Université Polytechnique HAUTS-DE-FRANCE

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE

Base de données

TP1, TP2, TP3, TP4

Elouan SAGNARD
ICY 4A - 19/03/2025

Sommaire :

0.Questions préliminaires

1) Cassandra

Introduction à Cassandra

1. Quelle entreprise a développé Cassandra à l'origine et pour quel usage ?

   Cassandra a été développée à l'origine par Facebook en 2008 pour gérer l'indexation du moteur de recherche de la boîte de messagerie. Elle a ensuite été open-sourcée et est maintenant gérée par la fondation Apache.

2. Pourquoi Cassandra est-elle qualifiée de base de données NoSQL distribuée ?

   Cassandra est une base de données NoSQL distribuée car :

   - Elle ne repose pas sur un schéma relationnel rigide (comme les bases SQL traditionnelles).
   - Elle est conçue pour fonctionner sur plusieurs nœuds sans point unique de défaillance.
   - Elle offre une haute disponibilité et un partitionnement horizontal des données sur plusieurs serveurs.

3. Quelle est la principale différence entre le modèle BASE (Basically Available, Soft-state, Eventually consistent) utilisé par Cassandra et le modèle ACID des bases relationnelles ?

   - Le modèle ACID (Atomicité, Cohérence, Isolation, Durabilité) garantit la fiabilité des transactions en assurant une forte cohérence des données.
   - Le modèle BASE (Basically Available, Soft-state, Eventually consistent) repose sur une approche plus flexible, où les données sont hautement disponibles et peuvent être temporairement inconsistantes, mais finissent par converger vers un état cohérent. Cassandra suit BASE pour privilégier la scalabilité et la disponibilité.

Architecture et Réplication

4. Quelle est la différence entre SimpleStrategy et NetworkTopologyStrategy en termes de réplication des données ?

   - SimpleStrategy : Utilisée pour des déploiements sur un seul data center, elle réplique les données de manière séquentielle sur les nœuds voisins.

- NetworkTopologyStrategy : Adaptée aux environnements multi-datacenters, elle permet de définir un nombre de répliques spécifique par data center, assurant une meilleure tolérance aux pannes et une faible latence des requêtes.

5. Expliquez en quelques mots le rôle du Gossip Protocol dans Cassandra.

   Le Gossip Protocol est un protocole de communication pair-à-pair utilisé par Cassandra pour :

   - Assurer la découverte et la synchronisation des états entre les nœuds du cluster.
   - Détecter les pannes et redistribuer les responsabilités des nœuds défaillants.
   - Maintenir une cohérence décentralisée sans point unique de coordination.

6. Comment Cassandra assure-t-elle une tolérance aux pannes sans point unique de défaillance ?

   - Par réplication des données sur plusieurs nœuds à l'aide de stratégies comme NetworkTopologyStrategy.
   - Grâce à son architecture sans maître, où tous les nœuds sont égaux (peer-to-peer).
   - Par l'utilisation du Gossip Protocol pour détecter les défaillances et réassigner les responsabilités.
   - En appliquant le partitionnement des données (sharding) pour distribuer la charge de travail.

Modélisation des Données

7. Pourquoi Cassandra ne supporte-t-elle pas les jointures et comment peut-on contourner cette limitation ?

   - Cassandra ne supporte pas les jointures car elles nécessitent des opérations complexes et coûteuses en environnement distribué.
   - Pour contourner cette limitation, on adopte une modélisation dénormalisée
     - En dupliquant les données pour éviter les jointures.
     - En organisant les tables selon les requêtes les plus fréquentes (query-driven modeling).
     - En utilisant des indexes secondaires ou des tables d'agrégation pré-calculées.

2) DynamoDB

Introduction à DynamoDB

1. Qu'est-ce que DynamoDB et par quelle entreprise a-t-il été développé ?

   DynamoDB est une base de données NoSQL entièrement gérée, développée par Amazon Web Services (AWS). Elle a été conçue pour fournir une haute disponibilité, une faible latence et une scalabilité automatique pour les applications cloud modernes.
2. Pourquoi DynamoDB est-il particulièrement adapté aux applications nécessitant une forte scalabilité ?

   - Scalabilité automatique : DynamoDB ajuste automatiquement les ressources en fonction de la charge de travail.
   - Modèle de données distribué : Les données sont stockées de manière répartie sur plusieurs serveurs pour supporter des millions de requêtes par seconde.
   - Sans serveur (serverless) : Aucune gestion d'infrastructure, ce qui permet une montée en charge facile.
3. Quelle est la principale différence entre DynamoDB et une base de données relationnelle classique ?

DynamoDB (NoSQL)	**Base Relationnelle (SQL)**
Clés-valeurs et colonnes stockées dans des tables sans schéma fixe	Structurée avec un schéma rigide (tables, colonnes, relations)
Scalabilité horizontale (ajout de nœuds)	Scalabilité verticale (augmentation des ressources d'un seul serveur)
Pas de support des jointures complexes	Supporte les jointures et transactions ACID
Conçu pour haute disponibilité et performance	Optimisé pour cohérence stricte et relations complexes

Architecture et Scalabilité

4. Comment DynamoDB assure-t-il une haute disponibilité et une tolérance aux pannes ?

    - Réplication multi-AZ : Les données sont répliquées automatiquement sur trois zones de disponibilité (AZ) dans une région AWS.

    - Auto-scaling : DynamoDB ajuste automatiquement la capacité en fonction de la charge.

    - Mécanisme de partitionnement : DynamoDB répartit les données sur plusieurs partitions pour éviter les goulots d'étranglement.
5. Quelle est la différence entre le mode de capacité provisionnée et le mode On-Demand ?

    - Mode provisionné :
        - L'utilisateur définit une capacité de lecture/écriture (RCU/WCU).
        - Moins cher si la charge est prévisible.

    - Mode On-Demand :
        - Facturation basée sur l'utilisation réelle.
        - Idéal pour des charges fluctuantes ou imprévisibles.

6. Quel mécanisme DynamoDB utilise-t-il pour gérer la cohérence des données ? (forte cohérence vs cohérence éventuelle)

    - Lecture avec cohérence éventuelle (par défaut) : Les données peuvent prendre un certain temps avant d'être propagées sur toutes les copies.

    - Lecture avec forte cohérence (optionnelle) : Retourne toujours la version la plus récente des données mais avec un coût plus élevé en termes de latence et de disponibilité.

Sécurité et Gestion des Accès

7. Quels sont les principaux moyens de sécuriser l'accès aux données dans DynamoDB ?

    - AWS IAM (Identity & Access Management) : Contrôle d'accès avec des politiques IAM.

    - VPC Endpoints : Accès sécurisé via un VPC privé sans passer par Internet.

    - AWS WAF (Web Application Firewall) : Protection contre les attaques courantes (SQL Injection, DoS, etc.).

8. Comment le chiffrement des données est-il géré dans DynamoDB ?

    - Chiffrement au repos : Activé par défaut via AWS KMS (Key Management Service).
    - Chiffrement en transit : Utilisation de TLS (Transport Layer Security) pour sécuriser les communications.

Fonctionnalités avancées

9. À quoi sert la fonctionnalité TTL (Time To Live) ?

    - Permet de supprimer automatiquement les éléments expirés après un certain temps.
    - Idéal pour gérer les sessions utilisateurs, logs temporaires et caches.

10. En quoi DynamoDB Streams peut-il être utile dans une architecture AWS ?

    - Capture les modifications en temps réel des tables DynamoDB.
    - Permet des intégrations avec AWS Lambda pour déclencher des traitements automatiques (ex : mises à jour de caches, synchronisation avec d'autres bases).
    - Utilisé pour des architectures événementielles et du CDC (Change Data Capture).

11. Quelles sont les principales limitations de DynamoDB et comment peut-on les contourner ?

Limitation	Solution
Pas de jointures complexes	Dénormalisation des données (modélisation basée sur les accès)
Taille d'un élément limitée à 400 KB	Stocker les fichiers volumineux dans Amazon S3 et garder les références dans DynamoDB
Pas de transactions ACID complètes	Utiliser Transactions DynamoDB pour des mises à jour atomiques sur plusieurs éléments

	Facturation basée sur la consommation	Optimiser les accès et utiliser DAX (DynamoDB Accelerator) pour réduire les coûts

I.NoSQL Orienté Document avec MongoDB

Etape 1 :

```
#Etape 1
import matplotlib.pyplot as plt
import pymongo as pm
import pandas as pd
import numpy as np
import json

pd.reset_option('display.max_columns')
```

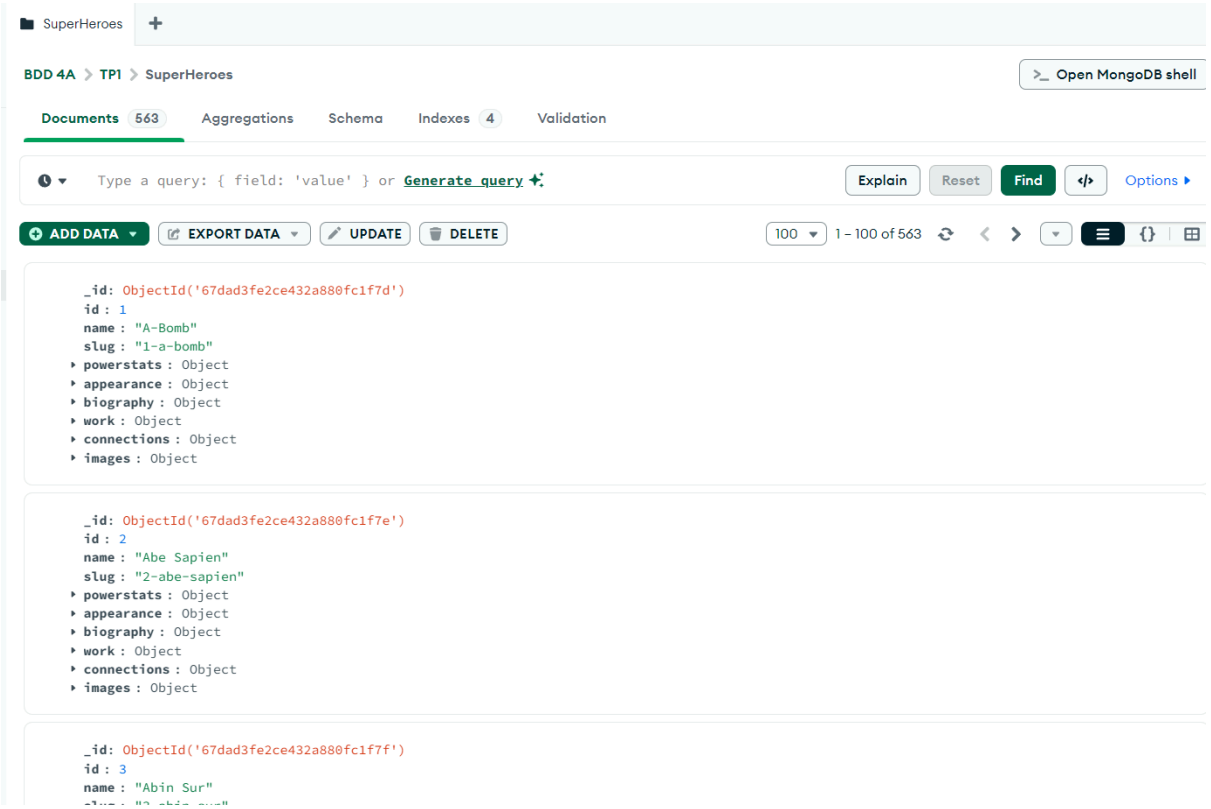
Etape 2 :

```
#Etape 2
client = pm.MongoClient("mongodb://localhost:27017/")
db = client["TP1"]  # Nom de la base de données
collection = db["SuperHeroes"]  # Nom de la collection

print("Connexion réussie à MongoDB !")
```

Etape 3 :

```
#Etape 3
with open('SuperHerosComplet.json', 'r', encoding='utf-8') as file:
    data = json.load(file)  # Charger le fichier JSON

result = collection.insert_many(data)
print('Inserted {} documents'.format(len(result.inserted_ids)))

print(collection.find_one())
```

On retrouve bien nos super héros dans MongoDB Compass



Etape 4 :

```python
#Etape 4
def index():
    # Création des index
    collection.create_index([("name", 1)])  # Index sur name
    collection.create_index([("powerstats.intelligence", 1)])  # Index sur intelligence
    collection.create_index([("biography.publisher", 1)])  # Index sur publisher
```

Etape 5 :

```python
#Etape 5
def extract_data():
    data = list(collection.find({}, {"_id": 0}))  # Récupération des données sans le champ _id
    df = pd.DataFrame(data)
    print("✅ Données extraites avec succès !")
    return df
```

Etape 6 :

```python
#Etape 6

def clean_nested_data(value):
    if isinstance(value, dict):
        return {key: clean_nested_data(val) for key, val in value.items()}
    elif isinstance(value, list):
        return [clean_nested_data(val) for val in value]
    elif value in [None, "-", np.nan]:
        return "Unknown"
    else:
        return value
```

```python
def clean_data(df):
    """ Étape 6 : Nettoyage et préparation des données """

    # Vérifier les valeurs manquantes et les remplacer par "inconnue"

    df = df.map(clean_nested_data)


    # Normalisation des formats (conversion hauteur en cm et poids en kg)
    # La fonction ajoute 2 nouvelles informations à chaque héros qui s'appellent "height_cm" et "weight_kg" pour ne cons
    def convert_height(height_list):
        """ Convertit la hauteur en cm """
        try:
            return int(height_list[1].replace(" cm", ""))  # Prend la valeur en cm
        except:
            return "Unknown"

    def convert_weight(weight_list):
        """ Convertit le poids en kg """
        try:
            return int(weight_list[1].replace(" kg", ""))  # Prend la valeur en kg
        except:
            return "Unknown"

    df["height_cm"] = df["appearance"].apply(lambda x: convert_height(x["height"]) if "height" in x else "inconnue")
    df["weight_kg"] = df["appearance"].apply(lambda x: convert_weight(x["weight"]) if "weight" in x else "inconnue")

    # Suppression des colonnes inutiles
    # On peut supprimer l'apparance car les seuls informations utiles sont la taille et le poids
    # Ces derniers sont ajouté en cm et en kg en dohors d'appearance
    df.drop(columns=["slug", "work", "appearance", "connections", "images"], inplace=True, errors="ignore")


    print("✅ Données nettoyées et préparées avec gestion des valeurs manquantes !")
    return df
```

```python
def show_summary(df):
    """ Étape 6 : Vérification finale - Affichage d'un résumé des données """
    print("\n📊 Aperçu des données :")
    print(df.head())
    #print("\n🔍 Statistiques générales :")
    #print(df.describe(include="all"))
```

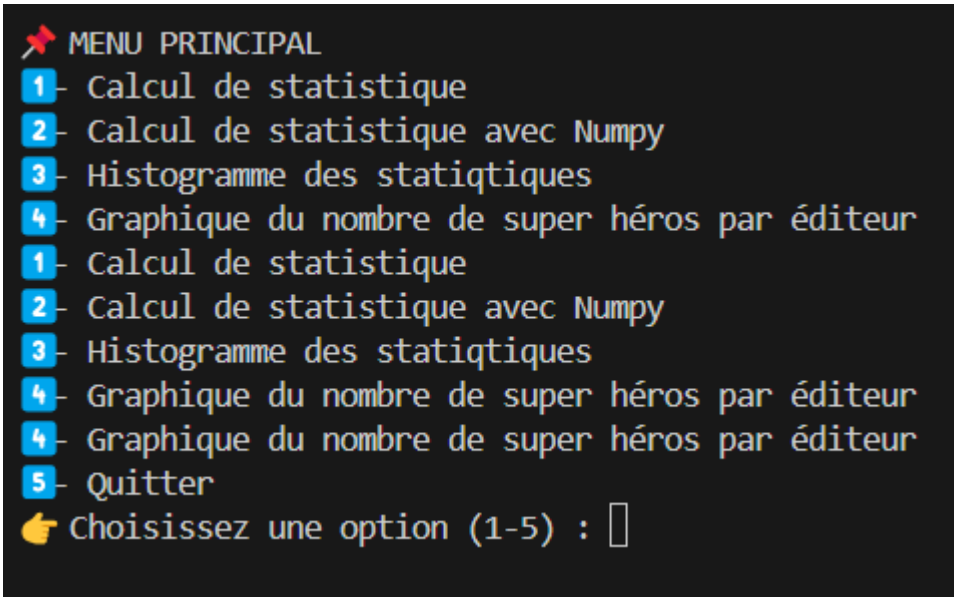
```python
def Etape6():
    df = None
    df = extract_data()
    df = clean_data(df)
    show_summary(df)
    return df
```

📊 Aperçu des données :

	id	name	powerstats	biography	height_cm	weight_kg
0	1	A-Bomb	{'intelligence': 38, 'strength': 100, 'speed':...	{'fullName': 'Richard Milhouse Jones', 'alterE...	203	441
1	2	Abe Sapien	{'intelligence': 88, 'strength': 28, 'speed': ...	{'fullName': 'Abraham Sapien', 'alterEgos': 'N...	191	65
2	3	Abin Sur	{'intelligence': 50, 'strength': 90, 'speed': ...	{'fullName': '', 'alterEgos': 'No alter egos f...	185	90
3	4	Abomination	{'intelligence': 63, 'strength': 80, 'speed': ...	{'fullName': 'Emil Blonsky', 'alterEgos': 'No ...	203	441
4	5	Abraxas	{'intelligence': 88, 'strength': 63, 'speed': ...	{'fullName': 'Abraxas', 'alterEgos': 'No alter...	0	0

Etape 7 :

```python
#Etape 7
def menu(df):
    vues(df)
    while True:
        print("\n📌 MENU PRINCIPAL")
        print("1️⃣ - Calcul de statistique")
        print("2️⃣ - Calcul de statistique avec Numpy")
        print("3️⃣ - Histogramme des statiqtiques")
        print("4️⃣ - Graphique du nombre de super héros par éditeur")
        print("5️⃣ - Quitter")


        choix = input("👉 Choisissez une option (1-5) : ")

        if choix == "1":
            calculate_statistics(df)
        elif choix == "2":
            calculate_statistics_numpy(df)
        elif choix == "3":
            histogramme_stat(df)
        elif choix == "4":
            graphique_editeur(df)
        elif choix == "5" :
            print("👋 Au revoir !")
            break
        else:
            print("❌ Option invalide, veuillez réessayer.")
```



Etape 8 :

```python
#Etape 8
def calculate_statistics(df):
    # Extraire les powerstats (force, intelligence, vitesse) dans une nouvelle DataFrame
    powerstats = df['powerstats'].apply(pd.Series)

    # Calculer les statistiques descriptives
    print("\nStatistiques descriptives des super-héros :")
    print(f"\nMoyenne de la force : {powerstats['strength'].mean():.2f}")
    print(f"Moyenne de l'intelligence : {powerstats['intelligence'].mean():.2f}")
    print(f"Moyenne de la vitesse : {powerstats['speed'].mean():.2f}")

    print(f"\nMédiane de la force : {powerstats['strength'].median():.2f}")
    print(f"Médiane de l'intelligence : {powerstats['intelligence'].median():.2f}")
    print(f"Médiane de la vitesse : {powerstats['speed'].median():.2f}")

    print(f"\nVariance de la force : {powerstats['strength'].var():.2f}")
    print(f"Variance de l'intelligence : {powerstats['intelligence'].var():.2f}")
    print(f"Variance de la vitesse : {powerstats['speed'].var():.2f}")

```

```
Statistiques descriptives des super-héros :

Moyenne de la force : 41.88
Moyenne de l'intelligence : 64.50
Moyenne de la vitesse : 40.24

Médiane de la force : 32.00
Médiane de l'intelligence : 63.00
Médiane de la vitesse : 33.00

Variance de la force : 1105.19
Variance de l'intelligence : 400.30
Variance de la vitesse : 599.32
```

Etape 9 :

```python
#Etape 9

def calculate_statistics_numpy(df):
    # Extraire les powerstats (force, intelligence, vitesse) dans une nouvelle DataFrame
    powerstats = df['powerstats'].apply(pd.Series)

    # Calculer les statistiques descriptives
    print("\nStatistiques descriptives des super-héros :")
    print(f"\nMoyenne de la force : {np.mean(powerstats['strength']):.2f}")
    print(f"Moyenne de l'intelligence : {np.mean(powerstats['intelligence']):.2f}")
    print(f"Moyenne de la vitesse : {np.mean(powerstats['speed']):.2f}")

    print(f"\nMédiane de la force : {np.median(powerstats['strength']):.2f}")
    print(f"Médiane de l'intelligence : {np.median(powerstats['intelligence']):.2f}")
    print(f"Médiane de la vitesse : {np.median(powerstats['speed']):.2f}")

    print(f"\nVariance de la force : {np.var(powerstats['strength']):.2f}")
    print(f"Variance de l'intelligence : {np.var(powerstats['intelligence']):.2f}")
    print(f"Variance de la vitesse : {np.var(powerstats['speed']):.2f}")
```

```
Statistiques descriptives des super-héros :

Moyenne de la force : 41.88
Moyenne de l'intelligence : 64.50
Moyenne de la vitesse : 40.24

Médiane de la force : 32.00
Médiane de l'intelligence : 63.00
Médiane de la vitesse : 33.00

Variance de la force : 1103.23
Variance de l'intelligence : 399.59
Variance de la vitesse : 598.25
```

Etape 10 :

```python
#Etape 10
def histogramme_stat(df):
    powerstats = df['powerstats'].apply(pd.Series)
    # Histogramme pour l'intelligence
    plt.figure(figsize=(10, 6))
    plt.hist(powerstats['intelligence'], bins=5, color='green', alpha=0.7, label="Intelligence", rwidth=0.5)
    plt.title("Distribution de l'Intelligence des Super-héros")
    plt.xlabel("Intelligence")
    plt.ylabel("Fréquence")
    plt.legend()
    plt.show()


    # Histogramme pour la force
    plt.figure(figsize=(10, 6))
    plt.hist(powerstats['strength'], bins=5, color='blue', alpha=0.7, label="Force", rwidth=0.5)
    plt.title("Distribution de la Force des Super-héros")
    plt.xlabel("Force")
    plt.ylabel("Fréquence")
    plt.legend()
    plt.show()
```

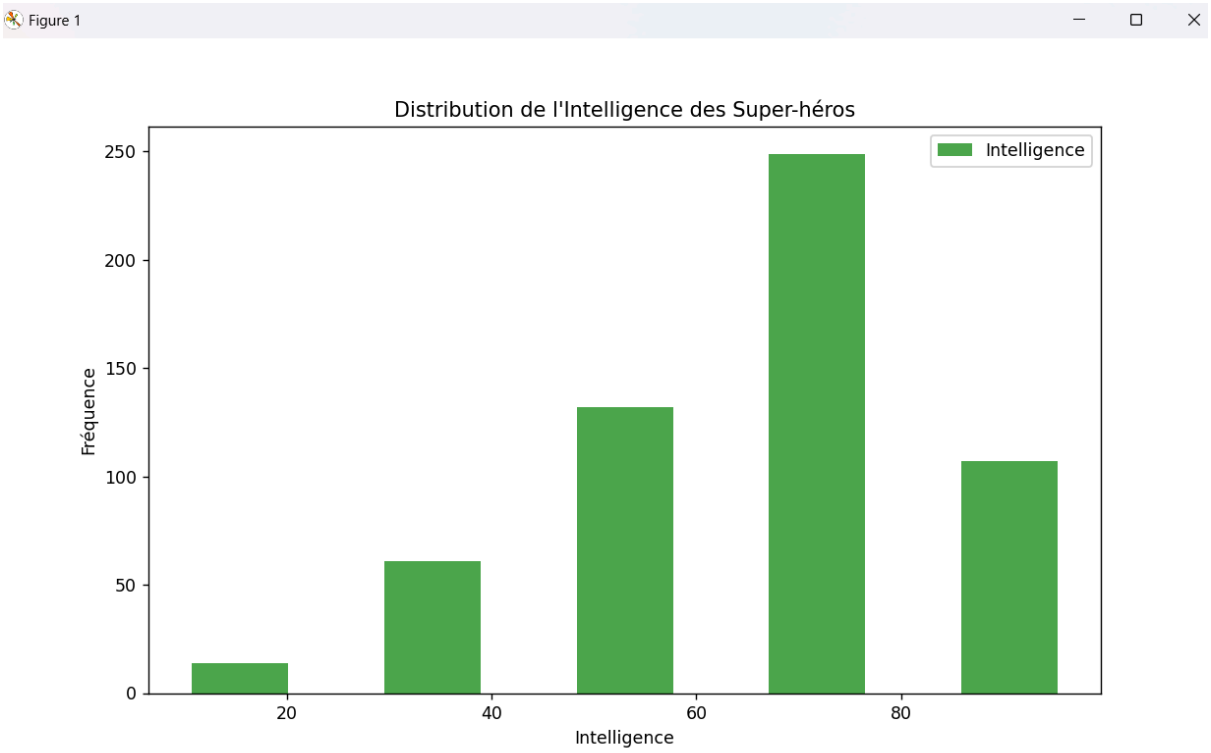


Figure 1
Distribution de l'Intelligence des Super-héros
Intelligence
Fréquence
Intelligence
0
50
100
150
200
250
20
40
60
80

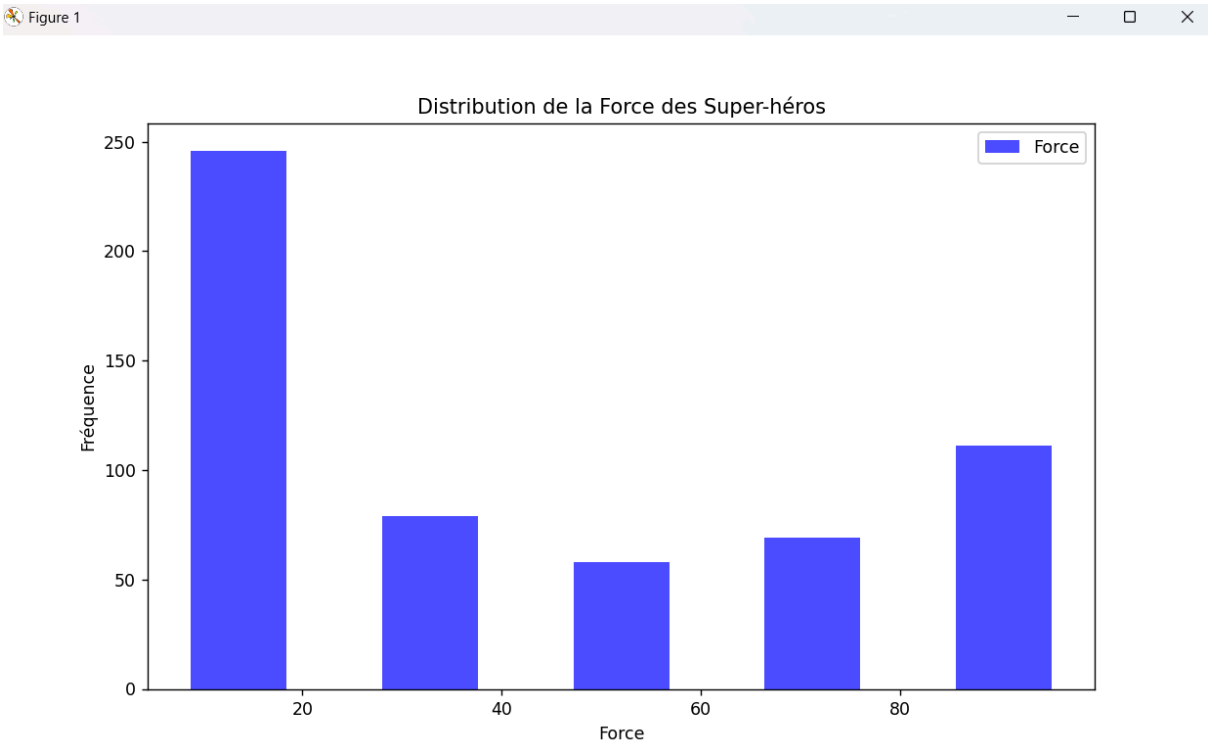
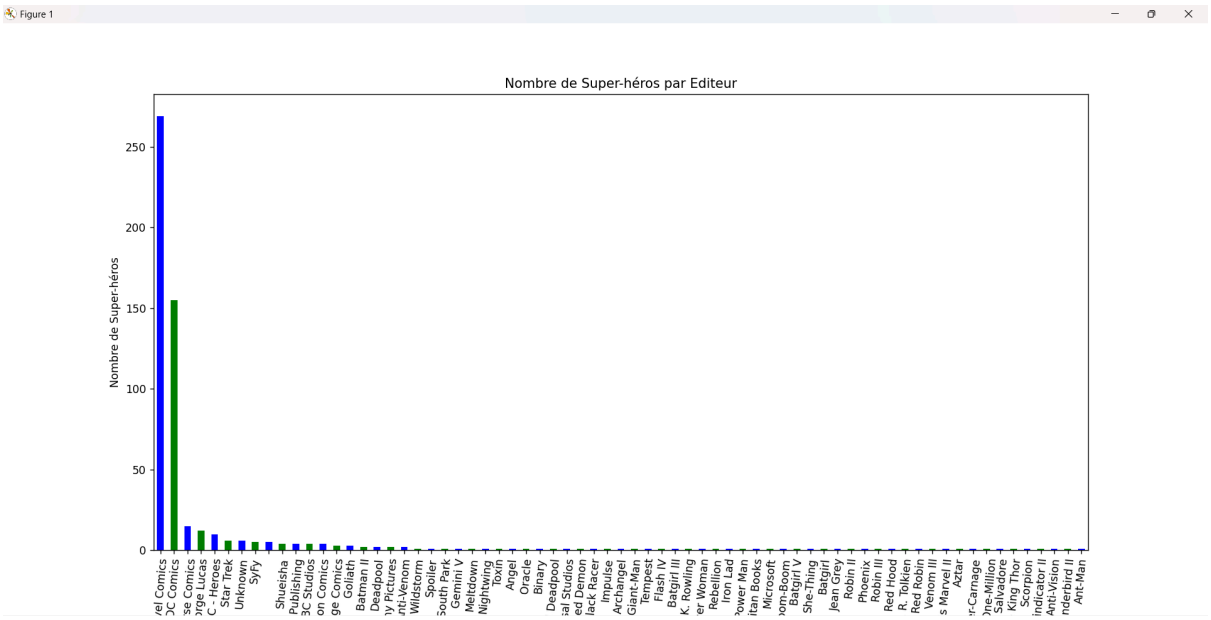


Figure 1
Distribution de la Force des Super-héros
Force
Fréquence
Force
0
50
100
150
200
250
20
40
60
80

Etape 11 :

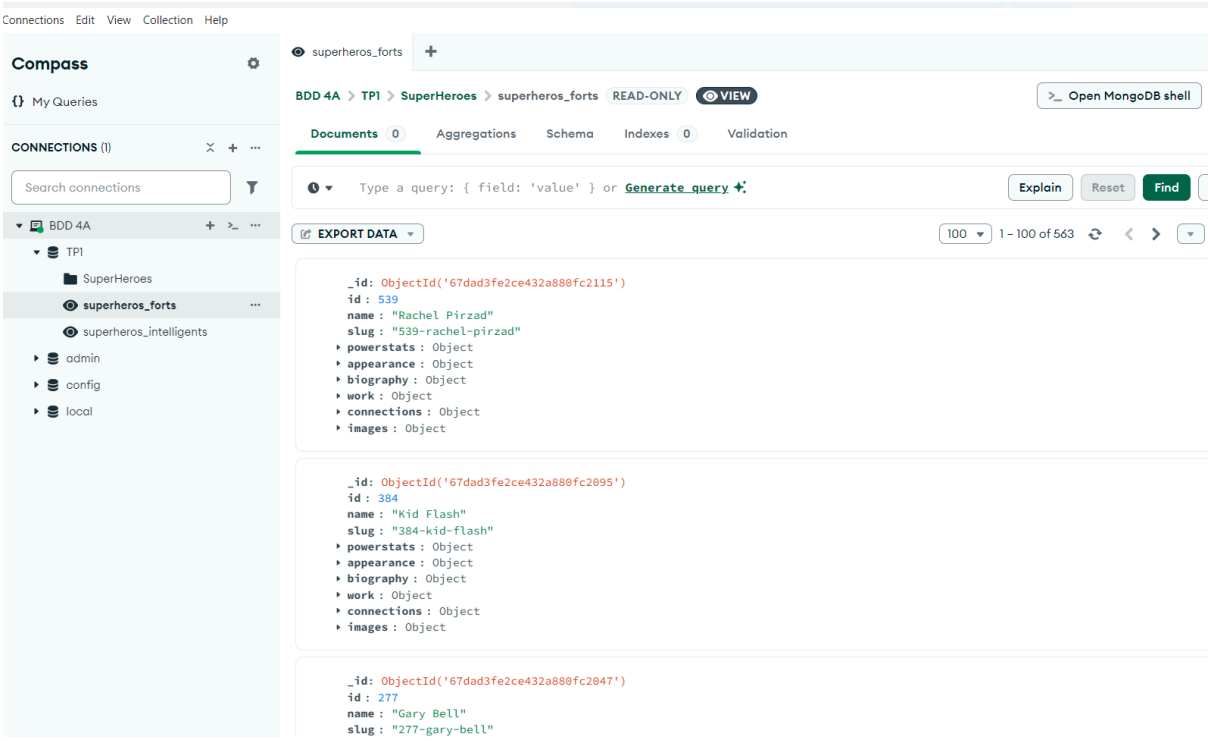
```python
#Etape 11
def extract_publishers(df):
    publishers = []
    for index, row in df.iterrows():
        # Vérifier si 'biography' existe et si 'publisher' existe dans le dictionnaire
        if isinstance(row.get("biography"), dict) and "publisher" in row["biography"]:
            publisher = row["biography"]["publisher"]
            publishers.append(publisher)
        else:
            publishers.append(None)  # Si 'publisher' ou 'biography' n'existe pas, ajouter None
    return publishers
```

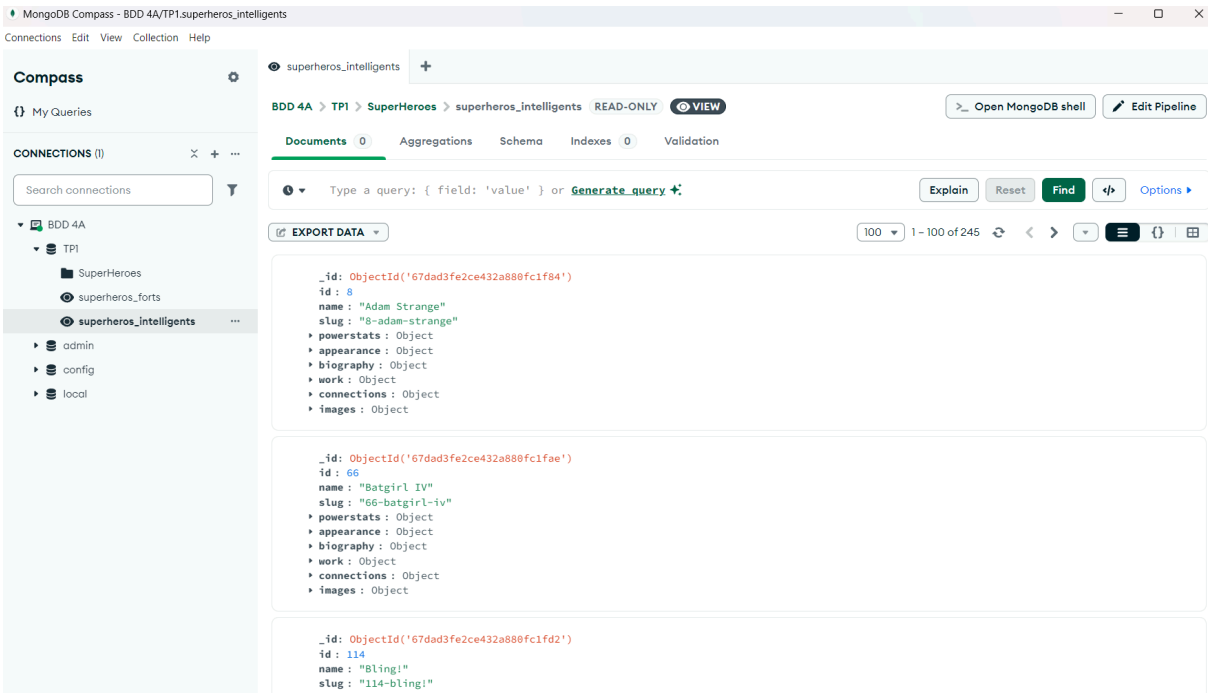
```python
def graphique_editeur(df):
    # Exemple de données
    editeurs = extract_publishers(df)
    editeurs_counts = pd.Series(editeurs).value_counts()

    # Graphique à barres pour les éditeurs
    editeurs_counts.plot(kind='bar', color=['blue', 'green'])
    plt.title("Nombre de Super-héros par Editeur")
    plt.xlabel("Editeur")
    plt.ylabel("Nombre de Super-héros")
    plt.xticks(rotation=85)
    plt.show()
```



Etape 12 et 13 :

```python
#Etape 12 et 13
def vues(df):
    # Extraction des statistiques de puissance sous forme de DataFrame
    powerstats = df['powerstats'].apply(pd.Series)

    # Calcul de la moyenne de l'intelligence
    average_intelligence = int(np.mean(powerstats['intelligence']))  # Convertir en entier pour éviter l'erreur

    print(f"Moyenne Intelligence : {average_intelligence}")


    # Créer une vue pour l'intelligence supérieure à la moyenne
    db.command('create', 'superheros_intelligents', viewOn="SuperHeroes", pipeline=[
        {"$match": {"powerstats.intelligence": {"$gt": average_intelligence}}}
    ])

    # Créer une vue pour l'intelligence supérieure à la moyenne
    db.command('create', 'superheros_forts', viewOn="SuperHeroes", pipeline=[
        {"$sort": {"powerstats.strength": 1}}
    ])
```





II.TP2 : NoSQL orienté colonne avec Cassandra

1) Installation

Etape 1 :
On installe Docker

Etape 2 :
On installe Cassandre avec cette ligne de commande :
docker pull cassandra:latest
Ensuite on exécute cette commande :

```
C:\Users\eloua>docker run --name cass_cluster cassandra:latest
```

Cela nous donne ce résultat :

```
INFO  [OptionalTasks:1] 2025-03-21 13:58:13,486 CassandraRoleManager.java:42
9 - Created default superuser role 'cassandra'
```

Etape 3 :
On lance cette commande, on est prêt à utiliser Cassandra

```
C:\Users\eloua>docker exec -it cass_cluster cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.3 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh>
```

2) Cassandra Query Language

Etape 4 : Création du Keyspace
CREATE KEYSPACE shoutapp WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 1};

Etape 5 : Vérification du Keyspace

```
cqlsh> DESCRIBE KEYSPACES;

shoutapp  system_auth         system_schema  system_views
system    system_distributed  system_traces  system_virtual_schema
```

```
cqlsh:shoutapp>
```

On se trouve bien dans shoutapp.

Etape 6 : Création des tables
On crée l'entièreté des tables

```
cqlsh:shoutapp> CREATE TABLE users (
            ...     username TEXT PRIMARY KEY,
            ...     password TEXT
            ... );
cqlsh:shoutapp> CREATE TABLE following (
            ...     username TEXT,
            ...     followed TEXT,
            ...     PRIMARY KEY (username, followed)
            ... );
cqlsh:shoutapp>
cqlsh:shoutapp> CREATE TABLE followers (
            ...     username TEXT,
            ...     following TEXT,
            ...     PRIMARY KEY (username, following)
            ... );
cqlsh:shoutapp>
cqlsh:shoutapp> CREATE TABLE shouts (
            ...     shout_id UUID PRIMARY KEY,
            ...     username TEXT,
            ...     body TEXT
            ... );
cqlsh:shoutapp>
cqlsh:shoutapp> CREATE TABLE usershouts (
            ...     username TEXT,
            ...     usershout_id UUID,
            ...     body TEXT,
            ...     PRIMARY KEY (username, usershout_id)
            ... );
cqlsh:shoutapp>
cqlsh:shoutapp> CREATE TABLE shoutwall (
            ...     username TEXT,
            ...     shoutwall_id UUID,
            ...     posted_by TEXT,
            ...     body TEXT,
            ...     PRIMARY KEY (username, shoutwall_id)
            ... );
cqlsh:shoutapp>
```

Etape 7 : Insertion des données

```
cqlsh:shoutapp> INSERT INTO users (username, password) VALUES ('homer', 'donuts');
cqlsh:shoutapp> INSERT INTO users (username, password) VALUES ('marge', 'family');
cqlsh:shoutapp> INSERT INTO users (username, password) VALUES ('bart', 'skateboard');
cqlsh:shoutapp> INSERT INTO users (username, password) VALUES ('lisa', 'saxophone');
cqlsh:shoutapp> INSERT INTO users (username, password) VALUES ('maggie', 'pacifier');
cqlsh:shoutapp>
cqlsh:shoutapp> INSERT INTO following (username, followed) VALUES ('homer', 'marge');
cqlsh:shoutapp> INSERT INTO following (username, followed) VALUES ('homer', 'bart');
cqlsh:shoutapp> INSERT INTO following (username, followed) VALUES ('marge', 'lisa');
cqlsh:shoutapp> INSERT INTO following (username, followed) VALUES ('bart', 'lisa');
cqlsh:shoutapp> INSERT INTO following (username, followed) VALUES ('lisa', 'maggie');
cqlsh:shoutapp>
cqlsh:shoutapp> INSERT INTO followers (username, following) VALUES ('marge', 'homer');
cqlsh:shoutapp> INSERT INTO followers (username, following) VALUES ('bart', 'homer');
cqlsh:shoutapp> INSERT INTO followers (username, following) VALUES ('lisa', 'marge');
cqlsh:shoutapp> INSERT INTO followers (username, following) VALUES ('lisa', 'bart');
cqlsh:shoutapp> INSERT INTO followers (username, following) VALUES ('maggie', 'lisa');
cqlsh:shoutapp>
cqlsh:shoutapp> INSERT INTO shouts (shout_id, username, body) VALUES (uuid(), 'homer', 'Mmm
... donuts');
cqlsh:shoutapp> INSERT INTO shouts (shout_id, username, body) VALUES (uuid(), 'marge', 'Tim
e to clean the house!');
cqlsh:shoutapp> INSERT INTO shouts (shout_id, username, body) VALUES (uuid(), 'bart', 'Eat
my shorts!');
cqlsh:shoutapp> INSERT INTO shouts (shout_id, username, body) VALUES (uuid(), 'lisa', 'Play
ing my saxophone.');
cqlsh:shoutapp> INSERT INTO shouts (shout_id, username, body) VALUES (uuid(), 'maggie', 'Go
o goo ga ga!');
cqlsh:shoutapp>
```

On insère les données

Etape 8 : Requêtes CQL
afficher tous les utilisateur :
*SELECT * FROM users;*

```
cqlsh:shoutapp> SELECT * FROM users;

 username | password
----------+------------
   maggie |    pacifier
    marge |      family
     lisa |   saxophone
     bart | skateboard
    homer |      donuts

(5 rows)
```

Afficher qui un utilisateur suit
SELECT followed FROM following WHERE username = 'homer';

```
cqlsh:shoutapp> SELECT followed FROM following WHERE username = 'homer';

 followed
----------
     bart
    marge

(2 rows)
```

Afficher qui suit un utilisateur :
SELECT following FROM followers WHERE username = 'lisa';

```
cqlsh:shoutapp> SELECT following FROM followers WHERE username = 'lisa';

 following
-----------
      bart
     marge

(2 rows)
```

Afficher tous les shouts :
*SELECT * FROM shouts;*

```
cqlsh:shoutapp> SELECT * FROM shouts;

 shout_id                             | body                      | username
--------------------------------------+---------------------------+----------
 adfb104d-3ef8-426f-a02c-28df257b83a4 |              Mmm... donuts |    homer
 94ca062b-54d6-49cf-a2e7-1146736c484c |            Eat my shorts! |     bart
 ffc9cb7e-3b0a-437a-b339-3a1f198b9841 | Time to clean the house!  |    marge
 3a4f2759-7bd9-475b-9771-3d43ab81b410 |            Goo goo ga ga! |   maggie
 8268b718-2151-419e-8871-64c62e6b6f25 |     Playing my saxophone. |     lisa

(5 rows)
```

Afficher les shouts d'un utilisateur spécifique :
*SELECT * FROM shouts WHERE username = 'bart' allow filtering;*

```
cqlsh:shoutapp> SELECT * FROM shouts WHERE username = 'bart' allow filtering;

 shout_id                             | body           | username
--------------------------------------+----------------+----------
 94ca062b-54d6-49cf-a2e7-1146736c484c | Eat my shorts! |     bart
```

Etape 9 :
Mettre à jour le mot de passe d'Homer :
UPDATE users SET password = 'newpassword' WHERE username = 'homer';

Homer commence à suivre Lisa :
INSERT INTO following (username, followed) VALUES ('homer', 'lisa');

Homer arrête de suivre Bart
DELETE FROM following WHERE username = 'homer' AND followed = 'bart';

Maggie commence à suivre Bart
INSERT INTO following (username, followed) VALUES ('maggie', 'bart');

Marge n'est plus suivi par Homer
DELETE FROM followers WHERE username = 'marge' AND following = 'homer';

Mettre à jour le shout d'Homer
UPDATE shouts SET body = 'I love donuts even more!' WHERE shout_id = adfb104d-3ef8-426f-a02c-28df257b83a4;

```
cqlsh:shoutapp> UPDATE shouts SET body = 'I love donuts even more!' WHERE shout_id = adfb104d-3ef8-426
f-a02c-28df257b83a4;
cqlsh:shoutapp> SELECT * FROM shouts;

 shout_id                             | body                      | username
--------------------------------------+---------------------------+----------
 adfb104d-3ef8-426f-a02c-28df257b83a4 | I love donuts even more!  |    homer
 94ca062b-54d6-49cf-a2e7-1146736c484c |           Eat my shorts!  |     bart
 ffc9cb7e-3b0a-437a-b339-3a1f198b9841 | Time to clean the house!  |    marge
 3a4f2759-7bd9-475b-9771-3d43ab81b410 |           Goo goo ga ga!  |   maggie
 8268b718-2151-419e-8871-64c62e6b6f25 |    Playing my saxophone.  |     lisa
```

Étape 10 :
Ajout de la colonne age dans users :
ALTER TABLE users ADD age INT;

Ajouter l'âge de Homer :
UPDATE users SET age = 36 WHERE username = 'homer';

```
cqlsh:shoutapp> select * from users ;

 username | age  | password
----------+------+-------------
   maggie | null |    pacifier
    marge | null |      family
     lisa | null |   saxophone
     bart | null |  skateboard
    homer | null | newpassword

(5 rows)
cqlsh:shoutapp> UPDATE users SET age = 36 WHERE username = 'homer';
cqlsh:shoutapp> select * from users ;

 username | age  | password
----------+------+-------------
   maggie | null |    pacifier
    marge | null |      family
     lisa | null |   saxophone
     bart | null |  skateboard
    homer |   36 | newpassword

(5 rows)
cqlsh:shoutapp>
```

Etape 11 :
Sélectionner tous les utilisateurs ayant 36 ans :
*SELECT * FROM users WHERE age = 36;*

```
cqlsh:shoutapp> SELECT * FROM users WHERE age = 36;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it
might involve data filtering and thus may have unpredictable performance. If you want to execute this
query despite the performance unpredictability, use ALLOW FILTERING"
cqlsh:shoutapp>
```

Problème : La requête ne fonctionne pas. Elle fonctionne en SQL mais Cassandra ne supporte pas les requêtes sans index sur une colonne non clé.

Etape 12 :
Création d'un index sur age et réessai :
CREATE INDEX age_index ON users(age);

*SELECT * FROM users WHERE age = 36;*

```
cqlsh:shoutapp> CREATE INDEX age_index ON users(age);
cqlsh:shoutapp> SELECT * FROM users WHERE age = 36;

 username | age | password
----------+-----+-------------
    homer |  36 | newpassword
```

Grâce à l'index, la commande fonctionne.

3) Cassandra avec JS

L'exécution du code ne fonctionne pas :

```
C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\cassandra-nodejs>node cassandra-exemple.js
There was an error NoHostAvailableError: All host(s) tried for query failed. First host tried, 127.0.0.1:9042: Error: connect ECONNREFUSED 127.0.0.1:9042
    at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1634:16) {
  errno: -4078,
  code: 'ECONNREFUSED',
  syscall: 'connect',
  address: '127.0.0.1',
  port: 9042
}. See innerErrors.
    at ControlConnection._borrowFirstConnection (C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\cassandra-nodejs\node_modules\cassandra-driver\lib\control-connection.js:307:15)
    at process.processTicksAndRejections (node:internal/process/task_queues:105:5)
    at async ControlConnection._initializeConnection (C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\cassandra-nodejs\node_modules\cassandra-driver\lib\control-connection.js:526:7)
    at async ControlConnection.init (C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\cassandra-nodejs\node_modules\cassandra-driver\lib\control-connection.js:212:5)
    at async Client._connect (C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\cassandra-nodejs\node_modules\cassandra-driver\lib\client.js:513:5)
    at async run (C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\cassandra-nodejs\cassandra-exemple.js:13:9) {
  info: 'Represents an error when a query cannot be performed because no host is available or could be reached by the driver.',
  innerErrors: {
    '127.0.0.1:9042': Error: connect ECONNREFUSED 127.0.0.1:9042
        at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1634:16) {
      errno: -4078,
      code: 'ECONNREFUSED',
      syscall: 'connect',
      address: '127.0.0.1',
      port: 9042
    }
  }
}
```

Pourtant on peut bien voir que mon utilisateur est bien administrateur de docker :

```
PS C:\WINDOWS\system32> net localgroup docker-users "eloua" /add
>>
L'erreur système 1378 s'est produite.

Le nom de compte spécifié est déjà membre du groupe.
```

On voit aussi que Docker a bien les droits sur le port 9042 :

```
PS C:\WINDOWS\system32> docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED       STATUS       PORTS                                                NAMES
1c05004948ea   cassandra:latest   "docker-entrypoint.s…"   2 hours ago   Up 2 hours   7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp   cass_cluster
```

III.TP3 : NoSQL orienté clé-valeur avec DynamoDB

0) Questions préliminaires

1) Installation sous windows

Etape 1 à 5 : fait

```
C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB>java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDB
Local.jar -sharedDb
Initializing DynamoDB Local with the following configuration:
Port:   8000
InMemory:       false
Version:        2.6.0
DbPath: null
SharedDb:       true
shouldDelayTransientStatuses:   false
CorsParams:     null
```

Etape 6 :

```
C:\Users\eloua>pip install boto3
Collecting boto3
  Downloading boto3-1.37.17-py3-none-any.whl.metadata (6.7 kB)
Collecting botocore<1.38.0,>=1.37.17 (from boto3)
  Downloading botocore-1.37.17-py3-none-any.whl.metadata (5.7 kB)
Collecting jmespath<2.0.0,>=0.7.1 (from boto3)
  Downloading jmespath-1.0.1-py3-none-any.whl.metadata (7.6 kB)
Collecting s3transfer<0.12.0,>=0.11.0 (from boto3)
  Downloading s3transfer-0.11.4-py3-none-any.whl.metadata (1.7 kB)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\eloua
\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\l
ocalcache\local-packages\python311\site-packages (from botocore<1.38.0,>=1.3
7.17->boto3) (2.9.0.post0)
Requirement already satisfied: urllib3!=2.2.0,<3,>=1.25.4 in c:\users\eloua\
appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\lo
calcache\local-packages\python311\site-packages (from botocore<1.38.0,>=1.37
.17->boto3) (2.3.0)
Requirement already satisfied: six>=1.5 in c:\users\eloua\appdata\local\pack
ages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-pac
kages\python311\site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.
38.0,>=1.37.17->boto3) (1.17.0)
Downloading boto3-1.37.17-py3-none-any.whl (139 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.6/139.6 kB 2.8 MB/s eta 0:00:00
Downloading botocore-1.37.17-py3-none-any.whl (13.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.4/13.4 MB 9.8 MB/s eta 0:00:00
Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Downloading s3transfer-0.11.4-py3-none-any.whl (84 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.4/84.4 kB 4.9 MB/s eta 0:00:00
Installing collected packages: jmespath, botocore, s3transfer, boto3
Successfully installed boto3-1.37.17 botocore-1.37.17 jmespath-1.0.1 s3trans
fer-0.11.4

[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update, run: C:\Users\eloua\AppData\Local\Microsoft\WindowsApps\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe -m pip install
 --upgrade pip
```

Etape 7 :

```
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\Dyn
amoDB> & C:/Users/eloua/AppData/Local/Microsoft/WindowsApps
/python3.11.exe "c:/Users/eloua/OneDrive/Documents/INSA hdf
/4A/S8/BDD/DynamoDB/Dynamodb.py"
Table TestTable created successfully.
Item inserted: {'id': '1', 'value': 'This is a test'}
Retrieved item: {'value': 'This is a test', 'id': '1'}
```

2) Premiers pas avec DynamoDB et les super héros

Etape 8 :
Dans Amazon DynamoDB, une clé primaire est utilisée pour identifier de manière unique chaque élément d'une table. Il existe deux types de clés primaires :

Clé primaire simple (Partition Key - HASH) :

Une seule colonne est utilisée comme clé primaire (ex. id).
DynamoDB utilise cette valeur pour répartir les données sur plusieurs partitions.
Clé primaire composite (Partition Key + Sort Key - HASH + RANGE) :

Une combinaison de deux colonnes (ex. id et name).
Permet des requêtes plus efficaces en utilisant la clé de tri.
Dans notre cas, nous allons utiliser id comme Partition Key (HASH).

Etape 9 :

```python
def create_table_heros(dynamodb):
    """Crée une table DynamoDB."""
    table_name = 'SuperHeroes'
    try:
        table = dynamodb.create_table(
            TableName=table_name,
            KeySchema=[
                {
                    'AttributeName': 'id',
                    'KeyType': 'HASH'
                }
            ],
            AttributeDefinitions=[
                {
                    'AttributeName': 'id',
                    'AttributeType': 'S'
                }
            ],
            ProvisionedThroughput={
                'ReadCapacityUnits': 10,
                'WriteCapacityUnits': 10
            }
        )
        table.wait_until_exists()
        print(f"Table {table_name} created successfully.")
    except ClientError as e:
        if e.response['Error']['Code'] == 'ResourceInUseException':
            print(f"Table {table_name} already exists.")
        else:
            raise
```

Etape 10 :

```python
def main():
    """Point d'entrée du script."""
    dynamodb = create_dynamodb_resource()
    table_name = 'SuperHeroes'

    # Créer la table
    create_table_heros(dynamodb)

    # Insérer et récupérer un élément
    item = {
    'id': '1',
    'name': 'A-Bomb',
    'slug': '1-a-bomb',
    'powerstats': {
        'intelligence': 38,
        'strength': 100,
        'speed': 17,
        'durability': 80,
        'power': 24,
        'combat': 64
    },
    'gender': 'Male',
    'race': 'Human',
    'height': "203 cm",
    'weight': "441 kg",
    'eyeColor': 'Yellow',
    'hairColor': 'No Hair',
    'placeOfBirth': 'Scarsdale, Arizona',
    'firstAppearance': 'Hulk Vol 2 #2 (April, 2008) (as A-Bomb)',
    'publisher': 'Marvel Comics',
```

```
    'alignment': 'good',
    'work': 'Musician, adventurer, author; formerly talk show host'
}

    item2 = {'id': '2',
    'nom': 'Abe Sapien',
    'intelligence': 88,
    'force': 28,
    'rapidité': 35,
    'durabilité': 65,
    'pouvoir': 100,
    'combat': 85
}

    insert_item(dynamodb, table_name, item)
    insert_item(dynamodb, table_name, item2)

    retrieved_item = get_item(dynamodb, table_name, {'id': '1'})
    print(f"Retrieved item: {retrieved_item}")
    retrieved_item = get_item(dynamodb, table_name, {'id': '2'})
    print(f"Retrieved item: {retrieved_item}")


if __name__ == '__main__':
    main()
```

En exécutant les commandes suivantes on peut voir que nos données sont bien insérées :

```
retrieved_item = get_item(dynamodb, table_name, {'id': '1'})
print(f"Retrieved item: {retrieved_item}")
print("")
retrieved_item = get_item(dynamodb, table_name, {'id': '2'})
print(f"Retrieved item: {retrieved_item}")
```

```
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB> & C:/Users/eloua/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/eloua/OneDrive/Documents/INSA hdf/4A/S8/BDD/DynamoDB/Dynamodb.py"
Retrieved item: {'firstAppearance': 'Hulk Vol 2 #2 (April, 2008) (as A-Bomb)', 'placeOfBirth': 'Scarsdale, Arizona', 'gender': 'Male', 'race': 'Human', 'work': 'Musician, adventurer, author; formerly talk show host', 'weight': '441 kg', 'eyeColor': 'Yellow', 'name': 'A-Bomb', 'powerstats': {'combat': Decimal('64'), 'power': Decimal('24'), 'strength': Decimal('100'), 'speed': Decimal('17'), 'intelligence': Decimal('38'), 'durability': Decimal('80')}, 'publisher': 'Marvel Comics', 'id': '1', 'hairColor': 'No Hair', 'alignment': 'good', 'slug': '1-a-bomb', 'height': '203 cm'}

Retrieved item: {'pouvoir': Decimal('100'), 'rapidité': Decimal('35'), 'durabilité': Decimal('65'), 'force': Decimal('28'), 'combat': Decimal('85'), 'id': '2', 'nom': 'Abe Sapien', 'intelligence': Decimal('88')}
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB>
```

Etape 11 :
On ajoute la fonction au code et dans le main on ajoute ceci :

```
# Créer la table
if (check_table_exists(dynamodb, table_name)==False):
    create_table_heros(dynamodb)
else:
    print("la table "+table_name+" existe déjà\n\n")
```

On voit lors de l'exécution que la table existe déjà :

```
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB> & C:/Users/e
loua/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/eloua/OneDri
ve/Documents/INSA hdf/4A/S8/BDD/DynamoDB/Dynamodb.py"
la table SuperHeroes existe déjà


Retrieved item: {'firstAppearance': 'Hulk Vol 2 #2 (April, 2008) (as A-Bomb)',
 'placeOfBirth': 'Scarsdale, Arizona', 'gender': 'Male', 'race': 'Human', 'wor
k': 'Musician, adventurer, author; formerly talk show host', 'weight': '441 kg
', 'eyeColor': 'Yellow', 'name': 'A-Bomb', 'powerstats': {'combat': Decimal('6
4'), 'power': Decimal('24'), 'strength': Decimal('100'), 'speed': Decimal('17'
), 'intelligence': Decimal('38'), 'durability': Decimal('80')}, 'publisher': '
```

Etape 12 :
L'affichage avec la fonction fonctionne correctement :

```
Scanning table...
{'firstAppearance': 'Hulk Vol 2 #2 (April, 2008) (as A-Bomb)', 'placeOfBirth':
 'Scarsdale, Arizona', 'gender': 'Male', 'race': 'Human', 'work': 'Musician, a
dventurer, author; formerly talk show host', 'weight': '441 kg', 'eyeColor': '
Yellow', 'name': 'A-Bomb', 'powerstats': {'combat': Decimal('64'), 'power': De
cimal('24'), 'strength': Decimal('100'), 'speed': Decimal('17'), 'intelligence
': Decimal('38'), 'durability': Decimal('80')}, 'publisher': 'Marvel Comics',
'id': '1', 'hairColor': 'No Hair', 'alignment': 'good', 'slug': '1-a-bomb', 'h
eight': '203 cm'}
{'pouvoir': Decimal('100'), 'rapidité': Decimal('35'), 'durabilité': Decimal('
65'), 'force': Decimal('28'), 'combat': Decimal('85'), 'id': '2', 'nom': 'Abe
Sapien', 'intelligence': Decimal('88')}
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB>
```

Etape 14 :

```python
#Etape 12 :
def find_heroes(dynamodb,table_name,attribut, valeur):
    table = dynamodb.Table(table_name)
    response = table.scan(
        FilterExpression=boto3.dynamodb.conditions.Attr(attribut).eq(valeur)
    )
    return response.get('Items', [])
```

```python
print(find_heroes(dynamodb, 'SuperHeroes', 'force', 28))
```

Quand on lance l'appel de la fonction on retrouve bien un super héros avec 28 de force :

```
[{'pouvoir': Decimal('100'), 'rapidité': Decimal('35'), 'durabilité': Decimal(
'65'), 'force': Decimal('28'), 'combat': Decimal('85'), 'id': '2', 'nom': 'Abe
 Sapien', 'intelligence': Decimal('88')}]
```

Etape 15 :
Pour retrouver Abe Sapien il est possible de chercher avec son id :

```
200     print(find_heroes(dynamodb, 'SuperHeroes', 'id', '2'))
201
202
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[]
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB> & C:/Users/e
loua/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/eloua/OneDri
ve/Documents/INSA hdf/4A/S8/BDD/DynamoDB/Dynamodb.py"
la table SuperHeroes existe déjà



[{'pouvoir': Decimal('100'), 'rapidité': Decimal('35'), 'durabilité': Decimal(
'65'), 'force': Decimal('28'), 'combat': Decimal('85'), 'id': '2', 'nom': 'Abe
 Sapien', 'intelligence': Decimal('88')}]
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB>
```

On peut aussi le faire avec son nom :

```
200     print(find_heroes(dynamodb, 'SuperHeroes', 'nom', 'Abe Sapien'))
201
202
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
'65'), 'force': Decimal('28'), 'combat': Decimal('85'), 'id': '2', 'nom': 'Abe
 Sapien', 'intelligence': Decimal('88')}]
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB> & C:/Users/e
loua/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/eloua/OneDri
ve/Documents/INSA hdf/4A/S8/BDD/DynamoDB/Dynamodb.py"
la table SuperHeroes existe déjà



[{'pouvoir': Decimal('100'), 'rapidité': Decimal('35'), 'durabilité': Decimal(
'65'), 'force': Decimal('28'), 'combat': Decimal('85'), 'id': '2', 'nom': 'Abe
 Sapien', 'intelligence': Decimal('88')}]
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB>
```

3) création d'une Application de Gestion de Bibliothèque avec DynamoDB

Les différentes fonctions seront à retrouver dans le code Bibliotheque.py

```python
#Vérification de l'existance des tables
if (check_table_exists(dynamodb, table_livres)==False):
    create_livres_table(dynamodb)
else:
    print("la table "+table_livres+" existe déjà\n\n")


if (check_table_exists(dynamodb, table_emprunt)==False):
    create_emprunts_table(dynamodb)
else:
    print("la table "+table_emprunt+" existe déjà\n\n")

ajouter_livre(dynamodb, table_livres, "978-1234567890", "Python DynamoDB", "Alice Dupont", 2023)
print(recuperer_livre(dynamodb, table_livres, "978-1234567890"))
print(lister_livres(dynamodb, table_livres))
mettre_a_jour_disponibilite(dynamodb, table_livres, "978-1234567890", False)
print(recuperer_livre(dynamodb, table_livres, "978-1234567890"))
supprimer_livre(dynamodb, table_livres, "978-1234567890")
```

```
Livre 'Python DynamoDB' ajouté avec succès !
{'annee_publication': Decimal('2023'), 'ISBN': '978-1234567890', 'titre': 'Python DynamoDB', 'auteur': 'Alice Dupont', 'disponible': True}
[{'annee_publication': Decimal('2023'), 'ISBN': '978-1234567890', 'titre': 'Python DynamoDB', 'auteur': 'Alice Dupont', 'disponible': True}]
Disponibilité mise à jour pour ISBN 978-1234567890.
{'annee_publication': Decimal('2023'), 'ISBN': '978-1234567890', 'titre': 'Python DynamoDB', 'auteur': 'Alice Dupont', 'disponible': False}
Livre ISBN 978-1234567890 supprimé.
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB>
```

Test des fonctions d'emprunt :

```python
# Test emprunt
emprunter_livre(dynamodb, table_livres, table_emprunt, "978-1234567890", "Jean Dupont")
print(recuperer_emprunt(dynamodb, table_emprunt, "emprunt_id_test"))
print(emprunts_par_utilisateur(dynamodb, table_emprunt, "Jean Dupont"))
print(emprunts_en_retard(dynamodb, table_emprunt))
retourner_livre(dynamodb, table_livres, table_emprunt, "emprunt_id_test")
```

```
Livre 978-1234567890 emprunté par Jean Dupont avec succès !
None
[{'date_emprunt': '2025-03-24', 'emprunt_id': 'a4a1e471-aa33-4061-a20c-6c0d9559bf75', 'ISBN': '978-1234567890', 'date_retour': None, 'utilisateur': '
Jean Dupont'}, {'date_emprunt': '2025-03-24', 'emprunt_id': '09f337a7-fbfe-4c78-98dc-56cf78c7a858', 'ISBN': '978-1234567890', 'date_retour': None, 'u
tilisateur': 'Jean Dupont'}]
[]
Emprunt introuvable ou déjà retourné.
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB>
```

```python
#Autres tests
print(livres_par_auteur(dynamodb, table_livres, "Alice Dupont"))
print(emprunts_depasse_duree(dynamodb, table_emprunt, 30))
print(livres_plus_empruntes(dynamodb, table_emprunt, 3))
```

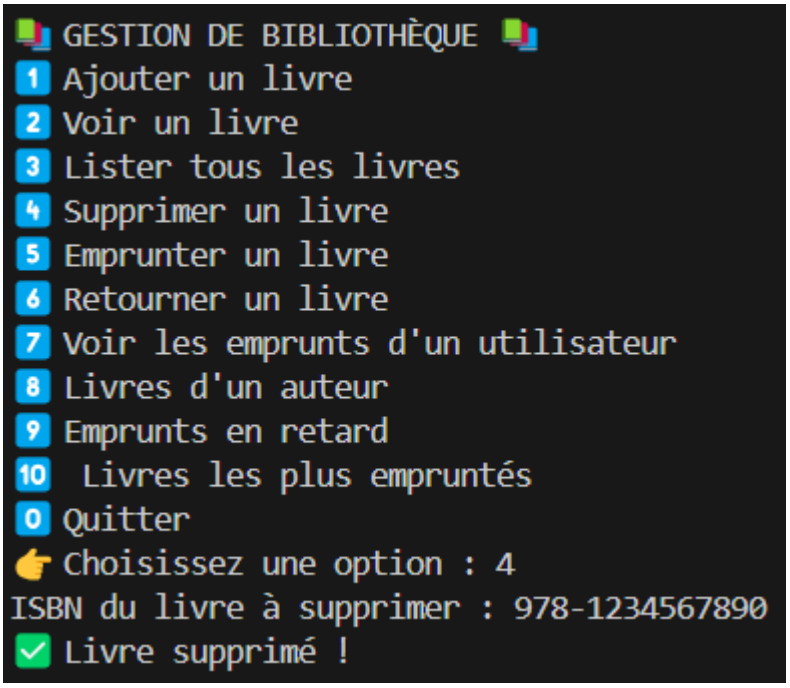
```
[{'annee_publication': Decimal('2023'), 'ISBN': '978-1234567890', 'titre': 'Python DynamoDB', 'auteur': 'Alice Dupont', 'disponible': False}]
[]
[('978-1234567890', 4)]
PS C:\Users\eloua\OneDrive\Documents\INSA hdf\4A\S8\BDD\DynamoDB>
```

On ajoute le menu pour en ligne de commande pour appeler les différentes fonctions :

```
📚 GESTION DE BIBLIOTHÈQUE 📚
1 Ajouter un livre
2 Voir un livre
3 Lister tous les livres
4 Supprimer un livre
5 Emprunter un livre
6 Retourner un livre
7 Voir les emprunts d'un utilisateur
8 Livres d'un auteur
9 Emprunts en retard
10  Livres les plus empruntés
0 Quitter
👉 Choisissez une option : 1
ISBN : 1
Titre : test1
Auteur : elouan
Année de publication : 2025
✅ Livre ajouté !
```

```
📚 GESTION DE BIBLIOTHÈQUE 📚
1 Ajouter un livre
2 Voir un livre
3 Lister tous les livres
4 Supprimer un livre
5 Emprunter un livre
6 Retourner un livre
7 Voir les emprunts d'un utilisateur
8 Livres d'un auteur
9 Emprunts en retard
10  Livres les plus empruntés
0 Quitter
👉 Choisissez une option : 2
ISBN du livre : 1
{'annee_publication': Decimal('2025'), 'ISBN': '1', 'titre': 'test1', 'auteur': 'elouan', 'disponible': True}
```

```
📚 GESTION DE BIBLIOTHÈQUE 📚
1 Ajouter un livre
2 Voir un livre
3 Lister tous les livres
4 Supprimer un livre
5 Emprunter un livre
6 Retourner un livre
7 Voir les emprunts d'un utilisateur
8 Livres d'un auteur
9 Emprunts en retard
10  Livres les plus empruntés
0 Quitter
👉 Choisissez une option : 3
{'annee_publication': Decimal('2025'), 'ISBN': '1', 'titre': 'test1', 'auteur': 'elouan', 'disponible': True}
{'annee_publication': Decimal('2023'), 'ISBN': '978-1234567890', 'titre': 'Python DynamoDB', 'auteur': 'Alice Dupont', 'disponible': False}
```



Le livre est bien supprimé :

```
📚 GESTION DE BIBLIOTHÈQUE 📚
1 Ajouter un livre
2 Voir un livre
3 Lister tous les livres
4 Supprimer un livre
5 Emprunter un livre
6 Retourner un livre
7 Voir les emprunts d'un utilisateur
8 Livres d'un auteur
9 Emprunts en retard
10  Livres les plus empruntés
0 Quitter
👉 Choisissez une option : 3
{'annee_publication': Decimal('2025'), 'ISBN': '1', 'titre': 'test1', 'auteur': 'elouan', 'disponible': True}
```

```
📚 GESTION DE BIBLIOTHÈQUE 📚
1 Ajouter un livre
2 Voir un livre
3 Lister tous les livres
4 Supprimer un livre
5 Emprunter un livre
6 Retourner un livre
7 Voir les emprunts d'un utilisateur
8 Livres d'un auteur
9 Emprunts en retard
10  Livres les plus empruntés
0 Quitter
👉 Choisissez une option : 5
ISBN du livre : 1
Nom de l'utilisateur : Sagnard
✅ Livre emprunté !
```

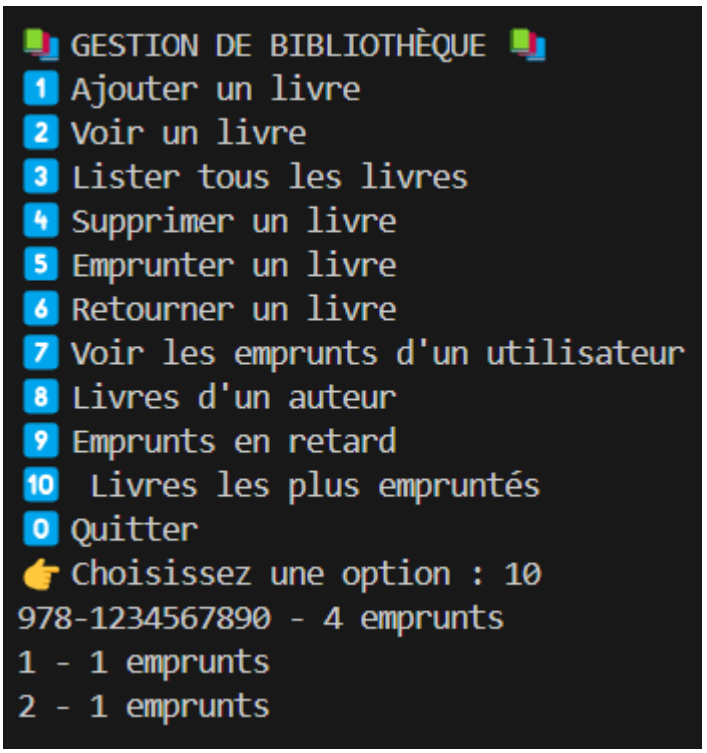
On retrouve bien l'emprunt dans la liste des emprunts :

```
📚 GESTION DE BIBLIOTHÈQUE 📚
1 Ajouter un livre
2 Voir un livre
3 Lister tous les livres
4 Supprimer un livre
5 Emprunter un livre
6 Retourner un livre
7 Voir les emprunts d'un utilisateur
8 Livres d'un auteur
9 Emprunts en retard
10  Livres les plus empruntés
0 Quitter
👉 Choisissez une option : 7
Nom de l'utilisateur : Sagnard
{'date_emprunt': '2025-03-25', 'emprunt_id': 'c091945f-2db7-4ea2-a5fe-a3dfa027d63a', 'ISBN': '1', 'date_retour': None, 'utilisateur': 'Sagnard'}
```

On peut voir ici la différence entre un livre qui a été retourné et un qui ne l'a pas été : Le premier a une date de retour et pas le 2eme :

```
📚 GESTION DE BIBLIOTHÈQUE 📚
1 Ajouter un livre
2 Voir un livre
3 Lister tous les livres
4 Supprimer un livre
5 Emprunter un livre
6 Retourner un livre
7 Voir les emprunts d'un utilisateur
8 Livres d'un auteur
9 Emprunts en retard
10  Livres les plus empruntés
0 Quitter
👉 Choisissez une option : 7
Nom de l'utilisateur : Sagnard
{'date_emprunt': '2025-03-25', 'emprunt_id': 'c091945f-2db7-4ea2-a5fe-a3dfa027
d63a', 'ISBN': '1', 'date_retour': '2025-03-25', 'utilisateur': 'Sagnard'}
{'date_emprunt': '2025-03-25', 'emprunt_id': '1bbc134e-4881-4880-8d87-91010e83
0c15', 'ISBN': '2', 'date_retour': None, 'utilisateur': 'Sagnard'}
```

```
📚 GESTION DE BIBLIOTHÈQUE 📚
1 Ajouter un livre
2 Voir un livre
3 Lister tous les livres
4 Supprimer un livre
5 Emprunter un livre
6 Retourner un livre
7 Voir les emprunts d'un utilisateur
8 Livres d'un auteur
9 Emprunts en retard
10  Livres les plus empruntés
0 Quitter
👉 Choisissez une option : 8
Nom de l'auteur : elouan
{'annee_publication': Decimal('2025'), 'ISBN': '1', 'titre': 'test1', 'auteur': 'elouan', 'disponible': False}
```



IV.Questionnaire de fin de module

Partie 1 : QCM

1) Quel type de base de données est MongoDB ?
   Réponse : c) Document
2) Quelle base de données est exclusivement proposée par AWS ?
   Réponse : b) DynamoDB
3) Quelle base de données utilise un modèle orienté colonnes ?
   Réponse : c) Cassandra
4) Laquelle de ces bases de données offre une gestion automatique de la scalabilité ?
   Réponse : b) DynamoDB
5) Quelle base de données offre un support ACID complet pour les transactions multi-documents ?
   Réponse : a) MongoDB

Partie 2 : Vrai ou faux

6) MongoDB stocke les données sous forme de BSON.
   Réponse : Vrai
7) Cassandra est mieux adapté aux applications nécessitant une haute disponibilité et de la scalabilité horizontale.
   Réponse : Vrai

8) DynamoDB est une base de données open-source.
   Réponse : Faux
9) Les trois bases de données offrent un modèle de cohérence strict.
   Réponse : Faux
10) MongoDB, DynamoDB et Cassandra nécessitent toutes une gestion manuelle complexe.
    Réponse : Faux

Partie 3 : Questions ouvertes

11) Quel est l'avantage principal de Cassandra par rapport aux deux autres bases de données ?
    Réponse : Cassandra est conçue pour la haute disponibilité et la scalabilité horizontale grâce à son architecture distribuée et sans point de défaillance unique.
12) Dans quel cas choisiriez-vous DynamoDB plutôt que MongoDB ou Cassandra ?
    Réponse : DynamoDB est un bon choix lorsqu'on a besoin d'une base de données NoSQL totalement managée, évolutive, avec une faible latence et intégrée aux services AWS, notamment pour les applications serverless.
13) Quel est l'inconvénient majeur de DynamoDB par rapport aux bases open-source ?
    Réponse : Son coût peut être élevé à grande échelle, et il est propriétaire d'AWS, ce qui entraîne un lock-in technologique.
14) Expliquez pourquoi MongoDB est souvent utilisé pour les applications web.
    Réponse : MongoDB permet une grande flexibilité avec son modèle de documents JSON-like, ce qui facilite le stockage et la manipulation des données complexes pour les applications web dynamiques.
15) Si vous deviez concevoir une application nécessitant une forte disponibilité et de faibles latences sur AWS, quelle base de données choisiriez-vous et pourquoi ?
    Réponse : DynamoDB serait un choix optimal car il est conçu pour offrir une latence ultra-faible, une scalabilité automatique et une haute disponibilité grâce à sa distribution multi-régionnelle.

Partie 4 : Exemple de Base de Données et Requêtes en MongoDB, DynamoDB et Cassandra

Etape 1 :

```
use ecommerce;

db.products.insertMany([
  {
    product_id: 1,
    name: "Laptop Lenovo",
    category: "Informatique",
    price: 1200,
```

```
    stock: 50,
    ratings: 4.5
  },
  {
    product_id: 2,
    name: "iPhone 14",
    category: "Smartphones",
    price: 999,
    stock: 30,
    ratings: 4.7
  }
]);
```

Etape 2 :

```
aws dynamodb put-item --table-name Products --item '{
  "product_id": {"N": "1"},
  "name": {"S": "Laptop Lenovo"},
  "category": {"S": "Informatique"},
  "price": {"N": "1200"},
  "stock": {"N": "50"},
  "ratings": {"N": "4.5"}
}'

aws dynamodb put-item --table-name Products --item '{
  "product_id": {"N": "2"},
  "name": {"S": "iPhone 14"},
  "category": {"S": "Smartphones"},
  "price": {"N": "999"},
  "stock": {"N": "30"},
  "ratings": {"N": "4.7"}
}'
```

Etape 3 :

```
CREATE KEYSPACE ecommerce WITH replication = {'class':
'SimpleStrategy', 'replication_factor': 1};

USE ecommerce;

CREATE TABLE products (
  product_id int PRIMARY KEY,
  name text,
  category text,
```

```
  price double,
  stock int,
  ratings float
);
```

Etape 4 :

```
INSERT INTO products (product_id, name, category, price, stock,
ratings)
VALUES (1, 'Laptop Lenovo', 'Informatique', 1200, 50, 4.5);

INSERT INTO products (product_id, name, category, price, stock,
ratings)
VALUES (2, 'iPhone 14', 'Smartphones', 999, 30, 4.7);
```

Etape 5 :
MongoDB :

```
db.products.find({ product_id: 1 });
```

DynamoDB :

```
aws dynamodb get-item --table-name Products --key '{"product_id": {"N":
"1"}}'
```

Cassandra :

```
SELECT * FROM products WHERE product_id = 1;
```

Etape 6 :
MongoDB :

```
db.products.find({ category: "Smartphones" });
```

DynamoDB :

```
aws dynamodb scan --table-name Products --filter-expression "category =
:cat" --expression-attribute-values '{":cat": {"S": "Smartphones"}}'
```

Cassandra :

```
SELECT * FROM products WHERE category = 'Smartphones' ALLOW FILTERING;
```

Etape 7 :
MongoDB :

```
db.products.updateOne({ product_id: 1 }, { $set: { stock: 40 } });
```

DynamoDB :

```
aws dynamodb update-item --table-name Products --key '{"product_id":
{"N": "1"}}' \
--update-expression "SET stock = :new_stock" \
--expression-attribute-values '{":new_stock": {"N": "40"}}'
```

Cassandra :

```
UPDATE products SET stock = 40 WHERE product_id = 1;
```

Etape 8 :
MongoDB :

```
db.products.deleteOne({ product_id: 1 });
```

DynamoDB :

```
aws dynamodb delete-item --table-name Products --key '{"product_id":
{"N": "1"}}'
```

Cassandra :

```
DELETE FROM products WHERE product_id = 1;
```

Etape 9 :
MongoDB :

```
db.products.find().sort({ price: -1 });
```

DynamoDB :
DynamoDB ne prend pas en charge le tri direct avec scan, il faut le gérer côté application.

Cassandra :

```
SELECT * FROM products ORDER BY price DESC ALLOW FILTERING;
```