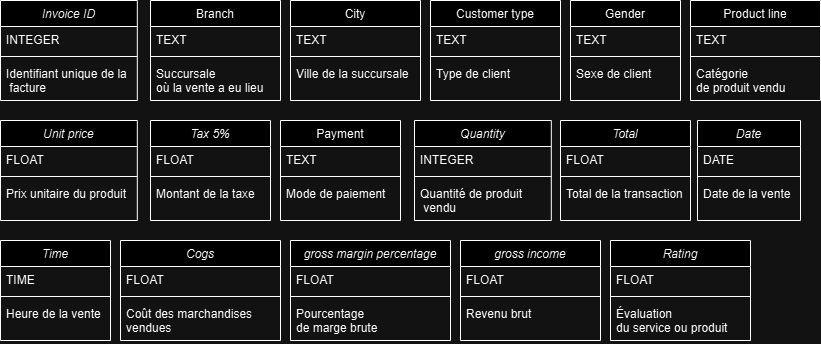
**Atelier 1 Raphael L3 IA**

Tâches réalisée :

Partie 1 : Recherche sur les bases de données relationnelles et NoSQL

**Bases de données relationnelles (SQL)** : **MySQL**

* Le modèle de données structuré (schéma, tables, relations).



* L'utilisation du langage SQL pour manipuler les données.
* La conformité ACID (Atomicité, Cohérence, Isolation, Durabilité).

**Bases de données NoSQL :** **MongoDB**

* Le type de base NoSQL : documents
* La flexibilité du schéma et la scalabilité horizontale.
* La conformité BASE (Basic Availability, Soft state, Eventual consistency).

Partie 2 : Choix d’un cas d'usage

**Choix du cas métier :**

**Boutique en ligne**  Gestion des produits, des clients, des commandes, avec besoin d’analyses transactionnelles et de catalogues produits. Exemple : supermarket\_sales.csv

**Définir les exigences métier :**

Nature des données : **Structurées**

Volume des données : **moyen.**

Partie 3 : Comparaison des bases de données SQL et NoSQL

**1 Analyse des caractéristiques SQL et NoSQL dans le code :**

**SQLite (SQL)** : Ici, la structure rigide des tables est bien illustrée. Les données sont organisées en tables avec des relations et des contraintes ACID. Le code montre des exemples de requêtes SQL avancées, comme les filtrages par date et par type de client, ainsi que des agrégations (moyenne, somme par groupe).

**MongoDB (NoSQL)** : Le code montre comment MongoDB gère des données moins structurées avec une scalabilité horizontale, flexible pour les collections de documents. Le code utilise MongoDB pour insérer des documents et exécuter des requêtes de filtrage, comme celles qui récupèrent les ventes par ville ou qui filtrent selon des conditions spécifiques.

**2 Comparaison des opérations et des types de requêtes :**

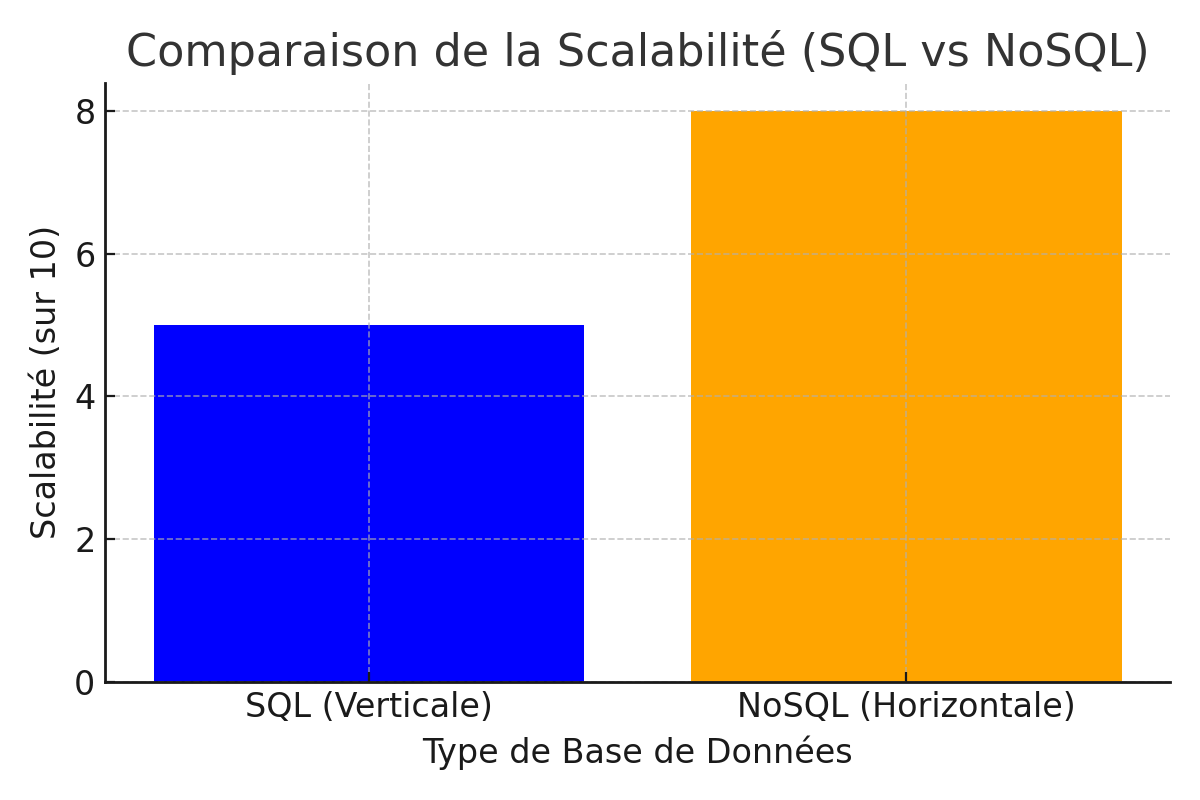
**Structure des données :**

**MongoDB** n’exige pas de structure de données fixe, contrairement à **SQLite** où chaque enregistrement doit respecter le schéma des tables.

**Opérations complexes et scalabilité :**

**SQLite** utilise des jointures et des requêtes SQL complexes pour des analyses détaillées (par exemple, total des ventes par succursale). Ce type de requête est moins performant à grande échelle, car SQLite est limité en termes de scalabilité verticale.

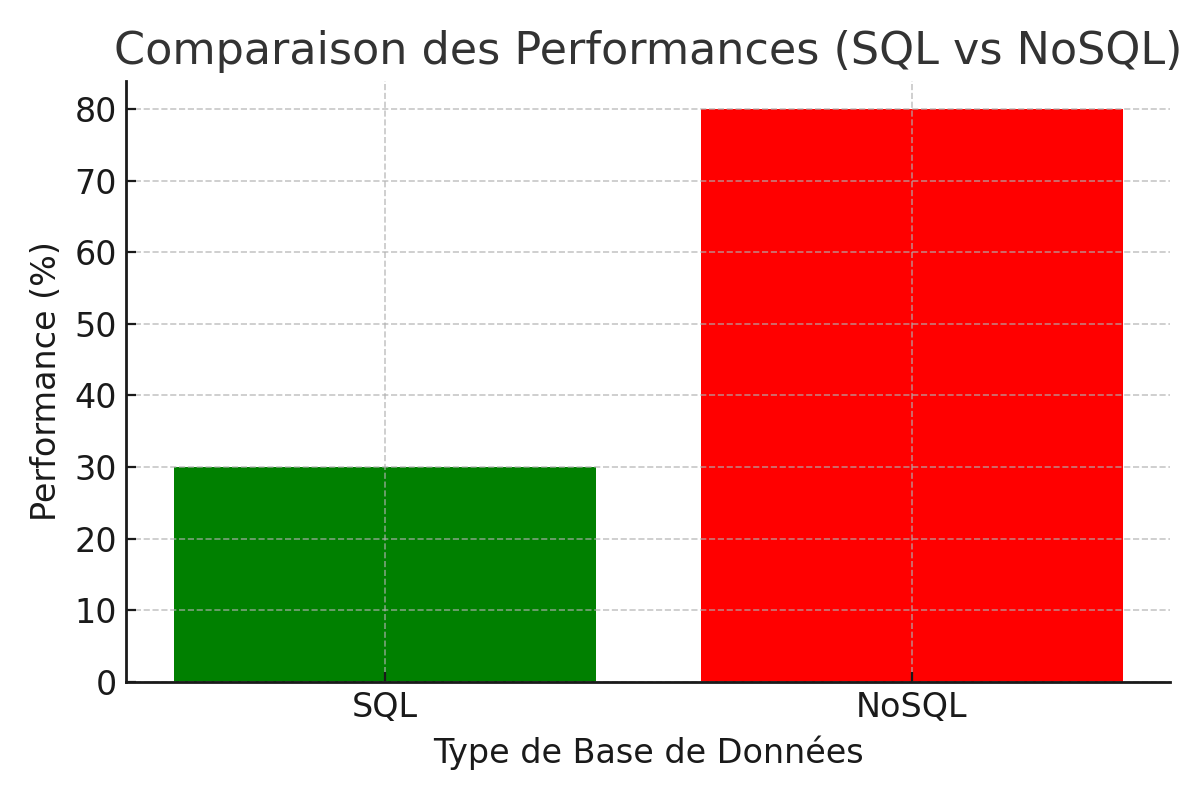
**MongoDB**, avec des pipelines d’agrégation, gère les grandes quantités de données avec une scalabilité horizontale.



Les bases de données NoSQL sont généralement considérées comme plus scalables que les bases de données SQL. Voici pourquoi :

**Scalabilité horizontale** : Les bases NoSQL sont conçues pour être facilement extensibles horizontalement (ajout de nouveaux serveurs pour gérer des charges croissantes). Par exemple, MongoDB et Cassandra permettent de distribuer les données sur plusieurs serveurs.

**SQL privilégie une scalabilité verticale** : Pour améliorer les performances, il faut augmenter les ressources du serveur principal (mémoire, CPU, etc.), ce qui peut devenir coûteux et limité par la capacité matérielle.



Performance (SQL vs NoSQL)

Les bases NoSQL sont également souvent plus performantes que les bases SQL, en particulier dans les cas suivants :

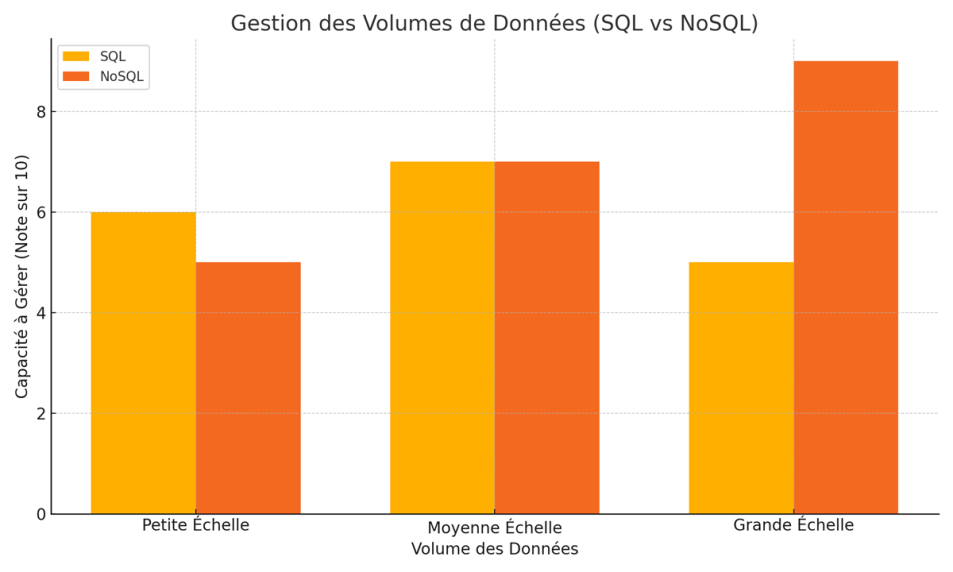
**Lecture/écriture rapide** : NoSQL excelle pour des opérations rapides sur de grandes quantités de données, grâce à l'absence de schéma rigide et à leur architecture optimisée pour des accès rapides.

**Adaptation à des cas spécifiques** : Chaque type de base NoSQL (clé-valeur, documents, colonnes, graphiques) est optimisé pour des cas d'utilisation particuliers.

Cependant, SQL peut surpasser NoSQL dans les scénarios impliquant :

**Transactions complexes** : SQL utilise le modèle ACID (Atomicité, Cohérence, Isolation, Durabilité), garantissant l'intégrité des données.

**Requêtes complexes** : Les bases relationnelles sont adaptées pour gérer des jointures complexes entre plusieurs tables.



**Petite échelle :** SQL a une capacité légèrement supérieure.

**Moyenne échelle :** SQL et NoSQL sont équivalents.

**Grande échelle :** NoSQL surpasse SQL grâce à sa scalabilité et à son adaptation aux environnements distribués.

**Conclusion**

Si le projet implique des données en grande quantité ou un besoin de gestion distribuée, NoSQL est une meilleure option.

Pour des projets plus petits ou nécessitant une forte intégrité relationnelle, SQL reste la solution idéale.

Partie 4 : Conclusion et recommandations

**MongoDB** est plus adapté si l’on souhaite gérer des données semi-structurées, évoluer facilement avec une scalabilité horizontale, et permettre des changements fréquents dans la structure des données (ex. pour des informations produit changeantes).

**SQLite** est bénéfique pour des opérations transactionnelles nécessitant la conformité ACID et des requêtes SQL complexes, mais avec des limites de scalabilité.

**Recommandation finale** : Pour mon cas d’usage c’est avec une base de données relationnelle car mes données sont structurées et le volume des données est **moyen** pas plus de 5000 lignes**.**