

**BHARTIYAVIDYABHAVAN’S**

**M.M.COLLEGEOFARTS,N.M.INSTITUTEOFSCIENCE**

**H.R.J.COLLEGEOFCOMMERCE BHAVAN’SCOLLEGE**

**MUNSHINAGAR,ANDHERIWEST,**

**MUMBAI-400058**

**CERTIFICATE**

This is to certify that Mr. Om Gadade Seat No. TYIT10 of T.Y.BSC Information Technology has satisfactorily completed the practical course in Next Generation Technologies as prescribed by the University of Mumbai during the academic year 2023-2024.

Internal Examiner External Examiner

Signature Signature

Co-ordinator’s College Stamp   
 Signature

INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **Practical**  **No.** | **Date** | **Practical Name** | **Sign** |
| **1** | 20/07/2023 | **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** | 11/08/2023 | **Simple Queries with MongoDB** |  |
| **3** | 19/08/2023 | **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** | 26/08/2023 | **Backup and Restore** |  |
| a. |  | Write a MongoDB query to create a backup of existing database. |  |
| b. |  | Write a MongoDB query to restore database from the backup. |  |
| **5** | 02/09/2023 | **Java and MongoDB** |  |
| a. |  | Connecting Java with MongoDB. |  |
| **6** | 09/09/2023 | **Python and MongoDB** |  |
| a. |  | Connecting Python with MongoDB and inserting, retrieving, updating, and deleting. |  |
| **7** | 16/09/2023 | **Programs on Basic jQuery** |  |
| a. |  | jQuery Basic, jQuery Events |  |
| b. |  | jQuery Selectors, jQuery Hide and Show effects |  |
| c. |  | jQuery fading effects, jQuery Sliding effects |  |

# **Practical 1 - MongoDB Basics**

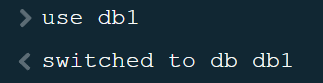
## Write a MongoDB query to create and drop database.

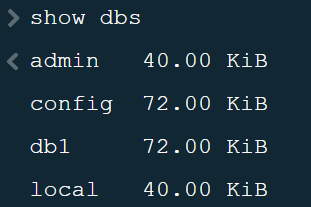
### Create : To create the database

Syntax : use database\_name

Code : use db1

Output:



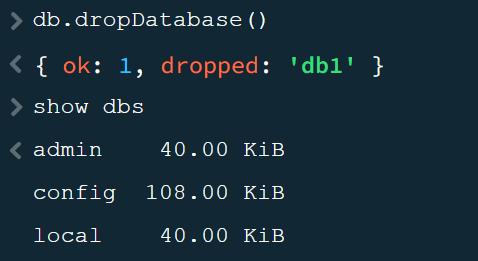


### Drop : Removes the current database, deleting the associated data files.

Syntax :- db.dropDatabase()

Code:- db.dropDatabase()

Output:-



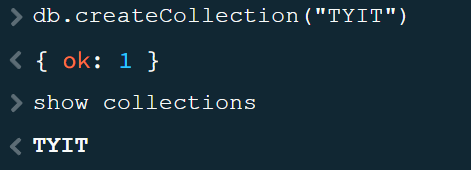
## Write a MongoDB query to create, display and drop collection

### Create : used to create collection in Database .

Syntax : db.createCollection(collection\_name)

Code : db.createCollection(“TYIT”)

Output :

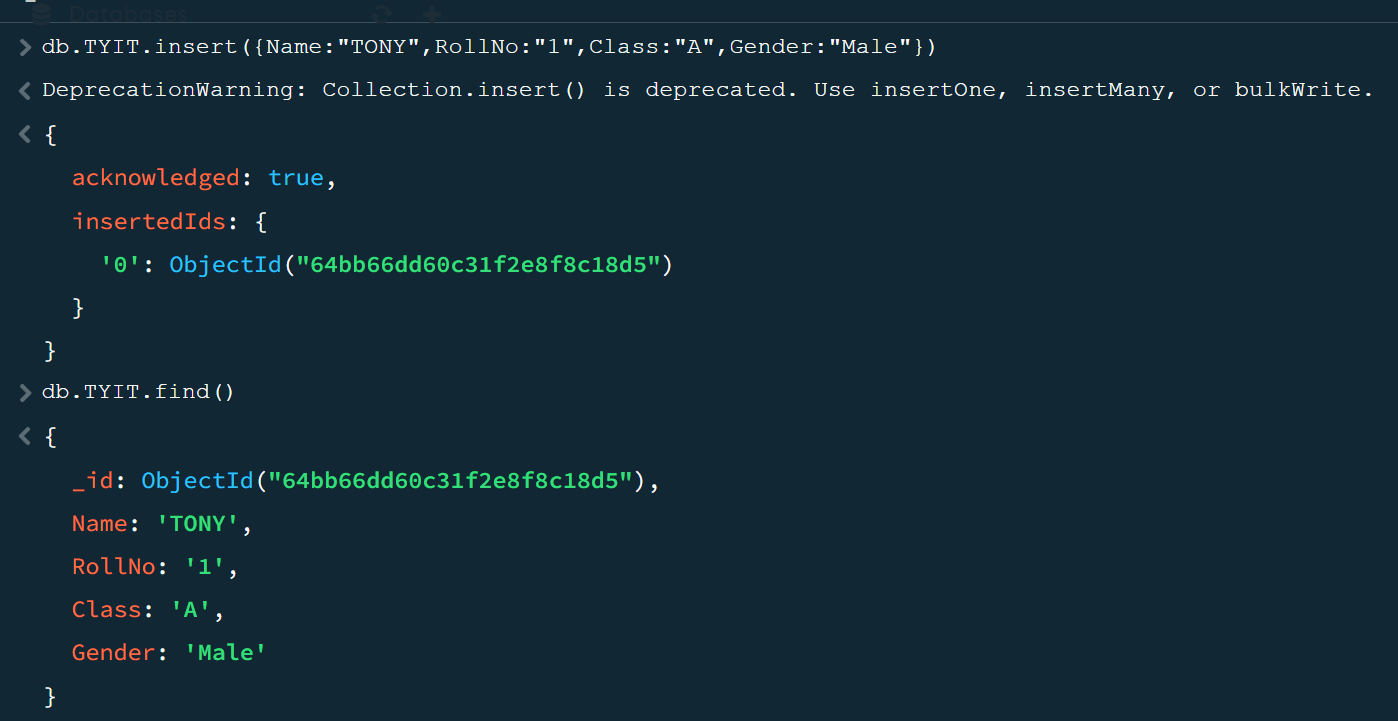


### Display

Syntax : db.collection\_name.find()

Code: db. TYIT.find()

Output :

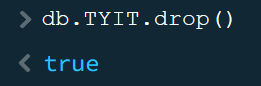


### Drop: used to drop a collection from the database.

Syntax : db.collection\_name.drop()

Code : db.TYIT.drop()

Output :

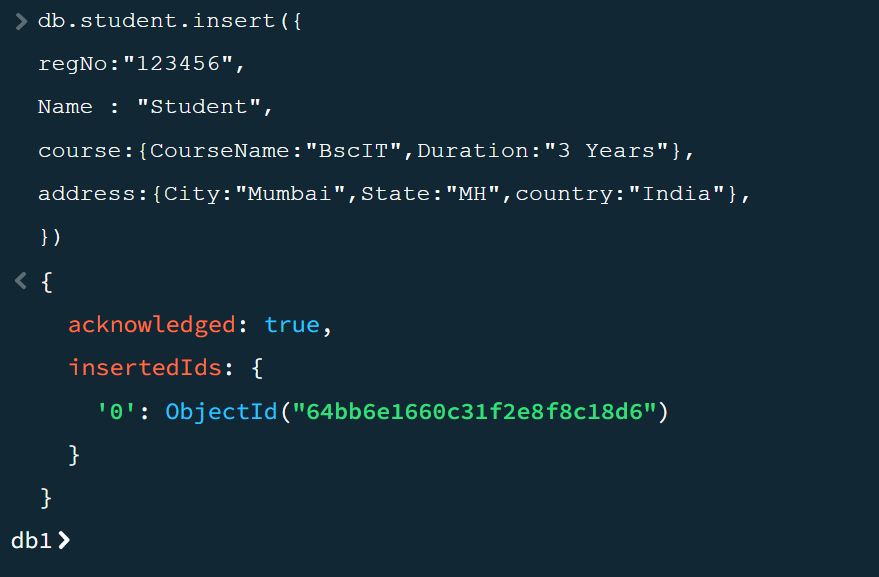


## Write a MongoDB query to insert, query, update and delete a document.

### To Insert Document

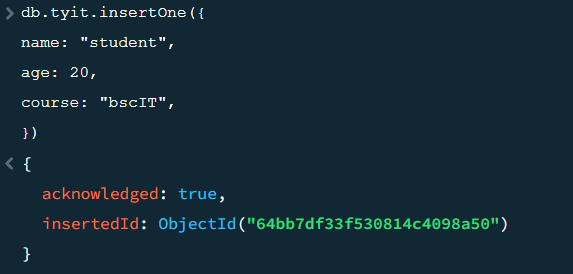
Syntax : db.collection\_name.insert(document)

Output:



### To Insert single Document

Syntax:- db.collection\_name.insertOne()



### Insert Multiple Document

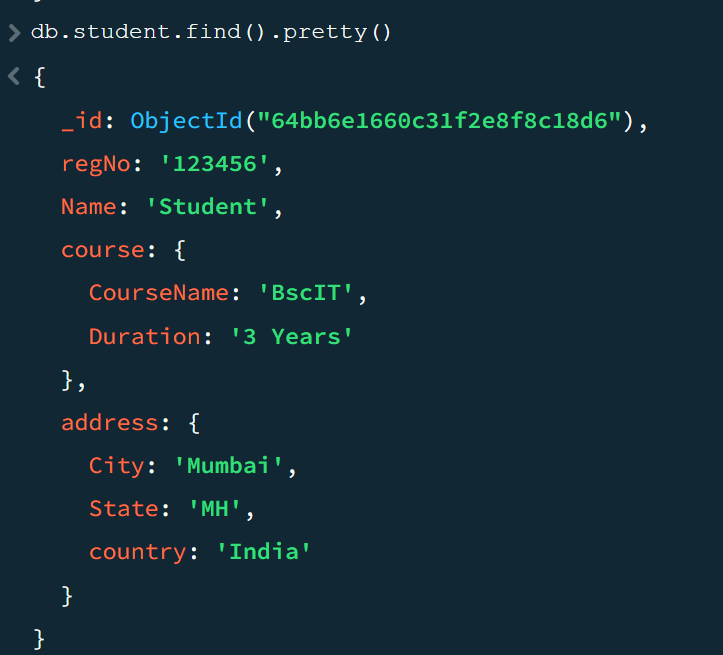
Syntax:- db.collection\_name.insertMany()



### To Query Document :- Use to retrieve [documents](https://www.mongodb.com/docs/manual/core/document/#std-label-bson-document-format) from a [collection](https://www.mongodb.com/docs/manual/core/databases-and-collections/#std-label-collections) i.e. query a collection for documents.

Syntax :- db.collection\_name.find().pretty()

Output:-



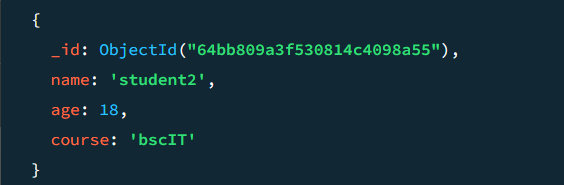
### To Update Document Update operations modify existing [documents](https://www.mongodb.com/docs/manual/core/document/#std-label-bson-document-format) in a [collection](https://www.mongodb.com/docs/manual/core/databases-and-collections/#std-label-collections).

Syntax :- db.collection\_name.update(SELECTION\_CRITERIA , UPDATED DATA )

### Update a single document

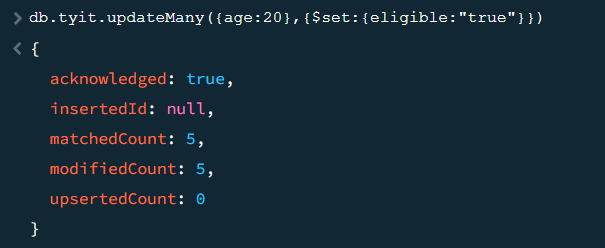
Syntax :- db.collection.updateOne();



Change after updating . 

### Update a multiple document

Syntax :- db.collection.updateMany();  
Output:-





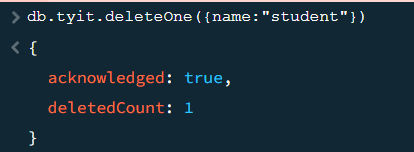
### To Delete Document :-

Syntax :- db.collection\_name.remove(DELETION\_CRITERIA )

### Remove one document:

Syntax: db.collection.deleteOne()

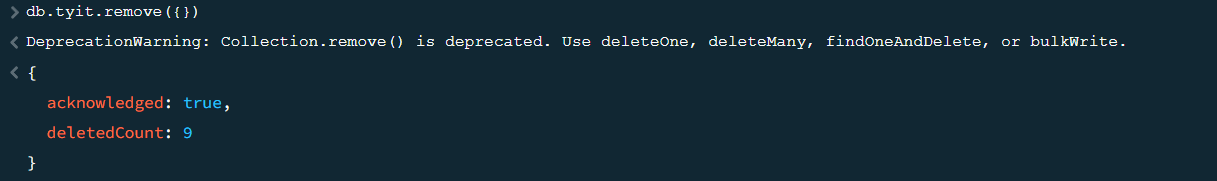
Output:-



### Remove all documents in the collection

Syntax: db.collection.remove({})

Output:-



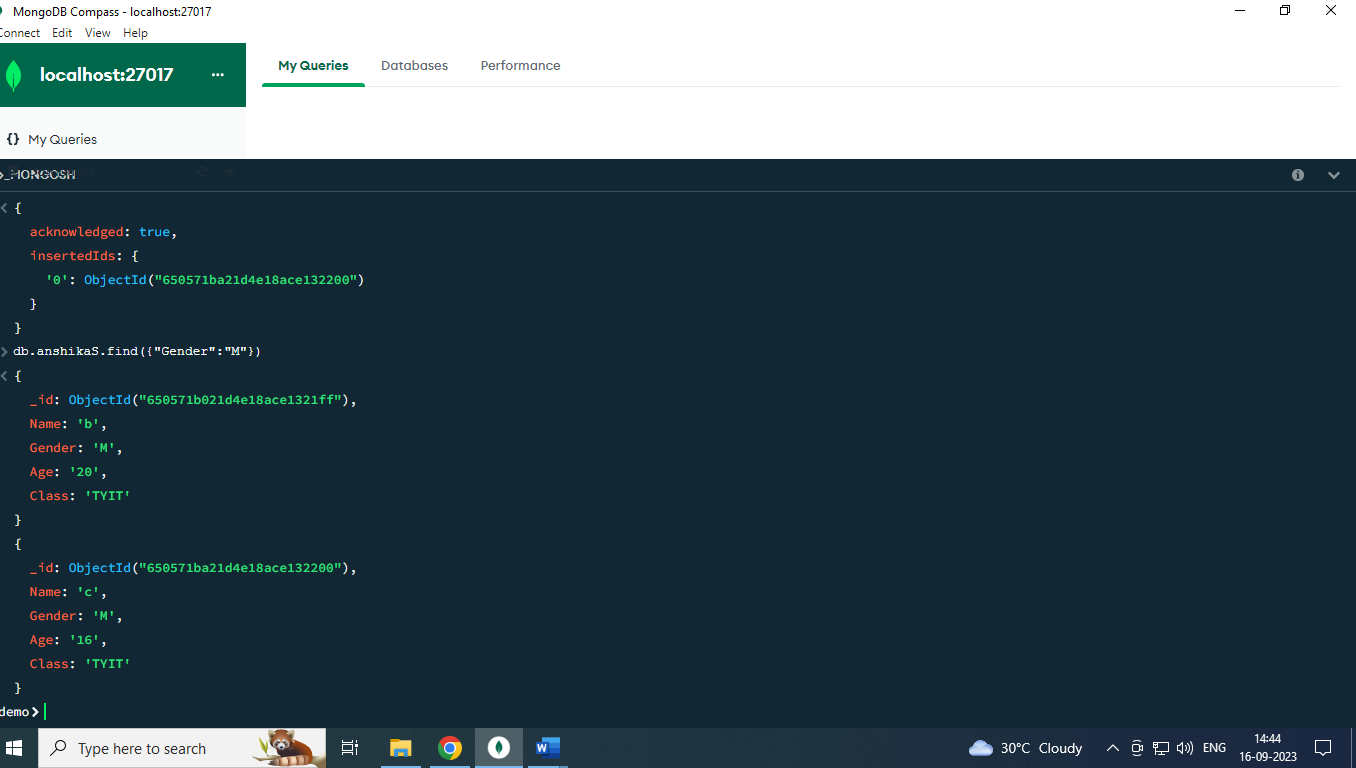
# **Practical 2: Simple Queries with MongoDB**

## Selector –

Syntax - db.collection\_name.find({“Key”:”Value”})

Code - db.anshikaS.find({“Gender”:”M”})

Output -

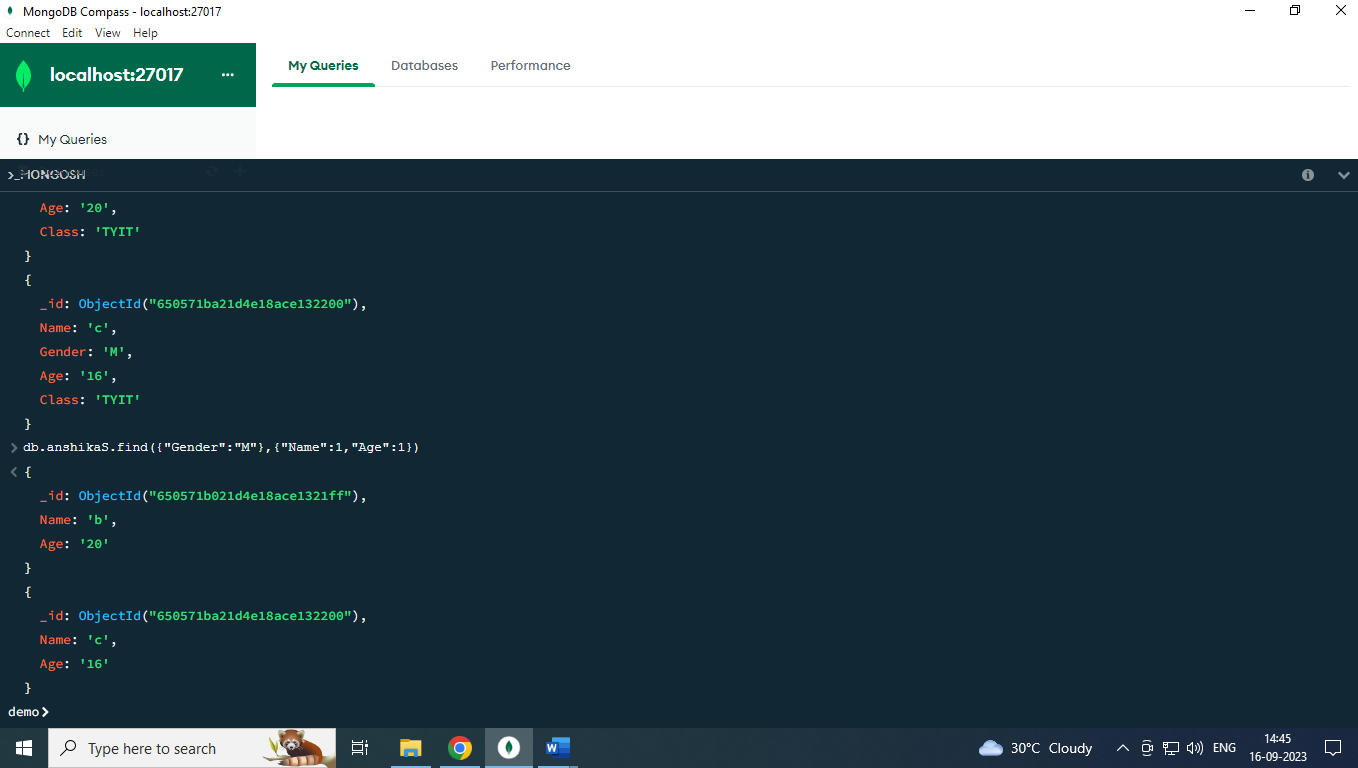


## Projector –

Syntax - db.collection\_name.find({“Key”:”Value”},{“Key”:Value,”Key”:Value})

Code - db.anshikaS.find({“Gender”:”M”},{“Name”:1,”Age”:1})

Output -

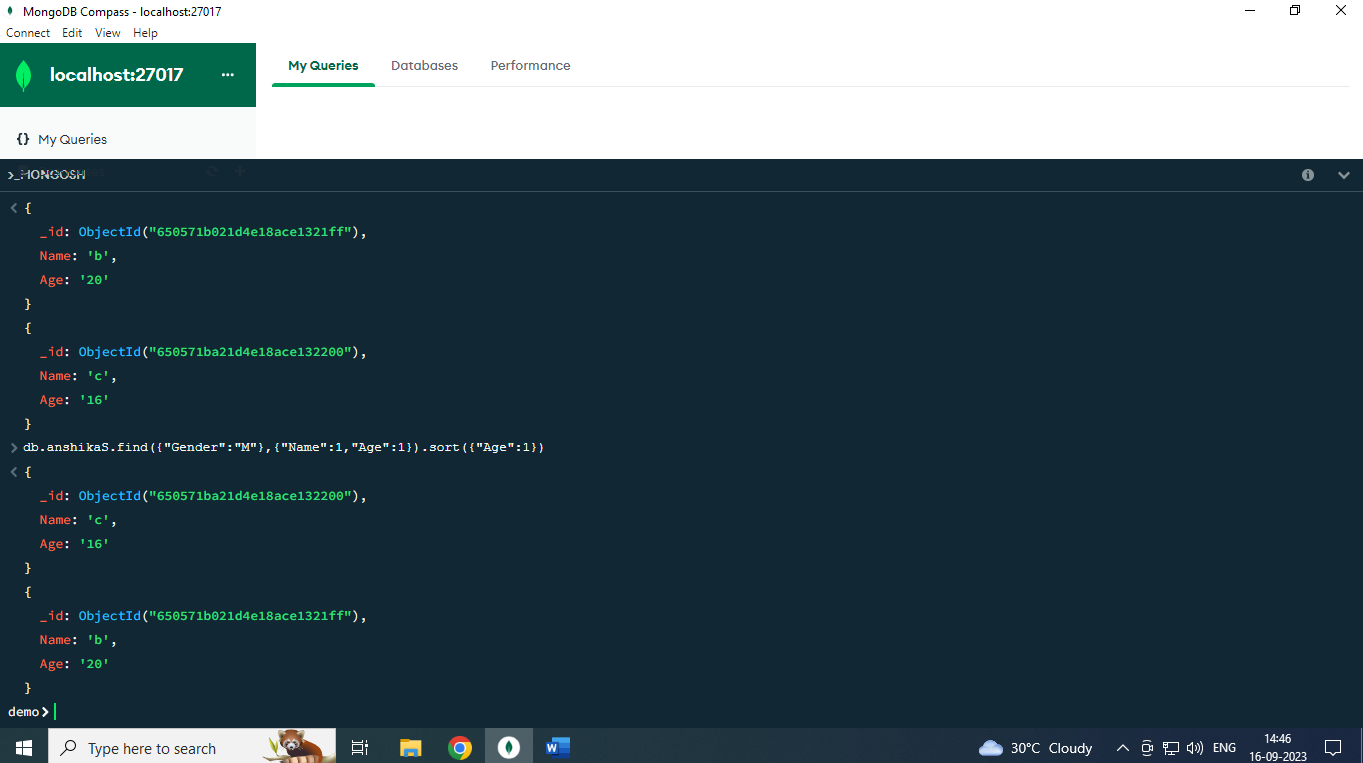


## Sort()-

Syntax - db.collection\_name.find({“Key”:”Value”},{“Key”:Value,”Key”:Value}).sort({“Key”:Value})

Code - db.anshikaS.find({“Gender”:”M”},{“Name”:1,”Age”:1}).sort({“Age”:1})

Output -

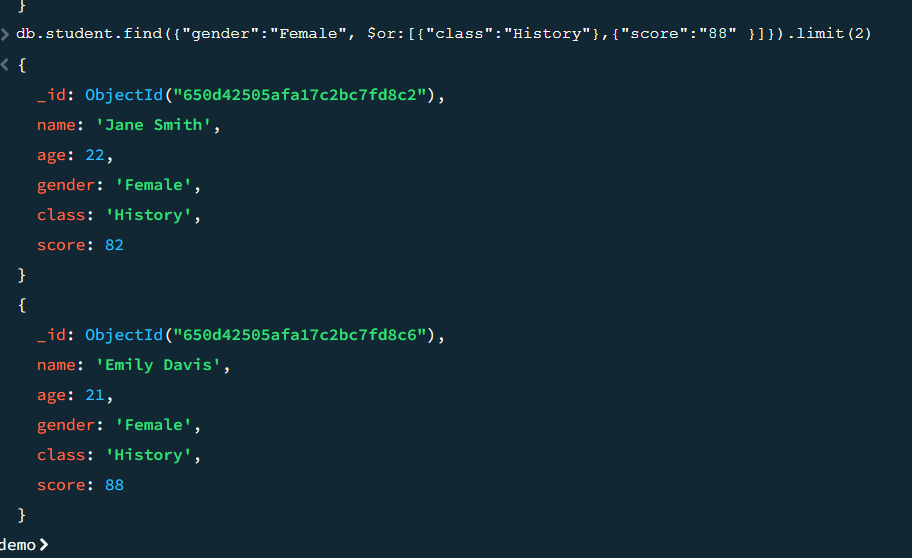


## Limit() –

Syntax - db.collection\_name.find({“Key”:”Value”, $or:[{“Key”:Value”},{“Key”:Value}]}).limit(Value)

Code - db.student.find({“gender”:”Female”, $or:[{“class”:History”},{“score”:”88” }]}).limit(2)

Output -



## Skip() –

Syntax - db.collection\_name.find({“Key”:”Value”, $or:[{“Key”:Value”},{“Key”:Value}]}).limit(Value).skip(Value)

Code - db.student.find({“gender”:”Female”, $or:[{“class”:History”},{“score”:”88” }]}).limit(2).skip(2)

Output -



## findOne() –

Syntax - db.collection\_name.findOne({“Key”:”Value”, {“Key”:Value,“Key”:Value})

db.collection\_name.findOne()

Code - db.student.findOne({“gender”:”Female”, {“name”:1,“age”:1})

db.student.findOne()

#### Output -

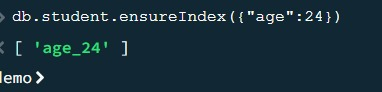


## ensureIndex –

Syntax - db.collection\_name.ensureIndex({“Key”:”Value”})

Code - db.student.ensureIndex({“age”:”24”})

Output -



## pretty()-

syntax - db.collection\_name.find().pretty()

code - db.student.find().pretty()

#### output -



## conditional operators –

Syntax -

$lt and $lte –

db.collection\_name.find({“Key”:{“$lt”:Value}})

db.collection\_name.find({“Key”:{“$lte”:Value}})

$gt and $gte –

db.collection\_name.find({“Key”:{“$gt”:Value}})

db.collection\_name.find({“Key”:{“$gte”:Value}})

$in and $nin

db.collection\_name.find({“Key”:{“$in”:[“Value”,”Value”]}})

db.collection\_name.find({“Key”:{“$nin”:[“Value”,”Value”]}})

code -

$lt and $lte –

db.student.find({“age”:{“$lt”:20 }})

db.student.find({“age”:{“$lte”:20 }})

$gt and $gte –

db.student.find({“age”:{“$gt”:28 }})

db.student.find({“age”:{“$gte”:28 }})

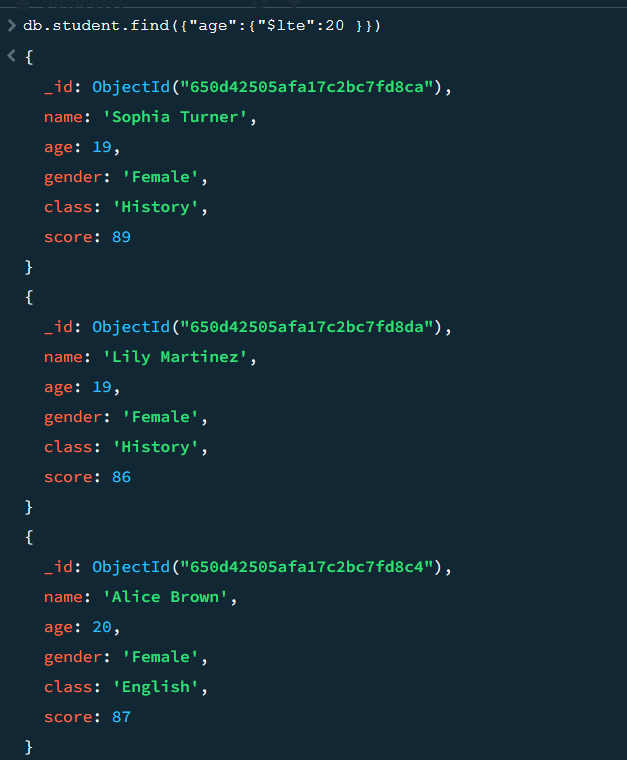
$in and $nin –

db.student.find({“class”:{“$in”:[“History”,”English”]}})

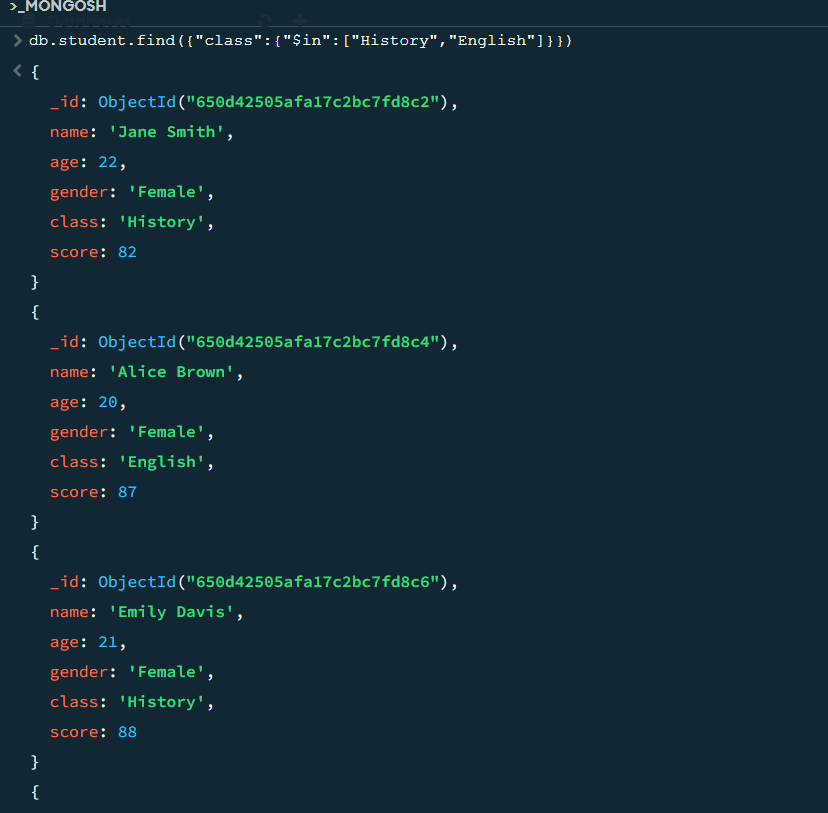
db.student.find({“class”:{“$nin”:[“History”,”English”]}})

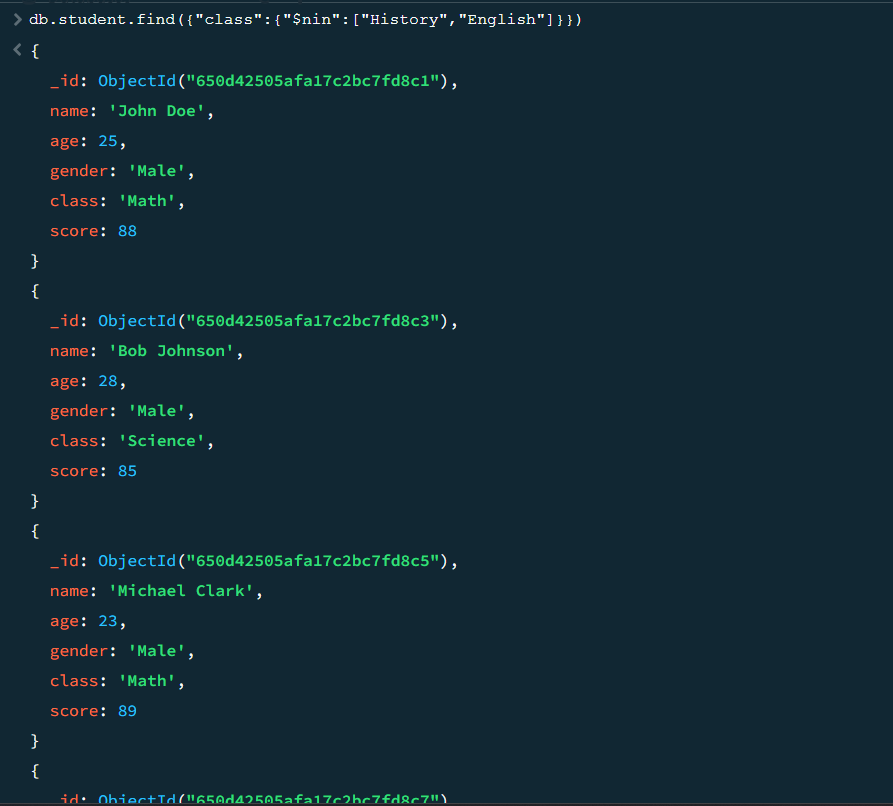
#### output -











# **Practical 3 – Implementing aggregation**

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

#### Syntax :

sum – {$sum:[<expression1>,<expression>…]}

Avg - {$avg:[<expression1>…]}

Min - {$min:[<expression1>…]}

Max - {$max:[<expression1>…]}

#### Code :

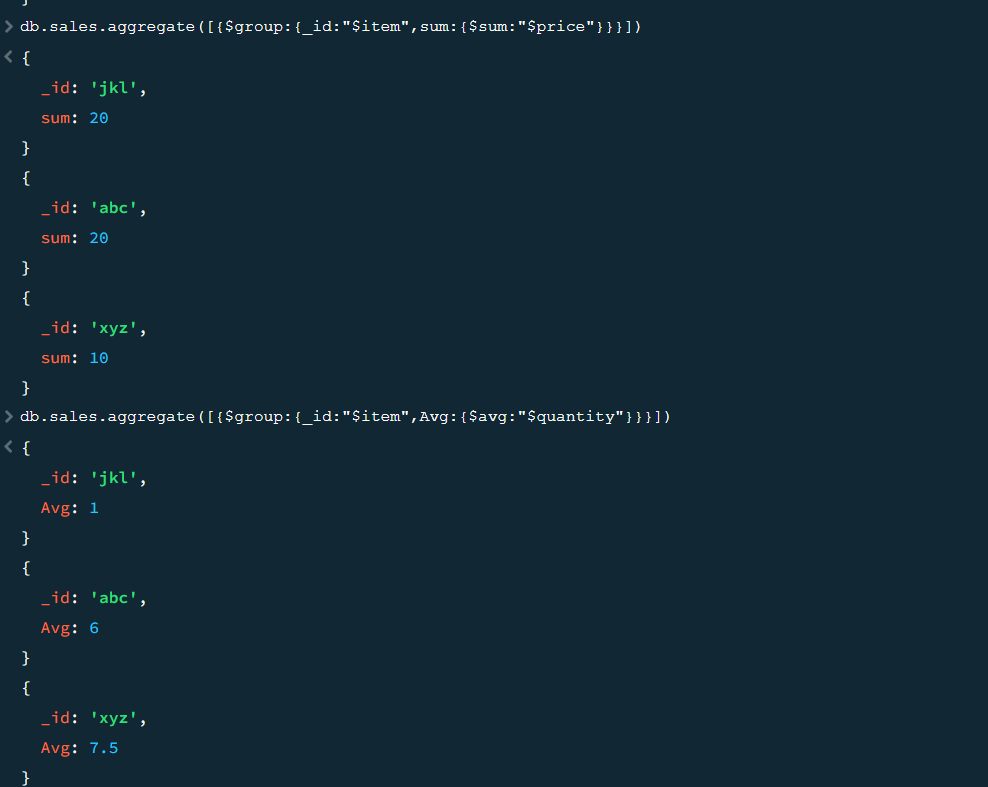
db.sales.aggregate([{$group:{\_id:”$item”,sum:{$sum:“$price”}}}])

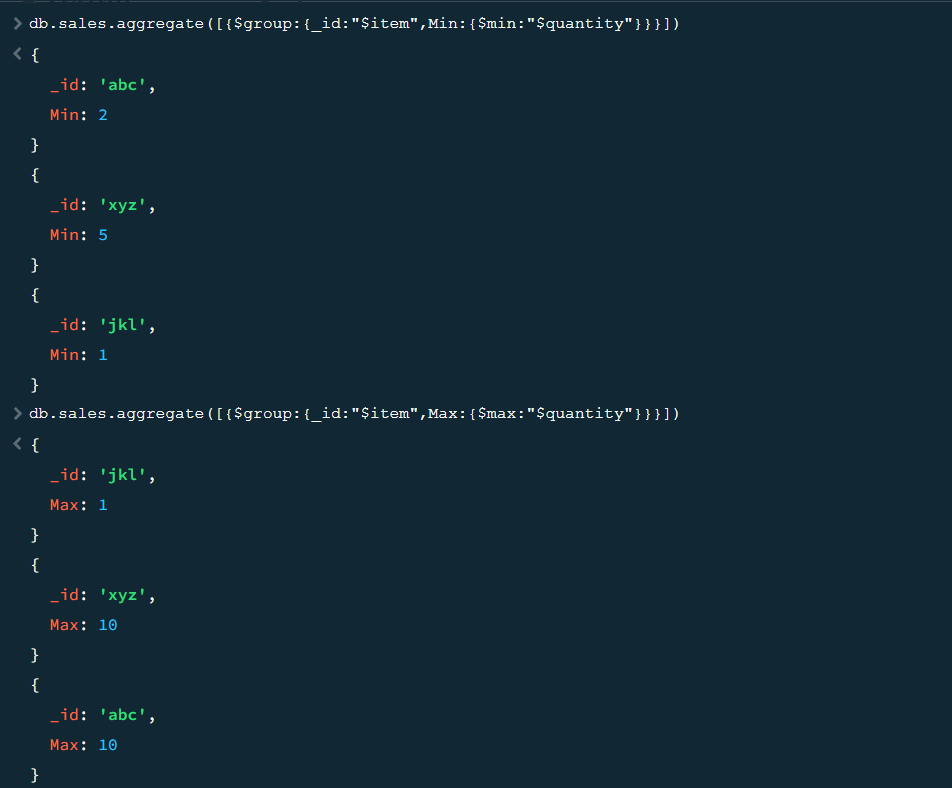
db.sales.aggregate([{$group:{\_id:”$item”,Avg:{$avg:“$quantity”}}}])

db.sales.aggregate([{$group:{\_id:”$item”,Min:{$min:“$quantity”}}}])

db.sales.aggregate([{$group:{\_id:”$item”,Max:{$max:“$quantity”}}}])

#### output –





## Write a MongoDB query to use push and addToSet expression.

#### Syntax :

Push – {$push:<expression>}

addToSet – {$addToSet:<expression>}

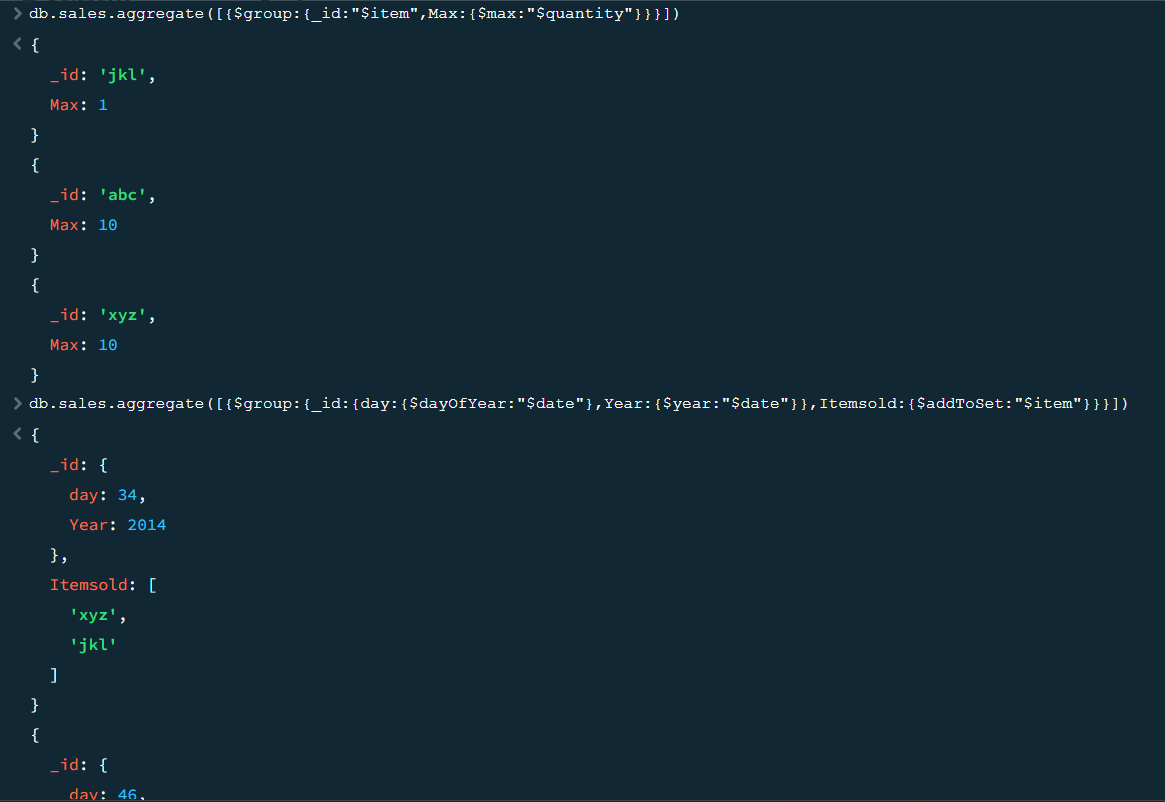
#### Code :

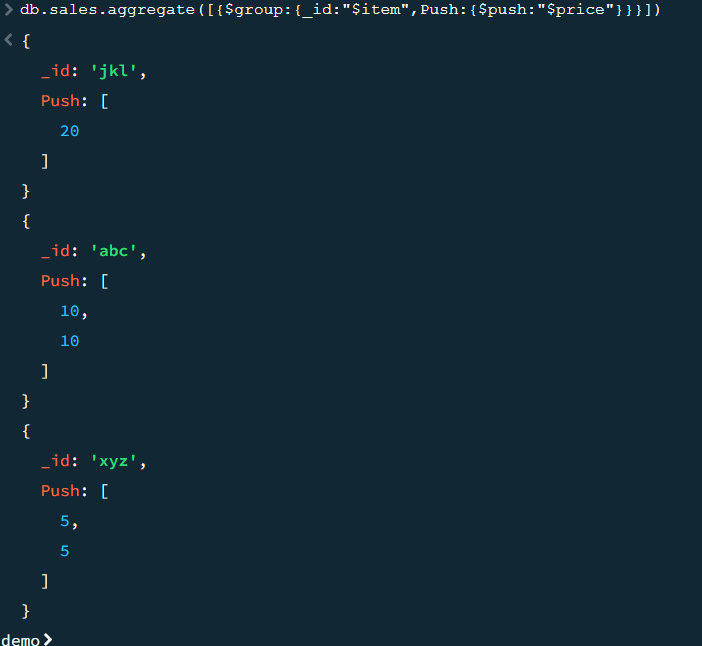
db.sales.aggregate([{$group:{\_id:”$item”,AddToSet:{$addToSet:”$price”}}}])

db.sales.aggregate([{$group:{\_id:{day:{$dayOfYear:”$date”},Year:{$year:”$date”}},Itemsold:{$addToSet:”$item”}}}])

db.sales.aggregate([{$group:{\_id:”$item”,Push:{$push:”$price”}}}])

#### Output-





## Write a MongoDB query to use first and last expression.

#### Syntax :

first – {$first:<expression>}

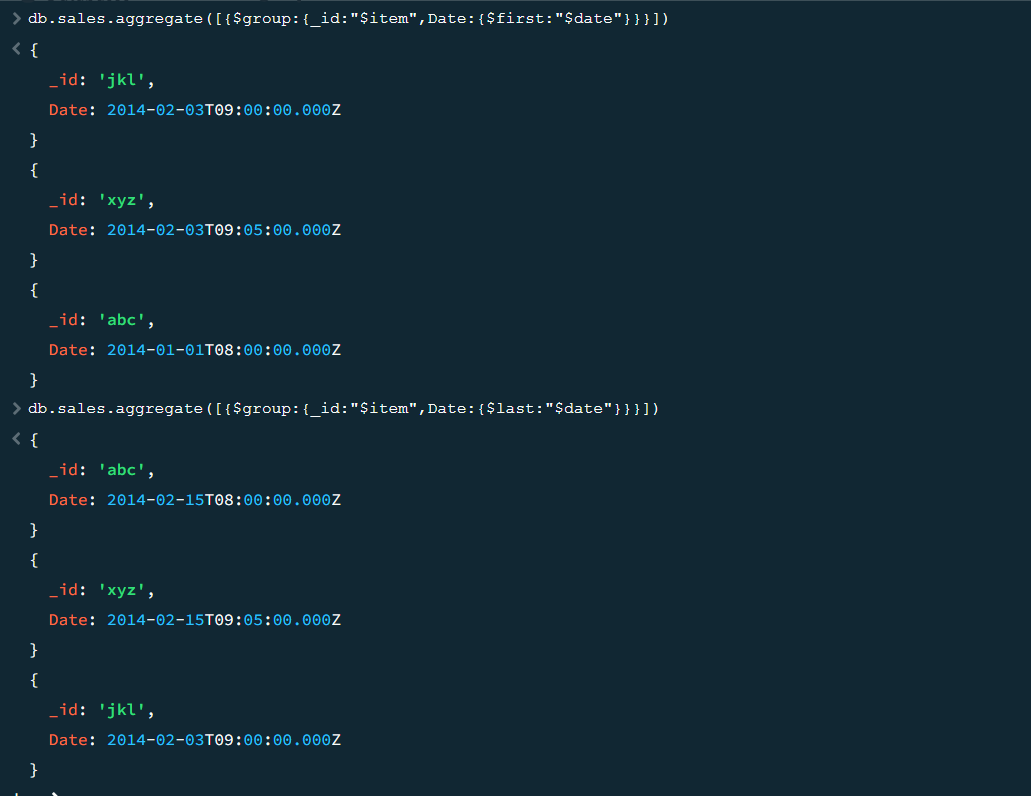
Last – {$last:<expression>}

#### Code :

db.sales.aggregate([{$group:{\_id:”$item”,Date:{$first:”$date”}}}])

db.sales.aggregate([{$group:{\_id:”$item”,Date:{$last:”$date”}}}])

#### output :

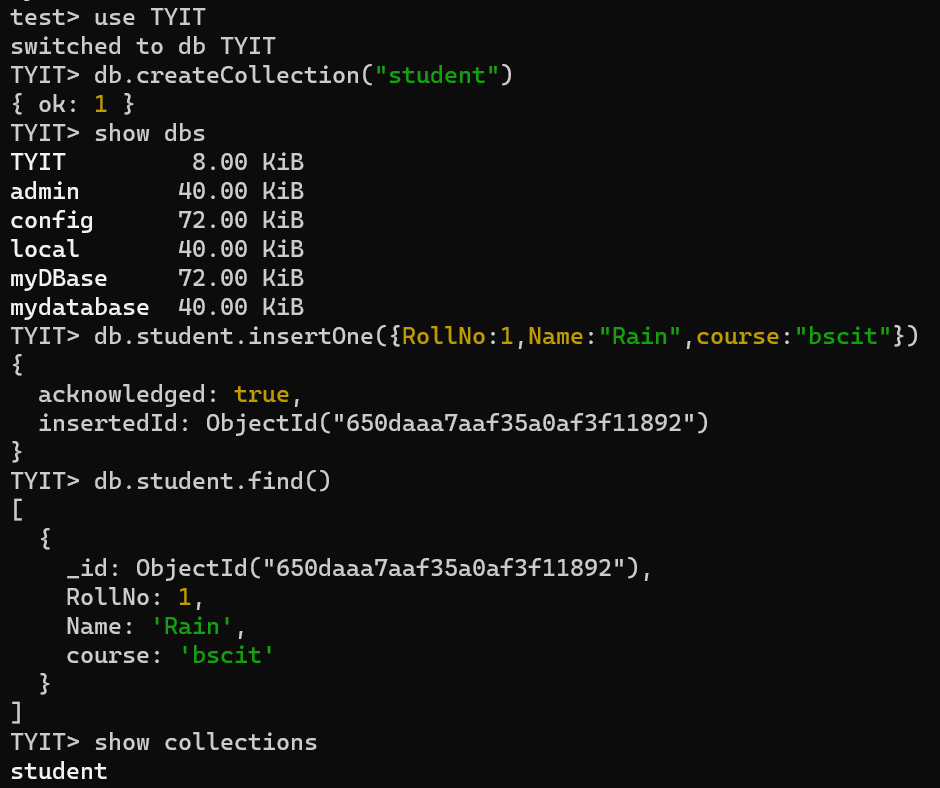


# **Practical 4 - Backup and Restore**

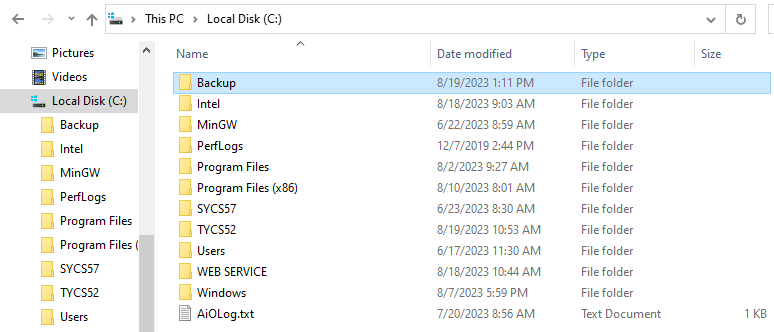
## Write a MongoDB query to create a backup of existing database.

#### Steps :

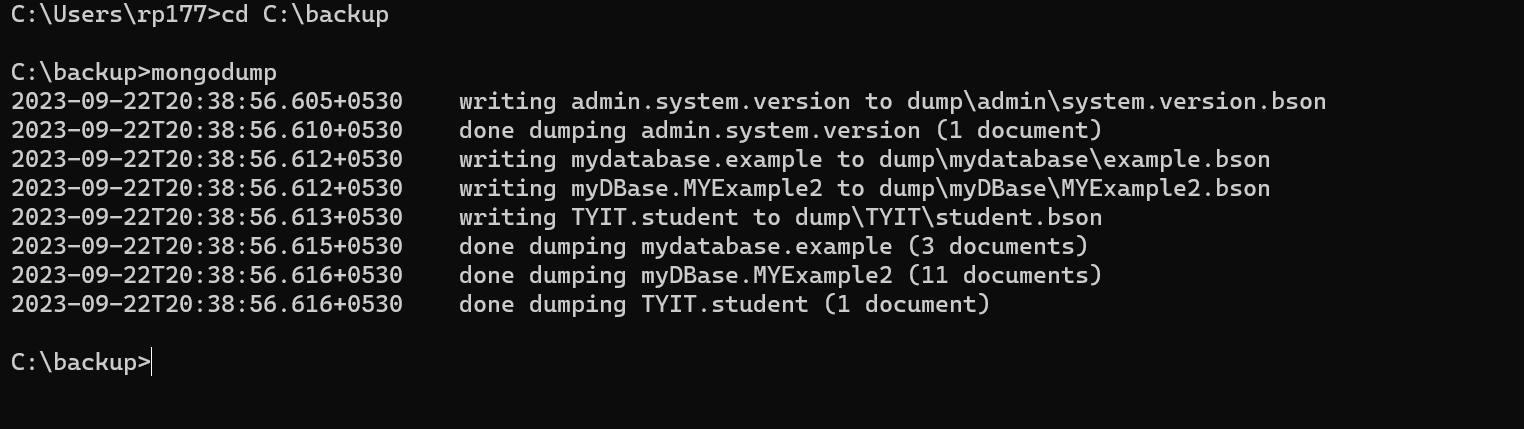
Step 1 -Creating database and collection



Step 2 – Create a folder named ‘backup’ in c: drive

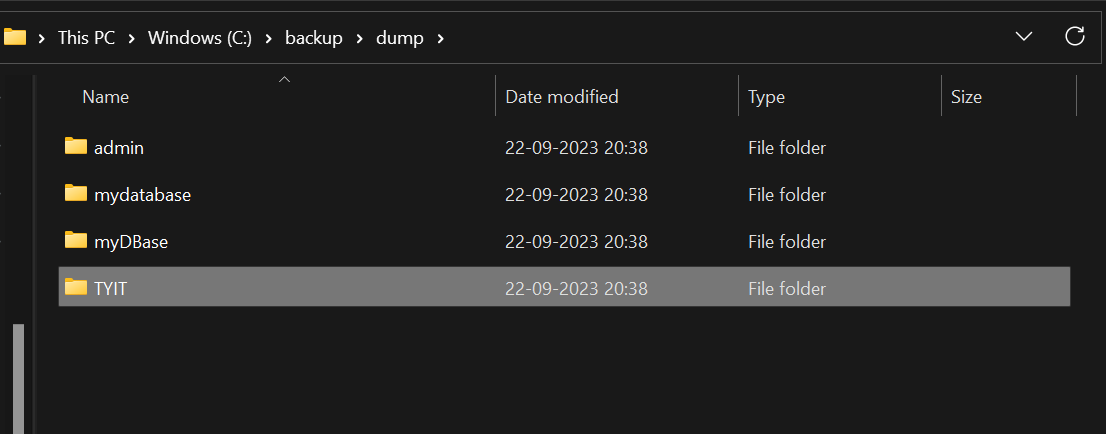


Step 3 – Change the directory in cmd to backup folder’s path and write mongodump



#### Output :

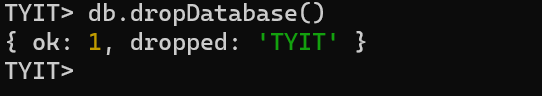
Backup Created –



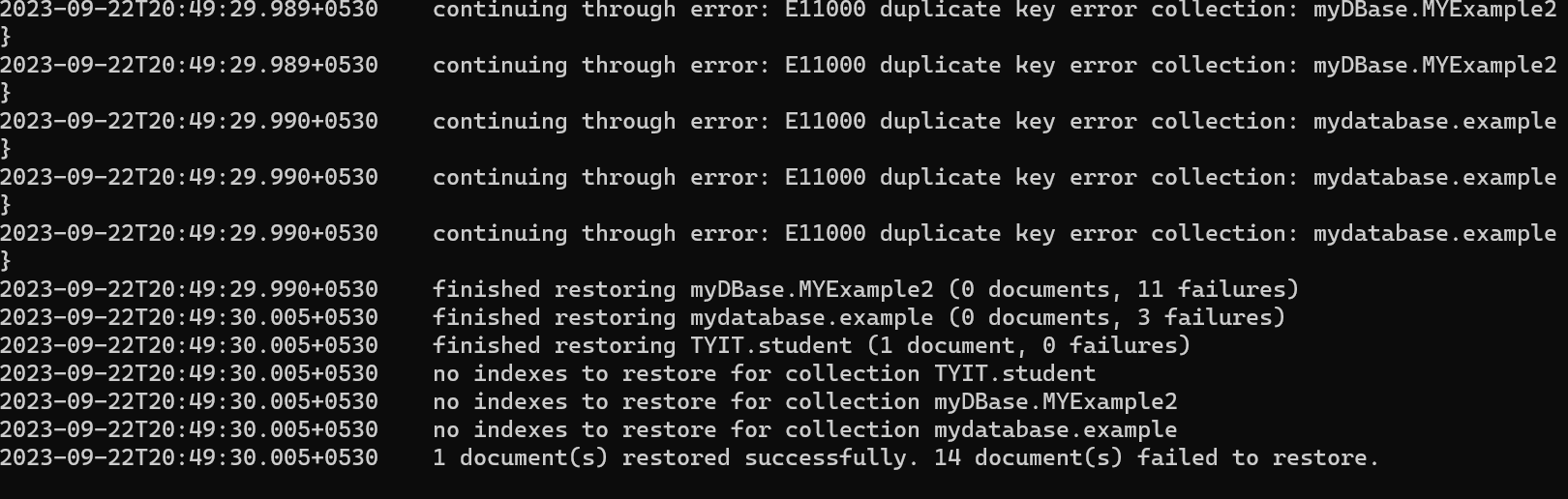
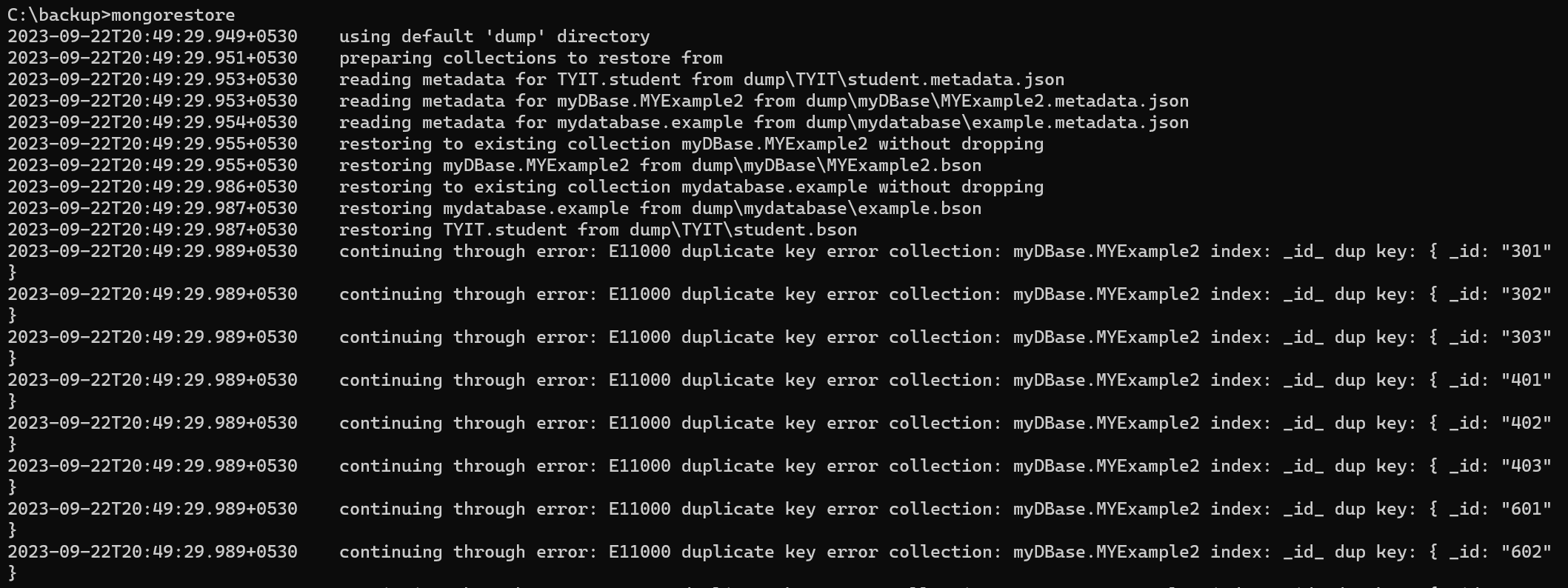
## Write a MongoDB query to restore database from the backup.

#### Steps :

Step 1 – drop the existing datadase

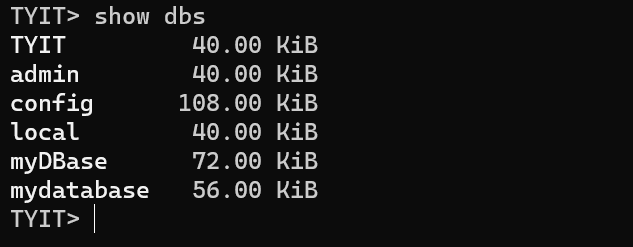


Step 2 – In the cmd, within backup directory write the command mongorestore



#### Output :

Successfully Restored



# **Practical 5 - Java and MongoDB**

## Connecting Java with MongoDB.

### Code :

package mongoconnection1;

import com.mongodb.MongoClient;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.MongoIterable;

public class Mongoconnection1 {

public static void main(String[] args) {

//Access a specific database

try ( // Connect to MongoDB server running on localhost at default port (27017)

MongoClient mongoClient = new MongoClient("localhost", 27017)) {

//Access a specific database

MongoDatabase database = mongoClient.getDatabase("admin");

System.out.println("Connected to MongoDB!");

MongoIterable<String> databaseNames = mongoClient.listDatabaseNames();

// Print the names of all databases

for (String dbName : databaseNames) {

System.out.println("Database Name: " + dbName);

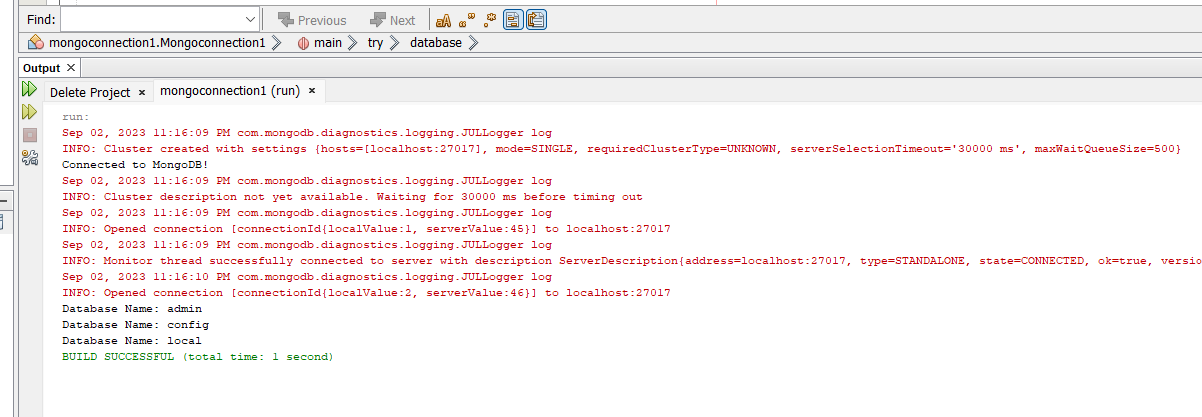
}

// Close the MongoDB client

}

}

### Output :



# **Practical 6 - Python and MongoDB**

## Connecting Python with MongoDB and inserting, retrieving, updating and deleting.

### Inserting & Retrieving

#### Source code :

from pymongo import MongoClient

#creating pymongo client

client = MongoClient('localhost',27017)

#getting database instance

db= client['mydatabase']

#creating collection

coll = db['example']

#Inserting document into a collection

data = [ {"\_id": "101", "name": "Ram","age": "26", "city": "Hyderabad"},

{"\_id": "102", "name": "Rahim", "age": "27", "city": "Bangalore"},

{"\_id": "103", "name": "Robert", "age": "28", "city": "Mumbai"}]

res = coll.insert\_many(data)

print("Data inserted ......")

print(res.inserted\_ids)

#Retrieving the first record using thefind\_one() method

print("First record of the collection: ")

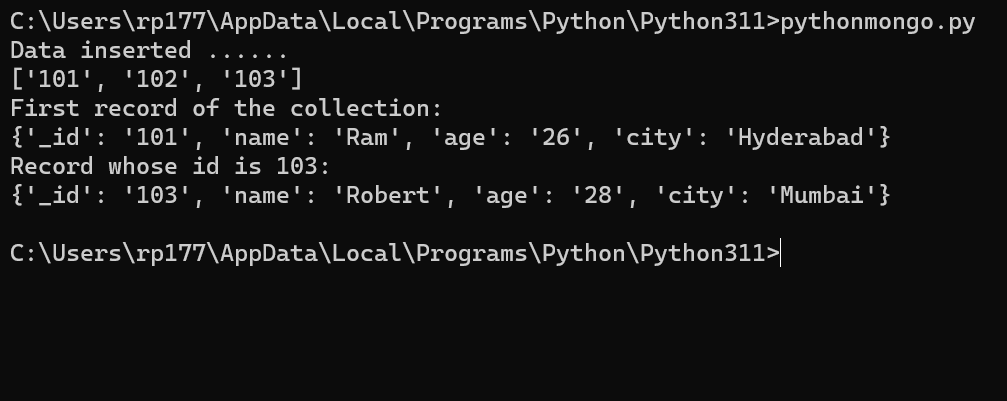
print(coll.find\_one())

#Retrieving a record with is 103 using #thefind\_one()method

print("Record whose id is 103: ")

print(coll.find\_one({"\_id": "103"}))

#### Output :



### 2) Update one & Update Many

#### Source code :

from pymongo import MongoClient

#Creating a pymongo client

client = MongoClient('localhost', 27017)

#Getting the database instance

db = client['myDBase']

#Creating a collection

coll = db['MYExample2']

#Inserting document into a collection

data = [

{"\_id": "401", "name": "Ram", "age": "26", "city":"Hyderabad"},

{"\_id":"402", "name": "Rahim", "age": "27", "city": "Bangalore"},

{"\_id":"403", "name": "Robert", "age": "28", "city": "Mumbai"}]

res = coll.insert\_many(data)

print("Data inserted ......")

#Retrieving all the records using the find() method

print("Documents in the collection: ")

for doc1 in coll.find():

print(doc1)

#updating one

coll.update\_one({"\_id":"302"},{"$set":{"city":"Visakhapatnam"} })

#Retrieving all the records using the find() method

print("Documents in the collection after update\_one operation: ")

for doc2 in coll.find():

print(doc2)

#updating many

coll.update\_many({},{"$set":{"city":"Visakhapatnam"} })

print("Documents in the collection after update\_many operation: ")

for doc3 in coll.find():

print(doc3)

#### Output :



### 3) Delete One & Delete Many

#### Source code :

from pymongo import MongoClient

#Creating a pymongo client

client = MongoClient('localhost', 27017)

#Getting the database instance

db = client['myDBase']

#Creating a collection

coll = db['MYExample2']

#Inserting document into a collection

data = [

{"\_id": "601", "name": "Ram", "age": "26", "city":"Hyderabad"},

{"\_id":"602", "name": "Rahim", "age": "27", "city": "Bangalore"},

{"\_id":"603", "name": "Robert", "age": "28", "city": "Mumbai"},

{"\_id": "604", "name": "Rain", "age": "25", "city":"Pune"},

{"\_id":"605", "name": "Ryu", "age": "23", "city": "Delhi"},

{"\_id":"606", "name": "Sora", "age": "26", "city": "Chennai"}]

res = coll.insert\_many(data)

print("Data inserted ......")

print(res.inserted\_ids)

#deleting one

coll.delete\_one({"\_id":"606"})

#Retrieving all the records using the find() method

print("Documents in the collection after update\_one operation: ")

for doc2 in coll.find():

print(doc2)

#deleting multiple documents

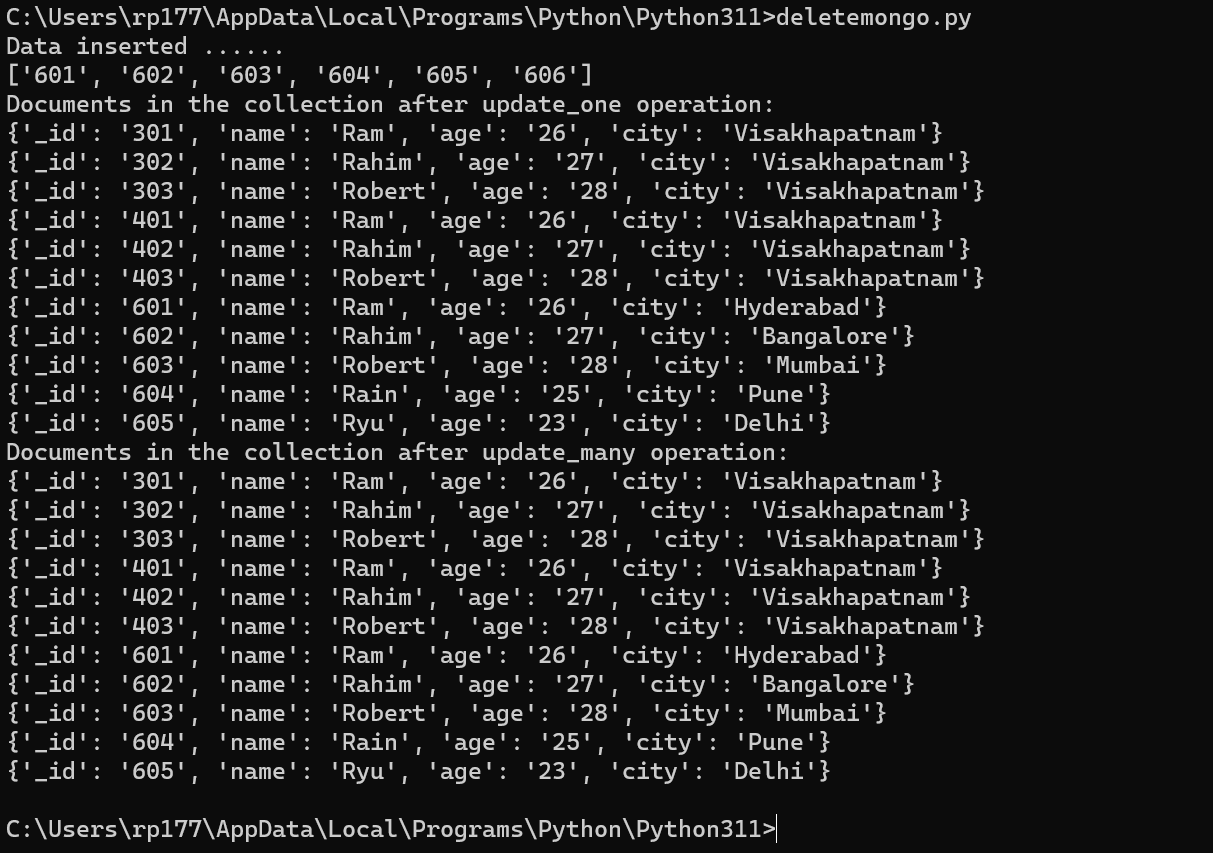
coll.delete\_many({"age":{"gt":"26"} })

print("Documents in the collection after update\_many operation: ")

for doc3 in coll.find():

print(doc3)

#### Output :



# **Practical 7 - Programs on Basic jQuery**

## jQuery Basic, jQuery Events

### jQuery basic

#### Code :

<!DOCTYPE html>

<html>

<head>

<title>The jQuery Example</title>

</head>

<body>

<p>This is 1st Paragraph.</p>

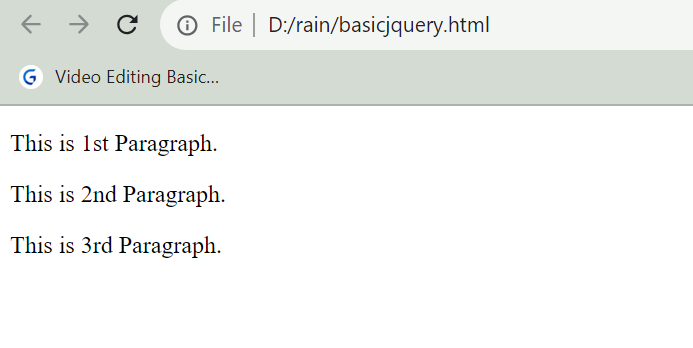
<p>This is 2nd Paragraph.</p>

<p>This is 3rd Paragraph.</p>

</body>

</html>

#### Output :



### jQuery Events – click event

#### Code :

<!DOCTYPE html>

<html>

<head>

<script src="https://code.jquery.com/jquery-3.7.1.js"></script>12:26 09-09-2023

<script>

$(document).ready(function(){

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

$(this).hide();

});

});

</script>

</head>

<body>

<p>If you click on me, I will disappear.</p>

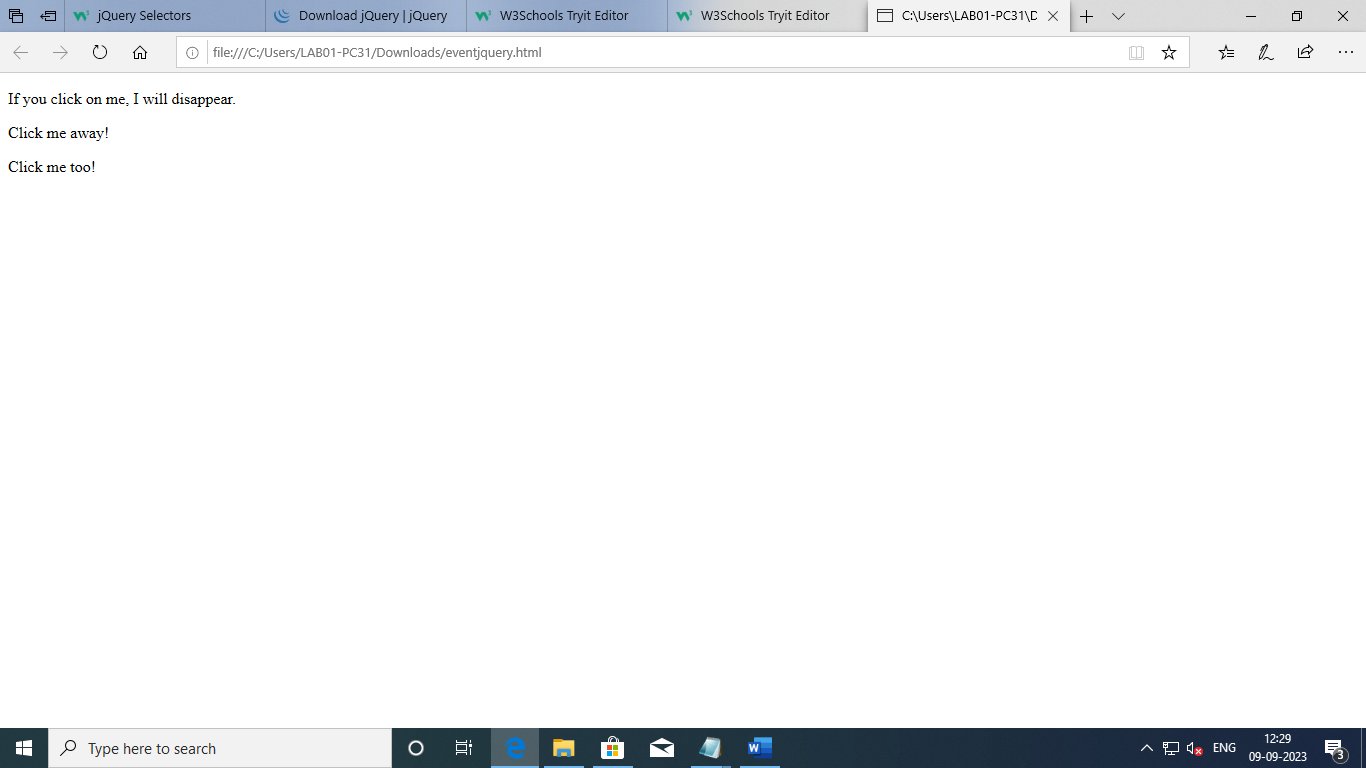
<p>Click me away!</p>

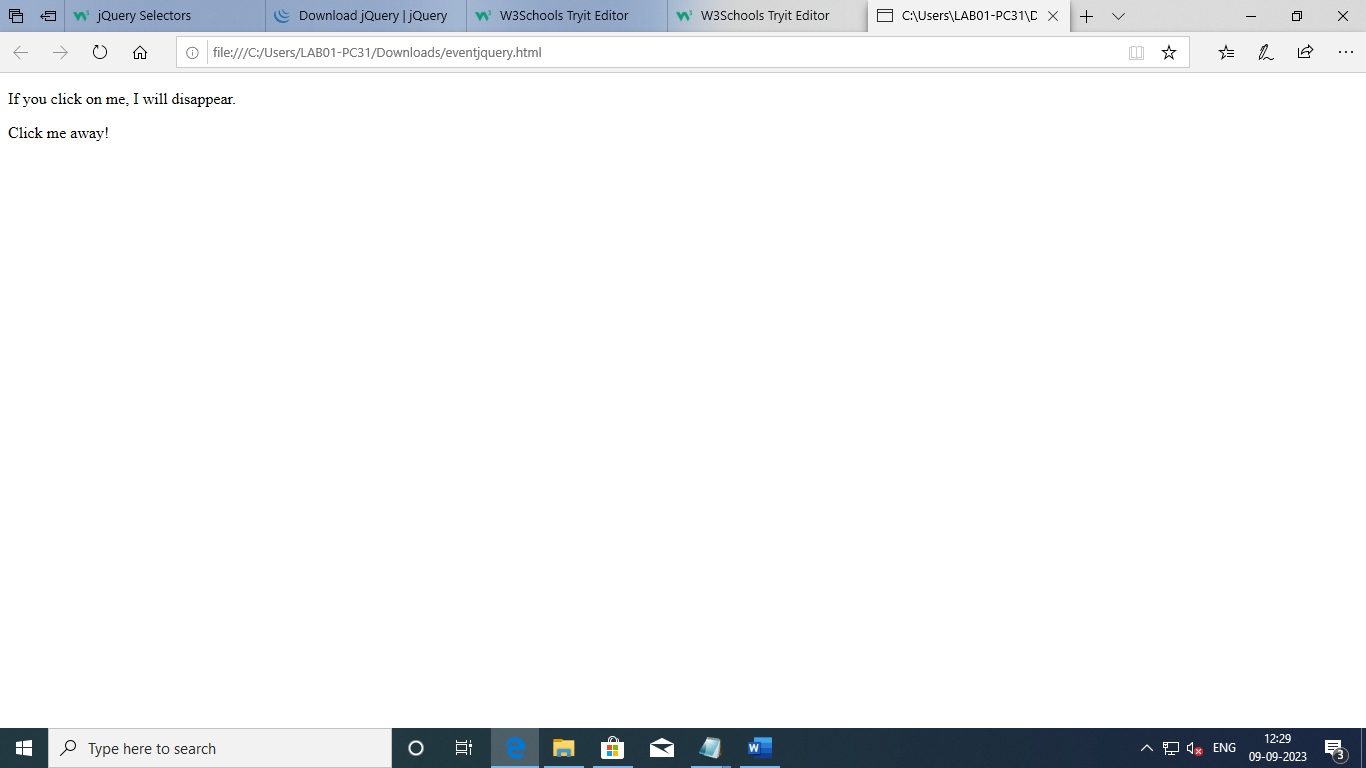
<p>Click me too!</p>

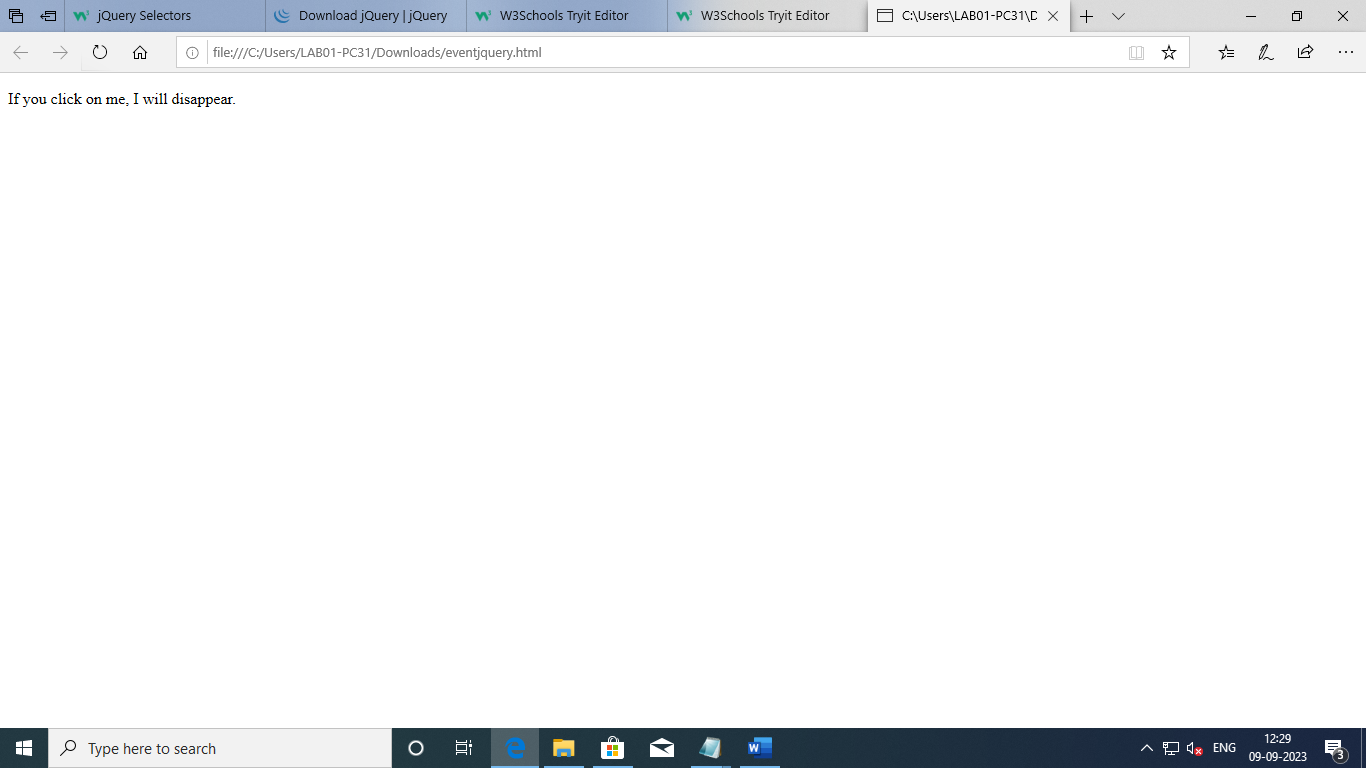
</body>

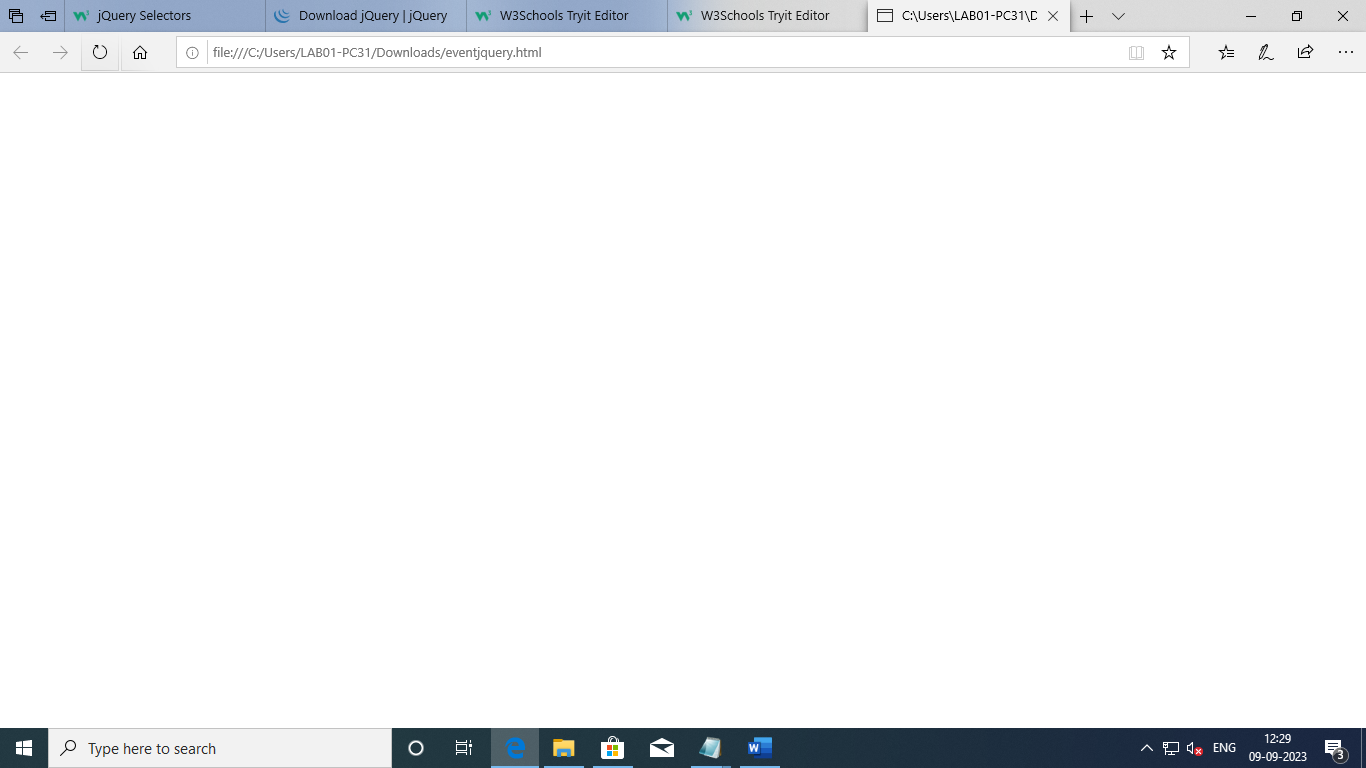
</html>

#### Output :









### double-click event

#### Code :

<html>

<head>

<script src="https://code.jquery.com/jquery-3.7.1.js"></script>

<script>

$(document).ready(function(){

$("p").dblclick(function(){

$(this).hide();

});

});

</script>

</head>

<body>

<p>If you double-click on me, I will disappear.</p>

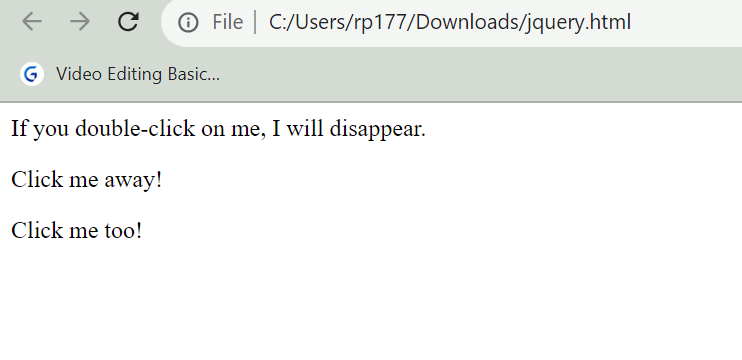
<p>Click me away!</p>

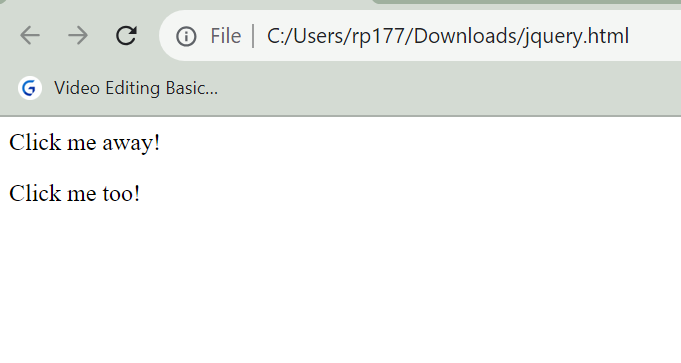
<p>Click me too!</p>

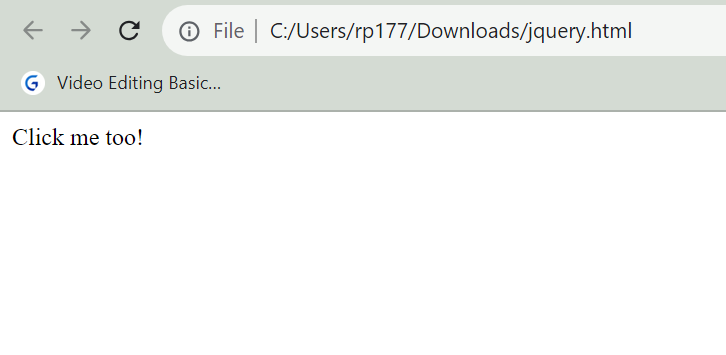
</body>

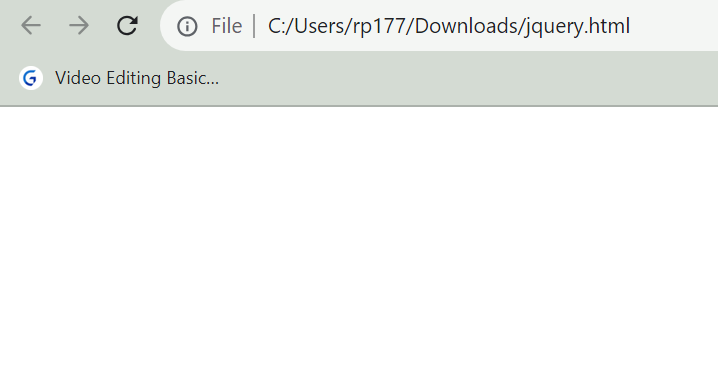
</html>

#### Output :









## jQuery Selectors, jQuery Hide and Show effects

### jQuery Selectors - Selector

#### Code :

<!DOCTYPE html>

<html>

<head>

<script src="https://code.jquery.com/jquery-3.7.1.js"></script>

<script>

$(document).ready(function(){

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

$("p").hide();

});

});

</script>

</head>

<body>

<h2>This is a heading</h2>

<p>This is a paragraph.</p>

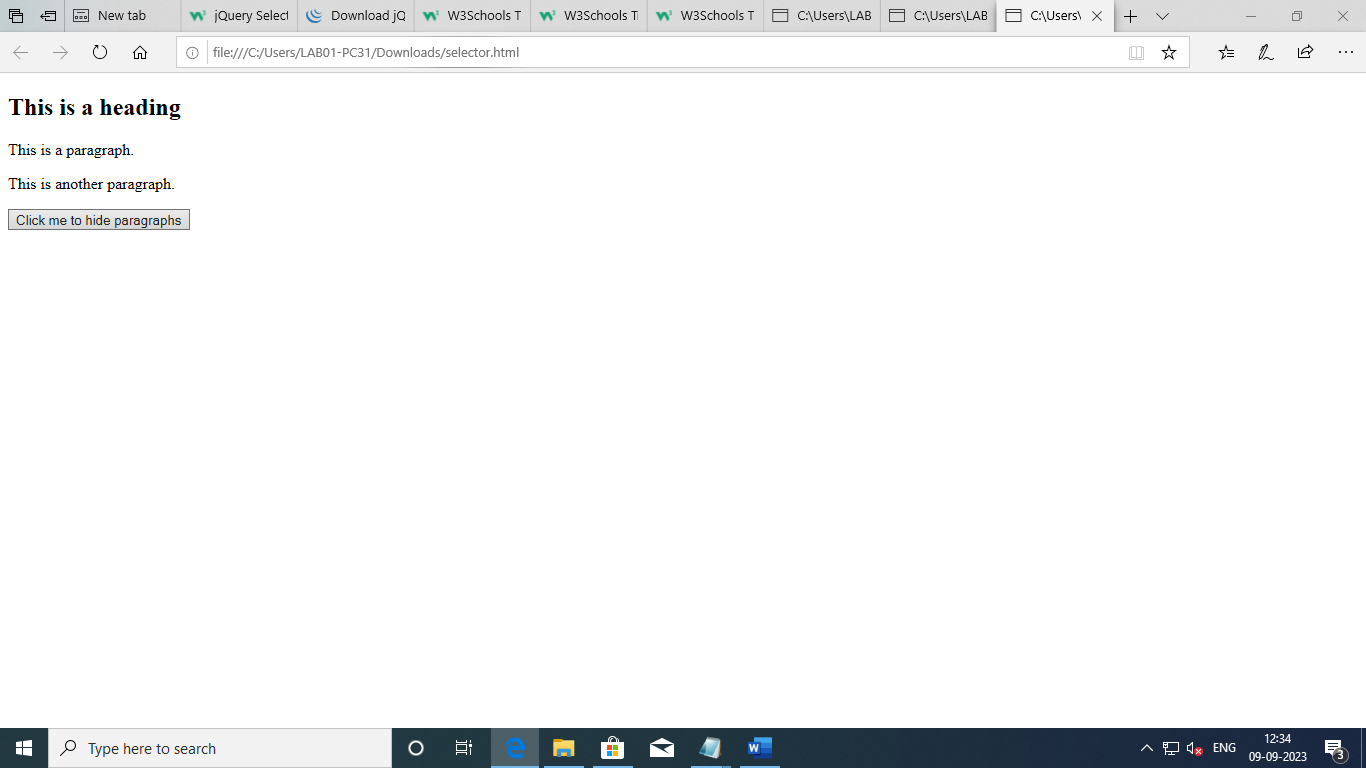
<p>This is another paragraph.</p>

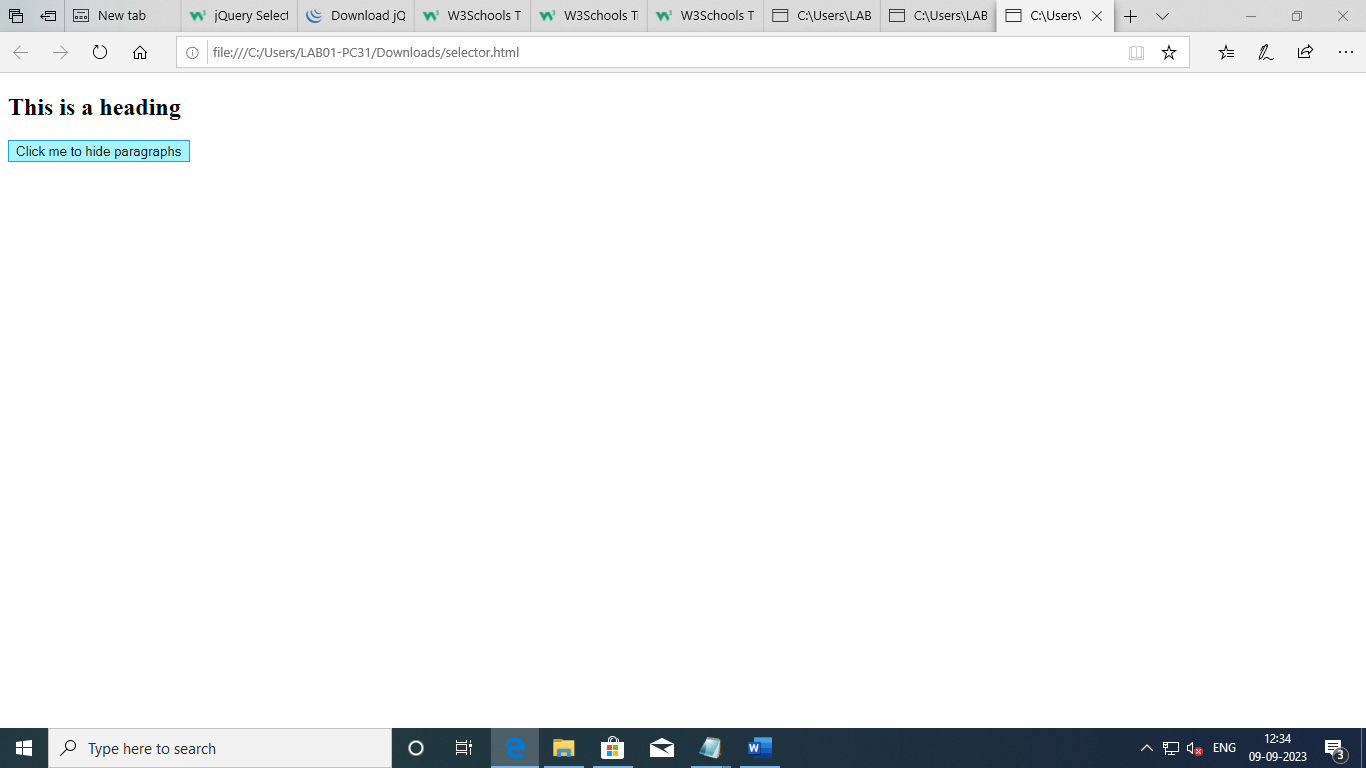
<button>Click me to hide paragraphs</button>

</body>

</html>

#### Output :

****

****

### ID Selector

#### Code :

<!DOCTYPE html>

<html>

<head>

<script src="https://code.jquery.com/jquery-3.7.1.js"></script>

<script>

$(document).ready(function(){

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

$("#test").hide();

});

});

</script>

</head>

<body>

<h2>This is a heading</h2>

<p>This is a paragraph.</p>

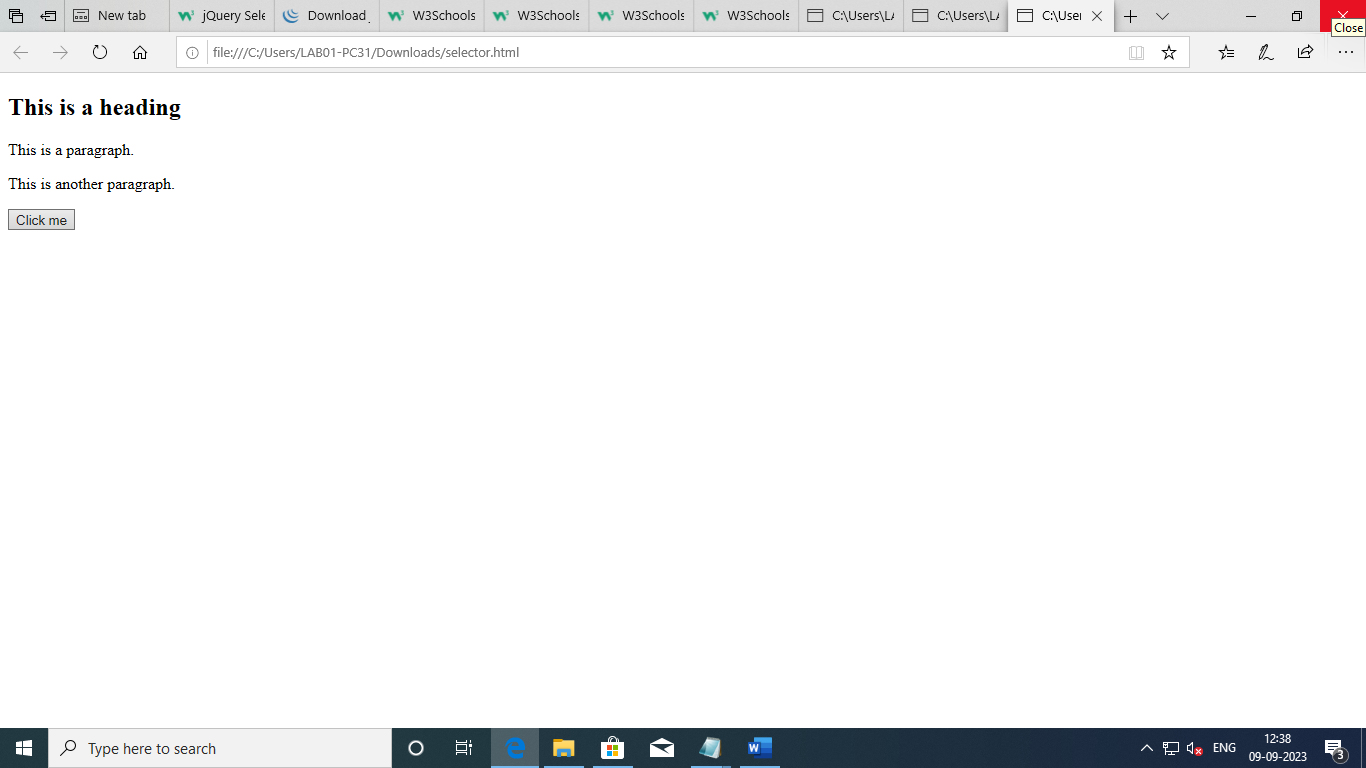
<p id="test">This is another paragraph.</p>

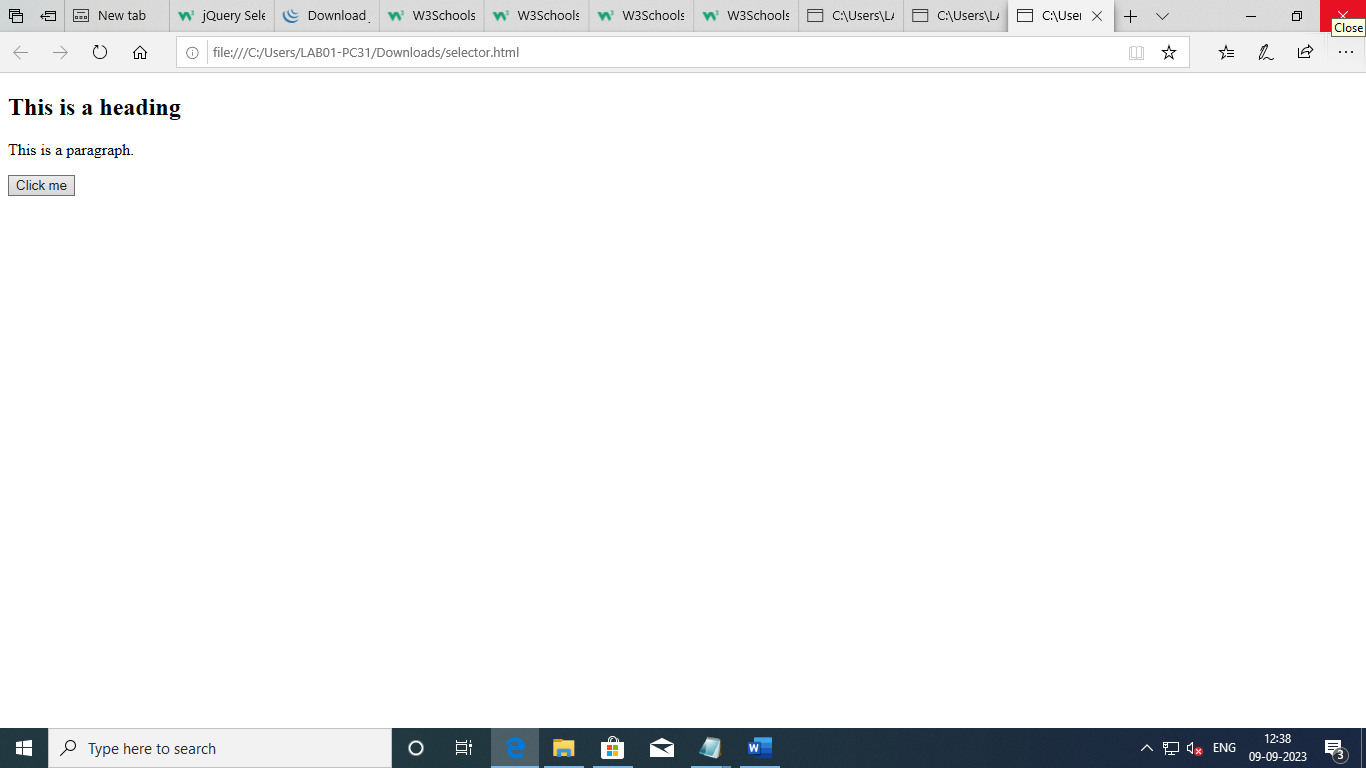
<button>Click me</button>

</body>

</html>

#### Output :

****

****

### jQuery Hide and show effect

#### Code :

<!DOCTYPE html>

<html>

<head>

<script src="https://code.jquery.com/jquery-3.7.1.js"></script>

<script>

$(document).ready(function(){

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

$("p").hide();

});

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

$("p").show();

});

});

</script>

</head>

<body>

<p>If you click on the "Hide" button, I will disappear.</p>

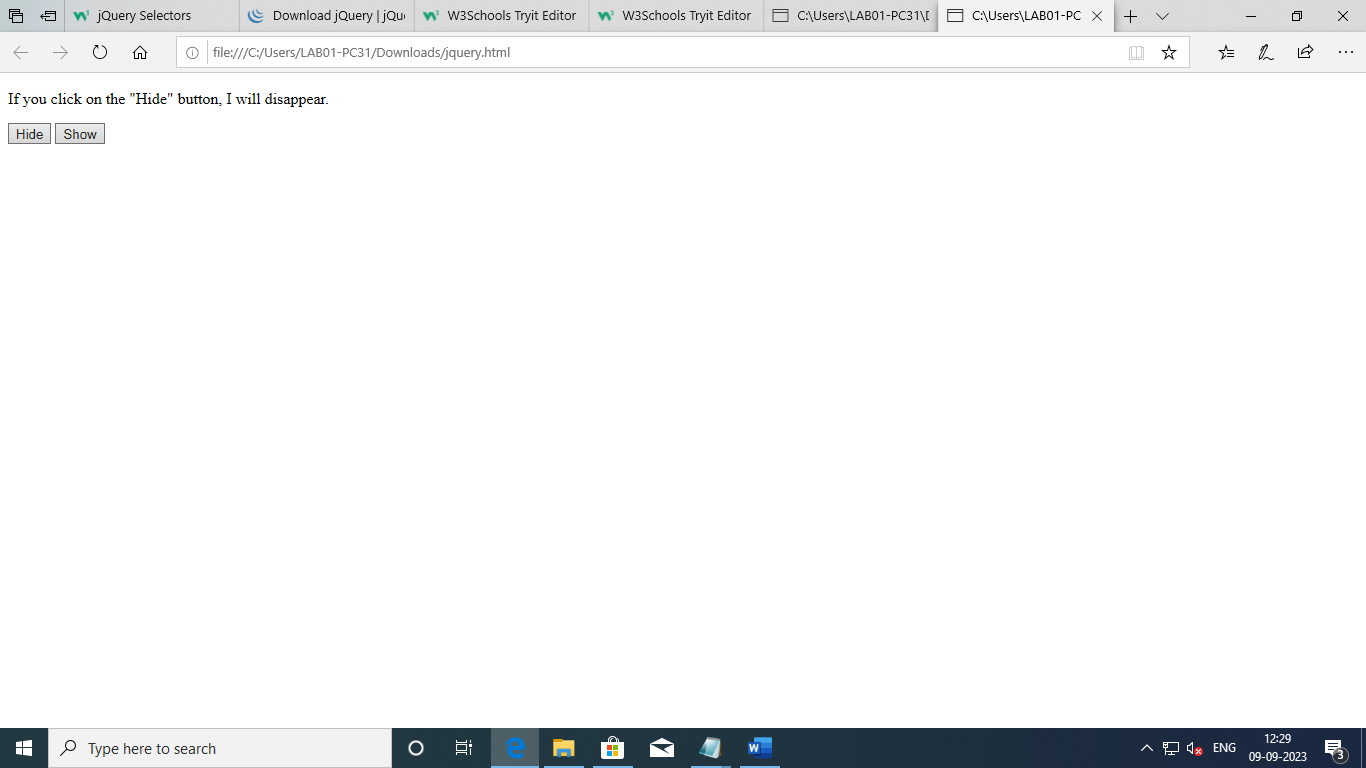
<button id="hide">Hide</button>

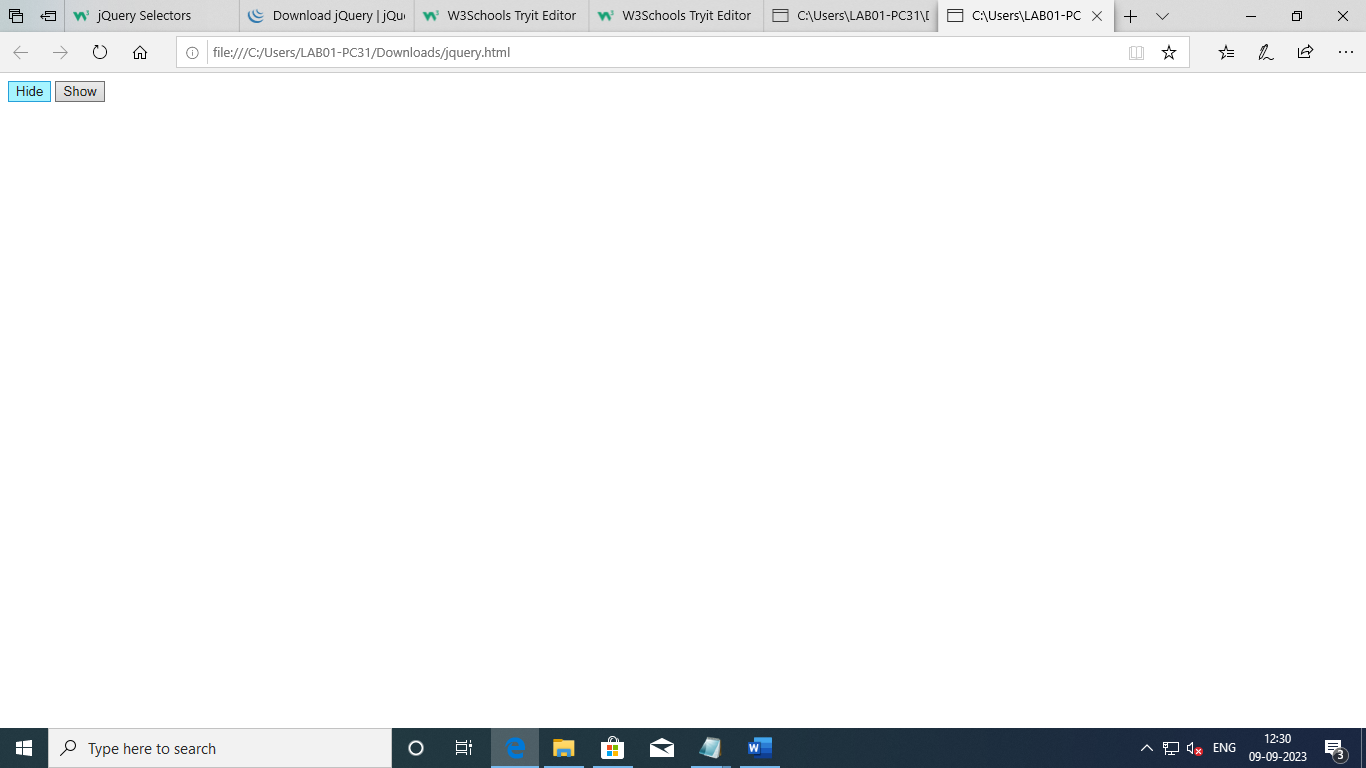
<button id="show">Show</button>

</body>

</html>

#### Output :





## jQuery fading effects, jQuery Sliding effects

### fadeIn()

#### Code :

<!DOCTYPE html>

<html>

<head>

<script src="https://code.jquery.com/jquery-3.7.1.js"></script>

<script>

$(document).ready(function(){

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

$("#div1").fadeIn();

$("#div2").fadeIn("slow");

$("#div3").fadeIn(3000);

});

});

</script>

</head>

<body>

<p>Demonstrate fadeIn() with different parameters.</p>

<button>Click to fade in boxes</button><br><br>

<div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div><br>

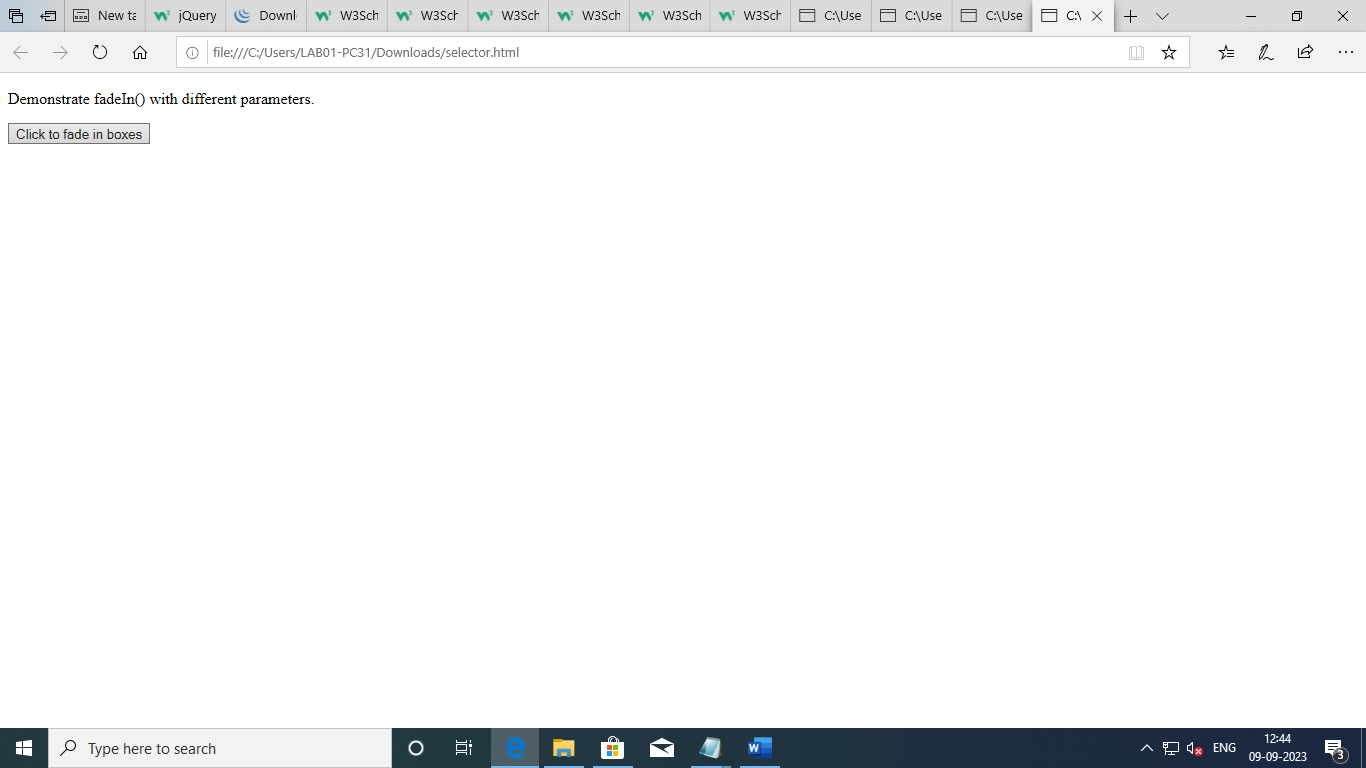
<div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div><br>

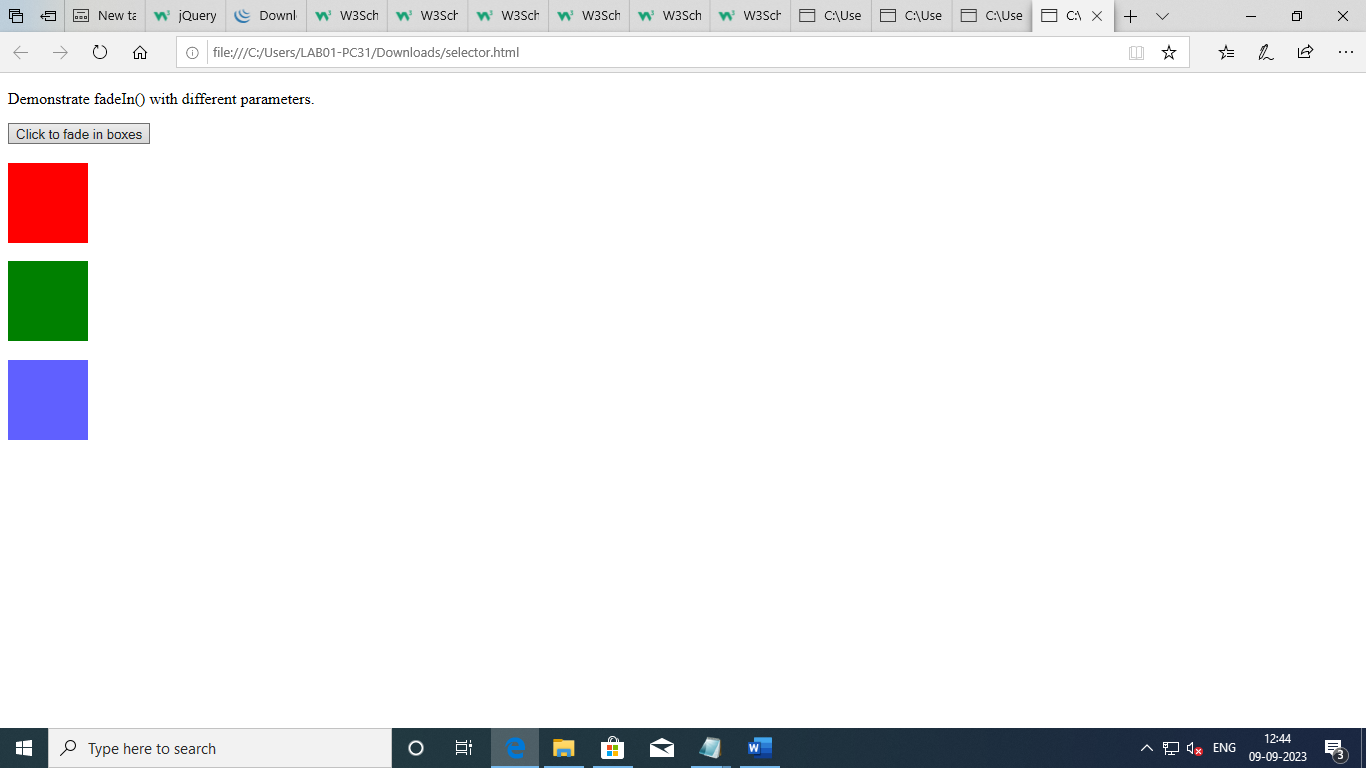
<div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>

</body>

</html>

#### Output :

****

****

### fadeout()

#### Code :

<!DOCTYPE html>

<html>

<head>

<script src="https://code.jquery.com/jquery-3.7.1.js"></script>

<script>

$(document).ready(function(){

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

$("#div1").fadeOut();

$("#div2").fadeOut("slow");

$("#div3").fadeOut(3000);

});

});

</script>

</head>

<body>

<p>Demonstrate fadeOut() with different parameters.</p>

<button>Click to fade out boxes</button><br><br>

<div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>

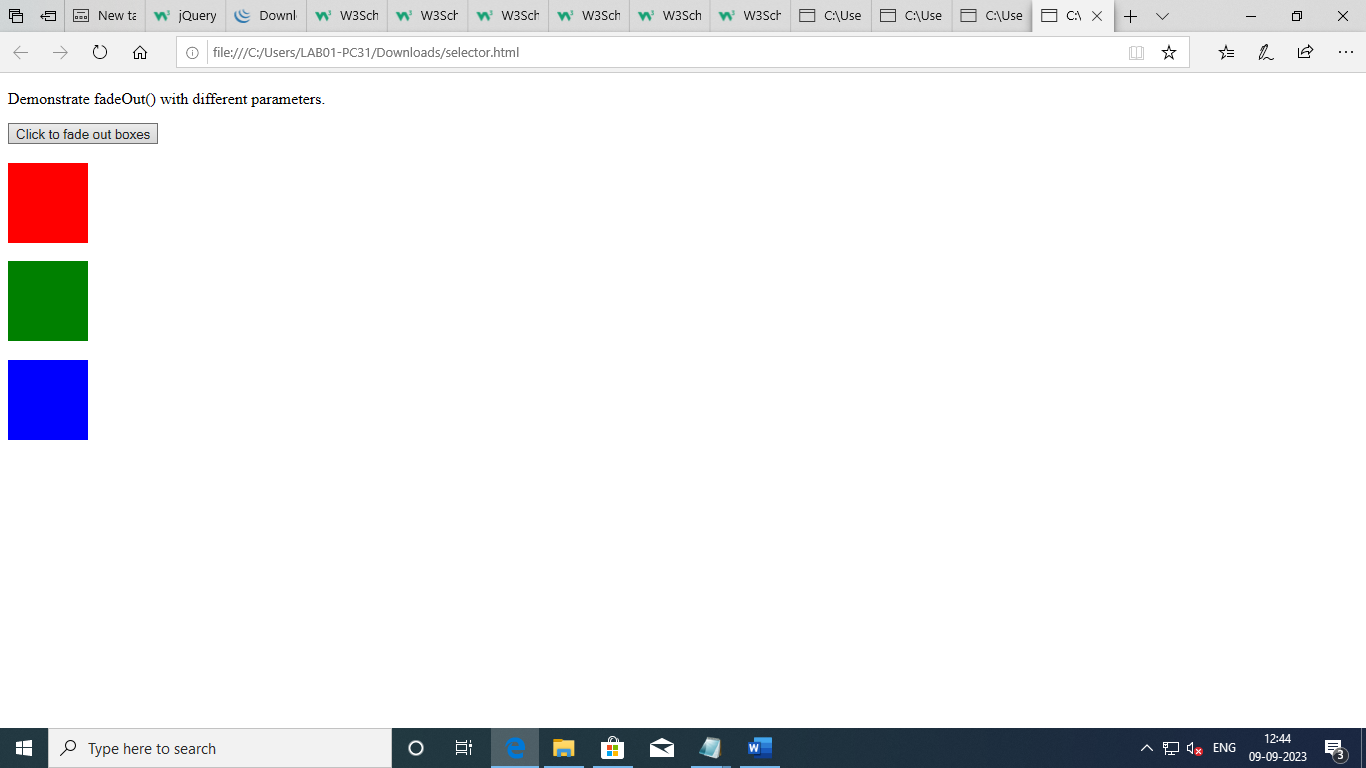
<div id="div2" style="width:80px;height:80px;background-color:green;"></div><br>

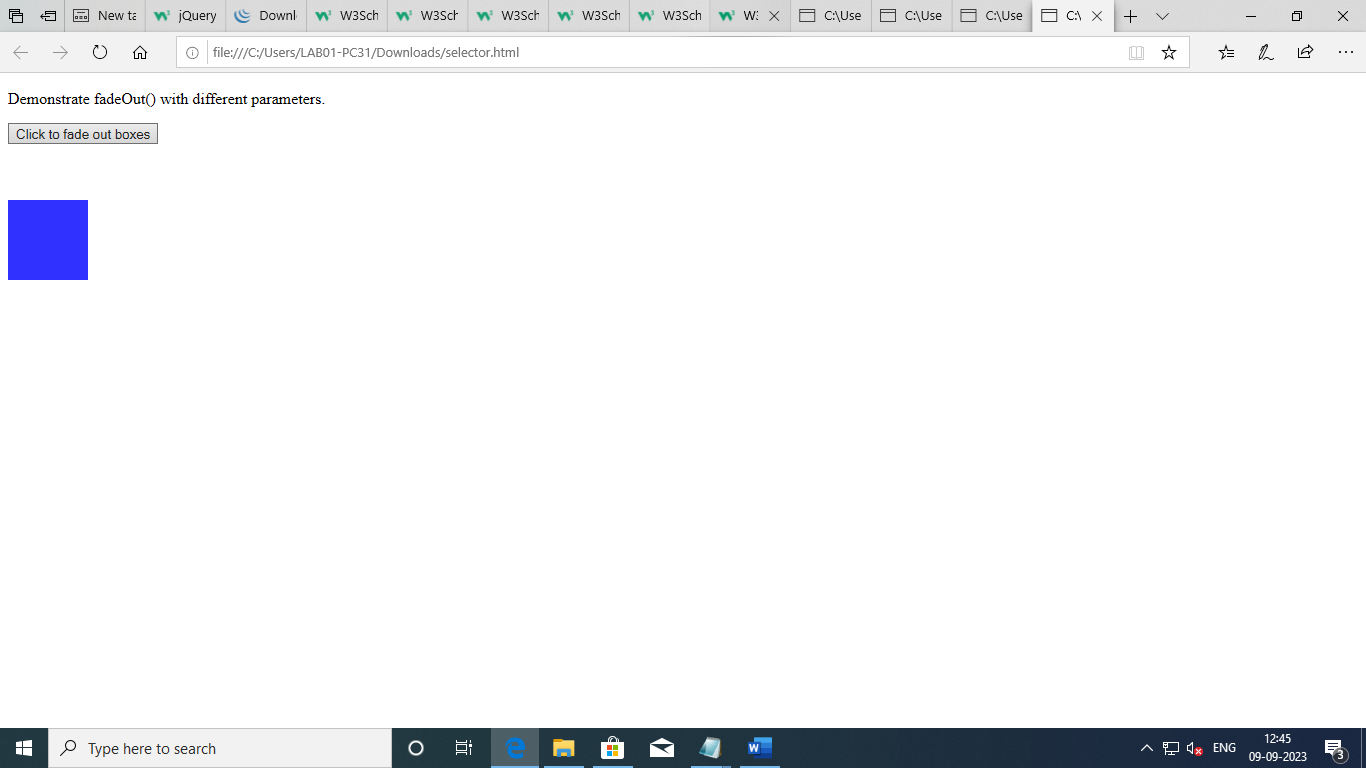
<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>

</body>

</html>

#### Output :





### Sliding effect – sliding up & down

#### Code :

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>

<script>

$(document).ready

(

function()

{ $("#flip").click

(

function() { $("#panel").slideDown("slow"); }

)

$("#up").click

(

function() { $("#panel").slideUp("slow"); }

); }

);

</script>

<style>

#panel, #flip, #up, #pan {

padding: 5px;

background-color: #e5eecc;

border: solid 1px #c3c3c3;

}

#panel, #pan {

padding: 15px;

display: none;

}

</style>

</head>

<body>

<div id="panel">Hello world!</div>

<button id="flip">Click to slide down panel</div>

<button id="up">Click to slide up panel</div>

</body>

</html>

#### Output :



