epsi
l'école d'ingénierie informatique

TD 1 : performance et sécurité

Exercice 1 : Performance et Efficacité du Stockage

L'objectif de cet exercice est de quantifier l'impact de l'utilisation des formats de stockage de données colonnaires (comme Parquet) par rapport aux formats ligne-orientés traditionnels (comme CSV), tant en termes d'espace disque que de temps de traitement.

1. Préparation du Jeu de Données (Fourni ou à Générer)

1. **Dataset de base :** Télécharger le dataset Kaggle suivant : Flight Delay Dataset — 2024

2. Conversion et Analyse du Stockage

1. **Chargement CSV :** Écrivez un script Python pour charger le fichier CSV dans une structure de données (Pandas DataFrame).
2. **Conversion Parquet :** Convertissez ce DataFrame en un fichier Parquet.
3. **Compression :** Générez une seconde version du fichier Parquet en utilisant une compression efficace (e.g., Snappy ou GZIP).
4. **Analyse d'Espace :** Mesurez la taille des trois fichiers (.csv, .parquet, .parquet.compressed).
    - **Question 1.1 :** Calculez et commentez le taux de réduction de l'espace de stockage du Parquet simple et du Parquet compressé par rapport au CSV.

3. Analyse des Temps de Traitement

1. **Mesure de Lecture Complète :** Chronométrez et comparez le temps nécessaire pour charger **l'intégralité** des trois fichiers (CSV, Parquet, Parquet compressé) en mémoire.
2. **Mesure de Lecture Ciblée (Columnar Advantage) :**
    - Sélectionnez **deux colonnes** uniquement (une de type string et une de type integer).
    - Chronométrez et comparez le temps nécessaire pour charger **uniquement ces deux colonnes** à partir des fichiers Parquet et du fichier CSV.
    - **Question 1.2 :** Expliquez en détail pourquoi le temps de lecture ciblée est significativement plus rapide pour le format Parquet que pour le CSV, en vous basant sur la nature du stockage (ligne vs. colonne).

epsi
l'école d'ingénierie informatique

Exercice 2 : Sécurité et Gouvernance des Données

L'objectif est d'appliquer des techniques de sécurité pour protéger les Informations Personnellement Identifiables (PII) d'un jeu de données.

1. Dataset Sensible

Télécharger le dataset se trouvant ici : clients_data

2. Application des Techniques de Sécurité

Appliquez les transformations suivantes en Python :

1. **Masquage (Masking) :**
   - Remplacez les champs nom et prénom par des valeurs cohérentes mais factices (en utilisant par exemple la librairie *faker*).
   - Masquez les numéros de téléphone
2. **Anonymisation (Généralisation) :**
   - Remplacez le champ ville_résidence par le code **régional** ou **départemental** correspondant (perte délibérée de précision).
3. **Pseudonymisation/Chiffrement (Encryption Simulation) :**
   - Le champ email doit être crypté de manière à pouvoir être déchiffré uniquement par une équipe désignée. Simulez cette opération en hachant la valeur (sha256 ou similaire) et expliquez comment une clé de chiffrement symétrique ou asymétrique serait utilisée en production.
   - Le champ id_client doit être remplacé par un pseudonyme unique.

3. Contrôle d'Accès Basé sur les Rôles (RBAC)

Créez une fonction get_data_by_role(role, dataframe) qui simule l'accès aux données en fonction du rôle de l'utilisateur :

Rôle	Accès aux Colonnes	Niveau de Confidentialité
Analyste_Marketing	id_client (pseudonymisé), montant_achat, ville_résidence (généralisée)	Accès aux données agrégées et anonymisées.
Support_Client_N1	id_client (pseudonymisé), nom (masqué), prénom (masqué), téléphone (masqué), montant_achat	Accès limité aux PII masquées pour identification.
Admin_Sécurité	Toutes les colonnes (y compris les champs chiffrés/hachés)	Accès complet, y compris la colonne email chiffrée.

1. **Question 2.1 :** Montrez l'exécution de cette fonction pour chaque rôle et décrivez comment vous garantiriez, dans un environnement réel (Cloud, Data Warehouse), que ces règles RBAC sont appliquées de manière infaillible.