Projet 11 Etude de marché pour la poule qui chante

OBJECTIFS DE CE NOTEBOOK

- Nettoyer et préparer les différents jeux de données

- Réaliser les jointures entre les fichiers

- Constituer un dataset unifié contenant les variables essentielles pour l'étude de marché

Sommaire

- Étape 1 - Importation des librairies et chargement des fichiers
  - 1.1 - Importation des librairies
  - 1.2 - Chargement des fichiers
- Etape 2 - Analyse exploratoire des fichiers
  - 2.1 - Analyse exploratoire du fichier stabilite_politique
  - 2.2 - Analyse exploratoire du fichier dispo_alimentaire
  - 2.3 - Analyse exploratoire du fichier PIB_habitant
  - 2.4 - Analyse exploratoire du fichier population
  - 2.5 - performance_logistique
  - 2.6 - Taux_droit_douane
- Étape 3 - Jointure des fichiers
  - 3.1 - Jointure
  - 3.2 - Finalisation du fichier final
  - 3.3 - Export du fichier final

Étape 1 - Importation des librairies et chargement des fichiers

1.1 - Importation des librairies

In [6]:
```
#Importation des librairies
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [7]:
```
!pip install xlrd
```

```
Requirement already satisfied: xlrd in c:\users\pc\anaconda3\lib\site-packages
(2.0.2)
```

In [8]:
```
#Importation de la librairie plotly express
!pip install plotly
import plotly.express as px
```

```
Requirement already satisfied: plotly in c:\users\pc\anaconda3\lib\site-packages
(5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\pc\anaconda3\lib\site-
packages (from plotly) (8.2.3)
Requirement already satisfied: packaging in c:\users\pc\anaconda3\lib\site-packag
es (from plotly) (24.1)
```

1.2 - Chargement des fichiers

In [10]:
```
#Importation du fichier Dispo_alimentaire_2017
dispo_alimentaire = pd.read_csv("Dispo_alimentaire_2017.csv")
#Importation du fichier Population_2000_2018
population_2000_2018 = pd.read_csv("Population_2000_2018.csv")
#Importation du fichier PIB_habitant
PIB_habitant = pd.read_csv("PIB_habitant.csv", skiprows=4)
#Importation du fichier stabilite_politique
stabilite_politique = pd.read_csv("stabilite_politique.csv", skiprows=4)
#Importation du fichier performance_logistique
performance_logistique = pd.read_csv("performance_logistique.csv", skiprows=4)
#Importation du fichier Taux_droit_douane
Taux_droit_douane = pd.read_excel("Taux_droit_douane.xls", skiprows=3)
```

Étape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier stabilite_politique

In [13]:
```
# Traitement du fichier stabilite_politique
# Restructuration du format
stabilite_politique = stabilite_politique.melt(
    id_vars=['Country Name', 'Country Code'],
    var_name='annee',
    value_name='stabilite'
)
# Renommage et nettoyage
stabilite_politique.rename(columns={'Country Name': 'pays'}, inplace=True)

# Convertir les années en numérique
stabilite_politique['annee'] = pd.to_numeric(stabilite_politique['annee'], error

# Convertir la stabilité politique en nombre (et ignorer les valeurs texte)
stabilite_politique['stabilite'] = pd.to_numeric(stabilite_politique['stabilite'

# Supprimer les lignes où la valeur est manquante
stabilite_politique = stabilite_politique.dropna(subset=['stabilite'])

# Convertir en entier
stabilite_politique['stabilite'] = stabilite_politique['stabilite'].round(0).ast
```

```
# Affichage des 5 premières lignes
stabilite_politique
```

Out[13]:

	pays	Country Code	annee	stabilite
10110	Afghanistan	AFG	1996.0	-2
10112	Angola	AGO	1996.0	-2
10113	Albanie	ALB	1996.0	0
10114	Andorre	AND	1996.0	1
10116	Émirats arabes unis	ARE	1996.0	1
...	...	...	...	...
17551	Kosovo	XKX	2023.0	0
17552	Yémen, Rép. du	YEM	2023.0	-3
17553	Afrique du Sud	ZAF	2023.0	-1
17554	Zambie	ZMB	2023.0	0
17555	Zimbabwe	ZWE	2023.0	-1

5025 rows × 4 columns

In [14]:
```
# Changement du type de la colonne stabilite
stabilite_politique['stabilite'] = stabilite_politique['stabilite'].round(0).ast

#Changement du type de la colonne annee
stabilite_politique['annee'] = stabilite_politique['annee'].astype(int)
stabilite_politique.head()
```

Out[14]:

	pays	Country Code	annee	stabilite
10110	Afghanistan	AFG	1996	-2
10112	Angola	AGO	1996	-2
10113	Albanie	ALB	1996	0
10114	Andorre	AND	1996	1
10116	Émirats arabes unis	ARE	1996	1

In [15]:
```
# Garder uniquement l'année 2022
stabilite_politique = stabilite_politique[
    (stabilite_politique['annee'] == 2022)
]
```

In [16]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(stabilite_pol
print("Le tableau comporte {} colonne(s)".format(stabilite_politique.shape[1]))
```

```
Le tableau comporte 205 observation(s) ou article(s)
Le tableau comporte 4 colonne(s)
```

In [17]:
```
#Vérifiaction des valeurs manquantes
stabilite_politique.isna().any()
```

Out[17]:
```
pays            False
Country Code    False
annee           False
stabilite       False
dtype: bool
```

Il n'y a pas de valeurs manquantes dans stabilite_politique

In [19]:
```
#Vérification des doublons sur toutes les colonnes
stabilite_politique_doublon = stabilite_politique[stabilite_politique.duplicated

# Afficher les doublons (si existants)
print("Nombre de doublons :", len(stabilite_politique_doublon))
```

```
Nombre de doublons : 0
```

Il n'y a pas de doublons dans stabilite_politique

In [21]:
```
#Suppression de la colonne Country Code
stabilite_politique = stabilite_politique.drop(columns=["Country Code"])
```

In [22]:
```
#Suppression de l'annee
stabilite_politique.drop(columns=["annee"], inplace=True)
```

In [23]:
```
#Affichage de la table
stabilite_politique
```

Out[23]:

	pays	stabilite
17024	Aruba	1
17026	Afghanistan	-3
17028	Angola	-1
17029	Albanie	0
17030	Andorre	2
...	...	...
17285	Kosovo	0
17286	Yémen, Rép. du	-2
17287	Afrique du Sud	-1
17288	Zambie	0
17289	Zimbabwe	-1

205 rows × 2 columns

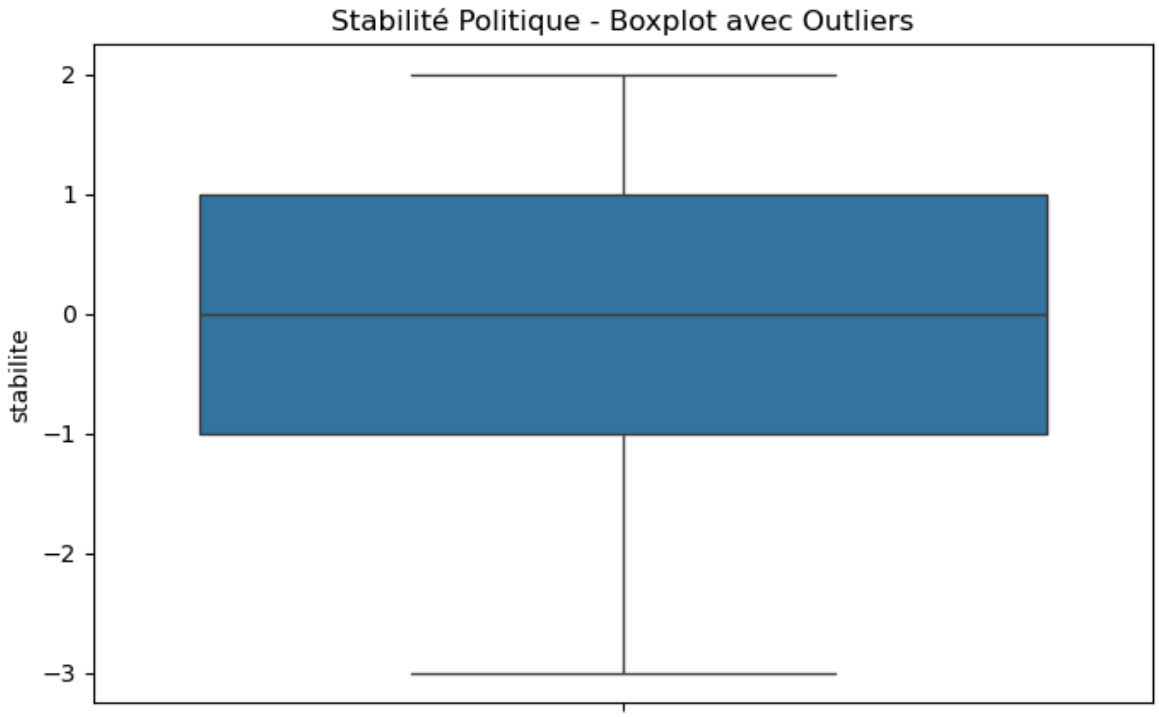
Les outliers

In [25]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Boxplot simple avec outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=stabilite_politique['stabilite'])
plt.title('Stabilité Politique - Boxplot avec Outliers')
plt.show()

# Compter les outliers
Q1 = stabilite_politique['stabilite'].quantile(0.25)
Q3 = stabilite_politique['stabilite'].quantile(0.75)
IQR = Q3 - Q1
outliers = stabilite_politique[(stabilite_politique['stabilite'] < Q1 - 1.5*IQR)
                               (stabilite_politique['stabilite'] > Q3 + 1.5*IQR)]

print(f"Outliers détectés : {len(outliers)}")
print(outliers['stabilite'])
```




```
Outliers détectés : 0
Series([], Name: stabilite, dtype: int32)
```

Il n'y a pas d'outliers

2.2 - Analyse exploratoire du fichier dispo_alimentaire

In [28]:
```
#Affichage de la table
dispo_alimentaire
```

Out[28]:

	Code Domaine	Domaine	Code zone	Zone	Code Élément	Élément	Code Produit
0	FBS	Nouveaux Bilans Alimentaire	2	Afghanistan	5511	Production	2511
1	FBS	Nouveaux Bilans Alimentaire	2	Afghanistan	5611	Importations - Quantité	2511
2	FBS	Nouveaux Bilans Alimentaire	2	Afghanistan	5072	Variation de stock	2511
3	FBS	Nouveaux Bilans Alimentaire	2	Afghanistan	5911	Exportations - Quantité	2511
4	FBS	Nouveaux Bilans Alimentaire	2	Afghanistan	5301	Disponibilité intérieure	2511
...	...	...	...	...	...	...	...
176595	FBS	Nouveaux Bilans Alimentaire	181	Zimbabwe	5142	Nourriture	2899
176596	FBS	Nouveaux Bilans Alimentaire	181	Zimbabwe	645	Disponibilité alimentaire en quantité (kg/pers...	2899
176597	FBS	Nouveaux Bilans Alimentaire	181	Zimbabwe	664	Disponibilité alimentaire (Kcal/personne/jour)	2899
176598	FBS	Nouveaux Bilans Alimentaire	181	Zimbabwe	674	Disponibilité de protéines en quantité (g/pers...	2899
176599	FBS	Nouveaux Bilans Alimentaire	181	Zimbabwe	684	Disponibilité de matière grasse en quantité (g...	2899

176600 rows × 14 columns

In [29]:
```
#Affichage des elements contenu dans la colonne Elements de dispo_alimentaire
dispo_alimentaire['Élément'].unique()
```

Out[29]:
```
array(['Production', 'Importations - Quantité', 'Variation de stock',
       'Exportations - Quantité', 'Disponibilité intérieure',
       'Aliments pour animaux', 'Semences', 'Pertes', 'Résidus',
       'Nourriture',
       'Disponibilité alimentaire en quantité (kg/personne/an)',
       'Disponibilité alimentaire (Kcal/personne/jour)',
       'Disponibilité de protéines en quantité (g/personne/jour)',
       'Disponibilité de matière grasse en quantité (g/personne/jour)',
       'Traitement', 'Autres utilisations (non alimentaire)',
       'Alimentation pour touristes'], dtype=object)
```

In [30]:
```
#Affichage des elements contenu dans la colonne Produit de dispo_alimentaire
dispo_alimentaire['Produit'].unique()
```

Out[30]:
```
array(['Blé et produits', 'Riz et produits', 'Orge et produits',
       'Maïs et produits', 'Seigle et produits', 'Avoine',
       'Millet et produits', 'Sorgho et produits', 'Céréales, Autres',
       'Pommes de Terre et produits', 'Ignames', 'Racines nda',
       'Sucre, canne', 'Sucre, betterave', 'Sucre Eq Brut',
       'Edulcorants Autres', 'Miel', 'Haricots', 'Pois',
       'Légumineuses Autres et produits', 'Noix et produits', 'Soja',
       'Arachides Decortiquees', 'Graines de tournesol',
       'Graines Colza/Moutarde', 'Graines de coton', 'Coco (Incl Coprah)',
       'Sésame', 'Olives', 'Plantes Oleiferes, Autre', 'Huile de Soja',
       "Huile d'Arachide", 'Huile de Tournesol',
       'Huile de Colza&Moutarde', 'Huile Graines de Coton',
       'Huile de Palmistes', 'Huile de Palme', 'Huile de Coco',
       'Huile de Sésame', "Huile d'Olive", 'Huile de Son de Riz',
       'Huile de Germe de Maïs', 'Huil Plantes Oleif Autr',
       'Tomates et produits', 'Oignons', 'Légumes, Autres',
       'Oranges, Mandarines', 'Citrons & Limes et produits',
       'Pamplemousse et produits', 'Agrumes, Autres', 'Bananes',
       'Pommes et produits', 'Ananas et produits', 'Dattes', 'Raisin',
       'Fruits, Autres', 'Café et produits', 'Feve de Cacao et produits',
       'Thé', 'Poivre', 'Piments', 'Girofles', 'Épices, Autres', 'Vin',
       'Bière', 'Boissons Fermentés', 'Boissons Alcooliques',
       'Alcool, non Comestible', 'Viande de Bovins',
       "Viande d'Ovins/Caprins", 'Viande de Suides',
       'Viande de Volailles', 'Viande, Autre', 'Abats Comestible',
       'Beurre, Ghee', 'Crème', 'Graisses Animales Crue', 'Oeufs',
       'Lait - Excl Beurre', 'Poissons Eau Douce',
       'Aliments pour enfants', 'Miscellanees', 'Manioc et produits',
       'Patates douces', 'Palmistes', 'Bananes plantains',
       'Huiles de Poissons', 'Huiles de Foie de Poisso', 'Perciform',
       'Poissons Pelagiques', 'Poissons Marins, Autres', 'Crustacés',
       'Cephalopodes', 'Mollusques, Autres', 'Animaux Aquatiques Autre',
       'Plantes Aquatiques', 'Sucre non centrifugé',
       'Viande de Anim Aquatiq'], dtype=object)
```

In [31]:
```
# Supprimer les doublons de colonnes inutiles
dispo_alimentaire = dispo_alimentaire.drop(columns=[
    "Code Domaine",
    "Code zone",
    "Code Élément",
    "Code Produit",
    "Code année",
    "Symbole",
    "Description du Symbole"
])
```

In [32]:
```
# Renommer les colonnes
dispo_alimentaire = dispo_alimentaire.rename(columns={
    "Domaine": "domaine",
    "Zone": "pays",
    "Élément": "element",
    "Produit": "produit",
    "Année": "annee",
    "Unité": "unite",
    "Valeur": "valeur"
})
```

```
dispo_alimentaire.head()
```

Out[32]:

	domaine	pays	element	produit	annee	unite	valeur
0	Nouveaux Bilans Alimentaire	Afghanistan	Production	Blé et produits	2017	Milliers de tonnes	4281.0
1	Nouveaux Bilans Alimentaire	Afghanistan	Importations - Quantité	Blé et produits	2017	Milliers de tonnes	2302.0
2	Nouveaux Bilans Alimentaire	Afghanistan	Variation de stock	Blé et produits	2017	Milliers de tonnes	-119.0
3	Nouveaux Bilans Alimentaire	Afghanistan	Exportations - Quantité	Blé et produits	2017	Milliers de tonnes	0.0
4	Nouveaux Bilans Alimentaire	Afghanistan	Disponibilité intérieure	Blé et produits	2017	Milliers de tonnes	6701.0

In [33]:
```
#Selection de la viande de volailles
dispo_alimentaire = dispo_alimentaire[dispo_alimentaire["produit"] == "Viande de
dispo_alimentaire.head()
```

Out[33]:

	domaine	pays	element	produit	annee	unite	valeur
651	Nouveaux Bilans Alimentaire	Afghanistan	Production	Viande de Volailles	2017	Milliers de tonnes	28.0
652	Nouveaux Bilans Alimentaire	Afghanistan	Importations - Quantité	Viande de Volailles	2017	Milliers de tonnes	29.0
653	Nouveaux Bilans Alimentaire	Afghanistan	Variation de stock	Viande de Volailles	2017	Milliers de tonnes	0.0
654	Nouveaux Bilans Alimentaire	Afghanistan	Disponibilité intérieure	Viande de Volailles	2017	Milliers de tonnes	57.0
655	Nouveaux Bilans Alimentaire	Afghanistan	Pertes	Viande de Volailles	2017	Milliers de tonnes	2.0

In [34]:
```
# Changement du type des colonnes
dispo_alimentaire["annee"] = pd.to_numeric(dispo_alimentaire["annee"], errors="c
dispo_alimentaire["valeur"] = pd.to_numeric(dispo_alimentaire["valeur"], errors=
dispo_alimentaire.head()
```

Out[34]:

	domaine	pays	element	produit	annee	unite	valeur
651	Nouveaux Bilans Alimentaire	Afghanistan	Production	Viande de Volailles	2017	Milliers de tonnes	28.0
652	Nouveaux Bilans Alimentaire	Afghanistan	Importations - Quantité	Viande de Volailles	2017	Milliers de tonnes	29.0
653	Nouveaux Bilans Alimentaire	Afghanistan	Variation de stock	Viande de Volailles	2017	Milliers de tonnes	0.0
654	Nouveaux Bilans Alimentaire	Afghanistan	Disponibilité intérieure	Viande de Volailles	2017	Milliers de tonnes	57.0
655	Nouveaux Bilans Alimentaire	Afghanistan	Pertes	Viande de Volailles	2017	Milliers de tonnes	2.0

In [35]:

```
# Pivot de la table pour créer des nouvelles colonnes
dispo_alimentaire = dispo_alimentaire.pivot_table(
    index=["pays", "produit", "annee"],
    columns="element",
    values="valeur",
    aggfunc="sum"
).reset_index()

# Renommer les colonnes d'intérêt
dispo_alimentaire = dispo_alimentaire.rename(columns={
    "Production": "production",
    "Importations - Quantité": "importation_qte",
    "Disponibilité intérieure": "disponibilite_interieure_qte"
})

# Garder uniquement les colonnes utiles
dispo_alimentaire = dispo_alimentaire[[
    "pays", "produit", "annee",
    "production", "importation_qte", "disponibilite_interieure_qte"
]]

# Vérifier le résultat
dispo_alimentaire.head()
```

Out[35]:

	element	pays	produit	annee	production	importation_qte	disponibilite_interie
0		Afghanistan	Viande de Volailles	2017	28.0	29.0	
1		Afrique du Sud	Viande de Volailles	2017	1667.0	514.0	
2		Albanie	Viande de Volailles	2017	13.0	38.0	
3		Algérie	Viande de Volailles	2017	275.0	2.0	
4		Allemagne	Viande de Volailles	2017	1514.0	842.0	

In [36]:
```
#Affichage des pays contenus dans la colonne pays
dispo_alimentaire['pays'].unique()
```

Out[36]:
```
array(['Afghanistan', 'Afrique du Sud', 'Albanie', 'Algérie', 'Allemagne',
       'Angola', 'Antigua-et-Barbuda', 'Arabie saoudite', 'Argentine',
       'Arménie', 'Australie', 'Autriche', 'Azerbaïdjan', 'Bahamas',
       'Bangladesh', 'Barbade', 'Belgique', 'Belize',
       'Bolivie (État plurinational de)', 'Bosnie-Herzégovine',
       'Botswana', 'Brésil', 'Bulgarie', 'Burkina Faso', 'Bélarus',
       'Bénin', 'Cabo Verde', 'Cambodge', 'Cameroun', 'Canada', 'Chili',
       'Chine - RAS de Hong-Kong', 'Chine - RAS de Macao',
       'Chine, Taiwan Province de', 'Chine, continentale', 'Chypre',
       'Colombie', 'Congo', 'Costa Rica', 'Croatie', 'Cuba',
       "Côte d'Ivoire", 'Danemark', 'Djibouti', 'Dominique',
       'El Salvador', 'Espagne', 'Estonie', 'Eswatini', 'Fidji',
       'Finlande', 'France', 'Fédération de Russie', 'Gabon', 'Gambie',
       'Ghana', 'Grenade', 'Grèce', 'Guatemala', 'Guinée',
       'Guinée-Bissau', 'Guyana', 'Géorgie', 'Haïti', 'Honduras',
       'Hongrie', 'Inde', 'Indonésie', "Iran (République islamique d')",
       'Iraq', 'Irlande', 'Islande', 'Israël', 'Italie', 'Jamaïque',
       'Japon', 'Jordanie', 'Kazakhstan', 'Kenya', 'Kirghizistan',
       'Kiribati', 'Koweït', 'Lesotho', 'Lettonie', 'Liban', 'Libéria',
       'Lituanie', 'Luxembourg', 'Macédoine du Nord', 'Madagascar',
       'Malaisie', 'Malawi', 'Maldives', 'Mali', 'Malte', 'Maroc',
       'Maurice', 'Mauritanie', 'Mexique', 'Mongolie', 'Monténégro',
       'Mozambique', 'Myanmar', 'Namibie', 'Nicaragua', 'Niger',
       'Nigéria', 'Norvège', 'Nouvelle-Calédonie', 'Nouvelle-Zélande',
       'Népal', 'Oman', 'Ouganda', 'Ouzbékistan', 'Pakistan', 'Panama',
       'Paraguay', 'Pays-Bas', 'Philippines', 'Pologne',
       'Polynésie française', 'Portugal', 'Pérou', 'Roumanie',
       "Royaume-Uni de Grande-Bretagne et d'Irlande du Nord", 'Rwanda',
       'République centrafricaine', 'République de Corée',
       'République de Moldova', 'République dominicaine',
       'République démocratique populaire lao',
       'République populaire démocratique de Corée',
       'République-Unie de Tanzanie', 'Saint-Kitts-et-Nevis',
       'Saint-Vincent-et-les Grenadines', 'Sainte-Lucie', 'Samoa',
       'Sao Tomé-et-Principe', 'Serbie', 'Sierra Leone', 'Slovaquie',
       'Slovénie', 'Soudan', 'Sri Lanka', 'Suisse', 'Suriname', 'Suède',
       'Sénégal', 'Tadjikistan', 'Tchad', 'Tchéquie', 'Thaïlande',
       'Timor-Leste', 'Togo', 'Trinité-et-Tobago', 'Tunisie',
       'Turkménistan', 'Turquie', 'Ukraine', 'Uruguay', 'Vanuatu',
       'Venezuela (République bolivarienne du)', 'Viet Nam', 'Yémen',
       'Zambie', 'Zimbabwe', 'Égypte', 'Émirats arabes unis', 'Équateur',
       "États-Unis d'Amérique", 'Éthiopie', 'Îles Salomon'], dtype=object)
```

In [37]:
```python
#Renommer les pays
# Dictionnaire de renommage
renommage_pays = {
    "République populaire démocratique de Corée": "Corée du Nord",
    "Chine - RAS de Hong-Kong": "Taiwan province de Chine",
    "Iran (République islamique d')": "Iran",
    "Venezuela (République bolivarienne du)": "Venezuela"
}

# Application du renommage
dispo_alimentaire["pays"] = dispo_alimentaire["pays"].replace(renommage_pays)
```

In [38]:
```python
#Vérification de valeurs manquantes
dispo_alimentaire.isna().any()
```

Out[38]:
```
element
pays                              False
produit                           False
annee                             False
production                         True
importation_qte                    True
disponibilite_interieure_qte       True
dtype: bool
```

Il y a des valeurs manquantes dans les colonnes production, importation_qte, disponibilite_interieure_qte nous allons donc les supprimer

In [40]:
```
#Suppression des lignes avec des valeurs manquantes
dispo_alimentaire = dispo_alimentaire.dropna()
dispo_alimentaire
```

Out[40]:

	element	pays	produit	annee	production	importation_qte	disponibilite_interie
0		Afghanistan	Viande de Volailles	2017	28.0	29.0	
1		Afrique du Sud	Viande de Volailles	2017	1667.0	514.0	
2		Albanie	Viande de Volailles	2017	13.0	38.0	
3		Algérie	Viande de Volailles	2017	275.0	2.0	
4		Allemagne	Viande de Volailles	2017	1514.0	842.0	
...		...	...	...	...	...	
167		Émirats arabes unis	Viande de Volailles	2017	48.0	433.0	
168		Équateur	Viande de Volailles	2017	340.0	0.0	
169		États-Unis d'Amérique	Viande de Volailles	2017	21914.0	123.0	
170		Éthiopie	Viande de Volailles	2017	14.0	1.0	
171		Îles Salomon	Viande de Volailles	2017	0.0	6.0	

168 rows × 6 columns

Les outliers

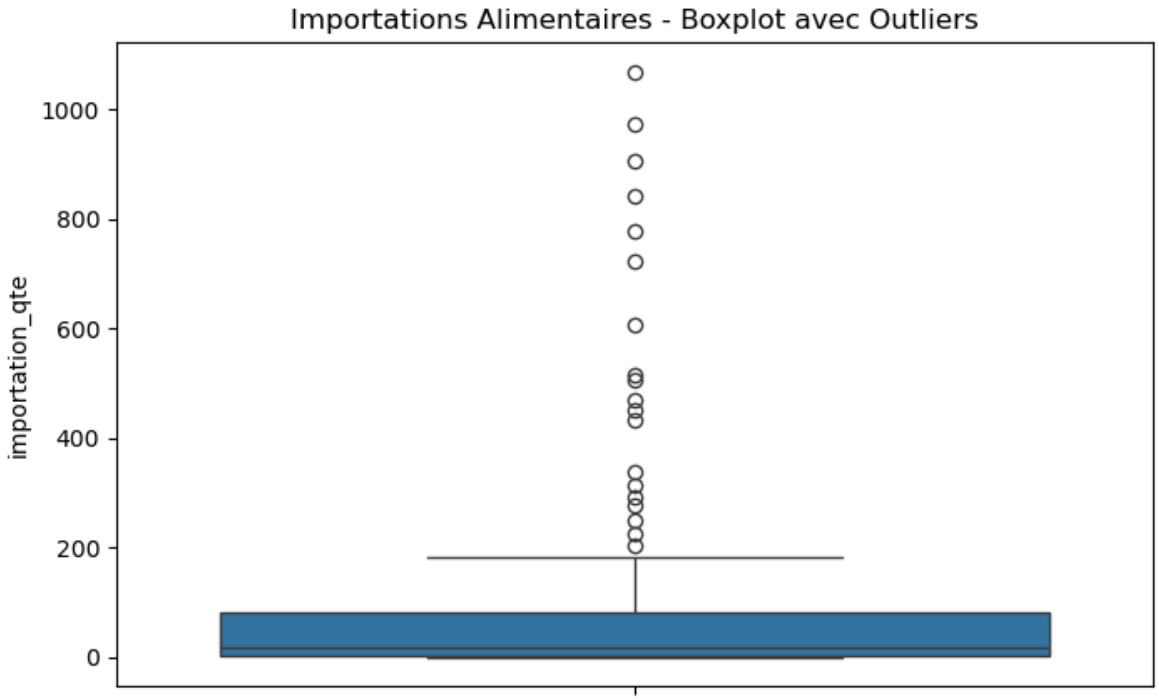
In [42]:

```
# Boxplot simple avec outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=dispo_alimentaire['importation_qte'])
plt.title('Importations Alimentaires - Boxplot avec Outliers')
plt.show()

# Compter les outliers
Q1 = dispo_alimentaire['importation_qte'].quantile(0.25)
Q3 = dispo_alimentaire['importation_qte'].quantile(0.75)
IQR = Q3 - Q1
```

```
outliers = dispo_alimentaire[(dispo_alimentaire['importation_qte'] < Q1 - 1.5*IQ
                              (dispo_alimentaire['importation_qte'] > Q3 + 1.5*IQ

print(f"Outliers détectés : {len(outliers)}")
print(outliers[['pays', 'importation_qte']])
```



Outliers détectés : 19

element	pays	importation_qte
1	Afrique du Sud	514.0
4	Allemagne	842.0
5	Angola	277.0
7	Arabie saoudite	722.0
16	Belgique	338.0
31	Taiwan province de Chine	907.0
34	Chine, continentale	452.0
40	Cuba	312.0
46	Espagne	205.0
51	France	506.0
52	Fédération de Russie	226.0
69	Iraq	470.0
75	Japon	1069.0
98	Mexique	972.0
117	Pays-Bas	608.0
118	Philippines	249.0
124	Royaume-Uni de Grande-Bretagne et d'Irlande du...	779.0
162	Viet Nam	291.0
167	Émirats arabes unis	433.0

Les valeurs aberrantes observées s'expliquent par des volumes d'importation de viande de volaille nettement plus élevés que ceux des autres pays, ce qui reflète une forte demande pour ce produit.

2.3 - Analyse exploratoire du fichier PIB_habitant

In [45]:
```
# Mise à jour du format (chaque ligne = un pays + une année)
PIB_habitant = PIB_habitant.melt(
    id_vars=['Country Name', 'Country Code'],
    var_name='annee',
    value_name='pib_hab'
)

# Renommage et nettoyage
PIB_habitant.rename(columns={'Country Name': 'pays'}, inplace=True)
PIB_habitant['annee'] = pd.to_numeric(PIB_habitant['annee'], errors='coerce')
PIB_habitant = PIB_habitant.dropna(subset=['pib_hab'])
```

In [46]:
```
# Filtrer sur 2022
PIB_habitant = PIB_habitant[(PIB_habitant['annee'] == 2022)]
PIB_habitant.head()
```

Out[46]:

	pays	Country Code	annee	pib_hab
17024	Aruba	ABW	2022.0	41649.450792
17025	NaN	AFE	2022.0	4229.086486
17026	Afghanistan	AFG	2022.0	2122.995815
17027	NaN	AFW	2022.0	5169.145547
17028	Angola	AGO	2022.0	7924.888806

In [47]:
```
# Suppression de la colonne Country Code
PIB_habitant= PIB_habitant.drop(columns=['Country Code'])

# Changement de la colonne 'annee' en entier
PIB_habitant['annee'] = PIB_habitant['annee'].astype(int)
PIB_habitant['pib_hab'] = PIB_habitant['pib_hab'].astype(int)

# Affichage du résultat
PIB_habitant.head()
```

Out[47]:

	pays	annee	pib_hab
17024	Aruba	2022	41649
17025	NaN	2022	4229
17026	Afghanistan	2022	2122
17027	NaN	2022	5169
17028	Angola	2022	7924

In [48]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(PIB_habitant.
print("Le tableau comporte {} colonne(s)".format(PIB_habitant.shape[1]))
```

```
Le tableau comporte 247 observation(s) ou article(s)
Le tableau comporte 3 colonne(s)
```

In [49]:
```
#Vérification des doublons sur toutes les colonnes
PIB_habitant_doublons = PIB_habitant[PIB_habitant.duplicated()]
```

```
# Afficher les doublons (si existants)
print("Nombre de doublons :", len(PIB_habitant_doublons))
```

```
Nombre de doublons : 0
```

Il n'y a pas de doublons dans PIB_habitant

In [51]:
```
#Vérifiaction des valeurs manquantes
PIB_habitant.isna().any()
```

Out[51]:
```
pays        True
annee      False
pib_hab    False
dtype: bool
```

Il y a des valeurs manquantes dans PIB_habitant dans la colonne pays donc on va les supprimer

In [53]:
```
#Suppression des valeurs manquantes de la colonne pays
PIB_habitant = PIB_habitant.dropna(subset=["pays"])
```

In [54]:
```
#Suppression de la colonne annee
PIB_habitant.drop(columns=["annee"], inplace=True)
```

In [55]:
```
#Affichage de la table
PIB_habitant.head()
```

Out[55]:

	pays	pib_hab
17024	Aruba	41649
17026	Afghanistan	2122
17028	Angola	7924
17029	Albanie	19446
17030	Andorre	68470

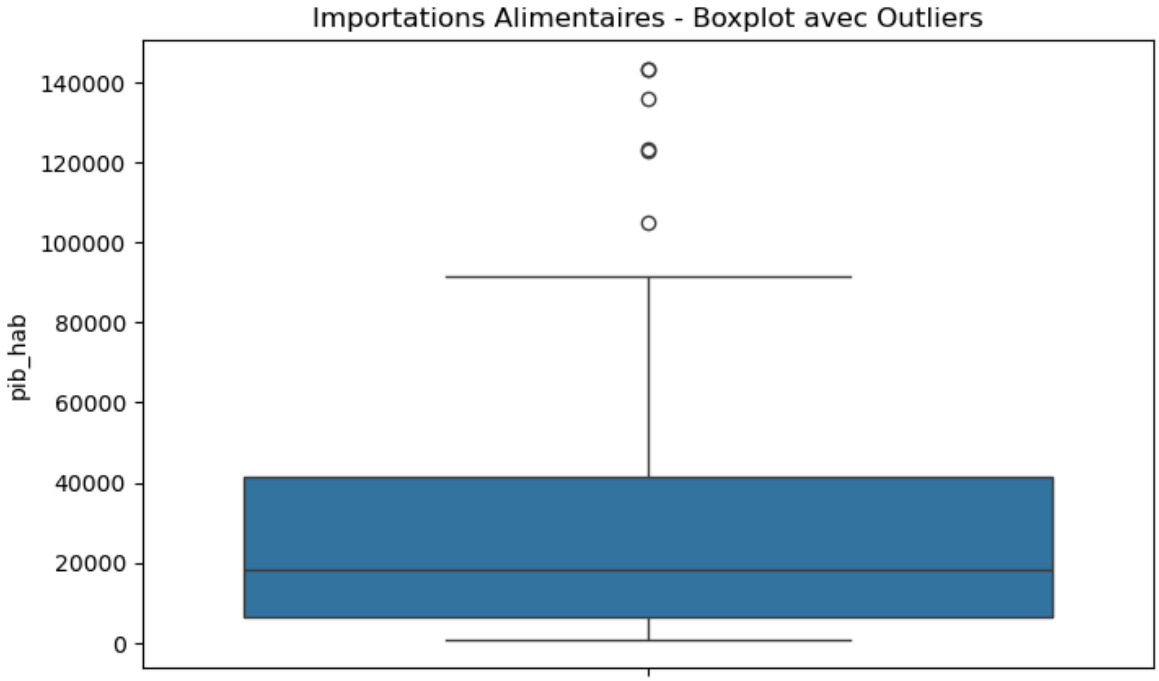
Les outliers

In [57]:
```
# Boxplot avec outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=PIB_habitant['pib_hab'])
plt.title('Importations Alimentaires - Boxplot avec Outliers')
plt.show()

# Compter les outliers
dispo_reset = PIB_habitant.reset_index(drop=True)

Q1 = dispo_reset['pib_hab'].quantile(0.25)
Q3 = dispo_reset['pib_hab'].quantile(0.75)
IQR = Q3 - Q1
outliers = dispo_reset[(dispo_reset['pib_hab'] < Q1 - 1.5*IQR) |
                       (dispo_reset['pib_hab'] > Q3 + 1.5*IQR)]
```

```
print(f"Outliers détectés : {len(outliers)}")
print(outliers[['pays', 'pib_hab']])
```




```
Outliers détectés : 6
           pays  pib_hab
24      Bermudes   105142
101      Irlande   136104
133   Luxembourg   143381
162      Norvège   123150
183        Qatar   122920
191    Singapour   143094
```

Ces outliers correspondent à des pays à très haut revenu par habitant. Leur PIB/habitant très élevé reflète des économies développées ou à forte spécialisation financière et énergétique.

2.4 - Analyse exploratoire du fichier population

In [60]:
```
#Affichage de la table population
population_2000_2018.head()
```

Out[60]:

	Code Domaine	Domaine	Code zone	Zone	Code Élément	Élément	Code Produit	Produit
0	OA	Séries temporelles annuelles	2	Afghanistan	511	Population totale	3010	Population-Estimations
1	OA	Séries temporelles annuelles	2	Afghanistan	511	Population totale	3010	Population-Estimations
2	OA	Séries temporelles annuelles	2	Afghanistan	511	Population totale	3010	Population-Estimations
3	OA	Séries temporelles annuelles	2	Afghanistan	511	Population totale	3010	Population-Estimations
4	OA	Séries temporelles annuelles	2	Afghanistan	511	Population totale	3010	Population-Estimations

In [61]:
```
#Supprimer les colonnes inutiles
colonnes_a_supprimer = [
    'Code Domaine', 'Domaine', 'Code zone', 'Code Élément',
    'Code Produit', 'Code année', 'Symbole', 'Description du Symbole', 'Note']
population_2000_2018 = population_2000_2018.drop(columns=colonnes_a_supprimer, e
```

In [62]:
```
#Renommer les colonnes pour plus de clarté
population_2000_2018 = population_2000_2018.rename(columns={
    'Zone': 'pays',
    'Élément': 'Type_donnée',
    'Année': 'annee',
    'Unité': 'Unite',
    'Valeur': 'Population'})
# Réinitialiser l'index
population_2000_2018 = population_2000_2018.reset_index(drop=True)
```

In [63]:
```
# Valeurs manquantes
population_2000_2018.isna().any()
```

Out[63]:
```
pays           False
Type_donnée    False
Produit        False
annee          False
Unite          False
Population     False
dtype: bool
```

Il n'y a pas de valeurs manquantes dans population_2000_2018

In [65]:
```
# Verification des doublons
population_2000_2018_doublons = population_2000_2018[population_2000_2018.duplic
# Afficher les doublons (si existants)
print("Nombre de doublons :", len(population_2000_2018_doublons))
```

```
Nombre de doublons : 0
```

In [66]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(population_20
print("Le tableau comporte {} colonne(s)".format(population_2000_2018.shape[1]))
```

```
Le tableau comporte 4411 observation(s) ou article(s)
Le tableau comporte 6 colonne(s)
```

Il n'y a pas de doublons dans population_2000_2018

In [68]:
```
# Changement du type des données
population_2000_2018['annee'] = population_2000_2018['annee'].astype(int)
population_2000_2018['Population'] = population_2000_2018['Population'].astype(i
population_2000_2018.head()
```

Out[68]:

	pays	Type_donnée	Produit	annee	Unite	Population
0	Afghanistan	Population totale	Population-Estimations	2000	1000 personnes	20779
1	Afghanistan	Population totale	Population-Estimations	2001	1000 personnes	21606
2	Afghanistan	Population totale	Population-Estimations	2002	1000 personnes	22600
3	Afghanistan	Population totale	Population-Estimations	2003	1000 personnes	23680
4	Afghanistan	Population totale	Population-Estimations	2004	1000 personnes	24726

In [69]:
```
population_2000_2018.loc[population_2000_2018['Unite'] == '1000 personnes', 'Pop
population_2000_2018['Unite'] = population_2000_2018['Unite'].replace('1000 pers
population_2000_2018.head()
```

Out[69]:

	pays	Type_donnée	Produit	annee	Unite	Population
0	Afghanistan	Population totale	Population-Estimations	2000	personnes	20779000
1	Afghanistan	Population totale	Population-Estimations	2001	personnes	21606000
2	Afghanistan	Population totale	Population-Estimations	2002	personnes	22600000
3	Afghanistan	Population totale	Population-Estimations	2003	personnes	23680000
4	Afghanistan	Population totale	Population-Estimations	2004	personnes	24726000

In [70]:
```
#Suppression des colonnes inutiles
population_2000_2018 = population_2000_2018.drop(columns=['Unite', 'Type_donnée'
```

In [71]:
```
# Filtrer sur 2018
population = population_2000_2018[(population_2000_2018['annee'] == 2018)]
population.head()
```

Out[71]:

	pays	annee	Population
18	Afghanistan	2018	37171000
37	Afrique du Sud	2018	57792000
56	Albanie	2018	2882000
75	Algérie	2018	42228000
94	Allemagne	2018	83124000

Les outliers

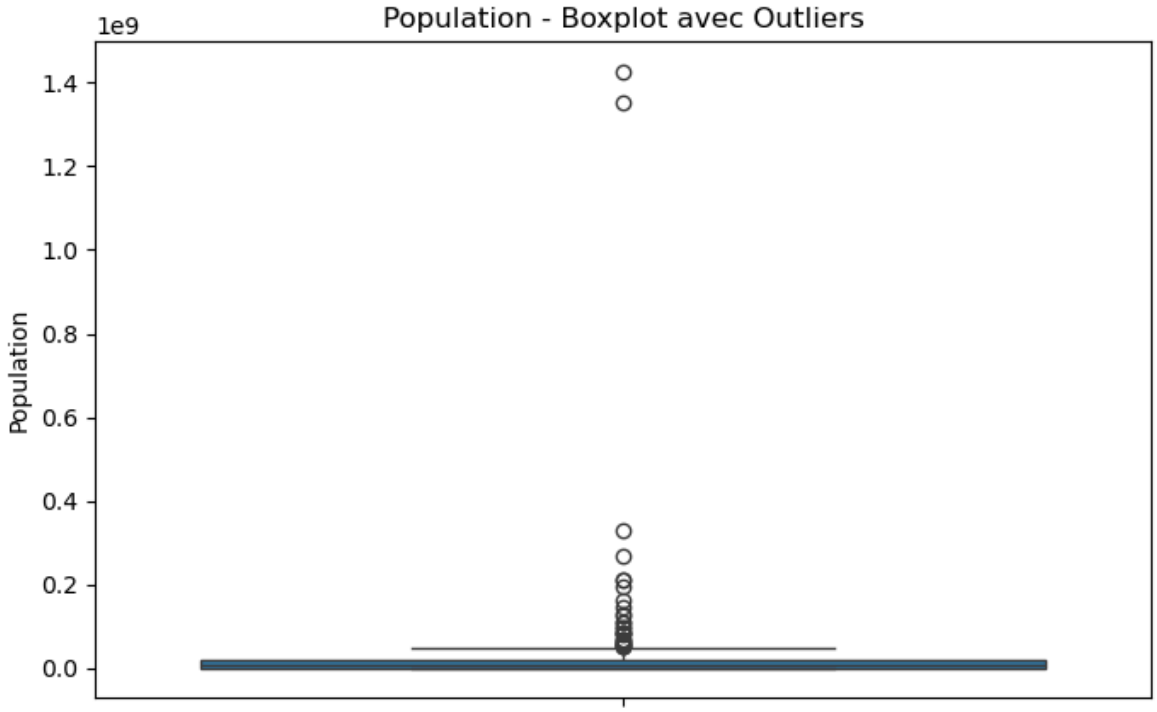
In [73]:

```
# Boxplot avec outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=population['Population'])
plt.title('Population - Boxplot avec Outliers')
plt.show()

# Compter les outliers
data_reset = population.reset_index(drop=True)

Q1 = data_reset['Population'].quantile(0.25)
Q3 = data_reset['Population'].quantile(0.75)
IQR = Q3 - Q1
outliers = data_reset[(data_reset['Population'] < Q1 - 1.5*IQR) |
                      (data_reset['Population'] > Q3 + 1.5*IQR)]

print(f"Outliers détectés : {len(outliers)}")
print(outliers[['pays', 'Population']])
```



```
Outliers détectés : 29
```

	pays	Population
1	Afrique du Sud	57792000
4	Allemagne	83124000
19	Bangladesh	161376000
31	Brésil	209469000
43	Chine, continentale	1427647000
46	Colombie	49661000
57	Égypte	98423000
65	États-Unis d'Amérique	327096000
66	Éthiopie	109224000
67	Fédération de Russie	145734000
70	France	64990000
103	Inde	1352642000
104	Indonésie	267670000
105	Iran (République islamique d')	81800000
110	Italie	60627000
112	Japon	127202000
115	Kenya	51392000
139	Mexique	126190000
146	Myanmar	53708000
152	Nigéria	195874000
160	Pakistan	212228000
168	Philippines	106651000
176	République de Corée	51171000
178	République démocratique du Congo	84068000
182	République-Unie de Tanzanie	56313000
185	Royaume-Uni de Grande-Bretagne et d'Irlande du...	67141000
218	Thaïlande	69428000
226	Turquie	82340000
232	Viet Nam	95545000

Ces outliers sont les pays les plus peuplés du monde. Leur population est tellement grande qu'ils se démarquent clairement des autres.

2.5 - performance_logistique

In [76]:
```
performance_logistique.head()
```

Out[76]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964
0	Aruba	ABW	Indice de performance logistique : performance...	LP.LPI.OVRL.XQ	NaN	NaN	NaN	NaN	NaN
1	NaN	AFE	Indice de performance logistique : performance...	LP.LPI.OVRL.XQ	NaN	NaN	NaN	NaN	NaN
2	Afghanistan	AFG	Indice de performance logistique : performance...	LP.LPI.OVRL.XQ	NaN	NaN	NaN	NaN	NaN
3	NaN	AFW	Indice de performance logistique : performance...	LP.LPI.OVRL.XQ	NaN	NaN	NaN	NaN	NaN
4	Angola	AGO	Indice de performance logistique : performance...	LP.LPI.OVRL.XQ	NaN	NaN	NaN	NaN	NaN

5 rows × 70 columns

In [77]:
```
# Sélectionner les colonnes utiles
performance_logistique = performance_logistique.loc[:, ["Country Name", "Country
+ performance_logistique.columns[performance_logistique.columns.str.isdigit()].t

# Transformer le tableau large → long (avec une colonne "annee")
performance_logistique = performance_logistique.melt(
    id_vars=["Country Name", "Country Code"],
    var_name="annee",
    value_name="indice_performance_logistique"
)

# Changement du nom des colonnes
performance_logistique = performance_logistique.rename(columns={
    "Country Name": "pays",
    "Country Code": "code_pays"
})
```

In [78]:
```
# Supprimer les lignes sans valeur (NaN)
performance_logistique = performance_logistique.dropna(subset=["indice_performan

# Convertir l'année en entier
performance_logistique["annee"] = performance_logistique["annee"].astype(int)

# Affichage du résultat
performance_logistique.head()
```

Out[78]:

	pays	code_pays	annee	indice_performance_logistique
12503	NaN	AFE	2007	2.385238
12504	Afghanistan	AFG	2007	1.210000
12505	NaN	AFW	2007	2.303333
12506	Angola	AGO	2007	2.480000
12507	Albanie	ALB	2007	2.080000

In [79]:
```
# Séparer les données de 2022 et des autres années
perf_2022 = performance_logistique[performance_logistique['annee'] == 2022].copy
perf_autres = performance_logistique[performance_logistique['annee'] != 2023].co

# Trier les autres années pour chaque pays (priorité à l'année la plus récente)
perf_autres = perf_autres.sort_values(by=['pays', 'annee'], ascending=[True, Fal

# Garder la donnée la plus proche par pays
perf_proche = perf_autres.drop_duplicates(subset='pays', keep='first')

# Remplir les NaN de 2022 avec les valeurs de l'année proche
performance_logistique = perf_2022.combine_first(perf_proche)

# Affichage
print("Données 2022 complétées dans performance_logistique :")
performance_logistique
```

```
Données 2022 complétées dans performance_logistique :
```

Out[79]:

	pays	code_pays	annee	indice_performance_logistique
12739	Timor-Leste	TLS	2007	1.710000
14379	Azerbaïdjan	AZE	2014	2.448376
14968	Éthiopie	ETH	2016	2.376767
15061	Mozambique	MOZ	2016	2.684105
15142	Tanzanie	TZA	2016	2.990185
...	...	...	...	...
16749	Viet Nam	VNM	2022	3.300000
16751	Monde	WLD	2022	3.000000
16754	Yémen, Rép. du	YEM	2022	2.200000
16755	Afrique du Sud	ZAF	2022	3.700000
16757	Zimbabwe	ZWE	2022	2.500000

217 rows × 4 columns

In [80]:
```
performance_logistique.isna().any()
```

Out[80]:
```
pays                                True
code_pays                          False
annee                              False
indice_performance_logistique      False
dtype: bool
```

Il y a des valeurs manquantes dans la colonne pays donc nous allons les supprimer

In [82]:
```
#Suppression des valeurs manquantes dans la colonne pays
performance_logistique = performance_logistique.dropna(subset=["pays"])
performance_logistique
```

Out[82]:

	pays	code_pays	annee	indice_performance_logistique
12739	Timor-Leste	TLS	2007	1.710000
14379	Azerbaïdjan	AZE	2014	2.448376
14968	Éthiopie	ETH	2016	2.376767
15061	Mozambique	MOZ	2016	2.684105
15142	Tanzanie	TZA	2016	2.990185
...	...	...	...	...
16749	Viet Nam	VNM	2022	3.300000
16751	Monde	WLD	2022	3.000000
16754	Yémen, Rép. du	YEM	2022	2.200000
16755	Afrique du Sud	ZAF	2022	3.700000
16757	Zimbabwe	ZWE	2022	2.500000

215 rows × 4 columns

In [83]:
```
#Affichage du nombre de doublons
performance_logistique.duplicated().sum()
```

Out[83]:
```
0
```

Il n'y a pas de doublons dans performance_logistique

In [85]:
```
#Suppression de la colonne code_pays
performance_logistique = performance_logistique.drop(columns=["code_pays"])
```

In [86]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(performance_l
print("Le tableau comporte {} colonne(s)".format(performance_logistique.shape[1]
```

```
Le tableau comporte 215 observation(s) ou article(s)
Le tableau comporte 3 colonne(s)
```

In [87]:
```
#Suppression de la colonne annee
performance_logistique.drop(columns=["annee"], inplace=True)
```

In [88]:
```
#Affichage du dataset
```

```
performance_logistique
```

Out[88]:

	pays	indice_performance_logistique
12739	Timor-Leste	1.710000
14379	Azerbaïdjan	2.448376
14968	Éthiopie	2.376767
15061	Mozambique	2.684105
15142	Tanzanie	2.990185
...	...	...
16749	Viet Nam	3.300000
16751	Monde	3.000000
16754	Yémen, Rép. du	2.200000
16755	Afrique du Sud	3.700000
16757	Zimbabwe	2.500000

215 rows × 2 columns

On remarque qu'il y a des régions en plus des pays, pour continuer l'analyse on va choisir de conserver uniquement les pays qui correspondent avec le fichier dispo_alimentaire

In [90]:

```
# Extraire les pays uniques en supprimant les valeurs manquantes
pays_perf = set(performance_logistique["pays"].dropna())
pays_dispo = set(dispo_alimentaire["pays"].dropna())

# Identifier les pays présents dans performance_logistique mais absents de dispo
pays_sans_correspondance = pays_perf - pays_dispo

# Afficher la liste
print("Pays sans correspondance dans dispo_alimentaire :")
for pays in sorted(pays_sans_correspondance):
    print("-", pays)
```

Pays sans correspondance dans dispo_alimentaire :
- Afrique du Nord et Moyen-Orient (BIRD et IDA)
- Afrique du Nord et Moyen-Orient (hors revenu élevé)
- Afrique subsaharienne
- Afrique subsaharienne (BIRD et IDA)
- Afrique subsaharienne (hors revenu élevé)
- Amérique du Nord
- Amérique latine et Caraïbes
- Amérique latine et Caraïbes (BIRD et IDA)
- Amérique latine et Caraïbes (hors revenu élevé)
- Asie de l’Est et Pacifique
- Asie de l’Est et Pacifique (BIRD et IDA)
- Asie de l’Est et Pacifique (hors revenu élevé)
- Asie du Sud
- Asie du Sud (BIRD et IDA)
- Autres petits états
- BIRD et IDA
- BIRD seulement
- Bahreïn
- Bhoutan
- Bolivie
- Brunéi Darussalam
- Burundi
- Chine
- Chine, RAS de Hong Kong
- Comores
- Congo, République du
- Congo, République démocratique du
- Corée, République de
- Djibouti
- Europe centrale et les pays baltes
- Europe et Asie centrale
- Europe et Asie centrale (BIRD et IDA)
- Europe et Asie centrale (hors revenu élevé)
- Faible revenu
- Fragile et les situations de conflit touchées
- Guinée équatoriale
- IDA mélange
- IDA seulement
- IDA totale
- Iran, République islamique d’
- Le monde arabe
- Libye
- Maldives
- Moldova
- Monde
- Moyen-Orient, Afrique du Nord, Afghanistan et Pakistan
- Ouzbékistan
- Papouasie-Nouvelle-Guinée
- Pays les moins avancés: classement de l’ONU
- Pays membres de l'OCDE
- Pays pauvres très endettés (PPTE)
- Petits états
- Petits états des Caraïbes
- Petits états insulaires du Pacifique
- Qatar
- Revenu faible et intermédiaire
- Revenu intermédiaire
- Revenu intermédiaire, tranche inférieure
- Revenu intermédiaire, tranche supérieure

- Revenu élevé
- Royaume-Uni
- République arabe syrienne
- République démocratique populaire lao
- République fédérale de Somalie
- République kirghize
- République slovaque
- République tchèque
- Singapour
- Tanzanie
- Union européenne
- Yémen, Rép. du
- Zone euro
- de Post-dividende démographique
- de Pré-dividende démographique
- de dividende précoce démographique
- de dividende tardif démographique
- Égypte, République arabe d'
- Érythrée
- États-Unis

In [91]:
```
#Conservation des pays avec des correspondances avec la table dispo_alimentaire
perf = performance_logistique.copy()
dispo = dispo_alimentaire.copy()

# Identifier les pays communs aux deux fichiers
pays_communs = set(dispo["pays"].dropna()).intersection(set(perf["pays"].dropna(

# Filtrer le fichier performance_logistique sur les pays communs
performance_logistique = perf[perf["pays"].isin(pays_communs)].reset_index(drop=

# Afficher le nombre de lignes conservées
print(f"Fichier performance_logistique filtré : {len(performance_logistique )} l
```

```
Fichier performance_logistique filtré : 136 lignes conservées pour 136 pays.
```

In [92]:
```
#Affichage de la table
performance_logistique
```

Out[92]:

	pays	indice_performance_logistique
0	Timor-Leste	1.710000
1	Azerbaïdjan	2.448376
2	Éthiopie	2.376767
3	Mozambique	2.684105
4	Côte d'Ivoire	3.080000
...	...	...
131	Uruguay	3.000000
132	Venezuela	2.300000
133	Viet Nam	3.300000
134	Afrique du Sud	3.700000
135	Zimbabwe	2.500000

136 rows × 2 columns

Les outliers

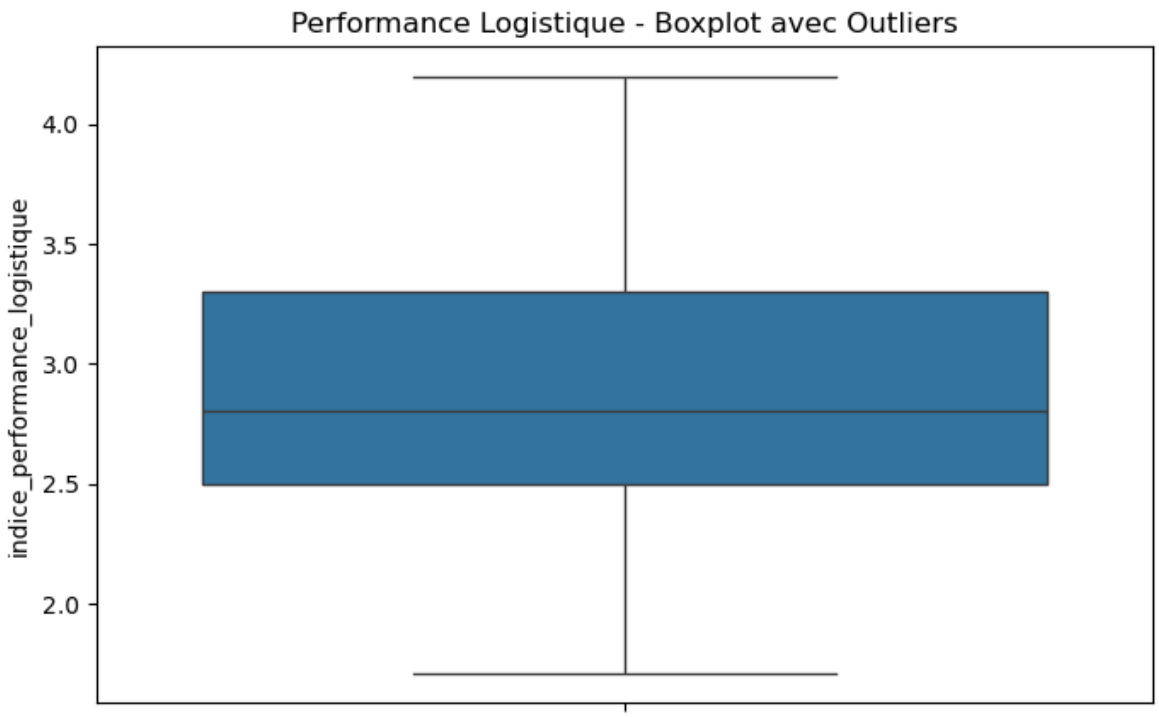
In [94]:

```
# Boxplot avec outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=performance_logistique['indice_performance_logistique'])
plt.title('Performance Logistique - Boxplot avec Outliers')
plt.show()

# Compter les outliers
data_reset = performance_logistique.reset_index(drop=True)

Q1 = data_reset['indice_performance_logistique'].quantile(0.25)
Q3 = data_reset['indice_performance_logistique'].quantile(0.75)
IQR = Q3 - Q1
outliers = data_reset[(data_reset['indice_performance_logistique'] < Q1 - 1.5*IQ
                       (data_reset['indice_performance_logistique'] > Q3 + 1.5*I

print(f"Outliers détectés : {len(outliers)}")
print(outliers[['pays', 'indice_performance_logistique']])
```




```
Outliers détectés : 0
Empty DataFrame
Columns: [pays, indice_performance_logistique]
Index: []
```

2.6 - Taux_droit_douane

In [96]:
```
#Affichage de la table Taux_droit_douane
Taux_droit_douane.head()
```

Out[96]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1
0	Aruba	ABW	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	NaN	NaN	NaN	NaN	
1	NaN	AFE	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	NaN	NaN	NaN	NaN	
2	Afghanistan	AFG	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	NaN	NaN	NaN	NaN	
3	NaN	AFW	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	NaN	NaN	NaN	NaN	
4	Angola	AGO	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	NaN	NaN	NaN	NaN	

5 rows × 69 columns

In [97]:
```
# Suppression des colonnes "Unnamed" inutiles s'il y en a
Taux_droit_douane = Taux_droit_douane.loc[:, ~Taux_droit_douane.columns.str.cont
```

In [98]:
```
# Transformation du format
Taux_droit_douane = Taux_droit_douane.melt(
    id_vars=['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code']
    var_name='annee',
    value_name='%droit_douane'
)

# Renommage des colonnes
Taux_droit_douane.rename(columns={
    'Country Name': 'pays',
    'Indicator Name': 'indicateur'
}, inplace=True)

# Conversion des années en numérique et suppression des NaN
```

```
Taux_droit_douane['annee'] = pd.to_numeric(Taux_droit_douane['annee'], errors='c
Taux_droit_douane = Taux_droit_douane.dropna(subset=['%droit_douane'])
```

In [99]:
```
# Séparer les données de 2022 et des autres années
Taux_droit_douane = Taux_droit_douane[Taux_droit_douane['annee'] == 2022].copy()
douane_autres = Taux_droit_douane[Taux_droit_douane['annee'] != 2022].copy()

# Garder la donnée la plus proche par pays
douane_autres = douane_autres.sort_values(by=['pays', 'annee'], ascending=[True,
douane_proche = douane_autres.drop_duplicates(subset='pays', keep='first')

# Remplir les NaN de 2022 avec les valeurs de l'année proche
Taux_droit_douane = Taux_droit_douane.combine_first(douane_proche)

# Affichage
print(" Données 2022 complétées :")
Taux_droit_douane
```

```
Données 2022 complétées :
```

Out[99]:

	pays	Country Code	indicateur	Indicator Code	annee	%droit_douane
16496	Angola	AGO	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	2022	8.85
16497	Albanie	ALB	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	2022	0.26
16501	Argentine	ARG	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	2022	6.45
16505	Australie	AUS	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	2022	0.99
16506	Autriche	AUT	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	2022	1.33
...	...	...	...	...	...	...
16746	Venezuela	VEN	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	2022	12.84
16749	Viet Nam	VNM	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	2022	1.07
16750	Vanuatu	VUT	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	2022	11.48

	pays	Country Code	indicateur	Indicator Code	annee	%droit_douane
16755	Afrique du Sud	ZAF	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	2022	4.66
16756	Zambie	ZMB	Taux des droits de douane, appliqués, moyenne ...	TM.TAX.MRCH.WM.AR.ZS	2022	4.08

144 rows × 6 columns

In [100…
```
# Suppression des colonnes inutiles
Taux_droit_douane = Taux_droit_douane.drop(
    columns=['Country Code', 'Indicator Code', 'annee', 'indicateur']).reset_ind
```

In [101…
```
Taux_droit_douane.isna().any()
```

Out[101…
```
pays            False
%droit_douane   False
dtype: bool
```

In [102…
```
#Conservation des pays avec des correspondances avec la table dispo_alimentaire
douane_2 = Taux_droit_douane.copy()
dispo_2 = dispo_alimentaire.copy()

# Identifier les pays communs aux deux fichiers
pays_communs = set(dispo_2["pays"].dropna()).intersection(set(douane_2["pays"].d

# Filtrer le fichier Taux_droit_douane sur les pays communs
Taux_droit_douane = douane_2[douane_2["pays"].isin(pays_communs)].reset_index(dr

# Afficher le nombre de lignes conservées
print(f"Fichier Taux_droit_douane : {len(Taux_droit_douane)} lignes conservées p
```

```
Fichier Taux_droit_douane : 119 lignes conservées pour 119 pays.
```

In [103…
```
Taux_droit_douane
```

Out[103…

	pays	%droit_douane
0	Angola	8.85
1	Albanie	0.26
2	Argentine	6.45
3	Australie	0.99
4	Autriche	1.33
...	...	...
114	Venezuela	12.84
115	Viet Nam	1.07
116	Vanuatu	11.48
117	Afrique du Sud	4.66
118	Zambie	4.08

119 rows × 2 columns

Les outliers

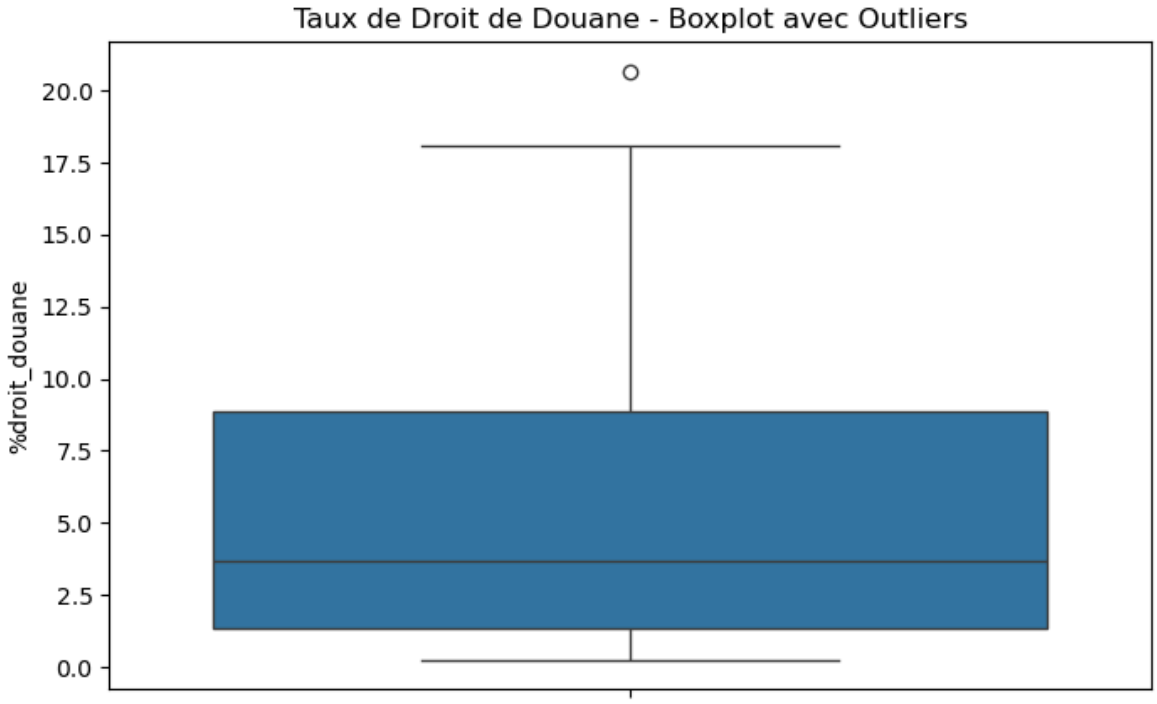
In [105…

```
# Boxplot avec outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=Taux_droit_douane['%droit_douane'])
plt.title('Taux de Droit de Douane - Boxplot avec Outliers')
plt.show()

# Compter les outliers
data_reset = Taux_droit_douane.reset_index(drop=True)

Q1 = data_reset['%droit_douane'].quantile(0.25)
Q3 = data_reset['%droit_douane'].quantile(0.75)
IQR = Q3 - Q1
outliers = data_reset[(data_reset['%droit_douane'] < Q1 - 1.5*IQR) |
                      (data_reset['%droit_douane'] > Q3 + 1.5*IQR)]

print(f"Outliers détectés : {len(outliers)}")
print(outliers[['pays', '%droit_douane']])
```




```
Outliers détectés : 1
            pays  %droit_douane
98  Îles Salomon          20.66
```

Cet outlier vient du fait que les Îles Salomon appliquent un taux de droit de douane bien plus élevé que la moyenne des autres pays.

Étape 3 - Jointure des fichiers

3.1 - Jointure

Fusion de la table dispo_alimentaire et population

In [110…
```
# Fusion des deux tables uniquement sur 'pays'
dispo_alimentaire_hab = dispo_alimentaire.merge(population, on="pays", how="inne
```

In [111…
```
# Conversion des unités : de milliers de tonnes en tonnes
dispo_alimentaire_hab["production_tonnes"] = dispo_alimentaire_hab["production"]
dispo_alimentaire_hab["importation_tonnes"] = dispo_alimentaire_hab["importation
dispo_alimentaire_hab["disponibilite_interieure_tonnes"] = dispo_alimentaire_hab

# Calcul par habitant (tonnes par personne) – en évitant la division par zéro
dispo_alimentaire_hab["production_par_habitant"] = dispo_alimentaire_hab["produc
dispo_alimentaire_hab["importation_par_habitant"] = dispo_alimentaire_hab["impor
dispo_alimentaire_hab["disponibilite_interieure_par_habitant"] = dispo_alimentai

# Conversion en kg par habitant
dispo_alimentaire_hab["production_kg_par_habitant"] = dispo_alimentaire_hab["pro
dispo_alimentaire_hab["importation_kg_par_habitant"] = dispo_alimentaire_hab["im
dispo_alimentaire_hab["disponibilite_interieure_kg_par_habitant"] = dispo_alimen
```

```
#Changement du nom de la colonne Population en population
dispo_alimentaire_hab.rename(columns={"Population": "population"}, inplace=True)
# Affichage du résultat
dispo_alimentaire_hab.head()
```

Out[111…

	pays	produit	annee_x	production	importation_qte	disponibilite_interieure_q
0	Afghanistan	Viande de Volailles	2017	28.0	29.0	57
1	Afrique du Sud	Viande de Volailles	2017	1667.0	514.0	2118
2	Albanie	Viande de Volailles	2017	13.0	38.0	47
3	Algérie	Viande de Volailles	2017	275.0	2.0	277
4	Allemagne	Viande de Volailles	2017	1514.0	842.0	1739

In [112…

```
#Nombre de pays dans le fichier dispo_alimentaire_hab
dispo_alimentaire['pays'].nunique()
```

Out[112…

```
168
```

In [113…

```
colonnes_a_supprimer = [
    'produit',
    'annee_x',
    'production',
    'importation_qte',
    'disponibilite_interieure_qte',
    'annee_y',
    'production_tonnes',
    'importation_tonnes',
    'disponibilite_interieure_tonnes',
    'production_par_habitant',
    'importation_par_habitant',
    'disponibilite_interieure_par_habitant'
]

dispo_alimentaire_hab.drop(columns=colonnes_a_supprimer, inplace=True)

# Affichage du DataFrame nettoyé
dispo_alimentaire_hab.head()
```

Out[113...

	pays	population	production_kg_par_habitant	importation_kg_par_habitant	dis
0	Afghanistan	37171000	0.753275	0.780178	
1	Afrique du Sud	57792000	28.844823	8.893965	
2	Albanie	2882000	4.510756	13.185288	
3	Algérie	42228000	6.512267	0.047362	
4	Allemagne	83124000	18.213753	10.129445	

Fusion des tables dispo_alimentaire_hab et pib_hab

In [115...

```
dispo_alimentaire_pib = dispo_alimentaire_hab.merge(PIB_habitant, on="pays", how
dispo_alimentaire_pib
```

Out[115...

	pays	population	production_kg_par_habitant	importation_kg_par_habitant
0	Afghanistan	37171000	0.753275	0.780178
1	Afrique du Sud	57792000	28.844823	8.893965
2	Albanie	2882000	4.510756	13.185288
3	Algérie	42228000	6.512267	0.047362
4	Allemagne	83124000	18.213753	10.129445
...	...	...	...	...
141	Zimbabwe	14438000	4.779055	0.415570
142	Émirats arabes unis	9630000	4.984424	44.963655
143	Équateur	17084000	19.901662	0.000000
144	Éthiopie	109224000	0.128177	0.009155
145	Îles Salomon	652000	0.000000	9.202454

146 rows × 6 columns

Fusion des tables dispo_alimentaire_pib et stabilite_politique

In [117...

```
donnees_1 = dispo_alimentaire_pib.merge(stabilite_politique, on="pays", how="inn
donnees_1
```

Out[117…

	pays	population	production_kg_par_habitant	importation_kg_par_habitant
0	Afghanistan	37171000	0.753275	0.780178
1	Afrique du Sud	57792000	28.844823	8.893965
2	Albanie	2882000	4.510756	13.185288
3	Algérie	42228000	6.512267	0.047362
4	Allemagne	83124000	18.213753	10.129445
...	...	...	...	...
141	Zimbabwe	14438000	4.779055	0.415570
142	Émirats arabes unis	9630000	4.984424	44.963655
143	Équateur	17084000	19.901662	0.000000
144	Éthiopie	109224000	0.128177	0.009155
145	Îles Salomon	652000	0.000000	9.202454

146 rows × 7 columns

Fusion des tables donnees et performance_logistique

In [119…

```
donnees_2 = donnees_1.merge(performance_logistique, on="pays", how="inner")
donnees_2
```

Out[119…

	pays	population	production_kg_par_habitant	importation_kg_par_habitant
0	Afghanistan	37171000	0.753275	0.780178
1	Afrique du Sud	57792000	28.844823	8.893965
2	Albanie	2882000	4.510756	13.185288
3	Algérie	42228000	6.512267	0.047362
4	Allemagne	83124000	18.213753	10.129445
...	...	...	...	...
129	Zimbabwe	14438000	4.779055	0.415570
130	Émirats arabes unis	9630000	4.984424	44.963655
131	Équateur	17084000	19.901662	0.000000
132	Éthiopie	109224000	0.128177	0.009155
133	Îles Salomon	652000	0.000000	9.202454

134 rows × 8 columns

Fusion des tables donnees et imposition

In [121…

```
donnees_finale = donnees_2.merge(Taux_droit_douane, on="pays", how="inner")
donnees_finale
```

Out[121…

	pays	population	production_kg_par_habitant	importation_kg_par_habitant	d
0	Afrique du Sud	57792000	28.844823	8.893965	
1	Albanie	2882000	4.510756	13.185288	
2	Algérie	42228000	6.512267	0.047362	
3	Allemagne	83124000	18.213753	10.129445	
4	Angola	30809000	1.363238	8.990879	
...	...	...	...	...	
105	Uruguay	3449000	9.567991	0.869817	
106	Viet Nam	95545000	9.608038	3.045685	
107	Zambie	17351000	2.824045	0.691603	
108	Équateur	17084000	19.901662	0.000000	
109	Îles Salomon	652000	0.000000	9.202454	

110 rows × 9 columns

In [122…

```
# Arrondir les colonnes
colonnes = [
    'production_kg_par_habitant',
    'importation_kg_par_habitant',
    'disponibilite_interieure_kg_par_habitant'
]

# Conversion en float et arrondi à 2 chiffres après la virgule
for col in colonnes:
    if col in donnees_finale.columns:
        donnees_finale[col] = donnees_finale[col].astype(float).round(2)
```

3.2 - Fichier final

On obtient un fichier finale avec 110 pays contenant 8 variables

In [125…

```
# Liste de renommage
renommer_colonnes = {
    'pays': 'pays',
    'population': 'population',
    'production_kg_par_habitant': 'production_hab',
    'importation_kg_par_habitant': 'importation_hab',
    'disponibilite_interieure_kg_par_habitant': 'disponibilite_hab',
    'pib_hab': 'pib_hab',
    'stabilite': 'stabilite',
    'indice_performance_logistique': 'indice_perf_logistique',
    '%droit_douane': 'douane'
```

```
}

# Appliquer le renommage
donnees_finale = donnees_finale.rename(columns=renommer_colonnes)
```

In [126…
```
donnees_finale
```

Out[126…

	pays	population	production_hab	importation_hab	disponibilite_hab	pib_ha
0	Afrique du Sud	57792000	28.84	8.89	36.65	1475
1	Albanie	2882000	4.51	13.19	16.31	1944
2	Algérie	42228000	6.51	0.05	6.56	1583
3	Allemagne	83124000	18.21	10.13	20.92	6758
4	Angola	30809000	1.36	8.99	10.35	792
…	…	…	…	…	…	
105	Uruguay	3449000	9.57	0.87	9.57	3300
106	Viet Nam	95545000	9.61	3.05	12.62	1390
107	Zambie	17351000	2.82	0.69	3.46	384
108	Équateur	17084000	19.90	0.00	19.96	1519
109	Îles Salomon	652000	0.00	9.20	4.60	269

110 rows × 9 columns

3.3 - Export du fichier final

In [128…
```
# Exporter donnees_finale en CSV
donnees_finale.to_csv('donnees_finale.csv', index=False, encoding='utf-8')
```