Ejemplos MongoDB

Básicos de MongoDB con BD "amigos"

1. Obtener los datos de la colección ordenados por nombre

```
db.amigos.find().sort({"nombre":1})
```

2. Visualizar los datos de manera descendente por nombre

```
db.amigos.find().sort({"nombre":-1})
```

3. Buscar el amigo de nombre Maleni

```
db.amigos.find({nombre:"Marleni"})
```

4. Visualizar el teléfono de Marleni

```
db.amigos.find({nombre:"Marleni"},{"teléfono":1})
```

5. Visualizar nombre y nota de los alumnos de 1DAM

```
db.amigos.find({curso:"1DAM"},{"nombre":1,"nota":1})
```

6. Visualizar cuantos alumnos hay en 1DAM y 2DAM

```
db.amigos.find({"curso":"1DAM"}).count()

db.amigos.find({"curso":"2DAM"}).count()
```

7. Obtener los alumnos con nota mayor o igual que 6

```
db.amigos.find({"nota":{$gte:6}})
```

8. Obtener los alumnons 1DAM con notas entre 7 y 9 incluidos

```
db.amigos.find({curso:"1DAM","nota":{$gte:7,$lte:9}})
```

9. Obtener los documentos con una nota distinta de 7

```
db.amigos.find({"nota":{$ne:7}})
```

10. Obtener los documentos cuya nota sea 5,7 y 8 visualizando solo el nombre y el curso

```
db.amigos.find({$or:[{nota:5},{nota:7},{nota:8}]})
```

11. Visualizar los documentos de 1DAM que tienen una nota superior a 7

```
db.amigos.find({$and:[{curso:"1DAM"},{"nota":{$gt:7}}]})
```

12. Visualizar los que tienen de nombre Ana o Marleni

```
db.amigos.find({$or:[{nombre:"Ana"},{nombre:"Marleni"}]})
```

13. Visualizar los que son de 2DAM y tienen una nota =7

```
db.amigos.find({$and:[{curso:"2DAM"},{nota:{$eq:7}}]})
```

14. Obtener los que tienen una nota mayor que 7

```
db.amigos.find({nota:{$gt:7}})
```

15. Visualizar nombre, curso y nota de los que su nota no sea mayor que 7

```
db.amigos.find({nota:{$gt:7}},{"nombre":1 ,"curso":1, "nota":1})
```

16. Obtener los registros que tengan nota

```
db.amigos.find({"nota":{$exists:true}})
```

17. Cambiar el nombre a Ana por Ana María

```
db.amigos.updateOne({nombre: "Ana"},{$set:{"nombre":"Ana
maria"}})
```

18. Modifica la edad de Ana María a 24 años

```
db.amigos.updateOne({nombre: "Ana maria"},{$set:{"edad":24}})
```

19. Borra la edad de Maleni

```
db.amigos.updateOne({nombre: "Marleni"},{$unset:{"edad":""}})
```

20. Incrementa la edad de Ana María en una unidad

```
db.amigos.update({nombre:"Ana maria"},{$inc:{"edad":1}})
```

21. subir la nota un punto a los de 1DAM

```
db.amigos.updatemany({curso:"1DAM"},{$inc:{"nota":1}})
```

22. borra a Maleni

```
db.amigos.deleteOne({nombre:"Marleni"})
```

23. borrar la base de datos amigos

```
db.amigos.drop()
```

Arrays con BD "libros" con operaciones de visualización y actualización.

24. Libros que tengan el tema UML

```
db.libros.find({temas:"UML"})
```

25. Libros que tengan el tema UML o Neodatis

```
db.libros.find({$or:[{temas:"UML"},{temas:"Neodatis"}]})
```

26. Libros de la editorial garceta con pvp>25 y que tengan el tema UML o Neodatis

```
db.libros.find({$and:[{$or:[{temas:"UML"},{temas:"Neodatis"}]},{
editorial:"Garceta"},{pvp:{$gte:25}}]})
```

27. Añadir el tema Mongodb al libro con código 1

```
db.libros.updateOne({codigo:1}, {$push:{"temas": "mongodb"}})
```

28. Añadir el tema base de datos a los libros que no lo tengan

```
db.libros.updateMany({libros:1},{$addToSet:{temas:"Base de
datos"}})
```

```
(la sentencia no da error, pero no inserta nada)
```

29. Añadir varios temas al array con código 1 y 2

```
db.libros.updateMany({$or:[{codigo:2},{codigo:1}]},{$push:
```

```
{"temas":{$each:["java","php","subneting"]}}})
```

30. Borrar el primer tema del libro con codigo 3

```
db.libros.updateOne({codigo:3},{$pop:{temas:1}})
```

31. Borrar de todos los libros los elementos Base de Datos y JSON si los tienen

```
db.libros.updateMany({editorial:"Garceta"},{$pullAll:{temas:["Ba
se de datos","json"]}})
```

32. Visualiza los libros de la editorial garceta con precio entre 20 y 25 (incluidos) € y que tengan el tema Socket

```
db.libros.updateMany({editorial:"Garceta"},{$pullAll:{temas:
["Base de datos","json"]}})
```

33. Baja 5 euros a los libros de la editorial garceta

```
db.libros.updateMany({editorial:"Garceta"},{$inc:{pvp: -5}})
```

Funciones de Agregado con BD "empleados"

34. Visualiza los empleados del departamento 10

```
db.emp.find({dept_no:10})
```

35. Visualiza los empleados del departamento 10 y 20

```
db.emp.find({$or:[{ dept_no:10},{ dept_no:20}]})
```

36. Obtener los empleados con salario mayor que 1300 sean VENDEDORES

```
db.emp.find({oficio:"VENDEDOR", salario:{$gte:1300}})
```

37. sube el salario de los analistas en 100€

```
db.emp.updateMany({oficio:"ANALISTA",{$inc:{salario: 100}})
```

38. Decrementa la comisión a 20€ a los que la tengan

```
db.emple.updateMany({"comisión":{$exists:true}},{$inc:{comisión:
-20}})
```

Funciones Pipeline con BD "artículos"

39. obtener las denominaciones de los artículos y la categoría convertida a mayúscula ambas

```
db.articulos.aggregate([{$project:
            {denominación:{$toUpper:  "$denominación"},
            categoría:{$toUpper:"$categoría"}}}])
```

40. Obtener la denominación en mayúsculas, el importe de las ventas y el stock actual

```
 db.articulos.aggregate([{$project:
          {articulo:{$toUpper:"$denominación"},
          importe:{$multiply:["$pvp","$uv"]},
          stockactual:  {$subtract:["$stock","$uv"]}}}])
```

41. Respecto al ejercicio anterior, si el stockactual es negativo añadir un (a_reponer true)

```
db.articulos.aggregate([{$project:
            {articulo:{$toUpper:"$denominación"},
            importe:{$multiply:["$pvp","$uv"]},
            stockactual:{$subtract:["$stock","$uv"]},
            a_reponer:{$cond:[{$lte:[{$subtract:["$stock","$uv"]},0]},
            true,false]}}}])
```

42. Por cada categoría obtener el número de artículos, el número de unidades vendidas y el total del importe

```
db.articulos.aggregate([{$group:
          {_id:"$categoría",
          contador:{$sum:1},
          sumaunidades:{$sum:"$uv"},
          totalimporte:{$sum:{$multiply:["$pvp","$uv"]}}}}])
```

43. Igual que el ejercicio anterior, pero mostrando solo la categoría deportes

```
db.articulos.aggregate([{$match:{categoría:"Deportes"}},
          {$group:
          {_id:"$categoría",
          contador:{$sum:1},
          sumaunidades:{$sum:"$uv"},
          totalimporte:{$sum:{$multiply:["$pvp","$uv"]}}}}])
```

44. Obtener el precio más caro

```
db.articulos.aggregate([{$sort:
          {pvp:-1, denominación:-1}} ,
          {$group:{_id:null,mascaro:
          {$first:"$denominación"},
          precio:{$first:"$pvp"}}}])
```

45. Obtener la suma del importe de los artículos cuta denominación empieza por M o P

```
db.articulos.aggregate([{$project:
          {primerchar:{$substr:["$denominación",0,1]},
          import:{$multiply:["$pvp","$uv"]}}},
          {$match:{primerchar:{$in:["M","P"]}}},
          {$group:{_id:1,totalimporte:{$sum: "$import"}}}])
```

46. Obtener por cada categoría el articulo con el precio más caro

```
db.articulos.aggregate([{$group:
        {_id:"$categoría", mascaro:{$max:"$pvp"}}}])
```

Funciones Pipeline con BD "trabajadores"

47. Devolver la población obtener el nombre descompuesto en nombre, apellido1 y apellido2, el primer, segundo y último oficio del array

```
db.trabajadores.aggregate([{$sort:{
"direccion.población":1}},{$project:{población:
"$direccion.poblacion",nombre:"$nombre.nomb",ape1:"$nombre.ape1"
,ape2:"$nombre.ape2",oficio1:{$arrayElemAt:["$oficios",0]},
oficio2:{$arrayElemAt:["$oficios",1]},oficioultimo:
{$arrayElemAt:["$oficios", -1]}}}])
```

48. Devolver los elementos que tienen el array de los trabajadores (oficios y primas) y los arrays concatenados

```
db.trabajadores.aggregate([{$project:{nombre:"$nombre.nomb",numO
ficios:{$size:{"$ifNull":["$oficios",[]]}},numPrimas:{$size:
{"$ifNull":["$primas",[]]}},concatenados:{$concatArrays:
["$oficios","$primas"]}}}])
```

49. Devolver el numero maximo de trabajadores y la media de los trabajadores que han tenido el oficio de analista

```
db.trabajadores.aggregate([{$match:{oficios:"Analista"}},{$group
:{_id:"analista",contador:{$sum:1},media:{$avg:"$edad"}}}])
```