Gloire BAYOUNDOULA
INFOA3

Matière : **Bases de données NoSQL**
Partie : **MongoDB**
Enseignant : **Monsieur Samir Youcef**

Rendu TP1 - Partie 2

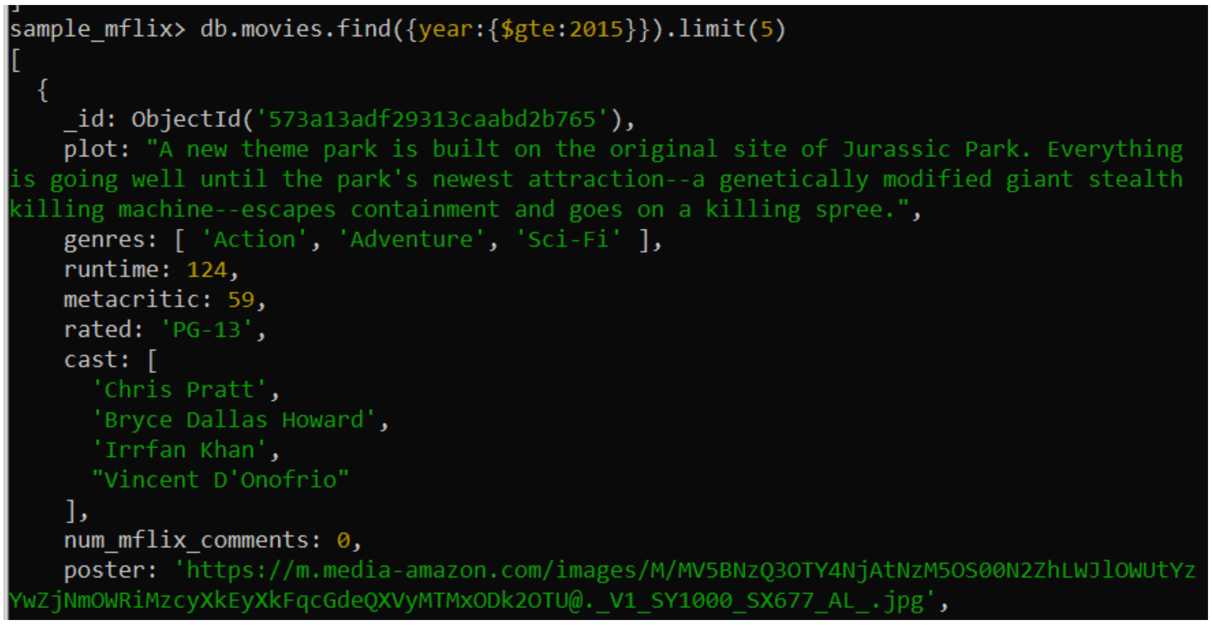
Pour cette partie, nous allons voir les bases de données NoSQL MongoDB.
MongoDB est une base de données NoSQL orientée **documents JSON/BSON**, distribuée et scalable, conçue pour stocker des données semi-structurées avec flexibilité.
Chaque document est un objet JSON riche, pouvant contenir des tableaux et des sous-documents, et les collections n'imposent pas de schéma fixe. MongoDB offre des **requêtes puissantes**, des **index variés** (single, compound, text, géospatial), de l'**agrégation** pour transformations complexes, et supporte la **réplication** et le **sharding** pour haute disponibilité et scalabilité horizontale.
Persistante sur disque et optimisée pour de gros volumes, elle permet des opérations atomiques au niveau du document et s'intègre facilement aux applications modernes, mais a une latence plus élevée que des moteurs en mémoire comme Redis.

• Réponses aux questions

Partie 1 – Filtrer et projeter les données

1. Afficher les 5 films sortis depuis 2015.

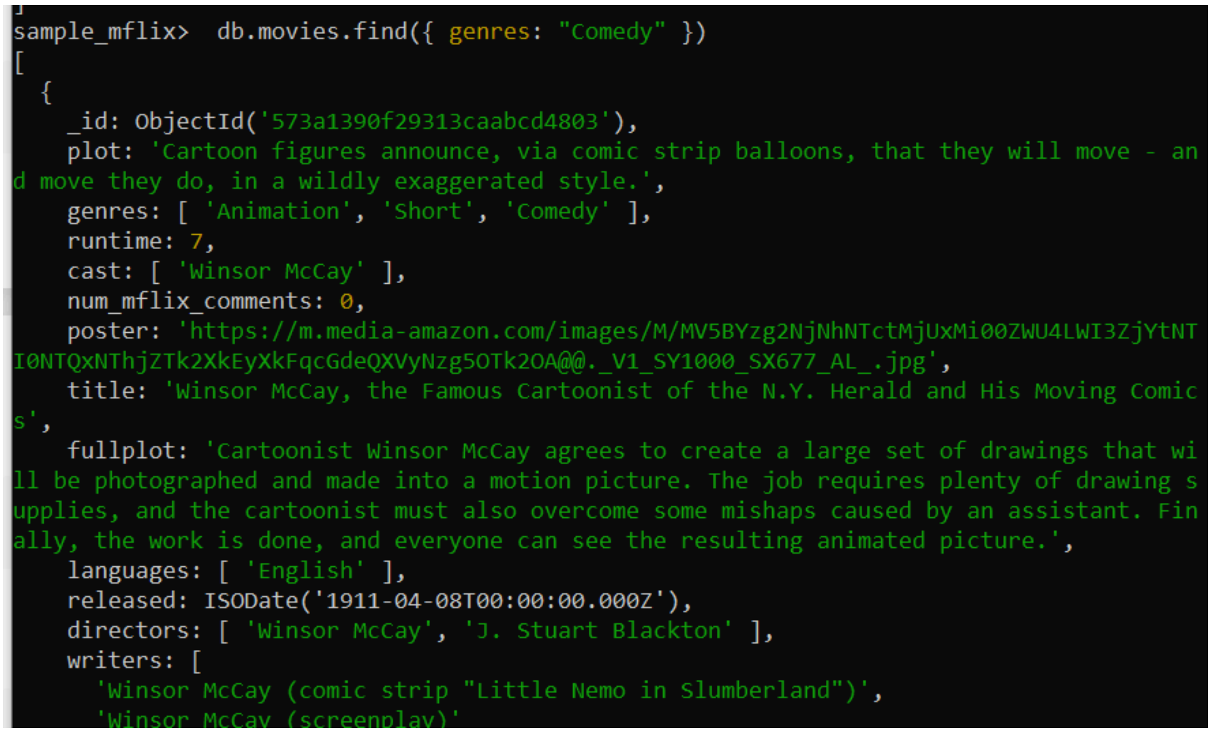
db.movies.find({ year: { $gte: 2015 } }).limit(5)



```
sample_mflix> db.movies.find({year:{$gte:2015}}).limit(5)
[
  {
    _id: ObjectId('573a13adf29313caabd2b765'),
    plot: "A new theme park is built on the original site of Jurassic Park. Everything
is going well until the park's newest attraction--a genetically modified giant stealth
killing machine--escapes containment and goes on a killing spree.",
    genres: [ 'Action', 'Adventure', 'Sci-Fi' ],
    runtime: 124,
    metacritic: 59,
    rated: 'PG-13',
    cast: [
      'Chris Pratt',
      'Bryce Dallas Howard',
      'Irrfan Khan',
      "Vincent D'Onofrio"
    ],
    num_mflix_comments: 0,
    poster: 'https://m.media-amazon.com/images/M/MV5BNzQ3OTY4NjAtNzM5OS00N2ZhLWJlOWUtYz
YwZjNmOWRiMzcyXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_SY1000_SX677_AL_.jpg',
```

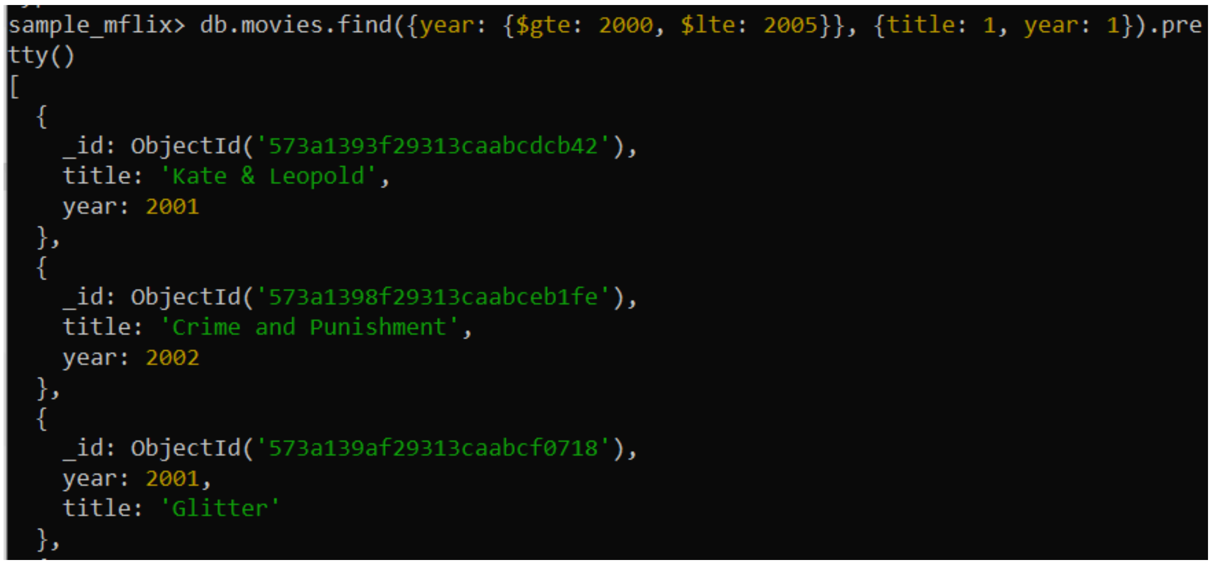
2. Trouver tous les films dont le genre est "Comedy"

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



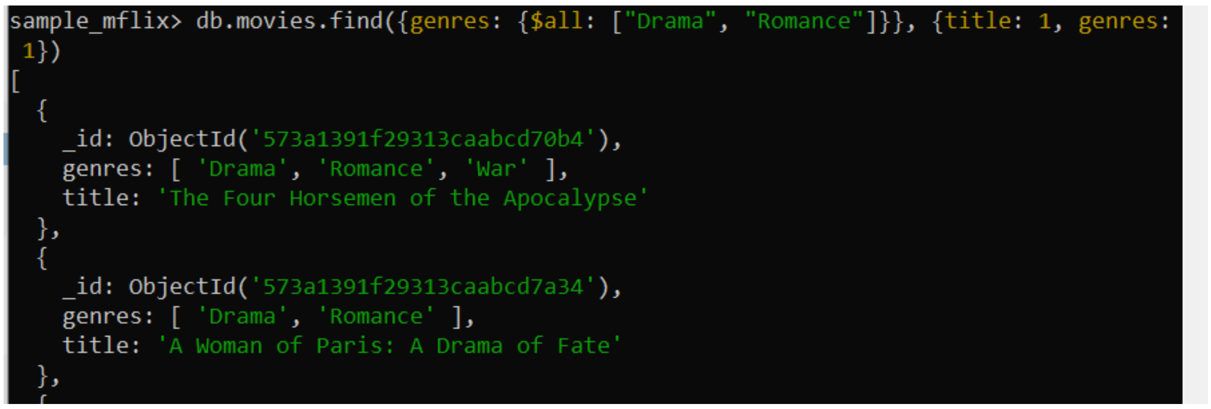
```
sample_mflix>  db.movies.find({ genres: "Comedy" })
[
  {
    _id: ObjectId('573a1390f29313caabcd4803'),
    plot: 'Cartoon figures announce, via comic strip balloons, that they will move - an
d move they do, in a wildly exaggerated style.',
    genres: [ 'Animation', 'Short', 'Comedy' ],
    runtime: 7,
    cast: [ 'Winsor McCay' ],
    num_mflix_comments: 0,
    poster: 'https://m.media-amazon.com/images/M/MV5BYzg2NjNhNTctMjUxMi00ZWU4LWI3ZjYtNT
I0NTQxNThjZTk2XkEyXkFqcGdeQXVyNzg5OTk2OA@@._V1_SY1000_SX677_AL_.jpg',
    title: 'Winsor McCay, the Famous Cartoonist of the N.Y. Herald and His Moving Comic
s',
    fullplot: 'Cartoonist Winsor McCay agrees to create a large set of drawings that wi
ll be photographed and made into a motion picture. The job requires plenty of drawing s
upplies, and the cartoonist must also overcome some mishaps caused by an assistant. Fin
ally, the work is done, and everyone can see the resulting animated picture.',
    languages: [ 'English' ],
    released: ISODate('1911-04-08T00:00:00.000Z'),
    directors: [ 'Winsor McCay', 'J. Stuart Blackton' ],
    writers: [
      'Winsor McCay (comic strip "Little Nemo in Slumberland")',
      'Winsor McCay (screenplay)'
```

3. Afficher les sorties entre 2000 et 2005.
db.movies.find({year: {$gte: 2000, $lte: 2005}}, {title: 1, year: 1}).pretty()



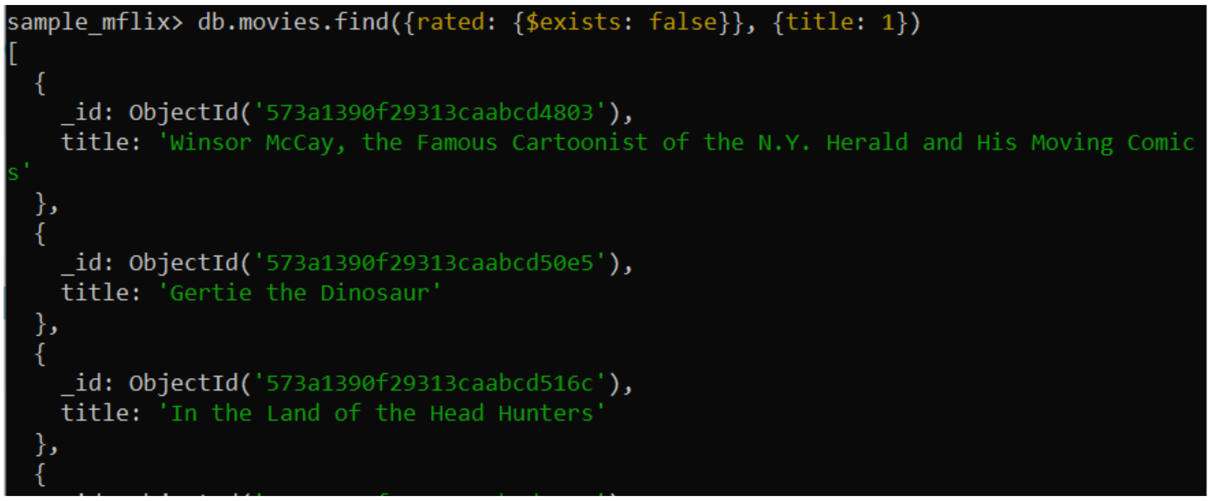
```
sample_mflix> db.movies.find({year: {$gte: 2000, $lte: 2005}}, {title: 1, year: 1}).pre
tty()
[
  {
    _id: ObjectId('573a1393f29313caabcdcb42'),
    title: 'Kate & Leopold',
    year: 2001
  },
  {
    _id: ObjectId('573a1398f29313caabceb1fe'),
    title: 'Crime and Punishment',
    year: 2002
  },
  {
    _id: ObjectId('573a139af29313caabcf0718'),
    year: 2001,
    title: 'Glitter'
  },
```

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({genres: {$all: ["Drama", "Romance"]}}, {title: 1, genres:
 1})
[
  {
    _id: ObjectId('573a1391f29313caabcd70b4'),
    genres: [ 'Drama', 'Romance', 'War' ],
    title: 'The Four Horsemen of the Apocalypse'
  },
  {
    _id: ObjectId('573a1391f29313caabcd7a34'),
    genres: [ 'Drama', 'Romance' ],
    title: 'A Woman of Paris: A Drama of Fate'
  },
```

5. Afficher les films sans champ rated .
db.movies.find({rated: {$exists: false}}, {title: 1})

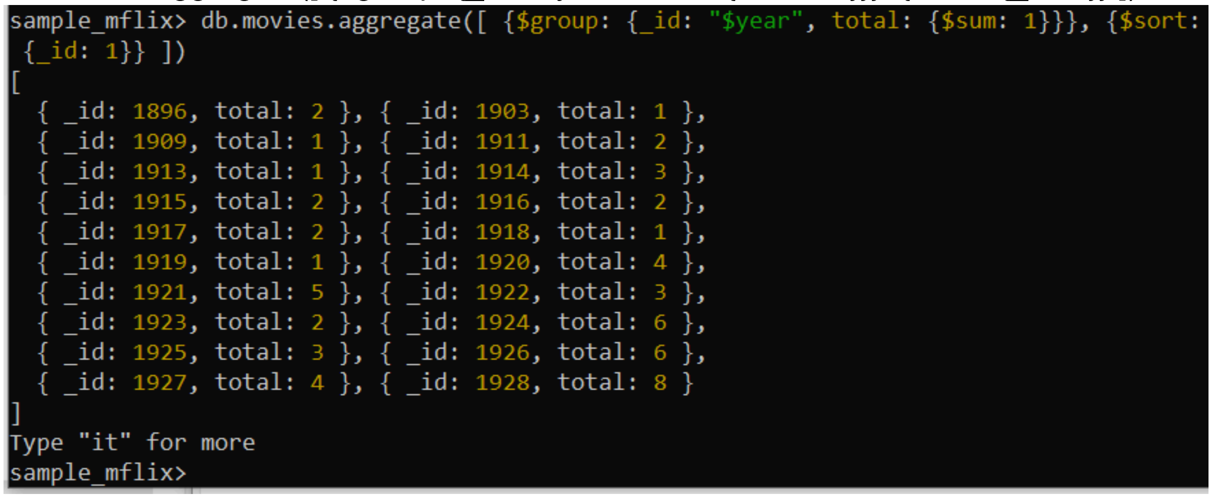


```
sample_mflix> db.movies.find({rated: {$exists: false}}, {title: 1})
[
  {
    _id: ObjectId('573a1390f29313caabcd4803'),
    title: 'Winsor McCay, the Famous Cartoonist of the N.Y. Herald and His Moving Comic
s'
  },
  {
    _id: ObjectId('573a1390f29313caabcd50e5'),
    title: 'Gertie the Dinosaur'
  },
  {
    _id: ObjectId('573a1390f29313caabcd516c'),
    title: 'In the Land of the Head Hunters'
  },
  {
```

Partie 2 – Agrégation

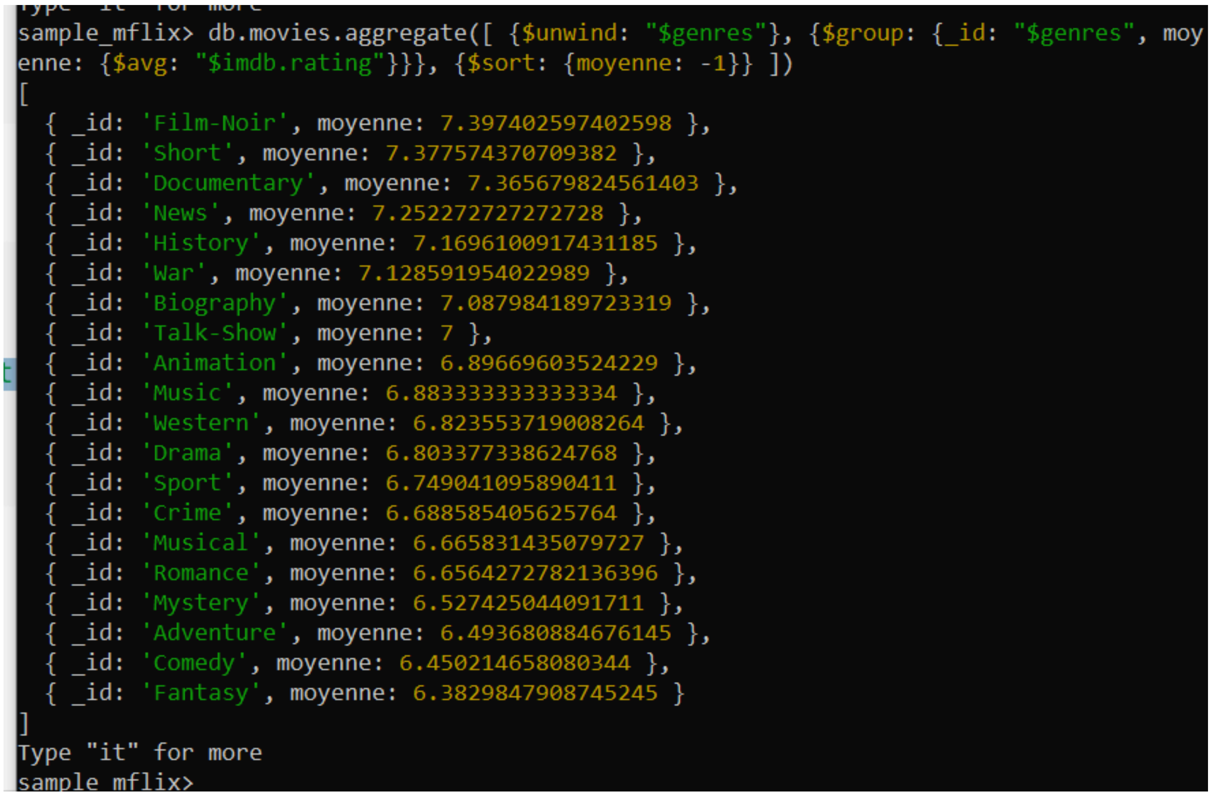
6. Afficher le nombre de films par année.

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: 1896, total: 2 }, { _id: 1903, total: 1 },
  { _id: 1909, total: 1 }, { _id: 1911, total: 2 },
  { _id: 1913, total: 1 }, { _id: 1914, total: 3 },
  { _id: 1915, total: 2 }, { _id: 1916, total: 2 },
  { _id: 1917, total: 2 }, { _id: 1918, total: 1 },
  { _id: 1919, total: 1 }, { _id: 1920, total: 4 },
  { _id: 1921, total: 5 }, { _id: 1922, total: 3 },
  { _id: 1923, total: 2 }, { _id: 1924, total: 6 },
  { _id: 1925, total: 3 }, { _id: 1926, total: 6 },
  { _id: 1927, total: 4 }, { _id: 1928, total: 8 }
]
Type "it" for more
sample_mflix>
```

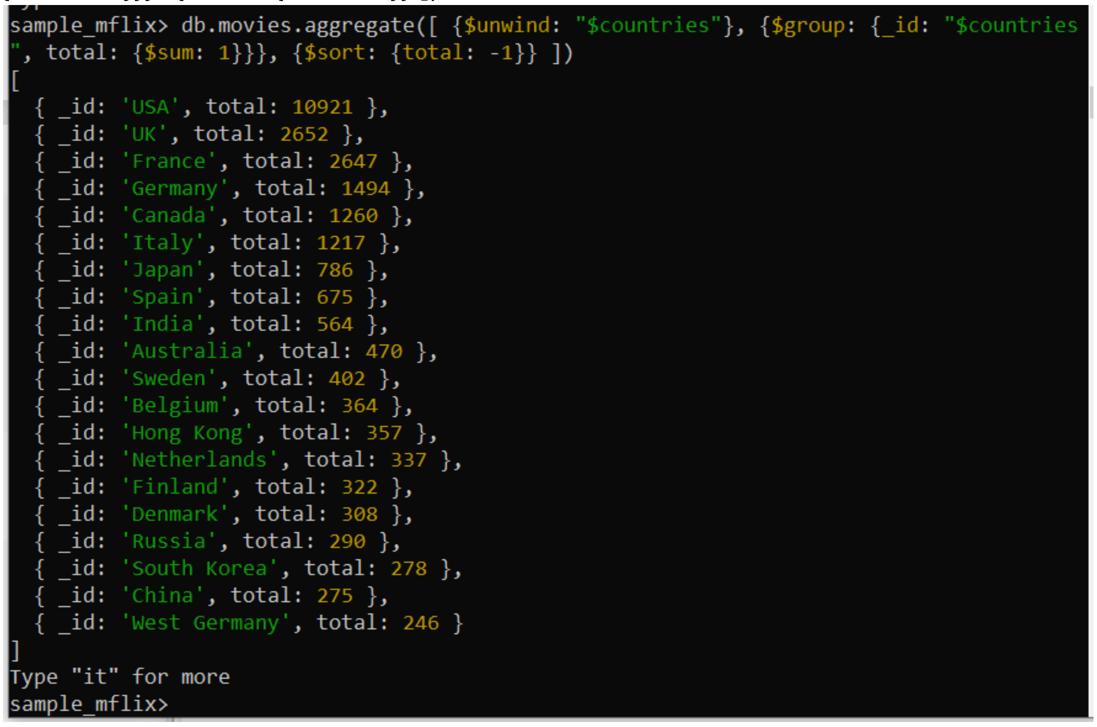
7. Afficher la moyenne des notes IMDb par genre.
db.movies.aggregate([{$unwind: "$genres"}, {$group: {_id: "$genres", moyenne: {$avg: "$imdb.rating"}}}, {$sort: {moyenne: -1}}])



```
sample_mflix> db.movies.aggregate([ {$unwind: "$genres"}, {$group: {_id: "$genres", moy
enne: {$avg: "$imdb.rating"}}}, {$sort: {moyenne: -1}} ])
[
  { _id: 'Film-Noir', moyenne: 7.397402597402598 },
  { _id: 'Short', moyenne: 7.377574370709382 },
  { _id: 'Documentary', moyenne: 7.365679824561403 },
  { _id: 'News', moyenne: 7.252272727272728 },
  { _id: 'History', moyenne: 7.1696100917431185 },
  { _id: 'War', moyenne: 7.128591954022989 },
  { _id: 'Biography', moyenne: 7.087984189723319 },
  { _id: 'Talk-Show', moyenne: 7 },
  { _id: 'Animation', moyenne: 6.89669603524229 },
  { _id: 'Music', moyenne: 6.883333333333334 },
  { _id: 'Western', moyenne: 6.823553719008264 },
  { _id: 'Drama', moyenne: 6.803377338624768 },
  { _id: 'Sport', moyenne: 6.749041095890411 },
  { _id: 'Crime', moyenne: 6.688585405625764 },
  { _id: 'Musical', moyenne: 6.665831435079727 },
  { _id: 'Romance', moyenne: 6.6564272782136396 },
  { _id: 'Mystery', moyenne: 6.527425044091711 },
  { _id: 'Adventure', moyenne: 6.493680884676145 },
  { _id: 'Comedy', moyenne: 6.450214658080344 },
  { _id: 'Fantasy', moyenne: 6.3829847908745245 }
]
Type "it" for more
sample_mflix>
```

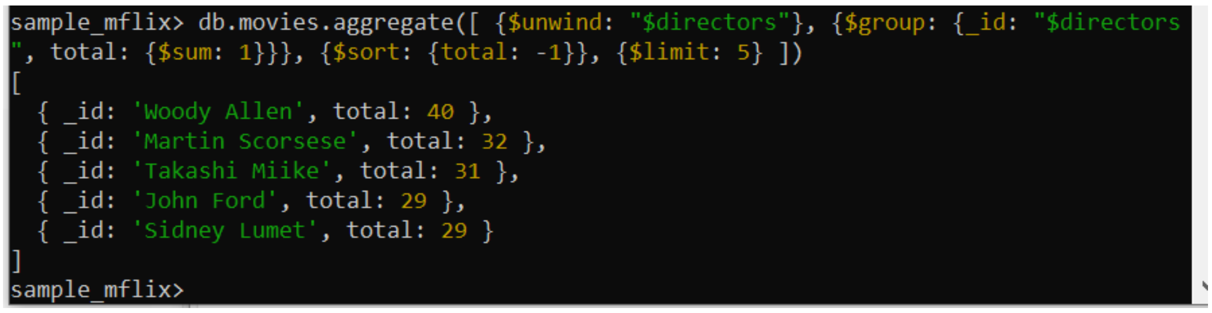
8. Afficher le nombre de films par pays

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



```
sample_mflix> db.movies.aggregate([ {$unwind: "$countries"}, {$group: {_id: "$countries
", total: {$sum: 1}}}, {$sort: {total: -1}} ])
[
  { _id: 'USA', total: 10921 },
  { _id: 'UK', total: 2652 },
  { _id: 'France', total: 2647 },
  { _id: 'Germany', total: 1494 },
  { _id: 'Canada', total: 1260 },
  { _id: 'Italy', total: 1217 },
  { _id: 'Japan', total: 786 },
  { _id: 'Spain', total: 675 },
  { _id: 'India', total: 564 },
  { _id: 'Australia', total: 470 },
  { _id: 'Sweden', total: 402 },
  { _id: 'Belgium', total: 364 },
  { _id: 'Hong Kong', total: 357 },
  { _id: 'Netherlands', total: 337 },
  { _id: 'Finland', total: 322 },
  { _id: 'Denmark', total: 308 },
  { _id: 'Russia', total: 290 },
  { _id: 'South Korea', total: 278 },
  { _id: 'China', total: 275 },
  { _id: 'West Germany', total: 246 }
]
Type "it" for more
sample_mflix>
```

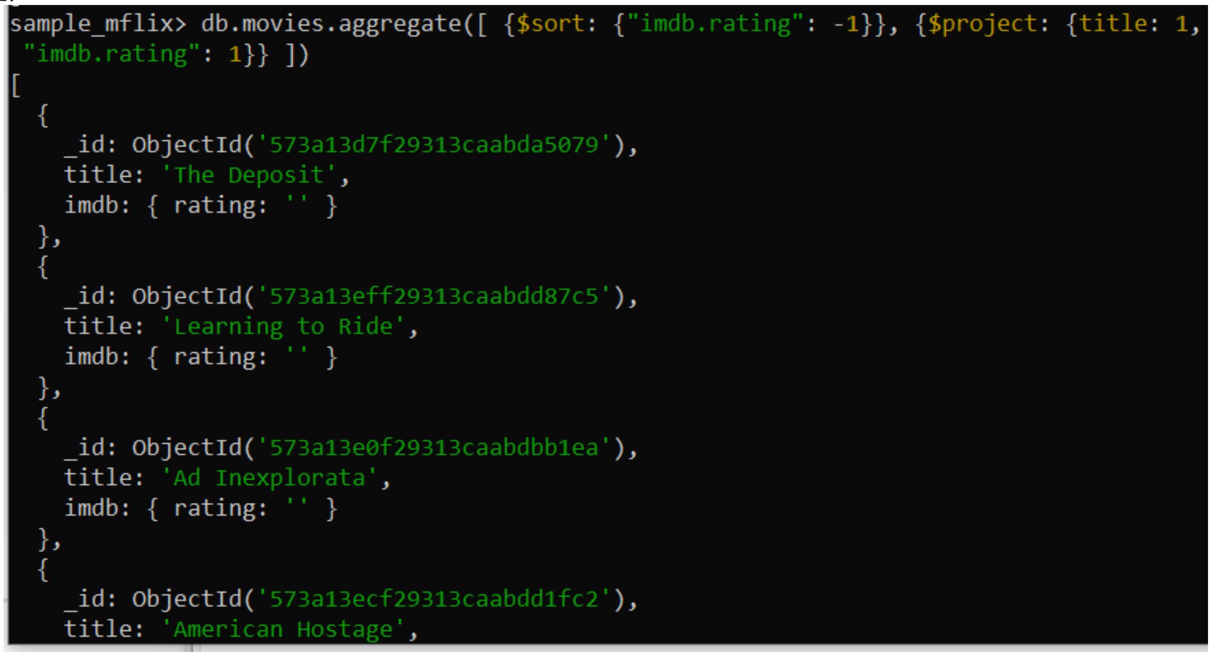
9. Afficher les top 5 réalisateurs.
db.movies.aggregate([{$unwind: "$directors"}, {$group: {_id: "$directors", total: {$sum: 1}}}, {$sort: {total: -1}}, {$limit: 5}])



```
sample_mflix> db.movies.aggregate([ {$unwind: "$directors"}, {$group: {_id: "$directors
", total: {$sum: 1}}}, {$sort: {total: -1}}, {$limit: 5} ])
[
  { _id: 'Woody Allen', total: 40 },
  { _id: 'Martin Scorsese', total: 32 },
  { _id: 'Takashi Miike', total: 31 },
  { _id: 'John Ford', total: 29 },
  { _id: 'Sidney Lumet', total: 29 }
]
sample_mflix>
```

10. Afficher les films triés par note IMDb.

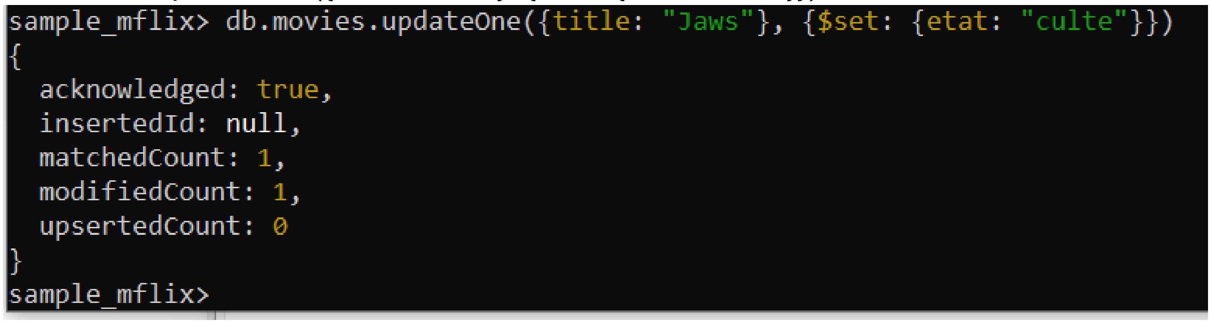
db.movies.aggregate([{$sort: {"imdb.rating": -1}}, {$project: {title: 1, "imdb.rating": 1}}])



```
sample_mflix> db.movies.aggregate([ {$sort: {"imdb.rating": -1}}, {$project: {title: 1,
 "imdb.rating": 1}} ])
[
  {
    _id: ObjectId('573a13d7f29313caabda5079'),
    title: 'The Deposit',
    imdb: { rating: '' }
  },
  {
    _id: ObjectId('573a13eff29313caabdd87c5'),
    title: 'Learning to Ride',
    imdb: { rating: '' }
  },
  {
    _id: ObjectId('573a13e0f29313caabdbb1ea'),
    title: 'Ad Inexplorata',
    imdb: { rating: '' }
  },
  {
    _id: ObjectId('573a13ecf29313caabdd1fc2'),
    title: 'American Hostage',
```

Partie 3 – Mises à jour

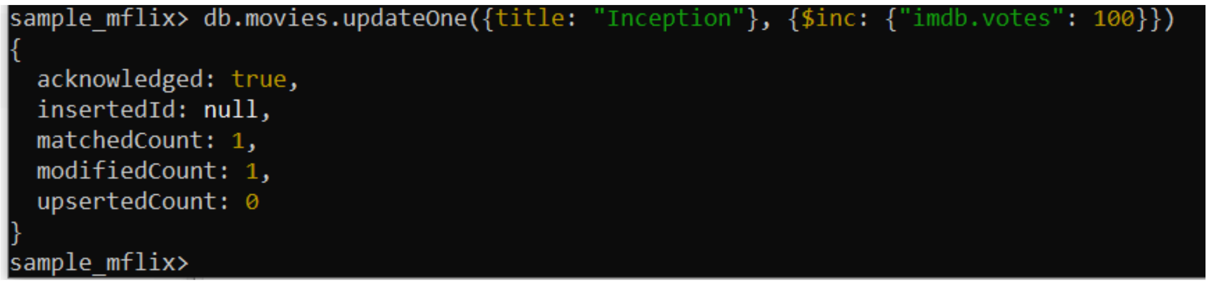
11. Ajouter un champ etat .
db.movies.updateOne({title: "Jaws"}, {$set: {etat: "culte"}})



```
sample_mflix> db.movies.updateOne({title: "Jaws"}, {$set: {etat: "culte"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sample_mflix>
```

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

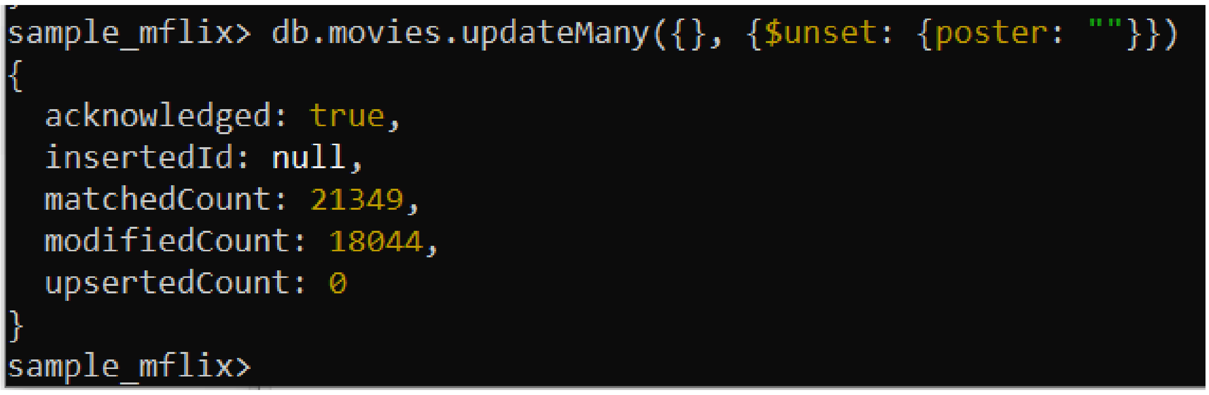
db.movies.updateOne({title: "Inception"}, {$inc: {"imdb.votes": 100}})



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

13. Supprimer le le champ poster

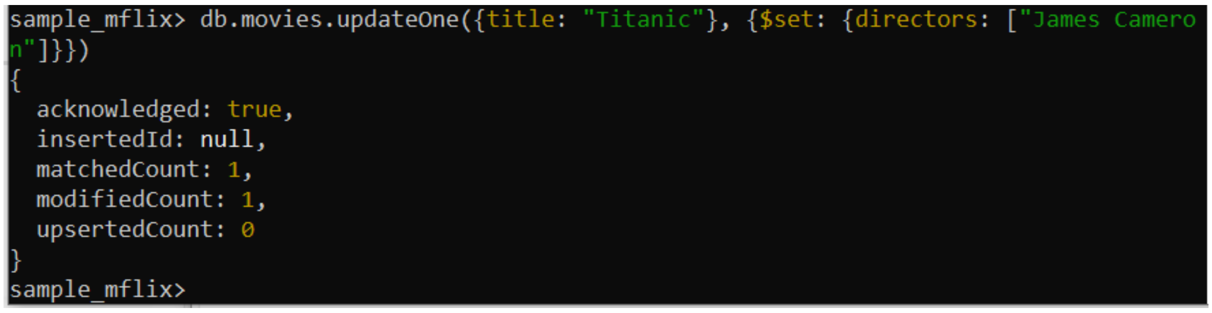
db.movies.updateMany({}, {$unset: {poster: ""}})



```
sample_mflix> db.movies.updateMany({}, {$unset: {poster: ""}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 21349,
  modifiedCount: 18044,
  upsertedCount: 0
}
sample_mflix>
```

14. Modifier le réalisateur.

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

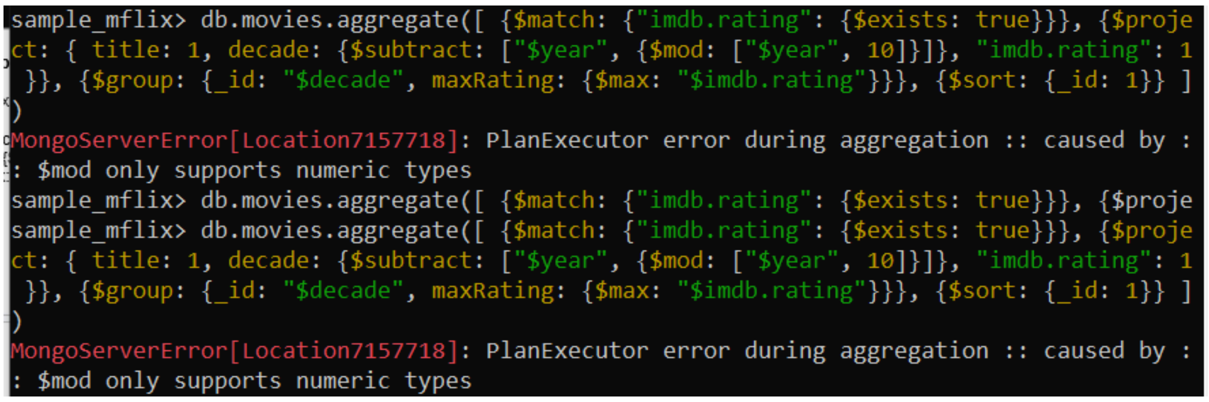


```
sample_mflix> db.movies.updateOne({title: "Titanic"}, {$set: {directors: ["James Camero
n"]}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sample_mflix>
```

Partie 4 – Requêtes complexes

15. Afficher les films les mieux notés par décennie.

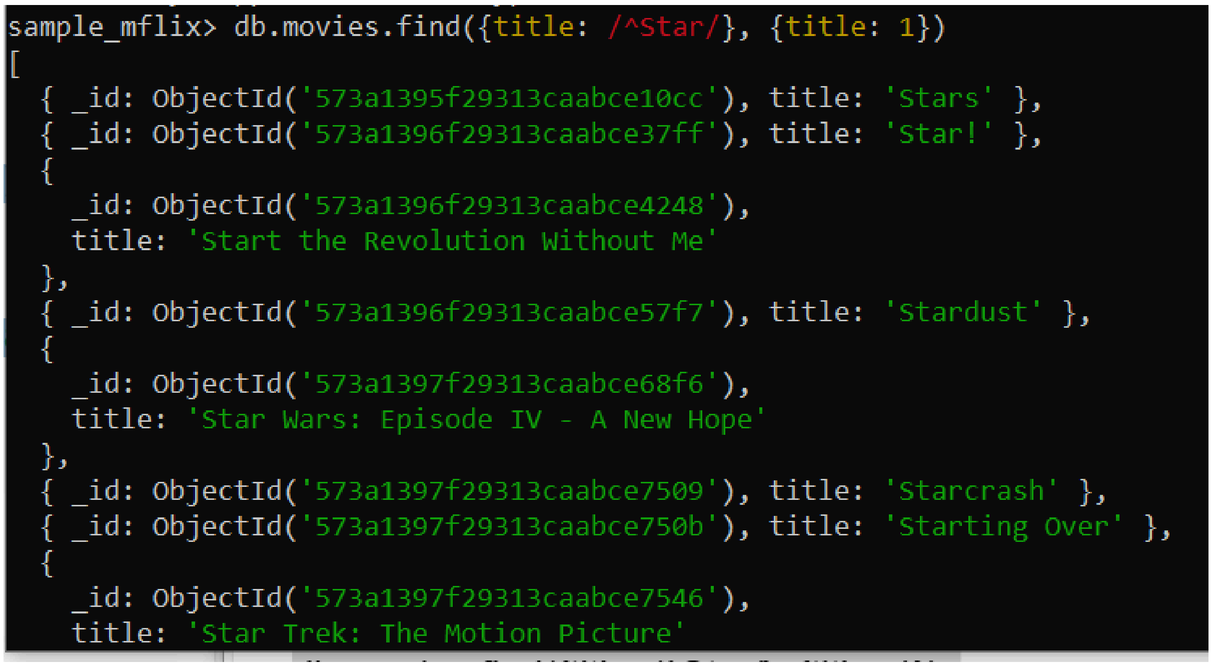
db.movies.aggregate([{$match: {"imdb.rating": {$exists: true}}}, {$project: { title: 1, decade: {$subtract: ["$year", {$mod: ["$year", 10]}]}, "imdb.rating": 1 }}, {$group: {_id: "$decade", maxRating: {$max: "$imdb.rating"}}}, {$sort: {_id: 1}}])



```
sample_mflix> db.movies.aggregate([ {$match: {"imdb.rating": {$exists: true}}}, {$proje
ct: { title: 1, decade: {$subtract: ["$year", {$mod: ["$year", 10]}]}, "imdb.rating": 1
 }}, {$group: {_id: "$decade", maxRating: {$max: "$imdb.rating"}}}, {$sort: {_id: 1}} ]
)
MongoServerError[Location7157718]: PlanExecutor error during aggregation :: caused by :
: $mod only supports numeric types
sample_mflix> db.movies.aggregate([ {$match: {"imdb.rating": {$exists: true}}}, {$proje
sample_mflix> db.movies.aggregate([ {$match: {"imdb.rating": {$exists: true}}}, {$proje
ct: { title: 1, decade: {$subtract: ["$year", {$mod: ["$year", 10]}]}, "imdb.rating": 1
 }}, {$group: {_id: "$decade", maxRating: {$max: "$imdb.rating"}}}, {$sort: {_id: 1}} ]
)
MongoServerError[Location7157718]: PlanExecutor error during aggregation :: caused by :
: $mod only supports numeric types
```

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

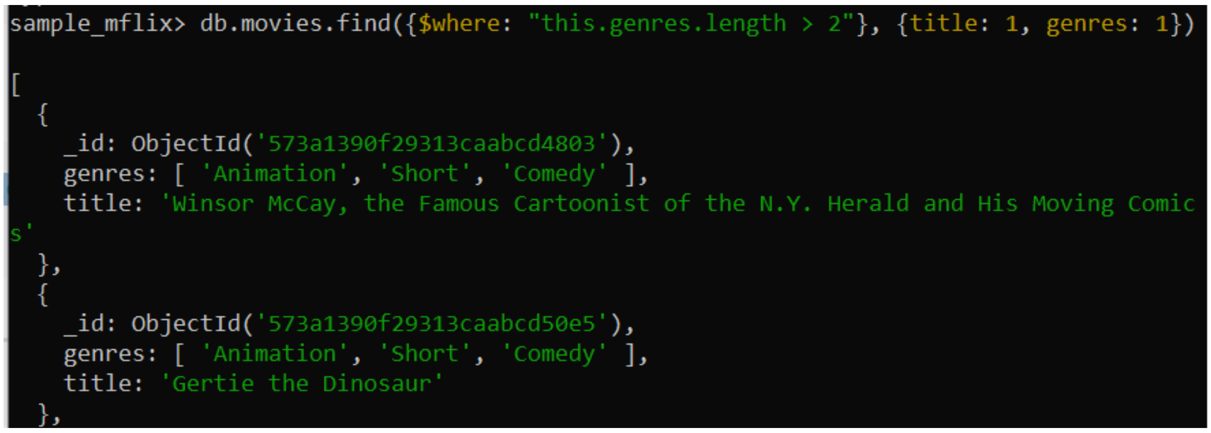
db.movies.find({title: /^Star/}, {title: 1})



```
sample_mflix> db.movies.find({title: /^Star/}, {title: 1})
[
  { _id: ObjectId('573a1395f29313caabce10cc'), title: 'Stars' },
  { _id: ObjectId('573a1396f29313caabce37ff'), title: 'Star!' },
  {
    _id: ObjectId('573a1396f29313caabce4248'),
    title: 'Start the Revolution Without Me'
  },
  { _id: ObjectId('573a1396f29313caabce57f7'), title: 'Stardust' },
  {
    _id: ObjectId('573a1397f29313caabce68f6'),
    title: 'Star Wars: Episode IV - A New Hope'
  },
  { _id: ObjectId('573a1397f29313caabce7509'), title: 'Starcrash' },
  { _id: ObjectId('573a1397f29313caabce750b'), title: 'Starting Over' },
  {
    _id: ObjectId('573a1397f29313caabce7546'),
    title: 'Star Trek: The Motion Picture'
```

17. afficher les films avec plus de 2 genres.

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

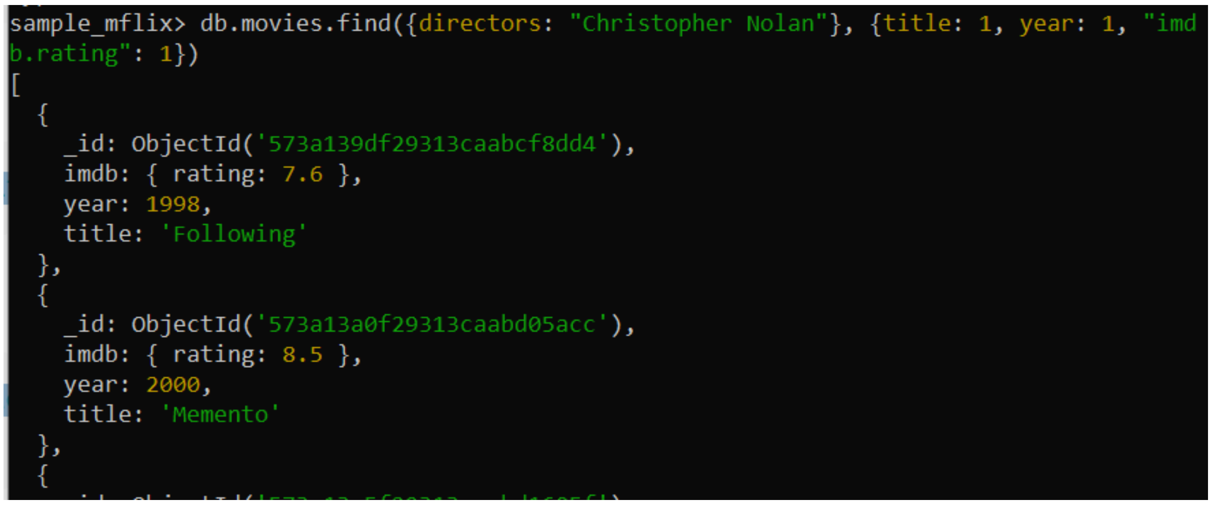


```
sample_mflix> db.movies.find({$where: "this.genres.length > 2"}, {title: 1, genres: 1})

[
  {
    _id: ObjectId('573a1390f29313caabcd4803'),
    genres: [ 'Animation', 'Short', 'Comedy' ],
    title: 'Winsor McCay, the Famous Cartoonist of the N.Y. Herald and His Moving Comic
s'
  },
  {
    _id: ObjectId('573a1390f29313caabcd50e5'),
    genres: [ 'Animation', 'Short', 'Comedy' ],
    title: 'Gertie the Dinosaur'
  },
```

18. Afficher les films de Christopher Nolan.

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

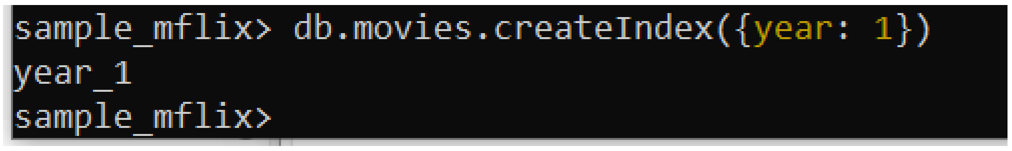


```
sample_mflix> db.movies.find({directors: "Christopher Nolan"}, {title: 1, year: 1, "imd
b.rating": 1})
[
  {
    _id: ObjectId('573a139df29313caabcf8dd4'),
    imdb: { rating: 7.6 },
    year: 1998,
    title: 'Following'
  },
  {
    _id: ObjectId('573a13a0f29313caabd05acc'),
    imdb: { rating: 8.5 },
    year: 2000,
    title: 'Memento'
  },
  {
```

Partie 5 – Indexation

19. Créer un index sur year

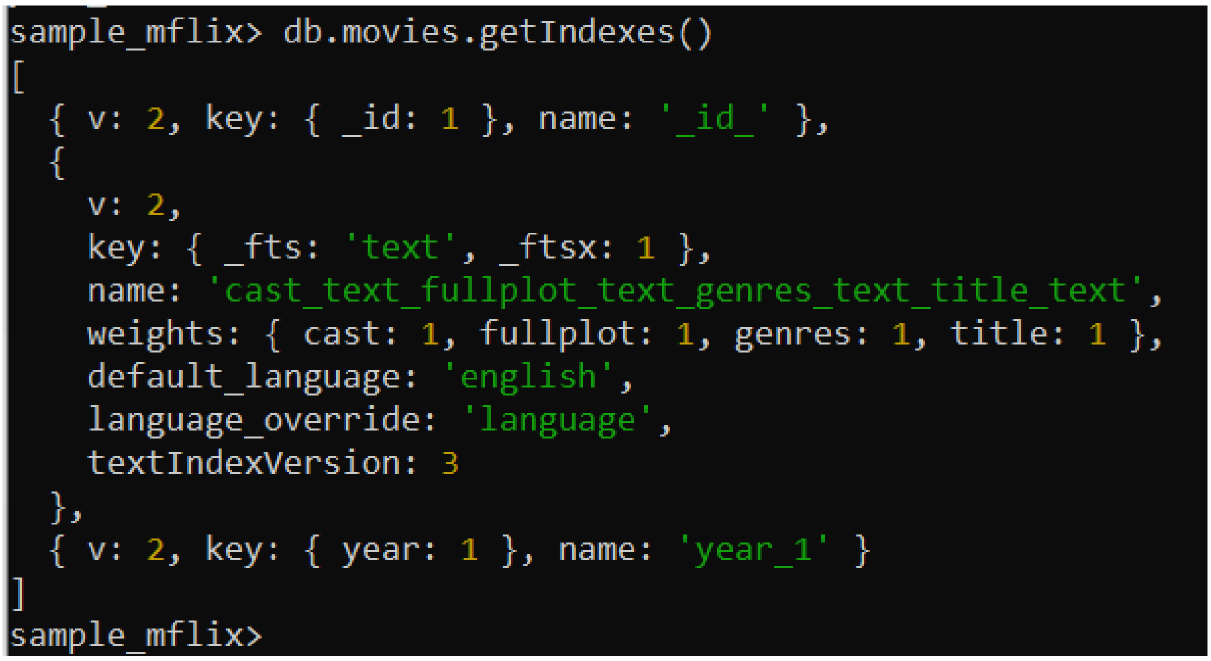
db.movies.createIndex({year: 1})



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

20. Vérifier les index existants.

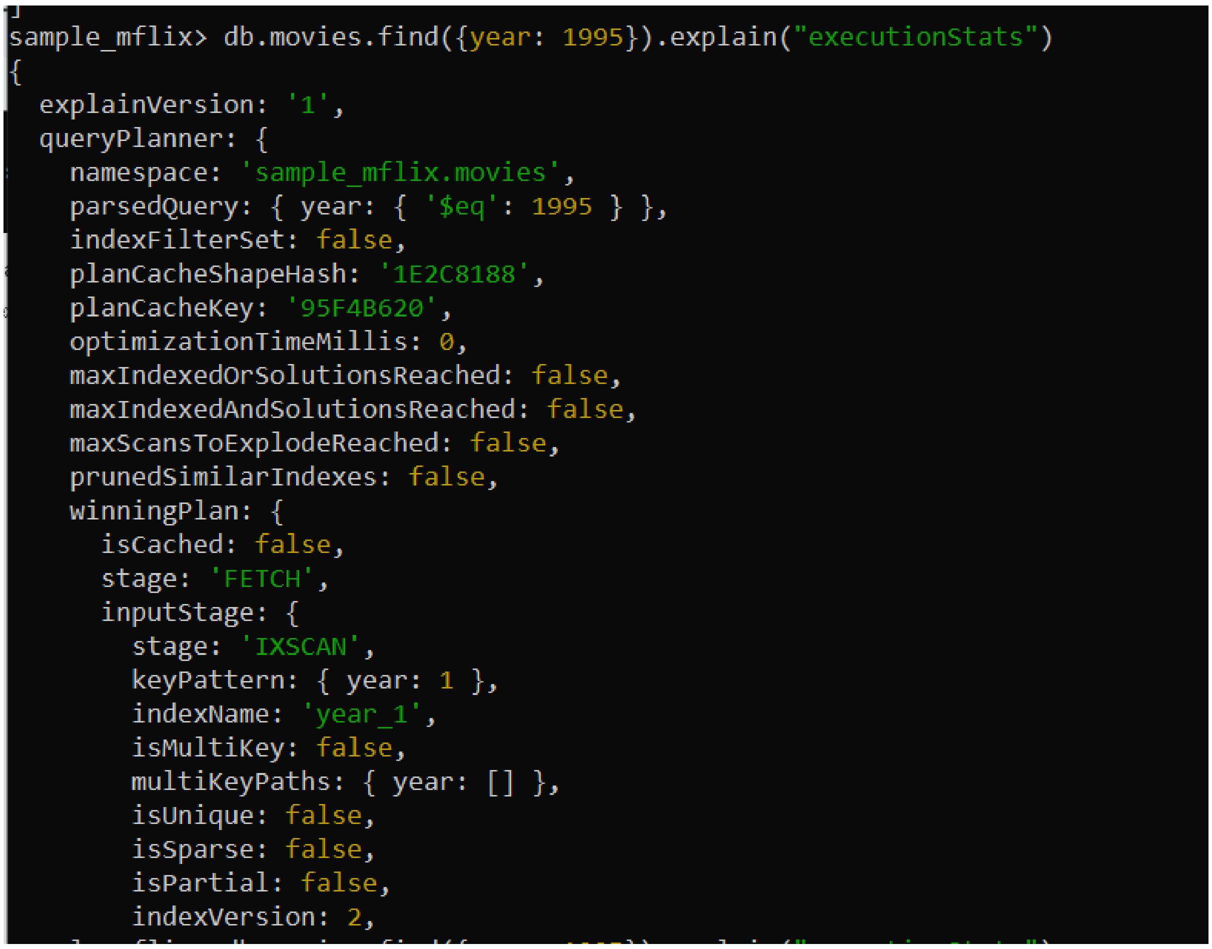
db.movies.getIndexes()



```
sample_mflix> db.movies.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  {
    v: 2,
    key: { _fts: 'text', _ftsx: 1 },
    name: 'cast_text_fullplot_text_genres_text_title_text',
    weights: { cast: 1, fullplot: 1, genres: 1, title: 1 },
    default_language: 'english',
    language_override: 'language',
    textIndexVersion: 3
  },
  { v: 2, key: { year: 1 }, name: 'year_1' }
]
sample_mflix>
```

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

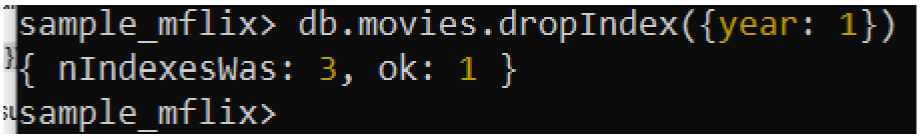
db.movies.find({year: 1995}).explain("executionStats")



```
sample_mflix> db.movies.find({year: 1995}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'sample_mflix.movies',
    parsedQuery: { year: { '$eq': 1995 } },
    indexFilterSet: false,
    planCacheShapeHash: '1E2C8188',
    planCacheKey: '95F4B620',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { year: 1 },
        indexName: 'year_1',
        isMultiKey: false,
        multiKeyPaths: { year: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
```

22. Supprimer l'index sur year .

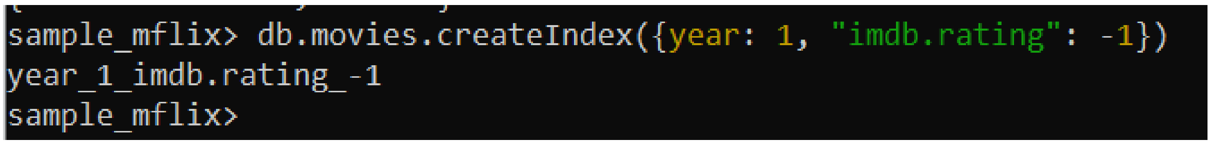
db.movies.dropIndex({year: 1})



```
sample_mflix> db.movies.dropIndex({year: 1})
{ nIndexesWas: 3, ok: 1 }
sample_mflix>
```

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>
```