book> db.book.insertMany( [ {refno:1001, name:"Python prgm", author:"shyam", coauthor:"rakesh", pages:350, price:450 },

{ refno:1002, name:"Java prgm", author:"amita", coauthor:"", pages:300, price:500 },

{ refno:1003, name:"html", author:"amita", coauthor:"shyam", pages:225, price:300 },

{ refno:1004, name:"MongoDB", author:"shyam", coauthor:"", pages:440,price:500},

{ refno:1005,name:"Mysql",author:"shrikant",coauthor:"nitin",pages:330,price:630}])

{

acknowledged: true,

insertedIds: {

'0': ObjectId('66325f0cab89fdc2182202da'),

'1': ObjectId('66325f0cab89fdc2182202db'),

'2': ObjectId('66325f0cab89fdc2182202dc'),

'3': ObjectId('66325f0cab89fdc2182202dd'),

'4': ObjectId('66325f0cab89fdc2182202de')

}

}

=====================================================================================

book> db.book.find().pretty()

[

{

\_id: ObjectId('66325f0cab89fdc2182202da'),

refno: 1001,

name: 'Python prgm',

author: 'shyam',

coauthor: 'rakesh',

pages: 350,

price: 450

},

{

\_id: ObjectId('66325f0cab89fdc2182202db'),

refno: 1002,

name: 'Java prgm',

author: 'amita',

coauthor: '',

pages: 300,

price: 500

},

{

\_id: ObjectId('66325f0cab89fdc2182202dc'),

refno: 1003,

name: 'html',

author: 'amita',

coauthor: 'shyam',

pages: 225,

price: 300

},

{

\_id: ObjectId('66325f0cab89fdc2182202dd'),

refno: 1004,

name: 'MongoDB',

author: 'shyam',

coauthor: '',

pages: 440,

price: 500

},

{

\_id: ObjectId('66325f0cab89fdc2182202de'),

refno: 1005,

name: 'Mysql',

author: 'shrikant',

coauthor: 'nitin',

pages: 330,

price: 630

}

]

========================================================================================

1. Count how many books wrote by Amita.

book> db.book.find({author:"amita"}).count()

2

======================================================================================

2. Count total books written by Shyam.

book> db.book.find({author:"shyam"},{coauthor:"shyam"}).count() // wrong way

2

========================================================================================

2. Count total books written by Shyam.

book> db.book.find({$or:[{author:"shyam"},{coauthor:"shyam"}]}).count() // SYNTAX OF using $or

3

=====================================================================================================================

3. How many author's books are available in library?

book> db.book.find({ author : { $ne : null } }).count()

5

========================================================================================================================

4. Find average cost of books in library

$avg Returns the average value of the numeric values. $avg ignores non-numeric values.

book> db.book.aggregate([{$group : {\_id : null, price\_average : {$avg : "$price"}}}]) //working agar RHS meh col\_name hai toh $ aaye ga

[ { \_id: null, price\_average: 476 } ]

===============================================================================================================================

https://www.javamadesoeasy.com/2017/03/avg-operator-average-of-salary-by-group.html#2

========================================================================================================================

5. Find total number of pages of books authored by Shyam.

book> db.book.aggregate([{$match: { author :"shyam"}},{$group : {\_id : null, price\_sum : {$sum : "$pages"}}}])

[ { \_id: null, price\_sum: 790 } ]

==========================================================================================================================

CAUTION------CAUTION

book> db.book.aggregate([ { $match: { author :"shyam" } },{$group : {pages\_sum : {$sum : "$pages"}}}])

MongoServerError[Location15955]: a group specification must include an \_id

The \_id field is mandatory, but you can set it to null if you do not wish to aggregate with respect to a key, or keys. Not utilising it would result in a single aggregate value over the fields. It is thus acting a 'reserved word' in this context, indicating what the resulting 'identifier'/key is for each group.

===============================================================================================================================

6. Sort the books by descending order of pages.

book> db.book.find().sort({pages:-1})

[

{

\_id: ObjectId('66325f0cab89fdc2182202dd'),

refno: 1004,

name: 'MongoDB',

author: 'shyam',

coauthor: '',

pages: 440,

price: 500

},

{

\_id: ObjectId('66325f0cab89fdc2182202da'),

refno: 1001,

name: 'Python prgm',

author: 'shyam',

coauthor: 'rakesh',

pages: 350,

price: 450

},

{

\_id: ObjectId('66325f0cab89fdc2182202de'),

refno: 1005,

name: 'Mysql',

author: 'shrikant',

coauthor: 'nitin',

pages: 330,

price: 630

},

{

\_id: ObjectId('66325f0cab89fdc2182202db'),

refno: 1002,

name: 'Java prgm',

author: 'amita',

coauthor: '',

pages: 300,

price: 500

},

{

\_id: ObjectId('66325f0cab89fdc2182202dc'),

refno: 1003,

name: 'html',

author: 'amita',

coauthor: 'shyam',

pages: 225,

price: 300

}

]

========RELATED TO AGGREGATE IN MONGODB============

* db.employee.aggregate([...]): This is the aggregation pipeline for the “employee” collection.
* [...]: The square brackets enclose an array of stages.
* Inside the array, we have a $group stage.
* The square brackets [...] denote an array in MongoDB queries.
* In the context of aggregation, the array contains the stages that form the aggregation pipeline.
* By using an array, you can specify multiple stages in a specific order, allowing for complex data transformations.

db.posts.aggregate([

// Stage 1: Only find documents that have more than 1 like

{

$match: { likes: { $gt: 1 } }

},

// Stage 2: Group documents by category and sum each categories likes

{

$group: { \_id: "$category", totalLikes: { $sum: "$likes" } }

}

])

==========AGGREGATE END==========================