**Group:-B**

**ASSIGNMENT:-3**

**Problem Statement:**-Create the collection Books in mongoDB having the following fields :

TITAL,DESCRIPTION,BY,URL,TAGS AND LIKES.

**>db.createCollection("books")**

{ "ok" : 1 }

**>db.books.insert({title:"dbms",description:"sql",by:"korth",url:"www.dbms.com",tags:"a",likes:20})**

**>db.books.insert({title:"dbms",description:"nosql",by:"jhon",url:"www.dbms123.com",tags:"a",likes:50})**

**>db.books.insert({title:"nosql",description:"mongo\_nosql",by:"jhon",url:"**[**www.rdbms.com**](http://www.rdbms.com/)**",tags:"b",likes:150})**

**>db.books.insert({title:"mongo-nosql",description:"mongo\_nosql",by:"jhon",url:"www.nosql.com",tags:"b",likes:250})**

1] Find the numbers of books published by Jhon.

**>db.books.aggregate([{$match:{by:"jhon"}},{$group:{\_id:null,count:{$sum:1}}}])**

{ "result" : [ { "\_id" : null, "count" : 3 } ], "ok" : 1 }

**>db.books.find().pretty()**

*{*

*"\_id" : ObjectId("59dc4226c9d9469e0c9165c5"),*

*"title" : "dbms",*

*"description" : "sql",*

*"by" : "korth",*

*"url" : "*[*www.dbms.com*](http://www.dbms.com/)*",*

*"tags" : "a",*

*"likes" : 20*

*}*

*{*

*"\_id" : ObjectId("59dc424dc9d9469e0c9165c6"),*

*"title" : "dbms",*

*"description" : "nosql",*

*"by" : "jhon",*

*"url" : "*[*www.dbms123.com*](http://www.dbms123.com/)*",*

*"tags" : "a",*

*"likes" : 50*

*}*

*{*

*"\_id" : ObjectId("59dc4299c9d9469e0c9165c7"),*

*"title" : "nosql",*

*"description" : "mongo\_nosql",*

*"by" : "jhon",*

*"url" : "*[*www.rdbms.com*](http://www.rdbms.com/)*",*

*"tags" : "b",*

*"likes" : 150*

*}*

*{*

*"\_id" : ObjectId("59dc42bdc9d9469e0c9165c8"),*

*"title" : "mongo-nosql",*

*"description" : "mongo\_nosql",*

*"by" : "jhon",*

*"url" : "*[*www.nosql.com*](http://www.nosql.com/)*",*

*"tags" : "b",*

*"likes" : 250*

*}*

2]Find the books which have minimum likes and maximum likes published by Jhon.

**>db.books.aggregate([{$match:{by:"jhon"}},{$group:{\_id:"$by",min\_likes:{$min:"$likes"}}}])**

{ "\_id" : "jhon", "min\_likes" : 50 }

**>db.books.aggregate([{$match:{by:"jhon"}},{$group:{\_id:"$by",max\_likes:{$max:"$likes"}}}])**

{ "\_id" : "jhon", "max\_likes" : 250 }

3]Find the average number of likes of the book published by Jhon.

**>db.books.aggregate([{$match:{by:"jhon"}},{$group:{\_id:"$by",avg\_likes:{$avg:"$likes"}}}])**

{ "\_id" : "jhon", "avg\_likes" : 150 }

4]Find the first and last book published by Jhon.

**>db.books.aggregate([{$match:{by:"jhon"}},{$group:{\_id:"$by",first\_author:{$first:"$by"}}}])**

{ "\_id" : "jhon", "first\_author" : "jhon" }

**db.books.aggregate([{$match:{by:"jhon"}},{$group:{\_id:"$by",last\_author:{$last:"$title"}}}])**

{ "\_id" : "jhon", "last\_author" : "mongo-nosql" }

5]Create an Index on author name.

**db.books.ensureIndex({"by":1})**

*{*

*"createdCollectionAutomatically" : false,*

*"numIndexesBefore" : 1,*

*"numIndexesAfter" : 2,*

*"ok" : 1*

*}*

>**db.books.getIndexes()**

*[*

*{*

*"v" : 1,*

*"key" : {*

*"\_id" : 1*

*},*

*"name" : "\_id\_",*

*"ns" : "aggr.books"*

*},*

*{*

*"v" : 1,*

*"key" : {*

*"by" : 1*

*},*

*"name" : "by\_1",*

*"ns" : "aggr.books"*

*}*

*]*

6]Display the books published by Jhon and check if it is uses the index which we have created.

**db.books.find({by:"jhon"}).pretty()**

{

*"\_id" : ObjectId("59dc424dc9d9469e0c9165c6"),*

*"title" : "dbms",*

*"description" : "nosql",*

*"by" : "jhon",*

*"url" : "*[*www.dbms123.com*](http://www.dbms123.com/)*",*

*"tags" : "a",*

*"likes" : 50*

*}*

*{*

*"\_id" : ObjectId("59dc4299c9d9469e0c9165c7"),*

*"title" : "nosql",*

*"description" : "mongo\_nosql",*

*"by" : "jhon",*

*"url" : "*[*www.rdbms.com*](http://www.rdbms.com/)*",*

*"tags" : "b",*

*"likes" : 150*

*}*

*{*

*"\_id" : ObjectId("59dc42bdc9d9469e0c9165c8"),*

*"title" : "mongo-nosql",*

*"description" : "mongo\_nosql",*

*"by" : "jhon",*

*"url" : "*[*www.nosql.com*](http://www.nosql.com/)*",*

*"tags" : "b",*

*"likes" : 250*

*}*