In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# importation des librairies
```

In [2]:

```python
#import des fichiers csv

tp = pd.read_csv('/Users/ismaelh/Documents/Data Analyst/Projet 4/population.csv') # total population

fs = pd.read_csv('/Users/ismaelh/Documents/Data Analyst/Projet 4/dispo_alimentaire.csv') # food supply

fa = pd.read_csv('/Users/ismaelh/Documents/Data Analyst/Projet 4/aide_alimentaire.csv') # food aid

un = pd.read_csv('/Users/ismaelh/Documents/Data Analyst/Projet 4/sous_nutrition.csv')# under nutrition
```

QUESTION 1: Proportion de personnes en état de sous-nutrition

In [83]:

```python
# voir l'etat des données dans les tables necessaires
# un,tp
```

In [4]:

```python
# conversion des virgules en points afin de pouvoir les convertirs en type float pour les calculs

un['Valeur'] = un['Valeur'].str.replace(',','.')     #data["column_name"]=data["column_name"].str.replace(',','.')
un['Valeur'] = un['Valeur'].str.replace('<0.1','0')
#print(un)
```

```
/var/folders/93/8vxwb4t10x7dvsxz5nyktctm0000gn/T/ipykernel_14932/718356731.py:4: FutureWarning: The default value of regex will change from True to False in a future version.
  un['Valeur'] = un['Valeur'].str.replace('<0.1','0')
```

In [5]:

```python
# Remplacement des valeurs NULL

un.fillna(0, inplace = True)
tp.fillna(0, inplace = True)
```

In [6]:

```python
# conversion en type float

un['Valeur'] = un['Valeur'].astype(float)
tp['Valeur'] = tp['Valeur'].astype(float)
tp['Année'] = tp['Année'].astype(int)
#un.dtypes, tp.dtypes
```

In [7]:

```python
# Regroupement des plages années en une année avec valeur moyenne

un['Année'] = un['Année'].replace(['2012-2014', '2013-2015', '2014-2016', '2015-2017', '2016-2018', '2017-2019']
                                 ,[2013, 2014, 2015, 2016, 2017, 2018])
#print(un)
```

In [8]:

```python
# Renommage des colonnes et mise à la même unité

tp['Valeur'] *= 1000
tp.rename(columns={"Valeur": "Population"}, inplace=True)
```

In [9]:

```python
un.rename(columns={"Valeur": "sous_nutrition"}, inplace=True)
un['sous_nutrition'] *= 1000000
```

In [10]:

```python
# Jointure entre les tables population et population en sous nutrition

tp_un = tp.merge(un)
#tp_un
```

In [11]:

```python
# Création d'un df avec les données pour l'année 2017

tp_2017 = tp_un.loc[tp_un['Année'] == 2017,['Zone','Population', 'sous_nutrition']]

#tp_2017
```

In [84]:

```python
# Remise à zéro de l'index

tp_2017.reset_index(drop=True, inplace=True)
print('population mondiale (milliards):',round(tp_2017['Population'].sum()/1000000000,2))
print('population en sous nutrition (milliards):',round(tp_2017['sous_nutrition'].sum()/1000000000,2))
```

```
population mondiale (milliards): 7.54
population en sous nutrition (milliards): 0.54
```

In [13]:

```python
# Résultat question 1

print('Population totale en sous nutrition ', round((tp_2017['sous_nutrition'].sum() * 100) / (tp_2017['Population']).sum(),2),'%')
```

```
Population totale en sous nutrition  7.1 %
```

QUESTION 2 : Nombre théorique de personnes qui pourraient être nourries

In [ ]:

```python
#calcul : dispo par pays et somme ->( dispo alim (kcal) * population du pays ) * 365
```

In [15]:

```python
#fs
```

In [17]:

```python
# Remplacement des valeurs NULL

fs.fillna(0, inplace = True)
```

In [18]:

```python
# Création d'un df avec les données de disponibilité alimenttaire pour l'année 2017

fs_pop = fs.merge(tp_2017)
```

In [19]:

```python
# Rajout d'une colonne avec les informations sur les calories par pays

fs_pop['Calories pays']= fs_pop['Disponibilité alimentaire (Kcal/personne/jour)'] * fs_pop['Population']* 365 / 1000
```

In [20]:

```python
# Détermination du total calorie

Total_calories = round(fs_pop['Calories pays'].sum())
Total_calories
```

Out[20]:

```
7635429388976
```

In [21]:

```python
# Résultat question 2

Nbre_pers_theorique = round((Total_calories / (365 * 2500))/1000000, 2)
print('Nombre de personnes pouvant être nourries en théorie (en milliards)',Nbre_pers_theorique)
print("Proportion d'humains pouvant être nourries en théorie:", round((Total_calories / (365 * 2500))/1000000 * 100 / ((tp_2017['Population']).sum()/1000000000),2), '%')
```

```
Nombre de personnes pouvant être nourries en théorie (en milliards) 8.37
Proportion d'humains pouvant être nourries en théorie: 110.92 %
```

Question 3 : Disponibilité alimentaire des produits végétaux

In [23]:

```python
# Création d'un df avec uniquement les produits végétaux

fs_veg = fs_pop.loc[fs_pop['Origine']== 'vegetale']

print('Disponibilité totale en des produit végétaux :' , fs_veg['Calories pays'].sum(), 'kcal')
```

```
Disponibilité totale en des produit végétaux : 6300178937197.865 kcal
```

In [24]:

```python
# Nombre de personne pouvant être nourries avec des produits végétaux uniquement

Nbre_pers_vege = round(fs_veg['Calories pays'].sum()/(365 * 2500) / 1000000, 2)
print('Nombre de personnes pouvant être nourries en théorie (en milliards)', Nbre_pers_vege)
print("Proportion d'humains pouvant être nourries en théorie:", round(Nbre_pers_vege  * 100 / (tp_2017['Population'].sum()/1000000000),2),'%')
```

```
Nombre de personnes pouvant être nourries en théorie (en milliards) 6.9
Proportion d'humains pouvant être nourries en théorie: 91.47 %
```

Question 4 : Utilisation de la disponibilité intérieure

In [25]:

```python
# Calcul de la disponibilité intérieure

fs_pop['dispo interieure net'] = (fs_pop['Disponibilité intérieure'] + fs_pop['Importations - Quantité']) - (fs_pop['Exportations - Quantité'] + fs_pop['Variation de stock'])
print( 'Disponibilité interieur net ( en milliards de tonnes)',(fs_pop['dispo interieure net'].sum()/1000000))
```

```
Disponibilité interieur net ( en milliards de tonnes) 9.751878
```

In [26]:

```python
# Détermination de la nourriture totale

dispo_nourriture = fs_pop['Nourriture'].sum()
dispo_nourriture
```

Out[26]:

```
4805525.0
```

Part de la disponibilité intérieure qui est attribuée à l'alimentation animale

In [27]:

```python
# Quantité totales d'aliments pour animaux

tot_anim = fs_pop['Aliments pour animaux'].sum()
tot_anim
```

Out[27]:

```
1288002.0
```

In [28]:

```python
# Résultat proportion allouée aux animaux

part_anim = round(tot_anim * 100 / fs_pop['dispo interieure net'].sum(), 2)
print( "Part attribué à l'alimentation animale ", part_anim, '%')
```

```
Part attribué à l'alimentation animale  13.21 %
```

Part de la disponibilité intérieure qui est perdue

In [29]:

```python
# Calcul des pertes

tot_perte = fs['Pertes'].sum()
tot_perte
```

Out[29]:

```
453698.0
```

In [30]:

```python
# Proportion des pertes

part_perdue = round((tot_perte * 100) / fs_pop['dispo interieure net'].sum(), 2)
print('Pourcentage de pertes', part_perdue,'%')
```

```
Pourcentage de pertes 4.65 %
```

Part de la disponibilité intérieure utilisée pour l'alimentation humaine

In [31]:

```python
# Proportion utilisé pour l'alimentation humaine

part_hum = round((dispo_nourriture * 100) / fs_pop['dispo interieure net'].sum(), 2)
print("pourcentage de la disponibilité interieure utilisée pour l'alimentation humaine", part_hum, '%')
```

```
pourcentage de la disponibilité interieure utilisée pour l'alimentation humaine 49.28 %
```

Question 5 : Répartition de l'utilisation des céréales entre les humains et animaux

In [32]:

```python
# Liste des céréales

liste_cereales = ["Blé et produits", "Riz et produits", "Orge et produits", "Maïs et produits", "Seigle et produits",
                  "Avoine", "Millet et produits", "Sorgho et produits", "Céréales, Autres"]
```

In [33]:

```python
# Total aliments pour animaux

fs['Aliments pour animaux'].sum()
```

Out[33]:

```
1304245.0
```

In [37]:

```python
# Total nourriture

fs['Nourriture'].sum()
```

Out[37]:

```
4876258.0
```

In [38]:

```python
# Création d'un df avec les produits de la liste céréales

fs_cer = fs.loc[fs['Produit'].isin(liste_cereales)]
#fs_cer
```

In [40]:

```python
# Répartition de l'utilisation des céréales entre humains et animaux

print("Proportion d'aliments pour animaux :",round(fs_cer["Aliments pour animaux"]. sum() * 100 / fs_cer["Disponibilité intérieure"].sum() ,2),'%')
print("Proportion de nourriture :", round(fs_cer["Nourriture"].sum()* 100 / fs_cer["Disponibilité intérieure"]. sum(), 2),'%')
```

```
Proportion d'aliments pour animaux : 69.34 %
Proportion de nourriture : 18.13 %
```

Question 6 : Utilisation du manioc en Thaïlande

1. la proportion de nombre de personne sous nutrition /population de Thaïlande.

2. le rapport entre la production et l'exportation du manioc

In [54]:

```python
# Création d'un df avec les données uniquement de la Thaïlande
Thai = fs.loc[fs['Zone']== 'Thaïlande']
```

In [55]:

```python
# Création d'un df en filtrant les produits pour avoir le Manioc uniquement

dispo_Thai = Thai.loc[fs['Produit']== 'Manioc']
#dispo_Thai
```

In [56]:

```python
# Création d'un df avec les données de population pour la Thaïlande

Thai_un_pop = tp_un.loc[tp_un['Zone']== 'Thaïlande']
#Thai_un_pop
```

In [57]:

```python
# Propotion de population en sous nutrition en Thaïlande

print('Population en sous nutrition en Thaïlande:' , round((Thai_un_pop['sous_nutrition'] * 100 / Thai_un_pop['Population']).mean(),2),'%')
```

```
Population en sous nutrition en Thaïlande: 8.91 %
```

In [58]:

```python
#  Proportion d'exportation du Manioc en Thaïlande

print("Proportion de la production par rapport à l'exportation pour la manic en Thaïlande : " , round(dispo_Thai['Exportations - Quantité'].sum() * 100 / dispo_Thai['Production'].sum(),2), '%')
```

```
Proportion de la production par rapport à l'exportation pour la manic en Thaïlande :  83.41 %
```

Question 7 : Pays ou la proportion de personnes sous-alimentées est la plus forte en 2017

In [43]:

```python
# Création d'un df avec la proportion de la population en sous nutrition par pays

tp_2017['proportion en %'] = round((tp_2017['sous_nutrition'] * 100) / (tp_2017['Population']),2)
#tp_2017
```

In [99]:

```python
# Classement des pays par pourcentage de population en sous nutrition (décroissant)

print('Top 10 des pays ou le taux des personnes sous alimentées est la plus forte')
tp_2017.sort_values(by = ['proportion en %'], ascending=False).head(10)
```

```
Top 10 des pays ou le taux des personnes sous alimentées est la plus forte
```

Out[99]:

	Zone	Population	sous_nutrition	proportion en %
78	Haïti	10982366.0	5300000.0	48.26
157	République populaire démocratique de Corée	25429825.0	12000000.0	47.19
108	Madagascar	25570512.0	10500000.0	41.06
103	Libéria	4702226.0	1800000.0	38.28
100	Lesotho	2091534.0	800000.0	38.25
183	Tchad	15016753.0	5700000.0	37.96
161	Rwanda	11980961.0	4200000.0	35.06
121	Mozambique	28649018.0	9400000.0	32.81
186	Timor-Leste	1243258.0	400000.0	32.17
0	Afghanistan	36296113.0	10500000.0	28.93

Question 8 : Pays ayant le plus bénéficié d'aide depuis 2013

In [47]:

```python
# Création d'un df avec les pays et le montant de l'aide

fa_pays = fa.groupby('Pays bénéficiaire').sum()
del fa_pays['Année']
#fa_pays
```

In [48]:

```python
# Création d'un df avecles pays et le montant de l'aide (décroissant)

class_pays_aide = fa_pays.sort_values(by=['Valeur'], ascending=False)
print("Top 10 des pays ayant reçu le plus d'aide depuis 2013 :")
class_pays_aide.head(10)
```

```
Top 10 des pays ayant reçu le plus d'aide depuis 2013 :
```

Out[48]:

Pays bénéficiaire	Valeur
République arabe syrienne	1858943
Éthiopie	1381294
Yémen	1206484
Soudan du Sud	695248
Soudan	669784
Kenya	552836
Bangladesh	348188
Somalie	292678
République démocratique du Congo	288502
Niger	276344

Question 9 : Pays ayant le plus / moins de disponibilité par habitant

In [49]:

```python
# Classement des pays (décroissant) selon leur disponibilité alimentaire en utilisant un pivot_table

dispo_habi = fs_pop.pivot_table('Disponibilité alimentaire (Kcal/personne/jour)', index = 'Zone' , aggfunc = 'sum')
print('pays ayant le plus de disponibilité par habitant')
dispo_habi.sort_values(by = ['Disponibilité alimentaire (Kcal/personne/jour)'], ascending = False).head(10)
```

```
pays ayant le plus de disponibilité par habitant
```

Out[49]:

Zone	Disponibilité alimentaire (Kcal/personne/jour)
Autriche	3770.0
Belgique	3737.0
Turquie	3708.0
États-Unis d'Amérique	3682.0
Israël	3610.0
Irlande	3602.0
Italie	3578.0
Luxembourg	3540.0
Égypte	3518.0
Allemagne	3503.0

In [50]:

```python
## Classement des pays (croissant) selon leur disponibilité alimentaire en utilisant un pivot_table

dispo_habi = fs_pop.pivot_table('Disponibilité alimentaire (Kcal/personne/jour)', index = 'Zone' , aggfunc = 'sum')
print('Pays ayant le moins de disponibilité par habitant')
dispo_plus = dispo_habi.sort_values(by = ['Disponibilité alimentaire (Kcal/personne/jour)'], ascending = True).head(10)
dispo_plus
```

```
Pays ayant le moins de disponibilité par habitant
```

Out[50]:

Zone	Disponibilité alimentaire (Kcal/personne/jour)
République centrafricaine	1879.0
Zambie	1924.0
Madagascar	2056.0
Afghanistan	2087.0
Haïti	2089.0
République populaire démocratique de Corée	2093.0
Tchad	2109.0
Zimbabwe	2113.0
Ouganda	2126.0
Timor-Leste	2129.0