National Institute Of Technology, Surathkal, Karnataka

Big Data Technology- MongoDB

Major：**Computational Data Science**

Rishabh Kesarwani - 202CD023

Mittapalli Jyothi Sai Jeevan Reddy- 202CD015

Shubham Sherwade - 202CD026

Project Advisor： **Dr. Pushparaj Shetty D**

Due Date：12-04-2021

MongoDB

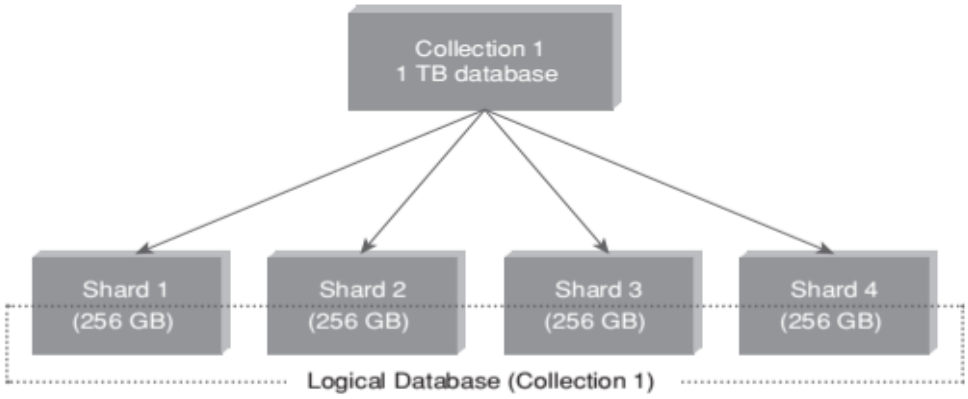
The relational database model has prevailed for decades. Of late a new kind of database is gaining ground in the enterprise called NoSQL (Not only SQL). Here , we will be exploring a NoSQL database called "MongoDB". We bring to you the features of MongoDB such as "Auto Sharding", "Replication", its "rich query language", "fast in-place update", etc.

Key Features

1) **Sharding** : Sharding is akin to horizontal scaling. It means that the large dataset is divided and distributed over multiple servers or shards. Each shard is an independent database and collectively they would constitute a logical database. The prime advantages of sharding are as follows:

1. Sharding reduces the amount of data that each shard needs to store and manage. For example, if the dataset was 1 TB in size and we were to distribute this over four shards, each shard would house just 256 GB data. Refer Figure below.As the cluster grows, the amount of data that each shard will store and manage will decrease.

2. Sharding reduces the number of operations that each shard handles. For example, if we were to insert data, the application needs to access only that shard which houses that data.

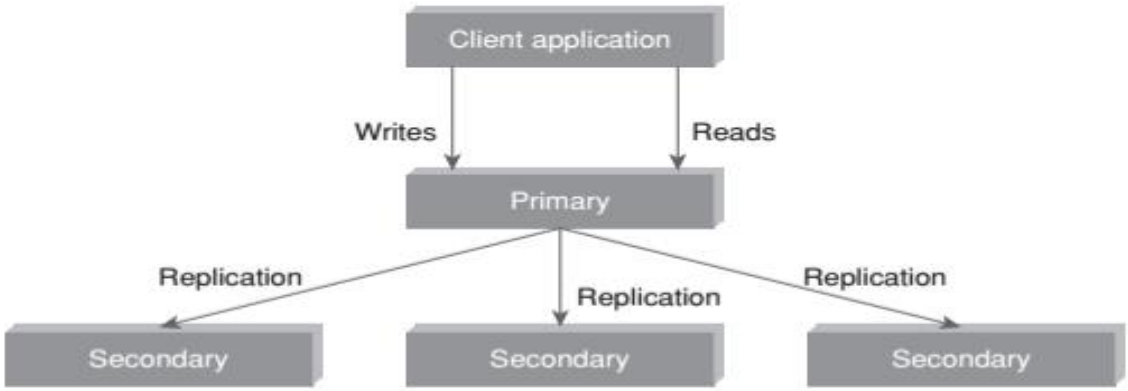


Process of Sharding in mongoDB

2) **Replication** :

Why replication? It provides data redundancy and high availability. It helps to recover from hardware failure and service interruptions. In MongoDB, the replica set has a single primary and several secondaries. Each write request from the client is directed to the primary.

The primary logs all write requests into its Oplog (operations log). The Oplog is then used by the secondary replica members to synchronize their data. This way there is strict adherence to consistency. Refer Figure below. The clients usually read from the primary. However, the client can also specify a read preference that will then direct the read operations to the secondary.



The process of Replication in MongoDB.

3) Rich Query Language

MongoDB supports a rich query language to support read and write operations (CRUD) as well as:

- Data Aggregation
- Text Search and Geospatial Queries

4) Updating Information In-Place

MongoDB updates the information in-place. This implies that it updates the data wherever it is available. It does not allocate separate space and the indexes remain unaltered. MongoDB is all for lazy-writes. It writes to the disk once every second. Reading and writing to disk is a slow operation as compared to reading and writing from memory. The fewer the reads and writes that we perform to the disk, the better is the performance. This makes MongoDB faster than its other competitors who write almost immediately to the disk. However, there is a tradeoff. MongoDB makes no guarantee that data will be stored safely on the disk.

Comparison with RDBMS

RDBMS	MongoDB
It is a relational database.	It is a non-relational and document-oriented database.
Not suitable for hierarchical data storage.	Suitable for hierarchical data storage.
It is vertically scalable i.e increasing RAM.	It is horizontally scalable i.e we can add more servers.
It has a predefined schema.	It has a dynamic schema.
It is quite vulnerable to SQL injection.	It is not affected by SQL injection.
It is row-based.	It is document-based.
It is slower in comparison with MongoDB.	It is almost 100 times faster than RDBMS.
Supports complex joins.	No support for complex joins.
It is column-based.	It is field-based.
It does not provide JavaScript client for querying.	It provides a JavaScript client for querying.
It supports SQL query language only.	It supports JSON query language along with SQL.

Terms Used in RDBMS and MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Record	Document
Columns	Fields/ Key Value pairs
Index	Index
Joins	Embedded documents
Primary Key identifier)	Primary key (_id is a

Let us look at how the statements are written in RDBMS and MongoDB

	RDBMS	MongoDB
Insert	Insert into Students (StudRollNo, StudName, Grade, Hobbies, DOJ) Values ('S101', 'Simon David', 'VII', 'Net Surfing', '10-Oct-2012')	db.Students.insert({_id:1, StudRollNo: 'S101', StudName: 'Simon David', Grade: 'VII', Hobbies: 'Net Surfing', DOJ: '10-Oct-2012'});
Update	Update Students set Hobbies ='Ice Hockey' where StudRollNo ='S101'	db.Students.update({StudRollNo: 'S101'}, {$set: {Hobbies : 'Ice Hockey'}})
	Update Students Set Hobbies ='Ice Hockey'	db.Students.update({},{$set: {Hobbies: 'Ice Hockey' }}, {multi:true})
Delete	Delete from Students where StudRollNo = 'S101'	db.Students.remove ({StudRollNo : 'S101'})
	Delete From Students	db.Students.remove({})
Select	Select * from Students	db.Students.find() db.Students.find().pretty()
	Select * from students where StudRollNo = 'S101'	db.Students.find({StudRollNo: 'S101'})
	Select StudRollNo, StudName, Hobbies from Students	db.Students.find({},{StudRollNo:1, StudName:1, Hobbies:1, _id:0})
	Select StudRollNo, StudName, Hobbies from Students where StudRollNo = 'S101'	db.Students.find({StudRollNo: 'S101'}, {StudRollNo : 1, StudName: 1, Hobbies : 1, _id:0})

Data Types in MongoDB The following are various data types in MongoDB

String	Must be UTF-8 valid. Most commonly used data type.
Integer	Can be 32-bit or 64-bit (depends on the server).
Boolean	To store a true/false value.
Double	To store floating point (real values).
Min/Max keys	To compare a value against the lowest or highest BSON
Arrays	To store arrays or list or multiple values into one key.
Timestamp	To record when a document has been modified or added
Null value.	To store a NULL value. A NULL is a missing or unknown
Date	To store the current date or time in Unix time format. One can create object of date and pass day, month and year to it.
Object ID	To store the document's id.
Binary data	To store binary data (images, binaries, etc.).
Code	To store javascript code into the document.
Regular expression	To store regular expression.

Some Common MongoDB commands and their usage details

1) In CommandLine -> **cd C:\Program Files\MongoDB\Server\4.4\bin**

To reach the location where the mongo application is present on which you would perform your operations on database

2) **-> mongo**

To connect to database

3) -> show dbs;

To show the database

4) ->exit

To exit the database

5) ->cls

To clear the screen

6) -> mongoimport.exe --db collection_name examples.json

To import the examples.json file into the mongo

7) ->mongo

To run the mongo application

8) ->show db

To see which are the databases present in the system

9) -> use database_name

To use the cooker database or to create a new data base. Unless we don't add somethong to the database, it will not show when we use show dbs command

10) ->db

To check which is the curret active database

11) ->doc2={"title":"Dosa","Taste":"Cripsy and yummy",people_serve:4}

-> db.collection_name.insertOne(doc2)

To insert the document in the database

12)**db.collection_name.insertOne({**

... "name": "Hitesh",

... "email": "hitesh@hiteshchoudhary.com",

... "contact": "9999999999",

... "courseCount": 4,

... "isVerified": true

... })

To insert the document in the MongoDB Database

13)->db.collection_name.deleteOne({feature_name:"unique_attribute"})

To delete one element in the MongoDB database

14)->**db.collection_name.deleteMany({})**

This would delete the every document present in the collection

15)->db.collection_name.deleteMany({"City":"Prayagraj"})

To delete where the filtering attribute is city:Prayagraj

16)->**db.collection_name.updateOne({name:"Rishabh"},{$set :{"Attendence":5}})**

To update one data in the collection

17)->**db.studentData.updateMany({"isVerified":true},{$set :{"City":"Prayagraj"}})**

To update many things they atleast should have something common in them i.e. isVerified:true is common in some documents that's why we were able to update some thing from studentData

18)->db.studentData.find({courseCount:{$gt:1}}).pretty()

To find the documents where the greater than 1 coursecount is there

19)->db.studentData.find({},{email:1,_id:0})

To print the coulumns with only email other details are not desired

20)->db.studentData.find({},{email:1})

To print the columns where _id will also be displayed as we have not specifed it to 0. By deafult _id is always 1 so unless we specify that it wont be hide.

21)>db.studentData.find({},{email:1,_id:0}).toArray()

To convert the email that we were getting into the array which is more visually applealing

22)->db.studentData.updateMany({},{$set:{profilepic:{small:50, mid:100, large:200}}})

To create the new attribute profilepic with various datapoints

23)>db.studentData.updateOne({"name":"Hitesh"},{$set :{"profilepic.mid":500}})

To identify the document with name Hitesh and then changing the profilepicmid value to 500

24) >**db.studentData.updateOne({"name":"Hitesh"},{$set :{lastlogin:["Monday","Tuesday","Wednesday"]}})**

To update the document with collection with an array of lostlogin details

25)->db.studentData.findOne({name:"Hitesh"}).lastlogin

To access the array of lastlogin

26) →**db.dropDatabase()**

To drop the selected database

27) →**db.collection.drop()**

To drop a collection from database

28)→**db.studentData.count()**

To find the number of documents in the studentData collection

29)→**db.studentData.find().sort({name:1})**

To sort the documents from the studentData collection in ascending order of name

30)→**db.studentData.find().skip(2)**

To skip the first 2 documents from the studentData collection

31)→**db.studetnData.find().limit(3)**

To display only first 3 documents from the studentData collection

Some MongoDB examples and their output

Example 1-

Documents inserts in the mongoDB database

->{ "_id" : 1, "item" : "abc1", description: "product 1", qty: 300 }

{ "_id" : 2, "item" : "abc2", description: "product 2", qty: 200 }

{ "_id" : 3, "item" : "xyz1", description: "product 3", qty: 250 }

{ "_id" : 4, "item" : "VWZ1", description: "product 4", qty: 300 }

{ "_id" : 5, "item" : "VWZ2", description: "product 5", qty: 180 }

->**db.inventory.aggregate(**

[{ $project: { item: 1,result: { $or: [{ $gt: ["$qty", 250] }, { $lt: ["$qty", 200] }] }}}])

OUTPUT:

{ "_id" : 1, "item" : "abc1", "result" : true }

{ "_id" : 2, "item" : "abc2", "result" : false }

{ "_id" : 3, "item" : "xyz1", "result" : false }

{ "_id" : 4, "item" : "VWZ1", "result" : true }

{ "_id" : 5, "item" : "VWZ2", "result" : true }

Example 2-

```
db.sales.insertMany([

  { "_id" : 1, "item" : "abc", "price" : NumberDecimal("10"), "quantity" : NumberInt("2"), "date" : ISODate("2014-03-01T08:00:00Z") },

  { "_id" : 2, "item" : "jkl", "price" : NumberDecimal("20"), "quantity" : NumberInt("1"), "date" : ISODate("2014-03-01T09:00:00Z") },

  { "_id" : 3, "item" : "xyz", "price" : NumberDecimal("5"), "quantity" : NumberInt( "10"), "date" : ISODate("2014-03-15T09:00:00Z") },

  { "_id" : 4, "item" : "xyz", "price" : NumberDecimal("5"), "quantity" :  NumberInt("20") , "date" : ISODate("2014-04-04T11:21:39.736Z") },

  { "_id" : 5, "item" : "abc", "price" : NumberDecimal("10"), "quantity" : NumberInt("10") , "date" : ISODate("2014-04-04T21:23:13.331Z") },

  { "_id" : 6, "item" : "def", "price" : NumberDecimal("7.5"), "quantity": NumberInt("5" ) , "date" : ISODate("2015-06-04T05:08:13Z") },

  { "_id" : 7, "item" : "def", "price" : NumberDecimal("7.5"), "quantity": NumberInt("10") , "date" : ISODate("2015-09-10T08:43:00Z") },

  { "_id" : 8, "item" : "abc", "price" : NumberDecimal("10"), "quantity" : NumberInt("5" ) , "date" : ISODate("2016-02-06T20:20:13Z") },

])
```

```
->db.sales.aggregate([

  // First Stage

  {$match : { "date": { $gte: new ISODate("2014-01-01"), $lt: new ISODate("2015-01-01") } }},

  // Second Stage

  {$group : { _id : { $dateToString: { format: "%Y-%m-%d", date: "$date" } },totalSaleAmount: { $sum: { $multiply: [ "$price", "$quantity" ] } },
averageQuantity: { $avg: "$quantity" },count: { $sum: 1 }}},

  // Third Stage
```

```
  {$sort : { totalSaleAmount: -1 }}

 ])
```

OUTPUT:-

```
{ "_id" : "2014-04-04", "totalSaleAmount" : NumberDecimal("200"), "averageQuantity" :
15, "count" : 2 }
{ "_id" : "2014-03-15", "totalSaleAmount" : NumberDecimal("50"), "averageQuantity" : 10,
"count" : 1 }
{ "_id" : "2014-03-01", "totalSaleAmount" : NumberDecimal("40"), "averageQuantity" : 1.5,
"count" : 2 }
```

First Stage:

The $match stage filters the documents to only pass documents from the year 2014 to the next stage.

Second Stage:

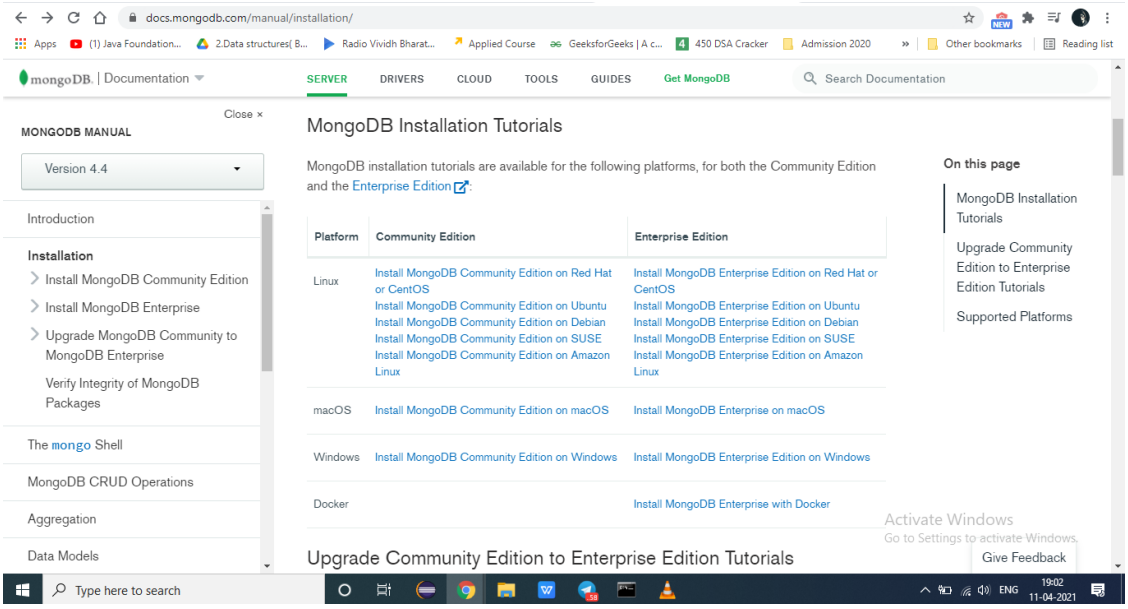
The $group stage groups the documents by date and calculates the total sale amount, average quantity, and total count of the documents in each group.

Third Stage:

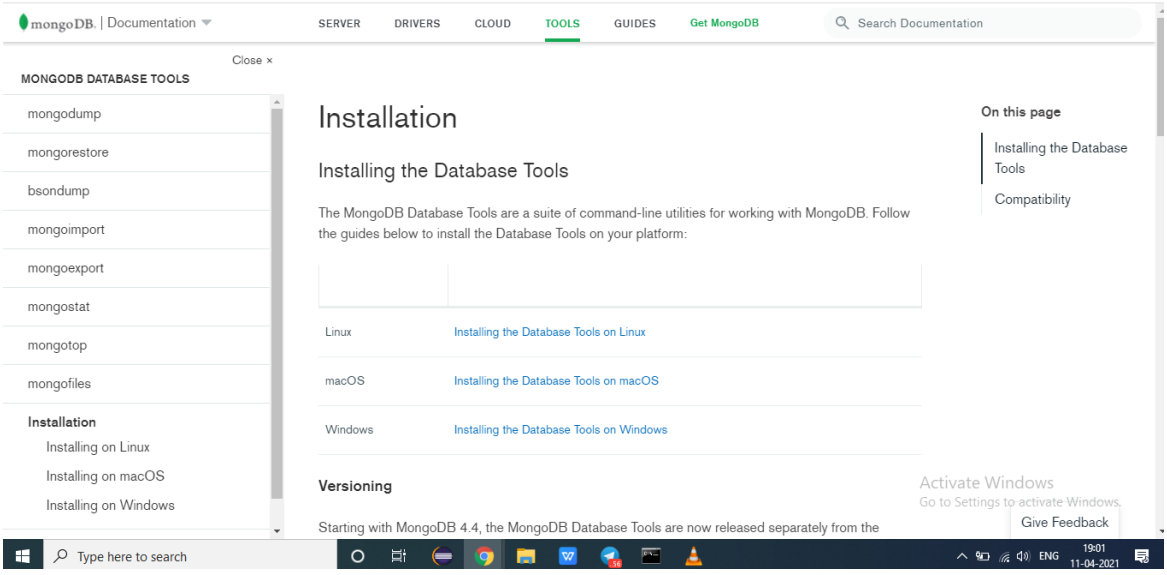
The $sort stage sorts the results by the total sale amount for each group in descending order.

MongoDB Installation Guide

Step 1) [MongoDb Installation for Windows, Mac and Linux] . Visit the above site to install MongoDB 4.4 in your device.



Step2)[MongoDB Database Supplementary Files] . Unzip these files and paste in the bin folder of the MongoDB



Step3) To check MongoDB is installed or not. Go to Terminal or Command Prompt .

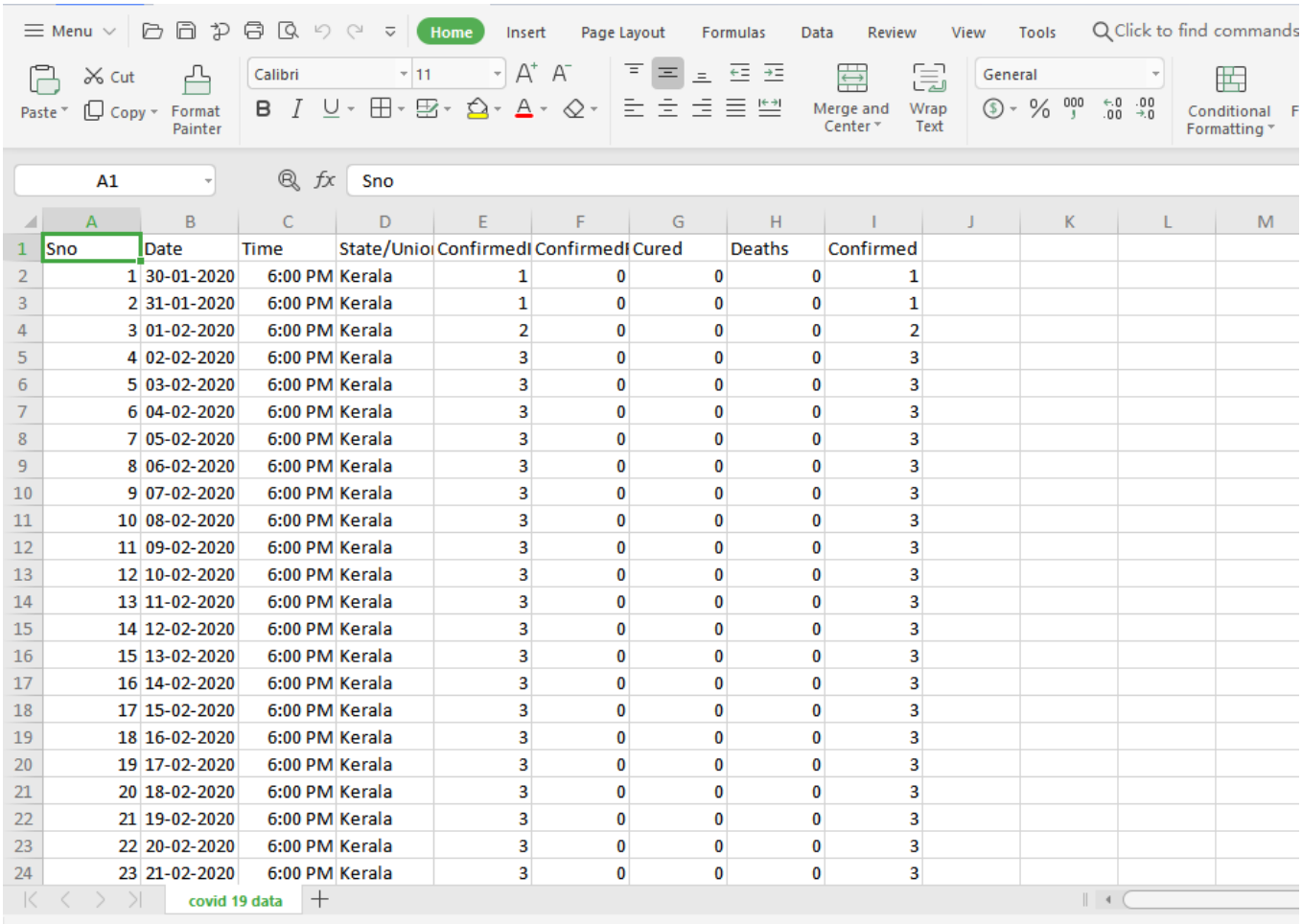
Go to the bin folder of MongoDB in CMD.

-> cd C:\Address_of_bin_MongoDB

Type -> mongo (To run mongo application)

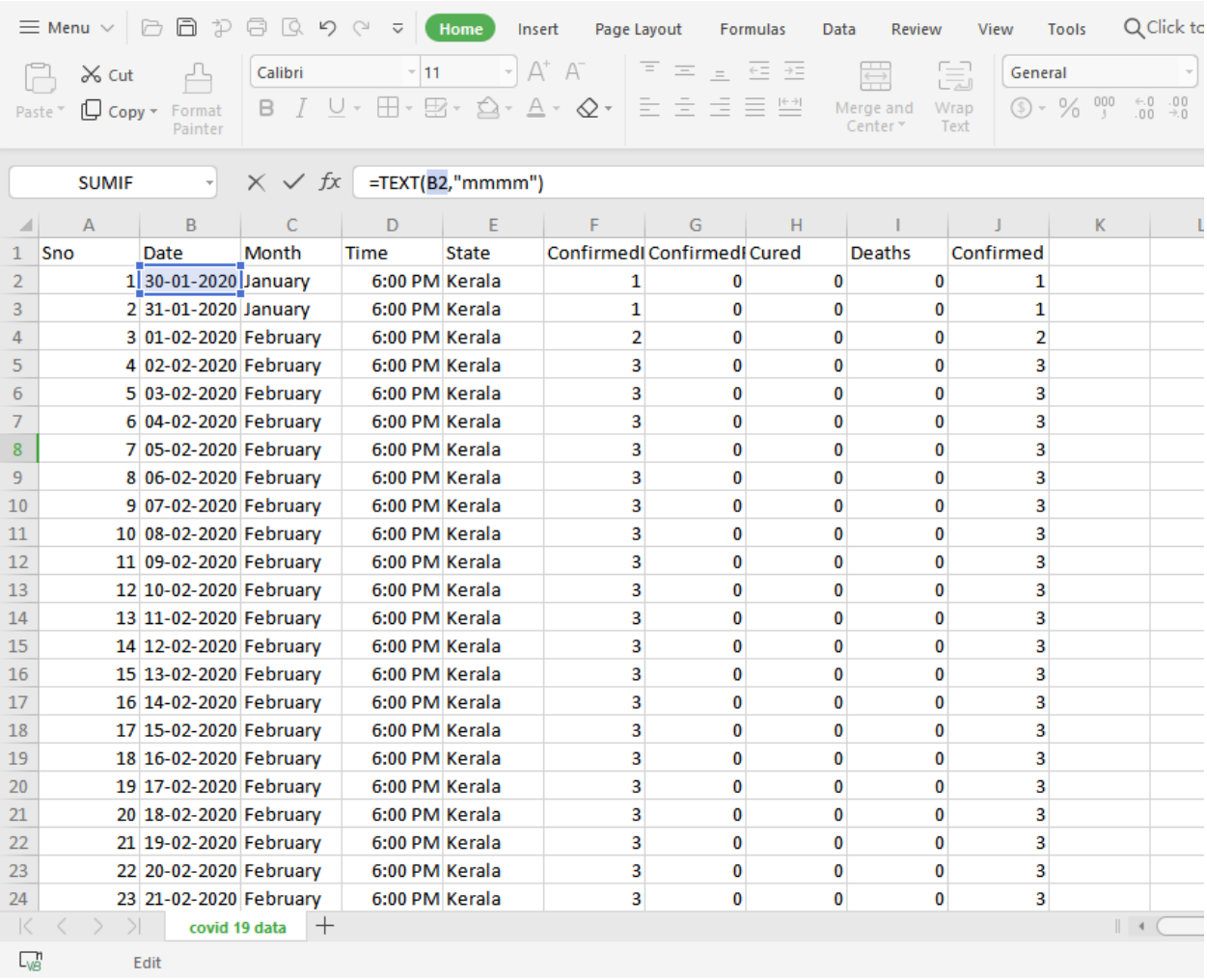
CSV File Preprocessing Before Import

Before Preprocessing as the dates are in mm-dd-yyyy format and in question we want to perform operation on months



Sno	Date	Time	State/Unio	ConfirmedI	ConfirmedF	Cured	Deaths	Confirmed
1	30-01-2020	6:00 PM	Kerala	1	0	0	0	1
2	31-01-2020	6:00 PM	Kerala	1	0	0	0	1
3	01-02-2020	6:00 PM	Kerala	2	0	0	0	2
4	02-02-2020	6:00 PM	Kerala	3	0	0	0	3
5	03-02-2020	6:00 PM	Kerala	3	0	0	0	3
6	04-02-2020	6:00 PM	Kerala	3	0	0	0	3
7	05-02-2020	6:00 PM	Kerala	3	0	0	0	3
8	06-02-2020	6:00 PM	Kerala	3	0	0	0	3
9	07-02-2020	6:00 PM	Kerala	3	0	0	0	3
10	08-02-2020	6:00 PM	Kerala	3	0	0	0	3
11	09-02-2020	6:00 PM	Kerala	3	0	0	0	3
12	10-02-2020	6:00 PM	Kerala	3	0	0	0	3
13	11-02-2020	6:00 PM	Kerala	3	0	0	0	3
14	12-02-2020	6:00 PM	Kerala	3	0	0	0	3
15	13-02-2020	6:00 PM	Kerala	3	0	0	0	3
16	14-02-2020	6:00 PM	Kerala	3	0	0	0	3
17	15-02-2020	6:00 PM	Kerala	3	0	0	0	3
18	16-02-2020	6:00 PM	Kerala	3	0	0	0	3
19	17-02-2020	6:00 PM	Kerala	3	0	0	0	3
20	18-02-2020	6:00 PM	Kerala	3	0	0	0	3
21	19-02-2020	6:00 PM	Kerala	3	0	0	0	3
22	20-02-2020	6:00 PM	Kerala	3	0	0	0	3
23	21-02-2020	6:00 PM	Kerala	3	0	0	0	3

After Preprocessing , Here we have added a extra column I.e. **Months which is desired attribute on which we would perform query**



	A	B	C	D	E	F	G	H	I	J	K
1	Sno	Date	Month	Time	State	ConfirmedI	ConfirmedF	Cured	Deaths	Confirmed	
2	1	30-01-2020	January	6:00 PM	Kerala	1	0	0	0	1	
3	2	31-01-2020	January	6:00 PM	Kerala	1	0	0	0	1	
4	3	01-02-2020	February	6:00 PM	Kerala	2	0	0	0	2	
5	4	02-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
6	5	03-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
7	6	04-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
8	7	05-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
9	8	06-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
10	9	07-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
11	10	08-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
12	11	09-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
13	12	10-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
14	13	11-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
15	14	12-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
16	15	13-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
17	16	14-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
18	17	15-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
19	18	16-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
20	19	17-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
21	20	18-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
22	21	19-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
23	22	20-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	
24	23	21-02-2020	February	6:00 PM	Kerala	3	0	0	0	3	

First make the extra column beside of B column and then use the formula **=TEXT(B2,"mmmm")**

To convert the mm-dd-yyyy format into months.

Now our CSV file is ready to be Imported

Importing The CSV file and Creating the DataBase

Step 1) Import the csv file into the mongo. Before Importing paste the CSV file in the bin folder of MongoDB

->mongoimport -d BigData -c covid --type csv --headerline --file covidData.csv

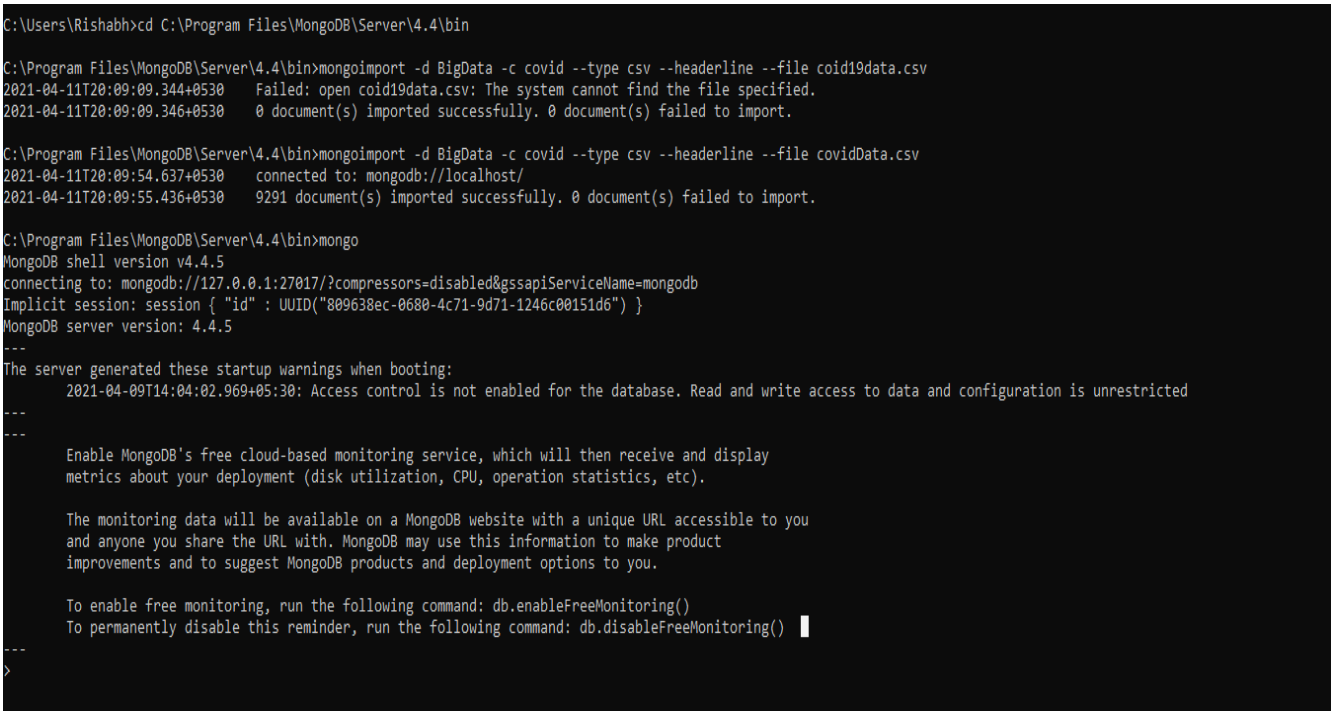
Step2) To use the BigData database which we have created while importing the csv file

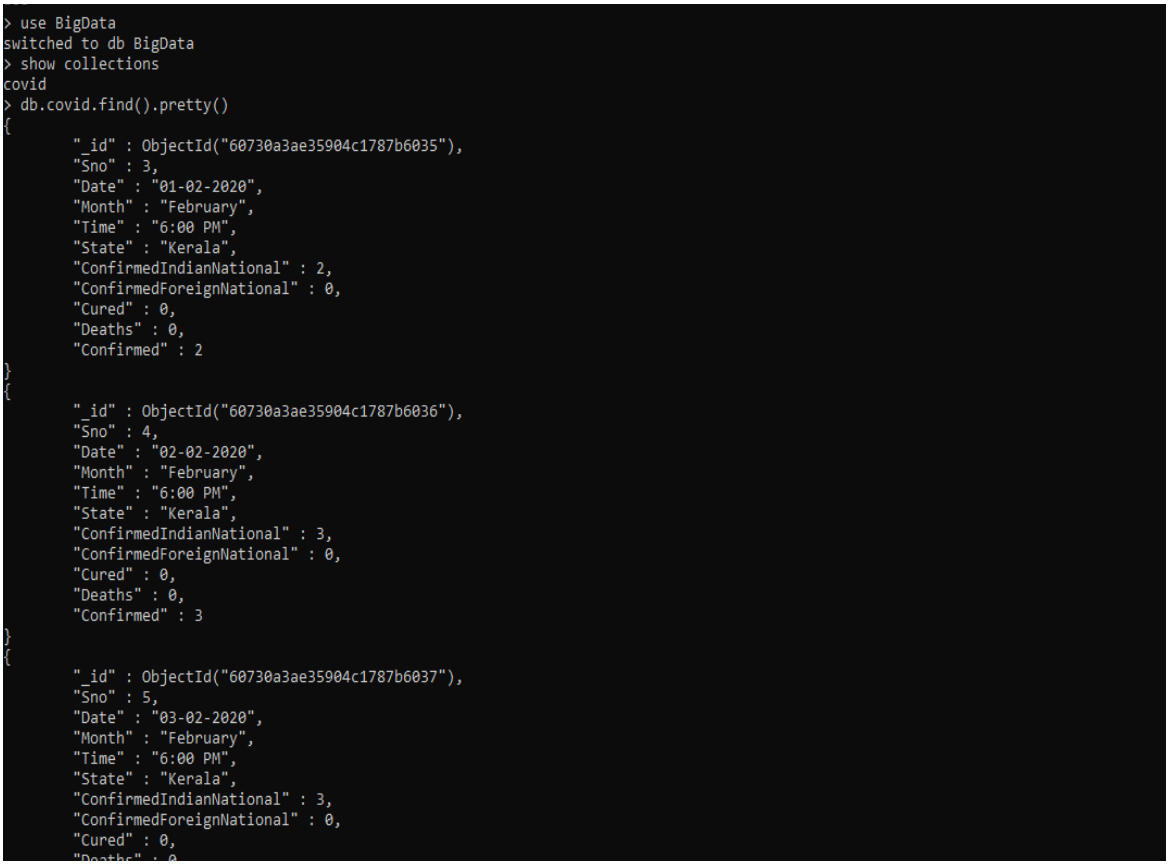
-> use BigData

Step3) Now to check whether your data is imported sucessfully type

->**db.covid.find().pretty()**

Step4)Now since our data is sucessfully imported so we are ready to perform query to solve our desired problem

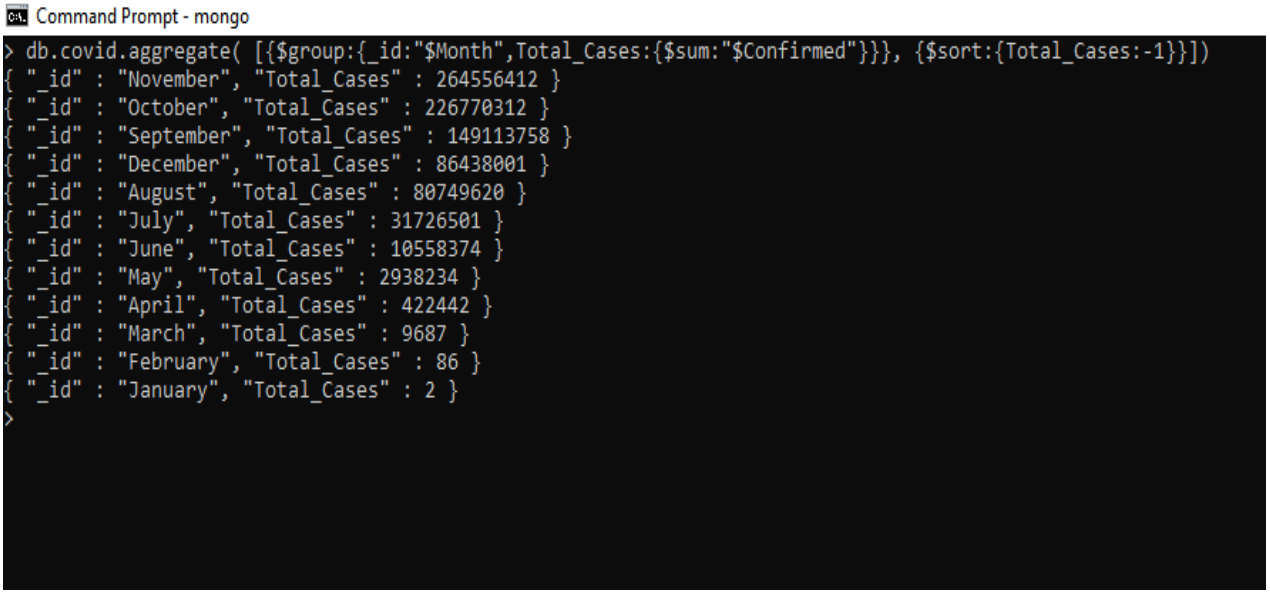




```
> use BigData
switched to db BigData
> show collections
covid
> db.covid.find().pretty()
{
        "_id" : ObjectId("60730a3ae35904c1787b6035"),
        "Sno" : 3,
        "Date" : "01-02-2020",
        "Month" : "February",
        "Time" : "6:00 PM",
        "State" : "Kerala",
        "ConfirmedIndianNational" : 2,
        "ConfirmedForeignNational" : 0,
        "Cured" : 0,
        "Deaths" : 0,
        "Confirmed" : 2
}
{
        "_id" : ObjectId("60730a3ae35904c1787b6036"),
        "Sno" : 4,
        "Date" : "02-02-2020",
        "Month" : "February",
        "Time" : "6:00 PM",
        "State" : "Kerala",
        "ConfirmedIndianNational" : 3,
        "ConfirmedForeignNational" : 0,
        "Cured" : 0,
        "Deaths" : 0,
        "Confirmed" : 3
}
{
        "_id" : ObjectId("60730a3ae35904c1787b6037"),
        "Sno" : 5,
        "Date" : "03-02-2020",
        "Month" : "February",
        "Time" : "6:00 PM",
        "State" : "Kerala",
        "ConfirmedIndianNational" : 3,
        "ConfirmedForeignNational" : 0,
        "Cured" : 0,
        "Deaths" : 0
```

Ques 1) **Filter the month in which heighest people are get infected to Covid-19 virus?**

->db.covid.aggregate([{$group:{_id:"$Month",Total_Cases:{$sum:"$Confirmed"}}}, {$sort:{Total_Cases:-1}}])



We can observed that in the Month of **November** the maximum no. Of Total case came.

Ques 2) Obtain state in which survival rate is high.

-> db.covid.aggregate([{$group:{_id:"$State", Cured:{$sum:"$Cured"},Confirmed:{$sum:"$Confirmed"}}},{$project:{_id:"$_id",survival_rate:{$divide:["$Cured","$Confirmed"]}}}, {$sort:{survival_rate:1}}]).pretty()

```
> db.covid.aggregate( [{$group:{_id:"$State", Cured:{$sum:"$Cured"},Confirmed:{$sum:"$Confirmed"}}},{$project:{_id:"$_id",survival_rate:{$divide:["$Cured","$Confirmed"]
}}}, {$sort:{survival_rate:-1}}]).pretty()
{ "_id" : "Punjab***", "survival_rate" : 0.9274634614700757 }
{ "_id" : "Chandigarh***", "survival_rate" : 0.9197365055001279 }
{ "_id" : "Maharashtra***", "survival_rate" : 0.917730183647828 }
{
        "_id" : "Dadra and Nagar Haveli and Daman and Diu",
        "survival_rate" : 0.9165327675771805
}
{ "_id" : "Bihar", "survival_rate" : 0.9078906057560079 }
{ "_id" : "Tamil Nadu", "survival_rate" : 0.8967739523616147 }
{ "_id" : "Odisha", "survival_rate" : 0.8967045734037459 }
{ "_id" : "Andhra Pradesh", "survival_rate" : 0.896366107422855 }
{
        "_id" : "Andaman and Nicobar Islands",
        "survival_rate" : 0.8902557254512042
}
{ "_id" : "Delhi", "survival_rate" : 0.8774177389482465 }
{ "_id" : "Haryana", "survival_rate" : 0.8748806116404771 }
{ "_id" : "Telengana", "survival_rate" : 0.8744670865988973 }
{ "_id" : "Goa", "survival_rate" : 0.8744322141547343 }
{ "_id" : "Jharkhand", "survival_rate" : 0.8740487499658246 }
{ "_id" : "Assam", "survival_rate" : 0.8717882085910155 }
{ "_id" : "West Bengal", "survival_rate" : 0.870046009456807 }
{ "_id" : "Uttar Pradesh", "survival_rate" : 0.8637364518552308 }
{ "_id" : "Madhya Pradesh", "survival_rate" : 0.8593871734137332 }
{ "_id" : "Rajasthan", "survival_rate" : 0.8577917549525238 }
{ "_id" : "Punjab", "survival_rate" : 0.8552693998476679 }
Type "it" for more
> S
```

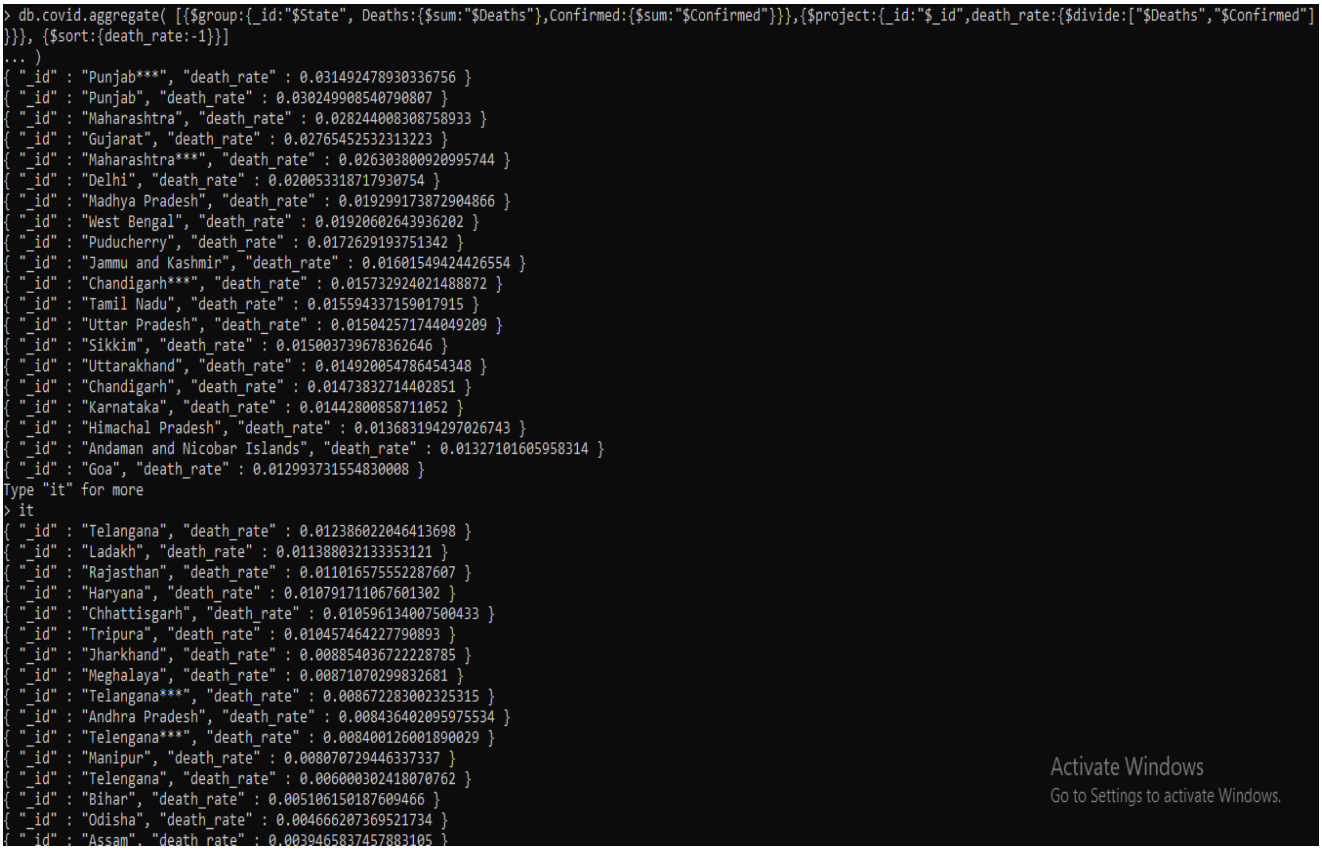
As we have sorted the data in the descending order . so we can oberve the top 20 states with the highest survival rate.

Survival rate=cured/confirmed

Punjab is the state with the highest survival rate

Ques 3)Check for state in which death rate is more than 1%

–>db.covid.aggregate([{$group:{_id:"$State", Deaths:{$sum:"$Deaths"},Confirmed:{$sum:"$Confirmed"}}},{$project:{_id:"$_id",death_rate:{$divide:["$Deaths","$Confirmed"]}}}, {$sort:{death_rate:-1}}]



Death rate = deaths/confirmed

As we have have sorted the data in the descending .Here we have to multiply by each death rate by 100 to see whether the states have death rate greater than 1%.

We can observe that from Punjab toTripura death rate is very high .