TP1 : Redis et MongoDB :

Partie 1 :

Dans cette partie, on s'intéresse surtout à Redis. Redis est un SGBD NoSQL en mémoire. Il fonctionne comme un magasin clé-valeur, où les données sont associées à une clé unique et peuvent être récupérées à partir de cette clé. Grâce à son utilisation avec une autre base de données relationnelle, il arrive a mettre les données sur disque mais pas en temps réel donc si il y a une panne on ne récupère pas les dernières valeurs.

Dans ce TP, nous utiliserons Redis pour manipuler les principales opérations clé-valeur et comprendre comment fonctionne un SGBD en mémoire.

J'ai lancé Redis via Docker. A l'aide de la commande suivante que vous trouverez sur mon screen, j'ai réussi à lancer un conteneur Redis à partir de l'image redis.

```
PS C:\Users\Fei3> docker run --name monRedis redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
5325bfd41068: Pull complete
8116b2f58ddb: Pull complete
5d05fbd0b692: Pull complete
b4c7acd54b97: Pull complete
4f4fb700ef54: Pull complete
98ac9c138461: Pull complete
8e44f01296e3: Pull complete
Digest: sha256:43355efd22490e31ca14b9d569367d05121e2be61fd8e47937563ae2a80952ae
Status: Downloaded newer image for redis:latest
Starting Redis Server
1:C 27 Nov 2025 07:47:48.503 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 27 Nov 2025 07:47:48.503 * Redis version=8.4.0, bits=64, commit=00000000, modified=1, pid=1, just started
1:C 27 Nov 2025 07:47:48.503 * Configuration loaded
1:M 27 Nov 2025 07:47:48.503 * monotonic clock: POSIX clock_gettime
1:M 27 Nov 2025 07:47:48.504 * Running mode=standalone, port=6379.
1:M 27 Nov 2025 07:47:48.504 * <bf> RedisBloom version 8.4.0 (Git=unknown)
1:M 27 Nov 2025 07:47:48.504 * <bf> Registering configuration options: [
1:M 27 Nov 2025 07:47:48.504 * <bf>      { bf-error-rate       :      0.01 }
1:M 27 Nov 2025 07:47:48.504 * <bf>      { bf-initial-size     :       100 }
1:M 27 Nov 2025 07:47:48.504 * <bf>      { bf-expansion-factor :         2 }
1:M 27 Nov 2025 07:47:48.504 * <bf>      { cf-bucket-size      :         2 }
1:M 27 Nov 2025 07:47:48.504 * <bf>      { cf-initial-size     :      1024 }
1:M 27 Nov 2025 07:47:48.504 * <bf>      { cf-max-iterations   :        20 }
1:M 27 Nov 2025 07:47:48.504 * <bf>      { cf-expansion-factor :         1 }
1:M 27 Nov 2025 07:47:48.504 * <bf>      { cf-max-expansions   :        32 }
1:M 27 Nov 2025 07:47:48.504 * <bf> ]
```

puis j'ai pu accéder à mon client Redis (redis-cli) à l'intérieur du conteneur Docker nommé monRedis.

```
PS C:\Users\Fei3> docker exec -it monRedis redis-cli
127.0.0.1:6379>
```

Je suis donc sur le localhost qui écoute sur le port 6379.
Pour définir des clés et lui attribuer des valeurs je fais donc :

```
127.0.0.1:6379> SET cle1 "fatimazahra"
OK
```

```
127.0.0.1:6379> SET cle2:10 "arthur"
OK
```

J'ai donc ma clé valeur : (cle1, fatimazahra), (cle2, arthur) qui ont été définies.
Pour récupérer une clé :

```
127.0.0.1:6379> GET cle2:10
"arthur"
```

Pour supprimer une clé :

```
127.0.0.1:6379> del cle2:10
(integer) 1
```

Le 1 sert a dire que ma clé existe bien, qu'il a pu la retrouver et la supprimer.
Si je retente de la supprimer, il me renvoie 0 parce que ma clé n'existe plus et il ne peut plus la supprimer.

```
127.0.0.1:6379> del cle2:10
(integer) 0
```

L'un des cas d'utilisation de Redis c'est pour compter le nombre de visites d'un site.
Pour démontrer ceci, je vais commencer par créer une clé que je vais initialiser le compteur à 0 pour dire que pour l'instant je n'ai aucun visiteur sur le site.

```
127.0.0.1:6379> SET 27novembre 0
OK
```

J'incrémente ma clé :

```
127.0.0.1:6379> incr 27novembre
(integer) 1
127.0.0.1:6379> incr 27novembre
(integer) 2
127.0.0.1:6379> incr 27novembre
(integer) 3
127.0.0.1:6379> incr 27novembre
(integer) 4
127.0.0.1:6379> incr 27novembre
(integer) 5
```

Redis permet de traiter la concurrence donc le cas où plusieurs utilisateurs se connectent en même temps.
Pour décrémenter une clé:

```
127.0.0.1:6379> decr 27novembre
(integer) 4
127.0.0.1:6379> decr 27novembre
(integer) 3
127.0.0.1:6379> decr 27novembre
(integer) 2
```

Pour savoir la durée de vie d'une clé :

```
127.0.0.1:6379> ttl cle1
(integer) -1
```

Ceci veut dire que la durée de vie de ma cle1 est indéfinie c'est à dire jusqu'à ce que Redis ne lui reste plus de mémoire RAM.
Si je veux fixer une durée de vie d'une clé:

```
127.0.0.1:6379> expire cle1 120
(integer) 1
```

Ma cle1 expire au bout de 120s.
Si je veux voir combien il lui reste en durée de vie :

```
127.0.0.1:6379> ttl cle1
(integer) 26
```

On peut définir plusieurs structures de données avec Redis.
On va définir une liste :

```
127.0.0.1:6379> RPUSH mesCours "BDA"
(integer) 1
```

```
127.0.0.1:6379> RPUSH mesCours "NoSQL"
(integer) 2
```

Ici, j'ai inséré à droite de ma liste monCours les cours BDA et NoSQL.
Pour afficher le contenu de ma liste :

```
127.0.0.1:6379> LRANGE mesCours 0 -1
1) "BDA"
2) "NoSQL"
```

0 c'est pour dire qu'il doit afficher à partir du premier élément de ma liste et -1 veut dire qu'il doit afficher tout c e qu'il y a dans ma liste.
Donc si je veux récupérer que le premier élément l'affichage doit commencer à l'indice 0 et finir à 0 :

```
127.0.0.1:6379> LRANGE mesCours 0 0
1) "BDA"
```

Pour supprimer un élément de ma liste :

```
127.0.0.1:6379> LPOP mesCours
"BDA"
```

Le L dans LPOP c'est pour left donc il supprime l'élément à gauche de ma liste.
Pour cibler les éléments à droite, on utilise RPOP.
Si j'affiche ma liste je me retrouve avec :

```
127.0.0.1:6379> LRANGE mesCours 0 -1
1) "NoSQL"
```

On manipule les set maintenant, ce sont des listes ou les éléments doivent être distincts.
Dans une liste je peux ajouter un même élément autant de fois que je veux ce qui n'est pas le cas dans un set.
On déclare un set utilisateurs auquel je vais ajouter les prénoms de mes utilisateurs:

```
127.0.0.1:6379> SADD utilisateurs "arthur"
(integer) 1
127.0.0.1:6379> SADD utilisateurs "ethan"
(integer) 1
127.0.0.1:6379> SADD utilisateurs "dima"
(integer) 1
127.0.0.1:6379> SADD utilisateurs "catherine"
(integer) 1
```

si j'essaye d'ajouter ethan une deuxième fois je vois bien qu'il me renvoie 0 pour me dire qu'il ne le prend pas en compte comme ethan existe déjà dans mon set.

```
127.0.0.1:6379> SADD utilisateurs "ethan"
(integer) 0
```

Pour afficher les éléments d'un set:

```
127.0.0.1:6379> SMEMBERS utilisateurs
1) "arthur"
2) "ethan"
3) "dima"
4) "catherine"
```

Je ne peux pas supprimer un élément via son indice comme dans une liste classique car dans un set les indices n'ont pas d'importance. Pour supprimer un élément je précise sa valeur:

```
127.0.0.1:6379> SREM utilisateurs "arthur"
(integer) 1
```

Il me renvoie 1 pour me dire que arthur existe et qu'il a pu le supprimer. Si je réessaye, il me renvoie donc 0 pour m'informer que arthur n'existe plus dans mon set :

```
127.0.0.1:6379> SREM utilisateurs "arthur"
(integer) 0
```

Pour l'union des ensembles :
Je crée un nouveau set :

```
127.0.0.1:6379> SADD autesUtilisateurs "malo"
(integer) 1
127.0.0.1:6379> SADD autesUtilisateurs "agathe"
(integer) 1
```

L'union :

```
127.0.0.1:6379> SUNION utilisateurs autesUtilisateurs
1) "malo"
2) "catherine"
3) "ethan"
4) "agathe"
5) "dima"
```

On manipule des sets ordonnées maintenant :
Pour créer un set ordonne :

```
127.0.0.1:6379> ZADD score 10 "arthur"
(integer) 1
127.0.0.1:6379> ZADD score 9 "ethan"
(integer) 1
127.0.0.1:6379> ZADD score 5 "dima"
(integer) 1
127.0.0.1:6379> ZADD score 7 "catherine"
(integer) 1
```

Pour afficher les éléments d'un set ordonne :

```
127.0.0.1:6379> ZRANGE score 0 -1
1) "dima"
2) "catherine"
3) "ethan"
4) "arthur"
```

Les éléments ont été renvoyés par ordre croissant, du plus petit score au plus grand.
Si je veux les renvoyer dans l'ordre décroissant :

```
127.0.0.1:6379> ZREVRANGE score 0 -1
1) "arthur"
2) "ethan"
3) "catherine"
4) "dima"
```

Si on veut connaître l'indice d'un élément :

```
127.0.0.1:6379> ZRANK score "arthur"
(integer) 3
```

On manipule maintenant les hash:
Pour définir un hash :

```
127.0.0.1:6379> HSET cleH:11 username "fatimazahra"
(integer) 1
127.0.0.1:6379> HSET cleH:11 age "22"
(integer) 1
127.0.0.1:6379> HSET cleH:11 email "fatimazahra.elyounoussi@univ-paris13.fr"
(integer) 1
```

Pour récupérer tous les éléments de mon hash :

```
127.0.0.1:6379> HGETALL cleH:11
1) "username"
2) "fatimazahra"
3) "age"
4) "22"
5) "email"
6) "fatimazahra.elyounoussi@univ-paris13.fr"
```

Pour définir un hash en une fois pas ligne par ligne :

```
127.0.0.1:6379> HMSET cleM:12 username "arthur" age "23" email "arthur.bassette@univ-paris13.fr"
OK
```

On peut incrémenter un hash, je vais incrémenter l'âge d'Arthur de 10 :

```
127.0.0.1:6379> HINCRBY cleM:12 age 10
(integer) 33
```

On peut récupérer toutes les valeurs de la clé hash:

```
127.0.0.1:6379> HVALS cleM:12
1) "arthur"
2) "33"
3) "arthur.bassette@univ-paris13.fr"
```

=> Redis est schemaless, il n'a pas de schéma commun, il offre une flexibilité et un passage à l'échelle facile.

On manipule des PubSub maintenant, utilisés dans les applications en temps réel.

Je m'inscrit à un canal mescours.

```
127.0.0.1:6379> HVALS cleM:12
1) "arthur"
2) "33"
3) "arthur.bassette@univ-paris13.fr"
127.0.0.1:6379> SUBSCRIBE mescours user:1
1) "subscribe"
2) "mescours"
3) (integer) 1
1) "subscribe"
2) "user:1"
3) (integer) 2
Reading messages... (press Ctrl-C to quit or any key to type command)
```

Un utilisateur publie un cours :

```
127.0.0.1:6379> PUBLISH mescours "nouveau cours sur MongoDB"
(integer) 1
```

Je reçois le cours en temps réel :

```
127.0.0.1:6379> SUBSCRIBE mescours user:1
1) "subscribe"
2) "mescours"
3) (integer) 1
1) "subscribe"
2) "user:1"
3) (integer) 2
1) "message"
2) "mescours"
3) "nouveau cours sur MongoDB"
Reading messages... (press Ctrl-C to quit or any ke
```

=> tous les utilisateurs du canal reçoivent ce message.

Cette fois ci un utilisateur m'envoie à moi user1 un message :

```
127.0.0.1:6379> PUBLISH user:1 "Bonjour user 1"
(integer) 1
```

Je le reçois :

```
127.0.0.1:6379> SUBSCRIBE mescours user:1
1) "subscribe"
2) "mescours"
3) (integer) 1
1) "subscribe"
2) "user:1"
3) (integer) 2
1) "message"
2) "mescours"
3) "nouveau cours sur MongoDB"
1) "message"
2) "user:1"
3) "Bonjour user 1"
Reading messages... (press Ctrl-C to quit or any key to
```

Je peux aussi m'inscrire à plusieurs canaux, je m'inscris à tous les canaux qui commencent par mes:

```
127.0.0.1:6379> PSUBSCRIBE mes*
1) "psubscribe"
2) "mes*"
3) (integer) 1
Reading messages... (press Ctrl-C to qui
```

Un utilisateur publie une note sur mesnotes qui commence par mes:

```
127.0.0.1:6379> PUBLISH mesnotes "une nouvelle note"
(integer) 1
```

Je les reçois :

```
127.0.0.1:6379> PSUBSCRIBE mes*
1) "psubscribe"
2) "mes*"
3) (integer) 1
1) "pmessage"
2) "mes*"
3) "mesnotes"
4) "une nouvelle note"
Reading messages... (press Ctrl-C to quit or a
```

Pour avoir toutes les clés sauvegardes pour cette session :

```
127.0.0.1:6379> KEYS *
1) "27novembre"
2) "cleM:12"
3) "utilisateurs"
4) "score"
5) "mesCours"
6) "cleH:11"
7) "autesUtilisateurs"
```

=> Redis met à disposition de ses utilisateurs 16 bases de données par défaut.

Par défaut on est connecté sur la base 0. Si je veux changer de base :

```
127.0.0.1:6379> SELECT 1
OK
```

Je suis sur la base 1 !

Si je cherche a savoir toutes les clés sauvegardées, y'en a pas :

```
127.0.0.1:6379[1]> KEYS *
(empty array)
```

Je reviens sur la base 0.

Partie 2 :

On s'intéresse maintenant à MongoDB.

J'installe le jeu de données et je lance MongoDB.

Partie 1 – Filtrer et projeter les données

Afficher les 5 films sortis depuis 2015 :

```
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 ne
west 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/MV5BNzQ3OTY4NjAtNzM5OS00N2ZhLWJlOWUtYzYwZjNmOWRiMzcyXkEyXkFqcGdeQXVyMTM
xODk2OTU@._V1_SY1000_SX677_AL_.jpg',
    title: 'Jurassic World',
    fullplot: '22 years after the original Jurassic Park failed, the new park (also known as Jurassic World) is open for
 business. After years of studying genetics the scientists on the park genetically engineer a new breed of dinosaur. Whe
n everything goes horribly wrong, will our heroes make it off the island?',
    languages: [ 'English' ],
    released: ISODate('2015-06-12T00:00:00.000Z'),
    directors: [ 'Colin Trevorrow' ],
    writers: [
      'Rick Jaffa (screenplay)',
```

Trouver tous les films dont le genre est "Comedy" :

```
sample_mflix> db.movies.find({ genres: "Comedy" })
[
  {
    _id: ObjectId('573a1390f29313caabcd4803'),
    plot: 'Cartoon figures announce, via comic strip balloons, that they will move - and 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/MV5BYzg2NjNhNTctMjUxMi00ZWU4LWI3ZjYtNTI0NTQxNThjZTk2XkEyXkFqcGdeQXVyNzg5OTk2OA@@._V1_SY1000_SX677_AL_.jpg',
    title: 'Winsor McCay, the Famous Cartoonist of the N.Y. Herald and His Moving Comics',
    fullplot: 'Cartoonist Winsor McCay agrees to create a large set of drawings that will be photographed and made into a motion picture. The job requires plenty of drawing supplies, and the cartoonist must also overcome some mishaps caused by an assistant. Finally, 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)'
    ],
    awards: { wins: 1, nominations: 0, text: '1 win.' },
    lastupdated: '2015-08-29 01:09:03.630000000',
    year: 1911,
    imdb: { rating: 7.3, votes: 1034, id: 1737 },
    countries: [ 'USA' ],
    type: 'movie',
    tomatoes: {
      viewer: { rating: 3.4, numReviews: 89, meter: 47 },
      lastUpdated: ISODate('2015-08-20T18:51:24.000Z')
    }
  },
  {
    _id: ObjectId('573a1390f29313caabcd50e5'),
    plot: 'The cartoonist, Winsor McCay, brings the Dinosaurus back to life in the figure of his latest creation, Gertie the Dinosaur.',
    genres: [ 'Animation', 'Short', 'Comedy' ],
    runtime: 12,
    cast: [ 'Winsor McCay', 'George McManus', 'Roy L. McCardell' ],
    num_mflix_comments: 0,
    poster: 'https://m.media-amazon.com/images/M/MV5BMTQxNzI4ODQ3NF5BMl5BanBnXkFtZTgwNzY5NzMwMjE@._V1_SY1000_SX677_AL_.jpg',
    title: 'Gertie the Dinosaur',
    fullplot: 'Winsor Z. McCay bets another cartoonist that he can animate a dinosaur. So he draws a big friendly herbivore called Gertie. Then he get into his own picture. Gertie walks through the picture, eats a tree, meets her creator, and takes him carefully on her back for a ride.',
    languages: [ 'English' ],
    released: ISODate('1914-09-15T00:00:00.000Z'),
```

Afficher les sortis entre entre 2000 et 2005 :

```
sample_mflix> db.movies.find({year: {$gte: 2000, $lte: 2005}}, {title: 1, year:
... 1}).pretty()
[
  {
    _id: ObjectId('573a1393f29313caabcdcb42'),
    title: 'Kate & Leopold',
    year: 2001
  },
  {
    _id: ObjectId('573a1398f29313caabceb1fe'),
    title: 'Crime and Punishment',
    year: 2002
  },
  {
    _id: ObjectId('573a139af29313caabcf0718'),
    year: 2001,
    title: 'Glitter'
  },
  {
    _id: ObjectId('573a139af29313caabcf0782'),
    year: 2000,
    title: 'In the Mood for Love'
  },
  {
    _id: ObjectId('573a139af29313caabcf0809'),
    year: 2003,
    title: 'The Manson Family'
  },
  {
    _id: ObjectId('573a139af29313caabcf0869'),
    title: 'The Dancer Upstairs',
    year: 2002
  },
  {
    _id: ObjectId('573a139af29313caabcf0d0b'),
    year: 2000,
    title: 'State and Main'
  },
  {
    _id: ObjectId('573a139af29313caabcf0e7f'),
    title: 'April Captains',
    year: 2000
  },
  {
```

Afficher les films de genres “Drama” ET “Romance”:

```
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'
  },
  {
    _id: ObjectId('573a1391f29313caabcd7b98'),
    genres: [ 'Drama', 'Romance', 'Thriller' ],
    title: 'He Who Gets Slapped'
  },
  {
    _id: ObjectId('573a1391f29313caabcd7da6'),
    genres: [ 'Drama', 'Romance' ],
    title: 'Wild Oranges'
  },
  {
    _id: ObjectId('573a1391f29313caabcd89aa'),
    genres: [ 'Drama', 'Romance', 'War' ],
    title: 'Wings'
  },
  {
    _id: ObjectId('573a1391f29313caabcd91d7'),
```

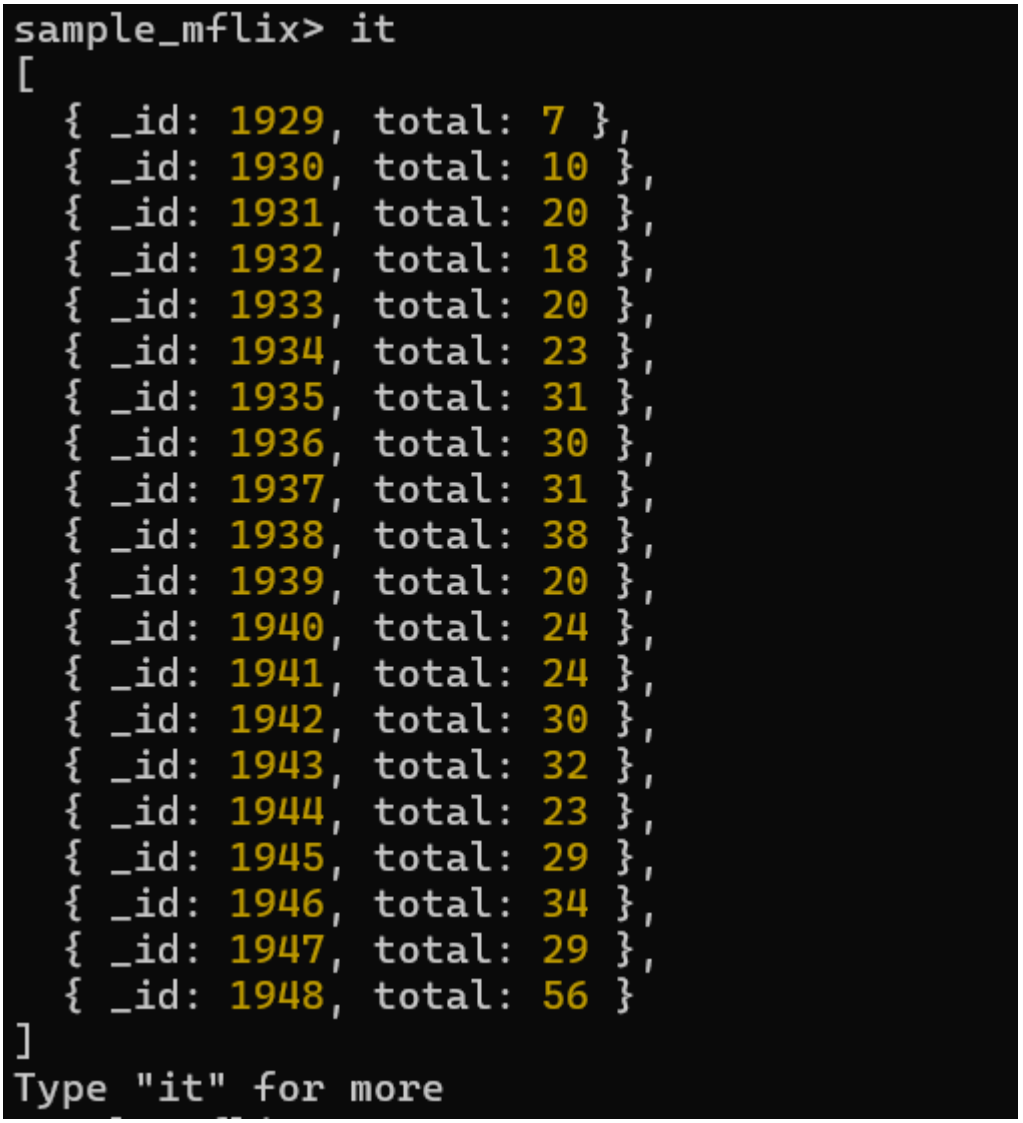
Afficher les films sans champ rated :

```
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 Comics'
  },
  {
    _id: ObjectId('573a1390f29313caabcd50e5'),
    title: 'Gertie the Dinosaur'
  },
  {
    _id: ObjectId('573a1390f29313caabcd516c'),
    title: 'In the Land of the Head Hunters'
  },
  {
    _id: ObjectId('573a1390f29313caabcd5293'),
    title: 'The Perils of Pauline'
  },
  { _id: ObjectId('573a1390f29313caabcd5a93'), title: 'Civilization' },
  {
    _id: ObjectId('573a1390f29313caabcd6223'),
    title: 'The Poor Little Rich Girl'
  },
  {
    _id: ObjectId('573a1390f29313caabcd6377'),
    title: 'Wild and Woolly'
  },
  { _id: ObjectId('573a1390f29313caabcd63d6'), title: 'The Blue Bird' },
  { _id: ObjectId('573a1391f29313caabcd6ea2'), title: 'The Saphead' },
```

Partie 2 – Agrégation

Afficher le nombre de films par année :

```
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
```



Afficher la moyenne des notes IMDb par genre:

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

Afficher le nombre de films par pays :

```
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
```

Afficher les top 5 réalisateurs :

```
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: 'Sidney Lumet', total: 29 },
  { _id: 'Steven Spielberg', total: 29 }
]
```

Afficher les films triés par note IMDb :

```
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',
    imdb: { rating: '' }
  },
  {
    _id: ObjectId('573a13f3f29313caabdde93d'),
    title: 'The Birth of Sakè',
    imdb: { rating: '' }
```

Partie 3 – Mises à jour :

Ajouter un champ etat :

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

Incrémenter les votes IMDb de 100, par exemple :

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

Supprimer le le champ poster :

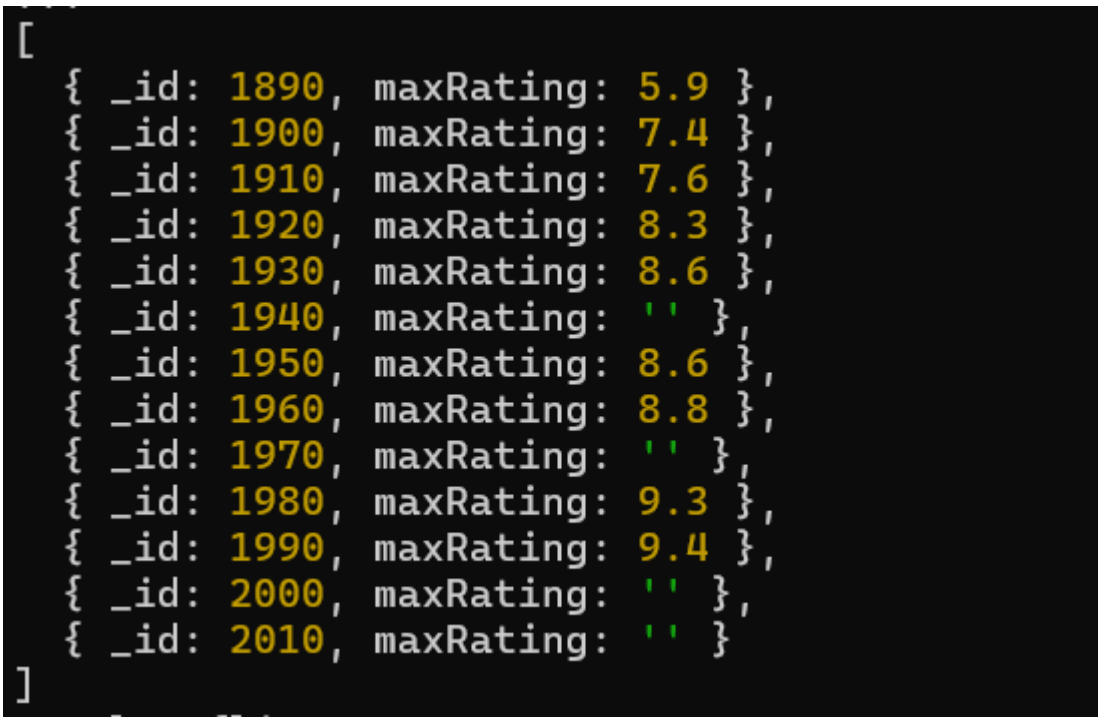
```
sample_mflix> db.movies.updateMany({}, {$unset: {poster: ""}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 21349,
  modifiedCount: 18044,
  upsertedCount: 0
}
```

Modifier le réalisateur :

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

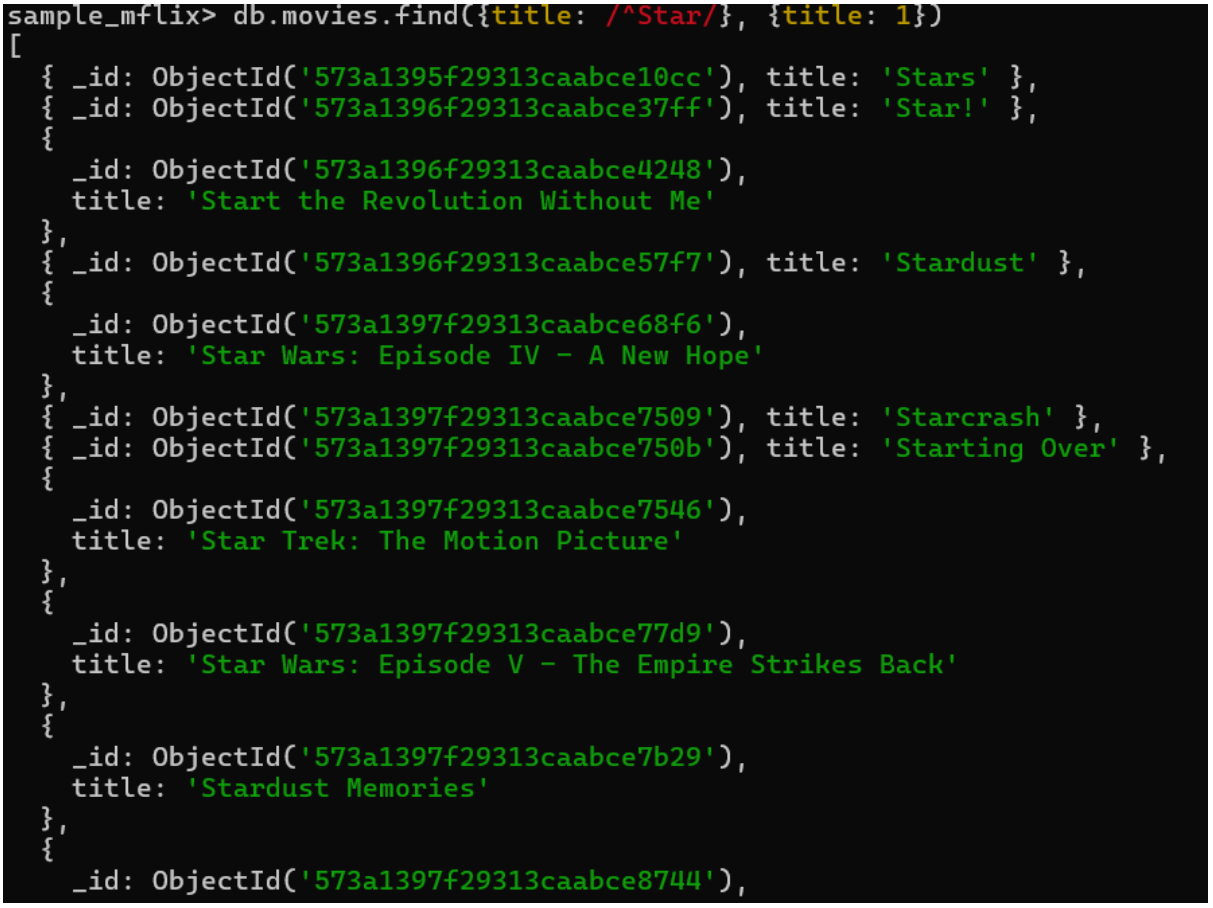
Partie 4 – Requêtes complexes

Afficher les films les mieux notés par décennie :



```
[
  { _id: 1890, maxRating: 5.9 },
  { _id: 1900, maxRating: 7.4 },
  { _id: 1910, maxRating: 7.6 },
  { _id: 1920, maxRating: 8.3 },
  { _id: 1930, maxRating: 8.6 },
  { _id: 1940, maxRating: '' },
  { _id: 1950, maxRating: 8.6 },
  { _id: 1960, maxRating: 8.8 },
  { _id: 1970, maxRating: '' },
  { _id: 1980, maxRating: 9.3 },
  { _id: 1990, maxRating: 9.4 },
  { _id: 2000, maxRating: '' },
  { _id: 2010, maxRating: '' }
]
```

Afficher les films dont le titre commence par “Star” :



```
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'
  },
  {
    _id: ObjectId('573a1397f29313caabce77d9'),
    title: 'Star Wars: Episode V - The Empire Strikes Back'
  },
  {
    _id: ObjectId('573a1397f29313caabce7b29'),
    title: 'Stardust Memories'
  },
  {
    _id: ObjectId('573a1397f29313caabce8744'),
```

Afficher les films avec plus de 2 genres :

```
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 Comics'
  },
  {
    _id: ObjectId('573a1390f29313caabcd50e5'),
    genres: [ 'Animation', 'Short', 'Comedy' ],
    title: 'Gertie the Dinosaur'
  },
  {
    _id: ObjectId('573a1390f29313caabcd587d'),
    genres: [ 'Biography', 'Crime', 'Drama' ],
    title: 'Regeneration'
  },
  {
    _id: ObjectId('573a1390f29313caabcd6223'),
    genres: [ 'Comedy', 'Drama', 'Family' ],
    title: 'The Poor Little Rich Girl'
  },
  {
    _id: ObjectId('573a1390f29313caabcd6377'),
    genres: [ 'Comedy', 'Western', 'Romance' ],
    title: 'Wild and Woolly'
  },
  {
    _id: ObjectId('573a1391f29313caabcd68d0'),
```

Afficher les films de Christopher Nolan :

```
sample_mflix> db.movies.find({directors: "Christopher Nolan"}, {title: 1, year: 1,
... "imdb.rating": 1})
[
  {
    _id: ObjectId('573a139df29313caabcf8dd4'),
    imdb: { rating: 7.6 },
    year: 1998,
    title: 'Following'
  },
  {
    _id: ObjectId('573a13a0f29313caabd05acc'),
    imdb: { rating: 8.5 },
    year: 2000,
    title: 'Memento'
  },
  {
    _id: ObjectId('573a13a5f29313caabd1605f'),
    imdb: { rating: 7.2 },
    year: 2002,
    title: 'Insomnia'
  },
  {
    _id: ObjectId('573a13aef29313caabd2c349'),
    title: 'Batman Begins',
    year: 2005,
    imdb: { rating: 8.3 }
  },
  {
    _id: ObjectId('573a13b5f29313caabd42722'),
    imdb: { rating: 9 },
```

Partie 5 – Indexation

Créer un index sur year :

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

Vérifier les index existants :

```
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' }
]
```

Comparer deux requêtes avec et sans index (utiliser explain()) :

```
sample_mflix> db.movies.find({year: 1995}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'sample_mflix.movies',
    parsedQuery: { year: { '$eq': 1995 } },
    indexFilterSet: false,
    queryHash: '82E400C1',
    planCacheShapeHash: '82E400C1',
    planCacheKey: '5F2A8919',
    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,
        direction: 'forward',
```

```
totalDocsExamined: 372,
```

```
executionTimeMillisEstimate: 0,
```

Supprimer l'index sur year :

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

créer un index composé sur year et imdb.rating :

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

Partie 3 :

1/ Vérifier que le donnees ont ete importees, en les comptant par exemple :

```
lesfilms> db.films.countDocuments()
278
```

J'utilise la fonction countDocuments() pour vérifier que tous les documents ont bien été importés dans la collection films. La valeur retournée correspond au nombre total de films présents dans la base lesfilms.

2/ Récupérer un seul document pour regarder la structure complète d'un film :

```
lesfilms> db.films.findOne()
{
  _id: 'movie:116',
  title: 'Match point',
  year: 2005,
  genre: 'Drame',
  summary: "Jeune professeur de tennis issu d'un milieu modeste, Chris Wilton se fait embaucher dans un club huppé des beaux quarti
ers de Londres. Il ne tarde pas à sympathiser avec Tom Hewett, un jeune homme de la haute société avec qui il partage sa passion po
ur l'opéra. Très vite, Chris fréquente régulièrement les Hewett et séduit Chloe, la sœur de Tom. Alors qu'il s'apprête à l'épouser
et qu'il voit sa situation sociale se métamorphoser, il fait la connaissance de la ravissante fiancée de Tom, Nola Rice, une jeune
Américaine venue tenter sa chance comme comédienne en Angleterre...",
  country: 'US',
  director: {
    _id: 'artist:1243',
    last_name: 'Allen',
    first_name: 'Woody',
    birth_date: 1935
  },
  actors: [
    {
      last_name: 'Rhys Meyers',
      first_name: 'Jonathan',
      birth_date: 1977
    },
    {
      last_name: 'Johansson',
      first_name: 'Scarlett',
      birth_date: 1984
    },
    { last_name: 'Mortimer', first_name: 'Emily', birth_date: 1971 },
    { last_name: 'Cox', first_name: 'Brian', birth_date: 1946 },
    { last_name: 'Wilton', first_name: 'Penelope', birth_date: 1946 }
```

3/ Afficher la liste des films d'action :

```
lesfilms> db.films.find(
...   { genre: "Action" }
... )
...
[
  {
    _id: 'movie:24',
    title: 'Kill Bill : Volume 1',
    year: 2003,
    genre: 'Action',
    summary: "Au cours d'une cérémonie de mariage en plein désert, un commando fait irruption dans la chapelle et tire sur les conv
ives. Laissée pour morte, la Mariée enceinte retrouve ses esprits après un coma de quatre ans. Celle qui a auparavant exercé les fo
nctions de tueuse à gages au sein du Détachement International des Vipères Assassines n'a alors plus qu'une seule idée en tête : ve
nger la mort de ses proches en éliminant tous les membres de l'organisation criminelle, dont leur chef Bill qu'elle se réserve pour
 la fin.",
    country: 'US',
    director: {
      _id: 'artist:138',
      last_name: 'Tarantino',
      first_name: 'Quentin',
      birth_date: 1963
    },
    actors: [
      { last_name: 'Thurman', first_name: 'Uma', birth_date: 1970 },
      { last_name: 'Liu', first_name: 'Lucy', birth_date: 1968 },
      { last_name: 'Carradine', first_name: 'David', birth_date: 1936 },
      { last_name: 'Madsen', first_name: 'Michael', birth_date: 1957 },
      { last_name: 'Hannah', first_name: 'Daryl', birth_date: 1960 },
      { last_name: 'A. Fox', first_name: 'Vivica', birth_date: 1964 },
      { last_name: 'Parks', first_name: 'Michael', birth_date: 1940 },
      { last_name: 'Chiba', first_name: 'Sonny', birth_date: 1939 },
```

4/ Afficher le nombre de films d'action :

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

5/ Afficher les films d'action produits en France :

```
lesfilms> db.films.find(
...     { genre: "Action", country: "FR" }
... )
...
[
  {
    _id: 'movie:4034',
    title: "L'Homme de Rio",
    year: 1964,
    genre: 'Action',
    summary: "Le deuxième classe Adrien Dufourquet est témoin de l'enlèvement de sa fiancée Agnès, fille d'un célèbre ethnologue. I
l 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
      },
      { last_name: 'Celi', first_name: 'Adolfo', birth_date: 1922 },
      { last_name: 'Servais', first_name: 'Jean', birth_date: 1910 },
      {
        last_name: 'Dorléac',
        first_name: 'Françoise',
        birth_date: 1942
      },
```

6/ Afficher les films d'action produits en France en 1963 :

```
lesfilms> 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 Mexicain fait promettre à son ami d'enfance, Fernand Naudin, de veiller sur ses intérêts et s
a 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'un garde du corps, Fernand impose d'emblée sa loi. Cependant, le belle Patricia lu
i réserve quelques surprises !",
    country: 'FR',
    director: {
      _id: 'artist:18563',
      last_name: 'Lautner',
      first_name: 'Georges',
      birth_date: 1926
    },
    actors: [
      { last_name: 'Ventura', first_name: 'Lino', birth_date: 1919 },
      { last_name: 'Frank', first_name: 'Horst', birth_date: 1929 },
      { last_name: 'Dalban', first_name: 'Robert', birth_date: 1903 },
      { last_name: 'Blier', first_name: 'Bernard', birth_date: 1916 },
      { last_name: 'Rich', first_name: 'Claude', birth_date: 1929 },
      { last_name: 'Sinjen', first_name: 'Sabine', birth_date: 1942 },
      { last_name: 'Lefebvre', first_name: 'Jean', birth_date: 1920 },
      { last_name: 'Bertin', first_name: 'Pierre', birth_date: 1891 },
      { last_name: 'Blanche', first_name: 'Francis', birth_date: 1919 }
```

7/ Films d'action français sans afficher le champ grades :

La commande initiale provoque l'erreur "Cannot do exclusion on field grades in inclusion projection".

Pour corriger, j'ai utilisé soit uniquement des exclusions :

```
lesfilms> db.films.find(
...   { genre: "Action", country: "FR" },
...   { _id: 0, grades: 0 }
... )
...
[
  {
    title: "L'Homme de Rio",
    year: 1964,
    genre: 'Action',
    summary: "Le deuxième classe Adrien Dufourquet 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
      },
      { last_name: 'Celi', first_name: 'Adolfo', birth_date: 1922 },
      { last_name: 'Servais', first_name: 'Jean', birth_date: 1910 },
      {
        last_name: 'Dorléac',
        first_name: 'Françoise',
        birth_date: 1942
      },
      { last_name: 'Renant', first_name: 'Simone', birth_date: 1911 }
    ]
  },
  {
    title: 'Nikita',
    year: 1990,
    genre: 'Action',
    summary: "Le braquage d'une pharmacie par une bande de junkies en manque de drogue tourne mal : une fusillade cause la mort de plusieurs personnes dont un p
```

Soit uniquement des inclusions :

```
lesfilms> db.films.find(
...   { genre: "Action", country: "FR" },
...   { _id: 0, title: 1, year: 1, genre: 1, country: 1 }
... )
...
[
  {
    title: "L'Homme de Rio",
    year: 1964,
    genre: 'Action',
    country: 'FR'
  },
  { title: 'Nikita', year: 1990, genre: 'Action', country: 'FR' },
  {
    title: 'Les tontons flingueurs',
    year: 1963,
    genre: 'Action',
    country: 'FR'
  }
]
```

8/ Masquer les ids des films :

```
lesfilms> db.films.find(
...   { genre: "Action", country: "FR" },
...   { _id: 0 }
... )
...
[
  {
    title: "L'Homme de Rio",
    year: 1964,
    genre: 'Action',
    summary: "Le deuxième classe Adrien Dufourquet 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
      },
      { last_name: 'Celi', first_name: 'Adolfo', birth_date: 1922 },
      { last_name: 'Servais', first_name: 'Jean', birth_date: 1910 },
      {
        last_name: 'Dorléac',
        first_name: 'Françoise',
        birth_date: 1942
      },
      { last_name: 'Renant', first_name: 'Simone', birth_date: 1911 }
    ],
    grades: [
      { note: 4, grade: 'D' },
      { note: 30, grade: 'E' },
      { note: 34, grade: 'E' },
      { note: 28, grade: 'F' }
    ]
```

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

```
lesfilms> db.films.find(
...   { genre: "Action", country: "FR" },
...   { _id: 0, title: 1, grades: 1 }
... )
...
[
  {
    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' },
      { note: 62, grade: 'D' }
    ]
  },
  {
    title: 'Les tontons flingueurs',
    grades: [
      { note: 35, grade: 'C' },
      { note: 48, grade: 'F' },
      { note: 64, grade: 'C' },
      { note: 42, grade: 'F' }
    ]
  }
]
```

10/ Afficher les titres et les notes des films d’action réalisés en France et ayant obtenus une note supérieure à 10 :

```
lesfilms> db.films.find(
...   { genre: "Action", country: "FR", "grades.note": { $gt: 10 } },
...   { _id: 0, title: 1, grades: 1 }
... )
...
[
  {
    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' },
      { note: 62, grade: 'D' }
    ]
  },
  {
    title: 'Les tontons flingueurs',
    grades: [
      { note: 35, grade: 'C' },
      { note: 48, grade: 'F' },
      { note: 64, grade: 'C' },
      { note: 42, grade: 'F' }
    ]
  }
]
```

11/ Afficher les films d’action réalisés en France ayant obtenus que des notes supérieures à 10 :

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

12/ Afficher les différents genres de la base lesfilms:

```
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 :

14/ Afficher tous les filmes dans lesquels joue au moins un des artistes suivants [”artist:4”,”artist:18”,”artist:11”] :

Dans cette base, les identifiants de la forme artist:XXX sont associés aux réalisateurs via le champ director._id, alors que la question semble parler d’artistes au sens d’acteurs.

De plus, les identifiants proposés dans l’énoncé (artist:4, artist:18, artist:11) n’existent pas dans les données réelles : aucune valeur de director._id ne correspond à ces IDs.

15/ Afficher tous les films qui n’ont pas de résumé :

```
lesfilms> db.films.find(
...   {
...     $or: [
...       { summary: { $exists: false } },
...       { summary: null },
...       { summary: "" }
...     ]
...   }
... )
...
[
  {
    _id: 'movie:181812',
    title: 'Star Wars, épisode IX',
    year: 2019,
    genre: 'Science-Fiction',
    summary: '',
    country: 'GB',
    director: {
      _id: 'artist:15344',
      last_name: 'Abrams',
      first_name: 'J.J.',
      birth_date: 1966
    },
    actors: [
      { last_name: 'Hamill', first_name: 'Mark', birth_date: 1951 },
      { last_name: 'Fisher', first_name: 'Carrie', birth_date: 1956 },
      {
        last_name: 'Dee Williams',
        first_name: 'Billy',
        birth_date: 1937
      },
      { last_name: 'Isaac', first_name: 'Oscar', birth_date: 1979 },
      { last_name: 'Boyega', first_name: 'John', birth_date: 1992 },
      { last_name: 'Driver', first_name: 'Adam', birth_date: 1983 },
      { last_name: 'Ridley', first_name: 'Daisy', birth_date: 1992 },
      {
        last_name: 'Marie Tran',
        first_name: 'Kelly',
        birth_date: 1989
```

16/ Afficher tous les films tournés avec Leonardo DiCaprio en 1997 :

```
lesfilms> db.films.find(
...   {
...     year: 1997,
...     "actors.first_name": "Leonardo",
...     "actors.last_name": "DiCaprio"
...   }
... )
...
[
  {
    _id: 'movie:597',
    title: 'Titanic',
    year: 1997,
    genre: 'Drame',
    summary: 'Southampton, 10 avril 1912. Le paquebot le plus grand et le plus moderne du monde, réputé pour son insubmersibilité, le « Titanic », appareille pour son premier voyag
e. 4 jours plus 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: 'Winslet', 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_date: 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 :

```
lesfilms> db.films.find(
...   {
...     $or: [
...       { "actors.first_name": "Leonardo", "actors.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 F
BI lui signifient 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: 'Tucker', first_name: 'Chris', birth_date: 1971 },
      { last_name: 'De Niro', first_name: 'Robert', birth_date: 1943 },
      { last_name: 'Grier', first_name: 'Pam', birth_date: 1949 },
      {
        last_name: 'L. Jackson',
        first_name: 'Samuel',
        birth_date: 1948
      },
      { last_name: 'Keaton', first_name: 'Michael', birth_date: 1951 },
      { last_name: 'Fonda', first_name: 'Bridget', birth_date: 1964 },
      { last_name: 'Bowen', first_name: 'Michael', birth_date: 1953 },
      { last_name: 'Forster', first_name: 'Robert', birth_date: 1941 }
    ],
    grades: [
      { note: 16, grade: 'E' },
      { note: 28, grade: 'C' },
      { note: 18, grade: 'D' },
      { note: 39, grade: 'B' }
    ]
  },
```