TP NoSQL
MongoDB

Nom : BOUJMIL
Prénom : Zeineb

1) Téléchargement de l'archive de données

C'est un format de données qui ressemble à JSON. MongoDB stocke ses documents **en interne** sous ce type .

```
C:\Users\BAZ INFO>curl https://atlas-education.s3.amazonaws.com/sampledata.archive -o sampledata.archive
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  353M  100  353M    0     0  1398k      0  0:04:18  0:04:18 --:--:--  953k
```

2) Copie de l'archive dans le conteneur Docker

```
C:\Users\BAZ INFO>docker cp "C:\Users\BAZ INFO\sampledata.archive" adoring_pike:/sampledata.archive
Successfully copied 370MB to adoring_pike:/sampledata.archive
```

3) Restauration des données avec `mongorestore`

Utiliser `mongorestore` (qui sait lire le BSON) plutôt que `mongoimport` qui travaille surtout avec du JSON.

```
# mongorestore --archive=/sampledata.archive
2025-11-27T10:23:25.200+0000    preparing collections to restore from
2025-11-27T10:23:25.309+0000    reading metadata for sample_airbnb.listingsAndReviews from arc
hive '/sampledata.archive'
2025-11-27T10:23:26.243+0000    restoring sample_airbnb.listingsAndReviews from archive '/samp
ledata.archive'
2025-11-27T10:23:28.159+0000    sample_airbnb.listingsAndReviews  38.1MB
```

C'est l'étape où les données sont effectivement importées dans le serveur MongoDB.

```
> show dbs
admin                0.000GB
config               0.000GB
local                0.000GB
sample_airbnb        0.051GB
sample_analytics     0.009GB
sample_geospatial    0.001GB
sample_guides        0.000GB
sample_mflix         0.088GB
sample_restaurants   0.006GB
sample_supplies      0.001GB
sample_training      0.040GB
sample_weatherdata   0.002GB
```

Partie 1 – Requêtes simples (find)

Forme générale :

```
db.<collection>.find(
 { /* FILTRE */ },
 { /* PROJECTION */ }
)
.sort({ /* TRI */ })
.skip(<nbDocsASauter>)
.limit(<nbDocsMax>)
.pretty()
```

Explication :

db.<collection> : collection sur laquelle on travaille (ex. db.movies).
FILTRE : objet qui décrit les conditions de sélection des documents. Exemple de clés possibles
- champ1: valeurExacte
- champ2: { $gte: valeurMin } (>=)
- champ3: { $exists: true } (champ présent)
- champ4: { $in: [val1, val2] } (valeur dans un ensemble)

PROJECTION : objet qui décrit les champs à afficher : { champ1: 1, champ2: 1, _id: 0 }.
sort({ champ: 1 }) : tri croissant (1) ou décroissant (-1).
skip(n) : saute les n premiers résultats (pagination).
limit(n) : limite le nombre de documents renvoyés.
pretty() : mise en forme lisible dans le shell Mongo.

Partie 2 – Agrégation (aggregate)

Forme générale :

```
db.<collection>.aggregate([
 { $match: { /* FILTRE INITIAL */ } },
 { $unwind: "$<champTableau>" },
 { $project: {
    <champA>: 1,
    <champB>: 1,
    <champCalcule>: { /* expression ou opération */ }
 }},
 { $group: {
    _id: "$<champDeRegroupement>",
    <champAgg1>: { $sum: <expression> },
    <champAgg2>: { $avg: "$<champNum>" },
    <champAgg3>: { $max: "$<champNum>" }
 }},
 { $sort: { <champTri>: 1 } },
 { $skip: <nbDocsASauter> },
 { $limit: <nbDocsMax> }
])
```

Explication :

$match : filtre les documents en entrée du pipeline, comme un find.
$unwind : "éclate" un champ tableau : un document par élément du tableau.
$project : choisit les champs qui continuent dans le pipeline et permet de créer des champs calculés.
Exemple d'expressions possibles : $subtract, $add, $mod, etc.
$group : regroupe les documents par une clé (_id) et applique des fonctions d'agrégation :
- $sum : somme ou compteur (avec 1)
- $avg : moyenne
- $max / $min : max ou min
$sort : tri des résultats du pipeline.
$skip / $limit : comme après un find, mais à l'intérieur du pipeline d'agrégation.

Partie 3 – Mises à jour (updateOne / updateMany)

Forme générale :

```
db.<collection>.updateOne(
 { /* FILTRE */ },
 {
  $set:   { /* champs à ajouter ou modifier */ },
  $inc:   { /* champs numériques à incrémenter */ },
  $unset: { /* champs à supprimer */ },
  $push:  { /* ajouter un élément dans un tableau */ },
  $pull:  { /* retirer un élément d'un tableau */ }
 },
 {
  upsert: <trueOuFalse>
 }
)
```

Explication :

updateOne : modifie le premier document qui vérifie le filtre.
updateMany : modifie tous les documents qui vérifient le filtre.
FILTRE : même logique que dans find (conditions de sélection).
$set : ajoute ou met à jour des champs (ex. $set: { champ: valeur }).
$inc : incrémente des champs numériques (ex. $inc: { compteur: 1 }).
$unset : supprime les champs (ex. $unset: { champ: "" }).
$push : ajoute un élément à un tableau (ex. $push: { liste: nouvelElement }).
$pull : retire un élément d'un tableau (ex. $pull: { liste: valeurASupprimer }).
upsert : si true, insère un nouveau document si aucun document ne correspond au filtre.

Partie 4 – Requêtes complexes (find + aggregate avancé)

```
db.<collection>.find(
 {
  $and: [
   { <champ1>: { $exists: true } },
   { <champ2>: { $gte: <valMin> } },
   { <champTexte>: /^<prefixe>/ },
   { $where: "this.<champTab>.length >= <n>" }
```

```
  ]
 },
 { /* PROJECTION */ }
)
.sort({ <champTri>: -1 })
.limit(<nbDocsMax>)
```

Explication :

$and / $or / $nor : combinent plusieurs conditions logiques.
Les conditions internes sont les mêmes que dans un filtre classique (comparaisons, $exists, etc.).
Les expressions régulières (regex) s'écrivent comme /^<prefixe>/ pour "commence par <prefixe>".
$where : autorise une condition JavaScript sur le document courant (this), utile pour des tests avancés (par ex. taille d'un tableau), mais plus lent.

Partie 5 – Indexation

Création d'un index général :

```
db.<collection>.createIndex(
 {
  <champ1>: 1,
  <champ2>: -1
 },
 {
  unique: <trueOuFalse>,
  name: "<nomIndex>",
  background: <trueOuFalse>
 }
)
```

Explication :

Le premier argument décrit la clé d'index :
- <champ1>: 1 → ordre croissant dans l'index
- <champ2>: -1 → ordre décroissant (utile pour certains tris).

Le second argument contient les options :
- unique : true interdit les doublons sur la combinaison de champs indexés.
- name : nom explicite de l'index (sinon Mongo en génère un automatiquement).
- background : true permet de créer l'index en tâche de fond (important en production).

Application :

Copier `films.json` depuis mon local vers le conteneur

```
PS C:\Users\BAZ INFO> docker cp "C:\Users\BAZ INFO\Downloads\films.json" adoring_pike:/films.json
Successfully copied 587kB to adoring_pike:/films.json
```

Importation de la collection des films dans la base lesfilms

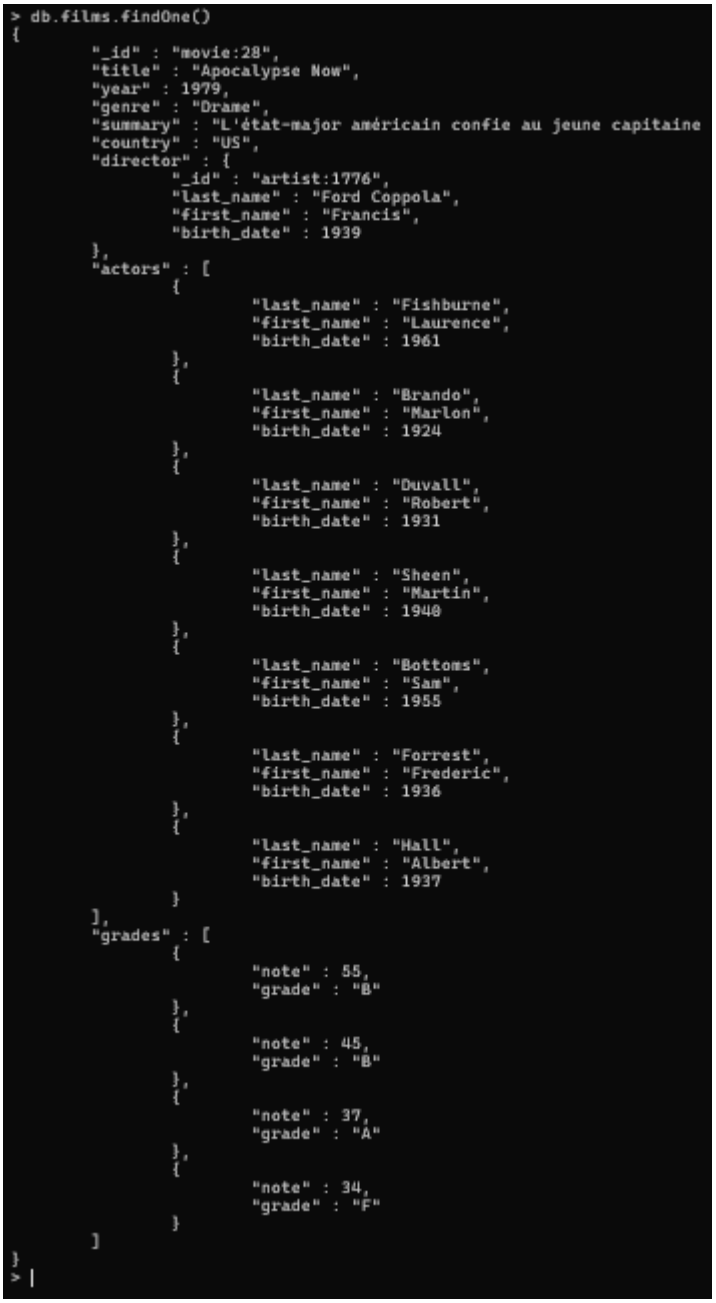
```
# mongoimport --db lesfilms --collection films --file /films.json --jsonArray
2025-11-28T23:33:27.012+0000    connected to: mongodb://localhost/
2025-11-28T23:33:30.008+0000    [########################] lesfilms.films       571KB/571KB (1
00.0%)
2025-11-28T23:33:31.023+0000    [########################] lesfilms.films       571KB/571KB (1
00.0%)
2025-11-28T23:33:31.024+0000    278 document(s) imported successfully. 0 document(s) failed to
 import.
#
```

1. Vérifier que les données ont été importées, en les comptant par exemple.

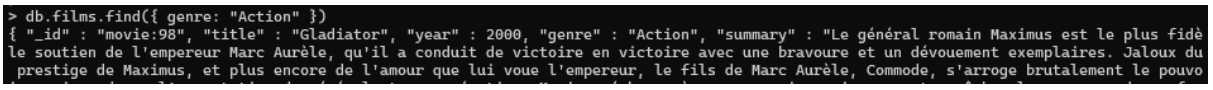
```
> use lesfilms
switched to db lesfilms
```

```
> db.films.count()
278
```

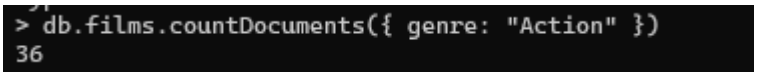
2. Avant de commencer à interroger la base de données des lesfilms, il est essentiel de bien comprendre la structure d'un document dans notre collection de films. Pour ce faire, nous allons utiliser la fonction findOne(), qui permet de récupérer un seul document. 1 db.films.findOne()



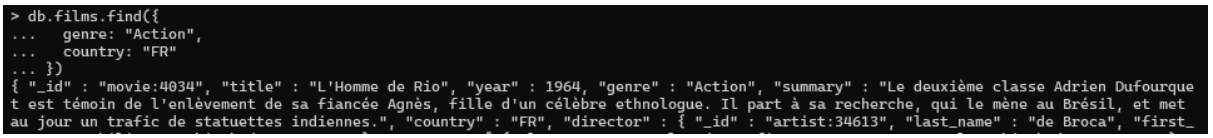
3. Afficher la liste des films d'action.



4. Afficher le nombre de films d'action.



5. Afficher les films d'action produits en France.



6. Afficher les films d'action produits en France en 1963.

```
> db.films.find({
...   genre: "Action",
...   country: "FR",
...   year: 1963
... })
{ "_id" : "movie:25253", "title" : "Les tontons flingueurs", "year" : 1963, "genre" : "Action", "summary" : "Sur son lit de mort, le Me
xicain fait promettre à son ami d'enfance, Fernand Naudin, de veiller sur ses intérêts et sa fille Patricia. Fernand découvre alors qu'
il se trouve à la tête d'affaires louches dont les anciens dirigeants entendent bien s'emparer. Mais, flanqué d'un curieux notaire et d
```

7. Jusqu'à maintenant, a chaque fois qu'un document est renvoyé, tous ses attributs sont affichés. Pour n'afficher que certains attributs, on utilise un filtre. Pour afficher les films d'action réalisées en France, sans les grades, le filtre est passé comme deuxième paramètre de la fonction find() comme ceci.

```
> db.films.find(
...   { genre: "Action", country: "FR" },
...   { grades: 0 }
... )
{ "_id" : "movie:4034", "title" : "L'Homme de Rio", "year" : 1964, "genre" : "Action", "summary" : "Le deuxième classe Adrien Dufourque
t est témoin de l'enlèvement de sa fiancée Agnès, fille d'un célèbre ethnologue. Il part à sa recherche, qui le mène au Brésil, et met
au jour un trafic de statuettes indiennes.", "country" : "FR", "director" : { "_id" : "artist:34613", "last_name" : "de Broca", "first_
name" : "Philippe", "birth_date" : 1933 }, "actors" : [ { "last_name" : "Belmondo", "first_name" : "Jean-Paul", "birth_date" : 1933 },
```

8. Vous avez certainement remarqué que les identifiants des documents sont affichés même si un filtre est appliqué aux résultats. Pour ne pas les afficher, vous pouvez ajouter sa valeur est la mettre a zero, comme ceci (cette requête nous renvois tous les films d'action réalisés en France sans les identifiants) :

```
> db.films.find(
...   { genre: "Action", country: "FR" },
...   { grades: 0, _id: 0 }
... )
{ "title" : "L'Homme de Rio", "year" : 1964, "genre" : "Action", "summary" : "Le deuxième classe Adrien Dufourquet est témoin de l'enlè
vement de sa fiancée Agnès, fille d'un célèbre ethnologue. Il part à sa recherche, qui le mène au Brésil, et met au jour un trafic de s
tatuettes indiennes.", "country" : "FR", "director" : { "_id" : "artist:34613", "last_name" : "de Broca", "first_name" : "Philippe", "b
irth_date" : 1933 }, "actors" : [ { "last_name" : "Belmondo", "first_name" : "Jean-Paul", "birth_date" : 1933 }, { "last_name" : "Celi"
```

9. Afficher les titres et les grades des films d'action réalisés en France sans leurs identifiants.

```
> db.films.find(
...   { genre: "Action", country: "FR" },
...   { title: 1, grades: 1, _id: 0 }
... )
{ "title" : "L'Homme de Rio", "grades" : [ { "note" : 4, "grade" : "D" }, { "note" : 30, "grade" : "E" }, { "note" : 34, "grade" : "E"
}, { "note" : 28, "grade" : "F" } ] }
{ "title" : "Nikita", "grades" : [ { "note" : 88, "grade" : "F" }, { "note" : 36, "grade" : "C" }, { "note" : 74, "grade" : "D" }, { "n
ote" : 62, "grade" : "D" } ] }
{ "title" : "Les tontons flingueurs", "grades" : [ { "note" : 35, "grade" : "C" }, { "note" : 48, "grade" : "F" }, { "note" : 64, "grad
e" : "C" }, { "note" : 42, "grade" : "F" } ] }
>
```

10. Les filtres affichés jusquà présent sont par valeur exacte. Afficher les titres et les notes des films d'action réalisés en France et ayant obtenus une note supérieure à 10. Remarquez que même si des films ont obtenu certaines notes inférieures à 10 sont affichés.

```
> db.films.find(
...   {
...     genre: "Action",
...     country: "FR",
...     "grades.note": { $gt: 10 }   // au moins une note > 10
...   },
...   { title: 1, grades: 1, _id: 0 }
... )
{ "title" : "L'Homme de Rio", "grades" : [ { "note" : 4, "grade" : "D" }, { "note" : 30, "grade" : "E" }, { "note" : 34, "grade" : "E"
}, { "note" : 28, "grade" : "F" } ] }
{ "title" : "Nikita", "grades" : [ { "note" : 88, "grade" : "F" }, { "note" : 36, "grade" : "C" }, { "note" : 74, "grade" : "D" }, { "n
ote" : 62, "grade" : "D" } ] }
{ "title" : "Les tontons flingueurs", "grades" : [ { "note" : 35, "grade" : "C" }, { "note" : 48, "grade" : "F" }, { "note" : 64, "grad
e" : "C" }, { "note" : 42, "grade" : "F" } ] }
>
```

"grades.note": { $gt: 10 } est appliquée au document entier, et non à chaque note séparément.
En MongoDB, lorsqu'un champ est un tableau (grades est un tableau d'objets {note, grade}), l'écriture "grades.note": { $gt: 10 } signifie : il existe au moins un élément du tableau grades dont la valeur note est strictement supérieure à 10.
Le film est donc sélectionné dès qu'une seule de ses notes dépasse 10, même si les autres notes du tableau sont inférieures. Comme la projection affiche tout le champ grades, on voit encore apparaître dans le résultat des notes < 10 : la requête filtre les films, pas les valeurs du tableau.

11. Si l'on souhaite afficher que des films ayant des notes supérieures à 10, on doit préciser que les notes obtenues sont toutes supérieures à 40. La requête suivante permet d'afficher les films d'action réalisés en France ayant obtenus que des notes supérieures à 10.

```
> db.films.find(
...   {
...     genre: "Action",
...     country: "FR",
...     grades: {
...       $not: { $elemMatch: { note: { $lte: 10 } } }
...     }
...   },
...   { title: 1, grades: 1, _id: 0 }
... )
{ "title" : "Nikita", "grades" : [ { "note" : 88, "grade" : "F" }, { "note" : 36, "grade" : "C" }, { "note" : 74, "grade" : "D" }, { "n
ote" : 62, "grade" : "D" } ] }
{ "title" : "Les tontons flingueurs", "grades" : [ { "note" : 35, "grade" : "C" }, { "note" : 48, "grade" : "F" }, { "note" : 64, "grad
e" : "C" }, { "note" : 42, "grade" : "F" } ] }
```

pour n'obtenir que les films dont toutes les notes sont > 10, on utilise au contraire :
grades: { $not: { $elemMatch: { note: { $lte: 10 } } } } : il n'existe aucun élément du tableau `grades` avec une note ≤ 10.

12. Afficher les différents genres de la base lesfilms.

```
> db.films.distinct("genre")
[
        "Action",
        "Adventure",
        "Aventure",
        "Comedy",
        "Comédie",
        "Crime",
        "Drama",
        "Drame",
        "Fantastique",
        "Fantasy",
        "Guerre",
        "Histoire",
        "Horreur",
        "Musique",
        "Mystery",
        "Mystère",
        "Romance",
        "Science Fiction",
        "Science-Fiction",
        "Thriller",
        "War",
        "Western"
]
>
```

13. Afficher les différents grades attribués.

```
> db.films.distinct("grades.grade")
[ "A", "B", "C", "D", "E", "F" ]
```

14. Afficher Tous les filmes dans lesquels joue au moins un des artistes suivants [''artist:4'',''artist:18'',''artist:11''].

```
> db.films.find({
...   "actors._id": { $in: ["artist:4", "artist:18", "artist:11"] }
... })
>
```

15. Afficher tous les films qui n'ont pas de résumés.

```
> db.films.find({
...   summary: { $exists: false }
... })
>
```

16. Afficher tous les films tournés avec Leonardo DiCaprio en 1997. 17. Afficher les films tournés avec Leonardo DiCaprio ou 1997.

```
> db.films.find({
...   year: 1997,
...   actors: {
...     $elemMatch: {
...       first_name: "Leonardo",
...       last_name: "DiCaprio"
...     }
...   }
... })
{ "_id" : "movie:597", "title" : "Titanic", "year" : 1997, "genre" : "Drame", "summary" : "Southampton, 10 avril 1912. Le paquebot le p
lus grand et le plus moderne du monde, réputé pour son insubmersibilité, le « Titanic », appareille pour son premier voyage. 4 jours pl
us tard, il heurte un iceberg. À son bord, un artiste pauvre et une grande bourgeoise tombent amoureux.", "country" : "US", "director"
: { "_id" : "artist:2710", "last_name" : "Cameron", "first_name" : "James", "birth_date" : 1954 }, "actors" : [ { "last_name" : "Winsle
t", "first_name" : "Kate", "birth_date" : 1975 }, { "last_name" : "Zane", "first_name" : "Billy", "birth_date" : 1966 }, { "last_name"
: "Fisher", "first_name" : "Frances", "birth_date" : 1952 }, { "last_name" : "DiCaprio", "first_name" : "Leonardo", "birth_date" : 1974
 }, { "last_name" : "Bates", "first_name" : "Kathy", "birth_date" : 1948 }, { "last_name" : "Stuart", "first_name" : "Gloria", "birth_d
ate" : 1910 } ], "grades" : [ { "note" : 40, "grade" : "C" }, { "note" : 91, "grade" : "B" }, { "note" : 45, "grade" : "D" }, { "note"
: 7, "grade" : "A" } ] }
>
```

17. Afficher les films tournés avec Leonardo DiCaprio ou 1997.

```
> db.films.find({
...   $or: [
...     {
...       actors: {
...         $elemMatch: {
...           first_name: "Leonardo",
...           last_name: "DiCaprio"
...         }
...       }
...     },
...     { year: 1997 }
...   ]
... })
{ "_id" : "movie:184", "title" : "Jackie Brown", "year" : 1997, "genre" : "Crime", "summary" : "Hôtesse de l'air, Jackie Brown arrondit
 ses fins de mois en convoyant de l'argent liquide pour le compte de Ordell Robbie, trafiquant d'armes. Quand la police et le FBI lui s
ignifient qu'ils comptent sur elle pour faire tomber Robbie, Jackie Brown échafaude un plan audacieux.", "country" : "US", "director" :
 { "_id" : "artist:138", "last_name" : "Tarantino", "first_name" : "Quentin", "birth_date" : 1963 }, "actors" : [ { "last_name" : "Tuck
```