MongoDB

Day2

Nada Mohamed Ahmed Hassan Eleshmawy

Mansoura Open Source

Date : 4/4/2025

MongoDB Inventory Lab 2

Part 1: Basic Aggregation

1. Calculate the number of products per category and sort by highest count first.

```
db.products.aggregate([{
        $group : {
        _id :"$category",
        count : {  $count :{} } }
}]) .sort({count:-1})
```

```
Inventory> db.products.aggregate([{$group : {_id :"$category", count : {$count :{} } }}]).sort({count:-1})
[
  { _id: 'Phone', count: 4 },
  { _id: 'Laptop', count: 3 },
  { _id: null, count: 2 },
  { _id: 'TV', count: 2 }
]
Inventory>
```

Or

```
db.products.aggregate([{
        $group : {
         _id :"$category" ,
         count : {$sum :1 }
         }
}]) .sort({count:-1})
```

```
Inventory> db.products.aggregate([{$group : {_id :"$category", count : {$sum :1 } }}]).sort({count:-1})
[
  { _id: 'Phone', count: 4 },
  { _id: 'Laptop', count: 3 },
  { _id: null, count: 2 },
  { _id: 'TV', count: 2 }
]
Inventory>
```

2. Find the maximum price for each product category and include a list of products in that category.

```
db.products.aggregate([{
        $group : {
```

```
        _id :"$category" ,
        maxPrice : {$max : “$price” } ,
        productsList : { $push : “$name”}
        }
}])
```

```
Inventory> db.products.aggregate([{ $group : { _id :"$category" , maxPrice : {$max : "$price" } ,productsList : {$push : "$name"} }}])
[
  { _id: null, maxPrice: 744.2, productsList: [ 'Catel' ] },
  {
    _id: 'Laptop',
    maxPrice: 44622.476457950004,
    productsList: [ 'Toshiba Laptop', 'Laptop Apple', 'HP Laptop' ]
  },
  {
    _id: 'Phone',
    maxPrice: 16572.47645795,
    productsList: [ 'Iphone7', 'Sony Phone', 'Iphone6', 'Samaung Phone' ]
  },
  {
    _id: 'TV',
    maxPrice: 11622.1,
    productsList: [ 'Samaung TV', 'LG TV' ]
  }
]
Inventory>
```

3. Retrieve all orders made by user "ahmed" with full product details populated.

```
db.orders.aggregate([
        { $match: { userId: ObjectId("6040adb69ad8e02d58cc3e0d") } },
        { $lookup: {
                        from: "products",
                        localField: "productsIds",
                        foreignField: "_id",
                        as: "productDetails"}
         },
        { $unwind: "$productDetails" }
])
```

```
Inventory> db.orders.aggregate([ { $match: { userId: ObjectId("6040adb69ad8e02d58cc3e0d") } }, { $lookup: { from: "products", localFi
eld: "productsIds", foreignField: "_id", as: "productDetails" } } , {$unwind :"$productDetails" }] )
[
  {
    _id: ObjectId('6040ae169ad8e02d58cc3e0e'),
    userId: ObjectId('6040adb69ad8e02d58cc3e0d'),
    productsIds: [
      ObjectId('589ba2fb2742a35b47dad21c'),
      ObjectId('589ba2fb2742a35b47dad21e')
    ],
    productDetails: {
      _id: ObjectId('589ba2fb2742a35b47dad21c'),
      name: 'Iphone7',
      price: 16572.47645795,
      category: 'Phone',
      vendor: 'Apple',
      stock: [ 20 ],
      quantity: 10
    }
  },
  {
    _id: ObjectId('6040ae169ad8e02d58cc3e0e'),
    userId: ObjectId('6040adb69ad8e02d58cc3e0d'),
    productsIds: [
      ObjectId('589ba2fb2742a35b47dad21c'),
      ObjectId('589ba2fb2742a35b47dad21e')
    ],
    productDetails: {
```

—————————————— or ————————————

db.orders.aggregate([

{ $lookup: {

from: "users",

localField: "userId",

foreignField: "_id",

as: "user"} },

{ $unwind: "$user" },

{ $match: { "user.name": "ahmed" } },

{ $lookup: {

from: "products",

localField: "productsIds",

foreignField: "_id",

as: "productDetails”} },

{ $unwind: "$productDetails" }

])

4. Calculate the highest total order value for user "ahmed".

```
db.orders.aggregate([
        { $lookup: {
                from: "users",
                localField: "userId",
                foreignField: "_id",
                as: "user"}
        },
        { $unwind: "$user" },
        { $match: { "user.name": "ahmed" } },
        {$lookup: {
                from: "products",
                localField: "productsIds",
                foreignField: "_id",
                as: "productDetails"}
      },
```

```
{$project: { total: { $sum: "$productDetails.price" } } },
 { $sort: { total: -1 } },
 { $limit: 1 }
])
```

```
Inventory> db.orders.aggregate([ { $lookup: { from: "users", localField: "userId", foreignField: "_id", as: "user" } }, { $match: { "
user.name": "ahmed" } }, { $lookup: { from: "products", localField: "productsIds", foreignField: "_id", as: "productDetails" } }, { $
project: { total: { $sum: "$productDetails.price" } } }, { $sort: { total: -1 } }, { $limit: 1 }] )
[
  {
    _id: ObjectId('6040ae169ad8e02d58cc3e0e'),
    total: 28194.576457950003
  }
]
Inventory>
```

5. Calculate the average price of products for each vendor and display vendors sorted by their average price.

```
db.products.aggregate([{
        $group : {
         _id :"$vendor" ,
        avgPrice: { $avg: "$price" }
         }
} , {
   $sort: { avgPrice: -1 }
 }
])
```

```
Inventory> db.products.aggregate([{ $group: { _id: "$vendor", avgPrice: { $avg: "$price" } } }, { $sort: { avgPrice: -1 } }] )
[
  { _id: 'Apple', avgPrice: 24272.350971966665 },
  { _id: { name: 'Toshiba', phone: '011111321' }, avgPrice: 11622.1 },
  { _id: { name: 'Samaung', phone: '123' }, avgPrice: 11622.1 },
  { _id: 'Samsumg', avgPrice: 11622.1 },
  { _id: 'Sony', avgPrice: 11622.1 },
  { _id: 'HP', avgPrice: 11622.1 },
  { _id: 'LG', avgPrice: 11622.1 },
  { _id: null, avgPrice: 744.2 },
  { _id: 'OSVendor', avgPrice: 500 }
]
Inventory>
```

Part 2: Advanced Queries and Projection

6. Find all Apple products and only return the first stock location using the $ positional operator.

db.products.find({ vendor: "Apple", stock: { $gt: 0 } }, { name: 1, "stock.$": 1})

```
Inventory> db.products.find( { vendor: "Apple", stock: { $gt: 0 } }, { name: 1, "stock.$": 1} )
[
  {
    _id: ObjectId('589ba2fb2742a35b47dad21c'),
    name: 'Iphone7',
    stock: [ 20 ]
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad220'),
    name: 'Laptop Apple',
    stock: [ 300 ]
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad222'),
    name: 'Iphone6',
    stock: [ 100 ]
  }
]
```

Or

db.products.find({ vendor: "Apple" }, { name: 1, stock: { $slice: 1 } })

```
Inventory> db.products.find( { vendor: "Apple" }, { name: 1, stock: { $slice: 1 } } )
[
  {
    _id: ObjectId('589ba2fb2742a35b47dad21c'),
    name: 'Iphone7',
    stock: [ 20 ]
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad220'),
    name: 'Laptop Apple',
    stock: [ 300 ]
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad222'),
    name: 'Iphone6',
    stock: [ 100 ]
  }
]
Inventory>
```

7. Find all products that have at least one stock location with more than 100 units using $elemMatch.

db.products.find(

{ stock: { $elemMatch: { $gt: 100 } } },

{ name: 1, stock: 1 }

)

```
Inventory> db.products.find( { stock: { $elemMatch: { $gt: 100 } } }, { name: 1, stock: 1 } )
[
  {
    _id: ObjectId('589ba2fb2742a35b47dad220'),
    name: 'Laptop Apple',
    stock: [ 300, 350, 600 ]
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad222'),
    name: 'Iphone6',
    stock: [ 100, 400 ]
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad224'),
    name: 'Samaung Phone',
    stock: [ 99, 999 ]
  }
]
Inventory>
```

8. Find all Laptop products and return only their name and price, excluding the _id field.

   db.products.find({ category: "Laptop" }, { _id: 0, name: 1, price: 1 })

```
Inventory> db.products.find( { category: "Laptop" }, { _id: 0, name: 1, price: 1 } )
[
  { name: 'Toshiba Laptop', price: 11622.1 },
  { name: 'Laptop Apple', price: 44622.476457950004 },
  { price: 11622.1, name: 'HP Laptop' }
]
Inventory>
```

9. Find all products with a price greater than 10000 and return their names in uppercase and price with a 10% discount.

   db.products.find({ price : {$gt : 1000} },

   { _id: 0,

   name: {$toUpper : "$name"},

   finalPriceWithDiscount : {$multiply : ["$price" ,0.9] }

   })

```
Inventory> db.products.find( { price : {$gt : 1000} }, { _id: 0, name: {$toUpper : "$name"},finalPriceWithDiscount : {$multiply : ["$
price" ,0.9] } } )
[
  { name: 'IPHONE7', finalPriceWithDiscount: 14915.228812155001 },
  { name: 'SAMAUNG TV', finalPriceWithDiscount: 10459.890000000001 },
  {
    name: 'TOSHIBA LAPTOP',
    finalPriceWithDiscount: 10459.890000000001
  },
  { name: 'SONY PHONE', finalPriceWithDiscount: 10459.890000000001 },
  { name: 'LAPTOP APPLE', finalPriceWithDiscount: 40160.22881215501 },
  { name: 'LG TV', finalPriceWithDiscount: 10459.890000000001 },
  { name: 'IPHONE6', finalPriceWithDiscount: 10459.890000000001 },
  { name: 'HP LAPTOP', finalPriceWithDiscount: 10459.890000000001 },
  { name: 'SAMAUNG PHONE', finalPriceWithDiscount: 10459.890000000001 }
]
Inventory>
```

10. Use projection to return only the second stock value for all products with at least 3 stock locations.

db.products.find({ "stock.2": { $exists: true } },{ name: 1, stock: { $slice: [1, 1] },_id: 0 })

```
Inventory> db.products.find( { "stock.2": { $exists: true } },{ name: 1, stock: { $slice: [1, 1] },_id: 0 } )
[
  { name: 'Samaung TV', stock: [ 80 ] },
  { name: 'Toshiba Laptop', stock: [ 67 ] },
  { name: 'Laptop Apple', stock: [ 350 ] }
]
Inventory>
```

Part 3: Update Operations

11. Update all products in the "Phone" category to add a new "features" array and increase price by 10%.

    db.products.updateMany({ category: "Phone" }, [{ $set: { features: [], price: { $multiply: ["$price", 1.1] } } }])

```
Inventory> db.products.updateMany( { category: "Phone" }, [ { $set: { features: [], price: { $multiply: ["$price", 1.10] } } }] )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}
Inventory> db.products.find( )
[
  { _id: ObjectId('589ba2fb2742a35b47dad21b'), price: 744.2 },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21c'),
    name: 'Iphone7',
    price: 18229.724103745004,
    category: 'Phone',
    vendor: 'Apple',
    stock: [ 20 ],
    quantity: 10,
    features: []
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21d'),
    name: 'Samaung TV',
    price: 11622.1,
```

    Note :

    $multiply is an **aggregation operator**, not valid directly in a regular $set operation like that.

    To use $multiply inside $set, **you must use the aggregation pipeline form of updateMany()**, which means wrapping the update in an array

12. For all products that have a stock array, add a new inventory location with 50 units.

    db.products.updateMany({stock :{$type : “array”},{$push : {stock : 50}})

```
Inventory> db.products.updateMany({stock :{$type : "array"} }, {$push : {stock : 50}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 9,
  modifiedCount: 9,
  upsertedCount: 0
}
Inventory> db.products.find( )
[
  { _id: ObjectId('589ba2fb2742a35b47dad21b'), price: 744.2 },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21c'),
    name: 'Iphone7',
    price: null,
    category: 'Phone',
    vendor: 'Apple',
    stock: [ 20, 50 ],
    quantity: 10,
    features: []
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21d'),
    name: 'Samaung TV',
    price: 11622.1,
    category: 'TV',
    vendor: { name: 'Samaung', phone: '123' },
```

13. Decrease the stock by 5 for the first stock location that has more than 50 units for "Apple" products.

```
db.products.updateOne(
  { vendor: "Apple", stock: { $gt: 50 } },
  { $inc: { "stock.$": -5 } }
);
```

```
Inventory> db.products.updateOne(
...   { vendor: "Apple", stock: { $gt: 50 } },
...   { $inc: { "stock.$": -5 } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Inventory> db.products.find({  vendor: "Apple", stock: { $gt: 50 }  })
[
  {
    _id: ObjectId('589ba2fb2742a35b47dad220'),
    name: 'Laptop Apple',
    price: 44622.476457950004,
    category: 'Laptop',
    vendor: 'Apple',
    stock: [ 295, 350, 600, 50 ],
    quantity: 2
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad222'),
    name: 'Iphone6',
    price: null,
    category: 'Phone',
    vendor: 'Apple',
    stock: [ 100, 400, 50 ],
    quantity: 199,
    features: []
  }
]
Inventory>
```

14. Use $pull to remove all stock values less than 10 from all products.

```
db.products.updateMany(
 {},
 { $pull: { stock: { $lt: 10 } } }
)
```

```
Inventory> db.products.updateMany(
...    {},
...    { $pull: { stock: { $lt: 10 } } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 11,
  modifiedCount: 3,
  upsertedCount: 0
}
```

15. Add a "lastUpdated" timestamp to all products that don't have it, then create a TTL index that expires documents after 30 days.

```
db.products.updateMany(
 { lastUpdated: { $exists: false } },
 { $set: { lastUpdated: new Date() } }
)
```

```
Inventory> db.products.updateMany(
...    { lastUpdated: { $exists: false } },
...    { $set: { lastUpdated: new Date() } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 11,
  modifiedCount: 11,
  upsertedCount: 0
}
```

// 30 * 24 * 60 * 60 = 2592000 seconds)

```
db.products.createIndex(
  { lastUpdated: 1 },
  { expireAfterSeconds: 2592000 }
)
```

```
Inventory> db.products.createIndex(
...     { lastUpdated: 1 },
...     { expireAfterSeconds: 2592000 }
... )
lastUpdated_1
```

```
Inventory> db.products.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  {
    v: 2,
    key: { _fts: 'text', _ftsx: 1 },
    name: 'name_text',
    weights: { name: 1 },
    default_language: 'english',
    language_override: 'language',
    textIndexVersion: 3
  },
  {
    v: 2,
    key: { lastUpdated: 1 },
    name: 'lastUpdated_1',
    expireAfterSeconds: 2592000
  }
]
Inventory>
```

Part 4: Indexes and Performance

16. Create a compound index for category and price, then query using it to verify performance.

db.products.createIndex({ category: 1, price: 1 })

```
Inventory> db.products.createIndex({ category: 1, price: 1 })
category_1_price_1
Inventory> db.products.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  {
    v: 2,
    key: { _fts: 'text', _ftsx: 1 },
    name: 'name_text',
    weights: { name: 1 },
    default_language: 'english',
    language_override: 'language',
    textIndexVersion: 3
  },
  {
    v: 2,
    key: { lastUpdated: 1 },
    name: 'lastUpdated_1',
    expireAfterSeconds: 2592000
  },
  { v: 2, key: { category: 1, price: 1 }, name: 'category_1_price_1' }
]
Inventory>
```

db.products.find({ category: "Phone", price: { $gt: 10000 } }).explain("executionStats")

```
Inventory> db.products.find({ category: "Phone", price: { $gt: 10000 } }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'Inventory.products',
    parsedQuery: {
      '$and': [ { category: { '$eq': 'Phone' } }, { price: { '$gt': 10000 } } ]
    },
    indexFilterSet: false,
    queryHash: '4241127D',
    planCacheKey: '7B6B7829',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { category: 1, price: 1 },
        indexName: 'category_1_price_1',
        isMultiKey: false,
        multiKeyPaths: { category: [], price: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
```

17. Create a unique index on the product name field, then attempt to insert a duplicate product.

db.products.createIndex({ name: 1 }, { unique: true })

```
Inventory> db.products.createIndex({ name: 1 }, { unique: true })
name_1
Inventory> db.products.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  {
    v: 2,
    key: { _fts: 'text', _ftsx: 1 },
    name: 'name_text',
    weights: { name: 1 },
    default_language: 'english',
    language_override: 'language',
    textIndexVersion: 3
  },
  {
    v: 2,
    key: { lastUpdated: 1 },
    name: 'lastUpdated_1',
    expireAfterSeconds: 2592000
  },
  { v: 2, key: { category: 1, price: 1 }, name: 'category_1_price_1' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
Inventory>
```

db.products.insertOne({ name: "Catel", price: 3000})

```
Inventory> db.products.insertOne({ name: "Catel", price: 3000})
MongoServerError: E11000 duplicate key error collection: Inventory.products index: name_1 dup key: { name: "Catel" }
Inventory>
```

18. Create a text index on the product name and use it to search for products containing "phone".

db.products.createIndex({ name: "text" })

```
Inventory> db.products.createIndex({ name: "text" })
name_text
Inventory> db.products.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  {
    v: 2,
    key: { _fts: 'text', _ftsx: 1 },
    name: 'name_text',
    weights: { name: 1 },
    default_language: 'english',
    language_override: 'language',
    textIndexVersion: 3
  },
  {
    v: 2,
    key: { lastUpdated: 1 },
    name: 'lastUpdated_1',
    expireAfterSeconds: 2592000
  },
  { v: 2, key: { category: 1, price: 1 }, name: 'category_1_price_1' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
```

db.products.find({ $text: { $search: "phone" } })

```
Inventory> db.products.find({ $text: { $search: "phone" } })
[
  {
    _id: ObjectId('589ba2fb2742a35b47dad224'),
    name: 'Samaung Phone',
    price: null,
    category: 'Phone',
    vendor: 'Samsumg',
    stock: [ 99, 999, 50 ],
    quantity: 230,
    features: [],
    lastUpdated: ISODate('2025-04-04T19:14:06.936Z')
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21f'),
    name: 'Sony Phone',
    price: null,
    category: 'Phone',
    vendor: 'Sony',
    stock: [ 50 ],
    quantity: 190,
    features: [],
    lastUpdated: ISODate('2025-04-04T19:14:06.936Z')
  }
]
Inventory> db.products.find({ $text: { $search: "phone" } }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'Inventory.products',
    parsedQuery: {
      '$text': {
        '$search': 'phone',
        '$language': 'english',
```

Lab 1 cont.

• select products with price greater than 1000 and less than 5000.

```
db.products.find({
        price : {$gt:1000 , $lt:5000}
})
```

```
Inventory> db.products.find({ price: { $gt: 1000, $lt: 5000 } });

Inventory>
```

Or

```
db.products.find({
        $and : [
                {price : {$gt:1000}},
                {price : {$lt:5000}}
        ]
})
```

```
Inventory> db.products.find({ $and: [ { price: { $gt: 1000 } }, { price: { $lt: 5000 } }] })

Inventory>
```

• display products which have phone number for vendor "using 2 ways "

```
db.products.find({
        “vendor.phone” : {$exists :1}
})
```

```
Inventory> db.products.find({ "vendor.phone" : {$exists :1}})
[
  {
    _id: ObjectId('589ba2fb2742a35b47dad21d'),
    name: 'Samaung TV',
    price: 11122.1,
    category: 'TV',
    vendor: { name: 'Samaung', phone: '123' },
    stock: [ 5, 70, 80, 34 ],
    quantity: 5
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21e'),
    name: 'Toshiba Laptop',
    price: 11122.1,
    category: 'Laptop',
    vendor: { name: 'Toshiba', phone: '011111321' },
    stock: [ 55, 67, 23, 1 ],
    quantity: 80
  }
]
Inventory>
```

Or

db.products.find({ "vendor.phone" : {$type :"string"}})

```
Inventory> db.products.find({ "vendor.phone" : {$type :"string"}})
[
  {
    _id: ObjectId('589ba2fb2742a35b47dad21d'),
    name: 'Samaung TV',
    price: 11122.1,
    category: 'TV',
    vendor: { name: 'Samaung', phone: '123' },
    stock: [ 5, 70, 80, 34 ],
    quantity: 5
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21e'),
    name: 'Toshiba Laptop',
    price: 11122.1,
    category: 'Laptop',
    vendor: { name: 'Toshiba', phone: '011111321' },
    stock: [ 55, 67, 23, 1 ],
    quantity: 80
  }
]
Inventory>
```

• **display products which are available in 4 stocks at the same time.**

```
db.products.find({
    stock :{$size : 4}
})
```

```
Inventory> db.products.find({ stock: { $size: 4 } })
[
  {
    _id: ObjectId('589ba2fb2742a35b47dad21d'),
    name: 'Samaung TV',
    price: 11122.1,
    category: 'TV',
    vendor: { name: 'Samaung', phone: '123' },
    stock: [ 5, 70, 80, 34 ],
    quantity: 5
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21e'),
    name: 'Toshiba Laptop',
    price: 11122.1,
    category: 'Laptop',
    vendor: { name: 'Toshiba', phone: '011111321' },
    stock: [ 55, 67, 23, 1 ],
    quantity: 80
  }
]
Inventory>
```

• **increase all products by 500 EGP.**

```
db.products.updateMany({},{
```

$inc: {price : 500}
})

```
Inventory> db.products.updateMany({}, { $inc: { price: 500 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 11,
  modifiedCount: 11,
  upsertedCount: 0
}
Inventory> db.products.find()
[
  { _id: ObjectId('589ba2fb2742a35b47dad21b'), price: 744.2 },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21c'),
    name: 'Iphone7',
    price: 16572.47645795,
    category: 'Phone',
    vendor: 'Apple',
    stock: [ 20, 70 ],
    quantity: 10
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21d'),
    name: 'Samaung TV',
    price: 11622.1,
    category: 'TV',
    vendor: { name: 'Samaung', phone: '123' },
    stock: [ 5, 70, 80, 34 ],
    quantity: 5
  },
```

- **replace stock #30 with #60 in all products.**

db.products.updateMany({stock:30},{$set : {"stock.$":60}})

```
Inventory> db.products.updateMany({stock:30},{$set : {"stock.$":60}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Inventory> db.products.find()
[
  { _id: ObjectId('589ba2fb2742a35b47dad21b'), price: 744.2 },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21c'),
    name: 'Iphone7',
    price: 16572.47645795,
    category: 'Phone',
    vendor: 'Apple',
    stock: [ 20, 70 ],
    quantity: 10
  },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21d'),
    name: 'Samaung TV',
    price: 11622.1,
    category: 'TV',
    vendor: { name: 'Samaung', phone: '123' },
    stock: [ 5, 70, 80, 34 ],
    quantity: 5
  },
```

- **remove stock 70 from all products.**

db.products.updateMany(
{stock:70},
{$pull : {stock:70}})

```
Inventory> db.products.updateMany({stock:70},{$pull : {stock:70}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}
Inventory> db.products.find()
[
  { _id: ObjectId('589ba2fb2742a35b47dad21b'), price: 744.2 },
  {
    _id: ObjectId('589ba2fb2742a35b47dad21c'),
    name: 'Iphone7',
    price: 16572.47645795,
    category: 'Phone',
    vendor: 'Apple',
    stock: [ 20 ],
    quantity: 10
  },
```

• **display only product name and vendor phone number.**

db.products.find({},{name:1,”vendor.phone”:1,_id:0})

```
Inventory> db.products.find({},{name:1,"vendor.phone":1,_id:0})
[
  {},
  { name: 'Iphone7' },
  { name: 'Samaung TV', vendor: { phone: '123' } },
  { name: 'Toshiba Laptop', vendor: { phone: '011111321' } },
  { name: 'Sony Phone' },
  { name: 'Laptop Apple' },
  { name: 'LG TV' },
  { name: 'Iphone6' },
  { name: 'HP Laptop' },
  { name: 'Samaung Phone' },
  { name: 'Catel' }
]
```

Or

db.products.find({name:{$exists:1} , "vendor.phone" : {$exists:1}},{name:1,"vendor.phone":1,_id:0})

```
Inventory> db.products.find({name:{$exists:1} , "vendor.phone" : {$exists:1}},{name:1,"vendor.phone":1,_id:0})
[
  { name: 'Samaung TV', vendor: { phone: '123' } },
  { name: 'Toshiba Laptop', vendor: { phone: '011111321' } }
]
Inventory>
```

• **display the most expensive product.**

db.products.find().sort({ price: -1 }).limit(1);

```
Inventory> db.products.find().sort({ price: -1 }).limit(1);
[
  {
    _id: ObjectId('589ba2fb2742a35b47dad220'),
    name: 'Laptop Apple',
    price: 44622.476457950004,
    category: 'Laptop',
    vendor: 'Apple',
    stock: [ 300, 350, 600 ],
    quantity: 2
  }
]
Inventory>
```