), "item" : "notebook", "qty" : 25, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }

{ "_id" : ObjectId("627de4a04686509f257bffd3"), "item" : "paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "B" }

{ "_id" : ObjectId("627de4a04686509f257bffd4"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "B" }

{ "_id" : ObjectId("627de4a04686509f257bffd5"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }

{ "_id" : ObjectId("627df05b3f0a383f2c883bc5"), "item" : "z" }



```
> db.inventory.update({status:"D"},{item:"z"},{upsert:true})
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("627df05b3f0a383f2c883bc5")
})
> db.inventory.find()
{ "_id" : ObjectId("627de4a04686509f257bffd1"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("627de4a04686509f257bffd2"), "item" : "notebook", "qty" : 25, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }
{ "_id" : ObjectId("627de4a04686509f257bffd3"), "item" : "paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "B" }
{ "_id" : ObjectId("627de4a04686509f257bffd4"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "B" }
{ "_id" : ObjectId("627de4a04686509f257bffd5"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("627df05b3f0a383f2c883bc5"), "item" : "z" }
>
```

12. Delete the document where qty is 100

> db.inventory.deleteOne({qty:100})

{ "acknowledged" : true, "deletedCount" : 0 }

> db.inventory.find()

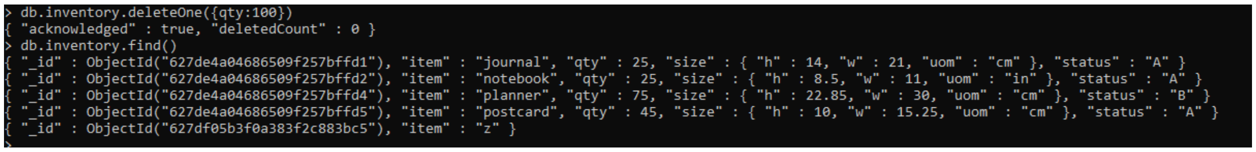
{ "_id" : ObjectId("627de4a04686509f257bffd1"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }

{ "_id" : ObjectId("627de4a04686509f257bffd2"), "item" : "notebook", "qty" : 25, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }

{ "_id" : ObjectId("627de4a04686509f257bffd4"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "B" }

{ "_id" : ObjectId("627de4a04686509f257bffd5"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }

{ "_id" : ObjectId("627df05b3f0a383f2c883bc5"), "item" : "z" }



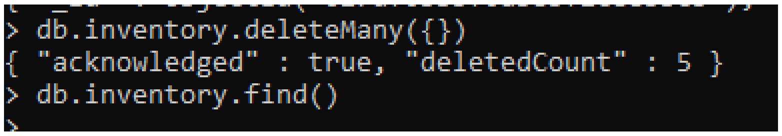
```
> db.inventory.deleteOne({qty:100})
{ "acknowledged" : true, "deletedCount" : 0 }
> db.inventory.find()
{ "_id" : ObjectId("627de4a04686509f257bffd1"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("627de4a04686509f257bffd2"), "item" : "notebook", "qty" : 25, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }
{ "_id" : ObjectId("627de4a04686509f257bffd4"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "B" }
{ "_id" : ObjectId("627de4a04686509f257bffd5"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("627df05b3f0a383f2c883bc5"), "item" : "z" }
>
```

13. Delete all the documents from inventory collection.

> db.inventory.deleteMany({})

{ "acknowledged" : true, "deletedCount" : 5 }

> db.inventory.find()

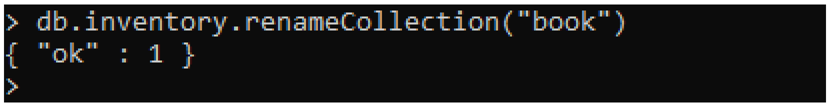


```
> db.inventory.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 5 }
> db.inventory.find()
>
```

14. Rename the collection.

> db.inventory.renameCollection("book")

{ "ok" : 1 }



```
> db.inventory.renameCollection("book")
{ "ok" : 1 }
>
```

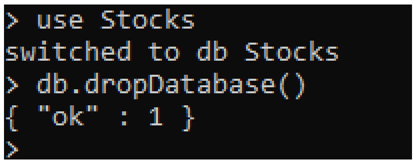
15. Delete the database.

> use Stocks

switched to db Stocks

> db.dropDatabase()

{ "ok" : 1 }



```
> use Stocks
switched to db Stocks
> db.dropDatabase()
{ "ok" : 1 }
>
```