**Exam Raphael L3 IA**

**Rapport : Pipeline de données avec Spark et Kafka**

**1. Configuration de l'environnement**

**1.1. Installation des outils**

**PySpark** : Installation via **pip install pyspark**. Cela permet d'utiliser Spark avec Python.

**Confluent Kafka** : Installation via **pip install confluent\_kafka** pour la communication avec Kafka.

**Aiven Kafka** : Utilisation d'Aiven pour éviter la configuration locale complexe de Kafka. Les fichiers de certificats SSL nécessaires ont été téléchargés depuis le tableau de bord Aiven.

**1.2. Vérification des certificats**

Avant de configurer Kafka, les fichiers SSL ont été vérifiés avec os.path.exists()pour garantir leur disponibilité :

**ca.pem : Certificat de l'autorité de certification.**

**service.cert : Certificat client.**

**service.key : Clé privée client.**

**1.3. Connexions et vérifications**

Une connexion réussie a été établie avec Aiven Kafka.

Le sujet utilisé par Kafka est nommé **Goro**.

**2. Pipeline ETL avec Spark (traitement par lots)**

**2.1. Extraction**

Un fichier CSV contenant les informations des étudiants a été chargé dans un DataFrame Spark :

**Colonnes : nom, prenom, age, filiere.**

Données d'entrée : Le fichier contient des informations structurées sur les étudiants.

**2.2. Transformations appliquées**

**Conversion des noms en majuscules** : Utilisation de **upper(col("nom"))**pour uniformiser les données.

**Filtrage des étudiants ayant plus de 20 ans** : Utilisation de **.filter(col("age") > 20)**.

**Calcul de la moyenne d'âge par filière** : Utilisation **de groupBy("filiere").agg(round(avg("age"), 1).alias("moyenne\_age")).**

**2.3. Chargement**

Les données transformées ont été sauvegardées dans un fichier CSV pour validation locale :

**Emplacement local : /content/students\_transformed.csv**.

Format : CSV avec en-têtes.

**3. Intégration Kafka et Spark Streaming**

**3.1. Producteur Kafka**

Les données brutes (nom, prénom, âge, filière) de la **Dataset non transformée et non filtrées** dont l’emplacement local **: /content/students.csv**. sont envoyées vers le sujet Goro.

Utilisation de **Producer** avec la configuration SSL pour **publier les messages.**

**3.2. Consommateur Kafka**

**Le consommateur** est configuré pour **lire les messages du sujet Goro**.

Les messages sont transformés en **DataFrame** Spark pour être transformés, filtrés et sauvegardés.

**3.3. Sauvegarde des données consommées**

Les données consommées sont sauvegardées dans un fichier CSV à l'emplacement suivant :

**Chemin : /content/students\_transformed\_by\_kafka.csv.**

Les colonnes utilisées sont identiques : nom, prenom, age, filiere.

**4. Analyses et statistiques**

**4.1. Transformations réalisées**

Uniformisation des noms en majuscules.

Filtrage des étudiants selon l'âge.

Calcul de statistiques (moyenne d'âge par filière).

**4.2. Défis rencontrés**

**Gestion des certificats SSL :**

Problème : Difficulté initiale à localiser et configurer les certificats pour Kafka.

Solution : Validation des chemins avec os.path.exists()avant exécution.

**Limitation des données sur Colab :**

Problème : Environnement Google Colab limitant les fichiers persistants.

Solution : Utilisation de Google Drive pour sauvegarder les fichiers produits.

**Intégration Spark Streaming :**

Problème : Configuration des flux en temps réel.

Solution : Utilisation de confluent\_kafkapour tester les messages, intégration en cours avec Spark Streaming.