Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menu bar, select Kernel→Restart) and then **run all cells** (in the menu bar, select Cell→Run All).

Make sure that in addition to the code, you provide written answers for all questions of the assignment.

Below, please fill in your name:

In [1]:
```
NAME = "Jesus Andres Correal Ortiz"
```

Assignment 1 - Data Cleaning & Visualization

(30 points total)

For this assignment, we will use an open dataset with data on various types of cereal and the corresponding customer ratings. Use the .csv file provided.

Create a dataframe by importing the file into Jupyter, and complete the tasks below. Be sure to show your Python code. You will not be graded on code efficiency, but your code should return a correct answer.

Dictionary

```
Content
Fields in the dataset:
•   Name: Name of cereal
•   mfr: Manufacturer of cereal
o   A = American Home Food Products;
o   G = General Mills
o   K = Kelloggs
o   N = Nabisco
o   P = Post
o   Q = Quaker Oats
o   R = Ralston Purina
•   type:
o   cold
o   hot
•   calories: calories per serving
•   protein: grams of protein
•   fat: grams of fat
•   sodium: milligrams of sodium
•   fiber: grams of dietary fiber
•   carbo: grams of complex carbohydrates
•   sugars: grams of sugars
•   potass: milligrams of potassium
•   vitamins: vitamins and minerals - 0, 25, or 100, indicating the
typical percentage of FDA recommended
```

- shelf: display shelf (1, 2, or 3, counting from the floor)
- weight: weight in ounces of one serving
- cups: number of cups in one serving
- rating: a rating of the cereals

Requirements

In [2]:

```
%pip install pandas==2.1.2
%pip install matplotlib==3.8.0
%pip install numpy==1.26.1
```

```
Collecting pandas==2.1.2
  Obtaining dependency information for pandas==2.1.2 from https://files.python
hosted.org/packages/4e/dd/4a77fb4cb7d207fbeb77dfc7c022131d295767504eabb5836fcd
63b644a1/pandas-2.1.2-cp311-cp311-macosx_11_0_arm64.whl.metadata
  Downloading pandas-2.1.2-cp311-cp311-macosx_11_0_arm64.whl.metadata (18 kB)
Requirement already satisfied: numpy<2,>=1.23.2 in /Applications/Anaconda/anac
onda3/lib/python3.11/site-packages (from pandas==2.1.2) (1.24.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /Applications/Anacond
a/anaconda3/lib/python3.11/site-packages (from pandas==2.1.2) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Applications/Anaconda/anaconda
3/lib/python3.11/site-packages (from pandas==2.1.2) (2022.7)
Collecting tzdata>=2022.1 (from pandas==2.1.2)
  Using cached tzdata-2023.3-py2.py3-none-any.whl (341 kB)
Requirement already satisfied: six>=1.5 in /Applications/Anaconda/anaconda3/li
b/python3.11/site-packages (from python-dateutil>=2.8.2->pandas==2.1.2) (1.16.
0)
Using cached pandas-2.1.2-cp311-cp311-macosx_11_0_arm64.whl (10.8 MB)
Installing collected packages: tzdata, pandas
  Attempting uninstall: pandas
    Found existing installation: pandas 1.5.3
    Uninstalling pandas-1.5.3:
      Successfully uninstalled pandas-1.5.3
Successfully installed pandas-2.1.2 tzdata-2023.3
Note: you may need to restart the kernel to use updated packages.
Collecting matplotlib==3.8.0
  Obtaining dependency information for matplotlib==3.8.0 from https://files.py
thonhosted.org/packages/af/f3/fb27b3b902fc759bbca3f9d0336c48069c3022e57552c4b0
095d997c7ea8/matplotlib-3.8.0-cp311-cp311-macosx_11_0_arm64.whl.metadata
  Downloading matplotlib-3.8.0-cp311-cp311-macosx_11_0_arm64.whl.metadata (5.8
kB)
Requirement already satisfied: contourpy>=1.0.1 in /Applications/Anaconda/anac
onda3/lib/python3.11/site-packages (from matplotlib==3.8.0) (1.0.5)
Requirement already satisfied: cycler>=0.10 in /Applications/Anaconda/anaconda
3/lib/python3.11/site-packages (from matplotlib==3.8.0) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Applications/Anaconda/ana
conda3/lib/python3.11/site-packages (from matplotlib==3.8.0) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /Applications/Anaconda/ana
conda3/lib/python3.11/site-packages (from matplotlib==3.8.0) (1.4.4)
Requirement already satisfied: numpy<2,>=1.21 in /Applications/Anaconda/anacon
da3/lib/python3.11/site-packages (from matplotlib==3.8.0) (1.24.3)
Requirement already satisfied: packaging>=20.0 in /Applications/Anaconda/anaco
nda3/lib/python3.11/site-packages (from matplotlib==3.8.0) (23.0)
Requirement already satisfied: pillow>=6.2.0 in /Applications/Anaconda/anacond
a3/lib/python3.11/site-packages (from matplotlib==3.8.0) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Applications/Anaconda/anac
onda3/lib/python3.11/site-packages (from matplotlib==3.8.0) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /Applications/Anaconda/
anaconda3/lib/python3.11/site-packages (from matplotlib==3.8.0) (2.8.2)
Requirement already satisfied: six>=1.5 in /Applications/Anaconda/anaconda3/li
b/python3.11/site-packages (from python-dateutil>=2.7->matplotlib==3.8.0) (1.1
6.0)
Using cached matplotlib-3.8.0-cp311-cp311-macosx_11_0_arm64.whl (7.5 MB)
Installing collected packages: matplotlib
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.7.1
    Uninstalling matplotlib-3.7.1:
      Successfully uninstalled matplotlib-3.7.1
Successfully installed matplotlib-3.8.0
Note: you may need to restart the kernel to use updated packages.
Collecting numpy==1.26.1
```

```
  Obtaining dependency information for numpy==1.26.1 from https://files.python
hosted.org/packages/e8/06/0512e2582fd27bb7b358fa1e4ffc0f6c89c89f5ada31df58c5fa
93171098/numpy-1.26.1-cp311-cp311-macosx_11_0_arm64.whl.metadata
  Downloading numpy-1.26.1-cp311-cp311-macosx_11_0_arm64.whl.metadata (115 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.1/115.1 kB 3.6 MB/s eta 0:00:
00
Using cached numpy-1.26.1-cp311-cp311-macosx_11_0_arm64.whl (14.0 MB)
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.24.3
    Uninstalling numpy-1.24.3:
      Successfully uninstalled numpy-1.24.3
ERROR: pip's dependency resolver does not currently take into account all the
packages that are installed. This behaviour is the source of the following dep
endency conflicts.
tables 3.8.0 requires blosc2~=2.0.0, which is not installed.
tables 3.8.0 requires cython>=0.29.21, which is not installed.
gensim 4.3.0 requires FuzzyTM>=0.4.0, which is not installed.
numba 0.57.0 requires numpy<1.25,>=1.21, but you have numpy 1.26.1 which is in
compatible.
tensorflow-macos 2.13.0 requires numpy<=1.24.3,>=1.22, but you have numpy 1.2
6.1 which is incompatible.
Successfully installed numpy-1.26.1
Note: you may need to restart the kernel to use updated packages.
```

Packages

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

pd.options.display.max_columns = None
pd.options.display.max_rows = None
%matplotlib inline
```

Read the file

In [4]:
```
df = pd.read_csv('./cereal.csv')
df.head()
```

Out[4]:

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

In [5]:

```
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 [6]:

```
# Chek for null values
df.isna().sum().sum()
```

Out[6]:

```
0
```

Questions

Question 1. *(4 points)*

- Create a new 'Type of Cereal' column in your dataframe (1 point) by copying the 'name' column. Write a function to replace the names of the cereal in your new column with one of these categories Bran, Wheat, Fiber, Protein, Crunch, Corn, Nut, Rice and Other (3 points). Hint: the function should look through the text in the cereal name and determine, based on its contents, how to categorize the cereal type.

In [7]:

```
# Copy the column 'name' to create the new column 'Type of Cereal'
df['Type of Cereal'] = df['name'].copy(deep=True)
df.head()
```

Out[7]:

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

In [8]:

```
def replace_cereal_type():
    """
    Replace value of 'Type of Cereal' column base in its name
    """

    # Create a list of categories and replace the values of the cell using regu
    catetories = ['Bran', 'Wheat', 'Fiber', 'Protein', 'Crunch', 'Corn', 'Nut'
    for category in catetories:
        df['Type of Cereal'].replace(regex=fr'.*{category}.*', value=category,

    # Replace the value for 'Other' if the value is not in a category.
    df['Type of Cereal'].replace(regex=r'^(?!Bran|Wheat|Fiber|Protein|Crunch|Co

replace_cereal_type()
```

In [9]:

```
# Check the unique values of the column Type of Cereal
df['Type of Cereal'].unique()
```

Out[9]:

```
array(['Bran', 'Other', 'Crunch', 'Corn', 'Wheat', 'Nut', 'Rice'],
      dtype=object)
```

Question 2. *(2 points)*

- Identify the negative values in the data set and replace them with the median value for that column.

In [10]:

```
# Get the min value of the columns to identify if they have negative values
df.min()
```

Out[10]:

```
name              100% Bran
mfr                       A
type                      C
calories                 50
protein                   1
fat                       0
sodium                    0
fiber                   0.0
carbo                  -1.0
sugars                   -1
potass                   -1
vitamins                  0
shelf                     1
weight                  0.5
cups                   0.25
rating            18.042851
Type of Cereal         Bran
dtype: object
```

The dataset shows negative values in the columns: carbo, sugars, potass

In [11]:

```python
def replace_negative_values(df, column_name):
    """
    Replace nevative values for the median
    ...
    Arguments
        df: DataFramw to process
        column_name: Column name
    """

    # Filter items with negative values
    filter = df[column_name] < 0
    count = len(df[filter])

    # Get the median for the column and replace values
    median = df[column_name].median()
    df.loc[filter, column_name] = median

    print(f"— Replacing {count} values on '{column_name}' column. Median = {m

# Get numeric columns
negative_columns = ['carbo', 'sugars', 'potass']
for column_name in negative_columns:
    replace_negative_values(df, column_name)
```

```
— Replacing 1 values on 'carbo' column. Median = 14.0
— Replacing 1 values on 'sugars' column. Median = 7.0
— Replacing 2 values on 'potass' column. Median = 90.0
```

In [12]:

```python
# Get the mean value of the columns to validate nevative values again
df[negative_columns].min()
```

Out[12]:

```
carbo      5.0
sugars     0.0
potass    15.0
dtype: float64
```

Now, there are no negative values in the columns.

Question 3. *(5 points)*

- Standardize the 'weight' column to 1. For this question, you will need to write a function to divide the remaining columns which contain nutritional information by the corresponding value in the weight column, and you will need to divide the value in the weight column by itself in order to get 1. For example, if an observation has a weight value of 1.33 and a calories value of 250, if you divide 250/1.33 you should get a calories value of 188 and a weight value of 1.

In [13]:

```
def standarize_values():
    """
    Standarize the values of the columns base on the 'weight' column
    """
    df.loc[:, 'calories': 'weight'] = df.loc[:, 'calories': 'weight'].div(df['

standarize_values()
df.head(10)
```

Out[13]:

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo
0	100% Bran	N	C	70.000000	4.000000	1.000000	130.000000	10.000000	5.000000
1	100% Natural Bran	Q	C	120.000000	3.000000	5.000000	15.000000	2.000000	8.000000
2	All-Bran	K	C	70.000000	4.000000	1.000000	260.000000	9.000000	7.000000
3	All-Bran with Extra Fiber	K	C	50.000000	4.000000	0.000000	140.000000	14.000000	8.000000
4	Almond Delight	R	C	110.000000	2.000000	2.000000	200.000000	1.000000	14.000000
5	Apple Cinnamon Cheerios	G	C	110.000000	2.000000	2.000000	180.000000	1.500000	10.500000
6	Apple Jacks	K	C	110.000000	2.000000	0.000000	125.000000	1.000000	11.000000
7	Basic 4	G	C	97.744361	2.255639	1.503759	157.894737	1.503759	13.533835
8	Bran Chex	R	C	90.000000	2.000000	1.000000	200.000000	4.000000	15.000000
9	Bran Flakes	P	C	90.000000	3.000000	0.000000	210.000000	5.000000	13.000000

Question 4. *(5 points)*

- Create a new column to categorize cereals as 'healthy' vs. 'unhealthy'. You can define your own version of healthy vs. unhealthy, or you can use the following: healthy cereals can be defined as those which have low calories (<100), low sodium (<150), low sugar (<9) high fiber (>3), and high protein (>2). All other cereals are unhealthy.

In [14]:
```python
def categorize_cereals():
    """
    This function categorizes dataset rows into 'healthy' vs. 'unhealthy' base
    """
    max_calories = 100 # 100
    max_sodium = 150 # 150
    max_sugar = 9 # 9
    min_fiber = 3 # 3
    min_protein = 2 # 2

    filter = (df['calories'] < max_calories) & (df['sodium'] < max_sodium) & (
    df['is_healthy'] = np.where(filter, 'healthy', 'unhealthy')

    # Get 3 healthy and 3 unhealthy items (Only for visualization)
    df.groupby(['is_healthy'], group_keys=False).apply(lambda x: x.sample(3))

categorize_cereals()
```

In [15]:
```python
df.head()
```

Out[15]:

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	sl
0	100% Bran	N	C	70.0	4.0	1.0	130.0	10.0	5.0	6.0	280.0	25.0	
1	100% Natural Bran	Q	C	120.0	3.0	5.0	15.0	2.0	8.0	8.0	135.0	0.0	
2	All-Bran	K	C	70.0	4.0	1.0	260.0	9.0	7.0	5.0	320.0	25.0	
3	All-Bran with Extra Fiber	K	C	50.0	4.0	0.0	140.0	14.0	8.0	0.0	330.0	25.0	
4	Almond Delight	R	C	110.0	2.0	2.0	200.0	1.0	14.0	8.0	90.0	25.0	

Question 5. *(2 points)*

- Based on your newly prepared data set, identify what % of cereals that each manufacturer produces are healthy.

In [16]:
```python
# List the manufacturer
df['mfr'].unique()
```

Out[16]:
```
array(['N', 'Q', 'K', 'R', 'G', 'P', 'A'], dtype=object)
```

In [17]:
```python
def calculate_healthy_percentage_q5() -> pd.DataFrame:
    """
    Calculates the percentage of healthy cereals produced by the manufacturer a
    new DataFrame with the name of the manufacturer and the percentage sorted i
    """
    # Group rows by on Manufacturer and is_healthy
    group = df[['mfr', 'is_healthy']].groupby(['mfr'])
    result_df = group['is_healthy'].apply(lambda x: (x == 'healthy').sum()) / g
    return result_df.rename('%').reset_index()

question5_df = calculate_healthy_percentage_q5()
question5_df.sort_values(by='%',ascending=False)
```

Out[17]:

	mfr	%
3	N	0.500000
4	P	0.111111
2	K	0.043478
0	A	0.000000
1	G	0.000000
5	Q	0.000000
6	R	0.000000

Question 6. *(2 points)*

- Calculate the average, minimum and maximum ratings for healthy vs. unhealthy cereals.

In [18]:

```
def calculate_average_q6(input_df) -> pd.DataFrame:
    """
    Calculate the average, minimum and maximum ratings for healthy vs. unhealt
    """
    result = input_df.groupby('is_healthy')['rating'].agg(['mean', 'max', 'min
    return result

question6_df = calculate_average_q6(df)
question6_df
```

Out[18]:

	is_healthy	mean	max	min
0	healthy	69.146753	93.704912	40.917047
1	unhealthy	40.826743	72.801787	18.042851

Question 7. *(2 points)*

- Calculate the average, minimum and maximum ratings for each type of cereal: Bran, Wheat, Fiber, Protein, Crunch, Corn, Nut, Rice and Other.

In [19]:

```
def calculate_ratings_q7(input_df):
    """
    Calculate the average, minimum and maximum ratings for each type of cereal
    """
    result = input_df.groupby('Type of Cereal')['rating'].agg(['mean', 'max',
    return result

question7_df = calculate_ratings_q7(df)
question7_df
```

Out[19]:

	Type of Cereal	mean	max	min
0	Bran	50.714179	93.704912	28.592785
1	Corn	40.482720	45.863324	35.782791
2	Crunch	26.078598	36.523683	18.042851
3	Nut	42.736791	53.371007	31.072217
4	Other	37.379947	55.333142	21.871292
5	Rice	47.771735	60.756112	40.560159
6	Wheat	56.333863	72.801787	36.176196

Question 8. *(3 points)*

- Create a stacked bar chart which shows how many of each type of cereal each manufacturer produces.

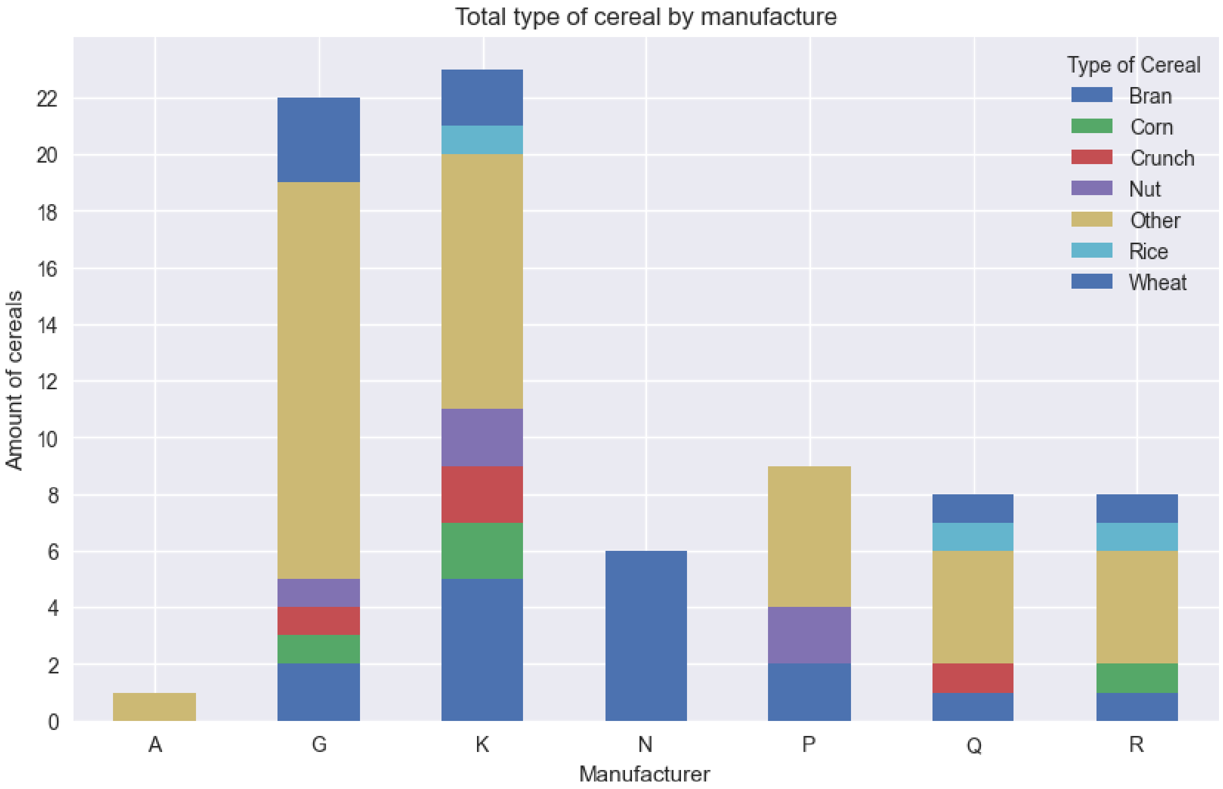
In [20]:

```python
# Group rows by manufacturer
mfrgroup_df = df.groupby('mfr')['Type of Cereal'].value_counts().reset_index()
mfrgroup_df

pivot_df = mfrgroup_df.pivot(index='mfr', columns='Type of Cereal', values='cou

# Set the theme
plt.style.use('seaborn-v0_8')

# Create graph
pivot_df.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.xlabel('Manufacturer')
plt.ylabel('Amount of cereals')
plt.title('Total type of cereal by manufacture')
plt.xticks(rotation=0)
plt.yticks(np.arange(00, 24, 2))
plt.show()
```



The chart shows the distribution of cereals by manufacturer. K has the highest productions and then G. A is the lower productor.

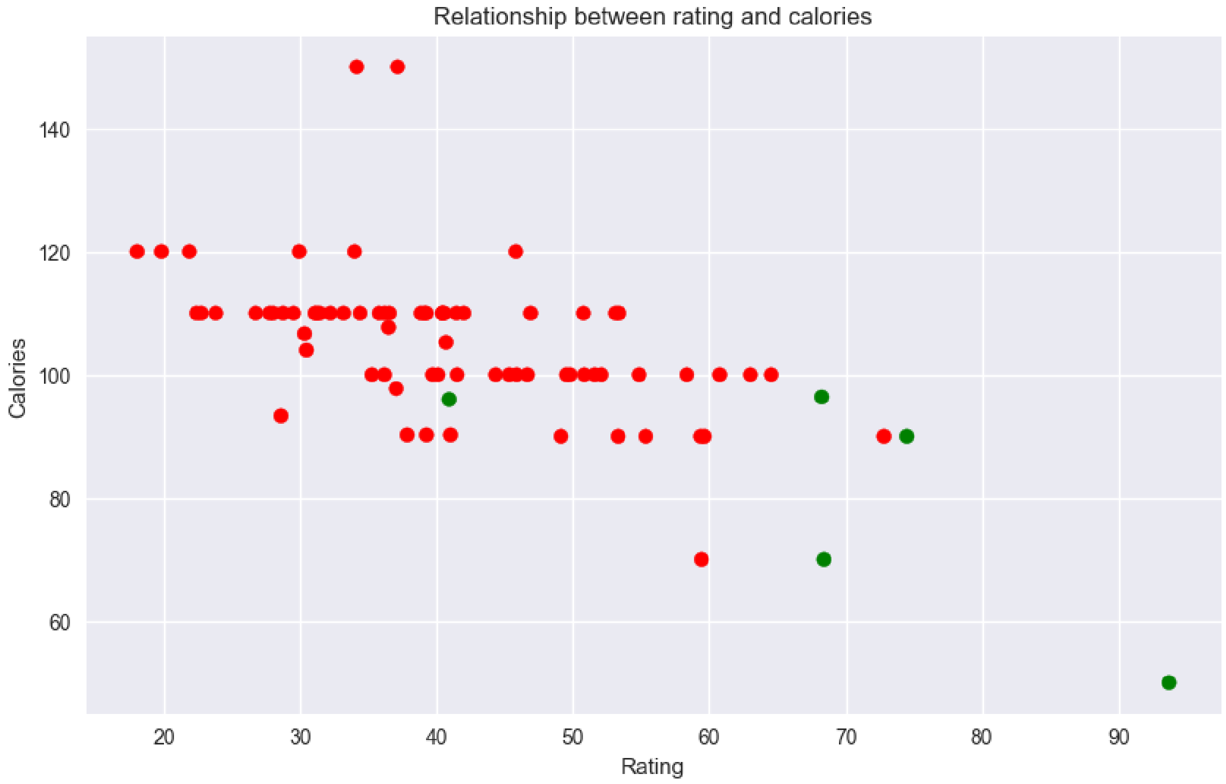
Question 9. *(3 points)*

- Create a 3-dimensional scatterplot which shows the relationship between rating and calories; the 3-rd dimension should be reflected in the color of the dots and should highlight whether the cereal is categorized as healthy or unhealthy.

In [21]:

```
corr_df = df['is_healthy'].apply(lambda x: 'g' if x == 'healthy' else 'r')

plt.figure(figsize=(10, 6))
plt.scatter(x=df['rating'], y=df['calories'], c=corr_df)
plt.xlabel('Rating')
plt.ylabel('Calories')
plt.title("Relationship between rating and calories")
plt.show();
```



Question 10. *(1 point)*

- Which shelf has the most healthy cereals?

In [22]:
```
df[df['is_healthy'] == 'healthy'].groupby('shelf')['is_healthy'].value_counts(
```

Out[22]:

	shelf	is_healthy	count
3	3.000000	healthy	2
0	1.000000	healthy	1
1	1.204819	healthy	1
2	2.400000	healthy	1

The shelf with the most healthy products is 3.000000

Question 11. *(1 point)*

- Based on the analysis conducted, what can you conclude about the cereal data set?

Conclussion

1. The dataset contains 77 rows and 16 columns, 8 int64 columns, 5 float columns and 3 string columns; it does not have any missing values, and the products were categorized by type of cereal, and it contains seven categories: Bran, Other, Crunch, Corn, Wheat, Nut and Rice.

2. The columns carbo, sugars and potass have negative values, and their values were replaced with the mean of each column because they are incorrect values.

3. Manufacturer N has the healthiest products among the companies, with 50% of healthy products, followed by P with 11% and K with 4%.

4. Healthy products have a higher rating than unhealthy products, with a maximum rating of 69.146753 against 40.826743.

5. The stacked bar chart shows the distribution of cereals by manufacturer. K has the highest production, and then G. A is the lower producer.

6. Finally, the scatter plot shows that only five products are healthy based on the thresholds.