

MONGODB

1. Create a Mongodb Database named “Inventory”.

use Inventory

```
test> use shop  
switched to db shop
```

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]
}
]);
```

```

shop> 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",

```

```

...     "_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]
...   }
... ]);
...
{
  acknowledged: true,
  insertedIds: { '_id': 1, '1': 2, '2': 3, '3': 4, '4': 5 }
}

```

3. Display all documents in the collection product.

`db.products.find()`

```
shop> db.Products.find()
[
  {
    _id: 1,
    name: 'xPhone',
    price: 799,
    releaseDate: ISODate('2011-05-14T00:00:00.000Z'),
    spec: { ram: 4, screen: 6.5, cpu: 2.66 },
    color: [ 'white', 'black' ],
    storage: [ 64, 128, 256 ]
  },
  {
    _id: 2,
    name: 'xTablet',
    price: 899,
    releaseDate: ISODate('2011-09-01T00:00:00.000Z'),
    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('2015-01-14T00:00:00.000Z'),
    spec: { ram: 12, screen: 9.7, cpu: 3.66 },
    color: [ 'blue' ],
    storage: [ 16, 64, 128 ]
  },
  {
    _id: 4,
    name: 'SmartPad',
    price: 699,
    releaseDate: ISODate('2020-05-14T00:00:00.000Z'),
    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('2022-09-14T00:00:00.000Z'),
    spec: { ram: 4, screen: 9.7, cpu: 1.66 },
    color: [ 'white', 'orange', 'gold', 'gray' ],
    storage: [ 128, 256 ]
  }
]
shop> db.Products.find({_id:2})
[
  {
    _id: 2,
    name: 'xTablet',
    price: 899,
    releaseDate: ISODate('2011-09-01T00:00:00.000Z'),
    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('2015-01-14T00:00:00.000Z'),
    spec: { ram: 12, screen: 9.7, cpu: 3.66 },
    color: [ 'blue' ],
    storage: [ 16, 64, 128 ]
  },
  {
    _id: 4,
    name: 'SmartPad',
    price: 699,
    releaseDate: ISODate('2020-05-14T00:00:00.000Z'),
    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('2022-09-14T00:00:00.000Z'),
    spec: { ram: 4, screen: 9.7, cpu: 1.66 },
    color: [ 'white', 'orange', 'gold', 'gray' ],
    storage: [ 128, 256 ]
  }
]
```

4. Display all the details of product with _id is 2.

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

```
shop> db.Products.find({_id:2})
[
  {
    _id: 2,
    name: 'xTablet',
    price: 899,
    releaseDate: ISODate('2011-09-01T00:00:00.000Z'),
    spec: { ram: 16, screen: 9.5, cpu: 3.66 },
    color: [ 'white', 'black', 'purple' ],
    storage: [ 128, 256, 512 ]
  }
]
```

5. Display the first document in the collection product.

`db.products.findone()`

```
shop> db.Products.findOne()
{
  _id: 1,
  name: 'xPhone',
  price: 799,
  releaseDate: ISODate('2011-05-14T00:00:00.000Z'),
  spec: { ram: 4, screen: 6.5, cpu: 2.66 },
  color: [ 'white', 'black' ],
  storage: [ 64, 128, 256 ]
}
```

6. Display name and price of product with _id is 5.

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

```
shop> db.Products.find({_id:5},{name:1,price:1})
[ { _id: 5, name: 'SmartPhone', price: 599 } ]
```

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
});
```

```
shop> db.Products.find({price:{ $eq: 899 }},{name:1,price:1});
[
  { _id: 2, name: 'xTablet', price: 899 },
  { _id: 3, name: 'SmartTablet', price: 899 }
]
```

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
});
```

```
shop> db.Products.find({ "spec.ram": 4 }, { name: 1, "spec.ram": 1 });
...
[
  { _id: 1, name: 'xPhone', spec: { ram: 4 } },
  { _id: 5, name: 'SmartPhone', spec: { ram: 4 } }
]
```

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
});
```

```
shop> db.Products.find({ color: { $eq: "black" } }, { name: 1, color: 1 });
...
[
  { _id: 1, name: 'xPhone', color: [ 'white', 'black' ] },
  { _id: 2, name: 'xTablet', color: [ 'white', 'black', 'purple' ] }
]
```

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
});
```

```
shop> db.Products.find({ releaseDate: { $eq: new ISODate("2020-05-14") } }, { name: 1, releaseDate: 1
});
...
[
  {
    _id: 4,
    name: 'SmartPad',
    releaseDate: ISODate('2020-05-14T00:00:00.000Z')
  }
]
```

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

```
db.Products.find({
  price: { $lt: 799 }
}, {
  name: 1,
  price: 1
});
```

```
shop> db.Products.find({ price: { $lt: 799 } }, { name: 1, price: 1 });
...
[
  { _id: 4, name: 'SmartPad', price: 699 },
  { _id: 5, name: 'SmartPhone', price: 599 }
]
```

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
});
```

```
shop> db.Products.find({ "spec.screen": { $lt: 7 } }, { name: 1, "spec.screen": 1 });
...
[ { _id: 1, name: 'xPhone', spec: { screen: 6.5 } } ]
```

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
});
```

```
shop> db.Products.find({ storage: { $lt: 128 } }, { name: 1, storage: 1 });
...
[
  { _id: 1, name: 'xPhone', storage: [ 64, 128, 256 ] },
  { _id: 3, name: 'SmartTablet', storage: [ 16, 64, 128 ] }
]
```

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

```
db.Products.find({
  price: { $in: [599, 799] }
}, {
  name: 1,
  price: 1
});
```

```
shop> db.Products.find({ price: { $in: [599, 799] } }, { name: 1, price: 1 });
...
[
  { _id: 1, name: 'xPhone', price: 799 },
  { _id: 5, name: 'SmartPhone', price: 599 }
]
```


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
});
```

```
shop> db.Products.find({ color: { $in: ["black", "white"] } }, { name: 1, color: 1 });
...
[
  { _id: 1, name: 'xPhone', color: [ 'white', 'black' ] },
  { _id: 2, name: 'xTablet', color: [ 'white', 'black', 'purple' ] },
  {
    _id: 4,
    name: 'SmartPad',
    color: [ 'white', 'orange', 'gold', 'gray' ]
  },
  {
    _id: 5,
    name: 'SmartPhone',
    color: [ 'white', 'orange', 'gold', 'gray' ]
  }
]
```

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

```
db.Products.find({
  price: { $nin: [599, 799] }
}, {
  name: 1,
  price: 1
});
```

```
shop> db.Products.find({ price: { $nin: [599, 799] } }, { name: 1, price: 1 });
...
[
  { _id: 2, name: 'xTablet', price: 899 },
  { _id: 3, name: 'SmartTablet', price: 899 },
  { _id: 4, name: 'SmartPad', price: 699 }
]
```

17. Display documents where the color array doesn't have an element that either "black" or "white":

```
db.Products.find({
  color: { $nin: ["black", "white"] }
}, {
  name: 1,
  color: 1
});
```

```
shop> db.Products.find({ color: { $nin: ["black", "white"] } }, { name: 1, color: 1 });
...
[ { _id: 3, name: 'SmartTablet', color: [ 'blue' ] } ]
```

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
});
```

```
shop> db.Products.find({ $and: [ { price: 899 }, { color: { $in: ["white", "black"] } } ] }, { name: 1, price: 1, color: 1 });
...
[
  {
    _id: 2,
    name: 'xTablet',
    price: 899,
    color: [ 'white', 'black', 'purple' ]
  }
]
```

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
});
```

```
shop> db.Products.find({ $or: [ { price: { $lt: 699 } }, { price: { $gt: 799 } } ] }, { name: 1, price: 1 });
...
[
  { _id: 2, name: 'xTablet', price: 899 },
  { _id: 3, name: 'SmartTablet', price: 899 },
  { _id: 5, name: 'SmartPhone', price: 599 }
]
```

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
});
```

```
shop> db.Products.find({}, { name: 1, spec: 1 }).sort({ "spec.ram": 1 });
...
[
  { _id: 1, name: 'xPhone', spec: { ram: 4, screen: 6.5, cpu: 2.66 } },
  {
    _id: 5,
    name: 'SmartPhone',
    spec: { ram: 4, screen: 9.7, cpu: 1.66 }
  },
  {
    _id: 4,
    name: 'SmartPad',
    spec: { ram: 8, screen: 9.7, cpu: 1.66 }
  },
  {
    _id: 3,
    name: 'SmartTablet',
    spec: { ram: 12, screen: 9.7, cpu: 3.66 }
  },
  {
    _id: 2,
    name: 'xTablet',
    spec: { ram: 16, screen: 9.5, cpu: 3.66 }
  }
]
```

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
});
```

```
shop> db.Products.find({ releaseDate: { $exists: 1 } }, { name: 1, releaseDate: 1 }).sort({ releaseDate: -1 });
...
[
  {
    _id: 5,
    name: 'SmartPhone',
    releaseDate: ISODate('2022-09-14T00:00:00.000Z')
  },
  {
    _id: 4,
    name: 'SmartPad',
    releaseDate: ISODate('2020-05-14T00:00:00.000Z')
  },
  {
    _id: 3,
    name: 'SmartTablet',
    releaseDate: ISODate('2015-01-14T00:00:00.000Z')
  },
  {
    _id: 2,
    name: 'xTablet',
    releaseDate: ISODate('2011-09-01T00:00:00.000Z')
  },
  {
    _id: 1,
    name: 'xPhone',
    releaseDate: ISODate('2011-05-14T00:00:00.000Z')
  }
]
```

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({
  name: 1,
  price: 1
});
```

```
shop> db.Products.find({ 'price': { $exists: 1 } }, { name: 1, price: 1 }).sort({ name: 1, price: 1 });
...
[
  { _id: 4, name: 'SmartPad', price: 699 },
  { _id: 5, name: 'SmartPhone', price: 599 },
  { _id: 3, name: 'SmartTablet', price: 899 },
  { _id: 1, name: 'xPhone', price: 799 },
  { _id: 2, name: 'xTablet', price: 899 }
]
```

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);
```

```
shop> db.Products.find({}, { name: 1, price: 1 }).sort({ price: -1, name: 1 }).limit(1);  
...  
[ { _id: 3, name: 'SmartTablet', price: 899 } ]  
shop> 
```