Implementation

For project implementation, I am using two MongoDB tools those are.

- MongoDB Compass
- MongoDB Shell

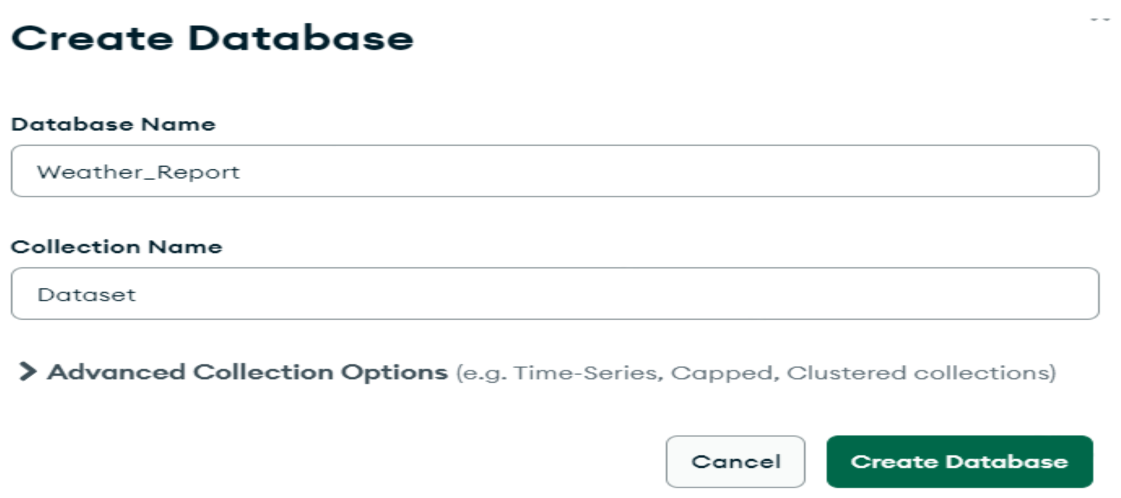
By using MongoDB Compass, I created databases and collections. I used the weather dataset from the Kaggle website for our implementation.

The dataset consists of 184 different nations' data.
The data contains 16 attributes and 17892 rows of data.

Here is the weather data:

- temp: Mean temperature for the day in degrees Fahrenheit to tenths.
- max: Maximum temperature reported during the day in Fahrenheit to tenths.
- min: Minimum temperature reported during the day in Fahrenheit to tenths.
- stp: Mean station pressure for the day in millibars to tenths.
- slp: Mean Sea level pressure for the day in millibars to tenths.
- dewp: Mean dew point for the day in degrees Fahrenheit to tenths.
- wdsp: Mean wind speed for the day in knots to tenths.
- prcp: Total precipitation (rain and/or melted snow) reported during the day in inches and hundredths.
- fog: Indicators (1 = yes, 0 = no/not reported) for the occurrence during the day.

Creating a Database and Collection in MongoDB Compass:

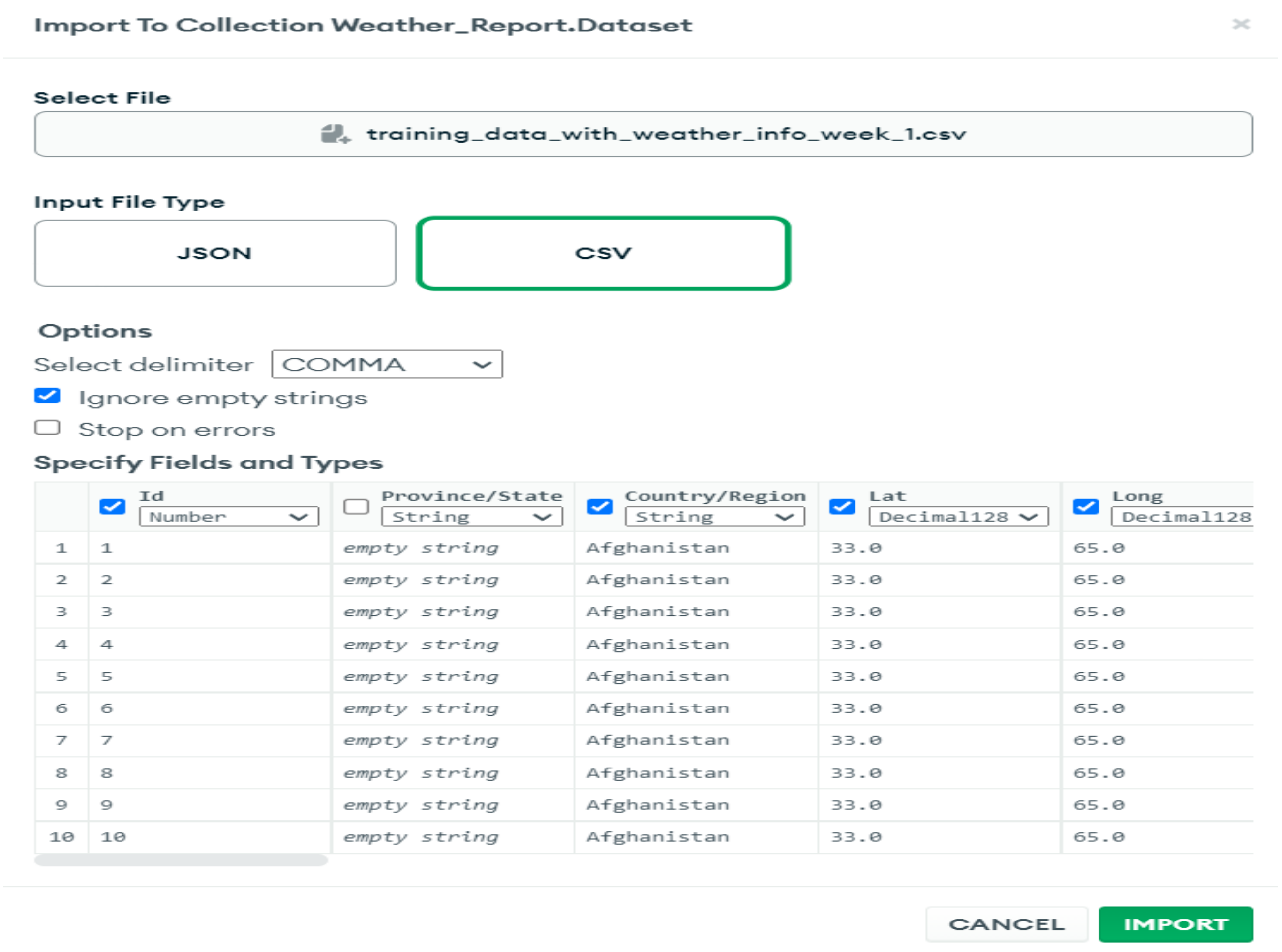

Create Database

Database Name: Weather_Report

Collection Name: Dataset

> Advanced Collection Options (e.g. Time-Series, Capped, Clustered collections)
Cancel
Create Database

Importing CSV file to the destination database:



Import To Collection Weather_Report.Dataset

Select File

training_data_with_weather_info_week_1.csv

Input File Type

JSON [✓ CSV]

Options

Select delimiter: COMMA

☑ Ignore empty strings

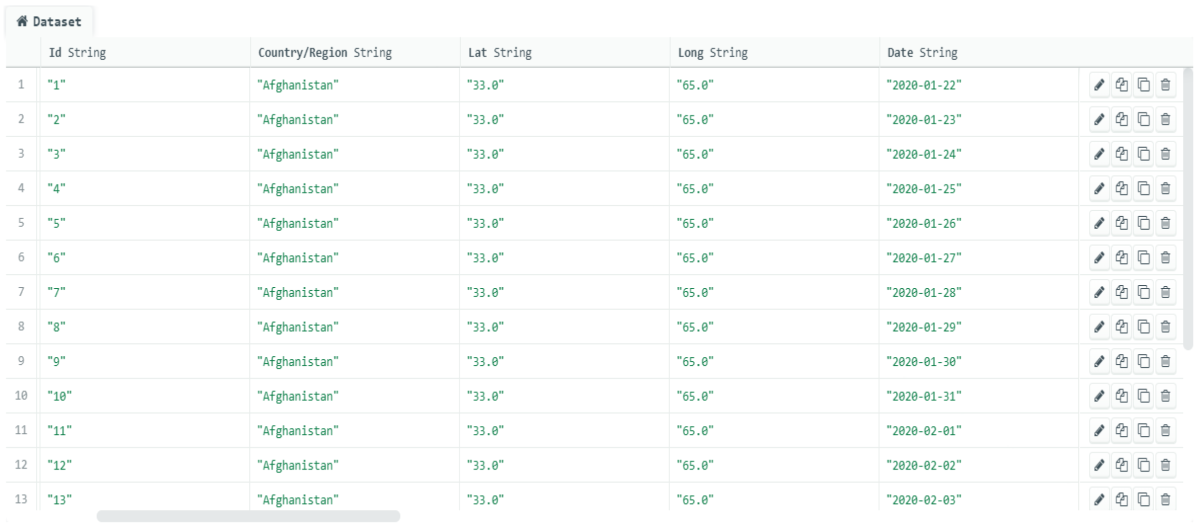
☐ Stop on errors

Specify Fields and Types

	☑ Id Number	☐ Province/State String	☑ Country/Region String	☑ Lat Decimal128	☑ Long Decimal128
1	1	*empty string*	Afghanistan	33.0	65.0
2	2	*empty string*	Afghanistan	33.0	65.0
3	3	*empty string*	Afghanistan	33.0	65.0
4	4	*empty string*	Afghanistan	33.0	65.0
5	5	*empty string*	Afghanistan	33.0	65.0
6	6	*empty string*	Afghanistan	33.0	65.0
7	7	*empty string*	Afghanistan	33.0	65.0
8	8	*empty string*	Afghanistan	33.0	65.0
9	9	*empty string*	Afghanistan	33.0	65.0
10	10	*empty string*	Afghanistan	33.0	65.0

I just removed some unwanted attributes from the dataset.

Imported Data:

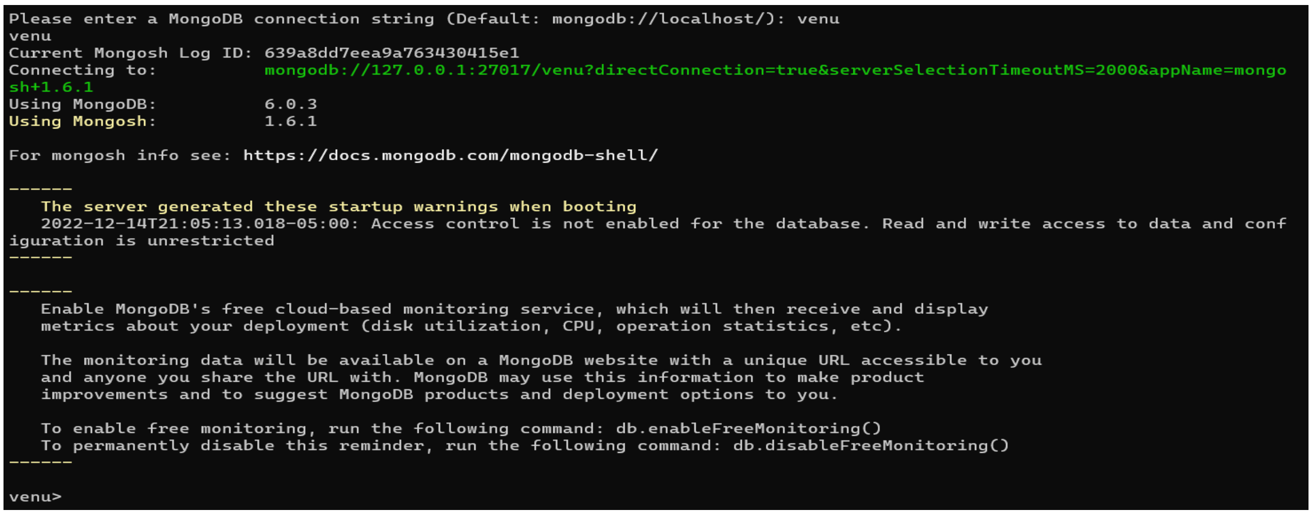


Dataset

	Id String	Country/Region String	Lat String	Long String	Date String
1	"1"	"Afghanistan"	"33.0"	"65.0"	"2020-01-22"
2	"2"	"Afghanistan"	"33.0"	"65.0"	"2020-01-23"
3	"3"	"Afghanistan"	"33.0"	"65.0"	"2020-01-24"
4	"4"	"Afghanistan"	"33.0"	"65.0"	"2020-01-25"
5	"5"	"Afghanistan"	"33.0"	"65.0"	"2020-01-26"
6	"6"	"Afghanistan"	"33.0"	"65.0"	"2020-01-27"
7	"7"	"Afghanistan"	"33.0"	"65.0"	"2020-01-28"
8	"8"	"Afghanistan"	"33.0"	"65.0"	"2020-01-29"
9	"9"	"Afghanistan"	"33.0"	"65.0"	"2020-01-30"
10	"10"	"Afghanistan"	"33.0"	"65.0"	"2020-01-31"
11	"11"	"Afghanistan"	"33.0"	"65.0"	"2020-02-01"
12	"12"	"Afghanistan"	"33.0"	"65.0"	"2020-02-02"
13	"13"	"Afghanistan"	"33.0"	"65.0"	"2020-02-03"

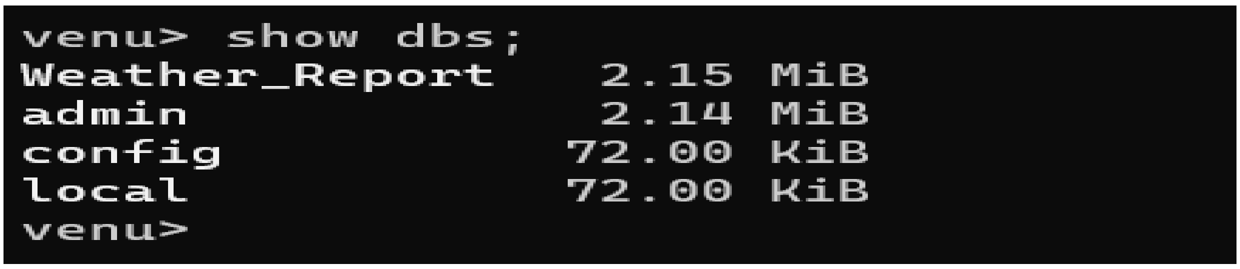
By using MongoDB Shell, we will develop a script for implementation.

Connecting to the MongoDB internal server:



```
Please enter a MongoDB connection string (Default: mongodb://localhost/): venu
venu
Current Mongosh Log ID: 639a8dd7eea9a763430415e1
Connecting to:          mongodb://127.0.0.1:27017/venu?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongo
sh+1.6.1
Using MongoDB:          6.0.3
Using Mongosh:          1.6.1

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

------
   The server generated these startup warnings when booting
   2022-12-14T21:05:13.018-05:00: Access control is not enabled for the database. Read and write access to data and conf
iguration 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()
------

venu>
```

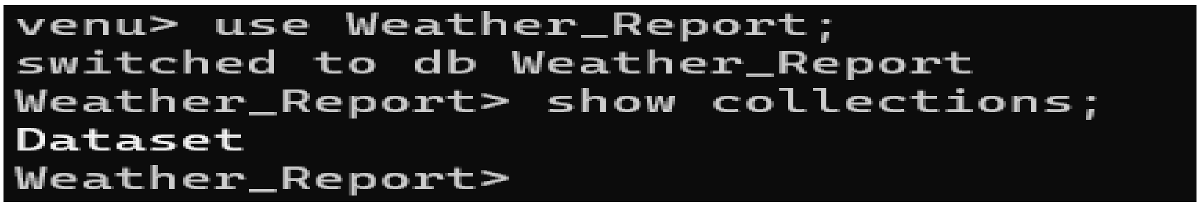
Displaying databases in the MongoDB Shell:



```
venu> show dbs;
Weather_Report   2.15 MiB
admin            2.14 MiB
config          72.00 KiB
local           72.00 KiB
venu>
```

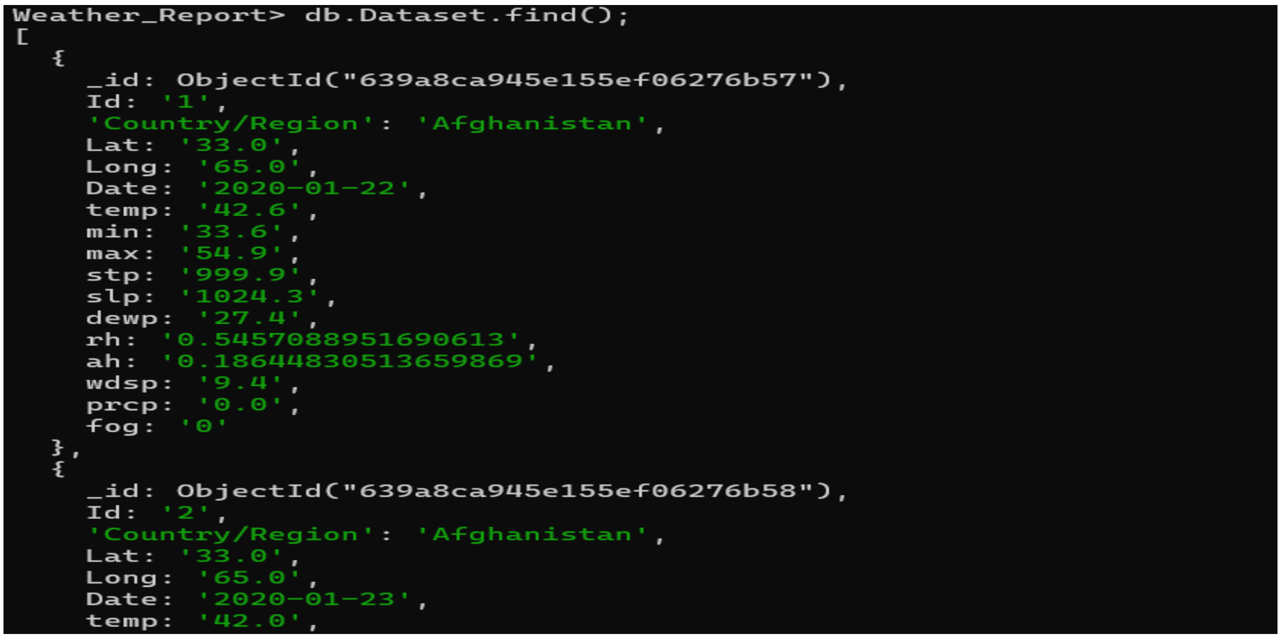
There are 4 databases in MongoDB.

Using the Weather_Report database and displaying collections:



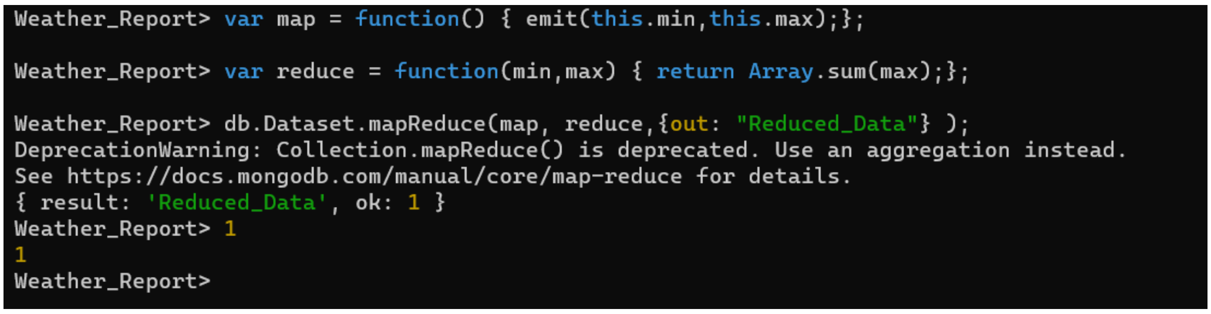
```
venu> use Weather_Report;
switched to db Weather_Report
Weather_Report> show collections;
Dataset
Weather_Report>
```

Displaying Dataset:



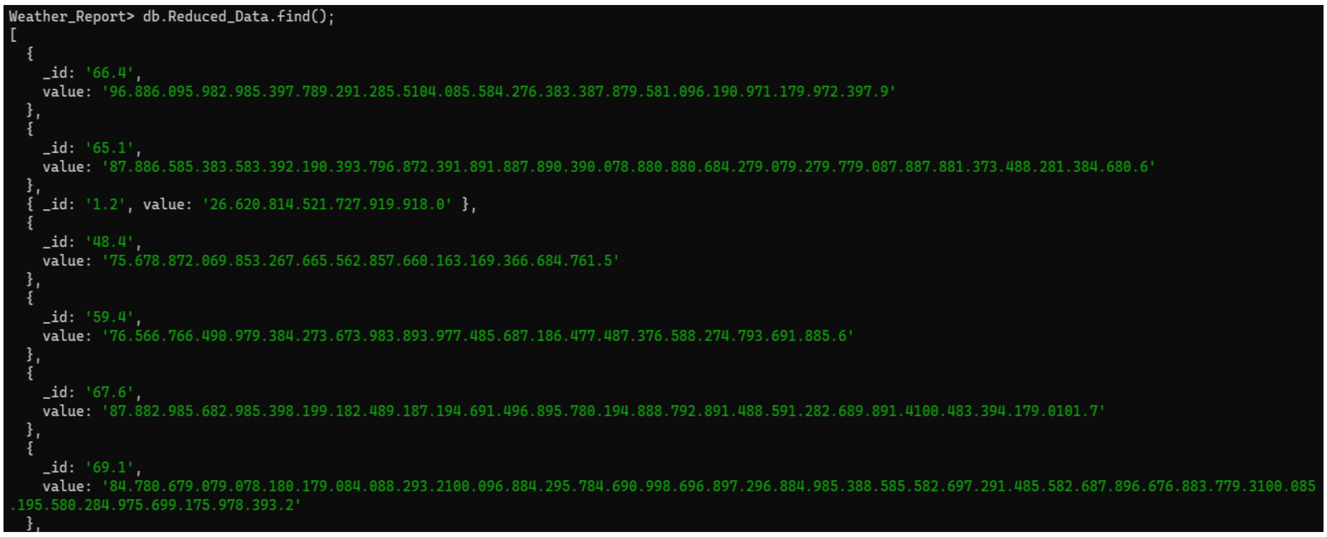
```
Weather_Report> db.Dataset.find();
[
  {
    _id: ObjectId("639a8ca945e155ef06276b57"),
    Id: '1',
    'Country/Region': 'Afghanistan',
    Lat: '33.0',
    Long: '65.0',
    Date: '2020-01-22',
    temp: '42.6',
    min: '33.6',
    max: '54.9',
    stp: '999.9',
    slp: '1024.3',
    dewp: '27.4',
    rh: '0.5457088951690613',
    ah: '0.18644830513659869',
    wdsp: '9.4',
    prcp: '0.0',
    fog: '0'
  },
  {
    _id: ObjectId("639a8ca945e155ef06276b58"),
    Id: '2',
    'Country/Region': 'Afghanistan',
    Lat: '33.0',
    Long: '65.0',
    Date: '2020-01-23',
    temp: '42.0',
```

Creating the MapReduce for mapping the attributes to access the attributes quickly. It improves performance.



```
Weather_Report> var map = function() { emit(this.min,this.max);};

Weather_Report> var reduce = function(min,max) { return Array.sum(max);};

Weather_Report> db.Dataset.mapReduce(map, reduce,{out: "Reduced_Data"} );
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.
See https://docs.mongodb.com/manual/core/map-reduce for details.
{ result: 'Reduced_Data', ok: 1 }
Weather_Report> 1
1
Weather_Report>
```

Displaying MapReduce collection:



```
Weather_Report> db.Reduced_Data.find();
[
  {
    _id: '66.4',
    value: '96.886.095.982.985.397.789.291.285.5104.085.584.276.383.387.879.581.096.190.971.179.972.397.9'
  },
  {
    _id: '65.1',
    value: '87.886.585.383.583.392.190.393.796.872.391.891.887.890.390.078.880.880.684.279.079.279.779.087.887.881.373.488.281.384.680.6'
  },
  { _id: '1.2', value: '26.620.814.521.727.919.918.0' },
  {
    _id: '48.4',
    value: '75.678.872.069.853.267.665.562.857.660.163.169.366.684.761.5'
  },
  {
    _id: '59.4',
    value: '76.566.766.490.979.384.273.673.983.893.977.485.687.186.477.487.376.588.274.793.691.885.6'
  },
  {
    _id: '67.6',
    value: '87.882.985.682.985.398.199.182.489.187.194.691.496.895.780.194.888.792.891.488.591.282.689.891.4100.483.394.179.0101.7'
  },
  {
    _id: '69.1',
    value: '84.780.679.079.078.180.179.084.088.293.2100.096.884.295.784.690.998.696.897.296.884.985.388.585.582.697.291.485.582.687.896.676.883.779.3100.085
.195.580.284.975.699.175.978.393.2'
  },
```

Summary

In this practicum, I observe the below results.

The practicum's main goal is to prove that MongoDB is the best solution for the problem that I discussed. In this implementation, I showed some MongoDB Functionalities. For that, I took a dataset from the Kaggle website which is related to weather information. The dataset was imported to the MongoDB Compass and stored in the collection object. Later I implemented the MapReduce function on the data in the MongoDB Shell. The MapReduce function maps the data into key-value pairs. So that we can access the data quickly manner. The weather company's major problem was solved by using the MapReduce function in MongoDB. Later this data might be used in mobile and web applications to cast weather updates on time and precisely. By this analysis, I conclude that MongoDB is the best solution for faster analysis.