PROJET 4 DATA ANALYST

Réalisez une étude de santé publique avec R ou Python

OBJECTIF DE CE NOTEBOOK

Bienvenue dans l'outil plébiscité par les analystes de données Jupyter.

Il s'agit d'un outil permettant de mixer et d'alterner codes, textes et graphique.

Cet outil est formidable pour plusieurs raisons:

- il permet de tester des lignes de codes au fur et à mesure de votre rédaction, de constater immédiatement le résultat d'un instruction, de la corriger si nécessaire.
- De rédiger du texte pour expliquer l'approche suivie ou les résultats d'une analyse et de le mettre en forme grâce à du code html ou plus simple avec **Markdown**
- d'agrémenter de graphiques

Pour vous aider dans vos premiers pas à l'usage de Jupyter et de Python, nous avons rédigé ce notebook en vous indiquant les instructions à suivre.

Il vous suffit pour cela de saisir le code Python répondant à l'instruction donnée.

Vous verrez de temps à autre le code Python répondant à une instruction donnée mais cela est fait pour vous aider à comprendre la nature du travail qui vous est demandée.

Et garder à l'esprit, qu'il n'y a pas de solution unique pour résoudre un problème et qu'il y a autant de résolutions de problèmes que de développeurs ;)...

Note jeremy Est ce qu'il faut faire le calcul de la sous nutrition sur les pays qu'on a ? Est ce qu'il faut faire des graphiques ? Rajouter le soja La liste des céréales est difficile a trouver ...

Etape 1 - Importation des librairies et chargement des fichiers

1.1 - Importation des librairies

In [6]:
```
#Importation de la librairie Pandas
import pandas as pd
import matplotlib.pyplot as plt
```

1.2 - Chargement des fichiers CSV

In [8]:
```
#Importation du fichier population.csv
population = pd.read_csv('population.csv')
population
#Importation du fichier dispo_alimentaire.csv
dispo_alimentaire = pd.read_csv('dispo_alimentaire.csv')
dispo_alimentaire
#Importation du fichier aide_alimentaire.csv
aide_alimentaire = pd.read_csv('aide_alimentaire.csv')
aide_alimentaire
#Importation du fichier sous_nutrition.csv
sous_nutrition = pd.read_csv('sous_nutrition.csv')
```

Etape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier population

In [11]:
```
#Afficher les dimensions du dataset
print(population.shape)
print("Le tableau comporte {} observation(s) ou article(s)".format(population.shape[0]))
print("Le tableau comporte {} colonne(s)".format(population.shape[1]))
```

```
(1416, 3)
Le tableau comporte 1416 observation(s) ou article(s)
Le tableau comporte 3 colonne(s)
```

In [12]:
```
#Consulter le nombre de colonnes
len(population.columns)
print("Il y a {} colonnes dans le fichier population".format(population.shape[1]))

#La nature des données dans chacune des colonnes + #Le nombre de valeurs présentes dans chacune des colonnes
display (population.info())
```

```
Il y a 3 colonnes dans le fichier population
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1416 entries, 0 to 1415
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Zone    1416 non-null   object 
 1   Année   1416 non-null   int64  
 2   Valeur  1416 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 33.3+ KB
None
```

Voici mon interprétation: Il n'y a pas de valeur Null dans le fichier population

In [14]:
```
#Affichage les 5 premières lignes de la table
population.head()
```

Out[14]:

	Zone	Année	Valeur
0	Afghanistan	2013	32269.589
1	Afghanistan	2014	33370.794
2	Afghanistan	2015	34413.603
3	Afghanistan	2016	35383.032
4	Afghanistan	2017	36296.113

Commentaire: La colonne "Valeur" n'est pas assez explicite, je vais renommer la colonne par "Population".

In [16]:
```
#Changement du nom de la colonne Valeur par Population
population.rename(columns={'Valeur':'Population'}, inplace=True)
population.head()
```

Out[16]:

	Zone	Année	Population
0	Afghanistan	2013	32269.589
1	Afghanistan	2014	33370.794
2	Afghanistan	2015	34413.603
3	Afghanistan	2016	35383.032
4	Afghanistan	2017	36296.113

Le paramètre '*Inplace*' permet de modifier l'ancien nom de colonne 'Valeur' de façon permanente.

In [18]:
```
#Nous allons harmoniser les unités. Pour cela, nous avons décidé de multiplier la population par 1000
population[['Population']]*=1000
print(population)
```

```
            Zone  Année  Population
0     Afghanistan   2013  32269589.0
1     Afghanistan   2014  33370794.0
2     Afghanistan   2015  34413603.0
3     Afghanistan   2016  35383032.0
4     Afghanistan   2017  36296113.0
...           ...    ...         ...
1411     Zimbabwe   2014  13586707.0
1412     Zimbabwe   2015  13814629.0
1413     Zimbabwe   2016  14030331.0
1414     Zimbabwe   2017  14236595.0
1415     Zimbabwe   2018  14438802.0

[1416 rows x 3 columns]
```

Commentaire: La colonne '*Population*' exprimée en millions d'habitants

In [20]:
```
#Affichage des 5 premières lignes de la table pour voir les modifications
population.head()
```

Out[20]:

	Zone	Année	Population
0	Afghanistan	2013	32269589.0
1	Afghanistan	2014	33370794.0
2	Afghanistan	2015	34413603.0
3	Afghanistan	2016	35383032.0
4	Afghanistan	2017	36296113.0

Commentaire: Voici le dataframe 'population' modifié avec le nom de colonne Population et ses valeurs exprimées en millions d'habitants

2.2 - Analyse exploratoire du fichier disponibilité alimentaire

In [23]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(dispo_alimentaire.shape[0]))
print("Le tableau comporte {} colonne(s)".format(dispo_alimentaire.shape[1]))
#Consulter le nombre de colonnes
dispo_alimentaire.info()
```

```
Le tableau comporte 15605 observation(s) ou article(s)
Le tableau comporte 18 colonne(s)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15605 entries, 0 to 15604
Data columns (total 18 columns):
 #   Column                                                Non-Null Count  Dtype  
---  ------                                                --------------  -----  
 0   Zone                                                  15605 non-null  object 
 1   Produit                                               15605 non-null  object 
 2   Origine                                               15605 non-null  object 
 3   Aliments pour animaux                                 2720 non-null   float64
 4   Autres Utilisations                                   5496 non-null   float64
 5   Disponibilité alimentaire (Kcal/personne/jour)        14241 non-null  float64
 6   Disponibilité alimentaire en quantité (kg/personne/an)  14015 non-null  float64
 7   Disponibilité de matière grasse en quantité (g/personne/jour)  11794 non-null  float64
 8   Disponibilité de protéines en quantité (g/personne/jour)  11561 non-null  float64
 9   Disponibilité intérieure                              15382 non-null  float64
 10  Exportations - Quantité                               12226 non-null  float64
 11  Importations - Quantité                               14852 non-null  float64
 12  Nourriture                                            14015 non-null  float64
 13  Pertes                                                4278 non-null   float64
 14  Production                                            9180 non-null   float64
 15  Semences                                              2091 non-null   float64
 16  Traitement                                            2292 non-null   float64
 17  Variation de stock                                    6776 non-null   float64
dtypes: float64(15), object(3)
memory usage: 2.1+ MB
```

Commentaire: A part les colonnes '*Zone*','*Produit*', et '*Origine*' qui contiennet aucune valeurs non-null, tous les autres colonnes ont plusieurs lignes avec des valeurs manquantes ou non renseignées.

In [25]:
```
#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

Out[25]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité d protéines e quantit (g/personne/jou
0	Afghanistan	Abats Comestible	animale	NaN	NaN	5.0	1.72	0.20	0.7
1	Afghanistan	Agrumes, Autres	vegetale	NaN	NaN	1.0	1.29	0.01	0.0
2	Afghanistan	Aliments pour enfants	vegetale	NaN	NaN	1.0	0.06	0.01	0.0
3	Afghanistan	Ananas	vegetale	NaN	NaN	0.0	0.00	NaN	Nal
4	Afghanistan	Bananes	vegetale	NaN	NaN	4.0	2.70	0.02	0.0

Commentaire: on aperçoit des valeurs '*NaN*' c'est-à-dire des valeurs manquantes ou non renseignées

In [27]:
```
#remplacement des NaN dans le dataset par des 0
dispo_alimentaire.fillna(0, inplace=True)
dispo_alimentaire
```

Out[27]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibil protéir qu (g/personne
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	
...	...	...	...	...	...	...	...	...	
15600	Îles Salomon	Viande de Suides	animale	0.0	0.0	45.0	4.70	4.28	
15601	Îles Salomon	Viande de Volailles	animale	0.0	0.0	11.0	3.34	0.69	
15602	Îles Salomon	Viande, Autre	animale	0.0	0.0	0.0	0.06	0.00	
15603	Îles Salomon	Vin	vegetale	0.0	0.0	0.0	0.07	0.00	
15604	Îles Salomon	Épices, Autres	vegetale	0.0	0.0	4.0	0.48	0.21	

15605 rows × 18 columns

In [28]:
```
#Multiplication de toutes les lignes contenant des milliers de tonnes en Kg
dispo_alimentaire.loc[:,['Aliments pour animaux','Autres Utilisations','Disponibilité intérieure','Exportations
dispo_alimentaire
```

Out[28]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibil protéir qu (g/personne
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	
...	...	...	...	...	...	...	...	...	
15600	Îles Salomon	Viande de Suides	animale	0.0	0.0	45.0	4.70	4.28	
15601	Îles Salomon	Viande de Volailles	animale	0.0	0.0	11.0	3.34	0.69	
15602	Îles Salomon	Viande, Autre	animale	0.0	0.0	0.0	0.06	0.00	
15603	Îles Salomon	Vin	vegetale	0.0	0.0	0.0	0.07	0.00	
15604	Îles Salomon	Épices, Autres	vegetale	0.0	0.0	4.0	0.48	0.21	

15605 rows × 18 columns

Commentaire: **Les colonnes** qui étaient en milliers de tonnes (*'Aliments pour animaux','Autres Utilisations','Disponibilité intérieure','Exportations - Quantité','Importations - Quantité','Nourriture','Pertes','Production','Semences','Traitement'et 'Variation de stock'*) **ont été converties en kg**.

In [30]:
```
#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

Out[30]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité d protéines e quantit (g/personne/jou
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	0.7
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	0.0
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	0.0
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	0.0
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	0.0

Commentaire: Le dataFrame 'dispo_alimentaire' modifié

2.3 - Analyse exploratoire du fichier aide alimentaire

In [33]:
```
#Afficher les dimensions du dataset
print ("Le tableau comporte {} observation(s) ou article(s)".format(aide_alimentaire.shape[0]))
print ("Le tableau comporte {} colonne(s)".format(aide_alimentaire.shape[1]))
```

```
Le tableau comporte 1475 observation(s) ou article(s)
Le tableau comporte 4 colonne(s)
```

In [34]:
```
#Consulter le nombre de colonnes
aide_alimentaire.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1475 entries, 0 to 1474
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Pays bénéficiaire  1475 non-null   object
 1   Année              1475 non-null   int64 
 2   Produit            1475 non-null   object
 3   Valeur             1475 non-null   int64 
dtypes: int64(2), object(2)
memory usage: 46.2+ KB
```

Commentaire: Pas de valeur null dans le fichier aide alimentaire

In [36]:
```
#Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

Out[36]:

	Pays bénéficiaire	Année	Produit	Valeur
0	Afghanistan	2013	Autres non-céréales	682
1	Afghanistan	2014	Autres non-céréales	335
2	Afghanistan	2013	Blé et Farin	39224
3	Afghanistan	2014	Blé et Farin	15160
4	Afghanistan	2013	Céréales	40504

In [37]:
```
#Changement du nom de la colonne Pays bénéficiaire par Zone
aide_alimentaire.rename(columns = {'Pays bénéficiaire':'Zone','Valeur':'Quantité'}, inplace=True)
aide_alimentaire
```

Out[37]:

	Zone	Année	Produit	Quantité
0	Afghanistan	2013	Autres non-céréales	682
1	Afghanistan	2014	Autres non-céréales	335
2	Afghanistan	2013	Blé et Farin	39224
3	Afghanistan	2014	Blé et Farin	15160
4	Afghanistan	2013	Céréales	40504
...	...	...	...	...
1470	Zimbabwe	2015	Mélanges et préparations	96
1471	Zimbabwe	2013	Non-céréales	5022
1472	Zimbabwe	2014	Non-céréales	2310
1473	Zimbabwe	2015	Non-céréales	306
1474	Zimbabwe	2013	Riz, total	64

1475 rows × 4 columns

Commentaire: La colonne '*Pays Bénéficiaire*' a été modifiée par '*Zone*' pour harmoniser avec les autres fichiers csv qui ont toute cette colonne identique. La colonne 'Valeur' a été modifiée par 'Quantité'.

In [39]:
```
#Multiplication de la colonne Aide_alimentaire qui contient des tonnes par 1000 pour avoir des kg
aide_alimentaire['Quantité']*=1000
aide_alimentaire
```

Out[39]:

	Zone	Année	Produit	Quantité
0	Afghanistan	2013	Autres non-céréales	682000
1	Afghanistan	2014	Autres non-céréales	335000
2	Afghanistan	2013	Blé et Farin	39224000
3	Afghanistan	2014	Blé et Farin	15160000
4	Afghanistan	2013	Céréales	40504000
...	...	...	...	...
1470	Zimbabwe	2015	Mélanges et préparations	96000
1471	Zimbabwe	2013	Non-céréales	5022000
1472	Zimbabwe	2014	Non-céréales	2310000
1473	Zimbabwe	2015	Non-céréales	306000
1474	Zimbabwe	2013	Riz, total	64000

1475 rows × 4 columns

Commentaire: Les quantités sont exprimées en kg

In [41]:
```
#Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

Out[41]:

	Zone	Année	Produit	Quantité
0	Afghanistan	2013	Autres non-céréales	682000
1	Afghanistan	2014	Autres non-céréales	335000
2	Afghanistan	2013	Blé et Farin	39224000
3	Afghanistan	2014	Blé et Farin	15160000
4	Afghanistan	2013	Céréales	40504000

2.3 - Analyse exploratoire du fichier sous nutrition

In [43]:
```
#Consulter le nombre de colonnes
#Afficher les 5 premières lignes de la table
sous_nutrition.info()
sous_nutrition.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1218 entries, 0 to 1217
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Zone    1218 non-null   object
 1   Année   1218 non-null   object
 2   Valeur  624 non-null    object
dtypes: object(3)
memory usage: 28.7+ KB
```

Out[43]:

	Zone	Année	Valeur
count	1218	1218	624
unique	203	6	139
top	Afghanistan	2012-2014	<0.1
freq	6	203	120

Commentaire: La colonne '*Valeur*' ne contient que 624 valeurs au lieu de 1218 et elle est de type 'object', c'est-à-dire qu'il y a des valeurs de type string (chaîne de caractères). Avec la fonction '.Describe()' on remarque que parmi les 624 valeurs, 120 mentionnent '<0.1' qui sont des valeurs de type string à convertir en numérique.

In [45]:
```
#Remplacement des données string de la colonne Valeur pour pouvoir convertir la colonne entière de valeurs numé
sous_nutrition['Valeur'] = sous_nutrition['Valeur'].replace('<0.1', 0.1)
```

In [46]:
```
#Conversion de la colonne sous nutrition en numérique
sous_nutrition['Valeur'] = pd.to_numeric(sous_nutrition['Valeur'])
```

In [47]:
```
#Puis remplacement des NaN en 0
sous_nutrition.fillna(0, inplace=True)
print(sous_nutrition)
```

```
            Zone      Année  Valeur
0     Afghanistan  2012-2014     8.6
1     Afghanistan  2013-2015     8.8
2     Afghanistan  2014-2016     8.9
3     Afghanistan  2015-2017     9.7
4     Afghanistan  2016-2018    10.5
...           ...        ...     ...
1213     Zimbabwe  2013-2015     0.0
1214     Zimbabwe  2014-2016     0.0
1215     Zimbabwe  2015-2017     0.0
1216     Zimbabwe  2016-2018     0.0
1217     Zimbabwe  2017-2019     0.0

[1218 rows x 3 columns]
```

In [48]:
```
#Multiplication de la colonne sous_nutrition par 1000000
sous_nutrition['Valeur']*=1000000
sous_nutrition
```

Out[48]:

	Zone	Année	Valeur
0	Afghanistan	2012-2014	8600000.0
1	Afghanistan	2013-2015	8800000.0
2	Afghanistan	2014-2016	8900000.0
3	Afghanistan	2015-2017	9700000.0
4	Afghanistan	2016-2018	10500000.0
...	...	...	...
1213	Zimbabwe	2013-2015	0.0
1214	Zimbabwe	2014-2016	0.0
1215	Zimbabwe	2015-2017	0.0
1216	Zimbabwe	2016-2018	0.0
1217	Zimbabwe	2017-2019	0.0

1218 rows × 3 columns

Commentaire: La colonne 'Valeur' a été convertie en millions d'habitants en état de sous-nutrition pour harmoniser les unités avec le fichier '*population_csv*'.

In [50]:
```
#changement du nom de la colonne Valeur par sous_nutrition
sous_nutrition.rename(columns={'Valeur':'sous_nutrition'}, inplace=True)
```

```
sous_nutrition
```

Out[50]:

	Zone	Année	sous_nutrition
0	Afghanistan	2012-2014	8600000.0
1	Afghanistan	2013-2015	8800000.0
2	Afghanistan	2014-2016	8900000.0
3	Afghanistan	2015-2017	9700000.0
4	Afghanistan	2016-2018	10500000.0
...	...	...	...
1213	Zimbabwe	2013-2015	0.0
1214	Zimbabwe	2014-2016	0.0
1215	Zimbabwe	2015-2017	0.0
1216	Zimbabwe	2016-2018	0.0
1217	Zimbabwe	2017-2019	0.0

1218 rows × 3 columns

Commentaire: Renommer la colonne '*Valeur*' par '*sous_nutrition*'

In [52]:
```
sous_nutrition.head()
```

Out[52]:

	Zone	Année	sous_nutrition
0	Afghanistan	2012-2014	8600000.0
1	Afghanistan	2013-2015	8800000.0
2	Afghanistan	2014-2016	8900000.0
3	Afghanistan	2015-2017	9700000.0
4	Afghanistan	2016-2018	10500000.0

Commentaire: La colonne 'sous_nutrition' est exprimée en millions d'habitants

Etape 3 - Analyse de la sous nutrition dans le monde entre 2013 et 2017

3.1 - Proportion de personnes en sous nutrition

In [56]:
```
# Création de la variable sous_nutrition pour l'année 2017
ss_nutrition_2017 = sous_nutrition.loc[sous_nutrition['Année'] == '2016-2018',['Zone','sous_nutrition']]
ss_nutrition_2017.rename(columns={'sous_nutrition':'sous_nutrition_2017'},inplace = True)
ss_nutrition_2017
```

Out[56]:

	Zone	sous_nutrition_2017
4	Afghanistan	10500000.0
10	Afrique du Sud	3100000.0
16	Albanie	100000.0
22	Algérie	1300000.0
28	Allemagne	0.0
...	...	...
1192	Venezuela (République bolivarienne du)	8000000.0
1198	Viet Nam	6500000.0
1204	Yémen	0.0
1210	Zambie	0.0
1216	Zimbabwe	0.0

203 rows × 2 columns

1ère étape : je crée le dataframe '*sous_nutrition_2017*' pour cibler l'année 2017 avec la fonction '*loc.*'qui va filtrer le nombre de personnes en sous nutrititon pour l'année 2017 dans l'intervalle '2016-2018'.

In [58]:
```
# Création du DataFrame 'Population' pour l'année 2017
pop_2017 = population.loc[population['Année'] == 2017,['Zone','Population']]
pop_2017
```

Out[58]:

	Zone	Population
4	Afghanistan	36296113.0
10	Afrique du Sud	57009756.0
16	Albanie	2884169.0
22	Algérie	41389189.0
28	Allemagne	82658409.0
...	...	...
1390	Venezuela (République bolivarienne du)	29402484.0
1396	Viet Nam	94600648.0
1402	Yémen	27834819.0
1408	Zambie	16853599.0
1414	Zimbabwe	14236595.0

236 rows × 2 columns

2ème étape: Création d'une dataFrame 'pop_2017' avec uniquement le nombre d'habitants en 2017 pour chaque pays

In [60]:
```
# Il faut tout d'abord faire une jointure entre la table population et la table sous nutrition, en ciblant l'an
pop_ss_nutrition_2017 = pop_2017.merge(ss_nutrition_2017, on= 'Zone')
pop_ss_nutrition_2017
```

Out[60]:

	Zone	Population	sous_nutrition_2017
0	Afghanistan	36296113.0	10500000.0
1	Afrique du Sud	57009756.0	3100000.0
2	Albanie	2884169.0	100000.0
3	Algérie	41389189.0	1300000.0
4	Allemagne	82658409.0	0.0
...	...	...	...
198	Venezuela (République bolivarienne du)	29402484.0	8000000.0
199	Viet Nam	94600648.0	6500000.0
200	Yémen	27834819.0	0.0
201	Zambie	16853599.0	0.0
202	Zimbabwe	14236595.0	0.0

203 rows × 3 columns

3ème étape: Je joins les dataframes 'pop_2017' et 'ss_nutrition_2017'pour créer la nouvelle dataframe 'pop_ss_nutrition_2017'.

In [62]:
```
# Affichage du dataset avec le dataset filtré par population en 2017 et par nombre de personne en état de sous
pop_ss_nutrition_2017
```

Out[62]:

	Zone	Population	sous_nutrition_2017
0	Afghanistan	36296113.0	10500000.0
1	Afrique du Sud	57009756.0	3100000.0
2	Albanie	2884169.0	100000.0
3	Algérie	41389189.0	1300000.0
4	Allemagne	82658409.0	0.0
...	...	...	...
198	Venezuela (République bolivarienne du)	29402484.0	8000000.0
199	Viet Nam	94600648.0	6500000.0
200	Yémen	27834819.0	0.0
201	Zambie	16853599.0	0.0
202	Zimbabwe	14236595.0	0.0

203 rows × 3 columns

In [63]:
```
# Création de la colonne 'Pourcentage_sous_nutrition_2017' et calcul du pourcentage dans le dataFrame "pop_ss_n
pop_ss_nutrition_2017['Pourcentage_sous_nutrition_2017'] = round((pop_ss_nutrition_2017['sous_nutrition_2017']/
#Calcul et affichage du nombre de personnes en état de sous nutrition
Proportion_ss_nutrition_2017 = round((pop_ss_nutrition_2017['sous_nutrition_2017'].sum() / pop_ss_nutrition_201
Proportion_ss_nutrition_2017
```

Out[63]: 7.13

Interprétation: En 2017,la proportion de personne en état de sous nutrition dans le monde est de 7.13%

3.2 - Nombre théorique de personne qui pourrait être nourries

In [66]:
```
#Combien mange en moyenne un être humain ? Source =>
Besoin_Kcal = 2250
```

Commentaire: Selon la FAO, les femmes ont en moyenne besoin de 2000kcal/jour et les hommes de 2500kcal/jour. La moyenne pour un être humain est donc de 2250kcal/jour

In [68]:
```
#On commence par faire une jointure entre le data frame population et Dispo_alimentaire afin d'ajouter dans ce
dispo_alimentaire_2017 = dispo_alimentaire.merge(pop_2017, on='Zone')
```

Commentaire: Jointure entre le dataframe 'dispo_alimentaire' et 'pop_2017' sur la colonne 'Zone' commune aux 2 dataframes.

In [70]:
```
#Affichage du nouveau dataframe
dispo_alimentaire_2017.head()
```

Out[70]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité d protéines e quantit (g/personne/jou
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	0.7
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	0.0
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	0.0
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	0.0
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	0.0

In [71]:
```
#Création de la colonne dispo_kcal avec calcul des kcal disponibles mondialement
dispo_alimentaire_2017['Dispo_Kcal'] = dispo_alimentaire_2017.groupby(['Zone'])['Disponibilité alimentaire (Kca
#Convertir la disponibilité kcal pers/jours par pers/an
dispo_alimentaire_2017['Dispo_Kcal'] *=365
dispo_alimentaire_2017
```

Out[71]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibil protéir qu (g/personne
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	
...	...	...	...	...	...	...	...	...	
15411	Îles Salomon	Viande de Suides	animale	0.0	0.0	45.0	4.70	4.28	
15412	Îles Salomon	Viande de Volailles	animale	0.0	0.0	11.0	3.34	0.69	
15413	Îles Salomon	Viande, Autre	animale	0.0	0.0	0.0	0.06	0.00	
15414	Îles Salomon	Vin	vegetale	0.0	0.0	0.0	0.07	0.00	
15415	Îles Salomon	Épices, Autres	vegetale	0.0	0.0	4.0	0.48	0.21	

15416 rows × 20 columns

Commentaire: La Colonne 'Dispo_Kcal' représente le total des disponibilité Kcal/pers/an pour un pays donnée

In [73]:
```
#Calcul du nombre d'humains pouvant être nourris
Nb_nourris_2017 = (dispo_alimentaire_2017['Dispo_Kcal']*dispo_alimentaire_2017['Population']/ (Besoin_Kcal*365)
Pop_nourrie_2017 = Nb_nourris_2017.drop_duplicates().sum()
print(Pop_nourrie_2017)
```

```
9297326501.036001
```

Interprétation: Il y a 9.297326501 milliards d'habitants qui peuvent être nourris dans le monde en 2017.

In [75]:
```
#Proportion de personnes pouvant être nourries
Proportion_nourries = round((Pop_nourrie_2017/pop_2017['Population'].sum())*100,2)
Proportion_nourries
```

Out[75]:
```
123.17
```

Interprétation: il y a 123.17% de personnes pouvant être nourries dans le monde en 2017.

3.3 - Nombre théorique de personne qui pourrait être nourrie avec les produits végétaux

In [78]:
```
#Transfert des données avec les végétaux dans un nouveau dataframe
df_Dispo_végétaux = dispo_alimentaire_2017.loc[dispo_alimentaire_2017['Origine'] =='vegetale', ['Zone','Produit
df_Dispo_végétaux
```

Out[78]:

	Zone	Produit	Population	Disponibilité alimentaire (Kcal/personne/jour)
1	Afghanistan	Agrumes, Autres	36296113.0	1.0
2	Afghanistan	Aliments pour enfants	36296113.0	1.0
3	Afghanistan	Ananas	36296113.0	0.0
4	Afghanistan	Bananes	36296113.0	4.0
6	Afghanistan	Bière	36296113.0	0.0
...	...	...	...	...
15406	Îles Salomon	Sésame	636039.0	0.0
15407	Îles Salomon	Thé	636039.0	0.0
15408	Îles Salomon	Tomates	636039.0	0.0
15414	Îles Salomon	Vin	636039.0	0.0
15415	Îles Salomon	Épices, Autres	636039.0	4.0

11751 rows × 4 columns

In [79]:
```
#Calcul du nombre de kcal disponible pour les végétaux
df_Dispo_végétaux['Dispo_Kcal_végétaux'] = df_Dispo_végétaux.groupby(['Zone','Population'])['Disponibilité alim
#Convertir la colonne Dispo_Kcal_végétaux en année
df_Dispo_végétaux[['Dispo_Kcal_végétaux']] *=365
df_Dispo_végétaux
```

Out[79]:

	Zone	Produit	Population	Disponibilité alimentaire (Kcal/personne/jour)	Dispo_Kcal_végétaux
1	Afghanistan	Agrumes, Autres	36296113.0	1.0	682915.0
2	Afghanistan	Aliments pour enfants	36296113.0	1.0	682915.0
3	Afghanistan	Ananas	36296113.0	0.0	682915.0
4	Afghanistan	Bananes	36296113.0	4.0	682915.0
6	Afghanistan	Bière	36296113.0	0.0	682915.0
...	...	...	...	...	...
15406	Îles Salomon	Sésame	636039.0	0.0	798255.0
15407	Îles Salomon	Thé	636039.0	0.0	798255.0
15408	Îles Salomon	Tomates	636039.0	0.0	798255.0
15414	Îles Salomon	Vin	636039.0	0.0	798255.0
15415	Îles Salomon	Épices, Autres	636039.0	4.0	798255.0

11751 rows × 5 columns

In [80]:
```
#Calcul du nombre d'humains pouvant être nourris avec les végétaux
Nb_nourris_végétaux_2017 = (df_Dispo_végétaux['Dispo_Kcal_végétaux']* df_Dispo_végétaux ['Population'])/ (Besoi
# les lignes dans la colonne 'Population' et Dispo_Kcal_végétaux' se répètent donc on supprime les doublons.
Nb_nourris_végétaux_2017.drop_duplicates().sum()
#Calcul de la proportion de personnes pouvant être nourries avec des végétaux
Proportion_nourris_végétaux_2017 = round((Nb_nourris_végétaux_2017.drop_duplicates().sum()/pop_2017['Population
Proportion_nourris_végétaux_2017
```

Out[80]:
```
101.63
```

Interprétation: il y a 101.63% de la population mondiale pouvant être nourries avec des végétaux en 2017.

3.4 - Utilisation de la disponibilité intérieure

Définitions du concept de "disponibilité intérieure" (lien de la FAO)

https://www.fao.org/4/x9892f/x9892f03.htm#TopOfPage

In [84]:
```
#Calcul de la disponibilité totale
dispo_intérieure_totale = dispo_alimentaire['Disponibilité intérieure'].sum()
dispo_intérieure_totale
```

Out[84]:
```
9848994000000.0
```

Interprétation: Il y a 9.848994000 milliards de kg d'aliments disponible dans le monde

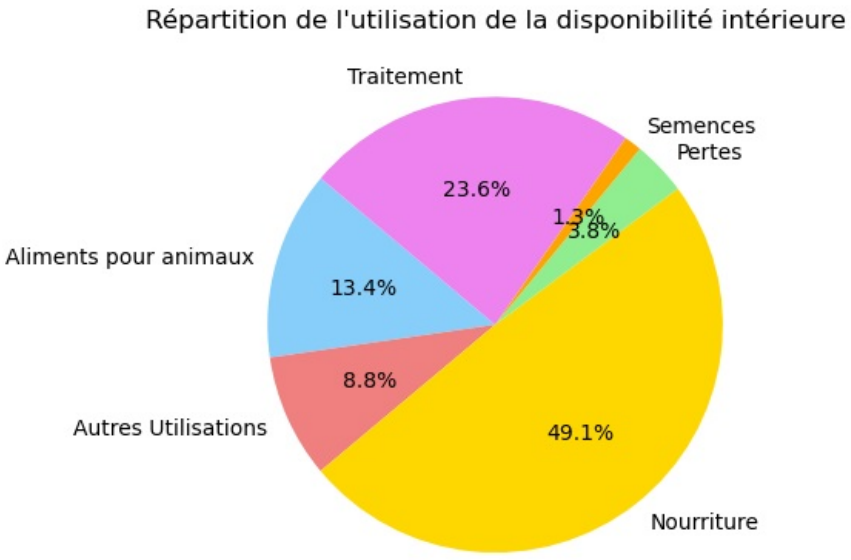
In [86]:
```
#Création d'une boucle for pour afficher les différentes valeurs en fonction des colonnes aliments pour animaux,
Proportion_utilisation_intérieure = dispo_alimentaire.loc[:,['Aliments pour animaux','Autres Utilisations','Nou

for elt in Proportion_utilisation_intérieure:
    Proportion_utilisation_intérieure[elt]=round((Proportion_utilisation_intérieure[elt]/ dispo_intérieure_tota
    print(Proportion_utilisation_intérieure[[elt]].sum())
```

```
Aliments pour animaux    12.41
dtype: float64
Autres Utilisations    8.12
dtype: float64
Nourriture    45.4
dtype: float64
Pertes    3.56
dtype: float64
Semences    1.16
dtype: float64
Traitement    21.83
dtype: float64
```

In [87]:
```
#Création du graphique pour les répartitions de la disponibilité intérieure dans le monde
proportions = Proportion_utilisation_intérieure.sum()
plt.pie(proportions, labels=proportions.index, autopct='%1.1f%%', startangle=140, colors=['lightskyblue', 'ligh
plt.title("Répartition de l'utilisation de la disponibilité intérieure")
plt.figure(figsize=(5, 5))
```

Out[87]:
```
<Figure size 500x500 with 0 Axes>
```




```
<Figure size 500x500 with 0 Axes>
```

3.5 - Utilisation des céréales

In [89]:
```
#Création d'une liste avec toutes les variables
Céréales = ['Blé','Seigle','Orge','Avoine','Maïs','Riz (Eq Blanchi)','Sorgho','Millet','Céréales Autres']
```

In [90]:
```
#Création d'un dataframe avec les informations uniquement pour ces céréales
Df_céréales = dispo_alimentaire[dispo_alimentaire['Produit'].isin(Céréales)]
Df_céréales.head()
```

Out[90]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité protéines quant (g/personne/jo
7	Afghanistan	Blé	vegetale	0.0	0.0	1369.0	160.23	4.69	36
32	Afghanistan	Maïs	vegetale	200000000.0	0.0	21.0	2.50	0.30	0
34	Afghanistan	Millet	vegetale	0.0	0.0	3.0	0.40	0.02	0
40	Afghanistan	Orge	vegetale	360000000.0	0.0	26.0	2.92	0.24	0
47	Afghanistan	Riz (Eq Blanchi)	vegetale	0.0	0.0	141.0	13.82	0.27	2

In [91]:
```
#Affichage de la proportion d'alimentation animale
```

```
Proportion_alimentation_animale = round((Df_céréales['Aliments pour animaux'].sum()/ Df_céréales['Disponibilité
Proportion_alimentation_animale
```

Out[91]: 35.91

Commentaire: 35.91% de la disponibilité intérieure totale de céréales est consacré à l'alimentation des animaux

In [93]:
```
#Affichage de la proportion d'alimentation humaine
Proportion_alimentation_humaine = round((Df_céréales['Nourriture'].sum()/ Df_céréales['Disponibilité intérieure
Proportion_alimentation_humaine
```

Out[93]: 43.02

Commentaire: 43.02% de la disponibilité intérieure totale en céréales est consacré à l'alimentation humaine

3.6 - Pays avec la proportion de personnes sous-alimentée la plus forte en 2017

In [96]:
```
#Création de la colonne proportion par pays
pop_ss_nutrition_2017['Pourcentage_sous_nutrition_2017'] = round((pop_ss_nutrition_2017['sous_nutrition_2017']/
pop_ss_nutrition_2017
```

Out[96]:

	Zone	Population	sous_nutrition_2017	Pourcentage_sous_nutrition_2017
0	Afghanistan	36296113.0	10500000.0	28.93
1	Afrique du Sud	57009756.0	3100000.0	5.44
2	Albanie	2884169.0	100000.0	3.47
3	Algérie	41389189.0	1300000.0	3.14
4	Allemagne	82658409.0	0.0	0.00
...	...	...	...	...
198	Venezuela (République bolivarienne du)	29402484.0	8000000.0	27.21
199	Viet Nam	94600648.0	6500000.0	6.87
200	Yémen	27834819.0	0.0	0.00
201	Zambie	16853599.0	0.0	0.00
202	Zimbabwe	14236595.0	0.0	0.00

203 rows × 4 columns

In [97]:
```
#Affichage après tri des 10 pires pays
proportion_pires_pays = pop_ss_nutrition_2017.sort_values('Pourcentage_sous_nutrition_2017',ascending= False).h
proportion_pires_pays
```

Out[97]:

	Zone	Population	sous_nutrition_2017	Pourcentage_sous_nutrition_2017
51	Dominique	71458.0	100000.0	139.94
164	Saint-Vincent-et-les Grenadines	109827.0	100000.0	91.05
98	Kiribati	114158.0	100000.0	87.60
167	Sao Tomé-et-Principe	207089.0	100000.0	48.29
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

3.7 - Pays qui ont le plus bénéficié d'aide alimentaire depuis 2013

In [99]:
```
#Calcul du total de l'aide alimentaire par pays
Total_aide_par_pays = aide_alimentaire.groupby(['Zone'])[['Quantité']].sum()
```

In [100…
```
#Affichage après tri des 10 pays qui ont bénéficié le plus de l'aide alimentaire
aide_alimentaire.groupby(['Zone'])[['Quantité']].sum().sort_values('Quantité',ascending= False).head(10)
```

Out[100…

Zone	Quantité
République arabe syrienne	1858943000
Éthiopie	1381294000
Yémen	1206484000
Soudan du Sud	695248000
Soudan	669784000
Kenya	552836000
Bangladesh	348188000
Somalie	292678000
République démocratique du Congo	288502000
Niger	276344000

3.8 - Evolution des 5 pays qui ont le plus bénéficiés de l'aide alimentaire entre 2013 et 2016

In [102…

```
#Création d'un dataframe avec la zone, l'année et l'aide alimentaire puis groupby sur zone et année
aide_alimentaire_pays_année = aide_alimentaire.groupby(['Zone','Année'])[['Quantité']].sum()
aide_alimentaire_pays_année
```

Out[102…

Zone	Année	Quantité
Afghanistan	**2013**	128238000
	2014	57214000
Algérie	**2013**	35234000
	2014	18980000
	2015	17424000
...	**...**	...
Égypte	**2013**	1122000
Équateur	**2013**	1362000
Éthiopie	**2013**	591404000
	2014	586624000
	2015	203266000

228 rows × 1 columns

In [103…

```
#Création d'une liste contenant les 5 pays qui ont le plus bénéficiées de l'aide alimentaire
Liste_pays_aidés = ['République arabe syrienne', 'Éthiopie','Yémen','Soudan du Sud', 'Soudan']
```

In [104…

```
#On filtre sur le dataframe avec notre liste
df_top_pays_aidés = aide_alimentaire[aide_alimentaire['Zone'].isin(Liste_pays_aidés)]
df_top_pays_aidés
```

Out[104…

	Zone	Année	Produit	Quantité
354	Éthiopie	2013	Autres non-céréales	170000
355	Éthiopie	2014	Autres non-céréales	466000
356	Éthiopie	2015	Autres non-céréales	244000
357	Éthiopie	2013	Blé et Farin	181066000
358	Éthiopie	2014	Blé et Farin	178646000
...	...	...	...	...
1447	Yémen	2015	Riz, total	1797000
1448	Yémen	2013	Sucre, total	161000
1449	Yémen	2014	Sucre, total	81000
1450	Yémen	2015	Sucre, total	7617000
1451	Yémen	2016	Sucre, total	2675000

155 rows × 4 columns

In [105…

```
# Affichage des pays avec l'aide alimentaire par année
Evolution_aide_pays = df_top_pays_aidés.pivot_table(index= ['Année'],columns=['Zone'], values=['Quantité'], agg
Evolution_aide_pays = Evolution_aide_pays.reset_index()
Evolution_aide_pays
```

Out[105…

	Année	Quantité				
Zone		**République arabe syrienne**	**Soudan**	**Soudan du Sud**	**Yémen**	**Éthiopie**
0	2013	563566000.0	330230000.0	196330000.0	264764000.0	591404000.0
1	2014	651870000.0	321904000.0	450610000.0	103840000.0	586624000.0
2	2015	524949000.0	17650000.0	48308000.0	372306000.0	203266000.0
3	2016	118558000.0	NaN	NaN	465574000.0	NaN

In [106…

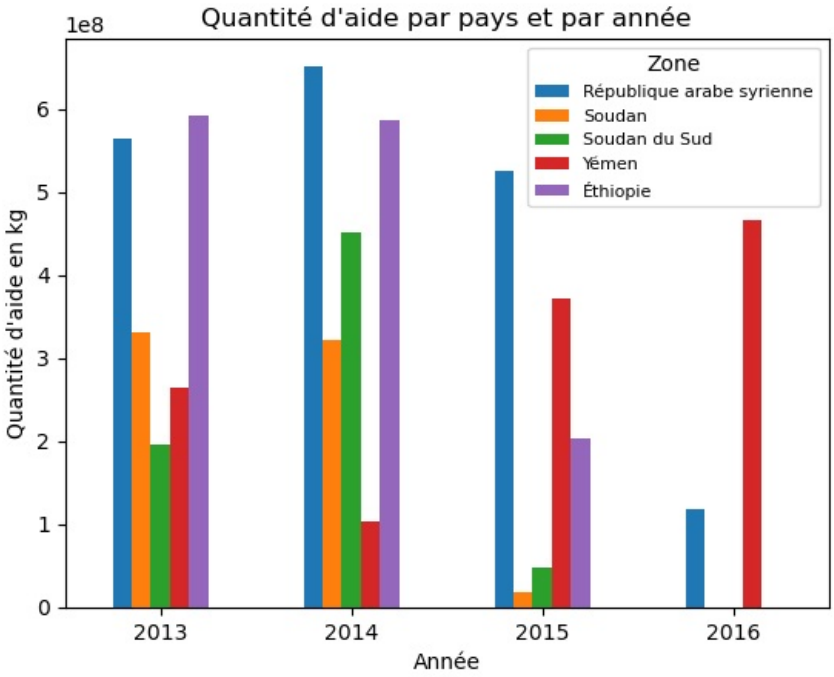
```
#Création d'un graphique en diagramme en barres empilées pour montrer l'évolution de l'aide alimentaire pour le

# Définir "Année" comme index
Evolution_aide_pays.set_index("Année",inplace=True)

# Tracer le diagramme en barres empilées
Evolution_aide_pays.plot(kind="bar",stacked=False)

# Ajout des titres et labels sur le graphique
plt.title("Quantité d'aide par pays et par année")
plt.xlabel("Année")
plt.ylabel("Quantité d'aide en kg")
plt.legend(['République arabe syrienne','Soudan','Soudan du Sud','Yémen','Éthiopie'],title="Zone", loc=0, fonts
plt.xticks(rotation=0)  # Garde les années bien lisibles

# Affichage du graphique
plt.show()
```



3.9 - Pays avec le moins de disponibilité par habitant

In [108…
```
#Calcul de la disponibilité en kcal par personne par jour par pays
dispo_alimentaire_pays = dispo_alimentaire.groupby(['Zone'])[['Disponibilité alimentaire (Kcal/personne/jour)']
dispo_alimentaire_pays
```

Out[108…

Zone	Disponibilité alimentaire (Kcal/personne/jour)
Afghanistan	2087.0
Afrique du Sud	3020.0
Albanie	3188.0
Algérie	3293.0
Allemagne	3503.0
...	...
Émirats arabes unis	3275.0
Équateur	2346.0
États-Unis d'Amérique	3682.0
Éthiopie	2129.0
Îles Salomon	2383.0

174 rows × 1 columns

In [246…
```
#Affichage des 10 pays qui ont le moins de dispo alimentaire par personne
pays_moins_dispo_alimentaire = dispo_alimentaire.groupby(['Zone'])[['Disponibilité alimentaire (Kcal/personne/j
pays_moins_dispo_alimentaire.reset_index()
```

Out[246…

	Zone	Disponibilité alimentaire (Kcal/personne/jour)
0	République centrafricaine	1879.0
1	Zambie	1924.0
2	Madagascar	2056.0
3	Afghanistan	2087.0
4	Haïti	2089.0
5	République populaire démocratique de Corée	2093.0
6	Tchad	2109.0
7	Zimbabwe	2113.0
8	Ouganda	2126.0
9	Timor-Leste	2129.0

3.10 - Pays avec le plus de disponibilité par habitant

In [111…

```
#Affichage des 10 pays qui ont le plus de dispo alimentaire par personne
pays_plus_dispo_alimentaire = dispo_alimentaire.groupby(['Zone'])[['Disponibilité alimentaire (Kcal/personne/jo
pays_plus_dispo_alimentaire
```

Out[111…

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

3.11 - Exemple de la Thaïlande pour le Manioc

In [113…

```
#Création d'un dataframe avec uniquement la Thaïlande
dispo_alimentaire_Thaïlande = dispo_alimentaire.loc[dispo_alimentaire['Zone'] == 'Thaïlande',:]
dispo_alimentaire_Thaïlande
```

Out[113…

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibili protéin qua (g/personne/
13759	Thaïlande	Abats Comestible	animale	0.0	0.0	3.0	1.11	0.09	
13760	Thaïlande	Agrumes, Autres	vegetale	0.0	0.0	0.0	0.09	0.00	
13761	Thaïlande	Alcool, non Comestible	vegetale	0.0	358000000.0	0.0	0.00	0.00	
13762	Thaïlande	Aliments pour enfants	vegetale	0.0	0.0	2.0	0.18	0.01	
13763	Thaïlande	Ananas	vegetale	0.0	0.0	10.0	10.02	0.04	
...	...	...	...	...	...	...	...	...	
13849	Thaïlande	Viande de Suides	animale	0.0	0.0	124.0	13.00	11.83	
13850	Thaïlande	Viande de Volailles	animale	0.0	0.0	52.0	13.69	3.62	
13851	Thaïlande	Viande, Autre	animale	0.0	0.0	0.0	0.03	0.01	
13852	Thaïlande	Vin	vegetale	0.0	0.0	0.0	0.12	0.00	
13853	Thaïlande	Épices, Autres	vegetale	0.0	0.0	16.0	1.70	0.30	

95 rows × 18 columns

In [114…
```
#Calcul de la disponibilité par habitant pour la Thaïlande ?
dispo_alimentaire_Thaïlande['Disponibilité alimentaire (Kcal/personne/jour)'].sum()
```

Out[114… 2785.0

Commentaire: La disponibilité Kcal est de 2785/pers/jour en Thaïlande. Donc il y a assez de ressources pour chaque habitant.

In [116…
```
#Création des variables pour la population et la sous-nutrition en Thaïlande en 2017
Pop_Thaïlande = population.loc[(population['Zone']== 'Thaïlande') & (population['Année']== 2017),]
ss_nutrition_Thaïlande = sous_nutrition.loc[(sous_nutrition['Zone'] == 'Thaïlande') & (sous_nutrition['Année']==

#Fusion des 2 datasets population thaïlandaise et la sous-nutrition
ss_nutrition_Thaïlande_2017 = pd.merge(Pop_Thaïlande,ss_nutrition_Thaïlande, on= 'Zone')

#Calcul de la proportion de personnes sous nourries en Thaïlande en 2017
proportion_ss_nutrition_Thaïlande_2017 = round((ss_nutrition_Thaïlande_2017['sous_nutrition']/ss_nutrition_Thaï
print(proportion_ss_nutrition_Thaïlande_2017)
```

```
0    8.96
dtype: float64
```

En 2017, près de 9% de la population totale souffrait de sous-nutrition en Thaïlande, la proportion est plus elevé que celui de la sous-nutrition dans le monde.

In [118…
```
# On calcule la proportion exportée en fonction des exportations totales en Thaïlande
dispo_manioc = dispo_alimentaire_Thaïlande.loc[dispo_alimentaire_Thaïlande['Produit'] == 'Manioc',:]
Proportion_export_manioc = round((dispo_manioc['Exportations - Quantité']/dispo_alimentaire_Thaïlande ['Exporta
Proportion_export_manioc.sum()
```

Out[118… 50.0

Interprétation: En Thaïlande, le manioc représente 50% des exportations totales alors que le pays peut garder cette ressource pour alimenter les 8.96% de thaïlandais en état de sous-nutrition

Etape 4 - Analyses complémentaires

4.1 -Evolution de l'aide alimentaire dans le monde

In [122…
```
#Calcul du total de l'aide alimentaire par année dans le monde
total_aide_alimentaire = aide_alimentaire.loc [:,['Année','Quantité']]
total_aide_alimentaire = total_aide_alimentaire.groupby('Année') [['Quantité']].sum().reset_index() #reset_inde
```

```
total_aide_alimentaire
```

Out[122…

	Année	Quantité
0	2013	4165674000
1	2014	3939152000
2	2015	2187507000
3	2016	743568000

In [123…

```
#Création de variable pour le total de l'aide alimentaire sur l'année 2013 et l'année 2017.
aide_alimentaire_2013 = total_aide_alimentaire.iloc[0,1]
aide_alimentaire_2017 = total_aide_alimentaire.iloc[3,1]

#Calcul du taux d'évolution
tx_evolution_aide_alimentaire = round(((aide_alimentaire_2017 - aide_alimentaire_2013)/aide_alimentaire_2013)*1
#Affichage du résultat
tx_evolution_aide_alimentaire
```

Out[123…

```
-82.15
```

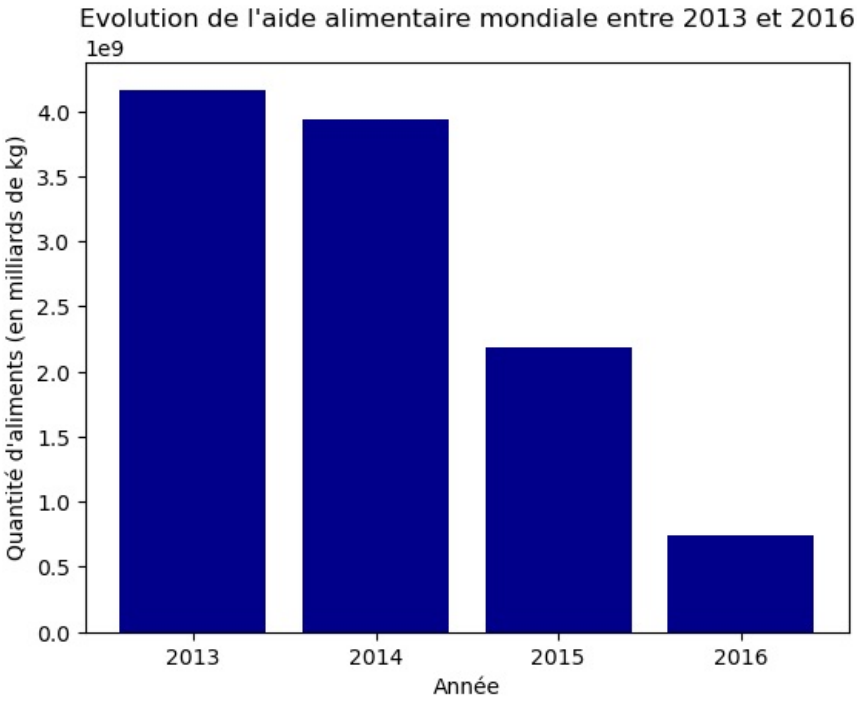
Commentaire: L'aide alimentaire a chuté de -82.15%.

In [125…

```
#Création du diagramme en barre pour représenter l'évolution de l'aide alimentaire
plt.bar(height= total_aide_alimentaire['Quantité'], x= total_aide_alimentaire['Année'], color= 'darkblue')
#Ajout du titre et des noms pour les axes
plt.title("Evolution de l'aide alimentaire mondiale entre 2013 et 2016")
plt.ylabel("Quantité d'aliments (en milliards de kg)")
plt.xlabel('Année')
#Réajustement de l'échelle sur l'axe des années
plt.xticks([2013,2014,2015,2016])
```

Out[125…

```
([<matplotlib.axis.XTick at 0x2b0967db380>,
  <matplotlib.axis.XTick at 0x2b0967dadb0>,
  <matplotlib.axis.XTick at 0x2b096620500>,
  <matplotlib.axis.XTick at 0x2b09843aa80>],
 [Text(2013, 0, '2013'),
  Text(2014, 0, '2014'),
  Text(2015, 0, '2015'),
  Text(2016, 0, '2016')])
```



4.2 -Evolution de la sous-nutrition dans le monde

In [127…

```
#Remplacement des intervalles d'années par des entiers pour le fichier sous-nutrition
sous_nutrition = sous_nutrition.replace({'2012-2014':2013,'2013-2015':2014,'2014-2016':2015,'2015-2017':2016,'2
#Création du dataframe avec le total de la sous nutrition dans le monde par année
total_ss_nutrition= sous_nutrition.loc[sous_nutrition['Année'] <= 2017,:].groupby('Année') [['sous_nutrition']]
total_ss_nutrition= total_ss_nutrition.reset_index()
total_ss_nutrition
```

```
C:\Users\mende\AppData\Local\Temp\ipykernel_2464\3110199914.py:2: FutureWarning: Downcasting behavior in `replac
e` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.in
fer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', T
rue)`
  sous_nutrition = sous_nutrition.replace({'2012-2014':2013,'2013-2015':2014,'2014-2016':2015,'2015-2017':2016,'
2016-2018':2017,'2017-2019':2018})
```

Out[127…

	Année	sous_nutrition
0	2013	530100000.0
1	2014	525600000.0
2	2015	526700000.0
3	2016	530600000.0
4	2017	537700000.0

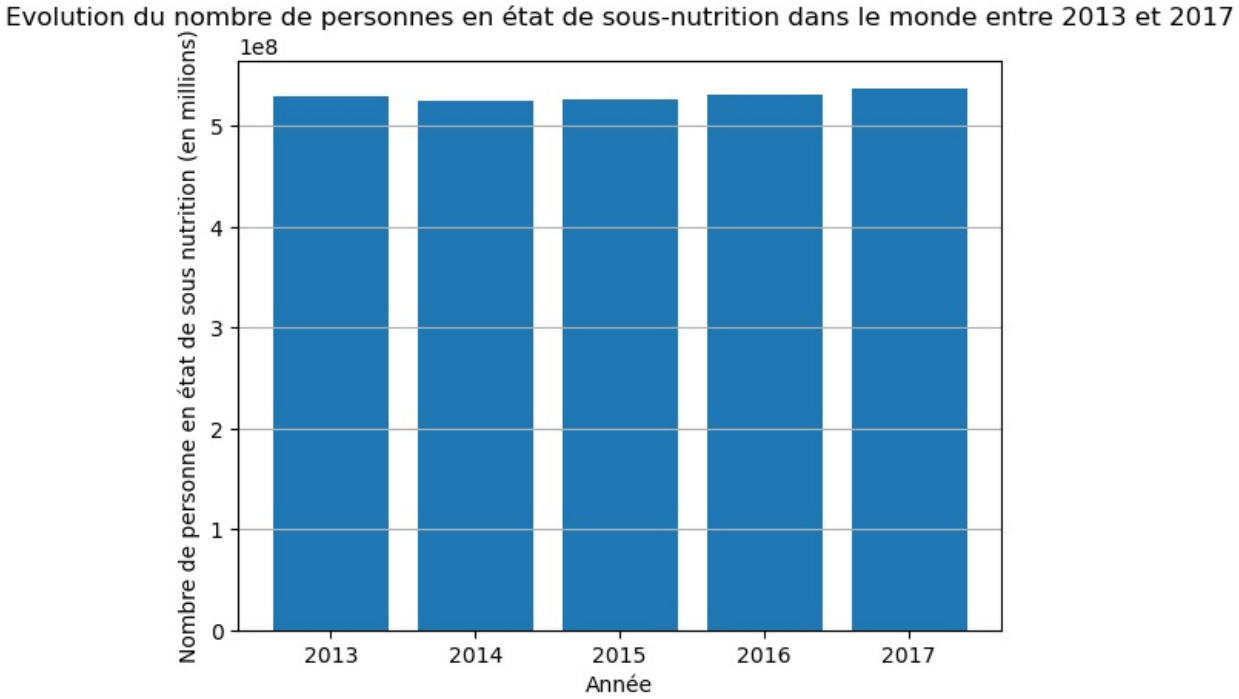
Commentaire: Sur la même période, il y a toujours autant de personnes sous alimentées. Le nombre reste stable.

In [129…
```
#Création de variable pour le nombre de personnes sous nourries sur l'année 2013 et l'année 2017.
ss_nourries_2013 = total_ss_nutrition.iloc[0,1]
ss_nourries_2017 = total_ss_nutrition.iloc[4,1]
#Calcul du taux d'évolution
tx_evolution_ss_nourries = round(((ss_nourries_2017 - ss_nourries_2013)/ss_nourries_2013)*100,2)
#Affichage du résultat
tx_evolution_ss_nourries
```

Out[129… 1.43

Commentaire: Le nombre de personnes sous nourries a augmenté de 1.43% entre 2013 et 2017.

In [131…
```
#Création du graphique pour visualiser l'évolution de personnes sous alimentées
plt.bar(total_ss_nutrition['Année'], total_ss_nutrition['sous_nutrition'])
#Ajout des titres pour le graphique et les axes
plt.title("Evolution du nombre de personnes en état de sous-nutrition dans le monde entre 2013 et 2017")
plt.xlabel("Année")
plt.ylabel("Nombre de personne en état de sous nutrition (en millions)")
#Redimension des axes
plt.xticks([2013,2014,2015,2016,2017])
plt.grid(axis='y')
```



4.3 -Evolution de la proportion de personnes en sous nutrition en Haïti

In [378…
```
#Création d'un dataset avec le nombre total d'habitants par année
population_Haïti = population.loc[(population['Zone'] == 'Haïti') & (population['Année'] <= 2017),:]
population_Haïti
```

Out[378…

	Zone	Année	Population
522	Haïti	2013	10400675.0
523	Haïti	2014	10549009.0
524	Haïti	2015	10695542.0
525	Haïti	2016	10839970.0
526	Haïti	2017	10982366.0

Commentaire: il y a une augmentation faible mais constante de la population

In [135…

```
#Création d'une variable pour modifier la colonne des intervalles d'années dans la dataset sous_nutrition
ss_nutrition_Haïti = sous_nutrition.replace({'2012-2014':2013,'2013-2015':2014,'2014-2016':2015,'2015-2017':201
#Afficher la sous-nutrition uniquement pour Haïti
ss_nutrition_Haïti = sous_nutrition.loc[ss_nutrition_Haïti['Zone']=='Haïti',:]
ss_nutrition_Haïti
```

Out[135…

	Zone	Année	sous_nutrition
468	Haïti	2013	5100000.0
469	Haïti	2014	5100000.0
470	Haïti	2015	5100000.0
471	Haïti	2016	5200000.0
472	Haïti	2017	5300000.0
473	Haïti	2018	5400000.0

Commentaire: la sous-nutrition augmente sensiblement entre 2013 et 2017

In [137…

```
#Affichage du dataset fusionné entre population_Haïti  et ss_nutrition_Haïti
pop_ss_nutrition_Haïti = pd.merge(population_Haïti ,ss_nutrition_Haïti, on= ['Année','Zone'])

#Création de la colonne 'proportion_ss_nourries' pour chaque année
pop_ss_nutrition_Haïti['Proportion_ss_nourries'] = round((pop_ss_nutrition_Haïti['sous_nutrition']/pop_ss_nutri
pop_ss_nutrition_Haïti
```

Out[137…

	Zone	Année	Population	sous_nutrition	Proportion_ss_nourries
0	Haïti	2013	10400675.0	5100000.0	49.04
1	Haïti	2014	10549009.0	5100000.0	48.35
2	Haïti	2015	10695542.0	5100000.0	47.68
3	Haïti	2016	10839970.0	5200000.0	47.97
4	Haïti	2017	10982366.0	5300000.0	48.26

Commentaire: près de la moitié de la population en état de sous-nutrition

4.4 -Nombre de personnes pouvant être nourries en Haïti entre 2013 et 2014

In [140…

```
#Création d'un dataFrame uniquement pour les disponibilité alimentaire en Haïti
dispo_alimentaire_Haïti = dispo_alimentaire.loc[dispo_alimentaire['Zone'] =='Haïti',]
dispo_alimentaire_Haïti.head()
```

Out[140…

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)
5913	Haïti	Abats Comestible	animale	1000000.0	0.0	4.0	1.46	0.12	0.73
5914	Haïti	Agrumes, Autres	vegetale	0.0	0.0	0.0	0.41	0.00	0.00
5915	Haïti	Alcool, non Comestible	vegetale	0.0	1000000.0	0.0	0.00	0.00	0.00
5916	Haïti	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.07	0.01	0.03
5917	Haïti	Ananas	vegetale	0.0	0.0	1.0	0.59	0.00	0.00

In [244…
```
#Création de la colonne (dispo_Kcal_annuelle) sur le dataset 'dispo_alimentaire_Haïti' et Calcul de la disponib
dispo_alimentaire_Haïti = dispo_alimentaire_Haïti.assign(dispo_Kcal_annuelle = dispo_alimentaire_Haïti['Disponi
dispo_alimentaire_Haïti['dispo_Kcal_annuelle']*=365
dispo_alimentaire_Haïti['dispo_Kcal_annuelle']
```

Out[244…
```
5913    762485.0
5914    762485.0
5915    762485.0
5916    762485.0
5917    762485.0
          ...
5995    762485.0
5996    762485.0
5997    762485.0
5998    762485.0
5999    762485.0
Name: dispo_Kcal_annuelle, Length: 87, dtype: float64
```

In [142…
```
#Fusion du dataset dispo_alimentaire Haïti et population_Haïti
df_dispo_population = pd.merge(dispo_alimentaire_Haïti,population_Haïti, on= 'Zone',how= 'left')

#Regrouper la disponibilité Kcal annuelle par année et par population
df_dispo_population.groupby(['Année','Population']).agg({'dispo_Kcal_annuelle': 'first'})
df_dispo_population.head()
```

Out[142…

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)	Di
0	Haïti	Abats Comestible	animale	1000000.0	0.0	4.0	1.46	0.12	0.73	1
1	Haïti	Abats Comestible	animale	1000000.0	0.0	4.0	1.46	0.12	0.73	1
2	Haïti	Abats Comestible	animale	1000000.0	0.0	4.0	1.46	0.12	0.73	1
3	Haïti	Abats Comestible	animale	1000000.0	0.0	4.0	1.46	0.12	0.73	1
4	Haïti	Abats Comestible	animale	1000000.0	0.0	4.0	1.46	0.12	0.73	1

5 rows × 21 columns

In [143…
```
#Création d'un dataframe avec uniquement la disponibilité Kcal par an et par population
Df_dispo_Kcal_an = df_dispo_population.loc[:,['dispo_Kcal_annuelle','Année','Population']].drop_duplicates()

#Création de la colonne 'Nb_nourries_an' et Calcul du nombre théorique de personnes pouvant être nourries entre
Df_dispo_Kcal_an['Nb_nourries_an'] = ((Df_dispo_Kcal_an['dispo_Kcal_annuelle']*Df_dispo_Kcal_an['Population']) /

#Création de la colonne 'Proportion_nourries_an' et calcul de la proportion de personnes pouvant être nourries.
Df_dispo_Kcal_an['Proportion_nourries_an'] = round(((Df_dispo_Kcal_an['Nb_nourries_an'] - Df_dispo_Kcal_an['Popu
Df_dispo_Kcal_an
```

Out[143…

	dispo_Kcal_annuelle	Année	Population	Nb_nourries_an	Proportion_nourries_an
0	762485.0	2013	10400675.0	9.656449e+06	-7.16
1	762485.0	2014	10549009.0	9.794169e+06	-7.16
2	762485.0	2015	10695542.0	9.930217e+06	-7.16
3	762485.0	2016	10839970.0	1.006431e+07	-7.16
4	762485.0	2017	10982366.0	1.019652e+07	-7.16

Commentaire: Entre 2013 et 2017, le nombre de personnes pouvant être nourries est toujours inférieur à la population totale. Donc il semble qu'il n' y ait pas assez de disponibilité alimentaire suffisante pour satisfaire les besoins de la population

4.5 - Impact de l'aide alimentaire sur la sous-nutrition en Haïti

In [146…
```
#Création du dataset sur l'aide alimentaire en Haïti
aide_alimentaire_Haïti = aide_alimentaire.loc[aide_alimentaire['Zone']=='Haïti',:]
aide_alimentaire_Haïti = aide_alimentaire_Haïti.groupby(['Année']) [['Quantité']].sum()

#Création d'un dataframe fusionnant l'aide alimentaire et la sous-nutrition
aide_ss_nutrition = ss_nutrition_Haïti.merge(aide_alimentaire_Haïti, on= 'Année')
aide_ss_nutrition
```

Out[146…

	Zone	Année	sous_nutrition	Quantité
0	Haïti	2013	5100000.0	61214000
1	Haïti	2014	5100000.0	33108000
2	Haïti	2015	5100000.0	9666000
3	Haïti	2016	5200000.0	12462000

Commentaire: on observe que pendant 3 années consécutives, le nombre de personnes sous-alimentées reste stable alors que la quantité d'aide alimentaire baisse fortement. Donc il y a d'autres facteurs à prendre en compte comme la production locale, la situation politique et économique,etc...

4.6 -Utilisation de la disponibilité intérieure en Haïti

In [149…
```
#Calcul du total de la disponibilité intérieure
dispo_intérieure_totale_Haïti = dispo_alimentaire_Haïti['Disponibilité intérieure'].sum()
dispo_intérieure_totale_Haïti
```

Out[149… 6175000000.0

Commentaire: Il y a plus de 6 milliards de kg d'aliments disponibles en Haïti

In [151…
```
#Répartition de l'utilisation de la disponibilité intérieure
proportion_utilisation_Haïti = dispo_alimentaire_Haïti.loc[:,['Aliments pour animaux','Autres Utilisations','No
for elt in proportion_utilisation_Haïti:
    proportion_utilisation_Haïti[elt] = round((proportion_utilisation_Haïti[elt]/dispo_intérieure_totale_Haïti
    print(proportion_utilisation_Haïti[[elt]].sum())
```

```
Aliments pour animaux    7.79
dtype: float64
Autres Utilisations    1.87
dtype: float64
Nourriture    66.43
dtype: float64
Pertes    10.99
dtype: float64
Semences    0.58
dtype: float64
Traitement    12.35
dtype: float64
```

Commentaire: Les ressources alimentaires sont majoritairement utilisées pour nourrir la population. Il y a tout de même près de 11% de pertes d'aliments (problème de stockage?, problème de transport des marchandises ?...)

4.7- Production alimentaire en Haïti

In [154…
```
#Création dataset avec la quantité de production, d'importation /exportations d'aliment
production_Haïti = dispo_alimentaire_Haïti.loc[:,['Produit','Disponibilité alimentaire (Kcal/personne/jour)','P
production_Haïti
```

Out[154…

	Produit	Disponibilité alimentaire (Kcal/personne/jour)	Production	Importations - Quantité	Exportations - Quantité
5913	Abats Comestible	4.0	11000000.0	5000000.0	0.0
5914	Agrumes, Autres	0.0	0.0	4000000.0	0.0
5915	Alcool, non Comestible	0.0	0.0	1000000.0	0.0
5916	Aliments pour enfants	1.0	0.0	1000000.0	0.0
5917	Ananas	1.0	7000000.0	0.0	0.0
...	...	...	...	...	...
5995	Viande de Suides	27.0	33000000.0	11000000.0	0.0
5996	Viande de Volailles	26.0	9000000.0	70000000.0	0.0
5997	Viande, Autre	4.0	11000000.0	0.0	0.0
5998	Vin	0.0	0.0	1000000.0	0.0
5999	Épices, Autres	0.0	0.0	0.0	0.0

87 rows × 5 columns

In [330…
```
#Calcul de la production totale haïtienne
totale_production_Haïti = production_Haïti['Production'].sum()
print(totale_production_Haïti)
```

```
5072000000.0
```

Commentaire: Haïti produit plus de 5 milliards de kg d'aliments

In [266…
```
#Calcul de l'importation totale haïtienne
totale_importation_Haïti = production_Haïti['Importations - Quantité'].sum()
print(totale_importation_Haïti)
```

```
1204000000.0
```

Commentaire: Haïti importe plus de 1 milliard de kg d'aliments

In [155…
```
#Calcul de la proportion de la production et de l'importation par rapport à la disponibilité intérieure du pays
Proportion_acquisition_aliment = dispo_alimentaire_Haïti.loc[:,['Production','Importations - Quantité']]
for elt in Proportion_acquisition_aliment:
    Proportion_acquisition_aliment[elt] = round((Proportion_acquisition_aliment[elt]/dispo_intérieure_totale_Ha
    print(Proportion_acquisition_aliment[[elt]].sum())
```

```
Production    82.14
dtype: float64
Importations - Quantité    19.53
dtype: float64
```

Commentaire: La production représente 82.14% de la disponibilité intérieure contre 19.53% pour les importations.

In [336…
```
#Top 15 des aliments produit dans le pays
production_Haïti = dispo_alimentaire_Haïti.loc[:,['Produit','Disponibilité alimentaire (Kcal/personne/jour)','P
top_production_Haïti = production_Haïti.groupby('Produit') [['Disponibilité alimentaire (Kcal/personne/jour)','
top_production_Haïti
```

Out[336…

Produit	Disponibilité alimentaire (Kcal/personne/jour)	Production	Importations - Quantité	Exportations - Quantité
Sucre, canne	3.0	1.200000e+09	0.0	0.0
Patates douces	118.0	6.000000e+08	4000000.0	0.0
Ignames	81.0	4.250000e+08	0.0	0.0
Manioc	63.0	4.180000e+08	0.0	0.0
Maïs	217.0	3.360000e+08	22000000.0	0.0
Fruits, Autres	25.0	3.210000e+08	6000000.0	11000000.0
Bananes	35.0	2.700000e+08	0.0	0.0
Bananes plantains	30.0	2.670000e+08	0.0	0.0
Légumes, Autres	12.0	1.450000e+08	16000000.0	0.0
Légumineuses Autres	99.0	1.170000e+08	5000000.0	0.0
Haricots	83.0	1.130000e+08	9000000.0	0.0
Riz (Eq Blanchi)	426.0	1.130000e+08	341000000.0	0.0
Sorgho	35.0	1.080000e+08	0.0	0.0
Lait - Excl Beurre	45.0	9.300000e+07	107000000.0	0.0
Boissons Alcooliques	71.0	9.000000e+07	1000000.0	0.0

Commentaire: En Haïti, on produit essentiellement du sucre de canne et des tubercules (tels que la patate douce, le manioc, l'igname ou le maïs). Par exemple, le pays produit 600 millions de kg de patates douces mais il en importe encore 4 millions de kg. On observe qu'elle produit beaucoup d'aliment qui ont une faible valeur nutritive.

En conclusion, il semble que la production locale n'est pas suffisante pour nourrir la population sur place. Le sucre de canne représente la plus grande part de la production locale mais elle n'en exporte pas. La vente du surplus pourrait constituer un levier économique pour le pays.

In [353…
```
#Calcul de la proportion de production d'aliment en fonction de la production totale
proportion_production = top_production_Haïti.loc[:,['Production']]

for elt in proportion_production:
    proportion_production[elt] = round((proportion_production[elt]/totale_production_Haïti)*100,2)
    print(proportion_production[[elt]])
```

Produit	Production
Sucre, canne	23.66
Patates douces	11.83
Ignames	8.38
Manioc	8.24
Maïs	6.62
Fruits, Autres	6.33
Bananes	5.32
Bananes plantains	5.26
Légumes, Autres	2.86
Légumineuses Autres	2.31
Haricots	2.23
Riz (Eq Blanchi)	2.23
Sorgho	2.13
Lait - Excl Beurre	1.83
Boissons Alcooliques	1.77

Commentaire: Parmi le top 15 des aliments les plus produits, la grande majorité concerne le sucre de canne et de la patate douce. Pour le reste, ils sont produits en dessous de 10%. Le pays produit une variété d'aliments mais en faible quantité et l'apport de calorie n'est pas suffisante pour nourrir sa population.

Recommandation: développer l'algriculure locale d'aliment qui ont un fort apport calorique comme le maïs, la patate douce ou le riz

In [357…

```
#Calcul de la proportion d'importation d'aliment en fonction du total des importations
proportion_importation = top_production_Haïti.loc[:,['Importations - Quantité']]

for elt in proportion_importation:
    proportion_importation[elt] = round((proportion_importation[elt]/totale_importation_Haïti)*100,2)
    print(proportion_importation[[elt]])
```

Produit	Importations - Quantité
Sucre, canne	0.00
Patates douces	0.33
Ignames	0.00
Manioc	0.00
Maïs	1.83
Fruits, Autres	0.50
Bananes	0.00
Bananes plantains	0.00
Légumes, Autres	1.33
Légumineuses Autres	0.42
Haricots	0.75
Riz (Eq Blanchi)	28.32
Sorgho	0.00
Lait - Excl Beurre	8.89
Boissons Alcooliques	0.08

Commentaire: Parmi le top 15 des aliments les plus produits, le pays est a une forte dépendance au riz et au lait. Haïti ne semble pas autosuffisant concernant son alimentation