Prise en main de MongoDB

Présentation de MongoDB

MongoDB est une base de données **NoSQL orientée documents**, ce qui signifie qu'elle stocke les données sous forme de **documents JSON** plutôt que dans des tables relationnelles. Elle est très adaptée aux applications nécessitant une **grande scalabilité** et une **flexibilité dans le schéma** des données. Chaque document peut contenir des sous-documents et des tableaux, permettant de représenter des données complexes de manière naturelle. MongoDB est largement utilisé pour des applications web, mobiles et des projets nécessitant un traitement rapide de grandes quantités de données non structurées.

Les commandes principales pour lancer MongoDB via Docker et importer les données :

Télécharger les données d'exemple

```
curl https://atlas-education.s3.amazonaws.com/sampledata.archive -o sampledata.archive
```

Lancer le conteneur MongoDB

```
docker run -d --name mongodb -p 27017:27017 mongo:7.0
```

Vérifier que le conteneur tourne

```
docker ps
```

Copier le fichier JSON dans le conteneur

```
docker cp "C:\Users\chayma\Desktop\ing3\NoSql\mango\films.json" mongodb:/films.json
```

Importer les données dans MongoDB

```
docker exec -it mongodb mongoimport --db sample_mflix --collection movies --file
/films.json --jsonArray
```

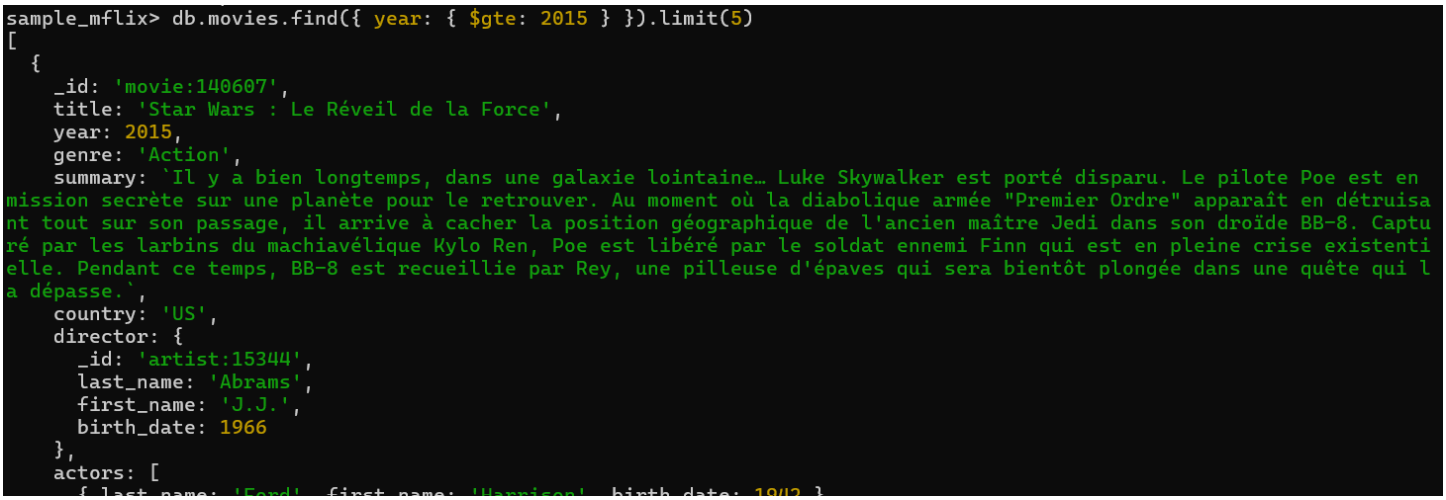
Se connecter au shell MongoDB

```
docker exec -it mongodb mongosh
```

Partie 1 – Filtrer et projeter les données

1. Afficher les 5 films sortis depuis 2015.

```
db.movies.find({ year: { $gte: 2015 } }).limit(5)
```



2. Trouver tous les films dont le genre est `"Comedy"`.

```
db.movies.find({ genres: "Comedy" })
```

```
sample_mflix> db.movies.find({ genre: "Comedy" })
[
  {
    _id: 'movie:75',
    title: 'Mars Attacks!',
    year: 1996,
    genre: 'Comedy',
    summary: "'We come in peace' is not what those green men from Mars mean when they invade our planet, armed with irre
sistible weapons and a cruel sense of humor.  This star studded cast must play victim to the alien's fun and games in th
is comedy homage to science fiction films of the '50s and '60s.",
    country: 'US',
    director: {
      _id: 'artist:510',
      last_name: 'Burton',
      first_name: 'Tim',
      birth_date: 1958
    },
    actors: [
      { last_name: 'Nicholson', first_name: 'Jack', birth_date: 1937 },
      { last_name: 'Close', first_name: 'Glenn', birth_date: 1947 },
      { last_name: 'Bening', first_name: 'Annette', birth_date: 1958 },
      { last_name: 'Brosnan', first_name: 'Pierce', birth_date: 1953 },
      { last_name: 'DeVito', first_name: 'Danny', birth_date: 1944 },
      { last_name: 'Short', first_name: 'Martin', birth_date: 1950 },
      {
        last_name: 'Jessica Parker',
        first_name: 'Sarah',
        birth_date: 1965
```

3. Afficher les sortis entre entre 2000 et 2005.

```
db.movies.find({year: {$gte: 2000, $lte: 2005}}, {title: 1, year:
1}).pretty()
```

```
sample_mflix> db.movies.find({year: {$gte: 2000, $lte: 2005}}, {title: 1, year: 1}).pretty()
[
  { _id: 'movie:24', title: 'Kill Bill : Volume 1', year: 2003 },
  { _id: 'movie:74', title: 'La Guerre des Mondes', year: 2005 },
  { _id: 'movie:116', title: 'Match point', year: 2005 },
  {
    _id: 'movie:120',
    title: 'The Lord of the Rings: The Fellowship of the Ring',
    year: 2001
  },
  {
    _id: 'movie:121',
    title: 'The Lord of the Rings: The Two Towers',
    year: 2002
  },
  {
    _id: 'movie:142',
    title: 'Le Secret de Brokeback Mountain',
    year: 2005
  },
  {
    _id: 'movie:122',
    title: 'The Lord of the Rings: The Return of the King',
    year: 2003
  },
  { _id: 'movie:146', title: 'Tigre et Dragon', year: 2000 },
  { _id: 'movie:153', title: 'Lost in Translation', year: 2003 },
```

4. Afficher les films de genres “Drama” ET “Romance”.

```
db.movies.find({genres: {$all: ["Drama", "Romance"]}}, {title: 1, genres:
1})
```

```
sample_mflix> db.movies.find({genre: {$all: ["Drama", "Romance"]}}, {title: 1, genres: 1})

sample_mflix>
```

5. Afficher les films sans champ `rated`.

```
db.movies.find({rated: {$exists: false}}, {title: 1})
```

```
sample_mflix> db.movies.find({rated: {$exists: false}}, {title: 1})
[
  { _id: 'movie:33', title: 'Impitoyable' },
  { _id: 'movie:24', title: 'Kill Bill : Volume 1' },
  { _id: 'movie:74', title: 'La Guerre des Mondes' },
  { _id: 'movie:116', title: 'Match point' },
  {
    _id: 'movie:120',
    title: 'The Lord of the Rings: The Fellowship of the Ring'
  },
  { _id: 'movie:11', title: 'La Guerre des étoiles' },
  { _id: 'movie:28', title: 'Apocalypse Now' },
  { _id: 'movie:121', title: 'The Lord of the Rings: The Two Towers' },
  { _id: 'movie:142', title: 'Le Secret de Brokeback Mountain' },
  { _id: 'movie:145', title: 'Breaking the Waves' },
  {
    _id: 'movie:122',
    title: 'The Lord of the Rings: The Return of the King'
  },
  { _id: 'movie:147', title: 'Les Quatre Cents Coups' },
  { _id: 'movie:146', title: 'Tigre et Dragon' },
  { _id: 'movie:153', title: 'Lost in Translation' },
  { _id: 'movie:184', title: 'Jackie Brown' },
  { _id: 'movie:192', title: 'Le Nom de la rose' },
  { _id: 'movie:155', title: 'The Dark Knight : Le Chevalier noir' },
  { _id: 'movie:180', title: 'Minority Report' },
```

-> Ces commandes illustrent les opérations de filtrage, projection et interrogation de tableaux dans MongoDB. Elles permettent de sélectionner des sous-ensembles précis de données et d'explorer rapidement le contenu de la collection **`movies`**.

Partie 2 – Agrégation

6. Afficher le nombre de films par année.

```
db.movies.aggregate([
    {$group: {_id: "$year", total: {$sum: 1}}},
    {$sort: {_id: 1}}
])
```

```
sample_mflix> db.movies.aggregate([
...   {$group: {_id: "$year", total: {$sum: 1}}},
...   {$sort: {_id: 1}}
... ])
[
  { _id: 1921, total: 1 }, { _id: 1927, total: 1 },
  { _id: 1931, total: 1 }, { _id: 1936, total: 2 },
  { _id: 1937, total: 1 }, { _id: 1940, total: 3 },
  { _id: 1941, total: 1 }, { _id: 1942, total: 1 },
  { _id: 1943, total: 1 }, { _id: 1944, total: 1 },
  { _id: 1946, total: 2 }, { _id: 1947, total: 1 },
  { _id: 1948, total: 1 }, { _id: 1949, total: 1 },
  { _id: 1950, total: 2 }, { _id: 1951, total: 1 },
  { _id: 1952, total: 2 }, { _id: 1954, total: 2 },
  { _id: 1955, total: 1 }, { _id: 1956, total: 2 }
]
```

7. Afficher la moyenne des notes IMDb par genre.

```
db.movies.aggregate([
    {$unwind: "$genres"},
    {$group: {_id: "$genres", moyenne: {$avg: "$imdb.rating"}}},
    {$sort: {moyenne: -1}}
])
```

```
sample_mflix> db.movies.aggregate([
...   {$unwind: "$genres"},
...   {$group: {_id: "$genres", moyenne: {$avg: "$imdb.rating"}}},
...   {$sort: {moyenne: -1}}
... ])

sample_mflix>
```

8. Afficher le nombre de films par pays.

```
db.movies.aggregate([
    {$unwind: "$countries"},
    {$group: {_id: "$countries", total: {$sum: 1}}},
    {$sort: {total: -1}}
])
```

```
sample_mflix> db.movies.aggregate([
...   // PAS BESOIN de $unwind si 'country' est une chaîne simple
...   { $group: { _id: "$country", total: { $sum: 1 } } },
...   { $sort: { total: -1 } }
... ])
[
  { _id: 'US', total: 144 },
  { _id: 'FR', total: 58 },
  { _id: 'GB', total: 22 },
  { _id: 'IT', total: 7 },
  { _id: 'CA', total: 7 },
  { _id: 'DE', total: 6 },
  { _id: 'SE', total: 6 },
  { _id: 'JP', total: 4 },
  { _id: 'BE', total: 4 },
  { _id: 'NZ', total: 3 },
  { _id: 'RU', total: 3 },
  { _id: 'AU', total: 3 },
  { _id: 'CN', total: 2 },
  { _id: 'DK', total: 2 },
  { _id: 'NL', total: 1 },
  { _id: 'HK', total: 1 },
  { _id: 'CZ', total: 1 },
  { _id: 'NO', total: 1 },
  { _id: 'ES', total: 1 },
  { _id: 'IE', total: 1 }
]
Type "it" for more
sample_mflix>
```

9. Afficher les top 5 réalisateurs.

```
db.movies.aggregate([
    {$unwind: "$directors"},
    {$group: {_id: "$directors", total: {$sum: 1}}},
    {$sort: {total: -1}},
    {$limit: 5}
])
```

```
sample_mflix> db.movies.aggregate([
...   { $unwind: "$director" },
...   { $group: { _id: "$director", total: { $sum: 1 } } },
...   { $sort: { total: -1 } },
...   { $limit: 5 }
... ])
[
  {
    _id: {
      _id: 'artist:488',
      last_name: 'Spielberg',
      first_name: 'Steven',
      birth_date: 1946
    },
    total: 13
  },
  {
    _id: {
      _id: 'artist:2636',
      last_name: 'Hitchcock',
      first_name: 'Alfred',
      birth_date: 1899
    },
    total: 10
  },
  {
    _id: {
      _id: 'artist:1243',
      last_name: 'Allen',
      first_name: 'Woody',
      birth_date: 1935
    },
    total: 8
  },
  {
    _id: {
      _id: 'artist:3556',
      last_name: 'Polański',
      first_name: 'Roman',
      birth_date: 1933
    },
    total: 7
  },
  {
    _id: {
      _id: 'artist:138',
      last_name: 'Tarantino',
```

10. Afficher les films triés par note IMDb.

```
db.movies.aggregate([
    {$sort: {"imdb.rating": -1}},
    {$project: {title: 1, "imdb.rating": 1}}
])
```

```
sample_mflix> db.movies.aggregate([
...   { $sort: { "imdb.rating": -1 } },
...   { $project: { title: 1, "imdb.rating": 1, _id: 0 } }
... ])
[
  { title: 'Impitoyable' },
  { title: 'Kill Bill : Volume 1' },
  { title: 'La Guerre des Mondes' },
  { title: 'Match point' },
  { title: 'The Lord of the Rings: The Fellowship of the Ring' },
  { title: 'La Guerre des étoiles' },
  { title: 'Apocalypse Now' },
  { title: 'The Lord of the Rings: The Two Towers' },
  { title: 'Le Secret de Brokeback Mountain' },
  { title: 'Breaking the Waves' },
  { title: 'The Lord of the Rings: The Return of the King' },
  { title: 'Les Quatre Cents Coups' },
  { title: 'Tigre et Dragon' },
  { title: 'Lost in Translation' },
  { title: 'Jackie Brown' },
  { title: 'Le Nom de la rose' },
  { title: 'The Dark Knight : Le Chevalier noir' },
  { title: 'Minority Report' },
  { title: 'La Mort aux trousses' },
  { title: 'Terminator' }
]
Type "it" for more
sample_mflix>
```

-> Ces commandes illustrent les deux méthodes fondamentales d'interrogation dans MongoDB : la recherche simple (find), utilisée ici pour filtrer les documents en fonction de l'absence d'un champ ($exists: false), et le pipeline d'agrégation (aggregate), qui permet d'effectuer des analyses statistiques avancées. L'agrégation est essentielle pour transformer les données, notamment en utilisant $unwind pour analyser les tableaux (comme les pays ou les réalisateurs), puis en utilisant $group avec des accumulateurs comme $sum et $avg pour calculer des totaux et des moyennes par catégorie (année, genre, pays), avant d'appliquer un tri ($sort) et de restreindre les résultats (par exemple, le $limit pour les top 5).

Partie 3 – Mises à jour

11. Ajouter un champ `etat`.

```
db.movies.updateOne({title: "Jaws"}, {$set: {etat: "culte"}})
```

```
Type "it" for more
sample_mflix> db.movies.updateOne({title: "Jaws"}, {$set: {etat: "culte"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

Champ de Retour	Valeur	Interprétation
matchedCount	0	**Aucun document** avec le titre "Jaws" n'a été trouvé.
modifiedCount	0	Aucun document n'a été modifié, car rien n'a été trouvé.

12. Incrémenter les votes IMDb de 100, par exemple.

```
db.movies.updateOne({title: "Inception"}, {$inc: {"imdb.votes": 100}})
```

```
sample_mflix> db.movies.updateOne({title: "Inception"}, {$inc: {"imdb.votes": 100}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Champ de Retour	Valeur	Interprétation
`matchedCount`	1	**Un document** avec le titre "Inception" a été trouvé.
`modifiedCount`	1	**Un document** a été modifié (le champ `imdb.votes` a été incrémenté de 100).

13. Supprimer le le champ `poste`

```
db.movies.updateMany({}, {$unset: {poster: ""}})
```

```
sample_mflix> db.movies.updateMany({}, {$unset: {poster: ""}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 278,
  modifiedCount: 0,
  upsertedCount: 0
}
```

Champ de Retour	Valeur	Interprétation
`matchedCount`	278	**278 documents** ont été trouvés (le filtre `{}` correspond à tous les documents).

modifiedCount	278	**278 documents** ont été modifiés (le champ poster a été supprimé).

14. Modifier le réalisateur.

```
db.movies.updateOne({title: "Titanic"}, {$set: {directors: ["James Cameron"]}})
```

```
sample_mflix> db.movies.updateOne(
...   {title: "Titanic"},
...   {
...     $set: {
...       director: "James Cameroun"
...     }
...   }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sample_mflix>
```

Champ de Retour	Valeur	Interprétation
acknowledged	true	La base de données a bien reçu et traité la commande.
matchedCount	1	**Un document** (le film "Titanic") correspondant au critère {title: "Titanic"} a été trouvé.
modifiedCount	1	**Ce même document** a été modifié. Le champ director a été mis à jour avec la nouvelle valeur "James Cameroun".
upsertedCount	0	Aucun nouveau document n'a été inséré (cette opération était une mise à jour, pas une insertion).

-> Cette section était entièrement consacrée aux manipulations de mise à jour des documents dans la collection movies en utilisant les commandes updateOne et updateMany. Ces opérations ont permis d'apprendre trois opérateurs de modification essentiels : l'opérateur $set, utilisé pour ajouter un nouveau champ (comme etat) ou modifier la valeur d'un champ existant (comme director); l'opérateur $inc, qui permet d'effectuer des ajustements numériques en incrémentant le nombre de votes IMDb ; et enfin, l'opérateur $unset, crucial pour supprimer définitivement un champ (comme poster) de tous les documents correspondants au filtre appliqué.

Partie 4 – Requêtes complexes

15. Afficher les films les mieux notés par décennie.

```
db.movies.aggregate([
    {$match: {"imdb.rating": {$exists: true}}},
      {$project:
  { title: 1,
       decade: {$subtract: ["$year", {$mod: ["$year", 10]}]},
  "imdb.rating": 1
    }},
    {$group: {_id: "$decade", maxRating: {$max: "$imdb.rating"}}},
    {$sort: {_id: 1}}
])
```

16. Afficher les films dont le titre commence par “Star”.

```
db.movies.find({title: /^Star/}, {title: 1})
```

```
sample_mflix> db.movies.find(
...   { title: /^Star/i },
...   { title: 1, _id: 0 }
... )
[
  { title: 'Starship Troopers' },
  { title: 'Star Wars, épisode III - La Revanche des Sith' },
  { title: 'Star Wars, épisode I - La Menace fantôme' },
  { title: "Star Wars, épisode II - L'Attaque des clones" },
  { title: 'Star Wars : Le Réveil de la Force' },
  { title: 'Star Wars, épisode IX' },
  { title: 'Star Wars : Les Derniers Jedi' }
]
```

17. afficher les films avec plus de 2 genres.

```
db.movies.find({$where: "this.genres.length > 2"}, {title: 1, genres: 1})
```

```
sample_mflix> db.movies.find(
...   { $where: "this.genre && this.genre.length > 2" },
...   { title: 1, genre: 1, _id: 0 }
... )
[
  { title: 'Impitoyable', genre: 'Western' },
  { title: 'Kill Bill : Volume 1', genre: 'Action' },
  { title: 'La Guerre des Mondes', genre: 'Aventure' },
  { title: 'Match point', genre: 'Drame' },
  {
    title: 'The Lord of the Rings: The Fellowship of the Ring',
    genre: 'Adventure'
  },
  { title: 'La Guerre des étoiles', genre: 'Aventure' },
  { title: 'Apocalypse Now', genre: 'Drame' },
  {
    title: 'The Lord of the Rings: The Two Towers',
    genre: 'Adventure'
  },
  { title: 'Le Secret de Brokeback Mountain', genre: 'Drame' },
  { title: 'Breaking the Waves', genre: 'Drame' },
  {
    title: 'The Lord of the Rings: The Return of the King',
    genre: 'Adventure'
  },
  { title: 'Les Quatre Cents Coups', genre: 'Drame' },
  { title: 'Tigre et Dragon', genre: 'Aventure' },
  { title: 'Lost in Translation', genre: 'Drame' },
  { title: 'Jackie Brown', genre: 'Crime' },
  { title: 'Le Nom de la rose', genre: 'Drame' },
  { title: 'The Dark Knight : Le Chevalier noir', genre: 'Drame' },
  { title: 'Minority Report', genre: 'Action' },
  { title: 'La Mort aux trousses', genre: 'Mystère' },
  { title: 'Terminator', genre: 'Action' }
]
Type "it" for more
sample_mflix>
```

18. Afficher les films de Christopher Nolan.

```
db.movies.find({directors: "Christopher Nolan"}, {title: 1, year: 1, "imdb.rating": 1})
```

```
sample_mflix> db.movies.find(
...     { directors: "Christopher Nolan" },
...     { title: 1, year: 1, "imdb.rating": 1, _id: 0 }
... )

sample_mflix> db.movies.findOne({ directors: "Christopher Nolan" })
null
sample_mflix>
```

Partie 5 – Indexation

19. Créer un index sur `year`

```
db.movies.createIndex({year: 1})
```

```
sample_mflix> db.movies.createIndex({year: 1})
year_1
sample_mflix>
```

20. Vérifier les index existants.

```
db.movies.getIndexes()
```

```
sample_mflix> db.movies.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { year: 1 }, name: 'year_1' }
]
sample_mflix>
```

21. Comparer deux requêtes avec et sans index (utiliser `explain(`).

```
db.movies.find({year: 1995}).explain("executionStats")
```

```
sample_mflix> db.movies.find({year: 1995}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'sample_mflix.movies',
    indexFilterSet: false,
    parsedQuery: { year: { '$eq': 1995 } },
    queryHash: '108EB0D0',
    planCacheKey: '5F26085F',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { year: 1 },
        indexName: 'year_1',
        isMultiKey: false,
        multiKeyPaths: { year: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { year: [ '[1995, 1995]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 7,
    executionTimeMillis: 1,
    totalKeysExamined: 7,
    totalDocsExamined: 7,
    executionStages: {
      stage: 'FETCH',
      nReturned: 7,
```

Observez les et pour voir l'effet de champs l'index. `"totalDocsExamined"`

22. Supprimer l'index sur .

```
db.movies.dropIndex({year: 1})
```

```
sample_mflix> db.movies.dropIndex({year: 1})
{ nIndexesWas: 2, ok: 1 }
sample_mflix> db.movies.find({year: 1995}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'sample_mflix.movies',
    indexFilterSet: false,
    parsedQuery: { year: { '$eq': 1995 } },
    queryHash: '108EB0D0',
    planCacheKey: '108EB0D0',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'COLLSCAN',
      filter: { year: { '$eq': 1995 } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 7,
    executionTimeMillis: 0,
    totalKeysExamined: 0,
    totalDocsExamined: 278,
    executionStages: {
      stage: 'COLLSCAN',
      filter: { year: { '$eq': 1995 } },
      nReturned: 7,
      executionTimeMillisEstimate: 0,
      works: 279,
      advanced: 7,
      needTime: 271,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
```

-> L'analyse de performance avec explain("executionStats") démontre que la présence de l'index year_1 permet à la requête de filtrage sur year: 1995 d'utiliser le plan IXSCAN (Index Scan), qui est très efficace : elle n'a eu besoin d'examiner que 7 documents et 7 clés d'index pour trouver les 7 résultats, le tout en 1 milliseconde. Après la suppression de cet index via db.movies.dropIndex({year: 1}), la même requête a été forcée d'utiliser le plan COLLSCAN (Collection Scan) ; elle a alors dû examiner 278 documents (soit tous les documents de la collection) pour trouver les 7 résultats, ce qui prouve que l'index était essentiel pour éviter un balayage complet de la collection et garantir une exécution rapide et ciblée.

23. créer un index composé sur `year` et `imdb.rating`.

```
db.movies.createIndex({year: 1, "imdb.rating": -1})
```

```
sample_mflix> db.movies.createIndex({year: 1, "imdb.rating": -1})
year_1_imdb.rating_-1
sample_mflix>
```