**Partie 1 : Analyse et critique de la structure**

**1. Justification**

* **Pourquoi cette structure a-t-elle été choisie ?**  
  Elle sépare les entités principales (utilisateurs, produits, commandes) pour assurer modularité, normalisation et clarté.
* **Avantages de séparer les commandes (orders) dans une collection distincte ?**  
  Facilite la gestion des relations utilisateur-commande et permet une scalabilité indépendante.
* **Pourquoi les produits sont-ils référencés par leur productId dans orders ?**  
  Réduit la redondance des données et garantit la synchronisation des informations produits.

**2. Alternatives**

* **Intégrer les commandes dans users : Avantages et inconvénients ?**
  + - Avantages : Accès rapide aux commandes d'un utilisateur.
    - Inconvénients : Complexité croissante avec des données volumineuses et redondance si plusieurs utilisateurs partagent des produits.
* **Détails des produits directement dans les commandes : Avantages et inconvénients ?**
  + - Avantages : Accès direct aux informations produits historiques.
    - Inconvénients : Problèmes de synchronisation si un produit est mis à jour.

**3. Impact sur les performances**

* **Requêtes favorisées** : Retrouver toutes les commandes d’un utilisateur grâce à la référence par userId.
* **Requêtes pénalisées** : Requêtes nécessitant un regroupement de données entre collections, ex. : jointures complexes pour récupérer des détails produits dans les commandes.

**4. Cas pratiques**

* **Suivi de l’historique des modifications de prix des produits :**  
  Ajouter par exemple une collection dédiée à l'historique des prix avec productId, oldPrice, newPrice, et timestamp.
* **Ajout d’un système de remises sur les commandes :**  
  Ajouter par exemple un champ discount dans orders contenant le montant ou le pourcentage de remise.