Department of Mathematical and Computational Science

Big Data Technology - MongoDB

Facuty Advisor- **Dr Pushparaj Shetty D**

Mittapalli Jyothi Sai Jeevan Reddy- 202CD015

Rishabh Kesarwani-202CD023

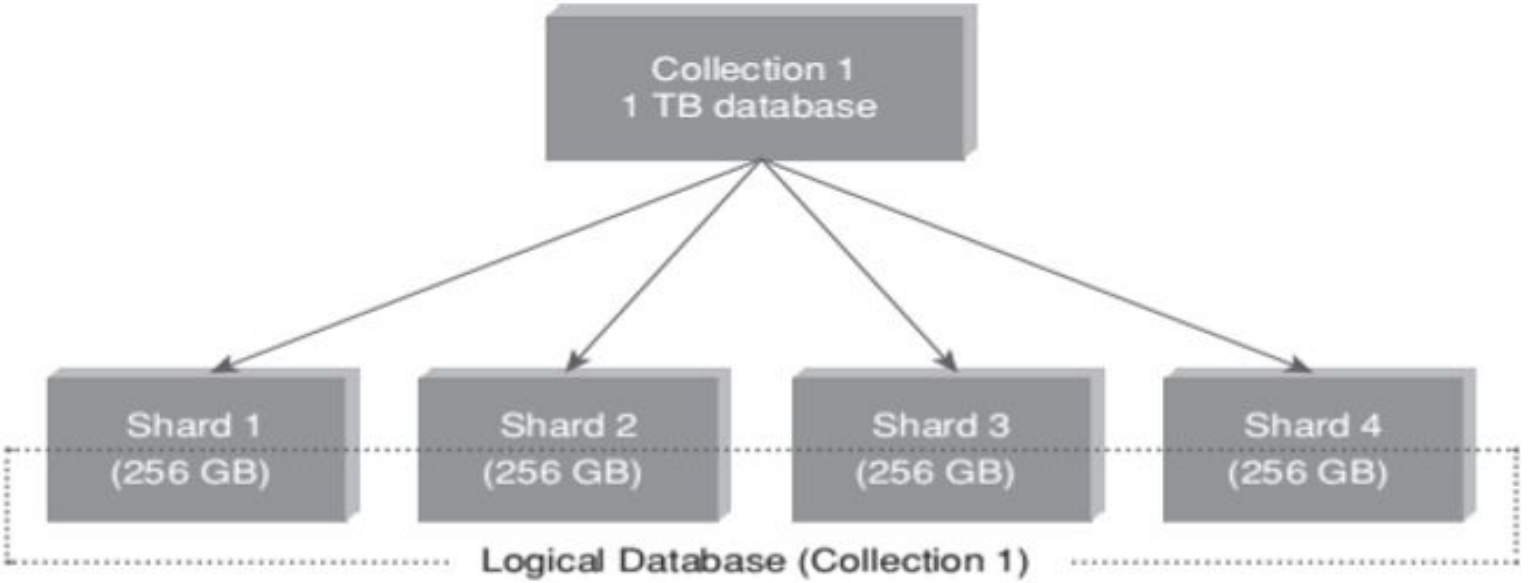
Shubham Sherwade- 202CD026

What is MongoDB ?

Key Features :

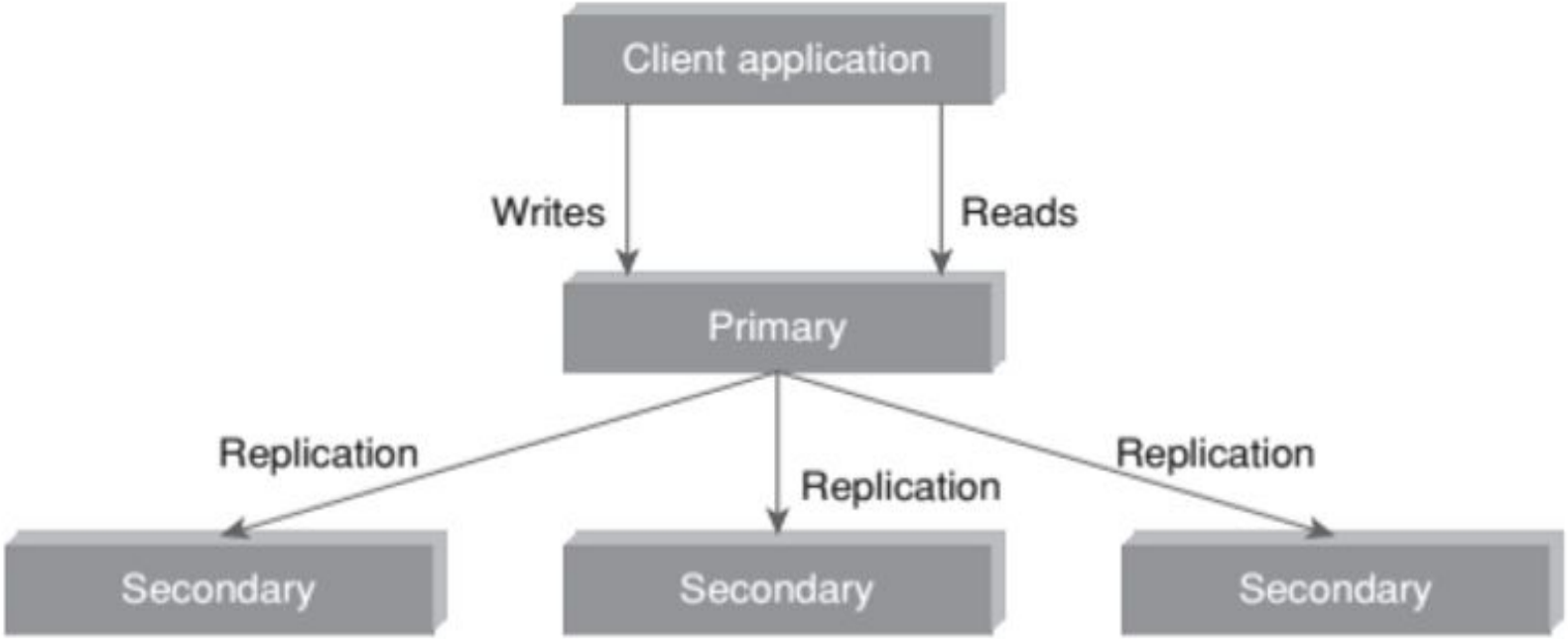
1) Sharding
2) Replication
3) Rich Query Language
4) Updating Information In-Place

1) Sharding :



Process of Sharding in mongoDB

2) Replication :



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.

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	Primary key (_id is a identifier)

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	To store a NULL value. A NULL is a missing or unknown value.
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.

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)

Some Common MongoDB commands and their usage details

→ **cd C:\Program Files\MongoDB\Server\4.4\bin**

-> mongo

-> show dbs;

-> show collections

->exit

->cls

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

->mongo

-> use database_name

-> show collections

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

-> db.collection_name.insertOne(doc2)

```
→ db.collection_name.insertOne({
      ... "name": "Hitesh",
      ... "email": "hitesh@hiteshchoudhary.com",
      ... "contact": "9999999999",
      ... "courseCount": 4,
      ... "isVerified": true
      ... })

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

->db.collection_name.deleteMany({})

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

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

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

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

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

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

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

```
->db.studentData.find().pretty()

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

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

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

->db.studentData.count()

->db.studentData.find().sort({name:1})
```

->db.studentData.find().skip(2)

->db.studetnData.find().limit(3)

->db.dropDatabase()

->db.dropDatabase()

Some complex Query commands in MongoDB

```
->db.orders.aggregate([
   { $match: { status: "A" } },
   { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }
])
->db.inventory.find( { $or: [ { quantity: { $lt: 20 } }, { price: 10 } ] } )
```

_>{ "_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] }] }}}])

{ "_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 }

Calculate the count, sum and average

```
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 }}

 ])
```

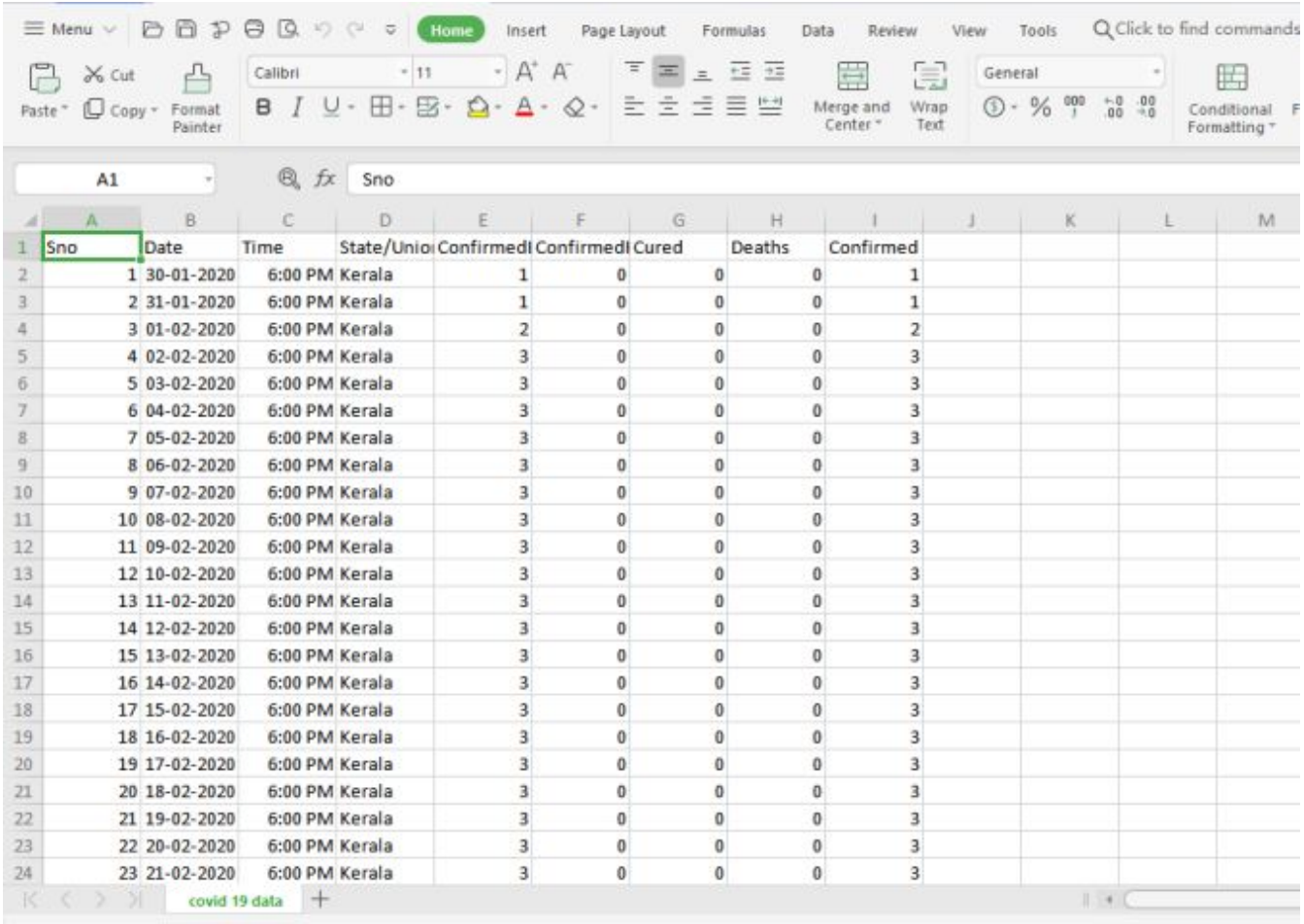
OUTPUT :

{ "_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 }

CSV File Preprocessing Before Importing



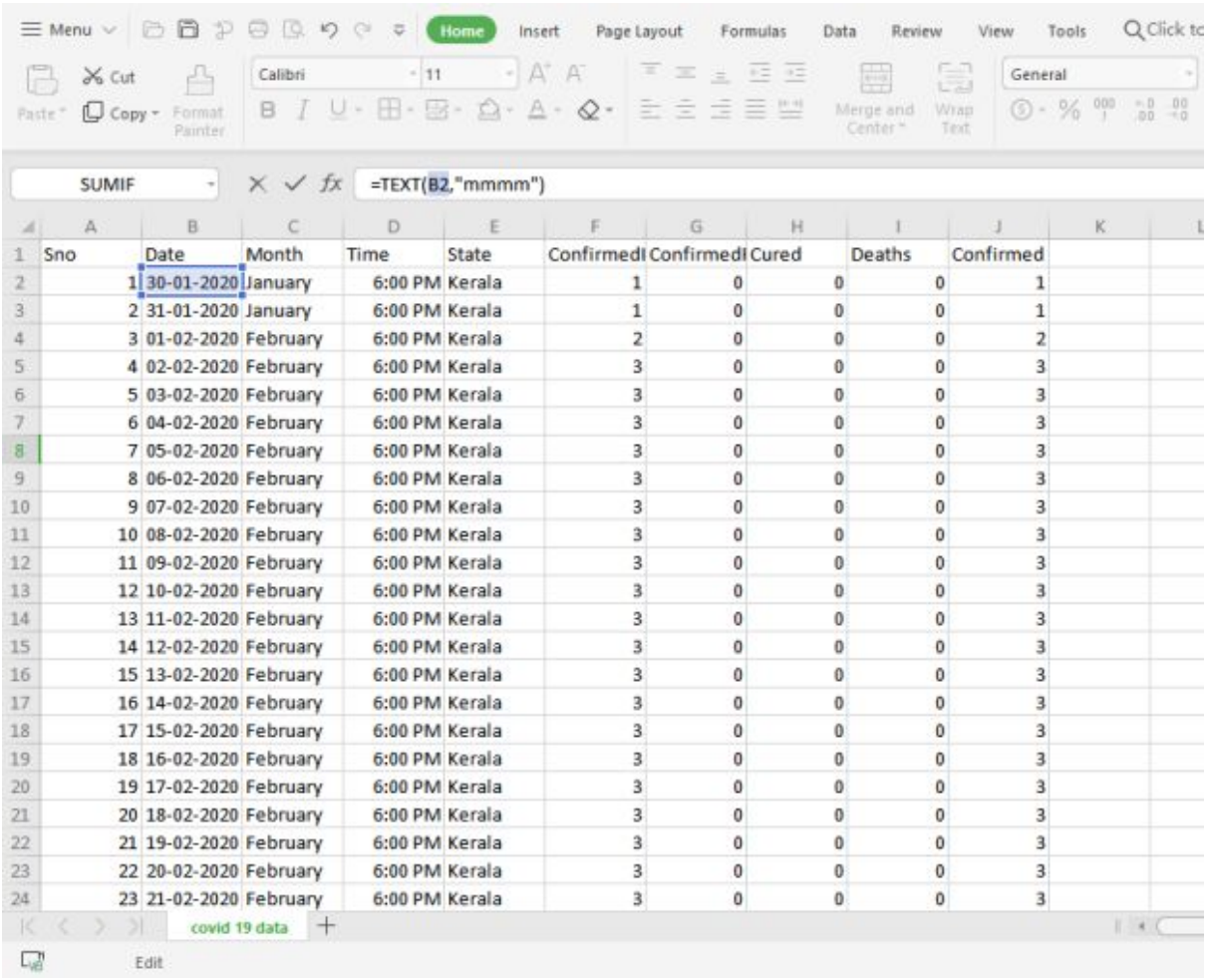
Menu Home Insert Page Layout Formulas Data Review View Tools Click to find commands

Paste Cut Copy Format Painter Calibri 11 Merge and Center Wrap Text General Conditional Formatting

A1 Sno

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

covid 19 data



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

Importing The CSV file and Creating the DataBase

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

->use BigData

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

```
C:\Users\Rishabh>cd C:\Program Files\MongoDB\Server\4.4\bin

C:\Program Files\MongoDB\Server\4.4\bin>mongoimport -d BigData -c covid --type csv --headerline --file coid19data.csv
2021-04-11T20:09:09.344+0530    Failed: open coid19data.csv: The system cannot find the file specified.
2021-04-11T20:09:09.346+0530    0 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Server\4.4\bin>mongoimport -d BigData -c covid --type csv --headerline --file covidData.csv
2021-04-11T20:09:54.637+0530    connected to: mongodb://localhost/
2021-04-11T20:09:55.436+0530    9291 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("809638ec-0680-4c71-9d71-1246c00151d6") }
MongoDB server version: 4.4.5
---
The server generated these startup warnings when booting:
        2021-04-09T14:04:02.969+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

```
> 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
```

Question 1) Filter the month in which highest people are get infected to Covid-19 virus?

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

Command Prompt - mongo

```
> db.covid.aggregate( [{$group:{_id:"$Month",Total_Cases:{$sum:"$Confirmed"}}}, {$sort:{Total_Cases:-1}}])
{ "_id" : "November", "Total_Cases" : 264556412 }
{ "_id" : "October", "Total_Cases" : 226770312 }
{ "_id" : "September", "Total_Cases" : 149113758 }
{ "_id" : "December", "Total_Cases" : 86438001 }
{ "_id" : "August", "Total_Cases" : 80749620 }
{ "_id" : "July", "Total_Cases" : 31726501 }
{ "_id" : "June", "Total_Cases" : 10558374 }
{ "_id" : "May", "Total_Cases" : 2938234 }
{ "_id" : "April", "Total_Cases" : 422442 }
{ "_id" : "March", "Total_Cases" : 9687 }
{ "_id" : "February", "Total_Cases" : 86 }
{ "_id" : "January", "Total_Cases" : 2 }
>
```

Question 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","$Confi
rmed"]}}}, {$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
> it
```

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"}}},{$proj
ect:{_id:"$_id",death_rate:{$divide:["$Deaths","$Confirmed"]}}},
{$sort:{death_rate:-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}}]
... )
{ "_id" : "Punjab***", "death_rate" : 0.031492478930336756 }
{ "_id" : "Punjab", "death_rate" : 0.030249908540790007 }
{ "_id" : "Maharashtra", "death_rate" : 0.028244008308758933 }
{ "_id" : "Gujarat", "death_rate" : 0.02765452532313223 }
{ "_id" : "Maharashtra***", "death_rate" : 0.026303800920995744 }
{ "_id" : "Delhi", "death_rate" : 0.020053318717930754 }
{ "_id" : "Madhya Pradesh", "death_rate" : 0.019299173872904866 }
{ "_id" : "West Bengal", "death_rate" : 0.01920602643936202 }
{ "_id" : "Puducherry", "death_rate" : 0.0172629193751342 }
{ "_id" : "Jammu and Kashmir", "death_rate" : 0.01601549424426554 }
{ "_id" : "Chandigarh***", "death_rate" : 0.015732924021488872 }
{ "_id" : "Tamil Nadu", "death_rate" : 0.015594337159017915 }
{ "_id" : "Uttar Pradesh", "death_rate" : 0.015042571744049209 }
{ "_id" : "Sikkim", "death_rate" : 0.015003739678362646 }
{ "_id" : "Uttarakhand", "death_rate" : 0.014920054786454348 }
{ "_id" : "Chandigarh", "death_rate" : 0.014738327144028 51 }
{ "_id" : "Karnataka", "death_rate" : 0.01442800858711052 }
{ "_id" : "Himachal Pradesh", "death_rate" : 0.013683194297026743 }
{ "_id" : "Andaman and Nicobar Islands", "death_rate" : 0.013271016059583 14 }
{ "_id" : "Goa", "death_rate" : 0.012993731554830008 }
Type "it" for more
> it
{ "_id" : "Telangana", "death_rate" : 0.012386022046413698 }
{ "_id" : "Ladakh", "death_rate" : 0.011388032133353121 }
{ "_id" : "Rajasthan", "death_rate" : 0.011016575552287607 }
{ "_id" : "Haryana", "death_rate" : 0.010791711067601302 }
{ "_id" : "Chhattisgarh", "death_rate" : 0.010596134007500433 }
{ "_id" : "Tripura", "death_rate" : 0.010457464227790893 }
{ "_id" : "Jharkhand", "death_rate" : 0.008854036712228785 }
{ "_id" : "Meghalaya", "death_rate" : 0.00871078299832681 }
{ "_id" : "Telangana***", "death_rate" : 0.0086722830023253 15 }
{ "_id" : "Andhra Pradesh", "death_rate" : 0.008436402095975534 }
{ "_id" : "Telengana***", "death_rate" : 0.008400126001890029 }
{ "_id" : "Manipur", "death_rate" : 0.008070729446337337 }
{ "_id" : "Telengana", "death_rate" : 0.006000302418070762 }
{ "_id" : "Bihar", "death_rate" : 0.005106150187609466 }
{ "_id" : "Odisha", "death_rate" : 0.004666207369521734 }
{ "_id" : "Assam", "death_rate" : 0.003946583745788 3105 }
```

Activate Windows
Go to Settings to activate Windows.

Thank You
For Your Attention