Spark Streaming TD

PART I

Ouvrez le fichier des prénoms.

1. Combien de colonnes contient-il ? A quoi correspondent-elles ? Quels sont les types des données ?

On a 4 Colonnes :

Sexe (homme / femme)

Preusuel (le nom)

Annais (l'année de naissance)

Nombre (le nombre de nouveau née avec le nom Preusuel à l'année Annais)

2. Qualifiez les données. Il y a-t-il des données à éviter ?

Oui, les valeur XXXX sont absurde dans la colonne « annais »

PART 2

I.

1. Instanciez une SparkSession

```
val spark = SparkUtils.spark()
```

2. Créez un DataStream qui collecte la donnée à partir du Socket

```
//Créez un DataStream qui collecte la donnée à partir du Socket
var df = spark.readStream
  .format("socket")
  .option("host", "localhost")
  .option("port", 9999)
  .load()
```

3. Ecrivez votre job, qui ne fait pour l'instant qu'écrire les données du DataStream dans la console (Utilisez les différents modes)

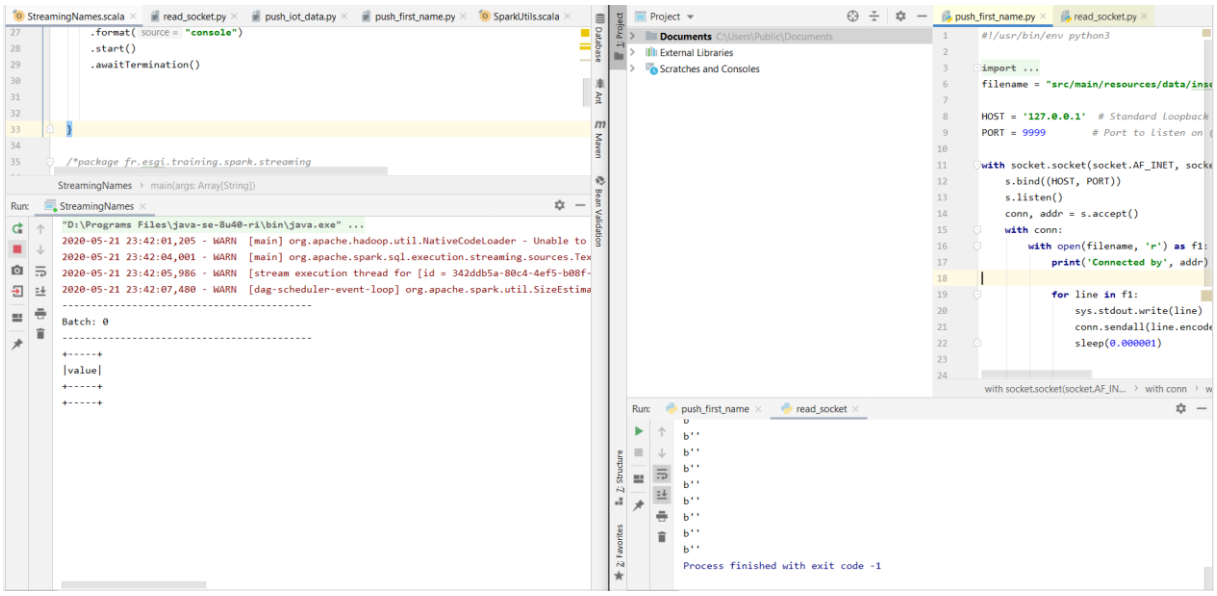
```
val aggregateDF = df

aggregateDF.writeStream.outputMode("append")
  .format("console")
  .start()
  .awaitTermination()
```

```
|  1;ADIB;2018;23|
|  1;ADIEL;2015;3|
+----------------+
only showing top 20 rows

-------------------------------------------
Batch: 31
-------------------------------------------
+----------------+
|           value|
+----------------+
|  1;ADLAN;2017;7|
| 1;ADLANE;1980;5|
```

4. Lancez le script socket, puis le job Spark.



5. Attendez la fin du job, avec un timeout de 3 minutes. Décrivez le résultat.

Le temps que le job Spark se lance, l'écriture dans la socket est interrompu car le script push_name.py échoue. Il y a en effet une erreur dans ce fichier qui interrompt le script prématurément.

Cependant pendant le peu de temps d'écriture du script, les données sont en sortie sur le output de read_socket.py

II.

1. Créez une nouvelle colonne où les données sont splittées (Vous devriez avoir un Array)

```
val aggregateDF = df
val aggregateDF1 = aggregateDF.withColumn("value_splited",  split(col("value"),
";"))
```

2. Transformez votre DataStream pour obtenir une structure utilisable (Colonnes pour chaque donnée)

```
val aggregateDF2 = aggregateDF1.withColumn("tmp", col("value_splited")).select(
  col("tmp").getItem(0).as("sexe"),
  col("tmp").getItem(1).as("preusuel"),
  col("tmp").getItem(2).as("annais"),
  col("tmp").getItem(3).as("nombre").cast(DataTypes.IntegerType))


aggregateDF2.writeStream
  .outputMode("append")
  .format("console")
  .start()
  .awaitTermination(3*60*100)
```

III.

1. Comptez le nombre de naissances par Sexe

```
val DFsexe =
aggregateDF2.select(col("sexe"),col("nombre")).groupBy("sexe").sum("nombre")
```

2. Quel sont les prénoms les plus donnés ?

```
val DFmaxPrenom=
aggregateDF2.select(col("preusuel"),col("nombre")).groupBy("preusuel").sum("nombre
").orderBy(sum("nombre").desc)
```

3. Quelles sont les années ayant le plus de naissance ?

```
val DFannee =
aggregateDF2.select(col("annais"),col("nombre")).groupBy("annais").sum("nombre").o
rderBy(sum("nombre").desc)
```

4. Quel autre donnée pourrait-on sortir de ce stream ?

➔ Le sexe par prénom

```
val DFsexePrenom =
aggregateDF2.select(col("prenom"),col("sexe")).groupBy("prenom","sexe").sum("nombr
e")
```

PART 2

1. Donnez la température moyenne de la pièce toutes les minutes

```
var aggregateDFiot1 = aggregateDF.withWatermark("timestamp", "10 minutes")
  .groupBy(col("id_iot"), window(col("timestamp"),"1 minute"))
  .mean("temp")

aggregateDFiot1.writeStream
  .outputMode("complete")
  .format("console")
  .start()
  .awaitTermination()
```

2. Donnez la température moyenne sur 1 minute de la pièce toutes les 30 secondes

```
var aggregateDFiot2 = aggregateDF.withWatermark("timestamp", "10 minutes")
  .groupBy(col("id_iot"), window(col("timestamp"), "1 minute", "30 seconds"))
  .mean("temp")

aggregateDFiot2.writeStream
  .outputMode("complete")
  .format("console")
  .start()
  .awaitTermination()
```

3. Quel temps avez-vous utilisé pour créer vos fenêtres ?

Question assez flou :

```
    window(col("timestamp"), "1 minute", "30 seconds")
```

On choisie en colonne la colonne qui contient l'heure, on crée ensuite des fenêtres de 1 min que l'on découpe en slides toute les 30 secondes.

4. Proposez d'autres calculs sur d'autres types de fenêtres.

➔ La température minimum, moyenne et maximum toute les heures d'une piéce

```
var aggregateDFiot4 = aggregateDF.withWatermark("timestamp", "10 minutes")
  .groupBy(col("id_iot"), window(col("time"), "1 hour")).agg(min(col("temp")),
avg(col("temp")), max(col("temp")))

aggregateDFiot4.writeStream
  .outputMode("complete")
  .format("console")
  .start()
  .awaitTermination()
```