Emerging Technologies

 Journal 2025-2026 

Index

|  |  |  |  |
| --- | --- | --- | --- |
| Practical No | Details | Date | Signature |
| **1** | **MongoDB Basics** |  |  |
| a | Write a MongoDB query to create and drop database. |  |  |
| b | Write a MongoDB query to create, display and drop collection |  |  |
| c | Write a MongoDB query to insert, query, update and delete a document. |  |  |
|  |  |  |  |
| **2** | **Simple Queries with MongoDB** |  |  |
|  |  |  |  |
| **3** | **Implementing Aggregation** |  |  |
| a | Write a MongoDB query to use sum, avg, min and max expression. |  |  |
| b | Write a MongoDB query to use push and addToSet expression. |  |  |
| c | Write a MongoDB query to use first and last expression. |  |  |
|  |  |  |  |
| **4** | **Replication, Backup and Restore** |  |  |
| a | Write a MongoDB query to create Replica of existing database. |  |  |
| b | Write a MongoDB query to create a backup of existing database. |  |  |
| c | Write a MongoDB query to restore database from the backup. |  |  |
|  |  |  |  |
| **5** | **Java and MongoDB** |  |  |
| a | Connecting Java with MongoDB and inserting, retrieving, updating and deleting. |  |  |
| **6** | **Python and MongoDB** |  |  |
| a | Connecting Python with MongoDB and inserting, retrieving, updating and deleting. |  |  |
|  |  |  |  |
| **7** | **Programs on Basic jQuery** |  |  |
| a | Query Basie, jQuery Events. |  |  |
| b | Query Selectors, jQuery Hide and Show effects. |  |  |
| c | Query fading effects. Query Sliding effects. |  |  |
|  |  |  |  |
| **8** | **jQuery Advanced** |  |  |
| a | jQuery Animation effects, jQuery Chaining |  |  |
| b | jQuery Callback, jQuery Get and Set Contents |  |  |
| c | jQuery Insert Content, jQuery Remove Elements and Attribute |  |  |
|  |  |  |  |
| **9** | **JSON** |  |  |
| a | Creating JSON |  |  |
| b | Parsing JSON |  |  |
| c | Persisting JSON |  |  |
|  |  |  |  |
| **10** | **Create a JSON file and import it to MongoDB** |  |  |
| a | Export MongoDB to JSON. |  |  |
| b | Write a MongoDB query to delete JSON object from MongoDB |  |  |

# **Practical No:1 – MongoDB Basics**

1. **Write a MongoDB query to create and drop database.**

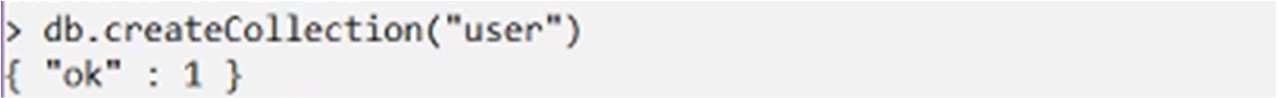
> show databases // checks currents databases.



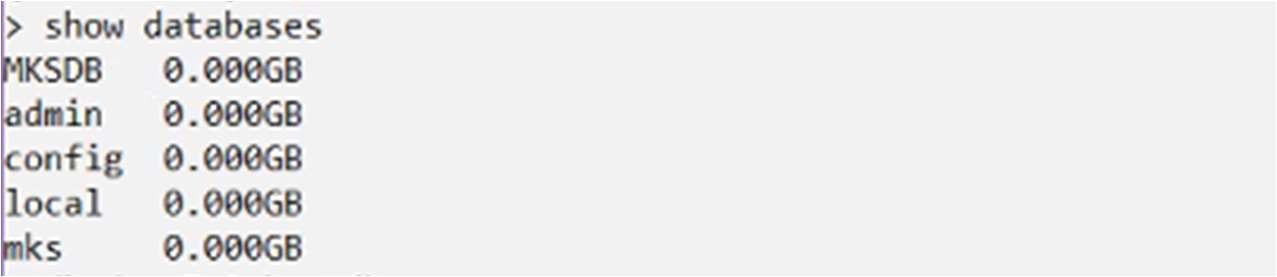
> use mks // Using database mks



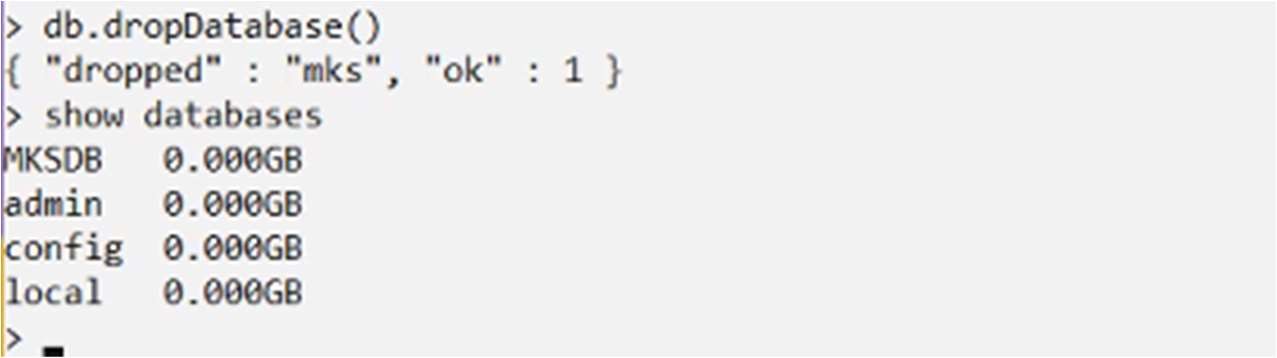
> db.createCollection(“user”) //Creating Empty Collection



> show databases //Database is Created



> db.dropDatabase() //Drop Database



1. **Write a MongoDB query to create, display and drop a collection**

> use sy



> db.user.insert({“name”: “ABC”, “rollno”:10})



> show collections



> db.user.find()



> db.user.drop()



> show collections



1. **Write a MongoDB query to insert, query, update and delete a document**

* Different Methods of inserting Documents

* 1. Insert Document

> use mks

> db.products.insert( { item: "card", qty: 15 } )



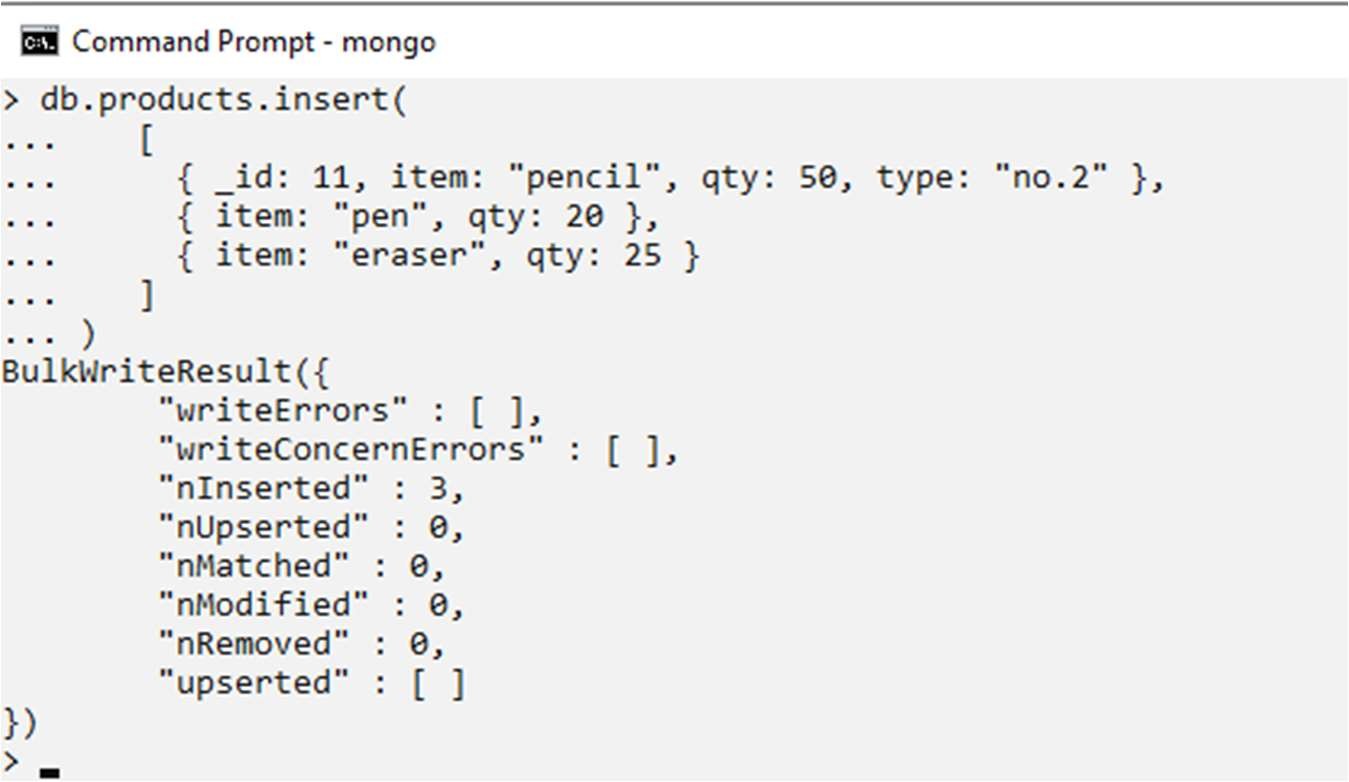
* 1. Insert Multiple Documents db.products.insert([

{ \_id: 11, item: "pencil", qty: 50, type: "no.2" },

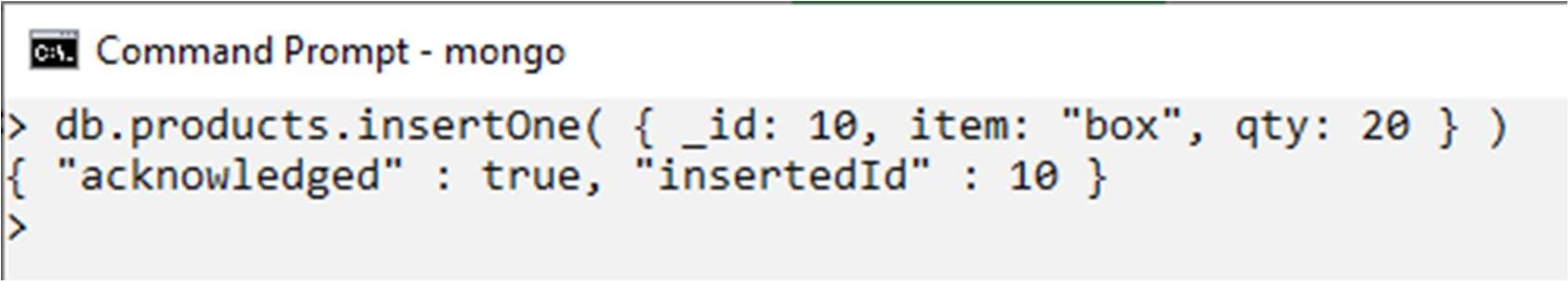
{ item: "pen", qty: 20 },

{ item: "eraser", qty: 25 }

])



* 1. Insert a single document into a collection using db.col.insertOne()



* Updating Document Queries :
  1. Updating Document using $set

Fetching Record with \_id:10 and Updating status from “A” to “Pending”

> db.inventory.find({"\_id":10})

> db.inventory.update({ \_id: 10 },{$set: {status: "Pending" }})

> db.inventory.find({"\_id":10}) // Checking After Update

 ii. Updating Document with overwriting.

Overwriting the Exiting Document.

> db.product.find()

> db.products.update({"item" : "pen"},{"item" : "pen", "qty" : 400, "COD":"Yes"}) > db.product.find()

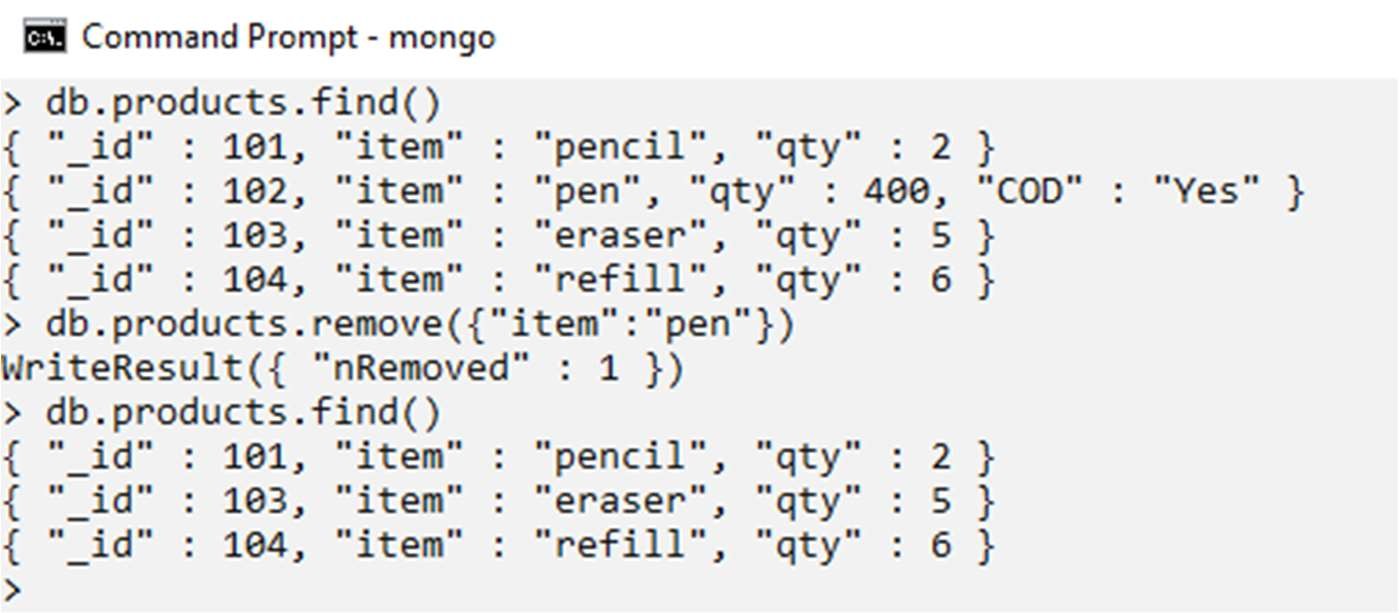


* Deleting Document
* Removing Document By Key: Value Pair Reference

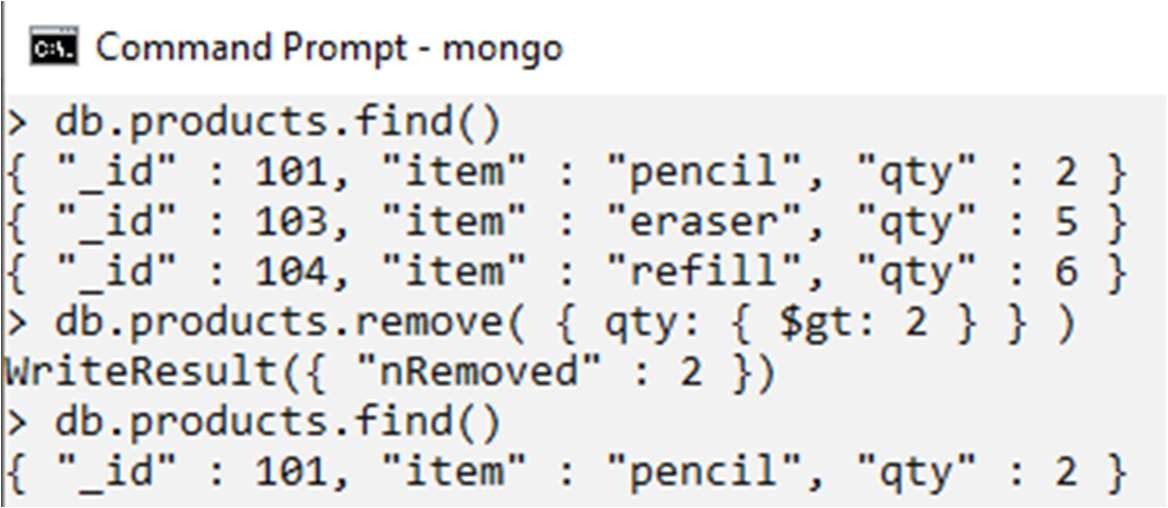
> db.products.find()

> db.products.remove({"item":"pen"})

> db.products.find()



* Remove All Documents that Match a Condition



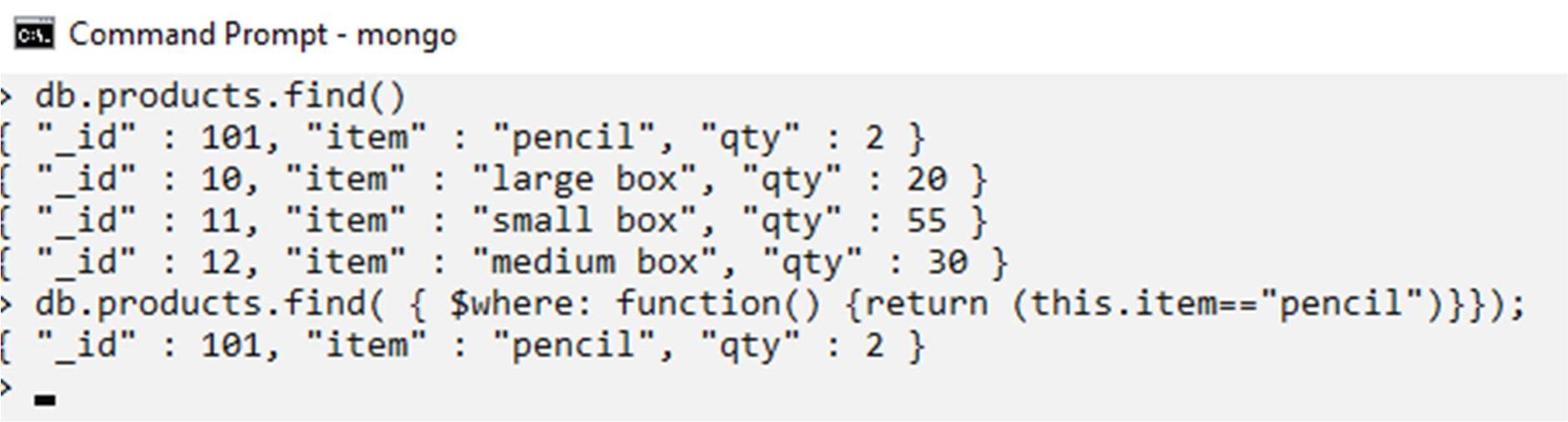
# **Practical No:2 –**

# **Simple Queries with MongoDB**

We shall use WHERE clause in this examples. $WHERE

> db.products.find()

> db.products.find( { $where: function() {return (this.item=="pencil")}});



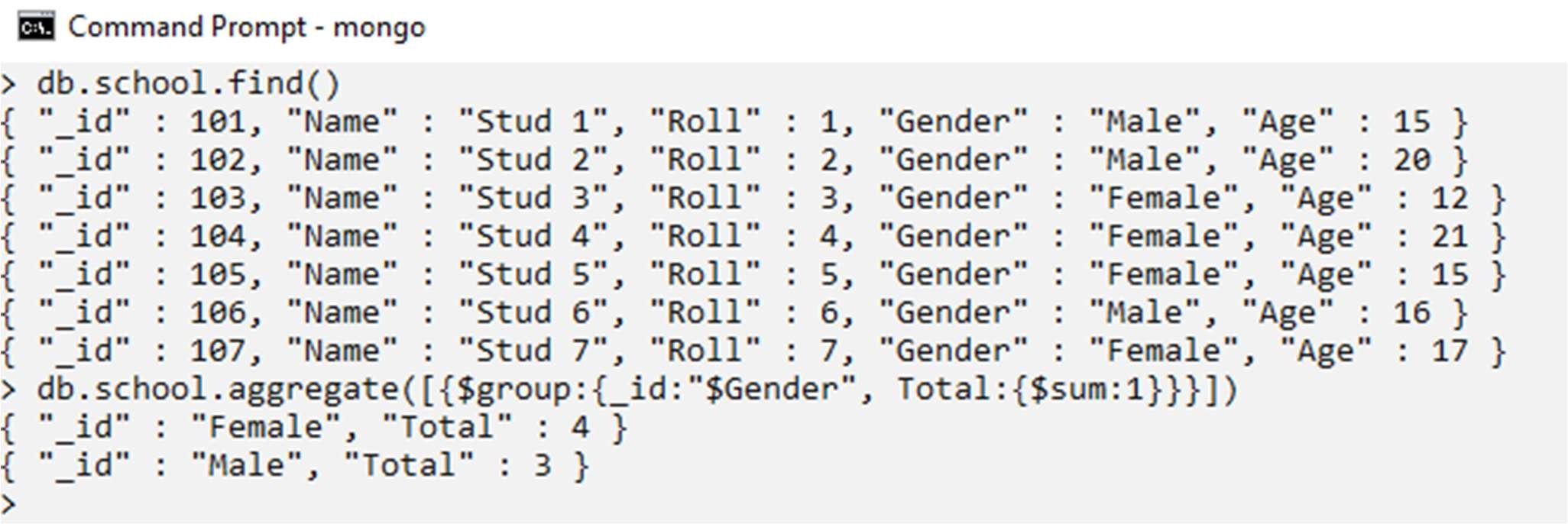
**Practical No:3 – Implementing Aggregation**

1. Write a MongoDB query to use sum, avg, min and max expression

* **Sum**

> db.school.find()

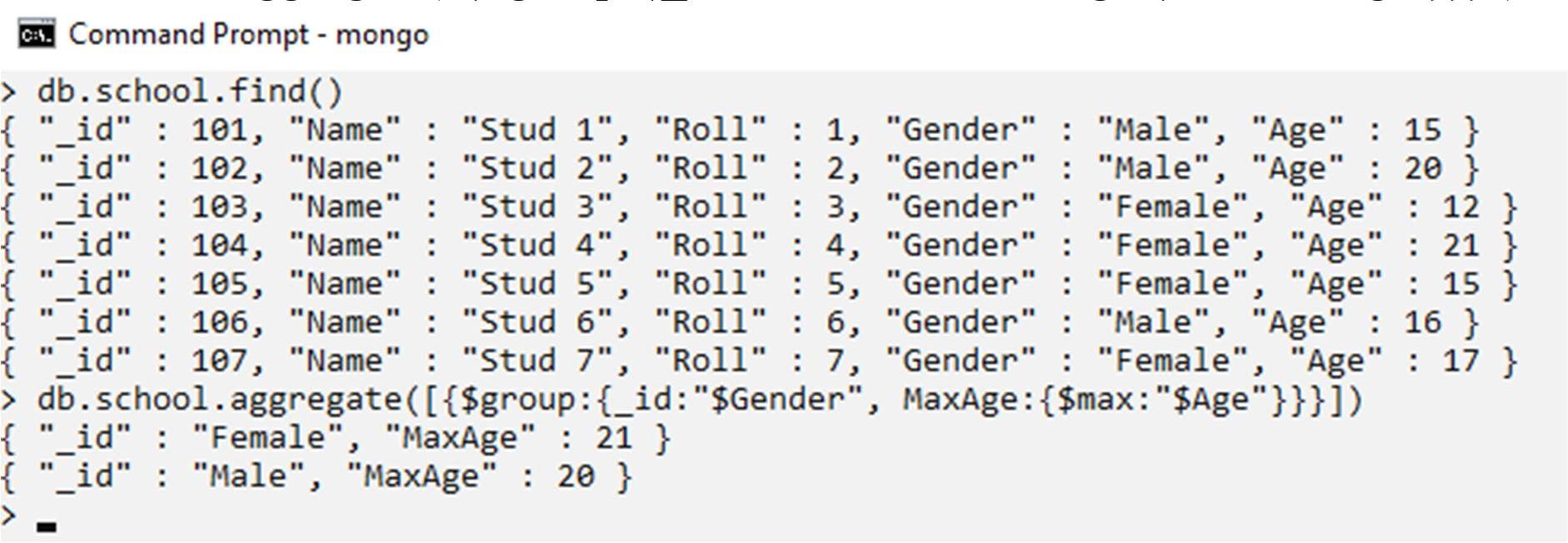
>db.school.aggregate([{$group:{\_id:"$Gender", Total:{$sum:1}}}])



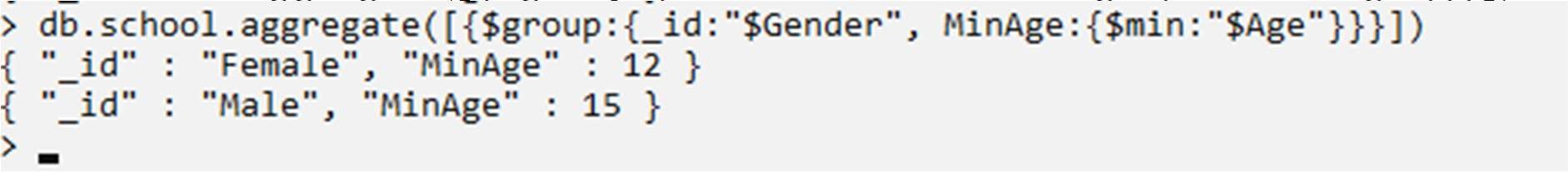
* **Min & Max Expression**

> db.school.find()

>db.school.aggregate([{$group:{\_id:"$Gender", MaxAge:{$max:"$Age"}}}])



> db.school.aggregate([{$group:{\_id:"$Gender", MinAge:{$min:"$Age"}}}])



> db.school.aggregate([{$group:{\_id:"$Gender", AvgAge:{$avg:"$Age"}}}])



1. Write a mongodb query to use Push and AddToSet Expressions.

$push: The $push operator appends a specified value to an array.

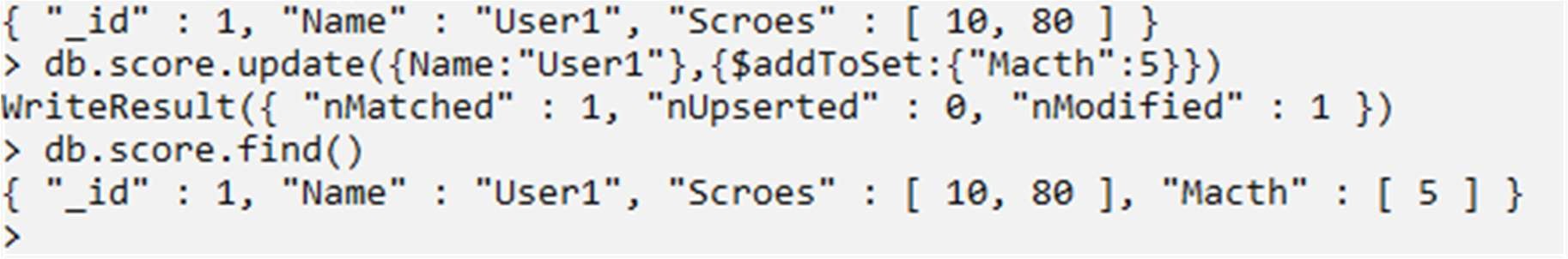
> db.score.find()

> db.score.update({Name:"User1"},{$push:{"Scroes":80}}) > db.score.find()



$addToSet: The operator adds the value to an array unless the value is already present.

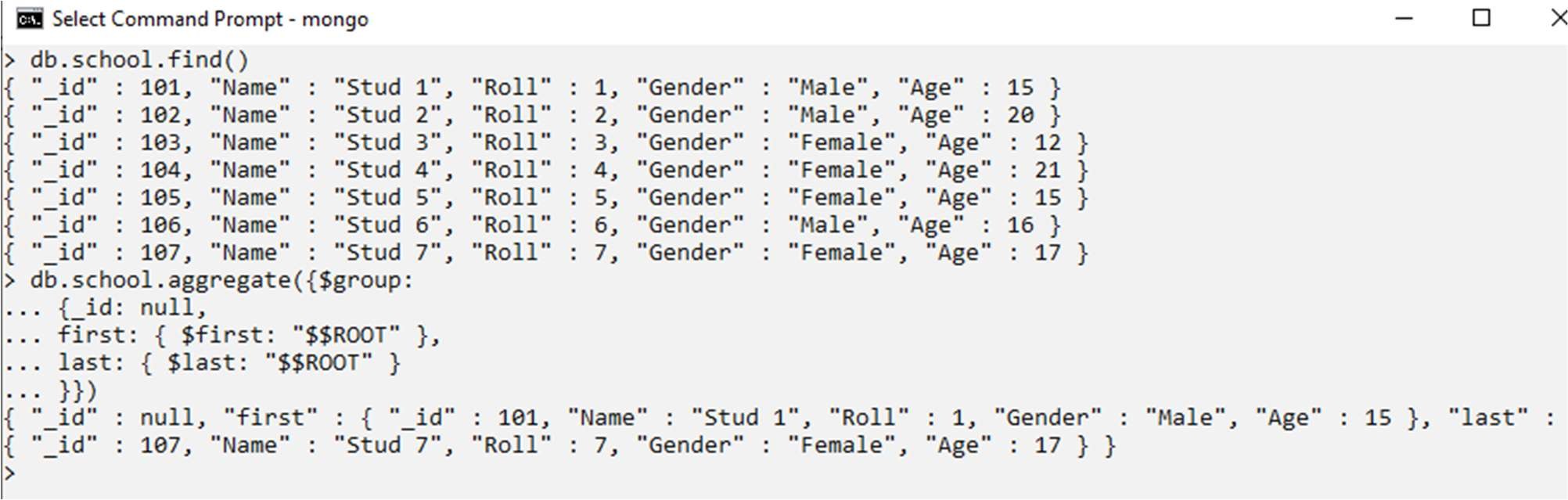
> db.score.update({Name:"User1"},{$addToSet:{"Macth":5}})



1. Write a mongodb query to use $first and $last expression.

> db.school.find()

> db.school.aggregate({$group:{\_id: null,first: { $first: "$$ROOT" },last: { $last: "$$ROOT" }}});

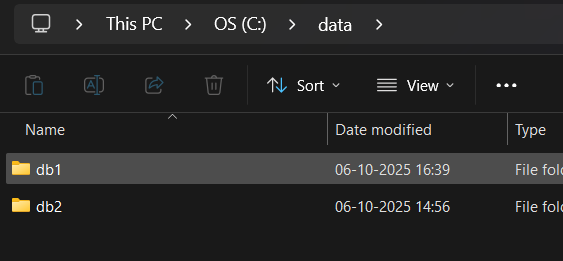


# **Practical No:4 –**

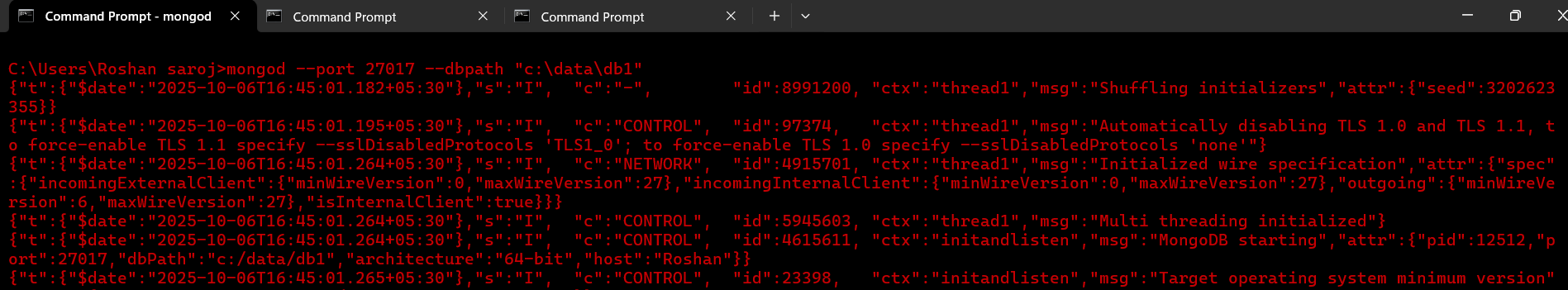
# **Replication, Backup and Restore**

1. **Write a MongoDB query to create Replica of existing database**

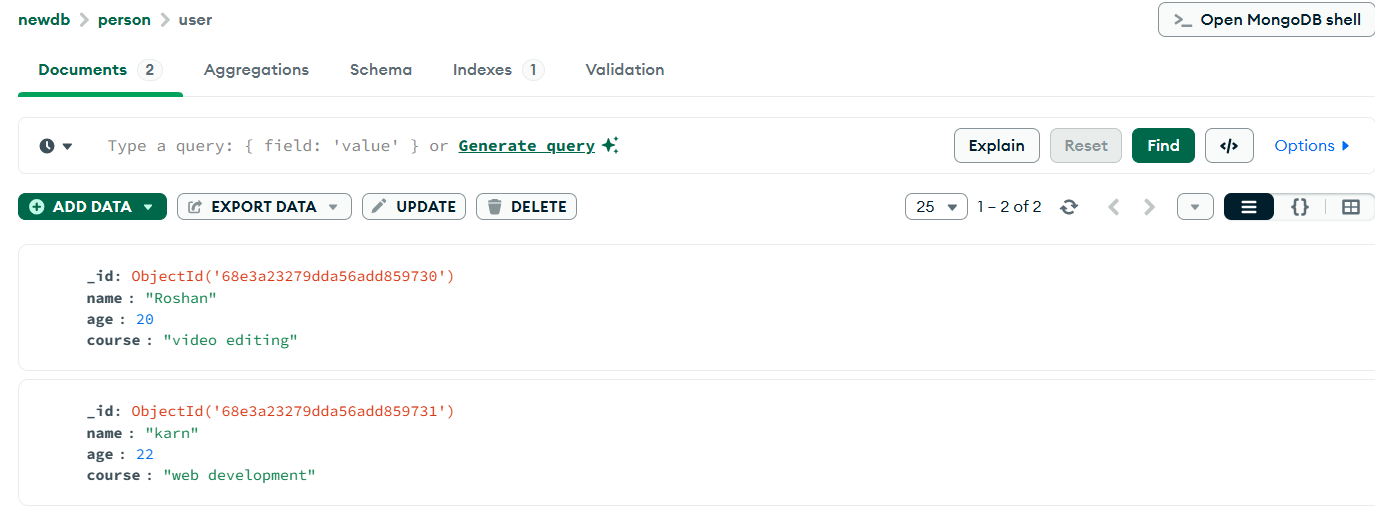
**> Step 1 —** Make folder in c:\

****

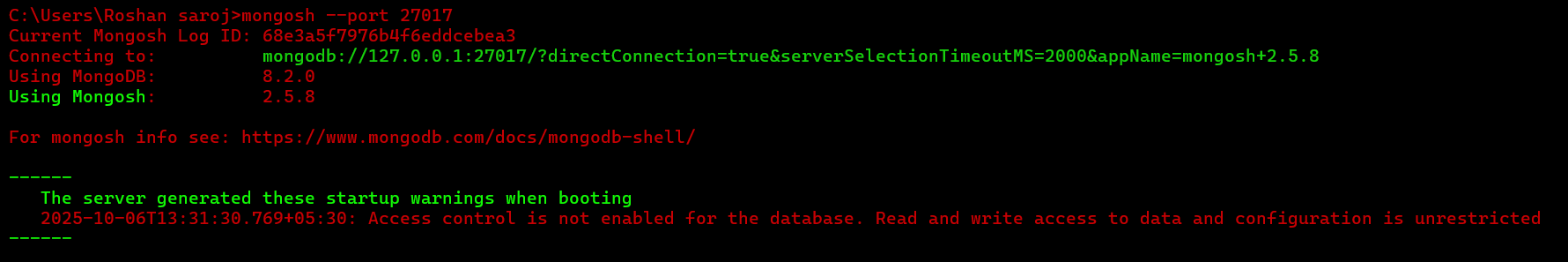
**> Step 2 —** In cmd write mongod with c:\ folder name



**> Step 3 —**  go in mongodb add the data



**> Step 4 —** type mongosh



**> Step 5 —**  show dbs



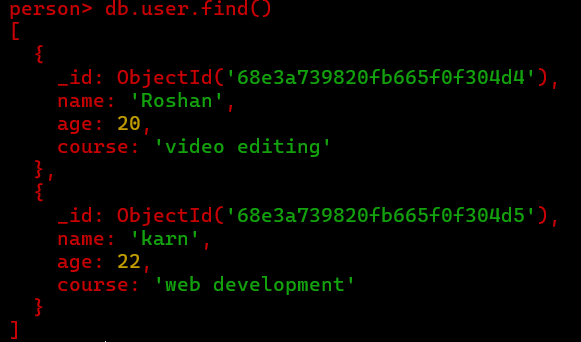
**> Step 6 —** use person



**> Step 7 —** show collections



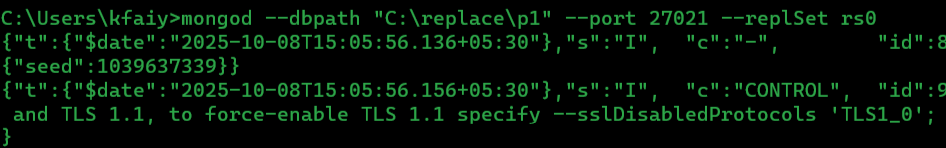
**> Step 8 —**  db.user.find()



**Step 9 :** Start First MongoDB Instance

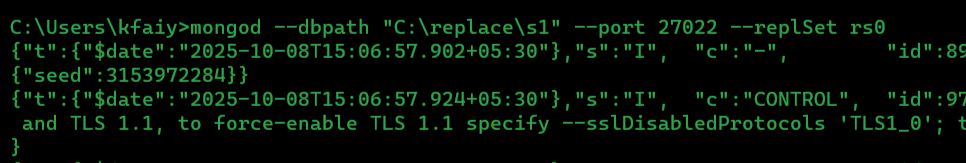
Open Command Prompt and run:

mongod --dbpath "C:\replace\p1" --port 27021 --replSet rs0



**Step 10 :** Start Second MongoDB Instance

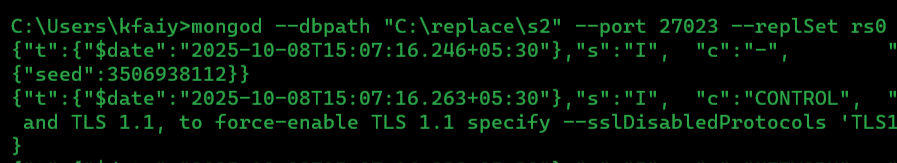
Open another Command Prompt and run:

mongod --dbpath "C:\replace\p2" --port 27022 --replSet rs0

**Step 4: Start Third MongoDB Instance**

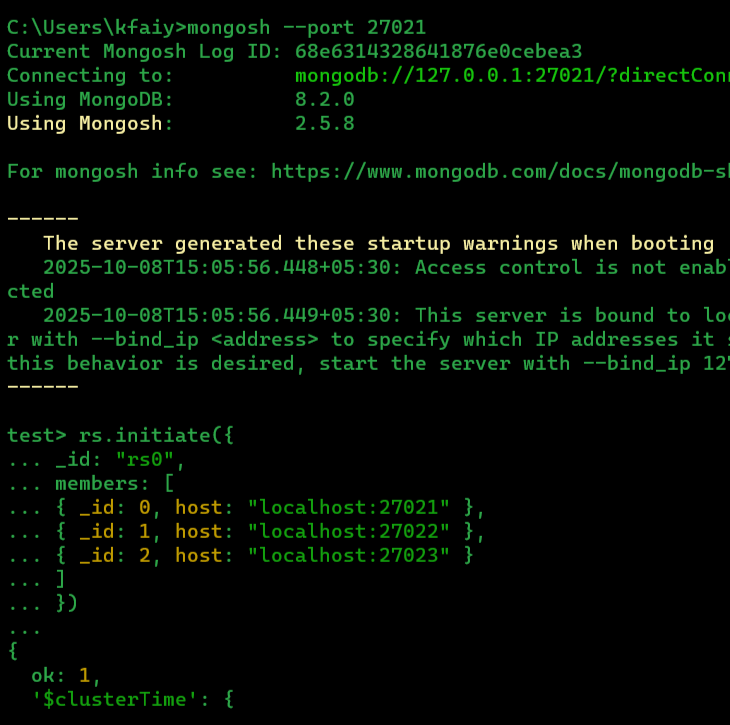
Open one more Command Prompt and run:

mongod --dbpath "C:\replace\p3" --port 27023 --replSet rs0



**Step 11:** Connect to MongoDB Shell

Open a new Command Prompt and connect to the first node:

 mongosh --port 27021

**Step 12 :** Initialize the Replica Set

**mongosh --port 27021**

rs.initiate({

\_id: "rs0", members: [

{ \_id: 0, host: "localhost:27021" },

{ \_id: 1, host: "localhost:27022" },

{ \_id: 2, host: "localhost:27023" }

]

})

**Check status:**

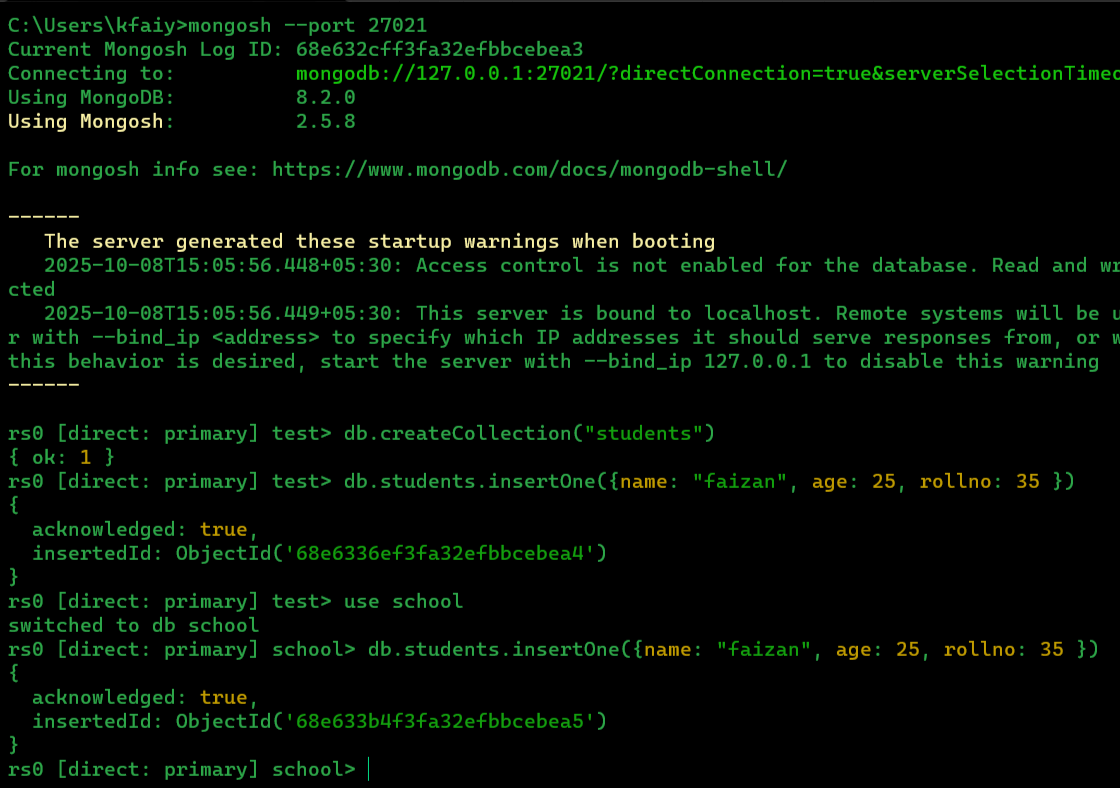
rs.status()



**Step 13:** Insert Data into Primary

Switch to a test database and insert a record:

use school

db.students.insertOne({ name: "Faizan”, age:25, rollno:35 })

**Step 14:** Read from Secondary (Testing Replication**)**

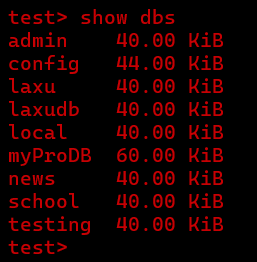
Connect to the secondary node:

mongosh --port 27022

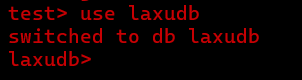


1. **Write a MongoDB query to create a backup of existing database.**

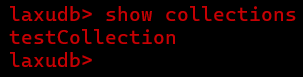
**> Step 1 —** Databases List



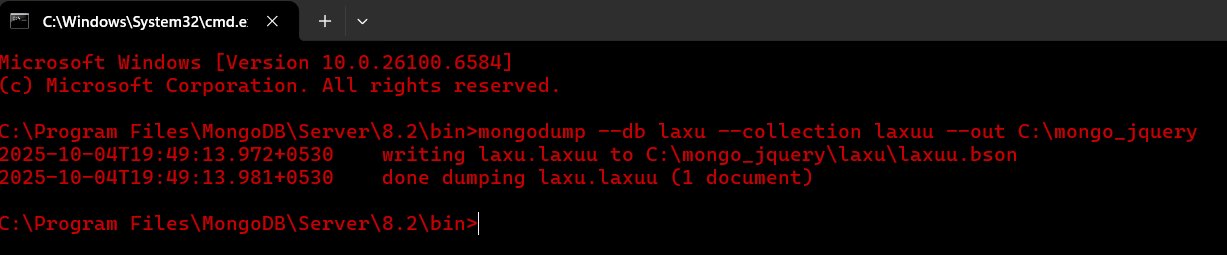
**> Step 2** — Use the Database



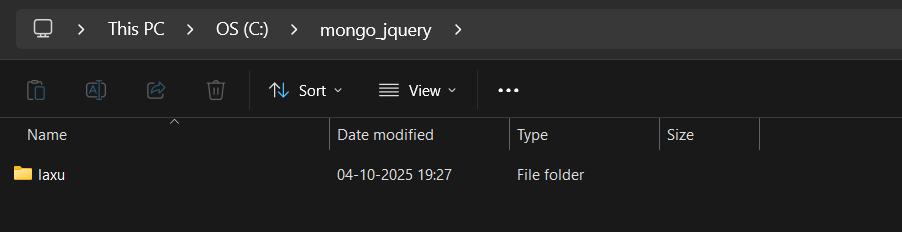
**> Step** 3 — Show Collections



**> Step 4** — Backup Command

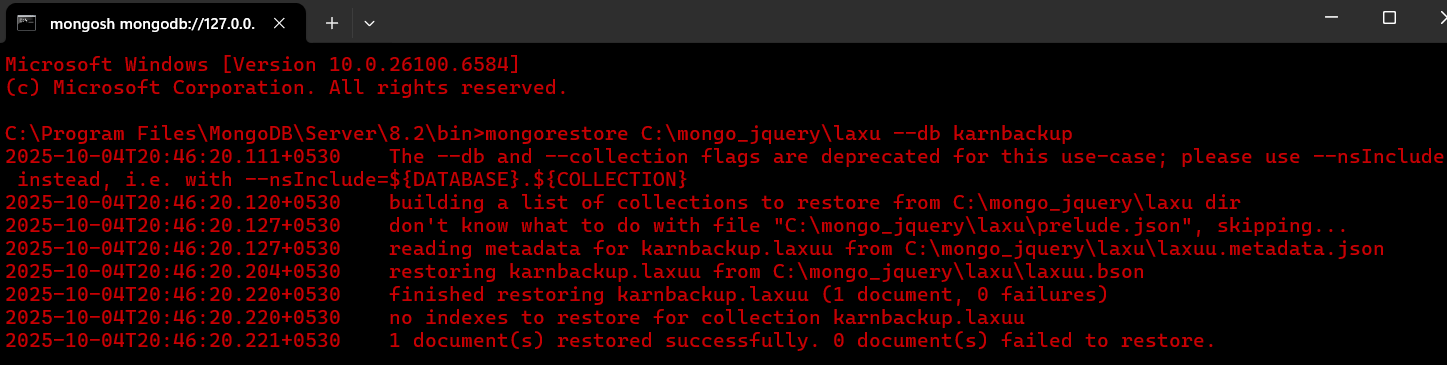


**> Step 5** — Backup Location

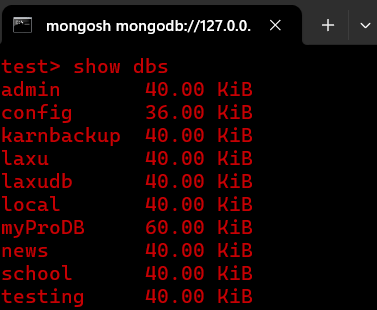


1. **Write a MongoDB query to restore database from the backup**

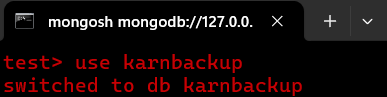
**> Step 1** — Restore Command



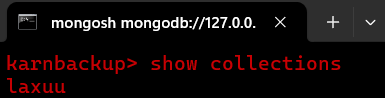
> **Step 2** — Check Databases



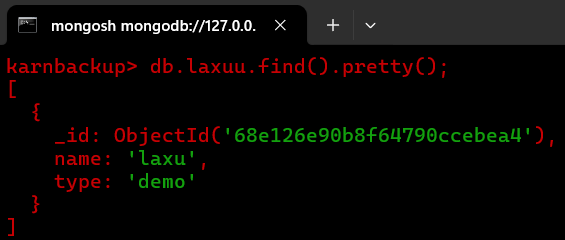
> **Step 3** — Use Restored Database



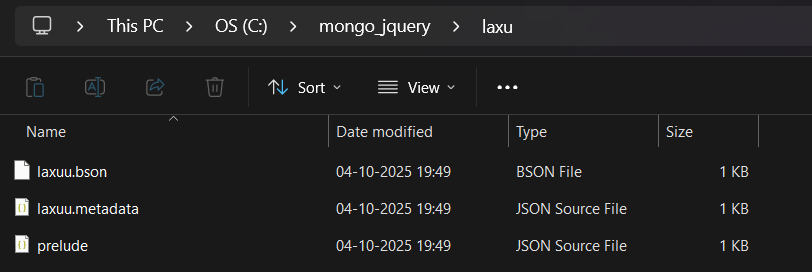
> **Step 4** — Show Collections



> **Step 5** — View Data



> **Step 6 ---** Restore File Location



# **Practical No:5 – Java and MongoDB**

Aim: Connecting Java with MongoDB and inserting, retrieving, updating and deleting

* **Insert: package**

insert.java.mongo; import

com.mongodb.MongoClient; import

com.mongodb.MongoCredential; import

com.mongodb.client.MongoCollection; import

com.mongodb.client.MongoDatabase; import

org.bson.Document; import

com.mongodb.client.FindIterable; import

java.util.Iterator; public class InsertJavaMongo {

public static void main(String[] args) {

MongoClient mongo=new MongoClient("localhost",27017);

MongoCredential credential;

credential=MongoCredential.createCredential("MKS","MakeDB","passwod".toCha rArray());

System.out.println("Credentials::"+credential);

MongoDatabase database=mongo.getDatabase("MakeDB");

System.out.println("Connected to database successfully"); database.createCollection("mycol"); System.out.println("Collection created");

MongoCollection<Document>collection=database.getCollection("mycol");

System.out.println("Collection selected");

Document document=new Document("title","Mongodb").append ("id",1)

.append("Discription","database").append("Created by", "MKS"); collection.insertOne(document); System.out.println("Document inserted");

show(collection);

}

static void show(MongoCollection<Document>collection)

{

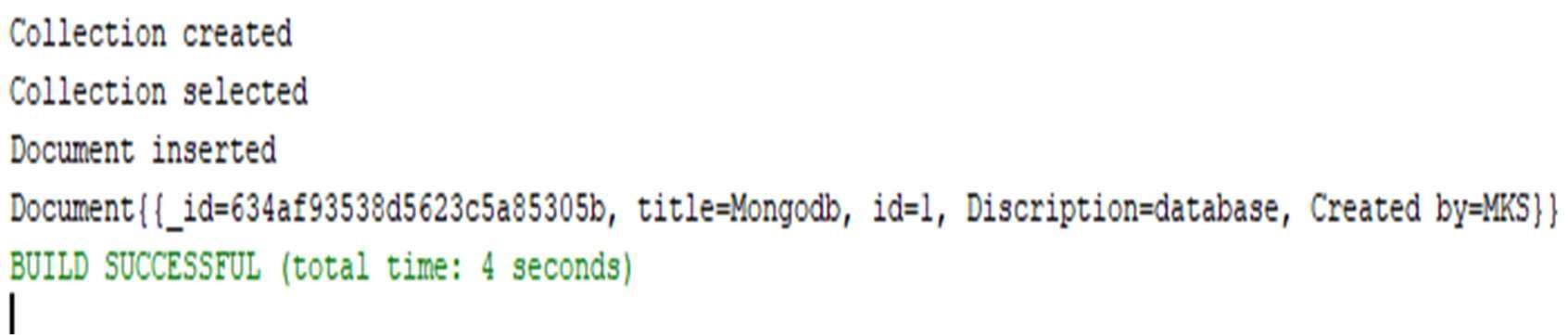
FindIterable<Document> iterDoc= collection.find(); int i=1; Iterator it=iterDoc.iterator(); while(it.hasNext()){

System.out.println(it.next()); i++; }

}

}

Otput:



* **Update:**

package update.java.mongo; import

com.mongodb.MongoClient; import com.mongodb.MongoCredential; import com.mongodb.client.MongoCollection; import com.mongodb.client.MongoDatabase; import org.bson.Document; import com.mongodb.client.FindIterable; import

java.util.Iterator; import

com.mongodb.client.model.Filters; import com.mongodb.client.model.Updates;

public class UpdateJavaMongo {

public static void main(String[] args) {

MongoClient mongo=new MongoClient("localhost",27017);

MongoCredential credential;

credential=MongoCredential.createCredential("MKS","MakeDB","passwod".toCha rArray());

System.out.println("Credentials::"+credential);

MongoDatabase database=mongo.getDatabase("MakeDB");

System.out.println("Connected to database successfully");

MongoCollection<Document>collection=database.getCollection("mycol"); System.out.println("Collection selected"); collection.updateOne(Filters.eq("id","1"),Updates.set("id",2));

System.out.println("Updated Successfully");

}

static void show(MongoCollection<Document>collection)

{

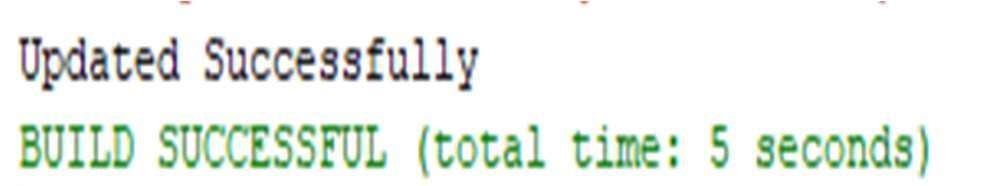
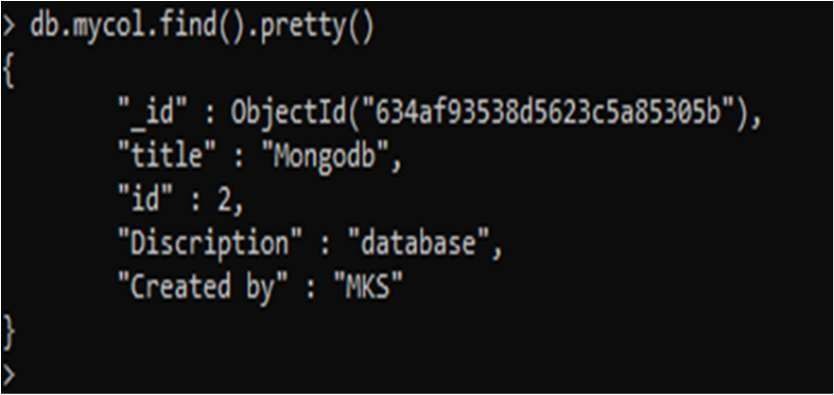
FindIterable<Document> iterDoc= collection.find(); int i=1; Iterator it=iterDoc.iterator(); while(it.hasNext()){

System.out.println(it.next()); i++; } }}

Output:

package delete.java.mongo; import

com.mongodb.MongoClient; import



com.mongodb.MongoCredential; import com.mongodb.client.MongoCollection; import com.mongodb.client.MongoDatabase; import org.bson.Document; import com.mongodb.client.FindIterable; import

java.util.Iterator; import

com.mongodb.client.model.Filters; import com.mongodb.client.model.Updates; public class DeleteJavaMongo {

public static void main(String[] args) {

MongoClient mongo=new MongoClient("localhost",27017);

MongoCredential credential;

credential=MongoCredential.createCredential("MKS","MakeDB","password".toCh arArray());

System.out.println("Credentials::"+credential);

MongoDatabase database=mongo.getDatabase("MakeDB");

System.out.println("Connected to database successfully");

MongoCollection<Document>collection=database.getCollection("mycol"); System.out.println("Collection selected"); collection.deleteOne(Filters.eq("id",2)); System.out.println("Document deleted"); show(collection);

} static void show(MongoCollection<Document>collection)

{

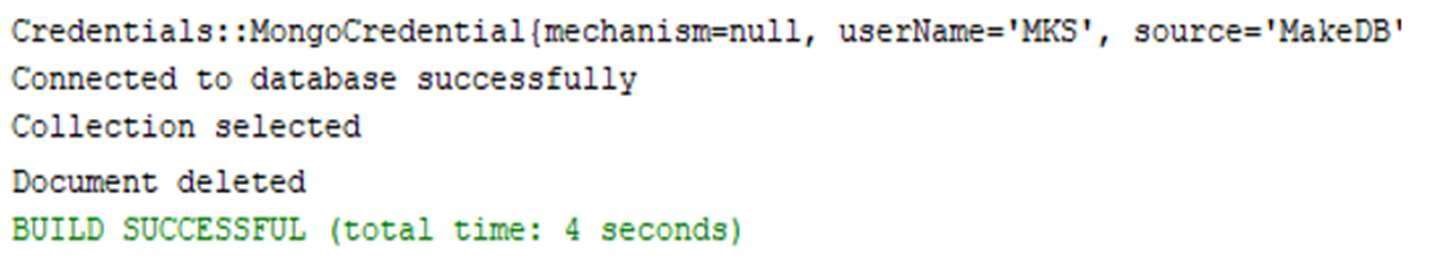
FindIterable<Document> iterDoc= collection.find(); int i=1; Iterator it=iterDoc.iterator(); while(it.hasNext()){

System.out.println(it.next()); i++; }

}

}

Output:



* **Delete Package**

retrieve.java.mongo; import

com.mongodb.MongoClient import com.mongodb.MongoCredential; import com.mongodb.client.MongoCollection; import com.mongodb.client.MongoDatabase; import  
org.bson.Document; import

com.mongodb.client.FindIterable; import

java.util.Iterator;

public class RetrieveJavaMongo { public static void

main(String[] args) {

MongoClient mongo=new MongoClient("localhost",27017);

MongoCredential credential;

credential=MongoCredential.createCredential("MKS","MakeDB","password".t oCharArray()); System.out.println("Credentials::"+credential);

MongoDatabase database=mongo.getDatabase("MakeDB");

System.out.println("Connected to database successfully");

MongoCollection<Document>collection=database.getCollection("mycol");

System.out.println("Collection selected"); show(collection);

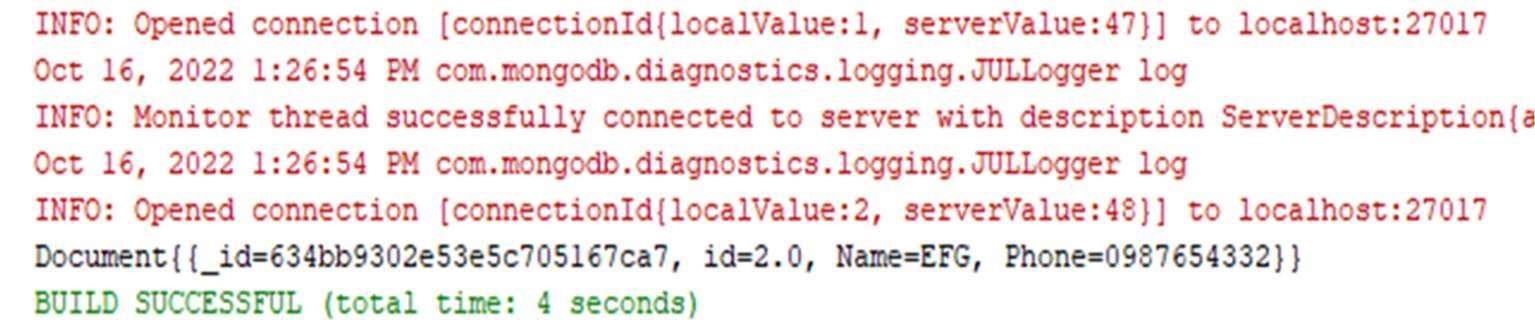
}

static void show(MongoCollection<Document>collection)

{

FindIterable<Document> iterDoc= collection.find(); int i=1; Iterator it=iterDoc.iterator(); while(it.hasNext()){ System.out.println(it.next()); i++;

}}}



# **Practical No:6 –**

# **Python and MongoDB**

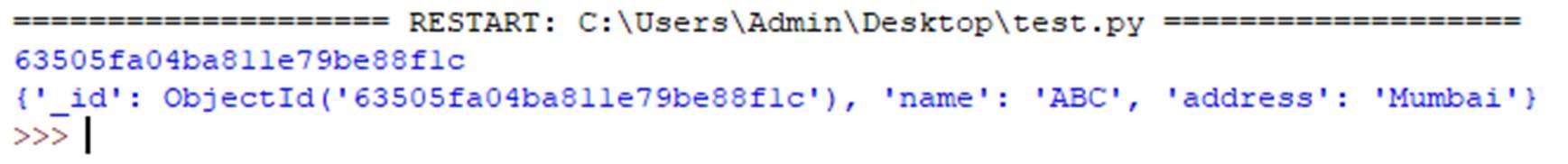
Aim: Connecting Python with MongoDB and inserting, retrieving, updating and deleting.

* Insert:

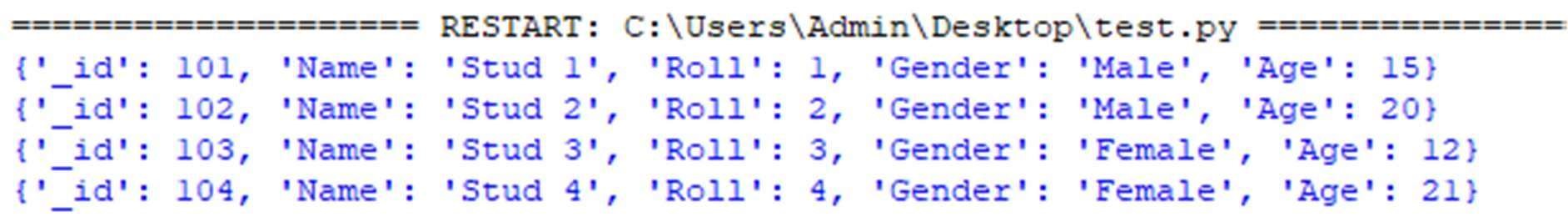
import pymongo myclient=pymongo.MongoClient("mongodb://localhost:27017/")

mydb=myclient["mks"] mycol=mydb["col1"]

x=mycol.insert\_one(={"name":"ABC","address":"Mumbai"}) print(x.inserted\_id) for x in mycol.find(): print(x) Output:



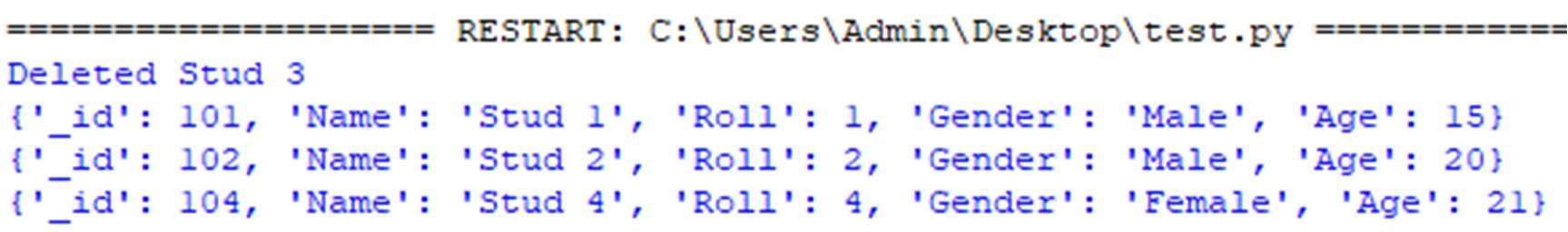
* Retrieve import pymongo myclient=pymongo.MongoClient("mongodb://localhost:27017/") mydb=myclient["test"] mycol=mydb["school"] for x in mycol.find():



* Delete import pymongo

myclient=pymongo.MongoClient("mongodb://localhost:27017/") mydb=myclient["test"] mycol=mydb["school"]

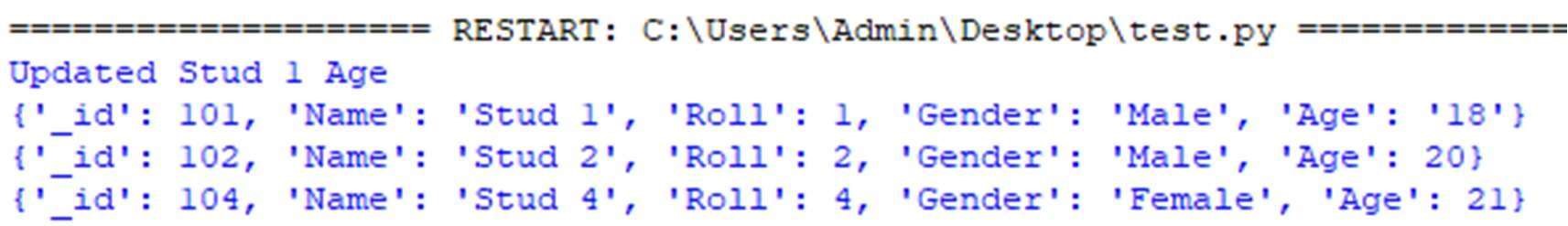
mycol.delete\_one({"Name":"Stud 3"}) print("Deleted Stud 3") for x in mycol.find():



* Update

import pymongo myclient=pymongo.MongoClient("mongodb://localhost:27017/") mydb=myclient["test"] mycol=mydb["school"] mycol.update\_one({"Name":"Stud 1"},{"$set":{"Age":"18"}})

print("Updated Stud 1 Age") for x in mycol.find(): print(x)



**Practical No:7–**

**Programs on Basic jQuery**

* 1. **jQuery Basic**

Index.html >

<html>

<body>

<h2>Create Object from JSON String</h2> <h3 id="demo"></h3> <script> var txt ='{"name":"XYZ","age":"17","City":"MUM"}' var

obj=JSON.parse(txt)

document.getElementById("demo").innerHTML="Name " + obj.name +

", Age " + obj.age; </script>

<

/body>

<

/html>



\

* 1. **jQuery Events**

Index.html >

<html> <head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("p").click(function(){ //For Double Click Event Write .dblclick

$(this).hide();

});

});

</script>

</head>

<body>

<p>This text will disapper if you click on it.</p>

<p>Click to make this text disapper!</p>

</body>

</html> Output:



B) jQuery Selectors jQuery Hide and Show effects

* **Tag Selector**

Note: We Will Select Paragraph Tag and will we give Background Color Using jQuery

Index.html >

<html> <head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("p").css("background-color","Aqua");

});

</script>

</head>

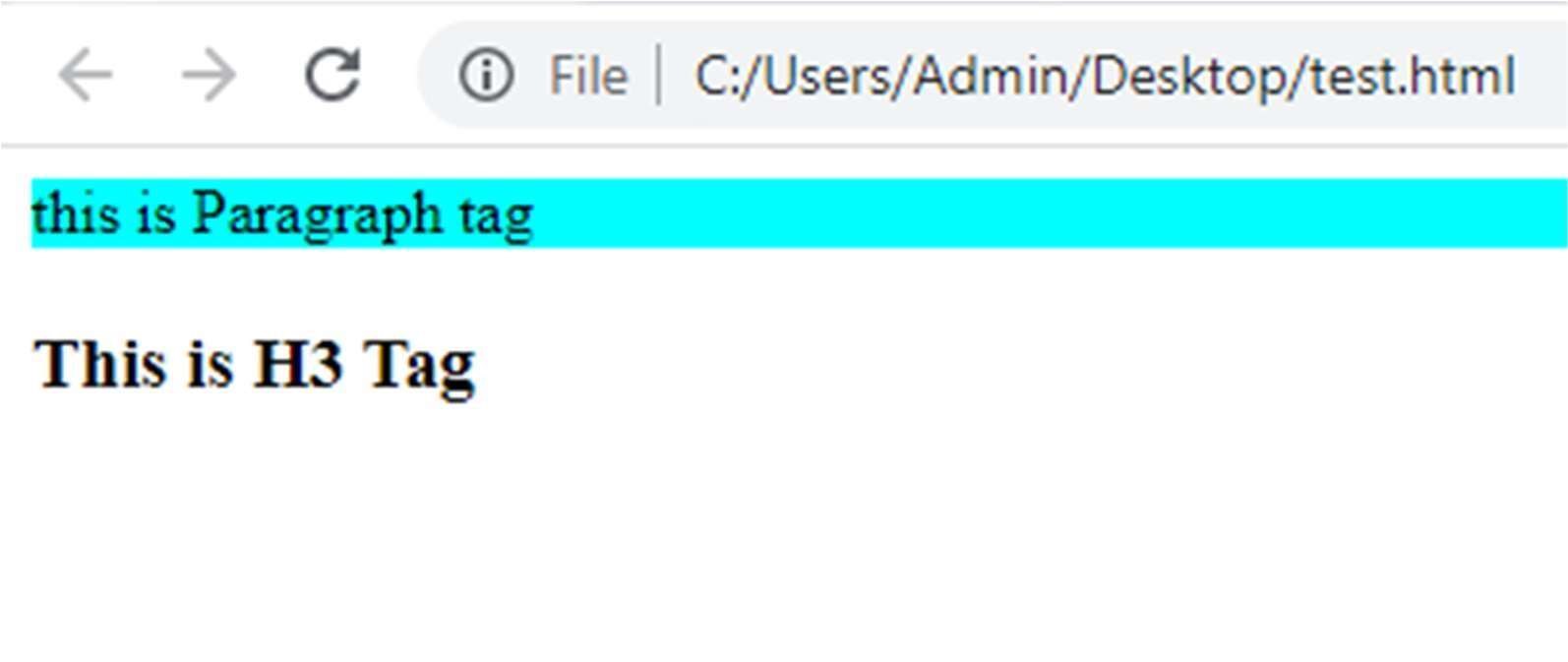
<body>

<p>this is Paragraph tag</p>

<h3>This is H3 Tag</h3>

</body>

</html>



* jQuery Hide Paragraph

Note: We Will Hide Paragraph Tag on Button Click Event.

Index.html >

<html> <head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("p").hide();

});

});

</script>

</head>

<body>

<h2>Welcome To JQuery</h2>

<p>Paragraph 1</p>

<p>Paragraph 2</p>

<button>Click to hide paragraphs.</button>

</body>

</html>

Output:



**C) jQuery fading effects, jQuery Sliding effects**

i) SlideUp

Index.html >

<html> <head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("#flip").click(function(){

$("#panel").slideUp("slow");

});});

</script> <style> #panel,#flip{ padding :10px; text-align:center; background-color : #ffcc00;

border: dashed; border-width: 2px; color: red; }

#panel{ padding:50px; color: black; }

</style> </head>

<body>

<div id="flip">This is SlideUp</div>

<div id="panel"><h2>This is Content !!</h2></div>

</body>

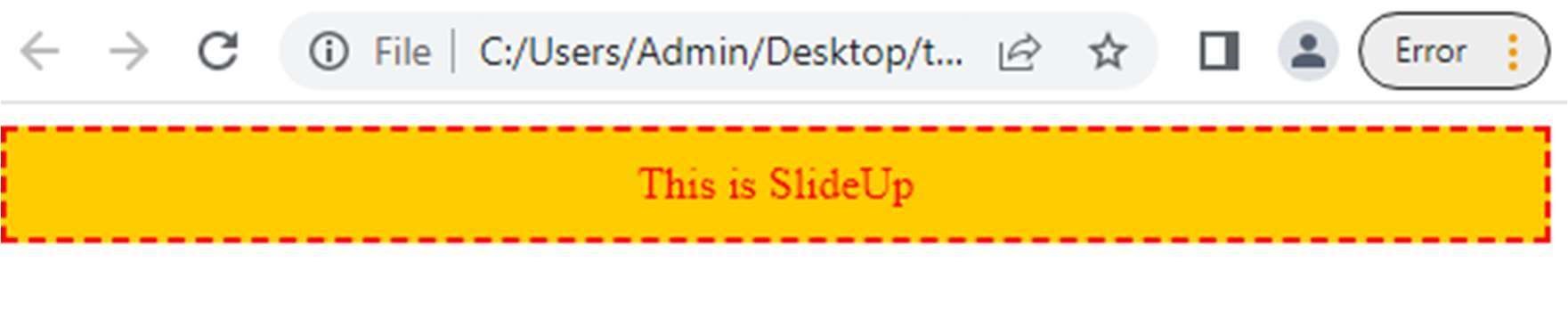
</html>

Output:

Before Click:



After Click:



ii) SlideDown

Index.html >

<html>

<head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("#panel").click(function(){

$("#flip").slideDown("slow");

});});

</script>

<style> #panel,#flip{

padding :10px; text-align:center; background-color : #ffcc00;

border: dashed; border-width: 2px; color: red; }

#panel{ padding:50px; color: black; }

</style> </head>

<body>

<div id="flip">This is SlideDown</div>

<div id="panel"><h2>This is Content !!</h2></div>

</body>

</html>

Output:

Before Click:



After Click:



iii) SlideToggle

Index.html >

<html>

<head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("#flip").click(function(){

$("#panel").slideToggle("slow"); // Toggle Contains Both SideUp & Down

});});

</script>

<style> #panel,#flip{

padding :10px; text-align:center; background-color : #ffcc00;

border: dashed; border-width: 2px; color: red; }

#panel{ padding:50px; color: black; }

</style> </head>

<body>

<div id="flip">This is Toogle</div>

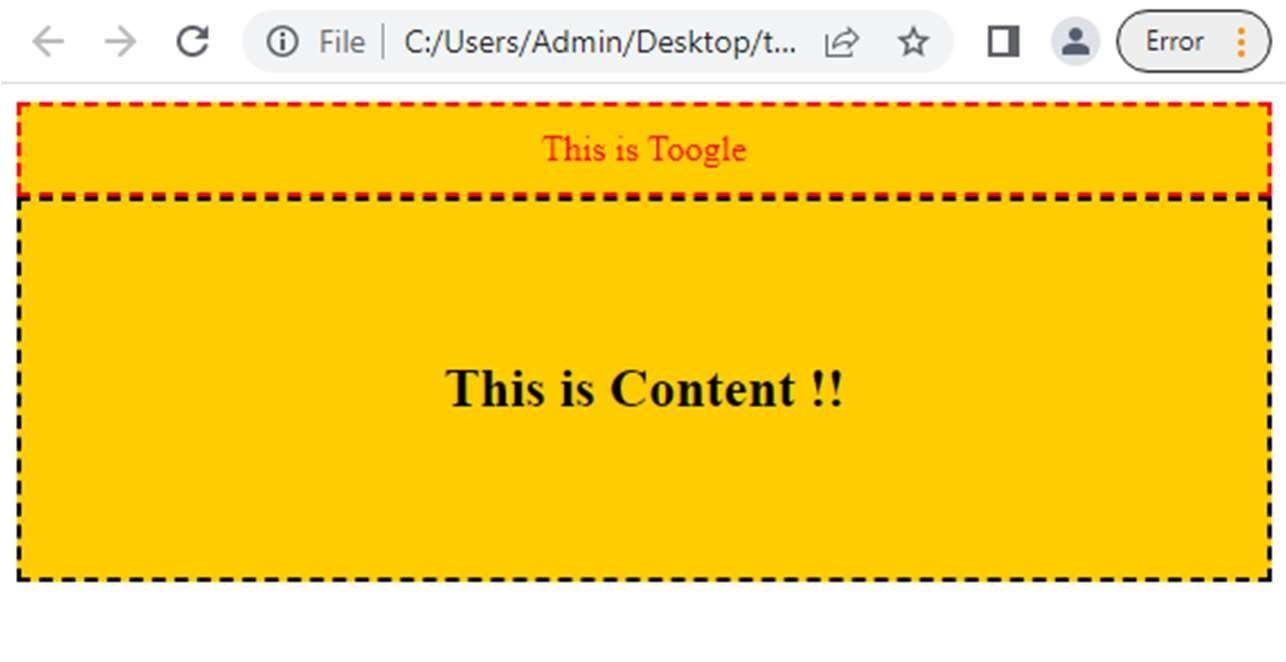
<div id="panel"><h2>This is Content !!</h2></div>

</body>

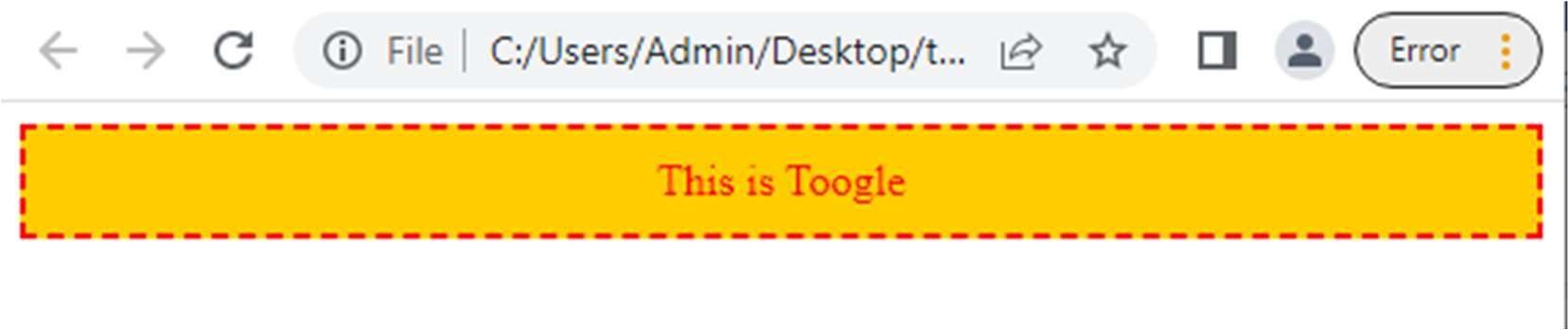
</html>

Output:

Before Click:



After Click:



**Practical No:8 – jQuery Advanced**

**A) jQuery Animation effects, jQuery Chaining**

* Animation effects

Code:

<html>

<head> <style> div { margin-top: 10px;

background:#ffcc00; border: dashed; border-

width: 2px; height:100px; width:100px; position:absolute;

}

</style> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("div").animate({left:'300px'});

});

});

</script>

</head>

<body>

<button>Start Animation</button>

<div>This is Square </div>

</body>

</html> Output:

|  |  |
| --- | --- |
| Before | After |
|  |  |

* Chaining effects

The technique called chaining, that allows us to run multiple jQuery commands, one after other, on the same elements.

Code:

<html>

<head> <style> div {

margin-top: 10px; background-color:#ffcc00; border: dashed; border-

width: 2px; height:100px; width:100px; position:absolute;

}

</style> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("div").animate({left:'250px'},4000,"linear")

.css("background-color","#cc00cc")

.css("border-radius","25px");

});

});

</script>

</head>

<body>

<button>Start Animation</button>

<div>This is Square</div>

</body>

</html> Output:

|  |  |
| --- | --- |
| Before | After |
|  |  |

**B) jQuery Callback, jQuery Get**

* Call Back

A callback function is executed after the current effect is finished: Syntax:

$(selector).hide(speed,callback);

Code:

<html> <head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("div").hide("slow",function(){ alert("Hide Successfully!"); });

});});

</script>

</head>

<body>

<button>Click to Hide</button>

<div style="background-color: #ffcc00; height: 100px; width: 100px">This is Div</div>

</body>

</html>

Output:

|  |  |
| --- | --- |
| Before | After |
|  |  |

* Get Content

Demonstrate to get content with the jQuery text() and html() methods:

Code:

<html> <head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){ $("#b1").click(function(){ alert("Text:"+$("#hi").text());

});

$("#b2").click(function(){

alert("HTML:"+$("#hi").html());

});

});

</script>

</head>

<body>

<h2 id="hi">This is h2 with <b>bold</b>, <i>italic</i>,

<u>Underline</u> fonts.<h2>

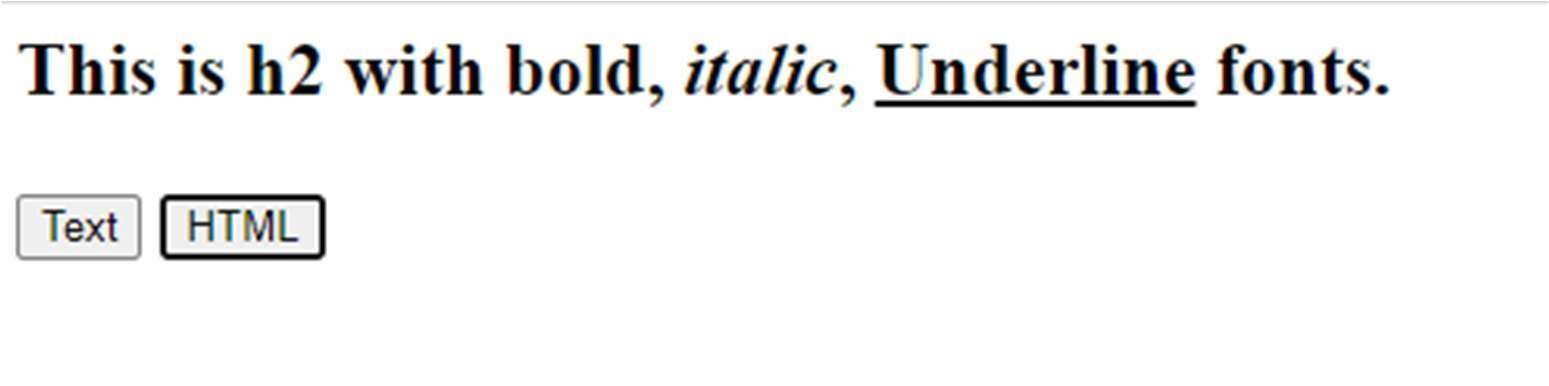
<button id="b1">Text</button>

<button id="b2">HTML</button>

</body>

</html> Output:

|  |  |
| --- | --- |
| Text | HTML |
|  |  |



**C) jQuery Insert Content, jQuery Remove Elements and Attribute**

* Inserting Content Using Append & Prepend

Code:

<html> <head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("#b1").click(function(){

$("p").append("<b> <br> This is Appened text</b>.");

});

$("#b2").click(function(){

$("p").prepend("<b>This is Prepended text</b><br>");

});

});

</script>

</head>

<body>

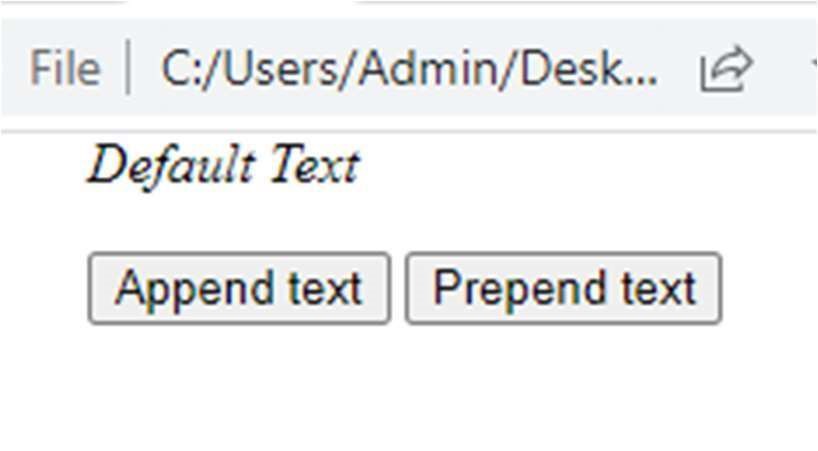
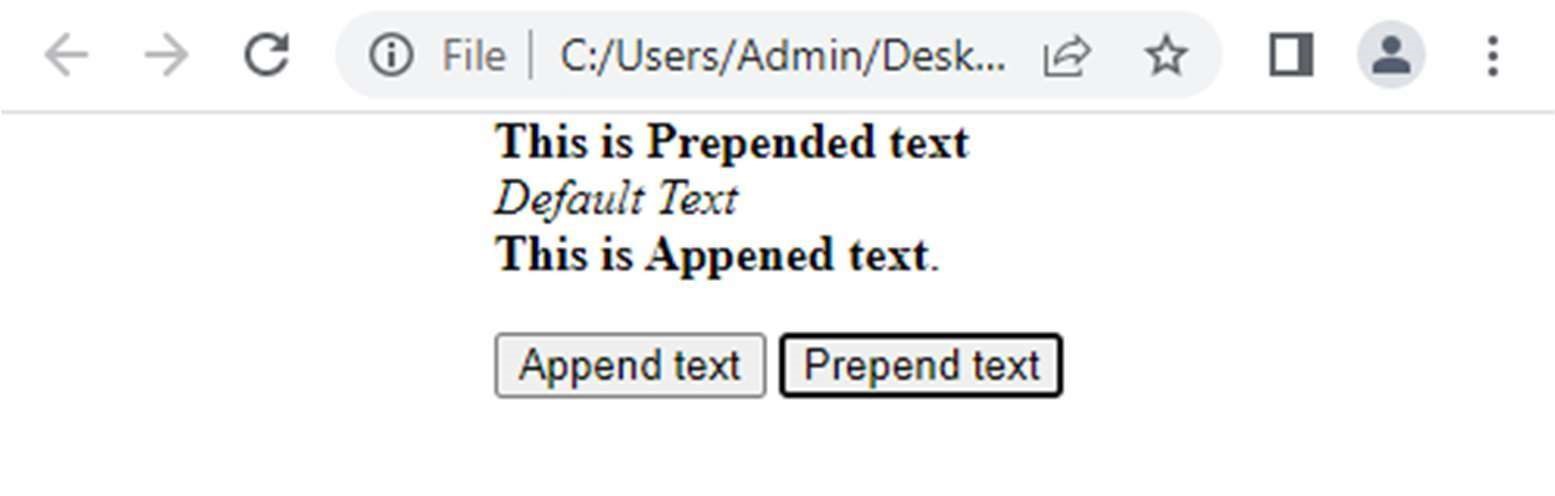
<p> <i>Default Text</i></p>

<button id="b1">Append text</button>

<button id="b2">Prepend text</button>

</body>

</html>



* Remove Element

Code:

<html> <head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("#b1").click(function(){

$("p").remove();

});

});

</script>

</head>

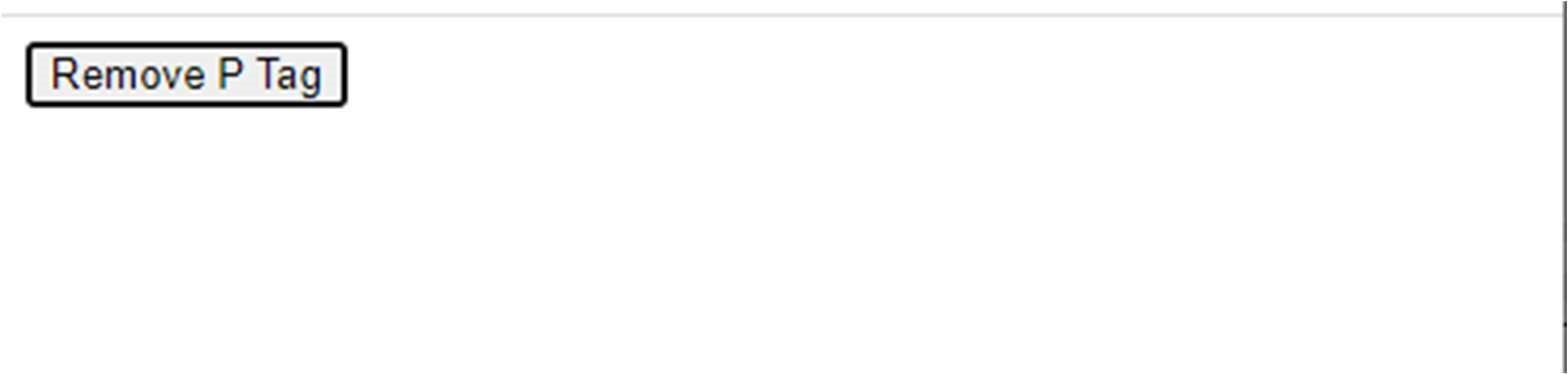
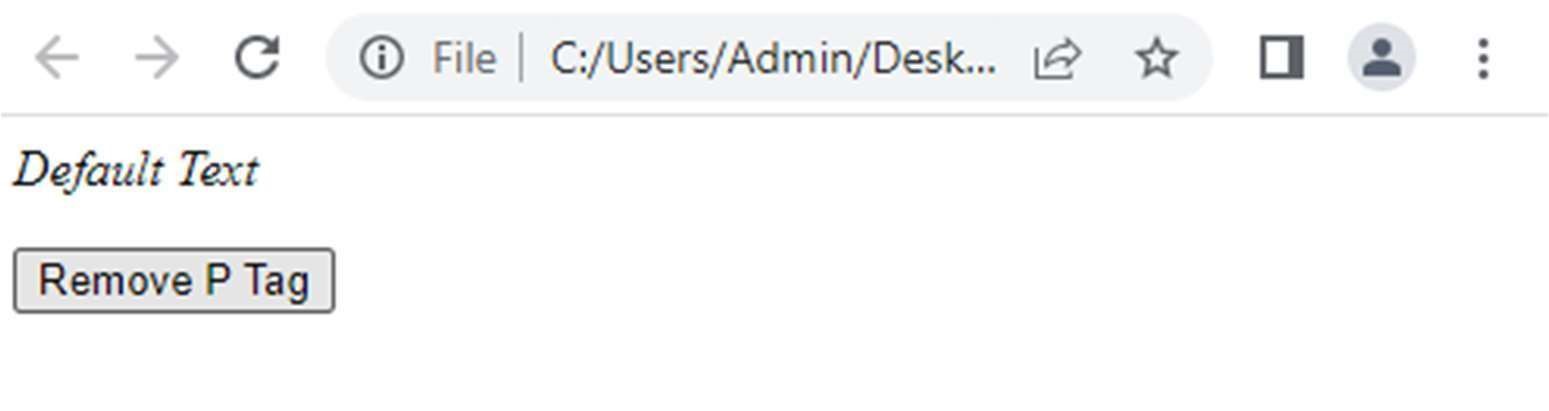
<body>

<p> <i>Default Text</i></p>

<button id="b1">Remove P Tag</button>

</body>

</html>



* Empty() Element

Code:

<html>

<head> <style> div{

height:200px; width:300px; border:2px solid black; background-color:#ffcc00;

}

</style> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

<script>

$(document).ready(function(){

$("button").click(function(){

$("#d1").empty();

});

});

</script>

</head>

<body>

<div id="d1">

<h3 align="center">Some Content </h3>

</div>

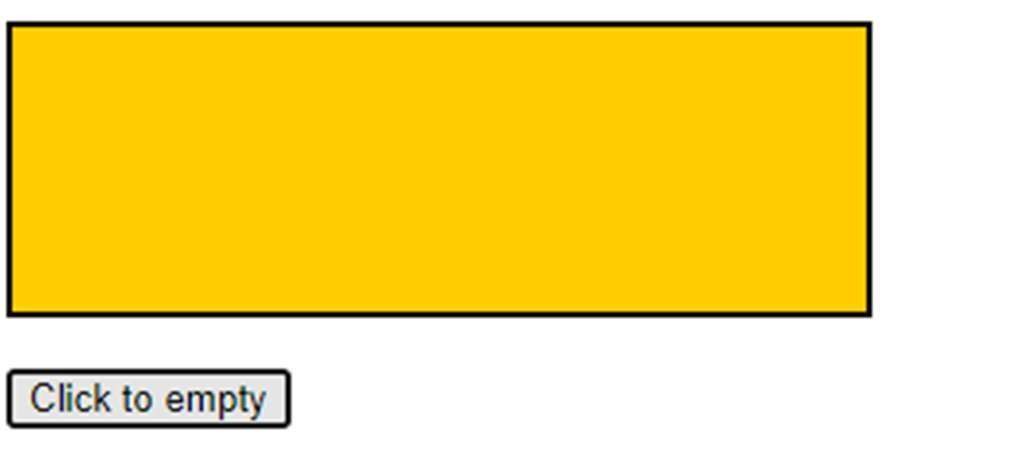
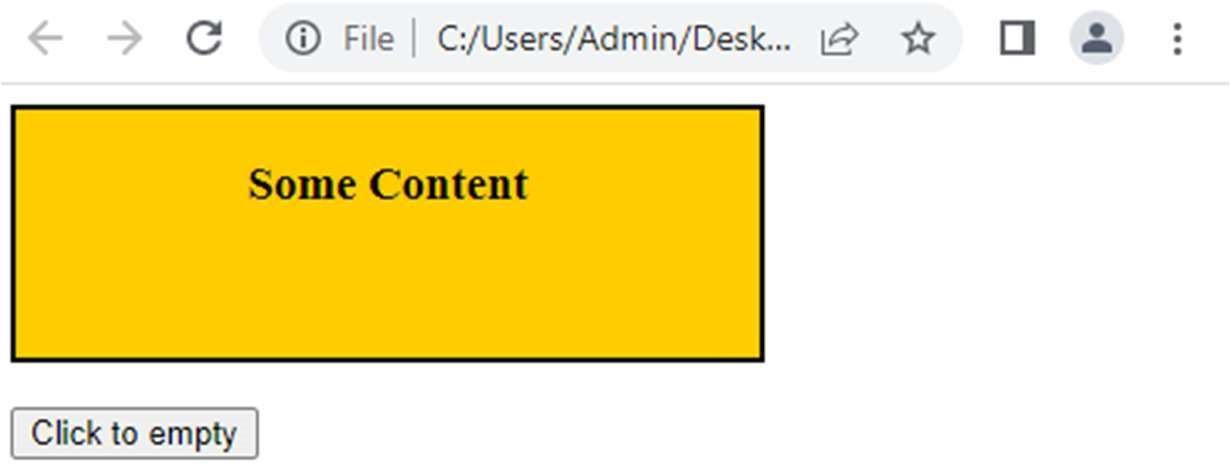
<br>

<button>Click to empty</button>

</body>

</html>

Output:



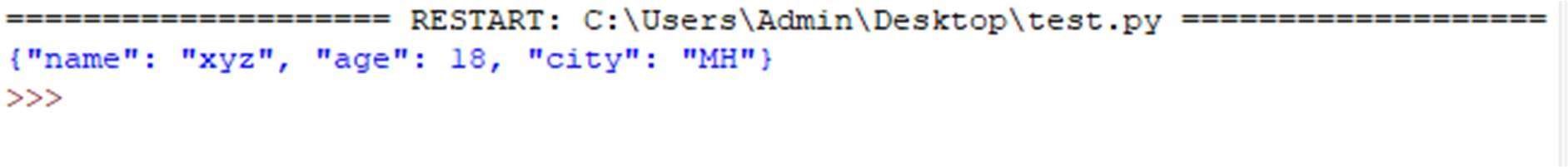
# **Practical No:9 – JSON**

1. Creating JSON

Code:

import json x={ "name":"xyz",

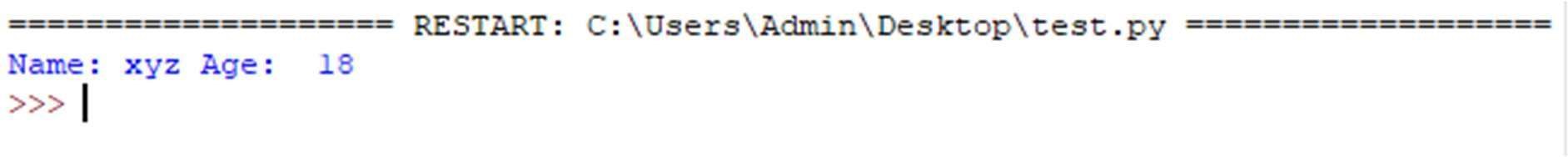
"age":18, "city":"MH" } y=json.dumps(x) print(y) Output:



1. Parsing JSON

Code:

import json x='{"name":"xyz","age":18,"city":"MH"}' y=json.loads(x) print("Name:", y["name"], "Age: ",y["age"]) Output:



1. Persisting JSON

First Create a Json File & Write some Document:

---------------------------------------------------------------------

{

"Name":"Test",

"Class":"TY",

"Sem":5

}

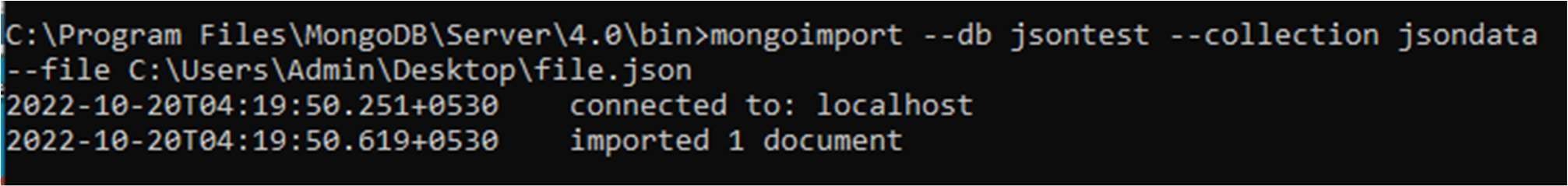
------------------------------------------------- Now, Open Command

Prompt and Go to “C:\Program Files\MongoDB\Server\4.0\bin” and Type Command:

mongoimport --db <Db\_Name> --collection <collection\_name> --file “json\_file\_path” & Hit Enter

Document Will be Inserted From Json File to MongoDB.

Output:





**Practical No:10 –**

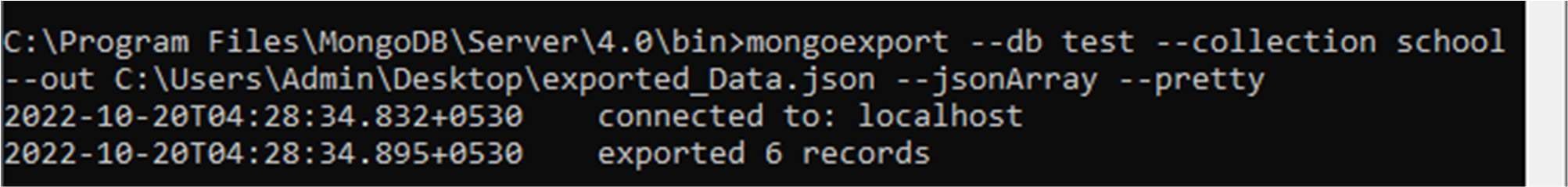
1. **Create a JSON file and import it to MongoDB Steps:**
2. Open Cmd
3. Type cd C:\Program Files\MongoDB\Server\4.0\bin
4. Run command to export data into json file Command:

mongoexport --db <Db\_Name> --collection <collection\_name>

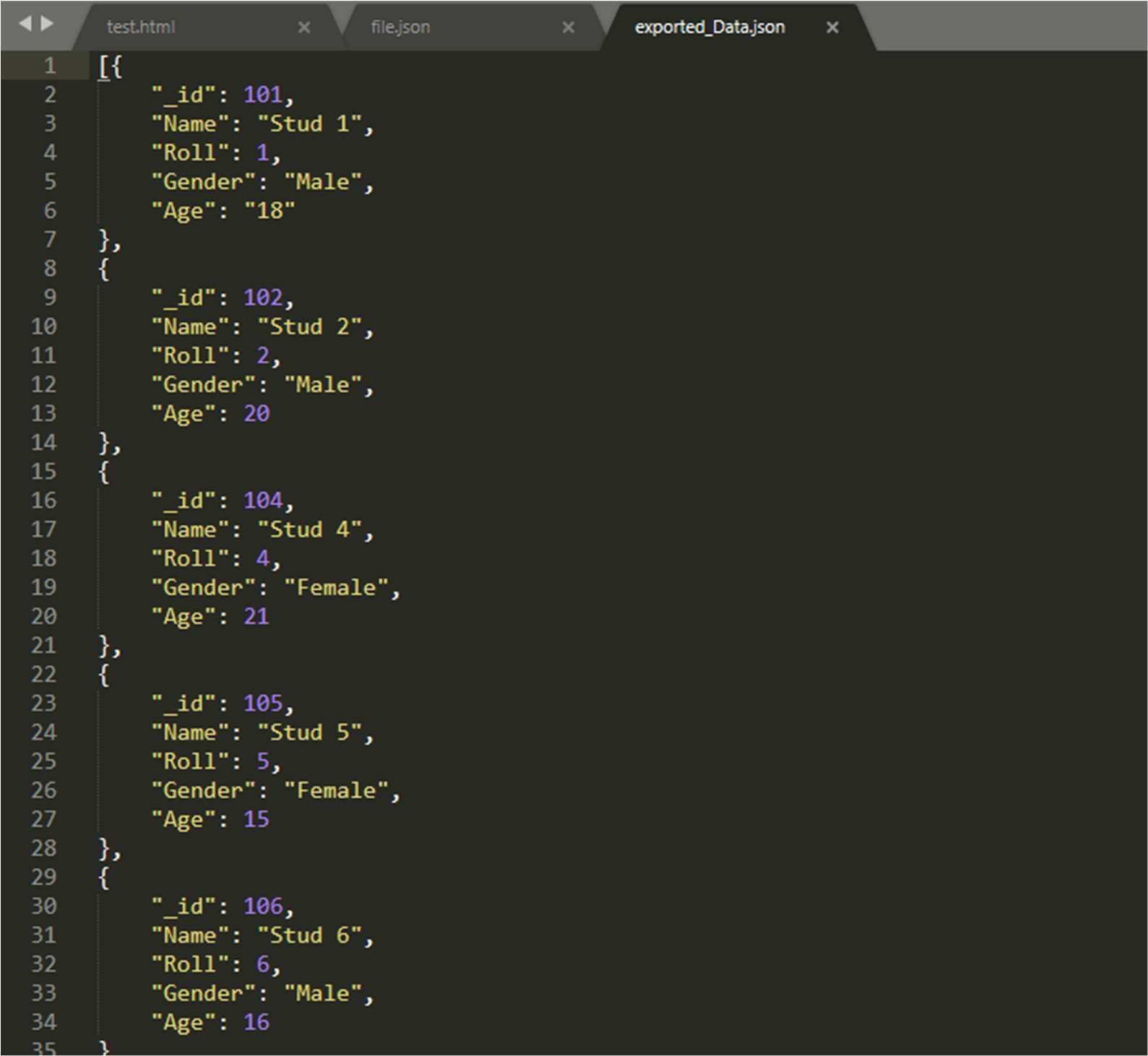
--out “Any\_File\_Name\_With\_Path” --jsonArray –pretty

1. All the Data will be exported into Json File.

Output:



Json File:



1. **Write a MongoDB query to delete JSON object from MongoDB**