Assignment-2
Map Reduce and Aggregate
ADBMSL

Name: Prasad Sanjay Khalkar
Roll No: 33138
TE-09 L-09
Problem Statement: Implement Map-reduce and aggregation, indexing with suitable example in MongoDB. Demonstrate the following: • Aggregation framework • Create and drop different types of indexes and explain() to show the advantage of the indexes

Theory:

MapReduce:
In MongoDB, map-reduce operations use custom JavaScript functions to map, or associate, values to a key. If a key has multiple values mapped to it, the operation reduces the values for the key to a single object.

Characteristics:
● Support non-sharded and sharded input collections.
● Can be used for incremental aggregation over large collections.
● Returns result set inline.
● Supports non-sharded and shardedinput collections.
● Uses a "pipeline" approach where objects are transformed as they pass
through a series of pipeline operators such as matching, projecting, sorting,andgrouping.

MapReduce uses the two following steps:
1) Map Function:
    ● var mapFunction1 = function()
    ● { emit(this.cust_id, this.amount);};
2) Reduce Function:
    ● var reduceFunction1 = function(key, values)
    ● {return Array.sum(values); };

Aggregation:
Aggregation operations process multiple documents and return computed results. You can use aggregation operations to:

- Group values from multiple documents together.
- Perform operations on the grouped data to return a single result.
- Analyze data changes over time.

Indexing:
Indexes are special data structures [1] that store a small portion of the collection's data set in an easy to traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field. The ordering of the index entries supports efficient equality matches and range-based query operations. In addition, MongoDB can return sorted results by using the ordering in the index.

Explain():
The explain command provides information on the execution of the following commands: aggregate, count, distinct, find, findAndModify, delete, mapReduce, and update.

Syntax:
{
explain: <command>,
verbosity: <string>,
comment: <any>
}

Implementation:
Map Reduce:

Database and the collection for the example:
The collection schema consists the ID of book, Book Name, Author Name and the status whether it is active or not.

```
> use book_database;
switched to db book_database
> db.createCollection("book_details",{ validator: { $jsonSchema:{
... bsonType:"object", required: ["bookID","bookName","authorName","status"],
... properties:{ bookID:{bsonType:"int"}, bookName:{bsonType:"string"},authorName:{bsonType:"string"},status:{bsonType:"string"} } } } } );
{ "ok" : 1 }
> db.book_details.insert({bookID:101,bookName:"Bisarjan",authorName:"R.N.Tagore",status:"Active"});
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 121,
                "errmsg" : "Document failed validation"
        }
})
> db.book_details.insert({bookID:NumberInt(101),bookName:"Bisarjan",authorName:"R.N.Tagore",status:"Active"});
WriteResult({ "nInserted" : 1 })
> db.book_details.insert({bookID:NumberInt(105),bookName:"The Merchant of Venice",authorName:"Shakespeare",status:"Active"});
WriteResult({ "nInserted" : 1 })
> db.book_details.insert({bookID:NumberInt(102),bookName:"Antony and Cleopatra",authorName:"Shakespeare",status:"Active"});
WriteResult({ "nInserted" : 1 })
> db.book_details.insert({bookID:NumberInt(111),bookName:"Geetanjali",authorName:"R.N.Tagore",status:"Active"});
WriteResult({ "nInserted" : 1 })
> db.book_details.insert({bookID:NumberInt(113),bookName:"Chitra",authorName:"R.N.Tagore",status:"Inactive"});
WriteResult({ "nInserted" : 1 })
> db.book_details.insert({bookID:NumberInt(123),bookName:"Origin of Species",authorName:"Charles Darwin",status:"Active"});
WriteResult({ "nInserted" : 1 })
```

The documents inserted in the collection:

```
> db.book_details.find();
{ "_id" : ObjectId("6163c5275daaad5e753eee55"), "bookID" : 101, "bookName" : "Bisarjan", "authorName" : "R.N.Tagore", "status" : "Active" }
{ "_id" : ObjectId("6163c5595daaad5e753eee56"), "bookID" : 105, "bookName" : "The Merchant of Venice", "authorName" : "Shakespeare", "status" : "Active" }
{ "_id" : ObjectId("6163c6005daaad5e753eee57"), "bookID" : 102, "bookName" : "Antony and Cleopatra", "authorName" : "Shakespeare", "status" : "Active" }
{ "_id" : ObjectId("6163c61a5daaad5e753eee58"), "bookID" : 111, "bookName" : "Geetanjali", "authorName" : "R.N.Tagore", "status" : "Active" }
{ "_id" : ObjectId("6163c63a5daaad5e753eee59"), "bookID" : 113, "bookName" : "Chitra", "authorName" : "R.N.Tagore", "status" : "Inactive" }
{ "_id" : ObjectId("6163c65d5daaad5e753eee5a"), "bookID" : 123, "bookName" : "Origin of Species", "authorName" : "Charles Darwin", "status" : "Active" }
{ "_id" : ObjectId("6163c6775daaad5e753eee5b"), "bookID" : 109, "bookName" : "Shakuntala", "authorName" : "Kalidas", "status" : "Active" }
{ "_id" : ObjectId("6163c6915daaad5e753eee5c"), "bookID" : 155, "bookName" : "Raghuvamsa", "authorName" : "Kalidas", "status" : "Active" }
{ "_id" : ObjectId("6163c6ae5daaad5e753eee5d"), "bookID" : 187, "bookName" : "Avigyan Sakuntalam", "authorName" : "Kalidas", "status" : "Inactive" }
{ "_id" : ObjectId("6163c6df5daaad5e753eee5e"), "bookID" : 199, "bookName" : "Time Machine", "authorName" : "H.G.Wells", "status" : "Active" }
{ "_id" : ObjectId("6163c70b5daaad5e753eee5f"), "bookID" : 148, "bookName" : "A tale of two cities", "authorName" : "Charles Dickens", "status" : "Active" }
{ "_id" : ObjectId("6163c73b5daaad5e753eee60"), "bookID" : 158, "bookName" : "David Copperfield", "authorName" : "Charles Dickens", "status" : "Inactive" }
```

The mapReduce function gives us a collection which tells us the active number of books for each author:

```
> var mapFunction1 = function(){ emit(this.authorName,1);}
> var reduceFunction1 = function(name,count){return Array.sum(count);};
> db.book_details.mapReduce(
... mapFunction1,
... reduceFunction1,
... { query:{status:"Active"}, out:"author_info" })
{
        "result" : "author_info",
        "timeMillis" : 973,
        "counts" : {
                "input" : 9,
                "emit" : 9,
                "reduce" : 3,
                "output" : 6
        },
        "ok" : 1
}
> db.author_info.find();
{ "_id" : "Charles Darwin", "value" : 1 }
{ "_id" : "Charles Dickens", "value" : 1 }
{ "_id" : "H.G.Wells", "value" : 1 }
{ "_id" : "Kalidas", "value" : 2 }
{ "_id" : "R.N.Tagore", "value" : 2 }
{ "_id" : "Shakespeare", "value" : 2 }
> show collections;
author_info
book_details
>
```

Aggregate:

The collection for this example consists of information related to students such as name, email ID, phone No, marks of 3 subject(Maths, ADBMS, ML) and div:

```
> db.students.find();
{ "_id" : ObjectId("6170e4dc6826486654664f0c"), "name" : "Prasad K", "emailID" : "prasadk@gmail.com", "phoneNo" : 1472583960, "marks" : { "maths" : 99, "ADBMS" : 95, "ML" : 97 }, "div" : 9 }
{ "_id" : ObjectId("6170e51768264866546 64f0d"), "name" : "Rohit K", "emailID" : "rohitk@gmail.com", "phoneNo" : 1478523690, "marks" : { "maths" : 98, "ADBMS" : 92, "ML" : 94 }, "div" : 10 }
{ "_id" : ObjectId("6170e53f6826486654664f0e"), "name" : "Tanmay B", "emailID" : "tanmayb@gmail.com", "phoneNo" : 1286608618, "marks" : { "maths" : 99, "ADBMS" : 99, "ML" : 99 }, "div" : 11 }
{ "_id" : ObjectId("6170e56f6826486654664f0f"), "name" : "Rahul J", "emailID" : "rahulj@gmail.com", "phoneNo" : 1478523690, "marks" : { "maths" : 94, "ADBMS" : 92, "ML" : 90 }, "div" : 9 }
{ "_id" : ObjectId("6170e59668264866546 64f10"), "name" : "Samay R", "emailID" : "samayr@gmail.com", "phoneNo" : 1048592818, "marks" : { "maths" : 97, "ADBMS" : 92, "ML" : 93 }, "div" : 10 }
{ "_id" : ObjectId("6170e5c26826486654664f11"), "name" : "Mithul N", "emailID" : "mithuln@gmail.com", "phoneNo" : 1478523690, "marks" : { "maths" : 97, "ADBMS" : 98, "ML" : 96 }, "div" : 11 }
{ "_id" : ObjectId("6170e5ec6826486654664f12"), "name" : "Lalit L", "emailID" : "lalitl@gmail.com", "phoneNo" : 1597534862, "marks" : { "maths" : 97, "ADBMS" : 91, "ML" : 90 }, "div" : 9 }
> db.students.find().pretty();
{
        "_id" : ObjectId("6170e4dc6826486654664f0c"),
        "name" : "Prasad K",
        "emailID" : "prasadk@gmail.com",
        "phoneNo" : 1472583960,
        "marks" : {
                "maths" : 99,
                "ADBMS" : 95,
                "ML" : 97
        },
        "div" : 9
}
{
        "_id" : ObjectId("6170e51768264866546 64f0d"),
        "name" : "Rohit K",
        "emailID" : "rohitk@gmail.com",
        "phoneNo" : 1478523690,
        "marks" : {
                "maths" : 98,
                "ADBMS" : 92,
                "ML" : 94
        },
        "div" : 10
}
```

Using aggregate function for finding totalMarks and averageMarks of each student:

```
> db.students.aggregate( {$addFields:{ totalMarks: {$add:["$marks.maths","$marks.ADBMS","$marks.ML"]}, avgMarks:{$avg:["$marks.maths","$marks.ADBMS","$marks.ML"]} } } ).pretty();
{
        "_id" : ObjectId("6170e4dc6826486654664f0c"),
        "name" : "Prasad K",
        "emailID" : "prasadk@gmail.com",
        "phoneNo" : 1472583960,
        "marks" : {
                "maths" : 99,
                "ADBMS" : 95,
                "ML" : 97
        },
        "div" : 9,
        "totalMarks" : 291,
        "avgMarks" : 97
}
{
        "_id" : ObjectId("6170e5176826486654664f0d"),
        "name" : "Rohit K",
        "emailID" : "rohitk@gmail.com",
        "phoneNo" : 1478523690,
        "marks" : {
                "maths" : 98,
                "ADBMS" : 92,
                "ML" : 94
        },
        "div" : 10,
        "totalMarks" : 284,
        "avgMarks" : 94.66666666666667
}
{
        "_id" : ObjectId("6170e53f6826486654664f0e"),
        "name" : "Tanmay B",
        "emailID" : "tanmayb@gmail.com",
        "phoneNo" : 1286608618,
        "marks" : {
                "maths" : 99,
                "ADBMS" : 99,
                "ML" : 99
        },
        "div" : 11,
        "totalMarks" : 297,
        "avgMarks" : 99
}
```

Using aggregate function for finding:

1) All the Students who have TotalMarks > 285

2) All the Students who have TotalMarks < 285

3) Average Marks for each division

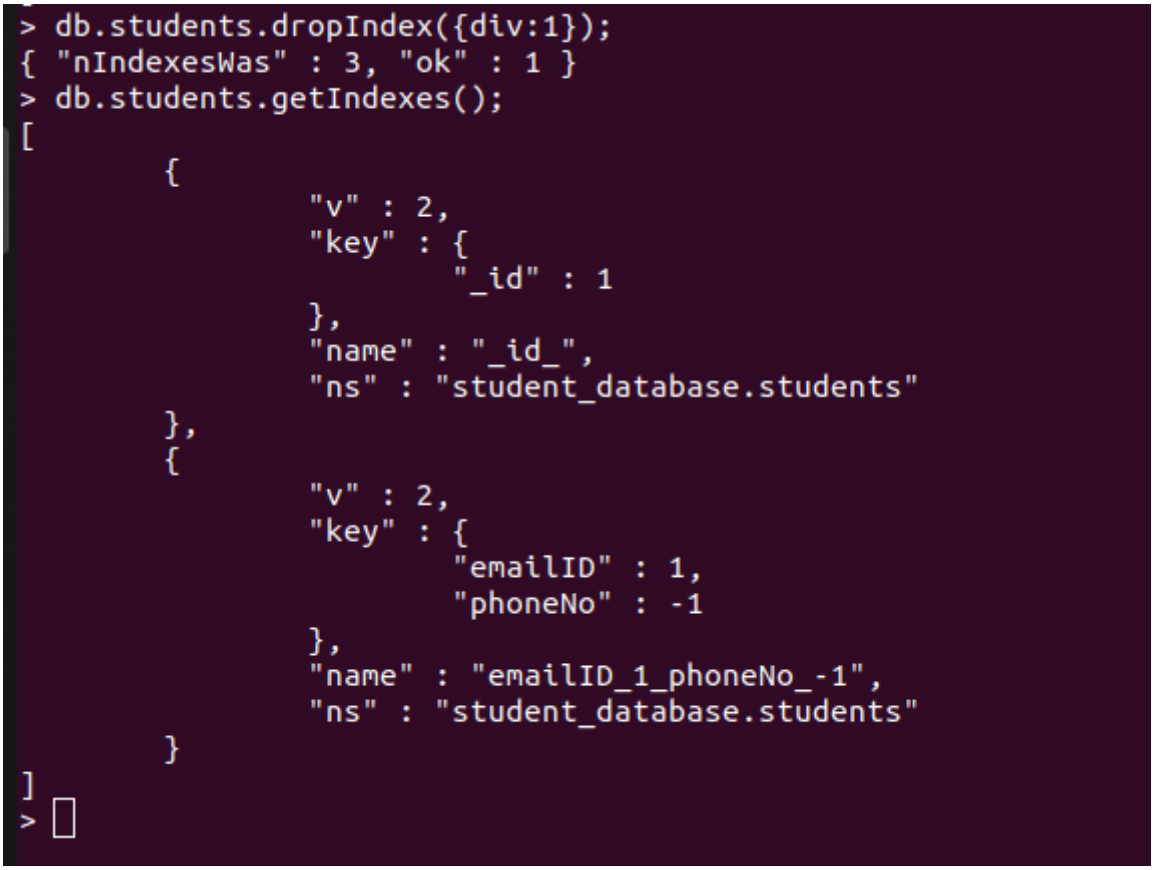
```
> db.students.aggregate( {$addFields:{ totalMarks: {$add:["$marks.maths","$marks.ADBMS","$marks.ML"]} }}, { $group: {_id:"$name", total:{$sum:"$totalMarks"} } }, {$sort:{total:1}}, {$match:{total:{$gte:2
85}}}});
{ "_id" : "Mithul N", "total" : 291 }
{ "_id" : "Prasad K", "total" : 291 }
{ "_id" : "Tanmay B", "total" : 297 }
> db.students.aggregate( {$addFields:{ totalMarks: {$add:["$marks.maths","$marks.ADBMS","$marks.ML"]} }}, { $group: {_id:"$name", total:{$sum:"$totalMarks"} } }, {$sort:{total:1}}, {$match:{total:{$lte:2
85}}}});
{ "_id" : "Rahul J", "total" : 276 }
{ "_id" : "Lalit L", "total" : 278 }
{ "_id" : "Samay R", "total" : 282 }
{ "_id" : "Rohit K", "total" : 284 }
> db.students.aggregate( {$addFields:{ avgMarks: {$avg:["$marks.maths","$marks.ADBMS","$marks.ML"]} }}, { $group: {_id:"$div", total:{$avg:"$avgMarks"} } }, {$sort:{total:1}});
{ "_id" : 9, "total" : 93.8888888888889 }
{ "_id" : 10, "total" : 94.33333333333334 }
{ "_id" : 11, "total" : 98 }
>
```

Indexing:

Creating Index:

```
> db.students.createIndex({div:1});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.students.createIndex({emailID:1, phoneNo:-1});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
> db.students.getIndexes();
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "student_database.students"
        },
        {
                "v" : 2,
                "key" : {
                        "div" : 1
                },
                "name" : "div_1",
                "ns" : "student_database.students"
        },
        {
                "v" : 2,
                "key" : {
                        "emailID" : 1,
                        "phoneNo" : -1
                },
                "name" : "emailID_1_phoneNo_-1",
                "ns" : "student_database.students"
        }
]
```

Dropping Index:



Conclusion:
Understood and implemented MapReduce, Aggregate and Indexing operations.