***** docs de Colección "palabras" (para consulta de T.26 aggregate y del Libro)**

-------- EJ palabras

```
db.createCollection("palabras");

db.palabras.insert( // creamos un documento con la palabra “tomate”
{ "_id":{"str":"52d87454483398c8f24222"}, // si no das tú el valor, genera un valor único
(índice y clave únicos) "palabra":"tomate",
"primera":"t",
"ultima":"e",
"tamaño":6,
"letras":["t", "o","m","a", "t","e"],
"estadis": {"vocales":3, "consonantes":3},
"caractsets":[
{"tipo":"consonantes","caracts":["t","m"]},
{"tipo":"vocales","caracts":["e","a"]}]
})

db.palabras.insert(
{ "_id":{"str":"52d87454483398c8f2429277"}, "palabra":"the",
"primera":"t",
"ultima":"e",
"tamaño":3,
"letras":["t","h","e"],
"estadis": {"vocales":1, "consonantes":2}, "caractsets":[
{"tipo":"consonantes","caracts":["t","h"]}, {"tipo":"vocales","caracts":["e"]}]
})

db.palabras.count();

db.getCollection('aficiones').find({})  // todos los docs
```

***** aggregate: Definir los PASOS antes de codificar**

---- EJ: (en prac-d-MongoDB >> log-MongoDB.org)

```
db.micole.insert([      // cuatro docus
   {a:1, b:2, c:3},
   {a:1, b:2, c:4},
   {a:0, b:2, c:3},
   {a:3, b:2, c:4}
])

// Enunciado:  Encontrar duplicados
  Queremos tener los IDs de los documentos que tiene a y b iguales
  agrupados por esos dos campos. Y Saber cuántos hay en cada grupo.


//  PASOS a dar:
```

```
      // 1. agrupa documentos por campos que coinciden a y b y eso será el ID
      // 2. cuenta cuantos documentos hay por grupo
      // 3. crear un campo nuevo para con todos los _id de los docu de cada grupo
      // 4. solo queremos los grupos que tengan duplicados (con más de 1 doc)

db.micole.aggregate(
   { $group: {
      _id: { a: "$a", b: "$b" },
      cuenta: { $sum:  1 },
      docsID: { $push: "$_id" }   }},
   { $match: {
      cuenta: { $gt : 1 }
   }}
)

----- resultado
{
   "_id" : {
      "a" : 1.0,
      "b" : 2.0
   },
   "cuenta" : 2.0,
   "docsID" : [
      ObjectId("609a4c5dcc8485e4d3a9a8f5"),
      ObjectId("609a4c5dcc8485e4d3a9a8f6")
   ]
}
```

***** aggregate : ---> tres $PUSH: un campo, un objeto o el contenido del documento**

- EJ: Queremos, para cada Apodo, todos **los Nombres** de los documentos

```
 db.aficiones.aggregate( [
   {$group : { _id : "$Apodo", componentes : {$push : "$Nombre"} }}
   ])
```

- incluir **cada ID como objeto "$_id"** (si voy a trabajar con ellos en un programa)
  es el campo _id, que es el ID del objeto

```
 db.aficiones.aggregate( [
 {$group : { _id : "$Apodo", componentes : {$push : "$_id"} }}
  ])
```

- incluir el contenido de **cada documento: "$$ROOT"**

```
 db.aficiones.aggregate( [
 {$group : { _id : "$Apodo", componentes : {$push : "$$ROOT"} }}
])
```

***** aggregate :--- EJ: FECHA**

Uso de un aggregate para devolver el campo fecha
en forma de strings formateadas usando $dateToString

Si tengo este documento en db.micole

```
db.micole.insert( {
"_id" : 33,
"nombre" : "abc",
"precio" : 10,
"cantidad" : 2,
"miFecha" : ISODate("2025-01-01T08:15:39.736Z")
} )
```

Puedo formatear partes de la fecha:
(para evitar que salgan todo los docs que no tienen esos campos
--> quito con $exists)

```
db.micole.aggregate( [
 {$match: {miFecha: {$exists: true } } },
 {$project: {
      AñoMesDia: { $dateToString: { format: "%Y-%m-%d", date: "$miFecha" } },
       hora:    { $dateToString: { format: "%H:%M:%S:%L", date: "$miFecha" } } }
  } ]
 );
```

***** EJ: ----> Cómo recorrer un array en un campo que está dentro**
de un doc que está en un array de docs. y separar pasos usando variables

ENUNCIADO: En la colección aficiones, Quiero tener cada Apodo y sus componentes.
Después imprimir el Apodo y el Nombre y Tema de cada uno de sus componentes
PASOS:
- 1.- Contexto:¿qué estructura de documentos busco? : en modo jerárquico
- 2.- ¿Cómo la recorro a mano?
- 3.- ¿Cómo lo hago con el foreach?

```
{
var resul = db.aficiones.aggregate( [
 {$group : { _id : "$Apodo", componentes : {$push : "$$ROOT"} }}
  ]);

resul.forEach(function(ele){
    print("\n  \n --- un ele ------- Apodo : " + ele._id);
                        //arrayComp = ele.componentes;
     ele.componentes.forEach(function(comp){
       print("---  un comp ---");
       print(comp.Tema + " - - -> " + comp.Nombre); })
        });
     }
```

*** **EJ: ----> $unwind : sobre la Colección - Restaurantes**

---> En vez de tener un docu. con un array : Todos los datos de un restaurante con un array de puntuaciones. Pasar a tener muchos documentos iguales que el original, excepto que en vez del array tengo uno de sus elementos (una puntuación) en cada documento.

----> EJEMPLO: Enunciado. Queremos tres documentos: las 3 mejores puntuaciones del restaurante Morris. Todos los datos del restaurante deben aparecer en cada documento.

```
// Análisis de las funciones: Lo descomponemos en PASOS, uno por operador:
// - solo para Morris,
// - separar cada puntuación de él en un documento aparte, junto con el resto de campos,
// - clasificarlos de mayor a menor por el valor de la puntuación.
// - Solo queremos los 3 primeros documentos que obtenga

db.restaurantes.aggregate( [
{$match:{ Nombre : "Morris"}},
{ $unwind:"$puntuaciones"},
{ $sort:{ 'puntuaciones.valor':-1 } },
{$limit :3}
]);
```

----------> PISTA para el Apartado 3 b) (para nota) A quien le gusta los mismos componentes?

-> para probar: necesita docus nuevos que cumplan las condiciones pedidas

```
{  // <---- Abrir llave, para que ejecute todo en el mismo bloque
var cursor = null;

// hay docs sin Nombre -> usa exists
var cursor = db.aficiones.find({Nombre: {$exists: true}},
{"Tema":1, "Nombre":1,"Apodo":1, _id: 0}).sort({"Tema":1,"Nombre":1, "Apodo":1 });
var docu1 =cursor.next();
var docu2 =cursor.next();
var hayIgual = 0;
while(cursor.hasNext()){

. . . .

}
```

***** ---> PISTAS PRAC5 apd3 f)   Rebaja un 10%**

```
-- leo en enun

db.aficiones.find().forEach(
 function (myDoc) {
     var descuento = myDoc.Precio * 0.10;
     var porcentaje =  . . . . .; // me invento la fórmula
     if (myDoc.Puntuacion < 7 )
           pongo el precio  . . .
    myDoc.Descuento = . . . .el procentaje ;
    print(myDoc);
    db.aficiones.save(myDoc);
});
```

***** ---> PISTAS PRAC5 apd3 g) (para nota)**

```
//-------- El formato resultante de la colección porNivel será de este estilo:

 {
  "_id" : ObjectId,
  "NomCal" : "Nivel Y",
  "Componentes" : [
    {
     "Componente" : {
      "_id" : ObjectId, "Tema" : "XXX", "Apodo" : " XXX ",
      "Nombre" : " XXX",
      "Puntuacion" : YYY,
      "Precio" : YYY, "Descuento" : YYY
```

```
                },
    "ValorDeCalidad" : YYY },
  {
    "Componente" : {
     . . . .
                },
     . . . .
               }
                 ]
}
.
```

------- 2. Carga en la colección PorNivel todos los componentes
de la colección Aficiones que correspondan.

```
 var arrayNivel1 = [ ];
 . . . 2 . . . .3 . . . 4


db.aficiones.find().forEach( function (myDoc) {
  var valorCalculado = (myDoc.Puntuacion*10);

if(valorCalculado <= 10){
arrayNivel1.push( { Componente: myDoc, ValorDeCalidad: valorCalculado } )
}
else if (valorCalculado <= 30){
    . . .  . 2  . . .3 . . . 4
}


db.PorNivel.save({ NomCal: "Nivel_1", Componentes: arrayNivel1 });
   . . . . 2. . . 3 . . . 4
```