Livrable Partie 2 : Ingestion et Transformation des Données

Code Spark pour nettoyer et transformer les données

Le code Apache Spark suivant a été développé pour effectuer le nettoyage et la transformation des données. Ce code traite les fichiers statiques ainsi que le flux publicitaire en temps réel. Le pipeline se divise en deux parties : le traitement en batch pour les données statiques et le traitement en temps réel pour les flux Kafka.

1. Nettoyage des données statiques :
  - Suppression des lignes invalides (par exemple, les transactions sans `transaction_id` ou montant).
  - Transformation des types de colonnes pour les rendre compatibles avec l'analyse (par exemple, conversion des timestamps en format `datetime`).
  - Agrégation des données par `customer_id` pour calculer les dépenses totales par client.

2. Transformation en temps réel :
  - Calcul du **CTR (Click-Through Rate)** sur les flux Kafka. Le CTR est calculé à partir des données d'impressions et de clics dans les flux publicitaires en temps réel.
  - Agrégation des données publicitaires pour le calcul des taux de clics par campagne.

Le code complet de l'implémentation est contenu dans les fichiers Python joints pour chaque partie du pipeline de transformation (batch et streaming).

Documentation sur les différentes transformations appliquées

Voici un résumé des principales transformations appliquées aux différentes sources de données lors de leur traitement dans Spark :

1. Transactions Clients :
  - **Validation des colonnes obligatoires** : Vérification que chaque transaction contient des colonnes critiques comme `transaction_id` et `amount`.
  - **Agrégation par `customer_id`** : Calcul des dépenses totales par client pour des analyses ultérieures.
  - **Conversion des formats de date** : Transformation des timestamps en format `datetime` pour des analyses temporelles, permettant de grouper les transactions par période (par exemple, par jour ou par mois).

2. Logs Serveurs :
  - **Filtrage des logs** : Identification des logs contenant les messages `ERROR` ou

`WARNING` pour détecter les problèmes de performance ou d'infrastructure.
  - **Extraction des informations clés** : Création de champs supplémentaires pour faciliter l'analyse des performances du système, comme le nombre d'erreurs par période ou par serveur.

3. Données Médias Sociaux :
  - **Tri des publications par nombre de `likes`** : Trier les publications par leur popularité en fonction du nombre de likes.
  - **Sélection des 5 publications les plus populaires** : Identification des posts qui génèrent le plus d'engagement, ce qui peut être utile pour des analyses de tendances ou d'intérêt.

4. Flux Publicitaires :
  - **Calcul du CTR** : Le **Click-Through Rate** est calculé comme suit :
    CTR = (clicks / impressions) * 100
  - **Stockage des données enrichies** : Après avoir calculé le CTR, les données sont stockées dans la **Processed Zone** (HDFS ou système local de stockage).

Pipeline d'ingestion en temps réel et batch

Le pipeline d'ingestion est divisé en deux étapes principales :

1. **Batch** :
  - Les fichiers statiques (CSV, JSON, texte) sont ingérés dans la **Raw Zone** (Stockage local) via **Spark** en mode batch.
  - Ces données sont ensuite traitées pour remplir la couche **Silver** après nettoyage et transformation.

2. **Temps Réel** :
  - Les données publicitaires provenant de **Kafka** sont consommées en temps réel.
  - Le traitement est effectué via **Spark Streaming**, permettant de calculer le CTR en temps réel et de stocker les données transformées dans la **Processed Zone** (Stockage local).

Exemple de datasets ingérés

Voici quelques exemples de données qui sont ingérées dans le **Data Lake** :

1. **Transactions Clients (CSV)** :

transaction_id	customer_id	product_id	amount	timestamp
1	101	301	200.50	2024-11-01T10:00:00
2	102	302	150.75	2024-11-01T11:00:00

| 3 | 103 | 303 | 300.00 | 2024-11-01T12:30:00 |

2. **Logs Serveurs (Text)** :
INFO 2024-11-01T10:00:00 Server started
ERROR 2024-11-01T10:05:00 Connection timeout
WARNING 2024-11-01T10:10:00 High memory usage

3. **Données Médias Sociaux (JSON)** :
{ "post_id": "p1", "user_id": "u1", "likes": 150, "comments": 12, "timestamp": "2024-11-01T12:00:00" }
{ "post_id": "p2", "user_id": "u2", "likes": 200, "comments": 25, "timestamp": "2024-11-01T13:00:00" }
{ "post_id": "p3", "user_id": "u3", "likes": 50, "comments": 5, "timestamp": "2024-11-01T14:00:00" }

4. **Flux Publicitaires (Kafka)** :
{ "ad_id": "201", "campaign_id": "301", "impressions": 1000, "clicks": 100, "timestamp": "2024-11-01T10:00:00" }
{ "ad_id": "202", "campaign_id": "302", "impressions": 1200, "clicks": 150, "timestamp": "2024-11-01T11:00:00" }
{ "ad_id": "203", "campaign_id": "303", "impressions": 800, "clicks": 50, "timestamp": "2024-11-01T12:00:00" }

Exécution du pipeline avec Spark

1. **Ingestion des données dans la Raw Zone (Batch)** :
   spark-submit --master "local[*]" bronze.py

2. **Transformation des données dans la Silver Layer** :
   spark-submit --master "local[*]" silver.py

3. **Calcul du CTR et stockage dans la Processed Zone** :
   spark-submit --master "local[*]" gold.py