Contrôle : Big Data

Partie 1 : Questions de compréhension

Q1. Expliquez ce qu'est **Apache Spark** et en quoi il diffère des outils classiques comme **Pandas** ou **Excel** pour le traitement de données massives.

Q2. Donnez les rôles des éléments suivants dans l'initialisation d'une session Spark :

```
SparkSession.builder.appName("TP").config("spark.driver.memory",
"2g").getOrCreate()
```

Q3. Lorsqu'on lit un fichier CSV dans PySpark, expliquez les effets des options suivantes :

- `.option("header", True)`
- `.option("inferSchema", True)`
- `.option("sep", ";")`

Q4. Que signifie `nullable = true` dans le schéma d'un DataFrame Spark ? Pourquoi est-ce important ?

Q5. Expliquez la différence entre les fonctions `filter()` et `groupBy().agg()` dans Spark. Donnez un exemple d'utilisation de chaque.

Q6. Qu'est-ce qu'une UDF dans PySpark ? Donnez un exemple d'utilisation simple.

Q7. En NLP avec PySpark :

- Que fait un `Tokenizer` ?
- À quoi sert `explode()` ?
- Quelle est la différence entre `orderBy(desc(...))` et `sort(...)` ?

Partie 2 : Manipulation de données avec PySpark

Vous disposez du fichier `students.csv` suivant :

```
id;name;age;grade
1;Ali;22;16
2;Sara;24;12
3;Ahmed;23;14
4;Rachida;22;10
5;Adil;24;18
6;Fatima;21;9
7;Khalid;23;17
```

Q1. Écrivez le code complet pour :

- Démarrer une session Spark
- Lire le fichier `students.csv`

Contrôle : Big Data

- Afficher les 5 premières lignes et le schéma

Q2. Filtrez les étudiants ayant une note **strictement supérieure à 13** et affichez leur nom et note.

Q3. Calculez la **moyenne des notes** par **âge**, triez les résultats par âge croissant.

Q4. Créez une **UDF** pour classer les étudiants selon leur note :

- A : grade > 15
- B : 13 < grade ≤ 15
- C : grade ≤ 13
  Ajoutez une colonne `"catégorie"` au DataFrame.

Q5. Créez un second DataFrame contenant les cours suivants :

```
courses_df = spark.createDataFrame([(1, "Math"), (2, "Bio"), (3, "Physique")], ["id", "cours"])
```

Effectuez une **jointure** avec le DataFrame des étudiants, en vous basant sur la colonne `"id"`. Affichez les résultats.

Q6. Repartitonnez les données sur 3 partitions. Expliquez pourquoi c'est utile en Big Data.

Partie 3 : Étude de texte

Fichier fourni : `discours.txt`

Q1. Chargez le fichier texte `discours.txt` dans un DataFrame et affichez son contenu.

Q2. Appliquez les étapes suivantes :

- Tokenisation du texte
- Transformation avec `explode()` pour avoir un mot par ligne

Q3. Supprimez les mots suivants du texte :
`["et", "la", "les", "des", "de", "en", "du", "un", "une", "pour"]`
Astuce : utilisez `.filter(~col("mot").isin([...]))`

Q4. Comptez les occurrences de chaque mot et affichez les **5 mots les plus fréquents**.

Q5. Enregistrez les résultats de la fréquence des mots dans un fichier CSV appelé `"frequence.csv"`.

Partie 4 : Traitement d'images avec PySpark

Contrôle : Big Data

Le dossier `images/` contient plusieurs fichiers `.jpg`.

Q1. Création de la session Spark

Écrivez le code pour :

- Importer les bibliothèques nécessaires.
- Créer une session Spark nommée `"Analyse d'Images"` avec 4 Go de mémoire pour le driver.

Q2. Chargement des fichiers binaires

Utilisez `spark.read.format("binaryFile")` pour :

- Charger uniquement les fichiers `.jpg` du dossier `images/`.
- Afficher le schéma du DataFrame obtenu.

Q3. Extraction des métadonnées

À partir du DataFrame précédent, extrayez les colonnes suivantes :

- Chemin complet du fichier (`chemin`)
- Taille du fichier en Ko (`taille_ko`)
- Date de modification (`modificationTime`)

Astuce : Utilisez `input_file_name()`, `col("length") / 1024`, et `.alias(...)`.

Q4. Filtrage des images volumineuses

Filtrez les images dont la taille est **supérieure ou égale à 500 Ko**. Affichez les 5 premières lignes.

Q5. Sauvegarde et export

Sauvegardez uniquement la colonne `chemin` des images filtrées dans un fichier CSV nommé `images_analysées`.

Le fichier doit écraser l'ancien s'il existe.

Q6. Statistiques de répartition des tailles

Générez un tableau statistique regroupant les images selon leur taille (en Ko) et comptant le nombre d'occurrences pour chaque taille.

- Trie les tailles du plus fréquent au moins fréquent.
- Affiche le résultat.