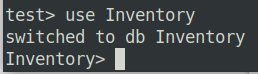
**1. Create a Mongodb Database named “Inventory”.**

use Inventory



2. Create a collection named ‘Products’ and Insert the following documents.

db.products.insertMany([

{ "\_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate": ISODate("2011-05-

14"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66

},"color":["white","black"],"storage":[64,128,256]},

{ "\_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate": ISODate("2011-09-

01") , "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66

},"color":["white","black","purple"],"storage":[128,256,512]},

{ "\_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate": ISODate("2015-

01-14"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66

},"color":["blue"],"storage":[16,64,128]},

{ "\_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate": ISODate("2020-05-

14"),"spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66

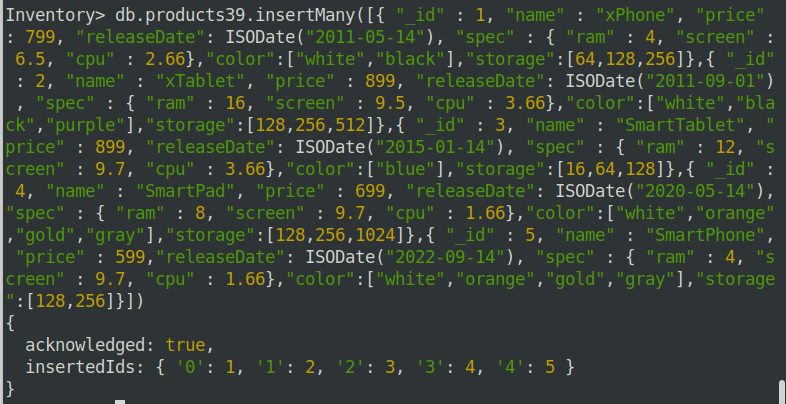
},"color":["white","orange","gold","gray"],"storage":[128,256,1024]},

{ "\_id" : 5, "name" : "SmartPhone", "price" : 599,"releaseDate": ISODate("2022-

09-14"), "spec" : { "ram" : 4, "screen" : 9.7, "cpu" : 1.66

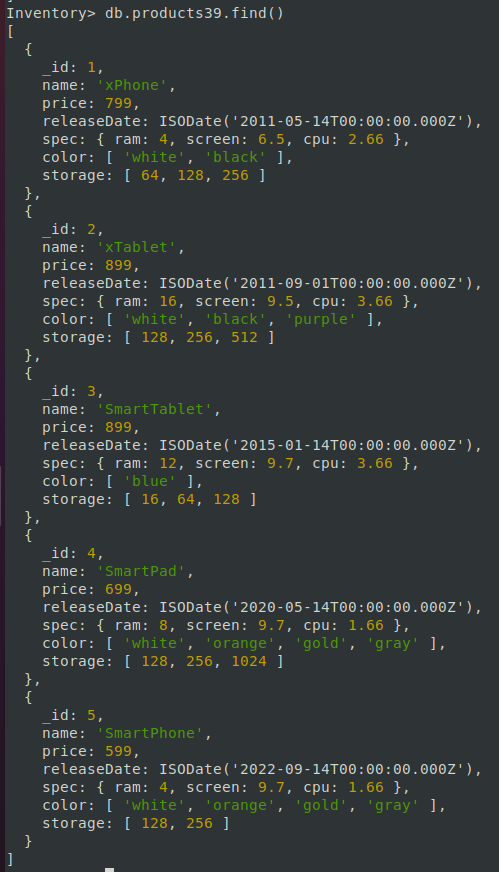
},"color":["white","orange","gold","gray"],"storage":[128,256]}

])



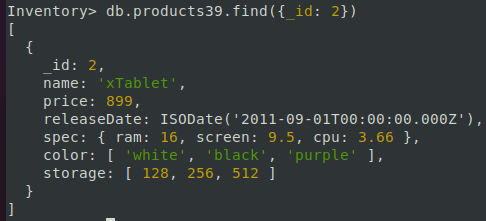
**3. Display all documents in the collection product.**

db.products.find()



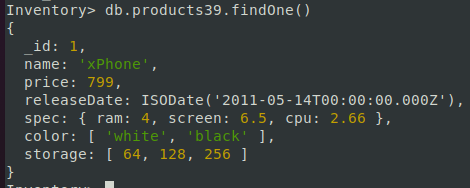
**4. Display all the details of product with \_ id is 2.**

db.products.find({\_id: 2})



**5. Display the first document in the collection product.**

db.products.findOne()



**6. Display name and price of product with \_ id is 5.**

db.products.find({ \_id: 5}, { name: 1,price: 1})



**7. Query the products collection to select all documents where the value of the price field**

**equals 899.**

db.products.find({

price: {

$eq: 899

}

}, {

name: 1,

price: 1

})

Or

db.products.find({

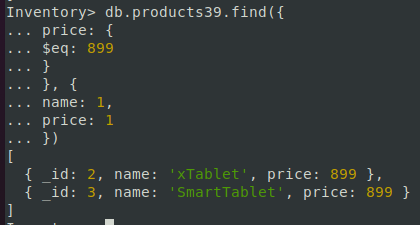
price: 899

}, {

name: 1,

price: 1

})



**8. Search for documents where the value of the ram field in the spec document equals 4.**

db.products.find({

"spec.ram": {

$eq: 4

}

}, {

name: 1,

"spec.ram": 1

})

Or

db.products.find({

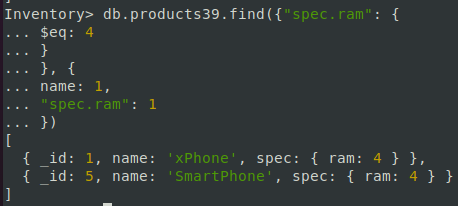
"spec.ram": 4

}, {

name: 1,

"spec.ram": 1

})



**9. Query the products collection to find all documents where the array color contains an**

**element with the value "black".**

db.products.find({

color: {

$eq: "black"

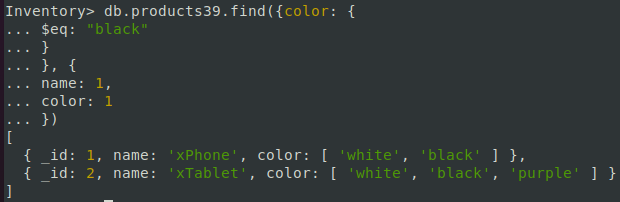
}

}, {

name: 1,

color: 1

})



**10. Select documents in the products collection with the published date is 2020-05-14.**

db.products.find({

releaseDate: {

$eq: new ISODate("2020-05-14")

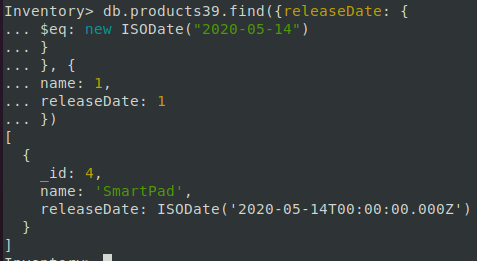
}

}, {

name: 1,

releaseDate: 1

})



**11. select documents from the products collection where price is less than 799.**

db.products.find({

price: {

$lt: 799

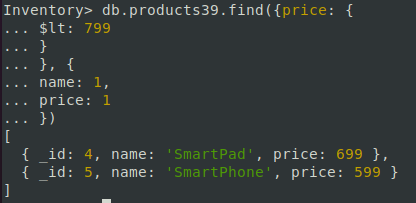
}

}, {

name: 1,

price: 1

})



**12. select documents where the value of the screen field in the spec document is less than 7.**

db.products.find({

"spec.screen": {

$lt: 7

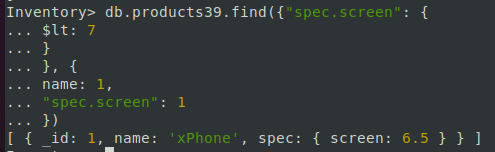
}

}, {

name: 1,

"spec.screen": 1

})



**13. query the products collection to find all documents where the array storage has at least**

**one element less than 128.**

db.products.find({

storage: {

$lt: 128

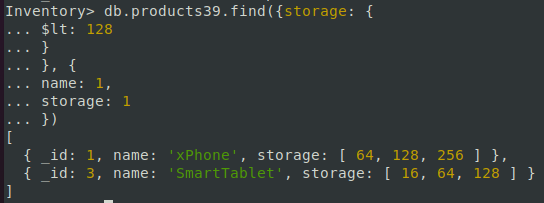
}

}, {

name: 1,

storage: 1

})



**14. Display documents from the products collection whose the price is either 599 or 799.**

db.products.find({

price: {

$in: [699, 799]

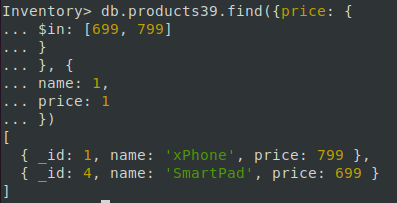
}

}, {

name: 1,

price: 1

})



**15. Display documents where the color array has at least one element either "black" or "white".**

db.products.find({

color: {

$in: ["black", "white"]

}

}, { name: 1,

color: 1

})



**16. Display documents from the products collection whose price is neither 599 or 799.**

db.products.find({

price: {

$nin: [699, 799]

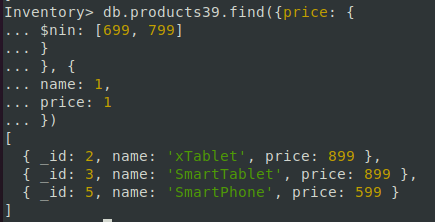
}

}, {

name: 1,

price: 1

})



**17. Display documents where the color array doesn’t have an element that is either "black" or "white".**

db.products.find({

color: {

$nin: ["black", "white"]

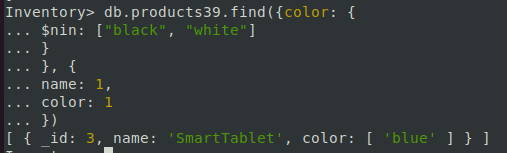
}

}, {

name: 1,

color: 1

})



**18. Display all documents in the products collection where the value in the price field is equal**

**to 899 and the value in the color field is either "white" or "black".**

db.products.find({

$and: [{

price: 899

}, {

color: {

$in: ["white", "black"]

}

}]

}, {

name: 1,

price: 1,

color: 1

})



**19. Select all documents where the price is less than 699 or greater than 799.**

db.products.find({

$or: [

{ price: {$lt: 699} },

{ price: {$gt: 799} }

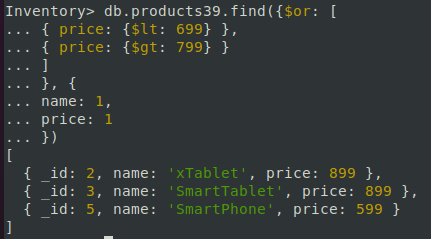
]

}, {

name: 1,

price: 1

})



Sorting Documents

Use the sort() method to sort the documents by one or more fields. Specify { field: 1 }

to sort documents by the field in ascending order and { field: -1 } to sort documents by

the field in descending order. Use the dot notation { "embeddedDoc.field" : 1 } to sort

the documents by the field in the embedded documents (embeddedDoc).

**20. Sorts the products by the values in the ram field in the spec embedded documents. It**

**includes the \_id, name, and spec fields in the matching documents.**

db.products.find({}, {

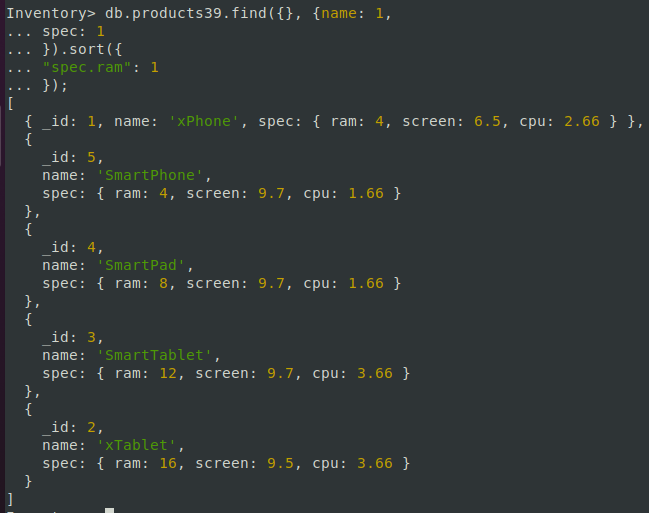
name: 1,

spec: 1

}).sort({

"spec.ram": 1

});



**21. Sorts the products by the values in the releaseDate field in descending order.**

db.products.find({

releaseDate: {

$exists: 1

}

}, {

name: 1,

releaseDate: 1

}).sort({

releaseDate: -1

});



**22. Sort the products by name and price in ascending order. It selects only documents where**

**the price field exists and includes the \_id, name, and price fields in the matching documents.**

db.products.find({

'price': {

$exists: 1

}

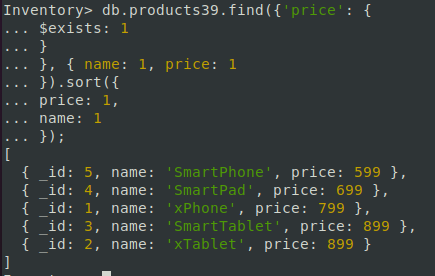
}, { name: 1, price: 1

}).sort({

price: 1,

name: 1

});



**23. Get the most expensive product in the products collection. It includes the \_id, name,**

**and price fields in the returned documents.**

db.products.find({}, {

name: 1,

price: 1

}).sort({

price: -1,

name: 1

}).limit(1);

