LOADING DATASET FOR CLEANING AND ANALYSIS

```
import pandas as pd
import csv
```

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("openfoodfacts/world-food-facts")
```

```
Using Colab cache for faster access to the 'world-food-facts' dataset.
```

```
import os
file_list = os.listdir(path)
```

```
import pandas as pd


file_path = os.path.join(path, file_list[0])

# Load the TSV file into a pandas DataFrame
data = pd.read_csv(file_path, sep='\t')
```

```
/tmp/ipython-input-961860644.py:7: DtypeWarning: Columns (0,3,5,19,20,24,25,26,27,2
  data = pd.read_csv(file_path, sep='\t')
```

```
data.shape
```

```
(356027, 163)
```

```
# Dropping columns that are entirely empty
data.dropna(axis=1, how='all', inplace=True)

# Calculating the percentage of missing values for each column
missing_percentages = data.isnull().sum() / len(data) * 100
```

```
data.shape
```

```
(356027, 147)
```

```
categories_tags = data['categories_tags']
categories_tags.isnull().sum()
```

```
np.int64(252752)
```

USER STORY 1:DATA INGESTION & CLEAN UP

```
### Taking the columns needed For Analysis
needed = [
    "product_name",
    "categories_tags",
    "energy_100g",
    "fat_100g",
    "sugars_100g",
    "fiber_100g",
    "proteins_100g",
    "ingredients_text"
]

[col for col in needed if col in data.columns]
```

```
['product_name',
 'categories_tags',
 'energy_100g',
 'fat_100g',
 'sugars_100g',
 'fiber_100g',
 'proteins_100g',
 'ingredients_text']
```

```
###Filtering Out the Data That Will Be Used For Analysis
data_needed = data[needed].copy()
data_needed.shape
```

```
(356027, 8)
```

```
data_user1 = data_needed.dropna(
    subset=["product_name", "sugars_100g", "proteins_100g"]
)
data_user1.shape
```

```
(275113, 8)
```

```
### Removing biologically impossible data
data_user1 = data_user1[
    (data_user1["sugars_100g"].between(0, 100)) &
    (data_user1["proteins_100g"].between(0, 100))
]
```

```
data_user1.shape
```

```
(275095, 8)
```

STORY 2: The Category Wrangler

```
###Checking contents of the category_tag
data_needed["categories_tags"].isnull().sum()
```

```
np.int64(252752)
```

```
data_needed["categories_tags"]
```

	categories_tags
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
...	...
356022	NaN
356023	en:salty-snacks,en:appetizers,en:chips-and-fri...
356024	NaN
356025	NaN
356026	NaN

356027 rows × 1 columns

dtype: object

```
##Removing Rows with Nans or missing values
data_user2 = data_needed.dropna(subset=['categories_tags'])
data_user2
```

	product_name	categories_tags	energy_100g	fat_100g	sugars_1
47	Filet de bœuf	fr:filet-de-boeuf	NaN	NaN	N
176	Salade Cesar	en:plant-based-foods-and-beverages,en:plant-ba...	1210.0	12.0	
177	Danoises à la cannelle roulées	en:sugary-snacks,en:biscuits-and-cakes,en:past...	1520.0	14.4	2
179	Flute	en:plant-based-foods-and-beverages,en:plant-ba...	NaN	NaN	N
182	Chaussons tressés aux pommes	en:sugary-snacks,en:biscuits-and-cakes,en:past...	1090.0	10.7	2
...	...	...	...	...	
356016	Szprot w oleju roslinnym	pl:szprot	NaN	NaN	N
356017	Thé vert Earl grey	en:plant-based-foods-and-beverages,en:beverage...	21.0	0.2	
356019	Rillette d'oie	en:meats,en:spreads,en:prepared-meats,en:salte...	NaN	NaN	N
356020	NaN	en:plant-based-foods-and-beverages,en:plant-ba...	NaN	NaN	N
356023	乐吧泡菜味薯片	en:salty-snacks,en:appetizers,en:chips-and-fri...	NaN	NaN	N

103275 rows × 8 columns

```
def parse_tags(tags):
    if pd.isna(tags):
        return []
    return [t.replace("en:", "").strip() for t in tags.split(",")]
```

```
### Creating the primary categories to be assigned to products
category_map = {
    "Sweet Snacks": [
        "sweet-snacks", "chocolate", "biscuits", "cookies", "cakes", "candies", "d
    ],
    "Savory Snacks": [
        "salty-snacks", "crisps", "chips", "popcorn", "nuts", "pretzels"
```

```
        ],
        "Dairy & Protein": [
            "dairy", "cheese", "yogurt", "milk", "protein", "whey", "meat" ,"poultries
        ],
        "Fruits & Plant-Based": [
            "fruits", "vegetables", "plant-based","plant-based-foods" ,"legumes"
        ],
        "Beverages": [
            "beverages", "drinks", "juice", "tea", "coffee"
        ]
    }
```

```
## Creating the Logic For Assigning a Product To a category
def assign_primary_category(tags):
    parsed_tags = parse_tags(tags)

    for category, keywords in category_map.items():
        if any(keyword in tag for tag in parsed_tags for keyword in keywords):
            return category

    return "Other"
```

```
###Applying Logic to Datasets
data_needed["primary_category"] = data_needed["categories_tags"].apply(assign_prim
data_user2["primary_category"] = data_user2["categories_tags"].apply(assign_primar
```

```
/tmp/ipython-input-217120248.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
  data_user2["primary_category"] = data_user2["categories_tags"].apply(assign_prima
```

```
data_user2["primary_category"]
```

	primary_category
47	Other
176	Fruits & Plant-Based
177	Sweet Snacks
179	Fruits & Plant-Based
182	Sweet Snacks
...	...
356016	Other
356017	Fruits & Plant-Based
356019	Dairy & Protein
356020	Savory Snacks
356023	Savory Snacks

103275 rows × 1 columns

dtype: object

```
###Confirming if categories exists in all records
data_user2["primary_category"].value_counts()
```

primary_category	count
Fruits & Plant-Based	27132
Other	22045
Dairy & Protein	21786
Sweet Snacks	16497
Beverages	11179
Savory Snacks	4636

dtype: int64

```
###Sanity check of category assigment
check_data_user2 = data_user2[
    ["product_name", "categories_tags", "primary_category"]
]
check_data_user2.head(10)
```

	product_name	categories_tags	primary_category
47	Filet de bœuf	fr:filet-de-boeuf	Other
176	Salade Cesar	en:plant-based-foods-and-beverages,en:plant-ba...	Fruits & Plant-Based
177	Danoises à la cannelle roulées	en:sugary-snacks,en:biscuits-and-cakes,en:past...	Sweet Snacks
179	Flute	en:plant-based-foods-and-beverages,en:plant-ba...	Fruits & Plant-Based
182	Chaussons tressés aux pommes	en:sugary-snacks,en:biscuits-and-cakes,en:past...	Sweet Snacks
183	Pain Burger Artisan	fr:boulange	Other
184	lentilles vertes	en:plant-based-foods-and-beverages,en:plant-ba...	Fruits & Plant-Based
185	Root Beer	en:beverages,en:carbonated-drinks,en:sodas,en:...	Beverages
186	Biscuits sablés fourrage au cacao	en:sugary-snacks,en:biscuits-and-cakes,en:bisc...	Sweet Snacks
187	Quiche Lorraine	en:meals,en:pizzas-pies-and-quiches,en:quiches...	Other

```
### Selecting the necessary columns for the rest of user story for analysis

dash_df = data_user2[
    ["product_name",
     "primary_category",
     "proteins_100g",
     "sugars_100g",
     "fiber_100g",
     "fat_100g",
     "energy_100g",
     "ingredients_text"]
]
dash_df
```

	product_name	primary_category	proteins_100g	sugars_100g	fiber_100g	fat
47	Filet de bœuf	Other	NaN	NaN	NaN	
176	Salade Cesar	Fruits & Plant-Based	22.00	0.0	2.00	
177	Danoises à la cannelle roulées	Sweet Snacks	4.79	28.1	2.05	
179	Flute	Fruits & Plant-Based	NaN	NaN	NaN	
182	Chaussons tressés aux pommes	Sweet Snacks	3.33	24.7	2.00	
...	...	...	...	...	...	
356016	Szprot w oleju roslinnym	Other	NaN	NaN	NaN	
356017	Thé vert Earl grey	Fruits & Plant-Based	0.50	0.5	0.20	
356019	Rillette d'oie	Dairy & Protein	NaN	NaN	NaN	
356020	NaN	Savory Snacks	NaN	NaN	NaN	
356023	乐吧泡菜味薯片	Savory Snacks	NaN	NaN	NaN	

103275 rows × 8 columns

```
###Removing Missing Values In Required Columns
dash_df = dash_df.dropna(
    subset=["proteins_100g", "sugars_100g", "primary_category"]
)
dash_df.isnull().sum()
```

	0
product_name	461
primary_category	0
proteins_100g	0
sugars_100g	0
fiber_100g	28435
fat_100g	3198
energy_100g	71
ingredients_text	7566

dtype: int64

```
dash_df.columns
```

```
Index(['product_name', 'primary_category', 'proteins_100g', 'sugars_100g',
       'fiber_100g', 'fat_100g', 'energy_100g', 'ingredients_text'],
      dtype='object')
```

USER STORY 3:The Nutrient Matrix

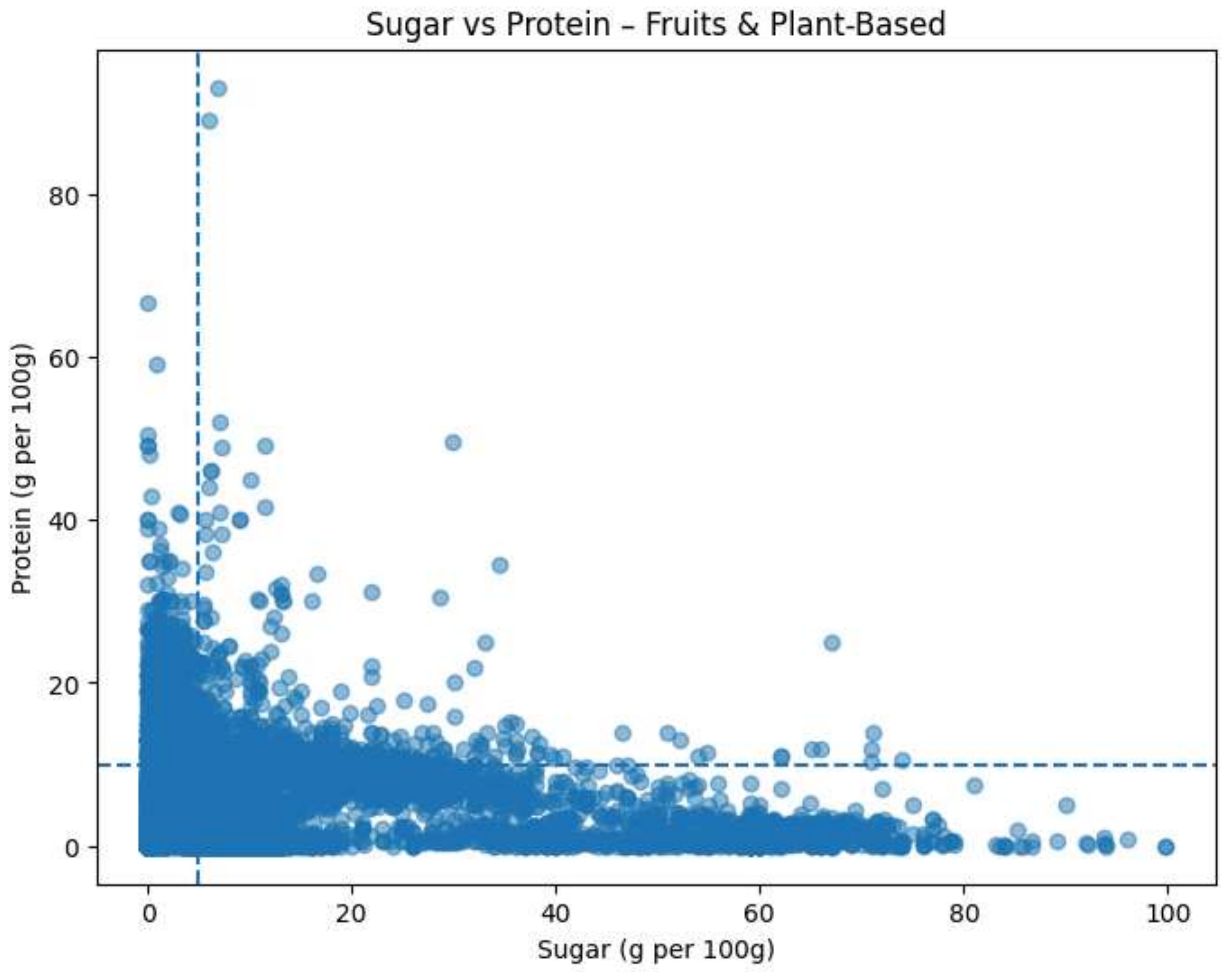
```
###Creating the Scatter Plot
import matplotlib.pyplot as plt
import seaborn as sns
category = "Fruits & Plant-Based"

subset = dash_df[dash_df["primary_category"] == category]

plt.figure(figsize=(8, 6))
plt.scatter(
    subset["sugars_100g"],
    subset["proteins_100g"],
    alpha=0.5
)

plt.axvline(x=5, linestyle="--")
plt.axhline(y=10, linestyle="--")

plt.title(f"Sugar vs Protein – {category}")
plt.xlabel("Sugar (g per 100g)")
plt.ylabel("Protein (g per 100g)")
plt.show()
```




```
#Saving the dash_df DataFrame to a CSV file
dash_df.to_csv('dash_df.csv', index=False)
```

```
##USER 5 Finding high proteing ingredients
high_protein_df = dash_df[dash_df["proteins_100g"] >= 20]

high_protein_df["ingredients_text"] = (
    high_protein_df["ingredients_text"]
    .str.lower()
)

high_protein_df
```

```
/tmp/ipython-input-2933130306.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
  high_protein_df["ingredients_text"] = (
```

	product_name	primary_category	proteins_100g	sugars_100g	fiber_100g	fat
176	Salade Cesar	Fruits & Plant-Based	22.00	0.0	2.0	
307	Whey Protein aus Molke Vanilla	Dairy & Protein	78.05	6.0	0.1	
309	Whey Protein aus Molke 1000 Gramm Vanilla	Dairy & Protein	78.05	6.0	0.1	
312	Whey Protein aus Molke 500 Gramm Vanilla	Dairy & Protein	78.05	6.0	0.1	
547	Honey roast ham	Other	24.50	2.9	0.1	
...	...	...	...	...	...	
355506	Edam Big Block Cheese	Dairy & Protein	26.40	0.0	NaN	
355528	Meadow Fresh	Dairy & Protein	24.00	39.0	NaN	
355538	Crunchy Peanut Butter	Savory Snacks	27.20	5.5	NaN	
355568	Original Beef Jerky	Savory Snacks	44.00	20.0	NaN	