Projet Big Data - Fromagerie

15 février 2024

Membres du Groupe :

RANDRIANARIVONY Rijandrisolo
GROBOST Garmi
MENTSEUR Fares
SALAHY Jaouad

1. Problématiques

1.1 Problématique business:

La société Brennais, établie dans la région de Rennes, fabrique et commercialise des fromages.
Pour renforcer la loyauté de sa clientèle, elle a lancé un programme de fidélisation, qui récompense les achats de fromage par l'octroie de points échangeables contre une sélection de cadeaux.

Dans le but d'optimiser la gestion de ce programme de fidélité, Brennais envisage l'élaboration de tableaux de bord. Cette initiative vise principalement deux objectifs analytiques :

- Premièrement, analyser la dynamique du nombre de clients adhérant au programme de fidélité, afin de mieux orienter les opérations marketing. L'objectif est de stimuler l'engagement des clients existants tout en attirant de nouveaux participants, contribuant ainsi à l'expansion des ventes.

- Deuxièmement, évaluer et réduire les coûts de distribution des cadeaux dans le but d'accroître les marges bénéficiaires de l'entreprise.

Cette démarche stratégique permettra à Brennais non seulement de fidéliser sa clientèle grâce à un programme attractif mais aussi d'optimiser ses opérations commerciales pour une meilleure rentabilité.

1.2 Problématique technique:

Face à une augmentation significative des volumes de données générées par son programme de fidélisation, ainsi qu'à la complexité croissante de leur analyse, la société Brennais s'est orientée vers l'adoption de technologies Big Data et NoSQL. Cette démarche vise à exploiter plus efficacement ces données pour renforcer le programme de fidélité et dévoiler de nouvelles opportunités de croissance.

Dans ce contexte, Brennais a choisi Hadoop comme plateforme de traitement de données distribuées qui permet de stocker et de traiter de grands volumes de données de manière efficace et économique.
Son écosystème, comprenant notamment HBase, un système de gestion de base de données NoSQL qui s'appuie sur le modèle BigTable, est particulièrement bien adapté.

Les données actuellement conservées sous forme de fichiers CSV présentent des limitations en termes d'exploitation directe, notamment en ce qui concerne l'analyse en temps réel et la scalabilité.

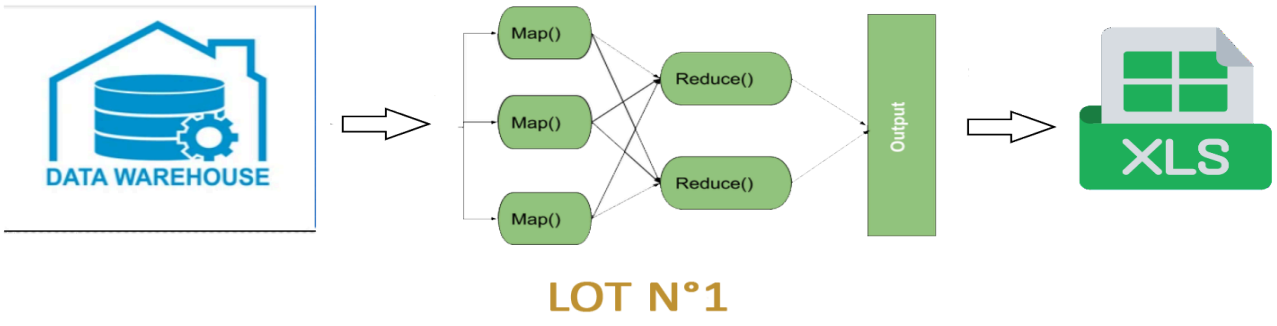
Une migration des données vers Hbase est donc planifiée. Voici l’analyse préalable à cette migration.

2. Qualification des données

Le tableau ci-dessous regroupe les différents champs du fichier csv source et renseigne sur la qualité exploitable des données ainsi que sur leur distribution statistique:

Nom	Désignation	Type	% valeur null	% valeur 0	count	mean	std	min	25%	50%	75%	max	Commentaire / Limite
codcli	code client	str	0,00%		135277								
genrecli	genre client	str	0,54%		134552								
nomcli	nom client	str	0,00%		135277								
prenomcli	prénom client	str	0,24%		134953								
cpcli	code postal client	str	0,00%		135277								
villecli	ville client	str	0,00%		135277								
codcde	numéro de commande	str	0,00%		135277								
datcde	date de commande	date	0,00%		135275								12 dates présentent une année inf. à 1900 ou sont null.
timbrecli	timbre client	float	0,00%	96,64%	135273	0,11	0,76	0,00	0,00	0,00	0,00	20,25	97% de valeur à 0. Données non exploitables.
timbrecde	timbre commande	float	0,01%		135268	5,08	2,36	0,64	2,75	5,30	6,50	17,90	
Nbcolis	nombre de colis inclus dals la commande	int	0,01%		135269	1,02	0,15	1,00	1,00	1,00	1,00	7,00	
cheqcli	chèque client	float	0,03%	3,93%	135234	4,71	2,60	0,00	2,86	4,60	6,20	180,00	
barchive	status d'archivage de la commande	bool	0,00%		135277								
bstock	disponibilité en stock	bool	0,00%		135277								
codobj	code article	int	0,00%		135277	68,61	34,23	20,00	42,00	65,00	84,00	168,00	
qte	quantité article	int	0,00%		135274	1,44	1,44	1,00	1,00	1,00	1,00	300,00	
Colis	numero de colis associé à l'objet dans la commande	int	0,01%	0,00%	135269	1,02	0,15	0,00	1,00	1,00	1,00	7,00	
libobj	libellé article	str	0,00%		135274								
Tailleobj	taille de l'article	str	64,10%		48558								2/3 des enregistrements sont manquantes.
Poidsobj	poids de l'article	int	0,00%	30,52%	135277	150,37	247,47	0,00	0,00	60,00	250,00	9300,00	1/3 des données sont à 0.
points	valeur de l'article en points clients	int	0,19%	15,14%	135015	12,60	157,76	-2500,00	0,00	60,00	80,00	360,00	21k enregistrements présentent des valeurs < 0.
indispobj	non-disponibilité en stock de l'article	bool	0,00%		135277								
libcondit	libellé du conditionnement	str	0,00%		135277								
prixcond	prix du conditionnement	int	0,00%	56,75%	135277	51,44	112,70	0,00	0,00	0,00	34,00	655,00	Données non exhaustives.
puobj	non applicable	int	0,00%		135277	0,00	0,00	0,00	0,00	0,00	0,00	0,00	
Total enregistrement					**135277**								

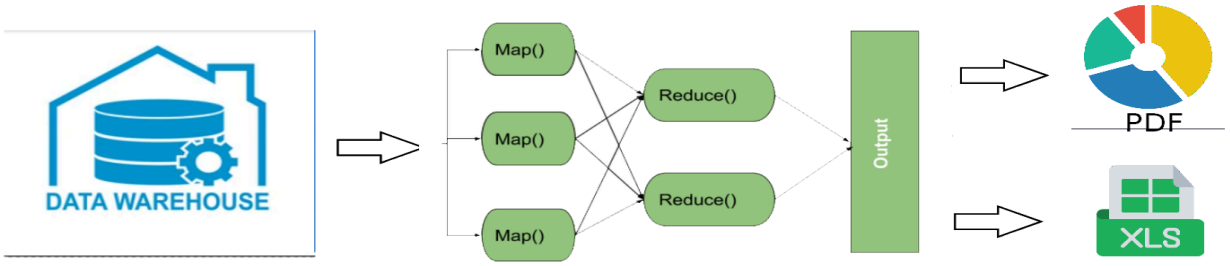
3. Procédures d'import des données



LOT N°1

L'objectif de cette procédure est d'importer des données d'un data warehouse, en format csv, en utilisant MapReduce et de sauvegarder les résultats dans un fichier Excel, en utilisant Python.

- Configuration de l'Environnement (cluster hadoop) pour l'exécution de tâches MapReduce
- Développement du Programme MapReduce avec les contraintes imposées
- Exécution du Job MapReduce
- Sauvegarde dans un fichier Excel.



LOT N°2

La procédure du lot N°2 a été pareille que celle du lot N°1. Les différences étaient les contraintes imposées pour les filtres et la nécessité d'avoir, en plus, une sortie en pdf avec un graphe (PIE) par ville pour le lot N°2.

LOT N°3

Cette procédure appliquée au lot N°3 fournit une méthodologie complète pour l'import de données depuis un ordinateur local vers un cluster Hadoop avec HBase, en utilisant Docker. Le script Python avec HappyBase facilite la lecture, le formatage, avant l'insertion des données dans le système distribué.

4. Procédures de structuration

4.1 Modèle de données:

Il a été décidé de garder les données sous la forme d'une seule table dans un premier temps.

4.2 Traitement des valeurs null:

- Pour les données int et float, les valeurs null ont été remplacées par 0.
- Pour les données str, les valeurs null ont été remplacées par une chaîne de caractère vide.

4.3 Traitement des valeurs zéro:

- Pour les quantités "qte", les valeurs NULL et 0 ont été remplacées par la valeur 1 par hypothèse qu'un objet est commandé pour au moins une quantité.

4.4 Traitement des valeurs booléenne:

- Pour les champs "barchive", "bstock" et "indispobj" dont leurs valeurs est 0 ou 1, ont été renseignés par les valeurs booléenne True ou False.

5. Algorithmes d'analyse des données

5.1 Logique du Reducer Lot 1 :

```python
import sys
from io import StringIO
import pandas as pd



# Lire l'entrée du stdin (cat fichier.csv)
data = sys.stdin.buffer.read()

# Transformer data en un DataFrame Pandas
df = pd.read_csv(StringIO(data.decode('utf-8')), sep=',', header=None,
                 names=['code_commande', 'ville_client', 'timbre_commande', 'quantite'], na_values=['NULL'])

# Grouper par code commande, pour effacer les doublons de commande et une aggrégation sur la quantité pour calculer la quantité total de chaque commmande
groupedby_data = df.groupby(['code_commande', 'ville_client', 'timbre_commande']).agg({'quantite': 'sum'}).reset_index()
# Faire un tri sur la quantité et timbre commande pour avoir un classement décroissant des commandes
sorted_data = groupedby_data.sort_values(by=['quantite', 'timbre_commande'], ascending=[False, False])

# Faire une limit à 100 et exporter les résultats dans un fichier xlsx
sorted_data.head(100).to_excel('resultats/lot1/projet_hadoop_bigdata_lot1.xlsx', index=True)
```

Ce script Python prend en entrée des données depuis `stdin` au format CSV, les transforme en un DataFrame Pandas, puis effectue les opérations suivantes :

- Les données CSV sont lues depuis l'entrée standard (`stdin`).

- Le module `io.StringIO` est utilisé pour créer un objet de fichier en mémoire à partir des données binaires lues, puis le décodage est effectué en utilisant UTF-8 pour obtenir une chaîne de caractères.
- Les données sont converties en un DataFrame Pandas en utilisant `pd.read_csv()`. Les colonnes sont nommées 'code_commande', 'ville_client', 'timbre_commande' et 'quantite'.
- Les valeurs 'NULL' sont traitées comme valeurs manquantes (`na_values=['NULL']`).
- Les données sont groupées par 'code_commande', 'ville_client' et 'timbre_commande'.
- Une agrégation est effectuée sur la colonne 'quantite' pour calculer la somme de la quantité pour chaque groupe.
- Les résultats sont réinitialisés pour obtenir un DataFrame plat.
- Les données agrégées sont triées en fonction de la quantité de commande et du timbre de commande, de manière décroissante.
- Les 100 premières lignes des résultats triés sont exportées vers un fichier Excel (`.xlsx`) dans le répertoire `resultats/lot1/`.

5.2 Logique du Reducer Lot 2 :

```python
import sys
from io import StringIO
import pandas as pd
import matplotlib.pyplot as plt

# Lire l'entrée du stdin (cat fichier.csv)
data = sys.stdin.buffer.read()
# Transformer data en un DataFrame Pandas
df = pd.read_csv(StringIO(data.decode('utf-8')), sep=',', header=None,
                 names=['code_commande', 'ville_client', 'timbre_commande', 'quantite', 'libobj'], na_values=['NULL'])
# Grouper par code commande, pour effacer les doublons de commande
groupedby_data = df.groupby(['code_commande', 'ville_client', 'timbre_commande']).agg(
    {'quantite': ['sum', 'mean']}).reset_index()
# Faire un tri sur la quantité et timbre commande pour avoir un classement décroissant des commandes
sorted_data = groupedby_data.sort_values(by=[('quantite', 'sum'), 'timbre_commande'], ascending=[False, False])
# Garder que les 100 meilleures commandes
sample_data = sorted_data.head(100)
# Extraire un échantillon aléatoire qui représente 5%
random_data = sample_data.sample(frac=0.05)
# Exporter dans un fichier Xlsx
random_data.to_excel('resultats/lot2projet_hadoop_bigdata_lot2.xlsx', index=True)
# Données nécessaires pour le graphe en PIE
labels = random_data['ville_client']
sizes = random_data[('quantite', 'sum')]
# Tracer le diagramme circulaire
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title('Distribution aléatoire des 5 meilleurs commandes')
plt.savefig('resultats/lot2/Pie_Diagram_Lot2.pdf')
```

Ce script prend en entrée des données CSV à partir de `stdin`, les traite et les analyse, puis génère deux sorties :

- Les données CSV sont lues depuis l'entrée standard (`stdin`) et converties en un DataFrame Pandas.

- Les données sont groupées par 'code_commande', 'ville_client' et 'timbre_commande', puis agrégées pour calculer la somme et la moyenne de la quantité de commande.
- Les données sont ensuite triées en fonction de la somme de la quantité de commande et du timbre de commande, de manière décroissante.
- Les 100 meilleures commandes sont sélectionnées et un échantillon aléatoire représentant 5% de ces données est extrait.
- Les données sélectionnées sont exportées vers un fichier Excel (`resultats/lot2projet_hadoop_bigdata_lot2.xlsx`).
- Les données de l'échantillon aléatoire sont utilisées pour générer un diagramme circulaire.
- Les étiquettes du diagramme correspondent aux villes des clients, et les tailles des secteurs représentent les quantités de commande.
- Le diagramme est sauvegardé au format PDF (`resultats/lot2/Pie_Diagram_Lot2.pdf`).

5.3 Logique Lot 3 Question 1 :

```python
# Fonction, qui à partir d'une entrée dans hbase, on peut extraire les couple (Row_Qualifier, Value)
def get_hbase_data(datas):
    for key, value in scanner:
        decoded_data = {k.decode('utf-8'): v.decode('utf-8') for k, v in value.items()}
        yield {'key': key.decode('utf-8'), **decoded_data}
# Connection à la base
connection = happybase.Connection('node175998-env-1839015-etudiant27.sh1.hidora.com', 11669)  # 9090
connection.open()
# Selectionner la table sur laquelle chercher les données
table_name = b'datafromagerie'
table = connection.table(table_name)
# Charger la table avec un filter sur la ville de NANTES
scanner = table.scan(filter="SingleColumnValueFilter('cf', 'villecli',=,'binary:NANTES')")
# Convertir le résultat chargé de la table en un DataFrame Pandas
data = list(get_hbase_data(scanner))
df = pd.DataFrame(data)
# Caster le type de la colonne (Date Commande) en type DataTime
df['cf:datcde'] = pd.to_datetime(df['cf:datcde'])
# Considérer les quantités égales à NULL à 1
df['cf:qte'] = [int(qte) if qte != "NULL" else 1 for qte in df['cf:qte']]
# Faire une premier Filtre sur l'année 2020
filtered_df = df[df['cf:datcde'].dt.year == 2020]
# Faire un premier regroupement sur le Code-Commande et Timbre-Commande avec une aggrégation SUM sur Quantité
# pour en garder qu'une seule valeur Timbre-Commande par Commande
groupedby_data = filtered_df.groupby(['cf:codcde', 'cf:villecli', 'cf:timbrecde']).agg({'cf:qte': 'sum'}).reset_index()
# Faire un tri décroissant sur la Quantité et TimbreCode pour en choisir la meilleure commande
sorted_data = groupedby_data.sort_values(by=['cf:qte', 'cf:timbrecde'], ascending=[False, False]).head(1)
# Renommer les colonnes pour l'export
sorted_data = sorted_data.rename(columns={'cf:codcde': 'Code commande',
                                          'cf:villecli': 'Ville client',
                                          'cf:timbrecde': 'Timbre commande',
                                          'cf:qte': 'Quantite'})
```

Ce script Python se connecte à une base de données HBase, scanne une table avec un filtre sur la ville de Nantes, puis effectue les opérations suivantes :

- Une fonction get_hbase_data est définie pour extraire les données de chaque ligne scannée dans HBase et les convertir en un dictionnaire Python.
- Le programme se connecte à la base de données HBase en utilisant HappyBase.
- Les données de la table HBase sont scannées avec un filtre qui ne sélectionne que les lignes où la colonne 'villecli' a la valeur 'NANTES'.
- Les données scannées sont converties en un DataFrame Pandas.
- La colonne 'cf:datcde' est convertie en type datetime.
- Les valeurs 'NULL' dans la colonne 'cf:qte' sont remplacées par 1.
- Les données sont filtrées pour ne conserver que celles de l'année 2020.
- Les données sont regroupées par 'cf:codcde' (code de commande), 'cf:villecli' (ville du client) et 'cf:timbrecde' (timbre de commande), en effectuant une somme sur la colonne 'cf:qte' (quantité).
- Les données sont triées de manière décroissante en fonction de la quantité de commande et du timbre de commande.
- Seule la meilleure commande est conservée (celle avec la quantité la plus élevée et le timbre de commande le plus élevé).
- Les résultats sont exportés vers un fichier CSV nommé 'meilleur_commande_nantes_2020.csv' dans le dossier 'resultats/lot3/'.

5.4 Logique Lot 3 Question 2 :

```python
import happybase
import pandas as pd

"""
Ce programme :
    - Connecte à la base Hbase
    - Scan toute la table avec un filter sur les années 2010 et 2015
    - Extraire le total de données pour chacune des années entre 2010 et 2015
    - Exporter le résulat sous forme d'un 'Graphe Pie'
    - Enregistrer le visule dans un fichier PDF
"""
# Définir les deux dates limites pour les filtres
start_date = 2010
end_date = 2015

# Fonction, qui à partir d'une entrée dans hbase, on peut extraire les couple (Row_Qualifier, Value)
def get_hbase_data(datas):
    for key, value in datas:
        decoded_data = {k.decode('utf-8'): v.decode('utf-8') for k, v in value.items()}
        yield {'key': key.decode('utf-8'), **decoded_data}
# Connection en base
connection = happybase.Connection('node175998-env-1839015-etudiant27.sh1.hidora.com', 11669)  # 9090
connection.open()
# Connection à la tabe
table_name = b'datafromagerie'
table = connection.table(table_name)
# Scan de la table avec un filter sur les deux années 2010 <= Date Commande <= 2015
scanner = table.scan(filter=f"SingleColumnValueFilter('cf', 'datcde', >=, 'binary:{start_date}', true, false) AND SingleColumnValueFilter(
# Transformer les données récupérées de la base en DataFrame PANDAS
data = list(get_hbase_data(scanner))
df = pd.DataFrame(data)
# Créer une colonne yearcde à partir de datcde pour faire les filtres
df['cf:yearcde'] = pd.to_datetime(df['cf:datcde']).dt.year
```

Ce script Python se connecte à une base de données HBase, scanne une table avec un filtre sur les années 2010 à 2015, puis effectue les opérations suivantes :

- Une fonction get_hbase_data est définie pour extraire les données de chaque ligne scannée dans HBase et les convertir en un dictionnaire Python.
- Le programme se connecte à la base de données HBase en utilisant HappyBase.
- Les données de la table HBase sont scannées avec un filtre qui sélectionne les lignes où la colonne 'datcde' (date de commande) est comprise entre 2010 et 2015.
- Les données scannées sont converties en un DataFrame Pandas.
- Une colonne 'cf:yearcde' est créée à partir de la colonne 'cf:datcde' pour stocker l'année de la commande.
- Les données sont regroupées par année de commande et code de commande pour calculer le nombre total de commandes pour chaque année.
- Le nombre total de commandes pour chaque année est exporté sous forme d'un graphique circulaire (pie chart).
- Le graphique est enregistré dans un fichier PDF.

5.5 Logique Lot 3 Question 3 :

```python
import happybase
import pandas as pd

"""
Ce programme :
    - Connecte à la base Hbase
    - Scan toute la table avec un filter sur la ville de Nantes
    - Calculer la meilleure commande sur l'année 2020
    - Exporter le résultat sur un fichier CSV
"""
# Fonction, qui à partir d'une entrée dans hbase, on peut extraire les couple (Row_Qualifier, Value)
def get_hbase_data(datas):
    for key, data in datas:
        decoded_data = {k.decode('utf-8'): v.decode('utf-8') for k, v in data.items()}
        yield {'key': key.decode('utf-8'), **decoded_data}
# Connection à la base
connection = happybase.Connection('node175998-env-1839015-etudiant27.sh1.hidora.com', 11669)  # 9090
connection.open()
# Selectionner la table sur laquelle chercher les données
table_name = b'datafromagerie'
table = connection.table(table_name)
# List de colonnes à charger de la table
row_to_fetch = [b'cf:codcli', b'cf:prenomcli', b'cf:nomcli', b'cf:qte', b'cf:timbrecde', b'cf:codcde']
# Charger toute la table
scanner = table.scan(columns=row_to_fetch)
# Convertir le résultat chargé de la table en un DataFrame Pandas
data = list(get_hbase_data(scanner))
df = pd.DataFrame(data)
# Considérer les quantités égales à NULL comme égale à 1
df['cf:qte'] = [int(qte) if qte != "NULL" else 1 for qte in df['cf:qte']]
# Considérer les timbres-commande égales à NULL comme égale à 0
df['cf:timbrecde'] = [float(timbre_code) if timbre_code != "NULL" else 0 for timbre_code in df['cf:timbrecde']]
```

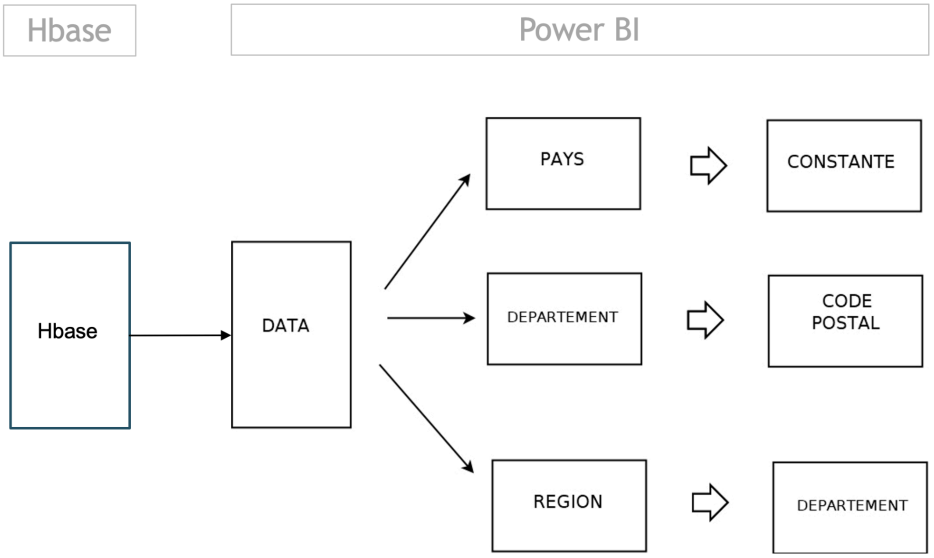
Ce script Python se connecte à une base de données HBase, scanne une table avec un filtre sur la ville de Nantes, puis effectue les opérations suivantes **:**

- Une fonction get_hbase_data est définie pour extraire les données de chaque ligne scannée dans HBase et les convertir en un dictionnaire Python.
- Le programme se connecte à la base de données HBase en utilisant HappyBase.
- Les données de la table HBase sont scannées avec un filtre qui sélectionne les lignes où la colonne 'villecli' (ville du client) est égale à 'Nantes'.
- Les données scannées sont converties en un DataFrame Pandas.
- Les quantités égales à NULL sont considérées comme égales à 1.
- Les timbres de commande égaux à NULL sont considérés comme égaux à 0.
- Les données sont regroupées par code client, prénom client et nom client.
- Pour chaque groupe, la somme des quantités commandées, la somme des timbres de commande et le nombre de commandes sont calculés.
- Les données sont triées par somme des timbres de commande en ordre décroissant, et seule la meilleure ligne est conservée.
- Les données de la meilleure ligne sont exportées dans un fichier Excel.
- Le fichier Excel est enregistré sous le nom 'meilleur_client_timbre_code.xlsx' dans le dossier 'resultats/lot3/'.

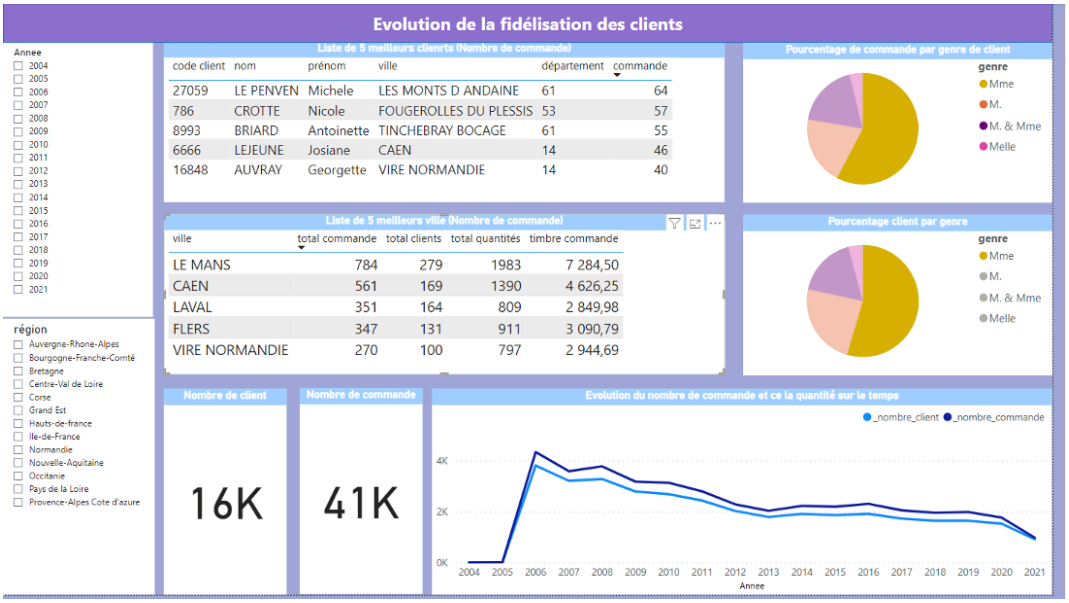
5.6 Logique Lot 4: La visualisation des données dans Power BI:

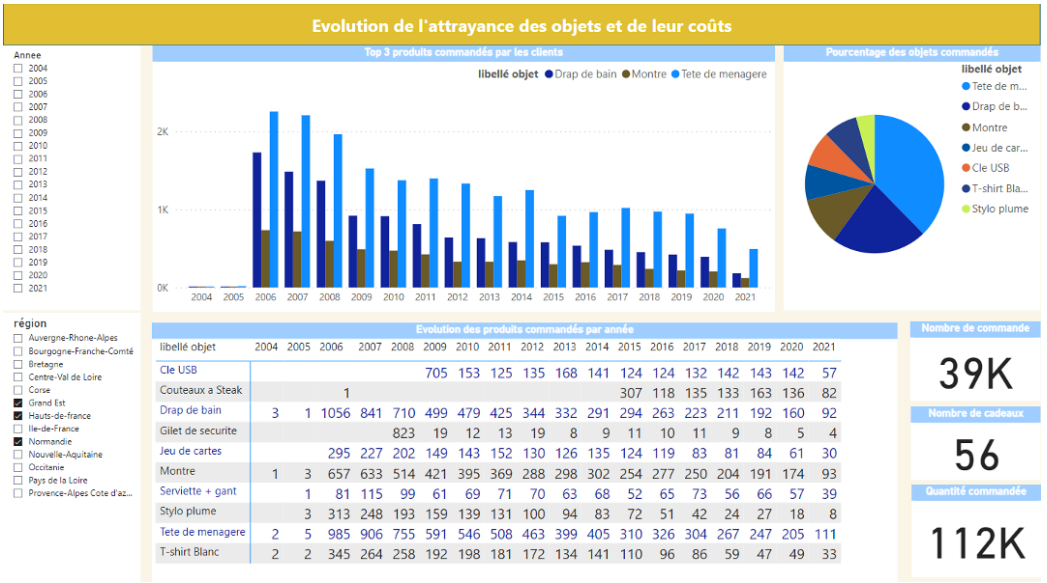
Des regroupements de données géographiques ont été ajoutés pour une meilleure synthétisation et comparaison des données dans l'espace: Pays, Département, Région.

Même si tous les enregistrements concernent la France, l'ajout 'pays' a permis de s'assurer que les données s'affichent bien sur la carte de France uniquement. Sans cela, des données s'affichent sur d'autres continents.



Tableaux de bords Power BI:





6. Recommandations

- **Nette baisse observée de l'adhésion des clients au programme de fidélisation:**
    - Identifier les root causes par le biais de méthodes qualitatives (enquête de satisfaction clients)
    - Mettre à jour le catalogue d'objets
    - Evaluer la cohérence des rapports points/objet

- **Les hommes et jeunes filles sont moins représentés:**
    - Adapter le catalogue aux goûts des différents segments clients
    - Cibler des actions marketing spécifiques

- **Croiser les données internes avec des données externes:**
    - Mettre en perspective la performance des ventes de l'entreprise avec le potentiel commerciale par ville

- **Monitorer le coûts de distribution par taille / poids au km parcouru**

- **Améliorer la qualité des données pour une meilleure exploitation:**
    - Exemple: 56 % des enregistrement sont vide pour le prix du conditionnement