Health Data Analysis Project - Data Cleaning

Peter Gray

August 29, 2025

Produced with Quarto and Python

Table of contents

List of Tables

List of Figures

List of Python and R Packages

Python	R
NumPy	tidyverse
Pandas	ggplot2
Matplotlib	gtsummary
lifelines	
random	
miceforest	

Data Cleaning

Import Data

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Don't do Scientific notation
pd.options.display.float_format = "{:.2f}".format

demographics = pd.read_sas(
    r"/home/pgr16/Documents/Data_Analysis/Health_Data_Analysis/Data/DEMO_L.xpt"
)

day_1_intake = pd.read_sas(
    r"/home/pgr16/Documents/Data_Analysis/Health_Data_Analysis/Data/DR1TOT_L.xpt"
)

food_codes = pd.read_sas(
    r"/home/pgr16/Documents/Data_Analysis/Health_Data_Analysis/Data/DRXFCD_L.xpt"
)
```

Clean Data

Select Columns of Interest

```
dm_columns  = ["SEQN", "SDDSRVYR", "RIDSTATR", "RIDAGEYR", "RIAGENDR",  "RIDRETH1", "DMDBORN4", "

demographics = demographics[dm_columns]

intake_columns = ["SEQN", "WTDRD1", "WTDR2D", "DR1DRSTZ", "DRABF", "DR1DAY", "DR1MRESP", "DBD100"

day_1_intake = day_1_intake[intake_columns]

food_code_columns = ["DRXFDCD", "DRXFCSD"]

food_codes = food_codes[food_code_columns]
```

Rename Data Columns

```
demographics.rename(columns={"SEQN": "SUBJID", "SDDSRVYR" : "CYCLE", "RIDAGEYR" : "AGE_YEARS_SCRE

day_1_intake.rename(columns={"SEQN": "SUBJID", "WTDRD1" : "SAMPLE_WEIGHT_DAY1", "WTDR2D" : "SAMPL

food_codes.rename(columns = {"DRXFDCD": "FOOD_CODE", "DRXFCSD" : "FOOD_NAME"}, inplace=True)

#Remove Scientific Notation
food_codes['FOOD_CODE'] = food_codes['FOOD_CODE'].astype(str)
```

Rename Column Values

```
# Demogrpahics

demographics["INTERVIEW_STATUS"] = np.where(demographics["INTERVIEW_STATUS"]  == 1, "Inteviewed C
demographics["INTERVIEW_STATUS"] = np.where(demographics["INTERVIEW_STATUS"]  == "2.0", "Intevie


demographics["GENDER"] = np.where(demographics["GENDER"]  == 1, "Male" , demographics["GENDER"])
demographics["GENDER"] = np.where(demographics["GENDER"]  == "2.0", "Female" , demographics["GEND

demographics["RACE"] = np.where(demographics["RACE"]  == 1, "Mexican American" , demographics["RA
demographics["RACE"] = np.where(demographics["RACE"]  == "2.0", "Other Hispanic" , demographics["
demographics["RACE"] = np.where(demographics["RACE"]  == "3.0", "Non-Hispanic White" , demographi
demographics["RACE"] = np.where(demographics["RACE"]  == "4.0", "Non-Hispanic Black" , demographi
demographics["RACE"] = np.where(demographics["RACE"]  == "5.0", "Other Race - Including Multi-Rac


demographics["BIRTH_COUNTRY"] = np.where(demographics["BIRTH_COUNTRY"]  == 1, "US" , demographics
demographics["BIRTH_COUNTRY"] = np.where(demographics["BIRTH_COUNTRY"]  == "2.0", "Others" , demo
demographics["BIRTH_COUNTRY"] = np.where(demographics["BIRTH_COUNTRY"]  == "77", "Refused" , demo
demographics["BIRTH_COUNTRY"] = np.where(demographics["BIRTH_COUNTRY"]  == "99", "Don't know" , d

demographics["EDUCATION"] = np.where(demographics["EDUCATION"]  == 1, "Less than 9th grade" , dem
demographics["EDUCATION"] = np.where(demographics["EDUCATION"]  == "2.0", "9-11th grade (Includes
demographics["EDUCATION"] = np.where(demographics["EDUCATION"]  == "3.0", "High school graduate/G
demographics["EDUCATION"] = np.where(demographics["EDUCATION"]  == "4.0", "Some college or AA deg
demographics["EDUCATION"] = np.where(demographics["EDUCATION"]  == "5.0", "College graduate or ab
demographics["EDUCATION"] = np.where(demographics["EDUCATION"]  == "7.0", "Refused" , demographic
demographics["EDUCATION"] = np.where(demographics["EDUCATION"]  == "9.0", np.nan , demographics["
demographics["EDUCATION"] = np.where(demographics["EDUCATION"]  == "nan", np.nan , demographics["


demographics["MARITAL_STATUS"] = np.where(demographics["MARITAL_STATUS"]  == 1, "Married/Living w
demographics["MARITAL_STATUS"] = np.where(demographics["MARITAL_STATUS"]  == "2.0", "Widowed/Divo
demographics["MARITAL_STATUS"] = np.where(demographics["MARITAL_STATUS"]  == "3.0", "Never marrie
demographics["MARITAL_STATUS"] = np.where(demographics["MARITAL_STATUS"]  == "77.0", "Refused" ,
demographics["MARITAL_STATUS"] = np.where(demographics["MARITAL_STATUS"]  == "99", np.nan , demog
demographics["MARITAL_STATUS"] = np.where(demographics["MARITAL_STATUS"]  == "nan", np.nan , demo


# Day 1 Intake

day_1_intake["RECALL_RELIABILITY"] = np.where(day_1_intake["RECALL_RELIABILITY"] == 1, "Reliable
day_1_intake["RECALL_RELIABILITY"] = np.where(day_1_intake["RECALL_RELIABILITY"] == "2.0", "Not r
day_1_intake["RECALL_RELIABILITY"] = np.where(day_1_intake["RECALL_RELIABILITY"] == "3.0", "Repor
day_1_intake["RECALL_RELIABILITY"] = np.where(day_1_intake["RECALL_RELIABILITY"] == "4.0", "Repor
day_1_intake["RECALL_RELIABILITY"] = np.where(day_1_intake["RECALL_RELIABILITY"] == "5.0", "Not d
```

```
day_1_intake["INFANT_BREAST_FED"] = np.where(day_1_intake["INFANT_BREAST_FED"] == 1, "Yes", day_1
day_1_intake["INFANT_BREAST_FED"] = np.where(day_1_intake["INFANT_BREAST_FED"] == "2.0", "No", da
day_1_intake["INFANT_BREAST_FED"] = np.where(day_1_intake["INFANT_BREAST_FED"]  == "nan", np.nan


day_1_intake["INTAKE_DAY_WEEK"] = np.where(day_1_intake["INTAKE_DAY_WEEK"]  == 1, "Sunday" , day_
day_1_intake["INTAKE_DAY_WEEK"] = np.where(day_1_intake["INTAKE_DAY_WEEK"] == "2.0", "Monday", da
day_1_intake["INTAKE_DAY_WEEK"] = np.where(day_1_intake["INTAKE_DAY_WEEK"] == "3.0", "Tuesday", d
day_1_intake["INTAKE_DAY_WEEK"] = np.where(day_1_intake["INTAKE_DAY_WEEK"] == "4.0", "Wednesday",
day_1_intake["INTAKE_DAY_WEEK"] = np.where(day_1_intake["INTAKE_DAY_WEEK"] == "5.0", "Thursday",
day_1_intake["INTAKE_DAY_WEEK"] = np.where(day_1_intake["INTAKE_DAY_WEEK"] == "6.0", "Friday", da
day_1_intake["INTAKE_DAY_WEEK"] = np.where(day_1_intake["INTAKE_DAY_WEEK"] == "7.0", "Saturday",
day_1_intake["INTAKE_DAY_WEEK"] = np.where(day_1_intake["INTAKE_DAY_WEEK"]  == "nan", np.nan , da


relationship_map = {
    1: "SP",
    2: "Mother of SP",
    3: "Father of SP",
    5: "Spouse of SP",
    6: "Child of SP",
    7: "Grandparent of SP",
    8: "Friend, partner, non-relative",
    9: "Translator, not a HH member",
    10: "Child care provider, caretaker",
    11: "Other relative",
    77: "Refused",
    99: np.nan
}

day_1_intake["MAIN_RESPONDENT"] = day_1_intake["MAIN_RESPONDENT"].map(relationship_map)


salt_usage_map = {
    1: "Ordinary salt",
    2: "Lite salt",
    3: "Salt substitute",
    4: "Doesn't use or add salt products at the table",
    91: "Other",
    99:  np.nan}

day_1_intake["SALT_AT_TABLE"] = day_1_intake["SALT_AT_TABLE"].map(salt_usage_map)


salt_usage_table_map = {
    1: "Rarely",
    2: "Occasionally",
```

```
    3: "Very often",
    7: "Refused",
    9: "Don't know",
    99:  np.nan}

day_1_intake["SALT_AT_TABLE"] = day_1_intake["SALT_AT_TABLE"].map(salt_usage_table_map)

salt_usage_prep_map = {
    1: "Never",
    2: "Rarely",
    3: "Occasionally",
    4: "Very often",
    9: "Don't know",
    99:  np.nan}

day_1_intake["SALT_IN_PREP"] = day_1_intake["SALT_IN_PREP"].map(salt_usage_prep_map)

day_1_intake["SPECIAL_DIET"] = np.where(day_1_intake["SPECIAL_DIET"]  == 1, "Yes" , day_1_intake[
day_1_intake["SPECIAL_DIET"] = np.where(day_1_intake["SPECIAL_DIET"]  == "2.0", "No" , day_1_inta

day_1_intake["WEIGHT_LOSS_DIET"] = np.where(day_1_intake["WEIGHT_LOSS_DIET"]  == 1, "Yes" , day_1
day_1_intake["WEIGHT_LOSS_DIET"] = np.where(day_1_intake["WEIGHT_LOSS_DIET"]  == "nan", np.nan ,

day_1_intake["LOW_FAT_DIET"] = np.where(day_1_intake["LOW_FAT_DIET"]  == 2, "Yes" , day_1_intake[
day_1_intake["LOW_FAT_DIET"] = np.where(day_1_intake["LOW_FAT_DIET"]  == "nan", np.nan , day_1_in

day_1_intake["LOW_SALT_DIET"] = np.where(day_1_intake["LOW_SALT_DIET"]  == 3, "Yes" , day_1_intak
day_1_intake["LOW_SALT_DIET"] = np.where(day_1_intake["LOW_SALT_DIET"]  == "nan", np.nan , day_1_

day_1_intake["LOW_SUGER_DIET"] = np.where(day_1_intake["LOW_SUGER_DIET"]  == 4 ,"Yes" , day_1_int
day_1_intake["LOW_SUGER_DIET"] = np.where(day_1_intake["LOW_SUGER_DIET"]  == "nan", np.nan , day_

day_1_intake["LOW_FIBER_DIET"] = np.where(day_1_intake["LOW_FIBER_DIET"]  == 5 ,"Yes" , day_1_int
day_1_intake["LOW_FIBER_DIET"] = np.where(day_1_intake["LOW_FIBER_DIET"]  == "nan", np.nan , day_


day_1_intake["HIGH_FIBER_DIET"] = np.where(day_1_intake["HIGH_FIBER_DIET"]  == 6 ,"Yes" , day_1_i
day_1_intake["HIGH_FIBER_DIET"] = np.where(day_1_intake["HIGH_FIBER_DIET"]  == "nan", np.nan , da

day_1_intake["CELIAC_DIET"] = np.where(day_1_intake["CELIAC_DIET"]  == 6 ,"Yes" , day_1_intake["C
day_1_intake["CELIAC_DIET"] = np.where(day_1_intake["CELIAC_DIET"]  == "nan", np.nan , day_1_inta
```

Standardise Measurements

```
day_1_intake["TOTAL_WATER_DRANK"] = day_1_intake["TOTAL_WATER_DRANK"]/1000
```

```
day_1_intake.rename(columns ={"TOTAL_WATER_DRANK": "TOTAL_WATER_DRANK_LITRE"}, inplace= True)
```

```
master = pd.merge(demographics, day_1_intake, how="right", on="SUBJID")

import os

master.to_csv(
    r"/home/pgr16/Documents/Data_Analysis/Health_Data_Analysis/Data/Master.csv"
)
```

Create Age Groups

```
conditions = [
    master["AGE_YEARS_SCREENING"] < 18,
    (master["AGE_YEARS_SCREENING"] >= 18) & (master["AGE_YEARS_SCREENING"] <= 35),
    (master["AGE_YEARS_SCREENING"] > 35) & (master["AGE_YEARS_SCREENING"] <= 60),
    master["AGE_YEARS_SCREENING"] > 60
]

choices = [
    "Less than 18",
    "18-35",
    "36-60",
    "Above 60"
]

master["AGE_GROUP"] = np.select(conditions, choices, default= "NaN")
```

Exploratory Analysis

```
plt.figure()
sns.heatmap(master.isnull(), cbar=False, cmap='viridis')
plt.xlabel('Columns')
plt.ylabel('Rows')
plt.show()
```

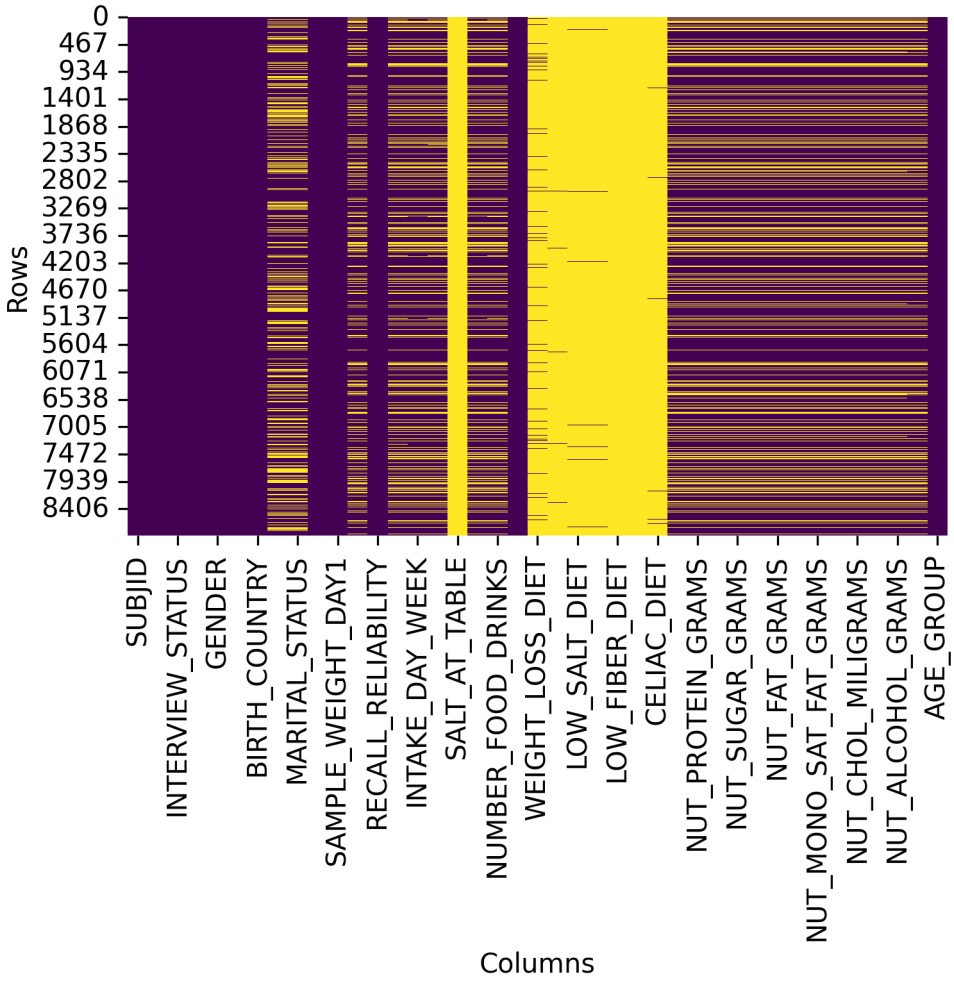
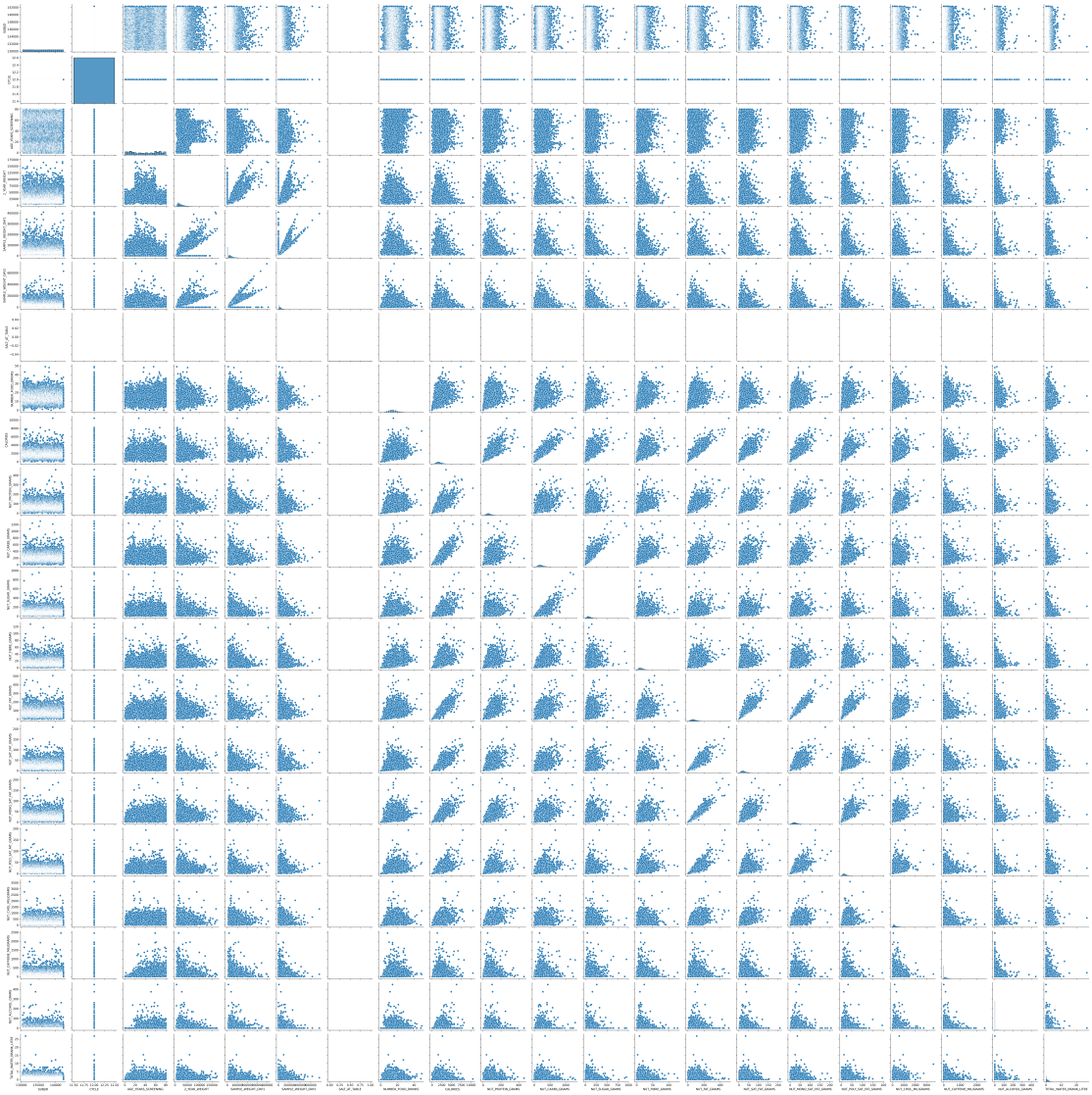


Figure 1: Missing Values Heatmap

```
plt.figure()
sns.pairplot(master)
plt.show()
```

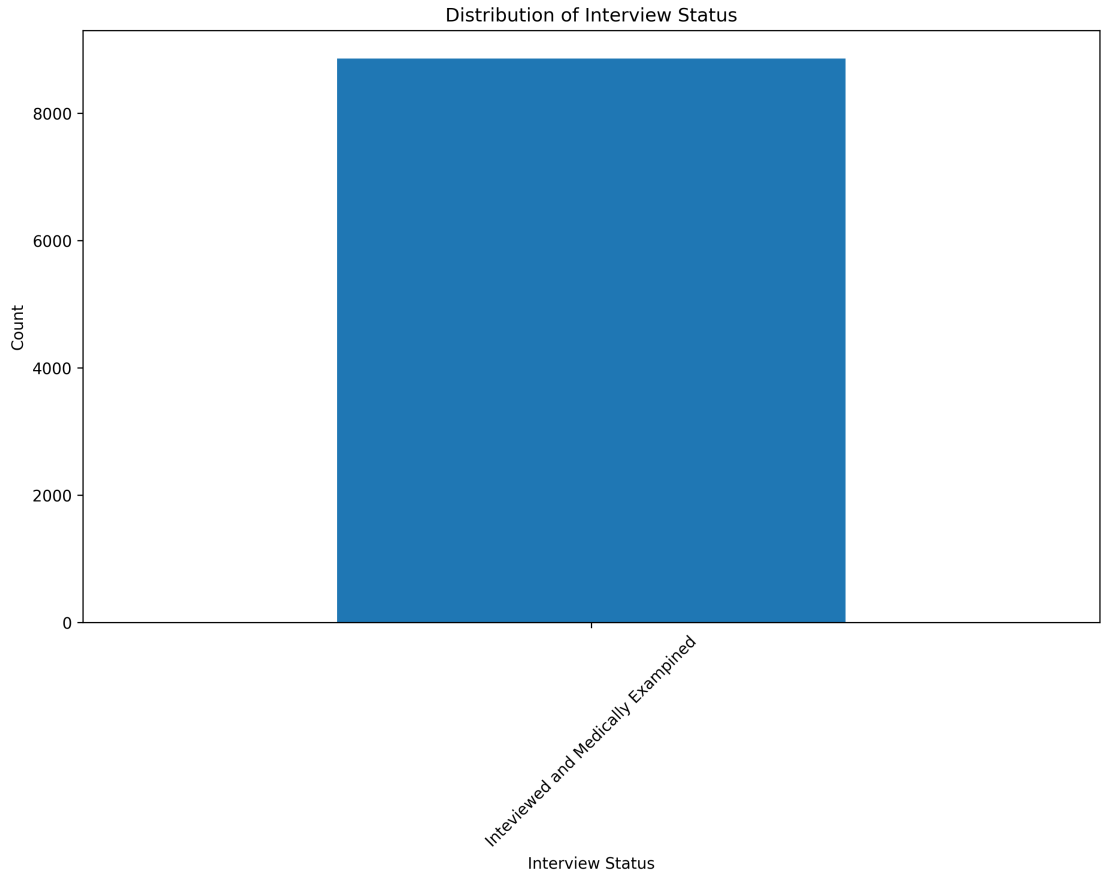
```
<Figure size 1650x1050 with 0 Axes>
```

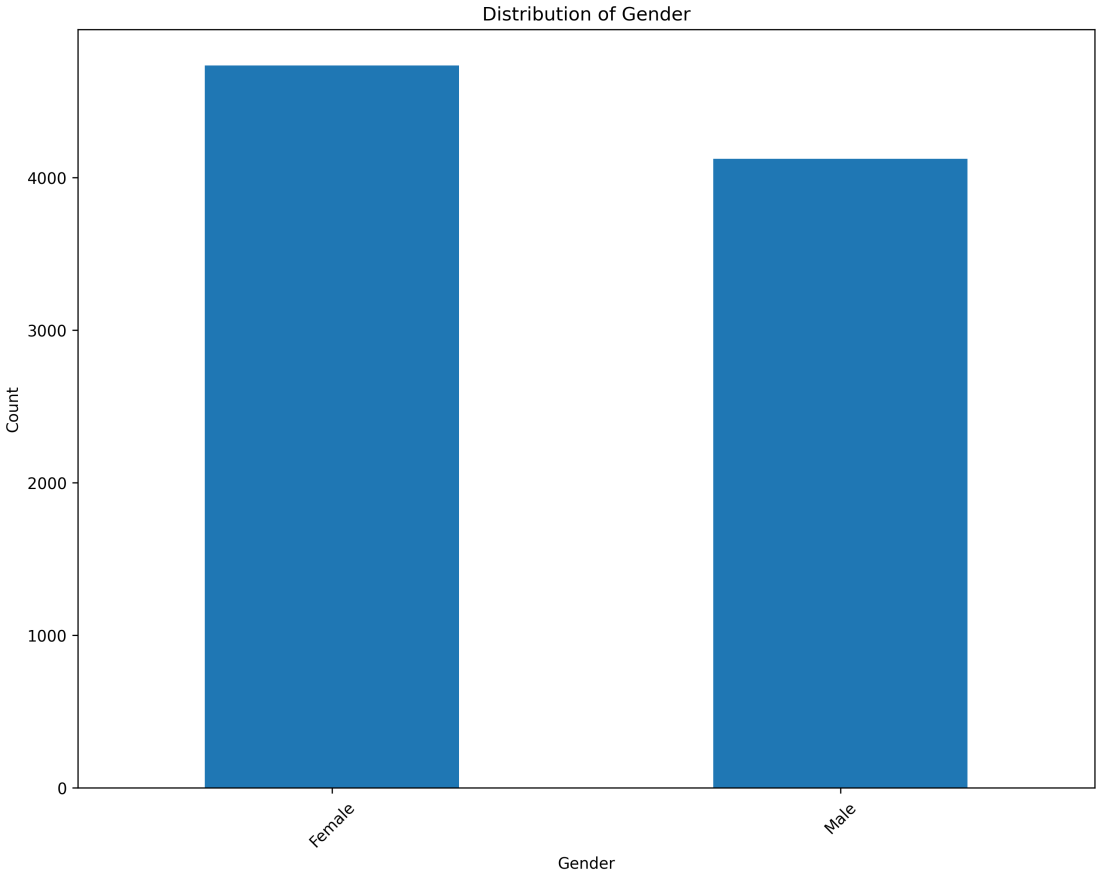
Pairpolot of Data

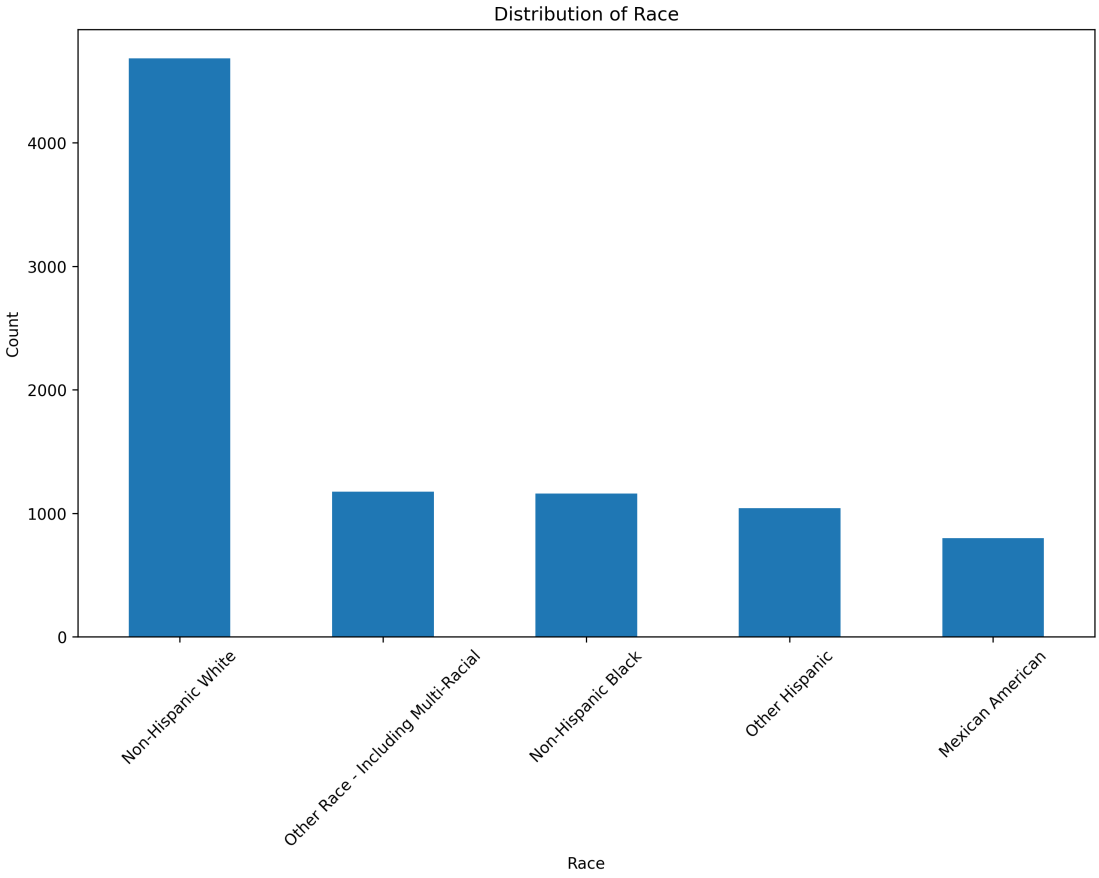


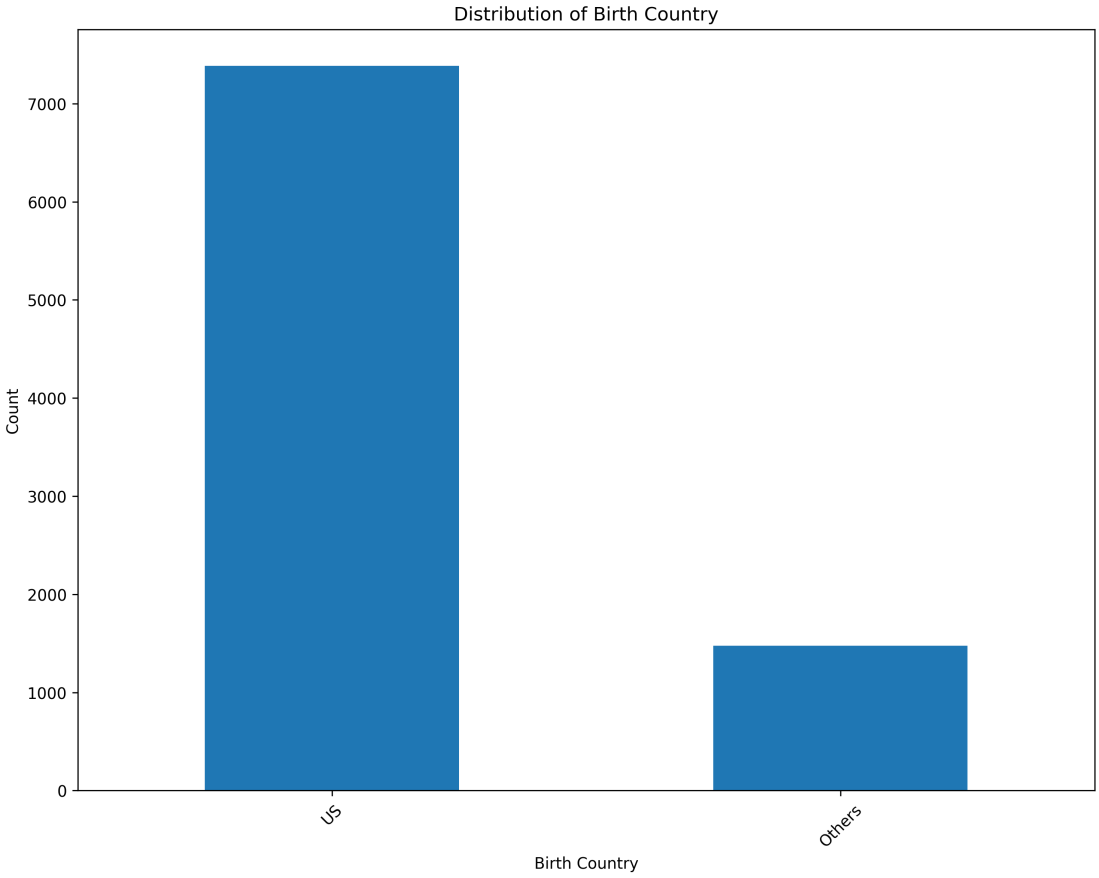
Visualisation of Categorical Variables

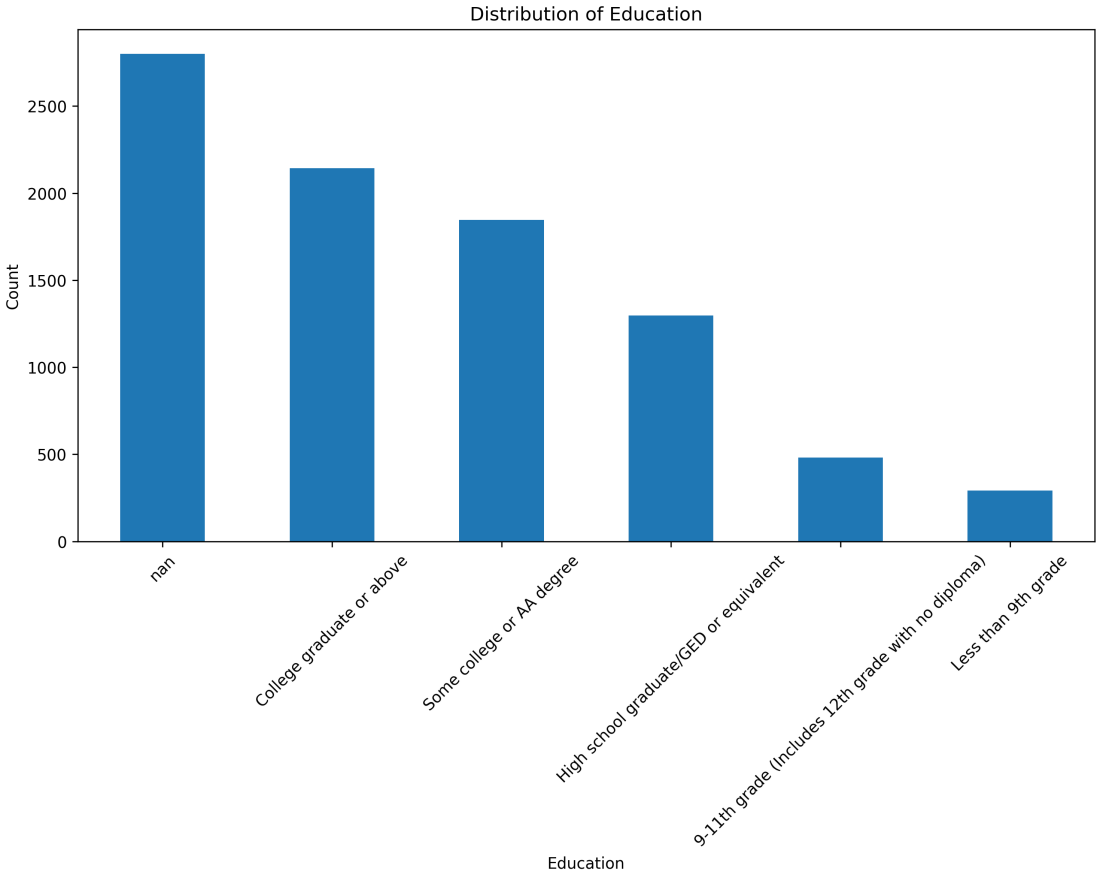
```
categorical_columns = [
    col for col in master.select_dtypes(include=["object", "category"]).columns
    if not col.endswith("_DIET")
]



for col in categorical_columns:

    plt.figure(figsize=(10, 8))
    master[col].value_counts(dropna=False).plot(kind='bar')
    plt.title(f"Distribution of {col.replace('_', ' ').title()}")
    plt.ylabel("Count")
    plt.xlabel(f"{col.replace('_', ' ').title()}")
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

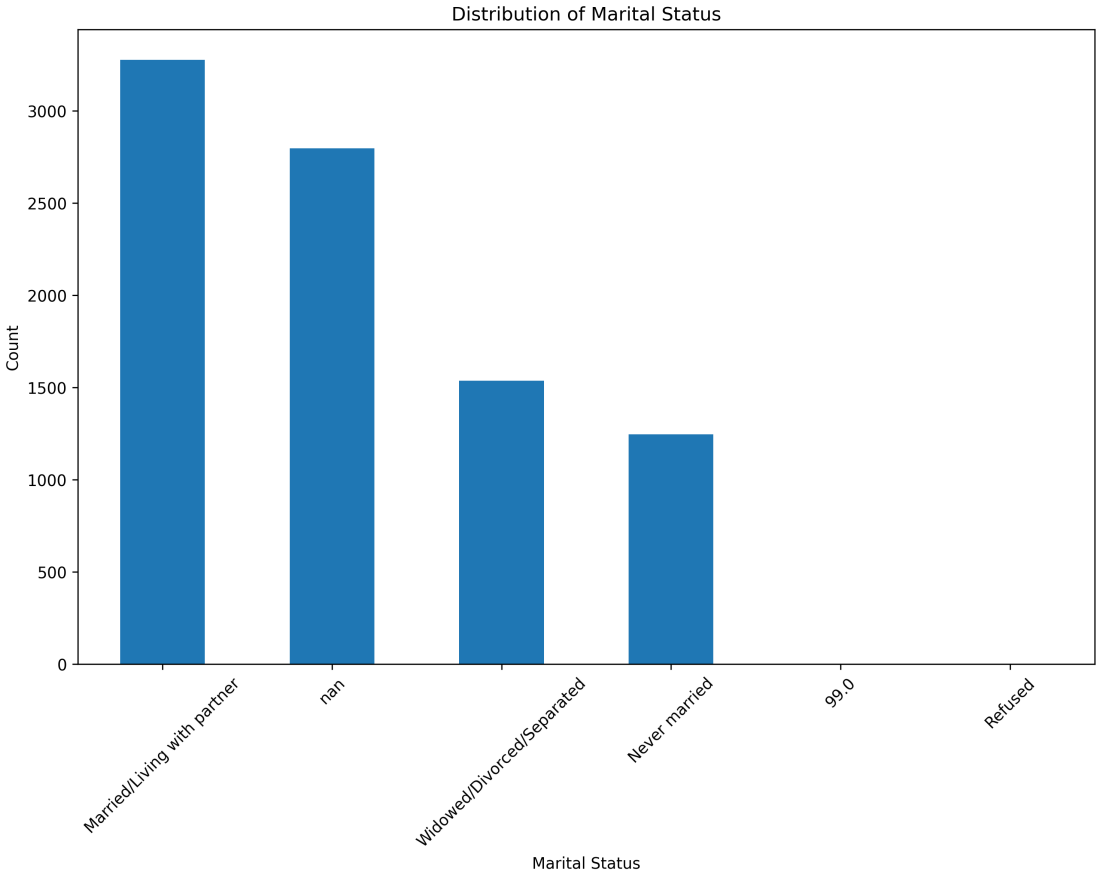


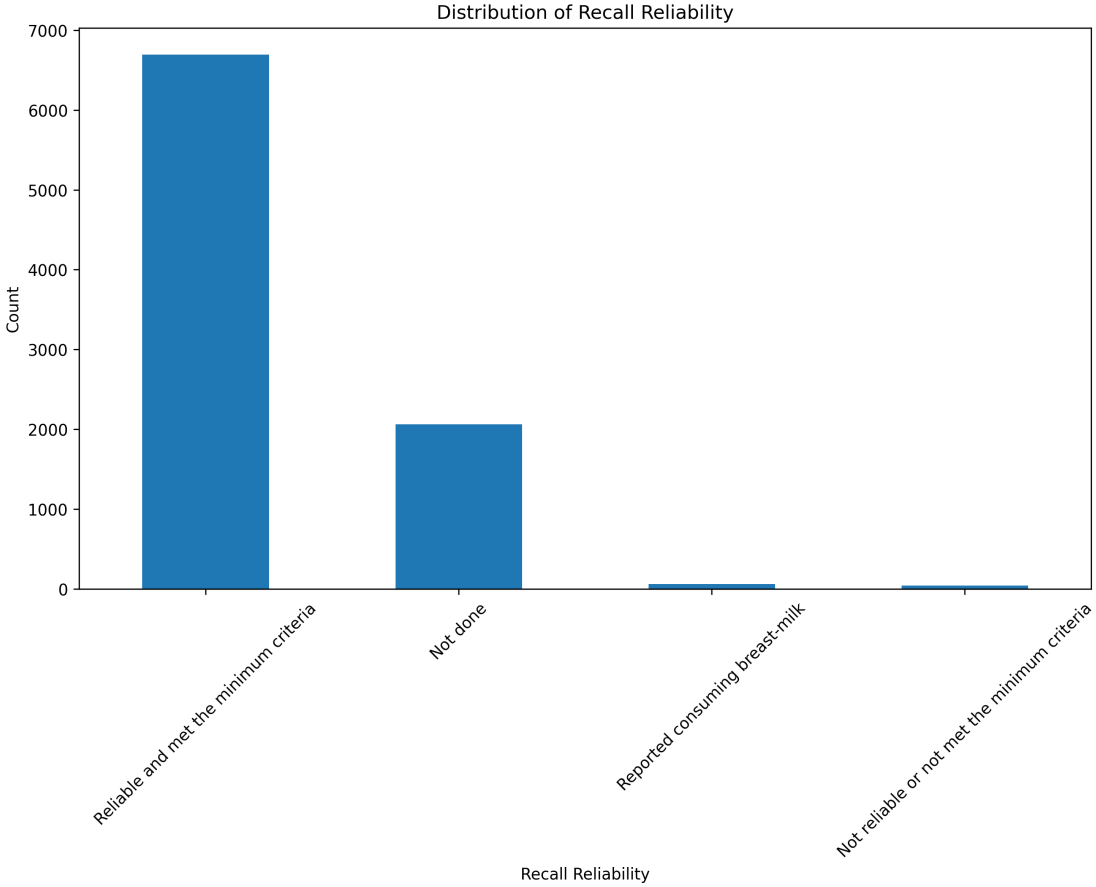

Distribution of Gender
Count
0
1000
2000
3000
4000
Female
Male
Gender

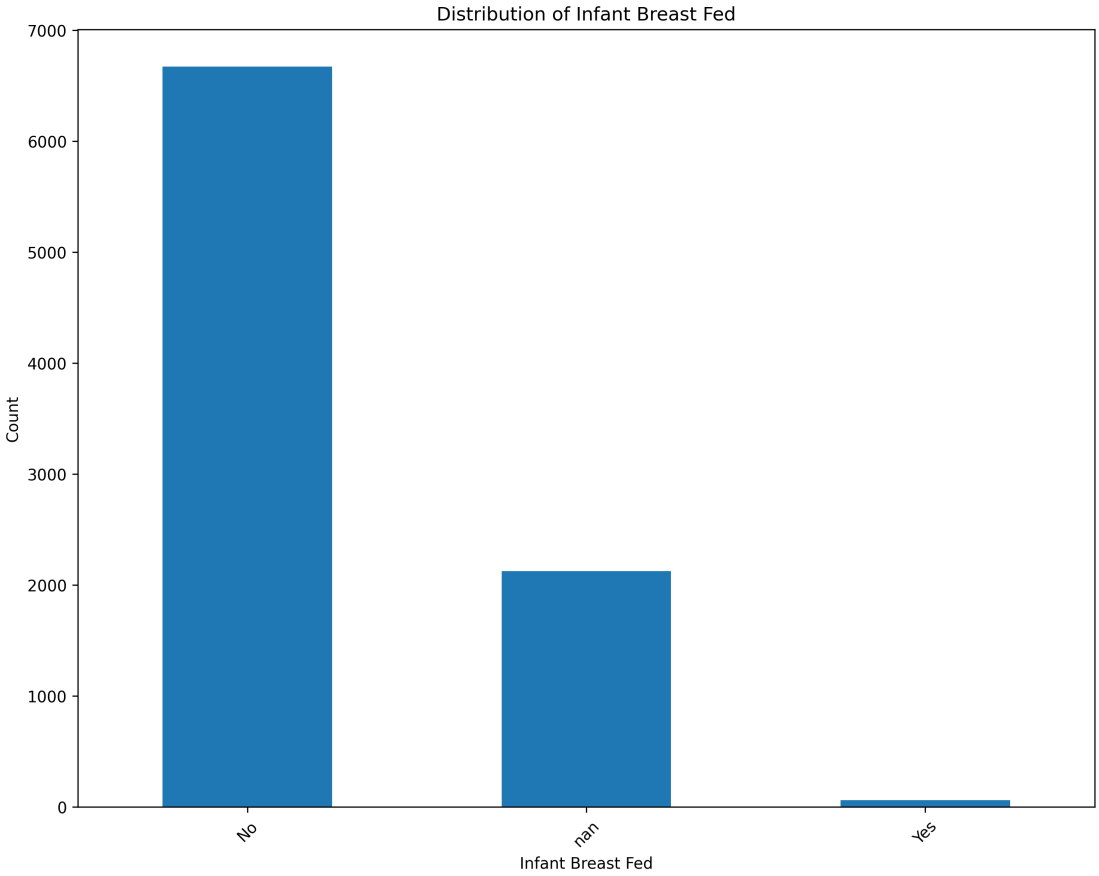

Distribution of Race
Count
4000
3000
2000
1000
0
Non-Hispanic White
Other Race - Including Multi-Racial
Non-Hispanic Black
Other Hispanic
Mexican American
Race

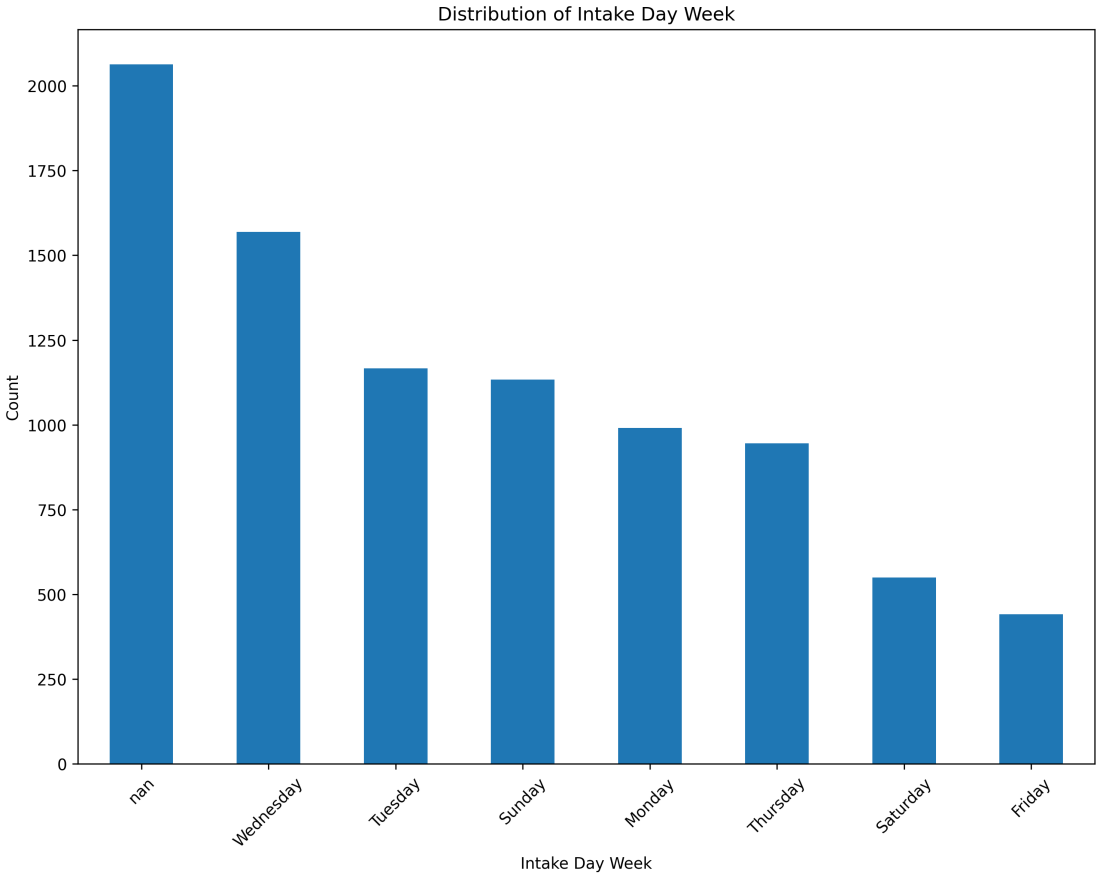

Distribution of Birth Country
Count
0
1000
2000
3000
4000
5000
6000
7000
US
Others
Birth Country

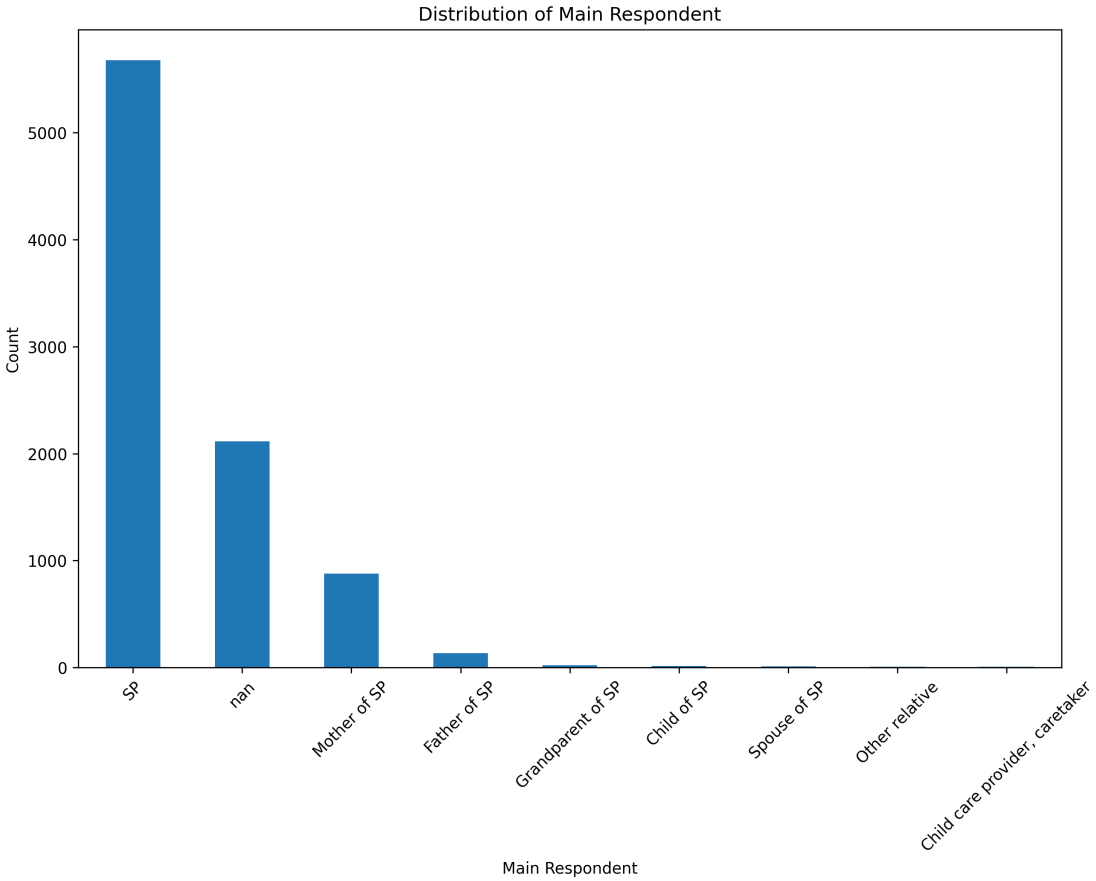

Distribution of Education
Count
2500
2000
1500
1000
500
0
nan
College graduate or above
Some college or AA degree
High school graduate/GED or equivalent
9-11th grade (Includes 12th grade with no diploma)
Less than 9th grade
Education

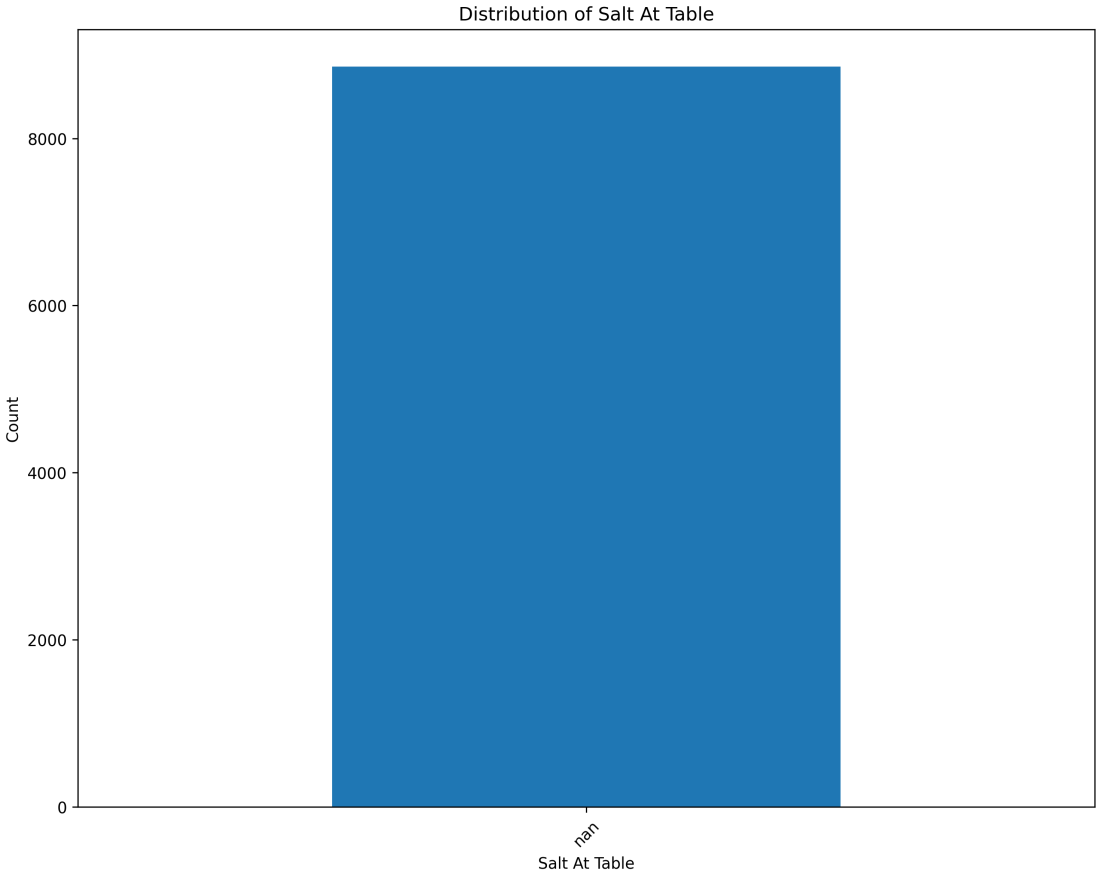

Distribution of Marital Status
Count
3000
2500
2000
1500
1000
500
0
Married/Living with partner
nan
Widowed/Divorced/Separated
Never married
99.0
Refused
Marital Status

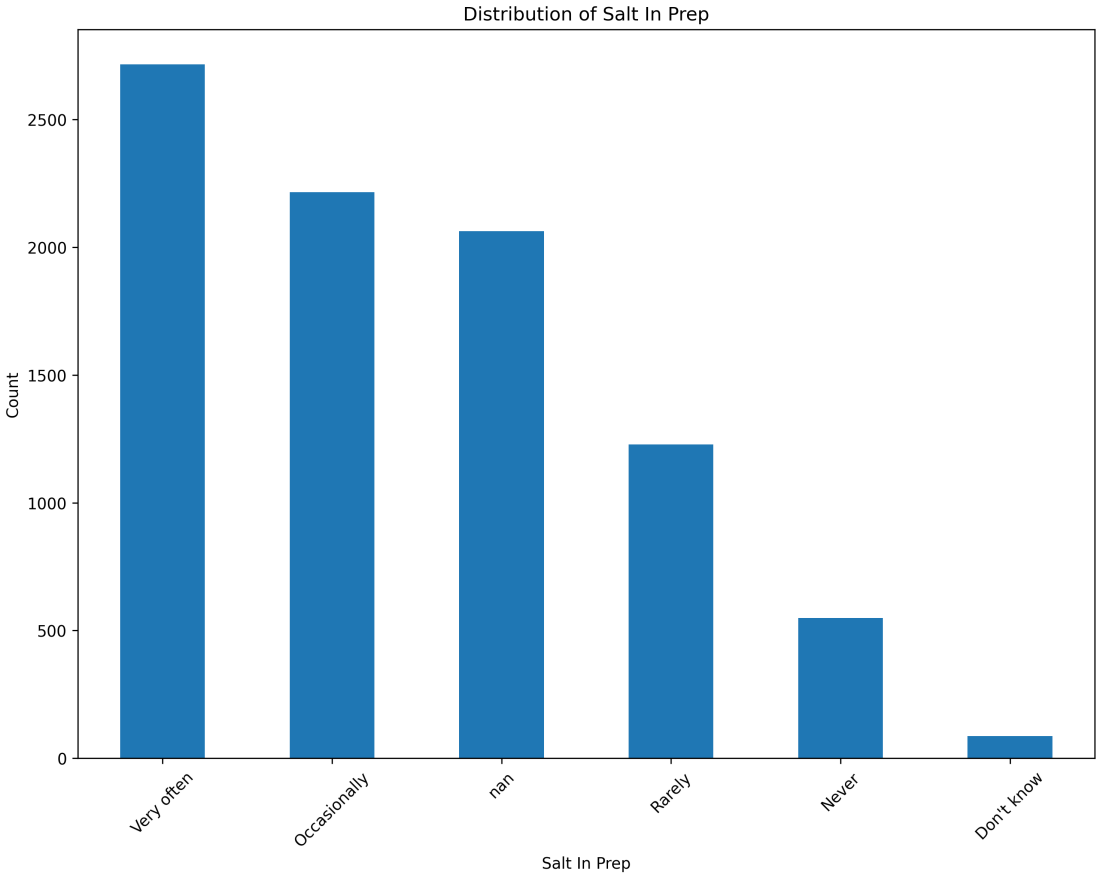

Distribution of Recall Reliability
Count
7000
6000
5000
4000
3000
2000
1000
0
Reliable and met the minimum criteria
Not done
Reported consuming breast-milk
Not reliable or not met the minimum criteria
Recall Reliability

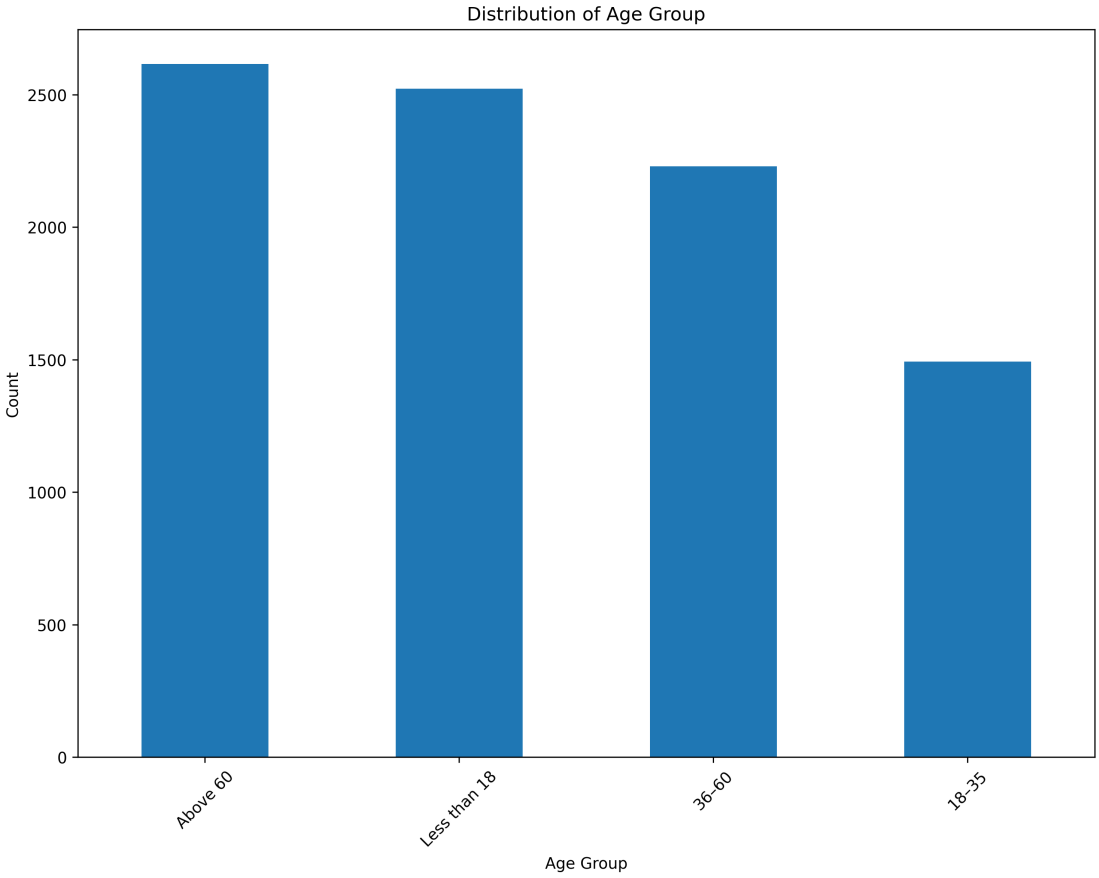

Distribution of Infant Breast Fed
Count
7000
6000
5000
4000
3000
2000
1000
0
No
nan
Yes
Infant Breast Fed


Distribution of Intake Day Week
Count
2000
1750
1500
1250
1000
750
500
250
0
nan
Wednesday
Tuesday
Sunday
Monday
Thursday
Saturday
Friday
Intake Day Week


Distribution of Main Respondent
Count
5000
4000
3000
2000
1000
0
SP
nan
Mother of SP
Father of SP
Grandparent of SP
Child of SP
Spouse of SP
Other relative
Child care provider, caretaker
Main Respondent


Distribution of Salt At Table
Count
8000
6000
4000
2000
0
nan
Salt At Table

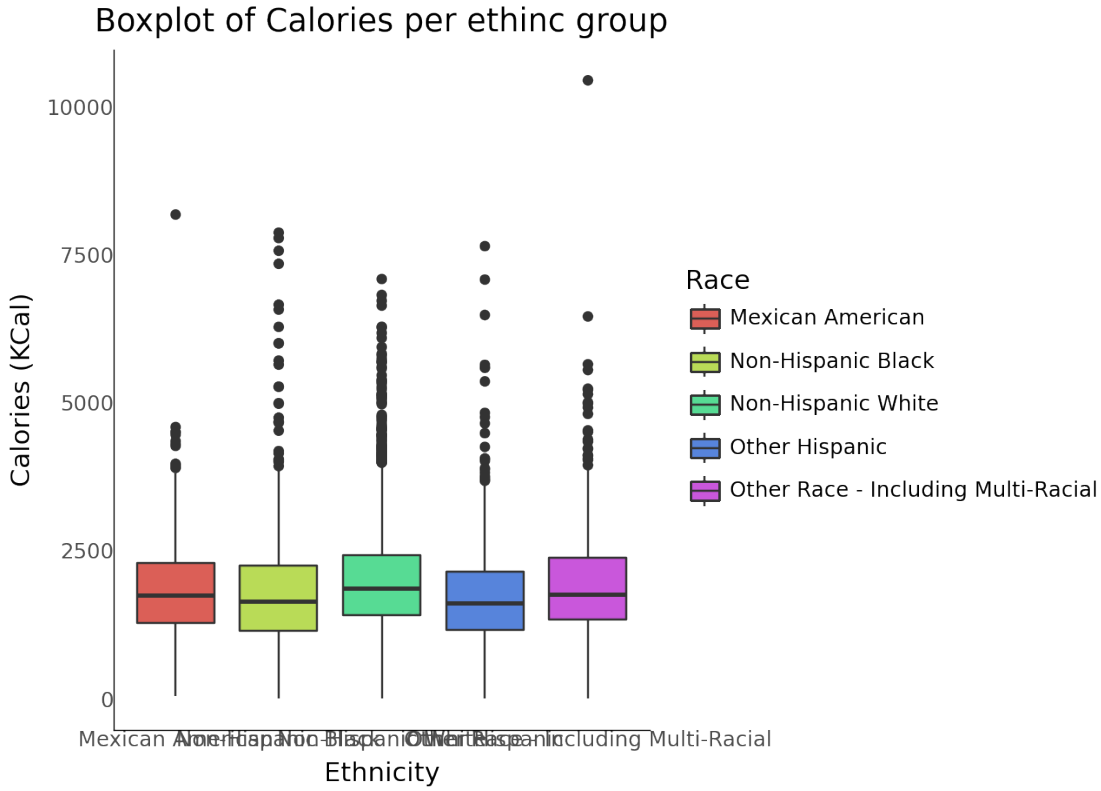

Distribution of Salt In Prep
Count
2500
2000
1500
1000
500
0
Very often
Occasionally
nan
Rarely
Never
Don't know
Salt In Prep


Distribution of Age Group
Count
2500
2000
1500
1000
500
0
Above 60
Less than 18
36-60
18-35
Age Group

Data Analysis

Calorie Intake per Ethnicity

```
from plotnine import *

calorie = master[master["RECALL_RELIABILITY"]== "Reliable and met the minimum criteria"]
calorie = calorie[calorie["INFANT_BREAST_FED"]== "No"]

title = "Boxplot of Calories per ethinc group"
p1 = (
  ggplot(calorie) +
  geom_boxplot(aes(x = "RACE", y = "CALORIES", fill = "RACE")) +
  theme_minimal() +
  labs(title = title,
  y = "Calories (KCal)",
  x = "Ethnicity",
  fill = "Race",
   caption = "Population comes from only the surverys classfied as 'Reliable and met the minimum
  theme(
    axis_line_y=element_line(color="black", size=0.5),
    axis_line_x=element_line(color="black", size=0.5),
    axis_ticks_x=element_text(),
    panel_grid_major=element_blank(),
    panel_grid_minor=element_blank()
  )


)

p1
```

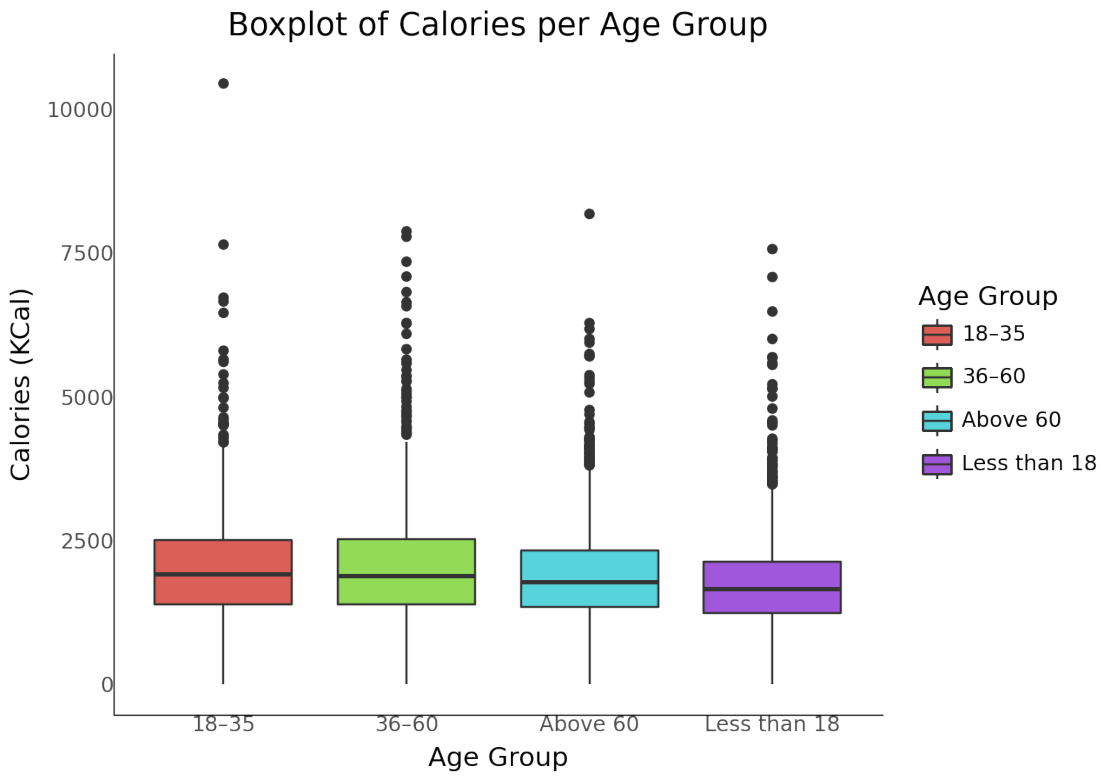


surverys classfied as 'Reliable and met the minimum criteria'

Box Plot of Calorie Intake per Age Group

```
title = "Boxplot of Calories per Age Group"

p2 = (

  ggplot(calorie) +
  geom_boxplot(aes(x = "AGE_GROUP", y = "CALORIES", fill = "AGE_GROUP")) +
  theme_minimal() +
  labs(title = title,
  y = "Calories (KCal)",
  x = "Age Group",
  fill = "Age Group",
   caption = "\n Population comes from only the surverys classfied as 'Reliable and met the minim
  theme(
    axis_line_y=element_line(color="black", size=0.5),
    axis_line_x=element_line(color="black", size=0.5),
    axis_ticks_x=element_text(),
    panel_grid_major=element_blank(),
    panel_grid_minor=element_blank()
```

```
  )


)

p2
```



comes from only the surverys classfied as 'Reliable and met the minimum criteria'

Weighted Analysis

```
from IPython.display import Markdown
from tabulate import tabulate


def weighted_mean_versus_normal_mean(data, value_col, weight_col, characteristic=None):
    weighted = (data[value_col] * data[weight_col]).sum() / data[weight_col].sum()
    normal = data[value_col].mean()
    thing = value_col.replace("NUT_", "")
    thing = thing.replace("_", " ").title()

    if thing.endswith("Grams"):
        thing = thing.replace("Grams", "(g)")
    elif thing.endswith("Miligrams"):
        thing = thing.replace("Miligrams", "(mg)")

    return {
        "Characteristic": thing,
        "Unweighted Mean": round(normal, 2),
        "Weighted Mean": round(weighted, 2),
    }


nutrient_cols = [
    col for col in master if col.startswith(("NUT_",)) or col == "SAMPLE_WEIGHT_DAY1"
]

nutrient_cols


table = []

for col in nutrient_cols:
    if col != "SAMPLE_WEIGHT_DAY1":  # skip weight column itself
        result = weighted_mean_versus_normal_mean(
            master, col, "SAMPLE_WEIGHT_DAY1", col
        )
        table.append(result)

table
```

```
[{'Characteristic': 'Protein (g)',
  'Unweighted Mean': np.float64(70.66),
  'Weighted Mean': np.float64(72.98)},
 {'Characteristic': 'Carbs (g)',
  'Unweighted Mean': np.float64(223.91),
```

```
 'Weighted Mean': np.float64(227.62)},
{'Characteristic': 'Sugar (g)',
 'Unweighted Mean': np.float64(97.06),
 'Weighted Mean': np.float64(97.13)},
{'Characteristic': 'Fibre (g)',
 'Unweighted Mean': np.float64(15.52),
 'Weighted Mean': np.float64(15.66)},
{'Characteristic': 'Fat (g)',
 'Unweighted Mean': np.float64(79.95),
 'Weighted Mean': np.float64(82.08)},
{'Characteristic': 'Sat Fat (g)',
 'Unweighted Mean': np.float64(26.03),
 'Weighted Mean': np.float64(26.75)},
{'Characteristic': 'Mono Sat Fat (g)',
 'Unweighted Mean': np.float64(26.89),
 'Weighted Mean': np.float64(27.57)},
{'Characteristic': 'Poly Sat Fat (g)',
 'Unweighted Mean': np.float64(18.38),
 'Weighted Mean': np.float64(18.86)},
{'Characteristic': 'Chol (mg)',
 'Unweighted Mean': np.float64(285.87),
 'Weighted Mean': np.float64(296.91)},
{'Characteristic': 'Caffeine (mg)',
 'Unweighted Mean': np.float64(114.79),
 'Weighted Mean': np.float64(116.0)},
{'Characteristic': 'Alcohol (g)',
 'Unweighted Mean': np.float64(5.5),
 'Weighted Mean': np.float64(6.18)}]
```