TP ETL Débutant (Python + Pandas)

Objectif: pratiquer les bases de l'ETL (Extract, Transform, Load) à partir des fichiers du dossier `sqlite_exports`, avec un script Python.

Date de rendu : 16 Février 2026

Pré-requis

- Python 3.9+
- `pandas` (déjà listé dans requirements)
- VS Code et un terminal

Données disponibles

Vous disposez des CSV suivants dans `sqlite_exports` :

- `customers.csv`, `sellers.csv` (dimensions acteurs)
- `products.csv` (dimension produits)
- `orders.csv` (commandes)
- `order_items.csv` (lignes de commande)
- `order_pymts.csv` (paiements)
- `order_reviews.csv` (avis)
- `geoloc.csv` (géolocalisation par code postal)
- `translation.csv` (ex: mapping de catégories)

Ces fichiers sont proches du schéma Olist; les clés habituelles sont:

- `orders.order_id` → joint avec `order_items.order_id`, `order_reviews.order_id`, `order_pymts.order_id`
- `orders.customer_id` → joint avec `customers.customer_id`
- `order_items.product_id` → joint avec `products.product_id`
- `order_items.seller_id` → joint avec `sellers.seller_id`

Plan du TP

1. Extract

- Lire chaque CSV avec `pandas.read_csv()` (astuce: `low_memory=False`).
- Inspecter les schémas: colonnes, types (`df.dtypes`), quelques lignes (`df.head()`).

2. Transform (nettoyage + enrichissement)

- Parser les dates (toutes les colonnes contenant `date` ou `time`).
- Supprimer les doublons si nécessaire (`drop_duplicates`).
- Gérer les valeurs manquantes (stratégie simple: remplissage ou suppression ciblée).
- Construire un jeu de faits en joignant `order_items` avec `orders`, `customers`, `sellers`, `products`.
- Calculer des métriques:
  - Chiffre d'affaires par mois (somme `price` + `freight_value` si dispo).
  - Top catégories par revenu.
  - Temps de livraison moyen (si colonnes de livraison présentes).
  - Score moyen des avis par mois.
- Enrichir `customers` avec `geoloc` si une clé comme `customer_zip_code_prefix` existe.
- (Bonus) Utiliser `translation.csv` pour traduire les catégories produits.

3. Load

- Sauvegarder les tables (faits/dims/aggregations) en CSV dans un dossier `outputs/`.
- Charger les mêmes tables dans une base SQLite (`etl.db`) avec `to_sql()`.

Tâches détaillées (à rendre)

- T1: Fonctions `extract_sources()`, `transform_data()`, `load_outputs()` dans `tp_etl.py`.
- T2: Rapport rapide (quelques lignes) sur le volume des données et types clés.
- T3: Table d'agrégation `monthly_revenue` avec colonnes: `year_month`, `revenue`.
- T4: Table `top_categories` (top 10) avec `product_category`, `revenue`.
- T5: Table `delivery_metrics` avec `year_month`, `avg_delivery_days` (si dates dispo).
- T6: Export SQLite: tables `dim_customers`, `dim_sellers`, `dim_products`, `fact_order_items`, `monthly_revenue`.
- Bonus: `reviews_monthly` avec `year_month`, `avg_review_score`.

Démarrage rapide

- Installer dépendances:

```
pip install -r requirements.txt
```

- Lancer le script starter:

```
python tp_etl.py --data-dir sqlite_exports --out outputs --sqlite
outputs/etl.db
```

Le script crée `outputs/` si besoin, écrit des CSV et remplit `outputs/etl.db`.

Conseils

- Rendez les jointures robustes: utilisez des `merge(..., how="left")` et traitez les clés manquantes.
- Normalisez les dates: `pd.to_datetime(col, errors="coerce")`.
- Évitez les régressions: affichez des logs simples (shapes des df).

Évaluation

- Qualité du code (lisibilité, fonctions bien découpées)
- Robustesse des transformations (gestion des données manquantes)
- Cohérence des métriques
- Exports correctement produits (CSV + SQLite)

Questions guidées

Extract

- Tailles et colonnes: quelles dimensions pour chaque table (utiliser `df.shape`, `df.columns`)?
- Types: quels `dtypes` détectés, quelles colonnes doivent être converties (dates, numériques)?
- Données manquantes: quelles colonnes critiques comportent des `NaN` et comment les traiter?

Transform

- Dates: quelles colonnes de dates parsées, quel taux de conversion (valeurs `NaT` restantes)?
- Doublons: où en trouvez-vous, et quelle stratégie appliquez-vous (`drop_duplicates`, clé unique)?
- Jointures: quelles clés utilisez-vous, quel pourcentage de correspondances, y a-t-il des lignes orphelines?
- Revenu: comment calculez-vous `item_total` (prix + frais), et gérez-vous les valeurs manquantes de `freight_value` ?

- Outliers: détectez-vous des valeurs anormales (prix, revenu) et comment les traitez-vous?
- Livraison: distribution des `delivery_days` (moyenne, médiane), anomalies (valeurs négatives ou extrêmes)?
- Catégories: appliquez-vous `translation.csv` (taux de couverture du mapping)?

Load

- CSV: quelles tables exportées, tailles et vérification des en-têtes?
- SQLite: quelles tables présentes et combien de lignes par table?
- Cohérence: constatez-vous des écarts entre les exports CSV et SQLite?

Qualité & robustesse

- Logging: comment tracez-vous les `shape` et conversions (dates, types)?
- Erreurs: comment gérez-vous les erreurs de lecture/écriture (chemins, permissions, colonnes absentes)?
- Validations: quelles vérifications simples avant export (colonnes essentielles présentes, types non vides)?

Performance

- Coûts: quelles opérations sont les plus coûteuses, temps d'exécution approximatif?
- Optimisations: types catégoriels, filtrage en amont, éventuel `chunksize` pour les gros CSV?

Reproductibilité

- Paramètres: quelles options d'entrée/sortie garantissent un run reproductible?
- Dépendances: comment figez-vous les versions (requirements, pinning de `pandas`)?

Bonne pratique et exploration !