

Projet Big Data Analytics :  
**Analyse de la Clientèle d'un Concessionnaire Automobile  
pour la Recommandation de Modèles de  
Véhicules**

Année Universitaire : 2023/2024

**Encadré par :**

Mr. Gabriel Mopolo

Mr. Sergio Simonian

Mr. Nicolas Pasquier

**Réalisé par :**

Fahd Ennia

Sara Mehdaoui

Ismail Bouchghel

Oussama Daafane

## Table des matières :

<b>1. Introduction générale .....</b>	4
<b>2. Présentation du projet.....</b>	5
2.1- Justification du Caractère Innovant du Sujet.....	5
2.2- Contexte Métier.....	5
2.3- Enjeux et Risques.....	5
2.3.1- Enjeux.....	5
2.3.2- Risque.....	5
2.3.3- Impact et Importance du Projet.....	5
<b>3. Répartition du travail en membre du groupe.....</b>	6
<b>4. Architecture du data lake .....</b>	6
4.1- Choix de l'architecture du Data Lake.....	6
4.2- Sources de données.....	7
4.3- Implémentation des sources de données.....	7
4.4- Alimentation du Data Lake.....	7
<b>5. Construction du data lake par étape .....</b>	10
5.1- Encodage de l'ensemble des fichier Excel sous format UTF-8 .....	10
5.2- Chargement des fichiers « CO2.csv » et « Catalogue.csv » dans HDFS.....	11
5.3- Chargement des fichiers « Clients_7.csv » et « Client_12.csv » dans MongoDB .....	13
5.4- Chargement du fichier « Immatriculations.csv » dans HBase .....	14
5.5- Chargement du fichier « Marketing.csv » dans MySQL .....	17
5.6- Transfert des données des clients stockées dans MongoDB vers la datalake Hive .....	20
5.7- Transfert des données d'immatriculation stockées dans HBase vers la datalake Hive .....	23
5.8- Transfert des données marketing stockées dans MySQL vers la datalake Hive .....	25
5.9- Transformation des données d'immatriculation de la table Hive externe « table_immatriculations_ext » .....	27
5.10- Exploration des données marketing stockées dans la table Hive « table_marketing » .....	34
5.11- Transformation et nettoyage des données clients de la table Hive « client_7_ext » .....	37
5.11.1- Vérification du respect du format du dictionnaire de donnée .....	37
5.11.2- Calcul de l'estimation du nombre total de valeurs non définies dans la table .....	38
5.11.3- Traitement des valeurs de la colonne « sexe » .....	43
5.11.4- Traitement des valeurs de la colonne « situationFamiliale » .....	46
5.11.5- Traitement des valeurs de la colonne « NbEnfantsAcharge » .....	48
5.11.6- Traitement des valeurs de la colonne « deuxiemevoiture ».....	50
5.11.7- Traitement des valeurs de la colonne « age » .....	51
5.11.8- Traitement des valeurs de la colonne « taux ».....	55
5.11.9- Traitement des valeurs de la colonne « Immatriculation » .....	57
5.11.9.1 – Vérification du format des immatriculations .....	57
5.11.9.2 – Vérification du principe d'unicité des immatriculations .....	58
5.11.10 – Traitement des valeurs NULL de la table « client_7_ext » .....	61
5.11.11 – Nombre d'enregistrement de la table « client_7_ext » après nettoyage et transformation ..	64

5.12- Transformation et nettoyage des données clients de la table Hive « client_12_ext » .....	65
5.12.1- Vérification du respect du format du dictionnaire de donnée .....	65
5.12.2- Calcul de l'estimation du nombre total de valeurs non définies dans la table.....	66
5.12.3- Traitement des valeurs de la colonne « sexe » .....	69
5.12.4- Traitement des valeurs de la colonne « situationFamiliale » .....	72
5.12.5- Traitement des valeurs de la colonne « NbEnfantsAcharge » .....	74
5.12.6- Traitement des valeurs de la colonne « deuxiemevoiture » .....	75
5.12.7- Traitement des valeurs de la colonne « age » .....	76
5.12.8- Traitement des valeurs de la colonne « taux » .....	77
5.12.9- Traitement des valeurs de la colonne « Immatriculation » .....	79
5.12.9.1 – Vérification du format des immatriculations .....	79
5.12.9.2 – Vérification du principe d'unicité des immatriculations .....	79
5.12.10 – Traitement des valeurs NULL de la table « client_12_ext » .....	81
5.12.11 – Nombre d'enregistrement de la table «client_12_ext» après nettoyage et transformation...	85
<b>6. Hadoop Map Reduce</b> .....	86
<b>7. Finalisation de la datalake et chargement du fichier résultat MapReduce</b> .....	95
7.1 Chargement du résultat du traitement MapReduce dans la datalake Hive .....	95
7.2 Exploration des données du catalogue de la table Hive « resultat_catalogue_co2 » .....	98
7.3 Construction du model d'analyse de la datalake .....	101
<b>8. Data Mining, Machine Learning et Deep Learning</b> .....	107
8.1 Analyse Exploratoire des données du catalogue du concessionnaire .....	107
8.2 Identification des catégories de véhicules en utilisant le Clustering .....	127
8.3 Analyse Exploratoire et application du Clustering aux catégories du modèle d'analyse .....	136
8.4 Création de modèles de connaissances et évaluation des résultats .....	165
8.5 Application du modèle de prédiction aux données Marketing .....	194
8.6 Construction de la Base de données résultat .....	199
8.6.1 Insertion des données de prédictions dans la base de données résultat. ....	199
8.6.2 Insertion des visualisations de l'analyse exploratoire dans la base de données résultat .....	200
<b>9. Création de Dashboard Power BI</b> .....	201
<b>10. Conclusion générale</b> .....	207
<b>11. Références et Bibliographie</b> .....	209
<b>12. Annexes</b> .....	210
12.1 Vidéo de présentation de votre projet.....	210
12.2 Dossier contenant les scripts et programmes de construction du lac de données .....	210
12.3 Dossier contenant les scripts et programmes Hadoop Map Reduce .....	210
12.4 Dossier contenant les scripts et programmes de visualisations et d'analyse de données .....	210

## 1. Introduction générale

Ce projet s'inscrit dans le cadre des cours de **Big Data, Data Visualisation et Data Analytics & Machine Learning**. Il vise à mettre en pratique les concepts acquis dans ces cours en analysant les données d'un concessionnaire automobile afin de proposer des recommandations de modèles de véhicules aux clients.

Les principaux objectifs de ce projet consistent à gérer efficacement les données en utilisant l'architecture BDA/DL, à mettre en place la visualisation des données pour une compréhension claire, à analyser les données du domaine d'application et à les catégoriser en fonction des critères pertinents, à développer un outil permettant aux vendeurs d'évaluer rapidement le type de véhicule le plus susceptible d'intéresser un client, ainsi qu'à concevoir un système capable d'envoyer une documentation précise sur le véhicule le plus adéquat à des clients spécifiquement ciblés par le service marketing.

Au cours du projet, nous avons accompli plusieurs missions essentielles pour atteindre nos objectifs. Nous avons débuté par la conception de l'architecture nécessaire à la gestion efficace des données. Ensuite, nous avons procédé au chargement des données vers les bases de données source, suivi par l'établissement d'un traitement de MapReduce avec Spark pour traiter les données concessionnaires à grande échelle. Nous avons ensuite extrait les données vers le DataLake afin de les rendre accessibles pour l'analyse ultérieure. Une fois les données stockées dans le DataLake, nous avons entrepris leur nettoyage et leur transformation pour garantir leur qualité et leur cohérence. Par la suite, nous avons procédé à l'analyse approfondie des données et à leur visualisation à partir du DataLake, offrant ainsi des insights précieux pour la prise de décision. Enfin, nous avons utilisé des techniques de Machine Learning pour faire des prédictions pertinentes, contribuant ainsi à la recommandation de modèles de véhicules adaptés aux clients.

Le plan du projet :

- **Architecture de Data Lake :** La mise en place de l'architecture, implémentation des sources de données et l'alimentation de Data Lake.
- **Construction de Datalake par étape :** Charger les fichiers CSV vers leurs bases de données, et les transfère vers le Datalake Hive, puis les transformer et les nettoyer.
- **Hadoop Map Reduce :** Le nettoyage et la transformation des données du fichier "CO2.csv" et "Catalogue.csv"
- **Finalisation de la Datalake et chargement du fichier résultat MapReduce :** Chargement des données résultats et traitement de datalake.
- **Analyse et Visualisation des Données par les Techniques de Data Mining, Machine Learning et Deep Learning:** Extraire des modèles de connaissances significatifs à partir des données disponibles et établire les Visualisation et l'Analyse.

Ce projet permettra au concessionnaire automobile de mieux cibler ses clients et de leur proposer des modèles de véhicules plus susceptibles de les intéresser. Cela devrait se traduire par une augmentation des ventes et une amélioration de la satisfaction des clients.

## **2. Présentation du Projet :**

Le projet vise à réaliser une analyse approfondie de la clientèle d'un concessionnaire automobile en vue de recommander des modèles de véhicules adaptés à leurs besoins et préférences. Pour ce faire, nous utiliserons des techniques avancées d'analyse de données, notamment le Big Data Analytics, afin d'exploiter efficacement les vastes ensembles de données disponibles.

### **2.1- Justification du Caractère Innovant du Sujet :**

L'analyse de la clientèle d'un concessionnaire automobile représente une approche novatrice dans le domaine, offrant une opportunité unique d'améliorer l'expérience client et d'optimiser les performances commerciales. En exploitant les données disponibles, nous pouvons déceler des tendances, des comportements d'achat et des préférences spécifiques des clients, permettant ainsi des recommandations personnalisées et une satisfaction client accrue.

### **2.2- Contexte Métier :**

Le contexte métier de notre projet est celui d'un concessionnaire automobile cherchant à optimiser ses activités commerciales grâce à l'analyse approfondie de ses données clients et produits. En s'appuyant sur un large ensemble de données telles que le catalogue de véhicules, l'historique des achats des clients et les données d'immatriculation, notre objectif est d'aider le concessionnaire à mieux comprendre les besoins et les préférences de sa clientèle, et ainsi à adapter son offre en conséquence.

### **2.3- Enjeux et Risques :**

#### **2.3.1- Enjeux :**

Améliorer la satisfaction client en recommandant des modèles de véhicules pertinents et adaptés à leurs besoins. Optimiser les ventes en proposant des produits qui correspondent mieux aux préférences des clients, augmentant ainsi les chances de conversion.

#### **2.3.2- Risques :**

Erreurs dans les données, telles que des valeurs manquantes ou des incohérences, pouvant fausser les résultats de l'analyse. La complexité de la segmentation de la clientèle peut rendre difficile la catégorisation précise des clients en groupes homogènes. La dépendance aux modèles de prédiction : les recommandations reposent sur des algorithmes de prédiction, ce qui implique une certaine incertitude quant à leur précision et à leur fiabilité.

#### **2.3.3- Impact et Importance du Projet :**

Ce projet revêt une importance capitale pour le concessionnaire automobile, car il offre la possibilité d'optimiser ses opérations commerciales et d'accroître la satisfaction de sa clientèle. En identifiant les modèles de véhicules les plus pertinents pour chaque segment de clientèle, le concessionnaire peut non seulement améliorer ses ventes, mais aussi renforcer sa réputation en tant que fournisseur de services personnalisés et de qualité.

### 3. Répartition du travail en membre du groupe

Bouchghel Ismail , Sara Mehdaoui , Oussama Daafane :

- Rédaction du rapport final du projet de fin d'année.

- Conception de l'architecture du Big Data Analytics/Data Lake.

- Construction et Implémentation des Data sources Hbase, Mongo Db, Mysql, chargement des fichiers sources en HDFS.

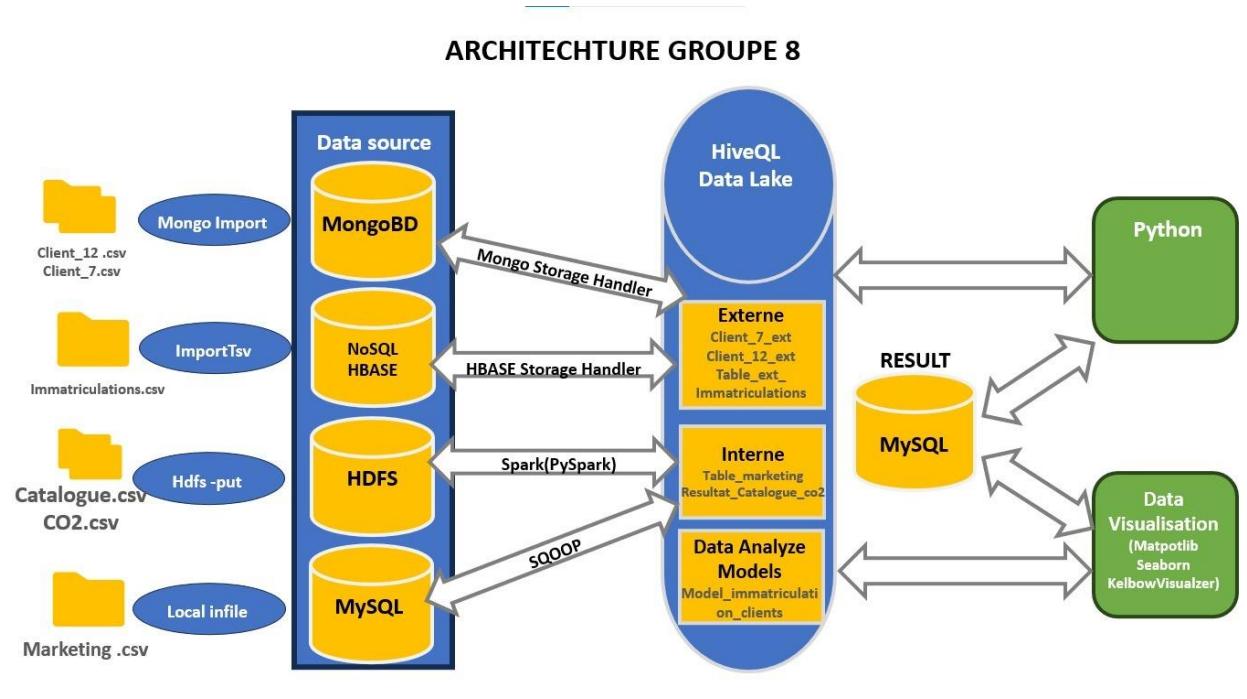
- Ecriture des scripts pour la phase de construction de la DataLake.

Fahd Ennia : Responsable de la mise en place de la DataLake, du traitement MapReduce, de l'analyse et de la visualisation des données en Data Mining.

Sara Mehdaoui : Utilisation de l'outil BI et Création de Dashboards pour les Analyses des Ventes de Voitures.

### 4. Architecture du Data Lake:

Dans le cadre de ce projet visant à aider un concessionnaire automobile à mieux cibler les véhicules susceptibles d'intéresser ses clients, la mise en place d'une architecture de Data Lake est cruciale pour la consolidation, le traitement et l'analyse des données. Suivant présente notre approche, en mettant en évidence les choix technologiques effectués et les stratégies d'alimentation des données.



#### 4.1- Choix de l'architecture du Data Lake :

Nous devons choisir entre deux options :

1. DataLake HIVEQL :

Utilisation du moteur SQL HIVE d'Hadoop comme frontal pour le DataLake. Toutes les données doivent être accessibles depuis HIVE pour construire les modèles d'analyse.

2. DataLake SQL :

Utilisation d'un moteur SQL du marché (Oracle, SQL Server, MySQL, PostgreSQL, etc.) comme frontal pour le DataLake.

Pour ce projet, nous optons pour la DataLake HIVE qui offre une intégration native avec l'écosystème Hadoop, ce qui est essentiel pour le traitement de volumes de données importants et complexes.

#### 4.2- Sources de données :

- HDFS (Hadoop Distributed File System) : Stockage des fichiers de données notamment le fichier « CO2.csv » et « Catalogue.csv » et puisque on a un traitement MapReduce a appliqué ça sera intéressant de ce bénéficier de la compatibilité native de Spark avec HDFS, ce qui simplifie l'intégration, le traitement et le chargement des données à partir de ce système de fichiers distribué et sa capacité à gérer de gros volumes de données de manière distribuée.
- MongoDB : Utilisé comme source de données pour les données Clients . L'alimentation se fera via un extracteur de données approprié.
- HBase : Stockage des informations sur les données relatives aux immatriculations.
- MySQL : Stockage des données marketing.

#### 4.3- Implémentation des sources de données :

Les fichiers CSV du catalogue et de CO2 ont été chargés dans HDFS à l'aide de la commande hdfs -put.

Les fichiers client ont été importés dans MongoDB à l'aide de mongoImport.

Les données d'immatriculation ont été chargées dans HBase à l'aide de ImportTsv.

Les données marketing ont été insérées dans MySQL à partir de fichiers locaux.

#### 4.4- Alimentation du Data Lake :

Nous utiliserons principalement des access drivers et des extracteurs de données pour rendre les données disponibles dans la DataLake.

- Spark : Charger les fichiers résultants du traitement MapReduce, à savoir « Catalogue.csv» et « CO2.csv », dans HDFS, puis les importer directement dans le datalake Hive.

- Mongo Storage Handler : Cette solution permet un accès direct aux données MongoDB via des tables externes dans Hive, facilitant ainsi leur intégration dans la DataLake sans nécessiter de déplacement physique des données clients depuis MongoDB
- HBase Storage Handler : Utilisé pour accéder et charger les données relatives aux immatriculations depuis HBase dans la DataLake. Cette approche offre une intégration transparente des données d'immatriculation dans la DataLake via des tables externes.
- Sqoop : Employé pour extraire les données marketing depuis MySQL vers des tables internes dans Hive, garantissant ainsi leur cohérence avec les autres données présentes dans l'environnement Hive.

Dans un premier temps, nous avons utilisé Spark pour charger et réaliser le Job MapReduce sur les fichiers CSV du catalogue et des émissions de CO<sub>2</sub> vers le système de fichiers HDFS. Cette stratégie a assuré un traitement parallèle efficace des données, ce qui est crucial pour la gestion de volumes importants de données.

Pour intégrer les données clients stockées dans MongoDB, nous avons tiré parti du « Mongo Storage Handler » de Hive. Cette solution a permis un accès direct aux données MongoDB via des tables externes dans Hive, simplifiant ainsi leur intégration dans la DataLake sans nécessiter de déplacement physique des données.

De manière similaire, nous avons utilisé le « HBase Storage Handler » pour accéder aux données d'immatriculation stockées dans HBase. Cette approche a permis une intégration transparente des données d'immatriculation dans la DataLake via des tables externes, offrant ainsi une cohérence et une accessibilité accrues aux données.

Enfin, pour importer les données marketing stockées dans MySQL, nous avons employé « Sqoop » pour extraire les données vers des tables internes dans Hive. Cette stratégie a facilité l'utilisation des données marketing dans la DataLake, tout en garantissant leur cohérence avec les autres données présentes dans l'environnement Hive.

Globalement, cette approche a permis une intégration fluide et efficiente des données provenant de diverses sources dans la DataLake Hive, créant ainsi une base solide pour les analyses et la prise de décisions du concessionnaire automobile.

- Tables internes et externes :

Une fois les données chargées dans les différents types de bases de données, nous les transformerons en tables internes et externes dans notre Data Lake. Les tables internes seront utilisées pour le stockage temporaire et le traitement des données, tandis que les tables externes permettront l'accès aux données sans déplacement physique.

- Modèle d'analyse et stockage des résultats de prédition :

Pour répondre aux besoins du concessionnaire automobile, Un modèle d'analyse efficace pour rapidement aider à évaluer le type de véhicule susceptible d'intéresser les clients est nécessaire. Le modèle va nous permettre de construire une vue complète des préférences et des comportements d'achat des clients (reliant les données des clients avec les informations d'immatriculation des

véhicules, éliminant les doublons et en garantissant que chaque client est associé à des véhicules pertinents). Cette modélisation des données garantit ainsi la cohérence et l'intégrité des données, tout en optimisant l'utilisation des ressources pour une gestion efficace des données à grande échelle.

Une approche courante serait de stocker les résultats de prédiction dans une base de données appropriée MySql, ce qui permettrait aux vendeurs d'accéder rapidement aux recommandations de véhicules pour les clients et au service marketing d'envoyer des documentations précises.

- Python et les bibliothèques de visualisation:

L'analyse des données implique plusieurs étapes cruciales, chacune soutenue par l'utilisation de Python et des bibliothèques de visualisation telles que Matplotlib et Seaborn. Tout d'abord, l'analyse exploratoire des données est facilitée par les fonctionnalités de manipulation de données et les capacités de visualisation fournies par ces bibliothèques, permettant ainsi l'identification des tendances et des valeurs aberrantes. Ensuite, l'identification des catégories de véhicules à partir des informations du Catalogue est rendue possible par l'application de techniques de clustering et de classification en utilisant des bibliothèques de machine learning. La fusion des données clients et immatriculations est également simplifiée offrant ainsi une vue complète des informations sur les clients et les véhicules qu'ils ont achetés. De plus, la création de modèles de classification supervisée pour prédire les catégories de véhicules est réalisée efficacement grâce aux fonctionnalités de machine learning de Python, avec une évaluation des performances visuellement représentée à l'aide de Matplotlib et Seaborn. Enfin, l'application des modèles de prédiction aux données marketing est une étape fluide, permettant ainsi de fournir des recommandations personnalisées aux clients potentiels.

En somme, l'utilisation cohérente de Python et des bibliothèques de visualisation dans l'architecture du projet garantit une analyse des données robuste, des modèles prédictifs performants et une prise de décision éclairée pour les besoins du projet.

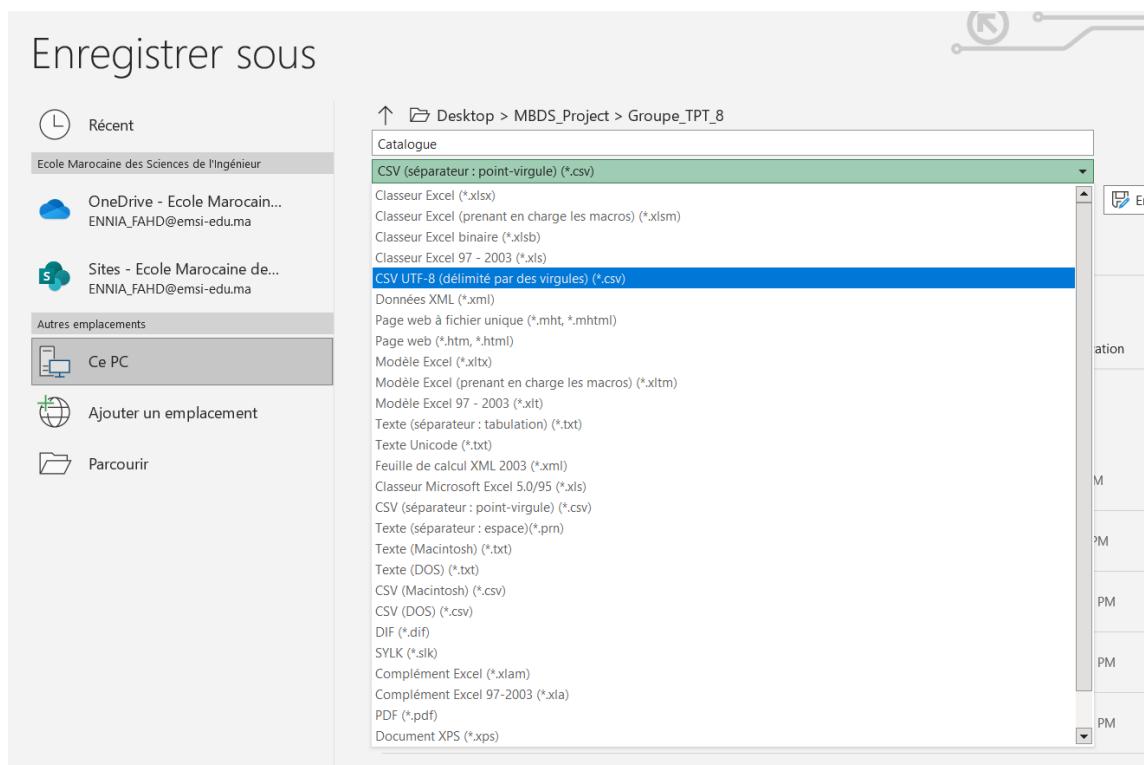
En suivant cette architecture, nous pouvons fournir au concessionnaire automobile les moyens nécessaires pour qu'un vendeur puisse évaluer rapidement le type de véhicule susceptible d'intéresser les clients et pour que le service marketing puisse envoyer des documentations stratégiques précises.

## 5. Construction du data lake par étape:

Une fois l'architecture établit, elle faut l'implémenter en suivant un ensemble d'étapes. La première est de charger l'ensemble des fichiers csv dans les bases de données correspondantes et de migrer ces données vers une data lake Hive.

### 5.1- Encodage de l'ensemble des fichier Excel sous format UTF-8

Avant d'importer les fichiers dans la base de données source, il est recommandé de les convertir au format UTF-8 afin d'éviter tout problème lié aux caractères spéciaux lors du chargement. J'ai utilisé deux approches pour cela : la première consiste à effectuer la conversion via Excel pour les fichiers de taille moyenne tels que "CO2.csv", "Client\_7.csv", "Client\_12.csv" et "Catalogue.csv", tandis que la deuxième approche implique l'encodage via l'application Notepad++ pour les fichiers Excel de grande taille comme le fichier "Immatriculation.csv".



ID	immatriculation	marque	nom	puissance	longueur	nbPlaces	nbPortes	couleur	occasion	prix
1	immatriculation,marqu									
2	3176 TS 67,Renault,L	Renault	S80 T6	272	très longue	5	5	blanc	false	50500
3	3721 QS 49,Volvo,S80	Volvo	S80 T6	272	très longue	5	5	noir	false	50500
4	9099 UV 26,Volkswag	Volkswagen	Touran 2.0 FSI	150	longue	7	5	rouge	false	27340
5	3563 IA 55,Peugeot,10	Peugeot	308	1.6	très longue	5	5	gris	true	16029
6	6963 AX 34,Audi,A2 1.	Audi	A2 1.4	1.4	très longue	5	5	blanc	false	27300
7	5592 HQ 89,Skoda,Supé	Skoda	Octavia 1.6	1.6	très longue	5	5	bleu	false	31790
8	674 CE 26,Renault,Meg	Renault	Mégane 1.6	1.6	très longue	5	5	rouge	false	22350
9	1756 FR 31,Mercedes,J	Mercedes-Benz	C-Klasse 2.0	2.0	très longue	5	5	gris	false	18130
10	6705 GX 50,BMW,120i,1	BMW	120i	2.0	très longue	5	5	bleu	false	60
11	4487 DR 75,Saab,9.3 1	Saab	9-3	1.8	très longue	5	5	rouge	false	27020
12	7080 NW 34,Jaguar,X-Type 2.5	Jaguar	X-Type 2.5	2.5	V6,197, longue	5	5	blanc	true	25970
13	9626 HF 36,Audi,A2 1.4,75	Audi	A2 1.4	1.4	courte	5	5	rouge	false	18310
14	2401 PA 98,Volvo,S80 T6	Volvo	S80 T6	272	tres longue	5	5	bleu	true	35350
15	826 YF 89,Renault,Laguna 2.0T	Renault	Laguna 2.0T	2.0T	170, longue	5	5	rouge	false	27300
16	8216 GR 23,Skoda,Superb	Skoda	Superb 2.0	2.0	V6,193, tres longue	5	5	bleu	false	31790
17	8076 YM 23,Jaguar,X-Type 2.5	Jaguar	X-Type 2.5	2.5	V6,197, longue	5	5	noir	false	37100
18	9277 JN 49,BMW,M5,507,tre	BMW	M5	507	tres longue	5	5	rouge	true	66360
19	4231 HC 31,Audi,A2 1.4,75	Audi	A2 1.4	1.4	courte	5	5	rouge	false	18310
20	2319 IQ 28,Ford,Mondeo 1.8	Ford	Mondeo 1.8	1.8	125,très longue	5	5	gris	false	23900
21	148 RS 75,BMW,M5,507,tre	BMW	M5	507	tres longue	5	5	blanc	true	66360
22	6786 JV 36,Skoda,Superb 2.8	Skoda	Superb 2.8	2.8	V6,193, tres longue	5	5	gris	false	31790
23	8049 KN 17,Renault,Mégane 1.6V	Renault	Mégane 1.6V	1.6V	135,moyenne	5	5	blanc	false	22350
24	9610 BR 52,Volkswagen,New	Volkswagen	Beetle 1.8	1.8	110,moyenne	5	5	blanc	true	18641
25	8745 KJ 12,Volkswagen,Polo 1.2	Volkswagen	Polo 1.2	1.2	6V,55,courte	5	3	gris	false	12200
26	5605 YN 37,BMW,M5,507,tre	BMW	M5	507	tres longue	5	5	rouge	true	66360
27	7341 QB 17,BMW,M5,507,tre	BMW	M5	507	tres longue	5	5	bleu	false	94800
28	9925 TY 41,Audi,A2 1.4,75	Audi	A2 1.4	1.4	courte	5	5	rouge	false	18310
29	6238 TQ 16,Audi,A2 1.4,75	Audi	A2 1.4	1.4	courte	5	5	noir	false	18310
30	4395 AS 40,BMW,M5,507,tre	BMW	M5	507	tres longue	5	5	noir	true	66360
31	1295 WZ 85,Peugeot,1007	Peugeot	1007	1.4,75	courte	5	5	gris	false	13750
32	3228 PI 22,Audi,A2 1.4,75	Audi	A2 1.4	1.4	courte	5	5	gris	false	18310
33	3367 NG 10,Volvo,S80 T6	Volvo	S80 T6	272	tres longue	5	5	gris	false	50500
34	5784 HC 14,Audi,A2 1.4,75	Audi	A2 1.4	1.4	courte	5	5	bleu	false	18310
35	6685 TE 75,Volkswagen,Golf 2.0	Volkswagen	Golf 2.0	FSI,150,moyenne	5,5, rouge	5	5	rouge	false	22900
36	6461 RY 26,Renault,Laguna 2.0T	Renault	Laguna 2.0T	2.0T	170, longue	5	5	noir	false	27300
37	1498 MN 80,Fiat,Croma 2.2	Fiat	Croma 2.2	147, longue	5,5, blanc	5	5	blanc	false	24780
38	4400 IT 29,BMW,M5,507,tre	BMW	M5	507	tres longue	5	5	gris	false	94800
39	9066 UX 89,Fiat,Croma 2.2	Fiat	Croma 2.2	147, longue	5,5, rouge	5	5	rouge	false	24780

## 5.2- Chargement des fichiers « CO2.csv » et « Catalogue.csv » dans HDFS

Avant de transférer les fichiers dans le système de fichiers distribué HDFS, commençons par une présentation des données du fichier Excel « Catalogue.csv » :

A	B	C	D	E	F	G	H	I	J	K
marque	nom	puissance	longueur	nbPlaces	nbPortes	couleur	occasion	prix		
Volvo	S80 T6		272 très longue	5	5 blanc	false		50500		
Volvo	S80 T6		272 très longue	5	5 noir	false		50500		
Volvo	S80 T6		272 très longue	5	5 rouge	false		50500		
Volvo	S80 T6		272 très longue	5	5 gris	true		35350		
Volvo	S80 T6		272 très longue	5	5 bleu	true		35350		
Volvo	S80 T6		272 très longue	5	5 gris	false		50500		
Volvo	S80 T6		272 très longue	5	5 bleu	false		50500		
Volvo	S80 T6		272 très longue	5	5 rouge	true		35350		
Volvo	S80 T6		272 très longue	5	5 blanc	true		35350		
Volvo	S80 T6		272 très longue	5	5 noir	true		35350		
Volkswagen	Touran 2.0 FSI		150 longue	7	5 rouge	false		27340		
Volkswagen	Touran 2.0 FSI		150 longue	7	5 gris	true		19138		

- Marque** : Le nom de la marque qui fabrique le véhicule.
- Nom** : Le nom spécifique du modèle de véhicule.
- Puissance** : La capacité du moteur, exprimée en chevaux.
- Longueur** : La classification de la longueur du véhicule.
- NbPlaces** : Le nombre total de places assises dans le véhicule.
- NbPortes** : Le nombre de portes d'accès au véhicule.
- Couleur** : La couleur de la carrosserie du véhicule.
- Occasion** : Un indicateur booléen pour spécifier si le véhicule est d'occasion.
- Prix** : Le prix de vente du véhicule exprimé en euros.

Examinons également les données du fichier "CO2.csv" :

,Marque / Modèle,Bonus / Malus,Rejets CO2 g/km,Cout energie
2,AUDI E-TRON SPORTBACK 55 (408ch) quattro,-6 000€ 1,0,319 €
3,AUDI E-TRON SPORTBACK 50 (313ch) quattro,-6 000€ 1,0,356 €
4,AUDI E-TRON 55 (408ch) quattro,-6 000€ 1,0,357 €
5,AUDI E-TRON 50 (313ch) quattro,-6 000€ 1,0,356 €
6,BMW i3 120 Ah,-6 000€ 1,0,204 €
7,BMW i3s 120 Ah,-6 000€ 1,0,204 €
8,CITROËN BERLINGO,-6 000€ 1,0,203 €
9,CITROËN C-ZERO,-6 000€ 1,0,491 €
10,DS DS3 CROSSBACK E-Tense (136ch),-6 000€ 1,0,251 €
11,HYUNDAI KONA electric 64 kWh,-6 000€ 1,0,205 €
12,HYUNDAI KONA electric 39 kWh,-6 000€ 1,0,205 €
13,JAGUAR I-PACE EV400 (400 CEE) - Automatique - 4 roues motrices,-6 000€ 1,0,271 €
14,"KIA e-NIRO Moteur Aélectrique synchrone A aimant permanent, 150kW (204ch)",-6 000€ 1,0,212 €
15,"KIA e-NIRO Moteur Aélectrique synchrone A aimant permanent, 100kW (136ch)",-6 000€ 1,0,203 €
16,KIA SOUL Moteur Aélectrique 150 kW (204ch),-6 000€ 1,0,214 €
17,KIA SOUL Moteur Aélectrique 100kW (136ch),-6 000€ 1,0,214 €
18,MERCEDES EQC 400 4MATIC,-6 000€ 1,0,291 €
19,MERCEDES VITO Tourer long eVITO (85kW) Traction 4x2 PTAC 3200kg,-6 000€ 1,0,411 €
20,MERCEDES VITO Tourer extra-long eVITO (85kW) Traction 4x2 PTAC 3200kg,-6 000€ 1,0,411 €
21,MINI MINI Cooper SE Hatch 3P (F56),-6 000€ 1,0,199 €
23,NISSAN Nouvelle NISSAN LEAF,-6 000€ 1,0,155 €
24,NISSAN Nouvelle NISSAN LEAF,-6 000€ 1,0,177 €
25,PEUGEOT 208 E-Tense (136 ch),-6 000€ 1,0,221 €
26,PEUGEOT 108 électrique (67ch),-6 000€ 1,0,241 €
27,PEUGEOT PARTNER,-6 000€ 1,0,203 €
28,RENAULT ZOE (43kW),-6 000€ 1,0,206 €
29,RENAULT ZOE (51kW),-6 000€ 1,0,206 €
30,RENAULT ZOE (53kW),-6 000€ 1,0,206 €

- Marque / Modèle :** Cette colonne combine le nom de la marque et le modèle du véhicule, fournissant ainsi une identification unique pour chaque entrée.
- Bonus/Malus :** Cette colonne indique si le véhicule est éligible à un bonus ou un malus écologique en fonction de ses émissions de CO2 ou d'autres critères environnementaux.
- Rejets CO2 g/km :** Cette colonne spécifie les émissions de dioxyde de carbone (CO2) du véhicule, mesurées en grammes par kilomètre parcouru.
- Coût Énergie :** Cette colonne représente le coût estimé en Euro de l'énergie nécessaire pour faire fonctionner le véhicule sur une certaine distance ou une certaine durée

Maintenant, nous allons créer un répertoire HDFS nommé "MBDS\_Projet" dans lequel nous stockerons les fichiers Excel "Catalogue.csv" et "CO2.csv" :

```
[vagrant@oracle-21c-vagrant ~]$ hdfs dfs -ls
Found 20 items
-rw-r--r-- 1 vagrant supergroup 46750706 2024-01-19 14:36 anonymisedData.zip
-rw-r--r-- 1 vagrant supergroup 8200 2024-01-19 14:39 assessments.csv
-rw-r--r-- 1 vagrant supergroup 9301 2024-01-16 09:41 common_words_en_subset.txt
-rw-r--r-- 1 vagrant supergroup 526 2024-01-19 14:40 courses.csv
-rw-r--r-- 1 vagrant supergroup 78 2024-01-17 09:13 graph_input.txt
-rw-r--r-- 1 vagrant supergroup 1669 2024-01-16 09:03 poeme.txt
drwxr-xr-x - vagrant supergroup 0 2024-01-18 12:20 res-words-filtered
drwxr-xr-x - vagrant supergroup 0 2024-01-18 12:20 res-words-unfiltered
drwxr-xr-x - vagrant supergroup 0 2024-01-16 10:14 resulsales
drwxr-xr-x - vagrant supergroup 0 2024-01-16 11:55 result
drwxr-xr-x - vagrant supergroup 0 2024-01-17 10:09 results
drwxr-xr-x - vagrant supergroup 0 2024-01-16 09:25 resultsAnagrame
drwxr-xr-x - vagrant supergroup 0 2024-01-16 09:38 resultsAnagrame2
drwxr-xr-x - vagrant supergroup 0 2024-01-16 09:41 resultsAnagrame3
-rw-r--r-- 1 vagrant supergroup 1237262 2024-01-16 10:03 sales_world_10k.csv
-rw-r--r-- 1 vagrant supergroup 5690310 2024-01-19 16:50 studentAssessment.csv
-rw-r--r-- 1 vagrant supergroup 3461652 2024-01-19 14:59 studentInfo.csv
-rw-r--r-- 1 vagrant supergroup 1109984 2024-01-19 14:41 studentRegistration.csv
-rw-r--r-- 1 vagrant supergroup 453836331 2024-01-19 14:41 studentVle.csv
-rw-r--r-- 1 vagrant supergroup 260126 2024-01-19 14:43 vle.csv
[vagrant@oracle-21c-vagrant ~]$ hdfs dfs -mkdir /MBDS_Projet
```

```
[vagrant@oracle-21c-vagrant ~]$ hdfs dfs -ls ./MBDS_Projet/
[vagrant@oracle-21c-vagrant ~]$ hadoop fs -put /vagrant/Groupe_TPT_8/C02.csv MBDS_Projet
[vagrant@oracle-21c-vagrant ~]$ hadoop fs -put /vagrant/Groupe_TPT_8/Catalogue.csv MBDS_Projet
[vagrant@oracle-21c-vagrant ~]$ hdfs dfs -ls MBDS_Projet
Found 2 items
-rw-r--r-- 1 vagrant supergroup      38916 2024-04-24 12:06 MBDS_Projet/C02.csv
-rw-r--r-- 1 vagrant supergroup      14114 2024-04-24 12:07 MBDS_Projet/Catalogue.csv
[vagrant@oracle-21c-vagrant ~]$
```

### 5.3- Chargement des fichiers « Clients\_7.csv » et « Client\_12.csv » dans MongoDB.

Tout d'abord, voici une présentation des données clients :

A	B	C	D	E	F	G	H
age	sexe	taux	situationFamiliale	nbEnfantsAcharge	2eme voiture	immatriculation	
24	F	427	En Couple		1 false	1674 SA 38	
73	M	1181	Marié(e)		3 true	3188 JC 82	
52	F	542	En Couple		2 false	2003 VX 12	
59	M	454	En Couple		4 false	5738 NG 25	
63	M	440	En Couple		1 false	4343 RQ 30	
79	F	238	En Couple		3 false	2540 IQ 68	
44	F	162	En Couple		0 true	9353 ZH 49	
81	M	883	En Couple		3 false	2179 XT 94	
73	M	593	Célibataire		0 false	8579 ZW 93	
64	F	546	En Couple		1 false	1437 AT 14	
58	M	519	Célibataire		0 false	6347 SJ 79	
22	M	598	Célibataire		0 false	8950 YE 98	
33	M	1039	En Couple		2 false	5665 QF 27	
36	M	711	Célibataire		0 false	1090 XZ 69	

- Age** : L'âge du client.
- Sexe** : Le genre de la personne.
- Taux** : Capacité d'endettement du client en euros (correspondant à 30% de son salaire).
- Situation Familiale** : La situation familiale du client.
- NbEnfantsAcharge** : Le nombre d'enfants à charge.
- 2eme voiture** : Indique si le client possède déjà un véhicule principal.
- Immatriculation** : Le numéro d'immatriculation unique du véhicule.

Maintenant, on importe les fichiers Excel utilisant MongoImport qui permet d'extraire les données depuis la source (fichier Client\_7 et Client\_12) jusqu'à notre base de données MongoDB :

```
[vagrant@oracle-21c-vagrant ~]$ mongoimport -d Clients -c Clients_7 --type csv --file /vagrant/Groupe_TPT_8/Clients_7.csv --headerline
2024-03-06T21:01:11.015+0100    connected to: localhost
2024-03-06T21:01:12.427+0100    imported 100000 documents
[vagrant@oracle-21c-vagrant ~]$ mongoimport -d Clients -c Clients_12 --type csv --file /vagrant/Groupe_TPT_8/Clients_12.csv --headerline
2024-03-06T21:03:05.501+0100    connected to: localhost
2024-03-06T21:03:06.945+0100    imported 100000 documents
```

## Rapport du Projet de fin d'année Construction du data lake par étape

Cette commande est utilisée pour importer les données clients depuis un fichier CSV dans les collections "Clients\_7" et "Clients\_12" de la base de données "Clients" de MongoDB, en utilisant la première ligne du fichier CSV comme en-têtes de colonne et spécifiant le chemin local des fichiers Excel Clients.

```

024-01-06T20:48:43.362+0100 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
024-01-06T20:48:43.362+0100 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
024-01-06T20:48:43.362+0100 I CONTROL [initandlisten]

use Clients
switched to db Clients
db.Client_7.find()
{
    "_id": ObjectId("65e8cbf7686b262fffb4d70"),
    "age": 24,
    "sex": "F",
    "taux": 497,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 1,
    "Zemevoiture": "false",
    "immatriculation": "1674 SA 38"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d71"),
    "age": 73,
    "sex": "M",
    "taux": 118,
    "situationFamiliale": "Marié(e)",
    "nbEnfantsAcharge": 2,
    "Zemevoiture": "true",
    "immatriculation": "3188 JC 81"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d72"),
    "age": 52,
    "sex": "F",
    "taux": 542,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 2,
    "Zemevoiture": "false",
    "immatriculation": "2003 VU 12"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d73"),
    "age": 59,
    "sex": "M",
    "taux": 454,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 4,
    "Zemevoiture": "false",
    "immatriculation": "5738 NG 25"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d74"),
    "age": 79,
    "sex": "F",
    "taux": 238,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 3,
    "Zemevoiture": "false",
    "immatriculation": "2540 IQ 68"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d75"),
    "age": 44,
    "sex": "F",
    "taux": 162,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "true",
    "immatriculation": "9353 ZH 49"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d76"),
    "age": 81,
    "sex": "M",
    "taux": 883,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 3,
    "Zemevoiture": "false",
    "immatriculation": "2179 TX 94"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d77"),
    "age": 73,
    "sex": "M",
    "taux": 593,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "8579 ZM 93"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d78"),
    "age": 63,
    "sex": "M",
    "taux": 440,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 1,
    "Zemevoiture": "false",
    "immatriculation": "4343 RQ 30"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d79"),
    "age": 64,
    "sex": "F",
    "taux": 546,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 1,
    "Zemevoiture": "false",
    "immatriculation": "1437 AT 14"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d7a"),
    "age": 58,
    "sex": "M",
    "taux": 519,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "6347 SJ 79"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d7b"),
    "age": 59,
    "sex": "M",
    "taux": 519,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "5665 QZ 68"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d7c"),
    "age": 33,
    "sex": "M",
    "taux": 519,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 2,
    "Zemevoiture": "false",
    "immatriculation": "5665 QZ 68"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d7d"),
    "age": 36,
    "sex": "W",
    "taux": 711,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "1990 XZ 69"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d7e"),
    "age": 77,
    "sex": "M",
    "taux": 450,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "4814 OZ 59"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d7f"),
    "age": 62,
    "sex": "Masculin",
    "taux": 542,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 4,
    "Zemevoiture": "true",
    "immatriculation": "2371 BI 55"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d80"),
    "age": 67,
    "sex": "M",
    "taux": 511,
    "situationFamiliale": "Seule",
    "nbEnfantsAcharge": 1,
    "Zemevoiture": "false",
    "immatriculation": "8386 PH 27"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d81"),
    "age": 74,
    "sex": "M",
    "taux": 168,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "5712 00 15"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d82"),
    "age": 48,
    "sex": "M",
    "taux": 207,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "9493 NM 45"
}
{
    "_id": ObjectId("65e8cbf7686b262fffb4d83"),
    "age": 58,
    "sex": "M",
    "taux": 207,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 2,
    "Zemevoiture": "false",
    "immatriculation": "9131 GN 26"
}

ype "it" for more

db.Clients_12.find()
{
    "_id": ObjectId("65e8cbf9686b262fffb4d78"),
    "age": 57,
    "sex": "M",
    "taux": 462,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "4658 WP 57"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d79"),
    "age": 58,
    "sex": "F",
    "taux": 525,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 3,
    "Zemevoiture": "true",
    "immatriculation": "8157 VN 51"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d7b"),
    "age": 18,
    "sex": "M",
    "taux": 728,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "9715 CH 60"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d7c"),
    "age": 58,
    "sex": "M",
    "taux": 921,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 4,
    "Zemevoiture": "false",
    "immatriculation": "3684 VW 75"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d7d"),
    "age": 84,
    "sex": "M",
    "taux": 1256,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "6524 FN 51"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d7e"),
    "age": 39,
    "sex": "M",
    "taux": 1515,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 4,
    "Zemevoiture": "true",
    "immatriculation": "6523 FS 95"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d7f"),
    "age": 65,
    "sex": "M",
    "taux": 495,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 3,
    "Zemevoiture": "false",
    "immatriculation": "4121 JK 76"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d77"),
    "age": 18,
    "sex": "M",
    "taux": 493,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 1,
    "Zemevoiture": "true",
    "immatriculation": "7053 EM 18"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d78"),
    "age": 39,
    "sex": "M",
    "taux": 113,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "7053 EM 41"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d81"),
    "age": 29,
    "sex": "M",
    "taux": 447,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 1,
    "Zemevoiture": "true",
    "immatriculation": "586 KD 71"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d82"),
    "age": 81,
    "sex": "F",
    "taux": 511,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 1,
    "Zemevoiture": "true",
    "immatriculation": "3114 JK 75"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d83"),
    "age": 45,
    "sex": "M",
    "taux": 886,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 1,
    "Zemevoiture": "false",
    "immatriculation": "5585 IJ 38"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d84"),
    "age": 20,
    "sex": "M",
    "taux": 1170,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "1102 NS 82"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d85"),
    "age": 39,
    "sex": "M",
    "taux": 805,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "5516 KO 66"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d86"),
    "age": 72,
    "sex": "M",
    "taux": 1384,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "619 CX 55"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d87"),
    "age": 53,
    "sex": "M",
    "taux": 1147,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "8685 NR 59"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d88"),
    "age": 56,
    "sex": "M",
    "taux": 563,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 1,
    "Zemevoiture": "false",
    "immatriculation": "2671 UZ 75"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d89"),
    "age": 53,
    "sex": "M",
    "taux": 229,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "6318 LC 79"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d8a"),
    "age": 42,
    "sex": "M",
    "taux": 892,
    "situationFamiliale": "En Couple",
    "nbEnfantsAcharge": 2,
    "Zemevoiture": "false",
    "immatriculation": "6563 YX 74"
}
{
    "_id": ObjectId("65e8cbf9686b262fffb4d8b"),
    "age": 45,
    "sex": "F",
    "taux": 792,
    "situationFamiliale": "Coblitaire",
    "nbEnfantsAcharge": 0,
    "Zemevoiture": "false",
    "immatriculation": "8475 PR 87"
}

ype "it" for more

```

Toutes les données des clients ont été importées avec succès, comme vous pouvez le constater.

#### **5.4- Chargement du fichier « Immatriculations.csv » dans HBase**

## Présentation des données du fichier :

immatriculation	marque	nom	puissance	longueur	nbPlaces	nbPortes	couleur	occasion	prix
3176 TS 67	Renault	Laguna 2.0T		170 longue	5	5 blanc	False		27300
3721 QS 49	Volvo	S80 T6		272 tres longue	5	5 noir	False		50500
9099 UV 26	Volkswagen	Golf 2.0 FSI		150 moyenne	5	5 gris	True		16029
3563 LA 55	Peugeot	1007 1.4		75 courte	5	5 blanc	True		9625
6963 AX 34	Audi	A2 1.4		75 courte	5	5 gris	False		18310
5592 HQ 89	Skoda	Superb 2.8 V6		193 tres longue	5	5 bleu	False		31790
674 CE 26	Renault	Megane 2.0 16V		135 moyenne	5	5 gris	False		22350
1756 PR 31	Mercedes	A200		136 moyenne	5	5 noir	True		18130
6705 GX 50	BMW	120i		150 moyenne	5	5 noir	True		25060
4487 DR 75	Saab	9.3 1.8T		150 longue	5	5 gris	True		27202
7080 NW 34	Jaguar	X-Type 2.5 V6		197 longue	5	5 blanc	True		25970
9626 HF 36	Audi	A2 1.4		75 courte	5	5 rouge	False		18310
2401 PA 98	Volvo	S80 T6		272 tres longue	5	5 bleu	True		35350
826 VF 89	Renault	Laguna 2.0T		170 longue	5	5 rouge	False		27300
8216 GR 23	Skoda	Superb 2.8 V6		193 tres longue	5	5 bleu	False		31790
8076 YM 23	Jaguar	X-Type 2.5 V6		197 longue	5	5 noir	False		37100
9277 JN 49	BMW	M5		507 tres longue	5	5 rouge	True		66360
4231 HC 31	Audi	A2 1.4		75 courte	5	5 rouge	False		18310
2319 IQ 28	Ford	Mondeo 1.8		125 longue	5	5 gris	False		23900
148 RS 75	BMW	M5		507 tres longue	5	5 blanc	True		66360
6786 JV 36	Skoda	Superb 2.8 V6		193 tres longue	5	5 gris	False		31790
8049 KN 17	Renault	Megane 2.0 16V		135 moyenne	5	5 blanc	False		22350
9610 BR 52	Volkswagen	New Beetle 1.8		110 moyenne	5	5 blanc	True		18641
8745 KJ 12	Volkswagen	Polo 1.2 6V		55 courte	5	3 gris	False		12200
5805 YN 37	BMW	M5		507 tres longue	5	5 gris	True		66360

- **Immatriculation** : Identifiant unique d'immatriculation du véhicule.
  - **Marque** : Nom de la marque du véhicule parmi une liste prédéfinie.
  - **Nom** : Désignation du modèle spécifique du véhicule.
  - **Puissance** : La force du moteur du véhicule mesurée en chevaux DIN.

- **Longueur** : Catégorisation de la taille du véhicule.
- **NbPlaces** : Nombre de sièges disponibles dans le véhicule.
- **NbPortes** : Quantité de portes équipant le véhicule.
- **Couleur** : Teinte du véhicule.
- **Occasion** : Indication de si le véhicule est d'occasion.
- **Prix** : Montant de vente du véhicule exprimé en euros.

Nous établissons une connexion à HBase, puis nous créons une table nommée "Immatriculations" avec une colonne de famille "cf" destinée à stocker toutes les données relatives aux immatriculations.

```
[vagrant@oracle-21c-vagrant ~]$ start-dfs.sh
Starting namenodes on [localhost]

Starting datanodes
Starting secondary namenodes [oracle-21c-vagrant]
[vagrant@oracle-21c-vagrant ~]$
[vagrant@oracle-21c-vagrant ~]$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
[vagrant@oracle-21c-vagrant ~]$ start-hbase.sh
localhost: running zookeeper, logging to /usr/local/hbase/bin/../logs/hbase-vagrant-zookeeper-oracle-21c-vagrant.out
hbarunning master, logging to /usr/local/hbase/logs/hbase-vagrant-master-oracle-21c-vagrant.out
se shell : running regionserver, logging to /usr/local/hbase/logs/hbase-vagrant-regionserver-oracle-21c-vagrant.out
[vagrant@oracle-21c-vagrant ~]$ hbase shell

HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/book.html#shell
Version 3.0.0-beta-1, r119d11c808aefaf82d22fe6cd265981506b9dc09, Tue Dec 26 07:40:00 UTC 2023
Took 0.0020 seconds
```

```
hbase:018:0> create 'Immatriculations','cf'
Created table Immatriculations
Took 1.2601 seconds
=> Hbase::Table - Immatriculations
hbase:018:0>
```

Les données du fichier Excel "Immatriculations.csv" sont importées à l'aide de l'outil ImportTsv, ce qui facilite l'extraction et le transfert des données vers HBase :

```
vagrant@oracle-21c-vagrant vagrant]$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=',' -Dimporttsv.columns=HBASE_ROW_KEY,cf:marque,cf:nom,cf:puissance,cf:longueur,cf:nbPlaces,cf:nbPortes,cf:couleur,cf:occasion,f:prix Immatriculations /vagrant/Groupe_TPT_8/Immatriculations.csv
```

```

2024-04-28T00:56:28.060 INFO [LocalJobRunner Map Task Executor #0 ()] mapred.Task: em Counters
  FILE: Number of bytes read=1234003368
  FILE: Number of bytes written=1080407
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=
  HDFS: Number of bytes written=0
  HDFS: Number of read operations=0
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=0
  HDFS: Number of bytes read erasure-coded=0

Map-Reduce Framework
  Map input records=362643
  Map output records=362643
  Input split bytes=112
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=1795
  CPU time spent (ms)=5820
  Physical memory (bytes) snapshot=252100608
  Virtual memory (bytes) snapshot=2895712256
  Total committed heap size (bytes)=254304368
  Peak Map Physical memory (bytes)=254304368
  Peak Map Virtual memory (bytes)=2895712256

ImportInv
  Bad Lines=0
  File Input Format Counters
    Bytes Read=22294353
  File Output Format Counters
    Bytes Written=0

2024-04-28T00:56:28.067 INFO [main ()] mapreduce.Job: map 100% reduce 0%
2024-04-28T00:56:28.069 INFO [localJobRunner] mapred.LocalJobRunner: Finishing task: attempt_local1111416186_0001_m_000003_0
2024-04-28T00:56:29.070 INFO [main ()] mapreduce.Job: map 100% reduce 0%
2024-04-28T00:56:29.070 INFO [main ()] mapreduce.Job: job_local1111416186_0001 completed successfully
2024-04-28T00:56:29.217 INFO [main ()] mapreduce.Job: mters
  FILE: Number of bytes read=32685207
  FILE: Number of bytes written=31028
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=
  HDFS: Number of bytes written=0
  HDFS: Number of read operations=0
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=0
  HDFS: Number of bytes read erasure-coded=0

Map-Reduce Framework
  Map input records=2000001
  Map output records=2000001
  Input split bytes=448
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0

2024-04-28T00:56:29.217 INFO [main ()] mapreduce.Job: mters
  FILE: Number of bytes read=32685207
  FILE: Number of bytes written=31028
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=
  HDFS: Number of bytes written=0
  HDFS: Number of read operations=0
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=0
  HDFS: Number of bytes read erasure-coded=0

Map-Reduce Framework
  Map input records=2000001
  Map output records=2000001
  Input split bytes=448
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0

```

Cette commande utilise l'utilitaire ImportTsv pour importer des données depuis le fichier « Immatriculation.csv » vers une table HBase nommée "Immatriculations". Elle spécifie que la virgule est utilisée comme séparateur de champ dans le fichier CSV et définit les colonnes de la table HBase, en les associant à la famille de colonnes "cf".

```

hbase:007:0> scan 'Immatriculations', {LIMIT -> 20}
ROW                                         COLUMN+CELL
 0 AB 42                                     column=cf:couleur, timestamp=2024-04-01T18:30:59.881, value=rouge
 0 AB 42                                     column=cf:longueur, timestamp=2024-04-01T18:30:59.881, value=moyenne
 0 AB 42                                     column=cf:marque, timestamp=2024-04-01T18:30:59.881, value=Renault
 0 AB 42                                     column=cf:nbPlaces, timestamp=2024-04-01T18:30:59.881, value=5
 0 AB 42                                     column=cf:nbPortes, timestamp=2024-04-01T18:30:59.881, value=3
 0 AB 42                                     column=cf:puissance, timestamp=2024-04-01T18:30:59.881, value=160
 0 AB 42                                     column=cf:occasion, timestamp=2024-04-01T18:30:59.881, value=False
 0 AB 42                                     column=cf:prix, timestamp=2024-04-01T18:30:59.881, value=22350
 0 AB 42                                     column=cf:puissance, timestamp=2024-04-01T18:30:59.881, value=135
 0 AC 37                                     column=cf:couleur, timestamp=2024-04-01T18:30:59.881, value=bleu
 0 AC 37                                     column=cf:longueur, timestamp=2024-04-01T18:30:59.881, value=court
 0 AC 37                                     column=cf:marque, timestamp=2024-04-01T18:30:59.881, value=Volkswagen
 0 AC 37                                     column=cf:nbPlaces, timestamp=2024-04-01T18:30:59.881, value=5
 0 AC 37                                     column=cf:nbPortes, timestamp=2024-04-01T18:30:59.881, value=3
 0 AC 37                                     column=cf:nom, timestamp=2024-04-01T18:30:59.881, value=Polo 1.2 6V
 0 AC 37                                     column=cf:occasion, timestamp=2024-04-01T18:30:59.881, value=False
 0 AC 37                                     column=cf:prix, timestamp=2024-04-01T18:30:59.881, value=12200
 0 AC 37                                     column=cf:puissance, timestamp=2024-04-01T18:30:59.881, value=55
 0 AJ 71                                     column=cf:couleur, timestamp=2024-04-01T18:30:59.881, value=blanc
 0 AJ 71                                     column=cf:longueur, timestamp=2024-04-01T18:30:59.881, value=longue
 0 AJ 71                                     column=cf:marque, timestamp=2024-04-01T18:30:59.881, value=Jaguar
 0 AJ 71                                     column=cf:nbPlaces, timestamp=2024-04-01T18:30:59.881, value=5
 0 AJ 71                                     column=cf:nbPortes, timestamp=2024-04-01T18:30:59.881, value=5
 0 AJ 71                                     column=cf:nom, timestamp=2024-04-01T18:30:59.881, value=X-Type 2.5 V6
 0 AJ 71                                     column=cf:occasion, timestamp=2024-04-01T18:30:59.881, value=True
 0 AJ 71                                     column=cf:prix, timestamp=2024-04-01T18:30:59.881, value=25970
 0 AJ 71                                     column=cf:puissance, timestamp=2024-04-01T18:30:59.881, value=197
 0 AJ 74                                     column=cf:couleur, timestamp=2024-04-01T18:30:59.881, value=bleu
 0 AJ 74                                     column=cf:longueur, timestamp=2024-04-01T18:30:59.881, value=moyenne
 0 AJ 74                                     column=cf:marque, timestamp=2024-04-01T18:30:59.881, value=BMW
 0 AJ 74                                     column=cf:nbPlaces, timestamp=2024-04-01T18:30:59.881, value=5
 0 AJ 74                                     column=cf:nbPortes, timestamp=2024-04-01T18:30:59.881, value=5
 0 AJ 74                                     column=cf:nom, timestamp=2024-04-01T18:30:59.881, value=120i
 0 AJ 74                                     column=cf:occasion, timestamp=2024-04-01T18:30:59.881, value=True
 0 AJ 74                                     column=cf:prix, timestamp=2024-04-01T18:30:59.881, value=25060
 0 AS 40                                     column=cf:puissance, timestamp=2024-04-01T18:30:59.881, value=150
 0 AM 40                                     column=cf:couleur, timestamp=2024-04-01T18:30:59.881, value=gris
 0 AM 40                                     column=cf:longueur, timestamp=2024-04-01T18:30:59.881, value=courte
 0 AM 40                                     column=cf:marque, timestamp=2024-04-01T18:30:59.881, value=Peugeot

```

Visualisation des 20 premières lignes en utilisant la commande "scan" avec la clause "LIMIT" pour spécifier le nombre de lignes à afficher.

```

hbase:003:0> count 'Immatriculations'
Current count: 1000, row: 1000 YY 97
Current count: 2000, row: 1006 HH 38
Current count: 3000, row: 1010 IB 59
Current count: 4000, row: 1015 CG 68
Current count: 5000, row: 1019 RA 38
Current count: 5000, row: 1023 EH 70

```

```

Current count: 1967000, row: 9864 PJ 71
Current count: 1968000, row: 9869 XR 77
Current count: 1969000, row: 9874 AS 17
Current count: 1970000, row: 9878 ZT 67
Current count: 1971000, row: 9883 CE 22
Current count: 1972000, row: 9888 FQ 64
Current count: 1973000, row: 9892 GU 24
Current count: 1974000, row: 9897 DX 40
Current count: 1975000, row: 9900 IO 92
Current count: 1976000, row: 9905 DP 60
Current count: 1977000, row: 991 FK 36
Current count: 1978000, row: 9914 IV 34
Current count: 1979000, row: 9919 PA 21
Current count: 1980000, row: 9923 UZ 54
Current count: 1981000, row: 9928 TA 47
Current count: 1982000, row: 9933 DJ 32
Current count: 1983000, row: 9938 DK 20
Current count: 1984000, row: 9941 XH 10
Current count: 1985000, row: 9947 AS 75
Current count: 1986000, row: 9951 DF 25
Current count: 1987000, row: 9956 FE 72
Current count: 1988000, row: 996 YV 33
Current count: 1989000, row: 9965 BN 69
Current count: 1990000, row: 997 CO 84
Current count: 1991000, row: 9973 YY 91
Current count: 1992000, row: 9979 AK 51
Current count: 1993000, row: 9982 YC 74
Current count: 1994000, row: 9987 UF 53
Current count: 1995000, row: 9991 YG 69
Current count: 1996000, row: 9996 UX 94
1996632 rows(s)
Took 66.7780 seconds
=> 1996632

```

Comme vous pouvez constater, l'importation des données a été réalisée avec succès, englobant un total de 1 996 633 enregistrements d'immatriculations et de leurs détails, ce qui a été confirmé en utilisant la commande « count ».

## 5.5- Chargement du fichier « Marketing.csv » dans MySQL

Avant d'importer les données marketing dans MySQL, il serait utile de procéder à une présentation des données :

A	B	C	D	E	F	G
age	sexe	taux	situationFamiliale	nbEnfantsAcharge	2eme voiture	
21	F	1396	Célibataire		0	false
35	M	223	Célibataire		0	false
48	M	401	Célibataire		0	false
26	F	420	En Couple		3	true
80	M	530	En Couple		3	false
27	F	153	En Couple		2	false
59	F	572	En Couple		2	false
43	F	431	Célibataire		0	false
64	M	559	Célibataire		0	false
22	M	154	En Couple		1	false
79	F	981	En Couple		2	false
55	M	588	Célibataire		0	false
19	F	212	Célibataire		0	false
34	F	1112	En Couple		0	false
60	M	524	En Couple		0	true
22	M	411	En Couple		3	true
58	M	1192	En Couple		0	false
54	F	452	En Couple		3	true
35	M	589	Célibataire		0	false
59	M	748	En Couple		0	true

- **Age** : Donnée numérique représentant l'âge du client en années.
- **Sexe** : Le genre de la personne.
- **Taux** : La capacité d'endettement du client en euros
- **SituationFamiliale** : La situation familiale du client.
- **NbEnfantsAcharge** : Le nombre d'enfants à charge du client.
- **2eme voiture** : Indique si le client possède déjà un véhicule principal.

Tout d'abord, nous créons une base de données MySQL nommée "Marketing\_MBDS", puis nous la sélectionnons en utilisant la commande « USE ».

```
mysql>
mysql> Create Database Marketing_MBDS;
ERROR 1007 (HY000): Can't create database 'Marketing_MBDS'; database exists
mysql> Use Marketing_MBDS;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
```

On utilise l'instruction « LOCAL INFILE » pour importer les données clients du fichier Marketing dans la base de donnée MySQL :

```
mysql> Load data local infile '/vagrant/Groupe_TPT_8/Marketing.csv' Into table Marketing_Db fields terminated by ',' lines terminated by '\n' ignore 1 rows;
ERROR 3948 (42000): Loading local data is disabled; this must be enabled on both the client and server sides
mysql> ■
```

Une erreur s'est produite, signalant que l'importation de données locales (via l'instruction LOCAL INFILE) est désactivée. Pour utiliser cette fonctionnalité, elle doit être activée à la fois du côté client et du côté serveur. Cela implique que le serveur MySQL doit être configuré avec le paramètre local\_infile activé, et que le client MySQL doit être également configuré pour autoriser l'utilisation de données locales lors des chargements de fichiers.

```
mysql> SHOW GLOBAL VARIABLES LIKE 'local_infile';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| local_infile | OFF  |
+-----+-----+
1 row in set (0.00 sec)

mysql> SET GLOBAL local_infile='ON';
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GLOBAL VARIABLES LIKE 'local_infile';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| local_infile | ON   |
+-----+-----+
1 row in set (0.01 sec)
```

```
mysql> Load data local infile '/vagrant/Groupe_TPT_8/Marketing.csv' Into table Marketing_Db fields terminated by ',' lines terminated by '\n' ignore 1 rows;
Query OK, 20 rows affected, 15 warnings (0.17 sec)
Records: 20 Deleted: 0 Skipped: 0 Warnings: 15
■
```

Après l'activation de la fonctionnalité "LOCAL INFILE", nous avons importé les données clients à partir du fichier Marketing. Cette commande charge les données d'un fichier CSV local dans une table nommée "Marketing\_Db", en délimitant les champs par des virgules, les lignes par des retours à la ligne, et en ignorant la première ligne du fichier. Nous allons à présent examiner ces données pour confirmer le bon déroulement de l'importation.

```
mysql> SELECT * FROM Marketing_Db \G; | sed 's/\r//';
***** 1. row *****
    age: 21
    sexe: F
    taux: 1396
situationFamiliale: Célibataire
    nbEnfantsAcharge: 0
    deuxieme_voiture: false
***** 2. row *****
    age: 35
    sexe: M
    taux: 223
situationFamiliale: Célibataire
    nbEnfantsAcharge: 0
    deuxieme_voiture: false
***** 3. row *****
    age: 48
    sexe: M
    taux: 401
situationFamiliale: Célibataire
    nbEnfantsAcharge: 0
    deuxieme_voiture: false
***** 4. row *****
    age: 26
    sexe: F
    taux: 420
situationFamiliale: En Couple
    nbEnfantsAcharge: 3
    deuxieme_voiture: true
***** 5. row *****
    age: 80
    sexe: M
    taux: 530
situationFamiliale: En Couple
    nbEnfantsAcharge: 3
    deuxieme_voiture: false
***** 6. row *****
    age: 27
    sexe: F
    taux: 153
situationFamiliale: En Couple
    nbEnfantsAcharge: 2
    deuxieme_voiture: false
***** 7. row *****
    age: 59
    sexe: F
    taux: 572
situationFamiliale: En Couple
    nbEnfantsAcharge: 2
    deuxieme_voiture: false
```

Cette commande SQL interroge la table "Marketing\_Db" pour sélectionner toutes les colonnes et toutes les lignes de données. L'option "\G" permet d'afficher les résultats sous forme de liste verticale plutôt que de les afficher horizontalement. Ensuite, la commande "sed 's/\r//'" est utilisée pour supprimer les caractères de retour chariot (\r) des résultats, ce qui peut être nécessaire pour formater correctement la sortie, notamment lorsque les données proviennent de fichiers CSV ou Excel. On remarque que l'ensemble des données du fichier marketing ont été chargé correctement dans la table « Marketing\_Db ».

On utilise la commande suivante afin de vérifier si le nombre de lignes de la table correspond bien au nombre de lignes du fichier "Marketing.csv" (on expecte 20 ligne) :

```
mysql> SELECT COUNT(*) FROM Marketing_Db;
+-----+
| COUNT(*) |
+-----+
|      20 |
+-----+
1 row in set (0.40 sec)

mysql> -
```

## 5.6- Transfert des données des clients stockées dans MongoDB vers la datalake Hive.

Pour commencer, nous établissons une connexion à Hive : Ces commandes ci-dessous sont utilisées pour démarrer les services Hive (Metastore et HiveServer2) en arrière-plan et pour ouvrir une session Beeline afin de se connecter à HiveServer2 et exécuter des requêtes Hive. Ensuite nous créons une base de données nommée "MBDS\_Projet" et nous la sélectionnons en utilisant la commande "USE".

```
Last login: Wed Apr 24 20:04:24 2024 from 10.0.2.2
[vagrant@oracle-21c-vagrant ~]$ nohup hive --service metastore > hive_metastore.log 2>&1 &
[1] 9374
[vagrant@oracle-21c-vagrant ~]$ nohup hiveserver2 > hive_server.log 2>&1 &
[2] 9514
[vagrant@oracle-21c-vagrant ~]$ beeline -u jdbc:hive2://localhost:10000 vagrant
*Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 2.3.9)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.9 by Apache Hive
0: jdbc:hive2://localhost:10000> -
```

```
0: jdbc:hive2://localhost:10000> CREATE DATABASE IF NOT EXISTS MBDS_Projet
0: jdbc:hive2://localhost:10000> ->
24/04/03 03:27:19 INFO ql.Driver: Compiling command[queryId=vagrant_20240403032719_c5f21d35-2e8d-4971-92c9-d499d8189548]: CREATE DATABASE IF NOT EXISTS MBDS_Projet
24/04/03 03:27:19 INFO tnx.TxnHandler: Added entries to MIN_HISTORY_LEVEL for current txns: [(627)] with min_open_txn: 627
24/04/03 03:27:19 INFO ql.Driver: Semantic Analysis Completed (retiral = false)
24/04/03 03:27:19 INFO ql.Driver: Returning Hive schema: Schema[fieldSchemas=null, properties=null]
24/04/03 03:27:19 INFO ql.Driver: Completed compiling command[queryId=vagrant_20240403032719_c5f21d35-2e8d-4971-92c9-d499d8189548]; Time taken: 0.294 seconds
24/04/03 03:27:19 INFO ql.Driver: ReexecDriver: Execution #1 of query
24/04/03 03:27:19 INFO logmgr.DlmManager: Setting lock to request transaction to txnid:627 for queryId=vagrant_20240403032719_c5f21d35-2e8d-4971-92c9-d499d8189548
24/04/03 03:27:19 INFO ql.Driver: Executing command[queryId=vagrant_20240403032719_c5f21d35-2e8d-4971-92c9-d499d8189548]: CREATE DATABASE IF NOT EXISTS MBDS_Projet
24/04/03 03:27:19 INFO sqldist.SQLStandaloneAccessController: Starting task [Stage-0:0:0L] in serial mode
24/04/03 03:27:19 INFO sqldist.SQLStandaloneAccessController: Created SQLStandaloneAccessController for session context : HiveAuthSessionContext [sessionIdString=af679ec-2d2a-4731-8478-254ae64f6a1a, clientType=HIVESERVER2]
24/04/03 03:27:19 INFO metastore.HiveMetaStore: 18: create_database: Database(name:MBDS_Projet, description:null, locationUri:hdfs://localhost:9000/user/hive/warehouse/mbds_projet.db, parameters:null, ownerName:vagrant, ownerType:USER, catalogName:hive)
24/04/03 03:27:19 INFO HiveMetaStore:audit: ugi=vagrant ip=unknown-ip-addr cmd=create_database: Database(name:MBDS_Projet, description:null, locationUri:hdfs://localhost:9000/user/hive/warehouse/mbds_projet.db, parameters:null, ownerName:vagrant, ownerType:USER, catalogName:hive)
24/04/03 03:27:19 INFO metastore.HiveMetaStore: 18: Opening new store with implementation class:org.apache.hadoop.hive.metastore.ObjectStore
24/04/03 03:27:19 INFO metastore.MetaStoreDirectSql: Using direct SQL, underlying DB is MySQL
24/04/03 03:27:19 INFO metastore.ObjectStore: Initialized ObjectStore
24/04/03 03:27:19 WARN metastore.ObjectStore: Failed to get database hive.MBDS_Projet, returning NoSuchObjectException
24/04/03 03:27:19 INFO util.FileUtils: Creating directory if it doesn't exist: hdfs://localhost:9000/user/hive/warehouse/mbds_projet.db
24/04/03 03:27:19 INFO ql.Driver: Completed executing command[queryId=vagrant_20240403032719_c5f21d35-2e8d-4971-92c9-d499d8189548]; Time taken: 0.416 seconds
24/04/03 03:27:20 INFO ql.Driver: OK
24/04/03 03:27:20 INFO logmgr.ObxTxnManager: Stopped heartbeat for query: vagrant_20240403032719_c5f21d35-2e8d-4971-92c9-d499d8189548
24/04/03 03:27:20 INFO tnx.TxnHandler: Expected to move at least one record from txm_components to completed_txm_components when committing txnl txnid:627
24/04/03 03:27:20 INFO tnx.TxnHandler: Removed committed transaction: (627) from MIN_HISTORY_LEVEL
0: root@vagrant:~#
```

0: jdbc:hive2://localhost:10000> USE MBDS\_Projet;
24/04/03 03:27:32 INFO ql.Driver: Compiling command[queryId=vagrant\_20240403032732\_dc20444b-b106-48a7-8d40-803ac1e011ef]: USE MBDS\_Projet
24/04/03 03:27:32 INFO metastore.HiveMetaStore: 17: get\_database: @hive#MBDS\_Projet
24/04/03 03:27:32 INFO HiveMetaStore.audit: ugi=vagrant ip=unknown-ip-addr cmd=get\_database: @hive#MBDS\_Projet
24/04/03 03:27:32 INFO ql.Driver: Semantic Analysis Completed (retiral = false)
24/04/03 03:27:32 INFO ql.Driver: Returning Hive schema: Schema[fieldSchemas=null, properties=null]
24/04/03 03:27:32 INFO ql.Driver: Completed compiling command[queryId=vagrant\_20240403032732\_dc20444b-b106-48a7-8d40-803ac1e011ef]; Time taken: 0.03 seconds
24/04/03 03:27:32 INFO ql.Driver: Executing command[queryId=vagrant\_20240403032732\_dc20444b-b106-48a7-8d40-803ac1e011ef]: USE MBDS\_Projet
24/04/03 03:27:32 INFO ql.Driver: Starting task [Stage-0:0:0L] in serial mode
24/04/03 03:27:32 INFO HiveMetaStore.audit: ugi=vagrant ip=unknown-ip-addr cmd=executeStatement: @hive#MBDS\_Projet
24/04/03 03:27:32 INFO metastore.HiveMetaStore: 19: get\_database: @hive#MBDS\_Projet
24/04/03 03:27:32 INFO metastore.MetaStoreDirectSql: Using direct SQL, underlying DB is MySQL
24/04/03 03:27:32 INFO metastore.ObjectStore: Initialized ObjectStore
24/04/03 03:27:32 INFO metastore.HiveMetaStore: 19: get\_database: @hive#MBDS\_Projet
24/04/03 03:27:32 INFO HiveMetaStore.audit: ugi=vagrant ip=unknown-ip-addr cmd=get\_database: @hive#MBDS\_Projet
24/04/03 03:27:32 INFO ql.Driver: Completed executing command[queryId=vagrant\_20240403032732\_dc20444b-b106-48a7-8d40-803ac1e011ef]; Time taken: 0.528 seconds
0: root@vagrant:~#

```
1: jdbc:hive2://localhost:10000> USE MBDS_Projet;
20/04/24 20:52:53 INFO ql.Driver: Compiling command(queryId=vagrant_20240424205253_e784a888-6590-44b7-9ca0-b49b2b84e9bc): USE MBDS_Projet
20/04/24 20:52:53 INFO metastore.HiveMetaStore: 2: get_database: @hive#MBDS_Projet
20/04/24 20:52:53 INFO HiveMetaStore.audit: ugi=vagrant ip=unknown-ip-addr cmd=get_database: @hive#MBDS_Projet
20/04/24 20:52:53 INFO ql.Driver: Semantic Analysis Completed (retrial = false)
20/04/24 20:52:54 INFO ql.Driver: Returning Hive schema: Schema(fieldSchemas:null, properties:null)
20/04/24 20:52:54 INFO ql.Driver: Completed compiling command(queryId=vagrant_20240424205253_e784a888-6590-44b7-9ca0-b49b2b84e9bc); Time taken: 0.895 seconds
20/04/24 20:52:54 INFO reexec.ReExecDriver: Execution #1 of query
20/04/24 20:52:54 INFO ql.Driver: Executing command(queryId=vagrant_20240424205253_e784a888-6590-44b7-9ca0-b49b2b84e9bc): USE MBDS_Projet
20/04/24 20:52:54 INFO ql.Driver: Starting task [Stage-0:DDL] in serial mode
20/04/24 20:52:54 INFO metastore.HiveMetaStore: 4: get_database: @hive#MBDS_Projet
20/04/24 20:52:54 INFO metastore.HiveMetaStore: 4: Opening raw store with implementation class:org.apache.hadoop.hive.metastore.ObjectStore
20/04/24 20:52:54 INFO metastore.ObjectStore: ObjectStore, initialize called
20/04/24 20:52:54 INFO metastore.MetaStoreDirectSql: Using direct SQL, underlying DB is MYSQL
20/04/24 20:52:54 INFO metastore.ObjectStore: Initialized ObjectStore
20/04/24 20:52:54 INFO metastore.HiveMetaStore: 4: get_database: @hive#MBDS_Projet
20/04/24 20:52:54 INFO HiveMetaStore.audit: ugi=vagrant ip=unknown-ip-addr cmd=get_database: @hive#MBDS_Projet
20/04/24 20:52:54 INFO ql.Driver: Completed executing command(queryId=vagrant_20240424205253_e784a888-6590-44b7-9ca0-b49b2b84e9bc); Time taken: 0.142 seconds
20/04/24 20:52:54 INFO ql.Driver: OK
0 rows affected (1.342 seconds)
0: jdbc:hive2://localhost:10000
```

Ces commandes ci-dessous créent deux table externe nommée "client\_7\_ext" et « client\_12\_ext » dans Hive, avec des colonnes correspondant à différents attributs de clients. Les données de cette table sont stockées en utilisant le gestionnaire de stockage MongoStorageHandler pour permettre l'intégration avec MongoDB. Les propriétés du SERDE (Serializer/Deserializer) sont définies pour mapper les colonnes de la table Hive aux champs correspondants dans la collection MongoDB. Les propriétés de la table (TBLPROPERTIES) spécifient l'URI de connexion à la base de données MongoDB (Collection « Clients 12 » et « Clients 7 ») où les données sont stockées.

```
CREATE EXTERNAL TABLE client_12_ext (
    . . . . . age INT,
    . . . . . sexe STRING,
    . . . . . taux INT,
    . . . . . situationFamiliale STRING,
    . . . . . nbEnfantsAcharge INT,
    . . . . . deuxiemeVoiture STRING,
    . . . . . immatriculation STRING
)
. . . . . STORED BY 'com.mongodb.hadoop.hive.MongoStorageHandler'
. . . . . WITH SERDEPROPERTIES('mongo.columns.mapping'='{
    . . . . . "age":"age",
    . . . . . "sexe":"sexe",
    . . . . . "taux":"taux",
    . . . . . "situationFamiliale":"situationFamiliale",
    . . . . . "nbEnfantsAcharge":"nbEnfantsAcharge",
    . . . . . "deuxiemevoiture":"2eme voiture",
    . . . . . "immatriculation":"immatriculation"
    . . . . . }')
. . . . . TBLPROPERTIES('mongo.uri'='mongodb://localhost:27017/Clients.Clients_12');
```

Nous vérifions si les tables "client\_7\_ext" et "client\_12\_ext" sont correctement chargées, que leurs données sont exactes, et que chaque table contient exactement 100 000 lignes à partir des Collections « Clients\_7 » et « Clients\_12 » de la Base de données « Clients » de MongoDB :

0: jdbc:hive2://localhost:10000> SELECT \* FROM client\_7\_ext LIMIT 10;

client_7_ext.age   client_7_ext sexe   client_7_ext.taux   client_7_ext.situationfamiliale   client_7_ext.nbenfantssachage   client_7_ext.deuxiemevoiture   client_7_ext.lmatriculation						
24	F	427	En Couple	1	false	1674 SA 38
73	M	1181	Marié(e)	3	true	3188 JC 82
52	F	542	En Couple	2	false	2083 VX 12
59	M	454	En Couple	4	false	5758 NG 25
79	F	238	En Couple	3	false	2540 IQ 68
44	F	162	En Couple	0	true	9353 ZH 49
81	M	883	En Couple	3	false	2179 XT 94
73	M	593	Célibataire	0	false	8579 2W 93
64	F	546	En Couple	1	false	1437 AT 14
63	M	440	En Couple	1	false	4343 RQ 30

```
0: jdbc:hive2://localhost:10000> SELECT * FROM client_12_ext LIMIT 10;
```

client_12_ext.age	client_12_ext.sexé	client_12_ext.taux	client_12_ext.situationfamiliale	client_12_ext.nbenfantscharge	client_12_ext.deuxiemevoiture	client_12_ext.immatriculation
58	M	921	En Couple	4	false	3684 VW 75
57	M	462	Célibataire	0	false	4658 VW 57
58	F	525	En Couple	3	true	8157 VN 51
18	M	728	Célibataire	0	false	9715 CH 68
33	M	1113	Célibataire	0	false	7679 ZL 41
84	M	1256	En Couple	0	false	8406 FN 51
19	M	515	En Couple	4	true	6523 SF 95
65	M	495	En Couple	3	false	4121 KK 76
18	M	493	En Couple	1	true	7853 EM 18
53	M	229	En Couple	0	false	6318 LC 79

0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) FROM client\_7\_ext;

```

24/03/14 00:17:58 INFO exec.Task: 2024-03-14 00:17:58,582 Stage-1 map = 0%, reduce = 0%
24/03/14 00:17:59 INFO exec.Task: 2024-03-14 00:17:59,749 Stage-1 map = 100%, reduce = 0%
24/03/14 00:18:01 INFO exec.Task: 2024-03-14 00:18:01,975 Stage-1 map = 100%, reduce = 100%
24/03/14 00:18:01 INFO exec.Task: Ended Job = job_local1758268510_0002
24/03/14 00:18:01 INFO ql.Driver: MapReduce Jobs Launched:
24/03/14 00:18:01 INFO ql.Driver: Stage-Stage-1: HDFS Read: 37584 HDFS Write: 0 SUCCESS
24/03/14 00:18:01 INFO ql.Driver: Total MapReduce CPU Time Spent: 0 msec
24/03/14 00:18:01 INFO ql.Driver: Completed executing command(queryId=vagrant_20240314001755_875f0560-040d-429e-9eb7-a6ee1fc97c16); Time taken: 6.179 seconds
24/03/14 00:18:01 INFO ql.Driver: OK
24/03/14 00:18:01 INFO lockmgr.DbTxnManager: Stopped heartbeat for query: vagrant_20240314001755_875f0560-040d-429e-9eb7-a6ee1fc97c16
24/03/14 00:18:01 INFO txn.TxnHandler: Expected to move at least one record from txn_components to completed_txn_components when committing txn! txnid:24
24/03/14 00:18:02 INFO txn.TxnHandler: Removed committed transaction: (24) from MIN_HISTORY_LEVEL
-----+
| _c0 |
-----+
| 100000 |
-----+
1 row selected (6.559 seconds)
0: jdbc:hive2://localhost:10000>

```

0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) FROM client\_12\_ext;

	_c0	
	100000	

## 5.7- Transfert des données d'immatriculation stockées dans HBase vers la datalake Hive :

À présent, nous procémons à l'extraction de toutes les données de la table HBase "Immatriculations" vers la Data Lake Hive "MBDS\_Projet". Cette commande ci-dessous crée une table externe nommée "table\_ext\_immatriculations" dans Apache Hive. Cette table est configurée pour stocker les données dans HBase à l'aide du gestionnaire de stockage "HBaseStorageHandler" d'Hive. Les colonnes de la table sont définies avec leurs types de données respectifs, et les correspondances entre les colonnes HBase et les colonnes Hive sont spécifiées dans les propriétés SERDEPROPERTIES. Enfin, les propriétés de la table sont définies pour spécifier le nom de la table HBase où les données seront stockées.

## Rapport du Projet de fin d'année Construction du data lake par étape

```

[jdbc:hive2://localhost:10000] CREATE EXTERNAL TABLE table_ext_immatriculations (
    >     immatriculation STRING,
    >     marque STRING,
    >     nom STRING,
    >     puissance INT,
    >     longueur STRING,
    >     nbPlaces INT,
    >     nbPortes INT,
    >     couleur STRING,
    >     occasion BOOLEAN,
    >     prix INT
    > )
    > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES (
    >     "hbase.columns.mapping" = ":key,cf:marque,cf:nom,cf:puissance,cf:longueur,cf:nbPlaces,cf:nbPortes,cf:couleur,cf:occasion,cf:prix"
    > )
    > TBLPROPERTIES("hbase.table.name" = "Immatriculations");
24/03/31 21:40:55 INFO jdbc.Driver: Compiling command(queryId=vagrant_20240331214055_9ec6b4f1-efd9-4dc9-b2e4-1f16b72c61d3): CREATE EXTERNAL TABLE table_ext_immatriculations (
immatriculation STRING,
marque STRING,
nom STRING,
puissance INT,
longueur STRING,
nbPlaces INT,
nbPortes INT,
couleur STRING,
occasion BOOLEAN,
prix INT
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES (
"hbase.columns.mapping" = ":key,cf:marque,cf:nom,cf:puissance,cf:longueur,cf:nbPlaces,cf:nbPortes,cf:couleur,cf:occasion,cf:prix"
)
TBLPROPERTIES("hbase.table.name" = "Immatriculations")
24/03/31 21:40:55 INFO txn.TxnHandler: Added entries to MIN_HISTORY_LEVEL for current txns: ([461]) with min_open_txn: 461
24/03/31 21:40:55 INFO parse.CalcitePlanner: Starting Semantic Analysis
24/03/31 21:40:55 INFO parse.CalcitePlanner: Creating table default.table_ext_immatriculations position=22
24/03/31 21:40:55 INFO metastore.HiveMetaStore: 3: get database: @hive#default
24/03/31 21:40:55 INFO HiveMetaStore.audit: ugi=vagrant ip=unknown-ip-addr cmd=get_database: @hive#default
24/03/31 21:40:55 INFO ql.Driver: Semantic Analysis Completed (retrial = false)
24/03/31 21:40:55 INFO ql.Driver: Returning Hive schema: Schema(fieldschemas=null,properties=null)
24/03/31 21:40:55 INFO reexec.HiveDriver: Compiled compiling command(queryId=vagrant_20240331214055_9ec6b4f1-efd9-4dc9-b2e4-1f16b72c61d3); Time taken: 0.086 seconds
24/03/31 21:40:55 INFO reexec.HiveDriver: Execution #1 of query
24/03/31 21:40:55 INFO lockman.DBIoManager: Setting lock request transaction to txnid:461 for queryId=vagrant_20240331214055_9ec6b4f1-efd9-4dc9-b2e4-1f16b72c61d3

```

Nous allons vérifier l'intégrité des données de la table "table\_ext\_immatriculations" pour s'assurer que les données sont correctement chargées. Nous essayerons de confirmer également que chaque table contient exactement 1 996 633 lignes, extraites de la table HBase "Immatriculations".

**0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) FROM table\_ext immatriculations;**

_c0
1996633
1 row selected (33.98)

0: jdbc:hive2://localhost:10000> SELECT \* FROM table\_ext\_immatriculations LIMIT 10;

table_ext_immatriculations.immatriculation	table_ext_immatriculations.marque	table_ext_immatriculations.nom	table_ext_immatriculations.puissance	table_ext_immatriculations.longueur	table_ext_immatriculations.nbplaces
0 AB 42	Renault	Megane 2.0 16V	135	moyenne	5
0 AC 37	bleu	Volkswagen	false	Polo 1.2 6V	22350
0 AJ 71	blanc	Jaguar	true	X-Type 2.5 V6	55
0 AS 74	bleu	BMW	true	12000	197
0 AW 40	gris	Peugeot	false	25970	150
0 AX 62	blanc	Volvo	true	1007 1.4	25860
0 AX 73	rouge	Ford	false	S80 T6	175
0 BC 73	noir	Audi	true	Mondeo 1.8	35350
0 BH 31	noir	BMW	false	A2 1.4	125
0 BH 74	noir	Jaguar	false	23980	75
0 BJ 79	gris	Volvo	true	18310	courte
0 BM 45	rouge	Peugeot	false	18310	507
0 BZ 21	bleu	Audi	true	X-Type 2.5 V6	94800
0 CD 43	noir	BMW	false	197	37100
0 CF 54	bleu	BMW	false	S80 T6	272
0 CG 81	gris	Audi	true	35350	175
0 CJ 90	blanc	BMW	false	1007 1.4	13750
0 CO 17	bleu	Renault	true	A2 1.4	18310
0 CQ 77	noir	Peugeot	false	Laguna 2.0T	507
0 CY 48	noir	Volvo	true	19110	9625
				75	12817
				272	50500
				moyenne	courte

## 5.8- Transfert des données marketing stockées dans MySQL vers la datalake Hive.

On commence par la création d'une table nommée "table\_marketing" avec plusieurs colonnes pour stocker des données relatives au marketing :

```
mysql> USE Marketing_MBDS
Database changed
mysql> CREATE TABLE Marketing_Db (    age INT,    sexe CHAR(1),    taux INT,    situationFamiliale VARCHAR(20),    nbEnfantsAcharge INT,    deuxieme_voiture VARCHAR(5) );
Query OK, 0 rows affected (0.22 sec)
```

Ensuite on utilise Sqoop pour importer des données depuis une base de données MySQL nommée "Marketing\_MBDS", en spécifiant le pilote JDBC de MySQL, le nom d'utilisateur admin « root » et la table source "Marketing\_Db".

Elle supprime le répertoire de destination cible (s'il existe), spécifie le délimiteur de champ et de ligne pour le fichier de données, et active l'importation vers Hive en spécifiant la table Hive cible "table\_marketing". Enfin, elle définit le répertoire cible dans HDFS et utilise un seul thread pour le transfert de données.

```
[vagrant@oracle-21c-vagrant ~]$  
[vagrant@oracle-21c-vagrant ~]$ sqoop import -D org.apache.sqoop.splitter.allow_text_splitter=true \  
> --connect jdbc:mysql://localhost:3306/Marketing_MBDS?characterEncoding=latin1 \  
> --driver com.mysql.cj.jdbc.Driver --target-dir /user/vagrant/Marketing_MBDS.Marketing_Db \  
> --username root \  
> --table Marketing_Db \  
> --fields-terminated-by ',' \  
> --lines-terminated-by '\n' \  
> --hive-import \  
> --hive-table MBDS_Projet.table_marketing \  
> -m 1
```

Nous confirmons que la table Hive "table\_Marketing" a été chargée avec succès à partir du traitement réalisé via Sqoop.

```
mysql> SELECT COUNT(*) FROM Marketing_Db;  
+-----+  
| COUNT(*) |  
+-----+  
|      20 |  
+-----+  
1 row in set (0.40 sec)  
  
mysql> ■
```

**0: jdbc:hive2://localhost:10000> SELECT \* FROM Marketing\_Db LIMIT 10;**

table_marketing.age	table_marketing.sexe	table_marketing.taux	table_marketing.situationfamiliale	table_marketing.nbenfantsachage	table_marketing.deuxieme_voiture
21	F	1396	Célibataire	0	false
35	M	223	Célibataire	0	false
48	M	481	Célibataire	0	false
26	F	420	En Couple	3	true
80	M	538	En Couple	3	false
27	F	153	En Couple	2	false
59	F	572	En Couple	2	false
43	F	431	Célibataire	0	false
64	M	559	Célibataire	0	false
22	M	154	En Couple	1	false
79	F	981	En Couple	2	false
55	M	588	Célibataire	0	false
19	F	212	Célibataire	0	false
34	F	1112	En Couple	0	false
60	M	524	En Couple	0	true
22	M	411	En Couple	3	true
58	M	1192	En Couple	0	false
54	F	452	En Couple	3	true
35	M	589	Célibataire	0	false
59	M	748	En Couple	0	true

## 5.9- Transformation des données d'immatriculation de la table Hive externe « table\_immatriculations\_ext ».

L'objectif de cette section est d'analyser et de valider la conformité des données d'immatriculation avec le dictionnaire de données mentionné ci-dessous :

*Immatriculations.csv : informations sur les immatriculations effectuées cette année*

Attribut	Type	Description	Domaine de valeurs
Immatriculation	caractères	Numéro unique d'immatriculation du véhicule	Texte au format « 9999 AA 99 »
Marque	caractères	Nom de la marque du véhicule	Audi, BMW, Dacia, Daihatsu, Fiat, Ford, Honda, Hyundai, Jaguar, Kia, Lancia, Mercedes, Mini, Nissan, Peugeot, Renault, Saab, Seat, Skoda, Volkswagen, Volvo

Nom	caractères	Nom du modèle de véhicule	S80 T6, Touran 2.0 FSI, Polo 1.2 6V, New Beatle 1.8, Golf 2.0 FSI, Superb 2.8 V6, Toledo 1.6, 9.3 1.8T, Vel Satis 3.5 V6, Megane 2.0 16V, Laguna 2.0T, Espace 2.0T, 1007 1.4, Primera 1.6, Maxima 3.0 V6, Almera 1.8, Copper 1.6 16V, S500, A200, Ypsilon 1.4 16V, Picanto 1.1, X-Type 2.5 V6, Matrix 1.6 FR-V 1.7, Mondeo 1.8, Croma 2.2, Cuore 1.0, Logan 1.6 MPI, M5, 120i, A3 2.0 FSI, A2 1.4
Puissance		Puissance en chevaux Din	[55, 507]
Longueur		Catégorie de longueur	courte, moyenne, longue, très longue
NbPlaces	numérique	Nombre de places	[5, 7]
NbPortes	numérique	Nombre de portes	[3, 5]
Couleur	catégoriel	Couleur	blanc, bleu, gris, noir, rouge
Occasion	booléen	Véhicule d'occasion ?	true, false
Prix	numérique	Prix de vente en euros	[7500, 101300]

Nous utilisons la commande SQL "DESCRIBE" pour afficher les noms et les types de colonnes de la table "table\_immatriculations\_ext" et vérifier s'ils sont conformes au dictionnaire de données :

0: jdbc:hive2://localhost:10000> DESC table\_ext\_immatriculations;

col_name	data_type	comment
immatriculation	string	
marque	string	
nom	string	
puissance	int	
longueur	string	
nbplaces	int	
nbportes	int	
couleur	string	
occasion	boolean	
prix	int	

Comme observé, la table Hive est conforme aux normes du dictionnaire. À présent, nous allons utiliser la fonction DISTINCT de MySQL pour examiner les plages de valeurs de chaque colonne de la table « table\_immatriculations\_ext » par rapport au dictionnaire, en commençant par la colonne "marque" :

```
0: jdbc:hive2://localhost:10000> SELECT      DISTINCT      marque      from
table_ext_immatriculations;
```

marque
Audi
BMW
Dacia
Daihatsu
Fiat
Ford
Jaguar
Kia
Lancia
Mercedes
Mini
Nissan
Peugeot
Renault
Saab
Seat
Skoda
Volkswagen
Volvo
marque

```
0: jdbc:hive2://localhost:10000> SELECT COUNT(*) from table_ext_immatriculations
where marque="marque";
```

+-----+	
_c0	
+-----+	
1	
	+-----+

Comme vous l'avez peut-être remarqué, la valeur "marque" est présente parmi les valeurs de la colonne "marque", alors qu'elle n'est pas répertoriée dans les domaines de valeur du dictionnaire de données. Il serait judicieux de vérifier l'ensemble des valeurs de l'enregistrement où cette valeur apparaît afin d'avoir une meilleure compréhension au niveau de la base de données source HBase.

```
hbase:036:0> scan 'Immatriculations', {FILTER => "SingleColumnValueFilter('cf', 'marque', =, 'binary:marque')", LIMIT =>1}
ROW
      COLUMN+CELL
immatriculation          column=cf:couleur, timestamp=2024-04-01T18:30:59.881, value=couleur
immatriculation          column=cf:longueur, timestamp=2024-04-01T18:30:59.881, value=longueur
immatriculation          column=cf:marque, timestamp=2024-04-01T18:30:59.881, value=marque
immatriculation          column=cf:nbPlaces, timestamp=2024-04-01T18:30:59.881, value=nbPlaces
immatriculation          column=cf:nbPortes, timestamp=2024-04-01T18:30:59.881, value=nbPortes
immatriculation          column=cf:nom, timestamp=2024-04-01T18:30:59.881, value=nom
immatriculation          column=cf:occasion, timestamp=2024-04-01T18:30:59.881, value=occasion
immatriculation          column=cf:prix, timestamp=2024-04-01T18:30:59.881, value=prix
immatriculation          column=cf:puissance, timestamp=2024-04-01T18:30:59.881, value=puissance
1 row(s)
Took 18.8859 seconds
```

La commande ci-dessus exécute un balayage sur la table "Immatriculations" dans HBase, filtrant les résultats pour inclure uniquement les lignes où la valeur de la colonne "marque" correspond exactement à "marque". Cette commande limite également le nombre de résultats renvoyés à une seule ligne. Cela permet de vérifier rapidement si des en-têtes de colonnes non désirés, tels que "marque", ont été importés dans la table HBase.

Il est observé que dans la même ligne, les autres colonnes contiennent le nom de leurs colonnes respectives. On en déduit que lors de l'importation des données vers HBase avec la composante "ImportTsv", l'en-tête du fichier Excel a également été importé. Par conséquent, il est nécessaire de supprimer cette ligne directement depuis la base de données source HBase. Il convient de noter que la table "table\_ext\_immatriculations" est une table externe, ce qui signifie que les données pointent vers leurs base de données source qui est HBase.

```
hbase:043:0> deleteall 'Immatriculations', 'immatriculation'
Took 0.0156 seconds
hbase:044:0> get 'Immatriculations', 'immatriculation'
COLUMN
0 row(s)
Took 0.0167 seconds
hbase:045:0> ■
```

La commande « **deleteall 'Immatriculations','immatriculation'** » supprime toutes les données associées à la ligne identifiée par la clé 'immatriculation' dans la table 'Immatriculations' de HBase. Cette ligne est celle qui contient les en-têtes de colonnes importées depuis le fichier Excel. Cela efface complètement les données de cette ligne, ce qui peut être utile pour nettoyer les enregistrements incorrects ou obsolètes.

La commande « get 'Immatriculations','immatriculation' » récupère toutes les données associées à la ligne identifiée par la clé 'immatriculation' dans la table 'Immatriculations' de HBase. Elle permet de vérifier si la ligne a été correctement supprimée ou si des données subsistent après l'opération de suppression.

0: jdbc:hive2://localhost:10000> SELECT DISTINCT marque from table\_ext\_immatriculations;

marque
Audi
BMW
Dacia
Daihatsu
Fiat
Ford
Jaguar
Kia
Lancia
Mercedes
Mini
Nissan
Peugeot
Renault
Saab
Seat
Skoda
Volkswagen
Volvo

0: jdbc:hive2://localhost:10000> SELECT DISTINCT nom from table\_ext\_immatriculations;

nom
1007 1.4
120i
9.3 1.8T
A2 1.4
A200
A3 2.0 FSI
Almera 1.8
Coupe 1.6 16V
Croma 2.2
Coupe 1.0
Golf 2.0 FSI
Laguna 2.0T
Logan 1.6 MPI
MS
Maxima 3.0 V6
Megane 2.0 16V
Mondeo 1.8
New Beagle 1.8
Picanto 1.1
Polo 1.2 6V
Primera 1.6
S500
S80 T6
Superb 2.8 V6
Toledo 1.6
Vel Satis 3.5 V6
X-Type 2.5 V6
Ypsilon 1.4 16V

0: jdbc:hive2://localhost:10000> SELECT DISTINCT puissance from table\_ext\_immatriculations;

puissance
55
58
65
75
90
102
109
110
115
125
135
136
147
150
170
193
197
200
245
272
306
507

0: jdbc:hive2://localhost:10000> SELECT DISTINCT longueur from table\_ext\_immatriculations;

longueur
courte
longue
moyenne
tres longue

0: jdbc:hive2://localhost:10000> SELECT DISTINCT nbportes from table\_ext\_immatriculations;

nbportes
3
5

0: jdbc:hive2://localhost:10000> SELECT DISTINCT nbplaces from table\_ext\_immatriculations;

nbplaces
5

```
0: jdbc:hive2://localhost:10000> SELECT DISTINCT couleur from table_ext_immatriculations;
```

couleur
blanc
bleu
gris
noir
rouge

```
0: jdbc:hive2://localhost:10000> SELECT DISTINCT prix from table_ext_immatriculations;
```

prix  
7500  
8540  
8850  
8990  
9450  
9625  
12200  
12740  
12817  
13500  
13750  
15644  
16029  
16450  
16730  
17346  
18130  
18200  
18310  
18641  
18650  
18880  
19110  
19950  
22350  
22900  
23900  
24780  
25060  
25900  
25970  
26630  
27020  
27300  
28500  
30000  
31790  
34440  
35350  
37100  
38600  
49200  
50500  
66360  
70910  
94800  
101300

Vérifions que les immatriculations respecte le format « 9999 AA 99 » : Cette commande SQL ci-dessous sélectionne toutes les valeurs distinctes de la colonne "immatriculation" où les valeurs de cette colonne ne correspondent pas au motif spécifié par l'expression régulière. L'expression régulière `^[0-9]{1,4} [A-Z]{2} [0-9]{2}$` correspond à un format d'immatriculation standard, consistant en 1 à 4 chiffres suivis de 2 lettres majuscules puis de 2 chiffres.

```
+-----+  
| immatriculation |  
+-----+  
+-----+
```

Étant donné que le résultat renvoie un tableau vide, nous en déduisons qu'il n'existe aucune immatriculation ne respectant pas le format spécifié. Nous allons maintenant vérifier le critère

d'unicité pour la colonne "immatriculation", on utilise une requête SQL qui sélectionne la colonne ainsi que le nombre d'occurrences de chaque immatriculation. Les résultats sont regroupés par immatriculation à l'aide de la clause GROUP BY. La clause HAVING COUNT(\*) > 1 spécifie que seules les immatriculations ayant plus d'une occurrence seront incluses dans les résultats.

```
+-----+  
| immatriculation | nb_occurrences |  
+-----+  
|  
+-----+
```

Le résultat de la requête renvoie une colonne vide, confirmant ainsi qu'il n'y a aucune occurrence de valeurs dupliquées dans la colonne "immatriculation", ce qui confirme qu'elle respecte l'unicité.

Après avoir analysé l'ensemble des valeurs pour chaque colonne, nous pouvons conclure que la table Hive "table\_immatriculations\_ext" est conforme aux règles définies dans le dictionnaire de données.

## **5.10- Exploration des données marketing stockées dans la table Hive « table\_marketing »**

L'objectif est de vérifier si les données marketing stockées dans la table HIve respectent le dictionnaire de données fourni, comme illustré dans l'image ci-dessous :

### *Marketing.csv : clients sélectionnés par le service marketing*

Attribut	Type	Description	Domaine de valeurs
Age	numérique	Age en années du clients	[18, 84]
Sexe	catégoriel	Genre de la personne	M, F
Taux	numérique	Capacité d'endettement du client en euros (30% du salaire)	[544, 74185]
SituationFamiliale	catégoriel	Situation familiale du client	Célibataire, Divorcée, En Couple, Marié(e), Seul, Seule
NbEnfantsAcharge	numérique	Nombre d'enfants à charge	[0, 4]
Zeme voiture	booléen	Le client possède déjà un véhicule principal ?	true, false

```
0: jdbc:hive2://localhost:10000> DESC table_marketing ;
```

col_name	data_type	comment
age	int	
sexe	char(1)	
taux	int	
situationfamiliale	varchar(20)	
nbenfantsacharge	int	
deuxieme_voiture	string	

0: jdbc:hive2://localhost:10000> SELECT DISTINCT age from table\_marketing;

age
19
21
22
26
27
34
35
43
48
54
55
58
59
60
64
79
80

0: jdbc:hive2://localhost:10000> SELECT DISTINCT sexe from table\_marketing;

sexe
F
M

0: jdbc:hive2://localhost:10000> SELECT DISTINCT taux from table\_marketing;

taux
153
154
212
223
401
411
420
431
452
524
530
559
572
588
589
748
981
1112
1192
1396

0: jdbc:hive2://localhost:10000> SELECT DISTINCT situationfamiliale from table\_marketing;

situationfamiliale
Célibataire
En Couple

2 rows selected (1.517 second)

0: jdbc:hive2://localhost:10000> SELECT DISTINCT nbenfantsacharge from table\_marketing;

nbenfantsacharge
0
1
2
3

0: jdbc:hive2://localhost:10000> SELECT DISTINCT deuxieme\_voiture from table\_marketing;

24/04/29 23:25:08 INFO Txn.TxnHa
deuxieme_voiture
false
true

Nous pouvons conclure que la table Hive "table\_marketing" est conforme au dictionnaire de données.

## 5.11- Transformation et nettoyage des données clients de la table Hive « client\_7\_ext »

### 5.11.1- Vérification du respect du format du dictionnaire de donnée

*Clients\_N.csv<sup>1</sup> : fichier clients concernant les achats de l'année en cours*

Attribut	Type	Description	Domaine de valeurs
Age	numérique	Age en années du clients	[18, 84]
Sexe	catégoriel	Genre de la personne	M, F
Taux	numérique	Capacité d'endettement du client en euros (30% du salaire)	[544, 74185]
SituationFamiliale	catégoriel	Situation familiale du client	Célibataire, Divorcée, En Couple, Marié(e), Seul, Seule
NbEnfantsAcharge	numérique	Nombre d'enfants à charge	[0, 4]
2eme voiture	booléen	Le client possède déjà un véhicule principal ?	true, false
Immatriculation	caractères	Numéro unique d'immatriculation du véhicule	Texte au format « 9999 AA 99 »

0: jdbc:hive2://localhost:10000> DESCRIBE client\_7\_ext;

col_name	data_type	comment
age	int	from deserializer
sexe	string	from deserializer
taux	int	from deserializer
situationfamiliale	string	from deserializer
nbenfantsacharge	int	from deserializer
deuxiemevoiture	string	from deserializer
immatriculation	string	from deserializer

La table respecte effectivement les noms et les types définis dans le dictionnaire de données. La prochaine étape consiste à vérifier si elle respecte les domaines de valeurs spécifiées pour chaque colonne de la table "client\_7\_ext".

### 5.11.2- Calcul de l'estimation du nombre total de valeurs non définies dans la table.

0: jdbc:hive2://localhost:10000> **SELECT DISTINCT age FROM client\_7\_ext;**

age
NULL
-1
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40

0: jdbc:hive2://localhost:10000> **SELECT COUNT(\*) FROM client\_7\_ext WHERE age = -1 OR age IS NULL;**

_c0
320

En observant la colonne "age", on constate la présence de deux valeurs indéfinies, à savoir NULL et "-1", avec un total de 320 occurrences. La requête SQL permet d'identifier et de compter les enregistrements où l'âge n'est pas spécifié ou est marqué comme indéfini (-1).

0: jdbc:hive2://localhost:10000> **SELECT DISTINCT sexe FROM client\_7\_ext;**

sexe
?
F
Femme
Féminin
Homme
M
Masculin
N/D

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN sexe = " " THEN 1 ELSE 0 END) AS missing_values,  
SUM(CASE WHEN sexe = '?' THEN 1 ELSE 0 END) AS question_marks,  
SUM(CASE WHEN sexe = 'N/D' THEN 1 ELSE 0 END) AS nd_values  
FROM client_7_ext;
```

missing_values	question_marks	nd_values
107	93	108

Il est observé que dans la colonne "sexe", certaines valeurs sont étiquetées comme "N/D", d'autres comme "?", et il y en a aussi qui sont vides. La requête compte le nombre de valeurs manquantes (107), le nombre de valeurs contenant le symbole "?" (93), ainsi que le nombre de valeurs étiquetées comme "N/D" (108) dans la colonne "sexe" de la table "client\_7\_ext". Chaque valeur est calculée séparément en utilisant des fonctions de somme conditionnelles.

0: jdbc:hive2://localhost:10000> SELECT DISTINCT taux FROM client\_7\_ext;

taux
NULL
-1
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168

0: jdbc:hive2://localhost:10000> **SELECT COUNT(\*) FROM client\_7\_ext WHERE taux is NULL and taux= -1;**

_c0
319

Il est observé qu'il existe deux valeurs indéfinies pour la colonne "taux", qui sont NULL et "-1". La requête permet de compter le nombre d'occurrences où la colonne "taux" est à la fois NULL et égale à « -1 » dans la table "client\_7\_ext" et retourne un total de 319 occurrences.

0: jdbc:hive2://localhost:10000> **SELECT DISTINCT SituationFamiliale FROM client\_7\_ext;**

situationfamiliale
?
Célibataire
Divorcée
En Couple
Marié(e)
N/D
Seul
Seule

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN SituationFamiliale = '' THEN 1 ELSE 0 END) AS missing_values,  
SUM(CASE WHEN SituationFamiliale = '?' THEN 1 ELSE 0 END) AS question_marks,  
SUM(CASE WHEN SituationFamiliale = 'N/D' THEN 1 ELSE 0 END) AS nd_values  
FROM client_7_ext;
```

missing_values	question_marks	nd_values
101	95	105

On remarque que dans la colonne "situationfamiliale", différentes valeurs sont attribuées : certaines sont marquées comme "N/D", d'autres comme "?", et il y en a également qui sont vides. Cette requête précédente calcule le nombre de valeurs manquantes (101), le nombre de valeurs contenant le symbole "?" (95), ainsi que le nombre de valeurs étiquetées comme "N/D" (105) dans la colonne. Chaque catégorie de valeurs est comptée individuellement à l'aide de fonctions de somme conditionnelles.

0: jdbc:hive2://localhost:10000> SELECT DISTINCT NbEnfantsAcharge FROM client\_7\_ext;

nbenfantsacharge
NULL
-1
0
1
2
3
4

0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) FROM client\_7\_ext WHERE NbEnfantsAcharge = -1 OR NbEnfantsAcharge IS NULL;

24/03/30 13:30:42 INFO txn.TxnHandler: R
+-----+
_c0
+-----+
318
+-----+

Il a été remarqué qu'il y a deux valeurs indéfinies dans la colonne "nbenfantsacharge", étant respectivement NULL et "-1". Cette même requête calcule le nombre d'occurrences des enregistrements possédant ces valeurs, qui est égal à 318.

```
0: jdbc:hive2://localhost:10000> SELECT DISTINCT deuxiemevoiture FROM client_7_ext;
```

deuxiemevoiture
?
false
true

```
0: jdbc:hive2://localhost:10000> SELECT  
SUM(CASE WHEN deuxiemevoiture = '' THEN 1 ELSE 0 END) AS missing_values,  
SUM(CASE WHEN deuxiemevoiture = '?' THEN 1 ELSE 0 END) AS question_marks  
FROM client_7_ext;
```

missing_values	question_marks
131	94

Il a été remarqué qu'il existe deux valeurs indéfinies, qui sont le vide et "?" dans la colonne "deuxiemevoiture". En utilisant la même méthode, la requête précédente calcule le nombre d'occurrences des valeurs vides (131) et des valeurs "?" (94) en utilisant des fonctions de somme conditionnelle.

Il est estimé qu'il existe au maximum 1791 enregistrements dans la table "client\_ext\_7" contenant des valeurs indéfinies. Cependant, étant donné qu'il y a un total de 100 000 enregistrements, ces valeurs indéfinies n'ont pas un impact significatif dans le contexte de l'analyse de données. Cependant, des ajustements seront réalisés pour traiter les valeurs non conformes au dictionnaire de données de cette table.

### 5. 11.3- Traitement des valeurs de la colonne « sexe »

Comme confirmé précédemment, la colonne "sexe" de la table Hive «client\_7\_ext» contient des valeurs vides, des "?" et des "N/D" qui ne respectent pas le domaine de valeurs de cette colonne

0: jdbc:hive2://localhost:10000> SELECT DISTINCT sexe from client\_7\_ext;

sexe
?
F
Femme
Féminin
Homme
M
Masculin
N/D

Puisque la table "client\_7\_ext" est une table externe, ses données sont liées à celles extraites des sources, en l'occurrence la collection "Client\_7" de la base de données "Clients" de MongoDB. Par conséquent, effectuer des manipulations directement sur la table Hive n'est pas possible, et il est préférable d'appliquer le traitement directement sur la base de données MongoDB.

```
> db.Clients_7.aggregate([
...   {
...     $group: {
...       _id: null,
...       missing_values: { $sum: { $cond: [{ $eq: ["$sexe", ""], 1, 0} ] } },
...       question_marks: { $sum: { $cond: [{ $eq: ["$sexe", "?"], 1, 0} ] } },
...       nd_values: { $sum: { $cond: [{ $eq: ["$sexe", "N/D"], 1, 0} ] } }
...     }
...   }
... ]);
{ "_id" : null, "missing_values" : 107, "question_marks" : 93, "nd_values" : 108 }
>
>
> db.Clients_7.distinct("sexe");
[ "F", "M", "", "Masculin", "Féminin", "?", "Homme", "Femme", "N/D" ]
>
```

Ce code MongoDB effectue une agrégation sur la collection "Clients\_7", regroupant les documents selon la colonne "sexe" pour calculer le nombre de valeurs manquantes, de "?", et de "N/D". L'argument « \_id: null » signifie qu'on ne souhaite pas effectuer de regroupement selon une clé spécifique, mais plutôt regrouper tous les documents ensemble. Les résultats de cette agrégation sont cohérents avec ceux trouvés dans la colonne correspondante de la table Hive «client\_7\_ext». De plus, la commande db.Clients\_7.distinct("sexe") est employée pour afficher toutes les valeurs distinctes présentes dans la colonne "sexe".

```
> db.Clients_7.updateMany(  
...   { $or: [ { sexe: "" }, { sexe: "?" }, { sexe: "N/D" } ] },  
...   { $set: { sexe: null } }  
... );  
{ "acknowledged" : true, "matchedCount" : 308, "modifiedCount" : 308 }  
>
```

Cette commande met à jour plusieurs documents dans la collection "Clients\_7" en remplaçant les valeurs vides, les "?" et les "N/D" de la colonne "sexe" par des valeurs nulles. Plus précisément, elle cible les documents où la colonne "sexe" a une de ces trois valeurs spécifiées en utilisant l'opérateur \$or. Ensuite, elle utilise l'opérateur \$set pour définir la valeur de la colonne à null pour chaque document correspondant. Les résultats de cette opération confirment que 308 documents ont été découverts et modifiés avec succès dans la collection, correspondant au nombre de valeurs indéfinies entre "?" et "N/D" identifiées précédemment.

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN sexe = ' ' THEN 1 ELSE 0 END) AS missing_values,
```

```
SUM(CASE WHEN sexe = '?' THEN 1 ELSE 0 END) AS question_marks,
```

```
SUM(CASE WHEN sexe = 'N/D' THEN 1 ELSE 0 END) AS nd_values
```

FROM client\_7\_ext;

missing_values	question_marks	nd_values
0	0	0

0: jdbc:hive2://localhost:10000> SELECT DISTINCT sexe FROM client\_7\_ext;

sexe
NULL
F
Femme
Féminin
Homme
M
Masculin

Comme on peut le constater, les données de la colonne "sexe" dans la table Hive "client\_7\_ext" ont été modifiées avec succès, remplaçant toutes les valeurs indéfinies par NULL. Cette méthode se révélera précieuse pour l'analyses de données envisagées dans le cadre du projet.

Maintenant que les valeurs indéfinies ont été traitées, l'étape suivante consiste à convertir les valeurs de la colonne vers le domaine de valeurs spécifié dans le dictionnaire, qui comprend "F" pour féminin et "M" pour masculin.

```
> db.Clients_7.distinct("sexe");
[ "F", "M", null, "Masculin", "Féminin", "Homme", "Femme" ]
```

```
> db.Clients_7.updateMany(
...   { sexe: { $in: ["Femme", "Féminin"] } },
...   { $set: { sexe: "F" } }
... );
{ "acknowledged" : true, "matchedCount" : 615, "modifiedCount" : 615 }
> db.Clients_7.updateMany(
...   { sexe: { $in: ["Homme", "Masculin"] } },
...   { $set: { sexe: "M" } }
... );
{ "acknowledged" : true, "matchedCount" : 1352, "modifiedCount" : 1352 }
> db.Clients_7.distinct("sexe");
[ "F", "M", null ]
```

La première requête utilise updateMany() pour mettre à jour les documents où la valeur de la colonne "sexe" est soit "Femme" soit "Féminin". La seconde requête met à jour les documents où la valeur de la colonne est soit "Homme" soit "Masculin". Ces valeurs sont remplacées respectivement par « F » et « M » à l'aide de l'opérateur \$set.

```
0: jdbc:hive2://localhost:10000> SELECT DISTINCT sexe FROM client_7_ext;
```

sexe
NULL
F
M

Suite au résultat de la dernière requête, on peut conclure que la colonne "sexe" ne contient désormais que les valeurs "F" et "M", conformément au dictionnaire de données spécifié.

#### 5. 11.4- Traitement des valeurs de la colonne « situationFamiliale »

Les champs "situationfamiliale" dans la table Hive "client\_7\_ext" et "situationFamiliale" dans la collection "Clients\_7" de MongoDB contiennent des entrées vides, des "?" et des "N/D" qui ne sont pas conformes au domaine de valeurs attendu pour ces champs.

0: jdbc:hive2://localhost:10000> **SELECT DISTINCT situationfamiliale FROM client\_7\_ext;**

situationfamiliale
?
Célibataire
Divorcée
En Couple
Marié(e)
N/D
Seul
Seule

```
> db.Clients_7.distinct("situationFamiliale");
[{"situationFamiliale": "", "id": "5d5a2a00e4b03f3753a2a000"}, {"situationFamiliale": "?", "id": "5d5a2a00e4b03f3753a2a001"}, {"situationFamiliale": "N/D", "id": "5d5a2a00e4b03f3753a2a002"}, {"situationFamiliale": "Seul", "id": "5d5a2a00e4b03f3753a2a003"}, {"situationFamiliale": "Seule", "id": "5d5a2a00e4b03f3753a2a004"}, {"situationFamiliale": "Marié(e)", "id": "5d5a2a00e4b03f3753a2a005"}, {"situationFamiliale": "Divorcée", "id": "5d5a2a00e4b03f3753a2a006"}, {"situationFamiliale": "Célibataire", "id": "5d5a2a00e4b03f3753a2a007"}, {"situationFamiliale": "En Couple", "id": "5d5a2a00e4b03f3753a2a008"}]
```

Ce code MongoDB ci-dessous met à jour les documents dans la collection Clients\_7, remplaçant les valeurs vides, les "?" et les "N/D" du champ "situationFamiliale" par null, ce qui est confirmé par la commande « distinct » montrant la liste des valeurs distinctes après la mise à jour, incluant désormais null parmi les valeurs valides.

```
> db.Clients_7.updateMany(  
...   { $or: [ { situationFamiliale: "" }, { situationFamiliale: "?" }, { situationFamiliale: "N/D" } ] },  
...   { $set: { situationFamiliale: null } }  
... );  
{ "acknowledged" : true, "matchedCount" : 301, "modifiedCount" : 301 }  
> db.Clients_7.distinct("situationFamiliale");  
[  
  "En Couple",  
  "Marié(e)",  
  "Célibataire",  
  "Seule",  
  null,  
  "Seul",  
  "Divorcée"  
]  
>
```

301 documents ont été identifiés selon les critères de recherche pour remplacer les valeurs " ? " et " N/D " par null, ce qui correspond à l'estimation antérieure des valeurs indéfinies pour la colonne "situationFamiliale" dans la table Hive.

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN SituationFamiliale = "" THEN 1 ELSE 0 END) AS missing_values,  
SUM(CASE WHEN SituationFamiliale = '?' THEN 1 ELSE 0 END) AS question_marks,  
SUM(CASE WHEN SituationFamiliale = 'N/D' THEN 1 ELSE 0 END) AS nd_values
```

FROM client\_7\_ext;

missing_values	question_marks	nd_values
0	0	0

0: jdbc:hive2://localhost:10000> SELECT DISTINCT situationfamiliale FROM client\_7\_ext;

situationfamiliale
NULL
Célibataire
Divorcée
En Couple
Marié(e)
Seul
Seule

Nous pouvons déduire que les valeurs indéfinies ont été effectivement remplacées par la valeur NULL lors de la vérification de la colonne correspondante dans la table Hive. Bien que ces valeurs soient valides par rapport au dictionnaire de données, d'un point de vue sémantique, il serait pertinent de les affiner davantage en les transformant en deux valeurs distinctes: "Célibataire" et "En Couple", les rendant ainsi plus significatives.

```
> db.Clients_7.distinct("situationFamiliale");
[  
    "En Couple",  
    "Marié(e)",  
    "Célibataire",  
    "Seule",  
    null,  
    "Seul",  
    "Divorcée"  
]  
> db.Clients_7.updateMany(  
...    { situationFamiliale: { $in: ["Seul", "Seule", "Divorcée"] } },  
...    { $set: { situationFamiliale: "Célibataire" } }  
... );  
{ "acknowledged" : true, "matchedCount" : 5283, "modifiedCount" : 5283 }  
> db.Clients_7.updateMany(  
...    { situationFamiliale: "Marié(e)" },  
...    { $set: { situationFamiliale: "En Couple" } }  
... );  
{ "acknowledged" : true, "matchedCount" : 645, "modifiedCount" : 645 }  
> db.Clients_7.distinct("situationFamiliale");  
[ "En Couple", "Célibataire", null ]  
>
```

Ces requêtes MongoDB mettent à jour les documents dans la collection « Clients\_7 », remplaçant les valeurs correspondantes par "Célibataire" pour "Seul", "Seule", et par "Divorcée" pour "En Couple" et "Marié(e)", résultant en une liste distincte de valeurs "En Couple", "Célibataire" et null.

0: jdbc:hive2://localhost:10000> **SELECT DISTINCT situationfamiliale FROM client\_7\_ext;**

situationfamiliale
NULL
Célibataire
En Couple

Effectivement, les données ont été modifiées avec succès dans la colonne "situationfamiliale" dans Hive, ce qui a permis de raffiner les domaines de valeurs pour une meilleure analyse des données dans les étapes ultérieures du projet.

### 5. 11.5- Traitement des valeurs de la colonne « NbEnfantsAcharge »

Comme observé précédemment, deux valeurs aberrantes (-1 et NULL) sont présentes dans la colonne "nbenfantsacharge" de la table Hive. Nous prévoyons de convertir la valeur -1 en NULL.

0: jdbc:hive2://localhost:10000> **SELECT DISTINCT nbenfantsacharge FROM client\_7\_ext;**

nbenfantsacharge
NULL
-1
0
1
2
3
4

Les opérations MongoDB suivantes effectuées sur la collection "Clients\_7" ont permis de traiter les valeurs anormales dans la colonne "nbEnfantsAcharge". Initialement, la liste des valeurs uniques comprenait 1, 2, 3, 4, 0, "", "?", et -1. Cependant, les valeurs vides et "?" ne correspondaient pas au domaine de valeurs prévu pour les colonnes correspondantes dans la table Hive. Ceci est dû au type de colonne numérique qui convertit automatiquement ces deux valeurs en NULL.

Suite à l'identification de 318 documents contenant des valeurs vides, -1 ou "?", ces valeurs ont été remplacées par null. Cette action a été validée par la modification de 318 documents, ce qui a conduit à une nouvelle liste de valeurs distinctes incluant 1, 2, 3, 4, 0, et null.

```
> db.Clients_7.distinct("nbEnfantsAcharge");
[ 1, 3, 2, 4, 0, "", "?", -1 ]
> db.Clients_7.find({ nbEnfantsAcharge: { $in: ["", -1, "?"] } }).count();
318
> db.Clients_7.updateMany(
...   { nbEnfantsAcharge: { $in: ["", -1, "?"] } },
...   { $set: { nbEnfantsAcharge: null } }
... );
{ "acknowledged" : true, "matchedCount" : 318, "modifiedCount" : 318 }
> db.Clients_7.distinct("nbEnfantsAcharge");
[ 1, 3, 2, 4, 0, null ]
```

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN NbEnfantsAcharge = '' THEN 1 ELSE 0 END) AS missing_values,
SUM(CASE WHEN NbEnfantsAcharge = '?' THEN 1 ELSE 0 END) AS question_marks,
SUM(CASE WHEN NbEnfantsAcharge= -1 THEN 1 ELSE 0 END) AS minus1_values
FROM client_7_ext;
```

missing_values	question_marks	minus1_values
0	0	0

0: jdbc:hive2://localhost:10000> SELECT DISTINCT NbEnfantsAcharge FROM client\_7\_ext;

nbenfantsacharge
NULL
0
1
2
3
4

Les deux derniers résultats montrent que les modifications apportées à la colonne "nbenfantsacharge" de la table Hive "client\_7\_ext" ont été réalisées avec succès.

### 5. 11.6- Traitement des valeurs de la colonne « deuxiemevoiture »

0: jdbc:hive2://localhost:10000> **SELECT DISTINCT deuxiemevoiture FROM client\_7\_ext;**

deuxiemevoiture
?
false
true

Étant donné que la colonne "deuxiemevoiture" de la table Hive contient des valeurs vides et des "?", il est impératif d'effectuer le traitement approprié pour assurer la conformité avec le dictionnaire de données.

```
> db.Clients_7.distinct("2eme voiture");
[ "false", "true", "?", "" ]
> db.Clients_7.find({ "2eme voiture": { $in: ["", "?"] } }).count();
225
> db.Clients_7.updateMany(
...   { "2eme voiture": { $in: ["", "?"] } },
...   { $set: { "2eme voiture": null } }
... );
{ "acknowledged" : true, "matchedCount" : 225, "modifiedCount" : 225 }
> db.Clients_7.distinct("2eme voiture");
[ "false", "true", null ]
```

Initialement, une requête a été effectuée pour trouver le nombre de documents où le champ "2eme voiture" est vide ou contient "?", révélant un total de 225 documents. Ensuite, une mise à jour a été exécutée sur ces documents, remplaçant les valeurs vides et "?" par null, ce qui a été confirmé par la modification réussie de 225 documents. Enfin, une commande distinct a été utilisée pour obtenir les valeurs distinctes dans le champ "2eme voiture" après la mise à jour, montrant les valeurs "false", "true" et null.

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN deuxiemevoiture = '' THEN 1 ELSE 0 END) AS missing_values,  
SUM(CASE WHEN deuxiemevoiture = '?' THEN 1 ELSE 0 END) AS question_marks  
FROM client_7_ext;
```

missing_values	question_marks
0	0

0: jdbc:hive2://localhost:10000> SELECT DISTINCT deuxiemevoiture FROM client\_7\_ext;

deuxiemevoiture
NULL
false
true

Nous pouvons observer que les modifications ont été appliquées avec succès en examinant les plages de valeurs de la colonne "deuxiemevoiture" dans la table Hive. Celles-ci sont maintenant restreintes à "true" ou "false", conformément aux attentes du dictionnaire de données pour cette colonne.

### 5. 11.7- Traitement des valeurs de la colonne « age »

Comme observé précédemment, deux valeurs indéfinies, "-1" et NULL, sont présentes dans la colonne "age" de la table "client\_7\_ext".

0: jdbc:hive2://localhost:10000> SELECT DISTINCT age FROM client\_7\_ext;

age
NULL
-1
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

Avant de traiter ces valeurs, il est crucial de noter que la colonne "age" est de type numérique, ce qui nécessite une vérification supplémentaire pour s'assurer qu'il n'y a pas d'autres valeurs indéfinies dans la collection "Clients\_7", comme nous l'avons observé lors du traitement de la colonne "nbEnfantsAcharge".

> db.Client\_7\_distinct("age");

56,
43,
50,
27,
"",
70,
80,
25,
51,
26,
30,
68,
45,
69,
83,
29,
38,
28,
72,
49,
21,
"?",
39,
46,
61,
78,
42,
32,
20,
75,
47,
35,
65,
60,
71,
34,
57,
84,
82,
76,
-1

En effet, en examinant la colonne "age" de la collection "Clients\_7", nous constatons la présence de valeurs vides et de "?" en plus des valeurs indéfinies découvertes précédemment, qui ont été converties en NULL lors de l'extraction des données vers la table Hive "client\_7\_ext". Il est maintenant nécessaire de traiter ces valeurs en les transformant en NULL dans la base de données MongoDB, tout comme cela a été fait pour les autres colonnes.

```
> db.Clients_7.find({ "age": { $in: ["", "?", -1] } }).count();
320
>
> db.Clients_7.updateMany(
...   { "age": { $in: ["", "?", -1] } },
...   { $set: { "age": null } }
... );
{ "acknowledged" : true, "matchedCount" : 320, "modifiedCount" : 320 }
>
```

Dans cette série d'opérations MongoDB, une requête a d'abord été exécutée pour trouver le nombre de documents où le champ "age" était soit vide, contenait "?", ou était égal à -1, révélant un total de 320 documents. Ensuite, une mise à jour a été effectuée sur ces documents, remplaçant les valeurs vides, "?", et -1 par null. Cette opération a été validée par la modification réussie de 320 documents.

```
> db.Clients_7.distinct("age");
[ 57,
  24,
  73,
  52,
  99,
  63,
  79,
  44,
  81,
  64,
  58,
  22,
  33,
  36,
  40,
  77,
  62,
  67,
  74,
  37,
  41,
  31,
  23,
  48,
  55,
  53,
  54,
  18,
  66,
  80,
  19,
  56,
  43,
  27,
  null,
  70,
  25,
  51,
  26,
  30,
  68,
  45,
  50,
  69,
  83,
  29,
  38,
  28,
  72,
  49,
  46,
  21,
  39,
  61,
  78,
  42,
  32,
  20,
  75,
  47,
  35,
  65,
  60,
  71,
  34,
  57,
  84,
  82,
  76 ]
```

En effet, les valeurs en dehors des plages attendues ont été converties en null dans la colonne "age" de MongoDB. À présent, nous vérifions que ces mêmes transformations ont été appliquées à la colonne correspondante de la table Hive.

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN Age = '' THEN 1 ELSE 0 END) AS missing_values,  
SUM(CASE WHEN Age = '?' THEN 1 ELSE 0 END) AS question_marks,  
SUM(CASE WHEN Age = -1 THEN 1 ELSE 0 END) AS minus1_values  
FROM client_7_ext;
```

missing_values	question_marks	minus1_values
0	0	0

0: jdbc:hive2://localhost:10000> SELECT DISTINCT age FROM client\_7\_ext;

age
NULL
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84

Ainsi, nous constatons que les valeurs indéfinies sont en effet remplacées par NULL, et que les valeurs respectent désormais l'intervalle d'âges spécifié dans le dictionnaire de données.

#### 5. 11.8- Traitement des valeurs de la colonne « taux »

Comme déjà remarqué, la colonne "age" de la table "client\_7\_ext" contient deux valeurs indéfinies, "-1" et NULL.

taux
NULL
-1
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168

Puisque la colonne « taux » de la table Hive est numérique, il est important de vérifier la présence éventuelle de valeurs spéciales dans la colonne "taux" de la collection "Clients\_7" autre que « -1 » et null,tout comme cela a été fait pour les colonnes "age" et "nbenfantsacharge".

> db.Clients\_7.distinct("taux");

559, 225, '', 164, 1216	474, 549, -1, 407,
1071, ?", 877,	

Effectivement, en plus des valeurs "-1" et null, on trouve également d'autres valeurs vides et "?" dans la colonne "age" de la collection "Clients\_7". Nous allons les traiter de la même manière que les autres colonnes numériques précédentes.

```
> db.Clients_7.find({ taux: { $in: ['', -1, '?'] } }).count();
319
> db.Clients_7.updateMany(
...   { taux: { $in: ['', -1, '?'] } },
...   { $set: { taux: null } }
... );
{ "acknowledged" : true, "matchedCount" : 319, "modifiedCount" : 319 }
```

Une requête a été lancée pour déterminer le nombre de documents où le champ "taux" est vide, égal à -1, ou contient "?", révélant un total de 319 documents. Ensuite, une mise à jour a été appliquée sur ces documents, remplaçant les valeurs vides, -1 et "?", par null. La modification a été appliquée avec succès à 319 documents, comme confirmé par la réponse de la base de données MongoDB.

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN taux = '' THEN 1 ELSE 0 END) AS missing_values,  
SUM(CASE WHEN taux = '?' THEN 1 ELSE 0 END) AS question_marks,  
SUM(CASE WHEN taux = -1 THEN 1 ELSE 0 END) AS minus1_values  
  
FROM client_7_ext;
```

missing_values	question_marks	minus1_values
0	0	0

```
0: jdbc:hive2://localhost:10000> SELECT DISTINCT taux FROM client_7_ext;
```

taux
NULL
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174

Les modifications ont été appliquées avec succès à la colonne "taux" de la table "client\_7\_ext", ce qui confirme la présence de valeurs NULL et de plages de taux valides conformes au dictionnaire des données.

## 5. 11.9- Traitement des valeurs de la colonne « Immatriculation »

### 5. 11.9.1 – Vérification du format des immatriculations

En se basant sur le dictionnaire de données, les immatriculations doivent suivre le format "9999 AA 99". Vérifions si cette règle est respectée :

```
0: jdbc:hive2://localhost:10000>SELECT DISTINCT immatriculation  
  
FROM client_7_ext
```

```
WHERE immatriculation NOT REGEXP '^[0-9]{1,4} [A-Z]{2} [0-9]{2}$';
```

immatriculation

Cette requête Hive sélectionne les valeurs distinctes de la colonne "immatriculation" de la table "client\_7\_ext" où les immatriculations ne correspondent pas au format spécifié par l'expression régulière '`^[0-9]{1,4} [A-Z]{2} [0-9]{2}$`'. Cette expression régulière exige que les immatriculations commencent par un nombre de 1 à 4 chiffres, suivis de deux lettres majuscules, puis de deux chiffres. Les valeurs qui ne respectent pas ce format sont sélectionnées. Comme le résultat renvoie une table vide, cela indique que toutes les immatriculations de la table respectent le format exigé.

### 5. 11.9. 2– Vérification du principe d'unicité des immatriculations

Il est illogique d'avoir plusieurs occurrences d'une même immatriculation, car chaque voiture est censée posséder sa propre immatriculation qui la définit et la rend unique.

```
0: jdbc:hive2://localhost:10000> SELECT immatriculation, COUNT(*) AS nb_occurrences  
FROM client_7_ext  
GROUP BY immatriculation  
HAVING COUNT(*) > 1;
```

immatriculation	nb_occurrences
1360 RL 35	2
2735 HR 51	2
3923 NO 19	2
4290 BC 14	2
593 EF 70	2
7277 XA 78	2
8069 XB 17	2
9522 LK 47	2
9890 VL 16	2

Cette requête Hive sélectionne les immatriculations de la table "client\_7\_ext" ainsi que le nombre d'occurrences de chaque immatriculation. Ensuite, elle regroupe les résultats par immatriculation et filtre les résultats pour ne conserver que les immatriculations ayant plus d'une occurrence, c'est-à-dire les immatriculations qui se répètent. Nous constatons qu'il y a 9 occurrences d'une même immatriculation, ce qui est inacceptable.

**0: jdbc:hive2://localhost:10000> SELECT \* FROM client\_7\_ext WHERE immatriculation IN (SELECT immatriculation FROM client\_7\_ext GROUP BY immatriculation HAVING COUNT(\*) > 1);**

client_7_ext.age	client_7_ext sexe	client_7_ext.taux	client_7_ext.situationfamiliale	client_7_ext.nbenfantscharge	client_7_ext.deuxiemevoiture	client_7_ext.immatriculation
24	F	199	En Couple	0	false	1360 RL 35
18	F	234	Célibataire	0	false	1360 RL 35
66	M	543	Célibataire	0	false	2735 HR 51
36	M	911	En Couple	1	true	2735 HR 51
36	M	737	En Couple	1	false	3923 NO 19
39	M	1150	En Couple	1	false	3923 NO 19
48	F	1236	En Couple	3	true	4290 BC 14
53	F	987	Célibataire	0	false	4290 BC 14
45	M	447	Célibataire	0	false	593 EF 70
53	M	999	Célibataire	0	false	593 EF 70
50	M	463	En Couple	0	false	7277 XA 78
68	F	549	Célibataire	0	false	7277 XA 78
70	M	503	En Couple	4	true	8069 XB 17
50	M	575	Célibataire	0	false	8069 XB 17
27	M	424	En Couple	1	true	9522 LK 47
78	M	241	Célibataire	0	false	9522 LK 47
30	M	504	Célibataire	0	false	9890 VL 16
49	M	492	Célibataire	0	false	9890 VL 16

Cette requête SQL sélectionne toutes les lignes de la table "client\_7\_ext" où l'immatriculation de chaque voiture est présente dans le sous-ensemble des immatriculations qui apparaissent plus d'une fois dans la même table. En d'autres termes, elle récupère toutes les lignes correspondant aux voitures ayant des immatriculations qui se répètent au moins une fois dans la table.

Il est notable que les neuf paires d'immatriculations des voitures des clients sont principalement différentes les unes des autres, ce qui rend difficile l'identification d'une solution générale pour résoudre cette ambiguïté d'unicité. Une solution envisageable et simple serait de supprimer une occurrence et de conserver une seule (étant donné que chaque immatriculation est répétée deux fois).

**0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) FROM Client\_7\_ext**

+	-	-	-	-	-	-
	_c0					
	100000					
+	-	-	-	-	-	-

Avant de supprimer les occurrences, la table contient 100 000 enregistrements. Logiquement, il y a 9 paires d'immatriculations qui se répètent. Normalement, si on supprime une seule occurrence de chaque paire, à la fin de l'opération de suppression, on aura 99 991 enregistrements dans la table.

```
> db.Clients_7.aggregate([
...     { $group: { _id: "$immatriculation", count: { $sum: 1 } } },
...     { $match: { count: { $gt: 1 } } }
... ]);
{ "_id" : "9522 LK 47", "count" : 2 }
{ "_id" : "3923 NO 19", "count" : 2 }
{ "_id" : "1360 RL 35", "count" : 2 }
{ "_id" : "593 EF 70", "count" : 2 }
{ "_id" : "2735 HR 51", "count" : 2 }
{ "_id" : "8069 XB 17", "count" : 2 }
{ "_id" : "4290 BC 14", "count" : 2 }
{ "_id" : "7277 XA 78", "count" : 2 }
{ "_id" : "9890 VL 16", "count" : 2 }
>
> var immatriculationsRepetees = db.Clients_7.aggregate([
...     { $group: { _id: "$immatriculation", count: { $sum: 1 } } },
...     { $match: { count: { $gt: 1 } } }
... ]).toArray();
> immatriculationsRepetees.forEach(function(doc) {
...     var immatriculation = doc._id;
...     // Supprimer une seule occurrence de l'immatriculation répétée
...     db.Clients_7.deleteOne({ immatriculation: immatriculation });
... });
> db.Clients_7.count();
99991
```

Cette séquence MongoDB utilise l'agrégation pour identifier les immatriculations qui se répètent plus d'une fois dans la collection "Clients\_7". Premièrement, elle groupe les documents par immatriculation et compte le nombre d'occurrences pour chaque immatriculation. Ensuite, elle filtre les groupes pour ne conserver que ceux avec un compte supérieur à un, ce qui identifie les immatriculations répétées. Ensuite, le script récupère ces immatriculations répétées et les parcourt, supprimant une occurrence de chaque immatriculation répétée dans la collection. Après cette opération, elle affiche le nombre total de documents restants dans la collection, qui est de 99991.

**0: jdbc:hive2://localhost:10000> SELECT immatriculation, COUNT(\*) AS nb\_occurrences**

FROM client 7 ext

## GROUP BY immatriculation

HAVING COUNT(\*) > 1;

immatriculation	nb_occurrences
N	1
M	1
P	1
S	1
T	1
V	1
Z	1
Y	1
A	1
B	1
C	1
D	1
E	1
F	1
G	1
H	1
I	1
J	1
K	1
L	1
M	1
N	1
O	1
P	1
Q	1
R	1
S	1
T	1
U	1
V	1
W	1
X	1
Z	1

En exécutant cette requête une deuxième fois, elle nous renvoie un tableau vide, indiquant ainsi que toutes les immatriculations sont uniques et qu'il n'existe aucune occurrence en répétition.

### 5. 11.10 – Traitement des valeurs NULL de la table « client\_7\_ext »

Il est essentiel de vérifier que l'ajout de valeurs NULL ne diminue pas la qualité de l'information et qu'elles restent pertinentes pour maximiser la qualité des données et obtenir de bons résultats lors de l'étape du Data Mining.

0: jdbc:hive2://localhost:10000> SELECT \*FROM client\_7\_ext

WHERE immatriculation is NULL OR age is NULL OR sexe is NULL OR taux is NULL  
OR situationFamiliale is NULL OR nbEnfantsAcharge IS NULL OR deuxiemevoiture IS NULL;

25	F	237	En Couple	2	NULL	3699 HF 38
45		NULL	En Couple	4	false	1873 TJ 24
59	F	512	Célibataire	NULL	false	3451 CE 73
NULL	F	996	Célibataire	0	false	3509 VG 66
27	M	NULL	En Couple	2	false	7644 SE 95
NULL	M	414	En Couple	4	false	3464 SB 42
31	M	576	En Couple	NULL	false	5212 OK 49
42	M	NULL	Célibataire	0	false	893 YN 50
34	M	NULL	Célibataire	0	false	5099 BC 54
53	M	1854	En Couple	NULL	false	8546 DA 59
29	F	422	En Couple	4	NULL	4678 RE 12
48		592	En Couple	1	false	8662 GB 84
25		459	En Couple	4	false	9625 IO 19
76	M	414	En Couple	NULL	true	79 YX 19
75	M	493	En Couple	1	NULL	6654 JV 36
NULL	M	544	Célibataire	0	false	2685 OA 33
76	M	963	Célibataire	NULL	false	3352 DW 38
48	M	734	NULL	0	false	4652 AQ 31
37	F	488	Célibataire	0	NULL	566 LU 47
68	M	522	En Couple	2	false	8351 TE 77
27	M	167	En Couple	NULL	false	280 FR 84
18	M	NULL	Célibataire	0	false	7234 EF 55
41	M	583	Célibataire	NULL	false	9638 UT 57
29	F	NULL	En Couple	2	true	7369 FF 88
38	M	156	NULL	4	false	1807 VH 18
26	M	468	En Couple	1	NULL	6329 MH 52
NULL	M	492	Célibataire	3	false	5375 LO 21
47	F	176	NULL	0	true	9983 RJ 78
NULL	M	153	Célibataire	1	false	4891 RD 57
54	M	866	Célibataire	0	NULL	2301 MB 19

Cette requête Hive sélectionne toutes les colonnes de la table "client\_7\_ext" pour les enregistrements où au moins l'une des colonnes suivantes contient une valeur NULL : "immatriculation", "age", "sexe", "taux", "situationFamiliale", "nbEnfantsAcharge" ou "deuxiemevoiture".

D'abord, examinons tous les enregistrements qui contiennent au moins une valeur NULL dans une colonne. Il serait également intéressant de compter le nombre de champs NULL pour chaque enregistrement.

0: jdbc:hive2://localhost:10000> SELECT t.\*,

(CASE WHEN immatriculation IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN age IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN sexe IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN taux IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN situationFamiliale IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN nbEnfantsAcharge IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN deuxiemevoiture IS NULL THEN 1 ELSE 0 END) AS nb\_colonnes\_null

FROM client\_7\_ext t

WHERE immatriculation IS NULL OR age IS NULL OR sexe IS NULL OR taux IS NULL  
OR situationFamiliale IS NULL OR nbEnfantsAcharge IS NULL OR deuxiemevoiture IS  
NULL;

35	F	589	En Couple	2	NULL	7698 BX 28	1
37	M	1389	Célibataire	NULL	false	6723 YX 21	1
26	M	408	Célibataire	0	NULL	7592 CQ 34	1
NULL	M	414	En Couple	0	false	9625 CC 23	1
28	M	NULL	En Couple	2	false	4660 OE 55	1
49	M	580	En Couple	2	NULL	2666 XD 12	1
24	M	1377	NULL	2	false	8931 TV 67	1
29	M	1136	NULL	3	false	7523 RP 20	1
46	M	NULL	En Couple	1	false	8728 ZG 85	1
44	M	191	NULL	0	false	9834 FF 74	1
67	F	175	NULL	0	false	9422 LR 48	1
30	M	543	Célibataire	NULL	false	4871 OR 32	1
NULL	M	208	Célibataire	0	false	4254 WH 70	1
61	M	NULL	Célibataire	0	false	3555 YR 75	1
20	M	550	Célibataire	3	NULL	8439 RH 84	1
74	M	NULL	Célibataire	2	false	9462 EP 38	1
51	F	583	Célibataire	NULL	false	4736 QQ 94	1
29	M	NULL	En Couple	3	false	8646 JU 95	1
27	F	NULL	En Couple	3	false	837 BY 12	1
56	F	NULL	En Couple	4	false	8506 YS 27	1
57	M	554	En Couple	3	NULL	2288 BR 54	1
68	M	238	En Couple	NULL	false	2762 IR 96	1
72	M	563	Célibataire	NULL	false	1877 XU 37	1
25	F	237	En Couple	2	NULL	3699 HF 30	1
45	F	NULL	En Couple	4	false	1873 TJ 24	1
59	F	512	Célibataire	NULL	false	3451 CE 73	1
NULL	F	996	Célibataire	0	false	3509 VG 66	1
27	M	NULL	En Couple	2	false	7644 SE 95	1
NULL	M	414	En Couple	4	false	3464 SB 42	1
31	M	576	En Couple	NULL	false	5212 OK 49	1
42	M	NULL	Célibataire	0	false	893 YM 50	1
34	M	NULL	Célibataire	0	false	5099 BC 54	1
53	M	1854	En Couple	NULL	false	8546 DA 59	1
29	F	422	En Couple	4	NULL	4678 RE 12	1
48	NULL	592	En Couple	1	false	8662 GB 84	1
25	NULL	459	En Couple	4	false	9625 IO 19	1
76	M	414	En Couple	NULL	true	79 YX 19	1
75	M	493	En Couple	1	NULL	6654 JV 36	1
NULL	M	544	Célibataire	0	false	2685 OA 33	1
76	M	963	Célibataire	NULL	false	3352 DW 30	1
48	M	734	NULL	0	false	4652 AQ 31	1
37	F	488	Célibataire	0	NULL	566 LU 47	1
60	M	522	En Couple	2	NULL	8351 TE 27	1
27	M	167	En Couple	NULL	false	280 FB 84	1
18	M	NULL	Célibataire	0	false	7254 EF 55	1
41	M	583	Célibataire	NULL	false	9638 UT 57	1
20	F	NULL	En Couple	2	true	7369 FF 88	1
38	M	156	NULL	4	false	1807 VH 18	1
26	M	468	En Couple	1	NULL	6329 MH 52	1
NULL	M	492	Célibataire	3	false	5375 LO 21	1
47	F	176	NULL	0	true	9983 RJ 78	1
NULL	M	153	Célibataire	1	false	4891 RD 57	1
54	M	866	Célibataire	0	NULL	2301 MB 19	1

Cette requête Hive sélectionne toutes les colonnes de la table ainsi qu'une nouvelle colonne "nb\_colonnes\_null" qui compte le nombre de colonnes ayant une valeur NULL pour chaque enregistrement. Elle utilise des expressions CASE pour attribuer la valeur 1 si une colonne est NULL et 0 sinon, puis additionne ces valeurs pour obtenir le nombre total de colonnes NULL. Les enregistrements sélectionnés sont ceux où au moins une des colonnes spécifiées ("immatriculation", "age", "sexe", "taux", "situationFamiliale", "nbEnfantsAcharge" ou "deuxiemevoiture") contient une valeur NULL.

Il est observé que la plupart des enregistrements ont seulement un champ NULL. Il serait pertinent de filtrer les enregistrements qui ont au moins deux champs NULL.

```
db.Clients_7.aggregate([
  ...
  {
    $addFields: {
      nb_colonnes_null: {
        $sum: [
          { $cond: [{ $eq: ["$immatriculation", null] }, 1, 0] },
          { $cond: [{ $eq: ["$age", null] }, 1, 0] },
          { $cond: [{ $eq: ["$sexe", null] }, 1, 0] },
          { $cond: [{ $eq: ["$taux", null] }, 1, 0] },
          { $cond: [{ $eq: ["$situationFamiliale", null] }, 1, 0] },
          { $cond: [{ $eq: ["$nbEnfantsAcharge", null] }, 1, 0] },
          { $cond: [{ $eq: ["$2eme voiture", null] }, 1, 0] }
        ]
      }
    }
  },
  {
    $match: {
      $and: [
        { $or: [{ immatriculation: null }, { age: null }, { sexe: null }, { taux: null }, { situationFamiliale: null }, { nbEnfantsAcharge: null }, { "2eme voiture": null }] },
        { nb_colonnes_null: { $gte: 2 } }
      ]
    }
  }
])
```
... 11 documents
```
db.Clients_7.aggregate([
  ...
])
```

```

Cette agrégation MongoDB ajoute un champ supplémentaire "nb\_colonnes\_null" à chaque document, qui compte le nombre de colonnes ayant une valeur NULL. Ensuite, elle filtre les documents pour ne conserver que ceux où au moins deux des colonnes spécifiées ("immatriculation", "age", "sexe", "taux", "situationFamiliale", "nbEnfantsAcharge" ou "2eme voiture") ont une valeur NULL.

Il est remarqué que 11 lignes ont exactement deux colonnes NULL, et que parmi ces lignes, 10 ont des valeurs NULL dans les colonnes "sexe" et "age", or ces données sont importantes et significatifs. Par conséquent, il est conclu que ces 11 lignes ne contribuent pas significativement à notre analyse par rapport aux lignes qui ont seulement une colonne NULL. Il est donc envisagé de supprimer ces 11 lignes.

```
> var documentsASupprimer = db.Clients_7.aggregate([
  ...
  {
    $addFields: {
      nb_colonnes_null: {
        $sum: [
          { $cond: [{ $eq: ["$immatriculation", null] }, 1, 0] },
          { $cond: [{ $eq: ["$age", null] }, 1, 0] },
          { $cond: [{ $eq: ["$sexe", null] }, 1, 0] },
          { $cond: [{ $eq: ["$taux", null] }, 1, 0] },
          { $cond: [{ $eq: ["$situationFamiliale", null] }, 1, 0] },
          { $cond: [{ $eq: ["$nbEnfantsAcharge", null] }, 1, 0] },
          { $cond: [{ $eq: ["$2eme voiture", null] }, 1, 0] }
        ]
      }
    }
  },
  {
    $match: {
      $and: [
        { $or: [{ immatriculation: null }, { age: null }, { sexe: null }, { taux: null }, { situationFamiliale: null }, { nbEnfantsAcharge: null }, { "2eme voiture": null }] },
        { nb_colonnes_null: { $gte: 2 } }
      ]
    }
  }
]).toArray();
> documentsASupprimer.forEach(function(document) {
  ...
  db.Clients_7.deleteOne({ _id: document._id });
  ...
});
```

Ce code MongoDB commence par agréger les documents de la collection "Clients\_7" afin d'ajouter un champ supplémentaire "nb\_colonnes\_null", qui compte le nombre de colonnes avec des valeurs NULL pour chaque document. Ensuite, elle filtre les documents pour ne conserver que ceux qui ont au moins deux colonnes NULL. Les documents ainsi identifiés sont stockés

dans la variable "documentsASupprimer". Ensuite, chaque document dans cette variable est parcouru, et pour chaque document, la séquence supprime ce dernier dans la collection.

```
0: jdbc:hive2://localhost:10000> SELECT *
  FROM (
    SELECT t.*,
      (CASE WHEN immatriculation IS NULL THEN 1 ELSE 0 END) +
      (CASE WHEN age IS NULL THEN 1 ELSE 0 END) +
      (CASE WHEN sexe IS NULL THEN 1 ELSE 0 END) +
      (CASE WHEN taux IS NULL THEN 1 ELSE 0 END) +
      (CASE WHEN situationFamiliale IS NULL THEN 1 ELSE 0 END) +
      (CASE WHEN nbEnfantsAcharge IS NULL THEN 1 ELSE 0 END) +
      (CASE WHEN deuxiemevoiture IS NULL THEN 1 ELSE 0 END) AS
nb_colonnes_null
  FROM client_7_ext t
) t
WHERE (immatriculation IS NULL OR age IS NULL OR sexe IS NULL OR taux IS
NULL OR situationFamiliale IS NULL OR nbEnfantsAcharge IS NULL OR
deuxiemevoiture IS NULL) AND nb_colonnes_null >= 2;
```

```
+-----+-----+-----+-----+-----+-----+-----+
| t.age | t.sex | t.taux | t.situationfamiliale | t.nbenfantsacharge | t.deuxiemevoiture | t.immatriculation | t.nb_colonnes_null |
+-----+-----+-----+-----+-----+-----+-----+
No rows selected (1.54 seconds)
```

En exécutant cette requête une deuxième fois, il est observé qu'il n'y a plus aucun enregistrement dans la table Hive "client\_7\_ext" avec au moins deux champs NULL. À l'origine, il était envisageable de conserver les enregistrements contenant uniquement un champ NULL, car la perte d'information serait négligeable.

#### 5.11.11 – Nombre d'enregistrement de la table « client\_7\_ext » après nettoyage et transformation

D'après les traitements effectués précédemment, 9 enregistrements ont été supprimés en raison d'occurrences d'immatriculations en double, et 11 enregistrements ont été supprimés en raison de

la présence d'au moins deux champs NULL. Logiquement, il devrait rester 99980 enregistrements dans la table.

0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) from client\_7\_ext;

| _c0   |
|-------|
| 99980 |

En effet, le décompte est correct et la table Hive "client\_7\_ext" est conforme aux normes du dictionnaire de données.

## 5. 12-Transformation et nettoyage des données clients de la table Hive « client\_12\_ext »

L'approche utilisée et les problèmes rencontrés avec la table Hive "client\_12\_ext" sont très similaires à ceux de la table "client\_7\_ext". Afin d'éviter les répétitions, nous allons simplement vous présenter les commandes mise en place.

### 5. 12.1- Vérification du respect du format du dictionnaire de donnée

0: jdbc:hive2://localhost:10000> DESC client\_12\_ext;

| col_name           | data_type | comment           |
|--------------------|-----------|-------------------|
| age                | int       | from deserializer |
| sexe               | string    | from deserializer |
| taux               | int       | from deserializer |
| situationfamiliale | string    | from deserializer |
| nbenfantsacharge   | int       | from deserializer |
| deuxiemeviture     | string    | from deserializer |
| immatriculation    | string    | from deserializer |

La table est conforme aux noms et types spécifiés dans le dictionnaire de données. La prochaine étape consiste à vérifier si elle respecte les plages de valeurs spécifiées pour chaque colonne de la table "client\_12\_ext".

### 5. 12.2- Calcul de l'estimation du nombre total de valeurs non définies dans la table.

0: jdbc:hive2://localhost:10000> SELECT DISTINCT age FROM client\_12\_ext;

|      |
|------|
| NULL |
| -1   |
| 18   |
| 19   |
| 20   |
| 21   |
| 22   |
| 23   |
| 24   |
| 25   |
| 26   |
| 27   |
| 28   |
| 29   |
| 30   |
| 31   |
| 32   |
| 33   |
| 34   |
| 35   |
| 36   |
| 37   |
| 38   |

0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) FROM client\_12\_ext WHERE age = -1 OR age IS NULL;

| _c0 |
|-----|
| 295 |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT sexe FROM client\_12\_ext;

| sexe     |
|----------|
| ?        |
| F        |
| Femme    |
| Féminin  |
| Homme    |
| M        |
| Masculin |
| N/D      |

0: jdbc:hive2://localhost:10000> SELECT

SUM(CASE WHEN sexe = " " THEN 1 ELSE 0 END) AS missing\_values,

```
SUM(CASE WHEN sexe = '?' THEN 1 ELSE 0 END) AS question_marks,  
  
SUM(CASE WHEN sexe = 'N/D' THEN 1 ELSE 0 END) AS nd_values  
  
FROM client_12_ext;
```

| missing_values               | question_marks | nd_values |
|------------------------------|----------------|-----------|
| 90                           | 98             | 95        |
| row selected (2.823 seconds) |                |           |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT taux FROM client\_12\_ext;

| taux |
|------|
| NULL |
| -1   |
| 150  |
| 151  |
| 152  |
| 153  |
| 154  |
| 155  |
| 156  |

0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) FROM client\_12\_ext WHERE taux is NULL and taux= -1;

| _c0 |
|-----|
| 339 |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT SituationFamiliale FROM client\_12\_ext;

| situationfamiliale |
|--------------------|
| ?                  |
| Célibataire        |
| Divorcée           |
| En Couple          |
| Marié(e)           |
| N/D                |
| Seul               |
| Seule              |

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN SituationFamiliale = '' THEN 1 ELSE 0 END) AS missing_values,  
SUM(CASE WHEN SituationFamiliale = '?' THEN 1 ELSE 0 END) AS question_marks,  
SUM(CASE WHEN SituationFamiliale = 'N/D' THEN 1 ELSE 0 END) AS nd_values  
FROM client_12_ext;
```

| missing_values | question_marks | nd_values |
|----------------|----------------|-----------|
| 104            | 100            | 92        |

row selected (2.993 seconds)

0: jdbc:hive2://localhost:10000> SELECT DISTINCT NbEnfantsAcharge FROM client\_12\_ext;

| nbenfantsacharge |
|------------------|
| NULL             |
| -1               |
| 0                |
| 1                |
| 2                |
| 3                |
| 4                |

row selected (10.107 ms)

0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) FROM client\_12\_ext WHERE NbEnfantsAcharge = -1 OR NbEnfantsAcharge IS NULL;

| _c0 |
|-----|
| 286 |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT deuxiemevoiture FROM client\_12\_ext;

| deuxiemevoiture |
|-----------------|
| ?               |
| false           |
| true            |

0: jdbc:hive2://localhost:10000> SELECT  
 SUM(CASE WHEN deuxiemevoiture = '' THEN 1 ELSE 0 END) AS missing\_values,  
 SUM(CASE WHEN deuxiemevoiture = '?' THEN 1 ELSE 0 END) AS question\_marks  
 FROM client\_12\_ext;

| missing_values | question_marks |
|----------------|----------------|
| 93             | 104            |

Il est estimé qu'il y a au maximum 1696 enregistrements dans la table "client\_ext\_12" contenant des valeurs indéfinies. Toutefois, étant donné qu'il y a un total de 100 000 enregistrements, ces valeurs indéfinies n'ont pas un impact significatif dans le contexte de l'analyse de données. Néanmoins, les mêmes ajustements seront effectués pour traiter les valeurs non conformes au dictionnaire de données de cette table, tout comme dans l'approche adoptée pour la table Hive "client\_7\_ext".

### 5. 12.3- Traitement des valeurs de la colonne « sexe »

0: jdbc:hive2://localhost:10000> SELECT DISTINCT sexe from client\_12\_ext;

| sexe     |
|----------|
| ?        |
| F        |
| Femme    |
| Féminin  |
| Homme    |
| M        |
| Masculin |
| N/D      |

```
> db.Clients_12.distinct("sexe");
[ "M", "F", "Masculin", "Féminin", "Femme", "Homme", "N/D", "", "?" ]
```

```
> use Clients
switched to db Clients
> db.Clients_12.aggregate([
...   {
...     $group: {
...       _id: null,
...       missing_values: { $sum: { $cond: [{ $eq: ["$sexe", ""], 1, 0} ] } },
...       question_marks: { $sum: { $cond: [{ $eq: ["$sexe", "?"], 1, 0} ] } },
...       nd_values: { $sum: { $cond: [{ $eq: ["$sexe", "N/D"] }, 1, 0] } }
...     }
...   }
... ]);
{ "_id" : null, "missing_values" : 90, "question_marks" : 98, "nd_values" : 95 }
```

```
> db.Clients_12.updateMany(
...   { $or: [ { sexe: "" }, { sexe: "?" }, { sexe: "N/D" } ] },
...   { $set: { sexe: null } }
... );
{ "acknowledged" : true, "matchedCount" : 283, "modifiedCount" : 283 }
> db.Clients_12.distinct("sexe");
[ "M", "F", "Masculin", "Féminin", "Femme", "Homme", null ]
> ■
```

```
0: jdbc:hive2://localhost:10000> SELECT
  SUM(CASE WHEN sexe = "" THEN 1 ELSE 0 END) AS missing_values,
  SUM(CASE WHEN sexe = '?' THEN 1 ELSE 0 END) AS question_marks,
  SUM(CASE WHEN sexe = 'N/D' THEN 1 ELSE 0 END) AS nd_values
FROM client_12_ext;
```

| missing_values | question_marks | nd_values |
|----------------|----------------|-----------|
| 0              | 0              | 0         |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT sexe FROM client\_12\_ext;

| sexe     |
|----------|
| NULL     |
| F        |
| Femme    |
| Féminin  |
| Homme    |
| M        |
| Masculin |

```
> db.Clients_12.updateMany(
...   { $or: [ { sexe: "" }, { sexe: "?" }, { sexe: "N/D" } ] },
...   { $set: { sexe: null } }
... );
{ "acknowledged" : true, "matchedCount" : 283, "modifiedCount" : 283 }
> db.Clients_12.distinct("sexe");
[ "M", "F", "Masculin", "Féminin", "Femme", "Homme", null ]
>
>
>
>
> db.Clients_12.distinct("sexe");
[ "M", "F", "Masculin", "Féminin", "Femme", "Homme", null ]
> db.Clients_12.updateMany(
...   { sexe: { $in: ["Femme", "Féminin"] } },
...   { $set: { sexe: "F" } }
... );
{ "acknowledged" : true, "matchedCount" : 646, "modifiedCount" : 646 }
> db.Clients_12.updateMany(
...   { sexe: { $in: ["Homme", "Masculin"] } },
...   { $set: { sexe: "M" } }
... );
{ "acknowledged" : true, "matchedCount" : 1380, "modifiedCount" : 1380 }
>
> db.Clients_12.distinct("sexe");
[ "M", "F", null ]
> ■
```

0: jdbc:hive2://localhost:10000> SELECT DISTINCT sexe FROM client\_12\_ext;

| sexe |
|------|
| NULL |
| F    |
| M    |

#### 5.12.4- Traitement des valeurs de la colonne « situationFamiliale »

```
0: jdbc:hive2://localhost:10000> SELECT DISTINCT situationfamiliale FROM client_12_ext;
```

| situationfamiliale |
|--------------------|
| ?                  |
| Célibataire        |
| Divorcée           |
| En Couple          |
| Marié(e)           |
| N/D                |
| Seul               |
| Seule              |

```
> db.Clients_12.distinct("situationFamiliale");
[
    "En Couple",
    "Célibataire",
    "Seule",
    "",
    "Marié(e)",
    "Seul",
    "?",
    "N/D",
    "Divorcée"
]
>
> db.Clients_12.updateMany(
...     { $or: [ { situationFamiliale: "" }, { situationFamiliale: "?" }, { situationFamiliale: "N/D" } ] },
...     { $set: { situationFamiliale: null } }
... );
{
    "acknowledged" : true,
    "matchedCount" : 296,
    "modifiedCount" : 296
}
> db.Clients_12.distinct("situationFamiliale");
[
    "En Couple",
    "Célibataire",
    "Seule",
    null,
    "Marié(e)",
    "Seul",
    "Divorcée"
]
```

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN SituationFamiliale = "" THEN 1 ELSE 0 END) AS missing_values,  
SUM(CASE WHEN SituationFamiliale = '?' THEN 1 ELSE 0 END) AS question_marks,  
SUM(CASE WHEN SituationFamiliale = 'N/D' THEN 1 ELSE 0 END) AS nd_values  
FROM client_12_ext;
```

| missing_values | question_marks | nd_values |
|----------------|----------------|-----------|
| 0              | 0              | 0         |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT situationfamiliale FROM client\_12\_ext;

| situationfamiliale |
|--------------------|
| NULL               |
| Célibataire        |
| Divorcée           |
| En Couple          |
| Marié(e)           |
| Seul               |
| Seule              |

```
> db.Clients_12.distinct("situationFamiliale");
[
    "En Couple",
    "Célibataire",
    "Seule",
    null,
    "Marié(e)",
    "Seul",
    "Divorcée"
]
> db.Clients_12.updateMany(
...     { situationFamiliale: { $in: ["Seul", "Seule", "Divorcée"] } },
...     { $set: { situationFamiliale: "Célibataire" } }
... );
{ "acknowledged" : true, "matchedCount" : 5316, "modifiedCount" : 5316 }
> db.Clients_12.updateMany(
...     { situationFamiliale: "Marié(e)" },
...     { $set: { situationFamiliale: "En Couple" } }
... );
{ "acknowledged" : true, "matchedCount" : 647, "modifiedCount" : 647 }
> db.Clients_12.distinct("situationFamiliale");
[ "En Couple", "Célibataire", null ]
> -
```

0: jdbc:hive2://localhost:10000> SELECT DISTINCT situationfamiliale FROM client\_12\_ext;

| situationfamiliale |
|--------------------|
| NULL               |
| Célibataire        |
| En Couple          |

#### 5.12.5- Traitement des valeurs de la colonne « NbEnfantsAcharge »

0: jdbc:hive2://localhost:10000> SELECT DISTINCT nbenfantsacharge FROM client\_12\_ext;

| nbenfantsacharge |
|------------------|
| NULL             |
| -1               |
| 0                |
| 1                |
| 2                |
| 3                |
| 4                |

```
db.Clients_12.updateMany(
...   { nbEnfantsAcharge: { $in: ["", -1, "?"] } },
...   { $set: { nbEnfantsAcharge: null } }
... );
{ "acknowledged" : true, "matchedCount" : 286, "modifiedCount" : 286 }
> db.Clients_12.find({ nbEnfantsAcharge: { $in: ["", -1, "?"] } }).count();
0
> db.Clients_12.distinct("nbEnfantsAcharge");
[ 4, 0, 3, 1, 2, null ]
> ■
```

0: jdbc:hive2://localhost:10000> SELECT DISTINCT NbEnfantsAcharge FROM client\_12\_ext;

| nbenfantsacharge |
|------------------|
| NULL             |
| 0                |
| 1                |
| 2                |
| 3                |
| 4                |

### 5. 12.6- Traitement des valeurs de la colonne « deuxiemevoiture »

0: jdbc:hive2://localhost:10000> SELECT DISTINCT deuxiemevoiture FROM client\_12\_ext;

| deuxiemevoiture |
|-----------------|
| ?               |
| false           |
| true            |

```
> db.Clients_12.distinct("2eme voiture");
[ "false", "true", "", "?" ]
> db.Clients_12.find({ "2eme voiture": { $in: ["", "?"] } }).count();
197
> db.Clients_12.updateMany(
...     { "2eme voiture": { $in: ["", "?"] } },
...     { $set: { "2eme voiture": null } }
... );
{ "acknowledged" : true, "matchedCount" : 197, "modifiedCount" : 197 }
> db.Clients_12.distinct("2eme voiture");
[ "false", "true", null ]
> ■
```

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN deuxiemevoiture = '' THEN 1 ELSE 0 END) AS missing_values,
SUM(CASE WHEN deuxiemevoiture = '?' THEN 1 ELSE 0 END) AS question_marks
FROM client_12_ext;
```

| missing_values | question_marks |
|----------------|----------------|
| 0              | 0              |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT deuxiemevoiture FROM client\_12\_ext;

| deuxiemevoiture |
|-----------------|
| NULL            |
| false           |
| true            |

### 5. 12.7- Traitement des valeurs de la colonne « age »

0: jdbc:hive2://localhost:10000> SELECT DISTINCT age FROM client\_12\_ext;

|      |
|------|
| NULL |
| -1   |
| 18   |
| 19   |
| 20   |
| 21   |
| 22   |
| 23   |
| 24   |
| 25   |
| 26   |
| 27   |
| 28   |
| 29   |
| 30   |
| 31   |
| 32   |
| 33   |

> db.Client\_12.distinct("age");

|      |
|------|
| 78,  |
| 51,  |
| 34,  |
| 32,  |
| 47,  |
| 70,  |
| 71,  |
| 64,  |
| 82,  |
| 73,  |
| 61,  |
| "" , |
| "?", |
| -1   |

```
> db.Clients_12.find({ "age": { $in: ["", "?", -1] } }).count();
295
> db.Clients_12.updateMany(
...   { "age": { $in: ["", "?", -1] } },
...   { $set: { "age": null } }
... );
{ "acknowledged" : true, "matchedCount" : 295, "modifiedCount" : 295 }
```

0: jdbc:hive2://localhost:10000> SELECT DISTINCT age FROM client\_12\_ext;

| age  |
|------|
| NULL |
| 18   |
| 19   |
| 20   |
| 21   |
| 22   |
| 23   |
| 24   |
| 25   |
| 26   |
| 27   |
| 28   |
| 29   |
| 30   |
| 31   |
| 32   |
| 33   |
| 34   |
| 35   |
| 36   |

### 5. 12.8- Traitement des valeurs de la colonne « taux »

0: jdbc:hive2://localhost:10000> SELECT DISTINCT taux FROM client\_12\_ext;

| taux |
|------|
| NULL |
| -1   |
| 150  |
| 151  |
| 152  |
| 153  |
| 154  |
| 155  |
| 156  |

> db.Clients\_12.distinct("taux");

1169,  
"?",  
241,

404,  
-1,  
539,  
880,  
1315.

513,  
857,  
",",  
757,

```
> db.Clients_12.find({ taux: { $in: ["", -1, "?"] } }).count();
339
> ■
```

```
> db.Clients_12.updateMany(
...   { taux: { $in: ["", -1, "?"] } },
...   { $set: { taux: null } }
... );
{ "acknowledged" : true, "matchedCount" : 339, "modifiedCount" : 339 }
>
```

0: jdbc:hive2://localhost:10000> SELECT

```
SUM(CASE WHEN taux = "" THEN 1 ELSE 0 END) AS missing_values,
SUM(CASE WHEN taux = '?' THEN 1 ELSE 0 END) AS question_marks,
SUM(CASE WHEN taux = -1 THEN 1 ELSE 0 END) AS minus1_values
```

FROM client\_12\_ext;

| missing_values | question_marks | minus1_values |
|----------------|----------------|---------------|
| 0              | 0              | 0             |

row selected (1.762 seconds)

0: jdbc:hive2://localhost:10000> SELECT DISTINCT taux FROM client\_12\_ext;

| taux |
|------|
| NULL |
| 150  |
| 151  |
| 152  |
| 153  |
| 154  |
| 155  |
| 156  |
| 157  |
| 158  |
| 159  |
| 160  |
| 161  |
| 162  |
| 163  |
| 164  |
| 165  |
| 166  |
| 167  |
| 168  |
| 169  |
| 170  |
| 171  |
| 172  |
| 173  |
| 174  |
| 175  |
| 176  |
| 177  |
| 178  |
| 179  |

## 5. 12.9- Traitement des valeurs de la colonne « Immatriculation »

### 5. 12.9.1 – Vérification du format des immatriculations

```
0: jdbc:hive2://localhost:10000>SELECT DISTINCT immatriculation
FROM client_12_ext
WHERE immatriculation NOT REGEXP '^[0-9]{1,4} [A-Z]{2} [0-9]{2}$';
```

| immatriculation |
|-----------------|
|                 |
|                 |

### 5. 12.9. 2– Vérification du principe d'unicité des immatriculations

```
0: jdbc:hive2://localhost:10000> SELECT immatriculation, COUNT(*) AS nb_occurrences
FROM client_12_ext
GROUP BY immatriculation
HAVING COUNT(*) > 1;
```

| immatriculation | nb_occurrences |
|-----------------|----------------|
| 105 WI 84       | 2              |
| 1608 CZ 60      | 2              |
| 203 TL 35       | 2              |
| 5507 KS 86      | 2              |
| 5546 NA 78      | 2              |
| 568 ZI 43       | 2              |
| 5907 OD 56      | 2              |
| 7002 DU 26      | 2              |
| 8264 EA 89      | 2              |
| 8521 EV 85      | 2              |
| 8985 RJ 57      | 2              |
| 9065 KM 42      | 2              |

Nous constatons qu'il y a 12 occurrences d'une même immatriculation, ce qui est inacceptable.

0: jdbc:hive2://localhost:10000> **SELECT \* FROM client\_12\_ext WHERE immatriculation IN (SELECT immatriculation FROM client\_7\_ext**

**GROUP BY immatriculation HAVING COUNT(\*) > 1);**

| client_12_ext.age | client_12_ext sexe | client_12_ext.taux | client_12_ext.situationfamiliale | client_12_ext.nbenfantscharge | client_12_ext.deuxiemevoiture | client_12_ext.immatriculation |
|-------------------|--------------------|--------------------|----------------------------------|-------------------------------|-------------------------------|-------------------------------|
| 56                | M                  | 171                | Célibataire                      | 0                             | false                         | 105 WI 84                     |
| 62                | M                  | 837                | Célibataire                      | 0                             | false                         | 105 WI 84                     |
| 50                | F                  | 937                | En Couple                        | 3                             | false                         | 1608 CZ 60                    |
| 25                | F                  | 1039               | En Couple                        | 1                             | false                         | 1608 CZ 60                    |
| 28                | M                  | 753                | En Couple                        | 2                             | true                          | 203 TL 35                     |
| 27                | F                  | 599                | En Couple                        | 0                             | true                          | 203 TL 35                     |
| 24                | M                  | 893                | En Couple                        | 2                             | false                         | 5507 KS 86                    |
| 21                | M                  | 1310               | En Couple                        | 3                             | true                          | 5507 KS 86                    |
| 50                | F                  | 164                | En Couple                        | 2                             | false                         | 5546 NA 78                    |
| 73                | F                  | 1104               | Célibataire                      | 0                             | false                         | 5546 NA 78                    |
| 58                | M                  | 434                | En Couple                        | 4                             | false                         | 568 ZI 43                     |
| 34                | M                  | 1114               | Célibataire                      | 0                             | false                         | 568 ZI 43                     |
| 76                | M                  | 1037               | Célibataire                      | 0                             | false                         | 5907 OD 56                    |
| 27                | M                  | 797                | Célibataire                      | 0                             | false                         | 5907 OD 56                    |
| 35                | M                  | 1020               | Célibataire                      | 0                             | false                         | 7002 DU 26                    |
| 62                | M                  | 788                | En Couple                        | 1                             | false                         | 7002 DU 26                    |
| 82                | M                  | 559                | Célibataire                      | 0                             | false                         | 8264 EA 89                    |
| 38                | F                  | 553                | Célibataire                      | 0                             | false                         | 8264 EA 89                    |
| 68                | F                  | 1144               | En Couple                        | 3                             | true                          | 8521 EV 85                    |
| 49                | M                  | 1105               | Célibataire                      | 0                             | false                         | 8521 EV 85                    |
| 33                | F                  | 493                | En Couple                        | 1                             | false                         | 8985 RJ 57                    |
| 55                | F                  | 166                | En Couple                        | 1                             | false                         | 8985 RJ 57                    |
| 23                | M                  | 211                | En Couple                        | 2                             | false                         | 9065 KM 42                    |
| 81                | F                  | 573                | En Couple                        | 3                             | false                         | 9065 KM 42                    |

0: jdbc:hive2://localhost:10000> **SELECT COUNT(\*) FROM Client\_12\_ext**

|        |
|--------|
| _c0    |
| 100000 |

Il est logique de constater la présence de 12 paires d'immatriculations en double. Normalement, si nous supprimons une seule occurrence de chaque paire, à la fin de l'opération de suppression, nous aurons 99 988 enregistrements dans la table.

```
> db.Clients_12.aggregate([
...   { $group: { _id: "$immatriculation", count: { $sum: 1 } } },
...   { $match: { count: { $gt: 1 } } }
... ]);
{
  "_id" : "8521 EV 85", "count" : 2
}
{
  "_id" : "1608 CZ 60", "count" : 2
}
{
  "_id" : "9065 KM 42", "count" : 2
}
{
  "_id" : "105 WI 84", "count" : 2
}
{
  "_id" : "5507 KS 86", "count" : 2
}
{
  "_id" : "8264 EA 89", "count" : 2
}
{
  "_id" : "5907 OD 56", "count" : 2
}
{
  "_id" : "203 TL 35", "count" : 2
}
{
  "_id" : "7002 DU 26", "count" : 2
}
{
  "_id" : "5546 NA 78", "count" : 2
}
{
  "_id" : "568 ZI 43", "count" : 2
}
{
  "_id" : "8985 RJ 57", "count" : 2
}
> var immatriculationsRepetees = db.Clients_12.aggregate([
...   { $group: { _id: "$immatriculation", count: { $sum: 1 } } },
...   { $match: { count: { $gt: 1 } } }
... ]).toArray();
> immatriculationsRepetees.forEach(function(doc) {
...   var immatriculation = doc._id
...   db.Clients_12.deleteOne({ immatriculation: immatriculation });
... });
> db.Clients_12.count();
99988
> -
```

0: jdbc:hive2://localhost:10000> SELECT immatriculation, COUNT(\*) AS nb\_occurrences

FROM client\_7\_ext

GROUP BY immatriculation

HAVING COUNT(\*) > 1;

| immatriculation | nb_occurrences |
|-----------------|----------------|
|                 |                |

En exécutant cette requête une deuxième fois, elle nous renvoie un tableau vide, indiquant ainsi que toutes les immatriculations sont uniques et qu'il n'existe aucune occurrence en répétition.

### 5. 12.10 – Traitement des valeurs NULL de la table « client\_12\_ext »

0: jdbc:hive2://localhost:10000> SELECT \*FROM client\_7\_ext

WHERE immatriculation is NULL OR age is NULL OR sexe is NULL OR taux is NULL OR situationFamiliale is NULL OR nbEnfantsAcharge IS NULL OR deuxiemevoiture IS NULL;

|      |   |      |             |      |       |             |
|------|---|------|-------------|------|-------|-------------|
| NULL | F | 496  | En Couple   | 1    | true  | 282 HF 46   |
| 29   | F | NULL | En Couple   | 1    | false | 9327 PG 50  |
| 65   | M | NULL | En Couple   | 2    | false | 8630 XJ 90  |
| NULL | M | 206  | En Couple   | 4    | false | 7083 SY 97  |
| 63   | M | 421  | En Couple   | 4    | false | 2126 NC 80  |
| 55   | M | 1024 | NULL        | NULL | false | 8417 KC 41  |
| 78   | M | 592  | En Couple   | 3    | false | 777 CQ 65   |
| 24   | M | NULL | En Couple   | 4    | false | 12460 ZB 98 |
| 76   | M | NULL | En Couple   | 2    | false | 3811 CO 52  |
| NULL | M | 569  | Célibataire | 1    | false | 8654 ET 43  |
| 51   | M | 194  | NULL        | 0    | false | 290 TF 49   |
| 53   | M | NULL | Célibataire | 2    | false | 5539 KF 91  |
| 29   | M | 420  | NULL        | 1    | false | 6168 BW 31  |
| 13   | F | 1254 | En Couple   | 2    | false | 485 BJ 83   |
| 22   | F | 568  | NULL        | 0    | false | 1858 VK 65  |
| 30   | F | 435  | En Couple   | NULL | false | 4515 HE 59  |
| 38   | F | NULL | En Couple   | 0    | false | 9877 AC 60  |
| 55   | M | 577  | NULL        | 3    | false | 9553 DZ 56  |
| 58   | F | 781  | NULL        | 2    | false | 2301 CB 76  |
| 70   | M | 473  | En Couple   | 1    | false | 5286 FS 65  |
| 62   | M | 568  | NULL        | 1    | true  | 9331 SQ 40  |
| 23   | M | 226  | NULL        | 1    | false | 8150 EM 60  |
| 26   | F | 463  | Célibataire | 0    | false | 8211 OL 43  |
| 30   | F | 570  | Célibataire | 0    | false | 7437 MR 66  |
| NULL | F | 748  | En Couple   | 0    | false | 5668 ND 10  |
| 38   | M | NULL | En Couple   | 1    | false | 439 NO 47   |
| 26   | M | 1169 | En Couple   | 3    | true  | 8421 SC 67  |
| NULL | M | 417  | Célibataire | 0    | false | 4173 LD 83  |
| 79   | M | NULL | Célibataire | 0    | false | 7855 KB 12  |
| 47   | M | 280  | Célibataire | 0    | false | 5851 CW 55  |
| NULL | M | 552  | En Couple   | NULL | false | 631 PF 28   |
| 23   | M | NULL | En Couple   | 4    | false | 981 XE 13   |
| 35   | F | 561  | Célibataire | 0    | false | 767 XC 90   |
| 22   | F | 566  | Célibataire | 0    | false | 9719 XF 31  |
| 23   | M | 858  | En Couple   | 1    | false | 5532 AH 80  |
| 71   | M | 835  | En Couple   | 2    | false | 6732 BB 71  |
| NULL | F | 545  | Célibataire | 0    | false | 5657 SL 36  |
| 28   | F | 1188 | NULL        | 0    | false | 2524 XH 73  |
| 27   | M | NULL | En Couple   | 0    | false | 4387 QE 55  |
| 22   | M | 555  | NULL        | 0    | false | 4338 LW 70  |
| 41   | M | 784  | NULL        | 0    | true  | 4880 HB 97  |
| 78   | M | 1223 | En Couple   | NULL | false | 8742 CM 18  |
| 67   | M | 567  | En Couple   | 0    | true  | 2983 SD 52  |
| 29   | M | 227  | Célibataire | NULL | false | 6901 UF 31  |
| 30   | M | 231  | NULL        | 0    | false | 5247 BU 47  |
| 26   | M | 933  | En Couple   | 4    | false | 9742 YA 91  |
| 28   | M | 1274 | En Couple   | NULL | false | 1547 TW 71  |
| 20   | M | 161  | En Couple   | 3    | true  | 1873 BK 62  |
| 40   | M | 1113 | Célibataire | 1    | false | 1433 HA 51  |
| 28   | M | 581  | Célibataire | 0    | false | 5297 YB 21  |
| NULL | M | 935  | Célibataire | 2    | false | 5509 PY 22  |
| 20   | M | 932  | En Couple   | NULL | false | 7180 UV 44  |
| 18   | M | 590  | NULL        | 0    | false | 439 SW 96   |
| 26   | M | 1068 | NULL        | 2    | false | 7023 QY 10  |
| 84   | F | 195  | En Couple   | NULL | true  | 4526 WL 22  |
| 65   | F | 511  | En Couple   | NULL | false | 3378 YG 75  |
| 48   | M | 1001 | En Couple   | 2    | false | 3765 XU 54  |
| NULL | M | 1269 | Célibataire | 0    | false | 1965 ZD 73  |
| 44   | M | 181  | Célibataire | 0    | false | 5165 ZG 62  |

0: jdbc:hive2://localhost:10000> SELECT t.\*,

(CASE WHEN immatriculation IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN age IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN sexe IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN taux IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN situationFamiliale IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN nbEnfantsAcharge IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN deuxiemevoiture IS NULL THEN 1 ELSE 0 END) AS nb\_colonnes\_null

FROM client\_7\_ext t

WHERE immatriculation IS NULL OR age IS NULL OR sexe IS NULL OR taux IS NULL  
OR situationFamiliale IS NULL OR nbEnfantsAcharge IS NULL OR deuxiemevoiture IS  
NULL;

|      |      |      |             |      |       |            |   |
|------|------|------|-------------|------|-------|------------|---|
| 35   | F    | 589  | En Couple   | 2    | NULL  | 7698 BX 28 | 1 |
| 37   | M    | 1389 | Célibataire | NULL | false | 6723 YX 21 | 1 |
| 26   | M    | 408  | Célibataire | 0    | NULL  | 7592 CQ 34 | 1 |
| NULL | M    | 414  | En Couple   | 0    | false | 9625 CC 23 | 1 |
| 28   | M    | NULL | En Couple   | 2    | false | 4668 OE 55 | 1 |
| 49   | M    | 580  | En Couple   | 2    | NULL  | 2666 XD 12 | 1 |
| 24   | M    | 1377 | NULL        | 2    | false | 8931 TV 67 | 1 |
| 29   | M    | 1136 | NULL        | 3    | false | 7523 RP 20 | 1 |
| 46   | M    | NULL | En Couple   | 1    | false | 8728 ZG 85 | 1 |
| 44   | M    | 191  | NULL        | 0    | false | 9834 FF 74 | 1 |
| 67   | F    | 175  | NULL        | 0    | false | 9422 LR 48 | 1 |
| 30   | M    | 543  | Célibataire | NULL | false | 4871 OR 32 | 1 |
| NULL | M    | 208  | Célibataire | 0    | false | 4254 WH 70 | 1 |
| 61   | M    | NULL | Célibataire | 0    | false | 3555 YR 75 | 1 |
| 20   | M    | 550  | Célibataire | 3    | NULL  | 8439 RH 84 | 1 |
| 74   | M    | NULL | Célibataire | 2    | false | 9462 EP 38 | 1 |
| 51   | F    | 583  | Célibataire | NULL | false | 4736 QQ 94 | 1 |
| 29   | M    | NULL | En Couple   | 3    | false | 8646 JU 95 | 1 |
| 27   | F    | NULL | En Couple   | 3    | false | 837 BY 12  | 1 |
| 56   | F    | NULL | En Couple   | 4    | false | 8506 YS 27 | 1 |
| 57   | M    | 554  | En Couple   | 3    | NULL  | 2288 BR 54 | 1 |
| 68   | M    | 238  | En Couple   | NULL | false | 2762 IR 96 | 1 |
| 72   | M    | 563  | Célibataire | NULL | false | 1077 XU 37 | 1 |
| 25   | F    | 237  | En Couple   | 2    | NULL  | 3699 HF 30 | 1 |
| 45   | F    | NULL | En Couple   | 4    | false | 1873 TJ 24 | 1 |
| 59   | F    | 512  | Célibataire | NULL | false | 3451 CE 73 | 1 |
| NULL | F    | 996  | Célibataire | 0    | false | 3509 VG 66 | 1 |
| 27   | M    | NULL | En Couple   | 2    | false | 7644 SE 95 | 1 |
| NULL | M    | 414  | En Couple   | 4    | false | 3464 SB 42 | 1 |
| 31   | M    | 576  | En Couple   | NULL | false | 5212 OK 49 | 1 |
| 42   | M    | NULL | Célibataire | 0    | false | 893 YM 50  | 1 |
| 34   | M    | NULL | Célibataire | 0    | false | 5099 BC 54 | 1 |
| 53   | M    | 1054 | En Couple   | NULL | false | 8546 DA 59 | 1 |
| 29   | F    | 422  | En Couple   | 4    | NULL  | 4678 RE 12 | 1 |
| 48   | NULL | 592  | En Couple   | 1    | false | 8662 GB 84 | 1 |
| 25   | NULL | 459  | En Couple   | 4    | false | 9625 TO 19 | 1 |
| 76   | M    | 414  | En Couple   | NULL | true  | 79 YX 19   | 1 |
| 75   | M    | 493  | En Couple   | 1    | NULL  | 6654 JV 36 | 1 |
| NULL | M    | 544  | Célibataire | 0    | false | 2685 OA 33 | 1 |
| 76   | M    | 963  | Célibataire | NULL | false | 3352 DW 30 | 1 |
| 48   | M    | 734  | NULL        | 0    | false | 4652 AQ 31 | 1 |
| 37   | F    | 488  | Célibataire | 0    | NULL  | 566 LU 47  | 1 |
| 60   | M    | 522  | En Couple   | 2    | NULL  | 8351 TE 27 | 1 |
| 27   | M    | 167  | En Couple   | NULL | false | 280 FB 84  | 1 |
| 18   | M    | NULL | Célibataire | 0    | false | 7254 EF 55 | 1 |
| 41   | M    | 583  | Célibataire | NULL | false | 9638 UT 57 | 1 |
| 20   | F    | NULL | En Couple   | 2    | true  | 7369 FF 88 | 1 |
| 38   | M    | 156  | NULL        | 4    | false | 1807 VH 18 | 1 |
| 26   | M    | 468  | En Couple   | 1    | NULL  | 6329 MH 52 | 1 |
| NULL | M    | 492  | Célibataire | 3    | false | 5375 LO 21 | 1 |
| 47   | F    | 176  | NULL        | 0    | true  | 9983 RJ 78 | 1 |
| NULL | M    | 153  | Célibataire | 1    | false | 4891 RD 57 | 1 |
| 54   | M    | 866  | Célibataire | 0    | NULL  | 2301 MB 19 | 1 |

Il est observé que la plupart des enregistrements ont seulement un champ NULL. Il serait pertinent de filtrer les enregistrements qui ont au moins deux champs NULL.

```
> db.Clients_12.aggregate([
...   {
...     $addFields: {
...       nb_colonnes_null: {
...         $sum: [
...           { $cond: [{ $eq: ["$immatriculation", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$age", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$sexe", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$taux", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$situationFamiliale", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$nbEnfantsAcharge", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$2eme voiture", null] }, 1, 0] }
...         ]
...       }
...     }
...   },
...   {
...     $match: {
...       $and: [
...         { $or: [{ immatriculation: null }, { age: null }, { sexe: null }, { taux: null }, { situationFamiliale: null }, { nbEnfantsAcharge: null }, { "2eme voiture": null }] },
...         { nb_colonnes_null: { $gte: 2 } }
...       ]
...     }
...   }
... ])
```

```
"_id": ObjectId("661581dc4817545a7e046f36"), "age": 27, "sexe": null, "taux": null, "situationFamiliale": "Célibataire", "nbEnfantsAcharge": 0, "2eme voiture": "false", "immatriculation": "2271 GH 52", "nb_colonnes_null": 2 },
"._id": ObjectId("661581db4817545a7e046dfc"), "age": null, "sexe": "M", "taux": null, "situationFamiliale": "En Couple", "nbEnfantsAcharge": 0, "2eme voiture": "false", "immatriculation": "1561 TI 75", "nb_colonnes_null": 2 },
"._id": ObjectId("661581db4817545a7e046d997"), "age": 58, "sexe": "M", "taux": null, "situationFamiliale": null, "nbEnfantsAcharge": 4, "2eme voiture": "false", "immatriculation": "1081 RP 47", "nb_colonnes_null": 2 },
"._id": ObjectId("661581db4817545a7e046d788"), "age": 56, "sexe": null, "taux": null, "situationFamiliale": "En Couple", "nbEnfantsAcharge": 2, "2eme voiture": "false", "immatriculation": "627 AX 72", "nb_colonnes_null": 2 },
"._id": ObjectId("661581db4817545a7e046ed13"), "age": 56, "sexe": null, "taux": 243, "situationFamiliale": "En Couple", "nbEnfantsAcharge": null, "2eme voiture": "false", "immatriculation": "5107 NO 84", "nb_colonnes_null": 2 },
"._id": ObjectId("661581db4817545a7e046dce7"), "age": 79, "sexe": "M", "taux": 473, "situationFamiliale": "En Couple", "nbEnfantsAcharge": null, "2eme voiture": "false", "immatriculation": "3188 US 65", "nb_colonnes_null": 2 },
"._id": ObjectId("661581db4817545a7e046d946"), "age": 42, "sexe": "M", "taux": null, "situationFamiliale": null, "nbEnfantsAcharge": 4, "2eme voiture": "false", "immatriculation": "1664 KZ 82", "nb_colonnes_null": 2 },
"._id": ObjectId("661581db4817545a7e0465eab"), "age": null, "sexe": "M", "taux": null, "situationFamiliale": "En Couple", "nbEnfantsAcharge": 2, "2eme voiture": "false", "immatriculation": "5091 LA 41", "nb_colonnes_null": 2 },
"._id": ObjectId("661581db4817545a7e0469323"), "age": null, "sexe": "F", "taux": null, "situationFamiliale": "En Couple", "nbEnfantsAcharge": 1, "2eme voiture": "false", "immatriculation": "3239 MU 15", "nb_colonnes_null": 2 },
"._id": ObjectId("661581dc4817545a7e046f478"), "age": 55, "sexe": null, "taux": 545, "situationFamiliale": "Célibataire", "nbEnfantsAcharge": null, "2eme voiture": "false", "immatriculation": "9655 NC 34", "nb_colonnes_null": 2 }
```

Nous observons que 11 enregistrements présentent deux colonnes avec des valeurs NULL, dont plus de la moitié ont des valeurs NULL dans des colonnes telles que le sexe, l'âge et le nombre d'enfants à charge. Ces données sont critiques et importantes pour notre analyse. Par conséquent, nous concluons qu'il est nécessaire de supprimer ces 11 lignes, car elles n'apportent pas une grande

valeur significative par rapport aux lignes ne présentant qu'une seule colonne avec une valeur NULL.

```

var documentsASupprimer = db.Clients_12.aggregate([
...   {
...     $addFields: {
...       nb_colonnes_null: {
...         $sum: [
...           { $cond: [{ $eq: ["$immatriculation", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$age", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$sexe", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$taux", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$situationFamiliale", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$nbEnfantsAcharge", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$2eme voiture", null] }, 1, 0] }
...         ]
...       }
...     }
...   },
...   {
...     $match: {
...       $and: [
...         { $or: [{ immatriculation: null }, { age: null }, { sexe: null }, { taux: null }, { situationFamiliale: null }, { nbEnfantsAcharge: null }, { "2eme voiture": null } ] },
...         { nb_colonnes_null: { $gte: 2 } }
...       ]
...     }
...   }
... ]).toArray();
documentsASupprimer.forEach(function(document) {
...   db.Clients_12.deleteOne({ _id: document._id });
... });

db.Clients_12.aggregate([
...   {
...     $addFields: {
...       nb_colonnes_null: {
...         $sum: [
...           { $cond: [{ $eq: ["$immatriculation", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$age", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$situationFamiliale", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$nbEnfantsAcharge", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$2eme voiture", null] }, 1, 0] },
...           { $cond: [{ $eq: ["$deuxiemevoiture", null] }, 1, 0] }
...         ]
...       }
...     }
...   },
...   {
...     $match: {
...       $and: [
...         { $or: [{ immatriculation: null }, { age: null }, { sexe: null }, { taux: null }, { situationFamiliale: null }, { nbEnfantsAcharge: null }, { "2eme voiture": null } ] },
...         { nb_colonnes_null: { $gte: 2 } }
...       ]
...     }
...   }
... ]). toArray();

```

0: jdbc:hive2://localhost:10000> SELECT \*

FROM (

SELECT t.\*,

(CASE WHEN immatriculation IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN age IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN sexe IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN taux IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN situationFamiliale IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN nbEnfantsAcharge IS NULL THEN 1 ELSE 0 END) +

(CASE WHEN deuxiemevoiture IS NULL THEN 1 ELSE 0 END) AS  
nb\_colonnes\_null

FROM client\_12\_ext t

) t

WHERE (immatriculation IS NULL OR age IS NULL OR sexe IS NULL OR taux IS  
NULL OR situationFamiliale IS NULL OR nbEnfantsAcharge IS NULL OR  
deuxiemevoiture IS NULL) AND nb\_colonnes\_null >= 2;

| t.age | t.sexé | t.taux | t.situationfamiliale | t.nbenfantsacharge | t.deuxiemevoiture | t.immatriculation | t.nb_colonnes_null |
|-------|--------|--------|----------------------|--------------------|-------------------|-------------------|--------------------|
| +     | +      | +      | +                    | +                  | +                 | +                 | +                  |
| +     | +      | +      | +                    | +                  | +                 | +                 | +                  |

### 5.12.11 – Nombre d'enregistrement de la table « client\_12\_ext » après nettoyage et transformation

D'après les traitements effectués précédemment, 9 enregistrements ont été supprimés en raison d'occurrences d'immatriculations en double, et 11 enregistrements ont été supprimés en raison de la présence d'au moins deux champs NULL. Logiquement, il devrait rester 99980 enregistrements dans la table.

0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) from client\_12\_ext;

| _c0          |
|--------------|
| 99977        |
| row selected |

En effet, le décompte est correct et la table Hive "client\_7\_ext" est conforme aux normes du dictionnaire de données.

## 6. Hadoop Map Reduce

Durant la phase de Map Reduce, nous avons employé pyspark dans un environnement Jupyter Notebook pour effectuer le nettoyage et la transformation des données du fichier CO2.csv.

### Import des modules nécessaires

```
Entrée [ ]:
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.functions import split, col, regexp_extract, substring_index, regexp_replace, round
from pyspark.sql.functions import mean, when, avg, lit, upper
```

### Initialisation de la session Spark

```
Entrée [ ]:
spark = SparkSession.builder \
    .appName("Intégration des données CO2 dans le catalogue du concessionnaire") \
    .getOrCreate()

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
```

Nous avons utilisé PySpark pour interagir avec Apache Spark et effectuer des opérations sur des données structurées. On a importé des modules essentiels comme SparkSession pour initialiser une session Spark et des fonctions de traitement de données telles que F pour accéder à des fonctions prédéfinies. Ensuite, nous avons crée une session Spark nommée "Intégration des données CO2 dans le catalogue du concessionnaire" à l'aide de SparkSession.builder. Cette session Spark permettra d'exécuter des opérations de transformation et de nettoyage sur les données CO2.

### Chargement des données

```
Entrée [ ]:
co2_data = spark.read.csv("/MBDS_Projet/CO2.csv", header=True)

co2_data.show()
```

| [_c0] | Marque / Modèle       | Bonus / Malus | Rejets CO2 g/km | Cout énergie |
|-------|-----------------------|---------------|-----------------|--------------|
| 2     | AUDI E-TRON SPORT...  | -6 000€ 1     | 0               | 319 €        |
| 3     | AUDI E-TRON SPORT...  | -6 000€ 1     | 0               | 356 €        |
| 4     | AUDI E-TRON 55 (4...) | -6 000€ 1     | 0               | 357 €        |
| 5     | AUDI E-TRON 50 (3...) | -6 000€ 1     | 0               | 356 €        |
| 6     | BMW i3s 120 Ah        | -6 000€ 1     | 0               | 204 €        |
| 7     | BMW i3s 120 Ah        | -6 000€ 1     | 0               | 204 €        |
| 8     | CITROEN BERLINGO      | -6 000€ 1     | 0               | 203 €        |
| 9     | CITROEN C-ZERO        | -6 000€ 1     | 0               | 491 €        |
| 10    | DS DS3 CROSSBACK ...  | -6 000€ 1     | 0               | 251 €        |
| 11    | HYUNDAI KONA elec...  | -6 000€ 1     | 0               | 205 €        |
| 12    | HYUNDAI KONA elec...  | -6 000€ 1     | 0               | 205 €        |
| 13    | JAGUAR I-PACE EV4...  | -6 000€ 1     | 0               | 271 €        |
| 14    | KIA e-NIRO Moteur...  | -6 000€ 1     | 0               | 212 €        |
| 15    | KIA e-NIRO Moteur...  | -6 000€ 1     | 0               | 203 €        |
| 16    | KIA SOUL Moteur Á...  | -6 000€ 1     | 0               | 214 €        |
| 17    | KIA SOUL Moteur Á...  | -6 000€ 1     | 0               | 214 €        |
| 18    | MERCEDES EQC 400 ...  | -6 000€ 1     | 0               | 291 €        |
| 19    | MERCEDES VITO Tou...  | -6 000€ 1     | 0               | 411 €        |
| 20    | MERCEDES VITO Tou...  | -6 000€ 1     | 0               | 411 €        |
| 21    | MINI MINI Cooper ...  | -6 000€ 1     | 0               | 199 €        |

only showing top 20 rows

On a lu le fichier CSV contenant des données sur le CO2. Il utilise la méthode spark.read.csv() pour charger les données à partir du fichier "CO2.csv" situé dans le répertoire "/MBDS\_Projet/", en spécifiant que la première ligne du fichier contient les en-têtes de colonne (header=True).

Ensuite, nous avons affiché les premières lignes de données à l'aide de la méthode show() pour permettre une visualisation rapide des données chargées.

### Nettoyage des données

Entrée [ ]:

```
# Suppression du caractère de l'euro dans la colonne "Cout energie" et renommer la colonne par "Cout Energie"
co2_data_cleaned = co2_data.withColumn("Cout Energie", F.regexp_replace("cout energie", "€", ""))
co2_data_cleaned = co2_data_cleaned.drop("cout energie")

# Supprimer les caractères non numériques (garder uniquement les chiffres et les décimales)
co2_data_cleaned = co2_data_cleaned.withColumn("Cout Energie", regexp_replace(col("Cout Energie"), "[^0-9.]", ""))

# Convertir la colonne en type numérique
co2_data_cleaned = co2_data_cleaned.withColumn("Cout Energie", co2_data_cleaned["Cout Energie"].cast("double"))

co2_data_cleaned.show()
```

| <u>_c0</u> | Marque / Modèle           | Bonus / Malus | Rejets CO2 g/km | Cout Energie |
|------------|---------------------------|---------------|-----------------|--------------|
| 2          | AUDI E-TRON SPORTBACK ... | -6 000€ 1     | 0               | 319.0        |
| 3          | AUDI E-TRON SPORTBACK ... | -6 000€ 1     | 0               | 356.0        |
| 4          | AUDI E-TRON 55 (4...)     | -6 000€ 1     | 0               | 357.0        |
| 5          | AUDI E-TRON 50 (3...)     | -6 000€ 1     | 0               | 356.0        |
| 6          | BMW i3 120 Ah             | -6 000€ 1     | 0               | 204.0        |
| 7          | BMW i3s 120 Ah            | -6 000€ 1     | 0               | 204.0        |
| 8          | CITROEN BERLINGO          | -6 000€ 1     | 0               | 203.0        |
| 9          | CITROEN C-ZERO            | -6 000€ 1     | 0               | 491.0        |
| 10         | DS DS3 CROSSBACK ...      | -6 000€ 1     | 0               | 251.0        |
| 11         | HYUNDAI KONA elec...      | -6 000€ 1     | 0               | 205.0        |
| 12         | HYUNDAI KONA elec...      | -6 000€ 1     | 0               | 205.0        |
| 13         | JAGUAR I-PACE EV40...     | -6 000€ 1     | 0               | 271.0        |
| 14         | KIA e-NIRO Moteur...      | -6 000€ 1     | 0               | 212.0        |
| 15         | KIA e-NIRO Moteur...      | -6 000€ 1     | 0               | 203.0        |
| 16         | KIA SOUL Moteur A...      | -6 000€ 1     | 0               | 214.0        |
| 17         | KIA SOUL Moteur A...      | -6 000€ 1     | 0               | 214.0        |
| 18         | MERCEDES EQC 400 ...      | -6 000€ 1     | 0               | 291.0        |
| 19         | MERCEDES VITO Tou...      | -6 000€ 1     | 0               | 411.0        |
| 20         | MERCEDES VITO Tou...      | -6 000€ 1     | 0               | 411.0        |
| 21         | MINI MINI Cooper ...      | -6 000€ 1     | 0               | 199.0        |

Nous avons effectué plusieurs opérations de nettoyage des données sur la colonne "Cout energie" du DataFrame "co2\_data". Tout d'abord, nous renommons la colonne en "Cout Energie" et supprime le symbole de l'euro en remplaçant tous les occurrences de "€" par une chaîne vide. Ensuite, nous avons supprimé tous les caractères non numériques de la colonne, ne conservant que les chiffres et les décimales. Enfin, nous avons convertit la colonne en type numérique. Après chaque opération, nous affichons les premières lignes du DataFrame nettoyé à l'aide de la méthode show().

Entrée [ ]: # Séparation de la colonne "Marque / Modèle" en "Marque" et "Modèle"

```
# La séparation de la colonne "Marque / Modèle" en deux colonnes distinctes "Marque" et "Modèle" a été réalisée dans le but de mieux faciliter la manipulation et l'analyse des données.
# Cette séparation permet de distinguer clairement la marque du véhicule de son modèle, ce qui facilite la manipulation et l'analyse.
```

```
co2_data_cleaned = co2_data_cleaned.withColumn("Marque", regexp_extract(col("Marque / Modèle"), r"(\w+)", 1))
co2_data_cleaned = co2_data_cleaned.withColumn("Modèle", regexp_extract(col("Marque / Modèle"), r"\s(.*)", 1))
co2_data_cleaned = co2_data_cleaned.drop("Marque / Modèle")
```

```
co2_data_cleaned.show()
```

| <u>_c0</u> | Bonus / Malus | Rejets CO2 g/km | Cout Energie | Marque   | Modèle                 |
|------------|---------------|-----------------|--------------|----------|------------------------|
| 2          | -6 000€ 1     | 0               | 319.0        | AUDI     | E-TRON SPORTBACK ...   |
| 3          | -6 000€ 1     | 0               | 356.0        | AUDI     | E-TRON SPORTBACK ...   |
| 4          | -6 000€ 1     | 0               | 357.0        | AUDI     | E-TRON 55 (488ch)...   |
| 5          | -6 000€ 1     | 0               | 356.0        | AUDI     | E-TRON 50 (313ch)...   |
| 6          | -6 000€ 1     | 0               | 284.0        | BMW      | i3 120 Ah              |
| 7          | -6 000€ 1     | 0               | 284.0        | BMW      | i3s 120 Ah             |
| 8          | -6 000€ 1     | 0               | 203.0        | CITROEN  | BERLINGO               |
| 9          | -6 000€ 1     | 0               | 491.0        | CITROEN  | C-ZERO                 |
| 10         | -6 000€ 1     | 0               | 251.0        | DS       | DS3 CROSSBACK E-...    |
| 11         | -6 000€ 1     | 0               | 205.0        | HYUNDAI  | KONA electric 64 kWh   |
| 12         | -6 000€ 1     | 0               | 205.0        | HYUNDAI  | KONA electric 39 kWh   |
| 13         | -6 000€ 1     | 0               | 271.0        | JAGUAR   | I-PACE EV400 (400...)  |
| 14         | -6 000€ 1     | 0               | 212.0        | KIA      | e-NIRO Moteur A01...   |
| 15         | -6 000€ 1     | 0               | 203.0        | KIA      | e-NIRO Moteur A01...   |
| 16         | -6 000€ 1     | 0               | 214.0        | KIA      | SOUL Moteur A0lelec... |
| 17         | -6 000€ 1     | 0               | 214.0        | KIA      | SOUL Moteur A0lelec... |
| 18         | -6 000€ 1     | 0               | 291.0        | MERCEDES | EQC 400 4MATIC         |
| 19         | -6 000€ 1     | 0               | 411.0        | MERCEDES | VITO Tourer long ...   |
| 20         | -6 000€ 1     | 0               | 411.0        | MERCEDES | VITO Tourer extra...   |
| 21         | -6 000€ 1     | 0               | 199.0        | MINI     | MINI Cooper SE Ha...   |

only showing top 20 rows

Extraction de la colonne "Marque / Modèle" en deux colonnes distinctes : "Marque" et "Modèle". Cette opération de séparation vise à mieux organiser et structurer les données en distinguant clairement la marque du véhicule de son modèle. Cela facilite la manipulation et l'analyse ultérieures des données en permettant une compréhension plus claire et une exploration plus efficace des informations contenues dans le dataframe.

| Entrée [ ]: | <pre># Supprimer les caractères après le symbole de l'euro dans la colonne "Bonus / Malus" co2_data_cleaned = co2_data_cleaned.withColumn("Bonus / Malus", regexp_replace("Bonus / Malus", "€.*", "€"))  # Définir une nouvelle colonne "Devise" en extrayant la devise de la colonne "Bonus / Malus" co2_data_cleaned = co2_data_cleaned.withColumn("Devise", regexp_extract(col("Bonus / Malus"), r'(-?\d+\s?\d*)\s?([€\$])', 2))  # Supprimer le symbole de l'euro dans la colonne "Bonus / Malus" co2_data_cleaned = co2_data_cleaned.withColumn("Bonus / Malus", regexp_replace("Bonus / Malus", "€", ""))  # Remplacer "-" par 0 seulement si la colonne entière contient un "-" co2_data_cleaned = co2_data_cleaned.withColumn("Bonus / Malus",                                               when(col("Bonus / Malus") == "-", "0")                                               .otherwise(col("Bonus / Malus")))  co2_data_cleaned.show()</pre>                                                                                                                                                                                                                                                                                                             |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <pre>+---+-----+-----+-----+-----+-----+   _c0 Bonus / Malus Rejets CO2 g/km Cout Energie  Marque   Modèle  Devise  +---+-----+-----+-----+-----+-----+   2  -6 000  0  319.0  AUDI E-TRON SPORTBACK ...  €    3  -6 000  0  356.0  AUDI E-TRON SPORTBACK ...  €    4  -6 000  0  357.0  AUDI E-TRON 55 (408ch)...  €    5  -6 000  0  356.0  AUDI E-TRON 50 (313ch)...  €    6  -6 000  0  204.0  BMW i3 120 Ah  €    7  -6 000  0  204.0  BMW i3s 120 Ah  €    8  -6 000  0  203.0  CITROEN  BERLINGO  €    9  -6 000  0  491.0  CITROEN  C-ZERO  €    10 -6 000  0  251.0  DS DS3 CROSSBACK E-T...  €    11 -6 000  0  205.0  HYUNDAI KONA electric 64 kWh  €    12 -6 000  0  205.0  HYUNDAI KONA electric 39 kWh  €    13 -6 000  0  271.0  JAGUAR I-PACE EV400 (400...  €    14 -6 000  0  212.0  KIA e-NIRO Moteur Aéol...  €    15 -6 000  0  203.0  KIA e-NIRO Moteur Aéol...  €    16 -6 000  0  214.0  KIA SOUL Moteur Aélec...  €    17 -6 000  0  214.0  KIA SOUL Moteur Aélec...  €    18 -6 000  0  291.0 MERCEDES  EQC 400 4MATIC  €    19 -6 000  0  411.0 MERCEDES  VITO Tourer long ...  €    20 -6 000  0  411.0 MERCEDES  VITO Tourer extra...  €    21 -6 000  0  199.0  MINI MINI Cooper SE Ha...  €  +---+-----+-----+-----+-----+-----+</pre> |

Dans cette partie, on a effectué plusieurs manipulations sur la colonne "Bonus / Malus" du dataframe. Tout d'abord, nous avons supprimé les caractères après le symbole de l'euro dans cette colonne. Ensuite, nous avons extrait la devise de la colonne "Bonus / Malus" pour la stocker dans une nouvelle colonne appelée "Devise". Après cela, nous avons supprimé le symbole de l'euro de la colonne "Bonus / Malus". Enfin, nous avons remplacé les tirets "-" par 0 uniquement si la colonne entière contient un tiret, assurant ainsi une uniformité dans la représentation des valeurs.

```
Entrée [ ]: # Utiliser une expression régulière pour extraire les parties numériques ainsi que les signes "+", "-"
co2_data_cleaned = co2_data_cleaned.withColumn("Bonus / Malus",
    when(
        co2_data_cleaned["Bonus / Malus"].rlike(r'^[+-]?\d+(\.\d+)?'),
        co2_data_cleaned["Bonus / Malus"]
    ).otherwise("0"))

# Utiliser une expression régulière pour extraire le signe de la colonne "Bonus / Malus"
co2_data_cleaned = co2_data_cleaned.withColumn("Signe", regexp_extract(co2_data_cleaned["Bonus / Malus"], r'([+-])', 1))

# Supprimer les caractères non numériques (garder uniquement les chiffres et les décimales)
co2_data_cleaned = co2_data_cleaned.withColumn("Bonus / Malus", regexp_replace(col("Bonus / Malus"), "[^0-9.]", ""))

# Convertir la colonne de chaînes en colonne de doubles en conservant le signe
co2_data_cleaned= co2_data_cleaned.withColumn("Bonus / Malus", col("Bonus / Malus").cast("double"))

# Multiplier les valeurs de la colonne "Bonus/Malus" par -1 lorsque le signe est "-"
co2_data_cleaned = co2_data_cleaned.withColumn("Bonus / Malus",
    when(col("Signe") == "-", col("Bonus / Malus") * -1)
    .otherwise(col("Bonus / Malus")))

# Supprimer la colonne "Signe"
co2_data_cleaned = co2_data_cleaned.drop("Signe")

co2_data_cleaned.show()
```

| CO2 | Bonus / Malus | Rejets CO2 g/km | Cout Energie | Marque   | Modèle                | Devise |
|-----|---------------|-----------------|--------------|----------|-----------------------|--------|
| 2   | -6000.0       | 0               | 319.0        | AUDI     | E-TRON SPORTBACK ...  | €      |
| 3   | -6000.0       | 0               | 356.0        | AUDI     | E-TRON SPORTBACK ...  | €      |
| 4   | -6000.0       | 0               | 357.0        | AUDI     | E-TRON 55 (408ch) ... | €      |
| 5   | -6000.0       | 0               | 356.0        | AUDI     | E-TRON 50 (333ch) ... | €      |
| 6   | -6000.0       | 0               | 284.0        | BMW      | i3 120 Ah             | €      |
| 7   | -6000.0       | 0               | 284.0        | BMW      | i3s 120 Ah            | €      |
| 8   | -6000.0       | 0               | 203.0        | CITROËN  | BERLINGO              | €      |
| 9   | -6000.0       | 0               | 491.0        | CITROËN  | C-ZERO                | €      |
| 10  | -6000.0       | 0               | 251.0        | DS       | DS3 CROSSBACK E-T...  | €      |
| 11  | -6000.0       | 0               | 205.0        | HYUNDAI  | KONA electric 64 kWh  | €      |
| 12  | -6000.0       | 0               | 205.0        | HYUNDAI  | KONA electric 36 kWh  | €      |
| 13  | -6000.0       | 0               | 271.0        | JAGUAR   | I-PACE EV400 (400...  | €      |
| 14  | -6000.0       | 0               | 212.0        | KIA      | e-NIRO Moteur A01...  | €      |
| 15  | -6000.0       | 0               | 283.0        | KIA      | e-NIRO Moteur A01...  | €      |
| 16  | -6000.0       | 0               | 214.0        | KIA      | SOUL Moteur A01lec... | €      |
| 17  | -6000.0       | 0               | 214.0        | KIA      | SOUL Moteur A01lec... | €      |
| 18  | -6000.0       | 0               | 291.0        | MERCEDES | EQC 400 4MATIC        | €      |
| 19  | -6000.0       | 0               | 411.0        | MERCEDES | VITO Tourer long ...  | €      |
| 20  | -6000.0       | 0               | 411.0        | MERCEDES | VITO Tourer extra...] | €      |
| 21  | -6000.0       | 0               | 199.0        | MINI     | MINI Cooper SE Ha...  | €      |

Initialement, on a opté à utiliser une expression régulière pour extraire les parties numériques ainsi que les signes "+" et "-", en attribuant la valeur "0" aux entrées qui ne correspondent pas à ce modèle. Ensuite, la prochaine étape était d'extraire le signe de la colonne "Bonus / Malus" dans une nouvelle colonne appelée "Signe". Nous avons supprimé ensuite les caractères non numériques de la colonne "Bonus / Malus" pour ne conserver que les chiffres et les décimales. Après cela, nous convertissons la colonne en nombres flottants. Ensuite, nous avons multiplié les valeurs de la colonne "Bonus / Malus" par -1 lorsque le signe est "-", assurant ainsi que les valeurs négatives sont correctement traitées. Enfin, nous avons supprimé la colonne "Signe" car elle n'est plus nécessaire.

#### Préparation des valeurs calculées :

la moyenne pour "Bonus / Malus"

```
Entrée [ ]: # Calcul de la moyenne pour "Bonus / Malus" (en excluant les valeurs nulles)
avg_bonus_malus = co2_data_cleaned.filter(col("Bonus / Malus") != 0.0).groupBy("Marque").agg(F.avg("Bonus / Malus")).alias("Moyenne")
avg_bonus_malus.show()
```

| Marque     | Moyenne Bonus/Malus en euro |
|------------|-----------------------------|
| MERCEDES   | 8237.358422939069           |
| HYUNDAI    | -6000.0                     |
| SKODA      | -6000.0                     |
| NISSAN     | 5802.4                      |
| CITROËN    | -6000.0                     |
| AUDI       | -6000.0                     |
| MINI       | -6000.0                     |
| PEUGEOT    | -6000.0                     |
| JAGUAR     | -6000.0                     |
| TESLA      | -6000.0                     |
| BMW        | -6000.0                     |
| VOLKSWAGEN | -6000.0                     |
| KIA        | -6000.0                     |
| SMART      | -6000.0                     |
| RENAULT    | -6000.0                     |
| DS         | -6000.0                     |

Ce partie effectue le calcul de la moyenne du "Bonus / Malus" pour chaque marque de véhicule, en excluant les valeurs nulles. Nous avons commencé par filtrer les lignes où la valeur du "Bonus / Malus" est différente de zéro, puis regroupe les données par marque. Ensuite, nous avons calculé

la moyenne du "Bonus / Malus" pour chaque groupe de marque. Le résultat est affiché sous forme de tableau, montrant la moyenne du "Bonus / Malus" pour chaque marque en euros.

```
La moyenne pour "Rejets CO2 g/km"

Entrée [ ]: # Convertir la colonne "Rejets CO2 g/km" en type numérique
co2_data_cleaned = co2_data_cleaned.withColumn("Rejets CO2 g/km", co2_data_cleaned["Rejets CO2 g/km"].cast("double"))

# Calcul de la moyenne pour "Rejets CO2 g/km" (en excluant les valeurs égales à zéro)
avg_co2_emission = co2_data_cleaned.filter((col("Rejets CO2 g/km").rlike(r'^\d+(\.\d+)?$')) & (col("Rejets CO2 g/km") != 0)) \
    .groupBy("Marque") \
    .agg(F.avg("Rejets CO2 g/km").alias("Moyenne Rejets CO2 g/km"))

avg_co2_emission.show()

+-----+-----+
| Marque|Moyenne Rejets CO2 g/km|
+-----+-----+
MERCEDES	189.55479452054794
PORSCHE	69.85714285714286
HYUNDAI	26.0
TOYOTA	32.0
SKODA	31.0
LAND	69.0
NISSAN	200.0
BENTLEY	84.0
AUDI	43.5
MINI	43.0
PEUGEOT	31.66666666666668
VOLVO	42.45454545454545
BMW	43.88235294117647
VOLKSWAGEN	32.8
KIA	31.0
DS	33.0
MITSUBISHI	40.0
+-----+-----+
```

Conversion de la colonne "Rejets CO2 g/km" en type numérique, ce qui est essentiel pour effectuer des calculs statistiques. Ensuite, il filtre les valeurs de cette colonne pour exclure celles qui sont égales à zéro, puis regroupe les données par marque de véhicule. Par la suite, de la même manière nous avons calculé la moyenne des rejets de CO2 par kilomètre pour chaque groupe de marque. Enfin, nous avons affiché les résultats sous forme de tableau, présentant la moyenne des rejets de CO2 par kilomètre pour chaque marque.

```
La moyenne du coût de l'énergie

Entrée [ ]: # Calcul du coût d'énergie pour chaque marque
energy_cost_per_brand = co2_data_cleaned.groupBy("Marque").agg(F.avg("Cout Energie").alias("Cout Energie Moyen en euro"))
energy_cost_per_brand.show()

+-----+-----+
| Marque|Cout Energie Moyen en euro|
+-----+-----+
MERCEDES	749.9796610169492
PORSCHE	89.71428571428571
HYUNDAI	151.0
TOYOTA	43.0
SKODA	98.88888888888889
NISSAN	681.2
LAND	78.0
CITROEN	347.0
BENTLEY	102.0
AUDI	191.6
MINI	126.0
PEUGEOT	144.1666666666666
JAGUAR	271.0
VOLVO	72.727272727273
TESLA	245.8888888888889
BMW	88.52631578947368
VOLKSWAGEN	96.0
KIA	157.6666666666666
SMART	191.36363636363637
RENAULT	206.0
+-----+
only showing top 20 rows
```

Ce segment de code effectue le calcul du coût moyen de l'énergie pour chaque marque de véhicule. Il regroupe de la même manière les données par marque, puis calcule la moyenne du coût de

l'énergie pour chaque groupe de marque. Ensuite, il affiche le résultat sous forme de tableau, présentant le coût moyen de l'énergie en euros pour chaque marque.

```
Entrée [ 1]: # Joindre les trois dataframes en un seul sur la colonne "Marque"
merged_data = avg_bonus_malus.join(avg_co2_emission, "Marque", "outer").join(energy_cost_per_brand, "Marque", "outer")

# Afficher le tableau combiné avec les moyennes arrondies à 3 chiffres après la virgule
merged_data = merged_data.select(
    col("Marque"),
    round(col("Moyenne Bonus/Malus en euro"), 3).alias("Moyenne Bonus/Malus en euro"),
    round(col("Moyenne Rejets CO2 g/km"), 3).alias("Moyenne Rejets CO2 g/km"),
    round(col("Cout Energie Moyen en euro"), 3).alias("Cout Energie Moyen en euro")
)

# Remplacer les valeurs NULL dans les colonnes "Moyenne Bonus/Malus" et "Moyenne Rejets CO2" par 0
merged_data = merged_data.na.fill(0, subset=["Moyenne Bonus/Malus en euro", "Moyenne Rejets CO2 g/km"])

# Afficher le tableau combiné
merged_data.show(truncate=False)
```

| Marque     | Moyenne Bonus/Malus en euro | Moyenne Rejets CO2 g/km | Cout Energie Moyen en euro |
|------------|-----------------------------|-------------------------|----------------------------|
| AUDI       | -6000.0                     | 143.5                   | 191.6                      |
| BENTLEY    | 0.0                         | 84.0                    | 102.0                      |
| BMW        | -6000.0                     | 143.882                 | 80.526                     |
| CITROEN    | -6000.0                     | 0.0                     | 347.0                      |
| DS         | -6000.0                     | 33.0                    | 159.0                      |
| HYUNDAI    | -6000.0                     | 26.0                    | 151.0                      |
| JAGUAR     | -6000.0                     | 0.0                     | 271.0                      |
| LADA       | -6000.0                     | 31.0                    | 157.667                    |
| LAND       | 0.0                         | 0.0                     | 178.0                      |
| MERCEDES   | 8237.358                    | 189.555                 | 749.98                     |
| MINI       | -6000.0                     | 43.0                    | 126.0                      |
| MITSUBISHI | 0.0                         | 40.0                    | 98.0                       |
| NISSAN     | 5802.4                      | 200.0                   | 681.2                      |
| PEUGEOT    | -6000.0                     | 31.667                  | 144.167                    |
| PORSCHE    | 0.0                         | 69.857                  | 89.714                     |
| RENAULT    | -6000.0                     | 0.0                     | 206.0                      |
| SKODA      | -6000.0                     | 31.0                    | 98.889                     |
| SMART      | -6000.0                     | 0.0                     | 191.364                    |
| TESLA      | -6000.0                     | 0.0                     | 245.889                    |
| TOYOTA     | 0.0                         | 132.0                   | 43.0                       |

only showing top 20 rows

L'idée était d'effectué une jointure des trois dataframes sur la colonne "Marque". Ensuite, nous avons sélectionné les colonnes pertinentes et arrondit les valeurs des moyennes à trois chiffres après la virgule. Nous avons remplacé également les valeurs NULL dans les colonnes "Moyenne Bonus/Malus" et "Moyenne Rejets CO2" par 0. Enfin, on a affiché le tableau combiné résultant, présentant les moyennes du bonus/malus, des rejets CO2 et du coût de l'énergie moyens pour chaque marque, avec les valeurs arrondies.

```
Entrée [ 2]: # Calcul de la moyenne globale pour "Bonus / Malus" (en excluant les valeurs nulles)
global_avg_bonus_malus = co2_data_cleaned.filter(col("Bonus / Malus") != 0).agg(F.avg("Bonus / Malus")).collect()[0][0]

# Calcul de la moyenne globale pour "Rejets CO2 g/km" (en excluant les valeurs égales à zéro)
global_avg_co2_emission = co2_data_cleaned.filter(col("Rejets CO2 g/km") != 0).agg(F.avg("Rejets CO2 g/km")).collect()[0][0]

# Calcul de la moyenne globale pour "Cout Energie"
global_avg_energy_cost = co2_data_cleaned.agg(F.avg("Cout Energie")).collect()[0][0]

# Formater les moyennes avec trois chiffres après la virgule
global_avg_bonus_malus = "{:.3f}".format(global_avg_bonus_malus)
global_avg_co2_emission = "{:.3f}".format(global_avg_co2_emission)
global_avg_energy_cost = "{:.3f}".format(global_avg_energy_cost)

# Affichage des moyennes globales
print("Moyenne globale de Bonus/Malus (en excluant les valeurs nulles) : ", global_avg_bonus_malus)
print("Moyenne globale de Rejets CO2 (en excluant les valeurs égales à zéro) : ", global_avg_co2_emission)
print("Moyenne globale de Cout Energie : ", global_avg_energy_cost)

Moyenne globale de Bonus/Malus (en excluant les valeurs nulles) : 6137.231
Moyenne globale de Rejets CO2 (en excluant les valeurs égales à zéro) : 158.495
Moyenne globale de Cout Energie : 561.465
```

Suivant on prépare essentiellement les données du catalogue pour la jointure avec les données combinées, en les convertissant en majuscules pour faciliter la correspondance avec les données précédemment traitées.

Nous avons calculé les moyennes globales pour les colonnes "Bonus/Malus", "Rejets CO2" et "Cout Energie", en excluant les valeurs nulles ou égales à zéro. Les moyennes sont formatées avec trois chiffres après la virgule et enregistrées dans des variables. Enfin, les moyennes globales sont affichées avec leur libellé respectif.

## Jointure avec Catalogue

La première étape dans la construction du résultat MapReduce consiste à effectuer une opération de jointure avec le fichier CSV "Catalogue.csv". Les données de ce fichier sont importées dans une

Spark. Avant la jointure, la fonction `upper()` est utilisée pour convertir les valeurs de la colonne "marque" en majuscules, standardisant ainsi leur format. Cette étape de standardisation est effectuée pour faciliter la jointure ultérieure avec d'autres données où la colonne "marque" peut être présentée dans différents formats ou casse. La nouvelle colonne "marque\_upper" ainsi créée servira à cette fin lors de la jointure avec le DataFrame «`merged_data`» où la colonne "marque" est également en majuscules.

```
# Left join entre catalog_data et merged_data par la colonne "Marque"
joined_data = catalog_data.join(merged_data, catalog_data["marque_upper"] == merged_data["Marque"], "left").drop(merged_data["Marque"])

# Supprimer la colonne "marque_upper"
joined_data = joined_data.drop("marque_upper")

joined_data.show()
```

On effectue une jointure de type "left join" entre les DataFrames « catalog\_data » et « merged\_data » sur la colonne "Marque\_upper" du DataFrame catalog\_data et la colonne "Marque" du DataFrame « merged\_data ». La jointure vise à combiner les données des deux DataFrames en fonction de la correspondance des valeurs dans ces colonnes. Une fois la jointure effectuée, la colonne "Marque\_upper" est supprimée du DataFrame résultant pour éviter la redondance. Enfin, les données résultantes sont affichées pour inspection.

```
Entrée [1]: # Remplacer les valeurs NULL par les moyennes globales
joined_data = joined_data.fillna({
    "Moyenne Bonus/Malus en euro": global_avg_bonus_malus,
    "Moyenne Rejets CO2 g/km": global_avg_co2_emission,
    "Cout Energie Moyen en euro": global_avg_energy_cost
})

# Afficher les données résultantes
joined_data.show()
```

| marque                    | nom puissance          | longueur nbPlaces nbPortes couleur occasion prix Moyenne Bonus/Malus en euro Moyenne Rejets CO2 g/km Cout Energie Moyen en euro |
|---------------------------|------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Volvo                     | S80 T6 272 très longue | 5  5  blanc  false 50500  0.0                                                                                                   |
| Volvo                     | S80 T6 272 très longue | 5  5  noir  false 50500  0.0                                                                                                    |
| Volvo                     | S80 T6 272 très longue | 5  5  rouge  false 50500  0.0                                                                                                   |
| Volvo                     | S80 T6 272 très longue | 5  5  gris  true 35350  0.0                                                                                                     |
| Volvo                     | S80 T6 272 très longue | 5  5  bleu  true 35350  0.0                                                                                                     |
| Volvo                     | S80 T6 272 très longue | 5  5  gris  false 50500  0.0                                                                                                    |
| Volvo                     | S80 T6 272 très longue | 5  5  bleu  false 50500  0.0                                                                                                    |
| Volvo                     | S80 T6 272 très longue | 5  5  rouge  true 35350  0.0                                                                                                    |
| Volvo                     | S80 T6 272 très longue | 5  5  blanc  true 35350  0.0                                                                                                    |
| Volvo                     | S80 T6 272 très longue | 5  5  noir  true 35350  0.0                                                                                                     |
| Volkswagen Touran 2.0 FSI | 150  longue            | 7  5  rouge  false 27340  -6000.0                                                                                               |
| 32.8                      | 96.0  longue           | 7  5  gris  true 19138  -6000.0                                                                                                 |
| 32.8                      | 96.0  longue           | 7  5  bleu  true 19138  -6000.0                                                                                                 |
| 32.8                      | 96.0  longue           | 7  5  gris  false 27340  -6000.0                                                                                                |
| 32.8                      | 96.0  longue           | 7  5  bleu  false 27340  -6000.0                                                                                                |
| 32.8                      | 96.0  longue           | 7  5  blanc  true 19138  -6000.0                                                                                                |
| 32.8                      | 96.0  longue           | 7  5  noir  true 19138  -6000.0                                                                                                 |
| Volkswagen Touran 2.0 FSI | 150  longue            | 7  5  rouge  true 19138  -6000.0                                                                                                |

La dernière étape du construction du résultat du traitement MapReduce consiste à remplacer les valeurs nulles dans le DataFrame « joined\_data » par les moyennes globales calculées précédemment pour les colonnes "Moyenne Bonus/Malus en euro", "Moyenne Rejets CO2 g/km" et "Cout Energie Moyen en euro". Nous utilisons la méthode fillna() pour spécifier les valeurs à utiliser pour remplir les valeurs nulles dans chaque colonne respective. Ensuite, il affiche les données résultantes du DataFrame.

```
Entrée [ 1]: # Compter le nombre de lignes
nombre_de_lignes = joined_data.count()
print("Nombre total de lignes dans joined_data : ", nombre_de_lignes)
[Stage 126:>   (0 + 1) / 1]
Nombre total de lignes dans joined_data : 270

Entrée [ 1]: # Afficher la totalité des données
joined_data.show(270)
```

| marque | nom    | puissance       | longueur        | nbPlaces | nbPortes | couleur | occasion | prix  | Moyenne Bonus/Malus en euro | Moyenne |
|--------|--------|-----------------|-----------------|----------|----------|---------|----------|-------|-----------------------------|---------|
| Volvo  | S80 T6 | 272 très longue | 5               | 5        | blanc    | false   | 50500    |       | 0.0                         | 0.0     |
| Volvo  | S80 T6 | 72.727          | 272 très longue | 5        | 5        | noir    | false    | 50500 |                             | 0.0     |
| Volvo  | S80 T6 | 72.727          | 272 très longue | 5        | 5        | rouge   | false    | 50500 |                             | 0.0     |
| Volvo  | S80 T6 | 72.727          | 272 très longue | 5        | 5        | gris    | true     | 35350 |                             | 0.0     |
| Volvo  | S80 T6 | 72.727          | 272 très longue | 5        | 5        | bleu    | true     | 35350 |                             | 0.0     |
| Volvo  | S80 T6 | 72.727          | 272 très longue | 5        | 5        | gris    | false    | 50500 |                             | 0.0     |
| Volvo  | S80 T6 | 72.727          | 272 très longue | 5        | 5        | bleu    | false    | 50500 |                             | 0.0     |

Nous calculons le nombre total de lignes dans le DataFrame « joined\_data » en utilisant la méthode count() afin d'avoir une idée sur la dimension du dataframe. Ensuite, il affiche ce nombre total de lignes ainsi que la totalité des données du DataFrame avec la méthode show() en spécifiant le nombre maximum de lignes à afficher, soit 270 (nombre de ligne de le dataframe « joinded\_data ») dans ce cas. Cela permet de visualiser l'ensemble des données pour la vérification.

```
Entrée [ ]: # Définir le chemin de destination du fichier CSV
output_path = "/MBDS_Projet/resultat_Catalogue_CO2.csv"

# Exporter les données résultantes au format CSV
joined_data.write.csv(output_path, mode="overwrite", header=True)
```

et on arrête la session Spark, ce qui libère les ressources utilisées par Spark et termine l'exécution du programme.

```
Entrée [ ]: # Arrêt de la session Spark
spark.stop()
```

définit le chemin de destination du fichier CSV résultant en HDFS. Ensuite, il exporte les données résultantes du DataFrame « joined\_data » au format CSV en utilisant la méthode write.csv(). Il spécifie le mode de réécriture (mode="overwrite") pour écraser le fichier s'il existe déjà et inclut les noms de colonnes en tant qu'en-têtes (header=True).

- Charger les données dans le répertoire local de la machine virtuelle :

```
Entrée [27]: # Exporter les données au format CSV localement
joined_data.to_csv("resultat_Catalogue_CO2.csv", index=False)
```

Vous pouvez accéder au fichier "resultat\_Catalogue\_CO2.csv" dans le rendu du projet, ou bien vous pouvez également le télécharger directement en utilisant le lien suivant : [https://drive.google.com/file/d/1bfh1\\_5auKf45xuIR2oyDjYes-AfUHyLY/view?usp=sharing](https://drive.google.com/file/d/1bfh1_5auKf45xuIR2oyDjYes-AfUHyLY/view?usp=sharing)

On arrête la session Spark, ce qui libère les ressources utilisées par Spark et termine l'exécution du programme.

```
Entrée [42]: # Arrêt de la session Spark
spark.stop()
```

Nous exportons également le résultat du traitement MapReduce vers le répertoire local de la machine virtuelle sous le nom de fichier CSV "rезультат\_Catalogue\_CO2.csv". Vous pouvez y

accéder soit via le rendu du projet PFA, soit en utilisant le lien Google Drive suivant :  
[https://drive.google.com/file/d/1bfh1\\_5auKf45xuIR2oyDjYes-AfUHyLY/view?usp=sharing](https://drive.google.com/file/d/1bfh1_5auKf45xuIR2oyDjYes-AfUHyLY/view?usp=sharing)

Enfin, une fois l'exportation terminée, il arrête la session Spark en appelant la méthode spark.stop(). Le fichier "resultat\_Catalogue\_CO2.csv" sera intégré dans le datalake Hive dans la prochaine étape.

Voici un aperçu du résultat du traitement MapReduce dans l'image ci-dessous :

| A          | B              | C               | D        | E        | F        | G       | H        | I     | J                           | K                       | L                          |
|------------|----------------|-----------------|----------|----------|----------|---------|----------|-------|-----------------------------|-------------------------|----------------------------|
| marque     | nom            | puissance       | longueur | nbPlaces | nbPortes | couleur | occasion | prix  | Moyenne Bonus/Malus en euro | Moyenne Rejets CO2 g/km | Cout Energie Moyen en euro |
| Volvo      | S80 T6         | 272 très longue | 5        | 5 blanc  | false    | 50500   |          | 0     | 42,455                      | 72,727                  |                            |
| Volvo      | S80 T6         | 272 très longue | 5        | 5 noir   | false    | 50500   |          | 0     | 42,455                      | 72,727                  |                            |
| Volvo      | S80 T6         | 272 très longue | 5        | 5 rouge  | false    | 50500   |          | 0     | 42,455                      | 72,727                  |                            |
| Volvo      | S80 T6         | 272 très longue | 5        | 5 gris   | true     | 35350   |          | 0     | 42,455                      | 72,727                  |                            |
| Volvo      | S80 T6         | 272 très longue | 5        | 5 bleu   | true     | 35350   |          | 0     | 42,455                      | 72,727                  |                            |
| Volvo      | S80 T6         | 272 très longue | 5        | 5 gris   | false    | 50500   |          | 0     | 42,455                      | 72,727                  |                            |
| Volvo      | S80 T6         | 272 très longue | 5        | 5 bleu   | false    | 50500   |          | 0     | 42,455                      | 72,727                  |                            |
| Volvo      | S80 T6         | 272 très longue | 5        | 5 rouge  | true     | 35350   |          | 0     | 42,455                      | 72,727                  |                            |
| Volvo      | S80 T6         | 272 très longue | 5        | 5 blanc  | true     | 35350   |          | 0     | 42,455                      | 72,727                  |                            |
| Volvo      | S80 T6         | 272 très longue | 5        | 5 noir   | true     | 35350   |          | 0     | 42,455                      | 72,727                  |                            |
| Volkswagen | Touran 2.0 FSI | 150 longue      | 7        | 5 rouge  | false    | 27340   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Touran 2.0 FSI | 150 longue      | 7        | 5 gris   | true     | 19138   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Touran 2.0 FSI | 150 longue      | 7        | 5 bleu   | true     | 19138   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Touran 2.0 FSI | 150 longue      | 7        | 5 gris   | false    | 27340   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Touran 2.0 FSI | 150 longue      | 7        | 5 bleu   | false    | 27340   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Touran 2.0 FSI | 150 longue      | 7        | 5 blanc  | true     | 19138   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Touran 2.0 FSI | 150 longue      | 7        | 5 noir   | true     | 19138   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Touran 2.0 FSI | 150 longue      | 7        | 5 noir   | false    | 27340   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Touran 2.0 FSI | 150 longue      | 7        | 5 bleu   | true     | 19138   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Touran 2.0 FSI | 150 longue      | 7        | 5 gris   | true     | 19138   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Touran 2.0 FSI | 150 longue      | 7        | 5 blanc  | false    | 27340   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Polo 1.2 6V    | 55 courte       | 5        | 3 blanc  | true     | 8540    |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Polo 1.2 6V    | 55 courte       | 5        | 3 blanc  | false    | 12200   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Polo 1.2 6V    | 55 courte       | 5        | 3 noir   | false    | 12200   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Polo 1.2 6V    | 55 courte       | 5        | 3 noir   | true     | 8540    |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Polo 1.2 6V    | 55 courte       | 5        | 3 bleu   | true     | 8540    |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Polo 1.2 6V    | 55 courte       | 5        | 3 bleu   | false    | 12200   |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Polo 1.2 6V    | 55 courte       | 5        | 3 rouge  | true     | 8540    |          | -6000 | 32.8                        | 96                      |                            |
| Volkswagen | Polo 1.2 6V    | 55 courte       | 5        | 3 rouge  | false    | 12200   |          | -6000 | 32.8                        | 96                      |                            |

## 7. Finalisation de la datalake et chargement du fichier résultat MapReduce

### 7.1 Chargement du résultat du traitement MapReduce dans la datalake Hive

Comme mentionné précédemment, une fois que le traitement MapReduce est terminé, le résultat est exporté vers HDFS dans le répertoire « MBDS\_Projet », comme illustré ci-dessous :

```
vagrant@oracle-21c-vagrant vagrant]$ hdfs dfs -ls /MBDS_Projet
Found 3 items
rw-r--r-- 1 vagrant supergroup 38916 2024-04-28 00:49 /MBDS_Projet/C02.csv
rw-r--r-- 1 vagrant supergroup 14114 2024-04-28 00:50 /MBDS_Projet/Catalogue.csv
rwxr-xr-x - vagrant supergroup 0 2024-04-28 01:52 /MBDS_Projet/resultat_Catalogue_CO2.csv
vagrant@oracle-21c-vagrant vagrant]$
```

La prochaine étape implique la création d'une table Hive qui servira à stocker les données issues du fichier « resultat\_Catalogue\_CO2.csv ».

Ce script SQL crée une table Hive appelée "resultat\_catalogue\_co2". Cette table comporte plusieurs colonnes pour stocker des informations sur les véhicules, telles que la marque, le nom du modèle, la puissance, la longueur, le nombre de places, le nombre de portes, la couleur, l'indicateur d'occasion, le prix, la moyenne des bonus/malus écologiques, la moyenne des rejets de CO2 et le coût énergétique moyen. La table est configurée pour utiliser un format de ligne délimité avec des champs séparés par des virgules. De plus, la première ligne du fichier est ignorée, supposément l'en-tête des colonnes.

La commande ci-dessous est utilisée pour charger les données situées dans le fichier « résultat\_Catalogue\_CO2.csv », qui est stocké dans le chemin '/MBDS\_Projet/' de HDFS, dans la table Hive « resultat\_catalogue\_co2 ». L'option OVERWRITE indique que si la table « résultat\_catalogue\_co2 » existe déjà, ses données seront écrasées par les nouvelles données chargées à partir du fichier CSV.

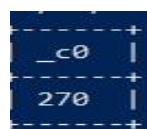
```
0: jdbc:hive2://localhost:10000>
0: jdbc:hive2://localhost:10000>
0: jdbc:hive2://localhost:10000> LOAD DATA INPATH '/MBDS_Projet/resultat_Catalogue_C02.csv' OVERWRITE INTO TABLE resultat_catalogue_c02;
```

À présent, nous allons vérifier que les données ont été entièrement et correctement chargées dans la table Hive.

```
0: jdbc:hive2://localhost:10000> SELECT * from resultat_catalogue_co2;
```

| resultat_catalogue_co2.marque   | resultat_catalogue_co2.nom  | resultat_catalogue_co2.longueur          | resultat_catalogue_co2.puissance          | resultat_catalogue_co2.longueur           | resultat_catalogue_co2.replace            | resultat_catalogue_co2.nb_replaces        | resultat_catalogue_co2.cout_energie_moyen | resultat_catalogue_co2.couleur |
|---------------------------------|-----------------------------|------------------------------------------|-------------------------------------------|-------------------------------------------|-------------------------------------------|-------------------------------------------|-------------------------------------------|--------------------------------|
| resultat_catalogue_co2.occasion | resultat_catalogue_co2.prix | resultat_catalogue_co2.moyenne_bon_malus | resultat_catalogue_co2.moyenne_rejets_co2 | resultat_catalogue_co2.moyenne_rejets_co2 | resultat_catalogue_co2.cout_energie_moyen | resultat_catalogue_co2.cout_energie_moyen | resultat_catalogue_co2.cout_energie_moyen | resultat_catalogue_co2.couleur |
| marque                          | nom                         | NULL                                     | longueur                                  | NULL                                      | NULL                                      | NULL                                      | NULL                                      | couleur                        |
| VOLVO                           | T6                          | 272                                      | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | blanc                          |
| Volvo                           | S80 T6                      | 272                                      | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | noir                           |
| false                           | 580 60                      | 7346.0                                   | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | rouge                          |
| Volvo                           | T6                          | 272                                      | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | gris                           |
| false                           | 580 60                      | 7346.0                                   | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | bleu                           |
| Volvo                           | S80 T6                      | 272                                      | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | gris                           |
| true                            | 35550                       | 7346.0                                   | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | blanc                          |
| Volvo                           | T6                          | 272                                      | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | noir                           |
| true                            | 35550                       | 7346.0                                   | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | rouge                          |
| Volvo                           | S80 T6                      | 272                                      | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | bleu                           |
| false                           | 580 60                      | 7346.0                                   | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | gris                           |
| Volvo                           | T6                          | 272                                      | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | rouge                          |
| false                           | 580 60                      | 7346.0                                   | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | bleu                           |
| Volvo                           | S80 T6                      | 272                                      | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | gris                           |
| true                            | 35550                       | 7346.0                                   | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | blanc                          |
| Volvo                           | T6                          | 272                                      | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | noir                           |
| true                            | 35550                       | 7346.0                                   | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | rouge                          |
| Volvo                           | S80 T6                      | 272                                      | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | bleu                           |
| false                           | 580 60                      | 7346.0                                   | très longue                               | 42.455                                    | 5                                         | 72.727                                    | 5                                         | gris                           |
| Volkswagen                      | Touran 2.0 FSI              | 158                                      | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | rouge                          |
| false                           | 27340                       | 158                                      | 3528.571                                  | 47.429                                    | 7                                         | 96.0                                      | 5                                         | blanc                          |
| Volkswagen                      | Touran 2.0 FSI              | 158                                      | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | noir                           |
| true                            | 19138                       | 3528.571                                 | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | gris                           |
| false                           | 19138                       | 3528.571                                 | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | bleu                           |
| Volkswagen                      | Touran 2.0 FSI              | 158                                      | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | gris                           |
| true                            | 27340                       | 3528.571                                 | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | bleu                           |
| Volkswagen                      | Touran 2.0 FSI              | 158                                      | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | blanc                          |
| false                           | 19138                       | 3528.571                                 | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | noir                           |
| Volkswagen                      | Touran 2.0 FSI              | 158                                      | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | rouge                          |
| true                            | 19138                       | 3528.571                                 | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | blanc                          |
| Volkswagen                      | Touran 2.0 FSI              | 158                                      | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | noir                           |
| false                           | 27340                       | 3528.571                                 | longue                                    | 47.429                                    | 7                                         | 96.0                                      | 5                                         | rouge                          |
| Volkswagen                      | Polo 1.2 6V                 | 55                                       | courte                                    | 47.429                                    | 5                                         | 96.0                                      | 3                                         | blanc                          |
| false                           | 19138                       | 3528.571                                 | courte                                    | 47.429                                    | 5                                         | 96.0                                      | 3                                         | noir                           |

**0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) from resultat\_catalogue\_co2;**



Comme vous pouvez le remarquer, toutes les données du fichier "resultat\_Catalogue\_CO2.csv" issues du traitement MapReduce ont été chargées avec succès dans le datalake.

## 7.2 Exploration des données du catalogue de la table Hive « resultat\_catalogue\_co2 »

| Catalogue.csv : catalogue de véhicules |            |                              |                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------------------------|------------|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Attribut                               | Type       | Description                  | Domaine de valeurs                                                                                                                                                                                                                                                                                                                                                                                |
| Marque                                 | caractères | Nom de la marque du véhicule | Audi, BMW, Dacia, Daihatsu, Fiat, Ford, Honda, Hyundai, Jaguar, Kia, Lancia, Mercedes , Mini, Nissan, Peugeot, Renault, Saab, Seat, Skoda, Volkswagen, Volvo                                                                                                                                                                                                                                      |
| Nom                                    | caractères | Nom du modèle de véhicule    | S80 T6, Touran 2.0 FSI, Polo 1.2 6V, New Beatle 1.8, Golf 2.0 FSI, Superb 2.8 V6, Toledo 1.6, 9.3 1.8T, Vel Satis 3.5 V6, Megane 2.0 16V, Laguna 2.0T, Espace 2.0T, 1007 1.4, Primera 1.6, Maxima 3.0 V6, Almera 1.8, Copper 1.6 16V, S500, A200, Ypsilon 1.4 16V, Picanto 1.1, X-Type 2.5 V6, Matrix 1.6 FR-V 1.7, Mondeo 1.8, Croma 2.2, Cuore 1.0, Logan 1.6 MPI, M5, 120i, A3 2.0 FSI, A2 1.4 |
| Puissance                              | numérique  | Puissance en chevaux Din     | [55, 507]                                                                                                                                                                                                                                                                                                                                                                                         |
| Longueur                               | catégoriel | Catégorie de longueur        | courte, moyenne, longue, très longue                                                                                                                                                                                                                                                                                                                                                              |
| NbPlaces                               | numérique  | Nombre de places             | [5, 7]                                                                                                                                                                                                                                                                                                                                                                                            |
| NbPortes                               | numérique  | Nombre de portes             | [3, 5]                                                                                                                                                                                                                                                                                                                                                                                            |
| Couleur                                | catégoriel | Couleur                      | blanc, bleu, gris, noir, rouge                                                                                                                                                                                                                                                                                                                                                                    |
| Occasion                               | booléen    | Véhicule d'occasion ?        | true, false                                                                                                                                                                                                                                                                                                                                                                                       |
| Prix                                   | numérique  | Prix de vente en euros       | [7500, 101300]                                                                                                                                                                                                                                                                                                                                                                                    |

Cette section vise à confirmer que les données du catalogue du concessionnaire incluses dans la table Hive respectent le dictionnaire de données fourni ci-dessus.

0: jdbc:hive2://localhost:10000> DESC from resultat\_catalogue\_co2;

| col_name            | data_type | comment |
|---------------------|-----------|---------|
| marque              | string    |         |
| nom                 | string    |         |
| puissance           | int       |         |
| longueur            | string    |         |
| nbplaces            | int       |         |
| nbportes            | int       |         |
| couleur             | string    |         |
| occasion            | boolean   |         |
| prix                | int       |         |
| moyenne_bonus_malus | double    |         |
| moyenne_rejets_co2  | double    |         |
| cout_energie_moyen  | double    |         |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT marque from resultat\_catalogue\_co2;

| marque     |
|------------|
| Audi       |
| BMW        |
| Dacia      |
| Daihatsu   |
| Fiat       |
| Ford       |
| Honda      |
| Hyundai    |
| Jaguar     |
| Kia        |
| Lancia     |
| Mercedes   |
| Mini       |
| Nissan     |
| Peugeot    |
| Renault    |
| Saab       |
| Seat       |
| Skoda      |
| Volkswagen |
| Volvo      |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT nom from resultat\_catalogue\_co2;

| nom              |
|------------------|
| 1007 1.4         |
| 120i             |
| 9.3 1.8T         |
| A2 1.4           |
| A200             |
| A3 2.0 FSI       |
| Almera 1.8       |
| Copper 1.6 16V   |
| Croma 2.2        |
| Cuore 1.0        |
| Espace 2.0T      |
| FR-V 1.7         |
| Golf 2.0 FSI     |
| Laguna 2.0T      |
| Logan 1.6 MPI    |
| M5               |
| Matrix 1.6       |
| Maxima 3.0 V6    |
| Megane 2.0 16V   |
| Mondeo 1.8       |
| New Beetle 1.8   |
| Picanto 1.1      |
| Polo 1.2 6V      |
| Primera 1.6      |
| S500             |
| S80 T6           |
| Superb 2.8 V6    |
| Toledo 1.6       |
| Touran 2.0 FSI   |
| Vel Satis 3.5 V6 |
| X-Type 2.5 V6    |
| Ypsilon 1.4 16V  |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT puissance from resultat\_catalogue\_co2;

| puissance |
|-----------|
| 55        |
| 58        |
| 65        |
| 75        |
| 90        |
| 102       |
| 103       |
| 109       |
| 110       |
| 115       |
| 125       |
| 135       |
| 136       |
| 147       |
| 150       |
| 165       |
| 170       |
| 193       |
| 197       |
| 200       |
| 245       |
| 272       |
| 306       |
| 507       |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT longueur from resultat\_catalogue\_co2;

| longueur    |
|-------------|
| courte      |
| longue      |
| moyenne     |
| très longue |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT nbportes from resultat\_catalogue\_co2;

| nbportes |
|----------|
| 3        |
| 5        |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT nbplaces from resultat\_catalogue\_co2;

| nbplaces |
|----------|
| 5        |
| 7        |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT couleur from resultat\_catalogue\_co2;

| couleur |
|---------|
| blanc   |
| bleu    |
| gris    |
| noir    |
| rouge   |

0: jdbc:hive2://localhost:10000> SELECT DISTINCT occasion from resultat\_catalogue\_co2;

| occasion |
|----------|
| false    |
| true     |

Comme vous pouvez le constater, la table respecte parfaitement les noms des colonnes, leurs types correspondants ainsi que les plages de valeurs spécifiées pour les données relatives au catalogue.

### 7.3 Construction du model d'analyse de la datalake

L'objectif est de réaliser une jointure entre les données des clients et les immatriculations. Pour cela, dans un premier temps, nous allons sélectionner les immatriculations des tables "client\_7\_ext" et "client\_12\_ext" à partir de la table "table\_ext\_immatriculations" dans une vue matérialisée.

0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) from client\_7\_ext;

```
+  
_c0 |  
+  
99980 |  
+
```

0: jdbc:hive2://localhost:10000> SELECT COUNT(\*) from client\_12\_ext;

```
+-----+  
| _c0 |  
+-----+  
| 99977 |  
+-----+
```

En théorie, si nous fusionnons les deux tables et que les immatriculations des clients sont uniques entre elles, nous devrions obtenir 199957 immatriculations.

Cette commande crée une vue matérialisée nommée "vue\_ext\_immatriculations\_intermediaire". Cette vue sélectionne toutes les colonnes de la table "table\_ext\_immatriculations" où les

immatriculations sont présentes dans les résultats de deux sous-requêtes. Ces sous-requêtes récupèrent les immatriculations de deux tables distinctes : "client\_7\_ext" et "client\_12\_ext". En utilisant l'opérateur UNION, les immatriculations uniques provenant des deux tables sont combinées. Ainsi, la vue matérialisée contiendra toutes les immatriculations présentes dans la table "table\_ext\_immatriculations" qui correspondent aux immatriculations des clients présentes dans les tables "client\_7\_ext" ou "client\_12\_ext".

```
1 row selected (0.385 seconds)
0: jdbc:hive2://localhost:10000> SELECT * from vue_ext_immatriculations_intermediaire LIMIT 10;
```

| 0 AS 74 | 5 | BMW     | 5 | 1201             | bleu  | 150 | true  | moyenne     | 25060 |
|---------|---|---------|---|------------------|-------|-----|-------|-------------|-------|
| 0 BZ 21 | 5 | Audi    | 5 | A2 1.4           | noir  | 75  | false | courte      | 18310 |
| 0 CQ 77 | 5 | Peugeot | 5 | 1007 1.4         | noir  | 75  | true  | courte      | 9625  |
| 0 DQ 29 | 5 | Peugeot | 5 | 1007 1.4         | gris  | 75  | true  | courte      | 9625  |
| 0 FP 65 | 5 | Volvo   | 5 | S80 T6           | rouge | 272 | false | tres longue | 50500 |
| 0 JO 29 | 5 | Renault | 5 | Vel Satis 3.5 V6 | gris  | 245 | false | tres longue | 49200 |
| 0 MD 67 | 5 | BMW     | 5 | M5               | blanc | 507 | false | tres longue | 94800 |
| 0 ME 78 | 5 | Audi    | 5 | A2 1.4           | gris  | 75  | false | courte      | 18310 |
| 0 NK 32 | 5 | BMW     | 5 | M5               | blanc | 507 | false | tres longue | 94800 |
| 0 OX 10 | 5 | Renault | 5 | Megane 2.0 16V   | blanc | 135 | true  | moyenne     | 15644 |

Comme vous pouvez le constater, nous examinons les données relatives aux immatriculations des clients présentes à la fois dans les tables "client\_7\_ext" et "client\_12\_ext".

```
1 row selected (0.214 seconds)
0: jdbc:hive2://localhost:10000> SELECT COUNT(*) from vue_ext_immatriculations_intermediaire;
```

| _c0    |
|--------|
| 199943 |

Lors de l'insertion des immatriculations des clients provenant des tables « client\_7\_ext » et « client\_12\_ext » dans la vue matérialisée « vue\_ext\_immatriculations\_intermediaire », qui regroupe l'ensemble des données d'immatriculation des clients, nous constatons une différence de nombres. Cela suggère qu'il existe des immatriculations communes entre les deux tables « client\_7\_ext » et « client\_12\_ext ». Nous devons maintenant vérifier cette hypothèse.

```
0: jdbc:hive2://localhost:10000> SELECT immatriculation, COUNT(*) AS nb_occurrences
```

FROM (

SELECT immatriculation FROM client\_7\_ext

UNION ALL

SELECT immatriculation FROM client\_12\_ext

) AS combined\_clients

GROUP BY immatriculation

HAVING COUNT(\*) > 1;

| immatriculation | nb_occurrences |
|-----------------|----------------|
| 186 MN 34       | 2              |
| 2298 CY 44      | 2              |
| 3014 OR 44      | 2              |
| 3786 ZM 27      | 2              |
| 3910 NF 86      | 2              |
| 3989 YB 64      | 2              |
| 5617 WE 28      | 2              |
| 5886 NW 27      | 2              |
| 7145 CH 89      | 2              |
| 7528 CX 55      | 2              |
| 8521 UD 59      | 2              |
| 8960 GM 31      | 2              |
| 9105 PG 41      | 2              |
| 9505 VT 49      | 2              |

Cette requête sélectionne les immatriculations présentes à la fois dans les tables « client\_7\_ext » et « client\_12\_ext » en les combinant dans une sous-requête avec l'opérateur UNION ALL. Ensuite, elle regroupe ces immatriculations en comptant le nombre d'occurrences de chaque immatriculation. La clause HAVING est utilisée pour filtrer les résultats afin de ne renvoyer que les immatriculations qui apparaissent plus d'une fois dans les deux tables combinées. Enfin, la requête renvoie les immatriculations avec leur nombre d'occurrences supérieur à 1. Cela permet d'identifier les immatriculations communes entre les deux tables.

Effectivement, il y a 14 immatriculations qui se répètent entre les tables « client\_7\_ext » et « client\_12\_ext », ce qui explique la différence observée précédemment entre les nombres. Il est donc essentiel de prendre cela en compte lors de la jointure entre les données clients et les immatriculations.

Ensuite, nous allons insérer les données des clients de la table "client\_7\_ext" en les associant aux données d'immatriculation de la vue matérialisée "vue\_ext\_immatriculations\_intermediaire" dans la table "model\_immatriculations\_clients". Cette table servira de modèle d'analyse pour le datalake que nous avons créé précédemment.

Cette commande crée une nouvelle table appelée "model\_immatriculations\_clients". Les données de cette table sont extraites de la vue matérialisée "vue\_ext\_immatriculations\_intermediaire" (alias "i") et de la table "client\_7\_ext" (alias "c7"). La jointure est réalisée en reliant les données des deux sources sur la colonne "immatriculation". Les colonnes de la vue matérialisée sont incluses dans la nouvelle table, ainsi que certaines colonnes spécifiques de la table "client\_7\_ext" telles que l'âge, le sexe, le taux, la situation familiale, le nombre d'enfants à charge et la deuxième voiture.

```
0: jdbc:hive2://localhost:10000> SELECT * from model_immatriculations_clients LIMIT 10;
```

| Statistiques détaillées sur les véhicules immatriculés |           |                              |                   |                              |       |                          |                        |                                    |                              |
|--------------------------------------------------------|-----------|------------------------------|-------------------|------------------------------|-------|--------------------------|------------------------|------------------------------------|------------------------------|
| Identité du conducteur                                 |           | Caractéristiques du véhicule |                   | Informations administratives |       | Statistiques de sécurité |                        | Statistiques de performance        |                              |
| Prénom                                                 | Nom       | Type                         | Modèle            | Age                          | Sexe  | Nombre d'occasions       | Nombre de km parcourus | Nombre de voitures dans la famille | Nombre de personnes assurées |
| Paul                                                   | Jones     | Homme                        | Audi A2           | 21                           | Homme | 1                        | 18310 km               | 1                                  | 5 personnes                  |
| John                                                   | Doe       | Femme                        | Peugeot 1007      | 22                           | Femme | 1                        | 9625 km                | 1                                  | 5 personnes                  |
| Mary                                                   | Smith     | Homme                        | Peugeot 207       | 23                           | Homme | 1                        | 9625 km                | 1                                  | 5 personnes                  |
| David                                                  | Wilson    | Femme                        | BMW M5            | 24                           | Femme | 1                        | 94800 km               | 1                                  | 5 personnes                  |
| Sarah                                                  | Johnson   | Homme                        | Renault Vel Satis | 25                           | Homme | 1                        | 49200 km               | 1                                  | 5 personnes                  |
| Emily                                                  | Anderson  | Femme                        | BMW 3 Series      | 26                           | Femme | 1                        | 94800 km               | 1                                  | 5 personnes                  |
| Michael                                                | Williams  | Homme                        | Volkswagen Polo   | 27                           | Homme | 1                        | 12200 km               | 1                                  | 5 personnes                  |
| Alexander                                              | Greenwood | Femme                        | Volvo S80         | 28                           | Femme | 1                        | 50500 km               | 1                                  | 5 personnes                  |
| Olivia                                                 | Perry     | Homme                        | Jaguar X-Type     | 29                           | Homme | 1                        | 25970 km               | 1                                  | 5 personnes                  |

```
0: jdbc:hive2://localhost:10000> SELECT COUNT(*) FROM model.immatriculations.clients;
```

```
+-----+  
| _c0 |  
+-----+  
| 99980 |  
+-----+  
1 row selected
```

Comme vous pouvez le remarquer, toutes les données des clients de la table "client\_7\_ext" avec leurs données d'immatriculation correspondantes ont été insérées avec succès.

Nous allons maintenant insérer les données des clients provenant de la table « client\_12\_ext » avec leurs immatriculations correspondantes tout en garantissant l'unicité de la colonne « immatriculation » dans le modèle d'analyse existant.

```
0 rows affected (12.234 seconds)
0: jdbc:hive2://localhost:10000> INSERT INTO model_immatriculations_clients
...>     SELECT i.*, c12.age, c12.sexe, c12.taux, c12.situationFamiliale, c12.nbEnfantsAcharge, c12.deuxiemevoiture
...>     FROM vue_ext_immatriculations_intermediaire i
...>     JOIN client_12_ext c12 ON i.immatriculation = c12.immatriculation
...>     WHERE i.immatriculation NOT IN (
...>         SELECT immatriculation FROM model_immatriculations_clients
...>     );
...>
```

Cette commande insère des données dans la table « model\_immatriculations\_clients » en sélectionnant toutes les colonnes de la « vue\_ext\_immatriculations\_intermediaire » (i.\*), ainsi que certaines colonnes spécifiques de la table « client\_12\_ext ». La jointure se fait sur la colonne immatriculation entre la vue et la table. La clause WHERE exclut les immatriculations qui existent déjà dans la table model\_immatriculations\_clients (ceux des clients provenant de la table «client\_7\_ext »), assurant ainsi l'unicité des données immatriculées dans la nouvelle table.

```
1 row selected (32.087 seconds)
0: jdbc:hive2://localhost:10000> SELECT COUNT(*) from model_immatriculations_clients;
...>
```

| _c0    |
|--------|
| 199943 |

Comme vous pouvez le remarquer, le nombre d'enregistrements correspond à la dimension lors de la vue « vue\_ext\_immatriculations\_intermediaire », qui inclut les données d'immatriculation correspondant de l'ensemble des clients. Vérifions qu'il n'y a pas d'occurrence au niveau des immatriculations dans le modèle d'analyse.

```
0: jdbc:hive2://localhost:10000> SELECT immatriculation, COUNT(*) AS nb_occurrences
...> FROM model_immatriculations_clients
...> GROUP BY immatriculation
...> HAVING COUNT(*) > 1;
```

| immatriculation | nb_occurrences |
|-----------------|----------------|
|                 |                |

Comme vous pouvez le constater, nous avons établi notre modèle d'analyse qui stockera toutes les données des clients avec leurs immatriculations correspondantes, en respectant le principe

d'unicité des immatriculations. Il ne reste plus qu'à supprimer la vue matérialisée avec la commande suivante :

```
0: jdbc:hive2://localhost:10000> DROP MATERIALIZED VIEW IF EXISTS  
vue_ext_immatriculations_intermediaire;
```

## 8. Analyse des Données du sous projet BDA/DL par les Techniques de Data Mining, Machine Learning et Deep Learning:

Cette section vise à expliquer les différentes étapes et approches utilisées depuis l'analyse des données du catalogue du concessionnaire jusqu'à la prédiction de la catégorie des voiture à partir des données clients. Vous pouvez également consulter le notebook documenté de manière détaillée soit dans le rapport PFA ou via le lien Google Drive suivant :

[https://drive.google.com/file/d/1ojW-f3dVhHYRpQ\\_i5PFRXesLZi1ch28V/view?usp=sharing](https://drive.google.com/file/d/1ojW-f3dVhHYRpQ_i5PFRXesLZi1ch28V/view?usp=sharing)

### 8.1 Analyse Exploratoire des données du catalogue du concessionnaire

Visant à assister le concessionnaire automobile dans sa quête pour mieux cibler les véhicules susceptibles d'intéresser sa clientèle. Cette phase constitue un pivot essentiel, où nous nous efforçons de tirer profit de ces techniques avancées d'analyse de données afin de découvrir des informations subtiles et significatives à partir des données fournies. Cette approche élaborée vise à répondre de manière précise et adaptée aux besoins et aux objectifs du client, en lui fournissant des insights précieux pour orienter ses décisions stratégiques.

Dans le présent rapport, nous détaillerons avec les méthodologies déployées pour cette analyse, mettant en lumière les choix opérés en matière de gestion et de traitement des données. Nous explorerons en profondeur les processus de Data Mining, de Machine Learning et de Deep Learning que nous avons orchestrés, soulignant notre démarche pour extraire des modèles de connaissances significatifs à partir des données disponibles. Cette analyse critique sera accompagnée d'une interprétation éclairée de ces modèles, dévoilant ainsi des insights pertinents et stratégiques pour notre client concessionnaire ,nos découvertes et nos recommandations basées sur cette analyse approfondie.

Nous avons suivi une méthodologie rigoureuse pour mener à bien notre analyse des données. Cela comprenait les étapes suivantes :

- **Installation des modules requis, Connexion et récupération des données de la base de données Hive:**

Nous avons installé les bibliothèques Python nécessaires, telles que PyHive, pandas, matplotlib, seaborn et scikit-learn, pour faciliter l'interfaçage avec les données et effectuer des analyses avancées. Nous nous sommes connectés à la base de données Hive locale et avons extrait les données de la table resultat\_catalogue\_co2 pour effectuer une analyse exploratoire approfondie par la suite.

#### ✓ Installation des modules nécessaire

- PyHive: interface Python pour les bases de données de type Apache Hive. Il permet aux développeurs d'interagir avec les données stockées dans Hive à partir de leur code Python.

```
!pip install pyhive
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pyhive in /home/vagrant/.local/lib/python3.9/site-packages (0.7.0)
Requirement already satisfied: thrift in /home/vagrant/.local/lib/python3.9/site-packages (from pyhive) (1.0.0)
Requirement already satisfied: python-dateutil in /home/vagrant/.local/lib/python3.9/site-packages (from pyhive) (2.8.2)
Requirement already satisfied: six>=1.5 in /home/vagrant/.local/lib/python3.9/site-packages (from python-dateutil>pyhive) (1.16.0)
```

- thrift\_sasl est une dépendance utilisée pour communiquer avec des systèmes distribués comme Apache Hive, Apache HBase, etc., qui utilisent le protocole Thrift.

```
!pip install thrift_sasl
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: thrift_sasl in /home/vagrant/.local/lib/python3.9/site-packages (0.4.3)
Requirement already satisfied: pure-sasl>=0.6.2 in /home/vagrant/.local/lib/python3.9/site-packages (from thrift_sasl) (0.6.2)
Requirement already satisfied: six>=1.13.0 in /home/vagrant/.local/lib/python3.9/site-packages (from thrift_sasl) (1.16.0)
Requirement already satisfied: thrift>=0.10.0 in /home/vagrant/.local/lib/python3.9/site-packages (from thrift_sasl) (0.20.0)
```

```
!pip install pandas
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in /home/vagrant/.local/lib/python3.9/site-packages (2.2.1)
Requirement already satisfied: numpy<2,>=1.22.4 in /home/vagrant/.local/lib/python3.9/site-packages (from pandas) (1.26.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /home/vagrant/.local/lib/python3.9/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /home/vagrant/.local/lib/python3.9/site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /home/vagrant/.local/lib/python3.9/site-packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in /home/vagrant/.local/lib/python3.9/site-packages (from python-dateutil>2.8.2>pandas) (1.16.0)
```

```
!pip install matplotlib
```

Il convient de souligner que toutes les données du datalake sont conformes au dictionnaire de données fourni, cela est rendu possible grâce au traitement effectué à l'aide de requêtes Hive avant de passer à l'analyse des données.

On se connecte à la base de données Hive locale, exécute une requête SQL pour récupérer les données de la table resultat\_catalogue\_co2, puis stocke ces données dans un DataFrame Pandas pour une manipulation ultérieure.

▼ Analyse Exploratoire

- On se connecte à la base de données Hive locale, exécute une requête SQL pour récupérer les données de la table résultat\_catalogue\_co2, puis stocke ces données dans un DataFrame Pandas pour une manipulation ultérieure.

```
from pyhive import hive
import pandas as pd

# Connexion à Hive
conn = hive.Connection(host='localhost', port=10000, username='hive')

# Création d'un curseur
cursor = conn.cursor()

# Exécution d'une requête SQL
cursor.execute('SELECT * FROM résultat_catalogue_co2')

# Récupération des noms de colonnes
columns = [desc[0] for desc in cursor.description]

# Récupération des résultats
results = cursor.fetchall()

# Crédit d'un DataFrame Pandas avec les résultats et les noms de colonnes
catalogue_data = pd.DataFrame(results, columns=columns)

catalogue_data
```

|   | resultat_catalogue_co2.marque | resultat_catalogue_co2.nom | resultat_catalogue_co2. |
|---|-------------------------------|----------------------------|-------------------------|
| 0 | Volvo                         | S80 T6                     |                         |
| 1 | Volvo                         | S80 T6                     |                         |
| 2 | Volvo                         | S80 T6                     |                         |
| 3 | Volvo                         | S80 T6                     |                         |

L'objectif est de comprendre la structure des données, leurs distributions et et d'explorer les relations qui les relient entre elles cela comprenait la génération d'histogrammes, de boîtes à moustaches, de diagrammes en violon et d'autres visualisations pour Explorer tous les types de variables, qu'elles soient numériques ou catégorielles.

Suivant est ce qui etait realisé :

- La liste des variables numériques a explorer extraite :

▼ Analyse des valeurs numériques

```
colonnes_a_selectionner = [
    'résultat_catalogue_co2.puissance',
    'résultat_catalogue_co2.nbpieces',
    'résultat_catalogue_co2.nbportes',
    'résultat_catalogue_co2.prix',
    'résultat_catalogue_co2.moyenne_bonus_malus',
    'résultat_catalogue_co2.moyenne_rejets_co2',
    'résultat_catalogue_co2.cout_énergie_moyen'
]

# Utilisation des colonnes sélectionnées dans la DataFrame
numeric_data = catalogue_data[colonnes_a_selectionner]
numeric_data
```

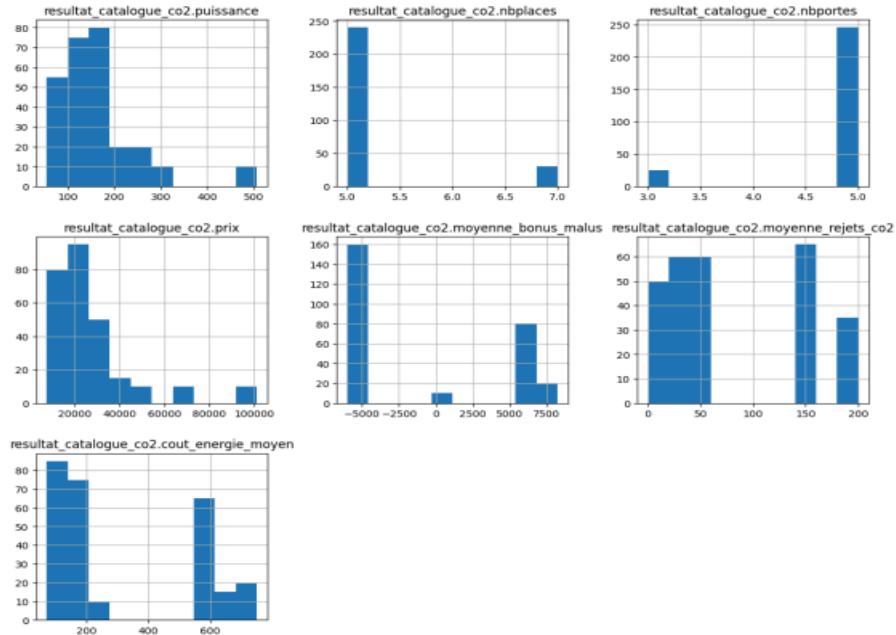
- Visualisations univariées :

Elles sont utiles pour comprendre la répartition des valeurs d'une variable, identifier les tendances, repérer les valeurs aberrantes et évaluer la forme de la distribution en utilisant des histogrammes, des diagrammes en boîte (boxplots) et des graphiques de densité pour représenter la distribution des variables numériques.

- Histogramme:

```
import matplotlib.pyplot as plt

# Effectuer une analyse univariée pour toutes les variables numériques avec une taille de figure plus grande
numeric_data.hist(figsize=(14, 12))
plt.suptitle("Distribution des variables numériques")
plt.show()
```



Chaque histogramme représente la fréquence des valeurs de la variable correspondante. Cette visualisation permet d'avoir un aperçu rapide de la distribution des données et peut aider à détecter d'éventuelles tendances ou anomalies.

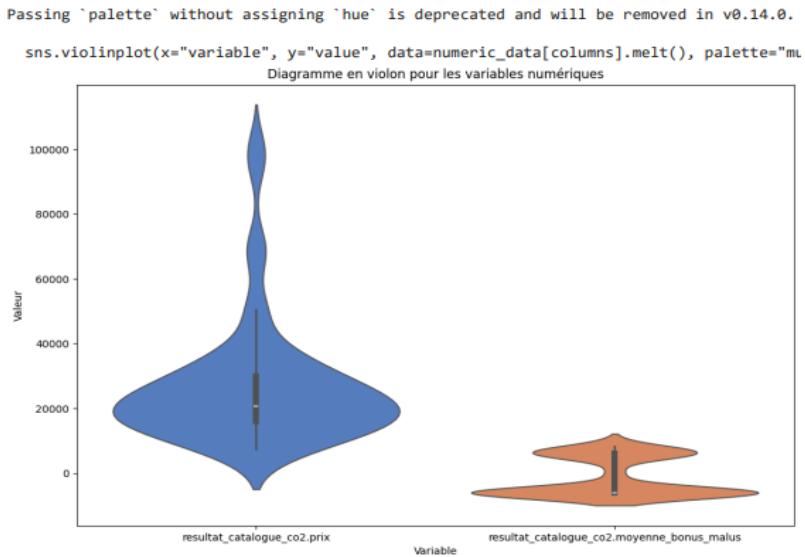
Les résultats révèlent que la majorité des véhicules ont une puissance comprise entre 100 et 200 chevaux, un prix situé entre 20 000 et 25 000 euros, cinq portes et cinq places, un bonus-malus moyen de -6000 euros, des émissions de CO<sub>2</sub> moyennes entre 0 et 50 g/km et un coût énergétique moyen compris entre 30 et 200 euros.

Dans de nombreux cas, les valeurs aberrantes sont perçues comme des points de données inhabituels ou des anomalies qui s'écartent considérablement de la tendance générale des données et peuvent être le résultat d'erreurs de mesure, de saisie incorrecte des données ou de comportements anormaux. Cependant, il existe des situations où ces valeurs ne sont pas des erreurs, mais plutôt des représentations valides de sous-groupes ou de variantes des données. Ce qui était constaté, Prenons l'exemple des voitures à trois portes dans l'ensemble de données sur le nombre de portes des véhicules. Bien que la majorité des voitures puissent avoir cinq portes, il existe une catégorie distincte de voitures à trois portes qui est parfaitement légitime sur le marché automobile. Dans ce cas, ces valeurs sont une sous-catégorie prévue des données. En reconnaissant ces sous-catégories comme des éléments significatifs de l'ensemble de données, on évite de les traiter comme des anomalies à exclure, ce qui pourrait fausser les conclusions de l'analyse. Au contraire, en les intégrant dans l'analyse, on obtient une compréhension plus riche et plus nuancée des données et des tendances qui les sous-tendent.

- Diagramme en violon:

```
import seaborn as sns
columns=['resultat_catalogue_co2.prix', 'resultat_catalogue_co2.moyenne_bonus_malus']
# Définir la taille de la figure
plt.figure(figsize=(12, 8))

# Tracer le diagramme en violon
sns.violinplot(x="variable", y="value", data=numeric_data[melt()], palette="muted")
plt.title("Diagramme en violon pour les variables numériques")
plt.xlabel("Variable")
plt.ylabel("Valeur")
plt.show()
```



Le diagramme en violon offre une visualisation plus détaillée de la forme de la distribution des données, y compris la médiane, les quartiles, les valeurs aberrantes potentielles, les pics, les creux et la densité de probabilité relative. On constate la distribution des valeurs pour chaque variable

numérique, avec les contours des violons représentant la densité de probabilité des données à différentes valeurs ce qui peut aider à identifier les zones de concentration des données et à détecter les schémas de regroupement ou de dispersion.

- Pour le graphe de violon du prix des véhicules :

La concentration importante des données entre 7500 et 35000 euros, avec un pic autour de 20000 euros, suggère que la plupart des véhicules achetés ont des prix dans cette fourchette. Cela peut correspondre à une gamme de voitures de différents types et de différentes marques qui sont populaires, accessibles sur le marché.

L'extension de la distribution jusqu'à 100000 euros indique la présence de véhicules plus coûteux, probablement des modèles haut de gamme ou des voitures de luxe, bien que leur nombre soit relativement faible par rapport aux véhicules de gamme moyenne.

- Pour le graphe de violon du bonus/malus moyen :

La concentration autour de +8000 et -6000 suggère que la plupart des véhicules ont un bonus/malus dans cette plage. Cela peut indiquer que les véhicules achetés sont majoritairement neufs ou récemment immatriculés, avec des niveaux de bonus/malus standard appliqués en fonction de leurs caractéristiques et de leurs émissions.

La faible concentration autour de 0 indique qu'il y a peu de véhicules avec un bonus/malus nul. Cela pourrait être dû à divers facteurs, tels les politiques gouvernementales favorisant les véhicules à faibles émissions ou une tendance des consommateurs à opter pour des véhicules plus respectueux de l'environnement.

En résumé, ces interprétations suggèrent que la plupart des véhicules achetés se situent dans une gamme de prix moyenne avec des bonus/malus standards, mais il existe également des variations avec des véhicules plus chers et des bonus/malus plus élevés ou plus bas, ce qui peut être influencé par des facteurs tels que la marque, le modèle, l'âge et les caractéristiques écologiques des véhicules.

- Statistiques descriptives :

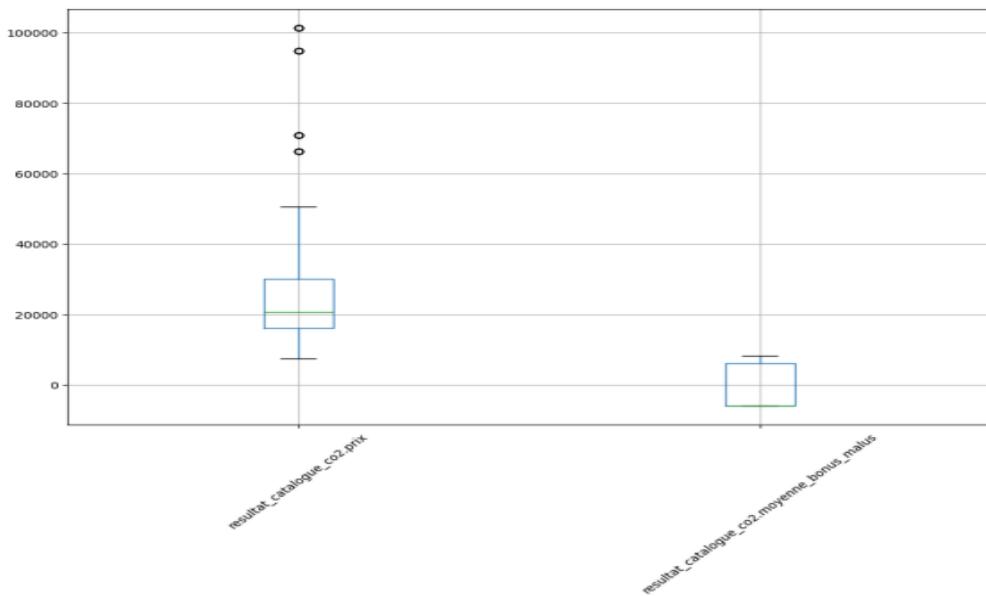
une vue d'ensemble des données en résumant leurs caractéristiques principales, telles que la moyenne, l'écart-type et les quartiles, ce qui permet de comprendre leur distribution. Ces mesures offrent des informations claires sur la tendance centrale, la dispersion et les valeurs extrêmes des

données, aidant ainsi à repérer les valeurs aberrantes potentielles et à prendre des décisions éclairées.

- la boîte à moustaches :

- On dessine un graphique qui nous permettra de visualiser la distribution des valeurs continues "prix" et "moyenne\_bonus\_malus", ainsi que les valeurs aberrantes éventuelles, les médianes, les quartiles, et la dispersion des données.

```
# Boites à moustaches pour les variables continues
catalogue_data[['resultat_catalogue_co2.prix','resultat_catalogue_co2.moyenne_bonus_malus']].boxplot(figsize=(12, 8))
plt.xticks(rotation=45)
plt.show()
```



Pour la variable *prix* :

La médiane de la distribution est d'environ 20000, ce qui signifie que la moitié des valeurs se situent en dessous de ce seuil et l'autre moitié au-dessus.

Les quartiles (Q1 et Q3) indiquent que 25% des données sont inférieures à 15000 et 75% des données sont inférieures à 30000, ce qui montre une certaine dispersion des valeurs.

La plage interquartile (IQR), qui va de Q1 à Q3, de 15000 à 30000 équivalente de  $30000 - 15000 = 15000$ , suggérant que la majorité des données sont concentrées dans cette fourchette.

Les valeurs aberrantes sont observées au-delà de 50000, avec des valeurs remarquables autour de 65000, 70000, 95000 et environ 100000.

Pour la variable Bonus/Malus :

Minimum value : La plus petite valeur observée dans la distribution est d'environ -6000.

Premier quartile (Q1) : Le 25ème centile de la distribution est de -6000. Cela signifie que 25% des valeurs sont inférieures ou égales à -6000.

Médiane : La valeur médiane de la distribution est également de -6000, ce qui indique que la moitié des données sont inférieures à cette valeur et l'autre moitié est supérieure.

Troisième quartile (Q3) : Le 75ème centile de la distribution est de 7500. Cela signifie que 75% des valeurs sont inférieures ou égales à 7500.

Maximum value : La valeur maximale observée dans la distribution est proche de 10000. Cela indique qu'il y a des véhicules avec un bonus élevé, mais ces valeurs sont moins fréquentes que les valeurs proches de la médiane.

```
# Statistiques descriptives
print(catalogue_data.describe())

    resultat_catalogue_co2.puissance  resultat_catalogue_co2.nbplaces \
count                270.000000          270.000000
mean               157.592593           5.222222
std                 90.551289           0.629707
min                 55.000000           5.000000
25%                109.000000           5.000000

50%                147.000000           5.000000
75%                170.000000           5.000000
max                 507.000000           7.000000

    resultat_catalogue_co2.nbpentes  resultat_catalogue_co2.prix \
count                279.000000          279.000000
mean                4.814815          26668.055556
std                  0.580798          19050.121112
min                 3.000000          7500.000000
25%                5.000000          16029.000000
50%                5.000000          20597.500000
75%                5.000000          30000.000000
max                 5.000000          101300.000000

    resultat_catalogue_co2.moyenne_bonus_malus \
count                270.000000
mean                -1145.543796
std                  6009.920623
min                 -6000.000000
25%                -6000.000000
50%                -6000.000000
75%                6137.231000
max                 8237.358000

    resultat_catalogue_co2.moyenne_rejets_co2 \
count                270.000000
mean                80.126426
std                  71.975386
min                  0.000000
25%                31.667000
50%                43.500000
75%                158.495000
max                 200.000000

    resultat_catalogue_co2.cout_energie_moyen \
count                270.000000
mean                320.952093
std                  237.278558
min                 72.727000
25%                96.000000
50%                206.000000
75%                561.465000
max                 749.980000
```

D'après l'analyse statistique descriptive précédente du DataFrame "resultat\_catalogue\_co2", nous en tirons les conclusions suivantes :

- "puissance" : La puissance moyenne des voitures est de 157.59 chevaux, avec un écart-type de 90.55. La puissance varie de 55 à 507 chevaux, avec des quartiles à 109, 147 et 170 chevaux pour le 25e, 50e et 75e percentiles respectivement.
- "nbplaces" : Le nombre moyen de places dans les voitures est de 5.22, avec un écart-type de 0.63. Le nombre de places varie de 5 à 7.
- "nbportes" : Le nombre moyen de portes est de 4.81, avec un écart-type de 0.58. Toutes les voitures ont au moins 3 portes et au plus 5.
- "prix" : Le prix moyen des voitures est de 26 668.06 unités, avec un écart-type de 19 050.12. Le prix varie de 7 500 à 101 300 unités, avec des quartiles à 16 029, 20 597 et 30 000 unités pour le 25e, 50e et 75e percentiles respectivement.
- "moyenne\_bonus\_malus" : La moyenne du bonus/malus est de -1145.54, avec un écart-type de 6009.92. Elle varie de -6000 à 8237.36
- "moyenne\_rejets\_co2" : La moyenne des rejets de CO2 est de 80.13, avec un écart-type de 71.98. Elle varie de 0 à 200.
- "cout\_énergie\_moyen" : Le coût moyen de l'énergie est de 320.95 unités, avec un écart-type de 237.28. Il varie de 72.73 à 749.98 unités.

Cette analyse statistique confirme les conclusions précédentes et offre une vue d'ensemble des variables contenues dans le DataFrame du catalogue.

**Visualisations bivariées** : Examiner les relations entre les variables numériques en utilisant des matrices de nuages de points (scatter plots) ou des heatmap de corrélation. Cela permet de détecter les corrélations entre les variables et d'identifier les associations significatives.



On constate des relations cruciales entre différentes variables. Tout d'abord, la corrélation entre la moyenne des rejets de CO<sub>2</sub> et le coût de l'énergie suggère que les véhicules émettant davantage de CO<sub>2</sub> sont souvent associés à des coûts d'exploitation plus élevés. Cette tendance pourrait influencer les choix de la clientèle et les stratégies de tarification.

De plus, la relation entre la moyenne des rejets de CO<sub>2</sub> et la moyenne du Bonus/Malus pourrait refléter les incitations fiscales ou les politiques environnementales qui influent sur le marché des véhicules. Une incitation à choisir des véhicules moins polluants, ce qui en retour, peut affecter les ventes et les bénéfices.

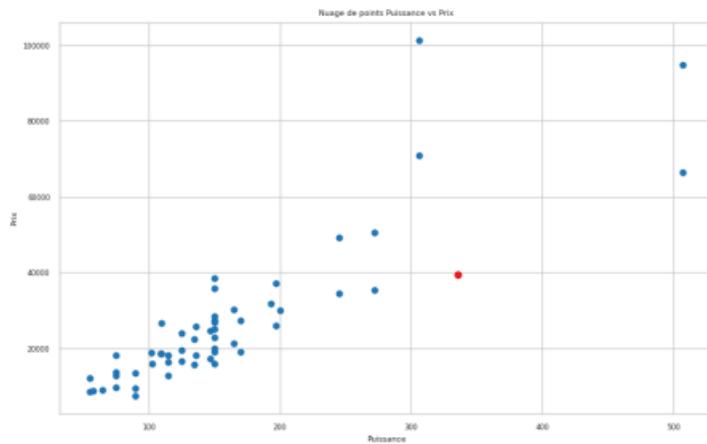
Quant à la corrélation entre le coût de l'énergie et la moyenne du Bonus/Malus, elle pourrait également indiquer que les véhicules éco-énergétiques, bénéficiant de bonus ou soumis à des malus fiscaux, ont des coûts d'exploitation différents, ce qui peut influencer aussi la décision d'achat.

Enfin, la forte linéarité entre la puissance et le prix suggère que les véhicules plus puissants sont généralement associés à des prix plus élevés, ce qui peut influencer la manière de segmentation du marché et définissons les stratégies de tarification.

Les relations linéaires identifiées entre les différentes variables nous indiquent quelles variables sont les plus importantes pour prédire ou classer certaines caractéristiques des véhicules.

```
# Nuages de points pour explorer les relations entre les variables continues
import numpy as np

# Ajouter un jitter aux données
puissance = catalogue_data['resultat_catalogue_co2.puissance']
prix = catalogue_data['resultat_catalogue_co2.prix']
plt.figure(figsize=(10, 6))
plt.scatter(puissance, prix, alpha=0.5)
plt.xlabel('Puissance')
plt.ylabel('Prix')
plt.title('Nuage de points Puissance vs Prix')
plt.grid(True)
plt.show()
```



Le nuage de points illustre la relation entre la puissance et le prix des véhicules dans notre ensemble de données. Nous observons une tendance linéaire générale, indiquant que les véhicules plus puissants tendent à avoir un prix plus élevé. Cependant, nous notons la présence de trois points éloignés du regroupement principal, représentant des véhicules à la fois puissants et coûteux. Ces

points pourraient correspondre à des cas exceptionnels, tels que des véhicules de luxe ou de haute performance.

#### ▼ Analyse des valeurs catégoriques

On travail a générer des histogrammes d'effectif des variables catégoriques présentes dans le DataFrame.

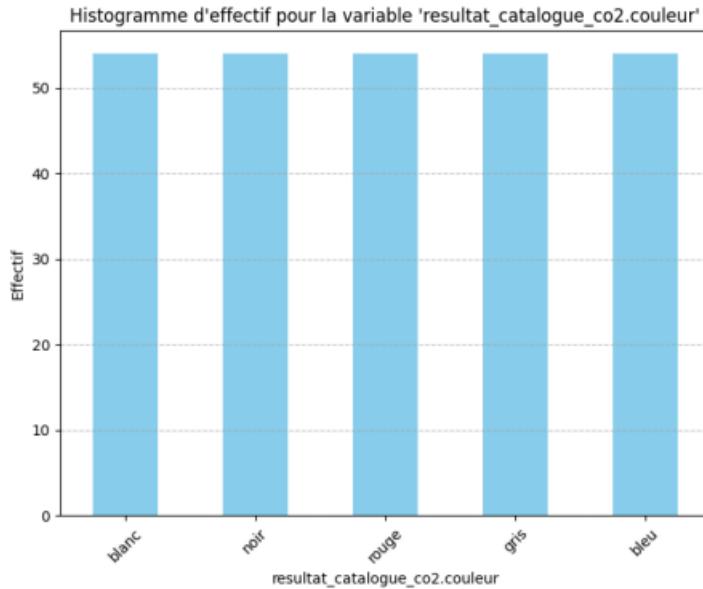
En parcourant chaque colonne catégorique de la liste `categorical_columns`, le code examine attentivement le nombre d'occurrences de chaque catégorie, puis crée des histogrammes à barres pour illustrer de manière visuelle la répartition des différentes catégories. Chaque histogramme est méticuleusement annoté avec un titre descriptif et des étiquettes pour les axes x et y, facilitant ainsi l'identification des variables analysées et la compréhension de l'effectif. De plus, les étiquettes de l'axe des x sont inclinées de 45 degrés pour une meilleure lisibilité, tandis qu'une grille subtile est ajoutée sur l'axe des y pour améliorer la visualisation des valeurs.

Enfin, chaque histogramme est présenté individuellement, permettant une exploration détaillée des distributions des variables catégoriques.

```
import matplotlib.pyplot as plt

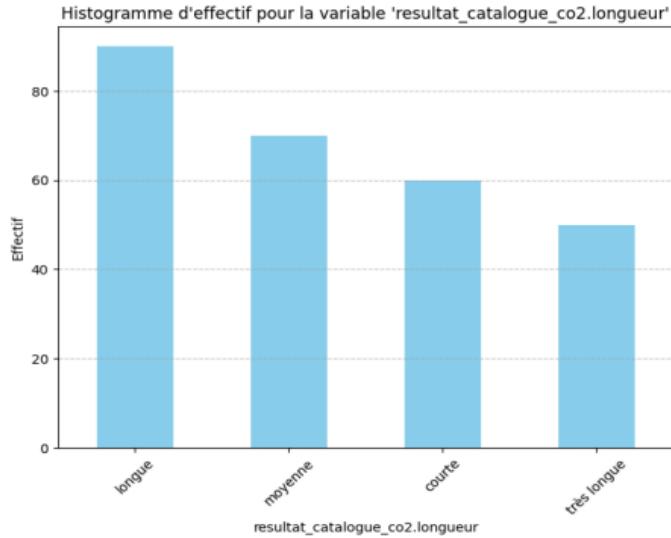
# Liste des noms de colonnes catégoriques
categorical_columns = ['resultat_catalogue_co2.couleur',
                      'resultat_catalogue_co2.longueur',
                      'resultat_catalogue_co2.occasion']

# Création d'histogrammes d'effectif pour chaque variable catégorique
for column in categorical_columns:
    plt.figure(figsize=(8, 6))
    catalogue_data[column].value_counts().plot(kind='bar', color='skyblue')
    plt.title(f"Histogramme d'effectif pour la variable '{column}'")
    plt.xlabel(column)
    plt.ylabel("Effectif")
    plt.xticks(rotation=45) # Rotation des étiquettes pour une meilleure lisibilité
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
```



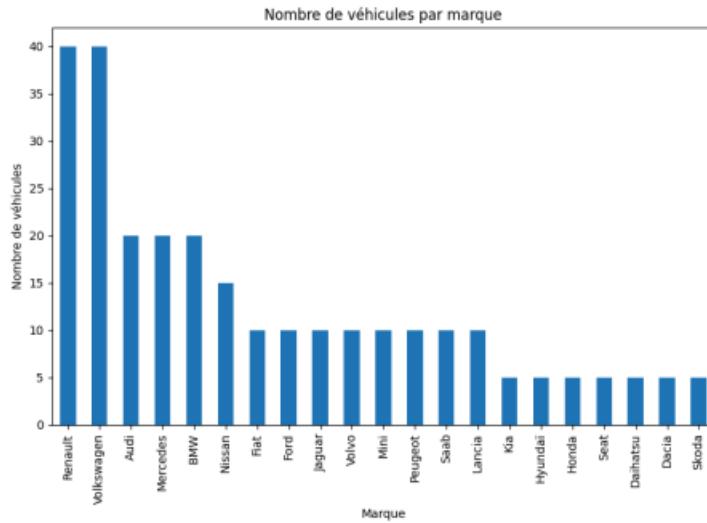
Les occurrences pour chaque couleur sont relativement similaires, ce qui indique une répartition équilibrée des couleurs dans le dataset.

Cela suggère que le dataset contient une variété de voitures de différentes couleurs, sans qu'une couleur ne domine de manière significative.

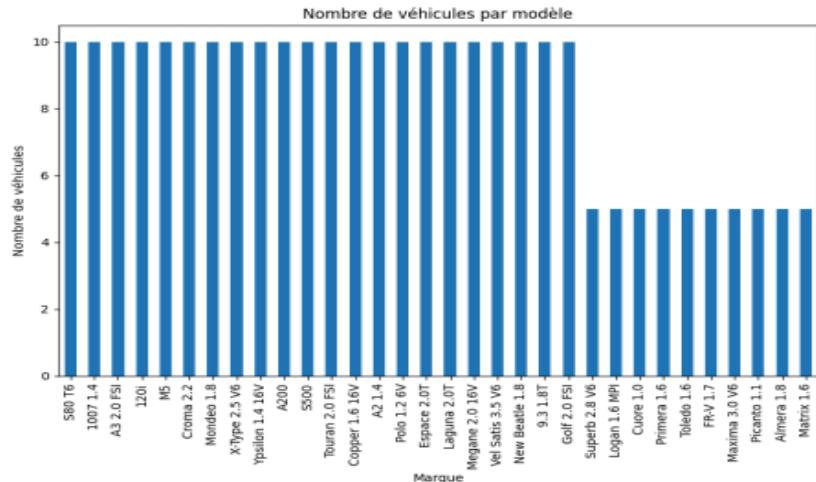


Il est noté une préférence générale pour les véhicules de longueur plus longue, avec une tendance à la dégradation de la préférence à mesure que la longueur diminue. Les voitures d'une longueur très longue sont moins désirées.

Cette tendance pourrait être liée à divers facteurs tels que la perception de l'espace intérieur, la capacité de transport de passagers et de marchandises, ou même des considérations esthétiques.

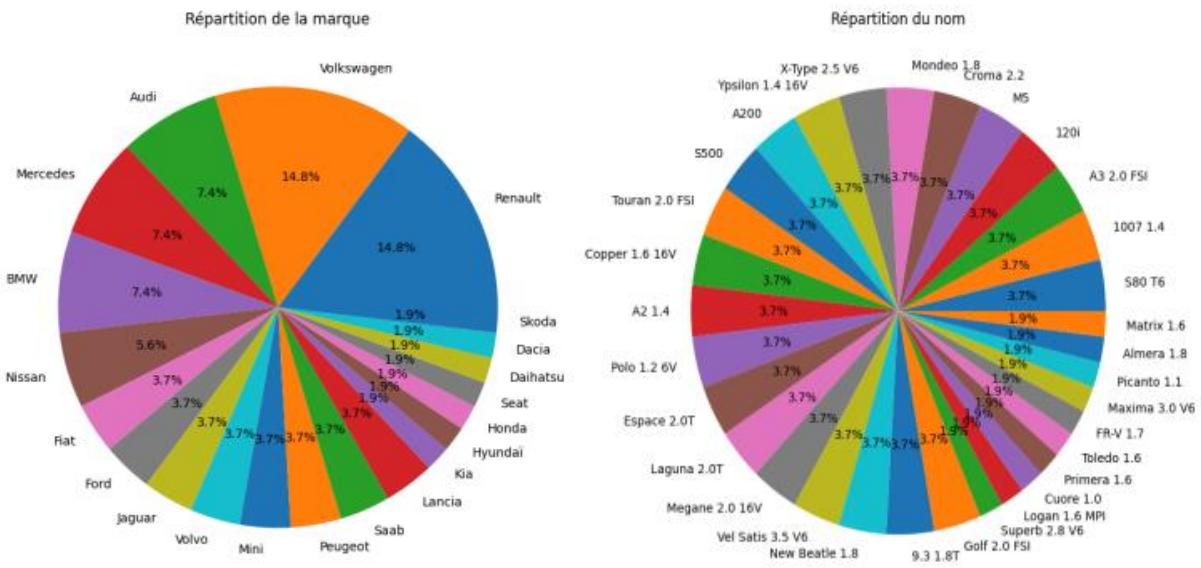


Pareillement, pour la colonne 'Marque', on observe une préférence marquée pour les marques Renault et Volkswagen par rapport aux autres marques.



En revanche, pour la colonne 'Modèle', on observe une préférence nettement moindre pour 11 modèles par rapport aux autres.

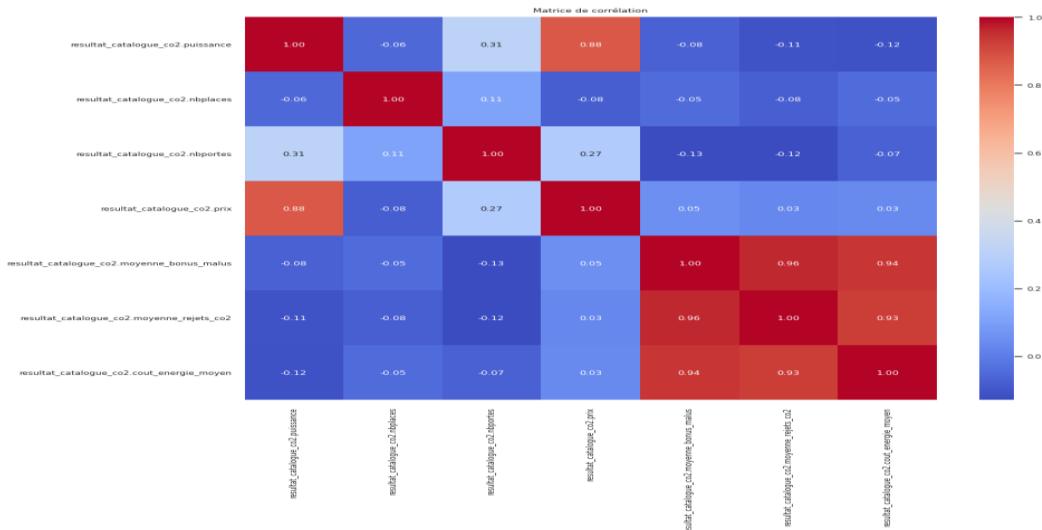
- Camembert des variables catégorielles:



Parmi les données de la DataFrame, il est notable que les marques Renault et Volkswagen sont prédominantes dans le catalogue suivi des marques Audi, Mercedes et BMW.

- heatmap (carte de chaleur) afin de visualiser la matrice de corrélation entre les variables numériques dans l'ensemble de données.

Le heatmap est un outil puissant qui peut aider à améliorer la performance des modèles de classification par la suite en identifiant les corrélations entre les variables, en sélectionnant les caractéristiques les plus informatives, en détectant les motifs cachés dans les données et en optimisant les hyperparamètres des algorithmes de machine learning. Son utilisation judicieuse peut donc conduire à des modèles plus précis et plus robustes.



On a constaté les résultats suivants très intéressants:

**Puissance et Prix :** Nous observons une corrélation positive très forte de 0.88 entre la puissance des véhicules et leur prix. Cette corrélation suggère que les véhicules plus puissants ont tendance à avoir des prix plus élevés. Ceci est cohérent avec l'intuition générale selon laquelle les véhicules dotés de plus de puissance et de performances ont tendance à être plus chers sur le marché.

**Coût énergétique et Bonus/Malus :** Une corrélation très élevée de 0.94 est observée entre le coût énergétique des véhicules et leur système de bonus/malus. Cette corrélation logique suggère que les véhicules qui consomment plus d'énergie ont tendance à être associés à des valeurs de bonus/malus plus élevées, ce qui peut refléter une politique de taxation basée sur les émissions de CO2 et la consommation de carburant.

**Coût énergétique et Moyenne des rejets de CO2 :** Nous observons également une corrélation élevée de 0.93 entre le coût énergétique des véhicules et leur moyenne de rejets de CO2. Cette corrélation met en évidence le lien entre la consommation de carburant et les émissions de CO2, ce qui est crucial dans le contexte de la réduction des émissions de gaz à effet de serre et de la promotion de véhicules plus écologiques.

**Moyenne des rejets de CO2 et Bonus/Malus :** Une corrélation très forte de 0.96 est observée entre la moyenne des rejets de CO2 et le système de bonus/malus. Cette corrélation suggère que les véhicules émettant davantage de CO2 sont généralement soumis à des valeurs de bonus/malus plus élevées, ce qui peut être une incitation financière à réduire les émissions de CO2 des véhicules.

## 8.2 Identification des catégories de véhicules en utilisant le Clustering

nous nous attelons à une tâche essentielle : l'identification des différentes catégories de véhicules à partir des informations fournies par le catalogue. Notre objectif est de créer des catégories telles que citadine, routière, sportive, etc., en tenant compte de diverses caractéristiques telles que la taille, la puissance, le prix, etc. Ces catégories ne sont pas simplement des classifications arbitraires ; elles doivent répondre à une variété de besoins des clients, reflétant ainsi les préférences et les exigences spécifiques du marché. Par exemple, une grande voiture pourrait convenir à une famille nombreuse, tandis qu'une petite voiture serait plus adaptée à la conduite en ville. En identifiant ces catégories de véhicules, nous jetons les bases pour les étapes suivantes du processus, où nous utiliserons ces classes prédéfinies pour prédire et recommander les véhicules les plus appropriés aux clients potentiels.

### - Prétraitement des données :

La transformation des variables catégoriques en un format encodé, conversion des variables pertinentes dans un format approprié pour l'analyse, et suppression des colonnes inutiles, afin de préparer les données pour une utilisation dans une approche de Clustering.

```
import pandas as pd

# Encdez la variable 'couleur' en variables indicatrices
couleur_encoded = pd.get_dummies(catalogue_data['resultat_catalogue_co2.couleur'], prefix='couleur').astype(int)

# Encdez la variable 'longueur' en variables indicatrices
longueur_encoded = pd.get_dummies(catalogue_data['resultat_catalogue_co2.longueur'], prefix='longueur').astype(int)

# Convertir la variable occasion en entiers (0 ou 1)
catalogue_data['resultat_catalogue_co2.occasion'] = catalogue_data['resultat_catalogue_co2.occasion'].astype(int)

# Concaténer toutes les colonnes de catalogue_data avec les encodages
X = pd.concat([catalogue_data, couleur_encoded, longueur_encoded], axis=1)

# Supprimer les colonnes 'resultat_catalogue_co2.longueur' et 'resultat_catalogue_co2.couleur'
X.drop(['resultat_catalogue_co2.marque', 'resultat_catalogue_co2.nom', 'resultat_catalogue_co2.longueur', 'resultat_catalogue_co2.couleur'], axis=1, inplace=True)

print(X.dtypes)
```

| Colonne                                    | Dtype   |
|--------------------------------------------|---------|
| resultat_catalogue_co2.puissance           | int64   |
| resultat_catalogue_co2.nbplaces            | int64   |
| resultat_catalogue_co2.nbportes            | int64   |
| resultat_catalogue_co2.occasion            | int64   |
| resultat_catalogue_co2.prix                | int64   |
| resultat_catalogue_co2.moyenne_bonus_malus | float64 |
| resultat_catalogue_co2.moyenne_rejets_co2  | float64 |
| resultat_catalogue_co2.cout_energie_moyen  | float64 |
| couleur_blan                               | int64   |
| couleur_bleu                               | int64   |
| couleur_gris                               | int64   |
| couleur_noir                               | int64   |
| couleur_rouge                              | int64   |
| longueur_courte                            | int64   |
| longueur_longue                            | int64   |
| longueur_moyenne                           | int64   |
| longueur_très longue                       | int64   |
| dtype: object                              |         |

### - Clustering :

Dans cette étape, nous utilisons l'algorithme K-means pour regrouper les données en clusters. Nous explorons différentes valeurs de K et visualisons les clusters dans un espace tridimensionnel afin d'obtenir un aperçu du nombre optimal de clusters pour nos données.

Le clustering nous permet de découvrir des structures intrinsèques dans nos données, en identifiant des groupes de véhicules similaires en fonction de leurs caractéristiques telles que la taille, la puissance et le prix. Cette approche nous permettra de définir des catégories distinctes qui peuvent être utilisées pour une recommandation précise et personnalisée des véhicules.

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Normaliser les données
scaler = StandardScaler()
data_scaled = scaler.fit_transform(X)

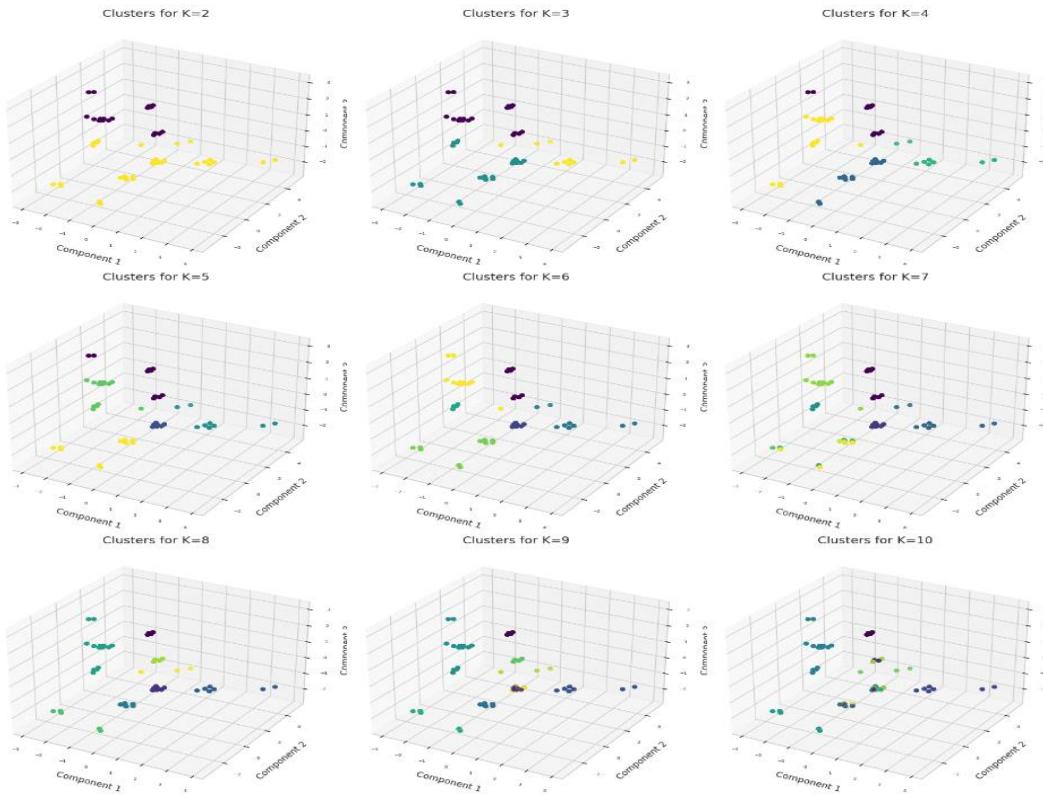
# Créer une grille de sous-plots pour chaque valeur de K
fig = plt.figure(figsize=(15, 15))
cluster_counts_per_k = [] # Liste pour stocker les nombres d'éléments dans chaque cluster pour chaque valeur de K
for i, k in enumerate(range(2, 11)):
    # Entrainer le modèle K-means
    kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42)
    kmeans.fit(data_scaled)

    # Calculer le nombre d'éléments pour chaque cluster
    cluster_labels = kmeans.labels_
    cluster_counts = pd.Series(cluster_labels).value_counts().sort_index()
    cluster_counts_per_k.append(cluster_counts)

# Réduire la dimensionnalité des données à 3D pour la visualisation
pca = PCA(n_components=3)
data_3d = pca.fit_transform(data_scaled)

# Visualiser les clusters dans l'espace 3D
ax = fig.add_subplot(3, 3, i+1, projection='3d')
ax.scatter(data_3d[:, 0], data_3d[:, 1], data_3d[:, 2], c=kmeans.labels_, cmap='viridis')
ax.set_title(f'Clusters for K={k}')
ax.set_xlabel('Component 1')
ax.set_ylabel('Component 2')
ax.set_zlabel('Component 3')

plt.tight_layout()
plt.show()
```



```
# Convertir la liste en DataFrame
cluster_counts_df = pd.DataFrame(cluster_counts_per_k).T
cluster_counts_df.columns = [f'K={k}' for k in range(2, 11)]

# Remplacer NaN par une chaîne vide
cluster_counts_df = cluster_counts_df.fillna('')

# Afficher le tableau
print("Nombre d'éléments pour chaque cluster pour chaque valeur de K :")
cluster_counts_df
```

Nombre d'éléments pour chaque cluster pour chaque valeur de K :

|   | K=2   | K=3   | K=4  | K=5  | K=6  | K=7  | K=8  | K=9  | K=10 |
|---|-------|-------|------|------|------|------|------|------|------|
| 0 | 90.0  | 90.0  | 50.0 | 50.0 | 40.0 | 40.0 | 20.0 | 20.0 | 20.0 |
| 1 | 180.0 | 130.0 | 95.0 | 50.0 | 50.0 | 50.0 | 50.0 | 25.0 | 21.0 |
| 2 |       | 50.0  | 45.0 | 45.0 | 45.0 | 45.0 | 35.0 | 35.0 | 35.0 |
| 3 |       | 80.0  | 65.0 | 20.0 | 20.0 | 35.0 | 35.0 | 21.0 |      |
| 4 |       |       | 60.0 | 60.0 | 33.0 | 70.0 | 70.0 | 70.0 |      |
| 5 |       |       |      | 55.0 | 55.0 | 25.0 | 25.0 | 25.0 |      |
| 6 |       |       |      |      | 27.0 | 20.0 | 20.0 | 21.0 |      |
| 7 |       |       |      |      |      | 15.0 | 15.0 | 15.0 |      |
| 8 |       |       |      |      |      |      | 25.0 | 21.0 |      |
| 9 |       |       |      |      |      |      |      | 21.0 |      |

À première vue, il semble qu'un nombre de clusters K compris entre 5 et 6 pourrait être intéressant pour diviser nos données en groupes distincts, mais cela nécessite une vérification à travers d'autres méthodes.

- La méthode du coude:

Dans notre projet, nous appliquons la méthode du coude pour déterminer le nombre optimal de clusters à utiliser dans le clustering des données de véhicules. En traçant la somme des distances au carré par rapport au nombre de clusters, nous identifierons le nombre de clusters qui représente le point où l'ajout de clusters supplémentaires ne contribue pas significativement à la réduction de la variation intra-cluster. Ce nombre optimal de clusters servira de base pour notre analyse de clustering ultérieure, nous permettant de créer des groupes significatifs de véhicules.

Cette méthode consiste à tracer le nombre de clusters (K) sur l'axe des abscisses et la somme des distances au carré des points de données par rapport à leur cluster le plus proche sur l'axe des ordonnées. En observant le graphique, on recherche le point où la diminution de la somme des distances commence à diminuer de manière significativement plus lente, formant ainsi un coude dans le graphique. Ce point est souvent interprété comme le nombre optimal de clusters pour les données.

L'étape suivante est de faire :

- Importation des bibliothèques : Les bibliothèques nécessaires, notamment scikit-learn pour l'algorithme KMeans, StandardScaler pour la normalisation des données, et KElbowVisualizer de Yellowbrick pour la visualisation du coude, sont importées.
- Normalisation des données : Les données sont normalisées à l'aide de la classe StandardScaler de scikit-learn, ce qui permet de mettre à l'échelle les variables pour qu'elles aient une moyenne nulle et une variance unitaire.
- Création de l'instance de KMeans : Une instance de l'algorithme KMeans est créée sans spécifier le nombre de clusters. Ce sera l'algorithme utilisé pour trouver le nombre optimal de clusters.
- Création du visualiseur de coude : Un objet KElbowVisualizer est créé en utilisant l'instance de KMeans et en spécifiant une plage de valeurs pour K (nombre de clusters) à tester, dans ce cas de 2 à 10 clusters.
- Ajustement du visualiseur : Le visualiseur est ajusté sur les données normalisées à l'aide de la méthode fit(). Cela permet au visualiseur de calculer les valeurs du critère de performance pour chaque valeur de K.
- Affichage du graphique : Enfin, le graphique du coude est affiché à l'aide de la méthode show() du visualiseur. Ce graphique montre la valeur du critère de performance (par défaut, WCSS - Within-Cluster Sum of Squares) en fonction du nombre de clusters (K), ce qui permet de sélectionner visuellement le nombre optimal de clusters en cherchant le coude dans la courbe.

- Yellowbrick est une bibliothèque Python qui facilite la visualisation de données et souvent utilisée en conjonction avec d'autres bibliothèques d'apprentissage automatique telles que scikit-learn pour explorer et comprendre les données, évaluer les modèles et diagnostiquer les problèmes.

```
!pip install yellowbrick

Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: yellowbrick in /home/vagrant/.local/lib/python3.9/site-packages (1.5)
Requirement already satisfied: matplotlib>=3.0.0,>=2.0.2 in /home/vagrant/.local/lib/python3.9/site-packages (from yellowbrick) (3.8.4)
Requirement already satisfied: scipy>=1.0.0 in /home/vagrant/.local/lib/python3.9/site-packages (from yellowbrick) (1.13.0)
Requirement already satisfied: scikit-learn>=1.0.0 in /home/vagrant/.local/lib/python3.9/site-packages (from yellowbrick) (1.4.1.post1)
Requirement already satisfied: numpy>=1.16.0 in /home/vagrant/.local/lib/python3.9/site-packages (from yellowbrick) (1.26.3)
Requirement already satisfied: cyeler>=0.10.0 in /home/vagrant/.local/lib/python3.9/site-packages (from yellowbrick) (0.12.1)

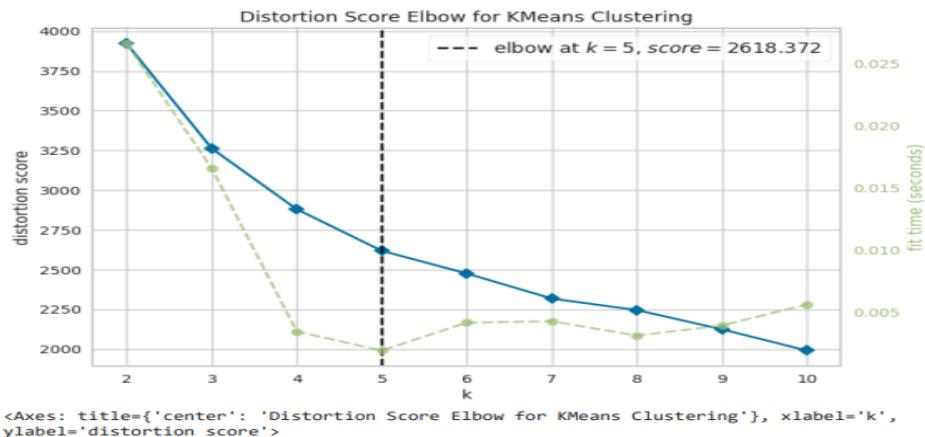
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from yellowbrick.cluster import KElbowVisualizer # Assurez-vous que cette ligne est présente
#par défaut se base sur le WCSS
# Créer une instance de KMeans pour chaque valeur de K à tester

# Normaliser les données
scaler = StandardScaler()
data_scaled = scaler.fit_transform(X)

model12 = KMeans()
visualizer = KElbowVisualizer(model12, k=(2,11))

# Ajuster le visualiseur sur les données normalisées
visualizer.fit(data_scaled)

# Afficher le graphique
visualizer.show()
```



L'objectif est de trouver le point où la décroissance de la performance ralentit, formant ainsi un "coude" dans le graphique. Ce point est généralement considéré comme le nombre optimal de clusters, car il capture la meilleure balance entre la complexité du modèle (plus de clusters) et sa capacité à expliquer les données (moins de dispersion intra-cluster). C'est le point où l'ajout de clusters supplémentaires n'apporte pas une amélioration significative de la performance du modèle.

Dans notre cas, le coude dans la courbe de distorsion se trouve à K=5, ce qui suggère que l'ajout de plus de clusters après cette valeur n'apporte pas une réduction significative de la distorsion intra-cluster. Ainsi, K=5 est considéré comme le nombre optimal de clusters pour nos données, offrant un bon compromis entre la complexité du modèle et la cohérence des clusters.

Le score associé à ce point, soit 2618.3, représente la valeur de la distorsion (ou WCSS) pour ce nombre de clusters. Plus ce score est bas, plus les clusters sont compacts et cohérents, ce qui est souhaitable dans une bonne partition de clustering. En d'autres termes, cette valeur indique que les

points à l'intérieur de chaque cluster sont proches les uns des autres, ce qui renforce la validité de la classification obtenue.

#### - La méthode de la silhouette:

Une approche puissante pour évaluer la qualité des clusters dans un ensemble de données en visualisant la cohésion et la séparation des clusters pour différentes valeurs de K, le nombre de clusters. Pour ce faire, nous dessinons un graphique où l'axe des abscisses représente les valeurs de K et l'axe des ordonnées représente les scores de silhouette associés.

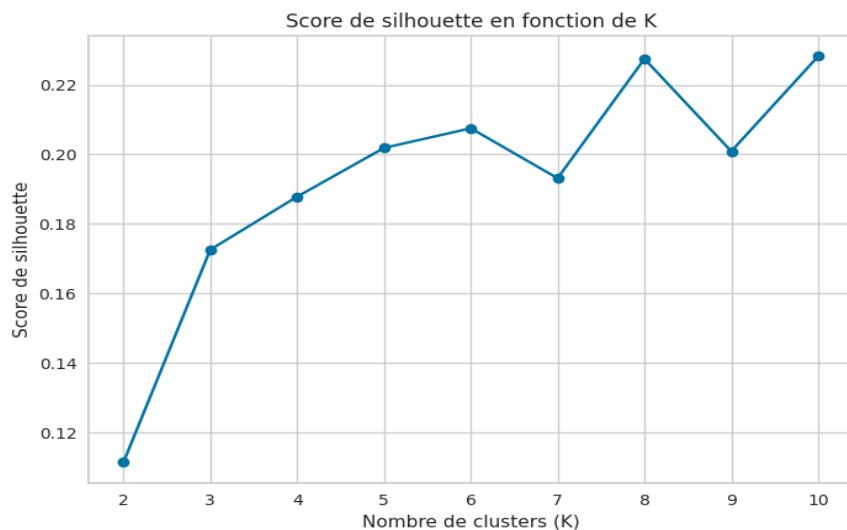
En examinant ce graphique, nous recherchons le pic le plus élevé, ce qui correspond au nombre optimal de clusters où le score de silhouette est maximisé. Cela indique que les clusters obtenus pour cette valeur de K ont une bonne séparation entre eux tout en maintenant une cohésion élevée à l'intérieur de chaque cluster. Ce nombre optimal de clusters nous permet d'obtenir une partition des données qui maximise à la fois la compacité intra-cluster et la séparation inter-cluster.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import silhouette_score

# Normaliser les données
scaler = StandardScaler()
data_scaled = scaler.fit_transform(X)

# Calculer le score de silhouette pour différentes valeurs de K
silhouette_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    cluster_labels = kmeans.fit_predict(data_scaled)
    silhouette_avg = silhouette_score(data_scaled, cluster_labels)
    silhouette_scores.append(silhouette_avg)

# Tracer le graphe du score de silhouette par rapport à k
plt.plot(range(2, 11), silhouette_scores, marker='o', linestyle='--')
plt.xlabel('Nombre de clusters (K)')
plt.ylabel('Score de silhouette')
plt.title('Score de silhouette en fonction de K')
plt.xticks(np.arange(2, 11, step=1))
plt.grid(True)
plt.show()
```



D'après la méthode de la silhouette, le plus grand pic correspond à k=8, tandis que selon la méthode du coude, elle indique un k=5.

À la lumière des résultats obtenus à partir des méthodes de la silhouette et du coude, nous sommes confrontés à un choix crucial pour déterminer le nombre optimal de clusters. Bien que la méthode de la silhouette identifie un pic significatif à  $k=8$ , la méthode du coude suggère plutôt un nombre de clusters optimal à  $k=5$ . Cependant, une analyse plus approfondie des résultats révèle une subtilité dans la structure des données.

Lorsque nous examinons attentivement les clusters dans l'espace tridimensionnel pour chaque valeur de  $k$ , nous observons que les configurations avec  $k=8$  et  $k=7$  présentent une superposition et un chevauchement entre les clusters. Cette ambiguïté dans la structure des données indique une difficulté à obtenir une séparation claire des groupes.

En revanche, pour les valeurs de  $k=5$  et  $k=6$ , nous constatons une meilleure séparation des clusters, avec des distances plus nettes entre eux. Cette cohérence dans la séparation des clusters suggère une structuration plus claire des données, offrant ainsi une représentation plus fidèle des groupes sous-jacents dans les données.

En considérant ces observations et en cherchant à obtenir une interprétation robuste et significative des clusters, nous concluons que les valeurs de  $k=5$  ou  $k=6$  sont les choix les plus appropriés pour la segmentation des données. Ces valeurs offrent un équilibre optimal entre la complexité du modèle et la clarté de la structure des clusters, ce qui permettra une recommandation plus précise et pertinente des véhicules aux clients.

- **Clustering des données avec  $n\_clusters = 5$ :**

à l'aide de l'algorithme K-means, visualisation des clusters dans un espace tridimensionnel et nuage de points après réduction de dimensionnalité avec l'analyse en composantes principales (ACP).

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Définir le nombre de clusters
n_clusters = 5

# Initialiser l'objet KMeans
kmeans = KMeans(n_clusters=n_clusters, random_state=42)

# Normaliser les données
scaler = StandardScaler()
data_scaled = scaler.fit_transform(X)

# Ajuster le modèle de clustering KMeans aux données
kmeans.fit(data_scaled)

# Obtenir les centres des clusters
cluster_centers = kmeans.cluster_centers_

# Obtenir les étiquettes des clusters pour chaque point de données
cluster_labels = kmeans.labels_

# Réduction de dimensionnalité avec l'ACP
pca = PCA(n_components=3)
X_pca = pca.fit_transform(data_scaled)

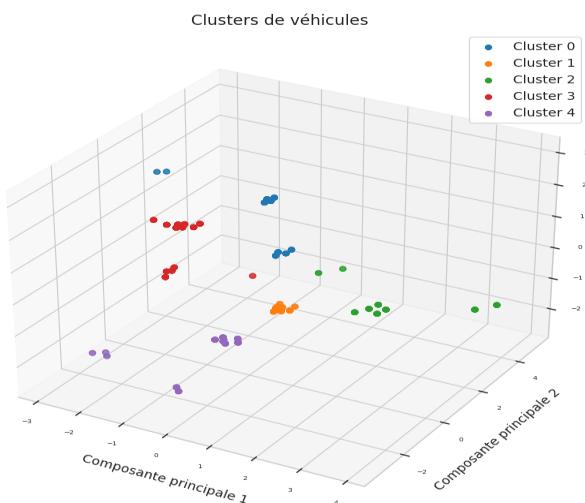
# Créer la visualisation 3D
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Plots pour chaque cluster
for cluster_label in range(n_clusters):
    ax.scatter(X_pca[cluster_labels == cluster_label, 0], X_pca[cluster_labels == cluster_label, 1], X_pca[cluster_labels == cluster_label, 2])

ax.set_title('Clusters de véhicules')
ax.set_xlabel('Composante principale 1')
ax.set_ylabel('Composante principale 2')
ax.set_zlabel('Composante principale 3')

ax.legend()

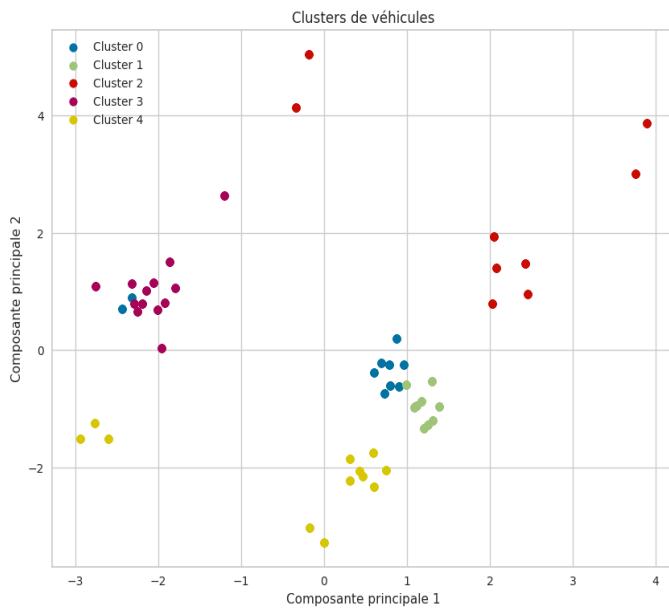
plt.show()
```



```
# Réduction de dimensionnalité avec l'ACP
pca = PCA(n_components=2)
X_pca = pca.fit_transform(data_scaled)

# Création du nuage de points avec les clusters comme étiquettes
plt.figure(figsize=(10, 8))
for cluster_label in range(n_clusters):
    plt.scatter(X_pca[cluster_labels == cluster_label, 0], X_pca[cluster_labels == cluster_label, 1], label=f'Cluster {cluster_label}')

plt.title('Clusters de véhicules')
plt.xlabel('Composante principale 1')
plt.ylabel('Composante principale 2')
plt.legend()
plt.show()
```



Le code suivant permet de calculer le nombre d'éléments appartenant à chaque cluster, fournissant ainsi une information supplémentaire sur la répartition des clusters.

```
# Créer un DataFrame avec les données des véhicules et les étiquettes des clusters
data_with_clusters = pd.DataFrame({'Cluster': cluster_labels}) # Remplacez cluster_labels par vos étiquettes de cluster
data_with_clusters = pd.concat([data_with_clusters, pd.DataFrame(X, columns=X.columns)], axis=1) # Remplacez X par vos données et X.columns

# Calculer le nombre d'éléments pour chaque cluster
cluster_counts = data_with_clusters['Cluster'].value_counts()

# Afficher le nombre d'éléments pour chaque cluster
print("Nombre d'éléments pour chaque cluster :")
print(cluster_counts)
```

```
Nombre d'éléments pour chaque cluster :
Cluster
3    65
4    60
0    50
1    50
2    45
Name: count, dtype: int64
```

On crée un DataFrame contenant les données des véhicules ainsi que les étiquettes des clusters assignées à chaque véhicule. On calcule les statistiques descriptives pour chaque cluster, telles que la moyenne, l'écart-type, les quartiles, etc pour chaque colonne du DataFrame et ce pour examiner les caractéristiques spécifiques de chaque cluster et comprendre comment les variables se comportent au sein de chaque groupe.

```
import pandas as pd

# Créer un DataFrame avec les données des véhicules et les étiquettes des clusters
data_with_clusters = pd.DataFrame({'Cluster': cluster_labels})
data_with_clusters = pd.concat([data_with_clusters, pd.DataFrame(X, columns=X.columns)], axis=1)

# Calculer les statistiques descriptives pour chaque cluster
cluster_stats = data_with_clusters.groupby('Cluster').describe()

# Boucle sur les colonnes du DataFrame
for col in cluster_stats.columns.levels[0]:
    print(f"Colonne : {col}")
    # Afficher les informations pour chaque colonne
    for cluster_label in range(n_clusters):
        print(f"Cluster {cluster_label}:")
        print(cluster_stats[col].loc[cluster_label].to_string())
        print("\n")

    Colonne : résultat_catalogue_co2.puissance
    Cluster 0:
    count      50.000000
    mean      159.200000
    std       27.949663
    min      103.000000
    25%     150.000000
    50%     165.000000
    75%     170.000000
    max      197.000000

    Cluster 1:
    count      50.000000
    mean      139.000000
    std       15.779087
    min      110.000000
    25%     135.000000
    50%     150.000000
    75%     150.000000
    max      150.000000

    Cluster 2:
    count      45.000000
    mean      317.000000
    std      107.785562
    min      193.000000
    25%     245.000000
    50%     272.000000
    75%     306.000000
    max      507.000000
```

Un nouveau DataFrame est créé en combinant les étiquettes des clusters avec les données des véhicules. Cela permet d'associer chaque ligne de données à son cluster correspondant.

Calcul des statistiques descriptives pour chaque cluster : En utilisant groupby, les données sont regroupées par cluster, puis des statistiques descriptives telles que la moyenne, l'écart-type, le minimum, le maximum et les quartiles sont calculées pour chaque colonne de données.

Boucle sur les colonnes du DataFrame est utilisée pour parcourir chaque colonne du DataFrame contenant les statistiques descriptives. Pour chaque colonne, les informations sont affichées pour chaque cluster, montrant les statistiques descriptives spécifiques pour cette colonne dans chaque cluster.

D'après les résultats, on peut déduire les interprétations suivantes :

- **Cluster 0: Familiale**

Les voitures de ce cluster sont principalement associées au type familial. Elles offrent une puissance moyenne élevée de 159,2 chevaux et un nombre de places moyen élevé de 6,2, avec une configuration typique de 5 portes. Environ 40% de ces voitures sont d'occasion, indiquant une certaine préférence pour le marché de l'occasion. Avec un prix moyen élevé de 24 306,30 €, ces véhicules sont souvent accompagnés d'un bonus-malus moyen négatif de -3572,55, ce qui pourrait refléter une tendance vers des moteurs plus puissants. De plus, ces voitures présentent une moyenne de rejets de CO<sub>2</sub> de 38,26 g/km et un coût moyen de l'énergie élevé, suggérant une certaine attention à l'efficacité énergétique. Les couleurs dominantes sont le blanc, le bleu, le gris, le noir et le rouge, avec une longueur majoritairement longue, offrant ainsi un espace généreux pour les familles.

- **Cluster 1: Compacte**

Ce cluster est associé aux voitures compactes, offrant une puissance moyenne modérée de 139 chevaux. Econçues pour accueillir en moyenne 5 personnes avec une configuration typique de 5 portes. Environ la moitié de ces voitures sont d'occasion, ce qui suggère une certaine accessibilité sur le marché de l'occasion. Avec un prix moyen modéré de 23 150,40 € et un bonus-malus moyen négatif de -6000, ces véhicules offrent un bon équilibre entre prix et performances. De plus, ils affichent une moyenne de rejets de CO<sub>2</sub> modérée de 30,60 g/km et un coût moyen de l'énergie modéré de 134,03 €. Les couleurs dominantes sont le blanc, le bleu, le gris, le noir et le rouge, avec une longueur majoritairement courte, idéale pour la conduite en ville.

- **Cluster 2: Sportive**

Ce cluster est associé aux voitures sportives, offrant une puissance moyenne très élevée de 317 chevaux. Elles sont conçues pour accueillir en moyenne 5 personnes avec une configuration typique de 5 portes. Près de 44,44% de ces voitures sont d'occasion, malgré un prix moyen très élevé de 59 405,56 €, ce qui indique une certaine demande pour des voitures de sport sur le marché de l'occasion. Avec un bonus-malus moyen négatif de -1502,81, ces véhicules sont souvent associés à des performances élevées. Cependant, ils présentent également une moyenne de rejets de CO<sub>2</sub> élevée de 64,75 g/km et un coût moyen de l'énergie élevé de 257,48 €. Les couleurs dominantes sont le blanc, le bleu, le gris, le noir et le rouge, avec une longueur majoritairement courte, soulignant leur nature sportive et compacte.

- **Cluster 3: Économique**

Les voitures de ce cluster sont principalement associées au type économique. Elles offrent une puissance moyenne modérée de 133,23 chevaux et sont conçues pour accueillir en moyenne 5 personnes avec une configuration typique de 5 portes. Environ 30,77% de ces voitures sont d'occasion, ce qui suggère une certaine demande pour des voitures économiques sur le marché de l'occasion. Avec un prix moyen modéré de 21 837,38 € et un bonus-malus moyen positif de 6383,06, ces véhicules offrent un bon rapport qualité-prix. Cependant, ils affichent une moyenne de rejets de CO<sub>2</sub> élevée de 172,85 g/km et un coût moyen de l'énergie élevé de 618,10 €. Les couleurs dominantes sont le blanc, le bleu, le gris, le noir et le rouge, avec une longueur majoritairement courte, soulignant leur nature économique et polyvalente.

- **Cluster 4: Citadine**

Les voitures de ce cluster sont principalement associées au type citadine. Elles offrent une puissance moyenne basse de 78,58 chevaux et sont conçues pour accueillir en moyenne 5 personnes avec une configuration typique de 4 portes. Environ 41,67% de ces voitures sont d'occasion, offrant ainsi une certaine accessibilité sur le marché de l'occasion. Avec un prix moyen bas de 12 247,67 € et un bonus-malus moyen négatif de -2965,69, ces véhicules sont idéaux pour une conduite en ville. De plus, ils affichent une moyenne de rejets de CO<sub>2</sub> modérée de 67,37 g/km et un coût moyen de l'énergie modéré de 246,47 €. Les couleurs dominantes sont le blanc, le bleu, le gris, le noir et le rouge, avec une longueur majoritairement moyenne, adaptée à la conduite urbaine.

- **Clustering des données avec n\_clusters = 6:**

Visualisation des clusters dans un espace tridimensionnel :

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Définir le nombre de clusters
n_clusters = 6

# Initialiser l'objet KMeans
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    •

# Normaliser les données
scaler = StandardScaler()
data_scaled = scaler.fit_transform(X)

# Ajuster le modèle de clustering KMeans aux données
kmeans.fit(data_scaled)

# Obtenir les centres des clusters
cluster_centers = kmeans.cluster_centers_

# Obtenir les étiquettes des clusters pour chaque point de données
cluster_labels = kmeans.labels_
```

```
# Réduction de dimensionnalité avec l'ACP
pca = PCA(n_components=3)
X_pca = pca.fit_transform(data_scaled)

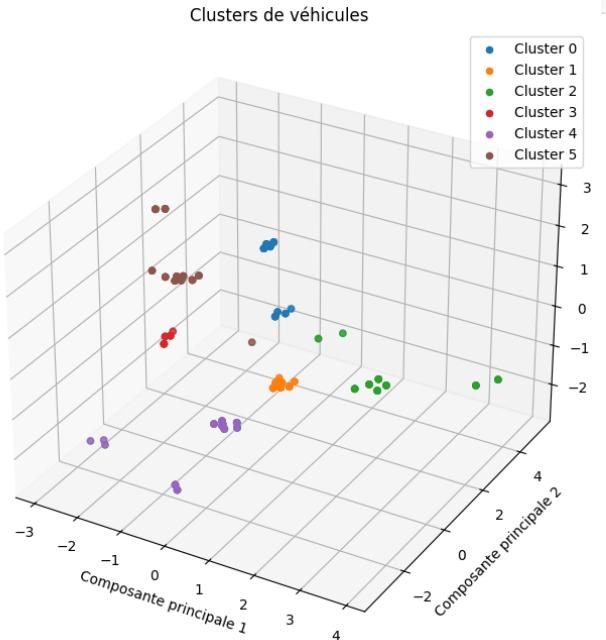
# Créer la visualisation 3D
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Plots pour chaque cluster
for cluster_label in range(n_clusters):
    ax.scatter(X_pca[cluster_labels == cluster_label, 0], X_pca[cluster_labels == cluster_label, 1], X_pca[cluster_labels == cluster_label, 2])

ax.set_title('Clusters de véhicules')
ax.set_xlabel('Composante principale 1')
ax.set_ylabel('Composante principale 2')
ax.set_zlabel('Composante principale 3')

ax.legend()

plt.show()
```



Calculer le nombre d'éléments appartenant à chaque cluster :

```
# Créer un DataFrame avec les données des véhicules et les étiquettes des clusters
data_with_clusters = pd.DataFrame({'Cluster': cluster_labels}) # Remplacez cluster_labels par vos étiquettes de cluster
data_with_clusters = pd.concat([data_with_clusters, pd.DataFrame(X, columns=X.columns)], axis=1) # Remplacez X par vos données et X.columns

# Calculer le nombre d'éléments pour chaque cluster
cluster_counts = data_with_clusters['Cluster'].value_counts()

# Afficher le nombre d'éléments pour chaque cluster
print("Nombre d'éléments pour chaque cluster :")
print(cluster_counts)

Nombre d'éléments pour chaque cluster :
Cluster
4    60
5    55
1    50
2    45
0    40
3    20
Name: count, dtype: int64
```

On calcule les statistiques descriptives pour chaque cluster et on examine ses caractéristiques spécifiques :

```
import pandas as pd

# Créer un DataFrame avec les données des véhicules et les étiquettes des clusters
data_with_clusters = pd.DataFrame({'Cluster': cluster_labels})
data_with_clusters = pd.concat([data_with_clusters, pd.DataFrame(X, columns=X.columns)], axis=1)

# Calculer les statistiques descriptives pour chaque cluster
cluster_stats = data_with_clusters.groupby('Cluster').describe()

# Boucle sur les colonnes du DataFrame
for col in cluster_stats.columns.levels[0]:
    print(f"Colonne : {col}")
    # Afficher les informations pour chaque colonne
    for cluster_label in range(n_clusters):
        print(f"Cluster {cluster_label}:")
        print(cluster_stats[col].loc[cluster_label].to_string())
        print("\n")

Colonne : resultat_catalogue_co2.puissance
Cluster 0:
count      40.000000
mean     170.500000
std      17.194215
min     150.000000
25%     161.250000
50%     167.500000
75%     176.750000
max     197.000000

Cluster 1:
count      50.000000
mean     139.000000
std      15.779087
min     110.000000
25%     135.000000
50%     150.000000
75%     150.000000
max     150.000000

Cluster 2:
count      45.000000
mean     317.000000
std     107.785562
min     193.000000
25%     245.000000
50%     272.000000
75%     306.000000
max     507.000000
```

- Choix du nombre de clusters optimal :

Choisir cinq clusters plutôt que six peut être une décision pour simplifier l'analyse et interprétation des données. En réduisant le nombre de clusters, on évite la complexité excessive et les chevauchements entre les groupes, ce qui facilite la distinction et l'affectation des données. Avec cinq clusters, chaque groupe peut être plus distinct et représentatif d'un profil spécifique, ce qui facilite leur interprétation et leur utilisation dans la prise de décision.

En revanche, choisir six clusters peut entraîner une division excessive des données, où certains groupes peuvent présenter des similitudes très proches et donc être difficiles à distinguer, comme c'est le cas pour les clusters 3 et 5, qui tous deux relèvent de la catégorie économique selon les résultats des statistiques descriptives. Cela peut rendre l'analyse plus ardue et moins informative, car les différences entre les clusters sont moins significatives. En optant pour cinq clusters, nous privilégions la simplicité, la clarté et la pertinence des groupes identifiés. Cette approche nous permet de maximiser la représentativité des clusters tout en minimisant la complexité de l'analyse, ce qui favorise une compréhension plus profonde et une utilisation plus efficace des résultats du clustering.

On utilise les résultats pour attribuer des catégories aux véhicules dans catalogue.

1. Obtention des étiquettes de cluster prédites : Les étiquettes de cluster prédites sont obtenues à partir de l'objet kmeans qui a été ajusté précédemment.
2. Ajout des étiquettes de cluster à la DataFrame : Les étiquettes de cluster prédites sont ajoutées comme une nouvelle colonne nommée 'Cluster' à la DataFrame catalogue\_data.
3. Création du dictionnaire pour mapper les numéros de cluster aux types de véhicules : Un dictionnaire est créé pour mapper les numéros de cluster à des catégories de véhicules spécifiques, telles que 'Familiale', 'Prestige', 'Sportive', 'Économique' et 'Citadine'.
4. Ajout de la colonne 'Catégorie' à partir du mapping des clusters : Une nouvelle colonne nommée 'Catégorie' est ajoutée à la DataFrame en utilisant la méthode map() pour mapper les numéros de cluster aux catégories de véhicules correspondantes.
5. Affichage des types de véhicules prédits pour chaque ligne : Les premières lignes de la DataFrame sont imprimées à nouveau, mais cette fois-ci avec une nouvelle colonne 'Catégorie', qui montre la catégorie prédite pour chaque véhicule en fonction de son cluster.

```
from sklearn.cluster import KMeans
import numpy as np

# Obtention des étiquettes de cluster prédictes
cluster_labels = kmeans.labels_

# Ajout des étiquettes de cluster à la dataframe
catalogue_data['Cluster'] = cluster_labels

# Affichage des classes prédictes pour chaque cluster
print(catalogue_data[['Cluster', 'resultat_catalogue_co2.marque', 'resultat_catalogue_co2.nom']].head())
# Création du dictionnaire pour mapper les numéros de cluster aux types de véhicules
cluster_type_mapping = {
    0: 'Familiale',
    1: 'Prestige',
    2: 'Sportive',
    3: 'Économique',
    4: 'Citadine'
}

# Ajout de la colonne 'Type' à partir du mapping des clusters
catalogue_data['Catégorie'] = catalogue_data['Cluster'].map(cluster_type_mapping)

# Affichage des types de véhicules prédicts pour chaque ligne
print(catalogue_data[['resultat_catalogue_co2.marque', 'resultat_catalogue_co2.nom', 'Cluster', 'Catégorie']].head())
```

|   | Cluster                       | resultat_catalogue_co2.marque | resultat_catalogue_co2.nom |           | Cluster | Catégorie |
|---|-------------------------------|-------------------------------|----------------------------|-----------|---------|-----------|
| 0 | 2                             | Volvo                         | S80 T6                     |           | 2       | Sportive  |
| 1 | 2                             | Volvo                         | S80 T6                     |           | 2       | Sportive  |
| 2 | 2                             | Volvo                         | S80 T6                     |           | 2       | Sportive  |
| 3 | 2                             | Volvo                         | S80 T6                     |           | 2       | Sportive  |
| 4 | 2                             | Volvo                         | S80 T6                     |           | 2       | Sportive  |
|   | resultat_catalogue_co2.marque | resultat_catalogue_co2.nom    | Cluster                    | Catégorie |         |           |
| 0 | Volvo                         | S80 T6                        | 2                          | Sportive  |         |           |
| 1 | Volvo                         | S80 T6                        | 2                          | Sportive  |         |           |
| 2 | Volvo                         | S80 T6                        | 2                          | Sportive  |         |           |
| 3 | Volvo                         | S80 T6                        | 2                          | Sportive  |         |           |
| 4 | Volvo                         | S80 T6                        | 2                          | Sportive  |         |           |

### 8.3 Analyse Exploratoire et application du Clustering aux catégories du modèle d'analyse

L'objectif est d'attribuer à chacun des véhicules la catégorie qui lui correspond en utilisant le modèle définissant les catégories de véhicules généré précédemment. Enfin, nous avons analysé les résultats pour comprendre les tendances du marché automobile.

Les données des clients et leurs immatriculations sont extraites à partir du modèle d'analyse du datalake Hive, puis jointes avec les données du catalogue en utilisant la marque du véhicule. Les données d'Immatriculations contiennent les informations sur les véhicules vendus cette année.

```
# Exécution d'une requête SQL
cursor.execute('SELECT * FROM MBDS_Projet.model_immatriculations_clients')

# Récupération des noms de colonnes
columns = [desc[0] for desc in cursor.description]

# Récupération des résultats
results = cursor.fetchall()

# Crédation d'un DataFrame Pandas avec les résultats et les noms de colonnes
immatriculation_client = pd.DataFrame(results, columns=columns)

immatriculation_client.head()
```

|   | model_immatriculations_clients.immatriculation | model_immatriculations_clients.marque | model_immatriculations_clients.nom | model_immatriculations_clients.type |
|---|------------------------------------------------|---------------------------------------|------------------------------------|-------------------------------------|
| 0 | 0 BZ 21                                        | Audi                                  | A2                                 | 1.4                                 |
| 1 | 0 CQ 77                                        | Peugeot                               | 1007                               | 1.4                                 |
| 2 | 0 DQ 29                                        | Peugeot                               | 1007                               | 1.4                                 |
| 3 | 0 JO 29                                        | Renault                               | Vel Satis                          | 3.5 V6                              |
| 4 | 0 NK 32                                        | BMW                                   | M5                                 |                                     |

```
merged_data = pd.merge(immatriculation_client, catalogue_data, how='left', left_on='model_immatriculations_clients.marque', right_on='resultat_catalogue_co2.marque')

# Sélectionnez les lignes distinctes en fonction de la colonne 'immatriculation' et conservez toutes les colonnes
merged_data = merged_data.drop_duplicates(subset='model_immatriculations_clients.immatriculation')

# Affichez le résultat
merged_data.head()
```

|    | model_immatriculations_clients.immatriculation | model_immatriculations_clients.marque | model_immatriculations_clients.nom | model_immatriculations_clients.type |
|----|------------------------------------------------|---------------------------------------|------------------------------------|-------------------------------------|
| 0  | 0 BZ 21                                        | Audi                                  | A2                                 | 1.4                                 |
| 20 | 0 CQ 77                                        | Peugeot                               | 1007                               | 1.4                                 |
| 30 | 0 DQ 29                                        | Peugeot                               | 1007                               | 1.4                                 |
| 40 | 0 JO 29                                        | Renault                               | Vel Satis                          | 3.5 V6                              |
| 80 | 0 NK 32                                        | BMW                                   | M5                                 |                                     |

5 rows × 28 columns

Les données sont nettoyées en éliminant les doublons et en supprimant certaines colonnes redondantes, aboutissant ainsi à la création d'un DataFrame final prêt pour une analyse plus poussée.

```
# Liste des colonnes à supprimer
columns_to_drop = ['resultat_catalogue_co2.longueur', 'resultat_catalogue_co2.puissance', 'resultat_catalogue_co2.nbplaces', 'resultat_catalogue_co2.prix']

# Supprimer les colonnes spécifiées
merged_data = merged_data.drop(columns=columns_to_drop).reset_index(drop=True)

merged_data
```

|        | model_immatriculations_clients.immatriculation | model_immatriculations_clients.marque | model_immatriculations_clients.nom | mode: |
|--------|------------------------------------------------|---------------------------------------|------------------------------------|-------|
| 0      | 0 BZ 21                                        | Audi                                  | A2 1.4                             |       |
| 1      | 0 CQ 77                                        | Peugeot                               | 1007 1.4                           |       |
| 2      | 0 DQ 29                                        | Peugeot                               | 1007 1.4                           |       |
| 3      | 0 JO 29                                        | Renault                               | Vel Satis 3.5 V6                   |       |
| 4      | 0 NK 32                                        | BMW                                   | M5                                 |       |
| ...    | ...                                            | ...                                   | ...                                | ...   |
| 199938 | 9999 GH 10                                     | Volkswagen                            | Polo 1.2 6V                        |       |
| 199939 | 9999 KX 26                                     | Audi                                  | A2 1.4                             |       |
| 199940 | 9999 LG 27                                     | Mercedes                              | A200                               |       |
| 199941 | 9999 NG 65                                     | Ford                                  | Mondeo 1.8                         |       |
| 199942 | 9999 NQ 33                                     | Peugeot                               | 1007 1.4                           |       |

99943 rows × 21 columns

- Exploration des données relative aux immatriculations de chaque colonne de la dataframe "merged\_data" :

```
columns=['model_immatriculations_clients.marque','model_immatriculations_clients.nom','model_immatriculations_clients.puissance','model_imma
# Pour chaque colonne de la DataFrame
for column in columns:
    # Obtenir les valeurs uniques de la colonne
    unique_values = merged_data[column].unique()
    # Afficher les valeurs uniques avec le nom de la colonne
    print(f"Colonnes: {column}")
    print(unique_values)

Colonnes: model_immatriculations_clients.marque
['Audi' 'Peugeot' 'Renault' 'BMW' 'Volkswagen' 'Volvo' 'Jaguar' 'Seat'
 'Ford' 'Saab' 'Mercedes' 'Fiat' 'Skoda' 'Nissan' 'Kia' 'Daihatsu'
 'Lancia' 'Dacia' 'Mini']
Colonnes: model_immatriculations_clients.nom
['A2 1.4' '1007 1.4' 'Vel Satis 3.5 V6' 'M5' 'Megane 2.0 16V'
 'Polo 1.2 6V' '580 16' 'X-Type 2.5 V6' 'Toledo 1.6' 'A3 2.0 FSI'
 'Mondeo 1.8' '120i' '9.3 1.8T' '5500' 'Croma 2.2' 'Laguna 2.0T'
 'Superb 2.8 V6' 'Maxima 3.0 V6' 'Golf 2.0 FSI' 'Picanto 1.1' 'Almera 1.8'
 'Cuore 1.0' 'Ypsilon 1.4 16V' 'A200' 'Primera 1.6' 'Logan 1.6 MPI'
 'Copper 1.6 16V' 'New Beetle 1.8']
Colonnes: model_immatriculations_clients.puissance
[ 75 245 507 135  55 272 197 102 150 125 306 147 170 193 200  65 115  58
 90 136 109 110]
Colonnes: model_immatriculations_clients.prix
[ 18310  9625 49200 94800 15644 12200 50500 25970 18880 13750
 28500 16730 25060 66360 34440 38600 70910 17346 35350 27300
 31790 37100 19110 30000 24780 22900 8540 8990 27020 16450
 12817 23900 101300 8850 22350 16029 9450 25900 18650 18130
 7500 12740 19950 18641 13500 18200 26630]
Colonnes: model_immatriculations_clients.longueur
['courte' 'tres longue' 'moyenne' 'longue']
Colonnes: model_immatriculations_clients.nbplaces
[5]
Colonnes: model_immatriculations_clients.nbportes
[5 3]
Colonnes: model_immatriculations_clients.couleur
['noir' 'gris' 'blanc' 'rouge' 'bleu']
Colonnes: model_immatriculations_clients.occasion
[False True]
```

Nous vérifions si les données des immatriculations du modèle d'analyse présentes dans le dataframe "immatriculations\_client" sont les mêmes après la jointure avec les données du catalogue dans le dataframe "merged\_data", afin de garantir la cohérence des données.

```
import pandas as pd

# Colonnes spécifiques à sélectionner dans merged_data pour la comparaison
colonnes_a_comparer = ['model_immatriculations_clients.immatriculation', 'model_immatriculations_clients.marque', 'model_immatriculations_clients.type']

# Sélectionnez les colonnes spécifiques dans merged_data et immatriculation_client
merged_data_selection = merged_data[colonnes_a_comparer].reset_index(drop=True)
immatriculation_selection = immatriculation_client[colonnes_a_comparer].reset_index(drop=True)

# Comparez les colonnes sélectionnées des deux DataFrames
comparaison = immatriculation_selection.compare(merged_data_selection)

# Comptez le nombre de lignes différentes
nombre_lignes_differentes = len(comparaison)

# Affichez le résultat
print("Nombre de lignes différentes entre les colonnes sélectionnées de immatriculation_client et merged_data : ", nombre_lignes_differentes)

# Vérifiez si les deux DataFrames sont égaux
sont_egaux = immatriculation_selection.equals(merged_data_selection)

# Affichez le résultat
print("Les DataFrames immatriculation_client et merged_data sont-ils égaux ?", sont_egaux)
```

```
Nombre de lignes différentes entre les colonnes sélectionnées de immatriculation_client et merged_data : 0  
Les DataFrames immatriculation_client et merged_data sont-ils égaux ? True
```

Selon les résultats, nous confirmons que les données des immatriculations du dataframe "merged\_data" sont cohérentes avec celles du dataframe "immatriculations\_clients". À présent, nous sélectionnons les données d'immatriculation à partir du dataframe "merged\_data" dont nous aurons besoin pour appliquer le Clustering à ces données.

```

import pandas as pd

columns=['model_immatriculations_clients.marque','model_immatriculations_clients.nom','model_immatriculations_clients.puissance','model_imma'

# Sélectionner les colonnes spécifiées dans merged_data
selected_data = merged_data[columns].reset_index(drop=True)

selected_data

```

|               | model_immatriculations_clients.marque | model_immatriculations_clients.nom | model_immatriculations_clients.puissance | model_imma |
|---------------|---------------------------------------|------------------------------------|------------------------------------------|------------|
| 0             | Audi                                  | A2 1.4                             |                                          | 75         |
| 1             | Peugeot                               | 1007 1.4                           |                                          | 75         |
| 2             | Peugeot                               | 1007 1.4                           |                                          | 75         |
| 3             | Renault                               | Vel Satis 3.5 V6                   |                                          | 245        |
| 4             | BMW                                   | M5                                 |                                          | 507        |
| ...           | ...                                   | ...                                |                                          | ...        |
| <b>199938</b> | Volkswagen                            | Polo 1.2 6V                        |                                          | 55         |
| <b>199939</b> | Audi                                  | A2 1.4                             |                                          | 75         |
| <b>199940</b> | Mercedes                              | A200                               |                                          | 136        |
| <b>199941</b> | Ford                                  | Mondeo 1.8                         |                                          | 125        |
| <b>199942</b> | Peugeot                               | 1007 1.4                           |                                          | 75         |

199943 rows × 12 columns

#### - Analyse Exploratoire sur les variables sélectionnées:

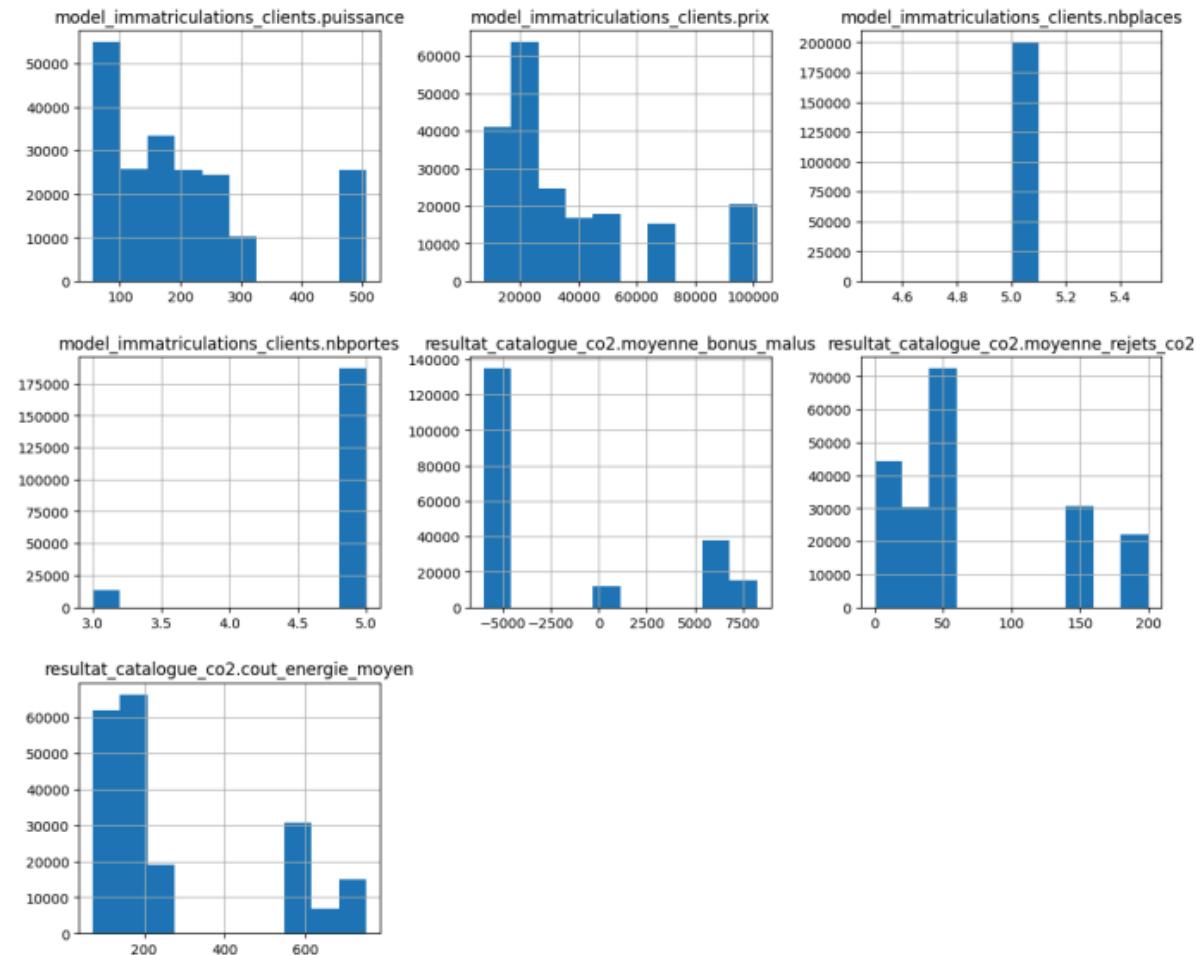
Procérons de la même manière à l'exploration des données d'immatriculations de la dataframe "merged\_data" comme nous l'avons fait pour celles du catalogue.

```
import matplotlib.pyplot as plt

numeric_data = merged_data[['model_immatriculations_clients.puissance', 'model_immatriculations_clients.prix', 'model_immatriculations_clients.nbplaces', 'model_immatriculations_clients.nbportes', 'resultat_catalogue_co2.moyenne_bonus_malus', 'resultat_catalogue_co2.moyenne_rejets_co2', 'resultat_catalogue_co2.cout_energie_moyen']]

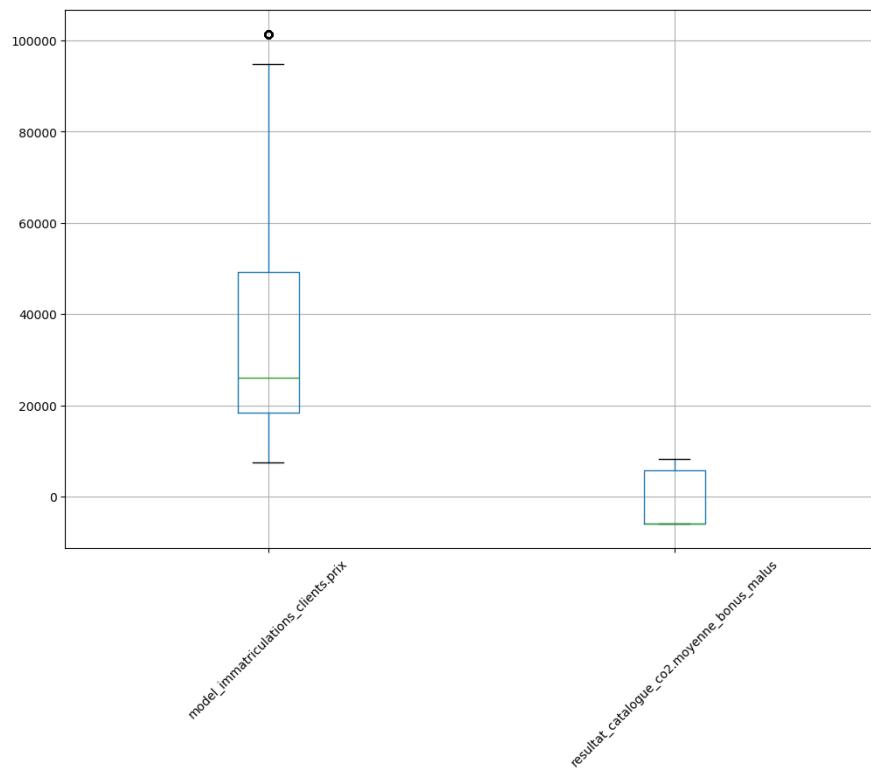
# Effectuer une analyse univariée pour toutes les variables numériques avec une taille de figure plus grande
numeric_data.hist(figsize=(14, 12))
plt.suptitle("Distribution des variables numériques")
plt.show()

Matplotlib created a temporary cache directory at /tmp/matplotlib-59euexdu because the default path (/home/vagrant/.config/matplotlib) is full
```



Les interprétations sont similaires à celles des données du catalogue, à l'exception de la puissance qui varie entre 30 et 100 chevaux et d'une moyenne d'émissions de CO2 de 50 g/km.

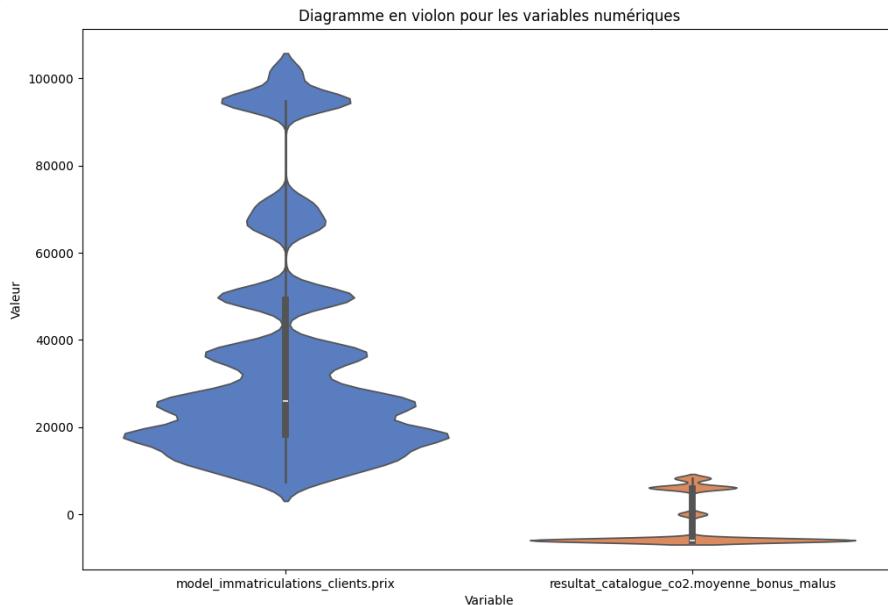
```
# Boîtes à moustaches pour les variables continues
numeric_data[['model_immatriculations_clients.prix', 'resultat_catalogue_co2.moyenne_bonus_malus']].boxplot(figsize=(12, 8))
plt.xticks(rotation=45)
plt.show()
```



Comme vous pouvez le constatez, les interprétations de la boîte à moustache ci-dessus sont similaires à celles des données du catalogue.

```
import seaborn as sns
columns=['model_immatriculations_clients.prix','resultat_catalogue_co2.moyenne_bonus_malus']
# Définir la taille de la figure
plt.figure(figsize=(12, 8))

# Tracer le diagramme en violon
sns.violinplot(x="variable", y="value", data=numeric_data[columns].melt(), palette="muted")
plt.title("Diagramme en violon pour les variables numériques")
plt.xlabel("Variable")
plt.ylabel("Valeur")
plt.show()
```



Selon le boxplot et le diagramme en violon, les observations sont les suivantes :

- Pour la colonne "Prix" des immatriculations, le premier quartile est d'environ 19 000 euros, le troisième quartile d'environ 48 000 euros et la médiane de 25 000 euros, avec une plage allant de 10 000 euros (minimum) à 92 000 euros (maximum) et une valeur aberrante d'environ 105 000 euros.
- Pour la colonne "Moyenne Bonus-Malus", l'interprétation reste la même que pour les données du catalogue.

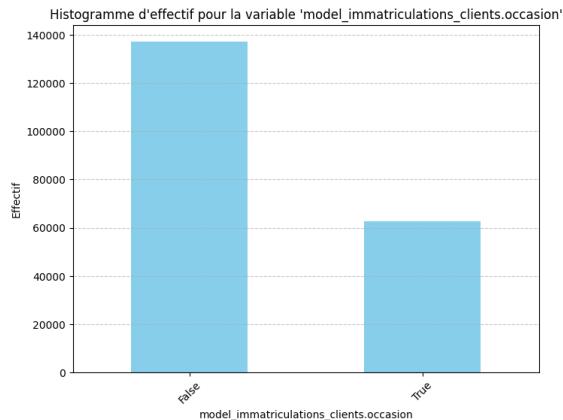
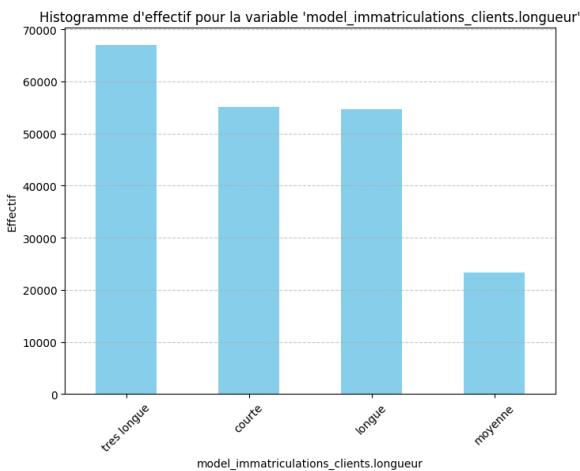
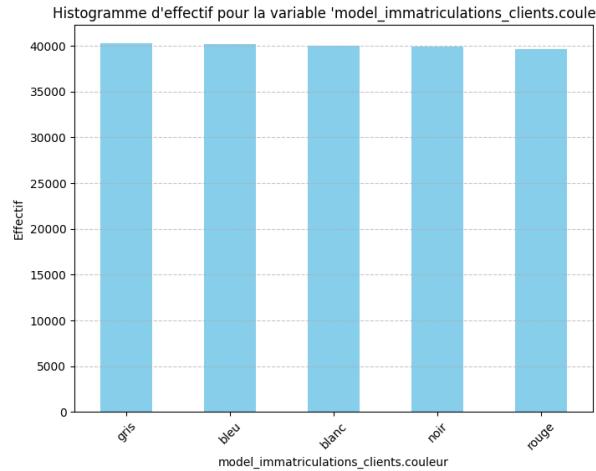
```

import matplotlib.pyplot as plt

# Liste des noms de colonnes catégoriques
categorical_columns = ['model_immatriculations_clients.couleur',
                       'model_immatriculations_clients.longueur',
                       'model_immatriculations_clients.occasion']

# Création d'histogrammes d'effectif pour chaque variable catégorique
for column in categorical_columns:
    plt.figure(figsize=(8, 6))
    merged_data[column].value_counts().plot(kind='bar', color='skyblue')
    plt.title(f"Histogramme d'effectif pour la variable '{column}'")
    plt.xlabel(column)
    plt.ylabel("Effectif")
    plt.xticks(rotation=45) # Rotation des étiquettes pour une meilleure lisibilité
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()

```



Les occurrences pour chaque couleur sont relativement similaires, ce qui indique une répartition équilibrée des couleurs dans le dataset. Cela suggère que le dataset contient une variété de voitures de différentes couleurs, sans qu'une couleur ne domine de manière significative.

Il est noté une préférence générale pour les véhicules de longueur plus longue, avec une tendance à la dégradation de la préférence à mesure que la longueur diminue. Les voitures d'une longueur

très longue sont moins désirées. Cette tendance pourrait être liée à divers facteurs tels que la perception de l'espace intérieur, la capacité de transport de passagers et de marchandises, ou même des considérations esthétiques.

Il est observé aussi que la majorité des voitures répertoriées dans le catalogue ne sont pas d'occasion.

```
# Statistiques descriptives
print(numeric_data.describe())

    model_immatriculations_clients.puissance \
count                199943.00000
mean                 198.576104
std                  136.902069
min                  55.000000
25%                 75.000000
50%                 150.000000
75%                 245.000000
max                  507.000000

    model_immatriculations_clients.prix \
count                199943.00000
mean                 35702.849722
std                  25742.400977
min                  7500.000000
25%                 18310.000000
50%                 25970.000000
75%                 49200.000000
max                  101300.000000

    model_immatriculations_clients.nbplaces \
count                199943.0
mean                  5.0
std                   0.0
min                  5.0
25%                  5.0
50%                  5.0
75%                  5.0
max                  5.0

    model_immatriculations_clients.nbportes \
count                199943.00000
mean                  4.868583
std                   0.495546
min                  3.000000
25%                  5.000000
50%                  5.000000
75%                  5.000000
max                  5.000000

    resultat_catalogue_co2.moyenne_bonus_malus \
count                199943.00000
mean                 -2282.533813
std                  5583.564452
min                 -6000.000000
25%                 -6000.000000
50%                 -6000.000000
75%                 5802.400000
max                  8237.358000

    resultat_catalogue_co2.moyenne_rejets_co2 \
count                199943.00000
mean                  66.297137
std                  66.443686
min                  0.000000
25%                 31.667000
50%                 43.500000
75%                 158.495000
```

En se basant sur ces résultats, nous pouvons tirer les conclusions suivantes :

- La puissance des véhicules immatriculés varie considérablement, allant de 55 à 507 chevaux. La moyenne de puissance est estimée à 198.58 chevaux, avec un écart-type de 136.90. Cela suggère une grande diversité dans la gamme de puissance des véhicules enregistrés.
- Les prix des véhicules immatriculés présentent également une grande variabilité. Le prix minimum observé est de 7 500 euros, tandis que le prix maximum est de 101 300 euros. En moyenne, le prix des véhicules est d'environ 35 702.85 euros, avec un écart-type de 25 742.40 euros.
- Tous les véhicules immatriculés ont cinq places, ce qui suggère une uniformité dans ce domaine. Le nombre moyen de places est également de 5, avec un écart-type de 0, ce qui indique une cohérence dans cette caractéristique des véhicules enregistrés.
- Concernant le nombre de portes, la variation est moindre par rapport à d'autres caractéristiques. Le nombre de portes varie principalement entre 3 et 5, avec une moyenne de 4.87 portes et un écart-type de 0.50. Cela suggère une prédominance des véhicules à 4 ou 5 portes dans l'échantillon.
- La moyenne du bonus-malus se situe autour de -2282.53 euros, avec des valeurs allant de -6000 à 8237.36 euros. Cette large plage de valeurs suggère des politiques de bonus-malus variables pour les véhicules immatriculés. L'écart-type élevé de 5583.56 euros indique une grande dispersion des données autour de la moyenne.
- Les rejets de CO2 présentent également une grande variabilité, avec une moyenne d'environ 66.30 g/km. Les valeurs vont de 0 g/km à 200 g/km, avec un écart-type de 66.44 g/km. Cela suggère des différences significatives dans les niveaux d'émissions de CO2 des véhicules immatriculés.
- Le coût moyen de l'énergie est d'environ 281.44 euros, avec des valeurs comprises entre 72.73 et 749.98 euros. L'écart-type de 221.66 euros suggère une variabilité substantielle dans les coûts associés à l'énergie des véhicules enregistrés.

```
# Traitement des valeurs manquantes
missing_values = numeric_data.isnull().sum()
print("Valeurs manquantes :\n", missing_values)

Valeurs manquantes :
model_immatriculations_clients.puissance      0
model_immatriculations_clients.prix            0
model_immatriculations_clients.nbplaces        0
model_immatriculations_clients.nbportes        0
model_immatriculations_clients.occasion        0
resultat_catalogue_co2.moyenne_bonus_malus    0
resultat_catalogue_co2.moyenne_rejets_co2     0
resultat_catalogue_co2.cout_energie_moyen     0
dtype: int64
```

Les données d'immatriculations du modèle d'analyse sont complètes, ne présentant aucune valeur manquante, et sont prêtes à être utilisées pour être appliquées à un modèle K-means de K=5.

- **Prétraitement des données pour l'analyse de clustering :** Encodage et conversion des variables catégorielles et binaires dans les données sélectionnées.

```
# Encdez la variable 'couleur' en variables indicatrices
couleur_encoded = pd.get_dummies(selected_data['model_immatriculations_clients.couleur'], prefix='couleur').astype(int)

# Encdez la variable 'longueur' en variables indicatrices
longueur_encoded = pd.get_dummies(selected_data['model_immatriculations_clients.longueur'], prefix='longueur').astype(int)

# Convertir la variable occasion en entiers (0 ou 1)
selected_data['model_immatriculations_clients.occasion'] = selected_data['model_immatriculations_clients.occasion'].astype(int)

# Concaténer toutes les colonnes de catalogue_data avec les encodages
selected_data = pd.concat([selected_data, couleur_encoded, longueur_encoded], axis=1)

# Supprimer les colonnes 'resultat_catalogue_co2.longueur' et 'resultat_catalogue_co2.couleur'
selected_data.drop(['model_immatriculations_clients.marque', 'model_immatriculations_clients.nom', 'model_immatriculations_clients.longueur'], axis=1, inplace=True)

selected_data
```

|        | model_immatriculations_clients.puissance | model_immatriculations_clients.prix |
|--------|------------------------------------------|-------------------------------------|
| 0      | 75                                       | 18310                               |
| 1      | 75                                       | 9625                                |
| 2      | 75                                       | 9625                                |
| 3      | 245                                      | 49200                               |
| 4      | 507                                      | 94800                               |
| ...    | ...                                      | ...                                 |
| 199938 | 55                                       | 12200                               |
| 199939 | 75                                       | 18310                               |
| 199940 | 136                                      | 25900                               |
| 199941 | 125                                      | 23900                               |
| 199942 | 75                                       | 13750                               |

199943 rows × 17 columns

Nous appliquons la méthode de codage One-Hot Encoding aux variables catégoriques "model\_immatriculations\_client.couleur" et "model\_immatriculations\_client.longueur" en utilisant la fonction `get_dummies()`. Nous convertissons également la colonne "model\_immatriculations\_client.occasion" en valeurs numériques puisqu'elle était initialement de type booléen. Ensuite, nous fusionnons les variables encodées avec la dataframe "selected\_data". Enfin, nous supprimons les variables catégoriques non significatives telles que "model\_immatriculations\_client.nom" et "model\_immatriculations\_client.marque".

- Application de l'algorithme KMeans pour effectuer une analyse de clustering et mapper les étiquettes des clusters aux données merged\_data, issues des ensembles de données d'immatriculation et de catalogue.

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
import numpy as np

# Définir le nombre de clusters
n_clusters = 5

# Initialiser l'objet KMeans
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
```

```
# Ajuster le modèle de clustering KMeans aux données
kmeans.fit(selected_data)

# Obtenir les centres des clusters
cluster_centers = kmeans.cluster_centers_

# Obtenir les étiquettes des clusters pour chaque point de données
cluster_labels = kmeans.labels_

# Ajout des étiquettes de cluster à la datafram
merged_data['Cluster'] = cluster_labels

# Création du dictionnaire pour mapper les numéros de cluster aux types de véhicules
cluster_type_mapping = {
    0: 'Familiale',
    1: 'Prestige',
    2: 'Sportive',
    3: 'Économique',
    4: 'Citadine'
}

# Ajout de la colonne 'Type' à partir du mapping des clusters
merged_data['Catégorie'] = merged_data['Cluster'].map(cluster_type_mapping)

# Affichage des types de véhicules prédits pour chaque ligne
merged_data
```

|        | model_immatriculations_clients.immatriculation | model_immatriculations_clients. |
|--------|------------------------------------------------|---------------------------------|
| 0      | 0 BZ 21                                        |                                 |
| 1      | 0 CQ 77                                        | F                               |
| 2      | 0 DQ 29                                        | F                               |
| 3      | 0 JO 29                                        |                                 |
| 4      | 0 NK 32                                        |                                 |
| ...    | ...                                            |                                 |
| 199938 | 9999 GH 10                                     | Volk                            |
| 199939 | 9999 KX 26                                     |                                 |
| 199940 | 9999 LG 27                                     | Mk                              |
| 199941 | 9999 NG 65                                     |                                 |
| 199942 | 9999 NQ 33                                     | F                               |

199943 rows × 23 columns

Affichons les noms des voitures associées à chaque Cluster:

```
# Boucle pour sélectionner les noms de voiture distincts pour chaque catégorie
for cat_id, cat_name in cluster_type_mapping.items():
    # Sélection des voitures de la catégorie actuelle
    voitures_categorie = merged_data[merged_data['Catégorie'] == cat_name]
    # Affichage des noms de voiture distincts de cette catégorie
    print(f"Noms de voiture pour la catégorie '{cat_name}':")
    print(voitures_categorie['model_immatriculations_clients.nom'].unique())
    print()
```

Noms de voiture pour la catégorie 'Familiale':  
 ['X-Type 2.5 V6' 'A3 2.0 FSI' '120i' 'Vel Satis 3.5 V6' '9.3 1.8T'  
 'S80 T6' 'Laguna 2.0T' 'Superb 2.8 V6' 'New Beetle 1.8']

Noms de voiture pour la catégorie 'Prestige':  
 ['M5' 'SS00']

Noms de voiture pour la catégorie 'Sportive':  
 ['Vel Satis 3.5 V6' 'S80 T6' 'M5' 'SS00']

Noms de voiture pour la catégorie 'Économique':  
 ['A2 1.4' '1007 1.4' 'Megane 2.0 16V' 'Polo 1.2 6V' 'Laguna 2.0T'  
 'Golf 2.0 FSI' 'Picanto 1.1' 'Cuore 1.0' 'Ypsilon 1.4 16V'  
 'Logan 1.6 MPI' 'Copper 1.6 16V' 'A3 2.0 FSI' 'New Beetle 1.8']

Noms de voiture pour la catégorie 'Citadine':  
 ['Toledo 1.6' 'Mondeo 1.8' 'Croma 2.2' 'Maxima 3.0 V6' '9.3 1.8T'  
 'Almera 1.8' 'A200' 'Primera 1.6' 'Ypsilon 1.4 16V']

- Prédiction des Catégories de voitures à partir des données Clients:

```
print(merged_data.dtypes)

model_immatriculations_clients.immatriculation      object
model_immatriculations_clients.marque              object
model_immatriculations_clients.nom                  object
model_immatriculations_clients.puissance          int64
model_immatriculations_clients.longueur            object
model_immatriculations_clients.nbplaces           int64
model_immatriculations_clients.nbpores             int64
model_immatriculations_clients.couleur             object
model_immatriculations_clients.occasion            bool
model_immatriculations_clients.prix               float64
model_immatriculations_clients.age                float64
model_immatriculations_clients.sex                object
model_immatriculations_clients.taux               float64
model_immatriculations_clients.situationfamiliale object
model_immatriculations_clients.nbenfantsacharge   float64
model_immatriculations_clients.deuxiemevoiture    object
resultat_catalogue_co2.marque                   object
resultat_catalogue_co2.nom                      object
resultat_catalogue_co2.moyenne_bonus_malus       float64
resultat_catalogue_co2.moyenne_rejets_co2        float64
resultat_catalogue_co2.cout_energie_moyen       float64
Cluster   int32
Catégorie                                       object
dtype: object
```

Sélection des colonnes spécifiées du model\_immatriculations\_clients:

```
columns=['model_immatriculations_clients.immatriculation','model_immatriculations_clients.age','model_immatriculations_clients.sex','model_immatriculations_clients.taux']

# Sélection des colonnes spécifiées
selected_columns = merged_data[columns].reset_index(drop=True)

# Affichage des données des colonnes sélectionnées
selected_columns
```

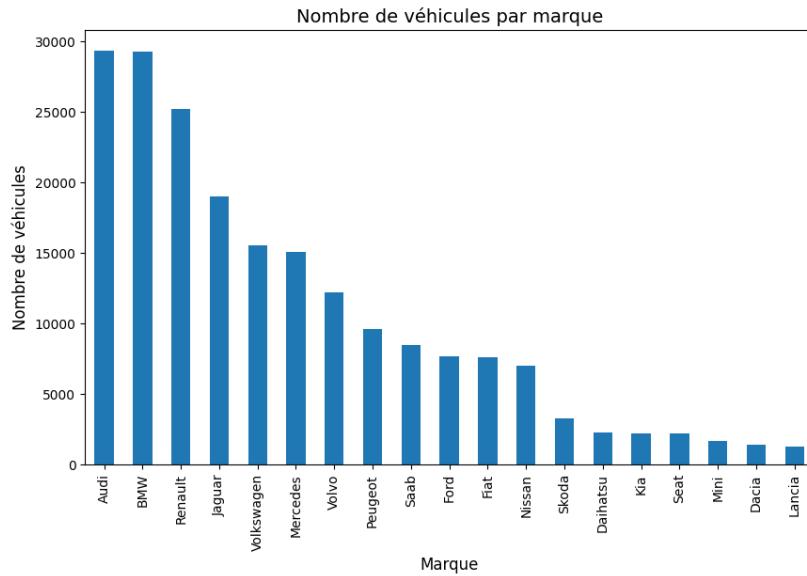
|        | model_immatriculations_clients.immatriculation | model_immatriculations_clients. |
|--------|------------------------------------------------|---------------------------------|
| 0      | 0 BZ 21                                        | 1                               |
| 1      | 0 CQ 77                                        | 1                               |
| 2      | 0 DQ 29                                        | 1                               |
| 3      | 0 JO 29                                        | 1                               |
| 4      | 0 NK 32                                        | 1                               |
| ...    | ...                                            |                                 |
| 199938 | 9999 GH 10                                     | 1                               |
| 199939 | 9999 KX 26                                     | 1                               |
| 199940 | 9999 LG 27                                     | 0                               |
| 199941 | 9999 NG 65                                     | 1                               |
| 199942 | 9999 NQ 33                                     | 1                               |

199943 rows × 9 columns

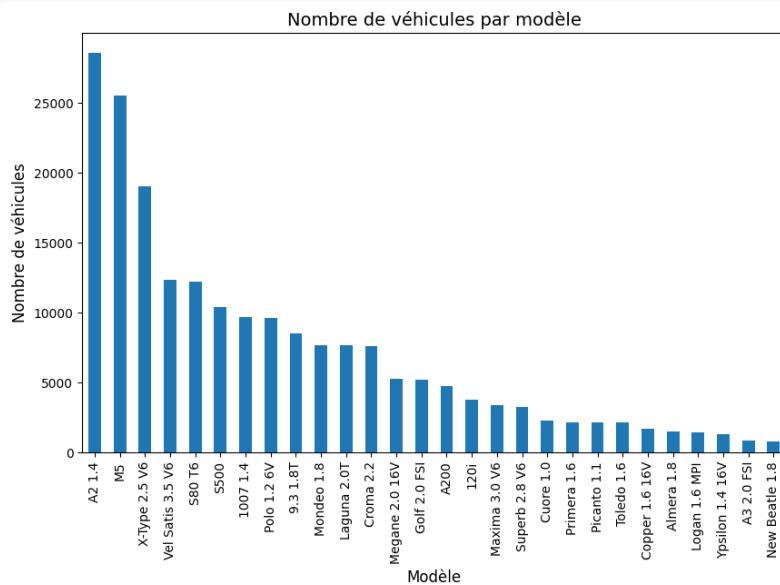
- Exploration des données des clients :

```
import matplotlib.pyplot as plt

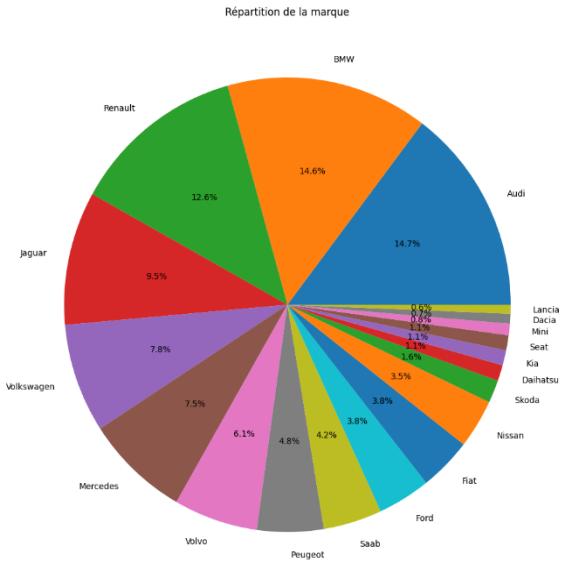
# Diagrammes en barres pour les variables catégorielles
plt.figure(figsize=(10, 6))
merged_data['model_immatriculations_clients.marque'].value_counts().plot(kind='bar')
plt.xlabel('Marque')
plt.ylabel('Nombre de véhicules')
plt.title('Nombre de véhicules par marque')
plt.show()
```



```
# Diagrammes en barres pour les variables catégorielles
plt.figure(figsize=(10, 6))
merged_data['model_immatriculations_clients.nom'].value_counts().plot(kind='bar')
plt.xlabel('Marque')
plt.ylabel('Nombre de véhicules')
plt.title('Nombre de véhicules par modèle')
plt.show()
```



```
# Camembert pour une seule variable catégorielle
plt.figure(figsize=(12, 15))
merged_data['model_immatriculations_clients.marque'].value_counts().plot.pie(autopct='%1.1f%%')
plt.title("Répartition de la marque")
plt.ylabel("")
plt.show()
```



Les graphiques suggèrent que les marques Renault, Audi et BMW sont prédominantes parmi les clients, de même que les modèles A2 1.4, M5 et X-Type 2.5 V6a.

Pretraitement des colonnes spécifiées aux clients :

- Explorations des données relatives aux clients :

```
print(selected_columns.dtypes)

model_immatriculations_clients.immatriculation      object
model_immatriculations_clients.age                  float64
model_immatriculations_clients sexe                 object
model_immatriculations_clients.taux                float64
model_immatriculations_clients.situationfamiliale   object
model_immatriculations_clients.nbenfantsacharge    float64
model_immatriculations_clients.deuxiemevoiture     object
Cluster   int32
Catégorie                                       object
dtype: object

# Pour chaque colonne de la DataFrame
for column in columns:
    # Obtenir les valeurs uniques de la colonne
    unique_values = selected_columns[column].unique()
    # Afficher les valeurs uniques avec le nom de la colonne
    print(f"Colonnes: {column}")
    print(unique_values)

Colonnes: model_immatriculations_clients.immatriculation
['0 BZ 21' '0 CQ 77' '0 DQ 29' ... '9999 LG 27' '9999 NG 65' '9999 NQ 33']
Colonnes: model_immatriculations_clients.age
[56. 27. 51. 39. 75. 41. 31. 50. 38. 44. 69. 30. 48. 26. 29. 19. 35. 42.
 65. 34. 28. 83. 54. 59. 20. 18. 47. 37. 72. 25. nan 32. 21. 23. 45. 78.
 33. 52. 55. 22. 57. 70. 43. 24. 66. 46. 71. 53. 40. 73. 58. 49. 64. 36.
 61. 77. 79. 68. 82. 60. 84. 67. 63. 81. 62. 74. 76. 80.]
Colonnes: model_immatriculations_clients.sex
['M' 'F' 'None']
Colonnes: model_immatriculations_clients.taux
[1382. 239. 234. ... 957. 1043. 923.]
Colonnes: model_immatriculations_clients.situationfamiliale
['Célibataire' 'En Couple' 'None']
Colonnes: model_immatriculations_clients.nbenfantsacharge
[ 0.  1.  3.  2.  4.  nan]
Colonnes: model_immatriculations_clients.deuxiemevoiture
['false' 'true' 'None']
Colonnes: Cluster
[3 2 1 0 4]
Colonnes: Catégorie
['Économique' 'Sportive' 'Prestige' 'Familiale' 'Citadine']
```

Remplacer les valeurs manquantes par le marqueur NaN:

```
import numpy as np

# Remplacer les valeurs nulles par NaN
selected_columns.fillna(np.nan, inplace=True)

# Pour chaque colonne de la DataFrame
for column in columns:
    # Obtenir les valeurs uniques de la colonne
    unique_values = selected_columns[column].unique()
    # Afficher les valeurs uniques avec le nom de la colonne
    print(f"Colonnes: {column}")
    print(unique_values)

Colonnes: model_immatriculations_clients.immatriculation
['0 BZ 21' '0 CQ 77' '0 DQ 29' ... '9999 LG 27' '9999 NG 65' '9999 NQ 33']
Colonnes: model_immatriculations_clients.age
[56. 27. 51. 39. 75. 41. 31. 50. 38. 44. 69. 30. 48. 26. 29. 19. 35. 42.
 65. 34. 28. 83. 54. 59. 20. 18. 47. 37. 72. 25. nan 32. 21. 23. 45. 78.
 33. 52. 55. 22. 57. 70. 43. 24. 66. 46. 71. 53. 40. 73. 58. 49. 64. 36.
 61. 77. 79. 68. 82. 60. 84. 67. 63. 81. 62. 74. 76. 80.]
Colonnes: model_immatriculations_clients.sex
['M' 'F' nan]
Colonnes: model_immatriculations_clients.taux
[1382. 239. 234. ... 957. 1043. 923.]
Colonnes: model_immatriculations_clients.situationfamiliale
['Célibataire' 'En Couple' nan]
Colonnes: model_immatriculations_clients.nbenfantsacharge
[ 0.  1.  3.  2.  4. nan]
Colonnes: model_immatriculations_clients.deuxiemevoiture
['false' 'true' nan]
Colonnes: Cluster
[3 2 1 0 4]
Colonnes: Catégorie
['Économique' 'Sportive' 'Prestige' 'Familiale' 'Citadine']
```

Information sur les colonnes :

```
# Information sur les colonnes
print(selected_columns.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199943 entries, 0 to 199942
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   model_immatriculations_clients.immatriculation  199943 non-null   object 
 1   model_immatriculations_clients.age                199336 non-null   float64 
 2   model_immatriculations_clients.sex                199361 non-null   object 
 3   model_immatriculations_clients.taux              199294 non-null   float64 
 4   model_immatriculations_clients.situationfamiliale 199352 non-null   object 
 5   model_immatriculations_clients.nbenfantsacharge  199346 non-null   float64 
 6   model_immatriculations_clients.deuxiemevoiture   199526 non-null   object 
 7   Cluster   199943 non-null   int32  
 8   Catégorie                                       199943 non-null   object 
dtypes: float64(3), int32(1), object(5)
memory usage: 13.0+ MB
None
```

Statistiques descriptives:

```
# Statistiques descriptives
print(selected_columns.describe())

model_immatriculations_clients.age \
count          199336.000000
mean           43.791322
std            18.309457
min            18.000000
25%            28.000000
50%            42.000000
75%            57.000000
max            84.000000
```

```

model_immatriculations_clients.taux \
count          199294.000000
mean           609.130461
std            336.114521
min            150.000000
25%           421.000000
50%           522.000000
75%           828.000000
max           1399.000000

model_immatriculations_clients.nbenfantsacharge      Cluster
count          199346.000000  199943.000000
mean           1.247249    2.173134
std            1.384227    1.399213
min            0.000000    0.000000
25%           0.000000    1.000000
50%           1.000000    3.000000
75%           2.000000    3.000000
max           4.000000    4.000000

```

À la lumière des résultats présentés ci-dessus, nous pouvons déduire les conclusions suivantes :

- La colonne "model\_immatriculations\_clients.age" présente un total de 199 336 observations valides. La moyenne d'âge des clients est d'environ 43,79 ans, avec un écart-type de 18,31 ans. L'âge minimum observé est de 18 ans, tandis que l'âge maximum est de 84 ans. Les trois quarts des clients ont un âge inférieur à 57 ans, avec une médiane à 42 ans.
- Pour la colonne "model\_immatriculations\_clients.taux", nous avons 199 294 observations valides. Le taux moyen est d'environ 609,13, avec un écart-type de 336,11. Le taux minimum est de 150, tandis que le maximum est de 1399. Les trois quarts des clients ont un taux inférieur à 828, avec une médiane à 522.
- Quant à la colonne "model\_immatriculations\_clients.nbenfantsacharge", nous avons 199 346 observations valides. Le nombre moyen d'enfants à charge est d'environ 1,25, avec un écart-type de 1,38. Le nombre minimum d'enfants à charge est de 0, tandis que le maximum est de 4. Les trois quarts des clients ont un nombre d'enfants à charge inférieur à 2, avec une médiane à 1.
- En ce qui concerne la colonne "Cluster", nous avons 199 943 observations valides. Les valeurs de ce cluster varient de 0 à 4.

Traitement des valeurs manquantes:

```

# Traitement des valeurs manquantes
missing_values = selected_columns.isnull().sum()
print("Valeurs manquantes :\n", missing_values)

Valeurs manquantes :
model_immatriculations_clients.immatriculation      0
model_immatriculations_clients.age                  607
model_immatriculations_clients.sex                582
model_immatriculations_clients.taux                649
model_immatriculations_clients.situationfamiliale  591
model_immatriculations_clients.nbenfantsacharge     597
model_immatriculations_clients.deuxiemevoiture     417
Cluster                      0
Catégorie                     0
dtype: int64

```

Vérifier s'il y a des lignes en double:

```

# Vérifier s'il y a des lignes en double
duplicates = selected_columns[selected_columns.duplicated()]
print("Lignes en double :\n", duplicates)

Lignes en double :
Empty DataFrame
Columns: [model_immatriculations_clients.immatriculation, model_immatriculations_clients.age, model_immatriculations_clients.sex, model_immatriculations_clients.taux, model_immatricu...]
Index: []

```

La dataframne ne présente pas de lignes en double, cependant elle comporte environ 3443 valeurs manquantes.

On divise les données sélectionnées (`selected_columns`) en deux parties :

- `categorical_data` : il contient les colonnes de type 'object', c'est-à-dire les colonnes contenant des données catégoriques.
- `numerical_data` : il contient les colonnes numériques spécifiées, à savoir '`model_immatriculations_clients.age`', '`model_immatriculations_clients.taux`', '`model_immatriculations_clients.nbenfantsacharge`'. Ce sont des colonnes contenant des données numériques.

```
categorical_data = selected_columns.select_dtypes(include=['object'])

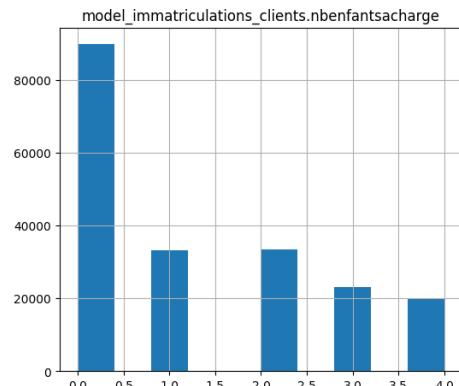
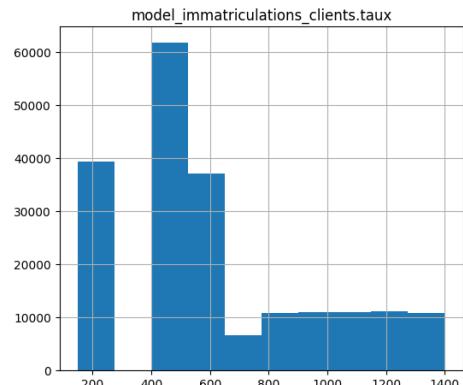
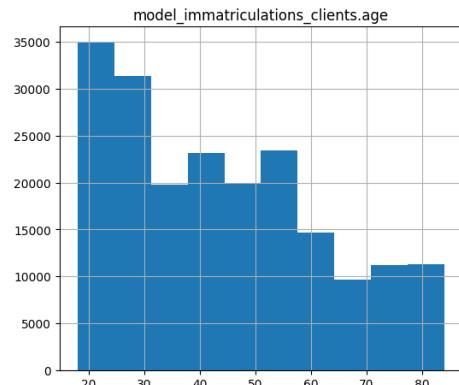
numerical_data = selected_columns[['model_immatriculations_clients.age', 'model_immatriculations_clients.taux', 'model_immatriculations_clients.nbenfantsacharge']]
```

### Explorer la distribution des variable numeriques:

```
import matplotlib.pyplot as plt

# Effectuer une analyse univariée pour toutes les variables numériques avec une taille de figure plus grande
numerical_data.hist(figsize=(14, 12))
plt.suptitle("Histogrammes des variables numériques")
plt.show()
```

Histogrammes des variables numériques



En analysant les graphiques fournis, on remarque que la majorité des clients ont un âge compris entre 20 et 30 ans, un taux situé approximativement entre 400 et 550 euros, et ne déclarent pas avoir d'enfants.

Exploration de la distribution des variables numérique pour chaque catégorie de la variable 'Catégorie':

```
import matplotlib.pyplot as plt
import seaborn as sns

# Fusionner les dataframes numerical_data et categorical_data sur l'index
merged_data = numerical_data.join(categorical_data['Catégorie'])

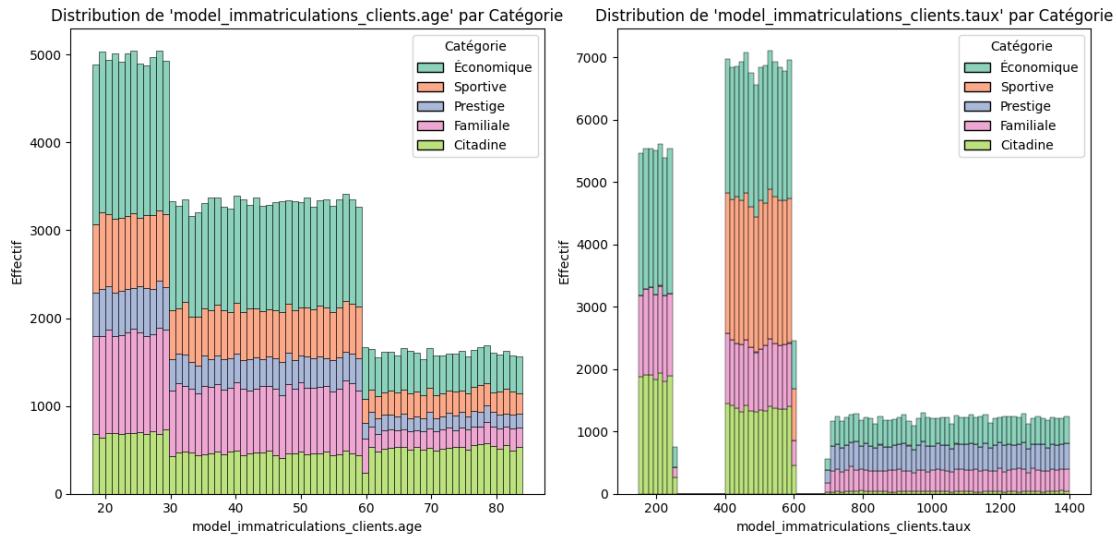
# Créer une grille de sous-graphiques pour chaque variable numérique
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

# Variables numériques à inclure dans le tracé
numerical_columns = ['model_immatriculations_clients.age', 'model_immatriculations_clients.taux']

# Boucle sur chaque variable numérique
for i, column in enumerate(numerical_columns):
    # Tracé de la distribution pour chaque catégorie de la variable 'Catégorie'
    sns.histplot(data=merged_data, x=column, hue='Catégorie', ax=axes[i], multiple='stack', palette='Set2')
    axes[i].set_title(f"Distribution de '{column}' par Catégorie")
    axes[i].set_xlabel(column)
    axes[i].set_ylabel("Effectif")

# Ajustement de l'espacement entre les sous-graphiques
plt.tight_layout()

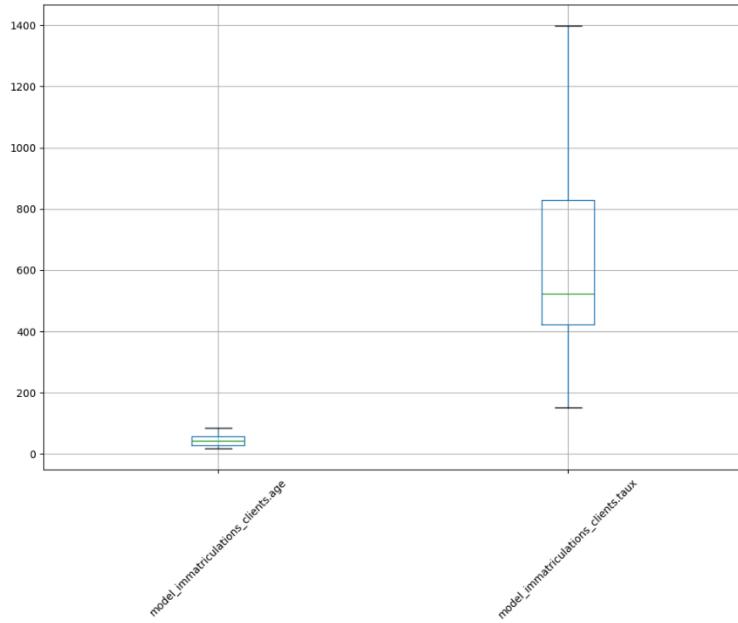
# Affichage du graphique
plt.show()
```



En se basant sur les graphiques précédents, une conclusion significative émerge : la majorité des clients présentent une nette préférence pour les voitures économiques, que ce soit en fonction de leur âge ou de leur taux.

Boîtes à moustaches pour les variables continues:

```
# Boîtes à moustaches pour les variables continues
numerical_data[['model_immatriculations_clients.age', 'model_immatriculations_clients.taux']].boxplot(figsize=(12, 8))
plt.xticks(rotation=45)
plt.show()
```



```
import seaborn as sns
import matplotlib.pyplot as plt

# Définir la taille de la figure
plt.figure(figsize=(8, 10))

# Tracer le diagramme en violon
sns.violinplot(x="variable", y="value", data=numerical_data[['model_immatriculations_clients.age', 'model_immatriculations_clients.taux']])
plt.title("Diagramme en violon pour les variables numériques")
plt.xlabel("Variable")
plt.ylabel("Valeur")
plt.show()
```

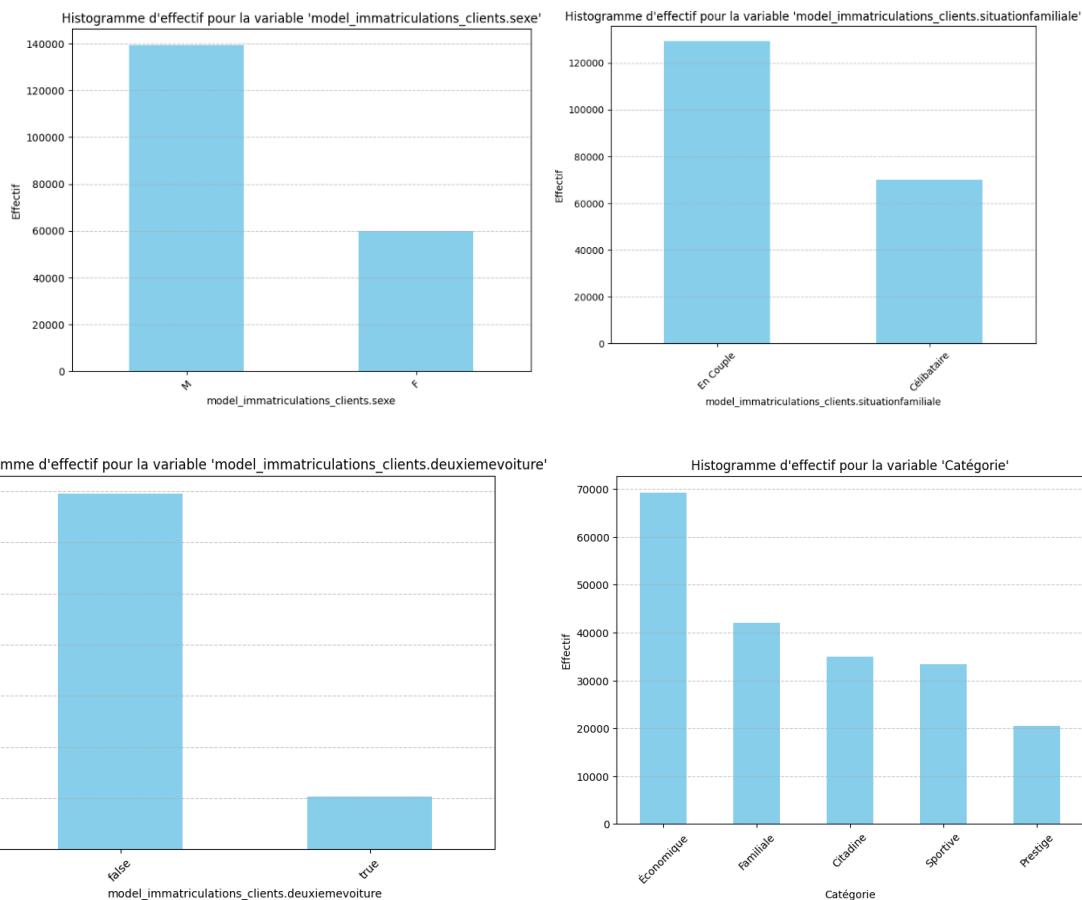
Le boxplot et le diagramme en violon fournissent des informations pertinentes sur les variables continues des données clients. Pour la variable "model\_immatriculations\_clients.taux", on observe un taux minimum d'environ 170 euros et un taux maximum de 1400 euros, avec une médiane de 500 euros, un premier quartile d'environ 420 euros et un troisième quartile de 820 euros. Pour la variable "model\_immatriculations\_clients.age", on constate un âge minimum d'environ 20 ans et un âge maximum d'environ 80 ans, avec une médiane d'environ 30 ans, un premier quartile de 25 ans et un troisième quartile de 40 ans.

### Exploration de la distribution des variables catégorique:

```
import matplotlib.pyplot as plt

# Liste des noms de colonnes catégoriques
categorical_columns = ['model_immatriculations_clients.sexe',
                       'model_immatriculations_clients.situationfamiliale',
                       'model_immatriculations_clients.deuxiemevoiture',
                       'Catégorie']

# Crédation d'histogrammes d'effectif pour chaque variable catégorique
for column in categorical_columns:
    plt.figure(figsize=(8, 6))
    categorical_data[column].value_counts().plot(kind='bar', color='skyblue')
    plt.title(f"Histogramme d'effectif pour la variable '{column}'")
    plt.xlabel(column)
    plt.ylabel("Effectif")
    plt.xticks(rotation=45) # Rotation des étiquettes pour une meilleure lisibilité
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
```



En observant la répartition des valeurs catégoriques, nous pouvons en conclure que la majorité des clients sont des hommes, en couple, n'ont pas de deuxième voiture et préfèrent les voitures économiques.

Exploration de la distribution des variables catégorique pour chaque catégorie de la variable 'Catégorie':

```
import matplotlib.pyplot as plt

# Effectif de chaque variable catégorique
sex_counts = categorical_data['model_immatriculations_clients.sex'].value_counts()
situation_counts = categorical_data['model_immatriculations_clients.situationfamiliale'].value_counts()
voiture_counts = categorical_data['model_immatriculations_clients.deuxiemevoiture'].value_counts()
category_counts = categorical_data['Catégorie'].value_counts()

# Création du graphique à barres empilées
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# Sexe par catégorie
sex_by_category = categorical_data.groupby(['Catégorie', 'model_immatriculations_clients.sex']).size().unstack()
sex_by_category.plot(kind='bar', stacked=True, ax=axes[0, 0], color=['skyblue', 'pink'])
axes[0, 0].set_title('Répartition du sexe par catégorie')
axes[0, 0].set_xlabel('Catégorie')
axes[0, 0].set_ylabel('Effectif')
axes[0, 0].legend(title='Sexe', loc='upper right')
axes[0, 0].grid(axis='y', linestyle='--', alpha=0.7)
```

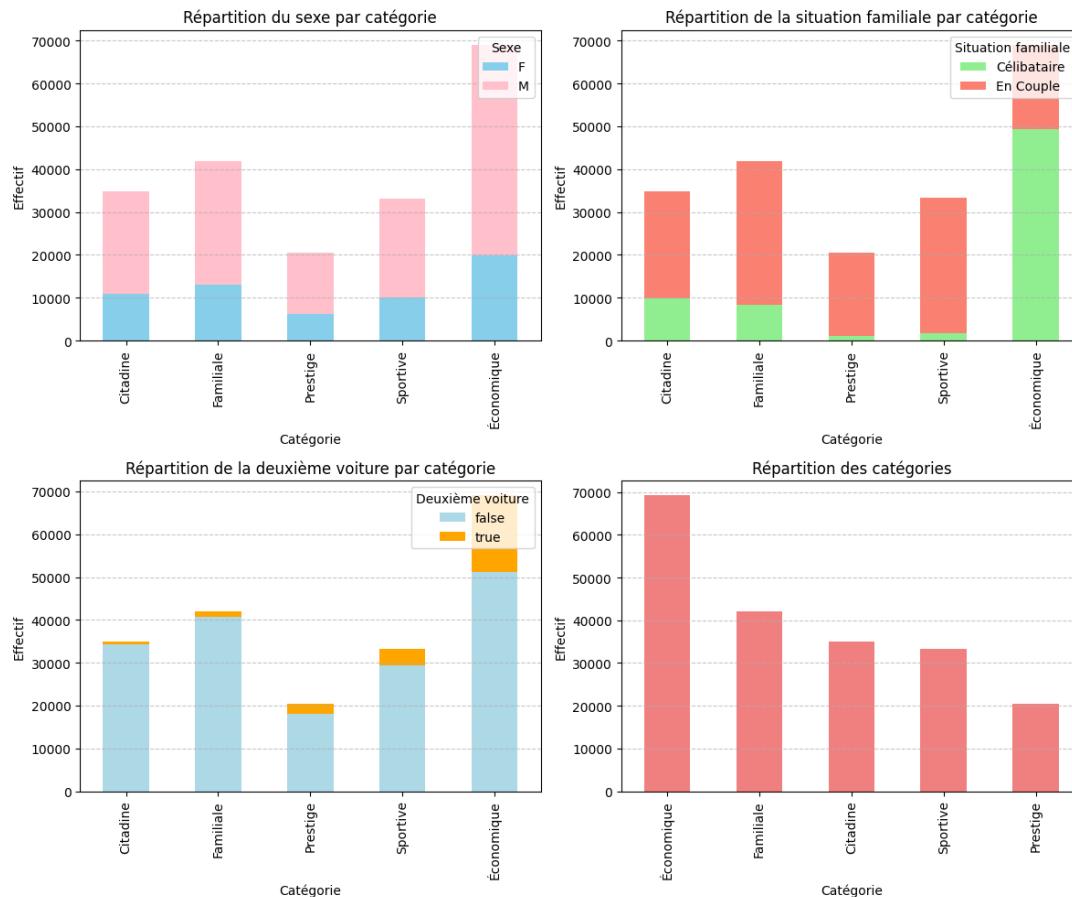
```
# Situation familiale par catégorie
situation_by_category = categorical_data.groupby(['Catégorie', 'model_immatriculations_clients.situationfamiliale']).size().unstack()
situation_by_category.plot(kind='bar', stacked=True, ax=axes[0, 1], color=['lightgreen', 'salmon'])
axes[0, 1].set_title('Répartition de la situation familiale par catégorie')
axes[0, 1].set_xlabel('Catégorie')
axes[0, 1].set_ylabel('Effectif')
axes[0, 1].legend(title='Situation familiale', loc='upper right')
axes[0, 1].grid(axis='y', linestyle='--', alpha=0.7)

# Deuxième voiture par catégorie
voiture_by_category = categorical_data.groupby(['Catégorie', 'model_immatriculations_clients.deuxiemevoiture']).size().unstack()
voiture_by_category.plot(kind='bar', stacked=True, ax=axes[1, 0], color=['lightblue', 'orange'])
axes[1, 0].set_title('Répartition de la deuxième voiture par catégorie')
axes[1, 0].set_xlabel('Catégorie')
axes[1, 0].set_ylabel('Effectif')
axes[1, 0].legend(title='Deuxième voiture', loc='upper right')
axes[1, 0].grid(axis='y', linestyle='--', alpha=0.7)

# Catégorie
category_counts.plot(kind='bar', ax=axes[1, 1], color='lightcoral')
axes[1, 1].set_title('Répartition des catégories')
axes[1, 1].set_xlabel('Catégorie')
axes[1, 1].set_ylabel('Effectif')
axes[1, 1].grid(axis='y', linestyle='--', alpha=0.7)

# Ajustement de l'espacement entre les sous-graphiques
plt.tight_layout()

# Affichage du graphique
plt.show()
```



Les résultats révèlent des informations intéressantes, notamment que la majorité des hommes célibataires préfèrent les voitures économiques et n'ont pas de deuxième voiture.

## Encodage des Données Numériques et Conversion des Variables Catégorielles en Valeurs Numériques :

Sélection de l'ensemble des données des clients dans un DataFrame.

```
numerical_columns=['model_immatriculations_clients.age','model_immatriculations_clients.sex','model_immatriculations_clients.taux','model_i
# Sélection des colonnes spécifiées
numerical_data_encoded=selected_columns[numerical_columns].reset_index(drop=True)

# Affichage des données des colonnes sélectionnées
numerical_data_encoded
```

|        | model_immatriculations_clients.age | model_immatriculations_clients.sex | model_immatriculations_clients.taux | model_immatriculat |
|--------|------------------------------------|------------------------------------|-------------------------------------|--------------------|
| 0      | 56.0                               | M                                  | 1382.0                              |                    |
| 1      | 27.0                               | F                                  | 239.0                               |                    |
| 2      | 51.0                               | M                                  | 234.0                               |                    |
| 3      | 51.0                               | M                                  | 552.0                               |                    |
| 4      | 39.0                               | M                                  | 963.0                               |                    |
| ...    | ...                                | ...                                | ...                                 |                    |
| 199938 | 24.0                               | M                                  | 414.0                               |                    |
| 199939 | 36.0                               | M                                  | 1112.0                              |                    |
| 199940 | 62.0                               | M                                  | 793.0                               |                    |
| 199941 | 34.0                               | M                                  | 467.0                               |                    |
| 199942 | 29.0                               | M                                  | 592.0                               |                    |

99943 rows × 7 columns

La prochaine étape consiste à convertir toutes les features catégoriques en numériques. Nous utiliserons un dictionnaire pour mapper les valeurs catégoriques avec leurs valeurs numériques correspondantes afin de ne pas perdre les valeurs NaN, contrairement à d'autres méthodes telles que le one-hot encoding.

```
# Créer un dictionnaire pour mapper les valeurs
mapping = {'Célibataire': 0, 'En Couple': 1}

# Appliquer le mapping à la colonne 'model_immatriculations_clients.situationfamiliale'
numerical_data_encoded['model_immatriculations_clients.situationfamiliale'] = numerical_data_encoded['model_immatriculations_clients.situati

# Créer un dictionnaire pour mapper les valeurs
mapping = {'M': 0, 'F': 1}

# Appliquer le mapping à la colonne 'model_immatriculations_clients.sex'
numerical_data_encoded['model_immatriculations_clients.sex'] = numerical_data_encoded['model_immatriculations_clients.sex'].map(mapping)

# Créer un dictionnaire pour mapper les valeurs
mapping = {'false': 0, 'true': 1}

# Appliquer le mapping à la colonne 'model_immatriculations_clients.deuxiemevoiture'
numerical_data_encoded['model_immatriculations_clients.deuxiemevoiture'] = numerical_data_encoded['model_immatriculations_clients.deuxiemevc

numerical_data_encoded
```

|        | model_immatriculations_clients.age | model_immatriculations_clients.sexé | model_immatriculations_clients.taux | model_immatricula |
|--------|------------------------------------|-------------------------------------|-------------------------------------|-------------------|
| 0      | 56.0                               | 0.0                                 |                                     | 1382.0            |
| 1      | 27.0                               | 1.0                                 |                                     | 239.0             |
| 2      | 51.0                               | 0.0                                 |                                     | 234.0             |
| 3      | 51.0                               | 0.0                                 |                                     | 552.0             |
| 4      | 39.0                               | 0.0                                 |                                     | 963.0             |
| ...    | ...                                | ...                                 |                                     | ...               |
| 199938 | 24.0                               | 0.0                                 |                                     | 414.0             |
| 199939 | 36.0                               | 0.0                                 |                                     | 1112.0            |
| 199940 | 62.0                               | 0.0                                 |                                     | 793.0             |
| 199941 | 34.0                               | 0.0                                 |                                     | 467.0             |
| 199942 | 29.0                               | 0.0                                 |                                     | 592.0             |

À présent, nous examinons les plages de valeurs et les types des features encodées :

```
#Unique values de la colonne 'deuxiemevoiture'
print(numerical_data_encoded['model_immatriculations_clients.deuxiemevoiture'].unique())

#Unique values de la colonne 'sexé'
print(numerical_data_encoded['model_immatriculations_clients.sexé'].unique())

#Unique values de la colonne 'situationfamiliale'
print(numerical_data_encoded['model_immatriculations_clients.situationfamiliale'].unique())

[ 0.  1. nan]
[ 0.  1. nan]
[ 0.  1. nan]

print(numerical_data_encoded.dtypes)

model_immatriculations_clients.age           float64
model_immatriculations_clients.sexé          float64
model_immatriculations_clients.taux          float64
model_immatriculations_clients.situationfamiliale float64
model_immatriculations_clients.nbenfantsacharge float64
model_immatriculations_clients.deuxiemevoiture float64
Cluster                           int32
dtype: object
```

Examions les relations, les poids et les contributions entre les features des clients et la variable cible, qui est le numéro de cluster à prédire.

Vérifions si il a une présence de multicollinearité parmi les données des clients.

```
!pip install statsmodels

Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: statsmodels in /home/vagrant/.local/lib/python3.9/site-packages (0.14.1)
Requirement already satisfied: numpy<2,>=1.18 in /home/vagrant/.local/lib/python3.9/site-packages (from statsmodels) (1.26.3)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in /home/vagrant/.local/lib/python3.9/site-packages (from statsmodels) (1.13.0)
Requirement already satisfied: pandas!=2.1.0,>=1.0 in /home/vagrant/.local/lib/python3.9/site-packages (from statsmodels) (2.2.1)
Requirement already satisfied: patsy==0.5.4 in /home/vagrant/.local/lib/python3.9/site-packages (from statsmodels) (0.5.6)
Requirement already satisfied: packaging>=21.3 in /home/vagrant/.local/lib/python3.9/site-packages (from statsmodels) (23.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /home/vagrant/.local/lib/python3.9/site-packages (from pandas!=2.1.0,>=1.0->statsmodels)
Requirement already satisfied: pytz>=2020.1 in /home/vagrant/.local/lib/python3.9/site-packages (from pandas!=2.1.0,>=1.0->statsmodels)
Requirement already satisfied: tzdata>=2022.7 in /home/vagrant/.local/lib/python3.9/site-packages (from pandas!=2.1.0,>=1.0->statsmodels)
Requirement already satisfied: six in /home/vagrant/.local/lib/python3.9/site-packages (from patsy>=0.5.4->statsmodels) (1.16.0)
```

```
import pandas as pd
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Filtrer les lignes ne contenant pas de NaN
numerical_data_no_nan = numerical_data_encoded.dropna(axis=0) #filtrer environ 3443 ligne peut-être négligeable puisqu'on possède à peu près

# Calculer le VIF pour chaque fonctionnalité
vif_data = pd.DataFrame()
vif_data["Feature"] = numerical_data_no_nan.columns
vif_data["VIF"] = [variance_inflation_factor(numerical_data_no_nan.values, i) for i in range(len(numerical_data_no_nan.columns))]

# Afficher les VIF
print("\nVariance Inflation Factor (VIF) :")
print(vif_data)

##Multicollinéarité en générale non élevé
```

```
Variance Inflation Factor (VIF) :
          Feature      VIF
0   model_immatriculations_clients.age  5.288967
1   model_immatriculations_clients.sex  1.395221
2   model_immatriculations_clients.taux  3.152209
3 model_immatriculations_clients.situationfamiliale  3.766689
4   model_immatriculations_clients.nbenfantsacharge  2.471532
5   model_immatriculations_clients.deuxiemevoiture  1.289799
6           Cluster  2.904124
```

La sortie affiche les facteurs de l'inflation de la variance (VIF) pour chaque variable. Le VIF mesure à quel point une variable est influencée par les autres variables dans le modèle de régression. Généralement, un VIF supérieur à 10 indique une forte multicollinéarité, ce qui signifie que les variables sont fortement corrélées entre elles, ce qui peut poser problème dans certains modèles de régression.

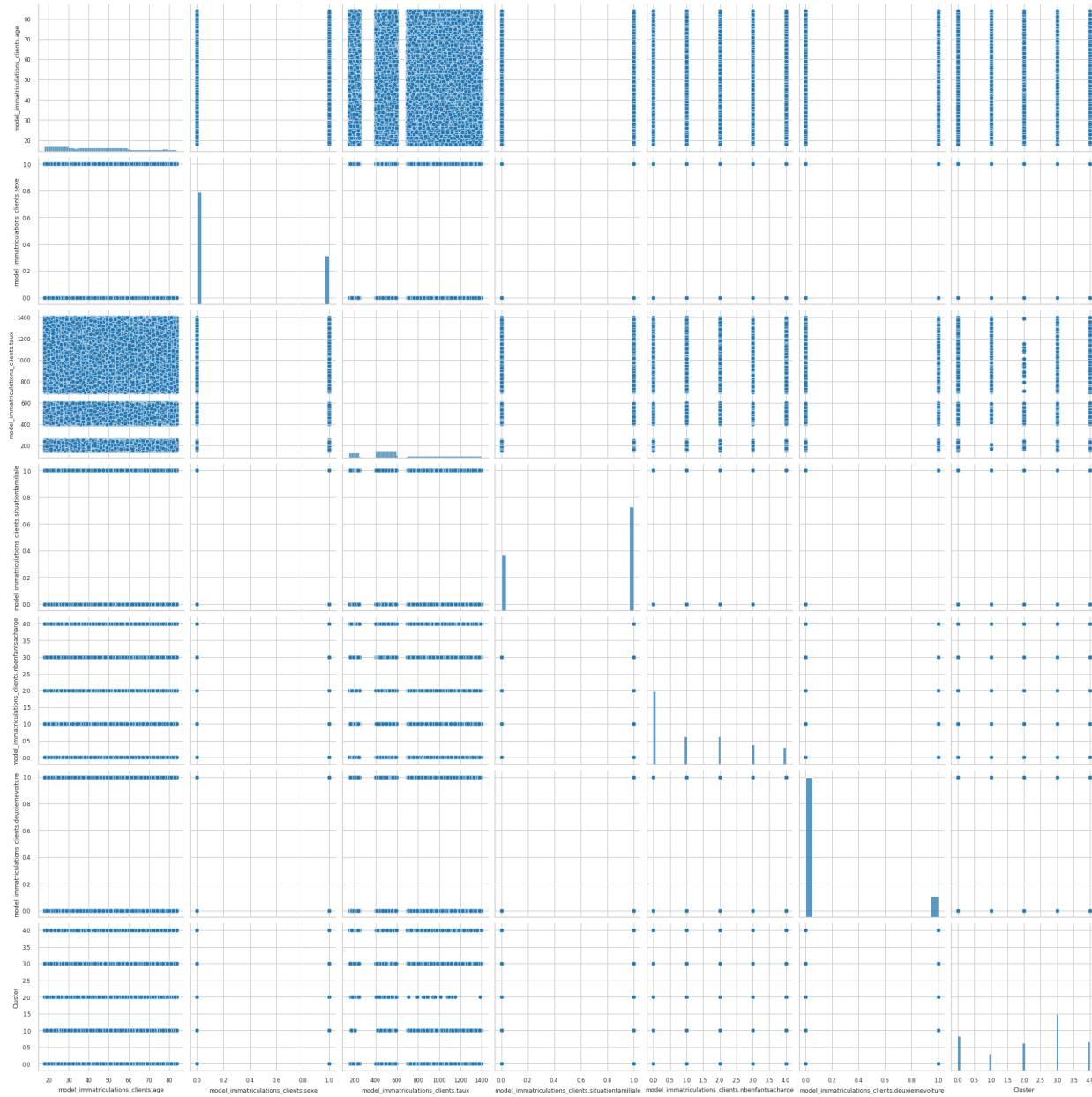
Cependant, dans ce cas, les VIF semblent être généralement bas, ce qui suggère qu'il n'y a pas de forte multicollinéarité entre les variables. Cela signifie que les variables numériques sélectionnées peuvent être considérées comme relativement indépendantes les unes des autres.

Créons une matrice de nuage de points pour examiner la linéarité entre les caractéristiques et la variable cible "Cluster".

```
import seaborn as sns
import matplotlib.pyplot as plt

# Ajuster les paramètres de visualisation
sns.set_context("paper", font_scale=0.7) # Ajuster la taille de police
sns.set_style("whitegrid") # Choisir un style de fond pour le graphique

# Créer une matrice de nuages de points avec des ajustements de paramètres
sns.pairplot(data=numerical_data_encoded)
plt.xticks(rotation=45) # Rotation des étiquettes des axes x pour éviter le chevauchement
plt.yticks(rotation=45) # Rotation des étiquettes des axes y pour éviter le chevauchement
plt.tight_layout() # Ajustement automatique de la disposition pour éviter le chevauchement des sous-graphiques
plt.show()
```



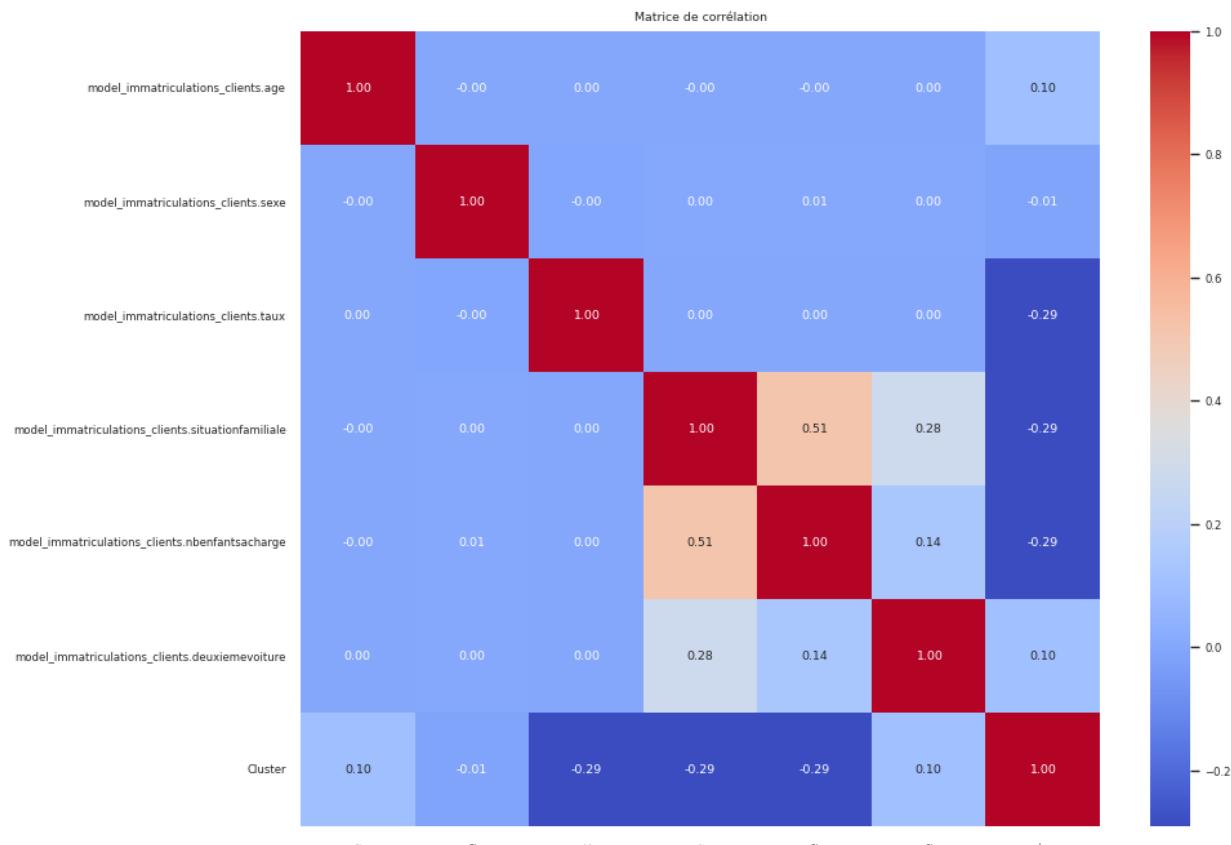
Selon les observations des nuages de points, il semble qu'il n'existe pas de relation linéaire évidente entre les caractéristiques des clients et la variable cible. Nous concluons qu'un modèle de régression ne produirait pas de meilleurs résultats pour prédire les catégories de voitures en fonction des données des clients.

Traçons une heatmap de corrélation pour étudier l'importance des features par rapport à la variable cible :

```
import seaborn as sns
import matplotlib.pyplot as plt

# Calculer la matrice de corrélation uniquement pour les variables numériques
correlation_matrix = numerical_data_encoded.corr()

# Visualiser la matrice de corrélation avec une heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Matrice de corrélation")
plt.show()
```



La heatmap de corrélation révèle plusieurs résultats importants :

- Une corrélation négative de -0.29 entre le nombre d'enfants à charge et la catégorie de voiture achetée, ce qui indique qu'il y a une tendance à acheter certains types de voitures en fonction du nombre d'enfants à charge.
- Une corrélation négative de -0.29 entre la situation familiale et la catégorie de voiture recommandée, ce qui suggère que la situation familiale peut influencer le choix de la voiture recommandée.

- Une corrélation négative de -0.29 entre le taux (capacité d'endettement du client) et la catégorie de voiture indique que le taux peut avoir une influence sur le choix du véhicule .
- Une corrélation positive de 0.10 entre la probabilité pour un client de posséder une deuxième voiture et la catégorie de voiture recommandée suggère que cette circonstance peut influencer la décision d'achat d'un autre véhicule.
- Nous observons une corrélation très faible de -0.01 entre le sexe et la catégorie de voiture, ce qui indique que le sexe n'est pas un facteur déterminant dans le choix de la catégorie de voiture recommandée.

Selon cette analyse, on conclut que la variable "sexe" devrait être exclue. elle n'est pas pertinente pour prédire les catégories de voitures, tandis que les autres variables devraient être conservées.

```
# Suppression de la variable "sexe" dans la dataframe encodés
numerical_data_encoded=numerical_data_encoded.drop(columns=["sexe"])
```

#### 8.4 Crédit de modèles de connaissances et évaluation des résultats

Dans cette phase du projet, nous nous sommes concentrés sur la création d'un modèle de classification supervisée permettant de prédire la catégorie de véhicules associée aux caractéristiques des clients. Nous avons exploité les données issues de la fusion entre les données des clients et des immatriculations pour entraîner et évaluer différents algorithmes de classification.

Nous avons testé plusieurs approches et algorithmes de classification, dont les arbres de décision, les random forests, les réseaux de neurones et les modèles de deep learning. Pour chacun de ces algorithmes, nous avons exploré différentes configurations de paramètres afin d'obtenir un classifieur aussi performant que possible.

L'évaluation et la comparaison des performances de chaque configuration algorithmique ont été réalisées à l'aide de matrices de confusion et de mesures d'évaluation telles que la précision, le rappel et le score F1. Ces métriques nous ont permis d'évaluer la capacité de chaque modèle à prédire avec précision la catégorie de véhicules en fonction des caractéristiques des clients.

En fin de compte, nous avons sélectionné le modèle présentant les meilleures performances . Ce modèle sera utilisé dans les étapes ultérieures du projet pour prédire de manière précise et fiable la catégorie de véhicules la plus appropriée pour chaque client en fonction de ses caractéristiques personnelles.

Nous divisons les données en variables explicatives (X) et cible (Y), puis nous comblons les valeurs manquantes de chaque colonne en utilisant la moyenne respective de chaque colonne. Cette approche assure un équilibre dans les plages de valeurs plutôt que d'utiliser une valeur par défaut. Ensuite, nous normalisons nos données d'entrée contenant les caractéristiques. L'ensemble des ces instructions permet de préparer les données à la modélisation.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

# Diviser les données en variables explicatives et cible
X = numerical_data_encoded.drop(columns=["Cluster"])
y = numerical_data_encoded["Cluster"]

# Gérer les valeurs manquantes
imputer = SimpleImputer(strategy="mean")
X_imputed = imputer.fit_transform(X)

# Normaliser les données
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)
```

Évaluation des performances du modèle de classification KNN :



Nous avons utilisé en premier temps l'algorithme de classification des k plus proches voisins (KNN) pour prédire les catégories de véhicules à partir de données numériques encodées. Ensuite, nous avons utilisé la validation croisée avec différentes valeurs de k pour évaluer les performances du modèle KNN en termes d'accuracy, de précision, de rappel et de score F1.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Définir les valeurs de k pour KNN
k_values = [5, 10, 15, 20, 25, 30, 35, 40]

# Initialiser un dictionnaire pour stocker les résultats
results = {'k': [], 'Accuracy': [], 'Precision': [], 'Recall': [], 'F1 Score': []}

# Numéro du K-Fold:
num_k_fold=15;
```

```
# Parcourir chaque valeur de k
for k in k_values:
    # Initialiser les listes pour stocker les scores de performance pour chaque fold
    accuracy_scores, precision_scores, recall_scores, f1_scores = [], [], [], []

    # Initialiser KFold avec le nombre de folds k
    kf = KFold(n_splits=num_k_fold, shuffle=True, random_state=42)

    # Parcourir chaque fold de KFold
    for train_index, test_index in kf.split(X_imputed):
        X_train, X_test = X_scaled[train_index], X_scaled[test_index]
        y_train, y_test = y[train_index], y[test_index]

        # Initialiser le modèle KNN avec la valeur de k actuelle
        knn = KNeighborsClassifier(n_neighbors=k)

        # Adapter le modèle aux données d'entraînement
        knn.fit(X_train, y_train)

        # Faire des prédictions sur les données de test
        y_pred = knn.predict(X_test)

        # Calculer les métriques de performance pour ce fold
        accuracy_scores.append(accuracy_score(y_test, y_pred))
        precision_scores.append(precision_score(y_test, y_pred, average='weighted'))
        recall_scores.append(recall_score(y_test, y_pred, average='weighted'))
        f1_scores.append(f1_score(y_test, y_pred, average='weighted'))

    # Stocker les résultats moyens pour cette valeur de k
    results['k'].append(k)
    results['Accuracy'].append(np.mean(accuracy_scores))
    results['Precision'].append(np.mean(precision_scores))
    results['Recall'].append(np.mean(recall_scores))
    results['F1 Score'].append(np.mean(f1_scores))

# Convertir les résultats en DataFrame
results_df_knn = pd.DataFrame(results)

# Afficher le DataFrame des résultats
results_df_knn
```

|   | k  | Accuracy | Precision | Recall   | F1 Score |
|---|----|----------|-----------|----------|----------|
| 0 | 5  | 0.709812 | 0.708350  | 0.709812 | 0.707560 |
| 1 | 10 | 0.725152 | 0.728882  | 0.725152 | 0.723446 |
| 2 | 15 | 0.728538 | 0.734890  | 0.728538 | 0.728534 |
| 3 | 20 | 0.731358 | 0.739915  | 0.731358 | 0.730981 |
| 4 | 25 | 0.732229 | 0.742479  | 0.732229 | 0.732736 |
| 5 | 30 | 0.733799 | 0.745451  | 0.733799 | 0.734086 |
| 6 | 35 | 0.734034 | 0.746581  | 0.734034 | 0.734781 |
| 7 | 40 | 0.734179 | 0.746970  | 0.734179 | 0.734656 |

Les résultats montrent que les performances du modèle KNN varient légèrement avec différentes valeurs de k. On remarque que le modèle KNN a atteint une meilleure accuracy d'environ 0.7341 pour k = 40. Cela indique que le modèle peut prédire avec précision les catégories de véhicules dans une certaine mesure. Cependant, il est important de noter que l'augmentation de k entraîne également une augmentation de la complexité du modèle, ce qui peut conduire à un surajustement si k est trop grand.

Voici une présentation des metric utilisés :

- Accuracy (Exactitude) : L'exactitude mesure la proportion de prédictions correctes parmi toutes les prédictions effectuées par le modèle. C'est le rapport entre le nombre total de prédictions correctes (vrais positifs et vrais négatifs) et le nombre total d'observations.
- Précision : La précision mesure la proportion de vrais positifs parmi toutes les observations identifiées comme positives (vrais positifs et faux positifs). Une précision élevée signifie que le modèle a moins de faux positifs, ce qui est important dans les cas où les faux positifs sont coûteux ou indésirables.
- Rappel : Le rappel moyen, qui mesure la proportion de véritables positifs correctement identifiés parmi tous les positifs réels (vrais positifs et faux négatifs). Un rappel élevé est important lorsque les faux négatifs sont coûteux ou indésirables, car il indique que le modèle identifie efficacement les vrais positifs.
- Score F1 : Le score F1 est une mesure de la précision d'un test. Il est calculé comme la moyenne harmonique de la précision et du rappel. La moyenne harmonique donne plus de poids aux valeurs les plus faibles. Ainsi, le score F1 est élevé si à la fois la précision et le rappel sont élevés. Il est particulièrement utile lorsque les classes sont déséquilibrées, car il prend en compte à la fois les faux positifs et les faux négatifs. Le score F1 est souvent utilisé comme métrique de performance globale pour les modèles de classification.

Nous pourrions maintenant explorer les performances du modèle KNN en testant différents paramètres, tout en maintenant un voisinage fixe de k=40. L'objectif est de déterminer si ces ajustements amélioreraient les performances du modèle.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Définir les options pour chaque paramètre à tester
algorithm_options = ['auto', 'ball_tree', 'kd_tree', 'brute'] # Algorithmes pour le calcul des voisins
leaf_size_options = [10, 20, 30, 40] # Taille de la feuille pour l'arbre (pour ball_tree ou kd_tree)

# Numéro k-fold
num_k_fold=15

# Initialiser un dictionnaire pour stocker les résultats
params_results = {'algorithm': [], 'leaf_size': [], 'Accuracy': [], 'Precision': [], 'Recall': [], 'F1 Score': []}

# Parcourir chaque combinaison de paramètres
for algorithm in algorithm_options:
    for leaf_size in leaf_size_options:
        # Initialiser les listes pour stocker les scores de performance pour chaque fold
        accuracy_scores, precision_scores, recall_scores, f1_scores = [], [], [], []

        # Initialiser KFold avec le nombre de folds k
        kf = KFold(n_splits=num_k_fold, shuffle=True, random_state=42)
```

```
# Parcourir chaque fold de KFold
for train_index, test_index in kf.split(X_imputed):
    X_train, X_test = X_scaled[train_index], X_scaled[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Initialiser le modèle KNN avec les valeurs de paramètres actuelles
    knn = KNeighborsClassifier(n_neighbors=40, algorithm=algorithm, leaf_size=leaf_size)

    # Adapter le modèle aux données d'entraînement
    knn.fit(X_train, y_train)

    # Faire des prédictions sur les données de test
    y_pred = knn.predict(X_test)

    # Calculer les métriques de performance pour ce fold
    accuracy_scores.append(accuracy_score(y_test, y_pred))
    precision_scores.append(precision_score(y_test, y_pred, average='weighted'))
    recall_scores.append(recall_score(y_test, y_pred, average='weighted'))
    f1_scores.append(f1_score(y_test, y_pred, average='weighted'))

    # Stocker les résultats moyens pour cette combinaison de paramètres
    params_results['algorithm'].append(algorithm)
    params_results['leaf_size'].append(leaf_size)
    params_results['Accuracy'].append(np.mean(accuracy_scores))
    params_results['Precision'].append(np.mean(precision_scores))
    params_results['Recall'].append(np.mean(recall_scores))
    params_results['F1 Score'].append(np.mean(f1_scores))

# Convertir les résultats en DataFrame
params_results_df_knn = pd.DataFrame(params_results)
params_results_df_knn
```

|    | algorithm | leaf_size | Accuracy | Precision | Recall   | F1 Score |
|----|-----------|-----------|----------|-----------|----------|----------|
| 0  | auto      | 10        | 0.734209 | 0.747004  | 0.734209 | 0.734688 |
| 1  | auto      | 20        | 0.734089 | 0.746940  | 0.734089 | 0.734589 |
| 2  | auto      | 30        | 0.734179 | 0.746970  | 0.734179 | 0.734656 |
| 3  | auto      | 40        | 0.734179 | 0.746970  | 0.734179 | 0.734656 |
| 4  | ball_tree | 10        | 0.734189 | 0.746986  | 0.734189 | 0.734667 |
| 5  | ball_tree | 20        | 0.734259 | 0.747061  | 0.734259 | 0.734744 |
| 6  | ball_tree | 30        | 0.734259 | 0.747069  | 0.734259 | 0.734743 |
| 7  | ball_tree | 40        | 0.734259 | 0.747069  | 0.734259 | 0.734743 |
| 8  | kd_tree   | 10        | 0.734209 | 0.747004  | 0.734209 | 0.734688 |
| 9  | kd_tree   | 20        | 0.734089 | 0.746940  | 0.734089 | 0.734589 |
| 10 | kd_tree   | 30        | 0.734179 | 0.746970  | 0.734179 | 0.734656 |
| 11 | kd_tree   | 40        | 0.734179 | 0.746970  | 0.734179 | 0.734656 |
| 12 | brute     | 10        | 0.734104 | 0.746930  | 0.734104 | 0.734594 |
| 13 | brute     | 20        | 0.734104 | 0.746930  | 0.734104 | 0.734594 |
| 14 | brute     | 30        | 0.734104 | 0.746930  | 0.734104 | 0.734594 |
| 15 | brute     | 40        | 0.734104 | 0.746930  | 0.734104 | 0.734594 |

Les résultats montrent que les performances du modèle KNN ne varient pas de manière significative en fonction des paramètres d'algorithme et la taille de la feuille utilisé dans le model

KNN avec k=40. Le changement de ces paramètres produisent tous des performances similaires en termes d'exactitude, de précision, de rappel et de score F1.

Stabilité des performances : Globalement, les résultats restent cohérents et stables, avec des valeurs de métrique similaires pour toutes les combinaisons de paramètres testées.

Cela suggère que le modèle KNN est robuste et peu sensible aux variations des paramètres d'algorithme et de taille de feuille dans ce contexte spécifique.

En résumé, ces résultats indiquent qu'il n'y a pas de différence significative dans les performances du modèle KNN pour différentes combinaisons de paramètres dans ce cas d'étude spécifique. Il est donc possible de sélectionner n'importe quelle combinaison d'algorithme et de taille de feuille pour construire le modèle final.

- **Optimisation des performances du modèle KNN et sélection de la meilleure combinaison d'hyperparamètre :**

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Diviser l'ensemble de données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Définir les paramètres à tester
param_grid = {
    'n_neighbors': [5, 10, 15, 20, 25, 30, 35, 40],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}

# Créer le modèle KNN
model = KNeighborsClassifier()

# Recherche sur grille pour trouver les meilleurs paramètres
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=15, scoring='accuracy', verbose=1, n_jobs=-1)
grid_search.fit(X_train, y_train)

# Afficher les meilleurs paramètres
print("Meilleurs paramètres:", grid_search.best_params_)

# Utiliser le meilleur modèle pour faire des prédictions sur l'ensemble de test
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Calculer les métriques
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Afficher les résultats
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)

Fitting 15 folds for each of 32 candidates, totalling 480 fits
Meilleurs paramètres: {'metric': 'manhattan', 'n_neighbors': 40, 'weights': 'uniform'}
Accuracy: 0.7361774487984196
Precision: 0.7478596208147643
Recall: 0.7361774487984196
F1 Score: 0.7362986423272094
```

Nous avons utilisé la méthode de recherche sur grille pour trouver les meilleurs paramètres pour le modèle KNN. La recherche sur grille est une technique d'optimisation qui évalue

systématiquement différentes combinaisons de paramètres du modèle et sélectionne celle qui donne les meilleurs résultats de performance se basant sur le métrique d'accuracy (exactitude).

- Nous avons exploré différentes valeurs pour divers hyperparamètres essentiels du modèle KNN : Nombre de voisins (n\_neighbors) : Nous avons testé des valeurs allant de 5 à 40.
- Poids des voisins (weights) : deux options testé, "uniform" où tous les voisins ont le même poids, et "distance" où les poids sont inversément proportionnels à la distance.
- Métrique de distance (metric) : deux métriques de distance testé, "euclidean" et "manhattan".

Meilleurs paramètres trouvés que la recherche sur grille a identifié :

- Métrique de distance : "manhattan"
- Nombre de voisins : 40
- Poids des voisins : "uniform"

En utilisant ces meilleurs paramètres, le modèle KNN a atteint les performances suivantes sur l'ensemble de test :

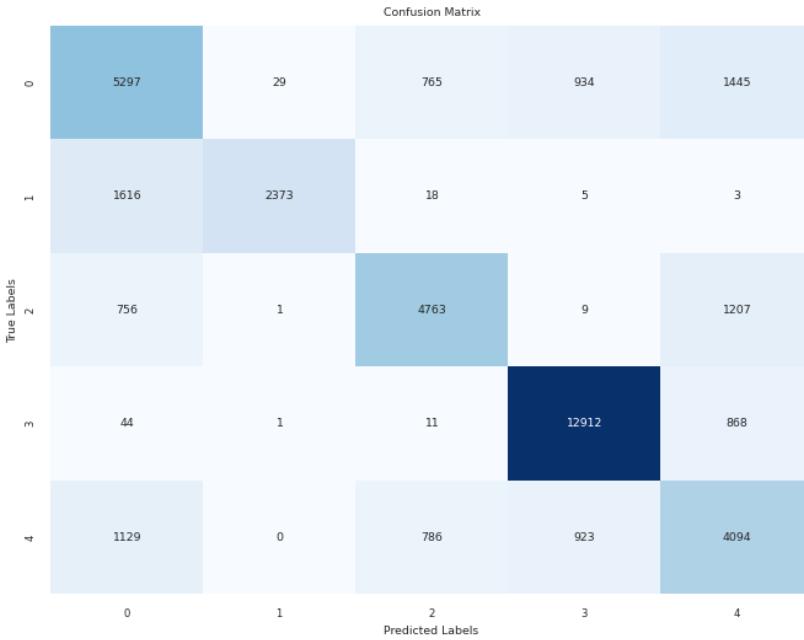
Accuracy: 0.736 Precision: 0.748 Recall: 0.736 F1 Score: 0.736 Ces résultats indiquent que le modèle KNN avec les paramètres optimisés a une performance raisonnable sur l'ensemble de test, avec une accuracy de 73.6%.

### Examinons la matrice de confusion pour évaluer les performances du modèle KNN :

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Calculer la matrice de confusion
conf_matrix = confusion_matrix(y_test, y_pred)

# Afficher la matrice de confusion sous forme de heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



Axe x (Predicted Labels) : Il représente les étiquettes prédites par le modèle.

Axe y (True Labels) : Il représente les étiquettes réelles des données.

Chaque cellule de la matrice contient le nombre d'observations qui appartiennent à la classe représentée par l'étiquette réelle (axe y) et ont été classées dans la classe représentée par l'étiquette prédite (axe x).

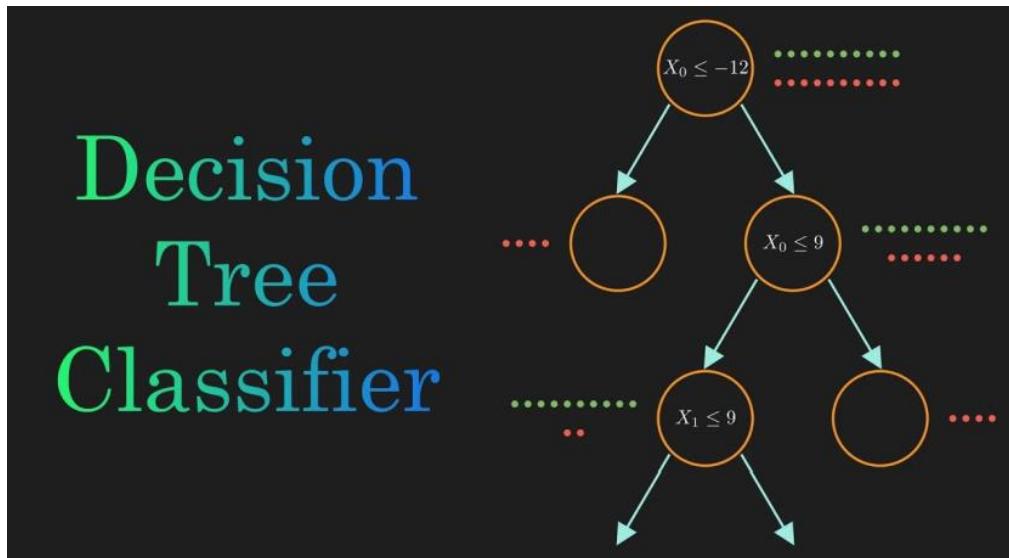
Les valeurs diagonales de la matrice représentent les prédictions correctes, tandis que les valeurs hors diagonale représentent les prédictions incorrectes. Plus les valeurs diagonales sont élevées et les valeurs hors diagonale sont faibles, meilleures sont les performances du modèle.

Dans notre cas on peut dire les valeurs les plus élevées sur la diagonale indiquent que le modèle a correctement classé un nombre important d'observations dans ces classes spécifiques.

- Pour la première classe : 5297 observations ont été correctement prédites contre 3545 incorrects.
- Pour la deuxième classe : 2373 observations ont été correctement prédites contre 31 incorrects.
- Pour la troisième classe : 4763 observations ont été correctement prédites contre 1580 incorrects.
- Pour la quatrième classe : 12912 observations ont été correctement prédites contre 1871 incorrects.
- Pour la cinquième classe : 4094 observations ont été correctement prédites contre 3523 incorrects.

Ces valeurs élevées sur la diagonale indiquent que le modèle a généralement bien performé pour classifier les données dans ces classes.

*Évaluation des performances du modèle d'arbre de décision :*



```
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import make_scorer, accuracy_score, precision_score, recall_score, f1_score

# Définir les métriques
metrics = {
    "Accuracy": make_scorer(accuracy_score),
    "Precision": make_scorer(precision_score, average="weighted"),
    "Recall": make_scorer(recall_score, average="weighted"),
    "F1 Score": make_scorer(f1_score, average="weighted")
}

# Définir les paramètres à tester
params = {
    "max_depth": [5, 7, 10],
    "min_samples_split": [2, 3, 5],
    "min_samples_leaf": [1, 2, 4]
}

# Stocker les résultats dans un DataFrame
results = []

# Définir le nombre de folds K
k = 15
```

```
# Effectuer la validation croisée pour chaque métrique, chaque combinaison de paramètres et k=15
print(f"K-Fold Cross Validation with K={k}")
kf = KFold(n_splits=k, shuffle=True, random_state=42)
for metric_name, scorer in metrics.items():
    print(f"Metric: {metric_name}")
    for max_depth in params["max_depth"]:
        for min_samples_split in params["min_samples_split"]:
            for min_samples_leaf in params["min_samples_leaf"]:
                clf = DecisionTreeClassifier(max_depth=max_depth, min_samples_split=min_samples_split, min_samples_leaf=min_samples_leaf)
                scores = cross_val_score(clf, X_scaled, y, cv=kf, scoring=scorer)
                mean_score = np.mean(scores)
                std_score = np.std(scores)
                results.append({
                    "Metric": metric_name,
                    "max_depth": max_depth,
                    "min_samples_split": min_samples_split,
                    "min_samples_leaf": min_samples_leaf,
                    "Mean Score": mean_score,
                    "Std Score": std_score
                }))
# Convertir les résultats en DataFrame
results_df = pd.DataFrame(results)

K-Fold Cross Validation with K=15
Metric: Accuracy
Metric: Precision
Metric: Recall
Metric: F1 Score

# Convertir les résultats en DataFrame
params_results_df_decision_tree = pd.DataFrame(results)
params_results_df_decision_tree.head(50)

K-Fold Cross Validation with K=15
Metric: Accuracy
Metric: Precision
Metric: Recall
Metric: F1 Score
```

## Convertir les résultats en Dataframe .

```
] # Convertir les résultats en DataFrame
params_results_df_decision_tree = pd.DataFrame(results)
params_results_df_decision_tree.head(50)
```

|    | Metric   | max_depth | min_samples_split | min_samples_leaf | Mean Score | Std Score |
|----|----------|-----------|-------------------|------------------|------------|-----------|
| 0  | Accuracy | 5         | 2                 | 1                | 0.735340   | 0.002429  |
| 1  | Accuracy | 5         | 2                 | 2                | 0.735340   | 0.002429  |
| 2  | Accuracy | 5         | 2                 | 4                | 0.735350   | 0.002435  |
| 3  | Accuracy | 5         | 3                 | 1                | 0.735340   | 0.002429  |
| 4  | Accuracy | 5         | 3                 | 2                | 0.735340   | 0.002429  |
| 5  | Accuracy | 5         | 3                 | 4                | 0.735350   | 0.002435  |
| 6  | Accuracy | 5         | 5                 | 1                | 0.735340   | 0.002429  |
| 7  | Accuracy | 5         | 5                 | 2                | 0.735340   | 0.002429  |
| 8  | Accuracy | 5         | 5                 | 4                | 0.735350   | 0.002435  |
| 9  | Accuracy | 7         | 2                 | 1                | 0.737410   | 0.003357  |
| 10 | Accuracy | 7         | 2                 | 2                | 0.737425   | 0.003326  |
| 11 | Accuracy | 7         | 2                 | 4                | 0.737520   | 0.003350  |
| 12 | Accuracy | 7         | 3                 | 1                | 0.737400   | 0.003347  |
| 13 | Accuracy | 7         | 3                 | 2                | 0.737420   | 0.003324  |
| 14 | Accuracy | 7         | 3                 | 4                | 0.737515   | 0.003348  |
| 15 | Accuracy | 7         | 5                 | 1                | 0.737405   | 0.003350  |
| 16 | Accuracy | 7         | 5                 | 2                | 0.737425   | 0.003322  |
| 17 | Accuracy | 7         | 5                 | 4                | 0.737520   | 0.003350  |
| 18 | Accuracy | 10        | 2                 | 1                | 0.735740   | 0.003337  |
| 19 | Accuracy | 10        | 2                 | 2                | 0.735850   | 0.003311  |

On utilise une approche de validation croisée à k-fold avec k=15 pour évaluer les performances d'un modèle d'arbre de décision. Il teste différentes combinaisons de paramètres pour l'arbre de décision, notamment la profondeur maximale de l'arbre (max\_depth), le nombre minimal d'échantillons requis pour diviser un nœud interne (min\_samples\_split), et le nombre minimal d'échantillons requis pour être une feuille (min\_samples\_leaf).

Pour chaque combinaison de paramètres, le code calcule les scores moyens et les écarts-types des scores obtenus pour chaque métrique d'évaluation, notamment l'exactitude (Accuracy), la précision (Precision), le rappel (Recall), et le score F1 (F1 Score).

Par exemple :

Les scores moyens d'exactitude (Accuracy) oscillent autour de 0,735 à 0,737 pour différentes valeurs de profondeur maximale, de nombre minimal d'échantillons pour diviser un nœud interne et de nombre minimal d'échantillons pour être une feuille. Cela signifie que le modèle d'arbre de décision parvient à bien généraliser sur les données de test avec ces paramètres. Les scores moyens de précision (Precision) sont généralement plus élevés, variant autour de 0,794 pour différentes combinaisons de paramètres. Cela suggère que le modèle a une capacité élevée à prédire correctement la catégorie de véhicule, en minimisant le nombre de faux positifs.

### Optimisation des performances du modèle d'arbre de décision et sélection de la meilleure combinaison d'hyperparamètre :

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Diviser l'ensemble de données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Définir les paramètres à tester
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Créer le modèle d'arbre de décision
model = DecisionTreeClassifier()

# Recherche sur grille pour trouver les meilleurs paramètres
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=15, scoring='accuracy', verbose=1, n_jobs=-1)
grid_search.fit(X_train, y_train)

# Afficher les meilleurs paramètres
print("Meilleurs paramètres:", grid_search.best_params_)

# Utiliser le meilleur modèle pour faire des prédictions sur l'ensemble de test
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Calculer les métriques
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
```

```
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
```

```
# Afficher les résultats
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Fitting 15 folds for each of 72 candidates, totalling 1080 fits
Meilleurs paramètres: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 5}
Accuracy: 0.7384530745955138
Precision: 0.7447775137997684
Recall: 0.7384530745955138
F1 Score: 0.7354613273957067
```

On construit et on optimise les hyperparamètres du modèle d'arbre de décision. Voici une interprétation et l'intérêt de chaque étape :

Division des données : Les données sont divisées en ensembles d'entraînement et de test. L'ensemble d'entraînement sera utilisé pour entraîner le modèle, tandis que

l'ensemble de test sera utilisé pour évaluer ses performances sur des données non vues auparavant. Cela permet de vérifier si le modèle généralise bien sur de nouvelles données.

Définition des paramètres à tester : Une grille de recherche est définie pour les hyperparamètres du modèle d'arbre de décision. Les hyperparamètres comprennent le critère de division des nœuds (gini ou entropy), la profondeur maximale de l'arbre, le nombre minimal d'échantillons requis pour diviser un nœud (min\_samples\_split), et le nombre minimal d'échantillons requis dans une feuille terminale (min\_samples\_leaf).

Recherche sur grille : Une recherche exhaustive des combinaisons d'hyperparamètres est effectuée sur une grille définie. Pour chaque combinaison, le modèle d'arbre de décision est entraîné sur l'ensemble d'entraînement à l'aide de la validation croisée (15 folds dans ce cas) et évalué sur la base de la métrique d'exactitude (accuracy). Cela permet de trouver les meilleurs hyperparamètres qui optimisent les performances du modèle.

Évaluation du modèle : Le modèle avec les meilleurs hyperparamètres est sélectionné, et ses performances sont évaluées sur l'ensemble de test à l'aide de différentes métriques telles que l'exactitude (accuracy), la précision (precision), le rappel (recall) et le score F1. Ces métriques fournissent une indication de la capacité du modèle à prédire correctement la catégorie de véhicules.

Dans l'ensemble, ces résultats suggèrent que le modèle d'arbre de décision avec les paramètres spécifiés a une performance raisonnable sur les données de test, avec une exactitude et un score F1 d'environ 73-74 %.

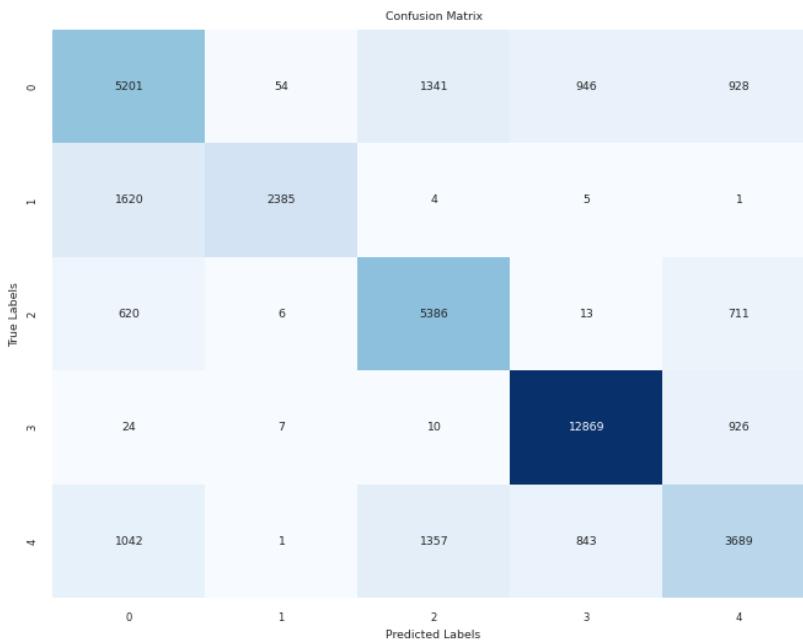
## Examinons la matrice de confusion pour évaluer les performances du modèle d'arbre de décision :

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

```
import matplotlib.pyplot as plt

# Calculer la matrice de confusion
conf_matrix = confusion_matrix(y_test, y_pred)

# Afficher la matrice de confusion sous forme de heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

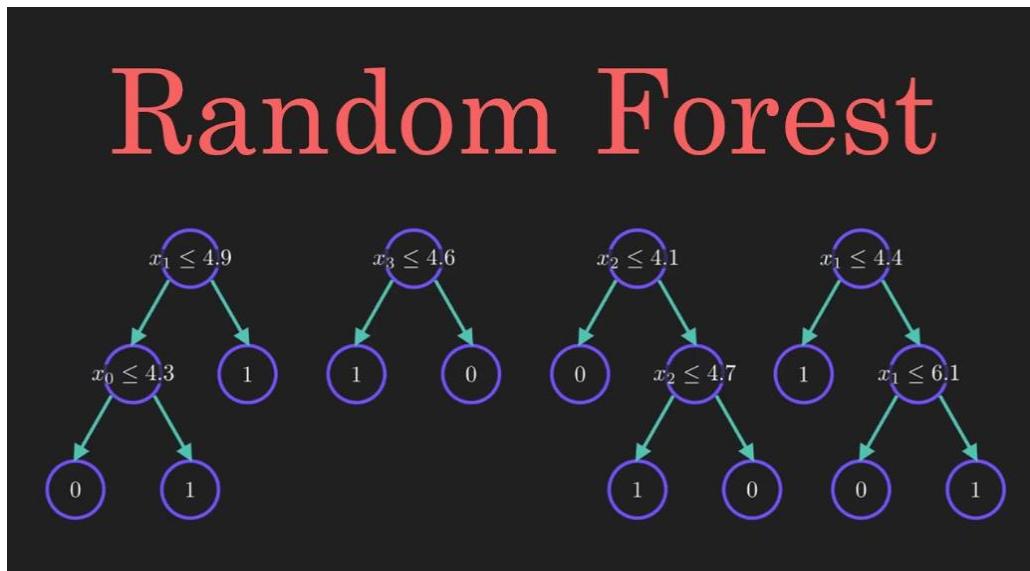


Dans ce cas on peut dire les valeurs les plus élevées sur la diagonale indiquent que le modèle a correctement classé un nombre important d'observations dans ces classes spécifiques.

- Pour la première classe : 5201 observations ont été correctement prédites contre 3288 incorrects.
- Pour la deuxième classe : 2385 observations ont été correctement prédites contre 68 incorrects.
- Pour la troisième classe : 5386 observations ont été correctement prédites contre 2712 incorrects.
- Pour la quatrième classe : 12869 observations ont été correctement prédites contre 1871 incorrects.
- Pour la cinquième classe : 3689 observations ont été correctement prédites contre 2566 incorrects.

Ces valeurs élevées sur la diagonale indiquent que le modèle a généralement bien performé pour classifier les données dans ces classes.

Evaluation des performances des performances d'un modèle RandomForest Classifier.



```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import make_scorer, accuracy_score, precision_score, recall_score, f1_score

# Définir les métriques
metrics = {
    "Accuracy": make_scorer(accuracy_score),
    "Precision": make_scorer(precision_score, average="weighted"),
    "Recall": make_scorer(recall_score, average="weighted"),
    "F1 Score": make_scorer(f1_score, average="weighted")
}

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Définir les paramètres à tester
params = {
    "n_estimators": [100, 200],
    "max_depth": [5, 10]
}

# Stocker les résultats dans un DataFrame
results = []

# Entrainement du modèle en fonction de plusieurs paramètre et calculer les métriques correspondants
for metric_name, scorer in metrics.items():
    print(f"Metric: {metric_name}")
    for n_estimators in params["n_estimators"]:
        for max_depth in params["max_depth"]:
            clf = RandomForestClassifier(n_estimators=n_estimators, max_depth=max_depth)
            clf.fit(X_train, y_train) # Ajuster le modèle aux données d'entraînement
            y_pred = clf.predict(X_test) # Faire des prédictions sur les données de test

            # Calculer la métrique spécifiée
            score = scorer(clf, X_test, y_test)
```

```
results.append({  
    "Metric": metric_name,  
    "n_estimators": n_estimators,  
    "max_depth": max_depth,  
    "Score": score  
})  
  
# Convertir les résultats en DataFrame  
results_df_random_forest = pd.DataFrame(results)  
  
Metric: Accuracy  
Metric: Precision  
Metric: Recall  
Metric: F1 Score
```

Nous testons l'algorithme de forêt aléatoire (RandomForestClassifier) pour classer les catégories de voitures. Grâce à cet algorithme, il n'est plus nécessaire d'utiliser la division des données d'entraînement avec KFold. En effet, le Random Forest utilise le principe du Bagging, une technique qui consiste à construire plusieurs arbres de décision indépendants à partir de sous-ensembles aléatoires des données d'entraînement. Chaque arbre est entraîné sur un échantillon différent des données, et ensuite les résultats sont agrégés pour produire une prédiction finale. Cela permet de réduire le surajustement et d'améliorer la performance du modèle en moyenne. Voici ce que chaque partie du code fait :

Définition des métriques : Nous définissons les métriques de performance que nous voulons évaluer, à savoir l'exactitude, la précision, le rappel et le score F1. Nous utilisons la fonction make\_scoring de Scikit-learn pour convertir ces métriques en fonctions de score que nous pouvons passer à la validation croisée.

Division des données : Les données sont divisées en ensembles d'entraînement et de test à l'aide de la fonction train\_test\_split. Cela nous permet d'évaluer les performances du modèle sur des données non vues.

Définition des paramètres à tester : Nous définissons les paramètres de l'algorithme que nous voulons optimiser, dans ce cas le nombre d'estimateurs (n\_estimators) et la profondeur maximale de chaque arbre dans la forêt (max\_depth).

Validation croisée et évaluation : Pour chaque métrique définie, et pour chaque combinaison de valeurs de n\_estimators et max\_depth, nous entraînons un modèle de forêt aléatoire sur les données d'entraînement et évaluons ses performances sur les données de test se basant sur les métriques spécifiée.

Stockage des résultats : Les résultats sont stockés dans un DataFrame.

```
results_df_random_forest
```

| Metric      | n_estimators | max_depth | Score    |
|-------------|--------------|-----------|----------|
| 0 Accuracy  | 100          | 5         | 0.735252 |
| 1 Accuracy  | 100          | 10        | 0.738328 |
| 2 Accuracy  | 200          | 5         | 0.736127 |
| 3 Accuracy  | 200          | 10        | 0.738578 |
| 4 Precision | 100          | 5         | 0.736347 |
| 5 Precision | 100          | 10        | 0.783360 |
| 6 Precision | 200          | 5         | 0.741589 |
| 7 Precision | 200          | 10        | 0.759655 |
| 8 Recall    | 100          | 5         | 0.729025 |
| 9 Recall    | 100          | 10        | 0.738728 |
| 10 Recall   | 200          | 5         | 0.735777 |
| 11 Recall   | 200          | 10        | 0.737828 |
| 12 F1 Score | 100          | 5         | 0.732042 |
| 13 F1 Score | 100          | 10        | 0.742562 |
| 14 F1 Score | 200          | 5         | 0.730130 |
| 15 F1 Score | 200          | 10        | 0.741167 |

Selon les résultats du model, nous concluons que la meilleure combinaison d'hyperparamètres est lorsque n\_estimators=200 et max\_depth=10, ce qui donne les mesures suivantes : l'exactitude est de 0.739, la précision est de 0.760, le rappel est de 0.738, et le score F1 est de 0.741.

### Optimisation des performances du modèle de forêt aléatoire et sélection de la meilleure combinaison d'hyperparamètre :

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import accuracy_score

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Définir les paramètres à tester
param_grid = {
    "n_estimators": [50, 100, 200],
    "max_depth": [5, 10]
}
# Créer le modèle RandomForest
rf_model = RandomForestClassifier(random_state=42)

# Créer l'objet GridSearchCV
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=5, scoring='accuracy')

# Effectuer la recherche sur grille sur les données d'entraînement
grid_search.fit(X_train, y_train)

# Obtenir les meilleurs paramètres et le meilleur score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

# Créer un nouveau modèle avec les meilleurs paramètres trouvés
best_rf_model = RandomForestClassifier(**best_params)

# Entrainer le meilleur modèle sur toutes les données d'entraînement
best_rf_model.fit(X_train, y_train)
```

```
# Faire des prédictions sur l'ensemble de test
y_pred = best_rf_model.predict(X_test)

# Calculer les métriques
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Afficher les résultats
print("Best Parameters:", best_params)
print("Best Score:", best_score)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Best Parameters: {'max_depth': 10, 'n_estimators': 50}
Best Score: 0.7353489151943708
Accuracy: 0.7383530470879491
Precision: 0.7810679007645149
Recall: 0.7383530470879491
F1 Score: 0.7426783082099092
```

Interprétation des résultats :

En ajoutant d'autre valeurs parmi la liste des paramètres, la meilleur combinaison trouvé sont max\_depth=10 et n\_estimators=50, avec un meilleur score d'accuracy de 0.735.

Le modèle entraîné avec ces paramètres a une accuracy de 0.738, une précision de 0.781, un rappel de 0.738 et un score F1 de 0.743 sur l'ensemble de test.

Ces résultats suggèrent que le modèle peut bien généraliser sur de nouvelles données avec une précision raisonnable.

**Examinons la matrice de confusion pour évaluer les performances du model de forêt aléatoire:**

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Calculer la matrice de confusion
conf_matrix = confusion_matrix(y_test, y_pred)

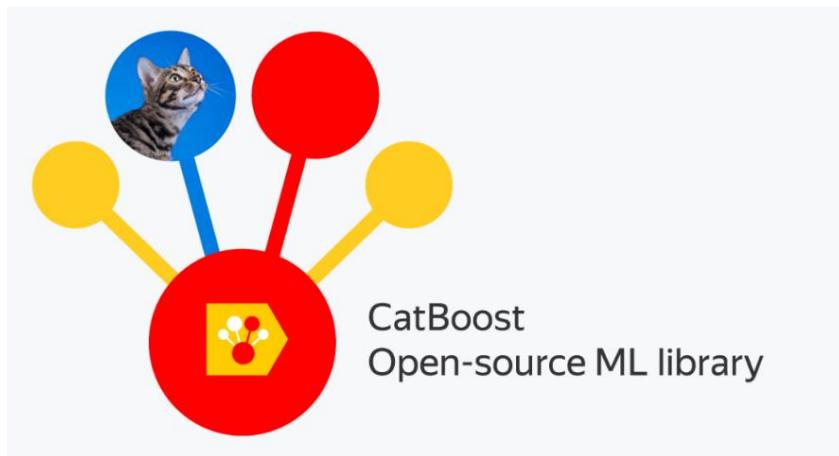
# Afficher la matrice de confusion sous forme de heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



Les valeurs les plus élevées sur la diagonale de la matrice de confusion indiquent que le modèle a correctement classé un nombre important d'observations dans ces classes spécifiques.

- Pour la première classe, 4839 observations ont été correctement prédites tandis que 2563 ont été incorrectement classées.
- Pour la deuxième classe, 2375 observations ont été correctement prédites tandis que seulement 19 ont été incorrectes.
- Pour la troisième classe, 4124 observations ont été correctement prédites tandis que 2712 ont été incorrectes.
- Pour la quatrième classe, 12876 observations ont été correctement prédites tandis que 333 ont été incorrectes.
- Pour la cinquième classe, 5312 observations ont été correctement prédites tandis que 5572 ont été incorrectes.

Ces valeurs élevées sur la diagonale indiquent que le modèle a généralement bien performé pour classifier les données dans ces classes, sauf pour la classe 4 où le nombre d'observations incorrectes est supérieur à celui des correctes.

*Evaluation des performances des performances d'un modèle CatBoost classifier*

**CatBoost** est une bibliothèque de gradient boosting open-source développée par Yandex. Elle est conçue pour les problèmes de classification et de régression et est particulièrement efficace pour les ensembles de données de grande taille. CatBoost utilise une méthode d'entraînement basée sur le gradient boosting, qui consiste à entraîner une série de modèles de faible complexité (appelés "weak learners") de manière séquentielle, en mettant l'accent sur les exemples qui ont été mal classés par les modèles précédents. Cette approche améliore progressivement les performances du modèle en corrigeant les erreurs et en renforçant les prédictions précises.

```
!pip install catboost
```

```
from catboost import CatBoostClassifier
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Définir les paramètres à tester
param_grid = {
    "iterations": [50, 100, 200], # Nombre d'itérations
    "learning_rate": [0.05, 0.1, 0.5], # Taux d'apprentissage
    "depth": [4, 6, 8] # Profondeur de l'arbre
}

# Créer le modèle CatBoost
cb_model = CatBoostClassifier(random_state=42, verbose=0)

# Créer l'objet GridSearchCV
grid_search = GridSearchCV(estimator=cb_model, param_grid=param_grid, cv=5, scoring='accuracy')

# Effectuer la recherche sur grille sur les données d'entraînement
grid_search.fit(X_train, y_train)
```

```
# Obtenir les meilleurs paramètres et le meilleur score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

# Créer un nouveau modèle avec les meilleurs paramètres trouvés
best_cb_model = CatBoostClassifier(**best_params, random_state=42, verbose=0)

# Entraîner le meilleur modèle sur toutes les données d'entraînement
best_cb_model.fit(X_train, y_train)

# Faire des prédictions sur l'ensemble de test
y_pred = best_cb_model.predict(X_test)

# Calculer les métriques
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Afficher les résultats
print("Best Parameters:", best_params)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

Best Parameters: {'depth': 6, 'iterations': 200, 'learning\_rate': 0.1}

Accuracy: 0.7401035284703293

Precision: 0.7506868764368027

Recall: 0.7401035284703293

F1 Score: 0.7395758196456492

Voici ce que chaque partie du code accomplit :

**Importation des bibliothèques** : Les bibliothèques nécessaires sont importées, y compris CatBoostClassifier pour le modèle de classification, GridSearchCV pour la recherche sur grille, train\_test\_split pour diviser les données en ensembles d'entraînement et de test, et différentes mesures de performance telles que accuracy\_score, precision\_score, recall\_score, et f1\_score.

**Division des données** : Les données sont divisées en ensembles d'entraînement et de test en utilisant train\_test\_split. Cela permet d'évaluer les performances du modèle sur des données non vues.

**Définition des paramètres à tester** : Une grille de paramètres est définie pour la recherche sur grille. Les paramètres comprennent le nombre d'itérations, le taux d'apprentissage et la profondeur de l'arbre pour le modèle CatBoost.

**Création du modèle CatBoost** : Un modèle CatBoostClassifier est créé avec un état aléatoire fixé et un niveau de verbosité réglé sur 0 pour supprimer les messages d'entraînement.

Recherche sur grille : GridSearchCV est utilisé pour effectuer une recherche sur grille à l'aide du modèle CatBoost et de la grille de paramètres définie.

La validation croisée à 5 plis est utilisée pour évaluer les performances des modèles.

Entraînement du meilleur modèle : Une fois les meilleurs paramètres trouvés, un nouveau modèle CatBoost est créé avec ces paramètres optimaux et est entraîné sur toutes les données d'entraînement.

Prédictions et évaluation : Le modèle entraîné est utilisé pour faire des prédictions sur l'ensemble de test, et les métriques d'évaluation telles que l'accuracy, la précision, le rappel et le F1-score sont calculées et affichées.

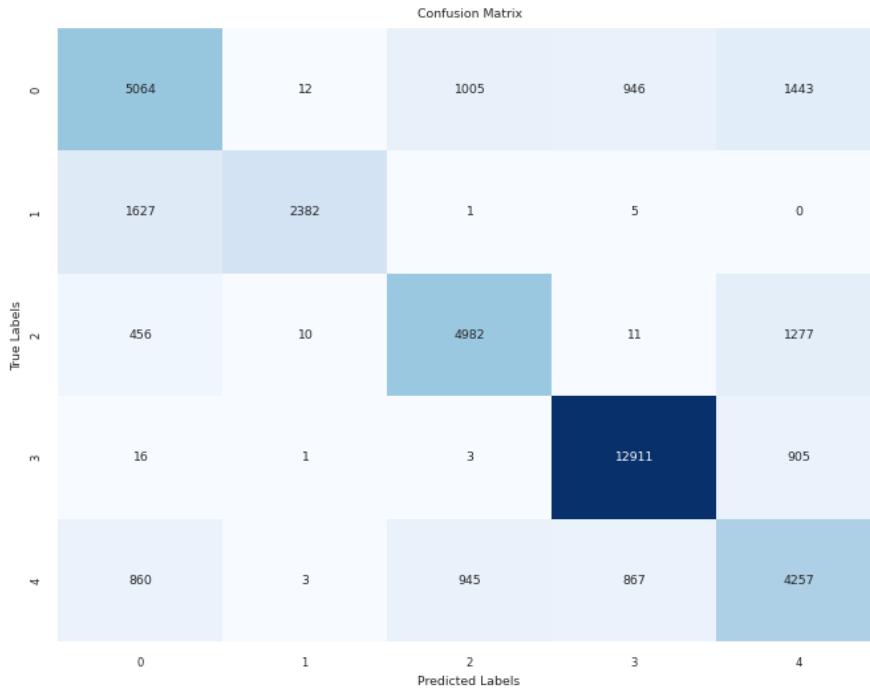
Le modèle CatBoost entraîné avec ces paramètres optimaux atteint une accuracy de 74.01%, une précision de 75 %, un rappel de 74.01%, et un F1-score de 73.96% sur l'ensemble de test. Ces performances indiquent une capacité satisfaisante du modèle à classifier correctement les données de test.

### **Examinons la matrice de confusion pour évaluer les performances du model de CatBoost:**

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Calculer la matrice de confusion
conf_matrix = confusion_matrix(y_test, y_pred)

# Afficher la matrice de confusion sous forme de heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

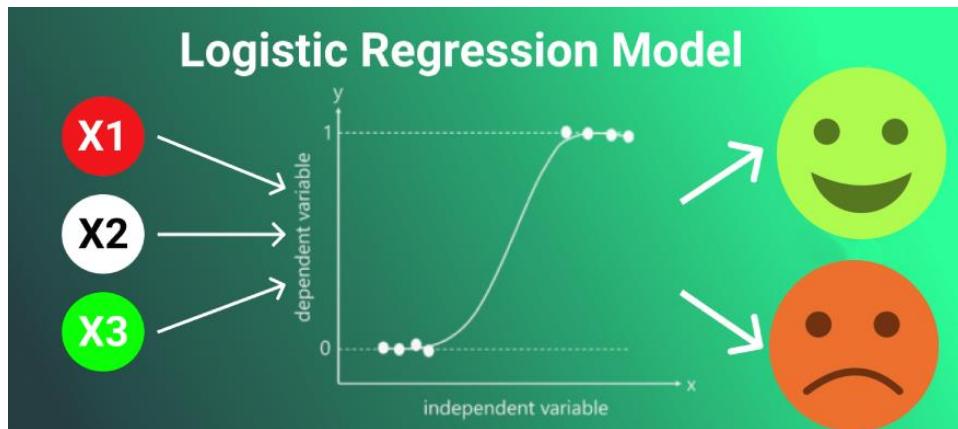


Les valeurs les plus élevées sur la diagonale de la matrice de confusion indiquent que le modèle a correctement classé un nombre important d'observations dans ces classes spécifiques.

- Pour la première classe, 5064 observations ont été correctement prédites tandis que 2959 ont été incorrectement classées.
- Pour la deuxième classe, 2382 observations ont été correctement prédites tandis que seules 26 ont été incorrectes.
- Pour la troisième classe, 4982 observations ont été correctement prédites tandis que 2712 ont été incorrectes.
- Pour la quatrième classe, 12911 observations ont été correctement prédites tandis que 1954 ont été incorrectes.
- Pour la cinquième classe, 4257 observations ont été correctement prédites tandis que 3615 ont été incorrectes.

Ces valeurs élevées sur la diagonale indiquent que le modèle a généralement bien performé dans la classification des données dans ces classes.

*Evaluation des performances des performances d'un modèle LogisticRegression*



Bien que la faible linéarité entre les caractéristiques ait été observée à travers la matrice de nuages de points tracée précédemment, et que nous ayons conclu qu'une régression linéaire ne serait pas efficace, il serait tout de même intéressant d'utiliser la régression logistique multinomiale pour confirmer cette supposition et explorer différentes approches.

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score

# Diviser l'ensemble de données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Définir les paramètres à tester
param_grid = {
    'solver': ['newton-cg', 'lbfgs', 'sag', 'saga'],
    'penalty': ['l2'],
    'max_iter': [100, 200, 300]
}

# Créer le modèle de régression logistique multinomiale
model = LogisticRegression(multi_class='multinomial')

# Recherche sur grille pour trouver les meilleurs paramètres
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=15, scoring='accuracy', verbose=1,
                           n_jobs=-1)
grid_search.fit(X_train, y_train)

# Afficher les meilleurs paramètres
print("Meilleurs paramètres:", grid_search.best_params_)

# Utiliser le meilleur modèle pour faire des prédictions sur l'ensemble de test
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
```

```
# Calculer les métriques
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Afficher les résultats
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

Fitting 15 folds for each of 12 candidates, totalling 180 fits

Meilleurs paramètres: {'max\_iter': 100, 'penalty': 'l2', 'solver': 'saga'}

Accuracy: 0.672484933356673

Precision: 0.6564252875702897

Recall: 0.672484933356673

F1 Score: 0.6619995571142966

Les résultats obtenus pour le modèle de régression logistique multinomiale semblent être parmi les plus bas jusqu'à présent. Les mesures de performance telles que l'accuracy (67.24%), la précision, le rappel (recall) et le score F1 sont plutôt faibles par rapport aux autres modèles déjà testés. Cela pourrait indiquer que ce modèle ne parvient pas à capturer la structure sous-jacente des données aussi efficacement que les autres modèles.

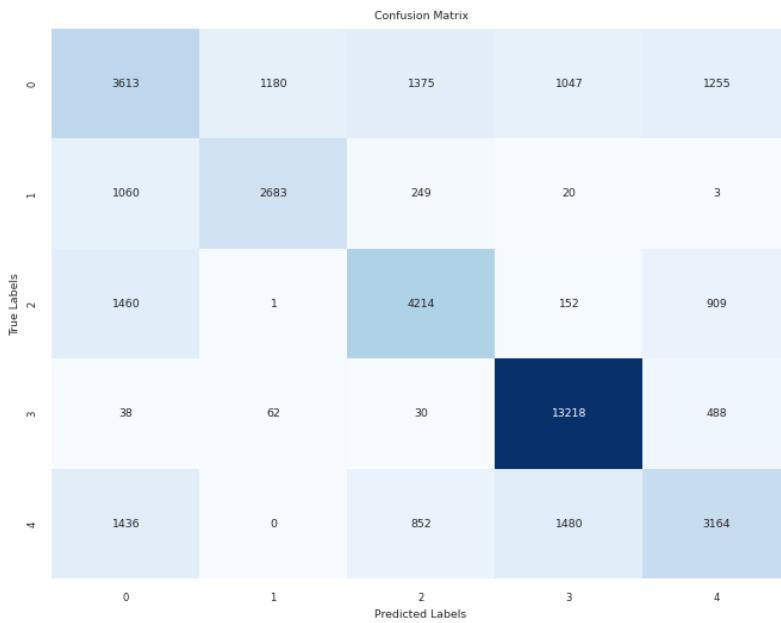
## Examinons la matrice de confusion pour évaluer les performances du model de Logistic Regression :

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Calculer la matrice de confusion
conf_matrix = confusion_matrix(y_test, y_pred)

# Afficher la matrice de confusion sous forme de heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
```

```
plt.show()
```



Selon la matrice de confusion, les résultats sont les suivants :

- Pour la première classe, 3613 observations ont été correctement prédites tandis que 4857 ont été incorrectement classées.
- Pour la deuxième classe, 2683 observations ont été correctement prédites tandis que 1332 ont été incorrectes.
- Pour la troisième classe, 4214 observations ont été correctement prédites tandis que 2522 ont été incorrectes.
- Pour la quatrième classe, 13218 observations ont été correctement prédites tandis que seulement 648 ont été incorrectes.
- Pour la cinquième classe, 3164 observations ont été correctement prédites tandis que 3768 ont été incorrectes.

Ces résultats suggèrent que le modèle a réussi à classifier correctement la plupart des observations dans ces classes spécifiques, bien que la prédiction pour les classes 0 et 4 n'ait pas été très performante.

### Analyse comparative des performances en utilisant des réseaux de neurones (Deep Learning):

Une approche intéressante consiste à explorer l'utilisation des réseaux de neurones pour les tâches de classification et à déterminer si le deep learning peut fournir des résultats plus intéressants que les méthodes d'apprentissage supervisé du Machine Learning. Cependant, en raison des limitations de performances de la machine virtuelle actuelle, l'exécution de modèles de deep learning n'est

malheureusement pas possible. Une solution envisageable serait d'exporter la dataframe encodée "numerical\_data\_encoded" en un fichier Excel nommé "results.xls" contenant les caractéristiques des clients et la variable cible "Cluster" vers les ressources de la machine locale, où l'algorithme de deep learning pourrait être exécuté, vous pouvez accéder à ce fichier dans le lien google drive suivant :

<https://docs.google.com/spreadsheets/d/1pVCnSrgTkCT8bpoKIK1HKfSG5MlypvUi/edit?usp=sharing&ouid=111367534350887208181&rtpof=true&sd=true>

```
# Enregistrer les résultats dans un nouveau fichier Excel
numerical_data_encoded.to_excel("results.xls", index=False, encoding='utf-8')
```

Vous pouvez accéder au code soit en ouvrant le fichier "deep\_learning.py" dans le dossier "Data Mining" du rendu du projet PFA, soit en cliquant sur le lien suivant :  
[https://drive.google.com/file/d/1aSZwdZjoi\\_j1BaXqjCT8qEArBaLrU3ZV/view?usp=sharing](https://drive.google.com/file/d/1aSZwdZjoi_j1BaXqjCT8qEArBaLrU3ZV/view?usp=sharing)

```
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split

# Charger les données à partir du fichier Excel
numerical_data_encoded = pd.read_excel("results.xls")

# Diviser les données en variables explicatives et cible
X = numerical_data_encoded.drop(columns=["Cluster"])
y = numerical_data_encoded["Cluster"]

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(*arrays: X, y, test_size=0.2, random_state=42)

# Créer le modèle
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    Dense(32, activation='relu'),
    Dropout(0.5),
    Dense(max(y)+1, activation='softmax') # max(y)+1 pour le nombre de clusters
])

# Compiler le modèle
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
# Compiler le modèle
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Entrainer le modèle
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.1)

# Évaluer le modèle sur l'ensemble de test
test_loss, test_accuracy = model.evaluate(X_test, y_test)

print("Test Accuracy:", test_accuracy)
```

Ce script utilise TensorFlow et Keras pour créer et entraîner un modèle de réseau de neurones artificiels à des fins de classification. Il utilise un réseau de neurones artificiels avec deux couches cachées et des couches de dropout pour la régularisation afin de classer les données en fonction des étiquettes de classe fournies dans la colonne "Cluster". Voici une description de chaque étape :

- 1- Importation des bibliothèques nécessaires : Les bibliothèques pandas, TensorFlow et scikit-learn sont importées. Pandas est utilisé pour manipuler les données, TensorFlow pour construire le modèle de réseau de neurones, et scikit-learn pour diviser les données en ensembles d'entraînement et de test.
- 2- Chargement des données : Les données sont chargées à partir d'un fichier Excel nommé "results.xls" à l'aide de la fonction `read_excel()` de pandas.
- 3- Division des données : Les données sont divisées en variables explicatives (X) et la variable cible (y). Les variables explicatives sont toutes les colonnes sauf la colonne "Cluster", qui contient les étiquettes de classe.
- 4- Division des données en ensembles d'entraînement et de test : Les données sont divisées en ensembles d'entraînement et de test à l'aide de la fonction `train_test_split()` de scikit-learn. Dans cet exemple, 80% des données sont utilisées pour l'entraînement et 20% pour les tests.
- 5- Construction du modèle : Un modèle séquentiel est créé à l'aide de l'API Sequential de Keras. Le modèle commence par une couche Dense de 64 neurones avec une fonction d'activation ReLU, suivie d'une couche Dropout pour la régularisation et éviter la saturation des neurones. Ensuite, il y a une autre couche Dense de 32 neurones avec ReLU et une autre couche Dropout. Enfin, la couche de sortie utilise une fonction d'activation softmax pour produire des probabilités pour chaque classe.
- 6- Compilation du modèle : Le modèle est compilé avec l'optimiseur "adam", la fonction de perte "sparse\_categorical\_crossentropy" et la métrique "accuracy".
- 7- Entraînement du modèle : Le modèle est entraîné sur les données d'entraînement pour 50 époques avec une taille de lot de 32. La validation est effectuée sur 10% des données d'entraînement.
- 8- Évaluation du modèle : Une fois l'entraînement terminé, le modèle est évalué sur l'ensemble de test pour obtenir la perte de test et la précision de test.

```
14/5/14/5 [=====] - 2s 1ms/step - loss: 1.5516 - accuracy: 0.3478 - val_loss: 1.5271 - val_accuracy: 0.3488
Epoch 44/50
1475/1475 [=====] - 2s 1ms/step - loss: 1.5316 - accuracy: 0.3478 - val_loss: 1.5272 - val_accuracy: 0.3488
Epoch 45/50
1475/1475 [=====] - 2s 1ms/step - loss: 1.5316 - accuracy: 0.3478 - val_loss: 1.5272 - val_accuracy: 0.3488
Epoch 46/50
1475/1475 [=====] - 2s 1ms/step - loss: 1.5316 - accuracy: 0.3478 - val_loss: 1.5271 - val_accuracy: 0.3488
Epoch 47/50
1475/1475 [=====] - 2s 1ms/step - loss: 1.5316 - accuracy: 0.3478 - val_loss: 1.5272 - val_accuracy: 0.3488
Epoch 48/50
1475/1475 [=====] - 2s 1ms/step - loss: 1.5316 - accuracy: 0.3478 - val_loss: 1.5271 - val_accuracy: 0.3488
Epoch 49/50
1475/1475 [=====] - 2s 1ms/step - loss: 1.5316 - accuracy: 0.3478 - val_loss: 1.5272 - val_accuracy: 0.3488
Epoch 50/50
1475/1475 [=====] - 2s 1ms/step - loss: 1.5316 - accuracy: 0.3478 - val_loss: 1.5271 - val_accuracy: 0.3488
410/410 [=====] - 0s 916us/step - loss: 1.5355 - accuracy: 0.3441
Test Accuracy: 0.34409093856811523
```

Le résultat obtenu dévoile une accuracy de 34,4% pour les données de test, ce qui suggère que l'utilisation des réseaux de neurones ne semble pas être une approche viable pour prédire les catégories de voitures en fonction des données des clients et donc elle est à exclure.

*Evaluation des performances de l'approche d'empilement (stacking) pour la classification utilisant plusieurs modèles de base:*



Le stacking, également appelé empilement, est une technique d'ensemble learning où les prédictions de plusieurs modèles de base sont utilisées comme données d'entrée pour un métamodèle. Plutôt que de se contenter de combiner les prédictions de différents modèles de manière pondérée ou de les voter, le stacking implique d'utiliser les prédictions des modèles de base comme caractéristiques pour former un modèle de niveau supérieur, généralement un modèle plus sophistiqué comme un arbre de décision, une régression linéaire ou un réseau de neurones. Ce métamodèle apprend ensuite à pondérer les prédictions des modèles de base pour produire une prédiction finale. Le stacking est utilisé pour améliorer les performances prédictives en exploitant les forces de différents types de modèles et en réduisant les faiblesses de chacun.

```
from sklearn.model_selection import train_test_split
from catboost import CatBoostClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import StackingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Define the base models
base_models = [
    ('rf', RandomForestClassifier(n_estimators=50, max_depth=10, random_state=42)),
    ('dt',
        DecisionTreeClassifier(random_state=42, criterion='entropy', max_depth=10, min_samples_leaf=2, min_samples_split=10)),
```

```
('cb', CatBoostClassifier(random_state=42, depth=6,iterations=200,learning_rate=0.05,verbose=0)),
('knn', KNeighborsClassifier(metric='manhattan',n_neighbors=40,weights='uniform'))
]

# Initialize the StackingClassifier
stacked_model = StackingClassifier(
    estimators=base_models,
    final_estimator=LogisticRegression(max_iter=1000, random_state=42)
)

# Train the StackingClassifier on the training data
stacked_model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = stacked_model.predict(X_test)

# Evaluate the performance of the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

Accuracy: 0.7390782465177924

Precision: 0.7524301150570941

Recall: 0.7390782465177924

F1 Score: 0.7390404486468256

Différents modèles de base ont été choisis en fonction de leur précision, sélectionnant ceux dont l'accuracy était d'au moins 70%. On a sélectionné les modèles avec leurs hyperparamètres optimisés pour maximiser ses performances respectives. Les modèles de base inclus sont : Random Forest (RandomForestClassifier), Decision Tree (DecisionTreeClassifier), CatBoost (CatBoostClassifier), K-Nearest Neighbors (KNeighborsClassifier)

Ensuite, un StackingClassifier est initialisé en spécifiant les modèles de base et le modèle final qui agrège les prédictions des modèles de base. Dans ce cas, le modèle final est une régression logistique (LogisticRegression). Le modèle d'empilement est entraîné sur les données d'entraînement. Pendant cette phase, les prédictions des modèles de base sont utilisées comme caractéristiques pour le modèle final. Une fois le modèle d'empilement entraîné, il est utilisé pour faire des prédictions sur les données de test. Enfin, les performances du modèle empilé sont

évaluées en calculant des métriques telles que l'accuracy, la précision, le rappel (recall) et le score F1 à l'aide des prédictions faites sur les données de test.

Le modèle de l'ensemble empilé semble être le plus performant à ce stade, avec une accuracy, un rappel et un score F1 de 0.739, ce qui représente le maximum obtenu jusqu'à présent au niveau de l'accuracy. La précision est également d'environ 0.75.

### Interprétation :

Accuracy : la proportion de prédictions correctes parmi toutes les prédictions effectuées par le modèle. Un score d'accuracy d'environ 0.74 signifie que le modèle prédit correctement la classe cible pour environ 74% des exemples dans l'ensemble de test.

Precision : Un score de précision d'environ 0.75 indique que lorsque le modèle prédit une classe, il a raison environ 75% du temps.

Recall : Avec un score de rappel d'environ 0.74, le modèle capture correctement environ 74% de toutes les observations positives.

F1 Score : Il représente l'harmonique moyenne de la précision et du rappel. Un score F1 d'environ 0.74 indique un bon équilibre entre la précision et le rappel.

### Résultat Final :

L'approche consistante à expérimenter plusieurs modèles avant de choisir le modèle final a été fructueuse. En testant une variété de modèles, nous avons pu évaluer leurs performances respectives et sélectionner celui qui offre les meilleures performances sur l'ensemble de test. Dans ce cas, l'ensemble empilé semble être le choix optimal pour appliquer la prédiction des catégories de voitures sur les données marketing.

## 8.5 Application du modèle de prédiction aux données Marketing

### - Récupération des données de la table Hive "table\_marketing":

```
# Exécution d'une requête SQL
cursor.execute('SELECT * FROM table_marketing')
# Récupération des noms de colonnes
columns = [desc[0] for desc in cursor.description]
# Récupération des résultats
results = cursor.fetchall()
# Création d'un DataFrame Pandas avec les résultats et les noms de colonnes
marketing = pd.DataFrame(results, columns=columns)
marketing
```

|    | table_marketing.age | table_marketing.sexé | table_marketing.taux | table_marketing.situationfamiliale | table_marketing.nbenfantsacharge | table_marketing.deuxieme_voiture |
|----|---------------------|----------------------|----------------------|------------------------------------|----------------------------------|----------------------------------|
| 0  | 21                  | F                    | 1396                 | Célibataire                        | 0                                | false                            |
| 1  | 35                  | M                    | 223                  | Célibataire                        | 0                                | false                            |
| 2  | 48                  | M                    | 401                  | Célibataire                        | 0                                | false                            |
| 3  | 26                  | F                    | 420                  | En Couple                          | 3                                | true                             |
| 4  | 80                  | M                    | 530                  | En Couple                          | 3                                | false                            |
| 5  | 27                  | F                    | 153                  | En Couple                          | 2                                | false                            |
| 6  | 59                  | F                    | 572                  | En Couple                          | 2                                | false                            |
| 7  | 43                  | F                    | 431                  | Célibataire                        | 0                                | false                            |
| 8  | 64                  | M                    | 559                  | Célibataire                        | 0                                | false                            |
| 9  | 22                  | M                    | 154                  | En Couple                          | 1                                | false                            |
| 10 | 79                  | F                    | 981                  | En Couple                          | 2                                | false                            |
| 11 | 55                  | M                    | 588                  | Célibataire                        | 0                                | false                            |
| 12 | 19                  | F                    | 212                  | Célibataire                        | 0                                | false                            |
| 13 | 34                  | F                    | 1112                 | En Couple                          | 0                                | false                            |
| 14 | 60                  | M                    | 524                  | En Couple                          | 0                                | true                             |
| 15 | 22                  | M                    | 411                  | En Couple                          | 3                                | true                             |
| 16 | 58                  | M                    | 1192                 | En Couple                          | 0                                | false                            |
| 17 | 54                  | F                    | 452                  | En Couple                          | 3                                | true                             |
| 18 | 35                  | M                    | 589                  | Célibataire                        | 0                                | false                            |

- Prétraitement des données récupérées de la même manière que pour les features du model de classification :

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

marketing_encoded = marketing.copy()

# Créer un dictionnaire pour mapper les valeurs
mapping = {'Célibataire': 0, 'En Couple': 1}

# Appliquer le mapping à la colonne 'model_immatriculations_clients.situationfamiliale'
marketing_encoded['table_marketing.situationfamiliale'] =
marketing_encoded['table_marketing.situationfamiliale'].map(mapping)

# Créer un dictionnaire pour mapper les valeurs
mapping = {'M': 0, 'F': 1}

# Appliquer le mapping à la colonne 'model_immatriculations_clients.situationfamiliale'
marketing_encoded['table_marketing.sexé'] = marketing_encoded['table_marketing.sexé'].map(mapping)

# Créer un dictionnaire pour mapper les valeurs
mapping = {'false': 0, 'true': 1}

# Appliquer le mapping à la colonne 'model_immatriculations_clients.situationfamiliale'
marketing_encoded['table_marketing.deuxieme_voiture'] =
marketing_encoded['table_marketing.deuxieme_voiture'].map(mapping)

# Normaliser les données
scaler = StandardScaler()
marketing_scaled = scaler.fit_transform(marketing_encoded)
```

Encodage des variables catégorielles : Les variables catégorielles telles que la situation familiale, le sexe et la possession d'une deuxième voiture ont été encodées en valeurs numériques. Cela est nécessaire car de nombreux algorithmes d'apprentissage automatique ne peuvent pas traiter directement les variables catégorielles et nécessitent des valeurs numériques en entrée.

Normalisation des données : Les données ont été normalisées à l'aide du StandardScaler de scikit-learn. La normalisation est une étape importante pour mettre toutes les caractéristiques à la même échelle, ce qui peut améliorer la convergence des algorithmes d'apprentissage automatique et leur performance globale.

- **Utilisation du modèle empilé précédemment construit (stacked\_model) pour les prédictions sur les données prétraitées marketing (marketing\_scaled):**

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Faire des prédictions sur les données du fichier marketing
marketing_pred = stacked_model.predict(marketing_scaled)

# Ajouter les prédictions au dataframe du fichier marketing
marketing['Catégorie'] = marketing_pred

# Création du dictionnaire pour mapper les numéros de cluster aux types de véhicules
cluster_type_mapping = {
    0: 'Familiale',
    1: 'Prestige',
    2: 'Sportive',
    3: 'Économique',
    4: 'Citadine'
}

# Ajout de la colonne 'Type' à partir du mapping des clusters
marketing['Catégorie'] = marketing['Catégorie'].map(cluster_type_mapping)

# Affichage des types de véhicules prédits pour chaque ligne
marketing
```

|    | table_marketing.age | table_marketing.sex | table_marketing.taux | table_marketing.situationfamiliale | table_marketing.nbenfantscharge | table_marketing.deuxieme_voiture | Catégorie        |
|----|---------------------|---------------------|----------------------|------------------------------------|---------------------------------|----------------------------------|------------------|
| 0  | 21                  | F                   | 1396                 | Célibataire                        | 0                               | 0                                | False Économique |
| 1  | 35                  | M                   | 223                  | Célibataire                        | 0                               | 0                                | False Économique |
| 2  | 48                  | M                   | 401                  | Célibataire                        | 0                               | 0                                | False Économique |
| 3  | 26                  | F                   | 420                  | En Couple                          | 3                               | 3                                | True Sportive    |
| 4  | 80                  | M                   | 530                  | En Couple                          | 3                               | 3                                | False Sportive   |
| 5  | 27                  | F                   | 153                  | En Couple                          | 2                               | 2                                | False Citadine   |
| 6  | 59                  | F                   | 572                  | En Couple                          | 2                               | 2                                | False Citadine   |
| 7  | 43                  | F                   | 431                  | Célibataire                        | 0                               | 0                                | False Économique |
| 8  | 64                  | M                   | 559                  | Célibataire                        | 0                               | 0                                | False Économique |
| 9  | 22                  | M                   | 154                  | En Couple                          | 1                               | 1                                | False Citadine   |
| 10 | 79                  | F                   | 981                  | En Couple                          | 2                               | 2                                | False Familiale  |
| 11 | 55                  | M                   | 588                  | Célibataire                        | 0                               | 0                                | False Économique |
| 12 | 19                  | F                   | 212                  | Célibataire                        | 0                               | 0                                | False Économique |
| 13 | 34                  | F                   | 1112                 | En Couple                          | 0                               | 0                                | False Familiale  |
| 14 | 60                  | M                   | 524                  | En Couple                          | 0                               | 0                                | True Économique  |
| 15 | 22                  | M                   | 411                  | En Couple                          | 3                               | 3                                | True Sportive    |
| 16 | 58                  | M                   | 1192                 | En Couple                          | 0                               | 0                                | False Familiale  |
| 17 | 54                  | F                   | 452                  | En Couple                          | 3                               | 3                                | True Sportive    |
| 18 | 35                  | M                   | 589                  | Célibataire                        | 0                               | 0                                | False Économique |
| 19 | 59                  | M                   | 748                  | En Couple                          | 0                               | 0                                | True Économique  |

```
# Enregistrer les résultats dans un nouveau fichier CSV
marketing.to_csv("marketing_resultat.csv", index=False, encoding='utf-8')
```

Les résultats du dataframe marketing, y compris les prédictions de catégorie de véhicules, sont enregistrés dans un nouveau fichier CSV appelé "marketing\_resultat.csv" qui sera chargés prochainement dans une base de données résultat (regardez les scripts accompagnés avec ce notebook). L'image ci-dessous illustre le résultat des prédictions des données marketing que vous pouvez consulter dans le rendu du rapport PFA ou via le lien Google Drive suivant :

[https://drive.google.com/file/d/1bD\\_p8JEL\\_4XMpOSEkK6ULraZ8p4bXlfF/view?usp=sharing](https://drive.google.com/file/d/1bD_p8JEL_4XMpOSEkK6ULraZ8p4bXlfF/view?usp=sharing)

| A                   | B                   | C                    | D                                  | E                               | F                                | G          |
|---------------------|---------------------|----------------------|------------------------------------|---------------------------------|----------------------------------|------------|
| table_marketing.age | table_marketing.sex | table_marketing.taux | table_marketing.situationfamiliale | table_marketing.nbenfantscharge | table_marketing.deuxieme_voiture | Catégorie  |
| 21 F                |                     | 1396                 | Célibataire                        | 0                               | 0                                | Économique |
| 35 M                |                     | 223                  | Célibataire                        | 0                               | 0                                | Économique |
| 48 M                |                     | 401                  | Célibataire                        | 0                               | 0                                | Économique |
| 26 F                |                     | 420                  | En Couple                          | 3                               | 3                                | Sportive   |
| 80 M                |                     | 530                  | En Couple                          | 3                               | 3                                | Sportive   |
| 27 F                |                     | 153                  | En Couple                          | 2                               | 2                                | Citadine   |
| 59 F                |                     | 572                  | En Couple                          | 2                               | 2                                | Citadine   |
| 43 F                |                     | 431                  | Célibataire                        | 0                               | 0                                | Économique |
| 64 M                |                     | 559                  | Célibataire                        | 0                               | 0                                | Économique |
| 22 M                |                     | 154                  | En Couple                          | 1                               | 1                                | Citadine   |
| 79 F                |                     | 981                  | En Couple                          | 2                               | 2                                | Familiale  |
| 55 M                |                     | 588                  | Célibataire                        | 0                               | 0                                | Économique |
| 19 F                |                     | 212                  | Célibataire                        | 0                               | 0                                | Économique |
| 34 F                |                     | 1112                 | En Couple                          | 0                               | 0                                | Familiale  |
| 60 M                |                     | 524                  | En Couple                          | 0                               | 0                                | Économique |
| 22 M                |                     | 411                  | En Couple                          | 3                               | 3                                | Sportive   |
| 58 M                |                     | 1192                 | En Couple                          | 0                               | 0                                | Familiale  |
| 54 F                |                     | 452                  | En Couple                          | 3                               | 3                                | Sportive   |
| 35 M                |                     | 589                  | Célibataire                        | 0                               | 0                                | Économique |
| 59 M                |                     | 748                  | En Couple                          | 0                               | 0                                | Économique |

```
# Fermeture du curseur et de la connexion
cursor.close()
conn.close()
```

Le curseur est utilisé pour exécuter la requête SQL et la connexion à la base de données sont fermés pour libérer les ressources.

Cela conclut la partie de l'analyse des données avec les techniques de Data Mining, Machine Learning et Deep Learning.

## 8.6 Construction de la Base de données résultat

### 8.6.1 Insertion des données de prédictions dans la base de données résultat.

Après la préparation des données de prédition et leur exportation dans le fichier « resultat\_catalogue\_co2.csv », nous créons une table MySQL et y chargeons le fichier Excel. Cette table contiendra toutes les prédictions des catégories de voitures en fonction des données clients.

**mysql> CREATE DATABASE IF NOT EXISTS BD\_resultat;**

**mysql> USE BD\_resultat;**

```
mysql> CREATE TABLE Marketing_resultat ( age INT, sexe VARCHAR(1), taux INT, situationFamiliale VARCHAR(20), nbEnfantsAcharge INT, deuxieme_voiture VARCHAR(5), Catégorie VARCHAR(10) );
Query OK, 0 rows affected (0.04 sec)

mysql> Load data local infile '/vagrant/Groupe_TPT_8/marketing_resultat.csv' Into table Marketing_resultat fields terminated by ',' lines terminated by '\n' ignore 1 rows;
ERROR 3948 (42000): Loading local data is disabled; this must be enabled on both the client and server sides
mysql>
mysql> Load data local infile '/vagrant/Groupe_TPT_8/marketing_resultat.csv' Into table Marketing_resultat fields terminated by ',' lines terminated by '\n' ignore 1 rows;
ERROR 3948 (42000): Loading local data is disabled; this must be enabled on both the client and server sides
mysql> SHOW GLOBAL VARIABLES LIKE 'local_infile';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| local_infile | OFF |
+-----+-----+
1 row in set (0.01 sec)

mysql> SET GLOBAL local_infile='ON';
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GLOBAL VARIABLES LIKE 'local_infile';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| local_infile | ON |
+-----+-----+
1 row in set (0.00 sec)

mysql> Load data local infile '/vagrant/Groupe_TPT_8/marketing_resultat.csv' Into table Marketing_resultat fields terminated by ',' lines terminated by '\n' ignore 1 rows;
Query OK, 20 rows affected (0.01 sec)
Records: 20 Deleted: 0 Skipped: 0 Warnings: 0
```

Nous débutons en créant la base de données résultat nommée "BD\_resultat" et nous la sélectionnons par la suite. Ensuite, une table intitulée "Marketing\_resultat" est créée, comprenant des colonnes correspondant aux données marketing, ainsi qu'une colonne supplémentaire "Catégorie" qui stockera la prédition de la catégorie de voiture la plus appropriée pour chaque client en fonction de ses données.

Ensuite, nous importons les données du fichier CSV 'marketing\_resultat.csv' situé dans le chemin '/vagrant/Groupe\_TPT\_8/' dans la table MySQL « Marketing\_resultat » en utilisant la commande « LOAD DATA LOCAL INFILE ». Cependant, cette commande échoue initialement car le chargement de données locales est désactivé à la fois sur le client et sur le serveur. Après avoir activé le chargement de données locales avec la commande « SET GLOBAL local\_infile='ON' », on tente à nouveau d'importer les données dans la table « Marketing\_resultat ».

Assurons-nous que toutes les données de prédition ont été chargées correctement et intégralement.

```
mysql> SELECT * FROM Marketing_resultat |G; | sed 's/\r/';
```

```
***** 15. row *****
    age: 60
    sexe: M
    taux: 524
situationFamiliale: En Couple
nbEnfantsAcharge: 0
deuxieme_voiture: true
    Catégorie: Économique
***** 16. row *****
    age: 22
    sexe: M
    taux: 411
situationFamiliale: En Couple
nbEnfantsAcharge: 3
deuxieme_voiture: true
    Catégorie: Sportive
***** 17. row *****
    age: 58
    sexe: M
    taux: 1192
situationFamiliale: En Couple
nbEnfantsAcharge: 0
deuxieme_voiture: false
    Catégorie: Familiale
***** 18. row *****
    age: 54
    sexe: F
    taux: 452
situationFamiliale: En Couple
nbEnfantsAcharge: 3
deuxieme_voiture: true
    Catégorie: Sportive
***** 19. row *****
    age: 35
    sexe: M
    taux: 589
situationFamiliale: Célibataire
nbEnfantsAcharge: 0
deuxieme_voiture: false
    Catégorie: Économique
***** 20. row *****
    age: 59
    sexe: M
    taux: 748
situationFamiliale: En Couple
nbEnfantsAcharge: 0
deuxieme_voiture: true
    Catégorie: Économique
90 rows in set (0.00 sec)
```

La requête renvoie un total de 20 enregistrements, ce qui correspond au nombre d'entrées dans le fichier Excel. Ces enregistrements respectent la structure de leurs colonnes et comprennent des données spécifiques aux clients ainsi que les catégories de voitures suggérées.

## 8.6.2 Insertion des visualisations de l'analyse exploratoire dans la base de données résultat.

Dans un premier temps, nous avons importé les images de visualisations dans un répertoire Google Drive, il contient l'ensemble des images des visualisations et il est accessible via le lien suivant : [https://drive.google.com/drive/folders/1rA8jz\\_o9wonjd164HJ6iPFgs9BgDjUQX?usp=sharing](https://drive.google.com/drive/folders/1rA8jz_o9wonjd164HJ6iPFgs9BgDjUQX?usp=sharing)

```
mysql> CREATE TABLE resultat_visualisation (url_dossier_visualisation_googledrive VARCHAR(100));
Query OK, 0 rows affected (0.30 sec)

mysql> INSERT INTO resultat_visualisation (url_dossier_visualisation_googledrive) VALUES ('https://drive.google.com/drive/folders/1rA8jz_o9wonjd164HJ6iPFgs9BgDjUQX?usp=sharing');
Query OK, 1 row affected (0.28 sec)

mysql> Select * from resultat_visualisation;
+-----+
| url_dossier_visualisation_googledrive |
+-----+
| https://drive.google.com/drive/folders/1rA8jz_o9wonjd164HJ6iPFgs9BgDjUQX?usp=sharing |
+-----+
1 row in set (0.00 sec)
```

Ensuite, nous avons créé une table MySQL appelée "resultat\_visualisation", dans laquelle nous insérons l'URL du lien Google Drive. Nous vérifions ensuite que l'insertion a été effectuée correctement, comme vous pouvez le constater dans l'image ci-dessus.

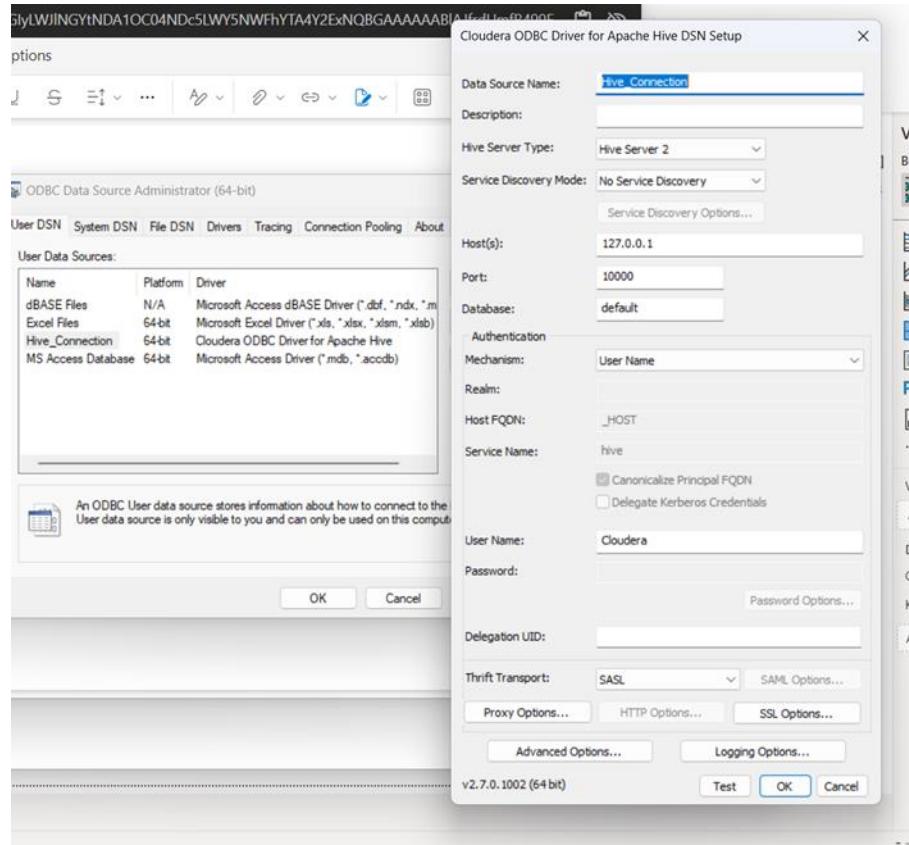
## 9. Creation de Dashboard avec Power Bi:

Dans le cadre de notre projet l'utilisation de Power BI a été d'une importance capitale pour l'interprétation et la communication des insights relevées à partir des données. En effet, Power BI nous a permis de créer des tableaux de bord interactifs et intuitifs, offrant ainsi une vue d'ensemble claire et précise de l'état et de la performance des ventes du concessionnaire automobile. Grâce à ses fonctionnalités avancées de visualisation, Power BI a fourni une plateforme centralisée pour l'analyse approfondie de la clientèle et des ventes. Ce faisant, il a grandement contribué à l'efficacité de notre processus d'analyse et à la prise de décision éclairée pour la recommandation de modèles de véhicules adaptés aux besoins des clients.

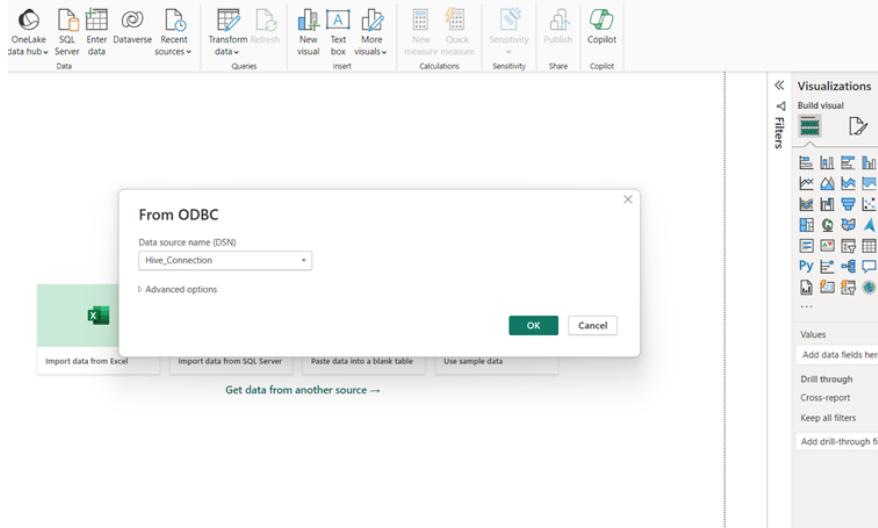
Pour l'implémentation des fonctionnalités visuelles de Power BI dans notre projet, nous nous sommes servis du pilote ODBC (Open Database Connectivity) fourni par Cloudera : le Hive ODBC Connector 2.7.0 for Cloudera Enterprise. .

Voici les étapes principales que nous avons suivies pour établir cette connexion :

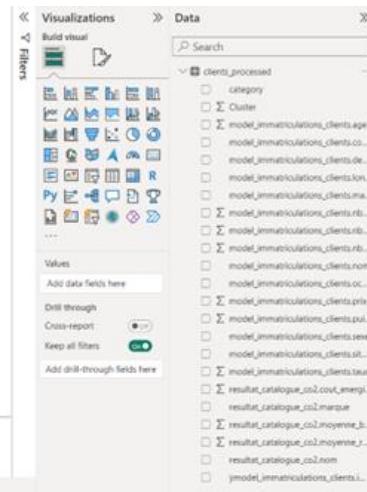
- Installation du pilote ODBC : Nous avons d'abord téléchargé et installé le pilote ODBC Hive Connector 2.7.0 fourni par Cloudera, en suivant les instructions d'installation fournies par le fournisseur.
- Configuration du pilote ODBC : Une fois le pilote installé, nous avons configuré la source de données dans l'administrateur ODBC de Windows. Nous avons fourni les détails de connexion requis, tels que l'adresse IP ou le nom d'hôte du serveur Hive, le port de connexion, les informations d'identification,



- Configuration de la connexion dans Power BI : Dans Power BI, nous avons ajouté une nouvelle source de données en sélectionnant l'option "Hive ODBC Connector" dans la liste des sources de données disponibles. Nous avons utilisé les mêmes détails de connexion que ceux configurés dans l'administrateur ODBC de Windows.

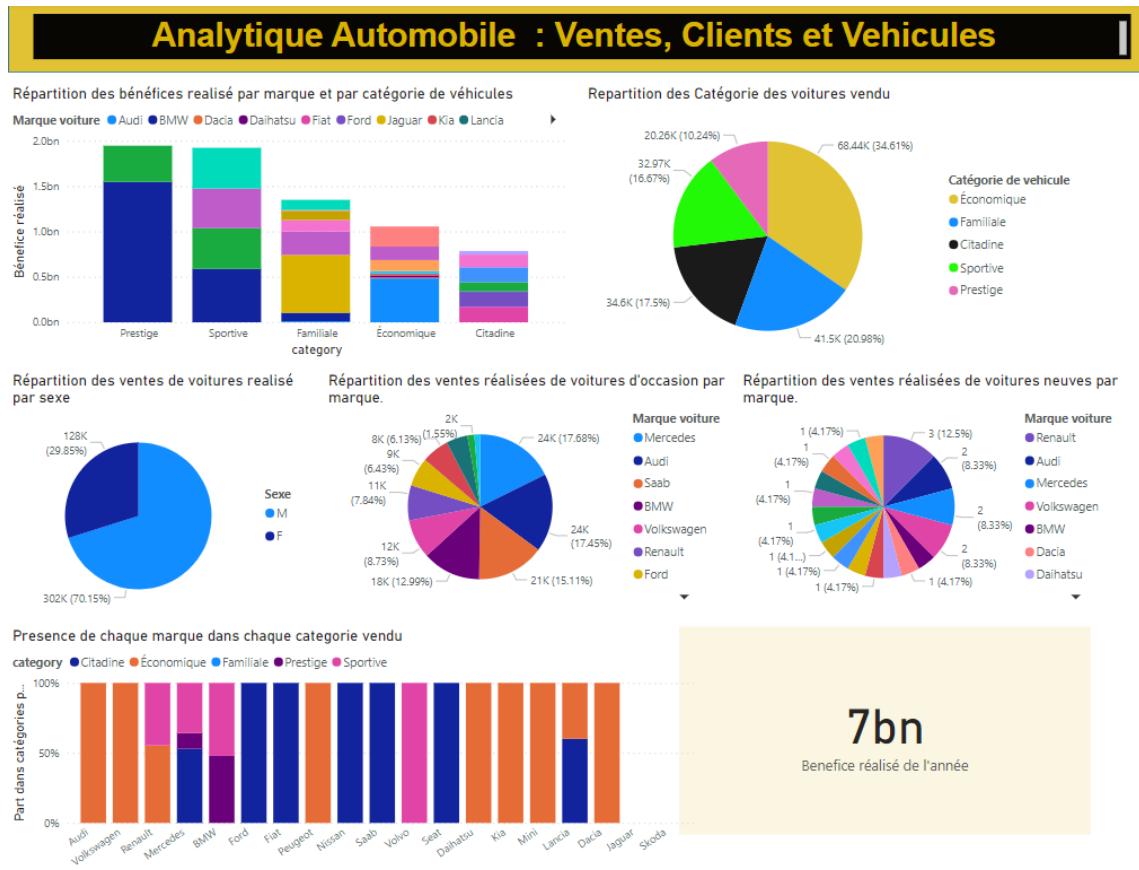


- La recuperation des données dans les tables Hive et on commence à construire les Dashboards :



## Dashboard Analytique Automobile : Ventes, Clients et Véhicules

Une vue d'ensemble complète des ventes, des clients et des véhicules du concessionnaire automobile. Ces visualisations offrent une perspective détaillée des performances de vente et des bénéfices générés par différentes marques, catégories de véhicules, sexe des clients et état (neuf ou d'occasion) des voitures.

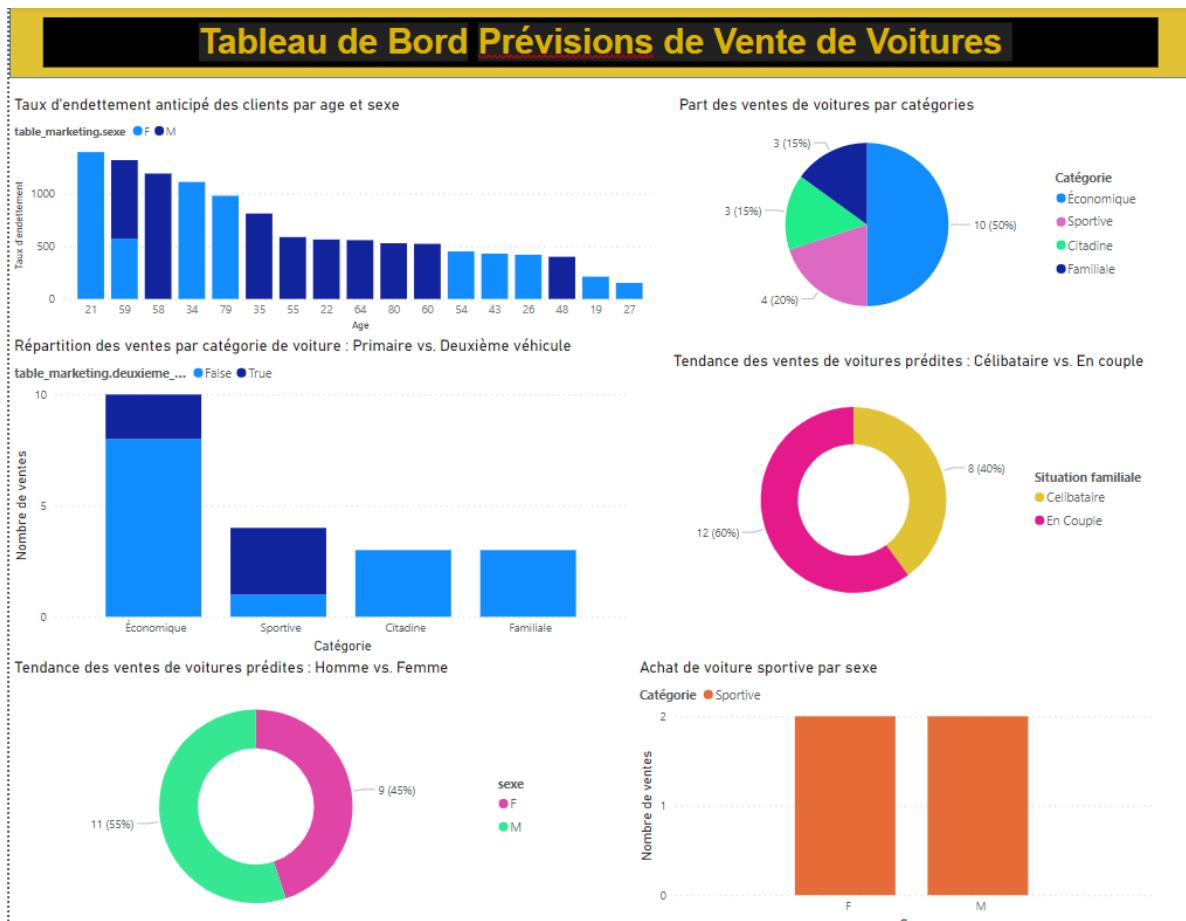


Voici un résumé de chaque graphique :

- **Répartition des bénéfices réalisés par marque et par catégorie de véhicules** : une vue croisée des bénéfices réalisés par chaque marque de voiture dans chaque catégorie de véhicules. Elle permet d'identifier les marques et les catégories de véhicules qui contribuent le plus aux bénéfices globaux.
- **Répartition des catégories des voitures vendues** : un aperçu de la répartition des ventes de voitures par catégorie, mettant en évidence les catégories les plus populaires parmi les clients.
- **Répartition des ventes de voitures réalisées par sexe** : Cette visualisation montre comment les ventes de voitures sont réparties entre les différents sexes des clients, offrant des insights sur les préférences d'achat entre hommes et femmes.
- **Répartition des ventes réalisées de voitures d'occasion par marque** : Cette visualisation met en évidence la répartition des ventes de voitures d'occasion par marque, permettant de voir quelles marques sont les plus populaires sur le marché de l'occasion.
- **Répartition des ventes réalisées de voitures neuves par marque** : celle-ci se concentre sur la répartition des ventes de voitures neuves par marque, fournissant des informations sur les préférences d'achat des clients pour les voitures neuves.
- **Présence de chaque marque dans chaque catégorie vendue** : Un aperçu de la présence de chaque marque de voiture dans chaque catégorie de véhicules vendue, montrant quelles marques sont représentées dans quels segments du marché.
- **Cadre affichant le bénéfice réalisé de l'année** : Ce cadre présente de manière concise le bénéfice total réalisé au cours de l'année, offrant un résumé des performances financières globales de l'entreprise.

Ces visualisations fournissent une analyse approfondie des ventes de voitures sous différents angles pour l'année en cours, ce qui permet de prendre des décisions éclairées en matière de marketing, de stratégie de vente et de gestion des stocks basées sur des performances commerciales passées. Il offre une rétrospective détaillée des ventes précédentes, permettant de comprendre les tendances émergentes et les modèles de comportement d'achat des clients. Les visualisations mettent l'accent sur les données historiques pour éclairer les décisions stratégiques et les actions futures.

### Dashboard Prévisions de Vente de Voitures:



Ce Dashboard des Prévisions de Vente de Voitures offre une analyse prospective des tendances du marché automobile, en mettant en lumière les facteurs démographiques les préférences des clients et les tendances émergentes qui influenceront les ventes à l'avenir. Nous explorons les données pour prédire les schémas d'achat, identifier les opportunités de marché et guider les stratégies de vente futures.

Voici un aperçu de chaque visualisation présente dans notre Dashboard des Prévisions de Vente de Voitures :

- Taux d'endettement anticipé des clients par âge et sexe : Explorez comment le taux d'endettement des clients varie en fonction de leur âge et de leur sexe. Cette visualisation offre des insights sur la capacité financière des clients à acheter une voiture.
- Part des ventes de voitures par catégories : Découvrez la répartition des ventes de voitures par catégorie, vous permettant de visualiser quelles catégories de voitures sont les plus populaires parmi les clients.
- Répartition des ventes par catégorie de voiture : Primaire vs. Deuxième véhicule : Explorez la répartition des ventes de voitures en distinguant entre les achats de véhicules primaires et les achats de deuxième véhicule. Cette visualisation offre des insights sur les habitudes d'achat des clients.

- Tendance des ventes de voitures prédictives : Célibataire vs. En couple : Découvrez comment les tendances de vente de voitures diffèrent entre les clients célibataires et ceux en couple. Cette visualisation offre des insights sur les préférences d'achat en fonction de l'état civil des clients.
- Tendance des ventes de voitures prédictives : Homme vs. Femme : Explorez les tendances de vente de voitures en distinguant entre les clients hommes et femmes. Cette visualisation offre des insights sur les différences d'achat entre les sexes.
- Achat de voiture sportive par sexe : Découvrez la répartition des ventes de voitures sportives en fonction du sexe des clients. Cette visualisation permet d'identifier les préférences d'achat des voitures sportives entre hommes et femmes.

Chacune des visualisations présentes dans notre Dashboard des Prévisions de Vente de Voitures offre une perspective sur les tendances et les prévisions de vente, fournissant ainsi une analyse détaillée et approfondie du marché automobile. Comparé au précédent dashboard, qui se concentrait sur l'analyse des ventes des performances commerciales passées, ce nouveau dashboard met l'accent sur les prévisions futures de vente et les facteurs qui influencent les décisions d'achat des clients.

En conclusion, cette partie de notre dashboard Power BI sur les Prévisions de Vente de Voitures offre une vision plus dynamique et orientée vers l'avenir du marché, en fournissant des insights précieux pour l'élaboration de stratégies de vente efficaces et la prise de décisions éclairées pour le succès futur du concessionnaire automobile.

## 10. Conclusion générale

Dans le cadre de ce projet , nous avons entrepris une analyse approfondie des données marketing afin de prédire les catégories de voitures les plus appropriées pour chaque client . Cette analyse a été menée à bien grâce à une combinaison de techniques de Data Mining,Machine learning et de Deep Learning .

**Le bilan des résultats obtenus pour l'entreprise :**

| Age | Sexe | Taux | Situation familiale | Nb. enfants à charge | Deuxième voiture | Catégorie  |
|-----|------|------|---------------------|----------------------|------------------|------------|
| 21  | F    | 1396 | Célibataire         | 0                    | false            | Économique |
| 35  | M    | 223  | Célibataire         | 0                    | false            | Économique |
| 48  | M    | 401  | Célibataire         | 0                    | false            | Économique |
| 26  | F    | 420  | En Couple           | 3                    | true             | Sportive   |
| 80  | M    | 530  | En Couple           | 3                    | false            | Sportive   |
| 27  | F    | 153  | En Couple           | 2                    | false            | Citadine   |
| 59  | F    | 572  | En Couple           | 2                    | false            | Citadine   |
| 43  | F    | 431  | Célibataire         | 0                    | false            | Économique |
| 64  | M    | 569  | Célibataire         | 0                    | false            | Économique |
| 22  | M    | 154  | En Couple           | 1                    | false            | Citadine   |
| 79  | F    | 981  | En Couple           | 2                    | false            | Familiale  |
| 55  | M    | 588  | Célibataire         | 0                    | false            | Économique |
| 19  | F    | 212  | Célibataire         | 0                    | false            | Économique |
| 34  | F    | 1112 | En Couple           | 0                    | false            | Familiale  |
| 60  | M    | 524  | En Couple           | 0                    | true             | Économique |
| 22  | M    | 411  | En Couple           | 3                    | true             | Sportive   |
| 58  | M    | 1192 | En Couple           | 0                    | false            | Familiale  |
| 54  | F    | 452  | En Couple           | 3                    | true             | Sportive   |
| 35  | M    | 569  | Célibataire         | 0                    | false            | Économique |
| 59  | M    | 748  | En Couple           | 0                    | true             | Économique |

Nous avons réussi à développer un modèle d'analyse efficace qui permet de prédire avec précision les catégories de voitures correspondant aux clients de l'entreprise. Ce modèle s'est

avéré être une ressource précieuse pour l'entreprise, lui permettant d'optimiser ses stratégies de marketing et de mieux cibler ses clients.

En analysant les données de marketing, nous avons pu observer des tendances significatives dans le comportement des clients. Par exemple, nous avons constaté que les clients célibataires sont plus enclins à opter pour des voitures de catégorie "Économique", tandis que les couples avec enfants préfèrent les voitures de catégorie "Familiale" ou "Sportive". Ces insights nous ont permis d'adapter nos stratégies de marketing pour répondre aux besoins spécifiques de chaque segment de clientèle .

Nous avons rencontré plusieurs défis tout au long du projet. La migration des données entre la base de données source et le datalake a été complexe, nécessitant une gestion minutieuse pour éviter les pertes de données. Pour résoudre ce problème, nous avons mis en place un gestionnaire de stockage dédié pour superviser le processus de transfert.

L'établissement du clustering dans le modèle d'analyse a également été un défi, nécessitant une analyse approfondie pour déterminer le nombre optimal de clusters et les séparer de manière logique. De plus, l'étiquetage des clusters a nécessité une attention particulière pour garantir leur pertinence.

Le manque de ressources dans notre système de gestion de base de données Hive a entravé la construction du modèle d'analyse, retardant ainsi le processus. Pour surmonter ce problème, nous avons introduit un intermédiaire pour optimiser l'utilisation des ressources disponibles.

Pour améliorer encore davantage notre solution, nous envisageons d'ajouter la capacité de prédire en temps réel en intégrant les données des bus dans notre modèle. Cela nous permettrait de fournir des recommandations en temps réel aux clients, améliorant ainsi leur expérience utilisateur et augmentant l'efficacité de notre système de recommandation.

Nous envisageons également d'explorer l'utilisation de Kafka Stream pour gérer le flux continu de données provenant des bus et d'autres sources en temps réel. Cela nous permettrait de traiter les données de manière efficace et de les intégrer rapidement dans notre processus de prédiction, améliorant ainsi la réactivité et la pertinence de nos recommandations.

Ce projet a été une expérience enrichissante à bien des égards, et nous avons beaucoup appris tout au long du processus. Sur le plan personnel, cette expérience nous a permis de développer et d'améliorer plusieurs compétences clés. Nous avons acquis une meilleure compréhension du processus complet, de la collecte des données à la construction de modèles prédictifs. En utilisant des outils comme Python et des bibliothèques telles que pandas et scikit-learn, nous avons manipulé efficacement de grandes quantités de données et construit des modèles d'apprentissage automatique. La collaboration au sein de l'équipe nous a aidés à résoudre les problèmes et à prendre des décisions stratégiques. Cette expérience nous a également permis de mieux comprendre l'importance de l'analyse de données dans le domaine du marketing. En résumé, ce projet a été une expérience enrichissante qui nous a permis de développer un large éventail de compétences.

## 11. Références et Bibliographie

- Alain Clapaud, « Qu'est-ce que le Data Lake, le nouveau concept "Big Data" en vogue », sur Le Journal du Net, 15 octobre 2015 (consulté le 22 juin 2016).

Lien : <https://www.journaldunet.fr/web-tech/guide-du-big-data/1165409-data-lake-ou-lac-de-donnees-la-solution-reine-du-big-data/>

- Robert Nisbet, John Elder et Gary Miner, « Handbook of Statistical Analysis & Data Mining Applications, Amsterdam/Boston, Academic Press», 2009.
- Tony Baer, « It's MongoDB's turn to change its open-source license », ZDNet, 16 octobre 2018.

<https://www.zdnet.com/article/its-mongodbs-turn-to-change-its-open-source-license/>

- « Changes in MySQL 8.4.0 (2024-04-30, LTS Release) », 30 avril 2024

Lien : <https://dev.mysql.com/doc/relnotes/mysql/8.4/en/news-8-4-0.html>

- JDBC Storage Handler:

<https://cwiki.apache.org/confluence/display/Hive/JDBC+Storage+Handler>

- Hive Storage Handler:

[https://phoenix.apache.org/hive\\_storage\\_handler.html?platform=hootsuite](https://phoenix.apache.org/hive_storage_handler.html?platform=hootsuite)

## 12. Annexes

### 12.1 Vidéo de présentation de votre projet

Vous pouvez accéder à la vidéo soit via le fichier du rapport remis, ou bien en utilisant le lien Google Drive suivant :

<https://drive.google.com/file/d/11ut0isGYyiDELZ0UEbRFJ5ffBR8Ft2Zg/view?usp=sharing>

### 12.2 Dossier contenant les scripts et programmes de construction du lac de données

[https://drive.google.com/drive/folders/1BM4TJqWmLPIwta1cm03F25h8NXbg\\_7VO?usp=sharing](https://drive.google.com/drive/folders/1BM4TJqWmLPIwta1cm03F25h8NXbg_7VO?usp=sharing)

### 12.3 Dossier contenant les scripts et programmes Hadoop Map Reduce

<https://drive.google.com/drive/folders/1fmDj2eetxabNNGeZNsOy6cbjervu0lr5?usp=sharing>

### 12.4 Dossier contenant les scripts et programmes de visualisations et d'analyse de données

<https://drive.google.com/drive/folders/14HyVEP0zphlJtBaW4nSndfwBnbVEHs7x?usp=sharing>