2020

Rendu Projet SPARK

CHABCHOUB Aymen
EL YADINI Said
LIN Dixuan
YOUSFI Omar
ESGI 4IABD CLASSE 2 |

Rendu Projet

1.Recupération des données

```
package fr.esgi.spark


import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions.round
import org.apache.spark.sql.types.IntegerType
object data3 {

  def main(args: Array[String]): Unit = {


      val spark = SparkSession.builder()
        .appName("Spark SQL TD1")
        .config("spark.driver.memory","512m")
        .master("local[8]")
        .getOrCreate()

      import spark.implicits._

      val data_items = spark.read
        .option("inferSchema", "true")
        .option("header", "true")
        .option("delimiter",";")
        .format("csv")
        .load("D:/Document D/COURS ESGI/SPARK/items.csv")

      val data_clients = spark.read
        .option("inferSchema", "true")
        .option("header", "true")
        .option("delimiter",";")
        .format("csv")
        .load("D:/Document D/COURS ESGI/SPARK/clients.csv")

      val data_transaction = spark.read
        .option("inferSchema", "true")
        .option("header", "true")
        .option("delimiter",";")
        .format("csv")
        .load("D:/Document D/COURS ESGI/SPARK/transaction.csv")
```

2.Fusion des datas

- Fusion des tables transaction et item. Puis renommage et création de 4 nouvelles colonnes (jour mois, jour semaine, mois, année) à partir de la conne **Date_Of_Transaction.**

```
val data_fusionItemTransaction_tmp =
data_items.join(data_transaction,data_items("id")===
data_transaction("Item_Id"),"inner")

val data_fusionItemTransaction  = data_fusionItemTransaction_tmp
    .select(col("Id") as "id item",col("Item_Name"),col("Category") as
"Category_item",col("Category_Id"),col("Item_Buyed_Price"),
col("Item_Selling_Price"),col("Transaction_Id"),col("Client_id"),col("Item_Id"),col("
Number"),year(col("Date_Of_Transaction")) as "annee",
      month(col("Date_Of_Transaction")) as
"mois",dayofmonth(col("Date_Of_Transaction")) as "jour
mois",dayofweek(col("Date_Of_Transaction")) as "jour semaine",
col("Date_Of_Transaction"),
      col("Shop_Name"),col("Shop_Id"))
```

- Fusion de la table précédemment fusionner avec la table Vendeur. Le but est d'avoir différente table pour pourvoir faire différente analyse en fonction de ce que nous recherchons.

```
val allFusion = data_clients.join(data_fusionItemTransaction,data_clients("id")===
data_fusionItemTransaction("Client_Id"),"inner")
```

3. Analyse

1.1)Nombre Article vendu par mois

```
data_fusionItemTransaction
.groupBy(col("annee"),col("mois"))
.agg(sum(col("Number")))
.orderBy(col("annee"),col("mois"))
.show(Int.MaxValue)
```

annee	mois	sum(Number)
2019	1	237035
2019	2	302365
2019	3	334926
2019	4	323210
2019	5	334627
2019	6	323430
2019	7	333096

2019	8	333773
2019	9	325148
2019	10	334321
2019	11	323951
2019	12	334382
2020	1	96672
2020	2	224
2020	3	217
2020	4	229
2020	5	221
2020	6	255
2020	7	262
2020	8	250
2020	9	266
2020	10	265
2020	11	231
2020	12	235
2021	1	82

1.2) Nombre de transaction par mois.

annee	mois	sum(Number)
2019	1	237035
2019	2	302365
2019	3	334926
2019	4	323210
2019	5	334627
2019	6	323430
2019	7	333096
2019	8	333773
2019	9	325148
2019	10	334321
2019	11	323951
2019	12	334382
2020	1	96672
2020	2	224
2020	3	217
2020	4	229
2020	5	221
2020	6	255
2020	7	262
2020	8	250

2020	9	266
2020	10	265
2020	11	231
2020	12	235
2021	1	82

- ⇨ Seule l'année 2019 semble être complète
- ⇨ Nous avons donc choisi de nous concentrer sur cette année

2) Moyenne article vendu par transaction par mois en 2019

```
(data_fusionItemTransaction
  .groupBy(col("annee"), col("mois"))
  .agg(mean(col("Number")))
  .orderBy(col("mois"))
  .where (col("annee") === 2019))
.show(Int.MaxValue)
```

année	mois	avg(Number)
2019	1	2.9937355545170945
2019	2	2.998998234512309
2019	3	3.000806364906999
2019	4	2.994404195000834
2019	5	2.999928280066341
2019	6	3.005128872205601
2019	7	2.999189641821685
2019	8	3.002987035187634
2019	9	3.0079280646086386
2019	10	3.0036206493809856
2019	11	2.998213756848808
2019	12	3.000233284283817

3) Prix moyen d'une transaction en 2019

```
val tmp = allFusion
  .where($"annee" ===2019)
  .agg(mean(col("Item_Selling_Price")*col("Number")) as "Vente")

tmp.withColumn("Vente",round(tmp("Vente"),2)).show(Int.MaxValue)
```

Vente moyenne
181,60 €

4) Prix moyen d'une transaction en 2019 par magasin

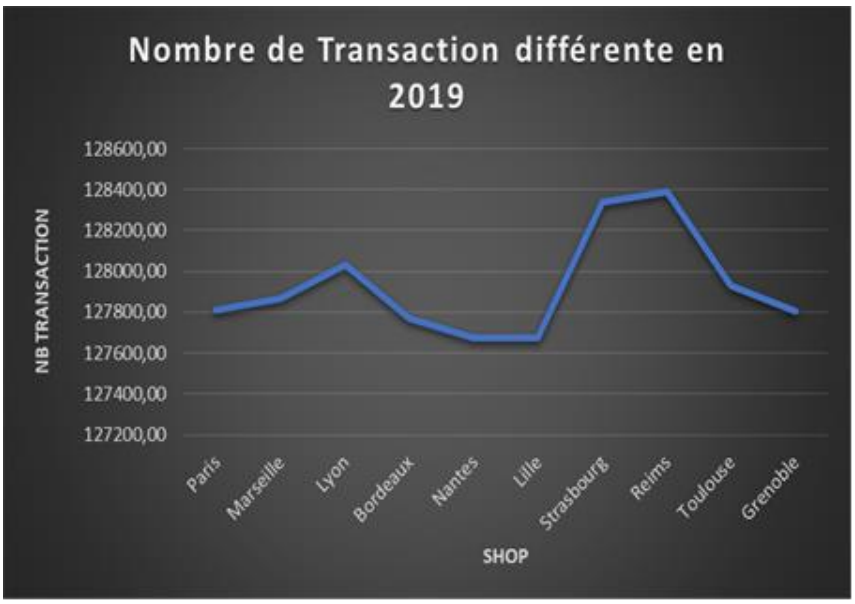
```
val tmp = allFusion
  .where($"annee" ===2019)
  .groupBy($"Shop_id",$"Shop_Name" )
  .agg(mean(col("Item_Selling_Price")*col("Number")) as "Vente")
  .orderBy($"Shop_Id")
```

Shop id	Shop Name	Vente moyenne
1	Paris	182,17 €
2	Marseille	181,53 €
3	Lyon	181,39 €
4	Bordeaux	181,38 €
5	Nantes	181,94 €
6	Lille	181,57 €
7	Strasbourg	181,26 €
8	Reims	181,62 €
9	Toulouse	181,34 €
10	Grenoble	181,82 €

5) Nombre de transaction différente en 2019 par magasin

```
allFusion
  .where($"annee" ===2019)
  .groupBy($"Shop_id",$"Shop_Name" )
  .agg(countDistinct($"Transaction_Id"))
    .orderBy($"Shop_Id")
    .show(Int.MaxValue)
```

Shop_id	Shop_Name	count(DISTINCT Transaction_Id)
1	Paris	127813,00
2	Marseille	127866,00
3	Lyon	128035,00
4	Bordeaux	127769,00
5	Nantes	127676,00
6	Lille	127675,00
7	Strasbourg	128341,00
8	Reims	128388,00
9	Toulouse	127929,00
10	Grenoble	127805,00



6) Nombre moyen de transaction différente par jours en 2019

```
val tmp = allFusion
  .where($"annee"===2019)
  .groupBy($"Shop_id",$"Shop_Name" , $"mois",$"jour mois")
  .agg(countDistinct($"Transaction_Id"))
    .orderBy($"Shop_Id")

tmp.agg(mean($"count(DISTINCT Transaction_Id)")as ).show()
```

Moyenne de transaction par jour
358,35

7) Nombre moyen de transaction différente par jours en 2019 par magasin

```
val tmp = allFusion
  .where($"annee" ===2019)
  .groupBy($"Shop_id",$"Shop_Name" , $"mois",$"jour mois")
  .agg(countDistinct($"Transaction_Id"))
    .orderBy($"Shop_Id")

tmp.groupBy($"Shop_id",$"Shop_Name").agg(mean($"count(DISTINCT
Transaction_Id)")).show()
```

Shop Id	Shop Name	Moyenne transaction
1	Paris	358,02
2	Marseille	358,17
3	Lyon	358,64
4	Bordeaux	357,90
5	Nantes	357,64
6	Lille	357,63
7	Strasbourg	359,50
8	Reims	359,50
9	Toulouse	359,50
10	Grenoble	359,50

8) La transaction la plus chère. (Plusieurs transactions sont égales, donc on en affiche 10)

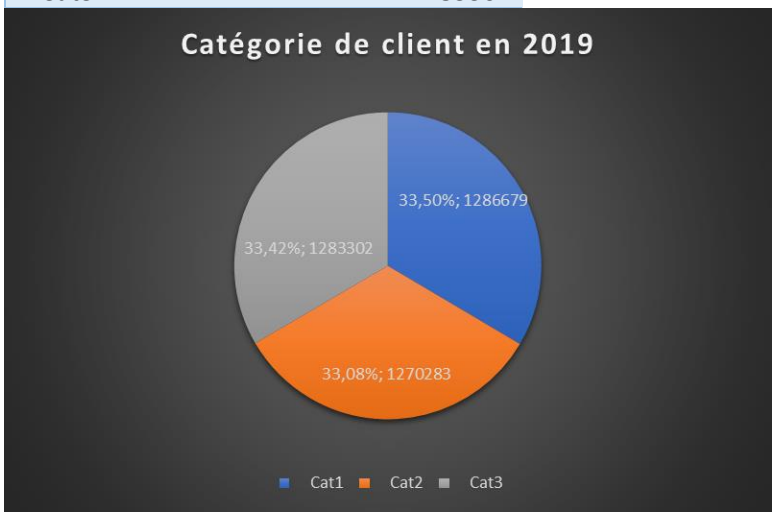
```
allFusion.createOrReplaceTempView("data1")
val sql_variable = spark.sql(""" Select * from data1
   | where (Item_Selling_Price*Number)=(SELECT max( Item_Selling_Price*Number) FROM data1)
  """.stripMargin)

sql_variable.show(10,false)
```

id	Name	First_Name	Address	Date	_CreationC	ategor	item	Item_Na	Categgory_	uye	Selling_	Transac	ient_id	tem_Id	mber	annee	mois	mo	semaine	Date	ansaction	Shop_Na	Sho
6684	Dos Santos	ThierryThierry	12 48 66 4520		09:59:46	Cat3	7204	Breton	UNKN	5	99	113.85	36888	6684	7204	5	2019	6	17	22019-06-17	01:50:50	Grenoble	10
9314	Fouquet	AndréAndré.F	84 43 36 8620		01:47:27	Cat3	1187	Chevallie	Furnit	5	99	113.85	37606	9314	1187	5	2019	9	30	22019-09-30	16:37:36	Marseille	2
6021	Thierry	ThibautThibau	05867343642		14:33:48	Cat2	7204	Breton	UNKN	5	99	113.85	39885	6021	7204	5	2019	2	4	22019-02-04	08:02:04	Nantes	5
8245	Goncalves	JulesJules.Gor	84 00 91 8320		13:38:50	Cat3	6096	Germai	UNKN	6	99	113.85	45447	8245	6096	5	2019	3	22	62019-03-22	03:25:41	Grenoble	10
3962	Pierre	LucyLucy.Pier	17 64 27 9420		12:28:50	Cat1	7162	Loiseau	Game	1	99	113.85	50290	3962	7162	5	2019	5	22	42019-05-22	19:20:47	Reims	8
474	Gomez	DenisDenis.Gc	34 71 07 7420		20:19:57	Cat3	6990	Rodrigue	Comp	1	99	113.85	52172	474	6990	5	2019	4	21	12019-04-21	17:50:04	Paris	1
3933	Thierry	MarieMarie.T	69 03 26 1720		20:34:02	Cat2	6096	Germai	UNKN	6	99	113.85	53959	3933	6096	5	2019	4	24	42019-04-24	04:55:00	Strasbou	7
8784	Boulay	GrégoireGrég	08285140632		22:19:59	Cat2	6096	Germai	UNKN	6	99	113.85	58592	8784	6096	5	2019	6	8	72019-06-08	00:15:43	Bordeaux	4
44	Monnier	LucieLucie.Mc	05 72 78 8320		09:51:35	Cat2	6096	Germai	UNKN	6	99	113.85	59112	44	6096	5	2020	1	2	52020-01-02	19:48:12	Marseille	2
5851	Millet	JeanneJeanne	05 02 66 0420		00:15:43	Cat3	7162	Loiseau	Game	1	99	113.85	63069	5851	7162	5	2019	11	26	32019-11-26	02:09:55	Lyon	3

9) Nombres d'articles achetés par catégories de clients en 2019

```
allFusion.createOrReplaceTempView("data")
val sql_variable = spark.sql(""" Select category ,sum(Number) from data where annee = 2019
group by category """.stripMargin)

sql_variable.show(Int.MaxValue)
```

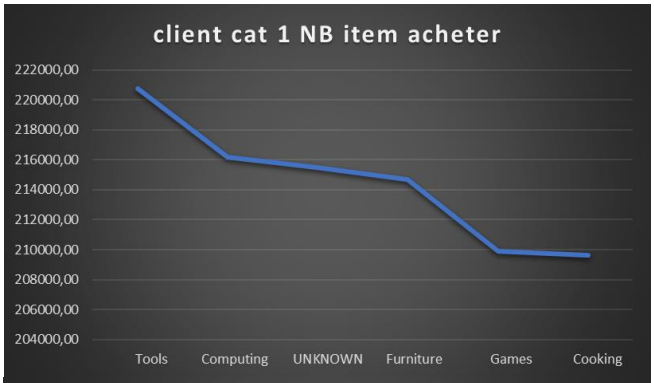
Catégory Clients	NB Article Acheté
Cat1	1286679
Cat2	1270283
Cat3	1283302

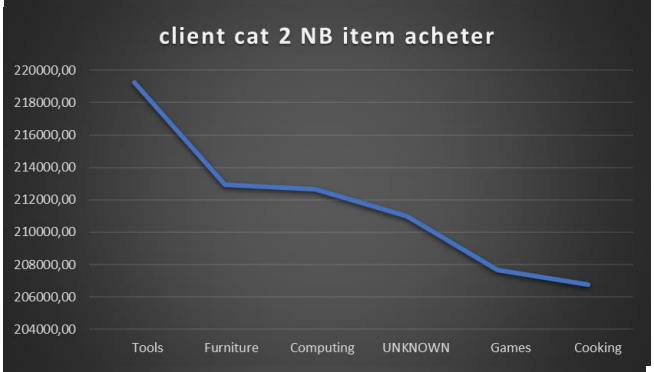


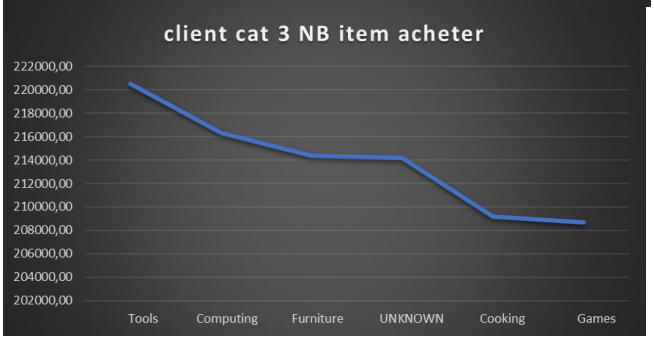
10) Nombres d'articles achetés par catégories items par catégories de clients.

```
allFusion.createOrReplaceTempView("data")
val sql_variable = spark.sql(""" Select category ,sum(Number),Category_item from data
 where annee = 2019 group by Category,Category_item order by category, sum(Number) desc"""
.stripMargin)

sql_variable.show(Int.MaxValue)
```

category	sum(Number)	Category_iten
Cat1	220764,00	Tools
Cat1	216200,00	Computing
Cat1	215500,00	UNKNOWN
Cat1	214685,00	Furniture
Cat1	209906,00	Games
Cat1	209624,00	Cooking
Cat2	219237,00	Tools
Cat2	212931,00	Furniture
Cat2	212665,00	Computing
Cat2	211014,00	UNKNOWN
Cat2	207684,00	Games
Cat2	206752,00	Cooking
Cat3	220494,00	Tools
Cat3	216301,00	Computing
Cat3	214405,00	Furniture
Cat3	214205,00	UNKNOWN
Cat3	209196,00	Cooking
Cat3	208701,00	Games







11.1) Capital de vente en 2019 par mois.

```
val tmp = (data_fusionItemTransaction
    .groupBy(col("annee"), col("mois"))
    .agg(sum(col("Item_Selling_Price")*col("Number")) as "somme")
    .orderBy(col("mois"))
    .where (col("annee") === 2019))
   // .show(Int.MaxValue)

tmp.withColumn("somme", format_number($"somme", 2)).show()
```

annee	mois	somme
2019	1	14344967,92
2019	2	18316861,44
2019	3	20245141,47
2019	4	19577317,54
2019	5	20261510,26
2019	6	19578059,20
2019	7	20122766,29
2019	8	20222867,59
2019	9	19691889,00
2019	10	20241776,27
2019	11	19606278,19
2019	12	20212403,34

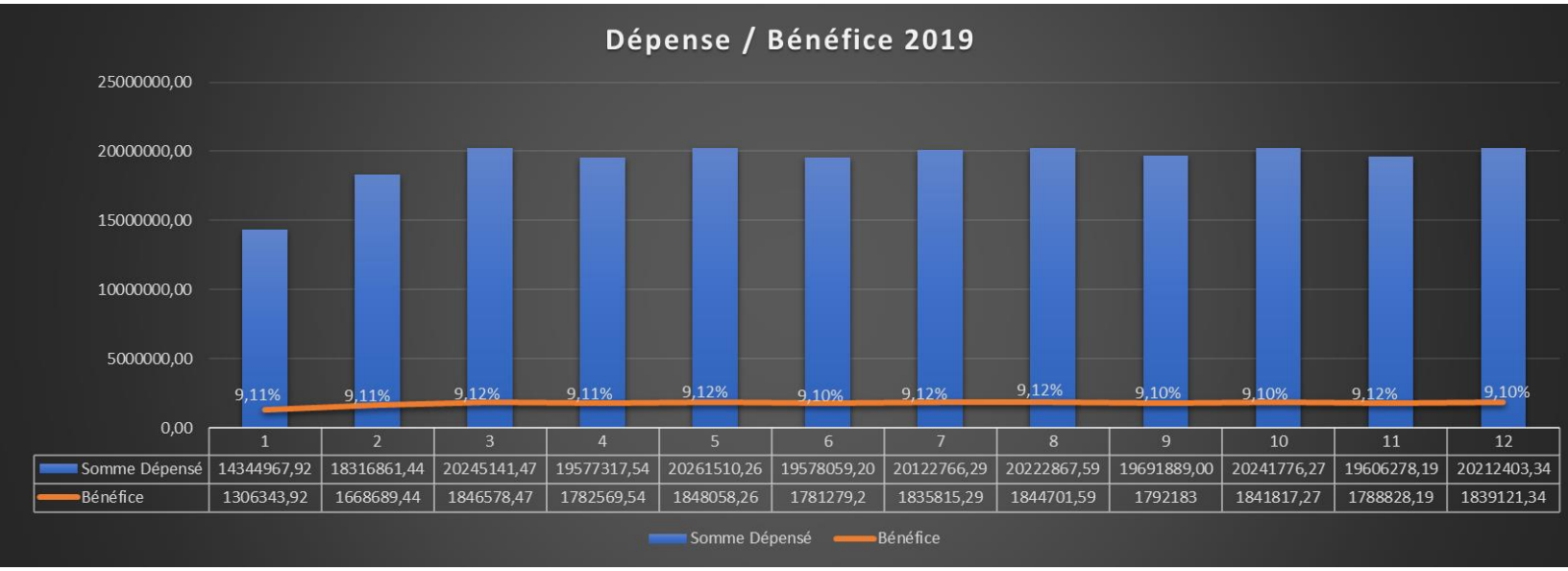
11.2) Bénéfice de vente en 2019 par mois

```
val tmp = (data_fusionItemTransaction
    .groupBy(col("annee"), col("mois"))
    .agg(sum(col("Item_Selling_Price")*col("Number") -
col("Item_Buyed_Price")*col("Number")
      ) as "benefice")
    .orderBy(col("mois"))
    .where (col("annee") === 2019))
   // .show(Int.MaxValue)

tmp.withColumn("benefice", format_number($"benefice", 2)).show()
```

annee	mois	benefice
2019	1	1306343,92
2019	2	1668689,44
2019	3	1846578,47
2019	4	1782569,54
2019	5	1848058,26
2019	6	1781279,2

2019	7	1835815,29
2019	8	1844701,59
2019	9	1792183
2019	10	1841817,27
2019	11	1788828,19
2019	12	1839121,34

➔



	1	2	3	4	5	6	7	8	9	10	11	12
Somme Dépensé	14344967,92	18316861,44	20245141,47	19577317,54	20261510,26	19578059,20	20122766,29	20222867,59	19691889,00	20241776,27	19606278,19	20212403,34
Bénéfice	1306343,92	1668689,44	1846578,47	1782569,54	1848058,26	1781279,2	1835815,29	1844701,59	1792183	1841817,27	1788828,19	1839121,34

12.1) Capitale de vente en 2019 par boutique

```
val tmp = (data_fusionItemTransaction
  .where (col("annee") === 2019))
  .groupBy(col("Shop_Id"), col("Shop_Name"))
  .agg(sum(col("Item_Selling_Price")*col("Number")) as "Vente totale")
  .orderBy(col("Shop_Id"))

tmp.withColumn("Vente",round(tmp("Vente totale"),2)).show()
```

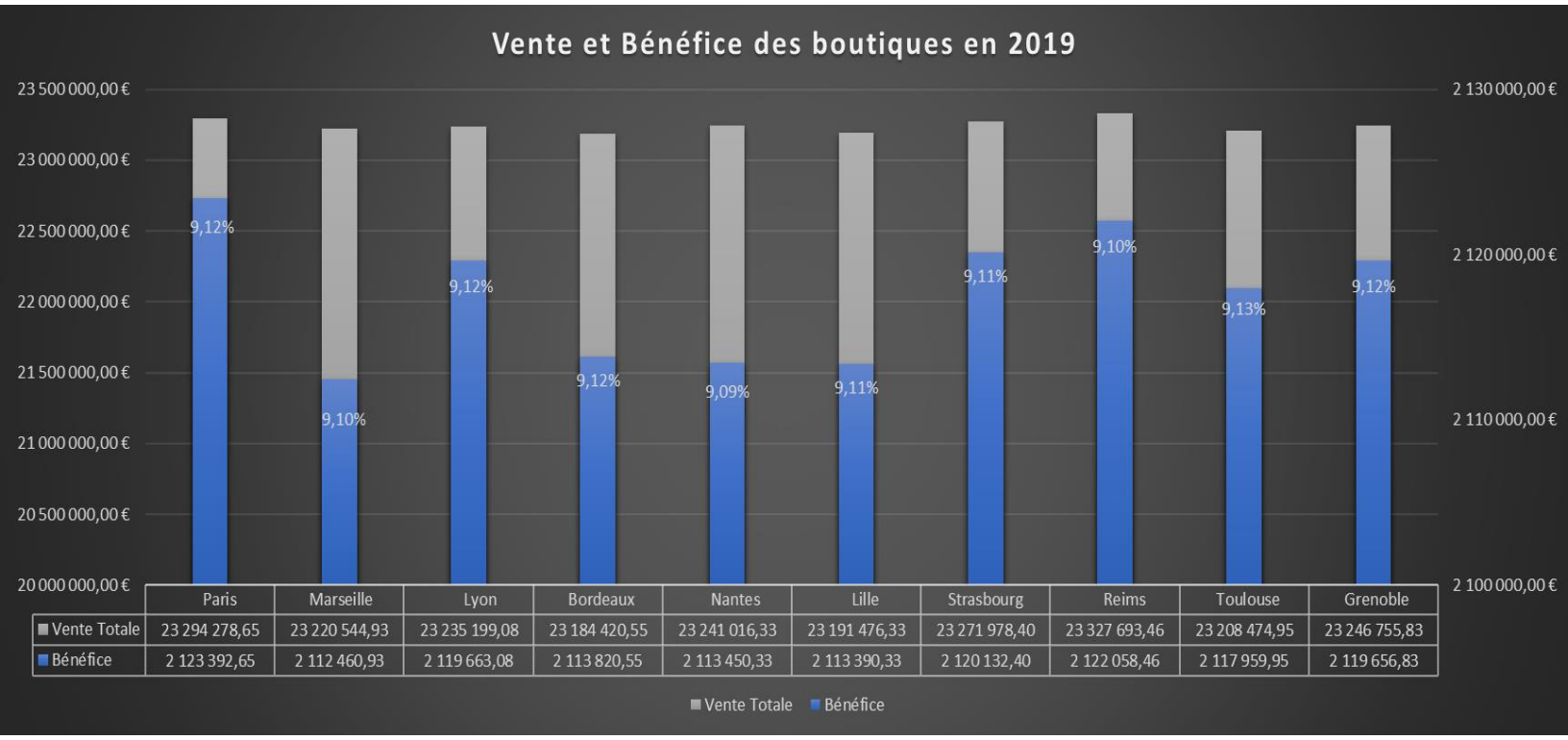
Shop_Id	Shop_Name	Vente totale
1	Paris	23 294 278,65
2	Marseille	23 220 544,93
3	Lyon	23 235 199,08
4	Bordeaux	23 184 420,55
5	Nantes	23 241 016,33
6	Lille	23 191 476,33
7	Strasbourg	23 271 978,40
8	Reims	23 327 693,46
9	Toulouse	23 208 474,95
10	Grenoble	23 246 755,83

12.2) Bénéfice de vente en 2019 par boutique

```
val tmp = (data_fusionItemTransaction
  .where (col("annee") === 2019))
  .groupBy(col("Shop_Id"), col("Shop_Name"))
  .agg(sum(col("Item_Selling_Price")*col("Number") -
col("Item_Buyed_Price")*col("Number")
  ) as "benefice")
  .orderBy(col("Shop_Id"))

tmp.withColumn("benefice",round(tmp("benefice"),2)).show()
```

Shop_Id	Shop_Name	benefice
1	Paris	2 123 392,65
2	Marseille	2 112 460,93
3	Lyon	2 119 663,08
4	Bordeaux	2 113 820,55
5	Nantes	2 113 450,33
6	Lille	2 113 390,33
7	Strasbourg	2 120 132,40
8	Reims	2 122 058,46
9	Toulouse	2 117 959,95
10	Grenoble	2 119 656,83

➔



13) Capitale de vente en 2019 par mois dans la ville de Paris par catégorie d'items

```
val tmp2 = data_fusionItemTransaction
  .where (col("annee") === 2019 && col("Shop_Name")==="Paris")
  .groupBy(col("mois"),col("Shop_Id"),
col("Shop_Name"),col("Category_Id"),col("Category_item"))
  .agg(sum(col("Item_Selling_Price")*col("Number")    ) as "Vente")
  .orderBy(col("Category_Id"),col("mois"),col("Shop_Id"))

tmp2.withColumn("Vente",tmp2("Vente").cast(IntegerType)).show(Int.MaxValue)
```

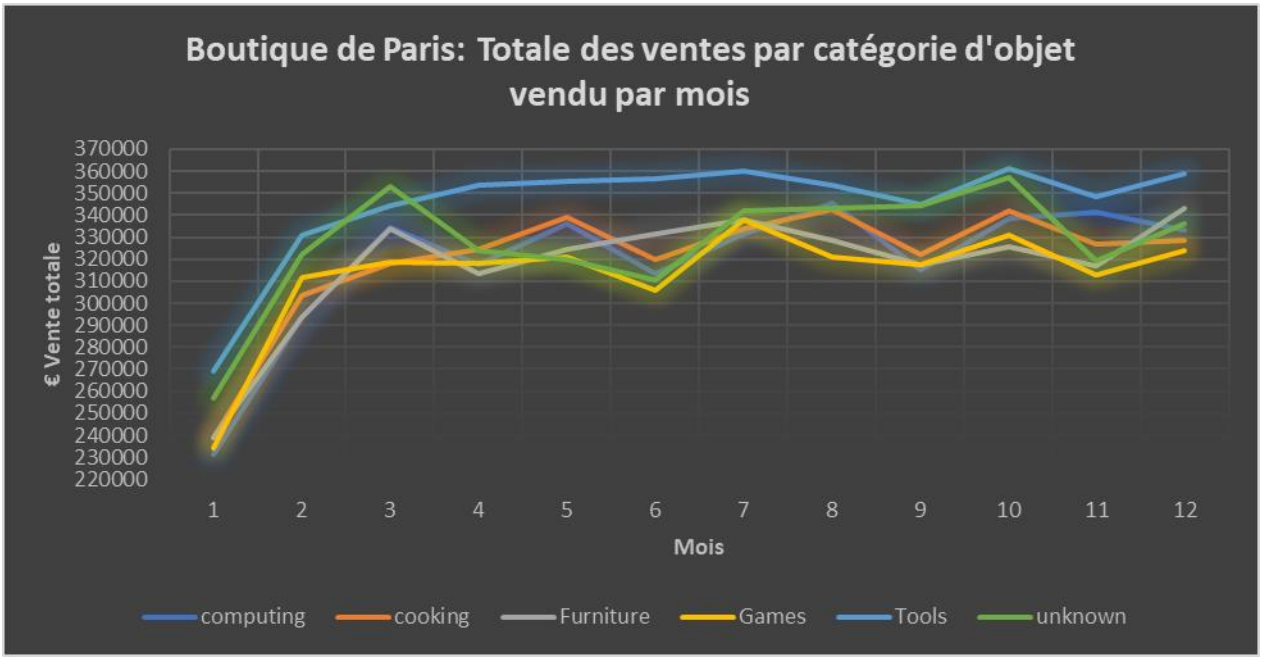
mois	Shop_Id	Shop_Name	Category	Vente
1	1	Paris	Computing	231448
2	1	Paris	Computing	293759
3	1	Paris	Computing	334268
4	1	Paris	Computing	317636
5	1	Paris	Computing	335968
6	1	Paris	Computing	313430
7	1	Paris	Computing	331470
8	1	Paris	Computing	345451

9	1	Paris	Computing	314870
10	1	Paris	Computing	338682
11	1	Paris	Computing	341520

…

…

8	1	Paris	UNKNOWN	343188
9	1	Paris	UNKNOWN	344281
10	1	Paris	UNKNOWN	357065
11	1	Paris	UNKNOWN	319276
12	1	Paris	UNKNOWN	336046



14) Bénéfice de vente en 2019 à Paris par catégories d'items

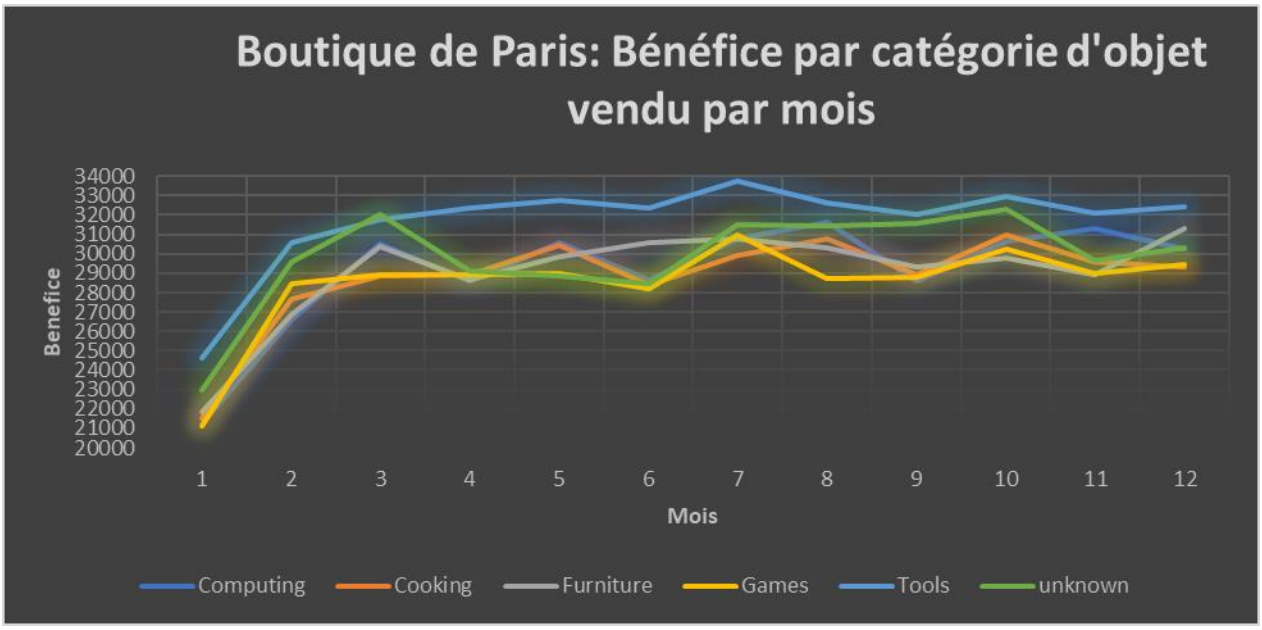
```
val tmp2 = data_fusionItemTransaction
  .where (col("annee") === 2019 && col("Shop_Name")==="Paris")
  .groupBy(col("mois"),col("Shop_Id"), col("Shop_Name"),col("Category_item"))
  .agg(sum(col("Item_Selling_Price")*col("Number") -
col("Item_Buyed_Price")*col("Number")    ) as "Vente")
  .orderBy(col("Category_item"),col("mois"),col("Shop_Id"))

tmp2.withColumn("Vente",tmp2("Vente").cast(IntegerType)).show(Int.MaxValue)
```

mois	Shop_Id	Shop_Name	Category	Vente
1	1	Paris	Computing	21235
2	1	Paris	Computing	26633
3	1	Paris	Computing	30532
4	1	Paris	Computing	28602
5	1	Paris	Computing	30590
6	1	Paris	Computing	28654
7	1	Paris	Computing	30793
8	1	Paris	Computing	31628

…

…

8	1	Paris	UNKNOWN	31443
9	1	Paris	UNKNOWN	31562
10	1	Paris	UNKNOWN	32290
11	1	Paris	UNKNOWN	29642
12	1	Paris	UNKNOWN	30297



15) **Nombres items vendu a paris par catégories items**

```
val tmp2 = data_fusionItemTransaction
  .where (col("annee") === 2019 && col("Shop_Name")==="Paris")
  .groupBy(col("mois"),col("Shop_Id"), col("Shop_Name"),col("Category_item"))
  .agg(sum(ccol("Number")) as "NBVente")
  .orderBy(col("Category_item"),col("mois"),col("Shop_Id"))
tmp2.withColumn("NBVente",tmp2("NBVente").cast(IntegerType)).show(Int.MaxValue)
```

mois	Shop_Id	Shop_Name	Category	NBVente
1	1	Paris	Computing	3953
2	1	Paris	Computing	5031
3	1	Paris	Computing	5524
4	1	Paris	Computing	5351
5	1	Paris	Computing	5625
6	1	Paris	Computing	5342
7	1	Paris	Computing	5625
8	1	Paris	Computing	5784

…

9	1	Paris	UNKNOWN	5550
10	1	Paris	UNKNOWN	5813
11	1	Paris	UNKNOWN	5140
12	1	Paris	UNKNOWN	5483

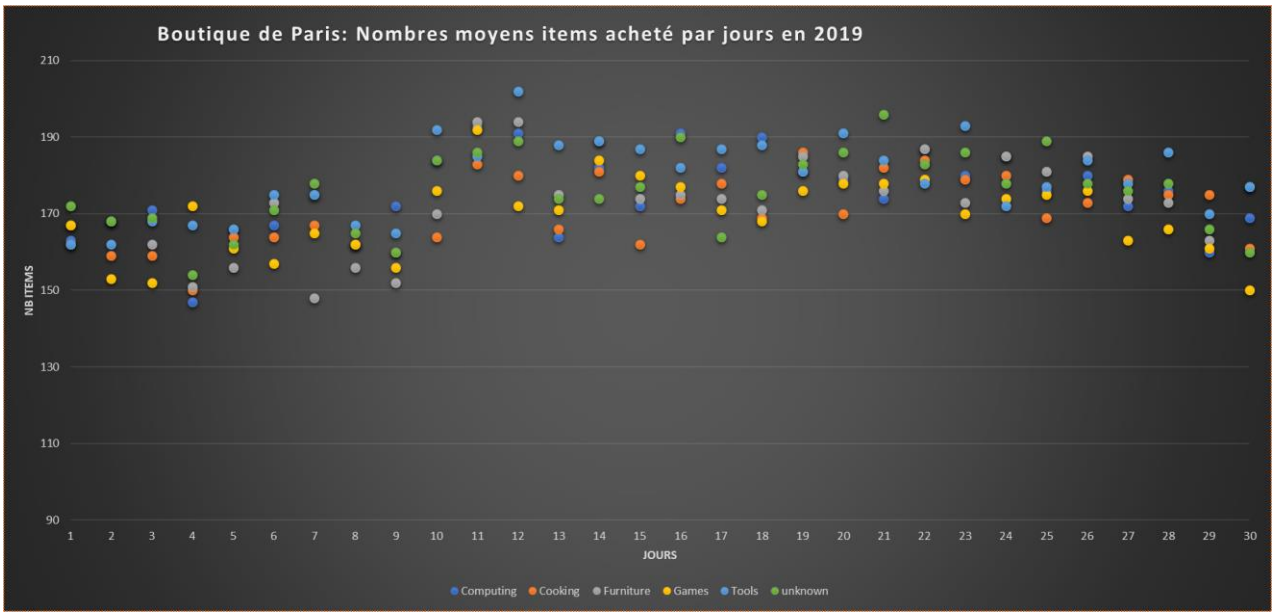
16) Nombres moyens items acheté par jours(1-30) en 2019 à Paris

```
val tmp2 = data_fusionItemTransaction
  .where (col("annee") === 2019 && col("Shop_Name")==="Paris")
  .groupBy(col("jour mois"),col("Shop_Id"), col("Shop_Name"),col("Category_item"))
  .agg(sum(col("Number")) as "NBVente")
  .orderBy(col("Category_item"),col("jour mois"),col("Shop_Id"))

tmp2.withColumn("NBVente",(tmp2("NBVente")/12).cast(IntegerType)).show(Int.MaxValue)
```

jour mois	Shop_Id	Shop_Name	Category	NBVente
1	1	Paris	Computing	163
2	1	Paris	Computing	168
3	1	Paris	Computing	171
4	1	Paris	Computing	147
5	1	Paris	Computing	166
6	1	Paris	Computing	167

…

28	1	Paris	UNKNOWN	178
29	1	Paris	UNKNOWN	166
30	1	Paris	UNKNOWN	160



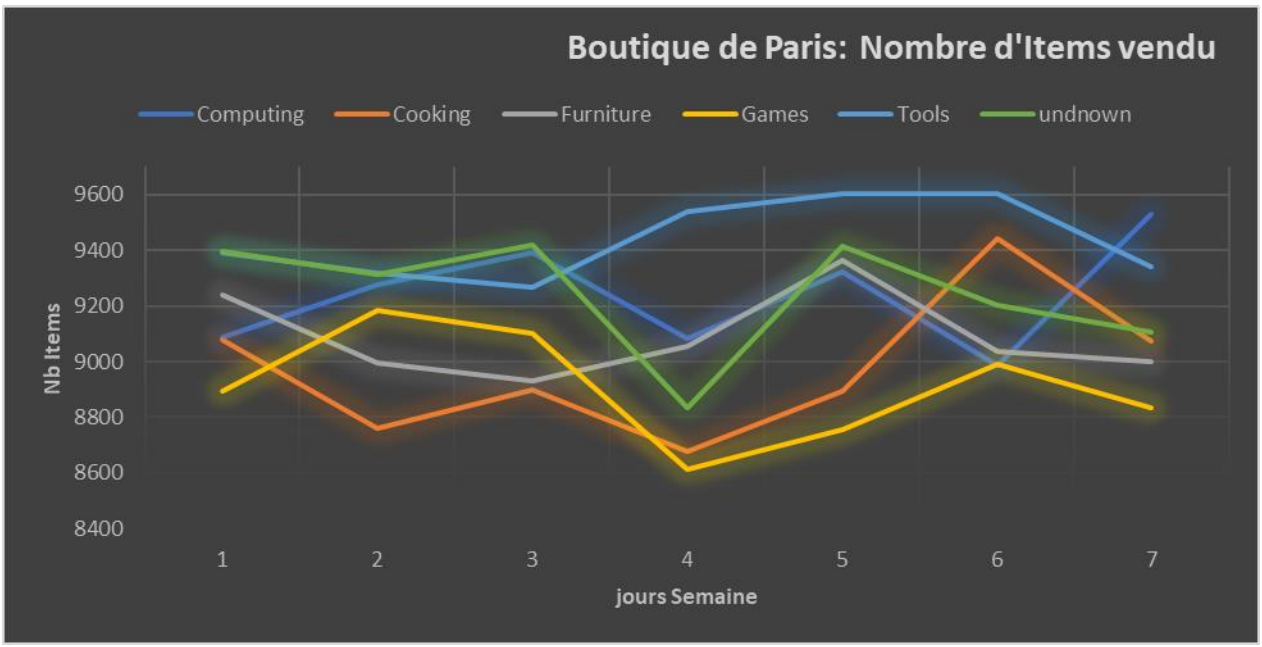
17) Nombres moyens items acheté par jours(lundi, mardi, …, dimanche) en 2019 à Paris

```
val tmp2 = data_fusionItemTransaction
  .where (col("annee") === 2019 && col("Shop_Name")==="Paris")
  .groupBy(col("jour mois"),col("Shop_Id"), col("Shop_Name"),col("Category_item"))
  .agg(sum(col("Number")) as "NBVente")
  .orderBy(col("Category_item"),col("mois"),col("Shop_Id"))

tmp2.withColumn("Vente",tmp2("Vente").cast(IntegerType)).show(Int.MaxValue)
```

jour semaine	Shop_Id	Shop_Name	Category	NBVente
1	1	Paris	Computing	9088
2	1	Paris	Computing	9278
3	1	Paris	Computing	9392
4	1	Paris	Computing	9082
5	1	Paris	Computing	9321
6	1	Paris	Computing	8984

…

3	1	Paris	UNKNOWN	9418
4	1	Paris	UNKNOWN	8836
5	1	Paris	UNKNOWN	9413
6	1	Paris	UNKNOWN	9202
7	1	Paris	UNKNOWN	9107


Boutique de Paris: Nombre d'Items vendu
Computing
Cooking
Furniture
Games
Tools
undnown
9600
9400
9200
9000
8800
8600
8400
Nb Items
1
2
3
4
5
6
7
jours Semaine