**Requêtes**

1. **Requêtes : Sélection**
2. Liste des collections de la base de données et nombre de documents

Via python et librairie pymongo

import pymongo

import pprint

connex = pymongo.MongoClient("mongodb://127.0.0.1:27017/")

db = connex.LBC\_db1

list\_collections = db.list\_collection\_names()

for collection in list\_collections:

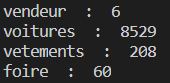
print(collection, " : ", db[collection].estimated\_document\_count())

Librairies importées

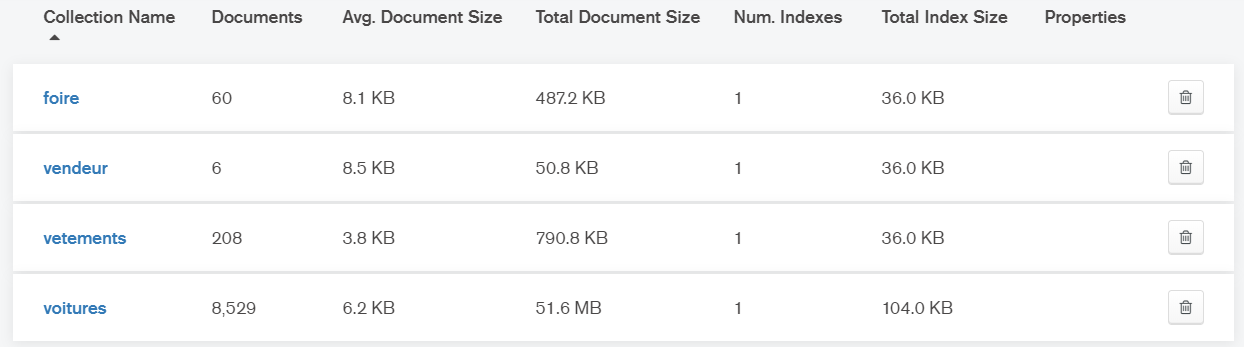
Connexion à la base de données mongoDB

Affichage du résultat

Le résultat affiché est :



Via Compass (interface graphique de mongoDB) :



1. Liste des vendeurs d’une collection (exemple : foire)

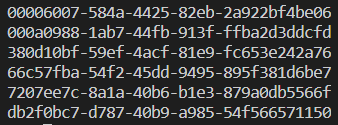
Via python et librairie pymongo

collection = "foire"

for vendeur in db[collection].distinct("owner.user\_id"):

print(vendeur)

Le résultat affiché est :



Via Compass (interface graphique de mongoDB) :

Collection

Requête

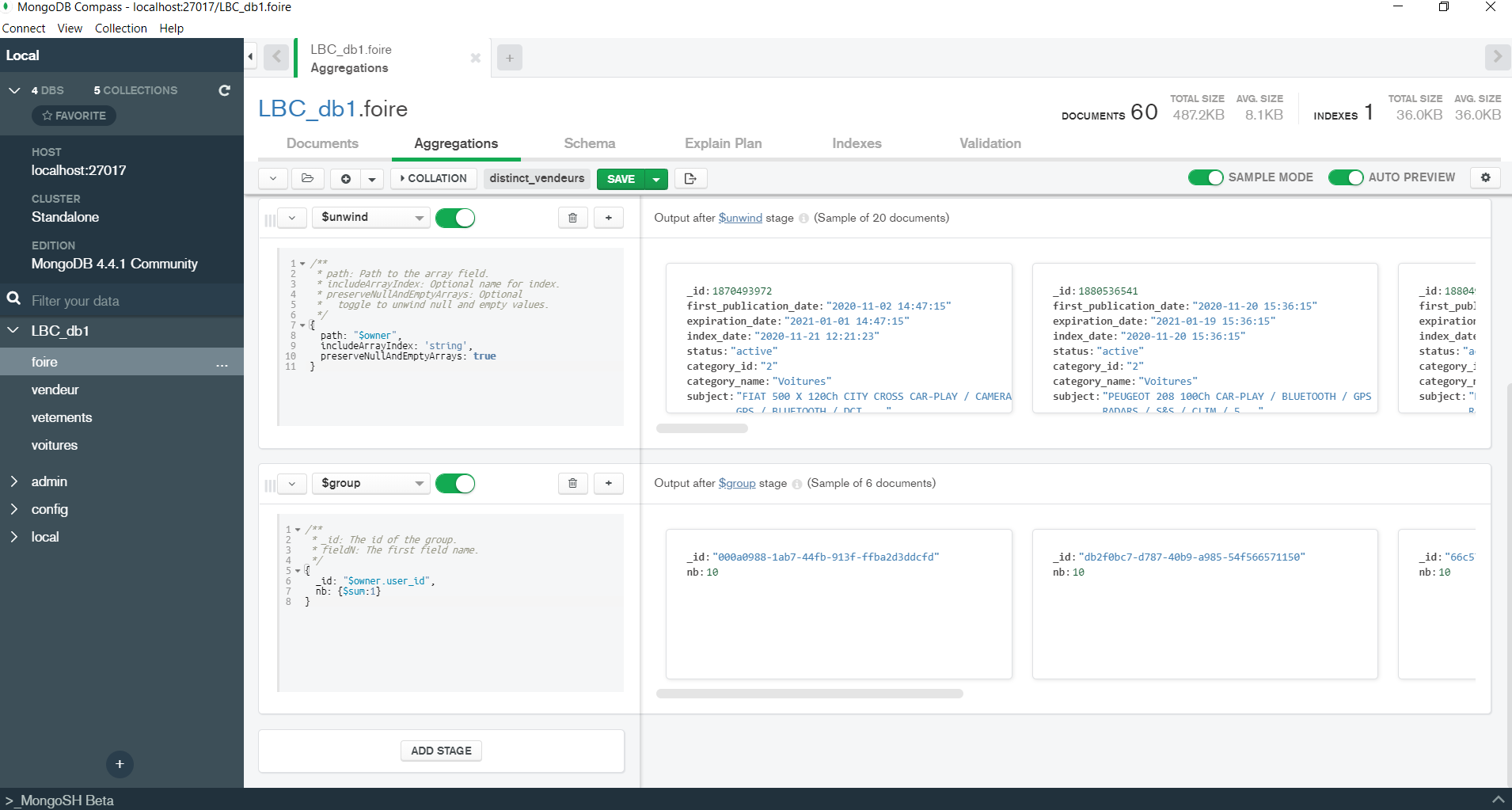
Résultat n°1 :

Identifiant du vendeur

Nombre d’annonces associées à ce vendeur

Résultat n°2

Nombre de résultats



1. **Requêtes : plus complexes**
2. Moyenne des prix des voitures

collection = "voitures"

res = db[collection].aggregate([

{

"$unwind" : "$price"

},

{

"$group": {

"\_id" : "$\_id",

"price0" : {"$first" : "$price"}

}

},

{

"$group": {

"\_id": 0,

"nb": {"$sum": 1},

"prix\_moyen" : {"$avg" : "$price0"}

}

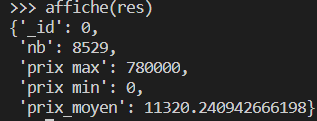
}])

Le résultat affiché est :



On pourrait nettoyer les données en étudiant plus précisément la distribution de la variable prix pour supprimer les valeurs aberrantes.

En modifiant la requête, on fait apparaître les prix minimum et maximum :



Il nous apparait clairement que le prix max (780000€) est aberrant. Il en est de même pour les prix minimums.

1. Map-Reduce : Liste des marques des voitures

Pour afficher l’ensemble des marques des voitures, on considère la structure des données. En effet, la marque est un attribut, contenu dans l’objet (liste) attributes. Tous les éléments de attributes ont la même structure : ce sont des dictionnaires avec des clés identiques.



La marque est donc la valeur en face de la clé « brand » contenue dans attributes.

Pour obtenir la liste des marques, on a donc pris le parti de faire une map-reduce :

collection = "voitures"

map = Code("function () {"

" for(var i=0;i<this.attributes.length;i++){"

" if(this.attributes[i].key == 'brand') {"

" emit(this.attributes[i].value, 1);}"

" }"

"}")

reduce = Code("function (key, values) {"

" return values.length;"

"}")

result = db[collection].map\_reduce(map, reduce, "map\_reduce\_brand")

for res in db.map\_reduce\_brand.find().sort([("value", pymongo. DESCENDING )]):

pprint.pprint(res)

On obtient le résultat suivant :

{'\_id': 'Renault', 'value': 1516.0}

{'\_id': 'Peugeot', 'value': 1311.0}

{'\_id': 'Citroen', 'value': 871.0}

{'\_id': 'Volkswagen', 'value': 647.0}

{'\_id': 'Bmw', 'value': 556.0}

{'\_id': 'Mercedes', 'value': 520.0}

{'\_id': 'Audi', 'value': 500.0}

{'\_id': 'Ford', 'value': 360.0}

{'\_id': 'Opel', 'value': 276.0}

{'\_id': 'Fiat', 'value': 217.0}

{'\_id': 'Nissan', 'value': 190.0}

{'\_id': 'Toyota', 'value': 146.0}

{'\_id': 'Seat', 'value': 144.0}

{'\_id': 'Mini', 'value': 133.0}

{'\_id': 'Dacia', 'value': 131.0}

{'\_id': 'Porsche', 'value': 102.0}

{'\_id': 'Alfa Romeo', 'value': 78.0}

{'\_id': 'Kia', 'value': 70.0}

{'\_id': 'Hyundai', 'value': 58.0}

{'\_id': 'Land Rover', 'value': 55.0}

{'\_id': 'Skoda', 'value': 53.0}

{'\_id': 'Volvo', 'value': 47.0}

{'\_id': 'Mazda', 'value': 44.0}

{'\_id': 'Suzuki', 'value': 42.0}

{'\_id': 'Chevrolet', 'value': 40.0}

{'\_id': 'Ds', 'value': 28.0}

{'\_id': 'Jeep', 'value': 28.0}

{'\_id': 'Mitsubishi', 'value': 27.0}

{'\_id': 'Jaguar', 'value': 26.0}

{'\_id': 'Honda', 'value': 24.0}

{'\_id': 'Lexus', 'value': 22.0}

{'\_id': 'Smart', 'value': 21.0}

{'\_id': 'Autres', 'value': 19.0}

{'\_id': 'Aixam', 'value': 16.0}

{'\_id': 'Abarth', 'value': 13.0}

{'\_id': 'Ligier', 'value': 13.0}

{'\_id': 'Dodge', 'value': 11.0}

{'\_id': 'Saab', 'value': 10.0}

{'\_id': 'Microcar', 'value': 9.0}

{'\_id': 'Chrysler', 'value': 8.0}

{'\_id': 'Ssangyong', 'value': 8.0}

{'\_id': 'Subaru', 'value': 6.0}

{'\_id': 'Rover', 'value': 6.0}

{'\_id': 'Lancia', 'value': 6.0}

{'\_id': 'Maserati', 'value': 5.0}

{'\_id': 'Lamborghini', 'value': 5.0}

{'\_id': 'Austin', 'value': 5.0}

{'\_id': 'Cadillac', 'value': 5.0}

{'\_id': 'Infiniti', 'value': 5.0}

{'\_id': 'Ford USA', 'value': 4.0}

{'\_id': 'Corvette', 'value': 4.0}

{'\_id': 'Tesla', 'value': 4.0}

{'\_id': 'Buick', 'value': 3.0}

{'\_id': 'Daihatsu', 'value': 3.0}

{'\_id': 'Aston Martin', 'value': 3.0}

{'\_id': 'Lotus', 'value': 3.0}

{'\_id': 'Mg', 'value': 2.0}

{'\_id': 'Isuzu', 'value': 2.0}

{'\_id': 'Alpine-Renault', 'value': 2.0}

{'\_id': 'Lada', 'value': 2.0}

{'\_id': 'Daewoo', 'value': 2.0}

{'\_id': 'Simpa JDM', 'value': 1.0}

{'\_id': 'Casalini', 'value': 1.0}

{'\_id': 'Alpine', 'value': 1.0}

{'\_id': 'Ferrari', 'value': 1.0}

{'\_id': 'Mega', 'value': 1.0}

{'\_id': 'Bertone', 'value': 1.0}

{'\_id': 'Chatenet', 'value': 1.0}

{'\_id': 'Cupra', 'value': 1.0}

{'\_id': 'Iveco', 'value': 1.0}

{'\_id': 'Pontiac', 'value': 1.0}

{'\_id': 'Triumph', 'value': 1.0}

Il y a au total 8478 voitures listées dans ce map reduce, or la collection contient 8529 voitures. Donc il y a 51 voitures qui n’avaient pas de marques répertoriées.

1. Moyenne des prix des voitures en fonction des marques

Le code est très similaire au précédent (2.b). Dans la fonction map, on renvoie le prix de la voiture, puis dans la fonction reduce, on calcule la moyenne des prix :

collection = "voitures"

map = Code("function () {"

" for(var i=0;i<this.attributes.length;i++){"

" if(this.attributes[i].key == 'brand') {"

" emit(this.attributes[i].value, this.price[0]);}"

" }"

"}")

reduce = Code("function (key, values) {"

" var total = 0;"

" for (var i = 0; i < values.length; i++) {"

" total += values[i];"

" }"

" return total/values.length;"

"}")

result = db[collection].map\_reduce(map, reduce, "map\_reduce\_price\_per\_brand")

for res in db.map\_reduce\_price\_per\_brand.find().sort([("value", pymongo. DESCENDING )]):

pprint.pprint(res)

On obtient le résultat suivant :

{'\_id': 'Lamborghini', 'value': 166318.2}

{'\_id': 'Ferrari', 'value': 94000.0}

{'\_id': 'Maserati', 'value': 69256.0}

{'\_id': 'Lotus', 'value': 68800.0}

{'\_id': 'Tesla', 'value': 61825.0}

{'\_id': 'Aston Martin', 'value': 59266.666666666664}

{'\_id': 'Porsche', 'value': 54283.13725490196}

{'\_id': 'Cupra', 'value': 50900.0}

{'\_id': 'Triumph', 'value': 49910.0}

{'\_id': 'Corvette', 'value': 48500.0}

{'\_id': 'Dodge', 'value': 37376.36363636364}

{'\_id': 'Land Rover', 'value': 25719.272727272728}

{'\_id': 'Autres', 'value': 25038.947368421053}

{'\_id': 'Jaguar', 'value': 24468.5}

{'\_id': 'Ds', 'value': 21506.428571428572}

{'\_id': 'Mercedes', 'value': 21395.771153846155}

{'\_id': 'Lexus', 'value': 20270.0}

{'\_id': 'Audi', 'value': 18331.394}

{'\_id': 'Buick', 'value': 17333.666666666668}

{'\_id': 'Abarth', 'value': 17220.0}

{'\_id': 'Jeep', 'value': 17198.571428571428}

{'\_id': 'Infiniti', 'value': 16874.0}

{'\_id': 'Volvo', 'value': 16619.468085106382}

{'\_id': 'Ford USA', 'value': 16600.0}

{'\_id': 'Chevrolet', 'value': 16556.975}

{'\_id': 'Subaru', 'value': 16480.0}

{'\_id': 'Bmw', 'value': 16004.48561151079}

{'\_id': 'Isuzu', 'value': 15345.0}

{'\_id': 'Bertone', 'value': 15000.0}

{'\_id': 'Honda', 'value': 14662.75}

{'\_id': 'Mazda', 'value': 14203.84090909091}

{'\_id': 'Casalini', 'value': 13990.0}

{'\_id': 'Ford', 'value': 13586.980555555556}

{'\_id': 'Cadillac', 'value': 12960.0}

{'\_id': 'Mini', 'value': 12311.30827067669}

{'\_id': 'Alpine-Renault', 'value': 11500.5}

{'\_id': 'Skoda', 'value': 11484.11320754717}

{'\_id': 'Kia', 'value': 11323.885714285714}

{'\_id': 'Volkswagen', 'value': 11046.911901081916}

{'\_id': 'Nissan', 'value': 10655.17894736842}

{'\_id': 'Hyundai', 'value': 10321.793103448275}

{'\_id': 'Mitsubishi', 'value': 10223.62962962963}

{'\_id': 'Austin', 'value': 9700.0}

{'\_id': 'Chatenet', 'value': 9490.0}

{'\_id': 'Suzuki', 'value': 9160.166666666666}

{'\_id': 'Alfa Romeo', 'value': 9142.679487179486}

{'\_id': 'Dacia', 'value': 8847.557251908396}

{'\_id': 'Toyota', 'value': 8752.787671232876}

{'\_id': 'Ligier', 'value': 8368.153846153846}

{'\_id': 'Peugeot', 'value': 7592.313501144165}

{'\_id': 'Seat', 'value': 7113.888888888889}

{'\_id': 'Pontiac', 'value': 7000.0}

{'\_id': 'Fiat', 'value': 6834.917050691244}

{'\_id': 'Renault', 'value': 6645.701187335092}

{'\_id': 'Mg', 'value': 6225.0}

{'\_id': 'Citroen', 'value': 6214.107921928818}

{'\_id': 'Chrysler', 'value': 6123.75}

{'\_id': 'Smart', 'value': 5966.142857142857}

{'\_id': 'Opel', 'value': 5950.978260869565}

{'\_id': 'Aixam', 'value': 5824.25}

{'\_id': 'Microcar', 'value': 5694.222222222223}

{'\_id': 'Simpa JDM', 'value': 5500.0}

{'\_id': 'Mega', 'value': 5000.0}

{'\_id': 'Lancia', 'value': 4151.666666666667}

{'\_id': 'Saab', 'value': 3996.6}

{'\_id': 'Daewoo', 'value': 3575.0}

{'\_id': 'Ssangyong', 'value': 3540.125}

{'\_id': 'Daihatsu', 'value': 2699.6666666666665}

{'\_id': 'Lada', 'value': 2600.0}

{'\_id': 'Iveco', 'value': 2500.0}

{'\_id': 'Rover', 'value': 885.0}

{'\_id': 'Alpine', 'value': 1.0}

On peut enregistrer ce résultat dans un DataFrame (librairie Pandas) puis sauvegarder dans un fichier csv.

Lamborghini : 166318.0

Ferrari : 94000.0

Maserati : 69256.0

Lotus : 68800.0

Tesla : 61825.0

Aston Martin : 59267.0

Porsche : 54283.0

Cupra : 50900.0

Triumph : 49910.0

Corvette : 48500.0

Dodge : 37376.0

Land Rover : 25719.0

Autres : 25039.0

Jaguar : 24468.0

Ds : 21506.0

Mercedes : 21396.0

Lexus : 20270.0

Audi : 18331.0

Buick : 17334.0

Abarth : 17220.0

Jeep : 17199.0

Infiniti : 16874.0

Volvo : 16619.0

Ford USA : 16600.0

Chevrolet : 16557.0

Subaru : 16480.0

Bmw : 16004.0

Isuzu : 15345.0

Bertone : 15000.0

Honda : 14663.0

Mazda : 14204.0

Casalini : 13990.0

Ford : 13587.0

Cadillac : 12960.0

Mini : 12311.0

Alpine-Renault : 11500.0

Skoda : 11484.0

Kia : 11324.0

Volkswagen : 11047.0

Nissan : 10655.0

Hyundai : 10322.0

Mitsubishi : 10224.0

Austin : 9700.0

Chatenet : 9490.0

Suzuki : 9160.0

Alfa Romeo : 9143.0

Dacia : 8848.0

Toyota : 8753.0

Smart : 5966.0

Opel : 5951.0

Aixam : 5824.0

Microcar : 5694.0

Simpa JDM : 5500.0

Mega : 5000.0

Lancia : 4152.0

Saab : 3997.0

Daewoo : 3575.0

Ssangyong : 3540.0

Daihatsu : 2700.0

Lada : 2600.0

Iveco : 2500.0

Rover : 885.0

Alpine : 1.0

1. Moyenne des prix des voitures en fonction de l’âge de la voiture

collection = "voitures"

map = Code("function () {"

" for(var i=0;i<this.attributes.length;i++){"

" if(this.attributes[i].key == 'regdate') {"

" emit(this.attributes[i].value, this.price[0]);}"

" }"

"}")

reduce = Code("function (key, values) {"

" var total = 0;"

" for (var i = 0; i < values.length; i++) {"

" total += values[i];"

" }"

" return total/values.length;"

"}")

result = db[collection].map\_reduce(map, reduce, "map\_reduce\_price\_per\_year")

affiche(db.map\_reduce\_price\_per\_year.find().sort([("\_id", pymongo. DESCENDING )]))

df = pd.DataFrame(db.map\_reduce\_price\_per\_year.find()).round(0)

df = df.sort\_values(by='\_id', ascending=False)

On obtient la liste suivante :

2020 : 26957.0

2019 : 26180.0

2018 : 23686.0

2017 : 19416.0

2016 : 17228.0

2015 : 15353.0

2014 : 13116.0

2013 : 11377.0

2012 : 10828.0

2011 : 7920.0

2010 : 7577.0

2009 : 6495.0

2008 : 5209.0

2007 : 5153.0

2006 : 3948.0

2005 : 3240.0

2004 : 3019.0

2003 : 2974.0

2002 : 3449.0

2001 : 2350.0

2000 : 2225.0

1999 : 1792.0

1998 : 1931.0

1997 : 1853.0

1996 : 1723.0

1995 : 31851.0

1994 : 6143.0

1993 : 2506.0

1992 : 1036.0

1991 : 7920.0

1990 : 17569.0

1989 : 9622.0

1988 : 12492.0

1987 : 8119.0

1975 : 8083.0

1974 : 34990.0

1973 : 31998.0

1972 : 23230.0

1971 : 25480.0

1970 : 20098.0

1969 : 13862.0

1968 : 21945.0

1967 : 39322.0

1966 : 43077.0

1965 : 27862.0

1964 : 8990.0

1963 : 10.0

1962 : 35015.0

1960 : 22347.0

1. Nombre de vêtements par catégorie

On s’intéresse maintenant à la collection « vetements » avec 208 lignes de données.

Il y a quatre catégories de vêtements :

* Femme
* Maternité
* Homme
* Enfant

On souhaite connaitre la répartition des annonces dans ces catégories.

Voici le code :

collection = "vetements"

map = Code("function () {"

" for(var i=0;i<this.attributes.length;i++){"

" if(this.attributes[i].key == 'clothing\_type') {"

" emit(this.attributes[i].value\_label, this.price[0]);}"

" }"

"}")

reduce = Code("function (key, values) {"

" return values.length;"

"}")

result = db[collection].map\_reduce(map, reduce, "vetement\_categorie\_nb")

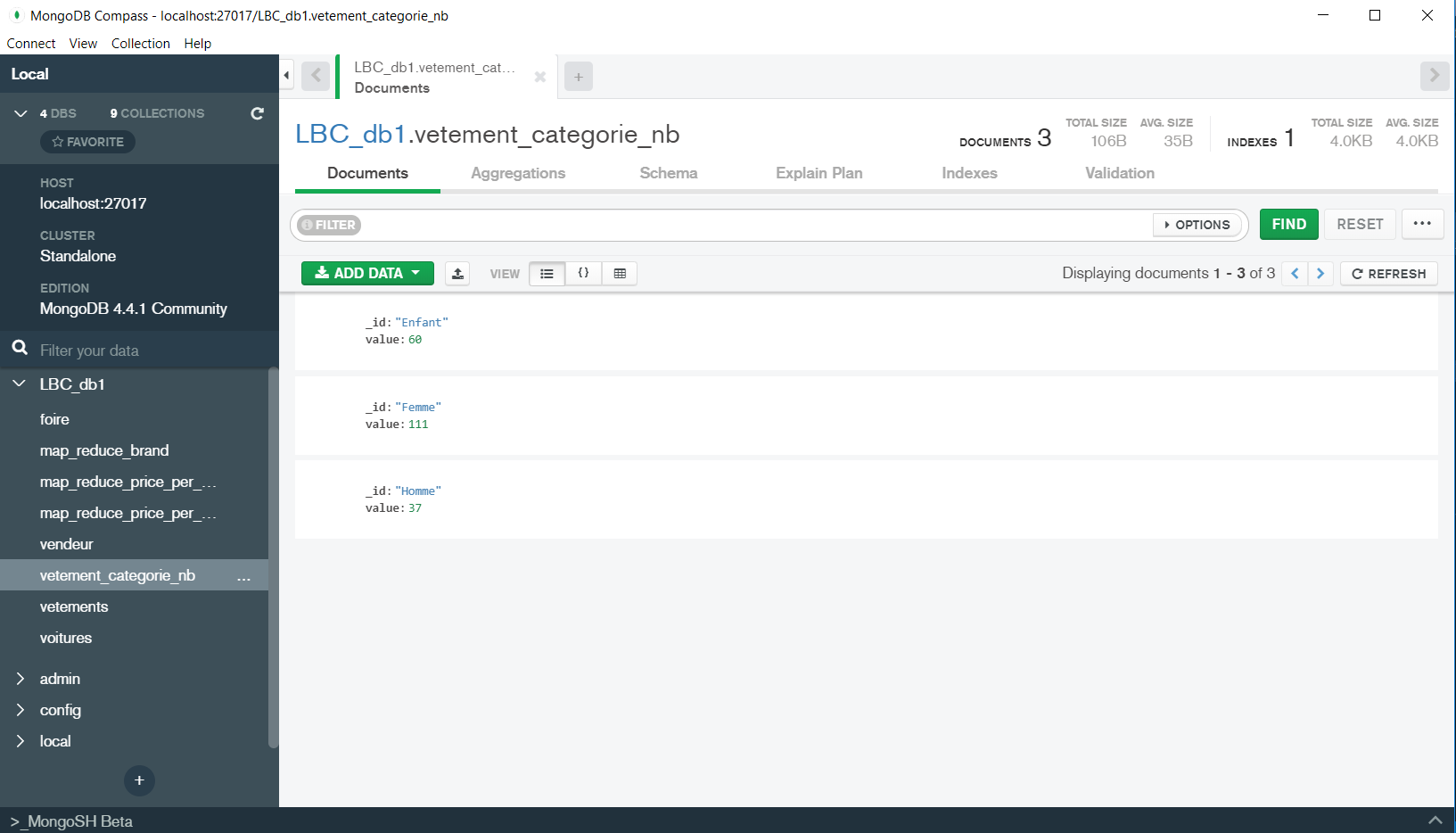
affiche(db.vetement\_categorie\_nb.find())

On peut visualiser le résultat dans l’interface Compass :

Collections : résultats des map-reduce

Map reduce

Visualisation des résultats de la map-reduce



Il n’y a aucune annonce de vêtements de maternité dans notre base de données. La catégorie avec le plus d’offres sont les vêtements de femme.

1. Prix des vêtements par catégorie

On s’intéresse aux prix moyens des 4 catégories de vêtements (voir map-reduce précédente). Voici le code :

collection = "vetements"

map = Code("function () {"

" for(var i=0;i<this.attributes.length;i++){"

" if(this.attributes[i].key == 'clothing\_type') {"

" emit(this.attributes[i].value\_label, this.price[0]);}"

" }"

"}")

reduce = Code("function (key, values) {"

" var total = 0;"

" for (var i = 0; i < values.length; i++) {"

" total += values[i];"

" }"

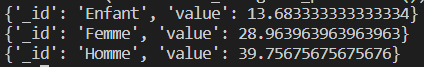
" return total/values.length;"

"}")

result = db[collection].map\_reduce(map, reduce, "vetement\_categorie\_price")

affiche(db.vetement\_categorie\_price.find())

On obtient le résultat suivant :



Les vêtements pour homme sont les plus chers.

En collectant davantage de données (344 données contre 208 précédemment), on obtient ces résultats (via Compass) :



1. Type de vêtements vendus

De la même façon que les catégories de vêtement, on peut regarder le type de vêtements le plus vendus. La requête s’écrit :

map = Code("function () {"

" for(var i=0;i<this.attributes.length;i++){"

" if(this.attributes[i].key == 'clothing\_category') {"

" emit(this.attributes[i].value\_label, this.price[0]);}"

" }"

"}")

reduce = Code("function (key, values) {"

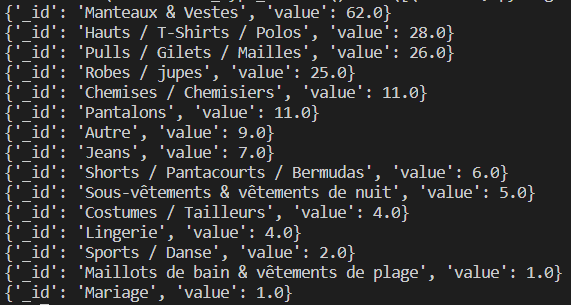
" return values.length;"

"}")

result = db[collection].map\_reduce(map, reduce, "vetement\_type\_nb")

affiche(db.vetement\_type\_nb.find().sort([("value", pymongo.DESCENDING)]))

On obtient le résultat suivant :



Les manteaux et vestes ont le vent en poupe.

Observons maintenant leur prix moyen en modifiant la fonction reduce :

reduce = Code("function (key, values) {"

" var total = 0;"

" for (var i = 0; i < values.length; i++) {"

" total += values[i];"

" }"

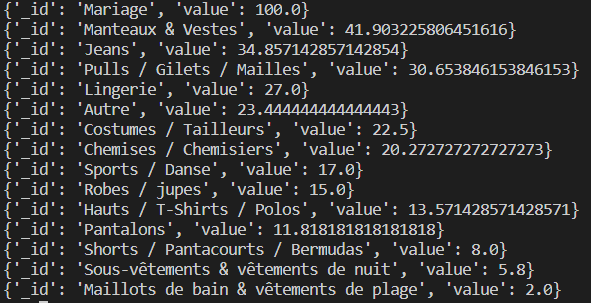
" return total/values.length;"

"}")

result = db[collection].map\_reduce(map, reduce, "vetement\_type\_prix")

affiche(db.vetement\_type\_prix.find().sort([("value", pymongo.DESCENDING)]))

Les résultats sont :



1. **Visualisation des données**
2. Distribution des prix par catégorie

Pour les voitures, nous souhaitons observer la distribution des prix.

import pandas as pd

import seaborn

import matplotlib.pyplot as plt

prix = []

for res in db["voitures"].find():

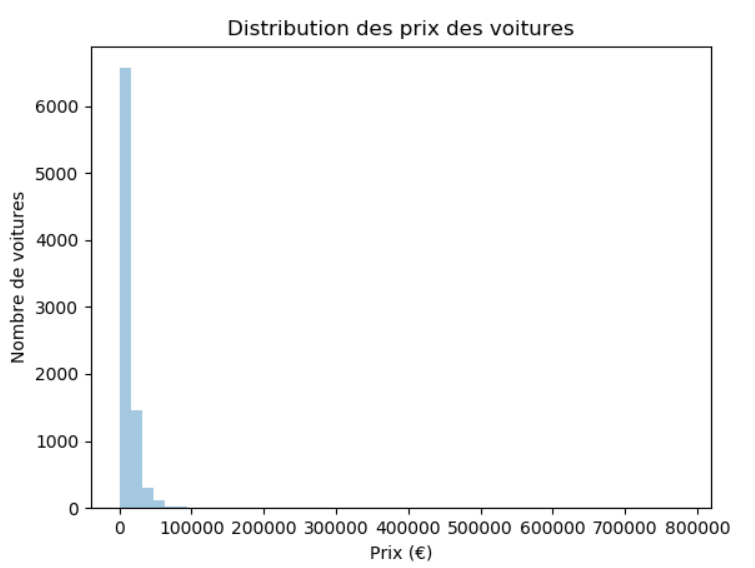
prix.append(res["price"][0])

fig = plt.figure(1)

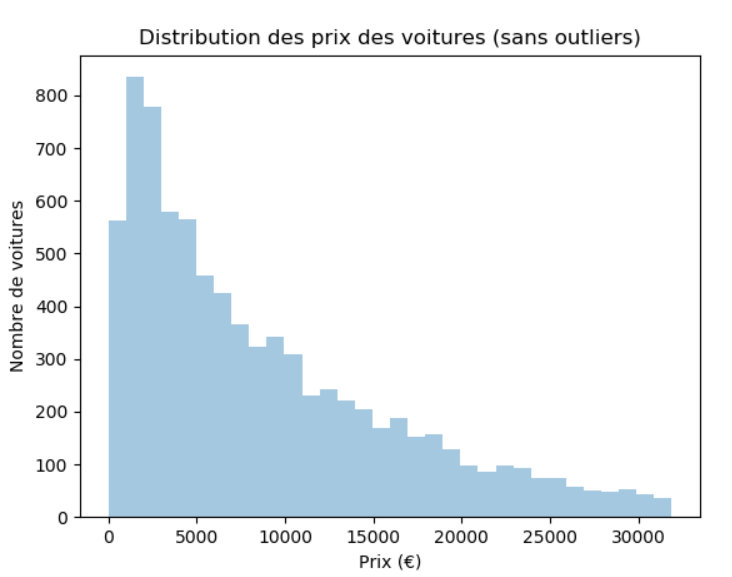
seaborn.distplot(prix, kde=False)

plt.show()

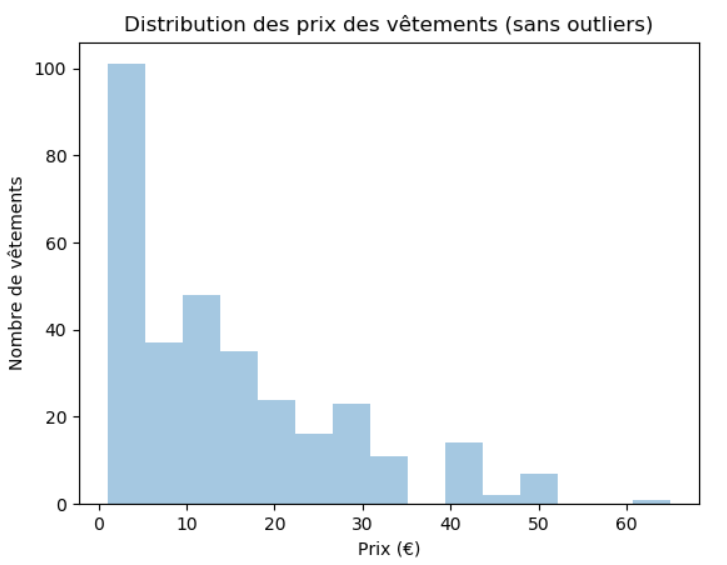
plt.savefig("dist\_prix\_voiture.png")



On remarque que des valeurs aberrantes perturbent la courbe. On va supprimer les outliers. On obtient la distribution suivante :



Pour la distribution des prix des vêtements, on obtient :



Distribution prix par catégorie (countplot)