Data Wrangling: CarbCon Iteration 1

Date: 21/08/2020

Environment: Python 3.7.3 and Jupyter 5.7.8

1. Introduction & Overview

This report focuses on the wrangling process of the following open data as a core dataset for CarbCon: Food Carbon Footprint Calculator:

- Food Product Greenhouse Gas emissions Data: Data contains Green House Gas emissions from per kg of food product based on food counter, food type, region, and year of study. Provides information regarding the number of kg CO2eq from food product to develop calculating and monitoring food's carbon footprint function
- Open Food Facts Data: Label Carbon Footprint: Contains Greenhouse Gas Emissions indicator data for each product as sold for 100g. Data utilisiation to complement the information regarding the number of greenhouse gas emission from the data source 1 in order to develop calculating and monitoring food's carbon footprint function

2. Importing Libraries

Importing initial libraries

In [5]:
```
import pandas as pd
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import numpy as np
import matplotlib.pyplot as plt #visualisation
%matplotlib inline
```

3. Exploration First Data

In [6]:
```
# read the files with pandas
foodDF_source = pd.read_excel('foodGHG.xlsx')
```

In [7]:
```
# display some of the data
foodDF_source.head()
```

Out[7]:

	Food counter	Food type	Sub-category	Region	Year of study	Report type	kg CO2-eq/kg produce, BFM or L after conversion	Notes (conventional farming assumed unless stated)	Reference	reference number
0	Dairy Counter	Butter	Butter	United Kingdom	2010.0	Journal	3.7	spreadable	Nilsson, K., A. Flysjö, J. Davis, S. Sim, N. U...	1
1	Dairy Counter	Butter	Butter	France	2010.0	Journal	7.2	NaN	Nilsson, K., A. Flysjö, J. Davis, S. Sim, N. U...	2
2	Dairy Counter	Butter	Butter	Canada	2006.0	Journal	7.3	NaN	Vergé, X. P. C., D. Maxime, J. A. Dyer, R. L. ...	3
3	Dairy Counter	Butter	Butter	Germany	2010.0	Journal	9	NaN	Nilsson, K., A. Flysjö, J. Davis, S. Sim, N. U...	4
4	Dairy Counter	Butter	Butter	United Kingdom	2012.0	EPD	9.5	NaN	Tesco (2012). "Product carbon footprint summar...	5

In [8]:
```
foodDF_source.tail()
```

Out[8]:

	Food counter	Food type	Sub-category	Region	Year of study	Report type	kg CO2-eq/kg produce, BFM or L after conversion	Notes (conventional farming assumed unless stated)	Reference	reference number
1726	Flours, Grains, Pulses and Nuts	Wheat	Cereal	Spain	2010.0	Report	0.76	NaN	Nemecek, T. (2010). How to establish life cycl...	1727
1727	Flours, Grains, Pulses and Nuts	Wheat	Cereal	Norway	2012.0	Conference	0.78	winter	Roer, A. G., A. Korsaeth, A. Johansen, A. K. B...	1728
1728	Flours, Grains, Pulses and Nuts	Wheat	Cereal	USA	2011.0	Journal	0.8	NaN	González, A. D., B. Frostell and A. Carlsson-K...	1729
1729	Flours, Grains, Pulses and Nuts	Wheat	Cereal	United Kingdom	2010.0	Journal	0.8	organic	Williams, A., E. Audsley and D. Sandars (2010)...	1730
1730	Flours, Grains, Pulses and Nuts	Wheat	Cereal	World	2008.0	Journal	1.1	NaN	Michaelowa, A. and B. Dransfeld (2008). "Green...	1731

In [9]:
```
# check the data types
foodDF_source.dtypes
```

Out[9]:
```
Food counter                                          object
Food type                                             object
Sub-category                                          object
Region                                                object
Year of study                                        float64
Report type                                           object
kg CO2-eq/kg produce, BFM or L after conversion       object
Notes (conventional farming assumed unless stated)    object
Reference                                             object
reference number                                       int64
dtype: object
```

4. Data Wrangling

In [10]:
```
# Drop unused columns
foodDF = foodDF_source.drop([
                      'Reference',
                      'reference number',
                      'Report type',], axis=1)
foodDF.head(5)
```

Out[10]:

	Food counter	Food type	Sub-category	Region	Year of study	kg CO2-eq/kg produce, BFM or L after conversion	Notes (conventional farming assumed unless stated)
0	Dairy Counter	Butter	Butter	United Kingdom	2010.0	3.7	spreadable
1	Dairy Counter	Butter	Butter	France	2010.0	7.2	NaN
2	Dairy Counter	Butter	Butter	Canada	2006.0	7.3	NaN
3	Dairy Counter	Butter	Butter	Germany	2010.0	9	NaN
4	Dairy Counter	Butter	Butter	United Kingdom	2012.0	9.5	NaN

In [11]:
```
# Rename column
foodDF = foodDF.rename(columns={'Food counter': 'Food Counter',
                                'Food type': 'Food Item',
                                'Sub-category': 'Category',
                                'Region': 'Region',
                                'Year of study': 'Year',
                                'kg CO2-eq/kg produce, BFM or L after conversion': 'Carbon Footprint (kg CO2-eq/kg)',
                                'Notes (conventional farming assumed unless stated)':'Farming'})

foodDF.head(5)
```

Out[11]:

	Food counter	Food Item	Category	Region	Year	Carbon Footprint (kg CO2-eq/kg)	Farming
0	Dairy Counter	Butter	Butter	United Kingdom	2010.0	3.7	spreadable
1	Dairy Counter	Butter	Butter	France	2010.0	7.2	NaN
2	Dairy Counter	Butter	Butter	Canada	2006.0	7.3	NaN
3	Dairy Counter	Butter	Butter	Germany	2010.0	9	NaN
4	Dairy Counter	Butter	Butter	United Kingdom	2012.0	9.5	NaN

4.1 Handling Null Value

Imputation will be used to handle null value that are not related to the carbon footprint value.

In [12]:
```
# check the null value
print(foodDF.isnull().sum())
```

```
Food counter                        0
Food Item                           0
Category                            3
Region                              4
Year                                4
Carbon Footprint (kg CO2-eq/kg)    56
Farming                           799
dtype: int64
```

In [13]:
```
# Impute conventional farming remark
foodDF['Farming']=foodDF['Farming'].fillna('conventional farming')
```

In [14]:
```
foodDF.head()
```

Out[14]:

	Food counter	Food Item	Category	Region	Year	Carbon Footprint (kg CO2-eq/kg)	Farming
0	Dairy Counter	Butter	Butter	United Kingdom	2010.0	3.7	spreadable
1	Dairy Counter	Butter	Butter	France	2010.0	7.2	conventional farming
2	Dairy Counter	Butter	Butter	Canada	2006.0	7.3	conventional farming
3	Dairy Counter	Butter	Butter	Germany	2010.0	9	conventional farming
4	Dairy Counter	Butter	Butter	United Kingdom	2012.0	9.5	conventional farming

In [15]:
```
# check the number of row and columns
foodDF.shape
```

Out[15]:
```
(1731, 7)
```

In [16]:
```
# drop null value in carbon footprint column
foodDF = foodDF.dropna(subset=['Carbon Footprint (kg CO2-eq/kg)'])
```

In [17]:
```
print(foodDF.isnull().sum())
```

```
Food counter                      0
Food Item                         0
Category                          3
Region                            4
Year                              3
Carbon Footprint (kg CO2-eq/kg)   0
Farming                           0
dtype: int64
```

In [18]:
```
foodDF[foodDF['Category'].isnull()]
```

Out[18]:

	Food counter	Food Item	Category	Region	Year	Carbon Footprint (kg CO2-eq/kg)	Farming
646	Fruit and Vegetable Counter	Melons (G)	NaN	Italy	2012.0	1.43	greenhouse, 'pavilion' tent covered in LDPE,...
647	Fruit and Vegetable Counter	Melons (G)	NaN	Italy	2012.0	1.43	greenhouse, shared rotation, no auxiliary heat...
648	Fruit and Vegetable Counter	Melons (G)	NaN	Italy	2012.0	1.24	greenhouse, tunnel greenhouse covered in LDPE,...

In [19]:
```
# Impute category
foodDF['Category']=foodDF['Category'].fillna('Fruit')
```

In [20]:
```
foodDF[foodDF['Region'].isnull()]
```

Out[20]:

	Food counter	Food Item	Category	Region	Year	Carbon Footprint (kg CO2-eq/kg)	Farming
442	Fruit and Vegetable Counter	Broccoli	Brassica	NaN	2014.0	0.66	conventional farming
1297	Meat Counter	Mussels	Shelfish	NaN	2010.0	9.5122	frozen @ POS
1298	Meat Counter	Mussels	Shelfish	NaN	2010.0	9.8379	canned @ POS
1299	Meat Counter	Mussels	Shelfish	NaN	2010.0	13.9017	fresh @ POS

In [21]:
```
foodDF[foodDF['Year'].isnull()]
```

Out[21]:

	Food counter	Food Item	Category	Region	Year	Carbon Footprint (kg CO2-eq/kg)	Farming
19	Dairy Counter	Cheese	Cheese	USA	NaN	8.377	conventional farming
31	Dairy Counter	Cheese	Cheese	Sweden	NaN	12.05	conventional farming
1490	Meat Counter	Trout	Fish Counter	Finland	NaN	2.443	farmed ungutted rainbow trout finland at super...

In [22]:
```
# Impute Region
foodDF['Region']=foodDF['Region'].fillna('Unknown')
```

4.2 Clean data by column

In [23]:
```
#List unique values in the df['name'] column
foodDF['Food Item'].unique()
```

Out[23]:
```
array(['Butter', 'Camembert Cheese ', 'Cheddar Cheese', 'Cheese',
       'Cheese ', 'Goats Cheese', 'Mozarella Cranolo ', 'Mozarrella',
       'Natural', 'Semi Hard Chees', 'Semi-Hard', 'Cream', 'Almond Milk',
       'Bufalo Milk', 'Coconut-Milk', 'Cows Milk', 'Goats Milk',
       'Soy-Milk', 'Yogurt', 'Apples ', 'Apples and Pears', 'Apricots ',
       'Artichokes ', 'Asparagus ', 'Avocados ', 'Bananas ', 'Beetroot ',
       'Broccoli ', 'Cabbage', 'Cabbages, Other Brassicas ',
       'Capsicums/Peppers ', 'Carrots', 'Carrots and Turnips ',
       'Cauliflowers ', 'Celery ', 'Cherries ', 'Chillies ',
       'Citrus Fruit, Misc. ', 'Citrus Small', 'Coconuts (Incl. Copra) ',
       'Cranberries, Blueberries ', 'Cucumbers ', 'Cucumbers (G)',
       'Cucumbers and Gherkins (G)', 'Currants and Gooseberries',
       'Dates ', 'Eggplants (Aubergines) ', 'Fennel', 'Figs ', 'Garlic ',
       'Gherkins', 'Gherkins (G)', 'Ginger ', 'Grapefruit and Pomelo ',
       'Grapes', 'Guavas', 'Kiwi Fruit ', 'Lemons and Limes ', 'Lettuce',
       'Lettuce ', 'Lettuce (G)', 'Manderin', 'Melons', 'Mushrooms',
       'Olives ', 'Onion', 'Oranges ', 'Peach', 'Peaches and Nectarines ',
       'Pears', 'Pears and Apples', 'Pears and Quinces',
       'Peppers/Capsicums', 'Peppers/Capsicums (G)', 'Melons (G)',
       'Pineapples ', 'Plums and Sloes ', 'Potatoes ', 'Pumpkins ',
       'Quinces and Pears  ', 'Quinces and Pears ',
       'Raspberries and Other Berries ',
       'Raspberries and Other Berries (G)', 'Rockmelon / Cantelope',
       'Spinach', 'Starchy Root', 'Strawberries', 'Strawberries (G)',
       'Swedes (Rutabage)', 'Tangerines, Mandarins Etc.', 'Tomatoes',
       'Tomatoes (G)', 'Watermelon', 'Zucchini/Button Squash ',
       'Zucchini/Button Squash (G)', 'Alfonsino', 'Anglerfish', 'Bass',
       'Beef', 'Bigeye Tuna', 'Buffalo', 'Carp', 'Catfish',
       'Cephalopodsm Varied (Squid)', 'Chicken', 'Cod',
       'Cod (Fish Stick)', 'Cuttlefish', 'Diamond Fish', 'Duck', 'Eel',
       'Eggs', 'Emu', 'Fish', 'Fish (Mixed)', 'Flatfish',
       'Flatfish Varied ', 'Fork Beard', 'Haddock', 'Hake',
       'Hake (Fish Stick)', 'Hake European ', 'Hake Fillet',
       'Hake Senegal ', 'Herring', 'Herring ', 'Kangaroo', 'Lamb', 'Ling',
       'Ling Common ', 'Lobster', 'Mackeral', 'Mackeral (Fish Stick',
       'Mackerel Atlantic ', 'Mackerel Horse ', 'Megrim', 'Mussels',
       'Octopus', 'Pilchard', 'Pollock', 'Pollock (Fish Stick)',
       'Pomfret', 'Pomfret Atlantic ', 'Porbeagle', 'Pork',
       'Prawns/Shrimp', 'Rabbit', 'Rhombus', 'Rock Fish', 'Salmon',
       'Sea Bass', 'Shark', 'Shark Mako ', 'Sole', 'Squid', 'Swordfish',
       'Trout', 'Tuna', 'Turbot', 'Turkey', 'Veal', 'Whiting',
       'Whiting Blue ', 'Almonds', 'Barley ', 'Beans',
       'Beans - Gigante/Butter', 'Beans - Green', 'Beans - Green Beans',
       'Beans - Green Beans (Phaseolus Vulgaris L.)',
       'Beans - Pinto  Usa Dried', 'Beans - Plake',
       'Beans – French and Runner', 'Cahsew Nuts', 'Cereals Misc. ',
       'Chestnuts', 'Chick Peas', 'Chick Peas ', 'Cowpeas',
       'Graham Flour', 'Ground Nuts', 'Hazlenuts', 'Lentils', 'Maize',
       'Maize Sweet Corn ', 'Millet ', 'Nuts Misc. ', 'Oat', 'Oat Berry',
       'Oatmeal', 'Oatmeal ', 'Oats ',
       'Palm Nuts-Kernels (Nut Equiv.)/Oil ', 'Peanuts', 'Peas',
       'Peas - Dry', 'Peas - Green ', 'Peas - Green – Shelled',
       'Peas - Yellow Dried', 'Pistachios', 'Quinoa', 'Rape Seed',
       'Rapeseed and Mustard Seed ', 'Rice', 'Rye ', 'Sesame Seed ',
       'Sorghum ', 'Soybean', 'Sunflower Seed ', 'Walnuts ', 'Wheat'],
      dtype=object)
```

In [24]:
```
#List unique values in the df['name'] column
foodDF['Region'].unique()
```

Out[24]:
```
array(['United Kingdom', 'France', 'Canada', 'Germany', 'Austria', 'USA',
       'Netherlands', 'World', 'Portugal', 'Australia', 'Norway', 'Spain',
       'New Zealand', 'Sweden', 'Bolonga', 'swiss', 'Italy', 'portugal',
       'Ireland', 'Denmark', 'Africa South ', 'Switzerland', 'Chile',
       'North America', 'Argentina', 'Finland', 'Belgium', 'Slovakia',
       'China', 'Luxembourg', 'Brazil', 'slovenia', 'Oceania',
       'Europe Eastern', 'Europe western', 'Bulgaria', 'netherland',
       'Czech Republic', 'lithuania', 'Russia', 'Perù', 'world',
       'finlsnd', 'Czech republic', 'greece', 'hungary', 'Mexico',
       'romania', 'Turkey', 'lativia', 'Poland', 'India', 'Estonia',
       'Morocco', 'East Asia', 'Belarus', 'bulgaria', 'Uganda',
       'Pakistan', 'Bangladesh', 'Central and South America',
       'North Africa', ' Asia South', 'Africa Sub Saharan ',
       'European Union', 'Japan', 'France southern', 'France northen',
       'Eurpoe', 'switzerland', 'denmark', 'Germany ', 'slow', 'peru',
       'Peru', 'Maldives', 'Ecuador', 'EU imported', 'Netherands',
       'Cavendish', 'Unknown', 'imported', 'Greece', 'Sicily', 'Holland',
       'Case study Spain', 'Spain (Valencia)', 'Brazil*', 'USA (Florida)',
       'Costa Rica', 'Ghana', 'iran', 'Jaoan', 'Phillipines', 'morroco',
       'Thailand', 'Malaysia', 'Indonesia', 'Bangledesh', 'maldives',
       'Madagascar', 'Somalia', 'Australia (NSW)', 'E Europe',
       'Germany, Poland and Denmark', 'EU 27', 'EU', 'EU Netherlands',
       'Other imported', 'Usa', 'Canada W. ', 'USA Mid-West', 'W  Europe',
       'Uruguay', 'Canada west', 'USA Mid-West ', 'Canada E', 'Oceana',
       'Columbia', 'Africa (near east and north)', 'Nth America',
       'Venizuala', 'Asia (East and South East)',
       'Sth America (latin America and caribean)', 'Africa (subsaharan)',
       'Asia (south)', 'Sth America (Latin America and Caribean)',
       'Vietnam', 'Australia SA', 'Canada (east)', 'brazil',
       'Canada (west)', 'Import', 'Denmark ', 'Reunion Island',
       'United Kingdom ', 'USA ', 'Island', 'Háskóli Islands', 'danish',
       'USA Iowa', 'scottish', 'Norwiegen', 'France ', 'corsica',
       'Canada east', 'US ', 'Corsica', 'Australia NT',
       'artisanal to market', 'Senegal', 'vietnam', 'Turnisia', 'trout',
       'Indian Ocean', 'Atlantic Ocean', 'Pacific Ocean', 'World ',
       'Swedne', 'USA (california)', 'Romania'], dtype=object)
```

In [25]:
```
#List unique values in the df['name'] column
foodDF['Year'].unique()
```

Out[25]:
```
array([2010.  , 2006.  , 2012.  , 2008.  , 2013.  ,     nan, 2005.  ,
       2007.  , 2002.  , 2011.  , 2014.  , 2004.  , 2003.  , 2009.  ,
       2000.  , 2001.  , 2015.  , 1998.  ,    3.79,  209.  ,    2.12,
         71.  , 2016.  , 2017.  ])
```

In [26]:
```
foodDF.shape
```

Out[26]:
```
(1675, 7)
```

In [27]:
```
# replace the year that are less than 1000 to unknown
foodDF['Year'].mask(foodDF['Year'] < 1000, 'Unknown', inplace=True)
```

In [28]:
```
#List unique values in the df['name'] column
foodDF['Year'].unique()
```

Out[28]:
```
array([2010.0, 2006.0, 2012.0, 2008.0, 2013.0, nan, 2005.0, 2007.0,
       2002.0, 2011.0, 2014.0, 2004.0, 2003.0, 2009.0, 2000.0, 2001.0,
       2015.0, 1998.0, 'Unknown', 2016.0, 2017.0], dtype=object)
```

In [29]:
```
# Impute Year
foodDF['Year']=foodDF['Year'].fillna('Unknown')
```

In [30]:
```
print(foodDF.isnull().sum())
```

```
Food counter                      0
Food Item                         0
Category                          0
Region                            0
Year                              0
Carbon Footprint (kg CO2-eq/kg)   0
Farming                           0
dtype: int64
```

4.3 Food Item Wrangling

In [31]:
```
foodDF.head()
```

Out[31]:

	Food counter	Food Item	Category	Region	Year	Carbon Footprint (kg CO2-eq/kg)	Farming
0	Dairy Counter	Butter	Butter	United Kingdom	2010	3.7	spreadable
1	Dairy Counter	Butter	Butter	France	2010	7.2	conventional farming
2	Dairy Counter	Butter	Butter	Canada	2006	7.3	conventional farming
3	Dairy Counter	Butter	Butter	Germany	2010	9	conventional farming
4	Dairy Counter	Butter	Butter	United Kingdom	2012	9.5	conventional farming

In [32]:
```
foodDF['Food Item'].unique()
```

Out[32]:
```
array(['Butter', 'Camembert Cheese ', 'Cheddar Cheese', 'Cheese',
       'Cheese ', 'Goats Cheese', 'Mozarella Cranolo ', 'Mozarrella',
       'Natural', 'Semi Hard Chees', 'Semi-Hard', 'Cream', 'Almond Milk',
       'Bufalo Milk', 'Coconut-Milk', 'Cows Milk', 'Goats Milk',
       'Soy-Milk', 'Yogurt', 'Apples ', 'Apples and Pears', 'Apricots ',
       'Artichokes ', 'Asparagus ', 'Avocados ', 'Bananas ', 'Beetroot ',
       'Broccoli ', 'Cabbage', 'Cabbages, Other Brassicas ',
       'Capsicums/Peppers ', 'Carrots', 'Carrots and Turnips ',
       'Cauliflowers ', 'Celery ', 'Cherries ', 'Chillies ',
       'Citrus Fruit, Misc. ', 'Citrus Small', 'Coconuts (Incl. Copra) ',
       'Cranberries, Blueberries ', 'Cucumbers ', 'Cucumbers (G)',
       'Cucumbers and Gherkins (G)', 'Currants and Gooseberries',
       'Dates ', 'Eggplants (Aubergines) ', 'Fennel', 'Figs ', 'Garlic ',
       'Gherkins', 'Gherkins (G)', 'Ginger ', 'Grapefruit and Pomelo ',
       'Grapes', 'Guavas', 'Kiwi Fruit ', 'Lemons and Limes ', 'Lettuce',
       'Lettuce ', 'Lettuce (G)', 'Manderin', 'Melons', 'Mushrooms',
       'Olives ', 'Onion', 'Oranges ', 'Peach', 'Peaches and Nectarines ',
       'Pears', 'Pears and Apples', 'Pears and Quinces',
       'Peppers/Capsicums', 'Peppers/Capsicums (G)', 'Melons (G)',
       'Pineapples ', 'Plums and Sloes ', 'Potatoes ', 'Pumpkins ',
       'Quinces and Pears  ', 'Quinces and Pears ',
       'Raspberries and Other Berries ',
       'Raspberries and Other Berries (G)', 'Rockmelon / Cantelope',
       'Spinach', 'Starchy Root', 'Strawberries', 'Strawberries (G)',
       'Swedes (Rutabage)', 'Tangerines, Mandarins Etc.', 'Tomatoes',
       'Tomatoes (G)', 'Watermelon', 'Zucchini/Button Squash ',
       'Zucchini/Button Squash (G)', 'Alfonsino', 'Anglerfish', 'Bass',
       'Beef', 'Bigeye Tuna', 'Buffalo', 'Carp', 'Catfish',
       'Cephalopodsm Varied (Squid)', 'Chicken', 'Cod',
       'Cod (Fish Stick)', 'Cuttlefish', 'Diamond Fish', 'Duck', 'Eel',
       'Eggs', 'Emu', 'Fish', 'Fish (Mixed)', 'Flatfish',
       'Flatfish Varied ', 'Fork Beard', 'Haddock', 'Hake',
       'Hake (Fish Stick)', 'Hake European ', 'Hake Fillet',
       'Hake Senegal ', 'Herring', 'Herring ', 'Kangaroo', 'Lamb', 'Ling',
       'Ling Common ', 'Lobster', 'Mackeral', 'Mackeral (Fish Stick',
       'Mackerel Atlantic ', 'Mackerel Horse ', 'Megrim', 'Mussels',
       'Octopus', 'Pilchard', 'Pollock', 'Pollock (Fish Stick)',
       'Pomfret', 'Pomfret Atlantic ', 'Porbeagle', 'Pork',
       'Prawns/Shrimp', 'Rabbit', 'Rhombus', 'Rock Fish', 'Salmon',
       'Sea Bass', 'Shark', 'Shark Mako ', 'Sole', 'Squid', 'Swordfish',
       'Trout', 'Tuna', 'Turbot', 'Turkey', 'Veal', 'Whiting',
       'Whiting Blue ', 'Almonds', 'Barley ', 'Beans',
       'Beans - Gigante/Butter', 'Beans - Green', 'Beans - Green Beans',
       'Beans - Green Beans (Phaseolus Vulgaris L.)',
       'Beans - Pinto  Usa Dried', 'Beans - Plake',
       'Beans – French and Runner', 'Cahsew Nuts', 'Cereals Misc. ',
       'Chestnuts', 'Chick Peas', 'Chick Peas ', 'Cowpeas',
       'Graham Flour', 'Ground Nuts', 'Hazlenuts', 'Lentils', 'Maize',
       'Maize Sweet Corn ', 'Millet ', 'Nuts Misc. ', 'Oat', 'Oat Berry',
       'Oatmeal', 'Oatmeal ', 'Oats ',
       'Palm Nuts-Kernels (Nut Equiv.)/Oil ', 'Peanuts', 'Peas',
       'Peas - Dry', 'Peas - Green ', 'Peas - Green – Shelled',
       'Peas - Yellow Dried', 'Pistachios', 'Quinoa', 'Rape Seed',
       'Rapeseed and Mustard Seed ', 'Rice', 'Rye ', 'Sesame Seed ',
       'Sorghum ', 'Soybean', 'Sunflower Seed ', 'Walnuts ', 'Wheat'],
      dtype=object)
```

In [33]:
```
foodDF['Category'].unique()
```

Out[33]:
```
array(['Butter ', 'Cheese', 'Cream', 'Milk', 'Yogurt', 'Pome', 'Drupe',
       'Tubers', 'Stem Shoots', 'True Berry', 'Musa', 'Roots', 'Brassica',
       'Stems Of Leaves', 'Hesperidium', 'Pepo', 'Multiple Fruit',
       'Bulbs', 'Stem', 'Leaves', 'Fruit', 'Fungai', 'Aggregate Fruit',
       'Fish Counter', 'Ruminants', 'Ruminant', 'Ruminant ', 'Shelfish',
       'Poultry', 'Non-Ruminants', 'Tree Nuts', 'Cereal', 'Legume',
       'Seeds'], dtype=object)
```

In [34]:
```
# remove whitespace from the food item
food_item_list = foodDF['Food Item'].tolist()

food_item_list = [item.strip() for item in food_item_list]
```

In [35]:
```
len(food_item_list)
```

Out[35]:
```
1675
```

In [36]:
```
print(len(foodDF['Food Item'].unique()))
print(len(set(food_item_list)))
```

```
211
205
```

In [37]:
```
# display food item list
set(food_item_list)
```

Out[37]: {'Alfonsino',
'Almond Milk',
'Almonds',
'Anglerfish',
'Apples',
'Apples and Pears',
'Apricots',
'Artichokes',
'Asparagus',
'Avocados',
'Bananas',
'Barley',
'Bass',
'Beans',
'Beans - Gigante/Butter',
'Beans - Green',
'Beans - Green Beans',
'Beans - Green Beans (Phaseolus Vulgaris L.)',
'Beans - Pinto Usa Dried',
'Beans - Plake',
'Beans – French and Runner',
'Beef',
'Beetroot',
'Bigeye Tuna',
'Broccoli',
'Bufalo Milk',
'Buffalo',
'Butter',
'Cabbage',
'Cabbages, Other Brassicas',
'Cahsew Nuts',
'Camembert Cheese',
'Capsicums/Peppers',
'Carp',
'Carrots',
'Carrots and Turnips',
'Catfish',
'Cauliflowers',
'Celery',
'Cephalopodsm Varied (Squid)',
'Cereals Misc.',
'Cheddar Cheese',
'Cheese',
'Cherries',
'Chestnuts',
'Chick Peas',
'Chicken',
'Chillies',
'Citrus Fruit, Misc.',
'Citrus Small',
'Coconut-Milk',
'Coconuts (Incl. Copra)',
'Cod',
'Cod (Fish Stick)',
'Cowpeas',
'Cows Milk',
'Cranberries, Blueberries',
'Cream',
'Cucumbers',
'Cucumbers (G)',
'Cucumbers and Gherkins (G)',
'Currants and Gooseberries',
'Cuttlefish',
'Dates',
'Diamond Fish',
'Duck',
'Eel',
'Eggplants (Aubergines)',
'Eggs',
'Emu',
'Fennel',
'Figs',
'Fish',
'Fish (Mixed)',
'Flatfish',
'Flatfish Varied',
'Fork Beard',
'Garlic',
'Gherkins',
'Gherkins (G)',
'Ginger',
'Goats Cheese',
'Goats Milk',
'Graham Flour',
'Grapefruit and Pomelo',
'Grapes',
'Ground Nuts',
'Guavas',
'Haddock',
'Hake',
'Hake (Fish Stick)',
'Hake European',
'Hake Fillet',
'Hake Senegal',
'Hazlenuts',
'Herring',
'Kangaroo',
'Kiwi Fruit',
'Lamb',
'Lemons and Limes',
'Lentils',
'Lettuce',
'Lettuce (G)',

'Ling',
'Ling Common',
'Lobster',
'Mackeral',
'Mackeral (Fish Stick',
'Mackerel Atlantic',
'Mackerel Horse',
'Maize',
'Maize Sweet Corn',
'Manderin',
'Megrim',
'Melons',
'Melons (G)',
'Millet',
'Mozarella Cranolo',
'Mozarrella',
'Mushrooms',
'Mussels',
'Natural',
'Nuts Misc.',
'Oat',
'Oat Berry',
'Oatmeal',
'Oats',
'Octopus',
'Olives',
'Onion',
'Oranges',
'Palm Nuts-Kernels (Nut Equiv.)/Oil',
'Peach',
'Peaches and Nectarines',
'Peanuts',
'Pears',
'Pears and Apples',
'Pears and Quinces',
'Peas',
'Peas - Dry',
'Peas - Green',
'Peas - Green – Shelled',
'Peas - Yellow Dried',
'Peppers/Capsicums',
'Peppers/Capsicums (G)',
'Pilchard',
'Pineapples',
'Pistachios',
'Plums and Sloes',
'Pollock',
'Pollock (Fish Stick)',
'Pomfret',
'Pomfret Atlantic',
'Porbeagle',
'Pork',
'Potatoes',
'Prawns/Shrimp',
'Pumpkins',
'Quinces and Pears',
'Quinoa',
'Rabbit',
'Rape Seed',
'Rapeseed and Mustard Seed',
'Raspberries and Other Berries',
'Raspberries and Other Berries (G)',
'Rhombus',
'Rice',
'Rock Fish',
'Rockmelon / Cantelope',
'Rye',
'Salmon',
'Sea Bass',
'Semi Hard Chees',
'Semi-Hard',
'Sesame Seed',
'Shark',
'Shark Mako',
'Sole',
'Sorghum',
'Soy-Milk',
'Soybean',
'Spinach',
'Squid',
'Starchy Root',
'Strawberries',
'Strawberries (G)',
'Sunflower Seed',
'Swedes (Rutabage)',
'Swordfish',
'Tangerines, Mandarins Etc.',
'Tomatoes',
'Tomatoes (G)',
'Trout',
'Tuna',
'Turbot',
'Turkey',
'Veal',
'Walnuts',
'Watermelon',
'Wheat',
'Whiting',
'Whiting Blue',
'Yogurt',
'Zucchini/Button Squash',
'Zucchini/Button Squash (G)'}

In [38]:

```
# replace items that contains these following
#keywords to convert them into broader category
item_to_replace = ["Apples", "Pinto", "Cabbage", "Carrot", "Squid", "Cucumber",
                   "Fish", "Beans - Green", "Gherkins", "Cod", "Flatfish", "Hake", "Kiwi",
                   "Lettuce", "Ling", "Macker", "Maize", "Peas - Green", "Citrus",
                  "Manderin", "Melons", "Moza", "Natural", "Oat", "Peach", "Pears", "Peppers",
                  "Plums", "Pollock", "Pomfret", "Pork", "Rape", "Raspberries", "Semi", "Shark", "Strawberries",
                  "Tangerines", "Tomatoes", "Whiting", "Zucchini"]
food_list = []
for item in food_item_list:
    for replace in item_to_replace:
        if replace in item:
            item = replace
    food_list.append(item)
```

In [39]:
```
set(food_list)
```

Out[39]: {'Alfonsino',
'Almond Milk',
'Almonds',
'Anglerfish',
'Apples',
'Apricots',
'Artichokes',
'Asparagus',
'Avocados',
'Bananas',
'Barley',
'Bass',
'Beans',
'Beans - Gigante/Butter',
'Beans - Green',
'Beans - Plake',
'Beans – French and Runner',
'Beef',
'Beetroot',
'Bigeye Tuna',
'Broccoli',
'Bufalo Milk',
'Buffalo',
'Butter',
'Cabbage',
'Cahsew Nuts',
'Camembert Cheese',
'Carp',
'Carrot',
'Catfish',
'Cauliflowers',
'Celery',
'Cereals Misc.',
'Cheddar Cheese',
'Cheese',
'Cherries',
'Chestnuts',
'Chick Peas',
'Chicken',
'Chillies',
'Citrus',
'Coconut-Milk',
'Coconuts (Incl. Copra)',
'Cod',
'Cowpeas',
'Cows Milk',
'Cranberries, Blueberries',
'Cream',
'Cucumber',
'Currants and Gooseberries',
'Cuttlefish',
'Dates',
'Duck',
'Eel',
'Eggplants (Aubergines)',
'Eggs',
'Emu',
'Fennel',
'Figs',
'Fish',
'Flatfish',
'Fork Beard',
'Garlic',
'Gherkins',
'Ginger',
'Goats Cheese',
'Goats Milk',
'Graham Flour',
'Grapefruit and Pomelo',
'Grapes',
'Ground Nuts',
'Guavas',
'Haddock',
'Hake',
'Hazlenuts',
'Herring',
'Kangaroo',
'Kiwi',
'Lamb',
'Lemons and Limes',
'Lentils',
'Lettuce',
'Ling',
'Lobster',
'Macker',
'Maize',
'Manderin',
'Megrim',
'Melons',
'Millet',
'Moza',
'Mushrooms',
'Mussels',
'Natural',
'Nuts Misc.',
'Oat',
'Octopus',
'Olives',
'Onion',
'Oranges',
'Palm Nuts-Kernels (Nut Equiv.)/Oil',
'Peach',
'Peanuts',

```
 'Pears',
 'Peas',
 'Peas - Dry',
 'Peas - Green',
 'Peas - Yellow Dried',
 'Peppers',
 'Pilchard',
 'Pineapples',
 'Pinto',
 'Pistachios',
 'Plums',
 'Pollock',
 'Pomfret',
 'Porbeagle',
 'Pork',
 'Potatoes',
 'Prawns/Shrimp',
 'Pumpkins',
 'Quinoa',
 'Rabbit',
 'Rape',
 'Raspberries',
 'Rhombus',
 'Rice',
 'Rockmelon / Cantelope',
 'Rye',
 'Salmon',
 'Sea Bass',
 'Semi',
 'Sesame Seed',
 'Shark',
 'Sole',
 'Sorghum',
 'Soy-Milk',
 'Soybean',
 'Spinach',
 'Squid',
 'Starchy Root',
 'Strawberries',
 'Sunflower Seed',
 'Swedes (Rutabage)',
 'Swordfish',
 'Tangerines',
 'Tomatoes',
 'Trout',
 'Tuna',
 'Turbot',
 'Turkey',
 'Veal',
 'Walnuts',
 'Watermelon',
 'Wheat',
 'Whiting',
 'Yogurt',
 'Zucchini'}
```

In [40]:
```
len(set(food_list))
```

Out[40]: 158

In [41]:
```
len(food_list)
```

Out[41]: 1675

In [42]:
```
# Refine the food items name by correct spelling and format
replace_list = [['Moza', 'Mozzarella'], ['Natural', 'Cheese'], ['Semi', 'Cheese'], ['Rape', 'Rapeseed'],
                ['Chick Peas', 'Chickpeas'], ['Camembert Cheese', 'Camembert'],['Cahsew Nuts', 'Cashew'],
                ['Hazlenuts', 'Hazelnuts'], ['Peas - Dry', 'Dry Peas'], ['Peas - Green', 'Green Peas'],
                ['Peas - Yellow Dried', 'Yellow Peas'], ['Beans - Gigante/Butter', 'Butter Beans'],
                ['Beans - Green', 'Green Beans'], ['Beans - Plake', 'Plake Beans'], ['Macker', 'Mackerel'],
                ['Beans – French and Runner', 'Runner Beans'], ['Cheddar Cheese', 'Cheddar'],
                ["Coconuts (Incl. Copra)","Coconuts"],["Palm Nuts-Kernels (Nut Equiv.)/Oil","Palm Oil"],
               ["Manderin","Mandarin"]]

food_list_final = []
for item in food_list:
    for item_replace in replace_list:
        if item_replace[0] in item:
            item = item_replace[1]
    food_list_final.append(item)
```

In [43]:
```
len(set(food_list_final))
```

Out[43]: 156

In [44]:
```
len(food_list_final)
```

Out[44]: 1675

In [45]:
```
# Lemmatize and Lowercase food item
food_list_final = [item.replace('-',' ') for item in food_list_final]
food_list_final = [item.lower() for item in food_list_final]
food_list_final = [lemmatizer.lemmatize(item) for item in food_list_final]
```

In [46]:
```
set(food_list_final)
```

Out[46]: {'alfonsino',
'almond',
'almond milk',
'anglerfish',
'apple',
'apricot',
'artichoke',
'asparagus',
'avocado',
'banana',
'barley',
'bass',
'bean',
'beef',
'beetroot',
'bigeye tuna',
'broccoli',
'bufalo milk',
'buffalo',
'butter',
'butter beans',
'cabbage',
'camembert',
'carp',
'carrot',
'cashew',
'catfish',
'cauliflower',
'celery',
'cereals misc.',
'cheddar',
'cheese',
'cherry',
'chestnut',
'chicken',
'chickpea',
'chilli',
'citrus',
'coconut',
'coconut milk',
'cod',
'cowpea',
'cows milk',
'cranberries, blueberries',
'cream',
'cucumber',
'currants and gooseberries',
'cuttlefish',
'date',
'dry peas',
'duck',
'eel',
'egg',
'eggplants (aubergines)',
'emu',
'fennel',
'fig',
'fish',
'flatfish',
'fork beard',
'garlic',
'gherkin',
'ginger',
'goats cheese',
'goats milk',
'graham flour',
'grape',
'grapefruit and pomelo',
'green beans',
'green peas',
'ground nuts',
'guava',
'haddock',
'hake',
'hazelnut',
'herring',
'kangaroo',
'kiwi',
'lamb',
'lemons and limes',
'lentil',
'lettuce',
'ling',
'lobster',
'mackerel',
'maize',
'mandarin',
'megrim',
'melon',
'millet',
'mozzarella',
'mushroom',
'mussel',
'nuts misc.',
'oat',
'octopus',
'olive',
'onion',
'orange',
'palm oil',
'pea',
'peach',
'peanut',

```
'pear',
'pepper',
'pilchard',
'pineapple',
'pinto',
'pistachio',
'plake beans',
'plum',
'pollock',
'pomfret',
'porbeagle',
'pork',
'potato',
'prawns/shrimp',
'pumpkin',
'quinoa',
'rabbit',
'rapeseed',
'raspberry',
'rhombus',
'rice',
'rockmelon / cantelope',
'runner beans',
'rye',
'salmon',
'sea bass',
'sesame seed',
'shark',
'sole',
'sorghum',
'soy milk',
'soybean',
'spinach',
'squid',
'starchy root',
'strawberry',
'sunflower seed',
'swedes (rutabage)',
'swordfish',
'tangerine',
'tomato',
'trout',
'tuna',
'turbot',
'turkey',
'veal',
'walnut',
'watermelon',
'wheat',
'whiting',
'yellow peas',
'yogurt',
'zucchini'}
```

4.4 Category Wrangling

In [47]:
```
# convert category to list and remove whitespace
food_category_list = foodDF['Category'].tolist()
food_category_list = [item.strip() for item in food_category_list]
```

In [48]:
```
# replace the category name
cat_replace_list = [['Fish Counter', 'Fish & Seafood'], ['Ruminant', 'Meat'], ['Shelfish', 'Fish & Seafood'],
                ['Poultry', 'Poultry & Egg'], ['Butter', 'Dairy'], ['Cheese', 'Dairy'],
                ['Cream', 'Dairy'],['Milk', 'Dairy'],['Yogurt', 'Dairy'],
                ['Cereal', 'Flours, Grains, Pulses and Nuts'], ['Legume', 'Flours, Grains, Pulses and Nuts'],
                ['Seeds', 'Flours, Grains, Pulses and Nuts'], ['Tree Nuts', 'Flours, Grains, Pulses and Nuts']]

food_category_temp = []
for item in food_category_list:
    for item_replace in cat_replace_list:
        if item_replace[0] in item:
            item = item_replace[1]
    food_category_temp.append(item)

categories = ['Fish & Seafood', 'Flours, Grains, Pulses and Nuts', 'Dairy', 'Meat', 'Poultry & Egg']

food_category_final = []
for item in food_category_temp:
    if item not in categories:
        item = 'Fruit and Vegetable'
    food_category_final.append(item)
```

In [49]:
```
len((food_category_final))
```

Out[49]:
```
1675
```

In [50]:
```
# integrate final lists to the dataframe
foodDF['Category'] = food_category_final
foodDF['Food Item'] = food_list_final
```

In [51]:
```
foodDF.head()
```

Out[51]:

	Food counter	Food Item	Category	Region	Year	Carbon Footprint (kg CO2-eq/kg)	Farming
0	Dairy Counter	butter	Dairy	United Kingdom	2010	3.7	spreadable
1	Dairy Counter	butter	Dairy	France	2010	7.2	conventional farming
2	Dairy Counter	butter	Dairy	Canada	2006	7.3	conventional farming
3	Dairy Counter	butter	Dairy	Germany	2010	9	conventional farming
4	Dairy Counter	butter	Dairy	United Kingdom	2012	9.5	conventional farming

In [52]:
```
# converting comma in carbon footprint value to point
cabon_foot_list = foodDF['Carbon Footprint (kg CO2-eq/kg)'].tolist()

carbon_foot_list_final = []
for item in cabon_foot_list:
    if "," in str(item):
        item = item.replace(",", ".")
        print(item)
    carbon_foot_list_final.append(item)
```

```
8.93
2.88
7.62
```

In [53]:
```
# convert carbon footprint value to float
foodDF['Carbon Footprint (kg CO2-eq/kg)'] = carbon_foot_list_final
foodDF['Carbon Footprint (kg CO2-eq/kg)']=(foodDF['Carbon Footprint (kg CO2-eq/kg)']).astype(float)
```

In [54]:
```
# group the value of each food item by median
foodDF_agg = foodDF.groupby(['Category', 'Food Item'])['Carbon Footprint (kg CO2-eq/kg)'].median()
```

In [55]:
```
#convert to dataframe
foodDF_agg = foodDF_agg.to_frame()
```

In [56]:
```
foodDF_agg.head()
```

Out[56]:

		Carbon Footprint (kg CO2-eq/kg)
Category	Food Item	
	almond milk	0.417903
	bufalo milk	3.570000
Dairy	butter	9.250000
	camembert	7.550000
	cheddar	13.024000

Export to csv and reload to dataframe

In [57]:
```
foodDF_agg.to_csv('dataset1.csv', index = True)
```

In [58]:
```
food1 = pd.read_csv('dataset1.csv')
```

In [59]:
```
food1.head()
```

Out[59]:

	Category	Food Item	Carbon Footprint (kg CO2-eq/kg)
0	Dairy	almond milk	0.417903
1	Dairy	bufalo milk	3.570000
2	Dairy	butter	9.250000
3	Dairy	camembert	7.550000
4	Dairy	cheddar	13.024000

5. Wrangling Second Data

The second data directly collected from the following URL and read into dataframe

In [60]:
```
URL = "https://world.openfoodfacts.org/cgi/search.pl?action=process&tagtype_0=labels&tag_contains_0=contains&tag_0=Carbon%20footprint&sort_by=unique_scans_n&page_size=20&download=on&format=xlsx"
food2df = pd.read_excel(URL)
```

In [61]:
```
food2df.head()
```

Out[61]:

	code	url	creator	created_t	last_modified_t	product_name	generic_name	quantity	packaging	packaging_tag
0	6181100042177	https://world.openfoodfacts.org/product/618110...	openfoodfacts-contributors	1592957293	1598716424	top mayo grand 500f	Variété	Inférieur	Déplorable	deplorab
1	5000299298282	https://world.openfoodfacts.org/product/500029...	kiliweb	1518793150	1598628043	Shake framboise/myrtille	NaN	550 gr	Barattolo	baratto
2	3189	https://world.openfoodfacts.org/product/3189	frankrzr	1551877246	1598626211	Polvo para preparar bebidas	NaN	140g	Papel metal	papel-met
3	7610200364005	https://world.openfoodfacts.org/product/761020...	dimoqwertz	1461504836	1598619178	Roblochon Val D'arve	NaN	150 g	NaN	Na
4	5010092093045	https://world.openfoodfacts.org/product/501009...	bcatelin	1389309305	1598344274	Soft white	White bread	800g	Plastic bag, en:ldpe-bag	plastic-bag,e ldpe-ba

5 rows × 170 columns

In [62]:
```
# only include required column
food2df_drop = food2df[['pnns_groups_1','pnns_groups_2','main_category','carbon-footprint_100g']]
```

In [63]:
```
food2df_drop.head()
```

Out[63]:

	pnns_groups_1	pnns_groups_2	main_category	carbon-footprint_100g
0	Fat and sauces	Dressings and sauces	en:70-fat-mayonnaise	900.0
1	Composite foods	One-dish meals	en:supplement	0.0
2	unknown	unknown	en:instant-beverages	100.0
3	Milk and dairy products	Cheese	fr:reblochons	119.7
4	Cereals and potatoes	Bread	en:breads	125.0

In [64]:
```
food2df_drop.shape
```

Out[64]:
```
(442, 4)
```

In [65]:
```
# drop null and 0 value in carbon footprint
food2df_drop = food2df_drop.dropna(subset=['carbon-footprint_100g'])
food2df_drop = food2df_drop[food2df_drop['carbon-footprint_100g'] != 0.0]
```

In [66]:
```
food2df_drop.shape
```

Out[66]:
```
(288, 4)
```

In [67]:
```
food2df_drop.head()
```

Out[67]:

	pnns_groups_1	pnns_groups_2	main_category	carbon-footprint_100g
0	Fat and sauces	Dressings and sauces	en:70-fat-mayonnaise	900.0
2	unknown	unknown	en:instant-beverages	100.0
3	Milk and dairy products	Cheese	fr:reblochons	119.7
4	Cereals and potatoes	Bread	en:breads	125.0
5	sugary-snacks	pastries	fr:brioches-aux-oeufs	290.0

In [68]:
```
# filter out french product
word = 'fr:'
#food2df_en = food2df_drop[~food2df_drop['main_category'].str.contains(word, na=False)]
food2df_en = food2df_drop[food2df_drop["main_category"].str.contains(word) == False]
```

In [69]:
```
food2df_en.shape
```

Out[69]:
```
(227, 4)
```

In [70]:
```
# remove composite food and fat,unknown, and sauces category
food2df_en = food2df_en[food2df_en["pnns_groups_1"].str.contains('Composite foods') == False]
food2df_en = food2df_en[food2df_en["pnns_groups_1"].str.contains('Fat and sauces') == False]
food2df_en = food2df_en[food2df_en["pnns_groups_1"].str.contains('unknown') == False]
```

In [71]:
```
food2df_en.shape
```

Out[71]:
```
(177, 4)
```

In [72]:
```
food2df_en.head()
```

Out[72]:

	pnns_groups_1	pnns_groups_2	main_category	carbon-footprint_100g
4	Cereals and potatoes	Bread	en:breads	125.0
6	Beverages	Sweetened beverages	en:sweetened-beverages	5.0
9	Cereals and potatoes	Cereals	en:rices	324.0
13	Sugary snacks	Chocolate products	en:dark-chocolates-with-orange	177.0
14	Fish Meat Eggs	Eggs	en:free-range-chicken-eggs	685.0

In [73]:
```
# refine main_category
food2_category = food2df_en['main_category'].tolist()
food2_category = [item.replace('en:','') for item in food2_category]
food2_category = [item.replace('-',' ') for item in food2_category]
food2df_en['main_category'] = food2_category
```

In [74]:
```
food2df_en.head()
```

Out[74]:

	pnns_groups_1	pnns_groups_2	main_category	carbon-footprint_100g
4	Cereals and potatoes	Bread	breads	125.0
6	Beverages	Sweetened beverages	sweetened beverages	5.0
9	Cereals and potatoes	Cereals	rices	324.0
13	Sugary snacks	Chocolate products	dark chocolates with orange	177.0
14	Fish Meat Eggs	Eggs	free range chicken eggs	685.0

In [75]:
```
# add filter from the first dataset to make sure no duplicates added
filters_list = [item.lower() for item in food1['Food Item'].tolist()]

filters = set(filters_list)
# add aditional filters
filters.update(['milk','beverage','flour','cereal','canned','prepared','sausage','dessert','tea','groceries','coffee','ham'])
```

In [76]:
```
#lowercase the item
food_item2_og = [item.lower() for item in food2df_en['main_category'].tolist()]
```

In [77]:
```
# created deleted list item by iterating through filters
deleted = []
for item in food_item2_og:
    for filt in filters:
        if filt in item:
            deleted.append(item)
```

In [78]:
```
# delete the food item
food2df_filtered = food2df_en[~food2df_en['main_category'].isin(deleted)]
food2df_filtered = food2df_filtered[food2df_filtered["pnns_groups_2"].str.contains('Processed meat') == False]
```

In [79]:
```
food_item2 = food2df_filtered['main_category'].tolist()

# lemmatize item
food_item2 = ['grisons' if item=='meat of the grisons' else item for item in food_item2]
food_item2 = [lemmatizer.lemmatize(item) for item in food_item2]
```

In [80]:
```
food2df_filtered['main_category'] = food_item2
```

In [81]:
```
# convert carbon footprint value
carbon = food2df_filtered['carbon-footprint_100g'].tolist()
carbon = [item/100 for item in carbon]
food2df_filtered['carbon-footprint_100g'] = carbon
```

In [82]:
```
# replace the category based on the first dataset
group1_og = food2df_filtered['pnns_groups_1'].tolist()
group2_og = food2df_filtered['pnns_groups_2'].tolist()

cat_replace_list = [['Beverages', 'Snack and Others'], ['Cereals and potatoes', 'Flours, Grains, Pulses and Nuts'],
                    ['Shelfish', 'Fish and Seafood'], ['Fruits and vegetables', 'Fruit and Vegetable'],
                    ['Salty snacks', 'Snack and Others'], ['Sugary snacks', 'Snack and Others']]

food_category_temp = []
for item in group1_og:
    for item_replace in cat_replace_list:
        if item_replace[0] in item:
            item = item_replace[1]
    food_category_temp.append(item)

food_category_final = []
for index in range(len(food_category_temp)):
    if food_category_temp[index] == 'Fish Meat Eggs':
        food_category_final.append(group2_og[index])
    else:
        food_category_final.append(food_category_temp[index])

for index in range(len(food_category_final)):
    if food_category_final[index] == 'Flours, Grains, Pulses and Nuts':
        food_category_final[index] = 'Grain and Nut'
```

In [101]:
```
set(food_category_final)
```

Out[101]:
```
{'Fish and seafood', 'Grain and Nut', 'Meat', 'Snack and Others'}
```

In [83]:
```
# integrate final list to dataframe
food2df_filtered['pnns_groups_1'] = food_category_final
```

In [84]:
```
food2df_filtered = food2df_filtered[['pnns_groups_1','main_category','carbon-footprint_100g']]
# Rename column
food2df_filtered = food2df_filtered.rename(columns={'pnns_groups_1': 'Category',
                                'main_category': 'Food Item',
                                'carbon-footprint_100g': 'Carbon Footprint (kg CO2-eq/kg)'})
```

In [85]:
```
# group by food item median
foodDF2_agg = food2df_filtered.groupby(['Category', 'Food Item'])['Carbon Footprint (kg CO2-eq/kg)'].median()
foodDF2_agg = foodDF2_agg.to_frame()
```

In [86]:
```
# convert to csv and reload to dataframe
foodDF2_agg.to_csv('food2.csv', index=True)
food2 = pd.read_csv('food2.csv')
```

In [87]:
```
food1.head()
```

Out[87]:

	Category	Food Item	Carbon Footprint (kg CO2-eq/kg)
0	Dairy	almond milk	0.417903
1	Dairy	bufalo milk	3.570000
2	Dairy	butter	9.250000
3	Dairy	camembert	7.550000
4	Dairy	cheddar	13.024000

In [119]:
```
food2.head()
```

Out[119]:

	Category	Food Item	Carbon Footprint (kg CO2-eq/kg)
0	Fish and seafood	sardine	0.120
1	Fish and seafood	sardines in oil and lemon	3.450
2	Fish and seafood	sardines in sunflower oil	3.700
3	Grain and Nut	bread	0.091
4	Grain and Nut	bread crumbs	2.750

In [173]:
```
# concat the dataframe and reset index
finalDF = pd.concat([food1, food2], sort=False)
finalDF = finalDF.sort_values(by=['Category','Food Item'])
finalDF = finalDF.reset_index(drop=True)
finalDF.head()
```

Out[173]:

	Category	Food Item	Carbon Footprint (kg CO2-eq/kg)
0	Dairy	almond milk	0.417903
1	Dairy	bufalo milk	3.570000
2	Dairy	butter	9.250000
3	Dairy	camembert	7.550000
4	Dairy	cheddar	13.024000

In [174]:
```
finalDF
```

Out[174]:

	Category	Food Item	Carbon Footprint (kg CO2-eq/kg)
0	Dairy	almond milk	0.417903
1	Dairy	bufalo milk	3.570000
2	Dairy	butter	9.250000
3	Dairy	camembert	7.550000
4	Dairy	cheddar	13.024000
...	...	...	...
186	Snack and Others	marshmallow	3.650000
187	Snack and Others	multifruit nectars	1.020000
188	Snack and Others	palmiers	3.650000
189	Snack and Others	swiss chocolates	11.755000
190	Snack and Others	white chocolates	14.400000

191 rows × 3 columns

In [175]:
```
# rounding carbon footprint value
carbonFootprint = finalDF['Carbon Footprint (kg CO2-eq/kg)'].tolist()
carbonFootprint = [round(item,2) for item in carbonFootprint]
finalDF['Carbon Footprint (kg CO2-eq/kg)'] = carbonFootprint
```

In [176]:
```
finalDF = finalDF.rename(columns={'Category': 'category',
                                  'Food Item': 'food',
                                  'Carbon Footprint (kg CO2-eq/kg)': 'footprint'})

finalDF.head()
```

Out[176]:

	category	food	footprint
0	Dairy	almond milk	0.42
1	Dairy	bufalo milk	3.57
2	Dairy	butter	9.25
3	Dairy	camembert	7.55
4	Dairy	cheddar	13.02

In [177]:
```
#Lowercase the category and reassign to dataframe
category_lower = [item.lower() for item in finalDF['category'].tolist()]
for index in range(len(category_lower)):
    if category_lower[index] == 'flours, grains, pulses and nuts':
        category_lower[index] = 'grain and nut'
    elif category_lower[index] == 'fish & seafood':
        category_lower[index] = 'fish and seafood'
    elif category_lower[index] == 'poultry & egg':
        category_lower[index] = 'poultry and egg'

finalDF['category'] = category_lower
```

In [178]:
```
# checking the dataframe
finalDF[finalDF['category'] == 'grain and nut']
```

Out[178]:

	category	food	footprint
54	grain and nut	almond	1.54
55	grain and nut	barley	0.43
56	grain and nut	bean	1.37
57	grain and nut	butter beans	0.39
58	grain and nut	cashew	1.44
59	grain and nut	cereals misc.	0.43
60	grain and nut	chestnut	0.43
61	grain and nut	chickpea	0.77
62	grain and nut	cowpea	0.48
63	grain and nut	dry peas	0.33
64	grain and nut	graham flour	0.47
65	grain and nut	green beans	0.46
66	grain and nut	green peas	1.38
67	grain and nut	ground nuts	0.99
68	grain and nut	hazelnut	0.96
69	grain and nut	lentil	1.03
70	grain and nut	maize	0.47
71	grain and nut	millet	0.47
72	grain and nut	nuts misc.	0.47
73	grain and nut	oat	0.57
74	grain and nut	palm oil	0.80
75	grain and nut	pea	1.01
76	grain and nut	peanut	0.83
77	grain and nut	pinto	0.73
78	grain and nut	pistachio	1.52
79	grain and nut	plake beans	0.26
80	grain and nut	quinoa	1.15
81	grain and nut	rapeseed	1.19
82	grain and nut	rice	2.55
83	grain and nut	runner beans	0.75
84	grain and nut	rye	0.38
85	grain and nut	sesame seed	0.88
86	grain and nut	sorghum	0.88
87	grain and nut	soybean	0.48
88	grain and nut	sunflower seed	1.41
89	grain and nut	walnut	1.51
90	grain and nut	wheat	0.52
91	grain and nut	yellow peas	0.20
147	grain and nut	bread	0.09
148	grain and nut	bread crumbs	2.75
149	grain and nut	brown breads	1.16
150	grain and nut	fresh pasta	2.89
151	grain and nut	frozen fries	2.23
152	grain and nut	tofu	2.00
153	grain and nut	torti	1.85

In [179]:
```
finalDF.reset_index(level=0, inplace=True)
finalDF = finalDF.rename(columns={'index': 'id'})

finalDF.head()
```

Out[179]:

	id	category	food	footprint
0	0	dairy	almond milk	0.42
1	1	dairy	bufalo milk	3.57
2	2	dairy	butter	9.25
3	3	dairy	camembert	7.55
4	4	dairy	cheddar	13.02

Export dataframe to csv and json

In [180]:
```
finalDF.to_csv('CarbConData.csv', index=False)
```

In [181]:
```
finalDF.to_json('CarbConData.json',orient="records")
```