

Projet : Transfromation Données – Conception Pipelines

1. Contexte et Objectif

Ce projet a été réalisé dans le cadre d'un TP de l'unité Conception de Pipelines à l'EPSI. L'objectif principal est de concevoir un pipeline ETL (Extract – Transform – Load) simple et robuste en Python afin de traiter un fichier de ventes brutes, le nettoyer, le transformer et l'enrichir pour l'analyse.

2. Technologies utilisées

- Python 3
- Bibliothèques : pandas, numpy, logging
- Environnement : Visual Studio Code
- Contrôle de version : Git + GitHub

3. Étapes de traitement

Voici les étapes principales du traitement réalisé :

- Chargement des données brutes depuis le fichier CSV 'ventes.csv'.
- Nettoyage des données : suppression des doublons, gestion des valeurs manquantes, suppression des lignes incohérentes (ex: quantités nulles, dates invalides).
- Validation : conversion du format de date, vérification des types et de la cohérence des prix et quantités.
- Transformation : création de la colonne 'Montant_total' (prix * quantité) puis normalisation de cette colonne.
- Enrichissement : ajout d'un segment client ('Basique', 'Standard', 'Premium') selon le montant total.
- Gestion des erreurs : simulation d'un traitement à risque avec gestion d'erreurs enregistrées dans 'logs/erreurs.log'.
- Export : sauvegarde finale dans 'ventes_transformees.csv'.

```

PS C:\Users\ilias\Desktop\EPSI\Conception Pipelines\Projet_Transformation_ConceptionPipelines> python main.py
Aperçu des 5 premières lignes :
   ID_produit  Nom_produit  Quantite_vendue  Prix_unitaire  Date_vente
0           1    Chemise           10.0           25.0  2022-01-05
1           2   Pantalon            8.0           35.0  2022-01-06
2           3  Chaussures           NaN           50.0  2022-01-07
3           4   Cravate           12.0           15.0  2022-01-08
4           5    Robe            15.0           45.0  2022-01-09

Infos sur les colonnes :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5266 entries, 0 to 5265
Data columns (total 5 columns):

```

```

RangeIndex: 5266 entries, 0 to 5265
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   ID_produit            5266 non-null   int64
1   Nom_produit           5255 non-null   object
2   Quantite_vendue       2594 non-null   float64
3   Prix_unitaire         2697 non-null   float64
4   Date_vente            5266 non-null   object
dtypes: float64(2), int64(1), object(2)
memory usage: 205.8+ KB
None

```

```

Statistiques :
   ID_produit  Nom_produit  Quantite_vendue  Prix_unitaire  Date_vente
count  5266.000000      5255      2594.000000    2697.000000      5266
unique      NaN            8            NaN            NaN      1335
top      NaN    Pantalon            NaN            NaN  invalid_date
freq      NaN            709            NaN            NaN      1679
mean    50.700152      NaN    10.811295    54.122903      NaN
std     29.088180      NaN    5.596689    25.663199      NaN
min      1.000000      NaN    0.000000    10.040000      NaN
25%     26.000000      NaN    5.900000    31.830000      NaN
50%     51.000000      NaN   10.900000    52.970000      NaN
75%     76.000000      NaN   15.700000    75.920000      NaN
max    100.000000      NaN   25.000000   99.930000      NaN

```

--- Nettoyage des données ---

```

Données nettoyées (5 premières lignes) :
   ID_produit  Nom_produit  Quantite_vendue  Prix_unitaire  Date_vente
0           1    Chemise           10.00000           25.0  2022-01-05
1           2   Pantalon            8.00000           35.0  2022-01-06
2           3  Chaussures          10.83086           50.0  2022-01-07
3           4   Cravate           12.00000           15.0  2022-01-08
4           5    Robe            15.00000           45.0  2022-01-09

```

```
Valeurs manquantes restantes :
ID_produit      0
Nom_produit     0
Quantite_vendue 0
Prix_unitaire   0
Date_vente      0
dtype: int64
```

```
--- Validation des données ---
✗ Erreur lors de la conversion des dates : time data "2022-" doesn't match format "%Y-%m-%d", at position 50. You might want to try:
  - passing 'format' if your strings have a consistent format;
  - passing 'format="%Y-%m-%d"' if your strings are all ISO8601 but not necessarily in exactly the same format;
  - passing 'format="mixed"', and the format will be inferred for each element individually. You might want to use 'dayfirst' alongside this.
✓ Prix et quantités validés

--- Transformation des données ---
✓ Colonne Montant_total ajoutée
✓ Colonne Montant_normalise ajoutée (normalisation Min-Max)
```

```
Aperçu final avec transformations :
```

	Nom_produit	Quantite_vendue	Prix_unitaire	Montant_total	Montant_normalise
0	Chemise	10.00000	25.0	250.000000	0.120447
1	Pantalon	8.00000	35.0	280.000000	0.135797
2	Chaussures	10.83086	50.0	541.543002	0.269627
3	Cravate	12.00000	15.0	180.000000	0.084628
4	Robe	15.00000	45.0	675.000000	0.337916

```

--- Enrichissement des données : segmentation ---
✓ Colonne Segment ajoutée
```

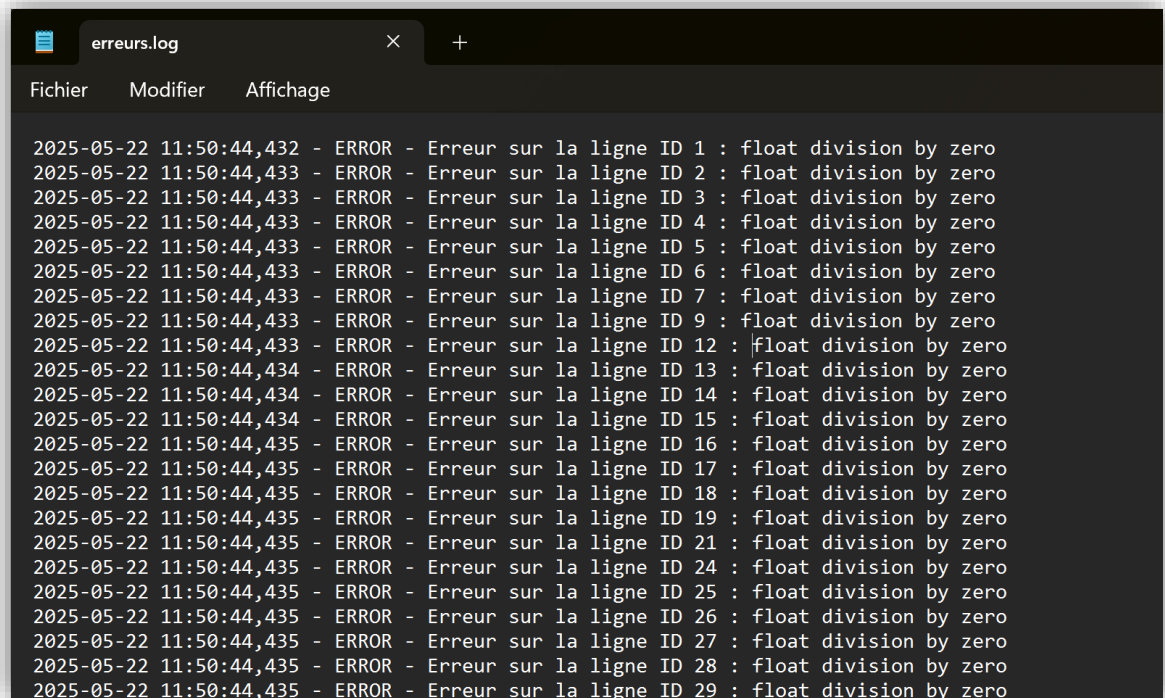
```
Aperçu avec le segment :
```

	Nom_produit	Montant_total	Segment
0	Chemise	250.000000	Standard
1	Pantalon	280.000000	Standard
2	Chaussures	541.543002	Premium
3	Cravate	180.000000	Basique
4	Robe	675.000000	Premium

```

--- Gestion des erreurs ---
✓ Erreurs capturées et enregistrées dans logs/erreurs.log

--- Export des données transformées ---
✓ Données exportées dans data/ventes_transformees.csv
```



```
erreurs.log
Fichier  Modifier  Affichage

2025-05-22 11:50:44,432 - ERROR - Erreur sur la ligne ID 1 : float division by zero
2025-05-22 11:50:44,433 - ERROR - Erreur sur la ligne ID 2 : float division by zero
2025-05-22 11:50:44,433 - ERROR - Erreur sur la ligne ID 3 : float division by zero
2025-05-22 11:50:44,433 - ERROR - Erreur sur la ligne ID 4 : float division by zero
2025-05-22 11:50:44,433 - ERROR - Erreur sur la ligne ID 5 : float division by zero
2025-05-22 11:50:44,433 - ERROR - Erreur sur la ligne ID 6 : float division by zero
2025-05-22 11:50:44,433 - ERROR - Erreur sur la ligne ID 7 : float division by zero
2025-05-22 11:50:44,433 - ERROR - Erreur sur la ligne ID 9 : float division by zero
2025-05-22 11:50:44,433 - ERROR - Erreur sur la ligne ID 12 : float division by zero
2025-05-22 11:50:44,434 - ERROR - Erreur sur la ligne ID 13 : float division by zero
2025-05-22 11:50:44,434 - ERROR - Erreur sur la ligne ID 14 : float division by zero
2025-05-22 11:50:44,434 - ERROR - Erreur sur la ligne ID 15 : float division by zero
2025-05-22 11:50:44,435 - ERROR - Erreur sur la ligne ID 16 : float division by zero
2025-05-22 11:50:44,435 - ERROR - Erreur sur la ligne ID 17 : float division by zero
2025-05-22 11:50:44,435 - ERROR - Erreur sur la ligne ID 18 : float division by zero
2025-05-22 11:50:44,435 - ERROR - Erreur sur la ligne ID 19 : float division by zero
2025-05-22 11:50:44,435 - ERROR - Erreur sur la ligne ID 21 : float division by zero
2025-05-22 11:50:44,435 - ERROR - Erreur sur la ligne ID 24 : float division by zero
2025-05-22 11:50:44,435 - ERROR - Erreur sur la ligne ID 25 : float division by zero
2025-05-22 11:50:44,435 - ERROR - Erreur sur la ligne ID 26 : float division by zero
2025-05-22 11:50:44,435 - ERROR - Erreur sur la ligne ID 27 : float division by zero
2025-05-22 11:50:44,435 - ERROR - Erreur sur la ligne ID 28 : float division by zero
2025-05-22 11:50:44,435 - ERROR - Erreur sur la ligne ID 29 : float division by zero
```

4. Structure du projet

Projet_Transformation_ConceptionPipelines/

├── data/

│ ├── ventes.csv

│ └── ventes_transformees.csv

├── historique/

│ └── ventes_transformees_YYYYMMDD_HHMMSS.csv

├── logs/

│ └── erreurs.log

└── main.py

5. Historisation des données

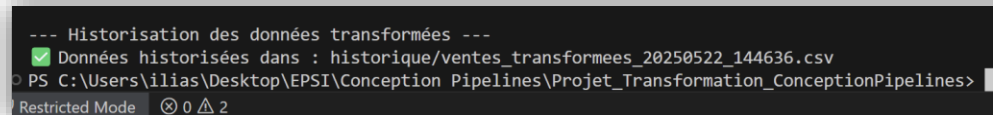
Dans la continuité du pipeline de transformation, une fonctionnalité d'historisation a été ajoutée. Cette étape permet de conserver une trace de chaque traitement réalisé en sauvegardant les données transformées dans un fichier CSV plat, horodaté automatiquement.

Chaque exécution du script Python génère un fichier de sortie unique dans le dossier `historique/` avec une date et heure au format `YYYYMMDD_HHMMSS`, comme dans l'exemple suivant :

[historique/ventes_transformees_20250522_144636.csv](#)

Cette approche permet de :

- Conserver un historique de toutes les versions des données
- Revenir à un état précédent si nécessaire
- Tracer précisément l'évolution des transformations dans le temps



```
--- Historisation des données transformées ---
✓ Données historisées dans : historique/ventes_transformees_20250522_144636.csv
PS C:\Users\ilias\Desktop\EPSI\Conception Pipelines\Projet_Transformation_ConceptionPipelines>
```

6. Conclusion

Ce projet a permis de mettre en pratique les étapes fondamentales d'un pipeline de données en Python. Il illustre les bonnes pratiques de traitement, de contrôle qualité et d'export de données. Le code est versionné sur GitHub et prêt à être enrichi par des visualisations ou une connexion à une base de données.