Université Sorbonne Paris Nord
École d'ingénieur Sup Galilée

TP MongoDB

Compte rendu du TP

Partie 2 et 3

Réalisé par :

Célian WIRTZ (celian.wirtz@edu.univ-paris13.fr)

3ème année
Année universitaire : 2025/2026

TP1 – PART 2

Partie 1 – Filtrer et projeter les données

On commence par récupérer les données de l'archive après avoir télé-charger MongoDB.

```
PS C:\Users\wirtz\Downloads> mongorestore --archive=sampledata.archive --nsInclude="*" --uri="mongodb://localhost:27017"

2025-11-27T10:55:20.848+0100    preparing collections to restore from
2025-11-27T10:55:20.870+0100    reading metadata for sample_training.inspections from archive 'sampledata.archive'
2025-11-27T10:55:20.871+0100    reading metadata for sample_training.companies from archive 'sampledata.archive'
```

Question 1 :

Ensuite dans le Shell de MongoDB, on se met dans sample_mflix pour trouver nos données de film.

```
> use sample_mflix
< switched to db 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 th
  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,
```

(Résultat trop long pour tout afficher, j'affiche juste le début)

Bon, on fait juste une projection pour que ce soit lisible :

```
db.movies.find({ year: { $gte: 2015 } }, { title: 1, _id: 0 }).limit(5)

{
  title: 'Jurassic World'
}
{
  title: 'The Stanford Prison Experiment'
}
{
  title: 'Ex Machina'
}
{
  title: 'Ant-Man'
}
{
  title: 'The Danish Girl'
```

Question 2 :

De même à partir de maintenant on prendra à chaque fois seulement le titre sauf indication contraire et une limite de 5 éléments.

```
> db.movies.find({ genres: "Comedy" }, { title: 1, _id: 0 }).limit(5)

< {
    title: 'Winsor McCay, the Famous Cartoonist of the N.Y. Herald and His Moving Comics'
  }
  {
    title: 'Gertie the Dinosaur'
  }
  {
    title: 'The Poor Little Rich Girl'
  }
  {
    title: 'Wild and Woolly'
  }
  {
    title: 'From Hand to Mouth'
```

Question 3 :

```
> db.movies.find({ year: { $gte: 2000, $lte: 2005 } }, { title: 1, _id: 0 }).limit(5)

< {
    title: 'Kate & Leopold'
  }
  {
    title: 'Crime and Punishment'
  }
  {
    title: 'Glitter'
  }
  {
    title: 'In the Mood for Love'
  }
  {
    title: 'The Manson Family'
```

Question 4 :

```
> db.movies.find({ genres: { $all: ["Drama", "Romance"] } }, { title: 1, _id: 0 }).limit(5)
< {
    title: 'The Four Horsemen of the Apocalypse'
  }
  {
    title: 'A Woman of Paris: A Drama of Fate'
  }
  {
    title: 'He Who Gets Slapped'
  }
  {
    title: 'Wild Oranges'
  }
  {
    title: 'Wings'
```

Question 5 :

```
db.movies.find({ rated: { $exists: false } }, { title: 1, _id: 0 }).limit(5)
{
  title: 'Winsor McCay, the Famous Cartoonist of the N.Y. Herald and His Moving Comics'
}
{
  title: 'Gertie the Dinosaur'
}
{
  title: 'In the Land of the Head Hunters'
}
{
  title: 'The Perils of Pauline'
}
{
  title: 'Civilization'
```

Partie 2 – Agrégation

Question 6 :

```
> db.movies.aggregate([{ $group: { _id: "$year", total: { $sum: 1 } } }, { $sort: { _id: 1 } }, { $limit: 5 }
< {
    _id: 1896,
    total: 2
  }
  {
    _id: 1903,
    total: 1
  }
  {
    _id: 1909,
    total: 1
  }
  {
    _id: 1911,
    total: 2
  }
  {
    _id: 1913,
    total: 1
```

Question 7 :

```
> db.movies.aggregate([{ $unwind: "$genres" }, { $group: { _id: "$genres", moyenne: { $avg: "$imdb.rating" }
< {
    _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
```

Question 8 :

```
> db.movies.aggregate([{ $unwind: "$countries" }, { $group: { _id: "$countries", total: { $sum: 1 } } }, { $s
< {
    _id: 'USA',
    total: 10921
  }
  {
    _id: 'UK',
    total: 2652
  }
  {
    _id: 'France',
    total: 2647
  }
  {
    _id: 'Germany',
    total: 1494
  }
  {
    _id: 'Canada',
    total: 1260
```

Question 9 :

```
> db.movies.aggregate([{ $unwind: "$directors" }, { $group: { _id: "$directors", total: { $sum: 1 } } }, { $s
< {
    _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
  }
```

Question 10 :

Là j'ai modifié la requête parce qu'on avait des rating vide qui s'affichait, les films où ce n'était pas rempli s'affichait en premier. Donc j'évite de prendre en compte ceux-là.

```
> db.movies.aggregate([{ $match: { "imdb.rating": { $ne: "" } } }, { $sort: { "imdb.rating": -1 } }, { $proje
< {
    title: 'Band of Brothers',
    imdb: {
      rating: 9.6
    }
  }
  {
    title: 'Planet Earth',
    imdb: {
      rating: 9.5
    }
  }
  {
    title: 'A Brave Heart: The Lizzie Velasquez Story',
    imdb: {
      rating: 9.4
    }
  }
```

Partie 3 – Mises à jour

Question 11 :

(Je fais avec Robin des bois parce que c'était plus simple pour le trouver)

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

```
num_mflix_comments : 0
poster : "https://m.media-amazon.com/images/M/MV5BYzRmMWIyNDEtYTRmYS00Y2FlLWJhOG…"
title : "Robin Hood"
fullplot : "Amid big-budget medieval pageantry, King Richard goes on the Crusades …"
▸ languages : Array (1)
released : 1922-10-18T00:00:00.000+00:00
▸ directors : Array (1)
▸ writers : Array (1)
▸ awards : Object
lastupdated : "2015-08-11 00:29:16.047000000"
year : 1922
▸ imdb : Object
▸ countries : Array (1)
type : "movie"
▸ tomatoes : Object
etat : "culte"
```

La catégorie état avec dedans « culte » a bien été rajouté.

Question 12 :

```
> db.movies.find({ title: "Inception" }, { "imdb.votes": 1, _id: 0 }).limit(5)
< {
    imdb: {
      votes: 1294646
    }
  }
> db.movies.updateOne({ title: "Inception" }, { $inc: { "imdb.votes": 100 } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.movies.find({ title: "Inception" }, { "imdb.votes": 1, _id: 0 }).limit(5)
< {
    imdb: {
      votes: 1294746
    }
```

Question 13 :

```
  consensus: 'Smart, innovative, and thrilling, Inception is that rare summer blockbuster that succeeds \
  rotten: 45,
  production: 'Warner Bros. Pictures',
  lastUpdated: 2015-09-12T17:13:32.000Z,
  fresh: 281
},
poster: 'https://m.media-amazon.com/images/M/MV5BMjAxMzY3NjcxNF5BMl5BanBnXkFtZTcwNTI5OTM0Mw@@._V1_SY1000_
num_mflix_comments: 1,
```

```
> db.movies.updateMany({}, { $unset: { poster: "" } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 21349,
    modifiedCount: 18044,
    upsertedCount: 0
  }
> db.movies.find({ title: "Inception" }, { _id: 0 }).limit(1)
```

```
    consensus: 'Smart, innovative, and thrilling, Inception is that rare summer blockbuster
    rotten: 45,
    production: 'Warner Bros. Pictures',
    lastUpdated: 2015-09-12T17:13:32.000Z,
    fresh: 281
  },
  num_mflix_comments: 1,
  released: 2010-07-16T00:00:00.000Z,
```

Question 14 :

```
> db.movies.find({ title: "Titanic" }, { directors: 1, _id: 0 }).limit(5)
< {}
  {
    directors: [
      'James Cameron'
    ]
  }
> db.movies.updateOne({ title: "Titanic" }, { $set: { directors:
    ["Il y avait assez de place sur la planche"] } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.movies.find({ title: "Titanic" }, { directors: 1, _id: 0 }).limit(5)
< {
    directors: [
      'Il y avait assez de place sur la planche'
    ]
```

Bon, James Cameron était déjà là, je l'ai donc changé.

Partie 4 – Requêtes complexes

Question 15 :

Fallait juste faire attention aux champs vides.

```
> db.movies.aggregate([{ $match: { "imdb.rating": { $exists: true, $ne: "" }, year: { $type: "number" } } },
< {
    _id: 1890,
    maxRating: 5.9
  }
  {
    _id: 1900,
    maxRating: 7.4
  }
  {
    _id: 1910,
    maxRating: 7.6
  }
  {
    _id: 1920,
    maxRating: 8.3
  }
  {
    _id: 1930,
```

Question 16 :

```
db.movies.find({ title: /^Star/ }, { title: 1, _id: 0 }).limit(5)
{
  title: 'Stars'
}
{
  title: 'Star!'
}
{
  title: 'Start the Revolution Without Me'
}
{
  title: 'Stardust'
}
{
  title: 'Star Wars: Episode IV - A New Hope'
```

Question 17 :

```
db.movies.find({ $where: "this.genres.length > 2" }, { title: 1, genres: 1, _id: 0 }).limit(2)
{
  genres: [
    'Animation',
    'Short',
    'Comedy'
  ],
  title: 'Winsor McCay, the Famous Cartoonist of the N.Y. Herald and His Moving Comics'
}
{
  genres: [
    'Animation',
    'Short',
    'Comedy'
  ],
  title: 'Gertie the Dinosaur'
}
```

Question 18 :

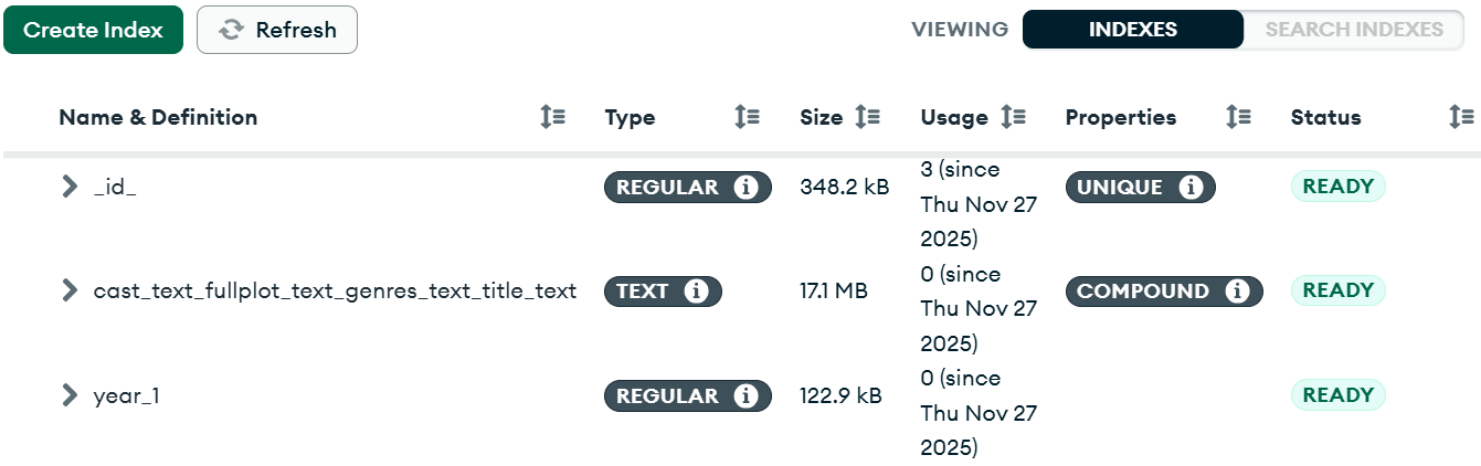
```
db.movies.find({ directors: "Christopher Nolan" }, { title: 1, directors: 1, _id: 0 }).limit(5)
{
  title: 'Following',
  directors: [
    'Christopher Nolan'
  ]
}
{
  title: 'Memento',
  directors: [
    'Christopher Nolan'
  ]
}
{
  title: 'Insomnia',
  directors: [
    'Christopher Nolan'
  ]
}
```

Partie 5 – Indexation

Question 19 et 20 :

En indexant, les requêtes filtrant ou triant par year seront plus rapides. Cependant cela prend de la place en mémoire et sur le disque. Et toutes opérations d'insertion ou de mise à jour sur year deviennent plus lentes car l'index doit être mis à jour.

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




```
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' }
]
```

Question 21 :

Sans indexation :

```
db.movies.find({ year: 1995 }).hint({ $natural: 1 }).explain("executionStats").executionStats
{
  executionSuccess: true,
  nReturned: 372,
  executionTimeMillis: 28,
  totalKeysExamined: 0,
  totalDocsExamined: 21349,
  executionStages: {
    isCached: false,
    stage: 'COLLSCAN',
    filter: { year: [Object] },
    nReturned: 372,
    executionTimeMillisEstimate: 21,
    works: 21350,
    advanced: 372,
    needTime: 20977,
```

Avec indexation :

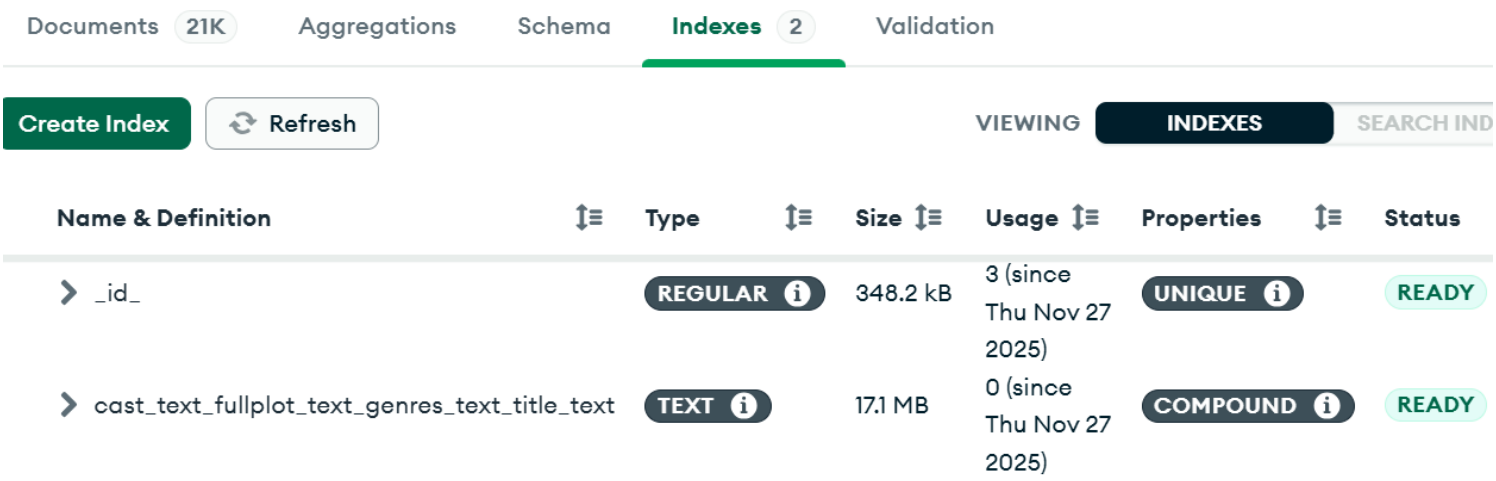
```
db.movies.find({ year: 1997 }).explain("executionStats").executionStats
{
  executionSuccess: true,
  nReturned: 439,
  executionTimeMillis: 2,
  totalKeysExamined: 439,
  totalDocsExamined: 439,
  executionStages: {
    isCached: false,
    stage: 'FETCH',
    nReturned: 439,
    executionTimeMillisEstimate: 0,
    works: 440,
    advanced: 439,
    needTime: 0,
```

On remarque clairement que le temps d’exécution est beaucoup plus

faible lorsqu'on utilise l'indexation ce qui est en effet le résultat attendu, en effet la recherche en utilisant des indexes demande dans un cas général beaucoup moins de calcul comme on a pu voir lors du cours de Base de Données avancée de l'année précédente.

Question 22 :

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



Question 23 :

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

```
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, 'imdb.rating': -1 },
    name: 'year_1_imdb.rating_-1'
  }
```

Ce type d'indexation peut être utile pour les requêtes qui :

Filtrent sur year et éventuellement sur imdb.rating ou qui Tri par imdb.rating à l'intérieur d'une année.

Par exemple :

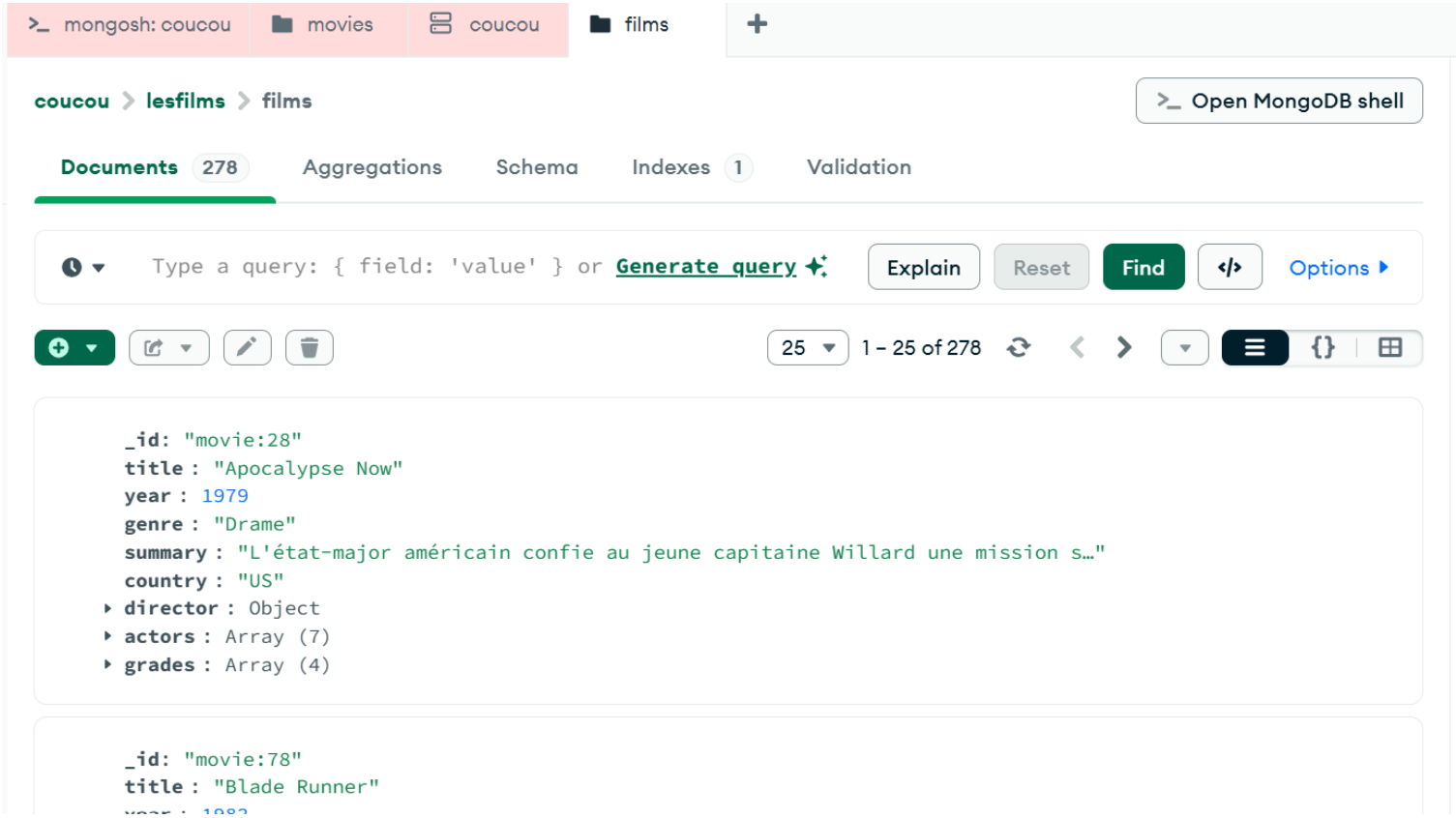
db.movies.find({ year: 2010 }).sort({ "imdb.rating": -1 }).limit(5)

Ici, MongoDB peut utiliser l'index pour filtrer par année et retourner directement les meilleurs ratings sans faire un tri supplémentaire.

TP1 – PART 3

Question 1 :

```
PS C:\Users\wirtz\Downloads> mongoimport --db lesfilms --collection films films.json --jsonArray
2025-11-28T13:57:38.758+0100    connected to: mongodb://localhost/
2025-11-28T13:57:38.848+0100    278 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\wirtz\Downloads>
```



Question 2 :

```
db.films.findOne()
{
  _id: 'movie:28',
  title: 'Apocalypse Now',
  year: 1979,
  genre: 'Drame',
  summary: "L'état-major américain c
  country: 'US',
  director: {
    _id: 'artist:1776',
    last_name: 'Ford Coppola',
    first_name: 'Francis',
    birth_date: 1939
  },
  actors: [
    {
      last_name: 'Fishburne',
      first_name: 'Laurence',
      birth_date: 1961
    },
    {
      last_name: 'Hall',
      first_name: 'Albert',
      birth_date: 1937
    }
  ],
  grades: [
    {
      note: 55,
      grade: 'B'
    },
    {
      note: 45,
      grade: 'B'
    },
    {
```

Question 3 :

```
db.films.find({ genre: "Action" }, { title: 1, _id: 0 }).limit(5)
{
  title: 'Kill Bill : Volume 1'
}
{
  title: 'Gladiator'
}
{
  title: 'Minority Report'
}
{
  title: 'Terminator'
}
{
  title: 'Terminator 2: Judgment Day'
```

Pour avoir la liste complète, faut retirer « .limit(5) ».

Question 4 :

```
> db.films.countDocuments({ genre: "Action" })
< 36
```

Question 5 :

```
> db.films.find({ genre: "Action", country: "FR" }, { title: 1, _id: 0 })
< {
    title: "L'Homme de Rio"
  }
  {
    title: 'Nikita'
  }
  {
    title: 'Les tontons flingueurs'
  }
```

Question 6 :

```
> db.films.find({ genre: "Action", country: "FR", year: 1963 }, { title: 1, _id: 0 })
< {
    title: 'Les tontons flingueurs'
  }
```

Question 7 :

Ah bah non, ça fait longtemps que je filtre déjà…

Question 8 :

De même je n'affiche jamais les identifiants…

Question 9 :

```
> db.films.find({ genre: "Action", country: "FR" }, { title: 1, grades: 1, _id: 0 }).limit(5)
< {
    title: "L'Homme de Rio",
    grades: [
      {
        note: 4,
        grade: 'D'
      },
      {
        note: 30,
        grade: 'E'
      },
      {
        note: 34,
        grade: 'E'
      },
      {
```

Question 10 :

```
> db.films.find(
    { genre: "Action", country: "FR", grades: { $elemMatch: { note: { $gt: 10 } } } },
    { title: 1, _id: 0 }
  ).limit(5)
< {
    title: "L'Homme de Rio"
  }
  {
    title: 'Nikita'
  }
  {
    title: 'Les tontons flingueurs'
  }
```

Là c'est si on veut exclure les films ayant uniquement des notes < 10.

```
> db.films.aggregate([
    { $match: { genre: "Action", country: "FR" } },
    { $unwind: "$grades" },
    { $group: { _id: "$_id", title: { $first: "$title" }, avgNote: { $avg: "$grades.note" } } },
    { $match: { avgNote: { $gt: 10 } } },
    { $project: { title: 1, avgNote: 1, _id: 0 } },
    { $limit: 5 }
  ])
< {
    title: 'Les tontons flingueurs',
    avgNote: 47.25
  }
  {
    title: "L'Homme de Rio",
    avgNote: 24
  }
  {
    title: 'Nikita',
    avgNote: 65
  }
```

Là c'est si on veut exclure les films ayant une moyenne < 10.

```
> db.films.find(
    { genre: "Action", country: "FR", grades: { $not: { $elemMatch: { note: { $lt: 10 } } } } },
    { title: 1, _id: 0 }
  ).limit(5)
< {
    title: 'Nikita'
  }
  {
    title: 'Les tontons flingueurs'
  }
```

Et là une seule note < 10 suffit pour l'enlever.

Question 12 :

```
> 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'
]
```

Question 13 :

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

Question 14 :

```
db.films.find({ "actors._id": { $in: ["artist:4", "artist:11", "artist:18"] } }, { title: 1, actors: 1, _id
```

Ou j'ai mal compris la question, ou il n'y a personne.

Question 15 :

```
> db.films.find({ summary: { $exists: false } }, { title: 1, _id: 0 })
<
> db.films.find({ summary: "" }, { title: 1, _id: 0 }).limit(5)
< {
    title: 'Star Wars, épisode IX'
  }
```

J’ai testé les deux possibilités.

Question 16 :

```
> db.films.find({ year: 1997, "actors.first_name": "Leonardo", "actors.last_name": "DiCaprio" }, { title: 1,
< {
    title: 'Titanic'
  }
```

Question 17 :

```
db.films.find({ $or: [ { year: 1997 }, { "actors.first_name": "Leonardo", "actors.last_name": "DiCaprio" }
{
  title: 'Jackie Brown',
  year: 1997
}
{
  title: 'Le monde perdu : Jurassic Park',
  year: 1997
}
{
  title: 'Starship Troopers',
  year: 1997
}
{
  title: 'Titanic',
  year: 1997
}
{
  title: 'Volte/Face',
```