1 GBA 465 Assignment 03 - Breakfast Cereals (Starter)

1.1 Part 1 - Importing the Data

In Part 1, you will import data (from the provided source files) using Pandas.

1.1.1 Import 1.1

Action: Read the data from each of the three (3) data files using the `read_csv` function, creating three (3) distinct DataFrames.

In [1]:
```
# Your implementation:
import pandas as pd
nutrition = pd.read_csv("breakfast-cereals-nutrition.csv")
products = pd.read_csv("breakfast-cereals-products.csv")
other = pd.read_csv("breakfast-cereals-other.csv")
nutrition
```

Out[1]:

	NAM	CAL	CAR	FAT	FIB	POT	PRO	SOD	SUG	VIT
0	100% Natural Bran	120.0	8.0	5.0	2.0	135.0	3.0	15.0	8.0	0.0
1	Cinnamon Toast Crunch	120.0	13.0	3.0	0.0	45.0	1.0	210.0	9.0	25.0
2	Muesli Raisins; Peaches; & Pecans	150.0	16.0	3.0	3.0	170.0	4.0	150.0	11.0	25.0
3	Cracklin' Oat Bran	110.0	10.0	3.0	4.0	160.0	3.0	140.0	7.0	25.0
4	Muesli Raisins; Dates; & Almonds	150.0	16.0	3.0	3.0	170.0	4.0	95.0	11.0	25.0
...	...	...	...	...	...	...	...	...	...	...
73	Raisin Squares	90.0	15.0	0.0	2.0	110.0	2.0	0.0	6.0	25.0
74	Shredded Wheat	80.0	16.0	0.0	3.0	95.0	2.0	0.0	0.0	0.0
75	Puffed Wheat	50.0	10.0	0.0	1.0	50.0	2.0	0.0	0.0	0.0
76	Puffed Rice	50.0	13.0	0.0	0.0	15.0	1.0	0.0	0.0	0.0
77	Cocoa Pebbles	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

78 rows × 10 columns

1.1.2 Import 1.2

Action: Combine all three (3) DataFrames into a single DataFrame. This DataFrame will be referred to as the "Master DataFrame".

In [2]:

```
# Your implementation:
# pd.concat([nutrition,products,other],join="inner", names="NAM", axis = 1)
merged1 = pd.merge(nutrition,products, on="NAM")
MasterDataFrame = pd.merge(merged1,other, on="NAM")
MasterDataFrame
```

Out[2]:

	NAM	CAL	CAR	FAT	FIB	POT	PRO	SOD	SUG	VIT	TYP	COM	CUP	SHE	W
0	100% Natural Bran	120.0	8.0	5.0	2.0	135.0	3.0	15.0	8.0	0.0	C	Q	1.00	3.0	1.
1	Cinnamon Toast Crunch	120.0	13.0	3.0	0.0	45.0	1.0	210.0	9.0	25.0	C	G	0.75	2.0	1.
2	Muesli Raisins; Peaches; & Pecans	150.0	16.0	3.0	3.0	170.0	4.0	150.0	11.0	25.0	C	R	1.00	3.0	1.
3	Cracklin' Oat Bran	110.0	10.0	3.0	4.0	160.0	3.0	140.0	7.0	25.0	C	K	0.50	3.0	1.
4	Muesli Raisins; Dates; & Almonds	150.0	16.0	3.0	3.0	170.0	4.0	95.0	11.0	25.0	C	R	1.00	3.0	1.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
73	Raisin Squares	90.0	15.0	0.0	2.0	110.0	2.0	0.0	6.0	25.0	C	K	0.50	3.0	1.
74	Shredded Wheat	80.0	16.0	0.0	3.0	95.0	2.0	0.0	0.0	0.0	C	N	1.00	1.0	0.
75	Puffed Wheat	50.0	10.0	0.0	1.0	50.0	2.0	0.0	0.0	0.0	C	Q	1.00	3.0	0.
76	Puffed Rice	50.0	13.0	0.0	0.0	15.0	1.0	0.0	0.0	0.0	C	Q	1.00	3.0	0.
77	Cocoa Pebbles	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	C	G	NaN	NaN	Na

78 rows × 16 columns

1.2 Part 2 - Exploring the Data

In Part 2, you will explore the structure and data of the Master DataFrame using Pandas.

1.2.1 Exploration 2.1

Action: Preview the first five (5) rows of data using `head` method.

In [3]:
```
# Your implementation:
print(MasterDataFrame.head(5))
```

```
                                 NAM    CAL   CAR  FAT  FIB    POT  PRO  \
0                  100% Natural Bran  120.0   8.0  5.0  2.0  135.0  3.0   
1              Cinnamon Toast Crunch  120.0  13.0  3.0  0.0   45.0  1.0   
2  Muesli Raisins; Peaches; & Pecans  150.0  16.0  3.0  3.0  170.0  4.0   
3                 Cracklin' Oat Bran  110.0  10.0  3.0  4.0  160.0  3.0   
4   Muesli Raisins; Dates; & Almonds  150.0  16.0  3.0  3.0  170.0  4.0   

     SOD   SUG   VIT TYP COM   CUP  SHE  WEI        RAT  
0   15.0   8.0   0.0   C   Q  1.00  3.0  1.0  33.983679  
1  210.0   9.0  25.0   C   G  0.75  2.0  1.0  19.823573  
2  150.0  11.0  25.0   C   R  1.00  3.0  1.0  34.139765  
3  140.0   7.0  25.0   C   K  0.50  3.0  1.0  40.448772  
4   95.0  11.0  25.0   C   R  1.00  3.0  1.0  37.136863  
```

1.2.2 Exploration 2.2

Action: Inspect the structure data using the `info` function.

In [4]:
```
# Your implementation:
print(MasterDataFrame.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 78 entries, 0 to 77
Data columns (total 16 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   NAM     78 non-null     object 
 1   CAL     77 non-null     float64
 2   CAR     77 non-null     float64
 3   FAT     77 non-null     float64
 4   FIB     77 non-null     float64
 5   POT     77 non-null     float64
 6   PRO     77 non-null     float64
 7   SOD     77 non-null     float64
 8   SUG     77 non-null     float64
 9   VIT     77 non-null     float64
 10  TYP     78 non-null     object 
 11  COM     78 non-null     object 
 12  CUP     77 non-null     float64
 13  SHE     77 non-null     float64
 14  WEI     77 non-null     float64
 15  RAT     77 non-null     float64
dtypes: float64(13), object(3)
memory usage: 10.4+ KB
None
```

1.2.3 Exploration 2.3

Action: Display the shape of the data using the `shape` attribute.

In [5]:
```
# Your implementation:

print(MasterDataFrame.shape)
```

```
(78, 16)
```

1.2.4 Exploration 2.4

Action: Display the statistical column breakdown using the `describe` function.

In [6]:

```
# Your implementation:
print(MasterDataFrame.describe())
```

```
              CAL        CAR        FAT        FIB         POT        PRO  \
count   77.000000  77.000000  77.000000  77.000000   77.000000  77.000000
mean   106.883117  14.597403   1.012987   2.151948   96.077922   2.545455
std     19.484119   4.278956   1.006473   2.383364   71.286813   1.094790
min     50.000000  -1.000000   0.000000   0.000000   -1.000000   1.000000
25%    100.000000  12.000000   0.000000   1.000000   40.000000   2.000000
50%    110.000000  14.000000   1.000000   2.000000   90.000000   3.000000
75%    110.000000  17.000000   2.000000   3.000000  120.000000   3.000000
max    160.000000  23.000000   5.000000  14.000000  330.000000   6.000000

              SOD        SUG         VIT        CUP        SHE        WEI  \
count   77.000000  77.000000   77.000000  77.000000  77.000000  77.000000
mean   159.675325   6.922078   28.246753   0.821039   2.207792   1.029610
std     83.832295   4.444885   22.342523   0.232716   0.832524   0.150477
min      0.000000  -1.000000    0.000000   0.250000   1.000000   0.500000
25%    130.000000   3.000000   25.000000   0.670000   1.000000   1.000000
50%    180.000000   7.000000   25.000000   0.750000   2.000000   1.000000
75%    210.000000  11.000000   25.000000   1.000000   3.000000   1.000000
max    320.000000  15.000000  100.000000   1.500000   3.000000   1.500000

             RAT
count  77.000000
mean   42.665705
std    14.047289
min    18.042851
25%    33.174094
50%    40.400208
75%    50.828392
max    93.704912
```

1.2.5 Exploration 2.5

Action: Display the columns using the `columns` attribute.

In [7]:

```
# Your implementation:

print(MasterDataFrame.columns)
```

```
Index(['NAM', 'CAL', 'CAR', 'FAT', 'FIB', 'POT', 'PRO', 'SOD', 'SUG', 'VIT',
       'TYP', 'COM', 'CUP', 'SHE', 'WEI', 'RAT'],
      dtype='object')
```

1.2.6 Exploration 2.6

Action: Display the index using the `index` attribute.

In [8]:

```
# Your implementation:

print(MasterDataFrame.index)
```

```
Int64Index([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
            17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
            34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
            51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
            68, 69, 70, 71, 72, 73, 74, 75, 76, 77],
           dtype='int64')
```

1.2.7 Exploration 2.9

Action: Display any rows or columns containing null values using the `isnull` method.

In [9]:
```
# Your implementation:
MasterDataFrame[MasterDataFrame.isnull().any(axis=1)]
```

Out[9]:

	NAM	CAL	CAR	FAT	FIB	POT	PRO	SOD	SUG	VIT	TYP	COM	CUP	SHE	WEI	I
77	Cocoa Pebbles	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	C	G	NaN	NaN	NaN	I

1.3 Part 3 - Manipulating the Data

In Part 3, you will manipulate the structure and data of the Master DataFrame using Pandas.

1.3.1 Manipulation 3.1

Action: Rename the columns in the Master DataFrame so they are more descriptive, using the following mapping (column names shown alphabetically):

- `CAL` to `CALORIES`
- `CAR` to `CARBS`
- `COM` to `COMPANY`
- `CUP` to `CUPS`
- `FAT` (this column does not need to be renamed)
- `FIB` to `FIBER`
- `NAM` to `NAME`
- `TYP` to `TYPE`
- `POT` to `POTASSIUM`
- `PRO` to `PROTEIN`
- `RAT` to `RATING`
- `SHE` to `SHELF`
- `SOD` to `SODIUM`
- `SUG` to `SUGAR`
- `VIT` to `VITAMINS`
- `WEI` to `WEIGHT`

In [10]:
```
# Your implementation:

MasterDataFrame = MasterDataFrame.rename(columns = {
    "CAL": "CALORIES", "CAR": "CARBS", "COM": "COMPANY", "CUP": "CUPS", "FIB": "F
    "TYP": "TYPE", "POT": "POTASSIUM", "PRO": "PROTEIN", "RAT": "RATING", "SHE":
    "SUG": "SUGAR", "VIT": "VITAMINS", "WEI": "WEIGHT",
})
```

1.3.2 Manipulation 3.2

Action: Reorder the columns in the Master DataFrame so that they appear in the following order:

1. `NAME`
2. `COMPANY`
3. `TYPE`
4. `RATING`
5. `SHELF`
6. `CALORIES`
7. `PROTEIN`
8. `SODIUM`
9. `FIBER`
10. `CARBS`
11. `VITAMINS`
12. `POTASSIUM`

13. FAT
14. SUGAR
15. WEIGHT
16. CUPS

In [11]:

```
# Your implementation:

#MasterDataFrame = MasterDataFrame.reindex(columns=["NAME","COMPANY","TYPE","RATI
#MasterDataFrame = MasterDataFrame[["NAME","COMPANY","TYPE","RATING","SHELF","CAL
#df = df.reindex(columns=column_names)
MasterDataFrame = MasterDataFrame[["NAME","COMPANY", "TYPE", "RATING", "SHELF", '
                                    "SODIUM", "FIBER", "CARBS", "VITAMINS", "POT
                                    "WEIGHT", "CUPS"]]
```

1.3.3 Manipulation 3.3

Action: Slice the Master DataFrame to exclude column `FIBER` .

In [12]:

```
# Your implementation
MasterDataFrame.loc[:, MasterDataFrame.columns!='FIBER']
```

Out[12]:

	NAME	COMPANY	TYPE	RATING	SHELF	CALORIES	PROTEIN	SODIUM	CARBS	VITA
0	100% Natural Bran	Q	C	33.983679	3.0	120.0	3.0	15.0	8.0	
1	Cinnamon Toast Crunch	G	C	19.823573	2.0	120.0	1.0	210.0	13.0	
2	Muesli Raisins; Peaches; & Pecans	R	C	34.139765	3.0	150.0	4.0	150.0	16.0	
3	Cracklin' Oat Bran	K	C	40.448772	3.0	110.0	3.0	140.0	10.0	
4	Muesli Raisins; Dates; & Almonds	R	C	37.136863	3.0	150.0	4.0	95.0	16.0	
...	...	...	...	...	...	...	...	...	...	
73	Raisin Squares	K	C	55.333142	3.0	90.0	2.0	0.0	15.0	
74	Shredded Wheat	N	C	68.235885	1.0	80.0	2.0	0.0	16.0	
75	Puffed Wheat	Q	C	63.005645	3.0	50.0	2.0	0.0	10.0	
76	Puffed Rice	Q	C	60.756112	3.0	50.0	1.0	0.0	13.0	
77	Cocoa Pebbles	G	C	NaN	NaN	NaN	NaN	NaN	NaN	

78 rows × 15 columns

1.3.4 Manipulation 3.4

Action: Set the index on the Master DataFrame to `NAME` .

In [13]:

```
# Your implementation:

MasterDataFrame = MasterDataFrame.set_index("NAME")
```

1.3.5 Manipulation 3.5

Action: Change the values in the `COMPANY` column, using the following mapping:

- `A to American Home Food Products`
- `G to General Mills`
- `K to Kelloggs`
- `N to Nabisco`
- `P to Post`
- `Q to Quaker Oats`
- `R to Ralston Purina`

In [14]:

```
# Your implementation:
MasterDataFrame["COMPANY"].replace({"A": "American Home Food Products", "Ger
                                   "K":"Kelloggs","N":"Nabisco","P":"Post","Q":"(
```

1.3.6 Manipulation 3.6

Action: Change the values in the `TYPE` column, using the following mapping:

- `C to Cold`
- `H to Hot`

In [15]:

```
# Your implementation:

MasterDataFrame["TYPE"].replace({"C":"Cold","H":"Hot"},inplace=True)
```

1.3.7 Manipulation 3.7

Action: Remove any rows containing null values using the `dropna` method.

In [16]:

```
# Your implementation:

MasterDataFrame.dropna()
MasterDataFrame
```

Out[16]:

	COMPANY	TYPE	RATING	SHELF	CALORIES	PROTEIN	SODIUM	FIBER	CARBS
NAME									
100% Natural Bran	Quaker Oats	Cold	33.983679	3.0	120.0	3.0	15.0	2.0	8.0
Cinnamon Toast Crunch	General Mills	Cold	19.823573	2.0	120.0	1.0	210.0	0.0	13.0
Muesli Raisins; Peaches; & Pecans	Ralston Purina	Cold	34.139765	3.0	150.0	4.0	150.0	3.0	16.0
Cracklin' Oat Bran	Kelloggs	Cold	40.448772	3.0	110.0	3.0	140.0	4.0	10.0
Muesli Raisins; Dates; & Almonds	Ralston Purina	Cold	37.136863	3.0	150.0	4.0	95.0	3.0	16.0
...	...	...	...	...	...	...	...	...	...
Raisin Squares	Kelloggs	Cold	55.333142	3.0	90.0	2.0	0.0	2.0	15.0
Shredded Wheat	Nabisco	Cold	68.235885	1.0	80.0	2.0	0.0	3.0	16.0
Puffed Wheat	Quaker Oats	Cold	63.005645	3.0	50.0	2.0	0.0	1.0	10.0
Puffed Rice	Quaker Oats	Cold	60.756112	3.0	50.0	1.0	0.0	0.0	13.0
Cocoa Pebbles	General Mills	Cold	NaN	NaN	NaN	NaN	NaN	NaN	NaN

78 rows × 15 columns

1.3.8 Manipulation 3.8

Action: Preview the first ten (10) rows of data using `head` method.

In [17]:

```
# Your implementation:
print(MasterDataFrame.head(10))
```

```
                                     COMPANY  TYPE     RATING  SHELF  \
NAME                                                                   
100% Natural Bran                Quaker Oats  Cold  33.983679    3.0   
Cinnamon Toast Crunch          General Mills  Cold  19.823573    2.0   
Muesli Raisins; Peaches; & Pecans  Ralston Purina  Cold  34.139765    3.0   
Cracklin' Oat Bran                  Kelloggs  Cold  40.448772    3.0   
Muesli Raisins; Dates; & Almonds  Ralston Purina  Cold  37.136863    3.0   
Great Grains Pecan                      Post  Cold  45.811716    3.0   
Cheerios                       General Mills  Cold  50.764999    1.0   
Nutri-Grain Almond-Raisin           Kelloggs  Cold  40.692320    3.0   
Honey Graham Ohs                 Quaker Oats  Cold  21.871292    2.0   
Cap'n'Crunch                     Quaker Oats  Cold  18.042851    2.0   

                                   CALORIES  PROTEIN  SODIUM  FIBER  CARBS  \
NAME                                                                         
100% Natural Bran                     120.0      3.0    15.0    2.0    8.0   
Cinnamon Toast Crunch                 120.0      1.0   210.0    0.0   13.0   
Muesli Raisins; Peaches; & Pecans     150.0      4.0   150.0    3.0   16.0   
Cracklin' Oat Bran                    110.0      3.0   140.0    4.0   10.0   
Muesli Raisins; Dates; & Almonds      150.0      4.0    95.0    3.0   16.0   
Great Grains Pecan                    120.0      3.0    75.0    3.0   13.0   
Cheerios                              110.0      6.0   290.0    2.0   17.0   
Nutri-Grain Almond-Raisin             140.0      3.0   220.0    3.0   21.0   
Honey Graham Ohs                      120.0      1.0   220.0    1.0   12.0   
Cap'n'Crunch                          120.0      1.0   220.0    0.0   12.0   

                                   VITAMINS  POTASSIUM  FAT  SUGAR  WEIGHT  \
NAME                                                                         
100% Natural Bran                       0.0      135.0  5.0    8.0    1.00   
Cinnamon Toast Crunch                  25.0       45.0  3.0    9.0    1.00   
Muesli Raisins; Peaches; & Pecans      25.0      170.0  3.0   11.0    1.00   
Cracklin' Oat Bran                     25.0      160.0  3.0    7.0    1.00   
Muesli Raisins; Dates; & Almonds       25.0      170.0  3.0   11.0    1.00   
Great Grains Pecan                     25.0      100.0  3.0    4.0    1.00   
Cheerios                               25.0      105.0  2.0    1.0    1.00   
Nutri-Grain Almond-Raisin              25.0      130.0  2.0    7.0    1.33   
Honey Graham Ohs                       25.0       45.0  2.0   11.0    1.00   
Cap'n'Crunch                           25.0       35.0  2.0   12.0    1.00   

                                   CUPS  
NAME                                     
100% Natural Bran                  1.00  
Cinnamon Toast Crunch              0.75  
Muesli Raisins; Peaches; & Pecans  1.00  
Cracklin' Oat Bran                 0.50  
Muesli Raisins; Dates; & Almonds   1.00  
Great Grains Pecan                 0.33  
Cheerios                           1.25  
Nutri-Grain Almond-Raisin          0.67  
Honey Graham Ohs                   1.00  
Cap'n'Crunch                       0.75  
```

1.4 Part 4 - Analyzing the Data

In Part 4, you will provide useful insights about the data in the Master DataFrame using Pandas.

1.4.1 Analysis 4.1

Actions:

- Calculate the five-number summary statistics for the amount of potassium per serving: minimum (0th percentile), lower quartile (25th percentile), median (50th percentile), upper quartile (75 percentile), and maximum (100th percentile).
- Output the following (replacing with the correct data):
  - `Potassium Per Serving Statistics`

    - `--------------------------------`
    - `Minimum (0th Percentile): 0.00`
    - `Lower Quartile (25th Percentile): 0.00`
    - `Median (50th Percentile): 0.00`
    - `Upper Quartile (75th Percentile): 0.00`
    - `Maximum (100th Percentile): 0.00`

In [18]:

```
# Your implementation:
x = "Potassium Per Serving Statistics"
print(x)
print('-'*len(x))

print("Minimum (0th Percentile):{:.2f}".format(MasterDataFrame['POTASSIUM'].descr
print("Lower Quartile (25th Percentile):{:.2f}".format(MasterDataFrame['POTASSIUM
print("Median (50th Percentile):{:.2f}".format(MasterDataFrame['POTASSIUM'].descr
print("Upper Quartile (75th Percentile):{:.2f}".format(MasterDataFrame['POTASSIUM
print("Maximum (100th Percentile):{:.2f}".format(MasterDataFrame['POTASSIUM'].des
# print ("My number is {0:.1f}".format(myNum))
```

```
Potassium Per Serving Statistics
--------------------------------
Minimum (0th Percentile):-1.00
Lower Quartile (25th Percentile):40.00
Median (50th Percentile):90.00
Upper Quartile (75th Percentile):120.00
Maximum (100th Percentile):330.00
```

1.4.2 Analysis 4.2

Actions:

- Determine the top 10 cereals by Consumer Reports rating.
- Output the following dynamically (replacing with the correct data):
    - `Top 10 Cereals by Consumer Reports Rating`
    - `----------------------------------------`
    - `1. Crispix (Kelloggs) 46.90`
    - `2. Fruity Pebbles (Post) 28.03`
    - ...
    - `10. Trix (General Mills) 27.75`

In [19]:

```
# Your implementation:
import math as m
# x = MasterDataFrame['RATING'].sort_values(ascending=False)
new = MasterDataFrame.sort_values('RATING',ascending = False)
Company = new['COMPANY'].head(10)
New1 = list(Company.index)
Company_list = list(Company.values)
Rating = round(new['RATING'].head(10),2)
Rating_Num = Rating.values

x = "Top 10 Cereals by Consumer Reports Rating"
print(x)
print("-"*len(x))
i = 1
for i in range(10):
    print(str(i+1)+ ". " + New1[i] + " (" + Company_list[i] + ") " + (str(Rating_
```

```
Top 10 Cereals by Consumer Reports Rating
-----------------------------------------
1. All-Bran with Extra Fiber (Kelloggs) 93.7
2. Shredded Wheat 'n'Bran (Nabisco) 74.47
3. Shredded Wheat spoon size (Nabisco) 72.8
4. 100% Bran (Nabisco) 68.4
5. Shredded Wheat (Nabisco) 68.24
6. Cream of Wheat (Quick) (Nabisco) 64.53
7. Puffed Wheat (Quaker Oats) 63.01
8. Puffed Rice (Quaker Oats) 60.76
9. Nutri-grain Wheat (Kelloggs) 59.64
10. All-Bran (Kelloggs) 59.43
```

1.4.2.1 Analysis 4.3

Actions:

Determine the "unhealthy" cereals which have at least 10 grams of sugar and at least 2 grams of fat per serving.

Output the following dynamically (replacing with the correct data):

Most Unhealthy Cereals

---

1. Cap'n'Crunch (Quaker Oats)

2. Fruity Pebbles (Post)

...

N. Honey-comb (Post)

In [20]:
```
MasterDataFrame
```

Out[20]:

	COMPANY	TYPE	RATING	SHELF	CALORIES	PROTEIN	SODIUM	FIBER	CARBS
NAME									
100% Natural Bran	Quaker Oats	Cold	33.983679	3.0	120.0	3.0	15.0	2.0	8.0
Cinnamon Toast Crunch	General Mills	Cold	19.823573	2.0	120.0	1.0	210.0	0.0	13.0
Muesli Raisins; Peaches; & Pecans	Ralston Purina	Cold	34.139765	3.0	150.0	4.0	150.0	3.0	16.0
Cracklin' Oat Bran	Kelloggs	Cold	40.448772	3.0	110.0	3.0	140.0	4.0	10.0
Muesli Raisins; Dates; & Almonds	Ralston Purina	Cold	37.136863	3.0	150.0	4.0	95.0	3.0	16.0
...	...	...	...	...	...	...	...	...	...
Raisin Squares	Kelloggs	Cold	55.333142	3.0	90.0	2.0	0.0	2.0	15.0
Shredded Wheat	Nabisco	Cold	68.235885	1.0	80.0	2.0	0.0	3.0	16.0
Puffed Wheat	Quaker Oats	Cold	63.005645	3.0	50.0	2.0	0.0	1.0	10.0
Puffed Rice	Quaker Oats	Cold	60.756112	3.0	50.0	1.0	0.0	0.0	13.0
Cocoa Pebbles	General Mills	Cold	NaN	NaN	NaN	NaN	NaN	NaN	NaN

78 rows × 15 columns

In [21]:
```
# Your implementation:

condition1 = MasterDataFrame['SUGAR'] >= 10
condition2 = MasterDataFrame['FAT'] >=2
UnhealthyCereals = MasterDataFrame[condition1 & condition2]
UnhealthyCereals
Name = list(UnhealthyCereals.index)
CompanyCereal = list(UnhealthyCereals['COMPANY'])


for i in range(UnhealthyCereals.shape[0]):
    print(str(i+1) + ". " + Name[i] + " (" + CompanyCereal[i] + ")")

#x = "Most Unhealthy Cereals"
#print(x)
#print("-"*len(x))
```

```
1. Muesli Raisins; Peaches; & Pecans (Ralston Purina)
2. Muesli Raisins; Dates; & Almonds (Ralston Purina)
3. Honey Graham Ohs (Quaker Oats)
4. Cap'n'Crunch (Quaker Oats)
5. Apple Cinnamon Cheerios (General Mills)
6. Oatmeal Raisin Crisp (General Mills)
7. Fruit & Fibre Dates; Walnuts; and Oats (Post)
8. Mueslix Crispy Blend (Kelloggs)
```

1.5 Part 5 - Visualizing the Data

In Part 5, you will create the following charts to help visualize the data in the Master DataFrame using Pandas, Matplotlib, or Seaborn.

1.5.1 Visualization 5.1

Actions:

- Display a bar chart showing the amount of protein per serving for all cereals, sorted from most protein to least protein.
- Label the title, x-axis, and y-axis.

Hints:

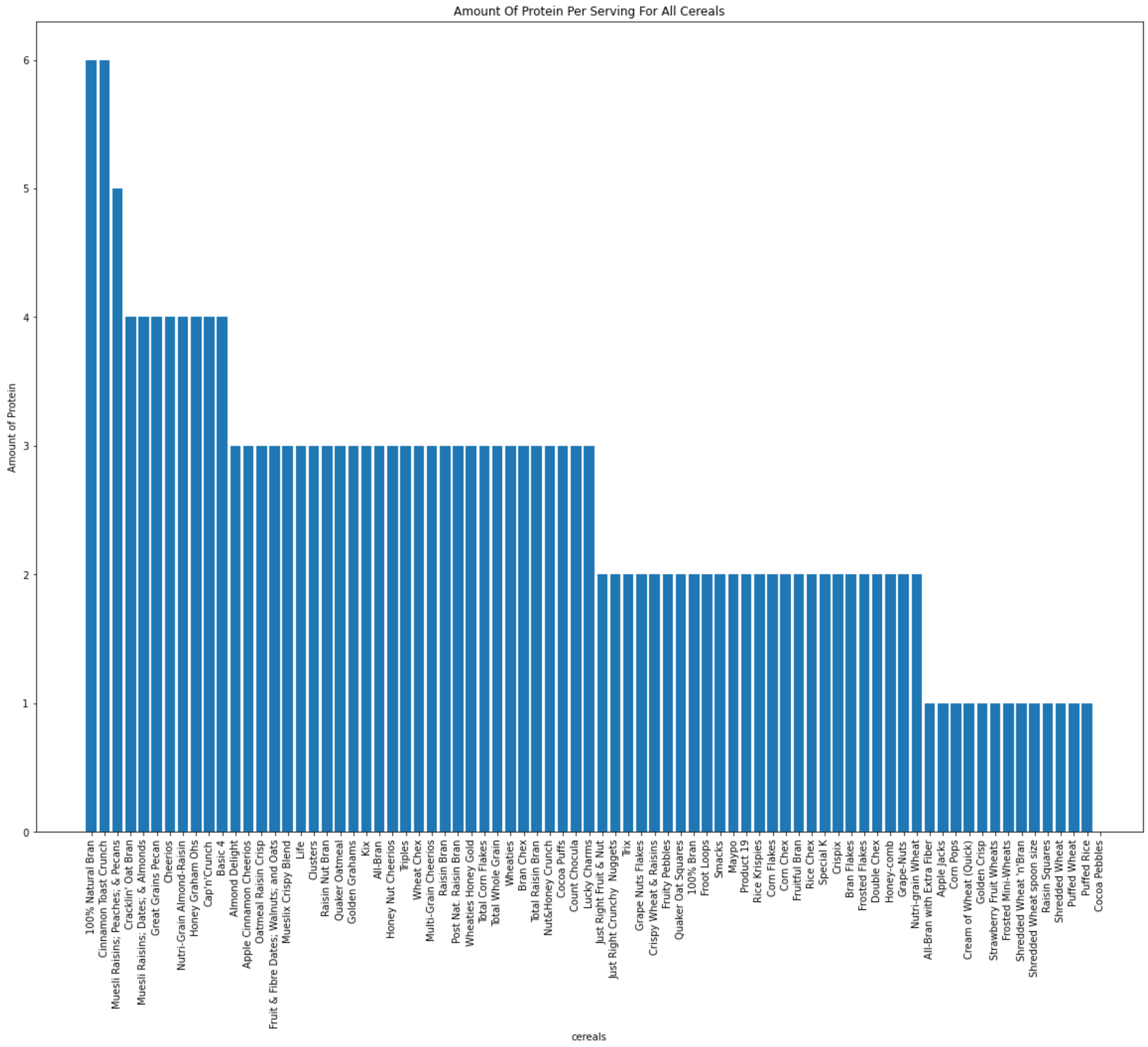
- You can use Pandas, Matplotlib, or Seaborn to create this bar chart.

In [22]:

```
# Your implementation:
import matplotlib.pyplot as plot
MasterDataFrame["PROTEIN"]
# create a figure and axis
figure, axis = plot.subplots(figsize = (20,15))

# count the occurrence of each score
count_data = MasterDataFrame["PROTEIN"]

# get the x and y data

# x axis data
cereals = count_data.index

# y axis data
Protein = count_data.sort_values(ascending=False)

# create the bar chart
axis.bar(cereals, Protein)

# set the title and labels
axis.set_title ("Amount Of Protein Per Serving For All Cereals")
axis.set_xlabel ('cereals')
plot.xticks(rotation = 90)
axis.set_ylabel ("Amount of Protein")
```

Out[22]:

```
Text(0, 0.5, 'Amount of Protein')
```



1.5.2 Visualization 5.2

Actions:

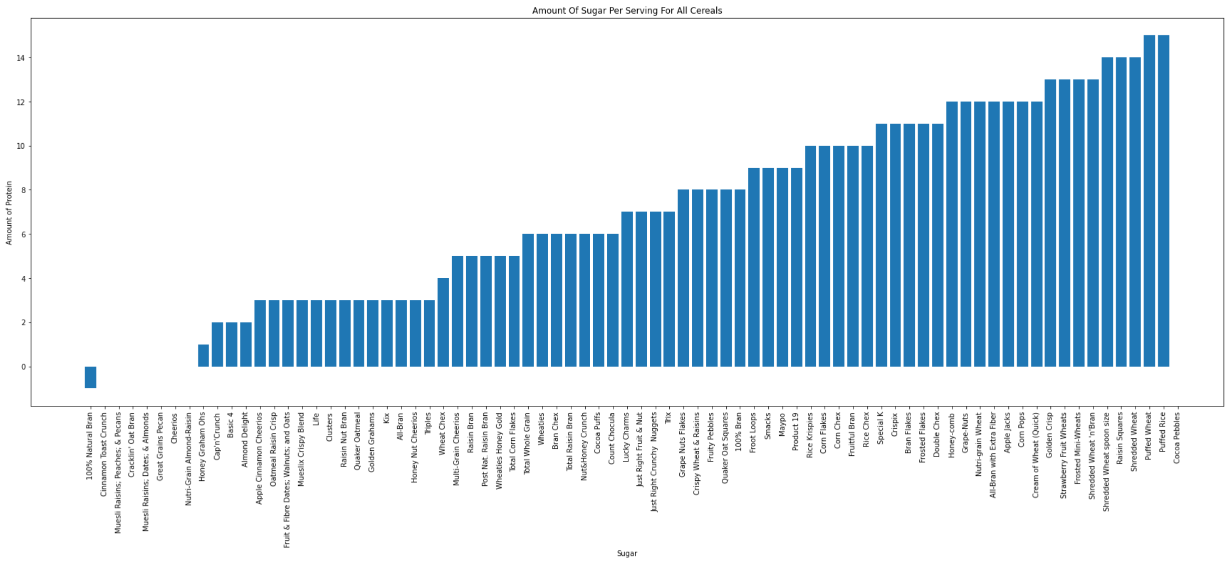
- Display a bar chart showing the amount of sugar per serving for all cereals, sorted from least sugar to most sugar.
- Label the title, x-axis, and y-axis.

Hints:

- You can use Pandas, Matplotlib, or Seaborn to create this bar chart.

In [23]:
```
# Your implementation:
import matplotlib.pyplot as plot
MasterDataFrame["SUGAR"]
# create a figure and axis
figure, axis = plot.subplots(figsize = (30,10))

# count the occurrence of each score
count_data = MasterDataFrame["SUGAR"]

# get the x and y data

# x axis data
Cereals = count_data.index

# y axis data
Sugar = count_data.sort_values(ascending=True)

# create the bar chart
axis.bar(Cereals,Sugar)

# set the title and labels
axis.set_title ("Amount Of Sugar Per Serving For All Cereals")
axis.set_xlabel ("Sugar")
plot.xticks(rotation = 90)
axis.set_ylabel ("Amount of Protein")
```

Out[23]:
```
Text(0, 0.5, 'Amount of Protein')
```



1.5.3 Visualization 5.3

Actions:

- Display a pie chart showing the percentage of cereals grouped by their shelf placement in supermarkets.
- Label the title, x-axis, and y-axis.

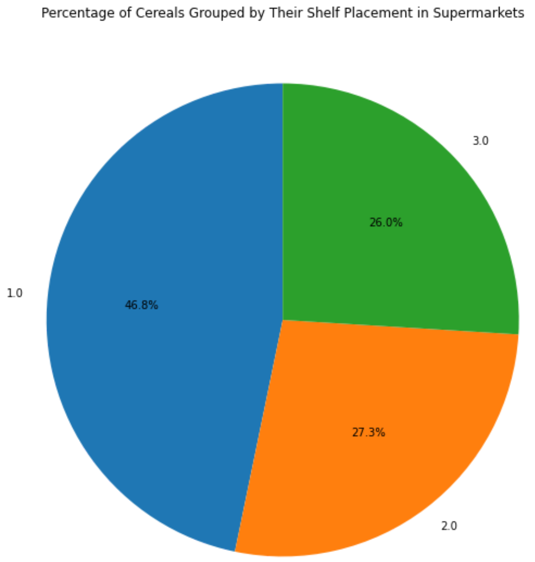
Hints:

- You can use Pandas, Matplotlib, or Seaborn to create this pie chart.

In [24]:

```
# Your implementation:
figure, axis = plot.subplots(figsize = (20,10))

# create the labels for the data points
slice_labels = ["1.0","2.0","3.0"]

# create the percent sizes for the slices (slices will be ordered and plotted cou
slice_sizes = MasterDataFrame['SHELF'].value_counts()

s = 0
for size in slice_sizes:

    s = s + size

print ("Total size: " + str (s))

# set the title and labels
axis.set_title ("Percentage of Cereals Grouped by Their Shelf Placement in Superm

# setting an "equal" aspect ration ensures that the pie chart is drawn as a circl
axis.axis ("equal")

# explode one of the slices
explode_slices = ( 0, 0, 0)

# set the formatting for the chart values
value_formatting = "%.1f%%"

# set the starting angle of the first slice
first_slice_start_angle = 90

# create the pie chart
axis.pie (slice_sizes, labels = slice_labels, autopct = value_formatting, explode

plot.show()
```

```
Total size: 77
```



1.5.4 Visualization 5.4

Actions:

- Display a histogram showing the number of cereals produced by each company.
- Label the title, x-axis, and y-axis.

Hints:

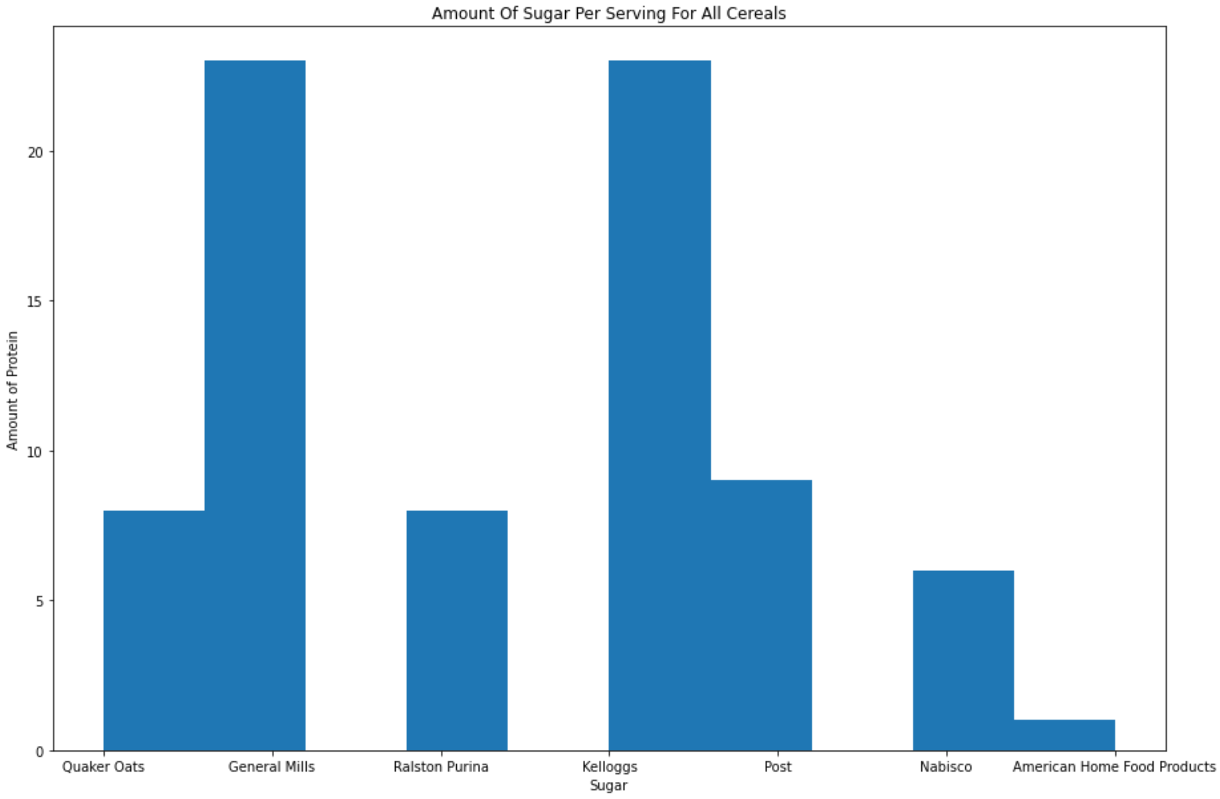
- You can use Pandas, Matplotlib, or Seaborn to create this histogram.

In [25]:

```
import matplotlib.pyplot as plot
MasterDataFrame["SUGAR"]
# create a figure and axis
figure, axis = plot.subplots(figsize = (15,10))

# count the occurrence of each score
axis.hist(MasterDataFrame['COMPANY'])

# get the x and y data

# x axis data


# y axis data


# create the bar chart


# set the title and labels
axis.set_title ("Amount Of Sugar Per Serving For All Cereals")
axis.set_xlabel ("Sugar")
axis.set_ylabel ("Amount of Protein")
```

Out[25]:

```
Text(0, 0.5, 'Amount of Protein')
```



1.5.5 Visualization 5.5

Actions:

- Display a histogram showing the number of cereals and their FDA-recommended daily vitamins and minerals per serving.
- Label the title, x-axis, and y-axis.

Hints:

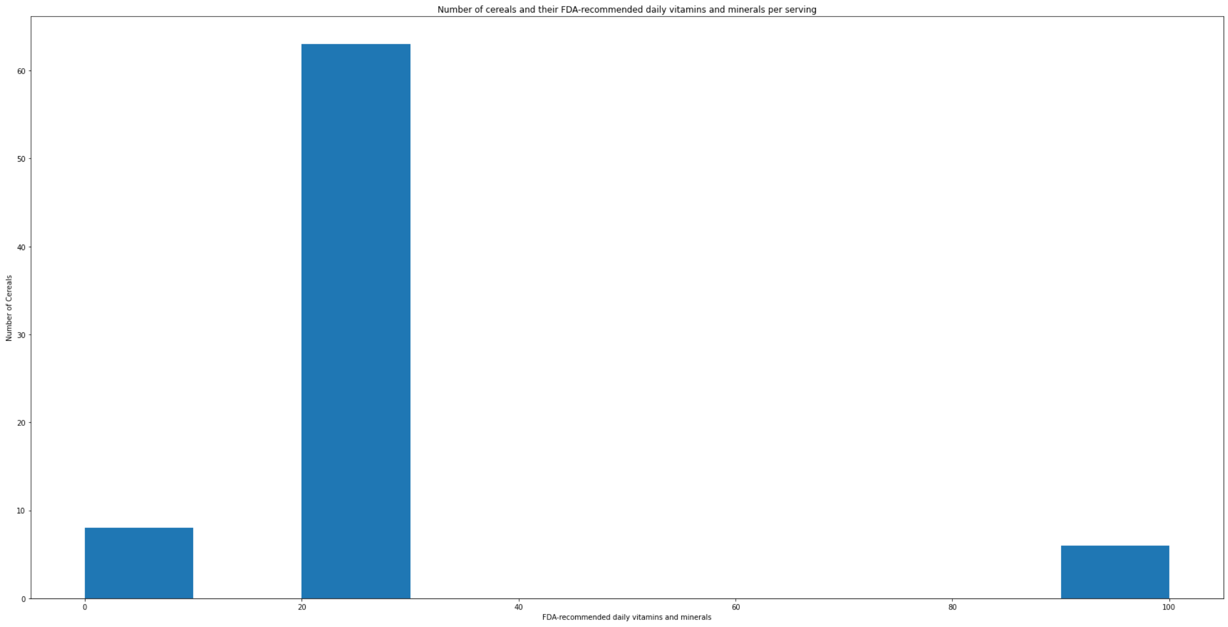
- You can use Pandas, Matplotlib, or Seaborn to create this histogram.

In [38]:

```
# Your implementation:
# create the figure and axis
figure, axis = plot.subplots(figsize = (30,15))

# plot the histogram
axis.hist(MasterDataFrame["VITAMINS"])

# set the title and labels
axis.set_title ("Number of cereals and their FDA-recommended daily vitamins and m
axis.set_xlabel ("FDA-recommended daily vitamins and minerals")
axis.set_ylabel ("Number of Cereals")
```

Out[38]:

```
Text(0, 0.5, 'Number of Cereals')
```



In [27]:

```
MasterDataFrame['VITAMINS'].unique()
```

Out[27]:

```
array([  0.,  25., 100.,  nan])
```

1.5.6 Visualization 5.6

Actions:

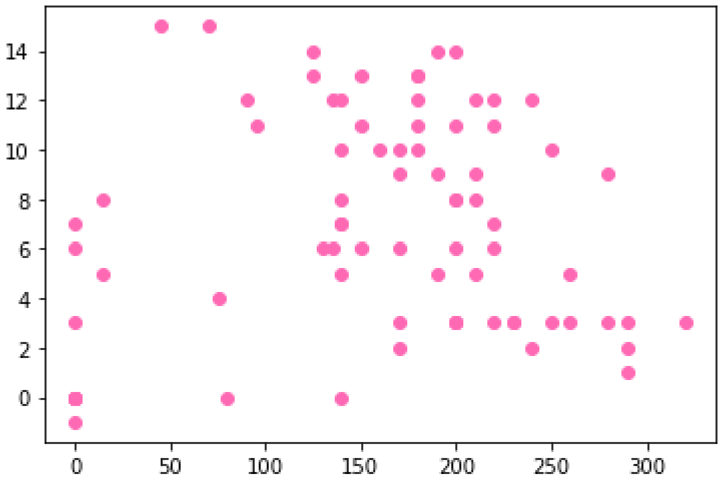
- Display a scatter plot showing the number of milligrams of sodium per serving vs. the number of grams of sugar per serving.
- Label the title, x-axis, and y-axis.

Hints:

- You can use Pandas, Matplotlib, or Seaborn to create this scatter plot.

```
In [28]: # Your implementation:

import pandas as pd
import matplotlib.pyplot as plot
import numpy as np

# Load the data sets

x = MasterDataFrame["SODIUM"]
y = MasterDataFrame["SUGAR"]
# create the scatter plot
#MasterDataFrame.plot.scatter(x = "SODIUM", y = "SUGAR", title = "number of milli
plot.scatter(x,y, color = 'hotpink')
```

```
Out[28]: <matplotlib.collections.PathCollection at 0x1d80ce2da60>
```



1.5.7 Visualization 5.7

Actions:

- Display a boxplot (also known as a five-number summary) showing the minimum (0th percentile), lower quartile (25th percentile), median (50th percentile), upper quartile (75 percentile), and maximum (100th percentile) for the amount of sugar for all cereals.
- Label the title, x-axis, and y-axis.

Hints:

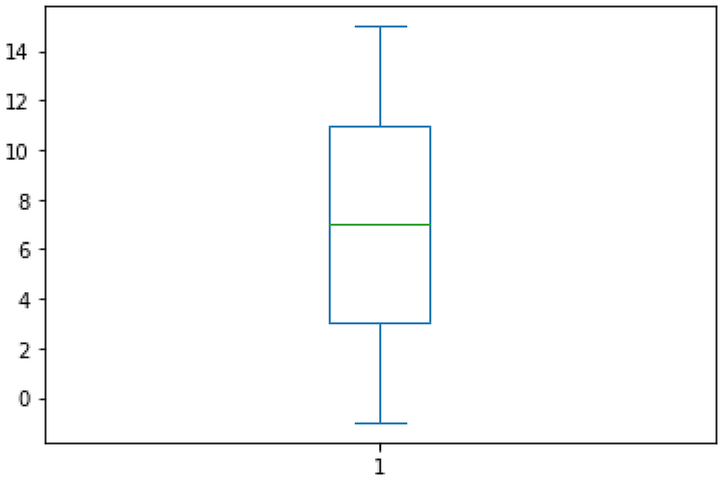
- You can use Pandas, Matplotlib, or Seaborn to create this box plot.

In [32]:

```
import matplotlib.pyplot as plot
Data_Sugar = MasterDataFrame.loc[MasterDataFrame.index,'SUGAR']
Data_Sugar.plot(kind='box')
plot.boxplot(MasterDataFrame["SUGAR"])

axis.set_title ("Quantiles for Amount of Sugar For All Cereals")
axis.set_xlabel ("Vitamins")
axis.set_ylabel ("Amount of Protein")
```

Out[32]:

```
Text(17.200000000000003, 0.5, 'Amount of Protein')
```



In [30]:

```
MasterDataFrame['SUGAR']
```

Out[30]:

```
NAME
100% Natural Bran                       8.0
Cinnamon Toast Crunch                   9.0
Muesli Raisins; Peaches; & Pecans      11.0
Cracklin' Oat Bran                      7.0
Muesli Raisins; Dates; & Almonds       11.0
                                       ...
Raisin Squares                          6.0
Shredded Wheat                          0.0
Puffed Wheat                            0.0
Puffed Rice                             0.0
Cocoa Pebbles                           NaN
Name: SUGAR, Length: 78, dtype: float64
```

In [31]:

```
MasterDataFrame['SUGAR'].describe()
```

Out[31]:

```
count    77.000000
mean      6.922078
std       4.444885
min      -1.000000
25%       3.000000
50%       7.000000
75%      11.000000
max      15.000000
Name: SUGAR, dtype: float64
```