Correction TP MongoDB

- Se connecter à WINDOWS
- Lancer la commande `mongod`
- Aller sur la page http ://www.lsis.org/elmouelhia/teaching.html, télécharger et décompresser l'archive `employes.rar`
- Dans la console, exécuter la commande
  `mongorestore --db gescom cheminabsoludufichier/employes.bson --port numeroPort`
- Se connecter à la base `gescom` avec la commande `mongo`

Exercice 1

Écrire les requêtes MongoDB qui permettent de :

1. afficher toutes les collections de la base

```
show collections;
```

2. afficher tous les documents de la base

```
var coll = db.getCollections()
for (var i = 0; i < coll.length; i++) {
    db.coll[i].find();
}
```

   ou si la base de données contient une seule collection

```
 db.employes.find();
```

3. compter le nombre de documents de la collection `employes`

```
 db.employes.find().count();
```

4. insérer de deux manières différentes deux employés avec les champs `nom`, `prénom` et soit `prime` soit `ancienneté`

```
db.employes.insert({nom:'Alan',prenom:'Joe',anciennete:10});
db.employes.save({nom:'Wick', prenom:'John',prime:150});
```

5. afficher la liste des employés dont le `prénom` est `David`

```
db.employes.find({prenom:'David'});
```

6. afficher la liste des employés dont le `prénom` commence ou se termine par `D`

```
db.employes.find({prenom:/^D.*|.*D$/});
```

7. afficher la liste des personnes dont le `prénom` commence par `D` et contient exactement 5 lettres

```
db.employes.find({prenom: /^D[a-z]{4}$/});
```

ou

```
db.employes.find({prenom:/^D/}).forEach(
    function(p){
        if(p.prenom.length == 5){
            print(p.prenom)
    }
})
```

8. afficher la liste des personnes dont le `prénom` commence et se termine par une voyelle

```
db.employes.find({prenom: /^[AEIOUY].*[aeiouy]$/});
```

9. afficher la liste des personnes dont le `prénom` commence et se termine par une même lettre

```
db.employes.find().forEach(function(p){
    let pre = p.prenom.toLowerCase();
    if(pre.substr(0,1)==pre.substr(pre.length-1,1)){
        print(pre)
    }
})
```

10. afficher les `nom` et `prénom` de chaque employé ayant une `ancienneté` > 10

```
db.employes.find(
    {anciennete:{$gt:10}},
    {_id:0,nom:1,prenom:1}
);
```

11. afficher les `nom` et `adresse` complète des employés ayant un attribut `rue` dans l'objet `adresse`

```
db.employes.find(
    {'adresse.rue': {$exists:true} },
    {nom:1, adresse:1}
);
```

12. incrémenter de 200 la `prime` des employés ayant déjà le champ `prime`

```
db.employes.updateMany(
    {prime:{$exists:true}},
    {$inc:{prime:200}}
);
```

ou

```
db.employes.update(
    {prime:{$exists:true}},
    {$inc:{prime:200}},
    {multi:true}
)
```

13. afficher les trois premières personnes ayant la plus grande valeur d'ancienneté

```
db.employes.find({anciennete : {$exists:true}}, {_id:0})
    .sort({anciennete: -1})
    .limit(3);
```

14. regrouper les personnes dont la ville de résidence est `Toulouse` (afficher `nom`, `prénom` et `ancienneté`)

```
db.employes.find(
    {'adresse.ville': 'Toulouse'},
    {nom:1, prenom:1, anciennete:1, _id:0}
);
```

15. afficher les personnes dont le prénom commence par `M` et la ville de résidence est soit `Foix` soit `Bordeaux`

```
db.employes.find({
        $and: [
                {prenom: /^M/},
                {
                        $or: [
                                {'adresse.ville':'Foix'},
                                {'adresse.ville':'Bordeaux'}
                        ]
                }
        ]
});
```

ou

```
db.employes.find({
    {
        prenom: { $regex: /^M.*/},
        "adresse.ville": {
            $in: ["Foix", "Bordeaux"]
        }
    }
);
```

16. mettre à jour l'adresse de `Dominique Mani` : nouvelle adresse ({ numero : 20, ville : 'Marseille', codepostal : '13015' }). Attention, il n'y aura plus d'attribut `rue` dans adresse

```
db.employes.update(
        {prenom: 'Dominique',nom: 'Mani'},
        {
            $set : {
                'adresse.numero' : 20,
                'adresse.ville' : 'Marseille',
                'adresse.codepostal' : '13015'
            },
            $unset :
                {'adresse.rue' : 1}
        }
);
```

17. attribuer une prime de 1 500 à tous les employés n'ayant pas de prime et dont la ville de résidence est différente de Toulouse, Bordeaux et Paris.

```
db.employes.updateMany(
    {$and:
        [
            {"adresse.ville":
                {$nin:["Paris","Toulouse","Bordeaux"]}
            },
            {prime:{$exists:false}}
        ]
     },
     {$set:{prime:1500}}
);
```

18. remplacer le champ `tel`, pour les documents ayant un champ tel), par un tableau nommé `téléphone` contenant la valeur du champ tel (le champ `tel` est à supprimer)

```
db.employes.find({tel: {$exists: true}}, {}).forEach(
    function(t) {
        db.employes.updateMany(
            {_id:t._id},
            {
                $push:{telephone: t.tel},
                $unset:{tel:1}
            }
        );
    }
}
```

19. créer un champ prime pour les documents qui n'en disposent pas et de l'affecter à 100 * nombre de caractère du nom de la ville

```
db.employes.find({prime: {$exists:0}}).forEach(
    function(doc){
        var length = doc.adresse.ville.length;
        var newPrime = 100*length;
        db.employes.update(
            {_id : doc._id},
            {
                $inc:
                    {prime: newPrime}
            }
        );
    }
);
```

20. créer un champ `mail` dont la valeur est égale soit à nom.prénom@formation.fr pour les employés ne disposant pas d'un champ téléphone, soit à prénom.nom@formation.fr (`nom` et `prénom` sont à remplacer par les vraies valeurs de chaque employé)

```
db.employes.find().forEach(
    function(p) {
        var email = p.nom + '.' + p.prenom + '@formation.fr'
           ;
        if (p.telephone)
            var email = p.prenom + '.' + p.nom + '@formation
               .fr';
        db.employes.updateMany(
            {_id:p._id},
            {$set:{mail: email}}
        );
    }
);
```

ou

```
db.employes.find().forEach(
    function(p){
        if(p.hasOwnProperty("telephone") == true){
            db.employes.update({_id:p._id},{$set:{email: p.
               prenom+"."+p.nom+"@formation.fr"}});
        }
        else{
            db.employes.update({_id:p._id},{$set:{email: p.
               nom+"."+p.prenom+"@formation.fr"}});
        }
    }
);
```

21. calculer et afficher la somme de l'ancienneté pour les employés disposant du même prénom

```
db.employes.aggregate(
    {$group:
        {
            _id: '$prenom',
            ancienneteCum: {$sum: '$anciennete'}
        }
    },
    {$sort: {_id: 1}}
);
```

Un grand merci pour Cédric, Gabriel, Icare et Kévin.