Maniesh Pukhraj

22BAI1351

TASK-2 WEEK -2

1. Find the top 2 products with the highest CPU speed and where the price is between 600 and 1000:

db.products5.find({

price: { $gte: 600, $lte: 1000 },

"spec.cpu": { $exists: true, $gt: 2 }

}).sort({"spec.cpu": -1}).limit(2)

2. Find all products where the price field exists and is a number, the CPU speed is greater than 2 GHz, and the storage array contains the value 512:

db.products5.find({

price: { $exists: true, $type: "number" },

"spec.cpu": { $gt: 2 },

storage: 512

})

3. Find the top 3 products with the largest screen size and where the storage array contains more than 2 elements:

db.products5.find({

"storage.2": { $exists: true }

}).sort({"spec.screen": -1}).limit(3)

4. Find products where the storage includes both 128 and 256:

db.products5.find({

storage: { $all: [128, 256] }

})

5. Update the price of xReader to 199 and sort all products by price in ascending order:

db.products5.updateOne(

{ name: "xReader" },

{ $set: { price: 199 } }

);

db.products5.find().sort({ price: 1 })

6. Find products that have the exact color combination of white, black, and purple:

db.products5.find({

color: { $all: ["white", "black", "purple"] }

})

7. Find the average screen size of products released after 2015.

db.products5.aggregate([

{ $match: { releaseDate: { $gt: ISODate("2015-01-01") } } },

{ $group: { \_id: null, avgScreenSize: { $avg: "$spec.screen" } } }

])

8. Calculate the total number of products with 'white' or 'black' color.  
db.products5.aggregate([

{ $match: { color: { $in: ["white", "black"] } } },

{ $count: "totalProducts" }

])

9. Find the product with the highest price.

db.products5.aggregate([

{ $sort: { price: -1 } },

{ $limit: 1 }

])

10. Calculate the total storage capacity (sum of all storage options across all products).

db.products5.aggregate([

{ $unwind: "$storage" },

{ $group: { \_id: null, totalStorage: { $sum: "$storage" } } }

])

11. Find the average quantity sold for each item where the price is greater than $10, then sort these averages in descending order, skip the first result, and limit the output to 2 items.

db.sales.aggregate([

{ $match: { price: { $gt: 10 } } },

{ $group: { \_id: "$item", avgQuantity: { $avg: "$quantity" } } },

{ $sort: { avgQuantity: -1 } },

{ $skip: 1 },

{ $limit: 2 }

])

12.Identify the total quantity sold for each item, but only include those items where the maximum price recorded is at least $25. Then, sort the results by the total quantity in ascending order and limit the result to 1 item.

db.sales.aggregate([

{ $group: { \_id: "$item", totalQuantity: { $sum: "$quantity" }, maxPrice: { $max: "$price" } } },

{ $match: { maxPrice: { $gte: 25 } } },

{ $sort: { totalQuantity: 1 } },

{ $limit: 1 }

])

13.Calculate the total and average quantity sold for each item where the size is "Grande", then sort the results first by the total quantity in descending order and then by the average quantity in descending order. Skip the first result and limit the output to 2 items.

db.sales.aggregate([

{ $match: { size: "Grande" } },

{ $group: { \_id: "$item", totalQuantity: { $sum: "$quantity" }, avgQuantity: { $avg: "$quantity" } } },

{ $sort: { totalQuantity: -1, avgQuantity: -1 } },

{ $skip: 1 },

{ $limit: 2 }

])

14.The total revenue generated from all sales

db.sales.aggregate([

{ $group: { \_id: null, totalRevenue: { $sum: { $multiply: ["$price", "$quantity"] } } } }

])

15.How many “Americanos” were sold in total?

db.sales.aggregate([

{ $match: { item: "Americanos" } },

{ $group: { \_id: null, totalAmericanos: { $sum: "$quantity" } } }

])

16.Increase the price of “Lattes” by 5 units

db.sales.updateMany(

{ item: "Lattes" },

{ $inc: { price: 5 } }

)

17.Unset the “size” field for all records where the price is less than 10

db.sales.updateMany(

{ price: { $lt: 10 } },

{ $unset: { size: "" } }

)

Indexing:

1.compound Index:

db.products2.createIndex({name:2,"spec.ram":1})

2.finding compound:

db.products2.find({name:"xPhone","spec.ram":4})

3.get all indexed

db.products2.getIndexes()

4.droping index:

db.products2.dropIndex({price:1})

5.getting info about documents having index

db.products2.find({price:899}).explain()