- Ould taleb dyhia
- Baziz ilhem
- Sarr Babacar

Exercice-Conception-Pipelines-T-pour-Transfor

1. Analyse des données

On commence par charger les données CSV dans un DataFrame et en faire une première analyse :

- Colonnes :
  - ID_produit : identifiant unique
  - Nom_produit : nom du produit
  - Quantite_vendue : nombre d'unités vendues
  - Prix_unitaire : prix d'une unité
  - Date_vente : date de vente
- Problèmes repérés :
  - Valeurs manquantes (Quantite_vendue)
  - Potentiels doublons
  - Valeurs aberrantes possibles dans Quantite_vendue ou Prix_unitaire (ex: quantités négatives, prix très élevés)

2. Suppression des doublons

```
# Suppression des doublons
df.drop_duplicates(inplace=True) # supprime les lignes qui sont exactement identiques
# et applique la modification directement.
print(df.shape) # pour voir combien de lignes qui on etait supprimer car c'est des doublons
```

3. Traitement des valeurs manquantes

```
# Supprimer les lignes où Quantite_vendue est manquante (optionnel mais
recommandé)
df = df[df['Quantite_vendue'].notna()]

# Convertir la colonne Quantite_vendue de type entier car on peut pas vendre
la moitié de quelque chose
df['Quantite_vendue'] = df['Quantite_vendue'].astype(int)

print("Le nombre de NaN dans la colonne 'Quantite_vendue' est :
{}".format(df['Quantite_vendue'].isna().sum()))
```

4. Gestion des valeurs aberrantes

```
# 4. Gestion des valeurs aberrantes
def winsorize_series(series):
    q1 = series.quantile(0.25)
    q3 = series.quantile(0.75)
    iqr = q3 - q1
    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr
    return series.clip(lower=lower, upper=upper)

df['Quantite_vendue'] = winsorize_series(df['Quantite_vendue'])
df['Prix_unitaire'] = winsorize_series(df['Prix_unitaire'])
```

5. Validation des données

```
# 5. Validation des données
{
    "doublons_restants": df.duplicated().sum(),
    "valeurs_manquantes": df.isna().sum(),
    "quantite_negative": (df['Quantite_vendue'] < 0).sum(),
    "prix_negative": (df['Prix_unitaire'] < 0).sum()
}
```

6. Gestion des erreurs

```
# 6. Gestion des erreurs (try/except)
try:
    df['Valeur_totale'] = df['Quantite_vendue'] * df['Prix_unitaire']
```

```
except Exception as e:
    logging.error(f"Erreur lors du calcul de la colonne 'Valeur_totale': {e}")
```

7. Transformation des données (minimum 2 transformations)

```
# 7. Transformations
# Ajout d'une colonne calculée :
df['Montant_total'] = df['Quantite_vendue'] * df['Prix_unitaire']
# 7. Transformations
# Normalisation Min-Max
montant_min = df['Montant_total'].min()
montant_max = df['Montant_total'].max()

df['Montant_total_normalise'] = (df['Montant_total'] - montant_min) /
(montant_max - montant_min)

# 7. Transformations
# Agrégation des quantités vendues par produit
df_agg =
df.groupby("Nom_produit")["Valeur_totale"].sum().reset_index().rename(columns=
{"Valeur_totale": "Vente_totale"})
```

8. Documentation

```
{'Traitements effectués': ['Suppression des doublons',
  'Imputation des valeurs manquantes avec la médiane',
  'Winsorisation des valeurs aberrantes',
  "Ajout de la colonne 'Valeur_totale'",
  'Agrégation des ventes par produit'],
 'Validation finale': {'doublons_restants': np.int64(0),
  'valeurs_manquantes': ID_produit                0
  Nom_produit               0
  Quantite_vendue           0
  Prix_unitaire             0
  Date_vente                0
  Valeur_totale             0
  Montant_total             0
  Montant_total_normalise   0
  dtype: int64,
  'quantite_negative': np.int64(0),
  'prix_negative': np.int64(0)}}
```

```
   ID_produit Nom_produit  Quantite_vendue  Prix_unitaire  Date_vente  \
0           1     Chemise               10           25.0  2022-01-05
1           2    Pantalon                8           35.0  2022-01-06
3           4     Cravate               12           15.0  2022-01-08
4           5        Robe               15           45.0  2022-01-09
6           7    Pantalon                8           35.0  2022-01-06

   Valeur_totale  Montant_total  Montant_total_normalise
0          250.0          250.0                 0.132627
1          280.0          280.0                 0.148542
3          180.0          180.0                 0.095491
4          675.0          675.0                 0.358092
6          280.0          280.0                 0.148542
```

Exercice-Conception-Pipelines-L-pour-Load

Ajouter la partie sauvegarde des donnees :

```python
import os
from datetime import datetime

# Créer un dossier de sauvegarde s'il n'existe pas
os.makedirs("archives", exist_ok=True)

# Générer un nom de fichier avec la date et l'heure actuelles
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
nom_fichier = f"sauvegardes/donnees_nettoyees_{timestamp}.csv"

# Sauvegarde du DataFrame dans un fichier CSV
df.to_csv(nom_fichier, index=False)

print(f"Fichier sauvegardé : {nom_fichier}")
```