TP Spark et performance

Dans ce TP, on va réutiliser le code du TP 1 (tpspark.py).
On va chercher à améliorer le temps de calcul

Pour cela, on va soumettre un job submit du script en modifiant quelques éléments du code pour
1 - afficher le temps de calcul total du script
2 - tester le caching pour améliorer le temps de calcul

Manip 1 : modifier le script pour afficher le temps de calcul de l'ensemble du script
1 - importer le package 'time' en début de script :
import time

2 - Ajouter les codes suivants :
a - au début du main

```
if __name__ == '__main__':
    print('============================')
    start = time.time()
    print('START TIME : ', start)
    print('============================')
```

b - à la fin du main (à l'intérieur du 'if' initial) :

```
print('============================')
end = time.time()
print('END TIME : ', end)
print('DURATION : ', end - start)
print('============================')

input("press ctrl+c to exit")
```

Note : le input(…) va permettre de maintenir la session spark active, pour avoir accès au sparkUI

Manip 2 : Soumettre un job spark et calculer le temps de référence

1 - Si les modifications du script ont été faites en local, pensez à envoyer le code sur votre sandbox.
Pour cela, lancer un terminal, et taper :

```
scp -P 2222 chemin/vers/script_locale.py root@localhost:/chemin/sandbox/
```

2 - Connectez vous à la sandbox :

```
ssh -p 2222 root@localhost
```

4 - Ajouter un paramètre d'encoding en tapant dans la console :

```
export PYTHONIOENCODING=utf8
```

5 - Lancer un job spark avec le script précédent, en tapant dans la console :

```
spark-submit chemin/sandbox/vers/fichier.py
```

6 - En même temps, vous pouvez jeter un oeil au SparkUI pour regarder les calculs effectués par Spark. Pour cela, ouvrez un navigateur, et entrez l'URL suivant :

```
localhost:4040
```

7 - De retour dans le terminal où vous avez lancé le job spark, attendez que l'exécution touche à sa fin, et lire le temps de calcul du script

Manip 3 : optimiser le temps de calcul de spark
Pour cela, on va utiliser la fonction cache pour sauvegarder en mémoire certaines dataframes

La commande est la suivante :

```
df.cache()
```

1 - A vous de jouer pour mettre en cache le(s) dataframe(s) qui vous semblent pertinents !
2 - Pensez à surveiller le SparkUI pour voir ce que fait votre mise en cache