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 [226…
```
#Importation de la librairie Pandas
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

In [226…
```
pd.__version__
```

Out[2263]:
```
'2.0.1'
```

1.2 - Chargement des fichiers Excel

In [226…
```
#Importation du fichier population.csv
population = pd.read_csv('population.csv')
```

In [226…
```
#Importation du fichier dispo_alimentaire.csv
dispo_alimentaire = pd.read_csv("dispo_alimentaire.csv")
```

In [226…
```
#Importation du fichier aide_alimentaire.csv
aide_alimentaire = pd.read_csv("aide_alimentaire.csv")
```

In [226…
```
#Importation du fichier sous_nutrition.csv
ss_nutrition = pd.read_csv("sous_nutrition.csv")
```

Etape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier population

In [226…
```
#Afficher les dimensions du dataframe population
print("Le tableau comporte {} observation(s) ou article(s)".format(population.sha
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)
```

In [226…
```
#Consulter le nombre de colonnes
population.shape
```

Out[2269]: (1416, 3)

In [227…
```
#La nature des données dans chacune des colonnes
population.dtypes
```

Out[2270]:
```
Zone       object
Année       int64
Valeur    float64
dtype: object
```

In [227…
```
#Le nombre de valeurs présentes dans chacune des colonnes
population.count()
```

Out[2271]:
```
Zone      1416
Année     1416
Valeur    1416
dtype: int64
```

In [227…
```
#Affichage les 5 premières lignes de la table
population.head(5)
```

Out[2272]:

	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

In [227…
```
# convertion de la population en million
population["Valeur"] = population["Valeur"] * 1000
population["Valeur"]
```

Out[2273]:
```
0       32269589.0
1       33370794.0
2       34413603.0
3       35383032.0
4       36296113.0
           ...
1411    13586707.0
1412    13814629.0
1413    14030331.0
1414    14236595.0
1415    14438802.0
Name: Valeur, Length: 1416, dtype: float64
```

In [227…
```
#changement du nom de la colonne Valeur par Population
population = population.rename(columns={"Valeur":"Population"})
```

In [227…
```
#Affichage les 5 premières lignes de la table pour voir les modifications
population.head()
```

Out[2275]:

	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

In [227…
```
#Afficher les dimensions du dataset
dispo_alimentaire.shape
```

Out[2276]:
```
(15605, 18)
```

In [227…
```
#Consulter le nombre de colonnes
print("Le tableau comporte {} lignes".format(dispo_alimentaire.shape[0]))
print("Le tableau comporte {} colonnes".format(dispo_alimentaire.shape[1]))
```

```
Le tableau comporte 15605 lignes

Le tableau comporte 18 colonnes
```

In [227…
```
#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

Out[2278]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)
0	Afghanistan	Abats Comestible	animale	NaN	NaN	5.0	1.72
1	Afghanistan	Agrumes, Autres	vegetale	NaN	NaN	1.0	1.29
2	Afghanistan	Aliments pour enfants	vegetale	NaN	NaN	1.0	0.06
3	Afghanistan	Ananas	vegetale	NaN	NaN	0.0	0.00
4	Afghanistan	Bananes	vegetale	NaN	NaN	4.0	2.70

In [227…
```
#remplacement des NaN dans le dataset par des 0 (méthode fillna())
dispo_alimentaire = dispo_alimentaire.fillna(0)
```

In [228…
```
dispo_alimentaire.head()
```

Out[2280]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70

In [228…
```
dispo_alimentaire.columns
```

Out[2281]:
```
Index(['Zone', 'Produit', 'Origine', 'Aliments pour animaux',
       'Autres Utilisations', 'Disponibilité alimentaire (Kcal/personne/jou
r)',
       'Disponibilité alimentaire en quantité (kg/personne/an)',
       'Disponibilité de matière grasse en quantité (g/personne/jour)',
       'Disponibilité de protéines en quantité (g/personne/jour)',
       'Disponibilité intérieure', 'Exportations - Quantité',
       'Importations - Quantité', 'Nourriture', 'Pertes', 'Production',
       'Semences', 'Traitement', 'Variation de stock'],
      dtype='object')
```

In [228…
```
dispo_alimentaire.info()
```

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 15605 entries, 0 to 15604

Data columns (total 18 columns):

 #   Column                                                Non-Null Co
unt  Dtype

---  ------                                                -----------
---  -----

 0   Zone                                                  15605 non-n
ull  object

 1   Produit                                               15605 non-n
ull  object

 2   Origine                                               15605 non-n
ull  object

 3   Aliments pour animaux                                 15605 non-n
ull  float64

 4   Autres Utilisations                                   15605 non-n
ull  float64

 5   Disponibilité alimentaire (Kcal/personne/jour)        15605 non-n
ull  float64

 6   Disponibilité alimentaire en quantité (kg/personne/an)  15605 non-n
ull  float64

 7   Disponibilité de matière grasse en quantité (g/personne/jour)  15605 non-n
ull  float64

 8   Disponibilité de protéines en quantité (g/personne/jour)  15605 non-n
ull  float64

 9   Disponibilité intérieure                              15605 non-n
ull  float64

 10  Exportations - Quantité                               15605 non-n
ull  float64

 11  Importations - Quantité                               15605 non-n
ull  float64

 12  Nourriture                                            15605 non-n
ull  float64
```

```
 13  Pertes                                                 15605 non-n
ull  float64

 14  Production                                             15605 non-n
ull  float64

 15  Semences                                               15605 non-n
ull  float64

 16  Traitement                                             15605 non-n
ull  float64

 17  Variation de stock                                     15605 non-n
ull  float64

dtypes: float64(15), object(3)

memory usage: 2.1+ MB
```

In [228…
```
#multiplication de toutes les lignes contenant des milliers de tonnes en Kg
list_columns = ['Aliments pour animaux', 'Autres Utilisations',
            'Disponibilité intérieure',
            'Exportations - Quantité', 'Importations - Quantité', 'Nourriture',
            'Semences', 'Traitement', 'Variation de stock']

dispo_alimentaire[list_columns] = dispo_alimentaire[list_columns] * 1000000
```

In [228…
```
#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

Out[2284]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70

2.3 - Analyse exploratoire du fichier aide alimentaire

In [228…
```
aide_alimentaire.shape
```

Out[2285]:
```
(1475, 4)
```

In [228…
```
#Afficher les dimensions du dataframe aide alimentaire
print("Le tableau comporte {} lignes".format(aide_alimentaire.shape[0]))
print("Le tableau comporte {} colonnes".format(aide_alimentaire.shape[1]))
```

```
Le tableau comporte 1475 lignes
```

```
Le tableau comporte 4 colonnes
```

In [228…
```
#Consulter le nombre de colonnes
print("Le tableau comporte {} colonnes".format(aide_alimentaire.shape[1]))
```

```
Le tableau comporte 4 colonnes
```

In [228…
```
#Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

Out[2288]:

	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 [228…
```
#changement du nom de la colonne Pays bénéficiaire par Zone
aide_alimentaire = aide_alimentaire.rename(columns={"Pays bénéficiaire":"Zone"
aide_alimentaire.head()
```

Out[2289]:

	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

In [229…
```
# renommer la colonne Valeur par Aide_alimentaire_Kg
aide_alimentaire = aide_alimentaire.rename(columns={"Valeur":"Aide_alimentaire
aide_alimentaire.head(1)
```

Out[2290]:

	Zone	Année	Produit	Aide_alimentaire_Kg
0	Afghanistan	2013	Autres non-céréales	682

In [229…
```
#Multiplication de la colonne Aide_alimentaire qui contient des tonnes par 100
aide_alimentaire["Aide_alimentaire_Kg"] = aide_alimentaire["Aide_alimentaire_K
```

In [229…
```
#Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

Out[2292]:

	Zone	Année	Produit	Aide_alimentaire_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

2.3 - Analyse exploratoire du fichier sous nutrition

In [229…
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} lignes".format(ss_nutrition.shape[0]))
print("Le tableau comporte {} colonnes".format(ss_nutrition.shape[1]))
```

```
Le tableau comporte 1218 lignes

Le tableau comporte 3 colonnes
```

In [229…
```
#Consulter le nombre de colonnes
print("Le tableau comporte {} colonnes".format(ss_nutrition.shape[1]))
```

```
Le tableau comporte 3 colonnes
```

In [229…
```
#Afficher les 5 premières lignes de la table
ss_nutrition.head()
```

Out[2295]:

	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

In [229…
```
# renommer la colonne "Valeur" en "sous_nutrition"
ss_nutrition = ss_nutrition.rename(columns={"Valeur":"sous_nutrition"})
ss_nutrition
```

Out[2296]:

	Zone	Année	sous_nutrition
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	NaN
1214	Zimbabwe	2014-2016	NaN
1215	Zimbabwe	2015-2017	NaN
1216	Zimbabwe	2016-2018	NaN
1217	Zimbabwe	2017-2019	NaN

1218 rows × 3 columns

In [229…
```
#Conversion de la colonne sous_nutrition en numérique
#Conversion de la colonne (avec l'argument errors=coerce qui permet de conver
ss_nutrition["sous_nutrition"] = pd.to_numeric(ss_nutrition["sous_nutrition"]
```

In [229…
```
# vérification du format de la colonne sous_nutrition
ss_nutrition["sous_nutrition"].dtypes
```

Out[2298]:
```
dtype('float64')
```

In [229…
```
#Puis remplacement des NaN en 0
ss_nutrition["sous_nutrition"] = ss_nutrition["sous_nutrition"].fillna(0)
ss_nutrition
```

Out[2299]:

	Zone	Année	sous_nutrition
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 × 3 columns

In [230…
```
#Multiplication de la colonne sous_nutrition par 1000000
ss_nutrition["sous_nutrition"] = ss_nutrition["sous_nutrition"] * 1000000
```

In [230…
```
#Afficher les 5 premières lignes de la table
ss_nutrition.head()
```

Out[2301]:

	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

In [230…
```
# Scinder la colonne "Année" en année médiane et créer une nouvelle colonne '
ss_nutrition[['Année début', 'Année fin']] = ss_nutrition['Année'].str.split(
ss_nutrition['Année médiane'] = ss_nutrition[['Année début', 'Année fin']].as
```

In [230…
```
ss_nutrition.head()
```

Out[2303]:

	Zone	Année	sous_nutrition	Année début	Année fin	Année médiane
0	Afghanistan	2012-2014	8600000.0	2012	2014	2013
1	Afghanistan	2013-2015	8800000.0	2013	2015	2014
2	Afghanistan	2014-2016	8900000.0	2014	2016	2015
3	Afghanistan	2015-2017	9700000.0	2015	2017	2016

4	Afghanistan	2016-2018	10500000.0	2016	2018	2017

In [230…
```
# Supprimer les colonnes non nécessaires
ss_nutrition = ss_nutrition.drop(columns=['Année', 'Année début', 'Année fin'
```

In [230…
```
# Afficher le nouveau dataframe
ss_nutrition
```

Out[2305]:

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

1218 rows × 3 columns

In [230…
```
# convertir la colonne "Année médiane" en date
ss_nutrition['Année médiane'] = pd.to_datetime(ss_nutrition['Année médiane'],
```

In [230…
```
ss_nutrition.info()
```

```
<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   sous_nutrition  1218 non-null   float64

 2   Année médiane   1218 non-null   datetime64[ns]

dtypes: datetime64[ns](1), float64(1), object(1)

memory usage: 28.7+ KB
```

3.1 - Proportion de personnes en sous nutrition

In [230…
```
population['Année'] = pd.to_datetime(population['Année'], format='%Y')
```

In [230…
```
population.info()
```

```
<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   datetime64[ns]

 2   Population  1416 non-null   float64

dtypes: datetime64[ns](1), float64(1), object(1)

memory usage: 33.3+ KB
```

In [231…
```
population_2017 = population[ population["Année"].dt.year == 2017 ]
population_2017
```

Out[2310]:

	Zone	Année	Population
4	Afghanistan	2017-01-01	36296113.0
10	Afrique du Sud	2017-01-01	57009756.0
16	Albanie	2017-01-01	2884169.0
22	Algérie	2017-01-01	41389189.0
28	Allemagne	2017-01-01	82658409.0
...	...	...	...
1390	Venezuela (République bolivarienne du)	2017-01-01	29402484.0
1396	Viet Nam	2017-01-01	94600648.0
1402	Yémen	2017-01-01	27834819.0
1408	Zambie	2017-01-01	16853599.0
1414	Zimbabwe	2017-01-01	14236595.0

236 rows × 3 columns

In [231…
```
ss_nutrition_2017_df = ss_nutrition[ ss_nutrition["Année médiane"].dt.year
ss_nutrition_2017_df
```

Out[2311]:

	Zone	sous_nutrition	Année médiane
4	Afghanistan	10500000.0	2017-01-01
10	Afrique du Sud	3100000.0	2017-01-01
16	Albanie	100000.0	2017-01-01
22	Algérie	1300000.0	2017-01-01
28	Allemagne	0.0	2017-01-01
...	...	...	...

1192	Venezuela (République bolivarienne du)	8000000.0	2017-01-01
1198	Viet Nam	6500000.0	2017-01-01
1204	Yémen	0.0	2017-01-01
1210	Zambie	0.0	2017-01-01
1216	Zimbabwe	0.0	2017-01-01

203 rows × 3 columns

In [231…
```
# JOINTURE ENTRE POPULATION ET SOUS-NUTRITION ANNEE 2017
# Il faut tout d'abord faire une jointure entre la table population et la t
ratio_population_en_sous_nutrition_df = population_2017.merge(ss_nutrition_
```

In [231…
```
ratio_population_en_sous_nutrition_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 203 entries, 0 to 202

Data columns (total 5 columns):

 #   Column          Non-Null Count  Dtype

---  ------          --------------  -----

 0   Zone            203 non-null    object

 1   Année           203 non-null    datetime64[ns]

 2   Population      203 non-null    float64

 3   sous_nutrition  203 non-null    float64

 4   Année médiane   203 non-null    datetime64[ns]

dtypes: datetime64[ns](2), float64(2), object(1)

memory usage: 8.1+ KB
```

In [231…
```
# AFFICHAGE DU DATASET AVEC JOINTURE
ratio_population_en_sous_nutrition_df.head()
```

Out[2314]:

	Zone	Année	Population	sous_nutrition	Année médiane
0	Afghanistan	2017-01-01	36296113.0	10500000.0	2017-01-01
1	Afrique du Sud	2017-01-01	57009756.0	3100000.0	2017-01-01
2	Albanie	2017-01-01	2884169.0	100000.0	2017-01-01
3	Algérie	2017-01-01	41389189.0	1300000.0	2017-01-01
4	Allemagne	2017-01-01	82658409.0	0.0	2017-01-01

In [231…
```
# PROPORTION DE SOUS-NUTRITION PAR RAPPORT A LA POPULATION TOTALE
ratio_population_en_sous_nutrition_df["ratio_sous_nutrition_mondiale"] = ((
print(f"En 2017, la sous-nutrition globale représentait {ratio_sous_nutriti
```

```
En 2017, la sous-nutrition globale représentait 7.1 % de la population mond
iale.
```

In [231…
```
# Calcul du pourcentage de sous-nutrition par pays
ratio_population_en_sous_nutrition_df['Pourcentage_Sous_Nutrition'] = ((rat
ratio_population_en_sous_nutrition_df.sort_values(by="Pourcentage_Sous_Nutr
```

Out[2316]:

	Zone	Année	Population	sous_nutrition	Année médiane	ratio_sous_nutrition_mondiale
78	Haïti	2017-01-01	10982366.0	5300000.0	2017-01-01	7.1
157	République populaire démocratique de Corée	2017-01-01	25429825.0	12000000.0	2017-01-01	7.1
108	Madagascar	2017-01-01	25570512.0	10500000.0	2017-01-01	7.1

In [231…
```
#Calcul et affichage du nombre de personnes en état de sous nutrition (anné
nombre_total_humains_sous_nutrition_2017 = ratio_population_en_sous_nutriti
print(f"Nombre total d'êtres humains en sous-nutrition : {nombre_total_huma
```

```
Nombre total d'êtres humains en sous-nutrition : 535700000.0, soit 7.1% de
la population mondiale !
```

3.2 - Nombre théorique de personne qui pourrait être nourries

In [231…
```
#Combien mange en moyenne un être humain ? Source => https://fr.wikipedia.
# la ration alimentaire moyenne nécessaire est de 2 500 kcal/personne/jour

# matières grasse :
# Selon wikipedia : 65g pour une femme et 90g pour un homme

# protéines :
# selon wikipédia et la FAO : 49g de protéines pour les hommes adultes et
```

A/ Jointure

In [231…
```
#On commence par faire une jointure entre le data frame population et Disp
df_2017 = dispo_alimentaire.merge(population, on="Zone", how="inner")
```

In [232…
```
#Affichage du nouveau dataframe
df_2017.sample(3)
```

Out[2320]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Dispo alimenta qu (kg/person
59734	Norvège	Sorgho	vegetale	0.0	0.0	0.0	
90199	Émirats arabes unis	Poissons Marins, Autres	animale	0.0	0.0	3.0	
88717	Zimbabwe	Beurre, Ghee	animale	0.0	0.0	3.0	

B/ Création de la colonne "dispo_kcal" avec calcul des kcal disponibles mondialement

In [232…
```
#Création de la colonne dispo_kcal avec calcul des kcal disponibles mondia
df_2017['dispo_kcal'] = df_2017['Disponibilité alimentaire (Kcal/personne/
df_2017.head()
```

Out[2321]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponil alimentair qua (kg/personn
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	
1	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	
2	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	
3	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	
4	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	

5 rows × 21 columns

C/ Calcul du nombre d'humains pouvant être nourris

In [232…
```
df_2017["Année"] = pd.to_datetime(df_2017['Année'], format='%Y')
```

In [232…
```
df_2017.dtypes
```

Out[2323]:
```
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 (g/personne/jour)
float64
Disponibilité de protéines en quantité (g/personne/jour)
float64
Disponibilité intérieure
float64
Exportations - Quantité
float64
Importations - Quantité
float64
Nourriture
float64
Pertes
float64
Production
float64
Semences
float64
Traitement
float64
```

```
Variation de stock
float64
Année                                                          datetim
e64[ns]
Population
float64
dispo_kcal
float64
dtype: object
```

In [232…
```
# Filtrage conditionnel avec dt.year pour se restreindre à l'année 2017
df_2017 = df_2017[ df_2017["Année"].dt.year == 2017 ]
```

In [232…
```
disponibilite_totale = df_2017["dispo_kcal"].sum().round(2)
disponibilite_totale
```

Out[2325]:
```
20918984627331.0
```

In [232…
```
# nombre moyen de calories consommées par personne par jour
nombre_moyen_calories = 2500
```

In [232…
```
# calcul du nombre d'humains pouvant être nourris
nombre_humains_nourris = (disponibilite_totale / nombre_moyen_calories).ro
```

In [232…
```
print(f"En 2017, on aurait pu nourrir approximativement : {nombre_humains_
```

```
En 2017, on aurait pu nourrir approximativement : 8367593851.0 d'êtres hum
ains !

Selon les estimations de l'ONU, la population mondiale en 2017 était d'env
irons : 7,44 milliards d'individus.
```

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

In [232…
```
#Transfert des données avec les végétaux dans un nouveau dataframe
vegetal_df = df_2017[ df_2017["Origine"] == "vegetale" ]
vegetal_df.head()
```

Out[2329]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Dispo alimenta qu (kg/person
10	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	
16	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	
22	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	
28	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	
40	Afghanistan	Bière	vegetale	0.0	0.0	0.0	

5 rows × 21 columns

In [233…
```
#Calcul du nombre de kcal disponible pour les végétaux
disponibilite_kcal_vegetaux = vegetal_df["dispo_kcal"].sum()
disponibilite_kcal_vegetaux
```

Out[2330]: 17260764211501.0

In [233…
```
# Calcul du nombre d'humains pouvant être nourris avec les végétaux
# Dans l'hypothèse qu'un adulte aurait besoin de 2500 Kcal/jour, on peut
humans_nourished_only_vegetables = (disponibilite_kcal_vegetaux / nombre_
```

In [233…
```
print(f"En 2017, on aurait pu nourrir {humans_nourished_only_vegetables}
```

```
En 2017, on aurait pu nourrir 6904305684.6 individus : soit 6,9 milliard
s d'individus !!

Selon les estimations de l'ONU, la population mondiale en 2017 était d'e
nvirons : 7,44 milliards d'individus.
```

3.4 - Utilisation de la disponibilité intérieure

In [233…
```
# Calcul de la disponibilité intérieure totale
food_availability = df_2017["Disponibilité intérieure"].sum()
food_availability
```

Out[2333]: 9733927000000.0

In [233…
```
#création d'une boucle for pour afficher les différentes valeurs en fonc
colonnes = ["Aliments pour animaux", "Pertes", "Nourriture", "Semences",

# Parcours des colonnes
for c in colonnes:
    total_par_colonne = ((df_2017[c].sum() / food_availability) * 100).r
    print(c)
    print(f"% de '{c}' par rapport à la disponibilité intérieure : {tota
    print("\n")
```

```
Aliments pour animaux

% de 'Aliments pour animaux' par rapport à la disponibilité intérieure
: 13.23 %



Pertes

% de 'Pertes' par rapport à la disponibilité intérieure : 4.65 %



Nourriture
```

```
% de 'Nourriture' par rapport à la disponibilité intérieure : 49.37 %



Semences

% de 'Semences' par rapport à la disponibilité intérieure : 1.58 %



Traitement

% de 'Traitement' par rapport à la disponibilité intérieure : 22.45 %



Autres Utilisations

% de 'Autres Utilisations' par rapport à la disponibilité intérieure :
8.82 %
```

3.5 - Utilisation des céréales

In [233…

```python
# Création d'une liste avec toutes les cereales
liste_cereales = ["Blé", "Riz (Eq Blanchi)", "Orge", "Maïs", "Seigle",

# Création d'un nouveau DataFrame en filtrant les données pour ne gard
muesli_df_2017 = df_2017[df_2017["Produit"].isin(liste_cereales)]

# Affichage des premières lignes du nouveau DataFrame
muesli_df_2017.head()
```

Out[2335]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	(k
46	Afghanistan	Blé	vegetale	0.0	0.0	1369.0	
76	Afghanistan	Céréales, Autres	vegetale	0.0	0.0	0.0	
196	Afghanistan	Maïs	vegetale	200000000.0	0.0	21.0	
208	Afghanistan	Millet	vegetale	0.0	0.0	3.0	
244	Afghanistan	Orge	vegetale	360000000.0	0.0	26.0	

5 rows × 21 columns

In [233…
```
# Calcul de la disponibilité intérieure totale des céréales
muesli_availability = muesli_df_2017["Disponibilité intérieure"].sum()
print(f"Somme globale des céréales disponibles sur terre : {muesli_ava
```

```
Somme globale des céréales disponibles sur terre : 2378371000000.0
```

In [233…
```
#création d'une boucle for pour afficher les différentes valeurs en fo
colonnes = ["Aliments pour animaux", "Pertes", "Nourriture", "Semences

for i in colonnes:
    ratio = ((muesli_df_2017[i].sum() / muesli_availability) * 100).ro
    print(i)
    print(f"Part des céréales pour {i}: {ratio} %")
    print("\n")
```

```
Aliments pour animaux

Part des céréales pour Aliments pour animaux: 36.0 %



Pertes

Part des céréales pour Pertes: 4.0 %



Nourriture

Part des céréales pour Nourriture: 43.0 %



Semences

Part des céréales pour Semences: 3.0 %



Traitement

Part des céréales pour Traitement: 4.0 %



Autres Utilisations

Part des céréales pour Autres Utilisations: 10.0 %
```

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

In [233…
```
#Création de la colonne proportion par pays
ratio_population_en_sous_nutrition_df["Proportion_sous_nutrition_par
```

In [233…
```
ratio_population_en_sous_nutrition_df.head()
```

Out[2339]:

	Zone	Année	Population	sous_nutrition	Année médiane	ratio_sous_nutrition_mor
0	Afghanistan	2017-01-01	36296113.0	10500000.0	2017-01-01	
1	Afrique du Sud	2017-01-01	57009756.0	3100000.0	2017-01-01	
2	Albanie	2017-01-01	2884169.0	100000.0	2017-01-01	
3	Algérie	2017-01-01	41389189.0	1300000.0	2017-01-01	
4	Allemagne	2017-01-01	82658409.0	0.0	2017-01-01	

In [234…
```
# Classement par ordre de sous-nutrition plus importante à la moins i
data_sorted = ratio_population_en_sous_nutrition_df.sort_values(by="P
data_sorted.head()
```

Out[2340]:

	Zone	Année	Population	sous_nutrition	Année médiane	ratio_sous_nutrition_
78	Haïti	2017-01-01	10982366.0	5300000.0	2017-01-01	
157	République populaire démocratique de Corée	2017-01-01	25429825.0	12000000.0	2017-01-01	
108	Madagascar	2017-01-01	25570512.0	10500000.0	2017-01-01	
103	Libéria	2017-01-01	4702226.0	1800000.0	2017-01-01	
100	Lesotho	2017-01-01	2091534.0	800000.0	2017-01-01	

In [234…
```
# Les 10 pays les plus touchés par la ss-nutrition
data_sorted.iloc[0:10, :]
```

Out[2341]:

	Zone	Année	Population	sous_nutrition	Année médiane	ratio_sous_nutrition_
78	Haïti	2017-01-01	10982366.0	5300000.0	2017-01-01	
157	République populaire démocratique de Corée	2017-01-01	25429825.0	12000000.0	2017-01-01	

108	Madagascar	2017-01-01	25570512.0	10500000.0	2017-01-01
103	Libéria	2017-01-01	4702226.0	1800000.0	2017-01-01
100	Lesotho	2017-01-01	2091534.0	800000.0	2017-01-01
183	Tchad	2017-01-01	15016753.0	5700000.0	2017-01-01
161	Rwanda	2017-01-01	11980961.0	4200000.0	2017-01-01
121	Mozambique	2017-01-01	28649018.0	9400000.0	2017-01-01
186	Timor-Leste	2017-01-01	1243258.0	400000.0	2017-01-01
0	Afghanistan	2017-01-01	36296113.0	10500000.0	2017-01-01

In [234…
```
ss_nutrition_10_pays_plus_atteints = data_sorted["Zone"].tolist()[0:1
print(f"Les 10 pays les plus touchés par la sous-nutrition en 2017, s
```

```
Les 10 pays les plus touchés par la sous-nutrition en 2017, sont :
['Haïti', 'République populaire démocratique de Corée', 'Madagascar',
'Libéria', 'Lesotho', 'Tchad', 'Rwanda', 'Mozambique', 'Timor-Leste',
'Afghanistan']
```

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

In [234…
```
#calcul du total de l'aide alimentaire par pays
total_aide_alimentaire_par_pays = aide_alimentaire.groupby("Zone").a
```

In [234…
```
#affichage après trie des 10 pays qui ont bénéficié le plus de l'aid
list_of_10 = total_aide_alimentaire_par_pays.sort_values(by='Aide_al
list_of_10
```

Out[2344]:

	Zone	Aide_alimentaire_Kg
0	République arabe syrienne	1858943000
1	Éthiopie	1381294000
2	Yémen	1206484000
3	Soudan du Sud	695248000
4	Soudan	669784000
5	Kenya	552836000
6	Bangladesh	348188000
7	Somalie	292678000
8	République démocratique du Congo	288502000
9	Niger	276344000

In [234…
```
top_5_countries = list_of_10["Zone"].tolist()[:5]
print(top_5_countries)
```

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

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

In [234…
```
# suppression de la colonne "Produit"
aide_alimentaire = aide_alimentaire.drop(columns="Produit")
```

In [234…
```
aide_alimentaire.columns
```

Out[2347]:
```
Index(['Zone', 'Année', 'Aide_alimentaire_Kg'], dtype='object')
```

In [234…
```
aide_alimentaire
```

Out[2348]:

	Zone	Année	Aide_alimentaire_Kg
0	Afghanistan	2013	682000
1	Afghanistan	2014	335000
2	Afghanistan	2013	39224000
3	Afghanistan	2014	15160000
4	Afghanistan	2013	40504000
...	...	...	...
1470	Zimbabwe	2015	96000
1471	Zimbabwe	2013	5022000
1472	Zimbabwe	2014	2310000
1473	Zimbabwe	2015	306000
1474	Zimbabwe	2013	64000

1475 rows × 3 columns

In [234…
```
#Création d'un dataframe avec la zone, l'année et l'aide alimentai
evolution = aide_alimentaire.groupby( ["Zone", "Année"] ).sum().re
```

In [235…
```
evolution.columns
```

Out[2350]:
```
Index(['Zone', 'Année', 'Aide_alimentaire_Kg'], dtype='object')
```

In [235…
```
evolution
```

Out[2351]:

	Zone	Année	Aide_alimentaire_Kg
0	Afghanistan	2013	128238000
1	Afghanistan	2014	57214000
2	Algérie	2013	35234000

3	Algérie	2014	18980000
4	Algérie	2015	17424000
...	...	...	...
223	Égypte	2013	1122000
224	Équateur	2013	1362000
225	Éthiopie	2013	591404000
226	Éthiopie	2014	586624000
227	Éthiopie	2015	203266000

228 rows × 3 columns

In [235…
```
#Création d'une liste contenant les 5 pays qui ont le plus bénéfic
top_5_countries
```

Out[2352]:
```
['République arabe syrienne', 'Éthiopie', 'Yémen', 'Soudan du Su
d', 'Soudan']
```

In [235…
```
# Liste des pays à filtrer
pays_filtre = top_5_countries

# Filtrer le DataFrame original en fonction des pays, méthode de f
filtered_df = evolution[evolution['Zone'].isin(pays_filtre)]

# Afficher le DataFrame filtré
print(filtered_df)
```

```
                          Zone  Année  Aide_alimentaire_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
```

In [235…
```
# Mise en forme : Pivoter le DataFrame pour avoir les années comme
pivot_df = filtered_df.pivot(index='Zone', columns='Année', values
pivot_df
```

Out[2354]:

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

In [235…
```
# Calculer le taux d'évolution entre 2013 et 2015 (Pays concernés
pivot_df['Taux_evolution_2013_2015'] = (((pivot_df[2015] - pivot_d
```

In [235…
```
# Afficher le DataFrame résultant
pivot_df
```

Out[2356]:

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

In [235…
```
# Calculer le taux d'évolution entre 2013 et 2016 (pays concernés
pivot_df['Taux_evolution_2013_2016'] = (((pivot_df[2016] - pivot_d
```

In [235…
```
# Afficher le DataFrame résultant
pivot_df
```

Out[2358]:

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

In [235…
```
# Evolution de ces 5 pays à travers les aides alimentaires :
```

```
pivot_df
```

Out[2359]:

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

En 2016, la Syrie bénéficie de moins d'aide alimentaire : -72% d'aides en 1 an.

Entre 2013 et 2016, le Yemen a reçu +75% d'aides alimentaires. Soit une augmentation de +35% depuis 2015.

In [236…

```
# Affichage des pays avec l'aide alimentaire par année
food_aid_by_year = evolution.pivot(index='Zone', columns='Année', 
food_aid_by_year
```

Out[2360]:

Année Zone	2013	2014	2015	2016
Afghanistan	128238000.0	57214000.0	NaN	NaN
Algérie	35234000.0	18980000.0	17424000.0	9476000.0
Angola	5000000.0	14000.0	NaN	NaN
Bangladesh	131018000.0	194628000.0	22542000.0	NaN
Bhoutan	1724000.0	146000.0	578000.0	218000.0
...	...	...	...	...
Zambie	328000.0	2698000.0	NaN	NaN
Zimbabwe	21252000.0	26600000.0	14718000.0	NaN
Égypte	1122000.0	NaN	NaN	NaN
Équateur	1362000.0	NaN	NaN	NaN
Éthiopie	591404000.0	586624000.0	203266000.0	NaN

76 rows × 4 columns

3.9 - Pays avec le moins de disponibilité par habitant

In [236…

```
#Calcul de la disponibilité en kcal par personne par jour par pay
calcul_dispo = dispo_alimentaire.groupby('Zone').agg({'Disponibil
calcul_dispo
```

Out[2361]:

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

174 rows × 2 columns

In [236…

```
#Affichage des 10 pays qui ont le moins de dispo alimentaire par
moins_dispo_alimentaire_10 = calcul_dispo.sort_values(by='Disponi
moins_dispo_alimentaire_10
```

Out[2362]:

	index	Zone	Disponibilité alimentaire (Kcal/personne/jour)
0	128	République centrafricaine	1879.0
1	166	Zambie	1924.0
2	91	Madagascar	2056.0
3	0	Afghanistan	2087.0
4	65	Haïti	2089.0
5	133	République populaire démocratique de Corée	2093.0
6	151	Tchad	2109.0
7	167	Zimbabwe	2113.0
8	114	Ouganda	2126.0
9	154	Timor-Leste	2129.0

In [236…

```
liste_10 = moins_dispo_alimentaire_10["Zone"].tolist()
print(f"Les 10 pays ayant la plus faible disponibilité en Kcal pa
```

```
Les 10 pays ayant la plus faible disponibilité en Kcal par jour e
t par personne sont : ['République centrafricaine', 'Zambie', 'Ma
dagascar', 'Afghanistan', 'Haïti', 'République populaire démocrat
ique de Corée', 'Tchad', 'Zimbabwe', 'Ouganda', 'Timor-Leste']
```

3.10 - Pays avec le plus de disponibilité par habitant

In [236…
```
#Affichage des 10 pays qui ont le plus de dispo alimentaire par
le_plus_dispo_alimentaire_10 = calcul_dispo.sort_values(by='Disp
le_plus_dispo_alimentaire_10
```

Out[2364]:

	index	Zone	Disponibilité alimentaire (Kcal/personne/jour)
0	11	Autriche	3770.0
1	16	Belgique	3737.0
2	159	Turquie	3708.0
3	171	États-Unis d'Amérique	3682.0
4	74	Israël	3610.0
5	72	Irlande	3602.0
6	75	Italie	3578.0
7	89	Luxembourg	3540.0
8	168	Égypte	3518.0
9	4	Allemagne	3503.0

In [236…
```
liste_10_plus = le_plus_dispo_alimentaire_10["Zone"].tolist()
print(f"Les 10 pays ayant la plus forte disponibilité en Kcal pa
```

```
Les 10 pays ayant la plus forte disponibilité en Kcal par jour
et par personne sont : ['Autriche', 'Belgique', 'Turquie', "Éta
ts-Unis d'Amérique", 'Israël', 'Irlande', 'Italie', 'Luxembour
g', 'Égypte', 'Allemagne']
```

3.11 - Exemple de la Thaïlande pour le Manioc

In [236…
```
#création d'un dataframe avec uniquement la Thaïlande

# Filtrer sur Thaïlande
dispo_alimentaire_thailande = dispo_alimentaire[dispo_alimenta
ss_nutrition_thailande = ss_nutrition[ss_nutrition['Zone'] ==
population_thaï = population[population["Zone"] == "Thaïlande"
```

In [236…
```
# Fusionner les DataFrames filtrés en utilisant la colonne "Zon
merged_df_thailande = pd.merge(ss_nutrition_thailande, dispo_al
merged_df_thaïlande = pd.merge(merged_df_thailande, population_
```

In [236…
```
# Affichage dataframe avec uniquement Thaïlande
merged_df_thaïlande.head(3)
```

Out[2368]:

	Zone_x	sous_nutrition	Année médiane	Produit	Origine	Aliments pour animaux	A Utilisa
0	Thaïlande	6200000.0	2013-01-01	Abats Comestible	animale	0.0	
1	Thaïlande	6200000.0	2013-01-01	Agrumes, Autres	vegetale	0.0	
2	Thaïlande	6200000.0	2013-01-01	Alcool, non Comestible	vegetale	0.0	35800(

3 rows × 23 columns

In [236…
```
merged_df_thaïlande = merged_df_thaïlande.drop(columns=["Zone_y
```

In [237…
```
merged_df_thaïlande = merged_df_thaïlande.rename(columns={"Zone
```

In [237…
```
dispo_kcal_thailande_2017 = merged_df_thaïlande[merged_df_thaï
dispo_kcal_thailande_2017 = dispo_kcal_thailande_2017['Disponib
dispo_kcal_thailande_2017
```

Out[2371]: 2785.0

Calculer et afficher la proportion de sous-nutrition en Thaïlande entre 2013 et 2018

In [237…
```
# Créer un dictionnaire pour stocker les résultats
results = {}

# boucler dans chacune des différentes années
for year in merged_df_thaïlande['Année'].unique():
    # Filtrer les données pour l'année spécifique
    year_data = merged_df_thaïlande[merged_df_thaïlande['Année

    # Calculer la somme totale de sous-nutrition et de populati
    total_sous_nutrition = year_data['sous_nutrition'].sum()
    total_population = year_data['Population'].sum()

    # Calculer la proportion de sous-nutrition en pourcentage p
    proportion = (total_sous_nutrition / total_population) * 10

    # Stocker le résultat dans le dictionnaire
    results[year] = proportion

# Afficher les résultats dans notre dictionnaire
for year, proportion in results.items():
    print(f"Proportion de sous-nutrition en Thaïlande en {year}
```

```
Proportion de sous-nutrition en Thaïlande en 2013-01-01 00:00:
00 : 9.10 %

Proportion de sous-nutrition en Thaïlande en 2014-01-01 00:00:
00 : 8.77 %

Proportion de sous-nutrition en Thaïlande en 2015-01-01 00:00:
00 : 8.59 %

Proportion de sous-nutrition en Thaïlande en 2016-01-01 00:00:
00 : 8.70 %

Proportion de sous-nutrition en Thaïlande en 2017-01-01 00:00:
00 : 8.96 %

Proportion de sous-nutrition en Thaïlande en 2018-01-01 00:00:
00 : 9.36 %
```

In [237…
```
# Autre méthode :
grouping = merged_df_thaïlande.groupby('Année')[ ['Population',
grouping
```

Out[2373]:

Année	Population	sous_nutrition

2013-01-01	6.473729e+09	589000000.0
2014-01-01	6.501681e+09	570000000.0
2015-01-01	6.527879e+09	560500000.0
2016-01-01	6.552274e+09	570000000.0
2017-01-01	6.574932e+09	589000000.0
2018-01-01	6.595703e+09	617500000.0

Taux de sous-nutrition en Thaïlande de 2013 à 2018

In [237…
```
# Afficher le % de sous-nutrition pour la Thaïlande de 2013 à 2
grouping['%_ss_nutrition'] = ((grouping['sous_nutrition'] / gro
grouping.reset_index()
grouping["%_ss_nutrition"]
```

Out[2374]:
```
Année
2013-01-01    9.10
2014-01-01    8.77
2015-01-01    8.59
2016-01-01    8.70
2017-01-01    8.96
2018-01-01    9.36
Name: %_ss_nutrition, dtype: float64
```

In [237…
```
# Filtrer sur le Manioc & l'année 2017
data_thailande_2017 = merged_df_thaïlande[
    (merged_df_thaïlande['Année'] == '2017-01-01') & (merged_df
]

# Afficher le DataFrame filtré
data_thailande_2017
```

Out[2375]:

	Country	sous_nutrition	Année	Produit	Origine	Aliments pour animaux	Uti
430	Thaïlande	6200000.0	2017-01-01	Manioc	vegetale	1.800000e+09	2.081

1 rows × 21 columns

Disponibilité alimentaire d'un Thaïlandais en Kcal/jour

In [237…
```
calcul_dispo[calcul_dispo['Zone'] == "Thaïlande"]
```

Out[2376]:

	Zone	Disponibilité alimentaire (Kcal/personne/jour)
153	Thaïlande	2785.0

Calcul de la proportion de manioc exportée en fonction de la production totale

In [237…
```
total_exportation_manioc = data_thailande_2017['Exportations -
```

In [237…
```
disponibilite_total_manioc = data_thailande_2017['Production']
disponibilite_total_manioc
```

Out[2378]:
```
30228000000.0
```

In [237…

```
#création d'une boucle for pour afficher les différentes valeu
colonnes = ["Exportations - Quantité", "Aliments pour animaux",

for z in colonnes:
    ratio = ((data_thailande_2017[z].sum() / disponibilite_tota
    print(f"{z} en 2017")
    print(f"Part du Manioc pour '{z}': {ratio} %")
    print("\n")
```

```
Exportations - Quantité en 2017

Part du Manioc pour 'Exportations - Quantité': 83.0 %



Aliments pour animaux en 2017

Part du Manioc pour 'Aliments pour animaux': 6.0 %



Pertes en 2017

Part du Manioc pour 'Pertes': 5.0 %



Nourriture en 2017

Part du Manioc pour 'Nourriture': 3.0 %



Semences en 2017

Part du Manioc pour 'Semences': 0.0 %



Traitement en 2017

Part du Manioc pour 'Traitement': 0.0 %



Autres Utilisations en 2017

Part du Manioc pour 'Autres Utilisations': 7.0 %
```

In [239…
```
# On calcule la proportion exportée en fonction de la proportio
data_thailande_2017['Proportion_Export'] = ((total_exportation_
data_thailande_2017
```

```
C:\Users\nbous\AppData\Local\Temp\ipykernel_18392\3470079024.p
y:2: SettingWithCopyWarning:



A value is trying to be set on a copy of a slice from a DataFr
ame.

Try using .loc[row_indexer,col_indexer] = value instead



See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view
-versus-a-copy
```

Out[2394]:

	Country	sous_nutrition	Année	Produit	Origine	Aliments pour animaux	Uti
430	Thaïlande	6200000.0	2017-01-01	Manioc	vegetale	1.800000e+09	2.08

1 rows × 22 columns

En 2017, la Thaïlande a exporté 83 % de sa production totale de manioc ! En comparant le taux de sous-nutrition qui était d'approximativement de 9%, et la disponibilité interieure de manioc qui était de 3%; on peut s'interroger sur la manière de comment est gérée la répartition alimentaire du pays.

En 2017, la Thaïlande avait 6,2 millions de personnes en état de sous-nutrition et paradoxalement sa disponiblité intérieur en Manioc était de 6,26 milliers de tonnes de Manioc.

In [238…
```
df_thaï = merged_df_thaïlande[merged_df_thaïlande['Année'] ==
df_thaï
```

Out[2389]:

	Country	sous_nutrition	Année	Produit	Origine	Aliments pour animaux	Utilis
380	Thaïlande	6200000.0	2017-01-01	Abats Comestible	animale	0.0	
381	Thaïlande	6200000.0	2017-01-01	Agrumes, Autres	vegetale	0.0	
382	Thaïlande	6200000.0	2017-01-01	Alcool, non Comestible	vegetale	0.0	35800
383	Thaïlande	6200000.0	2017-01-01	Aliments pour enfants	vegetale	0.0	

384	Thaïlande	6200000.0	2017-01-01	Ananas	vegetale	0.0
...	...	...	...	...	...	...
470	Thaïlande	6200000.0	2017-01-01	Viande de Suides	animale	0.0
471	Thaïlande	6200000.0	2017-01-01	Viande de Volailles	animale	0.0
472	Thaïlande	6200000.0	2017-01-01	Viande, Autre	animale	0.0
473	Thaïlande	6200000.0	2017-01-01	Vin	vegetale	0.0
474	Thaïlande	6200000.0	2017-01-01	Épices, Autres	vegetale	0.0

95 rows × 21 columns

In [239…
```
balance = data_thailande_2017.groupby('Produit').agg({'Proporti
```

In [239…
```
balance = data_thailande_2017.groupby('Produit').agg({'Proporti
balance.sort_values(by='Proportion_Export', ascending=False).il
```

Out[2396]:

Produit	Proportion_Export
Manioc	83.0

In [239…
```
balance.sort_values(by='Proportion_Export', ascending=True).ilo
```

Out[2391]:

Produit	Proportion_Export
Viande, Autre	-104.0
Dattes	0.0
Girofles	0.0
Graines Colza/Moutarde	0.0
Graines de coton	0.0

La Thaïlande exporte plus de viande qu'elle n'en produit.

Les produits les plus exportés sont le/les : café, crustacés, céréales, manioc et miel.

In [239…
```
# calcul de la balance commerciale de la Thaïlande par produit
# balance commerciale = exportations - importations : Si les im
df_thaï["balance_commerciale"] = df_thaï["Exportations - Quanti

# Grouper par produit et calculer la balance commerciale moyenn
balance_commerciale_par_produit = df_thaï.groupby('Produit')['b
balance_commerciale_par_produit.to_frame().sort_values(by='bala
```

```
C:\Users\nbous\AppData\Local\Temp\ipykernel_18392\72728045.py:
3: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFr
ame.

Try using .loc[row_indexer,col_indexer] = value instead



See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view
-versus-a-copy
```

Out[2397]:

	Produit	balance_commerciale
0	Blé	-1.881000e+09
1	Soja	-1.670000e+09
2	Lait - Excl Beurre	-1.041000e+09
3	Orge	-3.350000e+08
4	Pommes de Terre	-2.360000e+08
5	Pommes	-1.130000e+08
6	Raisin	-1.010000e+08
7	Oranges, Mandarines	-9.700000e+07
8	Poissons Pelagiques	-7.000000e+07
9	Tomates	-6.700000e+07
10	Arachides Decortiquees	-6.400000e+07
11	Oignons	-5.800000e+07
12	Piments	-5.100000e+07
13	Feve de Cacao	-4.200000e+07
14	Edulcorants Autres	-3.000000e+07
15	Légumineuses Autres	-2.800000e+07
16	Abats Comestible	-2.800000e+07
17	Boissons Alcooliques	-2.500000e+07
18	Huile de Tournesol	-1.900000e+07
19	Patates douces	-1.500000e+07
20	Cephalopodes	-1.400000e+07
21	Aliments pour enfants	-1.200000e+07
22	Beurre, Ghee	-1.100000e+07
23	Pois	-1.000000e+07
24	Vin	-8.000000e+06
25	Sésame	-6.000000e+06
26	Plantes Oleiferes, Autre	-6.000000e+06
27	Huile d'Olive	-5.000000e+06
28	Graines de tournesol	-5.000000e+06
29	Huil Plantes Oleif Autr	-5.000000e+06

30	Millet	-4.000000e+06
31	Huiles de Poissons	-4.000000e+06
32	Plantes Aquatiques	-3.000000e+06
33	Poivre	-3.000000e+06
34	Graines Colza/Moutarde	-2.000000e+06
35	Huile de Coco	-1.000000e+06
36	Huile de Colza&Moutarde	-1.000000e+06
37	Boissons Fermentés	-1.000000e+06
38	Viande d'Ovins/Caprins	-1.000000e+06
39	Crème	-1.000000e+06
40	Avoine	-1.000000e+06
41	Olives	-1.000000e+06

In [239…

```
# calcul de la balance commerciale de la Thaïlande par produit
# balance commerciale = exportations - importations : Si les in
df_thaï["balance_commerciale"] = df_thaï["Exportations - Quanti

# Grouper par produit et calculer la balance commerciale moyenn
balance_commerciale_par_produit = df_thaï.groupby('Produit')['b
balance_commerciale_par_produit.to_frame().sort_values(by='bala
```

```
C:\Users\nbous\AppData\Local\Temp\ipykernel_18392\4212853434.p
y:3: SettingWithCopyWarning:



A value is trying to be set on a copy of a slice from a DataFr
ame.

Try using .loc[row_indexer,col_indexer] = value instead



See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view
-versus-a-copy
```

Out[2398]:

	Produit	balance_commerciale
0	Manioc	2.396400e+10
1	Riz (Eq Blanchi)	6.810000e+09
2	Sucre Eq Brut	6.437000e+09
3	Fruits, Autres	2.410000e+09
4	Ananas	1.440000e+09
5	Crustacés	6.230000e+08
6	Huile de Palme	6.020000e+08
7	Viande de Volailles	5.250000e+08
8	Bananes	4.260000e+08
9	Maïs	3.850000e+08

10	Bière	2.320000e+08
11	Céréales, Autres	2.050000e+08
12	Poissons Marins, Autres	1.360000e+08
13	Perciform	1.160000e+08
14	Viande, Autre	9.200000e+07
15	Alcool, non Comestible	8.900000e+07
16	Légumes, Autres	8.600000e+07
17	Café	7.800000e+07
18	Mollusques, Autres	6.400000e+07
19	Huile de Soja	5.400000e+07
20	Huile de Palmistes	4.500000e+07
21	Viande de Bovins	4.300000e+07
22	Poissons Eau Douce	4.200000e+07
23	Huile de Son de Riz	2.900000e+07
24	Épices, Autres	2.900000e+07
25	Graisses Animales Crue	2.500000e+07
26	Animaux Aquatiques Autre	2.400000e+07
27	Thé	2.400000e+07
28	Oeufs	2.100000e+07
29	Viande de Suides	2.100000e+07
30	Pamplemousse	1.900000e+07
31	Palmistes	1.500000e+07
32	Citrons & Limes	1.400000e+07
33	Noix	1.400000e+07
34	Coco (Incl Coprah)	1.000000e+07
35	Haricots	7.000000e+06
36	Miel	5.000000e+06
37	Agrumes, Autres	4.000000e+06
38	Racines nda	4.000000e+06
39	Sorgho	2.000000e+06
40	Ignames	0.000000e+00
41	Sucre, canne	0.000000e+00

42 produits ont une balance commerciale négative □, c'est-à-dire que la quantité de ce produit importée est en moyenne supérieure à la quantité exportée.

Nous pouvons tirer quelques conclusions générales :

```
- Les produits en haut de la liste, tels que le
blé, le soja et le lait (excluant le beurre), ont
des balances commerciales négatives importantes.
Cela signifie que ces produits sont plus importés
```

qu' exportés en moyenne.

- Certains produits ont des balances commerciales moins négatives, indiquant que l'écart entre les importations et les exportations est moins important. Cela pourrait signifier que ces produits ont une situation commerciale relativement plus équilibrée.

- Certains produits, comme l'huile de coco, les graines de colza/moutarde et d'autres, ont des balances commerciales proches de zéro ou légèrement négatives. Cela pourrait suggérer que l'exportation et l'importation de ces produits sont assez équilibrées.

Etape 6 - Analyse complémentaires

In [239…

```
df_2017.head()
```

Out[2399]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponi alimen (Kcal/personne/
4	Afghanistan	Abats Comestible	animale	0.0	0.0	
10	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	
16	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	
22	Afghanistan	Ananas	vegetale	0.0	0.0	
28	Afghanistan	Bananes	vegetale	0.0	0.0	

5 rows × 21 columns

In [225…

```
#analyses complémentaires :
#"et toutes les infos que tu trouverais utiles pour mettre en
#le plus en difficulté au niveau alimentaire"

# Qui sont les plus grands importateurs de blé au monde ?

# Filtrer les lignes pour ne conserver que celles liées au pr
df_ble = df_2017[df_2017['Produit'] == 'Blé']

# Grouper par le pays (Zone) et sommer les importations de bl
result = df_ble.groupby('Zone')['Importations - Quantité'].su

# Trier les données du plus grand au plus petit
result = result.sort_values(by='Importations - Quantité', asc
```

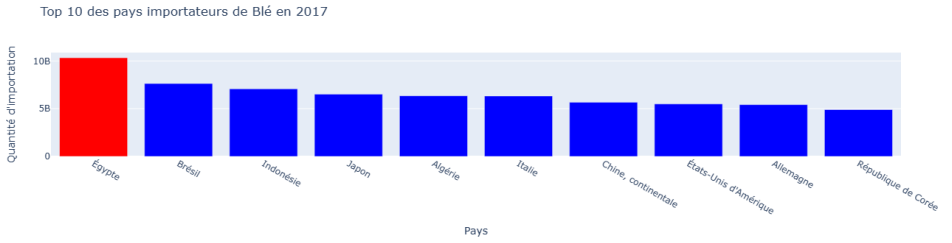
```
# Afficher le résultat
result.iloc[: 10]
```

Out[2251]:

	Zone	Importations - Quantité
166	Égypte	1.033100e+10
23	Brésil	7.630000e+09
69	Indonésie	7.074000e+09
77	Japon	6.520000e+09
3	Algérie	6.343000e+09
75	Italie	6.324000e+09
36	Chine, continentale	5.666000e+09
169	États-Unis d'Amérique	5.491000e+09
4	Allemagne	5.421000e+09
128	République de Corée	4.906000e+09

In [226…

```
import plotly.express as px
import pandas as pd

# Charger le DataFrame avec les données des importations de b
data = {
    'Zone': ['Égypte', 'Brésil', 'Indonésie', 'Japon', 'Algér
    'Importations - Quantité': [10331000000, 7630000000, 7074
}

df = pd.DataFrame(data)

# Créer un graphique à barres
fig = px.bar(df, x='Zone', y='Importations - Quantité', title

# Personnaliser la couleur de l'Égypte en utilisant color_dis
fig.update_traces(marker=dict(color=['red' if zone == 'Égypte

# Personnaliser le graphique (ajouter un titre et libellés d'
fig.update_xaxes(title_text='Pays')
fig.update_yaxes(title_text='Quantité d\'importation')

# Afficher le graphique
fig.show()
```



In [240…

```
# Qui sont les plus grands exportateurs de blé au monde ?
# 2 des exportateurs de blé les plus importants sont engagés

# Grouper par le pays (Zone) et sommer les importations de bl
result_export = df_ble.groupby('Zone')['Exportations - Quanti

# Trier les données du plus grand au plus petit
result_export = result_export.sort_values(by='Exportations -
```

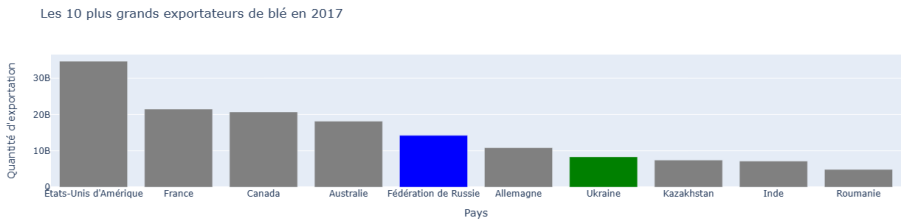
```
# Afficher le résultat
result_export.iloc[: 10]
```

Out[2400]:

	Zone	Exportations - Quantité
169	États-Unis d'Amérique	3.469100e+10
53	France	2.150200e+10
31	Canada	2.070400e+10
10	Australie	1.817100e+10
54	Fédération de Russie	1.424300e+10
4	Allemagne	1.087000e+10
158	Ukraine	8.331000e+09
79	Kazakhstan	7.442000e+09
68	Inde	7.168000e+09
125	Roumanie	4.866000e+09

In [240…

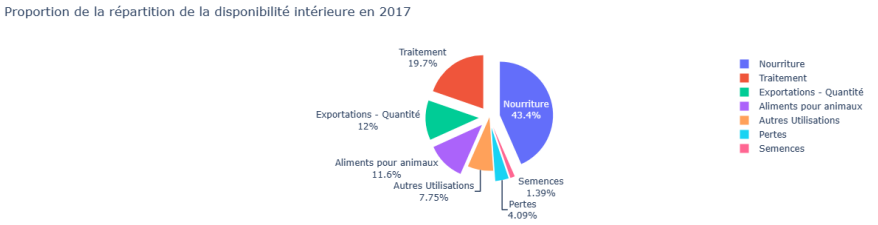
```
import plotly.express as px
import pandas as pd

# Charger le DataFrame avec les données des exportations de b
data = {
    'Zone': ['États-Unis d\'Amérique', 'France', 'Canada', 'A
    'Exportations - Quantité': [34691000000, 21502000000, 207
}

df = pd.DataFrame(data)

# Créer un graphique à barres
fig = px.bar(df, x='Zone', y='Exportations - Quantité', title

# Personnaliser la couleur de la Russie et de l'Ukraine en ut
fig.update_traces(marker=dict(color=['blue' if zone == 'Fédér

# Personnaliser le graphique (ajouter un titre et libellés d'
fig.update_xaxes(title_text='Pays')
fig.update_yaxes(title_text='Quantité d\'exportation')

# Afficher le graphique
fig.show()
```



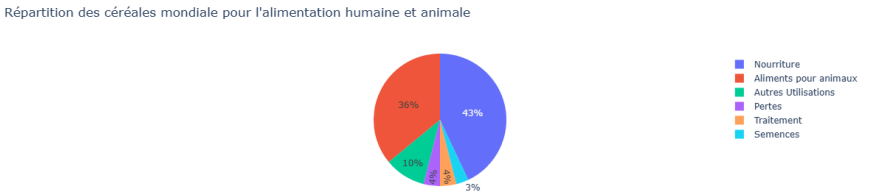
In [ ]:

```
import plotly.express as px

# Les pourcentages que vous avez calculés
pourcentages = [((df_2017[c].sum() / food_availability) * 100

# Création d'un DataFrame pour les données du graphique
data = {'Catégorie': colonnes, 'Pourcentage': pourcentages}

# Création du graphique circulaire avec les chiffres à l'inté
fig = px.pie(data, values='Pourcentage', names='Catégorie', t
```

```
                hover_data=['Catégorie', 'Pourcentage'],
                labels={'Catégorie': 'Catégorie'})

# Ajouter les chiffres à l'intérieur du graphique
fig.update_traces(textinfo='percent+label', pull=[0.2, 0.2, 0

# Afficher le graphique
fig.show()
```



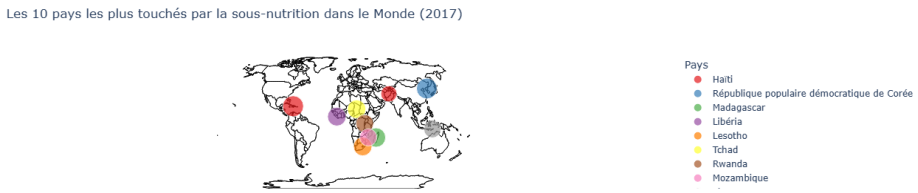
In [ ]:
```
import plotly.express as px

valeurs = [36, 4, 43, 3, 4, 10]
labels = ["Aliments pour animaux", "Pertes", "Nourriture", "S

# Création de la figure avec Plotly
fig = px.pie(names=labels, values=valeurs, title="Répartition
fig.show()
```



In [ ]:
```
import plotly.express as px
import pandas as pd

# Créez un DataFrame avec vos données
data = pd.DataFrame({
    'Pays': ['Haïti', 'République populaire démocratique de C
    'Latitude': [18.9712, 40.3399, -18.8792, 6.4281, -29.6090
    'Longitude': [-72.2852, 127.5101, 46.5850, -9.4295, 28.23
    'Sous_Nutrition_Pourcentage': [48.26, 47.19, 41.06, 38.28
})

# Créez une carte du monde avec Plotly Express
fig = px.scatter_geo(data,
                     lat='Latitude',
                     lon='Longitude',
                     hover_name='Pays',
                     size='Sous_Nutrition_Pourcentage',
                     projection='natural earth',
                     title='Les 10 pays les plus touchés par

# Personnalisez la carte
fig.update_geos(showcoastlines=True, coastlinecolor="Black",
fig.update_layout(geo=dict(showframe=False, showcountries=Tru

# Affichez la carte
fig.show()
```

In [222…

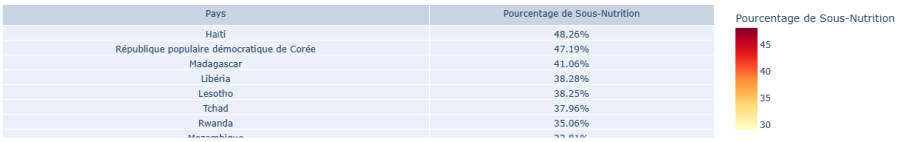
```
import plotly.express as px
import pandas as pd

# Créez un DataFrame avec vos données
data = pd.DataFrame({
    'Pays': ['Haïti', 'République populaire démocratique de C
    'Latitude': [18.9712, 40.3399, -18.8792, 6.4281, -29.6090
    'Longitude': [-72.2852, 127.5101, 46.5850, -9.4295, 28.23
    'Sous_Nutrition_Pourcentage': [48.26, 47.19, 41.06, 38.28
})

# Créez une liste de couleurs uniques pour chaque pays
colors = px.colors.qualitative.Set1[:len(data)]

# Créez une carte du monde avec Plotly Express en utilisant l
fig = px.scatter_geo(data,
                     lat='Latitude',
                     lon='Longitude',
                     hover_name='Pays',
                     size='Sous_Nutrition_Pourcentage',
                     color='Pays',  # Utilisez la colonne 'Pa
                     projection='natural earth',
                     title='Les 10 pays les plus touchés par
                     color_discrete_sequence=colors)  # Utili

# Personnalisez la carte
fig.update_geos(showcoastlines=True, coastlinecolor="Black",
fig.update_layout(geo=dict(showframe=False, showcountries=Tru

# Affichez la carte
fig.show()
```



In [222…

```
import plotly.graph_objects as go
import pandas as pd

# Créez un DataFrame avec vos données
data = pd.DataFrame({
    'Pays': ['Haïti', 'République populaire démocratique de C
    'Latitude': [18.9712, 40.3399, -18.8792, 6.4281, -29.6090
    'Longitude': [-72.2852, 127.5101, 46.5850, -9.4295, 28.23
    'Sous_Nutrition_Pourcentage': [48.26, 47.19, 41.06, 38.28
})

# Créez une légende personnalisée sous forme de tableau
legend_table = go.Table(
    header=dict(values=['Pays', 'Pourcentage de Sous-Nutritio
    cells=dict(values=[data['Pays'], data['Sous_Nutrition_Pou
```

```
    columnwidth=[0.6, 0.4],
    visible=True
)

# Créez une carte du monde avec Plotly Express
fig = go.Figure(go.Scattergeo(
    lat=data['Latitude'],
    lon=data['Longitude'],
    mode='markers',
    hoverinfo='text',
    marker=dict(
        size=data['Sous_Nutrition_Pourcentage'],
        colorscale='YlOrRd',
        color=data['Sous_Nutrition_Pourcentage'],
        colorbar=dict(title='Pourcentage de Sous-Nutrition')
    ),
    text=data['Pays'] + '<br>' + data['Sous_Nutrition_Pourcen
))

# Personnalisez la carte
fig.update_geos(showcoastlines=True, coastlinecolor="Black",
fig.update_layout(geo=dict(showframe=False, showcountries=Tru

# Ajoutez la légende personnalisée au-dessus de la carte
fig.add_trace(legend_table)

# Affichez la carte
fig.show()
```

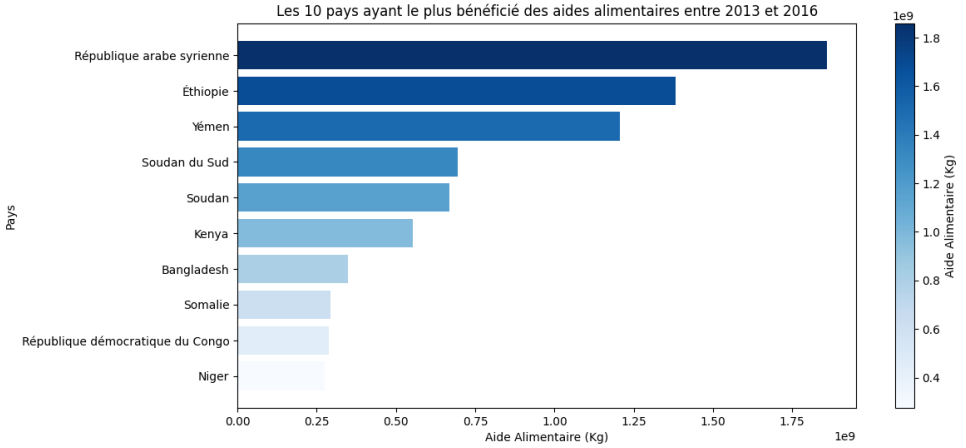


In [222…

```
import matplotlib.pyplot as plt
import numpy as np

# Créez un DataFrame avec vos données
data = pd.DataFrame({
    'Zone': ['République arabe syrienne', 'Éthiopie', 'Yémen'
    'Aide_alimentaire_Kg': [1858943000, 1381294000, 120648400
})

# Trier les données en fonction de la quantité d'aide aliment
data = data.sort_values(by='Aide_alimentaire_Kg', ascending=T

# Extraire les noms de pays triés et les quantités d'aide ali
pays = data['Zone']
aide_alimentaire = data['Aide_alimentaire_Kg']

# Créer un dégradé de bleu pour les couleurs
colors = plt.cm.Blues(np.linspace(0, 1, len(pays)))

# Créer un graphique à barres horizontal avec le dégradé de b
plt.figure(figsize=(12, 6))
bars = plt.barh(pays, aide_alimentaire, color=colors)
plt.xlabel('Aide Alimentaire (Kg)')
plt.ylabel('Pays')
plt.title('Les 10 pays ayant le plus bénéficié des aides alim

# Ajouter une barre de couleur pour le dégradé
```

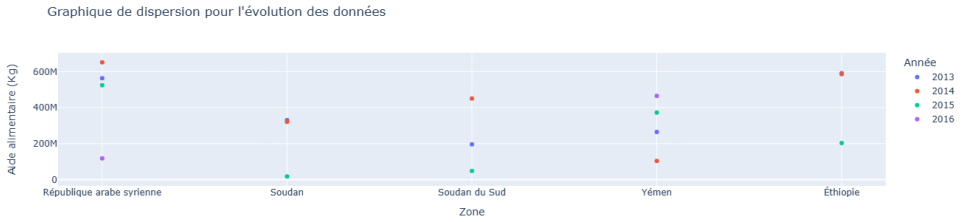
```
sm = plt.cm.ScalarMappable(cmap=plt.cm.Blues, norm=plt.Normal
sm.set_array([])
cbar = plt.colorbar(sm, orientation='vertical')
cbar.set_label('Aide Alimentaire (Kg)')

# Afficher le graphique
plt.show()
```

```
C:\Users\nbous\AppData\Local\Temp\ipykernel_18392\302421641.p
y:30: MatplotlibDeprecationWarning:



Unable to determine Axes to steal space for Colorbar. Using g
ca(), but will raise in the future. Either provide the *cax*
argument to use as the Axes for the Colorbar, provide the *ax
* argument to steal space from it, or add *mappable* to an Ax
es.
```



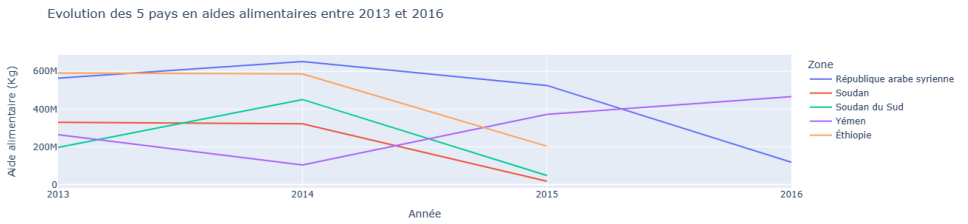
In [222…

```
import plotly.express as px
import pandas as pd

# Votre DataFrame pivoté
data = {
    'Zone': ["République arabe syrienne", "Soudan", "Soudan d
    2013: [563566000.0, 330230000.0, 196330000.0, 264764000.0
    2014: [651870000.0, 321904000.0, 450610000.0, 103840000.0
    2015: [524949000.0, 17650000.0, 48308000.0, 372306000.0,
    2016: [118558000.0, None, None, 465574000.0, None]
}

df = pd.DataFrame(data)

# Créez un scatter plot
fig = px.scatter(df.melt(id_vars='Zone', var_name='Année', va
                 x='Zone', y='Valeur', color='Année', title='
                 labels={'Valeur': 'Aide alimentaire (Kg)'},

# Personnalisez le graphique
fig.update_traces(marker=dict(size=10),
                  selector=dict(mode='markers+lines'))

# Affichez le graphique
fig.show()
```



In [222…

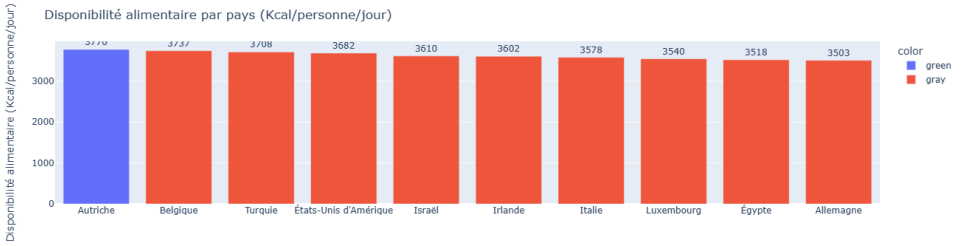
```python
import plotly.express as px
import pandas as pd

# Votre DataFrame pivoté
data = {
    'Zone': ["République arabe syrienne", "Soudan", "Soudan d
    2013: [563566000.0, 330230000.0, 196330000.0, 264764000.0
    2014: [651870000.0, 321904000.0, 450610000.0, 103840000.0
    2015: [524949000.0, 17650000.0, 48308000.0, 372306000.0,
    2016: [118558000.0, None, None, 465574000.0, None]
}

df = pd.DataFrame(data)

# Créez un graphique à courbes
fig = px.line(df.melt(id_vars='Zone', var_name='Année', value
              x='Année', y='Valeur', color='Zone', title='Evo
              labels={'Valeur': 'Aide alimentaire (Kg)'}, lin

# Supprimez les valeurs intermédiaires
fig.update_xaxes(type='category')

# Affichez le graphique
fig.show()
```



In [223…

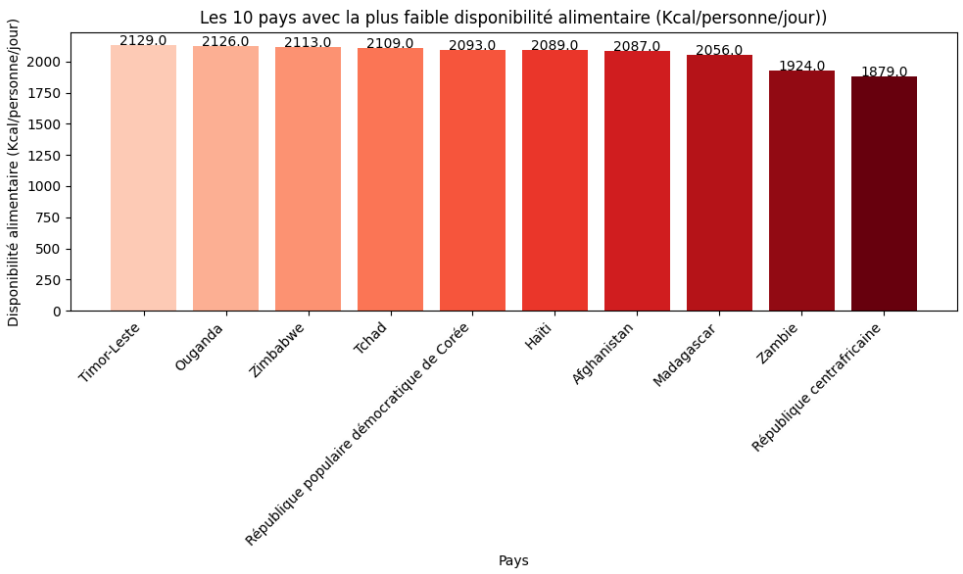
```python
import plotly.express as px
import pandas as pd

# Données
pays = ["Autriche", "Belgique", "Turquie", "États-Unis d'Amér
        "Irlande", "Italie", "Luxembourg", "Égypte", "Allemag
disponibilite = [3770.0, 3737.0, 3708.0, 3682.0, 3610.0, 3602

# Créer un DataFrame
data = pd.DataFrame({'Pays': pays, 'Disponibilité alimentaire

# Définir une couleur pour l'Autriche (vert) et les autres pa
colors = ['green' if pays == 'Autriche' else 'gray' for pays

# Créer le graphique à barres vertical avec les noms des pays
fig = px.bar(data, x='Pays', y='Disponibilité alimentaire (Kc
             color=colors, text='Disponibilité alimentaire (K
             labels={'Pays': '', 'Disponibilité alimentaire (

# Personnaliser le titre
fig.update_layout(title='Disponibilité alimentaire par pays (
                  xaxis_title='', yaxis_title='Disponibilité
```

```
# Afficher les noms des pays au-dessus des barres
fig.update_traces(textposition='outside')

# Afficher le graphique
fig.show()
```



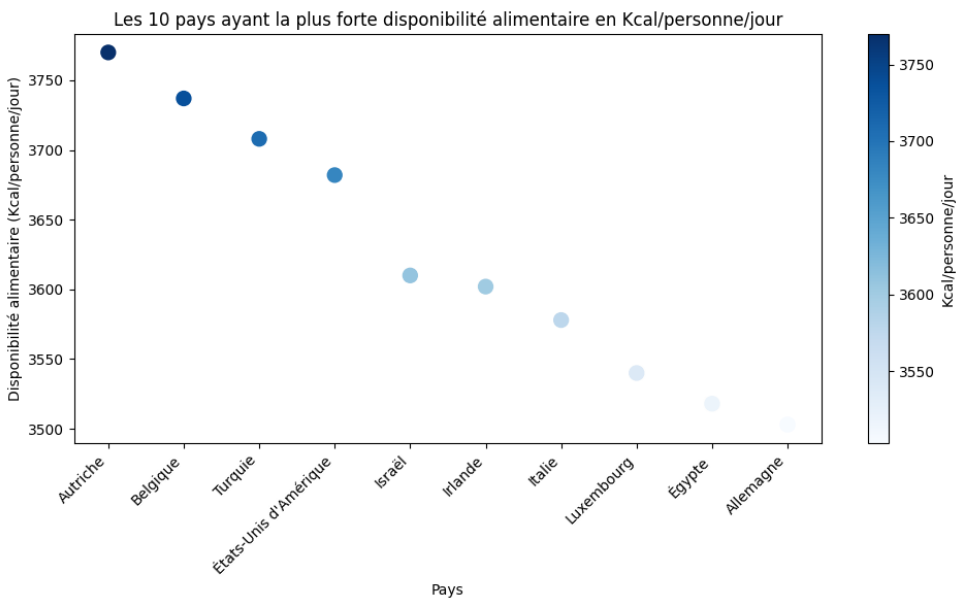
In [223…

```
import matplotlib.pyplot as plt
import numpy as np

# Données
pays = moins_dispo_alimentaire_10['Zone']
disponibilite = moins_dispo_alimentaire_10['Disponibilité ali

# Inverser l'ordre des données pour avoir le classement crois
pays = pays[::-1]
disponibilite = disponibilite[::-1]

# Créer une palette de couleurs en dégradé du rouge foncé à l
colors = plt.cm.Reds(np.linspace(0.2, 1, len(pays)))

# Créer le graphique à barres verticales avec le dégradé de c
plt.figure(figsize=(10, 6))
bars = plt.bar(pays, disponibilite, color=colors)

plt.ylabel('Disponibilité alimentaire (Kcal/personne/jour)')
plt.xlabel('Pays')
plt.title('Les 10 pays avec la plus faible disponibilité alim

# Ajouter des valeurs numériques au-dessus des barres
for bar, dispo in zip(bars, disponibilite):
    plt.text(bar.get_x() + bar.get_width()/2, dispo, f'{dispo

# Faire pivoter les étiquettes des pays pour une meilleure li
plt.xticks(rotation=45, ha="right")

# Afficher le graphique
plt.tight_layout()
plt.show()
```



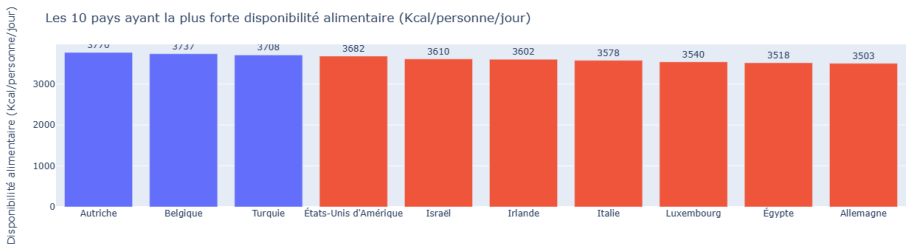
In [223…

```python
import matplotlib.pyplot as plt
import numpy as np

# Données
pays = ["Autriche", "Belgique", "Turquie", "États-Unis d'Amér
        "Irlande", "Italie", "Luxembourg", "Égypte", "Allemag
disponibilite = [3770.0, 3737.0, 3708.0, 3682.0, 3610.0, 3602

# Créer une palette de couleurs en dégradé de bleu à vert
colors = plt.cm.Blues(np.linspace(0.2, 1, len(pays)))

# Créer le graphique de dispersion avec le dégradé de couleur
plt.figure(figsize=(10, 6))
scatter = plt.scatter(pays, disponibilite, c=disponibilite, c

plt.ylabel('Disponibilité alimentaire (Kcal/personne/jour)')
plt.xlabel('Pays')
plt.title('Les 10 pays ayant la plus forte disponibilité alim

# Faire pivoter les étiquettes des pays pour une meilleure li
plt.xticks(rotation=45, ha="right")

# Ajouter un colorbar pour indiquer les valeurs associées aux
cbar = plt.colorbar(scatter)
cbar.set_label('Kcal/personne/jour')

# Afficher le graphique
plt.tight_layout()
plt.show()
```



In [223…

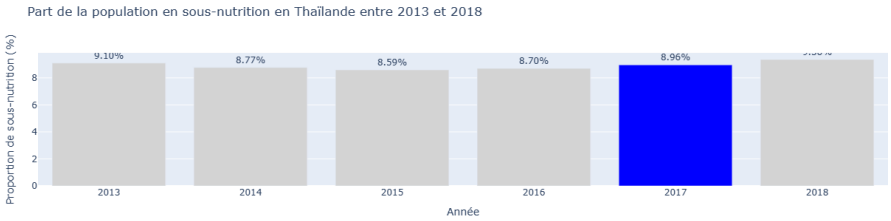
```python
import plotly.express as px
import pandas as pd

# Données
pays = ["Autriche", "Belgique", "Turquie", "États-Unis d'Amér
        "Irlande", "Italie", "Luxembourg", "Égypte", "Allemag
disponibilite = [3770.0, 3737.0, 3708.0, 3682.0, 3610.0, 3602

# Créer un DataFrame
data = pd.DataFrame({'Pays': pays, 'Disponibilité alimentaire

# Définir une couleur pour l'Autriche, la Belgique et la Turq
colors = ['gray' if pays in ['Autriche', 'Belgique', 'Turquie

# Créer le graphique à barres vertical avec les noms des pays
fig = px.bar(data, x='Pays', y='Disponibilité alimentaire (Kc
             color=colors, text='Disponibilité alimentaire (K
             labels={'Pays': '', 'Disponibilité alimentaire (

# Personnaliser le titre
fig.update_layout(title='Les 10 pays ayant la plus forte disp
                  xaxis_title='', yaxis_title='Disponibilité

# Afficher les noms des pays au-dessus des barres
fig.update_traces(textposition='outside')

# Masquer la légende
fig.update_layout(showlegend=False)

# Afficher le graphique
fig.show()
```



In [223…

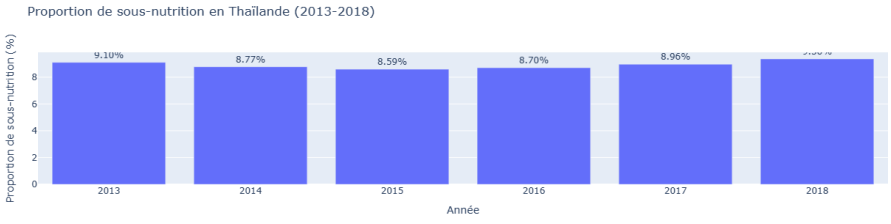
```python
import plotly.express as px
import pandas as pd

# Exemple de données (veuillez remplacer cela par vos propres
```

```
annees = [2013, 2014, 2015, 2016, 2017, 2018]
proportions = [9.10, 8.77, 8.59, 8.70, 8.96, 9.36]

# Créer un DataFrame avec les données
data = pd.DataFrame({'Année': annees, 'Proportion de sous-nut

# Créer un graphique à barres avec Plotly
fig = px.bar(data, x='Année', y='Proportion de sous-nutrition
             title='Part de la population en sous-nutrition e

# Personnaliser l'apparence du graphique
fig.update_traces(texttemplate='%{text:.2f}%', textposition='

# Mettre en relief l'année 2017 en utilisant une couleur diff
fig.update_traces(marker_color=['lightgray', 'lightgray', 'li

fig.update_xaxes(title_text='Année')
fig.update_yaxes(title_text='Proportion de sous-nutrition (%)

# Afficher le graphique
fig.show()
```



In [223…

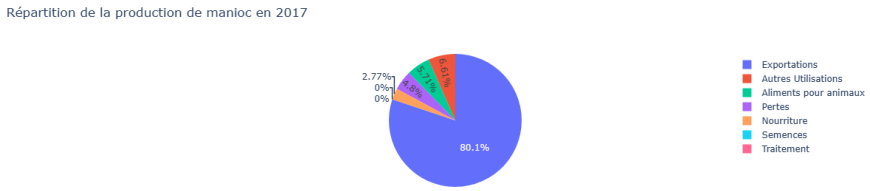
```
import plotly.express as px
import pandas as pd

# Exemple de données (veuillez remplacer cela par vos propres
annees = [2013, 2014, 2015, 2016, 2017, 2018]
proportions = [9.10, 8.77, 8.59, 8.70, 8.96, 9.36]

# Créer un DataFrame avec les données
data = pd.DataFrame({'Année': annees, 'Proportion de sous-nut

# Créer un graphique à barres avec Plotly
fig = px.bar(data, x='Année', y='Proportion de sous-nutrition
             title='Proportion de sous-nutrition en Thaïlande

# Personnaliser l'apparence du graphique
fig.update_traces(texttemplate='%{text:.2f}%', textposition='
fig.update_xaxes(title_text='Année')
fig.update_yaxes(title_text='Proportion de sous-nutrition (%)

# Afficher le graphique
fig.show()
```



In [223…

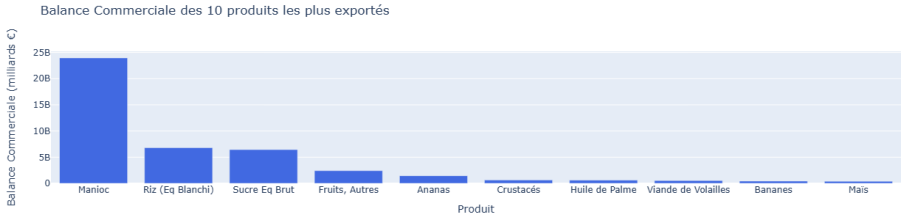
```
import plotly.express as px

# Créez un dictionnaire de données à partir des pourcentages
```

```
data = {
    'Catégorie': ['Exportations', 'Aliments pour animaux', 'P
    'Pourcentage': [83.41, 5.95, 5.0, 2.88, 0.0, 0.0, 6.88]
}

# Créez un DataFrame Pandas à partir du dictionnaire de donné
df = pd.DataFrame(data)

# Créez un graphique à secteurs (camembert) avec Plotly
fig = px.pie(df, names='Catégorie', values='Pourcentage', tit

# Affichez le graphique
fig.show()
```



In [223…

```
import plotly.express as px
import pandas as pd

# Créez un DataFrame à partir de vos nouvelles données
data = {
    "Produit": [
        "Manioc", "Riz (Eq Blanchi)", "Sucre Eq Brut", "Fruit
        "Crustacés", "Huile de Palme", "Viande de Volailles",
    ],
    "Balance Commerciale (milliards €)": [
        2.3964e10, 6.81e9, 6.437e9, 2.41e9, 1.44e9,
        6.23e8, 6.02e8, 5.25e8, 4.26e8, 3.85e8
    ]
}

df = pd.DataFrame(data)

# Créez un graphique à barres empilées avec Plotly
fig = px.bar(df, x='Produit', y='Balance Commerciale (milliar
             title='Balance Commerciale des 10 produits les p
             labels={'Balance Commerciale (milliards €)': 'Ba
             color_discrete_sequence=['royalblue'])

fig.update_layout(barmode='relative')
fig.show()
```



In [223…

```
import plotly.express as px
import pandas as pd

# Créez un DataFrame à partir de vos données
data = {
    "Produit": [
        "Blé", "Soja", "Lait - Excl Beurre", "Orge", "Pommes
```

```
        "Pommes", "Raisin", "Oranges, Mandarines", "Poissons
    ],
    "Balance Commerciale (milliards €)": [
        -1.881e9, -1.67e9, -1.041e9, -3.35e8, -2.36e8,
        -1.13e8, -1.01e8, -9.7e7, -7e7, -6.7e7
    ]
}

df = pd.DataFrame(data)

# Créez un graphique de barres avec Plotly
fig = px.bar(df, x='Balance Commerciale (milliards €)', y='Pr
fig.update_traces(texttemplate='%{text:.3s}', textposition='i
fig.update_layout(
    title='Balance Commerciale par Produit',
    xaxis_title='Balance Commerciale (milliards €)',
    yaxis_title='Produit'
)
fig.show()
```