REALISATIONS DES DIFFERENS DE L'EXEMPLE PRESENTE DANS LE CHAPITRE 2

PRATIQUE 1 :

- Création d'une SparkSession Object

Nous importons les différentes fonctions et types de données requis à partir despark.sql

```
brice@wambdevps:/opt/spark$ pyspark
Python 3.8.10 (default, Jun 22 2022, 20:18:18)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
22/11/02 05:41:12 WARN Utils: Your hostname, wambdevps resolves to a loopback address: 127.0.0.2, but we couldn't find any external IP address!
22/11/02 05:41:12 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/11/02 05:41:13 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
22/11/02 05:41:14 WARN MacAddressUtil: Failed to find a usable hardware address from the network interfaces; using random bytes: d7:61:6e:16:fb:0c:2b:d
b
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.3.1
      /_/

Using Python version 3.8.10 (default, Jun 22 2022 20:18:18)
Spark context Web UI available at http://wambdevps:4040
Spark context available as 'sc' (master = local[*], app id = local-1667364075009).
SparkSession available as 'spark'.
>>> from pyspark.sql import SparkSession
>>> spark=SparkSession.builder.appName('data_processing').getOrCreate()
22/11/02 05:42:17 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>> import pyspark.sql.functions as F
>>> from pyspark.sql.types import *
```

- Création d'une Dataframes

Nous créons une nouvelle trame de données avec cinq colonnes de certains types de données (chaîne et entier).

```
>>> schema=StructType().add("user_id","string").add("country","string").add("browser", "string").add("OS",'string').add("age", "integer")
>>> df=spark.createDataFrame([("A203",'India',"Chrome","WIN",33),("A201",'China',"Safari","MacOS",35),("A205",'UK',"Mozilla","Linux",25)],schema=schema
)
>>> df.printSchema()
root
 |-- user_id: string (nullable = true)
 |-- country: string (nullable = true)
 |-- browser: string (nullable = true)
 |-- OS: string (nullable = true)
 |-- age: integer (nullable = true)

>>> df.show()
+-------+-------+-------+-----+---+
|user_id|country|browser|   OS|age|
+-------+-------+-------+-----+---+
|   A203|  India| Chrome|  WIN| 33|
|   A201|  China| Safari|MacOS| 35|
|   A205|     UK|Mozilla|Linux| 25|
+-------+-------+-------+-----+---+

>>>
```

- traitement des valeurs nulles

Tout d'abord, nous créons un nouveau dataframe (df_na) qui contient des valeurs nulles dans deux de ses colonnes (le schéma est le même que dans la trame de données précédente). Par la première approche pour traiter les valeurs nulles, nous remplissons toutes les valeurs nulles dans la trame de données actuelle avec une valeur de 0, ce qui offre une solution rapide. Par la deuxième approche, nous remplaçons les valeurs nulles dans des colonnes spécifiques (pays, navigateur) par 'USA'et 'Safari',respectivement.

1) Première approche

```
>>> df_na=spark.createDataFrame([("A203",None,"Chrome","WIN",
... 33),("A201",'China',None,"MacOS",35),("A205",'UK',"Mozilla","Linux",25)],schema=schema)
>>> df_na.show()
+-------+-------+-------+-----+---+
|user_id|country|browser|   OS|age|
+-------+-------+-------+-----+---+
|   A203|   null| Chrome|  WIN| 33|
|   A201|  China|   null|MacOS| 35|
|   A205|     UK|Mozilla|Linux| 25|
+-------+-------+-------+-----+---+
```

```
>>> df_na.fillna('0').show()
+-------+-------+-------+-----+---+
|user_id|country|browser|   OS|age|
+-------+-------+-------+-----+---+
|   A203|      0| Chrome|  WIN| 33|
|   A201|  China|      0|MacOS| 35|
|   A205|     UK|Mozilla|Linux| 25|
+-------+-------+-------+-----+---+
```

```
>>> df_na.fillna( { 'country':'USA', 'browser':'Safari' } ).show()
+-------+-------+-------+-----+---+
|user_id|country|browser|   OS|age|
+-------+-------+-------+-----+---+
|   A203|    USA| Chrome|  WIN| 33|
|   A201|  China| Safari|MacOS| 35|
|   A205|     UK|Mozilla|Linux| 25|
+-------+-------+-------+-----+---+
```

- Suppression des lignes avec des valeurs nulles

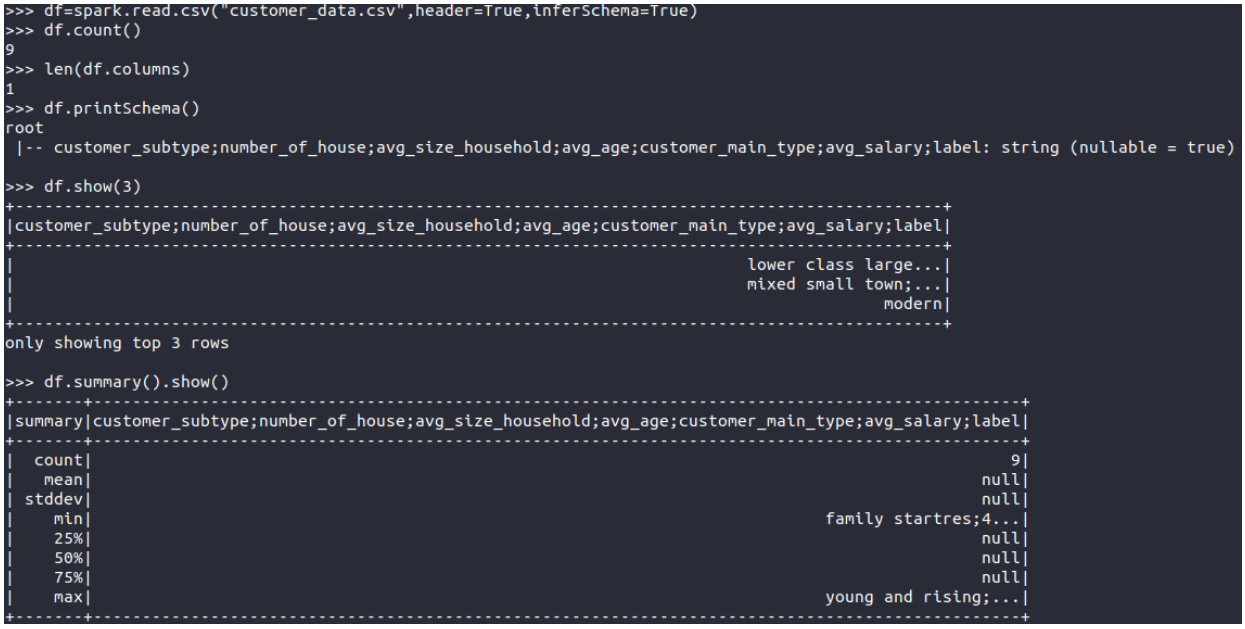
```
>>> df_na.na.drop().show()
+-------+-------+-------+-----+---+
|user_id|country|browser|   OS|age|
+-------+-------+-------+-----+---+
|   A205|     UK|Mozilla|Linux| 25|
+-------+-------+-------+-----+---+

>>> df_na.na.drop(subset='country').show()
+-------+-------+-------+-----+---+
|user_id|country|browser|   OS|age|
+-------+-------+-------+-----+---+
|   A201|  China|   null|MacOS| 35|
|   A205|     UK|Mozilla|Linux| 25|
+-------+-------+-------+-----+---+
```

2) deuxieme approche avec la fonction replace()

PRATIQUE 2 :

- Création d'une dataframe Spark EN lisant un fichier **.csv**

  Créons une dataframe Spark, en lisant un fichier (.csv, parquet,etc.). Le jeu de données contient un total de 1 colonne et 9 lignes. La fonction de résumé nous permet de voir les mesures statistiques de l'ensemble de données, telles que le min, le max et la moyenne des données numériques présentes dans la trame de données.



```
>>> df=spark.read.csv("customer_data.csv",header=True,inferSchema=True)
>>> df.count()
9
>>> len(df.columns)
1
>>> df.printSchema()
root
 |-- customer_subtype;number_of_house;avg_size_household;avg_age;customer_main_type;avg_salary;label: string (nullable = true)

>>> df.show(3)
+-----------------------------------------------------------------------------------------+
|customer_subtype;number_of_house;avg_size_household;avg_age;customer_main_type;avg_salary;label|
+-----------------------------------------------------------------------------------------+
|                                                                    lower class large...|
|                                                                    mixed small town;...|
|                                                                                modern|
+-----------------------------------------------------------------------------------------+
only showing top 3 rows

>>> df.summary().show()
+-------+-----------------------------------------------------------------------------------------------+
|summary|customer_subtype;number_of_house;avg_size_household;avg_age;customer_main_type;avg_salary;label|
+-------+-----------------------------------------------------------------------------------------------+
|  count|                                                                                              9|
|   mean|                                                                                           null|
| stddev|                                                                                           null|
|    min|                                                                        family startres;4...|
|    25%|                                                                                           null|
|    50%|                                                                                           null|
|    75%|                                                                                           null|
|    max|                                                                        young and rising;...|
+-------+-----------------------------------------------------------------------------------------------+
```