

La sous-nutrition mondiale¹

ETUDE DE SANTE PUBLIQUE



SOMMAIRE

PARTIE 1 : Exposé du contexte

- 1.1 Définitions
- 1.2 Evolution dans le temps
- 1.3 Répartition géographique
- 1.4 Quelques chiffres
- 1.5 Les causes principales
- 1.6 Le défi de 2050 : nourrir 10 milliards de personnes

PARTIE 2 : Analyse des causes et projections

- 2.1 Sous-alimentation et pauvreté
- 2.2 Une production suffisante pour nourrir toute la population
- 2.3 Problème de répartition intérieure
- 2.4 Problème de répartition alimentaire
- 2.5 Problème de répartition des usages
- 2.6 Trouver un équilibre entre apport d'origine animale et végétale
- 2.7 Synthèse

PARTIE 3 : Analyse des données

- 3.1 Analyse des fichiers sources
- 3.2 Création du dataframe global
 - 3.2.1 Dataframe *pop*
 - 3.2.2 Dataframe *sous_alim*
 - 3.2.3 Restriction de la relation *ss_alim*
 - 3.2.4 Jointure de la relation *pop* et de la relation *ss_alim*
 - 3.2.5 Union de la relation *animaux* et *vegetaux*
 - 3.2.6 Projection de la relation *table*
 - 3.2.7 Création d'une table pivot
 - 3.2.8 Jointure de la relation *pop_globale* et de la relation *table*
 - 3.2.9 Intégration des données du dataframe *cereales*
 - 3.2.10 Agrégation avec partitionnement

PARTIE 4 : Requêtes SQL

- 4.1 Structure de la database
- 4.2 Requête 1
- 4.3 Requête 2
- 4.4 Requête 3
- 4.5 Requête 4 (code détaillé)
- 4.6 Requête 5 (code détaillé)



PARTIE 1

EXPOSÉ DU CONTEXTE

Définitions de la sous-alimentation



Définition classique

Apport alimentaire insuffisant pour combler les dépenses énergétiques journalières

Cause de carences nutritionnelles

Nouvel indicateur : l'insécurité alimentaire

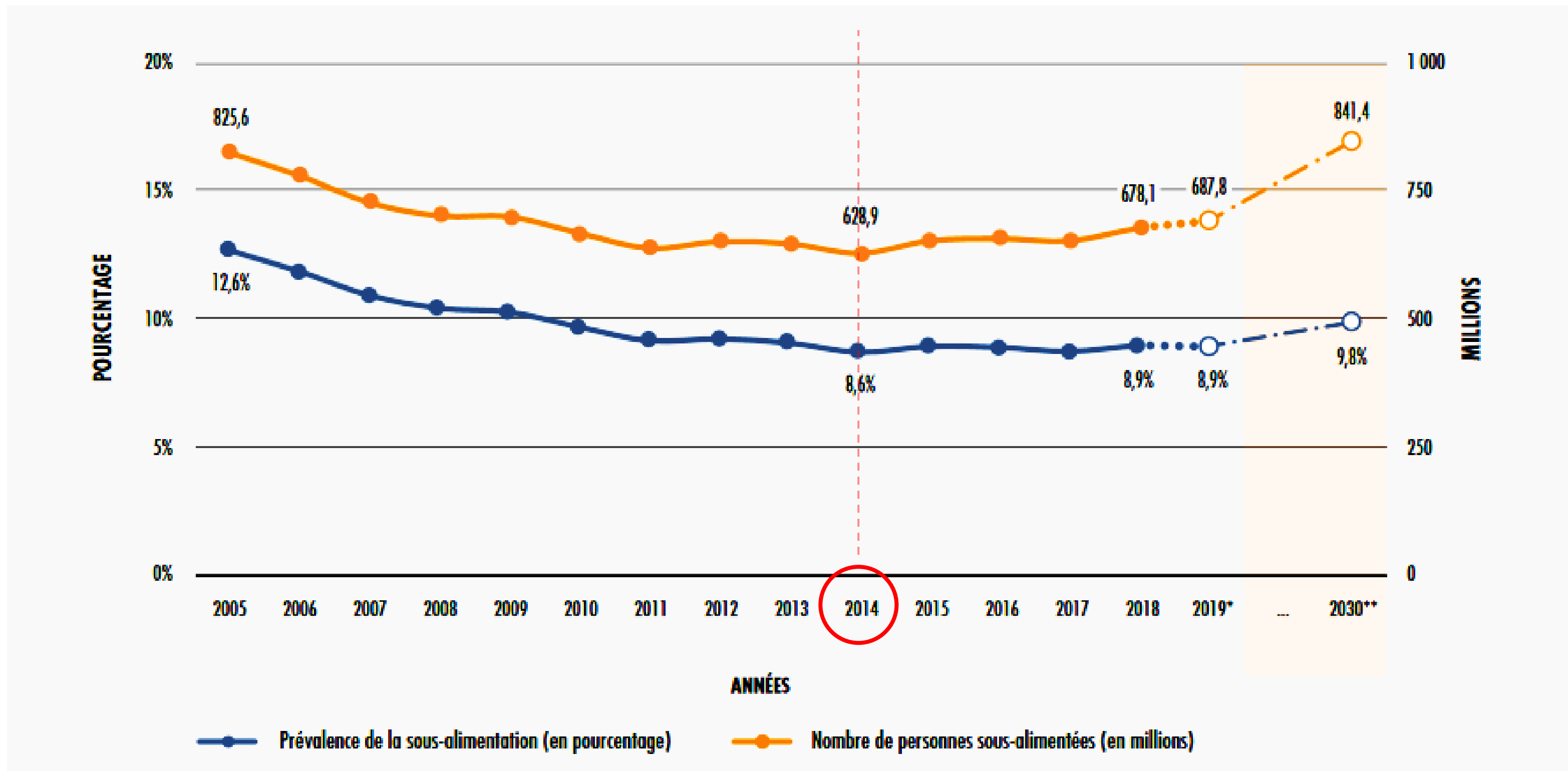
Indicateur apparu dans le rapport SOFI* de 2019

Notion plus vaste qui fait référence à l'accès régulier à une nourriture saine

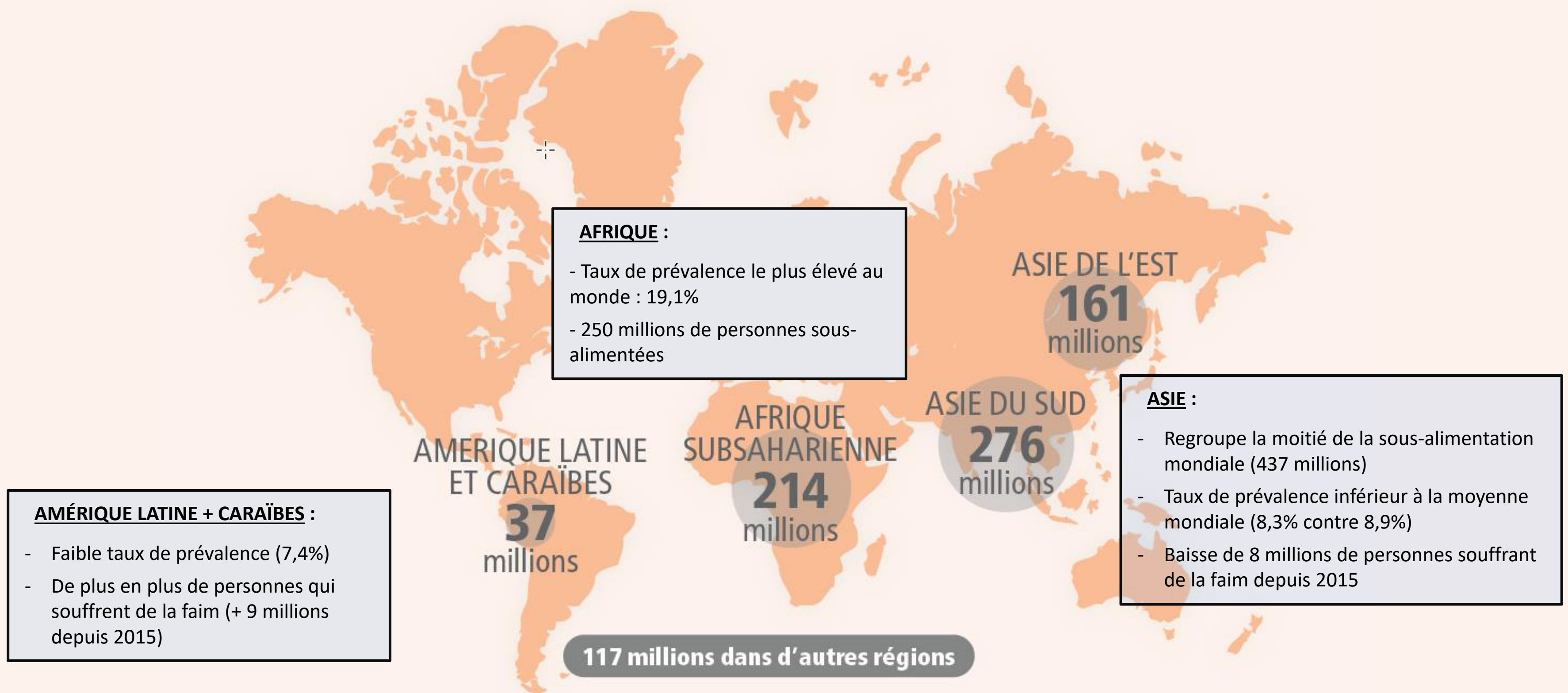


* Rapport SOFI : rapport annuel sur "L'état de la sécurité alimentaire dans le monde" par plusieurs agences des Nations unies (FAO, OMS, PAM, Unicef)

La sous-alimentation mondiale en augmentation depuis 2015



Des disparités mondiales importantes



Quelques chiffres

- 1 personne sur 9 dans le monde souffre de la faim
- 150 millions d'enfants souffrent d'un retard de croissance
- 98% des personnes sous-alimentées vivent dans les pays en développement
- 21 000 personnes meurent de faim chaque jour

Les causes de la faim dans le monde

- Les conflits armés

60% des sous-alimentés vivent dans des pays en conflit
(Yémen, Somalie, Soudan du Sud...)

- Le dérèglement climatique

cf. les phénomènes climatiques extrêmes
+ baisse de la biodiversité

- Le facteur économique

cf. la pauvreté et les inégalités sociales
+ récessions et ralentissement économiques
+ la libéralisation du marché agricole

- Les modes de consommation

cf. la part importante des apports d'origine animale dans l'alimentation

La cause économique de la faim dans le monde

- Importance du facteur économique mise en avant dans le dernier rapport SOFI *

=> 54 % des pays où la sous-alimentation a augmenté ces dernières années sont des pays dépendants des marchés internationaux de matières premières, principalement alimentaires

La rapport indique aussi "*que les pays les plus concernés sont ceux où les inégalités économiques sont fortes et où les dépenses publiques ont chuté*"

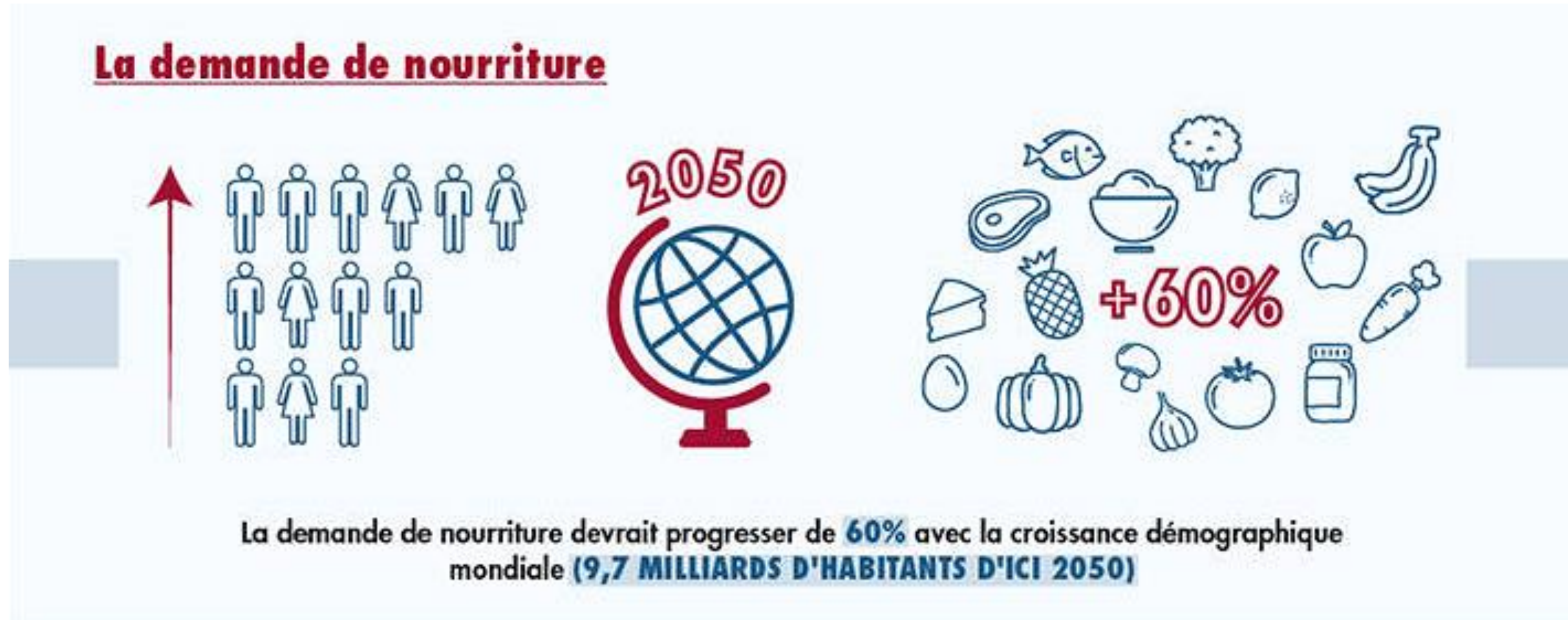
=> À mettre en relation avec la libéralisation du marché de l'alimentation

- Crise sanitaire et économique actuelle

=> aggravation de la situation : "*la pandémie de covid-19 pourrait, en 2020, ajouter 83 à 132 millions de personnes sous-alimentées*"

Perspectives pour 2050

Un défi auquel il faut se préparer





PARTIE 2

ANALYSE DES CAUSES ET PROJECTIONS

La faim touche les pays les plus pauvres

En intégrant le PIB par habitant dans nos données, on peut se rendre compte de l'ampleur du phénomène.

	pays	pib_hab_USD	part_sous_alim
0	Malawi	348	19.56
1	République centrafricaine	380	43.33
2	Congo	424	40.47
3	Éthiopie	499	26.89
4	Madagascar	541	35.77
5	Niger	552	10.66
6	Togo	621	19.07
7	Guinée-Bissau	634	23.47
8	Afghanistan	637	25.86
9	Mozambique	664	27.10
10	Gambie	700	10.82
11	Népal	715	8.99
12	Sierra Leone	716	26.26
13	Libéria	721	37.26
14	Rwanda	723	30.57
15	Guinée	769	15.33
16	Burkina Faso	787	20.08
17	Mali	805	6.54
18	Ouganda	806	33.53
19	Tchad	979	38.21

Par exemple, en **République centrafricaine**, où le **PIB par habitant** est l'un des **plus faibles** au monde (380 \$), **43,33% de la pop est sous-alimentée** (année 2013)

MÉTHODE

Intégration à nos données concernant les pays sous-alimentés des données de la banque mondiale sur le [PIB par habitant \(\\$ US courants\)](#) pour l'année 2013

	pays	pop	sous_alim
0	Afghanistan	30552000	7900000.0
1	Afrique du Sud	52776000	2600000.0
2	Albanie	3173000	200000.0

	pays	pib_hab_USD
0	Afghanistan	637.165523
1	Afrique du Sud	6832.456891
2	Albanie	4413.060861

Colonne commune :
"pays"

```
part_ss_alim = pd.merge(part_ss_alim_temp, pib_2013, on="pays", how="left")
part_ss_alim['part_ss_alim'] = part_ss_alim["sous_alim"] / part_ss_alim["pop"]
part_ss_alim['part_ss_alim'] = round(part_ss_alim['part_ss_alim'], 4)*100
part_ss_alim = part_ss_alim[["pays", "pib_hab_USD", "part_sous_alim"]]
part_ss_alim.sort_values("pib_hab_USD").head(20).reset_index(drop=True)
```

Une production suffisante pour nourrir tous les humains

La sous-alimentation est davantage liée à un **problème de répartition** qu'à un problème de production

Avec la disponibilité alimentaire mondiale totale, il est possible de nourrir :

- En termes de calories, 8 353 502 510 personnes, soit 119.38% de la population mondiale de 2013.
- En termes de protéines, 10 509 665 575 personnes, soit 150.2% de la population mondiale de 2013.

	pays	pop_(M)	code_produit	produit	nourriture_(Mt)	r_energie_poids_kcal_kg	part_proteines_poids
4144	France	64291	2731	Viande de Bovins	1531.0	1164.880692	0.144077
4145	France	64291	2732	Viande d'Ovins/Caprins	211.0	2335.500071	0.126784

1. Calcul de la disponibilité alimentaire mondiale (en terme de kcal et de protéines)

```
dispo_alim_monde_kcal = (table['nourriture_(Mt)'] * 1000000 * table['r_energie_poids_kcal_kg']).sum()
dispo_alim_monde_proteines = (table['nourriture_(Mt)'] * 1000000 * table['part_proteines_poids']).sum()
```

2. Détermination du nombre de personnes pouvant être nourries avec la capacité alimentaire actuelle

Résultat en terme de calories :

```
dispo_alim_monde_kcal_nb_pers = int(dispo_alim_monde_kcal / (2414*365))
dispo_alim_monde_kcal_relatif = ((dispo_alim_monde_kcal_nb_pers / pop_mondiale) * 100)
dispo_alim_monde_kcal_relatif = round(dispo_alim_monde_kcal_relatif, 2)
```

Résultat en terme de protéines :

```
dispo_alim_monde_proteines_nb_pers = int(dispo_alim_monde_proteines / besoins_prot_pers_an)
dispo_alim_monde_proteines_relatif = ((dispo_alim_monde_proteines_nb_pers / pop_mondiale) * 100)
dispo_alim_monde_proteines_relatif = round(dispo_alim_monde_proteines_relatif, 2)
```

Les **besoins journaliers par personne** en calories et en protéines retenus sont ceux préconisés par les autorités sanitaires, soit :

- 2414 kcal pour l'énergie ([source](#))
- 54 grammes de protéines ([source](#))

Problème de répartition interne entre exportations et consommation nationale

De nombreux pays qui connaissent la sous-alimentation exportent des produits agricoles qui pourraient être utilisés en interne

Exemple de la Thaïlande :

Les exportations de Manioc représentent 83.41% de sa production (en volume) alors que 8.36% de sa population est sous-alimentée

On peut développer cet exemple et étendre notre analyse à davantage de produits
→ soit les 6 produits suivants

Crustacés	Manioc
Viande de Volailles	Riz (Eq Blanchi)
Mollusques, Autres	Viande de Bovins

En baissant de 25% les exportations de ces 6 produits, la Thaïlande pourrait nourrir toute sa population car cette quantité représente :

- 200.38% de la population sous-alimentée en termes de calories
- 96.73% en termes de protéines.

MÉTHODE

- Dans le df ci-contre, on a calculé les exportations nettes
- Puis on calcule le nombre de personnes qu'il est possible de nourrir avec 25% du volume de ces exportations
- Enfin, on exprime ce résultat en % de la pop sous-alimentée du pays

produit	production_(Mt)	nourriture_(Mt)	exportations_nettes_(Mt)	r_energie_poids_kcal_kg	part_proteines_poids
Crustacés	718.0	96.0	623.0	509.562813	0.107008
Manioc	30228.0	871.0	23964.0	1123.261309	0.003931
Viande de Volailles	1470.0	917.0	525.0	1386.988855	0.119761
Mollusques, Autres	205.0	122.0	64.0	200.483730	0.026063
Viande de Bovins	195.0	172.0	43.0	1422.035756	0.142204
Riz (Eq Blanchi)	24054.0	7677.0	3172.0	3628.867798	0.063625

```
dispo_alim_sup_kcal_thai = (thai_ok['exportations_nettes_(Mt)'] * 1000000 * 0.25 * thai_ok["r_energie_poids_kcal_kg"]).sum()
sup_kcal = int(dispo_alim_sup_kcal_thai / (365*2414))

dispo_alim_sup_proteines_thai = (thai_ok['exportations_nettes_(Mt)'] * 1000000 * 0.25 * thai_ok["part_proteines_poids"]).sum()
sup_proteines = int(dispo_alim_sup_proteines_thai / 20)

nb_sous_alim = int(thai["nb_sous_alim_en_m"].unique()*1000000)
```

part de la pop sous-alimentée (en %) :

- En terme de calories :
`{round((sup_kcal/nb_sous_alim*100),2)}`
- En terme de protéines :
`{round((sup_proteines/nb_sous_alim)*100,2)}`

Problème de répartition alimentaire entre alimentation humaine et animale

Une part importante de la production agricole est destinée à l'alimentation animale

↳ Les exportations agricoles des pays sous-alimentés servent en partie comme alimentation des animaux

Exemple des céréales

Les céréales produites pour de l'alimentation dans le monde servent presque autant à nourrir les animaux (45%) que les humains (55%)

Les 3 produits les plus utilisés pour l'alimentation animale sont parmi les produits les plus exportés par les pays souffrant de malnutrition. C'est en particulier le cas pour le maïs, 2^{ème} produit exporté par ces pays

80% de sa production est destinée à nourrir des animaux
20% seulement pour l'alimentation humaine

MÉTHODE

Les 15 produits les plus exportés par les pays sous-alimentés :

- Huile de Palme (14.26%)
- Maïs (11.67%)
- Manioc (11.03%)
- Riz (Eq Blanchi) (10.19%)
- Sucre Eq Brut (8.03%)
- Blé (7.8%)
- Légumes, Autres (6.2%)
- Bananes (5.54%)
- Lait - Excl Beurre (4.61%)
- Fruits, Autres (4.52%)
- Soja (4.43%)
- Poissons Pelagiques (4.36%)
- Tomates (2.75%)
- Pommes (2.43%)
- Oranges, Mandarines (2.19%)

- On détermine les 15 produits les plus exportés par les pays dans lesquels la FAO recense des personnes en sous-nutrition
- > On réalise un `groupby` par produit sur lesquels on réalise la somme des exportations réalisées par tous ces pays
- > Puis on trie les résultats par ordre décroissant en limitant le résultat aux 15 premiers éléments

```
table_ss_nut_exp_sum = table_ss_nut.groupby(['produit']).sum()['exportations_(Mt)'].sort_values(ascending=False).head(15)
```

Situation paradoxale ⇒ les pays qui ont une population sous-alimentée sont aussi ceux qui vendent au reste du monde leur production agricole pour nourrir des animaux

Produits dont la part destinée à l'alimentation animale (par rapport à la part alimentaire totale) est la plus élevée

- Maïs (82.02%)
- Soja (67.4%)
- Manioc (49.77%)

```
# ratio entre la quantité destinée à la nourriture animale et  
# la quantité destinée à la nourriture animale et humaine
```

```
part_alim_anim_top3 = table_produit.sort_values('r_nour_anim_sur_nour_totale', ascending=False).head(3)
```


Problème de répartition des usages entre alimentation et autres utilisations

Environ 10% de la production végétale est utilisée à des fins non alimentaires

A plus ou moins long terme :

↳ **épuisement des ressources fossiles**
en particulier du pétrole



concurrence plus forte sur l'utilisation des productions agricoles entre :

- une finalité alimentaire
- et une finalité énergétique : **les biocarburants**

↳ Les exportations agricoles des pays sous-alimentés concernent des produits qui sont les plus utilisés pour un usage non alimentaire

Exemple des céréales

Les 3 produits les plus utilisés pour une utilisation non alimentaire sont parmi les produits les plus exportés par les pays sous-alimentés.

↳ Ainsi, l'huile de Palme, le produit le plus exporté, est aussi celui qui est le plus utilisé pour un usage non alimentaire.

Les 15 produits les plus exportés par les pays sous-alimentés :

- Huile de Palme (14.26%)
- Maïs (11.67%)
- Manioc (11.03%)
- Riz (Eq Blanchi) (10.19%)
- Sucre Eq Brut (8.03%)
- Blé (7.8%)
- Légumes, Autres (6.2%)
- Bananes (5.54%)
- Lait - Excl Beurre (4.61%)
- Fruits, Autres (4.52%)
- Soja (4.43%)
- Poissons Pelagiques (4.36%)
- Tomates (2.75%)
- Pommes (2.43%)
- Oranges, Mandarines (2.19%)

Produits les plus utilisés à des fins non alimentaires

- Huile de Palme (69.81%)
- Maïs (19.81%)
- Manioc (14.02%)

Situation délicate

- ↳ si les pays sous-alimentés continuent à se spécialiser dans ces produits agricoles
⇒ risque que les terres agricoles de ces pays soient utilisées à des fins plus industrielles qu'alimentaires

MÉTHODE

```
# ratio entre la quantité destinés aux "Autres utilisations" et la disponibilité intérieure
r_autres_top3 = table_produit.sort_values('r_autres_ut_sur_disp_int', ascending=False).head(3)
```

Défi 2050 : Nourrir 10 milliards d'êtres humains

Trouver un équilibre plus durable entre les apports nutritionnels d'origine animale et végétale

↪ Problème des apports d'origine animale : coût important

- Des surfaces agricoles considérables sont utilisées pour nourrir des animaux (1/3 des surfaces agricoles) → intérêt à les allouer à de la production directe d'aliments
- L'élevage intensif :
 - nécessite des ressources en eau importante
 - est responsable de 15% des émissions de gaz à effet de serre

Or au niveau nutritionnel : pour les pays de l'OCDE → le niveau de consommation actuel d'origine animale n'est pas justifié

➔ On pourrait donc baisser la part des apports d'origine animale

Si les Etats-Unis **diminuaient de 10%** leur production animale ⇒ **14 millions de tonnes de céréales** seraient libérées

↪ soit de quoi **nourrir 40 millions de personnes**

Quantité de céréales libérée

* Le raisonnement :

- ↪ Si baisse de 10% de la production animale, alors la quantité de céréales utilisée pour nourrir les animaux va baisser de 10%

* La méthode :

- ↪ On filtre les données : les céréales aux USA

```
usa_modif = table[(table['code_pays'] = 231) & (table['is_cereal'])]
```

- ↪ On calcule 10% de la quantité de céréales servant à l'alimentation animale

- ↪ On exprime le résultat en tonnes

```
nb_t_lib = int(usa_modif['alim_pour_animaux_(Mt)'].sum() * 1000 * 0.1)
```

```
# Calcul des kcal et des protéines des céréales destinées à nourrir les animaux
kcal_sup = (usa_modif['alim_pour_animaux_(Mt)'] * 1000000 * usa_modif['r_energie_poids_kcal_kg']).sum()
prot_sup = (usa_modif['alim_pour_animaux_(Mt)'] * 1000000 * usa_modif['part_proteines_poids']).sum()
```

```
# Calcul du nombre de personnes que l'on peut nourrir avec 10% de cette quantité
kcal_sup = int((kcal_sup / (2414*365)) * 0.1) # en terme de kcal
prot_sup = int((prot_sup / besoins_prot_pers_an) * 0.1) # en terme de protéines
```

```
print('Avec la quantité de céréales libérées, on peut nourrir :')
print(f' - {kcal_sup} personnes en terme de kcal')
print(f' - {prot_sup} personnes en terme de protéines')
```

```
Avec la quantité de céréales libérées, on peut nourrir :
- 43665474 personnes en terme de kcal
- 37689734 personnes en terme de protéines
```

Population que l'on peut nourrir

SYNTHESE

La faim dans le monde résulte davantage d'une mauvaise répartition que d'un manque de production

La disponibilité alimentaire mondiale est suffisante pour nourrir l'ensemble de la population

Il existe un problème de répartition à 3 niveaux :

- **répartition intérieure** entre consommation nationale et exportation
- **répartition alimentaire** entre alimentation humaine et animale
- **répartition des usages** entre alimentation et autres utilisations

Nourrir 10 milliards d'êtres humains en 2050 nécessite de rééquilibrer nos apports nutritionnels entre origine animale et végétale

↪ Constat

Part des apports d'origine animale trop importante dans les pays de l'OCDE au regard des besoins nutritionnels



↪ Proposition

Limiter l'apport nutritionnel à 3000 kcal par jour, dont 500 d'origine animale (actuellement le double dans les pays de l'OCDE)



PARTIE 3

ANALYSE DES DONNÉES
Fichiers sources
Fichiers d'analyse




Analyse des fichiers sources

Données extraites de la base de données de la FAO

Données sur les bilans alimentaires
(dernière MAJ : 22-12-2020)

5 csv que l'on peut classer en 2 groupes
(les fichiers d'un même groupe ont la même structure)

Population  fr_population.csv
 fr_sousalimentation.csv

Produits  fr_animaux.csv
 fr_céréales.csv
 fr_vegetaux.csv

↪ on importe ces fichiers dans notre notebook

* GROUPE POPULATION

1. fr_population.csv => df pop (175 lignes)

	Code Domaine	Domaine	Code zone	Zone	Code Élément	Élément	Code Produit	Produit	Code année	Année	Unité	Valeur	Symbole	Description du Symbole
0	FBSH	Bilans Alimentaire (Ancienne méthodologie et p...	2	Afghanistan	511	Population totale	2501	Population	2013	2013	1000 personnes	30552	NaN	Donnée officielle

2. fr_sousalimentation.csv => df ss_alim (1020 lignes)

	Code Domaine	Domaine	Code zone	Zone	Code Élément	Élément	Code Produit	Produit	Code année	Année	Unité	Valeur	Symbole	Description du Symbole	Note
0	FS	Données de la sécurité alimentaire	2	Afghanistan	6132	Valeur	210011	Nombre de personnes sous-alimentées (millions)...	20122014	2012-2014	millions	7.9	F	Estimation FAO	NaN

Remarques :

- La structure de ces 2 df sont identiques :
 - ↪ 1 seule colonne en plus dans le df `ss_alim` : "Note" → `sous_alim['Note'].nunique()` renvoie 0 → cette colonne ne contient que des NaN => on peut donc la supprimer
- Les unités concernant la valeur de la population sont différentes

Analyse des fichiers sources

* GROUPE PRODUIT

1. fr_vegetaux.csv => df vegetaux (104871 lignes)

	Code Domaine	Domaine	Code zone	Zone	Code Élément	Élément	Code Produit	Produit	Code année	Année	Unité	Valeur	Symbole	Description du Symbole
0	FBSH	Bilans Alimentaire (Ancienne méthodologie et p...	2	Afghanistan	5511	Production	2511	Blé	2013	2013	Milliers de tonnes	5169.0	S	Données standardisées
1	FBSH	Bilans Alimentaire (Ancienne méthodologie et p...	2	Afghanistan	5611	Importations - Quantité	2511	Blé	2013	2013	Milliers de tonnes	1173.0	S	Données standardisées

2. fr_animaux.csv => df animaux (37166 lignes)

	Code Domaine	Domaine	Code zone	Zone	Code Élément	Élément	Code Produit	Produit	Code année	Année	Unité	Valeur	Symbole	Description du Symbole
0	FBSH	Bilans Alimentaire (Ancienne méthodologie et p...	2	Afghanistan	5511	Production	2731	Viande de Bovins	2013	2013	Milliers de tonnes	134.0	S	Données standardisées

3. fr_céréales.csv => df cereales (891 lignes)

	Code Domaine	Domaine	Code zone	Zone	Code Élément	Élément	Code Produit	Produit	Code année	Année	Unité	Valeur	Symbole	Description du Symbole
0	FBSH	Bilans Alimentaire (Ancienne méthodologie et p...	2	Afghanistan	5511	Production	2511	Blé	2013	2013	Milliers de tonnes	5169	S	Données standardisées

Remarques :

- La structure de ces 2 df sont identiques (les colonnes sont identiques)
- Le df *cereales* est une **restriction** du df *vegetaux*
 - ⇒ on intégrera le df *cereales* en ajoutant une colonne à *is_cereal* au df *vegetaux* qui prendra un booléen comme valeur :
 - True si le produit est présent dans le df *cereales*
 - False s'il ne l'est pas

Création du df global

* OBJECTIF

On va combiner les données des différents df pour en créer un global

1. On commence par le df pop

```
pop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175 entries, 0 to 174
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  ---                ---
0   Code Domaine           175 non-null    object
1   Domaine                175 non-null    object
2   Code zone              175 non-null    int64
3   Zone                   175 non-null    object
4   Code Élément           175 non-null    int64
5   Élément                175 non-null    object
6   Code Produit           175 non-null    int64
7   Produit                175 non-null    object
8   Code année             175 non-null    int64
9   Année                  175 non-null    int64
10  Unité                  175 non-null    object
11  Valeur                 175 non-null    int64
12  Symbole                1 non-null      object
13  Description du Symbole 175 non-null    object
dtypes: int64(6), object(8)
memory usage: 19.3+ KB
```

```
pop['Symbole'].value_counts()
```

```
A    1
Name: Symbole, dtype: int64
```

La variable "Symbole" n'a qu'une valeur NaN et qui n'apparaît qu'une fois :
↳ La valeur A

1

```
pop[pop['Symbole'] == ('A')]
```

	Code Domaine	Domaine	Code zone	Zone	Code Élément	Élément	Code Produit	Produit	Code année	Année	Unité	Valeur	Symbole	Description du Symbole
33	FBSH	Bilans Alimentaire (Ancienne méthodologie et p...	351	Chine	511	Population totale	2501	Population	2013	2013	1000 personnes	1416667	A	Agrégat, peut inclure des données officielles,...

* On regarde à quelle ligne cela correspond
↳ La Chine

* On vérifie si elle apparaît plusieurs fois
↳ C'est le cas

* On supprime le doublon

```
pattern = pop.Zone.str.contains("chine", case=False)
chaine_chine = pop[pattern]
chaine_chine = chaine_chine[["Zone", "Valeur"]]
print(chaine_chine)
```

```
      Zone  Valeur
33      Chine 1416667
34  Chine - RAS de Hong-Kong    7204
35  Chine - RAS de Macao        566
36  Chine, continentale 1385567
37  Chine, Taiwan Province de  23330
```

```
pop = pop.drop(pop[pop['Symbole'] == 'A'].index)
```

2

```
pop.nunique()
```

```
Code Domaine           1
Domaine                1
Code zone              175
Zone                   175
Code Élément           1
Élément                1
Code Produit           1
Produit                1
Code année             1
Année                  1
Unité                  1
Valeur                 175
Symbole                1
Description du Symbole 2
dtype: int64
```

* De nombreuses colonnes n'ont qu'une seule valeur
↳ on peut les supprimer

* On en profite pour renommer les colonnes

```
my_columns = ['Code zone', 'Zone', 'Valeur']
pop = pop[my_columns]
pop.columns = ['code_pays', 'pays', 'pop_(en_M)']
pop.head(3)
```

	code_pays	pays	pop_(en_M)
0	2	Afghanistan	30552
1	202	Afrique du Sud	52776
2	3	Albanie	3173

3

Création du df global

On va calculer la population mondiale

En termes d'algèbre relationnelle, on va réaliser une **agrégation** en utilisant la fonction d'agrégation SOMME (SUM)

Agrégation avec partitionnement selon l'attribut pays sur lequel on applique la fonction d'agrégation suivante :

↳ la fonction SOMME (SUM) sur les populations nationales

- On utilise le df *pop*

```
pop_agregation = pop[['pays', 'pop_(en_M)']]
pop_agregation
```

- On va additionner les 174 valeurs de la colonne pop (en M)
 ↳ on additionne les populations nationales et on va obtenir en retour une seule valeur, à savoir la population mondiale

	pays	pop_(en_M)
0	Afghanistan	30552
1	Afrique du Sud	52776
2	Albanie	3173
3	Algérie	39208
...
171	Viet Nam	91680
172	Yémen	24407
173	Zambie	14539
174	Zimbabwe	14150

174 rows × 2 columns

On applique la fonction d'agrégation **SOMME**

```
(pop['pop_(en_M)'] * 1000).sum()
6997326000
```


Création du df global

2. On traite le df sous_alim

Même principe que pour le df *pop*

Particularités :

- Données présentes pour 5 intervalles de temps (2012-2014, ... , 2016-2018)
- Seuls les éléments de "Estimation FAO" contiennent des données chiffrées
- Présence de chaînes de caractères dans les valeurs de la variable valeur : "<0,1"

1

Après avoir nettoyé les données (suppression du doublon de la Chine notamment), on va traiter les particularités de ce df

- On restreint le df aux données chiffrées
- On ne garde que les colonnes utiles, en les renommant si besoin

```
ss_alim = sous_alim[sous_alim['Description du Symbole'] == 'Estimation FAO']
ss_alim = ss_alim[['Code zone', 'Zone', 'Année', 'Valeur']]
ss_alim.columns = ['code_pays', 'pays', 'annees', 'nb_sous_alim_en_m']
ss_alim.head(3)
```

	code_pays	pays	annees	nb_sous_alim_en_m
0	2	Afghanistan	2012-2014	7.9
1	2	Afghanistan	2013-2015	8.8
2	2	Afghanistan	2014-2016	9.6

2

- on remplace les intervalles d'années par l'année médiane

```
ss_alim['annees'] = ss_alim['annees'].replace({'2012-2014': str('2013')})
ss_alim['annees'] = ss_alim['annees'].apply(pd.to_numeric)
```

- Les strings '<0.1'

- On compte le nombre d'occurrences

```
ss_alim[ss_alim['nb_sous_alim_en_m'] == '<0.1'].value_counts().sum()
```

115

- 115 apparitions : non négligeable

↳ on ne va pas les supprimer mais les remplacer par un *float* 0.05

```
ss_alim['nb_sous_alim_en_m'] = ss_alim['nb_sous_alim_en_m'].replace('<0.1', float(0.05))
ss_alim['nb_sous_alim_en_m'] = ss_alim['nb_sous_alim_en_m'].apply(pd.to_numeric)
```

On obtient le df si contre :

```
ss_alim.head()
```

	code_pays	pays	annees	nb_sous_alim_en_m
0	2	Afghanistan	2013	7.9
1	2	Afghanistan	2014	8.8
2	2	Afghanistan	2015	9.6
3	2	Afghanistan	2016	10.2
4	2	Afghanistan	2017	10.6

Création du df global

3. On limite le df ss_alim aux données de l'année 2013

En termes d'algèbre relationnelle, on va réaliser une **restriction**
 ⇔ on filtre les lignes selon une condition

	code_pays	pays	annees	nb_sous_alim_en_m
0	2	Afghanistan	2013	7.9
1	2	Afghanistan	2014	8.8
2	2	Afghanistan	2015	9.6
3	2	Afghanistan	2016	10.2
4	2	Afghanistan	2017	10.6
5	202	Afrique du Sud	2013	2.6
6	202	Afrique du Sud	2014	2.8
7	202	Afrique du Sud	2015	3.2
8	202	Afrique du Sud	2016	3.4
9	202	Afrique du Sud	2017	3.5
10	3	Albanie	2013	0.2
11	3	Albanie	2014	0.2
12	3	Albanie	2015	0.2
13	3	Albanie	2016	0.2
14	3	Albanie	2017	0.2

	code_pays	pays	annees	nb_sous_alim_en_m
0	2	Afghanistan	2013	7.9
5	202	Afrique du Sud	2013	2.6
10	3	Albanie	2013	0.2

Restriction de la relation ss_alim étant donnée la condition `annees = 2013`

```
ss_alim_2013 = ss_alim[ss_alim['annees'] == 2013]
```


Création du df global

4. On regroupe les df pop et ss_alim_2013

En termes d'algèbre relationnelle, on va réaliser une **jointure**
 ⇔ on assemble les colonnes de 2 df avec la fonction **pd.merge**

1

code_pays	pays	pop_en_M
0	2	Afghanistan 30552
1	202	Afrique du Sud 52776
2	3	Albanie 3173

Colonnes communes qui vont permettre de faire la jointure

code_pays	pays	annees	nb_sous_alim_en_m
0	2	Afghanistan	2013 7.9
1	2	Afghanistan	2014 8.8
2	2	Afghanistan	2015 9.6

Mais d'abord, on s'occupe des unités des populations :
 - pop en milliers
 - Nb_sous_alim en millions
 ↪ On les convertit en personnes

Jointure interne de la relation pop et de la relation ss_alim selon la condition
pop.code_pays = ss_alim.code_pays

```
pop_global = pd.merge(pop, ss_alim_2013, how='left')
```

2

code_pays	pays	pop	code_pays	pays	annees	nb_sous_alim
0	2	Afghanistan 30552000	0	2	Afghanistan	2013 7900000
1	202	Afrique du Sud 52776000	5	202	Afrique du Sud	2013 2600000
2	3	Albanie 3173000	10	3	Albanie	2013 200000

* Le df pop contient tous les pays (174 lignes) alors que ss_alim_2013 ne contient que les pays qui ont une population sous-alimentée (119 lignes)
 ↪ on va donc merger depuis le df pop car on veut garder tous les pays

```
pop_global = pd.merge(pop, ss_alim_2013, how='left')
```

code_pays	pays	pop	annees	nb_sous_alim
0	2	Afghanistan	30552000	2013 7900000.0
1	202	Afrique du Sud	52776000	2013 2600000.0
2	3	Albanie	3173000	2013 200000.0
3	4	Algérie	39208000	2013 1700000.0
4	79	Allemagne	82727000	2013 NaN
...
169	236	Venezuela (République bolivarienne du)	30405000	2013 1900000.0
170	237	Viet Nam	91680000	2013 10400000.0
171	249	Yémen	24407000	2013 7200000.0
172	251	Zambie	14539000	2013 7000000.0
173	181	Zimbabwe	14150000	2013 6600000.0

174 rows × 5 columns

Création du df global

5. On regroupe les df vegetaux et animaux

En termes d'algèbre relationnelle, on va réaliser une **union**

⇔ on assemble les lignes de 2 df avec la fonction **pd.concat**

Union de la relation **animaux** et **vegetaux** de même schéma qui produit une nouvelle relation **table** également de même schéma

```
table = pd.concat([vegetaux, animaux])
```

- Avant de concaténer, on ajoute une colonne **origine** à chaque df
- Les df **vegetaux** et **animaux** ont le même schéma

```
vegetaux.insert(7, 'origine', 'vegetale')
animaux.insert(7, 'origine', 'animale')
```

Code Domaine	Domaine	Code zone	Zone	Code Élément	Élément	Code Produit	origine	Produit	Code année	Année	Unité	Valeur	Symbole	Description du Symbole
-----------------	---------	--------------	------	-----------------	---------	-----------------	---------	---------	---------------	-------	-------	--------	---------	---------------------------

=> on peut donc effectuer la concaténation

```
table = pd.concat([vegetaux, animaux])
```

```
table.sample(3)
```

Code Domaine	Domaine	Code zone	Zone	Code Élément	Élément	Code Produit	origine	Produit	Code année	Année	Unité	Valeur	Symbole	Description du Symbole	
20859	FBSH	Bilans Alimentaire (Ancienne méthodologie et p...	256	Luxembourg	5611	Importations - Quantité	2764	animale	Poissons Marins, Autres	2013	2013	Milliers de tonnes	2.00	S	Données standardisées
11029	FBSH	Bilans Alimentaire (Ancienne méthodologie et p...	58	Équateur	645	Disponibilité alimentaire en quantité (kg/pers...	2733	animale	Viande de Suides	2013	2013	kg	14.44	Fc	Donnée calculée
68620	FBSH	Bilans Alimentaire (Ancienne méthodologie et p...	147	Namibie	5072	Variation de stock	2620	vegetale	Raisin	2013	2013	Milliers de tonnes	0.00	S	Données standardisées

Création du df global

6. On supprime les colonnes inutiles dans le df table

En termes d'algèbre relationnelle, on va réaliser une **projection**

⇔ on sélectionne les colonnes qui nous intéressent dans le df

- On choisit les colonnes que l'on va garder

~~Code~~ ~~Domaine~~ ~~Domaine~~ Code zone Zone Code Élément Élément Code Produit origine Produit ~~Code~~ ~~année~~ Année ~~Unité~~ Valeur ~~Symbole~~ ~~Description du~~ ~~Symbole~~

```
table.drop(['Code Domaine', 'Domaine', 'Code année', 'Unité', 'Symbole', 'Description du Symbole'], axis=1, inplace=True)
```

- On renomme les colonnes

```
table.rename(columns={'Code zone': 'code_pays', 'Zone': 'pays', 'Code Élément': 'code_element', 'Élément': 'element',
                    'Code Produit': 'code_produit', 'Produit': 'produit', 'Année': 'annees', 'Valeur': 'valeur'}, inplace=True)
```

- On obtient le df suivant

	code_pays	pays	code_element	element	code_produit	origine	produit	annees	valeur
0	2	Afghanistan	5511	Production	2511	vegetale	Blé	2013	5169.00
1	2	Afghanistan	5611	Importations - Quantité	2511	vegetale	Blé	2013	1173.00
2	2	Afghanistan	5072	Variation de stock	2511	vegetale	Blé	2013	-350.00

Projection de la relation table sur ses attributs code_pays, pays, code_element, element, code_produit, origine, produit, annees et valeur

```
table.drop(['Code Domaine', 'Domaine', 'Code année', 'Unité', 'Symbole', 'Description du Symbole'], axis=1, inplace=True)
```

Création du df global

7. On réorganise le df pour pouvoir comparer les éléments

	code_pays	pays	code_element	element	code_produit	origine	produit	annees	valeur
0	2	Afghanistan	5511	Production	2511	vegetale	Blé	2013	5169.00
1	2	Afghanistan	5611	Importations - Quantité	2511	vegetale	Blé	2013	1173.00
2	2	Afghanistan	5072	Variation de stock	2511	vegetale	Blé	2013	-350.00
3	2	Afghanistan	5301	Disponibilité intérieure	2511	vegetale	Blé	2013	5992.00
4	2	Afghanistan	5527	Semences	2511	vegetale	Blé	2013	322.00
5	2	Afghanistan	5123	Pertes	2511	vegetale	Blé	2013	775.00
6	2	Afghanistan	5142	Nourriture	2511	vegetale	Blé	2013	4895.00
7	2	Afghanistan	645	Disponibilité alimentaire en quantité (kg/pers...	2511	vegetale	Blé	2013	160.23
8	2	Afghanistan	664	Disponibilité alimentaire (Kcal/personne/jour)	2511	vegetale	Blé	2013	1369.00
9	2	Afghanistan	674	Disponibilité de protéines en quantité (g/pers...	2511	vegetale	Blé	2013	36.91
10	2	Afghanistan	684	Disponibilité de matière grasse en quantité (g...	2511	vegetale	Blé	2013	4.69
11	2	Afghanistan	5511	Production	2805	vegetale	Riz (Eq Blanchi)	2013	342.00
12	2	Afghanistan	5611	Importations - Quantité	2805	vegetale	Riz (Eq Blanchi)	2013	119.00
13	2	Afghanistan	5301	Disponibilité intérieure	2805	vegetale	Riz (Eq Blanchi)	2013	461.00

Les données ne sont pas organisées de manière optimale

Il faudrait :

* 1 ligne par pays / produit

* les element en colonnes



On va créer une **table pivot**

	element	pays	produit	valeur	Aliments pour animaux	Autres utilisations (non alimentaire)	Disponibilité intérieure	Exportations - Quantité	Importations - Quantité	Nourriture	Pertes	Production	Semences	Traitement
101	Afghanistan	Blé			NaN	NaN	5992.0	NaN	1173.0	4895.0	775.0	5169.0	322.0	NaN

Création du df global

7. On réorganise le df pour pouvoir comparer les éléments

On va donc :

- * passer en index les éléments que l'on avait en colonne \longrightarrow `index=['code_pays', 'pays', 'origine', 'annees', 'code_produit', 'produit']`
- * passer les modalités possibles de la variable element en colonne \longrightarrow `columns=['element']`
- * ces modalités auront comme valeurs celles associées à la variable valeur \longrightarrow `values=['valeur']`

- On utilise la fonction **pivot_table** pour réorganiser notre df

```
table = table.pivot_table(index=['code_pays', 'pays', 'origine', 'annees', 'code_produit', 'produit'],
                           columns=['element'], values=['valeur'])
table = table.reset_index()
```

On va transformer notre table multi index en table classique avec un seul index
On en profite pour renommer les colonnes

```
table.columns = ['code_pays', 'pays',
                 'origine', 'annees',
                 'code_produit', 'produit',
                 'alim_pour_animaux_(Mt)',
                 'autres_utilisations_(Mt)',
                 'dispo_alim_(kcal-p-j)',
                 'dispo_alim_qte_(kg-p-an)',
                 'dispo_mat_grasse_(g-p-j)',
                 'dispo_proteines_qte_(g-p-j)',
                 'dispo_interieure_(Mt)',
                 'exportations_(Mt)',
                 'importations_(Mt)',
                 'nourriture_(Mt)',
                 'pertes_(Mt)',
                 'production_(Mt)',
                 'semences_(Mt)',
                 'traitement_(Mt)',
                 'var_stocks_(Mt)']
```

table.sample(4)												
code_pays	pays	origine	annees	code_produit	produit	...	nourriture_(Mt)	pertes_(Mt)	production_(Mt)	semences_(Mt)	traitement_(Mt)	var_stocks_(Mt)
6853	108	Kazakhstan	vegetale	2625	Fruits, Autres	...	492.0	5.0	75.0	NaN	NaN	0.0
9188	143	Maroc	vegetale	2555	Soja	...	0.0	2.0	1.0	NaN	62.0	0.0
14698	236	Venezuela (République bolivarienne du)	vegetale	2618	Ananas	...	393.0	70.0	462.0	NaN	NaN	0.0
10555	166	Panama	vegetale	2535	Ignames	...	20.0	1.0	24.0	2.0	NaN	NaN

4 rows × 21 columns

Création du df global

8. On intègre les données de la population (pop générale et pop sous-alimentée) au df principal table

En termes d'algèbre relationnelle, on va réaliser une **jointure**
 ⇔ on assemble les colonnes de 2 df avec la fonction **pd.merge**

Jointure interne de la relation **pop_globale** et de la relation **table** selon la condition **pop_globale.code_pays = table.code_pays**

```
table = pd.merge(pop_global, table, how="left")
```

	code_pays	pays	pop	annees	nb_sous_alim								
0	2	Afghanistan	30552000	2013	7900000.0								
1	202	Afrique du Sud	52776000	2013	2600000.0								

	code_pays	pays	origine	annees	code_produit	produit	...	nourriture_(Mt)	pertes_(Mt)	production_(Mt)	semences_(Mt)	traitement_(Mt)	var_stocks_(Mt)
6853	108	Kazakhstan	vegetale	2013	2625	Fruits, Autres	...	492.0	5.0	75.0	NaN	NaN	0.0
9188	143	Maroc	vegetale	2013	2555	Soja	...	0.0	2.0	1.0	NaN	62.0	0.0

Colonne communes qui vont permettre de faire la jointure

Remarque :

↳ Le doublon de la Chine n'a pas été traité dans le df **table** (contrairement au df **pop_global**)

```
table[table["pays"] == "Chine"].shape[0]
```

97

↳ On va donc merger depuis le df **pop_global** pour que le traitement du doublon soit pris en compte

```
table = pd.merge(pop_global, table, how="left")
```

	code_pays	pays	pop	annees	nb_sous_alim	origine	...	nourriture_(Mt)	pertes_(Mt)	production_(Mt)	semences_(Mt)	traitement_(Mt)	var_stocks_(Mt)
7460	106	Italie	60990000	2013	NaN	vegetale	...	90.0	NaN	NaN	NaN	24.0	NaN
7097	103	Iraq	33765000	2013	9100000.0	vegetale	...	49.0	NaN	0.0	NaN	NaN	NaN
10692	153	Nouvelle-Calédonie	256000	2013	50000.0	vegetale	...	0.0	NaN	NaN	NaN	NaN	NaN

Création du df global

9. On intègre les données du df cereales au df table

* On simplifie le df cereales

```
cereales = cereales[['Code Produit', 'Produit']]
cereales_prod = cereales.drop_duplicates(keep = 'first')
cereales_prod.columns = ['code_cereale', 'cereale']
cereales_prod = cereales_prod.reset_index(drop=True)
cereales_prod
```

	code_cereale	cereale
0	2511	Blé
1	2805	Riz (Eq Blanchi)
2	2513	Orge
3	2514	Maïs
4	2517	Millet
5	2515	Seigle
6	2516	Avoine
7	2518	Sorgho
8	2520	Céréales, Autres

* On crée une colonne is_cereal avec pour valeurs possibles False / True

↳ on initialise cette variable à False

```
table.insert(7, 'is_cereal', False)
```

* On crée une liste contenant les céréales

```
cereales_list = list(cereales_prod.cereale)
```

On met à jour les données de la colonne is_cereal

↳ si table['produit'] est présent dans la liste 'cereales_list', alors la variable is_cereal prend la valeur True

```
table.loc[table['produit'].isin(cereales_list), 'is_cereal'] = True
```

Toutes les données initiales ont été réunies au sein de ce df_global

	pays	pop	nb_sous_alim	origine	code_produit	is_cereal	produit	alim_pour_animaux_(Mt)	autres_utilisations_(Mt)	dispo_alim_(kcal-p-j)
8851	Madagascar	22925000	8199999.0	vegetale	2614	False	Agrumes, Autres	NaN	NaN	0.0
6625	Hongrie	9955000	NaN	vegetale	2601	False	Tomates	0.0	NaN	8.0
9477	Maurice	1244000	50000.0	vegetale	2574	False	Huile de Colza&Moutarde	NaN	NaN	10.0
9987	Myanmar	53259000	6600000.0	vegetale	2513	True	Orge	NaN	NaN	0.0
12433	République-Unie de Tanzanie	49253000	16399999.0	vegetale	2556	False	Arachides Decortiquees	NaN	210.0	82.0

Utilisation du df global

10. On souhaite connaître les 15 produits les plus exportés par les pays sous-alimentés

En termes d'algèbre relationnelle, on va réaliser une **agrégation**

↔ on sélectionne des groupes de lignes pour effectuer un calcul et renvoyer une unique valeur pour chaque groupe grâce à la fonction **groupby**

- On réalise un **groupby** par produits sur lesquels on réalise la somme des exportations réalisées par tous ces pays
- Puis on trie les résultats par ordre décroissant en limitant l'affichage aux 15 premiers éléments

```
table_ss_nut_exp_sum = table_ss_nut.groupby(['produit']).sum()['exportations_(Mt)'].sort_values(ascending=False).head(15)
table_ss_nut_exp_sum
```

produit	exportations_(Mt)
Huile de Palme	46324.0
Maïs	37906.0
Manioc	35851.0
Riz (Eq Blanchi)	33093.0
Sucre Eq Brut	26098.0
Blé	25342.0
Légumes, Autres	20139.0
Bananes	17988.0
Lait - Excl Beurre	14973.0
Fruits, Autres	14690.0
Soja	14391.0
Poissons Pelagiques	14181.0
Tomates	8935.0
Pommes	7897.0
Oranges, Mandarines	7106.0

35 851 correspond à la somme des exportations de Manioc réalisées par ces pays

groupby par produits revient ici à calculer automatiquement la somme des exportations de tous les produits.

Voici à quoi cela correspond pour par exemple le produit "Manioc"

```
table_manioc = table_ss_nut[table_ss_nut['produit'] == "Manioc"]
table_manioc = table_manioc.dropna(subset=['exportations_(Mt)'])
table_manioc = table_manioc[table_manioc['exportations_(Mt)'] != 0]
table_manioc[['pays', 'exportations_(Mt)']]
```

	pays	exportations_(Mt)
91	Afrique du Sud	2.0
462	Angola	0.0
637	Arabie saoudite	0.0
...
15182	Venezuela (République bolivarienne du)	0.0
15275	Viet Nam	8973.0
15450	Zambie	1.0

On calcule la somme de cette colonne

```
table_manioc['exportations_(Mt)'].sum()
```

35851.0

Agrégation avec partitionnement selon l'attribut produit sur lequel on applique la fonction d'agrégation suivante :

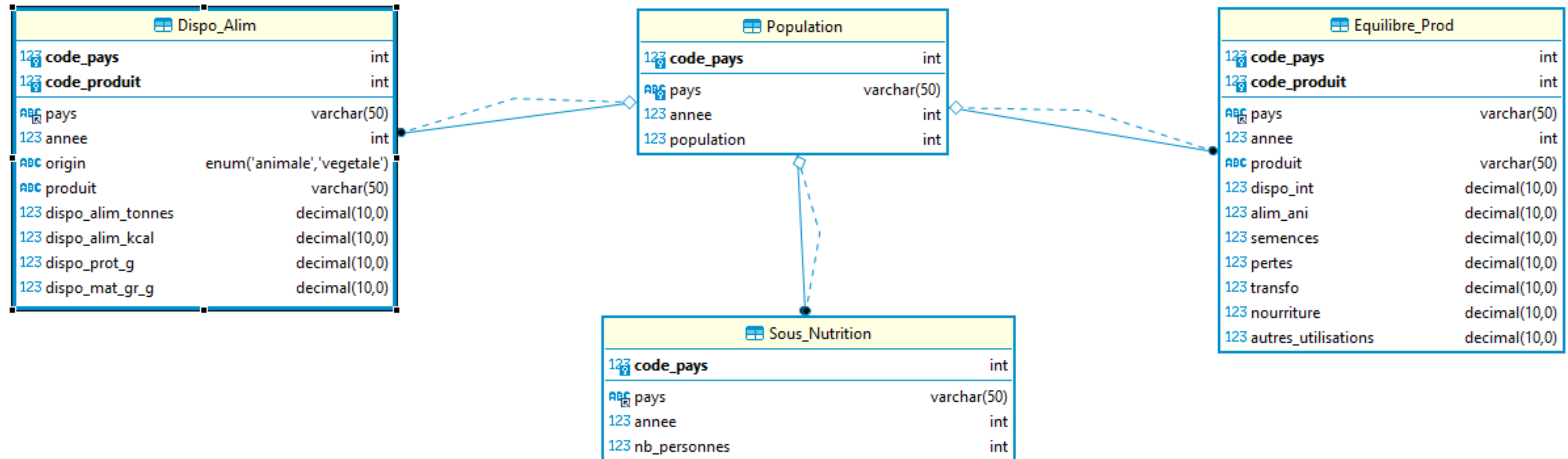
↳ la fonction SOMME (sum) sur les exportations des produits



PARTIE 4

REQUÊTES SQL

STRUCTURE DE LA DATABASE



Requête 1

Les 10 pays ayant le plus haut ratio **disponibilité alimentaire/habitant en termes de protéines** (en kg) par habitant, **puis en termes de kcal** par habitant.

```
SELECT pays,
       SUM(dispo_prot_g/1000)*365 AS proteine_hab_kg_an
FROM Dispo_Aliment
GROUP BY pays
ORDER BY proteine_hab_kg_an DESC
LIMIT 10
```

pays	proteine_hab_kg_an
Islande	47.0850
Chine - RAS de Hong-Kong	46.3550
Israël	45.6250
Lituanie	44.1650
Maldives	42.3400
Finlande	41.9750
Portugal	41.2450
Norvège	40.5150
Luxembourg	40.5150
Albanie	39.7850

```
SELECT pays,
       SUM(dispo_alim_kcal) AS kcal_hab_j
FROM Dispo_Aliment
GROUP BY pays
ORDER BY kcal_hab_j desc
LIMIT 10
```

pays	kcal_hab_j
Autriche	3770
Belgique	3737
Turquie	3708
États-Unis d'Amérique	3682
Israël	3610
Irlande	3602
Italie	3578
Luxembourg	3540
Égypte	3518
Allemagne	3503

Requête 2

Pour l'année 2013, les 10 pays ayant le plus faible ratio **disponibilité alimentaire/habitant** en termes de protéines (en kg) par habitant.

```
SELECT pays,  
       SUM(dispo_alim_kcal) AS kcal_hab_j  
FROM Dispo_Alimentaire  
GROUP BY pays  
ORDER BY kcal_hab_j  
LIMIT 10
```

pays	proteine_hab_kg_an
Libéria	12.7750
Mozambique	14.6000
Madagascar	15.6950
Guinée-Bissau	16.0600
République centrafricaine	16.0600
Haïti	16.0600
Zimbabwe	17.8850
Sao Tomé-et-Principe	18.2500
Congo	18.2500
République populaire démocratique de Corée	18.9800

Requête 3

La quantité totale (en kg) de produits perdus par pays en 2013

```
SELECT pays,  
       SUM(pertes*1000000) AS total_pertes_kg  
FROM Equilibre_Prod  
GROUP BY pays  
ORDER BY total_pertes_kg DESC  
LIMIT 15
```

pays	total_pertes_kg
Chine, continentale	89575000000
Brésil	75914000000
Inde	55930000000
Nigéria	19854000000
Indonésie	13081000000
Turquie	12036000000
Mexique	8289000000
Égypte	7608000000
Ghana	7442000000
États-Unis d'Amérique	7162000000
Viet Nam	6743000000
Pakistan	5897000000
Thaïlande	5749000000
Iran (République islamique d')	5450000000
Fédération de Russie	4997000000

Requête 4

Les 10 pays pour lesquels la proportion de personnes sous-alimentées est la plus forte.

```
SELECT Sous_Nutrition.pays,  
       Sous_Nutrition.annee,  
       Sous_Nutrition.nb_personnes / Population.population AS part_ss_nut  
FROM Sous_Nutrition  
LEFT JOIN Population on Sous_Nutrition.code_pays = Population.code_pays  
ORDER BY part_ss_nut DESC  
LIMIT 10
```

	pays	annee	part_ss_nut
	Dominique	2013	0.6944
	Haiti	2013	0.5040
	Kiribati	2013	0.4902
	Zambie	2013	0.4815
	Zimbabwe	2013	0.4664
	Saint-Vincent-et-les Grenadines	2013	0.4587
	République centrafricaine	2013	0.4333
	République populaire démocratique de Corée	2013	0.4258
	Congo	2013	0.4047
	Tchad	2013	0.3821

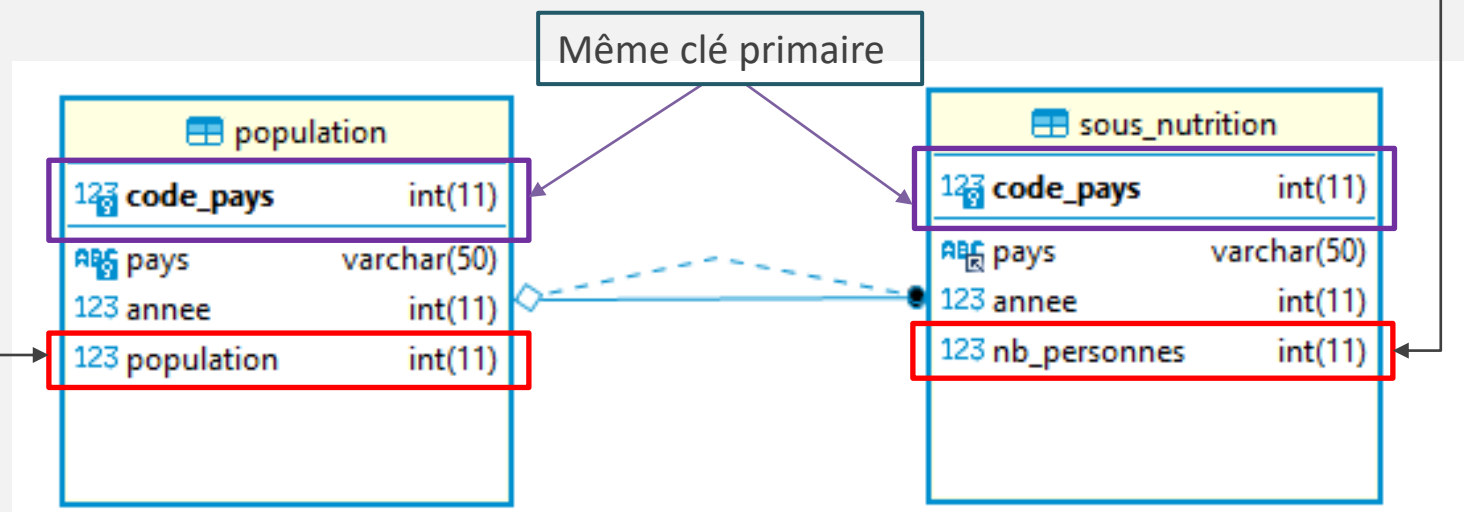
Requête 4

Les 10 pays pour lesquels la proportion de personnes sous-alimentées est la plus forte.

```
SELECT Sous_Nutrition.pays,
       Sous_Nutrition.annee,
       Sous_Nutrition.nb_personnes / Population.population AS part_ss_nut
FROM Sous_Nutrition
LEFT JOIN Population on Sous_Nutrition.code_pays = Population.code_pays
ORDER BY part_ss_nut DESC
LIMIT 10
```

Les données nécessaires pour satisfaire cette requête sont présentes dans 2 tables différentes :

- Attribut population de la table **Population**
- Attribut nb_personnes de la table **Sous_Nutrition**



* La table Population contient tous les pays (174 entrées) alors que Sous_Nutrition ne contient que les pays qui ont une population sous-alimentée (119 lignes)

↳ on va donc réaliser une jointure depuis la table population (donc ici LEFT JOIN) car on veut garder tous les pays

Jointure interne de la relation population et de la relation sous_nutrition selon la condition `sous_nutrition.code_pays = population.code_pays`

Requête 5

Les 10 produits pour lesquels le ratio **Autres utilisations/Disponibilité intérieure** est le plus élevé.

```
SELECT produit,
       avg(coalesce(autres_utilisations, 0) / dispo_int) AS r_autres_dispo_int
FROM Equilibre_Prod
WHERE dispo_int IS NOT NULL
GROUP BY produit
ORDER BY r_autres_dispo_int DESC
LIMIT 10
```

produit	r_autres_dispo_int
Alcool, non Comestible	0.96551724
Plantes Aquatiques	0.92066136
Huile de Palme	0.65252432
Huil Plantes Oleif Autr	0.54369236
Huile de Palmistes	0.53418095
Huile de Colza&Moutarde	0.46264783
Huiles de Poissons	0.40273617
Huile de Coco	0.36329495
Graisses Animales Crue	0.30480915
Manioc	0.23284628

Exemples d'utilisations non alimentaires

Huile de palme :

- Cosmétiques (savons, crèmes du visage, huiles pour le corps...)
- Biocarburant

Huile de coco :

- Cosmétiques (savon en particulier)
- Industrie chimique (détergents, plastiques...)

Requête 5

Les 10 produits pour lesquels le ratio **Autres utilisations/Disponibilité intérieure** est le plus élevé.

```
SELECT produit,
       avg(coalesce(autres_utilisations, 0) / dispo_int) AS r_autres_dispo_int
FROM Equilibre_Prod
WHERE dispo_int IS NOT NULL
GROUP BY produit
ORDER BY r_autres_dispo_int DESC
LIMIT 10
```

On utilise la **fonction d'agrégation AVG()** pour calculer la moyenne

- La division par 0 est impossible
 ↳ on ne peut utiliser AVG que si dispo_int est différent de 0 → `WHERE dispo_int IS NOT NULL`

- La fonction AVG ne s'applique que sur un type numérique non nul
 ↳ or nous avons de nombreuses nulles pour l'attribut autres_utilisations

Si on ne traite pas ces données, la fonction AVG ne prendra en compte que les lignes pour lesquelles autres_utilisations et dispo_int sont non nulles (1 seule ligne dans notre table)

Or ici on veut prendre en compte toutes les lignes avec une dispo_int non nulles

↳ on remplacera alors les valeurs nulles de autres_utilisations par 0

On utilise `coalesce(autres_utilisations, 0)` afin de remplacer les valeurs nulles par 0

ABC produit	ABC pays	123 autres_utilisations	123 dispo_int
Piments	Algérie	[NULL]	20
Piments	Argentine	[NULL]	4
Piments	Australie	[NULL]	3
Piments	Autriche	[NULL]	2
Piments	Bangladesh	[NULL]	124
Piments	Brésil	[NULL]	-1
Piments	Bulgarie	[NULL]	3

MERCI POUR VOTRE ATTENTION

ETUDE DE SANTE PUBLIQUE

La sous-nutrition mondiale

Frédéric Gainza - Janvier 2021