load and process USDA data

price data

In [102…

```
food_simplification = {
    "Broccoli florets": "Broccoli",
    "Broccoli heads": "Broccoli",
    "Carrots, cooked whole": "Carrots",
    "Carrots, raw whole": "Carrots",
    "Cauliflower florets": "Cauliflower",
    "Cauliflower heads": "Cauliflower",
    "Celery, trimmed bunches": "Celery",
    "Celery sticks": "Celery",
    "Collard greens": "Collard",
    "Carrots, baby": "Carrots",
    "Cucumbers with peel": "Cucumber",
    "Cucumbers without peel": "Cucumber",
    "Lettuce, romaine, heads": "Lettuce",
    "Lettuce, romaine, hearts": "Lettuce",
    "Mushrooms, whole": "Mushrooms",
    "Mushrooms, sliced": "Mushrooms",
    "Spinach, boiled": "Spinach",
    "Spinach, eaten raw": "Spinach",
    "Tomatoes, grape & cherry": "Tomatoes",
    "Tomatoes, roma & plum": "Tomatoes",
    "Tomatoes, large round": "Tomatoes",
    "Berries, mixed": "Berries",
    "Grapes (raisins)": "Grapes",
    "Mangoes": "Mango",
    "Plum (prunes)": "Plum"
}
```

In [162…

```
from pandas import read_csv, read_excel, concat
from json import dump

prices = concat([read_csv("Vegetable-Prices-2022.csv").rename(columns={"Vege
                 read_csv("Fruit-Prices-2022.csv").rename(columns={"Fruit":
display(prices)
print(prices["CupEquivalentUnit"].unique())
prices = prices[prices["Form"] != "Canned"]
prices = prices[prices["CupEquivalentUnit"] == "pounds"]
prices.index = [food_simplification.get(i,i) for i in prices.index]
prices.drop(columns=["Form", "RetailPriceUnit", "CupEquivalentUnit"], inplac
prices.drop(["Potatoes, french fries", "Dates"]+[i for i in prices.index if
prices = prices.groupby(level=0).mean()
display(prices)
prices.to_csv("prices.csv")

# 1 Cup = 0.236 L
## TODO convert this into per 100 grams to match the nutritional data
food_info = {}
```

```
for food, row in prices.iterrows():
    food_info[food] = {
        "price": row["RetailPrice"],
        "cupEQ": row["CupEquivalentSize"],
        "yield": row["Yield"],
    }

food_info.update({
    "Flax seeds": {
        "price": 3.86,  # sourced from Amazon
        "cupEQ": 0.37,   # sourced from https://www.aqua-calc.com/page/densi
        "yield": 1  # all of flaxseeds are eaten
    }
})

with open("food_info.json", 'w') as jsonOut:
    dump(food_info, jsonOut, indent=3)
```

	Form	RetailPrice	RetailPriceUnit	Yield	CupEquivalentSize	C
Food						
Acorn squash	Fresh	1.2136	per pound	0.4586	0.4519	
Artichoke	Fresh	2.4703	per pound	0.3750	0.3858	
Artichoke	Canned	3.4498	per pound	0.6500	0.3858	
Asparagus	Fresh	2.9531	per pound	0.4938	0.3968	
Asparagus	Canned	3.4328	per pound	0.6500	0.3968	
...	...	...	...	...	...	
Raspberries	Fresh	7.7338	per pound	0.9600	0.3197	
Raspberries	Frozen	6.1590	per pound	1.0000	0.3307	
Strawberries	Fresh	2.9682	per pound	0.9400	0.3197	
Strawberries	Frozen	3.3421	per pound	1.0000	0.3307	
Watermelon	Fresh	0.3820	per pound	0.5200	0.3307	

155 rows × 7 columns

```
['pounds' 'fluid ounces']
```

	RetailPrice	Yield	CupEquivalentSize	CupEquivalentPrice
Acorn squash	1.21360	0.45860	0.45190	1.196100
Apples	1.85410	0.90000	0.24250	0.499600
Apricots	5.63865	0.96500	0.25355	1.256150
Artichoke	2.47030	0.37500	0.38580	2.541500
Asparagus	4.88715	0.76365	0.39680	2.496100
...	...	...	...	...
Sweet potatoes	1.15650	0.88180	0.44090	0.578200
Tomatoes	2.43500	0.91000	0.37480	1.002833
Turnip greens	2.72095	0.76300	0.33620	1.195350
Watermelon	0.38200	0.52000	0.33070	0.242900
Zucchini	1.63590	0.76950	0.39680	0.843700

68 rows × 4 columns

nutritional data

In [133…

```
# create mappings
## nutrient IDs -> nutrient names
# nutrients = read_csv("FoodData_Central_csv_2024-10-31/sub_sample_result.cs
# nutrient_names = dict(zip(nutrients.index.to_list(), nutrients["nutrient_r
nutrients = read_csv("nutrient.csv").set_index("id")
nutrient_names = {}
for ID, row in nutrients.iterrows():
    nutrient_names[ID] = {"name": row["name"], "unit": row["unit_name"]}
display(nutrient_names)
# food IDs -> food names
food_metadata = read_csv("../food.csv").set_index("fdc_id")
food_names = dict(zip(food_metadata.index.to_list(), food_metadata["descript
```

```
{2047: {'name': 'Energy (Atwater General Factors)', 'unit': 'KCAL'},
 2048: {'name': 'Energy (Atwater Specific Factors)', 'unit': 'KCAL'},
 1001: {'name': 'Solids', 'unit': 'G'},
 1002: {'name': 'Nitrogen', 'unit': 'G'},
 1003: {'name': 'Protein', 'unit': 'G'},
 1004: {'name': 'Total lipid (fat)', 'unit': 'G'},
 1005: {'name': 'Carbohydrate, by difference', 'unit': 'G'},
 1006: {'name': 'Fiber, crude (DO NOT USE - Archived)', 'unit': 'G'},
 1007: {'name': 'Ash', 'unit': 'G'},
 1008: {'name': 'Energy', 'unit': 'KCAL'},
 1009: {'name': 'Starch', 'unit': 'G'},
 1010: {'name': 'Sucrose', 'unit': 'G'},
 1011: {'name': 'Glucose', 'unit': 'G'},
 1012: {'name': 'Fructose', 'unit': 'G'},
 1013: {'name': 'Lactose', 'unit': 'G'},
 1014: {'name': 'Maltose', 'unit': 'G'},
 1015: {'name': 'Amylose', 'unit': 'G'},
 1016: {'name': 'Amylopectin', 'unit': 'G'},
 1017: {'name': 'Pectin', 'unit': 'G'},
 1018: {'name': 'Alcohol, ethyl', 'unit': 'G'},
 1019: {'name': 'Pentosan', 'unit': 'G'},
 1020: {'name': 'Pentoses', 'unit': 'G'},
 1021: {'name': 'Hemicellulose', 'unit': 'G'},
 1022: {'name': 'Cellulose', 'unit': 'G'},
 1023: {'name': 'pH', 'unit': 'PH'},
 1024: {'name': 'Specific Gravity', 'unit': 'SP_GR'},
 1025: {'name': 'Organic acids', 'unit': 'G'},
 1026: {'name': 'Acetic acid', 'unit': 'MG'},
 1027: {'name': 'Aconitic acid', 'unit': 'MG'},
 1028: {'name': 'Benzoic acid', 'unit': 'MG'},
 1029: {'name': 'Chelidonic acid', 'unit': 'MG'},
 1030: {'name': 'Chlorogenic acid', 'unit': 'MG'},
 1031: {'name': 'Cinnamic acid', 'unit': 'MG'},
 1032: {'name': 'Citric acid', 'unit': 'MG'},
 1033: {'name': 'Fumaric acid', 'unit': 'MG'},
 1034: {'name': 'Galacturonic acid', 'unit': 'MG'},
 1035: {'name': 'Gallic acid', 'unit': 'MG'},
 1036: {'name': 'Glycolic acid', 'unit': 'MG'},
 1037: {'name': 'Isocitric acid', 'unit': 'MG'},
 1038: {'name': 'Lactic acid', 'unit': 'MG'},
 1039: {'name': 'Malic acid', 'unit': 'MG'},
 1040: {'name': 'Oxaloacetic acid', 'unit': 'MG'},
 1041: {'name': 'Oxalic acid', 'unit': 'MG'},
 1042: {'name': 'Phytic acid', 'unit': 'MG'},
 1043: {'name': 'Pyruvic acid', 'unit': 'MG'},
 1044: {'name': 'Quinic acid', 'unit': 'MG'},
 1045: {'name': 'Salicylic acid', 'unit': 'MG'},
 1046: {'name': 'Succinic acid', 'unit': 'MG'},
 1047: {'name': 'Tartaric acid', 'unit': 'MG'},
 1048: {'name': 'Ursolic acid', 'unit': 'MG'},
 1049: {'name': 'Solids, non-fat', 'unit': 'G'},
 1050: {'name': 'Carbohydrate, by summation', 'unit': 'G'},
 1051: {'name': 'Water', 'unit': 'G'},
 1052: {'name': 'Adjusted Nitrogen', 'unit': 'G'},
 1053: {'name': 'Adjusted Protein', 'unit': 'G'},
 1054: {'name': 'Piperine', 'unit': 'G'},
```

1055: {'name': 'Mannitol', 'unit': 'G'},
1056: {'name': 'Sorbitol', 'unit': 'G'},
1057: {'name': 'Caffeine', 'unit': 'MG'},
1058: {'name': 'Theobromine', 'unit': 'MG'},
1059: {'name': 'Nitrates', 'unit': 'MG'},
1060: {'name': 'Nitrites', 'unit': 'MG'},
1061: {'name': 'Nitrosamine,total', 'unit': 'MG'},
1062: {'name': 'Energy', 'unit': 'kJ'},
1063: {'name': 'Sugars, Total', 'unit': 'G'},
1064: {'name': 'Solids, soluble', 'unit': 'G'},
1065: {'name': 'Glycogen', 'unit': 'G'},
1066: {'name': 'Fiber, neutral detergent (DO NOT USE - Archived)',
 'unit': 'G'},
1067: {'name': 'Reducing sugars', 'unit': 'G'},
1068: {'name': 'Beta-glucans', 'unit': 'G'},
1069: {'name': 'Oligosaccharides', 'unit': 'G'},
1070: {'name': 'Nonstarch polysaccharides', 'unit': 'G'},
1071: {'name': 'Resistant starch', 'unit': 'G'},
1072: {'name': 'Carbohydrate, other', 'unit': 'G'},
1073: {'name': 'Arabinose', 'unit': 'G'},
1074: {'name': 'Xylose', 'unit': 'G'},
1075: {'name': 'Galactose', 'unit': 'G'},
1076: {'name': 'Raffinose', 'unit': 'G'},
1077: {'name': 'Stachyose', 'unit': 'G'},
1078: {'name': 'Xylitol', 'unit': 'G'},
1079: {'name': 'Fiber, total dietary', 'unit': 'G'},
1080: {'name': 'Lignin', 'unit': 'G'},
1081: {'name': 'Ribose', 'unit': 'G'},
1082: {'name': 'Fiber, soluble', 'unit': 'G'},
1083: {'name': 'Theophylline', 'unit': 'MG'},
1084: {'name': 'Fiber, insoluble', 'unit': 'G'},
1085: {'name': 'Total fat (NLEA)', 'unit': 'G'},
1086: {'name': 'Total sugar alcohols', 'unit': 'G'},
1087: {'name': 'Calcium, Ca', 'unit': 'MG'},
1088: {'name': 'Chlorine, Cl', 'unit': 'MG'},
1089: {'name': 'Iron, Fe', 'unit': 'MG'},
1090: {'name': 'Magnesium, Mg', 'unit': 'MG'},
1091: {'name': 'Phosphorus, P', 'unit': 'MG'},
1092: {'name': 'Potassium, K', 'unit': 'MG'},
1093: {'name': 'Sodium, Na', 'unit': 'MG'},
1094: {'name': 'Sulfur, S', 'unit': 'MG'},
1095: {'name': 'Zinc, Zn', 'unit': 'MG'},
1096: {'name': 'Chromium, Cr', 'unit': 'UG'},
1097: {'name': 'Cobalt, Co', 'unit': 'UG'},
1098: {'name': 'Copper, Cu', 'unit': 'MG'},
1099: {'name': 'Fluoride, F', 'unit': 'UG'},
1100: {'name': 'Iodine, I', 'unit': 'UG'},
1101: {'name': 'Manganese, Mn', 'unit': 'MG'},
1102: {'name': 'Molybdenum, Mo', 'unit': 'UG'},
1103: {'name': 'Selenium, Se', 'unit': 'UG'},
1104: {'name': 'Vitamin A, IU', 'unit': 'IU'},
1105: {'name': 'Retinol', 'unit': 'UG'},
1106: {'name': 'Vitamin A, RAE', 'unit': 'UG'},
1107: {'name': 'Carotene, beta', 'unit': 'UG'},
1108: {'name': 'Carotene, alpha', 'unit': 'UG'},
1109: {'name': 'Vitamin E (alpha-tocopherol)', 'unit': 'MG'},

1110: {'name': 'Vitamin D (D2 + D3), International Units', 'unit': 'IU'},
1111: {'name': 'Vitamin D2 (ergocalciferol)', 'unit': 'UG'},
1112: {'name': 'Vitamin D3 (cholecalciferol)', 'unit': 'UG'},
1113: {'name': '25-hydroxycholecalciferol', 'unit': 'UG'},
1114: {'name': 'Vitamin D (D2 + D3)', 'unit': 'UG'},
1115: {'name': '25-hydroxyergocalciferol', 'unit': 'UG'},
1116: {'name': 'Phytoene', 'unit': 'UG'},
1117: {'name': 'Phytofluene', 'unit': 'UG'},
1118: {'name': 'Carotene, gamma', 'unit': 'UG'},
1119: {'name': 'Zeaxanthin', 'unit': 'UG'},
1120: {'name': 'Cryptoxanthin, beta', 'unit': 'UG'},
1121: {'name': 'Lutein', 'unit': 'UG'},
1122: {'name': 'Lycopene', 'unit': 'UG'},
1123: {'name': 'Lutein + zeaxanthin', 'unit': 'UG'},
1124: {'name': 'Vitamin E (label entry primarily)', 'unit': 'IU'},
1125: {'name': 'Tocopherol, beta', 'unit': 'MG'},
1126: {'name': 'Tocopherol, gamma', 'unit': 'MG'},
1127: {'name': 'Tocopherol, delta', 'unit': 'MG'},
1128: {'name': 'Tocotrienol, alpha', 'unit': 'MG'},
1129: {'name': 'Tocotrienol, beta', 'unit': 'MG'},
1130: {'name': 'Tocotrienol, gamma', 'unit': 'MG'},
1131: {'name': 'Tocotrienol, delta', 'unit': 'MG'},
1132: {'name': 'Aluminum, Al', 'unit': 'UG'},
1133: {'name': 'Antimony, Sb', 'unit': 'UG'},
1134: {'name': 'Arsenic, As', 'unit': 'UG'},
1135: {'name': 'Barium, Ba', 'unit': 'UG'},
1136: {'name': 'Beryllium, Be', 'unit': 'UG'},
1137: {'name': 'Boron, B', 'unit': 'UG'},
1138: {'name': 'Bromine, Br', 'unit': 'UG'},
1139: {'name': 'Cadmium, Cd', 'unit': 'UG'},
1140: {'name': 'Gold, Au', 'unit': 'UG'},
1141: {'name': 'Iron, heme', 'unit': 'MG'},
1142: {'name': 'Iron, non-heme', 'unit': 'MG'},
1143: {'name': 'Lead, Pb', 'unit': 'UG'},
1144: {'name': 'Lithium, Li', 'unit': 'UG'},
1145: {'name': 'Mercury, Hg', 'unit': 'UG'},
1146: {'name': 'Nickel, Ni', 'unit': 'UG'},
1147: {'name': 'Rubidium, Rb', 'unit': 'UG'},
1148: {'name': 'Fluoride - DO NOT USE; use 313', 'unit': 'UG'},
1149: {'name': 'Salt, NaCl', 'unit': 'MG'},
1150: {'name': 'Silicon, Si', 'unit': 'UG'},
1151: {'name': 'Silver, Ag', 'unit': 'UG'},
1152: {'name': 'Strontium, Sr', 'unit': 'UG'},
1153: {'name': 'Tin, Sn', 'unit': 'UG'},
1154: {'name': 'Titanium, Ti', 'unit': 'UG'},
1155: {'name': 'Vanadium, V', 'unit': 'UG'},
1156: {'name': 'Vitamin A, RE', 'unit': 'MCG_RE'},
1157: {'name': 'Carotene', 'unit': 'MCG_RE'},
1158: {'name': 'Vitamin E', 'unit': 'MG_ATE'},
1159: {'name': 'cis-beta-Carotene', 'unit': 'UG'},
1160: {'name': 'cis-Lycopene', 'unit': 'UG'},
1161: {'name': 'cis-Lutein/Zeaxanthin', 'unit': 'UG'},
1162: {'name': 'Vitamin C, total ascorbic acid', 'unit': 'MG'},
1163: {'name': 'Vitamin C, reduced ascorbic acid', 'unit': 'MG'},
1164: {'name': 'Vitamin C, dehydro ascorbic acid', 'unit': 'MG'},
1165: {'name': 'Thiamin', 'unit': 'MG'},

1166: {'name': 'Riboflavin', 'unit': 'MG'},
1167: {'name': 'Niacin', 'unit': 'MG'},
1168: {'name': 'Niacin from tryptophan, determined', 'unit': 'MG'},
1169: {'name': 'Niacin equivalent N406 +N407', 'unit': 'MG'},
1170: {'name': 'Pantothenic acid', 'unit': 'MG'},
1171: {'name': 'Vitamin B-6, pyridoxine, alcohol form', 'unit': 'MG'},
1172: {'name': 'Vitamin B-6, pyridoxal, aldehyde form', 'unit': 'MG'},
1173: {'name': 'Vitamin B-6, pyridoxamine, amine form', 'unit': 'MG'},
1174: {'name': 'Vitamin B-6, N411 + N412 +N413', 'unit': 'MG'},
1175: {'name': 'Vitamin B-6', 'unit': 'MG'},
1176: {'name': 'Biotin', 'unit': 'UG'},
1177: {'name': 'Folate, total', 'unit': 'UG'},
1178: {'name': 'Vitamin B-12', 'unit': 'UG'},
1179: {'name': 'Folate, free', 'unit': 'UG'},
1180: {'name': 'Choline, total', 'unit': 'MG'},
1181: {'name': 'Inositol', 'unit': 'MG'},
1182: {'name': 'Inositol phosphate', 'unit': 'MG'},
1183: {'name': 'Vitamin K (Menaquinone-4)', 'unit': 'UG'},
1184: {'name': 'Vitamin K (Dihydrophylloquinone)', 'unit': 'UG'},
1185: {'name': 'Vitamin K (phylloquinone)', 'unit': 'UG'},
1186: {'name': 'Folic acid', 'unit': 'UG'},
1187: {'name': 'Folate, food', 'unit': 'UG'},
1188: {'name': '5-methyl tetrahydrofolate (5-MTHF)', 'unit': 'UG'},
1189: {'name': 'Folate, not 5-MTHF', 'unit': 'UG'},
1190: {'name': 'Folate, DFE', 'unit': 'UG'},
1191: {'name': '10-Formyl folic acid (10HCOFA)', 'unit': 'UG'},
1192: {'name': '5-Formyltetrahydrofolic acid (5-HCOH4', 'unit': 'UG'},
1193: {'name': 'Tetrahydrofolic acid (THF)', 'unit': 'UG'},
1194: {'name': 'Choline, free', 'unit': 'MG'},
1195: {'name': 'Choline, from phosphocholine', 'unit': 'MG'},
1196: {'name': 'Choline, from phosphotidyl choline', 'unit': 'MG'},
1197: {'name': 'Choline, from glycerophosphocholine', 'unit': 'MG'},
1198: {'name': 'Betaine', 'unit': 'MG'},
1199: {'name': 'Choline, from sphingomyelin', 'unit': 'MG'},
1200: {'name': 'p-Hydroxy benzoic acid', 'unit': 'MG'},
1201: {'name': 'Caffeic acid', 'unit': 'MG'},
1202: {'name': 'p-Coumaric acid', 'unit': 'MG'},
1203: {'name': 'Ellagic acid', 'unit': 'MG'},
1204: {'name': 'Ferrulic acid', 'unit': 'MG'},
1205: {'name': 'Gentisic acid', 'unit': 'MG'},
1206: {'name': 'Tyrosol', 'unit': 'MG'},
1207: {'name': 'Vanillic acid', 'unit': 'MG'},
1208: {'name': 'Phenolic acids, total', 'unit': 'MG'},
1209: {'name': 'Polyphenols, total', 'unit': 'MG'},
1210: {'name': 'Tryptophan', 'unit': 'G'},
1211: {'name': 'Threonine', 'unit': 'G'},
1212: {'name': 'Isoleucine', 'unit': 'G'},
1213: {'name': 'Leucine', 'unit': 'G'},
1214: {'name': 'Lysine', 'unit': 'G'},
1215: {'name': 'Methionine', 'unit': 'G'},
1216: {'name': 'Cystine', 'unit': 'G'},
1217: {'name': 'Phenylalanine', 'unit': 'G'},
1218: {'name': 'Tyrosine', 'unit': 'G'},
1219: {'name': 'Valine', 'unit': 'G'},
1220: {'name': 'Arginine', 'unit': 'G'},
1221: {'name': 'Histidine', 'unit': 'G'},

1222: {'name': 'Alanine', 'unit': 'G'},
1223: {'name': 'Aspartic acid', 'unit': 'G'},
1224: {'name': 'Glutamic acid', 'unit': 'G'},
1225: {'name': 'Glycine', 'unit': 'G'},
1226: {'name': 'Proline', 'unit': 'G'},
1227: {'name': 'Serine', 'unit': 'G'},
1228: {'name': 'Hydroxyproline', 'unit': 'G'},
1229: {'name': 'Cysteine and methionine(sulfer containig AA)', 'unit': 'G'},
1230: {'name': 'Phenylalanine and tyrosine (aromatic AA)', 'unit': 'G'},
1231: {'name': 'Asparagine', 'unit': 'G'},
1232: {'name': 'Cysteine', 'unit': 'G'},
1233: {'name': 'Glutamine', 'unit': 'G'},
1234: {'name': 'Taurine', 'unit': 'G'},
1235: {'name': 'Sugars, added', 'unit': 'G'},
1236: {'name': 'Sugars, intrinsic', 'unit': 'G'},
1237: {'name': 'Calcium, added', 'unit': 'MG'},
1238: {'name': 'Iron, added', 'unit': 'MG'},
1239: {'name': 'Calcium, intrinsic', 'unit': 'MG'},
1240: {'name': 'Iron, intrinsic', 'unit': 'MG'},
1241: {'name': 'Vitamin C, added', 'unit': 'MG'},
1242: {'name': 'Vitamin E, added', 'unit': 'MG'},
1243: {'name': 'Thiamin, added', 'unit': 'MG'},
1244: {'name': 'Riboflavin, added', 'unit': 'MG'},
1245: {'name': 'Niacin, added', 'unit': 'MG'},
1246: {'name': 'Vitamin B-12, added', 'unit': 'UG'},
1247: {'name': 'Vitamin C, intrinsic', 'unit': 'MG'},
1248: {'name': 'Vitamin E, intrinsic', 'unit': 'MG'},
1249: {'name': 'Thiamin, intrinsic', 'unit': 'MG'},
1250: {'name': 'Riboflavin, intrinsic', 'unit': 'MG'},
1251: {'name': 'Niacin, intrinsic', 'unit': 'MG'},
1252: {'name': 'Vitamin B-12, intrinsic', 'unit': 'UG'},
1253: {'name': 'Cholesterol', 'unit': 'MG'},
1254: {'name': 'Glycerides', 'unit': 'G'},
1255: {'name': 'Phospholipids', 'unit': 'G'},
1256: {'name': 'Glycolipids', 'unit': 'G'},
1257: {'name': 'Fatty acids, total trans', 'unit': 'G'},
1258: {'name': 'Fatty acids, total saturated', 'unit': 'G'},
1259: {'name': 'SFA 4:0', 'unit': 'G'},
1260: {'name': 'SFA 6:0', 'unit': 'G'},
1261: {'name': 'SFA 8:0', 'unit': 'G'},
1262: {'name': 'SFA 10:0', 'unit': 'G'},
1263: {'name': 'SFA 12:0', 'unit': 'G'},
1264: {'name': 'SFA 14:0', 'unit': 'G'},
1265: {'name': 'SFA 16:0', 'unit': 'G'},
1266: {'name': 'SFA 18:0', 'unit': 'G'},
1267: {'name': 'SFA 20:0', 'unit': 'G'},
1268: {'name': 'MUFA 18:1', 'unit': 'G'},
1269: {'name': 'PUFA 18:2', 'unit': 'G'},
1270: {'name': 'PUFA 18:3', 'unit': 'G'},
1271: {'name': 'PUFA 20:4', 'unit': 'G'},
1272: {'name': 'PUFA 22:6 n-3 (DHA)', 'unit': 'G'},
1273: {'name': 'SFA 22:0', 'unit': 'G'},
1274: {'name': 'MUFA 14:1', 'unit': 'G'},
1275: {'name': 'MUFA 16:1', 'unit': 'G'},
1276: {'name': 'PUFA 18:4', 'unit': 'G'},

```
 1277: {'name': 'MUFA 20:1', 'unit': 'G'},
 1278: {'name': 'PUFA 20:5 n-3 (EPA)', 'unit': 'G'},
 1279: {'name': 'MUFA 22:1', 'unit': 'G'},
 1280: {'name': 'PUFA 22:5 n-3 (DPA)', 'unit': 'G'},
 1281: {'name': 'TFA 14:1 t', 'unit': 'G'},
 1283: {'name': 'Phytosterols', 'unit': 'MG'},
 1284: {'name': 'Ergosterol', 'unit': 'MG'},
 1285: {'name': 'Stigmasterol', 'unit': 'MG'},
 1286: {'name': 'Campesterol', 'unit': 'MG'},
 1287: {'name': 'Brassicasterol', 'unit': 'MG'},
 1288: {'name': 'Beta-sitosterol', 'unit': 'MG'},
 1289: {'name': 'Campestanol', 'unit': 'MG'},
 1290: {'name': 'Unsaponifiable matter (lipids)', 'unit': 'G'},
 1291: {'name': 'Fatty acids, other than 607-615, 617-621, 624-632, 652-654,
686-689)',
  'unit': 'G'},
 1292: {'name': 'Fatty acids, total monounsaturated', 'unit': 'G'},
 1293: {'name': 'Fatty acids, total polyunsaturated', 'unit': 'G'},
 1294: {'name': 'Beta-sitostanol', 'unit': 'MG'},
 1295: {'name': 'Delta-7-avenasterol', 'unit': 'MG'},
 1296: {'name': 'Delta-5-avenasterol', 'unit': 'MG'},
 1297: {'name': 'Alpha-spinasterol', 'unit': 'MG'},
 1298: {'name': 'Phytosterols, other', 'unit': 'MG'},
 1299: {'name': 'SFA 15:0', 'unit': 'G'},
 1300: {'name': 'SFA 17:0', 'unit': 'G'},
 1301: {'name': 'SFA 24:0', 'unit': 'G'},
 1302: {'name': 'Wax Esters(Total Wax)', 'unit': 'G'},
 1303: {'name': 'TFA 16:1 t', 'unit': 'G'},
 1304: {'name': 'TFA 18:1 t', 'unit': 'G'},
 1305: {'name': 'TFA 22:1 t', 'unit': 'G'},
 1306: {'name': 'TFA 18:2 t not further defined', 'unit': 'G'},
 1307: {'name': 'PUFA 18:2 i', 'unit': 'G'},
 1308: {'name': 'PUFA 18:2 t,c', 'unit': 'G'},
 1309: {'name': 'PUFA 18:2 c,t', 'unit': 'G'},
 1310: {'name': 'TFA 18:2 t,t', 'unit': 'G'},
 1311: {'name': 'PUFA 18:2 CLAs', 'unit': 'G'},
 1312: {'name': 'MUFA 24:1 c', 'unit': 'G'},
 1313: {'name': 'PUFA 20:2 n-6 c,c', 'unit': 'G'},
 1314: {'name': 'MUFA 16:1 c', 'unit': 'G'},
 1315: {'name': 'MUFA 18:1 c', 'unit': 'G'},
 1316: {'name': 'PUFA 18:2 n-6 c,c', 'unit': 'G'},
 1317: {'name': 'MUFA 22:1 c', 'unit': 'G'},
 1318: {'name': 'Fatty acids, saturated, other', 'unit': 'G'},
 1319: {'name': 'Fatty acids, monounsat., other', 'unit': 'G'},
 1320: {'name': 'Fatty acids, polyunsat., other', 'unit': 'G'},
 1321: {'name': 'PUFA 18:3 n-6 c,c,c', 'unit': 'G'},
 1322: {'name': 'SFA 19:0', 'unit': 'G'},
 1323: {'name': 'MUFA 17:1', 'unit': 'G'},
 1324: {'name': 'PUFA 16:2', 'unit': 'G'},
 1325: {'name': 'PUFA 20:3', 'unit': 'G'},
 1326: {'name': 'Fatty acids, total sat., NLEA', 'unit': 'G'},
 1327: {'name': 'Fatty acids, total monounsat., NLEA', 'unit': 'G'},
 1328: {'name': 'Fatty acids, total polyunsat., NLEA', 'unit': 'G'},
 1329: {'name': 'Fatty acids, total trans-monoenoic', 'unit': 'G'},
 1330: {'name': 'Fatty acids, total trans-dienoic', 'unit': 'G'},
 1331: {'name': 'Fatty acids, total trans-polyenoic', 'unit': 'G'},
```

1332: {'name': 'SFA 13:0', 'unit': 'G'},
1333: {'name': 'MUFA 15:1', 'unit': 'G'},
1334: {'name': 'PUFA 22:2', 'unit': 'G'},
1335: {'name': 'SFA 11:0', 'unit': 'G'},
1336: {'name': 'ORAC, Hydrophyllic', 'unit': 'UMOL_TE'},
1337: {'name': 'ORAC, Lipophillic', 'unit': 'UMOL_TE'},
1338: {'name': 'ORAC, Total', 'unit': 'UMOL_TE'},
1339: {'name': 'Total Phenolics', 'unit': 'MG_GAE'},
1340: {'name': 'Daidzein', 'unit': 'MG'},
1341: {'name': 'Genistein', 'unit': 'MG'},
1342: {'name': 'Glycitein', 'unit': 'MG'},
1343: {'name': 'Isoflavones', 'unit': 'MG'},
1344: {'name': 'Biochanin A', 'unit': 'MG'},
1345: {'name': 'Formononetin', 'unit': 'MG'},
1346: {'name': 'Coumestrol', 'unit': 'MG'},
1347: {'name': 'Flavonoids, total', 'unit': 'MG'},
1348: {'name': 'Anthocyanidins', 'unit': 'MG'},
1349: {'name': 'Cyanidin', 'unit': 'MG'},
1350: {'name': 'Proanthocyanidin (dimer-A linkage)', 'unit': 'MG'},
1351: {'name': 'Proanthocyanidin monomers', 'unit': 'MG'},
1352: {'name': 'Proanthocyanidin dimers', 'unit': 'MG'},
1353: {'name': 'Proanthocyanidin trimers', 'unit': 'MG'},
1354: {'name': 'Proanthocyanidin 4-6mers', 'unit': 'MG'},
1355: {'name': 'Proanthocyanidin 7-10mers', 'unit': 'MG'},
1356: {'name': 'Proanthocyanidin polymers (>10mers)', 'unit': 'MG'},
1357: {'name': 'Delphinidin', 'unit': 'MG'},
1358: {'name': 'Malvidin', 'unit': 'MG'},
1359: {'name': 'Pelargonidin', 'unit': 'MG'},
1360: {'name': 'Peonidin', 'unit': 'MG'},
1361: {'name': 'Petunidin', 'unit': 'MG'},
1362: {'name': 'Flavans, total', 'unit': 'MG'},
1363: {'name': 'Catechins, total', 'unit': 'MG'},
1364: {'name': 'Catechin', 'unit': 'MG'},
1365: {'name': 'Epigallocatechin', 'unit': 'MG'},
1366: {'name': 'Epicatechin', 'unit': 'MG'},
1367: {'name': 'Epicatechin-3-gallate', 'unit': 'MG'},
1368: {'name': 'Epigallocatechin-3-gallate', 'unit': 'MG'},
1369: {'name': 'Procyanidins, total', 'unit': 'MG'},
1370: {'name': 'Theaflavins', 'unit': 'MG'},
1371: {'name': 'Thearubigins', 'unit': 'MG'},
1372: {'name': 'Flavanones, total', 'unit': 'MG'},
1373: {'name': 'Eriodictyol', 'unit': 'MG'},
1374: {'name': 'Hesperetin', 'unit': 'MG'},
1375: {'name': 'Isosakuranetin', 'unit': 'MG'},
1376: {'name': 'Liquiritigenin', 'unit': 'MG'},
1377: {'name': 'Naringenin', 'unit': 'MG'},
1378: {'name': 'Flavones, total', 'unit': 'MG'},
1379: {'name': 'Apigenin', 'unit': 'MG'},
1380: {'name': 'Chrysoeriol', 'unit': 'MG'},
1381: {'name': 'Diosmetin', 'unit': 'MG'},
1382: {'name': 'Luteolin', 'unit': 'MG'},
1383: {'name': 'Nobiletin', 'unit': 'MG'},
1384: {'name': 'Sinensetin', 'unit': 'MG'},
1385: {'name': 'Tangeretin', 'unit': 'MG'},
1386: {'name': 'Flavonols, total', 'unit': 'MG'},
1387: {'name': 'Isorhamnetin', 'unit': 'MG'},

1388: {'name': 'Kaempferol', 'unit': 'MG'},
1389: {'name': 'Limocitrin', 'unit': 'MG'},
1390: {'name': 'Myricetin', 'unit': 'MG'},
1391: {'name': 'Quercetin', 'unit': 'MG'},
1392: {'name': 'Theogallin', 'unit': 'MG'},
1393: {'name': "Theaflavin -3,3' -digallate", 'unit': 'MG'},
1394: {'name': "Theaflavin -3' -gallate", 'unit': 'MG'},
1395: {'name': 'Theaflavin -3 -gallate', 'unit': 'MG'},
1396: {'name': '(+) -Gallo catechin', 'unit': 'MG'},
1397: {'name': '(+)-Catechin 3-gallate', 'unit': 'MG'},
1398: {'name': '(+)-Gallocatechin 3-gallate', 'unit': 'MG'},
1399: {'name': 'Mannose', 'unit': 'G'},
1400: {'name': 'Triose', 'unit': 'G'},
1401: {'name': 'Tetrose', 'unit': 'G'},
1402: {'name': 'Other Saccharides', 'unit': 'G'},
1403: {'name': 'Inulin', 'unit': 'G'},
1404: {'name': 'PUFA 18:3 n-3 c,c,c (ALA)', 'unit': 'G'},
1405: {'name': 'PUFA 20:3 n-3', 'unit': 'G'},
1406: {'name': 'PUFA 20:3 n-6', 'unit': 'G'},
1407: {'name': 'PUFA 20:4 n-3', 'unit': 'G'},
1408: {'name': 'PUFA 20:4 n-6', 'unit': 'G'},
1409: {'name': 'PUFA 18:3i', 'unit': 'G'},
1410: {'name': 'PUFA 21:5', 'unit': 'G'},
1411: {'name': 'PUFA 22:4', 'unit': 'G'},
1412: {'name': 'MUFA 18:1-11 t (18:1t n-7)', 'unit': 'G'},
1413: {'name': 'MUFA 18:1-11 c (18:1c n-7)', 'unit': 'G'},
1414: {'name': 'PUFA 20:3 n-9', 'unit': 'G'},
2000: {'name': 'Total Sugars', 'unit': 'G'},
2003: {'name': 'SFA 5:0', 'unit': 'G'},
2004: {'name': 'SFA 7:0', 'unit': 'G'},
2005: {'name': 'SFA 9:0', 'unit': 'G'},
2006: {'name': 'SFA 21:0', 'unit': 'G'},
2007: {'name': 'SFA 23:0', 'unit': 'G'},
2008: {'name': 'MUFA 12:1', 'unit': 'G'},
2009: {'name': 'MUFA 14:1 c', 'unit': 'G'},
2010: {'name': 'MUFA 17:1 c', 'unit': 'G'},
2011: {'name': 'TFA 17:1 t', 'unit': 'G'},
2012: {'name': 'MUFA 20:1 c', 'unit': 'G'},
2013: {'name': 'TFA 20:1 t', 'unit': 'G'},
2014: {'name': 'MUFA 22:1 n-9', 'unit': 'G'},
2015: {'name': 'MUFA 22:1 n-11', 'unit': 'G'},
2016: {'name': 'PUFA 18:2 c', 'unit': 'G'},
2017: {'name': 'TFA 18:2 t', 'unit': 'G'},
2018: {'name': 'PUFA 18:3 c', 'unit': 'G'},
2019: {'name': 'TFA 18:3 t', 'unit': 'G'},
2020: {'name': 'PUFA 20:3 c', 'unit': 'G'},
2021: {'name': 'PUFA 22:3', 'unit': 'G'},
2022: {'name': 'PUFA 20:4c', 'unit': 'G'},
2023: {'name': 'PUFA 20:5c', 'unit': 'G'},
2024: {'name': 'PUFA 22:5 c', 'unit': 'G'},
2025: {'name': 'PUFA 22:6 c', 'unit': 'G'},
2026: {'name': 'PUFA 20:2 c', 'unit': 'G'},
2027: {'name': 'Proximate', 'unit': 'G'},
2028: {'name': 'trans-beta-Carotene', 'unit': 'UG'},
2029: {'name': 'trans-Lycopene', 'unit': 'UG'},
2032: {'name': 'Cryptoxanthin, alpha', 'unit': 'UG'},

2033: {'name': 'Total dietary fiber (AOAC 2011.25)', 'unit': 'G'},
2034: {'name': 'Insoluble dietary fiber (IDF)', 'unit': 'G'},
2035: {'name': 'Soluble dietary fiber (SDFP+SDFS)', 'unit': 'G'},
2036: {'name': 'Soluble dietary fiber (SDFP)', 'unit': 'G'},
2037: {'name': 'Soluble dietary fiber (SDFS)', 'unit': 'G'},
2038: {'name': 'High Molecular Weight Dietary Fiber (HMWDF)', 'unit': 'G'},
2039: {'name': 'Carbohydrates', 'unit': 'G'},
2040: {'name': 'Other carotenoids', 'unit': 'UG'},
2041: {'name': 'Tocopherols and tocotrienols', 'unit': 'MG'},
2042: {'name': 'Amino acids', 'unit': 'G'},
2043: {'name': 'Minerals', 'unit': 'MG'},
2044: {'name': 'Lipids', 'unit': 'G'},
2045: {'name': 'Proximates', 'unit': 'G'},
2046: {'name': 'Vitamins and Other Components', 'unit': 'G'},
2055: {'name': 'Total Tocopherols', 'unit': 'MG'},
2054: {'name': 'Total Tocotrienols', 'unit': 'MG'},
2053: {'name': 'Stigmastadiene', 'unit': 'MG'},
2052: {'name': 'Delta-7-Stigmastenol', 'unit': 'MG'},
2049: {'name': 'Daidzin', 'unit': 'MG'},
2050: {'name': 'Genistin', 'unit': 'MG'},
2051: {'name': 'Glycitin', 'unit': 'MG'},
2057: {'name': 'Ergothioneine', 'unit': 'MG'},
2058: {'name': 'Beta-glucan', 'unit': 'G'},
2059: {'name': 'Vitamin D4', 'unit': 'UG'},
2060: {'name': 'Ergosta-7-enol', 'unit': 'MG'},
2061: {'name': 'Ergosta-7,22-dienol', 'unit': 'MG'},
2062: {'name': 'Ergosta-5,7-dienol', 'unit': 'MG'},
2063: {'name': 'Verbascose', 'unit': 'G'},
2064: {'name': 'Oligosaccharides', 'unit': 'MG'},
2065: {'name': 'Low Molecular Weight Dietary Fiber (LMWDF)', 'unit': 'G'},
2068: {'name': 'Vitamin E', 'unit': 'MG'},
2067: {'name': 'Vitamin A', 'unit': 'UG'},
2069: {'name': 'Glutathione', 'unit': 'MG'}}

In [ ]:

```
# inconsistent category codes are used between these files


# all_foods = read_csv("../food.csv").set_index("fdc_id")
# categories = read_csv("food_category.csv")#.set_index("")
```

In [135…

```
# nutrients = read_csv("FoodData_Central_csv_2024-10-31/fndds_ingredient_nut
foods = read_csv("../food_nutrient.csv").set_index("fdc_id")
foods = foods[foods["amount"] > 0]
display(foods)
```

```
/var/folders/j8/f5pb70wx5gn1qv1nz77lfhyr0000gn/T/ipykernel_94432/602315285.p
y:2: DtypeWarning: Columns (10) have mixed types. Specify dtype option on im
port or set low_memory=False.
  foods = read_csv("../food_nutrient.csv").set_index("fdc_id")
```

fdc_id	id	nutrient_id	amount	data_points	derivation_id	min	max
1105904	13706930	1293	53.33	NaN	71.0	NaN	NaN
1105904	13706916	1008	867.00	NaN	71.0	NaN	NaN
1105904	13706928	1258	13.33	NaN	71.0	NaN	NaN
1105904	13706929	1292	20.00	NaN	71.0	NaN	NaN
1105904	13706914	1004	93.33	NaN	71.0	NaN	NaN
...	...	...	...	...	...	...	...
2721948	34555377	1005	17.86	NaN	71.0	NaN	NaN
2721948	34555389	1253	107.00	NaN	71.0	NaN	NaN
2721948	34555392	1292	14.29	NaN	71.0	NaN	NaN
2721948	34555383	1092	536.00	NaN	71.0	NaN	NaN
2721948	34555379	2000	3.57	NaN	73.0	NaN	NaN

18110227 rows × 12 columns

In [ ]:

```
foods_json = {}
for fdc, row in foods.iterrows():
    if fdc not in foods_json:
        foods_json[fdc] = {
            "name": food_names[fdc],
            "nutrients": {nutrient_names[row["nutrient_id"]]["name"]: row["a
        }
    else:    foods_json[fdc]["nutrients"].update({nutrient_names[row["nutrie
print(len(foods_json.keys()))
```

In [ ]:

```
from json import dump
with open("../food_nutrients.json", 'w') as jsonOut:
    dump(foods_json, jsonOut, indent=3)
```

In [140…

```
# only examine fresh (raw) foods, since this is the scope of this project
from json import load, dump
with open("../food_nutrients.json", 'r') as jsonIn:
    foods_json = load(jsonIn)

print(len(foods_json.keys()))

fresh_foods_json = {}
for ID, content in foods_json.items():
    if " raw" not in str(content["name"]):  continue
    fresh_foods_json[ID] = content

names = set(x["name"] for x in fresh_foods_json.values())
print(len(names))

with open("fresh_foods_nutrients.json", 'w') as jsonOut:
```

```
    dump(fresh_foods_json, jsonOut, indent=3)


fresh_foods_names = {}
for ID, content in fresh_foods_json.items():
    name = content["name"]
    fresh_foods_names[name] = content
    fresh_foods_names[name].pop("name")
    fresh_foods_names[name].update({"id": ID})

print(len(fresh_foods_names))
with open("fresh_foods_nutrients_names.json", 'w') as jsonOut:
    dump(fresh_foods_names, jsonOut, indent=3)
```

```
1902490
2254
2254
```

In [ ]:

```
with open("food_names.json", 'w') as jsonOut:
    dump(food_names, jsonOut, indent=3)

with open("nutrient_names.json", 'w') as jsonOut:
    dump(nutrient_names, jsonOut, indent=3)
```

physiological data

In [ ]:

```
# sourced for an active, 28 year-old, 150lb, man   https://www.nal.usda.gov/

# macronutritional needs, with noted adjustments
from pandas import read_csv, concat
from math import inf

macro_needs = read_csv("macronutritional_needs.csv").set_index("Macronutrier
macro_needs["low_bound"] = [""]*len(macro_needs)
macro_needs["high_bound"] = [""]*len(macro_needs)
macro_needs["units"] = [""]*len(macro_needs)
for nutrient, row in macro_needs.iterrows():
    if "-" in row["Recommended Intake Per Day"]:
        minimum, maximum = row["Recommended Intake Per Day"].split("-")
        macro_needs.at[nutrient, "low_bound"] = minimum.strip()
        macro_needs.at[nutrient, "high_bound"] = maximum.split()[0]
        macro_needs.at[nutrient, "units"] = maximum.split()[1]
    elif nutrient == "Saturated fatty acids":
        macro_needs.at[nutrient, "high_bound"] = str(round((3000/ 10 /9), 1)
        macro_needs.at[nutrient, "low_bound"] = str(0)
        macro_needs.at[nutrient, "units"] = "grams"
    elif nutrient == "Dietary Cholesterol":
        macro_needs.at[nutrient, "high_bound"] = str(800)  # arbitrary but a
        macro_needs.at[nutrient, "low_bound"] = str(0)
        macro_needs.at[nutrient, "units"] = "mg"
    elif nutrient == "Total Water":
        macro_needs.at[nutrient, "low_bound"] = "0.37"
        macro_needs.at[nutrient, "high_bound"] = row["Recommended Intake Per
        macro_needs.at[nutrient, "units"] = row["Recommended Intake Per Day"
    elif nutrient == "Protein":
```

```
        macro_needs.at[nutrient, "low_bound"] = "100"
        macro_needs.at[nutrient, "high_bound"] = "150"
        macro_needs.at[nutrient, "units"] = "grams"
    elif "As low" not in row["Recommended Intake Per Day"]:
        macro_needs.at[nutrient, "low_bound"] = row["Recommended Intake Per
        macro_needs.at[nutrient, "high_bound"] = "10000"
        macro_needs.at[nutrient, "units"] = row["Recommended Intake Per Day"
macro_needs.loc["Energy"] = {"low_bound": 2400, "high_bound": 3200, "units":

lst = []
for i in macro_needs.index:
    if i != "α-Linolenic Acid":     lst.append(i)
    else:  lst.append("Linolenic Acid")
macro_needs.index = lst
macro_needs.drop("Recommended Intake Per Day", axis=1, inplace=True)
display(macro_needs)

# micronutritional needs
vitamin_needs = read_csv("vitamin_needs.csv").set_index("Vitamin").rename(co
vitamin_needs["units"] = [""]*len(vitamin_needs)
for nutrient, row in vitamin_needs.iterrows():
    val, unit = row["low_bound"].split()
    vitamin_needs.at[nutrient, "low_bound"] = val
    vitamin_needs.at[nutrient, "units"] = unit
    vitamin_needs.at[nutrient, "high_bound"] = str(row["high_bound"]).split(
    if nutrient == "Carotenoids":
        vitamin_needs.at[nutrient, "high_bound"] = "100000"
        vitamin_needs.at[nutrient, "units"] = "mg"
    if nutrient == "Folate":   vitamin_needs.at[nutrient, "high_bound"] = "3
    if nutrient == "Choline":
        vitamin_needs.at[nutrient, "low_bound"] = "550"
        vitamin_needs.at[nutrient, "high_bound"] = "3300"
        vitamin_needs.at[nutrient, "units"] = "mg"

mineral_needs = read_csv("mineral_needs.csv").set_index("Mineral").rename(co
mineral_needs["units"] = [""]*len(mineral_needs)
for nutrient, row in mineral_needs.iterrows():
    val, unit = row["low_bound"].split()
    mineral_needs.at[nutrient, "low_bound"] = val
    mineral_needs.at[nutrient, "units"] = unit
    mineral_needs.at[nutrient, "high_bound"] = str(row["high_bound"]).split(
    if "Magnesium" in nutrient:  mineral_needs.at[nutrient, "high_bound"] =
    elif nutrient == "Phosphorus":
        mineral_needs.at[nutrient, "low_bound"] = "700"
        mineral_needs.at[nutrient, "high_bound"] = "4000"
        mineral_needs.at[nutrient, "units"] = "mg"
    elif nutrient == "Copper":
        mineral_needs.at[nutrient, "low_bound"] = "0.9"
        mineral_needs.at[nutrient, "high_bound"] = "10"
        mineral_needs.at[nutrient, "units"] = "mg"
    elif nutrient == "Potassium":
        mineral_needs.at[nutrient, "high_bound"] = "15000"


# combining the nutritional sources
nutrition = concat([macro_needs, vitamin_needs, mineral_needs])
```

```
display(nutrition)
print(nutrition.shape)

from json import dump
with open("nutrition.json", 'w') as jsonOut:
    dump(nutrition.T.to_dict(), jsonOut, indent=3)
```

	low_bound	high_bound	units
Carbohydrate	331	478	grams
Total Fiber	41	10000	grams
Protein	100	150	grams
Fat	65	114	grams
Saturated fatty acids	0	33.3	grams
Linolenic Acid	1.6	10000	grams
Linoleic Acid	17	10000	grams
Dietary Cholesterol	0	800	mg
Total Water	0.37	3.7	liters
Energy	2400	3200	kcal

	low_bound	high_bound	units
Carbohydrate	331	478	grams
Total Fiber	41	10000	grams
Protein	100	150	grams
Fat	65	114	grams
Saturated fatty acids	0	33.3	grams
Linolenic Acid	1.6	10000	grams
Linoleic Acid	17	10000	grams
Dietary Cholesterol	0	800	mg
Total Water	0.37	3.7	liters
Energy	2400	3200	kcal
Vitamin A	900	3,000	mcg
Vitamin C	90	2,000	mg
Vitamin D	15	100	mcg
Vitamin B6	1.3	100	mg
Vitamin E	15	1,000	mg
Vitamin K	120	10000	mcg
Thiamin	1.2	10000	mg
Vitamin B12	2.4	10000	mcg
Riboflavin	1.3	10000	mg
Folate	400	3000	mcg
Niacin	16	35	mg
Choline	550	10000	mg
Pantothenic Acid	5	10000	mg
Biotin	30	10000	mcg
Carotenoids	0	100000	mg
Calcium	1,000	2,500	mg
Chloride	2.3	3.6	g
Chromium	35	inf	mcg
Copper	0.9	10	mg
Fluoride	4	10	mg
Iodine	150	1,100	mcg
Iron	8	45	mg
Magnesium	400	10000	mg

	low_bound	high_bound	units
Manganese	2.3	11	mg
Molybdenum	45	2,000	mcg
Phosphorus	700	4000	mg
Potassium	3,400	15000	mg
Selenium	55	400	mcg
Sodium	1,500	2,300	mg
Zinc	11	40	mg

```
(40, 3)
```

In [4]:
```
food_physiology_mapping = {
    "PUFA 18:2": "Linoleic Acid",
    "PUFA 18:3": "Linolenic Acid",
    "Fatty acids, total saturated": "Saturated fatty acids",
    "Total lipid (fat)": "Fat",
    "Cholesterol": "Dietary Cholesterol",
    "Fiber, total dietary": "Total Fiber",
    "Carbohydrate, by difference": "Carbohydrate",
    "Water": "Total Water",

    "Zinc, Zn": "Zinc",
    "Phosphorus, P": "Phosophorous",
    "Magnesium, Mg": "Magnesium",
    "Iron, Fe": "Iron",
    "Sodium, Na": "Sodium",
    "Potassium, K": "Potassium",
    "Copper, Cu": "Copper",
    "Manganese, Mn": "Manganese",
    "Selenium, Se": "Selenium",
    "Manganese, Mn": "Manganese",
    "Calcium, Ca": "Calcium",
    "Iodine, I": "Iodine",
    "Molybdenum, Mo": "Molybdenum",

    "Vitamin C, total ascorbic acid": "Vitamin C",
    "Vitamin A, IU": "Vitamin A",
    "Vitamin B-6": "Vitamin B6",
    "Folate, DFE": "Folate",
    "Vitamin K (Menaquinone-4)": "Vitamin K",
    "Vitamin K (phylloquinone)": "Vitamin K",
    "Vitamin B-12": "Vitamin B12",
    "Pantothenic acid": "Pantothenic Acid",

    "Tocopherol, gamma": "Vitamin E",
    "Vitamin E (alpha-tocopherol)": "Vitamin E",
    "Tocotrienol, alpha": "Vitamin E",

    "Carotene, beta": "Carotenoids",
    "Lycopene": "Carotenoids",
    "Lutein + zeaxanthin": "Carotenoids",
```

```
    "Choline, total": "Choline",
}

from json import dump
dump(food_physiology_mapping, open("food_physiology_mapping.json", 'w'), inc
```

In [5]:
```
from json import load
food_physiology_mapping = load(open("food_physiology_mapping.json", 'r'))

for food, nutrients in fresh_foods_names.items():
    new_nutrients = {}
    for nutrient, value in nutrients["nutrients"].items():
        # if nutrient in food_physiology_mapping:  print(food_physiology_map
        nutrient = food_physiology_mapping.get(nutrient, nutrient)
        new_nutrients[nutrient] = value
    fresh_foods_names[food] = new_nutrients

from json import dump
dump(fresh_foods_names, open("fresh_foods_nutrients_names_physiology.json",
```

In [21]:
```
from collections import defaultdict

def average_dict_values(dicts):
    sum_counts = defaultdict(lambda: [0, 0])  # {key: [sum, count]}
    for d in dicts:
        for food2, nutrients in d.items():
            for nutrient, value in nutrients.items():
                sum_counts[nutrient] = sum_counts.get(nutrient, [0,0])
                sum_counts[nutrient][0] += value  # Sum of values
                sum_counts[nutrient][1] += 1      # Count occurrences
    return {key: sum_val / count for key, (sum_val, count) in sum_counts.ite


matches = {}
for food, pricing in food_info.items():
    for food2, nutrients in fresh_foods_physio_names.items():
        if all([x in food2.lower() for x in [f.strip().replace(",", '').lowe
            if food in matches:  matches[food].append({food2: nutrients})
            else: matches[food] = [{food2: nutrients}]
    if food not in matches:  print(food)  ;  continue
    matches[food] = average_dict_values(matches[food])

matches.update({
   "Flax seeds": {   # sourced from Cronometer's access to NCCDB
      "Carbohydrate": 34.36,
      "Protein": 18.04,
      "Manganese": 2.405,
      "Fat": 37.28,
      "Magnesium": 372,
      "Niacin": 3.756,
      "Total Fiber": 23.13,
      "Iron": 5.78,
      "Zinc": 4.74,
      "Calcium": 230,
```

```
      "Total Water": 6.8,
      "Copper": 1.344,
      "Sodium": 37,
      "Potassium": 793,
      "Nitrogen": 0.1397,
      "Vitamin B6": 0.393,
      "Thiamin": 0.537,
      "Riboflavin": 0.161,
      "Ash": 3.53,
      "Phosophorous": 556,
      "Saturated fatty acids": 3.281,
      "Vitamin E": 0.31,
      "Vitamin C": 0.6,
      "Energy": 514,
      "Choline": 78.7,
      "Folate": 41,
      "Linoleic Acid": 5.265,
      "Vitamin K": 3.7,
      "Linolenic Acid": 19.42,
      "Selenium": 135.9,
      "Pantothenic Acid": 0.985,
      "Carotenoids": 651
   }})

print(len(matches), len(food_info))
# print(matches.keys())

from json import dump
dump(matches, open("food_matches.json", 'w'), indent=3)
```

```
Flax seeds
70 70
```

define the constraints for each food

load the previously defined mappings of price, food contents, and nutritional needs

In [126…

```
from json import load

with open("food_info.json", 'r') as jsonIn:
    food_info = load(jsonIn)

with open("fresh_foods_nutrients_names.json", 'r') as jsonIn:
    fresh_foods_names = load(jsonIn)

with open("fresh_foods_nutrients_names_physiology.json", 'r') as jsonIn:
    fresh_foods_physio_names = load(jsonIn)

with open("nutrition.json", 'r') as jsonIn:
```

```
    nutrition = load(jsonIn)

with open("food_physiology_mapping.json", 'r') as jsonIn:
    food_physiology_mapping = load(jsonIn)

with open("food_matches.json", 'r') as jsonIn:
    food_matches = load(jsonIn)
```

finding matches between the data sources: nutritional need and food composition

In [148…

```
# upload my custom model construction API
from modelseedpy.core.optlanghelper import *

# define all of the relevant variables
##NOTE creating integer variables for servings of food, since these are easi
variables = {}
for food in food_info.keys():
    food = food.replace(" ", "_")
    variables[food] = tupVariable(food, Bounds(0, 5), "continuous")


constraints = {}
# determine the nutrients for which there are few foods defined
undefined_nutrients = {}
for nutrient, content in nutrition.items():
    foodCount = 0
    for food, pricing in food_info.items():
        if nutrient not in food_matches[food]:  continue
        foodCount += 1
    undefined_nutrients[nutrient] = foodCount
print(dict(sorted(undefined_nutrients.items(), key=lambda item: item[1])))

# nutrition constraint
#NOTE eq:  nutrient_lowBound <= sum_n( sum_f( var_f * amount_n,f ) ) <= nutr
grams_per_L = 998
for nutrient, content in nutrition.items():
    if undefined_nutrients[nutrient] < 6:  print(f"Skipping {nutrient} for l
    nutrient_foods = {}
    lb = float(str(content["low_bound"]).replace(",", ''))
    ub = float(str(content["high_bound"]).replace(",", ''))
    # print(nutrient, lb, ub)
    for food, pricing in food_info.items():
        if nutrient not in food_matches[food]:  continue
        amount = food_matches[food][nutrient]
        food = food.replace(" ", "_")
        if nutrient == "Total Water": amount /= grams_per_L
        nutrient_foods[food] = nutrient_foods.get(food, {})
        nutrient_foods[food].update({"elements": [variables[food].name, amou
    nutrient = nutrient.replace(" ", "_")
    # print(f"{nutrient} has {lb}-{ub} constraint")
    constraints[nutrient] = tupConstraint(name=nutrient,
                                          bounds=Bounds(lb, ub),
                                          expr={"elements": list(nutrient_fo
```

```
    # break

# volume constraint
# NOTE eq:  5 cups <= sum_f(var_f [lb or pint] * cupsPerServing_f [cup/lb or
volume_expression = {"elements": [], "operation": "Add"}
for food, pricing in food_info.items():
    food = food.replace(" ", "_")
    volume_expression["elements"].append({"elements": [variables[food].name,
constraints["volume"] = tupConstraint(name="volume", bounds=Bounds(5, 20), e

# number of foods constraint
# for food in food_info.keys():
#     variables[food+"_bin"] = tupVariable(food+"_bin", Bounds(0, 1), "binar
#     food = food.replace(" ", "_")
#     constraints[food+"_bin"] = tupConstraint(food+"_bin", bounds=Bounds(0,
#                                               expr={
#                                                   "elements": [
#                                                       variables[food].bounc
#                                                       {"elements": [-1, var
#                                                       {"elements": [-variab
#                                                   "operation": "Add"})

# constraints["foods"] = tupConstraint(name="foods", bounds=Bounds(5, 20), e
#     varName for varName in variables if "_bin" in varName], "operation": "


# define the objective
#NOTE eq: min  sum_f(var_f * pricePerServing_f)
objective = tupObjective("minimize cost of nutritional diet", [], "min")
for food, pricing in food_info.items():
    food = food.replace(" ", "_")
    objective.expr.append({
        "elements": [
            {"elements": [variables[food].name, pricing["price"]/pricing["yi
            "operation": "Mul"}],
        "operation": "Add"
    })

# create an Optland model from the defined variables, constraints, and objec
model = OptlangHelper.define_model("minimize_nutrition_cost", list(variables
                                   list(constraints.values()),
                                   objective, True)
with open("diet_optimization.lp", 'w') as lp:
    lp.write(model.to_lp())
```

```
{'Vitamin D': 0, 'Chloride': 0, 'Chromium': 0, 'Fluoride': 0, 'Iodine': 0,
'Phosphorus': 0, 'Dietary Cholesterol': 3, 'Molybdenum': 3, 'Vitamin B12':
5, 'Biotin': 11, 'Vitamin A': 47, 'Pantothenic Acid': 53, 'Linolenic Acid':
55, 'Carotenoids': 59, 'Manganese': 60, 'Saturated fatty acids': 62, 'Linole
ic Acid': 62, 'Choline': 62, 'Vitamin K': 63, 'Selenium': 63, 'Folate': 64,
'Sodium': 64, 'Vitamin E': 65, 'Energy': 66, 'Vitamin C': 66, 'Riboflavin':
66, 'Vitamin B6': 67, 'Thiamin': 67, 'Niacin': 67, 'Copper': 67, 'Total Fibe
r': 68, 'Iron': 68, 'Carbohydrate': 69, 'Protein': 69, 'Fat': 69, 'Total Wat
er': 69, 'Calcium': 69, 'Magnesium': 69, 'Potassium': 69, 'Zinc': 69}
Skipping Dietary Cholesterol for lack of foods
Skipping Vitamin D for lack of foods
Skipping Vitamin B12 for lack of foods
Skipping Chloride for lack of foods
Skipping Chromium for lack of foods
Skipping Fluoride for lack of foods
Skipping Iodine for lack of foods
Skipping Molybdenum for lack of foods
Skipping Phosphorus for lack of foods
```

In [122…

```
print(len(constraints))
```

```
32
```

In [161…

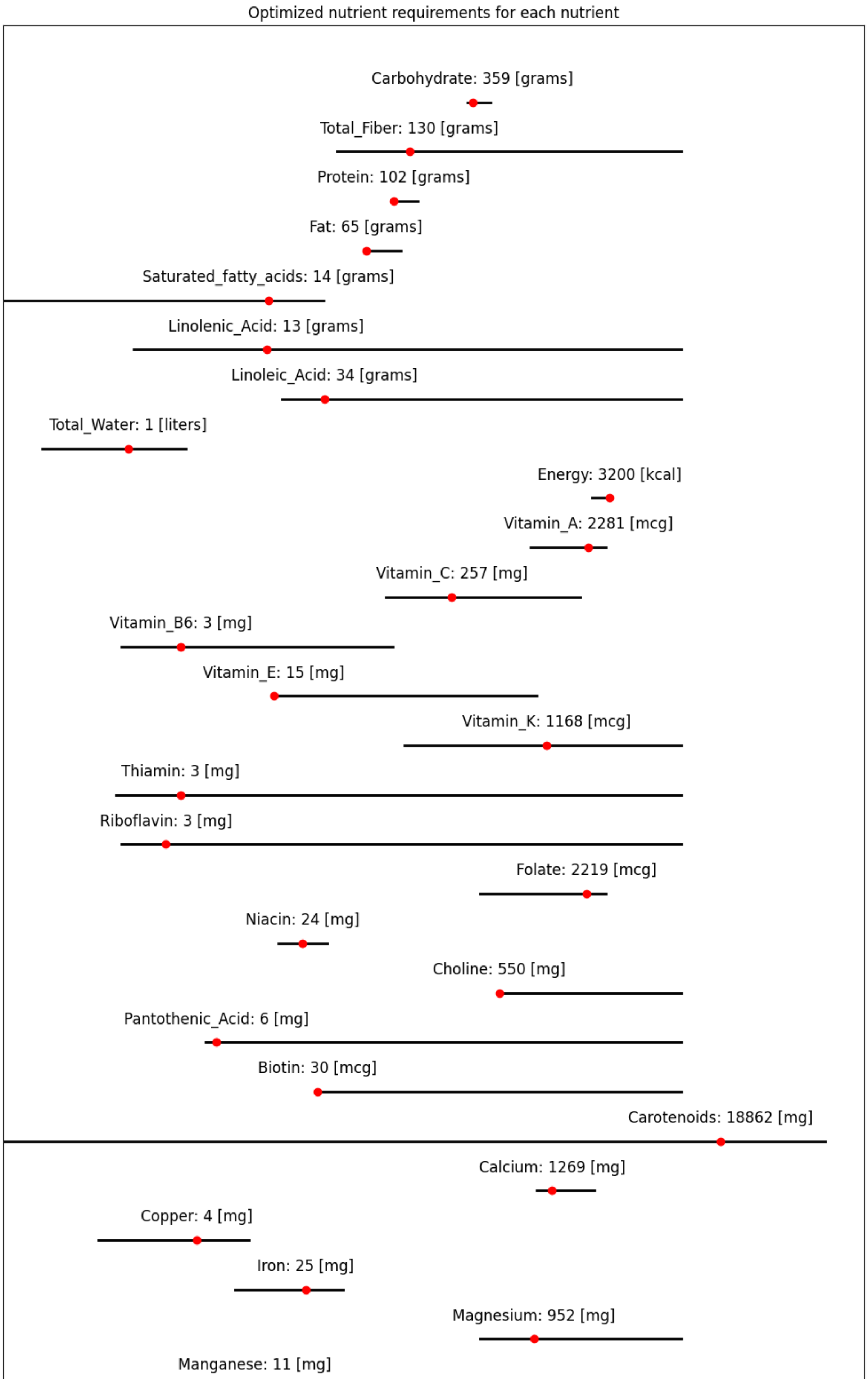
```
model.configuration.lp_method = "exact"
sol = model.optimize()
# pprint(constraints)
optimum_diet = {k:v for k,v in model.primal_values.items() if v>0}
pprint(optimum_diet)
optimum_csv = {"food": list(optimum_diet.keys()), "grams": [int(v*100) for v
from pandas import DataFrame
DataFrame(optimum_csv).to_csv("optimum_diet.csv")

# deduce the nutritional composition of the optimum diet
# optimum_nutrition = {}
# for food, amount in optimum_diet.items():

# for variable in model.variables:
#     print(f"Variable {variable.name} has {variable.primal} from {variable.
    # if variable.name == "volume": continue
    # if variable.primal < float(str(nutrition[variable.name.replace("_", '
    #     print(f"\tfails to meet the minimum {nutrition[variable.name.repla

print(model.objective.value)
constraint_values = {}
for constraint in model.constraints:
    constraint_values[constraint.name] = {"lb": constraint.lb, "val": constr
    print(f"Constraint {constraint.name} has {constraint.primal} from {const
    if constraint.name == "volume": continue
    if constraint.primal < float(str(nutrition[constraint.name.replace("_",
        print(f"\tfails to meet the minimum {nutrition[constraint.name.repla
    # print()
    # print(f"   Primal Value (LHS): {constraint.expression}")
    # print(f"Dual Value (Shadow Price): ")

display(constraint_values)
```

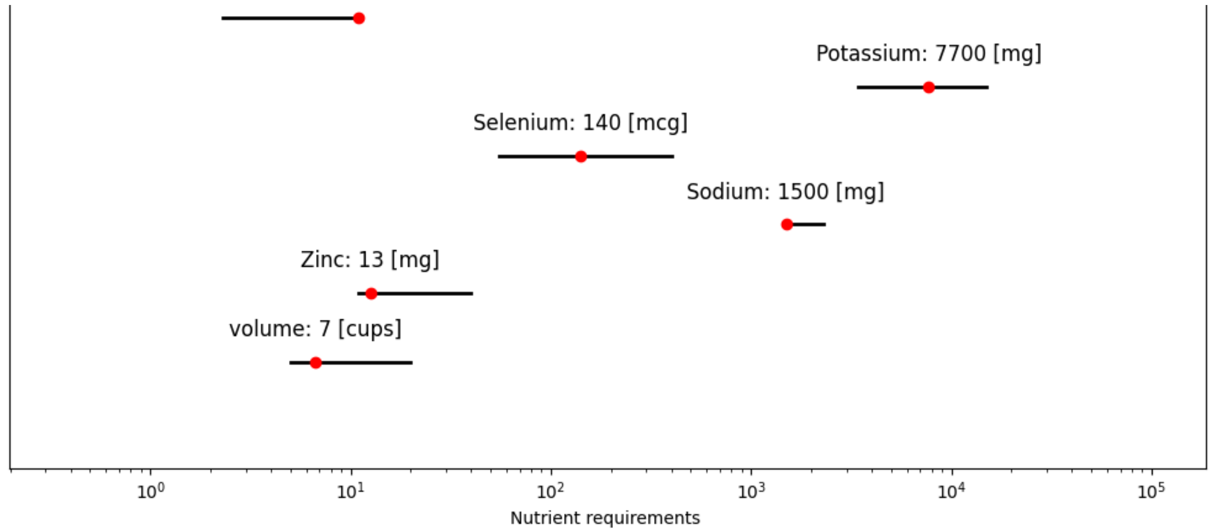
```python
# visualize the output values
from matplotlib import pyplot
from numpy import log

fig = pyplot.figure(figsize=(10, 20))
for y, (consName, content) in enumerate(reversed(constraint_values.items())):
    print(consName)
    # start, mid, end = log(content["lb"]), log(content["val"]), log(content
    start, mid, end = content["lb"], content["val"], content["ub"]
    end = 100000 if end == inf else end
    pyplot.plot([start, mid], [y,y], 'k-', linewidth=2)  # Lower segment
    pyplot.plot([mid, end], [y,y], 'k-', linewidth=2)  # Upper segment
    pyplot.scatter([mid], [y], color='red', zorder=3)  # Midpoint as a red c
    if consName == "volume":  text = consName ; unit = "[cups]"
    else:  text = consName ; unit = f"[{nutrition[consName.replace('_', ' ')
    # pyplot.text(start, y + 0.3, f"{round(start)} {unit}", fontsize=12, ha=
    pyplot.text(mid, y + 0.3, f"{text}: {round(mid)} {unit}", fontsize=12, h
    # pyplot.text(end, y + 0.3, f"{round(end)} {unit}", fontsize=12, ha='cen
pyplot.xscale("log")
pyplot.xlabel("Nutrient requirements ")
ax = pyplot.gca()
# pyplot.ylabel("Nutrient")
# pyplot.yaxis.set_visible(False)
ax.get_yaxis().set_visible(False)
pyplot.title("Optimized nutrient requirements for each nutrient")
pyplot.tight_layout()
pyplot.savefig("optimized_diet.png")
```

```
{'Blueberries': 2.492625928208191,
 'Carrots': 5.0,
 'Celery': 1.5587175120375159,
 'Collard': 1.970293249255337,
 'Corn': 1.7334239720014133,
 'Cucumber': 0.7534622730837051,
 'Flax_seeds': 0.3832712518868228,
 'Pinto_beans': 5.0,
 'Raspberries': 1.1830876190406283}
9.675305760373956
Constraint Carbohydrate has 358.96629815224026 from 331.0-478.0
Constraint Total_Fiber has 130.29962752215894 from 41.0-10000.0
Constraint Protein has 101.55059568691736 from 100.0-150.0
Constraint Fat has 65.0 from 65.0-114.0
Constraint Saturated_fatty_acids has 13.660795530865935 from 0.0-33.3
Constraint Linolenic_Acid has 13.295907478471745 from 1.6-10000.0
Constraint Linoleic_Acid has 33.51419410365214 from 17.0-10000.0
Constraint Total_Water has 1.467674341998467 from 0.37-3.7
Constraint Energy has 3200.0 from 2400.0-3200.0
Constraint Vitamin_A has 2280.945042231542 from 900.0-3000.0
Constraint Vitamin_C has 256.8785564237468 from 90.0-2000.0
Constraint Vitamin_B6 has 3.3743270784002437 from 1.3-100.0
Constraint Vitamin_E has 15.0 from 15.0-1000.0
Constraint Vitamin_K has 1167.5766110581428 from 120.0-10000.0
Constraint Thiamin has 3.382516063054318 from 1.2-10000.0
Constraint Riboflavin has 2.6708488030686066 from 1.3-10000.0
Constraint Folate has 2219.1721315046516 from 400.0-3000.0
Constraint Niacin has 23.64855257470022 from 16.0-35.0
Constraint Choline has 550.0 from 550.0-10000.0
Constraint Pantothenic_Acid has 5.936462991005018 from 5.0-10000.0
Constraint Biotin has 30.0 from 30.0-10000.0
Constraint Carotenoids has 18862.41393604274 from 0.0-100000.0
Constraint Calcium has 1269.4101949578724 from 1000.0-2500.0
Constraint Copper has 4.354587539569957 from 0.9-10.0
Constraint Iron has 24.9609015799035 from 8.0-45.0
Constraint Magnesium has 951.7629203393967 from 400.0-10000.0
Constraint Manganese has 11.0 from 2.3-11.0
Constraint Potassium has 7699.719202487697 from 3400.0-15000.0
Constraint Selenium has 140.09674895205563 from 55.0-400.0
Constraint Sodium has 1500.0 from 1500.0-2300.0
Constraint Zinc has 12.550853841296764 from 11.0-40.0
Constraint volume has 6.637564185578465 from 5-20
```

```
{'Carbohydrate': {'lb': 331.0, 'val': 358.96629815224026, 'ub': 478.0},
 'Total_Fiber': {'lb': 41.0, 'val': 130.29962752215894, 'ub': 10000.0},
 'Protein': {'lb': 100.0, 'val': 101.55059568691736, 'ub': 150.0},
 'Fat': {'lb': 65.0, 'val': 65.0, 'ub': 114.0},
 'Saturated_fatty_acids': {'lb': 0.0, 'val': 13.660795530865935, 'ub': 33.
3},
 'Linolenic_Acid': {'lb': 1.6, 'val': 13.295907478471745, 'ub': 10000.0},
 'Linoleic_Acid': {'lb': 17.0, 'val': 33.51419410365214, 'ub': 10000.0},
 'Total_Water': {'lb': 0.37, 'val': 1.467674341998467, 'ub': 3.7},
 'Energy': {'lb': 2400.0, 'val': 3200.0, 'ub': 3200.0},
 'Vitamin_A': {'lb': 900.0, 'val': 2280.945042231542, 'ub': 3000.0},
 'Vitamin_C': {'lb': 90.0, 'val': 256.8785564237468, 'ub': 2000.0},
 'Vitamin_B6': {'lb': 1.3, 'val': 3.3743270784002437, 'ub': 100.0},
 'Vitamin_E': {'lb': 15.0, 'val': 15.0, 'ub': 1000.0},
 'Vitamin_K': {'lb': 120.0, 'val': 1167.5766110581428, 'ub': 10000.0},
 'Thiamin': {'lb': 1.2, 'val': 3.382516063054318, 'ub': 10000.0},
 'Riboflavin': {'lb': 1.3, 'val': 2.6708488030686066, 'ub': 10000.0},
 'Folate': {'lb': 400.0, 'val': 2219.1721315046516, 'ub': 3000.0},
 'Niacin': {'lb': 16.0, 'val': 23.64855257470022, 'ub': 35.0},
 'Choline': {'lb': 550.0, 'val': 550.0, 'ub': 10000.0},
 'Pantothenic_Acid': {'lb': 5.0, 'val': 5.936462991005018, 'ub': 10000.0},
 'Biotin': {'lb': 30.0, 'val': 30.0, 'ub': 10000.0},
 'Carotenoids': {'lb': 0.0, 'val': 18862.41393604274, 'ub': 100000.0},
 'Calcium': {'lb': 1000.0, 'val': 1269.4101949578724, 'ub': 2500.0},
 'Copper': {'lb': 0.9, 'val': 4.354587539569957, 'ub': 10.0},
 'Iron': {'lb': 8.0, 'val': 24.9609015799035, 'ub': 45.0},
 'Magnesium': {'lb': 400.0, 'val': 951.7629203393967, 'ub': 10000.0},
 'Manganese': {'lb': 2.3, 'val': 11.0, 'ub': 11.0},
 'Potassium': {'lb': 3400.0, 'val': 7699.719202487697, 'ub': 15000.0},
 'Selenium': {'lb': 55.0, 'val': 140.09674895205563, 'ub': 400.0},
 'Sodium': {'lb': 1500.0, 'val': 1500.0, 'ub': 2300.0},
 'Zinc': {'lb': 11.0, 'val': 12.550853841296764, 'ub': 40.0},
 'volume': {'lb': 5, 'val': 6.637564185578465, 'ub': 20}}
```

volume
Zinc
Sodium
Selenium
Potassium
Manganese
Magnesium
Iron
Copper
Calcium
Carotenoids
Biotin
Pantothenic_Acid
Choline
Niacin
Folate
Riboflavin
Thiamin
Vitamin_K
Vitamin_E
Vitamin_B6
Vitamin_C
Vitamin_A
Energy
Total_Water
Linoleic_Acid
Linolenic_Acid
Saturated_fatty_acids
Fat
Protein
Total_Fiber
Carbohydrate


Optimized nutrient requirements for each nutrient
Carbohydrate: 359 [grams]
Total_Fiber: 130 [grams]
Protein: 102 [grams]
Fat: 65 [grams]
Saturated_fatty_acids: 14 [grams]
Linolenic_Acid: 13 [grams]
Linoleic_Acid: 34 [grams]
Total_Water: 1 [liters]
Energy: 3200 [kcal]
Vitamin_A: 2281 [mcg]
Vitamin_C: 257 [mg]
Vitamin_B6: 3 [mg]
Vitamin_E: 15 [mg]
Vitamin_K: 1168 [mcg]
Thiamin: 3 [mg]
Riboflavin: 3 [mg]
Folate: 2219 [mcg]
Niacin: 24 [mg]
Choline: 550 [mg]
Pantothenic_Acid: 6 [mg]
Biotin: 30 [mcg]
Carotenoids: 18862 [mg]
Calcium: 1269 [mg]
Copper: 4 [mg]
Iron: 25 [mg]
Magnesium: 952 [mg]
Manganese: 11 [mg]


Potassium: 7700 [mg]
Selenium: 140 [mcg]
Sodium: 1500 [mg]
Zinc: 13 [mg]
volume: 7 [cups]
10^0
10^1
10^2
10^3
10^4
10^5
Nutrient requirements

In [ ]: