Labo 1
Couchbase

1 Introduction

1.1 Objectif

L'objectif de ce laboratoire est d'exercer *Couchbase* et son langage de requête N1QL.

1.2 Organization

Ce laboratoire est à effectuer **par groupe de 2** étudiants. Tout plagiat sera sanctionné par la note de 1.

Ce laboratoire est à rendre pour le **06.10.2024** à 23h59 avec un commit sur Github Classroom.

1.3 Mise en place

Le code source du labo vous est fourni sur Github Classroom. Celui-ci contient :

- Un fichier `pom.xml` qui configure les dépendances du projet Maven.
- Un fichier `Main.java` qui se connecte à la base de donnée Couchbase et exécute les requêtes de `Requests.java`.
- Un fichier `QueryOutputFormatTest` qui permet de vérifier que vous avez le bon format d'output pour les requêtes. (Lancer les tests avant le rendu pour simplifier les tests automatiques de la correction)

Ainsi que 2 fichiers à compléter :

- Un fichier `Requests.java`.
- Un fichier `Indices.java` à compléter si un nouvel index est nécessaire pour l'exécution d'une requête.

Utiliser l'instance de Couchbase et le dataset que vous avez installée avec le document "Installation Couchbase".

2 Connexion

Vérifiez que vous pouvez vous connecter en exécutant la classe `Main`. Les noms des collections de `mflix-sample` devraient s'afficher dans votre console. Si besoin modifier la méthode `openConnection`.

3 Requêtes

Dans les requêtes utiliser le nom original des champs. Renommer uniquement lorsque c'est indiqué entre parenthèse.

⋆ 1. Implémenter la méthode `inconsistentRating` pour retourner les films (`imdb_id`, `tomatoes_rating` `imdb_rating`) dont le rating tomatoes n'est pas zéro et dont l'écart entre le rating imdb et tomatoes est plus que 7. Utiliser l'attribut `tomatoes.viewer.rating` comme le rating tomatoes.

⋆ 2. Implementer la méthode `hiddenGem` pour retourner le titre des films parfaitement notés (10) par les critiques tomatoes mais qui n'ont pas été notés par les viewers tomatoes.

⋆ 3. Implémenter la méthode `topReviewers` pour retourner l'email des 10 personnes ayant fait le plus de commentaires ainsi que le nombre de leurs commentaires (`cnt`).

⋆ 4. Implémenter la méthode `greatReviewers` pour retourner l'email des personnes ayant fait plus de 300 commentaires. Le résultat de la requête doit être une liste de String.

⋆⋆ 5. Implémenter la méthode `bestMoviesOfActor` pour retourner l'identifiant IMDB (`imdb_id`), le rating imdb et le casting des films où le rating IMDB est un numéro plus grand que 8 et dans lequel un acteur donné (par exemple "Ralph Fiennes") joue.

⋆⋆ 6. Implémenter la méthode `plentifulDirectors` pour retourner le nom et le nombre de films des directeurs
(`director_name`, `count_film`) ayant dirigé plus que 30 films.

⋆⋆ 7. Implémenter la méthode `confusingMovies` pour retourner l'id (`movie_id`) et le titre des films qui ont plus de 20 directeurs :

⋆⋆ 8. Implémenter les méthodes `commentsOfDirector1` et `commentsOfDirector2` qui implémentent 2 manières différentes (JOIN et sous-requête) pour retourner le text (doit être de type `string` et non pas de type `array`) de tous les commentaires sur tous les films (`movie_id`) dirigés par un directeur donné (par exemple "Woody Allen").

⋆⋆⋆ 9. Implémenter la méthode `removeEarlyProjection` qui permet de mettre à jour la collection afin de supprimer les projections d'un film donné avant 18h. La méthode retourne le nombre de mise à jour effectuée.

⋆⋆⋆ 10. (Bonus) Implémenter la méthode `nightMovies` pour retourner les films qui sont projetés uniquement à partir de 18h.