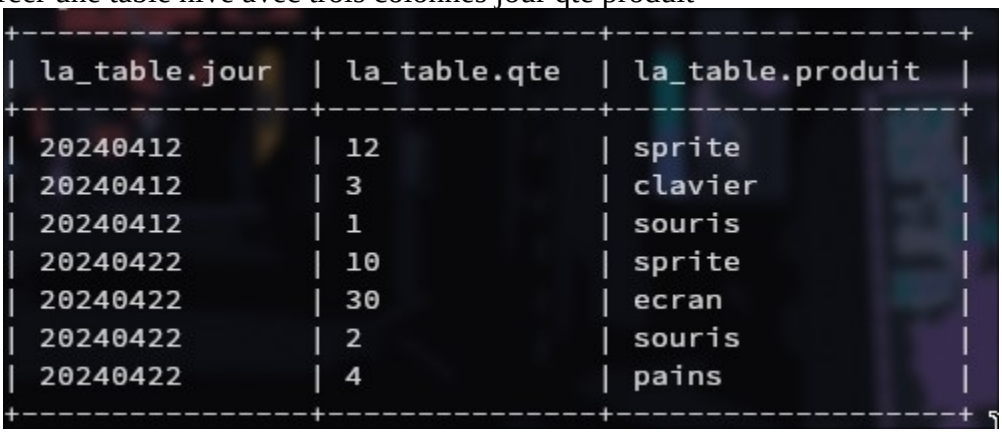
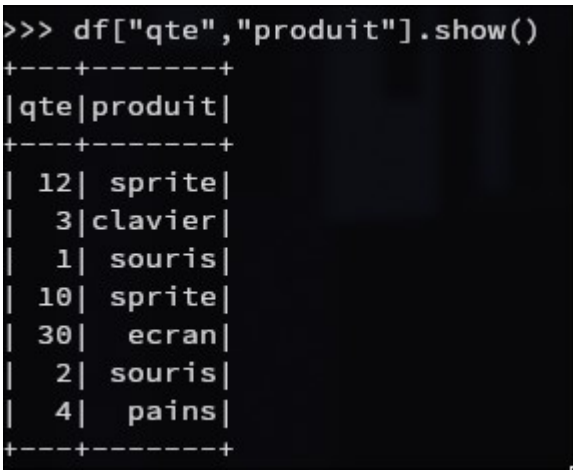
Exercice

- créer une table hive avec trois colonnes jour qte produit



la_table.jour	la_table.qte	la_table.produit
20240412	12	sprite
20240412	3	clavier
20240412	1	souris
20240422	10	sprite
20240422	30	ecran
20240422	2	souris
20240422	4	pains

- se connecter à la table hive depuis spark et charger le contenu sous forme de dataframe
df=spark.sql("select * from test.la_table")
1 - afficher deux colonnes de votre choix
(un select aussi aurait fait l'affaire)



```
>>> df["qte","produit"].show()
+---+-------+
|qte|produit|
+---+-------+
| 12| sprite|
|  3|clavier|
|  1| souris|
| 10| sprite|
| 30|  ecran|
|  2| souris|
|  4|  pains|
+---+-------+
```

2 - créer une nouvelle colonne qui contiendra la liste des produits mais en majuscules
On aura besoin d'importer deux fonction de la librairies pyspark (upper et col)

from pyspark.sql.functions import upper, col

```
>>> df.withColumn("maj_produit", upper(col("produit"))).show()
+--------+---+-------+-----------+
|    jour|qte|produit|maj_produit|
+--------+---+-------+-----------+
|20240412| 12| sprite|     SPRITE|
|20240412|  3|clavier|    CLAVIER|
|20240412|  1| souris|     SOURIS|
|20240422| 10| sprite|     SPRITE|
|20240422| 30|  ecran|      ECRAN|
|20240422|  2| souris|     SOURIS|
|20240422|  4|  pains|      PAINS|
+--------+---+-------+-----------+
```

3 - créer une colonne telle que si la quantité est inférieure à 20, la valeur sera A, sinon la valeur sera B
On aura besoin d'une autre fonction
from pyspark.sql.functions import when

```
>>> df.withColumn("large_qte", when(df["qte"]<20, "A").otherwise("B")).show()
+--------+---+-------+-----------+---------+
|    jour|qte|produit|maj_produit|large_qte|
+--------+---+-------+-----------+---------+
|20240412| 12| sprite|     SPRITE|        A|
|20240412|  3|clavier|    CLAVIER|        A|
|20240412|  1| souris|     SOURIS|        A|
|20240422| 10| sprite|     SPRITE|        A|
|20240422| 30|  ecran|      ECRAN|        B|
|20240422|  2| souris|     SOURIS|        A|
|20240422|  4|  pains|      PAINS|        A|
+--------+---+-------+-----------+---------+
```

5 - créer une colonne qui va transformer la colonne jour au format de date de votre choix
from pyspark.sql.functions import to_date, date_format
On doit d'abord convertir en format date avec cette commande

```
>>> df.select(col("jour"),to_date(col("jour"), "yyyyMMdd").alias("date")).show()
+--------+----------+
|    jour|      date|
+--------+----------+
|20240412|2024-04-12|
|20240412|2024-04-12|
|20240412|2024-04-12|
|20240422|2024-04-22|
|20240422|2024-04-22|
|20240422|2024-04-22|
|20240422|2024-04-22|
+--------+----------+
```

Ensuite on peut modifier ce format en utilisant *date_format*

```
>>> df.select( date_format(col("date"),"ddMMyyyy")).show()
+--------------------------+
|date_format(date, ddMMyyyy)|
+--------------------------+
|                  12042024|
|                  12042024|
|                  12042024|
|                  22042024|
|                  22042024|
|                  22042024|
|                  22042024|
+--------------------------+
```

6 - créer un dataframe et appliquer un groupe by par produit
On crée un nouveau dataframe et on applique le groupBy. Pour afficher le résultat il faut effectuer une opération (ici somme par produit)

```
>>> df=spark.sql("select * from test.la_table").withColumn("maj_produit", upper(col("produit")))
>>> df.groupBy("produit").sum("qte").show()
+-------+--------+
|produit|sum(qte)|
+-------+--------+
|clavier|       3|
| souris|       3|
| sprite|      22|
|  pains|       4|
|  ecran|      30|
+-------+--------+
```

7 -sauvegarder ce dernier dataframe dans une nouvelle table hive

```
>>> df.write.saveAsTable('test.new_table')
>>>
```