TP DataBrciks (Pyspark)

Lib en Scala

```
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._
import org.apache.spark.storage.StorageLevel
import spark.sqlContext.implicits
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.expressions._
```

EXO1

1) **def reader (path):DataFrame**

Lire le fichier CSV et retourne un Dataframe, avec les options suivantes

a) Séparateur = "#"

b) header true

c) Utilise la fonction **withColumn** et **substring_index** afin de créer la colonne TVA & HTT

Data: https://github.com/idiattara/Airflow_THIES_SORBONE/blob/main/data_exo1.csv

```
display(reader(path))
```

(2) Spark Jobs

Table

	Id_Client	HTT_TVA	MetaData	HTT	TVA
1	1	100,5|0,19	{"MetaTransaction":[{"Ville":"Paris","Date_End_contrat":"2020-12-23 00:00:00"},{"TypeProd":"Laitier","produit":["yaourt","laitcoco"]}]}	100,5	0,19
2	2	120,546|0,20	{"MetaTransaction":[{"Ville":"Alger","Date_End_contrat":"2023-12-24 00:00:00"},{"TypeProd":"Boison","produit":["coca","fanta"]}]}	120,546	0,20
3	3	123,6|0,201	{"MetaTransaction":[{"Ville":"Dakar","Date_End_contrat":"2020-10-24 00:00:00"},{"TypeProd":"Laitier","produit":["yaourt"]}]}	123,6	0,201
4	4	5,546|0,15	{"MetaTransaction":[{"Ville":"Abidjan","Date_End_contrat":"2028-01-23 00:00:00"},{"TypeProd":"Laitier","produit":["laitcoco"]}]}	5,546	0,15

2) def calculTTC (DataFrame) : DataFrame

- ✓ Calcule le TTC => HTT+TVA*HTT, le TTC doit être arrondi à 2 chiffres après la virgule
- ✓ Pensez à remplacer ", " par ". " afin de caster en double TVA et HTT
- ✓ Supprime la colonne TVA, HTT

```
display(CalculTTC(reader(path)))
```

(2) Spark Jobs

Table

	Id_Client	HTT_TVA	MetaData	1.2 TTC
1	1	100,5|0,19	{"MetaTransaction":[{"Ville":"Paris","Date_End_contrat":"2020-12-23 00:00:00"},{"TypeProd":"Laitier","produit":["yaourt","laitcoco"]}]}	119.6
2	2	120,546|0,20	{"MetaTransaction":[{"Ville":"Alger","Date_End_contrat":"2023-12-24 00:00:00"},{"TypeProd":"Boison","produit":["coca","fanta"]}]}	144.66
3	3	123,6|0,201	{"MetaTransaction":[{"Ville":"Dakar","Date_End_contrat":"2020-10-24 00:00:00"},{"TypeProd":"Laitier","produit":["yaourt"]}]}	148.44
4	4	5,546|0,15	{"MetaTransaction":[{"Ville":"Abidjan","Date_End_contrat":"2028-01-23 00:00:00"},{"TypeProd":"Laitier","produit":["laitcoco"]}]}	6.38

3) def Extract_Date_End_Contrat_Ville (DataFrame) : DataFrame

- ✓ Créer une nouvelle colonne **Date_End_contrat et Ville** en utilisant la méthode select **from_json** et **regexp_extract** pour extraire YYYY-MM-DD
- ✓ Supprime la colonne metaData

Code à compléter (Scala en python)

```
def Extract_Date_End_Contrat_Ville(df:DataFrame):DataFrame={
  val schema_MetaTransaction = new StructType()
                                    .add("Ville",StringType, false)
                                    .add("Date_End_contrat",StringType,false)
val schema = new StructType()
                .add("MetaTransaction", ArrayType(schema_MetaTransaction), true)
```

```
display(Extract_Date_End_Contrat_Ville((CalculTTC(reader(path)))))
```

(2) Spark Jobs

Table +

	Id_Client	HTT_TVA	1.2 TTC	Ville	Date_End_contrat
1	1	100,5|0,19	119.6	Paris	2020-12-23
2	2	120,546|0,20	144.66	Alger	2023-12-24
3	3	123,6|0,201	148.44	Dakar	2020-10-24
4	4	5,546|0,15	6.38	Abidjan	2028-01-23

4) def Contrat_Status (DataFrame) : DataFrame

- ✓ Créer un nouvelle colonne **Contrat_Status** avec "Expired" si le contrat a expiré et sinon "Actif"

Just now (3s) 14

```
display(Contrat_Status(Extract_Date_End_Contrat_Ville((CalculTTC(reader(path))))))
```

(2) Spark Jobs

Table +

	Id_Client	HTT_TVA	1.2 TTC	Ville	Date_End_contrat	Contrat_Status
1	1	100,5|0,19	119.6	Paris	2020-12-23	Expired
2	2	120,546|0,20	144.66	Alger	2023-12-24	Expired
3	3	123,6|0,201	148.44	Dakar	2020-10-24	Expired
4	4	5,546|0,15	6.38	Abidjan	2028-01-23	Actif

4 rows | 2.77s runtime

Exo2

Partie1 : Load Data

Question1 :

def readerjson(path):Seq[DataFrame]

Cette fonction lit un fichier **JSON** et retourne deux DataFrames, un qui contient les informations des transactions et un autre celles des devis

Data: https://github.com/idiattara/Airflow_THIES_SORBONE/blob/main/exo2.json

```
val dftransaction=readerjson(path)(0)
dftransaction.show
```

▸ (2) Spark Jobs

▸ dftransaction: org.apache.spark.sql.DataFrame = [Devis: string, IdTransaction: long ... 3 more fields]

```
+-----+-------------+--------+------+------------+
|Devis|IdTransaction|   Pays|  Prix| TypeProduit|
+-----+-------------+--------+------+------------+
|  EUR|            1| France|    20|     Laitier|
|  TND|            2|Tunisie|  5000|      Boison|
|  USD|            3|Senegal|   200|Electronique|
|  EUR|            4| France|   200|     Laitier|
|  EUR|            5| France|   205|      Boison|
|  TND|            6|Tunisie|   500|     Laitier|
|  TND|            7|Tunisie|145000|Electronique|
|  CFA|            8|Senegal|   200|Electronique|
+-----+-------------+--------+------+------------+
```

```
val dftauxdevis=readerjson(path)(1)
dftauxdevis.show
```

▸ (2) Spark Jobs

▸ dftauxdevis: org.apache.spark.sql.DataFrame = [Devis: string, Taux: double]

```
+-----+------+
|Devis|  Taux|
+-----+------+
|  EUR|   1.0|
|  USD|  0.88|
|  CFA|0.0015|
|  TND|  0.31|
+-----+------+
```

Partie 2 Agrégation simple

Question1 :

def NbCommandeParProduitDansPay ():DataFrame

Cette fonctionne calcule le **nombre** de commande de chaque produit pour chaque **pays**

```
NbCommandeParProduitDansPay().show
```

▸ (6) Spark Jobs

```
+------------+------+-------+-------+
| TypeProduit|France|Senegal|Tunisie|
+------------+------+-------+-------+
|       Boison|     1|      0|      1|
|      Laitier|     2|      0|      1|
|Electronique|     0|      2|      1|
+------------+------+-------+-------+
```

Partie3 Cross data

Question 1 :

def PrixEur ():DataFrame

La fonction fait la **jointure** du Dataframe **dftransaction et** et celui de **dftauxdevis** afin de convertir les prix en euro

```
PrixEur().show
```

▸ (1) Spark Jobs

```
+-----+-------------+-------+------+------------+------+-------+
|Devis|IdTransaction|   Pays|  Prix| TypeProduit|  Taux|PrixEur|
+-----+-------------+-------+------+------------+------+-------+
|  EUR|            1| France|    20|     Laitier|   1.0|   20.0|
|  TND|            2|Tunisie|  5000|      Boison|  0.31| 1550.0|
|  USD|            3|Senegal|   200|Electronique|  0.88|  176.0|
|  EUR|            4| France|   200|     Laitier|   1.0|  200.0|
|  EUR|            5| France|   205|      Boison|   1.0|  205.0|
|  TND|            6|Tunisie|   500|     Laitier|  0.31|  155.0|
|  TND|            7|Tunisie|145000|Electronique|  0.31|44950.0|
|  CFA|            8|Senegal|   200|Electronique|0.0015|    0.3|
+-----+-------------+-------+------+------------+------+-------+
```

Partie4 agrégation avec windows partition

Question1 :

def Top2TransactionPerPays ():DataFrame

Donne les informations des deux premières transactions qui ont génère plus d'argent pour chaque **Pays**

```
Top2TransactionPerPays().show
```

▸ (2) Spark Jobs

```
+-----+-------------+-------+------+------------+------+-------+----+
|Devis|IdTransaction|   Pays|  Prix| TypeProduit|  Taux|PrixEur|rank|
+-----+-------------+-------+------+------------+------+-------+----+
|  EUR|            5| France|   205|      Boison|   1.0|  205.0|   1|
|  EUR|            4| France|   200|     Laitier|   1.0|  200.0|   2|
|  USD|            3|Senegal|   200|Electronique|  0.88|  176.0|   1|
|  CFA|            8|Senegal|   200|Electronique|0.0015|    0.3|   2|
|  TND|            7|Tunisie|145000|Electronique|  0.31|44950.0|   1|
|  TND|            2|Tunisie|  5000|      Boison|  0.31| 1550.0|   2|
+-----+-------------+-------+------+------------+------+-------+----+
```

Partie 5 Agrégation Combinée

Question 1 :

def Cube ():DataFrame

Fait une agrégation combinée sur TypeProduit et Pays en utilisant le fonction **cube**

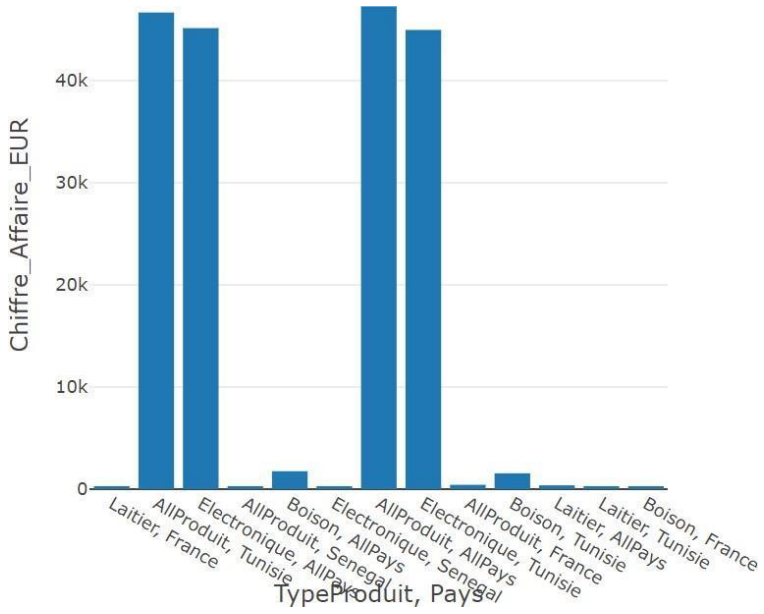
```
Cube().show
```

▸ (2) Spark Jobs

```
+-------+------------+------------------+
|   Pays| TypeProduit|Chiffre_Affaire_EUR|
+-------+------------+------------------+
| France|     Laitier|             220.0|
|Tunisie|  AllProduit|           46655.0|
|AllPays|Electronique|           45126.3|
|Senegal|  AllProduit|             176.3|
|AllPays|      Boison|            1755.0|
|Senegal|Electronique|             176.3|
|AllPays|  AllProduit|           47256.3|
|Tunisie|Electronique|           44950.0|
| France|  AllProduit|             425.0|
|Tunisie|      Boison|            1550.0|
|AllPays|     Laitier|             375.0|
|Tunisie|     Laitier|             155.0|
| France|      Boison|             205.0|
+-------+------------+------------------+
```

Question 2 :

a) Réaliser le graphe de la question Question1 en utilisant l'Api SQL



b) Tirez une conclusion

Exo2(Airflow)

Les Data Engineers de la Fab XFV souhaitent procéder à la préparation des logs afin que les data scientist puissent appliquer de l'analyse exploratoire

Description :

L'objectif est de parser logs issus du serveu 4G Prenium . Le log est constitué d'un timestamp puis d'une url

2017-10-05 00:01:09/map/1.0/slab/standard/256/19/263920/186677

- stanard represente **Breakdown_Type** (le nombre après slab)
- 19 reprsente **Breakdown_Level** (le nombre après 256)

Probelematique

- On souhaite savoir quels étaient le nombre successif de panne pour chaque **Breakdown_Type**
- Les niveaux de panne(**Breakdown_Level**) , séparés par des virgules, l'ordre n est pas obligatoire

https://github.com/idiattara/Airflow_THIES_SORBONE/blob/main/break_down.txt

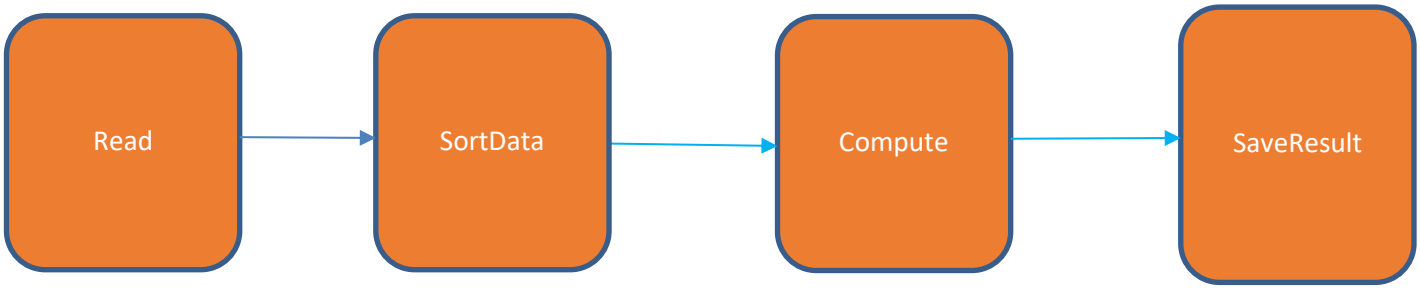
Exemple d'input trié, mais lorsque vous ferez vos développements, partez du principe que les données ne seront pas triées, car lorsque vous me présenterez votre solution, je ferai les tests sur un autre fichier avec des timestamps différents.

```
2017-10-05 00:01:09@/map/1.0/slab/standard/256/19/263920/186677
2017-10-05 00:01:09@/map/1.0/slab/standard/256/12/2056/1387
2017-10-05 00:01:09@/map/1.0/slab/standard/256/14/8338/5848
2017-10-05 00:01:09@/map/1.0/slab/standard/256/19/263921/186678
2017-10-05 00:01:09@/map/1.0/slab/traffic/256/14/8338/5850
2017-10-05 00:01:09@/map/1.0/slab/traffic_hd/256/12/2061/1387
2017-10-05 00:01:09@/map/1.0/multi-descr/standard_hd/256/13/7773,6273;7773,6274;7774;7778,6273;7778,6274
2017-10-05 00:01:09@/map/1.0/slab/traffic_hd/256/12/2060/1396
2017-10-05 00:01:09@/map/1.0/multi-descr/standard/256/14/8529,5662;8529,5663;8529,5664;8530,5662;8530,5663;8530
2017-10-05 00:01:09@/map/1.0/slab/standard_hd/256/14/8620/6047
2017-10-05 00:01:10@/map/1.0/slab/traffic/256/14/8336/5847
2017-10-05 00:01:10@/map/1.0/slab/traffic/256/17/134034/90800
2017-10-05 00:01:10@/map/1.0/slab/standard/256/19/260681/184714
2017-10-05 00:01:10@/map/1.0/slab/standard/256/17/67270/47296
2017-10-05 00:01:10@/map/1.0/slab/public_transport_hd/256/15/33165/22527
2017-10-05 00:01:10@/map/1.0/slab/standard/256/18/126515/84965
2017-10-05 00:01:10@/map/1.0/slab/standard/256/18/134034/89724
2017-10-05 00:01:10@/map/1.0/slab/standard/256/19/270890/191082
2017-10-05 00:01:10@/map/1.0/slab/standard/256/14/8248/6005
2017-10-05 00:01:10@/map/1.0/slab/standard_hd/256/18/132782/90227
```

Out

Breakdown_Type	Count_Successive	Breakdowns_Level
standard	4	19,12,14,19
traffic	1	14
traffic_hd	1	12
standard_hd	1	13
traffic_hd	1	12
standard	1	14
standard_hd	1	14
traffic	2	14,17
standard	2	19,17
public_transport_hd	1	15
standard	4	18,18,19,14
standard_hd	1	18

Architecture de votre Dags



1. La tâche **Read** lit vos données et retourne un DataFrame contenant deux colonnes : timestamp et log.
2. La tâche **SortData** trie les données par timestamp en ordre croissant.
3. La tâche **Compute** effectue les calculs demandés.
4. La tâche **SaveResult** sauvegarde le résultat dans un fichier CSV