Devoir Spark

1. **Télécharment des deux fichiers de données**

   wget https://cadastre.data.gouv.fr/data/etalab-dvf/latest/csv/2019/full.csv.gz
   wget https://www.data.gouv.fr/fr/datasets/r/b3b26ad1-a143-4651-afd6-dde3908196fc

2. **Extraction du fichier gz**

   gunzip full.csv.gz

3. **Renommage des fichiers**

   mv full.csv dmde_valeur_fonciere.csv
   mv b3b26ad1-a143-4651-afd6-dde3908196fc Etablissements_Geoloc.csv

4. **Chargement des deux fichiers dans spark dans les dataframes**

   dvf=spark.read.option("header","true").option("delimiter",",").csv("/home/ec2-user/dmde_valeur_fonciere.csv",inferSchema=True)

   df_etablissement=spark.read.option("header","true").option("delimiter",";").csv("/home/ec2-user/Etablissements_Geoloc.csv",inferSchema=True)

5. **Impression des schemas pour comprendre la structure des fichiers**

   dvf.printSchema()

```
>>> dvf.printSchema()
root
 |-- id_mutation: string (nullable = true)
 |-- date_mutation: timestamp (nullable = true)
 |-- numero_disposition: integer (nullable = true)
 |-- nature_mutation: string (nullable = true)
 |-- valeur_fonciere: double (nullable = true)
 |-- adresse_numero: integer (nullable = true)
 |-- adresse_suffixe: string (nullable = true)
 |-- adresse_nom_voie: string (nullable = true)
 |-- adresse_code_voie: string (nullable = true)
 |-- code_postal: integer (nullable = true)
 |-- code_commune: string (nullable = true)
 |-- nom_commune: string (nullable = true)
 |-- code_departement: string (nullable = true)
 |-- ancien_code_commune: integer (nullable = true)
 |-- ancien_nom_commune: string (nullable = true)
 |-- id_parcelle: string (nullable = true)
 |-- ancien_id_parcelle: string (nullable = true)
 |-- numero_volume: string (nullable = true)
 |-- lot1_numero: string (nullable = true)
 |-- lot1_surface_carrez: double (nullable = true)
 |-- lot2_numero: string (nullable = true)
 |-- lot2_surface_carrez: double (nullable = true)
```

```
 |-- lot3_numero: string (nullable = true)
 |-- lot3_surface_carrez: double (nullable = true)
 |-- lot4_numero: integer (nullable = true)
 |-- lot4_surface_carrez: double (nullable = true)
 |-- lot5_numero: integer (nullable = true)
 |-- lot5_surface_carrez: double (nullable = true)
 |-- nombre_lots: integer (nullable = true)
 |-- code_type_local: integer (nullable = true)
 |-- type_local: string (nullable = true)
 |-- surface_reelle_bati: integer (nullable = true)
 |-- nombre_pieces_principales: integer (nullable = true)
 |-- code_nature_culture: string (nullable = true)
 |-- nature_culture: string (nullable = true)
 |-- code_nature_culture_speciale: string (nullable = true)
 |-- nature_culture_speciale: string (nullable = true)
 |-- surface_terrain: integer (nullable = true)
 |-- longitude: double (nullable = true)
 |-- latitude: double (nullable = true)

>>> df_etablissement.printSchema()
root
 |-- Code établissement: string (nullable = true)
 |-- Appellation officielle: string (nullable = true)
 |-- Dénomination principale: string (nullable = true)
 |-- Patronyme uai: string (nullable = true)
 |-- Secteur Public/Privé: string (nullable = true)
 |-- Adresse: string (nullable = true)
 |-- Lieu dit: string (nullable = true)
 |-- Boite postale: string (nullable = true)
 |-- Code postal: integer (nullable = true)
 |-- Localite d'acheminement: string (nullable = true)
 |-- Commune: string (nullable = true)
 |-- Coordonnee X: double (nullable = true)
 |-- Coordonnee Y: double (nullable = true)
 |-- EPSG: string (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- Qualité d'appariement: string (nullable = true)
 |-- Localisation: string (nullable = true)
 |-- Code nature: integer (nullable = true)
 |-- Nature: string (nullable = true)
 |-- Code état établissement: integer (nullable = true)
 |-- Etat établissement: string (nullable = true)
 |-- Code département: string (nullable = true)
 |-- Code région: integer (nullable = true)
 |-- Code académie: integer (nullable = true)
 |-- Code commune: string (nullable = true)
 |-- Département: string (nullable = true)
```

|-- Région: string (nullable = true)
|-- Académie: string (nullable = true)
|-- Position: string (nullable = true)
|-- secteur_prive_code_type_contrat: integer (nullable = true)
|-- secteur_prive_libelle_type_contrat: string (nullable = true)

6. Approche de corrélation

Une approche de corrélation c'est de pouvoir par exemple présenter par commune le nombre d'établissement, le nombre total de lot vendu, la valeur foncière moyenne dans la commune, la superficie moyenne du terrain.

7. Implémentation de la solution

Première Etape : **regroupement des données sur les valeurs foncières**

Regroupement des données sur la demande de valeur foncière par commune. Par exemple par Commune le nombre total de lot vendu, la superficie totale vendue, la moyenne de la valeur foncière

dvf_agg=dvf.groupBy("code_commune","nom_commune").agg({"valeur_fonciere": "avg","nombre_lots":"sum","surface_terrain":"sum"})

Deuxième Etape : regroupement des données sur les établissements

Regroupement des données sur les établissements par commune, ici nous allons nous intéresser au nombre d'établissement présent dans chaque commune.

df_etablissement_agg=df_etablissement.groupBy("Code commune","Commune").agg({"Code établissement":"count"})

Troisième étape : **Jointure gauche sur le code_commune**

jointure gauche des deux nouveaux dataframes obtenus sur le champ code_commune

df = dvf_agg.join(df_etablissement_agg, dvf_agg.code_commune == df_etablissement_agg["Code commune"],how='left'

Quatrième étape : **Nettoyage des données**

Nettoyage des valeurs null En fait la valeur null apparait après la jointure lorsque nous avons des codes commune pour lesquels nous n'avons pas d'établissements. Au lieu de la valeur null nous allons mettre zero. De plus nous allons supprimer les colonnes redondantes issue de la jointure.

df_final=df.drop('Code commune','Commune').fillna({'count(Code établissement)':0})

Sixième étape : Enregistrement du dataframe sur disque en format CSV

df_final.write.csv("/home/ec2-user/dvf_correlation.csv",header=True)

```
>>> df_final.show()
+------------+--------------------+--------------------+--------------------+---------------+------------------------+
|code_commune|        nom_commune|avg(valeur_fonciere)|sum(surface_terrain)|sum(nombre_lots)|count(Code établissement)|
+------------+--------------------+--------------------+--------------------+---------------+------------------------+
|       01431|       Vaux-en-Bugey|  231819.92727272728|               33942|              1|                       1|
|       01424|            Tramoyes|            135000.0|                 478|              0|                       1|
|       02074|Bertaucourt-Epourdon|             33000.0|                2248|              0|                       1|
|       02755|               Urcel|             77300.0|                7409|              0|                       1|
|       07128|           Lalouvesc|             95725.0|                2020|              8|                       1|
|       08298|   Montcy-Notre-Dame|             87290.0|                6048|              0|                       1|
|       08336|               Osnes|  60346.666666666664|                2294|              0|                       0|
|       09161|              Léran|   70936.53846153847|               46562|              0|                       1|
|       09247|         Rivèrenert|               250.5|                 132|              0|                       0|
|       10038|   Bercenay-le-Hayer|            135000.0|                2053|              0|                       0|
|       11076|       Castelnaudary|  147570.43364341088|               52057|             22|                      15|
|       14543|                Rots|             32140.5|               10427|              0|                       2|
|       14009|          Amfreville|            138000.0|                 427|              0|                       1|
|       16014|             Angeduc|  1433.3333333333333|                2002|              0|                       0|
|       17443|              Thairé|  124886.30769230769|               18795|              0|                       1|
|       17171|                Geay|            288750.0|               15905|              0|                       1|
|       17350|Saint-Julien-de-l...|  65423.333333333336|                4392|              0|                       2|
|       18109|        Henrichemont|   64511.07692307692|               73389|              0|                       2|
|       18111|          Humbligny|  34354.770000000004|                8370|              0|                       0|
|       19214|Saint-Julien-aux-...|   45828.04878048781|              161296|              0|                       1|
+------------+--------------------+--------------------+--------------------+---------------+------------------------+
only showing top 20 rows
```

8. **Ecriture du job spark**

Le job Spark est obtenu en rassemblant toutes les lignes de code précédentes dans un fichier python. Après avoir bien évidement construit l'objet représentant la session spark.

```
# coding: utf-8
#chargement des données.
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("dvf_correlation").getOrCreate()
dvf=spark.read.option("header","true").option("delimiter",",").csv("/home/ec2-user/dmde_valeur_fonciere.csv",inferSchema=True)
df_etablissement=spark.read.option("header","true").option("delimiter",";").csv("/home/ec2-user/Etablissements_Geoloc.csv",inferSchema=True)

#Aggrégation des données
dvf_agg=dvf.groupBy("code_commune","nom_commune").agg({"valeur_fonciere": "avg","nombre_lots":"sum","surface_terrain":"sum"})
df_etablissement_agg=df_etablissement.groupBy("Code commune","Commune").agg({"Code établissement":"count"})

#jointure sur les deux nouveau dataframe
df = dvf_agg.join(df_etablissement_agg, dvf_agg.code_commune == df_etablissement_agg["Code commune"],how='left')

#Nettoyage
df_final=df.drop('Code commune','Commune').fillna({'count(Code établissement)':0})

#Save to file
df_final.write.csv("/home/ec2-user/dvf_correlation.csv",header=True)

#reload de written dataframe
#test=spark.read.option("header","true").option("delimiter",",").csv("/home/ec2-user/dvf_correlation.csv",inferSchema=True)
```