Rapport de Projet (ETL Buisness Intelligence )

Le projet **supermarket\_etl\_project** est organisé autour d’un pipeline ETL complet permettant d’extraire, transformer, charger, analyser et visualiser des données de ventes issues d’un supermarché. Sa structure est pensée pour faciliter la modularité, la maintenabilité et la traçabilité des traitements.

**1. Organisation des fichiers et dossiers**

* **Dossier racine (supermarket\_etl\_project)** : contient la configuration globale (config.py), le rapport final (rapport\_de\_projet.pdf), ainsi que plusieurs sous-dossiers fonctionnels.
* **data** :
  + raw : contient les données sources brutes collectées, provenant de différentes sources (open data, générateurs de données, fichiers CSV d’origine).
  + processed : stocke les données nettoyées et transformées, prêtes à être utilisées dans la phase de chargement ou d’analyse.
* **exports** : répertoire pour les résultats d’export, avec une organisation par dates et formats (CSV, HTML, PNG, PDF, etc.).
  + Sous-dossiers charts pour les graphiques générés.
  + Sous-dossiers archive pour conserver l’historique des exports.
* **scripts** : contient le code source en Python réparti en plusieurs modules :
  + etl (par exemple extract.py, transform.py, load.py) pour les étapes du pipeline ETL.
  + analysis pour les scripts d’analyse et génération de rapports statistiques (ex : top 10 produits).
  + dim\_populationi (probablement pour la gestion des dimensions de la base, ex : création des dimensions client, magasin, produit, temps).
  + utils pour les fonctions utilitaires partagées (extraction SQL, connexion base de données, export de fichiers, etc.).
* **sql** : contient les scripts SQL, notamment le script de création du schéma (create\_schema.sql) et les requêtes OLAP (olap\_queries.sql) utilisées pour les analyses multidimensionnelles.

**2. Flux et architecture fonctionnelle**

1. **Extraction des données** :  
   Les données brutes sont récupérées à partir de sources diverses (fichiers CSV bruts, open data comme Open Food Facts, Mockaroo pour les clients fictifs, fichiers de géolocalisation, calendriers). Ces fichiers sont stockés dans le dossier data/raw.
2. **Transformation** :  
   Les données sont nettoyées, harmonisées et enrichies via les scripts Python (transform.py et scripts dans dim\_populationi) pour construire les tables de dimensions (clients, produits, temps, magasins) ainsi que la table de faits des ventes. Les données transformées sont enregistrées dans data/processed.
3. **Chargement dans la base** :  
   Le chargement est effectué dans une base PostgreSQL selon un modèle en étoile défini par le script SQL create\_schema.sql. Les scripts load.py et load\_to\_postgre.py automatisent ce chargement.
4. **Analyse et restitution** :  
   Les requêtes OLAP (extraites de olap\_queries.sql) sont exécutées via les scripts dans scripts/analysis/ pour générer des rapports statistiques, tableaux et graphiques. Ces résultats sont exportés au format CSV, HTML et PNG dans exports.
5. **Visualisation** :  
   Un tableau de bord statique en HTML est généré dans le dossier dashboard pour la restitution visuelle synthétique.

**3. Synthèse**

Cette architecture découple clairement chaque étape du traitement et facilite :

* La gestion des données à différents stades (brut, transformé, chargé).
* L’automatisation des analyses avec des scripts modulaires.
* La traçabilité via un système d’exports organisé par date et type de fichier.
* L’évolutivité grâce à une séparation nette entre extraction, transformation, chargement, analyse et visualisation.

Structure de fichier :

supermarket\_etl\_project/

├── config.py

├── rapport\_de\_projet.pdf

├── dashboard/

│ └── static\_dashboard.html

├── data/

│ ├── raw/

│ └── processed/

├── exports/

│ ├── charts/

│ ├── csv/

│ ├── pdf/

│ └── archive/

├── scripts/

│ ├── analysis/

│ ├── dim\_populationi/

│ ├── utils/

│ ├── analyse.py

│ ├── extract.py

│ ├── load.py

│ ├── load\_to\_postgre.py

│ ├── main.py

│ └── transform.py

└── sql/

├── create\_schema.sql

└── olap\_queries.sql

**Choix techniques**

Pour mener à bien ce projet d’analyse des ventes et du comportement client dans un supermarché, plusieurs choix techniques ont été faits afin d’assurer la robustesse, la modularité et la performance du système.

Le projet est développé principalement en Python 3.13, qui offre une grande flexibilité pour la manipulation des données, le développement du pipeline ETL, ainsi que pour la création de requêtes et de visualisations. L’environnement Python permet également d’intégrer facilement des bibliothèques puissantes telles que Pandas pour la manipulation avancée des données, SQLAlchemy pour l’interaction avec la base de données, et Matplotlib et Seaborn pour la visualisation des résultats.

La base de données choisie est PostgreSQL, un système relationnel réputé pour sa robustesse et ses performances dans la gestion des données complexes et volumineuses. Nous avons opté pour un modèle en étoile, structurant les données autour d’une table de faits (fact\_ventes) et plusieurs tables de dimensions (dim\_produit, dim\_client, dim\_temps, dim\_magasin). Cette modélisation facilite les analyses multidimensionnelles via des requêtes OLAP.

Le pipeline ETL a été entièrement développé en Python, permettant un contrôle précis de chaque étape. Il comprend des modules distincts pour l’extraction des données depuis diverses sources hétérogènes (CSV, JSON, fichiers texte), la transformation (nettoyage, normalisation, enrichissement des données), et le chargement dans la base PostgreSQL. Cette séparation claire garantit la maintenabilité et la modularité du code.

Les requêtes analytiques sont centralisées dans un fichier SQL dédié, ce qui facilite leur gestion et leur réutilisation. Ces requêtes permettent d’obtenir des indicateurs clés tels que le top 10 des produits vendus, l’analyse des ventes par zone géographique, ou encore la corrélation entre catégories de produits et récurrence d’achat.

Pour la restitution visuelle des résultats, nous avons privilégié Matplotlib et Seaborn en raison de leur fiabilité et simplicité d’utilisation. Les résultats sont exportés sous plusieurs formats : CSV pour les données tabulaires, HTML pour les rapports interactifs, et PNG pour les graphiques. Cette diversité de formats répond aux besoins d’analyse, de partage et de présentation.

Enfin, le projet est organisé de manière modulaire avec une structure claire des dossiers, distinguant les scripts d’analyse, les utilitaires, les modules dédiés aux dimensions, ainsi que les fichiers de configuration. Cette organisation facilite la compréhension, la collaboration et l’évolution du projet.

**Modélisation (Version technique et synthétique)**

Le modèle de données est basé sur un **schéma en étoile** comportant une table de faits et plusieurs tables de dimensions.

**Tables**

* **fact\_ventes (table de faits)**
  + Clés étrangères : date\_id, produit\_id, client\_id, magasin\_id
  + Mesures : quantite, montant
* **Dimensions :**
  + **dim\_produit** : code (PK), libelle, categorie, fournisseur
  + **dim\_client** : client\_id (PK), age, sexe, fidelite, ville
  + **dim\_temps** : date\_id (PK), jour, semaine, mois, annee, ferie
  + **dim\_magasin** : magasin\_id (PK), ville, region, surface

**Contraintes**

* Les clés primaires uniques garantissent l’unicité dans chaque dimension.
* Les clés étrangères dans la table de faits assurent l’intégrité référentielle.
* Les agrégations et analyses OLAP sont facilitées par cette structure simple.

(schema table)