IMG_257

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

**Groupe TPA 4**

RAFAMANTANANTSOA Rotsy Vonimanitra

RAMAROSON Sandy Princia

RAMIANDRISOA Rantonantenaina Steve

RATSIRARSON Joharisoa

**Avril 2024 – 9 Juin 2024**

**Enseignants Encadreurs :**

Gabriel MOPOLO

Sergio SIMONIAN

Nicolas PASQUIER

**Résumé du projet**

Ce projet vise à mettre en place un service pour un concessionnaire automobile, permettant ainsi une meilleure compréhension des besoins des clients et la proposition des véhicules les plus adaptés à leurs attentes. En outre, cela favorisera leur essor dans l'industrie du commerce automobile.

C’est un projet de grande envergure incluant le traitement de grandes quantités de données avec la mise en place d’un DataLake et l’analyse des données y figurant.

À travers des méthodes, telles que l'accès aux sources de données via Access Driver et Data Extractor, la mise en place d'un DataLake sur HiveQL, l'utilisation de Hadoop Map Reduce pour le nettoyage et l'analyse des données avec Python, nous garantirons des résultats précis et fiables.

Les principaux résultats de ce travail incluront une architecture du projet optimisée, des données nettoyées et cohérentes, ainsi que des analyses prédictives pour identifier les véhicules les plus appropriés pour chaque client

En conclusion, ce projet permettra aux concessionnaires automobiles de proposer rapidement des véhicules adaptés aux clients qui se présentent en concession, renforçant ainsi leur compétitivité sur le marché.  
  
**Mots clés :** Big Data, Analyse de données, Data Lake, ELT, Hadoop Map Reduce, Python, Access Driver, HiveQL.

**Abstract**

The aim of this project is to set up a service for a car dealership, enabling it to better understand customers' needs and suggest the most suitable vehicles for them. It will also help them to grow in the automotive trade.

This is a large-scale project involving the processing of large quantities of data, with the setting up of a DataLake and the analysis of the data contained therein.

Through methods such as accessing data sources via Access Driver and Data Extractor, setting up a DataLake on HiveQL, using Hadoop Map Reduce for data cleansing and analyzing the data with Python, we will guarantee accurate and reliable results.

The main results of this work will include an optimized project architecture, cleansed and consistent data, and predictive analytics to identify the most appropriate vehicles for each customer.

In conclusion, this project will enable car dealers to rapidly offer suitable vehicles to customers who come to the dealership, thus strengthening their competitiveness on the market.

**Keywords :** Big Data, Data analysis, Data Lake, ELT, Hadoop Map Reduce, Python, Access Driver, HiveQL.

**Liste des figures**

[Figure 1 : Architecture DBA/DL 12](#_Toc168260336)

[Figure 2 : Résultat du processor GetMongo 20](#_Toc168260337)

[Figure 3 : Résultat des processors ReplaceText 20](#_Toc168260338)

[Figure 4 : Liste des fichiers temporaires HDFS 21](#_Toc168260339)

[Figure 5 : Résultat du processor ReplaceText 21](#_Toc168260340)

[Figure 6 : Log d'insertion des données dans Hive 21](#_Toc168260341)

[Figure 7 : Architecture du processus d'alimentation avec NiFi 22](#_Toc168260342)

**Liste des acronymes** **Table des matières**

Table des matières

[1. Introduction générale 7](#_Toc168260291)

[2. Présentation du projet 8](#_Toc168260292)

[3. Répartition du travail en membre du groupe 9](#_Toc168260293)

[4. Architecture du Data Lake 9](#_Toc168260294)

[5. Construction du data lake par étape 13](#_Toc168260295)

[1. Démarrage de la machine virtuelle et configuration 13](#_Toc168260296)

[2. Mise en place des sources de données 14](#_Toc168260297)

[a. Source Hadoop Distributed File System (HDFS) 14](#_Toc168260298)

[b. Source Oracle NoSQL 14](#_Toc168260299)

[c. Source MongoDB 15](#_Toc168260300)

[3. Mise en place du frontal de données sur HiveQL Data Lake 16](#_Toc168260301)

[a. Création des données virtuelles Hive 16](#_Toc168260302)

[b. Création des données physiques Hive 17](#_Toc168260303)

[c. Création des données de modèle d’analyse 23](#_Toc168260304)

[4. Exploitation des données sur Python 23](#_Toc168260305)

[6. Hadoop Map Reduce 23](#_Toc168260306)

[7. Analyse de données avec Python 23](#_Toc168260307)

[8. Conclusion générale 24](#_Toc168260308)

[9. Références et Bibliographie 25](#_Toc168260309)

[10. Annexes 25](#_Toc168260310)

[11.1 Vidéo de présentation 25](#_Toc168260311)

[11.2 Dossier contenant les scripts et programmes de construction du lac de données 25](#_Toc168260312)

[11.3 Dossier contenant les scripts et programmes Hadoop Map Reduce 25](#_Toc168260313)

[11.4 Dossier contenant les scripts et programmes d’analyse de données 25](#_Toc168260314)

# Introduction générale

" Vous cherchez à optimiser les ventes de votre concession automobile et à identifier de nouvelles opportunités de croissance ? Alors notre rapport analytique est votre meilleur allié !"

Ce projet vise à appliquer notre maîtrise des techniques du Big Data, du concept Map Reduce, de l’analyse des données et machine learning pour répondre au mieux les besoins des clients du concessionnaire automobile.

Nous aurons alors pour mission de gérer les données offertes par le concessionnaire avec l’architecture Big Data Analytics/Data Lake en les centralisant et en utilisant le concept de Map Reduce avec Hadoop pour effectuer les tâches complexes, et d’effectuer l’analyse des données en préparant les données en entrées des outils d’analyses sur Python avec SQL.

Le plan de notre rapport est structuré comme suit :

* Présentation du projet : en quoi est-il innovant et quelles sont les défis
* Répartition des tâches : présentation de la distribution des tâches entre les membres du groupe
* Architecture du Data Lake : construction et description du Data Lake
* Construction du Data Lake par étape : détails depuis la mise en place des sources de données jusqu’à l’exploitation des données dans les outils d’analyses
* Hadoop Map Reduce : explication de l’utilité et utilisation du processus
* Analyse de données avec Python : clustering des données, entrainement et prédiction
* Conclusion : bilan du projet en général et évaluation par rapport aux objectif fixés

# Présentation du projet

Dans un secteur automobile en constante évolution, où les attentes des clients sont de plus en plus exigeantes, l'analyse de données émerge comme un outil indispensable pour les concessionnaires. Notre projet se distingue par son approche novatrice qui combine les dernières avancées en Big Data et en Data Analytics & Machine Learning pour répondre aux besoins spécifiques de nos clients concessionnaires. En développant des solutions sur mesure, nous ouvrons la voie à une transformation significative de leur approche commerciale, les positionnant à la pointe de l'industrie.

Dans un marché automobile compétitif, répondre aux besoins des clients rapidement et de manière personnalisée est essentiel pour les concessionnaires. Notre projet se situe à un moment clé où l'utilisation intelligente des données devient essentielle pour prendre des décisions stratégiques. Cela offre aux concessionnaires une chance de se démarquer et de réussir dans un environnement en constante évolution.

La réussite de ce projet est d’une importance capitale pour nos clients concessionnaires, car elle détermine leur capacité à rester pertinents et compétitifs dans un marché en évolution rapide. Les principaux enjeux résident dans la capacité à traiter efficacement de grandes quantités de données tout en assurant leur qualité et leur pertinence.

De plus, la mise en œuvre comporte des risques, notamment en termes de complexité technique.

Tout d'abord, l'accès aux données à partir de différentes sources est un défi majeur, nécessitant l'utilisation de Access Driver et Data Extractor pour l'extraction, le chargement et la transformation des données (ELT).  
Ensuite, la centralisation de ces données dans le Data Lake, avec HiveQL comme frontal de données, est essentielle pour assurer leur accessibilité et leur intégrité.  
Une fois les données centralisées, le traitement massif avec Hadoop MapReduce garantit leur nettoyage et leur préparation pour l'analyse.  
Enfin, l'utilisation de Python pour l'analyse avancée et la prédiction marque la dernière étape de ce processus complexe.

Malgré ces défis, une approche méthodique et une répartition efficace des tâches sont essentielles pour garantir le succès de ce projet et la satisfaction de nos clients concessionnaires.

# Répartition du travail en membre du groupe

Voici les membres du groupe :

**13** - RAFAMANTANANTSOA Rotsy Vonimanitra

**29** – RAMAROSON Sandy Princia

**31** – RAMIANDRISOA Rantonantenaina Steve

**50** – RATSIRARSON Joharisoa

Nous allons voir ci-dessous la répartition du projet entre les membres du groupe :

* Architecture du data lake : Tous les membres
* Mise en place du data lake :
  + Utilitaires et sources de données :
    - HDFS (Catalogues, CO2) : **29**
    - Programme Python - Oracle NoSQL (Marketing) : **31**
    - Mongo import - MongoDB (Clients, Immatriculations) : **13**
  + Data Access Drivers :
    - HDFS Access Driver (Catalogues) : **29**
    - Oracle NoSQL Access Driver (Marketing) : **31**
  + Data Extractor - ELT :
    - NiFi (Clients, Immatriculations) : **13** - **29**
  + Frontal de données HiveQL Data Lake :
    - Données virtuelles (Catalogues, Marketing) : **29** - **31**
    - Données physiques (Clients, Immatriculations) : **13**
    - Données de modèles d’analyses (vue clients\_immatriculations) : **13**
  + Hadoop Map Reduce (Catalogues, CO2) : **31** - **50**
  + Utilitaire et outils d’analyse :
    - Clustering : **50** -**31**
    - Classifieur : **50**
    - Modèle de prediction : **50**
  + Result database : **50**
* Rédaction du TPA : **13**
* Vidéo de présentation du projet : Tous les membres

# Architecture du Data Lake

Notre architecture de Data Lake intègre plusieurs composants essentiels et utilise Hive Data Lake comme frontal de données.

Voici une description détaillée de chaque composant :

1. **Fichiers de Données**

* CO2.csv : Ce fichier continent les données sur les émissions de CO2
* Immatriculations.csv : Ce fichier contient les informations sur les immatriculations effectuées cette année
* Catalogue.csv : Ce fichier contient une liste des catalogues de véhicules disponibles chez le concessionnaire
* Clients\_3.csv et Clients\_16.csv : Ces fichiers contiennent les achats effectués par les clients numéros 3 et 16 cette année
* Marketing.csv : Ce fichier contient des groupes de clients sélectionnés par le service marketing

1. **Utilitaires**

* Utilitaire HDFS : Les fichiers CO2.csv et Catalogues.csv sont importés dans le système de fichiers distribué Hadoop (HDFS) à l'aide d'un utilitaire HDFS.
* Mongo import : Les fichiers Clients\_3.csv, Clients\_16.csv et Immatriculations.csv sont importés dans MongoDB via l’utilitaire mongo import
* Python Program Extractor : Le fichier marketing.csv est importé dans Oracle NoSQL via un programme d’extraction en Python

1. **Sources de données**

* HDFS : Stocke les données de CO2.csv, Catalogue.csv et Catalogue\_CO2.csv (explication ci-dessous dans la partie Hadoop Map Reduce).
* Oracle NoSQL : Stocke les données de Marketing.csv
* MongoDB : Stocke les données des fichiers Clients\_3.csv, Clients\_16.csv et Immatriculations.csv

1. **Hadoop Map Reduce**

* HDFS : Ecriture d’un programme map reduce avec Hadoop pour adapter le fichier CO2.csv et ajouter des informations aux données de la table Catalogue. On obtient une nouvelle source combinant Catalogue et CO2 qu’on a appelé Catalogue\_CO2 et qu’on a remis dans HDFS

1. **Alimentation du Data Lake**

* HDFS Acces driver : Pour l’accès direct aux données depuis la source Catalogue\_CO2
* Oracle NoSQL Access driver : Pour l’accès direct aux données depuis la source Marketing
* NiFi (Extract Load and Transform) : Pour l’alimentation des données physiques du Data Lake, ces données sont les données sur les Clients et les Immatriculations

1. **Les données du Data Lake HiveQL**

* Données virtuelles : Contient la table externe HDFS dont Catalogue\_CO2 et la table externe Oracle NoSQL Marketing. Les données sont accessibles via access drivers.
* Données physiques : Contient les managed tables (tables internes) comme Clients et Immatriculations. Les données sont chargées avec un extracteur de données (NiFi)
* Données des modèles d’analyses : Données qui font une jointure entre les clients et immatriculations sous forme de vue : clients\_immatriculations

1. **Outils d’analyse**

* PyHive : utilisé pour connecter les modèles d’analyses du Data Lake HiveQL à des outils d’analyse Python
* Python : Utilisé pour l’analyse des données.

1. **Base des Résultats**

* MongoDB : Les résultats de l’analyse des données sont visualisés à l’aide de Python et les données résultats des prédictions sont stockés dans une nouvelle table MongoDB

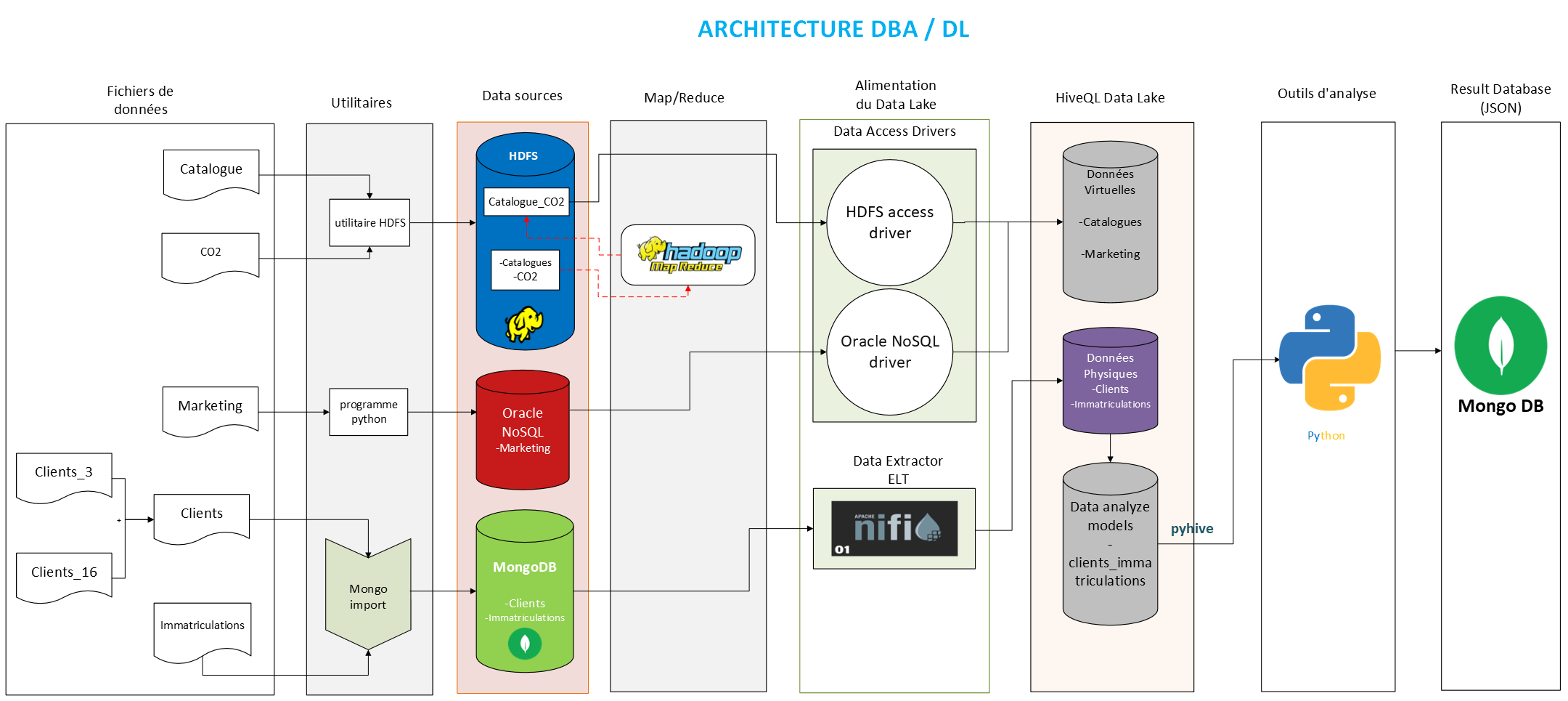


Figure 1 : Architecture DBA/DL

# Construction du data lake par étape

Nous présenterons ici les étapes de la mise en œuvre de l’architecture du data lake de la collecte des données jusqu’à leur exploitation dans les outils d’analyse avec Python.

L’ensemble des scripts et étapes montrés ci-dessous sont exécutables par une simple exécution du fichier **script-tpa.ksh.**

## Démarrage de la machine virtuelle et configuration

* Lancer l’invite de commande

cmd

* Naviguer vers le dossier où se trouve votre machine Vagrant

cd lienVersVotreMachineVirtuelle\INSTALL\_MV\_BIGDATA\_BOX>

* Arrêter si nécessaire puis démarrer la machine

vagrant halt

vagrant up

* Connectez-vous à la machine Vagrant

vagrant ssh

* Démarrer Hadoop HDFS

if ! jps | grep -q 'NameNode'; then

    start-dfs.sh

else

    echo "HDFS is already running."

fi

* Démarrer Hadoop YARN

if ! jps | grep -q 'ResourceManager'; then

    start-yarn.sh

else

    echo "YARN is already running."

fi

* Installer la librairie ‘pandas’, utile pour les programmes Python

if ! pip show pandas > /dev/null; then

    pip install pandas

else

    echo "pandas is already installed."

fi

* Démarrer le serveur Hadoop Hive

nohup hive --service metastore > /dev/null &

nohup hiveserver2 > /dev/null &

* *Copier* le dossier *Groupe\_TPA\_4* a la racine de votre machine vagrant

## Mise en place des sources de données

### Source Hadoop Distributed File System (HDFS)

* Fichiers : Catalogue.csv, CO2.csv
* Procédure :
  + Définition du répertoire de travail

SCRIPT\_DIR="/vagrant/Groupe\_TPA\_4/mapreduceCo2"

cd ${SCRIPT\_DIR}

* + Nettoyage des données de CO2 avec un programme Python

python3 cleanCO2.py

* + Importation des données de CO2 et Catalogue dans HDFS

hadoop fs -put modified\_CO2.csv

hadoop fs -put Catalogue.csv

* + Importation des données de Catalogue\_CO2, résultat du map reduce sur CO2, dans HDFS (Détails dans l’étape [6.Hadoop Map Reduce](#_Hadoop_Map_Reduce))

hadoop fs -mkdir -p /catalogueCO2

hadoop fs -put Catalogue\_CO2.csv /catalogueCO2

* + Créer des dossiers pour stocker les fichiers temporaires sur les clients et immatriculations. Ces fichiers seront utiles pour l’alimentation des tables internes de Hive depuis NiFi.

hdfs dfs -mkdir /clients

hdfs dfs -mkdir /immatriculations

### Source Oracle NoSQL

* Fichier : Marketing.csv
* Procédure :
  + Démarrage du serveur Oracle NoSQL s’il n’est pas encore démarré

if ! jps | grep -q 'kvstore'; then

    nohup java -Xmx64m -Xms64m -jar $KVHOME/lib/kvstore.jar kvlite -secure-config disable -root $KVROOT &

else

    echo "Oracle NoSQL server is already running."

fi

* + Définition du répertoire de travail

SCRIPT\_DIR="/vagrant/Groupe\_TPA\_4"

cd ${SCRIPT\_DIR}

* + Création d’un fichier sql pour l’insertion des données avec un programme Python > génération d’un fichier scriptInsert.sql

python3 ${SCRIPT\_DIR}/scriptInsertMarketing/scriptInsertMarketing.py

* + Lancement du client SQL de Oracle NoSQL

java -jar $KVHOME/lib/sql.jar -helper-hosts localhost:5000 -store kvstore <<EOF

* + Création de la table marketing et insertion des données dans Oracle NoSQL

load -file ${SCRIPT\_DIR}/scriptInsertMarketing/scriptInsert.sql

exit

EOF

### Source MongoDB

* Fichiers : Clients\_3.csv, Clients\_16.csv, Immatriculations.csv
* Procédure :
  + Définition du répertoire de travail

SCRIPT\_DIR="/vagrant/Groupe\_TPA\_4"

cd ${SCRIPT\_DIR}

* + Création des dossiers clients et immatriculations ou on traitera les fichiers csv

mkdir clients

mkdir immatriculations

* + Fusion des fichiers Clients\_3.csv et Clients\_16.csv (traitement avec awk)

cat M2\_DMA\_Clients\_4/Clients\_3.csv <(tail -n +2 M2\_DMA\_Clients\_17/Clients\_16.csv) > clients/ClientsWithoutId.csv

* + Nettoyage des données clients.csv (traitement avec awk)

cd clients

# modifier le fichier csv pour y mettre une colonne id

awk -F, 'BEGIN {OFS=FS; i="id"} {print i, $0; i++}' ClientsWithoutId.csv > ClientsToManage.csv

# renommer la colonne 2eme voiture en deuxiemeVoiture

awk 'BEGIN {FS=OFS=","} FNR==1 {$7="deuxiemeVoiture"} {print}'  ClientsToManage.csv > temp.csv && mv temp.csv ClientsToManage.csv

# mettre les colonnes numeriques avec valeurs non numeriques en 0 par defaut

awk -F',' 'BEGIN {OFS=FS} NR>1 {for (i=1; i<=NF; i++) {if (i==2 || i==4 || i==6) {if ($i=="" || $i=="?" || $i !~ /^[0-9]+$/) $i="0"}}}1' ClientsToManage.csv > temp.csv && mv temp.csv ClientsToManage.csv

#uniformiser la colonne sexe, M pour les valeurs commençant par M et les valeurs non définis (N/D, ? , "",...) et F pour les autres

awk -F',' 'BEGIN {OFS=FS} NR>1 {if ($3 ~ /^F/) $3="F"; else if ($3 ~ /^M/ || $3 == "Homme") $3="M"; else $3="NULL"} 1' ClientsToManage.csv > temp.csv && mv temp.csv ClientsToManage.csv

# mettre la colonne deuxiemeVoiture en false par defaut

awk -F',' 'BEGIN {OFS=FS} NR>1 {if ($7 != "true" && $7 != "false") $7="false"} 1' ClientsToManage.csv > Clients.csv

* + Nettoyage des données immatriculations.csv (traitement avec awk)

cd ${SCRIPT\_DIR}

# modifier le fichier csv pour y mettre une colonne id

awk -F, 'BEGIN {OFS=FS; i="id"} {print i, $0; i++}' M2\_DMA\_Immatriculations/Immatriculations.csv > immatriculations/ImmatriculationsToManage.csv

cd immatriculations

# mettre les colonnes numeriques avec valeurs non numeriques en 0 par defaut

awk -F',' 'BEGIN {OFS=FS} NR>1 {for (i=1; i<=NF; i++) {if (i==5 || i==7 || i==8 || i==11) {if ($i=="" || $i=="?" || $i !~ /^[0-9]+$/) $i="0"}}}1' ImmatriculationsToManage.csv > temp.csv && mv temp.csv ImmatriculationsToManage.csv

# mettre la colonne occasion en false par defaut

awk -F',' 'BEGIN {OFS=FS} NR>1 {if ($10 != "true" && $10 != "false") $10="false"} 1' ImmatriculationsToManage.csv > Immatriculations.csv

* + Création de la table clients et importation des données dans MongoDB

cd ${SCRIPT\_DIR}/clients

mongoimport --db Concessionaire --collection clients --type csv --file Clients.csv --headerline

* + Création de la table immatriculations et importation des données dans MongoDB

cd ${SCRIPT\_DIR}/immatriculations

mongoimport --db Concessionaire --collection immatriculations --type csv --file Immatriculations.csv --headerline

## Mise en place du frontal de données sur HiveQL Data Lake

### Création des données virtuelles Hive

* + - Table externe CATALOGUE\_CO2\_HDFS\_EXT

beeline -u jdbc:hive2://localhost:10000 -n oracle -p welcome1 -e "

DROP TABLE IF EXISTS CATALOGUE\_CO2\_HDFS\_EXT;

CREATE EXTERNAL TABLE CATALOGUE\_CO2\_HDFS\_EXT (

marque STRING, nom STRING, puissance INT, longueur STRING, nbPlaces INT,

nbPortes INT,

couleur STRING,

occasion boolean,

prix INT,

BonusMalus INT,

RejetsCO2 INT,

CoutEnergie INT)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE LOCATION '/catalogueCO2';

SELECT COUNT(\*) FROM CATALOGUE\_CO2\_HDFS\_EXT;

"

* + - Table externe MARKETING\_ONS\_EXT

beeline -u jdbc:hive2://localhost:10000 -n oracle -p welcome1 -e "

DROP TABLE IF EXISTS Marketing\_ONS\_EXT;

CREATE EXTERNAL TABLE Marketing\_ONS\_EXT  (

id int,

age int,

sexe string,

situationFamiliale string,

nbEnfantsAcharge int,

deuxieme\_voiture boolean

)

STORED BY 'oracle.kv.hadoop.hive.table.TableStorageHandler'

TBLPROPERTIES (

'oracle.kv.kvstore' = 'kvstore',

'oracle.kv.hosts' = 'localhost:5000',

'oracle.kv.hadoop.hosts' = 'localhost/127.0.0.1',

'oracle.kv.tableName' = 'marketing');

"

### Création des données physiques Hive

* + - Table interne CLIENTS dans la database CONCESSIONAIRE

beeline -u jdbc:hive2://localhost:10000 -n oracle -p welcome1 -e "

DROP TABLE IF EXISTS CONCESSIONAIRE.CLIENTS;

CREATE TABLE CONCESSIONAIRE.CLIENTS (

ID int,

AGE int,

SEXE string,

TAUX float,

SITUATIONFAMILIALE string,

NBENFANTSACHARGE int,

DEUXIEMEVOITURE boolean,

IMMATRICULATION string

)

ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe';

"

* + - Table interne IMMATRICULATIONS dans la database CONCESSIONAIRE

beeline -u jdbc:hive2://localhost:10000 -n oracle -p welcome1 -e "

DROP TABLE IF EXISTS CONCESSIONAIRE.IMMATRICULATIONS;

CREATE TABLE CONCESSIONAIRE.IMMATRICULATIONS (

ID int,

IMMATRICULATION string,

MARQUE string,

NOM string,

PUISSANCE int,

LONGUEUR string,

NBPLACES int,

NBPORTES int,

COULEUR string,

OCCASION boolean,

PRIX int

)

ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe';

"

* + - Alimentation des tables internes via Apache NiFi (ELT)

***Apache NiFi*** est une plateforme open-source permettant la gestion et l'automatisation des flux de données en temps réel. Son rôle en tant qu'ELT (Extract, Load, Transform) consiste à extraire des données depuis différentes sources, les charger dans un système de stockage, puis les transformer selon les besoins.

Dans notre architecture, il est utilisé pour charger les données des clients et immatriculations depuis MongoDB et les insérer dans les tables internes de Hive.

Il y a eu des contraintes sur les accès aux serveurs MongoDB et Hadoop Hive depuis la plateforme NiFi. Nous avons alors décidé de l’installer directement dans la machine virtuelle.  
NiFi utilise une interface graphique accessible via navigateur web pour gérer les processus et il fallait faire un port forwarding[[1]](#footnote-1)pour pouvoir y accéder. La plateforme est alors accessible via l’url <http://127.0.0.1:8444/nifi> (8443 vers 8444).

Voici les étapes suivis pour **l’installation du logiciel Apache NiFi sur votre machine virtuelle**, il n’y aura ensuite qu’à lancer le processus avec le bouton « Start » et l’alimentation des données physiques sur Hive se fera toute seule.

* Copier le fichier d’installation de NiFi depuis le lien drive suivant et collez-le a la racine de votre machine Vagrant : <https://drive.google.com/file/d/1OGgrQQ9FJ7FVWp1Out6PPdOO5hszGOi9/view?usp=drive_link>
* Dézipper le fichier, vous devez avoir une architecture de fichier comme ceci : /vagrant/nifi-1.19.1
* Activer l’accessibilité du port 8443, utilisé par défaut par NiFi. Dans vagrant ssh, exécuter :

sudo su

iptables -A INPUT -p tcp --dport 8443 -j ACCEPT

* Vous pouvez vérifier l’état du port en le voyant dans les résultats de la commande suivante :

iptables -L

* Créer une configuration d'alias personnalisée pour votre profil de machine virtuelle Vagrant.

Sous Windows, rendez-vous dans le dossier Your-username/.ssh et recherchez le fichier *config*, sinon créez le (sans extension).

Ajoutez les lignes suivantes pour créer l'alias oracle-21c-vagrant

Host oracle-21c-vagrant

    HostName 127.0.0.1

    User vagrant

    Port 2222

    UserKnownHostsFile /dev/null

    StrictHostKeyChecking no

    PasswordAuthentication no

    IdentityFile Your-username/.vagrant.d/boxes/package.box/0/virtualbox/vagrant\_private\_key

    IdentitiesOnly yes

    LogLevel FATAL

    PubkeyAcceptedKeyTypes +ssh-rsa

    HostKeyAlgorithms +ssh-rsa

* Utilisez le port forwarding pour accéder à NiFi à partir de l'extérieur

ssh -L 8444:127.0.0.1:8443 oracle-21c-vagrant

* Vous pouvez maintenant démarrer NiFi

cd /vagrant/nifi-1.19.1

./bin/nifi.sh start

* Attendez le démarrage de NiFi, vous pouvez vérifier les logs en attendant qu’une ligne affiche <http://127.0.0.1:8444/nifi>. Le démarrage peut être assez long selon votre machine (5 à 10 minutes).

tail -f -n 100 ./logs/nifi-app.log

* Une fois le logiciel démarré et l’interface s’affichant sur le navigateur, faite un clic droit sur votre souris et cliquez sur le bouton « Start ».
* Vérifier que les données ont bien été importés dans Hive

beeline -u jdbc:hive2://localhost:10000 -n oracle -p welcome1 -e "

select \* from CONCESSIONAIRE.CLIENTS;

select \* from CONCESSIONAIRE.IMMATRICULATIONS;

"

**Architecture du processus sur NiFi :**

* GetMongo : Accède aux données des collections MongoDB et les charges dans le Process Group

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 2 : Résultat du processor GetMongo

* Les 2 premiers ReplaceText : Formattent les json reçues depuis GetMongo en une ligne.



Figure 3 : Résultat des processors ReplaceText

* PutHDFS : Met chaque groupe de données dans un fichier HDFS temporaire. Un fichier HDFS correspond à une liste de json qui va être passé à un script d’insertion dans Hive.

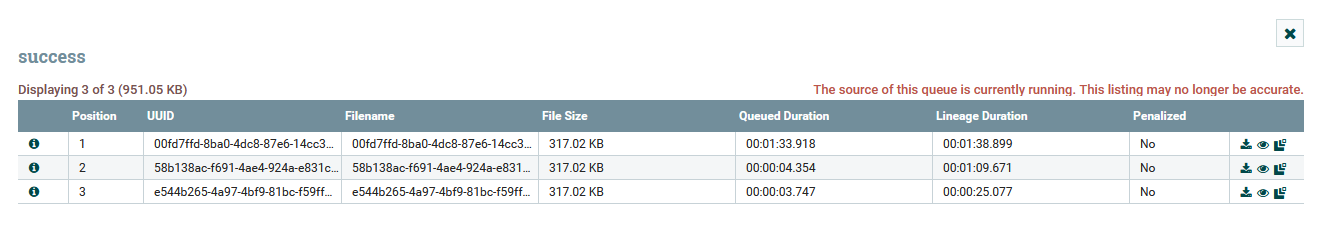


Figure 4 : Liste des fichiers temporaires HDFS

* ReplaceText : écrit le script d'insertion des fichiers temporaires HDFS dans la table Hive



Figure 5 : Résultat du processor ReplaceText

* PutHiveQL : exécute les scripts d’alimentation des tables internes clients et immatriculations dans Hive.

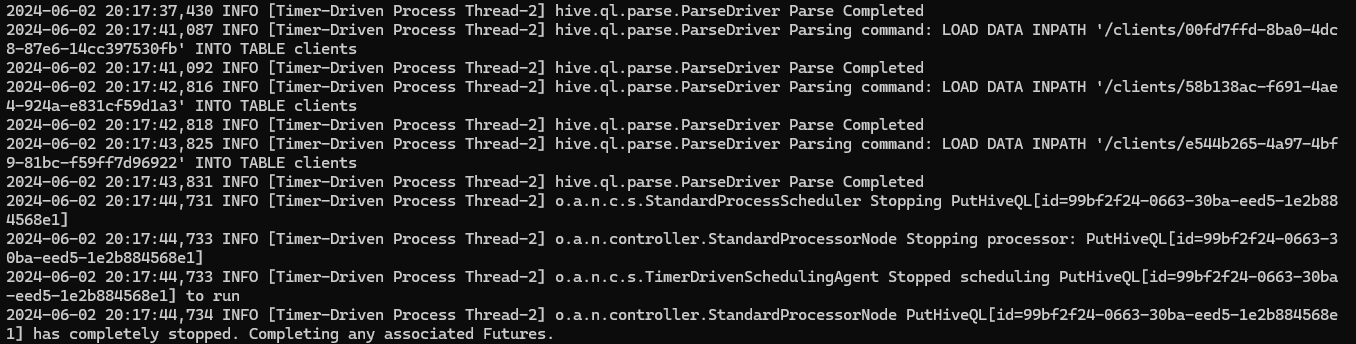


Figure 6 : Log d'insertion des données dans Hive

* LogMessage : affiche les messages d'erreurs et de succès dans le fichier log de NiFi pour le suivi des process.

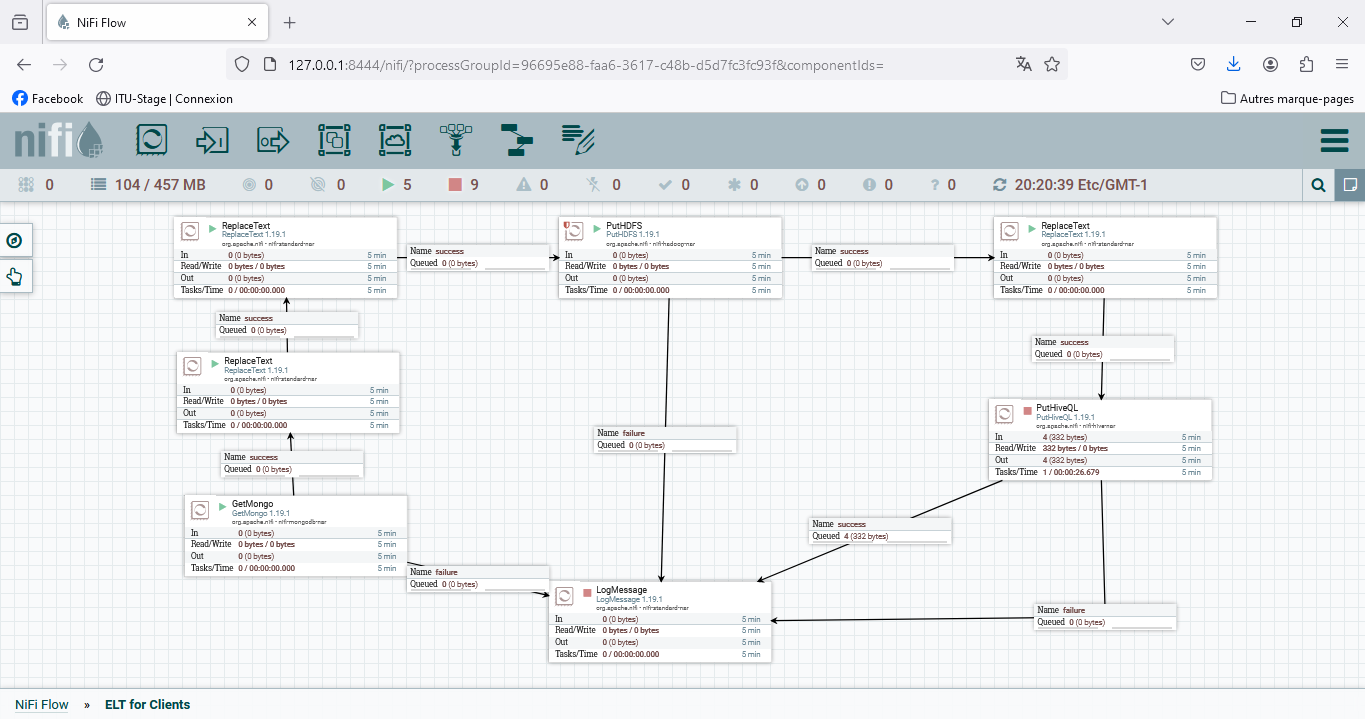


Figure 7 : Architecture du processus d'alimentation avec NiFi

### Création des données de modèle d’analyse

* + - Table de représentation de vue SQL CLIENTS\_IMMATRICULATIONS dans la database CONCESSIONAIRE

beeline -u jdbc:hive2://localhost:10000 -n oracle -p welcome1 -e "

CREATE VIEW CONCESSIONAIRE.CLIENTS\_IMMATRICULATIONS AS

select c.id,c.age, c.sexe, c.taux, c.situationfamiliale, c.nbenfantsacharge,c.deuxiemevoiture, c.immatriculation,i.marque, i.nom,i.puissance,  i.longueur, i.nbplaces, i.nbportes, i.couleur,i.occasion, i.prix

from concessionaire.clients c join concessionaire.immatriculations i on i.immatriculation = c.immatriculation;

"

## Exploitation des données sur Python

* 1. Installation PyHive

# Hadoop Map Reduce

On décrit ci-dessous les étapes de la mise en œuvre de l’activité autour de Hadoop Map Reduce :

# Analyse de données avec Python

1. **Analyse exploratoire des données**

Lors de cette phase, nous avons chargé les données de modèle d’analyse depuis le frontal hive. On y a créé une vue fusionnant les données Clients et Immatriculations, nommée ‘clients\_immatriculations’ afin d’obtenir une matrice dans laquelle chaque ligne correspondra à un client avec les caractéristiques du véhicule qu’il a acquis.

1. **Identification des catégories de véhicules**

On a fait le clustering des données dans cette étape et on a alors catégoriser les véhicules en **k** catégories :

1. **Application des catégories de véhicules définies aux données des Immatriculation**
2. **Construction d’un modèle de prédiction ou classifieur**
3. **Application du modèle de prédiction aux données Marketing**

# Conclusion générale

Au terme de ce projet d'analyse de données pour les concessionnaires automobiles, nous dressons un bilan positif des résultats obtenus pour l'entreprise. Grâce à notre solution, les concessionnaires disposent désormais d'un outil pour mieux cibler leurs clients et proposer des véhicules adaptés à leurs besoins, renforçant ainsi leur compétitivité sur le marché.

Cependant, nous avons rencontré des défis majeurs tout au long du projet, notamment en ce qui concerne l'accès à MongoDB et Hive depuis l'extracteur de données Nifi. La solution consistant à installer ces composants dans la machine virtuelle a permis de surmonter ces obstacles et de garantir le bon fonctionnement de notre système. […]

Pour l'avenir du projet, nous envisageons d'explorer de nouvelles possibilités d'amélioration, telles que l'optimisation des processus d'analyse de données et l'intégration de nouvelles sources de données pour enrichir notre modèle.

En ce qui concerne le bilan personnel, ce projet a été une expérience enrichissante qui nous a permis de mettre en œuvre les théories et les cours sur le Big Data, le MapReduce et l'analyse de données et de machine learning dans un contexte concret. Nous sommes fiers d'avoir contribué au succès de ce projet et nous sommes reconnaissants pour les enseignements et les compétences que nous avons acquis tout au long de ce parcours.

# Références et Bibliographie

* Bibliographie

[1] Nicolas Pasquier, Apprentissage d'Arbres de Décision, [UCApod - Cm03dmml010.Mp4 (univ-cotedazur.fr)](https://pod.univ-cotedazur.fr/video/13068-cm03dmml010mp4/)

[2] Nicolas Pasquier, Cours Magistral, [UCApod - Cm06dmml010.Mp4 (univ-cotedazur.fr)](https://pod.univ-cotedazur.fr/video/13076-cm06dmml010mp4/)

* Sites internets:

# Annexes

## 11.1 Vidéo de présentation

## 11.2 Dossier contenant les scripts et programmes de construction du lac de données

## 11.3 Dossier contenant les scripts et programmes Hadoop Map Reduce

## 11.4 Dossier contenant les scripts et programmes d’analyse de données

1. Le port forwarding est une technique qui permet de rediriger le trafic réseau entrant d'un port spécifique vers une machine ou un service différent. Ici, on a change le redirigé le trafic du port 8443 vers 8444. [↑](#footnote-ref-1)