CRUD create/read/update/delete avec MongoDB avec la base de données shop

Exercices inventory

Pour les exercices suivants créer un fonction de type cursor en JS :

```
const myCursor = args => db.collection.find(args)
```

Vous pouvez également trier vos documents à l'aide de la méthode sort :

```
const myCursorSort = (args, key) => db.collection.find(args).sort(key)
```

Créez une base de données **shop** et insérez les données suivantes :

```
db.inventory.insertMany( [
   {
      "sale" : true, "price" : 0.99,
      "society" : "Alex", type: "postcard", qty: 19,
      size: { h: 11, w: 29, uom: "cm" },
      status: "A",
      tags: ["blank", "blank", "blank"], "
      year" : 2019
    },
   {
       "sale" : false,
       "price" : 1.99,
       "society" : "Alan",
       type: "journal",
       qty: 25,
       size: { h: 14, w: 21, uom: "cm" },
       status: "A",
       tags: ["blank", "red", "blank", "blank"],
       "year" : 2019
   },
    {
       "sale" : true,
       "price" : 1.5,
       "society" : "Albert",
       type: "notebook",
       qty: 50,
       size: { h: 8.5, w: 11, uom: "in" },
       status: "A",
       tags: ["gray"],
       year : 2019
   },
   {
       "sale" : true,
```

```
    "price" : 7.99,
    "society" : "Alice",
    type: "lux paper",
    qty: 100,
    size: { h: 8.5, w: 11, uom: "in" },
    status: "D",
    year : 2020
},
 {
    "sale" : true,
    "price" : 2.99,
    "society" : "Sophie",
    type: "planner",
    qty: 75,
    size: { h: 22.85, w: 30, uom: "cm" },
    status: "D",
    tags: ["gel", "blue"],
    year : 2017
},
{
    "sale" : false,
    "price" : 0.99,
    "society" : "Phil",
    type: "postcard",
    qty: 45,
    size: { h: 10, w: 15.25, uom: "cm" },
    status: "A",
    tags: ["gray"],
    year : 2018
},
{
    "sale" : true,
    "price" : 4.99,
    "society" : "Nel",
    type: "journal",
    qty: 19,
    size: { h: 10, w: 21, uom: "cm" },
    status: "B",
    tags: ["blank", "blank", "blank", "red"],
    "year" : 2019,
    level : 100
},
{
    "sale" : true,
    "price" : 4.99,
    "society" : "Alex",
```

```
        type: "journal",
        qty: 15,
        size: { h: 17, w: 20, uom: "cm" },
        status: "C",
        tags: ["blank"],
        "year" : 2019
    },
    {
        "sale" : false,
        "price" : 5.99,
        "society" : "Tony",
        type: "journal",
        qty: 100,
        size: { h: 14, w: 21, uom: "cm" },
        status: "B",
        tags: ["blank","blank", "blank", "red"],
        "year" : 2020
    },
]);
```

1. Affichez tous les articles de type journal. Et donnez la quantité total de ces articles (propriété qty). Pensez à faire un script en JS.
2. Affichez les noms de sociétés depuis 2018 ainsi que leur quantité.
3. Affichez les types des articles pour les sociétés dont le nom commence par A.

Vous utiliserez une expression régulière classique : /^A/

4. Affichez le nom des sociétés dont la quantité d'articles est supérieur à 45.

Utilisez les opérateurs supérieur ou inférieur :

```
// supérieur >=
// {field: {$gte: value} }

// supérieur
// {field: {$gt: value} }

// inférieur <=
// {field: {$lte: value} }

// inférieur <
// {field: {$lt: value} }
```

5. Affichez le nom des sociétés dont la quantité d'article(s) est strictement supérieur à 45 et inférieur à 90.
6. Affichez le nom des sociétés dont le statut est A ou le type est journal.

7. Affichez le nom des sociétés dont le statut est A ou le type est journal et la quantité inférieur strictement à 100.
8. Affichez le type des articles qui ont un prix de 0.99 ou 1.99 et qui sont true pour la propriété sale ou ont une quantité strictement inférieur à 45.
9. Affichez le nom des scociétés et leur(s) tag(s) si et seulement ces sociétés ont des tags.

Vous pouvez utiliser l'opérateur d'existance de Mongo pour vérifier que la propriété existe, il permet de sélectionner ou non des documents :

```
{ field: { $exists: <boolean> } }
```

10. Affichez le nom des sociétés qui ont au moins un tag blank.

Modification du curseurs

```
// Retourne la collection à partir du 6 documents
db.students.find().skip( 5 )
// Limite le nombre de documents
db.students.find().limit( 5 )
```

Les méthodes combinées suivantes modifiants le curseur sont équivalentes (même résultat) :

```
db.students.find().sort( { name: 1 } ).limit( 5 )
db.students.find().limit( 5 ).sort( { name: 1 } )
```

Mettre à jour un document dans une collection

Repartons de la collection inventory, la méthode updateOne permet de modifier un document selon un critère de recherche.

Remarque : pour supprimer l'ensemble des documents dans une collection utilisez la syntaxe suivante.

```
db.inventory.remove({})
```

Pour supprimer la collection elle-même, utilisez la méthode drop :

```
db.inventory.drop()
```

- Exemple de modification avec la méthode updateOne

Structure :

- critère de recherche
- Modification avec l'opérateur set

Dans l'exemple suivant on modifie le premier document trouvé dont le status vaut B :

```
db.inventory.updateOne(
   { status: "B" },
   {
     $set: { "size.uom": "cm", status: "X" },
     $currentDate: { lastModified: true }
   }
)
```

Si MongoDB ne trouve pas le document, vous pouvez ajouter la propriété upsert qui permet de créer une ligne supplémentaire dans le document.

```
db.inventory.updateOne(
   { status: "XXX" },
   {
     $set: { "size.uom": "cm", status: "SUPER" },
     $currentDate: { lastModified: true }
   },
    {"upsert": true}
)

// Nouveau document
 // { "_id" : ObjectId("5ef43c659c3a4c119caf7ef5"), "status" : "SUPER" }
```

Vous pouvez également mettre à jour plusieurs documents en même temps à l'aide de la méthode suivante :

collection.updateMany()

Comme vous êtes dans une console avec JS vous pouvez écrire :

```
const query = {status : "A"};
const update = { "$mul": { "qty": 10 } };
const options = { "upsert": false }

const updateInventory = (query, update, options) => db.inventory.updateMany(query, update, c

updateInventory(query, update, options);
```

Exercice avec forEach

La méthode forEach permet d'itérer sur une collection :

```
db.collection.find().forEach(<function>)
```

1. En utilisant la fonction forEach et la fonction find augmentez de 50% la quantité de chaque document qui a un status C ou D. Utilisez l'opérateur set.
2. Augmentez maintenant de 150% les documents ayant un status A ou B et au moins 3 blanks dans leurs tags.

Méthode unset

Vous pouvez également supprimer un champ d'un document à l'aide de l'opérateur unset, ci-dessous on supprime les champs qty et status du premier document qui match avec la recherche :

```
db.inventory.updateOne(
   { status: "XXX" },
   { $unset: { qty: "", status: "" } },
    {"upsert": true}
)
```

Notez que vous pouvez également ajouter un nouveau champ avec l'opérateur set :

```
db.inventory.updateMany(
    { tags : { $exists : true }} ,
    { $set: { super : "super" } }
 )
```

Exercice suppression d'un champ

Supprimez la propriété level se trouvant dans un document.

Vérifiez que le champ est bien supprimé.

Opérateur switch

Vous pouvez utiliser l'opérateur switch afin de modifier un document selon une liste de cas :

```
   $switch: {
      branches: [
         { case: { $eq: [ 0, 5 ] }, then: "equals" },
         { case: { $gt: [ 0, 5 ] }, then: "greater than" },
         { case: { $gt: [ { $size :  "$notes" } , 2  ], then: "less than" }
      ],
      default : "nothing"
   }
}
```

Attention, dès que l'on rentre dans un cas on sort du switch/case.

Exercice switch

L'opérateur size permet de compter le nombre d'élément dans un array. Il faut cependant pré-fixer la propriété avec un dollar :

```
{ $size :  "$tags" }
```

Ajoutez la propriété **grade** pour les documents ayant la propriété tags uniquement. Grade prendra les valeurs suivantes selon le nombre de tags :

- si le nombre de tags est strictement supérieur à 2 : A
- si le nombre de tags est strictement supérieur à 3 : AA
- Et B sinon.

db.collection.deleteOne

La méthode suivante permet de supprimer un document à l'aide d'une condition :

```
db.orders.deleteOne( { "_id" : ObjectId("563237a41a4d68582c2509da") } );
```

Exercice d'application

Insérez le document suivant dans la collection inventory :

```
db.inventory.insertOne( {
   name : "Test sur les dates",
   creationts: ISODate("2020-06-27T08:41:57.114Z"),
   expiryts: ISODate("2020-08-27T08:41:57.114Z")
})
```

Pour récupérez le dernier document inséré tapez la ligne de commande suivante :

```
db.inventory.find().sort({_id: -1}).limit(1);
```

Maintenant supprimez ce document en fonction de sa date d'expiration :

```
db.inventory.deleteOne( { "expiryts" : { $lte: ISODate("2020-08-27T08:41:57.114Z") } } );
```

Vérifiez que ce document est bien supprimé de la collection.

db.collection.deleteMany

Cette méthode permet de supprimer plusieurs documents à l'aide d'un critère de sélection.

Exercice synthèse

1. Hydratation. Créez les champs **created_at** et **expired_at** pour chaque document de la collection inventory.

Vous utiliserez la méthode ISODate pour créer une date aléatoire ainsi que l'objet Date de JS. Aidez-vous de l'exemple ci-dessous :

- Exemple

Décallage d'un jour par rapport à la date actuelle :

```
// Pour un jour
// 1 x 24 hours x 60 minutes x 60 seconds x 1000 milliseconds
let day = 1*24*60*60*1000

// Ajoute un jour à la date actuel
new Date( ISODate().getTime() + day )
```

2. Ajoutez un champ qui calcule le nombre de jours qui reste avant la suppression du document.

Vous pouvez utiliser les opérateurs suivants :

```
// Pour faire une différence entre les dates
$subtract

// Pour calculer le nombre de jour
$divide
```