# This Python 3 environment comes with many helpful analytics libraries installed

# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python

# For example, here's several helpful packages to load

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

# Input data files are available in the read-only "../input/" directory

# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"

# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session

/kaggle/input/starbucks-menu/starbucks\_drinkMenu\_expanded.csv

/kaggle/input/starbucks-menu/starbucks-menu-nutrition-drinks.csv

/kaggle/input/starbucks-menu/starbucks-menu-nutrition-food.csv

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

import plotly.express as px

import plotly.graph\_objects as go

drink = pd.read\_csv("../input/starbucks-menu/starbucks\_drinkMenu\_expanded.csv")

drink.head()

###The original file can be accessed in this link: https://en.starbucksromania.ro/media/nutrition\_tcm71-14482.pdf###

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Trans Fat (g)** | **Saturated Fat (g)** | **Sodium (mg)** | **Total Carbohydrates (g)** | **Cholesterol (mg)** | **Dietary Fibre (g)** | **Sugars (g)** | **Protein (g)** | **Vitamin A (% DV)** | **Vitamin C (% DV)** | **Calcium (% DV)** | **Iron (% DV)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Coffee | Brewed Coffee | Short | 3 | 0.1 | 0.0 | 0.0 | 0 | 5 | 0 | 0 | 0 | 0.3 | 0% | 0% | 0% | 0% | 175 |
| **1** | Coffee | Brewed Coffee | Tall | 4 | 0.1 | 0.0 | 0.0 | 0 | 10 | 0 | 0 | 0 | 0.5 | 0% | 0% | 0% | 0% | 260 |
| **2** | Coffee | Brewed Coffee | Grande | 5 | 0.1 | 0.0 | 0.0 | 0 | 10 | 0 | 0 | 0 | 1.0 | 0% | 0% | 0% | 0% | 330 |
| **3** | Coffee | Brewed Coffee | Venti | 5 | 0.1 | 0.0 | 0.0 | 0 | 10 | 0 | 0 | 0 | 1.0 | 0% | 0% | 2% | 0% | 410 |
| **4** | Classic Espresso Drinks | Caffè Latte | Short Nonfat Milk | 70 | 0.1 | 0.1 | 0.0 | 5 | 75 | 10 | 0 | 9 | 6.0 | 10% | 0% | 20% | 0% | 75 |

This dataset shows every drink menus from Starbucks and their nutrition information.

for col in drink.columns:

print(col)

Beverage\_category

Beverage

Beverage\_prep

Calories

Total Fat (g)

Trans Fat (g)

Saturated Fat (g)

Sodium (mg)

Total Carbohydrates (g)

Cholesterol (mg)

Dietary Fibre (g)

Sugars (g)

Protein (g)

Vitamin A (% DV)

Vitamin C (% DV)

Calcium (% DV)

Iron (% DV)

Caffeine (mg)

We would like to remove the spaces at the left side of each string if there is any

Reference: <https://www.geeksforgeeks.org/remove-spaces-from-column-names-in-pandas/>

drink.columns = drink.columns.str.lstrip()

for col in drink.columns:

print(col)

Beverage\_category

Beverage

Beverage\_prep

Calories

Total Fat (g)

Trans Fat (g)

Saturated Fat (g)

Sodium (mg)

Total Carbohydrates (g)

Cholesterol (mg)

Dietary Fibre (g)

Sugars (g)

Protein (g)

Vitamin A (% DV)

Vitamin C (% DV)

Calcium (% DV)

Iron (% DV)

Caffeine (mg)

drink.columns

Index(['Beverage\_category', 'Beverage', 'Beverage\_prep', 'Calories',

'Total Fat (g)', 'Trans Fat (g) ', 'Saturated Fat (g)', 'Sodium (mg)',

'Total Carbohydrates (g) ', 'Cholesterol (mg)', 'Dietary Fibre (g)',

'Sugars (g)', 'Protein (g) ', 'Vitamin A (% DV) ', 'Vitamin C (% DV)',

'Calcium (% DV) ', 'Iron (% DV) ', 'Caffeine (mg)'],

dtype='object')

For 'Trans Fat (g) ', 'Total Carbohydrates (g) ', 'Protein (g) ', 'Vitamin A (% DV) ', 'Calcium (% DV) ', 'Iron (% DV) ', we would like to get rid of the space at the end

drink.columns = drink.columns.str.rstrip()

drink.columns

Index(['Beverage\_category', 'Beverage', 'Beverage\_prep', 'Calories',

'Total Fat (g)', 'Trans Fat (g)', 'Saturated Fat (g)', 'Sodium (mg)',

'Total Carbohydrates (g)', 'Cholesterol (mg)', 'Dietary Fibre (g)',

'Sugars (g)', 'Protein (g)', 'Vitamin A (% DV)', 'Vitamin C (% DV)',

'Calcium (% DV)', 'Iron (% DV)', 'Caffeine (mg)'],

dtype='object')

Now we would like to check if there are any null values in the dataset

drink.isna().any()

Beverage\_category False

Beverage False

Beverage\_prep False

Calories False

Total Fat (g) False

Trans Fat (g) False

Saturated Fat (g) False

Sodium (mg) False

Total Carbohydrates (g) False

Cholesterol (mg) False

Dietary Fibre (g) False

Sugars (g) False

Protein (g) False

Vitamin A (% DV) False

Vitamin C (% DV) False

Calcium (% DV) False

Iron (% DV) False

Caffeine (mg) True

dtype: bool

Null value(s) exist for the column "Caffeine (mg)"

Reference: <https://stackoverflow.com/questions/29530232/how-to-check-if-any-value-is-nan-in-a-pandas-dataframe>

We would like to count how many null values there are in the dataset

drink.isnull().sum()

Beverage\_category 0

Beverage 0

Beverage\_prep 0

Calories 0

Total Fat (g) 0

Trans Fat (g) 0

Saturated Fat (g) 0

Sodium (mg) 0

Total Carbohydrates (g) 0

Cholesterol (mg) 0

Dietary Fibre (g) 0

Sugars (g) 0

Protein (g) 0

Vitamin A (% DV) 0

Vitamin C (% DV) 0

Calcium (% DV) 0

Iron (% DV) 0

Caffeine (mg) 1

dtype: int64

There is only one null value

Reference: <https://chartio.com/resources/tutorials/how-to-check-if-any-value-is-nan-in-a-pandas-dataframe/>

We would like to find out which row has the null value

drink.isnull().any(axis=1)

drink[drink.isnull().any(axis=1)]

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Trans Fat (g)** | **Saturated Fat (g)** | **Sodium (mg)** | **Total Carbohydrates (g)** | **Cholesterol (mg)** | **Dietary Fibre (g)** | **Sugars (g)** | **Protein (g)** | **Vitamin A (% DV)** | **Vitamin C (% DV)** | **Calcium (% DV)** | **Iron (% DV)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **158** | Shaken Iced Beverages | Iced Brewed Coffee (With Milk & Classic Syrup) | 2% Milk | 90 | 1 | 0.5 | 0.0 | 5 | 25 | 18 | 0 | 18 | 2.0 | 2% | 0% | 6% | 0.00% | NaN |

For 158th row there is a null value for the column 'Caffeine (mg)'

Reference: <https://stackoverflow.com/questions/14247586/how-to-select-rows-with-one-or-more-nulls-from-a-pandas-dataframe-without-listin>

Now let's replace the null value with the correct value

Reference: <https://www.geeksforgeeks.org/python-pandas-dataframe-replace/>

new\_drink = drink.fillna(125)

We could not find the nutrition information that perfectly corresponds to the one in our initial dataset.

Referring to the link below we assumed the caffeine content of the drink is 125 (mg)

<https://www.starbucks.com/menu/product/482/iced?parent=%2Fdrinks%2Fcold-coffees%2Ficed-coffees>

new\_drink.loc[[158]]

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Trans Fat (g)** | **Saturated Fat (g)** | **Sodium (mg)** | **Total Carbohydrates (g)** | **Cholesterol (mg)** | **Dietary Fibre (g)** | **Sugars (g)** | **Protein (g)** | **Vitamin A (% DV)** | **Vitamin C (% DV)** | **Calcium (% DV)** | **Iron (% DV)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **158** | Shaken Iced Beverages | Iced Brewed Coffee (With Milk & Classic Syrup) | 2% Milk | 90 | 1 | 0.5 | 0.0 | 5 | 25 | 18 | 0 | 18 | 2.0 | 2% | 0% | 6% | 0.00% | 125 |

Here, we can see that the null value has been replaced successfully

new\_drink.isna().any()

Beverage\_category False

Beverage False

Beverage\_prep False

Calories False

Total Fat (g) False

Trans Fat (g) False

Saturated Fat (g) False

Sodium (mg) False

Total Carbohydrates (g) False

Cholesterol (mg) False

Dietary Fibre (g) False

Sugars (g) False

Protein (g) False

Vitamin A (% DV) False

Vitamin C (% DV) False

Calcium (% DV) False

Iron (% DV) False

Caffeine (mg) False

dtype: bool

It's confirmed that there is no null value anymore

Now we need to check the data type for each column

new\_drink.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 242 entries, 0 to 241

Data columns (total 18 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Beverage\_category 242 non-null object

1 Beverage 242 non-null object

2 Beverage\_prep 242 non-null object

3 Calories 242 non-null int64

4 Total Fat (g) 242 non-null object

5 Trans Fat (g) 242 non-null float64

6 Saturated Fat (g) 242 non-null float64

7 Sodium (mg) 242 non-null int64

8 Total Carbohydrates (g) 242 non-null int64

9 Cholesterol (mg) 242 non-null int64

10 Dietary Fibre (g) 242 non-null int64

11 Sugars (g) 242 non-null int64

12 Protein (g) 242 non-null float64

13 Vitamin A (% DV) 242 non-null object

14 Vitamin C (% DV) 242 non-null object

15 Calcium (% DV) 242 non-null object

16 Iron (% DV) 242 non-null object

17 Caffeine (mg) 242 non-null object

dtypes: float64(3), int64(6), object(9)

memory usage: 34.2+ KB

We want the data type for 'Total Fat (g)', 'Vitamin A (% DV)', 'Vitamin C (% DV)', 'Calcium (% DV)', 'Iron (% DV)' and 'Caffeine (mg)' to be either integer-type or float-type, otherwise data plotting is not feasible.

While we supsect that there is a typo for 'Total Fat (g)', it seems that for the other aforementioned columns the system is interpreting the data-type as object because of the '%' sign.

First let's start with checking the unique values for 'Total Fat (g)'.

print(new\_drink["Total Fat (g)"].unique())

['0.1' '3.5' '2.5' '0.2' '6' '4.5' '0.3' '7' '5' '0.4' '9' '1.5' '4' '2'

'8' '3' '11' '0' '1' '10' '15' '13' '0.5' '3 2']

All of the values are string-type so we would like to change them to float-type.

Before we convert them to float-type, we would like to first change '3 2' to '32'.

Let's check what the correct value should be

According to the Starbucks website the Total Fat should between 11 to 17 (g) depending on the size, so let's assume it's 16

new\_drink.loc[new\_drink['Total Fat (g)'] == '3 2']

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Trans Fat (g)** | **Saturated Fat (g)** | **Sodium (mg)** | **Total Carbohydrates (g)** | **Cholesterol (mg)** | **Dietary Fibre (g)** | **Sugars (g)** | **Protein (g)** | **Vitamin A (% DV)** | **Vitamin C (% DV)** | **Calcium (% DV)** | **Iron (% DV)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **237** | Frappuccino® Blended Crème | Strawberries & Crème (Without Whipped Cream) | Soymilk | 320 | 3 2 | 0.4 | 0.0 | 0 | 250 | 67 | 1 | 64 | 5.0 | 6% | 8% | 20% | 10% | 0 |

new\_drink["Total Fat (g)"] = new\_drink["Total Fat (g)"].str.replace('3 2','16')

print(new\_drink["Total Fat (g)"].unique())

['0.1' '3.5' '2.5' '0.2' '6' '4.5' '0.3' '7' '5' '0.4' '9' '1.5' '4' '2'

'8' '3' '11' '0' '1' '10' '15' '13' '0.5' '16']

new\_drink.iloc[[237]]

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Trans Fat (g)** | **Saturated Fat (g)** | **Sodium (mg)** | **Total Carbohydrates (g)** | **Cholesterol (mg)** | **Dietary Fibre (g)** | **Sugars (g)** | **Protein (g)** | **Vitamin A (% DV)** | **Vitamin C (% DV)** | **Calcium (% DV)** | **Iron (% DV)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **237** | Frappuccino® Blended Crème | Strawberries & Crème (Without Whipped Cream) | Soymilk | 320 | 16 | 0.4 | 0.0 | 0 | 250 | 67 | 1 | 64 | 5.0 | 6% | 8% | 20% | 10% | 0 |

The value '3 2' has been replaced to '16'

Now let's convert the values into float-type

new\_drink['Total Fat (g)'] = new\_drink['Total Fat (g)'].astype(float)

print(new\_drink["Total Fat (g)"].unique())

[ 0.1 3.5 2.5 0.2 6. 4.5 0.3 7. 5. 0.4 9. 1.5 4. 2.

8. 3. 11. 0. 1. 10. 15. 13. 0.5 16. ]

print(new\_drink['Total Fat (g)'].dtypes)

float64

We would like to go over the same process for 'Vitamin A (% DV)'

print(new\_drink["Vitamin A (% DV)"].unique())

['0%' '10%' '6%' '15%' '20%' '30%' '25%' '8%' '4%' '2%' '50%']

First, we would like to get rid of the '%' sign

new\_drink['Vitamin A (% DV)'] = new\_drink['Vitamin A (% DV)'].str.replace('%', '')

print(new\_drink["Vitamin A (% DV)"].unique())

['0' '10' '6' '15' '20' '30' '25' '8' '4' '2' '50']

Convert them to integer-type

new\_drink['Vitamin A (% DV)'] = new\_drink['Vitamin A (% DV)'].astype(int)

print(new\_drink["Vitamin A (% DV)"].unique())

[ 0 10 6 15 20 30 25 8 4 2 50]

print(new\_drink['Vitamin A (% DV)'].dtypes)

int64

We would like to go over the same process for 'Vitamin C (% DV)'

print(new\_drink["Vitamin C (% DV)"].unique())

['0%' '2%' '4%' '6%' '10%' '15%' '20%' '80%' '100%' '8%']

new\_drink['Vitamin C (% DV)'] = new\_drink['Vitamin C (% DV)'].str.replace('%', '')

print(new\_drink["Vitamin C (% DV)"].unique())

['0' '2' '4' '6' '10' '15' '20' '80' '100' '8']

new\_drink['Vitamin C (% DV)'] = new\_drink['Vitamin C (% DV)'].astype(int)

print(new\_drink["Vitamin C (% DV)"].unique())

[ 0 2 4 6 10 15 20 80 100 8]

print(new\_drink['Vitamin C (% DV)'].dtypes)

int64

We would like to go over the same process for 'Calcium (% DV)'

print(new\_drink["Calcium (% DV)"].unique())

['0%' '2%' '20%' '30%' '40%' '50%' '15%' '25%' '35%' '45%' '10%' '60%'

'6%' '8%']

new\_drink['Calcium (% DV)'] = new\_drink['Calcium (% DV)'].str.replace('%', '')

print(new\_drink["Calcium (% DV)"].unique())

['0' '2' '20' '30' '40' '50' '15' '25' '35' '45' '10' '60' '6' '8']

new\_drink['Calcium (% DV)'] = new\_drink['Calcium (% DV)'].astype(int)

print(new\_drink["Calcium (% DV)"].unique())

[ 0 2 20 30 40 50 15 25 35 45 10 60 6 8]

print(new\_drink['Calcium (% DV)'].dtypes)

int64

We would like to go over the same process for 'Iron (% DV)'

print(new\_drink["Iron (% DV)"].unique())

['0%' '8%' '15%' '25%' '10%' '20%' '30%' '40%' '50%' '6%' '2%' '4%'

'0.00%' '6.00%' '8.00%' '10.00%' '15.00%' '35%']

new\_drink['Iron (% DV)'] = new\_drink['Iron (% DV)'].str.replace('%', '')

print(new\_drink["Iron (% DV)"].unique())

['0' '8' '15' '25' '10' '20' '30' '40' '50' '6' '2' '4' '0.00' '6.00'

'8.00' '10.00' '15.00' '35']

new\_drink['Iron (% DV)'] = new\_drink['Iron (% DV)'].astype(float)

print(new\_drink["Iron (% DV)"].unique())

[ 0. 8. 15. 25. 10. 20. 30. 40. 50. 6. 2. 4. 35.]

print(new\_drink['Iron (% DV)'].dtypes)

float64

We would like to go over the same process for 'Caffeine (mg)'

print(new\_drink["Caffeine (mg)"].unique())

['175' '260' '330' '410' '75' '150' '85' '95' '180' '225' '300' '10' '20'

'25' '30' '0' 'Varies' '50' '70' '120' '55' '80' '110' 'varies' '165'

'235' '90' 125 '125' '170' '15' '130' '140' '100' '145' '65' '105']

new\_drink.loc[new\_drink['Caffeine (mg)'] == 'varies']

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Trans Fat (g)** | **Saturated Fat (g)** | **Sodium (mg)** | **Total Carbohydrates (g)** | **Cholesterol (mg)** | **Dietary Fibre (g)** | **Sugars (g)** | **Protein (g)** | **Vitamin A (% DV)** | **Vitamin C (% DV)** | **Calcium (% DV)** | **Iron (% DV)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **130** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | Short Nonfat Milk | 80 | 0.1 | 0.1 | 0.0 | 0 | 45 | 16 | 0 | 16 | 4.0 | 6 | 0 | 10 | 0.0 | varies |
| **131** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | 2% Milk | 90 | 2.0 | 1.0 | 0.1 | 10 | 50 | 15 | 0 | 15 | 3.0 | 6 | 0 | 10 | 0.0 | varies |
| **132** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | Soymilk | 80 | 1.5 | 0.2 | 0.0 | 0 | 40 | 14 | 0 | 13 | 3.0 | 4 | 0 | 10 | 6.0 | varies |
| **133** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | Tall Nonfat Milk | 120 | 0.1 | 0.1 | 0.0 | 5 | 65 | 23 | 0 | 23 | 5.0 | 10 | 0 | 20 | 0.0 | varies |
| **134** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | 2% Milk | 140 | 3.0 | 1.5 | 0.1 | 15 | 75 | 23 | 0 | 23 | 5.0 | 8 | 0 | 15 | 0.0 | varies |
| **135** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | Soymilk | 130 | 2.5 | 0.3 | 0.0 | 0 | 60 | 21 | 1 | 19 | 4.0 | 6 | 0 | 20 | 8.0 | varies |
| **136** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | Grande Nonfat Milk | 150 | 0.2 | 0.1 | 0.0 | 5 | 85 | 31 | 0 | 31 | 7.0 | 15 | 0 | 25 | 0.0 | varies |
| **137** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | 2% Milk | 190 | 4.0 | 2.0 | 0.1 | 15 | 95 | 31 | 0 | 30 | 7.0 | 10 | 0 | 25 | 0.0 | varies |
| **138** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | Soymilk | 170 | 3.5 | 0.4 | 0.0 | 0 | 80 | 27 | 1 | 25 | 6.0 | 8 | 0 | 25 | 10.0 | varies |
| **139** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | Venti Nonfat Milk | 190 | 0.2 | 0.1 | 0.0 | 5 | 110 | 39 | 0 | 39 | 9.0 | 15 | 0 | 30 | 0.0 | varies |
| **140** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | 2% Milk | 230 | 5.0 | 2.5 | 0.2 | 20 | 125 | 38 | 0 | 38 | 9.0 | 15 | 0 | 30 | 0.0 | varies |
| **141** | Tazo® Tea Drinks | Tazo® Full-Leaf Tea Latte | Soymilk | 210 | 4.0 | 0.5 | 0.0 | 0 | 100 | 34 | 1 | 32 | 7.0 | 10 | 0 | 30 | 15.0 | varies |

Since 'Tazo® Full-Leaf Tea Latte' is the only beverage with 'varies' for 'Caffeine (mg)', let's assume it's 50

new\_drink['Caffeine (mg)'] = new\_drink['Caffeine (mg)'].replace('varies', '50')

print(new\_drink["Caffeine (mg)"].unique())

['175' '260' '330' '410' '75' '150' '85' '95' '180' '225' '300' '10' '20'

'25' '30' '0' 'Varies' '50' '70' '120' '55' '80' '110' '165' '235' '90'

125 '125' '170' '15' '130' '140' '100' '145' '65' '105']

new\_drink.loc[new\_drink['Caffeine (mg)'] == 'Varies']

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Trans Fat (g)** | **Saturated Fat (g)** | **Sodium (mg)** | **Total Carbohydrates (g)** | **Cholesterol (mg)** | **Dietary Fibre (g)** | **Sugars (g)** | **Protein (g)** | **Vitamin A (% DV)** | **Vitamin C (% DV)** | **Calcium (% DV)** | **Iron (% DV)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **102** | Tazo® Tea Drinks | Tazo® Tea | Short | 0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 0.0 | Varies |
| **103** | Tazo® Tea Drinks | Tazo® Tea | Tall | 0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 0.0 | Varies |
| **104** | Tazo® Tea Drinks | Tazo® Tea | Grande | 0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 0.0 | Varies |
| **105** | Tazo® Tea Drinks | Tazo® Tea | Venti | 0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 0.0 | Varies |
| **167** | Shaken Iced Beverages | Shaken Iced Tazo® Tea (With Classic Syrup) | Grande | 80 | 0.0 | 0.0 | 0.0 | 0 | 0 | 21 | 0 | 21 | 0.0 | 0 | 0 | 0 | 0.0 | Varies |
| **168** | Shaken Iced Beverages | Shaken Iced Tazo® Tea (With Classic Syrup) | Venti | 120 | 0.0 | 0.0 | 0.0 | 0 | 0 | 31 | 0 | 31 | 0.0 | 0 | 0 | 0 | 0.0 | Varies |
| **169** | Shaken Iced Beverages | Shaken Iced Tazo® Tea Lemonade (With Classic S... | Tall | 100 | 0.0 | 0.0 | 0.0 | 0 | 0 | 25 | 0 | 24 | 0.1 | 0 | 10 | 0 | 0.0 | Varies |
| **170** | Shaken Iced Beverages | Shaken Iced Tazo® Tea Lemonade (With Classic S... | Grande | 130 | 0.0 | 0.0 | 0.0 | 0 | 0 | 33 | 0 | 33 | 0.1 | 0 | 15 | 0 | 0.0 | Varies |
| **171** | Shaken Iced Beverages | Shaken Iced Tazo® Tea Lemonade (With Classic S... | Venti | 190 | 0.0 | 0.0 | 0.0 | 0 | 0 | 49 | 0 | 49 | 0.1 | 0 | 20 | 0 | 0.0 | Varies |
| **172** | Smoothies | Banana Chocolate Smoothie | Grande Nonfat Milk | 280 | 2.5 | 1.5 | 0.0 | 5 | 150 | 53 | 7 | 34 | 20.0 | 10 | 15 | 20 | 0.0 | Varies |

We will assume the caffeine content for the same beverage category is constant.

(Tazo® Tea Drinks = 10, Shaken Iced Beverages = 20, Smoothies = 30)

**Do more research and fix the values**

new\_drink.iloc[102:106] = new\_drink.iloc[102:106].replace('Varies', '10')

new\_drink.iloc[102:106]

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Trans Fat (g)** | **Saturated Fat (g)** | **Sodium (mg)** | **Total Carbohydrates (g)** | **Cholesterol (mg)** | **Dietary Fibre (g)** | **Sugars (g)** | **Protein (g)** | **Vitamin A (% DV)** | **Vitamin C (% DV)** | **Calcium (% DV)** | **Iron (% DV)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **102** | Tazo® Tea Drinks | Tazo® Tea | Short | 0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 0.0 | 10 |
| **103** | Tazo® Tea Drinks | Tazo® Tea | Tall | 0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 0.0 | 10 |
| **104** | Tazo® Tea Drinks | Tazo® Tea | Grande | 0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 0.0 | 10 |
| **105** | Tazo® Tea Drinks | Tazo® Tea | Venti | 0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 0.0 | 10 |

new\_drink.iloc[167:172] = new\_drink.iloc[167:172].replace('Varies', '20')

new\_drink.iloc[167:172]

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Trans Fat (g)** | **Saturated Fat (g)** | **Sodium (mg)** | **Total Carbohydrates (g)** | **Cholesterol (mg)** | **Dietary Fibre (g)** | **Sugars (g)** | **Protein (g)** | **Vitamin A (% DV)** | **Vitamin C (% DV)** | **Calcium (% DV)** | **Iron (% DV)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **167** | Shaken Iced Beverages | Shaken Iced Tazo® Tea (With Classic Syrup) | Grande | 80 | 0.0 | 0.0 | 0.0 | 0 | 0 | 21 | 0 | 21 | 0.0 | 0 | 0 | 0 | 0.0 | 20 |
| **168** | Shaken Iced Beverages | Shaken Iced Tazo® Tea (With Classic Syrup) | Venti | 120 | 0.0 | 0.0 | 0.0 | 0 | 0 | 31 | 0 | 31 | 0.0 | 0 | 0 | 0 | 0.0 | 20 |
| **169** | Shaken Iced Beverages | Shaken Iced Tazo® Tea Lemonade (With Classic S... | Tall | 100 | 0.0 | 0.0 | 0.0 | 0 | 0 | 25 | 0 | 24 | 0.1 | 0 | 10 | 0 | 0.0 | 20 |
| **170** | Shaken Iced Beverages | Shaken Iced Tazo® Tea Lemonade (With Classic S... | Grande | 130 | 0.0 | 0.0 | 0.0 | 0 | 0 | 33 | 0 | 33 | 0.1 | 0 | 15 | 0 | 0.0 | 20 |
| **171** | Shaken Iced Beverages | Shaken Iced Tazo® Tea Lemonade (With Classic S... | Venti | 190 | 0.0 | 0.0 | 0.0 | 0 | 0 | 49 | 0 | 49 | 0.1 | 0 | 20 | 0 | 0.0 | 20 |

new\_drink.iloc[172] = new\_drink.iloc[172].replace('Varies', '30')

new\_drink.iloc[[172]]

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Trans Fat (g)** | **Saturated Fat (g)** | **Sodium (mg)** | **Total Carbohydrates (g)** | **Cholesterol (mg)** | **Dietary Fibre (g)** | **Sugars (g)** | **Protein (g)** | **Vitamin A (% DV)** | **Vitamin C (% DV)** | **Calcium (% DV)** | **Iron (% DV)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **172** | Smoothies | Banana Chocolate Smoothie | Grande Nonfat Milk | 280 | 2.5 | 1.5 | 0.0 | 5 | 150 | 53 | 7 | 34 | 20.0 | 10 | 15 | 20 | 0.0 | 30 |

print(new\_drink["Caffeine (mg)"].unique())

['175' '260' '330' '410' '75' '150' '85' '95' '180' '225' '300' '10' '20'

'25' '30' '0' '50' '70' '120' '55' '80' '110' '165' '235' '90' 125 '125'

'170' '15' '130' '140' '100' '145' '65' '105']

new\_drink['Caffeine (mg)'] = new\_drink['Caffeine (mg)'].astype(int)

print(new\_drink["Caffeine (mg)"].unique())

[175 260 330 410 75 150 85 95 180 225 300 10 20 25 30 0 50 70

120 55 80 110 165 235 90 125 170 15 130 140 100 145 65 105]

print(new\_drink['Caffeine (mg)'].dtypes)

int64

new\_drink.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 242 entries, 0 to 241

Data columns (total 18 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Beverage\_category 242 non-null object

1 Beverage 242 non-null object

2 Beverage\_prep 242 non-null object

3 Calories 242 non-null int64

4 Total Fat (g) 242 non-null float64

5 Trans Fat (g) 242 non-null float64

6 Saturated Fat (g) 242 non-null float64

7 Sodium (mg) 242 non-null int64

8 Total Carbohydrates (g) 242 non-null int64

9 Cholesterol (mg) 242 non-null int64

10 Dietary Fibre (g) 242 non-null int64

11 Sugars (g) 242 non-null int64

12 Protein (g) 242 non-null float64

13 Vitamin A (% DV) 242 non-null int64

14 Vitamin C (% DV) 242 non-null int64

15 Calcium (% DV) 242 non-null int64

16 Iron (% DV) 242 non-null float64

17 Caffeine (mg) 242 non-null int64

dtypes: float64(5), int64(10), object(3)

memory usage: 34.2+ KB

Finally, the dataset is now ready for data plotting.

Data plotting will be carried out for Beverages listed below

new\_drink.Beverage\_category.unique()

array(['Coffee', 'Classic Espresso Drinks', 'Signature Espresso Drinks',

'Tazo® Tea Drinks', 'Shaken Iced Beverages', 'Smoothies',

'Frappuccino® Blended Coffee', 'Frappuccino® Light Blended Coffee',

'Frappuccino® Blended Crème'], dtype=object)

plt.style.use('default')

plt.figure(figsize=(10,6),edgecolor='1',dpi=100)

a=sns.countplot(x='Beverage\_category',color='darkgreen',data=new\_drink)

xticks=plt.xticks(rotation=65,family='serif')

yticks=plt.yticks(family='serif')

plt.xlabel(new\_drink['Beverage\_category'].all(),font='serif')

plt.ylabel('figure',font='serif')

a.spines['bottom'].set\_color('gray')

a.spines['left'].set\_color('gray')

sns.despine()

Above we display the number of items for each 'Type' using countplot. We are able to observe that 'Classic Espresso Drinks' has most number of items, following 'Tazo Tea Drinks' and 'Signature Espresso Drinks'.

new\_drink.Beverage.unique()

array(['Brewed Coffee', 'Caffè Latte',

'Caffè Mocha (Without Whipped Cream)',

'Vanilla Latte (Or Other Flavoured Latte)', 'Caffè Americano',

'Cappuccino', 'Espresso', 'Skinny Latte (Any Flavour)',

'Caramel Macchiato',

'White Chocolate Mocha (Without Whipped Cream)',

'Hot Chocolate (Without Whipped Cream)',

'Caramel Apple Spice (Without Whipped Cream)', 'Tazo® Tea',

'Tazo® Chai Tea Latte', 'Tazo® Green Tea Latte',

'Tazo® Full-Leaf Tea Latte',

'Tazo® Full-Leaf Red Tea Latte (Vanilla Rooibos)',

'Iced Brewed Coffee (With Classic Syrup)',

'Iced Brewed Coffee (With Milk & Classic Syrup)',

'Shaken Iced Tazo® Tea (With Classic Syrup)',

'Shaken Iced Tazo® Tea Lemonade (With Classic Syrup)',

'Banana Chocolate Smoothie', 'Orange Mango Banana Smoothie',

'Strawberry Banana Smoothie', 'Coffee',

'Mocha (Without Whipped Cream)', 'Caramel (Without Whipped Cream)',

'Java Chip (Without Whipped Cream)', 'Mocha', 'Caramel',

'Java Chip', 'Strawberries & Crème (Without Whipped Cream)',

'Vanilla Bean (Without Whipped Cream)'], dtype=object)

Analyze the Number of Items per Type The heatmap function is another cool way to show the distribution of calories of Beverage\_categories for each type.

Here, if you place the cursor on the bar it shows the name of categories.

Below we can observe that 'Signature Espresso Drinks' and 'Frappuccino Blended Coffee' have generally high calories of coffees among a number of coffees.

'white chocolate mocha venti' has the highest calories. 

px.density\_heatmap(x='Beverage\_category',y='Calories',data\_frame=new\_drink,width=900)

Below we analyze how are Calories and sugars related in linear graph. It is seen that they are strongly positive related.

plt.style.use('ggplot')

plt.figure(figsize=(8,5),dpi=80)

sns.scatterplot(x='Calories',y='Sugars (g)',data=new\_drink,alpha=0.8,s=60)

plt.title('Calories vs sugars',x=0.5,y=1.05)

Text(0.5, 1.05, 'Calories vs sugars')

Now, we are going to devide this berverage into 2 groups. One is 'Hot Beverages' another is 'Cold Beverages'.

We will be observed the average of nutrients of each berverage in each group.

Before that, we are going to get rid of some nutrients to describe the only tidy data.

new\_nutrients= new\_drink.drop(['Trans Fat (g)','Saturated Fat (g)','Sodium (mg)','Total Carbohydrates (g)',

'Cholesterol (mg)','Dietary Fibre (g)','Vitamin A (% DV)','Vitamin C (% DV)','Calcium (% DV)','Iron (% DV)'],axis=1)

new\_nutrients.head()

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Sugars (g)** | **Protein (g)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Coffee | Brewed Coffee | Short | 3 | 0.1 | 0 | 0.3 | 175 |
| **1** | Coffee | Brewed Coffee | Tall | 4 | 0.1 | 0 | 0.5 | 260 |
| **2** | Coffee | Brewed Coffee | Grande | 5 | 0.1 | 0 | 1.0 | 330 |
| **3** | Coffee | Brewed Coffee | Venti | 5 | 0.1 | 0 | 1.0 | 410 |
| **4** | Classic Espresso Drinks | Caffè Latte | Short Nonfat Milk | 70 | 0.1 | 9 | 6.0 | 75 |

Hot\_drink = new\_nutrients.drop(new\_nutrients.index[154:]) # index starts from 0, so 154 is actually number 155 so 155 ~ at the end.

Hot\_drink.head()

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Sugars (g)** | **Protein (g)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Coffee | Brewed Coffee | Short | 3 | 0.1 | 0 | 0.3 | 175 |
| **1** | Coffee | Brewed Coffee | Tall | 4 | 0.1 | 0 | 0.5 | 260 |
| **2** | Coffee | Brewed Coffee | Grande | 5 | 0.1 | 0 | 1.0 | 330 |
| **3** | Coffee | Brewed Coffee | Venti | 5 | 0.1 | 0 | 1.0 | 410 |
| **4** | Classic Espresso Drinks | Caffè Latte | Short Nonfat Milk | 70 | 0.1 | 9 | 6.0 | 75 |

Hot\_drink.iloc[-1] #check if it is the last Hot\_drink

Beverage\_category Tazo® Tea Drinks

Beverage Tazo® Full-Leaf Red Tea Latte (Vanilla Rooibos)

Beverage\_prep Soymilk

Calories 210

Total Fat (g) 4.0

Sugars (g) 32

Protein (g) 7.0

Caffeine (mg) 0

Name: 153, dtype: object

Cold\_drink = new\_nutrients.drop(new\_nutrients.index[0:156]) # 0~155

Cold\_drink.head()

|  | **Beverage\_category** | **Beverage** | **Beverage\_prep** | **Calories** | **Total Fat (g)** | **Sugars (g)** | **Protein (g)** | **Caffeine (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **156** | Shaken Iced Beverages | Iced Brewed Coffee (With Classic Syrup) | Venti | 130 | 0.1 | 31 | 0.4 | 235 |
| **157** | Shaken Iced Beverages | Iced Brewed Coffee (With Milk & Classic Syrup) | Tall Nonfat Milk | 80 | 0.1 | 18 | 2.0 | 90 |
| **158** | Shaken Iced Beverages | Iced Brewed Coffee (With Milk & Classic Syrup) | 2% Milk | 90 | 1.0 | 18 | 2.0 | 125 |
| **159** | Shaken Iced Beverages | Iced Brewed Coffee (With Milk & Classic Syrup) | Soymilk | 80 | 1.0 | 17 | 2.0 | 90 |
| **160** | Shaken Iced Beverages | Iced Brewed Coffee (With Milk & Classic Syrup) | Grande Nonfat Milk | 110 | 0.1 | 24 | 2.0 | 90 |

Analysis of Nutrients

Average Calories distribution for whole Type

Below we observe the hightest amount of calories for 'Smoothies', followed by 'Frappuccino blended coffee' and 'Signature Espresso Drinks'

like this, we are going to analyze 5 different tpyes of nutrients called 'Calories', 'Total Fat (g)', 'Sugars (g)', 'Protein (g)', 'Caffeine (mg)' for Hot\_drink and Cold\_dirnk respectively.

calories=pd.DataFrame(new\_drink.groupby('Beverage\_category')['Calories'].mean())

colors=['gray']\*9

colors[7]='#eb7a34'

colors[6]='blue'

colors[2]='lightpink'

fig = go.Figure(data=[go.Bar(

x=calories.index,

y=calories['Calories'],

marker\_color=colors

)])

fig.update\_layout(width=700,height=500)

fig.update\_xaxes(title='Type')

fig.update\_yaxes(title='Avg Calories')

fig.show()

We are going to start from Hot\_drink.

It is observed that generally White Chocolate Mocha has the highest nutrients value among Hot\_drink

calories=pd.DataFrame(Hot\_drink.groupby('Beverage')['Calories'].mean())

colors=['gray']\*17

colors[16]='#eb7a34'

fig = go.Figure(data=[go.Bar(

x=calories.index,

y=calories['Calories'],

marker\_color=colors

)])

fig.update\_layout(width=700,height=500)

fig.update\_xaxes(title='Hot\_drink')

fig.update\_yaxes(title='Avg Calories')

fig.show()

Total\_fat=pd.DataFrame(Hot\_drink.groupby('Beverage')['Total Fat (g)'].mean())

colors=['gray']\*17

colors[16]='#eb7a34'

fig = go.Figure(data=[go.Bar(

x=Total\_fat.index,

y=Total\_fat['Total Fat (g)'],

marker\_color=colors

)])

fig.update\_layout(width=700,height=500)

fig.update\_xaxes(title='Hot\_drink')

fig.update\_yaxes(title='Avg Total\_fat')

fig.show()

Sugars=pd.DataFrame(Hot\_drink.groupby('Beverage')['Sugars (g)'].mean())

colors=['gray']\*17

colors[5]='#eb7a34'

fig = go.Figure(data=[go.Bar(

x=Sugars.index,

y=Sugars['Sugars (g)'],

marker\_color=colors

)])

fig.update\_layout(width=700,height=500)

fig.update\_xaxes(title='Hot\_drink')

fig.update\_yaxes(title='Avg Sugars')

fig.show()

Protein=pd.DataFrame(Hot\_drink.groupby('Beverage')['Protein (g)'].mean())

colors=['gray']\*17

colors[16]='#eb7a34'

fig = go.Figure(data=[go.Bar(

x=Protein.index,

y=Protein['Protein (g)'],

marker\_color=colors

)])

fig.update\_layout(width=700,height=500)

fig.update\_xaxes(title='Hot\_drink')

fig.update\_yaxes(title='Avg Protein')

fig.show()

Caffeine=pd.DataFrame(Hot\_drink.groupby('Beverage')['Caffeine (mg)'].mean())

colors=['gray']\*17

colors[0]='#eb7a34'

fig = go.Figure(data=[go.Bar(

x=Caffeine.index,

y=Caffeine['Caffeine (mg)'],

marker\_color=colors

)])

fig.update\_layout(width=700,height=500)

fig.update\_xaxes(title='Hot\_drink')

fig.update\_yaxes(title='Avg Caffeine')

fig.show()

Now, we are going to see Cold\_drink.

It is observed that generally Java Chip(Without Whipped Cream) has the highest nutrients value among Cold\_drink

calories=pd.DataFrame(Cold\_drink.groupby('Beverage')['Calories'].mean())

colors=['gray']\*17

colors[7]='#eb7a34'

fig = go.Figure(data=[go.Bar(

x=calories.index,

y=calories['Calories'],

marker\_color=colors

)])

fig.update\_layout(width=700,height=500)

fig.update\_xaxes(title='Cold\_drink')

fig.update\_yaxes(title='Avg Calories')

fig.show()

Total\_fat=pd.DataFrame(Cold\_drink.groupby('Beverage')['Total Fat (g)'].mean())

colors=['gray']\*17

colors[7]='#eb7a34'

fig = go.Figure(data=[go.Bar(

x=Total\_fat.index,

y=Total\_fat['Total Fat (g)'],

marker\_color=colors

)])

fig.update\_layout(width=700,height=500)

fig.update\_xaxes(title='Cold\_drink')

fig.update\_yaxes(title='Avg Total\_fat')

fig.show()

Sugars=pd.DataFrame(Cold\_drink.groupby('Beverage')['Sugars (g)'].mean())

colors=['gray']\*17

colors[7]='#eb7a34'

fig = go.Figure(data=[go.Bar(

x=Sugars.index,

y=Sugars['Sugars (g)'],

marker\_color=colors

)])

fig.update\_layout(width=700,height=500)

fig.update\_xaxes(title='Cold\_drink')

fig.update\_yaxes(title='Avg Sugars')

fig.show()

Protein=pd.DataFrame(Cold\_drink.groupby('Beverage')['Protein (g)'].mean())

colors=['gray']\*17

colors[0]='#eb7a34'

fig = go.Figure(data=[go.Bar(

x=Protein.index,

y=Protein['Protein (g)'],

marker\_color=colors

)])

fig.update\_layout(width=700,height=500)

fig.update\_xaxes(title='Cold\_drink')

fig.update\_yaxes(title='Avg Protein')

fig.show()

Caffeine=pd.DataFrame(Cold\_drink.groupby('Beverage')['Caffeine (mg)'].mean())

colors=['gray']\*17

colors[4]='#eb7a34'

fig = go.Figure(data=[go.Bar(

x=Caffeine.index,

y=Caffeine['Caffeine (mg)'],

marker\_color=colors

)])

fig.update\_layout(width=700,height=500)

fig.update\_xaxes(title='Cold\_drink')

fig.update\_yaxes(title='Avg Caffeine')

fig.show()

By analyzing those data, we know that the highest 'Calories' and 'Total Fat (g)' were taken by the same beverage,

However, for 'Sugars (g)', 'Protein (g)', and 'Caffeine (mg)', it could be taken by different berverages.