

**Suivi des commandes des  
revendeurs**

# Distributech

Dossier technique

Nathalie BEDIEE

---

## Table des matières

<b>1. Introduction</b>	<b>2</b>
1.1. Contexte et rappel des objectifs	2
1.2. Périmètre technique du projet	2
<b>2. Spécifications fonctionnelles</b>	<b>3</b>
2.1. Entités principales et règles de gestion	3
2.2. Cas d'usage (utilisation, exploitation, maintenance)	4
<b>3. Architecture et modélisation</b>	<b>4</b>
3.1. Schéma global des flux	4
3.2. Composants logiciels et contraintes	5
3.3. Modélisation des données (MCD, MLD)	6
3.3.1. MCD – modèle conceptuel	6
3.3.2. MLD – modèle logique	7
3.3.3. Schéma relationnel	8
<b>4. Spécifications techniques du pipeline ETL</b>	<b>8</b>
4.1. Processus principal	8
4.2. Extract – lecture et validation des sources (CSV, SQLite)	10
4.3. Transform – contrôles, normalisation, gestion erreurs/doublons	11
4.4. Load – insertion en base	13
4.5. Gestion du stock (vue et export)	14
4.6. Tableau de bord	16
<b>5. Plan de test</b>	<b>18</b>
5.1. Objectif et périmètre	18
5.2. Jeux de données de test et cas de tests	19
5.3. Critères de réussite et acceptation	21
<b>6. Conclusion</b>	<b>23</b>

# 1. Introduction

## 1.1. Contexte et rappel des objectifs

**Distributech**, grossiste en équipements électroniques, reçoit les **commandes** de ses **revendeurs régionaux** au format **CSV** et gère les **mouvements de stock** dans une base **SQLite**.

Le besoin est de **centraliser** et **fiabiliser** ces données dans une **base relationnelle unique**, alimentée par un pipeline **ETL**.

Les objectifs principaux sont :

- **Centraliser toutes les données** dans une base unique :
  - **Commandes** envoyées par les revendeurs (CSV),
  - **Produits** du catalogue (SQLite),
  - **Revendeurs** et leurs **régions** (SQLite),
  - **Mouvements de stock** (réapprovisionnements, SQLite).
- **Automatiser l'intégration** via un pipeline **ETL** (*Extract => Transform => Load*).
- **Calculer et exporter l'état des stocks** à chaque cycle.
- Fournir aux équipes **logistique** et **commerciale** un **accès simple** aux informations consolidées (stock, commandes, chiffre d'affaires).

## 1.2. Périmètre technique du projet

Le présent **dossier technique** décrit la **solution mise en place** pour atteindre ces objectifs.

Il comprend :

- l'**architecture cible** (sources, ETL, base SQL, exports, tableau de bord),
- la **modélisation de la base relationnelle** (MCD, MLD),
- les **spécifications techniques du pipeline ETL**,
- la **conception du tableau de bord**,
- le **plan de test**.

## 2. Spécifications fonctionnelles

### 2.1. Entités principales et règles de gestion

- **Région** : zone géographique. Chaque revendeur appartient à une seule région.
- **Revendeur** : client régional, identifié par un ID, associé à une région.
- **Produit** : article du catalogue, identifié par un code, un nom et un coût unitaire.
- **Commande** : ensemble d'articles demandés par un revendeur, identifié par un numéro unique et une date.
- **Ligne de commande** : détail d'une commande (produit, quantité, prix unitaire).
- **Mouvement de production** : entrée de stock enregistrée dans SQLite (réassort).
- **Stock** : quantité disponible d'un produit, calculée à une date donnée :

$$\text{Stock}(\text{produit}) = \sum(\text{entrées production du produit}) - \sum(\text{sorties cmds du produit})$$

#### Règles de gestion clés :

- Chaque **revendeur** appartient à une seule **région** (référentiel SQLite).
- L'identifiant de **région** peut être absent dans les commandes CSV, mais il est systématiquement **retrouvé via le référentiel revendeur**.
- Chaque **commande** est associée à un seul revendeur et possède un **numéro unique**.
- Chaque **ligne de commande** est liée à un produit et contient la **quantité** et le **prix unitaire** au moment de la commande.
- Les identifiants (**revendeurs, produits**) sont uniques et non nuls.
- Les **quantités** et **coûts** sont strictement positifs.
- Les **dates** doivent être valides et ne pas être futures (dans la production).
- Les données incohérentes (valeurs négatives, références inconnues, doublons) sont **rejetées** et consignées dans les **logs**.

## 2.2. Cas d'usage (utilisation, exploitation, maintenance)

### ➤ En utilisation (équipes logistiques et commerciales)

- Consulter le **stock courant** et le **stock à une date donnée**.
- Consulter les **commandes** par période, produit ou région.
- Calculer le **chiffre d'affaires** par région ou global.
- Exporter les résultats au format **CSV** pour exploitation dans un tableur (ex : Excel).

### ➤ En exploitation (personnel technique)

- Déclencher et superviser l'**ETL**.
- Contrôler les **logs d'exécution**.
- Gérer les anomalies (fichiers manquants, données incohérentes).

### ➤ En maintenance (développeur / DBA)

- Assurer la **maintenance corrective** (erreurs, incidents).
- Assurer la **maintenance évolutive** (nouvelles règles, nouvelles sources).
- Mettre à jour la **documentation technique**.

## 3. Architecture et modélisation

### 3.1. Schéma global des flux

Le système repose sur deux **sources** de données :

- les **commandes** envoyées en **fichiers CSV** par les revendeurs,
- la base **SQLite** contenant les **référentiels** (régions, revendeurs, produits) et les **mouvements de production** (entrées de stock).

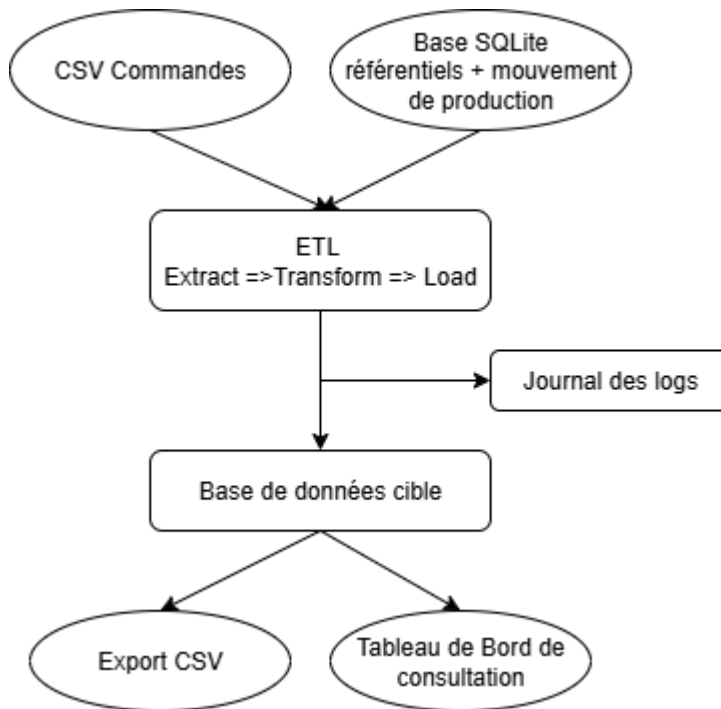
Ces sources alimentent un **pipeline ETL** en trois étapes (**Extract, Transform, Load**) développé en **Python**.

Tout incident (colonnes manquantes, incohérences, doublons) est enregistré dans un **journal des logs** pour assurer la traçabilité.

Les données sont ensuite centralisées dans une **base relationnelle SQL**. Les résultats sont accessibles via :

- des **exports CSV** (état des stocks, reporting),
- un **tableau de bord console** pour consultation simple.

Le schéma ci-dessous illustre le flux :



### 3.2. Composants logiciels et contraintes

- **Langage** : Python 3.x.
- **Bibliothèques principales** :
  - *pandas* pour le traitement de données,
  - *mysql.connector* pour la connexion à la base relationnelle,
  - *sqlite3* pour la lecture base SQLite.
- **Base relationnelle cible** : MySQL.
- **Compatibilité OS** : Windows et Linux.
- **Exécution** : mode **batch** (pipeline complet sans interaction utilisateur).
- **Organisation des répertoires de données**
  - *data/in* : dépôt des fichiers CSV entrants
  - *data/treated* : fichiers déplacés après traitement
  - *data/log* : journaux d'exécution
  - *data/stock* : exports CSV générés (état des stocks)

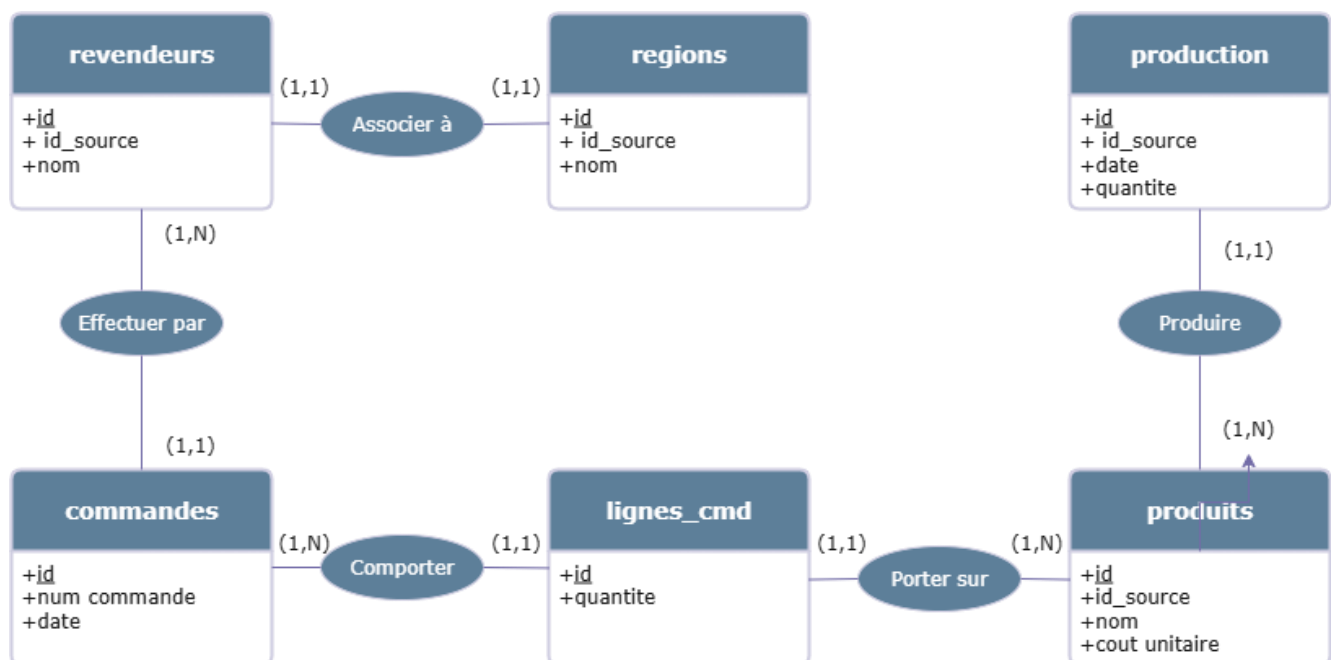
### 3.3. Modélisation des données (MCD, MLD)

La modélisation des données repose sur la méthode **Merise** avec deux représentations complémentaires :

1. **Modèle Conceptuel de Données (MCD)** : permet d'identifier les entités métier, leurs propriétés et les relations entre elles, sans contrainte technique.
2. **Modèle Logique de Données (MLD)** : traduit le MCD en tables et colonnes adaptées à une base relationnelle.
3. **Schéma relationnel** : illustre visuellement la structure finale de la base avec les clés primaires, étrangères et les associations.

#### 3.3.1. MCD – modèle conceptuel

Le schéma ci-dessous présente les principales entités (revendeurs, commandes, produits, etc.) et leurs relations métier.



### 3.3.2. MLD – modèle logique

À partir du MCD, le modèle logique définit les tables relationnelles, leurs attributs, et les contraintes (PK, FK, unicité, intégrité).

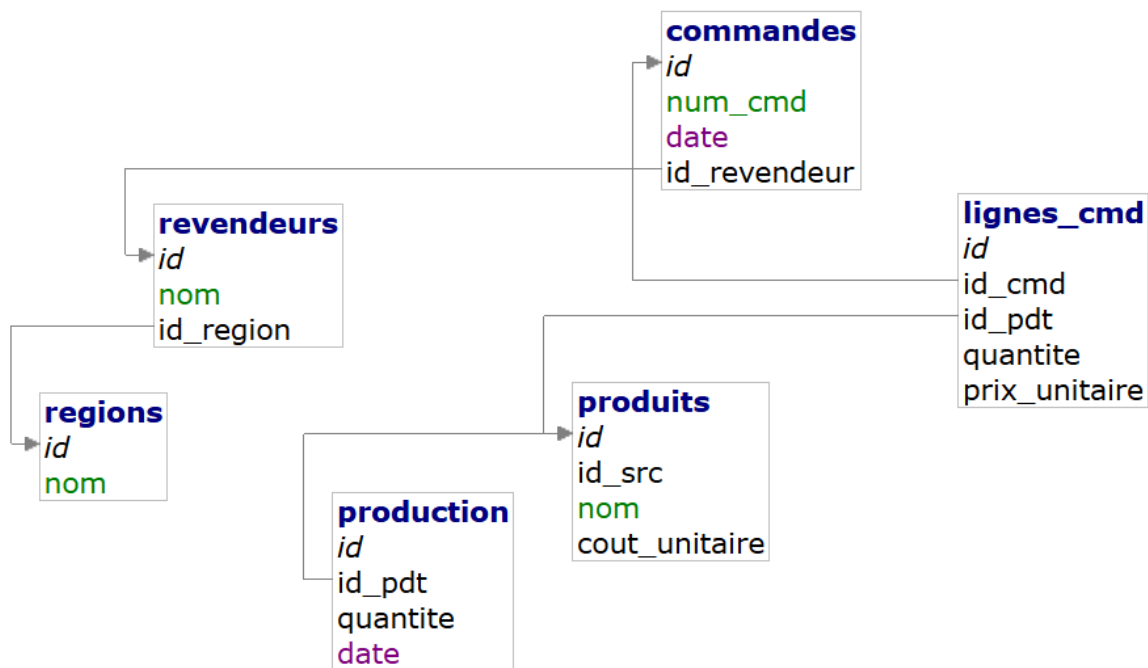
- **regions**
  - **id** : INT, PK
  - **nom** : VARCHAR(50) NOT NULL
- **revendeurs**
  - **id** : INT, PK
  - **nom** : VARCHAR(50) NOT NULL
  - **#id\_region** : INT, FK → regions.id, NOT NULL
- **produits**
  - **id** : INT, PK, AUTO\_INCREMENT
  - **id\_src** : INT NOT NULL
  - **nom** : VARCHAR(50) NOT NULL
  - **cout\_unitaire** : FLOAT NOT NULL
- **production**
  - **id** : INT, PK
  - **#id\_pdt** : INT, FK → produits.id, NOT NULL
  - **quantite** : INT NOT NULL
  - **date** : DATE NOT NULL
- **commandes**
  - **id** : INT, PK, AUTO\_INCREMENT
  - **num\_cmd** : VARCHAR(30) NOT NULL
  - **date** : DATE NOT NULL
  - **#id\_revendeur** : INT, FK → revendeurs.id, NOT NULL
- **lignes\_cmd**
  - **id** : INT, PK, AUTO\_INCREMENT
  - **#id\_cmd** : INT, FK → commandes.id, NOT NULL
  - **#id\_pdt** : INT, FK → produits.id, NOT NULL
  - **quantite** : INT NOT NULL
  - **prix\_unitaire** : FLOAT NOT NULL



### 3.3.3. Schéma relationnel

Le schéma relationnel ci-dessous a été généré à partir de l'outil **Adminer**, connecté à la base centrale **MySQL**.

Il représente la structure finale réellement implémentée : les tables avec leurs colonnes principales, ainsi que les clés primaires et étrangères qui assurent l'intégrité des liens entre entités.



## 4. Spécifications techniques du pipeline ETL

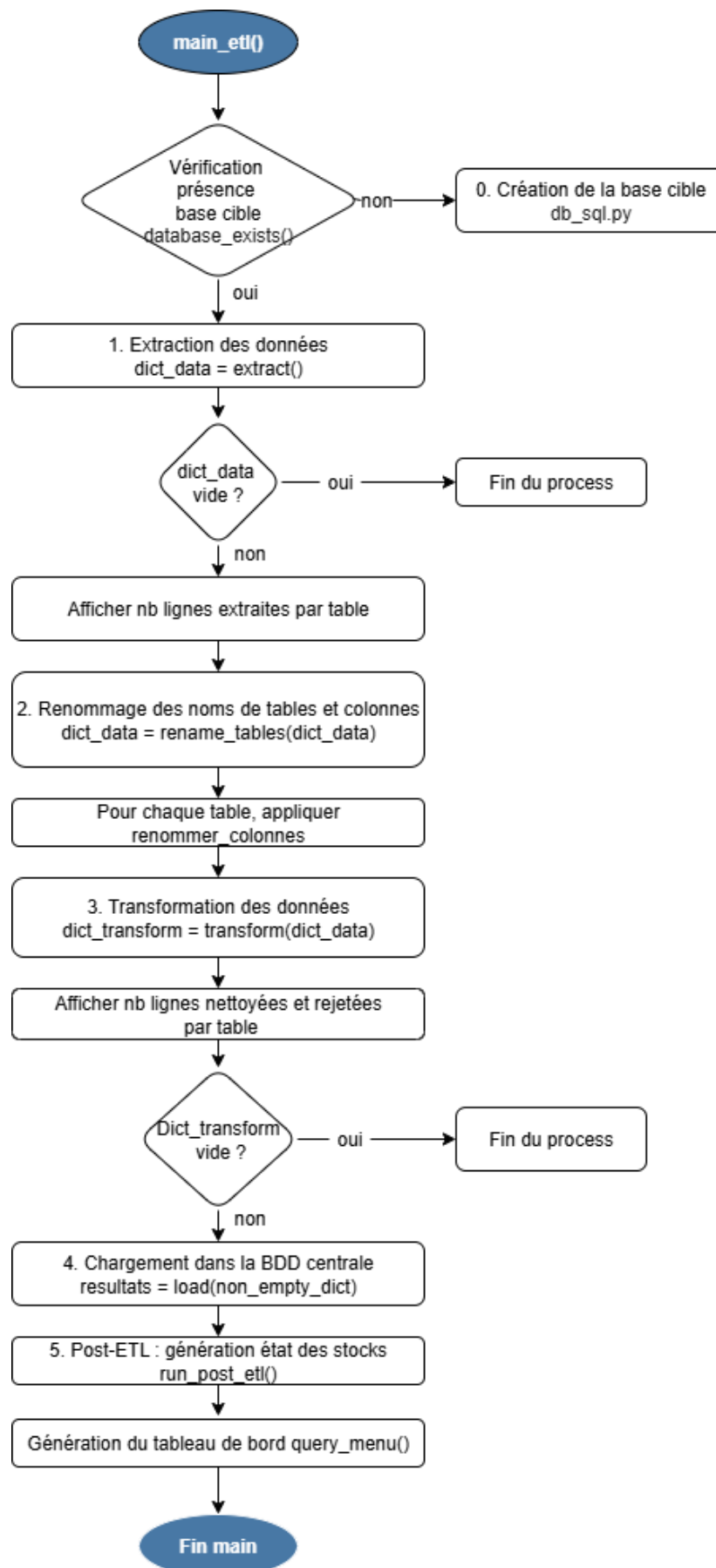
### 4.1. Processus principal

Le fichier **main\_etl.py** orchestre l'ensemble du pipeline ETL, depuis la vérification de la base centrale jusqu'à la génération de l'état de stock et l'accès au tableau de bord.

Chaque étape est exécutée en séquence, avec journalisation des événements ([INFO], [OK], [ALERTE], [ERREUR]) dans le log.

Le pipeline fonctionne en mode batch, sans interaction utilisateur, jusqu'au lancement du tableau de bord.

Le diagramme suivant illustre les principales étapes :



En complément, chaque fichier Python joue un rôle spécifique dans le processus.  
Le tableau ci-dessous synthétise ces responsabilités :

Fichier	Rôle principal
Main_etl.py	Pilotage global du pipeline ETL
db_sql.py	Création de la base centrale (si absente)
extract.py	Extraction des données depuis CSV et SQLite
transform.py	Nettoyage et validation des données
load.py	Chargement dans la base centrale
post_etl.py	Génération de l'état des stocks après ETL
commun.py	Fonctions partagées (renommage, logs, vérification base, etc.)
query_menu.py	Génère le tableau de Bord (interrogations SQL sur la base centrale, affichage / reporting)

#### 4.2. Extract – lecture et validation des sources (CSV, SQLite)

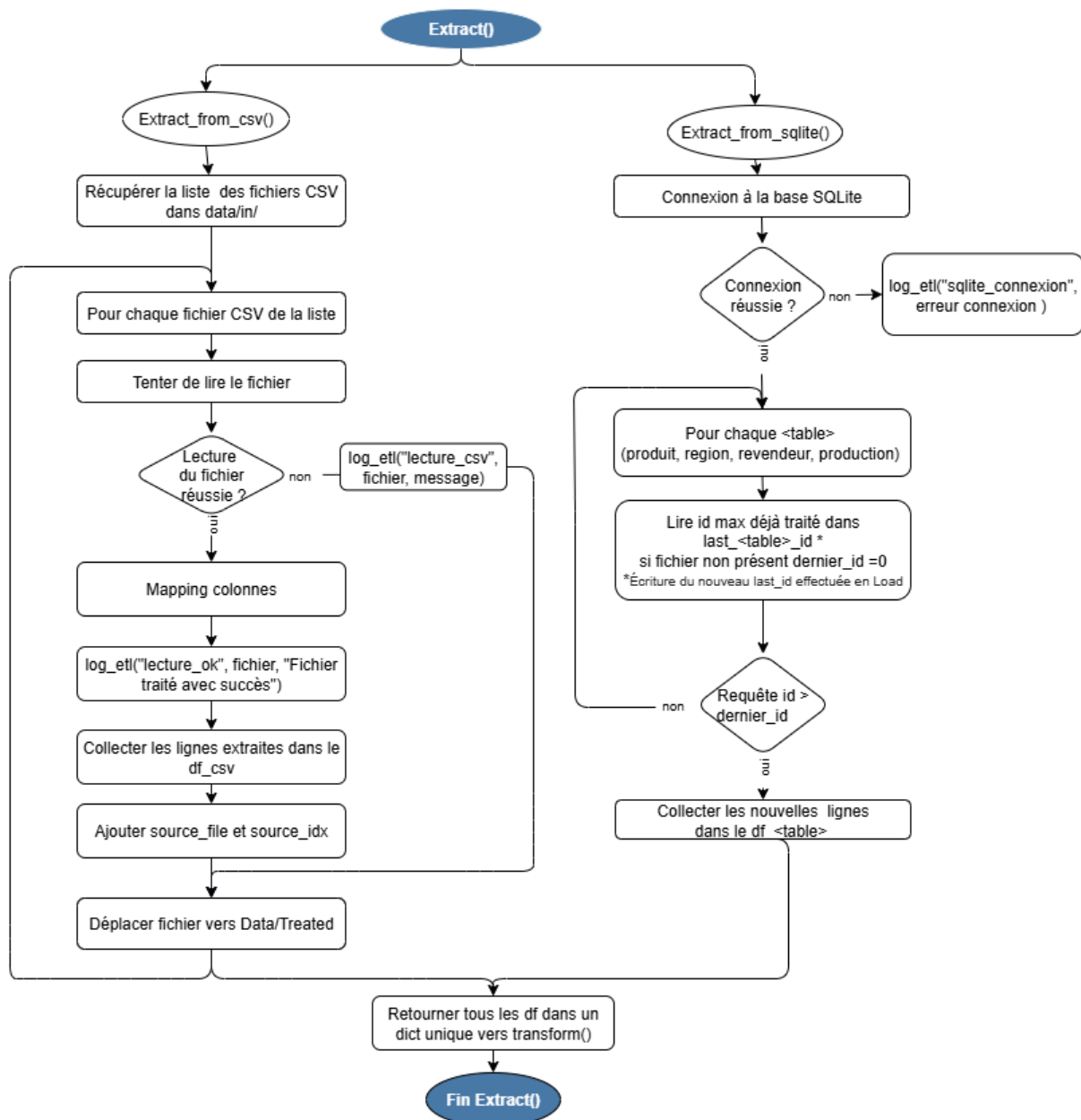
L'extraction est assurée par le script **extract.py**.

Deux sources sont prises en compte :

- les **fichiers CSV** contenant les commandes,
- la **base SQLite** fournissant les données de référence (produits, régions, revendeurs, production).

Pour chaque source, une gestion d'erreur (connexion, structure, lecture) est prévue : les incidents sont loggés et les fichiers ou lignes KO sont exclus du traitement.

Le schéma ci-dessous illustre le fonctionnement de ce module, en distinguant la branche CSV et la branche SQLite.



### 4.3. Transform – contrôles, normalisation, gestion erreurs/doublons

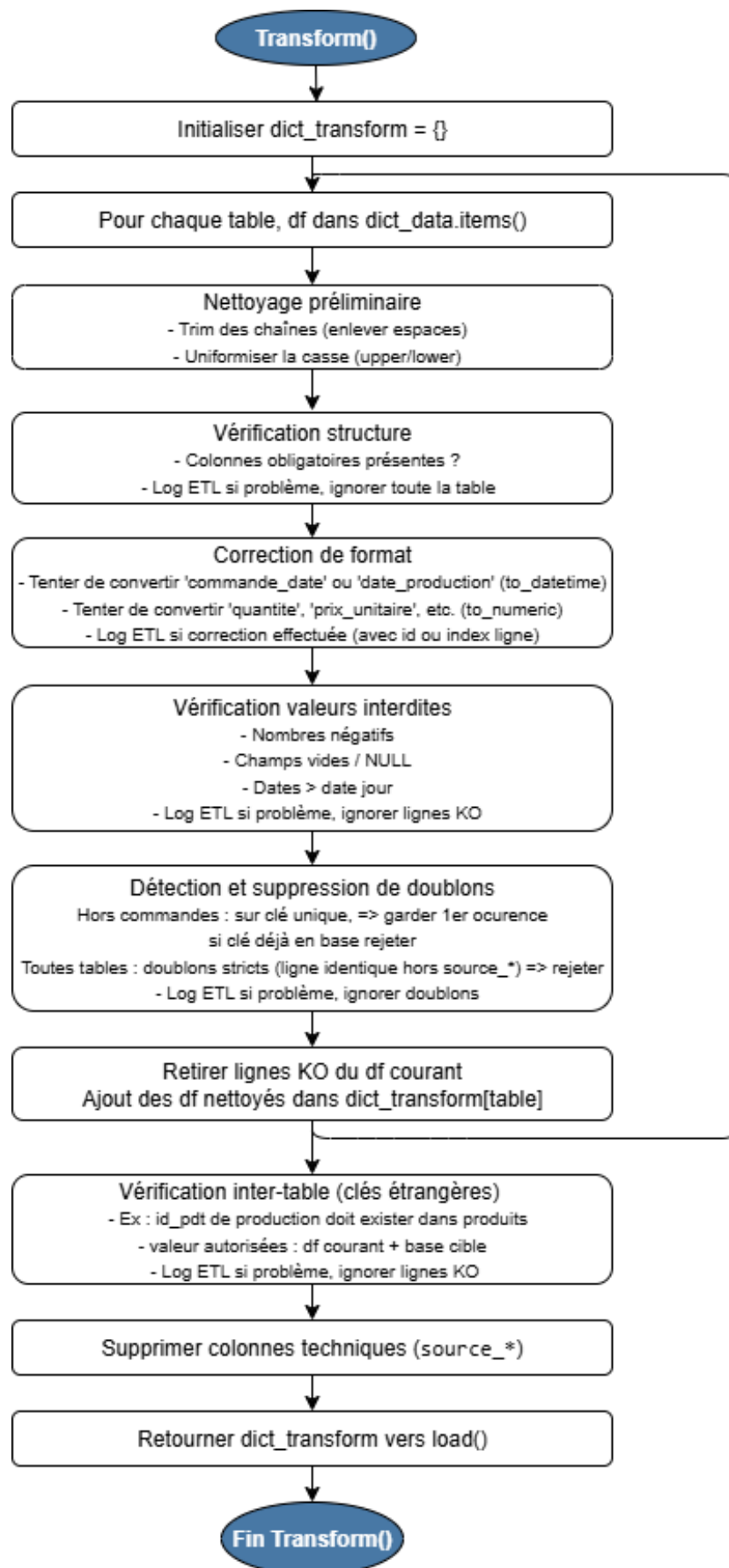
Le module **transform.py** applique les règles de qualité définies :

- normalisation des formats (dates, numériques, chaînes),
- suppression des doublons,
- gestion des incohérences détectées.

Chaque rejet est tracé, assurant la transparence des pertes de données.

À l'issue de cette étape, les tables sont prêtes pour une intégration fiable en base centrale.

Le processus suivi est représenté dans le diagramme ci-dessous.

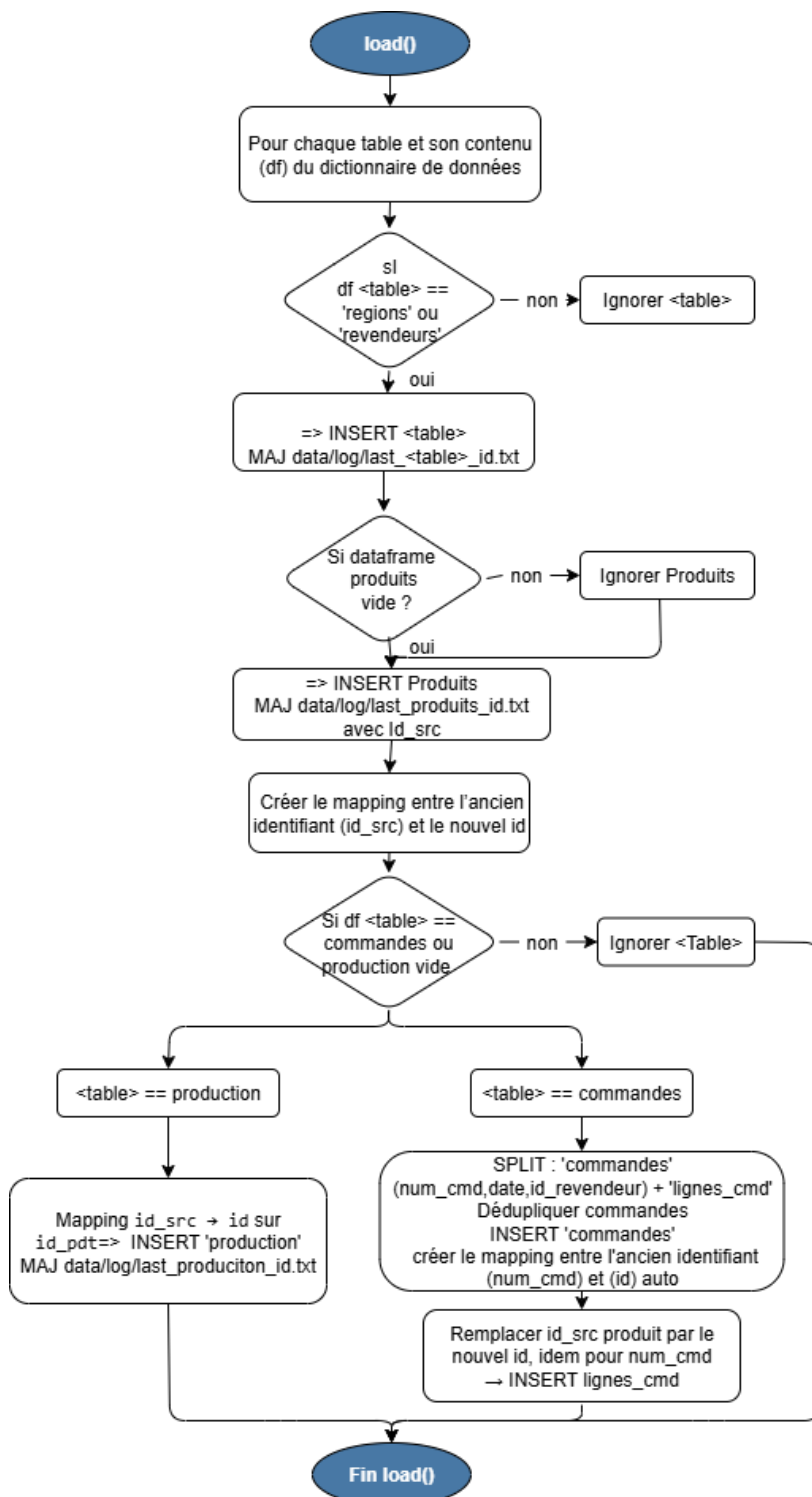


#### 4.4. Load – insertion en base

Le module **load.py** prend en charge le chargement des données transformées dans la base MySQL.

Les dépendances entre tables et la correspondance des clés sont gérées automatiquement.

Un suivi est enregistré : nombre de lignes insérées, anomalies détectées et erreurs critiques bloquantes.



#### 4.5. Gestion du stock (vue et export)

La gestion du stock et des indicateurs associés s'appuie sur trois **vues SQL** créées en fin de pipeline ETL par **post\_etl.py**. Elles centralisent les calculs et simplifient l'export.

- Vue **v\_stock** : État courant du stock.

Principe :

$$\text{Stock}(\text{produit}) = \sum(\text{entrées production du produit}) - \sum(\text{sorties cmds du produit})$$

```
CREATE OR REPLACE VIEW v_stock AS
SELECT
  p.id      AS produit_id,
  p.nom     AS produit_nom,
  COALESCE(SUM(prod.quantite), 0) AS total_entrees,
  COALESCE(SUM(lcmd.quantite), 0) AS total_sorties,
  COALESCE(SUM(prod.quantite), 0) - COALESCE(SUM(lcmd.quantite), 0) AS stock_courant
FROM produits p
LEFT JOIN production prod ON prod.id_pdt = p.id
LEFT JOIN lignes_cmd lcmd ON lcmd.id_pdt = p.id
GROUP BY p.id, p.nom;
```

Usage : détection des stocks négatifs et base de l'export CSV.

- Vue **v\_cmds\_par\_region** : Commandes agrégées par région et produit.

Principe :

$$\text{Cmds}(\text{region}, \text{produit}) = \sum(\text{quantités commandées du produit par la région})$$

```
CREATE OR REPLACE VIEW v_cmds_par_region AS
SELECT
  r.nom AS region_nom,
  p.id  AS produit_id,
  p.nom AS produit_nom,
  SUM(lc.quantite) AS total_commandes
FROM lignes_cmd lc
JOIN commandes c ON lc.id_cmd = c.id
JOIN revendeurs v ON c.id_revendeur = v.id
JOIN regions r ON v.id_region = r.id
JOIN produits p ON lc.id_pdt = p.id
GROUP BY r.nom, p.id, p.nom
ORDER BY r.nom, p.id;
```

Usage : analyse de la répartition des ventes par région.

➤ Vue **v\_chiffre\_affaires\_par\_region** : Chiffre d'affaires consolidé.

Principe :

- $CA(region) = \sum(quantité \times prix\ unitaire)$
- $Nb\_commandes(region) = \sum(commandes\ distinctes)$

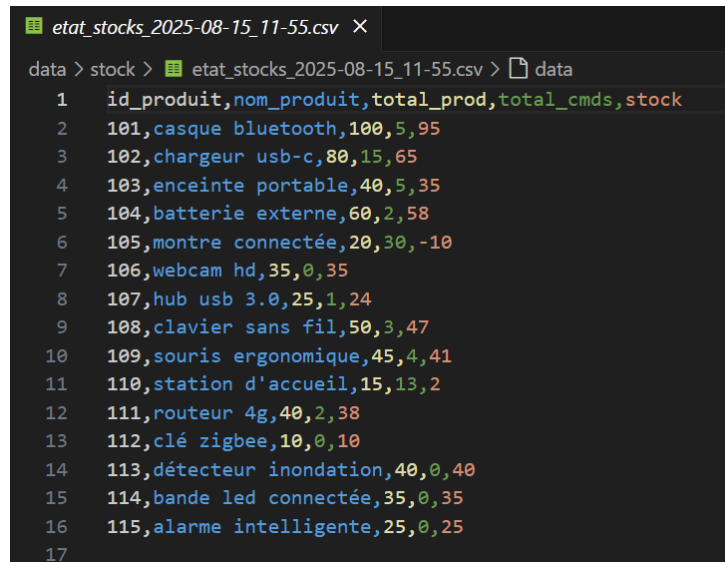
```
CREATE OR REPLACE VIEW v_chiffre_affaires_par_region AS
SELECT
    r.id    AS region_id,
    r.nom   AS region_nom,
    ROUND(SUM(lc.quantite * lc.prix_unitaire), 2) AS chiffre_affaires,
    COUNT(DISTINCT c.num_cmd) AS nb_commandes
FROM lignes_cmd lc
JOIN commandes c    ON lc.id_cmd      = c.id
JOIN revendeurs re  ON c.id_revendeur = re.id
JOIN regions r      ON re.id_region   = r.id
GROUP BY r.id, r.nom
ORDER BY r.nom;
```

➤ **Export CSV du stock**

En fin de pipeline, post\_etl.py lit v\_stock, signale les stocks négatifs, puis génère un fichier dans data/stock/.

- Nom : etat\_stocks\_YYYY-MM-DD\_HH-MM.csv
- Colonnes : produit\_id,produit\_nom,total\_entrees,total\_sorties,stock\_courant
- Format : UTF-8, séparateur ,
- Exploitation : ouverture dans un tableur.

Exemple de fichier :



```
etat_stocks_2025-08-15_11-55.csv X
data > stock > etat_stocks_2025-08-15_11-55.csv > data
1 id_produit,nom_produit,total_prod,total_cmds,stock
2 101,casque bluetooth,100,5,95
3 102,chargeur usb-c,80,15,65
4 103,enceinte portable,40,5,35
5 104,batterie externe,60,2,58
6 105,montre connectée,20,30,-10
7 106,webcam hd,35,0,35
8 107,hub usb 3.0,25,1,24
9 108,clavier sans fil,50,3,47
10 109,souris ergonomique,45,4,41
11 110,station d'accueil,15,13,2
12 111,routeur 4g,40,2,38
13 112,clé zigbee,10,0,10
14 113,détecteur inondation,40,0,40
15 114,bande led connectée,35,0,35
16 115,alarme intelligente,25,0,25
17
```



## 4.6. Tableau de bord

Le module `query_menu.py` fournit un **tableau de bord en ligne de commande** destiné aux équipes logistiques et commerciales. Il permet d'interroger directement la base relationnelle, de consulter les principaux indicateurs et d'exporter les résultats au format CSV.

### Fonctionnalités disponibles

Le tableau de bord propose six interrogations réparties en trois volets :

- **Stock global**
  1. État actuel (lecture de la vue `v_stock`).
  2. État à une date donnée (requête avec borne temporelle sur `production.date` et `commandes.date`).
- **Commandes par région**
  3. Totaux globaux par région et produit (vue `v_cmds_par_region`).
  4. Totaux limités à une période définie par l'utilisateur.
- **Chiffre d'affaires par région**
  5. Totaux globaux par région (vue `v_chiffre_affaires_par_region`).
  6. Totaux limités à une période définie par l'utilisateur.

L'écran principal se présente ainsi :

```
***** TABLEAU DE BORD *****

[ STOCK GLOBAL ]
  1. Actuel
  2. A une date précise

[ COMMANDES PAR REGION ]
  3. Globales
  4. Sur une période

[ CHIFFRE D'AFFAIRES PAR REGION ]
  5. Globales
  6. Sur une période

  0. Quitter

Votre choix : █
```

## Affichage et export

Les résultats sont affichés sous forme de tableau dans la console. L'utilisateur peut ensuite confirmer l'export en CSV :

- dossier de destination : data/stock/
- encodage UTF-8, séparateur ,
- nommage horodaté (stock\_global\_actuel\_2025-08-18\_21-30.csv)

## Exemples de résultats

### ➤ Stock global (actuel)

```
***** STOCK GLOBAL (actuel) *****

produit_id    produit_nom  total_entrees  total_sorties  stock_courant
1             casque bluetooth    100           10           90
2             chargeur usb-c      240           15          225
3             enceinte portable    80            5           75
4             batterie externe    60            2           58
5             montre connectée    60           30           30
6             webcam hd           35            0           35
7             hub usb 3.0          25            1           24
8             clavier sans fil     50            6           44
9             souris ergonomique   90            4           86
10            station d'accueil     30           13           17
11            routeur 4g            40            2           38
12            clé zigbee            10            0           10
13            détecteur inondation   40            0           40
14            bande led connectée   35            0           35
15            alarme intelligente   25            0           25

15 ligne(s).
Exporter en CSV ? (o/n) : █
```

### ➤ Chiffre d'affaires par région (à partir d'une date)

```
Votre choix : 4
Date début (YYYY-MM-DD, vide = commande jusqu'à date fin) : 2025-07-15
Date fin (YYYY-MM-DD, vide = commande à partir de date début) :

***** COMMANDES PAR REGION (à partir du 2025-07-15) *****

region  id_produit    produit  total_commandes
auvergne-rhône-alpes  102    chargeur usb-c           3
bretagne              110  station d'accueil           3
île-de-france         107      hub usb 3.0             1
occitanie              104  batterie externe           2
occitanie              109  souris ergonomique         2

5 ligne(s).
Exporter en CSV ? (o/n) : █
```

### ➤ Chiffre d'affaires par région (sur une période)

```
Votre choix : 6
Date début (YYYY-MM-DD, vide = commande jusqu'à date fin) : 2025-07-01
Date fin (YYYY-MM-DD, vide = commande à partir de date début) : 2025-08-01

***** CHIFFRE D'AFFAIRES PAR REGION (du 2025-07-01 au 2025-08-01) *****

  region  total_chiffre_affaires  nb_commandes
auvergne-rhône-alpes           40.5             1
bretagne                       274.5             3
île-de-france                  1628.6             6
occitanie                      184.8             3

4 ligne(s).
Exporter en CSV ? (o/n) :
```

## Gestion des dates et contrôles

- Les saisies de dates sont vérifiées au format YYYY-MM-DD.
- En cas d'inversion des bornes, l'ordre est corrigé automatiquement.
- Si une vue n'existe pas encore (ETL non exécuté), un message explicite invite à lancer le processus.

## 5. Plan de test

### 5.1. Objectif et périmètre

L'objectif du plan de test est de valider la robustesse et la fiabilité du pipeline ETL mis en place pour le projet Distributech.

Les tests couvrent deux périmètres principaux :

- **Tests CSV** : vérification du traitement des fichiers de commandes fournis par les revendeurs. Ces tests permettent de s'assurer que les contrôles de structure, de format, de cohérence métier et de gestion des doublons sont correctement appliqués.
- **Tests SQLite** : validation de l'intégration des mouvements de production insérés dans la base locale (ajouts valides, gestion des clés étrangères, agrégations de stock).

Les objectifs spécifiques sont :

- Détecter et rejeter systématiquement les données invalides (colonnes manquantes, dates incohérentes, références inexistantes, valeurs interdites).
- Charger correctement les données valides et les rendre disponibles dans la base centrale.
- Garantir la traçabilité des anomalies via les logs et l'archivage des fichiers.
- Vérifier que l'état du stock est correctement calculé à partir des données chargées et qu'une alerte est émise en cas de stock négatif.

## 5.2. Jeux de données de test et cas de tests

Les jeux de données de test ont été élaborés pour couvrir un maximum de situations réelles et de cas limites rencontrés lors du traitement des commandes et de la mise à jour des stocks. Ils se répartissent en deux catégories :

Un ensemble de fichiers de commandes factices a été créé dans le répertoire data/test/.

ID	Description du cas de test
TEST01	<b>Objet</b> : Fichiers et données cohérentes <b>Fichier</b> : commande_revendeur_tech_express.csv <b>Attendu</b> : fichier déplacé dans treated, données chargées. <b>Log</b> : Vues mises à jour et CSV stock genere
TEST02	<b>Objet</b> : Colonne manquante <b>Fichier</b> : cmd_revendeur_test_colonne_manquante.csv <b>Attendu</b> : fichier rejeté <b>Log</b> : [ERREUR] colonnes manquantes
TEST03	<b>Objet</b> : Date future <b>Fichier</b> : cmd_revendeur_test_date_future.csv <b>Attendu</b> : lignes rejetées <b>Log</b> : [ALERTE] date future
TEST04	<b>Objet</b> : Date incohérente (format invalide) <b>Fichier</b> : cmd_revendeur_test_date_incoherente.csv <b>Attendu</b> : lignes rejetées <b>Log</b> : [ALERTE] format date
TEST05	<b>Objet</b> : Doublons intra-fichier <b>Fichier</b> : cmd_revendeur_test_duplicata.csv <b>Attendu</b> : doublons supprimés (1re occurrence gardée) <b>Log</b> : [ALERTE] doublon
TEST06	<b>Objet</b> : Produit inexistant <b>Fichier</b> : cmd_revendeur_test_id_pdt_inexistant.csv <b>Attendu</b> : lignes rejetées (clé étrangère invalide) <b>Log</b> : [ALERTE] id_produit inconnu

TEST07	<b>Objet</b> : Revendeur non référencé <b>Fichier</b> : cmd_revendeur_test_autre_revendeur.csv <b>Attendu</b> : lignes rejetées (clé étrangère invalide) <b>Log</b> : [ALERTE] id_revendeur inconnu
TEST08	<b>Objet</b> : Revendeur vide <b>Fichier</b> : cmd_revendeur_test_id_revendeur_vide.csv <b>Attendu</b> : lignes rejetées <b>Log</b> : [ALERTE] champ vide
TEST09	<b>Objet</b> : Prix unitaire vide <b>Fichier</b> : cmd_revendeur_test_prix_vide.csv <b>Attendu</b> : lignes rejetées <b>Log</b> : [ALERTE] prix manquant
TEST10	<b>Objet</b> : Quantité négative <b>Fichier</b> : cmd_revendeur_test_qte_neg.csv <b>Attendu</b> : lignes rejetées <b>Log</b> : [ALERTE] quantité négative
TEST11	<b>Objet</b> : Quantité non numérique <b>Fichier</b> : cmd_revendeur_test_qte_non_num.csv <b>Attendu</b> : lignes rejetées <b>Log</b> : [ALERTE] quantité non numérique
TEST12	<b>Objet</b> : Cas valide menant à stock négatif <b>Fichier</b> : cmd_revendeur_test_valide_alerte_stock.csv <b>Attendu</b> : lignes chargées, stock < 0 possible <b>Log</b> : [OK] commandes chargées Une alerte console signale les produits nécessitant un réassort.
TEST13	<b>Objet</b> : Fichier doublon <b>Fichier</b> : commande_revendeur_tech_express_fichier_doublon.csv <b>Attendu</b> : fichier ignoré (déjà traité) <b>Log</b> : [INFO] fichier déjà traité
TEST14	<b>Objet</b> : Test de charge (multi-fichiers) <b>Fichiers</b> : tous les fichiers CSV de test placés ensemble dans data/in/ <b>Attendu</b> : le pipeline traite l'ensemble des fichiers dans un seul cycle d'exécution, sans crash ni ralentissement anormal <b>Log</b> : [INFO] n fichiers traités + bilan par table

TEST15	<p><b>Objet</b> : Ajout de nouvelle production valide depuis SQLITE</p> <p><b>Action</b> : insert dans la base SQLITE avec le script db_stock_add.py</p> <p><b>Attendu</b> : nouvelles entrées intégrées dans la base centrale</p> <p><b>Log</b> : [OK] production chargée</p>
--------	--

### 5.3. Critères de réussite et acceptation

Les critères de réussite définissent les conditions nécessaires pour considérer le pipeline ETL comme **validé**. Ils couvrent à la fois les aspects fonctionnels, techniques et de performance.

- **Traitement des fichiers CSV**

- Tous les fichiers placés en entrée sont correctement détectés et traités.
- Les fichiers valides sont intégrés dans la base centrale et archivés en treated/OK.
- Les fichiers invalides sont rejetés, archivés en treated/NOK et référencés dans le log.

- **Gestion des données**

- 100 % des données invalides sont rejetées avec une raison explicite (message de rejet en console et dans le log).
- Les données valides sont intégrées dans les tables cibles sans perte ni modification non souhaitée.
- Les doublons sont éliminés conformément à la règle métier (1re occurrence conservée).

- **Logs et traçabilité**

- Chaque exécution génère un bilan lisible en console et un log détaillé dans le fichier de suivi.
- Les messages affichés suivent une nomenclature simple : [OK] pour les fichiers valides, [ALERTE] pour ceux non conformes ou avec anomalie, [ERREUR] en cas de problème de lecture.
- Les anomalies sont systématiquement tracées avec la source, la table et l'identifiant ou la ligne concernée.

## • Calcul et export du stock

- La vue v\_stock reflète correctement l'état du stock par produit :  $\text{stock} = \Sigma(\text{production}) - \Sigma(\text{commandes})$ .
- Le stock peut être positif ou négatif (pas de correction forcée).
- Le fichier stock\_export.csv est généré à chaque exécution avec le contenu complet et fidèle à la vue.

## • Performance et charge

- Le pipeline supporte le traitement simultané de plusieurs fichiers en entrée sans crash ni ralentissement anormal.
- Le temps de traitement reste compatible avec un usage hebdomadaire (quelques secondes pour les jeux de test).

## • Exemple de log d'exécution

```
data > log > log_etl_2025-08-28.csv > data
1 timestamp;type_evenement;source;message
64 2025-08-28 10:03:29;lecture_ok;cmd_revendeur_test_duplicata.csv;2 lignes a traiter
65 2025-08-28 10:03:29;lecture_ok;commande_revendeur_tech_express_fichier_doublon.csv;5 lignes a traiter
66 2025-08-28 10:03:29;lecture_ok;cmd_revendeur_test_id_pdt_inexistant.csv;2 lignes a traiter
67 2025-08-28 10:03:29;structure;cmd_revendeur_test_colonne_manquante.csv;Colonnes manquantes: ['date', 'num_cmd']
68 2025-08-28 10:03:29;lecture_ok;cmd_revendeur_test_autre_revendeur.csv;2 lignes a traiter
69 2025-08-28 10:03:29;lecture_ok;cmd_revendeur_test_id_revendeur_vide.csv;2 lignes a traiter
70 2025-08-28 10:03:29;lecture_ok;cmd_revendeur_test_date_incoherente.csv;2 lignes a traiter
71 2025-08-28 10:03:29;lecture_ok;cmd_revendeur_test_qte_neg.csv;2 lignes a traiter
72 2025-08-28 10:03:29;lecture_ok;cmd_revendeur_test_prix_vide.csv;2 lignes a traiter
73 2025-08-28 10:03:29;lecture_ok;cmd_revendeur_test_valide_alerte_stock.csv;3 lignes a traiter
74 2025-08-28 10:03:29;lecture_ok;cmd_revendeur_test_qte_non_num.csv;2 lignes a traiter
75 2025-08-28 10:03:29;sqlite_ok;produit;0 lignes a traiter
76 2025-08-28 10:03:29;sqlite_ok;region;0 lignes a traiter
77 2025-08-28 10:03:29;sqlite_ok;revendeur;0 lignes a traiter
78 2025-08-28 10:03:29;sqlite_ok;production;0 lignes a traiter
79 2025-08-28 10:03:29;extract_ok;commandes;26 lignes
80 2025-08-28 10:03:29;rename_ok;global;Tables et colonnes renommées
81 2025-08-28 10:03:29;format;cmd_revendeur_test_date_incoherente.csv;Ligne 1 : valeurs NaN apres correction dans 'date'
82 2025-08-28 10:03:29;format;cmd_revendeur_test_qte_non_num.csv;Ligne 1 : valeurs NaN apres correction dans 'quantite'
83 2025-08-28 10:03:29;format;cmd_revendeur_test_prix_vide.csv;Ligne 1 : valeurs NaN apres correction dans 'prix_unitaire'
84 2025-08-28 10:03:29;valeur_interdite;cmd_revendeur_test_date_future.csv;Ligne 1 rejete (valeur interdite)
85 2025-08-28 10:03:29;valeur_interdite;cmd_revendeur_test_id_revendeur_vide.csv;Ligne 1 rejete (valeur interdite)
86 2025-08-28 10:03:29;valeur_interdite;cmd_revendeur_test_date_incoherente.csv;Ligne 1 rejete (valeur interdite)
87 2025-08-28 10:03:29;valeur_interdite;cmd_revendeur_test_qte_neg.csv;Ligne 1 rejete (valeur interdite)
88 2025-08-28 10:03:29;valeur_interdite;cmd_revendeur_test_prix_vide.csv;Ligne 1 rejete (valeur interdite)
89 2025-08-28 10:03:29;valeur_interdite;cmd_revendeur_test_qte_non_num.csv;Ligne 1 rejete (valeur interdite)
90 2025-08-28 10:03:29;doublon_bdd;commande_revendeur_tech_express_fichier_doublon.csv;Ligne 1 supprimee (cle deja presente en base sur num_cmd)
91 2025-08-28 10:03:29;doublon_bdd;commande_revendeur_tech_express_fichier_doublon.csv;Ligne 2 supprimee (cle deja presente en base sur num_cmd)
92 2025-08-28 10:03:29;doublon_bdd;commande_revendeur_tech_express_fichier_doublon.csv;Ligne 3 supprimee (cle deja presente en base sur num_cmd)
93 2025-08-28 10:03:29;doublon_bdd;commande_revendeur_tech_express_fichier_doublon.csv;Ligne 4 supprimee (cle deja presente en base sur num_cmd)
94 2025-08-28 10:03:29;doublon_bdd;commande_revendeur_tech_express_fichier_doublon.csv;Ligne 5 supprimee (cle deja presente en base sur num_cmd)
95 2025-08-28 10:03:29;doublon_strict;cmd_revendeur_test_duplicata.csv;Ligne 2 strictement identique supprimee
96 2025-08-28 10:03:29;cle_etrangere;cmd_revendeur_test_autre_revendeur.csv;Ligne 1 rejete (id_revendeur absent de revendeurs)
97 2025-08-28 10:03:29;cle_etrangere;cmd_revendeur_test_id_pdt_inexistant.csv;Ligne 1 rejete (id_pdt absent de produits)
98 2025-08-28 10:03:29;transform_ok;commandes;12 lignes nettoyees
99 2025-08-28 10:03:30;load_ok;commandes;11 lignes inserees
100 2025-08-28 10:03:30;load_ok;lignes_cmd;12 lignes inserees
101 2025-08-28 10:03:30;post_etl;stock;1 produit(s) avec stock negatif
102 2025-08-28 10:03:30;post_etl;global;Vues mises a jour et CSV stock genere
103
```

## Critère d'acceptation global :

Le pipeline ETL est accepté si, pour l'ensemble des jeux de test, les comportements observés correspondent aux résultats attendus (intégration des données valides, rejet tracé des données invalides, calcul correct du stock, export CSV généré, logs complets et lisibles).

## 6. Conclusion

Le projet *Distributech* a permis de concevoir et de valider un pipeline ETL fonctionnel, capable d'intégrer des données issues de différentes sources tout en garantissant leur cohérence.

Les tests effectués selon les critères d'acceptation montrent que la solution répond aux besoins et a montré un comportement fiable lors des jeux de test réalisés.

Ce résultat constitue une base solide pour une utilisation opérationnelle, tout en laissant la possibilité d'évolutions futures (optimisation des performances, élargissement des jeux de test).