PMN DAT 25.1
Mohamed Mustapha Abdennadher

# Projet de Modélisation de Données avec PySpark et Hive

## 1. Introduction

### 1.1 Objectifs du projet

Le projet a pour objectif de construire un entrepôt de données à partir de plusieurs sources brutes (clients, vélos, commandes et magasins) et de le rendre exploitable pour des analyses décisionnelles à l'aide de Power BI. L'entrepôt de données est implémenté en utilisant PySpark pour le nettoyage et la transformation des données, ainsi que Hive pour leur stockage et l'interrogation à l'aide d'Impala.

Toutes les étapes exécutées (excluant celles du debugging) sont dans le fichiers README

### 1.2 Contexte

Dans le cadre de la gestion des données d'une entreprise vendant des vélos, plusieurs fichiers de données au format CSV sont disponibles pour représenter différentes entités : clients, vélos, magasins, et commandes. Ces données nécessitent un nettoyage approfondi, une transformation et une modélisation sous forme de tables de faits et de dimensions pour faciliter les analyses futures.

## 2. Choix des technologies

**La principale raison du choix de ces technologies pour ce projet est étroitement liée à leur pertinence sur le marché actuel. La deuxième raison est l'opportunité d'apprendre et d'utiliser les concepts et outils les plus récents, découverts au cours de cette formation.**

### 2.1 PySpark

Nous avons choisi **PySpark** comme technologie principale pour le nettoyage et la transformation des données en raison de plusieurs avantages :

- **Scalabilité :** moteur de traitement de données distribué très performant, capable de traiter de grands ensembles de données en parallèle.
- **Compatibilité avec Hadoop et HDFS**
- **Flexibilité :** combine la simplicité du langage Python et les performances de Spark.
- **Support natif des formats distribués :** supporte nativement les formats optimisés pour le stockage distribué, tels que Parquet, maximisant l'efficacité lors de la gestion des grands ensembles de données.

### 2.2 Hive et Impala

Hive a été choisi pour son intégration avec l'écosystème Hadoop, et Impala permet d'interroger les données à faible latence :

- **Hive :** Hive fournit une interface SQL familière pour les analystes tout en permettant de gérer des données massives. Il permet de définir des schémas de données sur des fichiers stockés dans HDFS, comme les fichiers Parquet utilisés ici.
- **Impala :** Pour rendre les données accessibles en temps réel et rapidement interrogables par Power BI. Il permet une interactivité proche des bases de données traditionnelles mais sur des données massivement parallèles stockées dans HDFS.

### 2.3 Power BI

**Power BI** permet de visualiser des données de manière interactive et de créer des rapports dynamiques. C'est la modélisation et sa connexion avec Impala qui sont des nouvelles compétences que je souhaitais apprendre et implémenter.

**3. Parquet**

**3.1 Avantages du format Parquet**

Le choix du format **Parquet** pour le stockage des données est stratégique pour plusieurs raisons :

- **Format Colonnaire :** Parquet est un format de stockage colonnaire, ce qui signifie que les données sont stockées par colonne plutôt que par ligne. Cela le rend très efficace pour les opérations analytiques, où seules certaines colonnes sont nécessaires pour une requête.
- **Compression et stockage efficace :** Parquet permet de réduire considérablement l'espace disque occupé et sa compression au niveau des colonnes est plus efficace que sur des fichiers ligne à ligne.
- **Schéma et typage des données :** Parquet conserve le schéma et les types de données, ce qui permet de garantir l'intégrité et la cohérence des données au moment de la lecture et du traitement.
- **Compatibilité avec les systèmes distribués**

**3.2 Comparaison avec CSV**

- **Absence de typage :** Le format CSV ne supporte pas nativement les types de données, ce qui oblige à réinterpréter les données à chaque lecture, ajoutant une surcharge.
- **Manque d'efficacité pour les grandes données :** En raison de sa structure ligne à ligne, CSV n'est pas adapté aux opérations analytiques à grande échelle.

---

**4. Architecture de la solution**

**4.1 Schéma conceptuel**

La solution est construite autour d'une architecture en étoile (***star schema***) qui comprend :

- **Table des faits :**
  - fact_orders : contient des informations relatives aux ventes.
- **Tables de dimensions :**
  - customers_lookup : informations des clients.
  - bikes_lookup : détails des vélos.
  - calendar_lookup : table calendrier pour faciliter les analyses temporelles.
  - bikeshops_lookup : données sur les magasins de vélos.

**4.2 Technologies Utilisées**

- **Cloudera Quickstart VM :** Fournit un environnement préconfiguré pour Hadoop, Spark et Hive.

- **PySpark :** Utilisé pour nettoyer, transformer et enrichir les données.
- **Hive :** Stockage des données modélisées sous forme de tables externes en Parquet.
- **Power BI :** Connecté à Impala pour visualiser les données.
- **HDFS :** Stockage des fichiers bruts et des fichiers Parquet.

## 5. MVP/PoC

Dans la première instance de ce projet, nous avons mis en place un **MVP (Minimum Viable Product)** afin de valider les principales fonctionnalités de la plateforme avec les données disponible en csv et j'ai agit en tant qu'utilisateur avec plusieurs itérations.

### 5.1. Réalisation des Objectif du MVP

Plusieurs transformations ont étaient nécessaire pour que la chaîne de traitement des données fonctionne, principalement sur les colonnes de type date, les noms contenant des points que spark ne reconnait pas, ainsi que pour la création des fichiers parquet. Les tables crées permettent d'analyser les comportements d'achat des clients à partir des données de commande grâce aux commandes fournies pour Hive, tout en fournissant des visualisations simples pour les utilisateurs finaux avec Power Bi. Dans notre cas, c'est en redirigeant les ports et en utilisant le connecteur Impala dans Power Bi sur le localhost:50000.

### 5.2. Fonctionnalités du MVP

Le MVP a été conçu avec les fonctionnalités suivantes pour garantir que l'essentiel du pipeline de données fonctionne correctement et apporte une valeur immédiate aux utilisateurs.

1. **Intégration des sources de données**
   Les données brutes issues de fichiers CSV ont été intégrées dans **HDFS**. Nous avons utilisé **PySpark** pour lire, nettoyer et transformer ces données en tables **Parquet**, un format adapté pour les performances et l'optimisation du stockage. Le pipeline couvre l'ensemble du processus, depuis l'importation jusqu'à l'écriture des données transformées dans des répertoires spécifiques pour les clients, les produits, les magasins de vélos et les commandes.
2. **Analyse de base des comportements d'achat**
   Une table de faits (fact_orders) a été créée en se basant sur les commandes, les clients, et les produits. Elle permet d'analyser les comportements d'achat, les tendances des produits, et les volumes de vente par client, par produit et par magasin. Cette table est stockée sous forme de Parquet et est accessible via **Hive** pour l'interrogation SQL.
3. **Tableaux de bord simples avec visualisations clés**
   Le pipeline de données a été intégré avec **Power BI**, permettant ainsi de créer des tableaux de bord interactifs basés sur les données extraites du cluster Hadoop via **Impala**. Ces visualisations comprennent :
   - Le suivi des ventes par mois, par année, et par quarter.
   - Des analyses des performances des différents types frame de vélos.
   - Des visualisations des tendances des clients au fil du temps.

### 5.4. Critères de Succès

Les critères de succès pour ce MVP incluent :

- **Intégration réussie des données sans perte** : Les données provenant des fichiers CSV ont été correctement ingérées dans le système HDFS sans perte, avec des transformations précises pour correspondre aux besoins d'analyse.
- **Production de rapports d'analyse précis** : Grâce aux scripts PySpark et à la configuration de Hive, les rapports produits à partir de la table de faits (fact_orders) fournissent des informations fiables et exploitables sur les comportements d'achat.
- **Réception positive des utilisateurs initiaux** : en tant que développeur et utilisateur final, je confirme la possibilité de créer de analyses à souhaits via le données fournies.

**6. Automatisation du Pipeline avec un Script Shell et Cron**

Afin de garantir l'automatisation du processus de nettoyage et de transformation des données, un script Shell (run_cleaning_pipeline.sh) a été mis en place. Ce script exécute de manière séquentielle tous les fichiers Python responsables du nettoyage et du traitement des différents jeux de données.

**7. Prochaine étape : Intégration d'un Outil de Temps Réel (Real-Time)**

L'outil idéal pour compléter l'architecture actuelle serait **Apache Kafka** en combinaison avec **Apache Spark Streaming**.

- **Apache Kafka :** Kafka permet de gérer des volumes importants de données qui peuvent provenir de diverses sources, telles que des systèmes transactionnels, des capteurs IoT, ou encore des services web.
- **Apache Spark Streaming :** Spark Streaming permet de traiter des flux de données en temps réel. Il est conçu pour s’intégrer facilement avec Kafka, ce qui en fait une solution idéale pour traiter, nettoyer, et transformer les données avant de les stocker dans une base de données ou de les visualiser en temps réel.

**Flux de données en temps réel dans ce projet :**

1. **Ingestion des données :** Kafka serait utilisé pour recevoir les données de nouvelles commandes, clients ou vélos en temps réel. Ces données seraient publiées sous forme de messages dans des "topics".
2. **Traitement :** Spark Streaming récupérerait les messages de Kafka pour les traiter en temps réel. Le traitement inclurait des étapes similaires à celles effectuées sur les données historiques : nettoyage, transformation et enrichissement.
3. **Stockage :** Après traitement, les données seraient stockées dans HDFS en format Parquet, ou insérées directement dans des tables Hive, prêtes à être interrogées.
4. **Visualisation :** Pour la visualisation en temps réel, **Power BI** peut être configuré pour se connecter à Impala, qui permettrait d’interroger les nouvelles données fraîchement intégrées.

L'intégration de ces outils permettrait à l'entreprise d'analyser à la fois des données historiques et des flux de données en temps réel, offrant ainsi une vue globale des opérations tout en permettant de réagir rapidement aux événements récents.