RAMNIRANJAN JHUNJHUNWALA COLLEGE GHATKOPAR (W), MUMBAI - 400 086

DEPARTMENT OF INFORMATION TECHNOLOGY

2023 - 2024

T.Y. B. Sc. (I.T.) SEM V

Paper RJSUIT502 – NEXT GENERATION TECHNOLOGIES

Name : Rajbhar Sudesh Dinesh

Roll No.: 6417

Hindi Vidya Prachar Samiti's

RAMNIRANJAN JHUNJHUNWALA COLLEGE

Ghatkopar (W), Mumbai-400 086

Certificate

NAAC Accredited "A" Grade
तिद्या अमृतं अश्नुते
ESTD 1963
RJC
ISO 9001:2000
Best College
UNIVERSITY of MUMBAI

This is to certify that Mr./Ms. Sudesh Dinesh Rajbhar Roll No 6417 of T.Y.B.Sc.(I.T.) class has completed the required number of experiments in the subject of Next Generation Technology in the Department of Information Technology during the academic year 20 23- 2024 .

Professor In-Charge **Co-ordinator of IT Department**

Prof. Bharati Bhole

Prof. Archana Bhide

College Seal & Date **Examiner**

Index

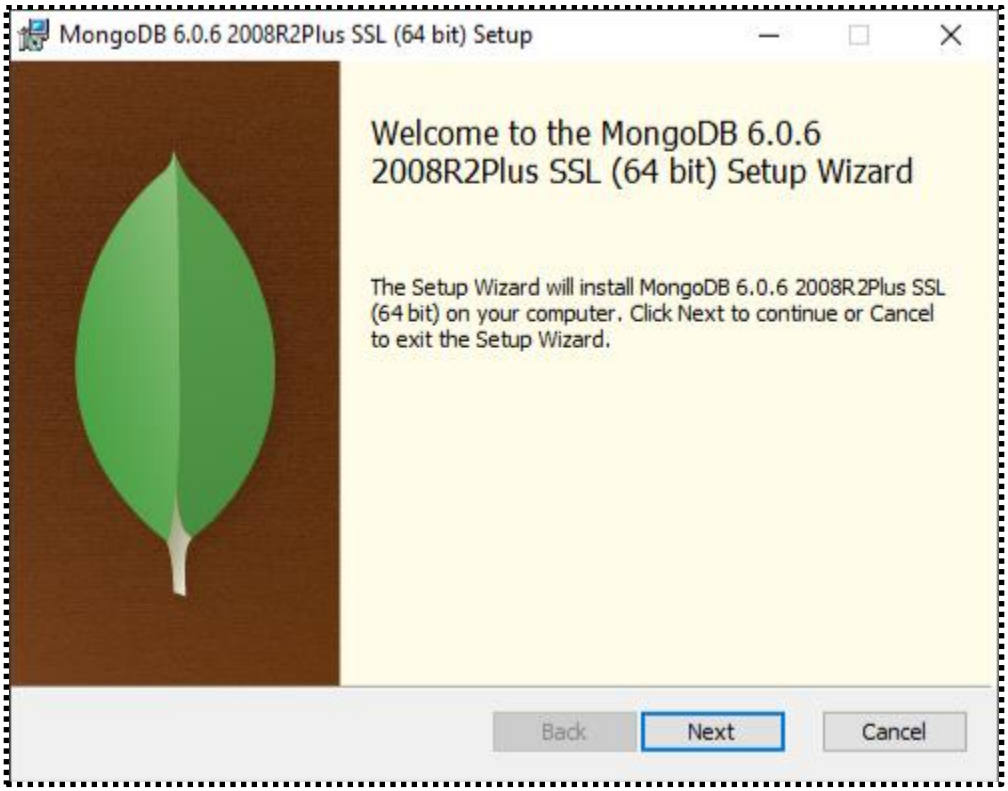
	h) Importing Data from .json file		
4	**Querying Mongo DB Database**	14/07/2023	
	a) find() and findOne() Method		
	b) pretty() Method		
	c) Filtering criteria in Mongo DB Queries/ Selecting Queries		
	d) Using operators in Queries		
	e) Projection Queries		
	f) limit() Method		
	g) skip() Method		
	h) Regular Expression in Mongo DB		
5	**Updating and Deleting Operations in Mongo DB**	21/07/2023	
	a) updateOne and updateMany() Methods		
	b) $set and $unset Operators		
	c) save() Method		
	d) deleteOne() and DeleteMany() Methods		
	e) remove() Methods		
6	**Aggregation Operations in Mongo DB**	21/07/2023	
	a) aggregate() Method		
7	**Sorting and Indexing Mongo DB Database**	04/08/2023	
	a) sort() Methods		
	b) Metadata Sort Technique		
	c) ensureIndex() and createIndex() Method		
8	**Replication, Backup, restore and sharding in Mongo DB**		

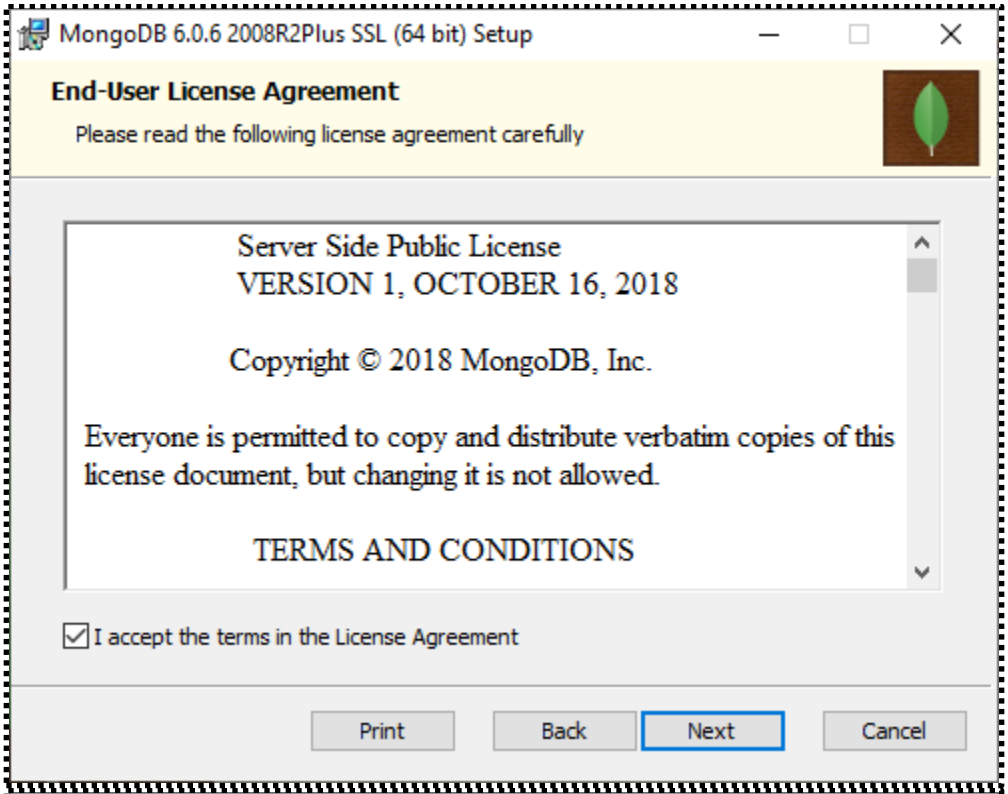
	a) Write a Mongo DB query to create a Backup of the Existing databases.	11/08/2023	
	b) Write a Mongo DB query to create restore the database from the backup.		
9	**Using Mongo DB with different programming languages**		
	a) Using Mongo DB with python	11/08/2023	
	b) Using Mongo DB with java	25/08/2023	
10	**Mongo DB Atlas**	01/09.2023	
11	**JQuery**	08/09/2023	

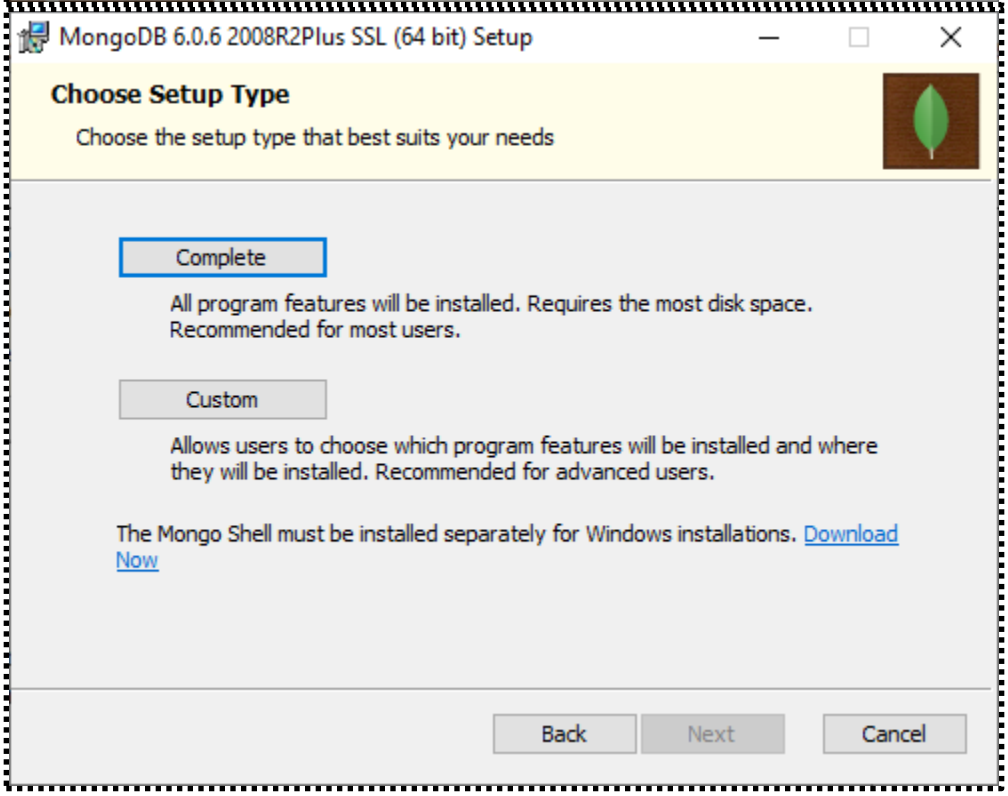
Practical No 1. Installation of Mongo DB and Mongo DB Shell Commands.

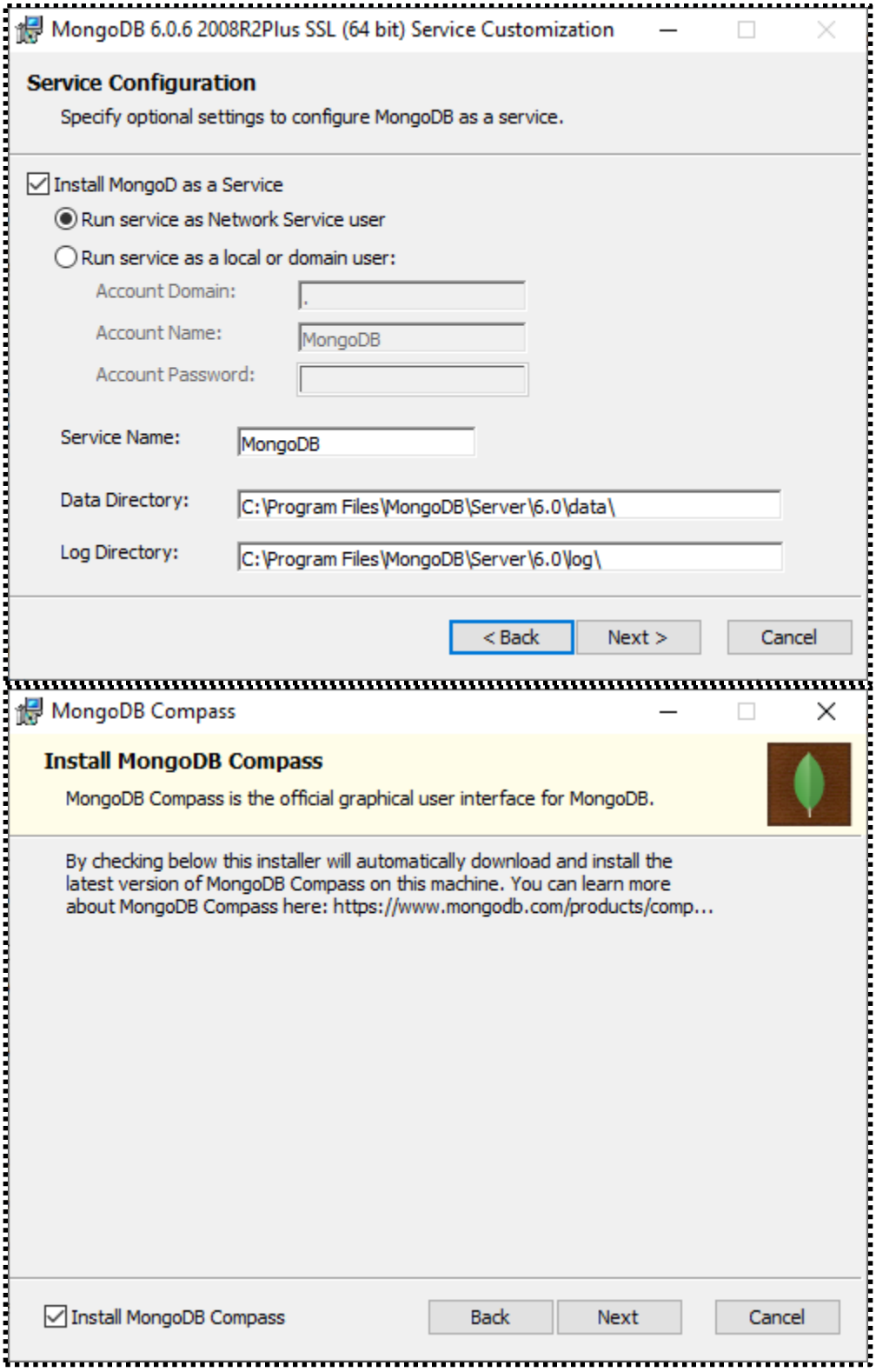
- Installation of Mongo DB .
- Mongo db Shell commands
- Using Mongo DB

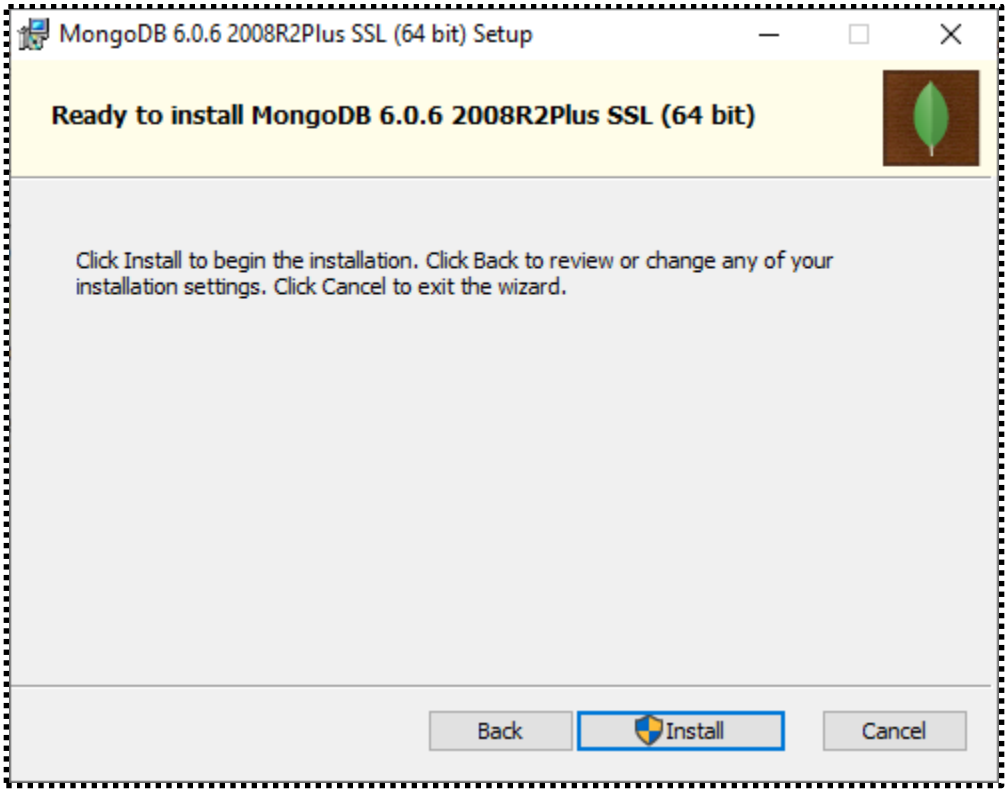
a) Installation of Mongo DB




MongoDB 6.0.6 2008R2Plus SSL (64 bit) Setup
End-User License Agreement
Please read the following license agreement carefully
Server Side Public License
VERSION 1, OCTOBER 16, 2018
Copyright © 2018 MongoDB, Inc.
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.
TERMS AND CONDITIONS
I accept the terms in the License Agreement
Print
Back
Next
Cancel

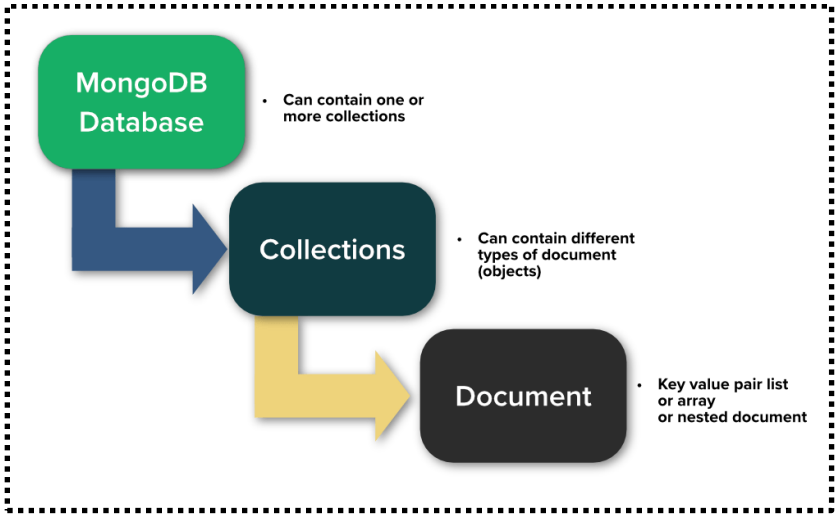

MongoDB 6.0.6 2008R2Plus SSL (64 bit) Setup
Choose Setup Type
Choose the setup type that best suits your needs
Complete
All program features will be installed. Requires the most disk space.
Recommended for most users.
Custom
Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.
The Mongo Shell must be installed separately for Windows installations. Download Now
Back
Next
Cancel

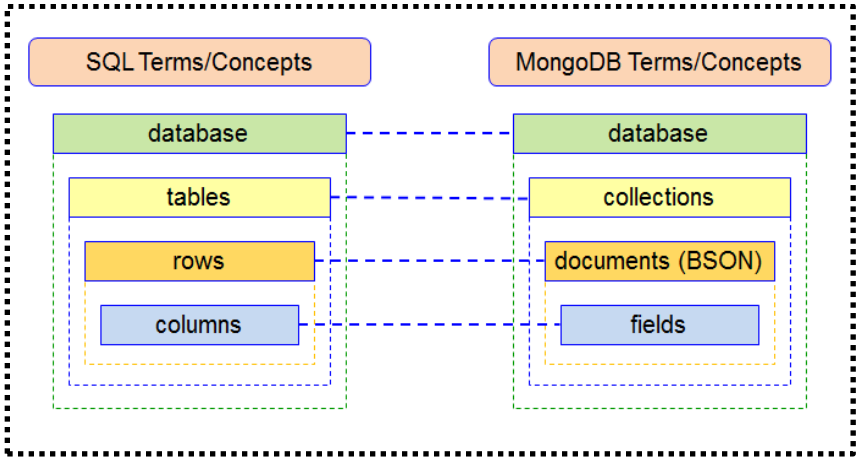

MongoDB 6.0.6 2008R2Plus SSL (64 bit) Service Customization
Service Configuration
Specify optional settings to configure MongoDB as a service.
Install MongoD as a Service
Run service as Network Service user
Run service as a local or domain user:
Account Domain:
.
Account Name:
MongoDB
Account Password:
Service Name:
MongoDB
Data Directory:
C:\Program Files\MongoDB\Server\6.0\data\
Log Directory:
C:\Program Files\MongoDB\Server\6.0\log\
< Back
Next >
Cancel
MongoDB Compass
Install MongoDB Compass
MongoDB Compass is the official graphical user interface for MongoDB.
By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: https://www.mongodb.com/products/comp...
Install MongoDB Compass
Back
Next
Cancel


MongoDB 6.0.6 2008R2Plus SSL (64 bit) Setup
Ready to install MongoDB 6.0.6 2008R2Plus SSL (64 bit)
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

b) Mongo DB Shell Commands

a) Create database
b) List database
c) Using Database
d) Create collection
e) List Collections
f) Insert Document into the Collection
g) Drop Collection
h) Delete Database
i) Help Command
j) Using Mongo DB Compass






```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Please enter a MongoDB connection string (Default: mongodb://localhost/):

Current Mongosh Log ID: 649508cffb5fb12bd571226e
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1
.10.1
Using MongoDB:          5.0.6
Using Mongosh:          1.10.1

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

------
   The server generated these startup warnings when booting
   2023-06-23T07:32:04.660+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------

test> show dbs
admin      48.00 KiB
config    108.00 KiB
employee   80.00 KiB
example    72.00 KiB
local     104.00 KiB
test       72.00 KiB
test>
```

Show Databases:

```
>_MONGOSH

> show dbs;
< admin      48.00 KiB
  config     36.00 KiB
  employee   80.00 KiB
  example    72.00 KiB
  local     104.00 KiB
  test       72.00 KiB
test>
```

09:05
23-06-2023

Create Database:

```
> use Example
< 'switched to db Example'
Example>
```

09:07
23-06-2023

Working Database:

```
> db
< Example
Example>
```

09:16
23-06-2023

Create Collection

```
> use example
< 'switched to db example'
> db.createCollection("ExCollection")
< { ok: 1 }
example>
```

09:18
23-06-2023

Show collections:

```
> show collections
< Employee
  ExCollection
example>
```

09:18
23-06-2023

Insert Data:

```
> db.ExCollection.insertOne({rollno:1,Name:"Sudesh"})
< { acknowledged: true,
    insertedId: ObjectId("6495169141945de88a67319a") }
example>
```

```
> db.ExCollection.insertOne({rollno:6417,Name:"Sudesh Rajbhar"})
< { acknowledged: true,
    insertedId: ObjectId("649517e841945de88a67319b") }
example>
```

09:21
23-06-2023

Drop Collection:

```
> db.ExCollection.drop()
< true
example>
```

09:28
23-06-2023

Drop Database:

```
> db.dropDatabase()
< { ok: 1, dropped: 'example' }
example>
```

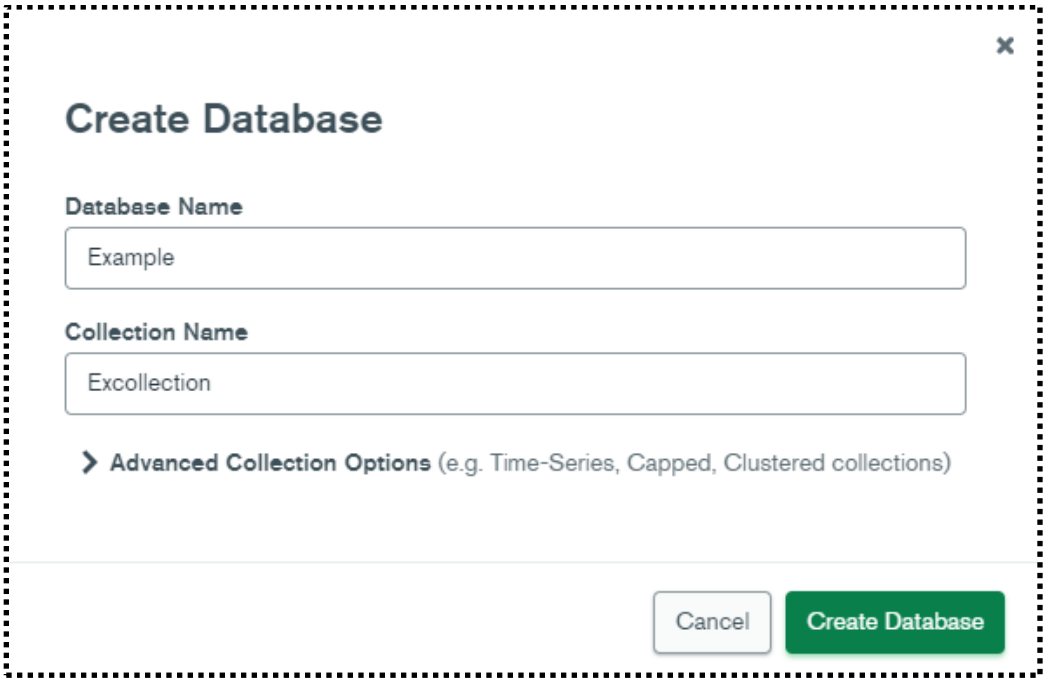
09:29
23-06-2023

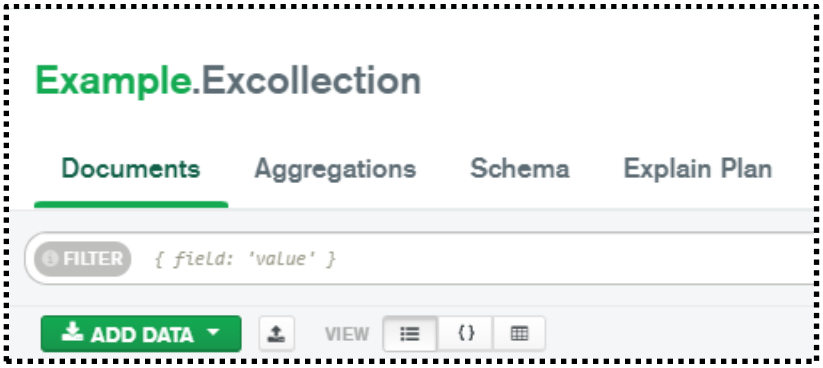
Database "Example" Dropped:

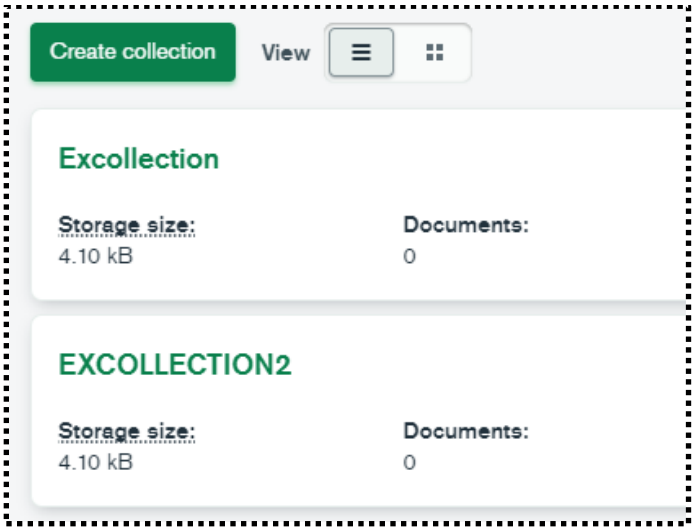
```
> show dbs
< admin      48.00 KiB
  config    108.00 KiB
  employee   80.00 KiB
  local     104.00 KiB
  test       72.00 KiB
example>
```

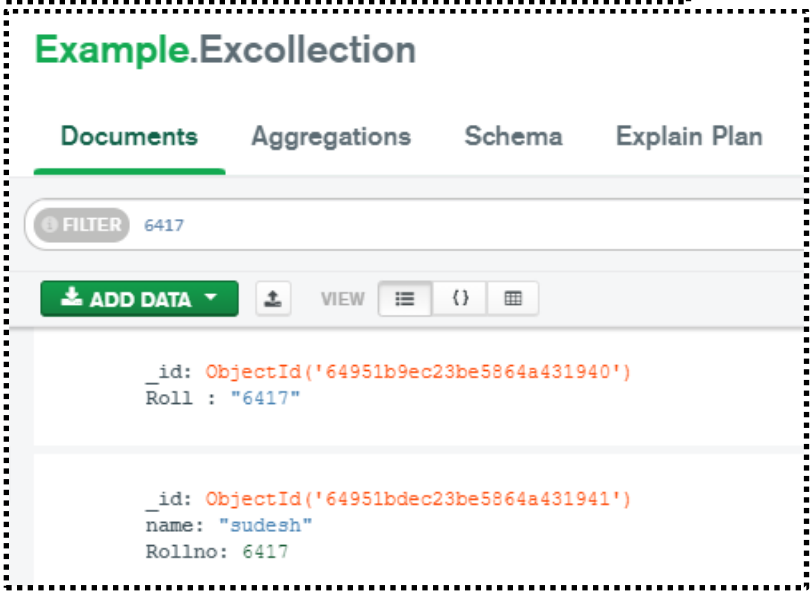
09:31
23-06-2023

C) Using Mongo DB Compass:




Example.Excollection
Documents
Aggregations
Schema
Explain Plan
FILTER
{ field: 'value' }
ADD DATA
VIEW


Create collection
View
Excollection
Storage size:
4.10 kB
Documents:
0
EXCOLLECTION2
Storage size:
4.10 kB
Documents:
0


Example.Excollection
Documents
Aggregations
Schema
Explain Plan
FILTER
6417
ADD DATA
VIEW
_id: ObjectId('64951b9ec23be5864a431940')
Roll : "6417"
_id: ObjectId('64951bdec23be5864a431941')
name: "sudesh"
Rollno: 6417

Question:

BSC IT—

FYIT—

Doc For 2 students

SYIT—

Doc for Marking 5 Subjects

TYIT—

Doc for project List

Create Database:

```
> use BSCIT
< 'switched to db BSCIT'
```

Create 3 Collections:

```
> db.createCollection("FYIT")
< { ok: 1 }
> db.createCollection("SYIT")
< { ok: 1 }
> db.createCollection("TYIT")
< { ok: 1 }
```

Inserting 3 Students in FYIT:

```
> db.FYIT.insertOne({Name:"Sudesh",Rollno:"6417"})
< { acknowledged: true,
    insertedId: ObjectId("64951e8341945de88a67319c") }
> db.FYIT.insertOne({Name:"Farhan",Rollno:"6425"})
< { acknowledged: true,
    insertedId: ObjectId("64951e9b41945de88a67319d") }
> db.FYIT.insertOne({Name:"Vivek",Rollno:"6409"})
< { acknowledged: true,
    insertedId: ObjectId("64951eb441945de88a67319e") }
```

find() students in FYIT:

```
> db.FYIT.find()
< { _id: ObjectId("64951e8341945de88a67319c"),
    Name: 'Sudesh',
    Rollno: '6417' }
  { _id: ObjectId("64951e9b41945de88a67319d"),
    Name: 'Farhan',
    Rollno: '6425' }
  { _id: ObjectId("64951eb441945de88a67319e"),
    Name: 'Vivek',
    Rollno: '6409' }
```

Inserting Marks to Student in SYIT:

```
> db.SYIT.insertOne({Name:"Sudesh",Rollno:"6417","DBMS":"55","ADJAVA":"56","PYTHON":"67","IMP":"23","DM":"45"})
< { acknowledged: true,
    insertedId: ObjectId("649521a641945de88a6731a0") }
> db.SYIT.insertOne({Name:"Farhan",Rollno:"6425","DBMS":"45","ADJAVA":"89","PYTHON":"46","IMP":"90","DM":"12"})
< { acknowledged: true,
    insertedId: ObjectId("649521de41945de88a6731a1") }
> db.SYIT.insertOne({Name:"Vivek",Rollno:"6409","DBMS":"90","ADJAVA":"91","PYTHON":"92","IMP":"93","DM":"94"})
< { acknowledged: true,
    insertedId: ObjectId("649521fd41945de88a6731a2") }
```

```
>_MONGOSH
    Name: 'Sudesh',
    Rollno: '6417',
    DBMS: '55',
    ADJAVA: '56',
    PYTHON: '67',
    IMP: '23',
    DM: '45' }
  { _id: ObjectId("649521de41945de88a6731a1"),
    Name: 'Farhan',
    Rollno: '6425',
    DBMS: '45',
    ADJAVA: '89',
    PYTHON: '46',
    IMP: '90',
    DM: '12' }
  { _id: ObjectId("649521fd41945de88a6731a2"),
    Name: 'Vivek',
    Rollno: '6409',
    DBMS: '90',
    ADJAVA: '91',
    PYTHON: '92',
    IMP: '93',
    DM: '94' }
```

Inserting Project Name in TYIT:

```
> db.TYIT.insertOne({Name:"Vivek",Rollno:"6409",PROJECT:"PIXEL PROGRAMMING"})
< { acknowledged: true,
    insertedId: ObjectId("649522d541945de88a6731a3") }
> db.TYIT.insertOne({Name:"Sudesh",Rollno:"6417",PROJECT:"Zoom Call App"})
< { acknowledged: true,
    insertedId: ObjectId("649522fa41945de88a6731a4") }
> db.TYIT.insertOne({Name:"Farhan",Rollno:"6425",PROJECT:"Arcadium"})
< { acknowledged: true,
    insertedId: ObjectId("649523Od41945de88a6731a5") }
BSCIT>
```

Find() IN TYIT:

```
> db.TYIT.find()
< { _id: ObjectId("649522d541945de88a6731a3"),
    Name: 'Vivek',
    Rollno: '6409',
    PROJECT: 'PIXEL PROGRAMMING' }
  { _id: ObjectId("649522fa41945de88a6731a4"),
    Name: 'Sudesh',
    Rollno: '6417',
    PROJECT: 'Zoom Call App' }
  { _id: ObjectId("6495230d41945de88a6731a5"),
    Name: 'Farhan',
    Rollno: '6425',
    PROJECT: 'Arcadium' }
BSCIT>
```

10:17
23-06-2023

PRACTICAL NO 2: Handling Collections In MONGO DB.

- ❖ Mongo DB Collection
- ❖ Implicit Creation of Collection
- ❖ Explicit Creation of Collection
- ❖ Capped Collection
- ❖ Time Series Collection
- ❖ Creating Collection with Document Validation
- ❖ Clustered Collection, Explain the following types of collection.

- **Mongo DB collections**

a) Implicit Creation of collection

In Mongo DB, collections are created implicitly when you insert documents into them. When you insert a document into a non-existing collection, Mongo DB automatically creates the collection for you.

Example:

```
db.MyCollection_6417.insertOne({Name:"Sudesh Rajbhar",Rollno:6417});
```

```
>_MONGOSH
> use mydb_6417;
< switched to db mydb_6417
> db.myCollection_6417.insertOne({Name:"Sudesh Rajbhar",Rollno:6417});
< {
    acknowledged: true,
    insertedId: ObjectId("649f4b3be3f50fbd6c122b52")
  }
mydb_6417>
```

b) Explicit creation of collection

In Mongo DB, explicit creation of collections refers to the act of creating a collection explicitly before inserting documents into it. Unlike implicit creation, where collections are automatically created when documents are inserted, explicit creation allows you to have more control over the collection's options and settings.

Example:

```
db.createCollection("ExpCollection_6417");
{

db.ExpCollection_6417.insertOne({Name:"Sudesh Dinesh Rajbhar",Rollno:6417});
```

```
>_MONGOSH
> db.createCollection("ExpCollection_6417");
< { ok: 1 }
> db.ExpCollection_6417.insertOne({Name:"Sudesh Dinesh Rajbhar",Rollno:6417});
< {
    acknowledged: true,
    insertedId: ObjectId("649f4c5ae3f50fbd6c122b53")
  }
mydb_6417>
```

c) Capped collection

Capped collections are a special type of collection in Mongo DB that have a fixed size and maintain the insertion order of documents. They are primarily used for capturing and retaining a fixed amount of data, such as logs or event streams.

Example:

A. Create a capped collection named "MyLogCollection' of size 524880. Also specify the maximum number of documents as 100.

```
db.createCollection("MyLogCollection_6417"),{capped:true,size:524880,max:100};
```

```
> db.createCollection("MyLogCollection_6417"),{capped:true,size:524880,max:100};
< { capped: true, size: 524880, max: 100 }
```

Inserting:

```
db.MyLogCollection_6417.insertOne({"id":1,"Name":"Sudesh Rajbhar","Roll":6417});
```

```
> db.MyLogCollection_6417.insertOne({"id":1,"Name":"Sudesh Rajbhar","Roll":6417});
< {
    acknowledged: true,
    insertedId: ObjectId("649f4e30e3f50fbd6c122b54")
  }
```

d) Time Series Collection.

A collection that efficiently stores sequences of measurements over a period of time.

A.Create a time series collection that captures weather data for the past 24 hours.

```
db.createCollection(
   "weather24h",
   {
     timeseries: {
       timeField: "timestamp",
       metaField: "data",
       granularity: "hours"
     },
     expireAfterSeconds: 86400
   }
)
```

```
>_MONGOSH
> db.createCollection("weather24h",
      {timeseries: {
            timeField: "timestamp",
            metaField: "data",
            granularity: "hours"
         },
         expireAfterSeconds: 86400})
< { ok: 1 }
mydb_6417>
```

e) Creating collection with document validation

Create a collection named 'validate' with validation rules and try to insert data that does not meet the validation requirement.

Given Validation Rules/ @jsonSchema:

- BSON Type as Object
- Phone field as required String field
- Email field as string field with @ and . Characters
- Status field with one of the two possible values "Unknown or Incomplete"

```
db.createCollection("validate",{validator:{$jsonSchema:{bsonType:"object",requir
ed:["phone","email","status"],properties:{phone:{bsonType:"int",description:"phon
e no. is required"},email:{bsonType:"string",enum:["@","."],description:"email
required"},status:{bsonType:"string",enum:["unknown","incomplete"],description:"
Must be either of unknown and incomplete"}}}})
```

```
db.createCollection("validate", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: [ "phone", "status" ],
      properties: {
        phone: {
          bsonType: "string",
          description: "Phone - Required."
        },
        status: {
          bsonType: "string",
```

```
        enum:["unknown","incomplete"],
        description: "status Required."
      },
      email: {
        bsonType: "string",
        enum:["@","."],
        description: "Category of post - Optional."
      }
    }
  }
}
})

db.post111.insertOne({phone:1111,status:"unknown",email:"abc@gmail.com"})
```

```
Databases
> db.createCollection("validate1", {
    validator: {
      $jsonSchema: {
        bsonType: "object",
        required: [ "phone", "status" ],
        properties: {
          phone: {
            bsonType: "string",
            description: "Phone - Required."
          },
          status: {
            bsonType: "string",
            enum:["unknown","incomplete"],
            description: "status Required."
          },
          email: {
            bsonType: "string",
            enum:["@","."],
            description: "Category of post - Optional."
          }
        }
      }
    }
  });


< { ok: 1 }
mydb_6417>
```

Inserting a collection that will fail to validate:

```
db.validate1.insertOne({phone:1111,status:"unknown",email:"abc@gmail.com"})
```

```
> db.validate1.insertOne({phone:1111,status:"unknown",email:"abc@gmail.com"})
▸ MongoServerError: Document failed validation
mydb_6417>
```

Creating another validate Collection to insert correct data for validation:

```
db.createCollection("validate1_6417", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["phone", "status"],
      properties: {
        phone: {
          bsonType: "int",
          description: "Phone number is required."
        },
        email: {
          bsonType: "string",
          description: "Email is required.",
          pattern: "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$"
        },
        status: {
          bsonType: "string",
          enum: ["unknown", "incomplete"],
          description: "Must be either 'unknown' or 'incomplete'."
        }
      }
    }
  }
})
```

Inserting:

```
db.validate1_6417.insertOne({
  phone: 1234567890,
  email: "example@example.com",
  status: "unknown"
})
```

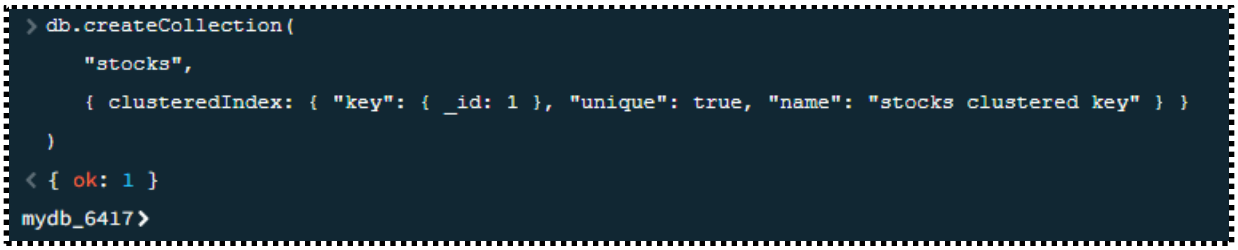
```
> db.validate1_6417.insertOne({
    phone: 1234567890,
    email: "example@example.com",
    status: "unknown"
  })
< {
    acknowledged: true,
    insertedId: ObjectId("64a77e7c3b801c4cfe070efc")
  }
```

f) Clustered collection

A. Create a Clustered collection named "Stocks" with the following features.
B. Cluster index key as 1.
C. Unique Clustered index key Value.
D. Clustered index name "stockCluserKey".

Create collection:

```
db.createCollection(
  "stocks",
  { clusteredIndex: { "key": { _id: 1 }, "unique": true, "name": "stocks clustered key" } }
)
```



Practical No 3: Insert/Import Operations in Mongo DB.

- Mongo DB Data insertions Methods
- Mongo DB Data Tyes
- insert() Method
- insertOne() Methods
- inertMany() Method
- Embedded Documents
- Importing Data from .csv file
- Importing Data from .json file

a) Mongo DB Data insertion Methods

db.collection.insertOne() : Inserts a single document into a collection.

db.collection.insertMany(): Inserts multiple documents into a collection.

Additional Methods for Inserts

The following methods can also add new documents to a collection:

db.collection.updateOne()

db.collection.updateOne(filter, update, options)

db.collection.updateMany()

db.collection.updateMany(filter, update, options)

db.collection.findAndModify()

db.collection.findAndModify(document)

db.collection.findOneAndUpdate()

db.collection.findOneAndUpdate(filter, update, options)

db.collection.findOneAndReplace()

db.collection.findOneAndReplace(filter, replacement, options)

db.collection.bulkWrite()

db.collection.bulkWrite()

b) Mongo DB Data Types

1. Explain the following data types in Mongo DB.

String: Used to store textual data. It can represent both single-word strings and longer text.

Integer: Represents whole numbers without decimal places. It can be used for storing numeric values, such as counts or identifiers.

Double: Represents floating-point numbers with decimal places. It is suitable for storing values that require precision, such as measurements or financial data.

Boolean: Represents a boolean value, which can be either true or false. It is often used for logical conditions and flags.

Date: Stores date and time information. It can be used for storing timestamps or any other time-related data.

Array: Stores an ordered list of values. Arrays can contain values of any data type, including other arrays or documents.

Object: Represents a document within Mongo DB. It is similar to a JSON object and consists of key-value pairs. Keys are strings, and values can be of any data type, including nested objects or arrays.

Object ID: A special type provided by Mongo DB to uniquely identify documents within a collection. ObjectId values are generated automatically and typically used as the document's primary key.

Null: Represents the absence of a value. It can be assigned to any field to indicate that it does not have a value.

c) insert() method

1. Insert the following information into the Student collection of the college database using insert() method.

Name:"Sudesh",Class:"TYBSCIT", College:"Rjcollege", Percentage:85.00% and roll no.

```
> db.Student.insert({
    Name: "Sudesh",Roll_no: 6417,
    Class: "TYBSCIT",
    College: "Rjcollege",
    Percentage: 85.00
  })
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("64a78c5c3b801c4cfe070f01")
    }
  }
College_6417>
```

09:24
07-07-2023 8

```
db.Student.insert({
  Name: "Sudesh",Roll_no: 6417,
  Class: "TYBSCIT",
  College: "Rjcollege",
  Percentage: 85.00
})
```

d) insertOne() Method

1.Insert the following information into the Student collection of college database using insertOne() method.

Name:"Sudesh",Class:"TYBSCIT", College:"Rjcollege", Percentage:85.00% and roll number.

```
> db.Student.insertOne({
    Name: "Sudesh Rajbhar",Roll_no:6417,
    Class: "TYBSCIT-g division",
    College: "Rjcollege science arts and commeerce",
    Percentage: 99.00
  })
< {
    acknowledged: true,
    insertedId: ObjectId("64a78c9c3b801c4cfe070f02")
  }
College_6417>
```

09:25 8
07-07-2023

```
db.Student.insertOne({
  Name: "Sudesh Rajbhar",Roll_no:6417,
  Class: "TYBSCIT-g division",
  College: "Rjcollege science arts and commeerce",
  Percentage: 99.00
})
```

d) insertMany() method.

Name:"Sudesh",Class:"TYBSCIT", College:"Rjcollege", Percentage:85.00%
Name:"Pranay",Class:"SYBSCIT", College:"Somaiya college", Percentage:80.00%
Name:"Vishal",Class:"FYBSCIT", College:"Gurunanak", Percentage:90.00%
And their roll numbers.

```
> db.Student.insertMany([
    {
      Name: "Sudesh",Roll_no:6417,
      Class: "TYBSCIT",
      College: "Rjcollege",
      Percentage: 85.00
    },
    {
      Name: "Pranay",Roll_no:6424,
      Class: "SYBSCIT",
      College: "Somaiya college",
      Percentage: 80.00
    },
    {
      Name: "Vishal",Roll_no:6423,
      Class: "FYBSCIT",
      College: "Gurunanak",
      Percentage: 90.00
    }
  ])
```

```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("64a78e323b801c4cfe070f03"),
      '1': ObjectId("64a78e323b801c4cfe070f04"),
      '2': ObjectId("64a78e323b801c4cfe070f05")
    }
  }
College_6417>
```

```
db.Student.insertMany([
 {
  Name: "Sudesh",Roll_no:6417,
  Class: "TYBSCIT",
  College: "Rjcollege",
  Percentage: 85.00
 },
 {
  Name: "Pranay",Roll_no:6424,
  Class: "SYBSCIT",
  College: "Somaiya college",
  Percentage: 80.00
 },
 {
  Name: "Vishal",Roll_no:6423,
  Class: "FYBSCIT",
  College: "Gurunanak",
  Percentage: 90.00
 }
])
```

e) Embedded Documents

1. Insert the following information into the Student collection of the college database with the embedded documents for the field of address.

Name:"Sudesh",Roll_no:6417,Class:"TYBSCIT",College:"MCC",Address:{BuildNo:01, Street:"Jn Road",City:"Mumbai"} ,PerMarks:"75.00%"

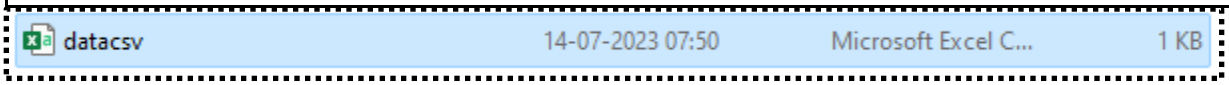
```
db.Student.insertOne({
  Name: "Sudesh",
  Roll_no: 6417,
  Class: "TYBSCIT",
  College: "MCC",
  Address: {
    BuildNo: 01,
    Street: "Jn Road",
    City: "Mumbai"
  },
  PerMarks: "75.00%"
})
```

f) Importing data from .csv files.

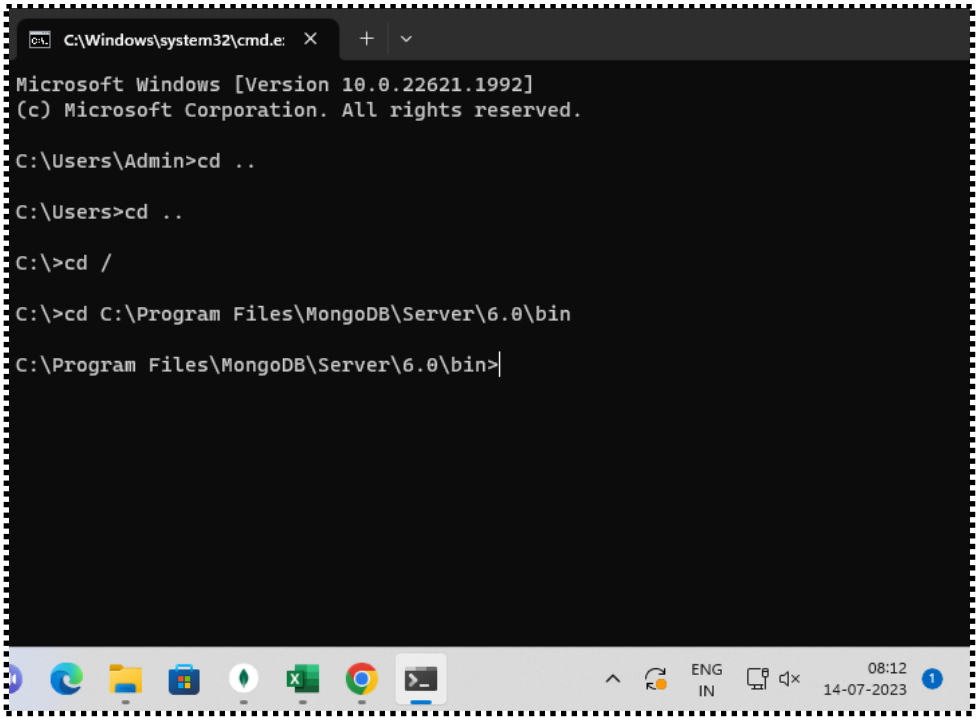
	A	B	C	D	E
1	Timestamp	Email address	Roll no	First Name	Last Name
2	06/14/22 09:17:18	sudeshdr03@gmail.com	6417	Sudesh	Rajbhar
3	06/15/22 10:17:18	pranayss@gmail.com	6424	Pranay	Sawant

Datacsv.csv:

https://drive.google.com/file/d/1JQyEB0O013XVVHv9E3WiK0iz_dlv9ZG2/view?usp=drive_link



Setup the path:




```
C:\Program Files\MongoDB\Server\6.0\bin\mongodb-database-tools-windows-x86_64-100.7.3\bin>mongoimport.exe --host localh
ost --db 6417_Sudesh --collection students --type csv --file c:\datacsv.csv --headerline
2023-07-14T08:35:37.737+0530    connected to: mongodb://localhost/
2023-07-14T08:35:37.762+0530    2 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Server\6.0\bin\mongodb-database-tools-windows-x86_64-100.7.3\bin>
```

```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd ..

C:\Users>cd ..

C:\>cd /

C:\>cd C:\Program Files\Mongo DB\Server\6.0\bin

C:\Program Files\Mongo DB\Server\6.0\bin>cd Mongo DB-database-tools-
windows-x86_64-100.7.3\bin


C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-
windows-x86_64-100.7.3\bin>mongoimport.exe --host localhost --db
6417_Sudesh --collection students --type csv --file c:\datacsv.csv

C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-
windows-x86_64-100.7.3\bin>mongoimport.exe --host localhost --db
6417_Sudesh --collection students --type csv --file c:\datacsv.csv --headerline
2023-07-14T08:35:37.737+0530    connected to: Mongo DB://localhost/
2023-07-14T08:35:37.762+0530    2 document(s) imported successfully. 0
document(s) failed to import.

C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-
windows-x86_64-100.7.3\bin>
```

Output:

```
C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-windows-x86_64-100.7.3\bin>type c:\datacsv.csv
Timestamp,Email address,Roll no,First Name,Last Name
06/14/22 09:17:18,sudeshdr03@gmail.com,6417,Sudesh ,Rajbhar
06/15/22 10:17:18,pranayss@gmail.com,6424,Pranay,Sawant
```

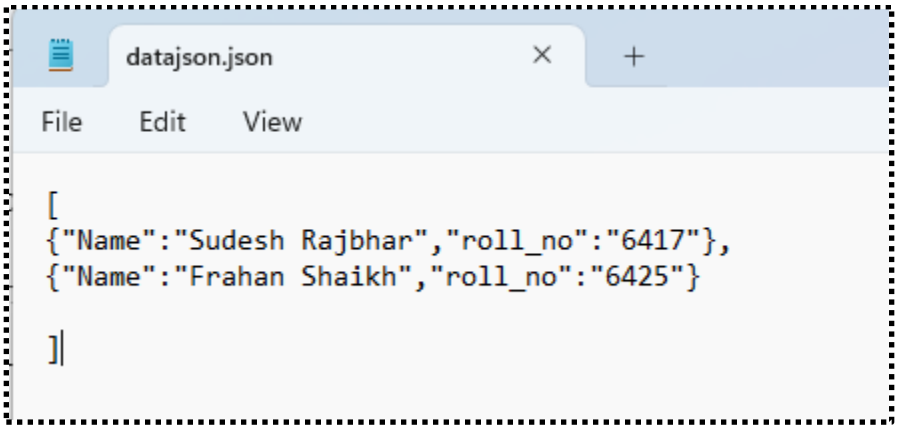
```
C:\Program Files\MongoDB\Server\6.0\bin\mongodb-database-tools
Timestamp,Email address,Roll no,First Name,Last Name
06/14/22 09:17:18,sudeshdr03@gmail.com,6417,Sudesh ,Rajbhar
06/15/22 10:17:18,pranayss@gmail.com,6424,Pranay,Sawant
```

Output in Shell

```
>_MONGOSH

> db.students.find()
< {
    _id: ObjectId("64b0bb8114466e1d085abc9d"),
    Timestamp: '06/14/22 09:17:18',
    'Email address': 'sudeshdr03@gmail.com',
    'Roll no': 6417,
    'First Name': 'Sudesh',
    'Last Name': 'Rajbhar'
  }
  {
    _id: ObjectId("64b0bb8114466e1d085abc9e"),
    Timestamp: '06/15/22 10:17:18',
    'Email address': 'pranayss@gmail.com',
    'Roll no': 6424,
    'First Name': 'Pranay',
    'Last Name': 'Sawant'
  }
6417_Sudesh>
```

g) Importing data from .json files.

Document:




```
[
{"Name":"Sudesh Rajbhar","roll_no":"6417"},
{"Name":"Frahan Shaikh","roll_no":"6425"}

]
```

Command Line:


```
C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-windows-x86_64-100.7.3\bin>mongoimport.exe --jsonArray --db 6417_Sudesh --collection students --file D:\datajson.json
2023-07-14T09:01:15.512+0530    connected to: Mongo DB://localhost/
2023-07-14T09:01:15.553+0530    2 document(s) imported successfully. 0 document(s) failed to import.
```

Output:

```
C:\Program Files\MongoDB\Server\6.0\bin\mongodb-dat
[
{"Name":"Sudesh Rajbhar","roll_no":"6417"},
{"Name":"Frahan Shaikh","roll_no":"6425"}

]
```

C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-windows-x86_64-100.7.3\bin>type d:\datajson.json
[
{"Name":"Sudesh Rajbhar","roll_no":"6417"},
{"Name":"Frahan Shaikh","roll_no":"6425"}
]

Output in shell:

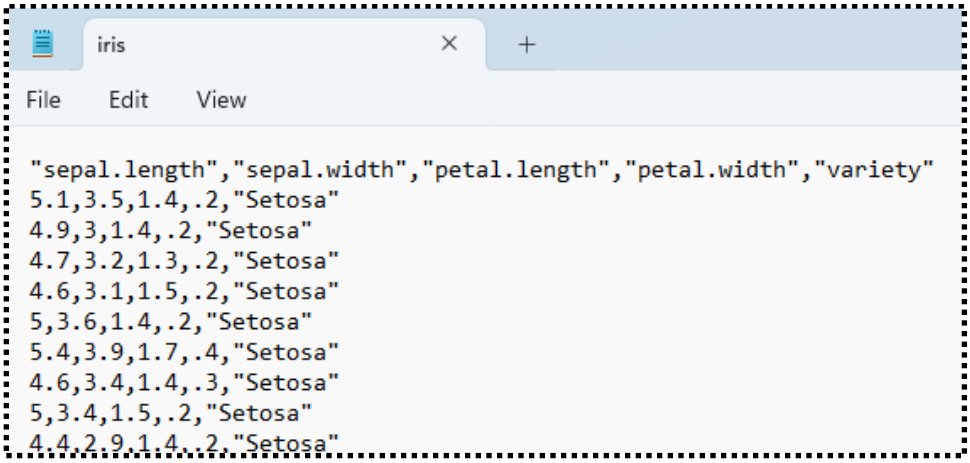
Practical No 4: Querying Mongo DB Database

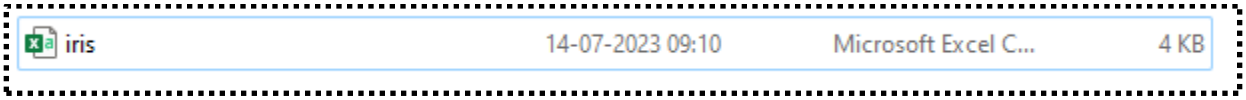
TOOL LINK:

https://fastdl.Mongo DB.org/tools/db/Mongo DB-database-tools-windows-x86_64-100.7.4.zip

- find() and findOne() Method
- pretty() Method
- Filtering criteria in Mongo DB Queries/ Selecting Queries
- Using operators in Queries
- Projection Queries
- limit() Method
- skip() Method
- Regular Expression in Mongo DB

Iris.csv:





Creating Database and Collection:

```
> use Species
< switched to db Species
> db.createCollection("Flower")
< { ok: 1 }
```

Command line:

```
C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-windows-x86_64-100.7.3\bin>mongoimport.exe --host localhost --db Species --collection Flower --type csv --file c:\iris.csv --headerline
2023-07-14T09:15:35.226+0530    connected to: Mongo DB://localhost/
2023-07-14T09:15:35.248+0530    150 document(s) imported successfully. 0 document(s) failed to import.
```

Output:

```
C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-windows-x86_64-100.7.3\bin>type c:\iris.csv
"sepal.length","sepal.width","petal.length","petal.width","variety"
5.1,3.5,1.4,.2,"Setosa"
4.9,3,1.4,.2,"Setosa"
4.7,3.2,1.3,.2,"Setosa"
4.6,3.1,1.5,.2,"Setosa"
```

Output in shell:

a) find()

Displaying data of only Setosa Variety:

```
> db.Flower.find({"variety":"Setosa"})
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddbf8"),
    sepal: {
      length: 4.9,
      width: 3
    },
    petal: {
      length: 1.4,
      width: 0.2
    },
    variety: 'Setosa'
  }
  {
    _id: ObjectId("64b0c4dfba3139d5f9cddbf9"),
    sepal: {
      length: 5.1,
      width: 3.5
    },
    petal: {
      length: 1.4,
      width: 0.2
    },
    variety: 'Setosa'
  }
```

Find the sepal width greater than 3.0 for all the Setosa variety.

```
> db.Flower.find({ "variety":"Setosa","sepal.width": { $gt: 3.0 } })
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddbf9"),
    sepal: {
      length: 5.1,
      width: 3.5
    },
    petal: {
      length: 1.4,
      width: 0.2
    },
    variety: 'Setosa'
  }
  {
    _id: ObjectId("64b0c4dfba3139d5f9cddbfa"),
    sepal: {
      length: 4.6,
      width: 3.1
    },
    petal: {
      length: 1.5,
      width: 0.2
    },
    variety: 'Setosa'
  }
```

```
> db.Flower.findOne({ "variety":"Setosa","sepal.width": { $gt: 3.0 } })
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddbf9"),
    sepal: {
      length: 5.1,
      width: 3.5
    },
    petal: {
      length: 1.4,
      width: 0.2
    },
    variety: 'Setosa'
  }
```

b) findOne()

Find the sepal width greater than 3.0 and sepal length less than 3.0 for all the Setosa variety.

```
> db.Flower.findOne({ "variety":"Setosa","sepal.width": { $gt: 3.0},"sepal.length":{$lt:3.0} })
< null
```

Find the sepal width greater than 3.0 and sepal width less than 3.0 for all the Setosa varieties.

```
> db.Flower.findOne({ "variety":"Setosa","sepal.width": { $gt: 3.0},"sepal.width":{$lt:3.0} })
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddbff"),
    sepal: {
      length: 4.4,
      width: 2.9
    },
    petal: {
      length: 1.4,
      width: 0.2
    },
    variety: 'Setosa'
  }
```

Find the sepal width greater than 3.0 and sepal length less than 3.0 for all the Setosa varieties.

```
> db.Flower.findOne({$and: [{ "sepal.width": { $gt: 3.0 } },{ "sepal.length": { $lt: 3.0}}]})
< null
```

Find the varieties of Setosa and virginica:

```
> db.Flower.findOne({$or: [{ "variety": "Setosa" },{ "variety":"Virginica"}]})
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddbf8"),
    sepal: {
      length: 4.9,
      width: 3
    },
    petal: {
      length: 1.4,
      width: 0.2
    },
    variety: 'Setosa'
  }
Species >
```

c) pretty()

Without pretty:

```
> db.Flower.find({"variety":"Setosa"})
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddbf8"),
    sepal: {
      length: 4.9,
      width: 3
    },
    petal: {
      length: 1.4,
      width: 0.2
    },
    variety: 'Setosa'
  }
```

With pretty:

```
> db.Flower.find({"variety":"Setosa"}).pretty()
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddbf8"),
    sepal: {
      length: 4.9,
      width: 3
    },
    petal: {
      length: 1.4,
      width: 0.2
    },
    variety: 'Setosa'
  }
```

c) Filtering criteria on Mongo DB Queries/selection Queries

Find one data for Setosa or virginica flowers that have sepal width greater than 3.0 and sepal.length less than 6.0

```
> db.Flower.findOne({$or: [{ "variety": "Setosa" },{ "variety":"Virginica"}],$and: [{ "sepal.width": { $gt: 3.0 } },{ "sepal.length": { $lt: 6.0}}]})
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddbf9"),
    sepal: {
      length: 5.1,
      width: 3.5
    },
    petal: {
      length: 1.4,
      width: 0.2
    },
    variety: 'Setosa'
  }
Species >
```

Display one documents for the versicolor species having sepal length equal to 6.4

```
> db.Flower.findOne({ "variety":"Versicolor","sepal.length": 6.4 })
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddc29"),
    sepal: {
      length: 6.4,
      width: 3.2
    },
    petal: {
      length: 4.5,
      width: 1.5
    },
    variety: 'Versicolor'
  }
Species>
```

Display all documents for the versicolor species having sepal length not equal to 4.3

```
> db.Flower.findOne({ "variety":"Versicolor","sepal.length": { $ne: 4.3 } })
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddc29"),
    sepal: {
      length: 6.4,
      width: 3.2
    },
    petal: {
      length: 4.5,
      width: 1.5
    },
    variety: 'Versicolor'
  }
Species>
```

Display all species not having the petal length as 1.3 ,1.5 or 1.7($nin)

```
> db.flower.findOne({"sepal.length":{$nin:[1.3,1.5,1.7]}})
< null
Species >
```

Display all species having the petal length is either 1.3,1.5 or 1.7($in)

```
> db.flower.findOne({"sepal.length":{$in:[1.3,1.5,1.7]}})
< null
Species >
```

Sort()

```
> db.Flower.find({ "variety": "Setosa" }).sort({ "_id": 1 })
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddbf8"),
    sepal: {
      length: 4.9,
      width: 3
    },
    petal: {
      length: 1.4,
      width: 0.2
    },
    variety: 'Setosa'
  }
```

Descending sort()

```
> db.Flower.find({ "variety": "Setosa" }).sort({"variety":-1, "_id": 1 })
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddbf8"),
    sepal: {
      length: 4.9,
      width: 3
    },
    petal: {
      length: 1.4,
      width: 0.2
    },
    variety: 'Setosa'
  }
```

Find all flowers with sepal .width greater than 2.0 in descending order and limit it to 4 documents.

```
> db.Flower.find({ "sepal.width": {$gt:2.0} }).sort({"variety":-1, "_id": 1 }).limit(4)
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddc5b"),
    sepal: {
      length: 6.3,
      width: 3.3
    },
    petal: {
      length: 6,
      width: 2.5
    },
    variety: 'Virginica'
  }
  {
    _id: ObjectId("64b0c4dfba3139d5f9cddc5c"),
    sepal: {
      length: 5.8,
      width: 2.7
    },
    petal: {
      length: 5.1,
      width: 1.9
```

Find all flowers with sepal .width greater than 2.0 in descending order and skip first 3 documents.

```
> db.Flower.find({ "sepal.width": {$gt:2.0} }).sort({"variety":-1, "_id": 1 }).skip(3)
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddc5e"),
    sepal: {
      length: 6.3,
      width: 2.9
    },
    petal: {
      length: 5.6,
      width: 1.8
    },
    variety: 'Virginica'
  }
  {
    _id: ObjectId("64b0c4dfba3139d5f9cddc5f"),
    sepal: {
      length: 4.9,
      width: 2.5
    },
    petal: {
      length: 4.5,
      width: 1.7
    },
    variety: 'Virginica'
  }
```

Find all flowers with sepal .width greater than 2.0 in descending order and skip first 10 documents and limit it to 5 documents.

```
> db.Flower.find({ "sepal.width": {$gt:2.0} }).sort({"variety":-1, "_id": 1 }).skip(10).limit(5)
< {
    _id: ObjectId("64b0c4dfba3139d5f9cddc66"),
    sepal: {
      length: 6.5,
      width: 3.2
    },
    petal: {
      length: 5.1,
      width: 2
    },
    variety: 'Virginica'
  }
```

Write a json file named employee.json and import into Mongo DB.

Myjson.json:

```
[
  {
    "id": 1,
    "first_name": "Sudesh", "last_name": "Rajbhar",
    "email" : "sudesh@gmail.com",  "salary" : 50000,
    "skills" : ["Angular" , "React" , "Mongo DB" , "Ajava" , "C#"],
    "department" : "IT"
  } ,
  {
    "id": 2,
    "first_name": "Sumit", "last_name": "Surve",
    "email" : "sumit@gmail.com",  "salary" : 3000,
    "skills" : ["Accounting" , "Tax" ],
    "department" : "Finance"
  } ,
  {
    "id": 3,
    "first_name": "Niketan", "last_name": "Singh",
    "email" : "niketan@gmail.com",  "salary" : 44500,
    "skills" : ["Sales" , "Marketing" ],
    "department" : "Finance"
  } ,
  {
    "id": 4,
    "first_name": "Vijay", "last_name": "Vishwakarma",
    "email" : "vijay@gmail.com",  "salary" : 22000,
    "skills" : ["Ajava" ],
    "department" : "CS"
  } ,
  {
    "id": 5,
    "first_name": "Kaushal", "last_name": "Shukla",
    "email" : "kaushal@gmail.com",  "salary" : 20000,
    "skills" : ["Accounting" , "Tax" ],
    "department" : "Finance"
  } ,
```

```
  {
     "id": 6,
     "first_name": "Mita", "last_name": "Yadav",
     "email" : "mita@gmail.com",  "salary" : 20000,
     "skills" : ["ASP.NET" , "Blockchain" ],
     "department" : "CS"
  } ,
  {
     "id": 7,
     "first_name": "Sumita", "last_name": "Mahale",
     "email" : "sumita@gmail.com",  "salary" : 15000,
     "skills" : ["Networking"],
     "department" : "IT"
  } ,
  {
     "id": 8,
     "first_name": "Anita", "last_name": "Kulkarni",
     "email" : "anita@gmail.com",  "salary" : 34000,
     "skills" : ["SAP" , "Analytics" ],
     "department" : "IT"
  } ,
  {
     "id": 9,
     "first_name": "Keval", "last_name": "Khona",
     "email" : "keval@gmail.com",  "salary" : 57000,
     "skills" : ["SAP"],
     "department" : "Stock"
  }
]
```


employee.json:
https://drive.google.com/file/d/1-IfsR8wfgyDBQuOmknKGFOc1J-ZY8Bzy/view?usp=sharing

Import into Mongo DB:

```
C:\>cd C:\Program Files\MongoDB\Server\6.0\bin

C:\Program Files\MongoDB\Server\6.0\bin>cd mongodb-database-tools-windows-x86_64-100.7.4

C:\Program Files\MongoDB\Server\6.0\bin\mongodb-database-tools-windows-x86_64-100.7.4>cd bin
```

C:\>cd C:\Program Files\Mongo DB\Server\6.0\bin

C:\Program Files\Mongo DB\Server\6.0\bin>cd Mongo DB-database-tools-windows-x86_64-100.7.4

C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-windows-x86_64-100.7.4>cd bin

```
C:\Program Files\MongoDB\Server\6.0\bin\mongodb-database-tools-windows-x86_64-100.7.4\bin>mongoimport.exe --jsonArray --
db office --collection employee --file D:\employee.json
2023-07-22T03:49:32.280-0700    connected to: mongodb://localhost/
2023-07-22T03:49:33.221-0700    9 document(s) imported successfully. 0 document(s) failed to import.
```

C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-windows-x86_64-100.7.4\bin>mongoimport.exe --jsonArray --db office --collection employee --file D:\employee.json
2023-07-22T03:49:32.280-0700 connected to: Mongo DB://localhost/
2023-07-22T03:49:33.221-0700 9 document(s) imported successfully. 0 document(s) failed to import.

C:\Windows\system32\cmd.exe

```
C:\Program Files\MongoDB\Server\6.0\bin\mongodb-database-tools-windows-x
[
    {
        "id": 1,
        "first_name": "Sudesh", "last_name": "Rajbhar",
        "email" : "sudesh@gmail.com",  "salary" : 50000,
        "skills" : ["Angular" , "React" , "Mongodb" , "Ajava" , "C#"],
        "department" : "IT"
    } ,
    {
        "id": 2,
        "first_name": "Sumit", "last_name": "Surve",
        "email" : "sumit@gmail.com",  "salary" : 3000,
        "skills" : ["Accounting" , "Tax" ],
        "department" : "Finance"
    } ,
    {
        "id": 3,
        "first_name": "Niketan", "last_name": "Singh",
        "email" : "niketan@gmail.com",  "salary" : 44500,
        "skills" : ["Sales" , "Marketing" ],
        "department" : "Finance"
    } ,
    {
        "id": 4,
        "first_name": "Vijay", "last_name": "Vishwakarma",
        "email" : "vijay@gmail.com",  "salary" : 22000,
        "skills" : ["Ajava" ],
        "department" : "CS"
    } ,
    {
        "id": 5,
        "first_name": "Kaushal", "last_name": "Shukla",
        "email" : "kaushal@gmail.com",  "salary" : 20000,
        "skills" : ["Accounting" , "Tax" ],
        "department" : "Finance"
    } ,
    {
        "id": 6,
        "first_name": "Mita", "last_name": "Yadav",
        "email" : "mita@gmail.com",  "salary" : 20000,
        "skills" : ["ASP.NET" , "Blockchain" ],
        "department" : "CS"
    } ,
    {
        "id": 7,
        "first_name": "Sumita", "last_name": "Mahale",
        "email" : "sumita@gmail.com",  "salary" : 15000,
        "skills" : ["Networking"],
        "department" : "IT"
    } ,
    {
```

Create database named Office and employee collection and find data

```
> use office
< switched to db office
> db.createCollection("employee")
< { ok: 1 }
```

1. Display the first occurrence of the first name Sudesh. (findOne)

```
>_MONGOSH
> db.employee.findOne({"first_name":"Sudesh"})
< {
    _id: ObjectId("64bbb43c64b383809012076e"),
    id: 1,
    first_name: 'Sudesh',
    last_name: 'Rajbhar',
    email: 'sudesh@gmail.com',
    salary: 50000,
    skills: [
      'Angular',
      'React',
      'Mongodb',
      'Ajava',
      'C#'
    ],
    department: 'IT'
  }
office>
```

2. Display the mail id of an employee having first name as vijay.

```
> db.employee.find({"first_name":"Vijay"},{"email":1})
< {
    _id: ObjectId("64bbb43c64b3838090120771"),
    email: 'vijay@gmail.com'
  }
office>
```

3. Display the list of employees having salary more than 2000 in ascending order of salaries. Also exclude the id field from the result.

```
>_MONGOSH
> db.employee.find({"salary":{$gt:2000}},{"_id":0,"excludeField":0}).sort({"salary":1})
< {
    id: 2,
    first_name: 'Sumit',
    last_name: 'Surve',
    email: 'sumit@gmail.com',
    salary: 3000,
    skills: [
      'Accounting',
      'Tax'
    ],
    department: 'Finance'
  }
  {
    id: 7,
    first_name: 'Sumita',
    last_name: 'Mahale',
    email: 'sumita@gmail.com',
    salary: 15000,
    skills: [
      'Networking'
    ],
    department: 'IT'
  }
```

4. Display the list of employees having salaries in the range of 30000 to 40000.

```
> db.employee.find({"salary":{$gte: 30000, $lte: 40000}})
< {
    _id: ObjectId("64bbb43c64b3838090120774"),
    id: 8,
    first_name: 'Anita',
    last_name: 'Kulkarni',
    email: 'anita@gmail.com',
    salary: 34000,
    skills: [
      'SAP',
      'Analytics'
    ],
    department: 'IT'
  }
office>
```

5)get the list of employees from the IT department.

```
>_MONGOSH
> db.employee.find({"department":"IT"})
< {
    _id: ObjectId("64bbb43c64b383809012076e"),
    id: 1,
    first_name: 'Sudesh',
    last_name: 'Rajbhar',
    email: 'sudesh@gmail.com',
    salary: 50000,
    skills: [
      'Angular',
      'React',
      'Mongodb',
      'Ajava',
      'C#'
    ],
    department: 'IT'
  }
```

```
  {
    _id: ObjectId("64bbb43c64b3838090120776"),
    id: 7,
    first_name: 'Sumita',
    last_name: 'Mahale',
    email: 'sumita@gmail.com',
    salary: 15000,
    skills: [
      'Networking'
    ],
    department: 'IT'
  }
office>
```

6. Get the list of employees from various departments with the skill tax.

```
>_MONGOSH
> db.employee.find({"skills":"Tax"})
< {
    _id: ObjectId("64bbb43c64b383809012076f"),
    id: 2,
    first_name: 'Sumit',
    last_name: 'Surve',
    email: 'sumit@gmail.com',
    salary: 3000,
    skills: [
      'Accounting',
      'Tax'
    ],
    department: 'Finance'
  }
  {
    _id: ObjectId("64bbb43c64b3838090120772"),
    id: 5,
    first_name: 'Kaushal',
    last_name: 'Shukla',
    email: 'kaushal@gmail.com',
    salary: 20000,
    skills: [
      'Accounting',
      'Tax'
    ],
    department: 'Finance'
  }
office>
```

7. Get the list of employees from various departments with the skill either tax or accounting.

```
>_MONGOSH

> db.employee.find({"skills":{$in: ["Tax","Accounting"]}})
< {
    _id: ObjectId("64bbb43c64b383809012076f"),
    id: 2,
    first_name: 'Sumit',
    last_name: 'Surve',
    email: 'sumit@gmail.com',
    salary: 3000,
    skills: [
      'Accounting',
      'Tax'
    ],
    department: 'Finance'
  }
  {
    _id: ObjectId("64bbb43c64b3838090120772"),
    id: 5,
    first_name: 'Kaushal',
    last_name: 'Shukla',
    email: 'kaushal@gmail.com',
    salary: 20000,
    skills: [
      'Accounting',
      'Tax'
    ],
    department: 'Finance'
  }
office>
```

8)display the list of employees with their name and department

```
>_MONGOSH
> db.employee.find({},{"first_name":1,"last_name":1,"department":1,"_id":0})
< {
    first_name: 'Sudesh',
    last_name: 'Rajbhar',
    department: 'IT'
  }
  {
    first_name: 'Sumit',
    last_name: 'Surve',
    department: 'Finance'
  }
  {
    first_name: 'Niketan',
    last_name: 'Singh',
    department: 'Finance'
  }
  {
    first_name: 'Vijay',
    last_name: 'Vishwakarma',
    department: 'CS'
  }
  {
    first_name: 'Kaushal',
    last_name: 'Shukla',
    department: 'Finance'
  }
```

9. List the employees with mail id kaushal@gmail.com

```
> db.employee.find({"email":"kaushal@gmail.com"})
< {
    _id: ObjectId("64bbb43c64b3838090120772"),
    id: 5,
    first_name: 'Kaushal',
    last_name: 'Shukla',
    email: 'kaushal@gmail.com',
    salary: 20000,
    skills: [
      'Accounting',
      'Tax'
    ],
    department: 'Finance'
  }
```

10. Get the list of employees from IT & CS departments with the skill of ASP.NET or blockchain.

```
< {
    first_name: 'Mita',
    last_name: 'Yadav',
    skills: [
      'ASP.NET',
      'Blockchain'
    ],
    department: 'CS'
  }
office>
```

>_MONGOSH

```
> db.employee.find(
    {
      $or: [
        {
          $and: [
            { "department": "IT" },
            { "skills": { $in: ["ASP.NET", "Blockchain"] } }
          ]
        },
        {
          $and: [
            { "department": "CS" },
            { "skills": { $in: ["ASP.NET", "Blockchain"] } }
          ]
        }
      ]
    },
    {
      "first_name": 1,
      "last_name": 1,
      "department": 1,
      "skills": 1,
      "_id": 0
    }
  )
```

11. Update the name of employee to amita who has employee id as 1.

```
db.employee.updateOne({ "_id": 1 },{ $set: { "first_name": "Amita" } })
```

```
> db.employee.updateOne({ "_id": 1 },{ $set: { "first_name": "Amita" } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 0
  }
```

12. Increment the slaary of all employees by 600 upees.

```
db.employee.updateMany({"_id":1},{$inc:{salary:600}})
```

```
> db.employee.updateMany({"_id":1},{$inc:{salary:600}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 0
  }
office>
```

13. List the employee with the highest salary.

```
db.employee.aggregate([{$group:{_id:null, salMAx:{$max:"$salary"}}}])
```

```
> db.employee.aggregate([
    {
      $group: {
        _id: null,
        salMax: { $max: "$salary" }
      }
    }
  ])
< {
    _id: null,
    salMax: 57000
  }
```

```
db.employee.find().sort({ "salary": -1 }).limit(1)
```

```
> db.employee.find().sort({ "salary": -1 }).limit(1)

< {
    _id: ObjectId("64bbb43c64b3838090120775"),
    id: 9,
    first_name: 'Keval',
    last_name: 'Khona',
    email: 'keval@gmail.com',
    salary: 57000,
    skills: [
      'SAP'
    ],
    department: 'Stock'
  }
```

14. Multiply the salary of the first employee by 30.

```
db.employee.updateOne({"_id":1},{$mul:{salary:30}})
```

```
> db.employee.updateOne({"_id":1},{$mul:{salary:30}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 0
  }
office>
```

BEFORE:/AFTER:

```
< {
    _id: ObjectId("64bbb43c64b383809012076e"),
    id: 1,
    first_name: 'Sudesh',
    last_name: 'Rajbhar',
    email: 'sudesh@gmail.com',
    salary: 50000,
    skills: [
      'Angular',
      'React',
      'Mongodb',
      'Ajava',
      'C#'
    ],
    department: 'IT'
  }
```

```
> db.employee.find({"id":1})
< {
    _id: ObjectId("64bbb43c64b383809012076e"),
    id: 1,
    first_name: 'Sudesh',
    last_name: 'Rajbhar',
    email: 'sudesh@gmail.com',
    salary: 1500000,
    skills: [
      'Angular',
      'React',
      'Mongodb',
      'Ajava',
      'C#'
    ],
    department: 'IT'
  }
```

15. Unset the email id of all employees.

```
db.employee.updateMany({},{$unset:{"email":""}})
```

```
> db.employee.updateMany({},{$unset:{"email":""}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 9,
    modifiedCount: 9,
    upsertedCount: 0
  }
```

After Updation: Email Is Gone:

```
< {
    _id: ObjectId("64bbb43c64b383809012076e"),
    id: 1,
    first_name: 'Sudesh',
    last_name: 'Rajbhar',
    salary: 1500000,
    skills: [
      'Angular',
      'React',
      'Mongodb',
      'Ajava',
      'C#'
    ],
    department: 'IT'
  }
```

16. Rename the first_name field to firstName

```
db.employee.updateMany({},{$rename:{"first_name":"firstName"})
```

```
> db.employee.updateMany({},{$rename:{"first_name":"firstName"}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 9,
    modifiedCount: 9,
    upsertedCount: 0
  }
office>
```

AFTER:

```
< {
    _id: ObjectId("64bbb43c64b383809012076e"),
    id: 1,
    last_name: 'Rajbhar',
    salary: 1500000,
    skills: [
      'Angular',
      'React',
      'Mongodb',
      'Ajava',
      'C#'
    ],
    department: 'IT',
    firstName: 'Sudesh'
  }
```

17. Update the skills of "anita" from SAP to ASP. (Updating array)

```
db.employee.updateOne({ "firstName": "Anita" },{ $set: { "skills.$[element]": "
ASP" } },{ arrayFilters: [{ "element": "SAP" }] })
```

```
> db.employee.updateOne({ "firstName": "Anita" },{ $set:

< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

AFTER:

```
  {
    _id: ObjectId("64bbb43c64b3838090120774"),
    id: 8,
    last_name: 'Kulkarni',
    salary: 34000,
    skills: [
      ' ASP',
      'Analytics'
    ],
    department: 'IT',
    firstName: 'Anita'
  }
```

18. Add 'research' skill to all employees.

```
db.employee.updateMany({},{$push:{"skills":{$each:["research"]}}})
```

```
> db.employee.updateMany({},{$push:{"skills":{$each:["research"]}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 9,
    modifiedCount: 9,
    upsertedCount: 0
  }
office>
```

AFTER:

```
{
  _id: ObjectId("64bbb43c64b3838090120776"),
  id: 7,
  last_name: 'Mahale',
  salary: 15000,
  skills: [
    'Networking',
    'research'
  ],
  department: 'IT',
  firstName: 'Sumita'
}
```

19. Remove 'research' skill from all emplyees.

```
db.employee.updateMany({},{$pull:{"skills":{$each:["research"]}}})
```

```
> db.employee.updateMany({},{ $pull: { "skills": "research" } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 9,
    modifiedCount: 9,
    upsertedCount: 0
  }
office>
```

AFTER:

```
  {
    _id: ObjectId("64bbb43c64b3838090120776"),
    id: 7,
    last_name: 'Mahale',
    salary: 15000,
    skills: [
      'Networking'
    ],
    department: 'IT',
    firstName: 'Sumita'
  }
```

20. Update the employees skills from tax to GST.

```
db.employee.updateMany({ "skills": "Tax" },{ $set: { "skills.$": "GST" } })
```

```
> db.employee.updateMany({ "skills": "Tax" },{ $set: { "skills.$": "GST" } })

< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 2,
    modifiedCount: 2,
    upsertedCount: 0
  }
office>
```

AFTER:

```
{
  _id: ObjectId("64bbb43c64b3838090120 76f"),
  id: 2,
  last_name: 'Surve',
  salary: 3000,
  skills: [
    'Accounting',
    'GST'
  ],
  department: 'Finance',
  firstName: 'Sumit'
}
```

```
{
  _id: ObjectId("64bbb43c64b3838090120772"),
  id: 5,
  last_name: 'Shukla',
  salary: 20000,
  skills: [
    'Accounting',
    'GST'
  ],
  department: 'Finance',
  firstName: 'Kaushal'
}
```

Practical No 5: Updating and deleting operations in Mongo DB

- updateOne() and Updatemany() methods.
- $set and $unset operators.
- save method()
- deleteOne() and deleteMany() methods.
- remove method()

Change the first name of the first employee to Morgan.

```
db.employee.updateOne({}, { $set: { first_name: "Morgan" } })
```

```
{
    _id: ObjectId("64b9fbe0f1598476a0e90ad1"),
    id: 1,
    first_name: 'Morgan',
    last_name: 'Farhan',
    email: 'shaikhfarhan@gmail.com',
    salary: 100000500,
    skills: [
      'Java',
      'Python',
      'SQL'
    ],
    department: 'Software Development',
```

Change the department of employee having first name as Vishal to USA.

Before:

```
{
  _id: ObjectId("64b9ffd4c688dc0e7959dc3a"),
  id: 6423,
  first_name: 'Vishal',
  last_name: 'Yadav',
  email: 'vishalyadav@gmail.com',
  salary: 70000,
  skills: [
    'React',
    'Vue',
    'Angular'
  ],
  department: 'IT '
}
```

```
db.employee.updateOne({ firstName: "Vishal" },{ $set: { department: "USA" } })
```

```
> db.employee.updateOne({ firstName: "Vishal" },{ $set: { department: "USA" } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 0
  }
```

AFTER:

```
{
  _id: ObjectId("64b9ffd4c688dc0e7959dc3a"),
  id: 6423,
  first_name: 'Vishal',
  last_name: 'Yadav',
  email: 'vishalyadav@gmail.com',
  salary: 70000,
  skills: [
    'React',
    'Vue',
    'Angular'
  ],
  department: 'USA'
}
```

Increase the salary of the first employee by 500.

BEFORE:

```
> db.employee.find()
< {
    _id: ObjectId("64b9fbe0f1598476a0e90ad1"),
    id: 1,
    first_name: 'Shaikh',
    last_name: 'Farhan',
    email: 'shaikhfarhan@gmail.com',
    salary: 100000000,
    skills: [
      'Java',
      'Python',
      'SQL'
    ],
    department: 'Software Development',
    firstName: 'Morgan'
  }
```

```
db.employee.updateOne({}, { $inc: { salary: 500 } })
```

```
> db.employee.updateOne({}, { $inc: { salary: 500 } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

```
> db.employee.find()
< {
    _id: ObjectId("64b9fbe0f1598476a0e90ad1"),
    id: 1,
    first_name: 'Shaikh',
    last_name: 'Farhan',
    email: 'shaikhfarhan@gmail.com',
    salary: 100000500,
    skills: [
      'Java',
      'Python',
      'SQL'
    ],
    department: 'Software Development',
    firstName: 'Morgan'
  }
```

Increase the salary of an employee to 80000 having first name as PRANAY.

BEFORE:

```
{
  _id: ObjectId("64b9ffd4c688dc0e7959dc3b"),
  id: 6424,
  first_name: 'PRANAY',
  last_name: 'Sawant',
  email: 'Pranaysawant@gmail.com',
  salary: 70000,
  skills: [
    'React',
    'Angular',
    'TypeScript'
  ],
  department: 'CS'
}
```

```
db.employee.updateOne({ first_name: "PRANAY" },{ $set: { salary: 80000 } })
```

```
> db.employee.updateOne({ first_name: "PRANAY" },{ $set: { salary: 80000 } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
```

```
  {
    _id: ObjectId("64b9ffd4c688dc0e7959dc3b"),
    id: 6424,
    first_name: 'PRANAY',
    last_name: 'Sawant',
    email: 'Pranaysawant@gmail.com',
    salary: 80000,
```

Add the appraisal date to the current date for all employees and increase the salary of each by 5000($currentDate operator)

Before salary:

```
  _id: ObjectId("64b9ffd4c688dc0e7959dc37"),
  id: 6417,
  first_name: 'Sudesh',
  last_name: 'Rajbhar',
  email: 'sudeshrajbhar@gmail.com',
  salary: 45000,
  skills: [
    'Networking',
    'Linux',
    'Security'
  ],
  department: 'IT',
```

```
db.employee.updateMany({},{ $inc: { salary: 5000 } } ,{ $currentDate: { date: { $type: "timestamp" } } })
```

```
{
  _id: ObjectId("64b9ffd4c688dc0e7959dc37"),
  id: 6417,
  first_name: 'Sudesh',
  last_name: 'Rajbhar',
  email: 'sudeshrajbhar@gmail.com',
  salary: 50000,
  skills: [
    'Networking',
    'Linux',
    'Security'
  ],
  department: 'IT',
  date: Timestamp({ t: 1690513703, i: 6 })
}
{
  _id: ObjectId("64b9ffd4c688dc0e7959dc38"),
  id: 6425,
  first_name: 'Shaikh',
  last_name: 'Farhan',
  email: 'shaikhfarhan@gmail.com',
  salary: 100005000,
```

Example 2:

```
db.employee.updateMany({},{$currentDate: { appraisalDate: true }, $inc: { salary: 5000 } })
```

```
> db.employee.updateMany({},{$currentDate: { appraisalDate: true }, $inc: { salary: 5000 } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 12,
    modifiedCount: 12,
    upsertedCount: 0
  }
```

```
{
  _id: ObjectId("64b9ffd4c688dc0e7959dc37"),
  id: 6417,
  first_name: 'Sudesh',
  last_name: 'Rajbhar',
  salary: 55000,
  skills: [
    'Networking',
    'Linux',
    'Security'
  ],
  department: 'IT',
  date: Timestamp({ t: 1690513703, i: 6 }),
  appraisalDate: 2023-07-28T03:40:08.668Z
}
```

List an employee with the highest salary.

```
db.employee.aggregate([{$group: {_id:null,maxSalry:{ $max: "$salary" }}}])
```

```
> db.employee.aggregate([{$group: {_id:null,maxSalry:{ $max: "$salary" }}}])
< {
    _id: null,
    maxSalry: 100005500
  }
```

List an employee with the lowest salary and ($min operator).

```
db.employee.aggregate([{$group: {_id:null, minSalary:{ $min: "$salary" }}}])
```

```
> db.employee.aggregate([{$group: {_id:null, minSalary:{ $min: "$salary" }}}])
< {
    _id: null,
    minSalary: 5600
  }
```

Calculate the 20 times the salary of the first employee. ($mul operator).

```
db.employee.aggregate([{ $project: { _id: 1, multipliedSalary: { $multiply: ["$salary", 20] } } }])
```

```
> db.employee.aggregate([{ $project: { _id: 1, multipliedSalary: { $multiply: ["$salary", 20] } } }])
< {
    _id: ObjectId("64b9fbe0f1598476a0e90ad1"),
    multipliedSalary: 2000110000
  }
```

Remove the email id field from the collection. ($unset operator).

```
db.employee.updateMany({}, { $unset: { email: 1 } })
```

```
> db.employee.updateMany({}, { $unset: { email: 1 } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 12,
    modifiedCount: 12,
    upsertedCount: 0
  }
```

```
{
  _id: ObjectId("64b9ffd4c688dc0e7959dc37"),
  id: 6417,
  first_name: 'Sudesh',
  last_name: 'Rajbhar',
  salary: 50000,
  skills: [
    'Networking',
    'Linux',
    'Security'
  ],
  department: 'IT',
  date: Timestamp({ t: 1690513703, i: 6 })
}
```

Rename the field 'firstName' to 'FirstName'. ($rename operator).

```
db.employee.updateMany({}, { $rename: { "first_name": "FirstName" } })
```

```
> db.employee.updateMany({}, { $rename: { "first_name": "FirstName" } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 12,
    modifiedCount: 12,
    upsertedCount: 0
  }
```

```
{
  _id: ObjectId("64b9ffd4c688dc0e7959dc37"),
  id: 6417,
  last_name: 'Rajbhar',
  salary: 55000,
  skills: [
    'Networking',
    'Linux',
    'Security'
  ],
  department: 'IT',
  date: Timestamp({ t: 1690513703, i: 6 }),
  appraisalDate: 2023-07-28T03:40:08.668Z,
  FirstName: 'Sudesh'
}
```

Set the lastName of an employee 'Aarti' to 'Shetty' and salary to 90000.

(updating multiple fields).

```
db.employee.updateOne({ FirstName: "Sudesh" },{ $set: { last_name: "Shetty", salary: 90000 } })
```

```
> db.employee.updateOne({ FirstName: "Sudesh" },{ $set: { last_name: "Shetty", salary: 90000 } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

```
> db.employee.find({FirstName:"Sudesh"})
< {
    _id: ObjectId("64b9fbe0f1598476a0e90ad2"),
    id: 6417,
    last_name: 'Shetty',
    salary: 90000,
    skills: [
      'Networking',
      'Linux',
      'Security'
    ],
    department: 'IT Support',
    date: Timestamp({ t: 1690513703, i: 2 }),
    appraisalDate: 2023-07-28T03:40:08.668Z,
    FirstName: 'Sudesh'
  }
```

Update the skill of Sudesh to marketing and 'Accounting' instead of Networking

and Linux. (Updating arrays).

```
db.employee.updateOne({ FirstName: "Sudesh"},{ $set: { "skills.1": "MARKETING", "skills.2": "Accounting" }}, {$unset:{"skills.1": "Networking", "skills.2": "Linux"}})
```

Update the 'Marketing' skill to 'Advertising' for all documents.

BEFORE:


```
{
  _id: ObjectId("64b9ffd4c688dc0e7959dc37"),
  id: 6417,
  last_name: 'Rajbhar',
  salary: 55000,
  skills: [
    'Networking',
    'Linux',
    'Security',
    'Marketing'
  ],
  department: 'IT',
  date: Timestamp({ t: 1690513703, i: 6 }),
  appraisalDate: 2023-07-28T03:40:08.668Z,
  FirstName: 'Sudesh'
}
{
  _id: ObjectId("64b9ffd4c688dc0e7959dc38"),
  id: 6425,
  last_name: 'Farhan',
  salary: 100010000,
  skills: [
    'Java',
    'Python',
    'SQL',
    'Marketing'
  ],
```


```
db.employee.updateMany({ skills: "Marketing" },{ $set: { "skills.$": "Advertising" } })
```


```
> db.employee.updateMany({ skills: "Marketing" },{ $set: { "skills.$": "Advertising" } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 12,
    modifiedCount: 12,
    upsertedCount: 0
  }
```

AFTER:

```
< {
    skills: [
      'Java',
      'Python',
      'SQL',
      'Advertising'
    ]
  }
  {
    skills: [
      'Networking',
      'MARKETING',
      'Accounting',
      'Advertising'
    ]
  }
  {
    skills: [
      'Ruby',
      'Rails',
      'PostgreSQL',
      'Advertising'
    ]
  }
  {
    skills: [
```

Update the skills element 'GST' to 'Tax'.

BEFORE:

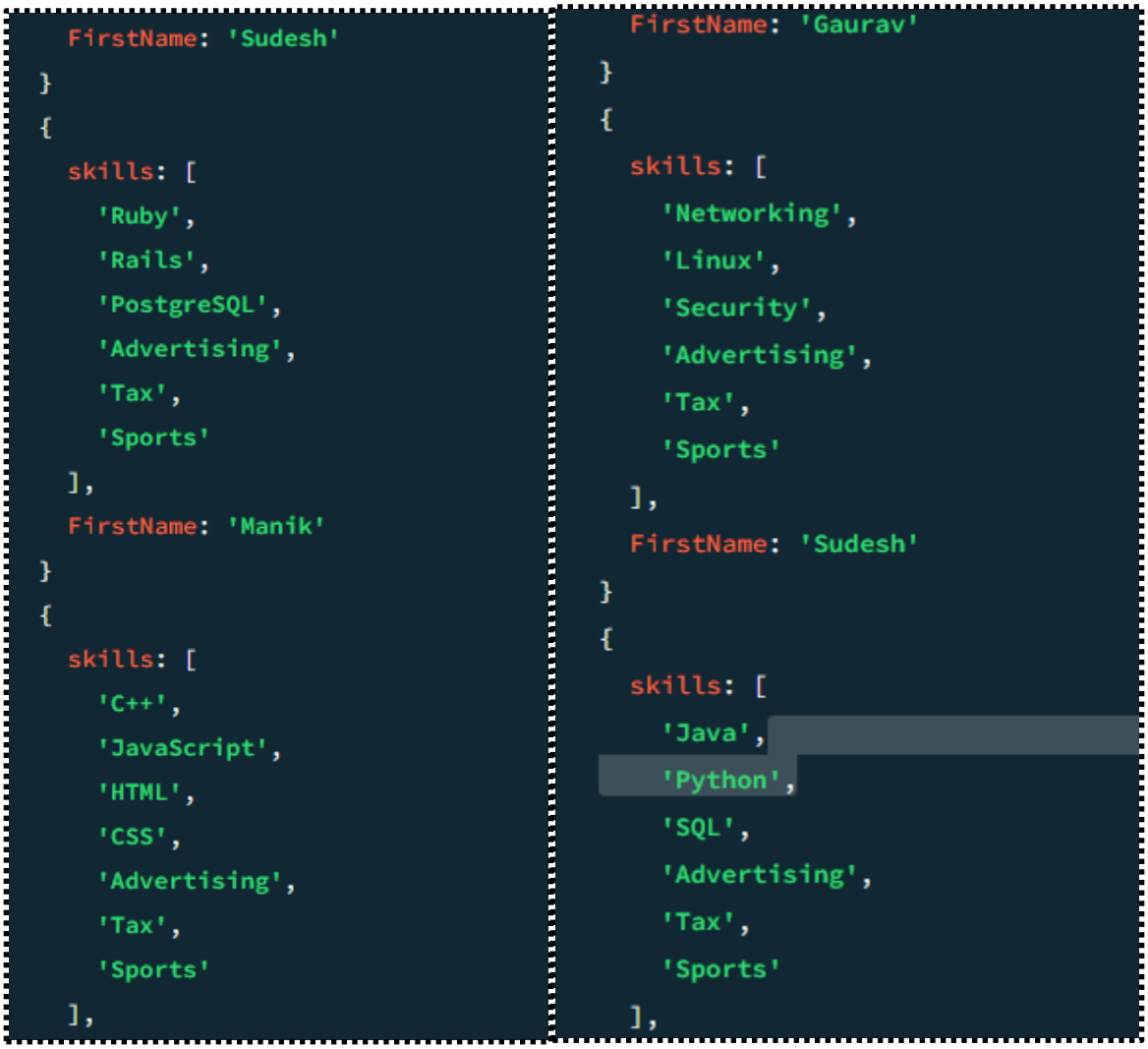
```
  FirstName: 'Sudesh'
}
{
  skills: [
    'Java',
    'Python',
    'SQL',
    'Advertising',
    'GST'
  ],
  FirstName: 'Shaikh'
}
{
  skills: [
    'React',
    'Vue',
    'Angular',
    'TypeScript',
    'Advertising',
    'GST'
  ],
```

```
db.employee.updateMany({ skills: "GST" },{ $set: { "skills.$": "Tax" } })
```

```
  FirstName: 'Sudesh'
}
{
  skills: [
    'Java',
    'Python',
    'SQL',
    'Advertising',
    'Tax'
  ],
  FirstName: 'Shaikh'
}
{
  skills: [
    'React',
    'Vue',
    'Angular',
    'TypeScript',
    'Advertising',
    'Tax'
  ],
```

Add 'sports' skill to all the existing documents.

```
db.employee.updateMany({}, { $push: { skills: "Sports" } })
```

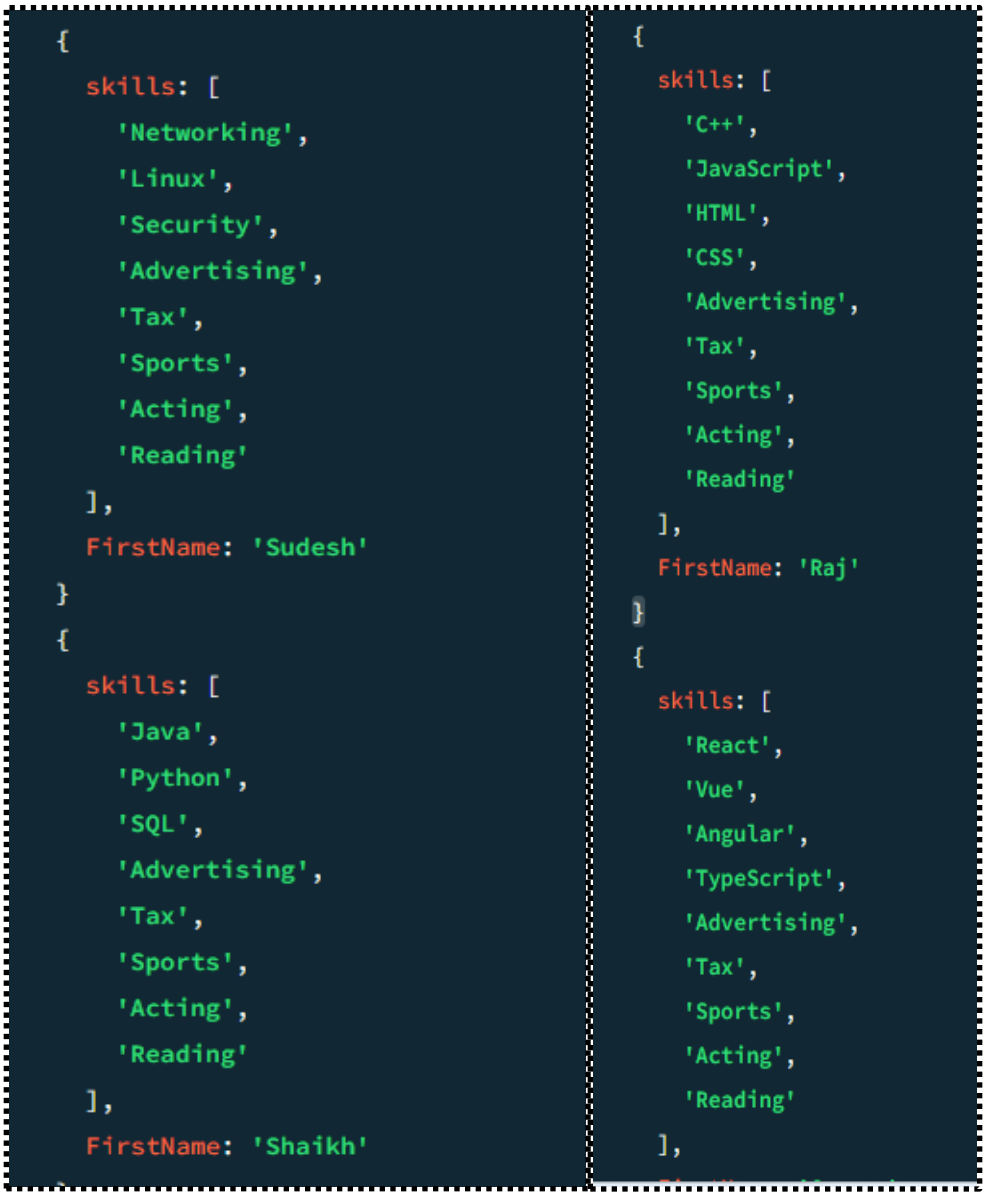


```
  FirstName: 'Sudesh'
}
{
  skills: [
    'Ruby',
    'Rails',
    'PostgreSQL',
    'Advertising',
    'Tax',
    'Sports'
  ],
  FirstName: 'Manik'
}
{
  skills: [
    'C++',
    'JavaScript',
    'HTML',
    'CSS',
    'Advertising',
    'Tax',
    'Sports'
  ],
  FirstName: 'Gaurav'
}
{
  skills: [
    'Networking',
    'Linux',
    'Security',
    'Advertising',
    'Tax',
    'Sports'
  ],
  FirstName: 'Sudesh'
}
{
  skills: [
    'Java',
    'Python',
    'SQL',
    'Advertising',
    'Tax',
    'Sports'
  ],
```

Add 'Acting' and 'Reading' skills to all the existing documents. ($push

and $each operators).

```
db.employee.updateMany({},{ $push: { skills: { $each: ["Acting", "Reading"] } } })
```

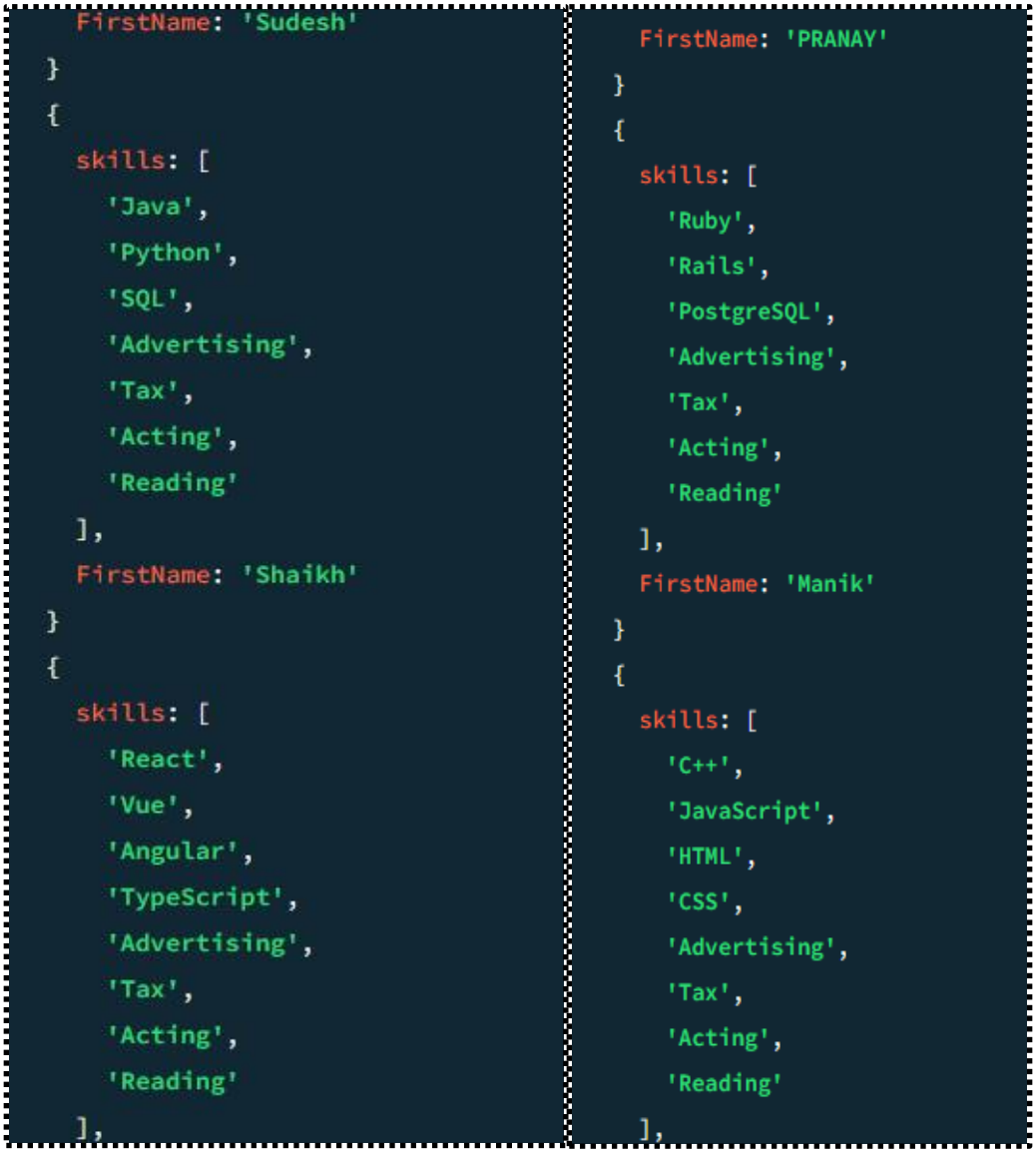


```
{
  skills: [
    'Networking',
    'Linux',
    'Security',
    'Advertising',
    'Tax',
    'Sports',
    'Acting',
    'Reading'
  ],
  FirstName: 'Sudesh'
}
{
  skills: [
    'Java',
    'Python',
    'SQL',
    'Advertising',
    'Tax',
    'Sports',
    'Acting',
    'Reading'
  ],
  FirstName: 'Shaikh'
```

```
{
  skills: [
    'C++',
    'JavaScript',
    'HTML',
    'CSS',
    'Advertising',
    'Tax',
    'Sports',
    'Acting',
    'Reading'
  ],
  FirstName: 'Raj'
}
{
  skills: [
    'React',
    'Vue',
    'Angular',
    'TypeScript',
    'Advertising',
    'Tax',
    'Sports',
    'Acting',
    'Reading'
  ],
```

Remove the 'sports' skill from all the documents.

```
db.employee.updateMany({}, { $pull: { skills: "Sports" } })
```



Replace document 1 with a new document using save() method.

```
db.employee.insertOne({newDocument})
```

```
const newDocument = {_id: 1,FirstName: "Sudeshh",last_name: "Rajbharr",skills: ["Sketching", "Driving"],salary: 100000}
```

```
> db.employee.insertOne({newDocument})
< {
    acknowledged: true,
    insertedId: ObjectId("64c348ae2e23d4c2f8c061dd")
  }
> const newDocument = {_id: 1,FirstName: "Sudeshh",last_name: "Rajbharr",skills: ["Sketching", "Driving"],salary: 100000}
```

```
{
  _id: ObjectId("64c348ae2e23d4c2f8c061dd"),
  newDocument: {
    _id: 1,
    FirstName: 'Sudeshh',
    last_name: 'Rajbharr',
    skills: [
      'Sketching',
      'Driving'
    ],
    salary: 100000
  }
}
```

Delete an employee having a salary of 55000.

```
> db.employee.find({ salary: 55000 }, { _id: 0, FirstName: 1 })
< {
    FirstName: 'Sudesh'
  }
```

```
db.employee.deleteOne({ salary: 55000 })
```

```
> db.employee.deleteOne({ salary: 55000 })
< {
    acknowledged: true,
    deletedCount: 1
  }
```

After deleting:

```
> db.employee.find({ salary: 55000 }, { _id: 0, FirstName: 1 })
<
office>
```

Delete all the employees having salaries greater than 70000.

Finding all employees with a salary greater than 70000.

```
> db.employee.find({ salary: { $gt: 70000 } }, { _id: 0, firstName: 1 })
< {
    firstName: 'Morgan'
  }
```

Delete all documents:

```
> db.employee.deleteMany({ salary: { $gt: 70000 } })
< {
    acknowledged: true,
    deletedCount: 7
  }
```

Document deleted:

```
> db.employee.find({ salary: { $gt: 70000 } }, { _id: 0, firstName: 1 })
<
office>
```

Remove the employee document having id 3 using deleteOne() method.

```
> db.employee.deleteOne({ id: 3 })
< {
    acknowledged: true,
    deletedCount: 1
  }
```

Practical 6 - Aggregation Operations in Mongo DB

a) aggregate() Method

Create the collection named "MyCollection" and insert the following documents and solve the following queries..

Given documents,

db.mycollection.insert([

{ title: 'Mongo DB Overview', description: 'Mongo DB is no sql database', by: 'tutorials point', url: 'http://www.rjcollege.com', tags: ['Mongo DB', 'database', 'NoSQL'], likes: 100 },

{ title: 'NoSQL Database', description: "NoSQL database doesn't have tables", by: 'tutorials point', url: 'http://www.rjcollege.com', tags: ['Mongo DB', 'database', 'NoSQL'], likes: 20, comments: [{ user:'user1', message: 'My first comment', dateCreated: new Date(2013,11,10,2,35), like: 0 }] }])

a) aggregate() Method

1. Write a Mongo DB query to use sum, avg, min and max expressions.

Example:

a. sum

db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])

```
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
< {
    _id: null,
    num_tutorial: 120
  }
office>
```

b. avg

db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])

```
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
< {
    _id: null,
    num_tutorial: 60
  }
office>
```

c. min

db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])

```
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
< {
    _id: null,
    num_tutorial: 20
  }
office>
```

d. max

db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])

```
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
< {
    _id: null,
    num_tutorial: 100
  }
office>
```

2. Write a Mongo DB query to use push and add ToSet expressions.

a. Push - Inserts the value to an array in the resulting document.

db.mycollection.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])

```
> db.mycollection.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
< {
    _id: null,
    url: [
      'http://www.rjcollege.com',
      'http://www.rjcollege.com'
    ]
  }
office>
```

b. addToSet - Inserts the value to an array in the resulting document but does not create duplicates.

b. addToSet - Inserts the value to an array in the resulting document but does not create duplicates.

db.mycollection.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])

```
> db.mycollection.aggregate([{$group : {_id : "$by_user", url : {$addToSet: "$url"}}}]
)
{ "_id" : null, "url" : [ "http://www.tutorialspoint.com" ] }
>
```

```
> db.mycollection.aggregate([
    {
      $group: {
        _id: "$by_user",
        url: { $push: "$url" }
      }
    }
  ]);
< {
    _id: 'Sby_user',
    url: [
      'Surl',
      'Surl'
    ]
  }
office>
```

3. Write a Mongo DB query to use the first and last expression.

a. First

db.mycollection.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])

```
> db.mycollection.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
< {
    _id: null,
    first_url: 'http://www.rjcollege.com'
  }
office>
```

b. Last

db.mycollection.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])

```
> db.mycollection.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])
< {
    _id: null,
    last_url: 'http://www.rjcollege.com'
  }
office>
```

Practical No 7: Sorting and Indexing.

- ❖ sort() Methods
- ❖ Metadata Sort Technique
- ❖ ensureIndex() and createIndex() Method

a. Sort() method

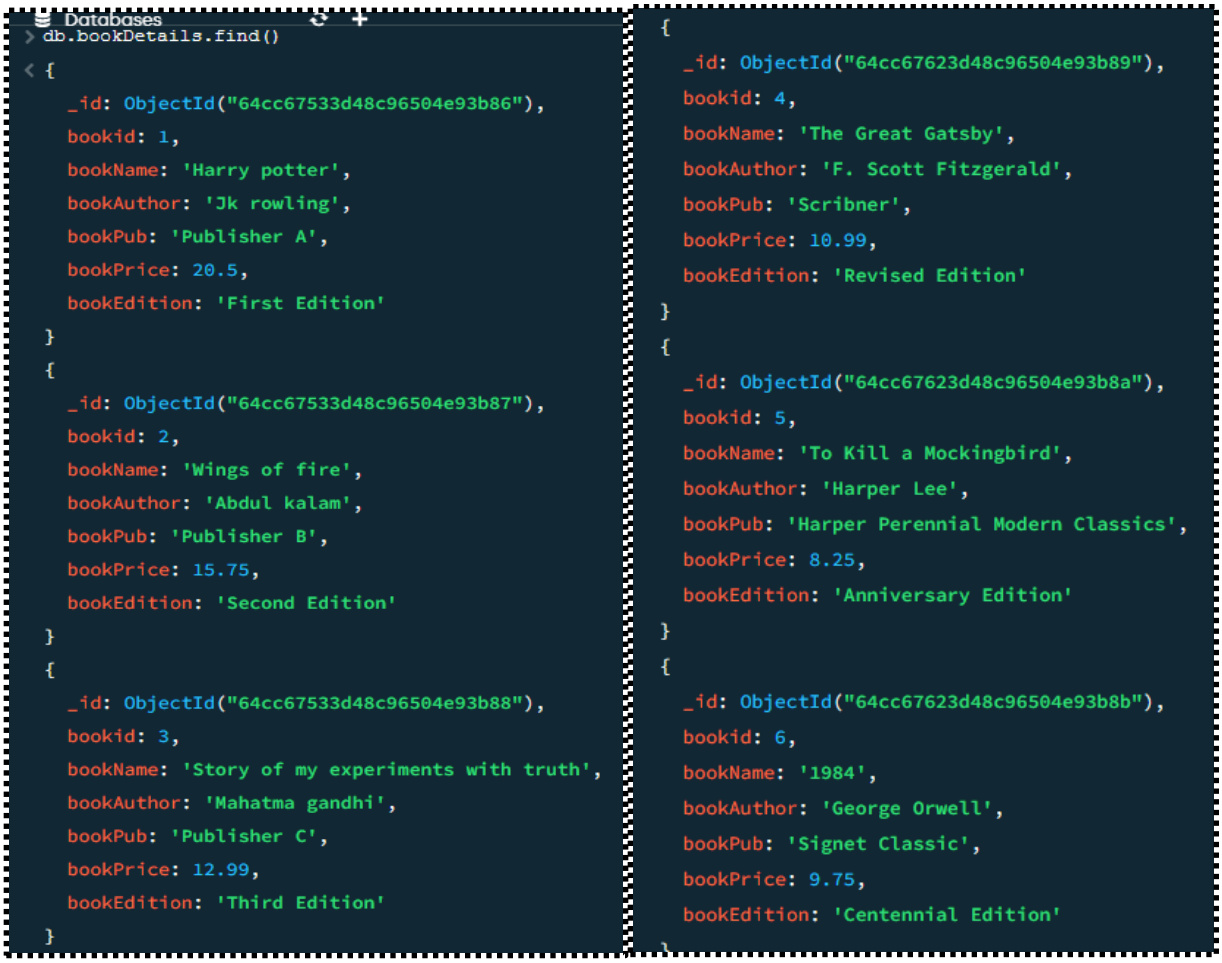
1. Create a collection named 'bookDetails' and insert some books ???? the fields like bookid, bookName, bookAuthor, bookPub, bookPrice and bookEdition.

```
db.bookDetails.insertMany([
 {
  bookid: 1,
  bookName: "Harry potter",
  bookAuthor: "Jk rowling",
  bookPub: "Publisher A",
  bookPrice: 20.50,
  bookEdition: "First Edition"
 },
 {
  bookid: 2,
  bookName: "Wings of fire",
  bookAuthor: "Abdul kalam",
  bookPub: "Publisher B",
  bookPrice: 15.75,
  bookEdition: "Second Edition"
 },
 {
  bookid: 3,
  bookName: "Story of my experiments with truth",
  bookAuthor: "Mahatma gandhi",
  bookPub: "Publisher C",
  bookPrice: 12.99,
  bookEdition: "Third Edition"
 }
{
```

```
  bookid: 4,
  bookName: "The Great Gatsby",
  bookAuthor: "F. Scott Fitzgerald",
  bookPub: "Scribner",
  bookPrice: 10.99,
  bookEdition: "Revised Edition"
 },
 {
  bookid: 5,
  bookName: "To Kill a Mockingbird",
  bookAuthor: "Harper Lee",
  bookPub: "Harper Perennial Modern Classics",
  bookPrice: 8.25,
  bookEdition: "Anniversary Edition"
 },
 {
  bookid: 6,
  bookName: "1984",
  bookAuthor: "George Orwell",
  bookPub: "Signet Classic",
  bookPrice: 9.75,
  bookEdition: "Centennial Edition"
 },
 {
  bookid: 7,
  bookName: "Pride and Prejudice",
  bookAuthor: "Jane Austen",
  bookPub: "Penguin Classics",
  bookPrice: 7.50,
  bookEdition: "Deluxe Edition"
 },
 {
  bookid: 8,
  bookName: "The Catcher in the Rye",
  bookAuthor: "J.D. Salinger",
  bookPub: "Little, Brown and Company",
  bookPrice: 12.00,
  bookEdition: "Collector's Edition"
 }
]);
```

```
> use book_6417
< switched to db book_6417
> db.createCollection("bookDetails")
< { ok: 1 }
> db.bookDetails.insertMany([
    {
      bookid: 1,
      bookName: "Harry potter",
      bookAuthor: "Jk rowling",
      bookPub: "Publisher A",
      bookPrice: 20.50,
      bookEdition: "First Edition"
    },
    {
```

```
> db.bookDetails.find({}, { _id: 0, bookName: 1 })
< {
    bookName: 'Harry potter'
  }
  {
    bookName: 'Wings of fire'
  }
  {
    bookName: 'Story of my experiments with truth'
  }
  {
    bookName: 'The Great Gatsby'
  }
  {
    bookName: 'To Kill a Mockingbird'
  }
  {
    bookName: '1984'
  }
  {
    bookName: 'Pride and Prejudice'
  }
  {
    bookName: 'The Catcher in the Rye'
  }
```



```
Databases
> db.bookDetails.find()
< {
    _id: ObjectId("64cc67533d48c96504e93b86"),
    bookid: 1,
    bookName: 'Harry potter',
    bookAuthor: 'Jk rowling',
    bookPub: 'Publisher A',
    bookPrice: 20.5,
    bookEdition: 'First Edition'
  }
  {
    _id: ObjectId("64cc67533d48c96504e93b87"),
    bookid: 2,
    bookName: 'Wings of fire',
    bookAuthor: 'Abdul kalam',
    bookPub: 'Publisher B',
    bookPrice: 15.75,
    bookEdition: 'Second Edition'
  }
  {
    _id: ObjectId("64cc67533d48c96504e93b88"),
    bookid: 3,
    bookName: 'Story of my experiments with truth',
    bookAuthor: 'Mahatma gandhi',
    bookPub: 'Publisher C',
    bookPrice: 12.99,
    bookEdition: 'Third Edition'
  }
  {
    _id: ObjectId("64cc67623d48c96504e93b89"),
    bookid: 4,
    bookName: 'The Great Gatsby',
    bookAuthor: 'F. Scott Fitzgerald',
    bookPub: 'Scribner',
    bookPrice: 10.99,
    bookEdition: 'Revised Edition'
  }
  {
    _id: ObjectId("64cc67623d48c96504e93b8a"),
    bookid: 5,
    bookName: 'To Kill a Mockingbird',
    bookAuthor: 'Harper Lee',
    bookPub: 'Harper Perennial Modern Classics',
    bookPrice: 8.25,
    bookEdition: 'Anniversary Edition'
  }
  {
    _id: ObjectId("64cc67623d48c96504e93b8b"),
    bookid: 6,
    bookName: '1984',
    bookAuthor: 'George Orwell',
    bookPub: 'Signet Classic',
    bookPrice: 9.75,
    bookEdition: 'Centennial Edition'
  }
```

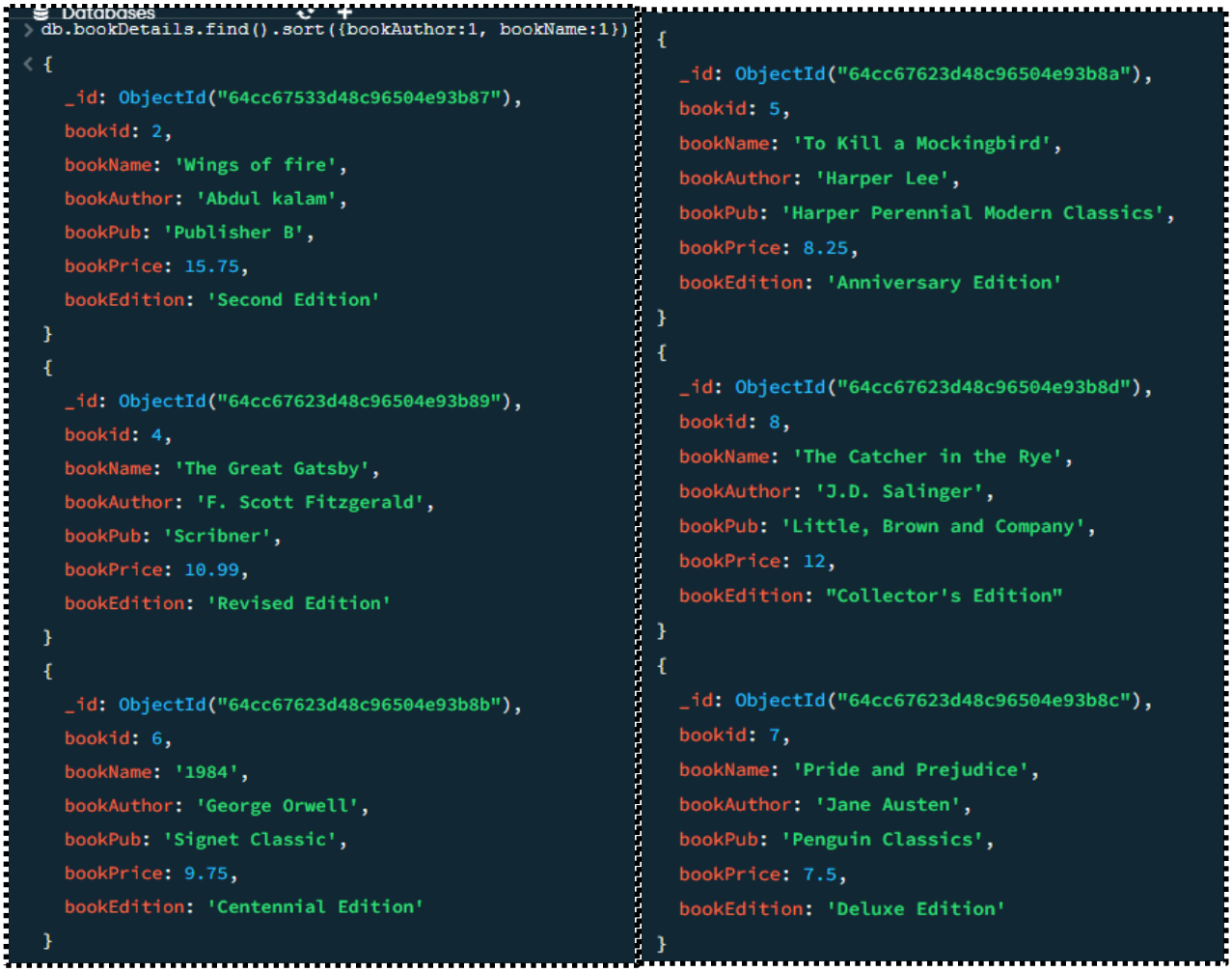
2. Sort the documents based on bookAuthor.

```
>db.BookDetails.find().sort({bookAuthor:1})
```

```
> db.bookDetails.find().sort({bookAuthor:1})
< {
    _id: ObjectId("64cc67533d48c96504e93b87"),
    bookid: 2,
    bookName: 'Wings of fire',
    bookAuthor: 'Abdul kalam',
    bookPub: 'Publisher B',
    bookPrice: 15.75,
    bookEdition: 'Second Edition'
  }
  {
    _id: ObjectId("64cc67623d48c96504e93b89"),
    bookid: 4,
    bookName: 'The Great Gatsby',
    bookAuthor: 'F. Scott Fitzgerald',
    bookPub: 'Scribner',
    bookPrice: 10.99,
    bookEdition: 'Revised Edition'
  }
{
  _id: ObjectId("64cc67623d48c96504e93b8b"),
  bookid: 6,
  bookName: '1984',
  bookAuthor: 'George Orwell',
  bookPub: 'Signet Classic',
  bookPrice: 9.75,
  bookEdition: 'Centennial Edition'
}
{
  _id: ObjectId("64cc67623d48c96504e93b8a"),
  bookid: 5,
  bookName: 'To Kill a Mockingbird',
  bookAuthor: 'Harper Lee',
  bookPub: 'Harper Perennial Modern Classics',
  bookPrice: 8.25,
  bookEdition: 'Anniversary Edition'
}
{
  _id: ObjectId("64cc67623d48c96504e93b8d"),
  bookid: 8,
  bookName: 'The Catcher in the Rye',
  bookAuthor: 'J.D. Salinger',
  bookPub: 'Little, Brown and Company',
  bookPrice: 12,
  bookEdition: "Collector's Edition"
}
```

3. Sort the documents on bookName and bookAuthor

```
db.BookDetails.find().sort({bookAuthor:1, bookName:1})
```

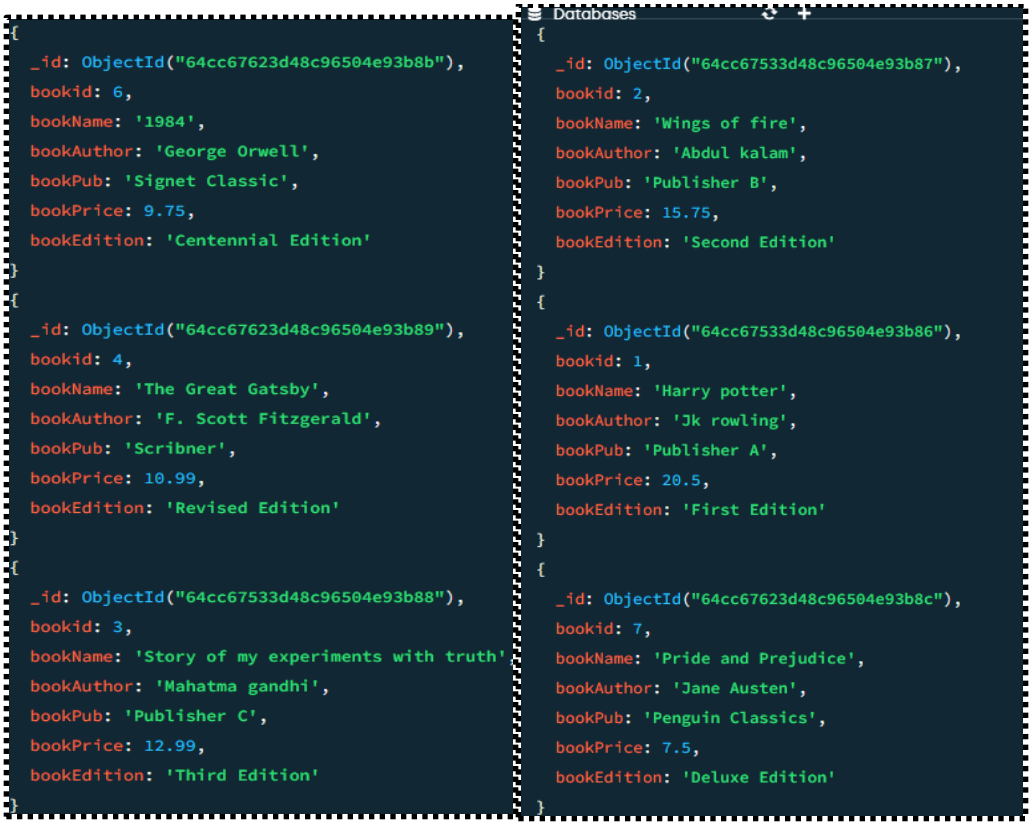


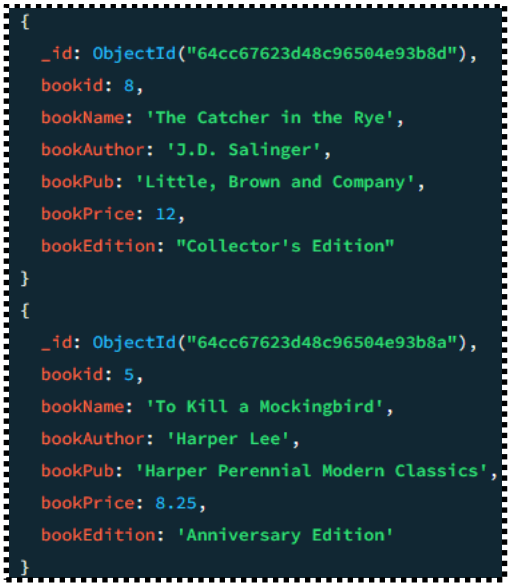
```
Databases
> db.bookDetails.find().sort({bookAuthor:1, bookName:1})
< {
    _id: ObjectId("64cc67533d48c96504e93b87"),
    bookid: 2,
    bookName: 'Wings of fire',
    bookAuthor: 'Abdul kalam',
    bookPub: 'Publisher B',
    bookPrice: 15.75,
    bookEdition: 'Second Edition'
  }
  {
    _id: ObjectId("64cc67623d48c96504e93b89"),
    bookid: 4,
    bookName: 'The Great Gatsby',
    bookAuthor: 'F. Scott Fitzgerald',
    bookPub: 'Scribner',
    bookPrice: 10.99,
    bookEdition: 'Revised Edition'
  }
  {
    _id: ObjectId("64cc67623d48c96504e93b8b"),
    bookid: 6,
    bookName: '1984',
    bookAuthor: 'George Orwell',
    bookPub: 'Signet Classic',
    bookPrice: 9.75,
    bookEdition: 'Centennial Edition'
  }
{
  _id: ObjectId("64cc67623d48c96504e93b8a"),
  bookid: 5,
  bookName: 'To Kill a Mockingbird',
  bookAuthor: 'Harper Lee',
  bookPub: 'Harper Perennial Modern Classics',
  bookPrice: 8.25,
  bookEdition: 'Anniversary Edition'
}
{
  _id: ObjectId("64cc67623d48c96504e93b8d"),
  bookid: 8,
  bookName: 'The Catcher in the Rye',
  bookAuthor: 'J.D. Salinger',
  bookPub: 'Little, Brown and Company',
  bookPrice: 12,
  bookEdition: "Collector's Edition"
}
{
  _id: ObjectId("64cc67623d48c96504e93b8c"),
  bookid: 7,
  bookName: 'Pride and Prejudice',
  bookAuthor: 'Jane Austen',
  bookPub: 'Penguin Classics',
  bookPrice: 7.5,
  bookEdition: 'Deluxe Edition'
}
```

```
{
  _id: ObjectId("64cc67533d48c96504e93b86"),
  bookid: 1,
  bookName: 'Harry potter',
  bookAuthor: 'Jk rowling',
  bookPub: 'Publisher A',
  bookPrice: 20.5,
  bookEdition: 'First Edition'
}
{
  _id: ObjectId("64cc67533d48c96504e93b88"),
  bookid: 3,
  bookName: 'Story of my experiments with truth',
  bookAuthor: 'Mahatma gandhi',
  bookPub: 'Publisher C',
  bookPrice: 12.99,
  bookEdition: 'Third Edition'
}
```

4. Sort the documents on bookPub in descending order.

```
db.bookDetails.find().sort({bookPub:-1})
```





b. Metadata Sort Technique:

5.Search the document for any one book name and sort the query result documents using the metadata 'bookPrice'.

Create index :

```
db.bookDetails.createIndex({ bookName: "text" })
```

```
> db.bookDetails.createIndex({ bookName: "text" })
< bookName_text
```

```
db.bookDetails.find(
  { $text: { $search: "Harry Potter" } },
  { score: { $meta: "textScore" }, _id: 0 }
)
```

```
> db.bookDetails.find(
    { $text: { $search: "Harry Potter" } },
    { score: { $meta: "textScore" }, _id: 0 }
  )
< {
    bookid: 1,
    bookName: 'Harry potter',
    bookAuthor: 'Jk rowling',
    bookPub: 'Publisher A',
    bookPrice: 20.5,
    bookEdition: 'First Edition',
    score: 1.5
  }
```

```
db.bookDetails.find(
  { $text: { $search: "Harry Potter" } },
  { score: { $meta: "textScore" }, _id: 0 }
).sort({sort_example:{$meta:"textScore"}})
```

```
{
  bookid: 1,
  bookName: 'Harry potter',
  bookAuthor: 'Jk rowling',
  bookPub: 'Publisher A',
  bookPrice: 20.5,
  bookEdition: 'First Edition',
  score: 1.5
}
ook_6417>
```

c)ensureIndex() and createIndex() Method

Create index :

```
db.bookDetails.createIndex({ bookName: "text" })
```

```
> db.bookDetails.createIndex({ bookName: "text" })
< bookName_text
```

ensureIndex():

```
db.bookDetails.ensureIndex({ bookName: "text" })
```

```
> db.bookDetails.ensureIndex({ bookName: "text" })
< [ 'bookName_text' ]
book_6417>
```

6) Check the existence of the index named bookPrice.

```
> db.bookDetails.exists({ bookName: "bookPrice" })
< {
    name: 'bookDetails',
    type: 'collection',
    options: {},
    info: {
      readOnly: false,
      uuid: new UUID("c23f0e04-ad1f-4672-a775-0defe31fb9d4")
    },
    idIndex: { v: 2, key: { _id: 1 }, name: '_id_' }
  }
book_6417>
```

7) Create an index named bookPriceList on the bookName in ascending Order and bookPrice in descending order .Example:

```
db.books.createIndex({ bookName: 1, bookPrice: -1 }, { name: "bookPriceList" })
```

```
> db.books.createIndex({ bookName: 1, bookPrice: -1 }, { name: "bookPriceList" })
< bookPriceList
book_6417>
```

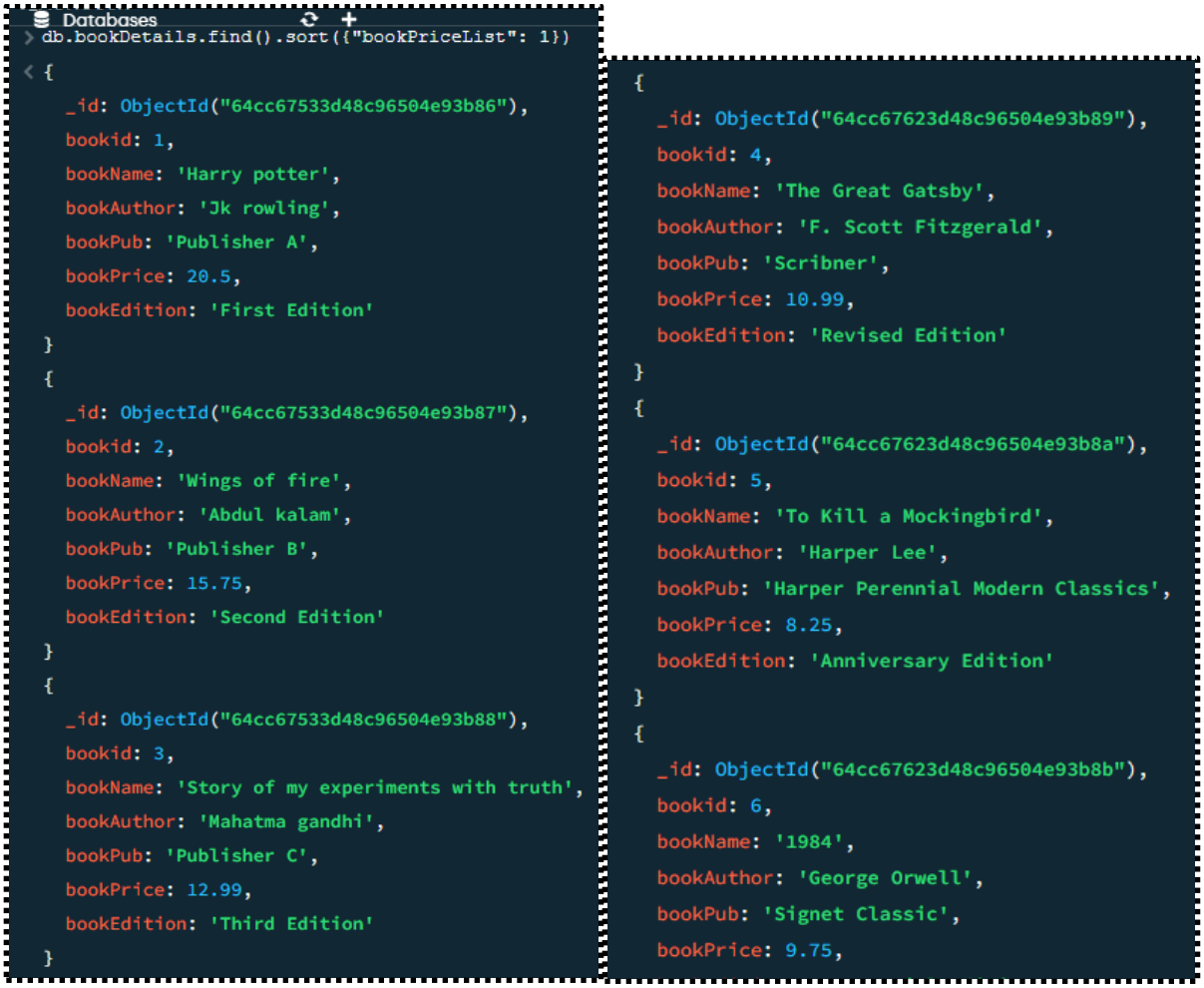
```
> db.books.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { bookName: 1, bookPrice: -1 }, name: 'bookPriceList' }
  ]
```

```
> db.bookDetails.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    {
      v: 2,
      key: { _fts: 'text', _ftsx: 1 },
      name: 'bookName_text',
      weights: { bookName: 1 },
      default_language: 'english',
      language_override: 'language',
      textIndexVersion: 3
    },
    { v: 2, key: { bookName: 1, bookPrice: -1 }, name: 'bookPriceList' }
  ]
```

8.Get the list of books sorted by 'bookPriceList' Index

```
db.bookDetails.find().sort({"bookPriceList": 1})
```

```
> db.books.find().sort({ bookName: 1, bookPrice: -1 })
<
> db.books.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { bookName: 1, bookPrice: -1 }, name: 'bookPriceList' }
  ]
```



```
Databases
> db.bookDetails.find().sort({"bookPriceList": 1})
< {
    _id: ObjectId("64cc67533d48c96504e93b86"),
    bookid: 1,
    bookName: 'Harry potter',
    bookAuthor: 'Jk rowling',
    bookPub: 'Publisher A',
    bookPrice: 20.5,
    bookEdition: 'First Edition'
  }
  {
    _id: ObjectId("64cc67533d48c96504e93b87"),
    bookid: 2,
    bookName: 'Wings of fire',
    bookAuthor: 'Abdul kalam',
    bookPub: 'Publisher B',
    bookPrice: 15.75,
    bookEdition: 'Second Edition'
  }
  {
    _id: ObjectId("64cc67533d48c96504e93b88"),
    bookid: 3,
    bookName: 'Story of my experiments with truth',
    bookAuthor: 'Mahatma gandhi',
    bookPub: 'Publisher C',
    bookPrice: 12.99,
    bookEdition: 'Third Edition'
  }
  {
    _id: ObjectId("64cc67623d48c96504e93b89"),
    bookid: 4,
    bookName: 'The Great Gatsby',
    bookAuthor: 'F. Scott Fitzgerald',
    bookPub: 'Scribner',
    bookPrice: 10.99,
    bookEdition: 'Revised Edition'
  }
  {
    _id: ObjectId("64cc67623d48c96504e93b8a"),
    bookid: 5,
    bookName: 'To Kill a Mockingbird',
    bookAuthor: 'Harper Lee',
    bookPub: 'Harper Perennial Modern Classics',
    bookPrice: 8.25,
    bookEdition: 'Anniversary Edition'
  }
  {
    _id: ObjectId("64cc67623d48c96504e93b8b"),
    bookid: 6,
    bookName: '1984',
    bookAuthor: 'George Orwell',
    bookPub: 'Signet Classic',
    bookPrice: 9.75,
```

9.Insert a document by creating a document object Bookobj in collection bookDetails

```
bookObjects = [
  {
    bookid: 10,
    bookName: "Another Book",
    bookAuthor: "Author Z",
    bookPub: "Publisher W",
    bookPrice: 19.99,
    bookEdition: "Second Edition"
  },
];
```

```
> db.bookDetails.insertMany(bookObjects);
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("64cc7af23d48c96504e93b8f")
    }
  }
book_6417>
```

Practical no 8. Backup, restore and Sharding in Mongo DB.

- ❖ Write a Mongo DB query to create a Backup of the Existing databases.
- ❖ Write a Mongo DB query to create restore the database from the backup.

1. Write a Mongo DB query to create a backup of existing databases.

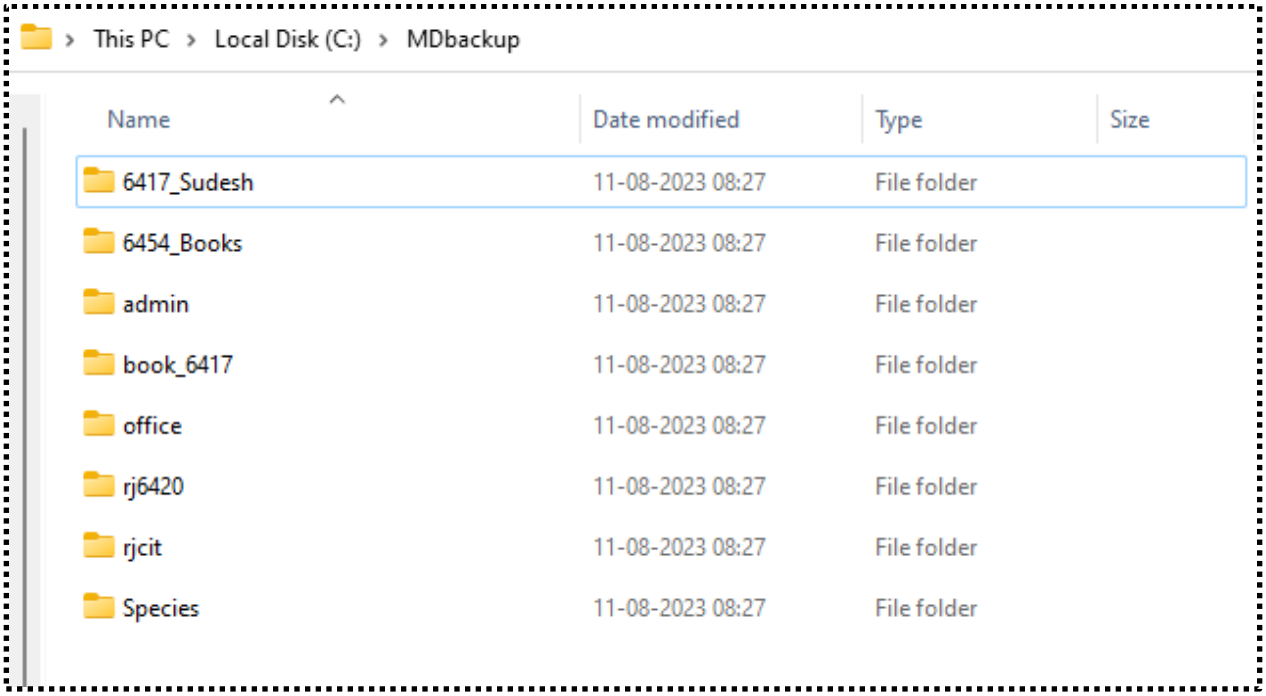
- Find Mongodump.exe and copy the path:

```
C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-windows-x86_64-100.7.3\bin>
```

- After that create a folder MD Backup and execute the command


```
C:\Program Files\Mongo DB\Server\6.0\bin\Mongo DB-database-tools-windows-x86_64-100.7.3\bin>mongodump.exe --out C:\MDbackup
```

```
C:\Program Files\MongoDB\Server\6.0\bin\mongodb-database-tools-windows-x86_64-100.7.3\bin>mongodump.exe --out C:\MDbackup
2023-08-11T08:27:52.099+0530    writing admin.system.version to C:\MDbackup\admin\system.version.bson
2023-08-11T08:27:52.122+0530    done dumping admin.system.version (1 document)
2023-08-11T08:27:52.123+0530    writing Species.Flower to C:\MDbackup\Species\Flower.bson
2023-08-11T08:27:52.123+0530    writing book_6417.bookDetails to C:\MDbackup\book_6417\bookDetails.bson
2023-08-11T08:27:52.124+0530    writing rj6420.student to C:\MDbackup\rj6420\student.bson
2023-08-11T08:27:52.124+0530    writing 6454_Books.bookdetails to C:\MDbackup\6454_Books\bookdetails.bson
2023-08-11T08:27:52.159+0530    done dumping rj6420.student (6 documents)
2023-08-11T08:27:52.169+0530    done dumping Species.Flower (150 documents)
2023-08-11T08:27:52.183+0530    done dumping 6454_Books.bookdetails (11 documents)
2023-08-11T08:27:52.183+0530    writing 6417_Sudesh.students to C:\MDbackup\6417_Sudesh\students.bson
2023-08-11T08:27:52.198+0530    done dumping book_6417.bookDetails (10 documents)
2023-08-11T08:27:52.198+0530    writing office.employee to C:\MDbackup\office\employee.bson
2023-08-11T08:27:52.198+0530    writing office.mycollection to C:\MDbackup\office\mycollection.bson
2023-08-11T08:27:52.198+0530    writing 6417_Sudesh.Student to C:\MDbackup\6417_Sudesh\Student.bson
2023-08-11T08:27:52.199+0530    done dumping 6417_Sudesh.students (4 documents)
2023-08-11T08:27:52.213+0530    done dumping office.employee (4 documents)
2023-08-11T08:27:52.213+0530    writing 6417_Sudesh.MyCollection_6417 to C:\MDbackup\6417_Sudesh\MyCollection_6417.bson
2023-08-11T08:27:52.228+0530    done dumping office.mycollection (2 documents)
2023-08-11T08:27:52.228+0530    writing book_6417.books to C:\MDbackup\book_6417\books.bson
2023-08-11T08:27:52.228+0530    done dumping 6417_Sudesh.Student (1 document)
2023-08-11T08:27:52.228+0530    writing office.MyCollection_6417 to C:\MDbackup\office\MyCollection_6417.bson
2023-08-11T08:27:52.229+0530    writing rjcit.products to C:\MDbackup\rjcit\products.bson
2023-08-11T08:27:52.229+0530    done dumping 6417_Sudesh.MyCollection_6417 (1 document)
2023-08-11T08:27:52.241+0530    done dumping book_6417.books (0 documents)
2023-08-11T08:27:52.249+0530    done dumping office.MyCollection_6417 (0 documents)
2023-08-11T08:27:52.262+0530    done dumping rjcit.products (0 documents)

C:\Program Files\MongoDB\Server\6.0\bin\mongodb-database-tools-windows-x86_64-100.7.3\bin>
```


This PC › Local Disk (C:) › MDbackup
Name
Date modified
Type
Size
6417_Sudesh 11-08-2023 08:27 File folder
6454_Books 11-08-2023 08:27 File folder
admin 11-08-2023 08:27 File folder
book_6417 11-08-2023 08:27 File folder
office 11-08-2023 08:27 File folder
rj6420 11-08-2023 08:27 File folder
rjcit 11-08-2023 08:27 File folder
Species 11-08-2023 08:27 File folder

2. Write a Mongo DB query to restore the database from backup.

Creating a database to delete:

```
> use restore_db
< switched to db restore_db
> db.createCollection("restore")
< { ok: 1 }
> db.restore.insertOne{"Name":"Sudesh"}
```

```
> db.restore.insertOne({Name:"Sudesh"})
< {
    acknowledged: true,
    insertedId: ObjectId("64e98e0e7b013c2f24c8fcf3")
  }
restore_db>
```

Create backup for the database:

```
C:\Program Files\MongoDB\Server\5.0\bin\mongodb-database-tools-windows-x86_64-100.5.4\bin>mongodump.exe --out D:\MDbackup
2023-08-26T11:43:34.653+0530    writing admin.system.version to D:\MDbackup\admin\system.version.bson
2023-08-26T11:43:34.686+0530    done dumping admin.system.version (1 document)
2023-08-26T11:43:34.686+0530    writing MyDB_6423.student_6423 to D:\MDbackup\MyDB_6423\student_6423.bson
2023-08-26T11:43:34.687+0530    writing species.flower to D:\MDbackup\species\flower.bson
2023-08-26T11:43:34.687+0530    writing College.Student to D:\MDbackup\College\Student.bson
2023-08-26T11:43:34.688+0530    writing species_6423.flower_6423 to D:\MDbackup\species_6423\flower_6423.bson
2023-08-26T11:43:34.702+0530    done dumping MyDB_6423.student_6423 (16 documents)
2023-08-26T11:43:34.744+0530    done dumping College.Student (13 documents)
2023-08-26T11:43:34.744+0530    writing gaurav_6463.emp_data to D:\MDbackup\gaurav_6463\emp_data.bson
2023-08-26T11:43:34.767+0530    done dumping species.flower (150 documents)
2023-08-26T11:43:34.767+0530    writing College.student to D:\MDbackup\College\student.bson
```

My database In backup:

```
2023-08-26T11:43:35.036+0530    writing restore_db.restore to D:\MDbackup\restore_db\restore.bson
2023-08-26T11:43:35.037+0530    writing MY_Exp_Collection.MY_Exp_Collection to D:\MDbackup\MY_Exp_Collection\MY_Exp_Collection.bson
2023-08-26T11:43:35.041+0530    done dumping restore_db.restore (1 document)
```

Now run the Mongorestore command:

```
C:\Program Files\MongoDB\Server\5.0\bin\mongodb-database-tools-windows-x86_64-100.5.4\bin>mongorestore --db restore_db D:\MDbackup\restore_db
2023-08-26T11:43:44.752+0530    The --db and --collection flags are deprecated for this use-case; please use --nsInclude instead, i.e. with --nsInclu
COLLECTION}
2023-08-26T11:43:44.754+0530    building a list of collections to restore from D:\MDbackup\restore_db dir
2023-08-26T11:43:44.754+0530    reading metadata for restore_db.restore from D:\MDbackup\restore_db\restore.metadata.json
```

C:\Program Files\Mongo DB\Server\5.0\bin\Mongo DB-database-tools-windows-x86_64-100.5.4\bin>mongorestore --db restore_db D:\MDbackup\restore_db

```
2023-08-26T11:46:24.123+0530    1 document(s) restored successfully. 0 document(s) failed to restore.
```

Delete the Database:

```
> db.dropDatabase()
< { ok: 1, dropped: 'restore_db' }
restore_db>
```

```
> db.dropDatabase()
< { ok: 1, dropped: 'bsccs' }
```

```
> show collections
< fy
  restore
restore_db>
```

```
> use restore_db
< switched to db restore_db
> show collections
< fy
  restore
> db.restore.find()
< {
    _id: ObjectId("64e99ac07b013c2f24c8fcf4"),
    Name: 'Sudesh'
  }
restore_db>
```

Practical No 9. Using Mongo DB with different programming languages

- ❖ Using Mongo DB with python
- ❖ Using Mongo DB with java

a. Python with Mongo DB

2. Inserting a Document.

Run this in VSCode terminal:

```
pip install pymongo
```

```
Collecting pymongo
  Downloading pymongo-4.4.1-cp310-cp310-win_amd64.whl (408 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 408.2/408.2 kB 669.9 kB/s eta 0:00:00
Collecting dnspython<3.0.0,>=1.16.0
  Downloading dnspython-2.4.2-py3-none-any.whl (300 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 300.4/300.4 kB 516.2 kB/s eta 0:00:00
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.4.2 pymongo-4.4.1

[notice] A new release of pip available: 22.2.2 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```python
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client.employees
col = db.employees
def insert_one():
    try:
        name1 = input("Enter your name: ")
        age1 = int(input("Enter your age: "))
        db.col.insert_one({
            "name": name1,
            "age": age1
        })
        print("Data inserted successfully")
    except Exception as e:
        print(str(e))
insert_one()
```

```
PS C:\TurboC++\Disk\TurboC3\BIN> & C:/Users/Admin/AppData/Local/Programs/Python/Python310/python.exe c:/TurboC++/Disk/TurboC3/BIN/6417_MDinsert.py
```

```
PS C:\TurboC++\Disk\TurboC3\BIN> & C
sert.py
Enter your name: Sudesh Rajbhar
Enter your age: 20
Data inserted successfully
PS C:\TurboC++\Disk\TurboC3\BIN>
```

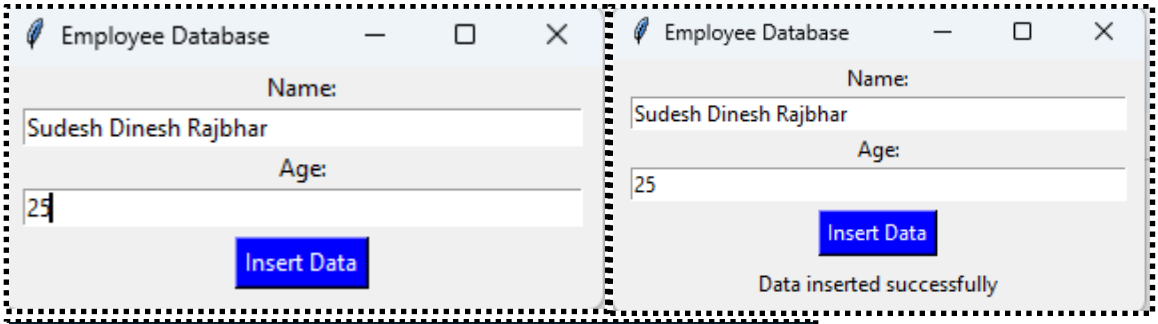
Inserting through Tkinter GUI:

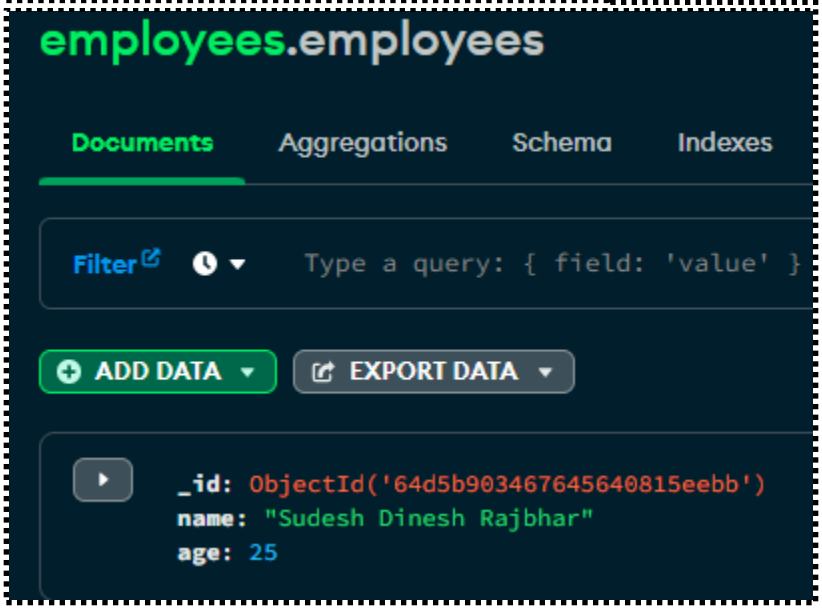
Code:

```
import tkinter as tk
from pymongo import MongoClient

client = MongoClient('localhost', 27017)
db = client.employees
col = db.employees

def insert_one():
    try:
        name1 = name_entry.get()
        age1 = int(age_entry.get())

        col.insert_one({
            "name": name1,
            "age": age1
        })

        result_label.config(text="Data inserted successfully")

    except Exception as e:
        result_label.config(text=str(e))

root = tk.Tk()
root.title("Employee Database")
root.geometry("300x100")

name_label = tk.Label(root, text="Name:")
name_label.pack()

name_entry = tk.Entry(root)
name_entry.pack(fill=tk.X, padx=10)

age_label = tk.Label(root, text="Age:")
age_label.pack()

age_entry = tk.Entry(root)
age_entry.pack(fill=tk.X, padx=10)

insert_button = tk.Button(root, text="Insert Data", command=insert_one,
bg="blue", fg="white")
```

```
insert_button.pack(pady=5)

result_label = tk.Label(root, text="")
result_label.pack()

root.mainloop()
```

Window:





2.Updating a database.

```
from pymongo import MongoClient
client = MongoClient('localhost',27017)
db = client.employees
col = db.employees
def update_one():
    try:
        name1 = input("Enter your name: ")
        age1 = int(input("Enter your age: "))
        db.col.update_one({"name": name1},{"$set":{"age":age1}})
        print("Records Updated Successfully")
    except Exception as e:
        print(str(e))
update_one()
```

```
PS C:\TurboC++\Disk\TurboC3\BIN> & C:/Users/Admin/AppData/Local/Programs/Python/Python310/python.exe c:/TurboC++/Disk/TurboC3/BIN/6417_MDupdate.py
Enter your name: Sudesha
Enter your age: 21
Records Updated Successfully
PS C:\TurboC++\Disk\TurboC3\BIN>
```

Before:

After:

3.delete the document.

```
from pymongo import MongoClient
client = MongoClient('localhost',27017)
db = client.employees
col = db.employees
def delete_one():
    try:
        name1 = input("Enter your name: ")
        db.col.delete_one({"name": name1})
        print("Data Deleted Successfully")
    except Exception as e:
        print(str(e))
delete_one()
```

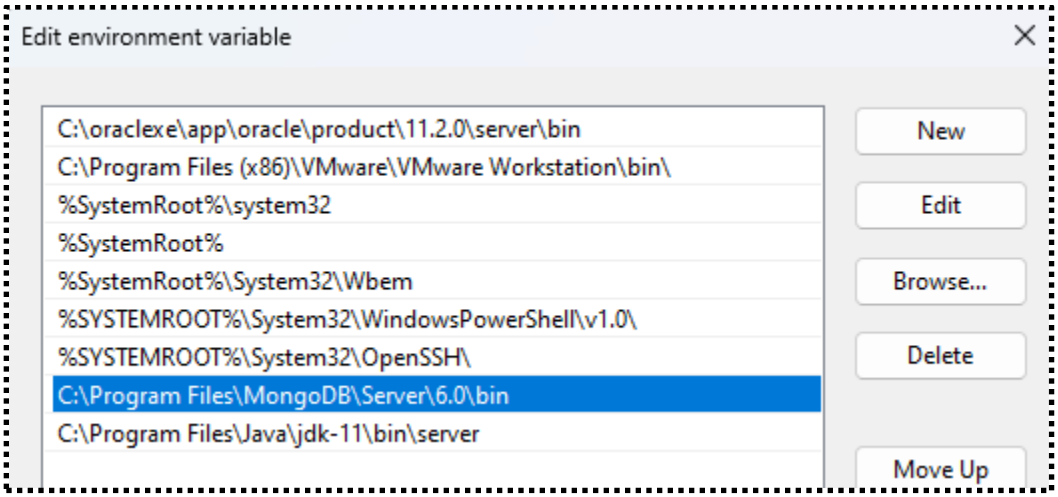
```
PS C:\TurboC++\Disk\TurboC3\BIN> & C:/Users/A
Enter your name: Sudesh Rajbhar
Data Deleted Successfully
PS C:\TurboC++\Disk\TurboC3\BIN>
```

```
> db.col.find()
<
employees>
```

:No documents found.

b. Java with Mongo DB.

Step 1:Set the Mongo DB server path

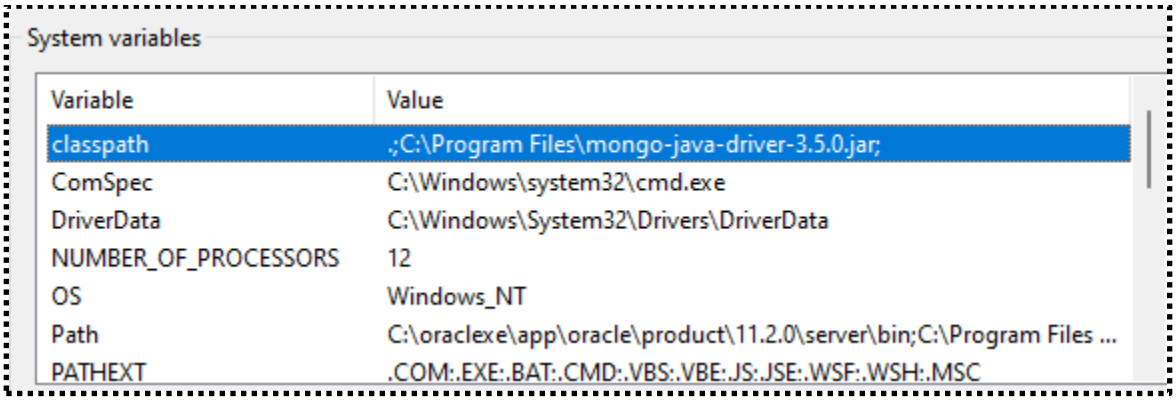


Step 2:

Go to environmental variable and come under the system variable ,click on "new"

Set variable name = classpath

Set variable value = .;directory.jar; example: (.;C:\myjar.jar;)



Step 3: Write the code and open the Mongo DB shell

My connection String:Mongo DB://localhost:27017

INSERTING:

Source Code:

```
import com.Mongo DB.client.MongoCollection;
import com.Mongo DB.client.MongoDatabase;
import com.Mongo DB.MongoClient;
import com.Mongo DB.MongoCredential;
import org.bson.Document;

public class Inserting {
    public static void main(String args[]) {
        MongoClient mongo = new MongoClient("localhost", 27017); // Creating a
Mongo client
        System.out.println("Connected to the database successfully");
        MongoDatabase database = mongo.getDatabase("Sudesh_6417"); // Accessing
the database
        MongoCollection<Document> collection =
database.getCollection("Student"); // accessing collection
        System.out.println("Collection sampleCollection selected
successfully");
        Document document = new Document();
        document.append("id", 6417);
        document.append("Name", "Sudesh");
        document.append("Class", "TYBSC IT");
        collection.insertOne(document);
        System.out.println("Document inserted successfully");
    }
}
```

Output:

```
javac .\Inserting.java
java Inserting
```

UPDATING:

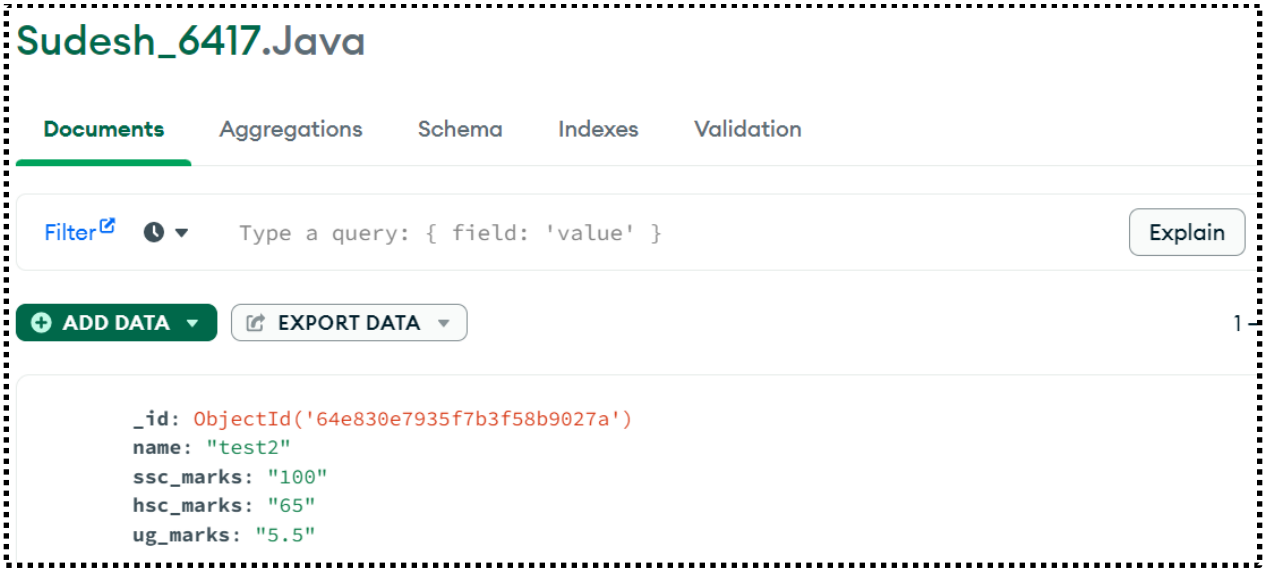
```
import java.net.UnknownHostException;

import com.Mongo DB.BasicDBObject;
import com.Mongo DB.DB;
import com.Mongo DB.DBCollection;
import com.Mongo DB.DBObject;
import com.Mongo DB.MongoClient;
import com.Mongo DB.WriteResult;

public class Updating {
    public static void main(String[] args) throws UnknownHostException {
        MongoClient mongo = new MongoClient("localhost", 27017);
        DB db = mongo.getDB("Sudesh_6417");
        DBCollection col = db.getCollection("Java");
        // Update single field in a single document
        DBObject query = new BasicDBObject("name", "Sudesh Rajbhar");
        DBObject update = new BasicDBObject();
        update.put("$set", new BasicDBObject("ssc_marks", "100"));
        WriteResult result = col.update(query, update);
        System.out.println("Document Updated ...");
        mongo.close();
    }
}
```

Output:

```
Aug 25, 2023 10:19:13 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:139}] to localhost:27017
Document Updated ...
```

```
mongoDb with Java> javac .\Updating.java
uses or overrides a deprecated API.
Xlint:deprecation for details.
mongoDb with Java> java Updating
```

SSC MARKS Changed:



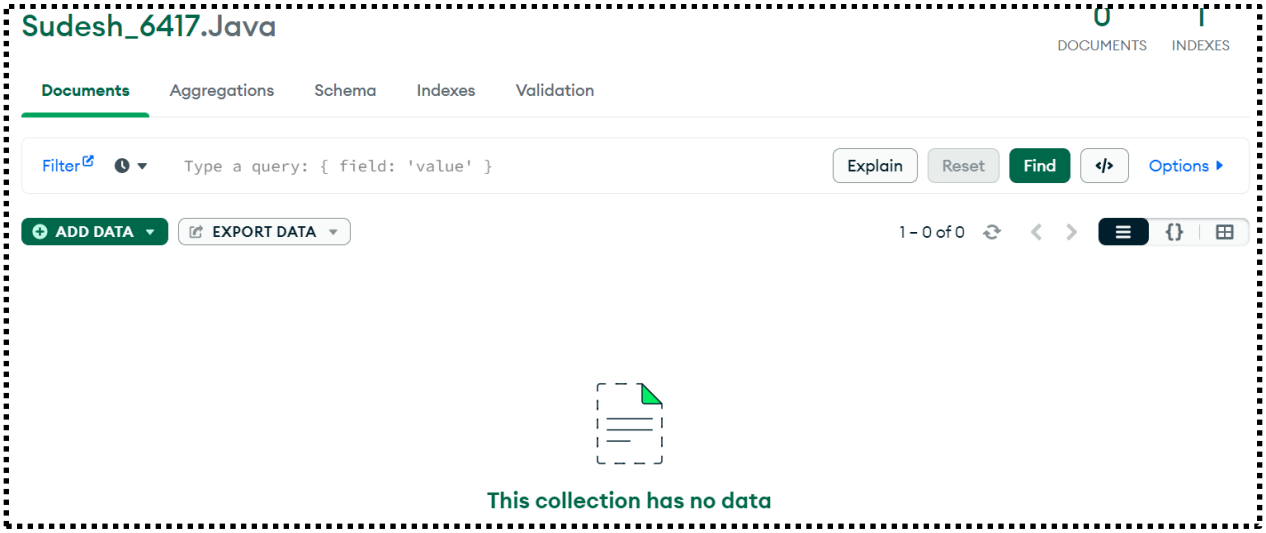
DELETING

Source Code:

```
import com.Mongo DB.client.MongoCollection;
import com.Mongo DB.client.MongoDatabase;
import com.Mongo DB.MongoClient;
import com.Mongo DB.MongoCredential;
import org.bson.Document;
import com.Mongo DB.client.model.Filters; //it store filter property

public class Deleting {
    public static void main(String args[]) {
        MongoClient mongo = new MongoClient("localhost", 27017);// Creating a
Mongo client
        System.out.println("Connected to the database successfully");
        MongoDatabase database = mongo.getDatabase("Sudesh_6417"); // Accessing
the database
        MongoCollection<Document> collection = database.getCollection("Java");
        System.out.println("Collection sampleCollection selected
successfully");

        // Deleting the documents
        collection.deleteOne(Filters.eq("name", "test2"));
        System.out.println("Document deleted successfully...");
    }
}
```

Output:

```
mongoDb with Java> javac .\Deleting.java
mongoDb with Java> java Deleting
Aug 25, 2023 10:23:55 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:141}] to localhost:27017
Document deleted successfully...
```


Sudesh_6417.Java
0
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' }
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 0 of 0
This collection has no data

SELECTION:

Source Code:

```
import com.Mongo DB.client.MongoCollection;
import com.Mongo DB.client.MongoDatabase;
import com.Mongo DB.MongoClient;
import com.Mongo DB.MongoCredential;
import org.bson.Document;
import java.util.Iterator;
import com.Mongo DB.client.FindIterable;

public class Select {
    public static void main(String args[]) {
        MongoClient mongo = new MongoClient("localhost", 27017);
        System.out.println("Connected to the database successfully");
        MongoDatabase database = mongo.getDatabase("6417_Sudesh");
        MongoCollection<Document> collection = database.getCollection("Java");
        System.out.println("Collection sampleCollection selected
successfully");
        FindIterable<Document> iterDoc = collection.find(); // Getting the
iterable object
        int i = 1;
        Iterator it = iterDoc.iterator(); // Getting the iterator
        while (it.hasNext()) {
            System.out.println(it.next());
            i++;
        }
    }
}
```

```
_id: ObjectId('64e834dabfd59ac80831da97')
Name: "Sudesh"
```

```
.main(Select.java:11)
\NGT\mongoDb with Java> javac Select.java
\NGT\mongoDb with Java> java Select
```

```
Connected to the database successfully
Collection sampleCollection selected successfully
Aug 25, 2023 10:30:30 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: No server chosen by ReadPreferenceServerSelector{readPreference=primary} from cluster description ClusterDe
scription{type=UNKNOWN, connectionMode=SINGLE, all=[ServerDescription{address=localhost:27017, type=UNKNOWN, stat
e=CONNECTING}]}. Waiting for 30000 ms before timing out
Aug 25, 2023 10:30:30 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:149}] to localhost:27017
Aug 25, 2023 10:30:30 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017,
 type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versionList=[5, 0, 5]}, minWireVersion=0, maxWi
reVersion=13, electionId=null, maxDocumentSize=16777216, roundTripTimeNanos=657100}
Aug 25, 2023 10:30:30 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:150}] to localhost:27017
Document{{_id=64e834dabfd59ac80831da97, Name=Sudesh}}
```

MAP REDUCE:

Create an employee collection and insert documents.

```
db.EMPLOYEE_6417.insert([
  {
    name: 'Sudesh Rajbhar',
    gender: 'Male',
    department: 'IT'
  },
  {
    name: 'Prabhat Sharma',
    gender: 'Female',
    department: 'CS'
  },
  {
    name: 'Raj Tripathi',
    gender: 'Male',
    department: 'IT'
  },
  {
    name: 'Pranay Sawant',
    gender: 'Male',
    department: 'HR'
  },
  {
    name: 'Vishal Yadav',
    gender: 'Male',
    department: 'HR'
  },
  {
    name: 'Jagadish',
    gender: 'Female',
    department: 'HR'
  },
  {
    name: 'Shivam',
    gender: 'Male',
    department: 'IT'
  },
  {
    name: 'Vivek',
    gender: 'Male',
    department: 'CS'}])
```

```
> db.EMPLOYEE_6417.insert([
    {
      name: 'Sudesh Rajbhar',
      gender: 'Male',
      department: 'IT'
    },
    {
      name: 'Prabhat Sharma',
      gender: 'Female',
      department: 'CS'
    },
    {
      name: 'Raj Tripathi',
      gender: 'Male',
      department: 'IT'
    },
    {
      name: 'Pranay Sawant',
      gender: 'Male',
      department: 'HR'
    },
    {
      name: 'Vishal Yadav',
      gender: 'Male',
      department: 'HR'
    },
    {
      name: 'Jagadish',
      gender: 'Female',
      department: 'HR'
    },
    {
      name: 'Shivam',
      gender: 'Male',
      department: 'IT'
    },
    {
      name: 'Vivek',
      gender: 'Male',
      department: 'CS'
    }
  ])
```

```
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("64dee3901dec782d638e7fd4"),
    '1': ObjectId("64dee3901dec782d638e7fd5"),
    '2': ObjectId("64dee3901dec782d638e7fd6"),
    '3': ObjectId("64dee3901dec782d638e7fd7"),
    '4': ObjectId("64dee3901dec782d638e7fd8"),
    '5': ObjectId("64dee3901dec782d638e7fd9"),
    '6': ObjectId("64dee3901dec782d638e7fda"),
    '7': ObjectId("64dee3901dec782d638e7fdb")
  }
}
```

Creating Map function:

```
var map =function (){emit(this.gender,1);};
```

Creating Reduce function:

```
var reduce= function(key,value){return Array.sum(value);};
```

Mapping:

```
db.EMPLOYEE_6417.mapReduce(map, reduce, {out: "mapreducecount1"})
```

OR USE AGGREGATE FUNCTION:

```
db.EMPLOYEE_6417.aggregate([
  {
    $group: {
      _id: "$gender",
      count: { $sum: 1 }
    }
  }
]);
```

```
db.EMPLOYEE_6417.aggregate([
 {
```

```
  $group: {
    _id: "$gender",
    count: { $sum: 1 }
  }
 }
]);
```

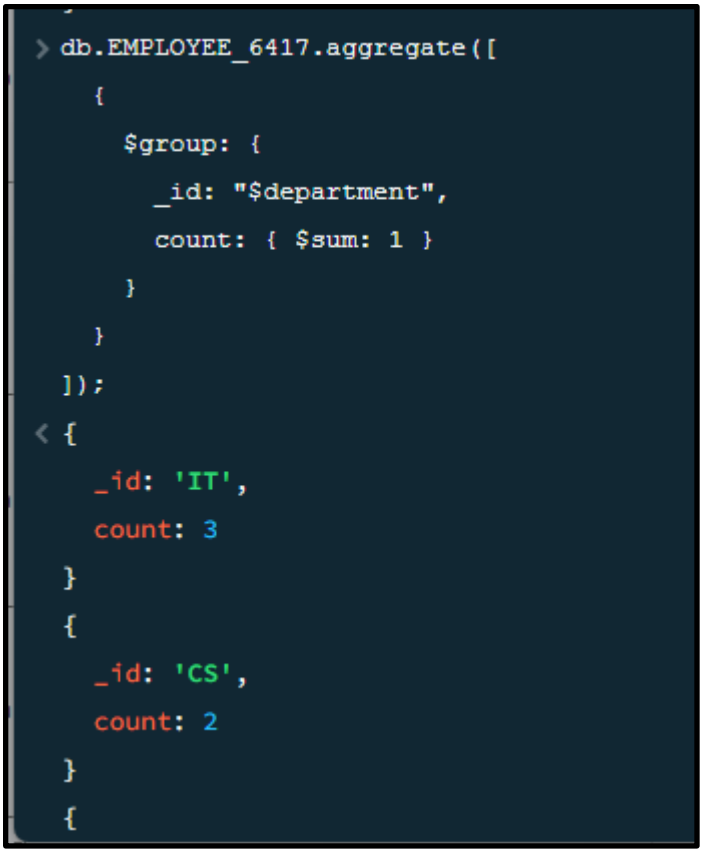
```
< {
    _id: 'Male',
    count: 6
  }
  {
    _id: 'Female',
    count: 2
  }
office>
```

Count total no of employees:

```
> db.EMPLOYEE_6417.aggregate([{$group: {_id: null,totalEmployees: { $sum: 1 }}}]);
< {
    _id: null,
    totalEmployees: 8
  }
```

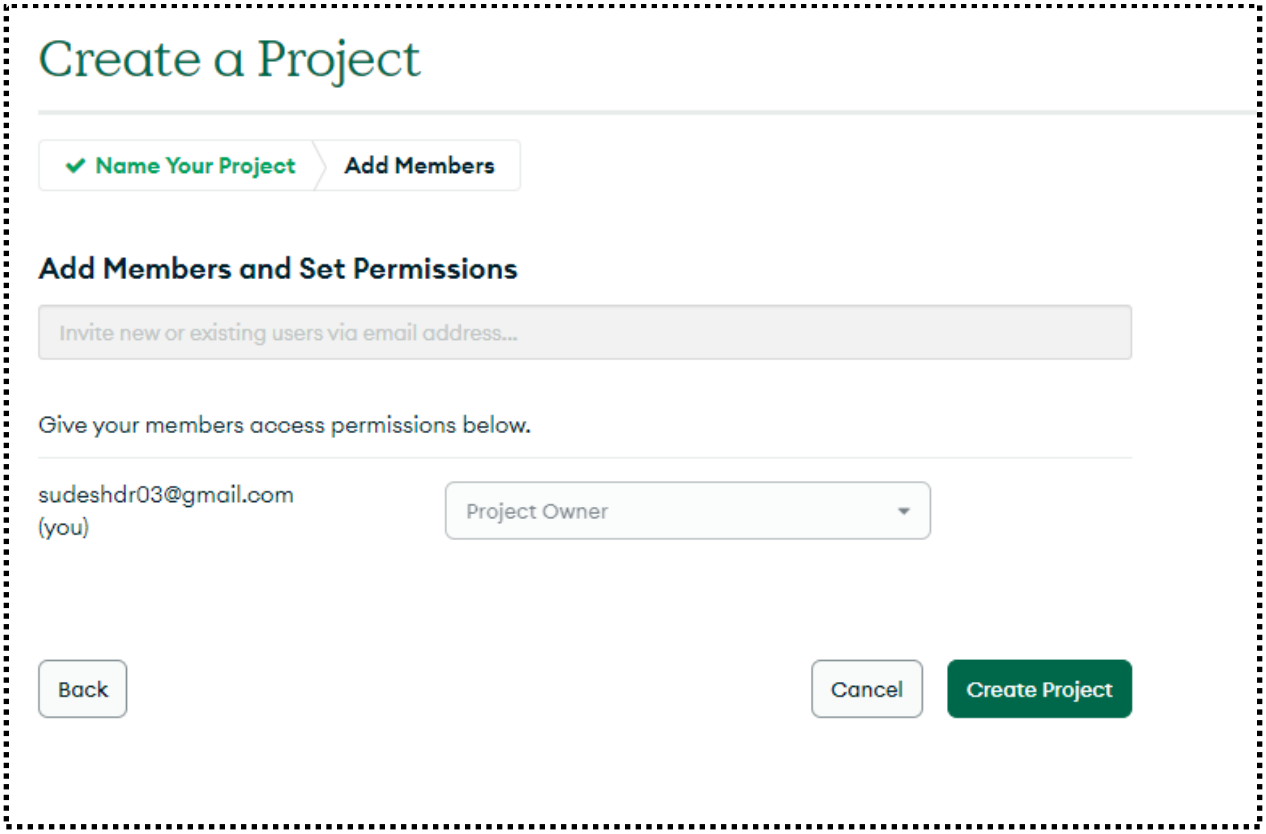
```
db.EMPLOYEE_6417.aggregate([{$group: {_id: null,totalEmployees: { $sum: 1 }}}]);
```

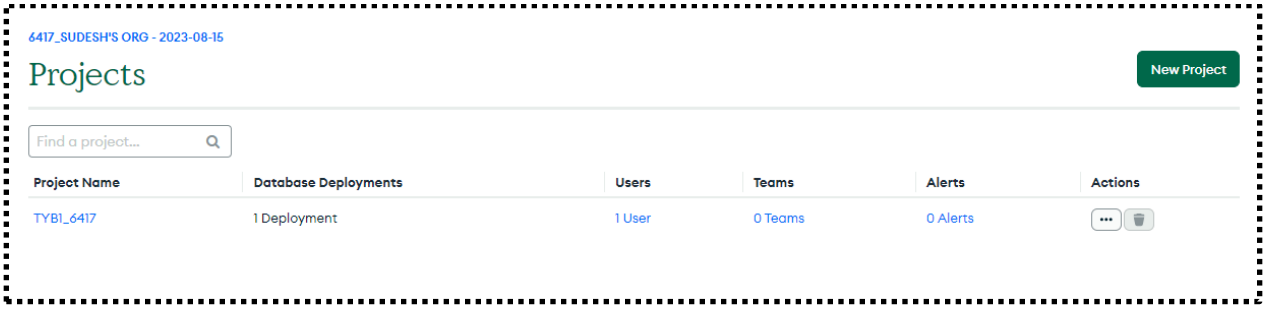
Count employees as per department:

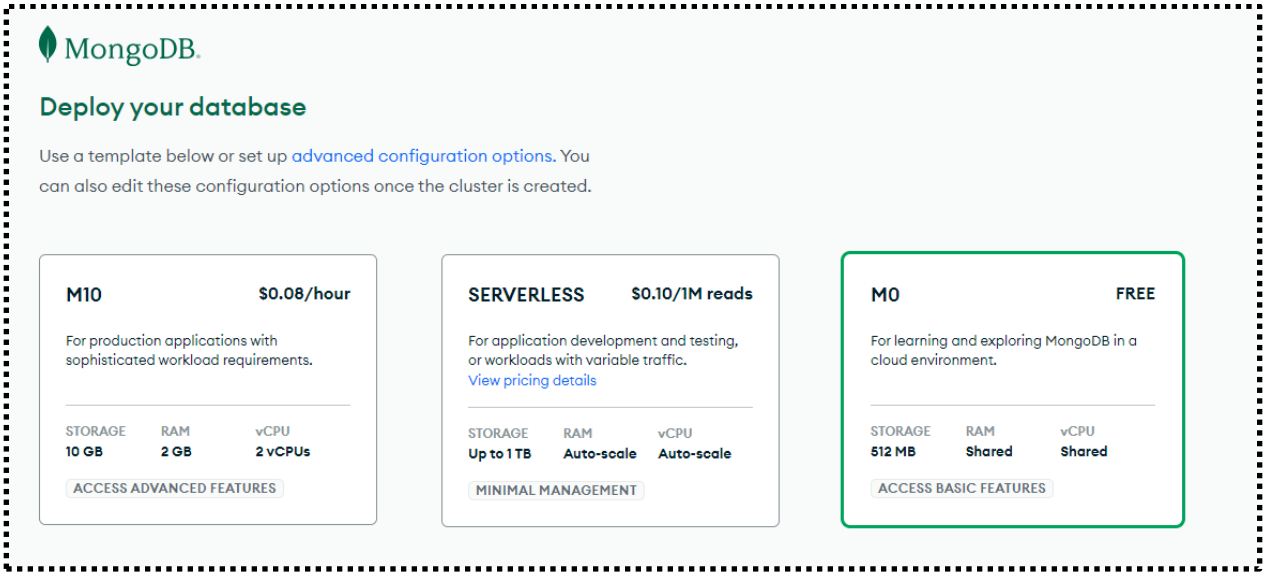


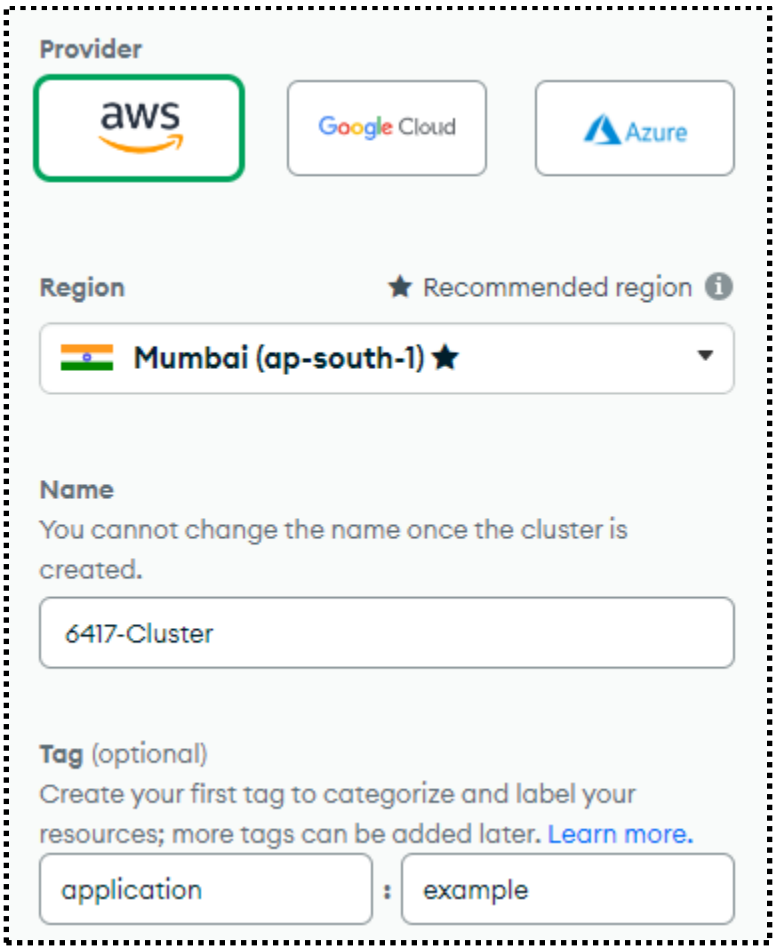

```
db.EMPLOYEE_6417.aggregate([
 {
  $group: {
   _id: "$department",
   count: { $sum: 1 }
  }
 }
]);
```

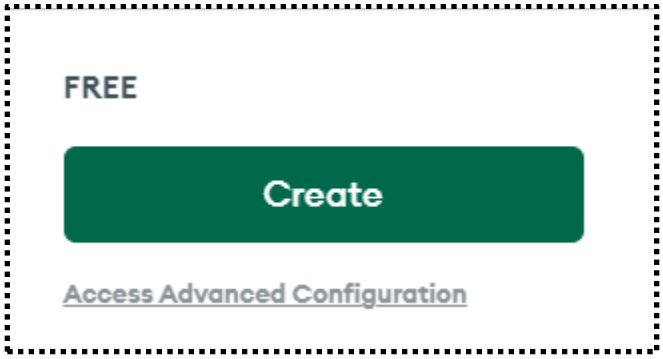
Practical No. 10 Mongo DB ATLAS:

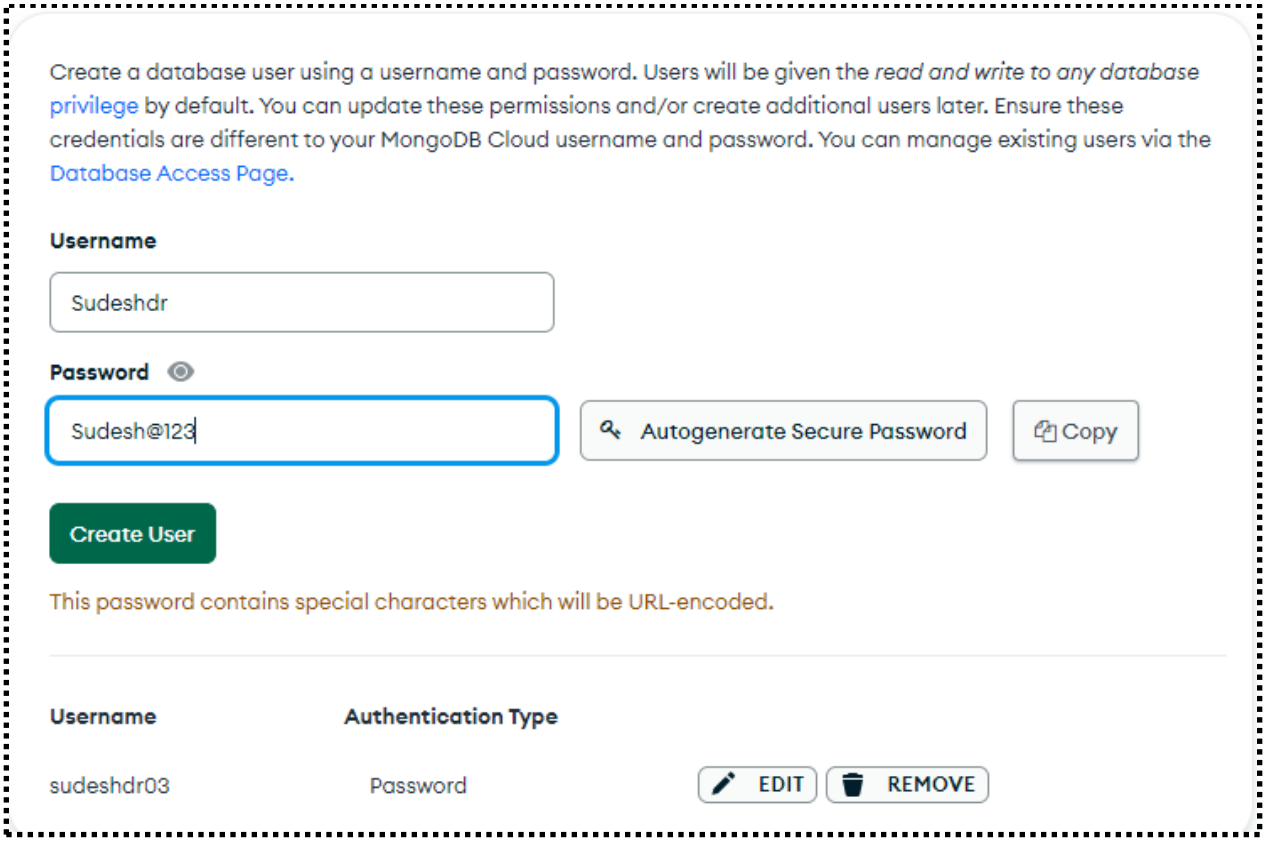




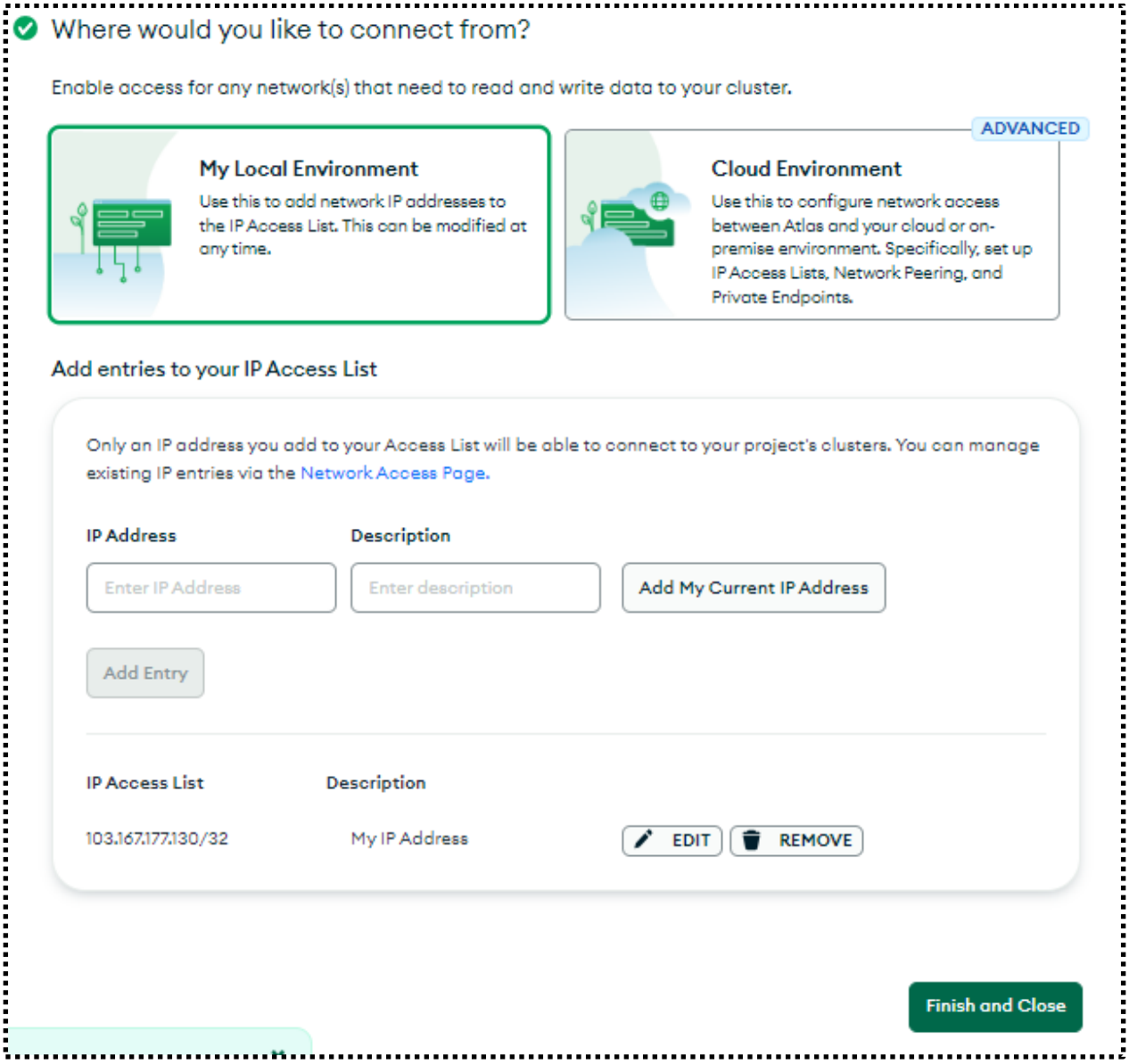

MongoDB.
Deploy your database
Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.
M10
$0.08/hour
For production applications with sophisticated workload requirements.
STORAGE
10 GB
RAM
2 GB
vCPU
2 vCPUs
ACCESS ADVANCED FEATURES
SERVERLESS
$0.10/1M reads
For application development and testing, or workloads with variable traffic.
View pricing details
STORAGE
Up to 1 TB
RAM
Auto-scale
vCPU
Auto-scale
MINIMAL MANAGEMENT
M0
FREE
For learning and exploring MongoDB in a cloud environment.
STORAGE
512 MB
RAM
Shared
vCPU
Shared
ACCESS BASIC FEATURES

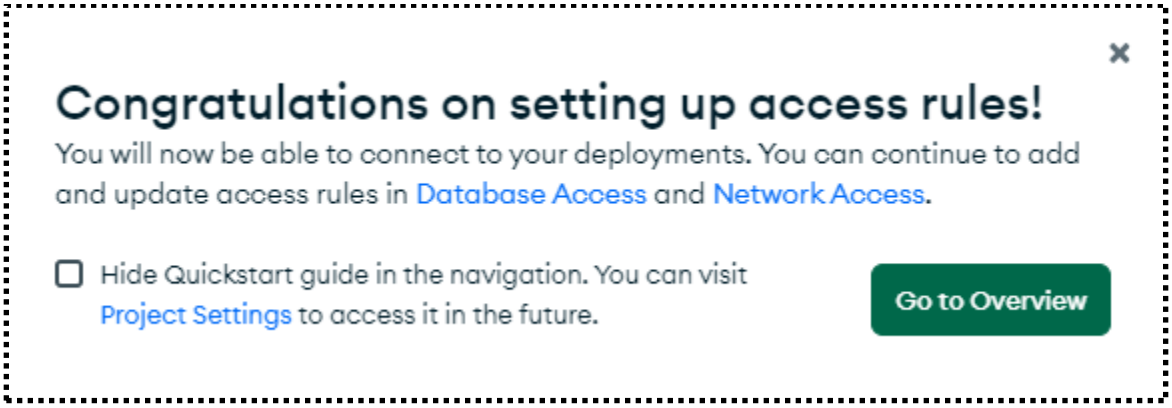

Provider
aws
Google Cloud
Azure
Region
Recommended region
Mumbai (ap-south-1)
Name
You cannot change the name once the cluster is created.
6417-Cluster
Tag (optional)
Create your first tag to categorize and label your resources; more tags can be added later. Learn more.
application
:
example

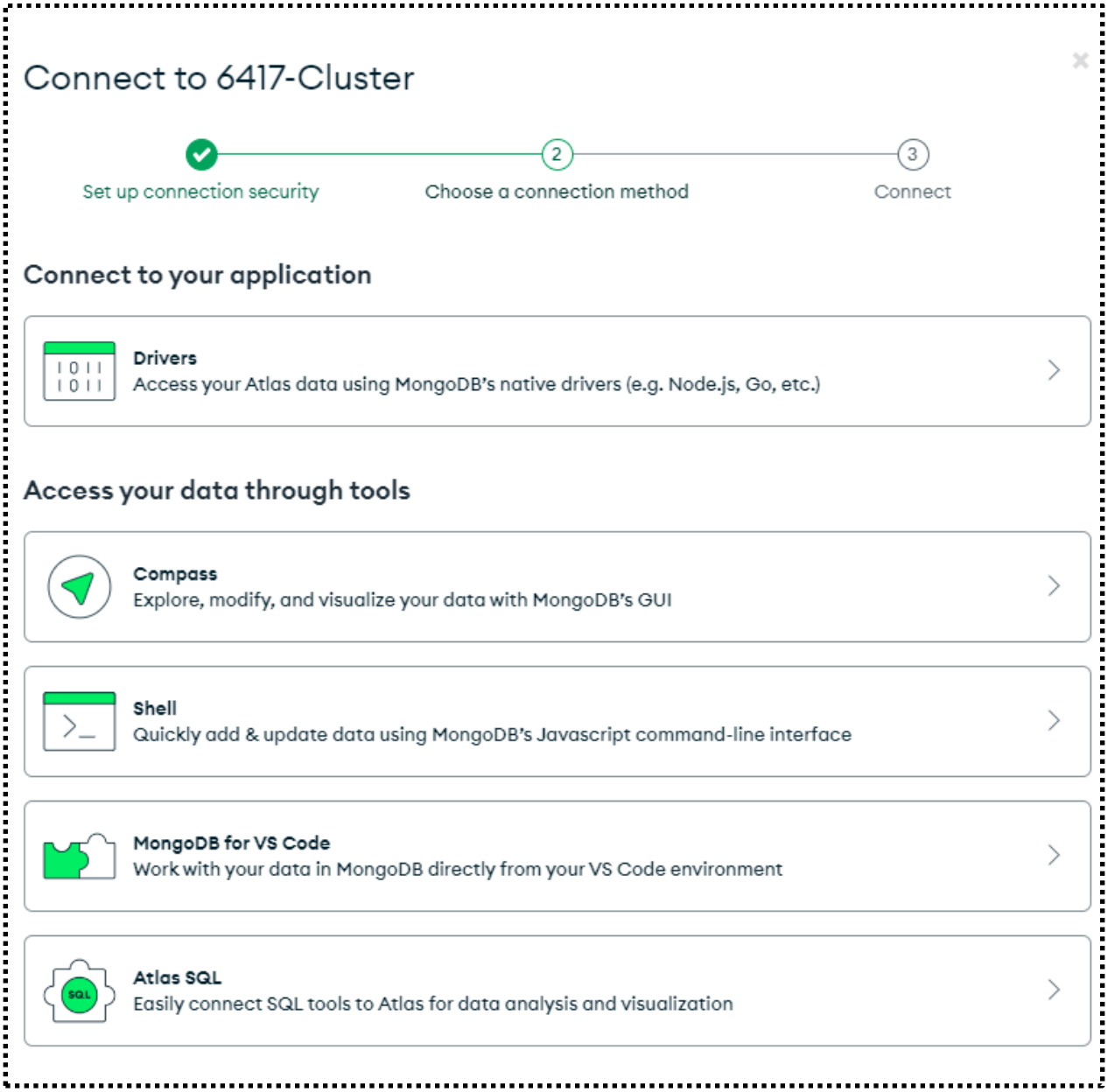

FREE
Create
Access Advanced Configuration


Create a database user using a username and password. Users will be given the read and write to any database privilege by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password. You can manage existing users via the Database Access Page.
Username
Sudeshdr
Password
Sudesh@123
Autogenerate Secure Password
Copy
Create User
This password contains special characters which will be URL-encoded.
Username
Authentication Type
sudeshdr03
Password
EDIT
REMOVE


FREE
Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.
Cancel
Create Cluster

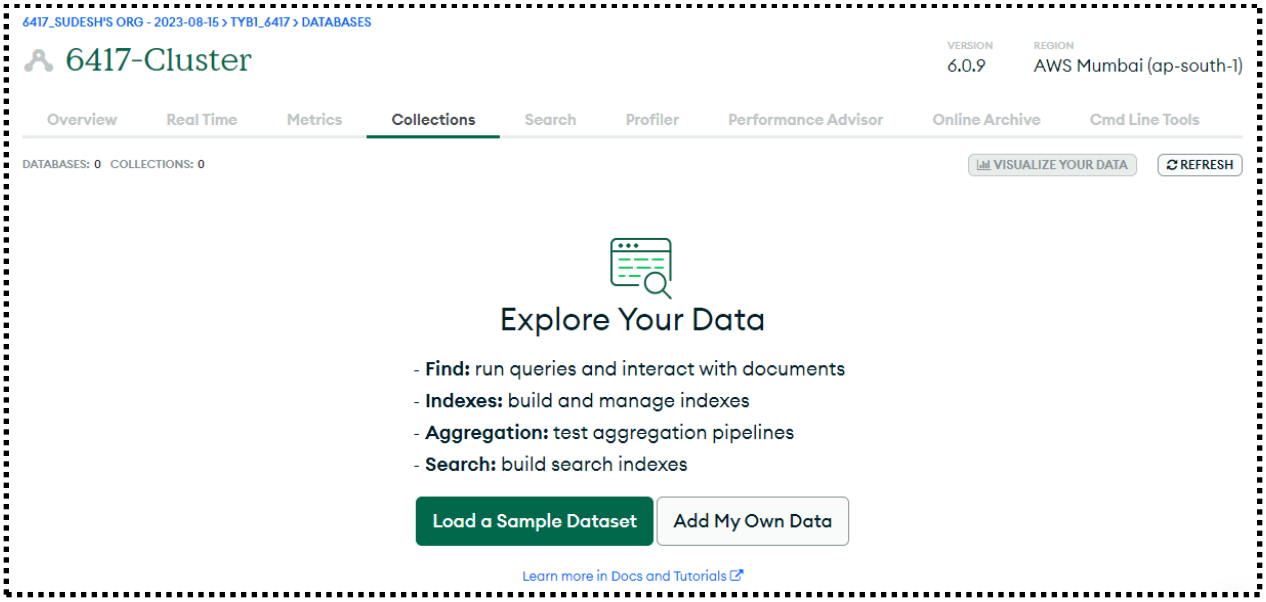

Where would you like to connect from?
Enable access for any network(s) that need to read and write data to your cluster.
My Local Environment
Use this to add network IP addresses to the IP Access List. This can be modified at any time.
ADVANCED
Cloud Environment
Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.
Add entries to your IP Access List
Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the Network Access Page.
IP Address
Description
Enter IP Address
Enter description
Add My Current IP Address
Add Entry
IP Access List
Description
103.167.177.130/32
My IP Address
EDIT
REMOVE
Finish and Close

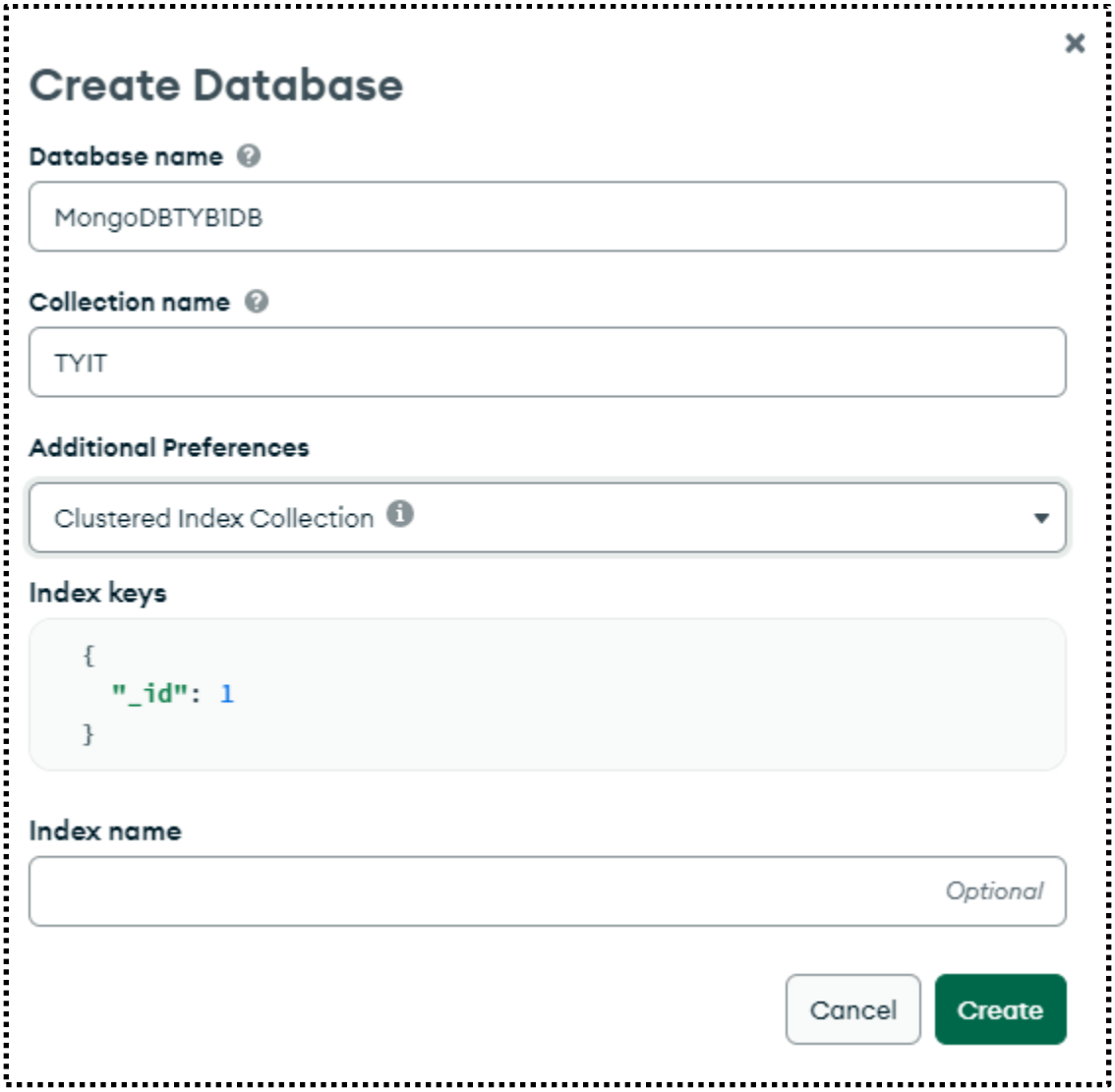

Congratulations on setting up access rules!
You will now be able to connect to your deployments. You can continue to add and update access rules in Database Access and Network Access.
Hide Quickstart guide in the navigation. You can visit Project Settings to access it in the future.
Go to Overview

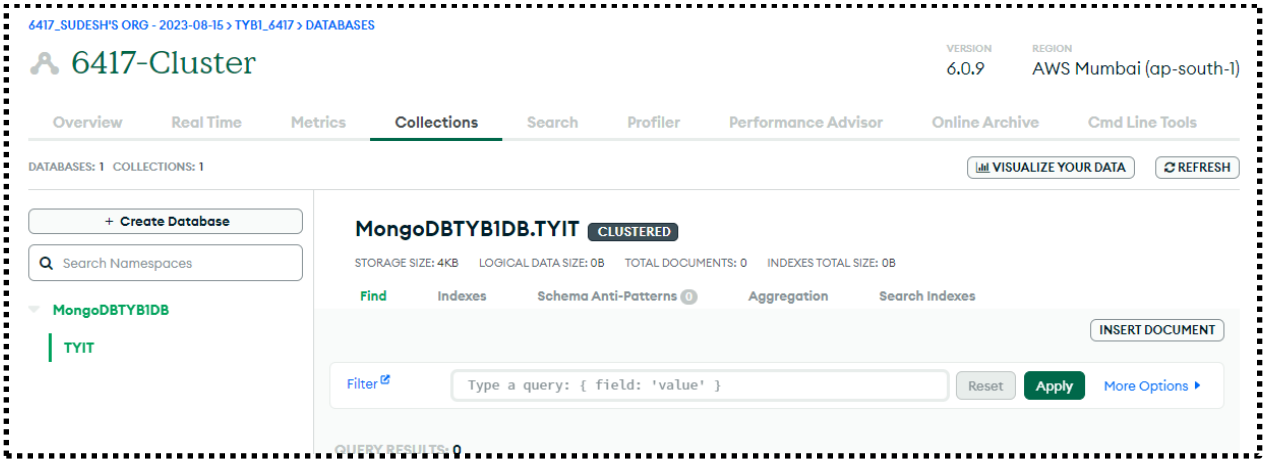

Connect to 6417-Cluster
Set up connection security
2
Choose a connection method
3
Connect
Connect to your application
Drivers
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)
Access your data through tools
Compass
Explore, modify, and visualize your data with MongoDB's GUI
Shell
Quickly add & update data using MongoDB's Javascript command-line interface
MongoDB for VS Code
Work with your data in MongoDB directly from your VS Code environment
Atlas SQL
Easily connect SQL tools to Atlas for data analysis and visualization

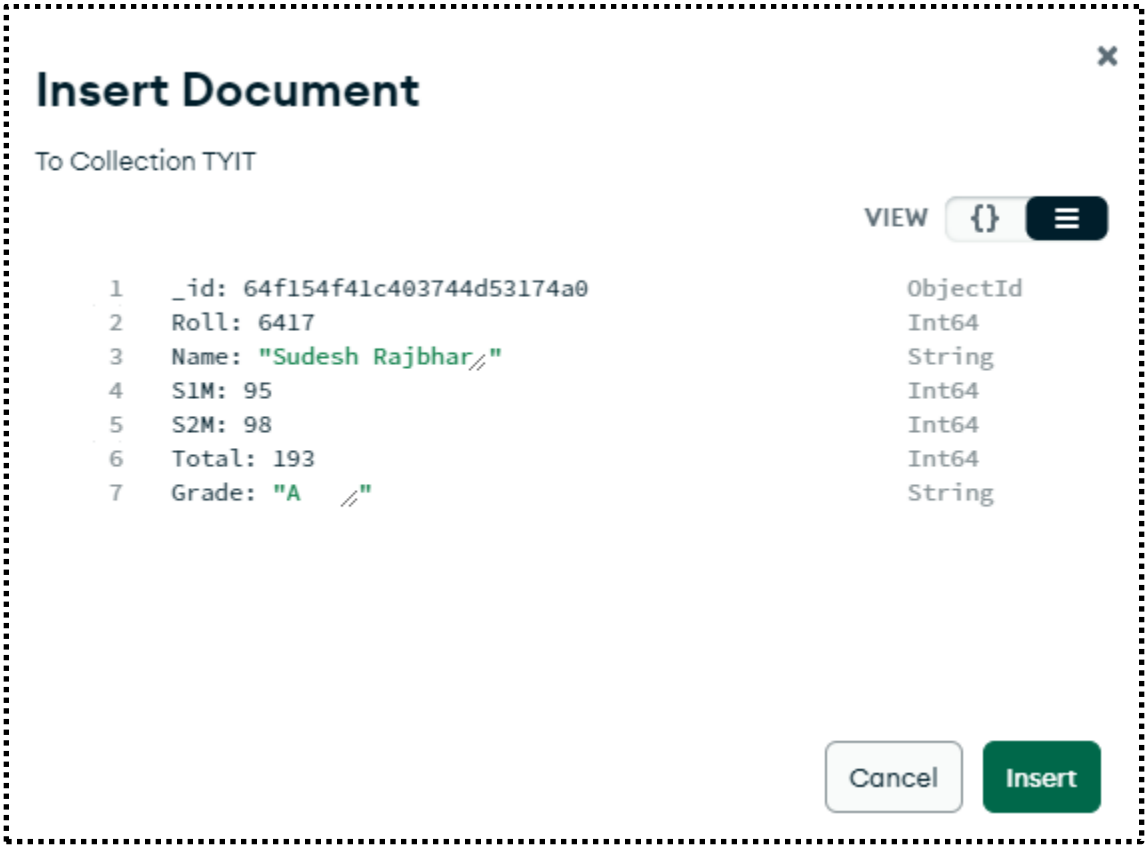
Connection String:

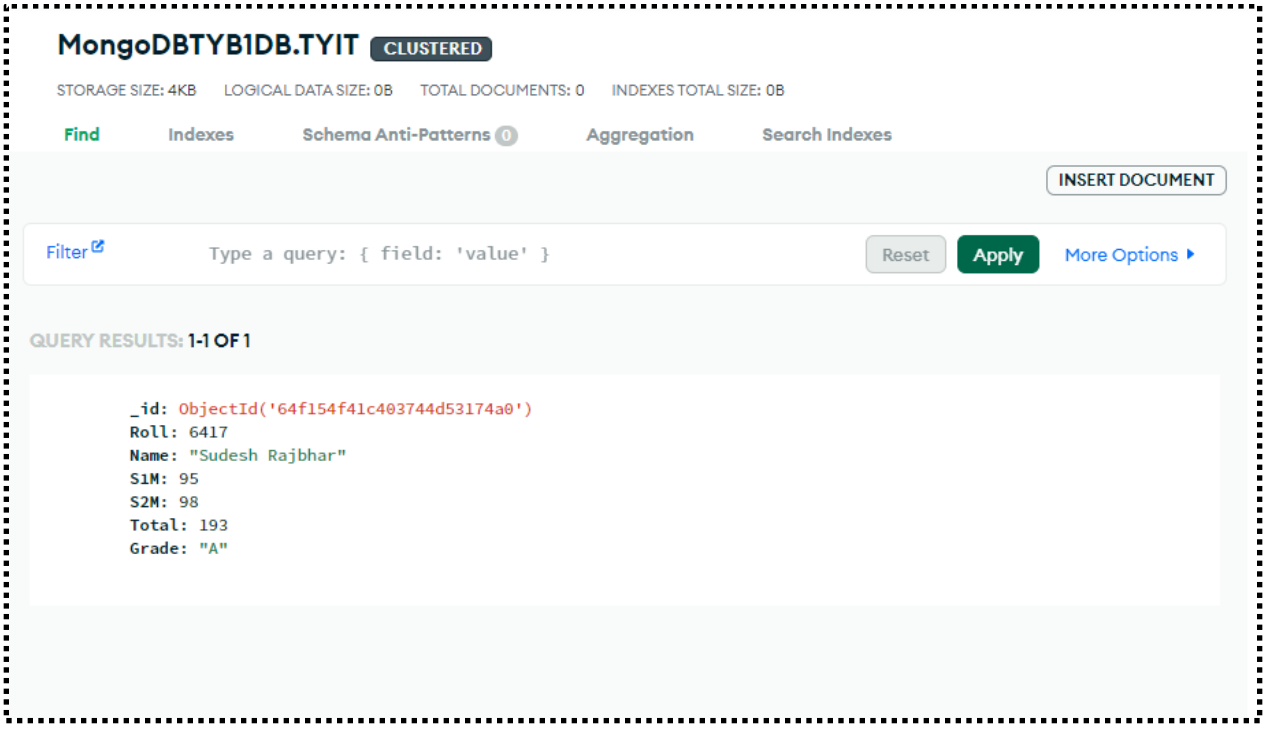
```
mongosh "Mongo DB+srv://6417-cluster.rxtdycc.Mongo DB.net/" --apiVersion 1 --username Sudeshdr
```



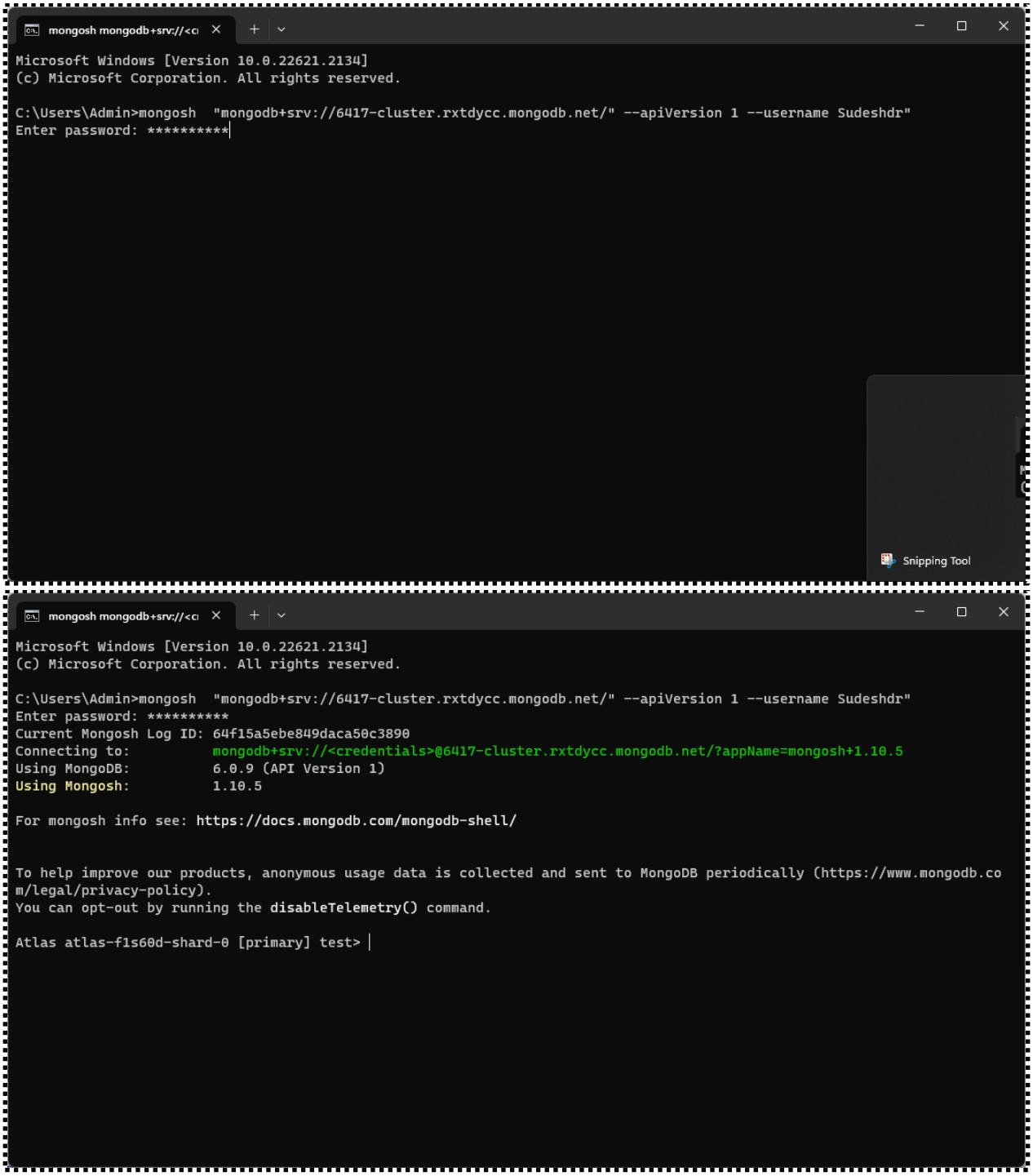

Create Database
Database name
MongoDBTYB1DB
Collection name
TYIT
Additional Preferences
Clustered Index Collection
Index keys
{
"_id": 1
}
Index name
Optional
Cancel
Create


6417_SUDESH'S ORG - 2023-08-15 > TYB1_6417 > DATABASES
6417-Cluster
VERSION
6.0.9
REGION
AWS Mumbai (ap-south-1)
Overview
Real Time
Metrics
Collections
Search
Profiler
Performance Advisor
Online Archive
Cmd Line Tools
DATABASES: 1 COLLECTIONS: 1
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
MongoDBTYB1DB
TYIT
MongoDBTYB1DB.TYIT
CLUSTERED
STORAGE SIZE: 4KB
LOGICAL DATA SIZE: 0B
TOTAL DOCUMENTS: 0
INDEXES TOTAL SIZE: 0B
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
More Options


Insert Document
To Collection TYIT
VIEW
1 _id: 64f154f41c403744d53174a0 ObjectId
2 Roll: 6417 Int64
3 Name: "Sudesh Rajbhar" String
4 S1M: 95 Int64
5 S2M: 98 Int64
6 Total: 193 Int64
7 Grade: "A " String
Cancel
Insert


MongoDBTYB1DB.TYIT CLUSTERED
STORAGE SIZE: 4KB LOGICAL DATA SIZE: 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 0B
Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset Apply More Options
QUERY RESULTS: 1-1 OF 1
_id: ObjectId('64f154f41c403744d53174a0')
Roll: 6417
Name: "Sudesh Rajbhar"
S1M: 95
S2M: 98
Total: 193
Grade: "A"

Open shell and Paste the connection string:

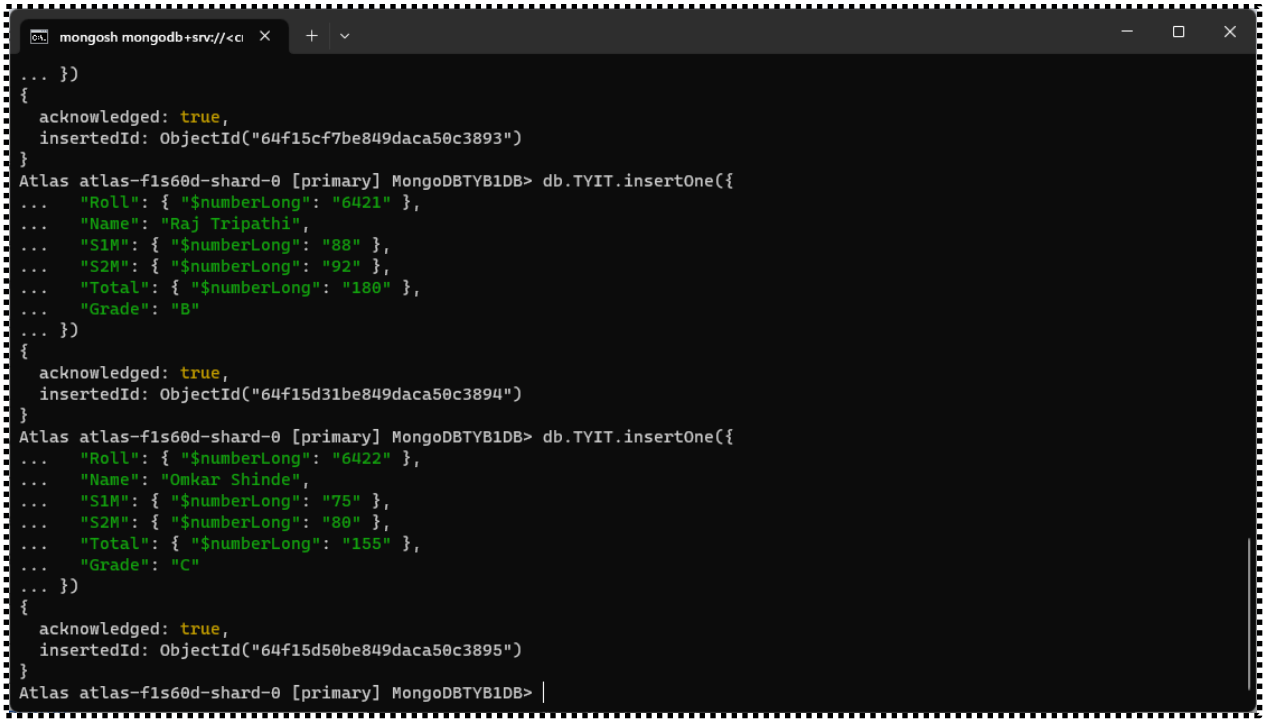


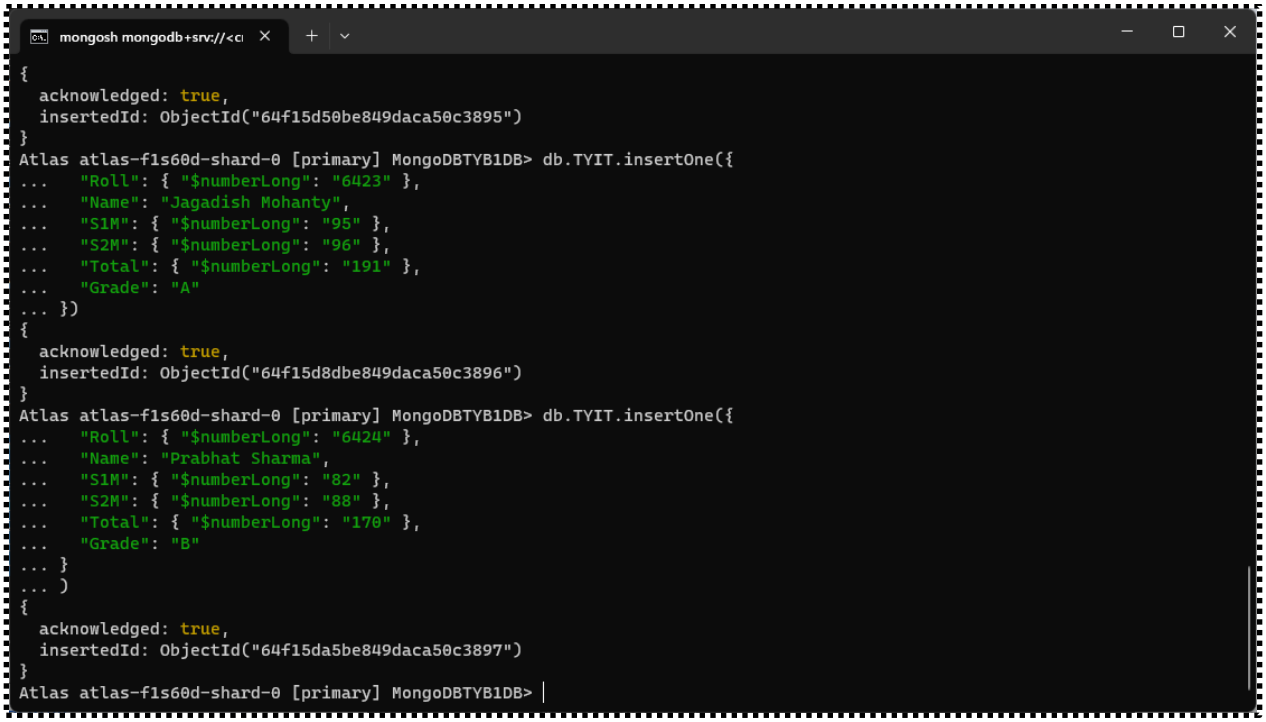
Check the database created on Atlas here:

```
mongosh mongodb+srv://<cl

Using Mongosh:          1.10.5

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


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

Atlas atlas-f1s60d-shard-0 [primary] test> use MongoDBTYB1DB
switched to db MongoDBTYB1DB
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> showCollections
ReferenceError: showCollections is not defined
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> showCollection
ReferenceError: showCollection is not defined
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> show collections
TYIT
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.find()
[
  {
    _id: ObjectId("64f154f41c403744d53174a0"),
    Roll: Long("6417"),
    Name: 'Sudesh Rajbhar',
    S1M: Long("95"),
    S2M: Long("98"),
    Total: Long("193"),
    Grade: 'A'
  }
]
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB>
```

Insert documents on Atlas:

```
mongosh mongodb+srv://<cl

... })
{
  acknowledged: true,
  insertedId: ObjectId("64f15ca0be849daca50c3891")
}
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.insertOne({
...   "Roll": { "$numberLong": "6419" },
...   "Name": "Vishal Yadav",
...   "S1M": { "$numberLong": "92" },
...   "S2M": { "$numberLong": "88" },
...   "Total": { "$numberLong": "180" },
...   "Grade": "B"
... })
{
  acknowledged: true,
  insertedId: ObjectId("64f15ce0be849daca50c3892")
}
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.insertOne({
...   "Roll": { "$numberLong": "6420" },
...   "Name": "Junaid Sayyed",
...   "S1M": { "$numberLong": "78" },
...   "S2M": { "$numberLong": "85" },
...   "Total": { "$numberLong": "163" },
...   "Grade": "C"
... })
{
  acknowledged: true,
  insertedId: ObjectId("64f15cf7be849daca50c3893")
}
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB>
```


mongosh mongodb+srv://<cl
... })
{
acknowledged: true,
insertedId: ObjectId("64f15cf7be849daca50c3893")
}
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.insertOne({
... "Roll": { "$numberLong": "6421" },
... "Name": "Raj Tripathi",
... "S1M": { "$numberLong": "88" },
... "S2M": { "$numberLong": "92" },
... "Total": { "$numberLong": "180" },
... "Grade": "B"
... })
{
acknowledged: true,
insertedId: ObjectId("64f15d31be849daca50c3894")
}
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.insertOne({
... "Roll": { "$numberLong": "6422" },
... "Name": "Omkar Shinde",
... "S1M": { "$numberLong": "75" },
... "S2M": { "$numberLong": "80" },
... "Total": { "$numberLong": "155" },
... "Grade": "C"
... })
{
acknowledged: true,
insertedId: ObjectId("64f15d50be849daca50c3895")
}
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB>


mongosh mongodb+srv://<cl
{
acknowledged: true,
insertedId: ObjectId("64f15d50be849daca50c3895")
}
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.insertOne({
... "Roll": { "$numberLong": "6423" },
... "Name": "Jagadish Mohanty",
... "S1M": { "$numberLong": "95" },
... "S2M": { "$numberLong": "96" },
... "Total": { "$numberLong": "191" },
... "Grade": "A"
... })
{
acknowledged: true,
insertedId: ObjectId("64f15d8dbe849daca50c3896")
}
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.insertOne({
... "Roll": { "$numberLong": "6424" },
... "Name": "Prabhat Sharma",
... "S1M": { "$numberLong": "82" },
... "S2M": { "$numberLong": "88" },
... "Total": { "$numberLong": "170" },
... "Grade": "B"
... }
...)
{
acknowledged: true,
insertedId: ObjectId("64f15da5be849daca50c3897")
}
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB>

Find all the Inserted Documenst:

```
mongosh mongodb+srv://<c
... )
{
  acknowledged: true,
  insertedId: ObjectId("64f15da5be849daca50c3897")
}
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.find()
[
  {
    _id: ObjectId("64f154f41c403744d53174a0"),
    Roll: Long("6417"),
    Name: 'Sudesh Rajbhar',
    S1M: Long("95"),
    S2M: Long("98"),
    Total: Long("193"),
    Grade: 'A'
  },
  {
    _id: ObjectId("64f15ca0be849daca50c3891"),
    Roll: { '$numberLong': '6418' },
    Name: 'Pranay Sawant',
    S1M: { '$numberLong': '85' },
    S2M: { '$numberLong': '90' },
    Total: { '$numberLong': '175' },
    Grade: 'B'
  },
  {
    _id: ObjectId("64f15ce0be849daca50c3892"),
    Roll: { '$numberLong': '6419' },
    Name: 'Vishal Yadav',
    S1M: { '$numberLong': '92' },
    S2M: { '$numberLong': '88' },
    Total: { '$numberLong': '180' },
    Grade: 'B'
  },
  {
    _id: ObjectId("64f15cf7be849daca50c3893"),
    Roll: { '$numberLong': '6420' },
    Name: 'Junaid Sayyed',
    S1M: { '$numberLong': '78' },
    S2M: { '$numberLong': '85' },
    Total: { '$numberLong': '163' },
    Grade: 'C'
```

```
mongosh mongodb+srv://<c
    S2M: { '$numberLong': '85' },
    Total: { '$numberLong': '163' },
    Grade: 'C'
  },
  {
    _id: ObjectId("64f15d31be849daca50c3894"),
    Roll: { '$numberLong': '6421' },
    Name: 'Raj Tripathi',
    S1M: { '$numberLong': '88' },
    S2M: { '$numberLong': '92' },
    Total: { '$numberLong': '180' },
    Grade: 'B'
  },
  {
    _id: ObjectId("64f15d50be849daca50c3895"),
    Roll: { '$numberLong': '6422' },
    Name: 'Omkar Shinde',
    S1M: { '$numberLong': '75' },
    S2M: { '$numberLong': '80' },
    Total: { '$numberLong': '155' },
    Grade: 'C'
  },
  {
    _id: ObjectId("64f15d8dbe849daca50c3896"),
    Roll: { '$numberLong': '6423' },
    Name: 'Jagadish Mohanty',
    S1M: { '$numberLong': '95' },
    S2M: { '$numberLong': '96' },
    Total: { '$numberLong': '191' },
    Grade: 'A'
  },
  {
    _id: ObjectId("64f15da5be849daca50c3897"),
    Roll: { '$numberLong': '6424' },
    Name: 'Prabhat Sharma',
    S1M: { '$numberLong': '82' },
    S2M: { '$numberLong': '88' },
    Total: { '$numberLong': '170' },
    Grade: 'B'
  }
]
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB>
```

QUESTIONS:

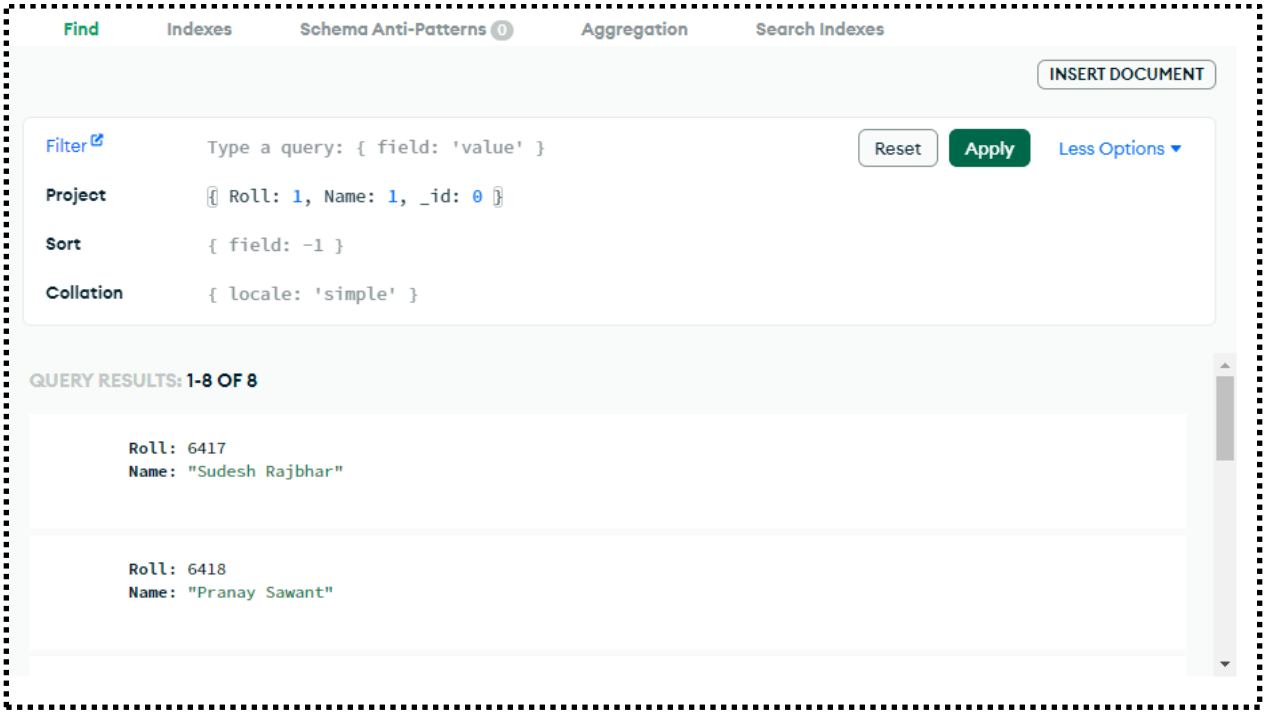
1. List of Students with their Roll Numbers.

```
db.TYIT.find({}, { Roll: 1, Name: 1, _id: 0 })
```

In Shell:

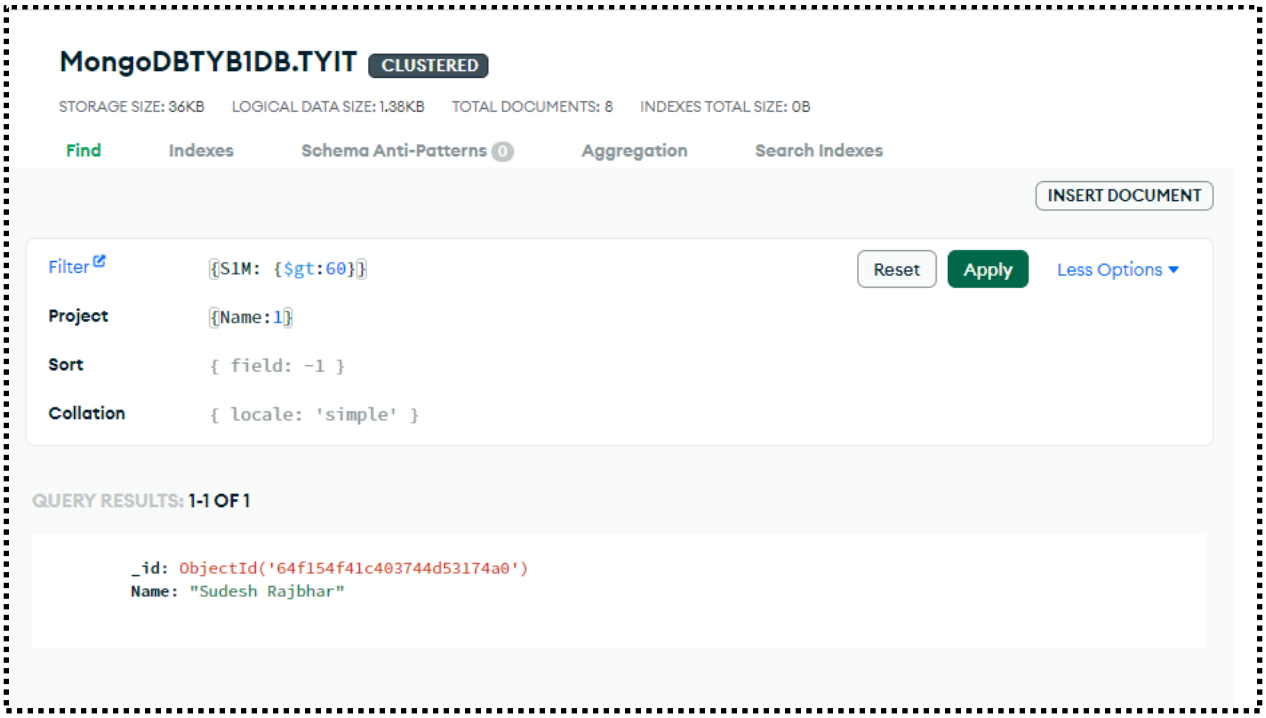
```
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.find({}, { Roll: 1, Name: 1, _id: 0 })
[
  { Roll: Long("6417"), Name: 'Sudesh Rajbhar' },
  { Roll: { '$numberLong': '6418' }, Name: 'Pranay Sawant' },
  { Roll: { '$numberLong': '6419' }, Name: 'Vishal Yadav' },
  { Roll: { '$numberLong': '6420' }, Name: 'Junaid Sayyed' },
  { Roll: { '$numberLong': '6421' }, Name: 'Raj Tripathi' },
  { Roll: { '$numberLong': '6422' }, Name: 'Omkar Shinde' },
  { Roll: { '$numberLong': '6423' }, Name: 'Jagadish Mohanty' },
  { Roll: { '$numberLong': '6424' }, Name: 'Prabhat Sharma' }
]
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB>
```

In Atlas:



2. List the Students with more than 60 marks in S1M

IN Atlas:

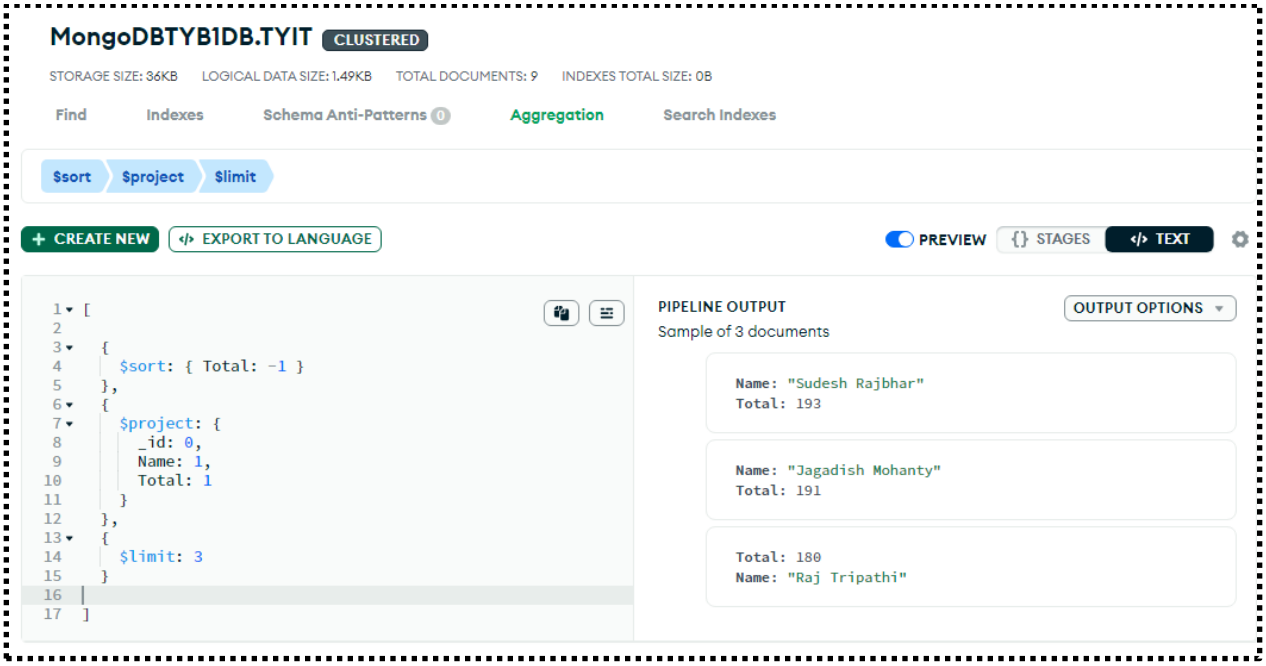


In shell:

```
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.find( { "S1M": { $gt: 60 } }, { "Name": 1, "Roll": 1, "_id": 0 } )
[ { Roll: Long("6417"), Name: 'Sudesh Rajbhar' } ]
```

3. List first three toppers.

IN ATLAS:



IN SHELL:

```
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.aggregate([
...     {
...       $sort: { Total: -1 }
...     },
...     {
...       $project: {
...         _id: 0,
...         Name: 1,
...         Total: 1
...       }
...     },
...     {
...       $limit: 3
...     }
... ])
[
  { Name: 'Sudesh Rajbhar', Total: { '$numberLong': '193' } },
  { Name: 'Jagadish Mohanty', Total: { '$numberLong': '191' } },
  { Name: 'Raj Tripathi', Total: { '$numberLong': '180' } }
]
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB>
```

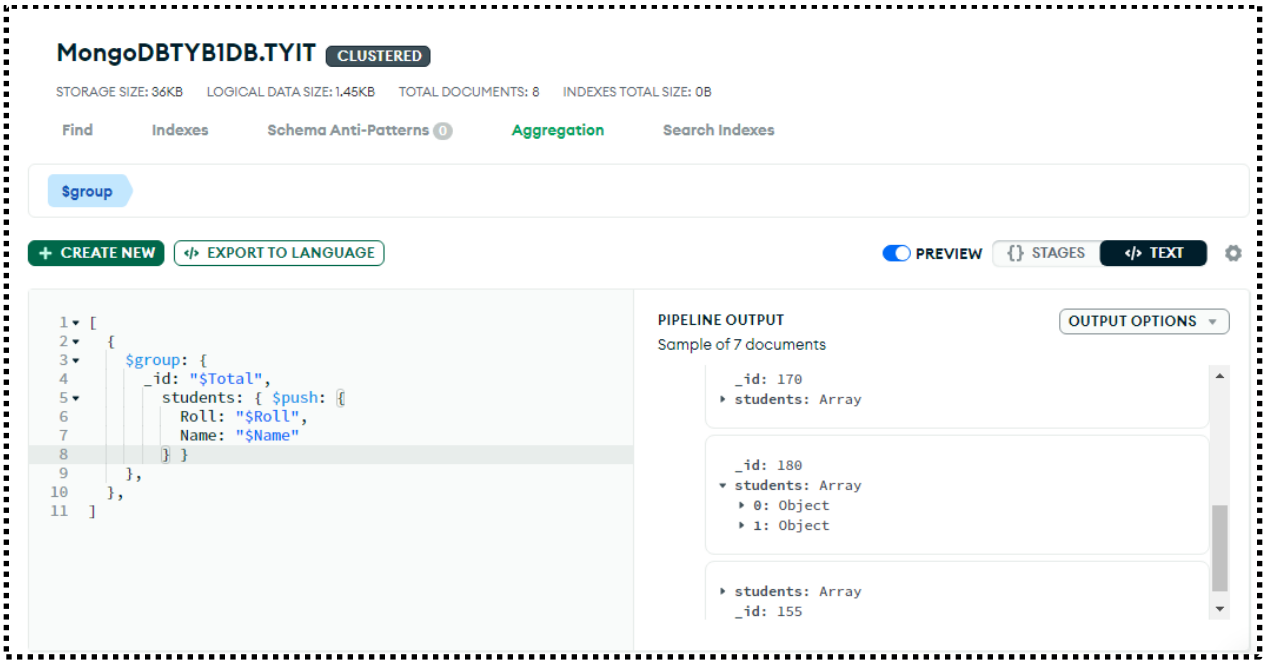
4. Display the student with C grade.

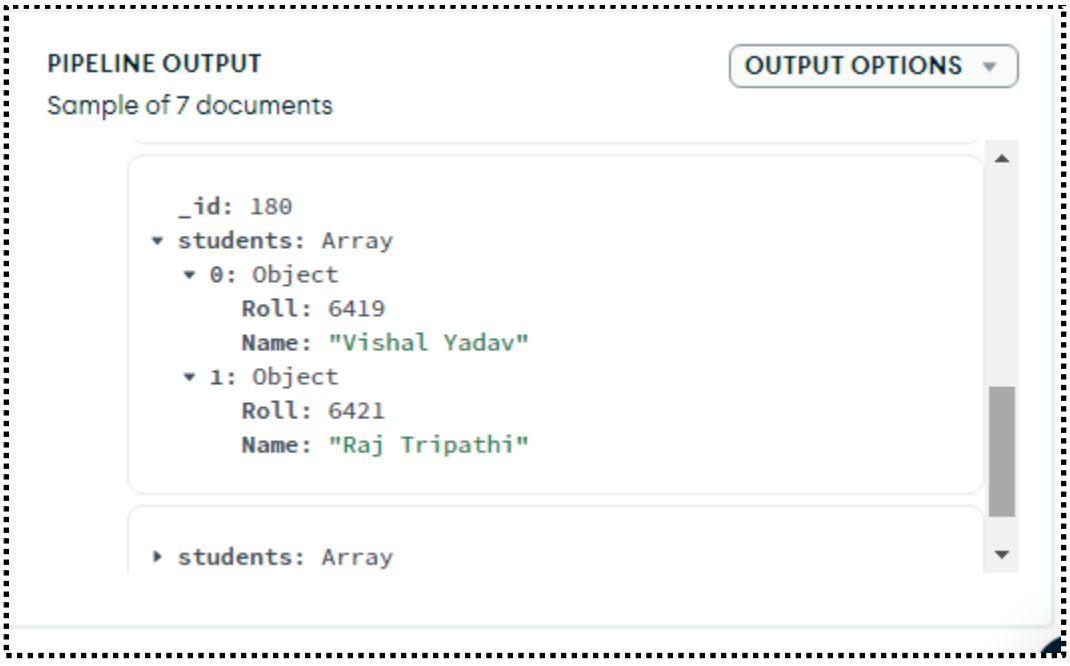
In Atlas:

In Shell:

```
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.find({ Grade: 'C' }, { _id: 0, Name: 1, Grade: 1 })
[
  { Name: 'Junaid Sayyed', Grade: 'C' },
  { Name: 'Omkar Shinde', Grade: 'C' }
]
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB>
```

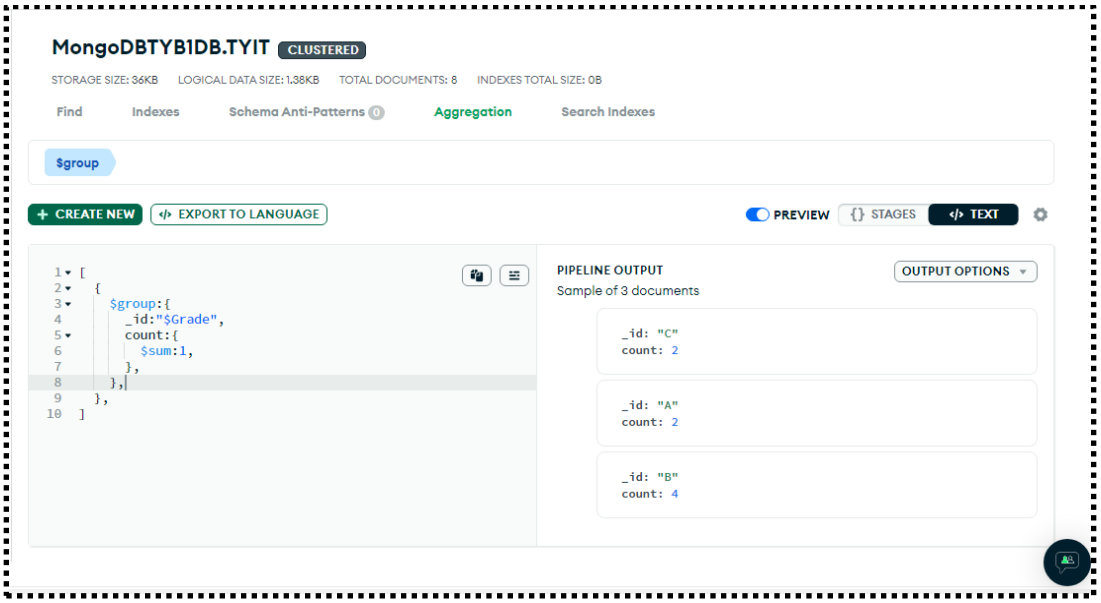
5. Group the students with similar total marks





6. Display grade wise count of students

In atlas:



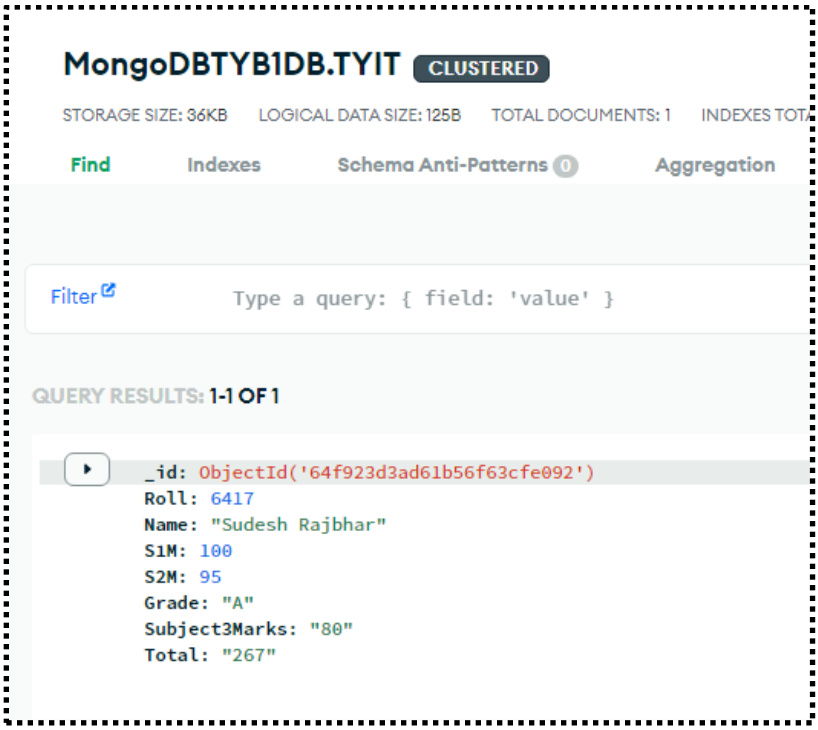
In shell:

```
...     {
...       $group: {
...         _id: "$Grade",
...         count: { $sum: 1 }
...       }
...     },
...     {
...       $project: {
...         _id: 0,
...         Grade: "$_id",
...         Count: "$count"
...       }
...     }
... ])
db.TYIT.aggregate([
  {
    $group: {
      _id: "$Grade",
      count: { $sum: 1 }
[
  { Grade: 'C', Count: 2 },
  { Grade: 'A', Count: 2 },
  { Grade: 'B', Count: 4 }
]
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.aggregate([
```

7. Alter the marks of subject two for any student.

db.TYIT.updateOne({ Roll: 6417 },{ $set: { S2M: 95 } })

Before:



After:

IN SHELL:

```
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.updateOne({ Roll: 6417 },{ $set: { S2M: 100 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

8. Add the subject 3 column into the same collection and update sem3 marks.

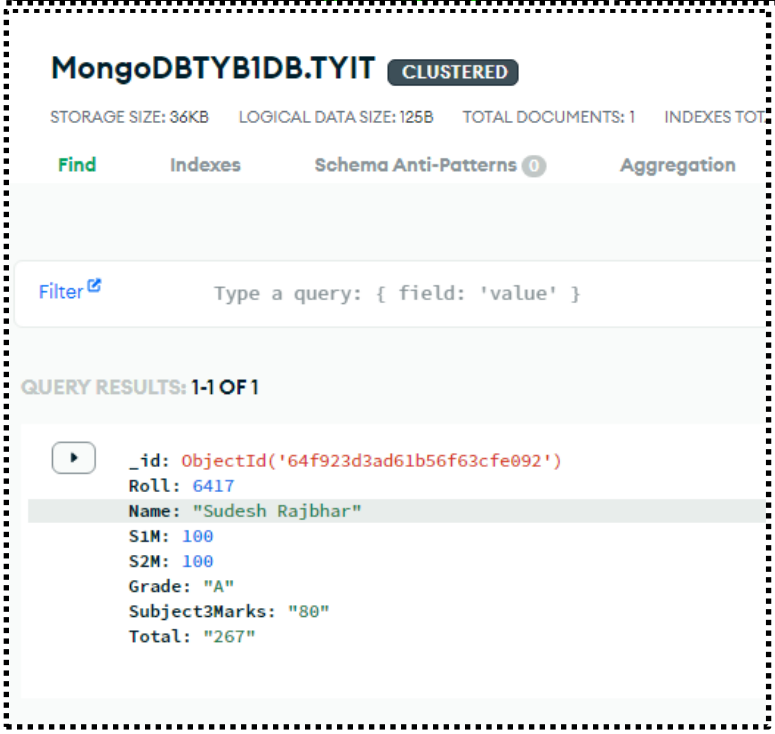
db.TYIT.updateMany({}, { $set: { "Subject3Marks": null } })

```
db.TYIT.find({ S1M: { $gt: 60 } },{ _id: 0, Name: 1, S1M: 1 });
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.updateMany({}, { $set: { "Subject3Marks": null } })
db.TYIT.updateMany({}, { $set: { "Subject3Marks": null } })
{
  acknowledged: true,
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB>
```

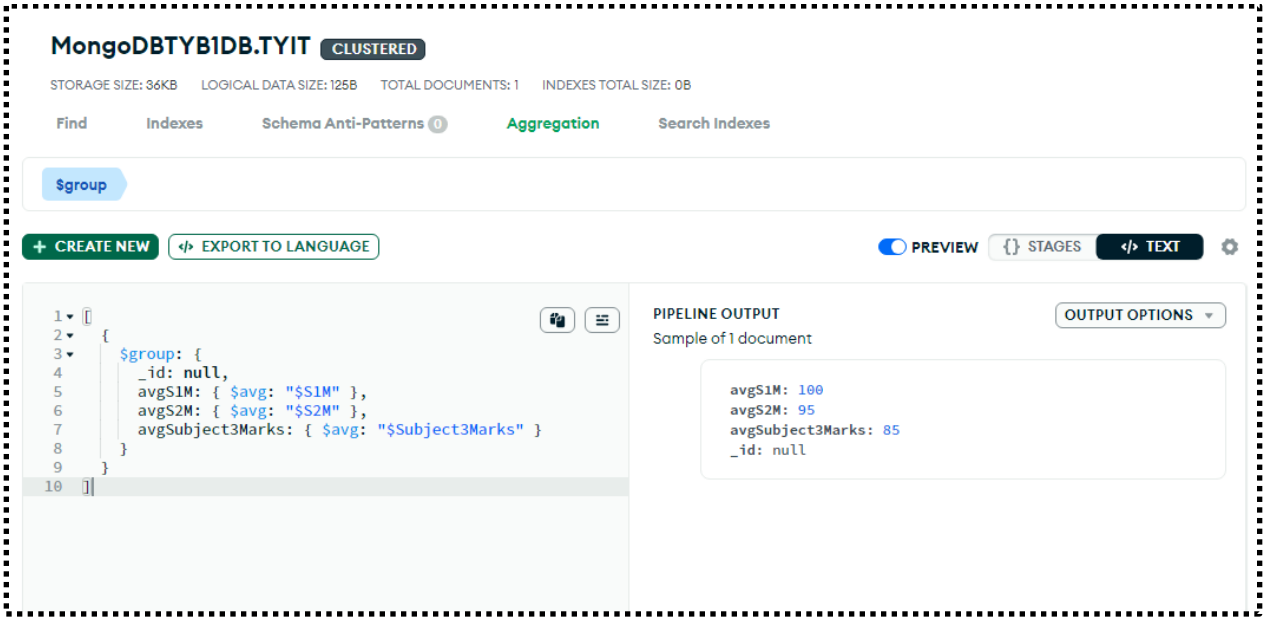
Add subject3marks:

db.TYIT.updateMany({}, { $set: { Subject3Marks: 80 } });

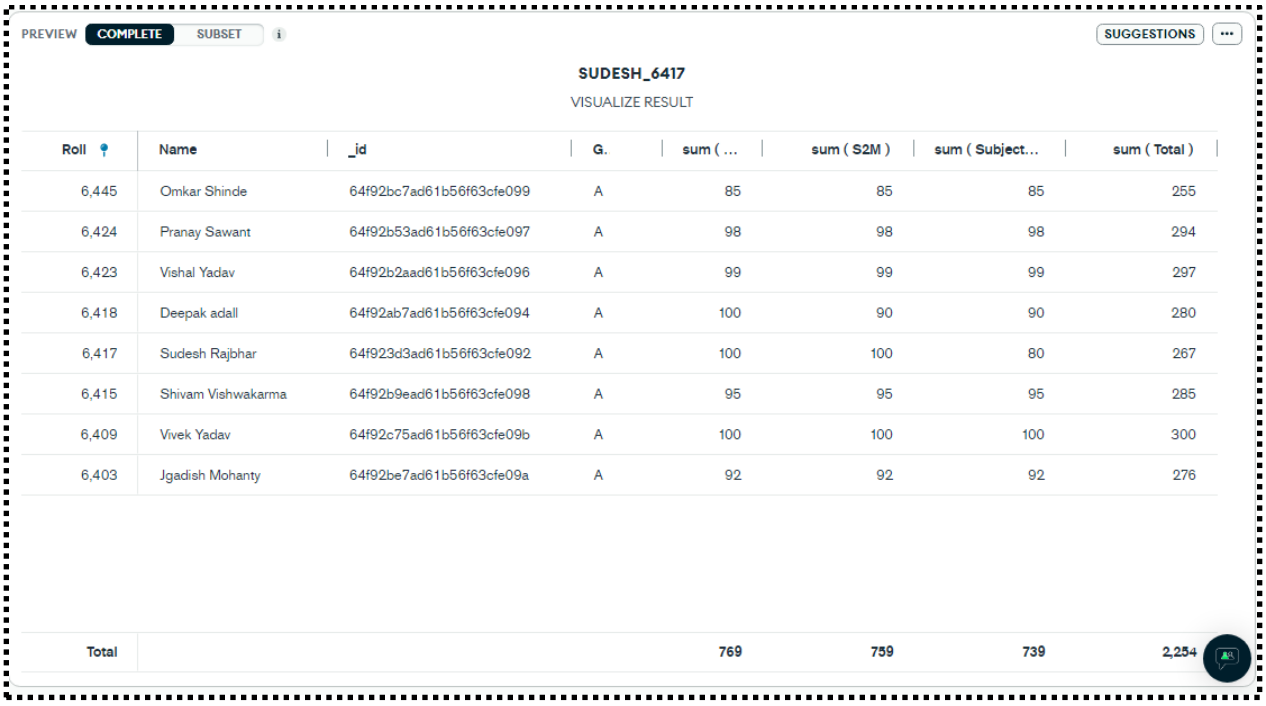
```
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB> db.TYIT.updateMany({}, { $set: { Subject3Marks: 80 } } );
ateMany({}, { $set: { Subject3Marks: 80 } } );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 0,
  upsertedCount: 0
}
Atlas atlas-f1s60d-shard-0 [primary] MongoDBTYB1DB>
```



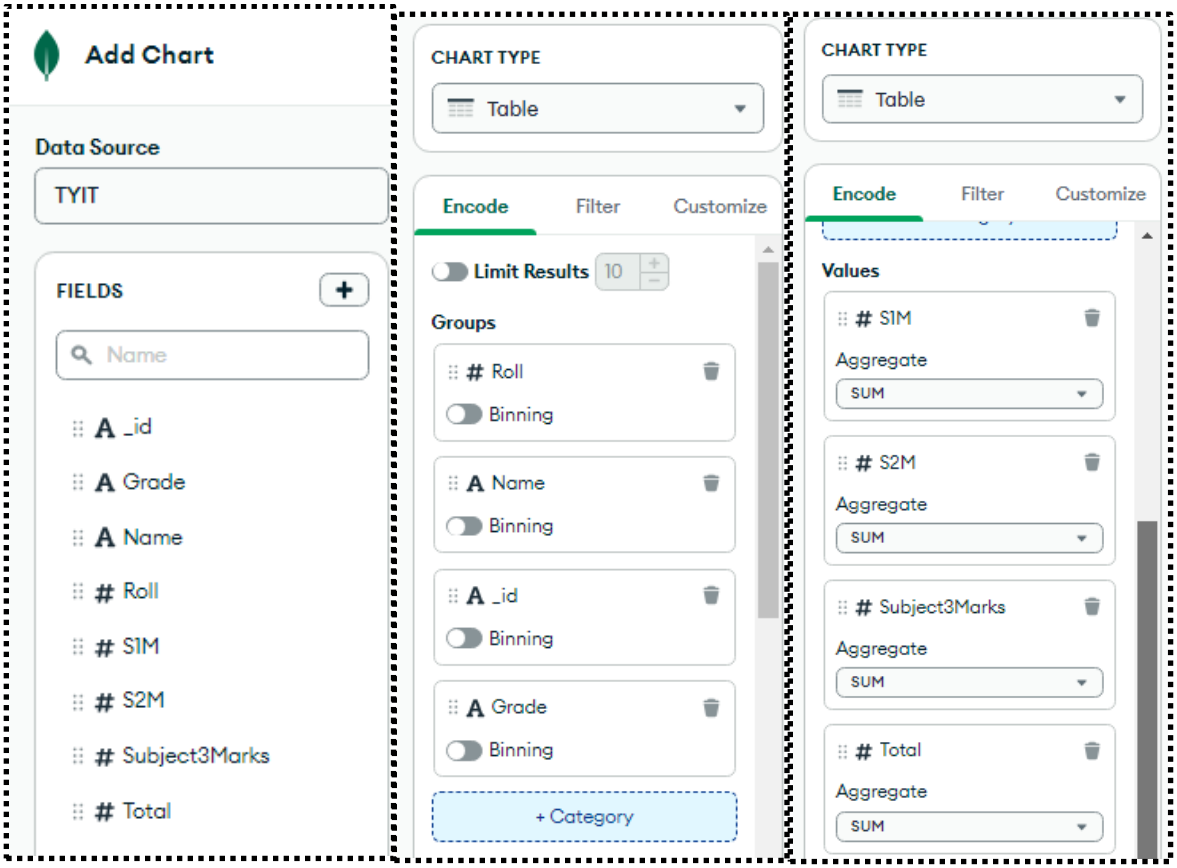
9. Display the average marks for all subjects



10.Visualize the result.



Roll	Name	_id	G.	sum (...	sum (S2M)	sum (Subject...	sum (Total)
6,445	Omkar Shinde	64f92bc7ad61b56f63cfe099	A	85	85	85	255
6,424	Pranay Sawant	64f92b53ad61b56f63cfe097	A	98	98	98	294
6,423	Vishal Yadav	64f92b2aad61b56f63cfe096	A	99	99	99	297
6,418	Deepak adall	64f92ab7ad61b56f63cfe094	A	100	90	90	280
6,417	Sudesh Rajbhar	64f923d3ad61b56f63cfe092	A	100	100	80	267
6,415	Shivam Vishwakarma	64f92b9ead61b56f63cfe098	A	95	95	95	285
6,409	Vivek Yadav	64f92c75ad61b56f63cfe09b	A	100	100	100	300
6,403	Jgadish Mohanty	64f92be7ad61b56f63cfe09a	A	92	92	92	276
Total				769	759	739	2,254


Add Chart
Data Source
TYIT
FIELDS
Name
_id
Grade
Name
Roll
S1M
S2M
Subject3Marks
Total
CHART TYPE
Table
Encode
Filter
Customize
Limit Results
10
Groups
Roll
Binning
Name
Binning
_id
Binning
Grade
Binning
+ Category
CHART TYPE
Table
Encode
Filter
Customize
Values
S1M
Aggregate
SUM
S2M
Aggregate
SUM
Subject3Marks
Aggregate
SUM
Total
Aggregate
SUM

Practical No 11: JQUERY.

- jQuery Basic
- jQuery Events

Jquery is a lighweight Javascript library which is blazing fast and concise.

Jquery has been designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS Animation and Ajax.

Develop a website using JQuery Language.

Source Code :

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Sudesh JQuery</title>
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>
   <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Poppins">

   <style>
      body {
         font-family: 'Poppins', sans-serif;
      }
      /* Header styles */
      header {
```

```
        background: linear-gradient(to right, #1100ff, #00d9ff); /* Gradient background */
        padding: 10px;
        display: flex;
        align-items: center;
        border-radius: 20px; /* Rounded ends */
    }

    /* Text color for the header */
    #header-text {
        color: rgb(58, 29, 138); /* Change the text color to white */
        flex-grow: 1; /* Allow header text to take available space */
    }

    /* Button styles with gradient background */
    #header-button {
        background: linear-gradient(to bottom, #ae00ff, #ff3300); /* Apply gradient color
to the button */
        border: none;
        color: white;
        padding: 15px; /* Increase button height to fill the header */
        cursor: pointer;
        border-radius: 5px;
        margin-left: 10px; /* Add margin for spacing */
        font-family: 'Poppins', sans-serif;
        height:fit-content;
    }

    /* Hover effect for the button */
    #header-button:hover {
        background: linear-gradient(to bottom, #ff3300, #ff6600); /* Change the gradient
color on hover */
```

```
    }
    .menu-container {
        display: flex; /* Display items horizontally */
        justify-content: space-around; /* Distribute items evenly */
        padding: 20px; /* Add some padding for spacing */
        color: #1100ff;
        background: linear-gradient(to bottom, #ff3300, #ff6600); /* Change the gradient
color on hover */
    }

    .menu-item {
        background-color: #fbff00;
        border: 1px solid #ccc;
        border-radius: 5px;
        width:200px;
        padding: 10px;
        flex-grow: 1; /* Allow menu items to take up equal space */
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        text-align: center;
        transition: transform 0.2s;
        font-size:larger;
    }

    .menu-item:hover {
        transform: scale(1.05);
        box-shadow: 0 6px 10px rgba(0, 0, 0, 0.2);
    }
    .image-container {
        display: flex;
        max-height: 70vh; /* Set a maximum height for the container */
```

```
        margin: 0 auto; /* Center the container horizontally */
        overflow: hidden; /* Hide overflowing content */
    }

    .image-container img {
        width: 50%;
        height: 50vh; /* 100% of viewport height */
    }
    </style>
</head>
<body>
  <fieldset>
  <header>
    <h1 id="header-text">Sudesh stocks</h1>
    <button id="header-button">Click me</button>
  </header>
  </fieldset>

  <fieldset>
    <div class="menu-container">
      <div class="menu-item" id="equity">Equity</div>
      <div class="menu-item" id="derivatives">Derivatives</div>
      <div class="menu-item" id="share">Share</div>
      <div class="menu-item" id="market">Market</div>
      <div class="menu-item" id="total">Total</div>
    </div>
  </fieldset>

  <fieldset>
    <fieldset>
      <div class="menu-container">
```

```
<div class="menu-item-container" id="stock1-container">
  <div class="menu-item">
    <h6>NIFTY 50</h6>
    <h6>19725.05</h6>
    <h6>116.05 (59%)</h6>
  </div>
</div>
<div class="menu-item-container" id="stock2-container">
  <div class="menu-item">
    <h6>BITCOIN</h6>
    <h6>12345.05</h6>
    <h6>567.05 (59%)</h6>

  </div>
</div>
<div class="menu-item-container" id="stock3-container">
  <div class="menu-item">
    <h6>SHIBU 50</h6>
    <h6>34567.05</h6>
    <h6>345.05 (45%)</h6>

  </div>
</div>
<div class="menu-item-container" id="stock4-container">
  <div class="menu-item">
    <h6>ETHERNET 50</h6>
    <h6>2345.05</h6>
    <h6>123.05 (89%)</h6>

  </div>
</div>
```

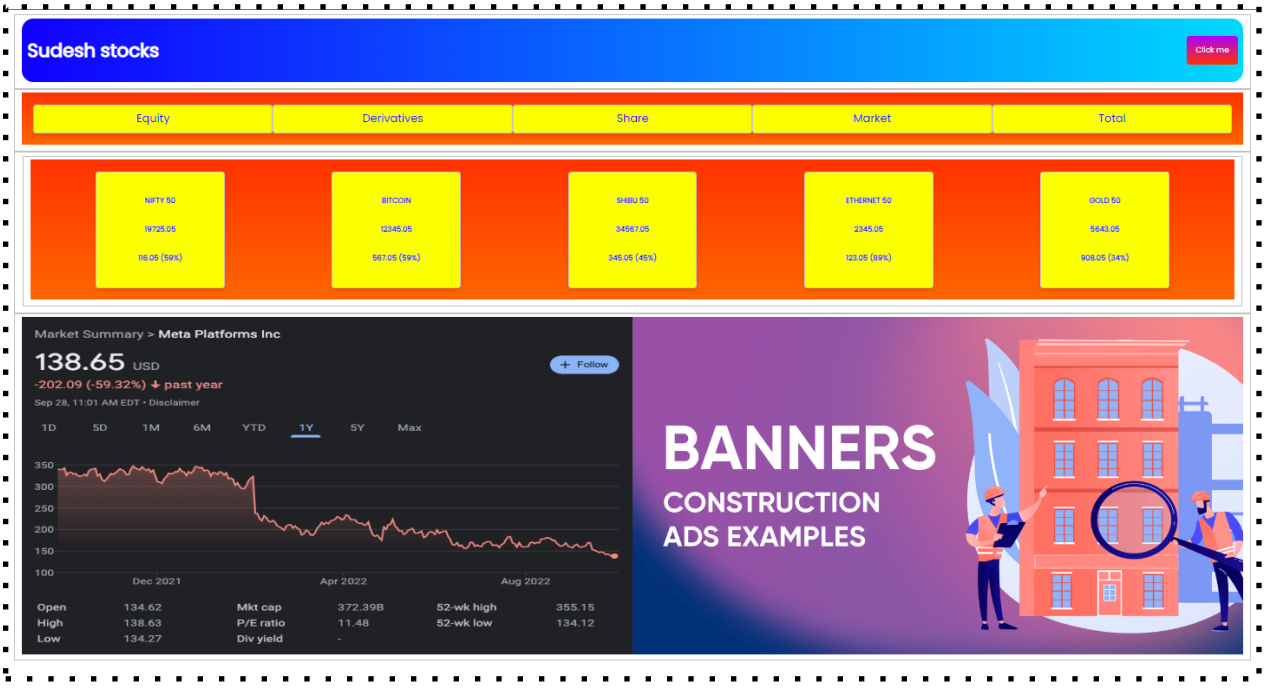
```
            <div class="menu-item-container" id="stock5-container">
              <div class="menu-item">
                <h6>GOLD 50</h6>
                <h6>5643.05</h6>
                <h6>908.05 (34%)</h6>

              </div>
            </div>
          </div>
        </fieldset>

</fieldset>

<fieldset>
  <div class="image-container">
    <img
src="https://lh4.googleusercontent.com/OVnEMWIFN_bwoyiEGYu2xsFdTuG9pF4sR4k4-
CTx9EN7zZ5Ee47dSWjaWwg4gLqchJoB7no18KEpxVXcaAAKvrVpuq_82lEoMj6D0Dg7gIxX
NoiZ7kbEgMN81OA8ySNJArYGirhqYCTB1M9B_vC-jKvaNDCOZqEnNBq-
j9OkZDCWRQhqMBZfSdr8vg" alt="Left Image" id="left-image">
    <img src="https://149362354.v2.pressablecdn.com/wp-content/uploads/2020/04/How-
to-Get-a-Low-Interest-Personal-Loan-in-India.png" alt="Image 1" id="image1">
    <img
src="https://viewst.com/static/4baa713bebe5f5fc8290365f5c99e50c/a8428/Construction
_banner_2.png" alt="Image 2" id="image2">
  </div>
</fieldset>
  <script>
    $(document).ready(function () {

      let image1Visible = true;
```

```
        $("#image1").show();
        $("#image2").hide();

        $(".image-container").click(function () {
            if (image1Visible) {
                $("#image1").hide();
                $("#image2").show();
            } else {
                $("#image1").show();
                $("#image2").hide();
            }
            image1Visible = !image1Visible;

  // When hovering over the header text, change the text color
  $("#header-text").hover(function () {
            // Change text color to a different color (e.g., red)
            $(this).css("color", "red");
        }, function () {
            // Revert text color to grey when the mouse leaves
            $(this).css("color", "White");
        });

        // Handle button click event
        $("#header-button").click(function () {
            // Show an alert with the message "Welcome to Sudesh stocks"
            alert("Welcome to Sudesh stocks");



        });

});
```

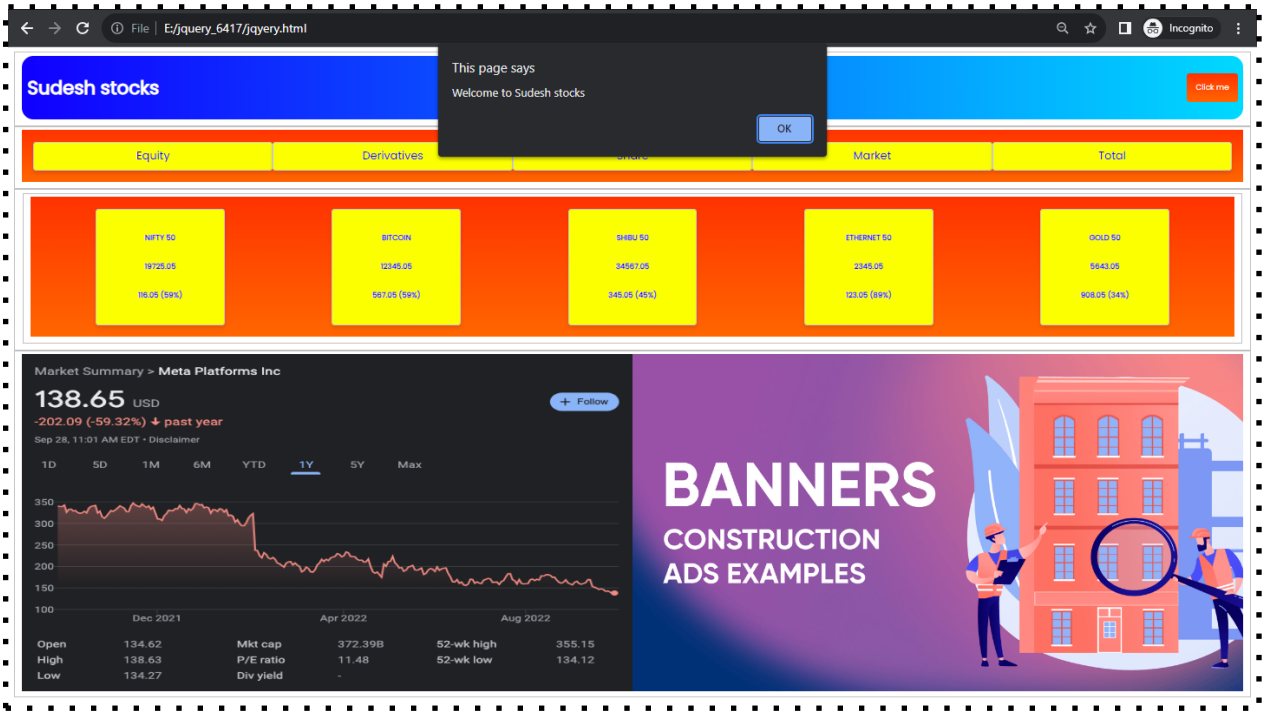
```
    });


  </script>
</body>
</html>
```

WebPage:

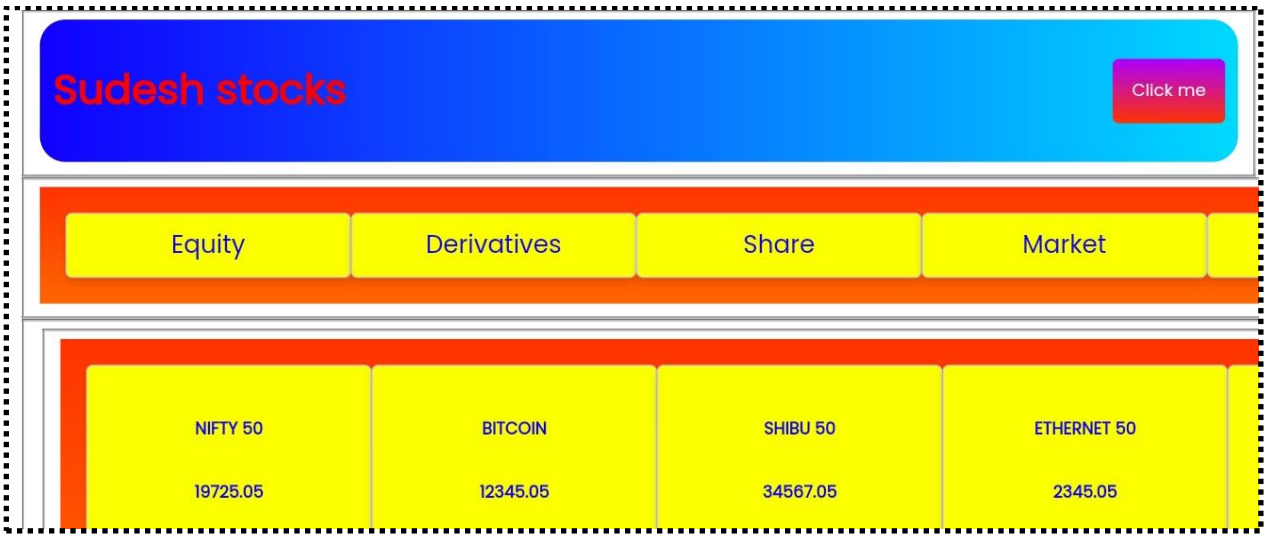
Advertisement Change on Click Event:



Alert Event on Click me Button:



Sudesh Stocks Color change to RED on Click:


Sudesh stocks
Click me
Equity
Derivatives
Share
Market
NIFTY 50
19725.05
BITCOIN
12345.05
SHIBU 50
34567.05
ETHERNET 50
2345.05