Emerging Technologies

Journal 2025-2026

Name: sahil kanojiya

Roll No : 15

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	jQuery Basic, jQuery Events		
b	jQuery Selectors, jQuery Hide and Show effects		
c	jQuery fading effects, jQuery Sliding effects		
8	**jQuery Advanced**		
a	jQuery Animation effects, jQuery Chaining		
b	jQuery Callback, jQuery Get		
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

A) Write a MongoDB query to create and drop database.

> show databases // checks currents databases.

```
C:\windows\system32\cmd.exe - mongo
> show databases
MKSDB   0.000GB
admin   0.000GB
config  0.000GB
local   0.000GB
```

> use mks // Using database mks

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

> db.createCollection("user") //Creating Empty Collection

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

> show databases //Database is Created

```
> show databases
MKSDB   0.000GB
admin   0.000GB
config  0.000GB
local   0.000GB
mks     0.000GB
```

> db.dropDatabase() //Drop Database

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

B) Write a MongoDB query to create, display and drop a collection

> use sy

```
C:\windows\system32\cmd.exe - mongo
> use sy
switched to db sy
```

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

```
> db.user.insert({"name":"ABC","rollno":10})
WriteResult({ "nInserted" : 1 })
```

> show collections

```
> show collections
user
```

> db.user.find()

```
> db.user.find()
{ "_id" : ObjectId("634ed24c6029d04034893a38"), "name" : "ABC", "rollno" : 10 }
```

> db.user.drop()

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

> show collections

```
> show collections
>
```

C) Write a MongoDB query to insert, query, update and delete a document

- <u>Different Methods of inserting Documents</u>

i. Insert Document

> use mks

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

Command Prompt - mongo

```
> use mks
switched to db mks
> db.products.insert( { item: "card", qty: 15 } )
WriteResult({ "nInserted" : 1 })
> db.products.find()
{ "_id" : ObjectId("634fe9f4b3bd865a20c2d731"), "item" : "card", "qty" : 15 }
>
```

ii. Insert Multiple Documents db.products.insert([
{ _id: 11, item: "pencil", qty: 50, type: "no.2" },
{ item: "pen", qty: 20 },
{ item: "eraser", qty: 25 }
])

Command Prompt - mongo

```
> db.products.insert(
...     [
...       { _id: 11, item: "pencil", qty: 50, type: "no.2" },
...       { item: "pen", qty: 20 },
...       { item: "eraser", qty: 25 }
...     ]
... )
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 3,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

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

```
Command Prompt - mongo
> db.products.insertOne( { _id: 10, item: "box", qty: 20 } )
{ "acknowledged" : true, "insertedId" : 10 }
>
```

- <u>Updating Document Queries :</u>

i. 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

```
Command Prompt - mongo
> db.inventory.find({"_id":10})
{ "_id" : 10, "item" : "sketch pad", "qty" : 95, "status" : "A" }
> db.inventory.update({ _id: 10 },{$set: {status: "Pending" }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.inventory.find({"_id":10})
{ "_id" : 10, "item" : "sketch pad", "qty" : 95, "status" : "Pending" }
>
```

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()

```
> db.products.find()
{ "_id" : 101, "item" : "pencil", "qty" : 2 }
{ "_id" : 102, "item" : "pen", "qty" : 4 }
{ "_id" : 103, "item" : "eraser", "qty" : 5 }
{ "_id" : 104, "item" : "refill", "qty" : 6 }
> db.products.update({"item" : "pen"},{"item" : "pen", "qty" : 400, "COD":"Yes"})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.products.find()
{ "_id" : 101, "item" : "pencil", "qty" : 2 }
{ "_id" : 102, "item" : "pen", "qty" : 400, "COD" : "Yes" }
{ "_id" : 103, "item" : "eraser", "qty" : 5 }
{ "_id" : 104, "item" : "refill", "qty" : 6 }
>
```

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

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

Command Prompt - mongo

```
> db.products.find()
{ "_id" : 101, "item" : "pencil", "qty" : 2 }
{ "_id" : 102, "item" : "pen", "qty" : 400, "COD" : "Yes" }
{ "_id" : 103, "item" : "eraser", "qty" : 5 }
{ "_id" : 104, "item" : "refill", "qty" : 6 }
> db.products.remove({"item":"pen"})
WriteResult({ "nRemoved" : 1 })
> db.products.find()
{ "_id" : 101, "item" : "pencil", "qty" : 2 }
{ "_id" : 103, "item" : "eraser", "qty" : 5 }
{ "_id" : 104, "item" : "refill", "qty" : 6 }
>
```

- Remove All Documents that Match a Condition

Command Prompt - mongo

```
> db.products.find()
{ "_id" : 101, "item" : "pencil", "qty" : 2 }
{ "_id" : 103, "item" : "eraser", "qty" : 5 }
{ "_id" : 104, "item" : "refill", "qty" : 6 }
> db.products.remove( { qty: { $gt: 2 } } )
WriteResult({ "nRemoved" : 2 })
> db.products.find()
{ "_id" : 101, "item" : "pencil", "qty" : 2 }
```

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")}});

Command Prompt - mongo

```
> db.products.find()
{ "_id" : 101, "item" : "pencil", "qty" : 2 }
{ "_id" : 10, "item" : "large box", "qty" : 20 }
{ "_id" : 11, "item" : "small box", "qty" : 55 }
{ "_id" : 12, "item" : "medium box", "qty" : 30 }
> db.products.find( { $where: function() {return (this.item=="pencil")}});
{ "_id" : 101, "item" : "pencil", "qty" : 2 }
>
```

Practical No:3 – Implementing Aggregation

A) 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}}}]

Command Prompt - mongo

```
> db.school.find()
{ "_id" : 101, "Name" : "Stud 1", "Roll" : 1, "Gender" : "Male", "Age" : 15 }
{ "_id" : 102, "Name" : "Stud 2", "Roll" : 2, "Gender" : "Male", "Age" : 20 }
{ "_id" : 103, "Name" : "Stud 3", "Roll" : 3, "Gender" : "Female", "Age" : 12 }
{ "_id" : 104, "Name" : "Stud 4", "Roll" : 4, "Gender" : "Female", "Age" : 21 }
{ "_id" : 105, "Name" : "Stud 5", "Roll" : 5, "Gender" : "Female", "Age" : 15 }
{ "_id" : 106, "Name" : "Stud 6", "Roll" : 6, "Gender" : "Male", "Age" : 16 }
{ "_id" : 107, "Name" : "Stud 7", "Roll" : 7, "Gender" : "Female", "Age" : 17 }
> db.school.aggregate([{$group:{_id:"$Gender", Total:{$sum:1}}}])
{ "_id" : "Female", "Total" : 4 }
{ "_id" : "Male", "Total" : 3 }
>
```

- Min & Max

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

Command Prompt - mongo

```
> db.school.find()
{ "_id" : 101, "Name" : "Stud 1", "Roll" : 1, "Gender" : "Male", "Age" : 15 }
{ "_id" : 102, "Name" : "Stud 2", "Roll" : 2, "Gender" : "Male", "Age" : 20 }
{ "_id" : 103, "Name" : "Stud 3", "Roll" : 3, "Gender" : "Female", "Age" : 12 }
{ "_id" : 104, "Name" : "Stud 4", "Roll" : 4, "Gender" : "Female", "Age" : 21 }
{ "_id" : 105, "Name" : "Stud 5", "Roll" : 5, "Gender" : "Female", "Age" : 15 }
{ "_id" : 106, "Name" : "Stud 6", "Roll" : 6, "Gender" : "Male", "Age" : 16 }
{ "_id" : 107, "Name" : "Stud 7", "Roll" : 7, "Gender" : "Female", "Age" : 17 }
> db.school.aggregate([{$group:{_id:"$Gender", MaxAge:{$max:"$Age"}}}])
{ "_id" : "Female", "MaxAge" : 21 }
{ "_id" : "Male", "MaxAge" : 20 }
>
```

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

```
> db.school.aggregate([{$group:{_id:"$Gender", MinAge:{$min:"$Age"}}}])
{ "_id" : "Female", "MinAge" : 12 }
{ "_id" : "Male", "MinAge" : 15 }
>
```

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

```
> db.school.aggregate([{$group:{_id:"$Gender", AvgAge:{$avg:"$Age"}}}])
{ "_id" : "Female", "AvgAge" : 16.25 }
{ "_id" : "Male", "AvgAge" : 17 }
>
```

B) 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()

Command Prompt - mongo

```
> db.score.find()
{ "_id" : 1, "Name" : "User1", "Scroes" : [ 10 ] }
> db.score.update({Name:"User1"},{$push:{"Scroes":80}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.score.find()
{ "_id" : 1, "Name" : "User1", "Scroes" : [ 10, 80 ] }
>
```

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

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

```
{ "_id" : 1, "Name" : "User1", "Scroes" : [ 10, 80 ] }
> db.score.update({Name:"User1"},{$addToSet:{"Macth":5}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.score.find()
{ "_id" : 1, "Name" : "User1", "Scroes" : [ 10, 80 ], "Macth" : [ 5 ] }
>
```

C) 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" }}});

Select Command Prompt - mongo

```
> db.school.find()
{ "_id" : 101, "Name" : "Stud 1", "Roll" : 1, "Gender" : "Male", "Age" : 15 }
{ "_id" : 102, "Name" : "Stud 2", "Roll" : 2, "Gender" : "Male", "Age" : 20 }
{ "_id" : 103, "Name" : "Stud 3", "Roll" : 3, "Gender" : "Female", "Age" : 12 }
{ "_id" : 104, "Name" : "Stud 4", "Roll" : 4, "Gender" : "Female", "Age" : 21 }
{ "_id" : 105, "Name" : "Stud 5", "Roll" : 5, "Gender" : "Female", "Age" : 15 }
{ "_id" : 106, "Name" : "Stud 6", "Roll" : 6, "Gender" : "Male", "Age" : 16 }
{ "_id" : 107, "Name" : "Stud 7", "Roll" : 7, "Gender" : "Female", "Age" : 17 }
> db.school.aggregate({$group:
... {_id: null,
... first: { $first: "$$ROOT" },
... last: { $last: "$$ROOT" }
... }})
{ "_id" : null, "first" : { "_id" : 101, "Name" : "Stud 1", "Roll" : 1, "Gender" : "Male", "Age" : 15 }, "last" :
{ "_id" : 107, "Name" : "Stud 7", "Roll" : 7, "Gender" : "Female", "Age" : 17 } }
>
```

Practical no 4 : Replication, Backup and Restore

4a: Replication

Objective:

To set up and verify replication in MongoDB using a replica set with three instances on a single system.

Step 1: Create Folders for Replica Set Members

Create three folders for the three MongoDB instances:

C:\replace\p1

C:\replace\s1 C:\replace\s2

Name	Date modified	Type
p1	10/8/2025 3:21 PM	File folder
s1	10/8/2025 3:22 PM	File folder
s2	10/8/2025 3:22 PM	File folder

Step 2: Start First MongoDB Instance

Open Command Prompt and run:

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

```
C:\Users\kfaiy>mongod --dbpath "C:\replace\p1" --port 27021 --replSet rs0
{"t":{"$date":"2025-10-08T15:05:56.136+05:30"},"s":"I",  "c":"-",        "id":8
{"seed":1039637339}}
{"t":{"$date":"2025-10-08T15:05:56.156+05:30"},"s":"I",  "c":"CONTROL",  "id":9
 and TLS 1.1, to force-enable TLS 1.1 specify --sslDisabledProtocols 'TLS1_0';
}
```

Step 3: Start Second MongoDB Instance

Open another Command Prompt and run:

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

```
C:\Users\kfaiy>mongod --dbpath "C:\replace\s1" --port 27022 --replSet rs0
{"t":{"$date":"2025-10-08T15:06:57.902+05:30"},"s":"I",  "c":"-",        "id":89
{"seed":3153972284}}
{"t":{"$date":"2025-10-08T15:06:57.924+05:30"},"s":"I",  "c":"CONTROL",  "id":97
 and TLS 1.1, to force-enable TLS 1.1 specify --sslDisabledProtocols 'TLS1_0'; t
}
```

Step 4: Start Third MongoDB Instance

Open one more Command Prompt and run:

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

```
C:\Users\kfaiy>mongod --dbpath "C:\replace\s2" --port 27023 --replSet rs0
{"t":{"$date":"2025-10-08T15:07:16.246+05:30"},"s":"I",  "c":"-",        "
{"seed":3506938112}}
{"t":{"$date":"2025-10-08T15:07:16.263+05:30"},"s":"I",  "c":"CONTROL",  "
 and TLS 1.1, to force-enable TLS 1.1 specify --sslDisabledProtocols 'TLS1
}
```

Step 5: Connect to MongoDB Shell

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

mongosh --port 27021

```
C:\Users\kfaiy>mongosh --port 27021
Current Mongosh Log ID: 68e6314328641876e0cebea3
Connecting to:          mongodb://127.0.0.1:27021/?directCon
Using MongoDB:          8.2.0
Using Mongosh:          2.5.8

For mongosh info see: https://www.mongodb.com/docs/mongodb-s

------
   The server generated these startup warnings when booting
   2025-10-08T15:05:56.448+05:30: Access control is not enab
cted
   2025-10-08T15:05:56.449+05:30: This server is bound to lo
r with --bind_ip <address> to specify which IP addresses it
this behavior is desired, start the server with --bind_ip 12
------

test> rs.initiate({
... _id: "rs0",
... members: [
... { _id: 0, host: "localhost:27021" },
... { _id: 1, host: "localhost:27022" },
... { _id: 2, host: "localhost:27023" }
... ]
... })
...
{
  ok: 1,
  '$clusterTime': {
```

Step 6: Initialize the Replica Set

mongosh --port 27021

```
rs.initiate({
_id:
"rs0",
member
s: [
{ _id: 0, host: "localhost:27021" },
{ _id: 1, host: "localhost:27022" },
{ _id: 2, host: "localhost:27023" }
]
})
```

Check status:

```
rs.status()
```

```
test> rs.initiate({
... _id: "rs0",
... members: [
... { _id: 0, host: "localhost:27021" },
... { _id: 1, host: "localhost:27022" },
... { _id: 2, host: "localhost:27023" }
... ]
... })
...
{
  ok: 1,
  '$clusterTime': {
```

Step 7: Insert Data into Primary

Switch to a test database and insert a record:

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

```
C:\Users\kfaiy>mongosh --port 27021
Current Mongosh Log ID: 68e632cff3fa32efbbcebea3
Connecting to:          mongodb://127.0.0.1:27021/?directConnection=true&serverSelectionTimec
Using MongoDB:          8.2.0
Using Mongosh:          2.5.8

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-10-08T15:05:56.448+05:30: Access control is not enabled for the database. Read and wr
cted
   2025-10-08T15:05:56.449+05:30: This server is bound to localhost. Remote systems will be u
r with --bind_ip <address> to specify which IP addresses it should serve responses from, or w
this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
------

rs0 [direct: primary] test> db.createCollection("students")
{ ok: 1 }
rs0 [direct: primary] test> db.students.insertOne({name: "faizan", age: 25, rollno: 35 })
{
  acknowledged: true,
  insertedId: ObjectId('68e6336ef3fa32efbbcebea4')
}
rs0 [direct: primary] test> use school
switched to db school
rs0 [direct: primary] school> db.students.insertOne({name: "faizan", age: 25, rollno: 35 })
{
  acknowledged: true,
  insertedId: ObjectId('68e633b4f3fa32efbbcebea5')
}
rs0 [direct: primary] school>
```

Step 8: Read from Secondary (Testing Replication)

Connect to the secondary node:

mongosh --port 27022

```
C:\Users\kfaiy>mongosh --port 27022
Current Mongosh Log ID: 68e633ee0e4d218c7fcebea3
Connecting to:          mongodb://127.0.0.1:27022/?directConnec
Using MongoDB:          8.2.0
Using Mongosh:          2.5.8

For mongosh info see: https://www.mongodb.com/docs/mongodb-shel

------
   The server generated these startup warnings when booting
   2025-10-08T15:06:58.158+05:30: Access control is not enablec
cted
   2025-10-08T15:06:58.158+05:30: This server is bound to local
r with --bind_ip <address> to specify which IP addresses it sho
this behavior is desired, start the server with --bind_ip 127.0
------

rs0 [direct: secondary] test> use school
switched to db school
rs0 [direct: secondary] school> db.students.find()
[
  {
    _id: ObjectId('68e633b4f3fa32efbbcebea5'),
    name: 'faizan',
    age: 25,
    rollno: 35
  }
]
rs0 [direct: secondary] school>
```

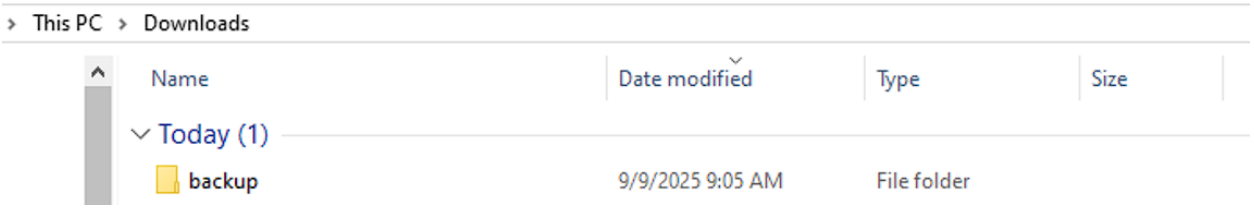
4b: Backup

Aim: To perform **Backup operations on a MongoDB database.**

Step 1: open Command Prompt (CMD) and check database and select any data which you want to restore

```
test> show dbs
MakeDB     32.00 KiB
MakeDB1    32.00 KiB
admin      32.00 KiB
config    108.00 KiB
demo       72.00 KiB
local     128.00 KiB
student    80.00 KiB
test       32.00 KiB
test>
```

Step 2: go to file explorer and create file



Step 3: Open Command Prompt (CMD)

cd "C:\Program Files\MongoDB\Server\6.0\bin"

step 4: Run the backup command

mongodump --db <database_name> --out <backup_folder_path>

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19045.5965]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\MongoDB\Server\4.0\bin>mongodump --db=students --out="C:\Users\admin\Downloads\backup"

C:\Program Files\MongoDB\Server\4.0\bin>mongodump --db=students --out="C:\Users\admin\Downloads\backup"

C:\Program Files\MongoDB\Server\4.0\bin>mongodump --db=student --out="C:\Users\admin\Downloads\backup"
2025-09-09T09:05:37.014+0530    writing student.student to
2025-09-09T09:05:37.016+0530    writing student.Student to
2025-09-09T09:05:37.057+0530    done dumping student.Student (0 documents)
2025-09-09T09:05:37.064+0530    done dumping student.student (2 documents)
```

4C: Restore

Aim: To perform **Restore operations on a MongoDB database.**

Step 1: Open Command Prompt (CMD) and show data base and delete any database using **db.dropDatabase()**

```
student> db.dropDatabase()
{ ok: 1, dropped: 'student' }
student> show dbs
MakeDB     32.00 KiB
MakeDB1    32.00 KiB
admin      32.00 KiB
config    108.00 KiB
demo       72.00 KiB
local     128.00 KiB
test       32.00 KiB
```

Step 2: Open Command Prompt (CMD)

cd "C:\Program Files\MongoDB\Server\6.0\bin"

Step 3: Run the restore command:
mongorestore --db <target_database> <path_to_backup_folder>

```
C:\Program Files\MongoDB\Server\4.0\bin>mongorestore --db=student "C:\Users\admin\Downloads\backup\student"
2025-09-09T09:23:27.459+0530    the --db and --collection args should only be used when restoring from a BSON file. Other uses are deprecated
2025-09-09T09:23:27.461+0530    building a list of collections to restore from C:\Users\admin\Downloads\backup\student dir
2025-09-09T09:23:27.464+0530    reading metadata for student.Student from C:\Users\admin\Downloads\backup\student\Student.metadata.json
2025-09-09T09:23:27.485+0530    no indexes to restore
2025-09-09T09:23:27.485+0530    finished restoring student.Student (0 documents)
2025-09-09T09:23:29.479+0530    restoring student.student from C:\Users\admin\Downloads\backup\student\student.bson
2025-09-09T09:23:29.480+0530    no indexes to restore
2025-09-09T09:23:29.480+0530    finished restoring student.student (2 documents)
2025-09-09T09:23:29.481+0530    done

C:\Program Files\MongoDB\Server\4.0\bin>
```

Step 4: Open Command Prompt (CMD) and show data base to check database is restore or not.

```
student> show dbs
MakeDB     32.00 KiB
MakeDB1    32.00 KiB
admin      32.00 KiB
config    108.00 KiB
demo       72.00 KiB
local     128.00 KiB
student    16.00 KiB
test       32.00 KiB
student>
```

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:

```
Collection created
Collection selected
Document inserted
Document{{_id=634af93538d5623c5a85305b, title=Mongodb, id=1, Discription=database, Created by=MKS}}
BUILD SUCCESSFUL (total time: 4 seconds)
```

* 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:

```
Updated Successfully
BUILD SUCCESSFUL (total time: 5 seconds)
```

```
> db.mycol.find().pretty()
{
        "_id" : ObjectId("634af93538d5623c5a85305b"),
        "title" : "Mongodb",
        "id" : 2,
        "Discription" : "database",
        "Created by" : "MKS"
}
>
```

* Delete

```
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.Fi


lters; 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:

```
Credentials::MongoCredential{mechanism=null, userName='MKS', source='MakeDB'
Connected to database successfully
Collection selected
Document deleted
BUILD SUCCESSFUL (total time: 4 seconds)
```

* 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++;
}}}
```

Output:

```
INFO: Opened connection [connectionId{localValue:1, serverValue:47}] to localhost:27017
Oct 16, 2022 1:26:54 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{a
Oct 16, 2022 1:26:54 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:48}] to localhost:27017
Document{{_id=634bb9302e53e5c705167ca7, id=2.0, Name=EFG, Phone=0987654332}}
BUILD SUCCESSFUL (total time: 4 seconds)
```

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:

```
==================== RESTART: C:\Users\Admin\Desktop\test.py ===================
63505fa04ba811e79be88f1c
{'_id': ObjectId('63505fa04ba811e79be88f1c'), 'name': 'ABC', 'address': 'Mumbai'}
>>>
```

- Retrieve

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

Output:

```
==================== RESTART: C:\Users\Admin\Desktop\test.py ===============
{'_id': 101, 'Name': 'Stud 1', 'Roll': 1, 'Gender': 'Male', 'Age': 15}
{'_id': 102, 'Name': 'Stud 2', 'Roll': 2, 'Gender': 'Male', 'Age': 20}
{'_id': 103, 'Name': 'Stud 3', 'Roll': 3, 'Gender': 'Female', 'Age': 12}
{'_id': 104, 'Name': 'Stud 4', 'Roll': 4, 'Gender': 'Female', 'Age': 21}
```

- 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():
    print(x)
```

Output:

```
==================== RESTART: C:\Users\Admin\Desktop\test.py ============
Deleted Stud 3
{'_id': 101, 'Name': 'Stud 1', 'Roll': 1, 'Gender': 'Male', 'Age': 15}
{'_id': 102, 'Name': 'Stud 2', 'Roll': 2, 'Gender': 'Male', 'Age': 20}
{'_id': 104, 'Name': 'Stud 4', 'Roll': 4, 'Gender': 'Female', 'Age': 21}
```

- 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)
```

Output:

```
==================== RESTART: C:\Users\Admin\Desktop\test.py =============
Updated Stud 1 Age
{'_id': 101, 'Name': 'Stud 1', 'Roll': 1, 'Gender': 'Male', 'Age': '18'}
{'_id': 102, 'Name': 'Stud 2', 'Roll': 2, 'Gender': 'Male', 'Age': 20}
{'_id': 104, 'Name': 'Stud 4', 'Roll': 4, 'Gender': 'Female', 'Age': 21}
```

Practical No:7– Programs on Basic jQuery

A.i) 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>
```

Output:

A.ii) jQuery Events

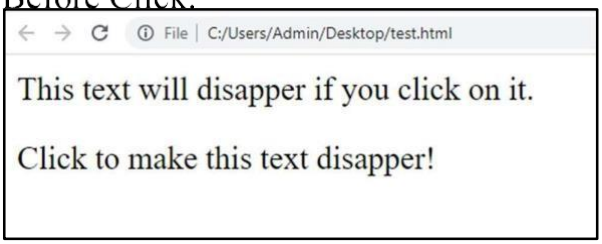
Index.html >

```
<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:

Before Click:



After Click:

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>
```

Output:

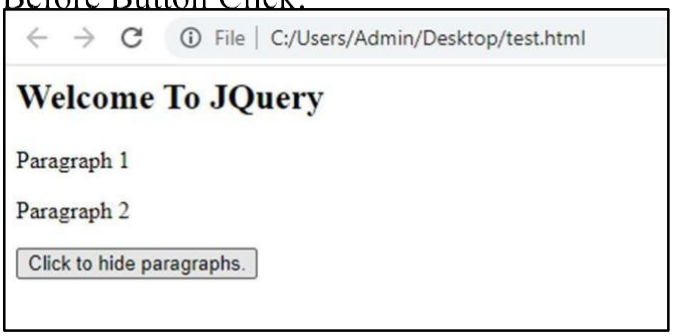
* jQuery Hide Paragraph :
Note: We Will Hide Paragraph Tag on Button Click Event.

Index.html >

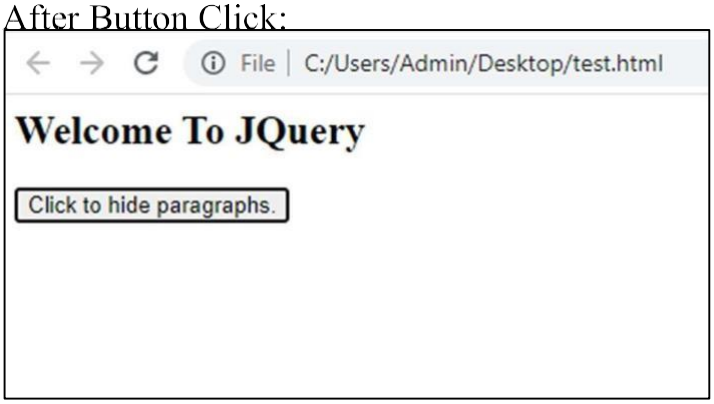
```
<html>  <head> <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"
>
</script>
<script>
$(document).ready(function(){
$("button").click(function(){
$("p").hide();
});
});
</script>
</head>
<body>
<h2>Welcome To JQuery</h2>
<p>Paragraph 1</p>
<p>Paragraph 2</p>
<button>Click to hide paragraphs.</button>
</body>
</html>
```

Output:

Before Button Click:



After Button Click:



C) jQuery fading effects, jQuery Sliding effects

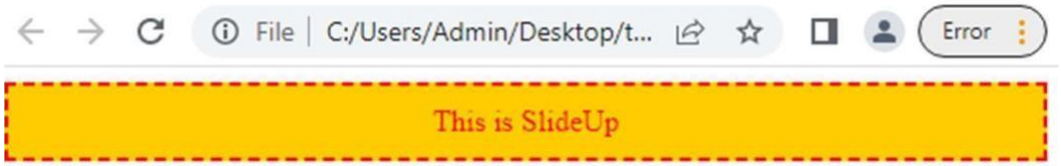
i) SlideUp

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:

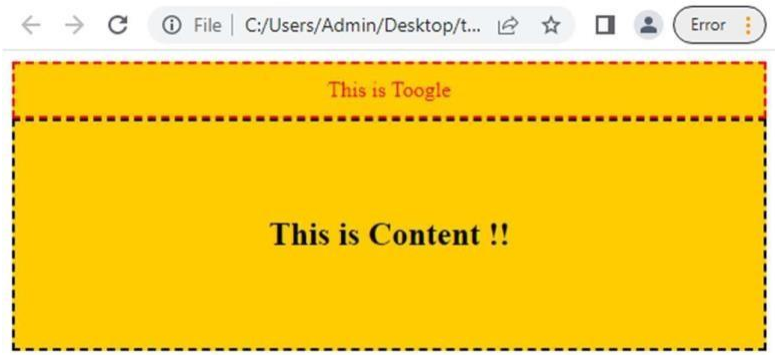
iii) SlideToggle

Index.html >

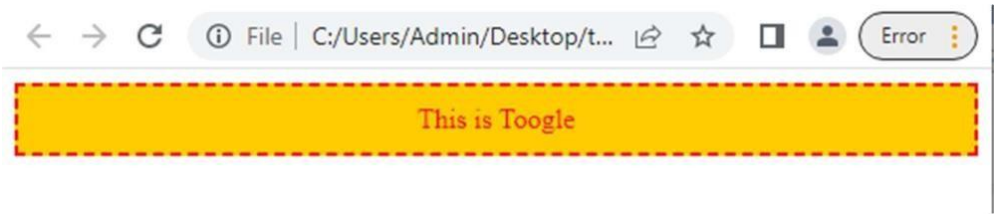
```
<html>
<head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">
</script>
<script>
$(document).ready(function(){
 $("#flip").click(function(){
 $("#panel").slideToggle("slow"); // Toggle Contains Both SideUp & Down
 });});
</script>
<style> #panel,#flip{
padding :10px; 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:


File | C:/Users/Admin/Desktop/t...
Error
This is Toogle
This is Content !!

After Click:


File | C:/Users/Admin/Desktop/t...
Error
This is Toogle

Practical No:8 – jQuery Advanced

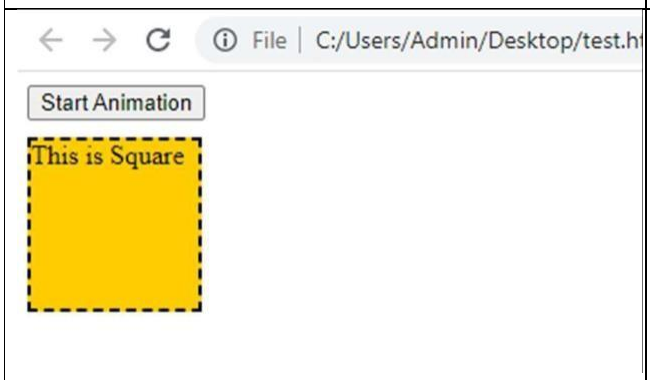
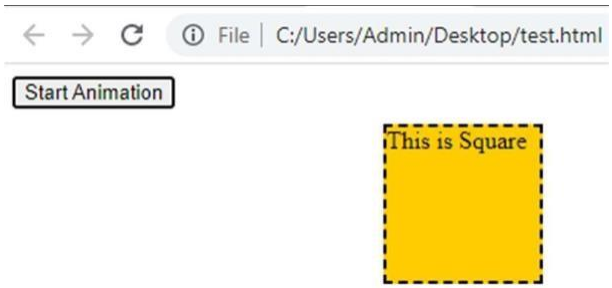
A) jQuery Animation effects, jQuery Chaining

* 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
	

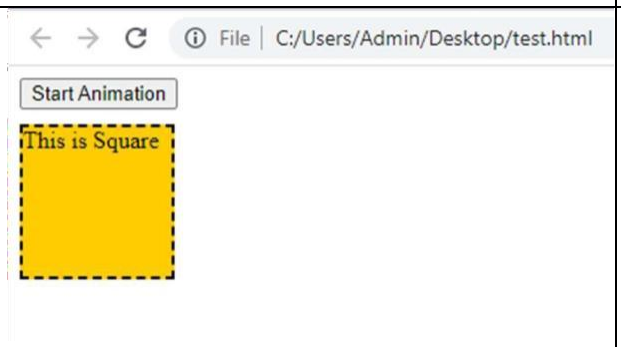
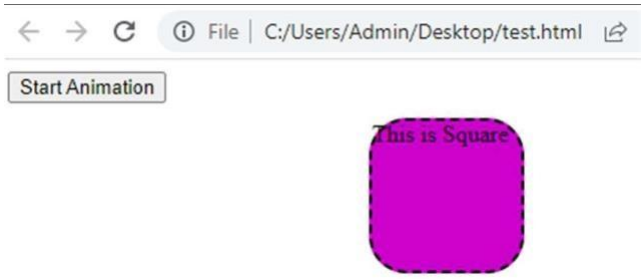
* Chaining effects

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

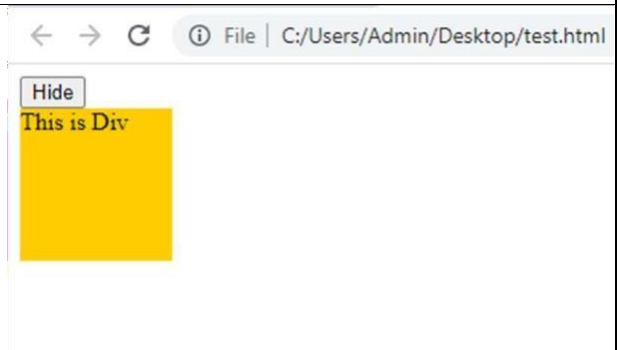
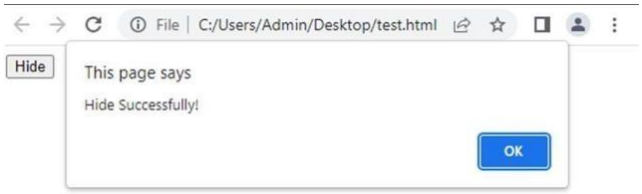
* Call Back:

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:

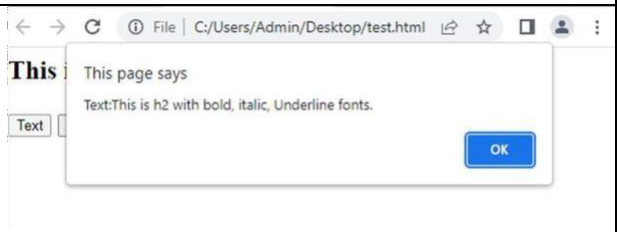
Demonstrate to get content with the jQuery text() and html() methods:

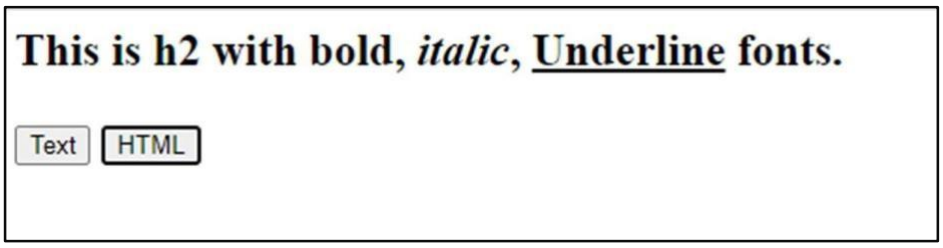
Code:

```
<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

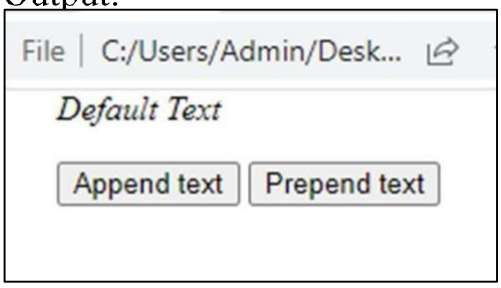
* Inserting Content Using Append & Prepend

Code:

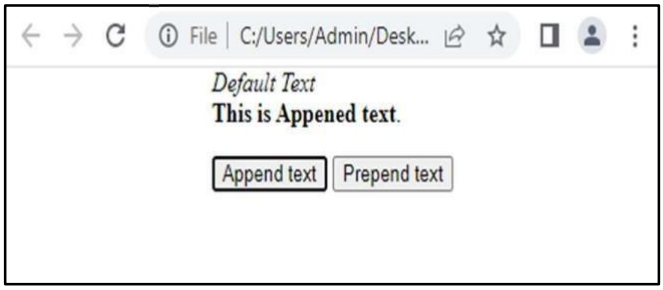
```
<html>  <head> <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"
>
</script>
<script>
$(document).ready(function(){
   $("#b1").click(function(){
```

```
    $("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>
```

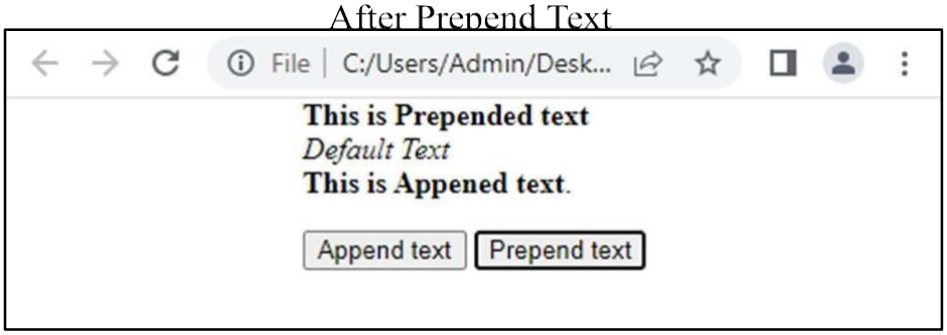
Output:



After Append Text



After Prepend Text



* Remove Element

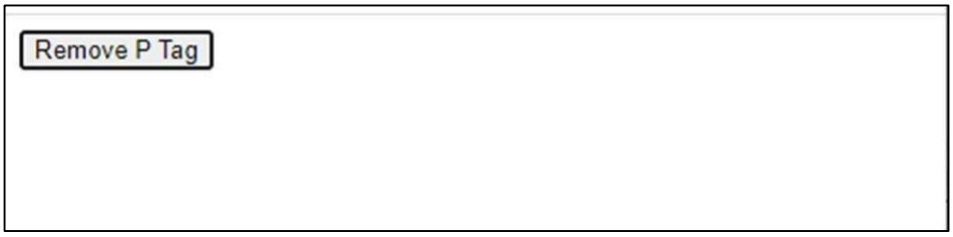
Code:

```
<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>
```

Before click:

After Click:



* Empty() Element

Code:

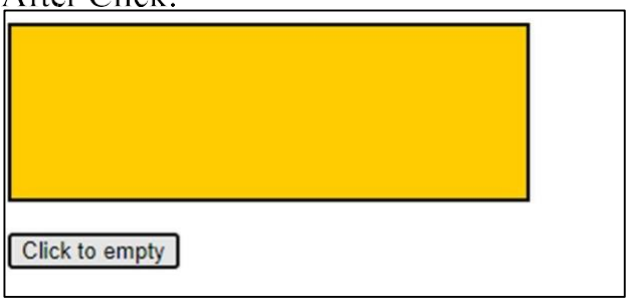
```
<html>
<head>    <style>
div{
     height:200px; width:300px;        border:2px solid
black; background-color:#ffcc00;
   }
   </style> <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">
</script>
<script>
$(document).ready(function(){
   $("button").click(function(){
   $("#d1").empty();
      });
 });
```

```
        </script>
    </head>
    <body>
    <div id="d1">
    <h3 align="center">Some Content </h3>
    </div>
    <br>
    <button>Click to empty</button>
    </body>
    </html>
```

Output:

Before click:

After Click:



Practical No:9 – JSON

A) Creating JSON

Code:

```
import json x={
"name":"xyz",
    "age":18,
"city":"MH"      }
y=json.dumps(x) print(y)
```

Output:

```
==================== RESTART: C:\Users\Admin\Desktop\test.py ====================
{"name": "xyz", "age": 18, "city": "MH"}
>>>
```

B) Parsing JSON

Code:

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

Output:

```
==================== RESTART: C:\Users\Admin\Desktop\test.py ====================
Name: xyz Age:  18
>>>
```

C) 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:

```
C:\Program Files\MongoDB\Server\4.0\bin>mongoimport --db jsontest --collection jsondata
--file C:\Users\Admin\Desktop\file.json
2022-10-20T04:19:50.251+0530    connected to: localhost
2022-10-20T04:19:50.619+0530    imported 1 document
```

```
> use jsontest
switched to db jsontest
> show collections
jsondata
> db.jsondata.find()
{ "_id" : ObjectId("63507f0e7d4bb02f72493985"), "Name" : "Test", "Class" : "TY", "Sem" : 5 }
>
```

Practical No:10 – Create a JSON file and import it to MongoDB

Steps:

1. Open Cmd
2. Type cd C:\Program Files\MongoDB\Server\4.0\bin
3. Run command to export data into json file Command:

mongoexport --db <Db_Name> --collection <collection_name>
--out “Any_File_Name_With_Path” --jsonArray –pretty

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

Output:

```
C:\Program Files\MongoDB\Server\4.0\bin>mongoexport --db test --collection school
--out C:\Users\Admin\Desktop\exported_Data.json --jsonArray --pretty
2022-10-20T04:28:34.832+0530    connected to: localhost
2022-10-20T04:28:34.895+0530    exported 6 records
```

Json File:

```
test.html   file.json   exported_Data.json
[{
    "_id": 101,
    "Name": "Stud 1",
    "Roll": 1,
    "Gender": "Male",
    "Age": "18"
},
{
    "_id": 102,
    "Name": "Stud 2",
    "Roll": 2,
    "Gender": "Male",
    "Age": 20
},
{
    "_id": 104,
    "Name": "Stud 4",
    "Roll": 4,
    "Gender": "Female",
    "Age": 21
},
{
    "_id": 105,
    "Name": "Stud 5",
    "Roll": 5,
    "Gender": "Female",
    "Age": 15
},
{
    "_id": 106,
    "Name": "Stud 6",
    "Roll": 6,
    "Gender": "Male",
    "Age": 16
}
```