**//Indexing**

**//To create index on rno in ascending order(1)-**

**//Single field Index example**

> db.stud.createIndex({rno:1})

**//To show the list of Index**

**//v is version**

**// key is on which field you created index**

**//ns-name space(database name.collection name)**

**//name- Name of index given by mongodb**

> db.stud.getIndexes()

[

{

"v" : 1,

"key" : {

"\_id" : 1

},

"ns" : "db1.stud",

"name" : "\_id\_"

},

{

"v" : 1,

"key" : {

"rno" : 1

},

"ns" : "db1.stud",

"name" : "rno\_1"

}

]

db.data.find()

{ "\_id" : ObjectId("59c89ba5bb6dda0479a91de0"), "rollno" : 1, "name" : "rachana", "marks" : 90 }

{ "\_id" : ObjectId("59c89bb8bb6dda0479a91de1"), "rollno" : 2, "name" : "nutan", "marks" : 92 }

{ "\_id" : ObjectId("59c89bc8bb6dda0479a91de2"), "rollno" : 3, "name" : "sakshi", "marks" : 88 }

{ "\_id" : ObjectId("59c89bdebb6dda0479a91de3"), "rollno" : 4, "name" : "shweta", "marks" : 75 }

{ "\_id" : ObjectId("59c89bf2bb6dda0479a91de4"), "rollno" : 5, "name" : "neha", "marks" : 62 }

{ "\_id" : ObjectId("59c89c0abb6dda0479a91de5"), "rollno" : 6, "name" : "rakhi", "marks" : 55 }

{ "\_id" : ObjectId("59c89e62bb6dda0479a91de6"), "rollno" : 5, "name" : "neha", "marks" : 62 }

{ "\_id" : ObjectId("59c89e6fbb6dda0479a91de7"), "rollno" : 5, "name" : "neha", "marks" : 62 }

{ "\_id" : ObjectId("59c8a0a9bb6dda0479a91de8"), "rollno" : 8, "name" : "NEHA", "marks" : 68 }

> db.data.find({name:'neha'}).explain()

{

"cursor" : "BasicCursor",

"isMultiKey" : false,

"n" : 3,

"nscannedObjects" : 9,

"nscanned" : 9,

"nscannedObjectsAllPlans" : 9,

"nscannedAllPlans" : 9,

"scanAndOrder" : false,

"indexOnly" : false,

"nYields" : 0,

"nChunkSkips" : 0,

"millis" : 0,

"server" : "dypcoe380-OptiPlex-380:27017",

"filterSet" : false

}

> db.data.ensureIndex({name:1})

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

> db.data.find({name:'neha'}).explain()

{

"cursor" : "BtreeCursor name\_1",

"isMultiKey" : false,

"n" : 3,

"nscannedObjects" : 3,

"nscanned" : 3,

"nscannedObjectsAllPlans" : 3,

"nscannedAllPlans" : 3,

"scanAndOrder" : false,

"indexOnly" : false,

"nYields" : 0,

"nChunkSkips" : 0,

"millis" : 0,

"indexBounds" : {

"name" : [

[

"neha",

"neha"

]

]

},

"server" : "dypcoe380-OptiPlex-380:27017",

"filterSet" : false

}

> db.data.find({name:1,marks:1}).explain()

{

"cursor" : "BtreeCursor name\_1",

"isMultiKey" : false,

"n" : 0,

"nscannedObjects" : 0,

"nscanned" : 0,

"nscannedObjectsAllPlans" : 0,

"nscannedAllPlans" : 0,

"scanAndOrder" : false,

"indexOnly" : false,

"nYields" : 0,

"nChunkSkips" : 0,

"millis" : 0,

"indexBounds" : {

"name" : [

[

1,

1

]

]

},

"server" : "dypcoe380-OptiPlex-380:27017",

"filterSet" : false

}

> db.data.find({name:'neha'}).hint({name:1}).explain()

{

"cursor" : "BtreeCursor name\_1",

"isMultiKey" : false,

"n" : 3,

"nscannedObjects" : 3,

"nscanned" : 3,

"nscannedObjectsAllPlans" : 3,

"nscannedAllPlans" : 3,

"scanAndOrder" : false,

"indexOnly" : false,

"nYields" : 0,

"nChunkSkips" : 0,

"millis" : 0,

"indexBounds" : {

"name" : [

[

"neha",

"neha"

]

]

},

"server" : "dypcoe380-OptiPlex-380:27017",

"filterSet" : false

}

>

**// To drop single index**

> db.stud.dropIndex({rno:1})

{ "nIndexesWas" : 3, "ok" : 1 }

**// To drop all indexes at a time**

> db.stud.dropIndexes()

{

"nIndexesWas" : 2,

"msg" : "non-\_id indexes dropped for collection",

"ok" : 1

}

**//Aggregation Example**

**//Create Product collection**

> db.product.insert({ctype:'p', item:'paste', amount:40, customer:'Mike'})

> db.product.insert({ctype:'o', item:'Guitar', amount:200, customer:'Tom'})

> db.product.insert({ctype:'p', item:'milk', amount:60, customer:'Mike'})

> db.product.insert({ctype:'p', item:'pizza', amount:150, customer:'Kiran'})

> db.product.insert({ctype:'p', item:'paste', amount:40, customer:'Kiran'})

> db.product.insert({ctype:'o', item:'pizza', amount:150, customer:'Devika'})

> db.product.insert({ctype:'p', item:'paste', amount:40, customer:'Mike'})

**//Syntax of aggregate**

db.product.aggregate(

[

{$match:{}},

{$group:{}},

{$sort :{}}

])

**// To find total amount of money spend by each customer**

db.product.aggregate([{$group:{\_id:"$customer",total:{$sum:"$amount"}}}])

{

"result" : [

{

"\_id" : "Devika",

"total" : 150

},

{

"\_id" : "Kiran",

"total" : 190

},

{

"\_id" : "Tom",

"total" : 200

},

{

"\_id" : "Mike",

"total" : 140

}

],

"ok" : 1

}

**//Find Total number of products Purchased by customer**

> db.product.aggregate([{$group:{\_id:"$customer",total:{$sum:1}}}])

{

"result" : [

{

"\_id" : "Devika",

"total" : 1

},

{

"\_id" : "Kiran",

"total" : 2

},

{

"\_id" : "Tom",

"total" : 1

},

{

"\_id" : "Mike",

"total" : 3

}

],

"ok" : 1

}

**//Maximum amount of product purchased by each customer**

> db.product.aggregate([{$group:{\_id:"$customer",total:{$max:"$amount"}}}])

{

"result" : [

{

"\_id" : "Devika",

"total" : 150

},

{

"\_id" : "Kiran",

"total" : 1\_id" : "pizza",

"total" : 300

},

{

"\_id" : "milk",

"total" : 60

},

{

"\_id" : "Guitar",

"total" : 200

},

{

"\_id" : "paste",

"total" : 120

},

{

"\_id" : "Tom",

"total" : 200

},

{

"\_id" : "Mike",

"total" : 60

}

],

"ok" : 1

}

**//Minimum amount of product purchased by each customer**

> db.product.aggregate([{$group:{\_id:"$customer",total:{$min:"$amount"}}}])

{

"result" : [

{

"\_id" : "Devika",

"total" : 150

},

{

"\_id" : "Kiran",

"total" : 40

},

{

"\_id" : "Tom",

"total" : 200

},

{

"\_id" : "Mike",

"total" : 40

}

],

"ok" : 1

}

**//Amount earned by each product**

> db.product.aggregate([{$group:{\_id:"$item",total:{$sum:"$amount"}}}])

{

"result" : [

{

"\_id" : "pizza",

"total" : 300

},

{

"\_id" : "milk",

"total" : 60

},

{

"\_id" : "Guitar",

"total" : 200

},

{

"\_id" : "paste",

"total" : 120

}

],

"ok" : 1

}

**//Product wise Sell**

> db.product.aggregate([{$group:{\_id:"$item",total:{$sum:1}}}])

{

"result" : [

{

"\_id" : "pizza",

"total" : 2

},

{

"\_id" : "milk",

"total" : 1

},

{

"\_id" : "Guitar",

"total" : 1

},

{

"\_id" : "paste",

"total" : 3

}

],

"ok" : 1

}

**//total amount paid by each customer in descending order**

> db.product.aggregate([{$group:{\_id:"$customer",total:{$sum:"$amount"}}},{$sort:{total:-1}}])

{

"result" : [

{

"\_id" : "Tom",

"total" : 200

},

{

"\_id" : "Kiran",

"total" : 190

},

{

"\_id" : "Devika",

"total" : 150

},

{

"\_id" : "Mike",

"total" : 140

}

],

"ok" : 1

}

**//Amount of Items purchased by customer having ctype as p**

> db.product.aggregate([{$match:{ctype:"p"}},{$group:{\_id:"$customer",total:{$sum:"$amount"}}}])

{

"result" : [

{

"\_id" : "Kiran",

"total" : 190

},

{

"\_id" : "Mike",

"total" : 140

}

],

"ok" : 1

}

Title :Implement Map reduces operation with suitable example using MongoDB

db.Journal.insert({'book\_id':1,'book\_name':'JavacdOOP','amt':500,'status':'Available'});

db.Journal.insert({'book\_id':1,'book\_name':'JavaOOP','amt':400,'status':'Not Available'});

db.Journal.insert({'book\_id':1,'book\_name':'Java','amt':300,'status':'Not Available'});

db.Journal.insert({'book\_id':2,'book\_name':'Java','amt':300,'status':'Available'});

db.Journal.insert({'book\_id':2,'book\_name':'OPP','amt':200,'status':'Available'});

db.Journal.insert({'book\_id':2,'book\_name':'C+','amt':200,'status':'Available'});

db.Journal.insert({'book\_id':3,'book\_name':'C+','amt':150,'status':'Available'});

db.Journal.insert({'book\_id':3,'book\_name':'C++','amt':200,'status':'Not Available'});

db.Journal.insert({'book\_id':4,'book\_name':'OPP C++','amt':300,'status':'Not Available'});

db.Journal.insert({'book\_id':5,'book\_name':'C++','amt':400,'status':'Available'});

db.Journal.insert({'book\_id':5,'book\_name':'C++Java','amt':400,'status':'Not Available'});

var mapfunction=function(){ emit(this.book\_id,this.amt)};

var reducefunction=function(key,value){return Array.sum(value);};

db.Journal.mapReduce(mapfunction,reducefunction,{'out':'new'});

db.Journal.mapReduce(mapfunction,reducefunction,{'out':'new'}).find().pretty();

db.new.find().pretty();