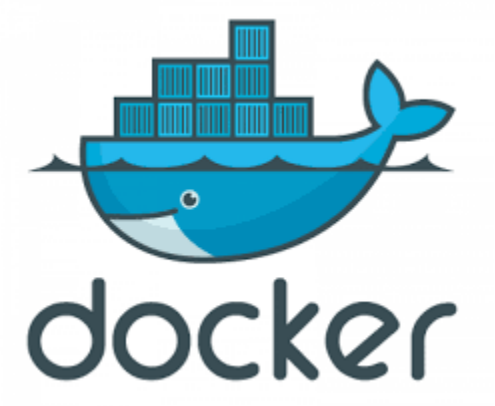
Rapport de projet Big Data



RÉALISÉE PAR

Youssef Ben Romdhane

2BD1

Introduction :

Ce rapport présente le processus d'installation Docker Desktop et d'une image Hadoop, ainsi que la gestion des conteneurs Docker et la configuration des services Hadoop. L'objectif est de mettre en place un environnement de travail avec Hadoop et Docker Desktop pour le traitement de données volumineuses à l'aide du fichier purchases.txt, tout en réalisant une Analyse des Données avec Spark en Scala.

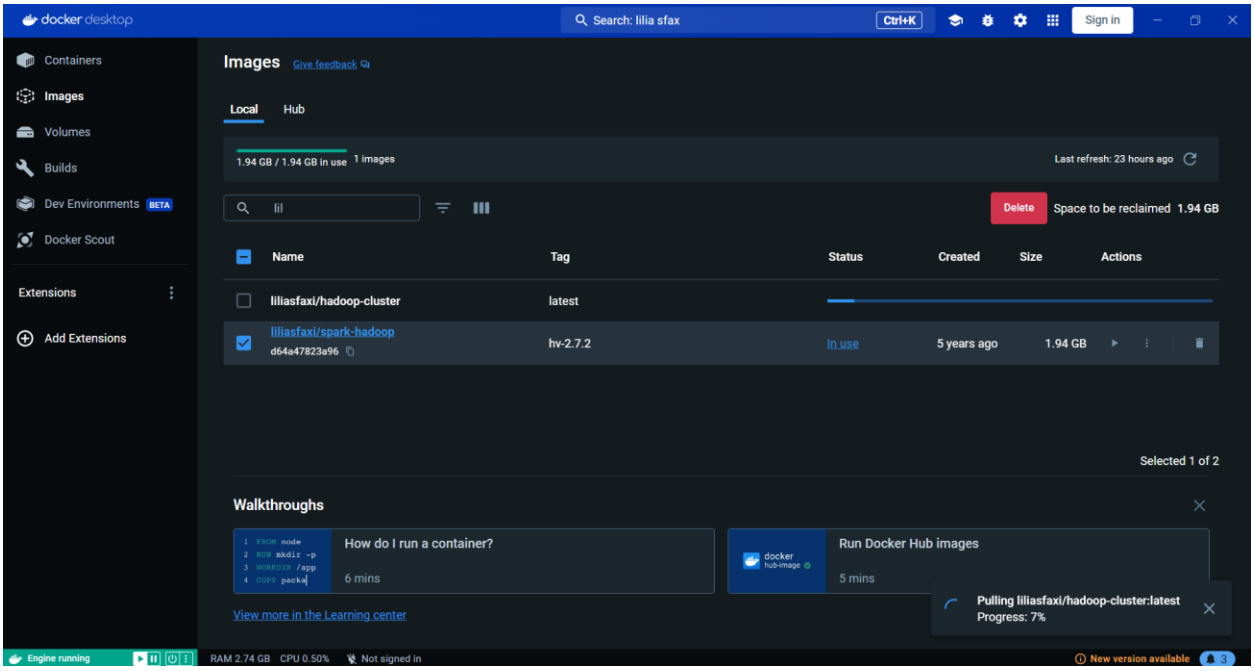
BILAN :

1/ Installation Docker
2/ Manipulation de Cluster Hadoop:
    A/ Configuration mapred-site.xml
    B/ Job MapReduce
3/ Analyse de données avec spark scala
4/Conclusion

Installation Docker

J'ai installé Docker sur Windows en utilisant Docker Desktop. J'ai suivi les instructions fournies dans ce lien pour l'installer:

https://www.docker.com/products/docker-desktop/



Ensuite, j'ai trouvé une image Docker qui contenait déjà Hadoop et Spark. Je l'ai téléchargée sur mon système en utilisant la commande appropriée.

docker pull liliasfaxi/hadoop-cluster:latest

Une fois l'image téléchargée, nous allons créer un réseau pour relier nos trois conteneurs et former notre cluster. Cela se fait avec la commande suivante :

```
docker network create --driver=bridge hadoop
```

Ensuite, nous allons créer nos conteneurs, chacun avec son propre port d'entrée, mais tous sur le même réseau pour qu'ils puissent communiquer en tant que cluster.

Master :

```
docker run -itd --net=hadoop -p 9870:9870 -p 8088:8088 -p 7077:7077 -p 16010:16010 --name hadoop-master --hostname hadoop-master liliasfaxi/hadoop-cluster:latest
```

Slaves:

```
docker run -itd -p 8040:8042 --net=hadoop --name hadoop-worker1
--hostname hadoop-worker1 liliasfaxi/hadoop-cluster:latest
```

```
docker run -itd -p 8041:8042 --net=hadoop --name hadoop-worker2
--hostname hadoop-worker2 liliasfaxi/hadoop-cluster:latest
```

Pour voir si ceci marche on va utiliser la requête docker ps:

```
C:\Users\PC>docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED         STATUS
 PORTS                                                                                                 NAMES
e3e6e68a65c3   liliasfaxi/hadoop-cluster:latest   "sh -c 'service ssh …"   8 seconds ago   Exited (0) 6 seconds ago
                                                                                                       goofy_solomon
ce874dbab49c   liliasfaxi/hadoop-cluster:latest   "sh -c 'service ssh …"   13 hours ago    Exited (255) 53 seconds ago
 7077/tcp, 8088/tcp, 9870/tcp, 16010/tcp, 0.0.0.0:8041->8042/tcp                                       hadoop-worker2
80919cab61c3   liliasfaxi/hadoop-cluster:latest   "sh -c 'service ssh …"   13 hours ago    Exited (255) 53 seconds ago
 7077/tcp, 8088/tcp, 9870/tcp, 16010/tcp, 0.0.0.0:8040->8042/tcp                                       hadoop-worker1
e9b8b03dcf06   liliasfaxi/hadoop-cluster:latest   "sh -c 'service ssh …"   13 hours ago    Exited (255) 53 seconds ago
 0.0.0.0:7077->7077/tcp, 0.0.0.0:8088->8088/tcp, 0.0.0.0:9870->9870/tcp, 0.0.0.0:16010->16010/tcp   hadoop-master
0238392d24d9   liliasfaxi/hadoop-cluster:latest   "sh -c 'service ssh …"   13 hours ago    Exited (0) 13 hours ago
                                                                                                       hungry_lalande
```

Manipulation de Cluster Hadoop

Nous déplaçons maintenant les fichiers requis dans le conteneur principal.

docker cp "C:\Users\PC\Downloads\purchases.txt~\purchases.txt" hadoop-master:/root/purchases.txt

```
C:\Users\PC>docker cp C:\Users\PC\Downloads\purchases.txt hadoop-master:/root/purchases.txt
Successfully copied 211MB to hadoop-master:/root/purchases.txt

C:\Users\PC>docker cp C:\Users\PC\Desktop\mapper.py hadoop-master:/root/mapper.py
Successfully copied 2.05kB to hadoop-master:/root/mapper.py

C:\Users\PC>docker cp C:\Users\PC\Desktop\reducer.py hadoop-master:/root/reducer.py
Successfully copied 2.56kB to hadoop-master:/root/reducer.py
```

Puis entrer dans le bash du master avec la requête :

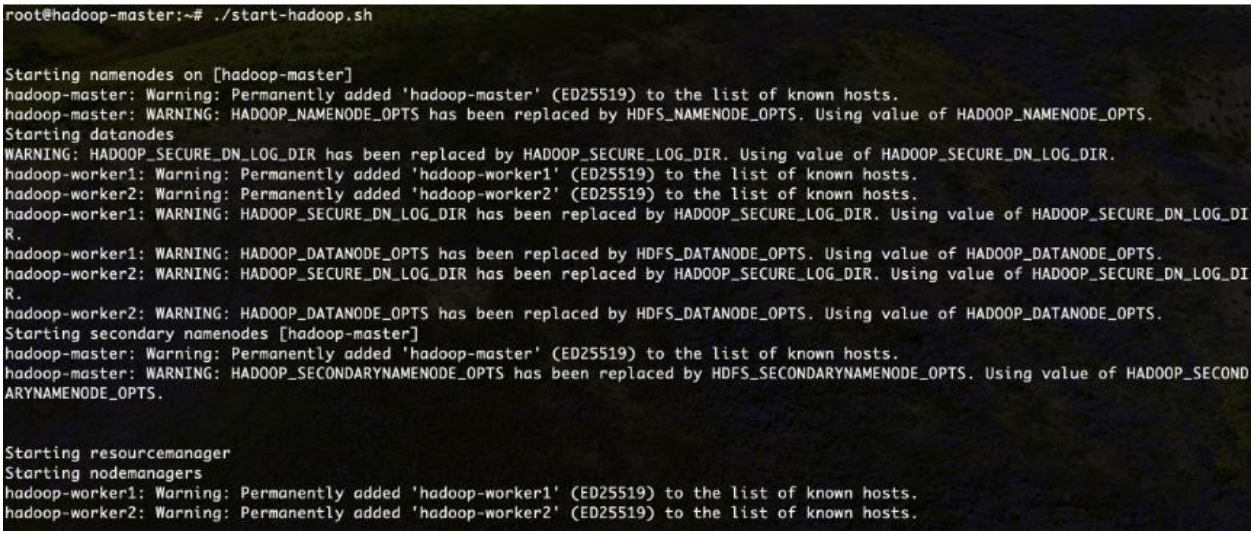
docker exec -it hadoop-master bash

```
C:\Users\PC>docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED         STATUS                     PO
RTS                                                                                                 NAMES
ce874dbab49c   liliasfaxi/hadoop-cluster:latest   "sh -c 'service ssh …"   8 seconds ago   Up 6 seconds               70
77/tcp, 8088/tcp, 9870/tcp, 16010/tcp, 0.0.0.0:8041->8042/tcp                                       hadoop-worker2
80919cab61c3   liliasfaxi/hadoop-cluster:latest   "sh -c 'service ssh …"   8 seconds ago   Up 7 seconds               70
77/tcp, 8088/tcp, 9870/tcp, 16010/tcp, 0.0.0.0:8040->8042/tcp                                       hadoop-worker1
e9b8b03dcf06   liliasfaxi/hadoop-cluster:latest   "sh -c 'service ssh …"   9 seconds ago   Up 7 seconds               0.
0.0.0:7077->7077/tcp, 0.0.0.0:8088->8088/tcp, 0.0.0.0:9870->9870/tcp, 0.0.0.0:16010->16010/tcp   hadoop-master
0238392d24d9   liliasfaxi/hadoop-cluster:latest   "sh -c 'service ssh …"   2 minutes ago   Exited (0) 2 minutes ago
                                                                                                    hungry_lalande


C:\Users\PC>docker exec -it hadoop-master bash
root@hadoop-master:~#
```

démarre le yarn et le hadoop:

./start-hadoop.sh



Configuration de mapred-site.xml :

Configuration ：

```
<configuration>
<!-- Configurations for MapReduce Applications: -->
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>yarn.app.mapreduce.am.env</name>
<value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
<property>
<name>mapreduce.map.env</name>
<value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
<property>
<name>mapreduce.reduce.env</name>
<value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
```

</configuration>

```
C:\Users\PC>docker exec -it hadoop-master bash
root@hadoop-master:~# ls
ds.csv  mapper.py      purchases2.txt  run-wordcount.sh  start-hadoop.sh
hdfs    purchases.txt  reducer.py      spark-warehouse   start-kafka-zookeeper.sh
root@hadoop-master:~# cd $HADOOP_CONF_DIR/
root@hadoop-master:/usr/local/hadoop/etc/hadoop# ls
capacity-scheduler.xml      hadoop-policy.xml        kms-log4j.properties        slaves
configuration.xsl           hdfs-site.xml            kms-site.xml                ssl-client.xml.example
container-executor.cfg      httpfs-env.sh            log4j.properties            ssl-server.xml.example
core-site.xml               httpfs-log4j.properties  mapred-env.cmd              yarn-env.cmd
hadoop-env.cmd              httpfs-signature.secret  mapred-env.sh               yarn-env.sh
hadoop-env.sh               httpfs-site.xml          mapred-queues.xml.template  yarn-site.xml
hadoop-metrics.properties   kms-acls.xml             mapred-site.xml
hadoop-metrics2.properties  kms-env.sh               mapred-site.xml.template
root@hadoop-master:/usr/local/hadoop/etc/hadoop# vi mapred-site.xml
root@hadoop-master:/usr/local/hadoop/etc/hadoop#
```

- Créer un répertoire dans HDFS, appelé *input* :
  `hdfs dfs –mkdir -p input`

- À partir du contenaire master, charger le fichier purchases dans le répertoire input (de HDFS) que vous avez créé:

  `hdfs dfs –put purchases.txt input`

- Pour afficher le contenu du répertoire *input*, la commande est:

  `hdfs dfs –ls input`

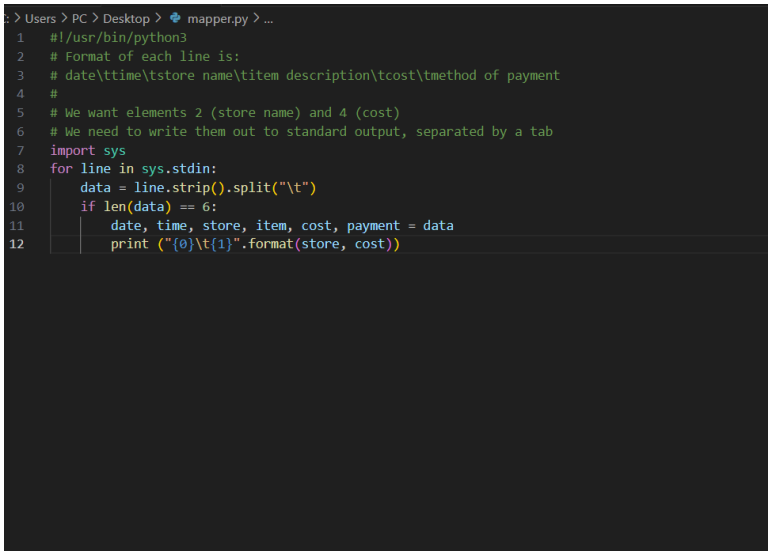
- Pour afficher les dernières lignes du fichier purchases:
-

  `hdfs dfs -tail input/purchases.txt`

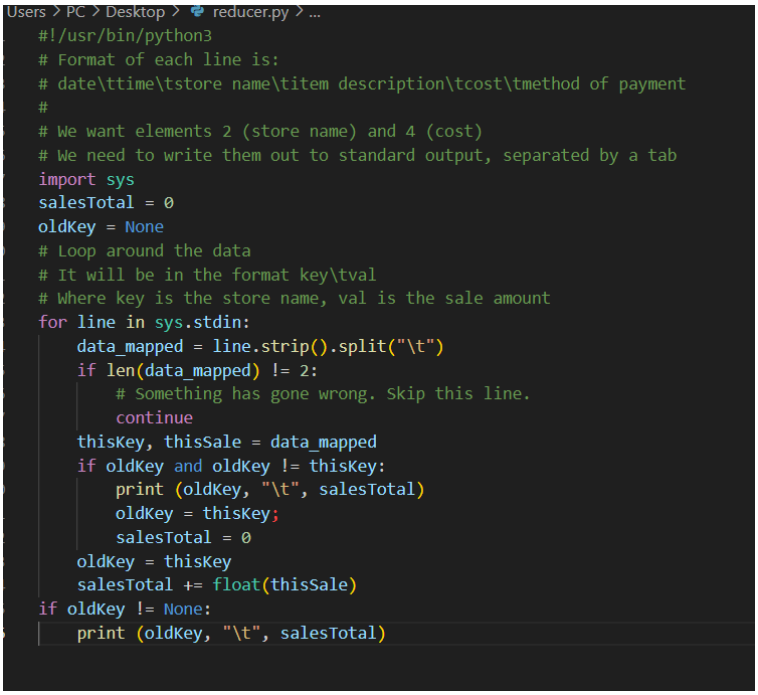
```
root@hadoop-master:~# hdfs dfs -tail input/purchases.txt
31      17:59   Norfolk Toys    164.34  MasterCard
2012-12-31      17:59   Chula Vista     Music   380.67  Visa
2012-12-31      17:59   Hialeah Toys    115.21  MasterCard
2012-12-31      17:59   Indianapolis    Men's Clothing  158.28  MasterCard
2012-12-31      17:59   Norfolk Garden  414.09  MasterCard
2012-12-31      17:59   Baltimore       DVDs    467.3   Visa
2012-12-31      17:59   Santa Ana       Video Games     144.73  Visa
2012-12-31      17:59   Gilbert Consumer Electronics    354.66  Discover
2012-12-31      17:59   Memphis Sporting Goods  124.79  Amex
2012-12-31      17:59   Chicago Men's Clothing  386.54  MasterCard
2012-12-31      17:59   Birmingham      CDs     118.04  Cash
2012-12-31      17:59   Las Vegas       Health and Beauty       420.46  Amex
2012-12-31      17:59   Wichita Toys    383.9   Cash
2012-12-31      17:59   Tucson  Pet Supplies    268.39  MasterCard
2012-12-31      17:59   Glendale        Women's Clothing        68.05   Amex
2012-12-31      17:59   Albuquerque     Toys    345.7   MasterCard
2012-12-31      17:59   Rochester       DVDs    399.57  Amex
2012-12-31      17:59   Greensboro      Baby    277.27  Discover
2012-12-31      17:59   Arlington       Women's Clothing        134.95  MasterCard
2012-12-31      17:59   Corpus Christi  DVDs    441.61  Discover
```

Job MapReduce:

Les fichiers mapper.py + reducer.py :



```
: > Users > PC > Desktop > mapper.py > ...
#!/usr/bin/python3
# Format of each line is:
# date\ttime\tstore name\titem description\tcost\tmethod of payment
#
# We want elements 2 (store name) and 4 (cost)
# We need to write them out to standard output, separated by a tab
import sys
for line in sys.stdin:
    data = line.strip().split("\t")
    if len(data) == 6:
        date, time, store, item, cost, payment = data
        print ("{0}\t{1}".format(store, cost))
```



```
Users > PC > Desktop > reducer.py > ...
#!/usr/bin/python3
# Format of each line is:
# date\ttime\tstore name\titem description\tcost\tmethod of payment
#
# We want elements 2 (store name) and 4 (cost)
# We need to write them out to standard output, separated by a tab
import sys
salesTotal = 0
oldKey = None
# Loop around the data
# It will be in the format key\tval
# Where key is the store name, val is the sale amount
for line in sys.stdin:
    data_mapped = line.strip().split("\t")
    if len(data_mapped) != 2:
        # Something has gone wrong. Skip this line.
        continue
    thisKey, thisSale = data_mapped
    if oldKey and oldKey != thisKey:
        print (oldKey, "\t", salesTotal)
        oldKey = thisKey;
        salesTotal = 0
    oldKey = thisKey
    salesTotal += float(thisSale)
if oldKey != None:
    print (oldKey, "\t", salesTotal)
```

Commande pour le job :

hadoop jar/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -input input/purchases.txt -output /new_output -mapper /root/Mapper.py -reducer/root/Reducer.py -file Mapper.py -file Reducer.py

Résultat avec la commande : hadoop fs -cat /new_output/part-00000

```
root@hadoop-master: ~
666 ResourceManager
990 Jps
175 NameNode
root@hadoop-master:~# hadoop fs -cat /root/new_output/part-00000
Albuquerque      10052311.419999966
Anaheim          10076416.359999955
Anchorage        9933500.400000004
Arlington        10072207.97000006
Atlanta          9997146.700000083
Aurora   9992970.91999996
Austin   10057158.900000017
Bakersfield      10031208.920000002
Baltimore        10096521.450000018
Baton Rouge      10131273.229999963
Birmingham       10076606.520000014
Boise    10039166.74
Boston   10039473.280000059
Buffalo          10001941.190000026
Chandler         9919559.860000018
Charlotte        10112531.339999998
Chesapeake       10038504.919999966
Chicago          10062522.070000188
Chula Vista      9974951.340000078
Cincinnati       10139505.740000034
Cleveland        10067835.840000032
Colorado Springs         10061105.870000029
Columbus         10035241.030000037
Corpus Christi   9976522.76999999
Dallas   10066548.449999982
Denver   10031534.86999997
```

Analyse de données avec spark scala

Aprés le démarrage de Hadoop dans le master container accéder à scala par la commande : spark-shell

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_191)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Quelques importations essentielles :

```
import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.{DataFrame, SparkSession}

import org.apache.spark.sql.functions._
```

```
import scala.sys.process._
```

Création de session :

```
val spark = SparkSession.builder().appName("CSV Reader").master("local").getOrCreate()
```

Lire le fichier CSV:

```
val df = spark.read.option("header", "true").option("delimiter", ";").csv("file:///root/ds.csv")
```

NB : if faut changer les types des colonnes car scala les prend comme le type String :

```
val dfTyped = df.selectExpr(
 "Sport",
 "Pays",
 "Tourisme",
 "Cinema",
 "Sexe",
 "Musique",
 "ChassePeche",
 "Fumer",
 "ReseauSocial",
 "Activite",
 "Satisfaction",
 "Cuisine",
 "CAST(Age AS INT) AS Age",
 "CAST(Poids AS DOUBLE) AS Poids",
 "CAST(NbFreresSoeurs AS INT) AS NbFreresSoeurs",
 "CAST(NbAnimaux AS INT) AS NbAnimaux",
 "CAST(NbEnfants AS INT) AS NbEnfants",
 "CAST(Taille AS DOUBLE) AS Taille",
```

"CAST(HeuresTV AS INT) AS HeuresTV",

"CAST(Nblivres AS INT) AS Nblivres")

Utiliser la commande dfTyped.printSchema() pour vérifier :

```
scala> dfTyped.printSchema()
root
 |-- Sport: string (nullable = true)
 |-- Pays: string (nullable = true)
 |-- Tourisme: string (nullable = true)
 |-- Cinema: string (nullable = true)
 |-- Sexe: string (nullable = true)
 |-- Musique: string (nullable = true)
 |-- ChassePeche: string (nullable = true)
 |-- Fumer: string (nullable = true)
 |-- ReseauSocial: string (nullable = true)
 |-- Activite: string (nullable = true)
 |-- Satisfaction: string (nullable = true)
 |-- Cuisine: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Poids: double (nullable = true)
 |-- NbFreresSoeurs: integer (nullable = true)
 |-- NbAnimaux: integer (nullable = true)
 |-- NbEnfants: integer (nullable = true)
 |-- Taille: double (nullable = true)
 |-- HeuresTV: integer (nullable = true)
 |-- Nblivres: integer (nullable = true)
```

Exploration des données (optionnel):

```
scala> dfTyped.show(5)
+----------+---------+---------+------------------+-----+---------+-----------+-----+------------+--------------+------
--------+-------+---+-----+--------------+---------+---------+------+--------+--------+
|     Sport|     Pays|Tourisme|           Cinema| Sexe|  Musique|ChassePeche|Fumer|ReseauSocial|      Activite|  Sati
sfaction|Cuisine|Age|Poids|NbFreresSoeurs|NbAnimaux|NbEnfants|Taille|HeuresTV|Nblivres|
+----------+---------+---------+------------------+-----+---------+-----------+-----+------------+--------------+------
--------+-------+---+-----+--------------+---------+---------+------+--------+--------+
|  Handball| Belgique|Maldives|Occasionnellement|Femme|   Hiphop|        Non|  Oui|   Instagram|Administration|     S
atisfait|    Oui| 31|103.0|             2|        3|        7| 170.0|       8|       4|
|  Handball|  Irlande|  Bresil|              Non|Femme|     Rock|        Non|  Oui|   Pinterest|       Support|     S
atisfait|    Non| 48| 63.0|             2|        3|        1| 181.0|       8|       4|
|      Polo|    Suede| Londres|              Oui|Homme|     Rock|        Non|  Non|    Facebook|    Production|Pas de
 r?ponse|    Non| 38|110.0|             9|        0|        1| 157.0|       5|      12|
|    Hockey|  Irlande|   Maroc|Occasionnellement|Homme|    House|        Non|  Oui|   Instagram|  Informatique|     S
atisfait|    Oui| 38| 81.0|            10|        3|        4| 199.0|       5|      10|
|Volleyball|Allemagne|  Egypte|              Non|Femme|Classique|        Non|  Oui|       Aucun|     Transport| Pas s
atisfait|    Non| 45| 76.0|             0|        6|        4| 183.0|      12|      15|
+----------+---------+---------+------------------+-----+---------+-----------+-----+------------+--------------+------
--------+-------+---+-----+--------------+---------+---------+------+--------+--------+
only showing top 5 rows
```

```
scala> // Find the most common countries reported by the respondents

scala> val mostCommonCountries = dfTyped.groupBy("Pays").count().orderBy(desc("count")).limit(5)
mostCommonCountries: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Pays: string, count: bigint]

scala> println("Most Common Countries:")
Most Common Countries:

scala> mostCommonCountries.show()
+---------+-----+
|     Pays|count|
+---------+-----+
|Allemagne|  120|
|   Serbie|  117|
| Autriche|  116|
|   Italie|  116|
|    Grece|  114|
+---------+-----+
```

```
scala> // Determine the average number of siblings and children reported by the respondents

scala> val avgSiblings = dfTyped.select(avg("NbFreresSoeurs")).first().getDouble(0)
avgSiblings: Double = 5.04

scala> val avgChildren = dfTyped.select(avg("NbEnfants")).first().getDouble(0)
avgChildren: Double = 3.9295

scala> println(s"Average Number of Siblings: $avgSiblings")
Average Number of Siblings: 5.04

scala> println(s"Average Number of Children: $avgChildren")
Average Number of Children: 3.9295
```

```
scala> val avgAge = dfTyped.select(avg("Age")).first().getDouble(0)
avgAge: Double = 40.979

scala> println(s"Average Age: $avgAge")
Average Age: 40.979
```

Analyse de données (à partir des KPIs):

Nombre moyen d'enfants par pays :

```
scala> val avgChildrenByCountry = dfTyped.groupBy("Pays").agg(avg("NbEnfants").alias("AvgChildrenPerCountry"))
avgChildrenByCountry: org.apache.spark.sql.DataFrame = [Pays: string, AvgChildrenPerCountry: double]

scala> avgChildrenByCountry.show(10)
+---------+---------------------+
|     Pays|AvgChildrenPerCountry|
+---------+---------------------+
| Autriche|   3.8879310344827585|
|  Hongrie|   3.4545454545454546|
| Finlande|                 3.76|
| Bulgarie|   3.8878504672897196|
|   France|   3.5934065934065935|
|  Espagne|   3.9894736842105263|
|  Croatie|    4.247422680412371|
|   Suisse|   3.7731958762886597|
|Allemagne|    3.816666666666667|
|  Pologne|    4.095238095238095|
+---------+---------------------+
```

Interprétation:

Ce KPI permet de comprendre la taille moyenne des familles dans chaque pays. Une valeur plus élevée pourrait indiquer une culture où les familles ont tendance à avoir plus d'enfants, tandis qu'une valeur plus basse pourrait indiquer le contraire.

Moyenne d'âge par activité :

```
scala> val avgAgeByActivity = dfTyped.groupBy("Activite").agg(avg("Age").alias("AvgAgeByActivity"))
avgAgeByActivity: org.apache.spark.sql.DataFrame = [Activite: string, AvgAgeByActivity: double]

scala> // Calcul de la moyenne d'??ge pour chaque activit??

scala> avgAgeByActivity.show()
+--------------+------------------+
|      Activite|  AvgAgeByActivity|
+--------------+------------------+
|  Informatique| 41.77832512315271|
|    Clientelle| 40.78431372549019|
|    Logistique|41.512953367875646|
|    Production|40.445544554455445|
|        Autres|40.876344086021504|
|     Marketing| 40.37931034482759|
|Administration|41.611702127659576|
|       Support|            40.305|
|     Transport| 41.09090909090909|
|    Commercial|41.049751243781095|
+--------------+------------------+
```

Interprétation:

Cette analyse fournit un aperçu de l'âge moyen des individus pratiquant différentes activités sportives ou de loisirs. Cela peut aider à mieux comprendre les préférences en fonction de l'âge.

Proportion de fumeurs par sexe :

```
scala> val smokerRatioBySex = dfTyped.groupBy("Sexe").agg((sum(when(col("Fumer") === "Oui", 1).otherwise(0)) / count("*
)).alias("SmokerRatio"))
smokerRatioBySex: org.apache.spark.sql.DataFrame = [Sexe: string, SmokerRatio: double]

scala> // Calcul de la proportion de fumeurs pour chaque sexe

scala> smokerRatioBySex.show()
+-----+------------------+
| Sexe|       SmokerRatio|
+-----+------------------+
|Femme|0.4895549500454133|
|Homme|0.5061179087875417|
+-----+------------------+
```

Interpétation:

Cette analyse permet de visualiser la répartition des fumeurs par sexe. Cela peut fournir des informations sur les différences de comportement liées au tabagisme entre les hommes et les femmes.

Moyenne de livres lus par pays :

```
scala> // Calcul de la moyenne du nombre de livres lus par pays dans le DataFrame

scala> avgBooksByCountry.show()
+-----------+------------------+
|       Pays| AvgBooksByCountry|
+-----------+------------------+
|   Autriche|12.801724137931034|
|    Hongrie|14.181818181818182|
|   Finlande|             12.88|
|   Bulgarie|              12.0|
|     France|12.153846153846153|
|    Espagne| 12.48421052631579|
|    Croatie|12.082474226804123|
|     Suisse|12.402061855670103|
|  Allemagne|            12.275|
|    Pologne|              13.0|
|Royaume Uni|11.795698924731182|
|      Suede|13.096774193548388|
|     Italie|12.224137931034482|
|  Slovaquie|13.754545454545454|
|    Irlande| 13.12621359223301|
|     Serbie|11.512820512820513|
|   Belgique|11.495575221238939|
|   Slovenie|13.150442477876107|
|   Portugal|12.670588235294117|
|      Grece|13.780701754385966|
+-----------+------------------+
```

Interprétation : Cette analyse fournit une vue d'ensemble de la consommation moyenne de livres par pays. Elle permet de comprendre les habitudes de lecture dans différentes régions du monde.

Nombre moyen d'heures passées devant la télévision par sexe :

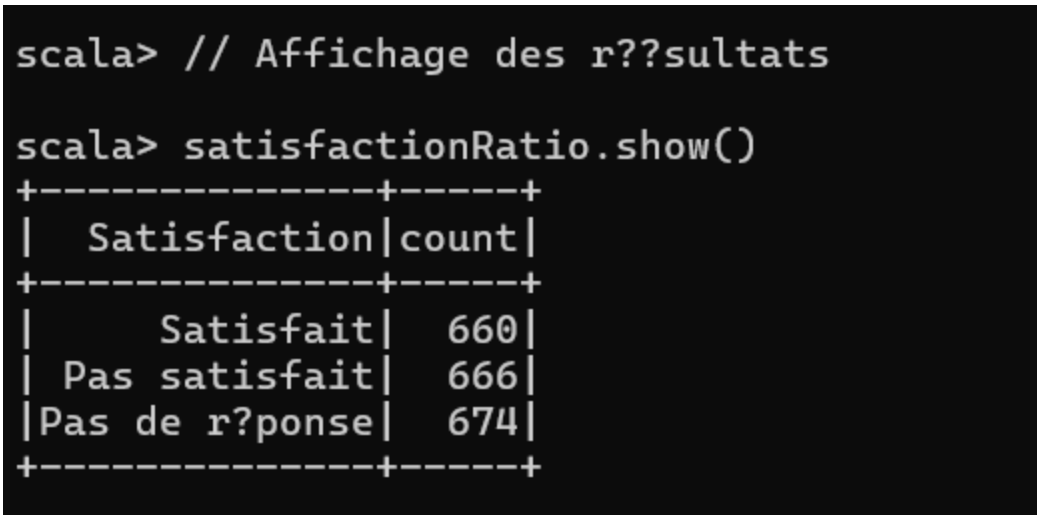
```
scala> val avgTVHoursBySex = dfTyped.groupBy("Sexe").agg(avg("HeuresTV").alias("AvgTVHoursBySex"))
avgTVHoursBySex: org.apache.spark.sql.DataFrame = [Sexe: string, AvgTVHoursBySex: double]

scala> // Calcul de la moyenne du nombre d'heures pass??es devant la t??l??vision par sexe dans le DataFrame

scala> avgTVHoursBySex.show()
+------+-----------------+
| Sexe|  AvgTVHoursBySex|
+------+-----------------+
|Femme|6.442325158946413|
|Homme|6.509454949944383|
+------+-----------------+
```

Interprétation : Cette analyse permet de comprendre la moyenne du temps passé devant la télévision pour chaque sexe. Elle peut révéler des différences de comportement en matière de visionnage de télévision entre les hommes et les femmes.

Ratio de personnes satisfaites par rapport aux personnes insatisfaites :



```
scala> // Affichage des r??sultats

scala> satisfactionRatio.show()
+--------------+-----+
|  Satisfaction|count|
+--------------+-----+
|     Satisfait|  660|
| Pas satisfait|  666|
|Pas de r?ponse|  674|
+--------------+-----+
```

Interprétation : Cette KPI vous donne un aperçu de la répartition des personnes satisfaites et insatisfaites dans votre ensemble de données, en présentant le ratio de personnes satisfaites par rapport aux personnes insatisfaites. Cela peut être utile pour évaluer le niveau de satisfaction global dans votre population d'étude.

Conclusion

En conclusion, cette analyse nous donne un aperçu précieux des tendances et des caractéristiques de la population étudiée. Il serait bénéfique d'approfondir l'analyse pour comprendre les facteurs sous-jacents qui influent sur le bonheur, la satisfaction et les comportements des individus. Ces informations pourraient être utilisées pour informer les politiques et les interventions visant à améliorer le bien-être et la qualité de vie des populations concernées.