Activity_Bootstrap-4_Portfolio

1. Design a single page portfolio as per the below output using bootstrap 4 flexes.

:: CODE ::

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="../css/bootstrap.min.css">
    <script src="../js/bootstrap.bundle.min.js"></script>
    <title>Portfolio</title>
    <style>
      .test {
      padding: 60px 0px !important;
      }
    </style>
  </head>
  <body>
    <!-- header -->
    <div class="container">
      <div class="row">
        <div class="col-md-4 col-lg-4">
          <img src="../image/bhargav.jpg" alt="John Doe" class="img-thumbnail" height="100">
        </div>
        <div class="col-md-8 col-lg-8">
```

```
<div class="d-flex flex-column">
  <!-- name -->
  <div class="bg-dark d-flex text-white justify-content-between align-items-center p-5">
    <div>
      <h1 class="display-4">John Aderson</h1>
    </div>
    <div><i class="fa fa-facebook"></i></div>
    <div><i class="fa fa-google-plus"></i></div>
    <div><i class="fa fa-linkedin"></i></div>
    <div><i class="fa fa-youtube"></i></div>
  </div>
  <!-- designation -->
  <div class="text-white text p-4" style="background:#000">
    Web and Mobile Developer
  </div>
  <!-- button -->
  <div class="d-flex text-white text-center">
    <div class="bg-success p-4 flex-fill test">
      <i class="fa fa-home"></i>
      <div>Home</div>
    </div>
    <div class="bg-danger p-4 flex-fill test">
      <i class="fa fa-home"></i>
      <div>Resume</div>
    </div>
    <div class="bg-warning p-4 flex-fill test">
      <i class="fa fa-home"></i>
      <div>Work</div>
    </div>
```

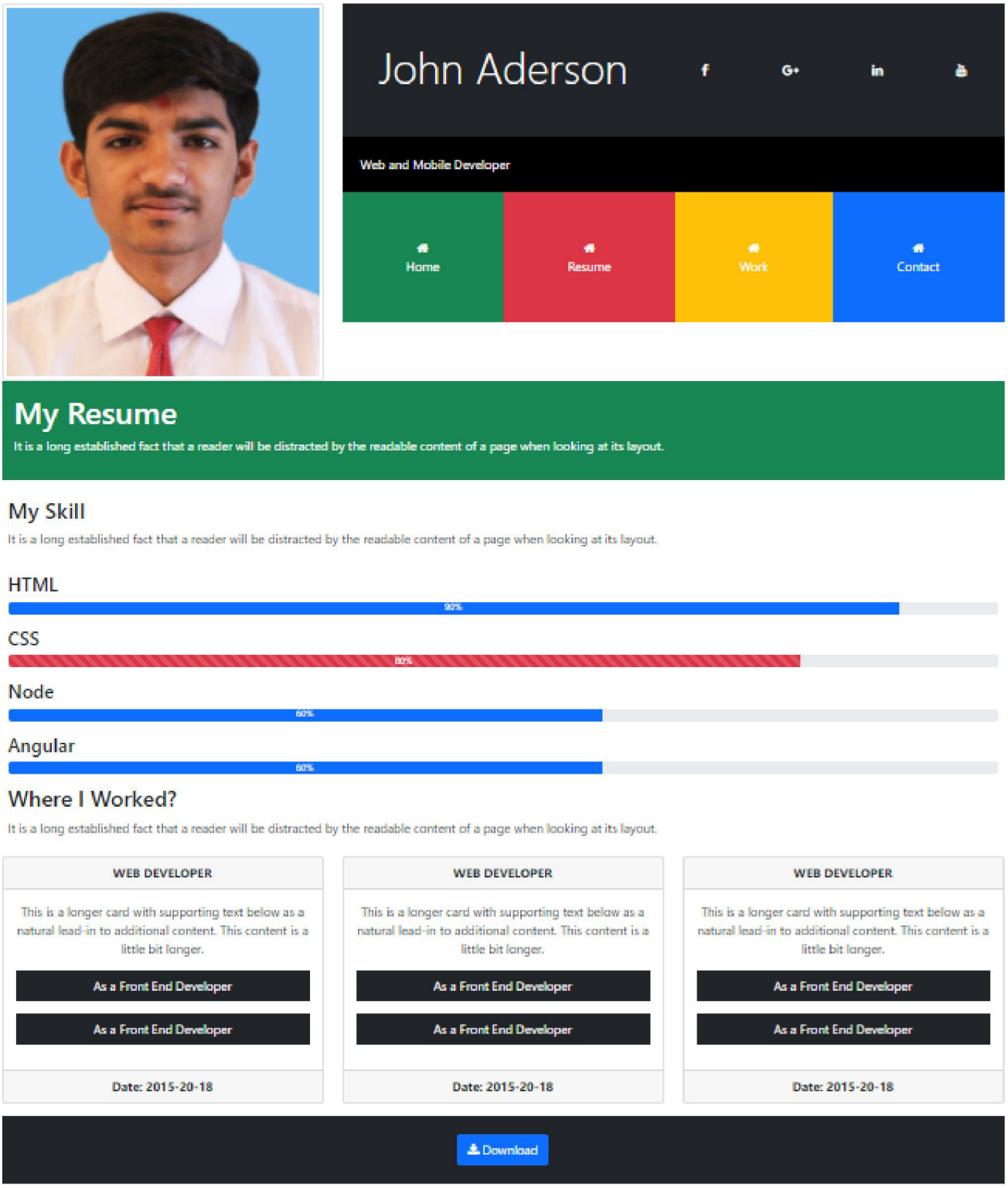
```
                <div class="bg-primary p-4 flex-fill test">
                  <i class="fa fa-home"></i>
                  <div>Contact</div>
                </div>
              </div>
            </div>
          </div>
        </div>
      <div class="container">
        <div class="bg-success p-3 text-white">
          <h1 class="resume">My Resume</h1>
          <p>It is a long established fact that a reader will be distracted by the readable content of a page
when
            looking at its layout.
          </p>
        </div>
        <div class="my_skill mt-3">
          <div class="p-2">
            <h3>My Skill</h3>
            <p>It is a long established fact that a reader will be distracted by the readable content of a page
when
              looking at its layout.
            </p>
          </div>
          <div class="p-2">
            <h4>HTML</h4>
            <div class="progress">
              <div class="progress-bar " role="progressbar" style="width: 90%" aria-valuenow="90"
```

```
                aria-valuemin="0" aria-valuemax="100">90%</div>
            </div>
          </div>
          <div class="p-2">
            <h4>CSS</h4>
            <div class="progress">
              <div class="progress-bar progress-bar-striped bg-danger" role="progressbar" style="width:
80%"
                aria-valuenow="25" aria-valuemin="0" aria-valuemax="100">80%</div>
            </div>
          </div>
          <div class="p-2">
            <h4>Node</h4>
            <div class="progress">
              <div class="progress-bar bg-primary" role="progressbar" style="width: 60%" aria-
valuenow="50"
                aria-valuemin="0" aria-valuemax="100">60%</div>
            </div>
          </div>
          <div class="p-2">
            <h4>Angular</h4>
            <div class="progress">
              <div class="progress-bar  bg-primary" role="progressbar" style="width: 60%" aria-
valuenow="75"
                aria-valuemin="0" aria-valuemax="100">60%</div>
            </div>
          </div>
        </div>
      </div>
      <div class="container">
```

```
<div class="p-2">
  <h3>Where I Worked?</h3>
  <p>It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. </p>
</div>
<div class="worked">
  <div class="row row-cols-1 row-cols-md-3 g-4">
    <div class="col">
      <div class="card  mb-3">
        <div class="card-header text-center"><strong>WEB DEVELOPER</strong></div>
        <div class="card-body text-center">
          <p class="card-text text-center">This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer. </p>
          <p class=" bg-dark p-2 text-white">As a Front End Developer</p>
          <p class=" bg-dark p-2 text-white">As a Front End Developer</p>
        </div>
        <div class="card-footer text-center"><strong>Date: 2015-20-18</strong></div>
      </div>
    </div>
    <div class="col">
      <div class="card  mb-3">
        <div class="card-header text-center"><strong>WEB DEVELOPER</strong></div>
        <div class="card-body text-center">
          <p class="card-text text-center">This is a longer card with supporting text below as a natural lead-in
            to additional content. This content is a little bit longer.
          </p>
          <p class=" bg-dark p-2 text-white">As a Front End Developer</p>
          <p class=" bg-dark p-2 text-white">As a Front End Developer</p>
        </div>
```

```
            <div class="card-footer text-center"><strong>Date: 2015-20-18</strong></div>
          </div>
        </div>
        <div class="col">
          <div class="card mb-3 text-center">
            <div class="card-header text-center"><strong>WEB DEVELOPER</strong></div>
            <div class="card-body">
              <p class="card-text text-center">This is a longer card with supporting text below as a
natural lead-in
                to additional content. This content is a little bit longer.
              </p>
              <p class=" bg-dark p-2 text-white">As a Front End Developer</p>
              <p class=" bg-dark p-2 text-white">As a Front End Developer</p>
            </div>
            <div class="card-footer text-center"><strong>Date: 2015-20-18</strong></div>
          </div>
        </div>
      </div>
    </div>
    <div class="container">
      <div class="download p-4 mb-5 bg-dark text-center">
        <button class="btn btn-primary"><i class="fa fa-download"></i> Download</button>
      </div>
    </div>
  </body>
</html>
```

:: OUTPUT ::


John Aderson
Web and Mobile Developer
Home
Resume
Work
Contact
My Resume
It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout.
My Skill
It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout.
HTML
90%
CSS
80%
Node
60%
Angular
60%
Where I Worked?
It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout.
WEB DEVELOPER
This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.
As a Front End Developer
As a Front End Developer
Date: 2015-20-18
WEB DEVELOPER
This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.
As a Front End Developer
As a Front End Developer
Date: 2015-20-18
WEB DEVELOPER
This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.
As a Front End Developer
As a Front End Developer
Date: 2015-20-18
Download

Hands on for Querying Documents

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

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

```
> db.StudentData.insertMany([{name:"Kishan",branch:"CE",gender:"Male",status:"A"},{
name:"Nishnt",branch:"CE",gender:"Male",status:"A"},{name:"Trupal",branch:"CV",gend
er:"Male",status:"D"},{name:"Manan",branch:"IT",gender:"Male",status:"A"},{name:"Ut
sav",branch:"CE",gender:"Male",status:"D"}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61f29f2cd0a69a0e664f2b84"),
                ObjectId("61f29f2cd0a69a0e664f2b85"),
                ObjectId("61f29f2cd0a69a0e664f2b86"),
                ObjectId("61f29f2cd0a69a0e664f2b87"),
                ObjectId("61f29f2cd0a69a0e664f2b88")
        ]
}
```

1. Find all the result of the given collection.

   CODE : AND : OUTPUT :

```
> db.StudentData.find()
{ "_id" : ObjectId("61f29f2cd0a69a0e664f2b84"), "name" : "Kishan", "branch" : "CE",
 "gender" : "Male", "status" : "A" }
{ "_id" : ObjectId("61f29f2cd0a69a0e664f2b85"), "name" : "Nishnt", "branch" : "CE",
 "gender" : "Male", "status" : "A" }
{ "_id" : ObjectId("61f29f2cd0a69a0e664f2b86"), "name" : "Trupal", "branch" : "CV",
 "gender" : "Male", "status" : "D" }
{ "_id" : ObjectId("61f29f2cd0a69a0e664f2b87"), "name" : "Manan", "branch" : "IT",
"gender" : "Male", "status" : "A" }
{ "_id" : ObjectId("61f29f2cd0a69a0e664f2b88"), "name" : "Utsav", "branch" : "CE",
"gender" : "Male", "status" : "D" }
```

2. Show the result in pretty format?

CODE : AND : OUTPUT :

```
> db.StudentData.find().pretty()
{
        "_id" : ObjectId("61f29f2cd0a69a0e664f2b84"),
        "name" : "Kishan",
        "branch" : "CE",
        "gender" : "Male",
        "status" : "A"
}
{
        "_id" : ObjectId("61f29f2cd0a69a0e664f2b85"),
        "name" : "Nishnt",
        "branch" : "CE",
        "gender" : "Male",
        "status" : "A"
}
{
        "_id" : ObjectId("61f29f2cd0a69a0e664f2b86"),
        "name" : "Trupal",
        "branch" : "CV",
        "gender" : "Male",
        "status" : "D"
}
{
        "_id" : ObjectId("61f29f2cd0a69a0e664f2b87"),
        "name" : "Manan",
        "branch" : "IT",
        "gender" : "Male",
        "status" : "A"
}
{
        "_id" : ObjectId("61f29f2cd0a69a0e664f2b88"),
        "name" : "Utsav",
        "branch" : "CE",
        "gender" : "Male",
        "status" : "D"
}
```

3. Get only CE data as a output.

CODE : AND : OUTPUT :

```
> db.StudentData.find({branch:"CE"}).pretty()
{
        "_id" : ObjectId("61f29f2cd0a69a0e664f2b84"),
        "name" : "Kishan",
        "branch" : "CE",
        "gender" : "Male",
        "status" : "A"
}
{
        "_id" : ObjectId("61f29f2cd0a69a0e664f2b85"),
        "name" : "Nishnt",
        "branch" : "CE",
        "gender" : "Male",
        "status" : "A"
}
{
        "_id" : ObjectId("61f29f2cd0a69a0e664f2b88"),
        "name" : "Utsav",
        "branch" : "CE",
        "gender" : "Male",
        "status" : "D"
}
```

4. Get only CE data as a output with only name field.

CODE : AND : OUTPUT :

```
> db.StudentData.find({branch:"CE"},{name:1}).pretty()
{ "_id" : ObjectId("61f29f2cd0a69a0e664f2b84"), "name" : "Kishan" }
{ "_id" : ObjectId("61f29f2cd0a69a0e664f2b85"), "name" : "Nishnt" }
{ "_id" : ObjectId("61f29f2cd0a69a0e664f2b88"), "name" : "Utsav" }
```

5. Get the CE data without _ID field in it.

CODE : AND : OUTPUT :

```
> db.StudentData.find({branch:"CE"},{name:1,_id:0}).pretty()
{ "name" : "Kishan" }
{ "name" : "Nishnt" }
{ "name" : "Utsav" }
```

6. set the filter to “active : A” and get only the first field with “active : A” value.

CODE : AND : OUTPUT :

```
> db.StudentData.find({status:"A"}).limit(1).pretty()
{
        "_id" : ObjectId("61f29f2cd0a69a0e664f2b84"),
        "name" : "Kishan",
        "branch" : "CE",
        "gender" : "Male",
        "status" : "A"
}
```

7. Do the same as 6 question but with different method.

CODE : AND : OUTPUT :

```
> db.StudentData.findOne({status:"A"})
{
        "_id" : ObjectId("61f29f2cd0a69a0e664f2b84"),
        "name" : "Kishan",
        "branch" : "CE",
        "gender" : "Male",
        "status" : "A"
}
```

8. Do the same as 6 question but this time, get the 2nd field with active : A by skipping the 1st field.

CODE : AND : OUTPUT :

```
> db.StudentData.find({status:"A"}).limit(1).skip(1).pretty()
{
        "_id" : ObjectId("61f29f2cd0a69a0e664f2b85"),
        "name" : "Nishnt",
        "branch" : "CE",
        "gender" : "Male",
        "status" : "A"
}
```

Hands on with Aggregation & Grouping

1. List the category of books.

   CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : "$Category"}}])
{ "_id" : "programming" }
{ "_id" : "history" }
{ "_id" : "gk" }
{ "_id" : "fiction" }
{ "_id" : "novel" }
>
```

2. Find the total count of documents for each category.

   CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : "$Category", "Total Count" : {$sum : 1}}}]).pretty()
{ "_id" : "programming", "Total Count" : 2 }
{ "_id" : "fiction", "Total Count" : 1 }
{ "_id" : "history", "Total Count" : 2 }
{ "_id" : "gk", "Total Count" : 1 }
{ "_id" : "novel", "Total Count" : 1 }
>
```

3. Find the documents having copies greater than 30 according to the categories.

   CODE : AND : OUTPUT :

```
> db.books.aggregate([{$match : {qty : {$gt : 30}}},{$group : {_id : "$Category"}}])
{ "_id" : "programming" }
{ "_id" : "history" }
{ "_id" : "gk" }
{ "_id" : "fiction" }
>
```

4. Find the documents by grouping according to isbn number and sort by the same.

   CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : "$ISBN"}},{$sort : {_id : 1}}])
{ "_id" : 1234 }
{ "_id" : 12344 }
{ "_id" : 12354 }
{ "_id" : 32145 }
>
```

5. Find the documents by grouping to categories and sorting it by category name only for those documents having available status true.

CODE : AND : OUTPUT :

```
> db.books.aggregate([{$match : {status : "true"}},{$group : {_id : "$Category"}}
,{$sort : {_id : 1}}])
{ "_id" : "fiction" }
{ "_id" : "gk" }
{ "_id" : "history" }
{ "_id" : "novel" }
{ "_id" : "programming" }
```

6. Find the average no. of books based on the categories.

CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : "$Category", "Average" : {$avg : "$qty"}}}])
{ "_id" : "programming", "Average" : 36.5 }
{ "_id" : "fiction", "Average" : 45 }
{ "_id" : "history", "Average" : 37.5 }
{ "_id" : "gk", "Average" : 50 }
{ "_id" : "novel", "Average" : 25 }
>
```

7. Find the min and max no. of books based on the categories.

CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : "$Category", "Minimum" : {$min : "$qty"},
"Maximum" : {$max : "$qty"}}}])
{ "_id" : "programming", "Minimum" : 30, "Maximum" : 43 }
{ "_id" : "fiction", "Minimum" : 45, "Maximum" : 45 }
{ "_id" : "history", "Minimum" : 35, "Maximum" : 40 }
{ "_id" : "gk", "Minimum" : 50, "Maximum" : 50 }
{ "_id" : "novel", "Minimum" : 25, "Maximum" : 25 }
```

8. Find the total count of documents for each category.

CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : "$Category", "Total Count" : {$sum : 1}}}]).pretty()
{ "_id" : "programming", "Total Count" : 2 }
{ "_id" : "fiction", "Total Count" : 1 }
{ "_id" : "history", "Total Count" : 2 }
{ "_id" : "gk", "Total Count" : 1 }
{ "_id" : "novel", "Total Count" : 1 }
```

9. Find the total count of not available books for each category.

CODE : AND : OUTPUT :

```
> db.books.aggregate([{$match : {status : "NA"}},{$group : {_id : "$Category",
"Total Count" : {$sum : 1}}}])
{ "_id" : "programming", "Total Count" : 1 }
{ "_id" : "history", "Total Count" : 2 }
>
```

10. Find the total no. of copies of books for each category.

CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : "$Category", "Total Copies" : {$sum : "$qty"}}}])
{ "_id" : "programming", "Total Copies" : 73 }
{ "_id" : "history", "Total Copies" : 75 }
{ "_id" : "gk", "Total Copies" : 50 }
{ "_id" : "fiction", "Total Copies" : 45 }
{ "_id" : "novel", "Total Copies" : 25 }
```

11. Sort the above output in descending order of copies.

CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : "$Category", "Total Copies" : {$sum : "$qty"}}},{$sort :
 {"Total Copies" : -1}}])
{ "_id" : "history", "Total Copies" : 75 }
{ "_id" : "programming", "Total Copies" : 73 }
{ "_id" : "gk", "Total Copies" : 50 }
{ "_id" : "fiction", "Total Copies" : 45 }
{ "_id" : "novel", "Total Copies" : 25 }
```

12. Find the documents by grouping category and isbn number along with the count of each group.

CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : ["$Category" , "$ISBN"], "Total Count" :
 {$sum : 1}}}])
{ "_id" : [ "fiction", 12344 ], "Total Count" : 1 }
{ "_id" : [ "gk", 1234 ], "Total Count" : 1 }
{ "_id" : [ "programming", 1234 ], "Total Count" : 1 }
{ "_id" : [ "history", 1234 ], "Total Count" : 1 }
{ "_id" : [ "programming", 12354 ], "Total Count" : 1 }
{ "_id" : [ "novel", 1234 ], "Total Count" : 1 }
{ "_id" : [ "history", 32145 ], "Total Count" : 1 }
```

13. Find the documents by grouping category and isbn number along with the count of each group.

    But the output should be displayed as follows:

    Expected output:

    { "Category" : "history", "ISBN" : 32145, "total_docs" : 1 }

    { "Category" : "history", "ISBN" : 1234, "total_docs" : 2 }

    { "Category" : "fiction", "ISBN" : 1234, "total_docs" : 1 }

    { "Category" : "programming", "ISBN" : 1234, "total_docs" : 1 }

    { "Category" : "programming", "ISBN" : 12354, "total_docs" : 1 }

    { "Category" : "gk", "ISBN" : 1234, "total_docs" : 1 }

    { "Category" : "novel", "ISBN" : 1234, "total_docs" : 1 }

    CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : {"Category" : "$Category" , "ISBN" : "$
ISBN", "Total Count" : {$sum : 1}}}},{$project : {Category : "$_id.Category",
ISBN : "$_id.ISBN", "Total Count" : "$_id.Total Count" , _id : 0}}])
{ "Category" : "fiction", "ISBN" : 12344, "Total Count" : 1 }
{ "Category" : "programming", "ISBN" : 1234, "Total Count" : 1 }
{ "Category" : "history", "ISBN" : 1234, "Total Count" : 1 }
{ "Category" : "programming", "ISBN" : 12354, "Total Count" : 1 }
{ "Category" : "history", "ISBN" : 32145, "Total Count" : 1 }
{ "Category" : "gk", "ISBN" : 1234, "Total Count" : 1 }
{ "Category" : "novel", "ISBN" : 1234, "Total Count" : 1 }
```

14. Find the documents by grouping category and isbn number along with the total no of copies for each group. Display the group having highest copies.

    Expected output:

    { "Category" : "history", "ISBN" : 32145, "total_copies" : 50 }

    CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : {"Category" : "$Category" , "ISBN" : "$ISBN", "Total
Copies" : {$sum : "$qty"}}}},{$project : {Category : "$_id.Category", ISBN : "$_id.ISBN", "T
otal Copies" : "$_id.Total Copies" , _id : 0}},{$sort : {"Total Copies" : -1}},{$limit : 1}]
)
{ "Category" : "gk", "ISBN" : 1234, "Total Copies" : 50 }
>
```

Hands on with Aggregation & Match

1. Find the documents having copies greater than 30 according to the categories.

   CODE : AND : OUTPUT :

```
> db.books.aggregate([{$match : {qty : {$gt : 30}}},{$group : {_id : "$Category"}}])
{ "_id" : "programming" }
{ "_id" : "history" }
{ "_id" : "gk" }
{ "_id" : "fiction" }
>
```

2. Find the documents by grouping according to isbn number and sort by the same.

   CODE : AND : OUTPUT :

```
> db.books.aggregate([{$group : {_id : "$ISBN"}},{$sort : {_id : 1}}])
{ "_id" : 1234 }
{ "_id" : 12344 }
{ "_id" : 12354 }
{ "_id" : 32145 }
>
```

3. Find the documents by grouping to categories and sorting it by category name only for those documents having available status true.

   CODE : AND : OUTPUT :

```
> db.books.aggregate([{$match : {status : "true"}},{$group : {_id : "$Category"}}
,{$sort : {_id : 1}}])
{ "_id" : "fiction" }
{ "_id" : "gk" }
{ "_id" : "history" }
{ "_id" : "novel" }
{ "_id" : "programming" }
```

4. Find the total count of not available books for each category.

   CODE : AND : OUTPUT :

```
> db.books.aggregate([{$match : {status : "NA"}},{$group : {_id : "$Category",
"Total Count" : {$sum : 1}}}])
{ "_id" : "programming", "Total Count" : 1 }
{ "_id" : "history", "Total Count" : 2 }
>
```

Hands on with Querying using Regex Expression

```
> use reg
switched to db reg
> db.createCollection("products")
{ "ok" : 1 }
```

```
> db.products.insertMany([{_id:100,sku:"abc123",description:"Single line descriptio
n."},{_id:101,sku:"abc789",description:"Frist line\nSecond line"},{_id:102,sku:"xyz
456",description:"Many spaces before line"},{_id:103,sku:"XYZ789",description:"Mult
ipal\nline description"}])
{ "acknowledged" : true, "insertedIds" : [ 100, 101, 102, 103 ] }
```

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

   CODE : AND : OUTPUT :

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

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

   CODE : AND : OUTPUT :

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

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

   CODE : AND : OUTPUT :

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

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

   CODE : AND : OUTPUT :

```
> db.products.find({description:{"$regex":/^.i/}})
{ "_id" : 100, "sku" : "abc123", "description" : "Single line description." }
```

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

CODE : AND : OUTPUT :

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

6. Find the documents where any line from the description starts with ‘S’.

CODE : AND : OUTPUT :

```
> db.products.find({description:{"$regex":/^S/,$options:"m"}})
{ "_id" : 100, "sku" : "abc123", "description" : "Single line description." }
{ "_id" : 101, "sku" : "abc789", "description" : "Frist line\nSecond line" }
```

Hands-On for Querying with filters

1. Create a database named "Stocks".

   CODE : AND : OUTPUT :

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

2. Create an empty collection named "inventory".

   CODE : AND : OUTPUT :

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

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

   CODE : AND : OUTPUT :

```
> 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",q
ty:100,size:{h:8.5,w:11,uom:"in"},status:"D"},{item:"planner",qty:75,size:{h:22.85,
w:30,uon:"cm"},status:"D"},{item:"postcard",qty:45,size:{h:10,w:15.25,uom:"cm"},sta
tus:"A"}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61e6a7f68c3bafd9b0ecc6db"),
                ObjectId("61e6a7f68c3bafd9b0ecc6dc"),
                ObjectId("61e6a7f68c3bafd9b0ecc6dd"),
                ObjectId("61e6a7f68c3bafd9b0ecc6de"),
                ObjectId("61e6a7f68c3bafd9b0ecc6df")
        ]
}
```

4. Select all documents from inventory collection.

   CODE : AND : OUTPUT :

```
> db.inventory.find()
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6db"), "item" : "journal", "qty" : 25, "si
ze" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6dc"), "item" : "notebook", "qty" : 50, "s
ize" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6dd"), "item" : "paper", "qty" : 100, "siz
e" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "D" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6de"), "item" : "planner", "qty" : 75, "si
ze" : { "h" : 22.85, "w" : 30, "uon" : "cm" }, "status" : "D" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6df"), "item" : "postcard", "qty" : 45, "s
ize" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
```

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

CODE : AND : OUTPUT :

```
> db.inventory.find({status:"D"})
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6dd"), "item" : "paper", "qty" : 100, "siz
e" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "D" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6de"), "item" : "planner", "qty" : 75, "si
ze" : { "h" : 22.85, "w" : 30, "uon" : "cm" }, "status" : "D" }
```

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

CODE : AND : OUTPUT :

```
> db.inventory.find({status:{$in:["A","D"]}})
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6db"), "item" : "journal", "qty" : 25, "si
ze" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6dc"), "item" : "notebook", "qty" : 50, "s
ize" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6dd"), "item" : "paper", "qty" : 100, "siz
e" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "D" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6de"), "item" : "planner", "qty" : 75, "si
ze" : { "h" : 22.85, "w" : 30, "uon" : "cm" }, "status" : "D" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6df"), "item" : "postcard", "qty" : 45, "s
ize" : { "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.

CODE : AND : OUTPUT :

```
> db.inventory.find({$and:[{status:"A"},{qty:{$lt:30}}]})
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6db"), "item" : "journal", "qty" : 25, "si
ze" : { "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.

CODE : AND : OUTPUT :

```
> db.inventory.find({$or:[{status:"A"},{qty:{$lt:30}}]})
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6db"), "item" : "journal", "qty" : 25, "si
ze" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6dc"), "item" : "notebook", "qty" : 50, "s
ize" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6df"), "item" : "postcard", "qty" : 45, "s
ize" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
```

9. Update qty to 25 where item is notebook

CODE : AND : OUTPUT :

```
> db.inventory.updateOne({item:"notebook"},{$set:{qty:25}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.inventory.find()
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6db"), "item" : "journal", "qty" : 25, "si
ze" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6dc"), "item" : "notebook", "qty" : 25, "s
ize" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6dd"), "item" : "paper", "qty" : 100, "siz
e" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "D" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6de"), "item" : "planner", "qty" : 75, "si
ze" : { "h" : 22.85, "w" : 30, "uon" : "cm" }, "status" : "D" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6df"), "item" : "postcard", "qty" : 45, "s
ize" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
```

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

CODE : AND : OUTPUT :

```
> db.inventory.updateMany({status:"D"},{$set:{status:"B"}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
> db.inventory.find()
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6db"), "item" : "journal", "qty" : 25, "si
ze" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6dc"), "item" : "notebook", "qty" : 25, "s
ize" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "A" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6dd"), "item" : "paper", "qty" : 100, "siz
e" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "B" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6de"), "item" : "planner", "qty" : 75, "si
ze" : { "h" : 22.85, "w" : 30, "uon" : "cm" }, "status" : "B" }
{ "_id" : ObjectId("61e6a7f68c3bafd9b0ecc6df"), "item" : "postcard", "qty" : 45, "s
ize" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "status" : "A" }
```

11. Write a query to demonstrate usage of upsert.

CODE : AND : OUTPUT :

```
> db.inventory.updateMany({item:"pen"},{$set:{status:"B"}},{upsert:true})
{
        "acknowledged" : true,
        "matchedCount" : 0,
        "modifiedCount" : 0,
        "upsertedId" : ObjectId("61e6d3ef96675fcbd7e48582")
}
> db.inventory.find({item:"pen"})
{ "_id" : ObjectId("61e6d3ef96675fcbd7e48582"), "item" : "pen", "status" : "B" }
```

12. Delete the document where qty is 100

CODE : AND : OUTPUT :

```
> db.inventory.deleteMany({qty:100})
{ "acknowledged" : true, "deletedCount" : 1 }
```

13. Delete all the documents from inventory collection.

CODE : AND : OUTPUT :

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

14. Rename the collection.

CODE : AND : OUTPUT :

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

15. Delete the database.

CODE : AND : OUTPUT :

```
> db.dropDatabase()
{ "dropped" : "Stocks", "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

Hands-on with Node.js

1. Install Nodejs to set up the project.

   Follow the step : ➔

   Step-1 : Frist off all you can download a NodeJS set up file from the browser follow this link : Download | Node.js (nodejs.org) or **https://nodejs.org/dist/v14.15.4/node-v14.15.4-x64.msi**

   Step-2 : Ones download this file than install the NodeJS in your system and than after set environment variable path as a globally from any location.

   EX → C:\Program Files\MongoDB\Server\4.4\bin

2. Create a function expression in node js to count the number of items in an array.

   CODE ::

   ```
   var arr = [10,20,30,40,50];

   console.log(arr.length)
   ```

   :: OUTPUT ::

   ```
   PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\Hands on with No
   de.js> node count
   5
   ```

3. Create a module that contains three functions and export the same using nodejs.

   CODE ::

   ```
   var sum = (a, b) => {
      return a + b;
   }
   var print = (s) => {
      return s;
   }
   var sqr = (a) => {
      return a * a;
   ```

```
}
console.log("Let's Start..!")
module.exports = {
   sum,
   print,
   sqr
};
```

:: OUTPUT ::

```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\Hands on with No
de.js> node function_export
Let's Start..!
```

4. Create a module to import the above created function and utilize it.

CODE ::

```
var fun = require('./function_export')
console.log(fun.sum(9, 17));
console.log(fun.print("Welcome to Node...!!!"));
console.log(fun.sqr(7));
```

:: OUTPUT ::

```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\Hands on with No
de.js> node function_import
Let's Start..!
26
Welcome to Node...!!!
49
```

5. Create a server using http module and listen to port 3000.

CODE ::

```
var http = require('http')
var server = http.createServer((function(req, res) {
   res.writeHead(200, {
      "Content-Type": "text/html"
```

```
    });
    res.write("Welcome to Http Module...!!!")
    res.end();
})).listen(3000);

console.log("Server Started...!!!");
```

:: OUTPUT ::

```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\Hands on with No
de.js> node http_server
Server Started...!!!
```

6. Display three different users(Student, admin and default) html pages as per the requested url.

CODE ::

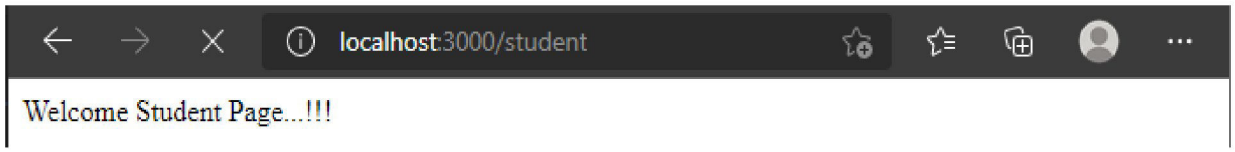
```
var http = require('http')
var server = http.createServer((function(req, res) {
    if (req.url == '/') {
        res.writeHead(200, {
            "Content-Type": "text/html"
        });
        res.write("Welcome Default Page...!!!")
    } else if (req.url == '/student') {
        res.writeHead(200, {
            "Content-Type": "text/html"
        });
        res.write("Welcome Student Page...!!!")
    } else {
        res.writeHead(200, {
            "Content-Type": "text/html"
        });
        res.write("Welcome Admin Page...!!!")
```

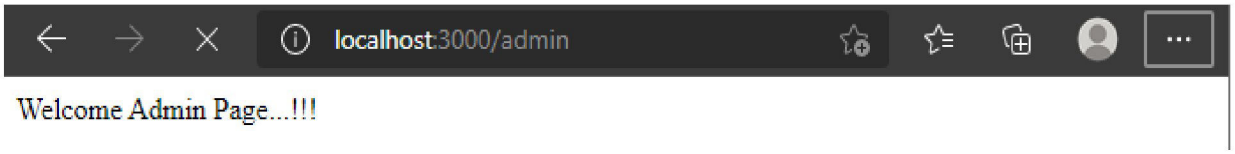
```
    }
})).listen(3000);


console.log("Server Started...!!!");
```

:: OUTPUT ::

```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\Hands on with Node.js> node http_role_user
Server Started...!!!
```





7. Create and import the class type object to display the full name of the user.

CODE ::

name.js

```
var name = {
    fname: "Bhargav",
    lname: "Vasani"
};
module.exports = name;
```

import_class.js

```
var http = require('http')
var result = require('./name')
var url = require('url')
```

```
var querystring = require('querystring')

var server = http.createServer((function(req, res) {
   var query = url.parse(req.url).query;
   var fname = querystring.parse(query)['fname'];
   var lname = querystring.parse(query)['lname'];

   console.log(result.fname, result.lname)
   console.log("First name", result.fname)
   console.log("Last name", result.lname)

})).listen(3000);

console.log("Server Started...!!!");
```

:: OUTPUT ::

```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\Hands on with Node.js> node import_class
Server Started...!!!
Bhargav Vasani
First name Bhargav
Last name Vasani
```

Practice Basic CRUD

1. Create a database named "Students".

   CODE : AND : OUTPUT :

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

2. Create an empty collection named "studentData".

   CODE : AND : OUTPUT :

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

3. Insert only one record with appropriate fields.

   CODE : AND : OUTPUT :

```
> db.studentData.insert({name:"Bhargav",branch:"CE",gender:"Male"})
WriteResult({ "nInserted" : 1 })
```

4. Try to insert 5 records together using single query.

   CODE : AND : OUTPUT :

```
> db.studentData.insert([{name:"Kishan",branch:"CE",gender:"Male"},{name:"nishant",
branch:"CE",gender:"Male"},{name:"Trupal",branch:"IT",gender:"Male"},{name:"Manan",
branch:"IT",gender:"Male"},{name:"Utsav",branch:"CE",gender:"Male"}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 5,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

5. Fetch all the documents in formatted manner.

CODE : AND : OUTPUT :

```
> db.studentData.find().pretty()
{
        "_id" : ObjectId("61e69f6e064d610aaaaa4964"),
        "name" : "Bhargav",
        "branch" : "CE",
        "gender" : "Male"
}
{
        "_id" : ObjectId("61e6a1fc064d610aaaaa4965"),
        "name" : "Kishan",
        "branch" : "CE",
        "gender" : "Male"
}
{
        "_id" : ObjectId("61e6a1fc064d610aaaaa4966"),
        "name" : "nishant",
        "branch" : "CE",
        "gender" : "Male"
}
{
        "_id" : ObjectId("61e6a1fc064d610aaaaa4967"),
        "name" : "Trupal",
        "branch" : "IT",
        "gender" : "Male"
}
{
        "_id" : ObjectId("61e6a1fc064d610aaaaa4968"),
        "name" : "Manan",
        "branch" : "IT",
        "gender" : "Male"
}
{
        "_id" : ObjectId("61e6a1fc064d610aaaaa4969"),
        "name" : "Utsav",
        "branch" : "CE",
        "gender" : "Male"
}
```

6. Fetch the document based on the filter criteria.

CODE : AND : OUTPUT :

```
> db.studentData.find({branch:"IT"})
{ "_id" : ObjectId("61e6a1fc064d610aaaaa4967"), "name" : "Trupal", "branch" : "IT",
 "gender" : "Male" }
{ "_id" : ObjectId("61e6a1fc064d610aaaaa4968"), "name" : "Manan", "branch" : "IT",
"gender" : "Male" }
```

7. Delete one document from the "studentData".

CODE : AND : OUTPUT :

```
> db.studentData.deleteOne({name:"Kishan"})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.studentData.find().pretty()
{
        "_id" : ObjectId("61e69f6e064d610aaaaa4964"),
        "name" : "Bhargav",
        "branch" : "CE",
        "gender" : "Male"
}
{
        "_id" : ObjectId("61e6a1fc064d610aaaaa4966"),
        "name" : "nishant",
        "branch" : "CE",
        "gender" : "Male"
}
{
        "_id" : ObjectId("61e6a1fc064d610aaaaa4967"),
        "name" : "Trupal",
        "branch" : "IT",
        "gender" : "Male"
}
{
        "_id" : ObjectId("61e6a1fc064d610aaaaa4968"),
        "name" : "Manan",
        "branch" : "IT",
        "gender" : "Male"
}
{
        "_id" : ObjectId("61e6a1fc064d610aaaaa4969"),
        "name" : "Utsav",
        "branch" : "CE",
        "gender" : "Male"
}
```

8. Delete the documents based on the filter criteria.

CODE : AND : OUTPUT :

```
> db.studentData.deleteMany({branch:"CE"})
{ "acknowledged" : true, "deletedCount" : 3 }
> db.studentData.find().pretty()
{
        "_id" : ObjectId("61e6a1fc064d610aaaaa4967"),
        "name" : "Trupal",
        "branch" : "IT",
        "gender" : "Male"
}
{
        "_id" : ObjectId("61e6a1fc064d610aaaaa4968"),
        "name" : "Manan",
        "branch" : "IT",
        "gender" : "Male"
}
```