1. MongoDB par driver Python

L'objectif de cet exercice est d'utiliser un autre client de MongoDB Compass ou mongosh et le coder en langage Python. Cet exercice utilise le module `pymongo` qui est un driver mongoDB.

a. Dans votre console, installez le module pymongo (pip install pymongo) permettant d'utiliser les requêtes SQL vers mongoDB dans un script Python.
Puis créer un fichier tp6_ex1.py avec le contenu suivant :

```
from pymongo import MongoClient

# Connexion du client
CONNECTION_STRING = 'mongodb://localhost:27017/'
clientmongo = MongoClient(CONNECTION_STRING)
try:
   clientmongo.admin.command('ping')
   print("Connexion reussie a MongoDB")
except Exception as e:
   print(e)

# Sélection DB et récupération du curseur
col = clientmongo.local.restaurants
doc = col.find({'name':'Wendy S'})

# Recherche et affichage
print(doc[0]['restaurant_id'])
print(doc[1]['restaurant_id'])

# Fin du traitement
# Libération des ressources
doc.close()
clientmongo.close()
```

Que fait ce programme ? Tous les résultats de la recherche sur MongoDB sont ils affichés ?

b. Modifier le code Python pour afficher les 8 restaurant_id de la recherche find({'name':'Wendy S'}).
Vous pourrez vous aider de la documentation des curseurs pymongo.

c. En utilisant une fonction d'agrégation de comptage, affichez le nombre d'éléments de la collection. Plus d'informations sur le lien.

d. Modifier le document du restaurant « `Morris Park Bake Shop` » de la collection en changeant le type de cuisine : Bakery est remplacé par Italian. Affichez le résultat de la fonction de mise à jour de pymongo.

Exercice 2 : agrégations et pipeline

a. Dans moodle, récupérer en local la collection books.json puis importer-la dans mongoimport afin de créer une nouvelle collection appelée books

b. Agrégation simple :

La structure d'un document est la suivante :


```
    _id: 14,
    title: 'Coffeehouse',
    isbn: '1884777384',
    pageCount: 316,
    publishedDate: ISODate("1997-07-01T07:00:00.000Z"),
    thumbnailUrl: 'https://s3.amazonaws.com/AKIAJC5RLADLUMVRPFDQ.book-thumb-images/ash
er.jpg',
    shortDescription: 'Coffeehouse is an anthology of stories, poems and essays origin
ally published on the World Wide Web.',
    longDescription: `Coffeehouse is an anthology of stories, poems and essays origina
lly published on the World Wide Web. The purpose is to capture the zeitgeist of the we
b's creative community, and to give readers a chance to enjoy some of the best and mos
t notable original works that have appeared in this form. It showcases over forty indi
vidual web writers, among them Joseph Squier, Martha Conway, Jason Snell, David Alexan
der, Carl Steadman and Walter Miller. The intent is to show the variety and vitality o
f the web's blossoming literary "scene," and to capture the unique and highly iconocla
stic "personality" of the web community.`,
    status: 'PUBLISH',
    authors: [ 'Levi Asher', 'Christian Crumlish' ],
    categories: [ 'Miscellaneous' ]
```


Lister les titres et le nombre de pages des livres de la catégorie Web development.

On souhaite calculer le nombre total de pages de tous ces livres.


```
[
  { _id: 7, title: 'Zend Framework in Action', pageCount: 432 },
  { _id: 11, title: 'Flexible Rails', pageCount: 592 },
  { _id: 26, title: 'iBATIS in Action', pageCount: 384 },
  { _id: 29, title: 'jQuery in Action', pageCount: 376 },
  { _id: 32, title: 'Ruby for Rails', pageCount: 532 },
  { _id: 42, title: 'iPhone in Action', pageCount: 472 },
  { _id: 56, title: 'GWT in Practice', pageCount: 376 },
  { _id: 60, title: 'Ajax in Practice', pageCount: 536 },
  {
    _id: 61,
    title: 'Prototype and Scriptaculous in Action',
    pageCount: 544
  },
  { _id: 67, title: 'Wicket in Action', pageCount: 392 },
  { _id: 173, title: 'Laszlo in Action', pageCount: 552 },
  { _id: 203, title: 'iText in Action', pageCount: 688 },
  { _id: 200, title: 'Adobe AIR in Action', pageCount: 336 },
  {
    _id: 215,
    title: 'Algorithms of the Intelligent Web',
    pageCount: 368
  },
  { _id: 226, title: 'Sass and Compass in Action', pageCount: 300 },
  { _id: 335, title: 'The Engaging Web', pageCount: 325 },
  { _id: 516, title: 'Node.js in Action', pageCount: 300 }
]
```


Pour cela, additionner les valeurs des pageCount à l'aide d'une calculatrice.

Pour calculer le total de pages dans chaque catégorie de livres, on utilise maintenant l'agrégation avec les mots clés $group et $sum dans la requête suivante :
db.books.aggregate({$group:{"_id":"$categories",totalpage:{$sum:"$pageCount"}}})
qui retourne notamment :

```
{ _id: [ 'Microsoft' ], totalpage: 3654 },
{ _id: [ 'Java', 'Internet', 'Computer Graph' ], totalpage: 400 },
{ _id: [ 'Internet', 'Networking' ], totalpage: 341 },
{ _id: [], totalpage: 14432 },
{ _id: [ 'Web Development' ], totalpage: 7505 },
{ _id: [ 'Microsoft .NET' ], totalpage: 14499 }
```

$group est un opérateur de regroupement. Il crée un nouveau document avec 2 champs :

- _id contient une liste des valeurs du champ categories ($categories)
- totalpage est un champ généré contenant la liste des sommes ($sum) des différentes valeurs du champ pageCount ($pageCount)

Cette approche permet de réaliser l'équivalent de count en SQL.

L'agrégation autorise de nombreux autres opérateurs :
$avg, $min, $max, $first, $last, $push, $addToSet, $stdDevPop, $stdDevSamp
parmi lesquels : $avg calcule la moyenne, $min indique le minimum, $max indique le maximum… Plus de précisions sur les opérateurs.

Attention : l'agrégation regroupe les données dans _id par même catégorie. Il se peut que certaines noms de catégories très proches diffèrent d'une majuscule ou d'un accent. Une étape de normalisation est alors nécessaire.
La requête ci-dessous passe tous les noms de catégories en majuscule puis les trie dans l'ordre lexicographique :

```
db.users.aggregate(
  [   { $project : { categories:{$toUpper:"$_id"} , _id:0 } },
      { $sort : { categories : 1 } }
  ]
)
```

On peut également trier les résultats en dehors de l'étape de normalisation.
Ainsi, la requête suivante calcule la moyenne des pages par document pour chaque catégorie puis affiche les résultats dans l'ordre décroissant (-1) :
db.books.aggregate({$group:{"_id":"$categories",totalpage:{$avg:"$pageCount"}}},{$sort:{totalpage:-1}}). On a introduit une 2eme étape dans l'agrégation (group puis sort).

Enfin, on peut filtrer les résultats à l'aide d'un mot clé sur un champ. Par exemple, on souhaite filtrer sur la présence du mot development dans le champ short description. On insère une nouvelle étape dans le traitement à l'aide de $match :
db.books.aggregate({$match:{shortDescription:"development"}},{$group:{"_id":"$categories",totalpage:{$avg:"$pageCount"}}},{$sort:{totalpage:-1}})

En savoir plus : Aggregation with User Preference Data — MongoDB Manual

c. Pipeline d'agrégation :
Dans les exemples du paragraphe précédent, nous avons utiliser une requête aggregate comportant jusqu'à 3 commandes. Nous avons en fait créer un enchaînement de 3 étapes

(filtre, agrégation, tri). MongoDB appelle cette approche le pipeline d'agrégation composé de différents stages représentant les commandes successivement passées à aggregate.
Syntaxe : db.collection.aggregate([{ <stage> }, { <stage> }, ...])
Stage peut valoir : $match, $group, $sort et de nombreuses autres valeurs.

On va regrouper les documents suivant une clé basée sur le titre et l'auteur :
db.books.aggregate({$group:{"_id":{nom:"$title",creator:"$authors"}}})
Ce qui retourne des documents ressemblant à la collection projetée sur creator et author sans sélection. Vérifier que vous obtenez la même réponse.

```
tvertdb> db.books.aggregate({$group:{"_id":{nom:"$title",creator:"$authors"}}})
[
  {
    _id: { nom: 'Liferay in Action', creator: [ 'Richard Sezov', 'Jr' ] }
  },
  {
    _id: {
      nom: 'Learn SQL Server Administration in a Month of Lunches',
      creator: [ 'Don Jones' ]
    }
  },
  {
    _id: {
      nom: 'Clojure in Action, Second Edition',
      creator: [ 'Amit Rathore' ]
    }
  },
```

Ajoutons un stage group effectuant du comptage sur le nombre d'apparition des auteurs qui s'appelle maintenant creator sous champ de _id :
db.books.aggregate({$group:{"_id":{nom:"$title",creator:"$authors"}}},{$group:{_id:"$_id.creator",nb_auteur:{$sum:1}}})
Vérifier la réponse obtenue notamment :

```
[
  { _id: [ 'Steven Gutz' ], nb_auteur: 1 },
  { _id: [ 'Josiah Carlson' ], nb_auteur: 1 },
  { _id: [ 'Anthony Patton' ], nb_auteur: 2 },
```

d. En vous aidant des informations précédentes, construire une requête permettant d'afficher:

- le nombre total de livres dans la collection books
- le nombre total de pages de tous les livres de la collection books
- le nombre moyen de pages de livres dans chaque catégorie
- les 3 meilleurs auteurs avec le plus de livres dans la collection books
- le nombre total de livres publiés chaque année dans la collection books
- le nombre moyen et maximum de catégories par livre dans la collection books

En savoir plus : Aggregation Pipeline Stages — MongoDB Manual