

OPENCLASSROOMS



Parcours Data Analyst




**Food and Agriculture
Organization of the
United Nations**

PROJET 4

**REALISER UNE ETUDE DE SANTE
PUBLIQUE AVEC PYTHON**

Mor Niang



SUMMARY

-
- 1 Introduction
 - 2 Importation et nettoyage des données
 - 3 Requêtes et résultats
 - 4 Conclusion



Poste occupé : Data analyst

FAO : L'Organisation pour l'alimentation et l'agriculture est une organisation spécialisée du système des Nations unies.

Mission : réaliser une études de grande ampleur sur le thème de sous-nutrition dans le monde .

Objectif : donner un panorama de la malnutrition dans le monde.

Source de données :

<http://www.fao.org/faostat/fr/#data>



Outil utilisé : Jupyter Notebook

IMPORTATION ET NETTOYAGE DES DONNÉES

1. Importation des librairies
2. Fichier disponibilité alimentaire
3. Fichier insécurité alimentaire
4. Fichier population
5. Fichier aide alimentaire

1. Importation des librairies dans python

```
##Importation des Librairies Pandas et Matplotlib  
import pandas as pd  
import matplotlib.pyplot as plt
```

Pandas est une librairie python qui permet de manipuler facilement des données a analyser.

Matplotlib est une librairie python qui permet de faire des graphiques.

2. Importation du fichier disponibilité alimentaire

```
dispo_alimentaire = pd.read_csv("dispo_alimentaire.csv")
```

```
dispo_alimentaire.head()
```

	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)
0	Afghanistan	Abats Comestible	animale	NaN	NaN	5.0	1.72	0.20	0.77
1	Afghanistan	Agrumes, Autres	vegetale	NaN	NaN	1.0	1.29	0.01	0.02
2	Afghanistan	Aliments pour enfants	vegetale	NaN	NaN	1.0	0.06	0.01	0.03
3	Afghanistan	Ananas	vegetale	NaN	NaN	0.0	0.00	NaN	NaN
4	Afghanistan	Bananes	vegetale	NaN	NaN	4.0	2.70	0.02	0.05

Convertir les colonnes en kg/personne/an

```
#Convertir les colonnes g/personne/jour en kg/personne/an.
Conversion = ["Disponibilité de matière grasse en quantité (g/personne/jour)",
              "Disponibilité de protéines en quantité (g/personne/jour)"]
for x in Conversion:
    dispo_alimentaire[x] *= 365*0.001

dispo_alimentaire.head()
```

	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)
0	Afghanistan	Abats Comestible	animale	NaN	NaN	5.0	1.72	0.07300	0.28105
1	Afghanistan	Agrumes, Autres	vegetale	NaN	NaN	1.0	1.29	0.00365	0.00730
2	Afghanistan	Aliments pour enfants	vegetale	NaN	NaN	1.0	0.06	0.00365	0.01095

Convertir en kg les 10 variables d'utilisation

```
## Convertir en kg les 10 variables qui précisent les utilisations/moyens d'acquisition de la disponibilité aliment
## 1 tonne = 1000 kg alors 1000 tonnes = 1 000 000 Kg

Conversion_utilisation = ["Aliments pour animaux", "Disponibilité intérieure", "Autres Utilisations",
                          "Exportations - Quantité", "Importations - Quantité", "Nourriture", "Pertes", "Production",
                          "Semences", "Traitement", "Variation de stock"]

for x in Conversion_utilisation:
    dispo_alimentaire[x] *= 1000000

dispo_alimentaire.head()
```

Disponibilité intérieure	Exportations - Quantité	Importations - Quantité	Nourriture	Pertes	Production	Semences	Traitement	Variation de stock
53000000.0	NaN	NaN	53000000.0	NaN	53000000.0	NaN	NaN	NaN
41000000.0	2000000.0	40000000.0	39000000.0	2000000.0	3000000.0	NaN	NaN	NaN
2000000.0	NaN	2000000.0	2000000.0	NaN	NaN	NaN	NaN	NaN

Renommer les colonnes

```
## Renommer les colonnes g/personne/jour en kg/personne/an.
dispo_alimentaire.rename(columns = {
    "Disponibilité de matière grasse en quantité (g/personne/jour)" : "Disponibilité de matière grasse en quantité
    "Disponibilité de protéines en quantité (g/personne/jour)" : "Disponibilité de protéines quantité (kg/personne/
}, inplace=True)

dispo_alimentaire.head()
```

	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 (kg/personne/an)	Disponibilité de protéines quantité (kg/personne/an)
0	Afghanistan	Abats Comestible	animale	NaN	NaN	5.0	1.72	0.07300	0.28105
1	Afghanistan	Agrumes, Autres	vegetale	NaN	NaN	1.0	1.29	0.00365	0.00730
2	Afghanistan	Aliments pour enfants	vegetale	NaN	NaN	1.0	0.06	0.00365	0.01095

Remplacer les valeurs manquantes par 0

```
# Les données manquantes seront remplacées par 0.
```

```
dispo_alimentaire.fillna(0, inplace=True)  
dispo_alimentaire.head()
```

	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 (kg/personne/an)	Disponibilité de protéines quantité (kg/personne/an)
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.07300	0.28105
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.00365	0.00730
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.00365	0.01095
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00000	0.00000
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.00730	0.01825

Afficher le types de données du fichier disponibilité alimentaire

```
#Afficher Le types de données de chaque colonne du fichier disponibilité alimentaire.  
dispo_alimentaire.dtypes
```

```
Zone                object  
Produit             object  
Origine             object  
Aliments pour animaux    float64  
Autres Utilisations    float64  
Disponibilité alimentaire (Kcal/personne/jour)    float64  
Disponibilité alimentaire en quantité (kg/personne/an)    float64  
Disponibilité de matière grasse en quantité (kg/personne/an)    float64  
Disponibilité de protéines quantité (kg/personne/an)    float64  
Disponibilité intérieure    float64  
Exportations - Quantité    float64  
Importations - Quantité    float64  
Nourriture          float64  
Pertes              float64  
Production          float64  
Semences            float64  
Traitement          float64  
Variation de stock  float64  
dtype: object
```

3. Importation du fichier sous-nutrition

```
sous_nutrition = pd.read_csv("sous_nutrition.csv")  
  
sous_nutrition.head()
```

	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

3. Importation du fichier population

```
population = pd.read_csv("population.csv")  
population.head()
```

	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

4. Importation du fichier aide alimentaire

```
aide_alimentaire = pd.read_csv("aide-alimentaire-csv.csv")  
aide_alimentaire.head()
```

	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



Requêtes et résultats

DEMANDES DE MARC

1. La proportion des personnes en état de sous nutrition

Faire la jointure des fichiers population et sous nutrition

```
# Faire la jointure en utilisant pd.merge et on applique une filtre à la colonne année pour avoir  
population_sous_nutrition = pd.merge(population.loc[population["Année"] == 2017],  
                                     sous_nutrition.loc[sous_nutrition["Année"] == 2017],  
                                     on = "Zone")  
  
population_sous_nutrition.head()
```

	Zone	Année_x	Valeur_x	Année_y	Valeur_y
0	Afghanistan	2017	36296113.0	2017	10500000.0
1	Afrique du Sud	2017	57009756.0	2017	3100000.0
2	Albanie	2017	2884169.0	2017	100000.0

#Calculer la population mondiale en utilisant .sum()

```
population_mondiale = population_sous_nutrition["Population"].sum()  
print("La population mondiale est estimée à", population_mondiale,"personnes.")
```

La population mondiale est estimée à 7543798779.0 personnes.

##Calculer le nombre de personne en sous-nutrition en utilisant .sum()

```
personne_en_sous_nutrition = population_sous_nutrition["sous_nutrition"].sum()  
print( "Le nombre de personne en sous-alimentation est de", personne_en_sous_nutrition, "dans le monde.")
```

Le nombre de personne en sous-alimentation est de 535700000.0 dans le monde.

#Calculer la proportion de personne en sous-nutrition par rapport a la population mondiale.

```
print ("La proportion de personne en sous-nutrition est de",  
       round(((personne_en_sous_nutrition)/(population_mondiale))*100,2),"%")
```

La proportion de personne en sous-nutrition est de 7.1 %

2. Le nombre théorique de personne qui pourraient être nourries

```
## Fusionner les fichiers disponibilité alimentaire et population.  
## Utiliser pd.merge pour la fusion et appliquer un filtre à la colonne année pour avoir les données de 2017.  
## Indiquer les colonnes que l'on souhaite afficher après la fusion.  
  
dispo_population = pd.merge((dispo_alimentaire[["Zone", "Disponibilité alimentaire (Kcal/personne/jour)"]]),  
                             (population.loc[population["Année"] == 2017, ["Zone", "Valeur"]]),  
                             on = "Zone")  
dispo_population.head()
```

	Zone	Disponibilité alimentaire (Kcal/personne/jour)	Valeur
0	Afghanistan	5.0	36296113.0
1	Afghanistan	1.0	36296113.0
2	Afghanistan	1.0	36296113.0
3	Afghanistan	0.0	36296113.0
4	Afghanistan	4.0	36296113.0

```
# Créer une variable qui est égale à la somme de la disponibilité alimentaire par zone.
```

```
dispo_aliment_mondiale = dispo_population["disponibilité alimentaire par zone"].sum()  
print("La disponibilité alimentaire mondiale est de ", dispo_aliment_mondiale,  
      "kcal/personne/jour.")
```

```
La disponibilité alimentaire mondiale est de 20918984627331.0 kcal/personne/jour.
```

```
## Diviser la disponibilité alimentaire mondiale par 2500 qui est la calories moyenne journalière
```

```
print("Le nombre theorique de personnes qui pourraient etre nouries est de",  
      (dispo_aliment_mondiale)/2500,"personnes.")
```

```
Le nombre theorique de personnes qui pourraient etre nouries est de 8367593850.9324 personnes.
```

2. Le nombre théorique de personne qui pourraient être nourries avec des produits végétaux

```
## On filtre la colonne origine pour afficher que les produits végétaux.
```

```
dispo_vegetale = dispo_vegetale[dispo_vegetale["Origine"] == "vegetale"]  
dispo_vegetale.head()
```

	Zone	Disponibilité alimentaire (Kcal/personne/jour)	Origine	Valeur
1	Afghanistan	1.0	vegetale	36296113.0
2	Afghanistan	1.0	vegetale	36296113.0
3	Afghanistan	0.0	vegetale	36296113.0
4	Afghanistan	4.0	vegetale	36296113.0
6	Afghanistan	0.0	vegetale	36296113.0


```
## On calcule la disponibilité des produits d'origines végétales mondiale.
```

```
disponibilité_mondiale = dispo_vegetale["dispo produit vegetale"].sum()  
print("Disponibilité mondiale des produits d'origine végétale",disponibilité_mondiale,"Kcal/
```

```
Disponibilité mondiale des produits d'origine végétale 17260764211501.0 Kcal/personne/jours
```

```
print("Les produits d'origine végétaux peuvent nourrir théoriquement",  
      disponibilité_mondiale/2500, "personnes")
```

```
Les produits d'origine végétaux peuvent nourrir théoriquement 6904305684.6004 personnes
```

3. Autres calculs

A . La part destinée à l'alimentation animale

```
## Créer une variable Calculer alimentation_animale qui est le total des aliments  
alimentation_animale = dispo_alimentaire["Aliments pour animaux"].sum()  
print("La somme des aliments pour animaux est de ",alimentation_animale,"kg.")
```

La somme des aliments pour animaux est de 1304245000000.0 kg.

```
## Créer une variable Disponibilité_interieure_globale qui est la somme de disponibilité  
Disponibilité_interieure_globale = dispo_alimentaire["Disponibilité intérieure"].sum()  
print("La disponibilité interieur globale est de ",Disponibilité_interieure_globale,"kg")
```

La disponibilité interieur globale est de 9848994000000.0 kg

```
## Calcule la proportion des aliments pour aliments en faisant le rapport entre alime.  
part_animale = round(((alimentation_animale/Disponibilité_interieure_globale)*100),2)  
print("La proportion des aliments destinée aux animaux est " + str(part_animale)+"%")
```

La proportion des aliments destinée aux animaux est 13.24%

3. Autres calculs

B. La part perdue

```
## La perte totale d'aliment.  
alimentation_perdue = dispo_alimentaire["Pertes"].sum()  
print("L'alimentation totale perdue est de "+ str(alimentation_perdue)+"kg")
```

L'alimentation totale perdue est de 453698000000.0kg

```
## Calculer la proportion de l'alimentation perdue  
part_perdue = round(((alimentation_perdue/Disponibilité_interieure_globale)*100),2)  
print("la proportion de l'alimentation perdue est de " + str(part_perdue)+ "%")
```

la proportion de l'alimentation perdue est de 4.61%

3. Autres calculs

C. La part destinée à l'alimentation humaine

```
## Créer une variable alimentation_humaine qui est la somme des disponibilités  
alimentation_humaine = dispo_alimentaire["Nourriture"].sum()  
print("L'alimentation humaine globale est de ",alimentation_humaine,"kg")
```

L'alimentation humaine globale est de 4876258000000.0 kg

```
part_humaine = round(((alimentation_humaine/Disponibilité_interieure_globale)*100),2)  
print("La proportion de l'alimentation humaine est de "  
      + str(part_humaine)+ "%")
```

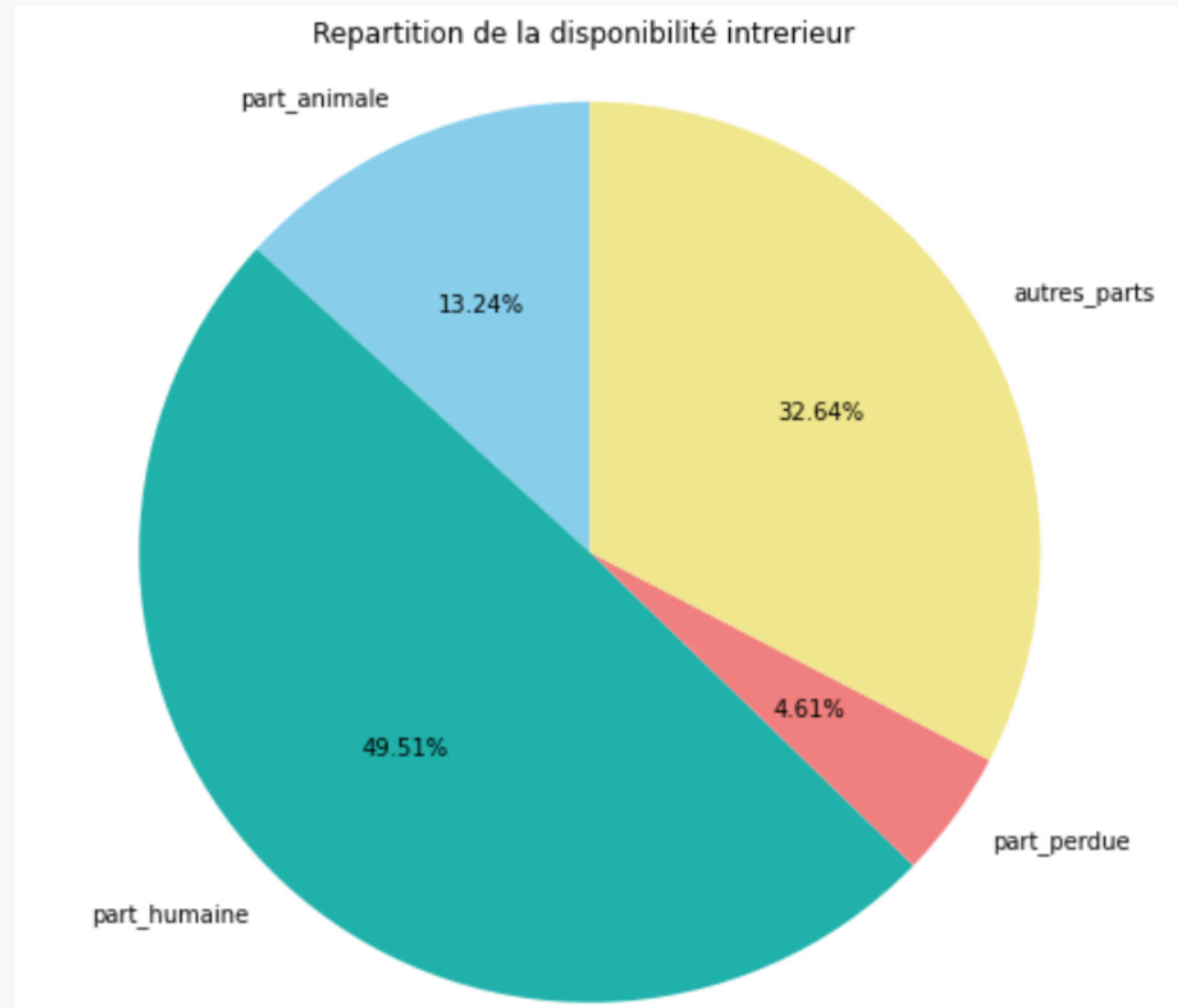
La proportion de l'alimentation humaine est de 49.51%

```
# Calculer la proportion des autres parts.  
autres_parts = 100 - (49.51 + 4.61 + 13.24)  
print("La proportion des autres parts représente " + str(autres_parts) + "%.")
```

La proportion des autres parts représente 32.64%.

3. Autres calculs

d . Représentation graphiques de l'utilisation de la disponibilité interieurs



DEMANDES DE MELANIE

1. Les pays pour lesquels la proportion de personnes sous-alimentées est la plus forte en 2017

```
## Créer un variable df qui affiche la liste des 10 pays dont le taux de sous-nutrition est le plus élevé.
```

```
df10 = population_sous_nutrition[["Zone", "taux de sous nutrition par pays"]].  
sort_values(by = ["taux de sous nutrition par pays"], ascending = False).head(10)  
df10
```

	Zone	taux de sous nutrition par pays
78	Haïti	48.26
157	République populaire démocratique de Corée	47.19
108	Madagascar	41.06
103	Libéria	38.28
100	Lesotho	38.25
183	Tchad	37.96
161	Rwanda	35.06
121	Mozambique	32.81
186	Timor-Leste	32.17
0	Afghanistan	28.93

2. Les pays qui ont bénéficiés le plus d'aide depuis 2013

```
## On calcule la somme des aides alimentaires par pays  
aide_par_pays = aide_alimentaire.groupby("Zone").sum()  
## On affiche par ordre decroissant les aides alimentaires pour avoir les 10 pays qui ont  
decroissant = aide_par_pays.sort_values("Valeur", ascending = False)  
##On supprimer la colonnes "Année"  
decroissant.drop("Année", axis =1 , inplace = True)  
decroissant.head(10)
```

	Valeur
Zone	
République arabe syrienne	1858943
Éthiopie	1381294
Yémen	1206484
Soudan du Sud	695248
Soudan	669784
Kenya	552836
Bangladesh	348188
Somalie	292678

3. Les pays ayant le plus de disponibilité par habitant

```
# Calculer la somme des disponibilités alimentaires (kcal/personne/jour) par pays et afficher par ordre décroissant
dispo_decroissant = dispo_alimentaire[["Zone", "Disponibilité alimentaire (Kcal/personne/jour)"]].  
groupby("Zone").sum().sort_values(by="Disponibilité alimentaire (Kcal/personne/jour)", ascending = False)  
dispo_decroissant.head(10)
```

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

4. Les pays ayant le moins de disponibilité par habitant

```
# Calculer la somme des disponibilités alimentaires (kcal/personne/jour) par pays et afficher par ordre  
dispo_decroissant = dispo_alimentaire[["Zone","Disponibilité alimentaire (Kcal/personne/jour)"]].  
groupby("Zone").sum().sort_values(by="Disponibilité alimentaire (Kcal/personne/jour)",ascending = True)  
dispo_decroissant.head(10)
```

Disponibilité alimentaire (Kcal/personne/jour)	
Zone	
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

DEMANDE DE JULIEN

1. Part des céréales destinés à l'alimentation animale

```
##On crée une liste des produits de types cereales.
```

```
cereale = ["Blé", "Maïs", "Riz (Eq Blanchi)", "Orge", "Seigle", "Avoine", "Millet", "Sorgho", "Céréales, Autres"]
```

```
df = dispo_alimentaire.loc[dispo_alimentaire["Produit"].isin(cereale),["Zone","Produit","Aliments pour animaux",  
    "Nourriture", "Pertes","Autres Utilisations","Traitement", "Semences", "Disponibilité intérieure"]]  
df.head()
```

	Zone	Produit	Aliments pour animaux	Nourriture	Pertes	Autres Utilisations	Traitement	Semences	Disponibilité intérieure
7	Afghanistan	Blé	0.0	4.895000e+09	775000000.0	0.0	0.0	322000000.0	5.992000e+09
12	Afghanistan	Céréales, Autres	0.0	0.000000e+00	0.0	0.0	0.0	0.0	0.000000e+00
32	Afghanistan	Maïs	200000000.0	7.600000e+07	31000000.0	0.0	0.0	5000000.0	3.130000e+08
34	Afghanistan	Millet	0.0	1.200000e+07	1000000.0	0.0	0.0	0.0	1.300000e+07

1. Part des céréales destinés à l'alimentation animale

```
##Créer une variable proportion_animale qui est egale au rapport entre la somme des aliments pour animaux et la s  
proportion_animale = round((((df["Aliments pour animaux"].sum())/ (df["Disponibilité intérieure"].sum()))*100), 2)  
print("La proportion des cereales destiné à l'alimentation animale est de " + str(proportion_animale) + "%")
```

La proportion des cereales destiné à l'alimentation animale est de 36.29%

2. Part des céréales destinés à l'alimentation humaine

```
##Créer une variable proportion_humaine qui est egale au rapport entre la somme des nourriture et la somme de  
proportion_humaine = round((((df["Nourriture"].sum())/ (df["Disponibilité intérieure"].sum()))*100), 2)  
print("La proportion des cereales destiné à l'alimentation humaine est de " + str(proportion_humaine) + "%")
```

La proportion des cereales destiné à l'alimentation humaine est de 42.75%

3. Analyse de l'utilisation du manioc par la Thaïlande aux égards de la proportion de personnes en sous-nutrition

```
#Calculer le ratio des exportation par rapport à la production de manioc.  
df["ratio"] = round(((df["Exportations - Quantité"]/df["Production"])*100),2)  
df
```

	Zone	Produit	Exportations - Quantité	Production	ratio
13809	Thaïlande	Manioc	2.521400e+10	3.022800e+10	83.41

```
#Afficher le taux de sous nutrition de Thaïlande.  
population_sous_nutrition.loc[population_sous_nutrition["Zone"] == "Thaïlande"]
```

	Zone	Année	Population	sous_nutrion	taux de sous nutrition par pays
185	Thaïlande	2017	69209810.0	6200000.0	8.96

3. Analyse de l'utilisation du manioc par la Thaïlande aux égards de la proportion de personnes en sous-nutrition

```
dispo_manioc = pd.merge((dispo_alimentaire.loc[(dispo_alimentaire["Zone"] == "Thaïlande")
& (dispo_alimentaire["Produit"] == "Manioc"), ["Zone", "Produit", "Exportations - Quantité", "Production"]]),
                        (population.loc[population["Année"] == 2017, ["Zone", "Valeur"]]),
                        on = "Zone")
dispo_manioc.head()
```

	Zone	Produit	Exportations - Quantité	Production	Valeur
0	Thaïlande	Manioc	2.521400e+10	3.022800e+10	69209810.0

```
#Le nombre théorique de personnes qui pourraient être nourries avec le produit manioc exportée.
personne_nourrie = dispo_manioc["Exportations - Quantité"]/2500
personne_nourrie
```

0	10085600.0
---	------------

Conclusion



Compétences développées :

1. Utilisation des librairies Pandas et Matplotlib
2. Manipulation des DataFrames
3. Création d'un environnement de développement (Jupyter Notebook).