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

```
#Importation de la librairie Pandas
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

1.2 - Chargement des fichiers Excel

```
#Importation du fichier population.csv
population = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/population.csv')

#Importation du fichier dispo_alimentaire.csv
dispoAlimentaire = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dispo_alimentaire.csv')

#Importation du fichier aide_alimentaire.csv
aideAlimentaire = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/aide_alimentaire.csv')

#Importation du fichier sous_nutrition.csv
sousNutrition = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/sous_nutrition.csv')
```

Nouvelle section

Etape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier population

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

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

```
#Consulter le nombre de colonnes
print('Dans cette table il y a',population.shape[1],'colonnes')
#La nature des données dans chacune des colonnes
print('Colonne | Type '), print(population.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
print('Cette table contient',population.shape[0],'valeurs')
```

```
Dans cette table il y a 3 colonnes
Colonne | Type
Zone       object
Année       int64
Valeur    float64
dtype: object
Cette table contient 1416 valeurs
```

```
#Affichage les 5 premières lignes de la table
print(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
```

```
#Nous allons harmoniser les unités. Pour cela, nous avons décidé de multiplier la population par 1000
population['Valeur'] *= 1000
#Multiplication de la colonne valeur par 1000
print(population)
```

```
              Zone  Année      Valeur
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]
```

```
#changement du nom de la colonne Valeur par Population
population.rename(columns={'Valeur':'Population'}, inplace=True)
```

```
#Affichage les 5 premières lignes de la table pour voir les modifications
print(population.head())
```

```
           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
```

2.2 - Analyse exploratoire du fichier disponibilité alimentaire

```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(dispoAlimentaire.shape[0]))
```

```
Le tableau comporte 15605 observation(s) ou article(s)
```

```
#Consulter le nombre de colonnes
print("Le tableau comporte {} colonne(s)".format(dispoAlimentaire.shape[1]))
```

```
Le tableau comporte 18 colonne(s)
```

```
#Affichage les 5 premières lignes de la table
dispoAlimentaire.head()
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	D al (kg/
0	Afghanistan	Abats Comestible	animale	NaN	NaN	5.0	
1	Afghanistan	Agrumes, Autres	vegetale	NaN	NaN	1.0	
2	Afghanistan	Aliments pour enfants	vegetale	NaN	NaN	1.0	
3	Afghanistan	Ananas	vegetale	NaN	NaN	0.0	
4	Afghanistan	Bananes	vegetale	NaN	NaN	4.0	

Étapes suivantes : Générer du code avec `dispoAlimentaire` Afficher les graphiques recommandés

```
#remplacement des NaN dans le dataset par des 0
dispoAlimentaire.fillna(0, inplace=True)
display(dispoAlimentaire.head())
dispoAlimentaire.info()
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	D al (kg/
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15605 entries, 0 to 15604
Data columns (total 18 columns):
 #   Column                                                Non-Null Count  D
---  ------                                                --------------  -
 0   Zone                                                  15605 non-null  o
 1   Produit                                               15605 non-null  o
 2   Origine                                               15605 non-null  o
 3   Aliments pour animaux                                 15605 non-null  f
 4   Autres Utilisations                                   15605 non-null  f
 5   Disponibilité alimentaire (Kcal/personne/jour)        15605 non-null  f
 6   Disponibilité alimentaire en quantité (kg/personne/an)  15605 non-null  f
 7   Disponibilité de matière grasse en quantité (g/personne/jour)  15605 non-null  f
 8   Disponibilité de protéines en quantité (g/personne/jour)  15605 non-null  f
 9   Disponibilité intérieure                              15605 non-null  f
 10  Exportations - Quantité                               15605 non-null  f
 11  Importations - Quantité                               15605 non-null  f
 12  Nourriture                                            15605 non-null  f
 13  Pertes                                                15605 non-null  f
 14  Production                                            15605 non-null  f
 15  Semences                                              15605 non-null  f
 16  Traitement                                            15605 non-null  f
 17  Variation de stock                                    15605 non-null  f
dtypes: float64(15), object(3)
memory usage: 2.1+ MB
```

```
#multiplication de toutes les lignes contenant des milliers de tonnes en Kg
colonnes_tonne_to_kg = ['Aliments pour animaux','Autres Utilisations','Disponibilité intérieure', 'Exportations - Quantité', 'Importatio
for colonne in colonnes_tonne_to_kg:
  dispoAlimentaire[colonne]*= 1000000


#Affichage les 5 premières lignes de la table
dispoAlimentaire.head()
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	D al (kg/
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	

Étapes suivantes : Générer du code avec `dispoAlimentaire` Afficher les graphiques recommandés

2.3 - Analyse exploratoire du fichier aide alimentaire

```
#Afficher les dimensions du dataset
display(aideAlimentaire.info())
print('la table aide alimentaire contient {} entrées'.format(aideAlimentaire.shape[0]))
```

```
<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
None
la table aide alimentaire contient 1475 entrées
```

```
#Consulter le nombre de colonnes
print('la table aide alimentaire contient {} colonnes'.format(aideAlimentaire.shape[1]))
```

```
la table aide alimentaire contient 4 colonnes
```

```
#Affichage les 5 premières lignes de la table
aideAlimentaire.head(5)
```

	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

Étapes suivantes : Générer du code avec `aideAlimentaire` Afficher les graphiques recommandés

```
#changement du nom de la colonne Pays bénéficiaire par Zone
aideAlimentaire.rename(columns={'Pays bénéficiaire':'Zone'}, inplace = True)

aideAlimentaire.head()
```

	Zone	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

Étapes suivantes : Générer du code avec aideAlimentaire | Afficher les graphiques recommandés

```
#changement du nom de la colonne Valeur par Aide alimentaire en kg
aideAlimentaire.rename(columns={'Valeur':'Aide alimentaire en kg'}, inplace = True)
#Multiplication de la colonne Aide_alimentaire qui contient des tonnes par 1000 pour avoir des kg
aideAlimentaire['Aide alimentaire en kg'] *=1000
```

```
#Affichage les 5 premières lignes de la table
aideAlimentaire.head()
```

	Zone	Année	Produit	Aide alimentaire en kg
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

Étapes suivantes : Générer du code avec aideAlimentaire | Afficher les graphiques recommandés

2.3 - Analyse exploratoire du fichier sous nutrition

```
#Afficher les dimensions du dataset
print('la table sous nutrition comporte {} entrées'.format(sousNutrition.shape[0]))
sousNutrition.info()
```

```
la table sous nutrition comporte 1218 entrées
<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
```

```
#Consulter le nombre de colonnes
print('la table sous nutrition comporte {} colonnes'.format(sousNutrition.shape[1]))
```

```
la table sous nutrition comporte 3 colonnes
```

Double-cliquez (ou appuyez sur Entrée) pour modifier

```
#Afficher les 5 premières lignes de la table
sousNutrition.head(5)
```

	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

Étapes suivantes : Générer du code avec sousNutrition | Afficher les graphiques recommandés

```
#Conversion de la colonne (avec l'argument errors=coerce qui permet de convertir automatiquement les lignes qui ne sont pas des nombres
sousNutrition['Valeur'] = pd.to_numeric(sousNutrition['Valeur'], errors ='coerce')
#Puis remplacement des NaN en 0
sousNutrition.fillna(0, inplace=True)


#changement du nom de la colonne Valeur par sous_nutrition
sousNutrition.rename(columns={'Valeur':'sous_nutrition'}, inplace = True)


#Multiplication de la colonne sous_nutrition par 1000000
sousNutrition['sous_nutrition']*=1000000


#Afficher les 5 premières lignes de la table
sousNutrition.head()
```

	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

Étapes suivantes : Générer du code avec sousNutrition | Afficher les graphiques recommandés

3.1 - Proportion de personnes en sous nutrition

```
# Il faut tout d'abord faire une jointure entre la table population et la table sous nutrition, en ciblant l'année 2017
prop_sous_nutrition = pd.merge(population.loc[population['Année'] == 2017, ['Zone','Population']],
sousNutrition.loc[sousNutrition['Année'] == '2016-2018', ['Zone','sous_nutrition']],
on = 'Zone'                                )


#Affichage du dataset
prop_sous_nutrition.head()
```

	Zone	Population	sous_nutrition
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

Étapes suivantes : Générer du code avec prop_sous_nutrition | Afficher les graphiques recommandés

```
#Calcul et affichage du nombre de personnes en état de sous nutrition
Total_sous_nutrition = prop_sous_nutrition['sous_nutrition'].sum()
print("il y a",Total_sous_nutrition,"de personnes dans le monde en état de sous nutrition")
```

```
il y a 535700000.0 de personnes dans le monde en état de sous nutrition
```

3.2 - Nombre théorique de personne qui pourrait être nourries

```
#Combien mange en moyenne un être humain ? Source => les recommandations d'apport journalier proposées par l'Anses sont les suivantes :

#Pour un adulte de 20 à 40 ans : 2 200 kcal pour une femme, 2 700 kcal pour un homme ;
#Pour un adulte de 40 à 60 ans : 2 000 kcal pour une femme, 2 500 kcal pour un homme.
moyenne_kcal = (2200+2700+2000+2500)/4
print("Un adulte doit manger en moyenne", moyenne_kcal, 'kcal par jour')
```

```
Un adulte doit manger en moyenne 2350.0 kcal par jour
```

```
#On commence par faire une jointure entre le data frame population et Dispo_alimentaire afin d'ajouter dans ce dernier la population
nb_theorique_pop_nourrie = pd.merge(dispoAlimentaire, population.loc[population['Année'] == 2017, ['Zone','Population']], on='Zone')
```

```
#Affichage du nouveau dataframe
nb_theorique_pop_nourrie['dispo_kcal'] = nb_theorique_pop_nourrie['Disponibilité alimentaire (Kcal/personne/jour)'] * nb_theorique_pop_r
nb_theorique_pop_nourrie.head()
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	D al (kg/
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	

Étapes suivantes : Générer du code avec nb_theorique_pop_nourrie Afficher les graphiques recommandés

```
#Calcul du nombre d'humains pouvant être nourris
display(nb_theorique_pop_nourrie.head())
display(nb_theorique_pop_nourrie.tail())

total_dispo_kcal_mondial = nb_theorique_pop_nourrie['dispo_kcal'].sum()
population_2017 = population.loc[population['Année']==2017]
print(population_2017)
population_totale = population_2017['Population'].sum()
print(population_totale, 'personnes dans le monde en 2017')
print("Total mondial des kcal disponibles:", total_dispo_kcal_mondial)
nombre_humains_nourris = round(total_dispo_kcal_mondial/(moyenne_kcal*365))
print("le nombre d'humains pouvant être nourris potentiellement graçe à la disponibilité mondiale est de", nombre_humains_nourris)
ratio =   round(((nombre_humains_nourris / population_totale) * 100),2)
print(f"cela correspond à {ratio}% de la population") #123%
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	(k
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	(kg
15411	Îles Salomon	Viande de Suides	animale	0.0	0.0	45.0	
15412	Îles Salomon	Viande de Volailles	animale	0.0	0.0	11.0	
15413	Îles Salomon	Viande, Autre	animale	0.0	0.0	0.0	
15414	Îles Salomon	Vin	vegetale	0.0	0.0	0.0	
15415	Îles Salomon	Épices, Autres	vegetale	0.0	0.0	4.0	

```
                                  Zone  Année  Population
4                            Afghanistan   2017  36296113.0
10                        Afrique du Sud   2017  57009756.0
16                               Albanie   2017   2884169.0
22                               Algérie   2017  41389189.0
28                             Allemagne   2017  82658409.0
...                                  ...    ...         ...
1390  Venezuela (République bolivarienne du)   2017  29402484.0
1396                            Viet Nam   2017  94600648.0
1402                               Yémen   2017  27834819.0
1408                              Zambie   2017  16853599.0
1414                            Zimbabwe   2017  14236595.0

[236 rows x 3 columns]
7548134111.0 personnes dans le monde en 2017
Total mondial des kcal disponibles: 7635429388975815.0
le nombre d'humains pouvant être nourris potentiellement grace à la disponibilité m
```

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

```
#Transfert des données avec les végétaux dans un nouveau dataframe
nb_theorique_pop_nourrie_vegetaux = nb_theorique_pop_nourrie.loc[nb_theorique_pop_nourrie['Origine']=='vegetale']
nb_theorique_pop_nourrie_vegetaux.head()
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Dis ali (kg/pe
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	
6	Afghanistan	Bière	vegetale	0.0	0.0	0.0	

Étapes suivantes : Générer du code avec nb_theorique_pop_nourrie_vegetaux | Afficher les graphiques recommandés

```
#Calcul du nombre de kcal disponible pour les végétaux
kcal_dispo_vegetaux = nb_theorique_pop_nourrie_vegetaux['dispo_kcal'].sum()
print('le nombre de kcal disponible pour les végétaux est de ',kcal_dispo_vegetaux, 'kcal')
```

```
le nombre de kcal disponible pour les végétaux est de  6300178937197865.0 kcal
```

```
#Calcul du nombre d'humains pouvant être nourris avec les végétaux
nombre_humains_nourris_vegetaux = round(kcal_dispo_vegetaux / (moyenne_kcal*365))
ratio_vegetaux = round((nombre_humains_nourris_vegetaux / population_totale)*100, 2)
print("le nombre d'humains pouvant être nourris avec les végétaux est de", nombre_humains_nourris_vegetaux)
print(f"cela correspond à {ratio_vegetaux} % de la population")
```

```
le nombre d'humains pouvant être nourris avec les végétaux est de 7345006047
cela correspond à 97.31 % de la population
```

3.4 - Utilisation de la disponibilité intérieure

```
#Calcul de la disponibilité totale
disponibilité_totale = dispoAlimentaire['Disponibilité intérieure'].sum()
print(disponibilité_totale) #9billions 848milliards de tonnes
```

```
9848994000000.0
```

```
#création d'une boucle for pour afficher les différentes valeurs en fonction des colonnes aliments pour animaux, pertes, nourritures,
#Production + Importations - Exportation + Variation de stock = Disponibilité intérieure = Nourriture + Pertes + Semences + Aliments pou
parts_utilisation = {}
utilisation = ['Nourriture','Pertes','Semences','Aliments pour animaux', 'Traitement', 'Autres Utilisations']
for x in utilisation:
  part=round((nb_theorique_pop_nourrie[x].sum() / disponibilité_totale)*100, 2)
  parts_utilisation[x]=part
  print(f"Part de {x} dans la disponibilité totale : {part}%")
```

```
Part de Nourriture dans la disponibilité totale : 48.79%
Part de Pertes dans la disponibilité totale : 4.59%
Part de Semences dans la disponibilité totale : 1.56%
Part de Aliments pour animaux dans la disponibilité totale : 13.08%
Part de Traitement dans la disponibilité totale : 22.19%
Part de Autres Utilisations dans la disponibilité totale : 8.72%
```

3.5 - Utilisation des céréales

```
#Création d'une liste avec toutes les variables
produits = dispoAlimentaire['Produit'].unique()
print(produits)
cereales='Céréales, Autres', 'Blé', 'Riz (Eq Blanchi)', 'Orge', 'Maïs', 'Seigle', 'Avoine','Millet', 'Sorgho'
print(cereales)
```

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

```
#Création d'un dataframe avec les informations uniquement pour ces céréales
df_cereales= dispoAlimentaire[dispoAlimentaire['Produit'].isin(cereales)]
display(df_cereales.head())
print(df_cereales['Disponibilité intérieure'].sum())
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	(k
7	Afghanistan	Blé	vegetale	0.0	0.0	1369.0	
12	Afghanistan	Céréales, Autres	vegetale	0.0	0.0	0.0	
32	Afghanistan	Maïs	vegetale	200000000.0	0.0	21.0	
34	Afghanistan	Millet	vegetale	0.0	0.0	3.0	
40	Afghanistan	Orge	vegetale	360000000.0	0.0	26.0	

```
2406999000000.0
```

```
#Affichage de la proportion d'alimentation animale
prop_alimentation_animale= ((df_cereales['Aliments pour animaux'].sum())/ (df_cereales['Disponibilité intérieure'].sum())*100)
print(f"{prop_alimentation_animale}% des céréales sont utilisés pour l'alimentation animale")
```

```
36.291456706047654% des céréales sont utilisés pour l'alimentation animale
```

```
#Affichage de la proportion d'alimentation humaine
prop_alimentation_humaine= round(((df_cereales['Nourriture'].sum())/ (df_cereales['Disponibilité intérieure'].sum())*100), 2)
print(f"{prop_alimentation_humaine}% des céréales sont utilisés pour l'alimentation humaine")
```

```
42.75% des céréales sont utilisés pour l'alimentation humaine
```

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

```
#Création de la colonne proportion par pays
prop_sous_nutrition.head()
prop_sous_nutrition['proportion par pays en %'] = round(((prop_sous_nutrition ['sous_nutrition'] / prop_sous_nutrition['Population']) *10
```

```
#Calcul de la sous nutrition moyenne dans le monde
total_sous_nutrition = prop_sous_nutrition['sous_nutrition'].sum ()
total_population = prop_sous_nutrition['Population'].sum()
average = (total_sous_nutrition) / (total_population)

print(total_sous_nutrition, average)
```

```
535700000.0 0.07101196833235417
```

```
#affichage après trie des 10 meilleurs pays
prop_sous_nutrition = prop_sous_nutrition.sort_values(by='proportion par pays en %', ascending = True)
prop_sous_nutrition.head(100)
```

	Zone	Population	sous_nutrition	proportion par pays en %
101	Lettonie	1951097.0	0.0	0.0
113	Malte	437933.0	0.0	0.0
115	Maurice	1264499.0	0.0	0.0
118	Micronésie (États fédérés de)	111459.0	0.0	0.0
120	Monténégro	627563.0	0.0	0.0
...	...	...	...	...
49	Danemark	5732274.0	0.0	0.0
17	Barbade	286232.0	0.0	0.0
10	Arménie	2944791.0	0.0	0.0
60	États-Unis d'Amérique	325084756.0	0.0	0.0
4	Allemagne	82658409.0	0.0	0.0

Étapes suivantes : Générer du code avec `prop_sous_nutrition` Afficher les graphiques recommandés

```
#affichage après trie des 10 pires pays
prop_sous_nutrition = prop_sous_nutrition.sort_values(by='proportion par pays en %', ascending = False)
prop_sous_nutrition.head(10)
```

	Zone	Population	sous_nutrition	proportion par pays en %
78	Haïti	10982366.0	5300000.0	48.2592
157	République populaire démocratique de Corée	25429825.0	12000000.0	47.1887
108	Madagascar	25570512.0	10500000.0	41.0629
103	Libéria	4702226.0	1800000.0	38.2797
100	Lesotho	2091534.0	800000.0	38.2494
183	Tchad	15016753.0	5700000.0	37.9576
161	Rwanda	11980961.0	4200000.0	35.0556
121	Mozambique	28649018.0	9400000.0	32.8109
186	Timor-Leste	1243258.0	400000.0	32.1735

Étapes suivantes : Générer du code avec prop_sous_nutrition | Afficher les graphiques recommandés

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

```
#calcul du total de l'aide alimentaire par pays
aide_par_pays = aideAlimentaire[['Zone','Aide alimentaire en kg']].groupby('Zone').sum()
display(aide_par_pays.head())
```

Zone	Aide alimentaire en kg
Afghanistan	185452000
Algérie	81114000
Angola	5014000
Bangladesh	348188000
Bhoutan	2666000

```
#affichage après trie des 10 pays qui ont bénéficié le plus de l'aide alimentaire
print(aide_par_pays.sort_values(by='Aide alimentaire en kg', ascending = False).head(10))
```

```
                                Aide alimentaire en kg
Zone
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

```
#Création d'un dataframe avec la zone, l'année et l'aide alimentaire puis groupby sur zone et année
evolutionAide = aideAlimentaire[['Zone','Année','Aide alimentaire en kg']]
evolutionAideParAn = evolutionAide.groupby(['Zone','Année']).sum()
display(evolutionAide.head())
evolutionAideParAn.head()
evolutionAideParAn.info()
```

	Zone	Année	Aide alimentaire en kg
0	Afghanistan	2013	682000
1	Afghanistan	2014	335000
2	Afghanistan	2013	39224000
3	Afghanistan	2014	15160000
4	Afghanistan	2013	40504000

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 228 entries, ('Afghanistan', 2013) to ('Éthiopie', 2015)
Data columns (total 1 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Aide alimentaire en kg  228 non-null    int64
dtypes: int64(1)
memory usage: 3.0+ KB
```

```
#Création d'une liste contenant les 5 pays qui ont le plus bénéficiées de l'aide alimentaire
top5= (aide_par_pays.sort_values(by='Aide alimentaire en kg', ascending = False).head(5))
PaysTop5 = top5.index.values
print(PaysTop5)
```

```
['République arabe syrienne' 'Éthiopie' 'Yémen' 'Soudan du Sud' 'Soudan']
```

```
#On filtre sur le dataframe avec notre liste
Top5parAn= evolutionAideParAn.reset_index()
Top5parAn= Top5parAn[Top5parAn['Zone'].isin(PaysTop5)]
```

```
# Affichage des pays avec l'aide alimentaire par année
Top5parAn.head(50)
```

	Zone	Année	Aide alimentaire en kg
157	République arabe syrienne	2013	563566000
158	République arabe syrienne	2014	651870000
159	République arabe syrienne	2015	524949000
160	République arabe syrienne	2016	118558000
189	Soudan	2013	330230000
190	Soudan	2014	321904000
191	Soudan	2015	17650000
192	Soudan du Sud	2013	196330000
193	Soudan du Sud	2014	450610000
194	Soudan du Sud	2015	48308000
214	Yémen	2013	264764000
215	Yémen	2014	103840000
216	Yémen	2015	372306000
217	Yémen	2016	465574000
225	Éthiopie	2013	591404000
226	Éthiopie	2014	586624000
227	Éthiopie	2015	203266000

Étapes suivantes : Générer du code avec Top5parAn Afficher les graphiques recommandés

```
#Aide alimentaire pour les 10 pays où la proportion de personnes en état de sous-nutrition est la plus forte en 2017
#Création de la liste des pays
paysSousNutritionClass = (prop_sous_nutrition.sort_values(by='proportion par pays en %', ascending = False).head(10))
paysTop10SousNutrition = paysSousNutritionClass['Zone'].values

#On filtre sur le dataframe avec notre liste
aideAlimentairePaysSousNutrition= aide_par_pays[aide_par_pays.index.isin(paysTop10SousNutrition)]

# Ajout de la table sous_nutrition
sousNutritionVsAideAlimentaire = pd.merge(aideAlimentairePaysSousNutrition, prop_sous_nutrition , on='Zone', how='left')
sousNutritionVsAideAlimentaire.head(10)

#Kg d'aide alimentaire par habitant
sousNutritionVsAideAlimentaire['kgAideAlimentaireParHabitant'] = sousNutritionVsAideAlimentaire['Aide alimentaire en kg'] / sousNutrition
```

```
sousNutritionVsAideAlimentaire.head(10)
```

	Zone	Aide alimentaire en kg	Population	sous_nutrition	proportion par pays en %	kgAideAlimentair
0	Afghanistan	185452000	36296113.0	10500000.0	28.9287	
1	Haïti	116450000	10982366.0	5300000.0	48.2592	
2	Lesotho	10624000	2091534.0	800000.0	38.2494	
3	Libéria	19846000	4702226.0	1800000.0	38.2797	
4	Madagascar	96678000	25570512.0	10500000.0	41.0629	
5	Mozambique	58612000	28649018.0	9400000.0	32.8109	
6	Rwanda	11408000	11980961.0	4200000.0	35.0556	
7	République populaire démocratique	187412000	25429825.0	12000000.0	47.1887	

Étapes suivantes : Générer du code avec sousNutritionVsAideAlimentaire | Afficher les graphiques recommandés

3.9 - Pays avec le moins de disponibilité par habitant

```
#Calcul de la disponibilité en kcal par personne par jour par pays
dispo= nb_theorique_pop_nourrie[['Zone','Disponibilité alimentaire (Kcal/personne/jour)','Population']]
dispoParPays= dispo.groupby(['Zone']).sum()
dispoParPays.head(10)
```

Zone	Disponibilité alimentaire (Kcal/personne/jour)	Population
Afghanistan	2087.0	2.177767e+09
Afrique du Sud	3020.0	5.415927e+09
Albanie	3188.0	2.653435e+08
Algérie	3293.0	3.849195e+09
Allemagne	3503.0	7.852549e+09
Angola	2474.0	2.385341e+09
Antigua-et-Barbuda	2416.0	8.683766e+06
Arabie saoudite	3255.0	3.111511e+09
Argentine	3226.0	4.086154e+09

Étapes suivantes : Générer du code avec dispoParPays | Afficher les graphiques recommandés

```
# Total disponibilité alimentaire (Kcal/personne/jour)
total_disponibilite = dispoParPays['Disponibilité alimentaire (Kcal/personne/jour)'].sum()

# Nbre de ligne
nombre_zones = dispoParPays.shape[0]

# Calcul Moyenne dispo
avg_dispo = total_disponibilite / nombre_zones
print(avg_dispo)
```

```
2841.639534883721
```

```
#Affichage des 10 pays qui ont le moins de dispo alimentaire par personne
dispoParPays.sort_values(by='Disponibilité alimentaire (Kcal/personne/jour)', ascending = True).head(10)
```

Zone	Disponibilité alimentaire (Kcal/personne/jour)	Population
République centrafricaine	1879.0	3.906620e+08
Zambie	1924.0	1.567385e+09
Madagascar	2056.0	2.352487e+09

3.10 - Pays avec le plus de disponibilité par habitant

République populaire

```
#Affichage des 10 pays qui ont le plus de dispo alimentaire par personne
dispoParPays.sort_values(by='Disponibilité alimentaire (Kcal/personne/jour)', ascending = False).head(10)
```

Zone	Disponibilité alimentaire (Kcal/personne/jour)	Population
Autriche	3770.0	8.378906e+08
Belgique	3737.0	1.084876e+09
Turquie	3708.0	7.543830e+09
États-Unis d'Amérique	3682.0	3.088305e+10
Israël	3610.0	7.501902e+08
Irlande	3602.0	4.515615e+08
Italie	3578.0	5.824675e+09
Luxembourg	3540.0	5.504763e+07
Égypte	3518.0	9.065604e+09

3.11 - Exemple de la Thaïlande pour le Manioc

```
#création d'un dataframe avec uniquement la Thaïlande
thailande = pd.merge(nb_theorique_pop_nourrie, sousNutrition, on ='Zone')
thailande = thailande[thailande['Zone'] == 'Thaïlande']
thailande = thailande[thailande['Année'] == '2016-2018']
thailande = thailande[thailande['Produit'] == 'Manioc']
thailande.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)	Disponibi proté q (g/personn
81724	Thaïlande	Manioc	vegetale	1.800000e+09	2.081000e+09	40.0	13.0	0.05	

1 rows × 22 columns