University of British Columbia

Social-Ecological Economic Development Studies (SEEDS) Sustainability Program

Student Research Report

CLIMATE-FRIENDLY FOOD SYSTEMS (CFFS) LABELLING PROJECT

An Evaluation Framework for the Operationalization of UBC Vancouver's Climate-Friendly Food Label

Prepared by: Silvia Huang, Climate-Friendly Food Systems Data Analyst

Supervised by: Juan Diego Martinez, Institute for Resources Environment and Sustainability

Prepared for: UBC Food Services and UBC Botanical Gardens

University of British Columbia

August 2021

Disclaimer: "UBC SEEDS Sustainability Program provides students with the opportunity to share the findings of their studies, as well as their opinions, conclusions, and recommendations with the UBC community. The reader should bear in mind that this is a student research project and is not an official document of UBC. Furthermore, readers should bear in mind that these reports may not reflect the current status of activities at UBC. We urge you to contact the research persons mentioned in a report or the SEEDS Sustainability Program representative about the current status of the subject matter of a report".



UBC SUSTAINABILITY

TABLE OF CONTENTS	
LIST OF FIGURES	2
LIST OF ABBREVIATIONS	2
EXECUTIVE SUMMARY	3
1. INTRODUCTION	5
1.1 RESEARCH CONTEXT & TOPIC	5
1.2 RESEARCH RELEVANCE	6
1.3 PROJECT PURPOSE, GOALS, AND OBJECTIVES	6
2. METHODOLOGY	7
2.1 RESEARCH METHODOLOGY AND METHODS	7
2.2 DATA COLLECTION	7
2.3 ASSUMPTIONS	9
2.4 EVALUATION OF MENU ITEMS	9
2.5 BASELINE AND LABEL CUT-OFFS	10
2.6 ADDITIONAL ATTRIBUTES	12
3. RESULTS	13
3.1 EXTERNAL FRAMEWORK	13
3.2 SUMMER PILOT	14
3.3 FALL LAUNCH	17
4. DISCUSSION	18
5. RECOMMENDATIONS	20
5.1 RECOMMENDATIONS FOR ACTION AND IMPLEMENTATION	20
5.2 RECOMMENDATIONS FOR FUTURE RESEARCH	20
6. CONCLUSION	21
REFERENCES	22
APPENDICES	23
APPENDIX A [CODE FOR EXTERNAL FRAMEWORK]	23
APPENDIX B [GHG EMISSION FACTORS LIST]	54

LIST OF FIGURES

- Figure 1: Flowchart for Calculating the GHG Emissions of a Bacon Sandwich
- Figure 2: Traffic Light Labelling System
- Figure 3: Evaluation Framework Flowchart
- Figure 4: GHG Emissions (Kg) Per Serving for Summer Pilot
- Figure 5: CFFS Label on Menu Board
- Figure 6: GHG Emissions (Kg) Per Serving vs. Per 100g for Summer Pilot
- Figure 7: Label Cut-offs for Summer Pilot by Food Groups

LIST OF ABBREVIATIONS

- GHG: Greenhouse Gas
- CFFS: Climate-Friendly Food Systems
- UBCFS: UBC Food Services
- OC: Optimum Control
- CAP: Climate Action Plan
- UBCFSP: UBC Food System Project

EXECUTIVE SUMMARY

How to make your daily menu choices climate-friendly? Roughly 26% of global total greenhouse gas (GHG) emissions generated by human activities were contributed by the food supply chain (Poore & Newecek, 2018). This brings a range of opportunities for actions to mitigate the effect of food systems on the climate.

The Climate-Friendly Food Systems (CFFS) labelling project at the University of British Columbia (UBC) takes action to provide students with the climate impact information of menu items they purchase every day that could help to educate, bring awareness and influence their purchasing behavior in a more climate-friendly way. This research report was prepared by the CFFS data analyst, a member of the CFFS Action Team and the CFFS Labelling Project Research Group. This report is focused on the data analysis and back-end implementation of the CFFS labelling project and is complementary to the report on the communication and definition side prepared by the CFFS communication and engagement coordinator.

The CFFS labelling project is part of the actions taken by UBC in response to the Climate Action Plan (CAP) 2030 scope 3 emission reduction goal. The CFFS Action Team was formed to accelerate transitions towards a climate-friendly food system and advance the UBC Food System Project mission and priorities. This project aims to evaluate the climate impact of menu items sold at UBC Food Services (UBCFS) venues and operationalize the CFFS food label to inform climate-friendly menu choices. The goal of this project includes creating a reproducible data analysis framework for calculating recipes' greenhouse gas (GHG) emissions, establishing a food GHG emission baseline at the UBC Vancouver campus, determining cut-offs for the CFFS traffic-light label, and further integrating additional CFFS attributes into the framework for expanded impact.

This project utilized a combination of literature review, discussion with peer institutions, and assessment of the feasibility in the UBC's context to decide the methodology. The primary data sources (recipes and sales data) were extracted from the UBCFS inventory management system, Optimum Control. The data on GHG emission factors came from external secondary data sources.

The main deliverable of the project is the external framework that conducts the evaluation process of recipes automatically once GHG emission factors have been assigned to each ingredient, and it will be further developed to incorporate additional attributes and adapt to the expansion of the CFFS label. The external framework is able to read the primary data automatically and output the total GHG emissions of each menu item. To determine the cut-offs for the levels of the label according to GHG emissions, we established a 2019 UBCFS GHG emission baseline and set cut-offs in accordance with the CAP 2030 GHG scope 3 reduction goals for food systems. For the initial pilot phase of the label implementation, we determined separate sets of cut-offs for different meal groups (i.e., lunch/dinner, breakfast, desserts/snacks) due to the incomparability between products from different meal groups.

To help the transition to a climate-friendly food system, we suggest that one way to mitigate the total food system emissions is to reduce the amount of meat and dairy consumption and replace them with plant-based protein products without compromising nutritional value. In addition, to improve the accuracy and specificity of current labels, we recommend UBC lead the engagement process and the establishment of a Pacific Northwest/Canadian-specific GHGe factors database by conducting research collaboratively with peer institutions.

Keywords: climate label, climate-friendly, reproducible framework, greenhouse gas, food systems

References

Poore, J., and T. Nemecek. 2018. "Reducing Food's Environmental Impacts through Producers and Consumers." *Science* 360 (6392):987–92. doi:10.1126/science.aaq0216.

1. INTRODUCTION

1.1 RESEARCH CONTEXT & TOPIC

Roughly 26% of global total greenhouse gas (GHG) emissions (13.7 billion tons of carbon dioxide equivalents (CO2eq)) generated by human activities were contributed by the food supply chain (Poore & Newecek, 2018). This brings a range of opportunities for climate action to mitigate the effect of food systems on the environment. In December 2019, UBC joined organizations and governments around the world to declare a climate emergency and renewed its commitment to sustainability, including a commitment to a Climate Action Plan (CAP) 2030 (an update from a 2020 plan) to accelerate UBC's climate actions. As part of the CAP 2030, food was identified as an area of opportunity under scope 3 (indirect) emissions.

The purpose of the Climate-Friendly Food Systems (CFFS) Action Team is to serve as engaged experts from the existing UBC Food System Project (UBCFSP) Steering Committee. The CFFS Action Team is responsible for the ideation, coordination, and development of student-led research, initiatives, and interdisciplinary collaborations that can accelerate transitions towards a climate-friendly food system and advance UBCFSP's mission and priorities. In response to UBC's CAP 2030, the CFFS Action Team aims to achieve a 50% GHG emission reduction associated with food systems by 2030 compared to 2019, starting with the development of a Food System Resilience & Climate Action Strategy, with support for campus-wide climate food labelling, and a toolkit to encourage more sustainable dietary choices and habits.

This project researches how to implement and operationalize the CFFS labels across campus by developing a back-end evaluation framework for the climate impact of menu items and implementing a label that indicates the impact of food sold at UBCFS. The main objective is constructing an evaluation framework for analyzing the recipes and ingredients to provide a

weighted metric that informs customers about the food's climate impact. The evaluation framework will incorporate a range of attributes that indicate aspects of the definition of CFFS for food products. The definition work and the additional attributes can be found in the complementary report developed by the CFFS Communications and Engagement Coordinator. Along with other education and engagement materials, the label will indicate and incorporate a range of CFFS attributes to give a comprehensive view of the food's climate impact that students purchase at UBCFS.

1.2 RESEARCH RELEVANCE

In order to mitigate GHG emissions and other climate impacts of the food system, various actions from the food production and consumption side are necessary. As a major food provider at the UBC campus, UBC Food Services contributed to a large proportion of the total GHG emissions from the food systems through students' daily meals. The action of providing students with the GHG emission information of menu items they purchase every day could help to educate and influence their purchasing behavior in a more climate-friendly way (Brunner et al., 2018). The CFFS label is a clear and efficient presentation to indicate the climate impact information of menu items, thus helping students make purchasing choices that take the climate impacts into consideration.

1.3 PROJECT PURPOSE, GOALS, AND OBJECTIVES

This project aims to operationalize the CFFS label by constructing an evaluation framework for analyzing the climate impact of menu items sold at UBC Food Services venues. This includes creating a reproducible data analysis framework for calculating recipes' GHG emissions, establishing a food GHG emissions baseline for the UBC Vancouver campus, deciding cut-offs for the CFFS label, and further integrating additional CFFS attributes into the framework.

2. METHODOLOGY

2.1 RESEARCH METHODOLOGY AND METHODS

This project utilized a combination of literature review, discussion with peer institutions, and assessment of the feasibility in the UBC's context, such as available data and department support, to decide the methodology that best met the goals and objectives of this research. Methods were also determined through discussion with researchers from the <u>University of</u> <u>Michigan</u>, Université Laval, and the University of Victoria who are working on similar climate food labelling projects.

The research methods include primary and secondary data collection, evaluating recipes' GHG emissions, developing an external data analysis framework, constructing a UBC GHG emission baseline, deciding label cut-offs, and incorporating additional attributes. Detailed explanations are provided below.

2.2 DATA COLLECTION

2.2.1 PRIMARY DATA COLLECTION

The raw recipe data for menu items sold at UBC food venues was extracted from the inventory management system Optimum Control (OC) of UBC by the FMIS Administrator of UBC Food Services. Due to system and administration restrictions, the data extraction was conducted manually instead of using database queries. Recipe data was extracted in XML file format, and each file contained one aspect of the recipe information, such as raw ingredients, preprocessed recipes used, and unit conversion information. The evaluation framework was designed in accordance with this data structure.

For the summer pilot at Mercante, in order to establish a 2019 GHG emissions baseline, the sales data for all products between January 1 and December 31, 2019 were extracted from

Optimum Control. Future iterations will include the residence dining halls sales and recipe emissions data to calculate the UBCFS GHG emission baseline.

2.2.2 Secondary Data Collection

The GHG emission factor data comes from three main sources, in the following order of preference:

- First, we used the World Resources Institute (WRI)'s Cool Food Calculator emission factors for most of the food groups. It provides GHG emission data based on life-cycle assessments for major food categories in the North American region from research conducted from January 2015 to December 2018. These represented the factors used in the large majority of our ingredients (67%).
- Second, we used the GHG emission data from The Big Climate Database, published by CONCITO (Denmark's green think tank), as a supplementary data source for food categories that are not in the Cool Food Calculator. It provides GHG emission data based on life-cycle assessments for major food categories in Denmark.
- Last, for some items that don't have emission factors available, we calculated their emission factors manually by approximating their ingredients using recipes stored in OC or recipes found online.

Note that the food groups were slightly adjusted from the Cool Food Calculators for better assignment of GHG emission factors on ingredients procured by UBC Food Services. For example, the GHG emission factors for more general-level food groups (i.e., fruits) were used for assigning ingredients that were not specified as less general food groups (i.e., apples, bananas, berries) and were renamed as "other" (i.e., other fruits) in the GHG emission factors list. See Appendix B for detailed food categories and emission factors.

2.3 ASSUMPTIONS

- To make the process of recipe evaluation consistent, accurate, and structured, several assumptions were made when evaluating their GHG emissions. The same GHG emission factor was assigned to different forms (puree, sliced, chopped, etc.) of the same raw ingredient.
- The same GHG emission factors were applied to different varieties of the same ingredient (i.e., red and yellow onions).
- GHG emissions are for the raw ingredients, and final weight of serving takes into account loss
 or addition of weight during cooking process (e.g. Water evaporation in beef vs. water
 absorption in pasta) or loss from cutting out inedible parts (prepping stage).
- GHG emissions from the cooking process were ignored.
- The GHG emission factor for water is zero, and we ignored the water use in the cooking process.
- We excluded the GHG emissions from sauces and dressings that have no dominant ingredients.

2.4 EVALUATION OF MENU ITEMS

The GHG emissions of each menu item are calculated by summing up the weight of every raw ingredient multiplied by their respective emission factors. Ingredients' emission factors are assigned according to their category in the Cool Food Calculator, which provides the data about the amount of GHGs emitted to the environment during the entire life cycle of a menu item.

For example, the process flowchart for calculating the GHG emissions of a bacon sandwich is shown in *Figure* 1 below. First, we get the raw ingredient (item) information and then categorize each item into the food categories in the GHG Emission Factors List. See Appendix B for all food

categories and associated GHG emission factors. Next, we assign the GHG emission factors based on the food category for each item and calculate the amount of GHG emissions in grams for each item used in this recipe. For recipes that use pre-processed recipes (preps), such as the garlic butter made of garlic and butter in this example, we calculate a GHG emission factor for this prep based on the items used and then calculate the total amount of GHG emissions in grams for this prep. Lastly, we sum up all the GHG emissions of each item or prep and use this sum and the food group (i.e., lunch/dinner, breakfast, or desserts/snacks) to determine the label color.

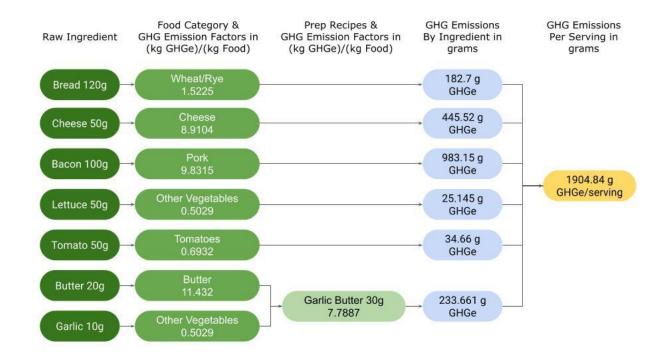


Figure 1: Flowchart for Calculating the GHG Emissions of a Bacon Sandwich

2.5 BASELINE AND LABEL CUT-OFFS

We decided to use the traffic light system to categorize foods by their climate impact into high, medium, and low levels, corresponding to the colors of red, yellow, and green. It would allow easy interpretation for customers to see the food's emission level by looking at the colors. See *Figure* 2 for the design and meaning of the labels implemented during the summer pilot.

To determine the cut-off levels of the label according to the GHG emissions of menu items, we decided to establish a UBCFS GHG emission baseline that represents the average GHG emissions per dish before the label is launched. In this way, we can set cut-offs in accordance with the 50% UBC CAP 2030 GHG reduction goals for food systems. This requires utilizing the sales and recipe data during a period and then calculating the average GHG emissions per dish. In addition, we decided to have separate sets of cut-offs for different meal groups (i.e., lunch/dinner, breakfast, desserts/snacks) due to the disparity in serving size and main ingredients.

The methods for determining cut-offs for the three levels of the label are shown below:

- Green: These food items have below-average GHG emissions compared to other food items sold within the same meal category (i.e., lunch/dinner, breakfast, or desserts/snacks) and have low enough emissions to achieve UBC's 50% reduction target in food-related GHG emissions.
- Yellow: These food items have below-average GHG emissions compared to other food items sold within the same meal category (i.e., lunch/dinner, breakfast, or desserts/snacks) but higher emissions than what is necessary to achieve UBC's 50% reduction target in food-related GHG emissions.
- Red: These food items have above-average GHG emissions compared to other food items sold within the same meal category (i.e., lunch/dinner, breakfast, or desserts/snacks). Food with red labels would drive the average GHG emissions higher, thus impeding the process for UBC in achieving the 50% reduction target in food-related GHG emissions.



Figure 2: Traffic Light Labelling System

2.6 ADDITIONAL ATTRIBUTES

Besides GHG emissions, the evaluation framework also considers the incorporation of one additional attribute for the fall launch to produce a more comprehensive CFFS label. The additional attributes were the metrics to define a Climate-Friendly Food System by the CFFS Action Team, which were developed based on aspects of climate change mitigation and adaptation.

The potential additional attributes are land use, nitrogen pollution, water use, local, which were developed from the CFFS definition research conducted by the CFFS Communication and Engagement Coordinator. To decide which additional attribute should be incorporated, we evaluated these attributes based on the availability of data, UBCFS's tracking capability for qualitative attributes, their impact on climate change mitigation and adaptation, and evaluation survey results.

3. RESULTS

3.1 EXTERNAL FRAMEWORK

The evaluation of menu items is an automatic process that is conducted by an external framework, a workflow documented in Python on Jupyter notebooks that calculates the GHG emission of menu items in an efficient and structured way. It reads the .xml files exported from Optimum Control and does most of the calculating process. See Appendix A for the code that constructed the external framework. The process flowchart for the whole evaluation process is shown in *Figure 3*:

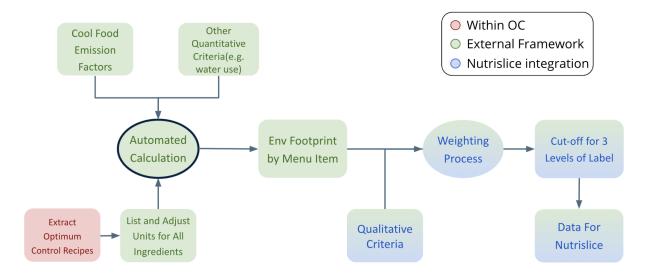


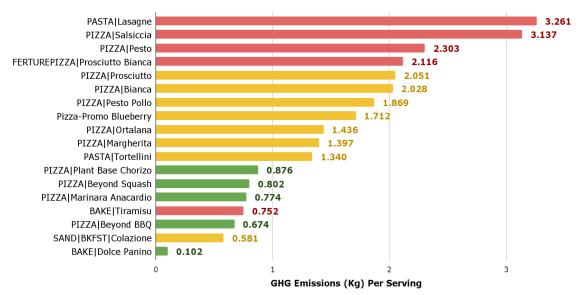
Figure 3: Evaluation Framework Flowchart

This flowchart presents the main steps and components that make up the whole evaluation process. And the color of each box indicates where this step takes place, or which system or software is associated with it. For a box that has two colors, it means it is associated with two systems or can happen in either place.

The first step is extracting raw ingredients and recipe data from the UBCFS inventory management system. Before feeding these raw data into the automated calculation process, it requires preprocessing and cleaning these data by listing and adjusting units for all ingredients and assigning them with associated GHG emission factors, which are from several external data sources such as the Cool Food Calculator. Data extraction from OC and preprocessing represent the largest time requirements every time new recipes need evaluation. Besides GHG emissions, we are planning to assign the ingredients with additional quantitative criteria data (i.e., land use) for the fall launch. After these data gets processed in the automated calculation step/external framework, it will output the environmental footprint of each menu item, and then we weigh these results with other qualitative attributes to have a weighted metric of the overall climate impact of each menu item. Lastly, we use the baseline data to decide the cut-offs for the three levels of labels, and the results can be shown on the Nutrislice, which is the online platform where students can see nutrition facts and also the climate label of the food they buy at UBC Food Services.

3.2 SUMMER PILOT

The summer pilot for the operationalization of CFFS labels took place at the Mercante, one of the UBC Food Services retail venues that remained open during summer 2021. The evaluation only focused on the GHG emissions of the menu items, most of which are pizzas that have almost the same serving size. The total GHG emission for each menu item, calculated by the external framework, is shown in *Figure* 4:



Note: the GHG emission results are based on 2021 data

Figure 4: GHG Emissions (Kg) Per Serving for Summer Pilot

The corresponding CFFS labels are available to students on the menu boards and also on the

Nutrislice. See Figure 5 for the actual look of labels.

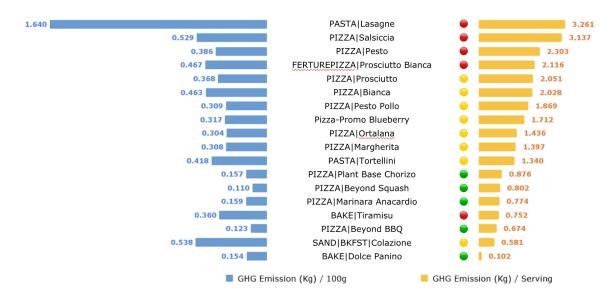


Figure 5: CFFS Label on Menu Board

The external framework also calculated the GHG emissions per 100g of the product for each

item. This gives another point of view for comparing the climate impact of the recipes. Although

there are a few products that have high per 100 gram GHG emissions, which indicates that they may use a lot of high-emission ingredients, the total emissions are low due to the small serving size. To make the label easier for interpretation by the customer and align with the goal of reducing total GHG emissions, we chose to assign labels based on total GHG emissions per serving of the products, see *Figure* 6.



Note: the GHG emission results are based on 2021 data

Figure 6: GHG Emissions (Kg) Per Serving vs. Per 100g for Summer Pilot

The label cut-offs for the summer pilot are shown in *Figure* 7. GHGs are evaluated based on meal categories (lunch/dinner, breakfast, or desserts/snacks). Menu items are categorized as green, yellow, or red, depending on whether they have below or above average GHG emissions compared to other food items sold at Mercante within the same meal category. The categories also consider if food items have low enough emissions to achieve UBC's food emissions targets.

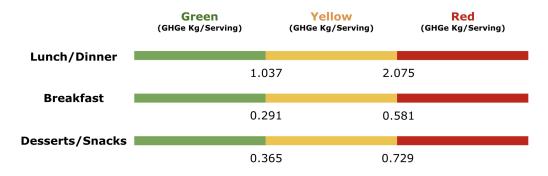


Figure 7: Label Cut-offs for Summer Pilot by Food Groups

3.3 FALL LAUNCH

The fall launch for the operationalization of CFFS labels will take place at the Open Kitchen, one of the three UBC Food Services residence dining halls that open during the 2021-2022 academic year. Besides GHG Emissions, the CFFS label is going to incorporate one additional attribute into the evaluation to produce a more comprehensive evaluation of the climate impact of menu items.

4. DISCUSSION

From the above analysis, we can see that food that contains ruminant meat and dairy products (i.e. beef, lamb, cheese, etc.) tends to have high GHG emissions, both per serving and per 100g measuring method. This could suggest that one way to lower the GHG emissions from the food system is by reducing the amount of meat and dairy consumption and switching to plant-based protein products (i.e., beans, tofu, etc.). For example, the difference between the Salsiccia Pizza (the pizza with the highest GHG emissions at Mercante with chorizo, tomato, basil, oregano, and mozzarella) and the Beyond BBQ Pizza (the pizza with the lowest GHG emissions at Mercante with beyond meat crumble, chipotle BBQ sauce, arugula, and mushrooms) is 2,463 grams of CO2eq, which is equivalent to the emissions from a 11.96-kilometer drive in an average passenger vehicle (average of 206g CO2 emissions per km driven, Canada Energy Regulator, 2019).

There are also some limitations in the evaluation framework. First, there are several processed products and packaged foods that are directly purchased from external suppliers, such as sauces, dressings, and snacks, etc. Therefore, the evaluation can only take the best estimation of their GHG emission factors by manually calculating the ingredients contained in these products using the available GHG factors.

Secondly, emissions from bucket items such as "parfait," "salad bar," and "build your own" represent an average with a lot of variance since they are customized by the client. The recipes for these products recorded in the system use the estimated average amount for each composition that customers may choose.

Lastly, there is human dependence on matching items with associated emission factors. Although manually matching takes less time and is more accurate, this may raise some problems if the

label is expanded to more food venues and thus human work will take more time. Besides, the information for ingredients stored in the Optimum Control is incomplete for some items, such as the unit information and conversion data, which need to be adjusted and inserted manually.

5. RECOMMENDATIONS

5.1 RECOMMENDATIONS FOR ACTION AND IMPLEMENTATION

To improve the evaluation framework and make it more resilient and suited for expanded

operations, the steps below could be used for development:

- Improve the recording and tracking of food information stored in the inventory management system and reduce the amount of missing data for ingredients and recipes.
- Incorporate the climate footprint data for ingredients into the inventory management system if feasible to embed the calculation process within the system.

5.2 RECOMMENDATIONS FOR FUTURE RESEARCH

• UBC can lead the engagement process to build a Pacific Northwest/Canadian specific GHGe factors database by conducting research together with peer institutions. This can also help to improve the accuracy and specificity of current labels.

6. CONCLUSION

In conclusion, the CFFS label evaluation framework is a resilient approach to conduct the evaluation process in an efficient and structured way that meets the needs for the future expansion of the CFFS label. However, there are a few limitations in the current framework due to the missing information from the data sources and the manual reliance on cleaning, assigning, and extracting data. The recommendation for the next steps is to streamline the extraction process and improve the tracking of ingredient information in the systems. It will require more time, resourcing, and close coordination between associated departments to produce a comprehensive CFFS label that indicates all-around information on the climate impact of menu items sold by UBCFS.

REFERENCES

- Brunner, F., Kurz, V., Bryngelsson, D., Hedenus, F., Department of Economics, Institutionen för nationalekonomi med statistik, Handelshögskolan, Göteborgs universitet, Gothenburg University, & School of Business, Economics, and Law. (2018). Carbon label at a university restaurant – label implementation and evaluation. *Ecological Economics, 146*, 658-667. https://doi.org/10.1016/j.ecolecon.2017.12.012
- Poore, J., and T. Nemecek. 2018. "Reducing Food's Environmental Impacts through Producers and Consumers." *Science 360* (6392):987–92. doi:10.1126/science.aaq0216.
- Searchinger, T., R. Waite, C. Hanson, J. Ranganathan, P. Dumas, and E. Matthews. 2019. "World Resources Report: Creating a Sustainable Food Future—A Menu of Solutions to Feed Nearly 10 Billion People by 2050 (Final Report)." Washington, DC: World Resources Institute. <u>http://www.sustainablefoodfuture.org</u>.
- Waite, R., D. Vennard, and G. Pozzi. 2019. "Tracking Progress Toward the Cool Food Pledge: Setting Climate
 Targets, Tracking Metrics, Using the Cool Food Calculator, and Related Guidance for Pledge
 Signatories." Technical Note. Washington, DC: World Resources Institute. Available online at:
 www.coolfoodpledge.org.

APPENDICES

APPENDIX A [CODE FOR EXTERNAL FRAMEWORK]

Climate-Friendly Food Systems (CFFS) Labelling Project

The University of British Columbia

Created by Silvia Huang

Part I: Data Preprocessing

Set up and Import Libraries

In [1]:	#install libraries if needed #1pip3 install pdpipe #1pip install watermark
In [2]:	<pre>import numpy as np import pandas as pd import pdpipe as pdp import matplotlib.pyplot as plt import glob import os import os from itertools import islice from decimal import Decimal import xml.etree.ElementTree as et from xml.etree.ElementTree import parse import openpyxl import pytest</pre>
In [3]:	<pre>#set the root path, change the directory into the project folder os.chdir("/Users/silvia/cffs-label")</pre>
In [4]:	<pre>#enable reading data in the scrolling window #pd.set_option("display.max_rows", None, "display.max_columns", None)</pre>

Load Data Files

Set Data File Path

```
In [5]: #selecting data file path for the chosen venue and time range
filepath_list = glob.glob(os.path.join(os.getcwd(), "data", "raw", "OK 21-22 Sep-Dec","*.oc"))
filepath_list
Out[5]: ['/Users/silvia/cffs-label/data/raw/OK 21-22 Sep-Dec/OK Al Forno_Custom Kitchen_Dim Sum_Global.oc',
'/Users/silvia/cffs-label/data/raw/OK 21-22 Sep-Dec/OK Square Meal.oc',
'/Users/silvia/cffs-label/data/raw/OK 21-22 Sep-Dec/OK Square Meal.oc',
'/Users/silvia/cffs-label/data/raw/OK 21-22 Sep-Dec/OK Grult Kitchen_Sides_Soup.oc',
'/Users/silvia/cffs-label/data/raw/OK 21-22 Sep-Dec/OK Grult Kitchen Break_Grill Kitchen Day_Grill Kitchen Features.oc']
```

Import Items List

```
In [6]: #Read items .xml files in the filepath_list and construct a dataframe
ItemId = []
Description = []
CaseQty = []
CaseQty = []
PakQty = []
PakU0M = []
InventoryGroup = []
for filepath in filepath_list:
    path = filepath + '/items.xml'
    if os.path.isfile(path):
        xtree = et.parse(path)
        xroot = xtree.getroot()
        for item in xtree.iterfind('Item'):
        ItemId.append(item.findtext('Description'))
        CaseQty.append(item.findtext('CaseQtY'))
        CaseQty.append(item.findtext('PakQtY'))
        PakQty.append(item.findtext('PakQtY'))
        PakQty.append(item.findtext('PakQtY'))
        PakQty.append(item.findtext('PakQtY'))
        PakQty.append(item.findtext('InventoryGroup'))
```

Items.reset_index(drop=True, inplace=True)

In [7]: Items

Qty Pak	PakQt	CaseUOM	CaseQty	Description	ItemId	
000	1.00	ea	1.000	5 SPICE POWDER	I-7631	0
500	2.50	LG CAN	6.000	ARTICHOKE 1/4 SALAD CUT TFC	I-4971	1
000	1.00	СТ	20.000	AVOCADO (20CT) MX	1-4473	2
000	454.00	bag	12.000	AVOCADO PULP CHUNKY	I-4973	3
000	1.00	lb	30.000	BAK CHOY BABY BC	I-4496	4
000	1000.00	Kg	10.000	YEAST B12 NUTRITIONAL BULK	I-29081	483
000	2.00	POUCH	4.000	YOGURT STRAWB POUCH	I-2171	484
000	1.00	TUB	1.000	YOGURT VANILLA STIRRED 650G	I-2281	485
000	1.00	lb	5.000	ZEST SUGARED LEMON	I-1489	486
000	1.00	lb	25.000	ZUCCHINI MED FCY MX	1-4967	487

488 rows × 7 columns

```
In [8]: Items.shape
```

Out[8]: (488, 7)

In [9]: Items.dtypes
Out[9]: ItemId
Description

1t[9]:	ItemId	object
	Description	object
	CaseQty	object
	CaseUOM	object
	PakQty	object
	PakUOM	object
	InventoryGroup dtype: object	object

In [10]: path = os.path.join(os.getcwd(), "data", "preprocessed", "Items_List.csv")
Items.to_csv(path, index = False, header = True)

Import Ingredients List

ouc[12]:		greatentia	ary	oom	Conversion	meractor	Recipe
	0	1-3388	1.000	L	1.00000000	0.3058	P-10496

	Ingredien	ld Qty	Uom	Conversion	InvFactor	Recipe			
1	1-46	60 2.270	Kg	2.20462000	0.6942	P-10496			
2		98 1.000	СТ	1.00000000	0.0013	P-12954			
3				1.00000000	0.0063	P-18318			
4	I-47	92 10.000	Kg	2.20462000	1.2048	P-18746			
3222				1.00000000		R-65038			
3223				1.00000000		R-65039			
3224				1.00000000		R-65040			
3225				1.00000000		R-65040			
3226	R-649	97 1.000	ea	1.00000000	1.0000	R-65042			
227	rows × 6 cc	lumns							
Ing	redients.s	hape							
(322	7,6)								
Ing	redients.c	ltypes							
Ingr Qty	edientId	object object							
Jom		object							
InvF	ersion actor	object object							
Reci dtyp	pe e: object	object							
), "data" = False,		essed", "Ing True)	edients_I	ist.csv")	
np	ort Prep	s List							
Prej Dese Pake Pake	pId = [] cription = Qty = [] UOM = [] entoryGrou	[]				onstruct a da			
	filepath	in filepa							
	<pre>path = fi if os.pat xtree xroot for > I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I I</pre>	h.isfile = et.pan = xtree : in xtree repId.app escriptic 'akQty.app 'akUOM.app nventory(<pre>(path): rse(path) getroot(e.iterfin pend(x.at on.append pend(x.fi gend(x.fi Group.app ['PrepId'</pre>) d('Prep') trib['id' (x.findte: ndtext('Pa end(x.find : PrepId,]) kt('Descri akQty')) akUOM')) dtext('Inv 'Descript	ption')) entoryGroup' ion': Descrip OM, 'Inventor	tion,	Inventory	roup}).d:
Prej	<pre>path = fi if os.pat xtree xroot for > F I I I I I I I I I I I I I S.pat xtree xroot for > S I I I S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S</pre>	h.isfile = et.pan = xtree : in xtree prepId.app tescriptic akQty.app akUOM.app nventory(taFrame(+ 'Pal	<pre>(path): rse(path): getroot(a.iterfin bend(x.at bend(x.fi Group.append ('PrepId' GQty': Patholic Scotter State Content State Content State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State</pre>) d('Prep') trib['id' (x.findte: ndtext('Pa end(x.find : PrepId,)) kt('Descri akQty')) akUOM')) itext('Inv 'Descript kUOM':PakU	entoryGroup' ion': Descrip	tion,	Inventory	roup}).d
Prej	<pre>path = fi if os.pat xtree xroot for > F F F p ps = pd.Da ps.reset_i</pre>	h.isfile = et.pan = xtree : in xtree prepId.app tescriptic akQty.app akUOM.app nventory(taFrame(+ 'Pal	<pre>(path): rse(path): getroot(a.iterfin bend(x.at bend(x.fi Group.append ('PrepId' GQty': Patholic Scotter State Content State Content State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State</pre>) d('Prep') trib['id' (x.findte: ndtext('Pa end(x.find : PrepId, kQty, 'Pal)) kt('Descri akQty')) akUOM')) itext('Inv 'Descript kUOM':PakU	entoryGroup' ion': Descrip	tion,	Inventory	;roup}).d:
Prej	<pre>path = fi if os.pat xtree xroot for > F F F p ps = pd.Da ps.reset_i</pre>	h.isfile = et.pan = xtree : in xtree prepId.app tescriptic akQty.app akUOM.app nventory(taFrame(+ 'Pal	<pre>(path): rse(path): getroot(a.iterfin bend(x.at bend(x.fi Group.append ('PrepId' GQty': Patholic Scotter State Content State Content State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State State</pre>) d('Prep') trib['id' (x.findte: ndtext('Pr ndtext('Pr end(x.find : PrepId, kQty, 'Pal nplace=Tru	<pre>)) tt('Descri kQty')) akUOM')) itext('Inv 'Descript kUOM':PakU ae)</pre>	entoryGroup' ion': Descrip	tion, yGroup':	Inventory	;roup}).d:
Prej Prej	<pre>path = fi if os.pat xtree xroot for > F F F ps = pd.Da ps.reset_i ps</pre>	h isfile = et.pan. = xtree. in xtree. repId.app eccriptic akty.app nventory(taFrame(i 'Pal ndex(drop	<pre>(path): se(path): getroot(.iterfin pend(x.at pend(x.fi pend(x.fi Group.app ('PrepId' QQty': Pa p=True, i</pre>) d('Prep') trib['id' t(x.findte: ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext('P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndtext(P; ndt	<pre>)) tt('Descrit akQty')) akUOM')) itext('Inv 'Descript cUOM':PakU ae) ty PakUOM</pre>	entoryGroup' ion': Descrip OM, 'Inventor InventoryGrou	tion, yGroup': p	Inventory	;roup}).d;
Prej Prej Prej 0	<pre>path = fi if os.pat xtree xroot for > F F F ps = pd.Da ps.reset_i ps PrepId</pre>	h isfile(= et.pan = et.pan = xtree t in xtree trepId.app escriptic akUOM.app nventory('Pal ndex(drop ndex(drop	<pre>(path): se(path); getroot, .iterfir pend(x.at pend(x.fi froup.app ('PrepId' coty': Pa p=True, i Descrip</pre>) d('Prep') trib['id' u(x.findte: ndtext('Pi end(x.find : PrepId, kQty, 'Pal nplace=Tru tion PakQ mole 2.75	<pre>)) tt('Descri akQty')) akUOM')) itext('Inv 'Descript cUOM':PakU ae) ty PakUOM S0 Kg</pre>	entoryGroup' ion': Descrip OM, 'Inventor MnventoryGrou	tion, yGroup': p	Inventory	;roup}).d:
Prej Prej Prej 0	<pre>path = fi if os.pat xtree xroot for > F F ps = pd.Da ps.reset_i ps Prepid P-56398</pre>	h isfile(= et.par. = xtree t in xtree repId.app lescriptic akUOM.app nventory('Pal ndex(drop BA' BA'	<pre>(path): se(path); getroot (.iterfin pend(x.at pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi perpid); perpid (opy): pa perrue, i Descrip TCH Guaca</pre>) d('Prep') trib['id' (x.findtext('Pi ndtext('Pi end(x.find : PrepId, kQty, 'Pal nplace=Tru tion PakQ mole 2.75 intro 0.50	<pre>)) tt('Descrit kQty')) akUOM')) itext('Inv 'Descript kUOM':PakUOM' ie) ty PakUOM iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii</pre>	entoryGroup' ion': Descrip OM, 'Inventor InventoryGrou PRE	p p	Inventory	;roup}).d:
Prej Prej Prej 0 1 2	<pre>path = fi if os.pat</pre>	h.isfile(= et.par. = xtree = xtree tin xtree repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repI	(path): rse(path); rse(path); rse(path); rse(path); rse(path); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath)) d('Prep') trib['id' u(x.findte: ndtext('Pr ndtext('Pr end(x.find : PrepId, kQty, 'Pal nplace=Tru tion PakQ mole 2.75 intro 0.50 eans 30.000 nion 1.20	<pre>)) ct('Descrit kQty')) akUOM')) itext('Inv 'Descript kUOM':PakU ae) cy PakUOM 60 Kg 60 Kg 60 Kg 60 Kg</pre>	entoryGroup' ion': Descrip OM, 'Inventor InventoryGrou PRE PRE	p p p	Inventory	;roup}).d:
Prej Prej 0 1 2 3	<pre>path = fi if os.pat</pre>	h.isfile(= et.par. = xtree = xtree tin xtree repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repId.app repI	(path): rse(path); rse(path); rse(path); pend(x.at pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi pend(x.fi) pend(x.fi pend(x.fi) pend(x.fi pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.fi) pend(x.f) d('Prep') trib['id' u(x.findte: ndtext('Pr ndtext('Pr end(x.find : PrepId, kQty, 'Pal nplace=Tru tion PakQ mole 2.75 intro 0.50 eans 30.000 nion 1.20	<pre>)) ct('Descrit kQty')) akUOM')) itext('Inv 'Descript kUOM':PakUOM' 'Descript kUOM':PakUOM 'Descript kUOM':RakUOM' 'Descript kUOM':PakUOM 'Descript 'Des</pre>	entoryGroup' ion': Descrip OM, 'Inventor InventoryGrou PRE PRE	p p p	Inventory	;roup}).d:
Prej Prej Prej 0 1 2 3 4 	path = fi if os.pat xroot for > for > ps = pd.Da ps.reset_i ps.reset_i P-56398 P-24750 P-41574 P-26068 of P-28258 	h.isfile(= et.pan = xtree tin xtree repId.app rescriptic akQty.app akUOM.app nventory(taFrame('Pal ndex(drop BA CH COOKED Can COO	(path): rse(path); rse(path); rse(path); rse(path); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attring); rse(attri) d('Prep') trib['id' trib['id' t(x.findte: ndtext('P, ndtext('P, ndtext('P, ndtext('P, kQty, 'Pal nplace=Tri tion PakQ mole 2.76 intro 0.50 eans 30.00 nion 1.20 Wein 48.00 	<pre>)) ct('Descript akQty')) akQty')) itext('Inv 'Descript cuoM':PakUOM ie) ty PakUOM io0 Kg i00 K</pre>	entoryGroup' ion': Descrip OM, 'Inventor InventoryGrou PRE PRE PRE	p p p	Inventory	:roup}).d:
Prej Prej Prej 0 1 2 3 4	path = fi if os.pat xroot for > for > ps = pd.Da ps.reset_i ps = Prepid P-56398 P-24750 P-41574 P-26068 of P-28258	h.isfile(= et.pan = xtree tin xtree repId.app rescriptic akQty.app akUOM.app nventory(taFrame('Pal ndex(drop BA CH COOKED Can COO	(path): rse(path); rse(path); rse(path); rse(path); rse(path); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath); rse(ath)) d('Prep') trib['id' trib['id' t(x.findte: ndtext('P, ndtext('P, ndtext('P, ndtext('P, kQty, 'Pal nplace=Tri tion PakQ mole 2.76 intro 0.50 eans 30.00 nion 1.20 Wein 48.00 	<pre>)) ct('Descript akQty')) akQty')) itext('Inv 'Descript cuoM':PakUOM ie) ty PakUOM io0 Kg i00 K</pre>	entoryGroup' ion': Descrip OM, 'Inventor InventoryGrou PRE PRE PRE	tion, yGroup': p p p p	Inventory	;roup}).d:
Prej Prej 0 1 2 3 4 488	path = fi if os.pat xroot for > for > ps = pd.Da ps.reset_i ps.reset_i P-56398 P-24750 P-41574 P-26068 of P-28258 	h isfile(= et.pan: = et.pan: = xtree t in xtree t in xtree trepId.app escriptic akUOM.app nventory(taFrame ('Pa) ndex(drop BA CH COOK COOKED[Can COO YIEI	(path): se(path): getroot (siterfin bend (x.at bend (x.at bend (x.fi froup.app ('PrepId' Quy': Pathological perrue, i Descrip TCH Guadal IOPPED Clife ED Black Ber ramelized O KED Chow I LD Smokie (YIELD Thai I) d('Prep') trib['id' trib['id' t(x.findte: ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr ndtext('Pr	<pre>)) tt('Descript kQty')) kkUOM')) ltext('Inv 'Descript kUOM':PakUOM ie) ty PakUOM iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii</pre>	entoryGroup' ion': Descrip OM, 'Inventor InventoryGrou PRE PRE PRE	tion, yGroup': p p p p	Inventory	;roup}).d:

	-	Prepid	Description	PakQty	PakUOM	InventoryGroup	
	491	P-46833	YIELD Yam Fries	800.000	g		
	492	P-57145	YIELD Yellow Pepper	8.300	Kg		
	493 r	ows × 5 co	lumns				
In [18]:	Pre	ps.shape					
Out[18]:							
Ouc[10]:	(455	, ,,					
In [19]:	Pre	ps.dtypes					
Out[19]:	Desc PakQ PakU Inve	ription ty					
In [20]:			th.join(os.getcwd(), " (path, index = False,			essed", "Preps_	List.csv")
	Impo	ort Proc	lucts List				
In [21]:	Proc Desc Sale	<pre>dId = [] cription esGroup = filepath path = f if os.pa xtre xroo for</pre>	<pre>[] in filepath_list: ilepath + '/Products.x th.isfile(path): t = et.parse(path) t = xtree.getroot() x in xtree.iterfind('P ProdId.append(x.attrib Description.append(x.fi SalesGroup.append(x.fi</pre>	<pre>ml' rod'): ['id']) indtext(' </pre>	('Descrip 'SalesGro	<pre>ption')) pup'))</pre>	
			d.DataFrame({'ProdId': et_index(drop=True, in			iption': Descri	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
In [22]:	Proc					iption': Descri	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
In [22]: Out[22]:	Proc	ducts.res		place =T		lption': Descri SalesGroup	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
	Proc	ducts.res ducts	et_index(drop= True , in	place -T r	rue)		<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
	Prod Prod	ducts.res ducts Prodid	et_index(drop=True, in Descriptio	place=Tr n s	rue) OK - CU	SalesGroup	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
	Proc Proc	ducts.res ducts Prodld R-30154 R-56337	et_index(drop= True , in Descriptio ADD Cracker	place= T r n s n	сие) ОК - СU	SalesGroup STOM KITCHEN	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
	Prod Prod 0 1 2	ducts.res ducts Prodld R-30154 R-56337	et_index(drop=True, in Descriptio ADD Cracker ALF Flatbread Mediterranea	place=T: n s n	cue) OK - CU: (SalesGroup STOM KITCHEN OK - AL FORNO	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
	Prod Prod 0 1 2 3	ducts.res ducts Prodid R-30154 R-56337 R-61779	et_index(drop=True, in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest	place= T n s n o	сие) ОК - СU ((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
	Prod Prod 0 1 2 3	ducts.res ducts Prodid R-30154 R-56337 R-61779 R-50590	et_index(drop=True, in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest ALF Flatbread O	place= T n s n o	сие) ОК - СU ((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO OK - AL FORNO	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
	Proc Proc 0 1 2 3 4 316	ducts.res ducts ProdId R-30154 R-56337 R-61779 R-50590 R-50494 R-64095	et_index(drop= True , in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest ALF Flatbread Proscuitt ALF Flatbread Proscuitt THANKSGIVING ONION GRAV	place=T1 n s n o K o o X	Crue) OK - CUS (((((((((((((((((((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO OK - AL FORNO OK - AL FORNO SQUARE MEAL	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
	Proc Proc 0 1 2 3 4 316	ducts.res ducts ProdId R-30154 R-56337 R-61779 R-50590 R-50494 R-64095	et_index(drop= True , in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest ALF Flatbread Proscuitt ALF Flatbread Proscuitt	place=T1 n s n o K o o X	Crue) OK - CUS (((((((((((((((((((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO OK - AL FORNO OK - AL FORNO SQUARE MEAL SQUARE MEAL	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
	Proc.	ducts.res ducts Prodld R-30154 R-56337 R-61779 R-50590 R-50494 R-64095 R-30673 R-30673	et_index(drop=True, in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest ALF Flatbread Oroscuitt ALF Flatbread Proscuitt THANKSGIVING ONION GRAV THANKSGIVING PUMPKIN PI VEG Bowl Polent	place=Tr n s s n o K o o Y E a	Crue) OK - CUS (((((((((((((((((((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO OK - AL FORNO OK - AL FORNO SQUARE MEAL SQUARE MEAL OK - SIDES	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
	Proc Proc 0 1 2 3 4 316 317 318 319	ducts.res ducts Proddd R-30154 R-56337 R-61779 R-50590 R-50494 R-64095 R-30673 R-35341 R-35637	et_index(drop=True, in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest ALF Flatbread Oroscuitt ALF Flatbread Proscuitt THANKSGIVING ONION GRAV THANKSGIVING PUMPKIN PI VEG Bowl Polent VEG French Toast Eggno	place=Tr n s n o K o o Y E a g	CWC) OK - CUS (((((((((((((((((((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO OK - AL FORNO OK - AL FORNO SQUARE MEAL SQUARE MEAL OK - SIDES OK - SIDES	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
	Proc Proc 1 2 3 4 316 317 318 319 320	ducts.res ducts Prodld R-30154 R-56337 R-61779 R-50590 R-50494 R-64095 R-30673 R-30673	et_index(drop=True, in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest ALF Flatbread Proscuitt ALF Flatbread Proscuitt THANKSGIVING ONION GRAV THANKSGIVING PUMPKIN PI VEG BowI Polent VEG French Toast Eggno VEG French Toast Corn Flak	place=Tr n s n o K o o Y E a g	CWC) OK - CUS (((((((((((((((((((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO OK - AL FORNO OK - AL FORNO SQUARE MEAL SQUARE MEAL OK - SIDES OK - SIDES	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates()</pre>
Out[22]:	Prod Prod 1 2 3 4 316 317 318 319 320 321 rd	ducts.res ducts ProdId R-30154 R-56337 R-61779 R-50590 R-50494 R-64095 R-36373 R-36341 R-56637 R-56451 ows × 3 co	et_index(drop=True, in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest ALF Flatbread Proscuitt ALF Flatbread Proscuitt THANKSGIVING ONION GRAV THANKSGIVING ONION GRAV THANKSGIVING PUMPKIN PI VEG Bow Polent VEG French Toast Eggno VEG FrenchToast Corn Flak	place=Tr n s n o K o o Y E a g	CWC) OK - CUS (((((((((((((((((((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO OK - AL FORNO OK - AL FORNO SQUARE MEAL SQUARE MEAL OK - SIDES OK - SIDES	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates() </pre>
	Prod Prod 1 2 3 4 316 317 318 319 320 321 rd	ducts.res ducts ProdId R-30154 R-56337 R-61779 R-50590 R-50494 R-64095 R-30673 R-35341 R-56637 R-56451	et_index(drop=True, in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest ALF Flatbread Proscuitt ALF Flatbread Proscuitt THANKSGIVING ONION GRAV THANKSGIVING ONION GRAV THANKSGIVING PUMPKIN PI VEG Bow Polent VEG French Toast Eggno VEG FrenchToast Corn Flak	place=Tr n s n o K o o Y E a g	CWC) OK - CUS (((((((((((((((((((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO OK - AL FORNO OK - AL FORNO SQUARE MEAL SQUARE MEAL OK - SIDES OK - SIDES	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates() </pre>
Out[22]:	Proc Proc 0 1 2 3 4 316 317 318 319 320 321 rc	ducts.res ducts Prodld R-30154 R-56337 R-61779 R-50590 R-50494 R-64095 R-30673 R-35341 R-35341 R-56451 ows × 3 co ducts.sha	et_index(drop=True, in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest ALF Flatbread Proscuitt ALF Flatbread Proscuitt THANKSGIVING ONION GRAV THANKSGIVING ONION GRAV THANKSGIVING PUMPKIN PI VEG Bow Polent VEG French Toast Eggno VEG FrenchToast Corn Flak	place=Tr n s n o K o o Y E a g	CWC) OK - CUS (((((((((((((((((((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO OK - AL FORNO OK - AL FORNO SQUARE MEAL SQUARE MEAL OK - SIDES OK - SIDES	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates() </pre>
Out[22]: In [23]:	Proc Proc 0 1 2 3 4 3 16 3 17 3 18 3 19 3 20 321 r (2 2 2 2 1 2 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 1 3 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1	ducts.res ducts Prodld R-30154 R-56337 R-61779 R-50590 R-50494 R-64095 R-30673 R-35341 R-35341 R-56451 ows × 3 co ducts.sha	et_index(drop=True, in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest ALF Flatbread Mushroom Pest ALF Flatbread Proscuitt ALF Flatbread Proscuitt THANKSGIVING ONION GRAV THANKSGIVING ONION GRAV THANKSGIVING ONION GRAV UEG French Toast Eggno VEG French Toast Corn Flak	place=Tr n s n o K o o Y E a g	CWC) OK - CUS (((((((((((((((((((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO OK - AL FORNO OK - AL FORNO SQUARE MEAL SQUARE MEAL OK - SIDES OK - SIDES	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates() </pre>
Out[22]: In [23]: Out[23]:	Prod Prod 0 1 2 3 4 316 317 318 319 320 321 rc (321 Prod Prod Prod	ducts.res ducts Prodld R-30154 R-56337 R-61779 R-50590 R-50494 R-64095 R-30673 R-35341 R-56637 R-35341 R-56637 R-56451 oux × 3 co ducts.sha , 3) ducts.dty	et_index(drop= True , in Descriptio ADD Cracker ALF Flatbread Mediterranea ALF Flatbread Mushroom Pest ALF Flatbread Mushroom Pest ALF Flatbread Proscuitt CALF Flatbread Proscuitt THANKSGIVING ONION GRAV THANKSGIVING PUMPKIN PI VEG French Toast Eggno VEG French Toast Corn Flak lumns pe pes object object	place=Tr n s n o K o o Y E a g	CWE) OK - CUS (((((((((((((((((((SalesGroup STOM KITCHEN OK - AL FORNO OK - AL FORNO OK - AL FORNO OK - AL FORNO SQUARE MEAL SQUARE MEAL OK - SIDES OK - SIDES	<pre>ption, 'SalesGroup': SalesGroup}).drop_duplicates() </pre>

Import Conversions List

```
In [26]: #Read conventions .xml files in the filepath_list and construct a dataframe
             ConversionId = []
             Multiplier = []
            ConvertFromQty = []
ConvertFromUom = []
             ConvertToQty = []
ConvertToUom = []
             for filepath in filepath_list:
    path = filepath + '/Conversions.xml'
                  if os.path.isfile(path):
                       stree = et.parse(path)
xroot = xtree.getroot()
for x in xtree.iterfind('Conversion'):
                            ConversionId.append(x.attrib['id'])
Multiplier.append(x.attrib['multiplier'])
                            ConvertFromQty.append(x.find('ConvertFrom').attrib['qty'])
ConvertFromUom.append(x.find('ConvertFrom').attrib['uom'])
ConvertToQty.append(x.find('ConvertTo').attrib['qty'])
                            ConvertToUom.append(x.find('ConvertTo').attrib['uom'])
             Conversions = pd.DataFrame({'ConversionId': ConversionId, 'Multiplier': Multiplier, 'ConvertFromQty': ConvertFromQty,
                                                ConvertFromUom': ConvertFromUom, 'ConvertToQty': ConvertToQty, 'ConvertToUom': ConvertToUom}
                                              ).drop_duplicates()
             Conversions.reset_index(drop=True, inplace=True)
In [27]: Conversions
                  ConversionId
                                  Multiplier ConvertFromQty ConvertFromUom ConvertToQty ConvertToUom
               0
                                 1.00000000
                                                         1.0000
                                                                              XXX
                                                                                            1.0000
                                                                                                                 L
                                 0.87719298
                                                         1.0000
                                                                              1.14L
                                                                                                                  L
           1
                                                                                            1.1400
               2
                                 0.66666667
                                                         1.0000
                                                                              1.5L
                                                                                            1.5000
                                                                                                                  L
                                 0.57142857
                                                         1.0000
                                                                             1.75 L
                                                                                            1.7500
              3
                                                                                                                  L
              4
                                 0.50000000
                                                         10000
                                                                                21
                                                                                            2 0000
                                                                                                                 T.
             ...
                             ...
            265
                       I-25492 0.00495050
                                                         1.0000
                                                                                ea
                                                                                         202.0000
                                                                                                                 g
            266
                        I-27407 0.01333333
                                                         1.0000
                                                                                          75.0000
                                                                                ea
                                                                                                                 g
            267
                       I-43559 0.02000000
                                                         1.0000
                                                                                CT
                                                                                          50.0000
                                                                                                                  g
            268
                       I-47525 0.00408163
                                                         1.0000
                                                                               cup
                                                                                         245.0000
                                                                                                                  g
            269
                       I-63034 0.01098901
                                                         1.0000
                                                                                СТ
                                                                                          91.0000
                                                                                                                  g
           270 rows × 6 columns
In [28]: Conversions.shape
Out[28]: (270, 6)
In [29]: Conversions.dtypes
Out[29]: ConversionId
                                   object
            Multiplier
                                   object
            ConvertFromOty
                                   object
                                  object
object
            ConvertFromUom
            ConvertToQty
            ConvertToUom
                                  object
            dtype: object
            path = os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")
Conversions.to_csv(path, index = False, header = True)
In [30]:
```

Data Summary

Out[31]:

	count	columns
Items	488	7
Preps	493	5
Ingredients	3227	6
Products	321	3
Conversions	270	6

Climate-Friendly Food Systems (CFFS) Labelling Project

The University of British Columbia

Created by Silvia Huang

Part II: Data Cleaning

Set up and Import Libraries

In [1]:	<pre>#install libraries if needed #!pip3 install pdpipe</pre>
In [2]:	<pre>import numpy as np import pandas as pd import pdpipe as pdp import matplotlib.pyplot as plt import glob import os import osv from itercols import islice from decimal import Decimal import xml.etree.ElementTree as et from xml.etree.ElementTree import parse import openpyxl import pytest from datetime import datetime</pre>
In [3]:	<pre>#set the root path, change the directory into the project folder os.chdir("/Users/silvia/cffs-label")</pre>
In [4]:	<pre>#enable reading data in the scrolling window #pd.set_option("display.max_rows", None, "display.max_columns", None)</pre>

Import Preprocessed Datasets

		object object float64 object float64 object object						
Items.	head()							
Item	d	Description	CaseQty	CaseUOM	PakQty	PakUOM	InventoryGroup	
0 I-76	31	5 SPICE POWDER	1.0	ea	1.0	lb	SPICES	
1 I-49	71 ARTICH	OKE 1/4 SALAD CUT TFC	6.0	LG CAN	2.5	Kg	PRODUCE	
2 1-447	'3	AVOCADO (20CT) MX	20.0	СТ	1.0	HEAD	PRODUCE	
3 1-497	'3 A	VOCADO PULP CHUNKY	12.0	bag	454.0	g	PRODUCE	

Out[7]: (488, 7)

In [8]: #read Ingredients_List.csv
Ingredients = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Ingredients_List.csv"))
Ingredients.dtypes Out[8]: IngredientId object Qty float64 Uom object Conversion float64

$ \begin{aligned} \begin{aligned} \begin{aligned} \begin{aligned} \begin{aligned} \begin{aligned} \begin{aligned} & f = 0 \\ \\ & f = 0 \\ & f = 0 \\ \\ & f$		InvFactor float64 Recipe object dtype: object
0 i i 388 100 L 100000 0.0038 P-10486 1 i 4680 100 C 0.0003 P-12816 2 i 4679 100 0.003 P-18386 3 i 4679 100 0.003 P-18386 4 i 4792 1000 Kg 220462 12948 P-18376 10110 Topredients-shape	In [9]:	Ingredients.head()
0 i i 388 100 L 100000 0.0038 P-10486 1 i 4680 100 C 0.0003 P-12816 2 i 4679 100 0.003 P-18386 3 i 4679 100 0.003 P-18386 4 i 4792 1000 Kg 220462 12948 P-18376 10110 Topredients-shape	Out[9]:	IngredientId Qty Uom Conversion InvFactor Recipe
2 i 4688 100 CT 100000 0003 P-12954 3 i 4792 100 BUNCH 100000 0005 P-18356 7: [10] Interdient.nenpe 100000 0005 P-18356 7: [10] Food Prop. List.cor Prop. Interdient.nenpe 7: [10] Food Prop. List.cor Prop. Interdient.nenpe 7: [10] Prop. Interdient.nenpe Interdient.nenpe 7: [10] Prop. Interdient.nenpe Prop. Interdient.nenpe 7: [10] Prop. Interdient.nenpe Interdient.nenpe 7: [10] Prop. Interdient.nenpe Interdient.nenpe 7: [10] Prop. Interdient.nenpe Interdient.nenpe		
$ \begin{array}{c} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 $		1 I-4660 2.27 Kg 2.20462 0.6942 P-10496
4 1.422 100 g 2242 1204 P4740		2 I-4598 1.00 CT 1.00000 0.0013 P-12954
In [10] Instant and the set of the se		3 I-4679 1.00 BUNCH 1.00000 0.0063 P-18318
0:vel10: (227, 6) 1n [11: Freed Preps_List.csv Preps = pl.:exd_csv(cs.path.join(cs.gstcwd()), "dsts", "preprocessed", "Preps_List.csv")) 0:vel11: Discription object Packyr floatel 0:vel12: Preps_Led() 0:vel12: Prepd 0:vel22: Prepd 0:vel23: COOKED[Clasmedized loon 226 0:vel23: COOKED[Clasmedized loon 226 11:11: Preps.hadp 0:vel23: COOKED[Clasmedized loon 226 11:12: Preps.hadp 0:vel23: roddette = pl.read active vel2 setter vel2 sett		4 I-4792 10.00 Kg 2.20462 1.2048 P-18746
In [11] Press Press List.csv Press Press List.csv Object Cut[11] Press List.csv Press List.csv Object Press List.csv Object Press List.csv Disct Press List.csv Press List.csv Press List.csv Disct Press List.csv Press List.csv Press List.csv Disct Press List.csv Disct Press List.csv <th>In [10]:</th> <th>Ingredients.shape</th>	In [10]:	Ingredients.shape
Props pulsed_cev(cs.pabl.join(cs.getoxd(), "dsta", "proprocessed", "Props_List.cev")) Cut[11] Propid Object PARCY Float Object PARCY Float Object Parcy Float Object Parcy Props_head() Immentor/Group Out[12] Props_head() Immentor/Group 0 P-66398 BATCHQuaramole 2750 Kg PREP 2 P-41574 COOKED[Daramelzed Onion 2500 Kg PREP 3 P-26068 COOKED[Caramelzed Onion 1200 Kg PREP 1n [413] Preps_chape Object PREP PREP 1n [413] Preps_chape Object PREP 1n [413] Product_List.csv Product_List.csv PREP 1n [414] Product_List.csv Product_List.csv Product_List.csv 1n Product_List.csv Object SaleGroup Product_List.csv 1n Product_List.csv	Out[10]:	(3227, 6)
Description itypes object itypes ob	In [11]:	<pre>Preps = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Preps_List.csv"))</pre>
Perplate Description PakUV Nave 0 P-56398 BATCH[Guacamole 2.750 Kg PREP 1 P-24750 CHOPED[Cillante 0.500 Kg Nave 2 P-4574 COOKED[Coramelized Onion 1.200 Kg PREP 3 P-26068 COOKED[Coramelized Onion 1.200 Kg PREP 4 P-26258 COOKED[Coramelized Onion 1.200 Kg PREP 0xt131: (43) 5 COOKED[Coramelized Onion 1.200 Kg PREP 0xt131: (43) 5 COOKED[Coramelized Onion 1.200 Kg PREP 0xt131: (43) 5 COOKED[Coramelized Onion 1.200 Kg PREP 0xt131: Freed Product List.csv Product science Onion Science Onion Science Onion 0xt131: FredId Description Science Onion Science Onion 0xt131: FredId Description Science Onion Science Onion	Out[11]:	Description object PakQty float64 PakUOM object InventoryGroup object
Product Product Object SalesGroup 1 Product ADD[Crackers OK - AL FORNO 1 Product Object SalesGroup 2 Product ADD[Crackers OK - AL FORNO 3 R-50590 ALF[Flatbread]Mediterranea OK - AL FORNO 3 R-50590 ALF[Flatbread]Mediterranea OK - AL FORNO 3 R-50590 ALF[Flatbread]Mediterranea OK - AL FORNO 3 R-50594 ALF[Flatbread]Proscuito OK -	In [12]:	Preps.head()
Product Product Object SalesGroup 1 Product ADD[Crackers OK - AL FORNO 1 Product Object SalesGroup 2 Product ADD[Crackers OK - AL FORNO 3 R-50590 ALF[Flatbread]Mediterranea OK - AL FORNO 3 R-50590 ALF[Flatbread]Mediterranea OK - AL FORNO 3 R-50590 ALF[Flatbread]Mediterranea OK - AL FORNO 3 R-50594 ALF[Flatbread]Proscuito OK -	Out[121:	PrepId Description PakQty PakUOM InventoryGroup
2 P-41574 COOKED[Black Beams 30.000 Kg PREP 3 P-26068 COOKED[Chow Mein 48.081 Kg PREP 4 P-28258 COOKED[Chow Mein 48.081 Kg PREP 1 1 Preps.shape		
3 P-26068 COOKED/Chamelized Onion 1.200 Kg PREP 4 P-28258 COOKED/Chow Mein 48.081 Kg PREP In (13); Preps.shape		
4 P-28258 COOKED[Chow Mein 48.081 Kg PREP in [13] Preps.shape		2 P-41574 COOKED Black Beans 30.000 Kg PREP
In [13] Preps.shape Out[13] (493, 5) In [14] <i>fread Product_list.csv</i> Products = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Products_List.csv")) Products = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Products_List.csv")) Out[14] Frodut Brodut object Brodut bject Out[15] Products.head() Out[16] Products.head() Out[17] Products.head() Out[18] Product.head() Out[19] Products.head() Out[19] Product.head() Out[19] Product.head() Out[19] Product.head() Out[19] Product.head() Out[19] Product.head() Out[19] Product.head() A R-6037 ALF[Flatbread]Moditeranean OK - AL FORNO A R-6030 A R-5044 ALF[Flatbread]Proscutto Out[19] Products.shape Out[19] Conversions.dtype Out[19] Conversions.dtype Out[10] Conversions.dtype Out[10]		3 P-26068 COOKEDJCaramelized Onion 1.200 Kg PREP
Out[13]: (493, 5) In [14]: fread Toroduct_List.csv Products = pl.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Products_List.csv")) Products.dtypes Out[14]: Prodid object bescription object dtype: object In [15]: Products.head() Out[15]: Prodid Description SalesGroup 0 R-30154 ADD[Crackers OK - CUSTOM KHTCHEN 1 R-56337 ALF[Flatbread]Mushroom Pesto OK - AL FORNO 2 R-61779 ALF[Flatbread]Mushroom Pesto OK - AL FORNO 3 R-50590 ALF[Flatbread]Mushroom Pesto OK - AL FORNO 4 R-50494 ÀLF[Flatbread]Proscuitto OK - AL FORNO 4 R-50494 ÀLF[Flatbread]Proscuitto OK - AL FORNO 5 In [16]: In [16]: products.shape Out[16]: (321, 3) In [17]: Conversions = pl.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")) Out[17]: Conversion d bject float64 converstromgty float64 convertromgty float64 convertromgty float64		4 P-28258 COOKED Chow Mein 48.081 Kg PREP
Out[13]: (493, 5) In [14]: fread Toroduct_List.csv Products = pl.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Products_List.csv")) Products.dtypes Out[14]: Prodid object bescription object dtype: object In [15]: Products.head() Out[15]: Prodid Description SalesGroup 0 R-30154 ADD[Crackers OK - CUSTOM KHTCHEN 1 R-56337 ALF[Flatbread]Mushroom Pesto OK - AL FORNO 2 R-61779 ALF[Flatbread]Mushroom Pesto OK - AL FORNO 3 R-50590 ALF[Flatbread]Mushroom Pesto OK - AL FORNO 4 R-50494 ÀLF[Flatbread]Proscuitto OK - AL FORNO 4 R-50494 ÀLF[Flatbread]Proscuitto OK - AL FORNO 5 In [16]: In [16]: products.shape Out[16]: (321, 3) In [17]: Conversions = pl.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")) Out[17]: Conversion d bject float64 converstromgty float64 convertromgty float64 convertromgty float64	Tn [13]:	Preps, shape
<pre>In [14]: fread Product_List.csv Products = pd.read_csv(os.path.join(os.getowd(), "data", "preprocessed", "Products_List.csv")) Products.dtypes Out[14]: Produt object SalesGroup object dtype: object In [15]: Produts.head() Out[15]: Produts.head() Out[15]: Produts.head() Out[16]: 0.8-30154 ADD[Crackers OK-CUSTOM KITCHEN 1 R-65337 ALF Flatbread Mushroom Pesto OK - AL FORNO 2 R-61779 ALF Flatbread Mushroom Pesto OK - AL FORNO 3 R-50590 ALF Flatbread Mushroom Pesto OK - AL FORNO 4 R-50494 ALF Flatbread OK OK - AL FORNO In [16]: (321, 3) In [17]: Conversions = pd.read_csv(os.path.join(os.getovd(), "data", "preprocessed", "Conversions_List.csv")) Out[15]: Conversion dbject Out[16]: Conversion dbject ConvertFromQty float64 ConvertFromQty float64</pre>		
Products = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Products_List.csv")) Out[14]: ProdId object Description object SalesGroup object In [15]: Products.head() Out[15]: ProdId Description SalesGroup O R-30154 ADD[Crackers OK-CUSTOM KITCHEN 1 R-56337 ALF]Flatbread]Mediterranean OK - AL FORNO 2 R-61779 ALF]Flatbread]Mediterranean OK - AL FORNO 2 R-61779 ALF]Flatbread]Mushroom Pesto OK - AL FORNO 0 K - AL FORNO 4 R-50494 ALF]Flatbread]OK OK - AL FORNO 0 K - AL FORNO In [16]: Products.shape Out[16]: (321, 3) In [17]: Conversions = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")) Out[17]: Conversions.dtypes	Out[13]:	(475, 5)
Description object SalesGroup object fn [15]: Products.head() Out[15]: Products.head() 0 R-30154 ADD[Crackers OK - CUSTOM KITCHEN 1 R-56337 ALF[Flatbread]Mediterranean 0 R-30154 ADD[Crackers OK - CUSTOM KITCHEN 1 R-56337 ALF[Flatbread]Mediterranean 0 R-30154 ADD[Crackers OK - CUSTOM KITCHEN 1 R-56337 ALF[Flatbread]Mediterranean 0 R-30154 ADD[Crackers OK - AL FORNO 2 R-61779 ALF[Flatbread]Mushroom Pesto OK - AL FORNO 3 R-50590 ALF[Flatbread]Proscuitto OK - AL FORNO 1n [16]: Products.shape Out[13]: Out[13]: (321, 3) In [17]: Conversions.dtypes Out[17]: Conversiondcov(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")) Out[17]: Conversionid object Multiplier float64 Converstrondtom object	In [14]:	<pre>Products = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Products_List.csv"))</pre>
Prodld Description SalesGroup 0 R-30154 ADD[Crackers OK-CUSTOM KITCHEN 1 R-56337 ALFIFlatbread]Mediterranean OK - AL FORNO 2 R-61779 ALFIFlatbread]Mushroom Pesto OK - AL FORNO 3 R-50590 ALFIFlatbread]OK OK - AL FORNO 4 R-50494 ALFIFlatbread]Proscuitto OK - AL FORNO In [16]: Products.shape Out[16]: (321, 3) In [17]: Conversions = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")) Out[17]: ConversionId object Out[17]: ConvertFromQty float64 ConvertFromQty float64 ConvertFromQty float64 ConvertFromQty float64	Out[14]:	Description object SalesGroup object
0 R-30154 ADD Crackers OK - CUSTOM KITCHEN 1 R-56337 ALF Flatbread Mushroom Pesto OK - AL FORNO 2 R-61779 ALF Flatbread Mushroom Pesto OK - AL FORNO 3 R-50590 ALF Flatbread OK OK - AL FORNO 4 R-50494 ALF Flatbread Proscuitto OK - AL FORNO 1 161: Products.shape	In [15]:	Products.head()
$\begin{bmatrix} 1 & R-56337 & ALF Flatbread Mediterranean & OK - AL FORNO \\ 2 & R-61779 & ALF Flatbread Mushroom Pesto & OK - AL FORNO \\ 3 & R-50590 & ALF Flatbread OK & OK - AL FORNO \\ 4 & R-50494 & ALF Flatbread Proscuitto & OK - AL FORNO \\ \hline \\ 1n [16]: \\ \hline \\ rotucts.shape \\ \hline \\ out[16]: \\ (321, 3) \\ \hline \\ 1n [17]: \\ \hline \\ Conversions.dtypes \\ \hline \\ Out[17]: \\ \hline \\ ConversionId & object \\ float64 \\ convertFromUcy & float64 \\ convertFromUcy & object \\ \hline \\ \end{array}$	Out[15]:	ProdId Description SalesGroup
2 R-61779 ALFIFIatbread Mushroom Pesto OK - AL FORNO 3 R-50590 ALFIFIatbread OK OK - AL FORNO 4 R-50494 ALFIFIatbread Proscuitto OK - AL FORNO In [16]: Products.shape In [17]: Conversions = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")) Conversions.dtypes Out[17]: ConversionId object float64 ConvertFromUcm object		0 R-30154 ADDICrackers OK - CUSTOM KITCHEN
3 R-50590 ALF Flatbread OK OK - AL FORNO 4 R-50494 ALF Flatbread Proscuitto OK - AL FORNO In [16]: Products.shape OK - AL FORNO Out[16]: (321, 3) In [17]: Conversions = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")) Out[17]: ConversionId object Multiplier float64 ConvertFromUw object		1 R-56337 ALF/Flatbread/Mediterranean OK - AL FORNO
4 R-50494 ALF[Flatbread]Proscuitto OK - AL FORNO In [16]: Products.shape Out[16]: (321, 3) In [17]: Conversions = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")) Out[17]: ConversionId object Multiplier float64 ConvertFromUw object		2 R-61779 ALF/Flatbread/Mushroom Pesto OK - AL FORNO
<pre>In [16]: Products.shape Out[16]: (321, 3) In [17]: Conversions = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")) Conversions.dtypes Out[17]: ConversionId object Multiplier float64 ConvertFromUy float64 ConvertFromUom object</pre>		3 R-50590 ALF Flatbread OK OK - AL FORNO
Out[16]: (321, 3) In [17]: Conversions = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")) Conversions.dtypes Out[17]: ConversionId object Multiplier float64 ConvertFromQty float64 ConvertFromUom object		4 R-50494 ALF/Flatbread/Proscuitto OK - AL FORNO
<pre>In [17]: Conversions = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_List.csv")) Conversions.dtypes Out[17]: ConversionId object Multiplier float64 ConvertFromQty float64 ConvertFromUom object</pre>	In [16]:	Products.shape
Out[17]: ConversionId object Multiplier float64 ConvertFromQty float64 ConvertFromUom object	Out[16]:	(321, 3)
Multiplier float64 ConvertFromUy float64 ConvertFromUom object	In [17]:	
ConvertToQty float64	Out[17]:	Multiplier float64 ConvertFromQty float64

or	nversions.h	ead()					
(ConversionId	Multiplier	ConvertFromQty	ConvertFromUom	ConvertToQty	ConvertToUom	
)	NaN	1.000000	1.0	xxx	1.00	L	
1	NaN	0.877193	1.0	1.14L	1.14	L	
2	NaN	0.666667	1.0	1.5L	1.50	L	
3	NaN	0.571429	1.0	1.75 L	1.75	L	
1	NaN	0.500000	1.0	2L	2.00	L	
Cor	nversions.s	hape					

Update Conversion List

C	onversionId	Multiplier	ConvertFromQty	ConvertFromUom	ConvertToQty	ConvertToUom	
0	I-1028	0.008333	1.0	СТ	120.0	g	
1	I-1034	0.008333	1.0	СТ	120.0	g	
2	I-1035	0.010000	1.0	СТ	100.0	g	
3	I-10605	0.008850	1.0	СТ	113.0	g	
4	I-1126	0.006667	1.0	СТ	150.0	g	
	Id = Upda Conversio	te_Conv.lo ns.drop(Co	te_Conv.iterrow oc[index, 'Conv onversions[Conv Update_Conv]	ersionId']	sionId'] ==	Id].index, inpla	ace = True)
Conv				_index(drop=Tru	e, inplace=F	alse).drop_dupl:	icates()
COIN		ld Multiplie	er ConvertFromQt	y ConvertFromUo	n ConvertToQ	ty ConvertToUom	
0	Na	N 1.00000	0 1.	0 XX	X 1.0	00 L	
1	Na	N 0.87719	3 1.	0 1.14	L 1.	14 L	
2	Na	N 0.66666	7 1.	0 1.5	L 1.9	50 L	
3	Na	N 0.57142	9 1.	0 1.75	L 1.:	75 L	
4	Na	N 0.50000	0 1.	0 2	L 2.0	00 L	
179	P-752	0.03524	2 16.	0 е	a 454.0	00 g	
180	P-763	0.01690	9 1.	0 srv	g 59.	14 ml	
481	P-977	0.06826	7 768.	0 slic	e 11250.0	00 g	
182	1-2966	65 0.02500	0 1.	0 eac	h 40.0	00 g	
183	1-3226	0.03333	3 1.	0 е	a 30.0	00 g	
84 ro	ows × 6 colu	umns					
				a", "preprocess e, header = Tru		sions_List.csv")
rea	nte Unit (Converte	er				
	port stand			rmation and con			conversions.csv"))

	Mult	iplier Com	vertFromQty Co	onvertFromUom	ConvertToQty	ConvertToUom	
3	0 4.9	2890	1	tsp	4.92890	ml	
	1 14.7	8700	1	Tbsp	14.78700	ml	
3	2 946.3	5000	1	qt	946.35000	ml	
1	3 473.1	7625	1	pt	473.17625	ml	
	4 28.3	4950	1	oz	28.34950	g	
6]:	liquid	_unit = S		td_Unit[' <mark>Conv</mark>	ertToUom'] =		<pre>rtFromUom'].tolist() 'romUom'].tolist()</pre>
7]:	def sto if el:	d_converte uom in S multipl Qty = f	loat(qty) [*] flo td_Unit.loc[s ty om	ertFromUom']. t.loc[Std_Uni at(multiplier	t['ConvertFr)	omUom'] == uom, == uom, 'Conve	'Multiplier'] srtToUom'].values[0]
]:		the std_c t std_con	onverter verter(0.25,'	lb') == (113.	398, 'g')		
9]:	spc_co def spo if	v = list(c_converte uom in l return	it converter filter(None, er(ingre, qty iquid_unit + std_converter	Conversions[' , uom): solid_unit:		'].tolist()))	
	el	convers multipl if mult	ier = convers iplier.empty:	& (Co ion['Multipli	nversions['C er']	nversionId'] == onvertToUom'] =	<pre>ingre) & (Conversions['ConvertFromUom'] == uom = 'g')]</pre>
		else: Qty Uom	<pre>urn std_conve = float(qty) = conversion urn (Qty, Uom</pre>	/float(multip [^{'ConvertToUo}	lier)	1	
	el						

Items with Non-standard Units

```
In [31]: col_names = list(Ingredients.columns.values)
Items_Nonstd = []
             for index, row in Ingredients.iterrows():
    Ingre = Ingredients.loc[index,'IngredientId']
    Uom = Ingredients.loc[index,'Uom']
    if Uom not in ['g', 'ml'] and Uom not in liquid_unit + solid_unit and Ingre.startswith('I') and Ingre not in Conversi
    Dict = {}
    Dict.update(dict(row))
    There Noresid a promotion [Dict]

                        Items_Nonstd.append(Dict)
             Items_Nonstd = pd.DataFrame(Items_Nonstd, columns = col_names)
Items_Nonstd.drop_duplicates(subset=['IngredientId'], inplace=True,)
Items_Nonstd
Out[31]:
               IngredientId Qty Uom Conversion InvFactor Recipe
              0
                      I-8856 6.000 ea
                                                      1.0 1.0000 R-64671
            1 I-64492 1.000 LOAF 1.0 0.0625 P-26234
              2
                     I-1254 0.500 CT
                                                      1.0 0.5000 P-28369
            3 I-1273 1.000 LOAF 1.0 0.1351 P-58370
              4
                    I-62225 4.000 CT
                                                       1.0 0.4444 P-64456
            5 I-62736 1.000 ea 1.0 1.0000 R-28249
```

	IngredientId	Qty	Uom	Conversion	InvFactor	Recipe
8	I-2281	0.076	650G	1.0	1.0000	R-30524
9	I-2669	0.125	PIE	1.0	1.0000	R-30673
13	I-1223	1.000	СТ	1.0	1.0000	R-54456
30	I-1252	1.000	СТ	1.0	1.0000	R-64997

In [32]: path = os.path.join(os.getcwd(), "data", "cleaning", "Items_Nonstd.csv")
Items_Nonstd.to_csv(path, index = False, header = True)

Clean Preps Units

In [33]: Preps['StdQty'] = np.nan
Preps['StdUom'] = np.nan

```
In [34]: #convert uom into 'g' or 'ml' for each prep using the unit converter
for index in Preps.index:
    PrepId = Preps.loc[index,'PrepId']
    Qty = Preps.loc[index,'PakQty']
    Uom = Preps.loc[index,'PakUOM']
    Preps.loc[index,'StdQty'] = spc_converter(PrepId, Qty, Uom)[0]
    Preps.loc[index,'StdUom'] = spc_converter(PrepId, Qty, Uom)[1]
```

In [35]: Preps

	Prepid	Description	PakQty	PakUOM	InventoryGroup	StdQty	StdUom
0	P-56398	BATCH Guacamole	2.750	Kg	PREP	2750.000000	g
1	P-24750	CHOPPED Cilantro	0.500	Kg	NaN	500.000000	g
2	P-41574	COOKED Black Beans	30.000	Kg	PREP	30000.000000	g
3	P-26068	COOKED Caramelized Onion	1.200	Kg	PREP	1200.000000	g
4	P-28258	COOKED Chow Mein	48.081	Kg	PREP	48081.000000	g
488	P-16305	YIELD Smokie (1pc)	1.000	ea	NaN	112.000005	g
489	P-50781	YIELD Thai Basil	200.000	g	NaN	200.000000	g
490	P-50676	YIELD Thyme	300.000	g	NaN	300.000000	g
491	P-46833	YIELD Yam Fries	800.000	g	NaN	800.000000	g
492	P-57145	YIELD Yellow Pepper	8.300	Kg	NaN	8300.000000	g

493 rows × 7 columns

```
In [36]: # save cleaned preps list to file
path = os.path.join(os.getcwd(), "data", "cleaning", "Preps_Unit_Cleaned.csv")
Preps.to_csv(path, index = False, header = True)
```

Get Preps with Nonstandard Unit

```
In [37]: col_names = list(Preps.columns.values)
Preps_Nonstd = []
              for index, row in Preps.iterrows():
                    stdUom = Preps.loc[index,'stdUom']
if StdUom not in ['g', 'ml']:
    Dict = {}
    Dict.update(dict(row))
                         Preps_Nonstd.append(Dict)
              Preps_Nonstd = pd.DataFrame(Preps_Nonstd, columns = col_names)
```

```
In [38]: Preps_Nonstd
```

Out[38]:		Prepid	Description	PakQty	PakUOM	InventoryGroup	StdQty	StdUom
	0	P-33556	COOKED Fried Fish	1.0	each	NaN	1.0	each
	1	P-64456	POP-UP Coconut Flan LM	9.0	PTN	NaN	9.0	PTN
	2	P-64513	POP Salmon En Papillote LM	1.0	PTN	NaN	1.0	PTN
	3	P-44585	PREP Lime WEDGE	8.0	piece	NaN	8.0	piece

	pld		stransmorth.		InventoryGroup	an contraction of a					
4 P-649	344	Sesame Tuna	1.0	PTN	NaN	1.0	PTN				
		s with nonstand d.read_csv(os.pa						date", '	2reps_(pdateUom	.csv"))
col nam	mes <mark>=</mark> list	(Preps Nonstd.co	olumns.v	values)							
Preps_1	Nonstd_na	= []									
	PrepId not	ps_Nonstd.loc[in t in Manual_Prep			ues:						
if	PrepId not Dict = { Dict.upd Preps_Not Nonstd = pe	t in Manual_Prep } ate(dict(row)) nstd_na.append(1 d.DataFrame(Prep	pU['Prep Dict) ps_Nonst	pId'].val	plumns = col_na						
if Preps_1	PrepId not Dict = { Dict.upda Preps_Not Nonstd = pe	t in Manual_Prep } ate(dict(row)) nstd_na.append(1 d.DataFrame(Prep	pU['Prep Dict) ps_Nonst	pId'].val			StdUom				
if Preps_1 Preps_1	PrepId not Dict = { Dict.upd Preps_Not Nonstd = pe Nonstd	t in Manual_Prep } ate(dict(row)) nstd_na.append(1 d.DataFrame(Prep	pU['Prep Dict) ps_Nonst	pId'].val	plumns = col_na		StdUom each				
if Preps_1 Preps_1 Pre	PrepId not Dict = { Dict.upd Preps_Not Nonstd = pe Nonstd Pld	t in Manual_Prep } ate(dict(row)) nstd_na.append(1 d.DataFrame(Prep Description	pU['Prep Dict) ps_Nonst PakQty	pld'].val td_na, co PakUOM	lumns = col_na	StdQty					
if Preps_1 Preps_1 Pre 0 P-335	PrepId not Dict = { Dict.upda Preps_Not Nonstd = po Nonstd pld 556 POP-0	t in Manual_Prey } ate(dict(row)) nstd_na.append(1 d.DataFrame(Prey Description COOKED FriedFish	pU['Prep Dict) ps_Nonst PakQty 1.0	pId'].val td_na, co PakUOM each	olumns = col_na InventoryGroup NaN	StdQty 1.0	each				
if Preps_1 Preps_1 Pre 0 P-338 1 P-644	PrepId not Dict = { Dict.updi Preps_Not Nonstd = po Nonstd pid 556 FOP-0 513 POP/Sali	t in Manual_Prey } ate(dict(row)) nstd_na.append(1 d.DataFrame(Prey Description COOKED FriedFish UP Coconut Flan LM	pU['Prep Dict) ps_Nonst PakQty 1.0 9.0	pId'].val td_na, co PakUOM each PTN	olumns = col_na InventoryGroup NaN NaN	StdQty 1.0 9.0	each PTN				

```
New Items
```

```
In [41]: # Load current Items List with assigned Emission Factors Category ID
Items_Assigned = pd.read_csv(os.path.join(os.getcwd(), "data", "mapping", "Items_List_Assigned.csv"))
Items_Assigned.head()
                                 Description CaseQty CaseUOM PakQty PakUOM InventoryGroup
Out[41]: ItemId CategoryID
                           1 CHUCK FLAT BONELESS FZN
          0 1-57545
                                                            3.30
                                                                       Kg
                                                                              1.0
                                                                                        Kg
                                                                                                    MEAT
                           1 BEEF STIRFRY COV FR 5.00
                                                                   Kg
         1 I-10869
                                                                              1.0
                                                                                       Kg
                                                                                                    MEAT
                                  BEEF OUTSIDE FLAT AAA
                                                                                                    MEAT
          2 I-7064
                            1
                                                            1.00
                                                                       Kg
                                                                            1.0
                                                                                        Kg
         3 I-37005 1 BEEF MEATBALLS 4.54 Kg 1000.0
                                                                                                    MEAT
                                                                                     g
          4 I-37002 1 BEEF INSIDE ROUND SHAVED
                                                           9.00
                                                                       Kg 1000.0
                                                                                        g
                                                                                                    MEAT
In [42]: Items_Assigned.shape
Out[42]: (1937, 8)
         Get the List of New Items
In [43]: #filter new items by itemID that not in the database and output them in a dataframe
col_names = list(Items.columns.values)
           New_Items_List = []
           for index, row in Items.iterrows():
              ItemId = Items.loc[index,'ItemId']
if ItemId not in Items_Assigned['ItemId'].values:
                   Dict = {}
Dict.update(dict(row))
                   New_Items_List.append(Dict)
           New_Items = pd.DataFrame(New_Items_List, columns = col_names)
In [44]: New_Items.insert(1, "CategoryID", '')
           New Items
Out[44]: ItemId CategoryID Description CaseQty CaseUOM PakQty PakUOM InventoryGroup
In [45]: New_Items.shape
```

```
Out[45]: (0, 8)
```

In [46]:	# store the list of new items into .csv file	
	if not New_Items.empty:	
	<pre>path = os.path.join(os.getcwd(), "data", "mapping", "new items", str(datetime.date(datetime.now()))+"_New_Items.csv")</pre>	
	New_Items.to_csv(path, index = False, header = True)	

Data Summary

Items_Nonstd 10

6

In [47]:	datasum = po	l.Datal	colur	<pre>_Items.shape, Preps_Nonstd.shape, Items_Nonstd.shape], nns = ['count', 'columns'], x = ['New_Items', 'Preps_Nonstd', 'Items_Nonstd'])</pre>
Out[47]:		count	columns	
	New_Items	0	8	
	Preps_Nonstd	5	7	

Climate-Friendly Food Systems (CFFS) Labelling Project

The University of British Columbia

Created by Silvia Huang

Part III: Update Information and Mapping

Set up and Import Libraries

In [1]:	#install libraries if needed
	#!pip3 install pdpipe
In [2]:	import numpy as np
	import pandas as pd
	import pdpipe as pdp
	import matplotlib.pyplot as plt
	import glob
	import os
	import csv
	from itertools import islice
	from decimal import Decimal
	import xml.etree.ElementTree as et
	from xml.etree.ElementTree import parse
	import openpyxl
	import pytest
	from datetime import datetime
In [3]:	#set the root path, change the directory into the project folder
TU [0]:	os.chdir("/Users/silvia/cffs-label")
In [4]:	#enable reading data in the scrolling window
	<pre>#pd.set_option("display.max_rows", None, "display.max_columns", None)</pre>

Import Preprocessed Datasets

	Prepid		Description	PakQty	PakUOM	InventoryGroup	StdQty	StdUom	
0	P-56398		BATCH Guacamole	2.750	Kg	PREP	2750.0	g	
1	P-24750		CHOPPED Cilantro	0.500	Kg	NaN	500.0	g	
2	P-41574		COOKED Black Beans	30.000	Kg	PREP	30000.0	g	
3	P-26068	со	OKED Caramelized Onion	1.200	Kg	PREP	1200.0	g	
4	P-28258		COOKED Chow Mein	48.081	Kg	PREP	48081.0	g	
	hge_facto hge_facto		<pre>= pd.read_csv(os.pa head()</pre>	ath.joir	(os.getc	wd(), "data",	"externa	al", "ghg	e_factors.csv"))
	Category I	D	Food Category	Active 1	otal Suppl	y Chain Emissions	(kg CO2 /	kg food)	
0		1	beef & buffalo meat					41.3463	
1		2	lamb/mutton & goat meat					41.6211	
2		3	pork (pig meat)					9.8315	
3		4	poultry (chicken, turkey)					4.3996	
4		5	butter					11.4316	
	itro_fact itro_fact			oath.joi	.n (os.get	.cwd(), "data",	"extern	nal", "ni	trogen_factors.csv"))
	Category I	D	Food Category	g N lost	/kg produc	:t			
0		1	beef & buffalo meat		329.5	0			
1		2	lamb/mutton & goat meat		231.1	5			
•		3	pork (pig meat)		132.8				

Catego	ory ID		Food Category	g N lost/kg pr	oduct					
3	4	poultry (chicken, turkey)		116.80					
4	5		butter		100.35					
		s = pd.r s.head()	ead_csv(os.pa	ath.join(os	.getcwd	(), "data'	', "exte	rnal", "	water_factor	s.csv"))
Catego	ory ID		Food Category	Freshwater W	/ithdrawa	ls (L/FU) S	tress-Wei	ghted Wate	er Use (L/FU)	
0	1	beet	f & buffalo meat			1677.200			61309.000	
1	2	lamb/mut	ton & goat meat			461.200			258.900	
2	3		pork (pig meat)			1810.300			54242.700	
3	4	poultry (chicken, turkey)			370.300			333.500	
4	5		butter			1010.176			50055.168	
Items_A	ssign								Items_List_A	assigned.csv"))
Iteml	d Cat	egoryID		Description	CaseQty	CaseUOM	PakQty	PakUOM	InventoryGrou	qı
0 I-5754	5	1	CHUCK FLAT BO	NELESS FZN	3.30	Kg	1.0	Kg	ME	AT
1 I-1086	9	1	BEEF STIF	FRY COV FR	5.00	Kg	1.0	Kg	ME	AT

1-57545	. Е	CHUCK FLAT BOINELESS FZIN	5.50	ĸġ	1.0	ĸġ	WEAT
I-10869	1	BEEF STIRFRY COV FR	5.00	Kg	1.0	Kg	MEAT
1-7064	1	BEEF OUTSIDE FLAT AAA	1.00	Kg	1.0	Kg	MEAT
I-37005	1	BEEF MEATBALLS	4.54	Kg	1000.0	g	MEAT
1-37002	1	BEEF INSIDE ROUND SHAVED	9.00	Kg	1000.0	g	MEAT
	I-10869 I-7064 I-37005	I-10869 1 I-7064 1 I-37005 1	I-108691BEEF STIRFRY COV FRI-70641BEEF OUTSIDE FLAT AAAI-370051BEEF MEATBALLS	I-10869 1 BEEF STIRFRY COV FR 5.00 I-7064 1 BEEF OUTSIDE FLAT AAA 1.00 I-37005 1 BEEF MEATBALLS 4.54	I-10869 1 BEEF STIRFRY COV FR 5.00 Kg I-7064 1 BEEF OUTSIDE FLAT AAA 1.00 Kg I-37005 1 BEEF MEATBALLS 4.54 Kg	I-10869 1 BEEF STIRFRY COV FR 5.00 Kg 1.00 I-7064 1 BEEF OUTSIDE FLAT AAA 1.00 Kg 1.0 I-37005 1 BEEF MEATBALLS 4.54 Kg 100.0	I-10869 1 BEEF STIRFRY COV FR 5.00 Kg 1.0 Kg I-7064 1 BEEF OUTSIDE FLAT AAA 1.00 Kg 1.0 Kg I-37005 1 BEEF MEATBALLS 4.54 Kg 100.0 g

Import Update Info

In [10]: #import list of prep that need convert uom to standard uom manually
Manual_PrepU = pd.read_csv(os.path.join(os.getcwd(), "data", "cleaning", "update", "Preps_UpdateUom.csv"))
Manual_PrepU.head()

Out[10]:		Prepid	Description	PakQty	PakUOM	InventoryGroup	StdQty	StdUom
	0	P-54697	LEMON Wedge 1/8	8.0	each	PREP	84.0	g
	1	P-35132	MARINATED Lemon & Herb Chx	185.0	ea	PREP	24050.0	g
	2	P-51992	YIELD Bread Sourdough 5/8	36.0	slice	NaN	1620.0	g
	3	P-26234	BATCH Roasted Garlic Bread	16.0	ea	PREP	1280.0	g
	4	P-26170	GRILLED NaanBread	1.0	ea	PREP	125.0	g

In [1]: #select the file path for new items list with category id New_Items_Added = pd.read_csv(os.path.join(os.getcwd(), "data", "mapping", "new items added", "New_Items_Added_7.csv")) New_Items_Added

InventoryGroup	PakUOM	PakQty	CaseUOM	CaseQty	Description	CategoryID	ItemId	
MEAT	СТ	42	CS	1	BURGER 40Z NATURAL HALAL	59	I-13422	0
FOOD - GROCERY	СТ	1	СТ	48	BURGER VEG MALIBU GARDENBURGER	59	1-63034	1
POULTRY	Kg	1	Kg	4	CHICK BREAST CRUNCH BREADED FZ	4	I-64468	2
BREAD	СТ	1	CT	12	CIABATTA BUN 5"X 5" PLAIN	24	I-1254	3
DAIRY	СТ	12	DOZ	15	EGG LRG 15 DOZEN LOOSE	11	I-62225	4
BREAD	ea	1	ea	1	HAMBURGER BUN WW VEGAN 100gr	24	1-62736	5
BREAD	LOAF	1	LOAF	1	LOAF GARLIC BREAD	24	I-64492	6
FOOD - GROCERY	ml	300	SM CAN	24	MILK CONDENSED SWEET	9	I-3356	7
BREAD	СТ	1	CT	12	PANINI SUB ITALIAN - 7"	24	I-1223	8
SPICES	lb	1	lb	5	PAPRIKA BULK	54	I-53707	9
SEAFOOD	lb	1	lb	1	SAL SOX FLT S/ON PBO PRV OW	12	I-5115	10
PRODUCTION FOOD	L	1	L	10	Soup 1 (10L mon-fri)	55	1-29357	11
PRODUCTION FOOD	L	1	L	10	Soup 2 (10L mon-fri)	55	I-29356	12
BREAD	LOAF	1	LOAF	1	SOURDOUGH BREAD COUNTRY	24	I-1273	13

		Itemic	d Categ	oryID			Description	CaseQ	ty Ca	seUOM	PakQty	Paku	JOM Inve	entoryGrou	р		
-	14	1-63866	6	37		ΤΟΜΑΤΟ Ρ	OLPA MUTTI		10	Kg	1		Kg FOOD	- GROCER	Y		
	15	I-228	1	10	YOGUR	F VANILLA ST	IRRED 650G		1	TUB	1	6	50G	DAIR	Y		
:	Ма	nual_Fa		pd.	ms that adju read_csv(os.p)				ata",	"mappi	.ng", "Ma	anual	_Adjust_Fac	ctors.cs	7"))		
		ItemId	Category	γID	Description	CaseQty	CaseUOM	PakQty	PakUC)M Inv	entoryGro		Active Total Supply Chain missions (kg CO2 / kg food)	g N lost/kg product	Freshwater Withdrawals (L/FU)	Stress- Weighted Water Use (L/FU)	
	0	۱- 52090		59	BURGER BEEF & MUSHROOM HALAL	1.0	CS	48.00	5	ст	ME	AT	25.00894	200.86	1038.84	37961.2	
	1	-ا 45558		59	Prep-Vegan Parmesan	1000.0	g	1.00		g F	RODUCTIO		3.85686	0.00	0.00	0.0	
	2	1-3352		59	MAYONNAISE PAIL TFC 4L	2.0	each	4.00		L	FOOI GROCE		3.55000	0.00	0.00	0.0	
	3	1-3223		59	COCONUT MILK 17/19% MILK FAT	6.0	LG CAN	2.84		L	FOOI GROCE		3.50000	0.00	1.00	1.0	
	4	l- 2898		59	MUSTARD DIJON WINE FLEUR	6.0	jar	1.00		Kg	FOOI GROCE		3.32600	0.00	0.00	0.0	
ι	Jp	date (Correc	t U	om for Pre	os											
1	νp		001100														
		r index Prep qty = uom = Preps	x, row Id = Ma Manua Manua s.loc[P	in M nual l_Pr l_Pr reps	<pre>ith manully a anual_PrepU. _PrepU.loc[index epU.loc[index ['PrepId'] == ['PrepId'] ==</pre>	iterrows() ndex, 'Pre x, 'StdQty x, 'StdUom = PrepId,	: pId'] '] '] 'stdQty']										
	Pr	eps.dro	op_dupl	icat	es(subset=['I	PrepId'],	inplace= T	rue,)									
	Pr	eps.hea	ad()														
		Prepid	I		Description	PakQty P	akUOM Inv	entoryG	roup	StdQty	StdUom						
	0	P-56398	3	BA	TCH Guacamole	2.750	Kg	F	PREP	2750.0	g						
	1	P-24750)	CH	HOPPED Cilantro	0.500	Kg		NaN	500.0	g						
	2	P-41574	1	COOF	KED Black Beans	30.000	Kg	F	REP	30000.0	g						
	3	P-26068	в сооке	ED Ca	ramelized Onion	1.200	Kg	F	PREP	1200.0	g						
	4	P-28258	3	COC	KED Chow Mein	48.081	Kg	F	PREP	48081.0	g						
	Pr	eps.sha	ape														
:	(49	3, 7)															
:					(os.getcwd()) index = False			", "Pre	eps_Li	st_Cle	aned.csv	V")					
1	m	oort L	ist of I	Nev	ltems witl	h Emissi	on Facto	ors Ca	tego	ry ID	Assign	ed					
:	It	ems_Ass	signed_	Upda	igned, New_It ted = pd.cond ted.head()			ndex(dr	rop =Tr	ue, in	place =F a	alse)	.drop_dupl:	icates()			
			Catego	-		Descriptio	n CaseQty	CaseU	ом р	akQtv	PakUOM	Inver	ntoryGroup				
	0	1-57545	outegu	1	CHUCK FLAT B			04360	Kg	1.0	Kg	mver	MEAT				
		1-10869		1		IRFRY COV F			Kg	1.0	Kg		MEAT				
	2	1-7064		1		IDE FLAT AA			Kg	1.0	Kg		MEAT				
		1-37005		1		F MEATBALL				1.0			MEAT				
					BEEF INSIDE RC						g						
	4	1-37002		1	DEEF INSIDE RU	JUNU SHAVE	9.00		Kg 1	1000.0	g		MEAT				

In [19]: Items_Assigned_Updated.shape Out[19]: (1937, 8) In [20]: Items_Assigned_Updated[['CategoryID']] = Items_Assigned_Updated[['CategoryID']].apply(pd.to_numeric) In [21]: path = os.path.join(os.getcwd(), "data", "mapping", "Items_List_Assigned.csv") Items_Assigned_Updated.to_csv(path, index = False, header = True)

Mapping Items to Footprint Factors

Out[22]:		ItemId	CategoryID	Description	CaseQty	CaseUOM	PakQty	PakUOM	InventoryGroup	(kg CO2 / kg food)
	0	-ا 57545	1	CHUCK FLAT BONELESS FZN	3.30	Kg	1.0	Kg	MEAT	41.3463
	1	-ا 10869	1	BEEF STIRFRY COV FR	5.00	Kg	1.0	Kg	MEAT	41.3463
	2	1-7064	1	BEEF OUTSIDE FLAT AAA	1.00	Kg	1.0	Kg	MEAT	41.3463
	3	l- 37005	1	BEEF MEATBALLS	4.54	Kg	1000.0	g	MEAT	41.3463
	4	-ا 37002	1	BEEF INSIDE ROUND SHAVED	9.00	Kg	1000.0	g	MEAT	41.3463
	1932	l- 29357	55	Soup 1 (10L mon-fri)	10.00	L	1.0	L	PRODUCTION FOOD	0.0000
	1933	l- 29356	55	Soup 2 (10L mon-fri)	10.00	L	1.0	L	PRODUCTION FOOD	0.0000
	1934	I-1273	24	SOURDOUGH BREAD COUNTRY	1.00	LOAF	1.0	LOAF	BREAD	1.5225
	1935	l- 63866	37	TOMATO POLPA MUTTI	10.00	Kg	1.0	Kg	FOOD - GROCERY	0.6932
	1936	I-2281	10	YOGURT VANILLA STIRRED 650G	1.00	TUB	1.0	650G	DAIRY	2.9782

1937 rows × 9 columns

	ItemId	CategoryID	Description	CaseQty	CaseUOM	PakQty	PakUOM	InventoryGroup	Emissions (kg CO2 / kg food)	lost/kg product
0	-ا 57545	1	CHUCK FLAT BONELESS FZN	3.30	Kg	1.0	Kg	MEAT	41.3463	329.50
1	-ا 10869	1	BEEF STIRFRY COV FR	5.00	Kg	1.0	Kg	MEAT	41.3463	329.50
2	I-7064	1	BEEF OUTSIDE FLAT AAA	1.00	Kg	1.0	Kg	MEAT	41.3463	329.50
3	-ا 37005	1	BEEF MEATBALLS	4.54	Kg	1000.0	g	MEAT	41.3463	329.50

	ItemId	CategoryID	Description	CaseQty	CaseUOM	PakQty	PakUOM	InventoryGroup	Active Total Supply Chain Emissions (kg CO2 / kg food)	g N lost/kg product
4	ا- 37002	1	BEEF INSIDE ROUND SHAVED	9.00	Kg	1000.0	g	MEAT	41.3463	329.50
1932	ا- 29357	55	Soup 1 (10L mon-fri)	10.00	L	1.0	L	PRODUCTION FOOD	0.0000	0.00
1933	ا- 29356	55	Soup 2 (10L mon-fri)	10.00	L	1.0	L	PRODUCTION FOOD	0.0000	0.00
1934	I-1273	24	SOURDOUGH BREAD COUNTRY	1.00	LOAF	1.0	LOAF	BREAD	1.5225	14.80
1935	۱- 63866	37	TOMATO POLPA MUTTI	10.00	Kg	1.0	Kg	FOOD - GROCERY	0.6932	7.90
1936	I-2281	10	YOGURT VANILLA STIRRED 650G	1.00	TUB	1.0	650G	DAIRY	2.9782	26.07

1937 rows × 10 columns

In [24]: # map water footprint factors

for index in mapping.index: if np.isnan(mapping.loc[index,'Category ID']): mapping.loc[index,'Freshwater Withdrawals (L/FU)'] = 0 mapping.loc[index,'Stress-Weighted Water Use (L/FU)'] = 0

mapping = mapping.drop(columns=['Category ID', 'Food Category'])
mapping

Out[24]: Active Total Stress-Supply Chain Emissions g N lost/kg product Freshwater Weighted Water Use (L/FU) Withdrawals (L/FU) ItemId CategoryID Description CaseQty CaseUOM PakQty PakUOM InventoryGroup (kg CO2/ kg food) CHUCK FLAT **0** 57545 1 BONELESS 3.30 Kg 1.0 Kg MEAT 41.3463 329.50 1677.200 61309.000 FZN BEEF 1 STIRFRY COV **1** 10869 5.00 1.0 MEAT 41.3463 329.50 1677.200 61309.000 Kg Kg FR BEEF **2** I-7064 OUTSIDE 41.3463 329.50 1677.200 61309.000 1 1.0 Kg MEAT 1.00 Kg FLAT AAA i. BEEF MEATBALLS 3 1 4.54 Kg 1000.0 g MEAT 41.3463 329.50 1677.200 61309.000 37005 BEEF INSIDE 1-4 1 ROUND 9.00 Kg 1000.0 g MEAT 41.3463 329.50 1677.200 61309.000 37002 SHAVED ... 1-Soup 1 (10L PRODUCTION 1.0 0.0000 1932 55 10.00 L L 0.00 1.000 1.000 29357 mon-fri) FOOD Soup 2 (10L PRODUCTION FOOD 1.0 55 10.00 L L 0.0000 0.00 1.000 1.000 1933 29356 mon-fri) SOURDOUGH BREAD **1934** I-1273 24 1.00 LOAF 1.0 LOAF BREAD 1.5225 14.80 419.200 12821.700 TOMATO FOOD -1-**1935** 63866 37 POLPA MUTTI 10.00 Kg 1.0 Kg 0.6932 790 77 000 4480 700 GROCERY YOGURT VANILLA 1.0 2.9782 262.409 13002.612 1936 I-2281 10 1.00 TUB 650G DAIRY 26.07 STIRRED 650G

1937 rows × 12 columns

Manully Adjust Footprint Factor for Specific Items

<pre>for index, row in Manual_Factor.iterrows(): itemId = Manual_Factor.loc[index, 'ItemId'] ghge = Manual_Factor.loc[index, 'Active Total Sup nitro = Manual_Factor.loc[index, 'g N lost/kg pro water = Manual_Factor.loc[index, 'Freshwater Witt str_water = Manual_Factor.loc[index, 'Stress-Weig mapping.loc[mapping['ItemId'] == itemId, 'Active mapping.loc[mapping['ItemId'] == itemId, 'g N loc mapping.loc[mapping['ItemId'] == itemId, 'Freshwa mapping.loc[mapping['ItemId'] == itemId, 'Stress- mapping.loc[mapping['ItemId'] == itemId, 'Stress- mapping.loc['temId'] == itemId, 'Stress- mapping.loc['temId'] == itemId'] == itemId, 'Stress- mapping.loc['temId'] == itemId'] == itemId'] == itemId']</pre>	oduct'] hdrawals (L/FU)'] ghted Water Use (L/FU)'] Total Supply Chain Emissions (kg CO2 / kg food)'] = ghge st/kg product'] = nitro ater Withdrawals (L/FU)'] = water
<pre>mapping.drop_duplicates(subset = ['ItemId'], inplace mapping.dtypes</pre>	=True)
mapping.ucypes	
ItemId	object
CategoryID	int64
Description	object
CaseQty	float64
CaseUOM	object
PakQty	float64
PakUOM	object
InventoryGroup	object
Active Total Supply Chain Emissions (kg CO2 / kg food	l) float64
g N lost/kg product	float64
Freshwater Withdrawals (L/FU)	float64
Stress-Weighted Water Use (L/FU)	float64
dtype: object	
mapping.shape	
(1937, 12)	
mapping	
	Active

:		ItemId	CategoryID	Description	CaseQty	CaseUOM	PakQty	PakUOM	InventoryGroup	Active Total Supply Chain Emissions (kg CO2 / kg food)	g N lost/kg product	Freshwater Withdrawals (L/FU)	Stress- Weighted Water Use (L/FU)
	0	ا- 57545	1	CHUCK FLAT BONELESS FZN	3.30	Kg	1.0	Kg	MEAT	41.3463	329.50	1677.200	61309.000
	1	ا- 10869	1	BEEF STIRFRY COV FR	5.00	Kg	1.0	Kg	MEAT	41.3463	329.50	1677.200	61309.000
	2	1-7064	1	BEEF OUTSIDE FLAT AAA	1.00	Kg	1.0	Kg	MEAT	41.3463	329.50	1677.200	61309.000
	3	l- 37005	1	BEEF MEATBALLS	4.54	Kg	1000.0	g	MEAT	41.3463	329.50	1677.200	61309.000
	4	-ا 37002	1	BEEF INSIDE ROUND SHAVED	9.00	Kg	1000.0	g	MEAT	41.3463	329.50	1677.200	61309.000
	1932	-ا 29357	55	Soup 1 (10L mon-fri)	10.00	L	1.0	L	PRODUCTION FOOD	0.0000	0.00	1.000	1.000
	1933	ا- 29356	55	Soup 2 (10L mon-fri)	10.00	L	1.0	L	PRODUCTION FOOD	0.0000	0.00	1.000	1.000
	1934	I-1273	24	SOURDOUGH BREAD COUNTRY	1.00	LOAF	1.0	LOAF	BREAD	1.5225	14.80	419.200	12821.700
	1935	ا- 63866	37	TOMATO POLPA MUTTI	10.00	Kg	1.0	Kg	FOOD - GROCERY	0.6932	7.90	77.000	4480.700
	1936	I-2281	10	YOGURT VANILLA STIRRED 650G	1.00	TUB	1.0	650G	DAIRY	2.9782	26.07	262.409	13002.612

1937 rows × 12 columns

In [29]: path = os.path.join(os.getcwd(), "data", "mapping", "Mapping.csv")
mapping.to_csv(path, index = False, header = True)

Climate-Friendly Food Systems (CFFS) Labelling Project

The University of British Columbia

Created by Silvia Huang

Part IV: Data Analysis

Set up and Import Libraries

In [1]:	#install libraries if needed #!pip3 install pdpipe
In [2]:	<pre>import numpy as np import pandas as pd import pdpipe as pdp import matplotlib.pyplot as plt import glob import os import os from itertools import islice from decimal import Decimal import xml.etree.ElementTree as et from xml.etree.ElementTree import parse import openpyxl import pytest pd.set_option('mode.chained_assignment', None)</pre>
In [3]:	<pre>#set the root path, change the directory into the project folder os.chdir("/Users/silvia/cffs-label")</pre>
In [4]:	#enable reading data in the scrolling window #pd.set option("display.max rows", None, "display.max columns", None)

Import Cleaned Datasets

ItemId Description CaseQty CaseUOM PakUOM InventoryGr dtype: obje	float64 object float64 object oup object							
Items.head	hda -	n CaseQtv	CaseUOM	PakQtv	PakUOM	InventoryGroup		
0 I-7631	5 SPICE POWDER		ea	1.0	lb	SPICES		
1 I-4971 A	RTICHOKE 1/4 SALAD CUT TFO	C 6.0	LG CAN	2.5	Kg	PRODUCE		
2 I-4473	AVOCADO (20CT) MX	X 20.0	СТ	1.0	HEAD	PRODUCE		
3 I-4973	AVOCADO PULP CHUNK	Y 12.0	bag	454.0	g	PRODUCE		
4 I-4496	BAK CHOY BABY BO	30.0	lb	1.0	lb	PRODUCE		
Ingredient Ingredient	s = pd.read_csv(os.pat s.dtypes	h.join(os	.getcwd(),	"data"	, "prepr	ocessed", "Ingre	dients_List.csv"))
IngredientI								
Qty Uom	float64 object							
Conversion	float64							
InvFactor	float64							
Recipe dtype: obje	object							
acype: obje								
Ingredient	s.head()							

	ngredientId	Qty	Uom	Conversion	InvFactor	Recipe		
0	I-3388	1.00	L	1.00000	0.3058	P-10496		
1	1-4660	2.27	Kg	2.20462	0.6942	P-10496		
2	I-4598	1.00	СТ	1.00000	0.0013	P-12954		
3	I-4679	1.00	BUNCH	1.00000	0.0063	P-18318		
4	1-4792	10.00	Kg	2.20462	1.2048	P-18746		
	eps = pd.r eps.dtypes	ead_c	sv(os.pa	th.join(os.	getcwd()	, "data",	"cleaning",	"Preps
Pak Pak Inv Std Std	cription Qty UOM entoryGrou Qty	p	object object float64 object object float64 object					
	eps.head() eps.shape							
(49	3, 7)							
	oducts = p oducts.dty		d_csv(os	.path.join(os.getcw	d(), "data	", "preproce	ssed", "Prod
Sal	dId cription esGroup pe: object	obj obj obj	ect					
: Pro	oducts.hea	d()						
:	Prodid		C	Description	5	alesGroup		
0	R-30154		AD	D Crackers O	K - CUSTO	M KITCHEN		
1	R-56337	ALFIFIa	tbread Me	diterranean	ОК -	AL FORNO		
2	R-61779 AL	FIFlatb	read Mush	room Pesto	OK -	AL FORNO		
3	R-50590		ALFIFI	atbread OK	OK -	AL FORNO		
4	R-50494	ALI	Flatbread	d Proscuitto	ОК -	AL FORNO		
	nversions nversions.			(os.path.jc	in(os.ge	tcwd(), "(lata", "prepr	ocessed", "C
Mul Con Con Con	versionId tiplier vertFromQt vertFromUc vertToQty vertToUom pe: object	Y m	object float64 float64 object float64 object					
: Co	nversions							
	Conversio	nId M	ultiplier	ConvertFrom	ty Conve	ertFromUom	ConvertToQty	ConvertToUom
C	N	aN 1	.000000		1.0	xxx	1.00	L
	Ν	aN C	.877193		1.0	1.14L	1.14	L
1			.666667		1.0	1.5L	1.50	L
		aN 0						L
1	. N		.571429		1.0	1.75 L	1.75	
1 2 3	N N	aN C	.571429		1.0 1.0	1.75 L 2L	1.75 2.00	
1 2 3 4	. N	aN C aN O	.500000		1.0	2L	2.00	L
1 2 3 4	N N	aN C aN 0. 	.500000		1.0 	2L 	2.00	L
1 2 3 4 479	P-75	aN C aN 0. 	.500000	1	1.0 6.0	2L ea	2.00 454.00	L g
1 2 3 4 479 480	P-76	aN C aN 0. 23 0 337 0	.500000 .035242 .016909	1	1.0 6.0 1.0	2L ea srvg	2.00 454.00 59.14	۲ ۰۰۰ ۲ ۳
1 2 3 4 479 480 481	: N N P-7: P-7: P-7:	aN C aN 0. 23 0 337 0 779 0	.500000 .035242 .016909 .068267	1 76	1.0 6.0 1.0 8.0	2L ea srvg slice	2.00 454.00 59.14 11250.00	L g m g
1 2 3 4 479 480	: N N P-7(P-7(P-9) : I-296	aN 0. aN 0. 23 0 337 0 779 0	.500000 .035242 .016909	1 76	1.0 6.0 1.0	2L ea srvg	2.00 454.00 59.14	۲ ۰۰۰ ۲ ۳

In [15]:	<pre>mapping = pd.read_csv(os.path.join(os.getcwd(),</pre>	"data",	"mapping",	"Mapping.csv"))
	mapping.dtvpes			

t[15]:	ItemId	object
	CategoryID	int64
	Description	object
	CaseQty	float64
	CaseUOM	object
	PakQty	float64
	PakUOM	object
	InventoryGroup	object
	Active Total Supply Chain Emissions (kg CO2 / kg food)	float64
	g N lost/kg product	float64
	Freshwater Withdrawals (L/FU)	float64
	Stress-Weighted Water Use (L/FU)	float64
	dtype: object	

In [16]: mapping

Out[16]:

:		itemid	CategoryID	Description	CaseQty	CaseUOM	PakQty	PakUOM	InventoryGroup	Active Total Supply Chain Emissions (kg CO2 / kg food)	g N lost/kg product	Freshwater Withdrawals (L/FU)	Stress- Weighted Water Use (L/FU)
	0	-ا 57545	1	CHUCK FLAT BONELESS FZN	3.30	Kg	1.0	Kg	MEAT	41.3463	329.50	1677.200	61309.000
	1	-ا 10869	1	BEEF STIRFRY COV FR	5.00	Kg	1.0	Kg	MEAT	41.3463	329.50	1677.200	61309.000
	2	1-7064	1	BEEF OUTSIDE FLAT AAA	1.00	Kg	1.0	Kg	MEAT	41.3463	329.50	1677.200	61309.000
	3	ا- 37005	1	BEEF MEATBALLS	4.54	Kg	1000.0	g	MEAT	41.3463	329.50	1677.200	61309.000
	4	۱- 37002	1	BEEF INSIDE ROUND SHAVED	9.00	Kg	1000.0	g	MEAT	41.3463	329.50	1677.200	61309.000
19	32	ا- 29357	55	Soup 1 (10L mon-fri)	10.00	L	1.0	L	PRODUCTION FOOD	0.0000	0.00	1.000	1.000
19	33	ا- 29356	55	Soup 2 (10L mon-fri)	10.00	L	1.0	L	PRODUCTION FOOD	0.0000	0.00	1.000	1.000
19	34	I-1273	24	SOURDOUGH BREAD COUNTRY	1.00	LOAF	1.0	LOAF	BREAD	1.5225	14.80	419.200	12821.700
19	35	l- 63866	37	TOMATO POLPA MUTTI	10.00	Kg	1.0	Kg	FOOD - GROCERY	0.6932	7.90	77.000	4480.700
19	36	I-2281	10	YOGURT VANILLA STIRRED 650G	1.00	TUB	1.0	650G	DAIRY	2.9782	26.07	262.409	13002.612

1937 rows × 12 columns

Unit Converter

In [17]: #import standard unit conversion information for items
Std_Unit = pd.read_csv(os.path.join(os.getcwd(), "data", "external", "standard_conversions.csv"))
Std_Unit.head()

Out[17]:		Multiplier	ConvertFromQty	ConvertFromUom	ConvertToQty	ConvertToUom
	0	4.92890	1	tsp	4.92890	ml
	1	14.78700	1	Tbsp	14.78700	ml
	2	946.35000	1	qt	946.35000	ml
	3	473.17625	1	pt	473.17625	ml
	4	28.34950	1	OZ	28.34950	g

In [18]: #import list of prep that need convert uom to standard uom manually
Manual_PrepU = pd.read_csv(os.path.join(os.getcwd(), "data", "cleaning", "update", "Preps_UpdateUom.csv"))
Manual_PrepU.head()

Out[18]: PrepId Description PakQty PakUOM InventoryGroup StdQty StdUom

		PrepId		Description	PakQty	PakUOM	Inver	ntoryGroup	StdQty	StdUom
	0	P-54697	LEN	ION Wedge 1/8	8.0	each		PREP	84.0	g
	1	P-35132 M	RINATED Len	non & Herb Chx	185.0	ea		PREP	24050.0	g
	2	P-51992	YIELD Bread	Sourdough 5/8	36.0	slice		NaN	1620.0	g
	3	P-26234	BATCHIRoast	ed Garlic Bread	16.0	ea		PREP	1280.0	g
	4	P-26170	GRIL	LED NaanBread	1.0	ea		PREP	125.0	g
]:	Pi Pi Pi Pi	<pre>cep_cov = M cep_cov.ins cep_cov.col</pre>	anual_Prep ert(1, "Mu umns = Con ['Multipli	<pre>nfo for prep U[['PrepId', ltiplier", ' versions.col er'] = Prep_</pre>	'PakQ ') umns	ty','PakUG	ом','			
]:		ConversionIc	Multiplier	ConvertFromQ	ty Cor	nvertFromUo	m C	onvertToQty	Conver	tToUom
	0	P-54697	t.	8	3.0	ea	ch	84.0		g
	1	P-35132	1	185	5.0		ea	24050.0		g
	2	P-51992		36	6.0	sli	се	1620.0		g
	3	P-26234	L.	16	6.0		ea	1280.0		g
	4	P-26170		1	1.0	2	ea	125.0		g
:	Co	rames = [Co onversions onversions		Prep_cov] t(frames).re	set_in	dex(drop=1	frue,	inplace=F	alse).c	irop_dupl
:		Conversio	nId Multipli	er ConvertFrom	nQty C	ConvertFrom	Uom	ConvertToQ	ty Conv	vertToUom
		۹ N	laN	1	1.0		xxx	1.	00	L
		1 N	laN 0.87719	93	1.0		1.14L	1.	14	L
		2 N	laN 0.66666	67	1.0		1.5L	1.	50	L
		3 N	laN 0.57142	9	1.0	1	.75 L	1.	75	L
		4 N	laN 0	.5	1.0		2L	2.	00	L
		••								
	64		641		1.0		un	650.		g
	65	0 P-496	36		12.0		ea	1440.	00	g
	65	i 1 P-148	333		1.0		PTN	500.	00	g
	65		365		2.0		each	100.		g
	65		laN Na	N	NaN		NaN		эN	NaN
				Server to	and a state of the				1940 B.	
	654	rows × 6 co	lumns							
:	11	Iquid_unit	= Std_Unit		t['Con	vertToUom				
]:		<pre>liquid_unit = Std_Unit.loc[Std_Unit['ConvertToUom'] == 'ml', 'ConvertFromUom'].tolist() solid_unit = Std_Unit.loc[Std_Unit['ConvertToUom'] == 'g', 'ConvertFromUom'].tolist() fconstruct a standard unit converter def std_converter(qty, uom): if uom in Std_Unit['ConvertFromUom'].tolist(): multiplier = Std_Unit.loc[Std_Unit['ConvertFromUom'] == uom, 'Multiplier'] Qty = float(qty)*float(multiplier) Uom = Std_Unit.loc[Std_Unit['ConvertFromUom'] == uom, 'ConvertToUom'].values[0] else: Qty = qty Uom = uom return (Qty, Uom)</pre>								
]:		test the st								
3]:	(1	13.398, 'g')							
4]:				erter for sp None, Conver			onId'].tolist()))	
	de			e, qty, uom) nit + solid_		#convert (to st	d uom for	ingred.	ients has

if uom in liquid_unit + solid_unit: #convert to std uom for ingredients has no specific convention ______ return std_converter(qty, uom) elif ingre in spc_cov: #convert to std uom for ingredients has specific convention instruction

```
Out[26]: (129.9999948000002, 'g')
```

GHG Factors Calculation for Preps

```
Preps['GHG Emission (g)'] = 0
Preps['GHG Emission(g)/StdUom'] = 0
In [27]:
                      Preps['N lost (g)'] = 0
                      Preps['N lost (g)/StdUom'] = 0
                      Preps['Freshwater Withdrawals (ml)'] = 0
Preps['Freshwater Withdrawals (ml)/StdUom'] = 0
                      Preps['Stress-Weighted Water Use (ml)'] = 0
                      Preps['Stress-Weighted Water Use (ml)/StdUom'] = 0
                      #calculate GHG, nitro, water footprints per gram/ml of each prep for items as ingredients only
In [28]:
                      def get_items_ghge_prep(index, row):
                              get_items_guge_prep(index, row):
ingres = Ingredients.loc[Ingredients['Recipe'] == Preps.loc[index, 'PrepId']]
ghg = Preps.loc[index, 'GHG Emission (g)']
nitro = Preps.loc[index, 'N lost (g)']
water = Preps.loc[index, 'Freshwater Withdrawals (ml)']
                              str_water = Preps.loc[index, 'Stress-Weighted Water Use (ml)']
weight = Preps.loc[index, 'StdQty']
#print('Index:', index, '\nIngres:\n', ingres)
                              #print('Index:', index, '\nIngres:
for idx, row in ingres.iterrows():
                                        ingre = ingres.loc[idx, 'IngredientId']
if ingre.startswith('I'):
                                                ingre.startswith('l'):
ghge = mapping.loc[mapping['ItemId'] == ingre, 'Active Total Supply Chain Emissions (kg CO2 / kg food)']
nitro_fac = mapping.loc[mapping['ItemId'] == ingre, 'g N lost/kg product']
water_fac = mapping.loc[mapping['ItemId'] == ingre, 'Freshwater Withdrawals (L/FU)']
str_water_fac = mapping.loc[mapping['ItemId'] == ingre, 'Stress-Weighted Water Use (L/FU)']
                                                  #print(ghge)
                                                Qty = float(ingres.loc[idx,'Qty'])
Uom = ingres.loc[idx,'Uom']
                                               if ingre in spc_cov:
    qty = spc_converter(ingre, Qty, Uom)[0]
    ghg += qty*float(ghge)
    nitro += qty*float(nitro_fac)/1000
    water += qty*float(water_fac)
    str_water += qty*float(str_water_fac)
                                                else:
                                                         qty = std_converter(Qty, Uom)[0]
                                                         gdg + = gty*float(gbg)
nitro += gty*float(nitro_fac)/1000
water += gty*float(water_fac)
                              water += qty*float(water_fac)
    str_water += qty*float(str_water_fac)
    #print(ingre, Qty, Uom, qty, float(ghge), gty*float(ghge))
    #print(ghg, nitro, water, str_water)
Preps.loc[index, 'GHG Emission(g)'] = float(ghg)
Preps.loc[index, 'GHG Emission(g)'StdUom'] = ghg/float(weight)
Preps.loc[index, 'N lost (g)'] = float(nitro)
Preps.loc[index, 'N lost (g)/StdUom'] = nitro/float(weight)
Preps.loc[index, 'Freshwater Withdrawals (ml)'] = float(water)
Preps.loc[index, 'Stress-Weighted Water Use (ml)'StdUom'] = str_water/float(weight)
In [29]: #calculate GHG, nitro, water footprints per gram/ml of each prep for other preps as ingredients
                      def get preps ghge prep(index, row):
                               ingres = Ingredients.loc[Ingredients['Recipe'] == Preps.loc[index, 'PrepId']]
                              ghg = Preps.loc[index, 'GHG Emission (g)']
nitro = Preps.loc[index, 'N lost (g)']
water = Preps.loc[index, 'Freshwater Withdrawals (ml)']
str_water = Preps.loc[index, 'Stress-Weighted Water Use (ml)']
                              weight = Preps.loc[index, 'StdQty']
#print('Index:', index, '\nIngres:\n', ingres)
```

```
for idx, row in ingres.iterrows():
                                     ingre = ingres.loc[idx, 'IngredientId']
if ingre.startswith('P') and len(ingres) > 1:
                                             ghge = Preps.loc[Preps['PrepId'] == ingre, 'GHG Emission(g)/StdUom']
                                              guge = freps.loc[Preps[ 'repId'] == ingre, 'N lost (g)/StdUom']
water_fac = Preps.loc[Preps[ 'PrepId'] == ingre, 'Preshwater Withdrawals (ml)/StdUom']
str_water_fac = Preps.loc[Preps[ 'PrepId'] == ingre, 'Stress-Weighted Water Use (ml)/StdUom']
                                              #print(ghge)
                                              gty = float(ingres.loc[idx,'Qty'])
Uom = ingres.loc[idx,'Uom']
if ingre in spc_cov:
                                                     qty = spc_converter(ingre, Qty, Uom)[0]
ghg += qty*float(ghge)
                                                      nitro += qty*float(nitro_fac)
water += qty*float(water_fac)
                                                      str_water += qty*float(str_water_fac)
                                              else:
                                                      qty = std_converter(Qty, Uom)[0]
                                                     dty = std_stdtctct(gty), tan,tr,
ghg += qty*float(gtge)
nitro += qty*float(nitro_fac)
water += qty*float(water_fac)
str water += qty*float(str_water_fac)
                            #print(ingre, Qty, Uom, qty, qty*float(sti_water_lat)
#print(ingre, Qty, Uom, qty, qty*float(ghge))
#print(ghg, nitro, water, str_water)
Preps.loc[index, 'GHG Emission (g)'] = float(ghg)
Preps.loc[index, 'GHG Emission(g)/StdUom'] = ghg/float(weight)
                            Preps.loc[index, 'N lost (g)'] = float(nitro)
Preps.loc[index, 'N lost (g)'] = float(nitro)
Preps.loc[index, 'N lost (g)/StdUom'] = nitro/float(weight)
Preps.loc[index, 'Freshwater Withdrawals (ml)'] = float(water)
Preps.loc[index, 'Stress-Weighted Water Use (ml)'] = float(str_water)
Preps.loc[index, 'Stress-Weighted Water Use (ml)/StdUom'] = str_water/float(weight)
                    #calculate GHG, nitro, water footprints per gram/ml of each prep for linked preps
In [301:
                     def link_preps(index, row):
                             ingres = Ingredients.loc[Ingredients['Recipe'] == Preps.loc[index, 'PrepId']]
                             ghg = Preps.loc[index, 'GHG Emission (g)']
nitro = Preps.loc[index, 'N lost (g)']
water = Preps.loc[index, 'Freshwater Withdrawals (ml)']
                             str water = Preps.loc[index, 'Stress-Weighted Water Use (ml)']
                             weight = Preps.loc[index, 'StdQty']
                             if len(ingres) == 1:
    ingre = ingres.iloc[0]['IngredientId']
                                     if ingre.startswith('P'):
                                              #print(ingres)
                                             #print(ingles)
ghge = Preps.loc[Preps['PrepId'] == ingre, 'GHG Emission(g)/StdUom']
nitro_fac = Preps.loc[Preps['PrepId'] == ingre, 'N lost (g)/StdUom']
water_fac = Preps.loc[Preps['PrepId'] == ingre, 'Freshwater Withdrawals (ml)/StdUom']
str_water_fac = Preps.loc[Preps['PrepId'] == ingre, 'Stress-Weighted Water Use (ml)/StdUom']
                                              Qty = float(ingres.iloc[0]['Qty'])
Uom = ingres.iloc[0]['Uom']
                                              if ingre in spc_cov:
                                                     qty = spc_converter(ingre, Qty, Uom)[0]
ghg = qty*float(ghge)
                                                      nitro = qty*float(nitro_fac)
water = qty*float(water_fac)
                                                      str_water = qty*float(str_water_fac)
                                              else:
                                                     qty = std_converter(Qty, Uom)[0]
                                                      ghg = gty*float(ghge)
                                                     nitro = qty*float(nitro_fac)
water = qty*float(water_fac)
                           water = qty*float(water_fac)
str_water = qty*float(str_water_fac)
#print(ingre, ghge, Qty, Uom, qty, weight)
#print(ghg, nitro, water, str_water)
Preps.loc(index, 'GHG Emission (g)'] = float(qhg)
Preps.loc(index, 'GHG Emission (g)/StdUom'] = ghg/float(weight)
Preps.loc(index, 'N lost (g)'] = float(nitro)
Preps.loc(index, 'N lost (g)/StdUom'] = nitro/float(weight)
Preps.loc(index, 'Freshwater Withdrawals (ml)'] = float(water)
Preps.loc(index, 'Stress-Weighted Water Use (ml)'StdUom'] = water/float(weight)
Preps.loc(index, 'Stress-Weighted Water Use (ml)/StdUom'] = str_water/float(weight)
In [31]: for index, row in Preps.iterrows():
                             get_items_ghge_prep(index , row)
In [32]: for index, row in Preps.iterrows():
                            link preps(index, row)
In [33]: for index, row in Preps.iterrows():
                     get_preps_ghge_prep(index, row)
In [34]: Preps
```

	Prepid	Description	PakQty	PakUOM	InventoryGroup	StdQty	StdUom	GHG Emission (g)	GHG Emission(g)/StdUom	N lost (g)	N lc (g)/StdUc
0	P- 56398	BATCH Guacamole	2.750	Kg	PREP	2750.0	g	1486.216896	0.540443	11.000185	0.0040
1	P- 24750	CHOPPED Cilantro	0.500	Kg	NaN	500.0	g	362.904983	0.725810	5.700834	0.0114
2	P- 41574	COOKED Black Beans	30.000	Kg	PREP	30000.0	g	12625.079663	0.420836	97.526164	0.0032
3	P- 26068	COOKED Caramelized Onion	1.200	Kg	PREP	1200.0	g	2629.549652	2.191291	22.086666	0.0184
4	P- 28258	COOKED Chow Mein	48.081	Kg	PREP	48081.0	g	54907.550000	1.141980	518.100000	0.0107
488	P- 16305	YIELD Smokie (1pc)	1.000	ea	NaN	112.0	g	1101.128000	9.831500	14.873600	0.1328
489	P- 50781	YIELD Thai Basil	200.000	g	NaN	200.0	g	228.111417	1.140557	3.583377	0.0179
490	P- 50676	YIELD Thyme	300.000	g	NaN	300.0	g	228.316600	0.761055	3.586600	0.0119
491	P- 46833	YIELD Yam Fries	800.000	g	NaN	800.0	g	397.440000	0.496800	5.000000	0.0062
492	P- 57145	YIELD Yellow Pepper	8.300	Kg	NaN	8300.0	g	5029.000000	0.605904	79.000000	0.0095

493 rows × 15 columns

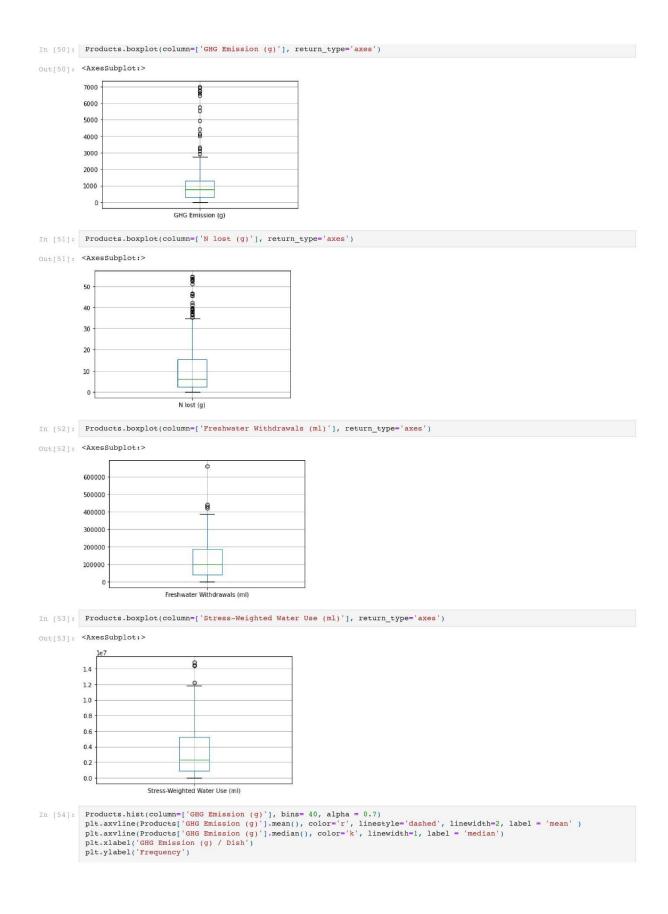
fn [35]: path = os.path.join(os.getcwd(), "data", "final", "Preps Footprints.csv")
Preps.to_csv(path, index = False, header = True)

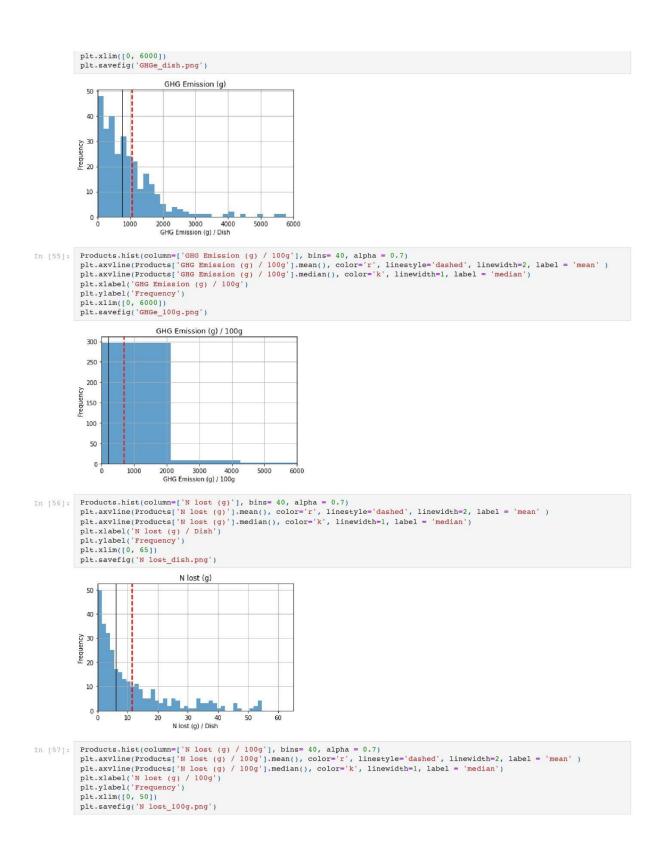
GHGe Calculation for Products

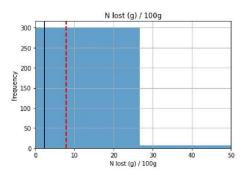
```
Products['Weight (g)'] = 0
Products['GHG Emission (g)'] = 0
Products['N lost (g)'] = 0
Products['Freshwater Withdrawals (ml)'] = 0
In [36]:
                     Products['Stress-Weighted Water Use (ml)'] = 0
In [37]:
                     #calculate GHG, nitro, water footprints per gram/ml of each product for items ingredients only
                    def get_items_ghge(index, row):
    ingres = Ingredients.loc[Ingredients['Recipe'] == Products.loc[index,'ProdId']]
                             ghg = Products.loc[index, 'GHG Emission (g)']
nitro = Products.loc[index, 'N lost (g)']
water = Products.loc[index, 'Freshwater Withdrawals (ml)']
str_water = Products.loc[index, 'Stress-Weighted Water Use (ml)']
                             str_water = Products.loc[index, Stress-weight
weight = Products.loc[index, 'Weight (g)']
#print('Index:', index, '\nIngres:\n', ingres)
for idx, row in ingres.iterrows():
    ingre = ingres.loc[idx, 'IngredientId']
    if are served bit(')
                                      if ingre.startswith('I'):
                                             ingre.startswith('I'):
ghge = mapping.loc[mapping['ItemId'] == ingre, 'Active Total Supply Chain Emissions (kg CO2 / kg food)']
nitro_fac = mapping.loc[mapping['ItemId'] == ingre, 'g N lost/kg product']
water_fac = mapping.loc[mapping['ItemId'] == ingre, 'Freshwater Withdrawals (L/FU)']
str_water_fac = mapping.loc[mapping['ItemId'] == ingre, 'Stress-Weighted Water Use (L/FU)']
                                             weight += qty
ghg += qty*float(ghge)
                                                       nitro += qty*float(nitro_fac)/1000
water += qty*float(water_fac)
                                                      str_water += qty*float(str_water_fac)
                                              else:
                                                       qty = std_converter(Qty, Uom)[0]
                                                       weight += qty
                                                       ghg += qty*float(ghge)
                                                      nitro += qty*float(nitro_fac)/1000
water += qty*float(water_fac)
                             water += gty*iloat(water_tat)
str_water += gty*float(str_water_fat)
#print(ingre, Oty, Uom, gty, float(ghge), gty*float(ghge))
Products.loc[index, 'GHG Emission (g)'] = float(ghg)
Products.loc[index, 'Weight (g)'] = float(weight)
Products.loc[index, 'N lost (g)'] = float(nitro)
```

```
Products.loc[index, 'Freshwater Withdrawals (ml)'] = float(water)
Products.loc[index, 'Stress-Weighted Water Use (ml)'] = float(str_water)
                   #calculate GHG, nitro, water footprints per gram/ml of each product for preps ingredients only
In [38]:
                   def get_preps_ghge(index, row):
    ingres = Ingredients.loc[Ingredients['Recipe'] == Products.loc[index,'ProdId']]
                          ingres = Ingredients.loc[ingredients[ kecipe ] == Froducts.loc[ing
ghg = Products.loc[index, 'GHG Emission (g)']
nitro = Products.loc[index, 'N lost (g)']
water = Products.loc[index, 'Freshwater Withdrawals (ml)']
str_water = Products.loc[index, 'Stress-Weighted Water Use (ml)']
                           weight = Products.loc[index, 'Weight (g)']
                           #print('Index:', index, '\nIngres:\n', ingres)
for idx, row in ingres.iterrows():
                                  ingre = ingres.loc[idx,'IngredientId']
if ingre.startswith('P'):
                                          ghge = Preps.loc[Preps['PrepId'] == ingre, 'GHG Emission(g)/StdUom']
nitro_fac = Preps.loc[Preps['PrepId'] == ingre, 'N lost (g)/StdUom']
water_fac = Preps.loc[Preps['PrepId'] == ingre, 'Freshwater Withdrawals (ml)/StdUom']
                                          str_water_fac = Preps.loc[Preps['PrepId'] == ingre, 'Stress-Weighted Water Use (ml)/StdUom']
                                          Qty = float(ingres.loc[idx, 'Qty'])
Uom = ingres.loc[idx, 'Uom']
if ingre in Conversions['ConversionId'].tolist():
                                                 qty = spc_converter(ingre, Qty, Uom)[0]
weight += qty
                                                 weight += qty
ghg += qty*float(ghge)
nitro += qty*float(nitro_fac)
water += qty*float(water_fac)
str_water += qty*float(str_water_fac)
                                          else:
                                                 qty = std_converter(Qty, Uom)[0]
                                                 weight += qty
ghg += qty*float(ghge)
                                                 nitro += qty*float(nitro_fac)
water += qty*float(water_fac)
                                                  str_water += qty*float(str_water_fac)
                          #print(ingre, Oty, Uom, qty, float(shge), qty*float(ghge))
Products.loc[index, 'GHG Emission (g)'] = float(ghge)
Products.loc[index, 'Weight (g)'] = float(weight)
Products.loc[index, 'N lost (g)'] = float(irro)
Products.loc[index, 'Freshwater Withdrawals (ml)'] = float(water)
Products.loc[index, 'Stress-Weighted Water Use (ml)'] = float(str_water)
                   #calculate GHG, nitro, water footprints per gram/ml of each product for other products ingredients
In [39]:
                   def get_products_ghge(index, row):
    ingres = Ingredients.loc[Ingredients['Recipe'] == Products.loc[index,'ProdId']]
                          ghg = Products.loc[index, 'GHG Emission (g)']
nitro = Products.loc[index, 'N lost (g)']
water = Products.loc[index, 'Freshwater Withdrawals (ml)']
str_water = Products.loc[index, 'Stress-Weighted Water Use (ml)']
                           weight = Products.loc[index, 'Weight (g)']
#print('Index:', index, '\nIngres:\n', ingres)
for idx, row in ingres.iterrows():
                                  ingre = ingres.loc[idx, 'IngredientId']
                                  if ingre.startswith('R'):
                                          ghge = Products.loc[Products['ProdId'] == ingre, 'GHG Emission (g)']
                                          guge = Froducts:loc[Froducts[ FrodId ] == ingre, 'N lost (g) ]
water_fac = Products.loc[Products['ProdId'] == ingre, 'N lost (g)']
str_water_fac = Products.loc[Products['ProdId'] == ingre, 'Stress-Weighted Water Use (ml)']
                                          Weight = Products.loc[Products['ProdId'] == ingre, 'Weight (g)']
Qty = float(ingres.loc[idx,'Qty'])
ghg += Qty*float(ghge)
                                          nitro += Qty*float(nitro_fac)
water += Qty*float(water_fac)
str_water += Qty*float(str_water_fac)
                                          weight += Qty*float(Weight)
                          weight += Gty*float(Weight)
#print(ingre, Oty, float(ghge), Oty*float(ghge))
Products.loc[index, 'GHG Emission (g)'] = float(ghg)
Products.loc[index, 'Weight (g)'] = float(weight)
Products.loc[index, 'N lost (g)'] = float(nitro)
Products.loc[index, 'Freshwater Withdrawals (ml)'] = float(water)
Products.loc[index, 'Stress-Weighted Water Use (ml)'] = float(str_water)
In [40]: for index, row in Products.iterrows():
                         get_items_ghge(index , row)
In [41]: for index, row in Products.iterrows():
                  get_preps_ghge(index, row)
In [42]: for index, row in Products.iterrows():
                   get_products_ghge(index, row)
In [43]: #filter out products using preps with unknown units
Preps_Nonstd = pd.read_csv(os.path.join(os.getcwd(), "data", "cleaning", "Preps_NonstdUom.csv"))
                   Preps_Nonstd
```

			PakQty		÷.		Qty StdU					
0	P-33556	COOKED Fried Fish	1.0	each		NaN	1.0 ea	ach				
1	P-64456	POP-UP Coconut Flan LM	9.0	PTN		NaN	9.0 P	TN				
2	P-64513	POP Salmon En Papillote LM	1.0	PTN		NaN	1.0 P	TN				
3	P-44585	PREP Lime WEDGE	8.0	piece		NaN	8.0 pie	ece				
4	P-64944	Sesame Tuna	1.0	PTN		NaN	1.0 P	TN				
đ	ingre <i>#prin</i> for i i	<pre>r_products(index, row): s = Ingredients.loc[Ing t(ingres) dx, row in ingres.itern ngre = ingres.loc[idx, f ingre in Preps_Nonsto print(ingre, index, Products.drop(index, break</pre>	gredients cows(): 'Ingredie d['PrepIc Products	entId'] i'].toli s.loc[in	.st():		oc[index	,'ProdId']]				
f		, row in Products.itern r_products(index, row)	rows():									
	64456 61	R-64457										
P- P-	33556 81 33556 10	R-64945 R-41868 2 R-33558 4 R-64487										
P: P:	roducts[roducts['GHG Emission (g) / 100 'N lost (g) / 100g'] = 'Freshwater Withdrawalg 'Stress-Weighted Water	round(10 s (ml) /	00*Produ 100g']	= round()	ost (g)'] 100*Produ	/Products cts['Free	s['Weight (g)' shwater Withdr], 3) awals (ml)'].	/Products		
	roducts											
									C1	GHG		
	Prodic	Description	SalesGr	oup W	eight (g)	GHG Emission (g)	N lost (g)	Freshwater Withdrawals (ml)	Stress- Weighted Water Use (ml)		N lost (g) / 100g	Fresh Withdra (ml) /
	Prodic 0 R- 30154		(ок- ом б	eight (g)	Emission (g)	N lost (g)	Withdrawals	Weighted Water Use	Emission (g) /	lost (g) /	Withdr (ml) /
	0 ^R	ADD/Crackers	s CUST KITCI	DK - TOM 6 HEN - AL 14F		Emission (g) 9.135000		Withdrawals (ml) 2515.199950	Weighted Water Use (ml)	Emission (g) / 100g	lost (g) / 100g 1.480	Withdra (ml) / 4192
	0 R [.] 30154	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom	CUST KITCI OK FOF	DK - TOM 6 HEN - AL 145 - AL 205	.000000	Emission (g) 9.135000 313.698698	0.088800	Withdrawals (ml) 2515.199950 56392.170652	Weighted Water Use (ml) 7.693020e+04	Emission (g) / 100g 152.250	lost (g) / 100g 1.480	Withdr (ml) / 4192 3889
	0 R [.] 30152 1 56333	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Elatbread OK		AL 165	.000000 3 .000000 5	Emission (g) 9.135000 313.698698	0.088800	Withdrawals (ml) 2515.199950 56392.170652 84931.503114	Weighted Water Use (ml) 7.693020e+04 2.459800e+06	Emission (g) / 100g 152.250 216.344	lost (g) / 100g 1.480 2.188 2.591	Withdra
	0 R 30154 1 R 56333 2 R 61779 3 R	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Flatbread Proscutter		AL 205 AL 205 AL 205 AL 205 AL 165 AL 205 AL 205	.000000 3 .000000 5 .000000 3	Emission (g) 9.135000 313.698698 597.890333 366.387741	0.088800 3.172877 5.311554 3.575417	Withdrawals (ml) 2515.199950 56392.170652 84931.503114	Weighted Water Use (ml) 7.693020e+04 2.459800e+06 3.685929e+06	Emission (g) / 100g 152.250 216.344 291.654 222.053	lost (g) / 100g 1.480 2.188 2.591	Withdr (ml) / 4192 3889 4143
	0 R 30154 1 56337 2 61775 3 50590 4 R	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Flatbread OK ALF Flatbread Proscuitto	S CUST KITCI OK FOF OK FOF COK FOF	AL 205 AL 205 AL 205 AL 205 AL 165 AL 205 AL 205	.000000 3 .000000 5 .000000 3	Emission (g) 9.135000 313.698698 597.890333 366.387741	0.088800 3.172877 5.311554 3.575417	Withdrawals (ml) 2515.199950 56392.170652 84931.503114 58538.457445	Weighted Water Use (ml) 7.693020e+04 2.459800e+06 3.685929e+06 2.584866e+06	Emission (g) / 100g 152.250 216.344 291.654 222.053	lost (g) / 100g 1.480 2.188 2.591 2.167	Withdr (ml) / 4192 3888 4143 3547
	0 30154 1 8 56333 2 8 61775 3 50599 4 50494 	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Flatbread Proscuitto ALF Flatbread Proscuitto	G CUST KITCI CUST KITCI FOF OK FOF FOF FOF FOF CK SQU	DK - HEN 6 - AL HEN 145 - AL NNO 205 - AL NNO 165 - AL NNO 210 DK -	.000000 3 .000000 3 .000000 3 .000000 7 	Emission (g) 9.135000 313.698698 597.890333 366.387741	0.088800 3.172877 5.311554 3.575417 8.992882 	Withdrawals (ml) 2515.199950 56392.170652 84931.503114 58538.457445 137562.460815 	Weighted Water Use (ml) 7.693020e+04 2.459800e+06 3.685929e+06 2.584866e+06 5.211512e+06	Emission (9) / 100g 152.250 216.344 291.654 222.053 372.226 	lost (g) / 100g 2.188 2.591 2.167 4.282	Withdr (ml) / 4192 388 4143 3547 6550
•	0 R 30154 1 56333 2 61775 3 50590 4 50494 6 64095	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Flatbread OK ALF Flatbread Proscuitto ALF Flatbread Proscuitto THANKSGIVING ONION GRAVY	CUST KITCI OK FOF FOF FOF OK FOF FOF CUST FOF SQU M	AL 145 AL 205 AL	.000000 3 .000000 3 .000000 3 .000000 7 	Emission 9.135000 113.698698 397.890333 366.387741 781.674378 39.004197	0.088800 3.172877 5.311554 3.575417 8.992882 	Withdrawals (ml) 2515.199950 56392.170652 84931.503114 58538.457445 137562.460815 	Weighted Water Use (ml) 7.693020e+04 2.459800e+06 3.685929e+06 2.584866e+06 5.211512e+06 	Emission (9) / 100g 152.250 216.344 291.654 222.053 372.226 	lost (g) / 100g 2.188 2.591 2.167 4.282 0.712	Withdr (ml) 4192 388 4143 3547 6550 17
31	 0 30154 1 56333 2 61775 3 50590 4 50494 <td>ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Flatbread OK ALF Flatbread Prosculto THANKSGIVING ONION GRAVY THANKSGIVING PUMPKIN PIE</td><td>G CUST KITCI CUST KITCI FOF COK FOF FOF FOF FOF FOF FOF FOF FOF FOF F</td><td>AL 145 AL 205 AL 205 AL</td><td>.000000 3 .000000 5 .000000 3 .000000 3 .000000 7 </td><td>Emission 9.135000 113.698698 597.890333 366.387741 781.674378 39.004197 0.190312</td><td>0.088800 3.172877 5.311554 3.575417 8.992882 0.640408</td><td>Withdrawals (m) 2515.199950 56392.170652 84931.503114 58538.457445 137562.460815 1600.018883 52.400000</td><td>Weighted Water Use (m) 7.693020e+04 2.459800e+06 3.685929e+06 5.211512e+06 5.211512e+06 </td><td>Emission (9) / 100g 216.344 291.654 222.053 372.226 43.338</td><td>lost (9) / 1.480 2.188 2.591 2.167 4.282 0.712 1.480</td><td>Withdr (ml) / 4192 388 4143 3547 6550 177 4192</td>	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Flatbread OK ALF Flatbread Prosculto THANKSGIVING ONION GRAVY THANKSGIVING PUMPKIN PIE	G CUST KITCI CUST KITCI FOF COK FOF FOF FOF FOF FOF FOF FOF FOF FOF F	AL 145 AL 205 AL	.000000 3 .000000 5 .000000 3 .000000 3 .000000 7 	Emission 9.135000 113.698698 597.890333 366.387741 781.674378 39.004197 0.190312	0.088800 3.172877 5.311554 3.575417 8.992882 0.640408	Withdrawals (m) 2515.199950 56392.170652 84931.503114 58538.457445 137562.460815 1600.018883 52.400000	Weighted Water Use (m) 7.693020e+04 2.459800e+06 3.685929e+06 5.211512e+06 5.211512e+06 	Emission (9) / 100g 216.344 291.654 222.053 372.226 43.338	lost (9) / 1.480 2.188 2.591 2.167 4.282 0.712 1.480	Withdr (ml) / 4192 388 4143 3547 6550 177 4192
31 31	 a) 30154 b) 30154 c) 30154<	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Flatbread OK ALF Flatbread Proscuitto ALF Flatbread Proscuitto THANKSGIVING ONION GRAVY THANKSGIVING PUMPKIN PIE VEG Bowl Polenta	G CUST KITCI CUST KITCI COK FOF COK FOF COK FOF COK FOF COK FOF COK SQU M COK SQU	AL 145 AL 205 AL 205 AL 165 AL 206 AL		Emission 9.135000 113.698698 597.890333 366.387741 781.674378 39.004197 0.190312	0.088800 3.172877 5.311554 3.575417 8.992882 0.640408 0.001850 7.191962	Withdrawals (m) 2515.199950 56392.170652 84931.503114 58538.457445 137562.460815 1600.018883 52.400000 67809.930649	Weighted Water Use (ml) 7.693020e+04 2.459800e+06 3.685929e+06 5.211512e+06 5.211512e+06 5.590614e+04 1.602713e+03	Emission (9) / 100g 216.344 291.654 222.053 372.226 43.338 152.250	lost (9) / 1.480 2.188 2.591 2.167 4.282 0.712 1.480 2.213	Withdr.(m)) // 4192 388 4143 3547 6550 6550 177 7 4192 2086
31 31 31	 a) 30154 b) 30154 c) 30154<	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Flatbread Proscuttor ALF Flatbread Proscuttor ALF Flatbread Proscuttor THANKSGIVING PUMPKIN PIE VEG Bowl Polentar	3 CUST KITCI 4 OK - SII 5 FOF 6 FOF 7 OK - SII 8 OK - SII 9 OK - SII	CK - AL 145 AL 205 AL 205 AL 165 AL 205 AL 205		Emission 9.135000 113.698698 397.890333 366.387741 781.674378 39.004197 0.190312 333.017684 758.878624	0.088800 3.172877 5.311554 3.575417 8.992882 0.640408 0.001850 7.191962 9.393660	Withdrawals (m) 2515.199950 56392.170652 84931.503114 58538.457445 137562.460815 1600.018883 52.400000 67809.930649	Weighted Water Use (ml) 7.693020e+04 2.459800e+06 3.685929e+06 5.211512e+06 5.590614e+04 1.602713e+03 4.046229e+06 3.348081e+06	Emission (9) / 100g 152.250 2216.344 2291.654 222.053 372.226 43.338 152.250 256.313	lost (9) / 1.480 2.188 2.591 2.167 4.282 0.712 1.480 2.213 2.851	Withdram (m) // // // // // // // // // // // // //
31 31 31 31 32	0 30154 1 56333 2 61775 3 50593 3 50594 4 50494	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Flatbread Proscuttor ALF Flatbread Proscuttor ALF Flatbread Proscuttor THANKSGIVING PUMPKIN PIE VEG Bowl Polentar	CUST KITCI CUST KITCI CUST KITCI FOF FOF CUST SQU M CUST SQU M CUST SQU M M CUST SQU M M CUST SQU M M CUST FOF FOF FOF FOF FOF FOF FOF FOF FOF FO	CK - AL 145 AL 205 AL 205 AL 165 AL 205 AL 205		Emission 9.135000 113.698698 397.890333 366.387741 781.674378 39.004197 0.190312 333.017684 758.878624	0.088800 3.172877 5.311554 3.575417 8.992882 0.640408 0.001850 7.191962 9.393660	Withdrawals (m) 2515.199950 56392.170652 84931.503114 58538.457445 137562.460815 1600.018883 52.400000 67809.930649 110045.912281	Weighted Water Use (ml) 7.693020e+04 2.459800e+06 3.685929e+06 5.211512e+06 5.590614e+04 1.602713e+03 4.046229e+06 3.348081e+06	Emission (9) / 100g 152.250 2216.344 2291.654 2222.053 372.226 372.226 43.338 256.313 230.312	lost (9) / 1.480 2.188 2.591 2.167 4.282 0.712 1.480 2.213 2.851	Withdram (m) // // // // // // // // // // // // //
31 31 31 32 316	0 30154 1 56333 2 61775 3 50593 3 50594 4 50494	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Flatbread Proscuitto ALF Flatbread Proscuitto ALF Flatbread Proscuitto THANKSGIVING ONION GRAVY THANKSGIVING PUMPKIN PIE VEG Bowl Polenta VEG French Toast Eggnog	CUST KITCI CUST KITCI CUST KITCI FOF FOF CUST SQU M CUST SQU M CUST SQU M M CUST SQU M M CUST SQU M M CUST FOF FOF FOF FOF FOF FOF FOF FOF FOF FO	CK - AL 145 AL 205 AL 205 AL 165 AL 205 AL 205		Emission 9.135000 113.698698 397.890333 366.387741 781.674378 39.004197 0.190312 333.017684 758.878624	0.088800 3.172877 5.311554 3.575417 8.992882 0.640408 0.001850 7.191962 9.393660	Withdrawals (m) 2515.199950 56392.170652 84931.503114 58538.457445 137562.460815 1600.018883 52.400000 67809.930649 110045.912281	Weighted Water Use (ml) 7.693020e+04 2.459800e+06 3.685929e+06 5.211512e+06 5.590614e+04 1.602713e+03 4.046229e+06 3.348081e+06	Emission (9) / 100g 152.250 2216.344 2291.654 2222.053 372.226 372.226 43.338 256.313 230.312	lost (9) / 1.480 2.188 2.591 2.167 4.282 0.712 1.480 2.213 2.851	Withdr (ml) / 4192 388 4143 3547
31 31 31 32 316 P:	0 30154 1 56333 2 61775 3 50590 4 50494 50590 6 64095 17 30673 8 R 9 56633 9 56633 6 64095 17 30673 8 R 9 56633 9 56633 6 700 5 rows × 1	ADD Crackers ALF Flatbread Mediterranear ALF Flatbread Mushroom Pesto ALF Flatbread Proscuitto ALF Flatbread Proscuitto ALF Flatbread Proscuitto THANKSGIVING ONION GRAVY THANKSGIVING PUMPKIN PIE VEG Bowl Polenta VEG French Toast Eggnog	CUST KITCI CUST KITCI CUST KITCI FOF FOF CUST SQU M CUST SQU M CUST SQU M M CUST SQU M M CUST SQU M M CUST FOF FOF FOF FOF FOF FOF FOF FOF FOF FO	CK - AL 145 AL 205 AL 205 AL 165 AL 205 AL 205		Emission 9.135000 113.698698 397.890333 366.387741 781.674378 39.004197 0.190312 333.017684 758.878624	0.088800 3.172877 5.311554 3.575417 8.992882 0.640408 0.001850 7.191962 9.393660	Withdrawals (m) 2515.199950 56392.170652 84931.503114 58538.457445 137562.460815 1600.018883 52.400000 67809.930649 110045.912281	Weighted Water Use (ml) 7.693020e+04 2.459800e+06 3.685929e+06 5.211512e+06 5.590614e+04 1.602713e+03 4.046229e+06 3.348081e+06	Emission (9) / 100g 152.250 2216.344 2291.654 2222.053 372.226 372.226 43.338 256.313 230.312	lost (9) / 1.480 2.188 2.591 2.167 4.282 0.712 1.480 2.213 2.851	Withd (mi) 4190 3882 4144 3554 4144 6555 6555 6555 77 7 7 4190 2086 3333







APPENDIX B [GHG EMISSION FACTORS LIST]

Category ID	Food Category	Active Total Supply Chain Emissions (kg CO2 / kg food)
1	beef & buffalo meat	41.3463
2	lamb/mutton & goat meat	41.6211
3	pork (pig meat)	9.8315
4	poultry (chicken, turkey)	4.3996
5	butter	11.4316
6	cheese	8.9104
7	ice cream	4.0163
8	cream	6.9824
9	milk (cow's milk)	2.2325
10	yogurt	2.9782
11	eggs	3.6615
12	fish (finfish)	4.9798
13	crustaceans (shrimp/prawns)	21.1274
14	mollusks	2.4351
15	animal fats	6.9693
16	other legumes	1.6042
17	beans and pulses (dried)	1.6776
18	peas	0.6995
19	peanuts/groundnuts	1.692
20	soybeans/tofu	1.7542
21	other grains/cereals	1.4785
22	corn (maize)	0.9734
23	oats (oatmeal)	2.3017
24	wheat/rye (bread, pasta, baked goods)	1.5225
25	rice	2.5345
26	tree nuts and seeds	4.2854
27	almond milk	0.7021
28	oat milk	0.9943
29	rice milk	0.6972
30	soy milk	0.489
31	other fruits	0.4306
32	apples	0.3581
33	bananas	0.7115

34	berries	1.6547
35	citrus fruit	0.3942
36	cabbages and other brassicas (broccoli)	0.622
37	tomatoes	0.6932
38	root vegetables	0.3062
39	onions and leeks	0.3015
40	other vegetables	0.5029
41	potatoes	0.397
42	cassava and other roots	0.397
43	sugars and sweeteners	1.6414
44	other vegetable oils	3.1509
45	soybeans (oil)	3.0336
46	palm (oil)	4.2483
47	sunflower (oil)	3.0231
48	rapeseed/canola (oil)	3.2401
49	olives (oil)	5.6383
50	barley (beer)	0.9535
51	wine grapes (wine)	1.3776
52	сосоа	10.456
53	coffee	16.6995
54	stimulants & spices misc.	9.3703
55	water & beverages	0
56	salt	0.44
57	vinegar	1.93
58	sauces & paste	0
59	manually adjusted	0
60	human labor	0
61	kitchen supplies	0