

TANTOT Etienne et ABDELLI Hamza

# RAPPORT PROJET

NoSQL



# Sommaire

Objectifs du projet	3
Description du Projet	3
Notre démarche	3
Difficultés rencontrées	5
Améliorations possibles	7
Apprentissage	7
Questions	8
Conclusion	32

# Objectif du projet

L'objectif de ce projet est d'explorer et d'interroger des bases de données NoSQL en utilisant MongoDB (base orientée document) et Neo4j (base orientée graphe). À travers ce projet, nous avons développé une application Python capable d'interagir avec ces bases de données et d'exécuter différentes requêtes pour extraire et analyser des informations.

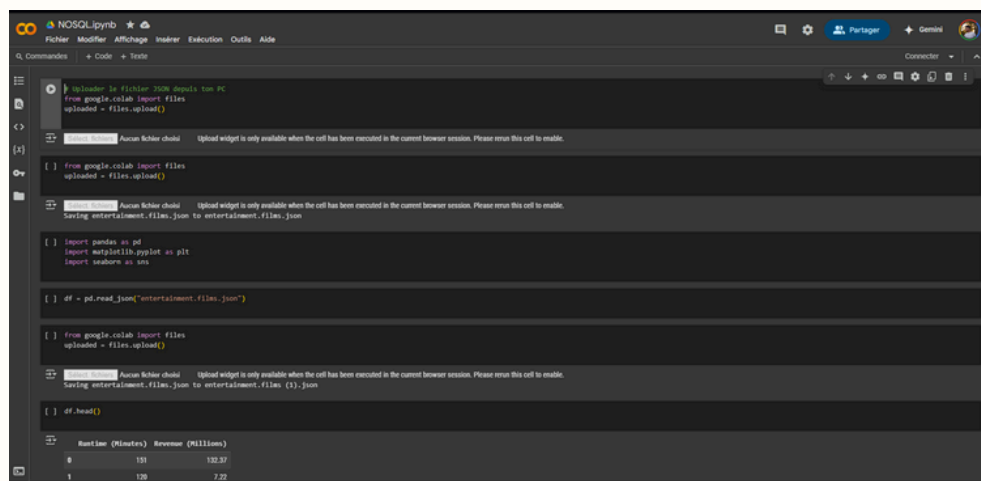
## Description du projet

L'application développée se connecte à des instances cloud de MongoDB et Neo4j pour :

- Récupérer des données via des requêtes NoSQL
  - Effectuer des analyses statistiques et graphiques
  - Visualiser les résultats à l'aide de bibliothèques Python comme Matplotlib et Seaborn
  - Intégrer les données de MongoDB dans Neo4j pour enrichir les relations entre les entités
- Les principales tâches réalisées incluent :
- La connexion sécurisée aux bases de données
  - L'extraction et l'analyse des données
  - La visualisation des résultats sous forme d'histogrammes, de graphes et de réseaux

## Notre démarche

Nous avons utilisé un notebook Jupyter sur GoogleColab pour la création des données qui sert de database pour le projet.



```
from google.colab import files
uploaded = files.upload()

from google.colab import files
uploaded = files.upload()

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

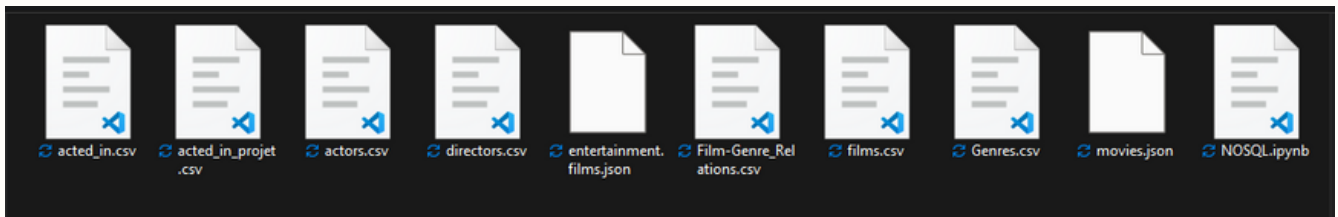
df = pd.read_json("entertainment_files.json")

from google.colab import files
uploaded = files.upload()

df.head()
```

Runtime (Minutes)	Revenue (Millions)	
0	131	132.37
1	120	7.22

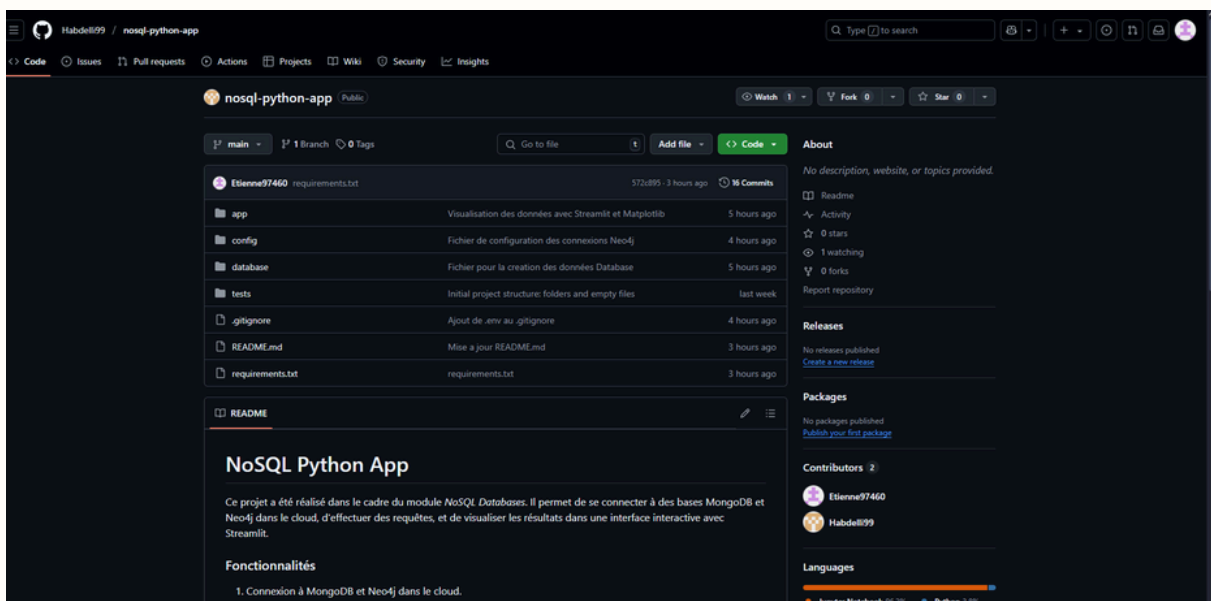
Ainsi, cela nous a permis de donner les databases pour le projet tel que des fichiers JSON et CSV contenant des informations sur les films, acteurs et relations entre eux.



### Exemple: movies.json

```
database > {} movies.json > ...
1 {"_id": "100", "_rev": "1-5993984b950851d9cb12590ec576a7f", "title": "The Departed", "genre": "Crime,Drama,Thriller", "Description": "An undercover cop and a mole in the police attempt t
2 {"_id": "21", "_rev": "1-f9df16eb7f3845ec7b1bd626e018b3c8", "title": "Gold", "genre": "Adventure,Drama,Thriller", "Description": "Kenny Wells, a prospector desperate for a lucky break, te
3 {"_id": "31", "_rev": "1-05e7f8b596d62a9b7e99f6e54a60efb3", "title": "Why Him?", "genre": "Comedy", "Description": "A holiday gathering threatens to go off the rails when Ned Fleming rea
4 {"_id": "18", "_rev": "1-14e438226f7c8d1bb309be480195472", "title": "Passengers", "genre": "Adventure,Drama,Romance", "Description": "A spacecraft traveling to a distant colony planet an
5 {"_id": "13", "_rev": "1-ce9ed34708f27efad4768001bfed7415", "title": "Rogue One", "genre": "Action,Adventure,Sci-Fi", "Description": "The Rebel Alliance makes a risky move to steal the pl
6 {"_id": "32", "_rev": "1-d508cfe795803a127bd87b4711ea7dc", "title": "Top Dog", "genre": "Drama,Thriller", "Description": "A wealthy art gallery owner is haunted by her ex-husband's novel
7 {"_id": "22", "_rev": "1-5ed667c227a26880cf9db8923d89e7ce", "title": "Manchester by the Sea", "genre": "Drama", "Description": "A depressed uncle is asked to take care of his teenage neph
8 {"_id": "1", "_rev": "1-1e23f81a16cc7cb9c4ecc0f644302e6", "title": "Guardians of the Galaxy", "genre": "Action,Adventure,Sci-Fi", "Description": "A group of intergalactic criminals are f
9 {"_id": "51", "_rev": "1-efdfbecc4f0bb1b642fde4a48ce766f9", "title": "Star Wars: Episode VII - The Force Awakens", "genre": "Action,Adventure,Fantasy", "Description": "Three decades after
10 {"_id": "12", "_rev": "1-5c19b01f2f8e8c3025e67c75ebe8461", "title": "Hidden Figures", "genre": "Biography,Drama,History", "Description": "The story of a team of female African-American
11 {"_id": "33", "_rev": "1-d26bc8aef805e9590c9c72df3a03d4b4", "title": "X-Men: Apocalypse", "genre": "Action,Adventure,Sci-Fi", "Description": "After the re-emergence of the world's first m
12 {"_id": "36", "_rev": "1-c8634397acc02875438aa511debdfdfbf", "title": "Captain America: Civil War", "genre": "Action,Adventure,Sci-Fi", "Description": "Political interference in the Avenger
13 {"_id": "17", "_rev": "1-98ba454503125820ce6d8f53667a1106", "title": "Hacksaw Ridge", "genre": "Biography,Drama,History", "Description": "WWII American Army Medic Desmond T. Doss, who se
14 {"_id": "35", "_rev": "1-35449a1b5339171ea6750455ab09de66", "title": "Resident Evil: The Final Chapter", "genre": "Action,Horror,Sci-Fi", "Description": "Alice returns to where the night
15 {"_id": "14", "_rev": "1-0faa2af40a25123a5755def3189e830", "title": "Moana", "genre": "Animation,Adventure,Comedy", "Description": "In Ancient Polynesia, when a terrible curse incurred b
16 {"_id": "25", "_rev": "1-9f1ac796de8da7a904cbaf9a0ac47c0", "title": "Independence Day: Resurgence", "genre": "Action,Adventure,Sci-Fi", "Description": "Two decades after the first Indep
17 {"_id": "28", "_rev": "1-3d8b82e23a6609c68545a592fa4506a7", "title": "Arrival", "genre": "Drama,Mystery,Sci-Fi", "Description": "When twelve mysterious spacecraft appear around the world,
18 {"_id": "55", "_rev": "1-096ba8c939a6c54dad042cfe52828a60", "title": "The Dark Knight", "genre": "Action,Crime,Drama", "Description": "When the menace known as the Joker wreaks havoc and
19 {"_id": "42", "_rev": "1-9063f07f4b31d42f6db8d36c9d21ffac", "title": "Moonlight", "genre": "Drama", "Description": "A chronicle of the childhood, adolescence and burgeoning adulthood of a
20 {"_id": "63", "_rev": "1-836e95840e9899525f798f97273eb0868", "title": "The Girl on the Train", "genre": "Crime,Drama,Mystery", "Description": "A divorcee becomes entangled in a missing per
21 {"_id": "26", "_rev": "1-461f893af22a09e8ce0ac2d9be74888a", "title": "Paris pieds nus", "genre": "Comedy", "Description": "Fiona visits Paris for the first time to assist her myopic Aunt.
22 {"_id": "39", "_rev": "1-d8585a08da328b956793be2289fd2b0c", "title": "The Magnificent Seven", "genre": "Action,Adventure,Western", "Description": "Seven gunmen in the old west gradually c
23 {"_id": "37", "_rev": "1-b6bca820661d3ca347e09978d79b4090", "title": "Interstellar", "genre": "Adventure,Drama,Sci-Fi", "Description": "A team of explorers travel through a wormhole in sp
24 {"_id": "16", "_rev": "1-2a9ceaf9e9c6b20a2d526007b38e530", "title": "The Secret Life of Pets", "genre": "Animation,Adventure,Comedy", "Description": "The quiet life of a terrier named Ma
25 {"_id": "46", "_rev": "1-faddb352379e27c9d8785416040b9013", "title": "Pirates of the Caribbean: On Stranger Tides", "genre": "Action,Adventure,Fantasy", "Description": "Jack Sparrow and
26 {"_id": "68", "_rev": "1-943c6c5d6283a969af552e73f721a84", "title": "Sully", "genre": "Biography,Drama", "Description": "The story of Chesley Sullenberger, an American pilot who became a
27 {"_id": "29", "_rev": "1-cfe662be49c49b29a93c483774e395", "title": "Bad Moms", "genre": "Comedy", "Description": "When three overworked and under-appreciated moms are pushed beyond thei
28 {"_id": "62", "_rev": "1-3cfb519a19893334c7e328801eaf9207", "title": "The Autopsy of Jane Doe", "genre": "Horror,Mystery,Thriller", "Description": "A father and son, both coroners, are pu
29 {"_id": "18", "_rev": "1-e918e810473551e1e875c124b49b8545", "title": "Jason Bourne", "genre": "Action,Thriller", "Description": "The CIA's most dangerous former operative is drawn out of
30 {"_id": "24", "_rev": "1-0c1b3c282f803db74ac966dc78d24cea", "title": "Trolls", "genre": "Animation,Adventure,Comedy", "Description": "After the Bergens invade Troll Village, Poppy, the ha
31 {"_id": "66", "_rev": "1-e07125530b9564f190511b279eb16402", "title": "Kingsman: The Secret Service", "genre": "Action,Adventure,Comedy", "Description": "A spy organization recruits an un
32 {"_id": "4", "_rev": "1-a20bb4dc3a57812543b3256ee026d84", "title": "Sing", "genre": "Animation,Comedy,Family", "Description": "In a city of humanoid animals, a hustling theater impresar
33 {"_id": "64", "_rev": "1-6f6aef7935f959d1682d26000ada440c", "title": "Fifty Shades of Grey", "genre": "Drama,Romance,Thriller", "Description": "Literature student Anastasia Steele's life
```

Nous avons utiliser Github pour la gestion des fichiers et permettre la simplicité du projet. Cela nous a permis d'assurer un suivi des modifications et de faciliter la gestion des différentes versions du projet.



Lien Github: <https://github.com/Habdelli99/nosql-python-app/tree/main>

# Difficultés rencontrés

Voici une reformulation plus détaillée de la section "Difficultés Rencontrées" :

## 1. Difficultés Rencontrées

Au cours du développement de notre projet, nous avons fait face à plusieurs défis techniques et méthodologiques qui ont demandé des solutions spécifiques pour garantir la bonne marche de l'application.

### 1.1. Connexion et Configuration

#### MongoDB Atlas :

L'un des premiers défis a été la mise en place de la connexion sécurisée à MongoDB Atlas. En effet, pour garantir la sécurité des données et des connexions, MongoDB exige que seules certaines adresses IP soient autorisées à se connecter à la base de données. Nous avons dû configurer cette liste d'IP autorisées dans le tableau de bord MongoDB Atlas, ce qui a pris du temps car il a fallu s'assurer que les adresses des serveurs et des environnements de développement étaient bien incluses.

#### Neo4j :

Un autre problème technique majeur concernait l'importation des données depuis MongoDB vers Neo4j. Nous avons choisi de transformer certaines données issues de MongoDB pour les adapter à la structure orientée graphes de Neo4j. Ce processus impliquait des manipulations complexes de formats de données, notamment des fichiers CSV. La gestion de ces fichiers et leur transformation pour correspondre aux exigences de Neo4j a nécessité un ajustement méthodologique et technique important.

### 1.2. Requêtes et Performances

#### Optimisation des requêtes MongoDB :

À mesure que la taille des ensembles de données augmentait, la performance des requêtes MongoDB devenait un problème de plus en plus critique. L'un des principaux défis était d'optimiser les requêtes pour qu'elles puissent traiter efficacement de grandes quantités de données sans provoquer de ralentissements. Nous avons dû réécrire certaines requêtes afin d'améliorer leur efficacité, en utilisant des opérateurs plus performants et en optimisant l'accès aux données.

### Indexation des champs :

Une autre solution pour améliorer les performances a été d'ajouter des index sur certains champs fréquemment utilisés dans nos requêtes MongoDB. Cela a permis de réduire le temps d'exécution des requêtes, en particulier celles impliquant des filtres ou des recherches par clé spécifique.

L'indexation a joué un rôle clé dans l'amélioration globale de la réactivité de l'application.

## 1.3. Visualisation

### Structuration des graphes Neo4j avec Neovis.js :

La visualisation des graphes Neo4j avec Neovis.js a présenté un défi, notamment pour structurer et optimiser les données afin qu'elles soient compréhensibles et interactives. La gestion des relations entre nœuds, la mise en forme des graphes et la performance en temps réel ont nécessité des ajustements pour éviter les lenteurs et garantir une interprétation correcte des données.

## 1.4. Gestion des erreurs et des exceptions

L'un des défis majeurs a été la gestion des erreurs lors des connexions et des opérations sur les bases de données MongoDB et Neo4j. En cas de connexion échouée ou de manipulation de données incorrectes, des erreurs peuvent survenir, ce qui nécessite une gestion adéquate des exceptions. Cela inclut la gestion des erreurs liées à l'authentification, à la perte de connexion ou à des données non valides dans les requêtes.

## 1.5. Problèmes de performance avec Streamlit

Streamlit est un excellent outil pour la création rapide d'interfaces utilisateur, mais lors du traitement de grandes quantités de données ou de visualisations complexes, l'application peut devenir lente. Nous avons rencontré des problèmes de performance lorsque les graphes ou les ensembles de données étaient trop volumineux. Il a fallu explorer des solutions d'optimisation, comme le découpage des données ou le rendu paresseux (lazy loading), pour garantir une interface fluide.

## 1.6. Synchronisation des bases de données

Un autre défi technique a été la synchronisation des données entre MongoDB et Neo4j. Les données stockées dans MongoDB devaient être adaptées et transférées sous une forme appropriée pour la base de données orientée graphes Neo4j. Cela impliquait de traiter des transformations de données, en assurant leur intégrité pendant le transfert, et en évitant toute perte d'information importante lors de la migration.

# Améliorations possibles

Ce projet a posé des bases solides pour l'exploitation de bases de données NoSQL en combinant MongoDB et Neo4j. Toutefois, plusieurs améliorations sont envisageables :

## Optimisation des performances :

Implémenter un caching des résultats de requêtes pour éviter des appels redondants à la base.  
Explorer des stratégies d'indexation avancées pour accélérer les recherches, notamment sur les relations dans Neo4j.

## Automatisation et scalabilité :

Mettre en place un pipeline d'ingestion automatisé pour synchroniser MongoDB et Neo4j en temps réel.  
Déployer l'application sur un service cloud comme AWS, Azure ou GCP pour assurer une scalabilité optimale.

## Amélioration de l'interface utilisateur :

Rendre la visualisation plus interactive avec Neovis.js en permettant des filtres dynamiques sur les graphes.  
Intégrer une fonctionnalité de recherche avancée pour explorer les bases en fonction de divers critères.

## Évolution fonctionnelle :

Ajouter une API REST ou GraphQL pour permettre une intégration plus facile avec d'autres applications.  
Explorer des modèles prédictifs (machine learning) pour proposer des recommandations basées sur l'analyse des données.

# Apprentisages

Ce projet a été une expérience enrichissante tant sur le plan technique que méthodologique.

## Sur le plan technique :

Maîtrise des bases de données documentaires (MongoDB) et graphes (Neo4j), avec leurs avantages et leurs limites.  
Optimisation des requêtes pour assurer une meilleure performance sur des ensembles de données volumineux.  
Utilisation de Streamlit pour créer une interface utilisateur simple et efficace.

## Sur le plan méthodologique :

Structuration d'un projet NoSQL en intégrant deux technologies complémentaires.  
Gestion des difficultés liées à la connexion aux bases de données dans le cloud et à la synchronisation des données.  
Travail sur la visualisation des graphes, qui nécessite une réflexion sur l'organisation des relations et leur interprétation.



The background features a complex, abstract design composed of numerous thin, light blue lines. These lines are arranged in a way that creates a sense of depth and movement, resembling a stylized, flowing ribbon or a series of overlapping, curved planes. The lines are most concentrated in the lower-left and upper-right areas, with some lines extending across the center of the page. The overall effect is a modern, artistic, and somewhat ethereal aesthetic.

# Questions



1) Afficher l'année où le plus grand nombre de films ont été sortis

```
> db.films.aggregate([
  { $group: { _id: "$year", total: { $sum: 1 } } },
  { $sort: { total: -1 } },
  { $limit: 1 }
])
< {
  _id: 2016,
  total: 73
}
```

2) Quel est le nombre de films sortis après l'année 1999 ?

```
> db.films.countDocuments({ year: { $gt: 1999 } })
< 99
```

3) Quelle est la moyenne des votes des films sortis en 2007 ?

```
> db.films.aggregate([
  { $match: { year: 2007 } },
  { $group: { _id: null, moyenne_votes: { $avg: "$Votes" } } }
])
< {
  _id: null,
  moyenne_votes: 192.5
}
```

4) Affichez un histogramme qui permet de visualiser le nombres de films par année

```
> db.films.aggregate([
  { $group: { _id: "$year", count: { $sum: 1 } } },
  { $sort: { _id: 1 } }
])
< {
  _id: null,
  count: 2
}
{
  _id: 1978,
  count: 1
}
{
  _id: 2006,
  count: 3
}
{
  _id: 2007,
  count: 2
}
{
  _id: 2008,
  count: 1
}
{
  _id: 2009,
  count: 2
}
```

```

{
  _id: 2010,
  count: 1
}
{
  _id: 2012,
  count: 2
}
{
  _id: 2013,
  count: 2
}
{
  _id: 2014,
  count: 5
}
{
  _id: 2015,
  count: 8
}
{
  _id: 2016,
  count: 73
}

```

5) Quelles sont les genres de films disponibles dans la bases ?

```

> db.films.aggregate([
  { $group: { _id: "$genre" } }
])
< {
  _id: 'Drama,Mystery,Sci-Fi'
}
{
  _id: 'Action,Adventure,Western'
}
{
  _id: 'Action,Thriller'
}
{
  _id: 'Horror'
}
{
  _id: 'Comedy,Drama,Music'
}
{
  _id: 'Adventure,Mystery,Sci-Fi'
}
{
  _id: 'Drama,Fantasy'
}
{
  _id: 'Biography,Comedy,Crime'
}
{

```

```

    _id: 'Biography,Drama'
  }
  {
    _id: 'Horror,Mystery,Thriller'
  }
  {
    _id: 'Drama,Romance,Thriller'
  }
  {
    _id: 'Action,Adventure,Biography'
  }
  {
    _id: null
  }
  {
    _id: 'Comedy,Crime,Drama'
  }
  {
    _id: 'Drama,History,Thriller'
  }
  {
    _id: 'Action,Drama,Romance'
  }
  {
    _id: 'Crime,Drama,Mystery'
  }
  {
    _id: 'Action,Crime,Drama'
  },
  {
    _id: 'Adventure,Drama,Fantasy'
  }
  {
    _id: 'Crime,Drama,Thriller'
  }

```

6) Quel est le film qui a généré le plus de revenu?

```

> db.films.find({ "Revenue (Millions)": { $ne: "" } })
.sort({ "Revenue (Millions)": -1 })
.limit(1)
< {
  _id: '51',
  _rev: '1-efdfbecc4f0bb1b6424fdea48ce766f9',
  title: 'Star Wars: Episode VII - The Force Awakens',
  genre: 'Action,Adventure,Fantasy',
  Description: 'Three decades after the defeat of the Galactic Empire, a new threat arises. The First Order attempts to rule the galaxy and only a ragtag
  Director: 'J.J. Abrams',
  Actors: 'Daisy R_idley, John Boyega, Oscar Isaac, Domhnall Gleeson',
  year: 2015,
  'Runtime (Minutes)': 136,
  rating: 'G',
  Votes: 661608,
  'Revenue (Millions)': 936.63,
  Metascore: 81
}

```

7) Quels sont les réalisateurs ayant réalisé plus de 5 films dans la base de données ?

Aucun réalisateur n'a réalisé plus de 5 films, le maximum étant 4 films (Christopher Nolan)

```
> db.films.aggregate([
  { $group: { _id: "$Director", total: { $sum: 1 } } },
  { $match: { total: { $gt: 5 } } },
  { $sort: { total: -1 } }
])
<
```

```
> db.films.aggregate([
  { $group: { _id: "$Director", total: { $sum: 1 } } },
  { $sort: { total: -1 } },
  { $limit: 10 }
])
< {
  _id: 'Christopher Nolan',
  total: 4
}
{
  _id: 'Martin Scorsese',
  total: 3
}
{
  _id: 'Peter Berg',
  total: 2
}
{
  _id: 'Quentin Tarantino',
  total: 2
}
{
  _id: 'Joss Whedon',
  total: 2
}
```

```
}
{
  _id: null,
  total: 2
}
{
  _id: 'Gore Verbinski',
  total: 2
}
{
  _id: 'Tate Taylor',
  total: 2
}
{
  _id: 'Denis Villeneuve',
  total: 2
}
{
  _id: 'Ben Young',
  total: 1
}
}
```

8) Quel est le genre de film qui rapporte en moyenne le plus de revenus ?

```
> db.films.aggregate([
  { $match: { "Revenue (Millions)": { $type: "number" } } },
  { $set: { genres: { $split: ["$genre", ","] } } },
  { $unwind: "$genres" },
  { $group: { _id: "$genres", avgRevenue: { $avg: "$Revenue (Millions)" } } },
  { $sort: { avgRevenue: -1 } },
  { $limit: 1 }
])
< {
  _id: 'Fantasy',
  avgRevenue: 265.8335714285714
}
```

9) Quels sont les 3 films les mieux notés (rating) pour chaque décennie (1990-1999, 2000-2009, etc.) ?

```
> db.films.aggregate([
  { $match: { Metascore: { $ne: "", $exists: true } } },
  { $addFields: {
    Metascore: { $toInt: "$Metascore" },
    decade: { $subtract: ["$year", { $mod: ["$year", 10] }] }
  }},
  { $sort: { decade: 1, Metascore: -1 } },
  {
    $group: {
      _id: "$decade",
      topFilms: { $push: { title: "$title", Metascore: "$Metascore" } }
    }
  },
  {
    $project: {
      topFilms: { $slice: ["$topFilms", 3] }
    }
  }
])
< {
  _id: 2000,
  topFilms: [
    {
      title: 'The Departed',
      Metascore: 85
    },
    {
      title: 'Avatar',
      Metascore: 83
    }
  ]
}
```

## 10) Quel est le film le plus long (Runtime) par genre ?

```
> db.films.aggregate([
  { $match: { "Runtime (Minutes)": { $type: "number" } } },
  { $set: { genres: { $split: ["$genre", ","] } } },
  { $unwind: "$genres" },
  {
    $group: {
      _id: "$genres",
      maxRuntime: { $max: "$Runtime (Minutes)" },
      film: { $first: "$title" }
    }
  },
  { $sort: { maxRuntime: -1 } }
])
< {
  _id: 'Crime',
  maxRuntime: 187,
  film: 'The Departed'
}
{
  _id: 'Drama',
  maxRuntime: 187,
  film: 'The Departed'
}
{
  _id: 'Mystery',
  maxRuntime: 187,
  film: 'Arrival'
}
```

```
  _id: 'Comedy',
  maxRuntime: 180,
  film: 'Why Him?'
}
{
  _id: 'Biography',
  maxRuntime: 180,
  film: 'Hidden Figures'
}
{
  _id: 'Action',
  maxRuntime: 169,
  film: 'Rogue One'
}
{
  _id: 'Sci-Fi',
  maxRuntime: 169,
  film: 'Rogue One'
}
{
  _id: 'Fantasy',
  maxRuntime: 169,
  film: 'Star Wars: Episode VII - The Force Awakens'
}
{
  _id: 'Adventure',
  maxRuntime: 169,
  film: 'Gold'
}
```

```

    _id: 'Romance',
    maxRuntime: 125,
    film: 'Passengers'
  }
  {
    _id: 'Horror',
    maxRuntime: 117,
    film: 'Resident Evil: The Final Chapter'
  }
  {
    _id: 'Animation',
    maxRuntime: 108,
    film: 'Moana'
  }

```

- 11) Créer une vue MongoDB affichant uniquement les films ayant une note supérieure à 80 (Metascore) et généré plus de 50 millions de dollars.

```

> db.createView(
  "films_high_rating_revenue",
  "films",
  [
    { $match: { Metascore: { $gt: 80 }, "Revenue (Millions)": { $gt: 50 } } },
    { $project: { title: 1, Metascore: 1, "Revenue (Millions)": 1 } }
  ]
)
< { ok: 1 }

```

- 12) Calculer la corrélation entre la durée des films (Runtime) et leur revenu (Revenue). (réaliser une analyse statistique.)

```

> db.films.find(
  { "Revenue (Millions)": { $type: "number" }, "Runtime (Minutes)": { $exists: true } },
  { "Runtime (Minutes)": 1, "Revenue (Millions)": 1, _id: 0 }
)
< {
  'Runtime (Minutes)': 151,
  'Revenue (Millions)': 132.37
}
{
  'Runtime (Minutes)': 120,
  'Revenue (Millions)': 7.22
}
{
  'Runtime (Minutes)': 111,
  'Revenue (Millions)': 60.31
}
{
  'Runtime (Minutes)': 116,
  'Revenue (Millions)': 100.01
}
{
  'Runtime (Minutes)': 133,
  'Revenue (Millions)': 532.17
}
{
  'Runtime (Minutes)': 116,
  'Revenue (Millions)': 10.64
}

```



```

    'Runtime (Minutes)': 137,
    'Revenue (Millions)': 47.7
  }
  {
    'Runtime (Minutes)': 121,
    'Revenue (Millions)': 333.13
  }
  {
    'Runtime (Minutes)': 136,
    'Revenue (Millions)': 936.63
  }
  {
    'Runtime (Minutes)': 127,
    'Revenue (Millions)': 169.27
  }
  {
    'Runtime (Minutes)': 144,
    'Revenue (Millions)': 155.33
  }
  {
    'Runtime (Minutes)': 147,
    'Revenue (Millions)': 408.08
  }
  {
    'Runtime (Minutes)': 139,
    'Revenue (Millions)': 67.12
  }
}

```

```

    'Runtime (Minutes)': 107,
    'Revenue (Millions)': 26.84
  }
  {
    'Runtime (Minutes)': 107,
    'Revenue (Millions)': 248.75
  }
  {
    'Runtime (Minutes)': 120,
    'Revenue (Millions)': 103.14
  }
  {
    'Runtime (Minutes)': 116,
    'Revenue (Millions)': 100.5
  }
  {
    'Runtime (Minutes)': 152,
    'Revenue (Millions)': 533.32
  }
  {
    'Runtime (Minutes)': 111,
    'Revenue (Millions)': 27.85
  }
  {
    'Runtime (Minutes)': 112,
    'Revenue (Millions)': 75.31
  }
}

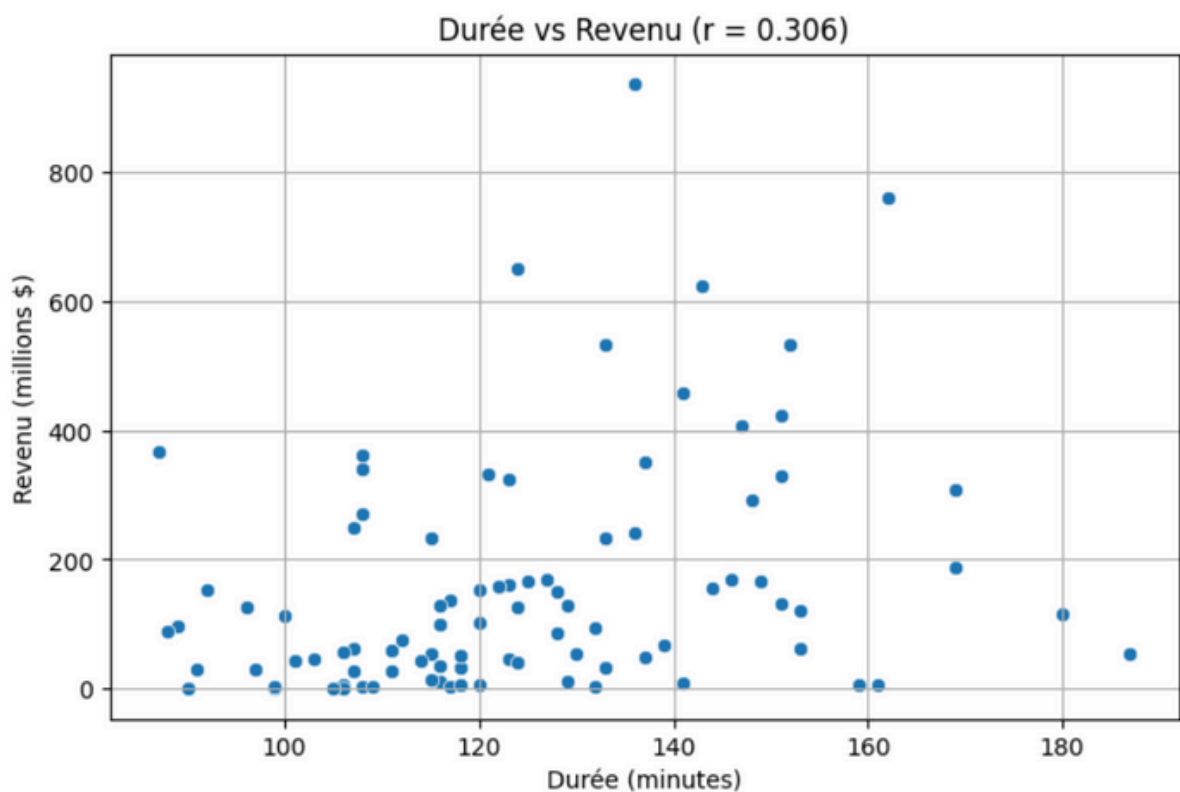
```

```

    title: 'The Dark Knight',
    Metascore: 82
  }
]
}
{
  _id: 1970,
  topFilms: [
    {
      title: 'Top Dog',
      Metascore: 67
    }
  ]
}
{
  _id: 2010,
  topFilms: [
    {
      title: 'Moonlight',
      Metascore: 99
    },
    {
      title: 'Manchester by the Sea',
      Metascore: 96
    },
    {
      title: 'La La Land',
      Metascore: 93
    }
  ]
}

```

Corrélation de Pearson : 0.306



- Coefficient de corrélation de Pearson (r) = 0.306
- Cela indique une corrélation positive faible à modérée entre la durée d'un film et son revenu.
- Autrement dit, plus un film est long, plus il a tendance à générer un revenu plus élevé, mais la relation n'est pas très forte

Cette analyse permet de dire que la durée peut influencer partiellement le succès commercial d'un film, mais d'autres facteurs doivent être pris en compte (qualité, acteurs, genre, marketing...).

### 13) Y a-t-il une évolution de la durée moyenne des films par décennie ?

Oui, la durée moyenne des films a évolué au fil des décennies, avec un pic dans les années 2000. Cela peut être lié à des changements dans les tendances du cinéma, les attentes du public ou les formats de diffusion.

Oui, il y a une évolution :

1970s : 116 min

2000s : 147.6 min (pic)

2010s : 120.1 min

```
> db.films.aggregate([
  { $match: { "Runtime (Minutes)": { $exists: true } } },
  { $addFields: { decade: { $subtract: ["$year", { $mod: ["$year", 10] } ] } } },
  { $group: {
    _id: "$decade",
    avgRuntime: { $avg: "$Runtime (Minutes)" }
  } },
  { $sort: { _id: 1 } }
])
< {
  _id: 1970,
  avgRuntime: 116
}
{
  _id: 2000,
  avgRuntime: 147.625
}
{
  _id: 2010,
  avgRuntime: 120.13186813186813
}
```

The background of the slide features a series of thin, light blue lines that flow and curve across the page, creating a sense of movement and data flow. These lines are most prominent on the left and right sides, framing the central text.

# Base de données Neo4j

## Création des nœuds films

**Definition** Constraints & Indexes (2)

Name

films.csv

☐ Filter file

Properties

Map from file

+

<input type="checkbox"/>	Name	Type	Column	ID
<input type="checkbox"/>	<b>_id</b>	string	_id	
<input type="checkbox"/>	<b>title</b>	string	title	
<input type="checkbox"/>	<b>year</b>	integer	year	
<input type="checkbox"/>	<b>Votes</b>	integer	Votes	
<input type="checkbox"/>	<b>Revenue</b>	float	Revenue	
<input type="checkbox"/>	<b>rating</b>	string	rating	
<input type="checkbox"/>	<b>director</b>	string	director	

## Création des nœuds actors

**Definition** Constraints & Indexes (2)

Label

Name

actors

File

Name

actors.csv

☐ Filter file

Properties

Map from file

+

<input type="checkbox"/>	Name	Type	Column	ID
<input type="checkbox"/>	<b>name</b>	string	name	

## Création des relations A\_jouer (acteurs → films)

Cette figure montre la création des relations A\_jouer à partir de deux fichiers CSV distincts :

**acted\_in.csv** : contient les relations issues de la base MongoDB  
(ex : Leonardo DiCaprio → The Departed)

**acted\_in\_projet.csv** : utilisé pour enrichir le graphe avec un acteur ajouté manuellement dans le cadre du projet (ex : Hamza → The Departed)



Relationship type ⓘ ↔ Reverse direction

Name

A\_jouer

File ⓘ

Name

acted\_in.csv ▼

☐ Filter file

Node ID mapping

	Node	ID	ID column
From	actors	name	actor ▼
To	films	title	film ▼

**Relationship type** ⓘ ↔ Reverse direction

Name

A\_jouer

---

**File** ⓘ

Name

acted\_in\_projet.csv ▼

☐ Filter file

---

**Node ID mapping**

	Node	ID	ID column
From	actors	name	actor ▼
To	films	title	film ▼

Création des nœuds genres + relations A\_pour\_genre (films → genres)

**Definition** Constraints & Indexes (2)

---

**Label** ⓘ

Name

genres

---

**File** ⓘ

Name

Genres.csv ▼

☐ Filter file

---

**Properties** Map from file +

<input type="checkbox"/>	Name	Type	Column	ID
<input type="checkbox"/>	name <span>✎</span>	string ▼	name ▼	<span>🔑</span>



Relationship type ⓘ

Reverse direction

Name

A\_pour\_genre

File ⓘ

Name

Film-Genre\_Relations.csv

☐ Filter file

Node ID mapping

	Node	ID	ID column
From	films	title	film
To	genres	name	genre

## Création des nœuds directors (réalisateurs)

Label ⓘ

Name

directors

File ⓘ

Name

directors.csv

☐ Filter file

Properties

Map from file +

<input type="checkbox"/>	Name	Type	Column	ID
<input type="checkbox"/>	name	string	name	

14) Quel est l'acteur ayant joué dans le plus grand nombre de films?

```
1 MATCH (a:actors)-[:A_jouer]->(f:films)
2 RETURN a.name AS acteur, COUNT(f) AS films_joués
3 ORDER BY films_joués DESC LIMIT 1;
4
```

acteur	films_joués
1 "Matthew McConaughey"	4

15) Quels sont les acteurs ayant joué dans des films où l'actrice Anne Hathaway a également joué ?

```
1 MATCH (anne:actors {name:'Anne Hathaway'})-[:A_jouer]->(f:films)->[:A_jouer]-(autre:actors)
2 RETURN DISTINCT autre.name AS acteurs_avec_Anne;
3
```

acteurs_avec_Anne
1 "Jessica Chastain"
2 "Mackenzie Foy"
3 "Matthew McConaughey"
4 "Austin Stowell"
5 "Jason Sudeikis"
6 "Tim Blake Nelson"

16) Quel est l'acteur ayant joué dans des films totalisant le plus de revenus ?

```
1 MATCH (a:actors)-[:A_jouer]->(f:films)
2 RETURN a.name AS acteur, SUM(f.Revenue) AS revenu_total
3 ORDER BY revenu_total DESC LIMIT 1;
4
```

acteur	revenu_total
1 "Chris Evans"	1490.35

17) Quelle est la moyenne des votes ?

```
1 MATCH (f:films)
2 RETURN avg(f.Votes) AS moyenne_votes;
3
```

Table RAW

	moyenne_votes
1	259963.74000000002

18) Quel est le genre le plus représenté dans la base de données ?

```
1 MATCH (f:films)-[:A_pour_genre]->(g:genres)
2 RETURN g.name AS genre, COUNT(*) AS total
3 ORDER BY total DESC LIMIT 1;
4 |
```

Table RAW

	genre	total
1	"Drama"	49

19) Films dans lesquels les acteurs ayant joué avec vous ont également joué ?

```
1 MATCH (:actors {name:"Hamza"})-[:A_jouer]->(:films)-[:A_jouer]-(coActors)-[:A_jouer]->(f:films)
2 RETURN DISTINCT f.title AS films_recommandés;
3 |
```

Table RAW

	films_recommandé
1	"Inception"
2	"The Wolf of Wall Street"
3	"Deepwater Horizon"
4	"Patriots Day"
5	"Jason Bourne"
6	"The Great Wall"

20) Quel réalisateur Director a travaillé avec le plus grand nombre d'acteurs distincts ?

```

1 MATCH (d:directors)-[:A_realise]->(f:films)-[:A_jouer]-(a:actors)
2 RETURN d.name AS réalisateur, COUNT(DISTINCT a) AS acteurs_uniques
3 ORDER BY acteurs_uniques DESC LIMIT 1;
4 |

```

	réalisateur	acteurs_uniques
1	"Christopher Nolan"	14

21) Quels sont les films les plus "connectés", c'est-à-dire ceux qui ont le plus d'acteurs en commun avec d'autres films ?

```

1 MATCH (f1:films)-[:A_jouer]-(a:actors)-[:A_jouer]-(f2:films)
2 WHERE id(f1) < id(f2)
3 RETURN f1.title AS film1, f2.title AS film2, COUNT(a) AS acteurs_communs
4 ORDER BY acteurs_communs DESC LIMIT 3;
5 |

```

	film1	film2	acteurs_communs
1	"Captain America a: Civil War"	"The Avengers"	3
2	"Pirates of the Caribbean: Dead Man's Chest"	"Pirates of the Caribbean: At W orld's End"	3
3	"The Dark Knigh t"	"The Prestige"	2

22) Trouver les 5 acteurs ayant joué avec le plus de réalisateurs différents.

```

1 MATCH (a:actors)-[:A_jouer]-(f:films)-[:A_realise]-(d:directors)
2 RETURN a.name AS acteur, COUNT(DISTINCT d) AS nb_realisateurs
3 ORDER BY nb_realisateurs DESC LIMIT 5;
4 |

```

	acteur	nb_realisateurs
1	"Scarlett Johansson"	4
2	"Ben Affleck"	4
3	"Matthew McConaughey"	4
4	"Chris Pratt"	4
5	"Bryce Dallas Howard"	3

23) Recommander un film à un acteur en fonction des genres des films où il a déjà joué (exemple : Leonardo DiCaprio) :

```

1 // Étape 1 : Récupérer les genres des films de l'acteur
2 MATCH (:actors {name: "Leonardo DiCaprio"})-[:A_jouer]->(f:films)-[:A_pour_genre]->(g:genres)
3 // Étape 2 : Recommander des films avec ces genres
4 MATCH (g)-[:A_pour_genre]->(filmReco:films)
5 // Étape 3 : Exclure les films qu'il a déjà joués
6 WHERE NOT EXISTS {
7   MATCH (:actors {name: "Leonardo DiCaprio"})-[:A_jouer]->(filmReco)
8 }
9
10 RETURN DISTINCT filmReco.title AS recommandation
11 LIMIT 5;
12

```

recommandation
1 "The Dark Knight"
2 "The Girl on the Train"
3 "Furious Seven"
4 "John Wick"
5 "The Hateful Eight"

24) Créer une relation INFLUENCE\_PAR entre réalisateurs en se basant sur des similarités dans les genres de films qu'ils ont réalisés.

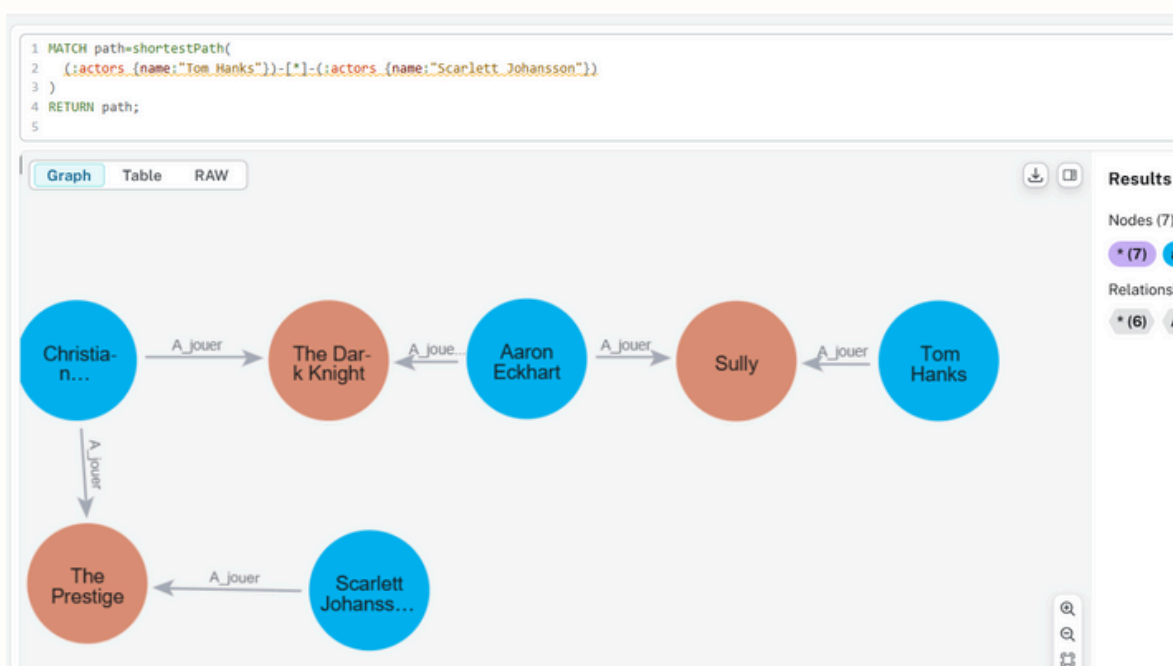
```

1 MATCH (d1:directors)-[:A_realise]->(f1:films)-[:A_pour_genre]->(g:genres),
2       (d2:directors)-[:A_realise]->(f2:films)-[:A_pour_genre]->(g)
3 WHERE d1 <> d2
4 MERGE (d1)-[:INFLUENCE_PAR]->(d2);
5

```

✓ Created 4358 relationships

25) Quel est le chemin le plus court entre deux acteurs ? (ex : Tom Hanks et Scarlett Johansson)



26) Analyser les communautés d'acteurs : Quels sont les groupes d'acteurs qui ont tendance à travailler ensemble ? (Utilisation d'algorithmes de détection de communauté comme Louvain.)

Nous avons utilisé l'algorithme Louvain de détection de communautés proposé par le module GDS de Neo4j. Identifier les groupes d'acteurs qui ont tendance à travailler ensemble en utilisant l'algorithme de détection de communauté Louvain proposé par la librairie GDS (Graph Data Science) de Neo4j. En projetant un graphe basé sur les relations `A_jouer` entre les nœuds `actors`, nous avons identifié plusieurs communautés d'acteurs ayant tendance à jouer ensemble.

```
1 CALL gds.graph.project(
2   'graph_actors',
3   'actors',
4   {
5     A_jouer: {
6       orientation: 'UNDIRECTED'
7     }
8   }
9 );
10
11 CALL gds.louvain.stream('graph_actors')
12 YIELD nodeId, communityId
13 RETURN gds.util.asNode(nodeId).name AS acteur, communityId
14 ORDER BY communityId;
15
```

```
CALL gds.graph.project(
  'graph_actors',
  'actors',
  {
    A_jouer: {
      orientation: 'UNDIRECTED'
    }
  }
);
```

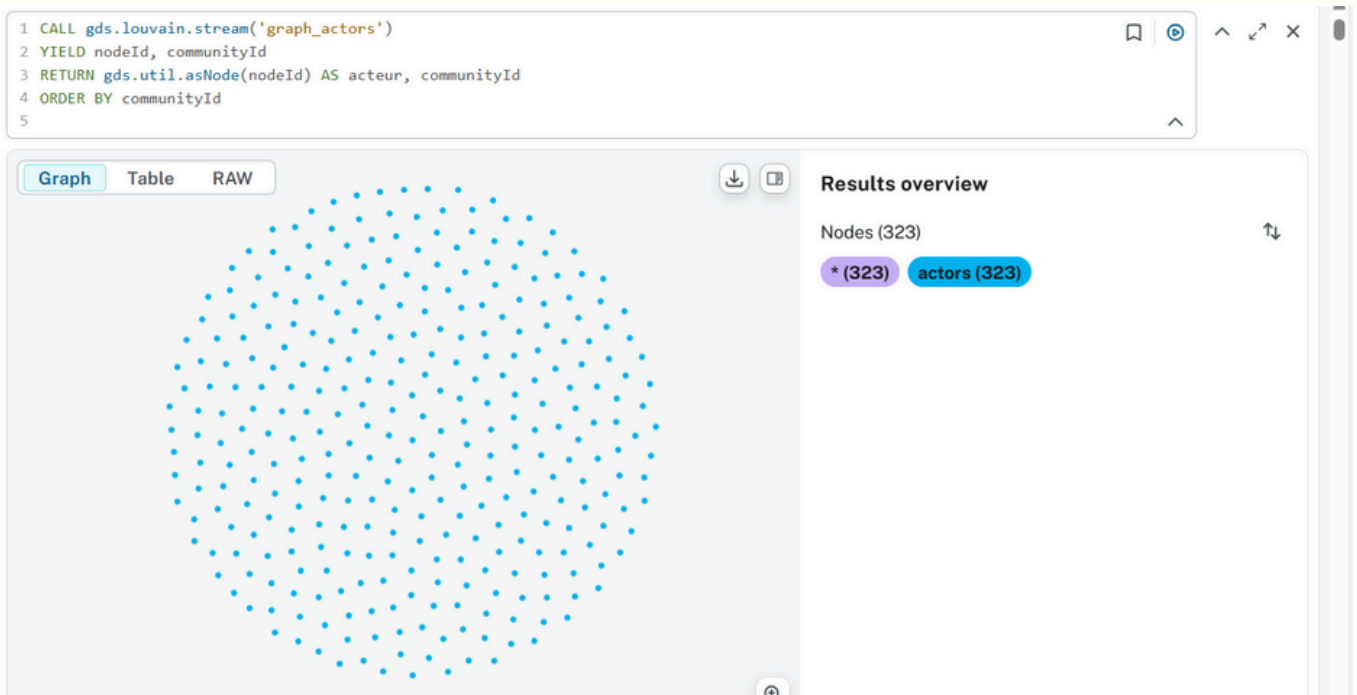
```
CALL gds.louvain.stream('graph_actors')
YIELD nodeId, communityId
RETURN gds.util.asNode(nodeId).name AS acteur, communityId
ORDER BY communityId;
```

No changes. Completed after 1009 ms

1 CALL gds.louvain.stream('graph\_actors')  
2 YIELD nodeId, communityId  
3 RETURN gds.util.asNode(nodeId) AS acteur, communityId  
4 ORDER BY communityId  
5 |

Graph Table RAW

	acteur	communityId
1	(:actors {name: "Aaron Eckhart"})	0
2	(:actors {name: "Aaron Poole"})	1
3	(:actors {name: "Aaron Taylor-Johnson"})	2
4	(:actors {name: "Adam Driver"})	3
5	(:actors {name: "Addison Timlin"})	4
6	(:actors {name: "Adèle Exarchopoulos"})	5



Plusieurs groupes d'acteurs ont été identifiés comme appartenant à la même communauté (par exemple : ceux qui ont souvent joué ensemble dans les mêmes films).

Ce travail peut servir à :

- Faire de la recommandation de casting
- Étudier les réseaux sociaux dans le cinéma
- Identifier les groupes d'acteurs influents

27) Quels sont les films qui ont des genres en commun mais qui ont des réalisateurs différents ?

```

1 MATCH (f1:films)-[:A_pour_genre]->(g:genres)-[:A_pour_genre]-(f2:films),
2       (f1)-[:A_realise]-(r1:directors),
3       (f2)-[:A_realise]-(r2:directors)
4 WHERE f1 <> f2 AND r1 <> r2
5 RETURN DISTINCT f1.title AS film1, f2.title AS film2, g.name AS genre, r1.name AS réalisateur1, r2.name AS
6       réalisateur2
7 LIMIT 10;

```

	film1	film2	genre	réalisateur1	réalisateur2
1	"The Dark Knight"	"The Departed"	"Crime"	"Christopher Nolan"	"Martin Scorsese"
2	"The Girl on the Train"	"The Departed"	"Crime"	"Tate Taylor"	"Martin Scorsese"
3	"Furious Seven"	"The Departed"	"Crime"	"James Wan"	"Martin Scorsese"
4	"John Wick"	"The Departed"	"Crime"	"Chad Stahelski"	"Martin Scorsese"
5	"The Hateful Eight"	"The Departed"	"Crime"	"Quentin Tarantino"	"Martin Scorsese"



			1"	e"	
5	"The Hateful Eight"	"The Departed"	"Crime"	"Quentin Tarantino"	"Martin Scorsese"
6	"Gone Girl"	"The Departed"	"Crime"	"David Fincher"	"Martin Scorsese"
7	"Don't Breathe"	"The Departed"	"Crime"	"Fede Alvarez"	"Martin Scorsese"
8	"Live by Night"	"The Departed"	"Crime"	"Ben Affleck"	"Martin Scorsese"
9	"The Accountant"	"The Departed"	"Crime"	"Gavin O'Connor"	"Martin Scorsese"
10	"Hounds of Love"	"The Departed"	"Crime"	"Ben Young"	"Martin Scorsese"

## 28) Recommander des films selon les préférences d'un acteur donné

1	// Étape 1 : genres préférés de l'acteur
2	MATCH (:actors {name:"Leonardo DiCaprio"})-[:A_jouer]->(f:films)-[:A_pour_genre]->(g:genres)
3	// Étape 2 : rechercher d'autres films dans ces genres
4	MATCH (g)-[:A_pour_genre]-(reco:films)
5	// Étape 3 : exclure les films déjà joués
6	WHERE NOT EXISTS {
7	MATCH (:actors {name:"Leonardo DiCaprio"})-[:A_jouer]->(reco)
8	}
9	RETURN DISTINCT reco.title AS recommandation
10	LIMIT 5;
11	

Table

RAW

recommandation	
1	"The Dark Knight"
2	"The Girl on the Train"
3	"Furious Seven"
4	"John Wick"
5	"The Hateful Eight"

## 29) Créer une relation de concurrence entre réalisateurs ayant réalisé des films similaires la même année.

1	MATCH (d1:directors)-[:A_realise]->(f1:films)-[:A_pour_genre]->(g:genres),
2	(d2:directors)-[:A_realise]->(f2:films)-[:A_pour_genre]->(g)
3	WHERE d1 <> d2 AND f1.year = f2.year
4	MERGE (d1)-[:CONCURRENCE]->(d2);
5	

✓

Created 2752 relationships

30) Identifier les collaborations les plus fréquentes entre réalisateurs et acteurs, puis analyser si ces collaborations sont associées à un succès commercial ou critique

### Trouver les collaborations fréquentes

```

1 MATCH (a:actors)-[:A_jouer]->(f:films)-[:A_realise]->(d:directors)
2 RETURN a.name AS acteur, d.name AS realisateur, COUNT(*) AS nb_collaborations
3 ORDER BY nb_collaborations DESC
4 LIMIT 10;
5

```

acteur	realisateur	nb_collaborations
1 "Keira Knightley"	"Gore Verbinski"	2
2 "Robert Downey Jr."	"Joss Whedon"	2
3 "Michael Caine"	"Christopher Nolan"	2
4 "Johnny Depp"	"Gore Verbinski"	2
5 "Orlando Bloom"	"Gore Verbinski"	2
6 "Chris Evans"	"Joss Whedon"	2
7 "Leonardo DiCaprio"	"Martin Scorsese"	2
8 "Christian Bale"	"Christopher Nolan"	2
9 "Mark Wahlberg"	"Peter Berg"	2
10 "Adam Driver"	"Martin Scorsese"	1

### Mesurer le succès moyen (Revenu ou Votes)

```

1 MATCH (a:actors)-[:A_jouer]->(f:films)-[:A_realise]->(d:directors)
2 WHERE f.Revenue IS NOT NULL
3 RETURN a.name AS acteur, d.name AS realisateur,
4 COUNT(*) AS nb_collaborations,
5 AVG(TOFLOAT(f.Revenue)) AS revenu_moyen
6 ORDER BY nb_collaborations DESC, revenu_moyen DESC
7 LIMIT 10;
8

```

acteur	realisateur	nb_collaborations	revenu_moyen
1 "Robert Downey Jr."	"Joss Whedon"	2	541.135
2 "Chris Evans"	"Joss Whedon"	2	541.135
3 "Keira Knightley"	"Gore Verbinski"	2	366.215
4 "Orlando Bloom"	"Gore Verbinski"	2	366.215
5 "Johnny Depp"	"Gore Verbinski"	2	366.215
6 "Michael Caine"	"Christopher Nolan"	2	293.2000000000005
7 "Christian Bale"	"Christopher Nolan"	2	293.2000000000005
8 "Leonardo DiCaprio"	"Martin Scorsese"	2	124.62
9 "Mark Wahlberg"	"Peter Berg"	2	46.57
10 "John Boyega"	"J.J. Abrams"	1	936.63

# Conclusion

Ce projet NoSQL nous a permis d'explorer et de manipuler deux bases de données NoSQL complémentaires : MongoDB, pour une gestion flexible des documents, et Neo4j, pour la modélisation des relations sous forme de graphes. Nous avons conçu une application en Python intégrant ces technologies, avec une interface en Streamlit permettant d'interagir avec les données. Tout au long du projet, nous avons rencontré divers défis, notamment liés à la configuration des bases de données, à l'optimisation des requêtes et à la visualisation des données.

Nous avons surmonté ces obstacles en mettant en place des stratégies telles que l'indexation des données, l'amélioration de la structuration des requêtes et l'optimisation des performances de l'interface. En plus des aspects techniques, ce projet nous a apporté une approche méthodologique solide, nous obligeant à structurer nos données de manière optimale et à garantir leur cohérence entre MongoDB et Neo4j. Il a également renforcé nos compétences en manipulation de données, en conception d'API et en développement d'interfaces interactives.

Enfin, cette expérience ouvre des perspectives d'amélioration et d'extension de notre application, notamment en intégrant des fonctionnalités plus avancées de visualisation et d'analyse. La scalabilité et l'automatisation pourraient également être améliorées pour rendre l'application plus performante et adaptable à des volumes de données plus importants.

Ainsi, ce projet a permis d'acquérir une compréhension approfondie des bases NoSQL et de développer des compétences pratiques en gestion et manipulation de données complexes. Les améliorations envisagées pourraient transformer cette application en un véritable outil analytique avancé.