Cereals! Exploratory Data Analysis

This project analyzes the nutritional information of 80 different cereals using Python. It includes data cleaning, exploratory data analysis, and visualization.

Features

- Data cleaning and preprocessing
- Exploratory data analysis
- Visualizations of nutritional content

The purpose of this project is to investigate and understand the data provided. **The Goal** is to use a dataframe constructed within Python, perform a cursory inspection of the provided dataset, and inform team members of your findings.

This activity has three parts:

Part 1: Understand the situation

- Prepare to understand and organize the provided taxi cab dataset and information.

Part 2: Understand the data

- Create a pandas dataframe for data learning, future exploratory data analysis (EDA), and statistical activities.

- Compile summary information about the data to inform next steps.

Part 3: Understand the variables

- Use insights from your examination of the summary data to guide deeper investigation into specific variables.

Task 1. Understand the situation

1. How can you best prepare to understand and organize the provided taxi cab information?

Task 2a. Build dataframe

Create a pandas dataframe for data learning, and future exploratory data analysis (EDA) and statistical activities.

Code the following,

- import pandas as pd. pandas is used for buidling dataframes.

- import numpy as np. numpy is imported with pandas

- df = pd.read_csv('cereal_data.csv')

In [6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt # visualizing data
%matplotlib inline
import seaborn as sns
```

In [3]:
```
df = pd.read_csv(r'cereal_data .csv', encoding= 'unicode_escape')
```

In [14]:
```
file_path = 'cereal_data .csv'  # Update with the correct file path
cereal_data = pd.read_csv(file_path)
```

Task 2b. Understand the data - Inspect the data

View and inspect summary information about the dataframe by coding the following:

1. df.head(10)
2. df.info()
3. df.describe()

In [7]:
```
df.head(10)
```

Out[7]:

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups
0	String	Categorical	Categorical	Int	Int	Int	Int	Float	Float	Int	Int	Int	Int	Float	Float
1	100% Bran	N	C	70	4	1	130	10	5	6	280	25	3	1	0.33
2	100% Natural Bran	Q	C	120	3	5	15	2	8	8	135	0	3	1	1
3	All-Bran	K	C	70	4	1	260	9	7	5	320	25	3	1	0.33
4	All-Bran with Extra Fiber	K	C	50	4	0	140	14	8	0	330	25	3	1	0.5
5	Almond Delight	R	C	110	2	2	200	1	14	8	-1	25	3	1	0.75
6	Apple Cinnamon Cheerios	G	C	110	2	2	180	1.5	10.5	10	70	25	1	1	0.75
7	Apple Jacks	K	C	110	2	0	125	1	11	14	30	25	2	1	1
8	Basic 4	G	C	130	3	2	210	2	18	8	100	25	3	1.33	0.75
9	Bran Chex	R	C	90	2	1	200	4	15	6	125	25	1	1	0.67

In [73]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 78 entries, 0 to 77
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   name      78 non-null     object
 1   mfr       75 non-null     object
 2   type      75 non-null     object
 3   calories  75 non-null     object
 4   protein   75 non-null     object
 5   fat       75 non-null     object
 6   sodium    75 non-null     object
 7   fiber     75 non-null     object
 8   carbo     75 non-null     object
 9   sugars    75 non-null     object
 10  potass    75 non-null     object
 11  vitamins  75 non-null     object
 12  shelf     75 non-null     object
 13  weight    75 non-null     object
 14  cups      75 non-null     object
 15  rating    75 non-null     object
dtypes: object(16)
memory usage: 9.9+ KB
```

In [8]:

```
df.describe()
```

Out[8]:

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
count	78	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75
unique	77	8	3	11	7	6	26	14	23	18	35	4	4	8	13	75
top	Muesli Raisins	K	C	110	3	1	0	0	13	3	90	25	3	1	1	Float
freq	2	23	71	29	27	30	9	19	8	13	5	60	33	62	28	1

Understand the data - Investigate the variables**

Key Points to Investigate the Variables Variable Types:

Categorical Variables: These are variables that represent categories or groups. Examples in a cereals dataset might include name, manufacturer, and type. Numerical Variables: These are variables that represent quantities and can be discrete or continuous. Examples might include calories, protein, fat, sodium, fiber, carbohydrates, sugars, and potassium. Variable Descriptions:

Name: The name of the cereal. Manufacturer: The company that manufactures the cereal. Type: Type of cereal (e.g., cold or hot). Calories: Number of calories per serving. Protein: Amount of protein per serving (grams). Fat: Amount of fat per serving (grams). Sodium: Amount of sodium per serving (milligrams). Fiber: Amount of dietary fiber per serving (grams). Carbohydrates: Amount of carbohydrates per serving (grams). Sugars: Amount of sugar per serving (grams). Potassium: Amount of potassium per serving (milligrams). Vitamins and Minerals: Percent of daily recommended vitamins and minerals.

Summary Statistics:

Calculate basic summary statistics (mean, median, mode, standard deviation) for numerical variables. Identify the range, minimum, and maximum values. Understand the distribution of the data (e.g., are there any skewed distributions?).

Missing Values:

Check for missing values in the dataset. Decide on an approach to handle missing values (e.g., remove rows, impute with mean/median). Data Types and Formats:

Verify that each variable is stored in the correct data type (e.g., numerical variables as integers or floats, categorical variables as strings). Convert data types if necessary. Variable Relationships:

Investigate relationships between variables. For example, how does the amount of sugar relate to calories? Use correlation analysis to understand linear relationships between numerical variables. Visualize relationships using scatter plots, pair plots, or heatmaps. Distribution Analysis:

Plot histograms or density plots for numerical variables to understand their distributions. Identify outliers or unusual values in the data.

In [11]:
```
#check for all null valuales
pd.isnull(df).sum()
```

Out[11]:
```
name        0
mfr         3
type        3
calories    3
protein     3
fat         3
sodium      3
fiber       3
carbo       3
sugars      3
potass      3
vitamins    3
shelf       3
weight      3
cups        3
rating      3
dtype: int64
```

In [13]:
```
#drop all null values
#Data cleaning (if needed)
df.dropna(inplace=True)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['name', 'mfr', 'type', 'calories', 'protein', 'fat', 'sodium', 'fiber',
       'carbo', 'sugars', 'potass', 'vitamins', 'shelf', 'weight', 'cups',
       'rating'],
      dtype='object')
```

In [8]:
```
df.tail(5)
```

Out[8]:

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
73	Triples	G	C	110	2	1	250	0	21	3	60	25	3	1	0.75	39.106174
74	Trix	G	C	110	1	1	140	0	13	12	25	25	2	1	1	27.753301
75	Wheat Chex	R	C	100	3	1	230	3	17	3	115	25	1	1	0.67	49.787445
76	Wheaties	G	C	100	3	1	200	3	17	3	110	25	1	1	1	51.592193
77	Wheaties Honey Gold	G	C	110	2	1	200	1	16	8	60	25	1	1	0.75	36.187559

In [11]:
```
df.shape
```

Out[11]:
```
(78, 16)
```

In [12]:
```
df.duplicated().sum()
```

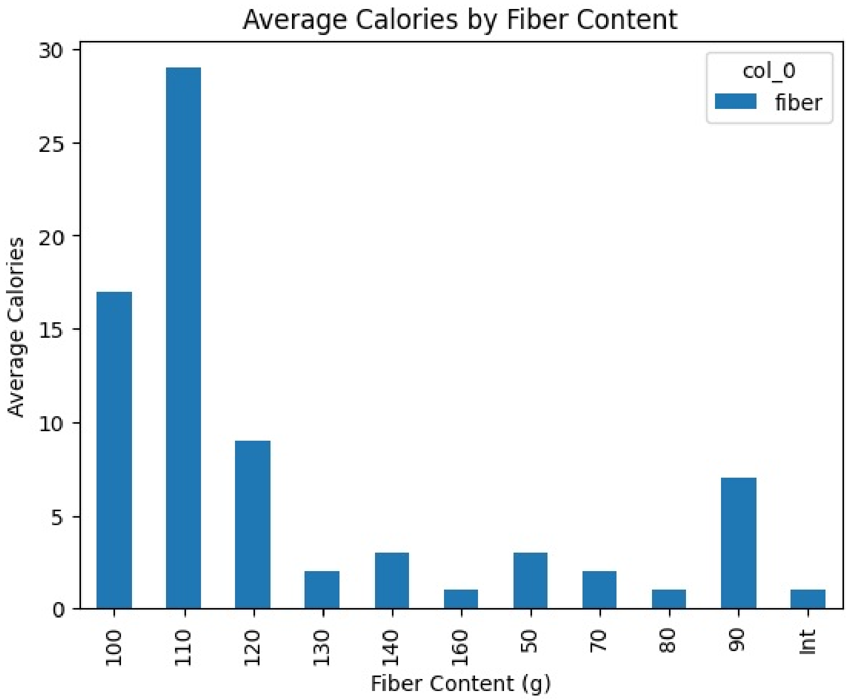
Out[12]:
```
1
```

In [26]:
```
df.nunique()
```

Out[26]:
```
name        77
mfr          8
type         3
calories    11
protein      7
fat          6
sodium      26
fiber       14
carbo       23
sugars      18
potass      35
vitamins     4
shelf        4
weight       8
cups        13
rating      75
dtype: int64
```

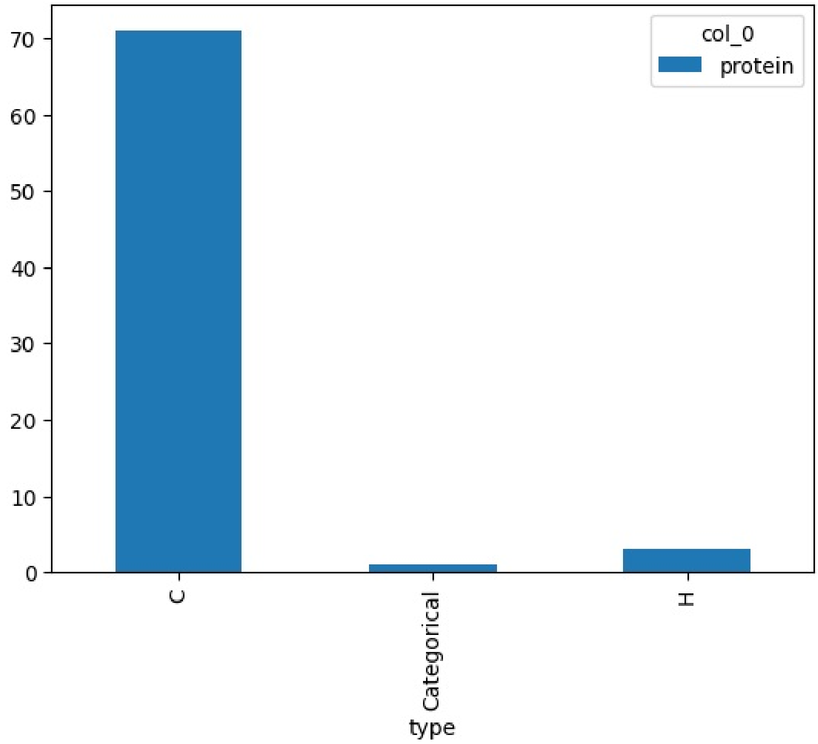
Distribution Analysis

In [49]:
```
pd.crosstab(df['calories'],['fiber']).plot(kind='bar',stacked='true')
plt.title('Average Calories by Fiber Content')
plt.xlabel('Fiber Content (g)')
plt.ylabel('Average Calories')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()
```



Rows (calories): Represents the unique values from the 'calories' column (100, 150, 200, 250, 300). Columns (fiber): Represents the unique values from the 'fiber' column ('high', 'low', 'medium'). Values: Indicates the frequency of occurrence of each combination. For example, there is 1 occurrence of 100 calories with 'low' fiber, 1 occurrence of 150 calories with 'low' fiber, and so forth.

In [13]:
```
pd.crosstab(df['type'],['protein']).plot(kind='bar',stacked='true')
plt.show()
```



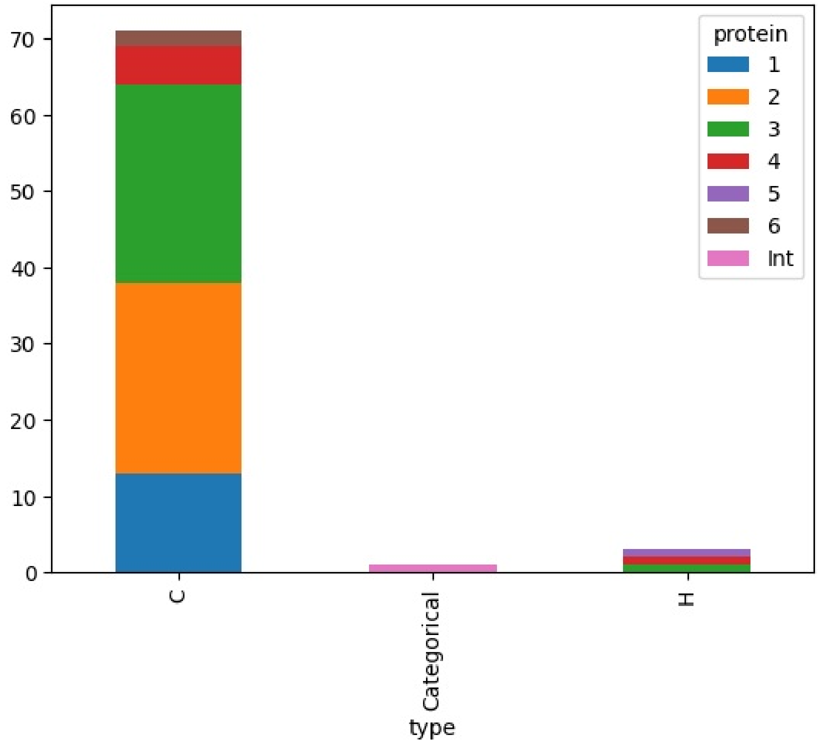
In [29]:
```
pd.crosstab(df['type'],df['protein'])
```

Out[29]:

protein	1	2	3	4	5	6	Int
type							
C	13	25	26	5	0	2	0
Categorical	0	0	0	0	0	0	1
H	0	0	1	1	1	0	0

Rows (type): Represents the unique values from the 'type' column ('meat', 'dairy', 'grains'). Columns (protein): Represents the unique values from the 'protein' column ('high', 'low', 'medium'). Values: Indicates the frequency of occurrence of each combination. For example, there is 1 occurrence of 'dairy' type with 'high' protein, 1 occurrence of 'grains' type with 'low' protein, and so forth. This crosstab allows you to quickly see how the distribution of protein levels varies across different types of food.

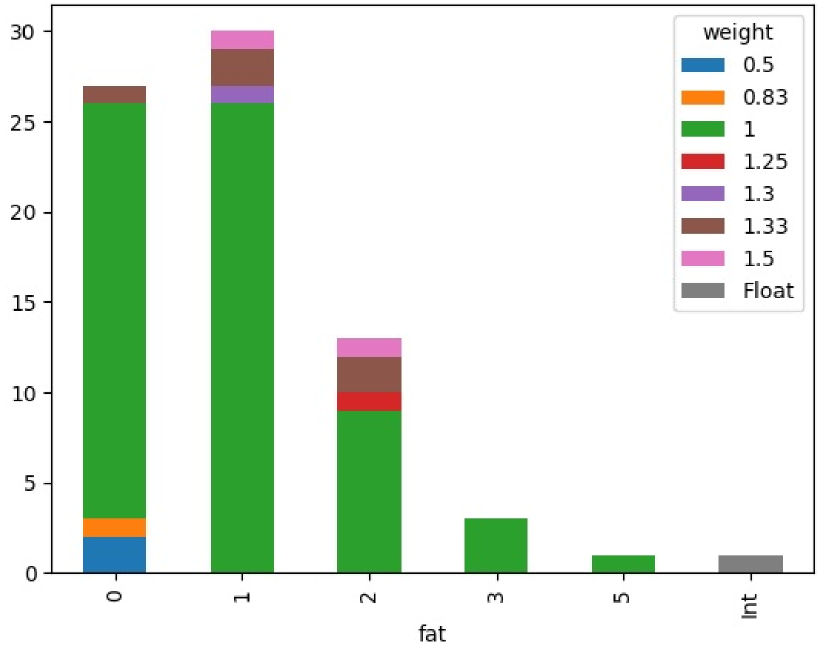
In [26]:
```
pd.crosstab(df['type'],df['protein']).plot(kind='bar',stacked='true')
plt.show()
```



In [52]:
```
plt.figure(figsize=(16,5))
pd.crosstab(df['fat'],df['weight']).plot(kind='bar',stacked='true')
```

```
plt.show()
```

```
<Figure size 1600x500 with 0 Axes>
```



Rows (fat): The unique values from the 'fat' column (high, low, medium). Columns (weight): The unique values from the 'weight' column (heavy, light, medium). Values: The frequency of each combination. For instance, there is 1 occurrence of 'high' fat and 'heavy' weight, 2 occurrences of 'low' fat and 'light' weight, and so on. This cross-tabulation helps in understanding the relationship between the two categorical variables 'fat' and 'weight' by showing how often each combination occurs in the dataset.

In [4]:
```
pd.crosstab(df['name'],['sugars'])
```

Out[4]:

col_0	sugars
name	
100% Bran	1
100% Natural Bran	1
All-Bran	1
All-Bran with Extra Fiber	1
Almond Delight	1
...	...
Triples	1
Trix	1
Wheat Chex	1
Wheaties	1
Wheaties Honey Gold	1

77 rows × 1 columns

Cereal Names: The rows represent different cereal names in the dataset. Sugar Content: The columns represent different levels of sugar content in the cereals. Frequency Counts: The values in the table indicate the count of each cereal name that corresponds to each level of sugar content.

In [28]:
```
pd.crosstab(df['name'],df['weight'])
```

Out[28]:

weight	0.5	0.83	1	1.25	1.3	1.33	1.5	Float
name								
100% Bran	0	0	1	0	0	0	0	0
100% Natural Bran	0	0	1	0	0	0	0	0
All-Bran	0	0	1	0	0	0	0	0
All-Bran with Extra Fiber	0	0	1	0	0	0	0	0
Almond Delight	0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...
Triples	0	0	1	0	0	0	0	0
Trix	0	0	1	0	0	0	0	0
Wheat Chex	0	0	1	0	0	0	0	0
Wheaties	0	0	1	0	0	0	0	0
Wheaties Honey Gold	0	0	1	0	0	0	0	0

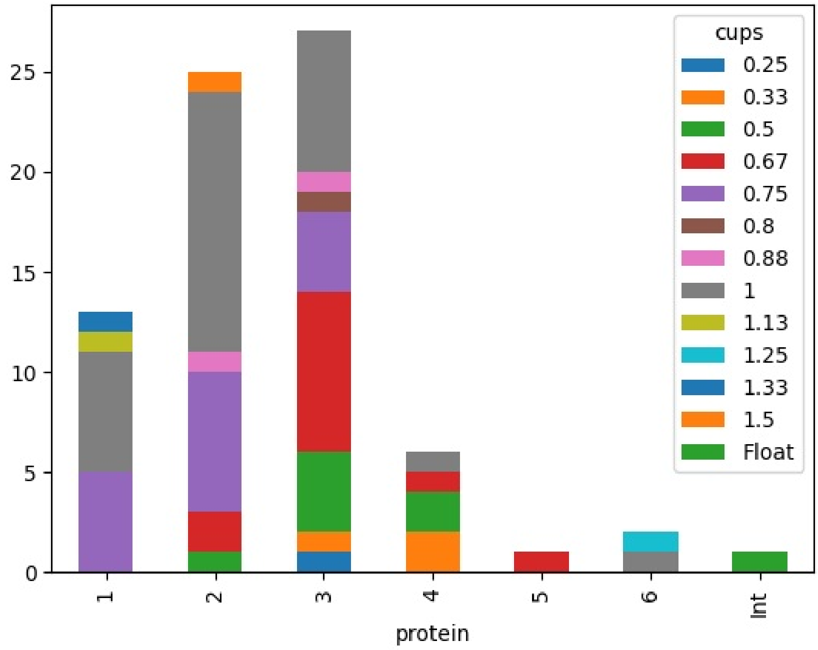
75 rows × 8 columns

Cereal Names: The rows represent the different names of cereals in the dataset. Weight Categories: The columns represent different weight categories of the cereal servings. Frequency Counts: The values in the table indicate how many times each cereal appears in each weight category.

In [48]:

```
plt.figure(figsize=(14,4))
pd.crosstab(df['protein'],df['cups']).plot(kind='bar',stacked='true')
plt.xticks(rotation=90)
plt.show()
```

<Figure size 1400x400 with 0 Axes>



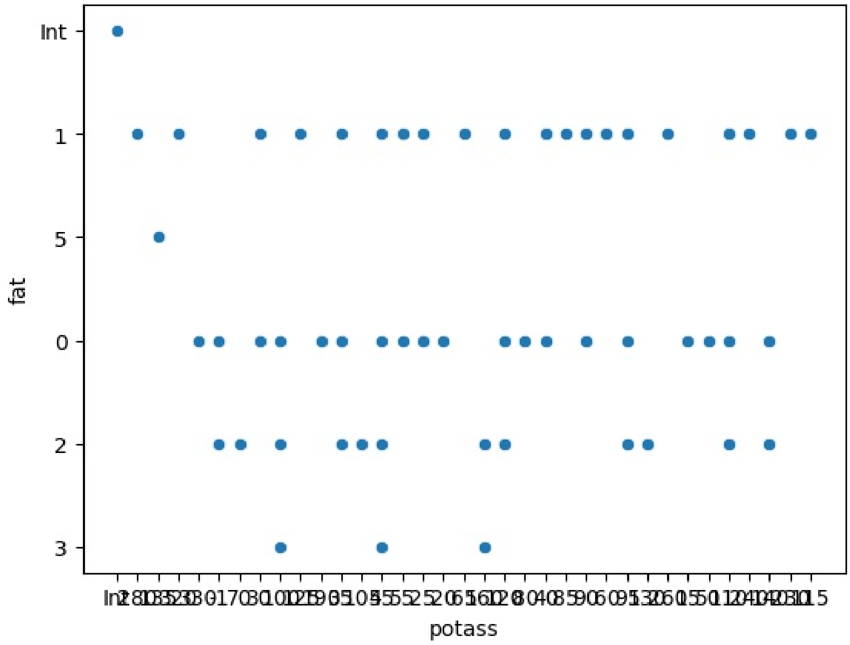
Analysis and Description Protein Levels: The x-axis represents the different levels of protein content in the cereals. Serving Size (Cups): The segments within each bar represent different serving sizes (measured in cups). Frequency: The height of each bar indicates the number of cereals that correspond to each combination of protein content and serving size.

Relationship Analysis

In [34]:

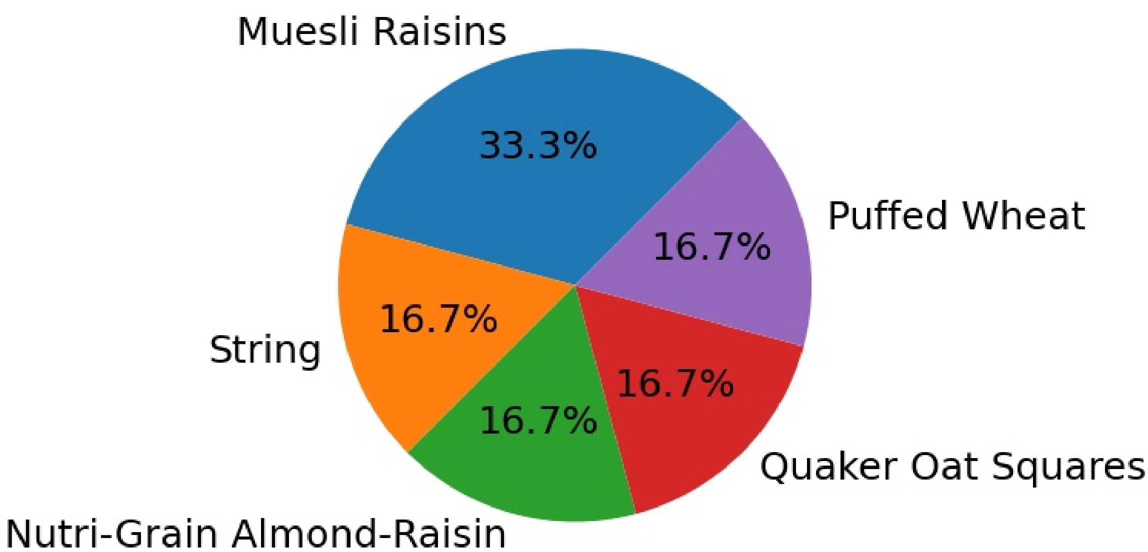
```
#Scatter plot of potass vs. fat
sns.scatterplot(x=df['potass'],y=df['fat'])
```

Out[34]: <Axes: xlabel='potass', ylabel='fat'>



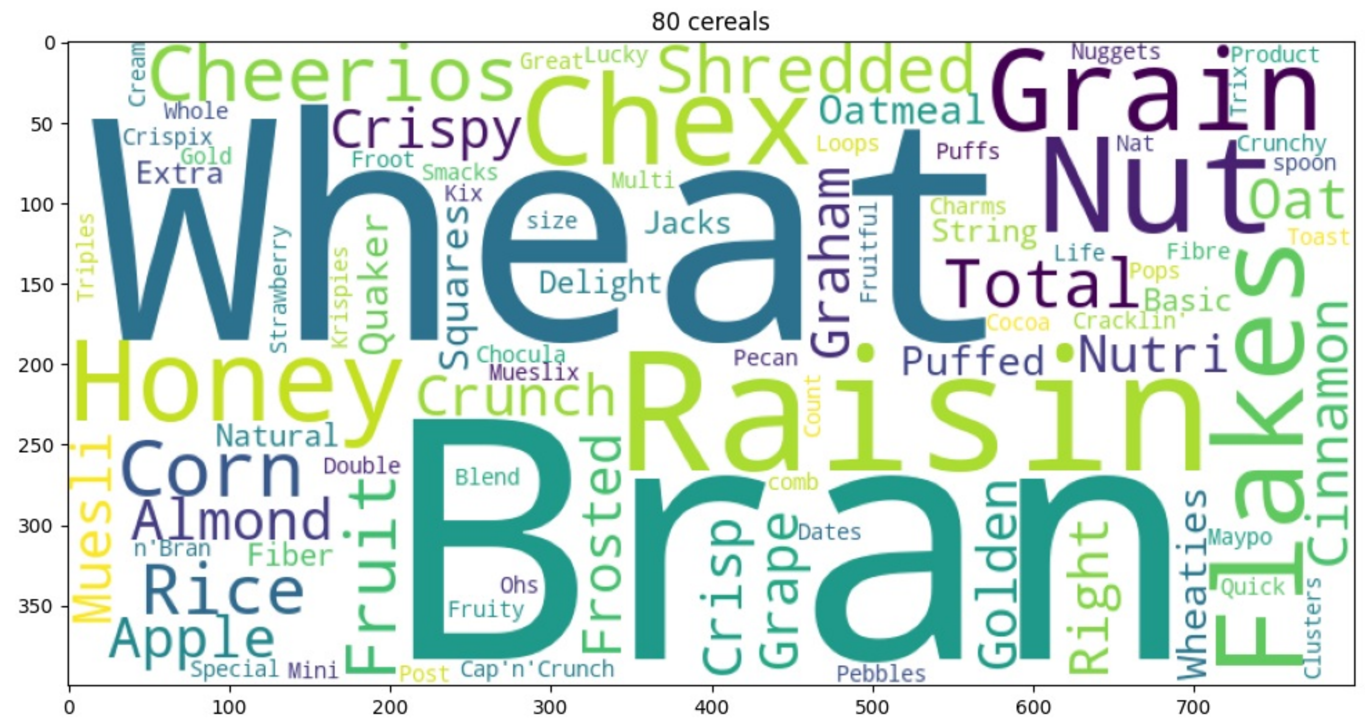
Analysis and Description X-Axis (Potassium): Represents the potassium content in the cereals. Y-Axis (Fat): Represents the fat content in the cereals. Data Points: Each point represents an individual cereal. The position of the point indicates the potassium and fat levels for that cereal.

In [53]:
```
plt.figure(figsize=(5,5))
d=(df['name'].value_counts(normalize=True)*100).head()
keys=df['name'].value_counts().head().index
colourz=['#B5DF00','#AD1FFF','#FFC93F','#5FB1FF','BF1B00']
exploda=(0.02,0.02,0.02,0.4,0.02)
plt.pie(d,labels=keys,autopct='%1.1f%%',startangle=45,textprops={'fontsize':18})
plt.savefig("audiencepie.png")
```



In [64]:
```
from wordcloud import WordCloud
```

In [72]:
```
all_review=' '.join(df['name'].dropna())
wordcloud=WordCloud(width=800, height=400, background_color='white').generate(all_review)
plt.figure(figsize=(12,8))
plt.imshow(wordcloud)
plt.title('80 cereals')
plt.show()
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js