AIM: Indexing ,Aggregation and Map Reduce in NoSQL-DB.

PRACTICE QUESTIONS:

1.Indexing

As a first step, let’s execute the following script for creating the Assets collection.

```
for( var iCounter=1;iCounter<=  1000000;iCounter++)
{
   db.Asset.insert(
     {
     "Name":"Voting"+iCounter,
     "Desc":"Story  about a college
     student"+iCounter,
      "Rank":iCounter,
     "Language":["English","Hindi","Tamil"],
     "AssetGrp":[
                 {
                 "GrpName":"16+",  "Desc":"  Can be admitted  in  college  16+ years old"+iCounter
                 }
               ]
     }
   )
```

Default_id Index

This is a unique Index created on the ID field of the collection. Please note that we cannot drop this Index, as it’s the default.

```
db.Asset.getIndexes()
```

Single Field Index

Apart from the Default_id Index created by MongoDB, it allows us to create a user-defined Index.

Let’s create a single field Index using CLI:

```
db.Asset.createIndex( { Rank : 1 } )
```

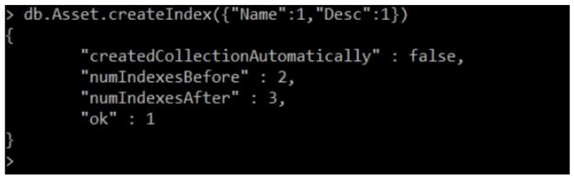
Value 1 will enable ascending sorting on grade wherein Value—1 will enable descending sorts on grade by traversing the Index in reverse order.

```
db.Asset.explain(true).find({Rank:1})
```

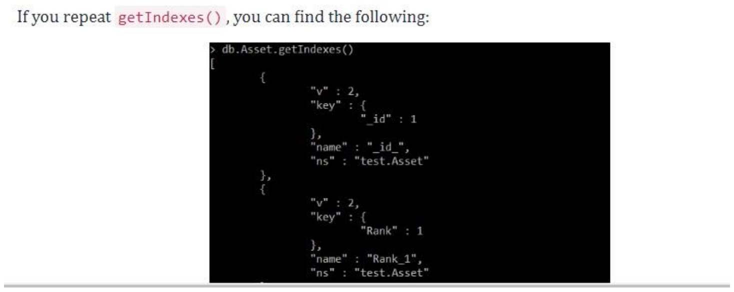
db.Asset.getIndexes()

Compound Index

MongoDB allows us to create a single structure to refer multiple fields.



```
> db.Asset.createIndex({"Name":1,"Desc":1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
>
```



If you repeat `getIndexes()`, you can find the following:

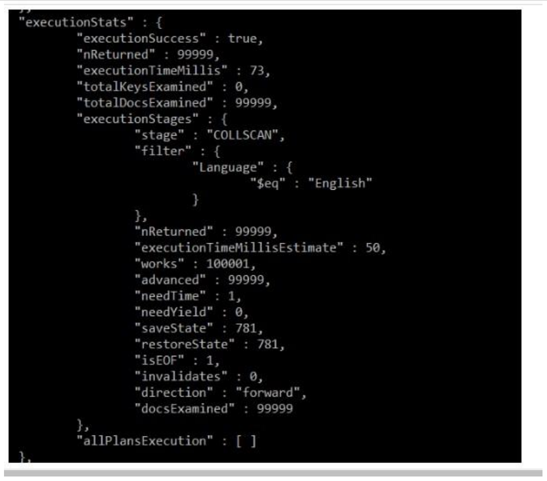
```
> db.Asset.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "test.Asset"
        },
        {
                "v" : 2,
                "key" : {
                        "Rank" : 1
                },
                "name" : "Rank_1",
                "ns" : "test.Asset"
```

Note: MongoDB imposes a limit of 31 fields in a compound Index.

Multi-Key Index

MongoDB allows multi-key Indexes to support efficient queries against arrays. It creates an Index key for each element in the array.

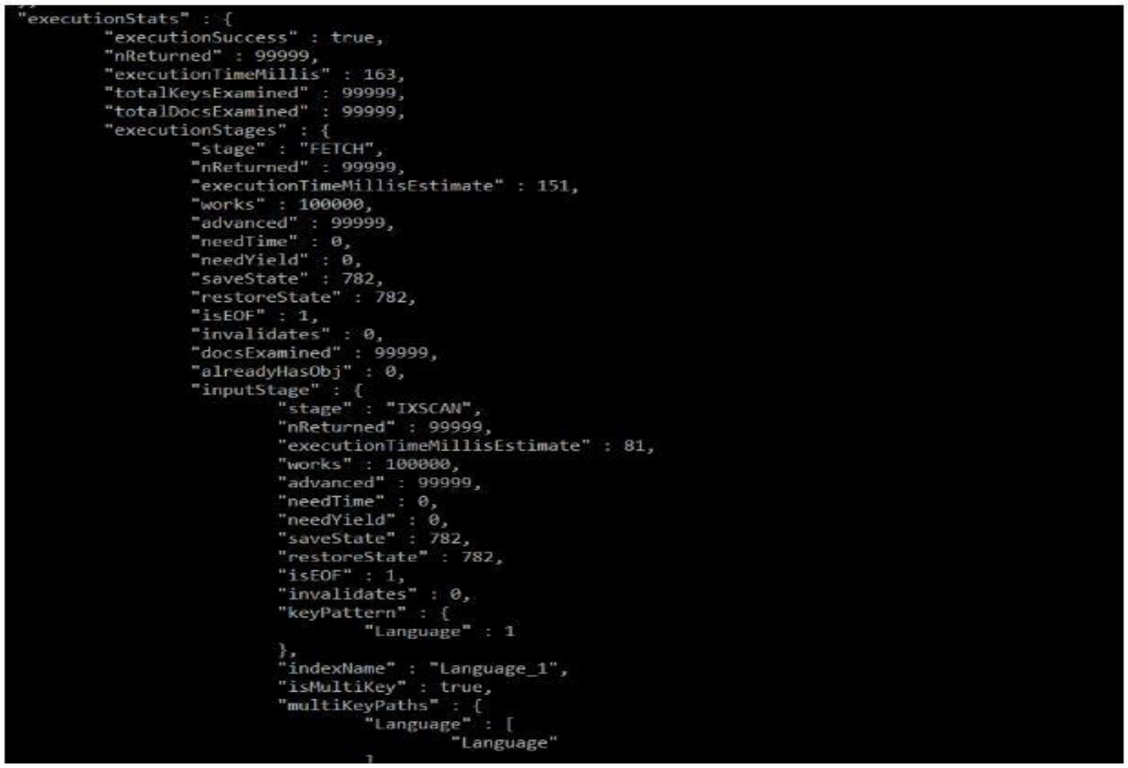
db.Asset.explain(true).find({"Language":"English"})



```
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 99999,
        "executionTimeMillis" : 73,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 99999,
        "executionStages" : {
                "stage" : "COLLSCAN",
                "filter" : {
                        "Language" : {
                                "$eq" : "English"
                        }
                },
                "nReturned" : 99999,
                "executionTimeMillisEstimate" : 50,
                "works" : 100001,
                "advanced" : 99999,
                "needTime" : 1,
                "needYield" : 0,
                "saveState" : 781,
                "restoreState" : 781,
                "isEOF" : 1,
                "invalidates" : 0,
                "direction" : "forward",
                "docsExamined" : 99999
        },
        "allPlansExecution" : [ ]
},
```

db.Asset.createIndex({"Language":1}

Result of the explain command after multi-index:



```
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 99999,
                "executionTimeMillis" : 163,
                "totalKeysExamined" : 99999,
                "totalDocsExamined" : 99999,
                "executionStages" : {
                        "stage" : "FETCH",
                        "nReturned" : 99999,
                        "executionTimeMillisEstimate" : 151,
                        "works" : 100000,
                        "advanced" : 99999,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 782,
                        "restoreState" : 782,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "docsExamined" : 99999,
                        "alreadyHasObj" : 0,
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "nReturned" : 99999,
                                "executionTimeMillisEstimate" : 81,
                                "works" : 100000,
                                "advanced" : 99999,
                                "needTime" : 0,
                                "needYield" : 0,
                                "saveState" : 782,
                                "restoreState" : 782,
                                "isEOF" : 1,
                                "invalidates" : 0,
                                "keyPattern" : {
                                        "Language" : 1
                                },
                                "indexName" : "Language_1",
                                "isMultiKey" : true,
                                "multiKeyPaths" : {
                                        "Language" : [
                                                "Language"
```

Special Index: Text Index

MongoDB allows us to create a text Index on the field or fields whose value is a string or an array of string elements.

```
db.Asset.createIndex(
      {
        Desc: "text"
      }
)
db.Asset.find( { $text: { $search: "college"  } } )
```

The above command will list all the documents that contain "Doting" in the description.

Do not have to create bellow index its just for your references.

Special Index: TTL

MongoDB supports time to live indexing, which can be used to remove certain documents after a certain time period. It is specifically useful on the log data and session information, as we don’t need this for a long time.

```
db.Events.insert(
{ "EventDesc":" user has clicked to detail page with trackcode 103",
                "EventDateTime": new Date()
                }
)
db.Events.createIndex( { "EventDateTime": 1 }, { expireAfterSeconds: 30 } )
```

You can notice the record disappearing in 30 seconds post-insertion.

Hashed Indexes

MongoDB allows us to create hashed Indexes to reduce the size of the Indexes. It is because only the hash is stored in the Index instead of the entire key.

```
db.collection.ensureIndex({attribName:'hashed'});
```

2. Aggregation

Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. In SQL count(*) and with group by is an equivalent of mongodb aggregation.

Syntax

Basic syntax of **aggregate()** method is as follows −

```
>db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

Expression	Description	Example
$sum	Sums up the defined value from all documents in the collection.	db.gnu.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
$avg	Calculates the average of all given values from all documents in the collection.	db.gnu.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
$min	Gets the minimum of the corresponding values from all documents in the collection.	db.gnu.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])

$max	Gets the maximum of the corresponding values from all documents in the collection.	db.gnu.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
$first	Gets the first document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage.	db.gnu.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
$last	Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage.	db.gnu.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])

Ex: Create collection name as “gnu”

Add 4-5 relevant documents in same collection.

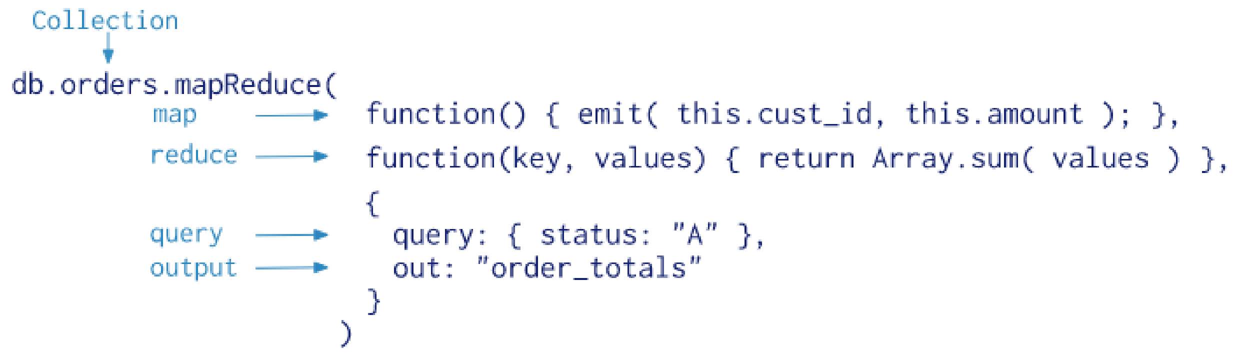
```
{
  “ title”: “MongoDB Overvie”",
  “Description” : “MongoDB is no sql database”,
  by_user: “prachi”,
  “url”: “http://www.ganpatuniversity.ac.in”,
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100
},
{
  “title”: “NoSQL Overview”,
  “Description”: “No sql database is very fast”,
  “by_user”: “pds”,
  “url”: “http://www.gnu.ac.in”,
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 10
},
{
  title: 'Neo4j Overview',
  description: 'Neo4j is no sql database',
  by_user: ”prachi shah”,
  url: 'http://www.neo4j.com',
  tags: ['neo4j', 'database', 'NoSQL'],
  likes: 750
}
```

Now from the above collection, if you want to display a list stating how many tutorials are written by each user, then you will use the following **aggregate()** method – Query to apply sum:

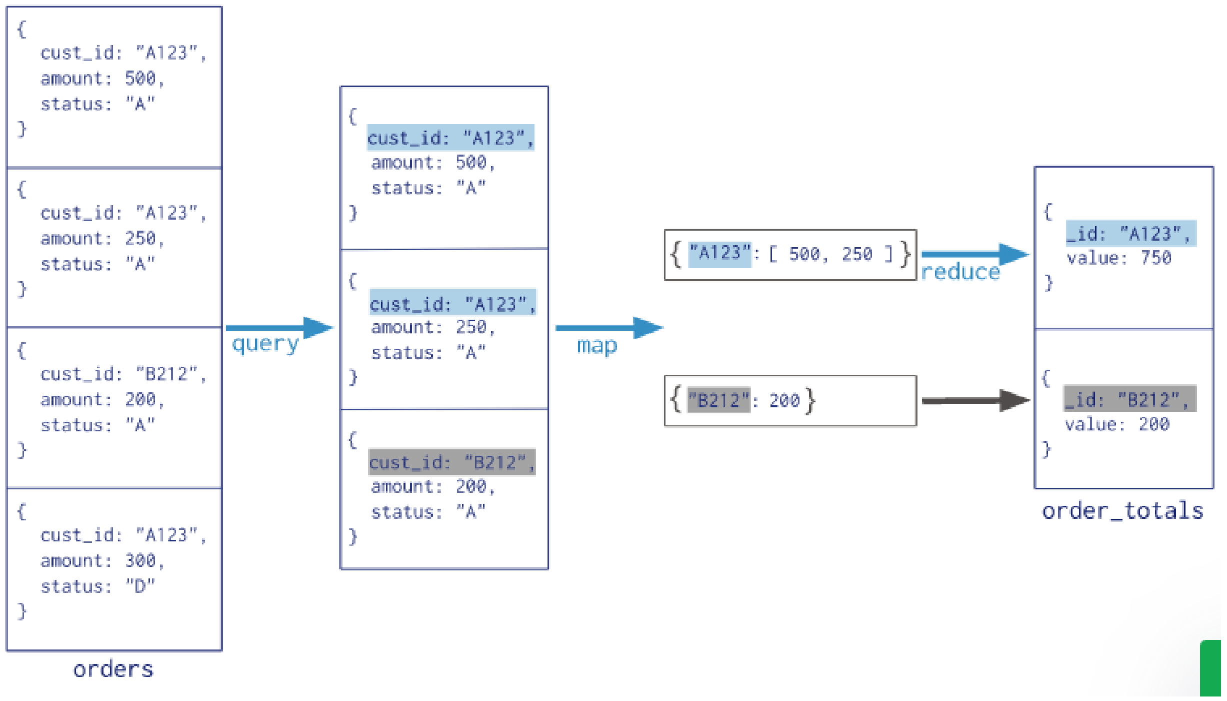
db.gnu.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : 1}}}])

3. Map-Reduce

As per the MongoDB documentation, Map-reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results. MongoDB uses mapReduce command for map-reduce operations. MapReduce is generally used for processing large data sets.



```
db.orders.mapReduce(
                     function() { emit( this.cust_id, this.amount ); },
                     function(key, values) { return Array.sum( values ) },
                     {
                       query: { status: "A" },
                       out: "order_totals"
                     }
                   )
```


{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
{ cust_id: "A123", amount: 300, status: "D" }
orders
query
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
map
{ "A123": [500, 250] }
{ "B212": 200 }
reduce
{ _id: "A123", value: 750 }
{ _id: "B212", value: 200 }
order_totals

Dataset

Map/reduce scenarios using the books collection.

```
db.books.insert( {
   "title" : "MongoDB: The Definitive Guide",
   "published" : "2013-05-23",
   "authors": [
      { "firstName" : "Kristina", "lastName" : "Chodorow" }
   ],
   "categories" : [ "Databases", "NoSQL", "Programming" ],
   "publisher" : { "name" : "O'Reilly" },
   "price" : 32.99
} )
```

```
db.books.insert( {
   "title" : "MongoDB Applied Design Patterns",
   "published" : "2013-03-19",
   "authors": [
      { "firstName" : "Rick", "lastName" : "Copeland" }
   ],
   "categories" : [ "Databases", "NoSQL", "Patterns", "Programming" ],
   "publisher" : { "name" : "O'Reilly" },
   "price" : 32.99
} )
```

```
db.books.insert( {
   "title" : "MongoDB in Action",
   "published" : "2011-12-16",
   "authors": [
      { "firstName" : "Kyle", "lastName" : "Banker" }
   ],
   "categories" : [ "Databases", "NoSQL", "Programming" ],
   "publisher" : { "name" : "Manning" },
   "price" : 30.83
} )
```

```
db.books.insert( {
   "title" : "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence",
   "published" : "2012-08-18",
   "authors": [
      { "firstName" : "Pramod J.", "lastName" : "Sadalage" },
      { "firstName" : "Martin", "lastName" : "Fowler" }
   ],
   "categories" : [ "Databases", "NoSQL" ],
   "publisher" : { "name" : "Addison Wesley" },
   "price" : 26.36
} )
```

```
db.books.insert( {
   "title" : "50 Tips and Tricks for MongoDB Developers",
   "published" : "2011-05-06",
   "authors": [
      { "firstName" : "Kristina", "lastName" : "Chodorow" }
   ],
   "categories" : [ "Databases", "NoSQL", "Programming" ],
   "publisher" : { "name" : "O'Reilly" },
   "price" : 25.08
} )
```

```
db.books.insert( {
   "title" : "MongoDB in Action, 2nd Edition",
   "published" : "2014-12-01",
   "authors": [
      { "firstName" : "Kyle", "lastName" : "Banker" },
      { "firstName" : "Peter", "lastName" : "Bakkum" },
      { "firstName" : "Tim", "lastName" : "Hawkins" }
   ],
   "categories" : [ "Databases", "NoSQL", "Programming" ],
   "publisher" : { "name" : "Manning" },
   "price" : 26.66
} )
```

```
db.books.insert( {
   "title" : "Node.js, MongoDB, and AngularJS Web Development",
   "published" : "2014-04-04",
   "authors": [
      { "firstName" : "Brad", "lastName" : "Dayley" }
   ],
   "categories" : [ "Databases", "NoSQL", "Programming", "Web" ],
   "publisher" : { "name" : "Addison Wesley" },
   "price" : 34.35
} )
```

Example: Count books by author

```
db.runCommand( {
   mapReduce: "books",
   map: function() {
      for (var index = 0; index < this.authors.length; ++index) {
         var author = this.authors[ index ];
         emit( author.firstName + " " + author.lastName, 1 );
      }
   },
   reduce: function(author, counters) {
      count = 0;

      for (var index = 0; index < counters.length; ++index) {
         count += counters[index];
      }

      return count;
   },
   out: { inline: 1 }
} )
```

O/P:

```
{
   "_id" : {
      "name" : "Addison Wesley"
   },
   "value" : {
      "count" : 2,
      "price" : 60.71,
      "average" : "US30.36"
   }
}
{
   "_id" : {
      "name" : "Manning"
   },
   "value" : {
      "count" : 2,
      "price" : 57.489999999999995,
      "average" : "US28.74"
   }
}
{
   "_id" : {
      "name" : "O'Reilly"
   },
   "value" : {
      "count" : 4,
      "price" : 116.36,
      "average" : "US29.09"
   }
}
```