1.

```
db.school.find({"science": {$exists:false}})          (exit or not)

db.school.find({"science" : null});

db.school.find({"science": {$type:"null"}})
```

2.

```
db.tShirtShop.find({"color": "white"});

db.tShirtShop.find({"color": {$eq:"white"}});

//both will provides same result
```

3. (provides only that values in which price includes)

```
db.tShirtShop.find({price:{$gt:500}})

db.tShirtShop.find({price:{$lte:500}})
```

4. // **$in**

```
db.tShirtShop.find({color:{$in:["white","black"]}})
```

5. // **$nin**

```
db.tShirtShop.find({color:{$ne:"black"}})

db.tShirtShop.find({color:{$nin:["black","white"]}})
```

6. // **$and** (only works when both cond. are true)

```
db.tShirtShop.find({$and:{color:"white"},{price:${gt:500}} })
```

7.

//db.student.find({score:{math: 230, science: 234}})

.

8. // **$or** (atleast 1 cond. true)

db.tShirtShop.find({$or:{color:"yellow"},{price:{$gt:800}}})

9. // **$nor**

db.tShirtShop.find({$nor:[{color:"green"},{price:{$gt:600}}] })

10. // **$not** (in which videos doesn't contains provides that values too)

db.tShirtShop.find({ db.playlists.find({videos:{$not:{$gt:50}}})

11. // **$expr** (dono ko compare krega jis document me prevbal jada hoga currBal se vo Output krega)

db.bank.find({$expr:{$gt:["$prevBalance","$currBalance"]}})

12. // (it provides only this Schema matched documents)

db.student.find({$jsonSchema:{required:["contact_no"],properties:{contact_no:{bsonType:"double"}}}})

13. // (For Update documents)

```
db.student.updateOne({"name":"A"},{$set:{"name":"Amit"}})
```

14. // (For Update Many)

```
db.student.updateMany({"age":"3"},{$set:{"age":5}})
```

15. // (replaceOne)

```
db.student.replaceOne({"age":"3"},{"age":5})
```

16.

// (Date updated)

```
db.student.updateMany({name:"Amit"},{$currentDate:{lastUpdated:true}})
```

17.

// (ADD or Update Properties in all documents)

```
db.student.updateMany({},{$set:{age:5}})
```

18.

// **$unset** (delete Properties from document)

```
db.student.updateOne({}, {$unset:{age:1}})
```

19.

// **$rename** (rename property)

```
db.student.updateOne({},{$rename:{name:"myname"}})
```

20.

// **$inc** (for increase/decrease value)

```
db.student.updateOne({myname:"Amit"},{$inc:{amount:+5000}})
```

21.

// **$inc** (for increase/decrease prop. value)

```
db.student.updateOne({myname:"Amit"},{$inc:{amount:-5000}})
```

22.

// **$inc** (update properties from array documents)

```
db.student.updateOne({myname:"B"},{$inc:{productInCart.1.quantity:+2}})
```

23.

// **$max /$min** (updates the value of the field to a specified value if the specified value is greater than the current value of the field)

```
db.studentMarks.updateOne({name:"kumar"},{$max:{marks:60}})
```

24.

// **$setOnInsert** /upsert (insert only if document doesn't exists before) db.studentMarks.updateOne({name:"Amisha"},{$setOnInsert:{name:"Amisha",age:17,class:12}},{upsert:true})

25.

// **.$** (this .$ sign will assign that array's index value)

db.school.updateOne({labs:4},{$set:{"labs.$":"Computer Science"}})

26.

// **.$[]**. (this .$[]. sign will add prop. in array field)(use this for add prop. in array's object's prop)

db.toppers.updateMany({},{$set:{"t_students.$[].uniform_free":true}})

------ Advanced Filtering ------

27.

// **.$[elm]**. (ADD/UPDATE document's array's object's field value, Use this for filter)

db.toppers.updateMany({},{$set:{"t_students.$[elm].grades":95}},{arrayFilters:[{"elm.grades":{$gte:95},"elm.ews":false}]})

OR (you want to add or update value or prop, query will be same for this)

db.toppers.updateMany({},{$set:{"t_students.$[elm].canteen_free":true}},{arrayFilters:[{"elm.grades":{$gte:95},"elm.ews":true}]})

28. // **$addToSet** (add more in array's, With this only 1 name will be updated at 1 time)

db.interschool.updateOne({sports_name:"Kabbadi"},{$addToSet:{players:"Aakesh"}})

29. **// $addToSet** (if name is already present, then it will skip otherwise add names)

db.interschool.updateOne({},{$addToSet:{players:{$each:["amit","rahul","lovish","Ram","ram"]}}})

30. // **$pop** (it will remove from 0 index of array $ -1 will remove last)

db.marriage.updateOne({}, { $pop: { plates: +1 } })

31. // **$pull** (it will remove a specified value or full document from an array)

db.itemsList.updateOne({},{$pull:{sweatFood:"Cake"}})

db.itemsList.updateOne({},{$pull:{fastFood:{name:"Burger"}}})

32. how to create $ insert value inside an array using push operator.

$push

db.annualFunction.updateOne({},{$push:{students:"Ram"}})

db.annualFunction.updateOne({},{$push:{students:{$each:["Ram","shyam"]}}})

33. how to remove specified values from an array.

$pullAll

```
db.annualFunction.updateOne({},{$pullAll:{students:["rohan","abhi"]}})
```

34. **diff btw $addToSet and $push**.

push opert. will add values again and again but

addToSet will add value if it is not present in an array

35. $each

Note :We can't use each opert. seprately.

Each operator will only use with any other opert. like push or addToset

36. how to insert value at a particular position inside an array.

$position

Note: it will only works with $each oper. without this we cant use it.

```
db.annualFunction.updateOne({},{$push:{students:{$each:["Arun"],$position:0}}})
```

37. how to limit a number inside an array

$slice

```
db.annualFunction.updateOne({},{$push:{students:{$each:[],$slice:-2}}})
```

38. how to sort array elements (sort: 1,-1)

Note: To use the sort modifier, it must appear with the $each modifier

We can't use it without each modifier

it will sort Capital letter first

```
db.annualFunction.updateOne({},{$push:{students:{$each:["a","Z","A","B","z","c"],$sort:1}}})
```

39. how to delete parent array element if nested array element matched the specified condition

```
db.classes.updateOne({},{$pull:{students:{subjects:{$elemMatch:{marks:99}}}}})
```

40. how to remove all elements from an array

```
db.demo.updateOne({},{$set:{myArray:[]}})
```

41. how to delete docuemnts stored inside nested array

$[]

```
db.classes.updateOne({},{$pull:{"students.$[].subjects":{subjectName:"Science"}}});
```

42. how to delete documents stored inside nested array if both array matched specified conditions

$[elm]

```
[
 {
   _id: ObjectId("6342e916e518a19c3b11f6a5"),
   class: '10 B',
   students: [
     {
       studentName: 'Ram',
       subjects: [ { subjectName: 'Maths', marks: 90 } ]
     },
     {
       studentName: 'Aman',
       subjects: [ { subjectName: 'Maths', marks: 80 } ]
     }
   ]
 }
]
```

```
db.classes2.updateOne({},{$pull:{"students.$[elm].subjects":{subjectName:"Maths"}}},{arrayFilters:[{"el
m.studentName":"Ram"}] })
```

output:

```
[
 {
   _id: ObjectId("6342e916e518a19c3b11f6a5"),
   class: '10 B',
   students: [
     { studentName: 'Ram', subjects: [(deletedOne after that)] },
     {
       studentName: 'Aman',
    subjects: [ { subjectName: 'Maths', marks: 80 } ]
   }
  ]
 }
]
```

43. how to update value stored inside a document in nested array

```
{
  _id: ObjectId("6342e916e518a19c3b11f6a5"),
  class: '10 B',
  students: [
   { studentName: 'Ram', subjects: [] },
   {
    studentName: 'Aman',
    subjects: [ { subjectName: 'Maths', marks: 90 } ]
   }
  ]
 }
```

```
db.classes2.updateOne({},{$set:{"students.$[elm].subjects.$[elm2].marks":95}},{arrayFilters:[{"elm.studentName":"Aman"},{"elm2.subjectName":"Maths"}]})
```

Output:

```
{
  _id: ObjectId("6342e916e518a19c3b11f6a5"),
  class: '10 B',
  students: [
   { studentName: 'Ram', subjects: [] },
   {
        studentName: 'Aman',
        subjects: [ { subjectName: 'Maths', marks: 95 } ]
      }
   ]
}
```

44. how to store string Object Array Bollean Null Min& Maxkey in mongodb

```
db.BinaryBsonTypes.find({address:{$type:"object"}})
db.BinaryBsonTypes.find({names:{$type:"array"}})
db.BinaryBsonTypes.find({Indian:{$type:"bool"}})
db.BinaryBsonTypes.find({Bottelwater:{$type:"null"}})

db.BinaryBsonTypes.insertOne({min_key:MinKey()})
db.BinaryBsonTypes.find({min_key:{$type:"minKey"}})

db.BinaryBsonTypes.insertOne({max_key:MaxKey()})
b.BinaryBsonTypes.find({max_key:{$type:"maxKey"}})
```

45. how to delete a collection

```
db.collectionName.drop()
```

46. how to store and execute javascript code in mongodb

```
db.executeJscode.insertOne({JsCode:"(function f(){return \"Hello\"}());"})
db.executeJscode.aggregate([{$match:{JsCode:{$type:"string"}}},{$project:{output:{$function:{body:function(x){ return eval(x)},args:["$JsCode"],lang:"js"}}}}])
```

47. how to Retrieve Particular Attributes from mongodb

```
db.bankAccounts.find({},{balance:1,_id:0})
```

(it will show only 1(balance) prop. from docuemnt)

48. how to convert string to date

```
db.date.aggregate([{$match:{myDateInstring:{$type:"string"}}},
{$addFields:{myDate:{$dateFromString:{dateString:"$myDateInstring",timezone:"Asia/Kolkata",format:"%m-%d-%Y"}}}}])
```

49. One to one relationship in mongodb

```
db.users.aggregate([{$lookup:{from:"aadhars",localField:"aadharsDetails",foreignField:"_id",as:"myAadharDetails"}}])
```

// if we have to show only few things,

```
db.users.aggregate([{$lookup:{from:"aadhars",localField:"aadharsDetails",foreignField:"_id",as:"myAadharDetails"}},{$project:{name:0,aadharsDetails:0}}])
```

50. One to many relationship in mongodb

```
student_details> db.debitCards.find()

[
 {
   _id: ObjectId("6348556ef5bc79b1a64d8ddf"),
   cardNumber: '123456789',
   exp: '11/28'
 },
 {
   _id: ObjectId("6348556ef5bc79b1a64d8de0"),
   cardNumber: '123456789',
   exp: '10/30'
 },
 {
   _id: ObjectId("6348556ef5bc79b1a64d8de1"),
   cardNumber: '123456789',
   exp: '12/26'
 }
]
```

And

```
student_details> db.user.find()
[
 {
   _id: ObjectId("634854f4f5bc79b1a64d8ddd"),
   name: 'Mike',
   products: [
     { pid: ObjectId("6348556ef5bc79b1a64d8ddf") },
     { pid: ObjectId("6348556ef5bc79b1a64d8de0") }
   ]
 },
 { _id: ObjectId("634854f4f5bc79b1a64d8dde"), name: 'Alien' }
]
```

```
db.user.aggregate([{$match:{name:"Mike"}},{$lookup:{from:"debitCards",localField:"products.pid",forei
gnField:"_id",as:"mydebitcardDetails"}}])
```

OR

```
db.user.aggregate([{$match:{name:"Mike"}},{$lookup:{from:"debitCards",localField:"products.pid",forei
gnField:"_id",as:"mydebitcardDetails"}},{$project:{products:0}}])
```

Output:

```
[
  {
    _id: ObjectId("634854f4f5bc79b1a64d8ddd"),
    name: 'Mike',
    mydebitcardDetails: [
      {
        _id: ObjectId("6348556ef5bc79b1a64d8ddf"),
        cardNumber: '123456789',
        exp: '11/28'
      },
      {
        _id: ObjectId("6348556ef5bc79b1a64d8de0"),
        cardNumber: '123456789',
        exp: '10/30'
      }
    ]
  }
]
```

----

60. Many to Many relationship

```
db.enrollment.aggregate([{$lookup:{from:"subjects",localField:"studetnId",foreignField:"_id",as:"studentdetails"}},{$lookup:{from:"subjects",localField:"subjectId",foreignField:"_id",as:"subjectDetail"}}])
```

2nd method

(student name wale me subject ki id fill kr di)

(subject wale me student name ki id fill kr di)

```
db.students.updateOne({name:"Abhi"},{$set:{subject_enrolled:[{firstSubject:ObjectId("63490608136a1426aa85ae54")}]}})
```

```
db.subjects.updateOne({name:"science"},{$set:{student_enrolled:[{firstStudent:ObjectId("634905bf136a1426aa85ae52") }]}})
```

61. ------- allOf -------

Field must match all specified schemas

62. -------- anyOf --------

Field must match at least one of the specified schemas

63. -------- oneOf -------

Field must match exactly one of the specified schemas

64. distinct

```
class:"10"

class:"11"

class:"11"
```

```
db.school.distinct("class")
[10,11]
```

67. how to rename a collection name

```
db.customer.renameCollection("customers")
```