Cognorise Infotech Internship

Name: Anuttama Mondal

Task 2: 80 CEREALS

Problem Statement:

If you like to eat cereal, do yourself a favor and avoid this dataset at all costs. After seeing these data it will never be the same for me to eat Fruity Pebbles again.

Import Libraries:

In [1]:
```
import pandas as pd         #For data manipulation and analysis
import numpy as mp          # For numerical computation and handling arrays
import matplotlib.pyplot as plt #For data visualization
import seaborn as sns           # For enhanced data visualization
from sklearn.model_selection import train_test_split      #For Data splitting
from datetime import datetime               #For Working with dates and times
import warnings
warnings.filterwarnings("ignore")           #disable warning
%matplotlib inline
```

In [2]:
```
df= pd.read_csv("cereal.csv")
```

In [3]:
```
df
```

Out[3]:

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
0	100% Bran	N	C	70	4	1	130	10.0	5.0	6	280	25	3	1.0	0.33	68.402973
1	100% Natural Bran	Q	C	120	3	5	15	2.0	8.0	8	135	0	3	1.0	1.00	33.983679
2	All-Bran	K	C	70	4	1	260	9.0	7.0	5	320	25	3	1.0	0.33	59.425505
3	All-Bran with Extra Fiber	K	C	50	4	0	140	14.0	8.0	0	330	25	3	1.0	0.50	93.704912
4	Almond Delight	R	C	110	2	2	200	1.0	14.0	8	-1	25	3	1.0	0.75	34.384843
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
72	Triples	G	C	110	2	1	250	0.0	21.0	3	60	25	3	1.0	0.75	39.106174
73	Trix	G	C	110	1	1	140	0.0	13.0	12	25	25	2	1.0	1.00	27.753301
74	Wheat Chex	R	C	100	3	1	230	3.0	17.0	3	115	25	1	1.0	0.67	49.787445
75	Wheaties	G	C	100	3	1	200	3.0	17.0	3	110	25	1	1.0	1.00	51.592193
76	Wheaties Honey Gold	G	C	110	2	1	200	1.0	16.0	8	60	25	1	1.0	0.75	36.187559

77 rows × 16 columns

Checking with the Rows for storing the Length:

In [4]:
```
df_len=len(df)
df_len
```

Out[4]:
```
77
```

Displaying first and last rows and columns of the dataset:

In [5]:
```
df.head()
```

Out[5]:

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
0	100% Bran	N	C	70	4	1	130	10.0	5.0	6	280	25	3	1.0	0.33	68.402973
1	100% Natural Bran	Q	C	120	3	5	15	2.0	8.0	8	135	0	3	1.0	1.00	33.983679
2	All-Bran	K	C	70	4	1	260	9.0	7.0	5	320	25	3	1.0	0.33	59.425505
3	All-Bran with Extra Fiber	K	C	50	4	0	140	14.0	8.0	0	330	25	3	1.0	0.50	93.704912
4	Almond Delight	R	C	110	2	2	200	1.0	14.0	8	-1	25	3	1.0	0.75	34.384843

In [6]:
```
df.tail()
```

Out[6]:

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
72	Triples	G	C	110	2	1	250	0.0	21.0	3	60	25	3	1.0	0.75	39.106174
73	Trix	G	C	110	1	1	140	0.0	13.0	12	25	25	2	1.0	1.00	27.753301
74	Wheat Chex	R	C	100	3	1	230	3.0	17.0	3	115	25	1	1.0	0.67	49.787445
75	Wheaties	G	C	100	3	1	200	3.0	17.0	3	110	25	1	1.0	1.00	51.592193
76	Wheaties Honey Gold	G	C	110	2	1	200	1.0	16.0	8	60	25	1	1.0	0.75	36.187559

View the information:

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 77 entries, 0 to 76
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   name      77 non-null     object
 1   mfr       77 non-null     object
 2   type      77 non-null     object
 3   calories  77 non-null     int64
 4   protein   77 non-null     int64
 5   fat       77 non-null     int64
 6   sodium    77 non-null     int64
 7   fiber     77 non-null     float64
 8   carbo     77 non-null     float64
 9   sugars    77 non-null     int64
 10  potass    77 non-null     int64
 11  vitamins  77 non-null     int64
 12  shelf     77 non-null     int64
 13  weight    77 non-null     float64
 14  cups      77 non-null     float64
 15  rating    77 non-null     float64
dtypes: float64(5), int64(8), object(3)
memory usage: 9.8+ KB
```

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

Out[8]:
```
(77, 16)
```

checking for any null or missing values:

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
name        0
mfr         0
type        0
calories    0
protein     0
fat         0
sodium      0
fiber       0
carbo       0
sugars      0
potass      0
vitamins    0
shelf       0
weight      0
cups        0
rating      0
dtype: int64
```

In [10]:
```
df.describe()
```

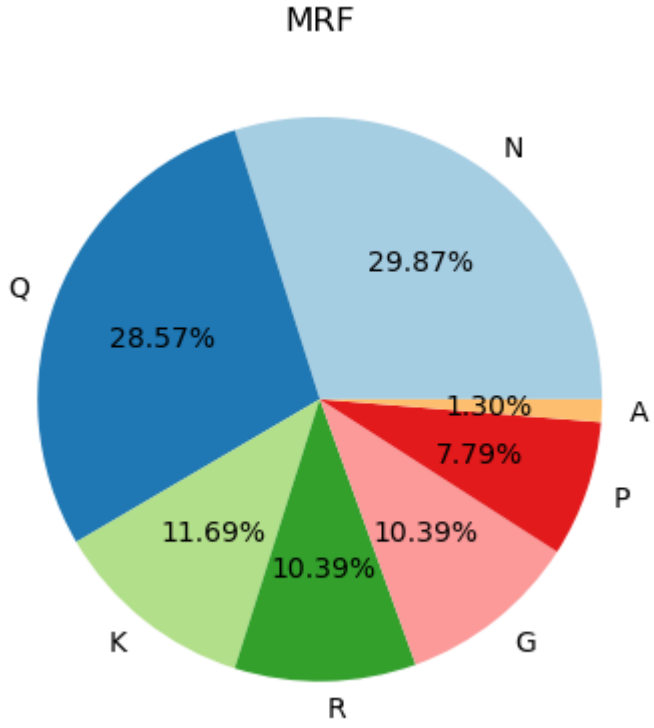
Out[10]:

	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rat
count	77.000000	77.000000	77.000000	77.000000	77.000000	77.000000	77.000000	77.000000	77.000000	77.000000	77.000000	77.000000	77.000
mean	106.883117	2.545455	1.012987	159.675325	2.151948	14.597403	6.922078	96.077922	28.246753	2.207792	1.029610	0.821039	42.665
std	19.484119	1.094790	1.006473	83.832295	2.383364	4.278956	4.444885	71.286813	22.342523	0.832524	0.150477	0.232716	14.047
min	50.000000	1.000000	0.000000	0.000000	0.000000	-1.000000	-1.000000	-1.000000	0.000000	1.000000	0.500000	0.250000	18.042
25%	100.000000	2.000000	0.000000	130.000000	1.000000	12.000000	3.000000	40.000000	25.000000	1.000000	1.000000	0.670000	33.174
50%	110.000000	3.000000	1.000000	180.000000	2.000000	14.000000	7.000000	90.000000	25.000000	2.000000	1.000000	0.750000	40.400
75%	110.000000	3.000000	2.000000	210.000000	3.000000	17.000000	11.000000	120.000000	25.000000	3.000000	1.000000	1.000000	50.828
max	160.000000	6.000000	5.000000	320.000000	14.000000	23.000000	15.000000	330.000000	100.000000	3.000000	1.500000	1.500000	93.704

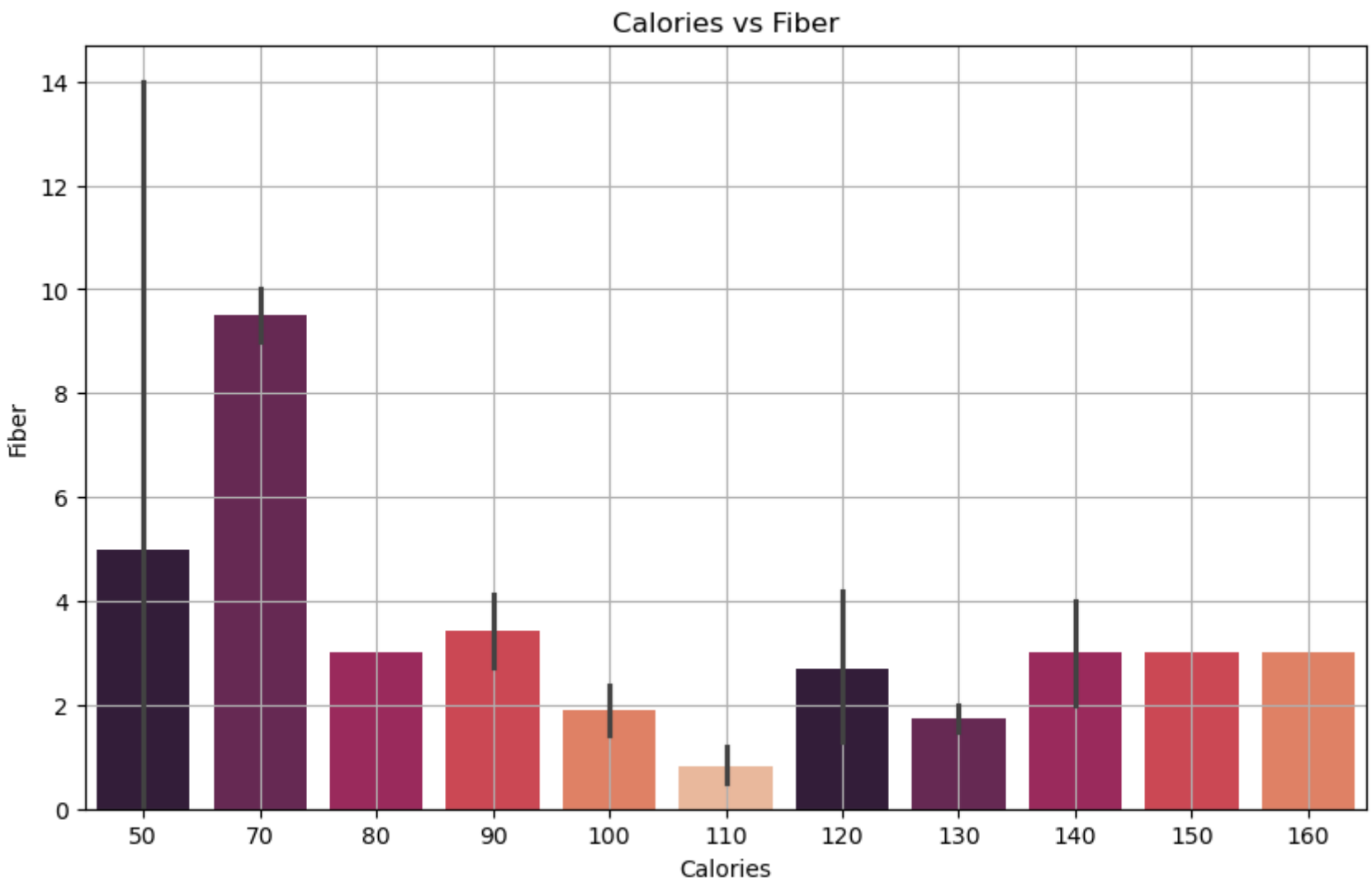
Using EDA(Exploratory Data Analysis):

In [12]:
```
colors= sns.color_palette('Paired')
labels=df['mfr'].dropna().unique()
plt.figure(figsize=(10,7))
plt.subplot(1,2,1)
plt.title('MRF')
plt.pie(df['mfr'].value_counts(),labels=labels,colors=colors, autopct='%.2f%%')
```

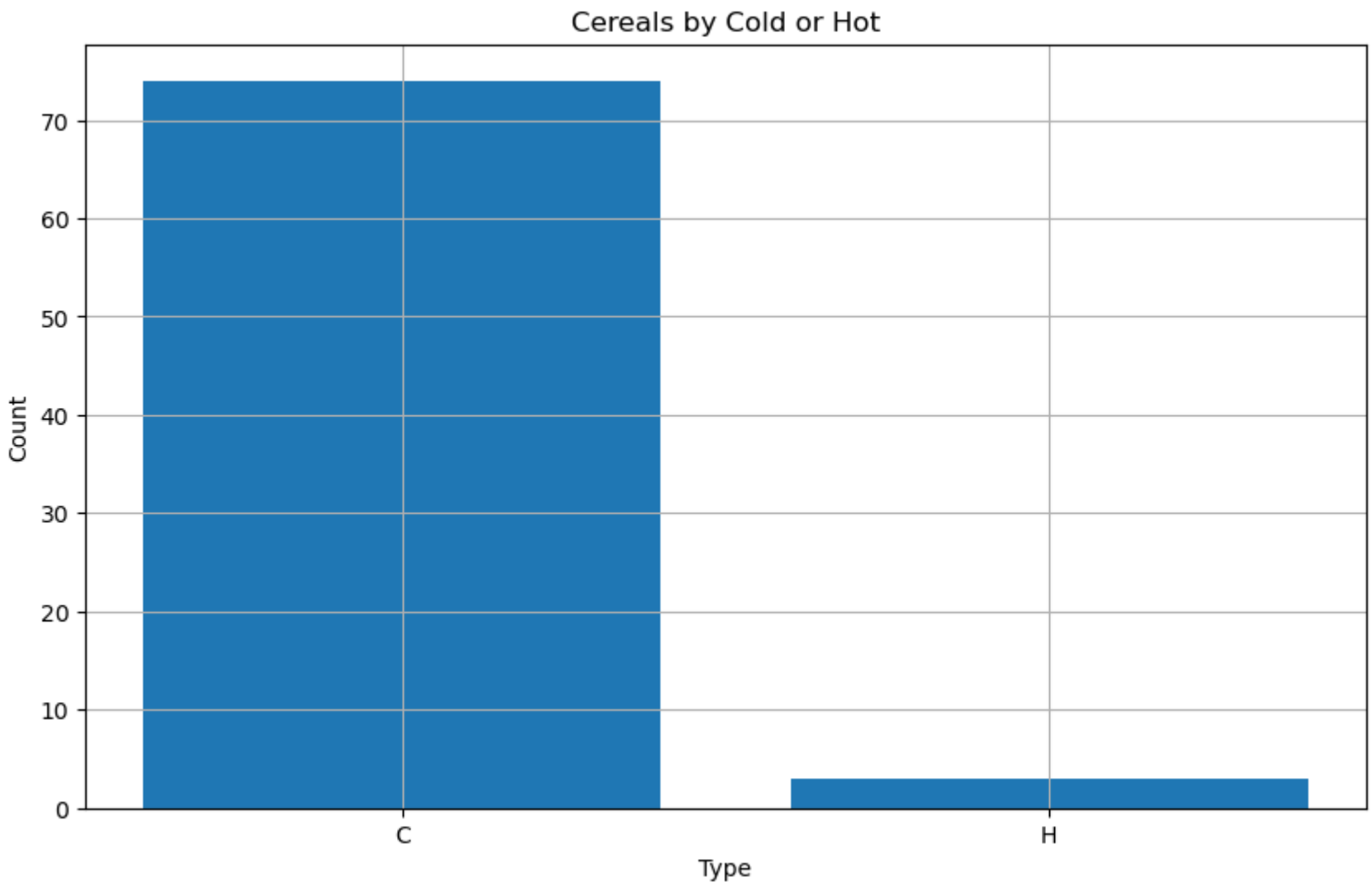
Out[12]:
```
([<matplotlib.patches.Wedge at 0x1b5cf662a60>,
  <matplotlib.patches.Wedge at 0x1b5cf6871c0>,
  <matplotlib.patches.Wedge at 0x1b5cf6878e0>,
  <matplotlib.patches.Wedge at 0x1b5cfd32040>,
  <matplotlib.patches.Wedge at 0x1b5cfd32760>,
  <matplotlib.patches.Wedge at 0x1b5cfd32e80>,
  <matplotlib.patches.Wedge at 0x1b5cfd3e5e0>],
 [Text(0.6501892803051899, 0.8872732948625353, 'N'),
  Text(-1.0266701582556739, 0.39490300853155025, 'Q'),
  Text(-0.6858387877960629, -0.8600146261281998, 'K'),
  Text(0.022438373412025387, -1.0997711213696342, 'R'),
  Text(0.6858387676659675, -0.860014642181415, 'G'),
  Text(1.0419233031728248, -0.352697930679257, 'P'),
  Text(1.0990845779679979, -0.04486747678339531, 'A')],
 [Text(0.35464869834828533, 0.483967251743201, '29.87%'),
  Text(-0.5600019045030948, 0.2154016410172092, '28.57%'),
  Text(-0.37409388425239787, -0.469098886979018, '11.69%'),
  Text(0.012239112770195666, -0.5998751571107095, '10.39%'),
  Text(0.3740938732723459, -0.46909889573531716, '10.39%'),
  Text(0.5683218017306316, -0.19238068946141287, '7.79%'),
  Text(0.599500678891635, -0.02447316915457926, '1.30%')])
```



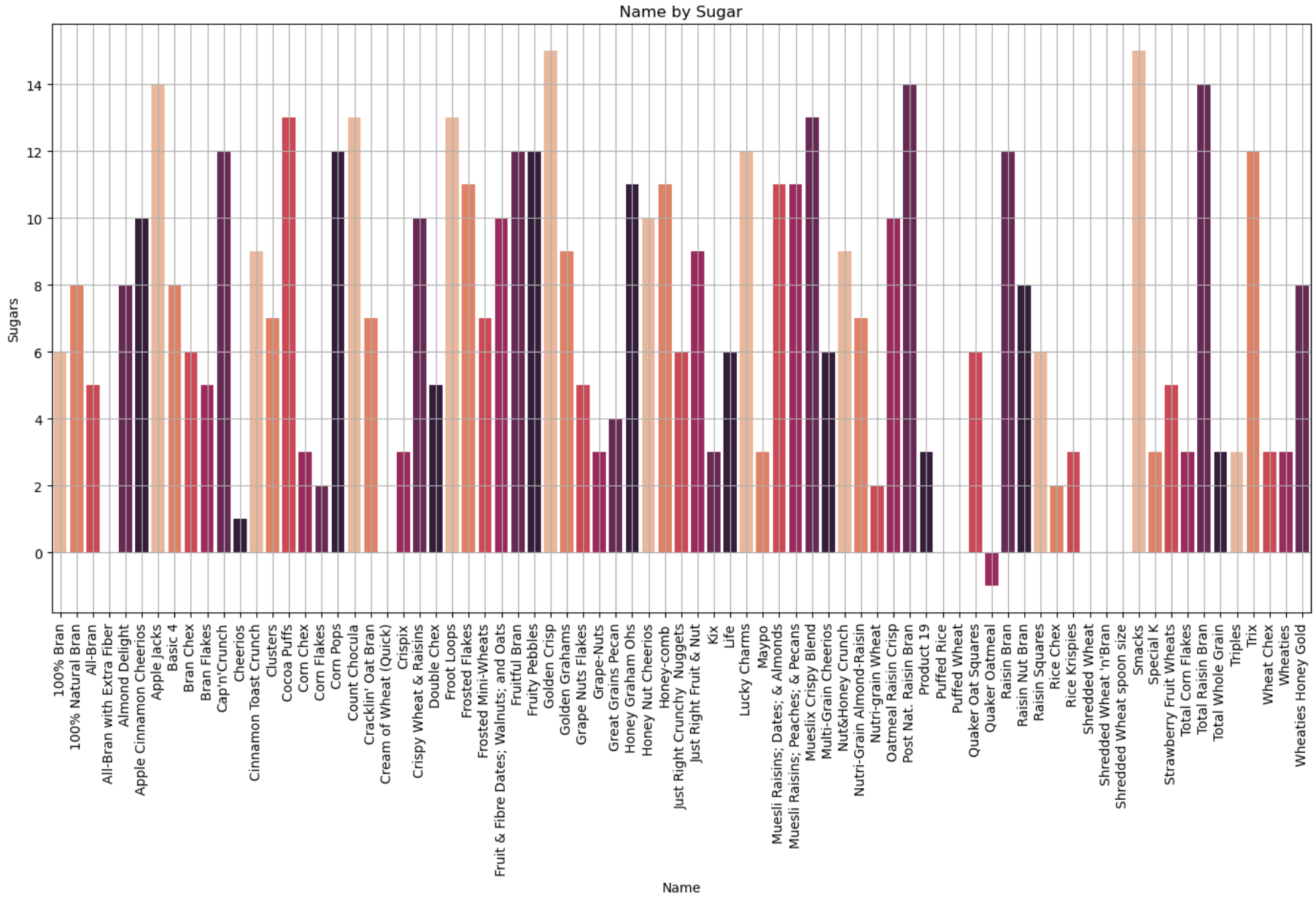
In [16]:
```
custom_palette = sns.color_palette("rocket",6)
plt.figure(figsize=(10,6))
sns.barplot(data=df, x='calories', y='fiber' ,palette=custom_palette)

plt.title('Calories vs Fiber')
plt.xlabel('Calories')
plt.ylabel('Fiber')
plt.grid(True)
plt.show()
```



In [31]:
```
plt.figure(figsize=(10,6))
plt.bar(df['type'].value_counts().index,df['type'].value_counts().values)
plt.title('Cereals by Cold or Hot')
plt.xlabel('Type')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```



In [32]:
```
custom_palette = sns.color_palette("rocket_r",6)
plt.figure(figsize=(17,8))
sns.barplot(data=df, x='name', y='sugars',palette=custom_palette)

plt.title('Name by Sugar')
plt.xlabel('Name')
plt.ylabel('Sugars')
plt.grid(True)
plt.xticks(rotation=90)
plt.show()
```

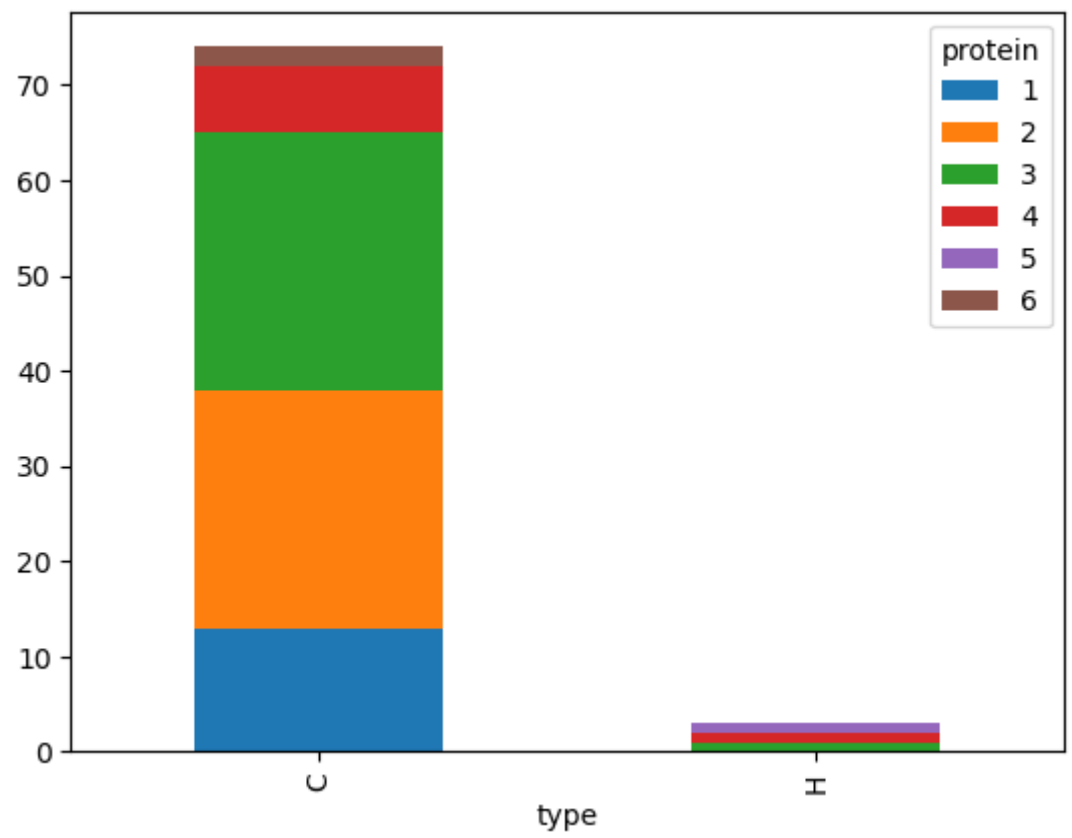


In [33]:
```
pd.crosstab(df['type'],df['protein'])
```

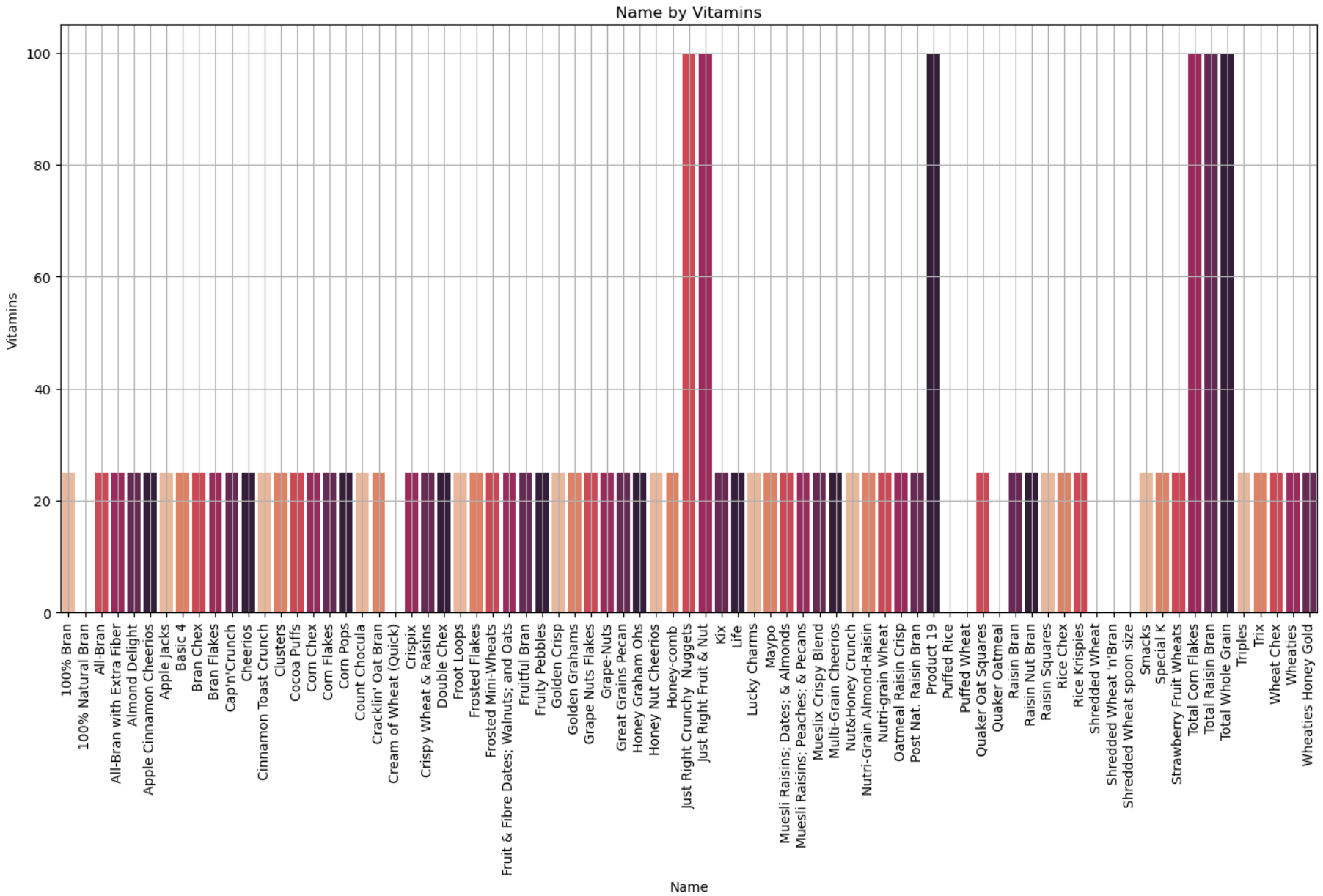
Out[33]:

type	protein 1	2	3	4	5	6
C	13	25	27	7	0	2
H	0	0	1	1	1	0

In [37]:
```
pd.crosstab(df['type'],df['protein']).plot(kind='bar',stacked='True')
plt.show()
```



In [38]:
```
custom_palette = sns.color_palette("rocket_r",6)
plt.figure(figsize=(17,8))
sns.barplot(data=df, x='name', y='vitamins',palette=custom_palette)

plt.title('Name by Vitamins')
plt.xlabel('Name')
plt.ylabel('Vitamins')
plt.grid(True)
plt.xticks(rotation=90)
plt.show()
```



In [40]:
```
pd.crosstab(df['name'],df['weight'])
```

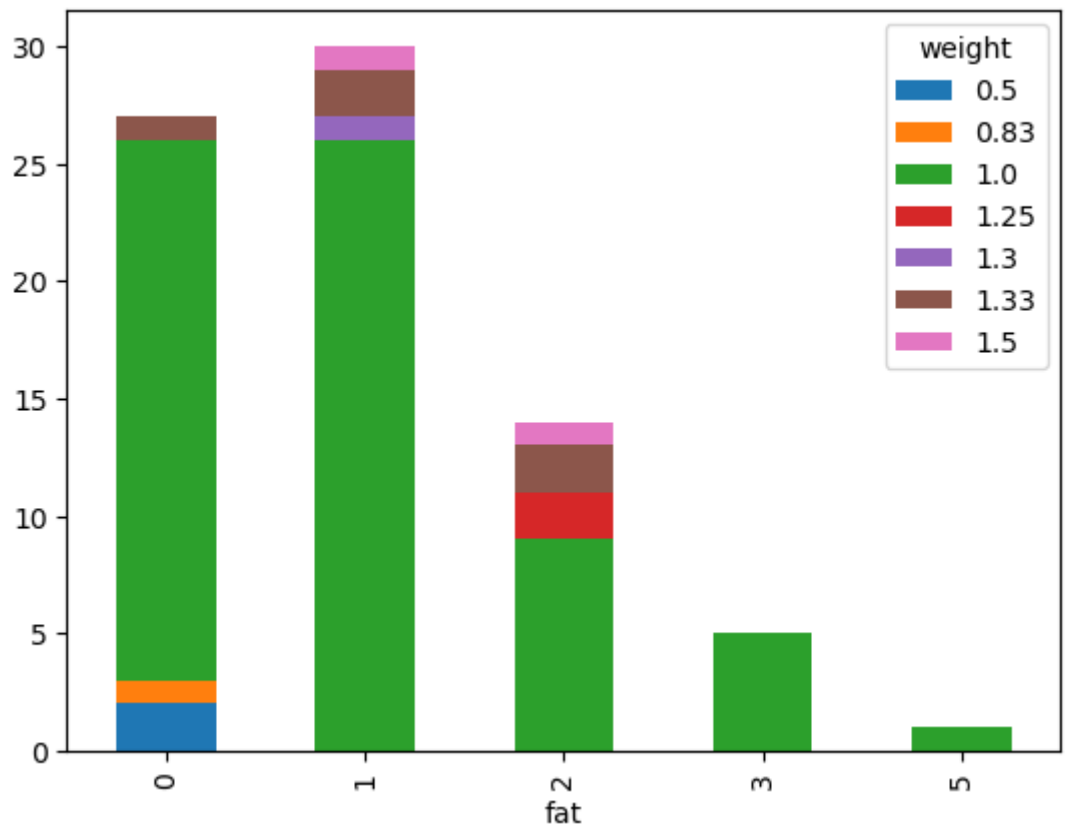
Out[40]:

name	weight 0.50	0.83	1.00	1.25	1.30	1.33	1.50
100% Bran	0	0	1	0	0	0	0
100% Natural Bran	0	0	1	0	0	0	0
All-Bran	0	0	1	0	0	0	0
All-Bran with Extra Fiber	0	0	1	0	0	0	0
Almond Delight	0	0	1	0	0	0	0
...	...	...	...	...	...	...	...
Triples	0	0	1	0	0	0	0
Trix	0	0	1	0	0	0	0
Wheat Chex	0	0	1	0	0	0	0
Wheaties	0	0	1	0	0	0	0
Wheaties Honey Gold	0	0	1	0	0	0	0

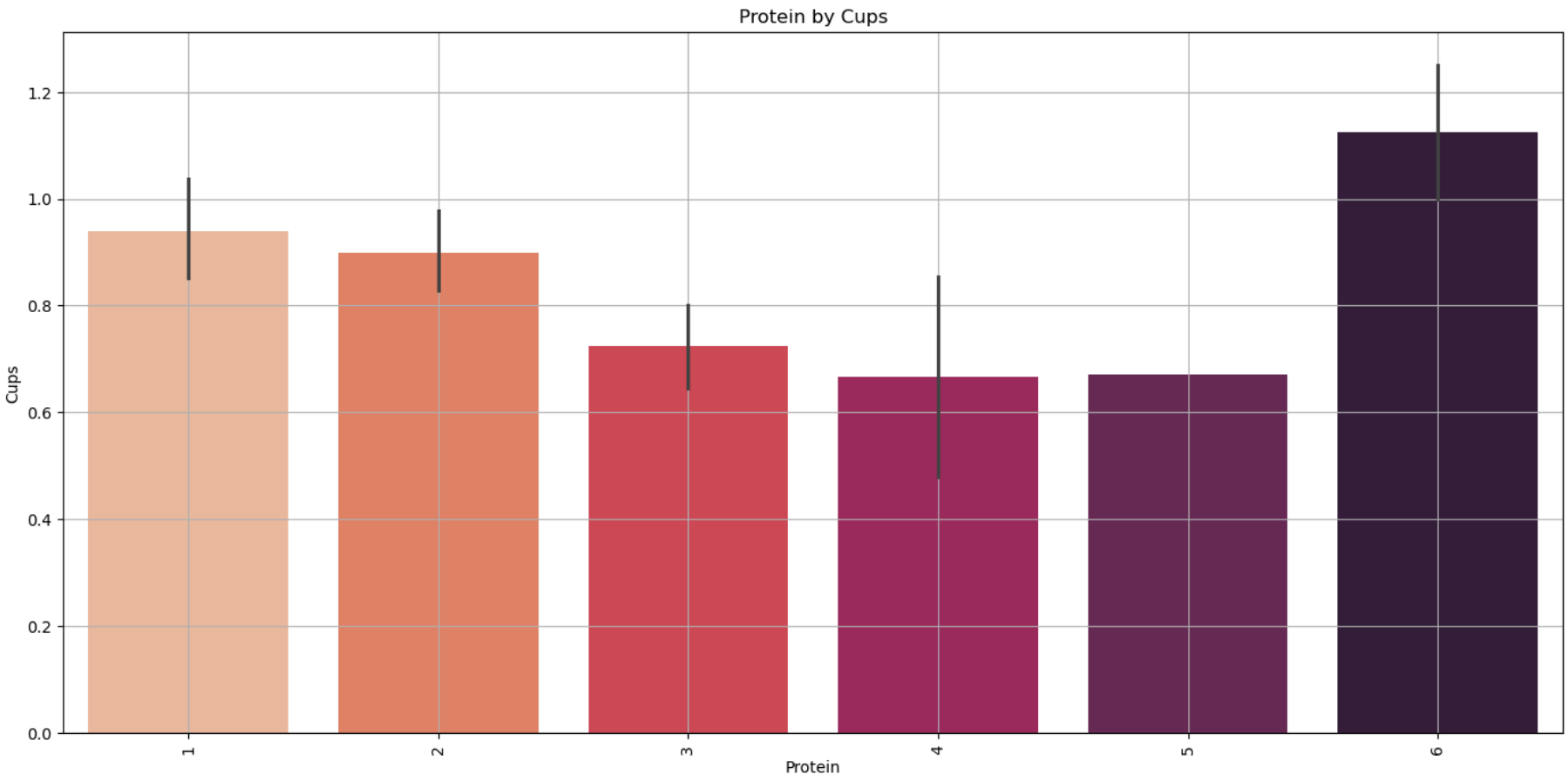
77 rows × 7 columns

In [43]:
```
plt.figure(figsize=(16,7))
pd.crosstab(df['fat'],df['weight']).plot(kind='bar',stacked='True')
plt.show()
```

```
<Figure size 1600x700 with 0 Axes>
```

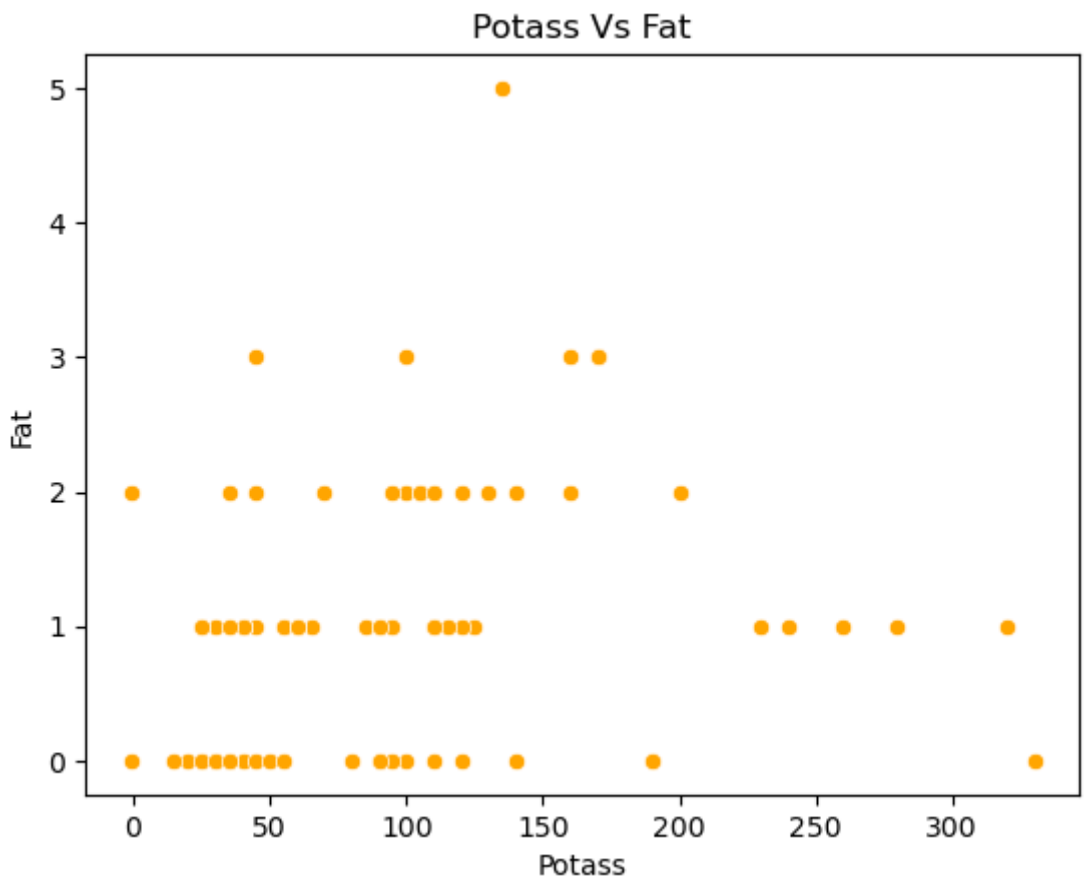


In [44]:
```
custom_palette = sns.color_palette("rocket_r",6)
plt.figure(figsize=(17,8))
sns.barplot(data=df, x='protein', y='cups',palette=custom_palette)

plt.title('Protein by Cups')
plt.xlabel('Protein')
plt.ylabel('Cups')
plt.grid(True)
plt.xticks(rotation=90)
plt.show()
```



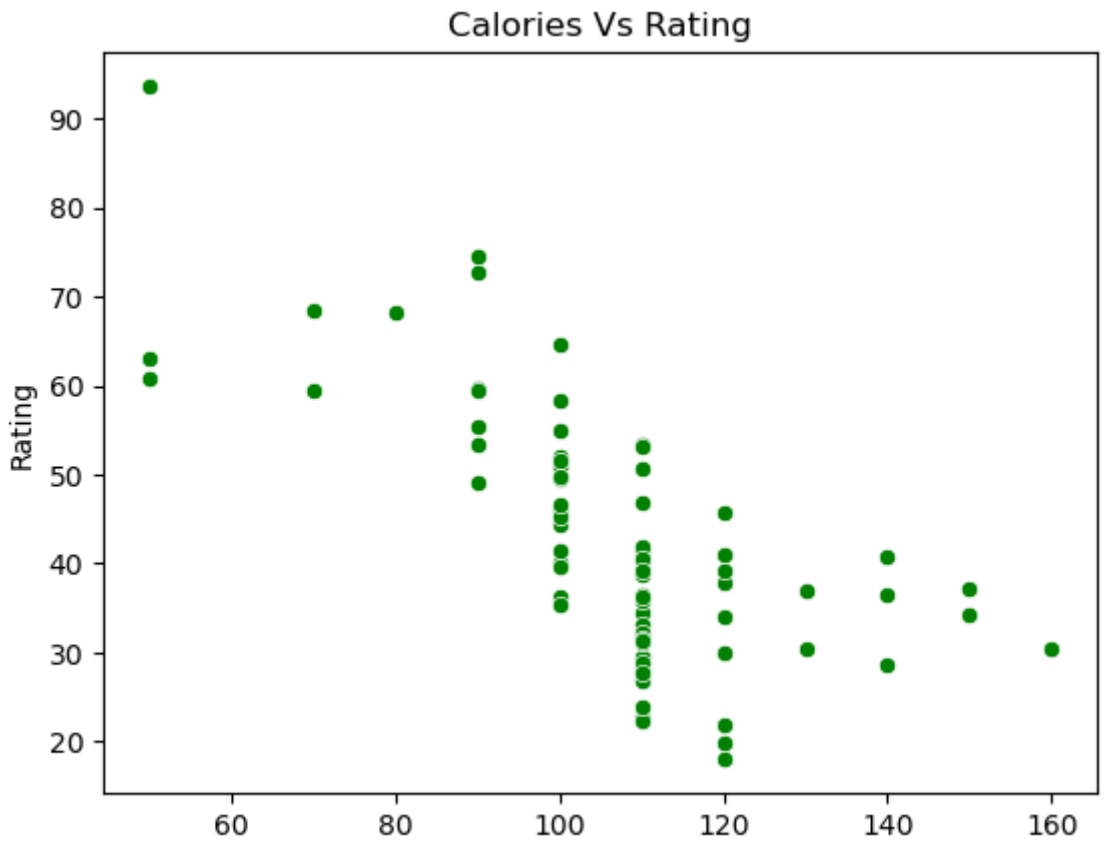
In [47]:
```
sns.scatterplot(x = 'potass', y = 'fat',color='orange',data = df)
plt.title("Potass Vs Fat")
plt.xlabel("Potass")
plt.ylabel("Fat")
```

Out[47]:
```
Text(0, 0.5, 'Fat')
```

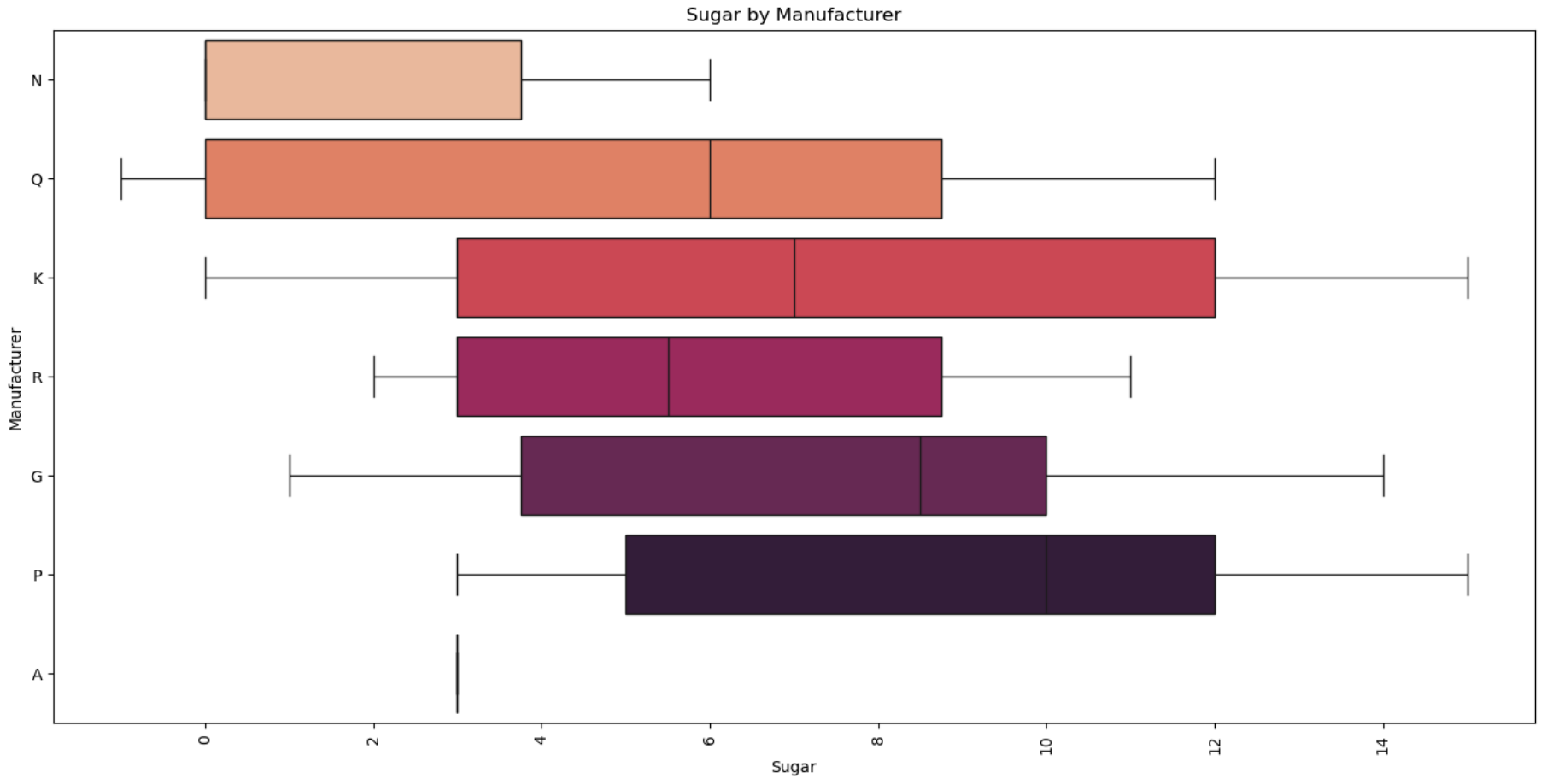


In [48]:
```
sns.scatterplot(x = 'calories', y = 'rating',color='green',data = df)
plt.title("Calories Vs Rating")
plt.xlabel("Calories")
plt.ylabel("Rating")
```

Out[48]:
```
Text(0, 0.5, 'Rating')
```

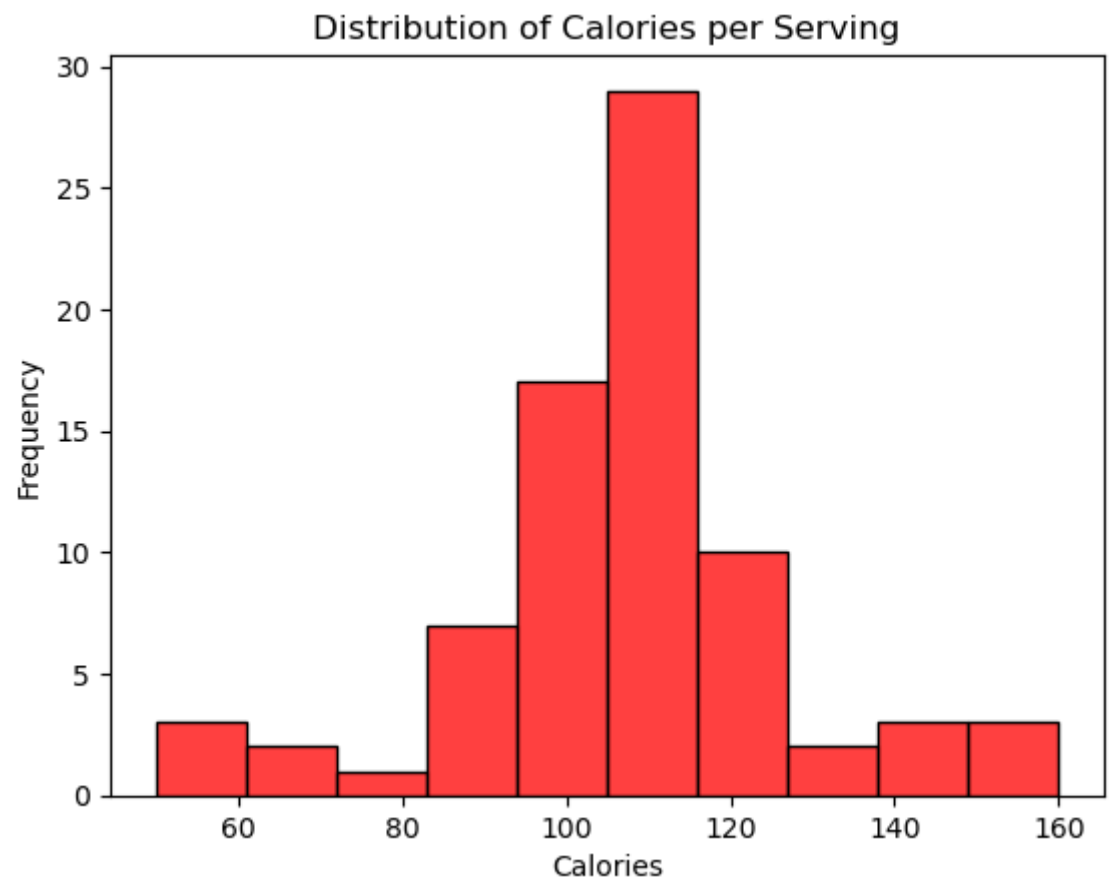


In [50]:
```
custom_palette = sns.color_palette("rocket_r",6)
plt.figure(figsize=(17,8))
sns.boxplot(data=df, x='sugars', y='mfr',palette=custom_palette)

plt.title('Sugar by Manufacturer')
plt.xlabel('Sugar')
plt.ylabel('Manufacturer')

plt.xticks(rotation=90)
plt.show()
```



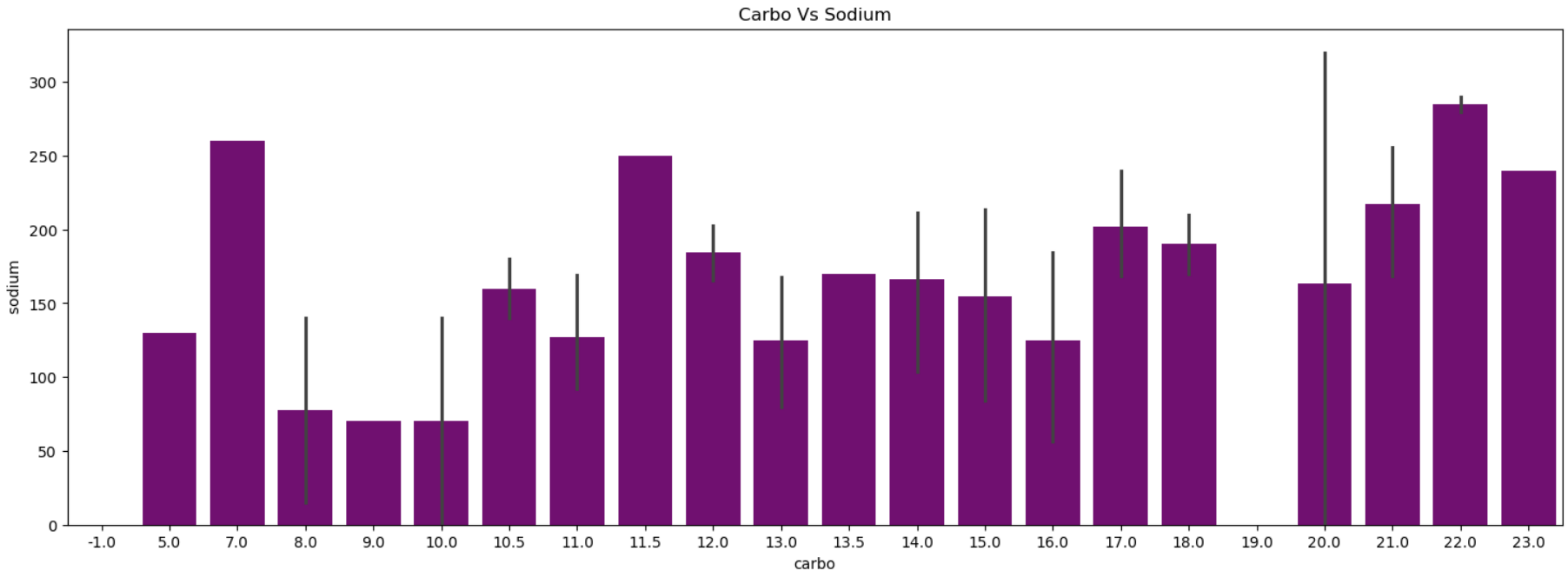
In [51]:
```
sns.histplot(df['calories'],color='red',bins=10);
plt.xlabel("Calories")
plt.ylabel("Frequency")
plt.title("Distribution of Calories per Serving")
```

Out[51]:
```
Text(0.5, 1.0, 'Distribution of Calories per Serving')
```



In [59]:
```
plt.figure(figsize=(18, 6))
sns.barplot(x = 'carbo', y = 'sodium',color='purple', data = df)
plt.title("Carbo Vs Sodium")
plt.xlabel("carbo")
plt.ylabel("sodium    ")
```

Out[59]:
```
Text(0, 0.5, 'sodium    ')
```


Carbo Vs Sodium
sodium
carbo
0
50
100
150
200
250
300
-1.0
5.0
7.0
8.0
9.0
10.0
10.5
11.0
11.5
12.0
13.0
13.5
14.0
15.0
16.0
17.0
18.0
19.0
20.0
21.0
22.0
23.0

THANK YOU