Rapport : Démarche - Adaptation et Intégration du fichier CO2.csv dans la table catalogue - HADOOP MAP REDUCE

Ce rapport détaille la démarche pour l'adaptation et l'intégration du fichier CO2.csv dans la table catalogue d'un concessionnaire automobile. L'objectif principal de cette intégration est d'enrichir la table catalogue avec de nouvelles informations sur les émissions de CO2, les coûts d'énergie et les valeurs de Bonus/Malus pour chaque marque. Cependant, pour obtenir ces données, il est nécessaire de nettoyer et de transformer les informations contenues dans CO2.csv, qui sont initialement réparties par marque et modèle de voiture, et non par marque. De plus, le fichier présente des données manquantes et des problèmes de format.

Nous avons choisi d'utiliser Apache Spark pour le nettoyage et la transformation des données (via MapReduce), ainsi que Hive pour intégrer ces données dans la table catalogue.

1. Chargement des Données

Démarrage de HDFS :

Pour démarrer HDFS, exécuter :

```
start-dfs.sh
```

Définition du chemin du répertoire HDFS :

Pour définir le répertoire où sont stockées les sources de données ainsi que le répertoire de destination des fichiers dans HDFS, utiliser la commande suivante :

```
export DATAHOME=/vagrant/MBDS_GRP5/DATA_EXTRACTOR/dataSources
export HDFSHOME=/vagrant/MBDS_GRP5/DATA_EXTRACTOR/hdfs
```

Création d'un répertoire "input" dans HDFS et transfert du fichier CO2.csv :

Créer un répertoire dans HDFS pour stocker le fichier CO2.csv et le transférer depuis le répertoire local :

```
hadoop fs -mkdir -p /user/vagrant/input
hadoop fs -put $DATAHOME/CO2.csv /user/vagrant/input/CO2.csv
```

2. Nettoyage et transformation des données (Map Reduce) avec Spark

Le script Spark suivant nettoie les caractères spéciaux, corrige les marques mal orthographiées, remplace les valeurs manquantes et prépare les données pour l'intégration en utilisant un traitement de Map Reduce.

Code source du programme Spark : `clean_map_reduce_co2.py` (Chemin dans le projet : `/DATA_EXTRACTOR/hdfs/clean_map_reduce_co2.py`)

Pour adapter le fichier CO2.csv avec la table catalogue, il doit être nettoyé et transformé via Map Reduce pour obtenir les émissions de CO2, les coûts d'énergie et les valeurs de Bonus/Malus par marque.

a. **Chargement des données :** Les données sont chargées depuis un fichier CSV spécifié dans un RDD à l'aide de la méthode **textFile** de Spark.

```
from pyspark import SparkContext
sc = SparkContext("local[*]", "NettoyageMapReduceCO2")
# Charger les données depuis le fichier CSV
co2_rdd = sc.textFile('/user/vagrant/input/CO2.csv')
```

b. **Nettoyage des caractères spéciaux :** Une fonction est définie pour corriger les caractères spéciaux et les fautes dans le fichier CSV. Chaque ligne nettoyée est ensuite divisée en colonnes en utilisant la virgule (‘,’) comme séparateur.

```
# Fonction pour nettoyer les caractères spéciaux et corriger les marques
mal orthographiées
def nettoyer_caracteres(line):
    # Remplacer les caractères spéciaux et les fautes typographiques
    cleaned = line.replace('Ã\xa0 aimant permanent,', 'à aimant permanent')
    cleaned = cleaned.replace('\xa0', '')  # Enlever les espaces non
conventionnels
    cleaned = cleaned.replace('Ã©', 'é')  # Corriger les caractères
accentués
    cleaned = cleaned.replace('€1', '€')  # Corriger le symbole de l'euro
    cleaned = cleaned.replace('€', '')  # Retirer le symbole de l'euro
    cleaned = cleaned.replace('Ã¨', 'è')  # Corriger les caractères
accentués
    return cleaned
# Nettoyer les lignes du CSV et les diviser en colonnes
lignes_propres_rdd = co2_rdd.map(nettoyer_caracteres).map(lambda line:
line.split(","))
```

c. **Correction des marques et suppression des lignes vides :** Une fonction est créée pour corriger les noms de certaines marques et filtrer les lignes vides du jeu de données.

```
# Fonction pour corriger les marques spécifiques et retirer les lignes
vides
def corriger_marque(ligne):
    marque = ligne[1].split(' ')[0]  # Utiliser le premier mot de la marque
    if marque == 'LAND':
        marque = 'LAND ROVER'
    elif marque in ['"VOLKSWAGEN', '"KIA']:
        marque = marque.replace('"', '')  # Enlever les guillemets
    return [ligne[0], marque, ligne[2], ligne[3], ligne[4]]
# Corriger les marques
marques_corrigees_rdd =
lignes_propres_rdd.map(corriger_marque).filter(lambda x: x[0] != '')
```

d. **Conversion des valeurs en entiers :** Une fonction est définie pour convertir les valeurs numériques en entiers, en remplaçant les valeurs manquantes par 0.

```
# Fonction pour convertir une chaîne en entier ou renvoyer 0 s'il y a une
valeur manquante
def convertir_en_entier(value):
    return int(value) if value != '-' else 0
# Convertir les valeurs appropriées en entiers
entiers_rdd = marques_corrigees_rdd.map(lambda x: (x[1],
convertir_en_entier(x[2]), convertir_en_entier(x[3]), int(x[4])))
```

e. **Regroupement par marque et calcul des moyennes finales (Map Reduce) :** Les données sont regroupées par marque (Map Reduce) en sommant les colonnes pertinentes (malusBonus, rejetsCO2, coutEnergie) et en comptant le nombre d'occurrences. Ensuite, les moyennes finales sont calculées pour chaque marque, en divisant chaque colonne par le nombre d'occurrences de la marque.

```
# Réduire par clé (marque) pour obtenir les sommes des colonnes
aggregated_rdd = entiers_rdd.map(lambda x: (x[0], (x[1], x[2], x[3], 1)))
reduced_rdd = aggregated_rdd.reduceByKey(lambda x, y: (x[0] + y[0], x[1] +
y[1], x[2] + y[2], x[3] + y[3]))
# Calculer les moyennes finales pour chaque marque
moyennes_rdd = reduced_rdd.map(lambda x: (x[0], x[1][0] / x[1][3], x[1][1]
/ x[1][3], x[1][2] / x[1][3]))
```

f. **Conversion des résultats et enregistrement :** Les résultats finaux sont convertis en chaînes de caractères et enregistrés dans un fichier texte à l'emplacement spécifié (**/user/vagrant/output/clean_co2**).

```
# Convertir les résultats en string pour l'enregistrement
resultats_string = moyennes_rdd.map(lambda x:
f"{x[0]},{x[1]},{x[2]},{x[3]}")
# Enregistrer les résultats dans des fichiers texte
resultats_string.saveAsTextFile('/user/vagrant/output/clean_co2')
```

Exécution du script Spark :

Pour exécuter ce script et obtenir le résultat attendu, utiliser la commande suivante :

```
spark-submit $HDFSHOME/clean_map_reduce_co2.py
```

Ainsi, le résultat, contenant les émissions de CO2, les coûts d'énergie et les valeurs de Bonus/Malus par marque, est stocké dans le dossier **/user/vagrant/output/clean_co2** dans HDFS.

3. Création de Table Externe dans Hive

Après avoir nettoyé et transformé les données, nous allons créer des tables externes et intégrer ces données dans le catalogue dans Hive.

Création de la table externe `co2_ext` :

Cette table externe pointe vers les données nettoyées et transformées stockées dans HDFS.

```
CREATE EXTERNAL TABLE IF NOT EXISTS co2_ext (
    marque STRING,
    malusBonus FLOAT,
    rejetsCO2 FLOAT,
    coutEnergie FLOAT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','

STORED AS TEXTFILE
LOCATION '/user/vagrant/output/clean_co2';
```

4. Intégration avec la Table Catalogue

Nettoyage et préparation de la table catalogue_ext :

Pour intégrer la table **co2_ext** avec la table **catalogue_ext**, il est nécessaire de commencer par nettoyer certaines données de la table catalogue existante pour créer une table intermédiaire nommée **cleaned_catalogue_ext**.

```
CREATE TABLE cleaned_catalogue_ext AS
SELECT
    id,
    CASE
        WHEN marque LIKE '%Hyunda%' THEN 'Hyundai'
        ELSE marque
    END AS marque,
    nom,
    puissance,
    longueur,
    nbplaces,
    nbportes,
    couleur,
    occasion,
    prix
FROM catalogue_ext;
```

Création de la table catalogue_co2 et intégration des données :

Ensuite, effectuer une jointure entre **cleaned_catalogue_ext** et **co2_ext** pour intégrer les données des émissions de CO2, des coûts d'énergie, et des valeurs de Bonus/Malus par marque avec les données du catalogue.

Lors de la jointure, pour les marques de voitures qui ne sont pas présentes dans **co2_ext**, insérer la moyenne des colonnes **malusBonus**, **rejetsCO2** et **coutEnergie** de toutes les marques de véhicules présentes dans les deux tables (**cleaned_catalogue_ext** et **co2_ext**).

Pour ce faire, voici les étapes à suivre:

1. Identifier les marques communes aux deux tables.
2. Calculer les moyennes des colonnes correspondantes dans **co2_ext** pour ces marques.
3. Utiliser ces moyennes pour remplacer les valeurs des marques manquantes dans **cleaned_catalogue_ext**.
4. Créer la table **catalogue_co2** en intégrant ces données.

```
CREATE TABLE IF NOT EXISTS catalogue_co2
AS
WITH marques_communs AS (
    SELECT DISTINCT co2.marque, co2.malusBonus, co2.rejetsCO2, co2.coutEnergie
    FROM co2_ext co2
    JOIN cleaned_catalogue_ext cat
    ON LOWER(co2.marque) = LOWER(cat.Marque)
),
moyennes_co2 AS (
    SELECT
        AVG(co2.malusBonus) AS avg_malusBonus,
        AVG(co2.rejetsCO2) AS avg_rejetsCO2,
        AVG(co2.coutEnergie) AS avg_coutEnergie
    FROM co2_ext co2
    JOIN marques_communs communs
    ON LOWER(co2.marque) = LOWER(communs.marque)
)
SELECT
    cat.*,
    COALESCE(co2.malusBonus, moyennes_co2.avg_malusBonus) AS malusBonus,
    COALESCE(co2.rejetsCO2, moyennes_co2.avg_rejetsCO2) AS rejetsCO2,
    COALESCE(co2.coutEnergie, moyennes_co2.avg_coutEnergie) AS coutEnergie
FROM cleaned_catalogue_ext cat
LEFT JOIN co2_ext co2
ON LOWER(co2.marque) = LOWER(cat.Marque)
CROSS JOIN moyennes_co2;
```

Résultat final

Cette démarche pour l'adaptation et l'intégration des données du fichier CO2.csv a permis d'enrichir la table catalogue (**catalogue_co2**) avec les informations sur les émissions de CO2, les coûts d'énergie et les valeurs de Bonus/Malus de chaque marque. Ces données pourraient potentiellement améliorer la qualité des modèles prédictifs lors des étapes ultérieures d'analyse de données.